

**دانشکده مهندسی کامپیوتر**

**گروه مهندسی نرم افزار**

عنوان پروژه:

بررسی ساختار فایل های PE

پیش نویس اول پروژه کلاسی **شماره 3** درس کامپایلر پیشرفته

دانشجویان:

**مرتضی ذاکری**

استاد:

**دکتر سعید پارسا**

پاییز 1395

**فهرست مطالب**

[1 مقدمه 1](#_Toc468394438)

[2 ساختار کلی 1](#_Toc468394439)

[2ـ1 DOS Header 4](#_Toc468394440)

[2ـ2 PE File Header 4](#_Toc468394441)

[2ـ3 Optional Header 4](#_Toc468394442)

[2ـ3ـ1 Data Directory 4](#_Toc468394443)

[2ـ4 Section Table 5](#_Toc468394444)

[2ـ5 Section ها 5](#_Toc468394445)

[2ـ5ـ1 برخی از Sectionهای رایج 5](#_Toc468394446)

[3 مطالعه موردی فایل PE یک برنامه ساده 7](#_Toc468394447)

[3ـ1 معرفی برنامه 7](#_Toc468394448)

[3ـ2 کد برنامه و فایل اجرایی 7](#_Toc468394449)

[3ـ3 برنامه نویسی نمایش ساختار فایل PE 9](#_Toc468394450)

[3ـ4 بررسی فایل اجرایی برنامه اعداد اول 10](#_Toc468394451)

[3ـ5 تفاوت نسخه های 32 بیتی و 64 بیتی فایل های PE 12](#_Toc468394452)

[3ـ6 نتیجه گیری 13](#_Toc468394453)

[4 منابع و ماخذ 13](#_Toc468394454)

**فهرست شکل ها**

[شکل ‏2‑1ساختار کلی فایل PE 2](#_Toc468394565)

[شکل ‏2‑2 ساختار فایل PE با جزئیات بیش تر 3](#_Toc468394566)

[شکل ‏2‑3 نمایش HEX از یک فایل PE 6](#_Toc468394567)

[شکل ‏3‑1 کد برنامه تعیین عدد اول 8](file:///D:\_04_3l\l3_university\Master\semester1_901\AdvancedCompiler\Projects\PortableEXEFormat\PEReport.docx#_Toc468394568)

[شکل ‏3‑2 ساختار فایل PE در ابزار NikPEViewer 10](#_Toc468394569)

[شکل ‏3‑3 بخش DOS Header 11](#_Toc468394570)

[شکل ‏3‑4 نمایش دودویی بخش DOS Header 11](#_Toc468394571)

[شکل ‏3‑5 پیدا کردن ثابت های رشته ای برنامه در فایل دودویی 12](#_Toc468394572)

# مقدمه

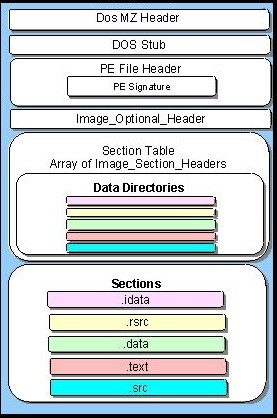
فایل PE یا Portable Executable قالب فایلی است که در سیستم ها عامل ویندوز به کار می رود (در هر دو نسخه 32 و 64 بیتی). PE قالب استاندارد شده ای برای فایل هایی با پسوند های رایج زیر است:

.exe, .dll, .obj, .acm, .ax, .cpl, .drv, .efi, …

در واقع PE داده ساختاری است شامل اطلاعات لازم برای ماژول بارکننده برنامه در سیستم عامل که نهایتا فایل را برای اجرا شدن در حافظه قرار می دهد. قبل از ابداع این قالب، قالبی به نام COFF وجود داشت که در سیستم های Windows NT استفاده می شد.

# ساختار کلی

اساسا فایل PE از دو بخش تقسیم شده است که هریک دارای زیر بخش های مختلفی می باشد. بخش اول Header و بخش دیگر Section نام دارد. شکل ‏2‑1 بخش های اصلی و زیر بخش های درون هریک را به صورت گرافیکی نشان می دهد.



شکل ‏2‑1ساختار کلی فایل PE

در شکل ‏2‑2 جزئیات بیش تری را می توان مشاهده کرد.



شکل ‏2‑2 ساختار فایل PE با جزئیات بیش تر

## DOS Header

DOS Header یا DOS MZ Header 64 بایت اول هر فایل PE را تشکیل می دهد. این قسمت توسط بارکننده DOS بررسی می شود و درصورتی که معتبر باشد برنامه DOS STUB اجرا می شود. اما هنگامی که فایل PE حاوی برنامه Win32 باشد، در بخش DOS Header پرش به ابتدای PE File Header انجام شده و DOS STUB نیز حاوی کد چاپ پیام زیر می شود.

"This program cannot run in DOS mode"

بنابراین هنگامی که فایل PE در محیط DOS اجرا شود، با ورود بخش حاوی برنامه کوچک DOS STUB پیام فوق به کاربر نمایش داده می شود.

## PE File Header

شامل اطلاعاتی مانند نوع پردازنده، تعداد کل Section های فایل PE جاری، مهر زمان (TimeStamp) و غیره است. آغاز این قسمت با فیلد موسوم به PE Signature است که محتوی کد اسکی 45 00 یا همان عبارت PE 00 می باشد.

## Optional Header

علی رقم وجود واژه اختیاری در نام این قسمت، اطلاعات مهمی در آن قرار می گیرند. چند قلم از این اطلاعات عبارتند از: MajorLinkerVersionو MinorLinkerVersion شماره نسخه برنامه Linker که این فایل را ایجاد کرده است. SizeOfCode اندازه همه Section های حاوی کد برنامه. AddressOfEntryPoint آدرسی که بار کننده اجرا را از آنجا آغاز می کند. این فیلد حاوی ادرس مجازی نسبی یه همان RVA می باشد که به محلی در بخش .text اشاره می کند و ... .

### Data Directory

یک زیر بخش مهم در Optional Header فیلدی تحت عنوان Data Directory است. این بخش در واقع آرایه ای از ساختارهای IMAGE\_DATA\_DIRECTORY است. هر ساختار آدرس مجازی نسبی، یک ساختار داده مهم در فایل PE همچون جدول آدرس واردات فایل (import address table) را ارائه می دهد.

## Section Table

بین بخش PE Header و Section ها این قسمت قرار گرفته است. Section Table در واقع شبیه یک دفترچه تلفن است که اطلاعاتی راجع به هر Section موجود در تصویر حافظه[[1]](#footnote-1) برنامه را در بردارد. Section ها بر اساس ادرس شروع مجازی خود در حافظه مرتب می شوند نه عناوینشان.

## Section ها

تعداد و نام Sectionها در فایل های PE می تواند مختلف باشد. اما برای هر Section اطلاعات یکسانی باید نگاهداری گردد. فایل EXE در مورد Sectionهایش اطلاعات نظیر آدرس فیزیکی، آدرس مجازی، اندازه اطلاعات و نوع دسترسی های موجود از قبیل خواندن، نوشتن و اجرا را نگهداری می کند.

### برخی از Sectionهای رایج

در این قسمت تعدادی از Sectionهایی را که در فایل های .exe و .obj وجود دارند معرفی می کنیم. معمولا عنوان هر Section با یک نطقه آغاز می شود ولی الزامی برای این کار وجود ندارد، یعنی می تواند بدون نقطه هم باشد.

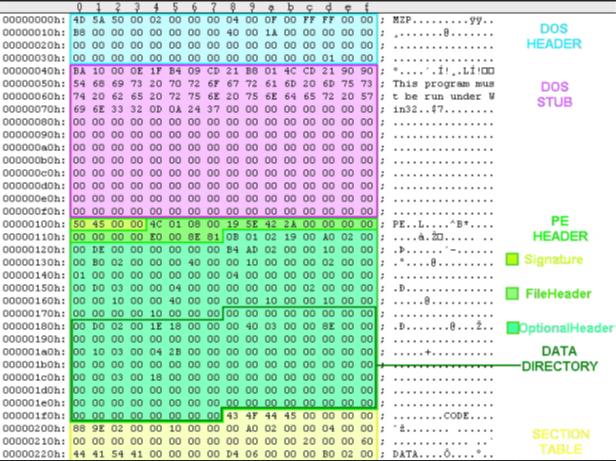
**.text** این Section جایی است که همه کدهای برنامه که توسط کامپایلر یا اسمبلر تولید شده است نوشته می شود.

**.rsrc** شامل تمام ماژول های به کار رفته در برنامه است.

**.idata** شامل داده ها و تابع هایی است که از ماژول های DLL دیگر در این فایل درج شده است.

و ... .

آن چه گفته شده توضیح بسیار مختصری از اجزای موجود در ساختار فایل PE بود و وارد بسیاری از جزئیات نشدیم. شکل ‏2‑3 نمایش یک فایل PE به صورت کد HEX می باشد که بخش های گفته شده در آن مشخص شده اند. همان طور که پیداست این پایین ترین سطح نمایشی از یک برنامه و در واقع نمایش در سطح ماشین است (صفر و یک). در بخش سوم این نوشتار یک فایل PE را در عمل بررسی می کنیم.



شکل ‏2‑3 نمایش HEX از یک فایل PE

# مطالعه موردی فایل PE یک برنامه ساده

## معرفی برنامه

در این قسمت یک برنامه بسیار کوچک به زبان C++ در Win32 می نویسیم و سپس آن را کامپایل می کنیم تا فایل اجرایی در قالب EXE تولید شود. در ادامه ساختار فایل EXE تولید شده را که از قالب فایل های PE پیروی می کند بررسی می کنیم تا موارد شرح داده شده در بخش 2 این نوشتار را به صورت عملی مشاهده نماییم.

دلیل آن که خود به نوشتن یک برنامه مبادرت ورزیدیم این است که از ساختار و عملکرد بیرونی کد اطلاع داشته باشیم و بتوانیم نتایج مشاهده شده را با آن چه بایستی وجود داشته باشد تطابق دهیم. به عبارت دیگر می خواهیم ساختار زمان اجرای یک برنامه کاربردی کوچک را که قرار است توسط بارکننده های ویندوزی بارگذاری شود با آگاهی و تسلط بر نحوه عملکرد و محتوای کد منبع و منابع مورد استفاده آن، مشاهده و بررسی نماییم. مینیمال بودن برنامه این امکان را به ما می دهد تا با پیچیدگی های ناشی از بزرگ بودن برنامه مواجه نشویم و بر روی هدف اصلی که شناخت ساختار فایل PE است تمرکز داشته باشیم.

برنامه ای که در این جا نوشته ایم در واقع یک عدد را از کنسول ورودی می خواند و تعیین می کند که اول است یا خیر و در نهایت پیام مناسب صادر می کند. برنامه شامل دستورات شرطی، حلقه تکرار، فراخوانی تابع، ثابت های رشته ای و دیگر جنبه های برنامه نویسی است که به نحوی که بتواند خیلی خوب ما را در یک روند آموزشی مناسب هدایت کند.

## کد برنامه و فایل اجرایی

کد برنامه به صورت شکل ‏3‑1 است. برنامه در محیط Microsoft Visual Studio 2015 نوشته و کامپایل شده است. دو نسخه اجرایی 64 و 32بیتی از برنامه ایجاد کرده ایم تا بتوانیم تفاوت های ساختار فایل های PE در نسخه های مختلف را بررسی کنیم.

شکل ‏3‑1 کد برنامه تعیین عدد اول

#include <iostream>

using namespace std;

void checkPrime(int n)

{

int flag = 1;

for (int i = 2; i\*i < n && flag; i++)

{

if (!(n%i))

{

flag = 0;

}

}

if (!flag)

{

cout << "Is Not Prime!\n";

}

else

{

cout << "Is Prime!\n";

}

}

int main()

{

int n;

cout << "Enter Number: ";

cin >> n;

checkPrime(n);

system("pause");

return 0;

}

## برنامه نویسی نمایش ساختار فایل PE

فایل PE یک محتوی یک کد دودویی است بنابراین از نوع متنی نیست و نمی توان با ویراستار های متنی آن را گشود و ساختار آن را مشاهده کرد. اما می توان آن را با یک ویرایشگر فایل های HEX یا همان باینری گشود. برنامه های بسیاری هستند که برای نمایش ساختار فایل PE استفاده می شوند و علاوه بر نمایش مقادیر دودویی اجزا را به صورت طبقه بندی شده نشان داده و اطلاعات را از پیکره فایل PE بیرون می کشند. این برنامه ها عمدتا در امکاناتی نظیر جست و جو و واسط کاربر تفاوت هایی دارند. ما فهرستی از برنامه ها را تست کردیم تا گزینه بهتر را انتخاب کنیم؛ از جمله:

PE Explorer, PEBrowse64, NikPEViewer, x64/32dbg, …

کلیه اجزای موجود در فایل های PE توسط ساختارهای برنامه نویسی زبان C/C++ تعریف می شوند. برای نمونه بخش DOS Header به صورت زیر در زبان C تعریف می شود، این ساختار در داخل فایل سرایند WINNT.H وجود دارد.

typedef struct \_IMAGE\_DOS\_HEADER { // DOS .EXE header

USHORT e\_magic; // Magic number

USHORT e\_cblp; // Bytes on last page of file

USHORT e\_cp; // Pages in file

USHORT e\_crlc; // Relocations

USHORT e\_cparhdr; // Size of header in paragraphs

USHORT e\_minalloc; // Minimum extra paragraphs needed

USHORT e\_maxalloc; // Maximum extra paragraphs needed

USHORT e\_ss; // Initial (relative) SS value

USHORT e\_sp; // Initial SP value

USHORT e\_csum; // Checksum

USHORT e\_ip; // Initial IP value

USHORT e\_cs; // Initial (relative) CS value

USHORT e\_lfarlc; // File address of relocation table

USHORT e\_ovno; // Overlay number

USHORT e\_res[4]; // Reserved words

USHORT e\_oemid; // OEM identifier (for e\_oeminfo)

USHORT e\_oeminfo; // OEM information; e\_oemid specific

USHORT e\_res2[10]; // Reserved words

LONG e\_lfanew; // File address of new exe header

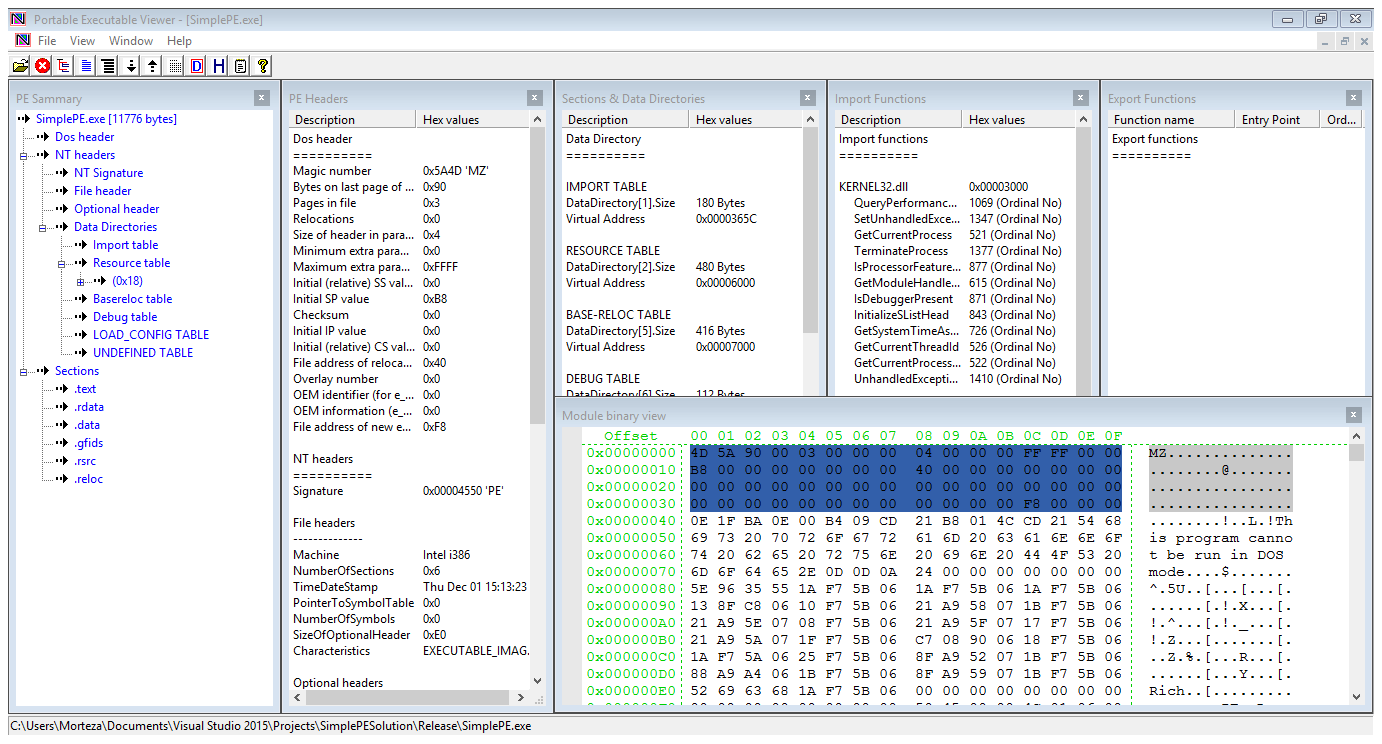
} IMAGE\_DOS\_HEADER, \*PIMAGE\_DOS\_HEADER;

\*\*\*

بدیهی است که API ویندوز توابعی را برای کار با این داده ساختار ها تدارک دیده است. برنامه های نمایش ساختار PE که در بالا به برخی از آن ها اشاره شد، از چنین توابعی استفاده می کنند. بنابراین نوشتن یک برنامه که بتواند نمایش صحیحی از فایل PE ارایه دهد کار ساده ای است.

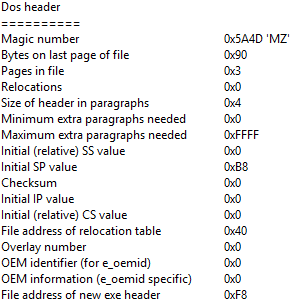
## بررسی فایل اجرایی برنامه اعداد اول

فایل با عنوان SimplePE\_x86.exe را توسط NikPEViewer باز می کنیم (شکل زیر). همان طور که پیداست، پنجره های زیر هر کدام بخش مجزایی از ساختار را نشان می دهند.

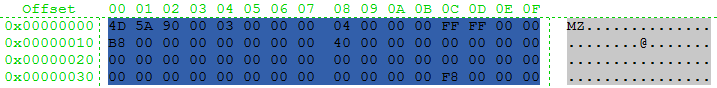


شکل ‏3‑2 ساختار فایل PE در ابزار NikPEViewer

بخش DOS MZ Header خالی است و تنها یک دستور پرش به F8 که ابتدای PE File Header است دارد. اطلاعات شکل ‏3‑3 از شکل ‏3‑4 که معادل دودویی آن است بیرون کشیده شده است.



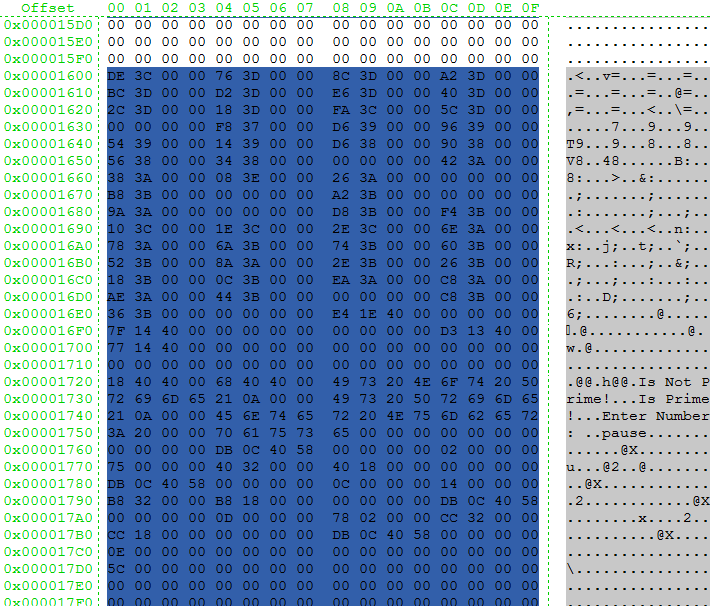
شکل ‏3‑3 بخش DOS Header



شکل ‏3‑4 نمایش دودویی بخش DOS Header

به همین ترتیب می توان سایر قسمت های بخش دوم این نوشتار را مشاهده کرد. به عنوان مثال کد برنامه در .text و ثابت های رشته ای در .rdata تعریف شده اند. هنگامی که برنامه اجرا می شود، پیام "Enter Number:" روی کنسول به نمایش در می آید. می خواهیم محل این پیام را در فایل باینری بیابیم. در منوی سمت چپ روی .rdata دوبار کلیک می کنیم تا محدوده مربوط به این Section در کد باینری یافت شود. حال این محدوده را مشاهده می کنیم.

چنان چه مشاهده می شود کلیه ثابت های رشته ای برنامه در قالب کد اسکی و به صورت متوالی ذخیره شده اند. ادرس محل ذخیره موجودیت غیر فعال (کد برنامه) نیز در این قسمت وجود دارد.



شکل ‏3‑5 پیدا کردن ثابت های رشته ای برنامه در فایل دودویی

## تفاوت نسخه های 32 بیتی و 64 بیتی فایل های PE

قالب فایل های اجرایی در ویندوزهای 64 بیتی (x64) نیز مانند ویندوز 32 بیتی (x86) همان قالب فایل PE است، اما تفاوت هایی دارد. در این جا کشف این تفاوت ها یک نسخه 64 بیتی از برنامه بررسی اعداد اول که در قسمت قبل دیدیم را ایجاد می کنیم. سپس آن را در محیط ابزار NikPEViewer باز می کنیم.

اولین تفاوت در حجم نسخه های 32 بیتی و 64بیتی است. نسخه 64 بیتی ما حدودا 1 کیلوبایت از نسخه 32 بیتی، بزرگتر است (13 کیلوبایت در مقابل 12 کیلوبایت).

آفست شروع PE Header File متفاوت است (0x00000100). آدرس های مجازی 64 بیتی هستند. به طور کلی اندازهSection ها و مقدار RVA ها نیز متفاوت است.

## نتیجه گیری

از ابزار NikPEViewer تفاوت های بیش تری را نمی توان تشخیص داد. با استفاده از ابزار IDA Pro مشاهده می شود از آن جایی که برنامه از نوع Win32 می باشد لذا در نسخه 64 بیتی نیز همچنان از ثبات های 32 بیتی استفاده شده است و از این لحاظ مزیتی ندارد. می توان این نتیجه را گرفت که با وجود تفاوت های جزیی ساختار کلی همان ساختار PE است و در واقع برای به حداکثر رسانی قابلیت حمل این فایل ها تفاوت اساسی در آن ها ایجاد نشده است.

# منابع و ماخذ

1. Image [↑](#footnote-ref-1)