**Experiment No 9: Pipelining Visualization**

|  |  |  |
| --- | --- | --- |
| **Sl No** | **Name** | **ID No** |
| **1** | **VISHWAS VASUKI GAUTAM** | **2019A3PS0443H** |

The aim of this experiment is to understand the basic principles of pipelining including the problems of data and branch hazards with the help of Ripes tool. Download the tool from CMS. Please pay attention to the demonstration given by the instructor.

Assume for all the following exercises the first instruction will get loaded in into IF stage 0th clock cycle (instead of 1st clock cycle as per our normal assumption)

**Exercise 9.1 Implement a simple RISC V program containing only add (add t0, t1, t2) instruction and explore (edit registers t1 and t2 to have non zero value.**

1. **For instruction add t0, t1, t2, briefly explain the things that happen (signal changes, Register changes and other effects if any) as the instruction goes through different pipeline stages.**

Answer: IF Stage:   
The program counter gets incremented, the instruction is fetched from the instruction memory using the program counter and all the signals are enabled in the IF staged. This instruction encoding is passed to the pipeline register

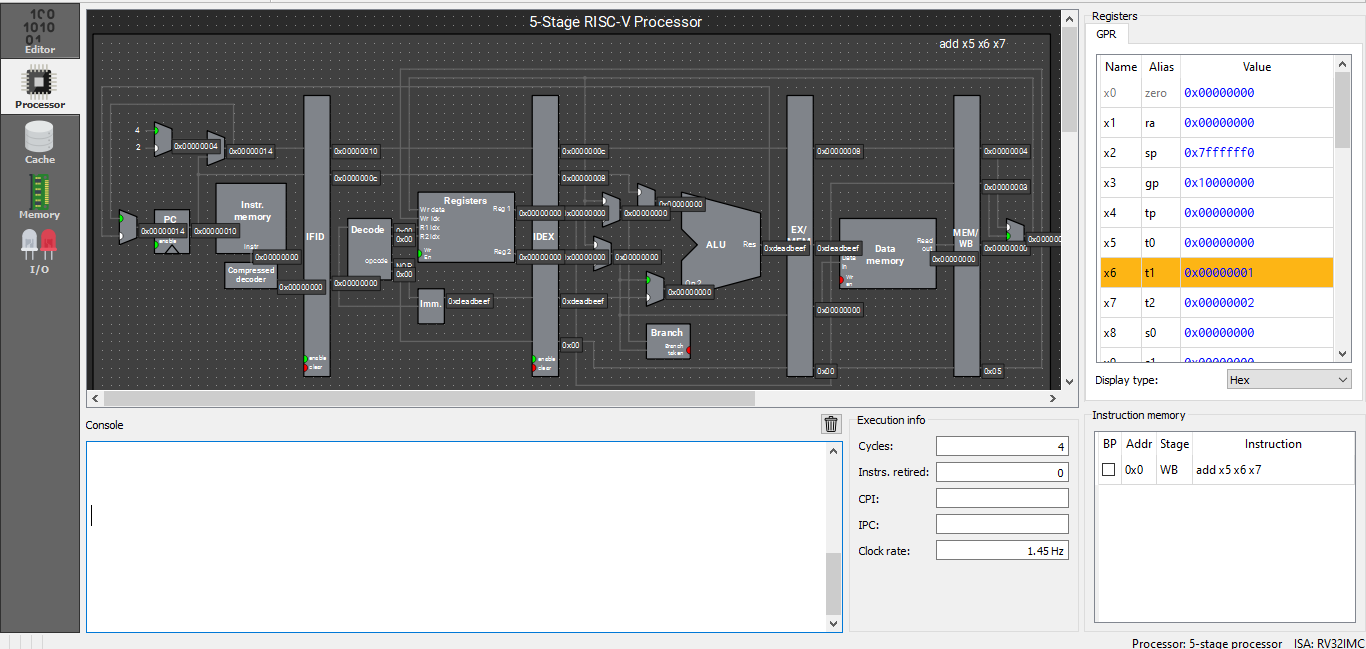
ID Stage:   
In the ID stage the instruction is decoding according to its instruction encoding and the t1, t2 values are extracted from the instruction encoding format. In addition the regWrite control signal and the write register is passed through to the future pipeline. All the other outputs and control signals obtained are passed to the next pipeline register stage

EX Stage:   
In the EX stage, the data in the registers are computed according to the operation specified by the control signals by the ALU and the this result is passed on to the next pipeline stage. Also, the other control signals, write register is passed onto the next stage. According to the instruction t1 and t2 are added in this stage.

MEM Stage:   
Here the data memory is not used since add is an R-type instruction and they don’t make use of the data memory. In general this stage is used to extract the contents or write contents into the data memory as per the EX stage result.

WB Stage: Here, the content in the EX result is written to the destination register t0 using the passed on regWrite control signal and the write register number.

1. **Copy the image of the pipeline at the end of 4th cycle (Show the value of important Signals).**

Answer: 

1. **How many clock cycles does it take for the result of the operation to be available in the destination register?**

Answer:5 (assuming that the instruction is loaded in the 0th cc)

1. **In which pipeline stages do different arithmetic instructions differ?**

Answer: EX stage

1. **One stage is not used by arithmetic instructions. Which one? Why?**

Answer: Data memory stage

**Exercise 9.2 Write the code below and test the Behaviour for 5-stage pipelined processor without forwarding**

**(Save non zero values in t1, t2, t3 and t4 register)**

**Code:**

**add t1, t2, t3**

**add t4, t1, t2**

**List the values stored in registers**

**t1: 0x4**

**t2: 0x2**

**t3: 0x5**

**t4: 0x1**

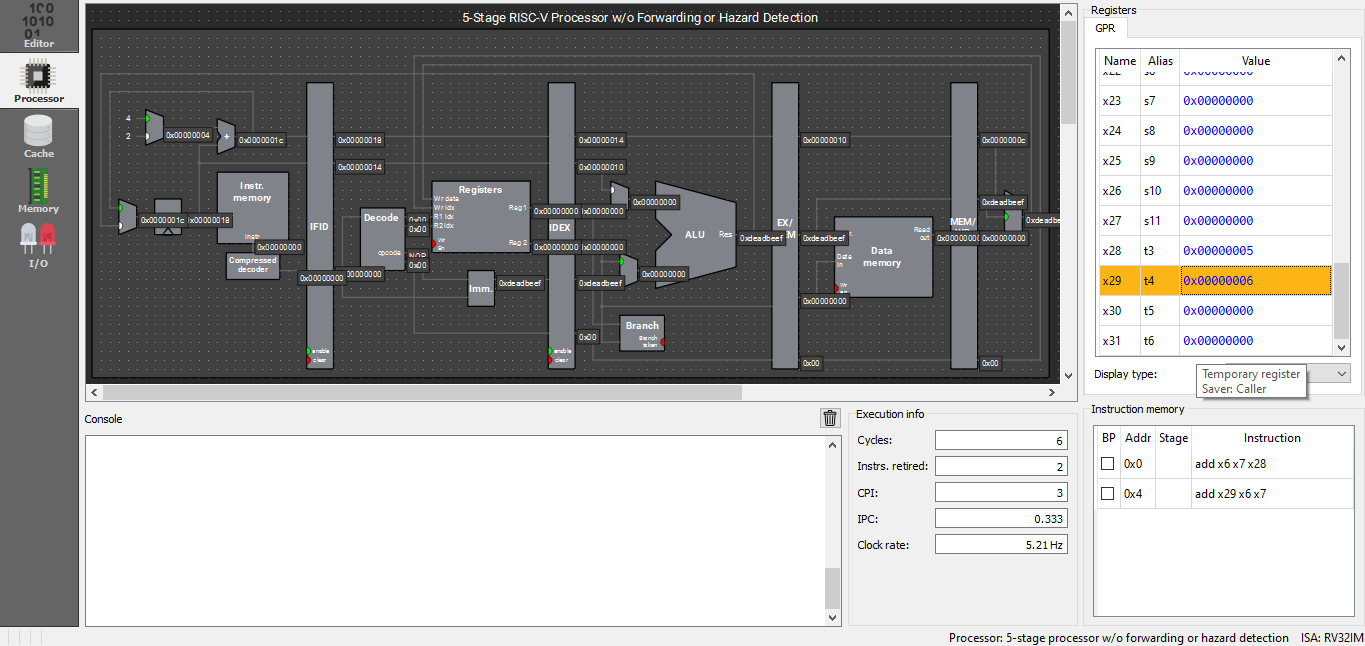
1. **At the end of which clock cycle will the destination register of the first add instruction (t1) gets updated?**

Answer: 5

1. **At the end of which clock cycle is the value of t1 needed in the second instruction?**

Answer: 3

1. **Copy the image of stage table after the completion of execution of second instruction.**

Answer: 

1. **What is the expected and actual value of t4 after the execution of second instruction?**

**Expected Value: 9**

**Actual Value: 6**

1. **What is the problem here? What is this kind of hazard called?**

Answer: Data hazard exists in such a case. And the it has to wait for the result of the previous instruction to execute.

**Exercise 9.3 Write the code below and test the Behaviour for 5-stage pipelined processor with forwarding**

**(Save non zero values in t1, t2, t3 and t4 register)**

**Code:**

**add t1, t2, t3**

**add t4, t1, t2**

**List the values stored in registers**

**t1: 0x4**

**t2: 0x2**

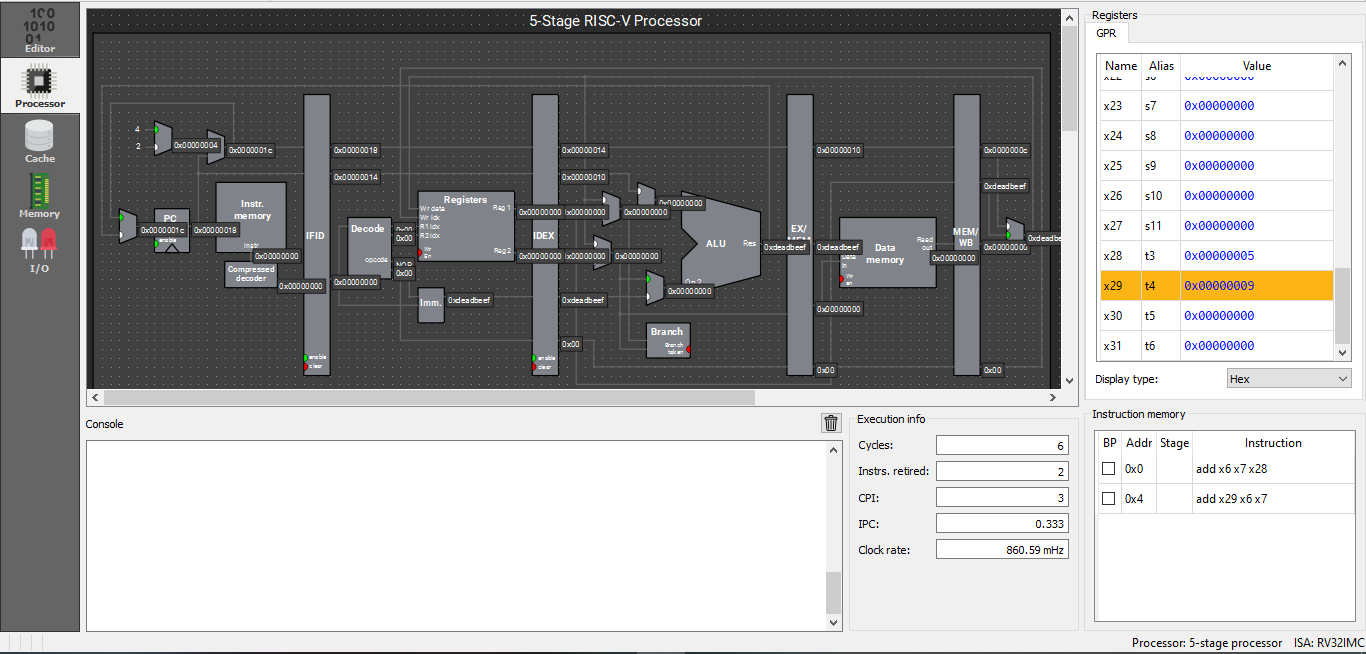
**t3: 0x5**

**t4: 0x1**

1. **At the end of which clock cycle will the destination register of the first add instruction, i.e. t1, gets updated?**

Answer: 5

1. **Copy the image of stage table after the completion of execution of second instruction.**

Answer: 

1. **What is the expected and actual value of t4 after the execution of second instruction?**

**Expected Value: 9**

**Actual Value: 9**

1. **How was the problem in the previous exercise resolved?**

Answer: **Forwarding solved the previous problem**

**Exercise 9.4 Write the code below and test the Behaviour for 5-stage pipelined processor without forwarding and without hazard detection**

**(Save non zero values in t1, t2 and t3. The default starting address of data memory is 0x10000000. In this case the data at address 0x10000000 will get loaded in t0, Check the memory to verify is 0x12340000 is stored.**

**Code:**

**.data**

**a: .word 0x12340000**

**.text**

**la t0, a**

**lw t1, 0(t0)**

**add t3, t1, t2**

**OR**

**(Save non zero values in t1, t2 and t3, let t0=0. In this case the data at address 0 will get loaded in t1, Check the memory to identify the data loaded 0th address [it will be equal to instruction code of first instruction i.e. lw t1, 0(t0)])**

**Code:**

**lw t1, 0(t0)**

**add t3, t1, t2**

**List the values stored in registers**

**t1: 1**

**t2: 2**

**t3: 3**

**List the data value stored in memory location pointed by t0: 0x2a303**

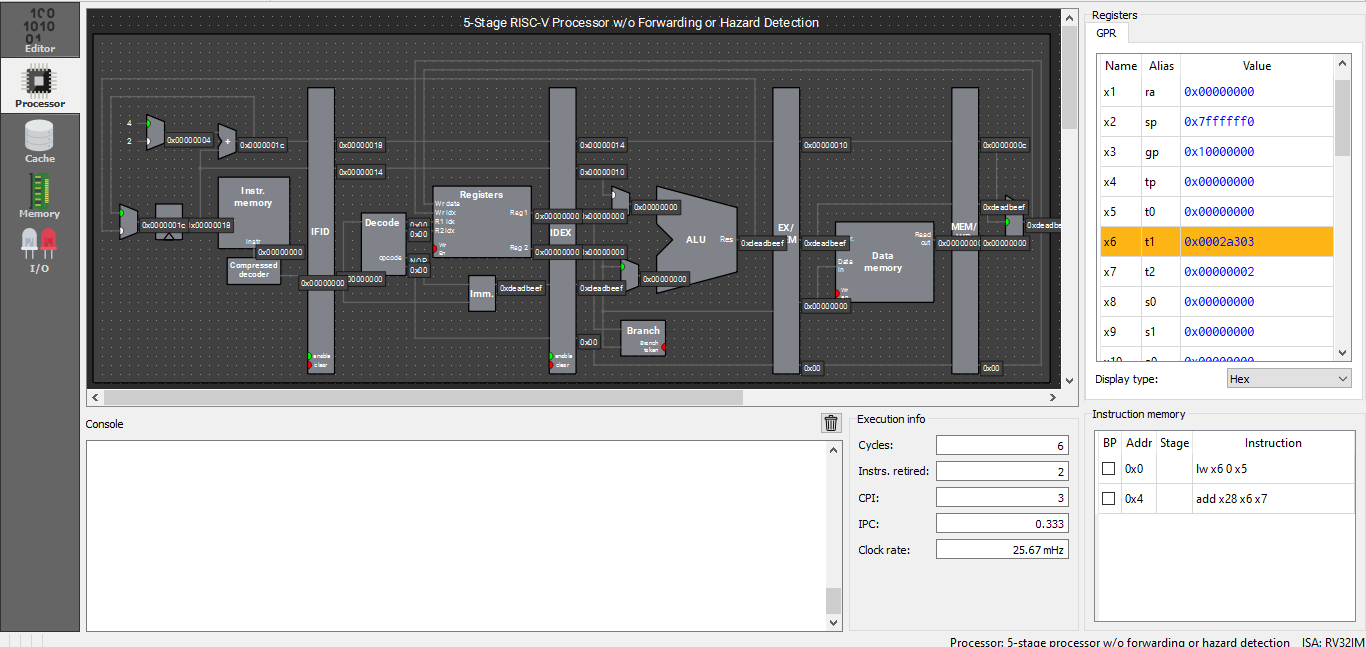
1. **At the end of which clock cycle will the destination register of the lw, i.e. t1, gets updated?**

Answer: **4**

1. **At the end of which clock cycle is the value of t1 needed in the add t3, t1, t2 instruction?**

Answer:2

1. **Copy the image of stage table after the completion of execution of add t3, t1, t2 instruction.**

Answer: 

1. **What is the expected and actual value of t3 after the execution of add t3, t1, t2 instruction?**

**Expected Value: 0x2a305**

**Actual Value: 0x3**

1. **What is the problem here? What is this kind of hazard called?**

Answer: No hazard detection unit. This is also data hazard.

**Exercise 9.5 Write the code below and test the Behaviour for 5-stage pipelined processor with forwarding and without hazard detection**

**(Save non zero values in t1, t2 and t3. The default starting address of data memory is 0x10000000. In this case the data at address 0x10000000 will get loaded in t0, Check the memory to verify is 0x12340000 is stored.**

**Code:**

**.data**

**a: .word 0x12340000**

**.text**

**la t0, a**

**lw t1, 0(t0)**

**add t3, t1, t2**

**OR**

**(Save non zero values in t1, t2 and t3, let t0=0. In this case the data at address 0 will get loaded in t1, Check the memory to identify the data loaded 0th address [it will be equal to instruction code of first instruction i.e. lw t1, 0(t0)])**

**Code:**

**lw t1, 0(t0)**

**add t3, t1, t2**

**List the values stored in registers**

**t1: 1**

**t2: 2**

**t3: 3**

**List the data value stored in memory location pointed by t0: 0x2a303**

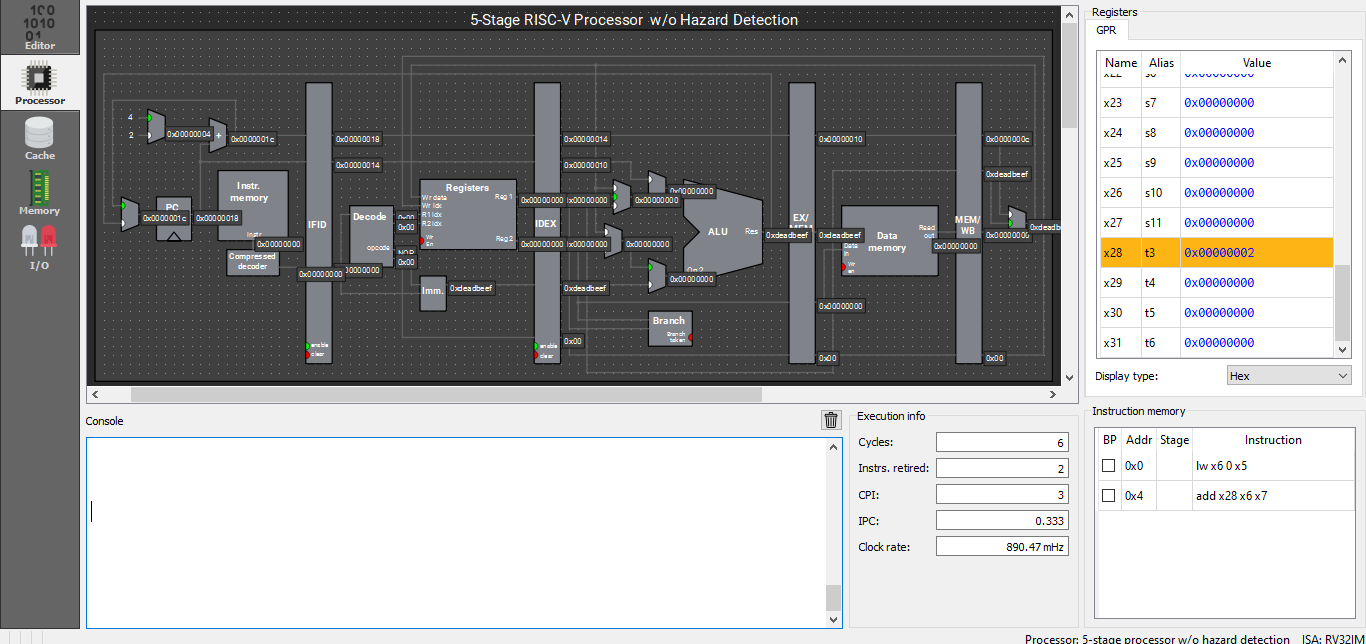
1. **At the end of which clock cycle will the data, that is to be written in to t1, be ready?**

Answer: **4**

1. **At the end of which clock cycle is the value of t1 needed in the add t3, t1, t2 instruction?**

Answer: 2

1. **Copy the image of stage table after the completion of execution of add t3, t1, t2 instruction.**

Answer: 

1. **What is the expected and actual value of t3 after the execution of add t3, t1, t2 instruction?**

**Expected Value: 0x2a305**

**Actual Value: 2**

1. **Why was the problem not resolved even after adding forwarding Unit?**

Answer: Because it is load instruction and needs a 1cc stall. It is load-use data hazard, and not a normal data hazard

**Exercise 9.5 Write the code below and test the Behaviour for 5-stage pipelined processor with forwarding and with hazard detection**

**(Save non zero values in t1, t2 and t3. The default starting address of data memory is 0x10000000. In this case the data at address 0x10000000 will get loaded in t0, Check the memory to verify is 0x12340000 is stored.**

**Code:**

**.data**

**a: .word 0x12340000**

**.text**

**la t0, a**

**lw t1, 0(t0)**

**add t3, t1, t2**

**OR**

**(Save non zero values in t1, t2 and t3, let t0=0. In this case the data at address 0 will get loaded in t1, Check the memory to identify the data loaded 0th address [it will be equal to instruction code of first instruction i.e. lw t1, 0(t0)])**

**Code:**

**lw t1, 0(t0)**

**add t3, t1, t2**

**List the values stored in registers**

**t1: 1**

**t2: 2**

**t3: 3**

**List the data value stored in memory location pointed by t0: 0x2a303**

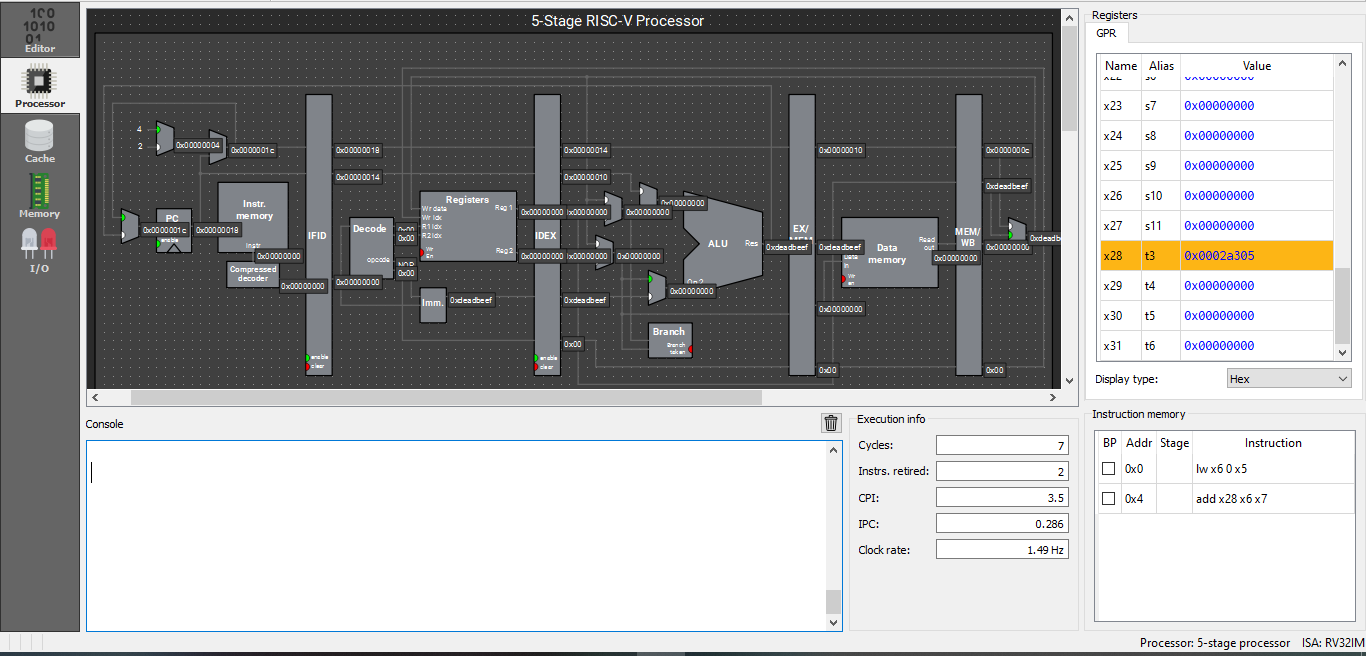
1. **At the end of which clock cycle will the data, that is to be written in to t1, ready?**

Answer: 2

1. **At the end of which clock cycle is the value of t1 needed in the add t3, t1, t2 instruction (including stall)?**

Answer: 2

1. **Copy the image of stage table after the completion of execution of add t3, t1, t2 instruction.**

Answer: 

1. **What is the expected and actual value of t3 after the execution of add t3, t1, t2 instruction?**

**Expected Value: 0x2a305**

**Actual Value: 0x2a305**

1. **How was the problem (encountered in previous two exercises) resolved?**

Answer: Hazard detection unit made a 1cc stall so now the t1 could be forward to the appropriate stage

**Exercise 9.6 Write the code below and test the Behaviour for 5-stage pipelined processor with forwarding and with hazard detection (this data path uses static Branch NOT TAKEN scheme)**

**Code**

**beq t1, t0, L1**

**sub t4, t3, t2**

**add t1, t2, t3**

**add t1, t2, t3**

**add t1, t2, t3**

**L1: sub t4, t2, t3**

**Case 1 (t0==t1) (Store Non zero values in t0, t1, t2, t3, t4 and make sure t0 is equal to t1)**

**List the values stored in registers**

**t0: 1**

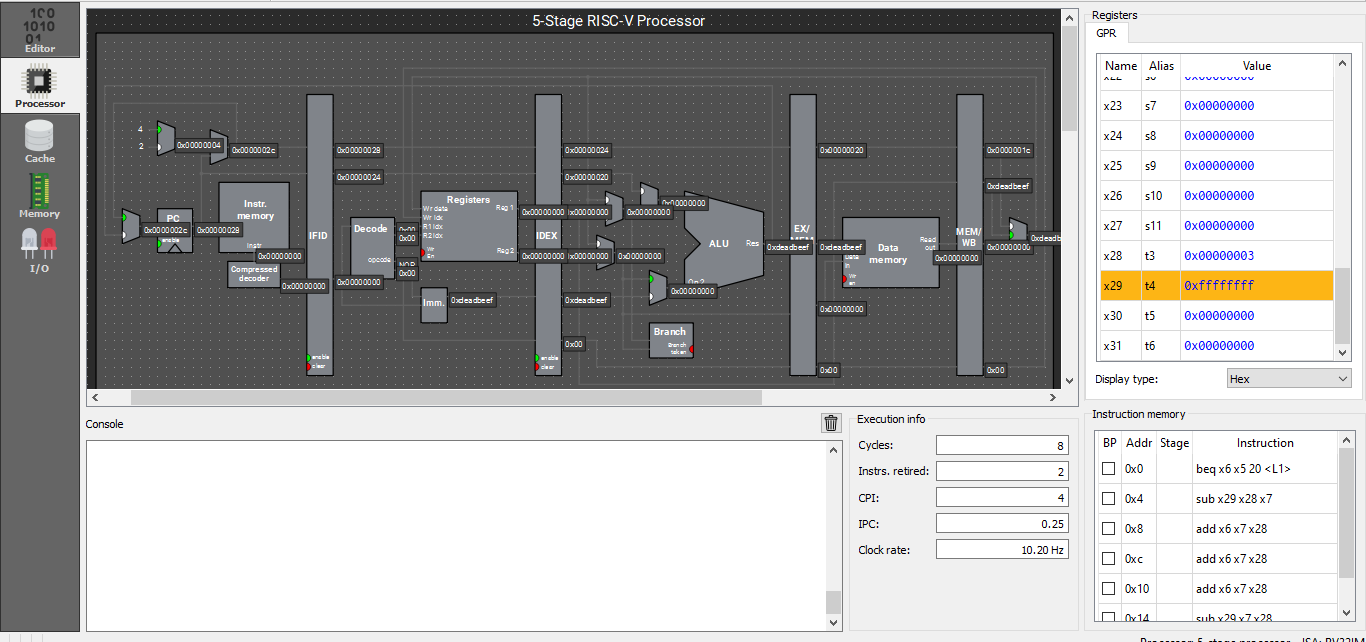
**t1: 1**

**t2: 2**

**t3: 3**

**t4: 4**

1. **Copy the image of stage table after the completion of execution.**

Answer: ****

1. **Explain the pipelined processor operation for this case.**

Answer: When branch instruction is executed at the end of 2nd cc, the processor knows that it needs to take the branch. So it flushes the pipeline register as it had previously assumed that the branch will not be taken. And later fetches the correct instr

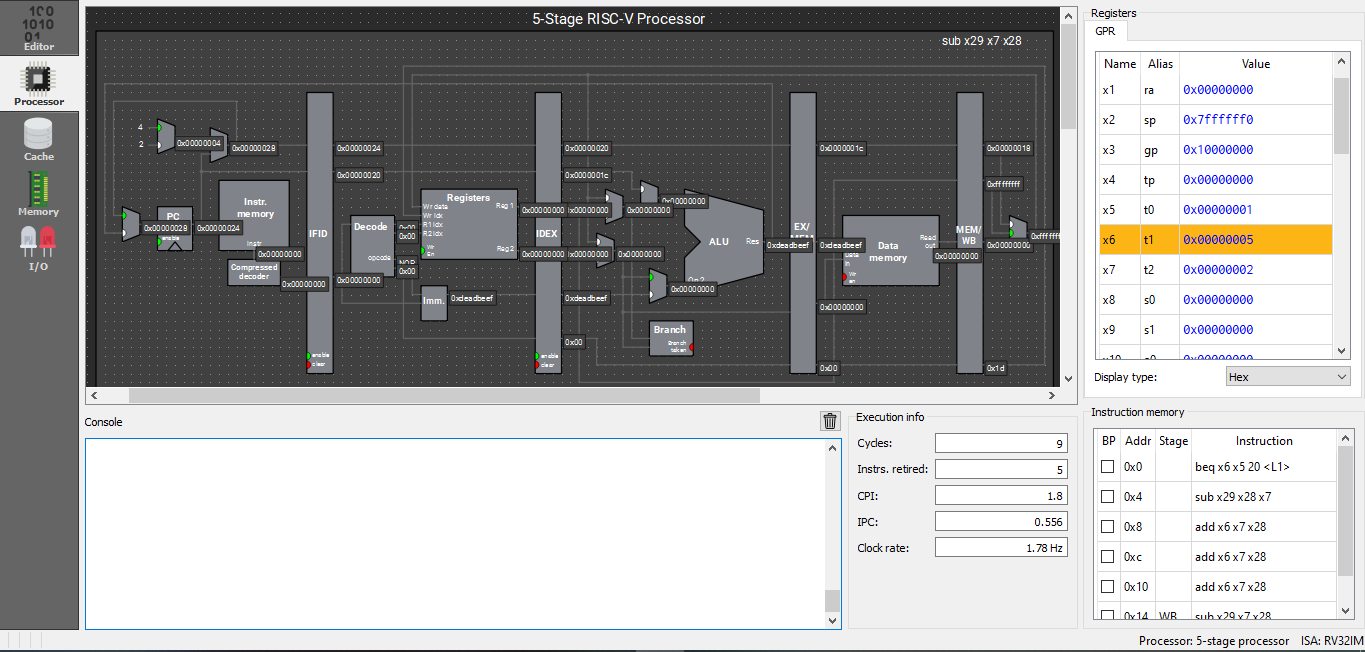
1. **What is the expected and actual value of t4 after the completion of execution?**

**Expected Value: 0xffffffff**

**Actual Value: 0xffffffff**

**Case 1 (t0! = t1) (Store Non zero values in t0, t1, t2, t3, t4 and make sure t0 is not equal to t1)**

1. **Copy the image of stage table after the completion of execution.**

Answer: 

1. **Explain the pipelined processor operation for this case.**

Answer: Here the prediction of branch not take is right so it just continues execution and fetches the rest of the instructions.

1. **What is the expected and actual value of t4 after the completion of execution?**

**Expected Value: 0x1**

**Actual Value: 0x1**

**General**

1. **List the concepts you learnt from this experiment (Conclusions/Observations)**

Answer: **In this experiment, we learnt to visualize the pipeline better.**