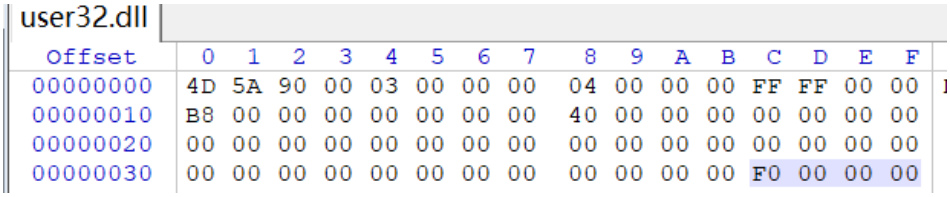
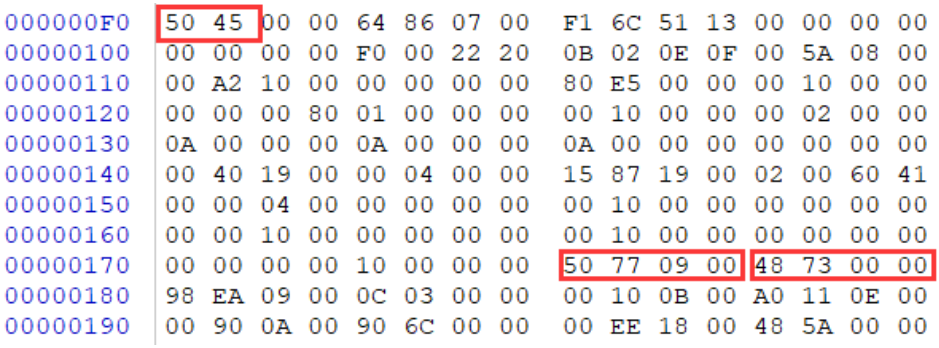
PE 文件结构解析作业

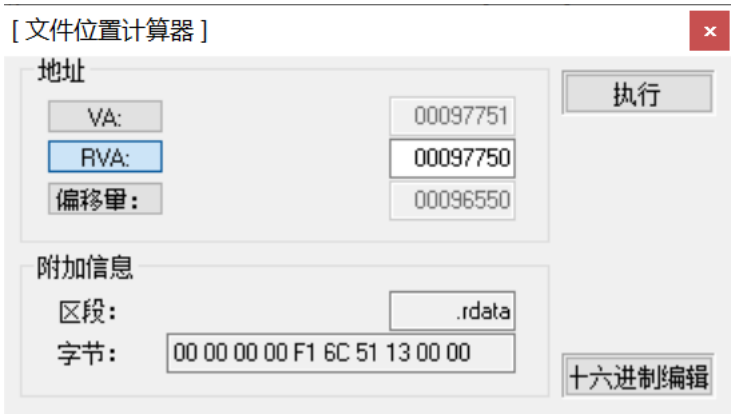
1. 请解析 USER32.DLL 前 5 个导出函数的信息，要求列举 AddressOfNames、AddressOfOrdinals、AddressOfFunctions 的详细数据。



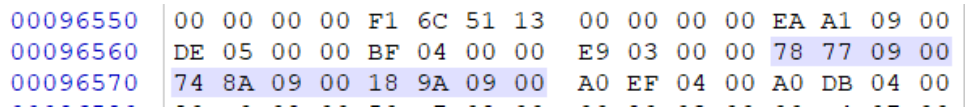
首先找到 pe 头偏移量：F0h



+78h 处开始为 DataDirectory[16]，其中输出表 RVA 为 97750h，大小为 7348h



算出偏移量为 96550h，根据输出表结构，有



	RVA	Offset
AddressOfFunctions	97778h	96578h
AddressOfNames	98A74h	97874h
AddressOfOrdinals	99A18h	98818h

adofFunctions

user32.dll

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00096570	74	8A	09	00	18	9A	09	00	A0	EF	04	00	A0	DB	04	00
00096580	80	B0	02	00	50	B7	02	00	00	20	03	00	00	D4	07	00
00096590	20	8A	01	00	40	B6	00	00	D0	3C	08	00	70	D4	07	00
000965A0	40	A2	02	00	70	83	07	00	20	90	07	00	D0	16	08	00
000965B0	50	10	02	00	60	8E	01	00	90	D4	07	00	10	20	03	00
000965C0	20	8D	02	00	30	20	03	00	40	20	03	00	F0	F7	02	00
000965D0	20	17	08	00	20	17	08	00	50	17	08	00	80	E8	07	00
000965E0	E0	F1	01	00	20	11	03	00	60	20	03	00	70	20	03	00

adofNames，同时得到函数名称 RVA

user32.dll

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00097870	50	2C	03	00	11	A2	09	00	28	A2	09	00	43	A2	09	00
00097880	57	A2	09	00	68	A2	09	00	7B	A2	09	00	94	A2	09	00
00097890	9F	A2	09	00	B9	A2	09	00	D2	A2	09	00	E0	A2	09	00
000978A0	E9	A2	09	00	F5	A2	09	00	01	A3	09	00	1E	A3	09	00
000978B0	33	A3	09	00	45	A3	09	00	59	A3	09	00	64	A3	09	00
000978C0	6F	A3	09	00	80	A3	09	00	97	A3	09	00	AF	A3	09	00
000978D0	C9	A3	09	00	E3	A3	09	00	FB	A3	09	00	0C	A4	09	00

前五函数名称

00099000	72	61	6D	65	41	72	72	69	76	61	6C	54	69	6D	65	73	rameArrivalTimes
00099010	00	41	63	74	69	76	61	74	65	4B	65	79	62	6F	61	72	ActivateKeyboar
00099020	64	4C	61	79	6F	75	74	00	41	64	64	43	6C	69	70	62	dLayout AddClipb
00099030	6F	61	72	64	46	6F	72	6D	61	74	4C	69	73	74	65	6E	oardFormatListen
00099040	65	72	00	41	64	64	56	69	73	75	61	6C	49	64	65	6E	er AddVisualIden
00099050	74	69	66	69	65	72	00	41	64	6A	75	73	74	57	69	6E	tifier AdjustWin
00099060	64	6F	77	52	65	63	74	00	41	64	6A	75	73	74	57	69	dowRect AdjustWi
00099070	6E	64	6F	77	52	65	63	74	45	78	00	41	64	6A	75	73	ndowRectEx Adjus
00099080	74	57	69	6E	64	6F	77	52	65	63	74	45	78	46	6F	72	tWindowRectExFor

adofOrdinals

user32.dll

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00098810	80	EA	09	00	8B	EA	09	00	02	00	03	00	04	00	05	00
00098820	06	00	07	00	08	00	09	00	0A	00	0B	00	0C	00	0D	00
00098830	0E	00	0F	00	10	00	11	00	12	00	13	00	14	00	15	00
00098840	16	00	17	00	18	00	19	00	1A	00	1B	00	1C	00	1D	00
00098850	1E	00	1F	00	20	00	21	00	22	00	23	00	24	00	25	00
00098860	26	00	27	00	28	00	29	00	2A	00	2B	00	2C	00	2D	00
00098870	2E	00	2F	00	35	00	36	00	37	00	38	00	39	00	3A	00
00098880	3B	00	3C	00	3D	00	3E	00	3F	00	40	00	41	00	42	00
00098890	43	00	44	00	45	00	46	00	47	00	48	00	49	00	4A	00
000988A0	4B	00	4C	00	4D	00	4E	00	4F	00	50	00	51	00	52	00

因而前 5 个导出函数信息如下

Ordinal	RVA	Symbol Name
0x05DE	0x0004EFA0	n/a
0x05DF	0x0004DBA0	"GetPointerFrameArrivalTimes"
0x05E0	0x0002B080	"ActivateKeyboardLayout"
0x05E1	0x0002B750	"AddClipboardFormatListener"
0x05E2	0x00032000	"AddVisualIdentifier"
0x05E3	0x0007D400	"AdjustWindowRect"
0x05E4	0x00018A20	"AdjustWindowRectEx"

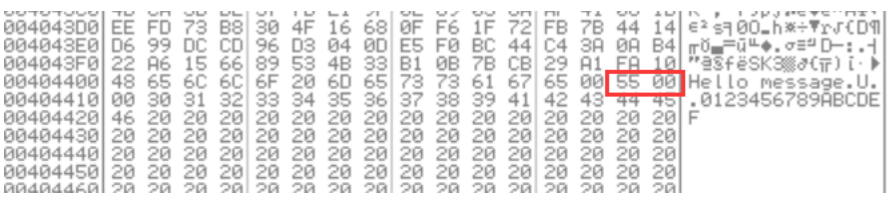
2. 编写程序并生成 exe 文件，要求定义 1048 个字节长度的 word 数组，在程序中对该数组赋随机数，然后查找该数组的最小值，并调用 MessageBox 函数和 ExitProcess 函数。

代码：

```
.386
.model flat,stdcall
option casemap:none
.stack 4096

;include windows.inc
include irvine32.inc
includelib irvine32.lib
includelib user32.lib
includelib kernel32.lib
includelib masm32.lib
;include user32.inc
;include kernel32.inc
ExitProcess PROTO, dwExitCode:DWORD

.data
cnt = 512                    ; 1048 字节即 512 单元的 WORD
arr word cnt DUP(?)
msg db 'Hello message', 0
minc word 0FFFFh             ;最小值

.code
main proc

    invoke MessageBox,NULL,offset msg, NULL, MB_OK
```

```
    mov esi, offset arr
    mov ecx, Lengthof arr
    mov bx, minc

;随机赋值
L1:
    mov eax, 10000h          ;随机范围 0~FFFFh
    call RandomRange         ;从链接库生成随机数
    mov [esi], ax            ;在数组中插入值

    cmp bx,ax                ;if minc < ax
    jb L1B                   ;不更新 minc
    mov bx, ax               ;else minc = ax

L1B:
    add esi, TYPE WORD       ;指向下一元素
    sub ecx,1
    cmp cx, 0                ;循环未结束
    ja L1

    mov minc, bx
    invoke ExitProcess,NULL

main endp
    end main
```

因为是伪随机数，所以最小值恒定为 0055h



3. 请解析题 2 生成 exe 文件的节表，加载前、后导入函数的详细信息。

各段地址以及偏移量的朴素寻找方法已在第一题中实现，因而此处借用 LordPE 提高效率

[区段表]

名称	VOffset	VSize	ROffset	RSize	标志
.text	00001000	000010C6	00000400	00001200	60000020
.rdata	00003000	00000336	00001600	00000400	40000040
.data	00004000	00000F05	00001A00	00000E00	C0000040

User32.dll

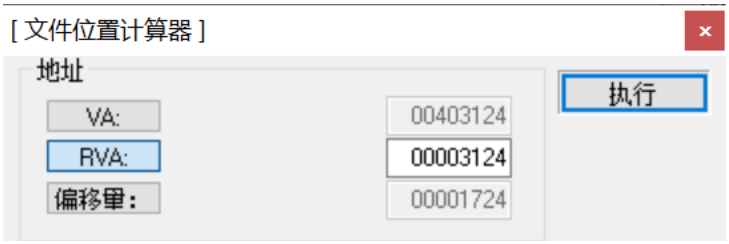
```
->Import Table
   1. ImageImportDescriptor:
    OriginalFirstThunk:  0x0000311C
    TimeDateStamp:       0x00000000  (GMT: Thu Jan 01 00:00:00 1970)
    ForwarderChain:      0x00000000
    Name:                0x00003132  ("USER32.dll")
    FirstThunk:          0x0000306C

    Ordinal/Hint API name
    ------------ ----------------------------------------
    0x020E       "MessageBoxA"
```

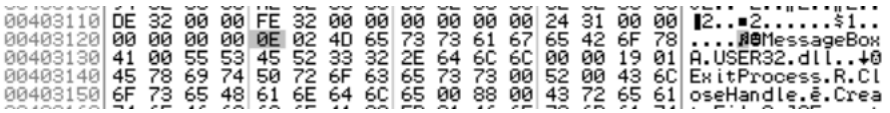
载入前：

OFT 与 FT 指向的 ImageThunkData 数组（OFFSET 171ch 与 166ch）

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00001660	DE	32	00	00	FE	32	00	00	00	00	00	00	24	31	00	00
00001670	00	00	00	00	1C	31	00	00	00	00	00	00	00	00	00	00
00001680	32	31	00	00	6C	30	00	00	B0	30	00	00	00	00	00	00
00001690	00	00	00	00	28	33	00	00	00	30	00	00	00	00	00	00
000016A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000016B0	16	32	00	00	1C	33	00	00	3E	31	00	00	4C	31	00	00
000016C0	5A	31	00	00	68	31	00	00	7A	31	00	00	94	31	00	00
000016D0	A6	31	00	00	B8	31	00	00	D6	31	00	00	E6	31	00	00
000016E0	F6	31	00	00	06	32	00	00	22	32	00	00	36	32	00	00
000016F0	46	32	00	00	5A	32	00	00	66	32	00	00	82	32	00	00
00001700	94	32	00	00	AE	32	00	00	B6	32	00	00	CE	32	00	00
00001710	DE	32	00	00	FE	32	00	00	00	00	00	00	24	31	00	00
00001720	00	00	00	00	0E	02	4D	65	73	73	61	67	65	42	6F	78
00001730	41	00	55	53	45	52	33	32	2E	64	6C	6C	00	00	19	01

载入后：通过 rva 计算 va，找到





```
00403110 DE 32 00 00 FE 32 00 00 00 00 00 00 24 31 00 00  ▮2..▪2......$1..
00403120 00 00 00 00 0E 02 4D 65 73 73 61 67 65 42 6F 78  ....♫☺MessageBox
00403130 41 00 55 53 45 52 33 32 2E 64 6C 6C 00 00 19 01  A.USER32.dll..↓☺
00403140 45 78 69 74 50 72 6F 63 65 73 73 00 52 00 43 6C  ExitProcess.R.Cl
00403150 6F 73 65 48 61 6E 64 6C 65 00 88 00 43 72 65 61  oseHandle.ê.Crea
```

Kernel32.dll

```
2. ImageImportDescriptor:
 OriginalFirstThunk:   0x000030B0
 TimeDateStamp:        0x00000000  (GMT: Thu Jan 01 00:00:00 1970)
 ForwarderChain:       0x00000000
 Name:                 0x00003328  ("KERNEL32.dll")
 FirstThunk:           0x00003000

 Ordinal/Hint API name
 ------------ ----------------------------------------
 0x0348       "LocalFree"
 0x0525       "WriteFile"
 0x0119       "ExitProcess"
 0x0052       "CloseHandle"
 0x0088       "CreateFileA"
 0x015D       "FormatMessageA"
 0x0156       "FlushConsoleInputBuffer"
 0x0186       "GetCommandLineA"
 0x01AC       "GetConsoleMode"
 0x01B2       "GetConsoleScreenBufferInfo"
 0x0202       "GetLastError"
 0x0203       "GetLocalTime"
 0x0264       "GetStdHandle"
 0x0277       "GetSystemTime"
 0x038B       "PeekConsoleInputA"
 0x03B4       "ReadConsoleA"
 0x03B5       "ReadConsoleInputA"
 0x03C0       "ReadFile"
 0x0431       "SetConsoleCursorPosition"
 0x043D       "SetConsoleMode"
 0x0446       "SetConsoleTextAttribute"
 0x04B2       "Sleep"
 0x04BD       "SystemTimeToFileTime"
 0x051A       "WriteConsoleA"
 0x0521       "WriteConsoleOutputCharacterA"
```

同理载入前：

OFT 与 FT 指向的 ImageThunkData 数组（OFFSET 16B0h 与 1600h）

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII
00001600	16	32	00	00	1C	33	00	00	3E	31	00	00	4C	31	00	00	2 3 >1 L1
00001610	5A	31	00	00	68	31	00	00	7A	31	00	00	94	31	00	00	Z1 h1 z1 ”1
00001620	A6	31	00	00	B8	31	00	00	D6	31	00	00	E6	31	00	00	¦1 ¸1 Ö1 æ1
00001630	F6	31	00	00	06	32	00	00	22	32	00	00	36	32	00	00	ö1 2 "2 62
00001640	46	32	00	00	5A	32	00	00	66	32	00	00	82	32	00	00	F2 Z2 f2 ‚2
00001650	94	32	00	00	AE	32	00	00	B6	32	00	00	CE	32	00	00	”2 ®2 ¶2 Î2
00001660	DE	32	00	00	FE	32	00	00	00	00	00	00	24	31	00	00	Þ2 þ2 $1
00001670	00	00	00	00	1C	31	00	00	00	00	00	00	00	00	00	00	1
00001680	32	31	00	00	6C	30	00	00	B0	30	00	00	00	00	00	00	21 l0 °0
00001690	00	00	00	00	28	33	00	00	00	30	00	00	00	00	00	00	(3 0
000016A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000016B0	16	32	00	00	1C	33	00	00	3E	31	00	00	4C	31	00	00	2 3 >1 L1
000016C0	5A	31	00	00	68	31	00	00	7A	31	00	00	94	31	00	00	Z1 h1 z1 ”1
000016D0	A6	31	00	00	B8	31	00	00	D6	31	00	00	E6	31	00	00	¦1 ¸1 Ö1 æ1
000016E0	F6	31	00	00	06	32	00	00	22	32	00	00	36	32	00	00	ö1 2 "2 62
000016F0	46	32	00	00	5A	32	00	00	66	32	00	00	82	32	00	00	F2 Z2 f2 ‚2
00001700	94	32	00	00	AE	32	00	00	B6	32	00	00	CE	32	00	00	”2 ®2 ¶2 Î2
00001710	DE	32	00	00	FE	32	00	00	00	00	00	00	24	31	00	00	Þ2 þ2 $1

载入后：通过 rva 计算 va，找到