ЛАБОРАТОРНА РОБОТА № 4.

Дослідження роботи команд переходів. Програмування задач з використанням алгоритмів розгалуження.

Мета: освоїти використання команд порівняння, умовного та безумовного переходів. Набути вміння використовувати арифметичні команди над знаковими даними та команди логічних операцій.

Команди керування програмою.

Команда безумовного переходу **JMP**.

Дана команда додає значення вказане в операнді до значення в регістрі EIP, який містить адресу команди, що розташована після JMP. Відповідно, якщо здійснюється перехід назад, то значення операнду буде від'ємне, якщо вперед – додатне.

Команди умовних переходів.

Працюють аналогічно до JMP та LOOP, за винятком того, що довжина переходу повинна бути в межах від -128 до +128 байт.

Переходи для беззнакових даних.

Мнемоніка	Опис	Прапорці, що перевіряються
JE/JZ	Перехід, якщо рівно/нуль	ZF
JNE/JNZ	Перехід, якщо не рівно/не нуль	ZF
JA/JNBE	Перехід, якщо вище/не нижче або рівно	ZF,CF
JAE/JNB	Перехід, якщо вище або рівно/ не нижче	CF
JB/JNAE	Перехід, якщо нижче/не вище або рівно	CF
JBE/JNA	Перехід, якщо нижче або рівно/не вище	CF,AF

Будь яку перевірку можна закодувати двома мнемокодами. Наприклад JB та JNAE генерують однаковий об'єктний код, хоча позитивну перевірку JB легше зрозуміти аніж заперечну JNAE.

Переходи для знакових даних.

Мнемоніка	Опис	Прапорці, що перевіряються
JE/JZ	Перехід, якщо рівно/нуль	ZF
JNE/JNZ	Перехід, якщо не рівно/не нуль	ZF
JG/JNLE	Перехід, якщо більше/не менше або рівно	ZF,SF,OF
JGE/JNL	Перехід, якщо більше або рівно/ не менше	SF,OF
JL/JNE	Перехід, якщо менше/не більше або рівно	SF,OF
JLE/JNG	Перехід, якщо менше або рівно/не більше	ZF,SF,OF

Команди переходу для умови рівно або нуль (JE/JZ) та не рівно або не нуль (JNE/JNZ) існують як для знакових так і для без знакових даних. Стан рівно/нуль наступає незалежно від знаку.

Спеціальні арифметичні перевірки.

Мнемоніка	Опис	Прапорці, що перевіряються
JS	Перехід, якщо є знак(від'ємне)	SF
JNS	Перехід, якщо немає знаку(додатнє)	SF
JC	Перехід, якщо є перенос (аналогічно JB)	CF
JNC	Перехід, якщо немає переносу	CF
JO	Перехід, якщо є переповнення	OF
JNO	Перехід, якщо немає переповнення	OF
JP/JPE	Перехід, якщо паритет парний	PF
JNP/JP	Перехід, якщо паритет непарний	PF

JCXZ (JECXZ) - ще одна команда умовного переходу перевіряє чи рівний вміст регістра CX (ECX) нулю. Ця команда повинна знаходитися безпосередньо після арифметичної команди чи команди порівняння. Одним з можливих місць застосування команди JCXZ

(JECXZ) може бути початок циклу, де вона перевіряє чи не містить регістр CX (ECX) нульового значення.

Оператор циклу **LOOP.**

Передає керування на мітку, вказану як операнд, поки значення в регістрі ECX не стане рівне нулю.

```
        mov  ecx, 5
Lab:
        mov  eax, ebx
        inc eax
        loop Lab
```

тобто оператори (mov eax, ebx) та (inc eax) повторяться 5 разів.

Існують додатково також різновиди цієї команди:

1. LOOPE(або LOOPZ) – передає керування за адресою, вказаною в операнді якщо ECX містить нульове значення і встановлений прапорець нуля, тобто ZF=1.
2. LOOPNE(або LOOPNZ) передає керування за адресою, вказаною в операнді якщо ECX містить нульове значення і прапорець нуля скинуто, тобто ZF=0.

Команди порівняння

CMP - Порівняння двох полів даних.

Команда виконує віднімання другого операнду від першого, але значення операндів не змінює. Операнди повинні бути однакової довжини: байт або слово. Команда може мати наступний формат:

CMP регістр-регістр;
CMP пам'ять-регістр;
CMP безпосереднє значення -регістр;
CMP регістр-пам’ять;
CMP безпосереднє значення –пам’ять

Ця команда впливає на прапорці AF,CF,OF, PF,SF,ZF

CMPS, CMPSB, CMPSW Порівняння рядків довільної довжини. CMPSB – порівнює по байту, CMPSW – порівнює по словах. Перший операнд цих команд адресується регістровою парою DS:SI, а другий – ES:DI. Якщо прапорець DF=0, то порівняння відбувається зліва направо і регістри SI та DI збільшуються після кожного порівняння, інакше – навпаки і регістри SI та DI зменшуються. Як правило, використовуються з префіксами повтору. Впливають на прапорці AF,CF,OF,PF,SF,ZF.

Логічні команди

Логічні команди відіграють важливу роль в організації програм і використовуються для скиду і встановлення біт та для арифметичних операцій. Всі вони обробляють один байти або одне слово в регістрі чи в пам’яті і встановлюють прапорці CF, OF, PF, SF, ZF

Перший операнд в логічних командах вказує на один байт чи слово в регістрі чи пам’яті і є єдиним значенням, яке може змінюватися після виконання команд..

AND Логічне «І». Порозрядна кон’юнкція бітів операндів. Якщо відповідні біти операндів рівні 1, то у першому(лівому) операнді встановлюється біт=1, в інших випадках результат =0.

OR логічне «АБО». Порозрядна диз’юнкція бітів операндів. Якщо хоча б один з порівнюваних бітів рівний 1, то результат рівний 1, якщо порівнювані біти рівні 0, то результат - 0 .

XOR логічне «виключне АБО». Якщо один з порівнюваних бітів рівний 0, а інший 1, то результат рівний 1, якщо порівнювані біти однакові, то результат -0.

NOT логічне «НЕ». Порозрядна інверсія бітів операнду. Встановлює протилежне значення бітів у байті чи слові в регістрі чи в пам’яті. Прапорці при цьому не змінюються.

TEST – Логічне «І» - встановлюються прапорці, проте біти не змінюються.

Приклади використання.

Нехай AL=1100 0101b; BH=0101 1100b.

```
1.  AND  AL,BH    ; Встановлює в AL 0100 0100
2.  OR   BH,AL    ; Встановлює в BH 1101 1101
3.  XOR  AL,AL    ; Встановлює  в AL 0000 0000
4.  AND  AL,00    ; Встановлює в AL 0000 0000
5.  AND  AL,0FH   ; Встановлює  в AL 0000 0101
6.  OR   CL,CL    ; Встановлює прапорці SF и ZF
```

В прикладах 3 та 4 демонструється спосіб очистки регістру. В прикладі 5 встановлюються в нуль ліві чотири біти регістру AL.

Приклад використання команди TEST.

```
1.  TEST BL,11110000B    ; будь який з лівих бітів в BL
      JNZ  ...           ; рівний одиниці ?
2.  TEST AL,00000001B    ; регістр  AL містить
      JNZ  ...           ; непарне значення?
3.  TEST DX,OFFH         ; регістр  DX містить
      JZ   ...           ; нульове значення?
```

Контрольні запитання:

1. Поняття про машинні коди, байт способу адресації та способи адресації.
2. Які команди безумовних переходів ви знаєте?
3. Які команди умовних переходів є в мові Асемблер?
4. Яка різниця в організації умовних переходів для знакових і без знакових даних?
5. Що робить команда CMP ?
6. Що робить команда TEST ?

Література:

1.Р.Джордейн.Справочник програмиста персональных компъютеров типа IBM PC XT и AT. -М."Финансы и статистика",1992,стор.13-31.

2.Л.О.Березко,В.В.Троценко. Особливості програмування в турбо-асемблері. -Київ,НМК ВО,1992.

3.Л.Дао. Программирование микропроцессора 8088.Пер.с англ.-М."Мир",1988.

4.П.Абель.Язык ассемблера для IBM PC и программирования. Пер. з англ.-М.,"Высшая школа",1992.

ЗАВДАННЯ:

1. Створити *.exe програму, яка реалізовує обчислення, заданого варіантом виразу.
   Вхідні дані слід вважати цілими числами зі знаком, розміром один байт.
   Результат обчислення виразу повинен записуватися у пам'ять.
   Уникнути випадку некоректних обчислень при діленні на нуль та при переповненні розрядної сітки (вивести відповідне текстове повідомлення).
2. За допомогою Debug, відслідкувати правильність виконання програми (продемонструвати результати проміжних та кінцевих обчислень) та проаналізувати отримані результати для різних вхідних даних.
3. Скласти звіт про виконану роботу з приведенням тексту програми та коментарів до неї.
4. Дати відповідь на контрольні запитання.

ВАРІАНТИ ЗАВДАННЯ:

A, B - знакові операнди, розміром один байт.

№	вираз	№	вираз	№	вираз
1	$X=\begin{cases}(8*a-b)/a+1, & a>b,\\ 25, & a=b,\\ (a-5)/b, & a<b.\end{cases}$	**11**	$X=\begin{cases}a/b+31, & a>b,\\ -9, & a=b,\\ (5*b-1)/a, & a<b.\end{cases}$	**21**	$X=\begin{cases}3*a/b+1, & a>b,\\ a+25, & a=b,\\ (b-2)/a, & a<b.\end{cases}$
2	$X=\begin{cases}(a-b)/a-3, & a>b,\\ 2, & a=b,\\ (4*a+1)/b, & a<b.\end{cases}$	**12**	$X=\begin{cases}(2+b)/a, & a>b,\\ -2, & a=b,\\ (a-5)/2*b, & a<b.\end{cases}$	**22**	$X=\begin{cases}(4*a-b)/b, & a>b,\\ -a, & a=b,\\ (b-a)/a, & a<b.\end{cases}$
3	$X=\begin{cases}-2*b/a+5, & a<b,\\ -5, & a=b,\\ (a-b)/b, & a>b.\end{cases}$	**13**	$X=\begin{cases}5*b/a+1, & a<b,\\ 12, & a=b,\\ (a-5)/b, & a>b.\end{cases}$	**23**	$X=\begin{cases}(6*b+1)/a, & a>b,\\ -b, & a=b,\\ (a-5)/b, & a<b.\end{cases}$
4	$X=\begin{cases}3*a/b+10, & a<b,\\ -51, & a=b,\\ (b-4)/a, & a>b.\end{cases}$	**14**	$X=\begin{cases}a/b+1, & a>b,\\ -2, & a=b,\\ 2*(a-b)/a, & a<b.\end{cases}$	**24**	$X=\begin{cases}a/b-1, & a<b,\\ 10-a, & a=b,\\ (-2*b-5)/a, & a>b,\end{cases}$
5	$X=\begin{cases}7*a/b-1, & a>b,\\ -25, & a=b,\\ (b-5)/a, & a>b.\end{cases}$	**15**	$X=\begin{cases}(3*a-5)/b, & a<b,\\ -4, & a=b,\\ (a+b)/a, & a>b.\end{cases}$	**25**	$X=\begin{cases}5*b/a+2, & a>b,\\ -11, & a=b,\\ (a-8)/b, & a<b.\end{cases}$
6	$X=\begin{cases}52*b/a+b, & a>b,\\ -125, & a=b,\\ (a-5)/b, & a<b.\end{cases}$	**16**	$X=\begin{cases}-4*b/a-1, & a<b,\\ -295, & a=b,\\ (a-235)/b, & a>b.\end{cases}$	**26**	$X=\begin{cases}2*a/b+2, & a>b,\\ 8, & a=b,\\ (b-9)/a, & a<b.\end{cases}$
7	$X=\begin{cases}(4*b-1)/a, & a>b,\\ 255, & a=b,\\ (a-5)/b, & a<b.\end{cases}$	**17**	$X=\begin{cases}2*a/b+1, & a>b,\\ -445, & a=b,\\ (b+5)/a, & a<b.\end{cases}$	**27**	$X=\begin{cases}-5+b/a, & a>b\\ 45, & a=b,\\ (3*a-6)/b, & a<b.\end{cases}$
8	$X=\begin{cases}a/b+7, & a>b,\\ -125, & a=b,\\ (3*b+9)/a, & a<b.\end{cases}$	**18**	$X=\begin{cases}5*a/b-4, & a<b,\\ -55, & a=b,\\ (b-5)/a, & a>b.\end{cases}$	**28**	$X=\begin{cases}7*a/b+20, & a>b,\\ 110, & a=b,\\ (a-b)/a, & a<b.\end{cases}$
9	$X=\begin{cases}a/b+11, & a>b,\\ -1, & a=b,\\ (2*b-9)/a, & a<b.\end{cases}$	**19**	$X=\begin{cases}-4*b/a+2, & a>b,\\ -57, & a=b,\\ (a-b)/b, & a<b.\end{cases}$	**29**	$X=\begin{cases}a/b+5, & a<b,\\ -b, & a=b,\\ (2*b-9)/a, & a>b.\end{cases}$
10	$X=\begin{cases}-7*b/a+1, & a>b,\\ -271, & a=b,\\ (a-b)/b, & a<b.\end{cases}$	**20**	$X=\begin{cases}6*a/b-37, & a>b,\\ 3, & a=b,\\ (a-b)/a, & a<b.\end{cases}$	**30**	$X=\begin{cases}b/a-2, & a>b,\\ -140, & a=b,\\ (-3*a-100)/b, & a<b.\end{cases}$

Приклад виконання.

Завдання: Написати програму, яка обчислює арифметичний вираз і результат записує в пам'ять. Вхідні дані слід вважати цілими числами зі знаком, розміром один байт. Результат обчислення виразу повинен записуватися у пам'ять. Уникнути випадку некоректних обчислень при діленні на нуль та при переповненні розрядної сітки.

Заданий вираз:

$$X = \begin{cases} a/b + 1, & a < b, \\ -365, & a = b, \\ (b-9)/a, & a > b. \end{cases}$$

Аналіз задачі.

Можливі два випадки ділення на нуль, які слід виявити і видати відповідне повідомлення, не переходячи до безпосередніх обчислень:

1) якщо $a < b$ і $a = 0$ і $b \neq 0$;

2) якщо $a > b$ і $a \neq 0$ і $b = 0$.

Водночас, випадок коли $a = b = 0$ є цілком допустимим.

Оскільки вхідні дані є знаковими, то для організації програми слід використовувати відповідні команди переходів та команди арифметики (в даному випадку ділення) знакових чисел.

Для уникнення можливого переповнення (при виконанні ділення), слід перейти до більшої розрядності операндів, тобто використати формат ділення 4 байт на 2 байти, при цьому сформувавши у відповідних регістрах коректні знакові числа, тобто розмножити старший біт вхідних даних.

Оскільки за умовою завдання всі вхідні дані є однобайтними і у виразі відсутня операція множення, то очевидно, що для результату достатньо передбачити два байти.

Повний текст програми, що реалізовує вказане завдання, та коментарі до неї приведено нижче.

```
.686
.model flat, stdcall
option casemap:none
include \masm32\include\windows.inc
include \masm32\include\kernel32.inc
include \masm32\include\user32.inc
includelib \masm32\lib\kernel32.lib
includelib \masm32\lib\user32.lib

.data

A      db     0
B      db     -127
X      dw     0

Hello        db 13, 10, '  X = a/b + 1    if a < b', 13, 10
             db '  X = -365       if a = b', 13, 10
             db '  X = (b - 9)/a  if a > b', 13, 10
Operands     db 13, 10, '  A =      B =      ', 13, 10
NumberOfCharsToWrite_Hello dd $-Hello
Error        db 13, 10, '  Error - divide by zero!', 13, 10
NumberOfCharsToWrite_Error dd $-Error
Result       db 13, 10, '  X =           ', 13, 10
NumberOfCharsToWrite_Result dd $-Result
format db '%hd', 0
hConsoleOutput dd 0
NumberOfCharsWritten dd 0

.code
```

```
start:
;вивід повідомлення Hello
mov al, A
cbw
push ax
push offset format
push offset [Operands+8]
call wsprintfA
mov al, B
cbw
push ax
push offset format
push offset [Operands+17]
call wsprintfA
push -11
call GetStdHandle
mov hConsoleOutput, eax
push 0
push offset NumberOfCharsWritten
push NumberOfCharsToWrite_Hello
push offset Hello
push hConsoleOutput
call WriteConsoleA

;перевірка на рівність A і B
mov al, A
cmp al, B
jne A_ne_B
mov X, -365
jmp Output_Result

A_ne_B:
jg A_g_B
cmp B, 0
je Output_Error
;обчислення X при a<b
mov al, B
cbw
mov bx, ax
mov al, A
cbw
cwd
idiv bx
add ax, 1
mov X, ax
jmp Output_Result

A_g_B:
cmp A, 0
je Output_Error
;обчислення X при a>b
cbw
mov bx, ax
mov al, B
cbw
sub ax, 9
cwd
idiv bx
mov X, ax
jmp Output_Result

;вивід результату
Output_Result:
push X
push offset format
```

```
push offset [Result+8]
call wsprintfA
push offset NumberOfCharsWritten
push NumberOfCharsToWrite_Result
push offset Result
push hConsoleOutput
call WriteConsoleA
jmp exit

;вивід повідомлення про ділення на нуль
Output_Error:
push offset NumberOfCharsWritten
push NumberOfCharsToWrite_Error
push offset Error
push hConsoleOutput
call WriteConsoleA
jmp exit

;вихід з програми
exit:
push 0
call ExitProcess
end start
```