ALMANARA UNI.

Mechatronics Eng>

Session 5

Microprocessors & Assembly Language

Session Objectives

- Gain practical hands on reuse of a piece of program.
- How to label.
- How to use call and return.
- Effect of use and call on stack.
- Conditional & unconditional jump.
- Use of TEST and CMP with conditional jump.

CALL RETURN

When **calling** an internal subroutine,

CALL passes control to a label specified after the **CALL** keyword.

When the external subroutine completes,

RETURN instruction is used to **return** to where you left off in the **calling** program.

CALL RETURN

Two instructions control the use of **assembly-language** procedures:

CALL: pushes the return address onto the stack and transfers control to a procedure.

RET: pops the return address off the stack and returns control to that location.

Types of calls

- 1)Near Call or Intra segment call
- 2)Far call or Inter Segment call

Operation for Near Call

- When 8086 executes a near CALL instruction:
  - it decrements the stack pointer by 2
  - copies the IP register contents on to the stack.
  - Then it copies address of first instruction of called procedure.

SP ←SP-2

IP ← stores onto stack

IP ←starting address of a procedure.

Operation of FAR CALL

- When 8086 executes a far call, it
- decrements the stack pointer by 2 and copies the contents of CS register to the stack.
- It the decrements the stack pointer by 2 again and copies the content of IP register to the stack.
- Finally it loads cs register with base address of segment having procedure and IP with address of first instruction in procedure.

SP ←sp-2
cs contents → stored on stack
SP ← sp-2
IP contents → stored on stack
CS ← Base address of segment having procedure
IP ← address of first instruction in procedure.

How to label

Label names: meaningful, make program easier to read and maintain.

Naming Rules:

label name must be unique.

Names for labels in assembly language programming consist of:

- Alphabetic letters (upper and lower case)
- Digits 0 through 9,
- Special characters question mark (?), period (.), at (@), underline (_), and a dollar sign ($).

First character of label ***must be:*** alphabetic character (cannot be a number).

Reserved words ***must not*** be used as labels in the program (Specially mnemonics for the instructions).

Store and move subroutine

```
                    ORG 100H
CALL FILL
CALL  MOVE
INT 5H
;
FILL:
        CLD;        Reset Dirction Flag
            ;       =count forward
    ; load source into ds:si,
    ; load target into es:di:
    ; load AL with value to be stored:
        MOV SI, 0200H
        MOV DI, 0200H
        MOV AL, 0CDH
        MOV CX,20H
        REP STOSB
        RET
```

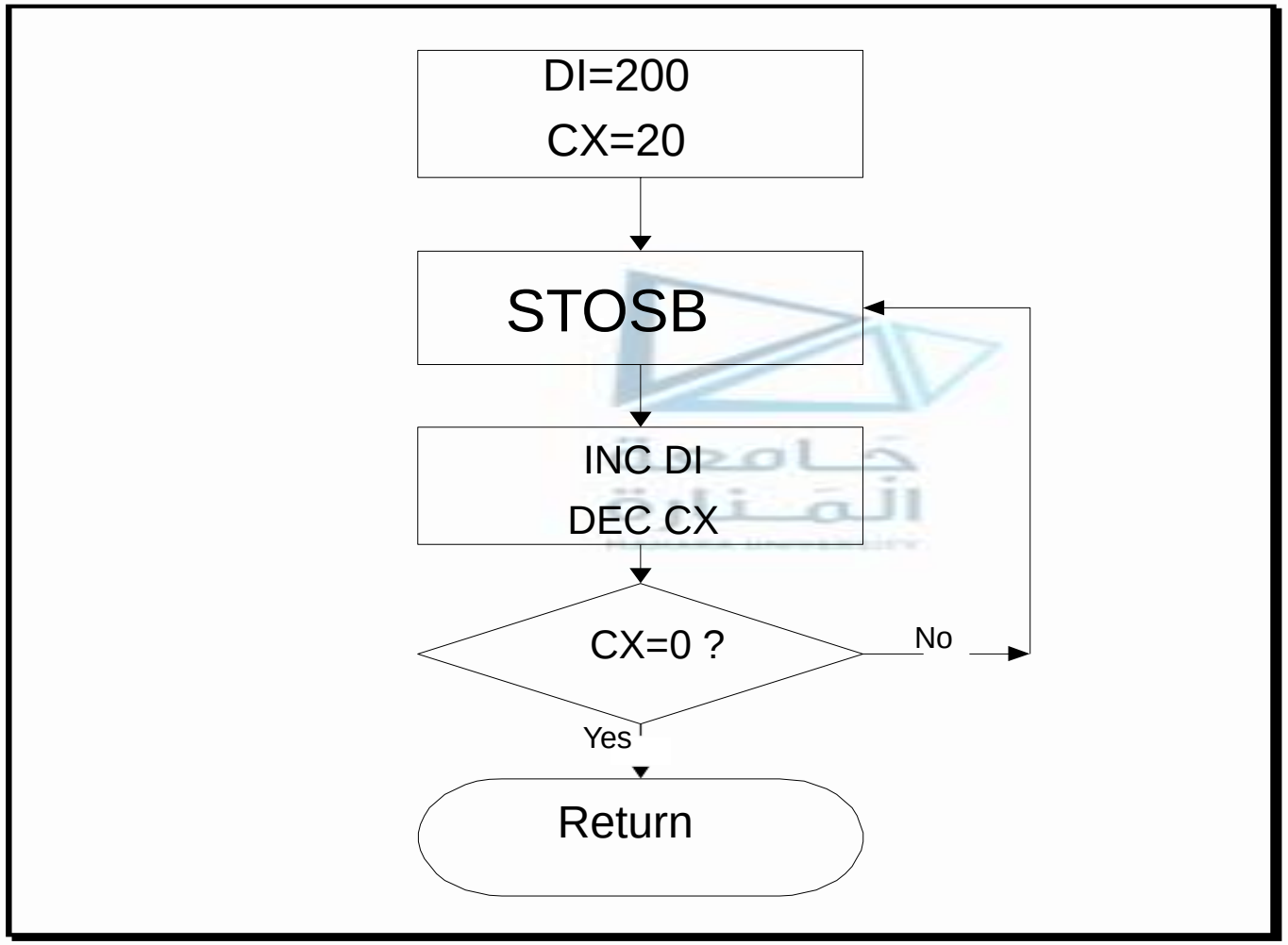
```
MOVE:   MOV SI, 0200H
        MOV DI, 0240H
        MOV CX,20H
        REP MOVSB
        RET
```

Using STOSB to Fill range 0100:0200 to range 0100:021F WITH CD

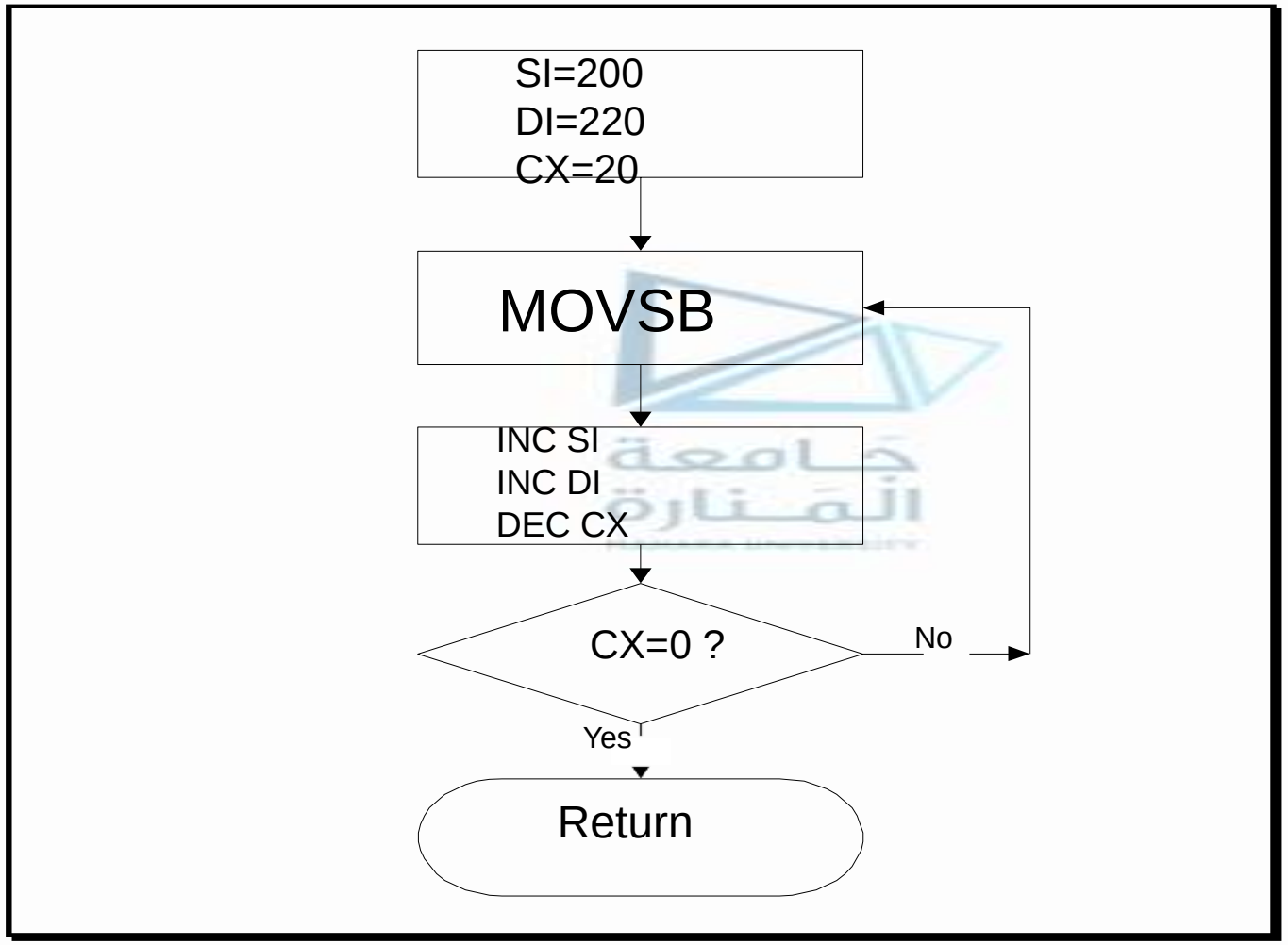
- move range 0100:0200 to 0100:021F
- To the range 0100:0220 to 0100:023F

- 21F-200=1F
- DS=0100, CS=0100, CX=20
- DI=0200, DF=0

ALGORITHM OF FILLING



ALGORITHM OF MOVING



MULTIPLICATION MUL
FLAGS &

MUL reg/ mem	
MUL reg	**For byte : (AX) ← (AL) x (reg8)** **For word : (DX)(AX) ← (AX) x (reg16)**
MUL mem	**For byte : (AX) ← (AL) x (mem8)** **For word : (DX)(AX) ← (AX) x (mem16)**
IMUL reg/ mem	
IMUL reg	**For byte : (AX) ← (AL) x (reg8)** **For word : (DX)(AX) ← (AX) x (reg16)**
IMUL mem	**For byte : (AX) ← (AX) x (mem8)** **For word : (DX)(AX) ← (AX) x (mem16)**

CMP & TEST

The **cmp** instruction is like the subtraction instruction, but it does not store the result anywhere. It just sets condition codes.

- NOTE: the operand order can be confusing

CMP & TEST

Using CMP fill 16 location (bytes) of the range 0700:0200 with numbers 1, 2, 3, 4…..:

```
MOV AL,00H
mov si, 0200h
mov cx,20h

FILL:
MOV [SI],AL
INC AL
INC SI
CMP AL,10H
JNZ FILL
```

CMP & TEST

Using TEST fill 16 location (bytes) of the range 0700:0200 with numbers 1, 2, 3, 4…..:

```
MOV AL,00H
mov si, 0200h
mov cx,20h

FILL:
MOV [SI],AL
INC AL
INC SI
TEST AL,10H
JZ FILL
```