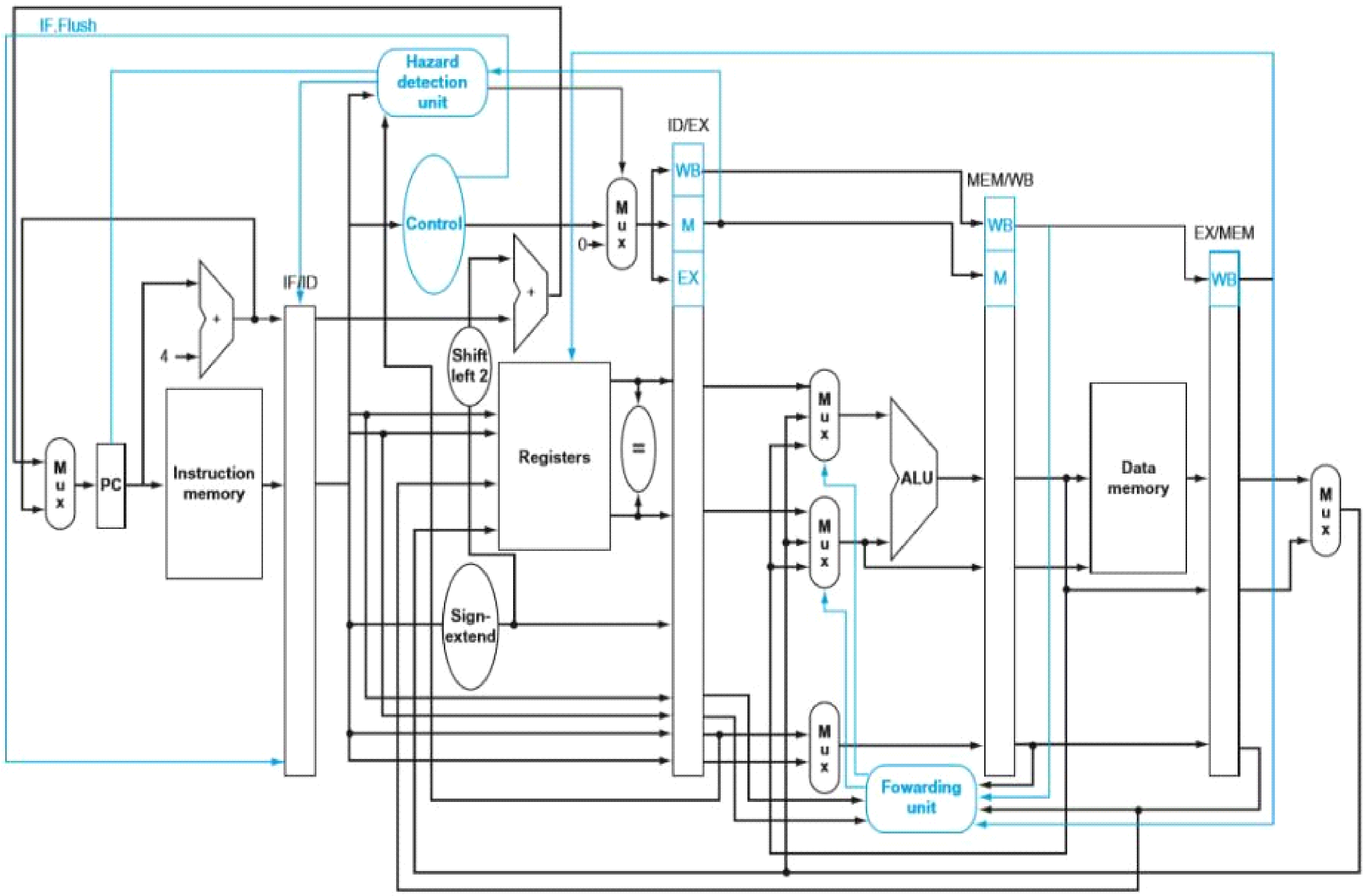
CO Project Report

Implementing pipelined MIPS processor using Verilog



Group: 46

❖ ***we divided our work among the team members into stages:***

1- First of all, we implemented **Single Cycle MIPS Processor**.
2- Then, we implemented the **4 Pipeline Register Files** and their test benches.
3- After that, we implemented **MEM to MEM forwarding** technique.
4- Then, we implemented the **ALU to ALU forwarding** technique.
5- Then, we implemented the **Forwarding to ID stage**.
6- After That, we implemented **Hazard Detection Unit** and its test bench.
7- Then, we implemented the **Branch Prediction Unit**.
8- Finally, we edited the **Top module** implemented in the first stage in order to work with the pipelined version.

9 -The **supported** instructions are [**add,sub** ,**sw**, **lw** , **sll** , **and**, **or**, **beq**].

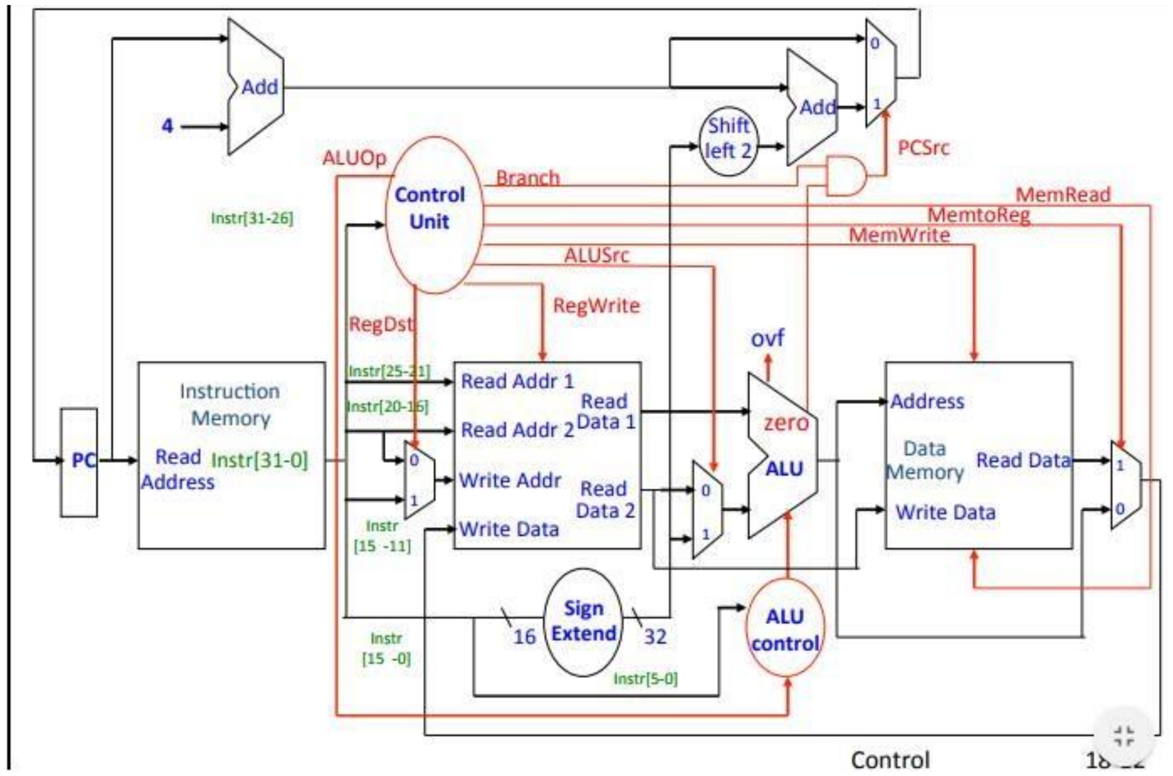


Figure 1: This image represents the single cycle processor

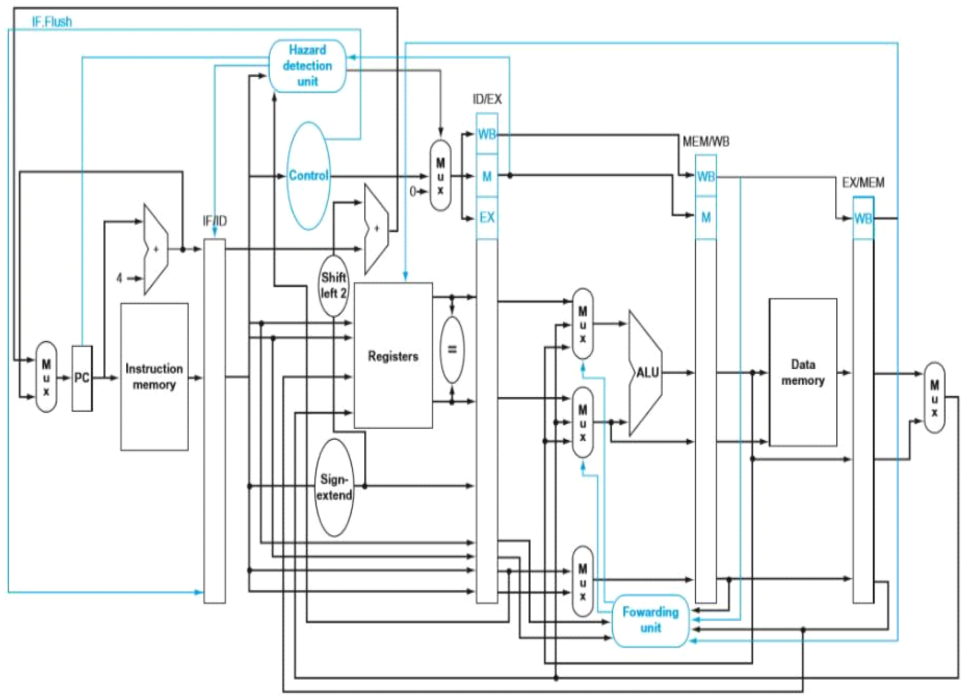


Figure 2: This image represents the pipelined processor.

Modules functionality:

ALU: supports operations [addition, subtraction, shift left logical, and ,or]

Alu control : *in case of lw / sw:* tell the alu to add the offset to the address.

In case of beq: tell the alu to subtract two input registers.

In case of R format: the alu control does the operation depending on the function field.

If function =100000=>add

If function =100010=>subtract

If function =100100=>and

If function =100101=>or

If function =000000=>sll

Register file: Save the data in registers faster than the data memory.

Mux: take two inputs and selection, and determines the output depending on the selection.

Control unit: takes op code as an input , and depending on its value it determines all signals of instruction memory , data memory , register file , all muxes , alu control , jumb and determine the branch condition.

Pc: take the current address, and every clock cycle it adds 4 to it , and in case of branch taken jump to the address of label and in case of jump it goes to the jump address.

Shift left logical by 2: it works in case of I format, Taking the immediate and multiply it by 4 to verify the equation: (pc+4)+4*immediate by the help of an adder module.

Adder : It takes 2 inputs and add them.

And: takes the zero flag from the alu , and branch signal from control unit , to determine the branch taken or not.

Sign extension: take 16 bits from the immediate and turns to 32 bits to support negative numbers.

Data memory : it stores large memory, and in case of lw, it reads the word inside it , in case sw it saves the word inside it.

Instruction memory: it stores instructions in binary, in order to use them to perform their required functions.

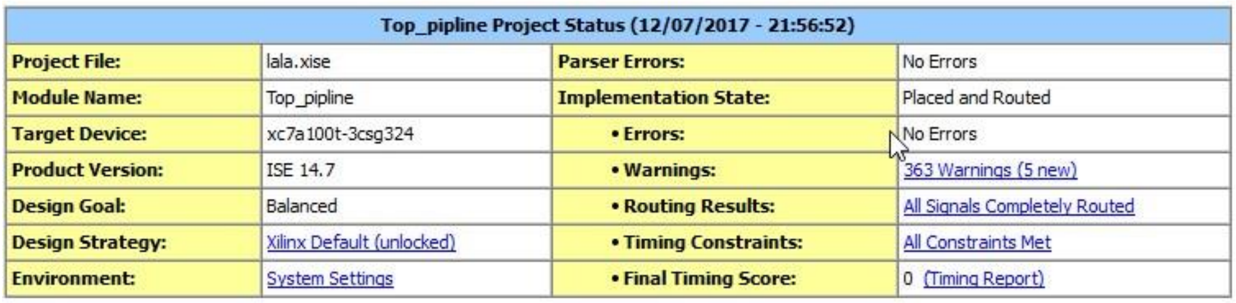
Data Hazard detection unit: detects if the instruction lw is followed by R format instruction.it stalls the pc for one cycle, and it preserves the data in the IF/ID pipeline register for another one cycle , and make all control signals equal to zero.

Prediction unit: it compares the values of the two outputs of the register file , to determine whether the branch will be taken or not.

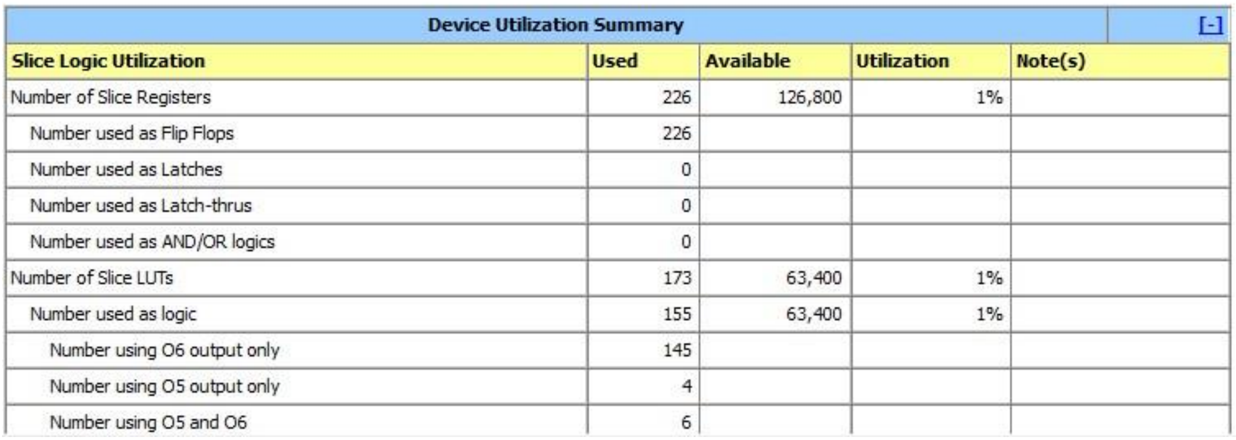
Forwarding: It checks whether there is a dependency between the operands of an instruction , and the result of another one.

***

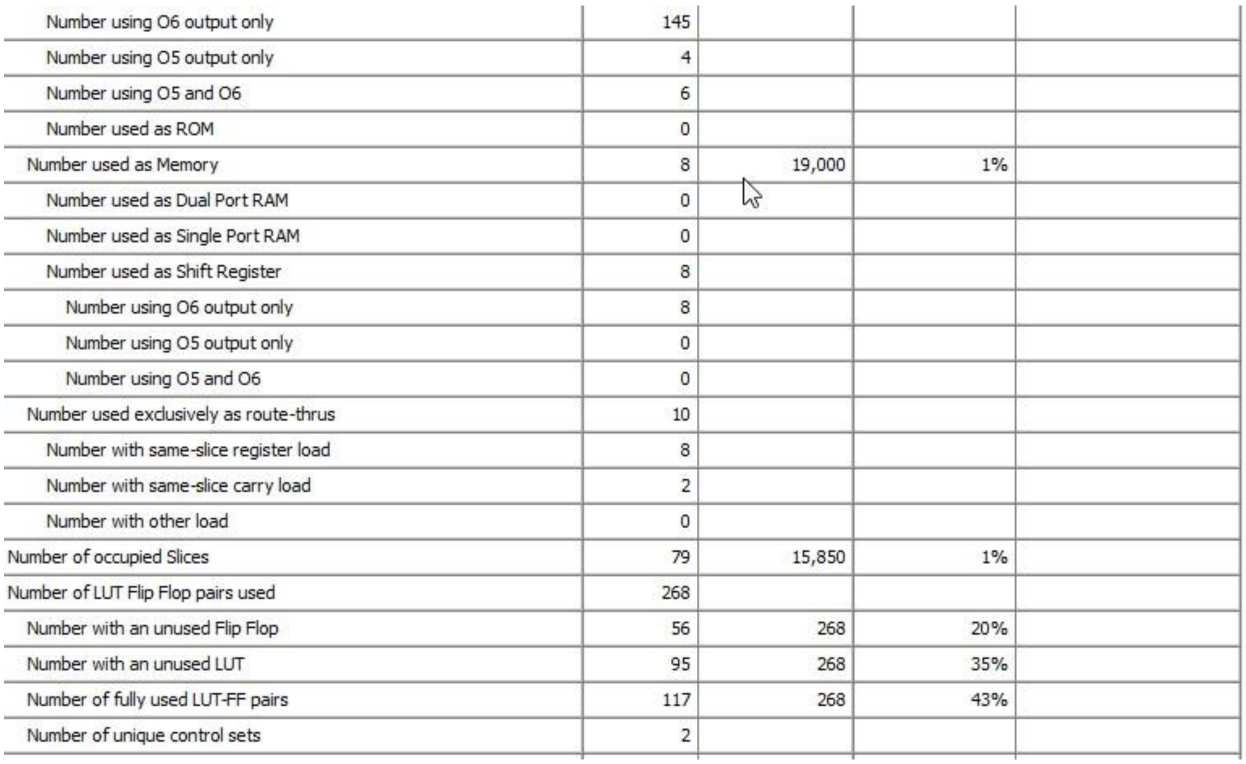
Synth result:



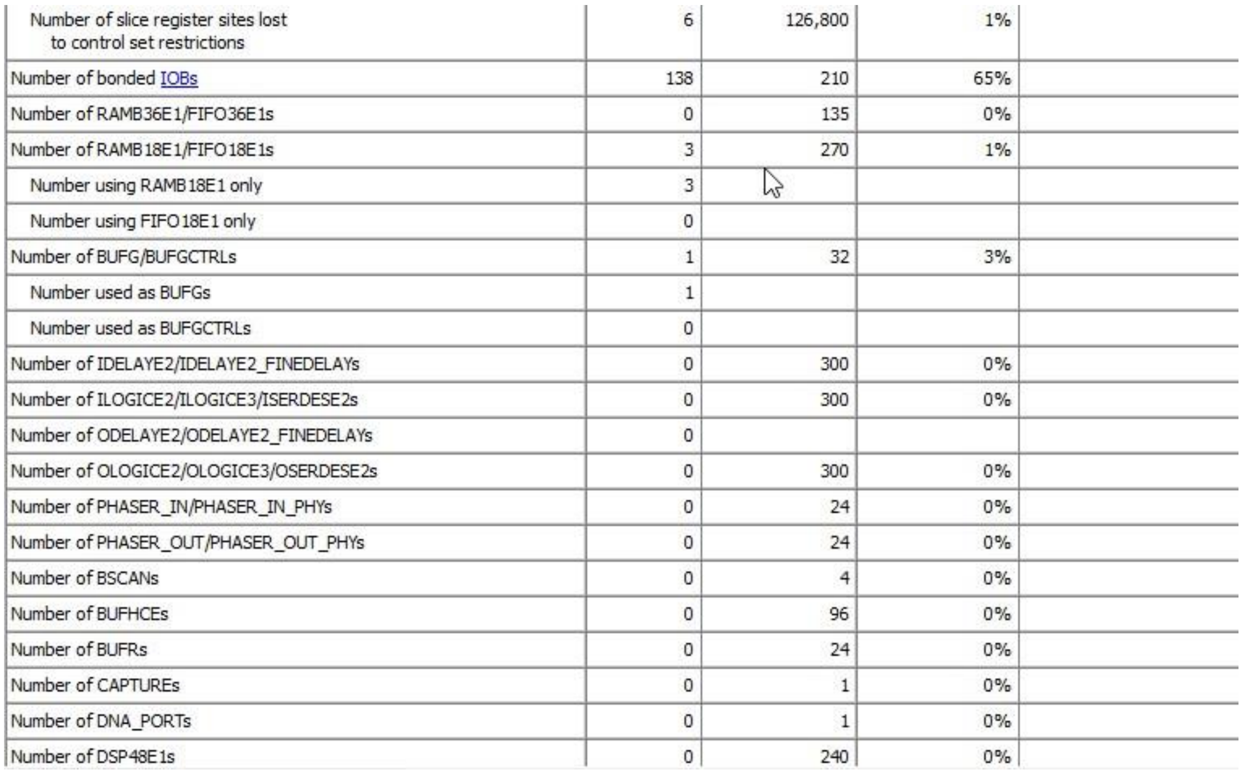
Top_pipline Project Status (12/07/2017 - 21:56:52)			
Project File:	lala.xise	**Parser Errors:**	No Errors
Module Name:	Top_pipline	**Implementation State:**	Placed and Routed
Target Device:	xc7a100t-3csg324	• **Errors:**	No Errors
Product Version:	ISE 14.7	• **Warnings:**	363 Warnings (5 new)
Design Goal:	Balanced	• **Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	All Constraints Met
Environment:	System Settings	• **Final Timing Score:**	0 (Timing Report)



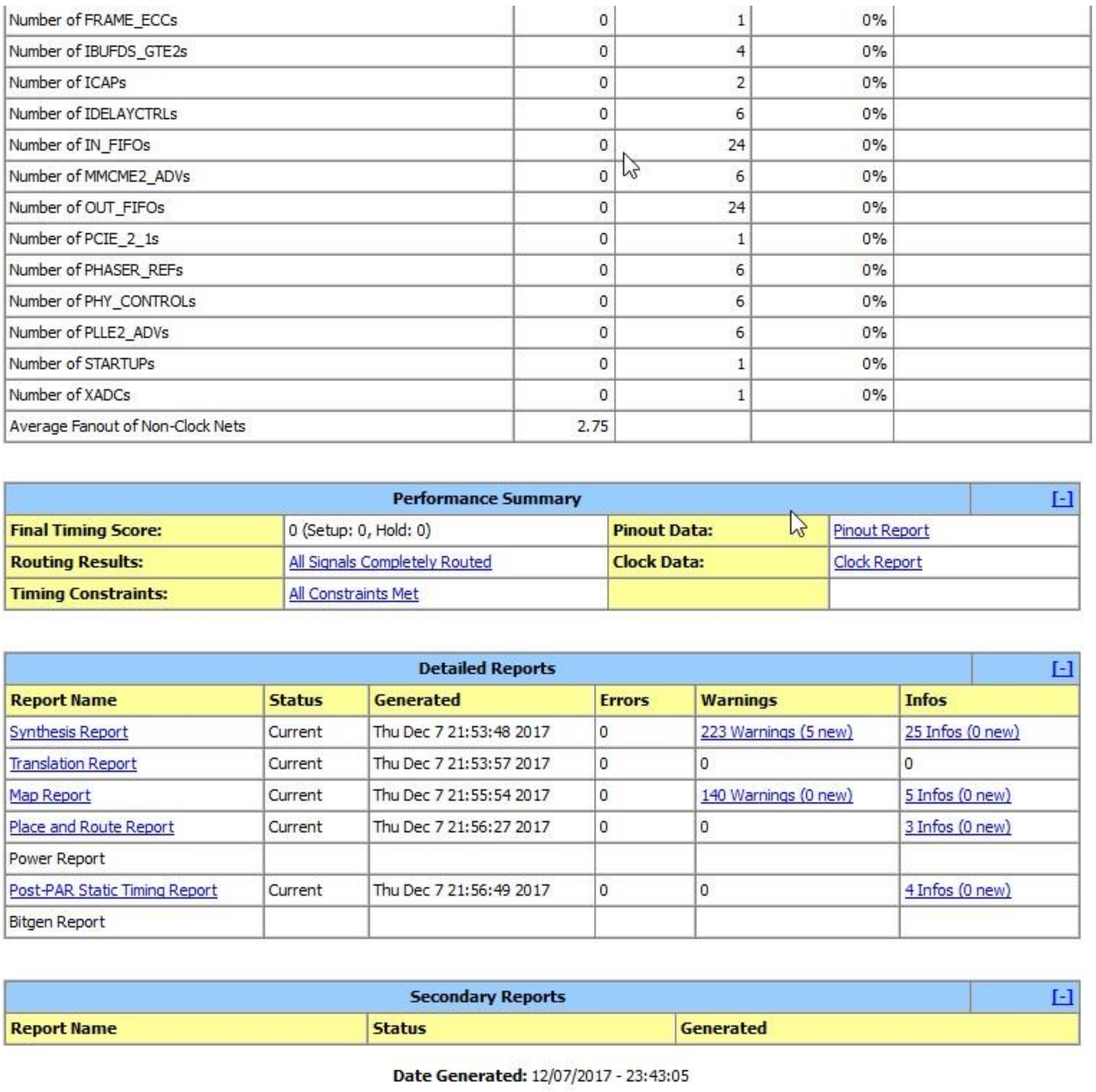
Device Utilization Summary				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	226	126,800	1%	
Number used as Flip Flops	226			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	173	63,400	1%	
Number used as logic	155	63,400	1%	
Number using O6 output only	145			
Number using O5 output only	4			
Number using O5 and O6	6			



Number using O6 output only	145			
Number using O5 output only	4			
Number using O5 and O6	6			
Number used as ROM	0			
Number used as Memory	8	19,000	1%	
Number used as Dual Port RAM	0			
Number used as Single Port RAM	0			
Number used as Shift Register	8			
Number using O6 output only	8			
Number using O5 output only	0			
Number using O5 and O6	0			
Number used exclusively as route-thrus	10			
Number with same-slice register load	8			
Number with same-slice carry load	2			
Number with other load	0			
Number of occupied Slices	79	15,850	1%	
Number of LUT Flip Flop pairs used	268			
Number with an unused Flip Flop	56	268	20%	
Number with an unused LUT	95	268	35%	
Number of fully used LUT-FF pairs	117	268	43%	
Number of unique control sets	2			



Number of slice register sites lost to control set restrictions	6	126,800	1%	
Number of bonded IOBs	138	210	65%	
Number of RAMB36E1/FIFO36E1s	0	135	0%	
Number of RAMB18E1/FIFO18E1s	3	270	1%	
Number using RAMB18E1 only	3			
Number using FIFO18E1 only	0			
Number of BUFG/BUFGCTRLs	1	32	3%	
Number used as BUFGs	1			
Number used as BUFGCTRLs	0			
Number of IDELAYE2/IDELAYE2_FINEDELAYs	0	300	0%	
Number of ILOGICE2/ILOGICE3/ISERDESE2s	0	300	0%	
Number of ODELAYE2/ODELAYE2_FINEDELAYs	0			
Number of OLOGICE2/OLOGICE3/OSERDESE2s	0	300	0%	
Number of PHASER_IN/PHASER_IN_PHYs	0	24	0%	
Number of PHASER_OUT/PHASER_OUT_PHYs	0	24	0%	
Number of BSCANs	0	4	0%	
Number of BUFHCEs	0	96	0%	
Number of BUFRs	0	24	0%	
Number of CAPTUREs	0	1	0%	
Number of DNA_PORTs	0	1	0%	
Number of DSP48E1s	0	240	0%	



Number of FRAME_ECCs	0	1	0%	
Number of IBUFDS_GTE2s	0	4	0%	
Number of ICAPs	0	2	0%	
Number of IDELAYCTRLs	0	6	0%	
Number of IN_FIFOs	0	24	0%	
Number of MMCME2_ADVs	0	6	0%	
Number of OUT_FIFOs	0	24	0%	
Number of PCIE_2_1s	0	1	0%	
Number of PHASER_REFs	0	6	0%	
Number of PHY_CONTROLs	0	6	0%	
Number of PLLE2_ADVs	0	6	0%	
Number of STARTUPs	0	1	0%	
Number of XADCs	0	1	0%	
Average Fanout of Non-Clock Nets	2.75			

Performance Summary			[-]
Final Timing Score:	0 (Setup: 0, Hold: 0)	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:	All Constraints Met		

Detailed Reports [-]

Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Thu Dec 7 21:53:48 2017	0	223 Warnings (5 new)	25 Infos (0 new)
Translation Report	Current	Thu Dec 7 21:53:57 2017	0	0	0
Map Report	Current	Thu Dec 7 21:55:54 2017	0	140 Warnings (0 new)	5 Infos (0 new)
Place and Route Report	Current	Thu Dec 7 21:56:27 2017	0	0	3 Infos (0 new)
Power Report					
Post-PAR Static Timing Report	Current	Thu Dec 7 21:56:49 2017	0	0	4 Infos (0 new)
Bitgen Report					

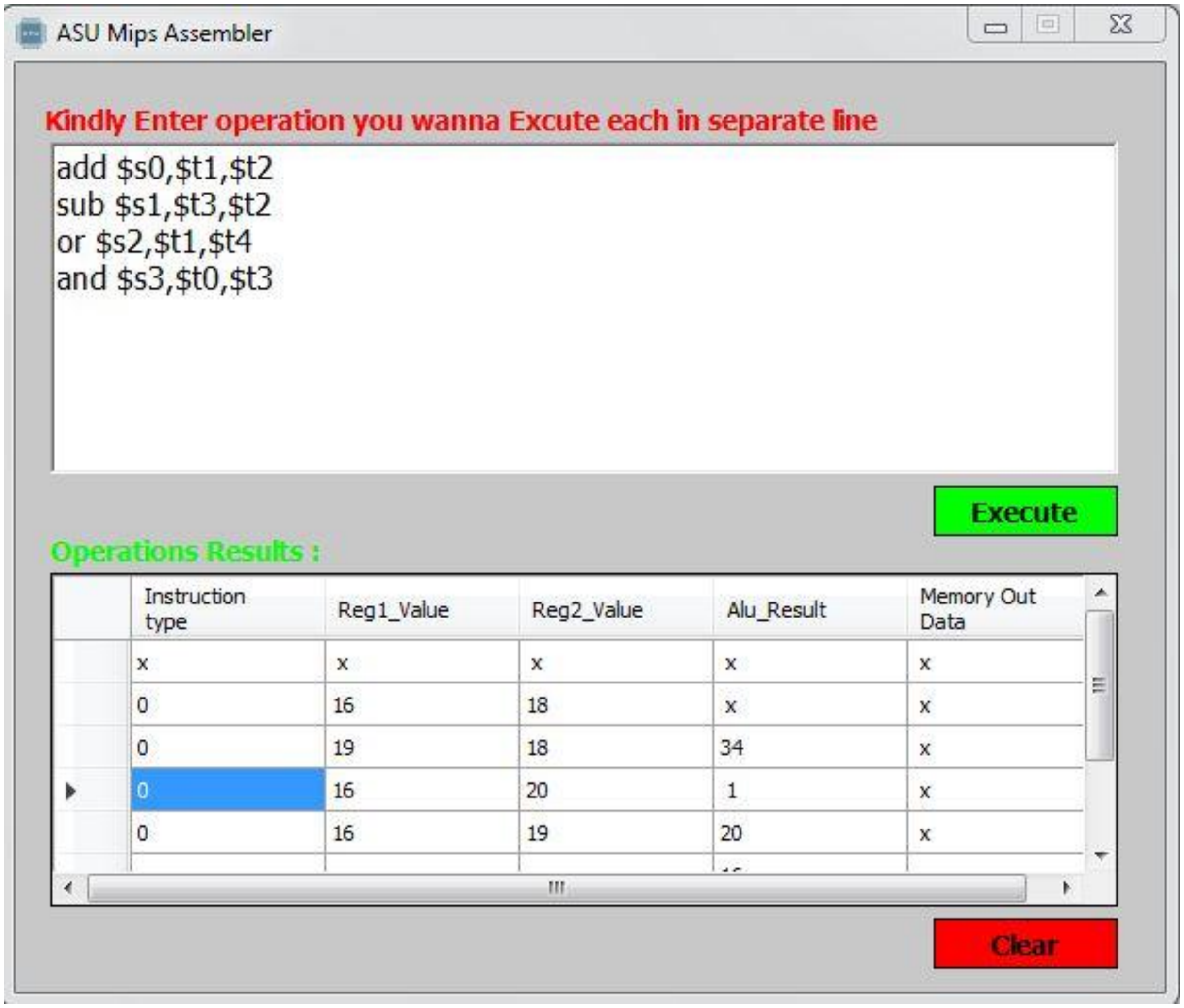
Secondary Reports [-]

Report Name	Status	Generated

Date Generated: 12/07/2017 - 23:43:05

Test Cases.

1-Pipeline Without Hazards.



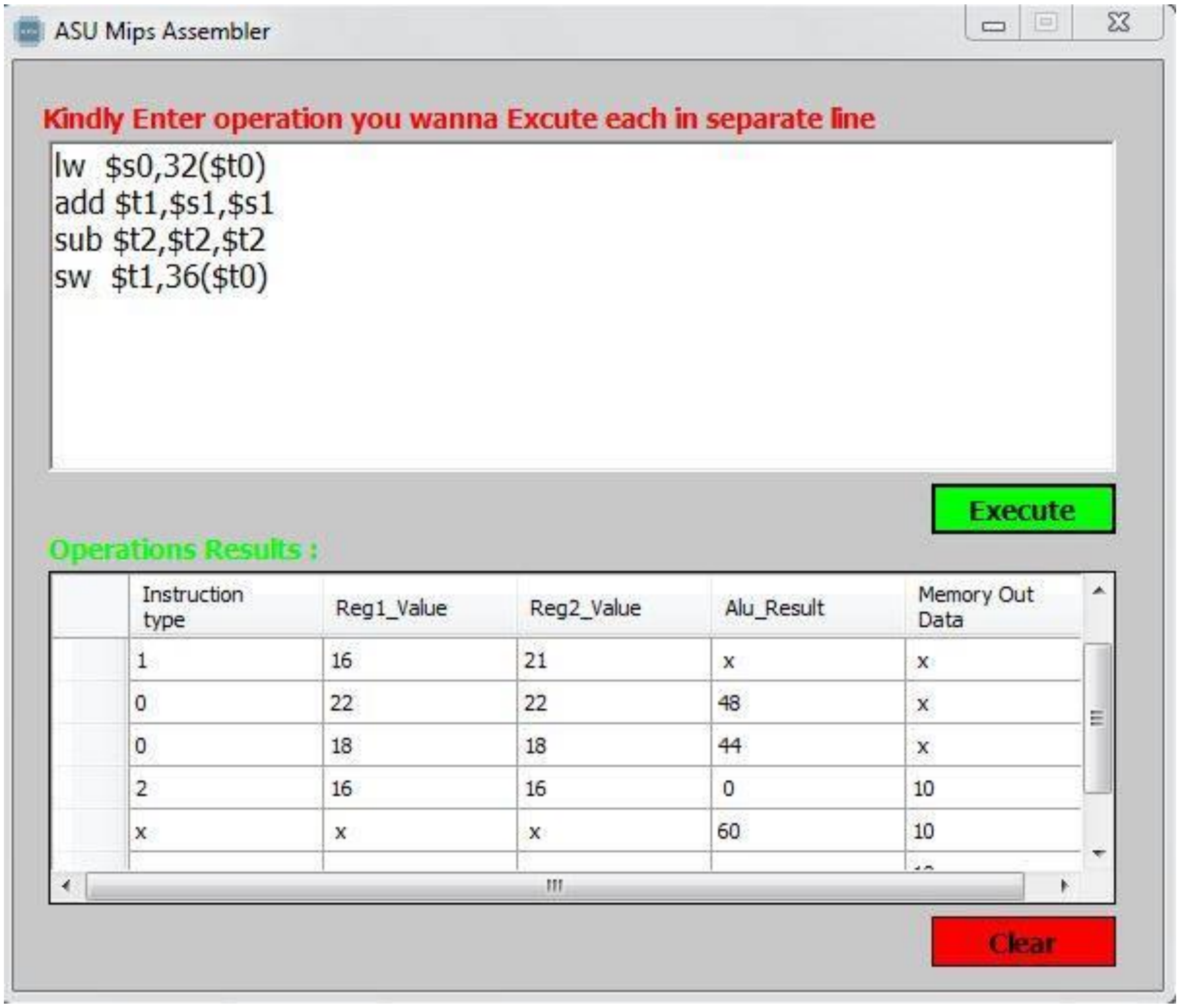
Kindly Enter operation you wanna Excute each in separate line

```
add $s0,$t1,$t2
sub $s1,$t3,$t2
or $s2,$t1,$t4
and $s3,$t0,$t3
```

Operations Results :

Instruction type	Reg1_Value	Reg2_Value	Alu_Result	Memory Out Data
x	x	x	x	x
0	16	18	x	x
0	19	18	34	x
0	16	20	1	x
0	16	19	20	x

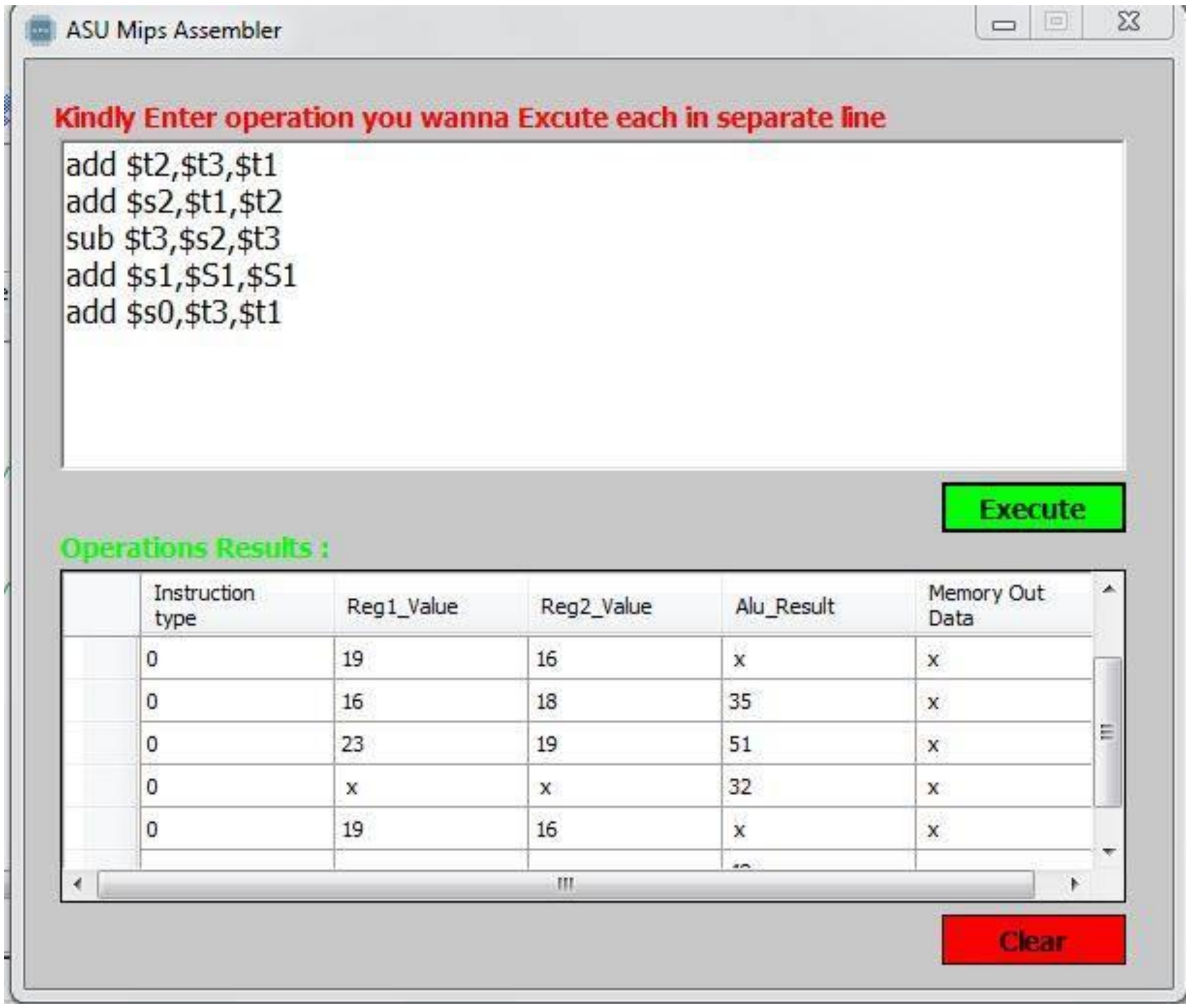
2-Data hazard.


ASU Mips Assembler
Kindly Enter operation you wanna Excute each in separate line
Execute
Operations Results :
Clear

```
lw  $s0,32($t0)
add $t1,$s1,$s1
sub $t2,$t2,$t2
sw  $t1,36($t0)
```

Instruction type	Reg1_Value	Reg2_Value	Alu_Result	Memory Out Data
1	16	21	x	x
0	22	22	48	x
0	18	18	44	x
2	16	16	0	10
x	x	x	60	10

3- ALU to ALU forwarding.

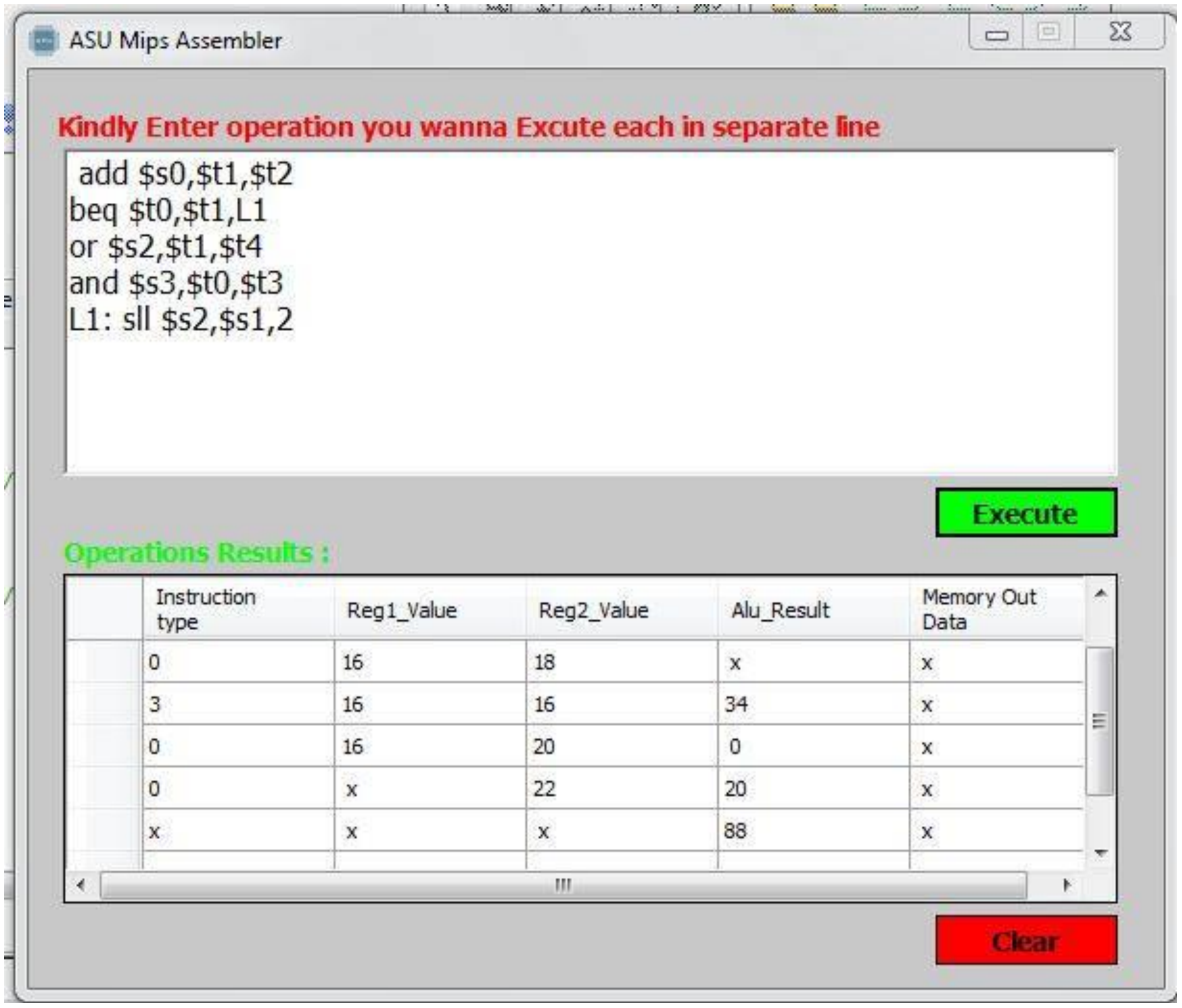


ASU Mips Assembler
Kindly Enter operation you wanna Excute each in separate line
Execute
Operations Results :
Clear

```
add $t2,$t3,$t1
add $s2,$t1,$t2
sub $t3,$s2,$t3
add $s1,$S1,$S1
add $s0,$t3,$t1
```

Instruction type	Reg1_Value	Reg2_Value	Alu_Result	Memory Out Data
0	19	16	x	x
0	16	18	35	x
0	23	19	51	x
0	x	x	32	x
0	19	16	x	x

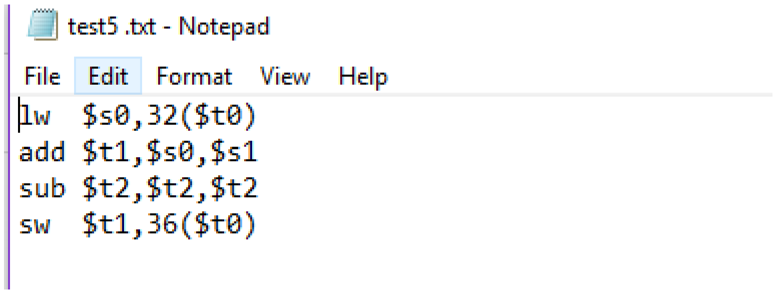
4-Beq not taken.


ASU Mips Assembler
Kindly Enter operation you wanna Excute each in separate line
Execute
Operations Results :
Clear

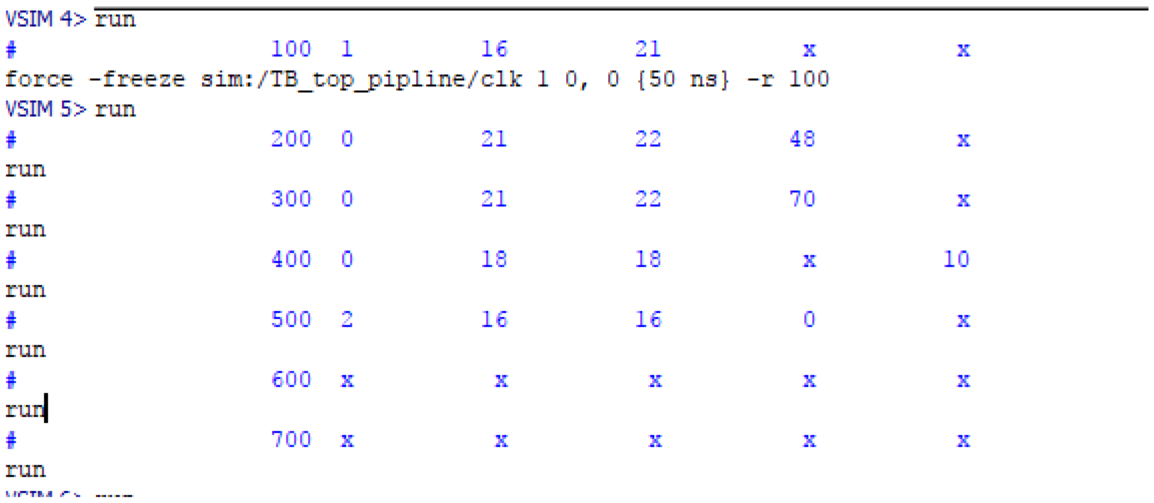
```
 add $s0,$t1,$t2
beq $t0,$t1,L1
or $s2,$t1,$t4
and $s3,$t0,$t3
L1: sll $s2,$s1,2
```

Instruction type	Reg1_Value	Reg2_Value	Alu_Result	Memory Out Data
0	16	18	x	x
3	16	16	34	x
0	16	20	0	x
0	x	22	20	x
x	x	x	88	x

5-Hazard and forwarding together.

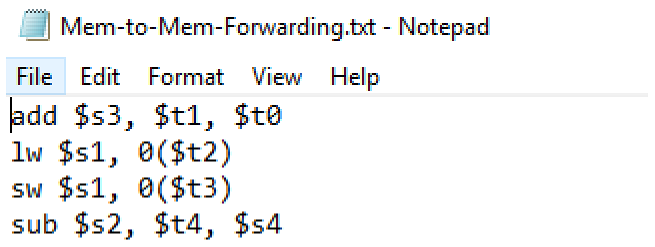


```
lw  $s0,32($t0)
add $t1,$s0,$s1
sub $t2,$t2,$t2
sw  $t1,36($t0)
```



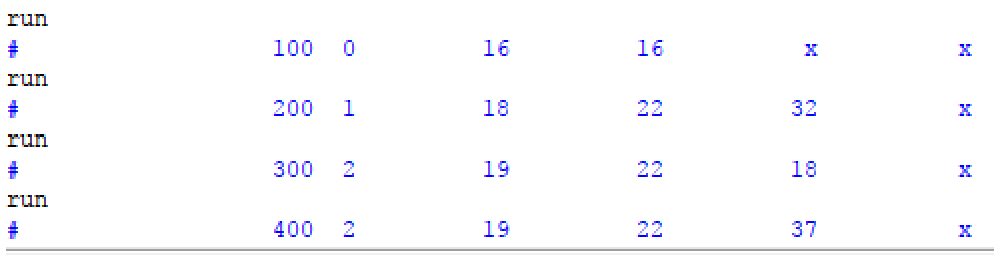
```
VSIM 4> run
#                   100  1         16         21          x          x
force -freeze sim:/TB_top_pipline/clk 1 0, 0 {50 ns} -r 100
VSIM 5> run
#                   200  0         21         22         48          x
run
#                   300  0         21         22         70          x
run
#                   400  0         18         18          x         10
run
#                   500  2         16         16          0          x
run
#                   600  x          x          x          x          x
run
#                   700  x          x          x          x          x
run
```

6-Memory to Memory forwarding.

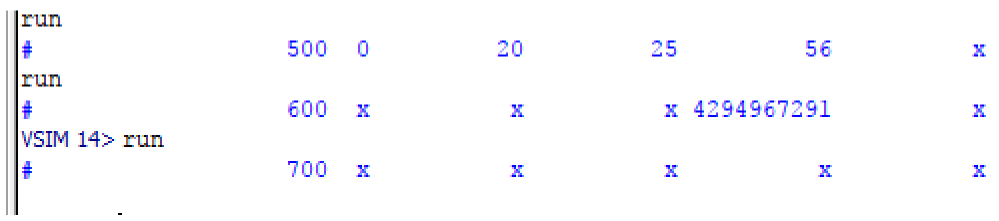


Mem-to-Mem-Forwarding.txt - Notepad

```
add $s3, $t1, $t0
lw $s1, 0($t2)
sw $s1, 0($t3)
sub $s2, $t4, $s4
```



```
run
#                   100  0         16         16          x          x
run
#                   200  1         18         22         32          x
run
#                   300  2         19         22         18          x
run
#                   400  2         19         22         37          x
```



```
run
#                   500  0         20         25         56          x
run
#                   600  x          x          x 4294967291          x
VSIM 14> run
#                   700  x          x          x          x          x
```