**Computer Organization and Assembly Language**

|  |  |
| --- | --- |
| **Lab 07** | |
| **Topic** | * JMP instructions * Conditional/Unconditional JMP instructions. |

***Part 1***

**JUMP INSTRUCTIONS**

**Two main types of jump instructions**

1. Unconditional jump:

To start executing instructions unconditionally

JMP start ; unconditional JMP instruction

VAR1 DB 5

VAR2 DW 77

start: ; start: is a label/address in a program

MOV AX, 2;

ADD AX, 1;

.EXIT

1. Conditional jump:

To start executing instructions based on some condition.

Some conditional jumps are as follows:

* JC;Jump if carry flag set
* JNC ;Jump if not carry
* JZ ;Jump if zero flag set
* JNZ ;Jump if zero flag not set
* JE ;Jump if zero flag set
* JNE;Jump if zero flag not set
* JS;Jump if sign flag is set
* JNS ;Jump if sign flag not set
* JP ;Jump if parity flag is set
* JNP ;Jump if not parity
* JO;Jump if overflow;
* JNO ;Jump if not overflow

Conditional jumps after signed operand comparison

* JG Jump if greater
* JNG Jump if not greater
* JGE Jump if greater or equal
* JNGE Jump if not greater or equal
* JL Jump if less
* JNL jump if not less
* JLE Jump if less or equal
* JNLE jump if not less or equal

Conditional jumps after unsigned operand comparison

* JA Jump if above
* JNA Jump if not above
* JAE Jump if above or equal
* JNAE Jump if not above or equal
* JB Jump if below
* JNB Jump if not below
* JBE Jump if below or equal
* JNBE jump if not below or equal

**Example: Conditional jmp instruction Example**

In this example it compares two number and stores that value in result variable.

mov al,num1

mov bl,num2

cmp al,bl

jng less: ; if AL is Less than BL

;greater

mov result,al ; Means AL is greater

jmp exit:

less:

mov result,bl ; Means BL is greater

exit:

ret

num1 db 0x20

num2 db 0x30

result db ?

**Example: Signed number comparison**

In this example it compares two number and stores that value in result in ax.

MOV AL, 5

MOV BL, -1

CMP AL, BL

JG L1 ; signed statement; jump if greater …

MOV AX, 0

JMP Exit

L1:

MOV AX, 1

Exit:

ret

**Example: Unsigned number comparison**

MOV AL, 5

MOV BL, -1

CMP AL, BL

JA L1 ; unsigned statement; jump if above …

MOV AX, 0

JMP Exit

L1:

MOV AX, 1

Exit:

ret

Note: In unsigned notation 5 is less than -1=FF=255

**IMUL INSTRUCTION**

One main difference in MUL and iMUL instruction is that iMUL is used when operands are treated as signed numbers

MOV AL, -1 ; AL=FF is (255 in unsigned format)

MOV BL, 2

MUL BL ; Produces AX=01FE hex (510 in decimal)

; Unsigned product considering unsigned operands.

MOV AL, -1 ; AL=FF is (-1 in signed format)

MOV BL, 2

IMUL BL ;Produces AX=FFFE hex (-2 in decimal)

;Signed product considering signed operands.

IDIV INSTRUCTION

One main difference in DIV and iDIV instruction is that IDIV is used when operands are treated as signed numbers.

MOV AL, 5 ; AL=05

MOV BL, -2 ; BL=FE (254 in decimal in unsigned notation)

DIV BL ; Produces Quotient AL=00 and Remainder AH=05

; considering unsigned operands.

MOV AL, 5 ;AL=05

MOV BL,-2 ; BL=FE (-2 in decimal in signed notation)

IDIV BL ; Produces Quotient AL=FE hex (-2 in decimal) and

;Remainder AH=01 considering unsigned operands

***Part 2*** (10 marks each)

**Task 1:**

Write a program to find whether number stored in AX is positive or negative. If number is Positive then store zero in BX else store one in BX.

Ans:

jmp start

num1 dw 0x5

start:

mov ax,num1

mov bx,0

cmp ax,bx

JL loop1

JG loop2

loop1:

mov bx,1

jmp exit

loop2:

mov bx,0

jmp exit

exit:

ret

**Task 2:**

Write an assembly language program to find the sum of first 20 odd numbers in a variable of size word.

Ans:

jmp Start

num1 dw 0x0

Start:

mov cx,20

mov ax,num1

mov bx,1

loop1:

add ax,bx

add bx,2

dec cx

cmp cx,0

jnz loop1

ret

**Task 3:**

Declare a word type array of 10 element then write a program that counts even elements in array. And Store count in memory.

**Ans:**

**jmp Start**

**num1 dw 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x9, 0x10**

**counter dw 0**

**Start:**

**mov cx,10**

**lea si,num1**

**loop1:**

**mov ax,[si]**

**and ax,0x01**

**jz loop2**

**loop3:**

**add si,2**

**dec cx**

**cmp cx,0**

**jnz loop1**

**ret**

**loop2:**

**add counter,1**

**jmp loop3**

**Task 4:**

Write an assembly language program to find the maximum and minimum number in an array of unsigned numbers?

Let array:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Value | 0xAB | 0x7E | 0xAB | 0xED | 0x4B | 0x44 | 0xAA | 0xCC | 0x12 | 0xEE |

Ans:

jmp Start

Value dw 0xAB,0x7E,0xAB,0xED,0x4B,0x44,0xAA,0xCC,0x12,0xEE

max dw 0

min dw 0

Start:

mov si,0

mov cx,10

mov ax,[Value+si]

mov min,ax

loop1:

mov ax,[Value+si]

cmp ax,[min]

jna loop2

loop4:

cmp ax,[max]

ja loop3

loop5:

add si,2

dec cx

cmp cx,0

jnz loop1

ret

loop2:

mov min,ax

jmp loop4

loop3:

mov max,ax

jmp loop5

**Task 5:**

Write an assembly language program to find the second maximum even number from an array of 10 elements taking signed numbers?

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Value | 0xAB | 0x7E | 0xAB | 0xED | 0x4B | 0x44 | 0xAA | 0xCC | 0x12 | 0xEE |

Ans: