VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering
Fall Semester 2024-2025

Student Name: DHRUV AGRAWAL Reg No: 22BCT0059 Email: dhruv.agrawal2022a@vitstudent.ac.in	
Faculty Name: **DHRUV AGRAWAL**	
Subject Code with Name: **BCSE204P – Micro-Controller and Micro-Processor**	
Assessment No.:	01
Date of Submission:	09/02/2025

1.1

↳ In lab Mold

① Create new folder.

② µ vision

③ project -> new µVision Project
select folder -> file name
] creation of Project

Legacy - Device.
P89V51RD2
New file.

④ Program

```
ORG 0000H   — memory from where it will start
MOV A, #05H
MOV R0, #05H
ADD A, R0
END
```

means 05 data else address
is used only in micro controller

⑤ Save
- .asm extension

⑥ Source Group
add existing file
build - to check errors.
debug.

⑦ In micro controller - by default stack Pointer (SP) value is 07.
PSW - to check status of result (even Parity / odd Parity)
(Program status window)

⑧ can execute stepwise or complete

Continue 1.1a

Aim: To write, perform, execute assembly language program for 8051 microcontroller for addition of 2 8-bit numbers and store result in accumulator (A)

Software used: KIEL µVISION 6.0

Program:

```
ORG   0000H          ; Set the Program starting address
                       to 0000H (hexadecimal)

MOV   A, #05H        ; Move the immediate value 05H
                       into the accumulator

MOV   R0, #05H       ; Move the immediate value 05H
                       into the register R0

ADD   A, R0          ; Add the value in R0 to
                       accumulator

H: SJMP   H          ; Create an infinite loop to halt
                       the program

END                  ; END of the program
```

Expected Output (Theoretical): 05H in binary

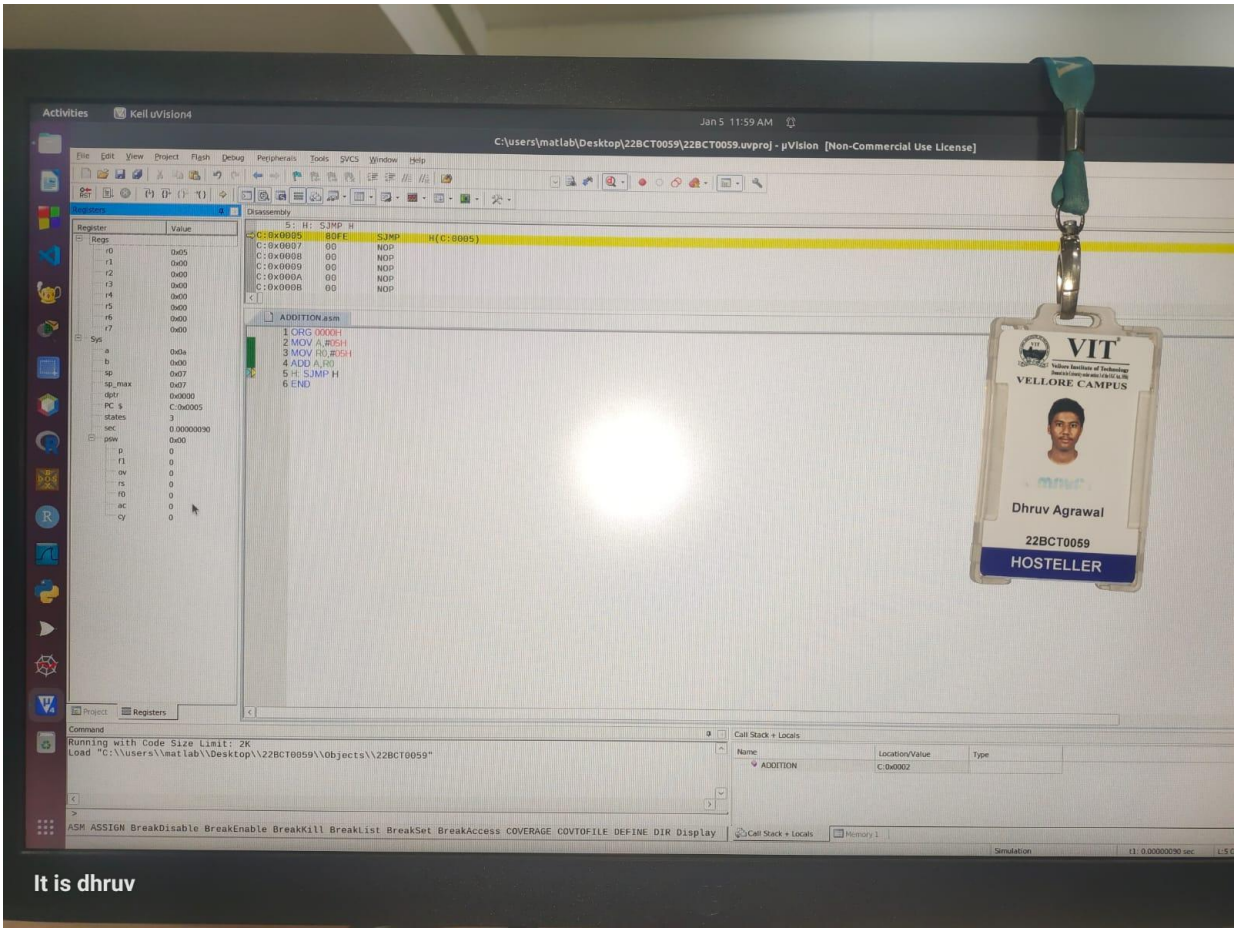
```
  0000 0101
+ 0000 0101
-----------
  0000 1010   -> in hexadecimal 0AH
```

Actual/Practical Output => 0AH

Carry bit = 0
AC = 0
OV = CY ⊕ CY'
= 0 ⊕ 0 = 0
P (Parity) = 0
Flag

∴ PSW content (8-bit)
00000000

Lab Screenshot (with PSW):



1.1 b

Aim: To write, perform, execute assembly language program for 8051 microcontroller for sub of 2 8-bit numbers and store result in accumulator.

Software used : KIEL uVISION6.0

Program:

```
ORG   0000H      ; set the Program starting address to 0000H
MOV   A, #0AH    ; Move the immediate value 0AH into the accumulator
MOV   R0, #05H   ; Move the immediate value 05H into the register R0
SUBB  A, R0      ; Subtract the value in R0 From accumulator with borrow
H: SJMP H        ; Create an infinite loop to halt the program
END              ; END the program
```

Expected Output (Theoretical) :

0AH → 0000 1010 - binary
05H → 0000 0101 - binary
0000 0101

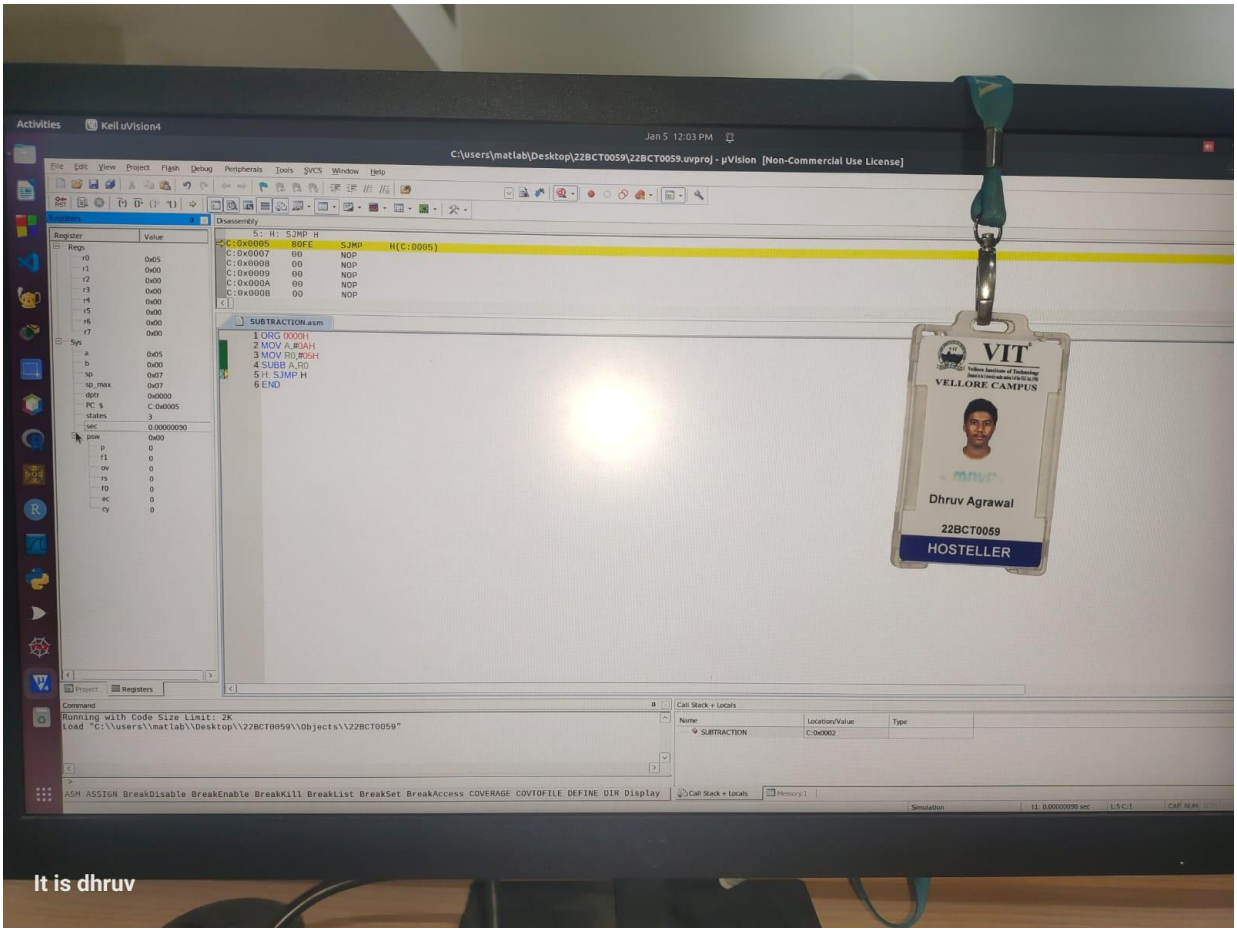
Converting to hexadecimal → 05H //

PSW content (8-bit) → CY = 0, AC = 0, OV = 0, P (Parity Flag) = 0

00000000

Actual / Practical Output = 05H

Lab Screenshot (with PSW):



1.1 C

Aim: To write, perform, execute the assembly language program for 8051 microcontroller for multiplication of 2- 8bit numbers and store result in Accumulator

Software used: KIEL µVISION 6.0

Program:

```
ORG  0000H      ; Set the program starting address to 0000H (hexadecimal)
MOV  A, #05H    ; Move the immediate value 05H into the accumulator.
MOV  B, #05H    ; move the immediate value 05H into the register B.
MUL  AB         ; Multiply the value in A by B.
H: SJMP H       ; create an infinite loop to halt the program.
END             ; END of the program
```

Expected Output (Theoretical):

05H → 0000 0101

05H → × 0000 0101

```
  0000 0101
 0 0000 000
 0 0000 101
 0 0000 000
-----------
 0001 1001
```

5 × 5 = 25

0001 1001

16 + 8 + 1

25

Converting to hex → 19H

PSW content → 0000 0001 (8-bit)

CY = 0

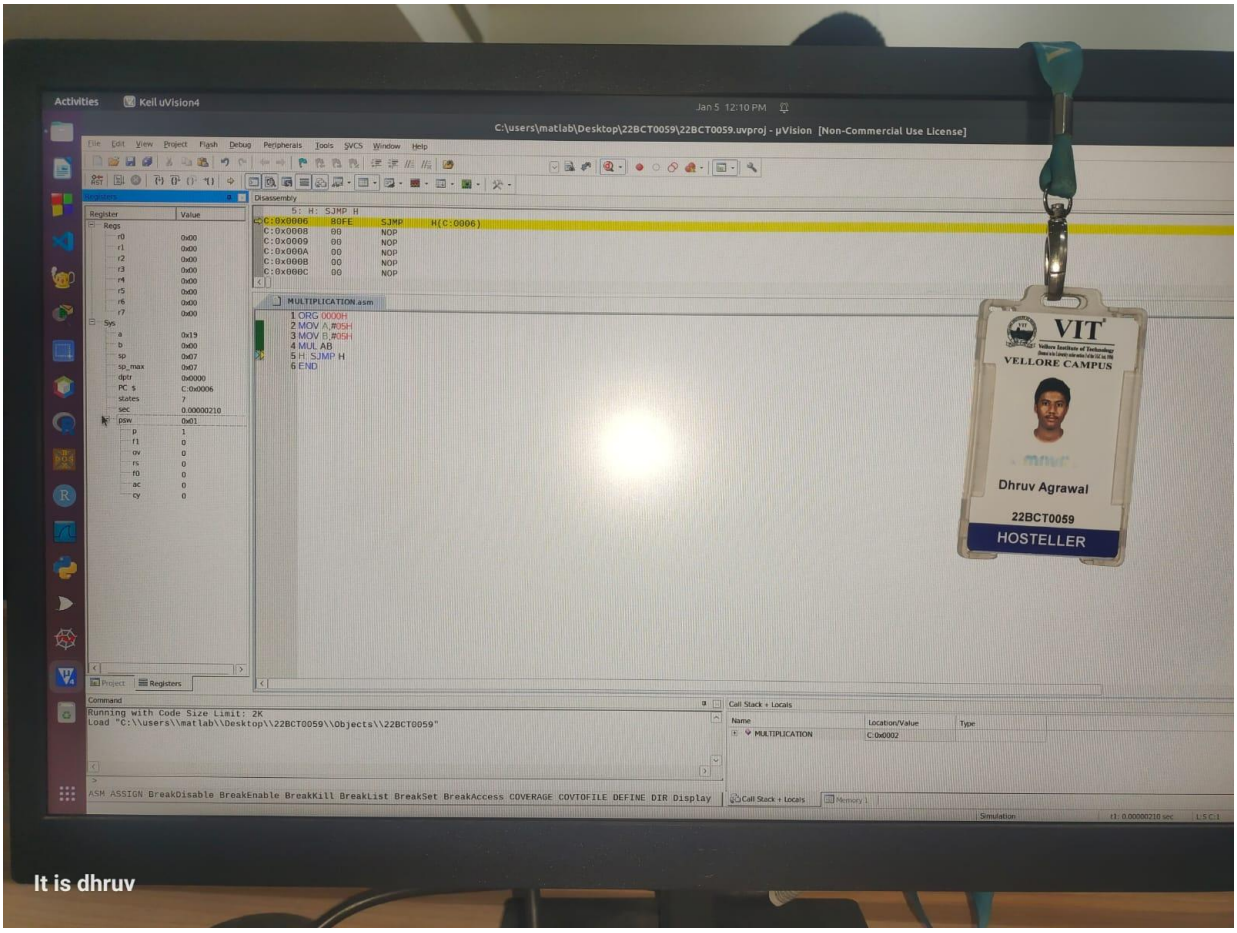
AC = 0

OV = 0 ⊕ 0 = 0

Parity flag (P) = 1

Actual / Practical = 19H

Lab Screenshot (with PSW):



1.1 D

Aim: To write, perform, execute the assembly language program for 8051 micro controller for division of 2 8-bit numbers and store result in accumulator

Software used : KIEL µVISION 6.0

Program:

```
ORG 0000H      ; set the program starting address to 0000H (hexadecimal)
MOV A, #20H    ; Move the immediate value 20H into the accumulator
MOV B, #05H    ; Move the immediate value 05H into the register B.
DIV AB         ; Divide the value in A from B and store result in A
H: SJMP H      ; create an infinite loop to halt the program
END            ; END of the program
```

Expected Output (Theoretical) :-

20H → 0010 0000 — binary
05H → 0000 0101 — binary

```
      0110  → Quotient
0101 | 100000
     - 0↓
       1000
     - 101↓
       0110
     - 101↓
        0010
      - 0
        0010 → Rem
```

Converting to HEX

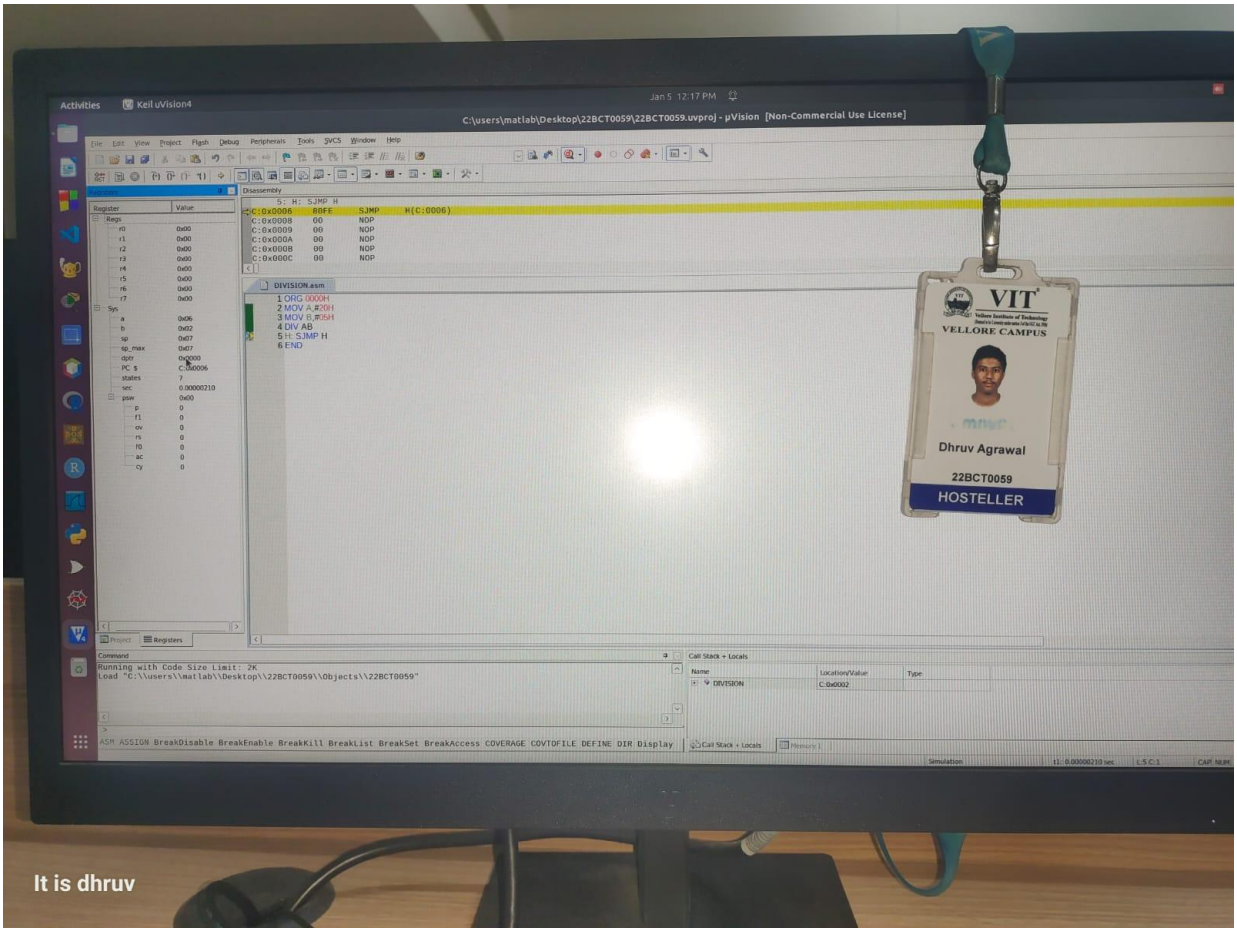
0110 → Q = 06H — stored in A
0010 → R = 02H — stored in B

PSW content (8-bit) → 00000000

CY = 0
AC = 0
OV = 0 ⊕ 0 = 0
Parity (P) = 0
Flag

Actual/Practical = 06H

Lab Screenshot (with PSW):



1.2

Aim: Write and assemble a program to add the following data and then use the simulator to examine the CY flag

92H, 23H, 66H, 87H, F5H

Software used: KEIL µVISION 6.0

Program:

```
        ORG 000H        ; set the program starting
                          address to 0000H (hexadecimal)
        MOV A,#92H      ; Move immediate value 92H
                          into the accumulator
        MOV R0,#23H     ; Move immediate value 23H
                          into the register R0
        ADD A,R0        ; Add the value of R0 into A
        JNC L1          ; Jump to L1 if no carry genrated
        INC R7          ; Increment R7
L1:     MOV R1,#66H     ; Move the immediate value 66H
                          into the register R1
        ADD A,R1        ; Add the value of R1 into A
        JNC L2          ; Jump to L2 if no carry genrated
        INC R7          ; Increment R7
L2:     MOV R2,#87H     ; Move the immediate value of 87H
                          into the register R2
        ADD A,R2        ; Add the value of R2 into A
        JNC L3          ; Jump to L3, if no carry genrated
        INC R7          ; Increment R7
L3:     MOV R3,#0F5H    ; Move the immediate value of F5H
                          into the register R3
        ADD A,R3        ; Add the value of R3 into A
        JNC L4          ; Jump to L4 if no carry genrated
        INC R7          ; Increment R7
L4:     END             ; End of the program
```

Expected output :- (Theoretical)

92H → 10010010
+ 23H → 00100011
= 10110101 → J- binary

no carry genrated ∴ R7 will remain 00H

10110101
+ 01100110 ⟶ 66H
= 1 00011011

carry genrated.
∴ Increment R7.
∴ R7 = 0+1 = 01H

00011011
+ 10000111 — 87H
= 10100010

no carry genrated ∴ R7 will remain 01H

10100010
+ 11110101 — F5H
= 1 10010111 → converting to hexadecimal

Carry genrated. Ans 97H

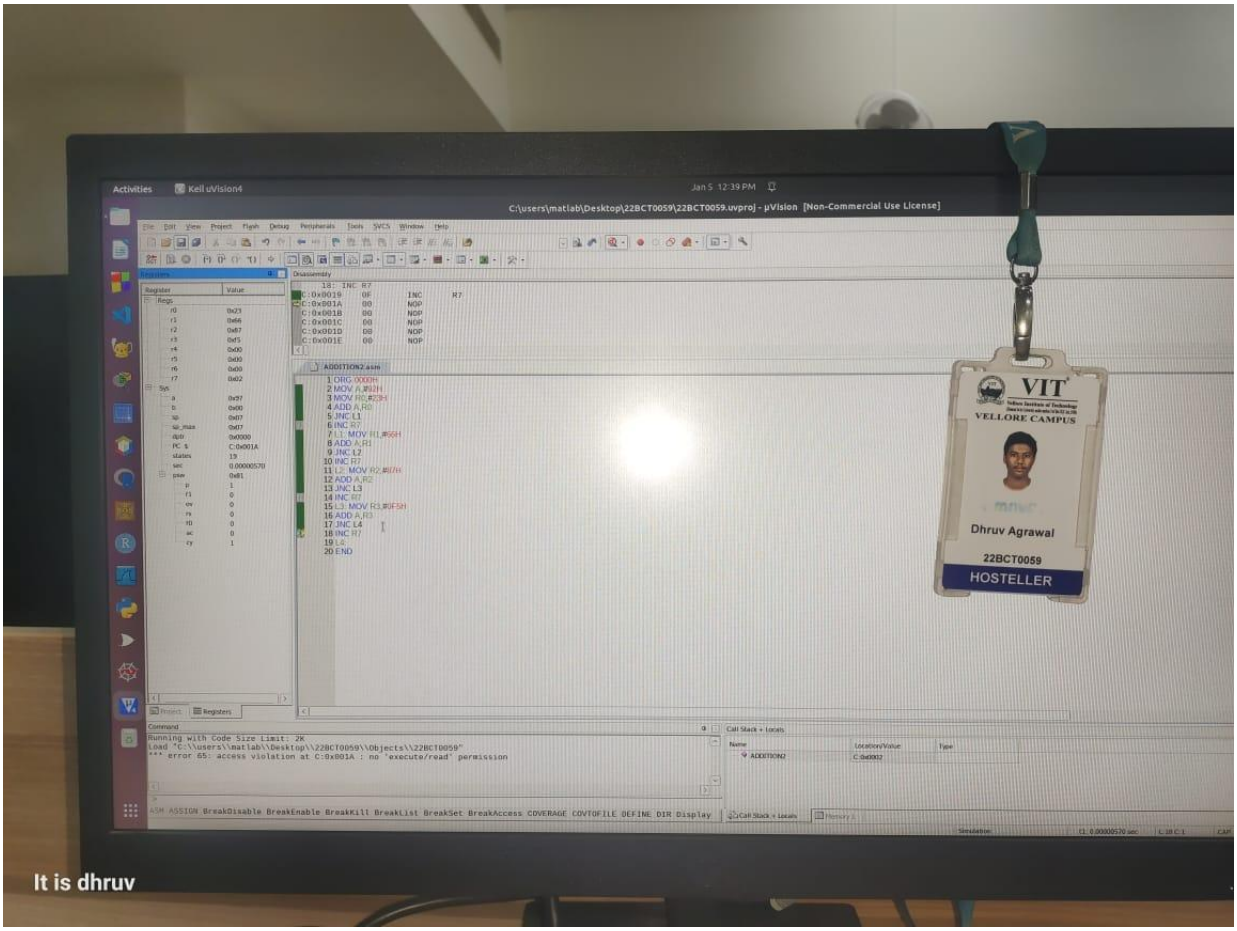
∴ Increment R7
R7 = 01H + 1 = 02H

Practical
Actual output = 02H

PSW content (8-bit) 10000001 → CY = 1
AC = 0
OV = 1 ⊕ 1 = 0
Parity (P) = 1
Flag

Conclusion : The operations addition, subtraction, multiplication division, addn with carry are important arithmetic operations. to perform complex mathematical operation. verifying, Executing these functions is neccesary for developing efficient and accurate embedded/micro controller based systems.

Lab Screenshot (with PSW):



1.3 A

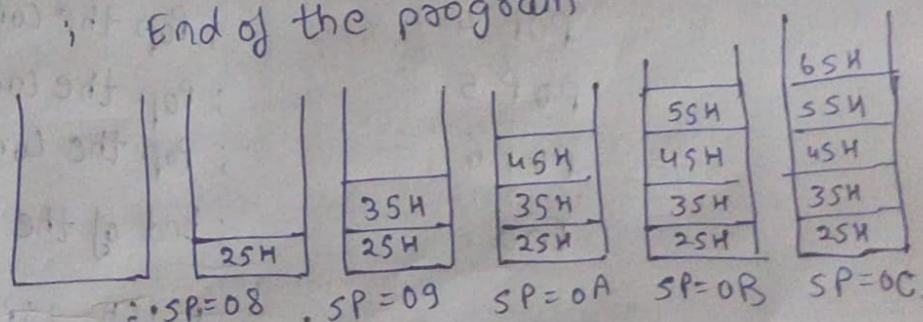
Aim: To write, perform and execute the assembly language program to load value into register R0 to R4 and then push it into stack. Examine the stack register values after execution of Program

Software used : Keil µVision 6.0

Program:

```
ORG 0000H ; Program start address (origin)
mov R0, #25H ; Load value 25H in R0
mov R1, #35H ; Load value 35H in R1
mov R2, #45H ; Load value 45H in R2
mov R3, #55H ; Load value 55H in R3
mov R4, #65H ; Load value 65H in R4
PUSH 0 ; Push value of R0 in stack
PUSH 1 ; Push value of R1 in stack
PUSH 2 ; Push value of R2 in stack
PUSH 3 ; Push value of R3 in stack
PUSH 4 ; Push value of R4 in stack
END ; End of the program
```

Expected Output:
Manual Calculation

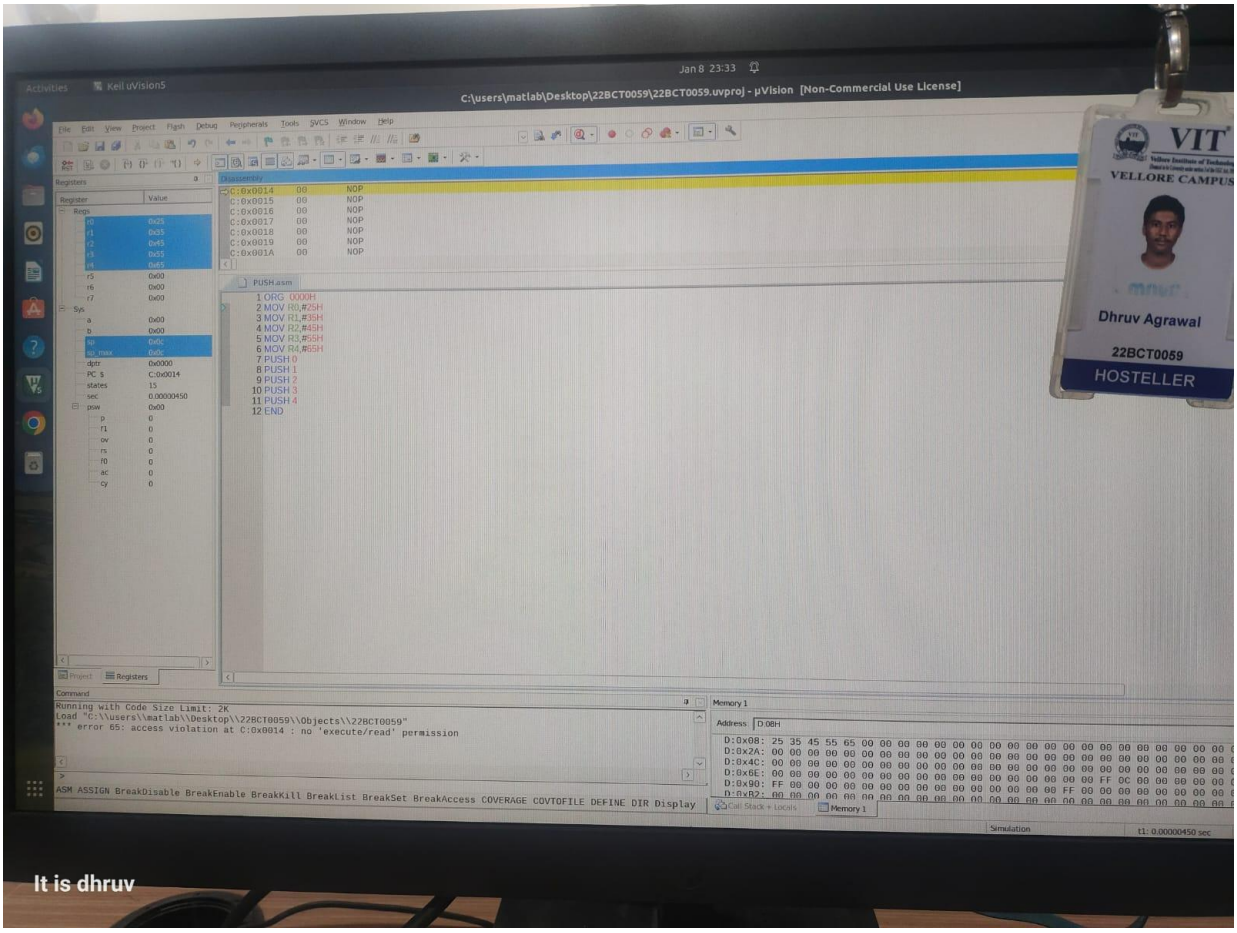


Final stack Pointer value is 0CH

: 0CH

Practical/ Actual Output

Lab Screenshot (with PSW):



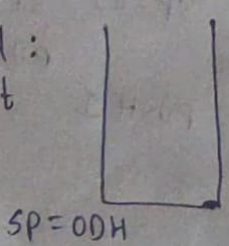
Aim: Write, perform and execute the assembly lang program
1.3B to set SP = 0D. Then put different value in each register from 08 to 0D, POP SP into R0 - R4

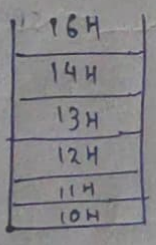
Software used: Keil uVision 6.0

Program:

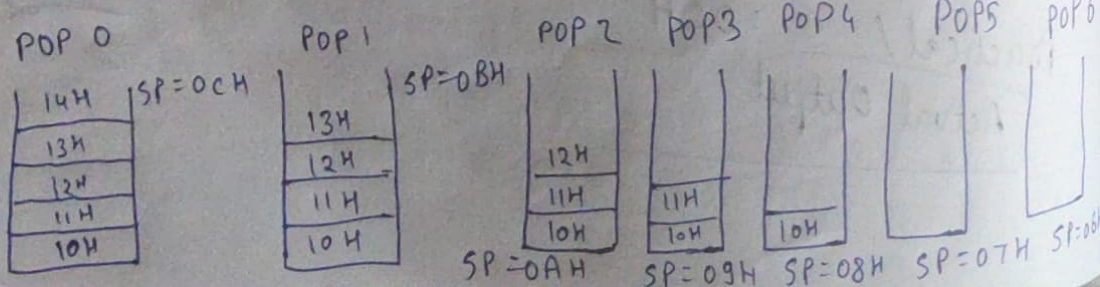
```
ORG 0000H      ;Program start address Coriging
MOV SP, #0DH   ;Move SP value to 0DH
MOV 08H, #10H  ;Move 10H in memory register location 08H
MOV 09H, #11H  ;Move 11H in register 09H
MOV 0AH, #12H  ;Move 12H in register 0AH
MOV 0BH, #13H  ;Move 13H in register 0BH
MOV 0CH, #14H  ;Move 14H in register 0CH
MOV 0DH, #16H  ;Move 16H in register 0DH
POP 0          ;Pop the content of stack in R0
POP 1          ;Pop the content of stack in R1
POP 2          ;Pop the content of stack in R2
POP 3          ;Pop the content of stack in R3
POP 4          ;Pop the content of stack in R4
POP 5          ;Pop the content of stack in R5
POP 6          ;Pop the content of stack in R6
End            ;End of the program
```

manual / Expected : Calculation output

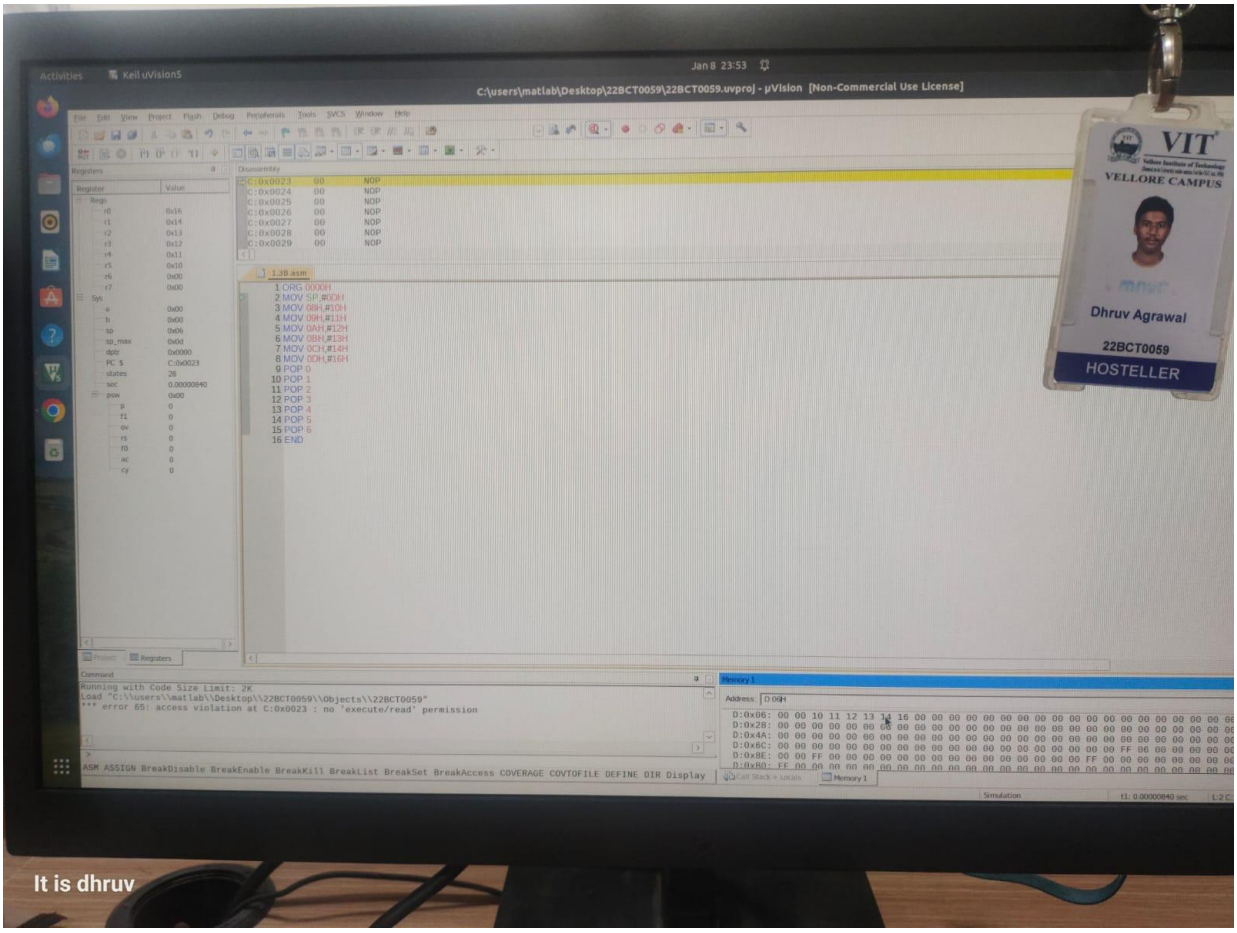




Practical output = 06H



Lab Screenshot (with PSW):



1.3.C.

Aim: Write, perform and execute assembly language program to load values into registers R0 to R4, then push it into stack, pop them back

Software used: Keil µVision 6.0

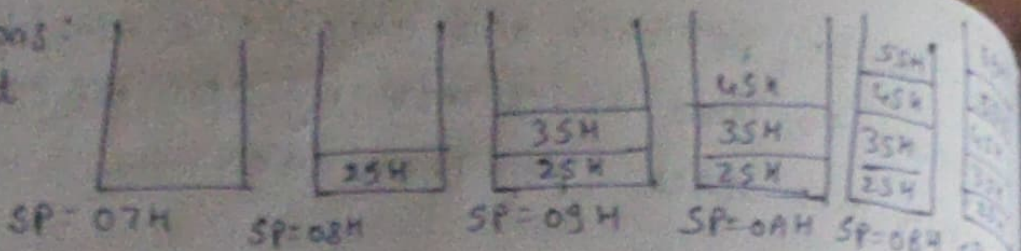
Program:

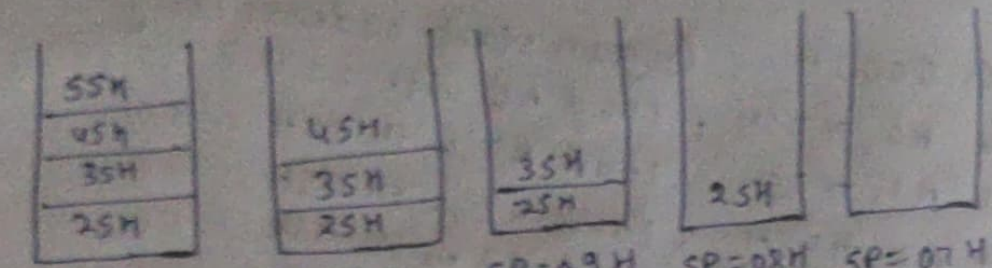
```
ORG 0000H      ; Program start address (origin)
MOV R0, #25H   ; Load R0 with 25H
MOV R1, #35H   ; Load R1 with 35H
MOV R2, #45H   ; Load R2 with 45H
MOV R3, #55H   ; Load R3 with 55H
MOV R4, #65H   ; Load R4 with 65H
PUSH 0         ; Push the content of R0 into stack
PUSH 1         ; Push the content of R1 into stack
PUSH 2         ; Push the content of R2 into stack
PUSH 3         ; Push the content of R3 into stack
PUSH 4         ; Push the content of R4 into stack
POP 10H        ; Pop the content of stack in 10H
POP 11H        ; Pop the content of stack in 11H
POP 12H        ; Pop the content of stack in 12H
POP 13H        ; Pop the content of stack in 13H
POP 14H        ; Pop the content of stack in 14H
END            ; End of the program
```

Practical = 07H

Output

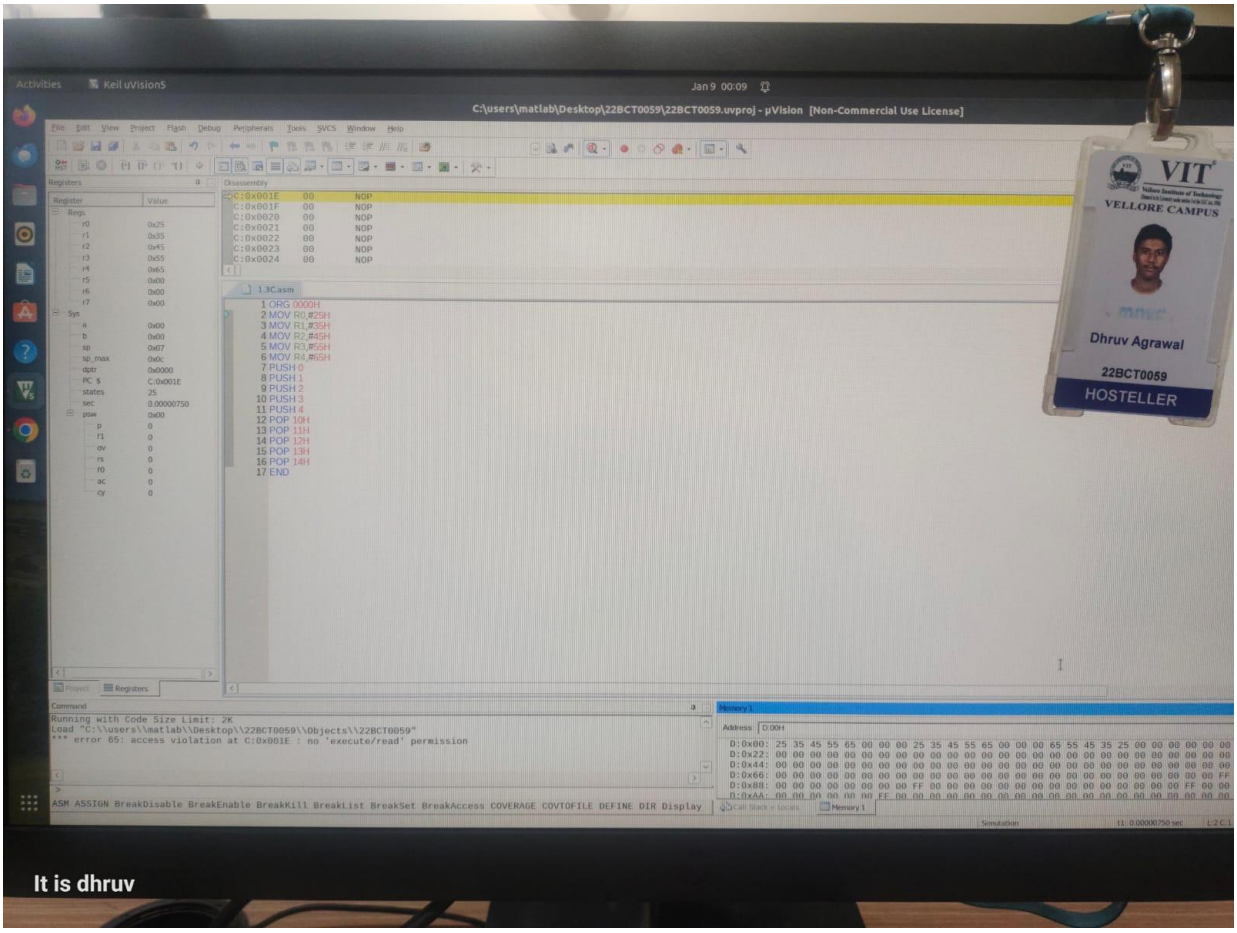
manual Calculations:
Expected Output





Conclusion/ Result: The Push and Pop operation are fundamental for stack management with Push adding data to stack from register and Pop, moving data from stack to register. Both are efficient in memory management and handling fn call, interrupt The entire process is executed on Keil µVision and answers/output is verified.

Lab Screenshot (with PSW):



1.4 A

Aim: Write, perform and execute assembly language program to transfer string of data from code space, starting at address 200H to RAM locations starting at 40H. data is

0200H: DB "VIT UNIVERSITY"

Software used: Keil µVision 6.0

Program:

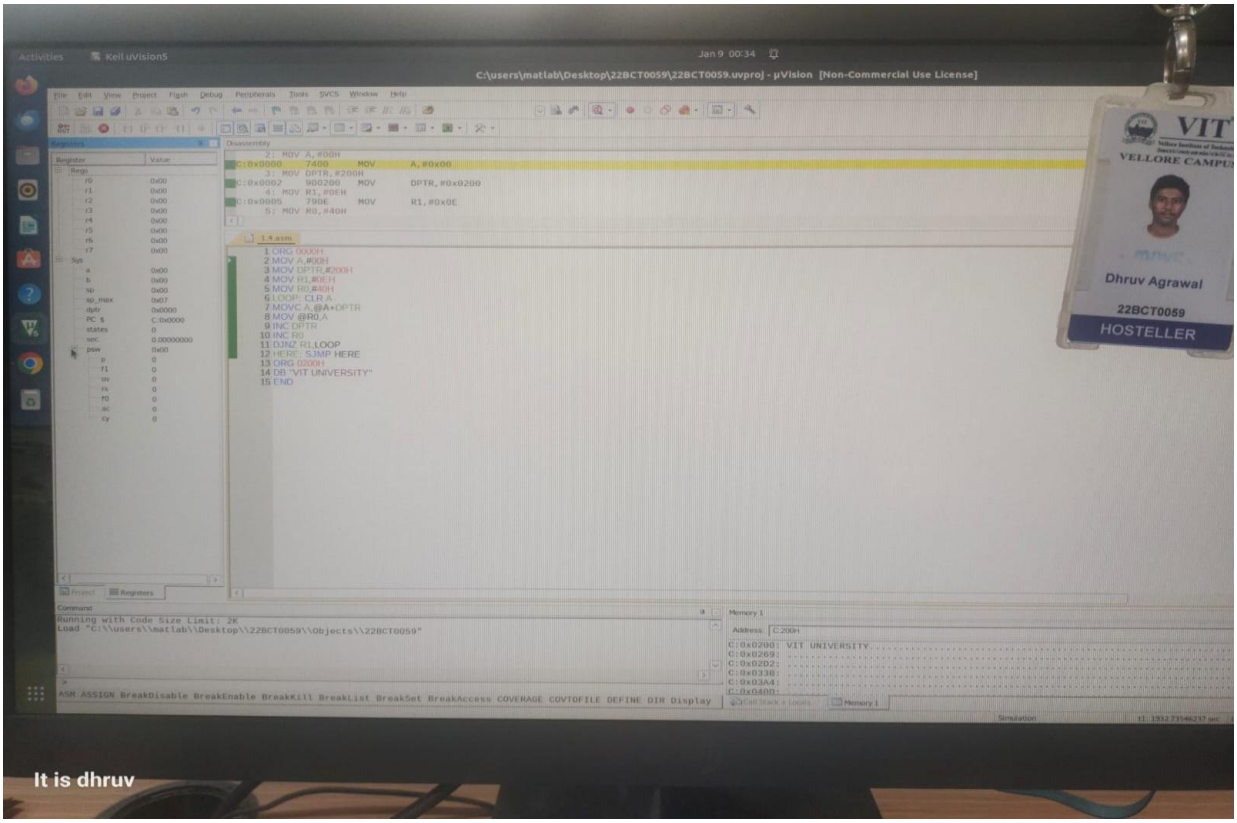
```
        ORG 0000H ; Program start address
        MOV A,#00H ; set value 00 in A /clear Accumulator
        MOV DPTR,#200H ; load DPTR with code memory add
        MOV R1, #0EH ; Load R1 with 0EH
        MOV R0, #40H ; move value 40H in R0
LOOP:   CLR A ; clear the accumulator
        MOVC A,@A+DPTR ; Read byte from code memory into A
        MOV @R0, A ; Write byte from A into RAM
        INC DPTR ; Increment DPTR to next code memory location
        INC R0 ; Increment R0 to next RAM location
        DJNZ R1, loop ; Decrement R1 and jump to loop until R1≠0
HERE:   SJMP HERE ; Infinite loop.
        ORG 0200H ; Code memory location where string is stored
        DB: "VIT UNIVERSITY" ; string to be copied
        END ; End of Program.
```

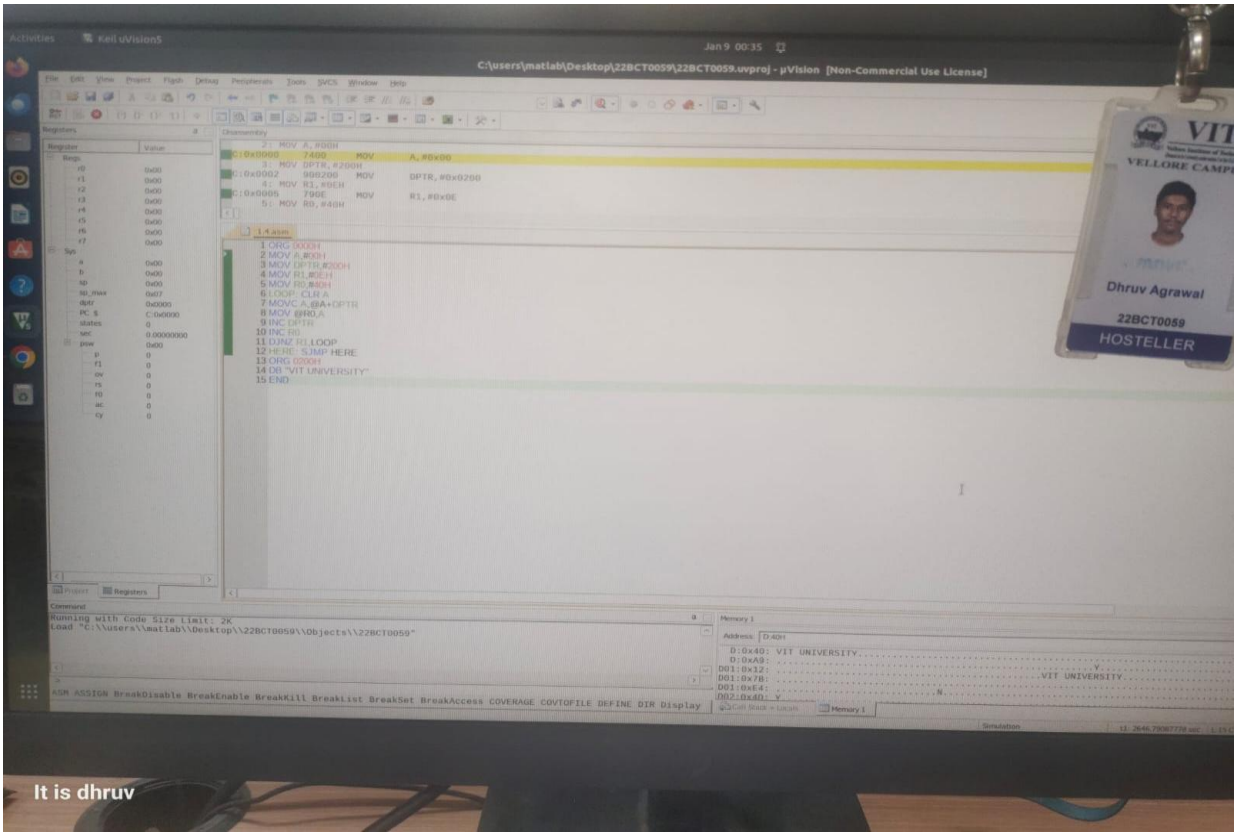
Expected Output:

D: 0x40 : VIT UNIVERSITY — RAM SPACE

Practical Output = VIT UNIVERSITY

Lab Screenshot (with PSW):





1.4B

Aim: To write, perform and execute assembly language program to transfer string of data from code space to RAM space in reverse order

0200H: MYDATA "VIT UNIVERSITY"

Software used: Keil uVision 6.0

Program:

```
        ORG 0000H              ; Program start address
        MOV DPTR, #0200H       ; Load DPTR with Code Memory add
        MOV R1, #14H           ; Load R1 with 14H
        MOV R0, #40H           ; Load R0 with 40H
LOOP:   CLR A                  ; clear Accumulator
        MOVC A, @A+DPTR        ; Read byte from Code memory into A
        MOV @R0, A             ; write byte from A into RAM
        DEC DPL                ; Decrement DPTR by 1
        INC R0                 ; Increment R0 to next RAM locatio
        DJNZ R1, LOOP          ; Decrement R1 and jump to loop if no
HERE:   SJMP HERE              ; Infinite loop

        ORG 0200H              ; Code memory location where string is stored
        DB "VIT UNIVERSITY"    ; String to be copied
        END                    ; End of the program
```

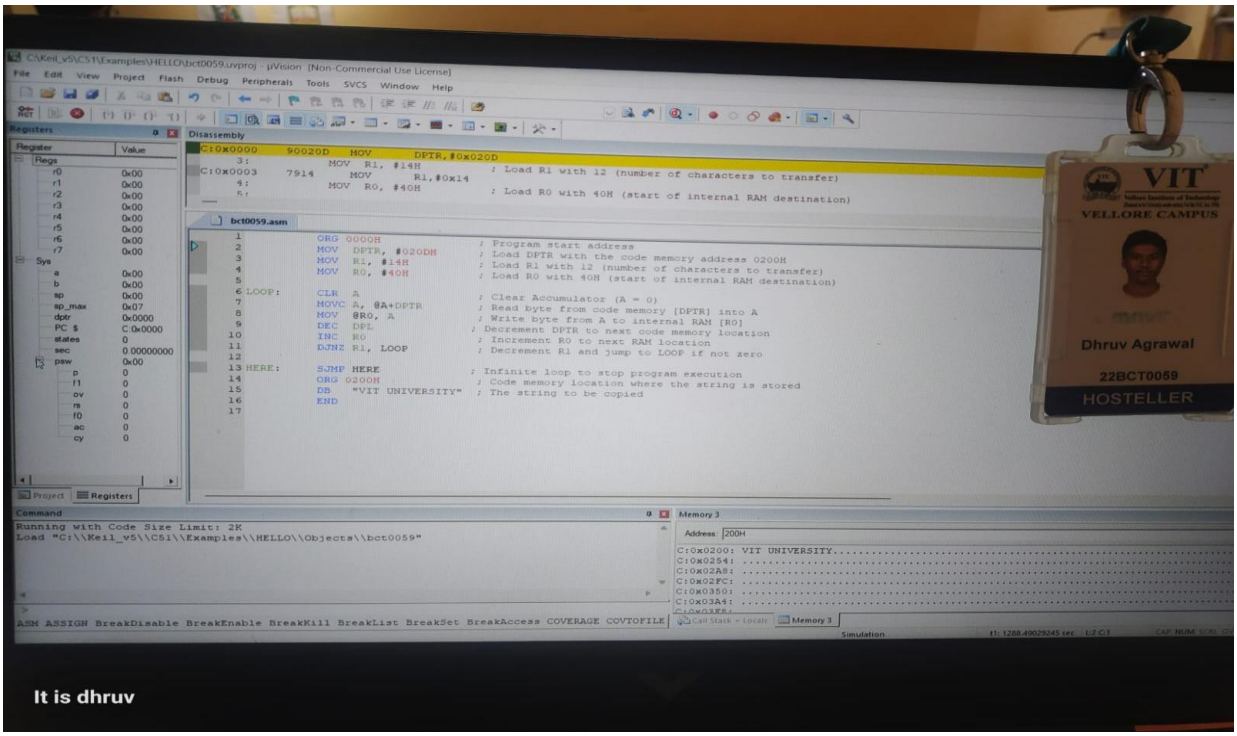
manual / Expected Calculation Output:

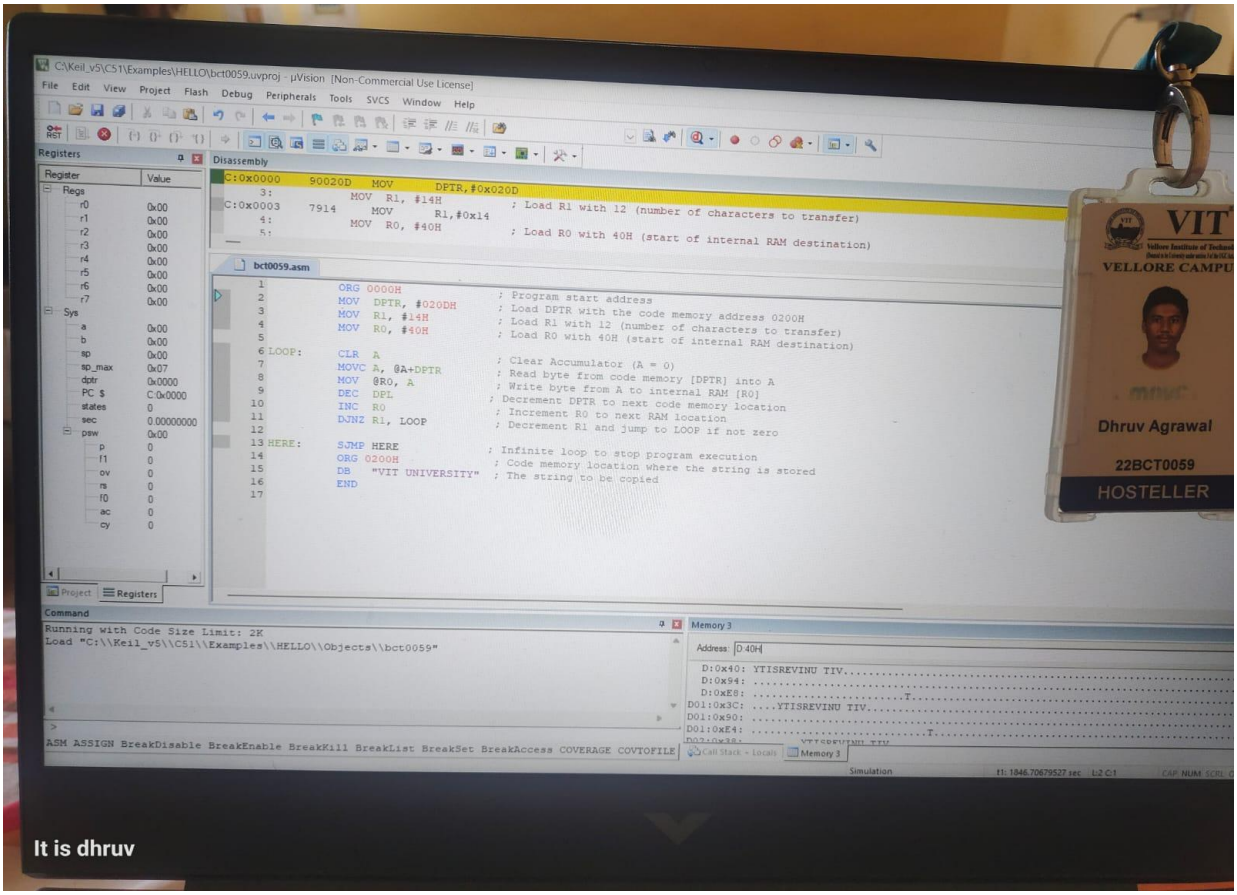
0200H : "VIT UNIVERSITY" — Code memory

0:0X40H: YTISREVINU TIV — RAM, space

PRACTICAL OUTPUT: YTISREVINU TIV

Lab Screenshot (with PSW):





1-5A

Aim: To write, perform and execute assembly language program to add 10 bytes of data and store result in register R2 and R3. Bytes are stored in RAM space 0200H

0200H: MYDATA DB: 92, 34, 84, 129, - - -

Software used: Keil µVision 6.0

Program:

```
        ORG 0000H       ; Program start address.
        MOV DPTR, #200H ; Load DPTR with code memory address
        MOV R0, #10H    ; Load R1 with 10H
LOOP:   CLR A           ; clear accumulator
        MOVC A, @A+DPTR ; Read byte from code memory into A
        ADD A, R2       ; Add value of R2 into A
        JNC NEXT        ; Jump to next location if C ≠ 1
        INC R3          ; Increment R3 by 1
NEXT:   INC DPTR        ; Increment DPTR to next code memory location
        MOV R2, A       ; Load value of A in R2
        DJNZ R0, LOOP   ; Decrement R0 and jump to loop if R0 ≠ 0
HERE:   SJMP HERE       ; Infinite loop
ORG: 200H               ; code memory location where string is stored
        DB 22H, 43H, 23H, 34H, 31H, 77H, 91H, 33H, 43H, 7H
        END             ; end of program
```

↳ Data to be added.

Expected output:

Practical = 272H

manual Calculation

22H + 43H + 23H + 34H + 31H + 77H + 91H + 33H + 43H + 7H

= 272H //

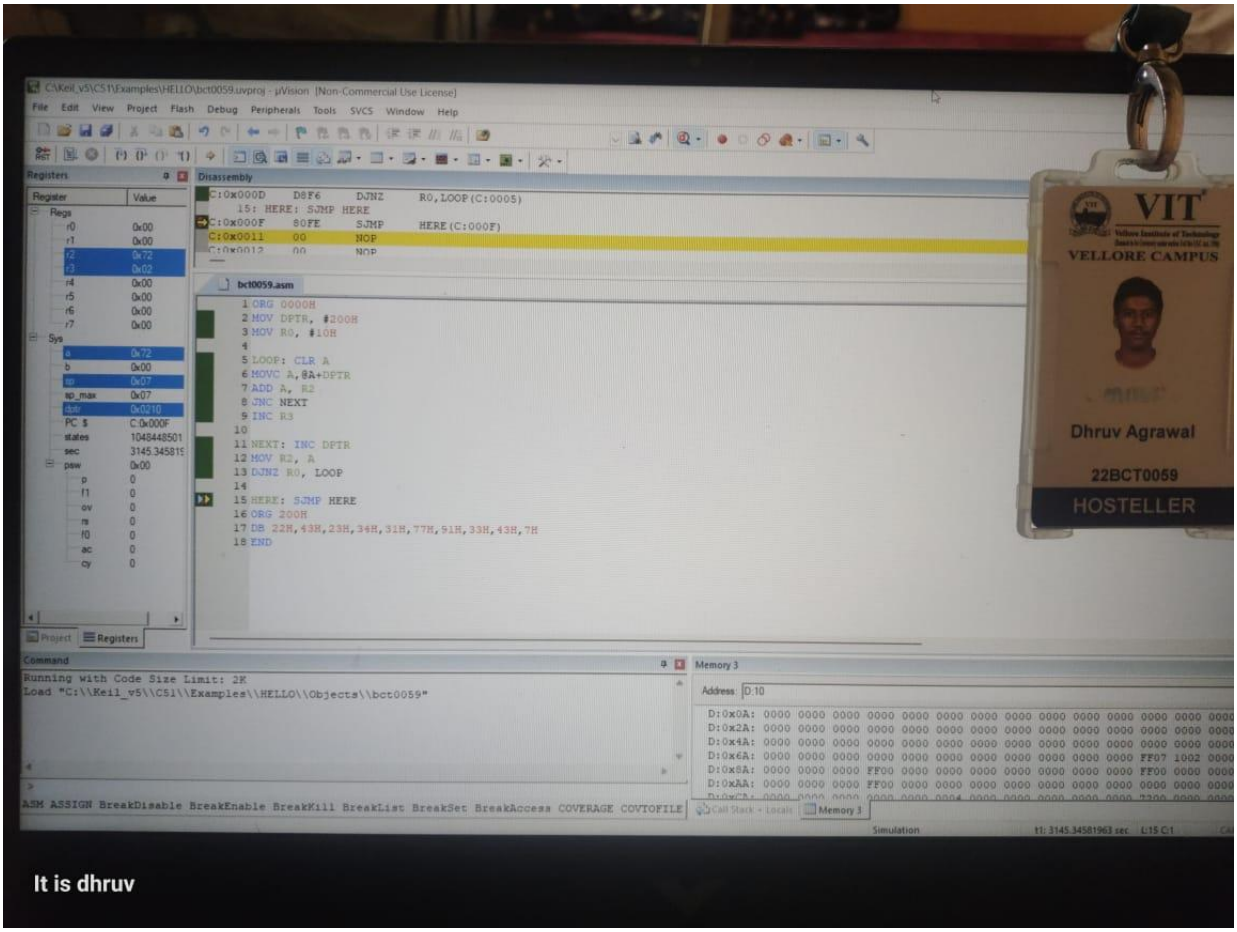
∴ 02 (R3) 72 (R2) / Ans.

= 1001110010 //

∴ Parity bit = 1 //

∴ PSW = 0x01 //

Lab Screenshot (with PSW):



1.5B

Aim: Write a Program to add 10 Bytes of BCD data and store result in R2 and R3. Bytes are stored in ROM space at 300H

MYDATA: DB 92H, 34H, 84H, 23H ...

Software used: Keil µVision 6.0

Program:

```
        ORG 0000H      ; Program start Address (origin)
        MOV DPTR, #300H ; Load DPTR with code memory add
        MOV R0, #10H   ; Load R0 with 10H
LOOP:   CLR A          ; clear the accumulator.
        MOVC A, @A+DPTR ; Read byte from code memory into A
        ADD A, R2      ; Add value of R2 into A
        DA A           ; Decimal adjust content of accumulator
        JNC NEXT       ; Jump to NEXT if carry ≠ 1
        INC R3         ; Increment R3 to next RAM location
NEXT:   INC DPTR       ; Increment DPTR by 1
        MOV R2, A      ; Load value of Accumulator in R2
        DJNZ R0, LOOP  ; Decrement R0 and jump to LOOP
                       until R0 ≠ 0
HERE:   SJMP HERE      ; Infinite Loop
        ORG 300H       ; code memory location
        DB 22H, 43H, 23H, 34H, 31H, 77H, 91H, 33H, 43H, 7H
        END            ; End of Program & Data in Code memory
```

manual Calculation / Expected Output : ✱ Note in BCD only 0 to 9 :

Practical = 404H

22H + 43H + 23H + 34H + 31H + 77H + 91H + 33H + 43 + 7H

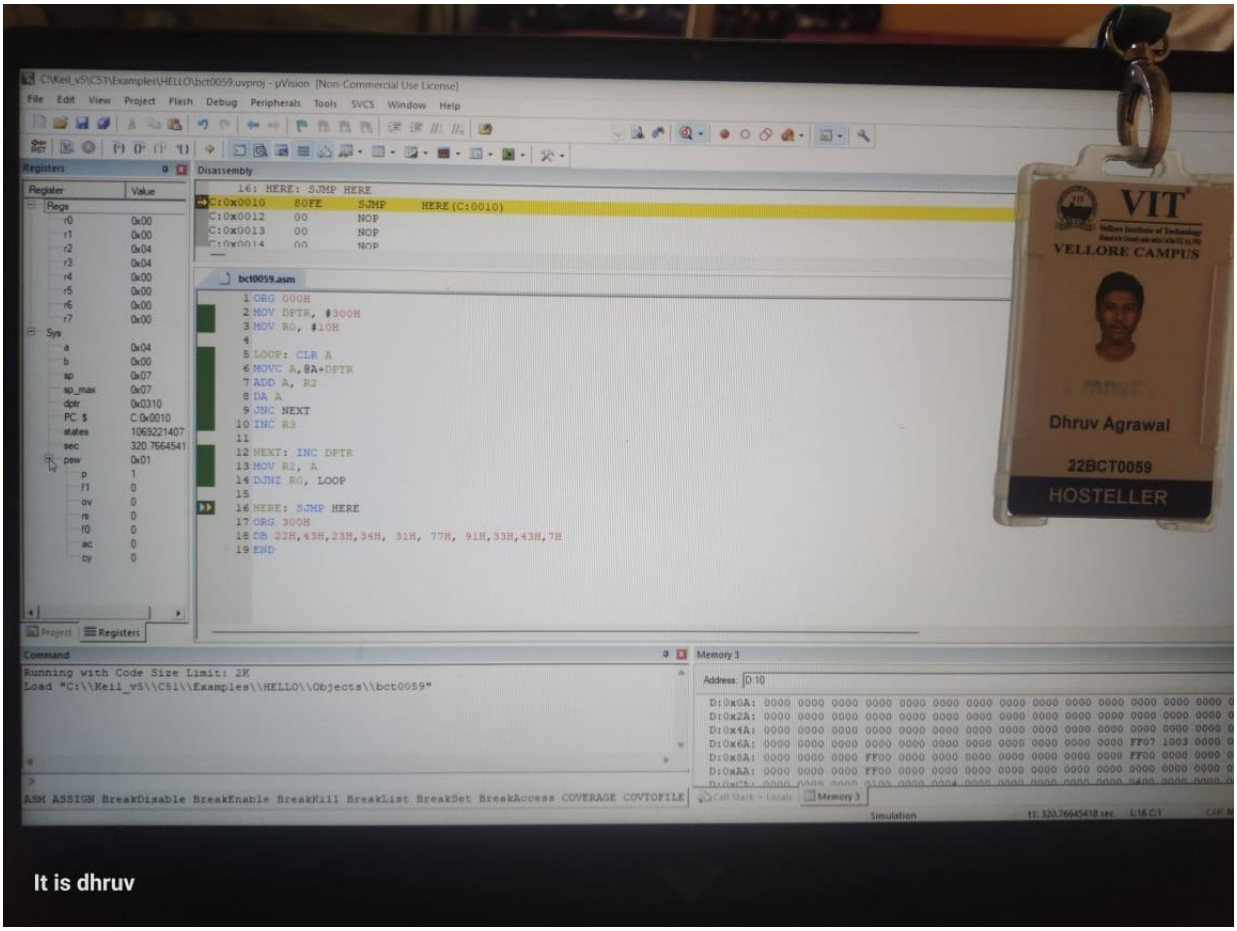
= 404H

∴ R2 = 04H, R3 = 04H

Ans

= (1000000 0100)2

Lab Screenshot (with PSW):



Conclusion: Transfer of data from code space to RAM space in reverse and same order is essential for data manipulation. It is typically done byte by byte. The data in code is added in accumulator to store the result in R2, R3. For BCD no., we need to use decimal adjust instruction.