Лабораторна робота № 7
Система команд. Організація умовних переходів

Мета роботи - Вивчення команд умовних переходів і способів їх використання в асемблерних програмах для реалізації розгалужень в обчисленнях.

Теоретичні відомості

Команди переходу забезпечують безумовні переходи усередині поточного кодового сегменту (внутрішній перехід) або з поточного сегменту в інший кодовий сегмент (зовнішній або міжсегментний перехід). Всі ці переходи реалізуються за допомогою команди **JMP**, яка має формат

JMP [type] OPR

де **type** - тип переходу: **SHORT** (короткий), **NEAR** (ближній) або **FAR** (дальній), за замовчуванням приймається **NEAR**;

OPR - ім'я, мітка або адресний вираз. Залежно від типу переходу (**type**) і/або структури операнда **OPR** можна виділити 5 видів команд безумовного переходу, з яких три визначають внутрішньосегментні переходи, а дві - міжсегментні переходи.

```
JMP  SHORT label      ; короткий перехід в межах -128...+127
                      ; байт відносно адреси наступної команди
JMP  [NEAR PTR] label      ; внутрішньосегментний прямий перехід
                      ; по зсуву відносно мітки label
JMP  [NEAR PTR] opr ; внутрішньосегментний непрямий перехід
                      ; opr - регістр або слово в пам'яті, де
                      ; міститься адреса переходу
JMP  [FAR PTR] label ; міжсегментний прямий перехід за
                      ; адресою (Seg: Offset) мітки label
JMP  [FAR PTR] opr    ; міжсегментний непрямий 9 (косвенний) перехід за
                      ; адресою (Seg: Offset), записаною в пам'яті, на
                      ; яку вказує операнд opr
```

Мітка **label** - це безпосередня мітка команди (ім'я із завершуючою двокрапкою) або мітка, визначена в кодовому сегменті за допомогою директиви **LABEL**. Операнд **opr** в командах непрямого переходу являє собою ім'я регістра, в якому міститься адреса переходу (тільки для внутрішніх переходів), або ім'я змінної, де записана адреса переходу (одне слово для внутрішньосегментного переходу або два слова для міжсегментного переходу), або ж адресний вираз, яким визначається адреса пам'яті, де зберігається адреса переходу. Слід підкреслити, що у разі внутрішньосегментного прямого переходу в команді зберігається не пряма адреса переходу (Offset), а зсув (Displacement) в байтах від наступної після **JMP** команди

до мітки переходу, який складається з поточним вмістом **IP** при виконання команди **JMP** і тим самим робить вміст **IP** рівним цільовій адресі (Offset) переходу в поточному сегменті. Це дозволяє переміщати кодовий сегмент в пам'яті без корекції інформації про переходи. У разі прямих міжсегментних переходів в команді зберігається повна логічна адреса переходу (Seg: Offset), яка повинна коректуватися при переміщеннях сегменту, в якому знаходиться точка переходу.

Окрім розглянутих вище команд безумовного переходу, в системі команд x86 є 17 команд умовного переходу, які, як і команда **JMP SHORT label**, забезпечують короткий перехід в сегменті в діапазоні -128 ... +127 байтів відносно адреси наступної команди. Всі вони мають 2-байтний формат; причому в другому байті міститься зсув (ціле із знаком), який розширюється знаковим розрядом до слова і складається з вмістом IP**,** якщо на момент виконання команди задана її мнемокодом умова ВИКОНАНА, інакше цей зсув не додається до IP і, отже, перехід не здійснюється. Деякі команди умовних переходів мають два, а то і 3 різних мнемокоду, наприклад, команди **JB, INAE і JC** мають абсолютно ідентичний машинний код, але різне контекстне "наповнення" програми:

JB label ;обійти перехід, якщо "менше" (CF=l)

JNAE label ; обійти перехід, якщо "не менше і не дорівнюється" (CF=1)

JC label , обійти перехід, якщо виникло перенесення/заем,т. е. CF=

Програміст вибирає з цих трьох рівноцінних команд ту, яка на його думку краще відображає умову переходу. Команди умовного переходу використовуються після арифметичних, логічних і інших команд, що впливають на прапорці, для розгалуження алгоритму залежно від результату виконання команди Особливо важливо правильно використовувати команди умовних переходів після команди порівняння оскільки шукається відмінність між порівнянням чисел із знаком і без знаку:

```
CMP   op1,op2   ; op1 і op2 числа із знаком або без знаку
перехід, якщо   op1 > op2 JG   (JNLE)    JA   (JNBE)
  op1 >= op2    JGE  (JNL)    JAE   (JNB, JNC)
  op1 <  op2    JL   (JNGE)   JB   (JNAE, JC)
  op1 <= op2    JLE  (JNG)    JBE  (JNA)
  op1 =  op2    JE   (JZ)     JE   (JZ)
  op1 <> op2    JNE  (JNZ)    JNE  (JNZ)
```

Якщо потрібно здійснити умовний внутрішньосегментний довгий або навіть міжсегментний умовний перехід, то застосовується наступний прийом.

```
Jcc  No_Jump   ; обхід JMP по протилежній умові
JMP  NEAR PTR або JMP FAR PTR ; по необхідній умові
```

```
No_Jump   ...  ; продовжити, якщо потрібна умова не виконана.
```

Наприклад, якщо за умови AH > AL (без знаку) потрібно перейти на мітку Too_Big в іншому сегменті, то це можна запрограмувати так:

```
CMP  AH,AL      ; порівняти операнди
JBE  Cont       ; обійти перехід, якщо AH <= AL
JMP  FAR PTR Too_Big     ; перейти, якщо AH > AL
Cont ...        ; продовження, якщо немає переходу за умовою
```

Використовуючи команди умовних і безумовних переходів, можна реалізувати різні види розгалуження в програмі, у тому числі і цикли. Але для реалізації циклів з певним числом повторень в системі команд x86 є спеціальні команди управління циклами, кожна з яких при виконанні зменшує на 1 вміст CX, а потім використовує його нове значення для ухвалення рішення про перехід:

```
LOOP  label      ; продовжити з label, якщо CX не рівний 0
LOOPE label      ; продовжити з label, якщо CX не рівний 0
                 ; і ZF-1, т .е. повторення циклу припиняється
                 ; якщо CX = 0 або ZF стане рівний 0
LOOPNE label     ; продовжити з label, якщо CX не рівний 0
                     ; і ZF = 0, тобто повторення циклу припиняється ;если
                 CX = 0 або ZF стане рівний 1
```

Для команди **LOOPE** є еквівалент **LOOPZ**, а для команди **LOOPNE** - **LOOPNZ**. Очевидно, що перед входом в цикл потрібно підготувати лічильник циклів в CX, для запобігання входу в цикл з нульовим значенням CX корисно використовувати команду короткого переходу.

JCXZ label ; перехід, якщо CX = 0

При використанні умовних команд управління циклом в тілі циклу повинні бути команди, що впливають на прапорець ZF.

```
; Фрагмент програми підрахунку кількості позитивних (SI) і
; негативних (DI) елементів в масиві цілих чисел ARR (слова)
; кількість елементів масиву міститься в CX.
; Підрахунок припиняється, якщо виявляться нульовий елемент в масиві
  SUB  BX,BX     ; підготов. покажчик елементів масиву
  MOV  SI,BX     ; і
  MOV  DI,BX     ; лічильники
  JCXZ Exit      ; обхід, якщо нульова кількість елементів
Again: CMP  Агг[BX],0 ; порівняти черговий елемент з нулем
  JE   Exit      ; завершити, якщо нульовий елемент
```

```
    JG   Great     ; > 0
    INC  DI        ; підрахунок негативних
    JMP  SHORT Next
Great   INC  SI    ; підрахунок позитивних
Next:   ADD  BX,2 ; покажчик на наступний елемент
    LOOP Again     ; продовжити, якщо не всі оброблені
    EXIT ...
```

Команди умовного переходу зручно застосовувати для перевірки різних умов. Нижче наведений перелік команд умовного переходу, відповідні прапорці та умови переходу.

Команда	Стан прапорців, які перевіряються	Умови переходу
JA	CF = 0 и ZF = 0	якщо вище
JAE	CF = 0	якщо вище або дорівнює
JB	CF = 1	якщо нижче
JBE	CF = 1 или ZF = 1	якщо нижче або дорівнює
JC	CF = 1	якщо перенос
JE	ZF = 1	якщо дорівнює
JZ	ZF = 1	якщо нуль
JG	ZF = 0 и SF = OF	якщо більше
JGE	SF = OF	якщо більше або дорівнює
JL	SF <> OF	якщо менше
JLE	ZF=1 или SF <> OF	якщо менше або дорівнює
JNA	CF = 1 и ZF = 1	якщо не вище
JNAE	CF = 1	якщо не вище або дорівнює
JNB	CF = 0	якщо не нижче
JNBE	CF=0 и ZF=0	якщо не нижче або дорівнює
JNC	CF = 0	якщо немає переносу
JNE	ZF = 0	якщо не дорівнює
JNG	ZF = 1 или SF <> OF	якщо не більше
JNGE	SF <> OF	якщо не більше або дорівнює
JNL	SF = OF	якщо не менше
JNLE	ZF=0 и SF=OF	якщо не менше або дорівнює
JNO	OF=0	якщо немає переповнення
JNP	PF = 0	якщо кількість одиничних бітів результата є непарною (непарний паритет)

JNS	SF = 0	якщо знак плюс(знаковий (старший) біт результата дорівнює нулю)
JNZ	ZF = 0	якщо немає нуля
JO	OF = 1	якщо перерповнення
JP	PF = 1	якщо кількість одиничних бітів є парною (парний паритет)
JPE	PF = 1	теж що і JP (парний паритет)
JPO	PF = 0	теж що і JNP (непарний паритет)
JS	SF = 1	якщо знак мінус (знаковий (старший) біт результата дорівнює 1)
JZ	ZF = 1	якщо нуль

Логічнй умови «більше» та «менше» відносяться до порівнянь цілочисельних значень зі знаком, умови «вище» та «нижче» - до порівнянь цілочисельних значень без знака. Є декілька мнемонічних позначень однієї і тієї ж команди. Це пояснюється тим, що для мікропроцесора i8086 команди умовного переходу могли здійснювати тільки короткі переходи в межах -128 до +127, починаючи від наступної команди. Починаючи з мікропроцесора i386, ці команди могли виконувати будь-які переходи в межах поточного сегмента команд. Це стало можливим шляхом введення в систему команд мікропроцесора додаткових команд. Для переходу між сегментами треба комбінувати команди умовного перехода та команду безумовного переходу jmp.

Застосування *jcxz/jecxz*:

Команда	Стан прапорців в eflags/flags	Умови переходу
JCXZ	не впливає	якщо регістр CX=0
JECXZ	не впливає	якщо регістр ECX=0

Команду jcxz/jecxz зручно використовувати для організації цикла та ланцюжкових команд, які використовують регістр *ecx/cx*. Ця команда виконує тільки близькі переходи в межах -128 до +127 байт, починаючи від наступної команди. Її використовують для попередньої перевірки лічильника цикла в регістрі cx для того, щоб обійти цикл, якщо його лічильник єнульовим.

Наприклад,
```
    jcxz    m1      ;обійти цикл, якщо cx=0
 cycl:
     ; деякий цикл
     loop    cycl
     ............
m1:     ...
```

Контрольні питання

1. Від чого залежить довжина команд безумовного переходу?
2. Яка інформація про перехід міститься в команді для безумовного переходу усередині сегменту?
3. На які групи можна розділити команди умовних переходів
4. Які способи адресації можна використовувати для непрямого внутрішньосегментного переходу?
5. Які можливості є для здійснення прямого міжсегментного переходу?

Індивідуальні завдання

Обчислити умовний цілочисельний вираз у форматах Integer та Word, використовуючи команди порівняння. Результат перевірити на область допустимих значень.

Варіанти

1) $X=\begin{cases}(a-b)/a+1, & \text{если } a>b,\\ 25, & \text{если } a=b,\\ (a-5)/b, & \text{если } a<b;\end{cases}$

2) $X=\begin{cases}(a-b)/a-3, & \text{если } a>b,\\ 2, & \text{если } a=b,\\ (a^3+1)/b, & \text{если } a<b;\end{cases}$

3) $X=\begin{cases}b/a+5, & \text{если } a<b,\\ -5, & \text{если } a=b,\\ (a*a-b)/b, & \text{если } a>b;\end{cases}$

4) $X=\begin{cases}a/b+10, & \text{если } a<b,\\ -51, & \text{если } a=b,\\ (a*b-4)/a, & \text{если } a>b;\end{cases}$

5) $X=\begin{cases}a/b-1, & \text{если } a>b,\\ -25, & \text{если } a=b,\\ (b^3-5)/a, & \text{если } a<b;\end{cases}$

6) $X=\begin{cases}a/b-1, & \text{если } a>b,\\ -25, & \text{если } a=b,\\ (b^3-5)/a, & \text{если } a<b;\end{cases}$

7) $X=\begin{cases}52*b/a+b, & \text{если } a>b,\\ -125, & \text{если } a=b,\\ (a-5)/b, & \text{если } a<b;\end{cases}$

8) $X=\begin{cases}(a*b-1)/a, & \text{если } a>b,\\ 255, & \text{если } a=b,\\ (a-5)/b, & \text{если } a<b;\end{cases}$

9) $X=\begin{cases}1-b/a, & \text{если } a>b,\\ -10, & \text{если } a=b,\\ (a-5)/b, & \text{если } a<b;\end{cases}$

10) $X=\begin{cases}a/b+31, & \text{если } a>b,\\ -25, & \text{если } a=b,\\ (5*b-1)/a, & \text{если } a<b;\end{cases}$

11) $X=\begin{cases}(2+b)/a & \text{если } a>b,\\ -2, & \text{если } a=b,\\ (a-5)/b, & \text{если } a<b;\end{cases}$

12) $X=\begin{cases}b/a+1, & \text{если } a<b,\\ 25, & \text{если } a=b,\\ (a^3-5)/b, & \text{если } a>b;\end{cases}$

13) $X=\begin{cases}b/a+61, & \text{если } a>b,\\ -5, & \text{если } a=b,\\ (b-a)/b, & \text{если } a<b;\end{cases}$

14) $X=\begin{cases}a/b+1, & \text{если } a>b,\\ -2, & \text{если } a=b,\\ (a-b)/a, & \text{если } a<b;\end{cases}$

15) $X=\begin{cases}(3*a-5)/b, & \text{если } a<b,\\ -4, & \text{если } a=b,\\ (a^3+b)/a, & \text{если } a>b;\end{cases}$

16) $X=\begin{cases}b/a-1, & \text{если } a<b,\\ -295, & \text{если } a=b,\\ (a-235)/b, & \text{если } a>b;\end{cases}$

17) $X=\begin{cases} 2*a/b+1, & \text{если } a>b, \\ -445, & \text{если } a=b, \\ (b+5)/a, & \text{если } a<b; \end{cases}$

18) $X=\begin{cases} a/b+1, & \text{если } a>b, \\ a+25, & \text{если } a=b, \\ (a*b-2)/a, & \text{если } a<b; \end{cases}$

19) $X=\begin{cases} b/a+10, & \text{если } a>b, \\ 3425, & \text{если } a=b, \\ (2*a-5)/b, & \text{если } a<b; \end{cases}$

20) $X=\begin{cases} (a*a-b)/a, & \text{если } a>b, \\ -a, & \text{если } a=b, \\ (a*b-1)/b, & \text{если } a<b; \end{cases}$

21) $X=\begin{cases} (b+1)/a, & \text{если } a>b, \\ -b, & \text{если } a=b, \\ (a-5)/b, & \text{если } a<b; \end{cases}$

22) $X=\begin{cases} a/b-1, & \text{если } a<b, \\ 25-a, & \text{если } a=b, \\ (b-5)/a, & \text{если } a>b; \end{cases}$

23) $X=\begin{cases} b/a+2, & \text{если } a>b, \\ -11, & \text{если } a=b, \\ (a-8)/b, & \text{если } a<b; \end{cases}$

24) $X=\begin{cases} a/b+2, & \text{если } a>b, \\ 8, & \text{если } a=b, \\ (b-9)/a, & \text{если } a<b; \end{cases}$

25) $X=\begin{cases} (b+5)/a, & \text{если } a<b, \\ -5, & \text{если } a=b, \\ (b-a)/b, & \text{если } a>b; \end{cases}$

26) $X=\begin{cases} a/b+1, & \text{если } a<b, \\ -71, & \text{если } a=b, \\ (a-b)/a, & \text{если } a>b; \end{cases}$

27) $X=\begin{cases} b/a-7, & \text{если } a>b, \\ 43, & \text{если } a=b, \\ (a^3-b)/b, & \text{если } a<b; \end{cases}$

28) $X=\begin{cases} -5+b/a, & \text{если } a>b, \\ 45, & \text{если } a=b, \\ (3*a-6)/b, & \text{если } a<b; \end{cases}$

29) $X=\begin{cases} a/b+7, & \text{если } a>b, \\ -125, & \text{если } a=b, \\ (3*b+9)/a, & \text{если } a<b; \end{cases}$

30) $X=\begin{cases} a/b-4, & \text{если } a<b, \\ -55, & \text{если } a=b, \\ (b-5)/a, & \text{если } a>b; \end{cases}$

31) $X=\begin{cases} a/b+20, & \text{если } a>b, \\ 110, & \text{если } a=b, \\ (a-b)/a, & \text{если } a<b; \end{cases}$

32) $X=\begin{cases} a/b+11, & \text{если } a>b, \\ -11, & \text{если } a=b, \\ (3*b-9)/a, & \text{если } a<b; \end{cases}$

Приклад

```
; Умовний перехід
;          ⎧ b*a-50, якщо a>b
;    X=    ⎨ -4,      якщо a=b
;          ⎩ (b-a-5)/b, якщо a<b

    .model small
    .stack 100h
    .data
            cr_lf db 0Ah, 0Dh, '$'
            a dw 20
            b dw 5
            x dw ?
            s dw 5 dup ('?')
            m dw '-'
            z1 dw ?
            z2 dw ?
            mesg db 'Dilennya na 0', '$'
    .code
    start:  mov ax, @data
            mov ds, ax
            mov ax, a
            mov bx, b
            cmp ax, bx
            jg @@3
            jl @@4
    ; a=b
            xor ax, ax
            mov ax, -4
            mov x, ax
            xor ax, ax
            xor bx, bx
            jmp @@minus
    ; a>b
    @@3:
            xor ax, ax
            mov ax, b
            imul a
            sub ax, 50
            mov x, ax
            js @@minus
            jns @@plus
    ; a<b
    @@4:
            xor ax,ax
```

```
        mov ax, b
        cmp ax, 0
        je @@Error
        sub ax, a
        add ax,-5 ; молодш.частина ax=ax+(-5)
        adc dx, 0ffffh ; старша частина dx=dx+(-5)
        cwd
        idiv b
        mov x, ax
        cmp x, 0
        jl @@minus
        jg @@plus
; ділення на 0
@@Error:
        mov ah,09h
        mov dx, offset mesg
        int 21h
        jmp @@Exit
; виведення від'ємного числа
@@minus:
        xor ax, ax
        xor bx, bx
        mov ax, x
        mov bx, m
        mov s, bx
        sub ax, 0001b       ; перетворення
        xor ax, 0FFFFh      ; у позитивне число  або neg ax
        mov z1, ax
        xor ax, ax
        mov ax, z1
        xor bx, bx
        mov bl, 1010b
        div bl
        add ax, 30h
        mov s+1, ax
        xor ax, ax
        mov ax, z1
        div bl
        mul bl
        mov z2, ax
        mov ax, z1
        sub ax, z2
        add ax, 30h
        mov s+2, ax
        mov s+3, '$'
```

```
        mov ah, 09h
        mov dx, offset s
        int 21h
        jmp @@Exit
; виведення позитивного числа
@@plus:
        xor ax, ax
        xor bx, bx
        mov ax, x
        mov bl, 1010b
        div bl
        add ax, 30h
        mov s, ax
        xor ax, ax
        mov ax, x
        div bl
        mul bl
        mov z2, ax
        mov ax, ax
        sub ax, z2
        add ax, 30h
        mov s+1, ax
        mov s+3, '$'
        mov ah,09h
        mov dx, offset s
        int 21h
        jmp @@Exit
@@Exit:
        mov ax, 4c00h
        int 21h
        end start
```