Машинно-зависимые языки программирования

Савельев Игорь Леонидович

План

- Флаги
- Команды безусловного перехода
- Команды условного перехода

0. Вопросы. Примеры

```
section .data
        a1 dw 2 ; extern a1
        b1 dw 1

section .text
global CMAIN
CMAIN:
        xor eax, eax
        mov eax, [a1]    ;  eax ?          0x00010002
```

a1	02	00
b1	01	00

0. Вопросы. Примеры

```
section .data
        a1 db 4
        b1 db 2
        c1 dw 240
section .text
global CMAIN                    ; 240 + 100 и 4/2
CMAIN:
        mov ax, [c1]
        add ax, 100             ; ax = 240 + 100 = 340
        mov [c1], ax            ; c1 = 340 (0x154)
        mov al, [a1]             ; 4
        mov bl, [b1]             ; 2
        div bl                  ; al = a1 / b1 = 2
        mov [a1], al            ; a1 = 2
```

0x0100

AL = AX/BL

0x82

0. Вопросы. Примеры

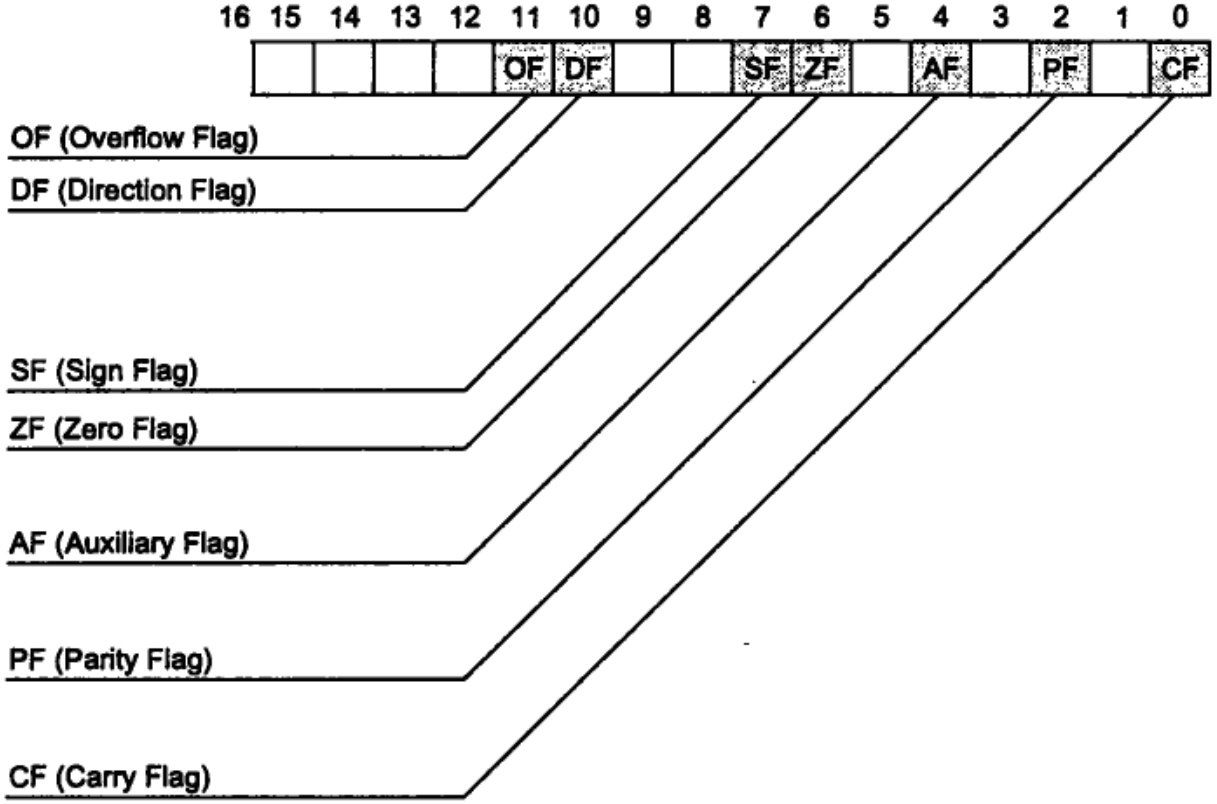
```
section .data
        a1 db 4
        b1 db 2
        c1 dw 240
section .text
global CMAIN                    ; 240 + 100 и 4/2
CMAIN:
        mov ax, [c1]
        add ax, 100             ; ax = 240 + 100 = 340
        mov [c1], ax            ; c1 = 340 (0x154)
        mov al, [a1]             ; 4
        cbw
        mov bl, [b1]             ; 2
        div bl                  ; al = a1 / b1 = 2
        mov [a1], al            ; a1 = 2
```

0. Вопросы. Примеры

```
section .data
        a1 db 4
        b1 db 2
        c1 dw 240
section .text
global CMAIN                    ; 240 + 100 и 4/2
CMAIN:
        mov ax, [c1]
        add ax, 100             ; ax = 240 + 100 = 340
        mov [c1], ax            ; c1 = 340 (0x154)
        mov al, [a1]             ; 4
        cbw
        mov bl, [b1]             ; 2
        div bl                  ; al = a1 / b1 = 2
        mov [a1], eax           ; a1 = 2
```

b1, c1?

Флаги


16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
OF
DF
SF
ZF
AF
PF
CF
OF (Overflow Flag)
DF (Direction Flag)
SF (Sign Flag)
ZF (Zero Flag)
AF (Auxiliary Flag)
PF (Parity Flag)
CF (Carry Flag)

Флаги

add ax, 0

or ax, ax

Флаг	Пояснение
CF=1	Результат сложения НЕ поместился в операнде-приемнике
PF=1	Результат сложения имеет четное число бит со значением 1
SF=1	Копируется СТАРШИЙ (ЗНАКОВЫЙ) бит результата сложения
ZF=1	Результат сложения равен НУЛЮ
OF=1	Если при сложении двух чисел ОДНОГО знака результат сложения получился БОЛЬШЕ допустимого значения. В этом случае приемник МЕНЯЕТ ЗНАК.

Флаги

```
add ax, 0
```

ax	SF	PF	ZF
0000.0001	0	0	0
	0	0	1
0000.1001	0	1	0
0000.0000	0	1	1
1000.1001	1	0	0
	1	0	1
1001.1001	1	1	0
	1	1	1

Флаги (CF и OF)

add al, bl	Результат	Перенес за разрядную сетку	Перенос в 7й бит	CF	OF	SF

mov al, 00111111b
mov bl, 00000001b
; 0011.1111
	63 + 0 = 63
	+63 + 0 = 63
		Нет	Нет	0	0	0

mov al, 11111101b
mov bl, 00000101b
; 0000.0010
	253 + 5 = 2
	-3 + 5 = 2
		Да	Да	1	0	0

mov al, 01000000b
mov bl, 11000000b
; 0000.0000
	64 + 192 = 0
	64 - 64 = 0
		Да	Да	1	0	0

mov al, 11000000b
mov bl, 11000000b
; 1000.0000
	192 + 192 = 128
	-64 - 64 = -128
		Да	Да	1	0	1

mov al, 01000000b
mov bl, 01000000b
; 1000.0000
	64 + 64 = 128
	+64 + 64 = -128
		Нет	Да	0	1	1

mov al, 10000000b
mov bl, 10000000b
; 0000.0000
	128 + 128 = 0
	-128 -128 = 0
		Да	Нет	1	1	0

XOR

Флаги (CF и OF)

sub al, bl	Результат	Заем в 7й бит	Заем из 7й бита	CF	OF	SF

mov al, 11100000b
mov bl, 00100000b
; 1100.0000
	224 – 32 = 192
	-32 – 32 = -64
		Нет	Нет	0	0	1

mov al, 00111111b
mov bl, 11111111b
; 0100.0000
	63 – 255 = 64
	63 – (-1) = 64
		Да	Да	1	0	0

mov al, 10000011b
mov bl, 10011010b
; 1110.1001
	131 – 154 = 233
	-125 – (-102) = -23
		Да	Да	1	0	1

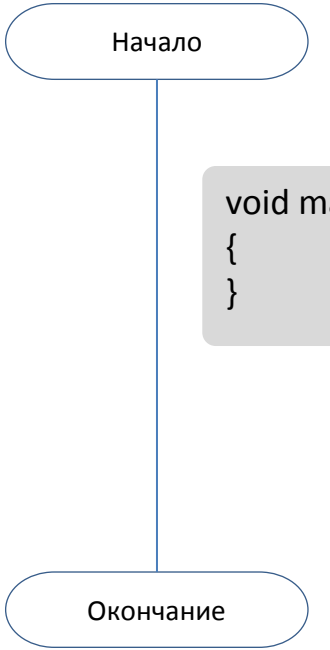
mov al, 00000000b
mov bl, 00000000b
; 0000.0000
	0 – 0 = 0	Нет	Нет	0	0	0

mov al, 10000000b
mov bl, 01000000b
; 0100.0000
	128 – 64= 64
	-128 – 64 = 64
		Нет	Да	0	1	0

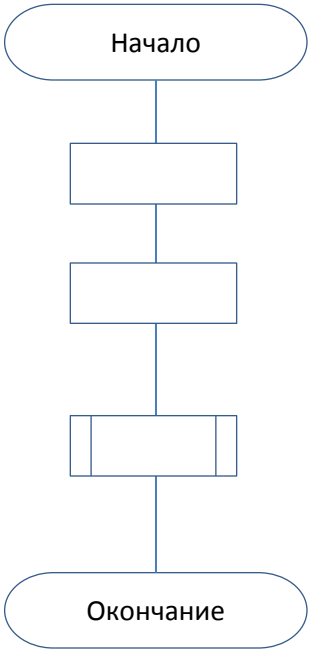
mov al, 01000000b
mov bl, 11000000b
; 1000.0000
	64 – 192 = 128
	64 – (-64) = 128
		Да	Нет	1	1	1

XOR

1. Команды безусловного перехода.




```
void main(void)
{
}
```




```
void main(void)
{
   int a = 0;

   a ++;


  printf("a = %d\n", a);

}
```

1. Команды безусловного перехода.

Команды передачи управления позволяют нарушить естественную последовательность выполнения команд.

Команды безусловной передачи управления:

JMP - Безусловная передача управления;

CALL - Вызов процедуры;

RET - Возврат из процедуры.

Команды условной передачи управления

Команды передачи управления НЕ меняют значения флагов.

1. Команды безусловного перехода.

Безусловный переход

Команда безусловного перехода имеет следующий синтаксис:

JMP <операнд>; аналог **goto**

Операнд указывает адрес перехода.

```
JMP   rel  ;  RIP = RIP + rel
JMP -8
JMP 24
```

```
JMP   reg;  RIP = addr in reg
JMP ebx
```

```
JMP   mem;  RIP = addr
JMP label
```

```
JMP  [req];  RIP = *reg
JMP [ebx]
```

```
JMP   [mem];  RIP = *addr
MOV [var], label
JMP [var]
```

1. Команды безусловного перехода.

```
    . . .
    jmp label

    mov ax, bx
    xor ebx, ebx

label:
    mov cx, dx
    . . .
```

```
section .data
    var db 0x1234
. . .

section .text
. . .
    jmp var    ; !!!!!!
```

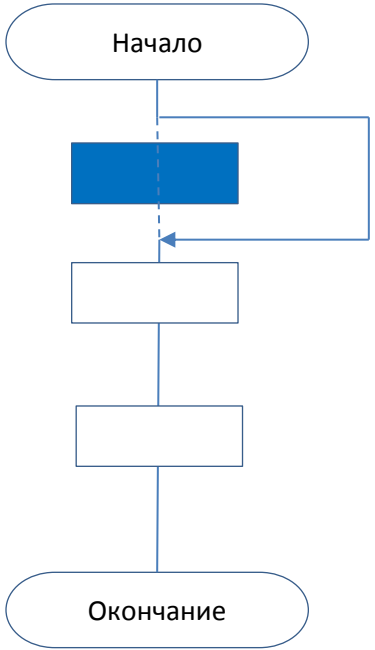
Буквы: от A до Z и от a до z
Цифры: от 0 до 9
Спецсимволы: знак вопроса (?)
точка (.) (только первый символ)
знак "коммерческое эт" (@)
подчеркивание (_)
доллар ($)

Первым символом в метке должна быть буква или спецсимвол. Цифра не может быть первым символом метки, а символы $ и ? иногда имеют специальные значения и обычно не рекомендуются к использованию

Если метка перед директивой/инструкцией, то двоеточие может не ставится

1. Команды безусловного перехода.

```
   org 100h            ;Программа начинается с адреса 100h
 start:
   jmp @mycode
 ;-----------------------------------------------------
    hello db 'Hello, world!‘, 0
;-----------------------------------------------------
@mycode:
    mov eax,1             ; номер системного вызова (sys_exit)
   int 0x80              ; вызов ядра
```

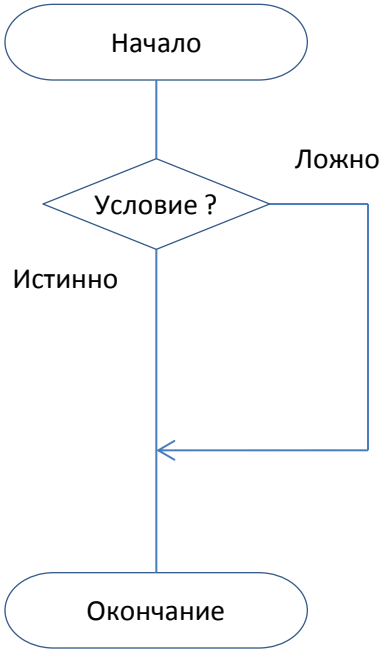


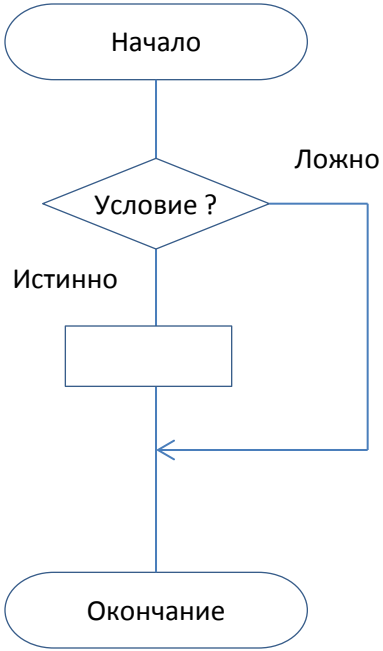
1. Команды безусловного перехода.

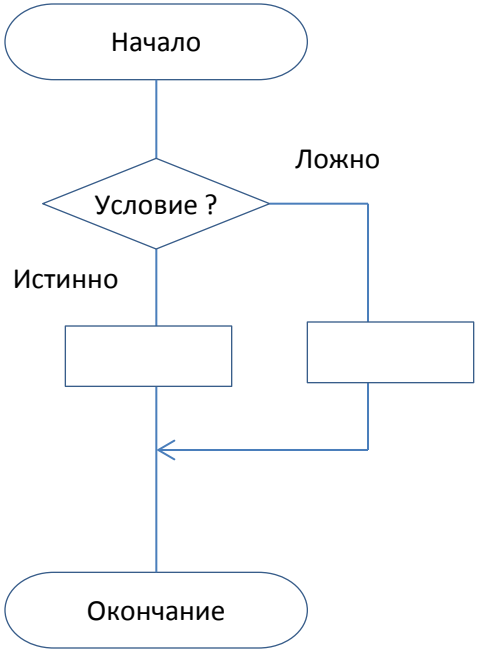
JMP $

выполняет безусловный переход на саму себя, так что создается вечный цикл из одной команды.

2. Команды условного перехода.







if-else, switch, ?: (тернар)

2. Команды условного перехода.

В системе команд процессора архитектуры x86 не предусмотрена поддержка условных логических структур, характерных для языков высокого уровня. Однако на языке ассемблера с помощью набора команд сравнения и условного перехода вы можете реализовать логическую структуру любой сложности. В языке высокого уровня любой условный оператор выполняется в два этапа. Сначала вычисляется значение условного выражения, а затем, в зависимости от его результата, выполняются те или иные действия. Проводя аналогию с языком ассемблера, можно сказать, что сначала выполняются такие команды, как **CMP, AND** или **SUB**, влияющие на **флаги** состояния процессора. Затем выполняется команда условного перехода, которая анализирует значение нужных флагов, и в случае если они установлены, выполняют переход по указанному адресу.

2. Команды условного перехода.

Что касается команд условного перехода, то их достаточно много, но все они записываются единообразно:

Jxx <метка>

Преобразуется в **Jxx rel; RIP = RIP + rel**

E — Equal (равно).
N — Not (не, отрицание).
G — Greater (больше) — применяется для чисел со **ЗНАКОМ**.
L — Less (меньше) — применяется для чисел со **ЗНАКОМ**.
A — Above (выше, больше) — применяется для чисел **БЕЗ ЗНАКА**.
B — Below (ниже, меньше) — применяется для чисел **БЕЗ ЗНАКА**.

Флаги (CF и OF)

sub al, bl	Результат	Заем в 7й бит	Заем из 7й бита	ZF	CF	OF	SF	uint	int

mov al, 11100000b
mov bl, 00100000b
; 1100.0000
	224 – 32 = 192
	-32 – 32 = -64
		Нет	Нет	0	0	0	1	>	<

mov al, 00111111b
mov bl, 11111111b
; 0100.0000
	63 – 255 = 64
	63 – (-1) = 64
		Да	Да	0	1	0	0	<	>

mov al, 10000011b
mov bl, 10011010b
; 1110.1001
	131 – 154 = 233
	-125 – (-102) = -23
		Да	Да	0	1	0	1	<	<

mov al, 00000000b
mov bl, 00000000b
; 0000.0000
	0 – 0 = 0	Нет	Нет	1	0	0	0	=	=

mov al, 10000000b
mov bl, 01000000b
; 0100.0000
	128 – 64= 64
	-128 – 64 = 64
		Нет	Да	0	0	1	0	>	<

mov al, 01000000b
mov bl, 11000000b
; 1000.0000
	64 – 192 = 128
	64 – (-64) = 128
		Да	Нет	0	1	1	1	<	>

XOR

2. Команды условного перехода. Переход по условию

Мнемокод	**Название**	**Условие перехода после команды CMP op_1, op_2**	**Значения флагов**	**Примечание**
JE	Переход если равно	op1 = op2	ZF = 1	Для всех чисел
JNE	Переход если не равно	op1 ≠ op2	ZF = 0	
JL/JNGE	Переход если меньше	op1 < op2	SF ≠ OF	Для чисел со знаком
JLE/JNG	Переход если меньше или равно	op1 ≤ op2	SF ≠ OF или ZF = 1	
JG/JNLE	Переход если больше	op1 > op2	SF = OF и ZF = 0	
JGE/JNL	Переход если больше или равно	op1 ≥ op2	SF = OF	
JB/JNAE	Переход если ниже	op1 < op2	CF = 1	Для чисел без знака
JBE/JNA	Переход если ниже или равно	op1 ≤ op2	CF = 1 или ZF = 1	
JA/JNBE	Переход если выше	op1 > op2	CF = 0 и ZF = 0	
JAE/JNB	Переход если выше или равно	op1 ≥ op2	CF = 0	

2. Команды условного перехода. Анализ флагов

Мнемокод	Условие перехода	Мнемокод	Условие перехода
JZ	ZF = 1	JNZ	ZF = 0
JS	SF = 1	JNS	SF = 0
JC	CF = 1	JNC	CF = 0
JO	OF = 1	JNO	OF = 0
JP	PF = 1	JNP	PF = 0

2. Команды условного перехода. Регистр CX

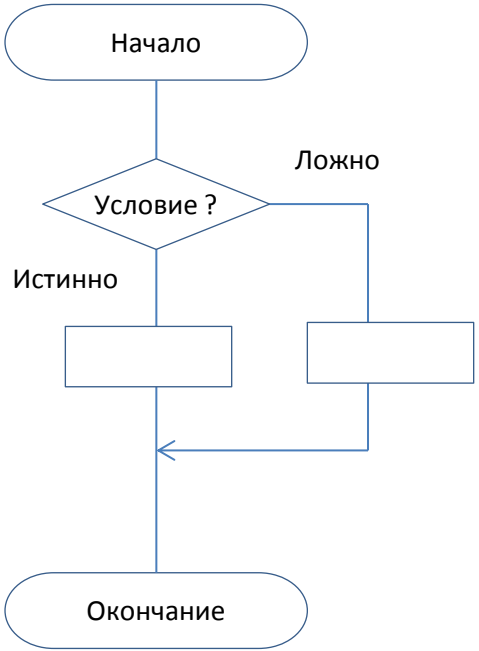
Три команды условного перехода, проверяющие не флаги, а значение регистра RCX, ECX или CX:

JCXZ <метка> ; Переход, если значение регистра CX равно 0

JECXZ <метка> ; Переход, если значение регистра ECX равно 0

JRCXZ <метка> ; Переход, если значение регистра RCX равно 0

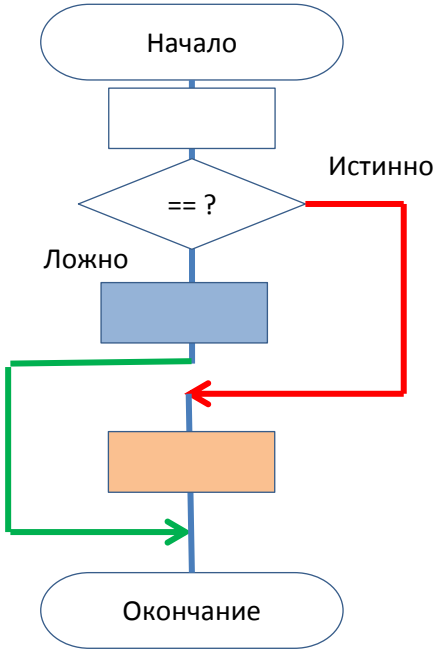
2. Команды условного перехода. Знаковые числа




```
if( a == 0 )
   {}
else
   {}
```

```
section .data
   a dd  5

section .text
   mov eax, [a]
   cmp eax, 0          ;  a == 0 ?
   je @a_is_0
   nop                 ; a != 0
   jmp @end_if
@a_is_0:
   nop                 ; a == 0
   ; jmp @end_if
@end_if:
   ret
```



2. Команды условного перехода.

JE	Переход если равно	op1 = op2	ZF = 1	Для всех чисел
JNE	Переход если не равно	op1 ≠ op2	ZF = 0	

```
section .data
   a dd  5

section .text
   mov eax, [a]
   cmp eax, 0;    PF = 1, ZF = 0
   je @a_is_0
   nop ; a != 0 Ложно
   jmp @end_if
@a_is_0:
   nop ; a == 0  Истина
   ; jmp @end_if
@end_if:
   ret
```

```
section .data
   a dd  0

section .text
   mov eax, [a]
   cmp eax, 0;         PF = 1, ZF = 1
   je @a_is_0
   nop ; a != 0 Ложно
   jmp @end_if
@a_is_0:
   nop ; a == 0  Истина
   ; jmp @end_if
@end_if:
   ret
```

2. Команды условного перехода

JE	Переход если равно	op1 = op2	ZF = 1	Для всех чисел
JNE	Переход если не равно	op1 ≠ op2	ZF = 0	

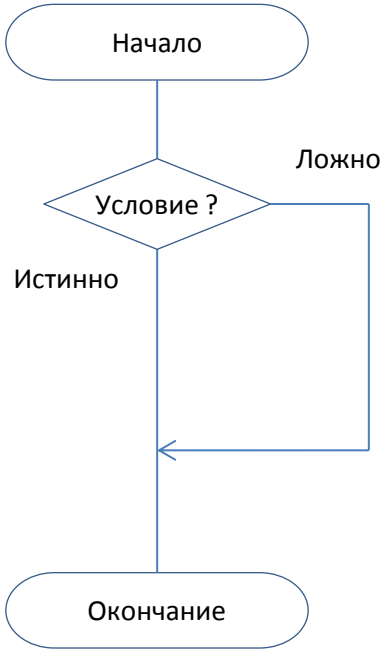
```
section .data
   a dd  5

section .text
   mov eax, [a]
   cmp eax, 0;    PF = 1, ZF = 0
   je @a_is_0
   nop ; a != 0 Ложно
   jmp @end_if
@a_is_0:
   nop ; a == 0  Истина
   ; jmp @end_if
@end_if:
   ret
```

```
section .data
   a dd  5

section .text
   mov eax, [a]
   cmp eax, 0;          PF = 1, ZF = 0
   jne @a_is_not0
   nop                  ; a == 0 Истина
   jmp @end_if
@a_is_not0:
   nop ;                 a != 0  Ложно
   ; jmp @end_if
@end_if:
   ret
```

2. Команды условного перехода. Минимальный пример




```
   mov eax, [a]
   cmp eax, 0;
   je @a_is_0
   jmp @end_if
@a_is_0:
@end_if:
   ret
```

2. Команды условного перехода. Знаковые числа

JL/JNGE	Переход если меньше	op1 < op2	SF ≠ OF

```
a db -1
b db 127
        xor eax, eax
        mov al, [a]
        mov bl, [b]
        sub al, bl          ; SF = 1, OF = 0
                            ;al == 0x80, -128
```

```
a db 255
b db 127
        xor eax, eax
        mov al, [a]
        mov bl, [b]
        sub al, bl          ; SF = 1, OF = 0
                            ;al == 0x80, 128
```

2. Команды условного перехода. Знаковые числа

JL/JNGE	Переход если меньше	op1 < op2	SF ≠ OF

```
   a db -1
   b db  127

   xor eax, eax
   mov al, [a]
   mov bl, [b]
   cmp al, bl          ; SF = 1, OF = 0
   jl @a_less_b
   nop ; a >= b
   jmp @end_if
@a_less_b:
   nop ; a < b         ; !!!!!!! OK
@end_if:
   ret
```

```
   a db 255
   b db  127

   xor eax, eax
   mov al, [a]
   mov bl, [b]
   cmp al, bl          ; SF = 1, OF = 0
   jl @a_less_b
   nop ; a >= b
   jmp @end_if
@a_less_b:
   nop ; a < b         ; !!!!!!! err
@end_if:
   ret
```

2. Команды условного перехода. Знаковые числа

JL/JNGE	Переход если меньше	op1 < op2	SF ≠ OF

```
   a db 127
   b db  -1

   xor eax, eax
   mov al, [a]
   mov bl, [b]
   cmp al, bl          ; SF = 1, OF = 1, CF = 1
   jl @a_less_b
   nop ; a >= b        ; !!! OK
   jmp @end_if
@a_less_b:
   nop ; a < b
@end_if:
   ret
```

```
a db 127
b db  255

xor eax, eax
mov al, [a]
mov bl, [b]
sub al, bl          ; SF = 1, OF = 1, CF = 1
                    ; ax = 0x80, 128 (-128)
```

2. Команды условного перехода. Беззнаковые числа

JB/JNAE	Переход если ниже	op1 < op2	CF = 1

```
   a db 255
   b db  127

   xor eax, eax
   mov al, [a]
   mov bl, [b]
   cmp al, bl          ; SF = 1, OF = 0, CF = 0
   jl @a_less_b
   nop ; a >= b
   jmp @end_if
@a_less_b:
   nop                 ; a < b  !!!!! err
@end_if:
   ret
```

```
   a db 255
   b db  127

   xor eax, eax
   mov al, [a]
   mov bl, [b]
   cmp al, bl          ; SF = 1, OF = 0, CF = 0
   jb @a_less_b
   nop                 ; a >= b    !!! OK
   jmp @end_if
@a_less_b:
   nop ; a < b
@end_if:
   ret
```

2. Команды условного перехода. Анализ флагов

JZ	ZF = 1
JNZ	ZF = 0

A == 0 ?

```
   mov eax, [a]
   cmp eax, 0
   jnz @a_is_not0
   nop                  ; a == 0
   jmp @end_if
@a_is_not0:
   nop                  ; a != 0
@end_if:
   ret
```

A четное ?

```
   mov eax, [a]
   and eax, 0x00000001
   jz @a_is_even
   nop                  ; a нечетное
   jmp @end_if
@a_is_even:
   nop                  ; a четное
@end_if:
   ret
```

2. Команды условного перехода. Знаковые числа

```
if (a1 = b1) then Fsign = 0;
if (a1 < b1) then Fsign = -1;
if (a1 > b1) then Fsign = 1;
```

```
section .data
   a1 dw -1
   b1 dw -2
section .bss
   Fsign resw 1

section .text
   xor eax, eax
   mov ax,  [a1]
   mov bx,  [b1]
   cmp ax,bx
   JE Equal                    ; a1  == b1
   JG Great                    ; a1 > b1
   mov [Fsign], WORD -1  ; a1 < b1
   jmp Cont
Great: mov [Fsign], WORD 1  ; a1 > b1
   jmp Cont
Equal: mov [Fsign], WORD 0 ; a1 == b1
Cont:
   ret
```

```
section .data
   a1 dw -1
   b1 dw -2
section .bss
   Fsign resw 1

section .text
   xor eax, eax
   mov [Fsign], WORD 0 ; a1 == b1
   mov ax,  [a1]
   mov bx,  [b1]
   cmp ax,bx
   JNGE Less                   ; a1 < b1
   JNLE Great                  ; a1 > b1
   jmp Cont                    ; a1 == b1
Great: mov [Fsign], WORD 1  ; a1 > b1
   jmp Cont
Less: mov [Fsign], WORD -1 ; a1 < b1
Cont:
   ret
```

2. Команды условного перехода. Беззнаковые числа

```
if (a1 = b1) then Fsign = 0;
if (a1 < b1) then Fsign = -1;
if (a1 > b1) then Fsign = 1;
```

```
section .data
   a1 dw 1
   b1 dw 2
section .bss
   Fsign resw 1

section .text
   xor eax, eax
   mov ax,  [a1]
   mov bx,  [b1]
   cmp ax,bx
   JE Equal                    ; a1  == b1
   JA Great                    ; a1 > b1
   mov [Fsign], WORD -1  ; a1 < b1
   jmp Cont
Great: mov [Fsign], WORD 1  ; a1 > b1
   jmp Cont
Equal: mov [Fsign], WORD 0 ; a1 == b1
Cont:
   ret
```

```
section .data
   a1 dw 1
   b1 dw 2
section .bss
   Fsign resw 1

section .text
   xor eax, eax
   mov [Fsign], WORD 0 ; a1 == b1
   mov ax,  [a1]
   mov bx,  [b1]
   cmp ax,bx
   JNAE Less                   ; a1 < b1
   JNBE Great                  ; a1 > b1
   jmp Cont                    ; a1 == b1
Great: mov [Fsign], WORD 1  ; a1 > b1
   jmp Cont
Less: mov [Fsign], WORD -1 ; a1 < b1
Cont:
   ret
```

2. Команды условного перехода.

Команды условного перехода реализуют короткий переход, т.е. смещение в пределах **[-128...127**]. Если требуется переход дальше, нужно воспользоваться двумя командами **Jcc** и **JMP**.

```
            JZ @mycode
             ….
@mycode:
            ….
```

```
            JNZ Next
            JMP @mycode
Next:
            ….
@mycode:
            ….
```

3. Условное копирование.

```
  . . .
   cmp eax, ebx
   jne .l1
   mov eax, edx
.l1:
   . . .
```

```
   . . .
   cmp eax, ebx
   cmove eax, edx
. . .
```

CMOVcc — Conditional Move

CMOVcc r16, r/m16

CMOV**cc** r32, r/m32

CMOV**cc** r64, r/m64

CMOV**A** ax, bx

CMOV**A** eax, edx

CMOV**A** rdx, rax

CMOV**A** ax, [bx]

CMOV**A** eax, [var]

CMOV**A** rdx, [rax + 4]

3. Условное копирование.

CMOVcc — Conditional Move

Jcc	CMOVcc
JE	CMOVE
JNE	CMOVNE
JL/JNGE	CMOVL/CMOVNGE
JLE/JNG	CMOVLE/CMOVNG
JG/JNLE	CMOVG/CMOVNLE
JGE/JNL	CMOVGE/CMOVNL
JB/JNAE	CMOVB/CMOVNAE
JBE/JNA	CMOVBE/CMOVNA
JA/JNBE	CMOVA/CMOVNBE
JAE/JNB	CMOVAE/CMOVNB

Jc	CMOVcc
JZ	CMOVZ
JS	CMOVS
JC	CMOVC
JO	CMOVO
JP	CMOVP/CMOVPE
JNZ	CMOVNZ
JNS	CMOVNS
JNC	CMOVNC
JNO	CMOVNO
JNP	CMOVNP/CMOVPO

CMPXCHG — Compare and Exchange

CMPXCHG r/m8, r8

CMPXCHG r/m16, r16

CMPXCHG r/m32, r32

CMPXCHG r/m64, r64

```
if (r/m8 == al)
   r/m8 = r8;
else
   al = r/m8;
```

```
if (n == a)
   n = 2;
```

```
    mov dl, 2
    mov al, [a]

    cmp al, [n]
    jne  @exit
    mov [n], dl
@exit:
              ; al = r/m8 - забываем
```

```
    mov dl, 2
    mov al, [a]

    cmpxcng  [n], dl
```

CMPXCHG8B/CMPXCHG16B

CMPXCHG8B m64

```
if (EDX:EAX == m64)
   m64 = ECX:EBX
else
   EDX:EAX == m64
```

CMPXCHG16B m128

```
if (RDX:RAX == m128)
   m128 = RCX:RBX
else
   RDX:RAX == m128
```

Спасибо