Assembly Language: [4]

Array dicloration:

BARRAY DB 10H, 20H, 30H
content

DUP operation:

DUP ⇒ Copy or duplicate the operator

Character string:

LETTERS DB 'ABC'

the assembler differentiates between upper and lower case.

MSG DB 'HELLO', 0AH, 0DH, '$'

MOV:

The mov instruction is used to transfer data between registers, between a register and a memory location, or to move a number directly into a register or memory location.

"MOV destination, source"

Emistat® Ondansetron USP
Healthcare
DENVAR® CEFIXIME USP
CamScanner

XCHG:

The Xchg operation is used to exchange the contents of two registers, and a memory location.

XCHG destination, Source

1. MOV Instruction - legal Operand Combination

Destination Operand

Destination operand	General Register (GR)	Segment Register	Memory Location	Constant
GR	Yes	Yes	Yes	NO
SR	Yes	NO	Yes	NO
ML	Yes	Yes	NO	NO
C	Yes	NO	Yes	NO

2. XCHG Instruction - legal operand combination

Source operand	GR	ML
GR	Yes	Yes
ML	Yes	NO

CS CamScanner

ADD and SUB:

ADD destination, Source

SUB destination, Source

Source Operand	GR	ML
GR	Yes	Yes
ML	Yes	NO

INC and DEC:

INC → by default add 1 → INC destination

DEC → by default sub 1 → DEC destination

NEG:

Use to Negate the contants of the destination. [by doing 2's complement]

NEG destination

High Level	Low Level
B = A	MOV AX, A; MOV B, AX;

Emistat® Ondansetron USP
Healthcare
DENVAR® CEFIXIME USP
CamScanner

High Level: → Low Level

A = 5 - A

```
MOV AX, 5
SUB AX, A   [5-A]
MOV A, AX
```

or,

```
NEG A      -> -A
ADD A, 5   -> (-A)+5
           => 5+(-A)
```

A = B - 2 × A

```
MOV AX, B;
Sub AX, A;   [B-A]
Sub AX, A;   [B-A-A]
MOV A, AX;
```

Memory Models:

size of code and data, a program can have is determined by specifying a memory model using .MODEL directive

Default → .MODEL small

Data Segment:

.data → directive to declare a data segment

↓

variable declear

STACK SEGMENT:

Purpose of the stack segment declaration is to set aside a block of memory to store the stack.

default → .STACK 100H

Code Segment:

The code segment contain's a program's instruction.

.CODE name

inside a code segment, instruction are organized as procedures.

name PROC

name ENDP

Emistat® Ondansetron USP
Healthcare
DENVAR® CEFIXIME USP
CamScanner

Putting together:

- MODEL SMALL
- STACK 100H
- DATA → data delinations go here
- CODE

MAIN PROC
→ instruction go heare

MAIN ENDP

END MAIN

All DOS

INT → I/O service (interrupt)
DOS (Disk Operating System)

INT 21H → A particular function is requested by placing a function number in the AH register and invoking Int 21H

```
MOV AH, 1;
INT 21h
```
→ input key function

```
MOV AH, 2 [display char]
MOV DL,'?' [Char '?']
INT 21h [display char]
```
(function)

Output Show ←

```
MOV AH, 4CH
INT 21H
```
→ Return 0
→ ending

DX = offset address of string
[The string must end with a '$' character]

LEA → Load effective address

LEA destination, source

→ Array এর নির্দিষ্ট element এর address বের করা
→ Pointer মান সেট করা
→ offset হিসেব করা

```
MSG  DB 'HELLO!$'
LEA  DX, MSG ;→ get message
MOV  AH, 9   ]
INT  21h     ] → display string function
```

Assembly Language → Loop / Decision making (4)

The JUMP Instructions → JXXX des_label

Conditional Jump ~~ist~~ instructions:
JZ, JG, JL, JNZ, JNG, JNL, JNGE, JNLE,

→ JG == JNLE
JL == JNGE

Unconditional Jump instruction:
JMP

Emistat Ondansetron USP
Healthcare
DENVAR CEFIXIME USP
CamScanner

The CMP Instructions

CMP destination, Source

↳ দুটো মানের (Source - destination)
But not affecting destination.
Only change in flag.

Conditional JUMP instructions:

Signed Jumps

Symbol	→ Description	→ condition for Jumps
JG/JNLE	# Jump if greater than # Jump if not less than or equal to	ZF=0 and SF=OF
JGE/JNL	# Jump if greater than or equal to # Jump if not less than or equal to	SF=OF
JL/JNGE	# Jump if less than # Jump if not greater or equal	SF < > OF
JLE/JNG	# Jump if less than or equal # Jump if not greater than	ZF=1 or SF < > OF

CamScanner

Unsigned Conditional Jumps:

Symbol	Description	Condition for Jump
JA/JNBE	# Jump if above # Jump if not below or equal	CF=0 and ZF=0
JAE/JNB	# Jump if above or equal # Jump if not below	CF=0
JB/JNAE	# Jump if below # Jump if not above or equal	CF=1
JBE/JNA	# Jump if equal # Jump if not greater then	ZF=1 OR SF <> OF

Syntex:

JMP label

This instruction will not test any condition, just move the control to the label.

Emistat Ondansetron USP
Healthcare
DENVAR CEFIXIME USP
CamScanner

Single-Flag Jumps:

Symbol	Description	Condition for Jumps
JE/JZ	# Jump if equal # Jump if equal to zero	ZF = 1
JNE/JNZ	# Jump if not equal # Jump if not zero	ZF = 0
JC	# Jump if carry	CF = 1
JNC	# Jump if No carry	CF = 0
JO	# Jump if Overflow	OF = 1
JNO	# Jump if not Overflow	OF = 0
JS	# Jump if sign negative	SF = 1
JNS	# Jump if non-negative	SF = 0
JP/JPE	# Jump if Parity even	PF = 1
JNP/JPO	# Jump if parity odd	PF = 0

Branching structure:

If-then (If) # If-then-else (If, else) # Case (Switch case, if, elseif, else)