NAME: **GINI CHACKO**

SEMESTER: **IV**

CLASS: **SE COMPS B**

BATCH: **B**

ROLL: **8942**

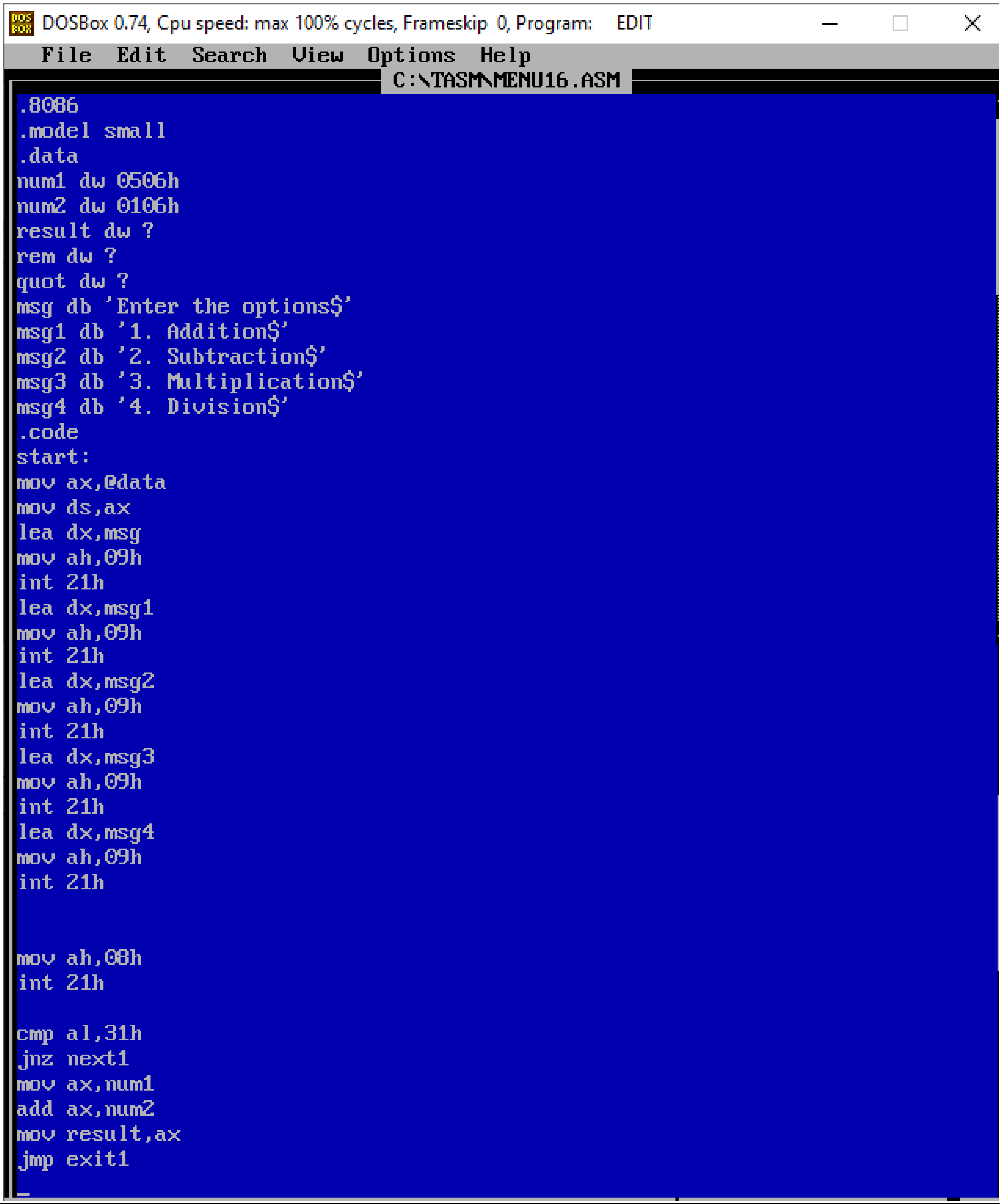
TOPIC: **MP EXPERIMENT 2 :**
Menu driven code for
16 bit ADDITION
16 bit SUBTRACTION
16 bit MULTIPLICATION
16 bit DIVISION

CODE:

```
.8086
.model small
.data
num1 dw 0506h
num2 dw 0106h
result dw ?
rem dw ?
quot dw ?
msg db 'Enter the options$'
msg1 db '1. Addition$'
msg2 db '2. Subtraction$'
msg3 db '3. Multiplication$'
msg4 db '4. Division$'
.code
start:
mov ax,@data
mov ds,ax
lea dx,msg
mov ah,09h
int 21h
lea dx,msg1
mov ah,09h
int 21h
lea dx,msg2
mov ah,09h
int 21h
lea dx,msg3
mov ah,09h
int 21h
lea dx,msg4
mov ah,09h
int 21h
```

```
mov ah,08h
int 21h

cmp al,31h
jnz next1
mov ax,num1
add ax,num2
mov result,ax
jmp exit1

next1: cmp al,32h
jnz next2
mov ax,num1
sub ax,num2
mov result,ax
jmp exit1

next2: cmp al,33h
jnz next3
mov ax,num1
mov bx,num2
mul bx
mov result,ax
mov ax,dx
jmp exit1

next3: cmp al,34h
jnz exit1
mov ax,0000h
mov bx,0000h
mov ax,num1
mov bx,num2
div bx
mov quot,ax
mov rem,dx
```

```
exit1:int 3h
int 21h
end start
```



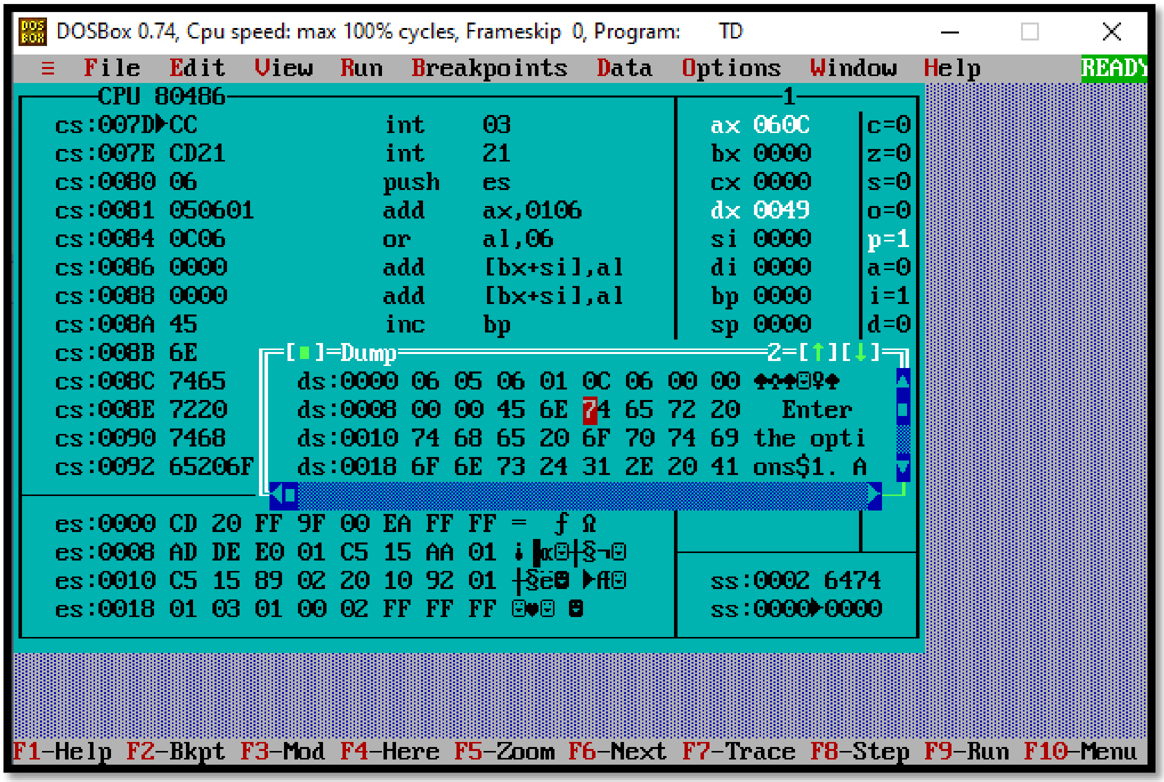

```
.8086
.model small
.data
num1 dw 0506h
num2 dw 0106h
result dw ?
rem dw ?
quot dw ?
msg db 'Enter the options$'
msg1 db '1. Addition$'
msg2 db '2. Subtraction$'
msg3 db '3. Multiplication$'
msg4 db '4. Division$'
.code
start:
mov ax,@data
mov ds,ax
lea dx,msg
mov ah,09h
int 21h
lea dx,msg1
mov ah,09h
int 21h
lea dx,msg2
mov ah,09h
int 21h
lea dx,msg3
mov ah,09h
int 21h
lea dx,msg4
mov ah,09h
int 21h


mov ah,08h
int 21h

cmp al,31h
jnz next1
mov ax,num1
add ax,num2
mov result,ax
jmp exit1
```

```
next1: cmp al,32h
jnz next2
mov ax,num1
sub ax,num2
mov result,ax
jmp exit1

next2: cmp al,33h
jnz next3
mov ax,num1
mov bx,num2
mul bx
mov result,ax
mov ax,dx
jmp exit1

next3: cmp al,34h
jnz exit1
mov ax,0000h
mov bx,0000h
mov ax,num1
mov bx,num2
div bx
mov quot,ax
mov rem,dx

exit1:int 3h
int 21h
end start
_
F1=Help                                                    Line:74     Col:1
```

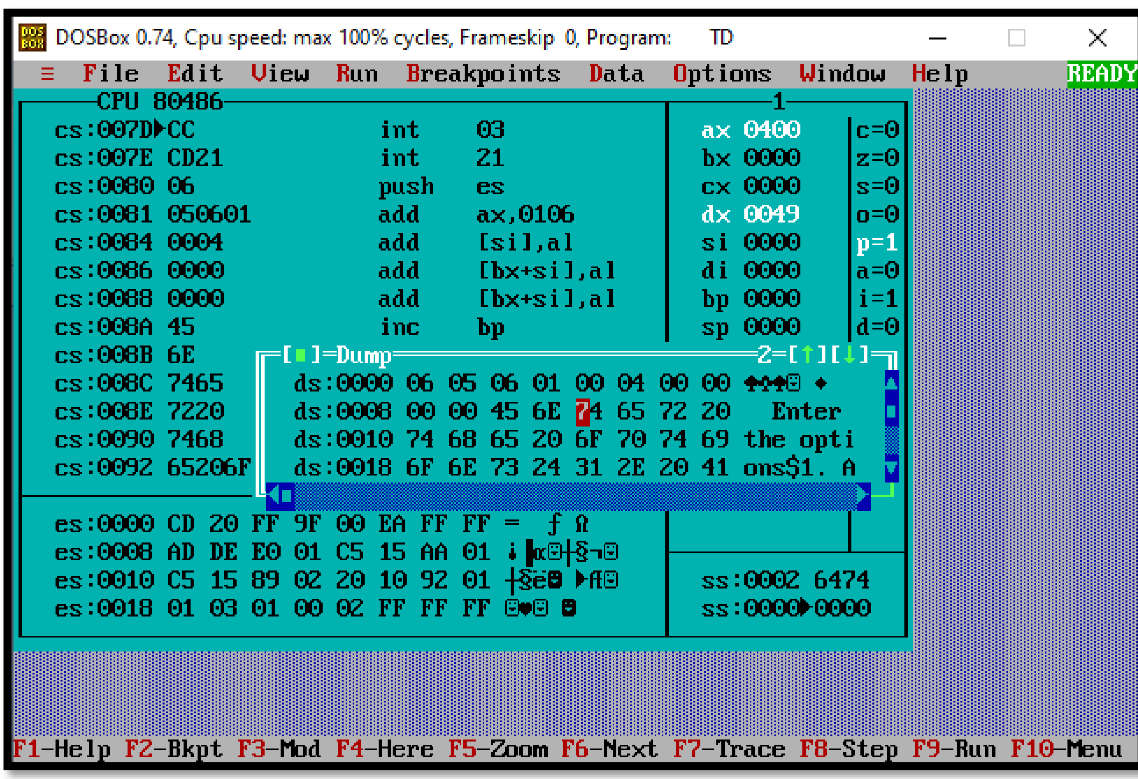
OUTPUT:

```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program:    TD

C:\TASM>edit

C:\TASM>td menu16
Turbo Debugger Version 3.1 Copyright (c) 1988,92 Borland International
Enter the options1. Addition2. Subtraction3. Multiplication4. Division
```

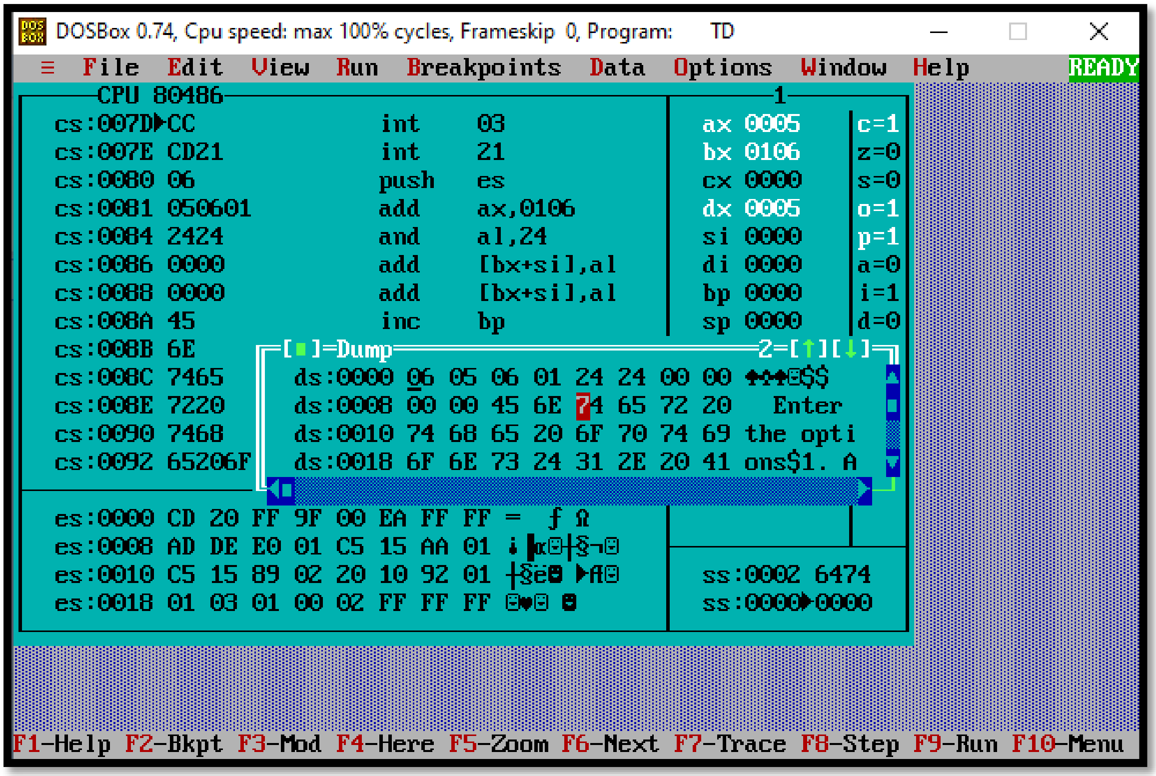
1. Addition :



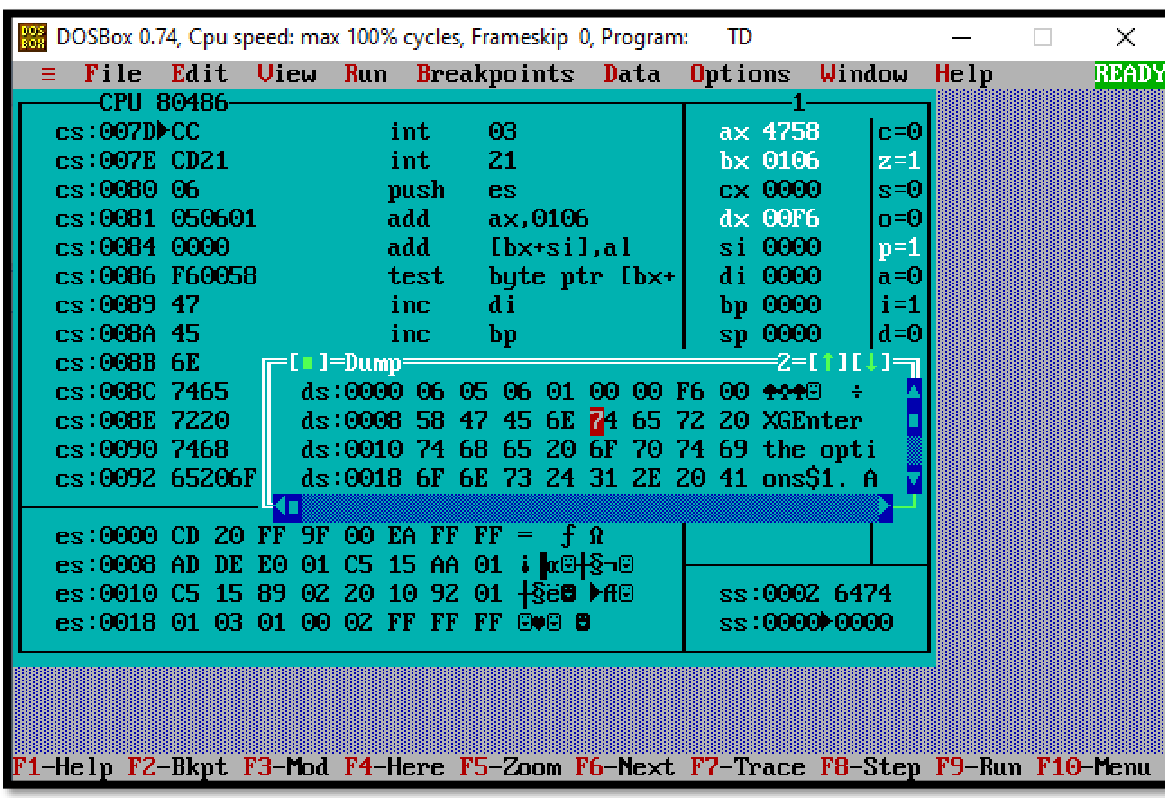
2. Subtraction :



3. Multiplication :



4. Division :



POSTLABS

1. Explain registers of 8086.

Page No.
Date.

ANS: In 8086 Microprocessor, the registers are categorized into mainly four types:
1) General Purpose Registers
2) Segment Registers
3) Pointers and Index Registers
4) Flag or Status Register

1] General Purpose Registers - The use of general-purpose register is to store temporary data. While the instructions are executed in the control unit, they may work on some numeric value or some operands. These need to be stored somewhere so that the processor can operate on them easily. So, these registers are used in these cases. There are 4 general purpose registers of 16-bit length each. Each of them is further divided into 2 subparts of 8 bits length each: one high, which stores the higher order bits and another low which stores the lower order bits.

1) AX = [AH : AL] 3) CX = [CH : CL]
2) BX = [BH : BL] 4) DX = [DH : DL]

2) Segment Registers - There are 4 segment registers in 8086 Microprocessor and each of them is of 16 bit. The code and instructions are stored inside these different segments.
1) Code Segment (CS) Register: The user cannot modify the content of these registers.
2) Data Segment (DS) Register: The user can modify the content of the data segment.
3) Stack Segment (SS) Register: The SS is used to store the information about the memory segment.

4) Extra Segment (ES) Register: If there is less space in that segment, then ES is used. ES is also used for copying purpose.

3) <u>Pointers and Index Registers</u> - The pointers will always store some address or memory location

1) Instruction Pointer (IP): The instruction pointer usually stores the address of the next instruction that is to be executed.

2) Base Pointer (BP): The base pointer stores the base address of the memory.

3) Stack Pointer (SP): The stack pointer points at the current top value of the stack.

4) Source Index (SI): It stores the offset address of the source.

5) Destination Index (DI): It stores the offset address of the destination.

4) <u>Flag or status Register</u> - The flag or status register is a 16-bit register which contains 9 flags, and the remaining 7 bits are idle in this register. These flags tell about the status of the processor after any arithmetic or logical operation. If flag value is 1, the flag is set and if it is 0, it is said to be reset.

2. Explain logical and physical address for 8086 with example.

Page No.:
Date:
YOUVA

ANS:

- Logical address is contained in the 16-bit IP, BP, SP, BX, SI or DI. It is also known as the offset address or the effective address.
- The physical address or the real address is formed by combining the offset and base segment addresses. This address is 20 bit and is primarily used for the accessing of the memory
- The logical address is a virtual address and can be viewed by the user. The user can't view the physical address directly. The logical address is used like a reference, to access the physical address.
- The fundamental difference between logical and physical address is that logical address is generated by CPU during a program execution whereas, the physical address refers to a location in the memory unit.
- The logical address is generated by the CPU while physical address is computed by MMU
- Identical logical address and physical address are generated by compile-time and Load time address binding methods. The logical and physical address generated while run-time address binding method differs from each other.
- The set of all logical addresses generated by CPU for a program is called Logical Address Space. However, the set of all physical address mapped to corresponding logical addresses is referred as Physical Address Space.