Hexadecimal dump	ASCII dump	Fields	Values	Explanation
4D 5A 00 00-00 00 00 00-00 00 00 00-00 00 00 00	MZ.....	e_magic	'MZ'	constant signature
Offset:0x30 00 00 00 00-00 00 00 00-00 00 00 00-40 00 00 00	.....@...	e_lfanew	0x40	offset of the PE Header ❶
Offset:0x40 50 45 00 00-4C 01 03 00-00 00 00 00-00 00 00 00	PE..L.....	Signature	'PE', 0, 0	constant signature
		Machine	0x14c [intel 386]	processor: ARM/MIPS/Intel/...
		NumberOfSections	3	number of sections ❷
00 00 00 00-E0 00 02 01...	....a...	SizeOfOptionalHeader	0xe0	relative offset of the section table ❷
		Characteristics	0x102 [32b EXE]	EXE/DLL/...
Offset:0x58 ...0B 01 00 00-00 00 00 00	.....	Magic	0x10b [32b]	32 bits/64 bits
		AddressOfEntryPoint	0x1000	where execution starts ❺
00 00 00 00-00 00 00 00-00 10 00 00-00 00 00 00	.....	ImageBase	0x400000	address where the file should be mapped in memory
00 00 00 00-00 00 40 00-00 10 00 00-00 02 00 00	.....@.....	SectionAlignment	0x1000	where sections should start in memory ❷
		FileAlignment	0x200	where sections should start on file ❷
00 00 00 00-00 00 00 00-04 00 00 00-00 00 00 00	.....	MajorSubsystemVersion	4 [NT 4 or later]	required version of Windows
00 40 00 00-00 02 00 00-00 00 00 00-02 00 00 00	.@.....	SizeOfImage	0x4000	total memory space required
00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00	.....	SizeOfHeaders	0x200	total size of the headers ❸
00 00 00 00-10 00 00 00...	.....	Subsystem	2 [GUI]	driver/graphical/command line/...
		NumberOfRvaAndSizes	16	number of data directories ❹
...00 00 00 00-00 00 00 00	.....			
00 20 00 00-00 00 00 00-00 00 00 00-00 00 00 00	.....	ImportsVA	0x2000	RVA* of the imports ❹
00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00	.....			

Offset:0x138

2E 74 65 78-74 00 00 00	.text...
00 10 00 00-00 10 00 00-00 02 00 00-00 02 00 00	.....
00 00 00 00-00 00 00 00-00 00 00 00-20 00 00 60	.....`
2E 72 64 61-74 61 00 00-00 10 00 00-00 20 00 00	.rdata.....
00 02 00 00-00 04 00 00-00 00 00 00-00 00 00 00	.....
00 00 00 00-40 00 00 40-2E 64 61 74-61 00 00 00	....@..@.data...
00 10 00 00-00 30 00 00-00 02 00 00-00 06 00 00	.....0.....
00 00 00 00-00 00 00 00-00 00 00 00-40 00 00 C0	.....@..+

Sections table

Name	VirtualSize (RVA*)	VirtualAddress (RVA*)	SizeOfRawData (physical size)	PointerToRawData (physical offset)	Characteristics
.text	0x1000	0x1000	0x200	0x200	CODE EXECUTE READ
.rdata	0x1000	0x2000	0x200	0x400	INITIALIZED READ
.data	0x1000	0x3000	0x200	0x600	DATA READ WRITE

For each section, a SizeofRawData sized block is read from the file at PointerToRawData offset.
It will be loaded in memory at address ImageBase + VirtualAddress in a VirtualSize sized block, with specific characteristics.

Offset:0x200/RVA:0x401000

6A 00 68 00-30 40 00 68-17 30 40 00-6A 00 FF 15	j.h.0@.h.0@.j. .
70 20 40 00-6A 00 FF 15-68 20 40 00	p.@.j. .h.@.

x86 assembly / Equivalent C code

```
push 0
push 0x403000
push 0x403017
push 0
call [0x402070]   ➤MessageBox(0, "Hello World!","a simple PE executable", 0);
push 0
call [0x402068]   ➤ExitProcess(0);
```

Offset:0x400/RVA:0x402000

3C 20 00 00-00 00 00 00-00 00 00 00-78 20 00 00	<.....x...
68 20 00 00-44 20 00 00-00 00 00 00-00 00 00 00	h...D.....
85 20 00 00-70 20 00 00-00 00 00 00-00 00 00 00	à...p.....
00 00 00 00-00 00 00 00-00 00 00 00-4C 20 00 00	.....L...
00 00 00 00-5A 20 00 00-00 00 00 00-00 00 45 78	....Z.....Ex
69 74 50 72-6F 63 65 73-73 00 00 00-4D 65 73 73	itProcess...Mess
61 67 65 42-6F 78 41 00-4C 20 00 00-00 00 00 00	ageBoxA.L.....
5A 20 00 00-00 00 00 00-6B 65 72 6E-65 6C 33 32	Z.....kernel32
2E 64 6C 6C-00 75 73 65-72 33 32 2E-64 6C 6C 00	.dll.user32.dll.

Imports structures

descriptors

- 0x203c → 0x204c, 0 (INT*) → 0,ExitProcess (Hint,Name)
- 0x2078 → kernel32.dll
- 0x2068 → 0x204c, 0 (IAT*)
- 0x2044 → 0x205a, 0 (INT*) → 0,MessageBoxA (Hint,Name)
- 0x2085 → user32.dll
- 0x2070 → 0x205a, 0 (IAT*)
- 0 0 0 0 0

All addresses here are RVAs.*

Consequences

after loading,
0x402068 will point to kernel32.dll´s ExitProcess
0x402070 will point to user32.dll´s MessageBoxA

Offset:0x600/RVA:0x403000

61 20 73 69-6D 70 6C 65-20 50 45 20-65 78 65 63	a.simple.PE.exec
75 74 61 62-6C 65 00 48-65 6C 6C 6F-20 77 6F 72	utable.Hello.wor
6C 64 21 00	ld!.

Strings

```
a simple PE executable\0
Hello world!\0
```

DOS header — shows it's a binary

PE header — shows it's a 'modern' binary

optional header — executable information

data directories — pointers to extra structures (exports, imports,...)

sections table — defines how the file is loaded in memory

code — what is executed

imports — link between the executable and (Windows) libraries

data — information used by the code

This is the whole file, however, most PE files contain more elements. Explanations are simplified, for conciseness.