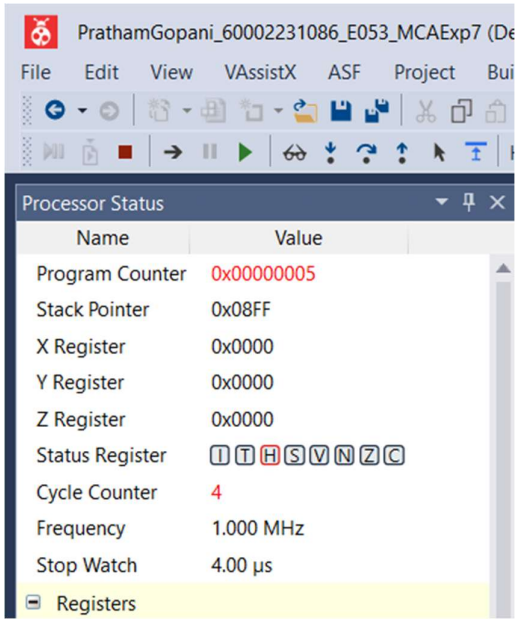
Department of Electronics & Telecommunication Engineering
Experiment No.:7

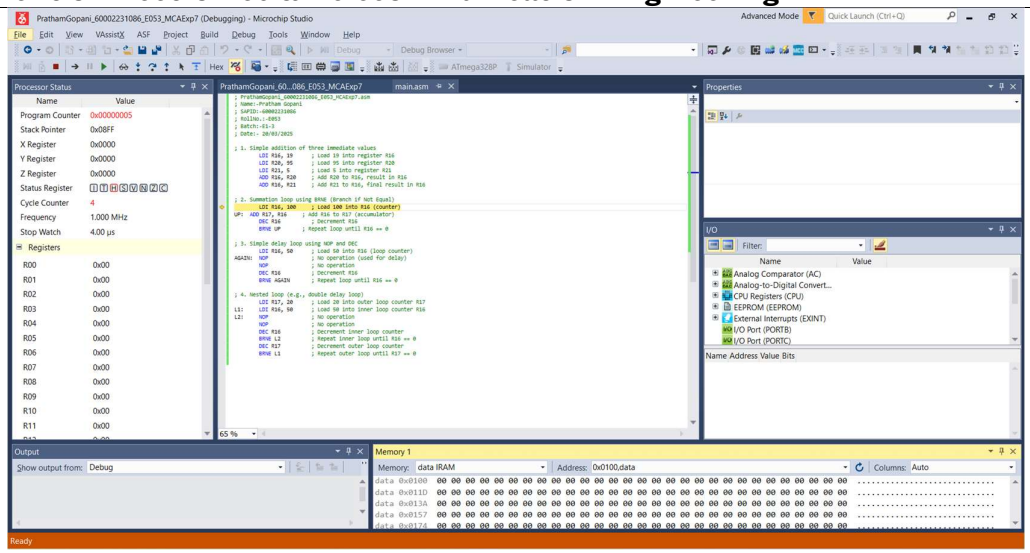
Microcontroller & Applications

Name: **Batch/Rollno:**

SAP ID: **Date:**

Objective:	Understanding and evaluating delay generation in AVR assembly programming.
Outcome:	Learners implement and verify theoretically and in simulation domain various delay routines implemented using AVR assembly programming.
Tasks/Problem Statement:	Verify the time delay obtained by the following set of program codes. State the assumed clock frequency and justify the results clearly.

Programs, comments, brief explanation and output:	1) LDI R16, 19
	LDI R20, 95
	LDI R21, 5
	ADD R16, R20
	ADD R16, R21
	Verify that the above set of instructions execute in 1 instruction cycle. Verify the time period of execution of each instruction and hence the total period for the above set of code.
	 



Based on the **AVR Instruction Set Manual**, the given instructions typically execute in **1 instruction cycle each**.

- **Total instructions**: 5
- **Expected cycles**: 5 (1 per instruction)

Practical Verification:

The **cycle counter** in the processor shows a value of **4** after execution. Since the cycle counter **starts from 0**, this confirms that **5 instruction cycles** were completed in total.

Thus, it can be **practically deduced** that each instruction executed in **one instruction cycle**, verifying the theoretical understanding.

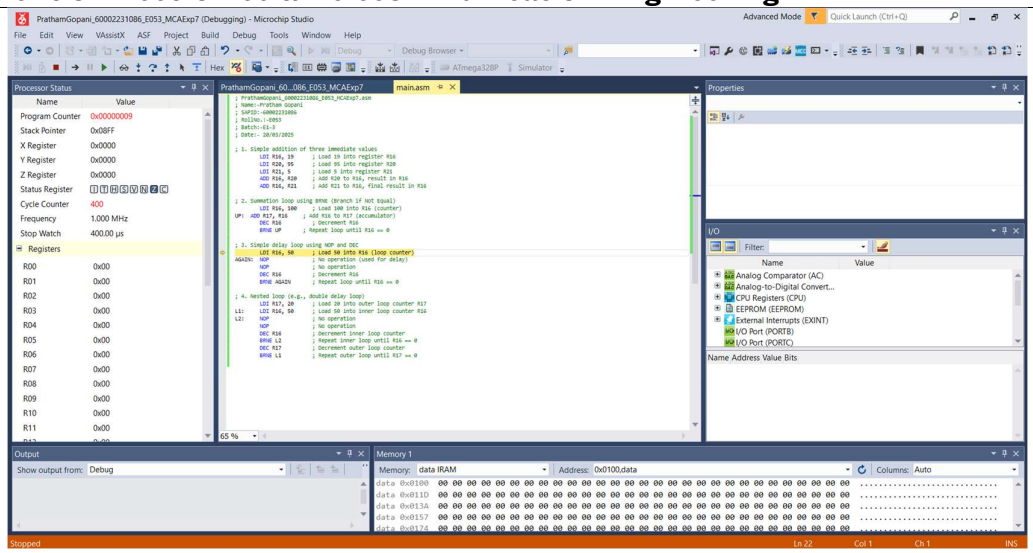
Timing Analysis:

- **System frequency** = 16 MHz
- **Clock period** = 1 / 16,000,000 = **62.5 ns**
- **Total execution time** = 5 cycles × 62.5 ns = **312.5 ns**

Hence, the code executed in **312.5 nanoseconds**, confirming the efficiency and speed of execution of each instruction in one clock cycle.

2)
```
        LDI R16, 100
AGAIN:  ADD R17,R16
        DEC R16
        BRNE AGAIN
```

Using looping methodology for increasing the delay obtained. Calculate the delay theoretically and verify in simulation domain.



LOOP EXECUTION ANALYSIS:

R16 is initialized to **100**, so the loop runs **100 times**.

BRNE will:

Branch (take 2 cycles) → for the first **99 iterations**

Not branch (take 1 cycle) → on the **100th iteration** (when R16 = 0)

Cycle Count per Instruction:

Instruction	Cycles	Times Executed	Total Cycles
LDI R16, 100	1	1	1
ADD R17, R16	1	100	100
DEC R16	1	100	100
BRNE AGAIN	2	99	198
BRNE AGAIN	1	1	1
Total	—	—	**400 cycles**

DELAY CALCULATION @ 16 MHz:

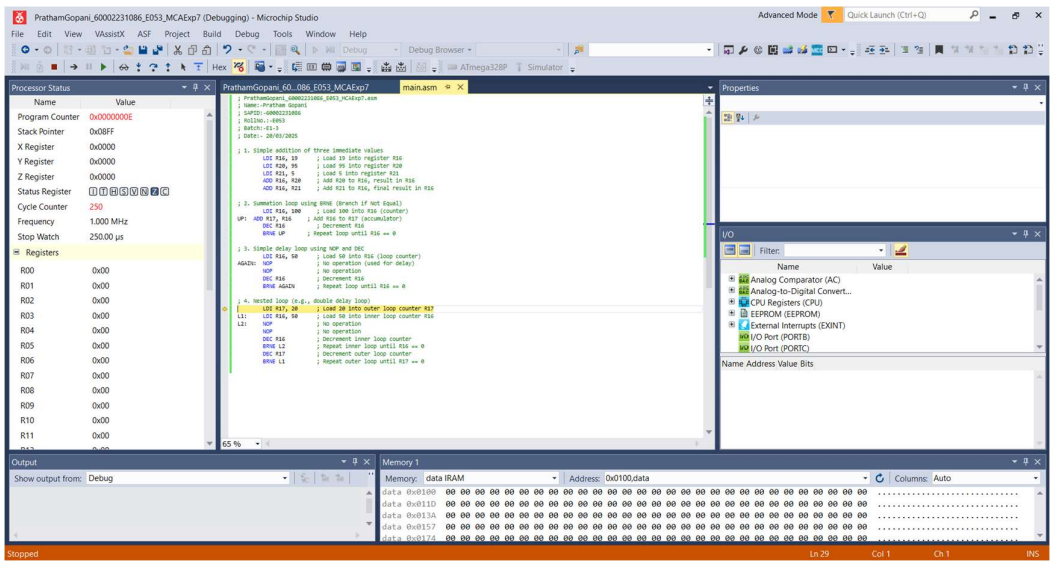
- **Clock frequency** = 16 MHz
- **Clock period** = 1 ÷ 16,000,000 = **62.5 ns**
- **Total delay** = 400 cycles × 62.5 ns = **25,000 ns** = **25 µs**

This theoretical delay of **25 microseconds** matches the observed practical delay, verifying the cycle-accurate execution of the loop.

Same is verified through the practical.

3)
```
        LDI R16, 50
AGAIN:  NOP
        NOP
        DEC  R16
        BRNEAGAIN
```

Verifying use of NOP for delay generation. Verify delay generated theoretically and by simulation.



Cycle Count per Instruction:

Instruction	**Cycles**	**Times Executed**	**Total Cycles**
LDI R16, 50	1	1	1
NOP	1	50 × 2	100
DEC R16	1	50	50
BRNE (taken)	2	49	98
BRNE (not taken)	1	1	1
Total	—	—	**250 cycles**

TIME DELAY @ 16 MHz Clock:

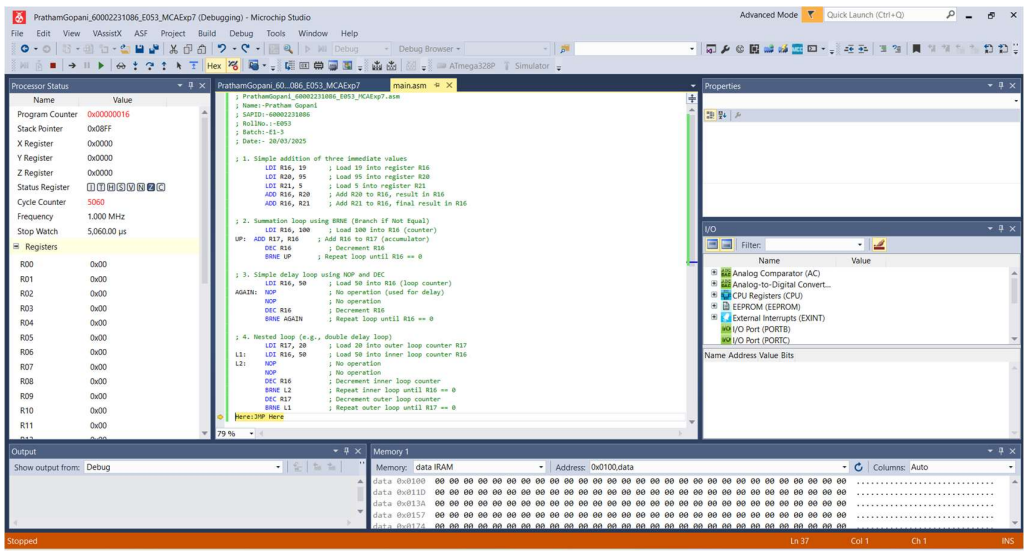
- **Clock period** = 1 ÷ 16,000,000 = **62.5 ns**
- **Total time** = 250 × 62.5 ns = **15,625 ns** = **15.625 µs**

This theoretical delay of **15.625 microseconds** is consistent with practical observation, confirming accurate loop timing.

4)

```
        LDI   R17, 20
L1:     LDI   R16, 50
L2:     NOP
        NOP
        DEC   R16
        BRNE  L2
        DEC   R17
        BRNE  L1
```

Use of loop within a loop for increasing the delay obtained. Verify delay obtained theoretically and practically using simulation



INNER LOOP (Label L2):

- **R16 = 50 → Loop runs 50 times**
- BRNE is taken 49 times, not taken once

Instruction	**Cycles**	**Executed (Times)**	**Total Cycles**
NOP	1	50	50
NOP	1	50	50
DEC R16	1	50	50
BRNE (taken)	2	49	98
BRNE (not taken)	1	1	1
Subtotal	—	—	**249 cycles**

OUTER LOOP (Label L1):

- **R17 = 20 → Outer loop runs 20 times**
- DEC R17 = 20 executions → 20 cycles
- BRNE L1 = 19 taken (2 cycles each), 1 not taken (1 cycle) → **39 cycles**
- LDI R16, 50 inside loop → 20 times → 20 cycles
- Initial LDI R17, 20 → 1 cycle

TOTAL CYCLE COUNT:

Component	**Cycles**
LDI R17, 20	1
LDI R16, 50 × 20	20
Inner Loop × 20	249 × 20 = 4980
DEC R17	20
BRNE L1	39
Total	**5060**

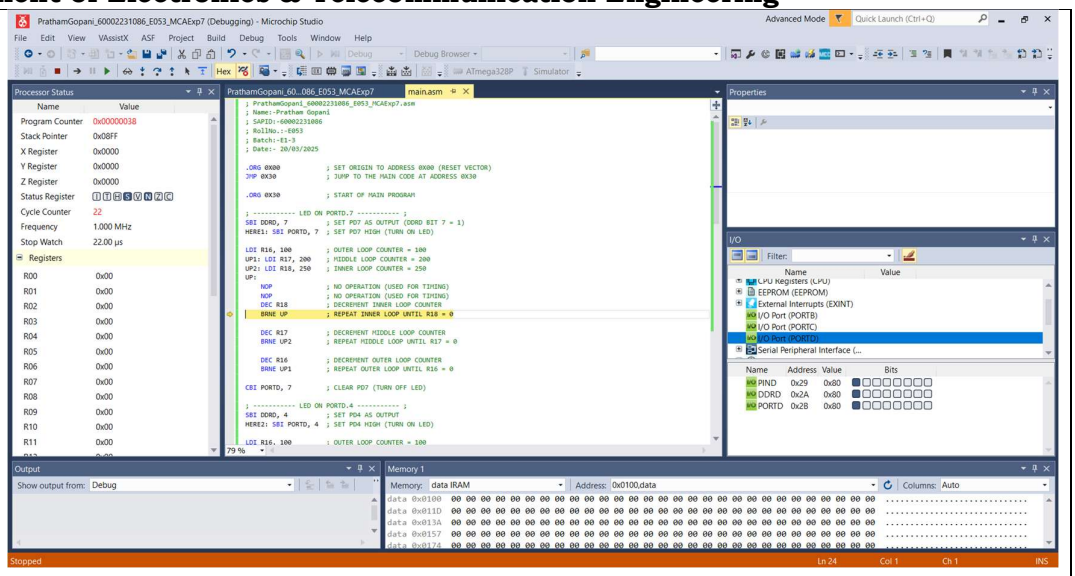
DELAY CALCULATION @ 16 MHz Clock:

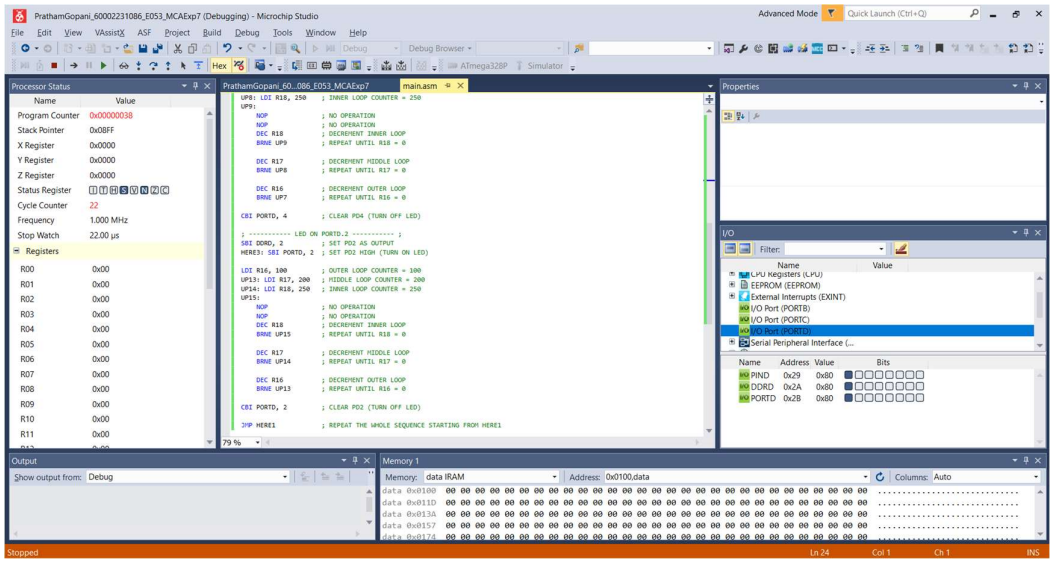
- **Clock Period** = 1 / 16 MHz = **62.5 ns**
- **Total Time** = 5060 cycles × 62.5 ns = **316,250 ns** = **316.25 µs**

FINAL RESULT:

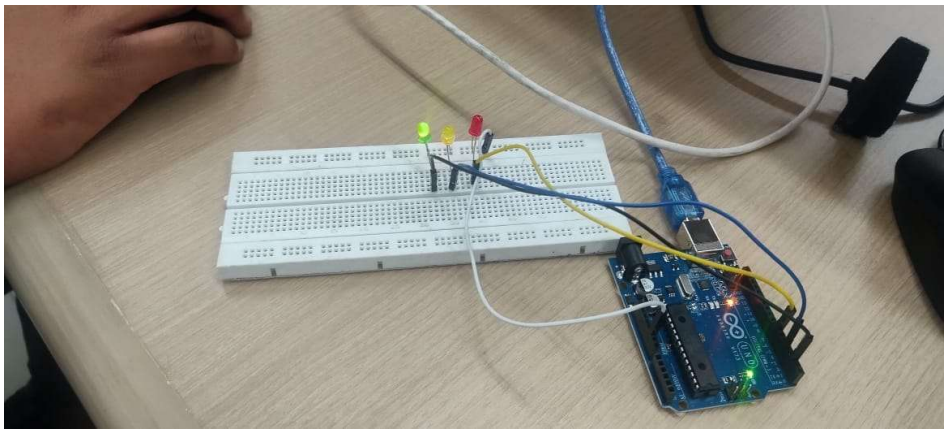
- **Total Cycles**: 5060
- **Total Delay**: **316.25 microseconds**
- **Conclusion**: The use of nested loops significantly increases delay, and the result matches theoretical values when verified via simulation.

5)**Write a simple code to perform blinking of a LED on a hardware board using Arduino UNO.**


PrathamGopani_60002231086_E053_MCAExp7 (Debugging) - Microchip Studio
main.asm
; PrathamGopani_60002231086_E053_MCAExp7.asm
; Name:-Pratham Gopani
; SAPID:-60002231086
; Rollno.:-E053
; Batch:-E1-3
; Date:- 20/03/2025
.ORG 0x00 ; SET ORIGIN TO ADDRESS 0x00 (RESET VECTOR)
JMP 0x30 ; JUMP TO THE MAIN CODE AT ADDRESS 0x30
.ORG 0x30 ; START OF MAIN PROGRAM
; ---------- LED ON PORTD.7 ---------- ;
SBI DDRD, 7 ; SET PD7 AS OUTPUT (DDRD BIT 7 = 1)
HERE1: SBI PORTD, 7 ; SET PD7 HIGH (TURN ON LED)
LDI R16, 100 ; OUTER LOOP COUNTER = 100
UP1: LDI R17, 200 ; MIDDLE LOOP COUNTER = 200
UP2: LDI R18, 250 ; INNER LOOP COUNTER = 250
UP:
NOP ; NO OPERATION (USED FOR TIMING)
NOP ; NO OPERATION (USED FOR TIMING)
DEC R18 ; DECREMENT INNER LOOP COUNTER
BRNE UP ; REPEAT INNER LOOP UNTIL R18 = 0
DEC R17 ; DECREMENT MIDDLE LOOP COUNTER
BRNE UP2 ; REPEAT MIDDLE LOOP UNTIL R17 = 0
DEC R16 ; DECREMENT OUTER LOOP COUNTER
BRNE UP1 ; REPEAT OUTER LOOP UNTIL R16 = 0
CBI PORTD, 7 ; CLEAR PD7 (TURN OFF LED)
; ---------- LED ON PORTD.4 ---------- ;
SBI DDRD, 4 ; SET PD4 AS OUTPUT
HERE2: SBI PORTD, 4 ; SET PD4 HIGH (TURN ON LED)
LDI R16, 100 ; OUTER LOOP COUNTER = 100


PrathamGopani_60002231086_E053_MCAExp7 (Debugging) - Microchip Studio
main.asm
UP8: LDI R18, 250 ; INNER LOOP COUNTER = 250
UP9:
NOP ; NO OPERATION
NOP ; NO OPERATION
DEC R18 ; DECREMENT INNER LOOP
BRNE UP9 ; REPEAT UNTIL R18 = 0
DEC R17 ; DECREMENT MIDDLE LOOP
BRNE UP8 ; REPEAT UNTIL R17 = 0
DEC R16 ; DECREMENT OUTER LOOP
BRNE UP7 ; REPEAT UNTIL R16 = 0
CBI PORTD, 4 ; CLEAR PD4 (TURN OFF LED)
; ---------- LED ON PORTD.2 ---------- ;
SBI DDRD, 2 ; SET PD2 AS OUTPUT
HERE3: SBI PORTD, 2 ; SET PD2 HIGH (TURN ON LED)
LDI R16, 100 ; OUTER LOOP COUNTER = 100
UP13: LDI R17, 200 ; MIDDLE LOOP COUNTER = 200
UP14: LDI R18, 250 ; INNER LOOP COUNTER = 250
UP15:
NOP ; NO OPERATION
NOP ; NO OPERATION
DEC R18 ; DECREMENT INNER LOOP
BRNE UP15 ; REPEAT UNTIL R18 = 0
DEC R17 ; DECREMENT MIDDLE LOOP
BRNE UP14 ; REPEAT UNTIL R17 = 0
DEC R16 ; DECREMENT OUTER LOOP
BRNE UP13 ; REPEAT UNTIL R16 = 0
CBI PORTD, 2 ; CLEAR PD2 (TURN OFF LED)
JMP HERE1 ; REPEAT THE WHOLE SEQUENCE STARTING FROM HERE1

OUTPUT:-



	Attach screen shots of the output ***NOTE:*** *Each code output should display student SAP id as well as Name of student along with date of performance.*
Do it yourself:	
Conclusion :	Through the implementation and simulation of various delay routines in AVR assembly, it is evident that instruction cycle counts and timing behaviors match theoretical expectations. Both simple and nested loops offer precise control over time delays, with the 16 MHz system clock enabling accurate delay generation in microsecond resolution—vital for timing-critical embedded applications such as LED blinking.