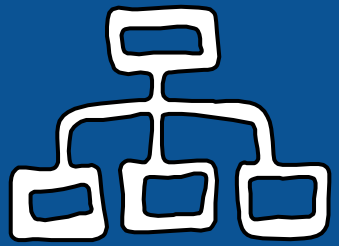
Архитектура ЭВМ и язык ассемблера

Семинар #29:

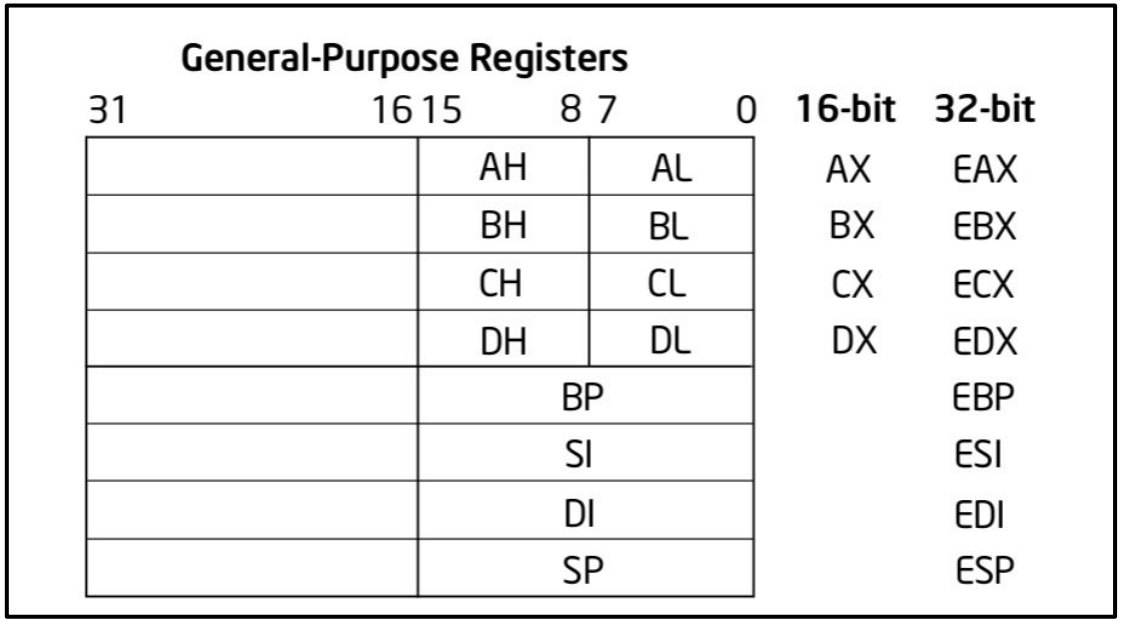
1. Регистры общего назначения в `IA-32`.
2. Арифметические операции: `ADD`, `SUB`, `IMUL`, `IDIV`.
3. Сдвиги и преобразования типов.
4. Изучение кодогенерации с дизассемблером.

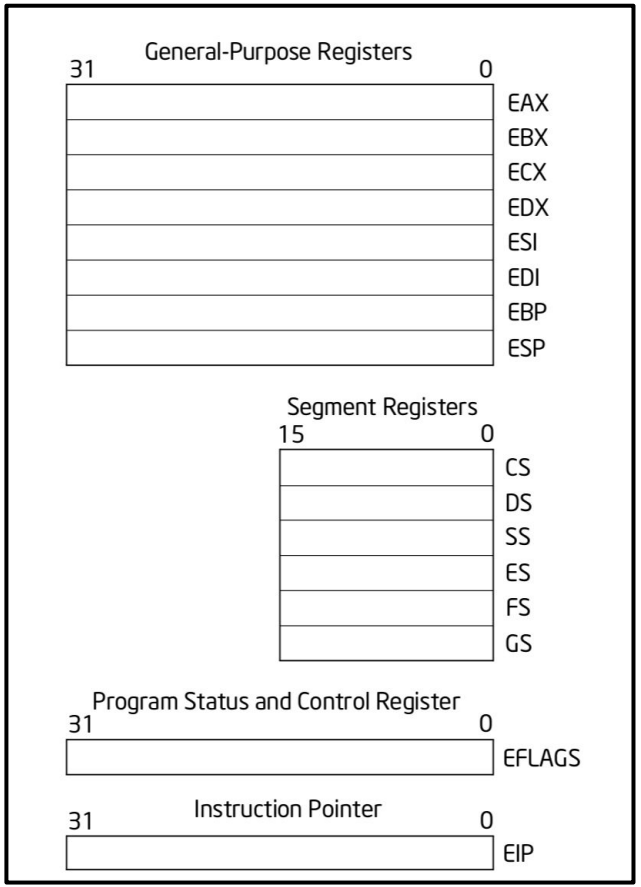
11.04.2025

Регистры общего назначения в i386



Регистры в IA-32



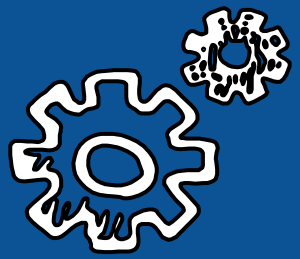


Назначение регистров общего назначения

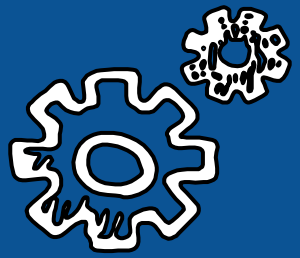
- **EAX** — Accumulator for operands and results data.
- **EBX** — Pointer to data in the DS segment.
- **ECX** — Counter for string and loop operations.
- **EDX** — I/O pointer.
- **ESI** — Pointer to data in the segment pointed to by the DS register; source pointer for string operations.
- **EDI** — Pointer to data (or destination) in the segment pointed to by the ES register; destination pointer for string operations.
- **ESP** — Stack pointer (in the SS segment).
- **EBP** — Pointer to data on the stack (in the SS segment).

Договорённость между разработчиками аппаратуры и разработчиками компиляторов

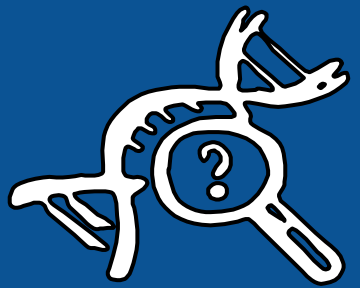
Арифметические операции: ADD, SUB, IMUL, IDIV



Сдвиги
и преобразования типов



Изучение кодогенерации с помощью дизассемблера



Записи в память

```
// Записи в переменные разного размера.
var_08bit = 0x01U;
var_16bit = 0x0001U;
var_32bit = 0x00000001U;
```

```
mov     byte [ebx+0x5c], 0x1
mov     word [ebx+0x5e], 0x1
mov     dword [ebx+0x60], 0x1
```

Сложение и вычитание

```
// Целочисленная арифметика.
var_08bit += 1;
var_16bit -= 1;
var_32bit *= 10;
var_32bit /= 5;
```

```
movzx   eax, byte [ebx+0x5c]
add     eax, 0x1
mov     byte [ebx+0x5c], al
movzx   eax, word [ebx+0x5e]
sub     eax, 0x1
mov     word [ebx+0x5e], ax
```

Умножение

```
// Целочисленная арифметика.
var_08bit += 1;
var_16bit -= 1;
var_32bit *= 10;
var_32bit /= 5;
```

```
mov     edx, dword [ebx+0x60]
mov     eax, edx
shl     eax, 0x2
add     eax, edx
add     eax, eax
mov     dword [ebx+0x60], eax
```

Деление

```
// Целочисленная арифметика.
var_08bit += 1;
var_16bit -= 1;
var_32bit *= 10;
var_32bit /= 5;
```

```
mov     eax, dword [ebx+0x60]
mov     edx, 0xcccccccd
mul     edx
mov     eax, edx
shr     eax, 0x2
mov     dword [ebx+0x60], eax
```

Деление (магия)

```
mov     eax, dword [ebx+0x60]
mov     edx, 0xcccccccd
mul     edx
mov     eax, edx
shr     eax, 0x2
mov     dword [ebx+0x60], eax
```

EDX = 0xCCCCCCCD = 0b1100110011001100... ≈ ⅘ * 0x100000000

EDX:EAX = EAX * EDX ≈ (⅘ * EAX) * 0x100000000

EDX ≈ ⅘ * EAX

RSLT ≈ ⅘ * EAX / 4

Доступы в память и битовые сдвиги

```
// Доступы в массив.
buffer[0] = 0xA;
var_08bit = buffer[1];

// Битовые сдвиги.
res0 = (ushifted << 10U);
res1 = (ushifted >> 10U);
res2 = (sshifted << 10U);
res3 = (sshifted >> 10U);
```

```
mov     byte [ebx+0x38], 0xa  {buffer}
movzx   eax, byte [ebx+0x39]  {data_804c039}
mov     byte [ebx+0x5c], al  {var_08bit}
mov     eax, 0xa
shl     eax, 0xa  {0x2800}
mov     dword [ebx+0x64], eax  {0x2800}  {res0}
mov     eax, 0xa
shr     eax, 0xa  {0x0}
mov     dword [ebx+0x68], eax  {0x0}  {res1}
mov     eax, 0xa
shl     eax, 0xa  {0x2800}
mov     dword [ebx+0x6c], eax  {0x2800}  {res2}
mov     eax, 0xa
sar     eax, 0xa  {0x0}
mov     dword [ebx+0x70], eax  {0x0}  {res3}
```

Преобразования типов

```
// Сужающие преобразования типов.
long long val64 = 1;
unsigned  val32 = 2;
short     val16 = 3;
char      val08 = 4;

val32 = (unsigned) val64;
val16 = (short)    val32;
val08 = (char)     val16;

// Расширяющие преобразования типов.
val64 = (long long) val32;
val32 = (unsigned)  val16;
val16 = (short)     val08;
```

```
mov     dword [ebp-0x10], 0x1
mov     dword [ebp-0xc {var_10}], 0x0
mov     dword [ebp-0x14 {var_18}], 0x2
mov     word [ebp-0x16 {var_1a}], 0x3
mov     byte [ebp-0x17 {var_1b}], 0x4
mov     eax, dword [ebp-0x10]  {0x1}
mov     dword [ebp-0x14], eax  {0x1}
mov     eax, dword [ebp-0x14]  {0x1}
mov     word [ebp-0x16 {var_1a_1}], ax  {0x1}
movzx   eax, word [ebp-0x16]  {0x1}
mov     byte [ebp-0x17 {var_1b_1}], al  {0x1}
mov     eax, dword [ebp-0x14]  {0x1}
mov     dword [ebp-0x10 {var_14}], eax  {0x1}
mov     dword [ebp-0xc {var_10_1}], 0x0
movsx   eax, word [ebp-0x16]  {0x1}
mov     dword [ebp-0x14 {var_18_1}], eax  {0x1}
movsx   ax, byte [ebp-0x17]  {0x1}
mov     word [ebp-0x16 {var_1a_2}], ax  {0x1}
```

Вопросы?

Красивые иконки взяты с сайта handdrawngoods.com