**MICROPROCESSOR AND COMPUTER ARCHITECTURE LABORATORY**

**UE19CS256**

**4TH SEMESTER, ACADEMIC YEAR 2020-21**

|  |  |  |
| --- | --- | --- |
| Name: Atul Anurag | SRN: PES2UG19CS075 | Section: B |

**Date:27-01-2021**

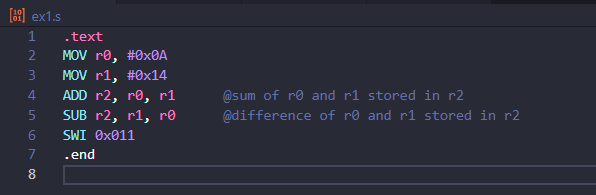
**WEEK#1**

Program Number: 1

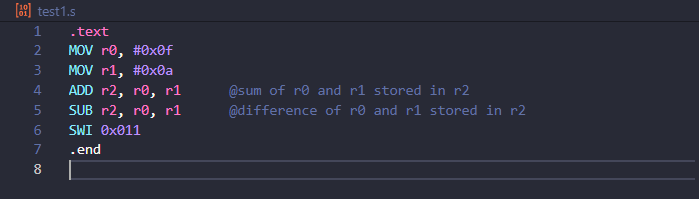
**Write an ALP using ARM instruction set to add and subtract two 32-bit numbers. Both numbers are in registers.**

1. ARM Assembly Code for each program

Example case:



Test Case:



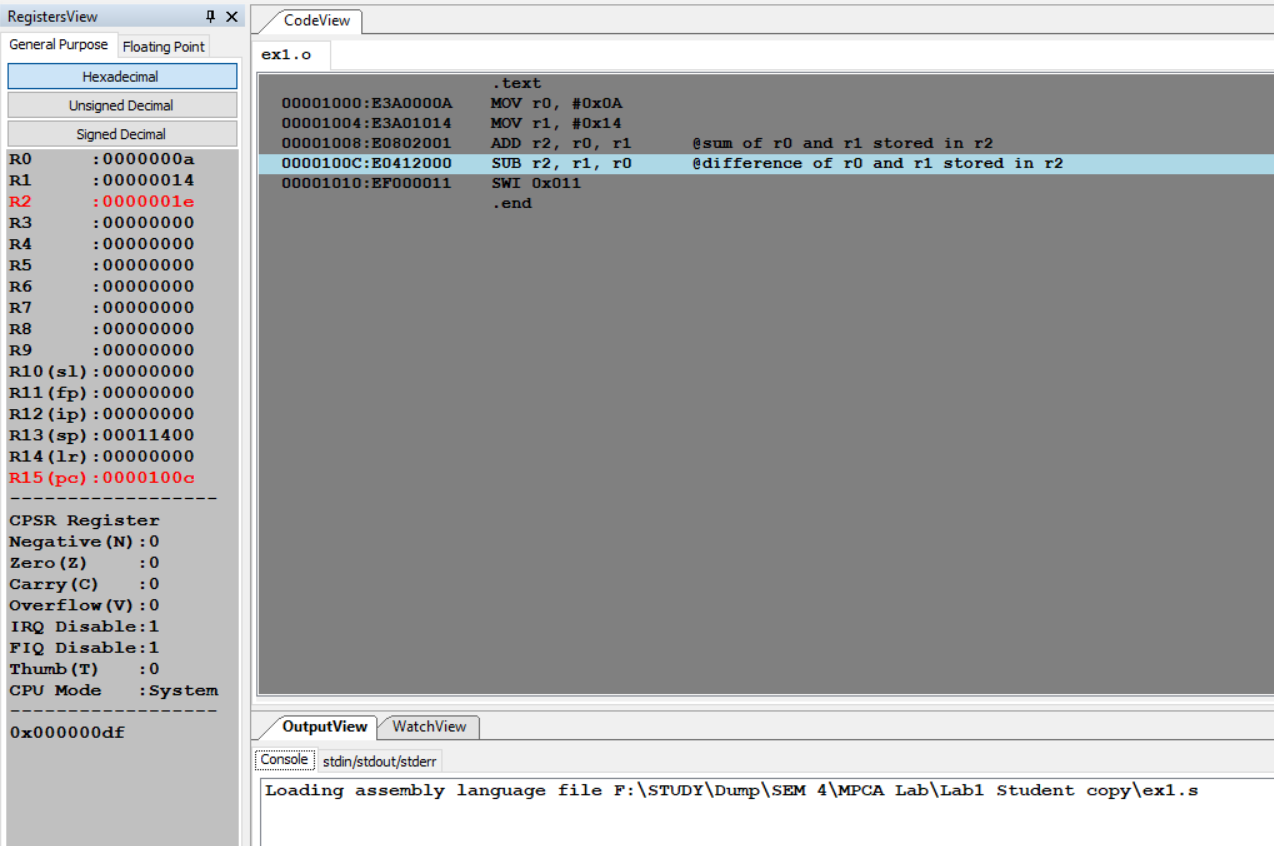
1. Output Screen Shot (Register Window, Output window)

The output should be verified with 2 test cases

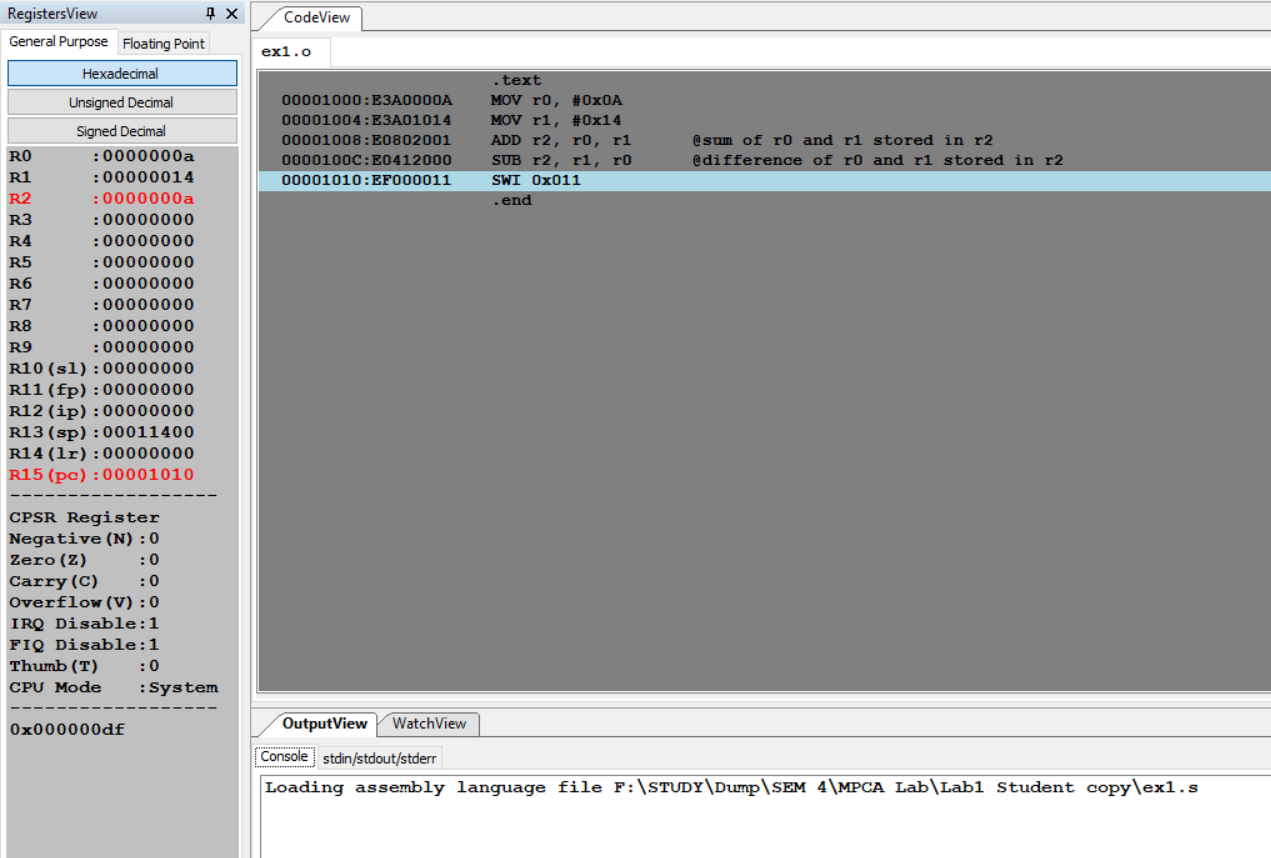
(one example shown in class, one example of own choice)

Example case:

Addition:

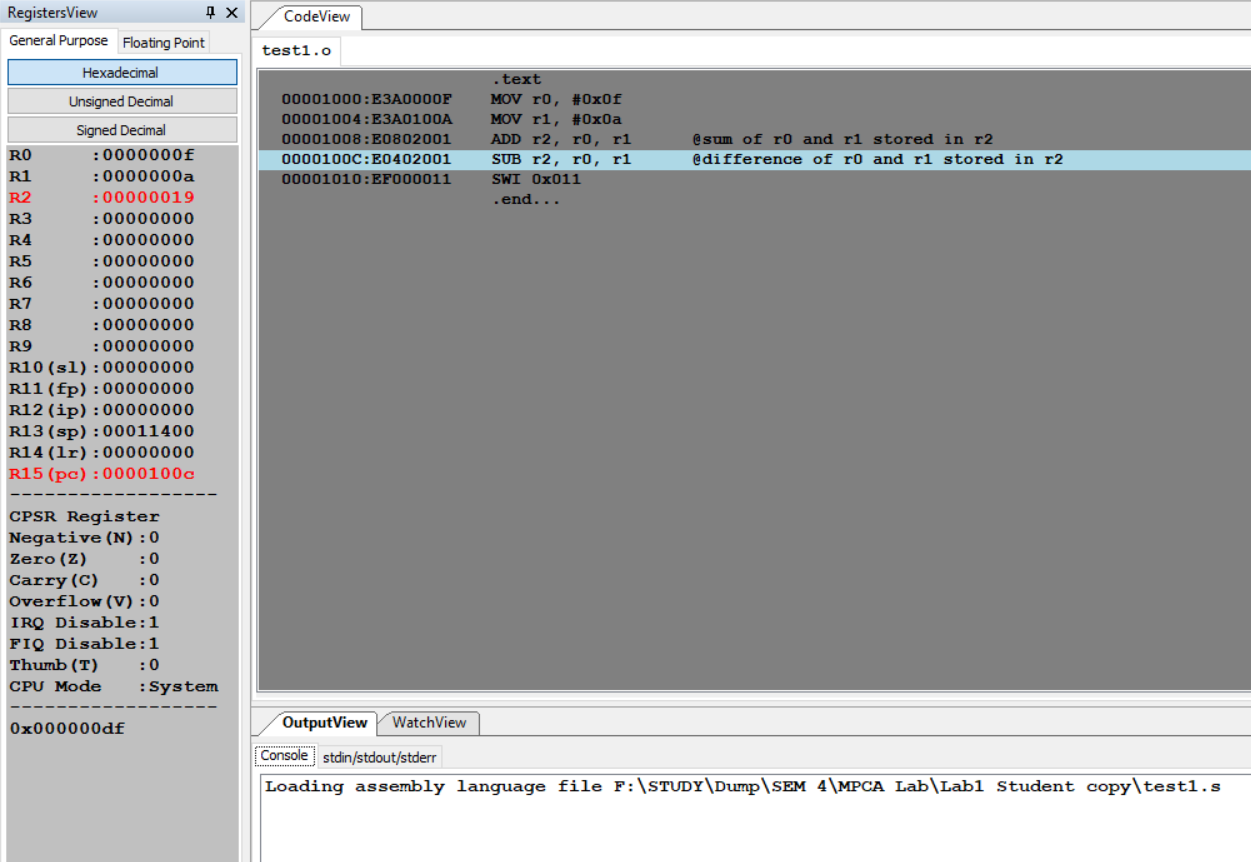


Subtraction:

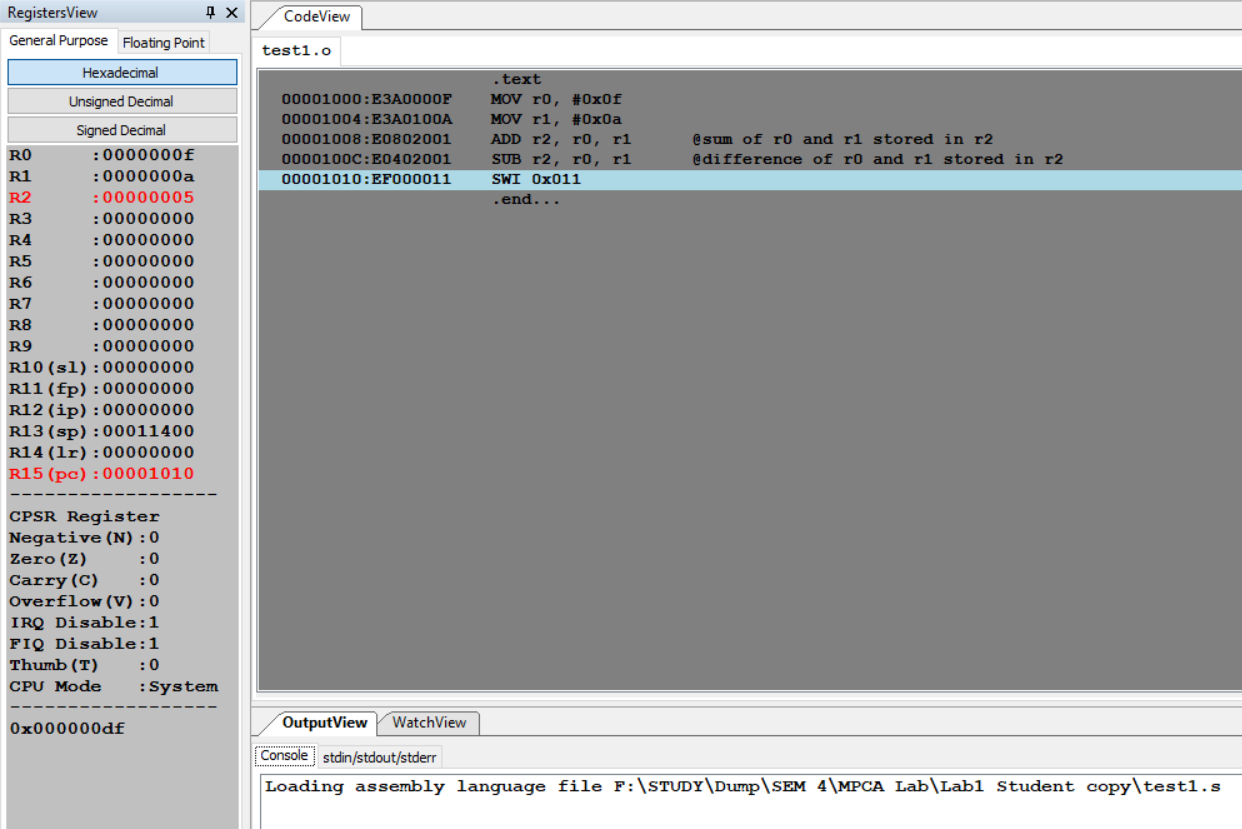


Test Case:

Addition:



Subtraction:



Output table for each program

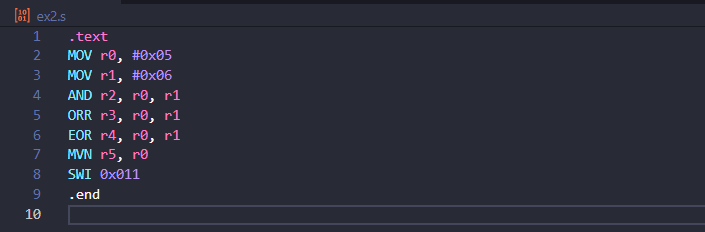
|  |  |  |  |
| --- | --- | --- | --- |
| **EXAMPLE CASE:**  **R0=10=Hex 0A**  **R1=20=Hex 14**  **After Addition R2=30=Hex 1E**  **After Subtraction R2 = 10 = Hex 0A** | | | |
| **R0** | **R1** | **Arithmetic Operation** | **Result** |
| 0x0A | 0x14 | ADD | R0=0x1E |
| 0x0A | 0x14 | SUBTRACT | R0=0x0A |
| **TEST CASE:**  **R0 = 15 = Hex 0F**  **R1 = 10 = Hex 0A**  **After Addition R2 = 25 = Hex 19**  **After Subtraction R2 = 5 = Hex 05** | | | |
| **R0** | **R1** | **Arithmetic Operation** | **Result** |
| 0x0f | 0x0a | ADD | R0=0x19 |
| 0x0f | 0x0a | SUBTRACT | R0=0x05 |

Program Number: 2

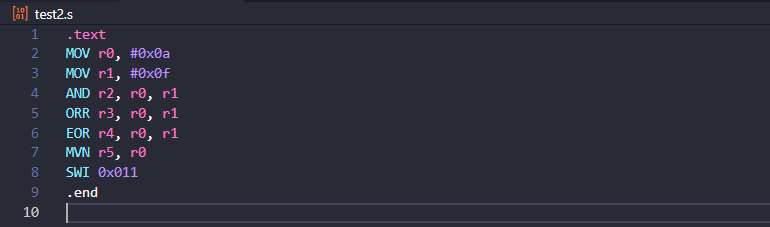
**Write an ALP to demonstrate logical operations. All operands are in registers.**

1. ARM Assembly Code for each program

Example Case:



Test Case:

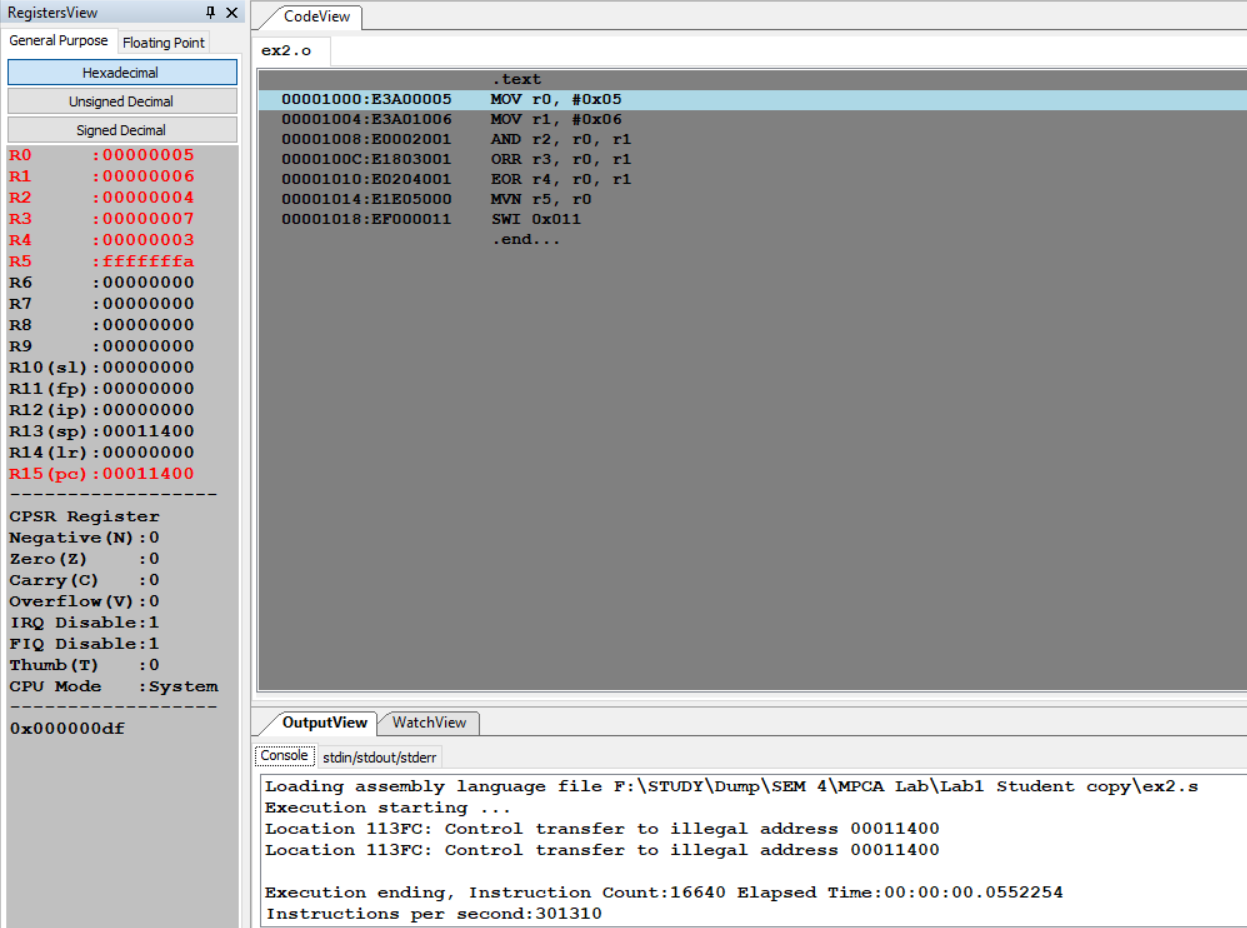


1. Output Screen Shot (Register Window, Output window)

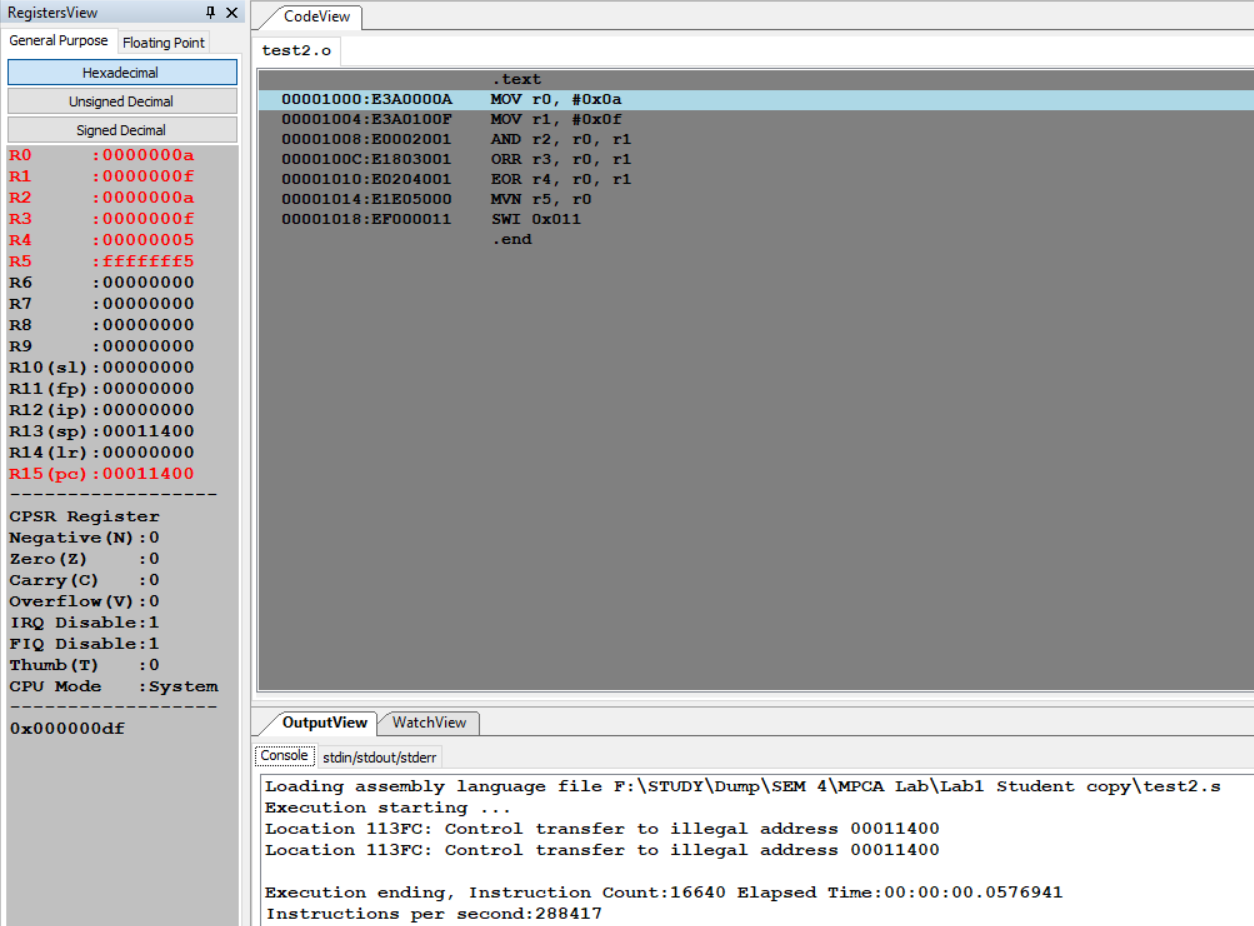
The output should be verified with 2 test cases

(one example shown in class, one example of own choice)

Example Case:



Test Case:



1. Output table for each program

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EXAMPLE CASE: | | | | |
| **R0** | **R1** | **Logical Operation** | **Instruction** | **Result** |
| 0x05 | 0x06 | AND | AND | R2=0x04 |
| 0x05 | 0x06 | OR | ORR | R3=0x07 |
| 0x05 | 0x06 | EX-OR | EOR | R4=0x03 |
| 0x05 |  | NOT | MVN | R5=0xfffffffa |

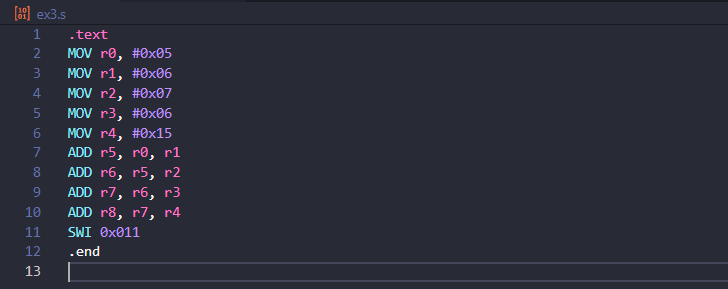
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TEST CASE: | | | | |
| **R0** | **R1** | **Logical Operation** | **Instruction** | **Result** |
| 0x0a | 0x0f | AND | AND | R2=0x0a |
| 0x0a | 0x0f | OR | ORR | R3=0x0f |
| 0x0a | 0x0f | EX-OR | EOR | R4=0x05 |
| 0x0a |  | NOT | MVN | R5=0xfffffff5 |

Program Number: 3

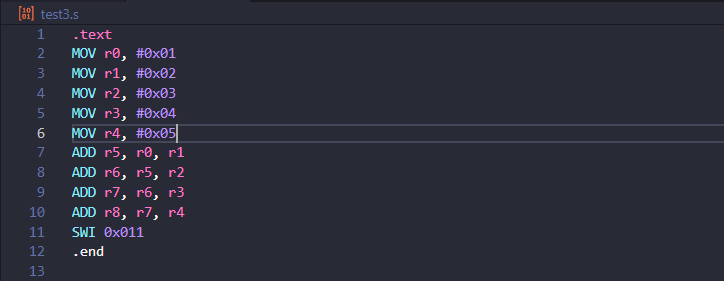
**Write an ALP to add 5 numbers where values are present in registers.**

1. ARM Assembly Code for each program.

Example Case:



Test Case:

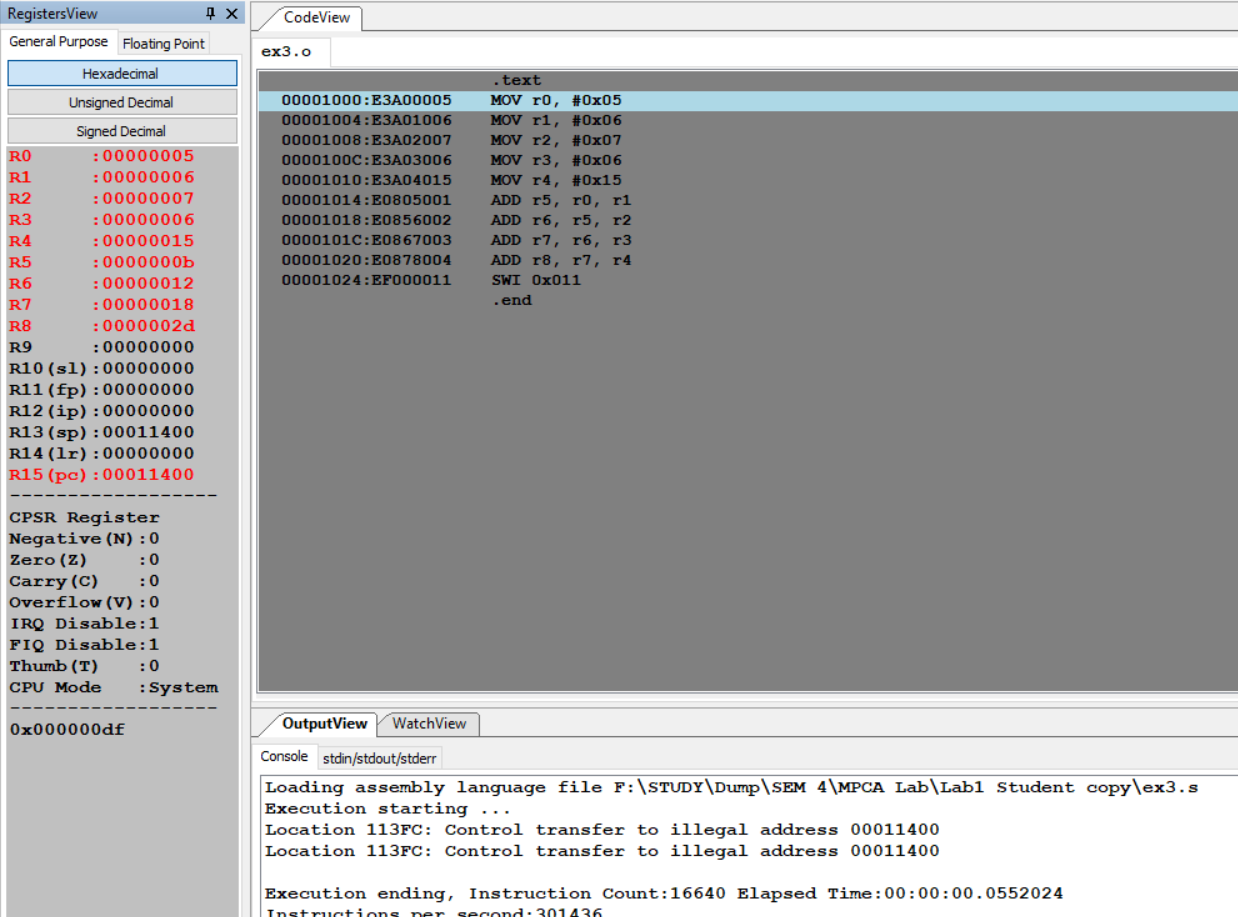


1. Output Screen Shot (Register Window, Output window)

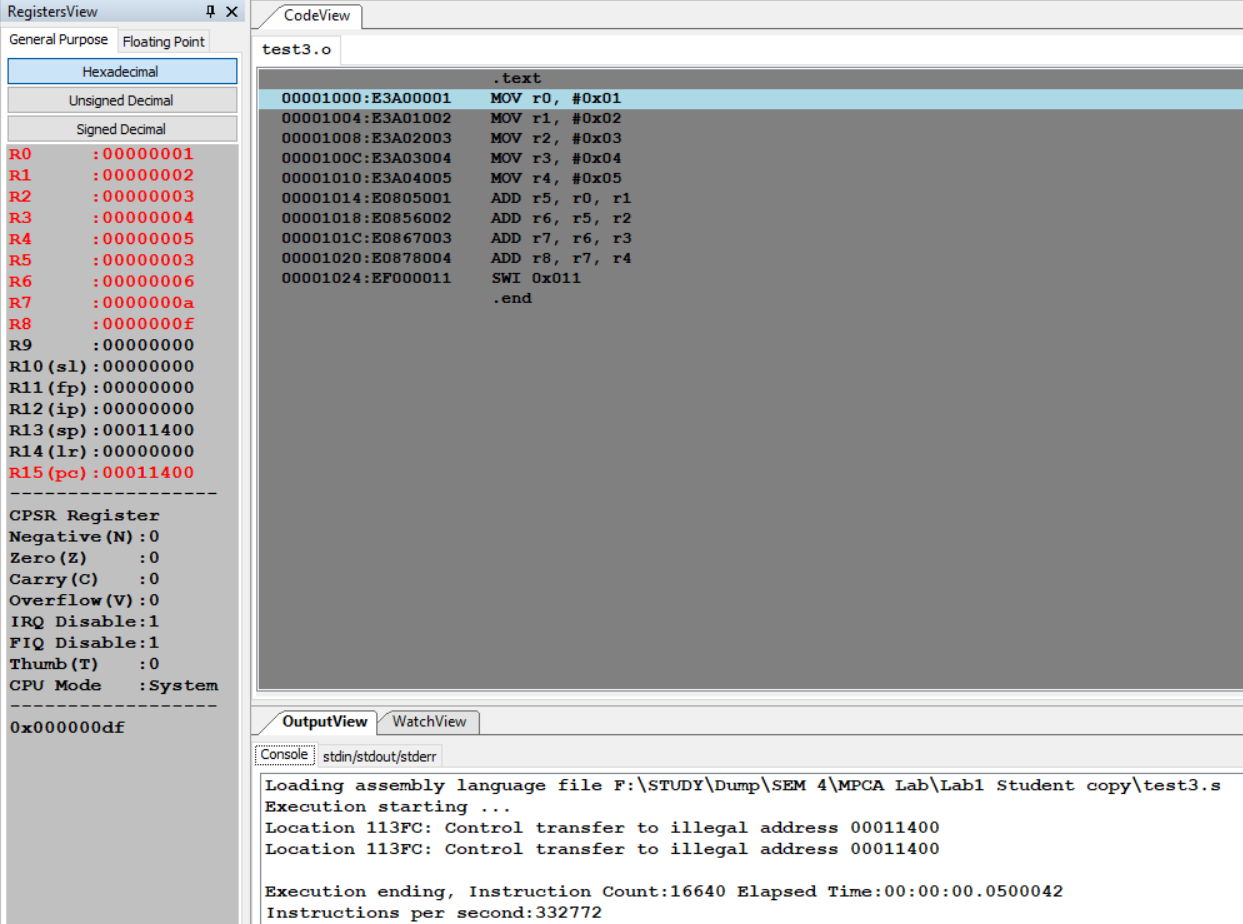
The output should be verified with 2 test cases

(one example shown in class, one example of own choice)

Example Case:



Test Case:



1. Output table for each program

|  |  |  |
| --- | --- | --- |
|  | EXAMPLE CASE: |  |
| R0 |  | 0x05 |
| R1 |  | 0x06 |
| R2 |  | 0x07 |
| R3 |  | 0x06 |
| R4 |  | 0x15 |
| R5 | R0+R1 | 0x0b |
| R6 | R5+R2 | 0x12 |
| R7 | R6+R3 | 0x18 |
| R8 | R7+R4 | 0x27 |

|  |  |  |
| --- | --- | --- |
|  | TEST CASE: |  |
| R0 |  | 0x01 |
| R1 |  | 0x02 |
| R2 |  | 0x03 |
| R3 |  | 0x04 |
| R4 |  | 0x05 |
| R5 | R0+R1 | 0x03 |
| R6 | R5+R2 | 0x06 |
| R7 | R6+R3 | 0x0a |
| R8 | R7+R4 | 0x0f |

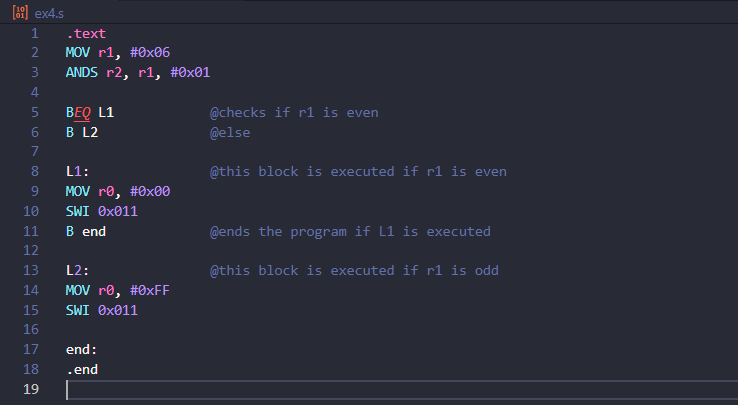
Program Number: 4

**Write an ALP using ARM instruction set to check if a number stored in a register is even or odd. If even, store 00 in R0, else store FF in R0**

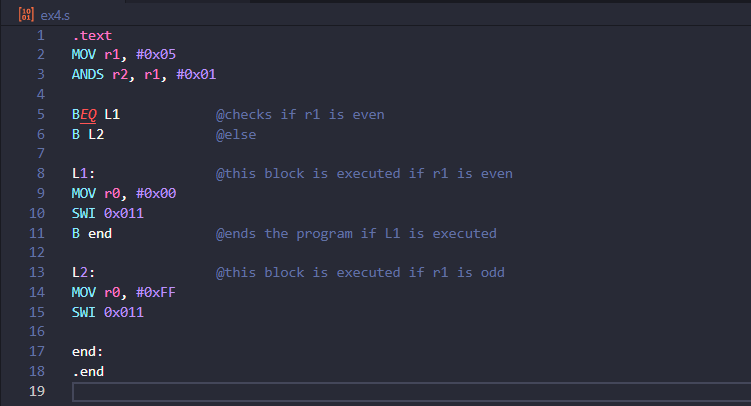
1. ARM Assembly Code for each program

Example Code:

Even:

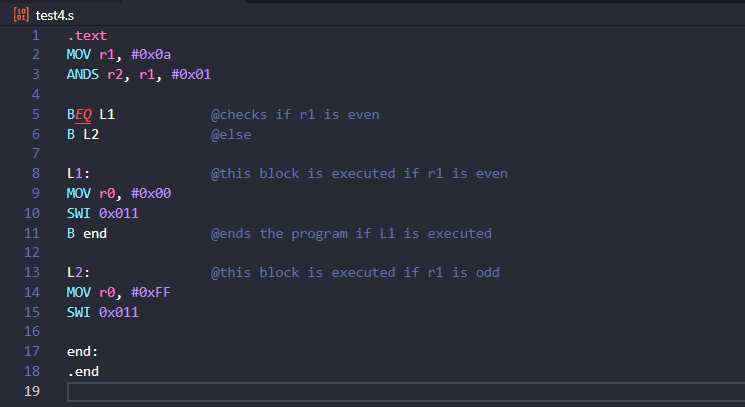


Odd:

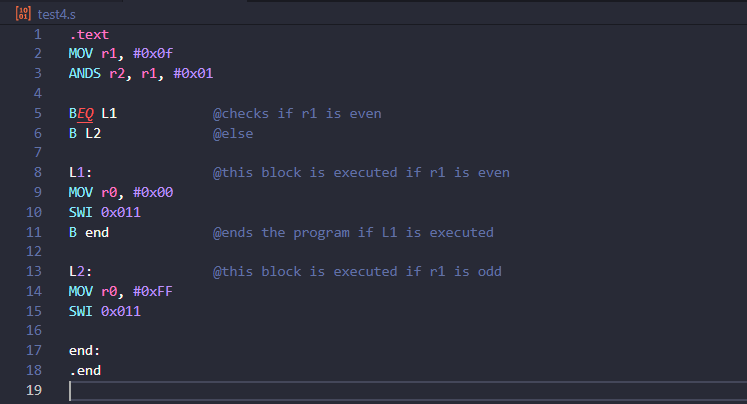


Test Case:

Even:



Odd:



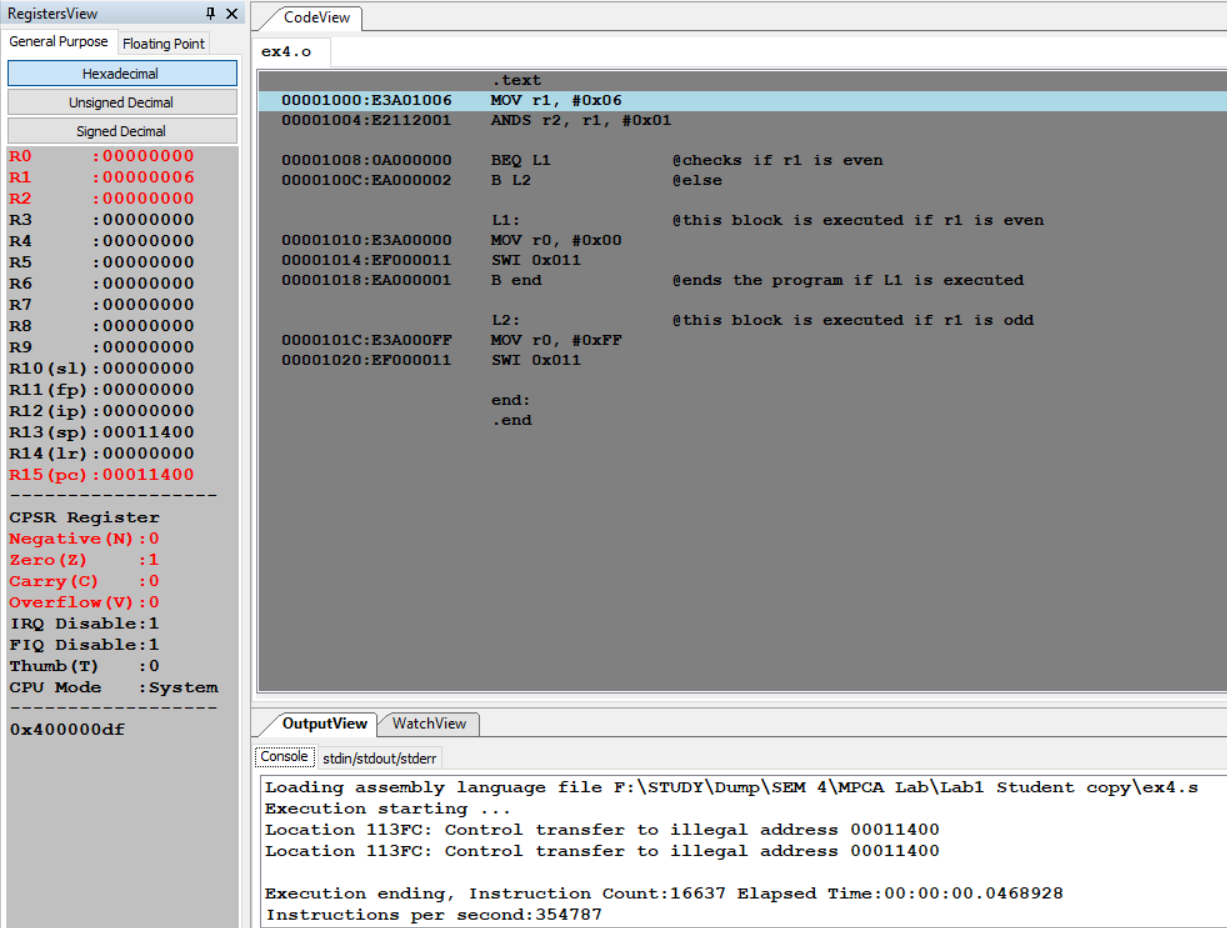
1. Output Screen Shot (Register Window, Output window)

The output should be verified with 2 test cases

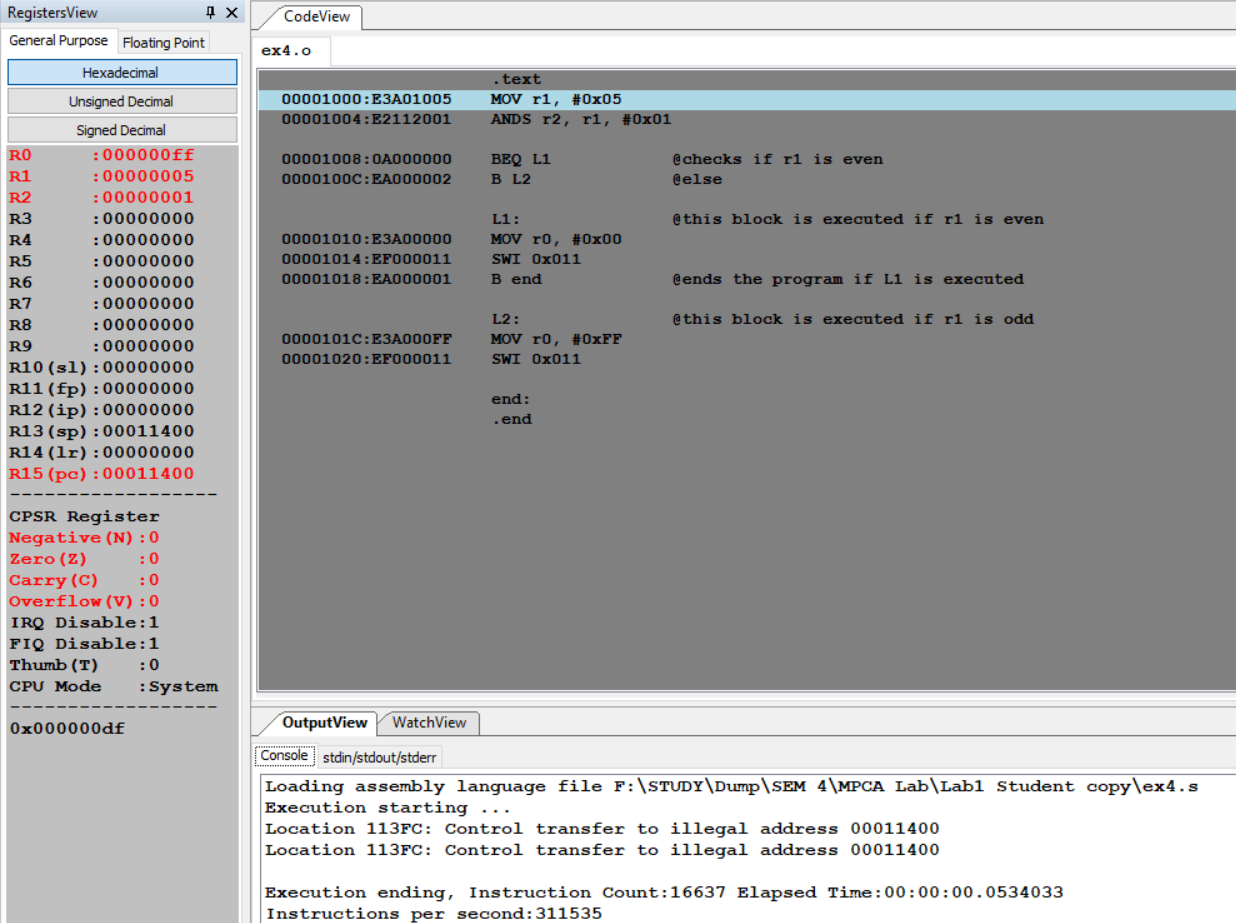
(one example shown in class, one example of own choice)

Example Case:

Even:

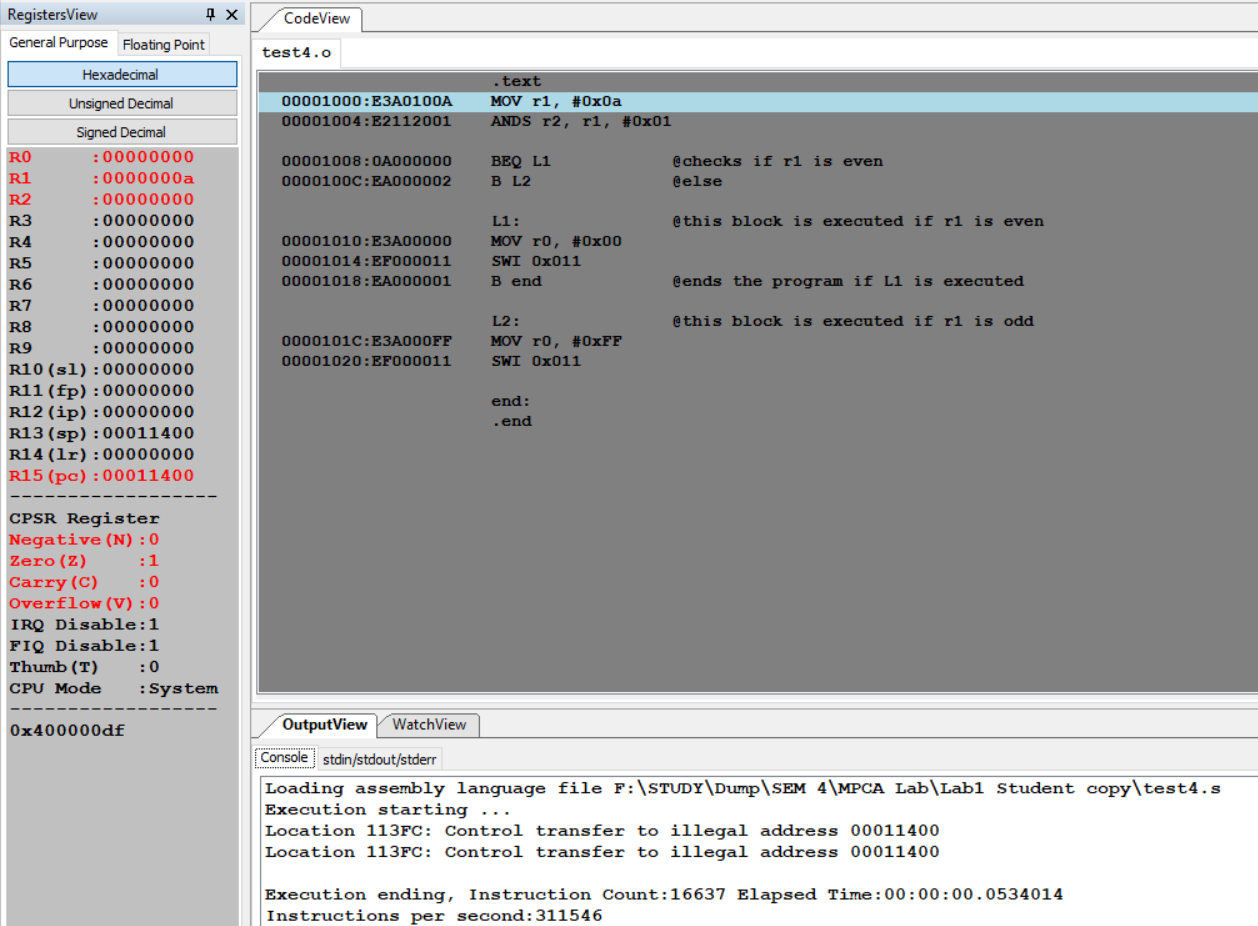


Odd:

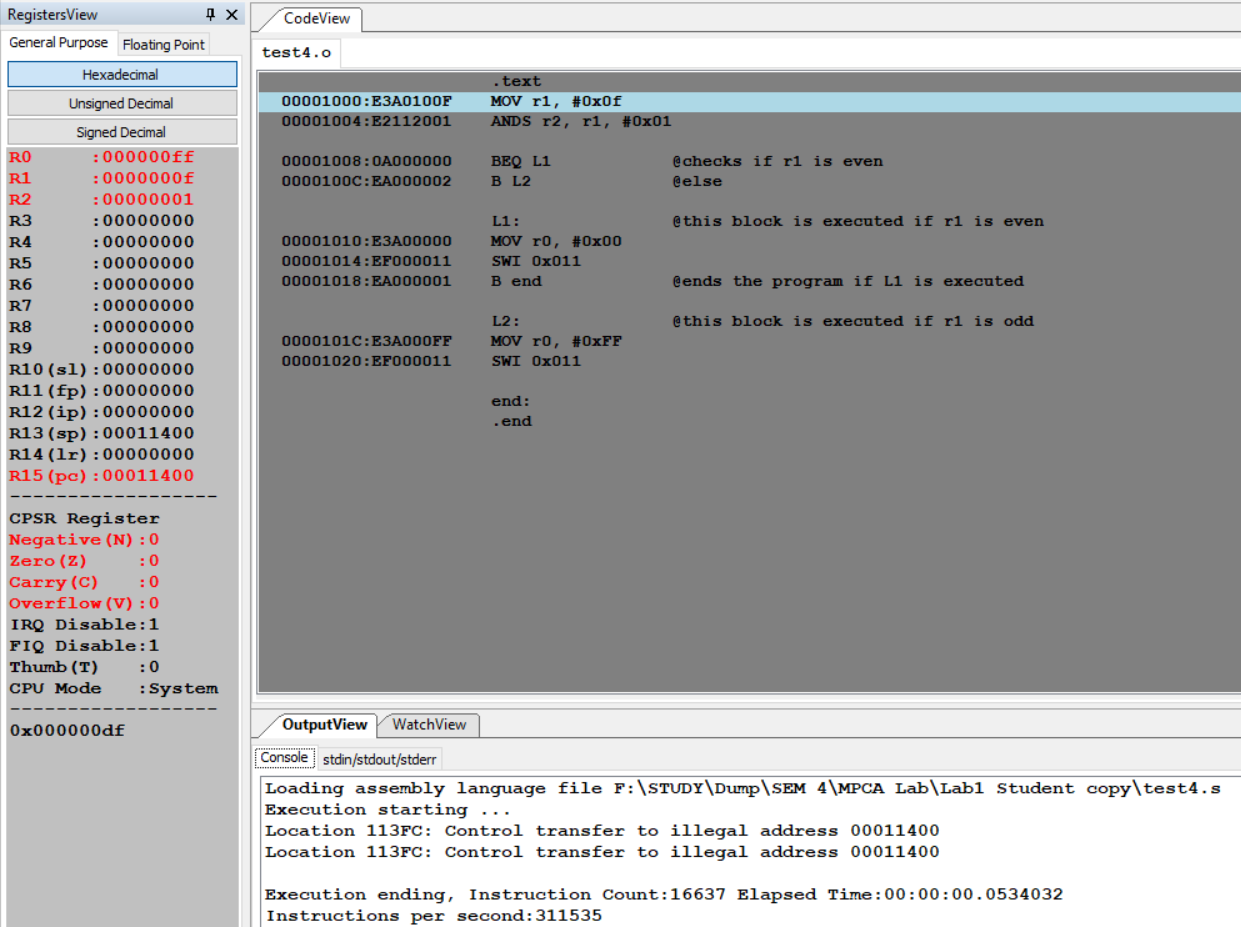


Test Case:

Even:



Odd:



1. Output table for each program

|  |  |  |  |
| --- | --- | --- | --- |
| EXAMPLE CASE | | | |
| CASE 1 | R1 |  | 0x06 |
|  | R2 | After AND operation | 0x00 |
|  | R0 | (EVEN) | 0x00 |
| CASE 2 | R1 |  | 0x05 |
|  | R2 | After AND operation | 0x01 |
|  | R0 | (ODD) | 0xFF |

|  |  |  |  |
| --- | --- | --- | --- |
| TEST CASE | | | |
| CASE 1 | R1 |  | 0x0A |
|  | R2 | After AND operation | 0x00 |
|  | R0 | (EVEN) | 0x00 |
| CASE 2 | R1 |  | 0x0F |
|  | R2 | After AND operation | 0x01 |
|  | R0 | (ODD) | 0xFF |

**Disclaimer:**

The programs and output submitted is duly written, verified and executed by me.

I have not copied from any of my peers nor from the external resource such as internet.

If found plagiarized, I will abide with the disciplinary action of the University.

Signature: Atul Anurag

Name: Atul Anurag

SRN: PES2UG19CS075

Section: B

Date: 27-01-2021