Введение в ассемблер

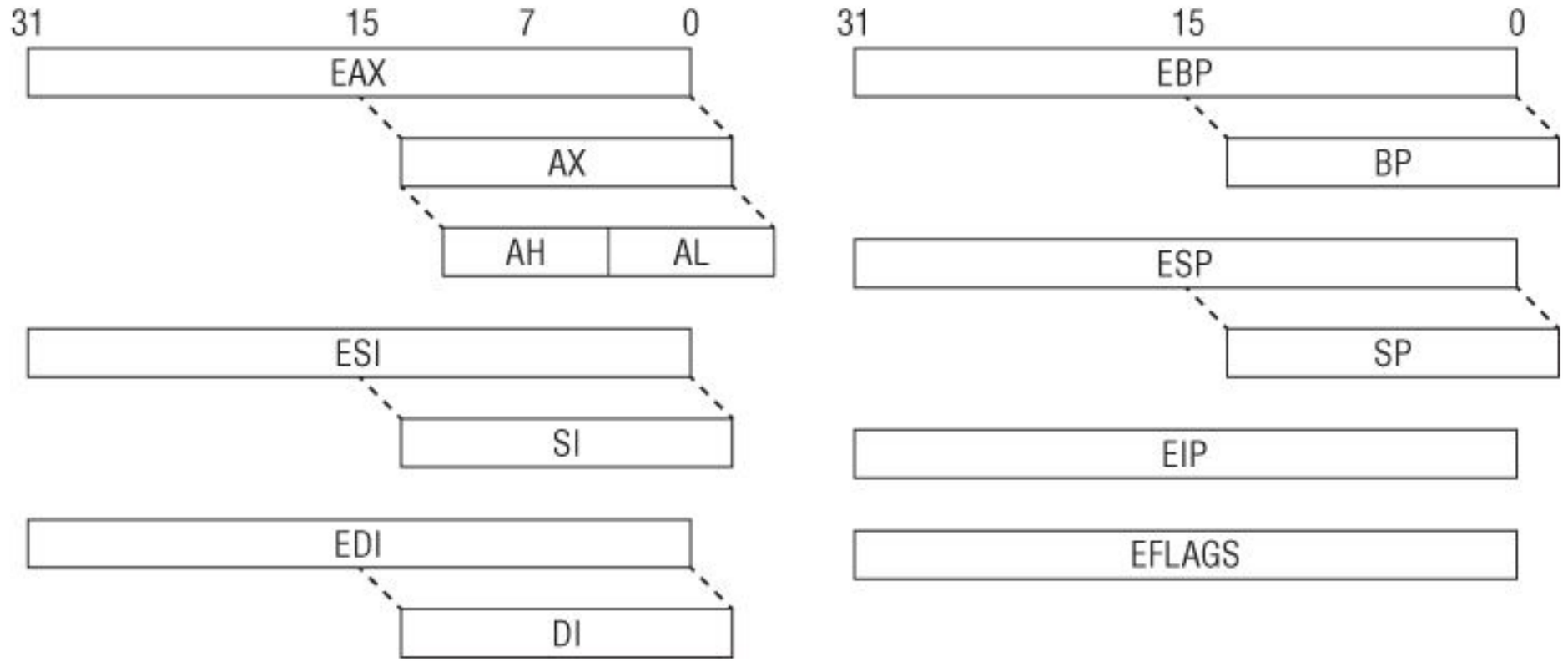
Мельник Богдан

Только x86 архитектура

Регистры

- EAX, EBX, ECX, EDX, EDI, ESI, EBP, ESP
- EIP, EFLAGS

Регистры



Картинка взята из Practical Reverse Engineering

Инструкции

- Перемещение данных
- Арифметические инструкции
- Управление потоком исполнения
- Вызов функций, операции со стеком

Инструкции

- Перемещение данных
- Арифметические инструкции
- Управление потоком исполнения
- Вызов функций, операции со стеком

Синтаксис

Intel

```
mov eax, 12345678h
mov eax, dword ptr[ebx]
mov eax, ecx
```

AT&T

```
movl $0x12345678, %eax
movl (%ebx), %eax
movl %ecx, %eax
```

Перемещение данных

```
mov eax, 12345678h
; eax = 0x12345678
mov ebx, eax
; ebx = eax
mov dword ptr [eax], 1
; *eax = 1
mov ecx, dword ptr [eax]
; ecx = *eax
```

Часть примеров взята из Practical Reverse Engineering

Перемещение данных

```
mov dword ptr [eax], ebx
; *eax = ebx
mov dword ptr [esi+34h], eax
; *(esi+34h) = eax
mov eax, dword ptr [esi+34h]
; eax = *(esi+34h)
mov edx, dword ptr [ecx+eax]
; edx = *(ecx+eax)
```

Часть примеров взята из Practical Reverse Engineering

Перемещение данных

```
lea eax, [Что нибудь]  ==  mov eax, Что нибудь
```

Перемещение данных

movsb/movsw/movsd
byte/word/dword ptr [edi] = byte/word/dword ptr [esi]

stosb/stosw/stosd
byte/word/dword ptr [edi] = al/ax/eax

lodsb/lodsw/lodsd
al/ax/eax/ = byte/word/dword ptr [esi]

После выполнения инструкций edi и/или esi изменяются. Увеличиваются или уменьшаются на 1/2/4 байта в зависимости от флага DF.

Арифметические операции

```
add eax, ebx          ; eax = eax + ebx
sub eax, ebx          ; eax = eax - ebx
inc eax               ; eax = eax + 1
dec eax               ; eax = eax - 1

or eax, 0FFFFFFFFh    ; eax = eax | 0xFFFFFFFF
and eax, 0FFh         ; eax = eax & 0xFF
xor eax, eax          ; eax = eax ^ eax
not eax               ; eax = ~eax
shl eax, 2            ; eax = eax << 2
shr eax, 2            ; eax = eax >> 2
mul ecx               ; edx:eax = eax * ecx
div ecx               ; eax = edx:eax / ecx
```

Управление потоком

Все условные конструкции if/else, switch и while/for траслируются в несколько типов инструкций.

Инструкции влияющие на регистр флагов:

- cmp
- test
- Все арифметические инструкции.

Инструкии меняющие поток исполнения:

- jmp
- jXX

Регистр флагов

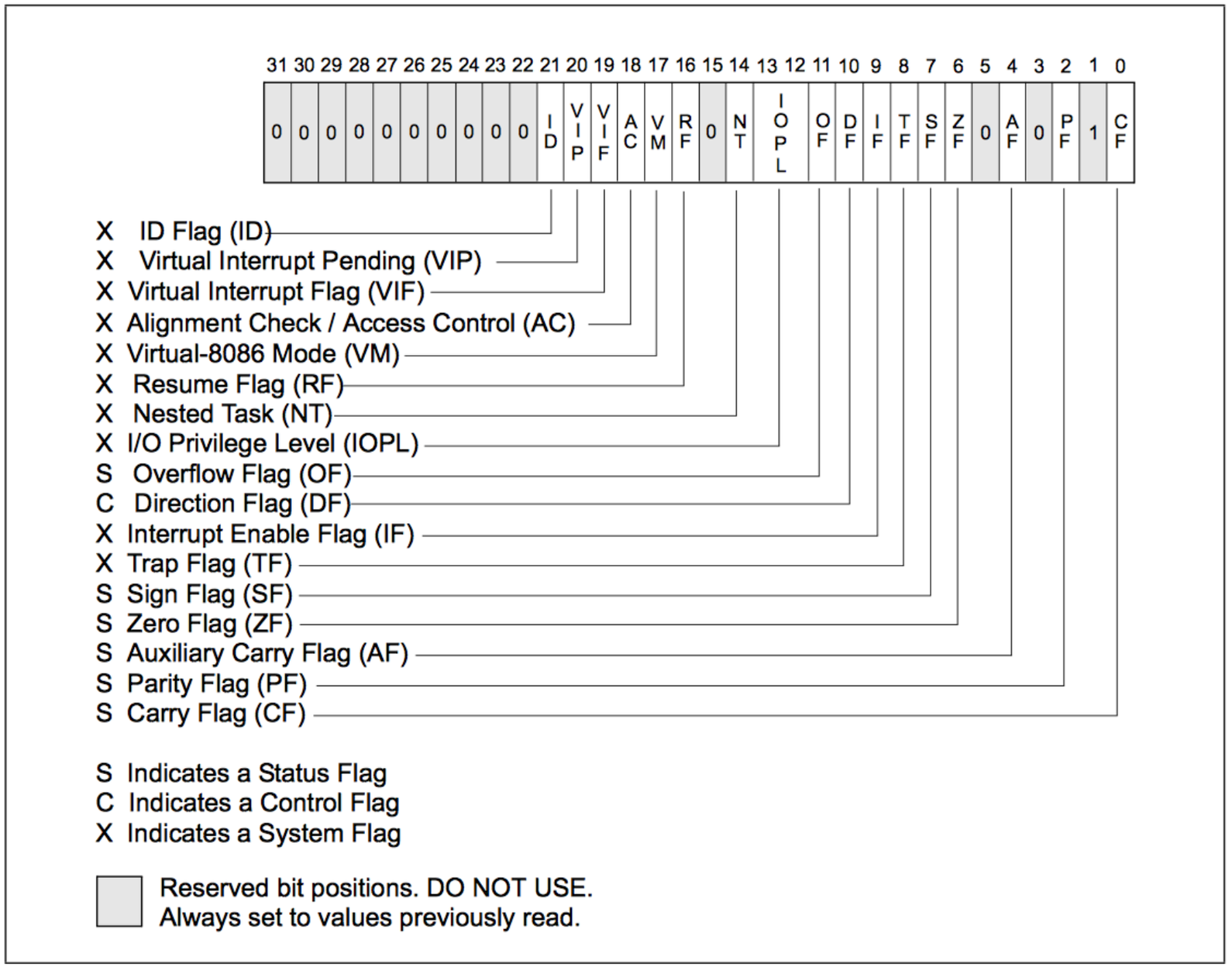


Figure 3-8. EFLAGS Register

ZF

Флаг нуля. Становится 1 если результат последней арифметической операции ноль.

```
sub eax, eax
xor eax, eax

mov eax, 5
sub eax, 5

test eax, eax     ; test это and без изменений
cmp eax, eax      ; cmp это sub без изменений
```

SF

Флаг знака. Становится 1 если результат последней арифметической операции отрицателен.

```
mov eax, 0
dec eax
```

```
mov eax, 3
mov ebx, 5
cmp eax, ebx
```

```
sub eax, ebx
```

CF

Флаг переноса. Становится 1 если мы либо переносим самый старший бит, либо заимствуем его.

```
mov eax, 3
sub eax, 5

mov eax, 80000000
shl eax, 1

mov eax, 80000000
add eax, 80000000
```

OF

Флаг переполнения. Становится 1 если мы вылазим за размер знакового типа.

```
mov al,07Fh
inc al
```

```
mov al, 80h
dec al
```

jmp

Используется для безусловного перехода, аналог goto в C.

```
mov eax, 1
 jmp a
 mov ebx, 10    ; Это не будет исполнено
a:
 mov ebx, 5
```

jXX

Используются для перехода в зависимости от значения регистра флагов.

```
JB/JNAE   — Меньше, CF == 1
JNB/JAE   — Не меньше, CF == 0
JE/JZ     — Равно, ZF == 1
JNE/JNZ   — Не равно, ZF == 0
JL        — Меньше знаково, (SF ^ OF) == 1
JGE/JNL   — Не меньше знаково, (SF ^ OF) == 0
JG/JNLE   — Больше знаково, ((SF ^ OF)|ZF) == 0
```

jXX

Используются для перехода в зависимости от значения регистра флагов.

```
mov eax, 1
mov ebx, 2
 cmp eax, ebx   ; CF = 1, ZF = 0
 je a           ; Переход если ZF == 1
 mov ecx, 0     ; Это выполняется
 jmp exit
a:
 mov ecx, 1     ; Это не выполняется
exit:
```

Что делает этот код?

```
 mov esi, 0BFFFFFF0h
 xor ecx, ecx
loop:
 lodsb
 test al, al
 je exit
 inc ecx
 jmp loop
exit:
```

Что делает этот код?

```
 mov eax, 12345678h
 xor ecx, ecx
 mov edx, 32
loop:
 test eax, 1
 je a
 inc ecx
a:
 shr eax, 1
 dec edx
 jne loop
```