**汇编语言与逆向技术实验报告**

**Lab4-Peviewer**

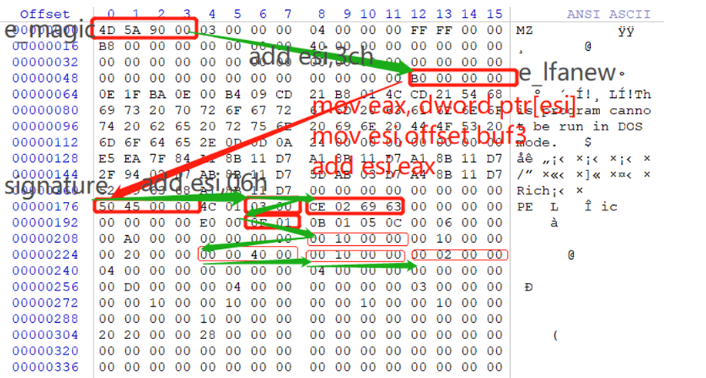
**学号：2110957 姓名：蒋薇 专业： 计算机科学与技术**

1. **实验内容**
   * 1. 窗口输出“HelloWorld!”字符串的汇编程序。输入PE文件的文件名，peviewer程序调用Windows API函数，打开指定的PE文件；
     2. 从文件的头部开始，读取IMAGE\_DOS\_HEADER结构中的e\_magic和e\_lfanew字段的值，按照实验演示的方式输出到命令行窗口；
     3. 继续读取PE文件的IMAGE\_NT\_HEADER结构中的Signature字段的值，按照实验演示的方式输出到命令行窗口；
     4. 继续读取IMAGE\_NT\_HEADER结构中的IMAGE\_FILE\_HEADER结构，从中读取出字段NumberOfSections、TimeDateStamp、Characteristics的值，按照实验演示的方式输出到命令行窗口；
     5. 继续读取IMAGE\_NT\_HEADER结构中的IMAGE\_OPTIONAL\_HEADER结构，从中读取字段AddressOfEntryPoint、ImageBase、SectionAlignment、FileAlignment的值，按照实验演示的方式输出到命令行窗口；
2. **实验步骤**
   * 1. peviewer程序的设计说明和控制流图

利用Windows API函数CreateFile、SetFilePointer、ReadFile、CloseHandle；

需要读取的内容是通过指针偏移量来定位的，确定好指针的起始位置，是Dos部首，还是PE文件头

文件头的信息 pe文件节表的信息 pe文件数据目录表的信息 从RVA到Frva的计算



* + 1. peviewer.asm的源代码和注释

.386

.model flat,stdcall

option casemap :none

include \masm32\include\windows.inc

include \masm32\include\masm32.inc

include \masm32\macros\macros.asm

include \masm32\include\kernel32.inc

includelib \masm32\lib\masm32.lib

includelib \masm32\lib\kernel32.lib

.data

hi BYTE "Please input a PE file:",0;输入pe文件的语句

filename BYTE 20 DUP(0);将要输入的文件名输入进这里头

hfile DWORD 0;文件句柄

buf3 DWORD 4000 dup(?);从文件中接收数据的容器

buf4 DWORD 4000 dup(0);

buf5 word 4000 dup(0)

;以下是一些元素名称

IDH BYTE "IMAGE\_DOS\_HEADER",0

em BYTE " e\_magic: ",0

el BYTE " e\_lfanew: ",0

INH BYTE "IMAGE\_NT\_HEADERS",0

sig BYTE " Signature: ",0

IFH BYTE "IMAGE\_FILE\_HEADERS",0

nos BYTE " NumberOfSections: ",0

tds BYTE " TimeDateStamp: ",0

chc BYTE " Characteics: ",0

IOH BYTE "IMAGE\_OPTIONAL\_HEADER",0

aop BYTE " AddressOfEntryPoint: ",0

ib BYTE " ImageBase: ",0

sa BYTE " SectionAligment: ",0

fa BYTE " FILEAligment: ",0

endl BYTE 0Ah,0Dh,0;用于换行

.code

main PROC

invoke StdOut,addr hi

invoke StdIn, addr filename,20

;调用createfile程序

invoke CreateFile, addr filename,\

GENERIC\_READ,\

FILE\_SHARE\_READ,\

0,\

OPEN\_EXISTING,\

FILE\_ATTRIBUTE\_ARCHIVE,\

0

mov hfile, eax;将读取到的文件句柄传入hfile

;调用setfilepointer程序

invoke SetFilePointer, hfile,0 ,0,FILE\_BEGIN

;Indicates that the starting point is zero or the beginning of the file.

;调用readfile程序

;;;;invoke ReadFile, hfile, addr buf3, 4000, 0, 0

;文件柄，盛接读取到的数据，读取的字节大小，指向读取的字节数的指针，NULL

;此时 hfile在e\_magic的位置 MZ的那个位置

invoke ReadFile, hfile, addr buf3,400 , 0 ,0;buf3是文件的入口

mov esi, offset buf3

mov eax, dword PTR [esi]

invoke dw2hex, eax, addr buf4

mov ax,word PTR[buf4+4]

mov buf5,ax

invoke StdOut,addr IDH

invoke StdOut,addr endl

invoke StdOut,addr em

;invoke StdOut, addr buf4;读出mz

invoke StdOut,addr buf5

mov ax,word PTR[buf4+6]

mov buf5,ax

invoke StdOut,addr buf5

invoke StdOut, addr endl;换行

invoke StdOut, addr el

add esi,3ch;指到e\_lfanew

mov eax,dword PTR[esi]

invoke dw2hex,eax,addr buf4

invoke StdOut,addr buf4

invoke StdOut,addr endl

invoke StdOut, addr endl

invoke StdOut, addr INH

invoke StdOut, addr endl

invoke StdOut, addr sig

mov edx,dword PTR [esi]

mov esi,offset buf3

add esi, edx; e\_lfanew内存的是nt头相对于文件的偏移地址

mov eax, dword PTR[esi] ;此时esi指在signature

invoke dw2hex, eax,addr buf4

invoke StdOut, addr buf4

invoke StdOut, addr endl

invoke StdOut, addr endl

invoke StdOut, addr IFH

invoke StdOut, addr endl

invoke StdOut, addr nos

add esi,6h;移动到FileHeader

mov eax,dword PTR[esi]

invoke dw2hex, eax,addr buf4

mov ax,word PTR [buf4+4]

mov buf5,ax;leeorange13

invoke StdOut,addr buf5

mov ax,word PTR[buf4+6]

mov buf5,ax;leeorange13

invoke StdOut,addr buf5

invoke StdOut,addr endl

invoke StdOut,addr tds

add esi,2h;移动到了timedatestamp

mov eax,dword PTR[esi]

invoke dw2hex,eax, addr buf4

invoke StdOut,addr buf4

invoke StdOut, addr endl

add esi,0eh

invoke StdOut,addr chc

mov eax,dword ptr[esi]

invoke dw2hex, eax,addr buf4

mov ax,word ptr[buf4+4]

mov buf5,ax

invoke StdOut, addr buf5

mov ax,word ptr[buf4+6]

mov buf5,ax

invoke StdOut,addr buf5

invoke StdOut,addr endl

invoke StdOut, addr endl

invoke StdOut,addr IOH

add esi ,12h;leorange13

invoke StdOut, addr endl

invoke StdOut, addr aop

mov eax, dword ptr[esi]

invoke dw2hex, eax, addr buf4

invoke StdOut,addr buf4

invoke StdOut, addr endl

invoke StdOut,addr ib

add esi,4h;!!!!

add esi,4h;

add esi,4h

mov eax,dword PTR[esi]

invoke dw2hex,eax,addr buf4

invoke StdOut, addr buf4

invoke StdOut,addr endl

invoke StdOut,addr sa

add esi,4h

mov eax,dword PTR[esi]

invoke dw2hex,eax, addr buf4

invoke StdOut,addr buf4

invoke StdOut,addr endl

add esi,4h

mov eax,dword ptr[esi]

invoke StdOut,addr fa

mov eax,dword PTR[esi]

invoke dw2hex,eax,addr buf4

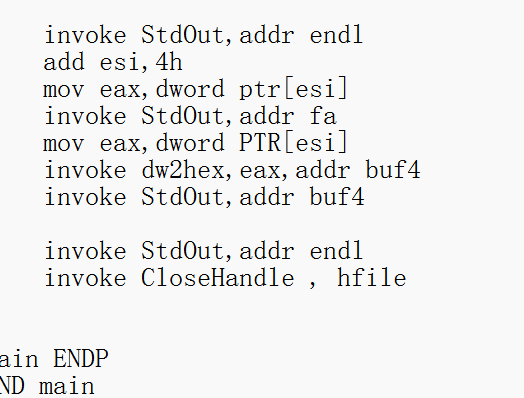
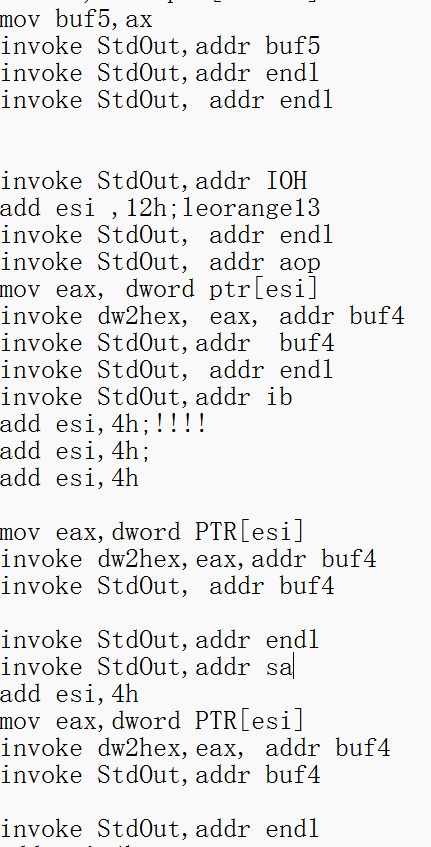
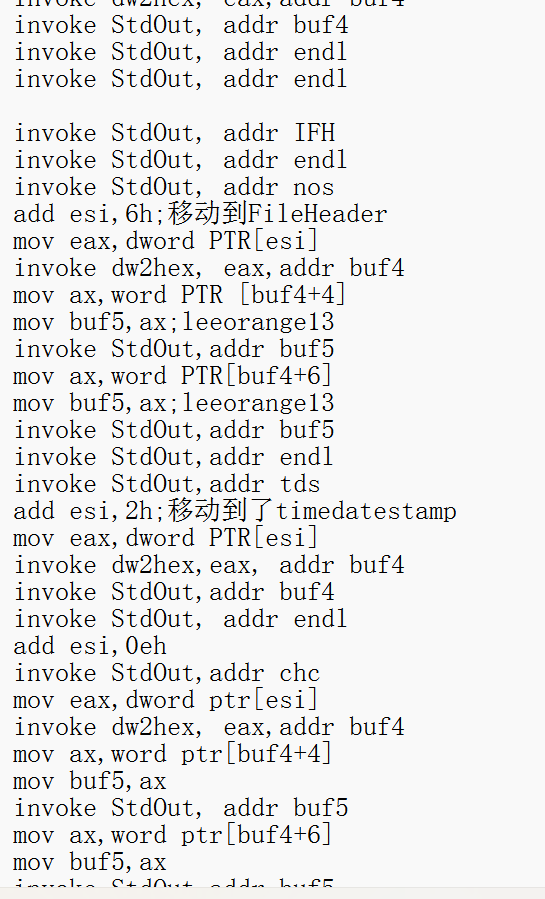
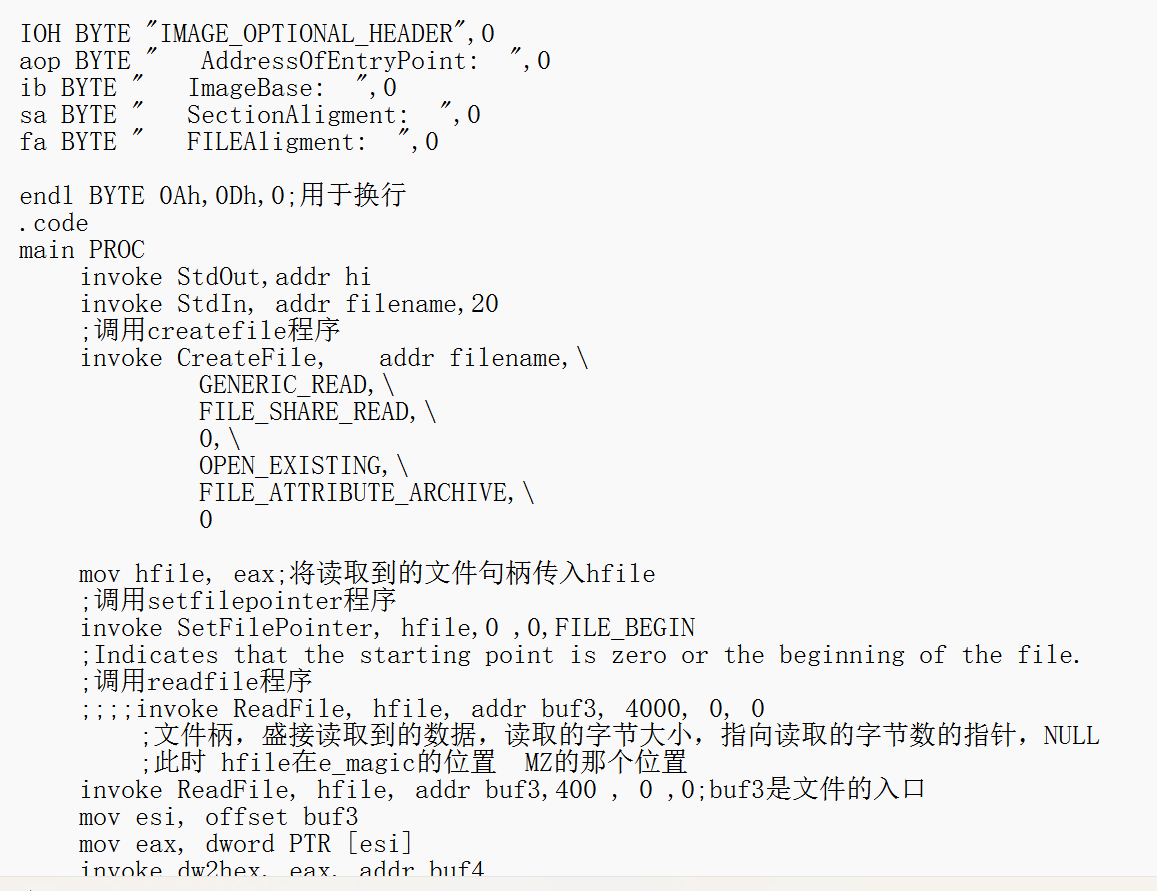
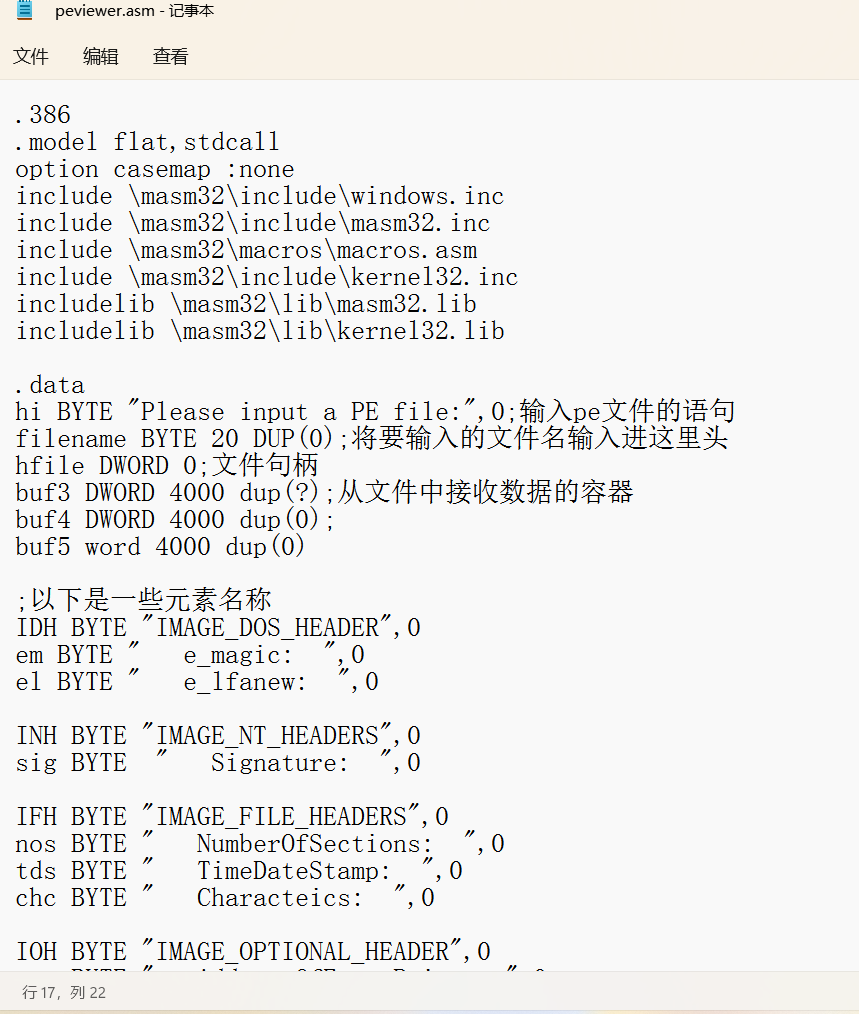
invoke StdOut,addr buf4

invoke StdOut,addr endl

invoke CloseHandle , hfile

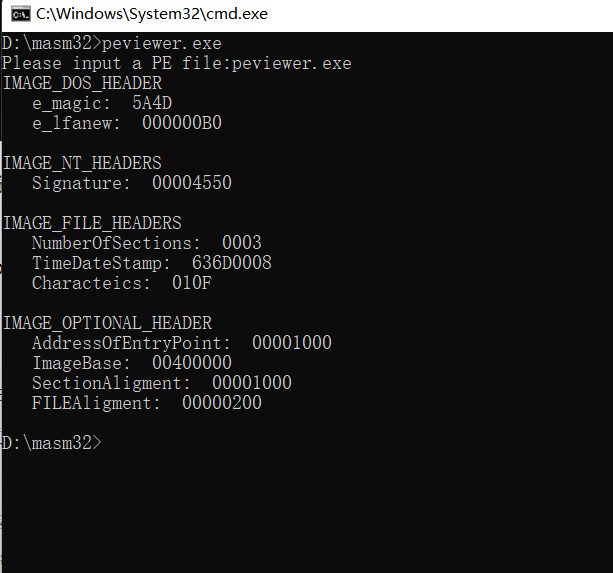
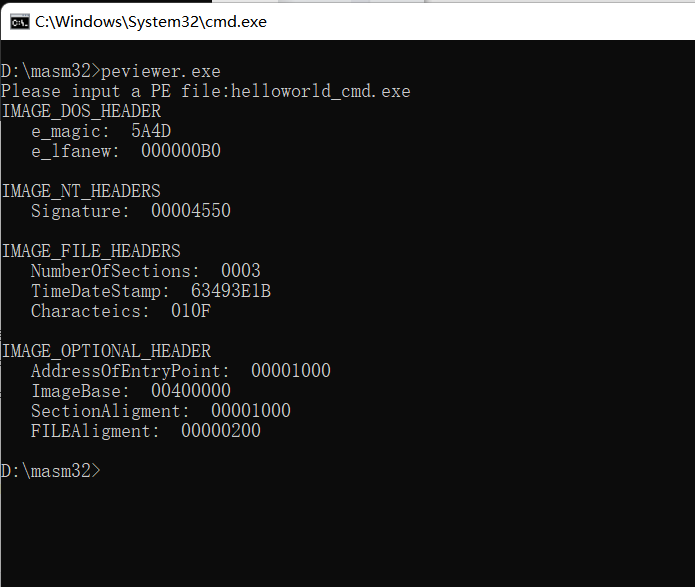
main ENDP

END main



* + 1. peviewer.exe运行截图





实验过程中的详细步骤，包括关键代码与截图等。

1. **实验心得**

PE文件：文件头(PE文件头，DOS文件头），节表（主要每个节的RVA地址，RVA地址是文件被装载到内存后数据相对于文件起始位置的偏移量），节区（属性相同的文件数据放在一起，可执行的，只读的放在一起）

* ****DOS文件头****

DOS头部分由MZ格式的文件头和可执行代码部分组成。而MZ格式的文件头是由一个IMAGE\_DOS\_HEADER结构定义的：

​

​

​

IMAE\_DOS\_HEADER STRUCT{ //DOS .EXE heade位置

e\_magic WORD ? //DOS可执行文件标记，为"MZ" 0x00

e\_cblp WORD ? //Bytes on last page of file 0x02

e\_cp WORD ? //Pages in file 0x04

e\_crlc WORD ? //Relocations 0x06

e\_cparhdr WORD ? //Size of header in paragraphs 0x08

e\_minalloc WORD ? //Minimum extra paragraphs needed 0x0A

e\_maxalloc WORD ? //Maximum extra paragraphs needed 0x0C

e\_ss WORD ? //DOS代码初始化堆栈段 0x0E

e\_sp WORD ? //DOS代码的初始化堆栈指针 0x10

e\_csum WORD ? //Checksum 0x12

e\_ip WORD ? //DOS代码入口IP 0x14

e\_cs WORD ? //DOS代码入口CS 0x16

e\_lfarlc WORD ? //File address of relocation table 0x18

e\_ovno WORD ? //Overlay number 0x1A

e\_res WORD 4 dup(?) //Reserved words 0x1C

e\_oemid WORD ? //OEM identifier (for e\_oeminfo) 0x24

e\_oeminfo WORD ? //OEM information; e\_oemid specific 0x26

e\_res2 WORD 10 dup(?) //Reserved words 0x28

e\_lfanew DWORD ? //指向PE文件头 0x3C

IMAGE\_DOS-HEADER ENDS

​

****PE文件头（NT文件头）****

NT文件头包括三个部分，分别是PE文件标识：Signature，FileHeader,OPtionalHeader。这三个结构都是在一个IMAGE\_NT\_HEADERS结构里面

IMAGE\_NT\_HEADERS STRUCT ;NT文件头

Signature DWORD ? ;PE文件标识

FileHeader IMAGE\_FILE\_HEADER <>

OptionalHeader IMAGE\_OPTIONAL\_HEADER32 <>

第一个字段就是用来标识PE文件的，第二个字段的结构IMAGE\_FILE\_HEADER包含了很多信息：

IMAGE\_FILE\_HEADER STRUCT

Machine WORD ? ;0004h - 运行平台

NumberOfSections WORD ? ;0006h - 文件的节区数目

TimeDateStamp DWORD ? ;0008h - 文件的创建日期和时间

PointerToSymbolTable DWORD ? ;000ch - 指向符号表（用于调试）

NumberOfSymbols DWORD ? ;0010h - 符号表中的符号数量（用于调试）

SizeOfOptionalHeader WORD ? ;0014h - IMAGE\_OPTIONAL\_HEADER32的结构长度

Characteristics WORD ? ;0016h - 文件属性

IMAGE\_FILE\_HEADER ENDS

​​

​

​

IMAE\_DOS\_HEADER STRUCT{ //DOS .EXE header 位置

e\_magic WORD ? //DOS可执行文件标记，为"MZ" 0x00

e\_cblp WORD ? //Bytes on last page of file 0x02

e\_cp WORD ? //Pages in file 0x04

e\_crlc WORD ? //Relocations 0x06

e\_cparhdr WORD ? //Size of header in paragraphs 0x08

e\_minalloc WORD ? //Minimum extra paragraphs needed 0x0A

e\_maxalloc WORD ? //Maximum extra paragraphs needed 0x0C

e\_ss WORD ? //DOS代码初始化堆栈段 0x0E

e\_sp WORD ? //DOS代码的初始化堆栈指针 0x10

e\_csum WORD ? //Checksum 0x12

e\_ip WORD ? //DOS代码入口IP 0x14

e\_cs WORD ? //DOS代码入口CS 0x16

e\_lfarlc WORD ? //File address of relocation table 0x18

e\_ovno WORD ? //Overlay number 0x1A

e\_res WORD 4 dup(?) //Reserved words 0x1C

e\_oemid WORD ? //OEM identifier (for e\_oeminfo) 0x24

e\_oeminfo WORD ? //OEM information; e\_oemid specific 0x26

e\_res2 WORD 10 dup(?) //Reserved words 0x28

e\_lfanew DWORD ? //指向PE文件头 0x3C

IMAGE\_DOS-HEADER ENDS

* ****节表****

IMAGE\_SECTION\_HEADER STRUCT

Namel db IMAGE\_SIZEOF\_SHORT\_NAME dup(?) ;8个字节的节区名称

union Misc

PhysicalAddress dd ?

VirtualSize dd ? ;节区的尺寸

ends

VirtualAddress dd ? ;节区的RVA地址

SizeOfRawData dd ? ;在文件中对齐后的尺寸

PointerToRawData dd ? ;在文件中的偏移

PointerToRelocations dd ? ;在OBJ文件中使用

PointerToLinenumbers dd ? ;行号表的位置（调试用的）

NumberOfRelocations dw ? ;在OBJ文件中使用

Characteristics dd ? ;节的属性

IMAGE\_SECTION\_HEADER ENDS