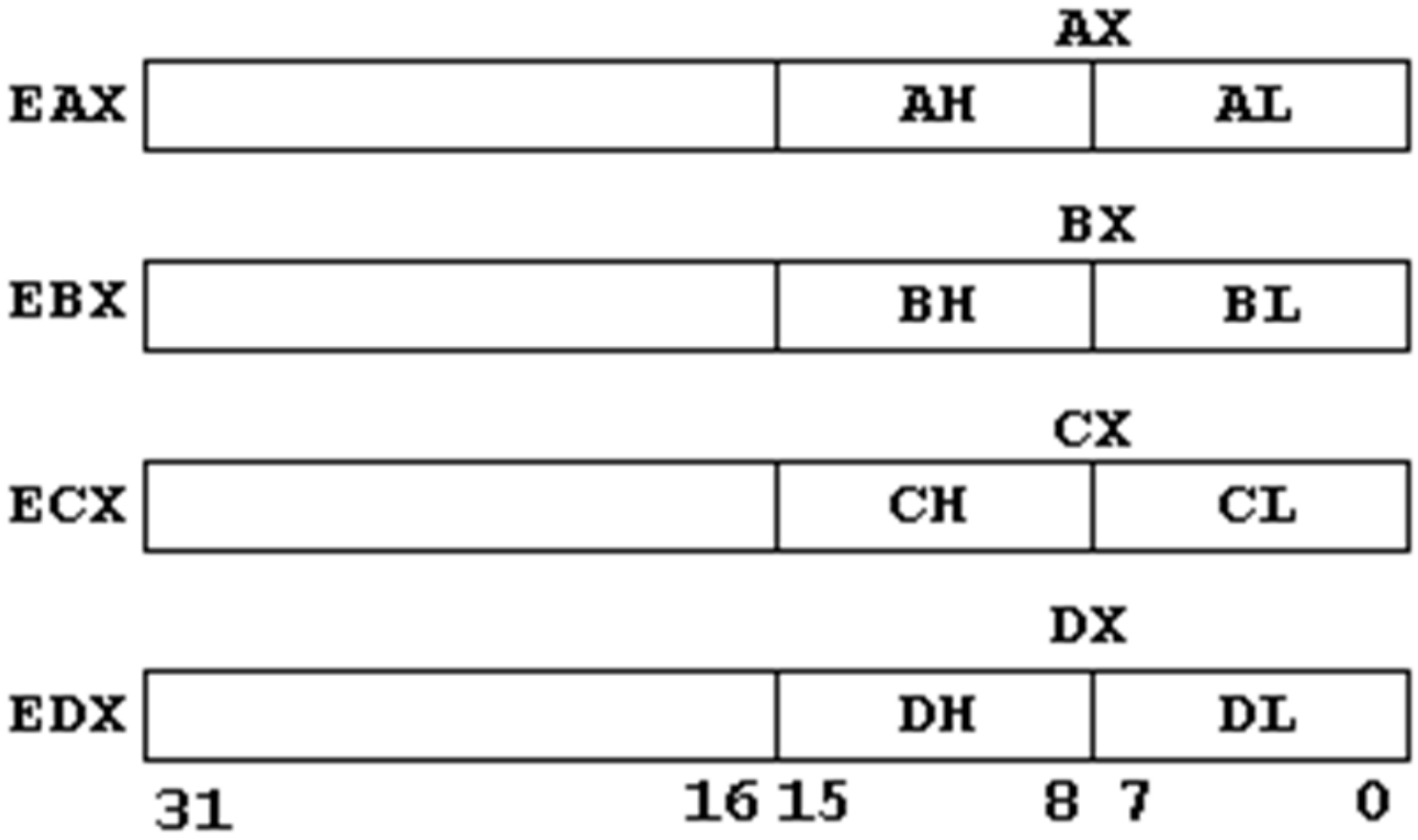
Архитектура x86

- CISC-архитектура

- Сегментная организация памяти
  - Реальный режим работы
  - Защищенный режим работы

Регистры общего назначения



Регистры общего назначения

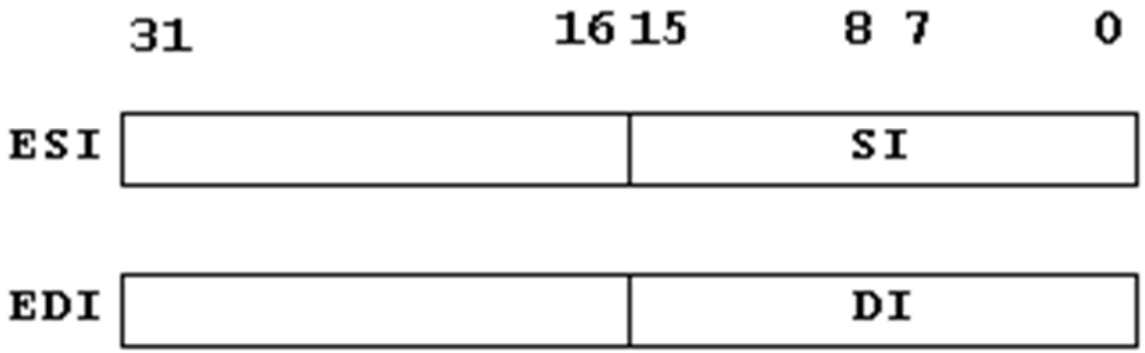
- ✓ **eax/ax/ah/al** (Accumulator register) — аккумулятор.
  Применяется для хранения промежуточных данных. В некоторых командах использование этого регистра обязательно;

- ✓ **ebx/bx/bh/bl** (Base register) — базовый регистр.
  Применяется для хранения базового адреса некоторого объекта в памяти;

- ✓ **ecx/cx/ch/cl** (Count register) — регистр-счетчик.
  Применяется в командах, производящих некоторые повторяющиеся действия. Его использование зачастую неявно и скрыто в алгоритме работы соответствующей команды.

- ✓ **edx/dx/dh/dl** (Data register) — регистр данных.
  Так же, как и регистр eax/ax/ah/al, он хранит промежуточные данные. В некоторых командах его использование обязательно; для некоторых команд это происходит неявно.

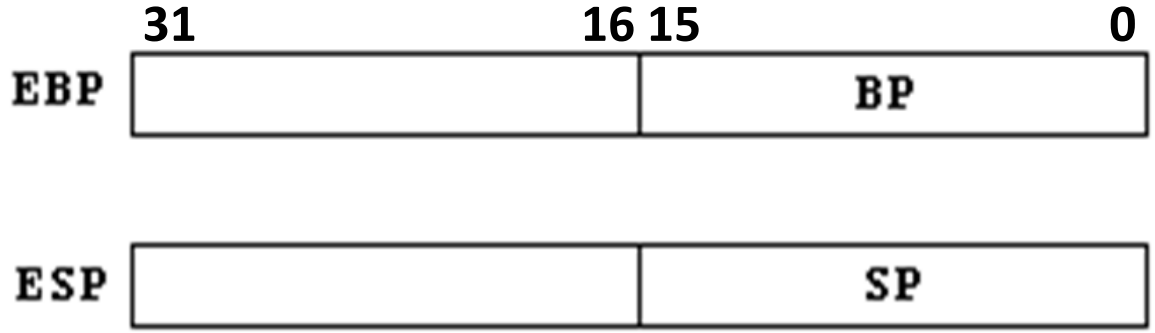
Индексные регистры



Предназначены для хранения индексов при работе с массивами

- ✓ **esi/si** (Source Index register) — индекс источника
- ✓ **edi/di** (Destination Index register) — индекс приемника (получателя)

Регистры-указатели



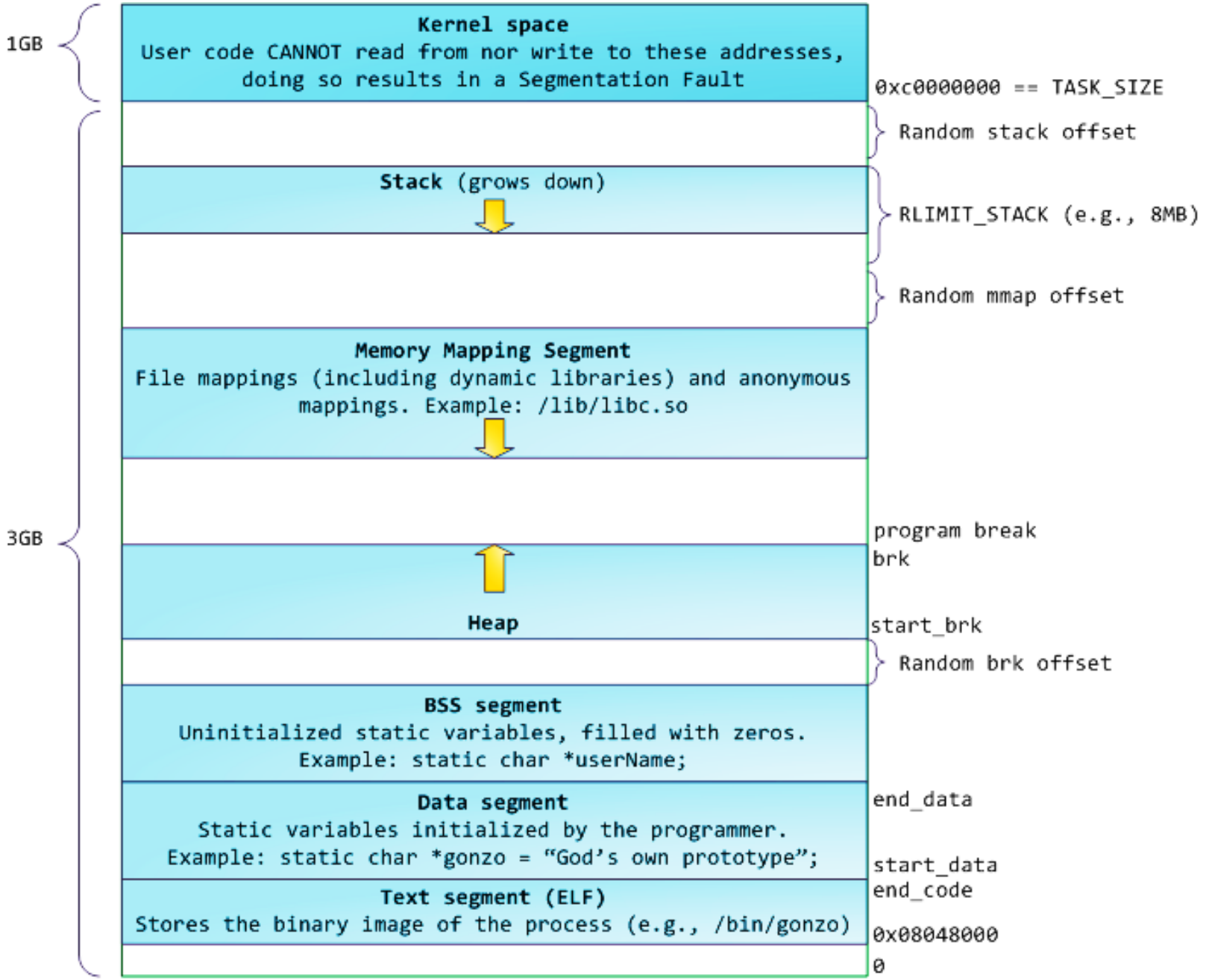
- ✓ **esp/sp** (Stack Pointer register) указывает на вершину стека

- ✓ **ebp/bp** (Base Pointer register) — регистр указателя базы кадра стека

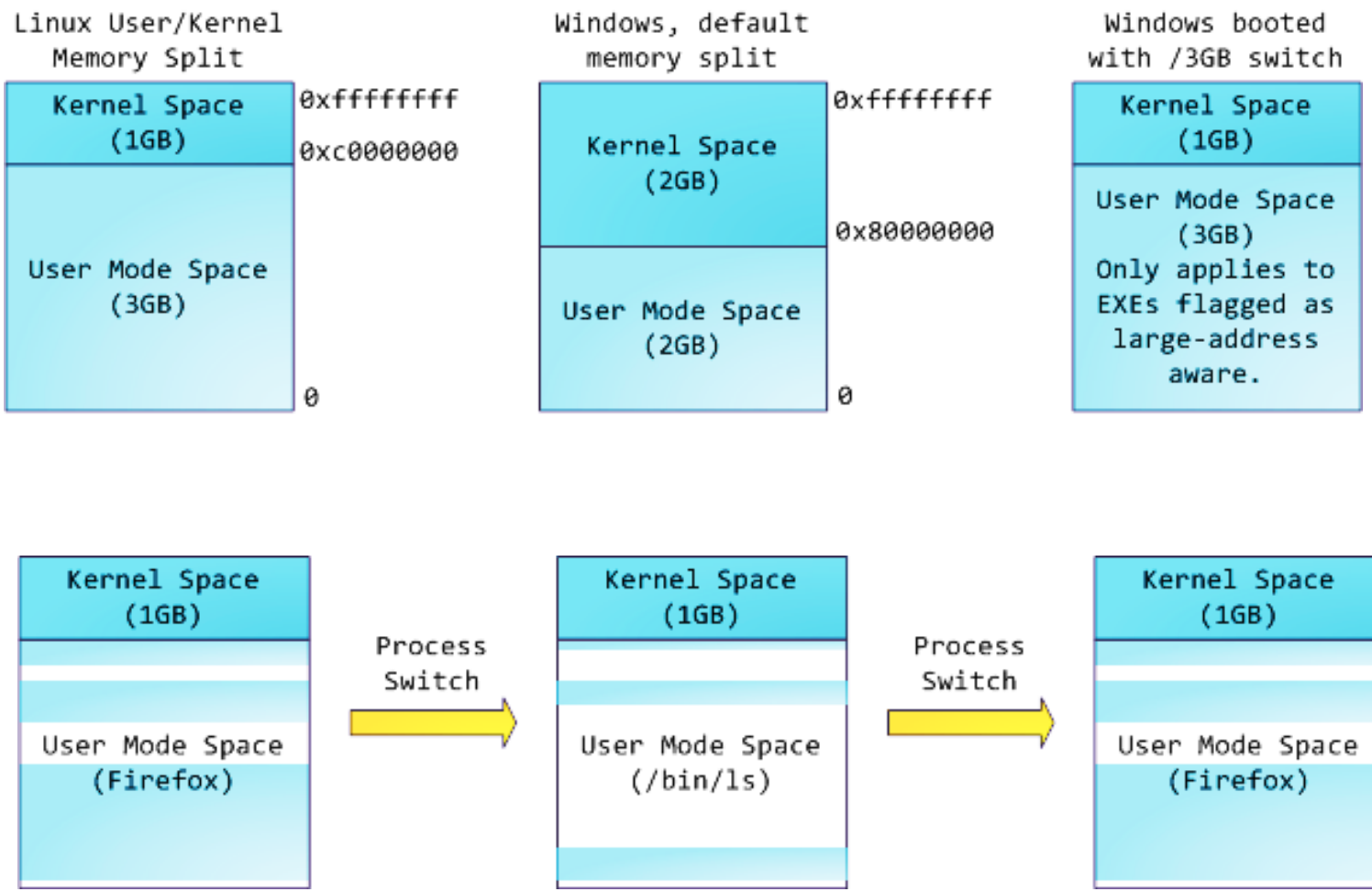
Сегментные регистры

- **CS** (Code Segment) – регистр сегмента кода, содержит начальный адрес сегмента кода
- **DS** (Data Segment) – регистр сегмента данных, содержит его начальный адрес
- **SS** (Stack Segment) – регистр сегмента стека, содержит начальный адрес в сегменте стека
- **ES** (Extra Segment) – регистр сегмента расширения

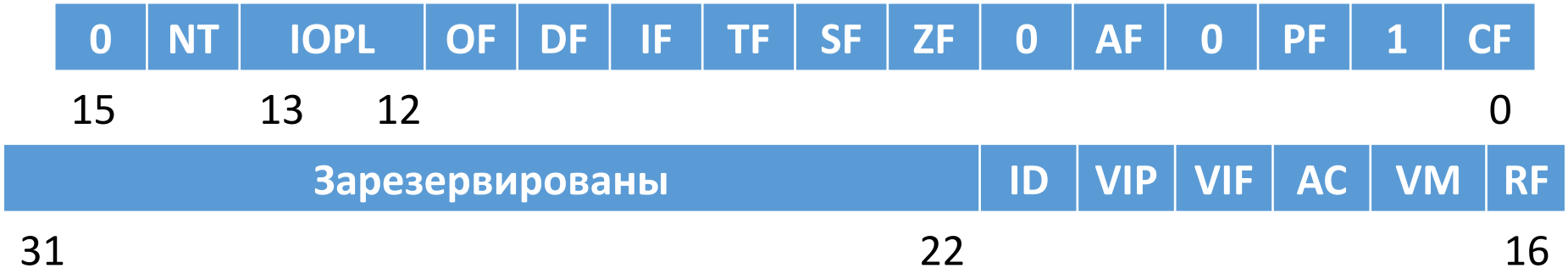
Виртуальное адресное пространство процесса



Виртуальное адресное пространство



Регистр флагов (EFLAGS/FLAGS)



Флаги состояния

CF	устанавливается при переносе из/заёме в (при вычитании) старший значащий бит результата и показывает наличие переполнения в беззнаковой целочисленной арифметике
PF	устанавливается, если младший значащий байт результата содержит чётное число ненулевых битов
AF	устанавливается при переносе из/заёме из бита 3 результата
ZF	устанавливается, если результат равен нулю
SF	равен значению старшего значащего бита результата, который является знаковым битом в знаковой арифметике
OF	устанавливается, если целочисленный результат слишком длинный для размещения в целевом операнде

Регистр флагов (EFLAGS/FLAGS)

Управляющий флаг	
DF	управляет строковыми инструкциями

Системные флаги	
IF	обнуление этого флага запрещает отвечать на маскируемые запросы на прерывание
TF	установка этого флага разрешает пошаговый режим отладки
IOPL	показывает уровень приоритета ввода-вывода
NT	устанавливается, когда текущая задача «вложена» в другую, прерванную задачу
RF	флаг маскирования ошибок отладки
VM	установка этого флага в защищённом режиме вызывает переключение в режим 8086
AC	включение контроля выравнивания операндов при обращениях к памяти
VIF	виртуальная копия флага IF
VIP	устанавливается для указания наличия отложенного прерывания
ID	поддержка инструкции CPUID

Указатель команд (EIP/IP)

Содержит смещение следующей подлежащей выполнению команды относительно содержимого сегментного регистра CS в текущем сегменте команд

Архитектура x86_64

x64 x86_64 AMD64

Два режима работы:

- Long mode («длинный» режим)
- Legacy mode («унаследованный», режим совместимости с x86)

Intel® 64 and IA-32 Architectures Software Developer's Manual

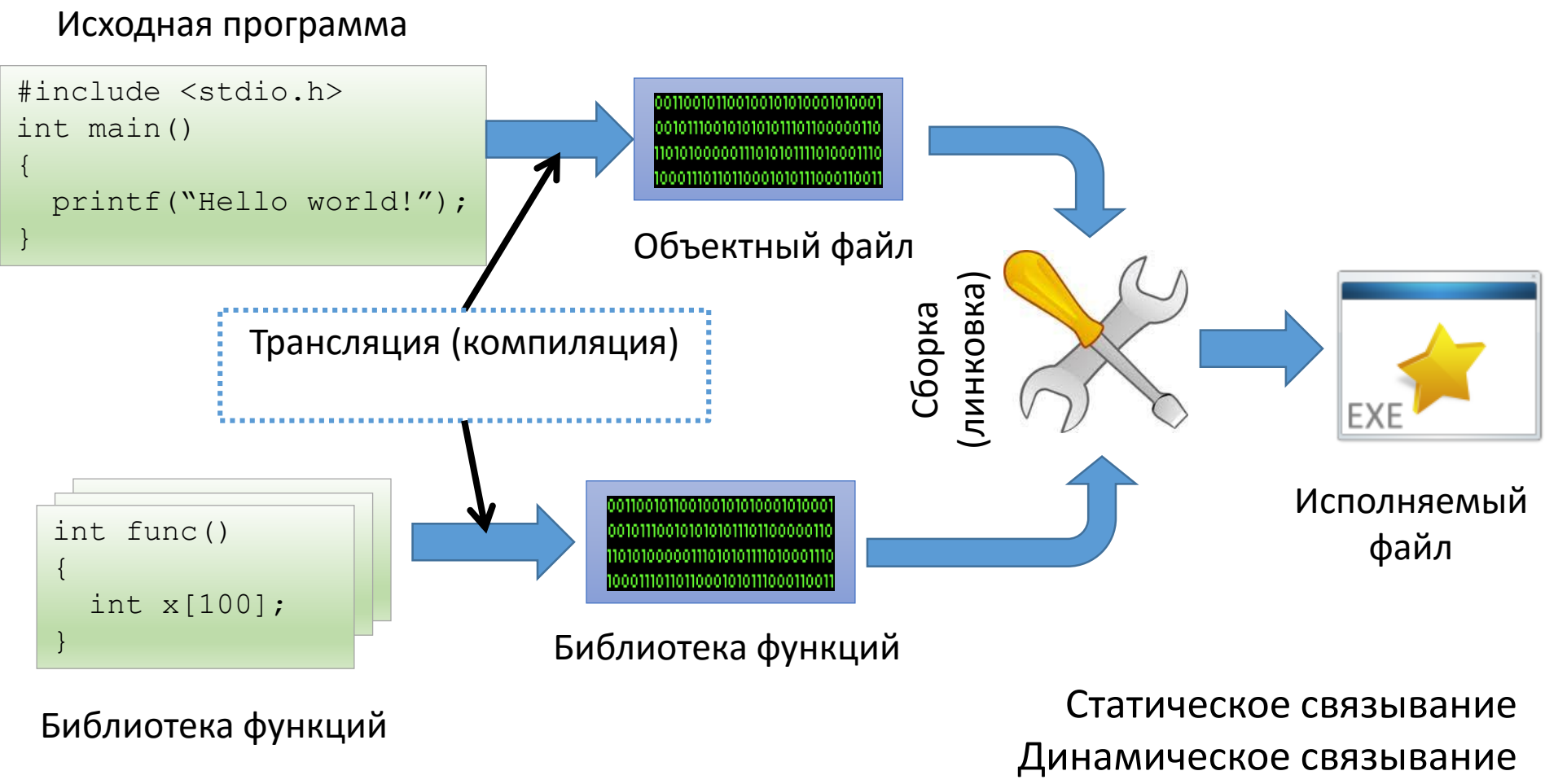
Регистры

- 16 64-разрядных регистров общего назначения
  (RAX, RDX, RCX, RBX, RSI, RDI, RSP, RBP, R8-R15)
- 16 128-разрядных регистров SSE (XMM0-XMM15)
- 8 80-битных регистров с плавающей точкой (ST0-ST7, доступны в режиме MMX/3DNow! как 64-разрядные регистры)
- 16 256-разрядных регистров AVX (YMM0-YMM15)
- 64-битный регистр RIP и 64-битный регистр флагов RFLAGS

Программная модель x86_64

	32-битный режим	64-битный режим
Общее адресное пространство процесса	4 Гб	16 Тб
Адресное пространство, доступное 32-битному процессу	2Гб (3Гб, если система загружена с ключом /3GB)	4Гб, если приложение скомпилировано с ключом /LARGEADDRESSAWARE (2Гб иначе)
Адресное пространство, доступное 64-битному процессу	Невозможно	8Тб

Компиляция и сборка программы

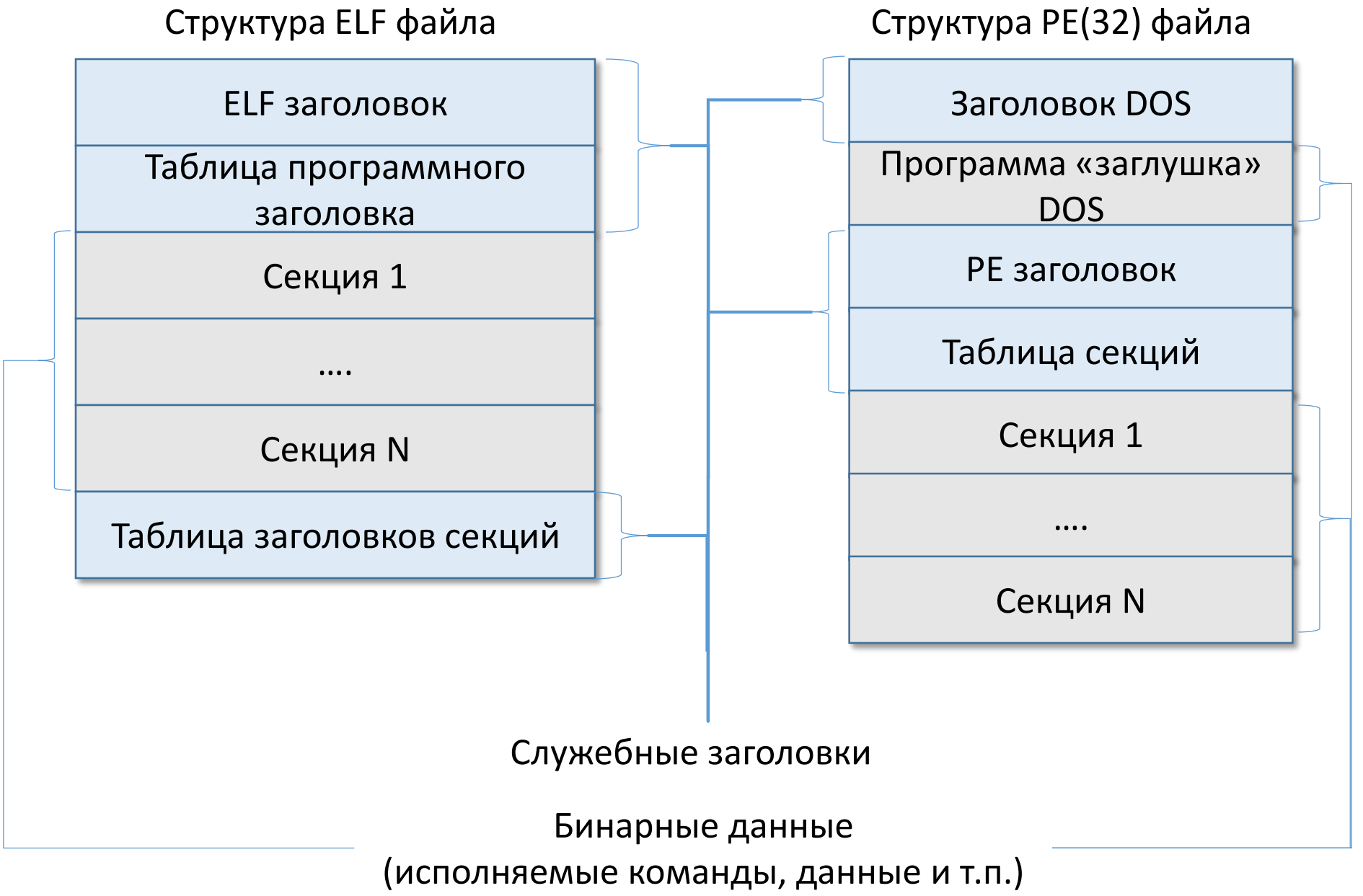


Исполняемый файл

- **Исполняемый файл** (*executable file*) — файл, содержащий программу в виде, в котором она может быть (после загрузки в память и настройки по месту) исполнена компьютером
- Исполняемый файл содержит:
  - ✓ заголовок;
  - ✓ машинные команды;
  - ✓ статические данные;
  - ✓ информацию о стеке;
  - ✓ данные для отладки программы;
  - ✓ и т.п.

COFF, ECOFF, A.OUT и т.д.
EXE (MZ, NE, LE, LX, PE), COM
ELF = Executable and Linkable Format
PE = Portable Executable

Структура исполняемого файла



Заголовок исполняемого файла в формате ELF

```
#include <elf.h>

typedef struct {
  unsigned char e_ident[EI_NIDENT]; /* Сигнатура и прочая информация */
  Elf32_Half e_type;    /* Тип объектного файла */
  Elf32_Half e_machine;/* Аппаратная платформа (архитектура) */
  Elf32_Word e_version;/* Номер версии */
  Elf32_Addr e_entry;/* Адрес точки входа (стартовый адрес программы) */
  Elf32_Off  e_phoff;/*Смещение от начала файла таблицы программных заголовков*/
  Elf32_Off  e_shoff;/*Смещение от начала файла таблицы заголовков секций*/
  Elf32_Word e_flags;/*Флаги процессора (не используется в архитектуре x86)*/
  Elf32_Half e_ehsize;/*Размер ELF-заголовка в байтах*/
  Elf32_Half e_phentsize; /*Размер записи в таблице программных заголовков*/
  Elf32_Half e_phnum;    /*Количество записей в таблице программных заголовков*/
  Elf32_Half e_shentsize; /*Размер записи в таблице заголовков секций*/
  Elf32_Half e_shnum;      /*Количество записей в таблице заголовков секций*/
  Elf32_Half e_shstrndx; /*Расположение сегмента, содержащего таблицу строк*/
} Elf32_Ehdr;
```

‘\x7F’,’E’,’L’,’F’

Утилита readelf

Заголовок исполняемого файла в формате ELF

```
user@user-pc:~/Test/OS$ readelf -a server
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Intel 80386
  Version:                           0x1
  Entry point address:               0x8048410
  Start of program headers:          52 (bytes into file)
  Start of section headers:          4432 (bytes into file)
  Flags:                             0x0
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         9
  Size of section headers:           40 (bytes)
  Number of section headers:         30
  Section header string table index: 27
```

Заголовок исполняемого файла DOS

```
#include <winnt.h>

typedef struct _IMAGE_DOS_HEADER {
    WORD e_magic;         // Magic number
    WORD e_cblp;          // Bytes on last page of file
    WORD e_cp;            // Pages in file
    WORD e_crlc;          // Relocations
    WORD e_cparhdr;       // Size of header in paragraphs
    WORD e_minalloc;      // Minimum extra paragraphs needed
    WORD e_maxalloc;      // Maximum extra paragraphs needed
    WORD e_ss;            // Initial (relative) SS value
    WORD e_sp;            // Initial SP value
    WORD e_csum;          // Checksum
    WORD e_ip;            // Initial IP value
    WORD e_cs;            // Initial (relative) CS value
    WORD e_lfarlc;        // File address of relocation table
    WORD e_ovno;          // Overlay number
    WORD e_res[4];        // Reserved words
    WORD e_oemid;         // OEM identifier (for e_oeminfo)
    WORD e_oeminfo;       // OEM information; e_oemid specific
    WORD e_res2[10];      // Reserved words
    LONG e_lfanew;        // File address of new exe header
} IMAGE_DOS_HEADER;
```

MZ

УКАЗЫВАЕТ НА
‘P’,’E’,’\0’,’\0’

Утилита
PEBrowse

Заголовки исполняемого файла WINDOWS

```
#include <winnt.h>

typedef struct _IMAGE_FILE_HEADER {
    WORD    Machine;
    WORD    NumberOfSections;
    DWORD   TimeDateStamp;
    DWORD   PointerToSymbolTable;
    DWORD   NumberOfSymbols;
    WORD    SizeOfOptionalHeader;
    WORD    Characteristics;
} IMAGE_FILE_HEADER;
typedef struct _IMAGE_OPTIONAL_HEADER {
...
    DWORD   SizeOfCode;
    DWORD   SizeOfInitializedData;
    DWORD   SizeOfUninitializedData;
    DWORD   AddressOfEntryPoint;
    DWORD   BaseOfCode;
    DWORD   BaseOfData;
...
} IMAGE_OPTIONAL_HEADER32;
```