CSE 331L / EEE 332L: Microprocessor Interfacing & Embedded System (Lab 2)

Section: 7 & 8, Fall 2019

Today's topics:

1. Instructions: Sub, inc, dec
2. Program Structure
3. Single and multi-key input/output

Instruction	**Operands**	**Descriptions**
SUB	REG, memory memory, REG REG, REG memory, immediate REG, immediate	Algorithm: operand1 = operand1 - operand2

Example :

```
ORG 100H
MOV AL, 5
SUB AL, 1       ; AL = 4

RET
```

Instruction	Operands	Descriptions
INC	REG MEM	Algorithm: operand = operand + 1

Example :

```
ORG 100H
MOV AL, 4
INC AL          ; AL = 5
RET
```

Instruction	Operands	Descriptions
DEC	REG MEM	Algorithm: operand = operand - 1

Example :

```
ORG 100H
MOV AL,86
DEC AL          ; AL=85
RET
```

Program Structure

- **Code Segment:** holds the instructions of the program, instructions are organized as procedures.

  Procedure is a part of code that can be called from your program in order to make some specific task. Procedures make program more structural and easier to understand. Generally procedure returns to the same point from where it was called.

  The syntax for procedure declaration:

```
name PROC
    ; here goes the code
    ; of the procedure ...
  RET
  name ENDP
```

  name - is the procedure name, the same name should be in the top and bottom, this is used to check the correct closing of procedures.

  Probably, you already know that RET instruction is used to return to the operating system. The same instruction is used to return from procedure (actually operating system sees your program as a special procedure).

  PROC and ENDP are compiler directives, so they are not assembled into any real machine code. Compiler just remembers the address of procedure.
  CALL instruction is used to call a procedure.

  Example:

```
ORG   100h
MOV   AL, 1
CALL  m2
RET             ; return to operating system.

m2    PROC
ADD   Al, BL
RET             ; return to caller.
m2    ENDP
END
```

- **Data Segment:** contains all the variable definitions
- **Stack Segment:** contains a block of memory to store the stack, needs enough space to store.

Memory Models: SMALL, MEDIUM, COMPACT, LARGE, HUGE (Determines the size of code and data of the program)

Program Structure	**Example**

.MODEL SMALL
.STACK 100H
 	
.DATA
;variables and constants
.CODE
MAIN PROC
;instructions of main procedure
MAIN ENDP
; other procedures
END MAIN
	.MODEL SMALL
	.STACK 100H
	
	.DATA
	VAR1 DB 5EH ;variable
	VAR2 DW 8DC6h
	k EQU 5 ;constant
	
	a DB 48h, 65h, 6Ch, 6Ch, 6Fh, 00h
	b DB 'Hello', 0 ;array
	c DB 4 DUP(9)
	d DB 4 DUP(1, 2)
	
	.CODE
	MAIN PROC
	MOV BX, k
	
	MOV AL, a[3] ;array element
	MOV SI, 3
	MOV CL, a[SI]
	RET
	Main ENDP
	END MAIN

Functions

Function #	Routine
1	Single-key input
2	Single-key output
9	Character string output
4CH	DOS exit function

Single-key Input/Output

```
.MODEL SMALL
.STACK 100H

.CODE
        MAIN PROC

        MOV AH, 1     ;input-key function
        INT 21H       ;ASCII code in AL

        MOV AH, 2     ;display character function
        MOV DL, AL    ;character from input stored in AL
        INT 21H       ;display character
        RET

        MOV AH, 4CH
        INT 21H       ;exit to DOS
        MAIN ENDP
END MAIN
```

Insert newline:

```
.MODEL SMALL
.STACK 100H

.CODE
        MAIN PROC

        MOV AH, 1
        INT 21H
        MOV BL, AL

        MOV AH, 2
        MOV DL, 10      ;0AH: NEWLINE
        INT 21H
        MOV DL, 13      ;0DH: CARRIAGE RETURN,
                        ;BRINGS THE POINTER TO THE BEGINNING OF LINE
        INT 21H

        MOV AH, 2
        MOV DL, BL
        INT 21H
        RET

        MOV AH, 4CH
        INT 21H
        MAIN ENDP
END MAIN
```

Multiple key Input

```
.MODEL SMALL
.STACK 100H

.CODE
        MAIN PROC

        MOV AH, 1
        INT 21H
        MOV BL, AL

        INT 21H
        MOV BH, AL

        INT 21H
        MOV CL, AL

        MOV AH, 2
        MOV DL, 10
        INT 21H
        MOV DL, 13
        INT 21H

        MOV AH, 2
        MOV DL, BL
        INT 21H
        RET

        MOV AH, 4CH
        INT 21H
        MAIN ENDP
END MAIN
```

TASK

1. Take 5 single-key input and display them using output function; each output should be in separate line.
2. Show the output of Task 1 in reverse order all in one line using space between each two characters.
3. Take 2 numbers as input, add them and show the answer.
4. Take 2 numbers as input, subtract them and show the answer. (the answer should be positive)

Home Task

1. Create a hollow diamond star pattern using '*'
   Example: