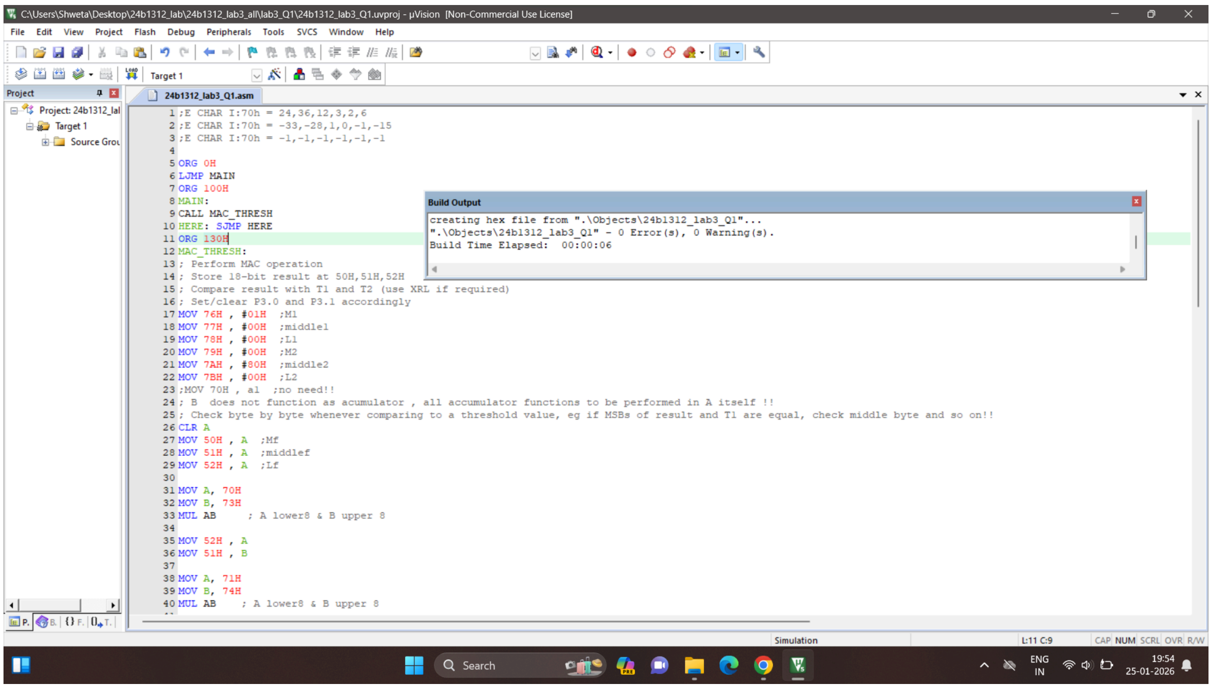
24B1312 , Supriya Anand Mishra
Microprocessors Lab 3
Electrical Eng, IITB

Q1)

SUCCESSFUL BUILD!!



C:\Users\Shweta\Desktop\24b1312_lab\24b1312_lab3_all\lab3_Q1\24b1312_lab3_Q1.uvproj - µVision [Non-Commercial Use License]

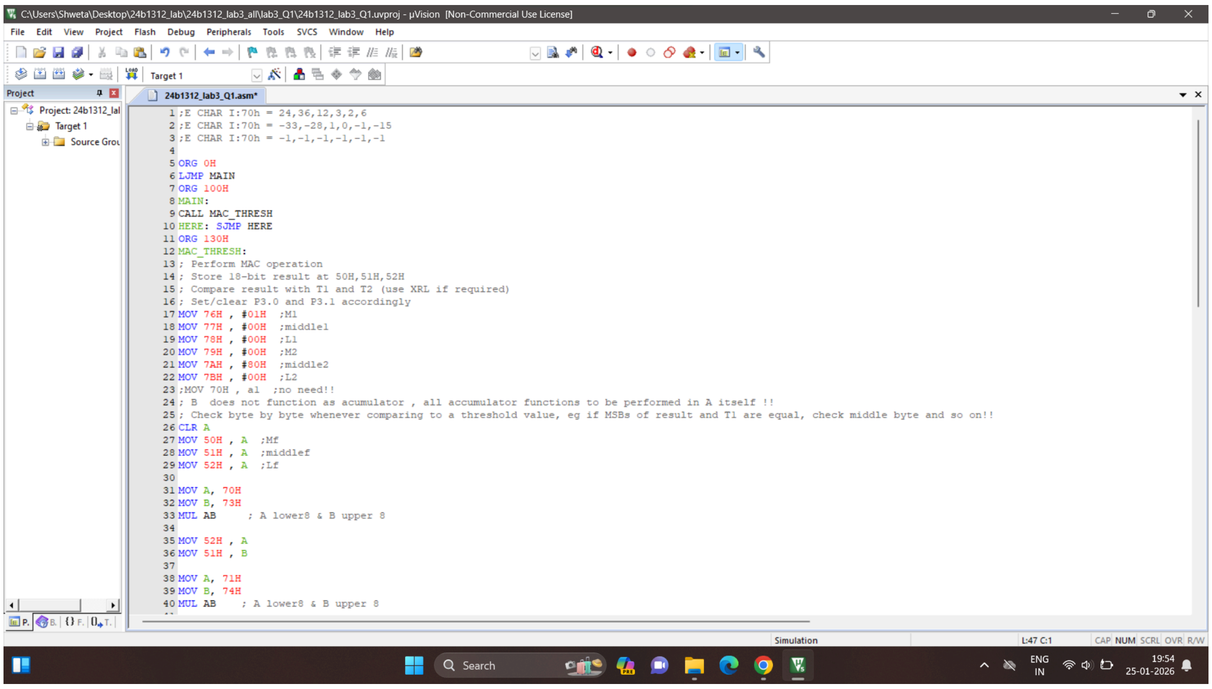
24b1312_lab3_Q1.asm

```
;E CHAR I:70h = 24,36,12,3,2,6
;E CHAR I:70h = -33,-28,1,0,-1,-15
;E CHAR I:70h = -1,-1,-1,-1,-1,-1

ORG 0H
LJMP MAIN
ORG 100H
MAIN:
CALL MAC_THRESH
HERE: SJMP HERE
ORG 130H
MAC_THRESH:
; Perform MAC operation
; Store 18-bit result at 50H,51H,52H
; Compare result with T1 and T2 (use XRL if required)
; Set/clear P3.0 and P3.1 accordingly
MOV 76H , #01H  ;M1
MOV 77H , #00H  ;middle1
MOV 78H , #00H  ;L1
MOV 79H , #00H  ;M2
MOV 7AH , #80H  ;middle2
MOV 7BH , #00H  ;L2
;MOV 70H , al  ;no need!!
; B  does not function as acumulator , all accumulator functions to be performed in A itself !!
; Check byte by byte whenever comparing to a threshold value, eg if MSBs of result and T1 are equal, check middle byte and so on!!
CLR A
MOV 50H , A  ;Mf
MOV 51H , A  ;middlef
MOV 52H , A  ;Lf

MOV A, 70H
MOV B, 73H
MUL AB     ; A lower8 & B upper 8

MOV 52H , A
MOV 51H , B

MOV A, 71H
MOV B, 74H
MUL AB    ; A lower8 & B upper 8
```

Build Output

```
creating hex file from ".\Objects\24b1312_lab3_Q1"...
".\Objects\24b1312_lab3_Q1" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:06
```

CODE FOR Q1!!

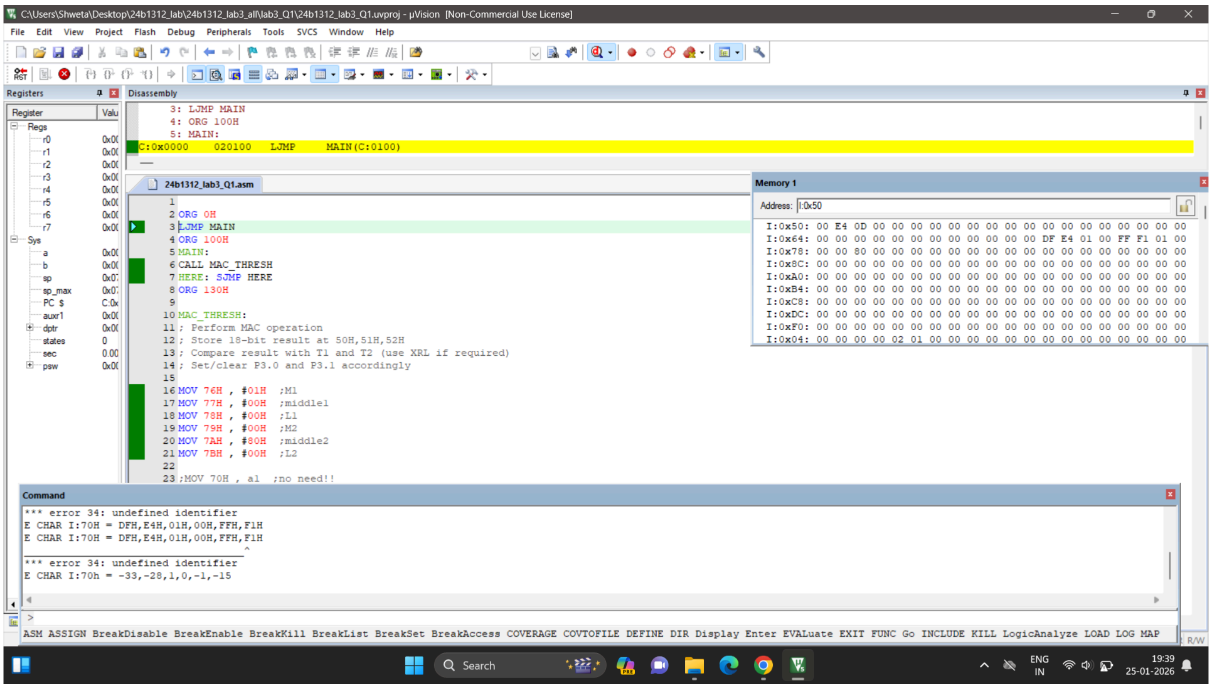


C:\Users\Shweta\Desktop\24b1312_lab\24b1312_lab3_all\lab3_Q1\24b1312_lab3_Q1.uvproj - µVision [Non-Commercial Use License]

24b1312_lab3_Q1.asm*

```
;E CHAR I:70h = 24,36,12,3,2,6
;E CHAR I:70h = -33,-28,1,0,-1,-15
;E CHAR I:70h = -1,-1,-1,-1,-1,-1

ORG 0H
LJMP MAIN
ORG 100H
MAIN:
CALL MAC_THRESH
HERE: SJMP HERE
ORG 130H
MAC_THRESH:
; Perform MAC operation
; Store 18-bit result at 50H,51H,52H
; Compare result with T1 and T2 (use XRL if required)
; Set/clear P3.0 and P3.1 accordingly
MOV 76H , #01H  ;M1
MOV 77H , #00H  ;middle1
MOV 78H , #00H  ;L1
MOV 79H , #00H  ;M2
MOV 7AH , #80H  ;middle2
MOV 7BH , #00H  ;L2
;MOV 70H , al  ;no need!!
; B  does not function as acumulator , all accumulator functions to be performed in A itself !!
; Check byte by byte whenever comparing to a threshold value, eg if MSBs of result and T1 are equal, check middle byte and so on!!
CLR A
MOV 50H , A  ;Mf
MOV 51H , A  ;middlef
MOV 52H , A  ;Lf

MOV A, 70H
MOV B, 73H
MUL AB     ; A lower8 & B upper 8

MOV 52H , A
MOV 51H , B

MOV A, 71H
MOV B, 74H
MUL AB    ; A lower8 & B upper 8
```

- **a1,a2,a3 = DFH,E4H,01H, b1,b2,b3 = 00H,FFH,F1H CASE !!**



C:\Users\Shweta\Desktop\24b1312_lab\24b1312_lab3_all\lab3_Q1\24b1312_lab3_Q1.uvproj - µVision [Non-Commercial Use License]

Disassembly

```
     3: LJMP MAIN
     4: ORG 100H
     5: MAIN:
C:0x0000    020100   LJMP     MAIN(C:0100)
```

24b1312_lab3_Q1.asm

```

ORG 0H
LJMP MAIN
ORG 100H
MAIN:
CALL MAC_THRESH
HERE: SJMP HERE
ORG 130H

MAC_THRESH:
; Perform MAC operation
; Store 18-bit result at 50H,51H,52H
; Compare result with T1 and T2 (use XRL if required)
; Set/clear P3.0 and P3.1 accordingly

MOV 76H , #01H  ;M1
MOV 77H , #00H  ;middle1
MOV 78H , #00H  ;L1
MOV 79H , #00H  ;M2
MOV 7AH , #80H  ;middle2
MOV 7BH , #00H  ;L2

;MOV 70H , a1  ;no need!!
```

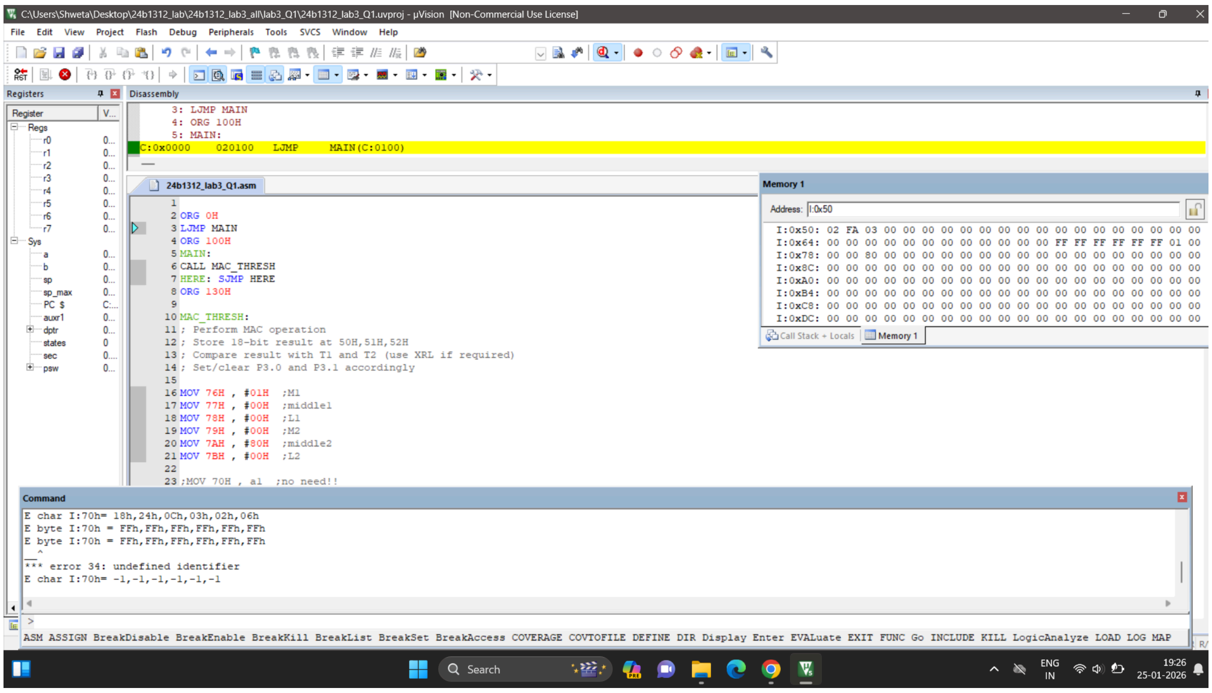
Memory 1 — Address: I:0x50

```
I:0x50: 00 E4 0D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x64: 00 00 00 00 00 00 00 00 00 00 00 00 DF E4 01 00 FF F1 01 00
I:0x78: 00 00 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x8C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0xA0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0xB4: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0xC8: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0xDC: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0xF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x04: 00 00 00 00 02 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Command

```
*** error 34: undefined identifier
E CHAR I:70H = DFH,E4H,01H,00H,FFH,F1H
E CHAR I:70H = DFH,E4H,01H,00H,FFH,F1H
                                     ^
*** error 34: undefined identifier
E CHAR I:70h = -33,-28,1,0,-1,-15
```

- **a1,a2,a3 = FFH,FFH,FFH, b1,b2,b3 = FFH,FFH,FFH CASE !!**



C:\Users\Shweta\Desktop\24b1312_lab\24b1312_lab3_all\lab3_Q1\24b1312_lab3_Q1.uvproj - µVision [Non-Commercial Use License]

Disassembly

```
     3: LJMP MAIN
     4: ORG 100H
     5: MAIN:
C:0x0000    020100   LJMP     MAIN(C:0100)
```

24b1312_lab3_Q1.asm

```

ORG 0H
LJMP MAIN
ORG 100H
MAIN:
CALL MAC_THRESH
HERE: SJMP HERE
ORG 130H

MAC_THRESH:
; Perform MAC operation
; Store 18-bit result at 50H,51H,52H
; Compare result with T1 and T2 (use XRL if required)
; Set/clear P3.0 and P3.1 accordingly

MOV 76H , #01H  ;M1
MOV 77H , #00H  ;middle1
MOV 78H , #00H  ;L1
MOV 79H , #00H  ;M2
MOV 7AH , #80H  ;middle2
MOV 7BH , #00H  ;L2

;MOV 70H , a1  ;no need!!
```

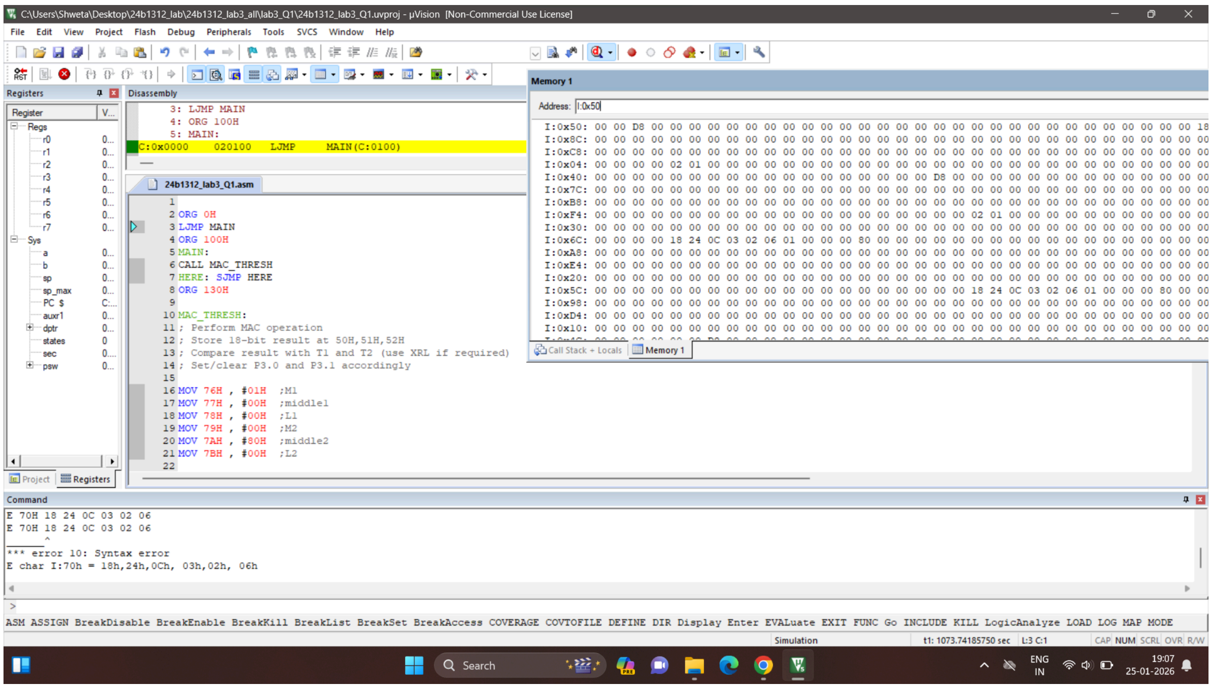
Memory 1 — Address: I:0x50

```
I:0x50: 02 FA 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x64: 00 00 00 00 00 00 00 00 00 00 00 00 FF FF FF FF FF FF 01 00
I:0x78: 00 00 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x8C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0xA0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0xB4: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0xC8: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0xDC: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Command

```
E char I:70h= 18h,24h,0Ch,03h,02h,06h
E byte I:70h = FFh,FFh,FFh,FFh,FFh,FFh
E byte I:70h = FFh,FFh,FFh,FFh,FFh,FFh
  ^
*** error 34: undefined identifier
E char I:70h= -1,-1,-1,-1,-1,-1
```

• **a1,a2,a3 = 18H,24H,0CH, b1,b2,b3 = 03H,02H,06H**


C:\Users\Shweta\Desktop\24b1312_lab\24b1312_lab3_all\lab3_Q1\24b1312_lab3_Q1.uvproj - µVision [Non-Commercial Use License]

Disassembly
```
     3: LJMP MAIN
     4: ORG 100H
     5: MAIN:
C:0x0000    020100   LJMP     MAIN(C:0100)
```

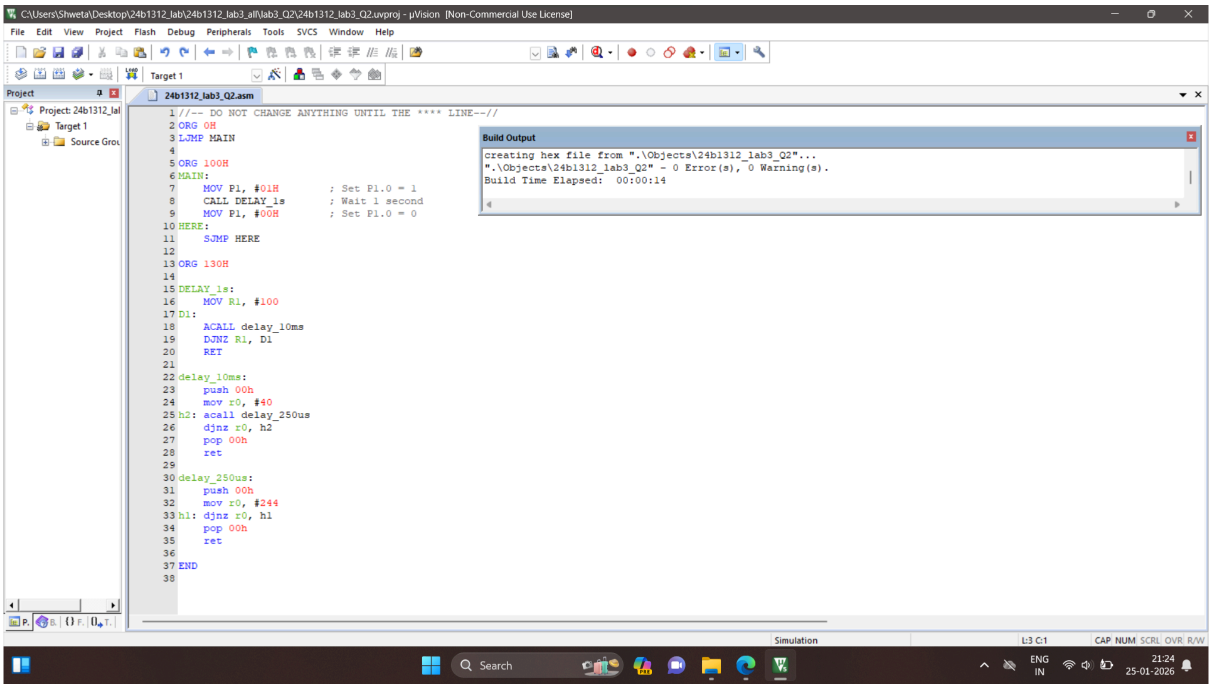
24b1312_lab3_Q1.asm
```

ORG 0H
LJMP MAIN
ORG 100H
MAIN:
CALL MAC_THRESH
HERE: SJMP HERE
ORG 130H

MAC_THRESH:
; Perform MAC operation
; Store 18-bit result at 50H,51H,52H
; Compare result with T1 and T2 (use XRL if required)
; Set/clear P3.0 and P3.1 accordingly

MOV 76H , #01H  ;M1
MOV 77H , #00H  ;middle1
MOV 78H , #00H  ;L1
MOV 79H , #00H  ;M2
MOV 7AH , #80H  ;middle2
MOV 7BH , #00H  ;L2

```

Memory 1 — Address: I:0x50
```
I:0x50: 00 00 D8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 18
I:0x8C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0xC8: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x04: 00 00 00 00 02 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 D8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x7C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0xB8: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0xF4: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 01 00 00 00 00 00 00 00 00 00 00 00 00
I:0x30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x6C: 00 00 00 00 18 24 0C 03 02 06 01 00 00 00 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0xA8: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0xE4: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x5C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 18 24 0C 03 02 06 01 00 00 00 80 00 00
I:0x98: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0xD4: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Command
```
E 70H 18 24 0C 03 02 06
E 70H 18 24 0C 03 02 06
      ^
*** error 10: Syntax error
E char I:70h = 18h,24h,0Ch, 03h,02h, 06h
```
```
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR Display Enter EVALuate EXIT FUNC Go INCLUDE KILL LogicAnalyze LOAD LOG MAP MODE
```
Simulation   t1: 1073.74185750 sec   L:3 C:1   19:07 25-01-2026

Q2)

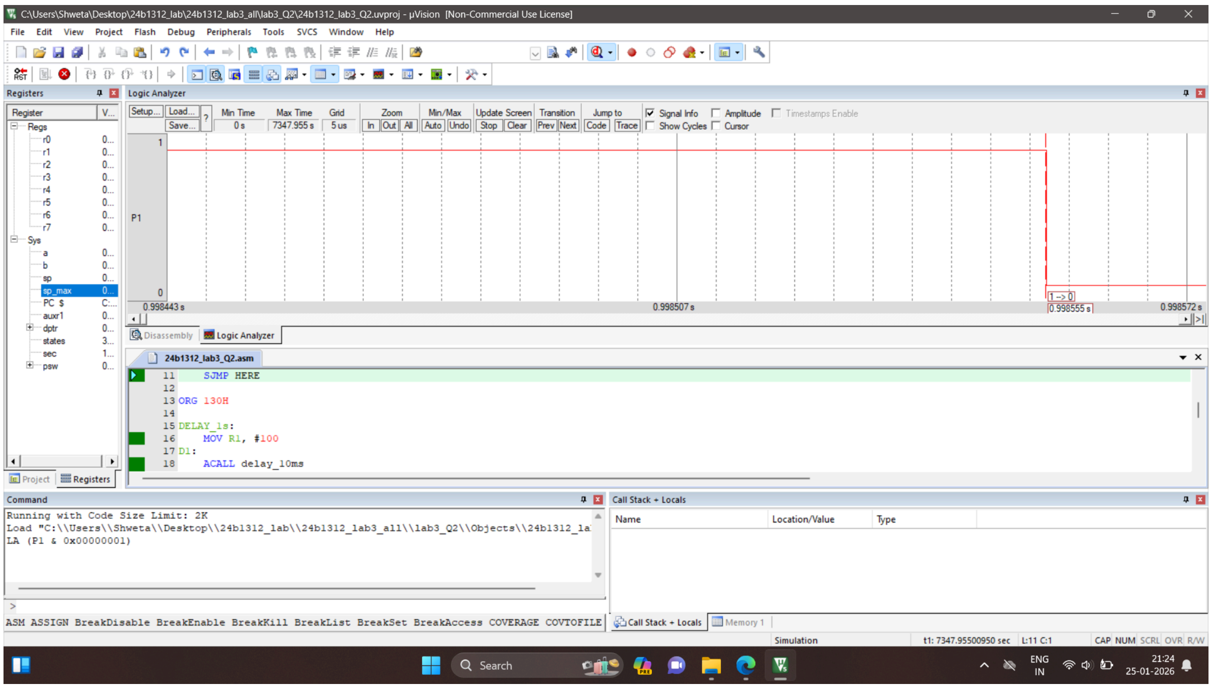
SUCCESSFUL BUILD!!


C:\Users\Shweta\Desktop\24b1312_lab\24b1312_lab3_all\lab3_Q2\24b1312_lab3_Q2.uvproj - µVision [Non-Commercial Use License]

24b1312_lab3_Q2.asm
```
//-- DO NOT CHANGE ANYTHING UNTIL THE **** LINE--//
ORG 0H
LJMP MAIN

ORG 100H
MAIN:
    MOV P1, #01H        ; Set P1.0 = 1
    CALL DELAY_1s       ; Wait 1 second
    MOV P1, #00H        ; Set P1.0 = 0
HERE:
    SJMP HERE

ORG 130H

DELAY_1s:
    MOV R1, #100
D1:
    ACALL delay_10ms
    DJNZ R1, D1
    RET

delay_10ms:
    push 00h
    mov r0, #40
h2: acall delay_250us
    djnz r0, h2
    pop 00h
    ret

delay_250us:
    push 00h
    mov r0, #244
h1: djnz r0, h1
    pop 00h
    ret

END

```

Build Output
```
creating hex file from ".\Objects\24b1312_lab3_Q2"...
".\Objects\24b1312_lab3_Q2" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:14
```
Simulation   L:3 C:1   21:24 25-01-2026

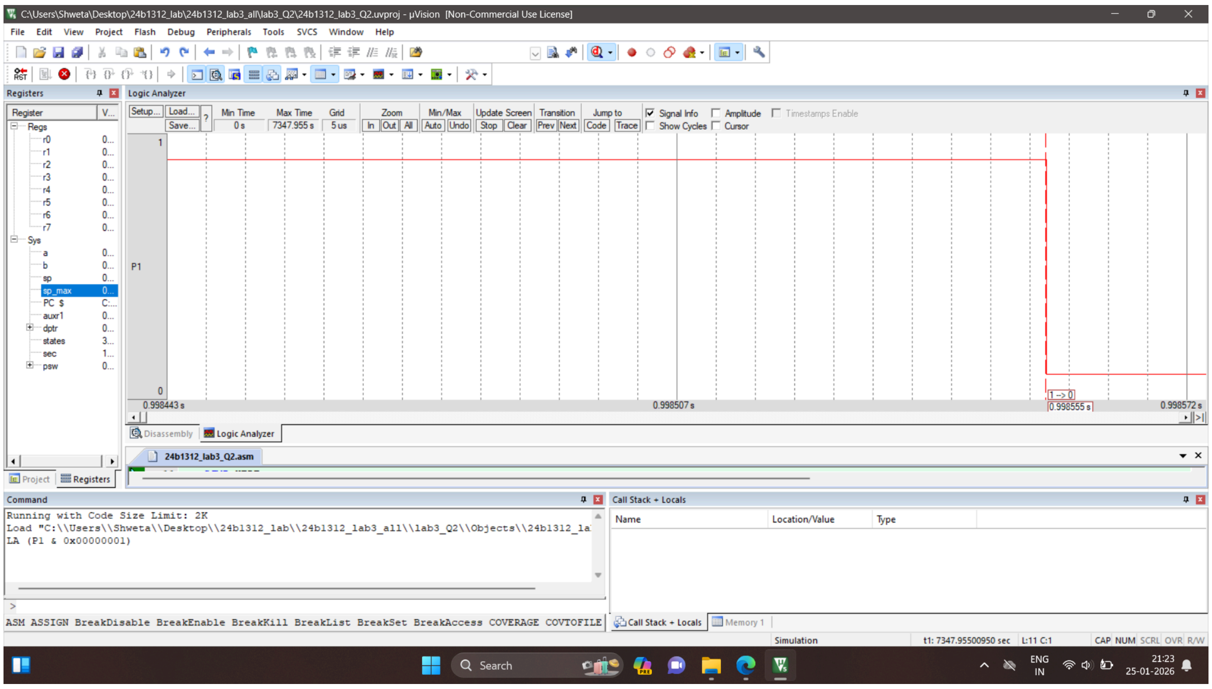
LOGIC ANALYZER!!



C:\Users\Shweta\Desktop\24b1312_lab\24b1312_lab3_all\lab3_Q2\24b1312_lab3_Q2.uvproj - µVision [Non-Commercial Use License]

```
      SJMP HERE

ORG 130H

DELAY_1s:
      MOV R1, #100
D1:
      ACALL delay_10ms
```

```
Running with Code Size Limit: 2K
Load "C:\\Users\\Shweta\\Desktop\\24b1312_lab\\24b1312_lab3_all\\lab3_Q2\\Objects\\24b1312_la
LA (P1 & 0x00000001)
```



C:\Users\Shweta\Desktop\24b1312_lab\24b1312_lab3_all\lab3_Q2\24b1312_lab3_Q2.uvproj - µVision [Non-Commercial Use License]

```
Running with Code Size Limit: 2K
Load "C:\\Users\\Shweta\\Desktop\\24b1312_lab\\24b1312_lab3_all\\lab3_Q2\\Objects\\24b1312_la
LA (P1 & 0x00000001)
```

AIM

To implement a Multiply–Accumulate (MAC) operation using 8051 assembly language and perform two-threshold comparison on the computed result.

Additionally, to generate an accurate 1-second delay using instruction cycle timing and verify it using a logic analyzer.

PROCEDURE / METHOD

- Inputs and weights were stored in internal RAM locations.
- MAC operation was implemented using repeated multiplication and 16-bit addition.
- The 18-bit MAC result was stored across three memory locations.
- Threshold comparison was performed and Port P3 bits were set accordingly.
- A 1-second delay was generated by repeatedly calling a calibrated 10 ms delay subroutine.

RESULT / OBSERVATION

- The MAC result was correctly computed and stored in memory locations 50H–52H.
- Threshold conditions were correctly detected and P3.0 and P3.1 were set/cleared as specified.
- Port P1.0 remained HIGH for approximately 1 second and then went LOW.
- The delay measured on the logic analyzer closely matched the expected 1 second duration at 24 MHz clock frequency.

CONCLUSION

The MAC operation and two-threshold decision logic were successfully implemented using 8051 assembly language.

A precise 1-second delay was generated using instruction cycle timing and verified experimentally using a logic analyzer, confirming correct program execution.

WHAT I LEARNT

- How to implement Multiply–Accumulate (MAC) operations in assembly language.
- How to handle multi-byte (18-bit) arithmetic in the 8051.
- How instruction cycle timing depends on clock frequency and machine cycles.
- How to construct long delays using smaller calibrated delay routines.
- How to verify timing behavior using a logic analyzer.

This experiment helped me understand instruction-level timing, multi-byte arithmetic, and how DSP-style MAC operations and delays are implemented on the 8051.
