Faculty Member:_____ **Dated:** _____

Semester:_____ **Section:** _____

Department of Electrical Engineering and Computer Science

EE-222 Microprocessor Systems

Lab12: 8086 Assembly Language

Name	Reg. No.	Report Marks / 10	Viva Marks / 5	Total/15

EXPERIMENT 12

INTRODUCTION TO ASSEMBLY LANGUAGE AND TURBO ASSEMBLER (TASM)

OBJECTIVES:-

1. Getting introduced to assembly language
2. Learning some basic commands
3. Introduction to the syntax of assembly language programming
4. Learning the use of turbo assembler (TASM)

EQUIPMENT:-

SOFTWARE:

- Turbo assembler (TASM) or Microsoft assembler (MASM)

DISCUSSION:

Before starting coding in assembly we should get familiarized with some basic coding parameters, assembly language syntax and some basics of microprocessors.

Introduction to Registers:

There are four type of registers in microprocessors:

1. AX
2. BX
3. CX
4. DX

AX, BX are mainly used for arithmetic operations and saving address. CX is used for saving the values for counts which is used in executing loop instructions and DX is mainly used for I/O operations.

PROGRAM STRUCTURE:-

MEMORY MODELS:

The size of code and data a program can have is determined by specifying memory model using the .MODEL directive. The models used are SMALL, LARGE, and HUGE but the appropriate one is .SMALL. The model directive should come before any segment definition.

DATA SEGMENT:

The data segment is used for all the variables definitions. We use **.DATA** directive followed by variable and constant declaration.

STACK SEGMENT:

The purpose of stack segment is to set aside a block of memory to store the stack. The declaration syntax is:

.stack size

If we write:

.stack 100h

100 bytes would be reserved for stack. If size omitted 1KB is the default size.

CODE SEGMENT:

Code segment contains all the programming instructions. The declaration syntax is:

.code

So with minor variations this form may be used

.model small

.stack 100h

.data

; Data definitions go here

.code

Main proc

; Instructions go here

Main endp

End main

Here main proc and main endp delineate the procedure. The last line here should be END directive followed by name of main procedure.

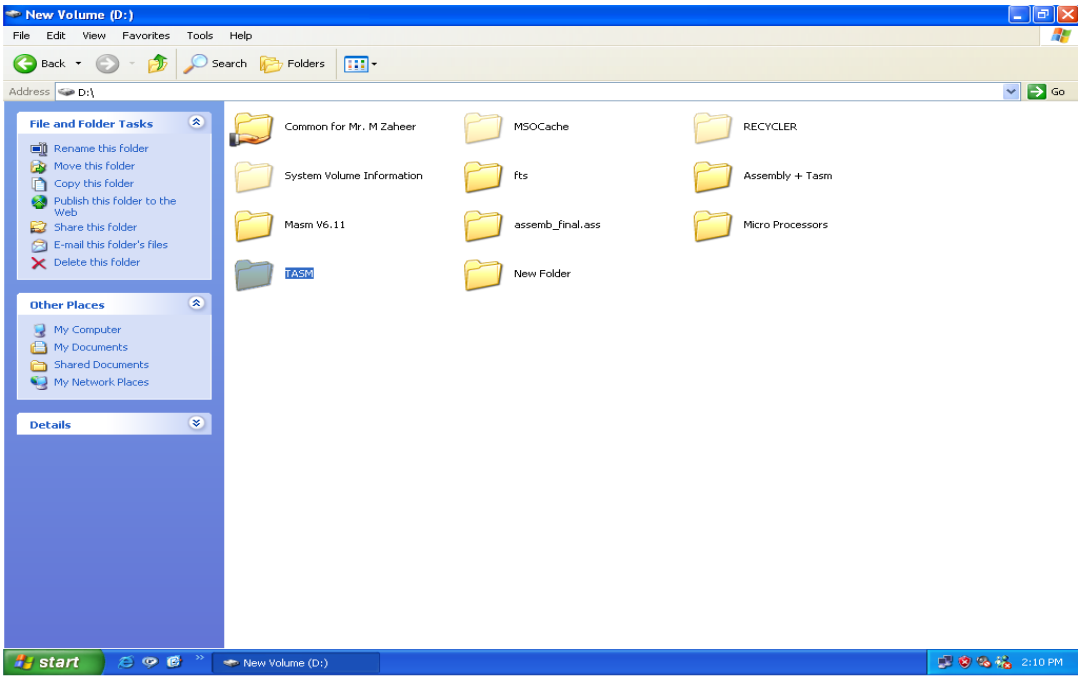
INT 21h:

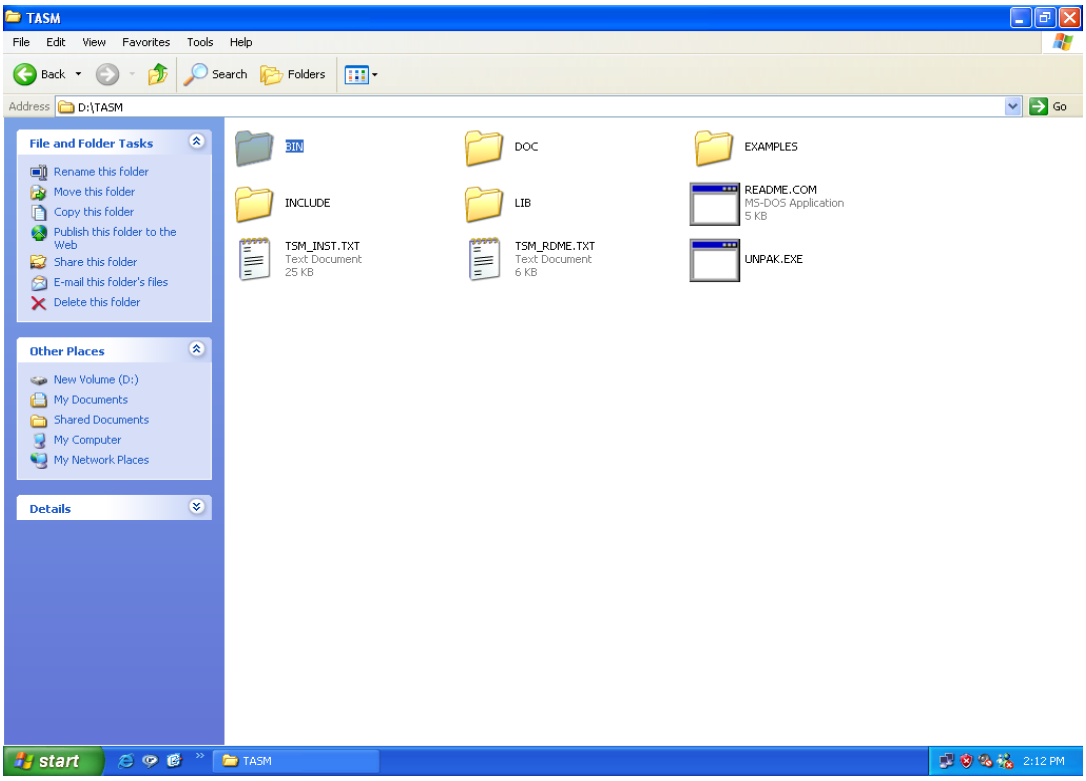
INT 21h may be used to invoke a large number of DOS functions; a particular function is requested by placing a function number in AH register and invoking INT 21h.here we are interested in following functions.

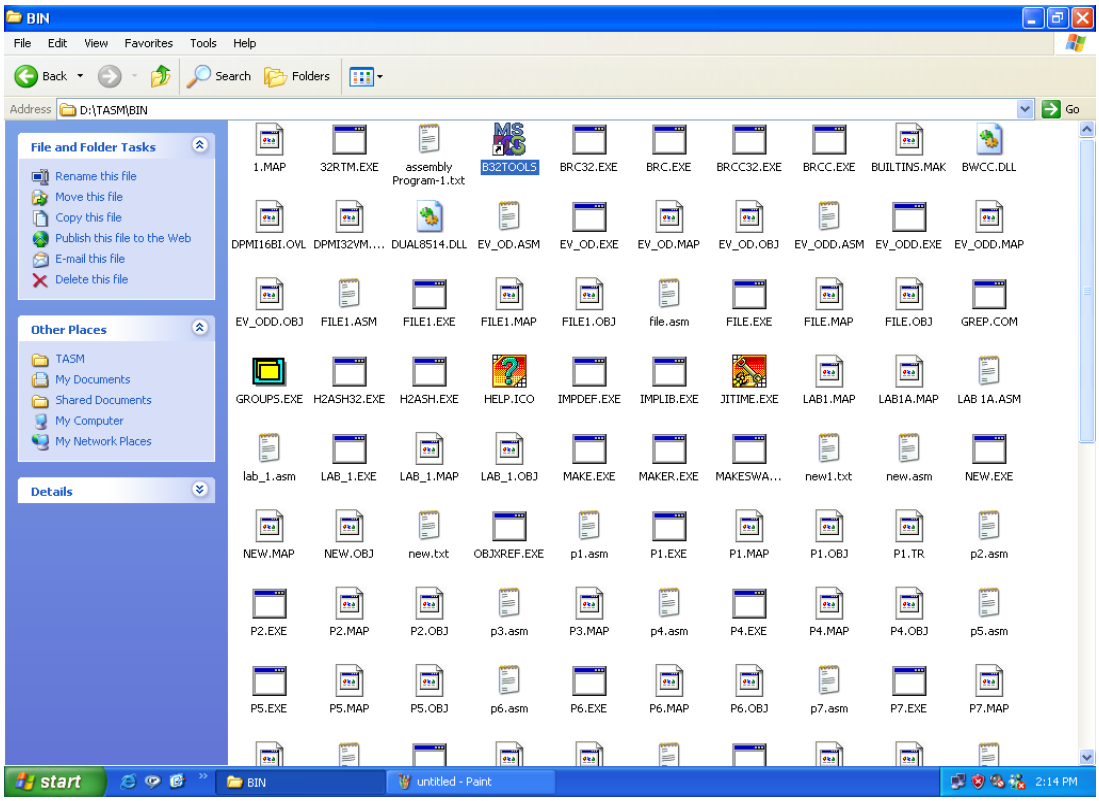
FUNCTION NUMBER	FUNCTIONS
1	Single key input
2	Single character output
9	Character string output

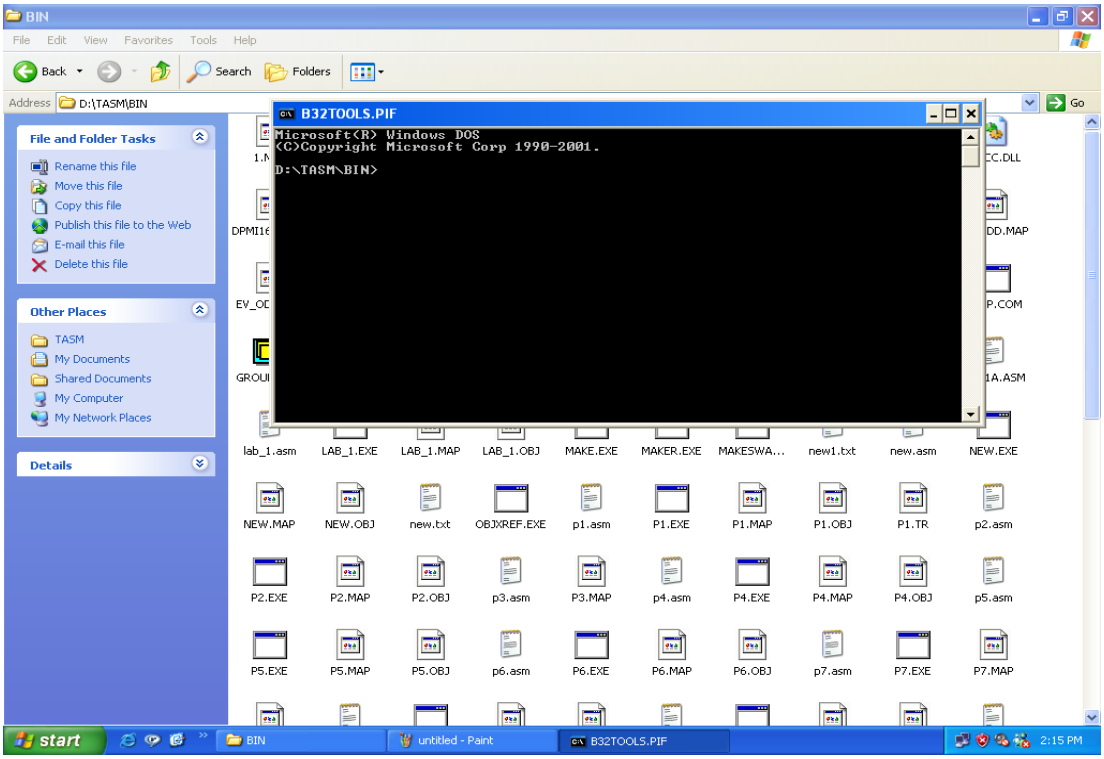
GETTING STARTED WITH TASM:-

- Open the command prompt and switch to the directory where TASM has been installed
- Go to BIN i.e. the address of the directory may look like C:\TASM\BIN
- You can open TASM\BIN\B32TOOLS
- Each step with screenshot is given to make each step easily understandable


New Volume (D:)
File Edit View Favorites Tools Help
Back Search Folders
Address D:\
Go
File and Folder Tasks
Rename this folder
Move this folder
Copy this folder
Publish this folder to the Web
Share this folder
E-mail this folder's files
Delete this folder
Other Places
My Computer
My Documents
Shared Documents
My Network Places
Details
Common for Mr. M Zaheer
MSOCache
RECYCLER
System Volume Information
fts
Assembly + Tasm
Masm V6.11
assemb_final.ass
Micro Processors
TASM
New Folder
start
New Volume (D:)
2:10 PM

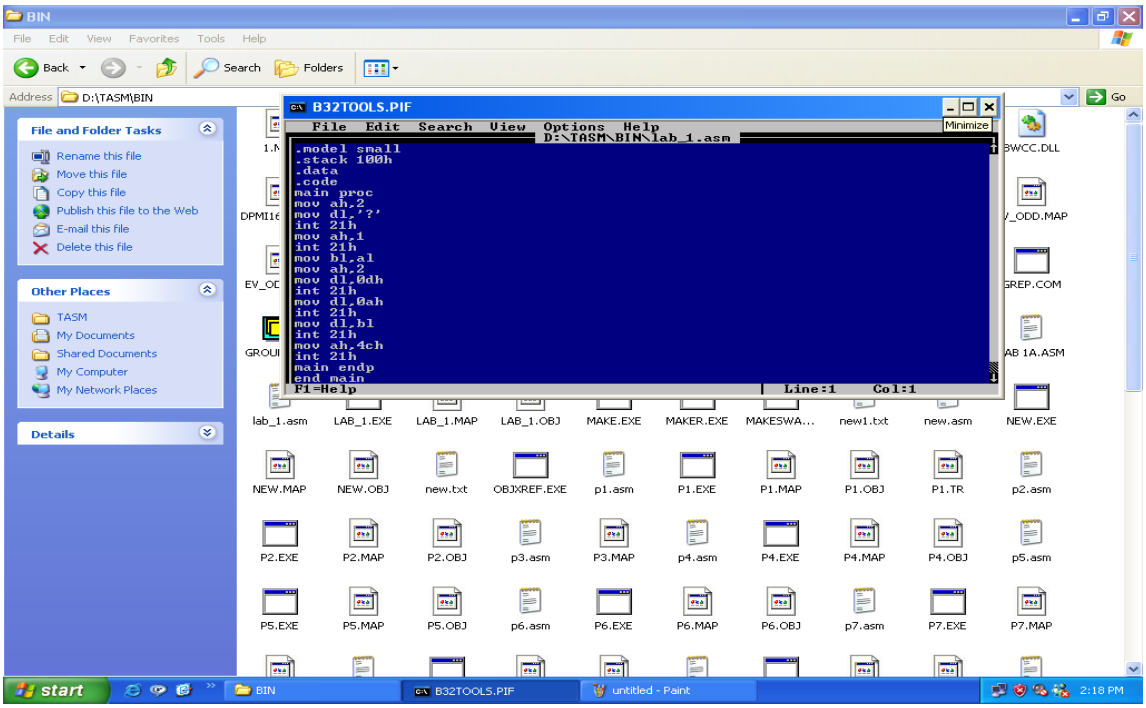

TASM
File Edit View Favorites Tools Help
Back Search Folders
Address D:\TASM
Go
File and Folder Tasks
Rename this folder
Move this folder
Copy this folder
Publish this folder to the Web
Share this folder
E-mail this folder's files
Delete this folder
Other Places
New Volume (D:)
My Documents
Shared Documents
My Computer
My Network Places
Details
BIN
DOC
EXAMPLES
INCLUDE
LIB
README.COM
MS-DOS Application
5 KB
TSM_INST.TXT
Text Document
25 KB
TSM_RDME.TXT
Text Document
6 KB
UNPAK.EXE
start
TASM
2:12 PM

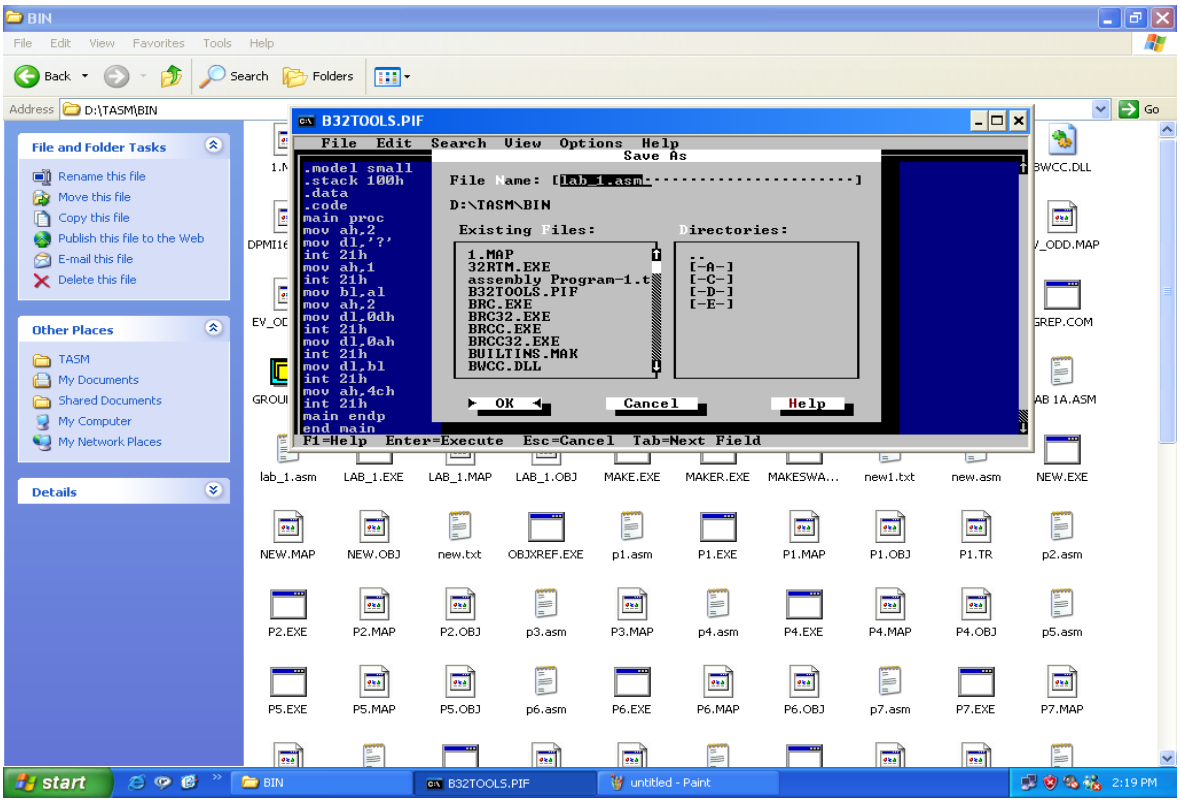




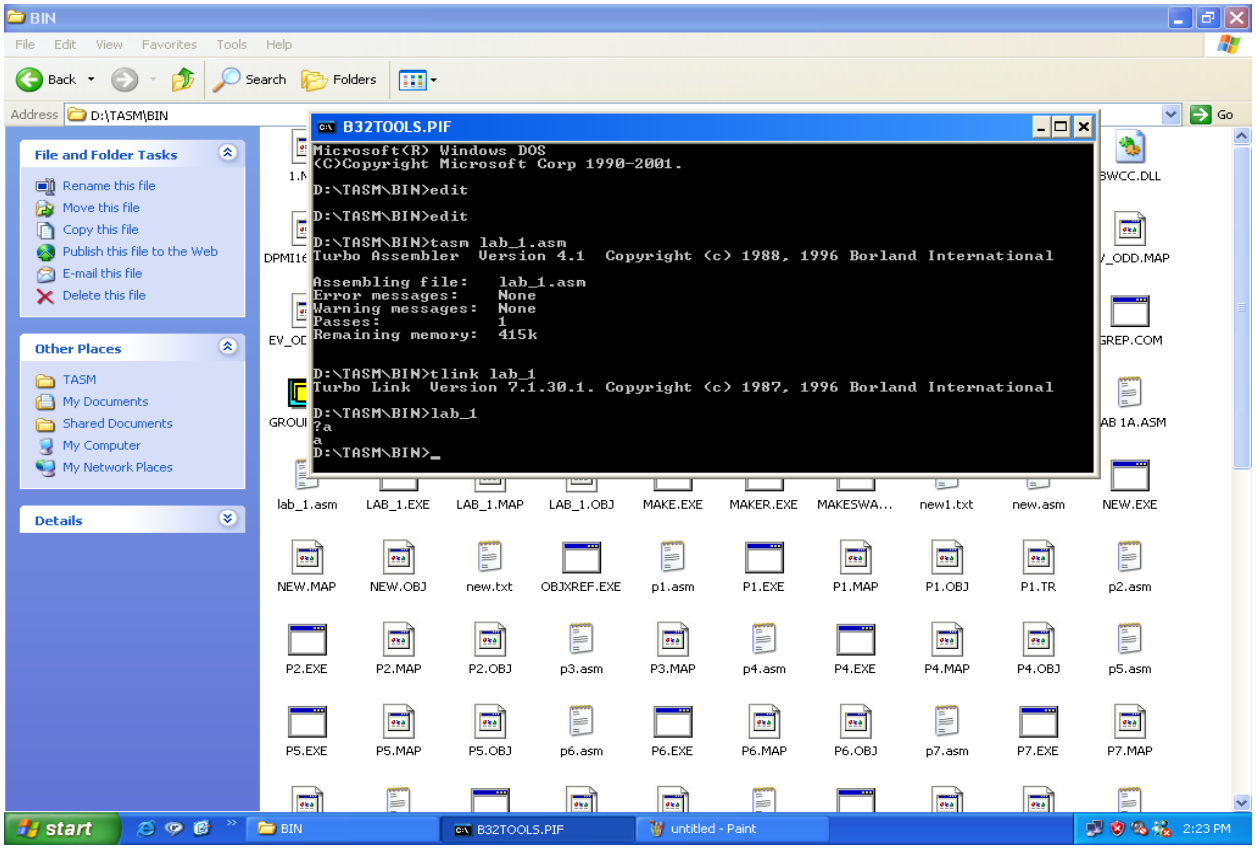
- Enter the edit command, that will open the TASM file editor

- Write your code and save your program in the bin directory of TASM, the file extension should be ".asm" as you are programming in assembly language





- Exit the editor
- Now enter “TASM ABC.ASM”
- This will compile your program and will show if there are some errors or warnings against your compiled code
- See the error message and trace it back in your code by again going to the editor
- If there is no error then enter “tlink ABC”, this will generate the “exe” file against your code, by just entering the name of the file i.e. ABC now your program will be executed
- Follow the above steps and execute the required code



SAMPLE CODE:-

PROGRAM DESCRIPTION:

Start with displaying a”**?”** and then reading a character from keyboard and displaying it on next line.

```
.model small
.stack 100h
.data
.code
```

```
main proc
mov ah,2
mov dl,'?'
int 21h
mov ah,1
int 21h
mov bl,al
mov ah,2
mov dl,0dh
int 21h
mov dl,0ah
int 21h
mov dl,bl
int 21h
mov ah,4ch
int 21h
main endp
end main
```

EXERCISE:

1. Edit the example code above and now move 3 into dl register and then run the program. Observe what is the output and why?
2. Manipulate the example code so that the user input that will now be printed on first position of first line. (Hint: the ? will be override by the user input)

Assembly Program Example 1

Write 8086 Assembly language program to add two 16-bit numbers stored in memory location 3000H – 3001H and 3002H – 3003H.

Discussion

8086 has 16-bit register. We can simply take the numbers from memory to AX and BX register, then add them using ADD instruction. When the Carry is present store carry into memory, otherwise only store AX into memory.

We are taking two numbers BCAD + FE2D = 1BADA.

Address	Data
...	...
3000	AD
3001	BC
3002	2D
3003	FE
...	...

Code

```
        MOV CX, 0000    ;Initialize Count register with 0000H
        MOV AX, [3000]  ;Load the first number into AX
        MOV BX, [3002]  ;Load second number into BX
        ADD AX, BX  ;Add AX and BX, and store to AX
        JNC STORE   ;If CY = 0, jump to STORE
        INC CX  ;Increase the Count register by 1
STORE:  MOV [3004], AX  ;Store the AX content into memory
        MOV [3006], CX  ;Store CX value into next memory location
        HLT     ;Terminate the program
```

Output

Address	Data
...	...
3004	DA
3005	BA
3006	01
...	...

Assembly Program Example 2

Write 8086 Assembly language program to multiply two 16-bit numbers stored in memory location 3000H – 3001H and 3002H – 3003H.

Discussion

8086 has 16-bit register. We can simply take the numbers from memory to AX and BX register, then add them using ADD instruction. When the Carry is present store carry into memory, otherwise only store AX into memory.

We can do multiplication in 8086 with MUL instruction. For 16-bit data the result may exceed the range, the higher order 16-bit values are stored at DX register.

We are taking two numbers BCAD * FE2D = 1BADA.

Address	Data
…	…
3000	AD
3001	BC
3002	2D
3003	FE
…	…

Code

```
MOV AX, [3000]  ;Take first 16-bit number from location 3000
MOV BX, [3002]  ;Take second 16-bit number from location 3002
MUL BX  ;Multiply AX and BX
MOV [3004], AX  ;Store Lower 16-bit at memory location 3004
MOV AX, DX  ;Take the DX to AX
MOV [3006], AX  ;Store Higher 16-bit at memory location 3006
HLT ;Terminate the program
```

Output

Address	Data
…	…
3004	69
3005	D0
3006	54
3007	BB
…	…