University of Central Punjab
(Incorporated by Ordinance No. XXIV of 2002 promulgated by Government of the Punjab)
FACULTY OF INFORMATION TECHNOLOGY

Computer Organization and Assembly Language

Lab 05	
Topic	• JMP instructions • Conditional/Unconditional JMP instructions (if-else) • Memory addressing modes, Data movement, Array, addition and subtraction instructions.

JUMP INSTRUCTIONS

Two main types of jump instructions

a) Unconditional jump:

To start executing instructions unconditionally

```
1  ;Unconditional Jump
2
3  [org 0x100]
4
5  JMP SKIP ; unconditional jmp instruction
6
7  MOV AX, 2;
8  ADD AX, 1;
9
10 SKIP:       ;SKIP IS A LABEL
11 MOV AX,7
12
13 MOV AX, 0X4C00
14 INT 21H
15
16 VAR1 : BD 5
17 VAR2 : BW 77
```

b) Condtional 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
JE	Jump if zero flag set
JNZ	Jump if zero flag not set
JNE	Jump if zero flag not set
JS	Jump if sign flag is set
JNS	Jump if sign flag not set
JO	Jump if overflow
JNO	Jump if not overflow

1. 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

2. 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

- **Compare Instruction**

  cmp operand1,operand2

  It subtracts operand2 from operand1 and updates the flags only ***without updating the value of the operands.***

Note:			**Considering 8-bits**
Unsigned	**5**	**<**	**255(0xFF)**
Signed	**5**	**>**	**-1(0xFF)**

Example 1: Signed number comparison

In this example it compares two numbers and stores **1** in ax if *al* is greaterthan *bl* else **0**.

```
;EXAMPLE 1

[org 0x100]
MOV AL, 5
MOV BL, 0xFF
CMP AL, BL
JG L1 ; signed statement; jump if greater …
MOV AX, 0

JMP Exit

L1:
MOV AX, 1

Exit:
mov ax,0x4c00
int 21h
```

Example 2: Unsigned number comparison

```
;EXAMPLE 2

[org 0x100]
 MOV AL, 5
 MOV BL, 0xFF
 CMP AL, BL
JA L1   ; unsigned statement; jump if above … MOV AX, 0
JMP Exit

 L1:
MOV AX, 1

Exit:
mov ax,0x4c00 int 21h
```

- ArrayName: db 1,2,3,4,5,6,7,8,9,10; *This will reserve 10 byte sized values in consecutive memory locations.*
- ArrayName: dw 1,2,3,4,5,6,7,8,9,10; *This will reserve 10 word sized values 20 bytes in consecutive memory locations.*

- var: *times* 10 db 7; Initializing array var with total 10 elements with value 7**.**

Sample Code 1: (Access data from an array….)

TO ACCESS NEXT ELEMENTS WITHIN AN ARRAY ADD OFFSETS TO ARRAY NAME DEPENDING UPON ARRAY SIZE TYPE.

```
;SAMPLE CODE 1

[org 0x100]
MOV AL, [ARRAY1+1]; THIS WILL LOAD SECOND ELEMENT FROM ARRAY1
MOV BX, [ARRAY2+2]; THIS WILL LOAD SECOND ELEMENT FROM ARRAY2
mov ax,0x4c00

int 21h

ARRAY1 DB 1,2,3,4,5;
ARRAY2 DW 0XA, 0XB, 0XC, 0XD, 0XE;
```

Sample Code 2: Access data from an array (Indexed register Indirect mode)

```
[org 0x100]
mov SI, VEC1     ;Loads the effective address of array VEC1 in SI
mov DI, VEC2    ;Loads the effective address of array VEC2 in DI

L1:
MOV AL, [SI]
MOV [DI], AL

INC SI; ;adds '1' to the destination operand.
INC DI;

DEC byte [COUNT];   Subtracts '1' from the destination operand.
JNZ L1; Repeats executing instructions from label L1 until last arithmetic operation
;produces 0.
mov ax,0x4c00
int 21h

VEC1: DB 1, 2, 5, 6,8
VEC2: DB 0,0,0,0,0
COUNT: DB 5
```

Practice Tasks

Problem #1:
Write the program that will take an array of 5 elements as input and find minimum value from it and store in the max variable.
Input:
Var: db 5,10,7,6,22
Min: db 0
Note:
After execution 5 will be store in min variable.

Problem #2:
Write the program that will take an input in variable marks and check the following things:

- if marks >=90 then store grade=0xA
- if marks >= 70 and <90 then store grade =0xB
- if marks >=50 and <70 then store grade =0xC
- if marks <50 then store grade =0XF

Marks: db 85
Grade: db 0
Note:
After execution 0xB will be store in grade variable.