Архитектура ЭВМ

Лекция 3. Что такое x86?

Филонов Павел
filonovpv@gmail.com

Московский Государственный Технический Университет
Гражданской Авиации

2021 г.

Повестка дня

Повестка дня

1. Краткая история процессоров Intel

Повестка дня

1. Краткая история процессоров Intel
2. Архитектура x86

Повестка дня

1. Краткая история процессоров Intel
2. Архитектура x86
3. Что означают следующие надписи: x86-64, i386, IA-32, IA-64?

Повестка дня

1. Краткая история процессоров Intel
2. Архитектура x86
3. Что означают следующие надписи: x86-64, i386, IA-32, IA-64?
4. Кто пишет понятней: AT&T или Intel?

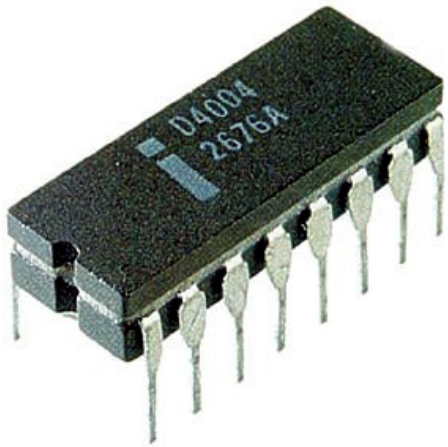
Повестка дня

1. Краткая история процессоров Intel
2. Архитектура x86
3. Что означают следующие надписи: x86-64, i386, IA-32, IA-64?
4. Кто пишет понятней: AT&T или Intel?
5. NASM — наше всё!

Повестка дня

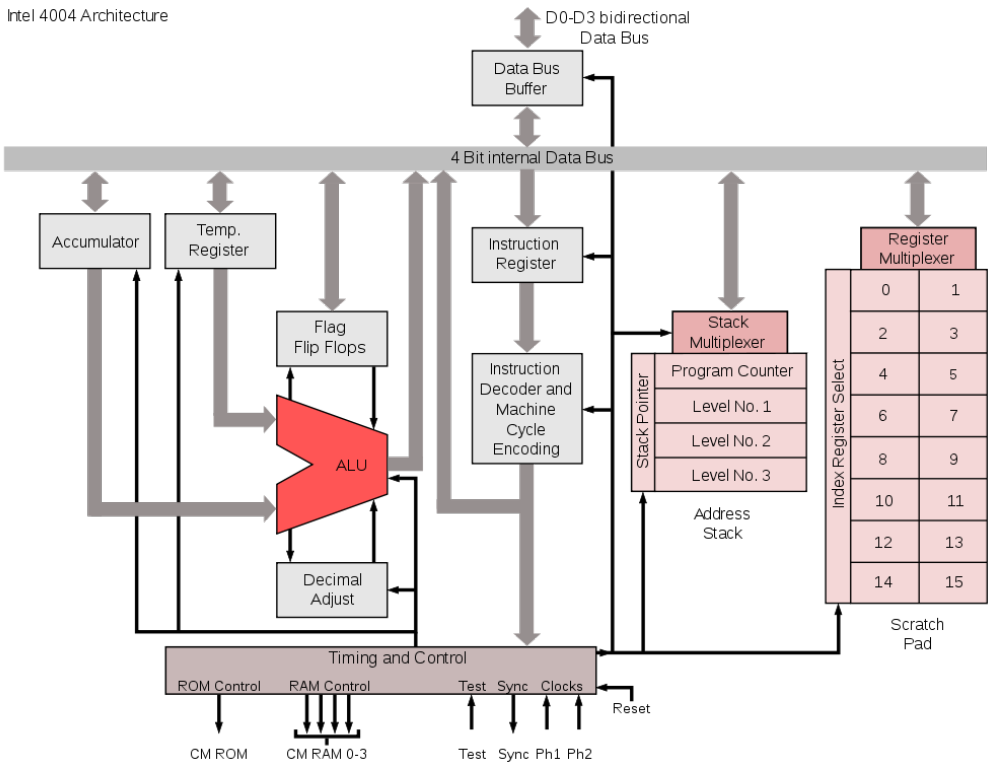
1. Краткая история процессоров Intel
2. Архитектура x86
3. Что означают следующие надписи: x86-64, i386, IA-32, IA-64?
4. Кто пишет понятней: AT&T или Intel?
5. NASM — наше всё!
6. Макросы нам помогут!

Intel 4004 — 1971 год



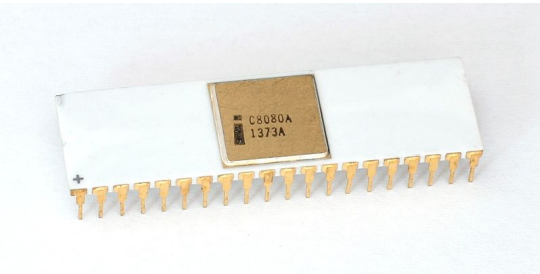
Основные характеристики:

- Тактовая частота 92,6 кГц
- Гарвардская архитектура
- Объём адресуемой памяти: 640 байт
- 16 4-х битных регистров
- Число инструкций — 46
- Число транзисторов — 2250



Блок-схема

Intel 8080 — 1974 год

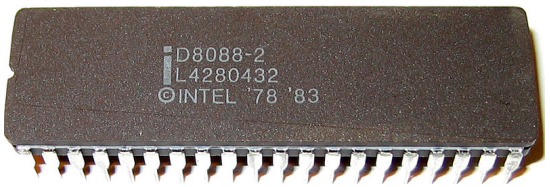


Основные характеристики:

- Тактовая частота: 2 мГц
- Прингстонская архитектура
- Объём адресуемой памяти: 64 Кбайт
- 7 8-ми битных регистров
- Число инструкций — 80
- Число транзисторов — 6000

Altair 8800

Intel 8088 — 1979 год



Основные характеристики:

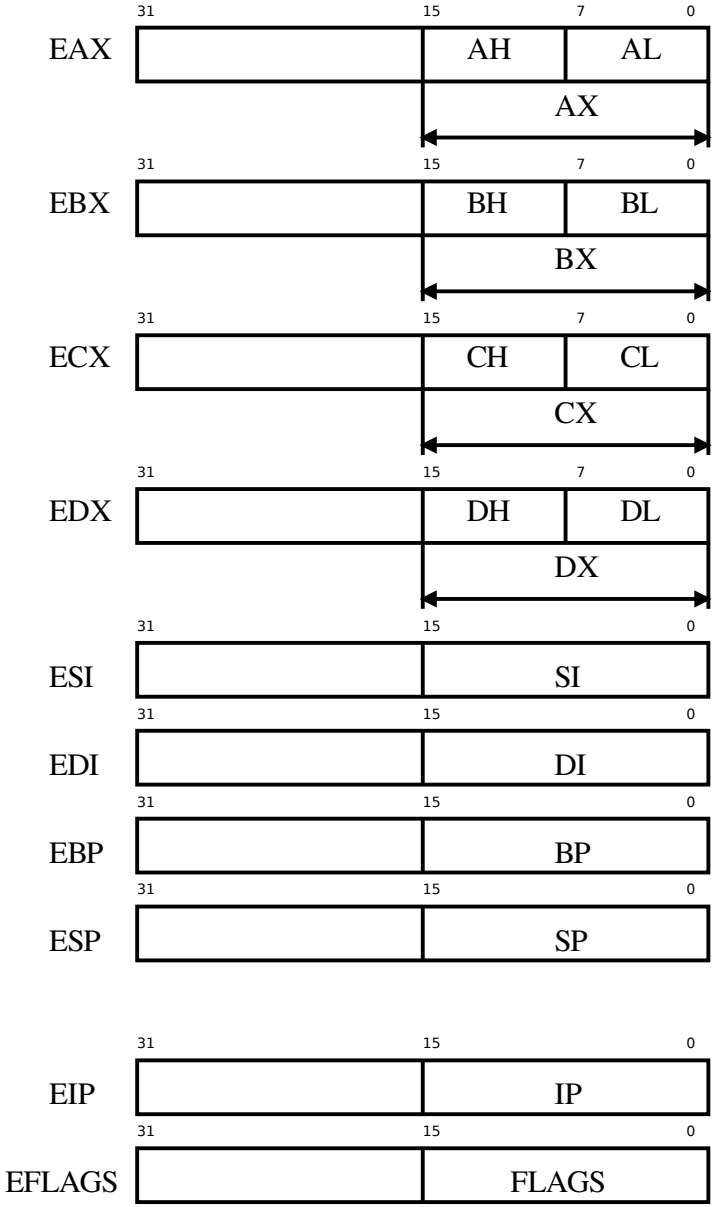
- Тактовая частота: 5 мГц
- Объём адресуемой памяти: 1 Мбайт
- 8 16-ти битных регистров
- Число инструкций — 98
- Число транзисторов — 29000

IBM PC

Intel 80386 (i386) — 1985 год

Основные характеристики:

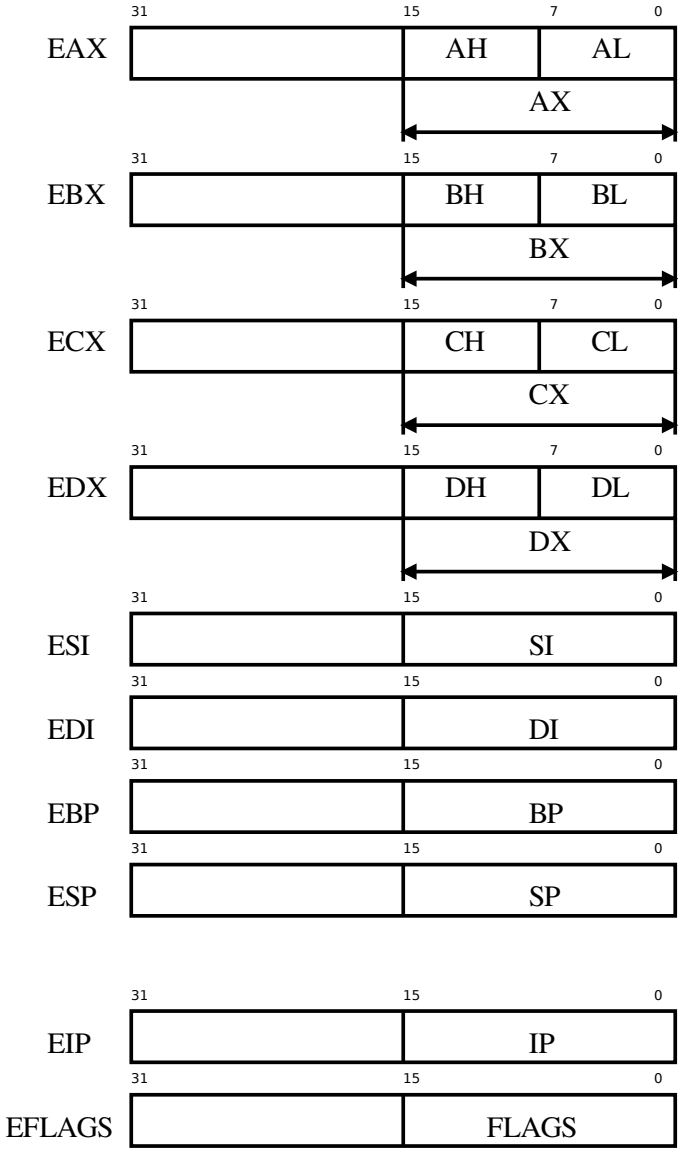
- Тактовая частота: 16, 20,25,33,40 мГц
- Объём адресуемой памяти: 4 Гбайт
- 8 32-х битных регистров
- Число инструкций — 150 (x86 или IA-32)
- Число транзисторов — 275000



И остальные модели

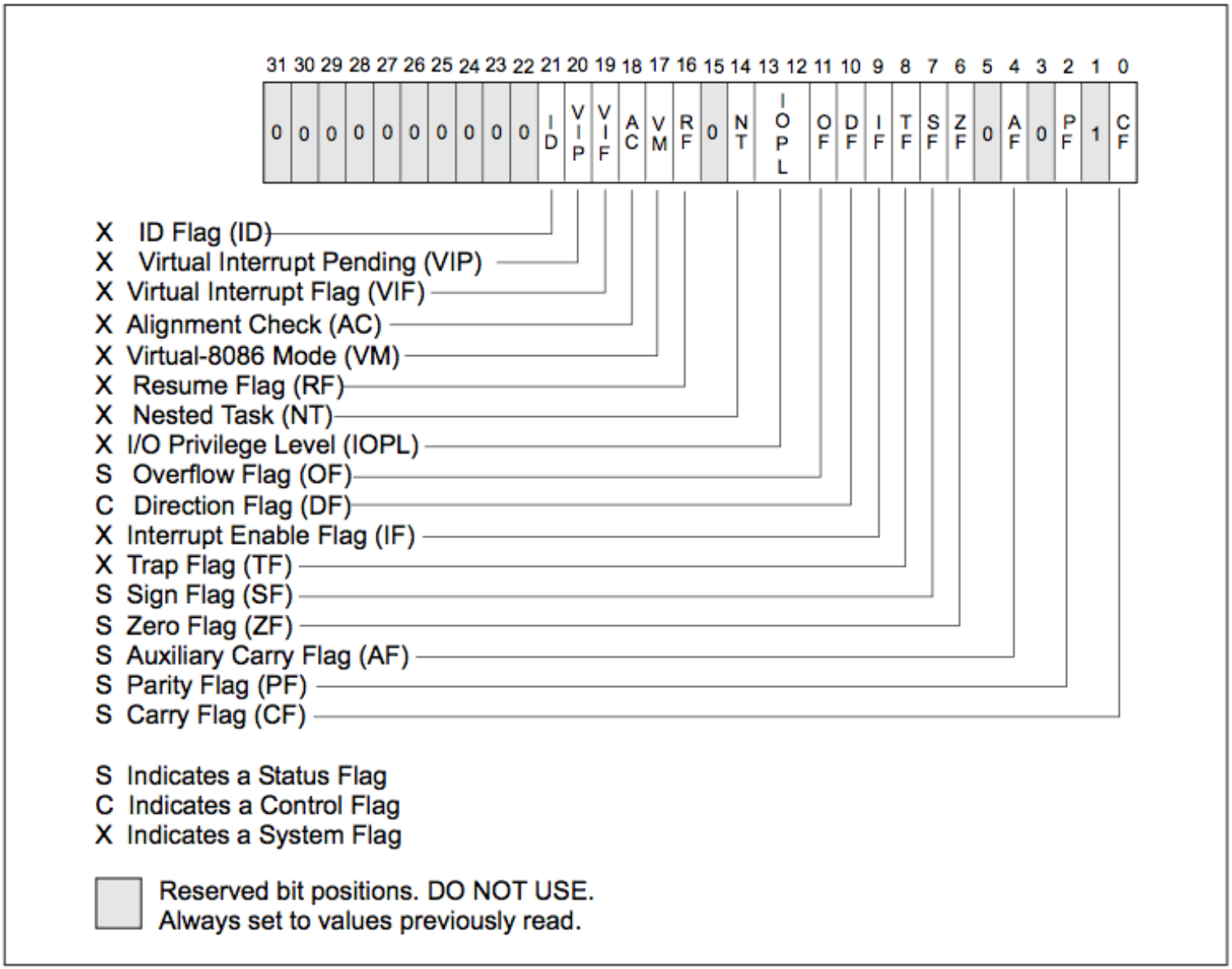
- **i486** — математический сопроцессор (FPU), кеш L1, L2
- **Pentium** — суперскалярная архитектура, предсказание ветвлений, раздельное кеширование кода и данных
- **Pentium II** — SIMD MMX инструкции
- **Pentium III** — RISC-ядро, SSE (Streaming SIMD Extensions)
- **Pentium 4** — Hyper-threading, SSE2, SSE3, EMT64T(x86-64 или IA-64)
- **Pentium D** — 2 ядра
- **Core 2** — 1,2,4 ядра, VT-x, SSSE3
- **Core i3,i5,i7** — Hyper-threading, встроенный графический процессор, SSE4

Архитектура x86



- Разрядность регистров — 32 бита
  - **EAX** — аккумулятор
  - **EBX** — адрес данных
  - **ECX** — счётчик циклов
  - **EDX** — для хранения данных
  - **ESI** — адрес источника
  - **EDI** — адрес приёмника
  - **EBP** — указатель на данные в стеке
  - **ESP** — указатель на вершину стека
  - **EIP** — счётчик команд
  - **EFLAGS** — регистр флагов
- Объём адресуемой памяти — 4 Гбайта (2^{32} байт)
- Плоская модель памяти
- Число инструкций общего назначения ~ 257

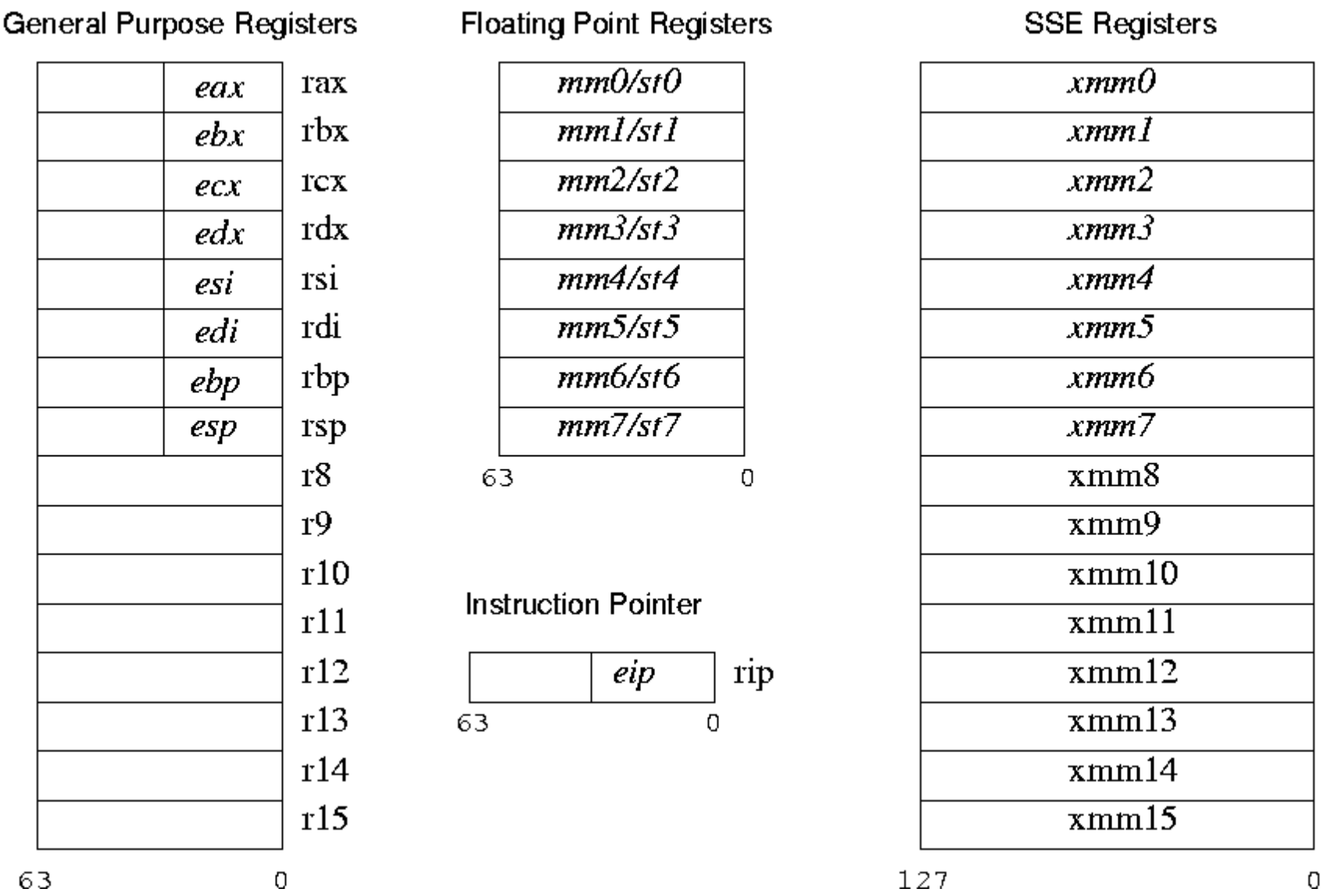
Регистр флагов EFLAGS



Флаги состояния

- **ZF** — zero flag (флаг нуля) показывает равенство результата нулю
- **CF** — carry flag (флаг переноса) показывает наличие переполнения в беззнаковой целочисленной арифметике
- **SF** — sign flags (флаг знака) показывает знак результата
- **OF** — overflow flags (флаг переполнения) показывает наличие переполнения в знаковой целочисленной арифметике
- **PF** — parity flags (флаг чётности) показывает чётность результата
- **AF** — auxiliary carry flags (вспомогательный флаг переноса) показывает наличие переполнения а двоично-десятичной арифметике (binary coded deciaml, BCD)
- **DF** — direction flag (флаг направления)

Архитектура x86-64 (IA-64)



А какие ещё есть архитектуры процессоров?

А какие ещё есть архитектуры процессоров?

- ARM — посмотрите на ваш мобильный телефон или планшет

А какие ещё есть архитектуры процессоров?

- ARM — посмотрите на ваш мобильный телефон или планшет
- Atmel — внимательно присмотритесь к стиральной машинке или холодильнику, а также поиграйте с Arduino

А какие ещё есть архитектуры процессоров?

- ARM — посмотрите на ваш мобильный телефон или планшет
- Atmel — внимательно присмотритесь к стиральной машинке или холодильнику, а также поиграйте с Arduino
- Microship PIC — та же бытовая техника

А какие ещё есть архитектуры процессоров?

- ARM — посмотрите на ваш мобильный телефон или планшет
- Atmel — внимательно присмотритесь к стиральной машинке или холодильнику, а также поиграйте с Arduino
- Microship PIC — та же бытовая техника
- PowerPC — до 2006 года основной процессор фирмы Apple

А какие ещё есть архитектуры процессоров?

- ARM — посмотрите на ваш мобильный телефон или планшет
- Atmel — внимательно присмотритесь к стиральной машинке или холодильнику, а также поиграйте с Arduino
- Microship PIC — та же бытовая техника
- PowerPC — до 2006 года основной процессор фирмы Apple
- SPARC — замечательные числодробилки

А какие ещё есть архитектуры процессоров?

- ARM — посмотрите на ваш мобильный телефон или планшет
- Atmel — внимательно присмотритесь к стиральной машинке или холодильнику, а также поиграйте с Arduino
- Microship PIC — та же бытовая техника
- PowerPC — до 2006 года основной процессор фирмы Apple
- SPARC — замечательные числодробилки
- MIPS — Sony PlayStation 2

А какие ещё есть архитектуры процессоров?

- ARM — посмотрите на ваш мобильный телефон или планшет
- Atmel — внимательно присмотритесь к стиральной машинке или холодильнику, а также поиграйте с Arduino
- Microship PIC — та же бытовая техника
- PowerPC — до 2006 года основной процессор фирмы Apple
- SPARC — замечательные числодробилки
- MIPS — Sony PlayStation 2
- Эльбрус — разработанные в РФ процессоры для ПК и серверов

Синтаксис ассемблера AT&T и Intel для архитектуры x86

Intel

```
global _start

section .data
    msg db 'Hello, world', 0x0A

section .text
_start:
    mov eax, 4
    mov ebx, 1
    mov ecx, msg
    mov edx, 13
    int 0x80

    mov eax, 1
    mov ebx, 0
    int 0x80
```

AT&T

```
.globl _start

.data
msg: .string "Hello, world\n"

.text
_start:
    movl $4,   %eax
    movl $1,   %ebx
    movl $msg, %ecx
    movl $13,  %edx
    int $0x80

    movl $1, %eax
    movl $0, %ebx
    int $0x80
```

NASM — Netwide Assembler (http://nasm.us)

Конкуренты: GAS, MASM, TASM, FASM, YASM

Преимущества

- Использует Intel синтаксис (GAS уходит)
- Поддерживает как x86, так и x86-64 (TASM уходит)
- Поддерживает различные расширения (MMX, SSE*)
- Работает в различных ОС (Windows, DOS, Linux, FreeBSD, Mac OS X и др) (MASM уходит)
- Продолжает развиваться (Последняя версия YASM — 2014 год)
- Есть литература на русском (FASM уходит)

Hello, world

hello.asm

```
%include "stud_io.inc" ; подключаем файл с макросами
global _start    ; объявляет метку _start как глобальную

section .text    ; секция машинных команд
_start:          ; с метки _start начинается выполнение
  PRINT "Hello, world" ; макрос вывода строк
  PUTCHAR 10     ; 10 - код символа переноса строки
  FINISH         ; макрос завершения программы
```

Hello 5

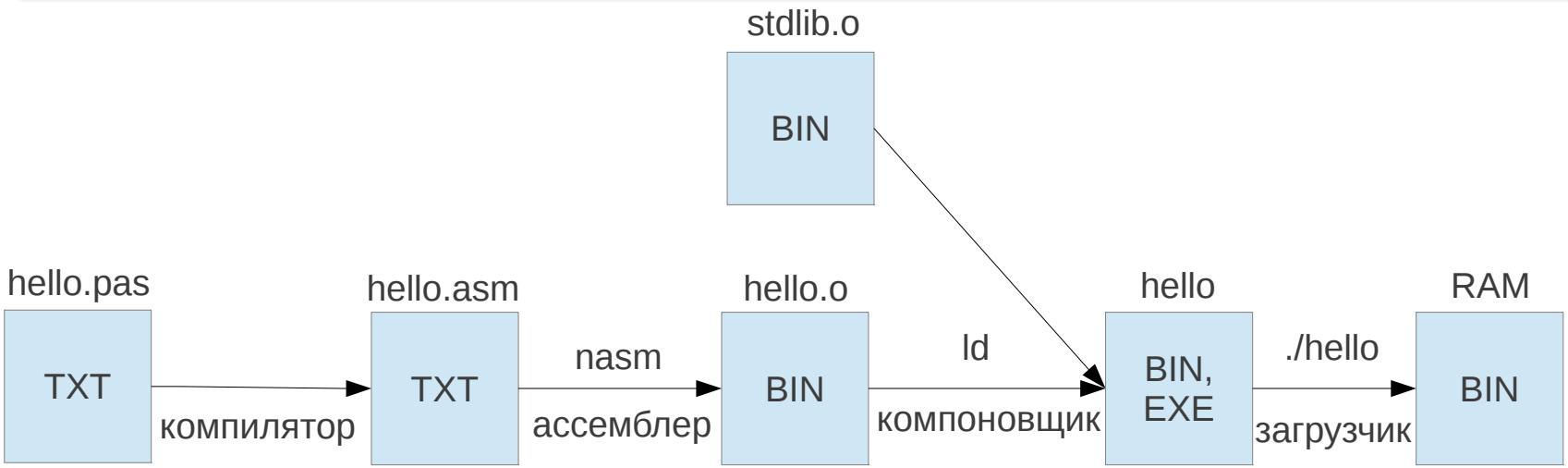
hello.asm

```
%include "stud_io.inc" ; подключаем файл с макросами
global _start    ; объявляет метку _start как глобальную

section .text    ; секция машинных команд
_start:          ; с метки _start начинается выполнение
  mov eax, 0     ; eax - счётчик цикла
again:           ; тело цикла
  PRINT "Hello"  ; макрос вывода строк
  PUTCHAR 10     ; 10 - код символа переноса строки
  inc eax        ; eax := eax + 1
  cmp eax, 5     ; сравнить eax и 5 (eax - 5)
  jl again       ; jump less (прыгнуть если меньше)

  FINISH         ; макрос завершения программы
```

Этапы сборки компилируемых программ

Сборка и выполнение hello.asm

```
nasm -f elf -l hello.lst -o hello.o hello.asm
ld -m elf_i386 -o hello hello.o
./hello
```



Макросы

Макрос — символьное имя, заменяющее несколько команд языка ассемблера

Макросы из stud_io.inc

- `PRINT` — принимает в качестве параметра текстовую строку, окружённую кавычками или апострофами, и выводит её на экран
- `PUTCHAR` — в качестве параметра принимает код символа, однобайтовый регистр (`AL`, `AH`, `BL`, `BH`, `CL`, `CH`, `DL`, `DH`) или исполнительный адрес и выводит на экран символ с соответствующим кодом
- `GETCHAR` — считывает код символа с клавиатуры и сохраняет его код в регистре `EAX`. Если символов больше нет, то в регистр `EAX` записывается -1 (0xFFFFFFFF)
- `FINISH` — завершает выполнение программы

Укажите правильную последовательность операций

1. компоновка
2. компиляция
3. ассемблирование
4. загрузка
5. выполнение

Укажите правильную последовательность операций

1. компоновка
2. компиляция
3. ассемблирование
4. загрузка
5. выполнение

Ответ: 2, 3, 1, 4, 5

Укажите правильную последовательность операций

1. компоновка
2. компиляция
3. ассемблирование
4. загрузка
5. выполнение

Ответ: 2, 3, 1, 4, 5

Какая инструкция уменьшает значение регистра на 1?

1. mov
2. inc
3. dec
4. cmp

Укажите правильную последовательность операций

1. компоновка
2. компиляция
3. ассемблирование
4. загрузка
5. выполнение

Ответ: 2, 3, 1, 4, 5

Какая инструкция уменьшает значение регистра на 1?

1. mov
2. inc
3. dec
4. cmp

Вопросы

Что делает эта программа?

```
%include "stud_io.inc"
global _start

section .text
_start:
    mov eax, 10
again:
    PUTCHAR '*'
    dec eax
    jnz again
    PUTCHAR 10

    FINISH
```

Вопросы

Что делает эта программа?

```
%include "stud_io.inc"
global _start

section .text
_start:
    mov ecx, 0
again:
    GETCHAR
    cmp eax, -1
    je exit
    inc ecx
    jmp again

exit:
    FINISH
```

Краткие итоги

Что мы узнали сегодня

- послушали краткую историю процессоров Intel
- узнали что такое x86 и чем он отличается от x86-64
- познакомились с азами архитектуры x86
- выбрали ассемблер для дальнейшей работы
- научились читать простые программы