**FR. Conceicao Rodrigues College of Engineering**
**Department of Computer Engineering**

# 2. Multiplication of Two 8/16/32 bit numbers

## 1. Course, Subject & Experiment Details

| | | | |
|---|---|---|---|
| **Academic Year** | **2023-24** | **Estimated Time** | **Experiment No. 2– 02 Hours** |
| **Course & Semester** | **S.E. (Comps) – Sem. IV** | **Subject Name** | **Microprocessor** |
| **Chapter No.** | **2** | **Chapter Title** | **Instruction Set and Programming** |
| **Experiment Type** | **Software** | **Subject Code** | **CSC405** |

**Rubrics**

| Timeline (2) | Practical Skill & Applied Knowledge (2) | Output (3) | Postlab (3) | Total (10) | Sign |
|---|---|---|---|---|---|
| | | | | | |

## 2. Aim & Objective of Experiment

**TO MULTIPLY TWO 8/16/32 BIT NUMBERS**

**Objective :**Program involves storing the two 8/16/32 bit numbers in memory locations and multiplying them the objective of this program is to give an overview of arithmetic instructions of 8086 for 32 bit operation

## 3. Software Required

TASM Assembler

**Prepared by: Prof. Heenakausar Pendhari**

## 4 . Brief Theoretical Description

**Pre-Requisites:**
1. Instructions of microprocessor 8086
2. Addressing mode of microprocessor 8086.
3. Flag register of microprocessor 8086
4. Knowledge of TASM directories.

**Theory:** MUL instruction This instruction multiplies byte/word present in source with AL/AX.

a) When two 8 bit numbers are multiplied a 16 bit product is made available in AX register where AH stores higher byte and AL stores lower byte of the product.

e.g. MUL BL

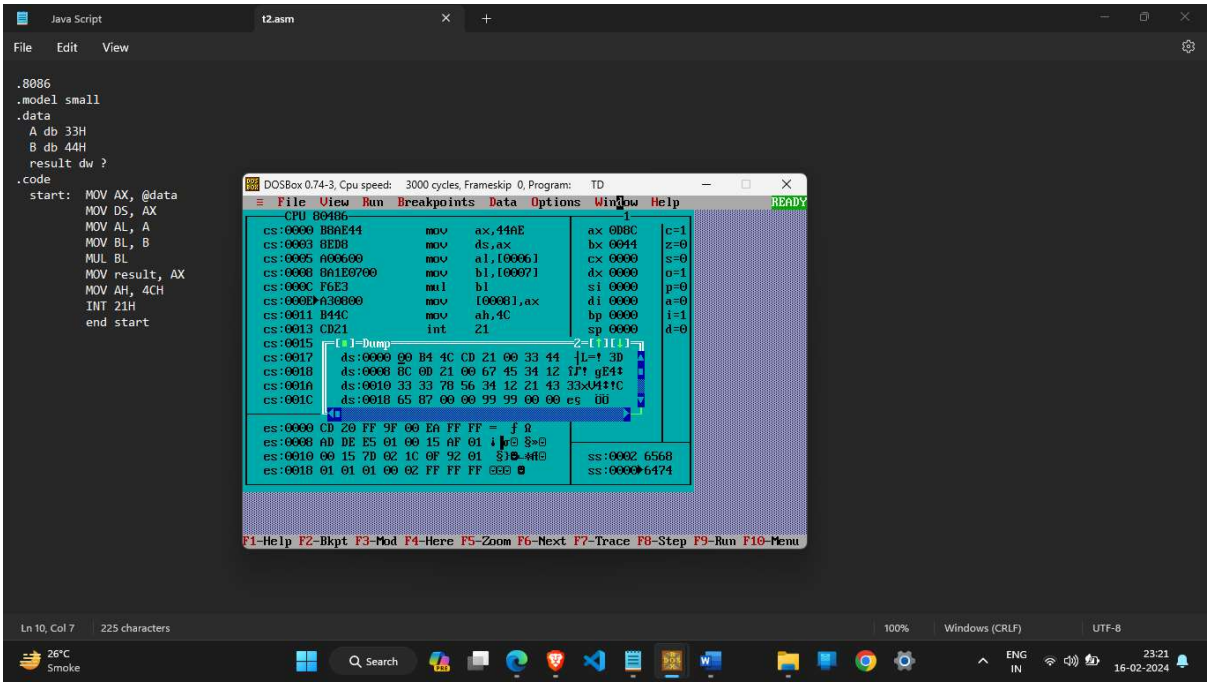
b) When two 16 bit numbers are multiplied a 32 bit product is made available in DX and AX register pair. DX has higher word and AX has lower word of the product.

e.g. MUL BX

## 5. Algorithm:
1. Initialize the data segment.
2. Store two 8/16/32 bit numbers in memory locations.
3. Move the 1st number in any TWO general purpose register.
4. Move the 2nd number in any other TWO general purpose register
5. Multiply the 2 numbers.
6. Store the result in memory location.
7. Stop.

## 6. Conclusion:
1. 8 bit multiplication

**Prepared by: Prof. Heenakausar Pendhari**




2. <u>16 bit multiplication</u>

**Prepared by: Prof. Heenakausar Pendhari**

```
.8086
.model small
.data
 A db 1002H
 B db 2215H
 result dw ?
.code
 start:
  MOV AX, @data
  MOV DS, AX
  MOV AX, A
  MOV BX, B
  MUL BX
  MOV result, AX; Lower bits
  MOV result+2, DX; Higher bits
  MOV AH, 4CH
  INT 21H
 end start
```


## 7. Postlab:

1. Write a program to Multiply two 32 bit number.

```
  num2l dw 4455H
  resultll dw 0000H
  resultlh dw 0000H
  resulthl dw 0000H
  resulthh dw 0000H
.code
```

**Prepared by: Prof. Heenakausar Pendhari**

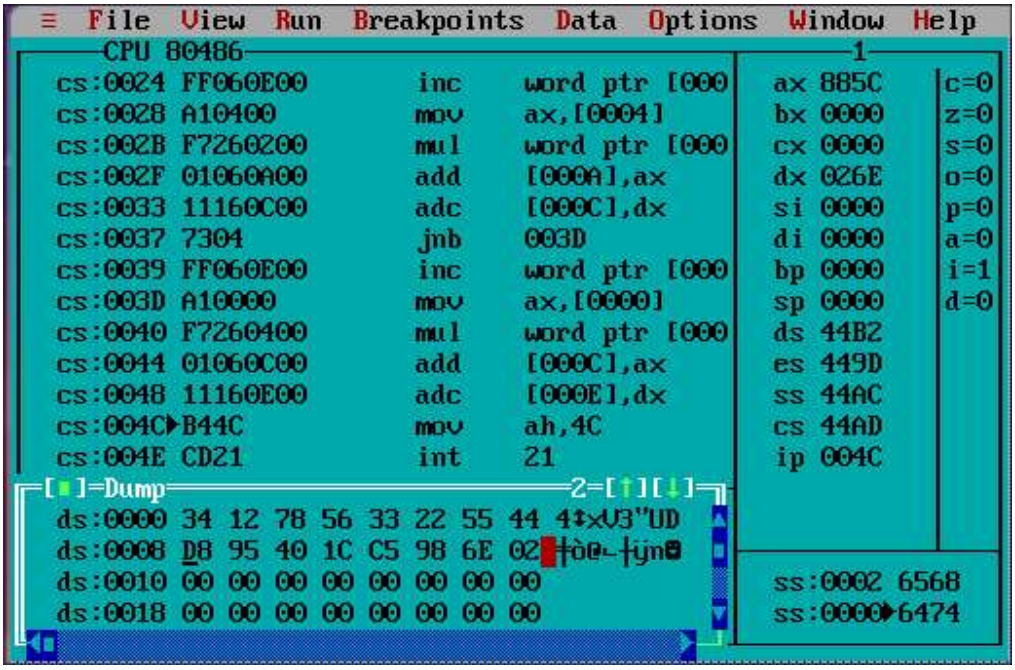
```
  start:
    MOV AX, @data
    MOV DS, AX

    MOV AX, num1l
    MUL num2l
    MOV resultll, AX; 16bit llsb.1
    MOV resultlh, DX; 16bit lhsb.1

    MOV AX, num1h;
    MUL num2l;
    ADD resultlh, AX; 16bit lhsb + lsb.2 can gen carry
    ADC resulthl, DX; 16bit hlsb, if there is a carry for
hl add it alongside. 1
    JNC skip1; ignore the carry addition if there is not
carry generated
    INC resulthh; if there was a carry generated in the
above addition the add it in the highest 16bit
    skip1:

    MOV AX, num2h;
    MUL num1l;
    ADD resultlh, AX;  16bit lhsb + lsb.3 can gen carry
    ADC resulthl, DX; 16bit hlsb + hsb, if there is a carry
for hl add it alongside. 2 can gen carry
    JNC skip2; ignore the carry addition if there is not
carry generated
    INC resulthh; if there was a carry generated in the
above addition the add it in the highest 16bit
    skip2:
    MOV AX, num1h;
    MUL num2h;
    ADD resulthl, AX; 16bit hlsb.3 can gen carry
    ADC resulthh, DX; move the 16 bit hsb in BX. 1

    MOV AH, 4CH
    INT 21H
  end start
```




2. What are the different types of addressing modes of 8086
   ⇒The addressing modes in 8086 are:
   - Immediate Addressing: The operand is specified directly in the instruction. For example: MOV AX, 1234H.Here 1234H is the hex number directly specified as the operand
   - Register Addressing: The operand is located in a register. For example: MOV AX, BX.
   - Direct Addressing: The operand is located at a specific memory address. For example: MOV AX, [1234H].Here [1234H] is an address directly specified in the opera

   - nd
   - Indirect Addressing: The address of the operand is stored in a register or memory location. For example: MOV AX, [BX]. Here, BX contains the address of the operand.
   - Indexed Addressing: The address of the operand is calculated by adding an index value to a base address. For example: MOV AX, [SI+10] Adds 10 to the value in SI to get the address.
   - Based Addressing: Similar to indexed addressing, but with a base register and optional displacement. For example: MOV AX, [BX+SI] or MOV AX, [BX+10].
   - Based Indexed Addressing: Combination of based and indexed addressing modes. For example: MOV AX, [BX+SI+10].

3. Briefly Explain The Pointers And Index Group of Registers.?

⇒ The pointers and index group of registers in the 8086 microprocessor are used to store memory addresses and offsets. They are primarily used for memory pointer operations and for accessing data structures such as arrays and strings. The pointer and index group of registers in the 8086 microprocessor include the stack pointer (SP), base pointer (BP), source index (SI), and destination index (DI) registers. These registers are 16-bit and are used for specific purposes within the CPU.

The stack pointer (SP) is used to point to the topmost item of the stack, the base pointer (BP) is used in accessing parameters passed by the stack, the source index (SI) register is used in pointer addressing of data, and the destination index (DI) register is used as a destination in some string-related operations.

**Prepared by: Prof. Heenakausar Pendhari**