**DOCUMENTATION**

**CS3725**

**Dr. Ashoke Deb**

**ASSIGNMENT 3**

**Dido Maulana – 201415635**

**Greg Berezny – 201421963**

Table of Contents

**USER GUIDE1**

(i)How to Run

(ii)Register Input Format

(ii)Memory Address Input Format

**PROGRAM STRUCTURE3**

(i)Buffer Format4

**TESTING5**

(i)add5

(ii)sub7

(iii)lw9

(iv)sw10

(v)beq12

**CODE14**

**USER GUIDE**

The program coded in Java will trace the progress of an instruction through the stages of MIPS machine. PC starts off as 4 as you start the trace which is a constant assumed value in this case to simulate the MIPS machine.

Supported MIPS instructions: **add, sub, lw ,sw, beq**

1. **How to Run**

**Step 1.** Open Command Prompt (cmd)

**Step 2**. Go to the directory where the files are located

**Step 3**. Type in; java MIPSTrace

**Step 4.** Enter inputs (**Refer to (ii) Input Format**)

1. **Register Input Format**

The way data being input into the program varies depending on the instruction you are going to trace through and we have assumed the input in this case will be in decimal. In these input format R are registers while offset is the 16-bit immediate value for the instruction.

* **add**

**Semantics**

R1 🡨 R2 + R3

**Format of Input**

add 1 2 3

* **sub**

**Semantics**

R1 🡨 R2 - R3

**Format of Input**

sub 1 2 3

* **lw**

**Semantics**

R1 🡨 MEMORY[R2 + OFFSET]

**Format of Input**

lw 1 offset 2

* **sw**

**Semantics**

MEMORY[$2 + OFFSET] 🡨 $1

**Format of Input**

sw 1 offset 2

* **beq**

**Semantics**

IF ($1 == $2) THEN PC 🡨 PC + 4 + 4 \* OFFSET

**Format of Input**

beq 1 2 offset

1. **Memory Address Input Format**

Memory address for read and write along with their values will be required in order to have an initial value within those possible target addresses. The format of entry is as follows;

**Semantics**

<address> <value>

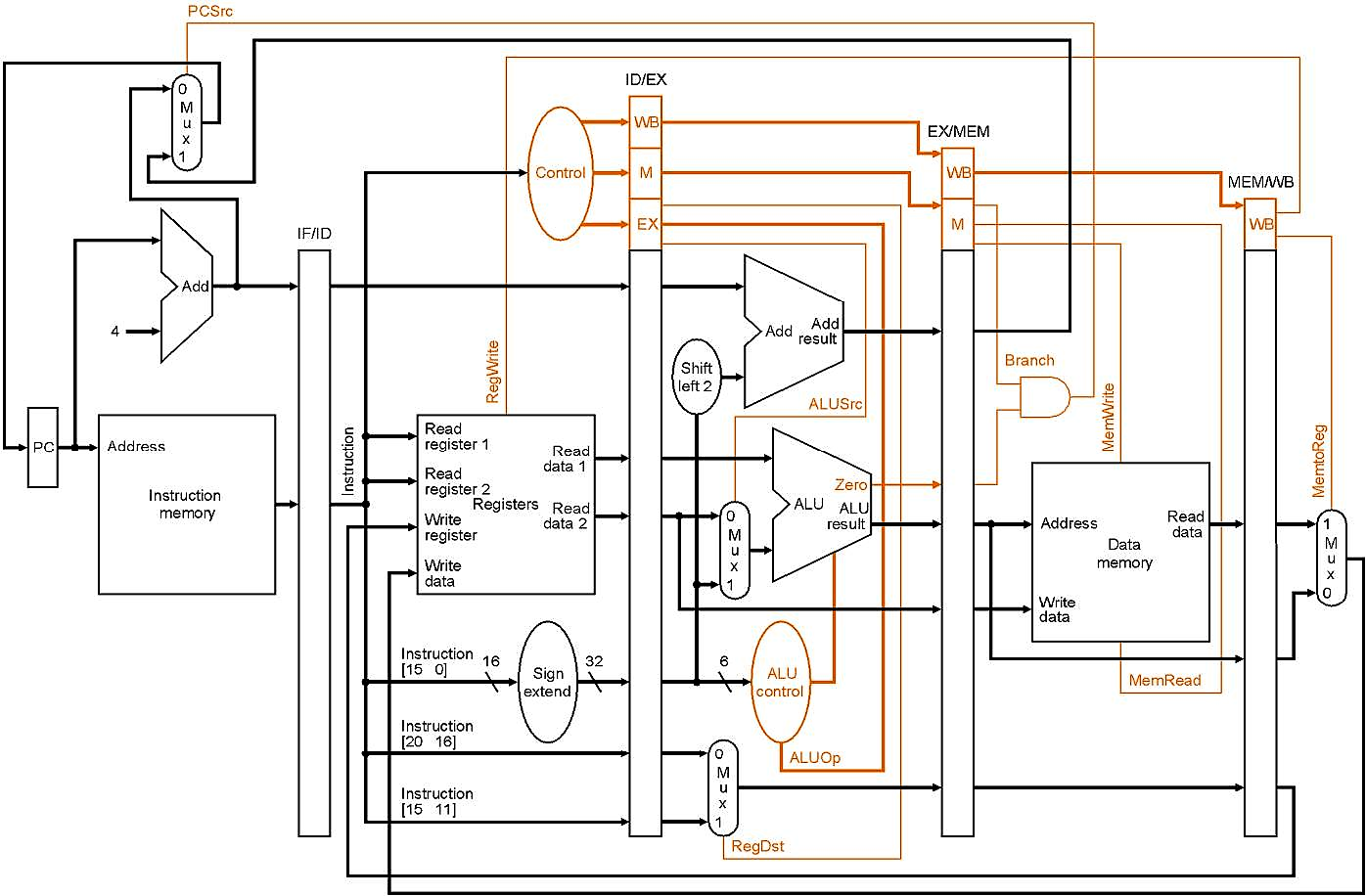
**Format of Input**

125 3

Where 125 is the memory address and 3 is the value within the address.

**PROGRAM STRUCTURE**

The program is coded in 1 class with several methods for each stage. As you go through each one of the buffers a new method is called to retrieve and process the information within the buffer as required. Some of the printed output within the program are labelled below within the image of the architectural view of the MIPS machine. Not all of the components within the structure were shown within the program. For example, due to the input assumption being in decimal we have ignored the Sign Extend unit.



**MUX12**

**MUX11**

**MUX14**

1. **Buffer Format**

In this case rs, rt and rd varies depending on the instruction type, as for R-format rd will always be operand1. The buffers for the other stages are the same for all instructions, some fields might not be applicable depending on the instruction being traced.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **opcode** | **rs (operand2)** | **rt (operand3)** | **rd (operand1)** | **shamt** | **funct** | **PC** |

**IF/ID (R-Format)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **opcode** | **rs** | **rt** | **offset** | **PC** |

**IF/ID (I-Type**

**ID/EX**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **rd** | **rt** | **offset (if applicable)** | **readData2** | **readData1** | **PC** |

**EX/MEM**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **destination** | **readData2** | **aluResult** | **zero** | **address** |

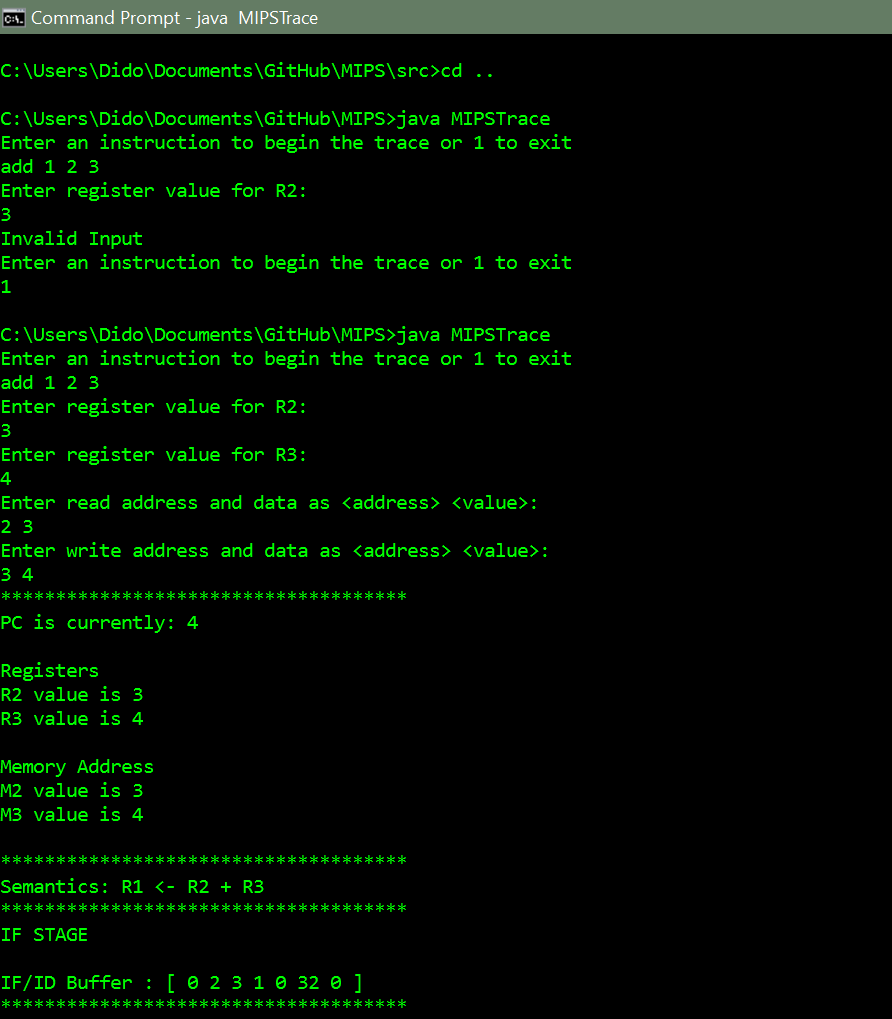
**MEM/WB**

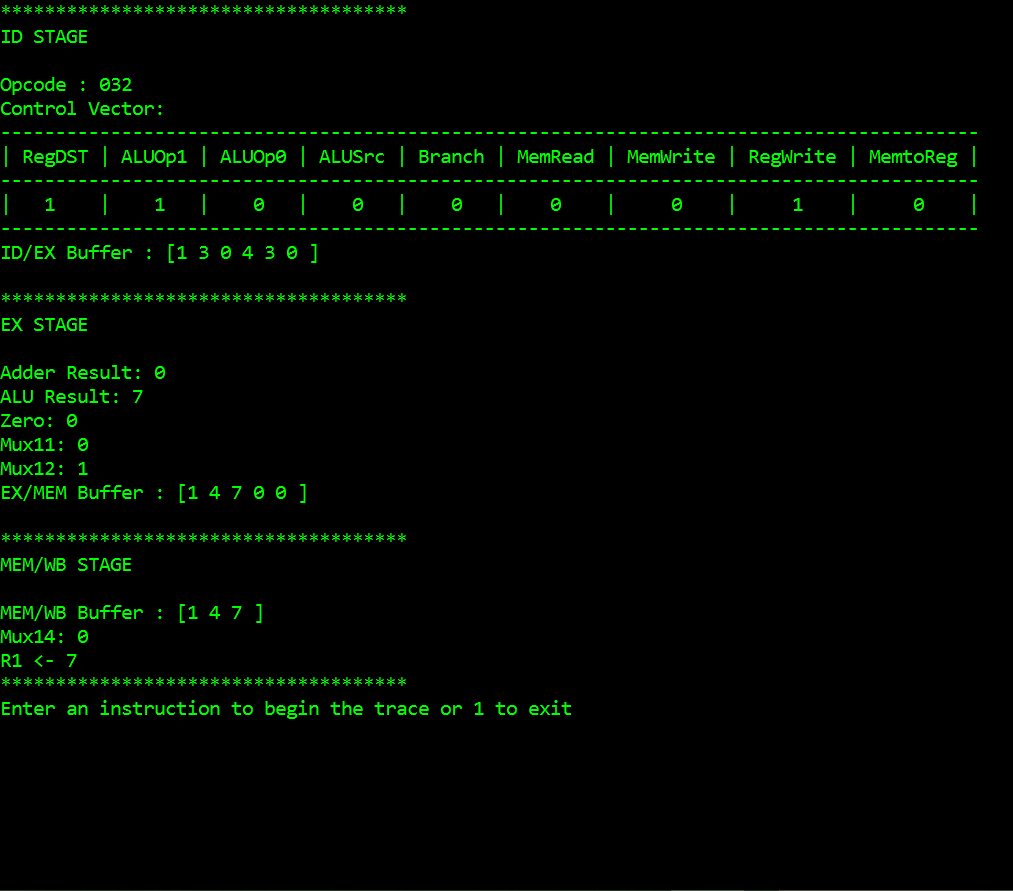
|  |  |  |
| --- | --- | --- |
| **destination** | **readData2** | **readData** |

**TESTING**

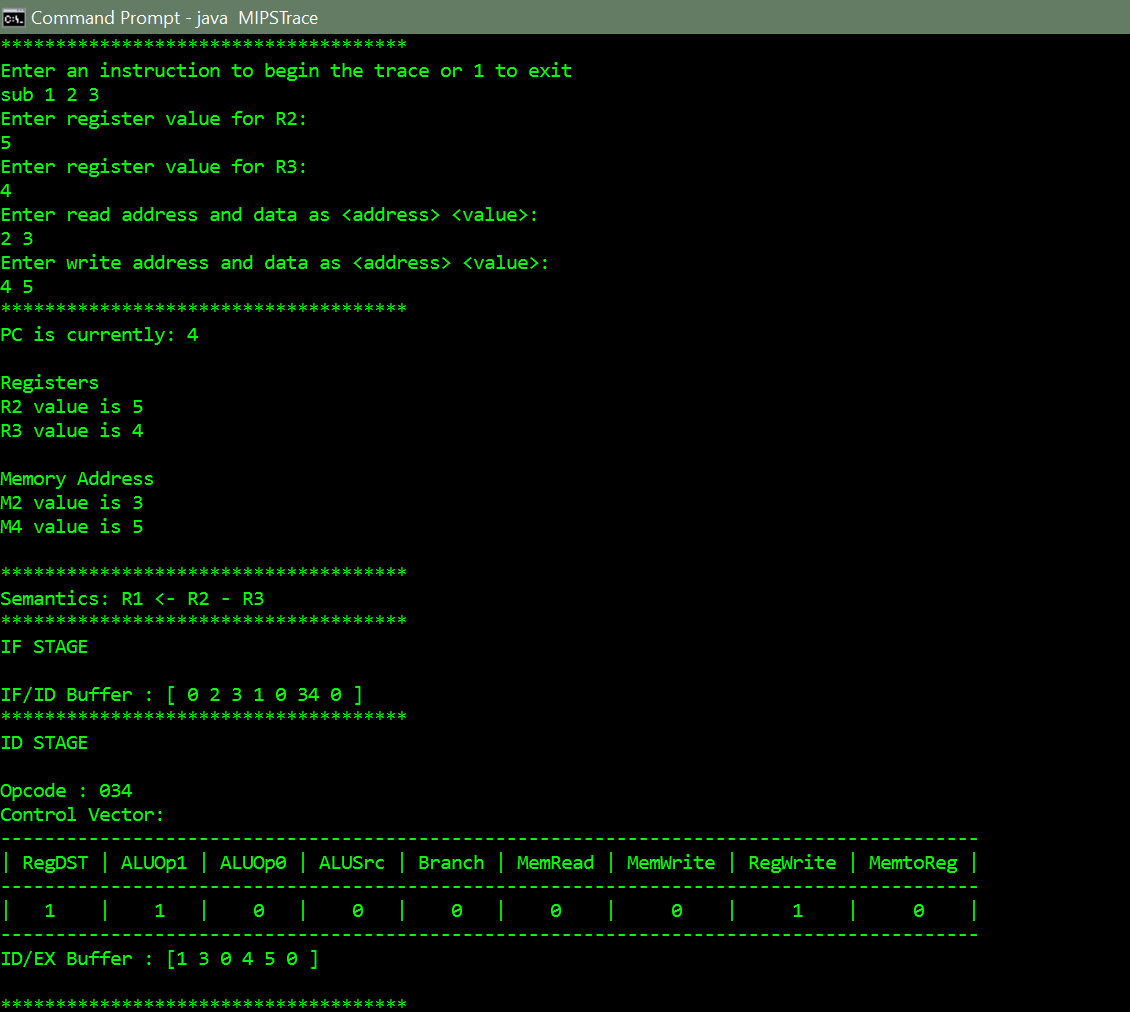
For the tests we will show the program tracing for all the instructions listed above which are; add, sub, lw, sw and beq.

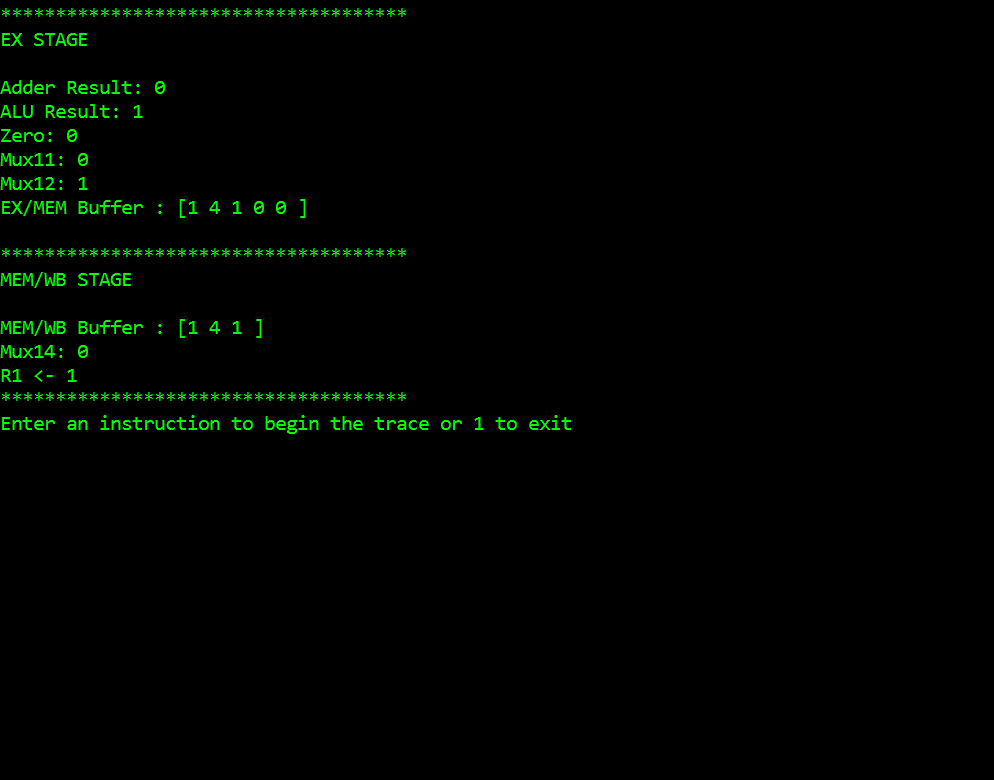
**(i)Add**



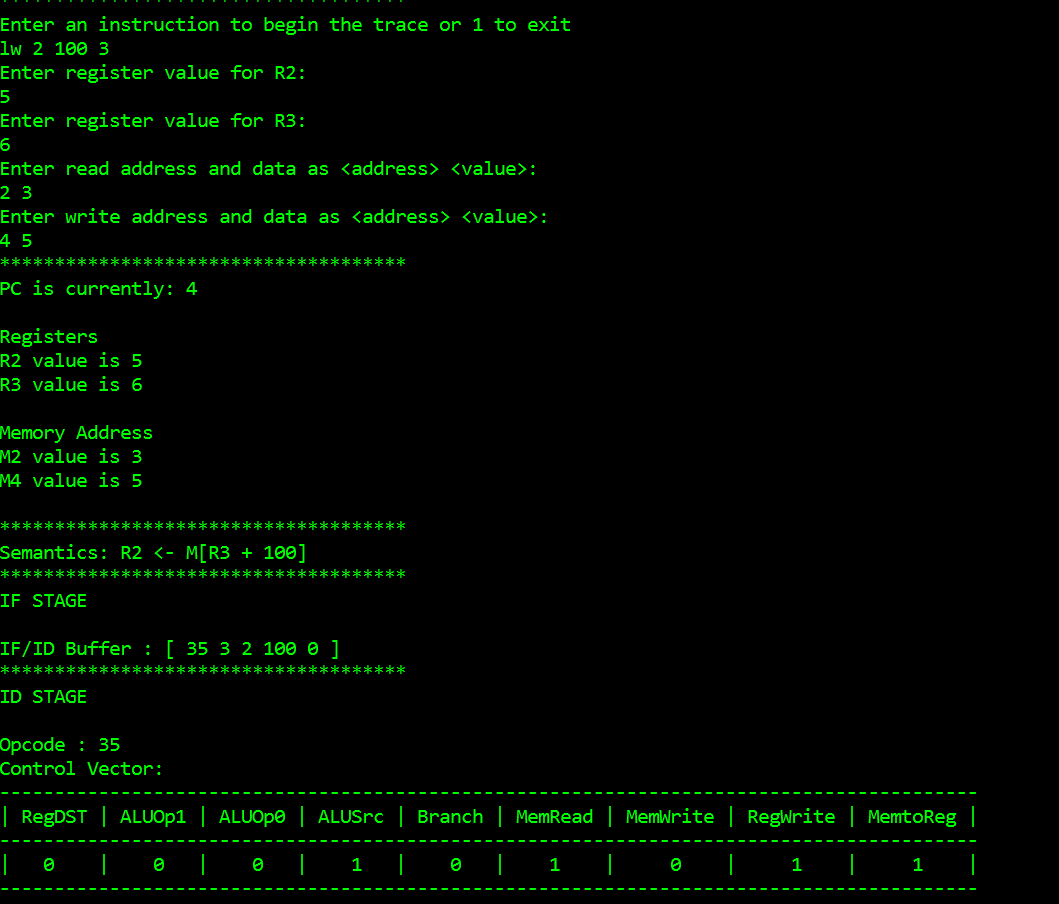


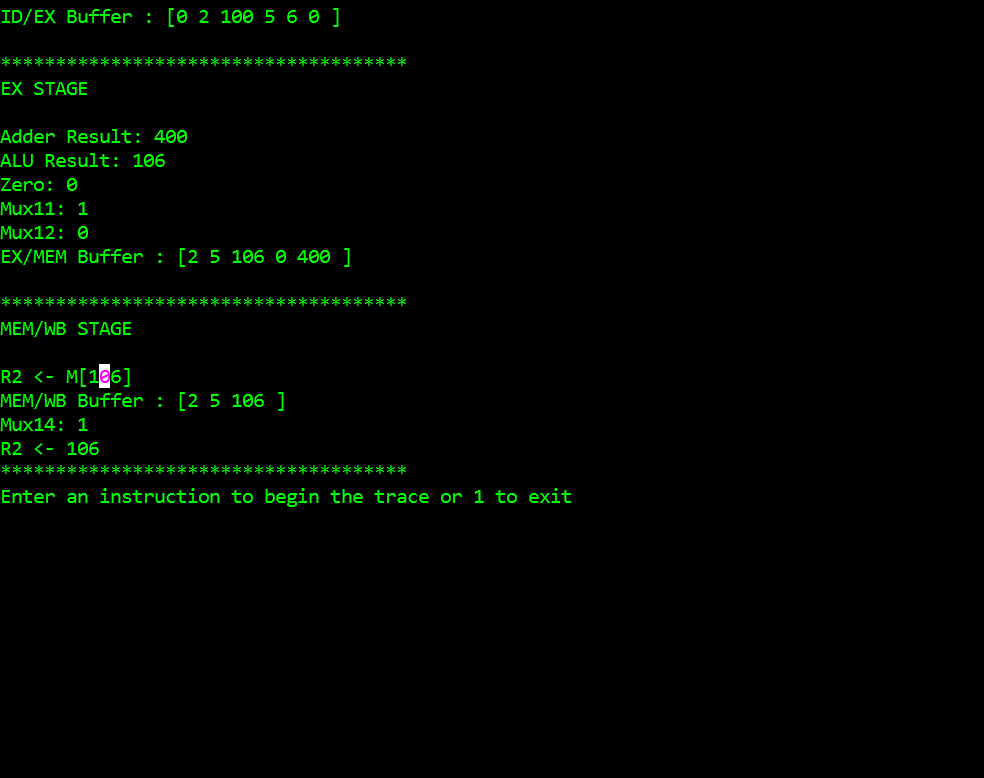
**(ii)Sub**



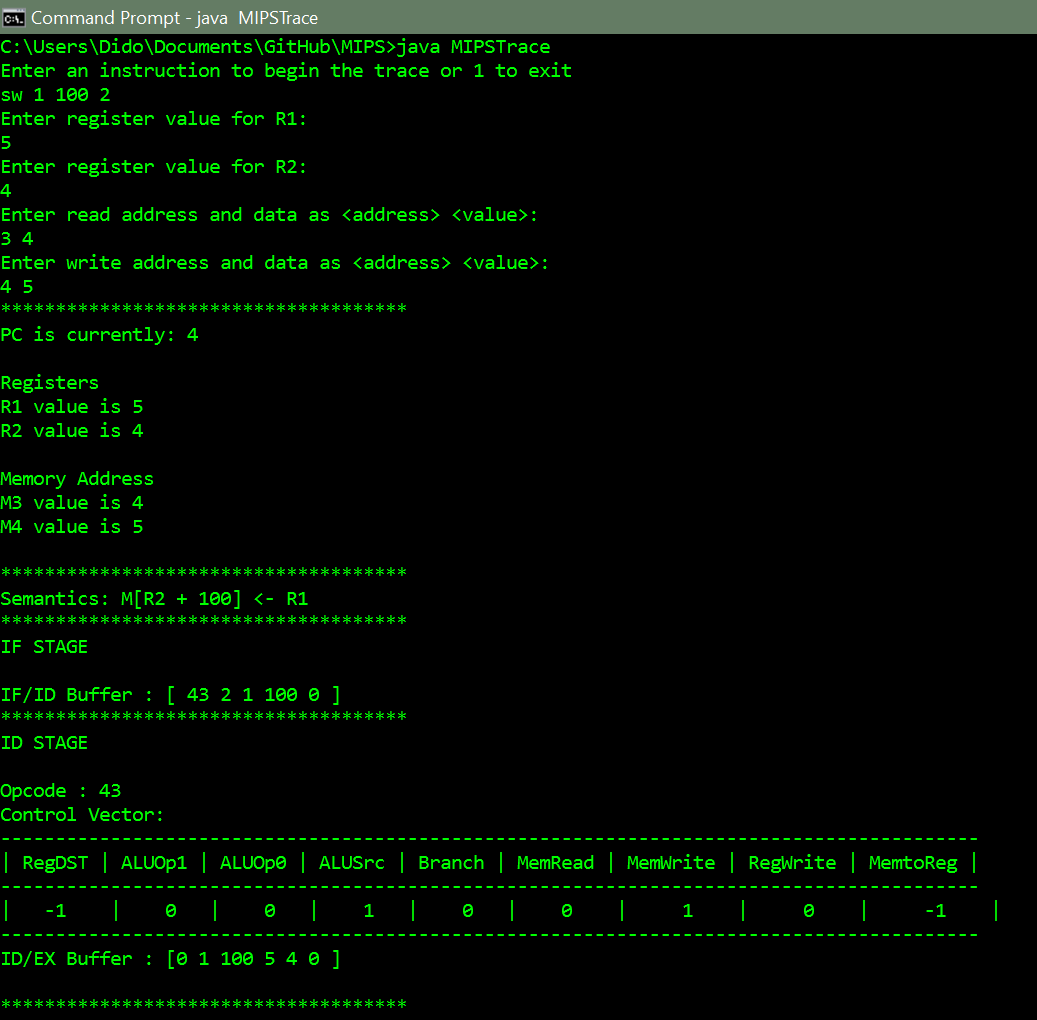


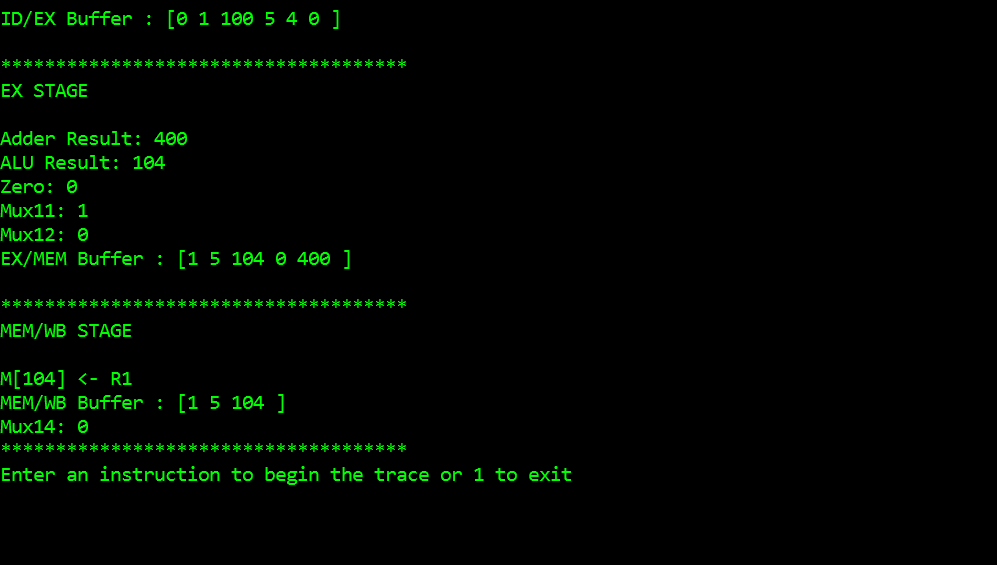
**(iii)LW**





**(iv)SW**





**BEQ**

