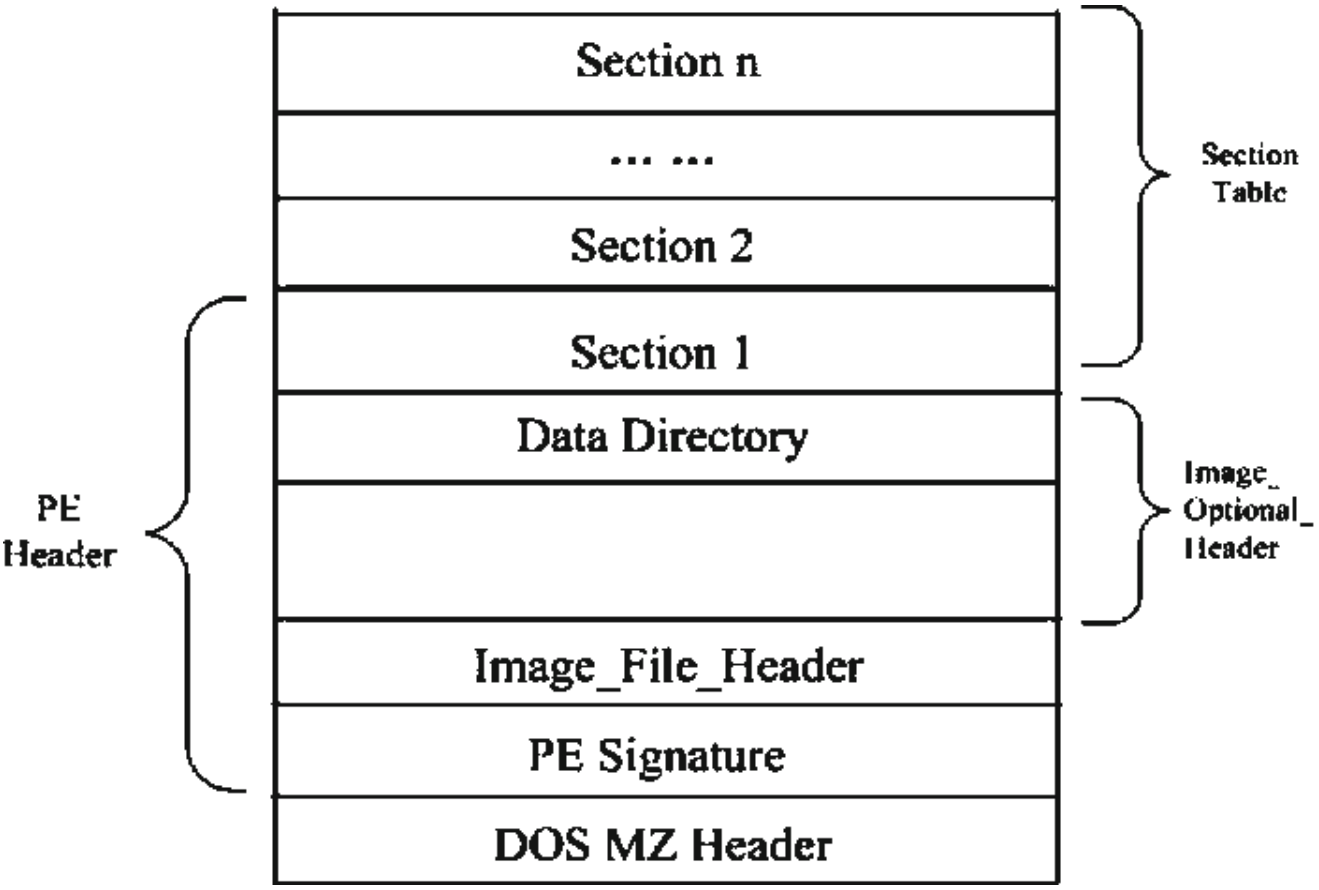
# Tuần 4: Tìm hiểu về PE Format

**A. Sơ lược về PE Format:**

Định dạng PE (Portable Executable) là định dạng cho executables, object code, Dlls … trên cả 2 phiên bản 32-bit và 64-bit của Windows. Các loại file có thể thực thi được trên Win32 như .exe, .dll, .com, .net, … Trong UEFI, PE cũng được đặt là đinh dạng tiêu chuẩn cho môi trường EFI.

Định dạng PE được Microsoft đưa vào từ phiên bản Windows NT 3.1. PE được sinh ra với mục đích làm cầu nối giữa nền DOS và hệ thống NT. Sau này, Microsoft phát triển PE trên cả nền tảng 64-bit với tên gọi PE+ (hay PE32+).

**B. Cấu trúc PE Format**




PE được chia thành 2 phần chính gồm Section và Header

**1. DOS MZ Header**

Phần Header này có kích thước 64 bytes và nằm ở đầu mỗi file. Vùng này cung cấp thông tin cho hệ thống DOS đọc và nhận biết file thực thi.

Cấu trúc:

```
struct _IMAGE_DOS_HEADER {
0x00   WORD e_magic;
0x02   WORD e_cblp;
0x04   WORD e_cp;
0x06   WORD e_crlc;
0x08   WORD e_cparhdr;
0x0a   WORD e_minalloc;
0x0c   WORD e_maxalloc;
0x0e   WORD e_ss;
0x10   WORD e_sp;
0x12   WORD e_csum;
0x14   WORD e_ip;
0x16   WORD e_cs;
0x18   WORD e_lfarlc;
0x1a   WORD e_ovno;
0x1c   WORD e_res[4];
0x24   WORD e_oemid;
0x26   WORD e_oeminfo;
0x28   WORD e_res2[10];
0x3c   DWORD e_lfanew;
};
```

Trường cần quan tâm:
- **e_magic**: đây là trường signature của định dạng PE. e_magic bao gồm 4 bytes đầu tiên của file. Trường này có giá trị là MZ\0\0 (ở dạng hexa).
- **e_lfanew**: trường này chứa giá trị là offset của Nt Header trong file thực thi.

**2. DOS Stub**

Trường này là một đoạn thông báo lỗi trên nền DOS.
Ví dụ:

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00   MZ..........ÿÿ..
00000010   B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ¸.......@.......
00000020   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000030   00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00   ............€...
00000040   0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68   ..°..´.Í!¸.LÍ!Th
00000050   69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F   is program canno
00000060   74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20   t be run in DOS
00000070   6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00   mode....$.......
00000080   50 45 00 00 64 86 11 00 4C 62 5C 59 00 34 0F 00   PE..d†..Lb\Y.4..
00000090   D0 54 00 00 F0 00 27 00 0B 02 02 18 00 0A 07 00   ĐT..ð.'.........
000000A0   00 4E 0A 00 00 18 00 00 00 15 00 00 00 10 00 00   .N..............
000000B0   00 00 40 00 00 00 00 00 00 00 10 00 00 00 02 00   ..@.............
000000C0   04 00 00 00 00 00 00 00 05 00 02 00 00 00 00 00   ................
```

**3. Nt Header**

Trường này chứa các thông tin cần thiết cho quá trình load file vào bộ nhớ.

Cấu trúc:

```
struct _IMAGE_NT_HEADERS {
0x00  DWORD Signature;
0x04  _IMAGE_FILE_HEADER FileHeader;
0x18  _IMAGE_OPTIONAL_HEADER OptionalHeader;
};
```

- **Signature**: gồm 4 bytes chứa PE: 0x50, 0x45, 0x00, 0x00
- **FILE_HEADER**: gồm 20 bytes tiếp theo, chứa thông tin về sơ đồ bố trí và đặc tính của file

Cấu trúc FILE_HEADER:

```
struct _IMAGE_FILE_HEADER {
0x00  WORD Machine;
0x02  WORD NumberOfSections;
0x04  DWORD TimeDateStamp;
0x08  DWORD PointerToSymbolTable;
0x0c  DWORD NumberOfSymbols;
0x10  WORD SizeOfOptionalHeader;
0x12  WORD Characteristics;
};
```

Trường cần chú ý:

- **Machine**: giá trị xác định PE File này được biên dịch cho dòng máy nào
- **NumberOfSections:** đây là trường chứa số section của file. Nếu muốn thêm/xoá section trong PE file, ta cần thay đổi tương ứng trường này.
- **Characteristics**: là bit cờ, xác định định dạng PE File.

- Optional Header: bao gồm 224 bytes tiếp theo sau FILE_HEADER. Cấu trúc này được định nghĩa trong windows.inc, đây là phần chứa thông tin về sơ đồ logic trong PE file.

Cấu trúc của Optional Header:

```
struct _IMAGE_OPTIONAL_HEADER {
0x00  WORD Magic;
0x02  BYTE MajorLinkerVersion;
0x03  BYTE MinorLinkerVersion;
0x04  DWORD SizeOfCode;
0x08  DWORD SizeOfInitializedData;
0x0c  DWORD SizeOfUninitializedData;
0x10  DWORD AddressOfEntryPoint;
0x14  DWORD BaseOfCode;
0x18  DWORD BaseOfData;
0x1c  DWORD ImageBase;
0x20  DWORD SectionAlignment;
0x24  DWORD FileAlignment;
0x28  WORD MajorOperatingSystemVersion;
0x2a  WORD MinorOperatingSystemVersion;
0x2c  WORD MajorImageVersion;
0x2e  WORD MinorImageVersion;
0x30  WORD MajorSubsystemVersion;
0x32  WORD MinorSubsystemVersion;
0x34  DWORD Win32VersionValue;
0x38  DWORD SizeOfImage;
0x3c  DWORD SizeOfHeaders;
0x40  DWORD CheckSum;
0x44  WORD Subsystem;
0x46  WORD DllCharacteristics;
0x48  DWORD SizeOfStackReserve;
0x4c  DWORD SizeOfStackCommit;
0x50  DWORD SizeOfHeapReserve;
0x54  DWORD SizeOfHeapCommit;
0x58  DWORD LoaderFlags;
0x5c  DWORD NumberOfRvaAndSizes;
0x60  _IMAGE_DATA_DIRECTORY DataDirectory[16];
};
```

Trường cần chú ý:

- **Magic** (2 bytes): xác định là file 32 bit (0B 01) hay 64 bit (0B 20)
- **AddressOfEntryPoint** (4bytes): chứa địa chỉ ảo tương đối (RVA) của câu lệnh đầu tiên sẽ được thực thi khi chương trình PE loader sẵn sàng để chạy PE File (.text). Nếu muốn chương trình bắt đầu từ một địa chỉ khác (để thực thi câu lệnh với mục đích khác) thì cần thay đổi địa chỉ này về địa chỉ tương đối của câu lệnh muốn thực thi.
- **ImageBase:** địa chỉ nạp được ưu tiên cho PE File.
- **Section Alignment:** Phần liên kết của các Section trong bộ nhớ, tức là một section luôn luôn được bắt đầu bằng bội số của sectionAlignment. Ví dụ: sectionAlignment là 1000h, section đầu tiên bắt đầu ở vị trí 401000h và kích thước là 10h, section tiếp theo sẽ bắt đầu tại địa chỉ 402000h.
- **File Alignment:** Phần liên kết của các Section trong File. Tương tự như SectionAlignment nhưng áp dụng với file.
- **SizeOfImage:** Toàn bộ kích thước của PE_IMAGE trong bộ nhớ, là tổng của tất cả các headers và sections được liên kết tới Section Alignment
- **SizeOfHeaders:** Kích thước của tất cả các headers + section table = kích thước file - tổng kích thước của các section trong file.
- **Data Directory:** là một mảng gồm 16 phần tử, trong đó mỗi phần liên quan đến một cấu trúc dữ liệu quan trọng trong PE File.

**4. Section Table**

Trường này bao gồm một mảng những cấu trúc kiểu IMAGE_SECTION_HEADER. Mỗi phần tử chứa thông tin về một section trong PE File.

Cấu trúc của một IMAGE_SECTION_HEADER:

```
typedef struct _IMAGE_SECTION_HEADER {
0x00    BYTE  Name[IMAGE_SIZEOF_SHORT_NAME];
        union {
0x08            DWORD PhysicalAddress;
0x08            DWORD VirtualSize;
        } Misc;
0x0c    DWORD VirtualAddress;
0x10    DWORD SizeOfRawData;
0x14    DWORD PointerToRawData;
0x18    DWORD PointerToRelocations;
0x1c    DWORD PointerToLinenumbers;
0x20    WORD  NumberOfRelocations;
0x22    WORD  NumberOfLinenumbers;
0x24    DWORD Characteristics;
};
```

Trường cần chú ý:

- **VirtualSize:** Kích thước thật sự của dữ liệu trên section tính theo byte, giá trị này có thể nhỏ hơn kích thước trên ổ đĩa (SizeOfRawData)

- **VirtualAddress:** RVA của section, là giá trị để ánh xạ khi section được load lên bộ nhớ

- **SizeOfRawData:** Kích thước section data trên ổ đĩa

- **PointerToRawData:** là offset từ vị trí đầu file tới section data.

- **Characteristics:** đặc tính của section: thực thi, dữ liệu khởi tạo.

**5. Import Directory Table**

Trường này là một Data Directory đặt ở đầu .idata section. Trường này bao gồm một dãy các cấu trúc IMAGE_IMPORT_DESCRIPTOR, mỗi một phần tử là 1 Dll được import. Dãy này không có kích thước cố định mà phụ thuộc vào số lượng các Dll được import.

Cấu trúc của IMAGE_IMPORT_DESCRIPTOR:

```
typedef struct _IMAGE_IMPORT_DESCRIPTOR {
    union {
        DWORD   Characteristics;
        DWORD   OriginalFirstThunk;
    } DUMMYUNIONNAME;
    DWORD   TimeDateStamp;
    DWORD   ForwarderChain;
    DWORD   Name;
    DWORD   FirstThunk;
} IMAGE_IMPORT_DESCRIPTOR;
typedef IMAGE_IMPORT_DESCRIPTOR UNALIGNED *PIMAGE_IMPORT_DESCRIPTOR;
```

Trường cần chú ý:

- **OriginalFirstThunk**: RVA của Import Lookup Table
- **TimeDateStamp**: dấu thời gian, có giá trị 0 nếu không giới hạn và 1 nếu giới hạn.
- **FowarderChain**: chỉ số của chuỗi forwarder đầu tiên.
- **Name**: RVA của một xâu ASCII chứa tên của Dll
- **FirstThunk**: RVA của Import Address Table.

**6. Export Directory Table**

Tương tự với Import Directory Table

*Nguồn tham khảo:

https://0xrick.github.io/win-internals/pe6/
https://docs.microsoft.com/en-us/windows/win32/debug/pe-format
https://malwology.com/2018/10/05/exploring-the-pe-file-format-via-imports/