8086 Memory Addressing Modes

Course Teacher:

Md. Obaidur Rahman, Ph.D.
Associate Professor
Department of Computer Science and Engineering (CSE)
Dhaka University of Engineering & Technology (DUET), Gazipur.

Course ID: CSE - 3811
Course Title: Microprocessor and Assembly Language

Lecture References:

- **Book:**
  - *Microprocessors and Interfacing: Programming and Hardware, Chapter # 2,* **Author:** Douglas V. Hall
  - *Assembly Language Programming And Organization of the IBM PC, Chapter # 4,* **Author:** Ytha Yu, Charles Marut.

- **Lecture Materials:**
  - *IBM PC Organization,* CAP/IT221
  - M.A. Sattar, Microprocessor and Microcontroller

Memory Model Segmentation

- **.model small**
  - Most widely used memory model.
  - The code must fit in 64k.
  - The data must fit in 64k.
- **.model medium**
  - The code can exceed 64k.
  - The data must fit in 64k.
- **.model compact**
  - The code must fit in 64k.
  - The data can exceed 64k.
- **.medium and .compact are opposites.**

Basic Structure of Assembly Language Program

```
.MODEL SMALL/COMPACT/MEDIUM
.STACK 100H
.DATA
----
----
.CODE
MAIN PROC
   MOV AX, @DATA
   MOV DS, AX
….
---
MAIN ENDP
   END MAIN
RET
```

How to Define Data Segment

- **db:** define byte
- **dw:** define word (2 bytes)
- **equ:** define a numeric constant to a name

```
.data

avariable DB 100 ; Define a data with 1-byte

astring DB "Hello" ; define 5 consecutive bytes with  ASCII valuees

maxint equ 35535  ; define maxint=35535
```

How to Define Stack Segment

.STACK size

Where, size is an optional number that specifies the stack area size in bytes.

.STACK 100h

Here, it sets 00100h bytes for the stack. If size is omitted then 1 KB size is set aside for the stack.

Addressing Mode and It's Categories

- The different ways in which a microprocessor can access data are referred to as its addressing modes.

- Addressing modes of 8086 Microprocessor are categorized as:
  - *Addressing Data*
  - *Addressing Program codes in memory*
  - *Addressing Stack in memory*
  - *Addressing I/O*
  - *Implied addressing*

1. Addressing Data

I. Immediate addressing
II. Direct addressing
III. Register [direct] addressing
IV. Register indirect addressing
V. Base-plus-index addressing
VI. Register relative addressing
VII. Base-relative-plus-index addressing

1. Addressing Data

I. Immediate addressing

- Data is immediately given in the instruction

MOV BL, 44

II. Direct addressing

- Data address is directly given in the instruction

MOV AX, [1234] ; AX ← Value in (DSx10h+1234)
MOV BX, DATA

1. Addressing Data

III. Register [direct] addressing

- Data is in a register (here BX register contains the data)

MOV AX, BX

IV. Register [indirect] addressing

- Register supplies the address of the required data

MOV CX, [BX] ; CX ← Value in (DSx10h+BX)

1. Addressing Data

v. Base-plus-index addressing

- Base register is either BX or BP
- Index register is either DI or SI

MOV DX, [BX+DI] ; DX ← Value in (DSx10h+BX+DI)

vi. Register relative addressing

- Register can be a base (BX, BP) or an index register (DI, SI)
- Mainly suitable to address array data

MOV AX, [BX+1000] ; AX ← Value in (DSx10h+BX+1000)

1. Addressing Data

VII. Base-relative-plus-index addressing

- Suitable for array addressing

MOV AX, [BX+DI+10] ; AX ← Value in (DSx10h+BX+DI+10)

2. Addressing Program Codes in Memory

- Used with JMP and CALL instructions
- 3 distinct forms:
  - Direct
  - Indirect
  - Relative

2. Addressing Program Codes in Memory

- Address is directly given in the instruction

  JMP 1000: 0000

  or **JMP doagain** ; doagain is a **label** in code

  CALL 1000:0000

  or **CALL doagain** ; doagain is a **procedure** in code

- Often known as *far* jump or *far* call

2. Addressing Program Codes in Memory

- Address can be obtained from
  - **a)** any GP registers (AX,BX,CX,DX,SP,BP,DI,SI)
  - **b)** any relative registers ([BP],[BX],[DI],[SI])
  - **c)** any relative register with displacement

JMP [DI] ; Jump to memory location (DSx10h + DI)

CALL [BX] ; Call the content in memory location (DSx10h + BX)

3. Addressing Stack in Memory

- **PUSH** and **POP** instructions are used to move data to and from stack (in particular from stack segment).

  PUSH AX

  PUSH BX

  POP AX

  POP BX

- (Alternative for SWAP Operation: ***XCHG AX, BX***)
- **CALL** also uses the stack to hold the return address for procedure.

4. Addressing Input and Output Port

- IN and OUT instructions are used to address I/O ports
- Could be *direct addressing*

  IN AL, 05h ; Here 05h is a input port number

- or *indirect addressing*

  OUT DX, AL ; DX contains the address of I/O port

- Only DX register can be used to point a I/O port

5. Implied Addressing

- No explicit address is given with the instruction
- implied within the instruction itself
- Examples:

CLC ; clear carry flag

HLT ; halts the program

RET ; return to DOS is equivalent of

MOV AH, 4Ch

INT 21h

Memory Banks

- **ODD Bank and EVEN BANK**

- If a 16-bit data is stored in a memory location started with even address and ends at odd address, then the 8086 processor can read the data in one cycle.

- If a 16-bit data is stored in a memory location started with a odd address and ends at even address, then the 8086 processor can read the data in two cycles.

- If 8-bit data is stored in either even or odd memory location then the 8086 processor can read it in one cycle.

Thank You !!