Ассемблер. Базовые инструкции.

Иван Викторович Михайлов

ИТМО, КТ

imihajlow@gmail.com

11.02.2015

Операнды

`instruction dst, src`

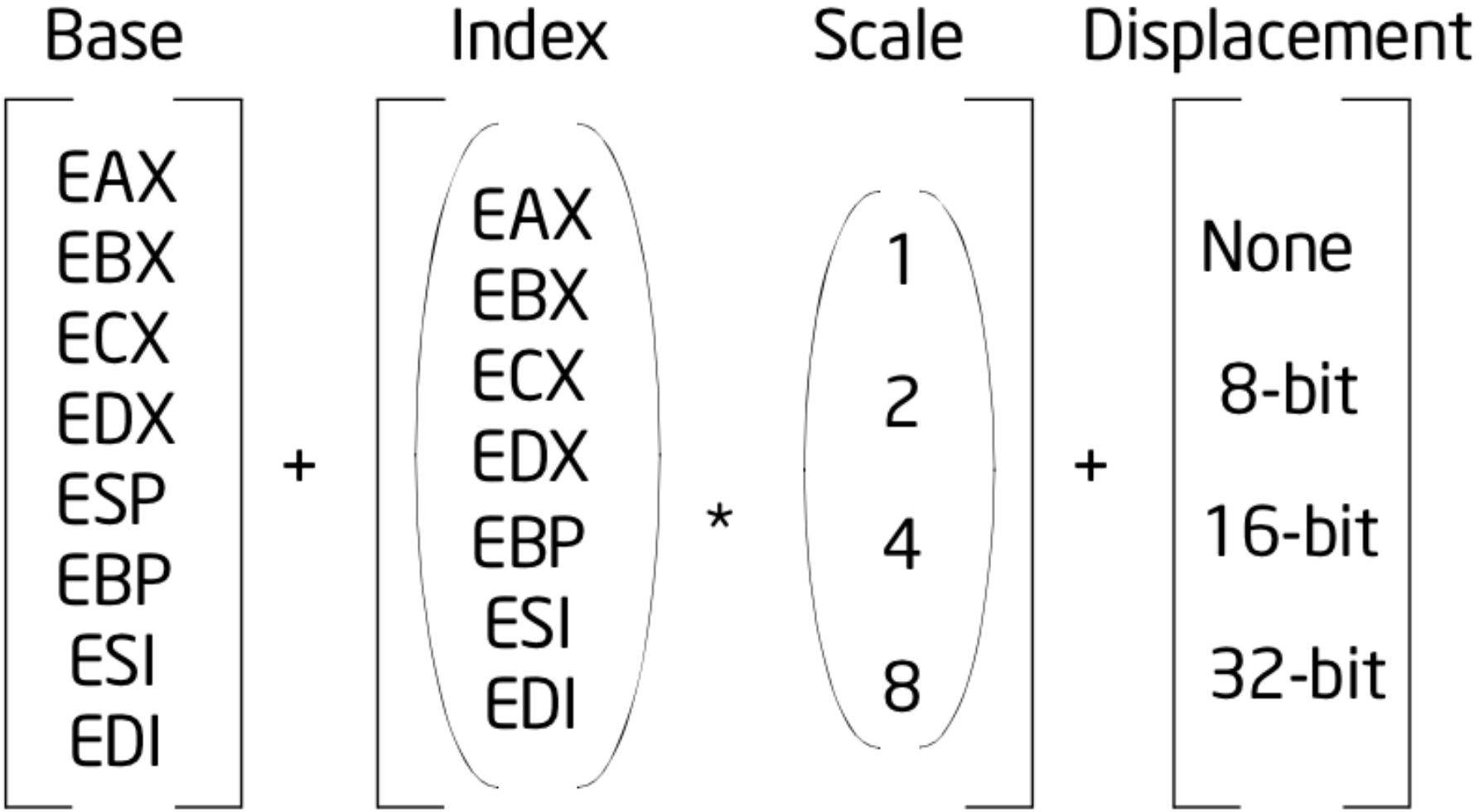
Первым приемник, вторым источник.

Виды операндов:

- Регистр;
- Константа;
- Адрес памяти.

Правило: нельзя из памяти в память.

Адресация



Offset = Base + (Index * Scale) + Displacement

Пересылка

MOV

`MOV dst, src`

dst := src

XCHG

`XCHG dst, src`

Значения в dst и src меняются местами.

Флаги не изменяются.

Арифметика

`instruction dst, src`

ADD	сложение	`dst += src`
SUB	вычитание	`dst -= src`
ADC	сложение с переносом	`dst += src + CF`
SBB	вычитание с переносом	`dst -= src - CF`
AND	побитовое И	`dst &= src`
OR	побитовое ИЛИ	`dst |= src`
XOR	побитовое XOR	`dst ^= src`
CMP	сравнение	`dst - src`
TEST	сравнение с маской	`dst & src`

Флаги изменяются согласно результату.

Арифметика

`instruction dst`

NOT	отрицание	`dst = ~dst`
NEG	замена знака	`dst = -dst`
INC	инкремент	`++dst`
DEC	декремент	`--dst`

Флаги изменяются согласно результату.

Сдвиги

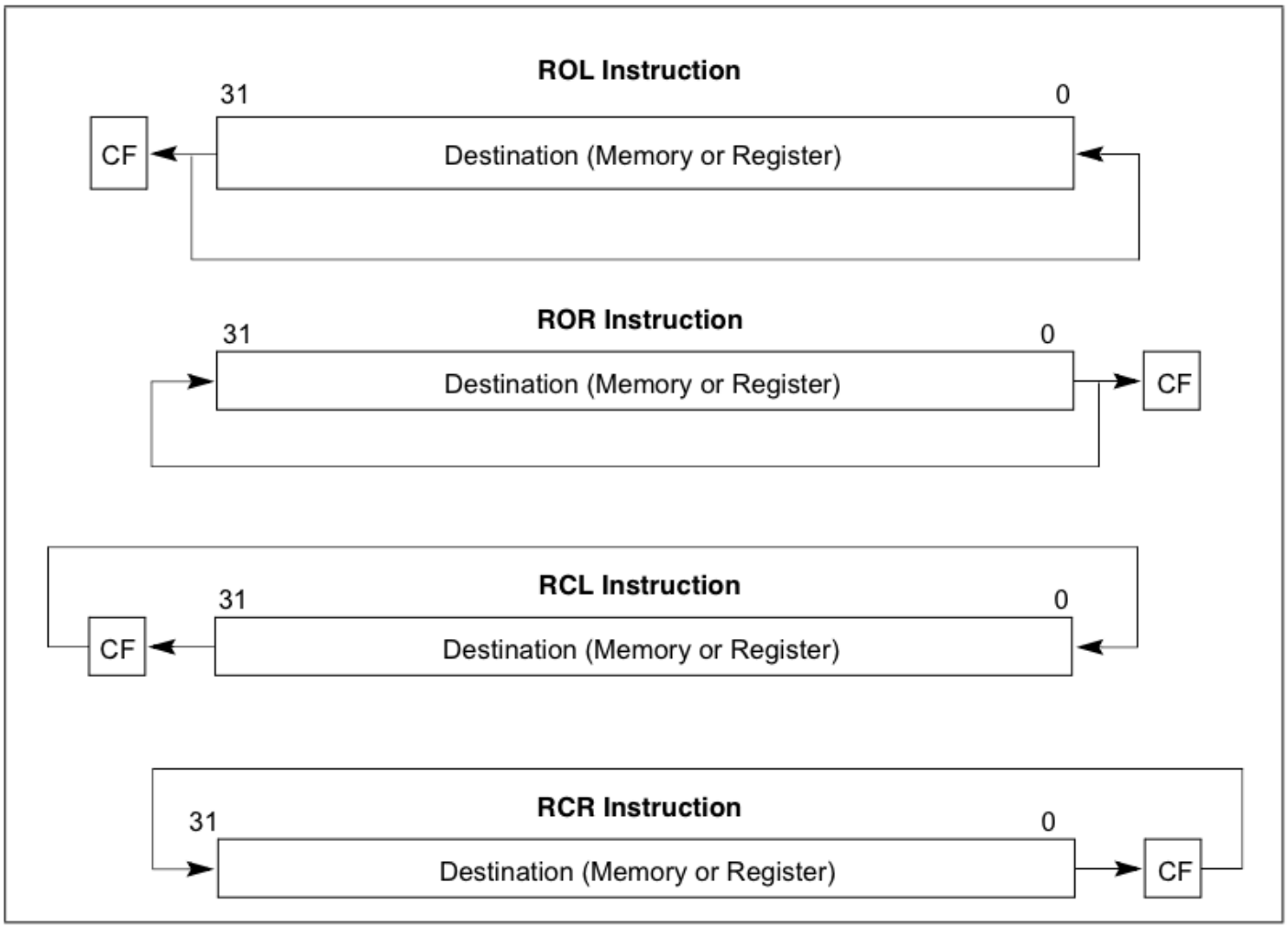
`instruction dst, `*`count`*

SHL/SAL	сдвиг влево	`dst = dst << count`
SHR	сдвиг вправо	`dst = dst >> count`
SAL	арифметический сдвиг вправо	

count – число или CL.

В CF попадает последний „выдвинутый” бит.

Вращения


ROL Instruction
31
0
CF
Destination (Memory or Register)
ROR Instruction
31
0
Destination (Memory or Register)
CF
RCL Instruction
31
0
CF
Destination (Memory or Register)
RCR Instruction
31
0
Destination (Memory or Register)
CF

Умножение

MUL

`MUL src` ; unsigned

`AX = AL * src8`

`DX:AX = AX * src16`

`EDX:EAX = EAX * src32`

OF и CF устанавливаются в 1, если старшая половина результата не ноль.

IMUL

Знаковое умножение.

`IMUL src`

`IMUL reg, src` ; regd *= src

`IMUL regd, src, imm` ; regd = src * imm

В случае с 2 и 3 операндами старшая часть результата теряется.

OF и CF устанавливаются в 1, если старшая половина результата не ноль.

Деление

DIV, IDIV

```
DIV src ; unsigned
IDIV src ; signed

AL = AX / src8; AH = AX % src8
AX = DX:AX / src16; DX = DX:AX % src16
EAX = EDX:EAX / src32; EDX = EDX:EAX % src32
```

Overflow!

Строковые операции

LODSB/LODSW/LODSD

```
LODSD:
    EAX = [ESI]
    if DF:
        ESI -= 4
    else:
        ESI += 4
```

STOSB/STOSW/STOSD

```
STOSD:
    [EDI] = EAX
    if DF:
        EDI -= 4
    else:
        EDI += 4
```

Строковые операции

MOVSB/MOVSW/MOVSD

```
MOVSD:
    [EDI] = [ESI]
    if DF:
        ESI -= 4
        EDI -= 4
    else:
        ESI += 4
        EDI += 4
```

Строковые операции

CMPSB/CMPSW/CMPSD

```
MOVSD:
    set flags from [ESI] - [EDI]
    if DF:
        ESI -= 4
        EDI -= 4
    else:
        ESI += 4
        EDI += 4
```

Строковые операции

REP/REPZ/REPE/REPNZ/REPNE

```
REP STOSx/LODSx/MOVSx/CMPSx

REP/REPZ/REPNZ instruction:
    while ECX != 0:
        instruction
        --ECX
        if REPZ and !ZF:
            break
        if REPNZ and ZF:
            break
```

Стек

PUSH/POP

```
PUSH src
POP dst
```

PUSHF/POPF

`PUSHFD`	EFLAGS
`POPFD`	EFLAGS
`PUSHF`	младшие 16 бит EFLAGS
`POPF`	младшие 16 бит EFLAGS

Передача управления

JMP

Безусловный переход (goto).

```
JMP src
```

Передача управления

Jcc

Условный переход.

`Jcc label`

Операнд – только число или метка!

JO	JNO	OF
JS	JNS	SF
JE/JZ	JNE/JNZ	ZF
JB/JNAE/JC	JNB/JAE/JNC	CF
JBE/JNA	JA/JNBE	CF or ZF
JGE/JNL	JL/JNGE	SF = OF
JLE/JNG	JG/JNLE	ZF or (SF != OF)
JP/JPE	JNP/JPO	PF
JCXZ		CX = 0
JECXZ		ECX = 0

Передача управления

LOOP

Цикл.

```
LOOP label
LOOPZ/LOOPE label
LOOPNZ/LOOPNE label
```

Псевдокод:

```
LOOP label:
    --ECX
    if ECX != 0:
        JMP label

LOOPcc label:
    --ECX
    if ECX != 0:
        Jcc label
```

Передача управления

CALL/RET

Вызов функции

`CALL src`

Возврат из функции `RET`

Что почитать

- Intel® 64 and IA-32 Architectures Software Developer's Manual. Volume 1: Basic Architecture (3.7, 5.1)
- Intel® 64 and IA-32 Architectures Software Developer's Manual. Volume 2: Instruction set reference.

Конец.