**ESD Lab Assignment No.1**

Name : Ashutosh Rajendra karve

Bits-ID : 2024ht01021

Mail : [2024ht01021@wilp.bits-pilani.ac.in](mailto:2024ht01021@wilp.bits-pilani.ac.in)

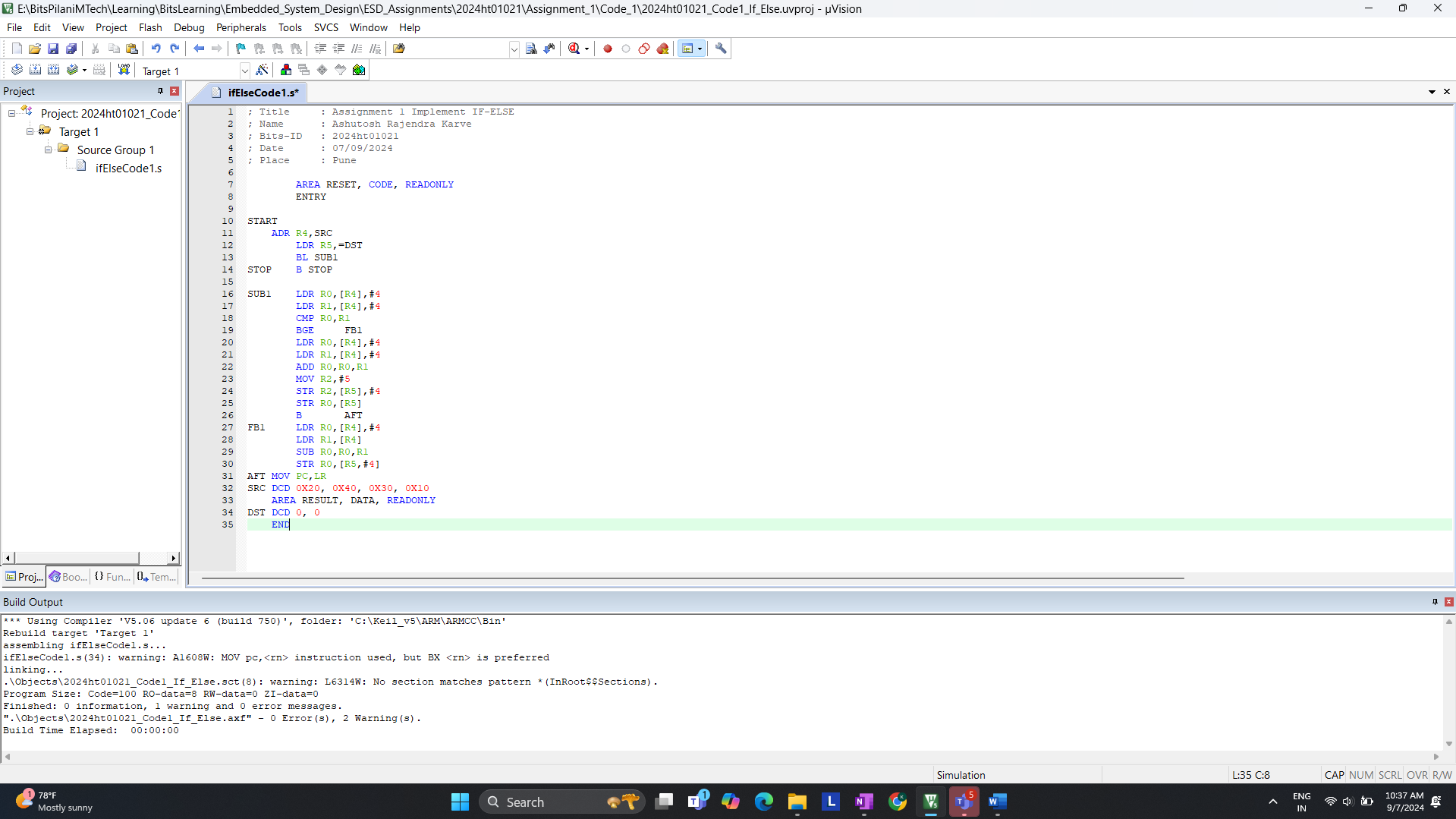
Contact : +91 9765541324

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 >>

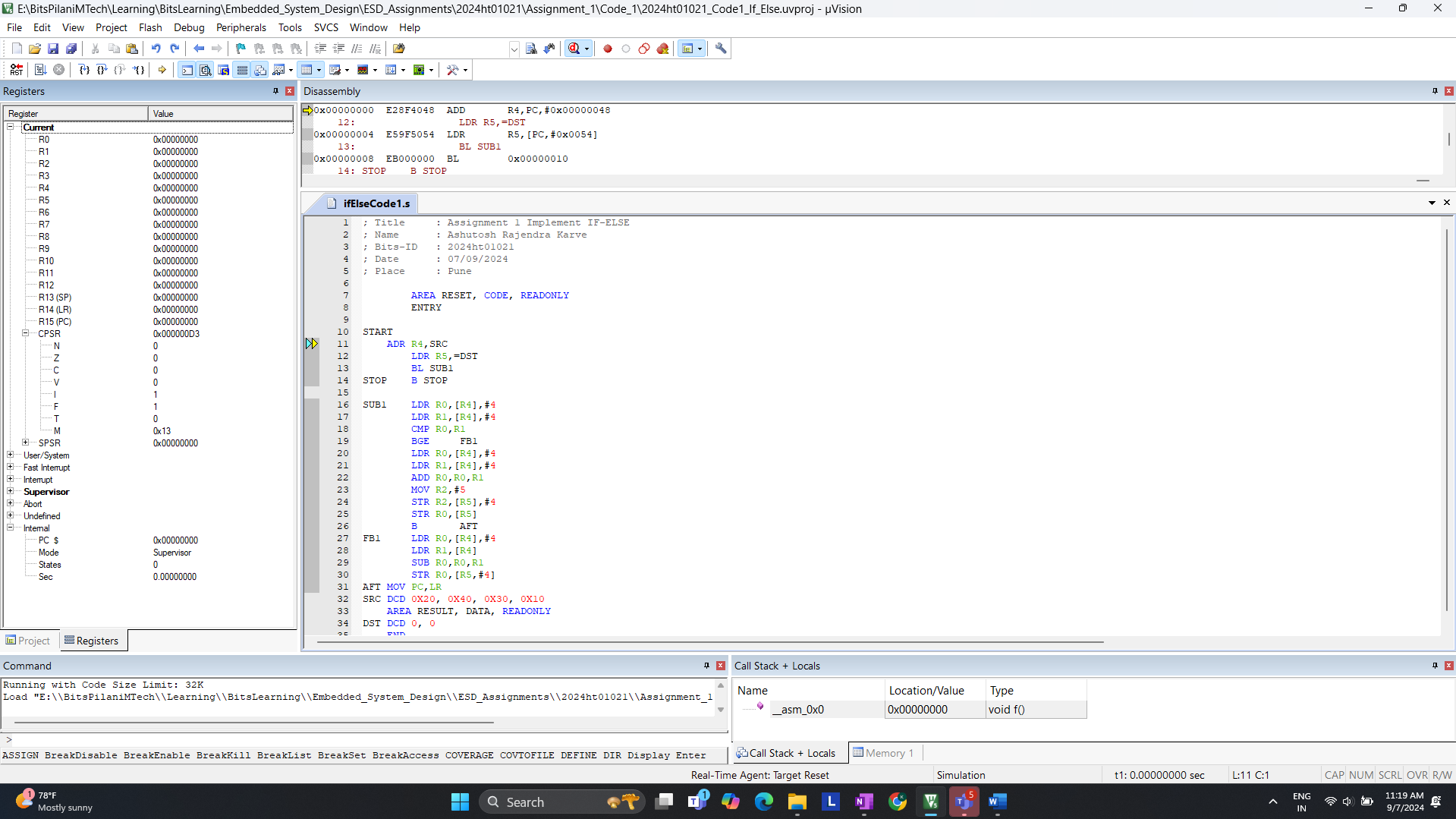


Code- 2 >>   
A screenshot of a computer

Description automatically generated

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

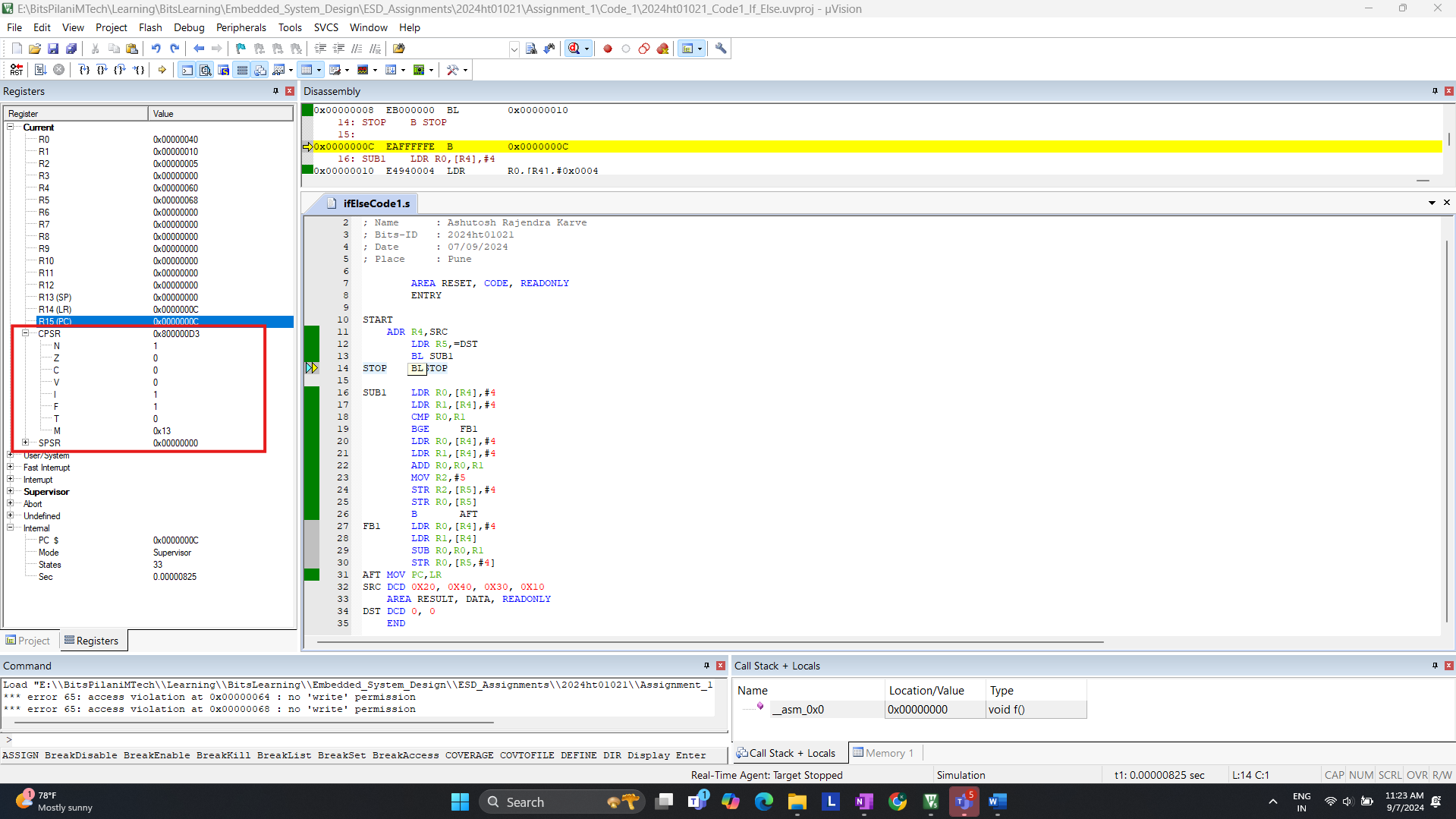
**CODE 1 >>**

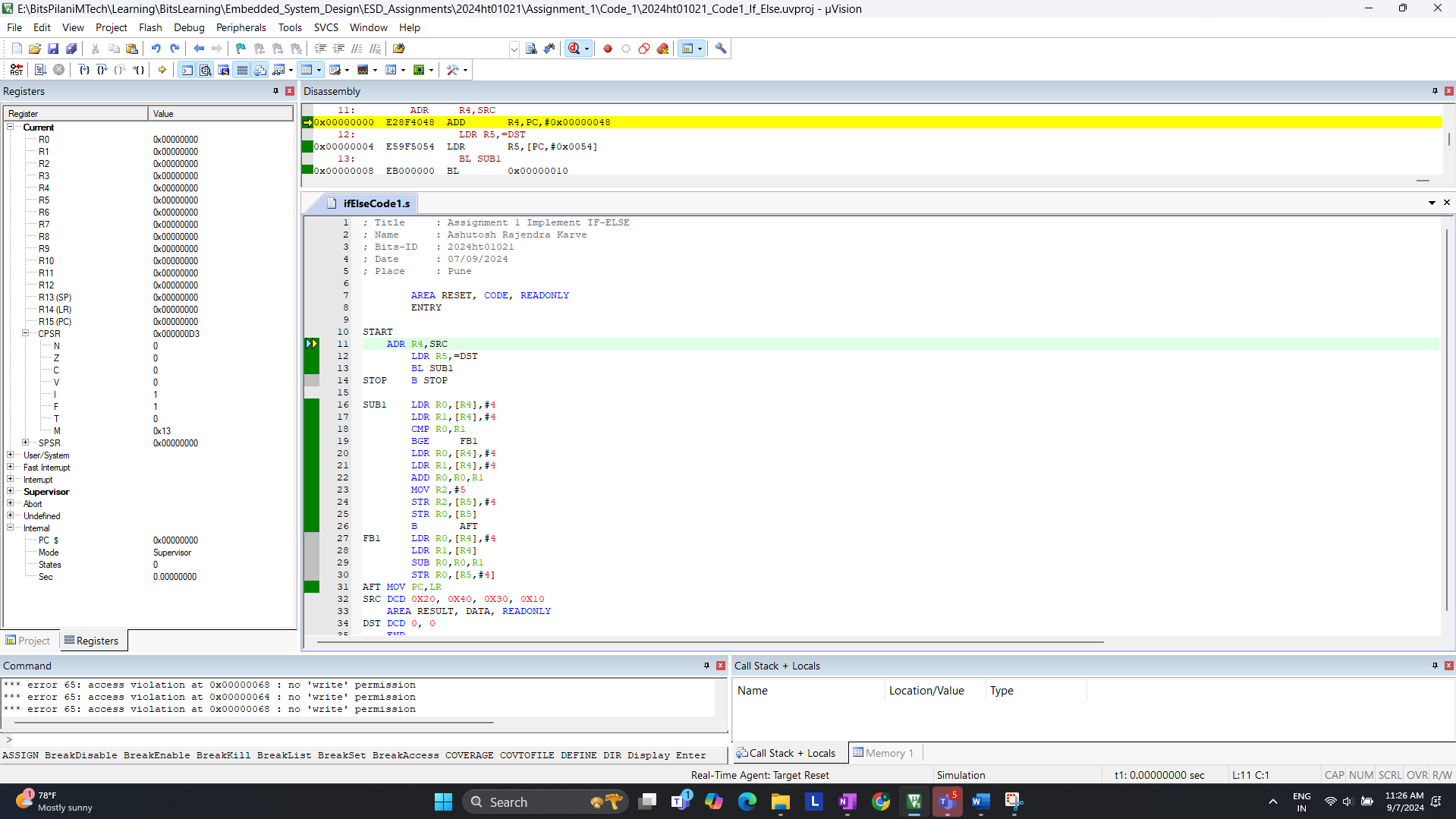
Start of debugging for code 1.  


After >> Step  
A computer screen shot of a computer

Description automatically generated

Observation of Code 1.





**Answer:**

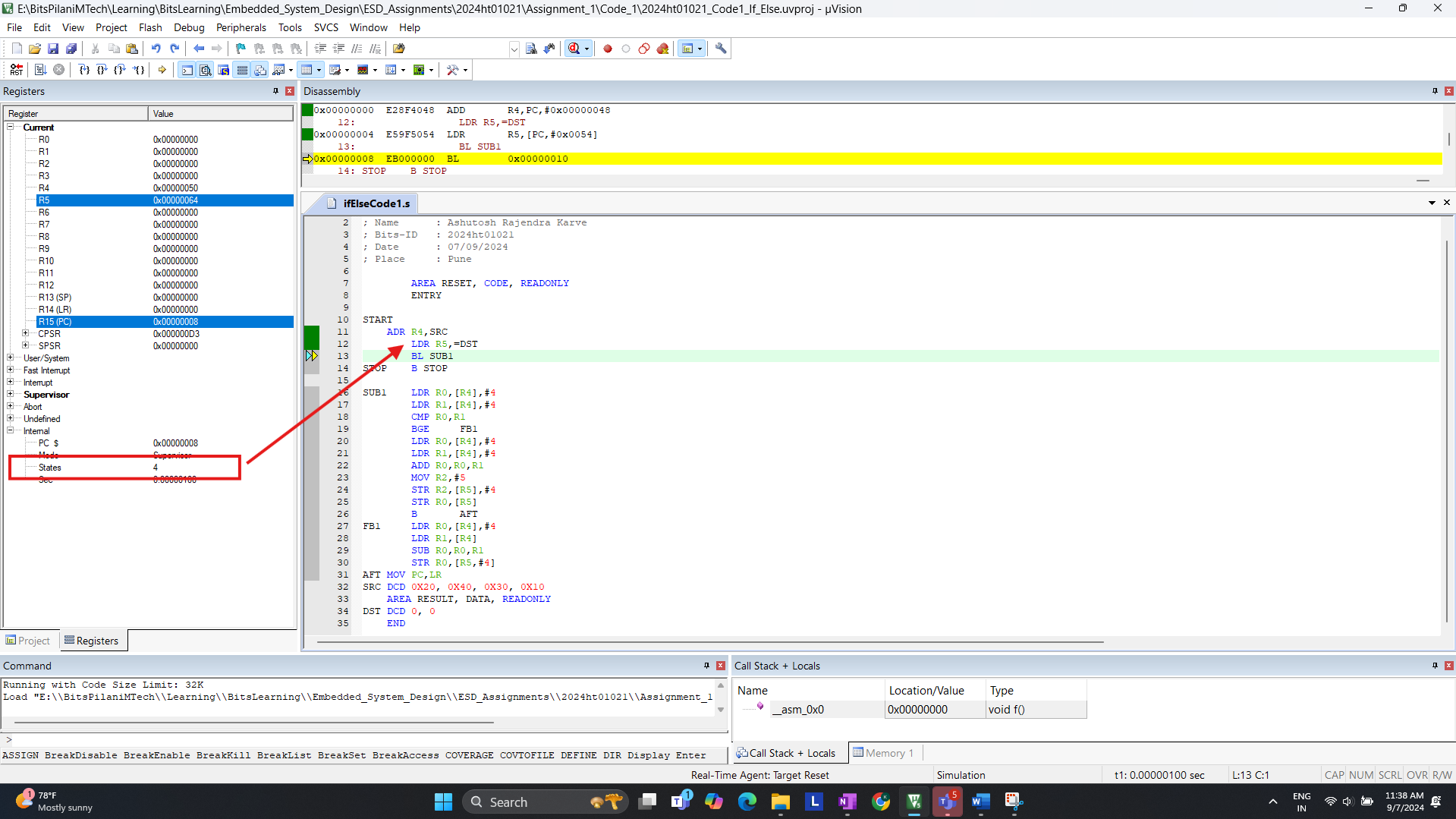
* Looking at CPSR. You will see that the processor is in **Supervisor Mode** after reset and **0 states**
* The Processor starts in **ARM State** (32-bit mode) on reset
* It takes total **33 states**

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

A screenshot of a computer

Description automatically generated

Example for LDR >> Code 1.

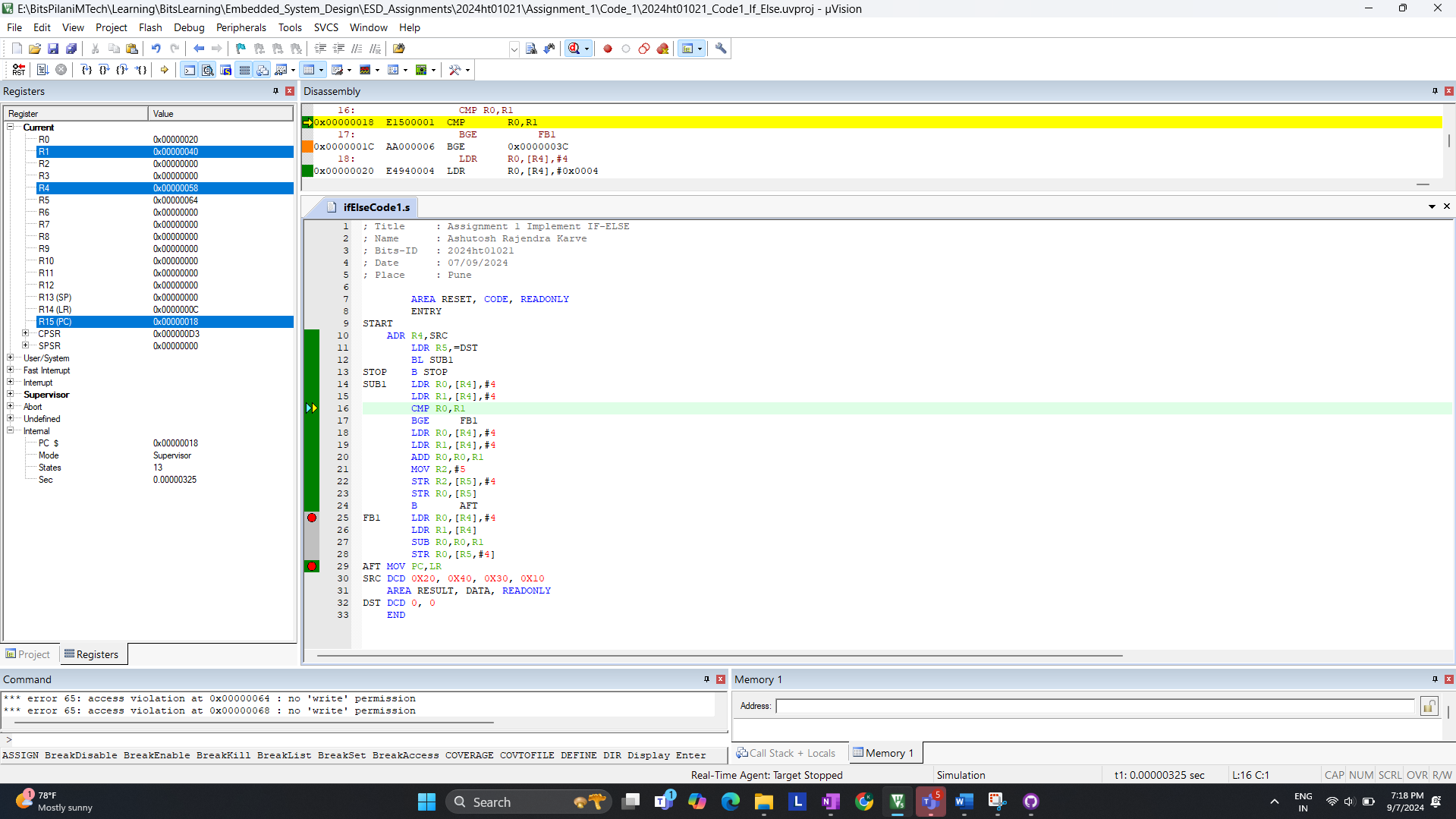


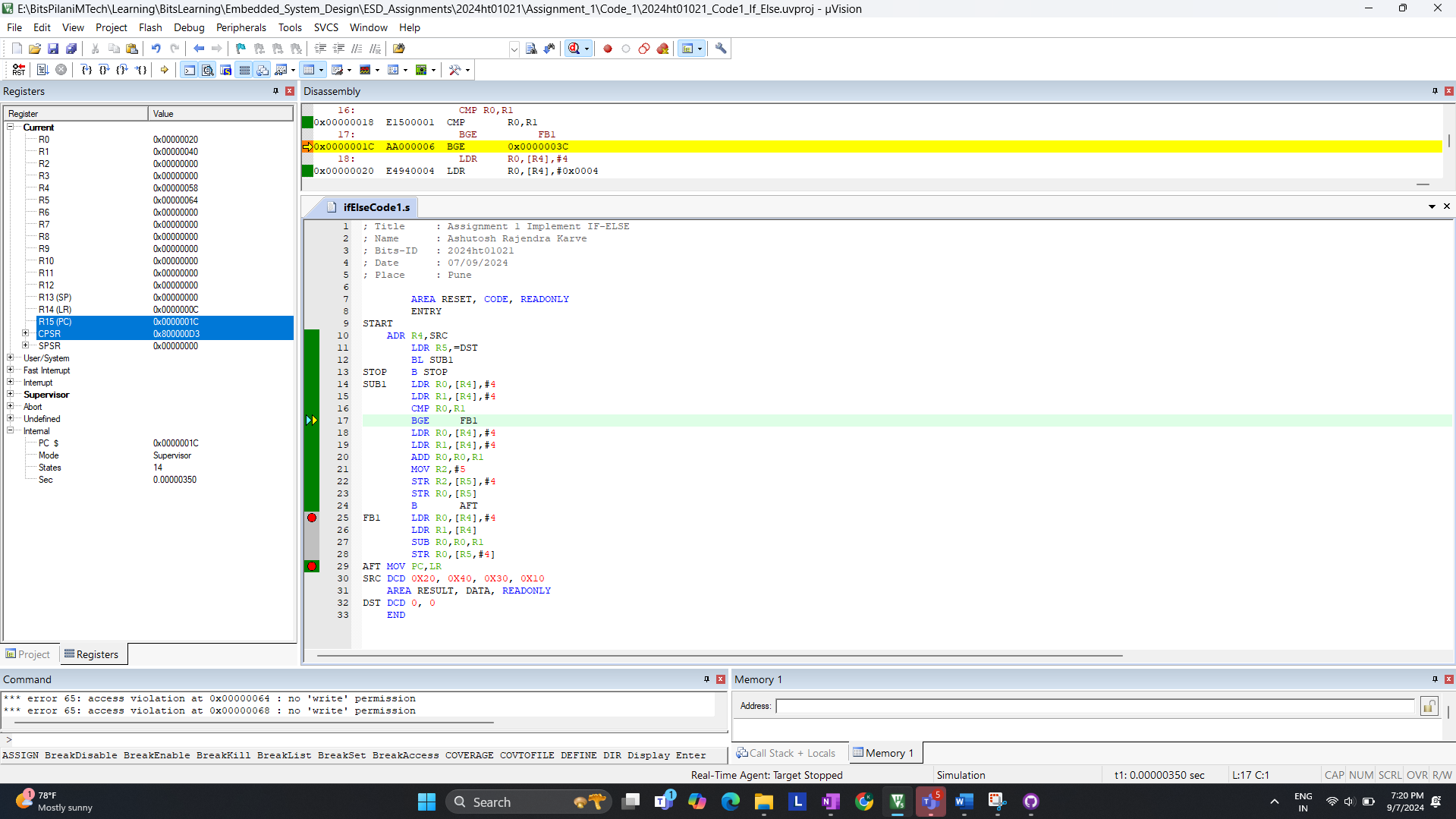
**Answer:**

* 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 observein the above screen shot.

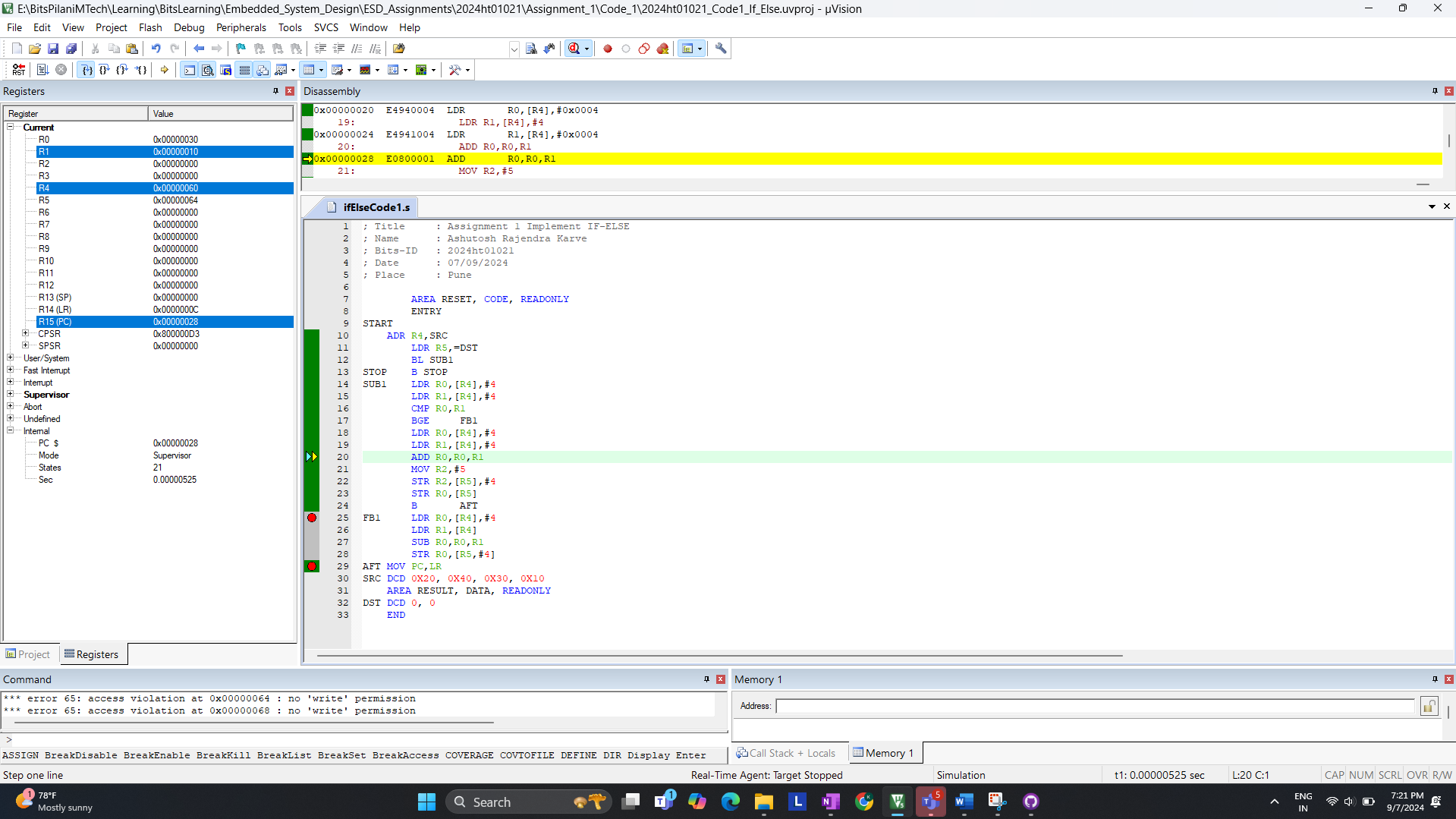
**c. Are the number of states taken for completion the same for BGE instruction if the branc (1) is taken(2) not taken? Please give the states are taken for each. (For Code – 1)**

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

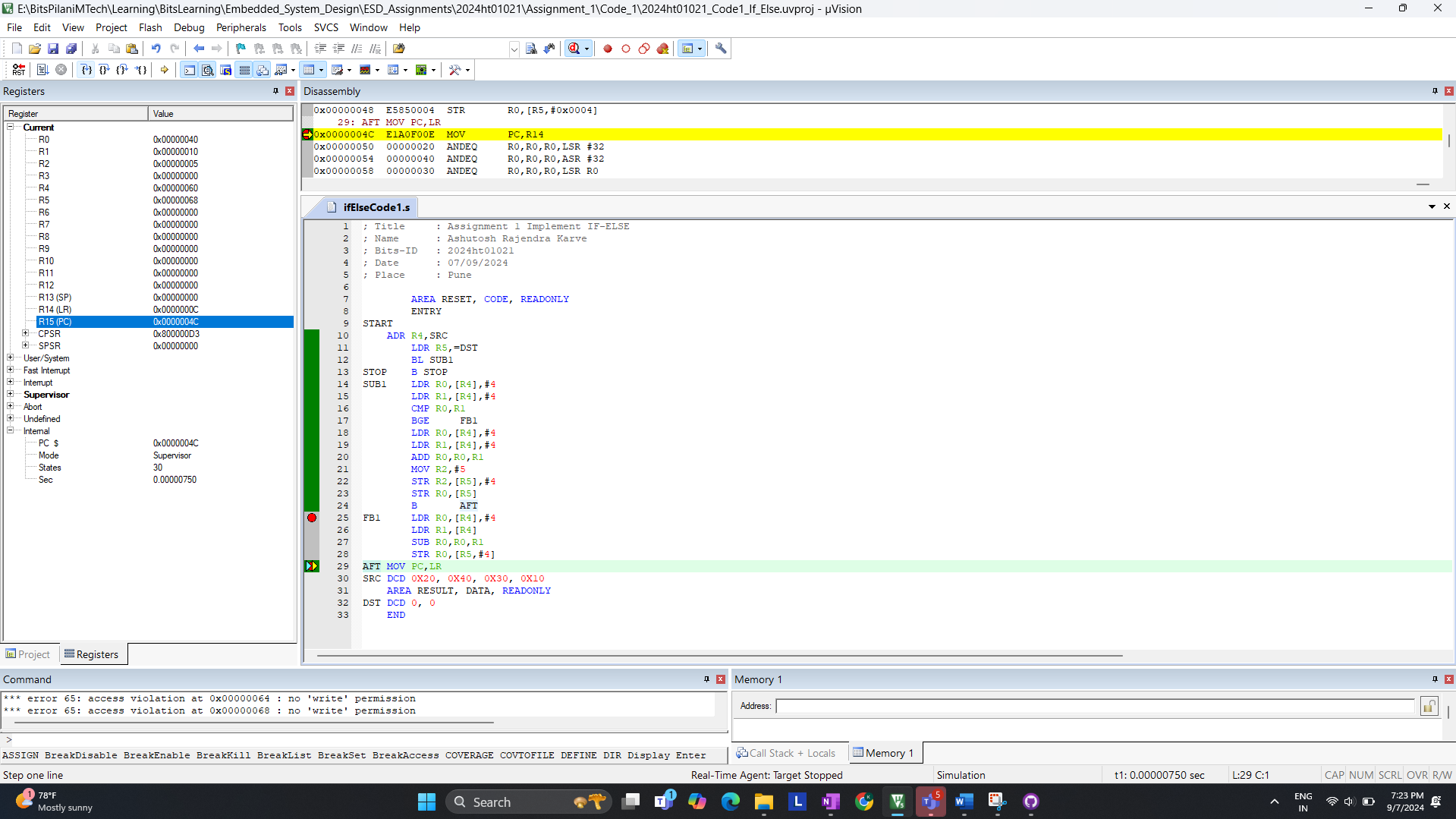
**1. Branch Is Taken >>**  
code: SRC DCD 0x20, 0x40, 0x30, 0x10 ( R0 < R1 ) (0x20 < 0x40)  


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


Now R0 and R1 is loaded with 30 & 10



We are in {if loop} since (R0 < R1 ) new values update for R0 is 40 and R2 is 5 which is our   
{If Block X , Y Value} State : 30

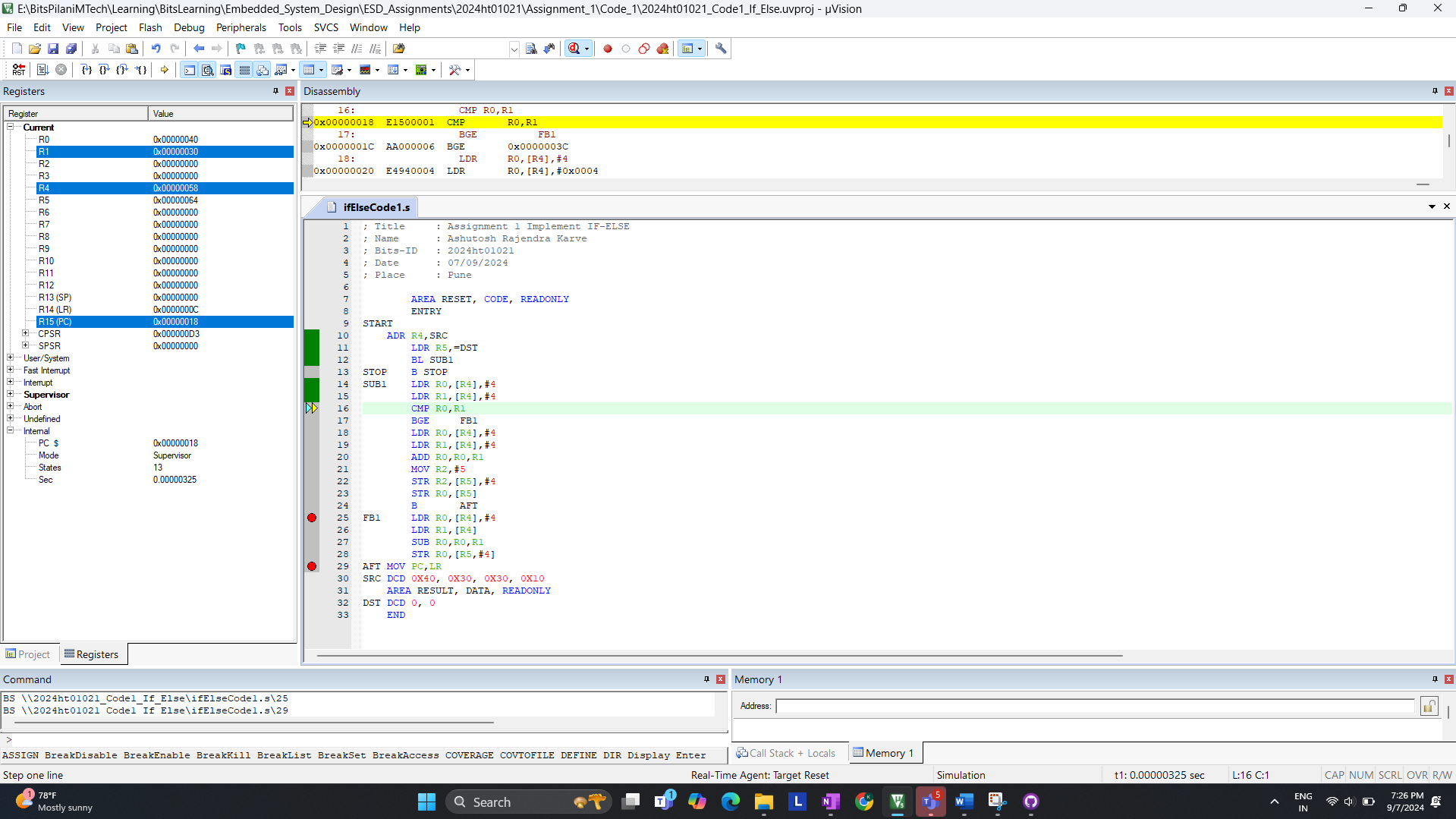


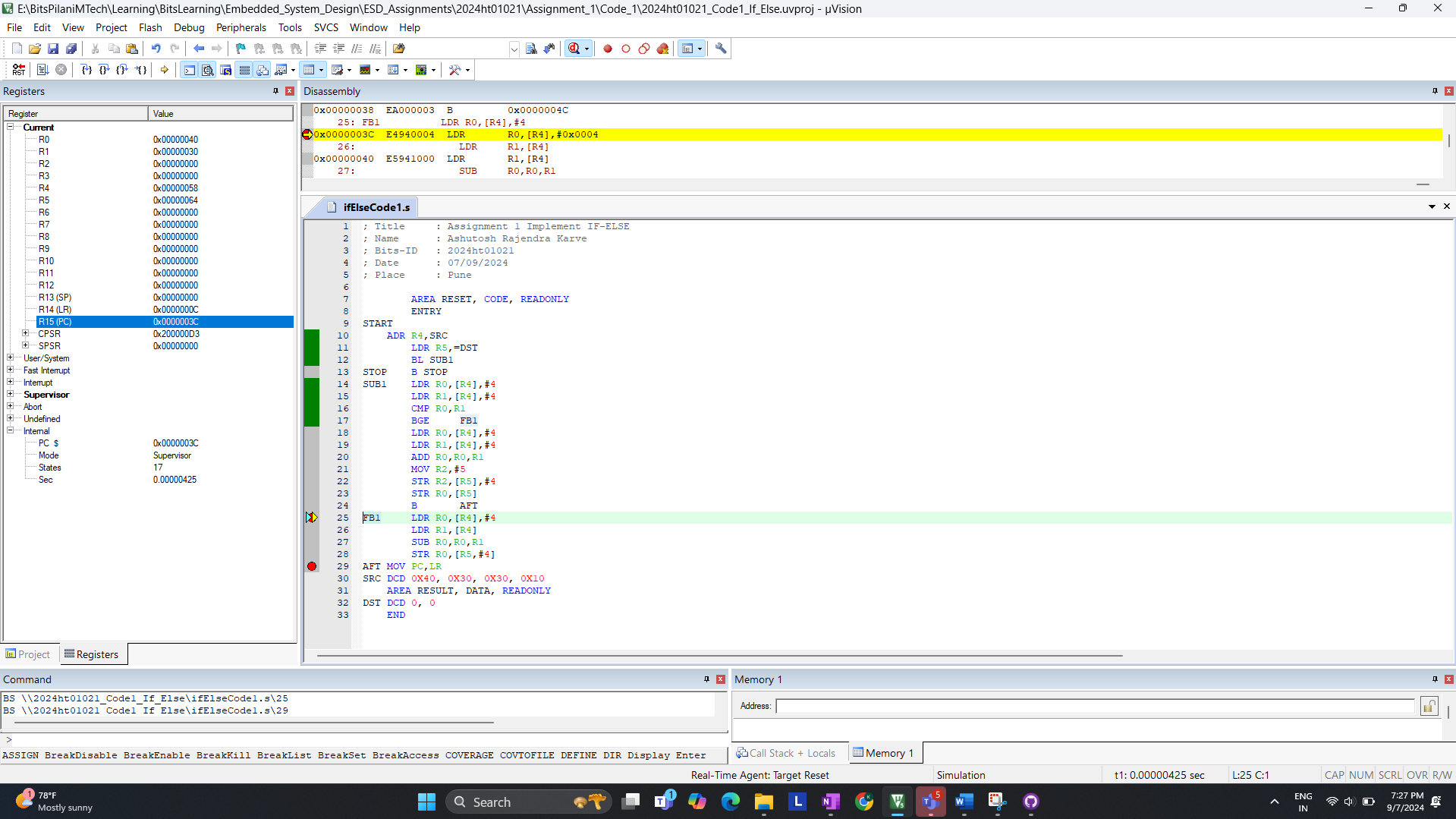
END OF CODE:

A computer screen shot of a program

Description automatically generated

>> 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 )  


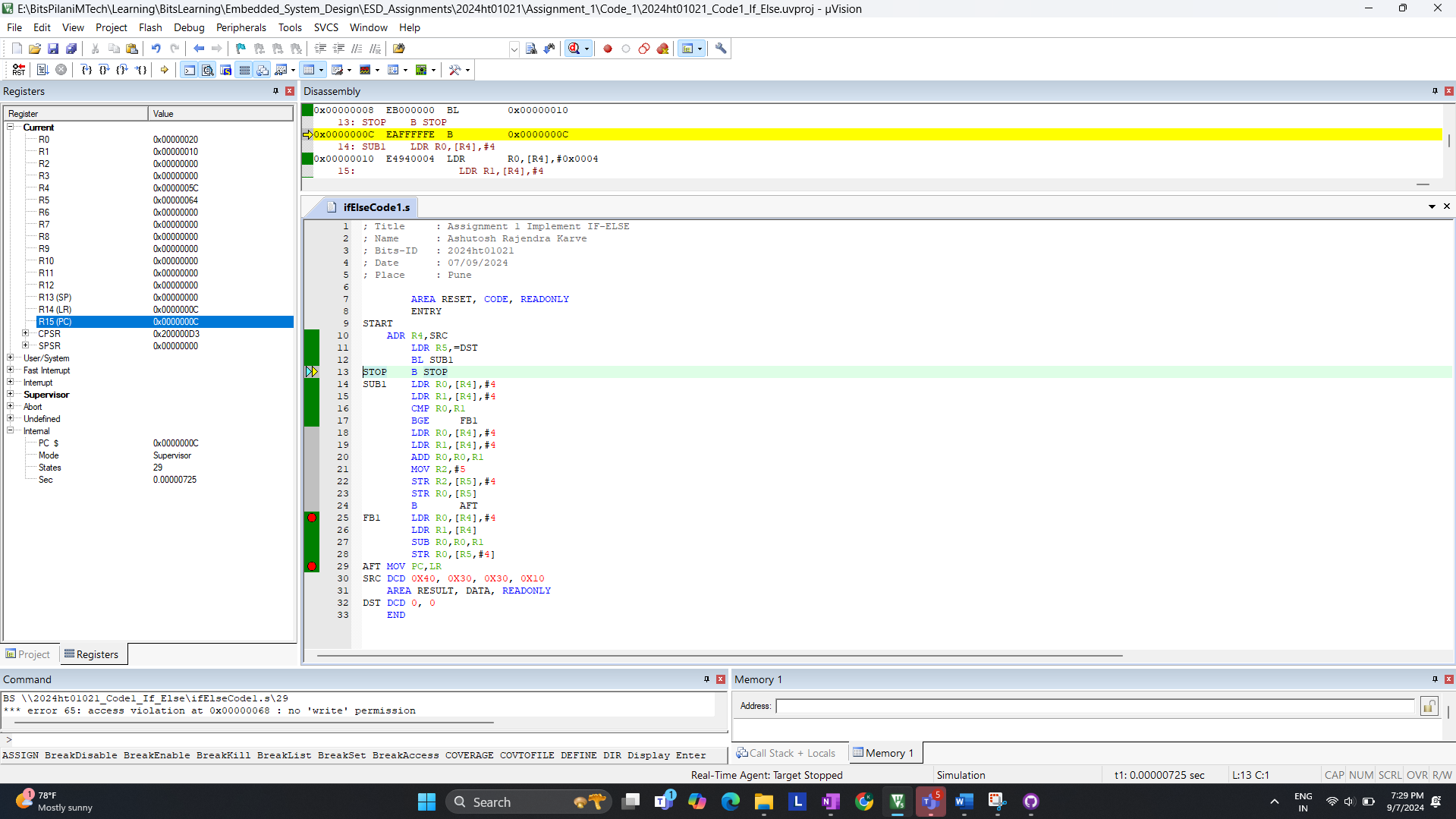
After comparison; it skip the IF-BLOCK enters ELSE-BLOCK; States 17  


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

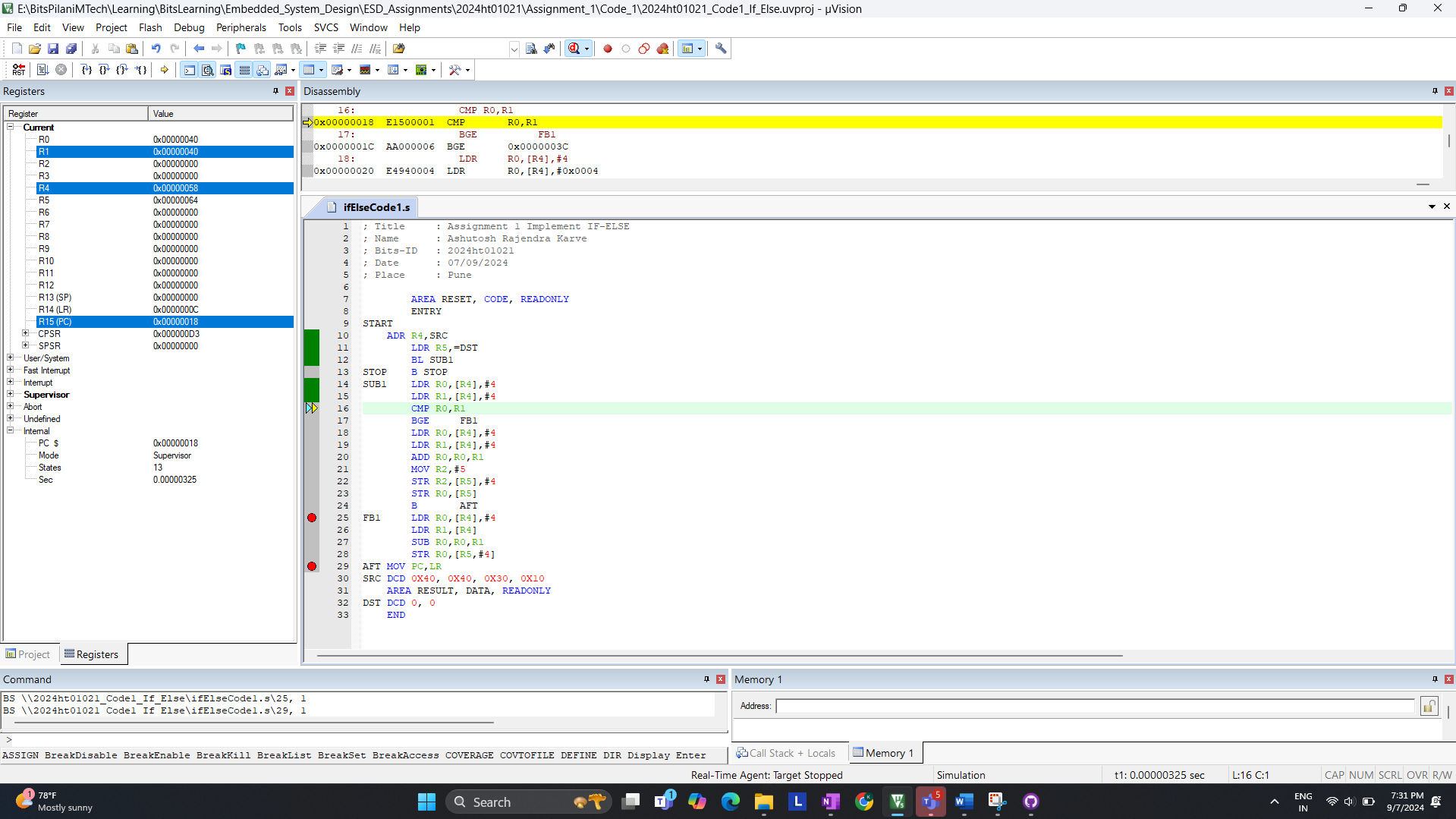
A computer screen shot of a program

Description automatically generated

END OF CODE:



>> Final State with Branch is not Taken is 29

II. code: SRC DCD 0x40, 0x40, 0x30, 0x10 ( R0 == R1)  
For now R0 & R1 is loaded with 40 & 40 ( 40 == 40 )  


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

A computer screen shot of a program

Description automatically generated

A computer screen shot of a program

Description automatically generated

END OF CODE:

A computer screen shot of a program

Description automatically generated

>> 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

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

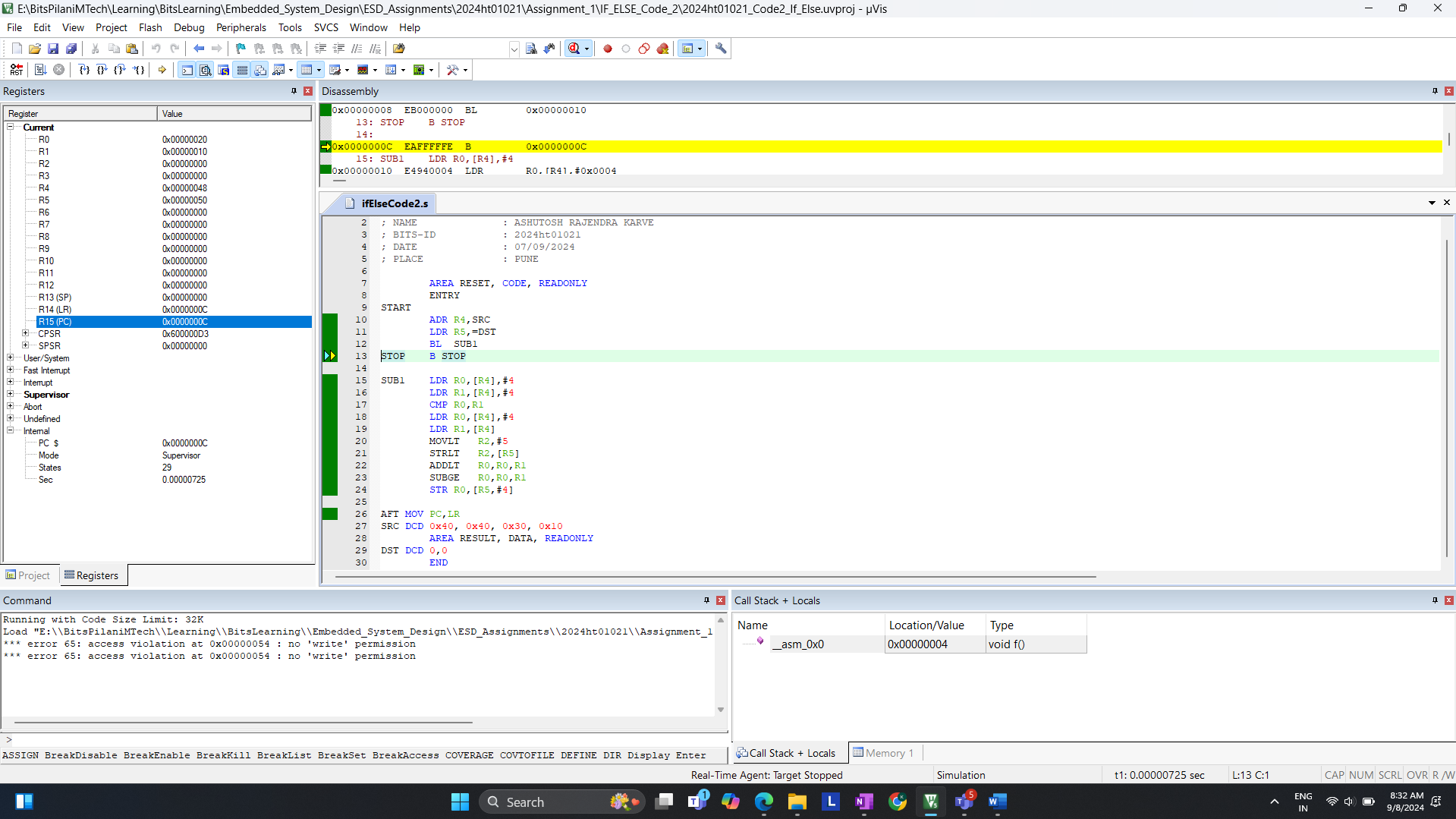
|  |  |  |
| --- | --- | --- |
| **Condition** | **Code-1-States** | **Code-2-States** |
| **a<b** | **33** | **30** |
| **a>b** | **29** | **29** |
| **a=b** | **29** | **29** |

**Screen Shots of Code-2-States**SRC DCD 0x40, 0x30,0x30,0x10 (a > b) States is 29

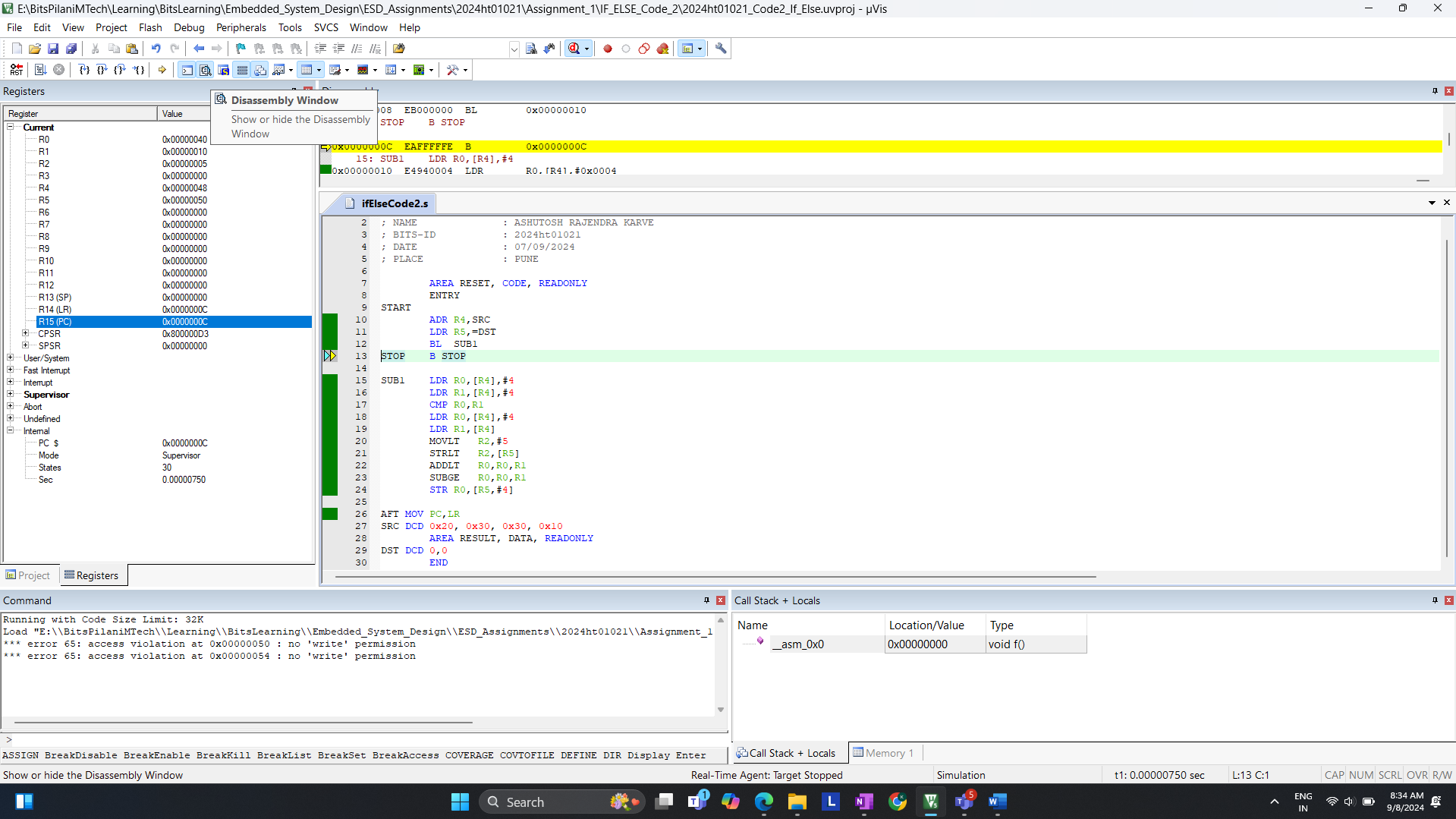
**A screenshot of a computer

Description automatically generated**

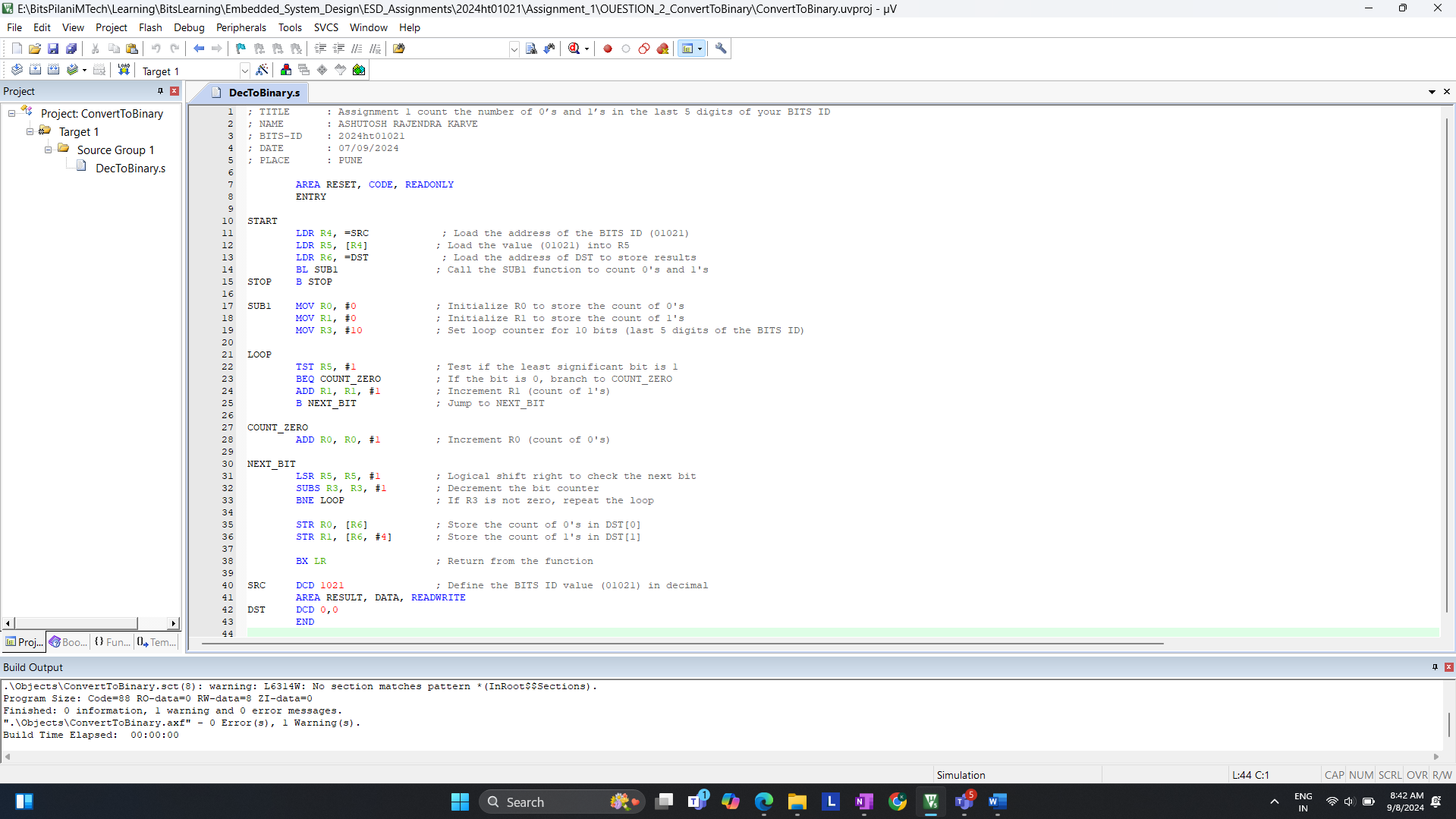
SRC DCD 0x40, 0x40,0x30,0x10 (a = b) States is 29

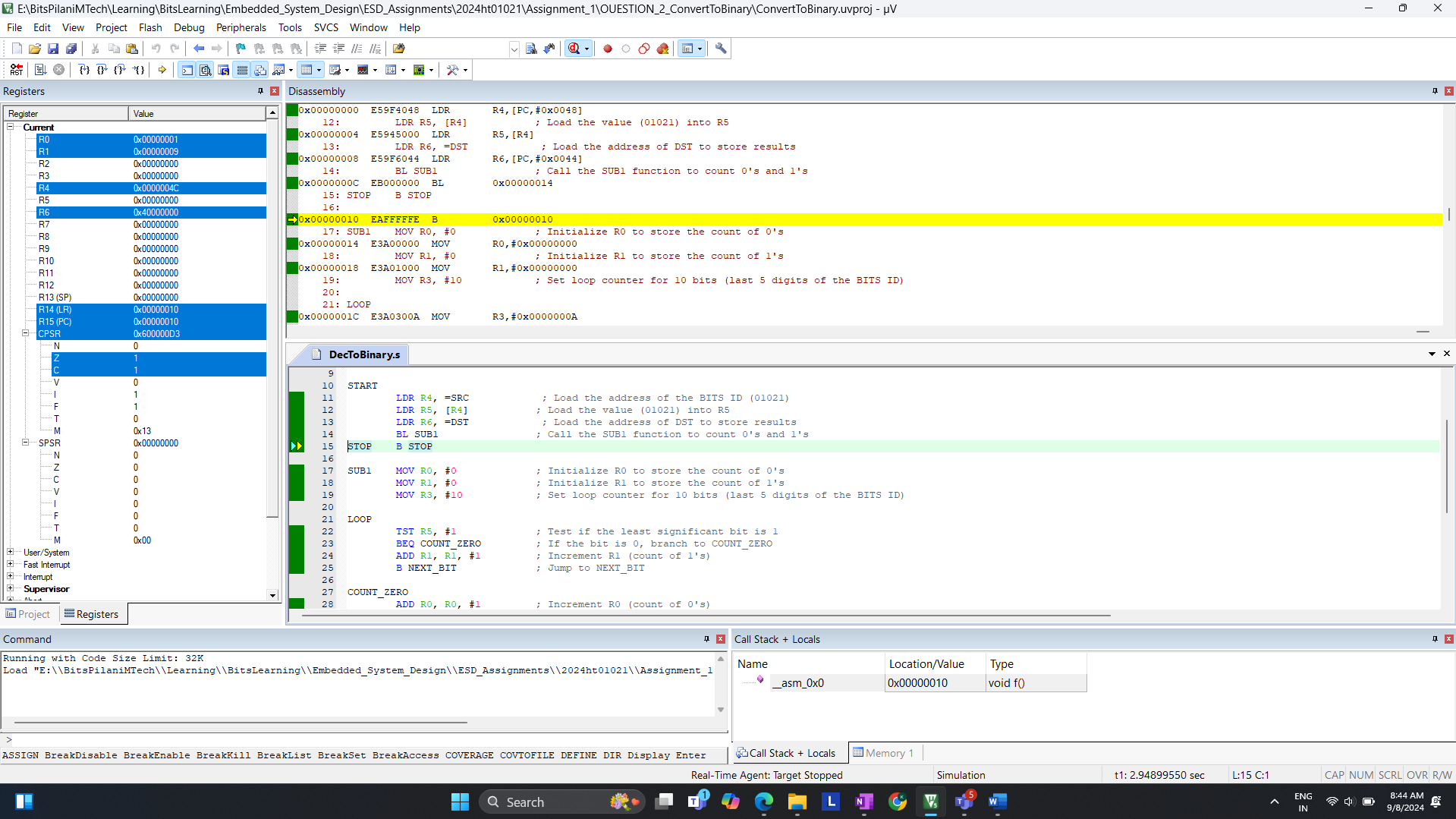
****

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



**Q.2. Write an assembly language program for ARM7TDMI 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.**

Code compilation with errors.  
****

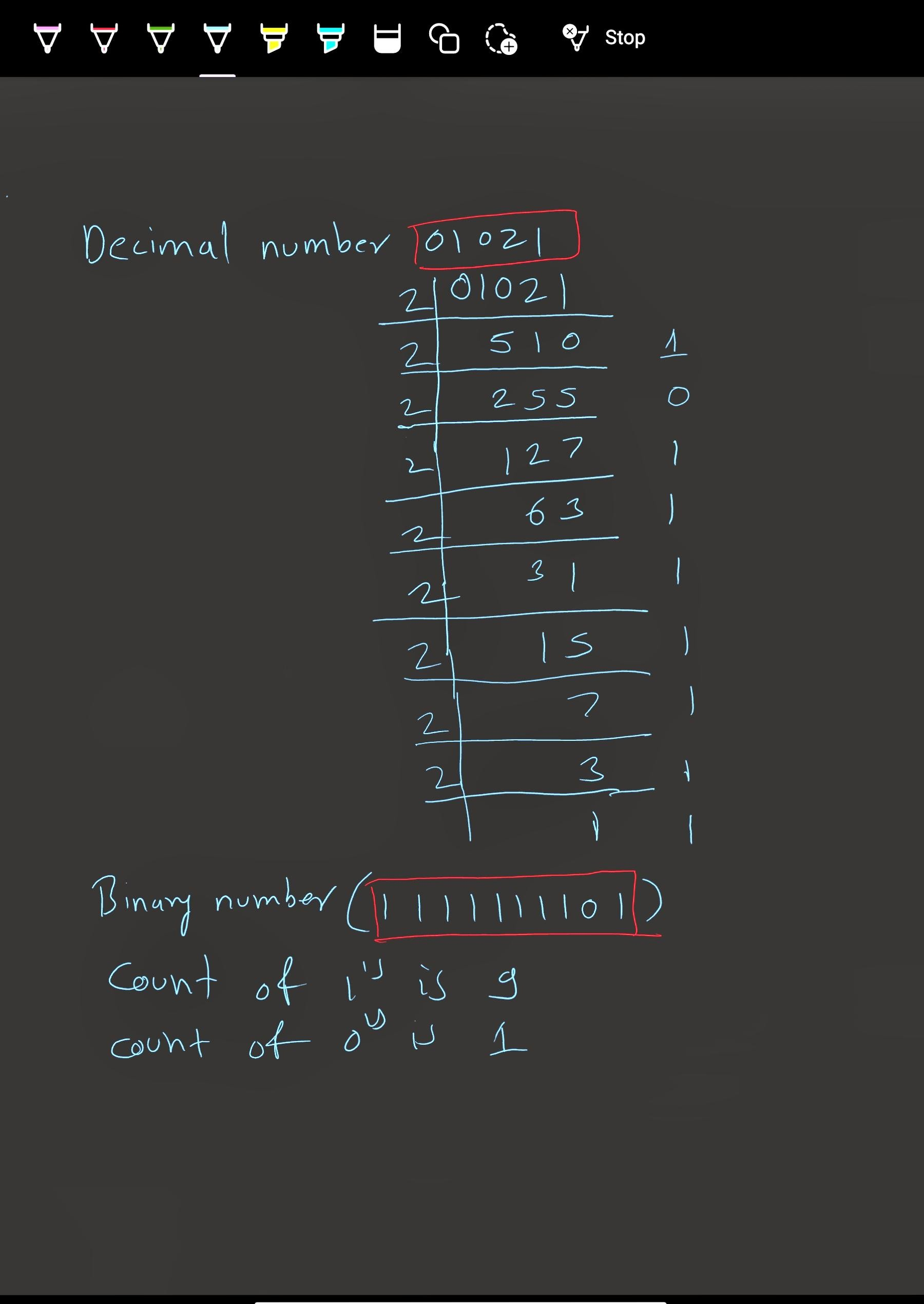
Count of 0’s & 1’s in memory after the execution; Check R0 & R1 ****

**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  
    
   

**Q.3. Write an assembly language program for ARM Cortex M3/4 to handle Supervisor Call (SVC) exception. The program should meet the following requirements:**

**1. The SVC should be called from an application task running at Thread unprivileged mode. [SVC number must be your BITS ID either last 3 digits or last 2 digits, depending on whether it's less than (or equal) or greater than to 255 (number must be given in decimal number format)]**

**2. Two parameters [each parameter is last 5 digits of your BITS ID (number must be given in decimal number format)] should be passed to the SVC handler.**

**3. The SVC number should be determined dynamically by examining the actual SVC instruction in 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 your BITS ID (either last 3 digits or last 2 digits, depending on whether it's less than or equal to 255), the handler should perform addition of the two passed parameters.**

**6. If the SVC number does not match your BITS ID, the handler should perform subtraction of the two passed parameters. Ensure that the program properly handles stack usage and resumes the application task after performing the required operation.**