Lab 1

Name- Tanana Kabir

ID- 1620101012

Sec-7

Course- CSE331L

Faculty- RSF

---

* Push and commit bike into github repository:

First of all; We have to open git bash and type:

$ git clone (pwd repo link) (master)

* then we have to go inside cloned bike

Femastin

→ After going inside the we have to again open git bash.

→ Then type: $git init

~~$ git add~~

$ git config --global user.name "xx"

$ git remote -v.

→ Then after doing this we have to enter inside the lab assignment file and create a new file of our work.

Then we have to put: $git add.

$ git commit -m "F commit"

$ git push -u origin master

$ git push -u main master

(After each of instruction we have to press the key enter)

Lab 2

*Syntax for a variable declaration

name DB value
name DW value

DB – Define Byte
DW – Define word

*Creating Constants:

name EQU <any expression>

For example.

K EQU 5
MOV AX, K

Creating Arrays

Array definition examples:

a DB 48h, 65h, 6Ch, 6Fh, 00h

b DB 'Hello', 0

* Accessing the value of any element in array using square brackets, for example:

MOV AL, a[3]

* Using any of the memory index registers BX, SI, DI, BP, for example:

MOV SI, 3

MOV AL, a[SI]

The syntax for DUP:

number DUP (value(s))

number - number of duplicates to make (any constant value)

value - expression that DUP will duplicate

for example:

c DB 6 DUP(9)

is an alternative way of declaring

c DB 9,9,9,9,9,9

one more example:

d DB 5 DUP(1,2)

is an alternative way of

declaring:

d DB 1,2,1,2,1,2,1,2,1,2

Memory Access:

We can use base registers BX, SI, DI, BP. Combining these registers inside [ ] symbols, we can get different memory locations.

[BX+SI] [SI] [BX+SI+d8]

[BX+DI] [DI] [BX+DI+d8]

[BP+SI] d16 (variable offset only) [BP+SI+d8]

[BP+DI] [BX] [BP+DI+d8]

[SI+d8] [BX + SI + d16] [SI + d16]

[DI+d8] [BX + SI + d16] [SI + d16]

[BP+d8] [BP + SI + d16] [BP + d16]

[BX + d8] [BP+DI+d16] [BX+d16]

Instructions:

Instruction	Operando	Description
INC	REG MEM	Increment. Algorithm: operand = Operand + 1 Example MOV AL, 4

		INC AL ; AL=6 RET
DEC	REG MEM	Decrement. Algorithm: operand=operand-1 Example: MOV AL, 86 DEC AL; AL=85 RET

LEA	REG, MEM	Load Effective address.
		Algorithm:
		REG = address of memory (offset)
		Example:
		MOV BX, 35h
		MOV DI, 112h
		LEA SI, [BX+DI]

Declaring Array:

Array Name db Size DUP(?)

Value initialize:

arr1 db 50 dup(5,10,12)

Index Values:

mov bx, offset arr0

mov [bx], 5 ; inc bx

mov [bx+1], 10

mov [bx+9], 9

OFFSET:

1. mov si, offset variable

2. mov si, variable

As a matter of style, when I wrote X86 assembler I would write it this way -

1. mov si, offset variable
2. mov si, [variable]

```
org 100h ; #include<stdio.h>
; a DB 10 ; int a=10
; b DW 15 ; int b=15

; MOV AX, K
; ADD AX K1
```

E

;K EQU 10

;K1 EQU 15

; a DB 10; variable

; a DB 10h,15h,10h,11h,12h;

array; int a[10]

; b DB 10 DUP (?); int a[10]={}

; int a[h];

; c DB 5 DUP (1,2); 1,2,1,2,1

; MOV BX,10

; MOV BX,5

;INC BX;c++

;DEC BX; c--

emastin

```
;MOV BX, 05h
; MOV DI, 12h
; LEA SI, [BX+DI]

a db 0h, 2h, 0h, 4h, 0h, 6h
ret ; return 0
```

Lab 3

A single program in assembly language is divided into four segments which are -

1. Data Segment.
2. Code Segment.
3. Stack Segment, and
4. Extra Segment.

Print: Hello World in Assembly language.

~~DATA Segm~~

```
DATA SEGMENT
    MESSAGE DB "HELLO WORLD!!!"
ENDS
CODE SEGMENT
    ASSUME DS:DATA CS:CODE
START:
    MOV AX, DATA
    MOV DS, AX
    LEA DX, MESSAGE
    MOV AH, 9
    INT 21H
    MOV AH, 4CH
    INT 21H
ENDS
END START
```

First line- DATA SEGMENT
Next line- MESSAGE DB "HELLO WORLD!!!"
Next line - DATA ENDS
Next line- CODE SEGMENT
Next line- ASSUME DS: DATA CS: CODE
Next line - START:
Next line- MOV AX, DATA
MOV DS, AX
Next line- LEA DX MESSAGE
MOV AH, 9
INT 21H
Next line- MOV AH, 4CH
INT 21H

Next time - END START

Execution of program explanation

— Hello World

Now —

~~DATA SEGMENT~~

~~MESSAGE DB "HI~~

DATA SEGMENT

MESSAGE DB "HELLO WORLD$"

START:

MOV AX, DATA

MOV DS, AX

LEA DX, MESSAGE

MOV AH, 9

INT 21H

MOV AH, 4CH

INT 21H

END START

Assembly Example 1 - Print 2 strings

```
.Model SMALL
.STACK 100H
.DATA
    STRING_1 DB 'I hate CSE$'
    STRING_2 DB 'But I Love Koothi!!!$'
.CODE
 MAIN PROC
 MOV AX, @DATA  ; initialize DS
 MOV DS, AX
 LEA DX, STRING_1 ; load & display
                    the STRING_1
~~MOV DS, AX~~
```

```
MOV AH,9
INT 21H

MOV AH,2    ; carriage return
MOV DL,0DH
INT 21H

MOV DL,0AH    ; line feed
INT 21H

LEA DX,STRING-2 ; load & display
                  the STRING-2
MOV AH,9
INT 21H

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

Pemastin

Assembly example 2- Read a String and Print it

```
.MODE SMALL
 .STACK 100H
.DATA
 MSG_1 EQU 'Enter the character:$'
 MSG_2 EQU 0DH,0AH,'The given
                character is:$'
 PROMPT_1 DB MSG_1
 PROMPT_2 DB MSG_2
.CODE
 MAIN PROC
    MOV AX,@DATA ; initialize DS

MOV DS AX
```

```
LEA DX, PROMPT_1 ;load and
                  display PROMPT_1

MOV AH,9
INT 21H

MOV AH,1  ;read a character
INT 21H

MOV BL,AL ;save the given
           character into BL

LEA DX, PROMPT_2 ; loood and
                   display PROMPT_2

MOV AH,9
INT 21H
```

```
MOV AH,2  ; display the character
MOV DL,BL
INT 21H

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

Assembly Example-3 Read a string from user and display this string in a new line.

```
.MODEL SMALL
.STACK 100H
```

```
.CODE
MAIN PROC
    MOV AH,1        ;Read a character
    INT 21H

    MOV BL,AL       ;save input
                    character into BL
    MOV AH,2        ;carriage return
    MOV DL,0DH
    INT 21H

    MOV DL,0AH      ;line feed
    INT 21H

    MOV AH,2        ;display the
    MOV DL,BL       character stored
    INT 21H         in BL
```

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

Assembly Example 4 - Read a string with gaps and print it

```
.MODEL SMALL
.STACK 64
.DATA
        STRING2 DB ?
        SYM DB '$'
        INPUT_MSG DB 0ah,0dh,0dh,0ah
        0DH 'Enter the Input: 0DH,0AH$'
```

Execution

```
OUTPUT_MS DB 0ah,0dh,0AH,
  0DH,'The output is',0AH,0dh,'$'
.CODE
MAIN PROC
    MOV AX,@DATA
    MOV DS,AX
    MOV DX,OFFSET INPUT_MS
    ; lea dx, input_m
    MOV AH,09
    INT 21H
    LEA SI,STRING
INPUT: MOV AH,01
    INT 21H
    MOV [SI],AL
```

```
INC SI
CMP AL,0DH
JNZ INPUT
; MOV AL,SYM
MOV [SI],'$'

OUTPUT: LEA DX,OUTPUT_M ;load
                and display PROMPT_2

 MOV AH,9
 INT 21H
 MOV DL,0AH
 MOV AH,02H
INT 21H
MOV DX,OFFSET STRING2
```

Femastin

```
MOV AH, 09H
INT 21H
MOV AH, 4CH
INT 21H
MAIN ENDP
END MAIN
```

Assembly Example 5- Printing string using MOV instruction

```
.MODEL
.Model small
.STACK
.DATA
MSG1 DB 'KI!!! Kamon lage
:D$'
```

@CODE
MOV AX, @DATA
MOV DS, AX

MOV dx, OFFSET MSG1 ; LEA dx, msg1
mov ah, 09h
int 21h

mov ah, 4ch
int 21h

END

Assembly Example - Print Digit from 0-9

.MODEL SMALL
.STACK 100H

```
.DATA
    PROMPT DB "the counting from 0 to 9 is: $"
.CODE
MAIN PROC
    MOV AX,@DATA    ;initialize DS
    MOV DS,AX
    LEA DX, PROMPT  ; load and print PROMPT
    MOV AH,9
    INT 21H

    MOV ECX,10      ; initialize cx
    MOV AH,2        ; set output function
    MOV DL,48       ; set DL with 0
```

```
@LOOP:          ; loop label
  INT 21H       ; print character

  INC DL        ; increment DL 1 ones
                  ASCII character
  DEC CX        ; decrement CX
  JNZ @LOOP     ; jump to label
                  @LOOP if CX is 0

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

Assembly Example 7- Sum of two integers

```
.MODEL SMALL
.STACK 100H

.DATA
  PROMPT_1 DB \'Enter the
              first digit :$\'
  PROMPT_2 DB \'Enter the second
              digit :$\'
  PROMPT_3 DB \'Sum of first
              and Second digit $\'

VALUE_1 DB?
VALUE_2 DB?
```

```
CODE
MAIN PROC
    MOV AX, @DATA  ; initialize DS
    MOV DS, AX
    LEA DX, PROMPT1  ; load and
                       display the PROMPT1
    MOV AH, 9
    INT 21H
    MOV AH, 1  ; read a character
    INT 21H
    SUB AL, 30H  ; save input digit
                   in VALUE in ASCII
    MOV VALUE_1, AL  code
```

Execution

```
MOV AH,2    ;carriage return
MOV DL,0DH
INT 21H

MOV DL,0AH  ; line feed
INT 21H
.
LEA DX,PROMPT_2 ; load and
                  display the PROMPT_2
MOV AH,9
INT 21H

MOV AH,1 ; read a character
INT 21H
↓
SUB AL,30H ;save second
             digit in VALUE_2 in ASCII
             code
```

```
MOV VALUE_2, AL

MOV AH, 2 ; carriage return
MOV DL, 0DH
INT 21H

MOV DL, 0AH ; line feed
INT 21H

LEA DX, PROMPT_3 ; load and
MOV AH, 9 display the PROMPT_3
INT 21H

MOV AL, VALUE_1 ; add first and
                  second digit
ADD AL, VALUE_2
```

Femastin

```
ADD AL,30H    ;convert ASCII do
              DECIMAL code
MOV AH,2      ;display the character
MOV DL,AL
INT 21H

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