Lecture Title

CHILITERNATIONAL DIFFERENCE OF THE PROPERTY OF

Course Code: Course Title:

Dept. of Computer Science Faculty of Science and Technology

Lecturer No:	1(c)	Week No:	3	Semester:	
Lecturer:	Dr. Md. Sohidul Islam sohidul@aiub.edu				

Lecture Outline



- 1. Learn Syntax
- 2. Variable declarations
- 3. Introduction of basic data movement
- 4. Program organization: Code, Data and stack



Instructions: XCHG

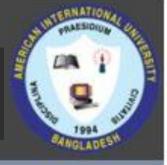
- ➤ MOV is used to **exchange** the contents between two registers or register and memory-location.
- > Syntax: XCHG destination, source XCHG AH, BL

[reads exchange value of AH with BL]



Instructions: ADD

- ➤ ADD is used to add content of two registers, register and memory-location or add a number to register or memory location.
- Syntax: ADD destination, sourceADD WORD1,AX [reads Add AX to WORD1]



Instructions: SUB

- > SUB is used to subtract content of two registers, register and memory-location or subtract a number from register or memory location.
- ➤ Syntax: SUB destination, source

 SUB AX,DX [reads Subtract DX from AX]



Instructions: INC

- ➤ **INC** is used to **add 1** to the contents of a register or memory-location.
- > Syntax: INC destination

INC WORD1 [reads Add 1 to WORD1]



Instructions: DEC

- > **DEC** is used to **subtract 1** from the contents of a register or memory-location.
- > Syntax: DEC destination

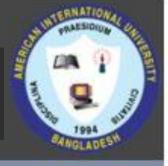
DEC WORD1 [reads subtract 1 from WORD1]



Instructions: NEG

- ➤ **NEG** is used to **negate** the contents of the destination NEG does this by replacing the contents by its two's complement.
- > Syntax: NEG destination

NEG BX [reads negate the contents of BX]



Agreement of Operator

• The operand of the preceding two-operand instruction MUST be same type. (i.e. both bytes or words). Thus,

• MOVAX,BYTE1 ; its illegal

• MOV AH,'A'; legal

• MOV AX,'A'; legal if source is a word

PARESIDIO DE LA CONTRACTION DE

Program Structure

- ➤ A program Consist of
 - Stack
 - Data
 - Code
- Each part occupies memory segments
- Program segment is translated into memory segment by assembler.
- The size of code and data of a program can be specified by **memory model** using **.MODEL** directive
- .MODEL Memory_model
- .MODEL SMALL [Code in ONE segment and Data in one segment]



Stack Segment

- Allocate a block of memory (stack area) to store the stack.
- The stack area should be big enough to contain the stack at its maximum size.
- Declaration:

.STACK size

.STACK 100H

- ** Allocates 100 bytes for stack area reasonable size for most applications
- ** If size is omitted 1KB is allocated for stack area.



Data Segment

- Contains all the **variable** definitions and sometimes Constant definitions (constant does not take any memory).
- To declare data segment **.DATA** directive is used followed by variable and constant declaration.

.DATA

WORD1 DW 2

BYTE1 DB 1

MSG DB 'THIS IS A MESSAGE'

MASK EQU 10010001B



Code Segment

- > Contains the program's instructions
- > Declaration:
 - >.CODE name [name is optional]

There is no need of **name** in SMALL program

➤ Inside a code segment, instructions are organized as procedures.

name PROC

; body of the procedure

name ENDP

➤ Here name = name of the procedure. PROC and ENDP are pseudo-ops



Program Structure

.MODEL SMALL

.STACK 100H

.DATA

; data definitions here

CODE

MAIN PROC

;instructions go here

MAIN ENDP

;other procedures go here

END MAIN

*** The last line of the program should be the END directive, followed by the name of main procedure



Instruction: LEA

➤ LEA: Load Effective address

LEA destination, source

➤ LEA puts copy of the source offset address into the destination.

i.e. LEA DX, MSG ; will load address of MSG to DX



Program Segment Prefix (PSP)

- PSP contains information about the program to facilitate the **program access** in this area
- DOS places its segment number in both DS and ES before program execution
- Usually, DS does not contain the segment number of the data segment.
- Thus, a program with data segment will start with these two instruction

MOV AX,@DATA [name of data segment define in .DATA] MOV DS,AX

References



- Assembly Language Programming and Organization of the IBM PC, Ytha Yu and Charles Marut, McGraw Hill, 1992. (ISBN: 0-07-072692-2).
- https://www.tutorialspoint.com/assembly_programming/index.htm

Books



- Assembly Language Programming and Organization of the IBM PC, Ytha Yu and Charles Marut, McGraw Hill, 1992. (ISBN: 0-07-072692-2).
- Essentials of Computer Organization and Architecture, (Third Edition), Linda Null and Julia Lobur
- W. Stallings, "Computer Organization and Architecture: Designing for performance", 67h Edition, Prentice Hall of India, 2003, ISBN 81 – 203 – 2962 – 7
- Computer Organization and Architecture by John P. Haynes.