ESD Lab Assignment No.1

Name : **Ashutosh Rajendra karve**

Bits-ID : **2024ht01021**

Mail : **2024ht01021@wilp.bits-pilani.ac.in**

Contact : **+91 9765541324**

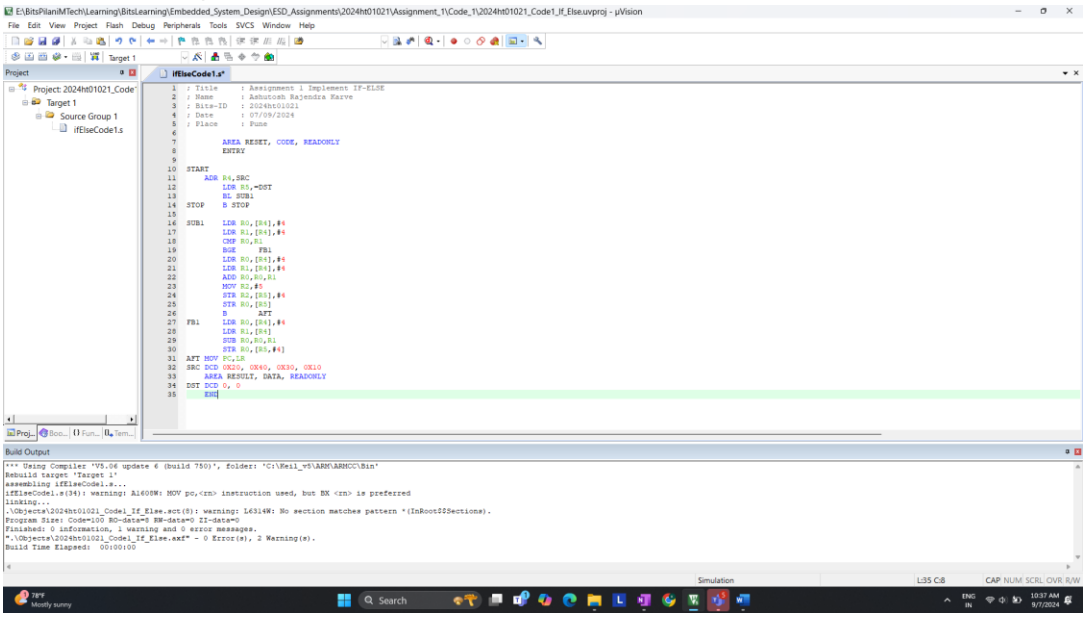
Data : **05/09/2024**

Place : **Pune, Maharashtra**

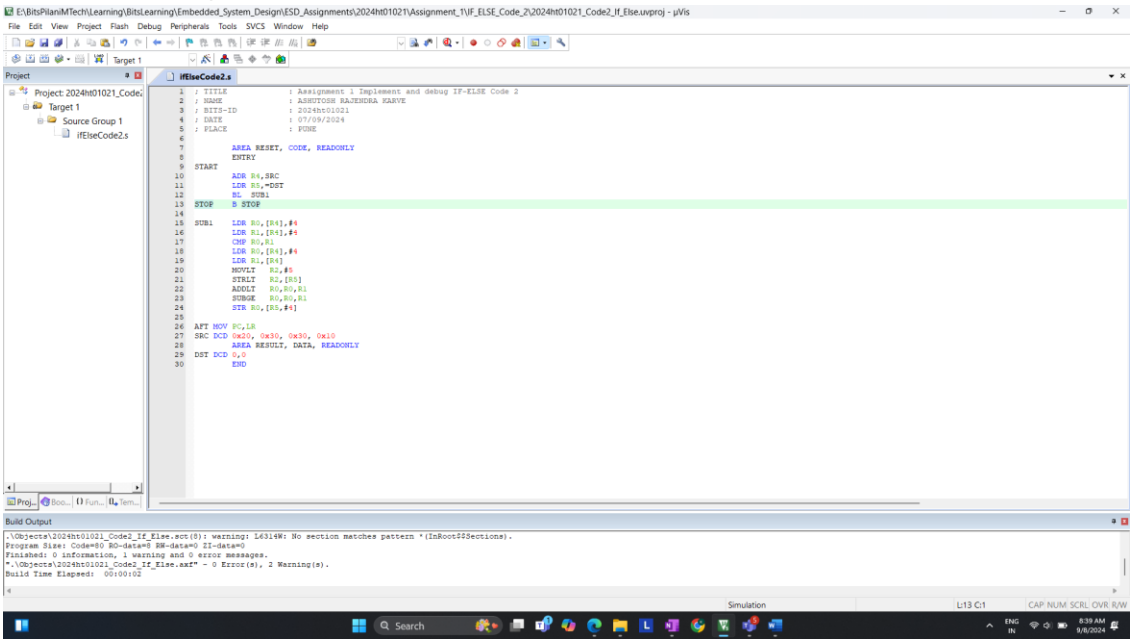
Q.1. Assembly Language Program (ALP) for an LPC2378 processor to implement following IF-ELSE statement are given below:

```
If ( a<b )
{
        x=5;
        y=c+d;
}
else
        y=c-d
```

Code-1 >> Compilation without errors



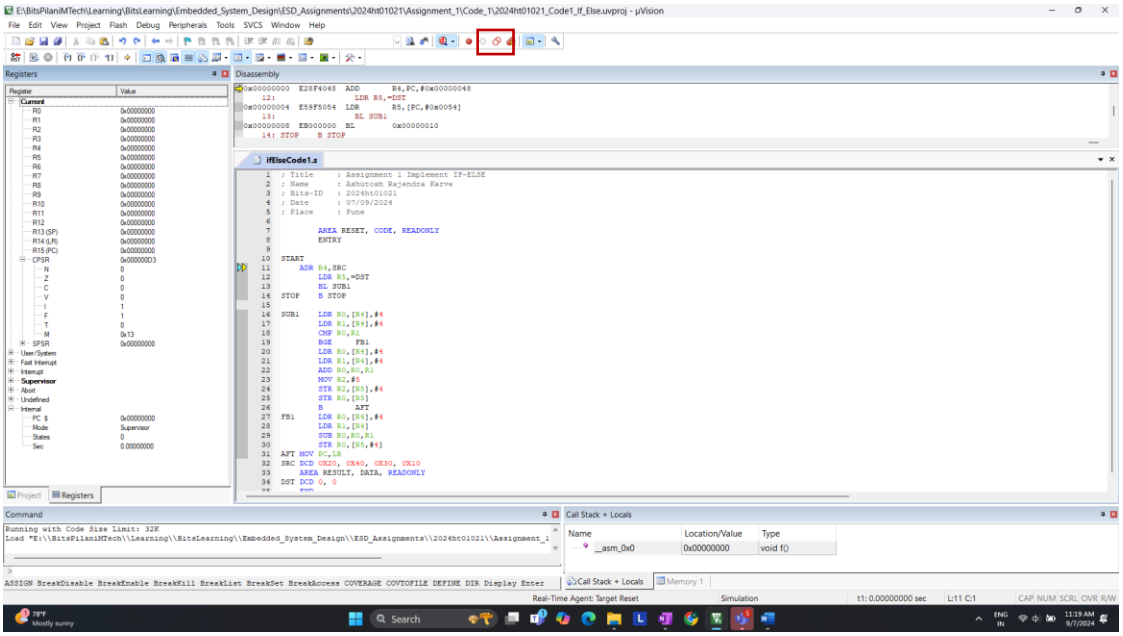
Code- 2 >> Compilation without error



a. On reset what is the LPC2378 processor's state and mode of operation?

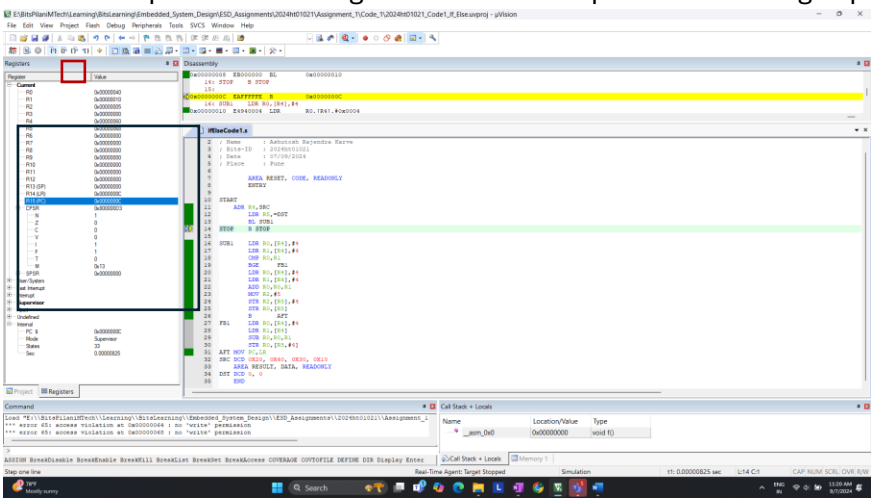
CODE 1 >>

Start of debugging for **code 1** >> Click on debug option as shown in below (figure 1.1)



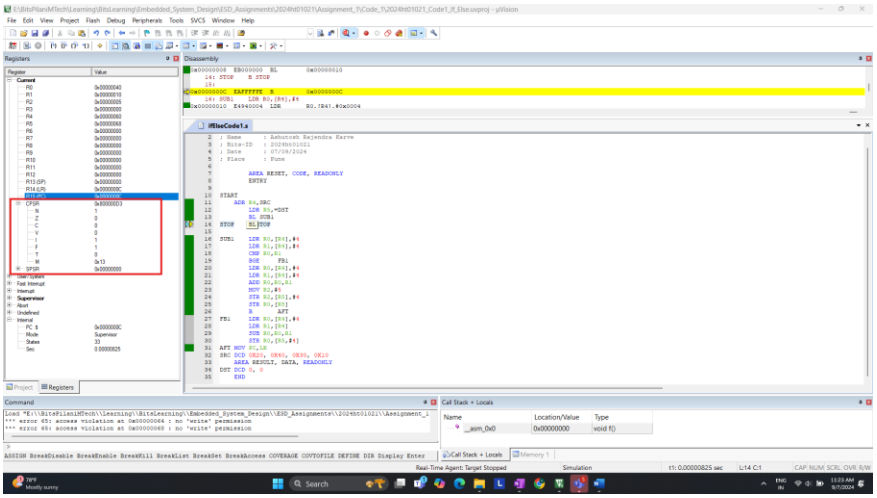
(Figure 1.1)

After >> Step >> Value of Registers have been updated according to program.

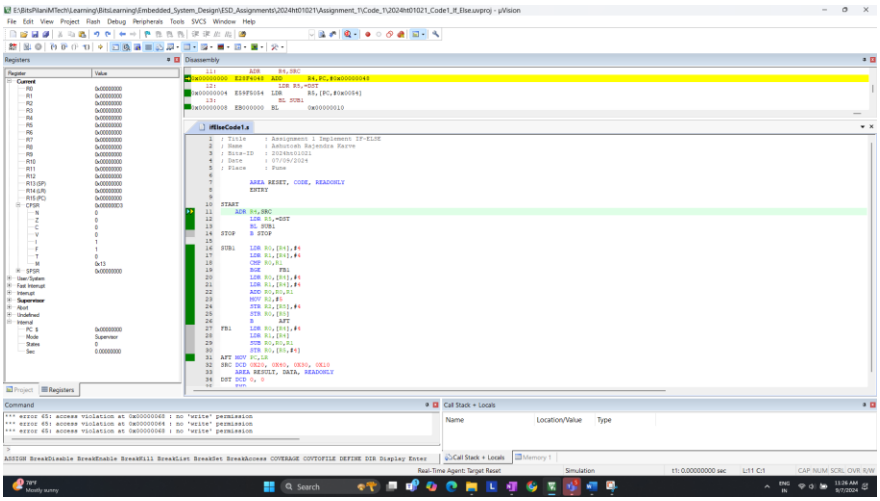


(Figure 1.2)

Observation of Code 1 >> Processor is in supervisor mode (Fig 1.3)



(Figure 1.3)



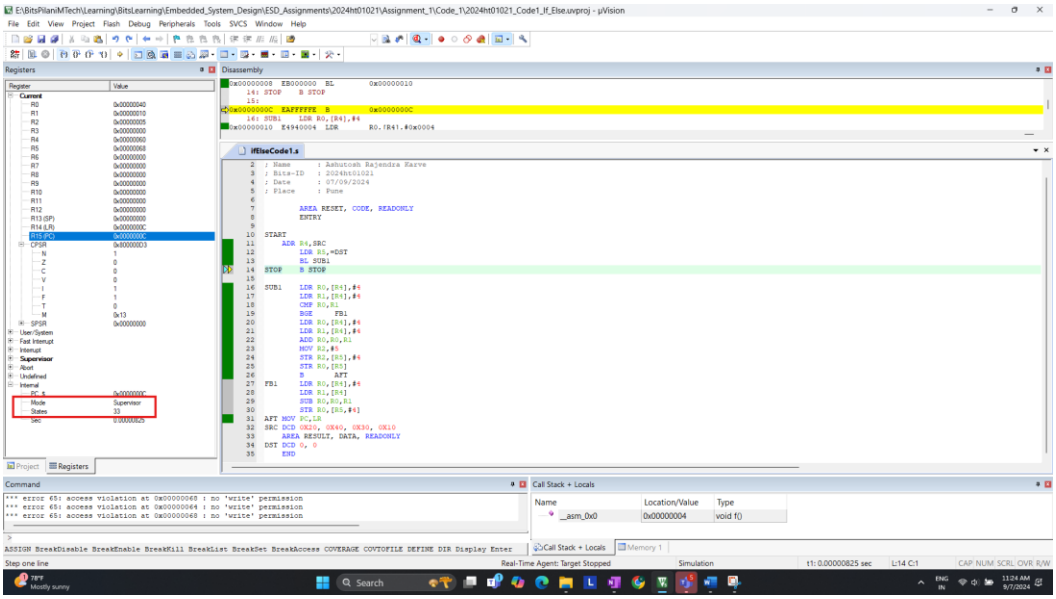
(Figure 1.4)

Answer:

- Looking at **CPSR**. You will see that the processor is in **Supervisor Mode** after reset and **0 states.** Check figure 1.4
- It takes a total of **33 states**. Check figure 1.3

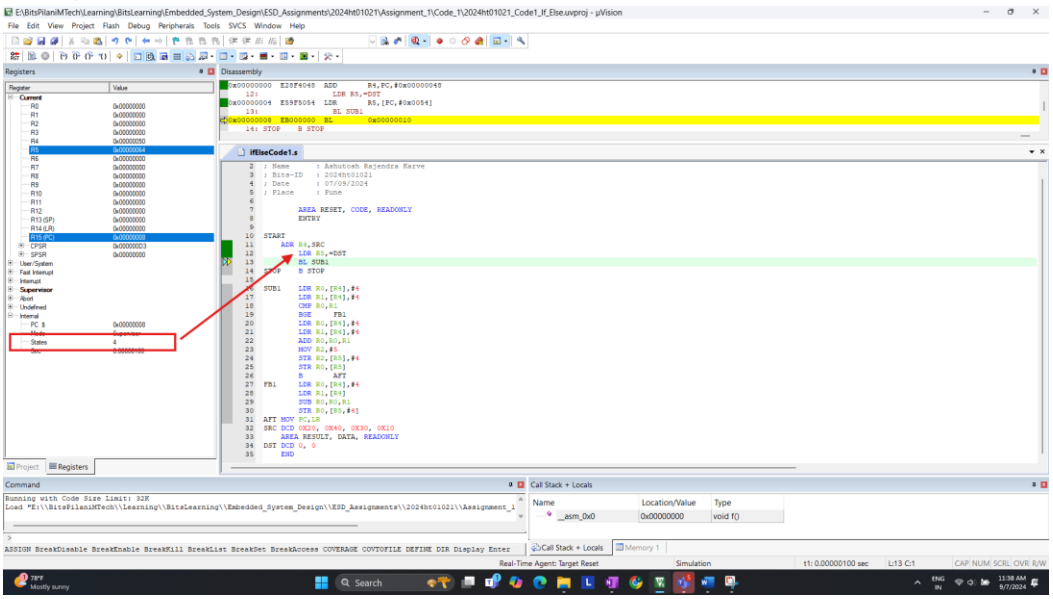
++++++++++++++++++++++++++END of **Q.1 a)**++++++++++++++++++++++++++++++

b. How many states are taken for the execution of an Arithmetic instruction, Load and Store instruction respectively (For Code-1



(Figure 1.5)

Example for **LDR** >> **ADR** take 1 state and after **LDR** it take total 4 states which means 4 – 1 = 3



(Figure 1.6)

Answer:

- **Observing**: Load (**LDR**) , Store (**STR**) & Arithmetic (**ADD**)
    - **LDR** instruction takes **3 Cycles** & **15 Total Cycles** for **LDR**
    - **STR** instruction takes **2 Cycles** & **4 Total Cycles** for **STR**
    - **ADD** instruction takes **1 Cycles** & **1 Total Cycles** for **ADD**
- **Total Cycle** takes place is **33 Cycles.** We can observe in the above screen shot.

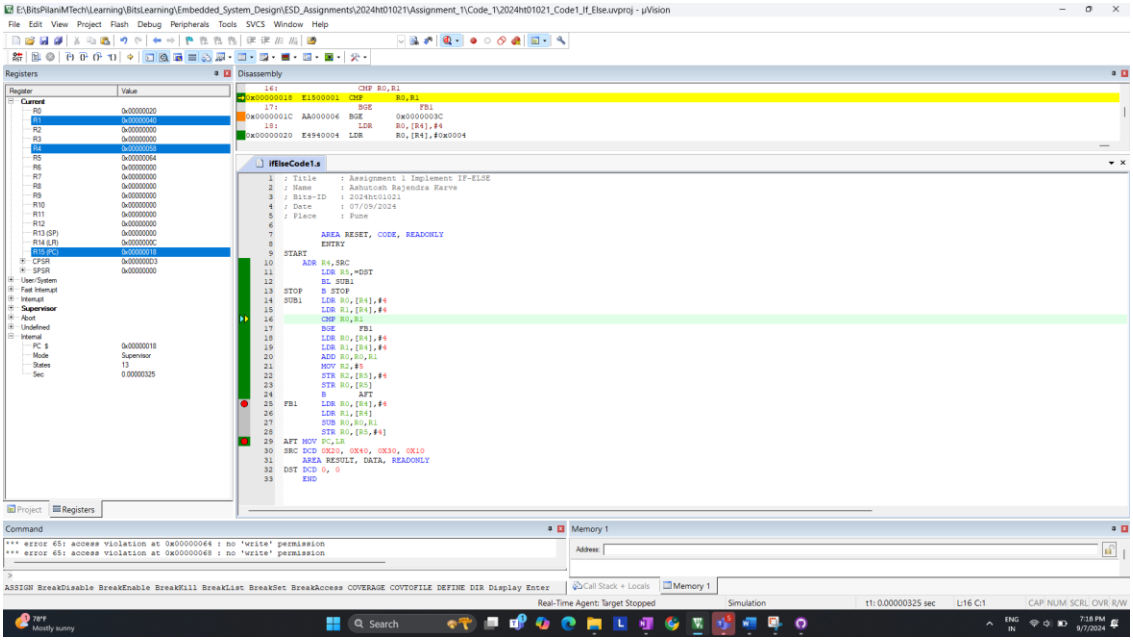
++++++++++++++++++++++++++END of **Q.1 b)**++++++++++++++++++++++++++++++

c. Are the number of states taken for completion the same for BGE instruction if the branch (1) is taken(2) not taken? Please give the states are taken for each. (For Code – 1)

We Need to determine the number of states for the BGE instruction when:
1. Branch is taken
2. Branch is not taken

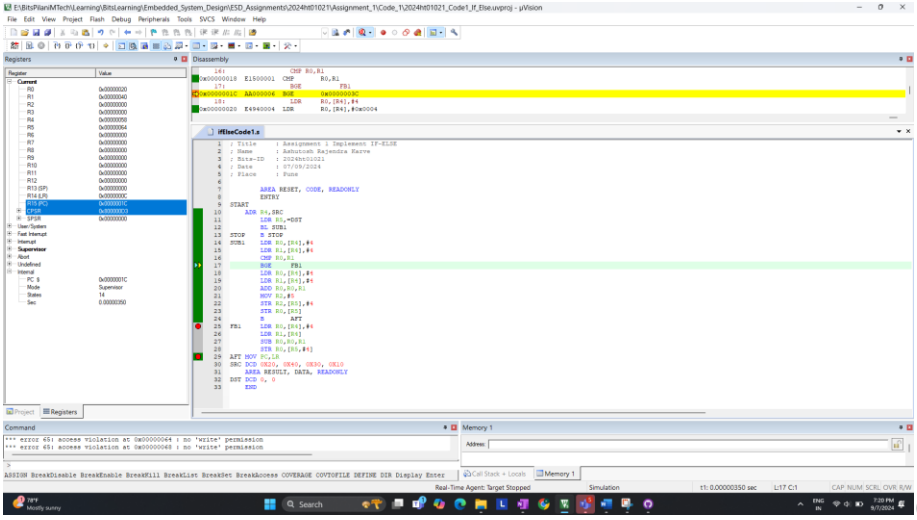
1. Branch Is Taken >>

code: SRC DCD 0x20, 0x40, 0x30, 0x10 (R0 < R1) (0x20 < 0x40)



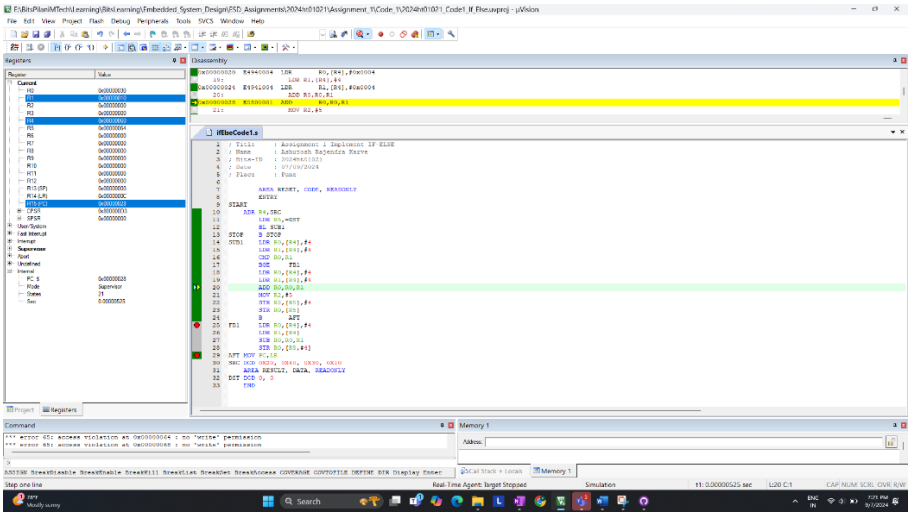
(Figure 1.7)

After comparison, for now R0 is loaded with 20 & R1 is Loaded with 40; States 13



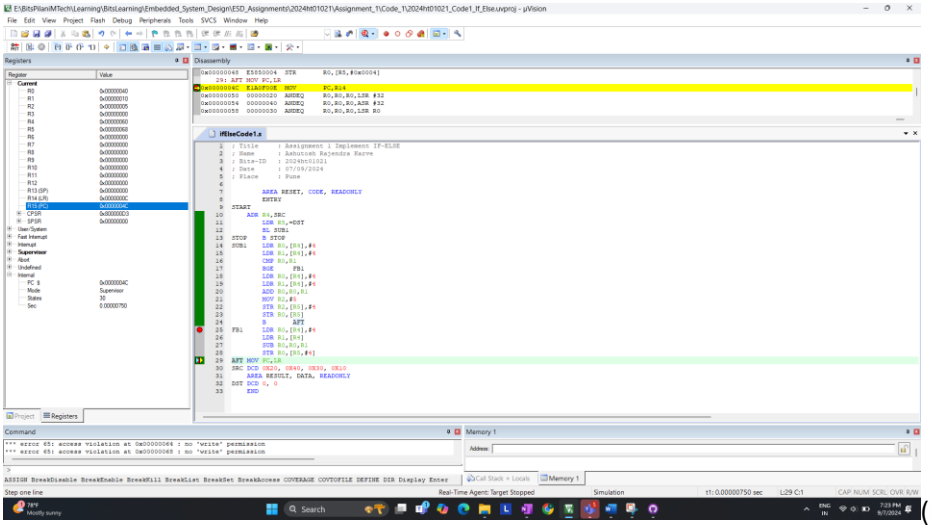
(Figure 1.8)

Now R0 and R1 is loaded with 30 & 10 after CMP we are in same block



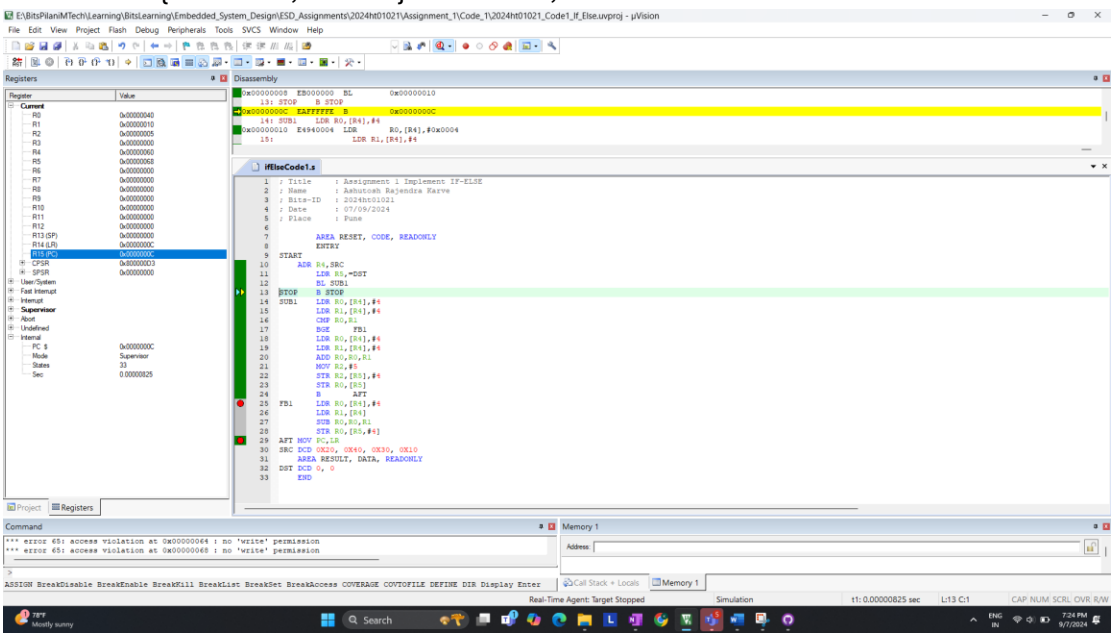
(Figure 1.9)

We are in {if loop} since (R0 < R1) new values update for R0 is 40 and R2 is 5



(Figure 2.0)

which is our {If Block X , Y Value} If is taken ; State : 30



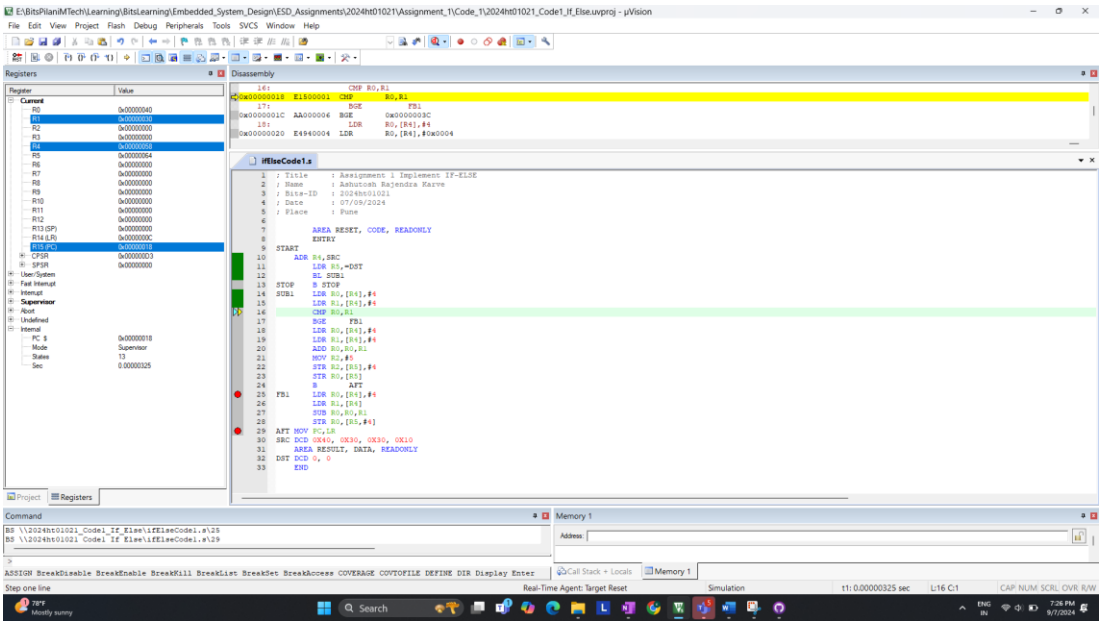
(Figure 2.1)

Answer: Final State with **Branch is Taken is 33**

2. Branch Is Not Taken:

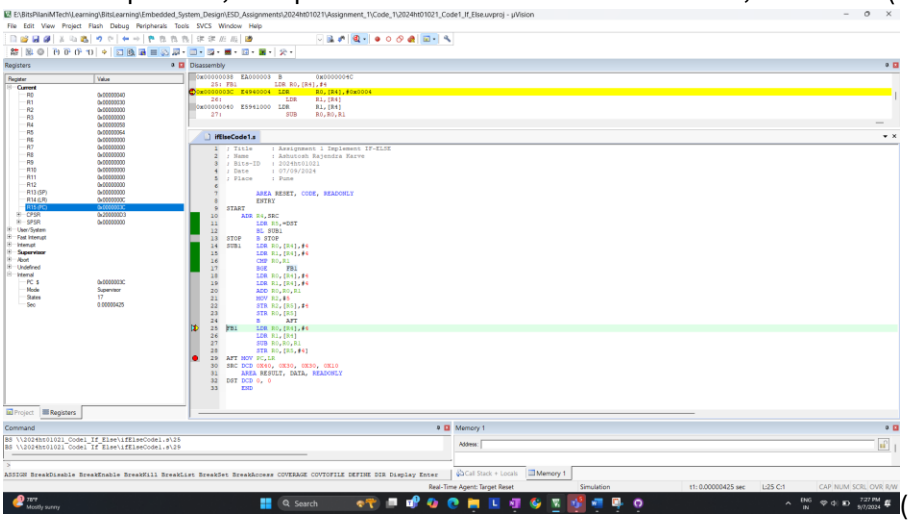
I. code: SRC DCD 0x40, 0x30, 0x30, 0x10 (R0 > R1)

For now value of R0 & R1 is loaded with 40,30 (40 > 30) check (Figure 2.2)



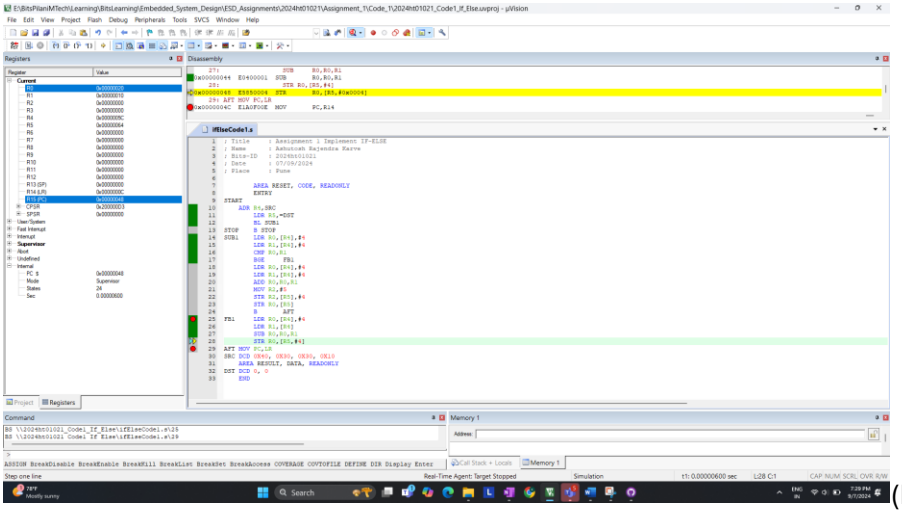
(Figure 2.2)

After comparison, it skip the IF-BLOCK enters ELSE-BLOCK; States 17 (check fig 2.3 line 25)

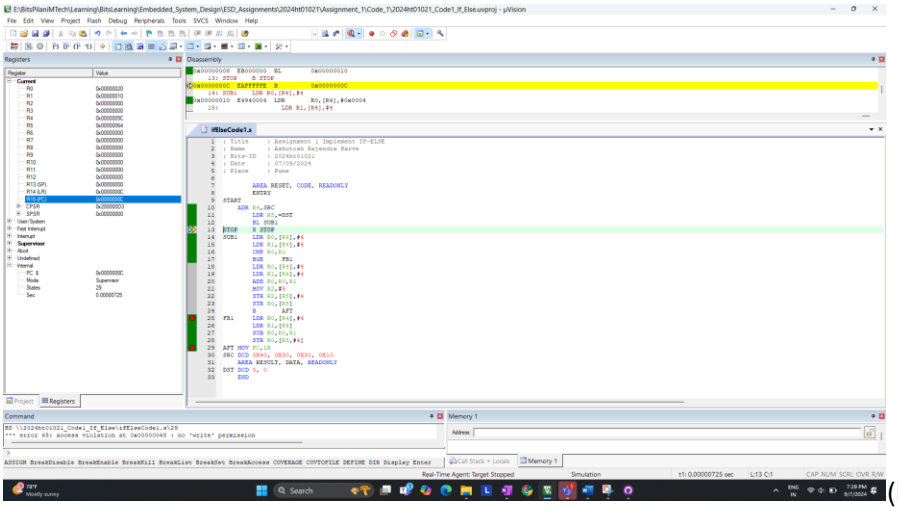


(Figure 2.3)

After Subtraction operation; Value of R0 Changes to 20; State 24



(Figure 2.4)

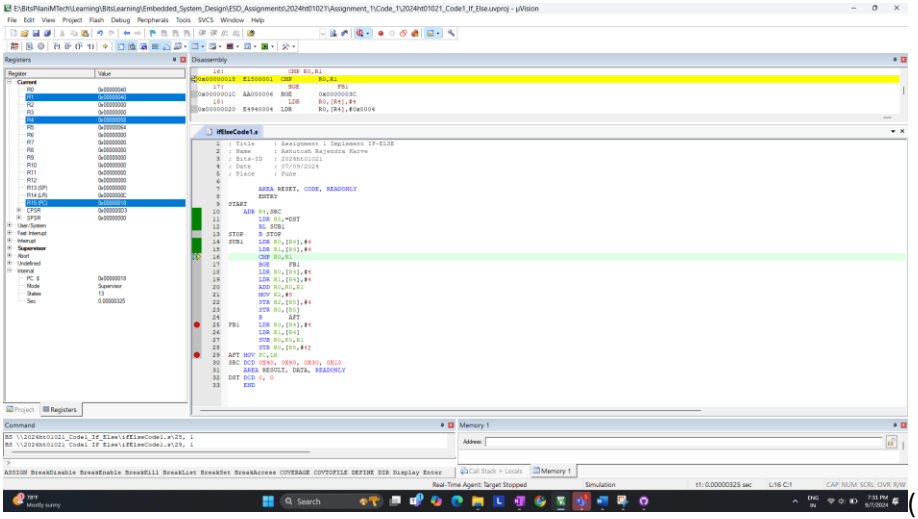


(Figure 2.5)

Answer: Final State with **Branch is not Taken is 29. Check Figure 2.5**

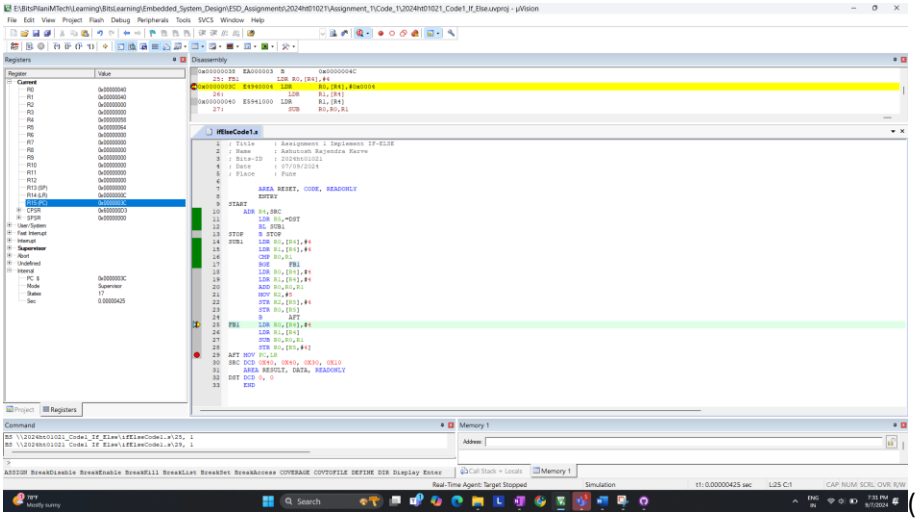
II. code: SRC DCD 0x40, 0x40, 0x30, 0x10 (R0 == R1)

For now R0 & R1 is loaded with 40 & 40 (40 == 40)

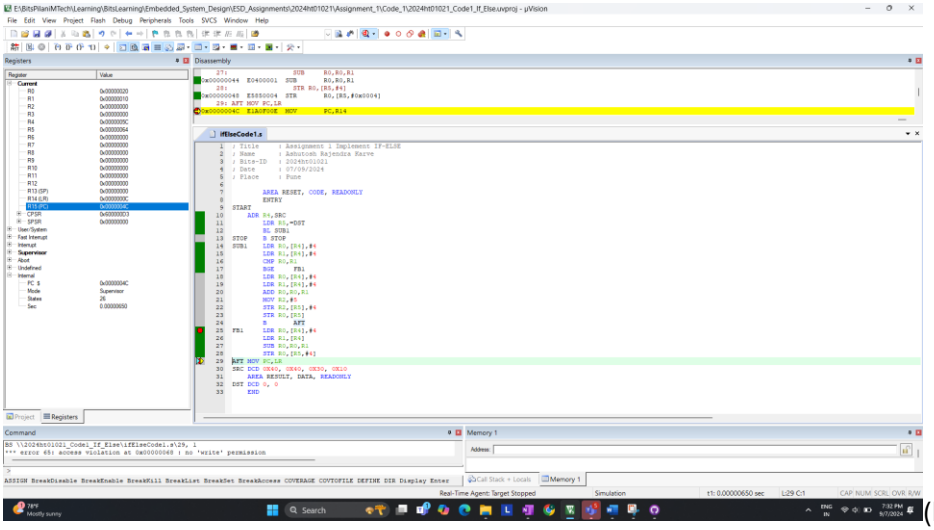


(Figure 2.6)

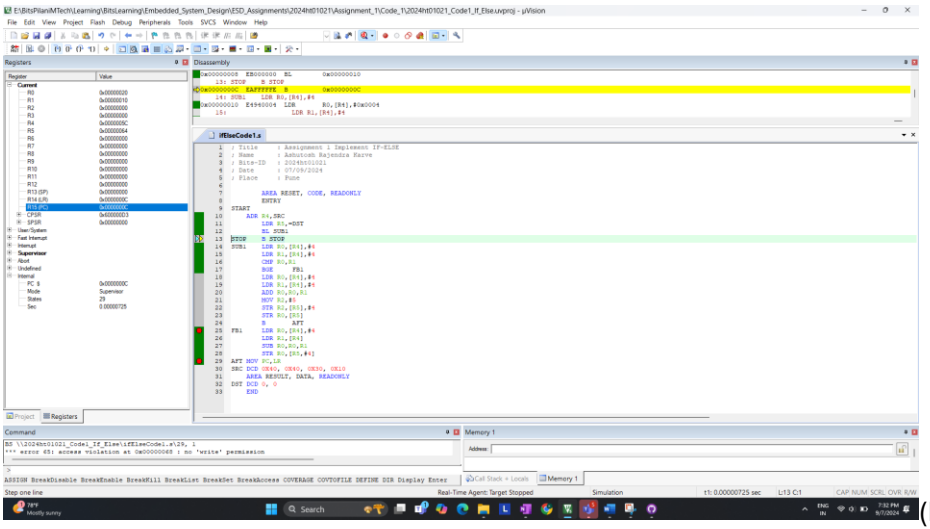
After comparison; it skip the IF-BLOCK enters ELSE-BLOCK; States 17



(Figure 2.7)



(Figure 2.8)



(Figure 2.9)

Final State with **Branch is not Taken is 29**

Answer:

- **Branch Taken:** The instruction **takes 33** cycles when the branch is taken.
- **Branch not Taken:** The instruction **takes 29** cycles when the branch is not taken.
- **Number of states** for completion is **not same** for BGE instruction

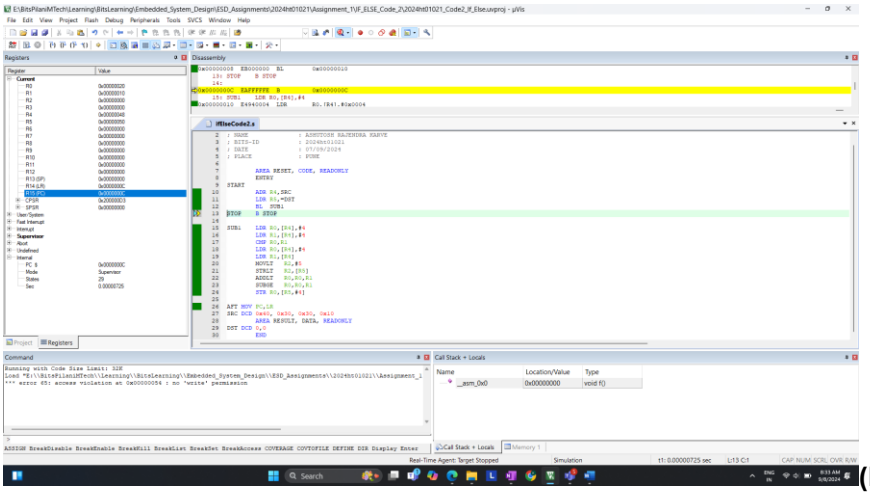
++++++++++++++++++++++++++END of **Q.1 c)**++++++++++++++++++++++++++++++

d. Measure the performance of code-1 and code-2 for the following conditions

Condition	Code-1-States	Code-2-States
a<b	33	30
a>b	29	29
a=b	29	29

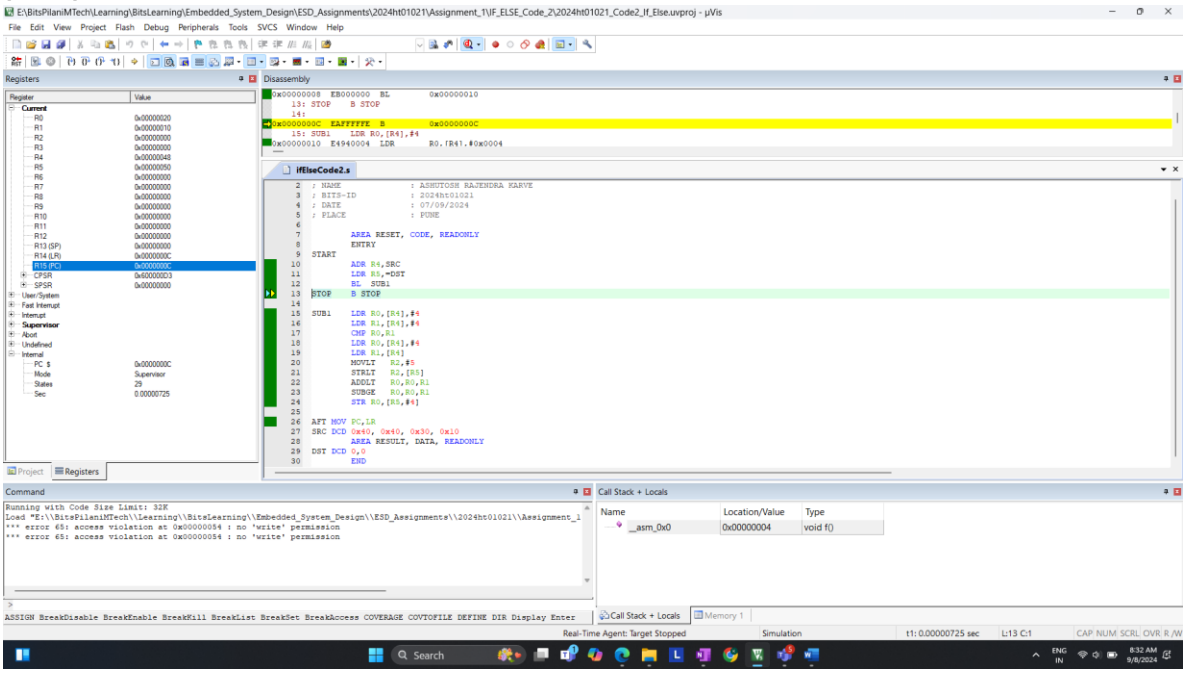
Screen Shots of Code-2-States

(a > b) : SRC DCD 0x40, 0x30,0x30,0x10 States is 29



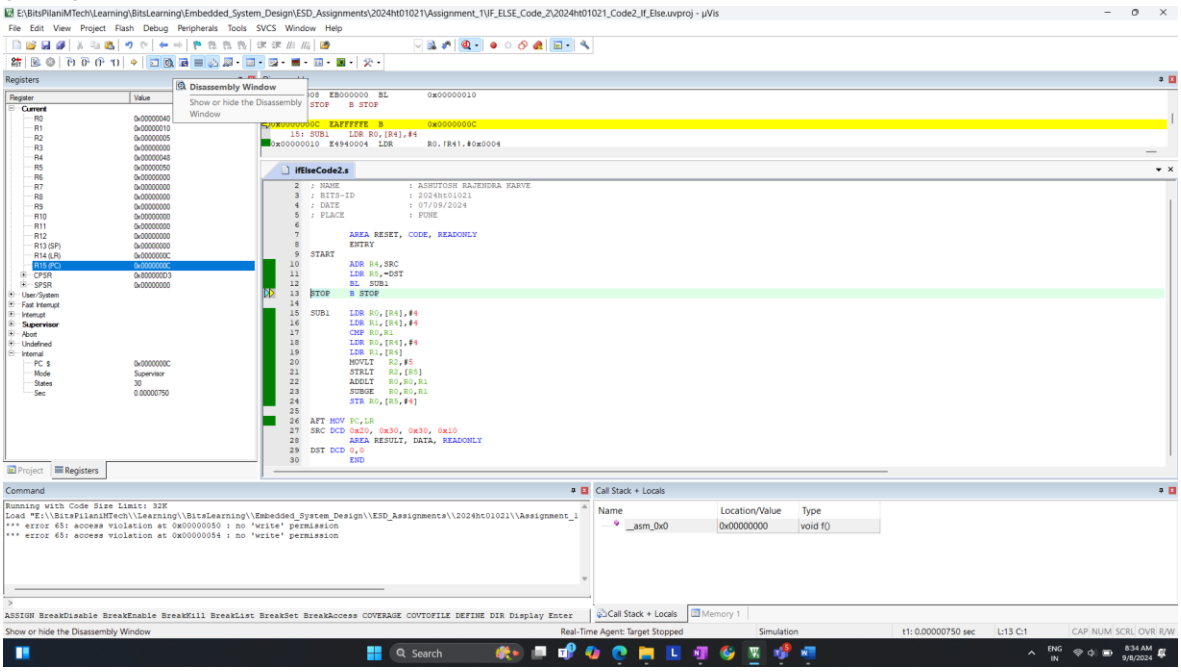
(Figure 3.0)

(a = b) : SRC DCD 0x40, 0x40,0x30,0x10 States is 29



(Figure 3.1)

(a < b) : SRC DCD 0x20, 0x30,0x30,0x10 States is 30

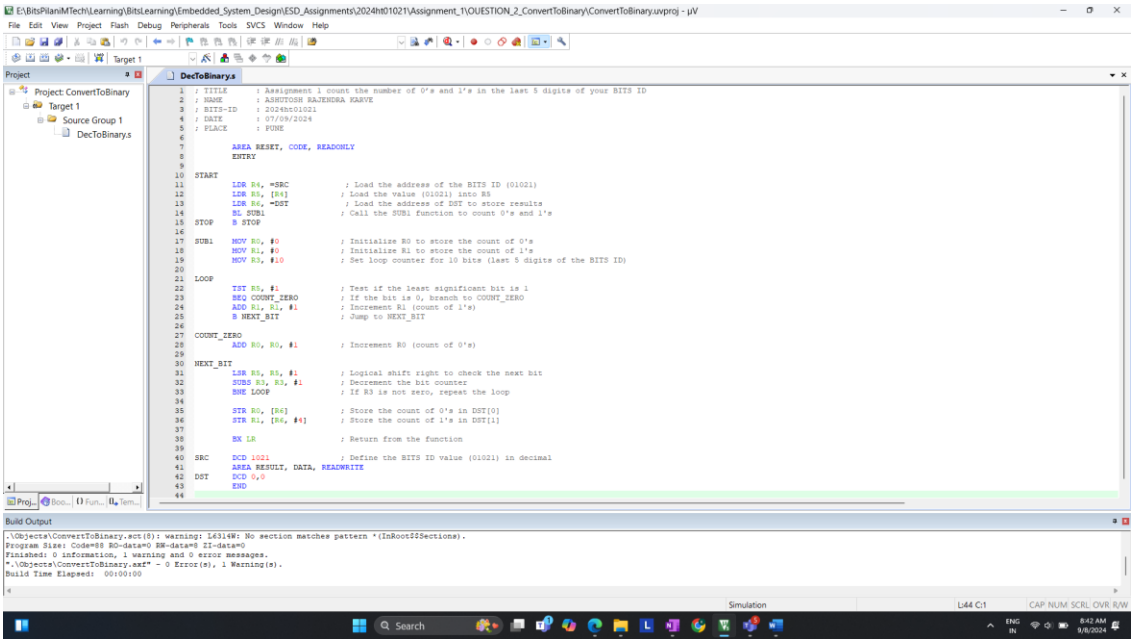


(Figure 3.2)

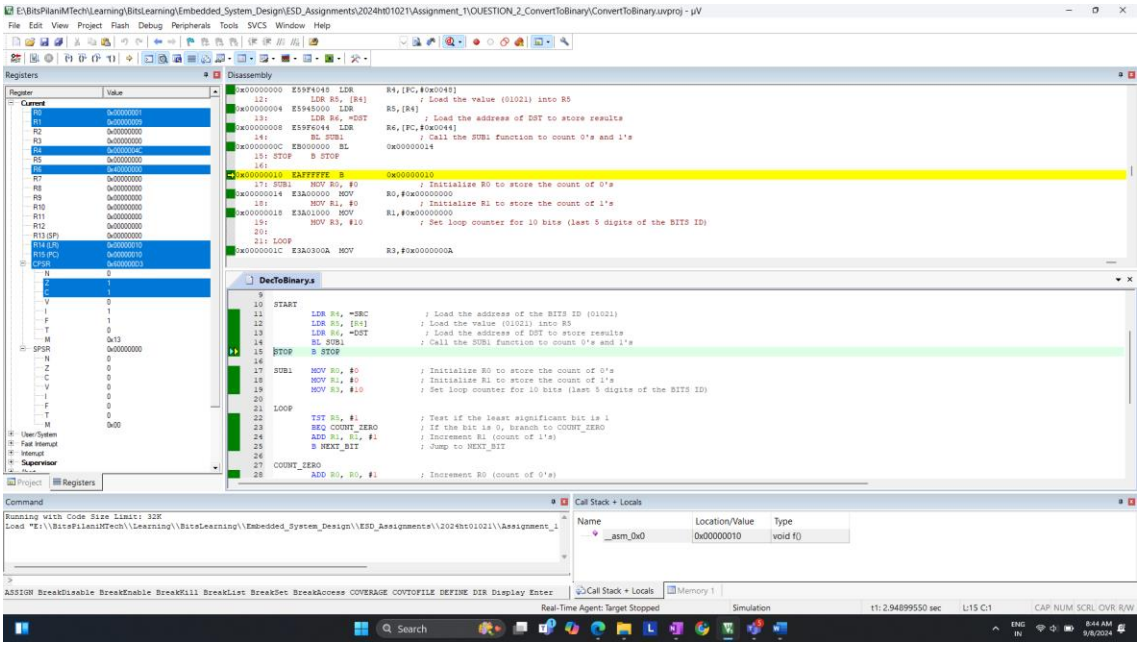
++++++++++++++++++++++++END of **Q.1 d)**++++++++++++++++++++++++++++++

Q.2. Write an assembly language program for LPC2378 to count the number of 0's and 1's in the last 5 digits of your BITS ID (number must be given in decimal number format). Store the number of 0's in register R0 and the number of 1's in register R1. Verify your result by performing manual calculations.

STEP 1 : Code compilation with errors.



Count of 0's & 1's in memory after the execution; Check R0 & R1



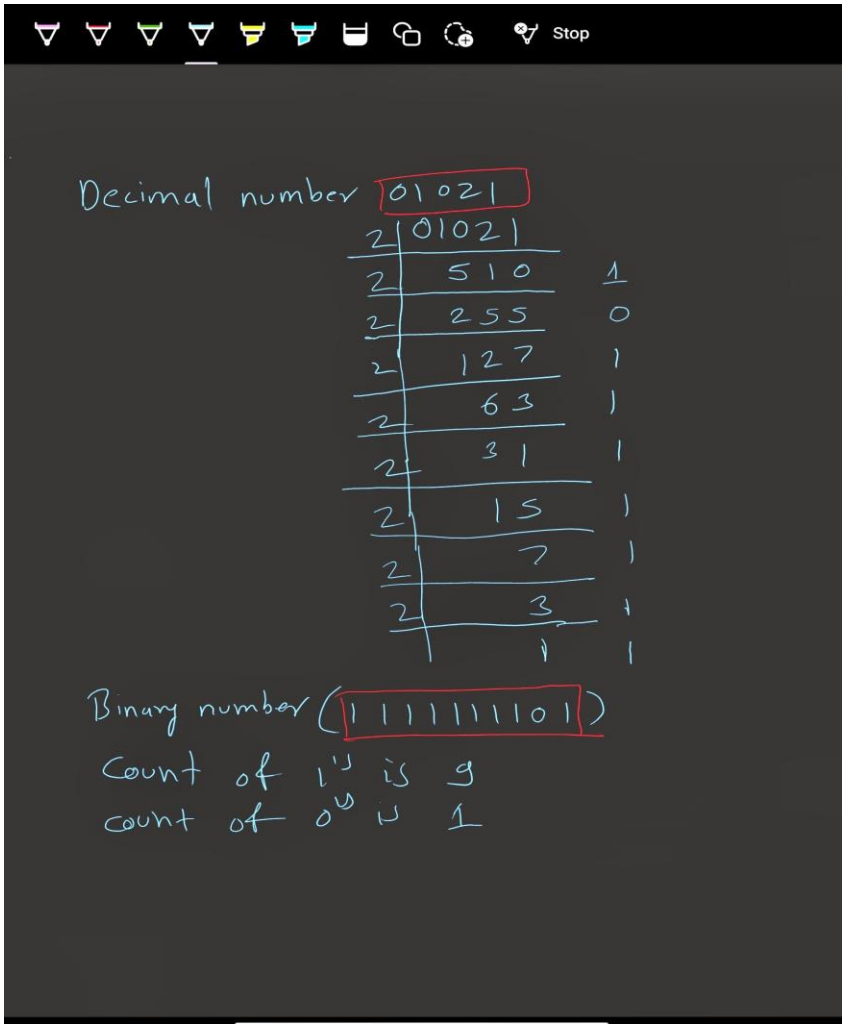
Verification:

1. Check the register value is updated

- R0: This register will contain the count of 0's
- R1: This register will contain the count of 1's

2. Manual Calculation

- BITS ID**: 01021**
- Binary equivalent of the last 5 digits (1021): **1111111101**
- Count of 1's: 9
- Count of 0's: 1



++++++++++++++++++++++++END of **Q.2)**++++++++++++++++++++++++++++

Q.3. Write an assembly language program for ARM Cortex M3/4 to handle Supervisor Call (SVC) exception.

Requirements:

1. The SVC should be called from an application task running at **Thread unprivileged mode.** The SVC number is derived from the last two digits of my BITS ID, i.e., **21**.

2. Two parameters, representing the last five digits of my BITS ID (**01021**), will be passed to the **SVC handler**.

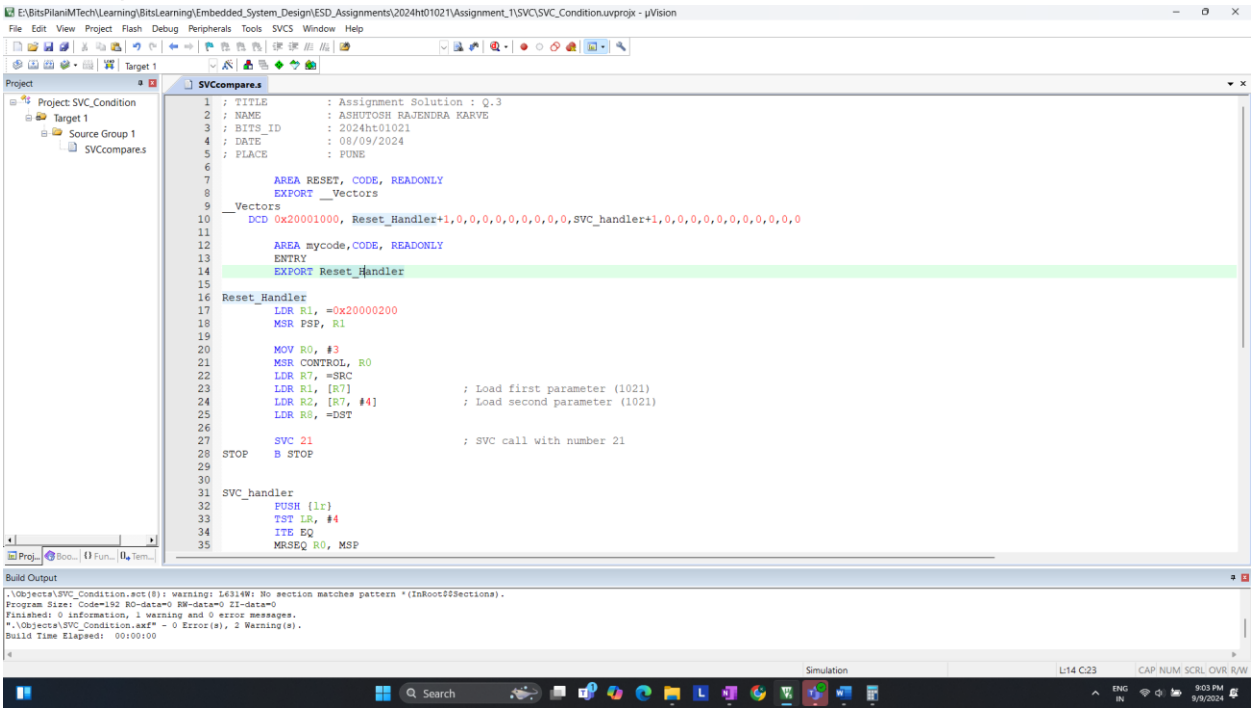
3. The SVC number will be dynamically determined by examining the SVC instruction in the memory.

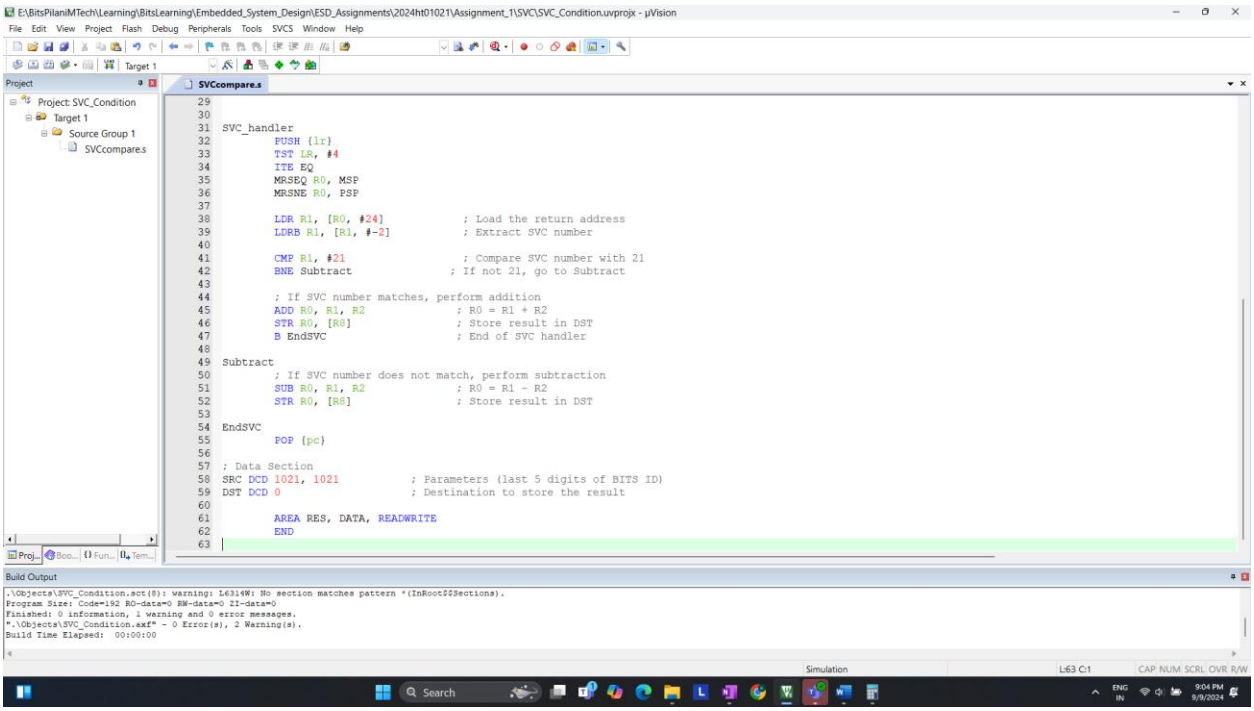
4. The SVC exception handler should set up a stack pointer in SVC mode before handling the exception.

5. If the SVC number matches my BITS ID (21), the handler will perform the addition of the two parameters.

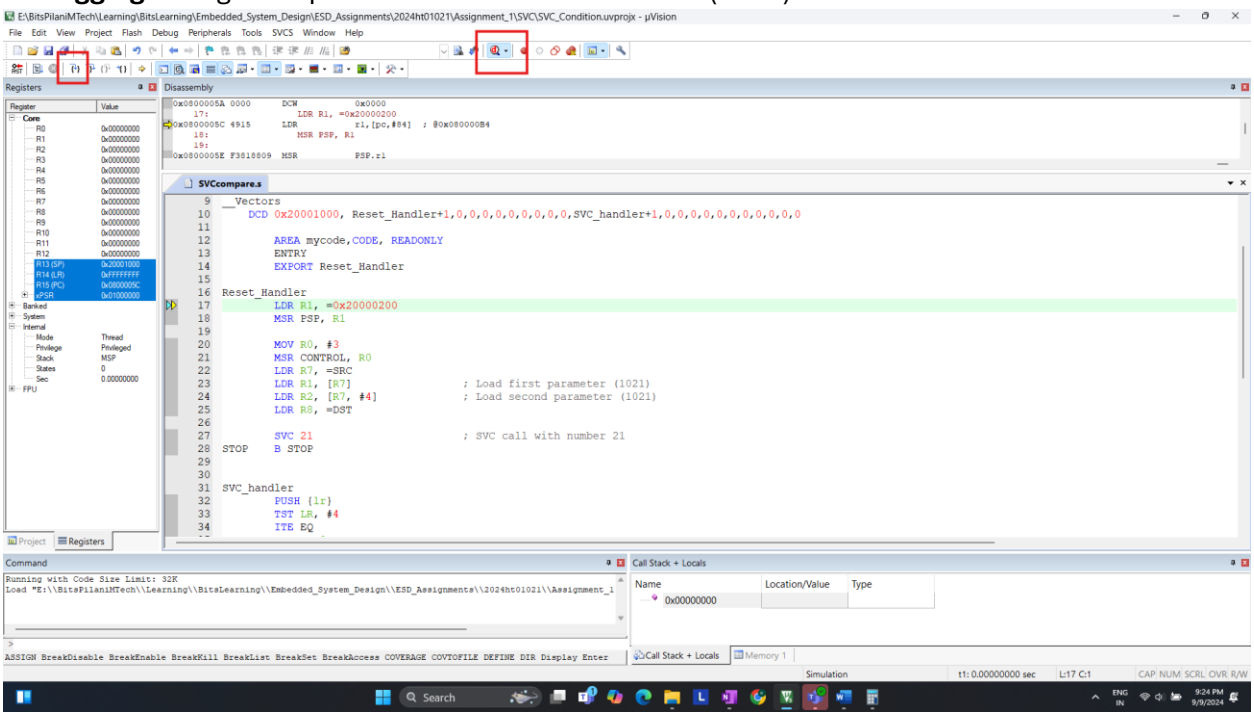
6. Otherwise, it will perform subtraction.

Assembly Code: Screen shot without errors



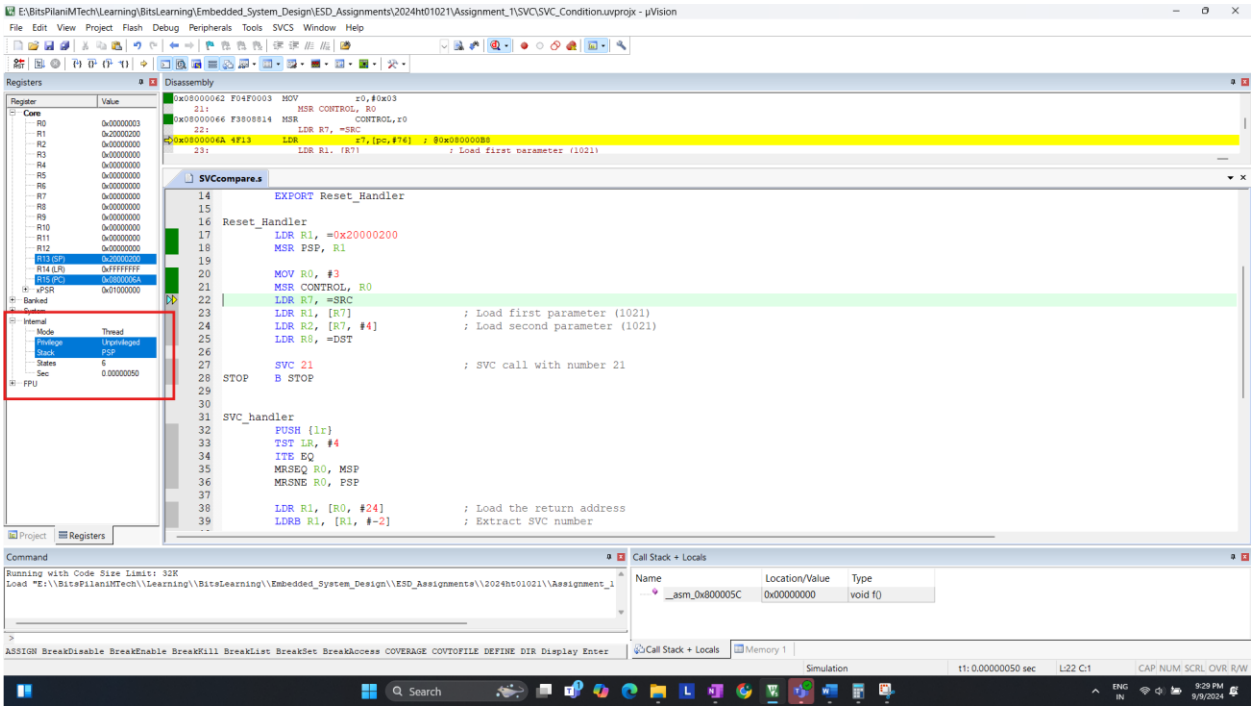


Debugging: Using of Step one line to show execution >> (FIG 1)



(FIGURE 1)

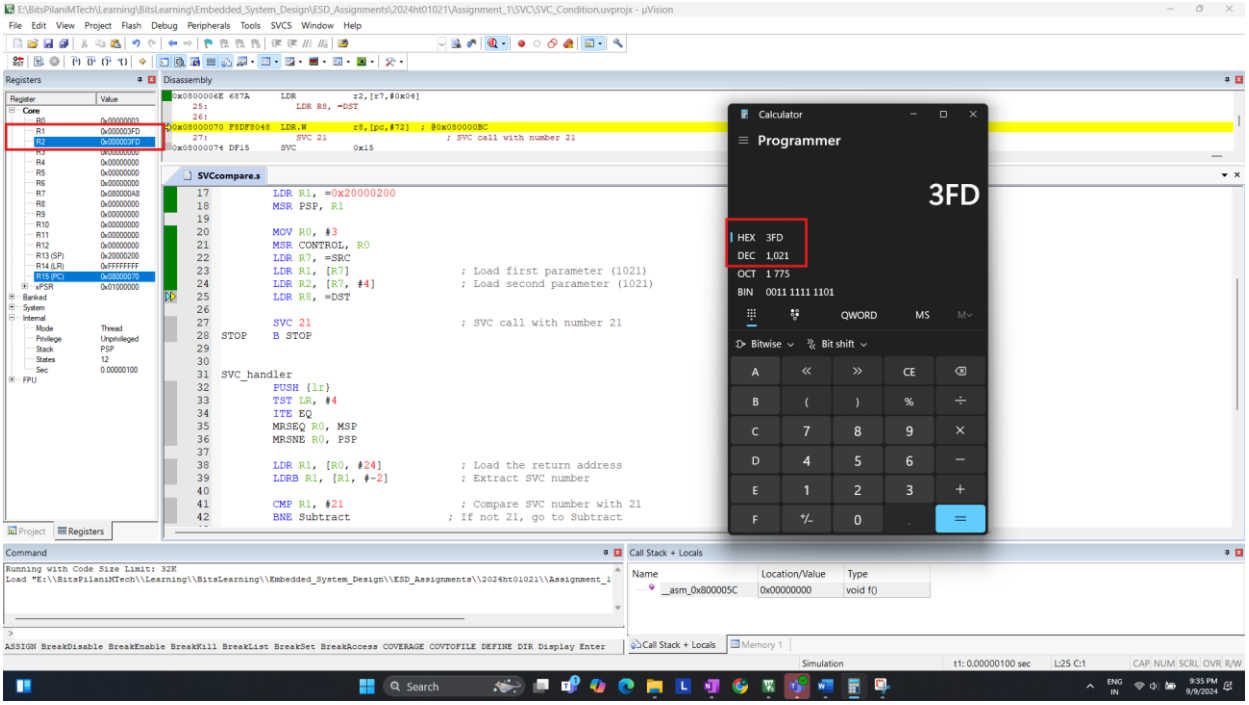
STEP 1: Setting Processor to Thread Unprivileged Mode >> (FIG 2)



(FIGURE 2)

We need to ensure that the processor is running in **Thread unprivileged mode**. This is achieved by writing to the **control** register to change the mode

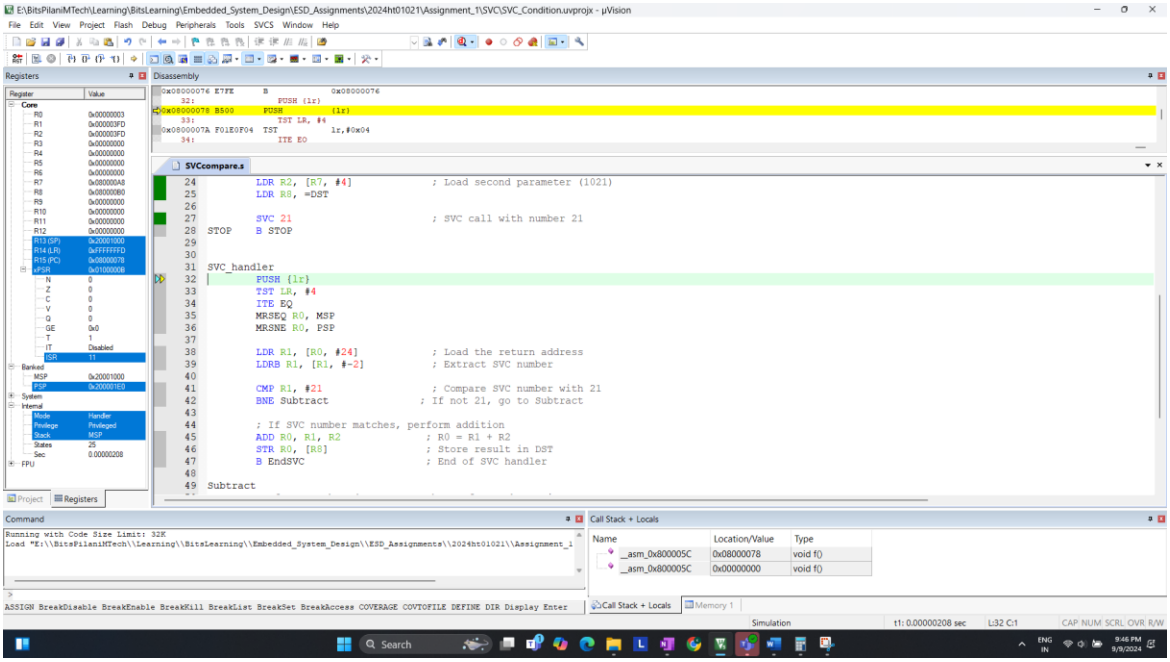
STEP 2: Loading Parameters (Last 5 digits of BITS ID) >> (FIG 3)



(FIGURE 3)

We **load two parameters** from memory, both of which are derived from the last 5 digits of my BITS ID **(1021)**. These will be used in the SVC handler.

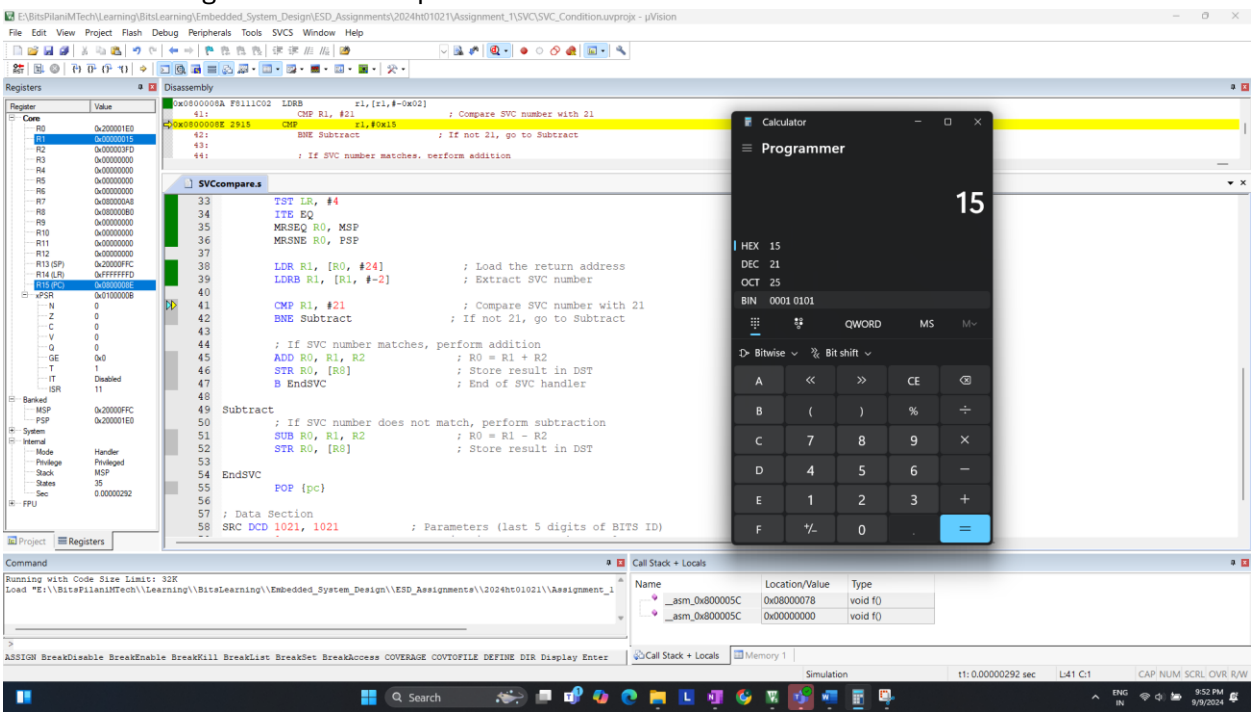
STEP 3: Triggering the SVC Exception >> (FIG 4)



(FIGURE 4)

We trigger the **SVC** exception using **SVC instruction**, passing **SVC Number**

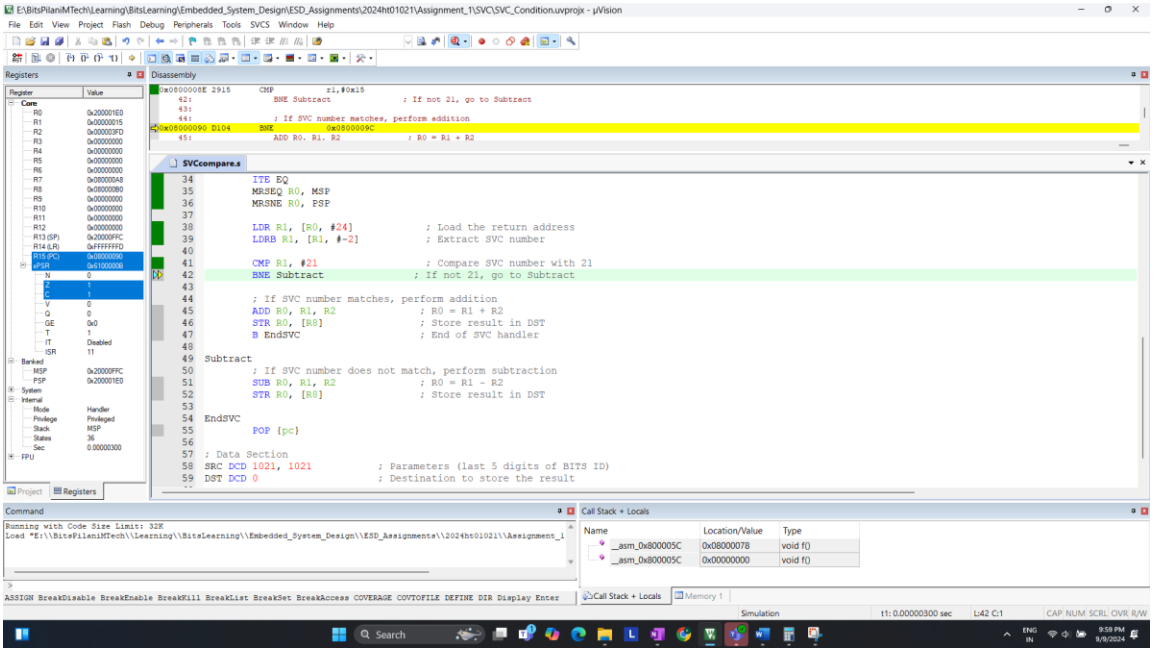
STEP 4: Handling SVC in the Exception Handler



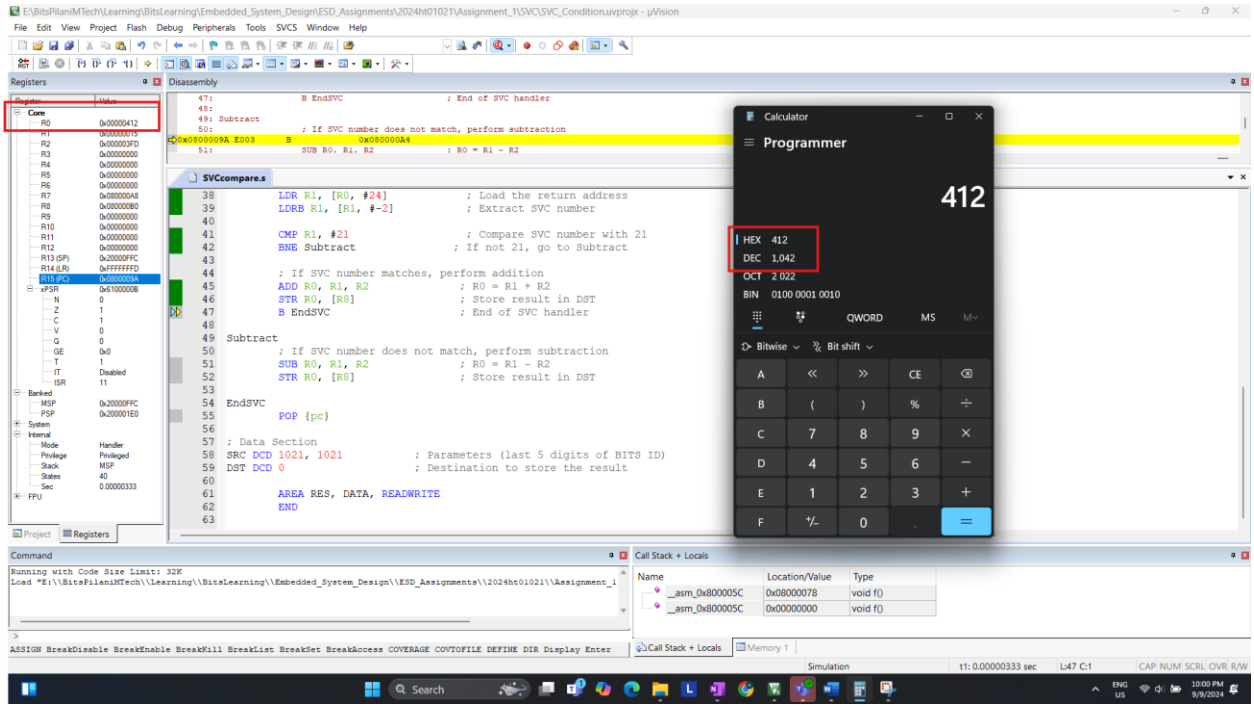
(FIGURE 5)

Inside the SVC handler, the program checks whether the SVC number matches 21. We can see R1 is loaded with 21 which will match and perform the **Addition** operation.

STEP 5: Operation after comparison >> (FIG 6) & (FIG 7)



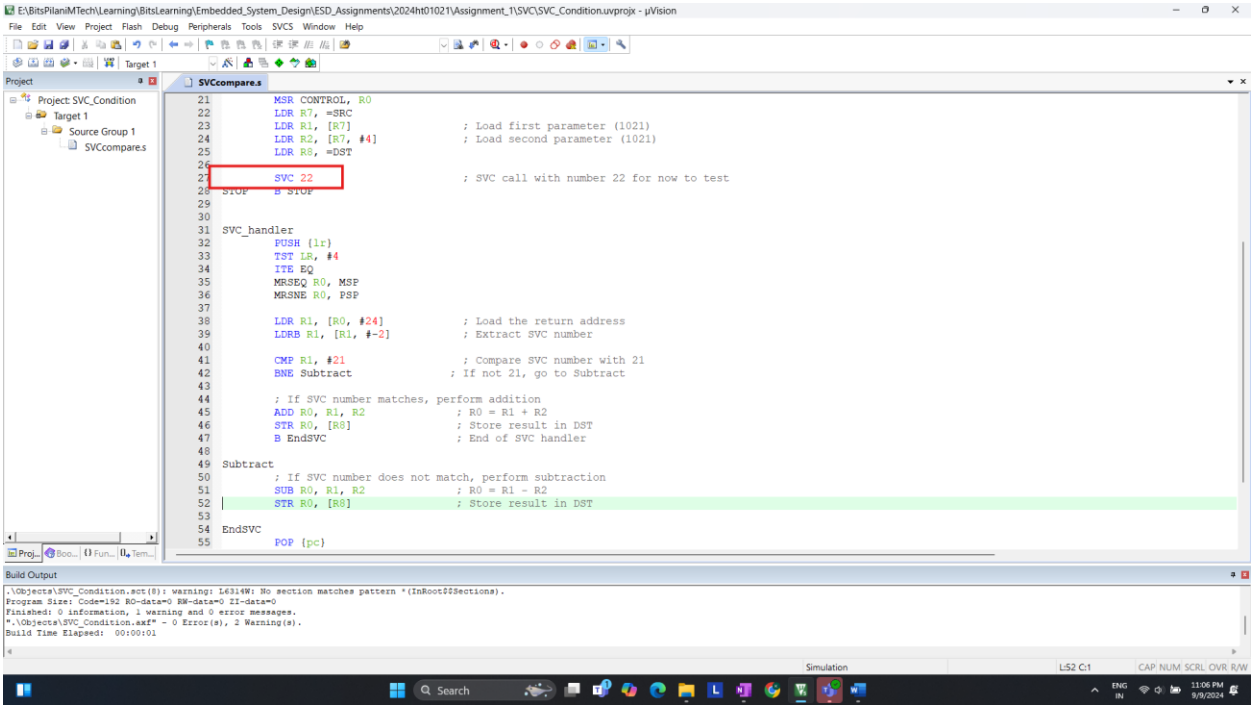
(FIGURE 6)



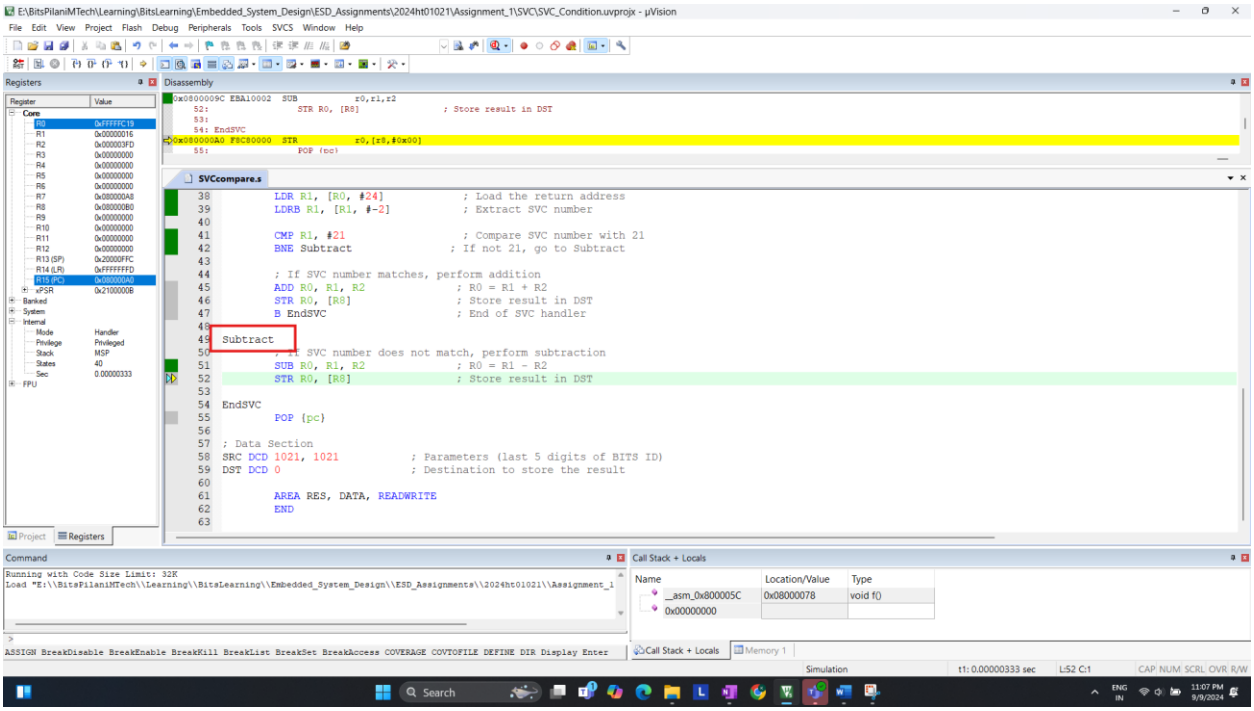
(FIGURE 7)

After executing the logic to extract the SVC number, SVC number (21) is compared, and the addition is performed because the SVC number matches.

ADDITION STEP TO VERIFY STEP 6: Test with different SVC number (Fig 8 & 9)



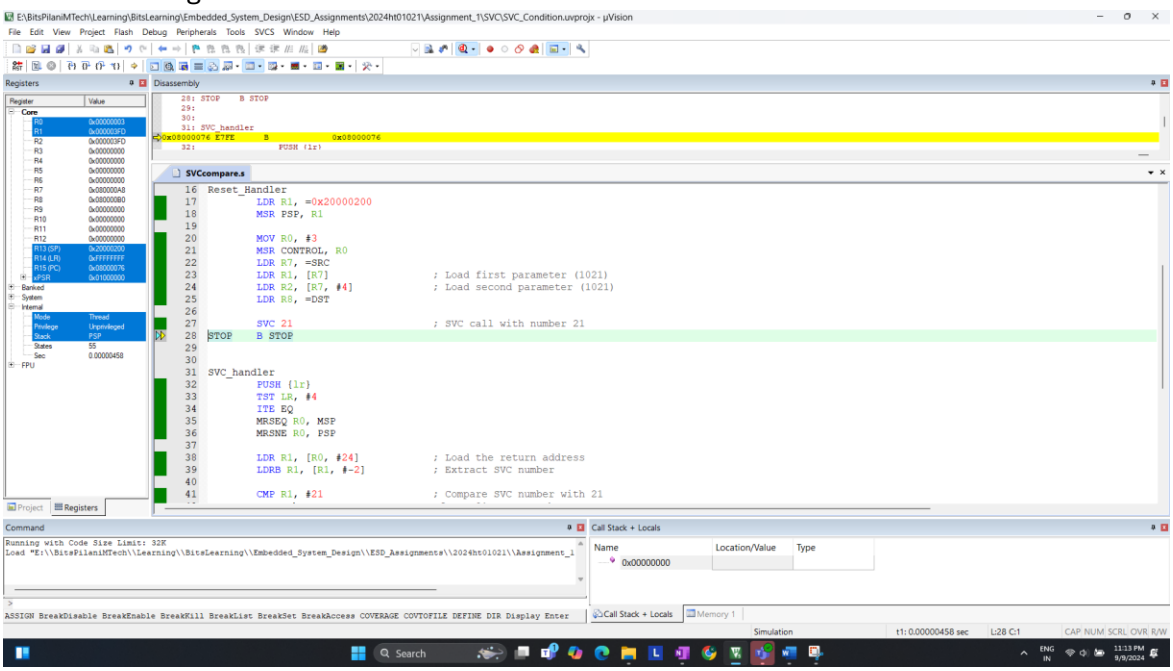
(FIGURE 8)



(FIGURE 9)

In Figure 9 we can observe that it is in subtract block, so logic is executed correctly.

STEP 7: Viewing Final Result



(FIGURE 10)

After completion the execution of the SVC handler, Final result stored in memory

++++++++++++++++++++++++++END of **Q.3)**++++++++++++++++++++++++++++++