WINDOWS EXECUTABLE FILE

PE FILE FORMAT

PE FILE

CR3DENZA

PE FILE

PORTABLE EXECUTABLE FILE

- Windows 운영체제에서 사용되는 실행 파일 형식
- UNIX의 COFF를 기반으로 MS에서 제작

  *COFF : Common Object File Format
- 메모리에서도 디스크에 저장된 파일 형태로 바로 실행될 수 있도록 설계
- Win32 의 기본 파일 형식
- EXE 파일, 동적 링크 라이브러리(DLL) 파일이 PE 파일 형식
- PE, PE32는 32비트 실행파일이고 PE+, PE32+는 64비트 실행파일

PE FILE

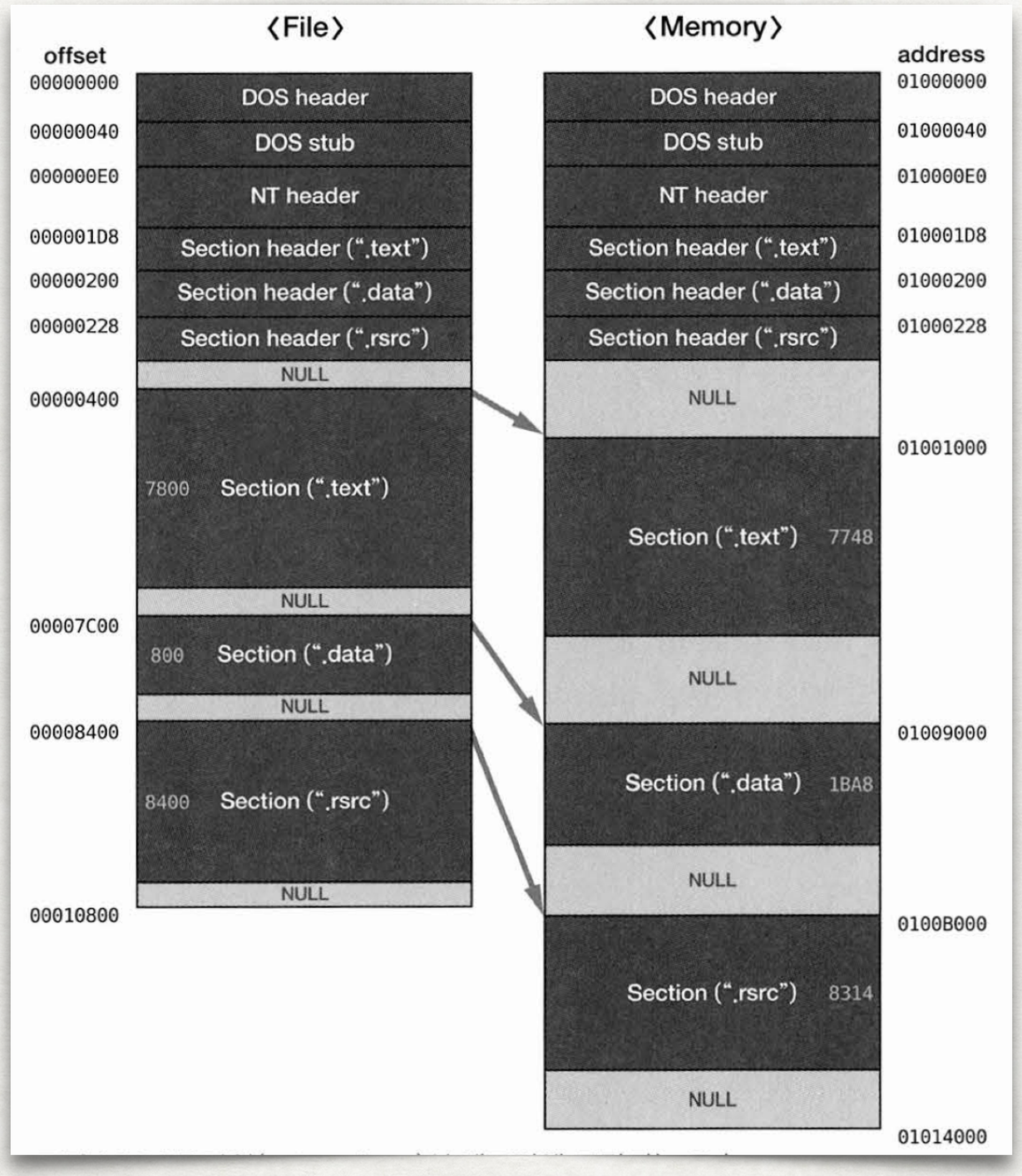
PORTABLE EXECUTABLE FILE

종류	주요확장자
실행 계열	EXE, SCR
드라이버 계열	SYS, VXD
라이브러리 계열	DLL, OCX, CPL, DRV
오브젝트 파일 계열	OBJ

*오브젝트 파일을 제외하고는 모두 실행가능한 파일

PE FILE

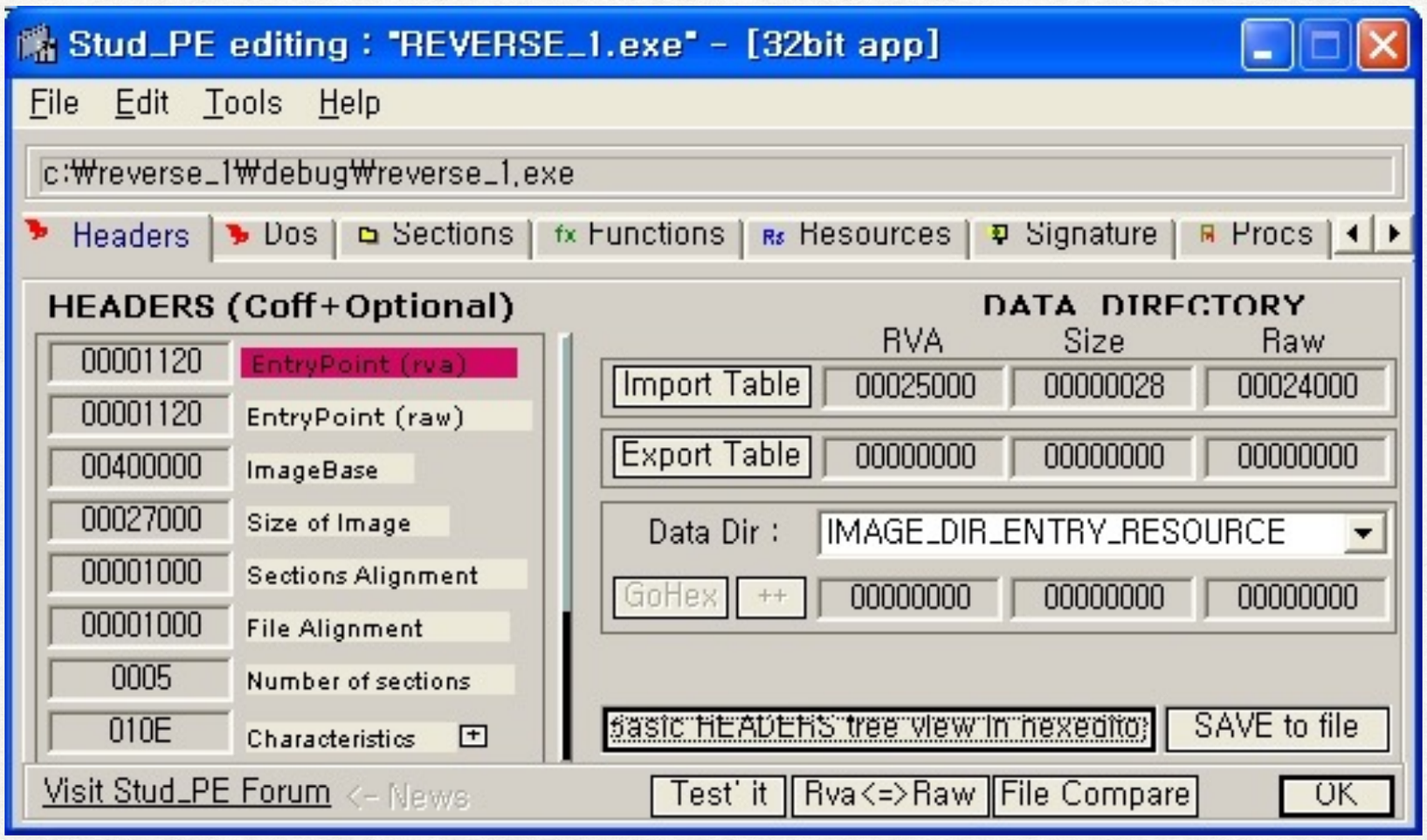
PORTABLE EXECUTABLE FILE



PE FILE

PORTABLE EXECUTABLE FILE

- PE 파일 구조 확인을 위해서 Winhex, 010 Editor 사용



VA & RVA

VIRTUAL ADDRESS & RELATIVE VIRTUAL ADDRESS

- VA 는 프로세스 가상 메모리의 절대주소
  *VA : Virtual Address

- RVA는 어느 기준 위치(ImageBase)에서부터의 상대주소
  *RVA : Relative Virtual Address

- VA = ImageBase + RVA

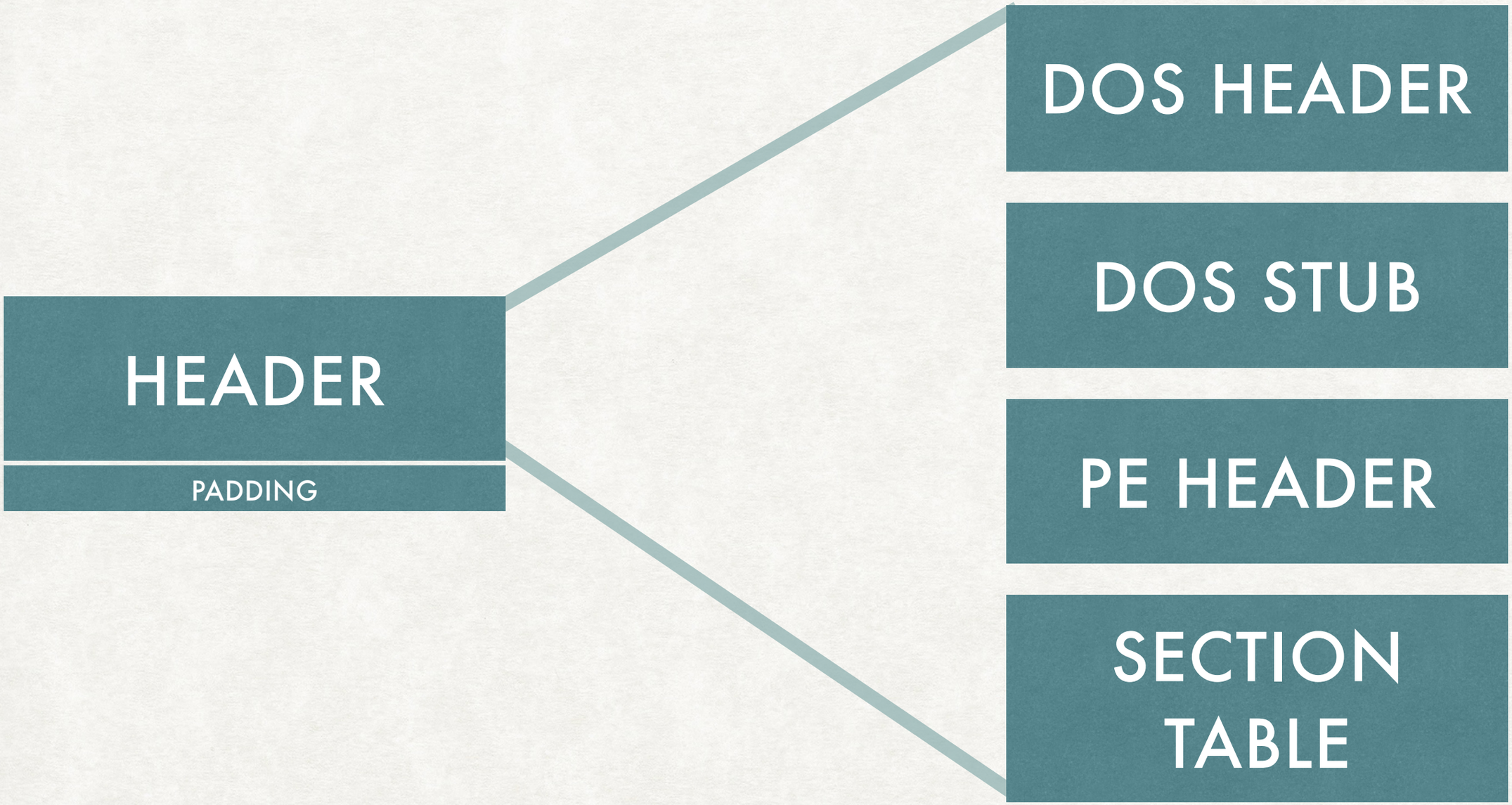
- RVA를 사용하면 바이너리가 메모리 어느 주소로 로드되던지 상관없음

PE FILE FORMAT

PORTABLE EXECUTABLE FILE

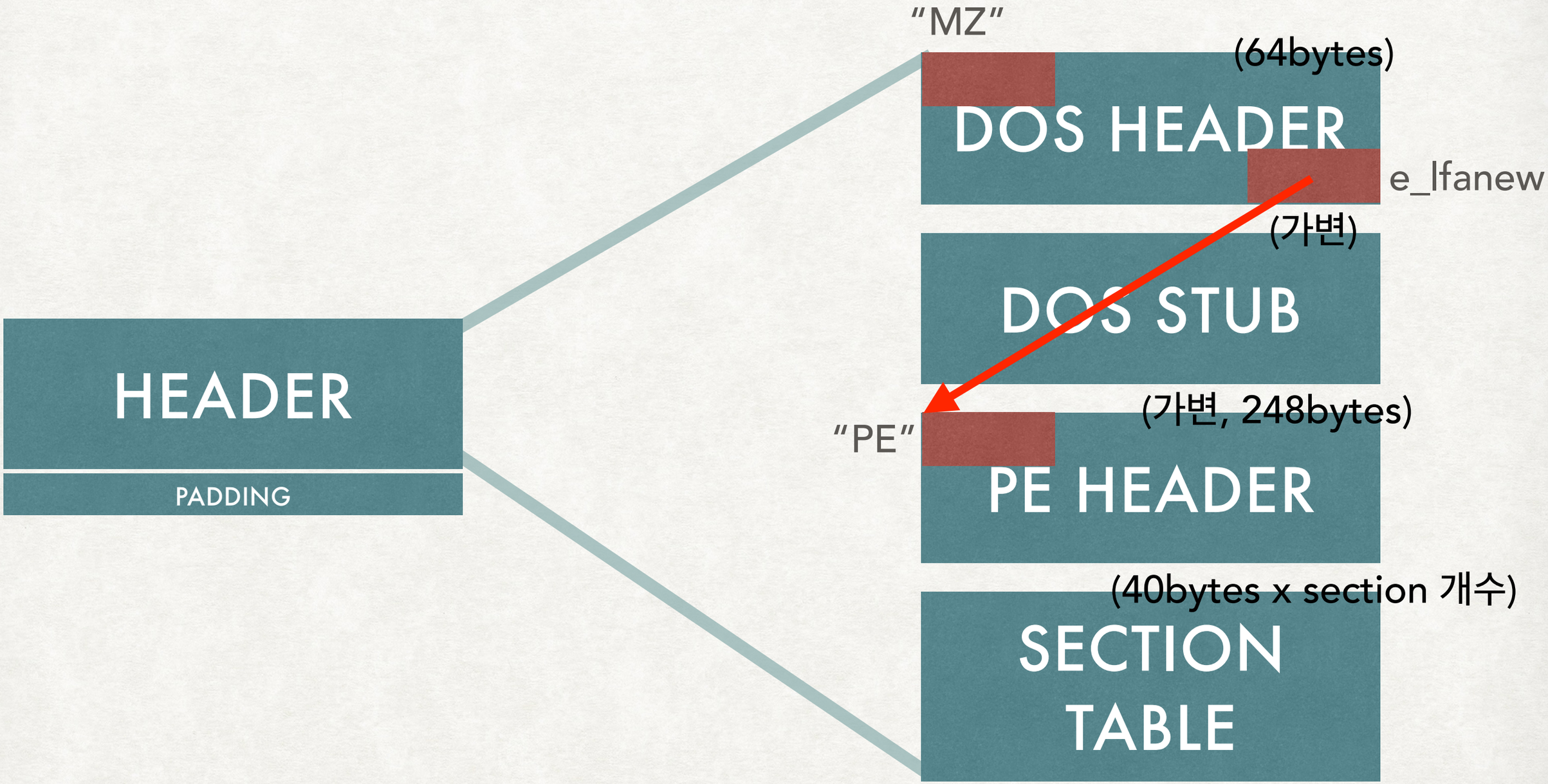
HEADER

PORTABLE EXECUTABLE FILE



HEADER

PORTABLE EXECUTABLE FILE



HEADER

e_magic

e_lfanew

Signature

```
0000h: 4D 5A 00 00 00 00 00 00 00 00 00 00 00 00 00 00  MZ..............
0010h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0030h: 00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 00  ............@...
0040h: 50 45 00 00 4C 01 03 00 00 00 00 00 00 00 00 00  PE..L...........
0050h: 00 00 00 00 E0 00 0F 01 0B 01 00 00 00 02 00 00  ....à...........
0060h: 00 00 00 00 00 00 00 00 00 10 00 00 00 10 00 00  ................
0070h: 00 20 00 00 00 00 40 00 00 10 00 00 00 02 00 00  . ....@.........
0080h: 04 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00  ................
0090h: 00 40 00 00 00 02 00 00 00 00 00 00 02 00 00 00  .@..............
00A0h: 00 00 01 00 00 10 00 00 00 00 01 00 00 10 00 00  ................
00B0h: 00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00  ................
00C0h: 00 30 00 00 60 00 00 00 00 00 00 00 00 00 00 00  .0..`...........
00D0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00E0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00F0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0100h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0110h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0120h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0130h: 00 00 00 00 00 00 00 00 2E 74 65 78 74 00 00 00  .........text...
0140h: 20 00 00 00 00 10 00 00 00 02 00 00 00 02 00 00  ................
0150h: 00 00 00 00 00 00 00 00 00 00 00 00 20 00 00 60  ............ ..`
0160h: 2E 64 61 74 61 00 00 00 23 00 00 00 00 20 00 00  .data...#.... ..
0170h: 00 02 00 00 00 04 00 00 00 00 00 00 00 00 00 00  ................
0180h: 00 00 00 00 40 00 00 C0 2E 72 64 61 74 61 00 00  ....@..À.rdata..
0190h: 60 00 00 00 00 30 00 00 00 02 00 00 00 06 00 00  `....0..........
01A0h: 00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 40  ............@..@
```

DOS HEADER

_IMAGE_DOS_HEADER

```
typedef struct _IMAGE_DOS_HEADER {
    WORD   e_magic;              // DOS signature : 4D5A ("MZ")
    WORD   e_cblp;
    WORD   e_cp;
    WORD   e_crlc;
    WORD   e_cparhdr;
    WORD   e_minalloc;
    WORD   e_maxalloc;
    WORD   e_ss;
    WORD   e_sp;
    WORD   e_csum;
    WORD   e_ip;
    WORD   e_cs;
    WORD   e_lfarlc;
    WORD   e_ovno;
    WORD   e_res[4];
    WORD   e_oemid;
    WORD   e_oeminfo;
    WORD   e_res2[10];
    LONG   e_lfanew;             // offset to NT header

} IMAGE_DOS_HEADER, *PIMAGE_DOS_HEADER;
```

PE HEADER

_IMAGE_NT_HEADERS

```
typedef struct _IMAGE_NT_HEADERS {
    DWORD                   Signature;
    IMAGE_FILE_HEADER       FileHeader;
    IMAGE_OPTIONAL_HEADER   OptionalHeader;
}IMAGE_NT_HEADERS, *PIMAGE_NT_HEADERS;
```

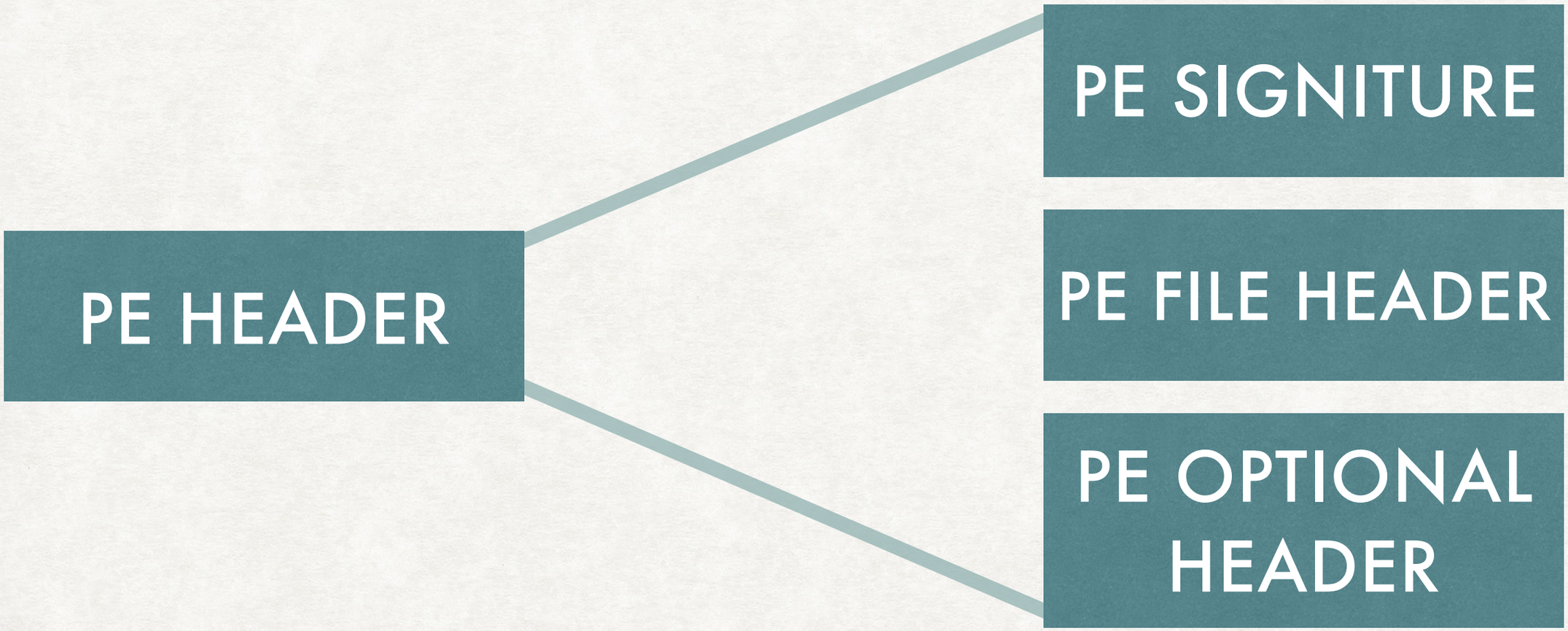
SECTION TABLE

_IMAGE_SECTION_HEADER

```
typedef struct _IMAGE_SECTION_HEADER {
    BYTE  Name[IMAGE_SIZEOF_SHORT_NAME];
    union {
      DWORD PhysicalAddress;
      DWORD VirtualSize;
    }Misc;
    DWORD VirtualAddress;
    DWORD SizeOfRawData;
    DWORD PointerToRawData;
    DWORD PointerToRelocations;
    DWORD PointerToLinenumbers;
    WORD  NumberOfRelocations;
    WORD  NumberOfLinenumbers;
    DWORD Characteristics;
}IMAGE_SECTION_HEADER, *PIMAGE_SECTION_HEADER;
```

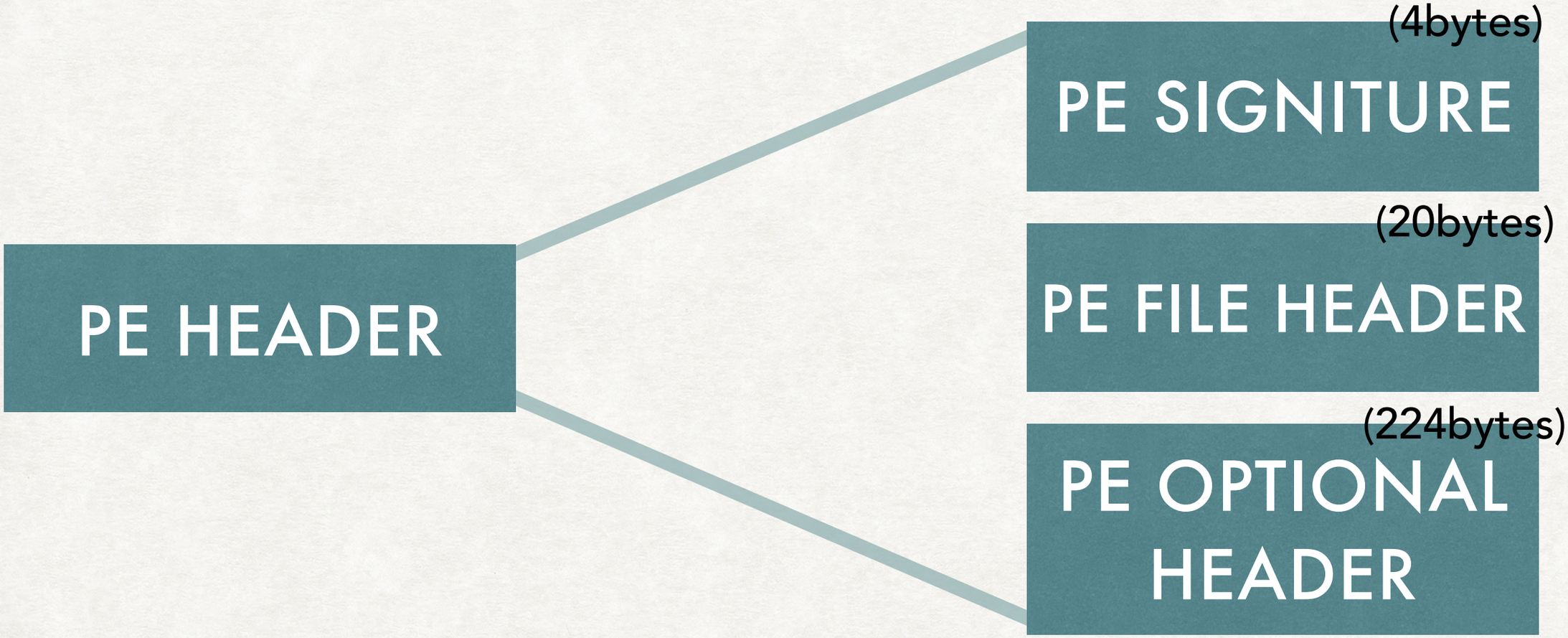
PE HEADER

PORTABLE EXECUTABLE FILE



PE HEADER

PORTABLE EXECUTABLE FILE



PE HEADER

PORTABLE EXECUTABLE FILE

PE Signature

PE FILE HEADER

```
0040h: 50 45 00 00 4C 01 03 00 00 00 00 00 00 00 00 00  PE..L...........
0050h: 00 00 00 00 E0 00 0F 01 0B 01 00 00 00 02 00 00  ....à...........
0060h: 00 00 00 00 00 00 00 00 00 10 00 00 00 10 00 00  ................
0070h: 00 20 00 00 00 00 40 00 00 10 00 00 00 02 00 00  . ....@.........
0080h: 04 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00  ................
0090h: 00 40 00 00 00 02 00 00 00 00 00 00 02 00 00 00  .@..............
00A0h: 00 00 01 00 00 10 00 00 00 00 01 00 00 10 00 00  ................
00B0h: 00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00  ................
00C0h: 00 30 00 00 60 00 00 00 00 00 00 00 00 00 00 00  .0..`...........
00D0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00E0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00F0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0100h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0110h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0120h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0130h: 00 00 00 00 00 00 00 00 2E 74 65 78 74 00 00 00  .........text...
```

PE OPTIONAL HEADER

PE SIGNATURE

SIGNATURE

```
DWORD PESignature;
```

PE FILE HEADER

_IMAGE_FILE_HEADER

```
typedef struct _IMAGE_FILE_HEADER {
    WORD  Machine;
    WORD  NumberOfSections;
    DWORD TimeDateStamp;
    DWORD PointerToSymbolTable;
    DWORD NumberOfSymbols;
    WORD  SizeOfOptionalHeader;
    WORD  Characteristics;
}IMAGE_FILE_HEADER, *PIMAGE_FILE_HEADER;
```

PE OPTIONAL HEADER

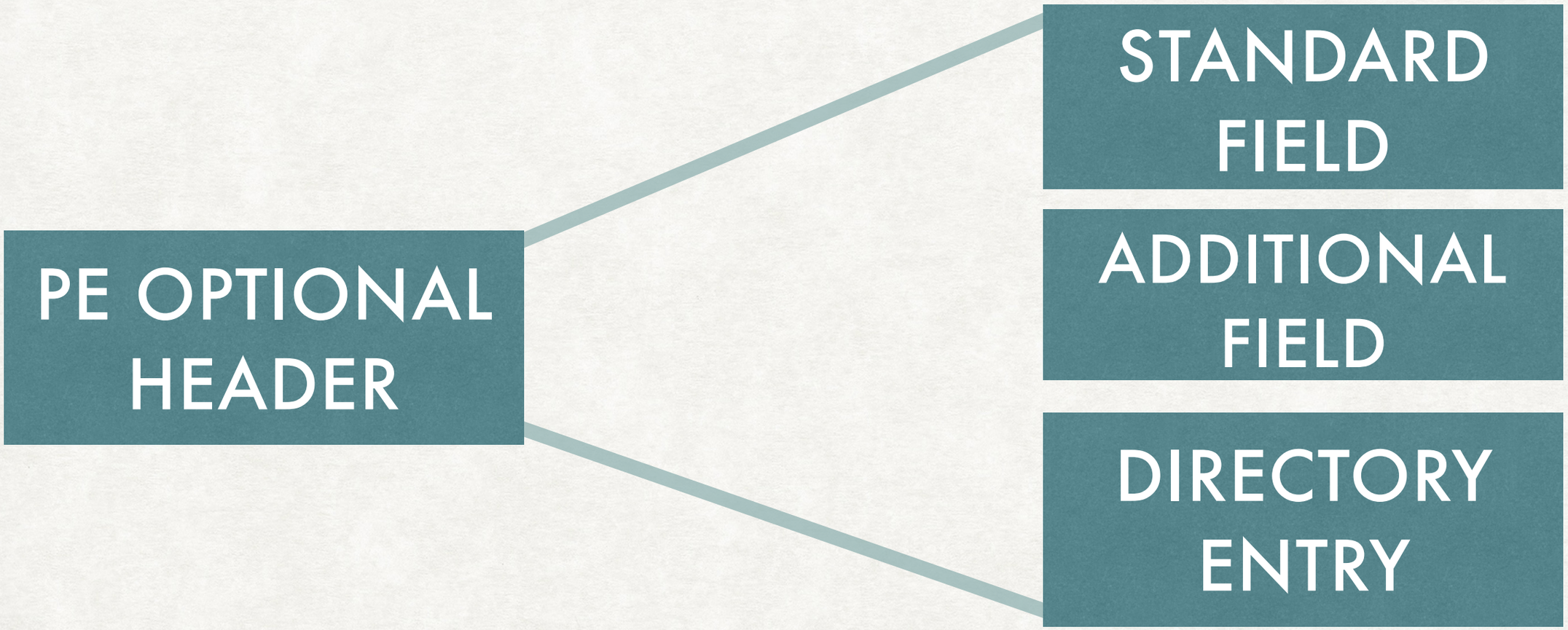
_IMAGE_OPTIONAL_HEADER

```
typedef struct _IMAGE_OPTIONAL_HEADER {
  WORD                 Magic;
  BYTE                 MajorLinkerVersion;
  BYTE                 MinorLinkerVersion;
  DWORD                SizeOfCode;
  DWORD                SizeOfInitializedData;
  DWORD                SizeOfUninitializedData;
  DWORD                AddressOfEntryPoint;
  DWORD                BaseOfCode;
  DWORD                BaseOfData;
  DWORD                ImageBase;
  DWORD                SectionAlignment;
  DWORD                FileAlignment;
  WORD                 MajorOperatingSystemVersion;
  WORD                 MinorOperatingSystemVersion;
  WORD                 MajorImageVersion;
  WORD                 MinorImageVersion;
  WORD                 MajorSubsystemVersion;
  WORD                 MinorSubsystemVersion;
  DWORD                Win32VersionValue;
  DWORD                SizeOfImage;
  DWORD                SizeOfHeaders;
  DWORD                CheckSum;
  WORD                 Subsystem;
  WORD                 DllCharacteristics;
  DWORD                SizeOfStackReserve;
  DWORD                SizeOfStackCommit;
  DWORD                SizeOfHeapReserve;
  DWORD                SizeOfHeapCommit;
  DWORD                LoaderFlags;
  DWORD                NumberOfRvaAndSizes;
  IMAGE_DATA_DIRECTORY DataDirectory[IMAGE_NUMBEROF_DIRECTORY_ENTRIES];
} IMAGE_OPTIONAL_HEADER, *PIMAGE_OPTIONAL_HEADER;
```

```
typedef struct _IMAGE_DATA_DIRECTORY {
  DWORD VirtualAddress;
  DWORD Size;
} IMAGE_DATA_DIRECTORY, *PIMAGE_DATA_DIRECTORY;
```

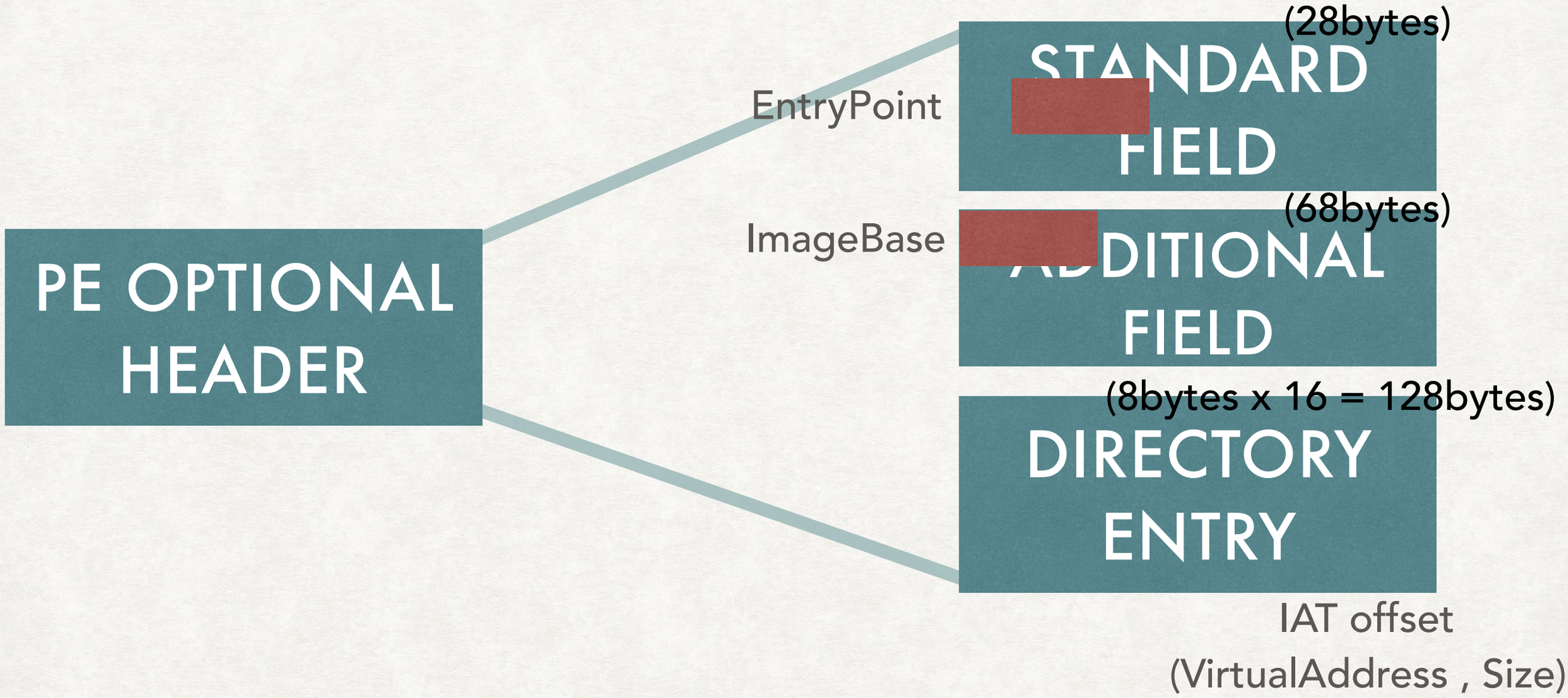
PE OPTIONAL HEADER

PORTABLE EXECUTABLE FILE



PE OPTIONAL HEADER

PORTABLE EXECUTABLE FILE



PE OPTIONAL HEADER

PORTABLE EXECUTABLE FILE

STANDARD FIELD ADDITIONAL FIELD DIRECTORY ENTRY

```
0050h: 00 00 00 00 E0 00 0F 01 0B 01 00 00 00 02 00 00  ....à...........
0060h: 00 00 00 00 00 00 00 00 00 10 00 00 00 10 00 00  ................
0070h: 00 20 00 00 00 00 40 00 00 10 00 00 00 02 00 00  . ....@.........
0080h: 04 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00  ................
0090h: 00 40 00 00 00 02 00 00 00 00 00 00 02 00 00 00  .@..............
00A0h: 00 00 01 00 00 10 00 00 00 00 01 00 00 10 00 00  ................
00B0h: 00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00  ................
00C0h: 00 30 00 00 60 00 00 00 00 00 00 00 00 00 00 00  .0..`...........
00D0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00E0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00F0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0100h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0110h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0120h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0130h: 00 00 00 00 00 00 00 00 2E 74 65 78 74 00 00 00  .........text...
```

PE OPTIONAL HEADER

_IMAGE_OPTIONAL_HEADER

```
typedef struct _IMAGE_OPTIONAL_HEADER {
  WORD                 Magic;
  BYTE                 MajorLinkerVersion;
  BYTE                 MinorLinkerVersion;
  DWORD                SizeOfCode;
  DWORD                SizeOfInitializedData;
  DWORD                SizeOfUninitializedData;
  DWORD                AddressOfEntryPoint;
  DWORD                BaseOfCode;
  DWORD                BaseOfData;
  DWORD                ImageBase;
  DWORD                SectionAlignment;
  DWORD                FileAlignment;
  WORD                 MajorOperatingSystemVersion;
  WORD                 MinorOperatingSystemVersion;
  WORD                 MajorImageVersion;
  WORD                 MinorImageVersion;
  WORD                 MajorSubsystemVersion;
  WORD                 MinorSubsystemVersion;
  DWORD                Win32VersionValue;
  DWORD                SizeOfImage;
  DWORD                SizeOfHeaders;
  DWORD                CheckSum;
  WORD                 Subsystem;
  WORD                 DllCharacteristics;
  DWORD                SizeOfStackReserve;
  DWORD                SizeOfStackCommit;
  DWORD                SizeOfHeapReserve;
  DWORD                SizeOfHeapCommit;
  DWORD                LoaderFlags;
  DWORD                NumberOfRvaAndSizes;
  IMAGE_DATA_DIRECTORY DataDirectory[IMAGE_NUMBEROF_DIRECTORY_ENTRIES];
} IMAGE_OPTIONAL_HEADER, *PIMAGE_OPTIONAL_HEADER;
```

Standard Field

Additional Field

```
typedef struct _IMAGE_DATA_DIRECTORY {
  DWORD VirtualAddress;
  DWORD Size;
} IMAGE_DATA_DIRECTORY, *PIMAGE_DATA_DIRECTORY;
```

Directory Entry

DIRECTORY ENTRY

_IMAGE_DATA_DIRECTORY

```
typedef struct _IMAGE_DATA_DIRECTORY {
  DWORD VirtualAddress;
  DWORD Size;
} IMAGE_DATA_DIRECTORY, *PIMAGE_DATA_DIRECTORY;
```

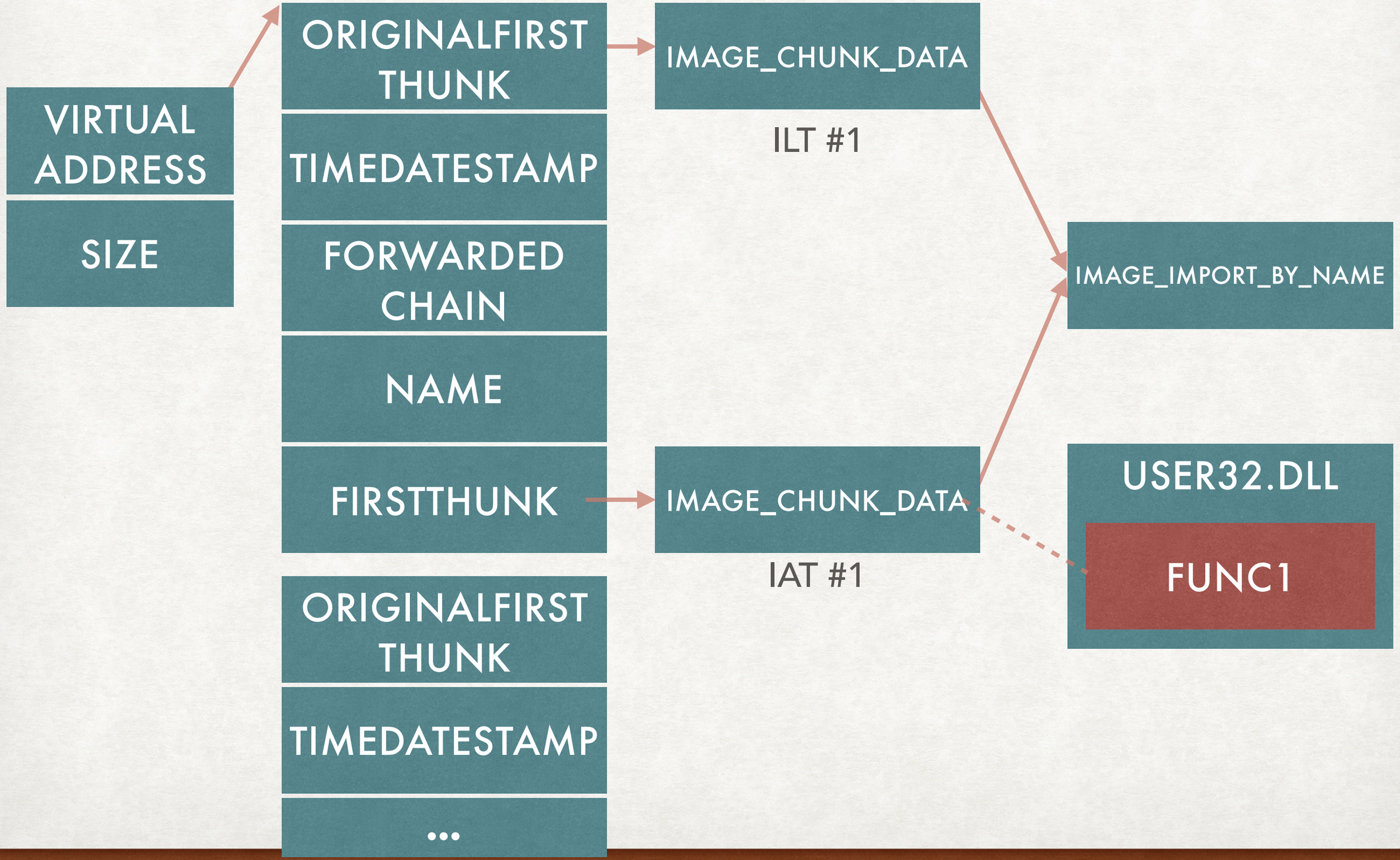
X 16개

```
// Directory Entries
#define IMAGE_DIRECTORY_ENTRY_EXPORT          0   // Export Directory
#define IMAGE_DIRECTORY_ENTRY_IMPORT          1   // Import Directory
#define IMAGE_DIRECTORY_ENTRY_RESOURCE        2   // Resource Directory
#define IMAGE_DIRECTORY_ENTRY_EXCEPTION       3   // Exception Directory
#define IMAGE_DIRECTORY_ENTRY_SECURITY        4   // Security Directory
#define IMAGE_DIRECTORY_ENTRY_BASERELOC       5   // Base Relocation Table
#define IMAGE_DIRECTORY_ENTRY_DEBUG           6   // Debug Directory
//      IMAGE_DIRECTORY_ENTRY_COPYRIGHT       7   // (X86 usage)
#define IMAGE_DIRECTORY_ENTRY_ARCHITECTURE    7   // Architecture Specific Data
#define IMAGE_DIRECTORY_ENTRY_GLOBALPTR       8   // RVA of GP
#define IMAGE_DIRECTORY_ENTRY_TLS             9   // TLS Directory
#define IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG    10   // Load Configuration Directory
#define IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT   11   // Bound Import Directory in headers
#define IMAGE_DIRECTORY_ENTRY_IAT            12   // Import Address Table
#define IMAGE_DIRECTORY_ENTRY_DELAY_IMPORT   13   // Delay Load Import Descriptors
#define IMAGE_DIRECTORY_ENTRY_COM_DESCRIPTOR 14   // COM Runtime descriptor
```

IAT, ILT

IAT, ILT

IMPORT ADDRESS TABLE, IMPORT LOOKUP TABLE



IAT, ILT

IMPORT ADDRESS TABLE, IMPORT LOOKUP TABLE

OriginalFirstThunk FirstThunk Name

```
0600h: 40 30 00 00 00 00 00 00 00 00 00 00 50 30 00 00  @0..........P0..
0610h: 30 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00..............
0620h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0630h: 60 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00  `0..............
0640h: 60 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00  `0..............
0650h: 75 73 65 72 33 32 2E 64 6C 6C 00 00 00 00 00 00  user32.dll......
0660h: 00 00 4D 65 73 73 61 67 65 42 6F 78 41 00 00 00  ..MessageBoxA...
```

IAT ILT Name String Function String

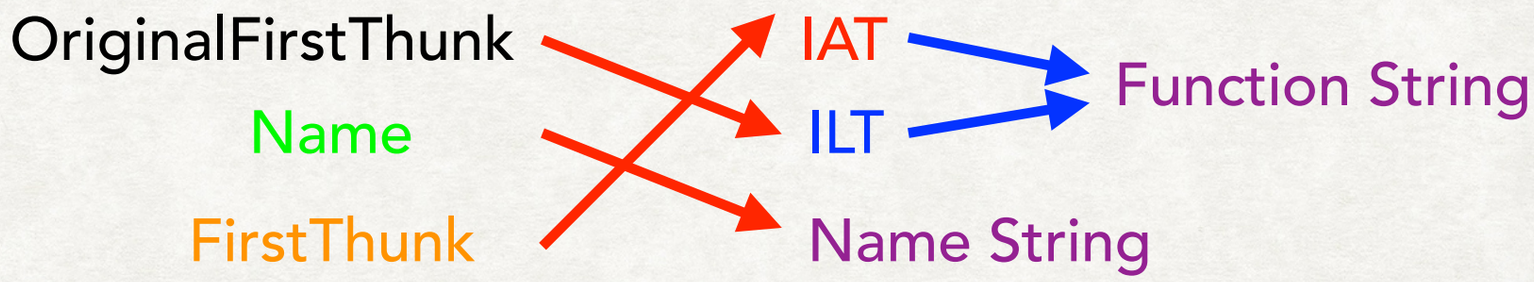
IAT, ILT

IMPORT ADDRESS TABLE, IMPORT LOOKUP TABLE

OriginalFirstThunk FirstThunk Name

```
0600h: 40 30 00 00 00 00 00 00 00 00 00 00 50 30 00 00 @0..........P0..
0610h: 30 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00..............
0620h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0630h: 60 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 `0..............
0640h: 60 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 `0..............
0650h: 75 73 65 72 33 32 2E 64 6C 6C 00 00 00 00 00 00 user32.dll......
0660h: 00 00 4D 65 73 73 61 67 65 42 6F 78 41 00 00 00 ..MessageBoxA...
```

IAT ILT Name String Function String



메모리에 로드되면 IAT는 user32.dll안의 MessageBoxA함수를 가리킨다

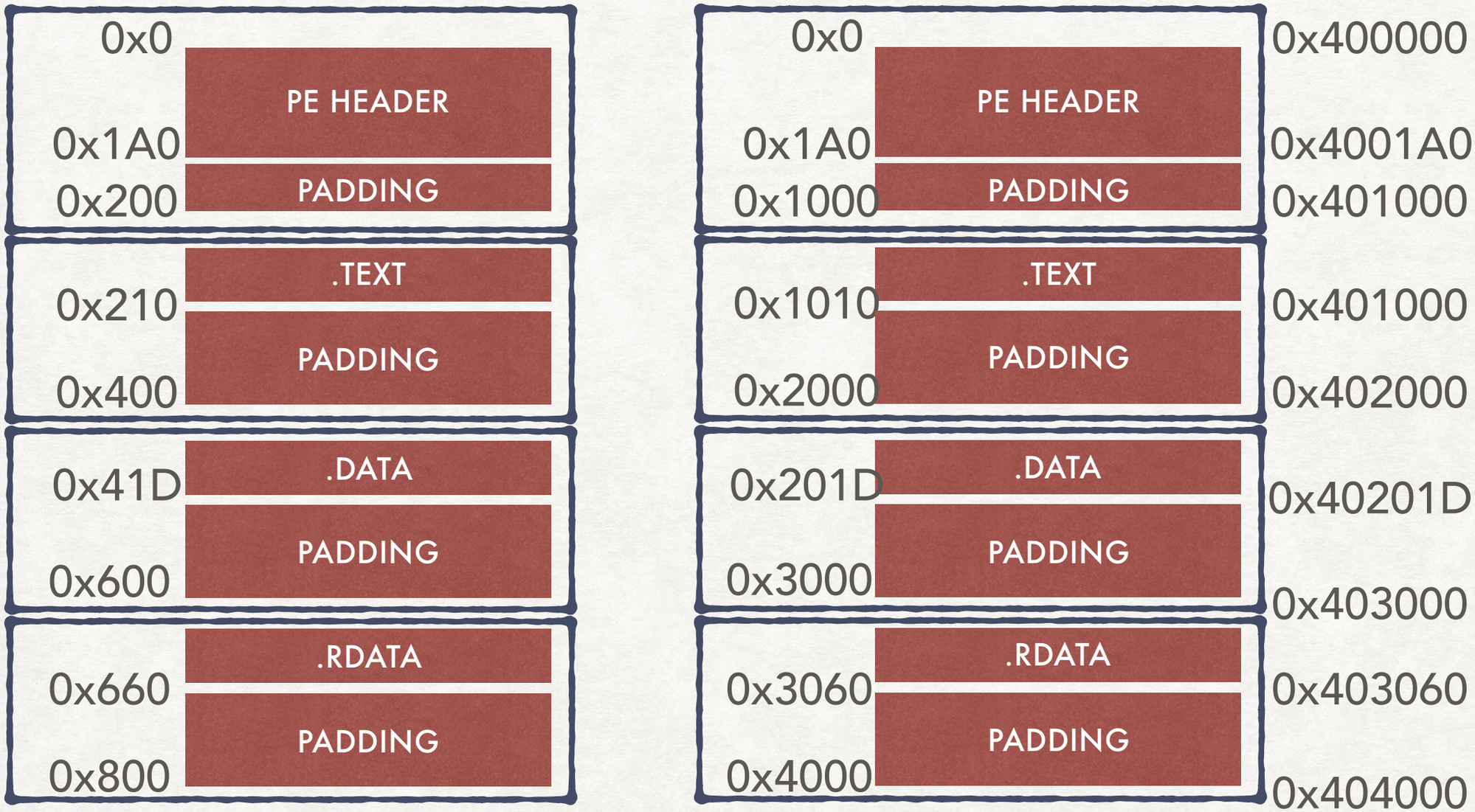
IAT, ILT

_IMAGE_IMPORT_DESCRIPTOR

```
typedef struct _IMAGE_IMPORT_DESCRIPTOR {
    UINT32   Characteristics;
    UINT32   TimeDateStamp;
    UINT32   ForwarderChain;
    UINT32   Name;
    PIMAGE_THUNK_DATA FirstThunk;
} IMAGE_IMPORT_DESCRIPTOR, *PIMAGE_IMPORT_DESCRIPTOR;
```

```
typedef struct _IMAGE_THUNK_DATA {
  union {
     PUINT32 Function;
     UINT32 Ordinal;
     PIMAGE_IMPORT_BY_NAME AddressOfData;
  } u1;
} IMAGE_THUNK_DATA, *PIMAGE_THUNK_DATA;
```

```
typedef struct _IMAGE_IMPORT_BY_NAME {
  UINT16  Hint;
  UINT8   Name[1];
} IMAGE_IMPORT_BY_NAME, *PIMAGE_IMPORT_BY_NAME;
```


0x0
PE HEADER
0x1A0
PADDING
0x200
.TEXT
0x210
PADDING
0x400
.DATA
0x41D
PADDING
0x600
.RDATA
0x660
PADDING
0x800
0x0
PE HEADER
0x1A0
PADDING
0x1000
.TEXT
0x1010
PADDING
0x2000
.DATA
0x201D
PADDING
0x3000
.RDATA
0x3060
PADDING
0x4000
0x400000
0x4001A0
0x401000
0x401000
0x402000
0x40201D
0x403000
0x403060
0x404000

PE FILE

PORTABLE EXECUTABLE FILE

http://research.microsoft.com/en-us/um/redmond/projects/invisible/include/loaders/pe_image.h.htm