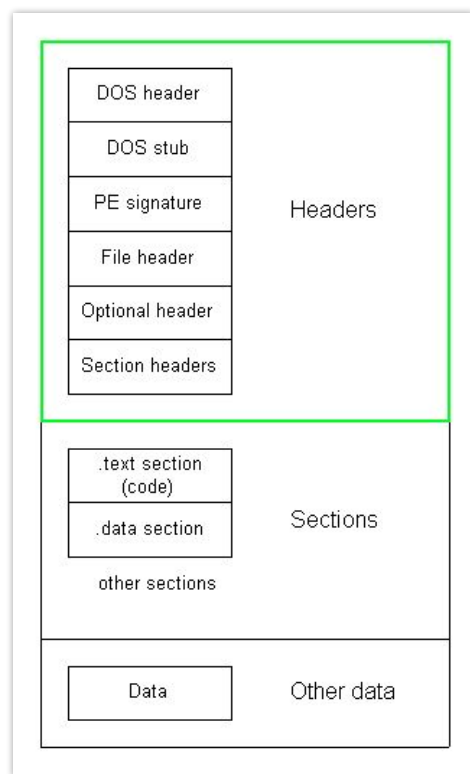


Структура исполняемого файла Portable Executable Часть 1

DOS headers, File header, Optional header, Section headers.

А также RAW, RVA-адресация и выравнивание

Файл PE



- ▣ PE – Portable Executable (Windows)
- ▣ Проецируется в память операционной системой
- ▣ Общая информация и правила проецирования записаны в заголовках

DOS header

DOS MZ Header

```
typedef 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;  
} IMAGE_DOS_HEADER, *PIMAGE_DOS_HEADER;
```

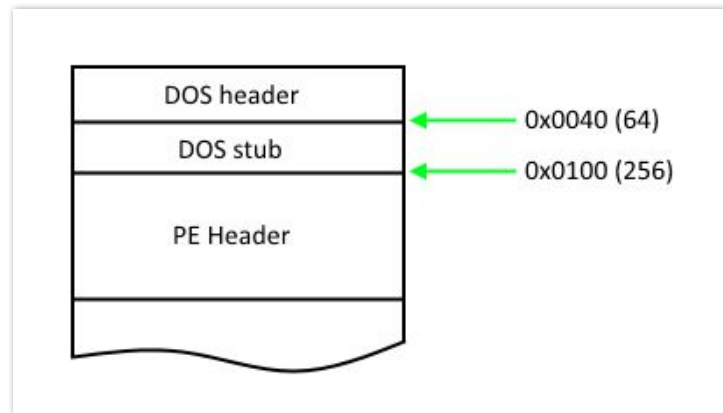
▢ ***e_magic*** == MZ.

▢ MZ == Mark Zbikowski.



e_lfanew — смещение PE заголовка относительно начала файла

DOS stub

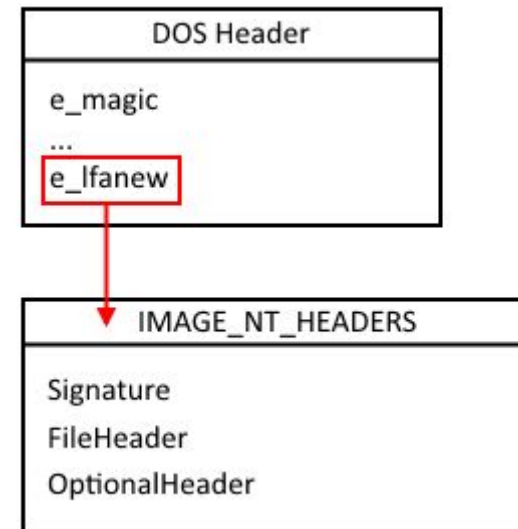


“This program cannot be run in DOS mode”

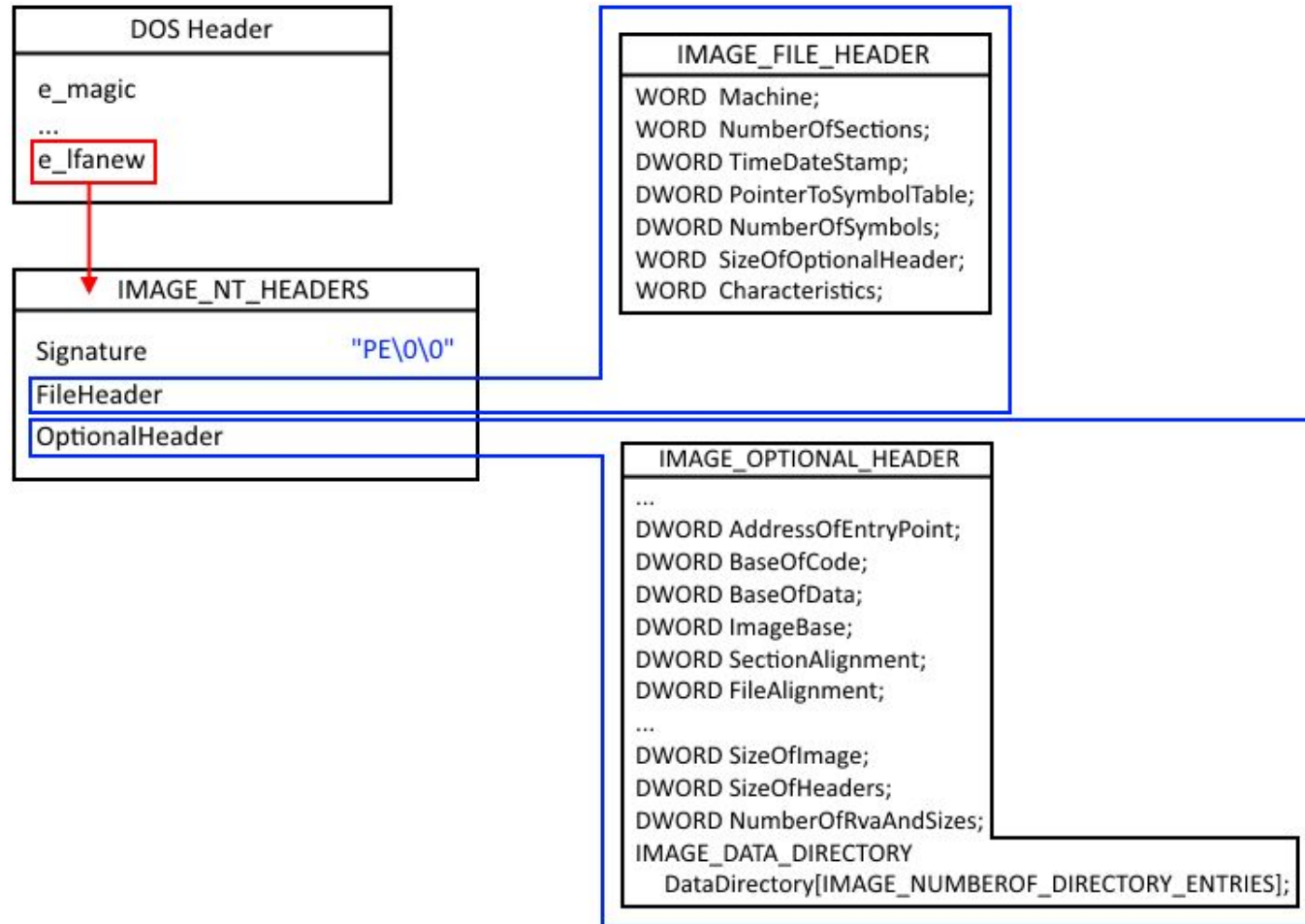
PE-Header

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

- **Signature** == “PE\0\0”
- **FileHeader** содержит базовые характеристики файла
- **OptionalHeader** содержит информацию, необходимую для загрузки файла



PE-Header



File-Header

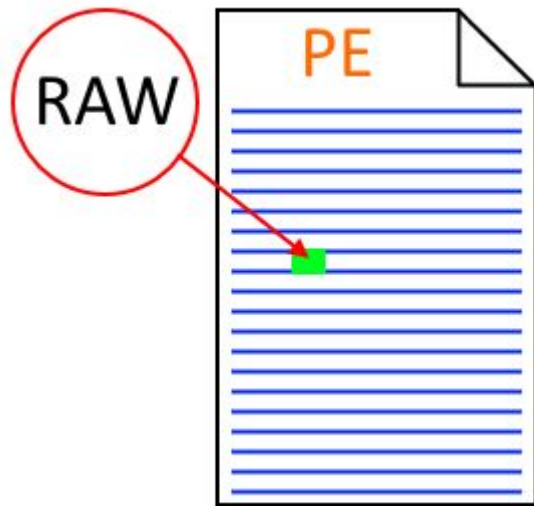
COFF - Common Object File Format

```
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;
```

- ▢ *Machine*: идентификатор архитектуры процессора, на которой данное приложение может выполняться
- ▢ *NumberOfSections*: DWORD — количество секций в файле
- ▢ *TimeDateStamp*: WORD — дата и время создания файла
- ▢ *PointerToSymbolTable*: RAW-смещение до таблицы символов
- ▢ *NumberOfSymbols*: количество записей в таблице символов
- ▢ *SizeOfOptionalHeader*: размер Optional header
- ▢ *Characteristics*: число с характеристиками образа исполняемого файла. Каждая характеристика устанавливается как соответствующий бит.

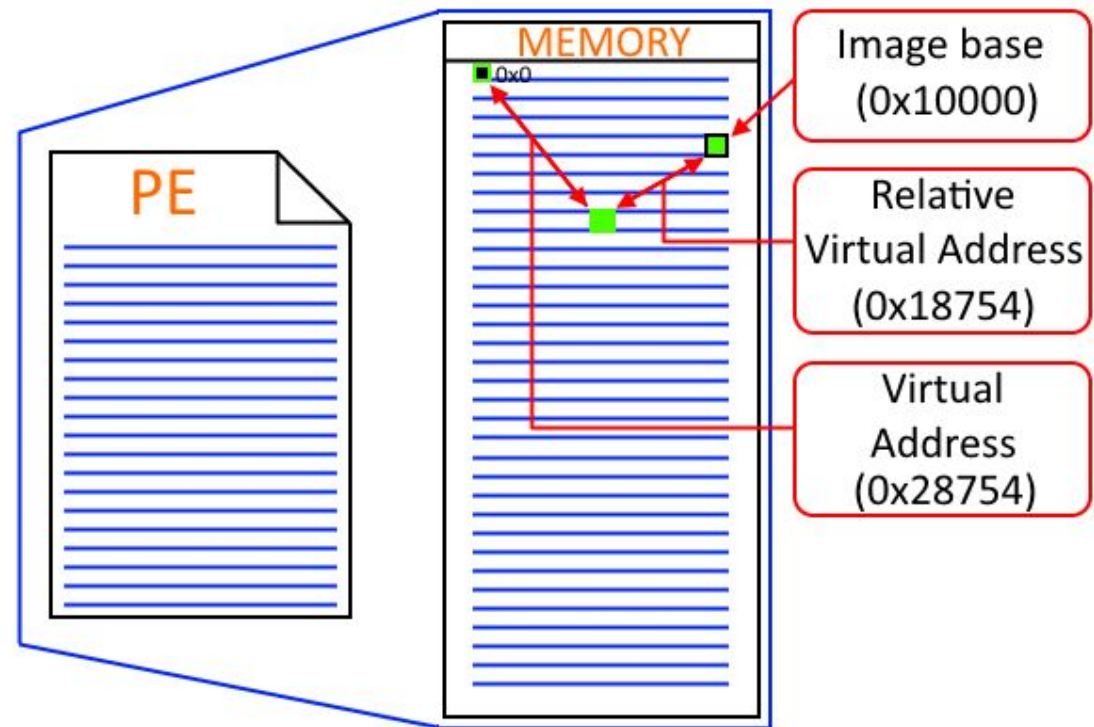
Типы адресации

□ RAW – абсолютный адрес



□ VA – виртуальный адрес

□ RVA – относительный виртуальный адрес



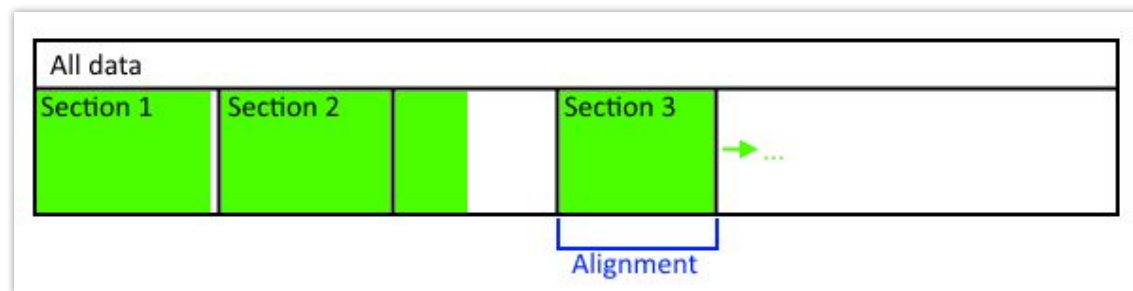
Optional-Header

```
typedef struct _IMAGE_OPTIONAL_HEADER {  
WORD Magic;  
...  
DWORD SizeOfCode;  
...  
DWORD AddressOfEntryPoint;  
DWORD BaseOfCode;  
DWORD BaseOfData;  
DWORD ImageBase;  
DWORD SectionAlignment;  
DWORD FileAlignment;  
...  
DWORD SizeOfImage;  
DWORD SizeOfHeaders;  
...  
DWORD NumberOfRvaAndSizes;  
IMAGE_DATA_DIRECTORY DataDirectory[IMAGE_NUMBEROF_DIRECTORY_ENTRIES];  
} IMAGE_OPTIONAL_HEADER, *PIMAGE_OPTIONAL_HEADER;
```

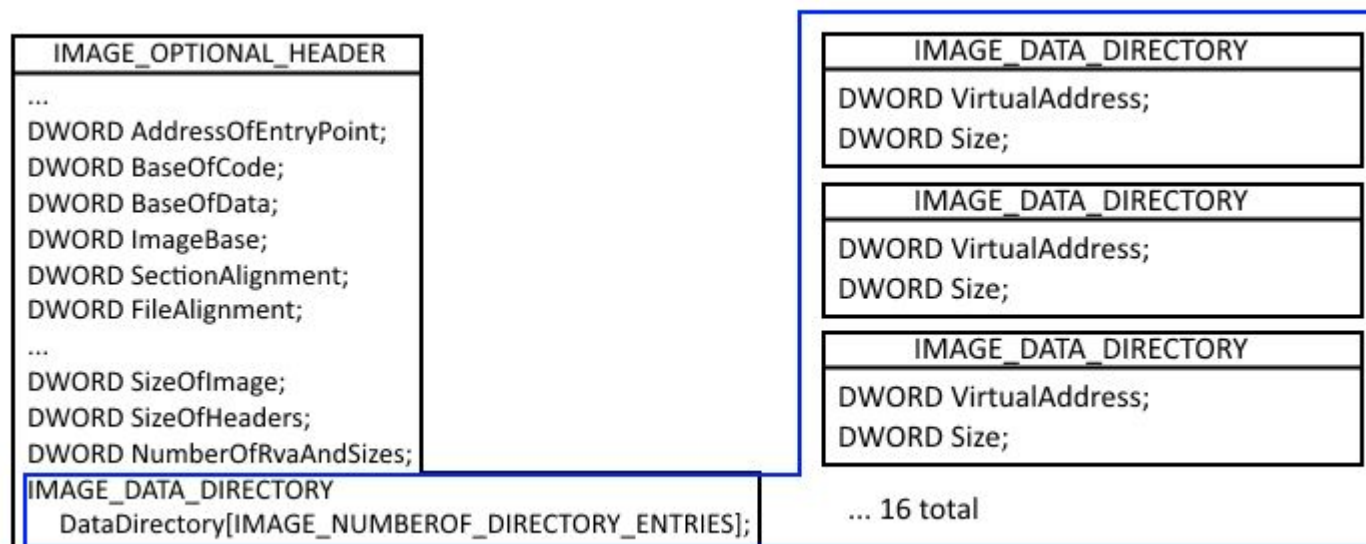
- ▮ *Magic*: имеет разные значения для приложений, собранных для 32 и 64-разрядных систем
- ▮ *SizeOfCode*: общий размер всех секций кода
- ▮ *AddressOfEntryPoint*: RVA-адрес точки входа (функции старта программы)
- ▮ *BaseOfCode*: RVA-адрес на начало секции кода
- ▮ *BaseOfData*: RVA-адрес на начало секции данных
- ▮ **ImageBase**: предпочтительный базовый адрес загрузки программы
- ▮ *SectionAlignment*: размер выравнивания секции при выгрузке в виртуальную память.
- ▮ *FileAlignment*: размер выравнивания секции внутри файла
- ▮ *SizeOfImage*: размер файла в памяти, включая все заголовки. Должен быть кратен *SectionAlignment*.
- ▮ *SizeOfHeaders*: размер всех заголовков (DOS, DOS-Stub, PE, Section) выровненный на *FileAlignment*.
- ▮ *NumberOfRvaAndSizes*: количество каталогов в таблице директорий (ниже сама таблица). Всегда равно константе *IMAGE_NUMBEROF_DIRECTORY_ENTRIES* (16).
- ▮ *DataDirectory* – массив структур, описывающих директории данных разных типов

Выравнивание

```
typedef struct _IMAGE_OPTIONAL_HEADER {  
    ...  
    DWORD SectionAlignment; // размер выравнивания секции при выгрузке в виртуальную память  
    DWORD FileAlignment; // размер выравнивания секции внутри файла  
    ...  
} IMAGE_OPTIONAL_HEADER, *PIMAGE_OPTIONAL_HEADER;
```



Директории данных



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

- VirtualAddress: RVA директории данных
- Size: размер директории данных

Директории данных

// 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

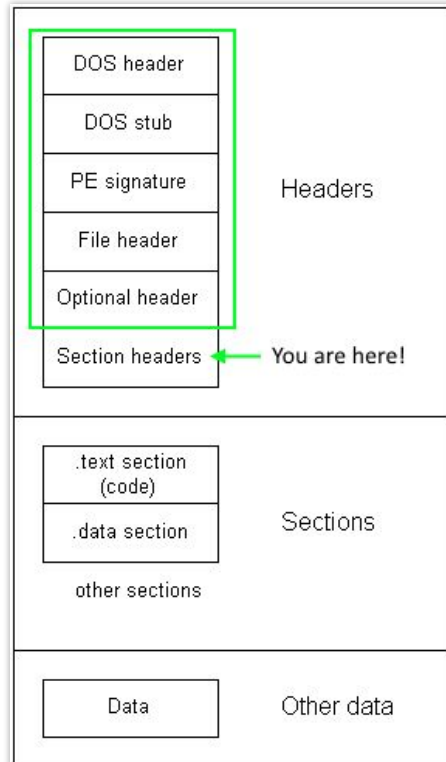
#define IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT 11 // Bound Import Directory in headers

#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

Секции



- Section headers – это массив структур типа `IMAGE_SECTION_HEADER`
- Каждая из структур описывает одну секцию исполняемого файла
- Количество секций определяется полем `NumberOfSections` в `IMAGE_FILE_HEADER`
- Сразу после массива структур `IMAGE_SECTION_HEADER`, описывающих секции, находятся сами секции

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;
```

- ▢ *Name*: имя секции, IMAGE_SIZEOF_SHORT_NAME == 8
- ▢ *VirtualSize*: размер секции в виртуальной памяти.
- ▢ *VirtualAddress*: RVA секции.
- ▢ *SizeOfRawData*: размер секции в файле. Должен быть кратен *FileAlignment*
- ▢ *PointerToRawData*: RAW смещение до начала секции. Также должен быть кратен *FileAlignment*
- ▢ *Characteristics*: битовые атрибуты доступа к секции и правила для её загрузки в виртуальную память

Вопросы?

