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

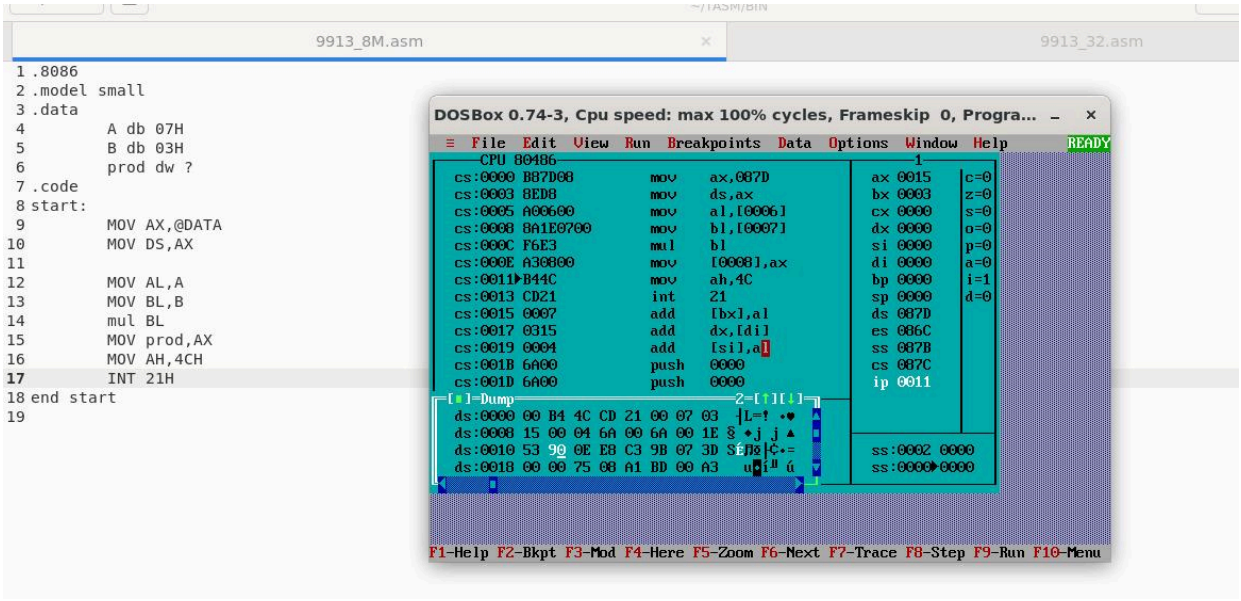
b) When two 16 bit numbers are multiplied a 32 bit product is made available in DX and AX register pair. DX has higher word and AX has lower word of the product.

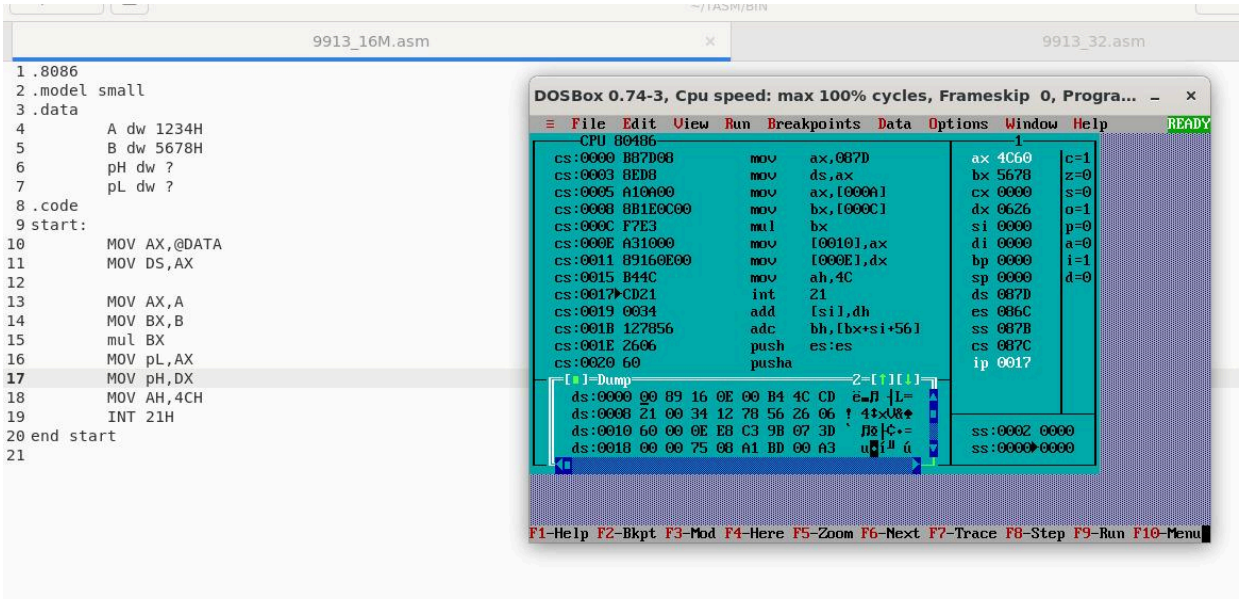
e.g. MUL BX

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:





7. Postlab:

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

Prepared by: Prof. Heenakausar Pendhari

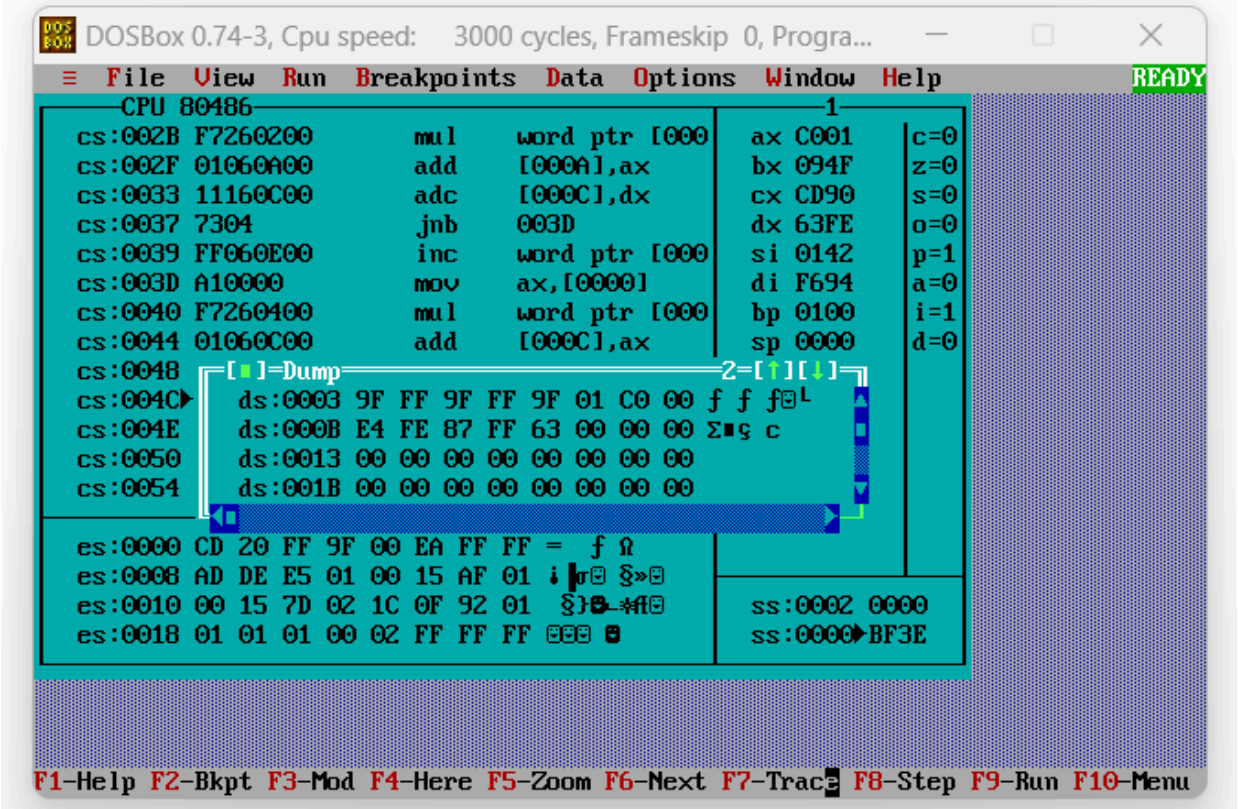
```
.8086
.model small
.data
  num1h dw 9FFFH
  num1l dw 9FFFH
  num2h dw 9FFFH
  num2l dw 9FFFH
  resultll dw 0000H
  resultlh dw 0000H
  resulthl dw 0000H
  resulthh dw 0000H
.code
  start:
    MOV AX, @data
    MOV DS, AX

    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 8086 microprocessor supports a wide range of addressing modes

i) Immediate:

- Operand is embedded directly within the instruction itself.
- Useful for small, constant values.
- Example: `MOV AX, 10` (Move the immediate value 10 into the AX register)

ii) Register:

- Operand resides in one of the 8086's internal registers (AX, BX, CX, DX, etc.).
- Fastest and most efficient for accessing register data.
- Example: `ADD AX, BX` (Add the value in BX to the value in AX)

iii) Direct:

- Operand's memory address is specified directly within the instruction.
- Requires 16 bits for addressing within the same segment.

- Example: `MOV BX, [1000h]` (Move the value from memory location 1000h to the BX register)

iv) Register Indirect:

Address of the operand is stored in one of the register (BX, SI, DI).

Useful for accessing data stored in memory locations whose addresses are dynamic or calculated during program execution.

Example: `MOV AX, [BX]` (Move the value from the memory location whose address is stored in BX to the AX register)

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

In the 8086 microprocessor, pointers and index registers are two special groups of registers used primarily for memory addressing and manipulation.

i) Pointers

Purpose: Store memory addresses that "point" to specific locations within a segment.

Key Registers:

IP (Instruction Pointer): Holds the offset address of the next instruction to be fetched by the Execution Unit (EU) from the Code Segment (CS).

SP (Stack Pointer): Points to the top of the stack within the Stack Segment (SS). The stack is used for storing temporary data, function parameters, and return addresses.

BP (Base Pointer): Often used to point to the base of a data structure or array within the Data Segment (DS).

ii)Index

Purpose: Primarily used for holding offsets in various addressing modes, making it easier to access elements within an array or data structure.

Key Registers:

SI (Source Index): Often used to hold the offset of the source operand in string operations or memory blocks.

DI (Destination Index): Frequently used to hold the offset of the destination address in string operations or memory blocks.

Prepared by: Prof. Heenakausar Pendhari