# Thadomal Shahani Engineering College

Bandra (W.), Mumbai- 400 050.

## CERTIFICATE

Certify that Mr./Miss Atharva Yadav

of IT Department, Semester IV with

Roll No. 127 has completed a course of the necessary

experiments in the subject MPL under my

supervision in the **Thadomal Shahani Engineering College**

Laboratory in the year 20 - 20

16/4/24

Teacher In-Charge

Head of the Department

Date 16/04/2024

Principal

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

# CONTENTS

| SR. NO. | EXPERIMENTS | PAGE NO. | DATE | TEACHERS SIGN. |
|---|---|---|---|---|
| 1. | Study of PC motherboard technology (Nouth and south bridge), internal components and connections used in computer system (LO5) | LO1 | 18.1.24 | |
| 2 | verify the truth table of various logic gates (basic and universal gates) (LO1) | LO2 | 25.1.24 | |
| 3 | Realize half adder and full adder [LO2] | LO2 | 1.02.24 | |
| 4 | Implementation of MUX and DEMUX (LO2) | LO2 | 8.02.24 | |
| 5 | write a 8086 program for 16 bit | LO3 | 7.03.24 | |
| 6 | write a 8086 program to convert 2 digit packet BCD to unpacked BCD (LO3) | LO3 | 18.03.24 | |
| 7 | write a 8086 program to move set of numbers from one memory block to another (LO4) | LO4 | 21.3.24 | |
| 8 | write 8086 program to count number of 1's 0's in 8 bit number (LO4) | LO4 | 28.3.24 | |
| 9. | write 8086 program to find even odd numbers from a given list (LO4) | LO4 | 4.4.24 | |
| 10. | write 8086 program to check whether a given string is a palindrome or not | LO5 | 4.4.24 | |
| 11. | write 8086 program to compute fact | LO5 | 8.4.24 | |
| 12. | write 8086 program to interfacing seven segment display (LO6) | LO6 | 8.4.24 | |
| 13. | written assg I based on the system | CO1 | 12.4.24 | |
| 14. | written ASSG II both's Algo and division method | CO4 | 12.4.24 | |

marks 16/4/24

marks 16/4/24

**Atharva Yadav**
**Roll No. 127**
**Batch S23**

**MPL Experiment No. 1:**

***Study of PC Motherboard Technology (South Bridge and North Bridge), Internal Components and Connections used in computer system.***

**Motherboard:**

**A motherboard serves as the central hub of a computer system, providing connectivity and support for essential components such as the CPU, memory, storage devices, and expansion cards. It acts as a foundation upon which all other hardware components are connected and communicate. Its importance lies in providing stability, compatibility, and scalability to the system, as different components can be upgraded or replaced as needed without requiring a change in the motherboard itself. Essentially, the motherboard acts as the backbone of a computer, enabling seamless interaction**

and coordination among its various parts to ensure efficient performance. Key functions of motherboard:

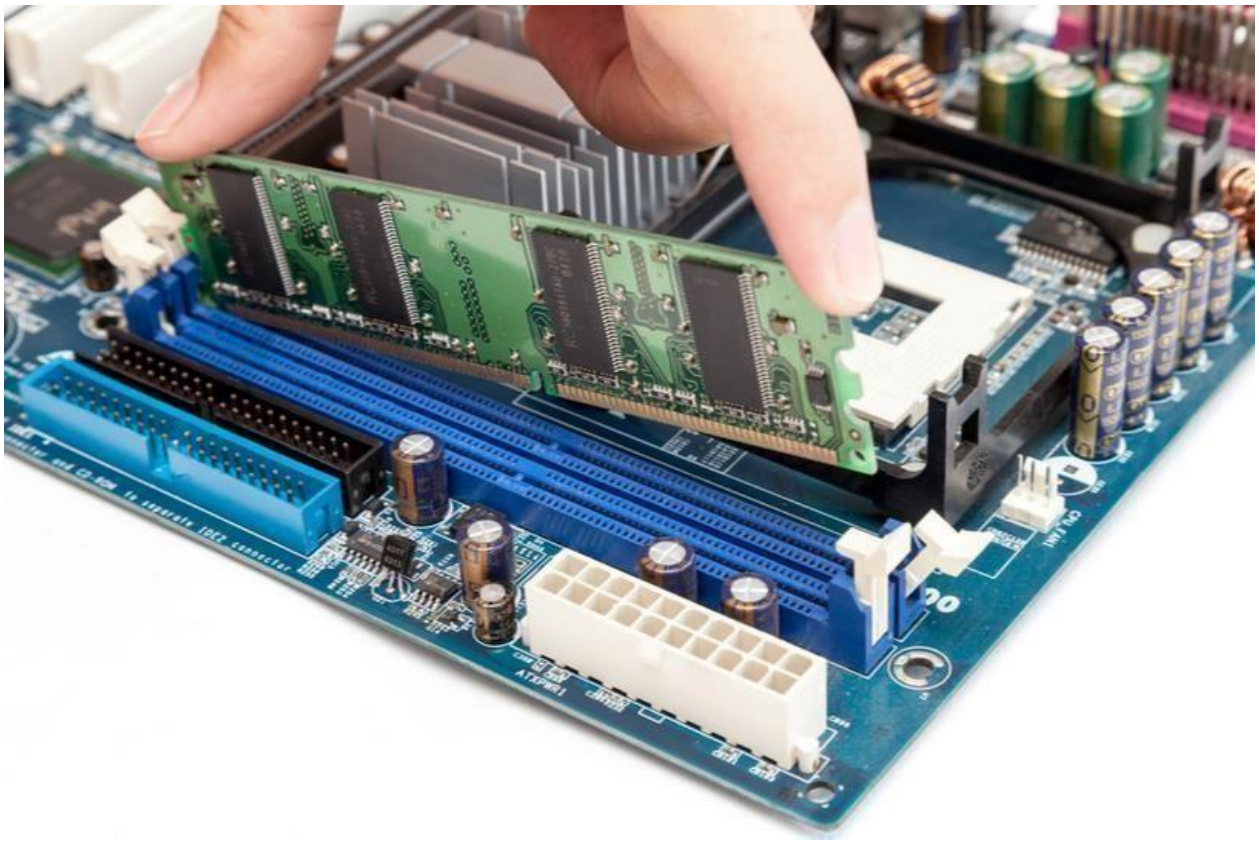
1. Connectivity: The motherboard offers a range of ports and slots for connecting peripherals and expansion cards, including USB ports, audio jacks, Ethernet ports, PCIe slots, and more, enabling seamless integration of various devices into the system.
2. Communication: Through its integrated chipset, the motherboard facilitates communication between the CPU, memory, storage devices, and expansion cards. Data flow is managed efficiently via interfaces like SATA for storage devices and PCIe for high-speed communication with peripherals such as graphics cards and network adapters.
3. Power Delivery: The motherboard serves as the conduit for power distribution from the PSU to the system components. It features connectors such as the 24-pin ATX power connector and CPU power connector, ensuring stable and reliable power delivery to the CPU, memory, expansion cards, and other peripherals.
4. Central Hub: Acting as the central hub of the computer system, the motherboard coordinates and synchronizes the operations of various hardware components, allowing them to work together seamlessly. It provides the foundation upon which the CPU, memory, storage, and other components interact, facilitating the smooth functioning of the entire system.

Key components on the

motherboard CPU Socket:


**The CPU socket on a motherboard serves as the interface between the central processing unit (CPU) and the motherboard. Its purpose is to securely hold the CPU in place and provide electrical connections for power delivery and communication between the CPU and other components on the motherboard, allowing the CPU to process instructions and perform computations within the computer system.**

**RAM Slots:**

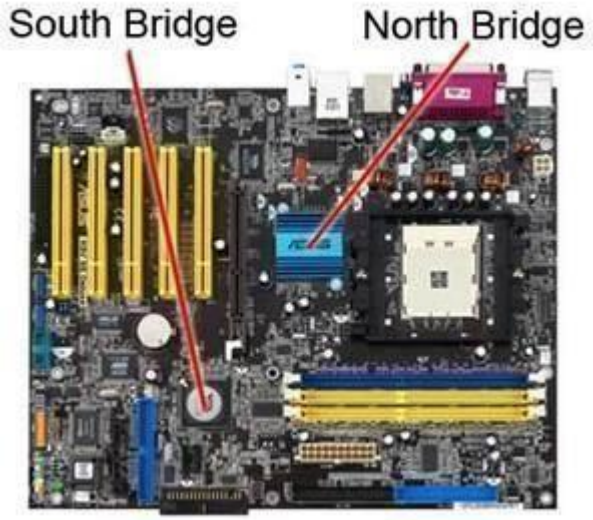


**RAM slots on a motherboard provide the interface for installing random access memory (RAM) modules. Their function is to securely hold the RAM modules in place and provide electrical connections for data transfer between the RAM and the CPU. RAM slots enable the CPU to quickly access and manipulate data, enhancing the overall performance and responsiveness of the computer system. Chipset:**


**The chipset on a motherboard serves as the central nervous system of the system, facilitating communication between the CPU, memory,**

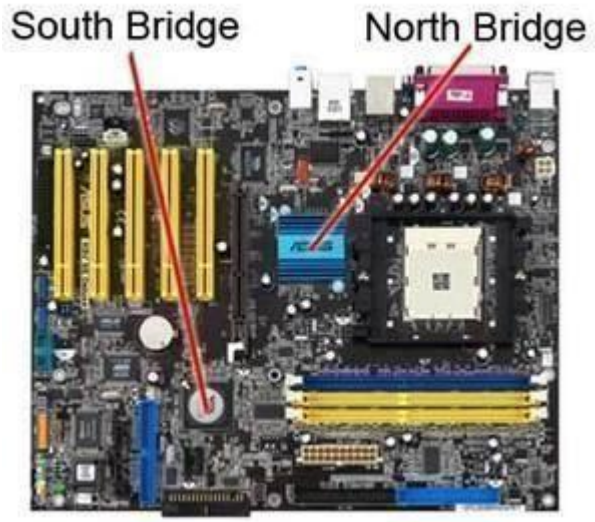
storage devices, and other peripherals. Its role involves managing data flow between these components through various interfaces such as SATA, PCIe, and USB, ensuring efficient operation and coordination within the computer system.

**North Bridge:**




1. **Intermediary Role: The North Bridge chipset serves as a crucial intermediary component, orchestrating high-speed communication between the CPU, RAM, and graphics card. It acts as a central hub for data exchange, optimizing the flow of information between these essential parts of the computer system.**
2. **Memory Management: Functioning as a Memory Controller Hub (MCH), the North Bridge chipset plays a pivotal role in memory management. It regulates the flow of data between the CPU and RAM modules, ensuring swift access to memory resources for processing tasks. This efficient memory management enhances overall system responsiveness and multitasking capabilities.**
3. **Graphics Processing: The North Bridge chipset facilitates communication pathways between the CPU and the graphics card, typically via PCI Express lanes. This enables seamless transfer of graphical data, essential for smooth rendering of images, videos, and immersive gaming experiences. By optimizing graphical data transfer, the chipset enhances graphical performance and overall system efficiency.**

4. **Efficient Data Transfer: Utilizing advanced technologies such as the Front Side Bus (FSB) or Direct Media Interface (DMI), the North Bridge chipset ensures rapid exchange of data between the CPU, RAM, and graphics card. This efficient data transfer mechanism minimizes latency and maximizes system performance, contributing to smoother operation and faster processing speeds.**
5. **System Optimization: The North Bridge chipset's coordination of high-speed communication pathways is integral to system optimization. By efficiently managing data transfer between critical components, it fosters harmonious interaction within the system, resulting in improved speed, responsiveness, and overall efficiency.**
6. **Overall Significance: In summary, the North Bridge chipset plays a pivotal role in enhancing system performance by optimizing communication between the CPU, RAM, and graphics card. Its efficient memory management, seamless graphics processing, and rapid data transfer capabilities contribute to an overall improvement in system speed, multitasking capabilities, and graphical performance.**
7. **Peripheral Connectivity: Beyond managing communication between the CPU, RAM, and graphics card, the North Bridge chipset also facilitates connectivity with peripheral devices. It provides interfaces and pathways for connecting peripherals such as hard drives, optical drives, and expansion cards, ensuring seamless integration and efficient data transfer between these devices and the rest of the system.**
8. **Scalability and Future Compatibility: The North Bridge chipset's design often incorporates features that support scalability and future compatibility. It may include support for various generations of CPUs and graphics cards, as well as expansion options for adding additional RAM or upgrading to newer peripheral devices. This ensures that the motherboard remains viable for longer periods, allowing users to adapt and expand their systems as technology advances. South Bridge:**

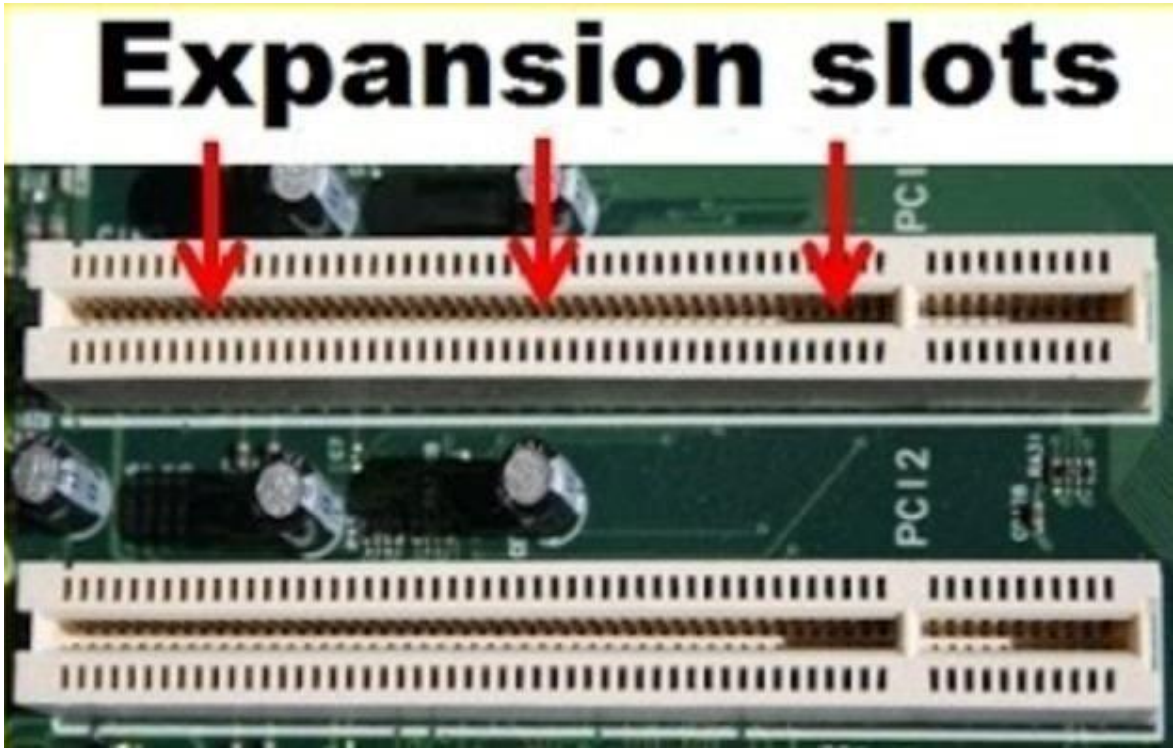



- **Input/Output (I/O) Management: The Southbridge chipset was the central hub for managing slower communication channels within a computer system. It acted as a bridge between the CPU and various peripheral devices.**
- **Peripheral Communication: It directly controlled devices like USB ports, audio jacks, network cards (sometimes), and legacy ports like serial or parallel. This allowed you to connect keyboards, mice, printers, speakers, and other external devices.**
- **Storage Control: The Southbridge managed slower storage interfaces like SATA (Serial ATA) for connecting hard disk drives (HDDs) and solid-state drives (SSDs). While not handling the high-speed data transfer between CPU and memory, it facilitated communication for data retrieval and storage.**
- **System Management Features: The Southbridge often housed the system BIOS (Basic Input/Output System), which is the low-level firmware responsible for booting up the computer and initializing hardware components.**
- **Interrupt Handling: It also managed interrupts, which are signals sent by devices to the CPU requesting attention. This ensured the CPU knew when a device needed data or had a task to complete.**

By handling these slower communication channels, the Southbridge chipset ensured smooth interaction between the CPU, peripherals, and storage devices, keeping your computer functioning efficiently. BIOS:

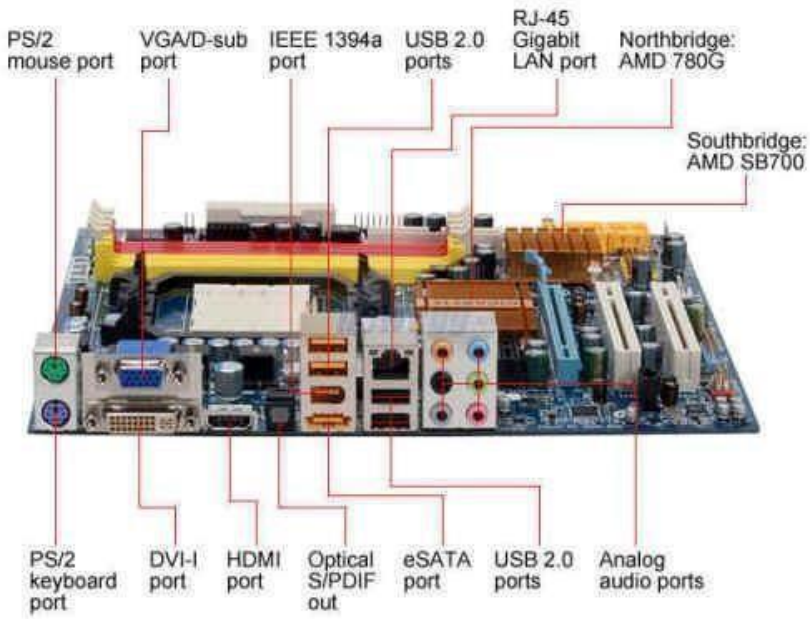

The BIOS (Basic Input/Output System) is a firmware embedded on the motherboard that initializes hardware components during the boot process and provides system configuration options. Its importance lies in its ability to perform essential tasks such as power-on selftest (POST), identifying and initializing hardware devices, and providing a user interface for configuring system settings like boot order, date/time, and hardware parameters.

Expansion Slots:




**Expansion slots, such as PCIe (Peripheral Component Interconnect Express) and PCI (Peripheral Component Interconnect), provide interfaces on the motherboard for adding additional hardware components, such as graphics cards, sound cards, network adapters, and storage controllers. Their significance lies in enabling users to expand and customize their computer systems according to their specific needs and requirements by adding new functionality or upgrading existing components without replacing the entire motherboard.**

**Ports:**




**Ports are physical interfaces on a computer or electronic device that allow for the connection of external peripherals and devices. They play a crucial role in facilitating communication between the computer and external devices, enabling data transfer, power supply, and signal transmission for various functions such as input, output, and storage.**

- **USB (Universal Serial Bus): USB ports are versatile interfaces used for connecting a wide range of peripheral devices to a computer, including keyboards, mice, printers, external storage devices, and smartphones. They facilitate high-speed data transfer, power delivery for charging devices, and support hot-swapping, making them essential for modern computing.**
- **PS/2 ports: PS/2 ports are legacy connectors primarily used for connecting keyboards and mice to a computer. These ports offer a reliable and low-latency connection for input devices, particularly in scenarios where USB ports may not be available or for compatibility with older peripherals.**
- **Video Output Ports (e.g., HDMI, VGA): Video output ports allow for connecting a computer to external displays such as monitors, TVs, or projectors. HDMI ports offer high- definition digital video and audio transmission, while VGA ports provide analog video output. They enable users to extend or mirror**

**their desktops, making them crucial for multimedia, gaming, and presentation purposes.**

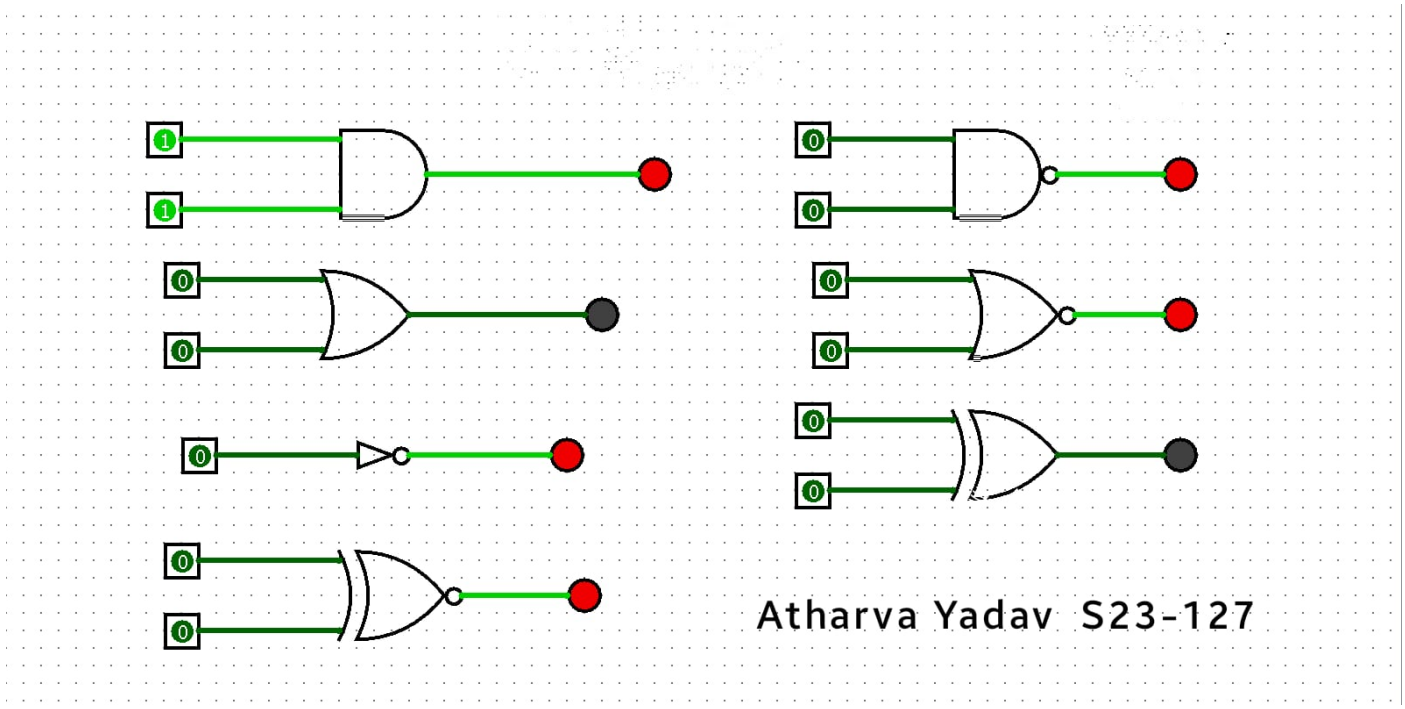
- **Ethernet (RJ45) port: The Ethernet port, typically using an RJ45 connector, provides a wired network connection for a computer to connect to a local area network (LAN) or the internet. It offers fast and reliable data transfer rates, making it essential for activities such as online gaming, video streaming, and file sharing, especially in scenarios where a stable and high-speed internet connection is required.**
- **Audio Ports: Audio ports, including headphone jacks and microphone inputs, allow for connecting audio devices such as headphones, speakers, microphones, and headsets to a computer. They facilitate audio input and output, enabling users to listen to multimedia content, communicate via voice chat, or record audio, enhancing the overall user experience of the system.**

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

ATHARVA YADAV
ROLL NO : 127
BATCH : S23

# MPL ASSIGNMENT NO. 1

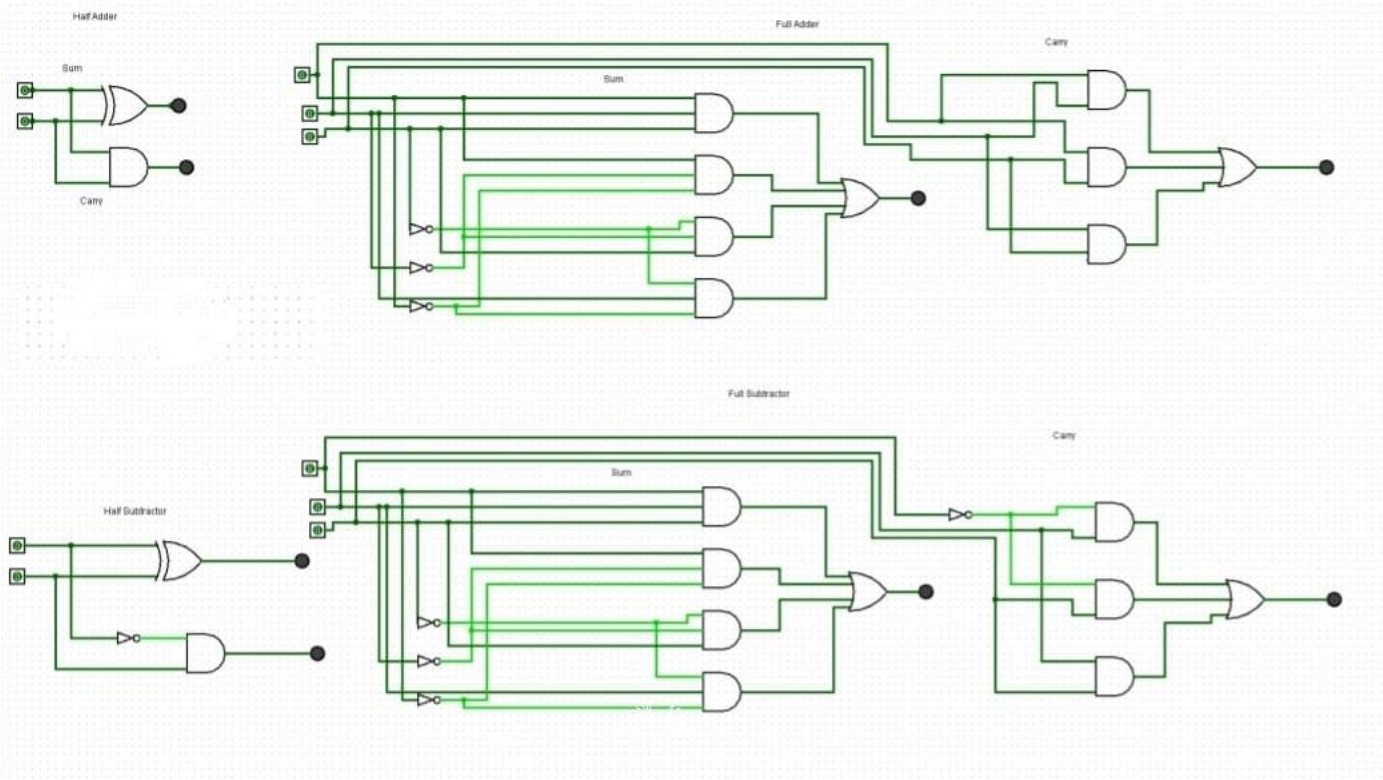
## BASIC GATES



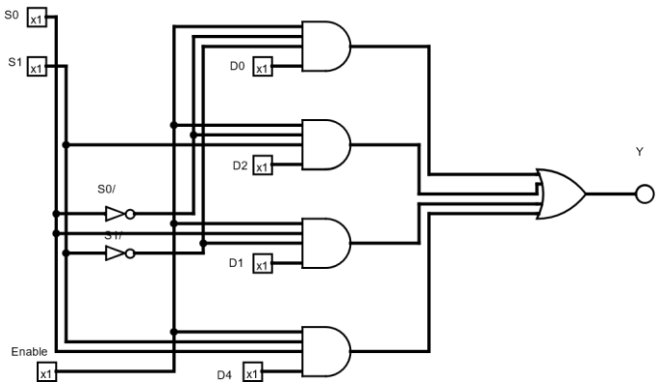
THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

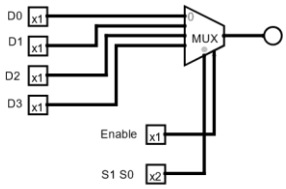
ATHARVA YADAV
ROLL NO : 127
BATCH : S23

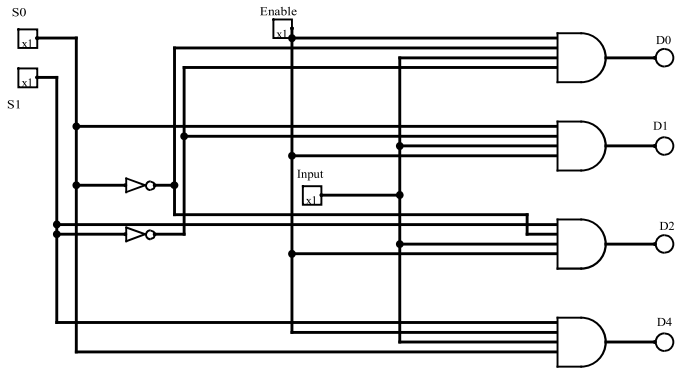
# MPL ASSIGNMENT NO. 2
# ADDER AND SUBTRACTOR(HALF AND FULL)

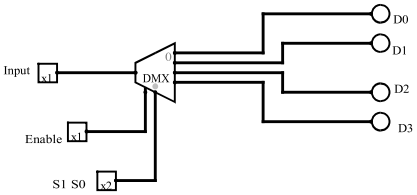



Atharva Yadav S23-127


S0
x1
S1
x1
D0
x1
D2
x1
S0/
S1/
D1
x1
Enable
x1
D4
x1
Y


D0
x1
D1
x1
D2
x1
D3
x1
MUX
Enable
x1
S1 S0
x2


S0
x1
S1
x1
Enable
x1
Input
x1
D0
D1
D2
D4

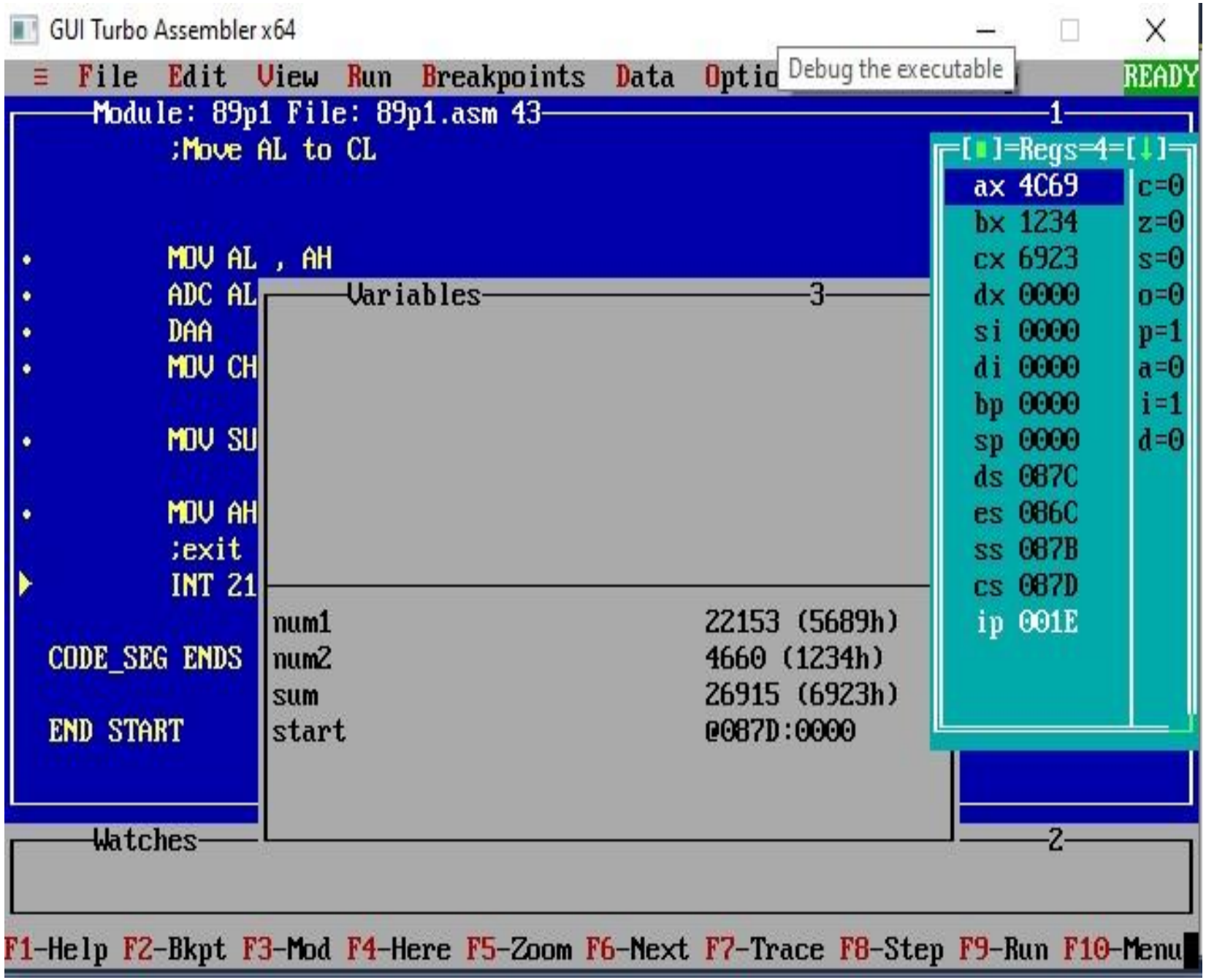

Input
x1
DMX
Enable
x1
S1 S0
x2
D0
D1
D2
D3

ATHARVA YADAV
ROLL NO : 127
BATCH : S23

# MPL ASSIGNMENT NO.5

## AIM: TO PERFORM 16 BIT BCD ADDITION

```
DATA_SEG SEGMENT     NUM1 DW 5689H     NUM2 DW
1234H     SUM DW 0 DATA_SEG ENDS

CODE_SEG SEGMENT

        ASSUME CS: CODE_SEG,DS: DATA_SEG          START:
                MOV AX,DATA_SEG
                ;initialize data segment reg
                MOV DS,AX
                ;Move NUM1 and NUM2 to AX and Bx respectively
                MOV AX , NUM1 MOV BX , NUM2
                ;Add AL and BL save it to AL and do DAA
                ADD AL , BL
                DAA
                MOV CL , AL
                ;Move AL to CL


                MOV AL , AH
                ADC AL , BH
                DAA
                MOV CH , AL

                MOV SUM , CX

                MOV AH,4CH
                ;exit to DOS         INT 21H

CODE_SEG ENDS

END START
```


GUI Turbo Assembler x64
File Edit View Run Breakpoints Data Optio
Debug the executable
READY
Module: 89p1 File: 89p1.asm 43
1
;Move AL to CL
MOV AL , AH
ADC AL
DAA
MOV CH
MOV SU
MOV AH
;exit
INT 21
CODE_SEG ENDS
END START
Variables
3
num1 22153 (5689h)
num2 4660 (1234h)
sum 26915 (6923h)
start @087D:0000
[▮]=Regs=4=[↓]
ax 4C69
bx 1234
cx 6923
dx 0000
si 0000
di 0000
bp 0000
sp 0000
ds 087C
es 086C
ss 087B
cs 087D
ip 001E
c=0
z=0
s=0
o=0
p=1
a=0
i=1
d=0
Watches
2
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

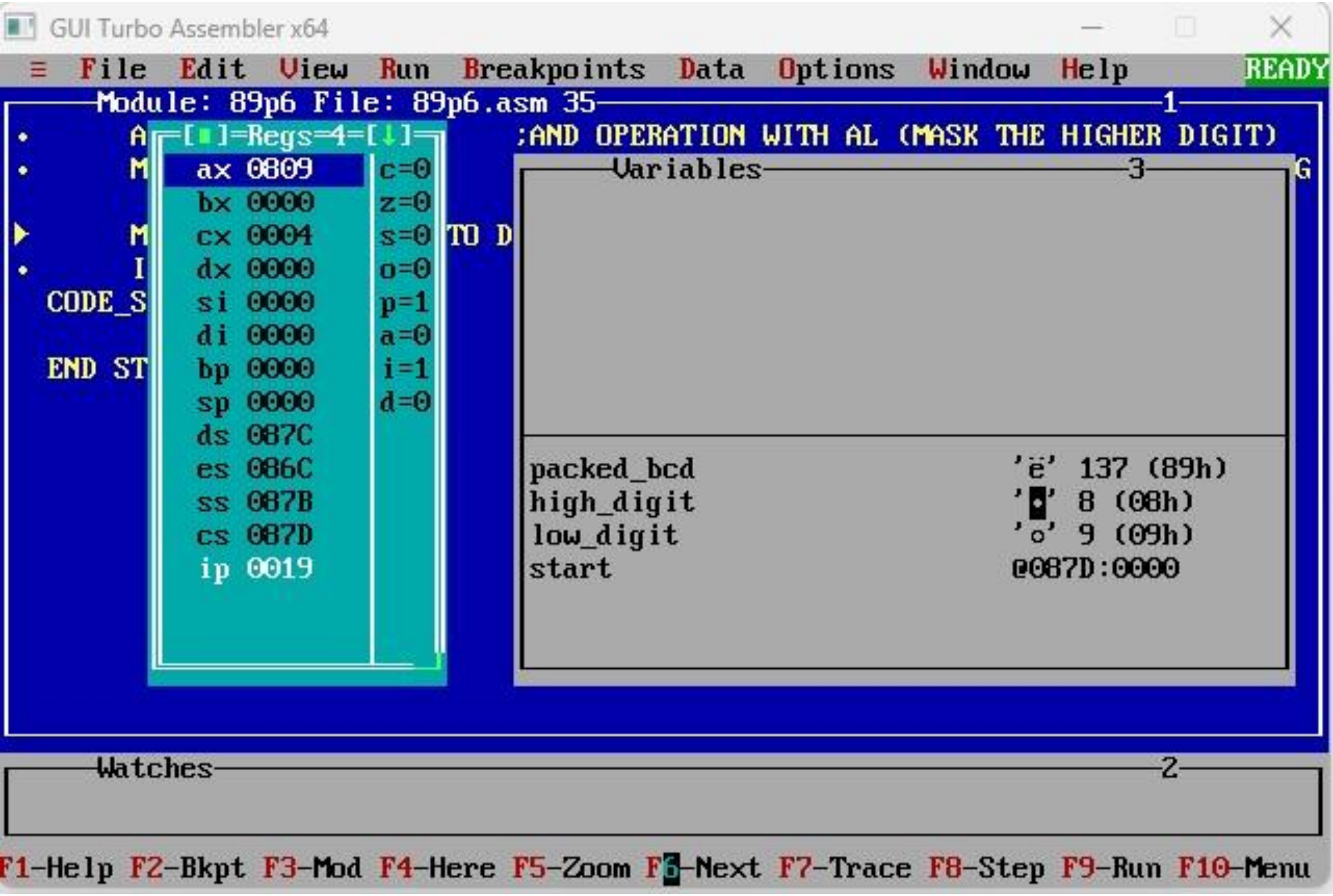
THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

ATHARVA YADAV
ROLL NO : 127
BATCH : S23

# MPL ASSGINMENT NO.6

## AIM: BCD to unpacked BCD

```
DATA_SEG SEGMENT
  PACKED_BCD DB 89H
HIGH_DIGIT DB ?   LOW_DIGIT
DB ?


DATA_SEG ENDS


CODE_SEG SEGMENT
        ASSUME CS:CODE_SEG , DS:DATA_SEG

 START:
        MOV AX , DATA_SEG   ;INITIALIZE THE DATA SEG
        MOV DS,AX

        MOV AL , PACKED_BCD   ;READ THE VALUE OF PACKED_BCD TRANSFERED AL
        AND AL , 0F0H        ;AND OPERATION WITH AL (MASK THE LOWER DIGIT)
        MOV CL , 4
          ROR AL , CL                    ;ROTATE BITS OF AL TO RIGHT BY 4
          MOV HIGH_DIGIT , AL         ;CONTENTS OF AL
(HIGH DIGITS) TRANFERED TO HIGH_DIGIT

        MOV AL , PACKED_BCD
        AND AL , 0FH          ;AND OPERATION WITH AL (MASK THE HIGHER DIGIT) MOV
        LOW_DIGIT , AL         ;CONTENTS OF AL
(LOW DIGITS) TRANFERED TO LOW_DIGIT

        MOV AH ,4CH  ;EXIT TO DOS
        INT 21H
CODE_SEG ENDS

END START
```


GUI Turbo Assembler x64
File Edit View Run Breakpoints Data Options Window Help
READY
Module: 89p6 File: 89p6.asm 35
1
[■]=Regs=4=[↓]
ax 0809 c=0
bx 0000 z=0
cx 0004 s=0
dx 0000 o=0
si 0000 p=1
di 0000 a=0
bp 0000 i=1
sp 0000 d=0
ds 087C
es 086C
ss 087B
cs 087D
ip 0019
;AND OPERATION WITH AL (MASK THE HIGHER DIGIT)
Variables
3
packed_bcd 'ë' 137 (89h)
high_digit '•' 8 (08h)
low_digit 'o' 9 (09h)
start @087D:0000
Watches
2
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

ATHARVA YADAV

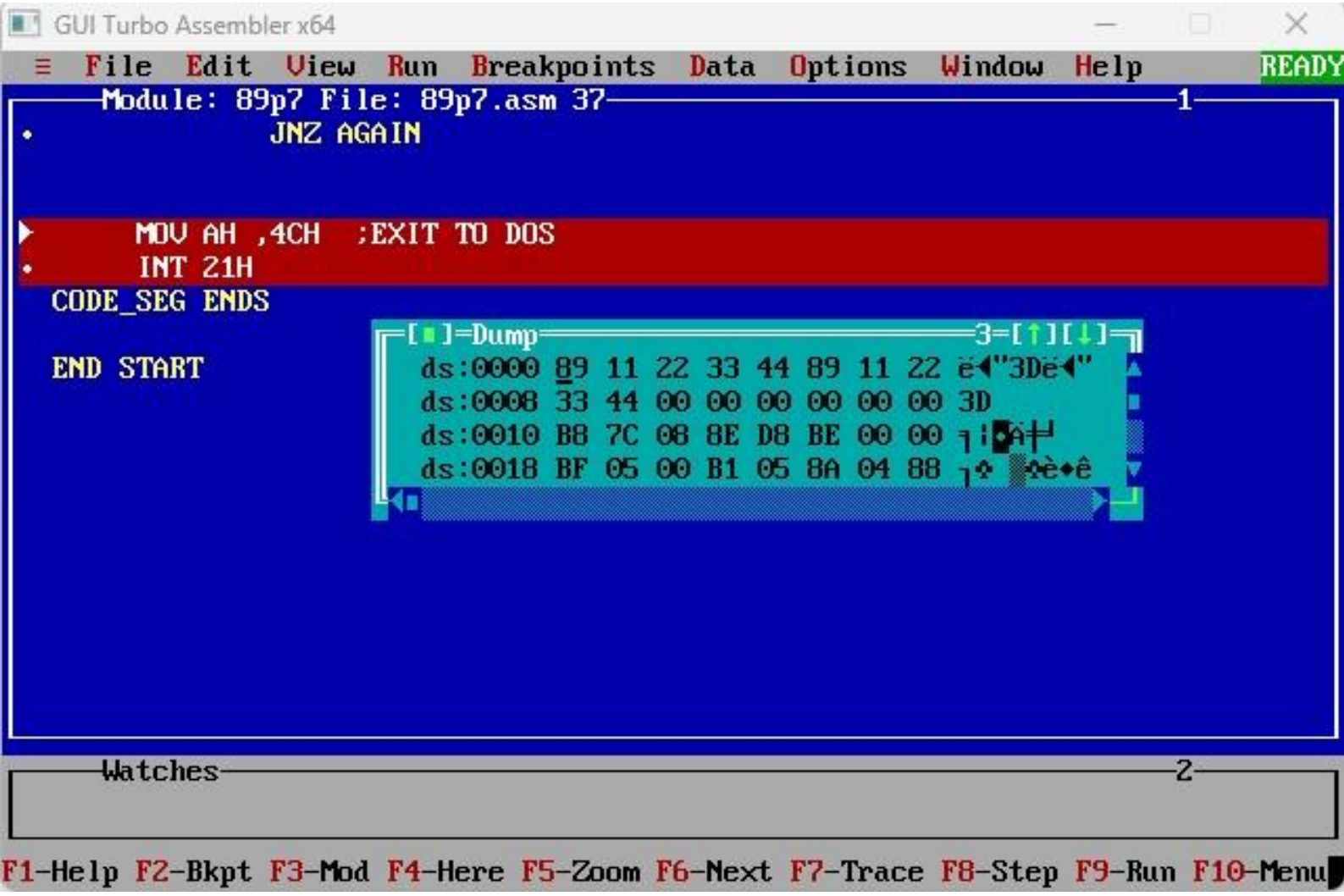
ROLL NO : 127

BATCH : S23

# MPL ASSGINMENT NO.7

## AIM:MOVE SET OF NUMBERS FROM ONE MEMORY BLOCK TO ANOTHER

```
DATA_SEG SEGMENT
  SRCARY DB 89H , 11H , 22H , 33H  , 44H
  DESARY DB 5 DUP (0)
DATA_SEG ENDS

CODE_SEG SEGMENT
        ASSUME CS:CODE_SEG , DS:DATA_SEG

 START:
        MOV AX , DATA_SEG   ;INITIALIZE THE DATA SEG
        MOV DS,AX


        MOV SI , OFFSET SRCARY   ;INITIALIZE POINTER TO SOURCE ARRAY   or  ; LEA SI , SRCARY
        MOV DI , OFFSET DESARY   ;INITIALIZE POINTER TO DESTINATION ARRAY   or  ; LEA DI , DESARY
         MOV CL , 05H                    ;SET THE COUNTER TO 5


    AGAIN : MOV AL , [SI]          ;READ THE ELEMENT FROM THE SOURCE ARRAY
                MOV [DI] , AL                    ;STORE THE ELEMENT IN AL TAKEN FROM THE SOURCE ARRAY TO DESTINATION ARRAY
                INC SI
                INC DI
                DEC CL                           ; DECREMENT THE COUNTER
                JNZ AGAIN


        MOV AH ,4CH  ;EXIT TO DOS
        INT 21H
CODE_SEG ENDS

END START
```

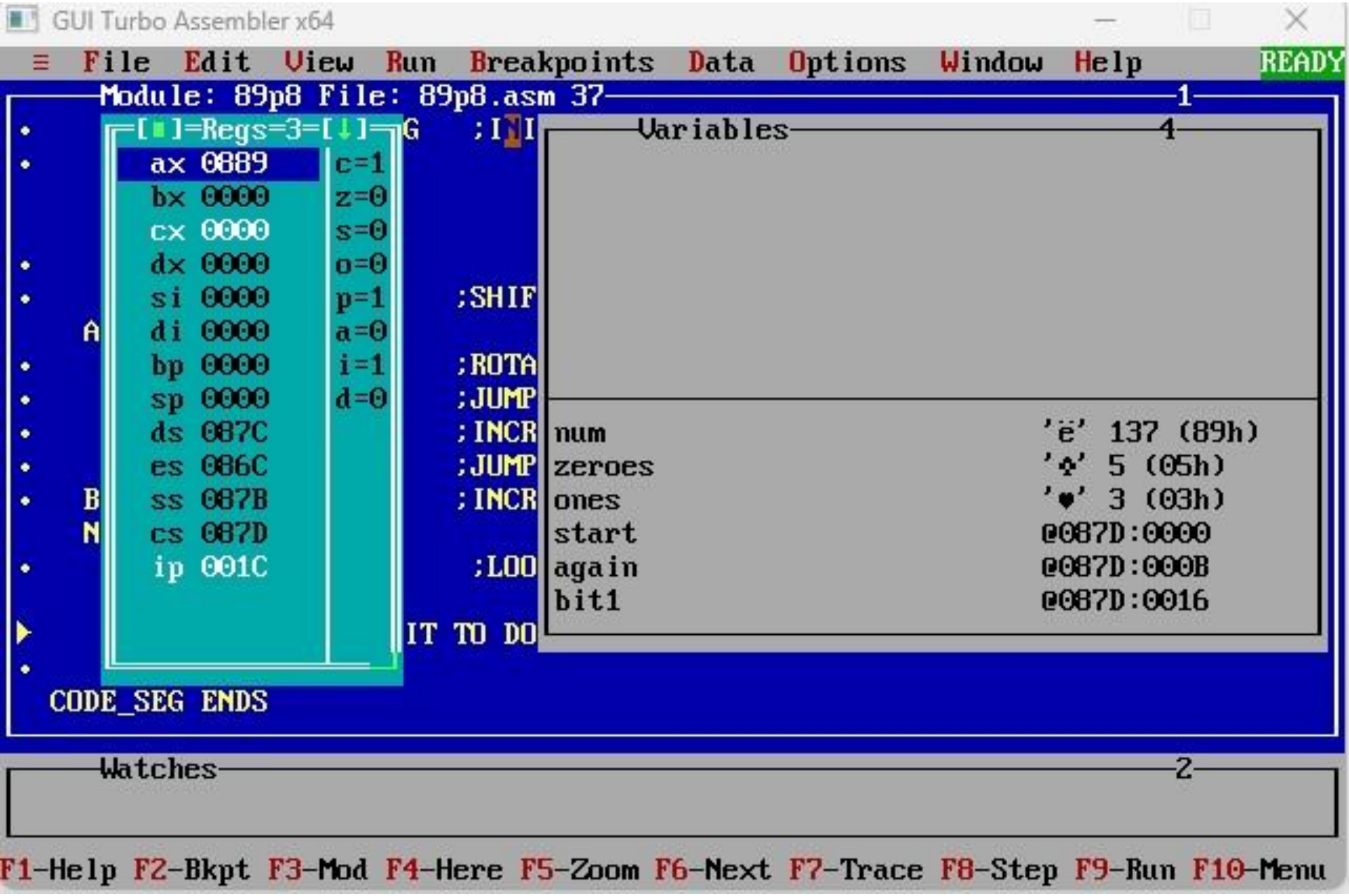

GUI Turbo Assembler x64
File Edit View Run Breakpoints Data Options Window Help READY
Module: 89p7 File: 89p7.asm 37 1
JNZ AGAIN
MOV AH ,4CH ;EXIT TO DOS
INT 21H
CODE_SEG ENDS
END START
[■]=Dump 3=[↑][↓]
ds:0000 89 11 22 33 44 89 11 22
ds:0008 33 44 00 00 00 00 00 00
ds:0010 B8 7C 08 8E D8 BE 00 00
ds:0018 BF 05 00 B1 05 8A 04 88
Watches 2
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

ATHARVA YADAV
ROLL NO : 127
BATCH : S23

# MPL ASSIGNMRNT NO.8

## AIM: TO COUNT NUMBER OF 1S AND NUMBER OF 0S IN A GIVEN 8-BIT NUMBER

```
DATA_SEG SEGMENT

 NUM DB 89H
 ZEROES DB 0   ONES
DB 0

DATA_SEG ENDS


CODE_SEG SEGMENT
        ASSUME CS:CODE_SEG , DS:DATA_SEG

 START:
        MOV AX , DATA_SEG   ;INITIALIZE THE DATA SEG
        MOV DS,AX


        MOV CX , 8
        MOV AL , NUM       ;SHIFT THE NUMBER TO AL
 AGAIN :
           ROR AL , 1                ;ROTATE AL
          JC BIT1          ;JUMP TO BIT 1
          INC ZEROES       ;INCREASE THE ZEROES
          JMP NEXT         ;JUMP TO NEXT
 BIT1: INC ONES       ;INCREAMENT THE ONES
 NEXT:
          LOOP AGAIN        ;LOOP TO AGAIN

        MOV AH ,4CH  ;EXIT TO DOS
        INT 21H
CODE_SEG ENDS

END START
```


GUI Turbo Assembler x64
File Edit View Run Breakpoints Data Options Window Help
READY
Module: 89p8 File: 89p8.asm 37
1
[■]=Regs=3=[↓]
ax 0889
bx 0000
cx 0000
dx 0000
si 0000
di 0000
bp 0000
sp 0000
ds 087C
es 086C
ss 087B
cs 087D
ip 001C
c=1
z=0
s=0
o=0
p=1
a=0
i=1
d=0
;SHIF
;ROTA
;JUMP
;INCR
;JUMP
;INCR
;LOO
IT TO DO
CODE_SEG ENDS
Variables
4
num 'ë' 137 (89h)
zeroes '○' 5 (05h)
ones '♥' 3 (03h)
start @087D:0000
again @087D:000B
bit1 @087D:0016
Watches
2
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

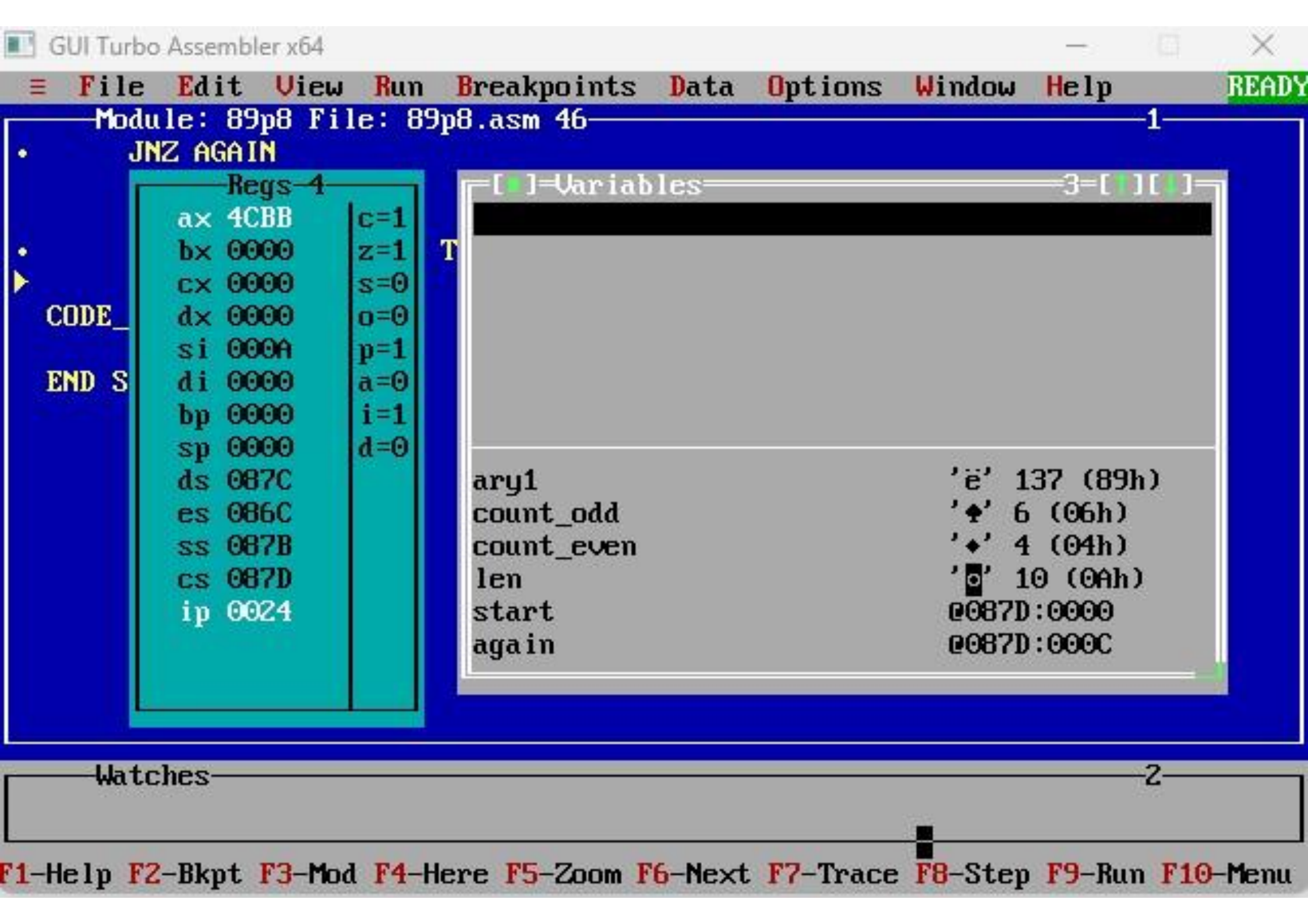
ATHARVA YADAV
ROLL NO : 127
BATCH : S23

# MPL ASSIGNMENT NO. 9

## AIM:TO FIND EVEN AND ODD NUMBERS FROM THE GIVEN LIST

```
DATA_SEG SEGMENT

 ARY1 DB 89H , 31H , 22H , 37H , 44H , 63H , 24H  , 10H , 99H , 77H
 COUNT_ODD DB 0                                          ;INITIALIZE THE COUNTER
 COUNT_EVEN DB 0     ;INITIALIZE THE COUNTER   LEN DB 10          ;LENGTH OF THE
ARRAY

DATA_SEG ENDS


CODE_SEG SEGMENT
        ASSUME CS:CODE_SEG , DS:DATA_SEG

START:
        MOV AX , DATA_SEG   ;INITIALIZE THE DATA SEG MOV
        DS,AX


        LEA SI , ARY1        ;[OR]  MOV SI , OFFSET ARY1 MOV CL , LEN
        ;INITIALIZE THE COUNTER

AGAIN :
        MOV AL , [SI]        ;READ THE ELEMENT
          ROR AL , 1                    ;TEST EVEN OR ODD BY ROTATING AL
        JC ODD1            ;GENERATE A CARRY IF ODD
        INC COUNT_EVEN       ;INCREMENT THE COUNTER IF EVEN
        JMP NEXT           ;JUMP TO THE LABEL NEXT

ODD1 :
        INC COUNT_ODD        ;INCREMENT IF ODD  NEXT :
          INC SI                      ;INCREMENT SI
        DEC CL                       ;DECREMENT THE ARRAY LENGTH
        JNZ AGAIN          ;JUMP TO THE LABEL AGAIN


        MOV AH ,4CH  ;EXIT TO DOS
```

```
        INT 21H
CODE_SEG ENDS

END START
```


GUI Turbo Assembler x64
File Edit View Run Breakpoints Data Options Window Help
READY
Module: 89p8 File: 89p8.asm 46
1
JNZ AGAIN
CODE_
END S
Regs 4
ax 4CBB
bx 0000
cx 0000
dx 0000
si 000A
di 0000
bp 0000
sp 0000
ds 087C
es 086C
ss 087B
cs 087D
ip 0024
c=1
z=1
s=0
o=0
p=1
a=0
i=1
d=0
Variables
3
ary1 'ë' 137 (89h)
count_odd '♠' 6 (06h)
count_even '♦' 4 (04h)
len '◙' 10 (0Ah)
start @087D:0000
again @087D:000C
Watches
2
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

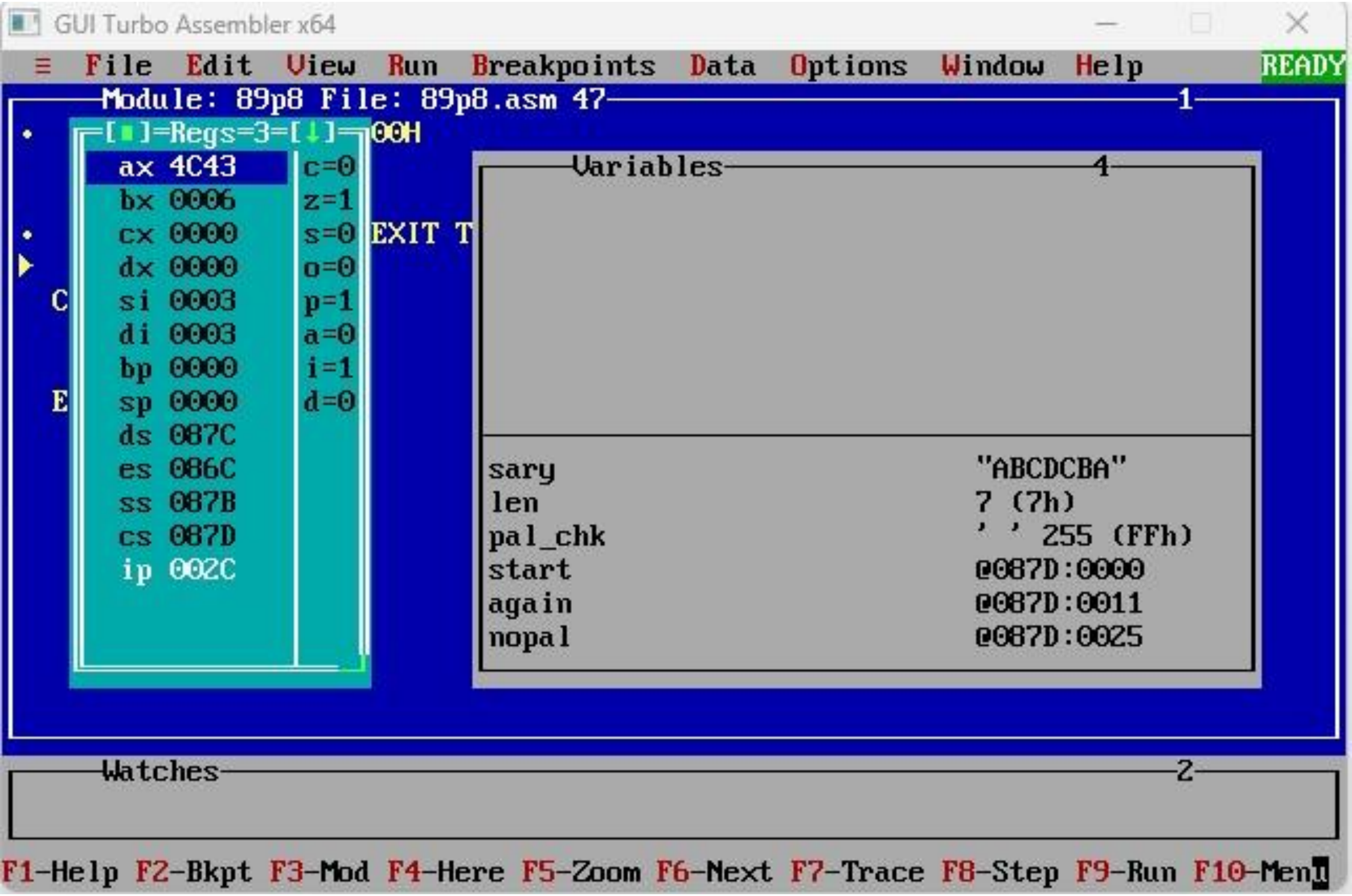
THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

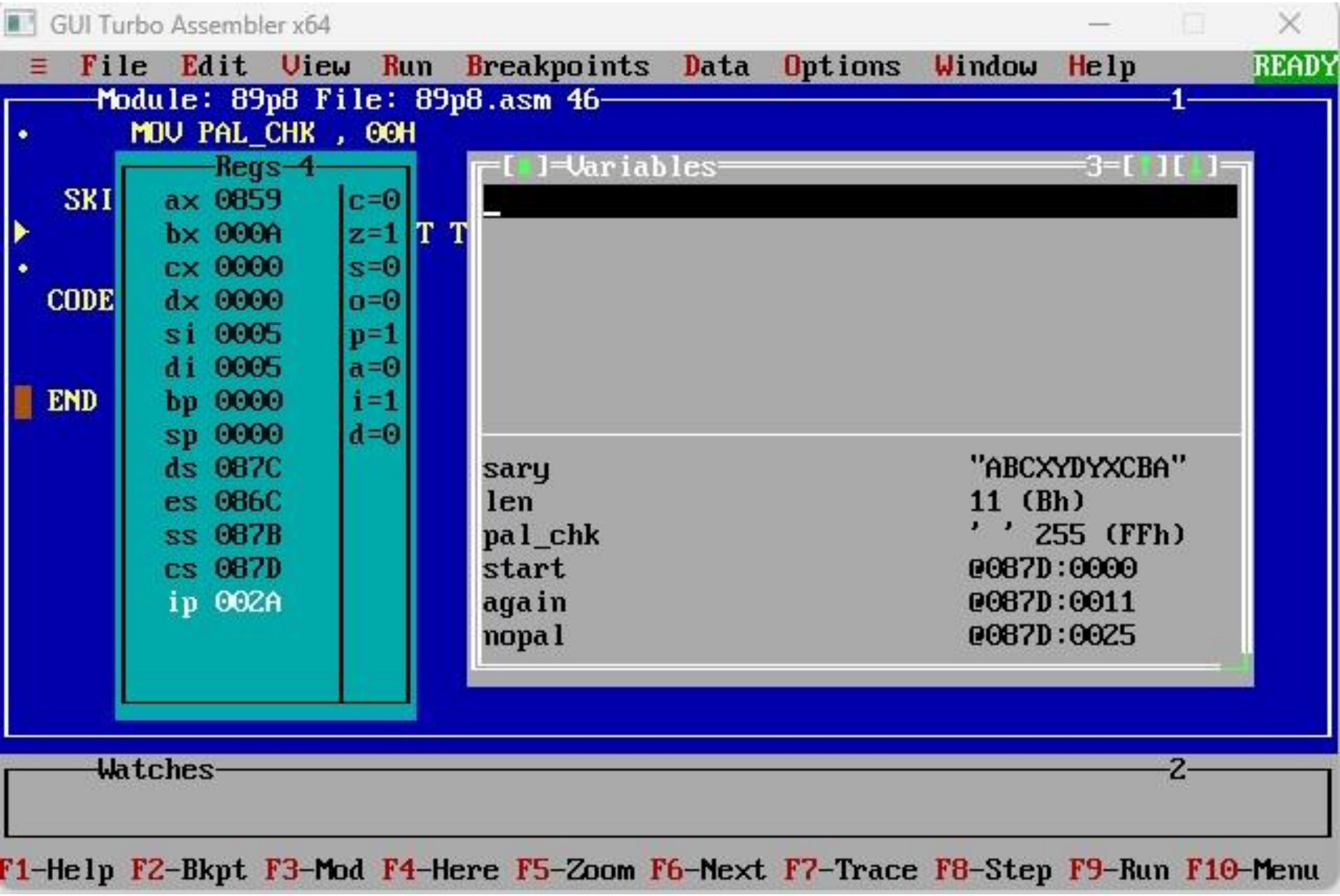
ATHARVA YADAV
ROLL NO : 127
BATCH : S23

# MPL ASSIGNMENT NO.10

## AIM:TO CHECK WHETHER THE GIVEN STRING IS PALINDROME OR NOT

```
DATA_SEG SEGMENT

 SARY DB 'ABCXYDYXCBA'                          ;STRING ON WHICH THE OPERATION US TO BE PERFORMED
 LEN DW 11                                      ;INITIALIZE THE COUNTER WITH LENGTH OF THE STRING
 PAL_CHK DB ?                  ;PALINDROME CHECK

DATA_SEG ENDS


CODE_SEG SEGMENT
        ASSUME CS:CODE_SEG , DS:DATA_SEG

 START:
        MOV AX , DATA_SEG   ;INITIALIZE THE DATA SEG
        MOV DS,AX


        MOV SI , OFFSET SARY
          MOV DI , SI                   ;MOVE SOURCE POINTER TO DESTINATION
        MOV BX , LEN          ;MOVE THE LENGTH TO BX
        DEC BX                          ;DECREMENT BX
          ADD DI , BX                   ;CHANGE DI
 AGAIN :
        MOV AL , [SI]
        CMP AL , [DI]     ;COMPARE START AND END POINTER
        JNE NOPAL         ;JUMP IF NOT EQUAL
          INC SI                ;INCREMENT SI
        DEC DI                  ;DECREMENT DI
          CMP SI , DI           ;COMPARE SI AND DI
        JC AGAIN          ;JUMP TO LABEL AGAIN
        MOV PAL_CHK , 0FFH ;IF PALINDROME STORE FFH
        JMP SKIP          ;JUMP TO LABEL SKIP

 NOPAL :
        MOV PAL_CHK , 00H

 SKIP :
        MOV AH ,4CH  ;EXIT TO DOS
        INT 21H
CODE_SEG ENDS


END START
```


GUI Turbo Assembler x64
File Edit View Run Breakpoints Data Options Window Help
READY
Module: 89p8 File: 89p8.asm 47
1
[■]=Regs=3=[↓]
00H
ax 4C43
bx 0006
cx 0000
dx 0000
si 0003
di 0003
bp 0000
sp 0000
ds 087C
es 086C
ss 087B
cs 087D
ip 002C
c=0
z=1
s=0
o=0
p=1
a=0
i=1
d=0
EXIT T
Variables
4
sary "ABCDCBA"
len 7 (7h)
pal_chk ' ' 255 (FFh)
start @087D:0000
again @087D:0011
nopal @087D:0025
Watches
2
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

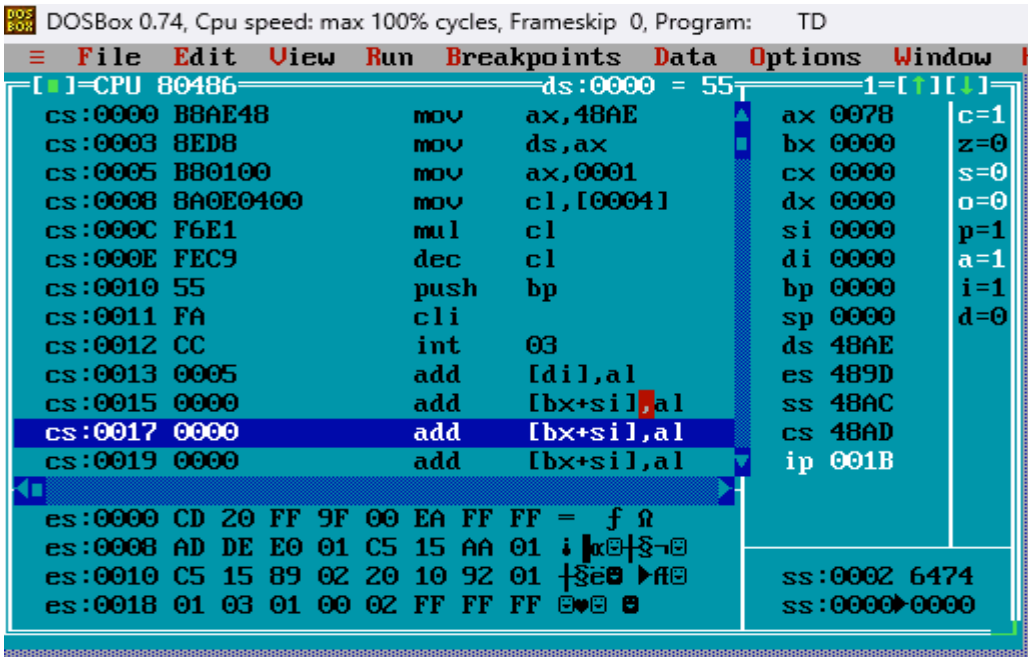

GUI Turbo Assembler x64
File Edit View Run Breakpoints Data Options Window Help
READY
Module: 89p8 File: 89p8.asm 46
1
MOV PAL_CHK , 00H
Regs 4
ax 0859
bx 000A
cx 0000
dx 0000
si 0005
di 0005
bp 0000
sp 0000
ds 087C
es 086C
ss 087B
cs 087D
ip 002A
c=0
z=1
s=0
o=0
p=1
a=0
i=1
d=0
SKI
CODE
END
T T
Variables
3
sary "ABCXYDYXCBA"
len 11 (Bh)
pal_chk ' ' 255 (FFh)
start @087D:0000
again @087D:0011
nopal @087D:0025
Watches
2
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

## Assignment No 11

**Aim:** Write a program for factorial

**Code:**


**Output:**



Atharva Yadav
Roll No 127
Batch S23

## Aim:

To Study Proteus Simulation using 7 Segment Display

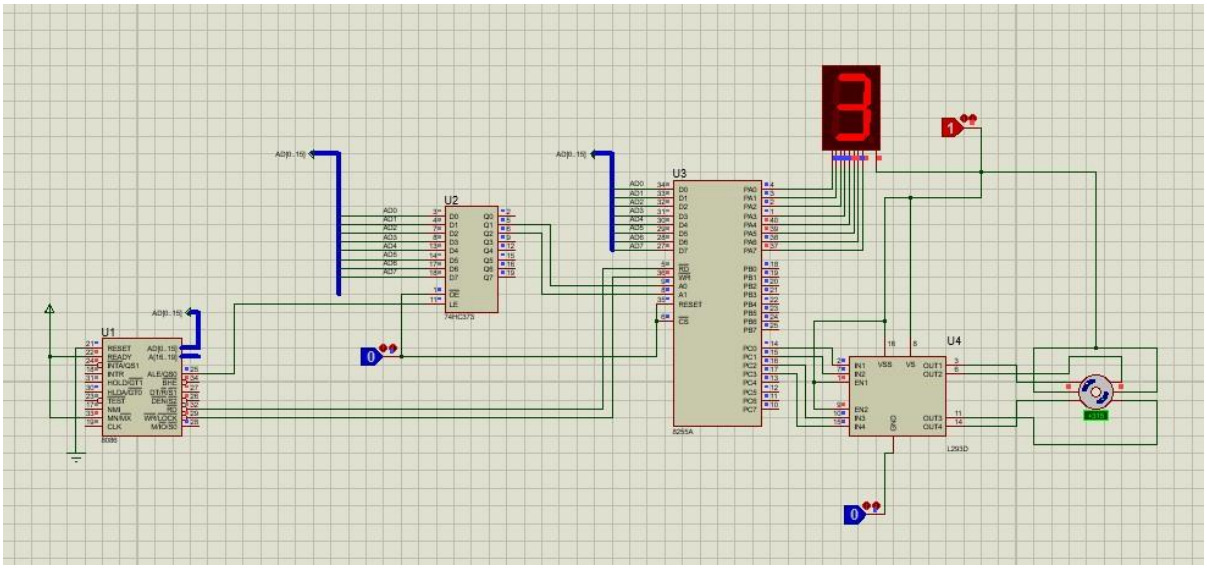
## Code:

```
DATA SEGMENT
  PORTA EQU
  00H PORTB
   EQU      02H
  PORTC EQU
  04H
  PORT_CON EQU
06H DATA ENDS

CODE
  SEGMENT
   MOV     AX,
  DATA MOV
  DS, AX

  ORG

0000H

START:
  MOV DX, PORT_CON MOV AL,
  10000000B OUT DX, AL

  MOV SI, 0
  MOV DI, 0

  L0: MOV CX,
 1FFFH L1: MOV AL, S1[SI] MOV
   DX, PORTA
```

```
  OUT DX, AL LOOP
  L1

  INC SI
  CMP SI,
  16 JL L0

   MOV         DX,
  PORT_CON MOV
  AL, 10000000B
  OUT DX, AL

LL0:MOV CX, 2FFFH
LL1:MOV AL,
S2[DI] MOV DX,
PORTC

 OUT DX,
  AL
  LOOP
  LL1

  INC DI
  CMP DI,
  4 JL LL0

 JMP

 START

 ORG

 1000H
 S1 DB 11000000B DB
  11111001B
  DB
  10100100B
  DB
  10110000B
  DB
  10011001B
  DB
  10010010B
  DB
```

```
 10000010B
 DB
 11011000B
 DB
 10000000B
 DB
 10010000B
 DB
 10001000B
 DB
 10000011B
 DB
 11000110B
 DB
 10100001B
 DB
 10000110B DB
 10001110B

S2 DB
 1101B
 DB
 1011B
 DB
 0111B DB
 1110B

CODEEND
S END
```

# Output:



ATHARVA YADAV
ROLL NO. 127
BATCH: S23
MPL

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

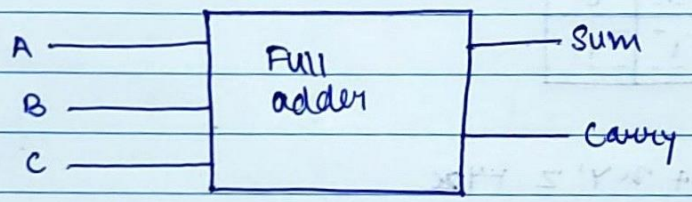
16/4/24

# MPL Assignment 1

Q] Implement Full adder and full subtractor using basic gates and universal gates

Ans

Full Adder is the adder that adds three inputs and produces two outputs. The first two inputs are A and B and the third input is an input carry-I-IN




| A | B | C | Sum | Carry |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Using K-map For Sum

| x \ yz | y'z' | y'z | yz | yz' |
|---|---|---|---|---|
| x' | 0 | 1 | 0 | 1 |
| x | 1 | 0 | 1 | 0 |

Sum (F) = $x'y'z + x'yz' + xy'z' + xyz$

= $x \oplus y \oplus z$

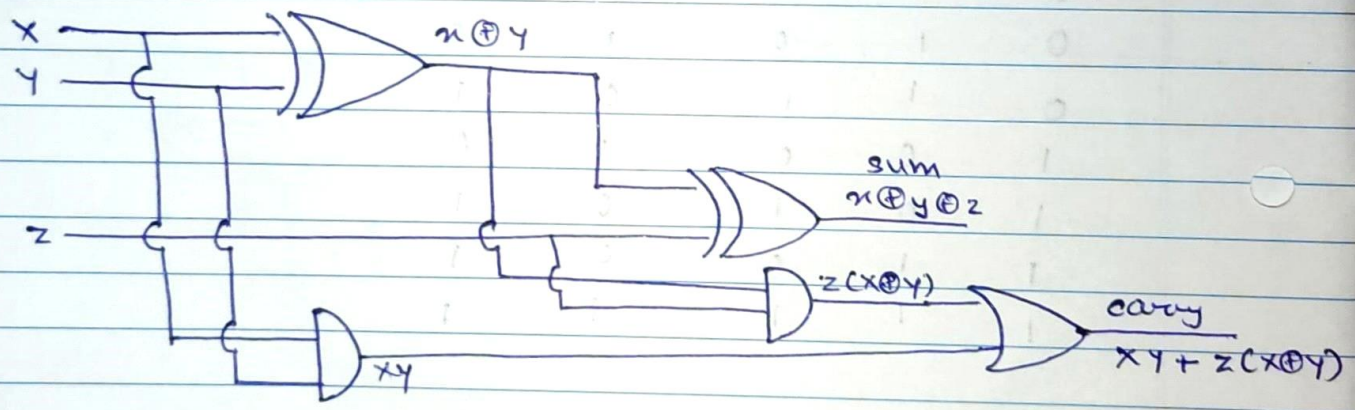
using K-map for Carry

| | $y'z'$ | $y'z$ | $yz$ | $yz'$ |
|---|---|---|---|---|
| $x'$ | 0 | 0 | 1 | 0 |
| $x$ | 0 | 1 | 1 | 1 |

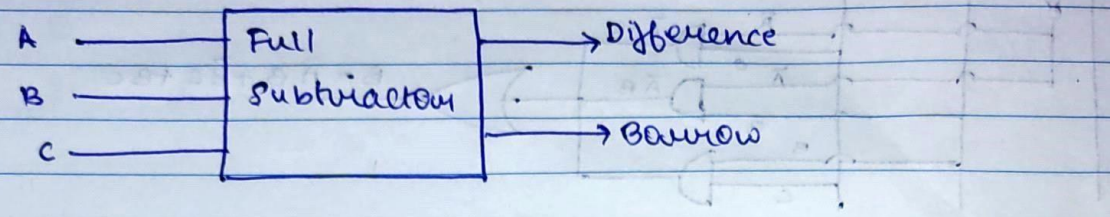
Carry (F) = $x'yz + xy'z + yx$

= $xy + z(x \oplus y)$

Logic Diagram



## Full Subtractor




| A | B | C | Sub | Borrow |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

using Kmap for S: Difference

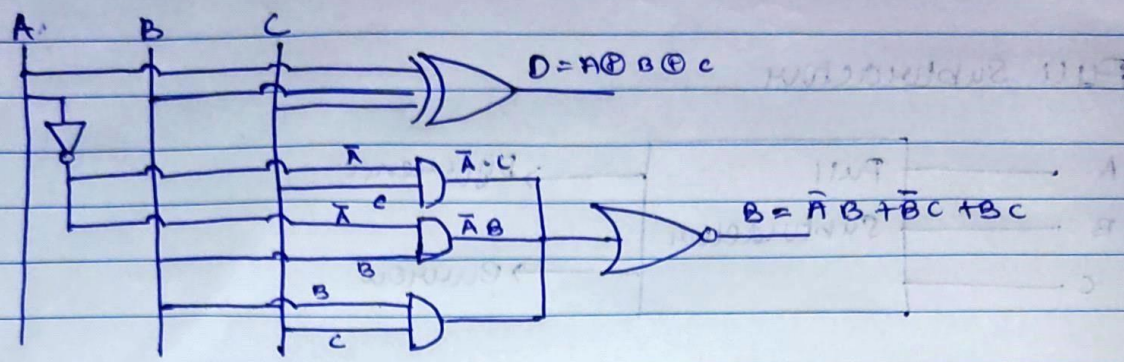
| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |

difference (S) = A ⊕ B ⊕ C

using Kmap for Borrow

| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 |

Borrow (S) = $A'C + A'B + BC$




Full subtractor.

Q] Write a short note on different types of flip flop with their truth table and characteristics equations.

Ans Flip flop is a digital circuit that stores a binary bit. In flip flop the clock signal controls the state of device. It is also called as memory element or binary storage device

There are basically 4 types of flip flop in digital electronics

1. SR Flip Flop
2. JK Flip Flop
3. D Flip flop
4. T Flip Flop

SR: This is the most common flip flop among all. The simple flip flop circuit has a set input (S) and reset input (R). In this, when you set 'S' as active the output 'Q' would be high and 'Q̄' would be low. Once the outputs are established, the wiring of the circuit is maintained until 'S' and 'R' go high or power is turned off.

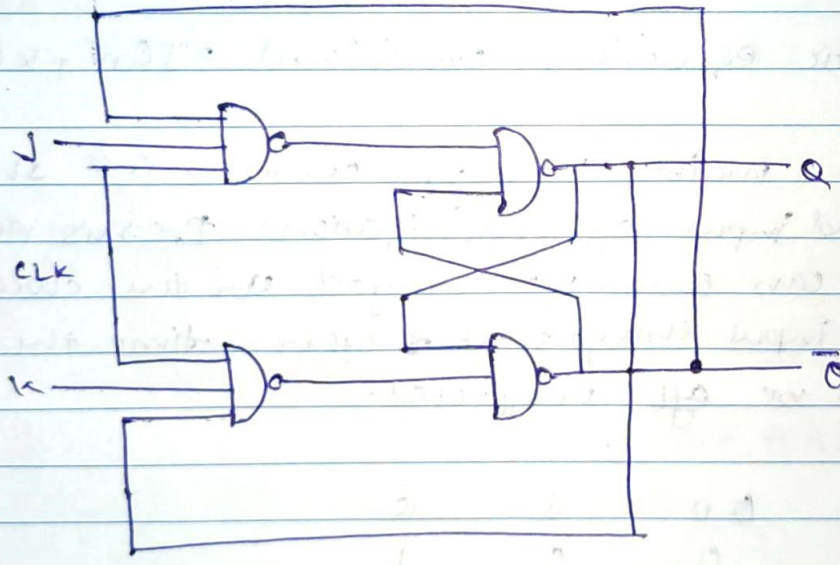
| S | R | Q | $\bar{Q}$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | ∞ | ∞ |

Characteristic equation is $Q_n\bar{R}+S$ or

$$Q(n+1) = S+\bar{R}Q_n$$

2. JK Flip Flop

Due to the undefined state in the SR flip flop another flipflop is required in electronics. The JK flip flop is an improvement on these where S=R=1 is not problem




The input condition for J=K=1 gives an output ~~to~~ inverting the output state. However the outputs are the same when one tests the circuit practically in simpler

words. If J and K data are different ie High and low then the output Q takes the values of J and ~~at~~ to the next clock edge. If ~~at~~ J and K both are low then no change occurs

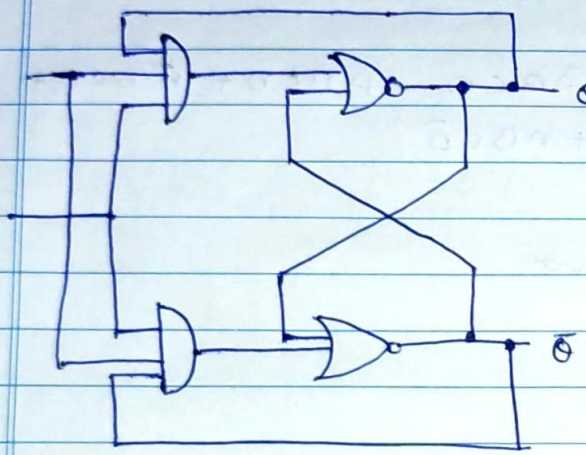
| J | K | Q | $\bar{Q}$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 |

Characteristic equations is $Q(nH) = J\bar{Q}n + K\bar{Q}n$

D Flip Flop:- mainly used for counters and shift register and input synchronization. In this flipflop the output can only be changed on the clock edge and if the input changes at ~~a~~ other time the output will be ~~off~~ unaffected.

| Clock | D | Q | $\bar{Q}$ |
|---|---|---|---|
| ↓ 0 | 0 | 0 | 1 |
| ↑ 1 | 0 | 0 | 1 |
| ↓ 0 | 1 | 0 | 1 |
| ↑ 1 | 1 | 1 | 0 |

T Flip Flop : A T Flip Flop is like a JK Flip Flop there are basically single input version of JK Flip Flop This modified from the JK is obtained by connecting inputs JDK together




| T | Q | Q(t+1) |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

$Q(n+1) = T\bar{Q}n + T\bar{Q}n$

Q8] Implement using Kmap

1. $f(\) = m(1,3,5,6,9,10,13) + d(0,2,14,15)$

| | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | X | 1 | 1 | X |
| 01 | 0 | 1 | 0 | 1 |
| 11 | 0 | 1 | X | X |
| 10 | 0 | 1 | 0 | 1 |

Boolean Alg.
$C\bar{D} + \bar{A}B + \bar{C}D$

$\bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}B\bar{C}\bar{D} + \bar{A}\bar{B}CD + \bar{A}B\bar{C}D + \bar{A}BC\bar{D} + A\bar{B}\bar{C}D + A\bar{B}CD + AB\bar{C}D + ABC\bar{D}$

11. $f(\) = \Sigma m(1,3,5,6,9,10,13) + d(0,2,7,14,15)$

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | X | 1 | 1 | X |
| 01 | | 1 | X | 1 |
| 11 | | 1 | X | X |
| 10 | | 1 | | 1 |

$\bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}B\bar{C}\bar{D} + \bar{A}\bar{B}CD + \bar{A}B\bar{C}D + \bar{A}BC\bar{D} + A\bar{B}\bar{C}D + A\bar{B}C\bar{D} + AB\bar{C}D + ABC\bar{D}$

Q7] Perform using 2's complement

i) $(52)_{10} - (65)_{10}$

$(52)_{10} \rightarrow 0110100$

$(65)_{10} \Rightarrow 1000001$

$-(65)_{10} \rightarrow 0111111$

$$\begin{array}{r} 0110100 \\ + \quad 0111111 \\ \hline 1101011 \end{array} \rightarrow \text{2's complement}$$

$-(0001101) = -13$

ii) $(27)_{10} - (32)_{10}$

$27 \rightarrow 011011$

$32 \rightarrow 100000$

$-32 \rightarrow 100000$

$$\begin{array}{r} 011011 \\ + \quad 100000 \\ \hline 111011 \end{array} \rightarrow \text{2's complement}$$

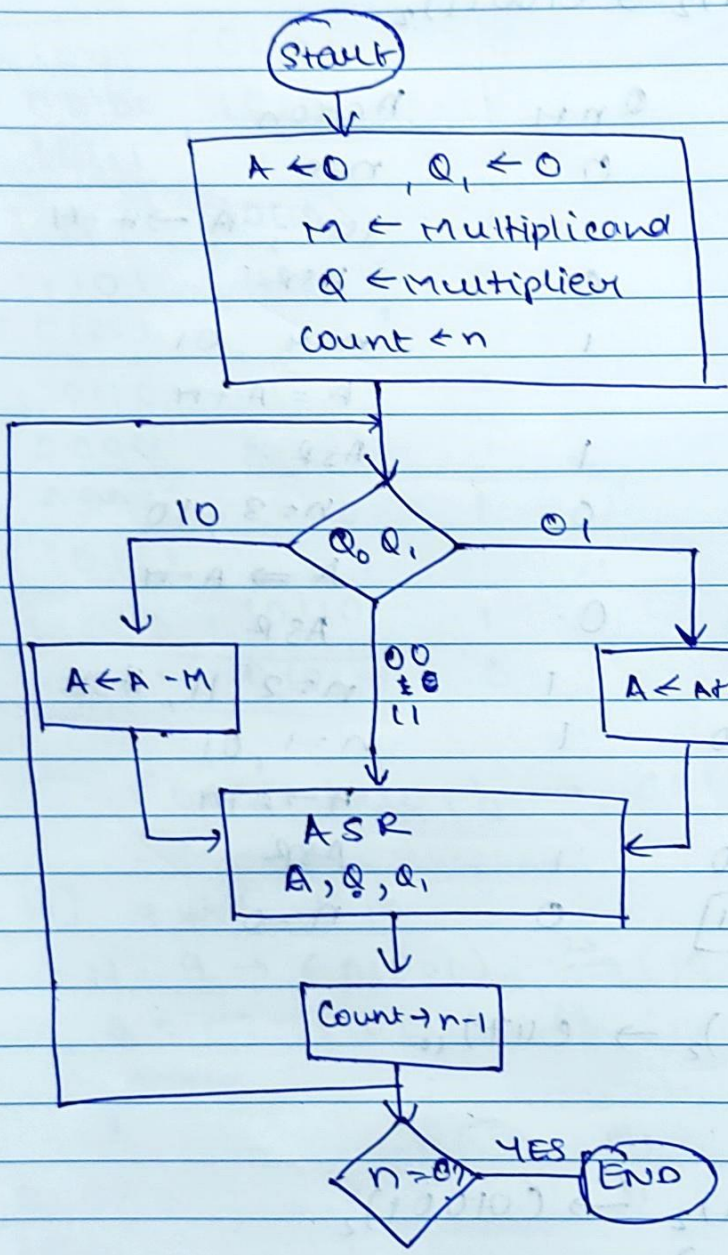
$-(000101) = -5$

Atharva Yadav
ROLLNO. ~~52~~ 127
Batch: S23

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

MPL Written Assignment 2 16/4/24

Q] Draw the flow chart of Booth's Algo. Multiply the following using the same.



1) 9 with 13

$M = 9_{10} = (01001)_2 \xrightarrow{2's} (10111)_2$

$Q = 13_{10} = (01101)_2 \xrightarrow{2's} (10011)_2$

| | A | $Q_n$ | $Q_{n+1}$ | Action |
|---|---|---|---|---|
| | 00000 | 01101 | 0 | n=5 |
| + | 10111 | | | 10 ; A → A − M |
| | 10111 | 01101 | 0 | ASR |
| * | 11011 | 10110 | 1 | n=4 , 01 |
| + | 01001 | | | A = A + M |
| 4 | 00100 | 10110 | 1 | ASR |
| | 00010 | 01011 | 0 | n=3 , 10 |
| | 10111 | | | A ⇒ A − M |
| | 11001 | 01011 | 0 | ASR |
| | 11100 | 10101 | 1 | n=2 , 11 , ASR |
| | 11110 | 01010 | 1 | n=1 , 01 |
| + | 01001 | | | A → A + M |
| 1 | 00111 | 01010 | 1 | ASR |
| | 00011 | 10101 | 0 | n=0 |

$(11101011)_2 \rightarrow (117)_{10}$

2] −9 with 13

$M = -9 = (10111)_2 \xrightarrow{2's} (01001)_2$

$Q = 13 \Rightarrow (01101)_2$

A + M → A + 10111

A + M → A + 01001

S23-127

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

| A | Qn | Qn+1 | Action |
|---|---|---|---|
| 00000 | 01101 | 0 | n=5, 10 |
| +01001 | | | A → A − M |
| 01001 | 01101 | 0 | ASR |
| + 00100 | 10110 | 1 | n=4, 01 |
| 10111 | | | A → A+M |
| 11011 | 10110 | 1 | ASR |
| 11101 | 11011 | 0 | n=3, 10 |
| + 01001 | | | A → A − M |
| 00110 | 11011 | 0 | ASR |
| 00011 | 01101 | 1 | n=2, 11, ASR |
| + 00001 | 10110 | 1 | n=1, 01 |
| 10111 | | | A → A+M |
| 11000 | 10110 | 1 | ASR |
| 11100 | 01011 | 0 | n=0 |

$(00011\ 10101)_2 \rightarrow (+117)_{10}$

iii) 9 with −13

M: $9 \rightarrow (01001)_2 \xrightarrow{2s} (10111)_2$

Q: $-13 \rightarrow (10011)_2 \xrightarrow{2s} (01101)_2$

| A | Qn | Qn+1 | Action |
|---|---|---|---|
| 00000 | 10011 | 0 | n=5, 10 |
| + 10111 | | | A → A − M |
| 10111 | 10011 | 0 | ASR |
| 11011 | 11001 | 1 | n=4, 11, ASR |
| 11101 | 11100 | 1 | n=3, 01 |
| + 01001 | | | A → A+M |
| 00110 | 11100 | 1 | ASR |

| | | | |
|---|---|---|---|
| 00011 | 01110 | 0 | n=2, 00, ASR |
| 00001 | 10111 | 0 | n=1, 10 |
| 10111 | | | A → A−M |
| 11000 | 10111 | 0 | ASR |
| 11100 | 01011 | 1 | n=0 |

$(00011\,10101)_2 \rightarrow (+117)_{10}$

iv) −9 with −13

M: $(-9)_{10} \rightarrow (10111)_2 \xrightarrow{2's} (01001)_2$

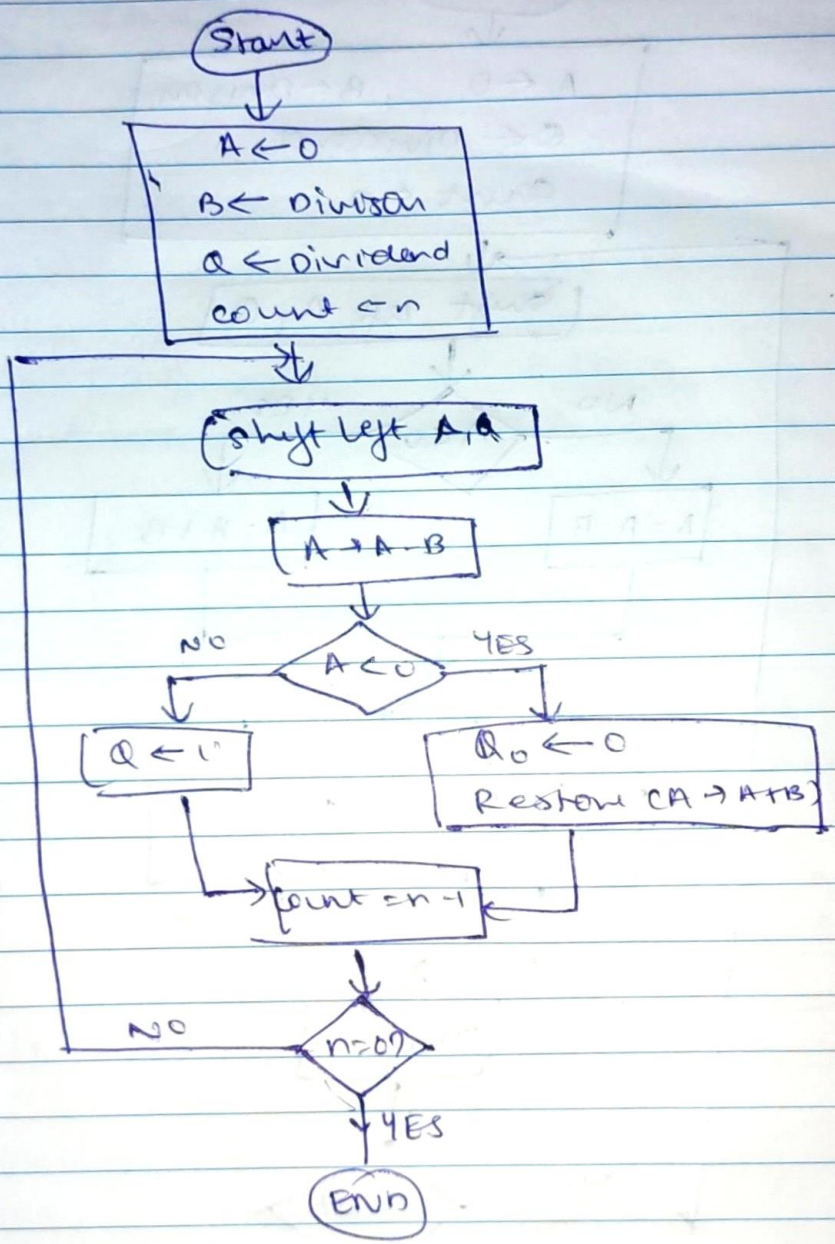
Q: $(-13)_{10} \rightarrow (10011)_2$

| A | Q | $Q_{-1}$ | Action |
|---|---|---|---|
| 00000 | 10011 | 0 | n=5, 10 |
| 01001 | | | A → A − M |
| 01001 | 10011 | 0 | ASR |
| 00100 | 11001 | 1 | n=4, 11, ASR |
| 00010 | 01100 | 1 | n=3, 01 |
| 10111 | | | A → A+M |
| 11001 | 01100 | 1 | ASR |
| 11100 | 10110 | 0 | n=2, 00, ASR |
| 11110 | 01011 | 0 | n=1, 10 |
| 01001 | | | A → A − M |
| 00111 | 01011 | 0 | ASR |
| 00011 | 10101 | 1 | n=0 |

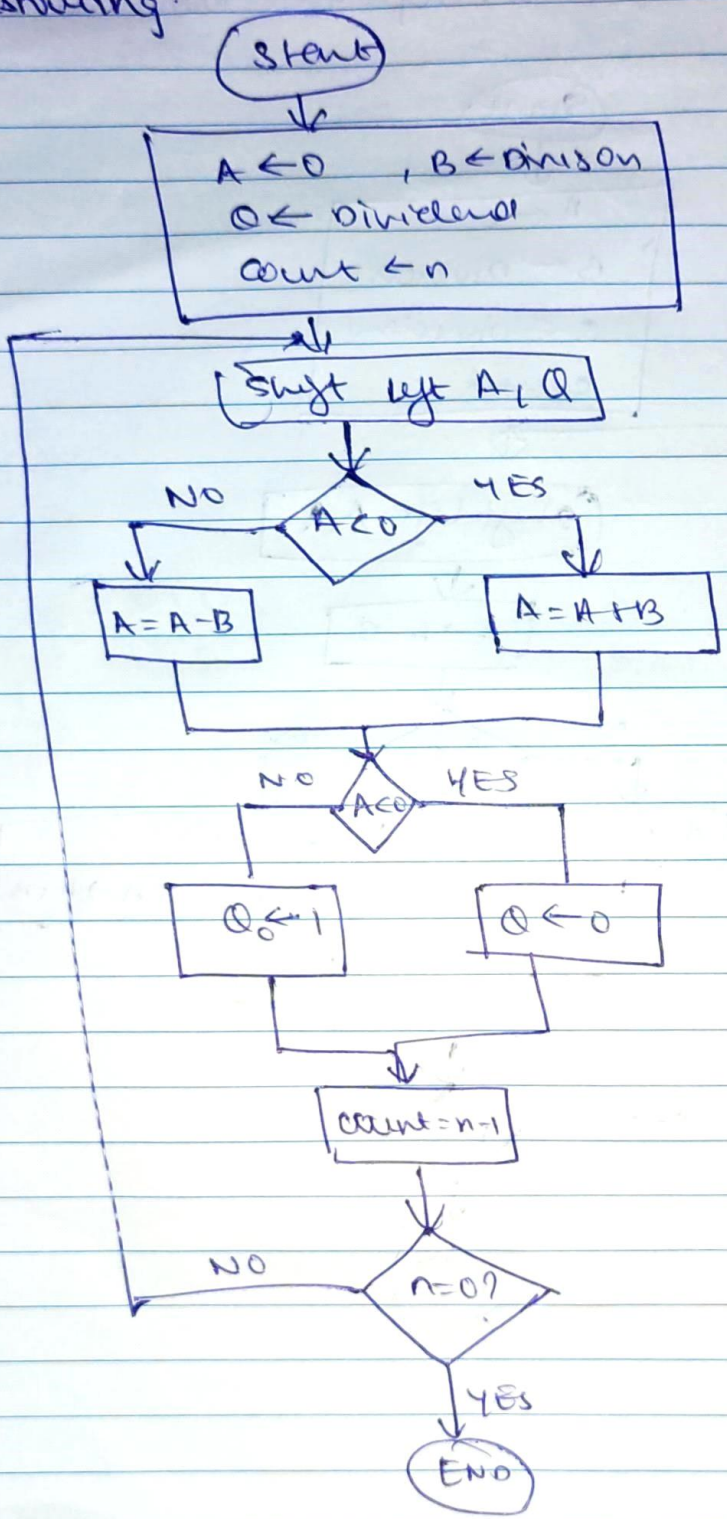
$(00011\,10101)_2 \rightarrow (117)_{10}$

Q] Draw Flowchart of Restoring and Non restoring

Restoring



Non Restoring



i) 13 divided by 9

$M = (01001)_2$

$Q = (1101)_2$

$n = 4$

| A | Q | |
|---|---|---|
| 00000 | 1101 | $n=4$, LS |
| 00001 | 101_ | $AC = AC - M$ |
| 10111 | | |
| 11000 | 1010 | $A<0$, $Q_0 = 0$, $n = 3$ |
| 10001 | 0103_ | $A<0$, $A = A+M$ |
| 01001 | | |
| 11010 | 0100 | $A<0$, $Q_0 = 0$, $n = 2$, LS |
| 10100 | 100_ | $A<0$, $A = A+M$ |
| +01001 | | |
| 11101 | 1000 | $A<0$, $Q_0 = 0$, $n = 1$, LS |
| 11011 | 000_ | $A<0$, $A = A+M$ |
| + 01001 | | |
| [1] 00100 | 0001 | $A>0$, $Q_0 = 1$, $n = 0$ |

↑ Quotient $= (01)_2 \rightarrow (1)_{10}$

↑ remainder $(100)_2 = (4)_{10}$

(ii) 13 divided by −9

$Q = (1101)_2 = (13)_{10}$

$M = (10111)_2 \overset{2s}{\leftarrow} (01001)_2 \leftarrow (9)_{10}$

$A+M = A + 10111$

$A-M = A + 01001$

| A | Q | Action |
|---|---|---|
| 00000 | 1101 | n=4, LS |
| 00001 | 101_ | A≥0, A=A-M |
| 01001 | | |
| 01010 | 1011 | A≥0 $Q_0$=1 |
| 10101 | | n=3, LS |
| +10111 | | A<0, A=A+M |
| 1 01100 | 0111 | A≥0, $Q_0$=1 |
| 11000 | 111_ | n=2, LS |
| 10111 | | A<0, A=A+M |
| 1 01111 | 1111 | A≥0, $Q_0$=1 |
| 11111 | 111_ | n=1, LS |
| +10111 | | A=A+M |
| 1 10110 | 1110 | A<0, $Q_0$=0 |

iii) -13 divided by 9

$Q = (0011)_2$

$M = (01001)_2$

| A | Q | Action |
|---|---|---|
| 00000 | 0111 | n=4, LS |
| 00000 | 011_ | A≥0, A=A-M |
| 10111 | | |
| 10 0111 | 0110 | A<0, $Q_0$=0 |
| 01110 | 110_ | n=3, LS |
| +10111 | | A≥0, A=A-M |
| 1 00101 | 1101 | A≥0, $Q_0$=1 |
| 01011 | 101_ | n=2, LS |
| 10111 | | A≥0, A=A-M |
| 1 00010 | 1011 | A≥0, $Q_0$=1 |

| | | |
|---|---|---|
| 00101 | 011 _ | n=1, LS |
| 10111 | | A≥0, A=A-M |
| 11100 | 0110 | A<0, $Q_0$=0 |
| | | n=0 |

iv) -13 divided by -9

Q = $(0011)_2$

M = $(10011)^{2s}$ → $(01300\,1)_2$

| A | Q | Action |
|---|---|---|
| 00000 | 0011 | n=4, LS |
| 00000 | 011 _ | A≥0, A=A-M |
| 01101 | | |
| 01101 | 0111 | A≥0, $Q_0$=1 |
| 11010 | 111 _ | n=3, LS |
| 10011 | | A=A+M |
| 101101 | 1111 | A≥0, $Q_0$=1 |
| 11011 | 111 _ | n=2, LS |
| 10011 | | A<0, A=A+M |
| 101110 | 1111 | A≥0, $Q_0$=1 |
| 11101 | 111 _ | n=1, LS |
| + 10011 | | A=A+M |
| 110000 | 1110 | A<0, $Q_0$=0 |
| | | n=0 |

Q3] 123.125

1111011.001

$1.1110011001 \times 2^6$

$(-1)^S (1.N) 2^{E-127} = (1.1110011001) \times 2^6$

$E - 127 = 6$

$E = 133$

$(133)_{10} = (10000101)_2$

| 1 | 10000101 | 11110011001...00 | $-123.125$ |
|---|---|---|---|

| 0 | 10000101 | 11110011001...00 | $123.125$ |
|---|---|---|---|

$(1.N) 2^{E-1023} = (1.1110011001) \times 2^6$

$E = 1029 = (10000000101)_2$

| 0 | 10000000101 | 11110011001...00 | $+123.125$ |
|---|---|---|---|

| 1 | 10000000101 | 11110011001...00 | $-123.125$ |
|---|---|---|---|

(ii) 0.125 and -0.125

$0.125 = (0.001)_2 = 1.0 \times 2^{-3}$

$(-1)^S \times N \times 2^{E-127} = 1.0 \times 2^{-3}$

$E - 127 = -3$

$E = 124$

$= (11110\ 11)_2$

| 0/1 | 01111011 | 0.....00 |
|---|---|---|

64 bit $1.N \times 2^{E-1023} = 10 \times 2^{-3}$

$E = 1020$

$= 011\ 1111\ 1100_2$

| 0/1 | 011111 1100 | 0 .... 000 |
|---|---|---|

(iii) 111.22 and -111.22

011 0111.001

32 bit

$1.1011001 \times 2^6$

$1.N \times 2^{E-127}$

$E = 132$

$= 1000\ 0100$

| 0/1 | 1000 0100 | 1011001 .. 00 |
|---|---|---|

64 bit

$E = 1028$

$= 100\ 0000\ 0000$

| 0/1 | 10000000000 | 1011001 ... 00 |
|---|---|---|

(iv) 1259.125

0 1001110 1011.001

32 bit

$E = 127 + 10 = 137$

| 0/1 | 100 0 1001 | 00 1110 111 .. 00 |
|---|---|---|

S23 - 127

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

E = 1033

| 0 | 1000000 1001 | 00111 01011 .. 00 |
|---|---|---|