**PRACTICAL # 04**

**OBJECT:**

Variables in assembly. Arithmetic multiplication and division using variables.

**THEORY:**

The variables are an important component for any nontrivial program. They hold data and information for the program. Variables (global) in the 8086 assembly reside in the **Data segment** of the program.

There are two types of variables in assembly; they are initialized and uninitialized variables.

Initialized variables are defined as

**age db 25**

An uninitialized variable is has initial value part as '?'

**displacement dw ?**

where

db = define byte (1 byte)

dw = define word (2 bytes)

similarly

dd = define double (2 words) (4 bytes)

dq = define quad (4 words) (8 bytes)

The syntax for defining variables in assembly is

***var\_name define\_bytes value***

where var\_name is the name of the variable, define\_bytes(db) is the datatype or number of bytes required for the variable, value is the value assigned to the variable.

**Using variables**

To use variables defined in the data segment, we first need to initialize the **DS** (data segment) register to point to the correct segment address. The following two instructions do the job,

MOV AX, @DATA

MOV DS, AX

The @DATA constant contains the data segment address for this program. The reason we did not move this constant value directly to DS register is that DS register can get a value from a register but not from a constant. So the instruction MOV DS, @DATA is illegal.

**Multiplication operation**

**Mul b**

The above instruction multiplies an 8 bit operand **b** with the value stored in register **AL** and stores the result in register **AX.**

In case of 16 bit operands, the first operand is stored in AX register and the result is stored in two registers, ie DX:AX. This is because the result of 16 bit operands may result in number requiring more than 16 bits.

The operand **b** can be a **register** or **memory.**

**mul ;Unsigned multiply**

**Syntax:**

mul op8

mul op16

op8: 8-bit register or memory

op16: 16-bit register or memory

**Output:**

If operand is op8, unsigned AX = AL \* op8

If operand is op16, unsigned DX::AX = AX \* op16

**Flags Affected:**

OF, SF=?, ZF=?, AF=?, PF=?, CF

**Division operation**

**div b**

The div instruction divides a number in register AX register with divisor **b** and stores remainder in the register **AH** and quotient in register **AL.**

**div ;Unsigned divide**

**Syntax:**

div op8

div op16

op8: 8-bit register or memory

op16: 16-bit register or memory

**Output:**

If operand is op8, unsigned AL = AX / op8 and AH = AX % op8

If operand is op16, unsigned AX = DX::AX / op16 and DX = DX::AX % op16

**Flags Affected:**

OF=?, SF=?, ZF=?, AF=?, PF=?, CF=?

**Notes**: The division operation for integers results in two integer values, not a floating point number. It performs both division and modulus operations in one instruction.

**Program:**

The program first defines variables for the multiplication and division operations. It also declares uninitialized variables for storing remainder and quotient for division operation. Finally outputs the result.

.model small

.stack 100h

.data

num1 db 2

num2 db 4

dividend dw 17

divisor db 5

remainder db ?

Quotient db ?

.code

main proc

;initialize the data segment register to use variables

mov ax, @data

mov ds, ax

; multiply

mov al, num1

mov bl, num2

mul bl

;output result (product)

mov dl, al ; result in AX register

mov ah, 2

add dl, 48 ; get ASCII value of the resultant digit

int 21h

mov AX, dividend

mov cl, divisor

div cl

; AH has remainder, AL has quotient

mov remainder, AH

mov Quotient, AL

mov ah, 2

mov dl, remainder

add dl, 48

int 21h

mov dl, Quotient

add dl, 48

int 21h

mov ah, 4ch ; return control to DOS/OS

int 21h

main endp

end main

In the above program, there are two new instructions mov ax, @data and mov ds, ax. The DS(Data Segment) register is not initialized when the program loads. These instructions are used to initialize the DS register to the location of Data Segment. The first statement loads the contents of the constant @data into AX register. These contents are the address of the Data Segment. The second statement now initializes the DS register with the address of Data Segment in AX register. The DS cannot directly get @data value, so the constant is first loaded into AX register and then AX register is loaded to DS register.

**ACTIVITIES**

**Activity 1**

Write the above program in text editor, assemble and link the program to generate executable file. Run the exe file and observe the output.

**Activity 2**

Write the same program in Emu8086 emulator code editor but with different operand values. Emulate and run the program to verify the outputs.

**REVIEW QUESTIONS**

1. Where are the remainder and the quotient stored in case of 16 bit number division?
2. What will happen in division operation if dividend is 0 and what will happen if divisor is 0?
3. What will be the result of using uninitialized variable in program?
4. Why does the multiplication of two 16-bit numbers require for their product two registers instead of one? Give some suitable example.