Faculty of Engineering
Computer and Systems Engineering Department

MIPS Processor Documentation

Fall (2015-2016)

1- Implementation:

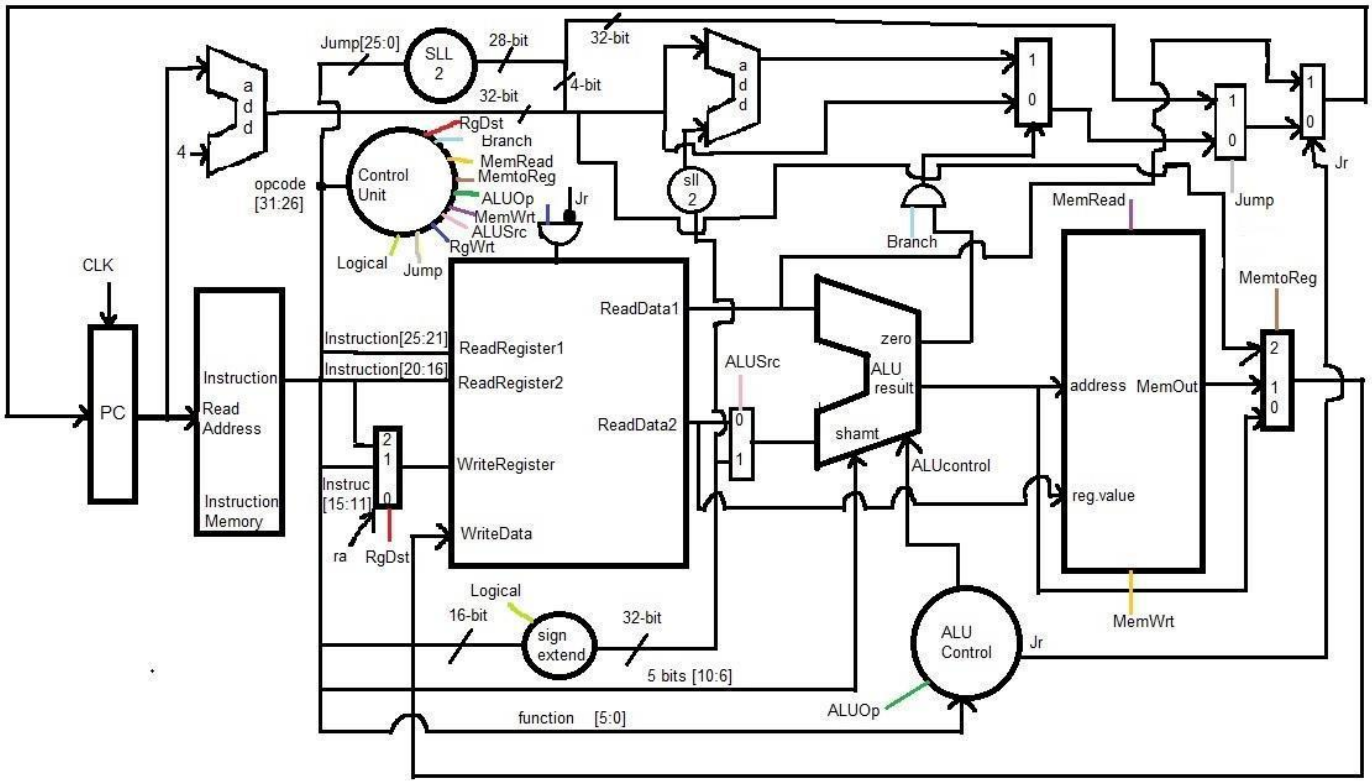
- Main Features:
  - We implemented a description for a fully structural module describing the single-cycle MIPS data-path, by writing and testing a simplified Verilog description. We instantiated other modules described behaviorally, while the top-level module is described structurally.
  - We supported the following instructions set architecture:
    - **Arithmetic:** add, addi.
    - **Load/Store:** lw, sw.
    - **Logic:** sll, and, andi, nor.
    - **Control flow:** beq, jal, jr.
    - **Comparison:** slt.

Language: Verilog, using ActiveHDL

3 programs

- The language we used: Verilog, using ActiveHDL.

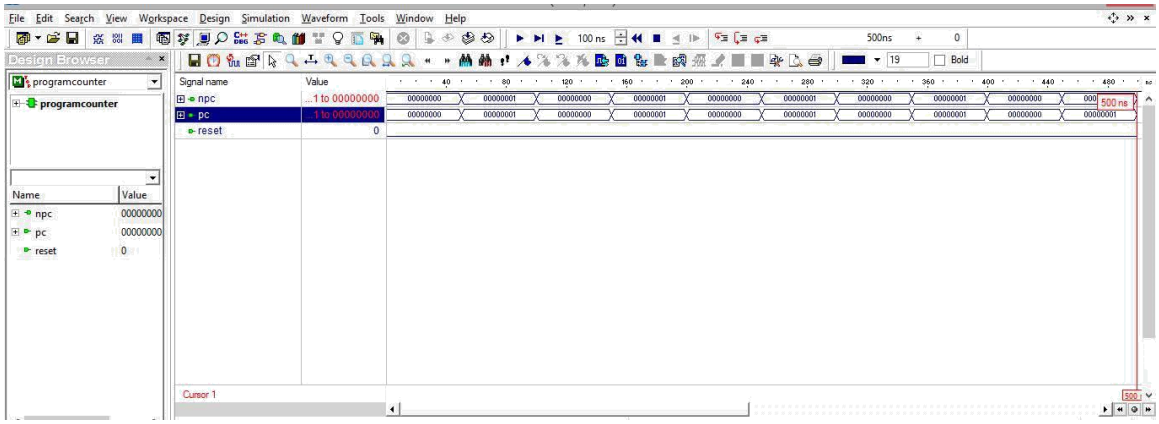
2- The data-path:



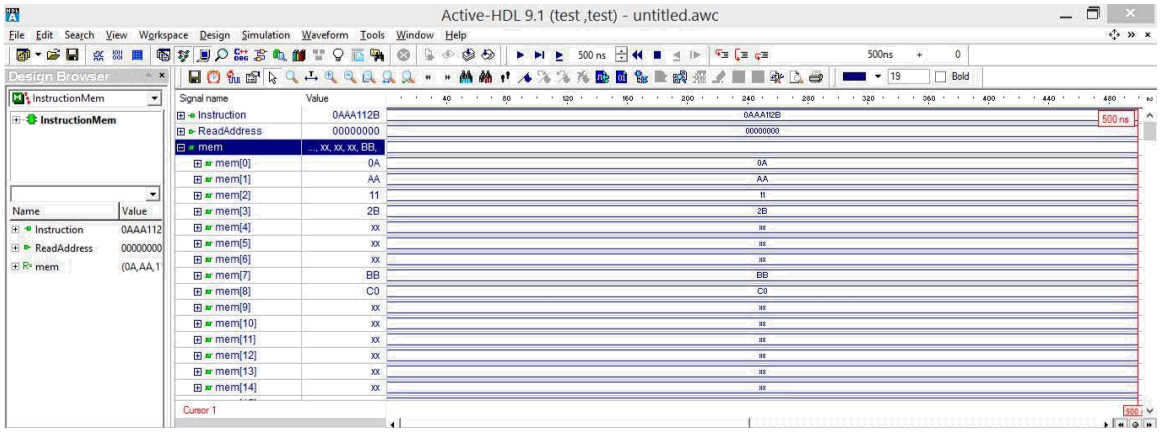
For higher resolution: http://i.imgur.com/KYTGGCu.jpg

3- User Guide:

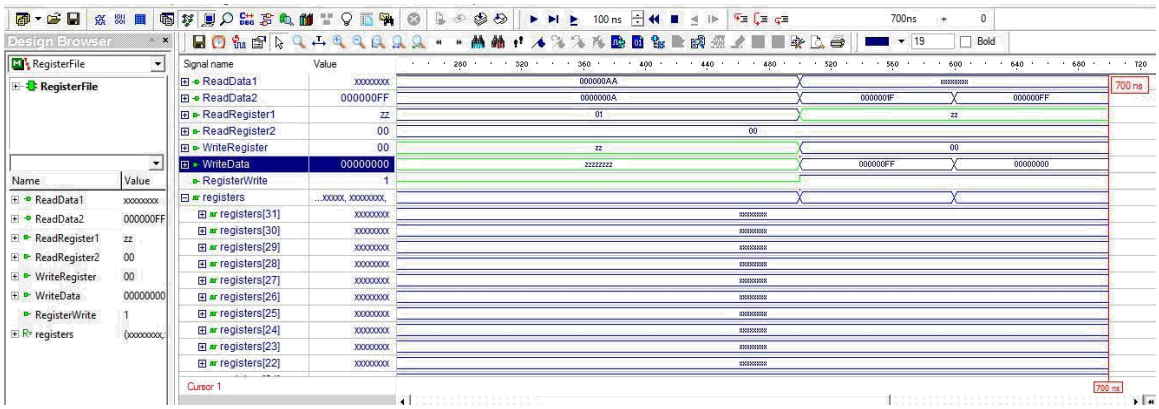
1- PC:



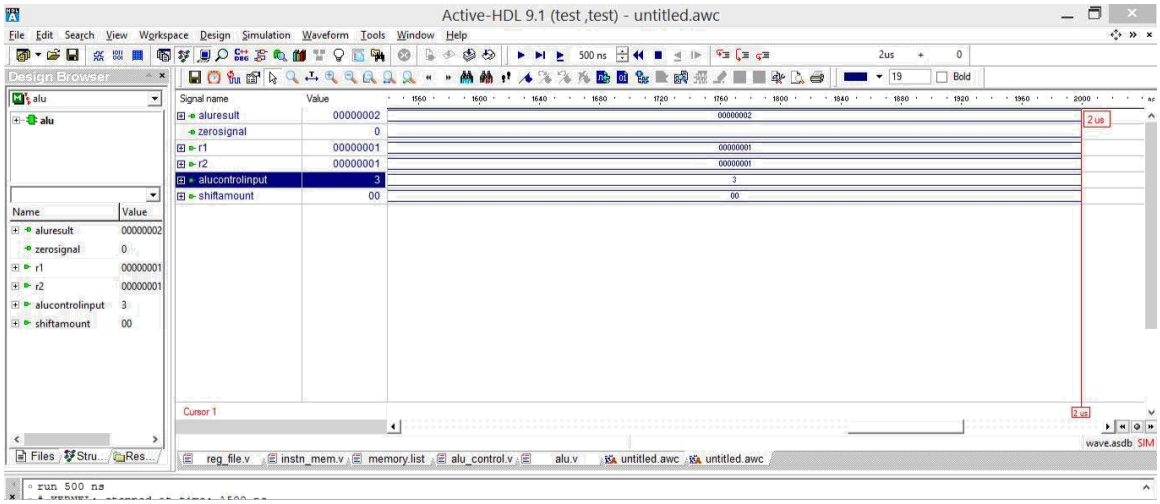
2- Instruction memory:



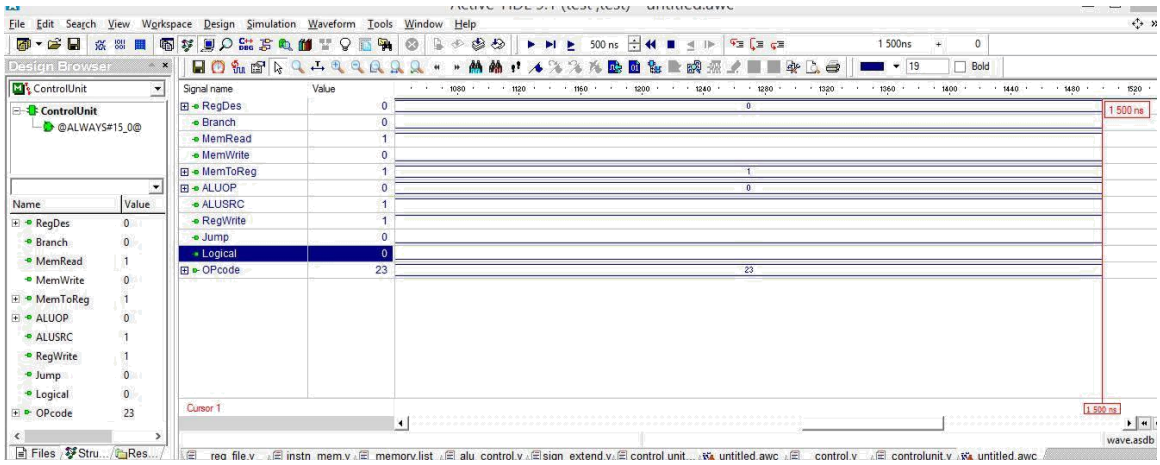
3- Register File:



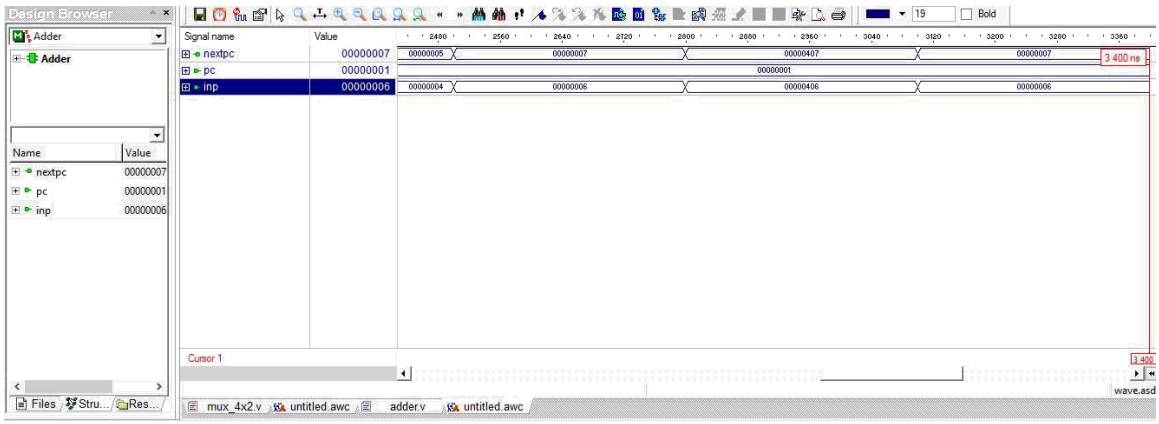
4- ALU:



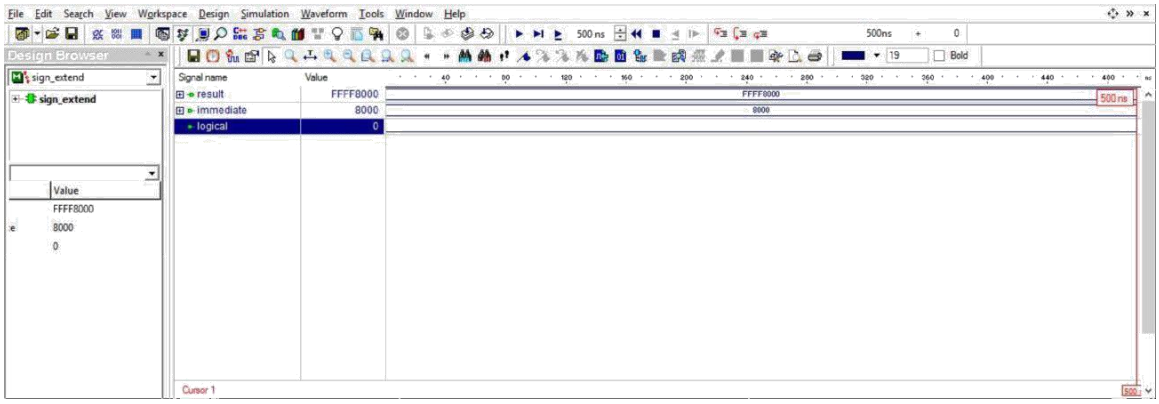
5- Control unit:

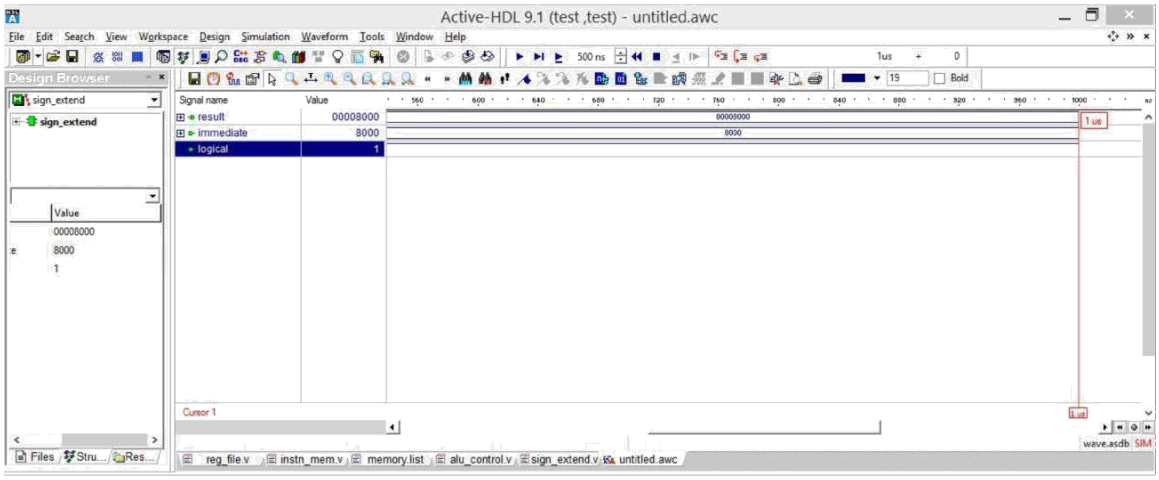


6- Adder:

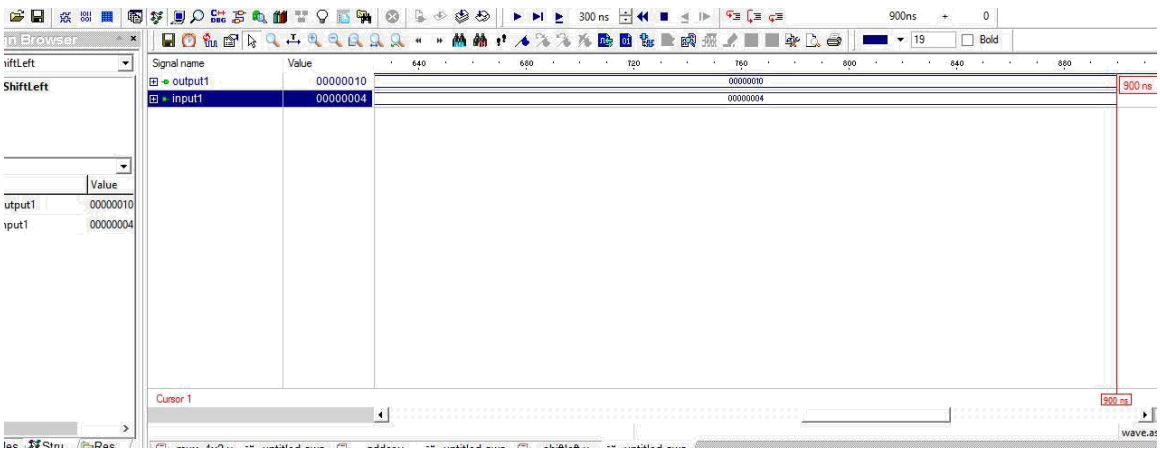


7- Sign-extend –in case logical and non-logical-:

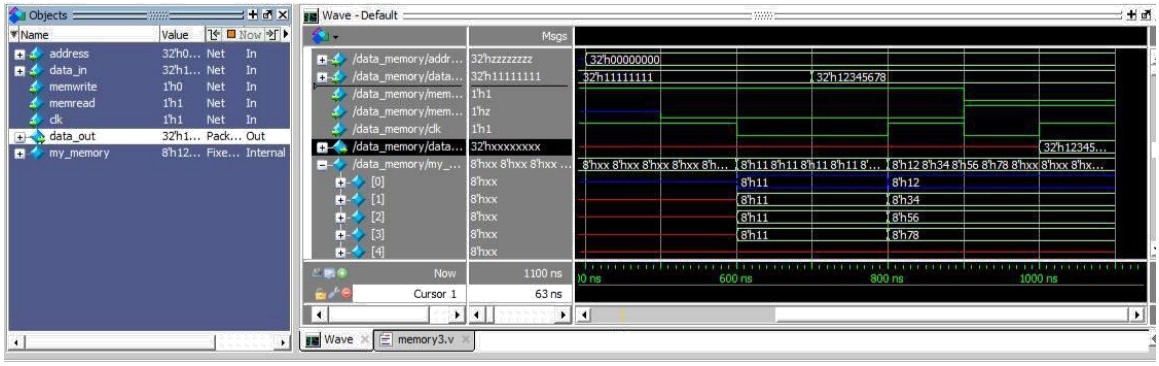




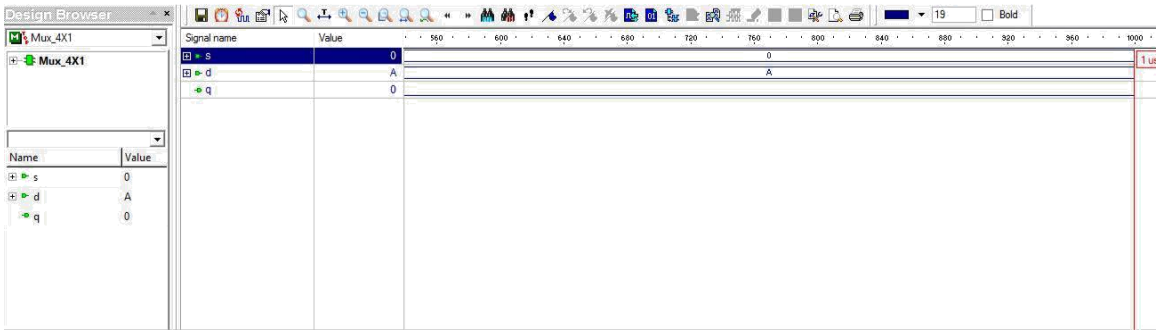
8- Shift left logical:



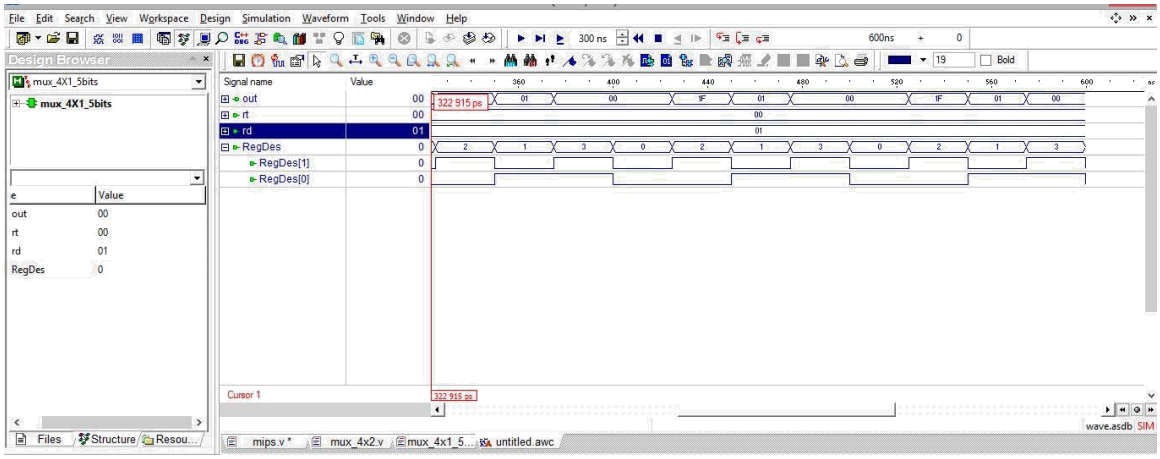
9- Data Memory:



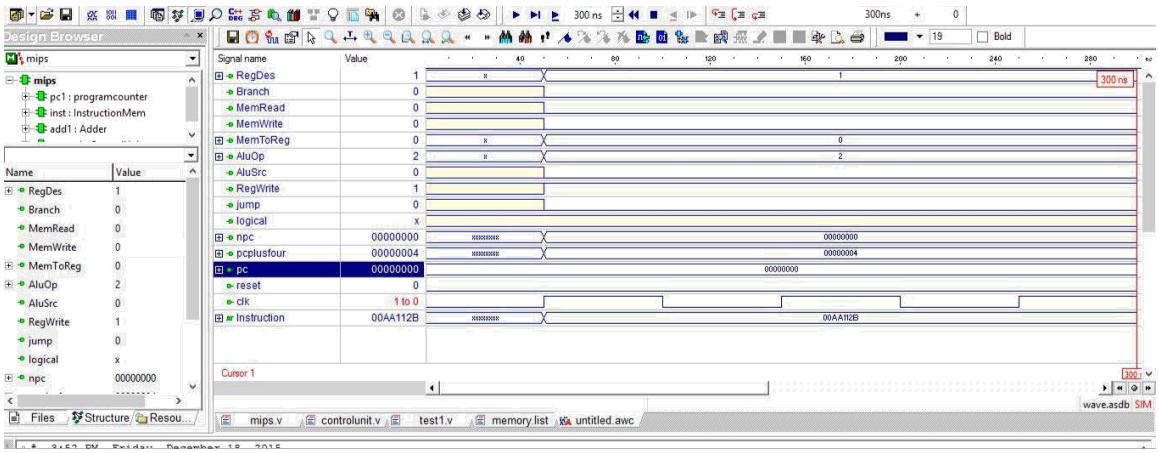
10- 4x1 Mux:



11- 4x1 5-bit mux:



12- MIPS:


Design Browser
mips
pc1 : programcounter
inst : InstructionMem
add1 : Adder
Name
Value
RegDes
Branch
MemRead
MemWrite
MemToReg
AluOp
AluSrc
RegWrite
jump
logical
npc
Files
Structure
Resou...
Signal name
Value
pcplusfour
pc
reset
clk
Instruction
00000000
00000004
00AA112B
1 to 0
300 ns
Cursor 1
mips.v
controlunit.v
test1.v
memory.list
untitled.awc
wave.asdb
SIM

4- Test Programs:

1) Assembly code: *a reasonable delay is assumed

```
addi $s1,$szero,5          #$s1=5

addi $s2,$zero,4           #$s2=4

add $s4,$zero,$zero        #$s4=0

slt $s3,$s2,$s1            #$s3=1

sw $s3,0($s4)  #$s3=1
```

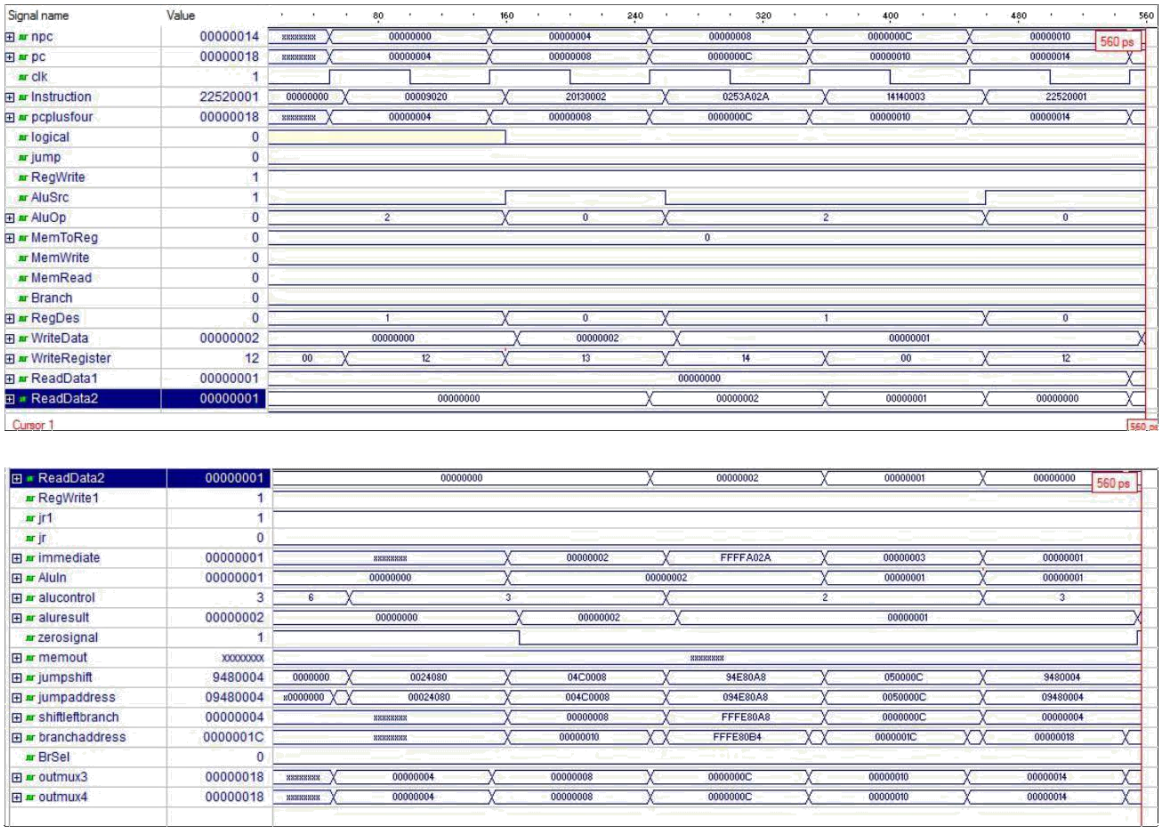
Expected value: 00000002

No. of cycles: 5

Code in binary:

```
00100010000100010000000000000101

00100010000100100000000000000100

00100010000101000000000000000000

00000010010100011001100000101010

10101110100100110000000000000000
```

Simulation of the output:



2) Assembly code: *reasonable delay is assumed

```
add $s2,$zero,$zero  #$s2=0
add $s3,$zero,2        #$s3=2
slt $s4,$s2,$s3   #check if s2<s3
beq $s4,$zero,exit
addi $s2,$s2,1        #$s2=1
exit:
```

No of cycles: 5

Expected value: 00000002

Code in binary:

00000000000000001001000000100000

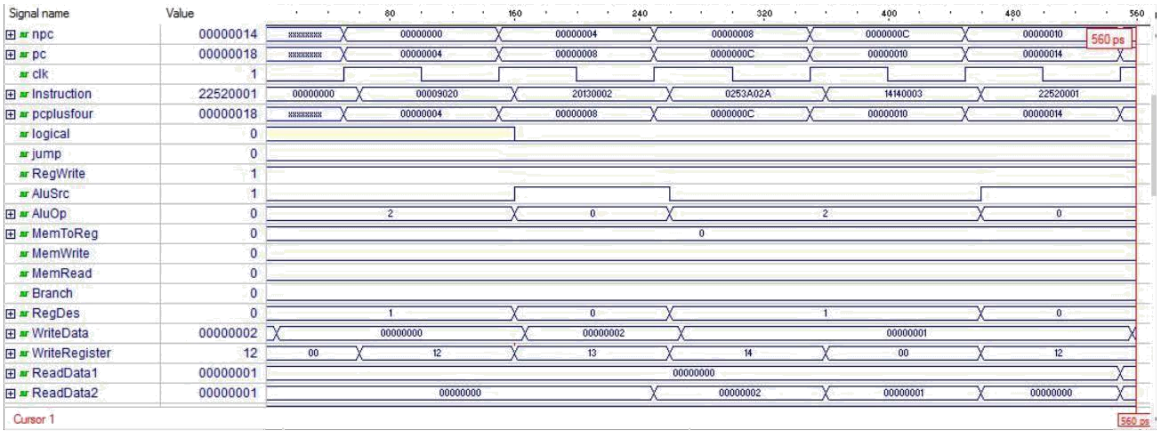
00100000000100110000000000000010

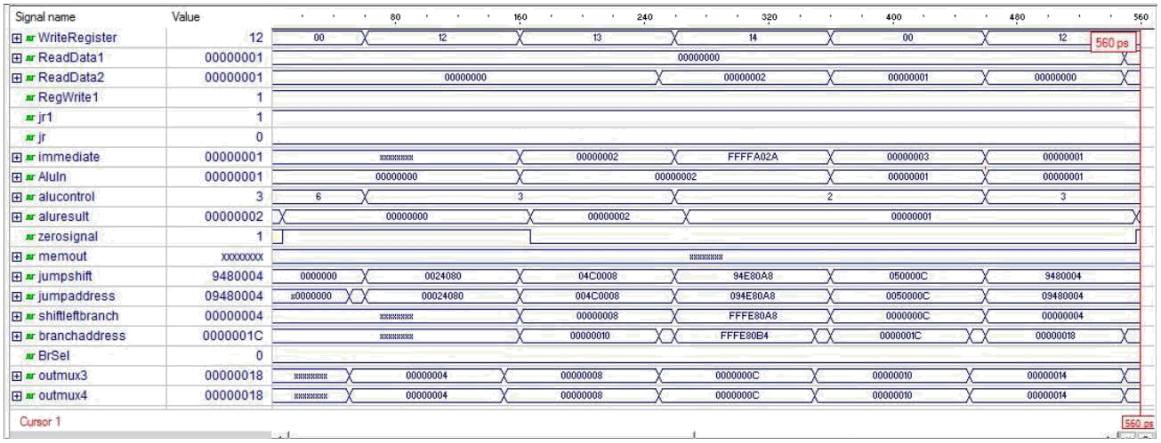
00000100101001110100000000101010

00010100000101000000000000000011

00100010010100100000000000000001

Simulation of the output:





3) **Assembly code:** *reasonable delay is assumed

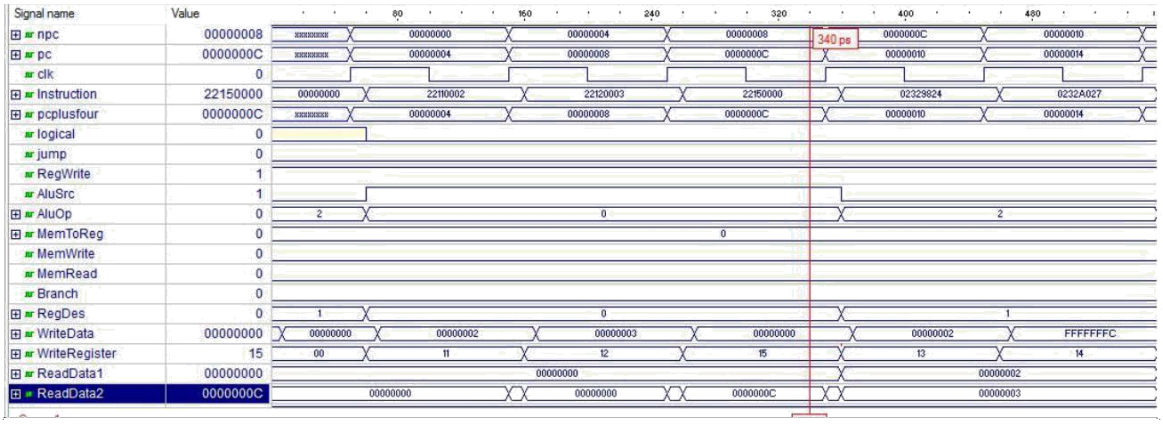
```
addi $s1,$zero,2     #$s1=2
addi $s2,$zero,3     #$s2=3
addi $s5,$zero,0     #$s5=0
and $s3,$s1,$s2      #$s3=1
nor $s4,$s1,$s2      #$s4=0
sw $s3,0($s5)
sw $s4,4($5)
```

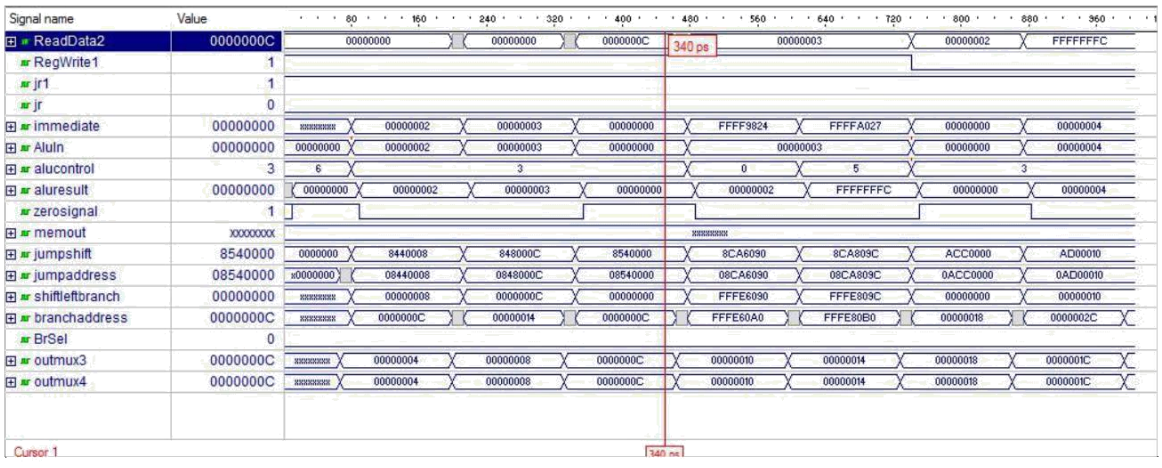
No of cycles: 7, **Expected value**: $s4=00000000

Code in Binary:

```
00100010000100010000000000000010
00100010000100100000000000000011
00100010000101010000000000000000
00000010001100101001100000100100
00000010001100101010000000100111
10101110101100110000000000000000
10101110101101000000000000000100
```

Simulation of the output:





4) Assembly code:

```
addi $s1,$zero,2     #$s1=2

addi $s2,$zero,3     #$s2=3

jr $s5

sll $s2,s2,1          #$s2=6

slt $s3,$s2,$s1        #$s3=0
```

Code in Binary:

00100000000100010000000000000010

00100000000100100000000000000011

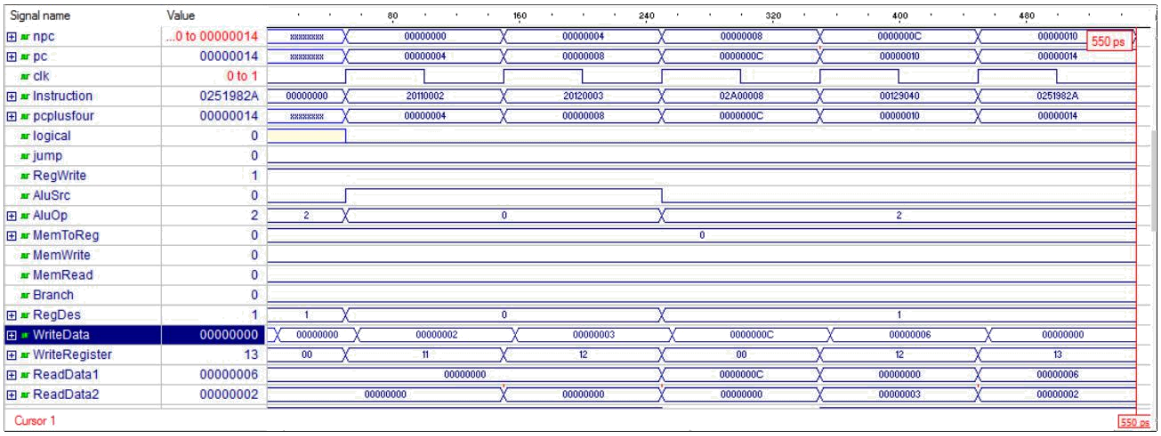
00000010101000000000000000001000

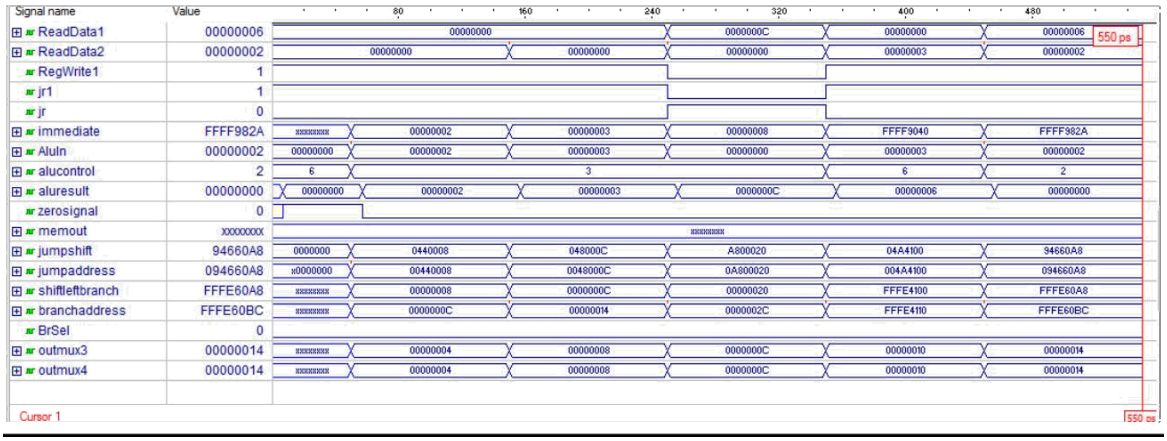
00000000000100101001000001000000

00000010010100011001100000101010

No of cycles = infinity because $s5=0

Simulation of the output:





5) Assembly code: *reasonable delay is assumed

```
addi $s0 ,$zero,10 label:
beq $s0,$zero,exit addi
$s0,$s0,-1
jal label
exit:
```

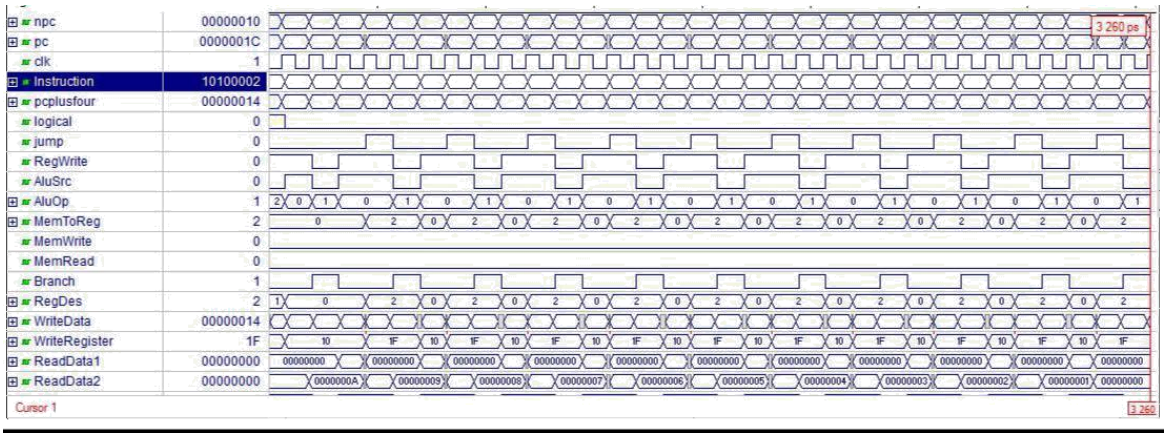
Code in binary:

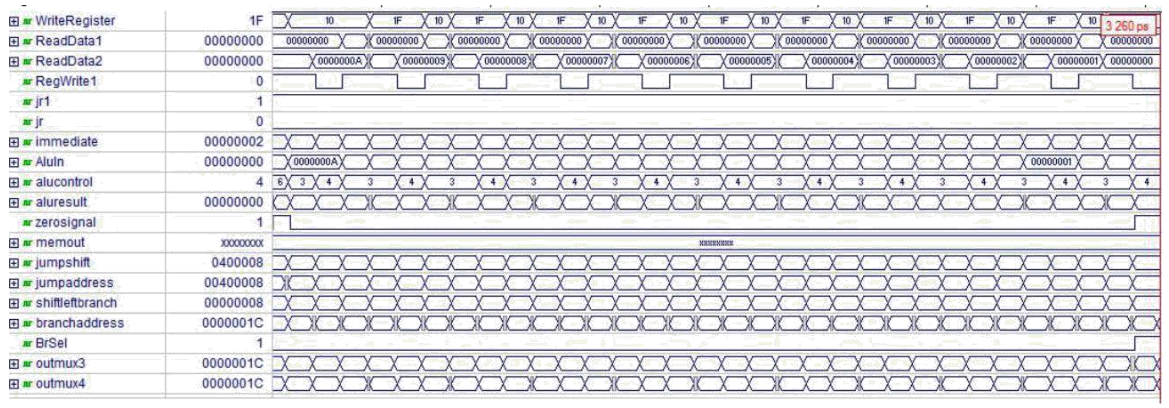
00100000000100000000000000001010
00010000000100000000000000000010
00100010000100001111111111111111
00001100000000000000000000000001

No of cycles: 32

Expected value: 00000014

Simulation of the output:





6) Assembly code: *reasonable delay is assumed

addi $s0,$0,3 #$s0= 00000011

addi $s1, $0, 7 #$s1=00000111

and $s1, $s1, $s0 #$s1= 00000011

slt $s2, $s1, $s0 #$s2=00000001

andi $s2, $s2, 2 #$s2= 00000000

No of cycles: 5

Expected value:00000000

Code in binary:

00100000000100000000000000000011

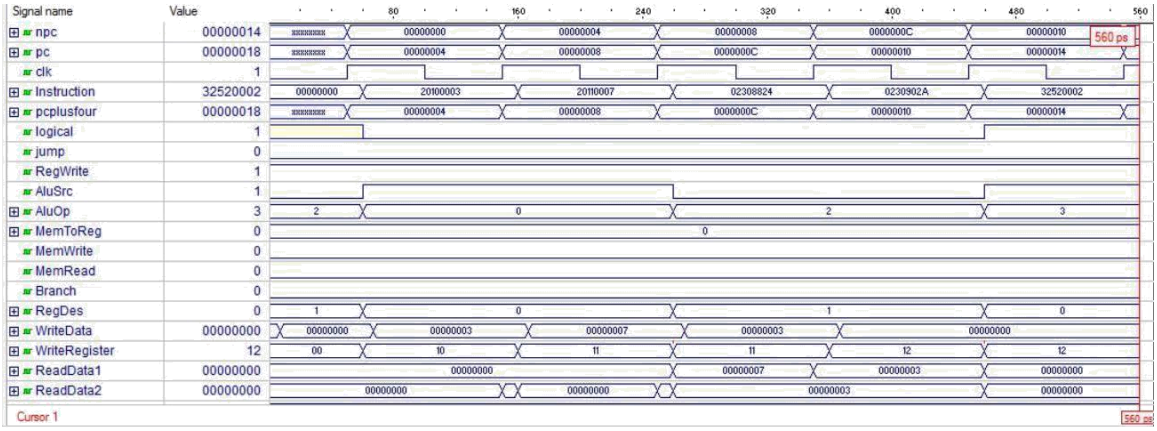
00100000000100010000000000000111

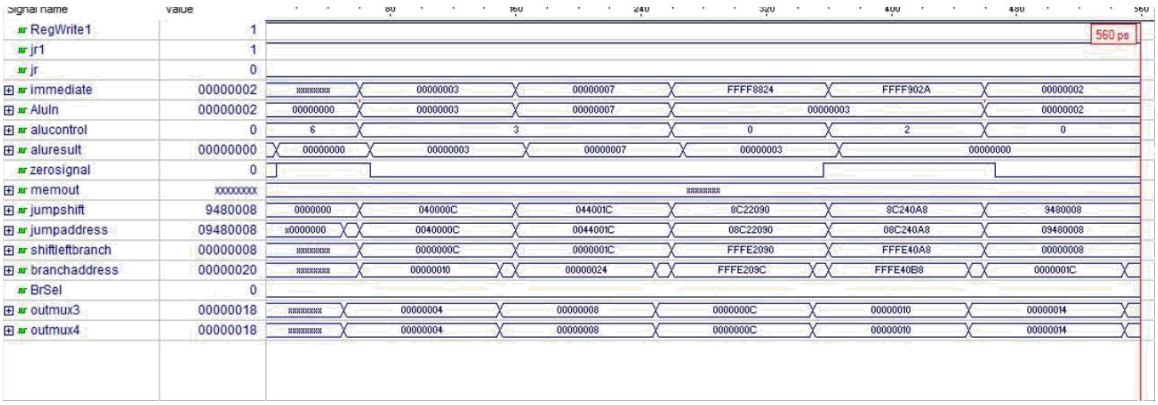
00000010001100001000100000100100

00000010001100001001000000101010

00110010010100100000000000000010

Simulation of the output:





7) Assembly code:

addi $s1,$zero,2

label:addi $s2,$zero,3

jal label

sll $s2,s2,1

slt $s3,$s2,$s1

No of cycles = infinity because $s5=0, but $ra=pc+4= c

Code in binary:

00100000000100010000000000000010

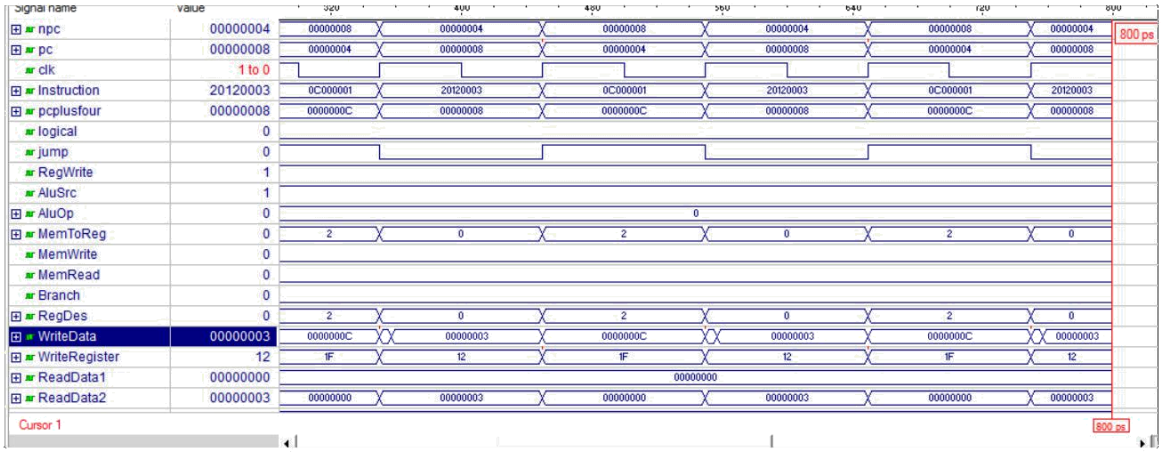
00100000000100100000000000000011

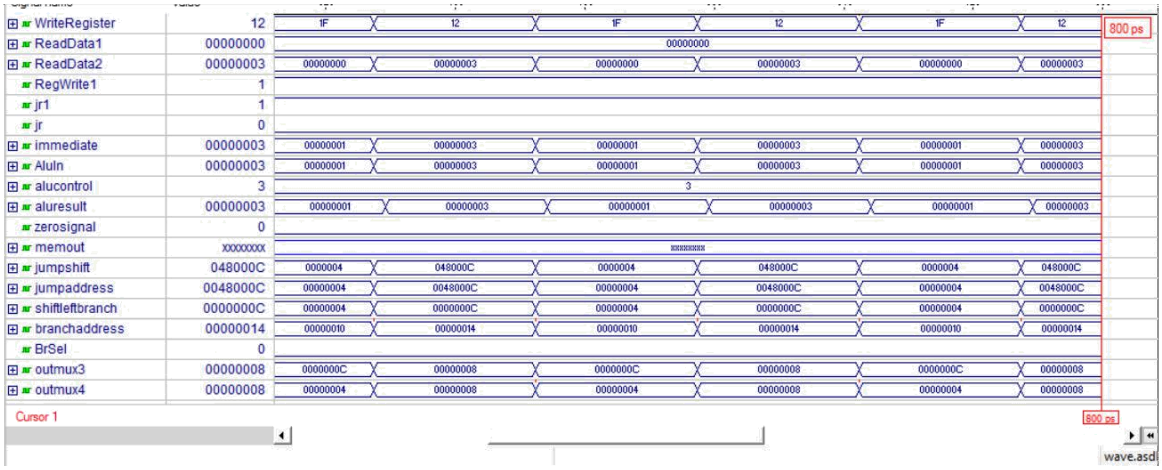
00001100000000000000000000000001

00000000000100101001000001000000

00000010010100011001100000101010

Simulation of the output:





8) Assembly code:

addi $s1 , $zero , 5
addi $s0 , $zero , 8
sw $s1 , 0($s0)
lw $s2 , 0($s0)

Code in binary:

00100000000100010000000000000101
00100000000100000000000000001000
10101110000100010000000000000000
10001110000100100000000000000000

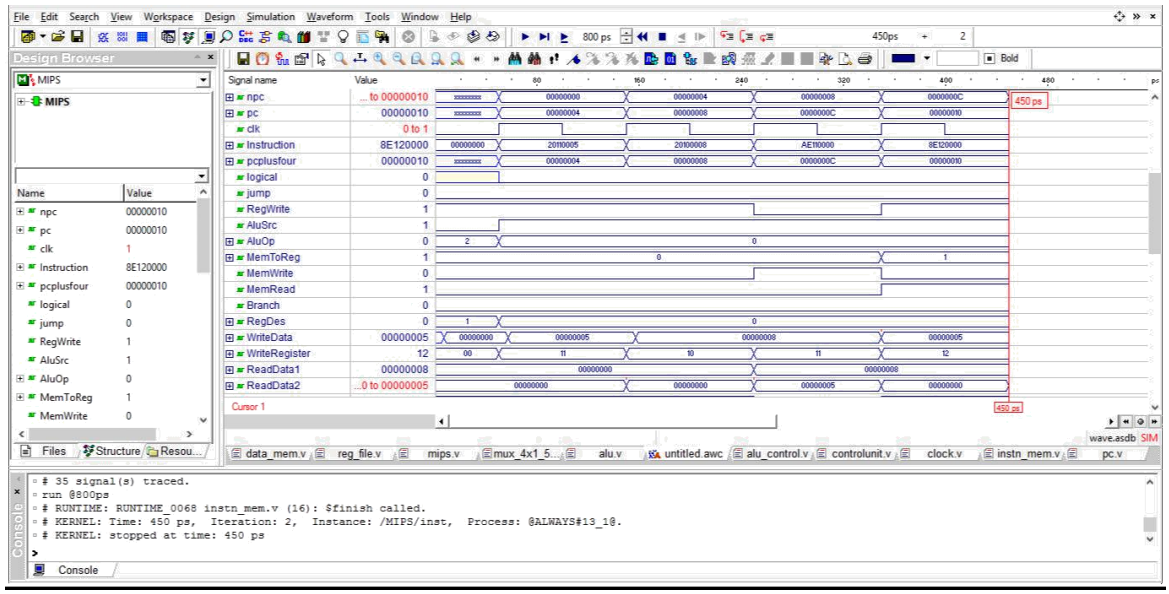
No of cycles =4

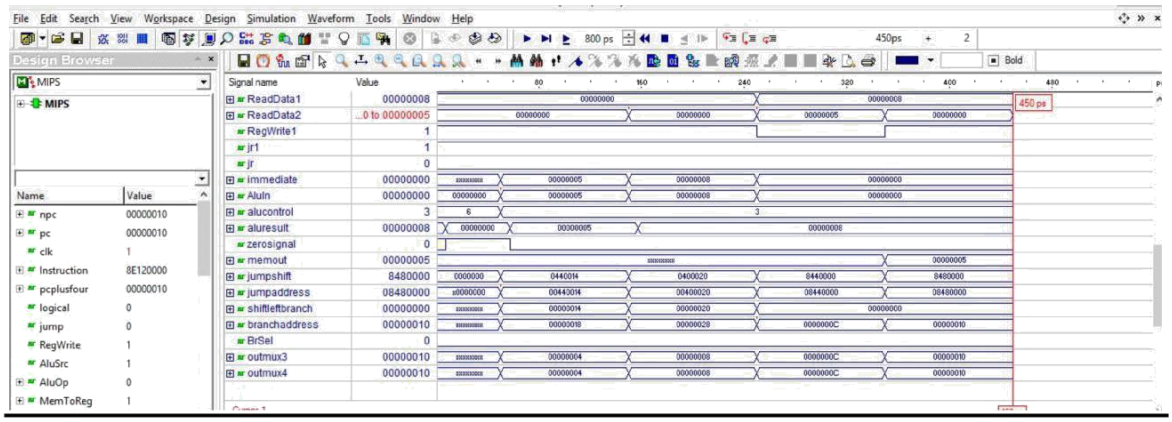
Expected values: $s1=5

$s0=8 $s2=5

memory[8]=5

Simulation of the output:





9) Assembly Code:

addi $s0, $0, 8 #$s0=00001000

addi $s1, $0, 11 #$s1= 00001011

nor $s2, $s1, $s0 #$s2=11110100

add $s2, $s2,$s1 #$s2=11111111

Code in binary:

00100000000100000000000000001000

00100000000100010000000000001011

00000010001100001001000000100111

00000010010100011001000000100000

No of cycles: 4

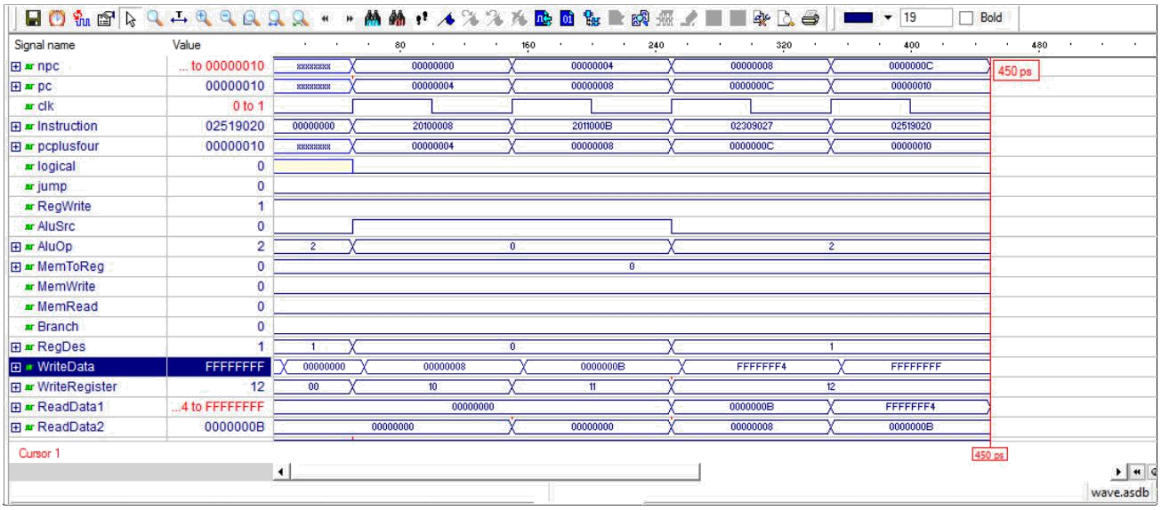
Expected values: $s0=8

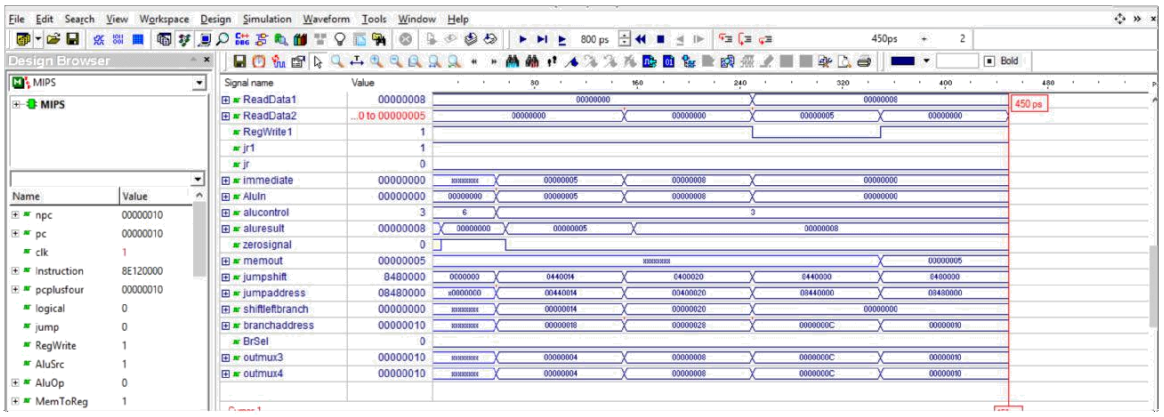
$s1=B

$s2=FFFFFFF4

$s2=FFFFFFFF

Simulation of the output:





---

10) Assembly Code: *reasonable delay is assumed

addi $s0, $0, 10 #$s0=

00001010 andi $s1, $s0, 8

#s1=00001000 sw $s2, 0($s1)

lw $s0, 0($s1)

No of cycles: 4 cycles

Expected value: 00000000

Code in binary:

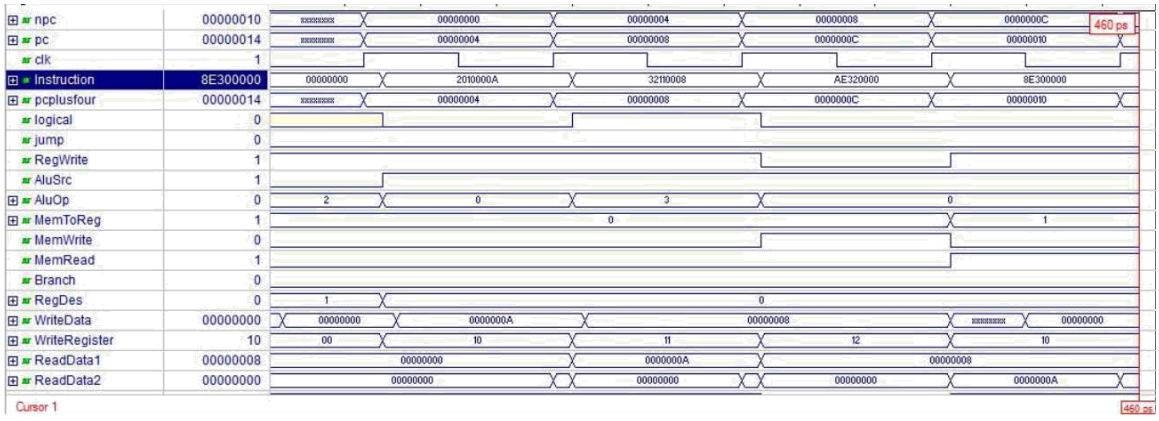
00100000000100000000000000001010

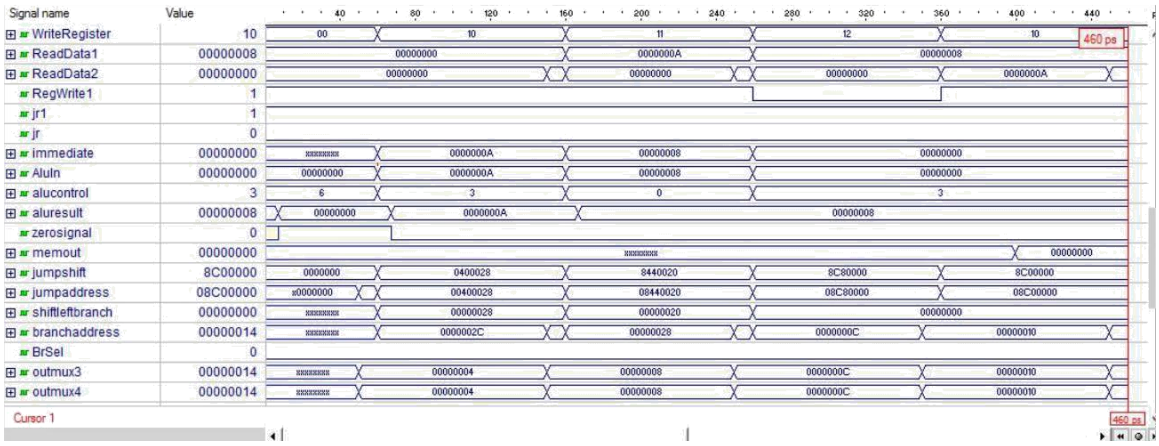
00110010000100010000000000001000

10101110001100100000000000000000

10001110001100000000000000000000

Simulation of the output:





11) Assembly code: *reasonable delay is assumed

add $s0, $zero, $zero #$s0=00000000

addi $s1, $0, 5 #$s1=00000101

slt $s2, $s0, $s1 #$s2=00000001

sll $S2, $s2, 2 #$s2=00000100

sw $s3, 0($s2)

No of cycles: 5

Expected value: 00000000

Code in binary:

00000000000000001000000000100000

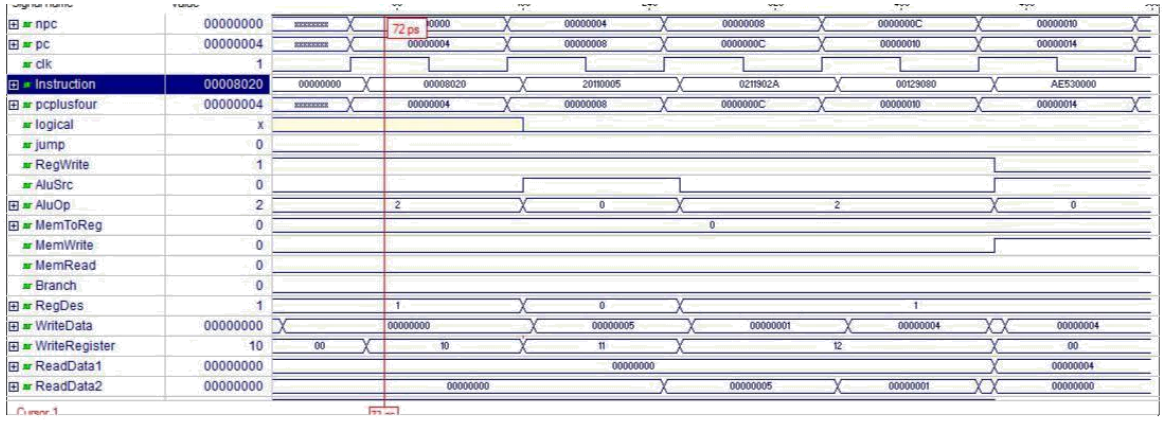
00100000000100010000000000000101

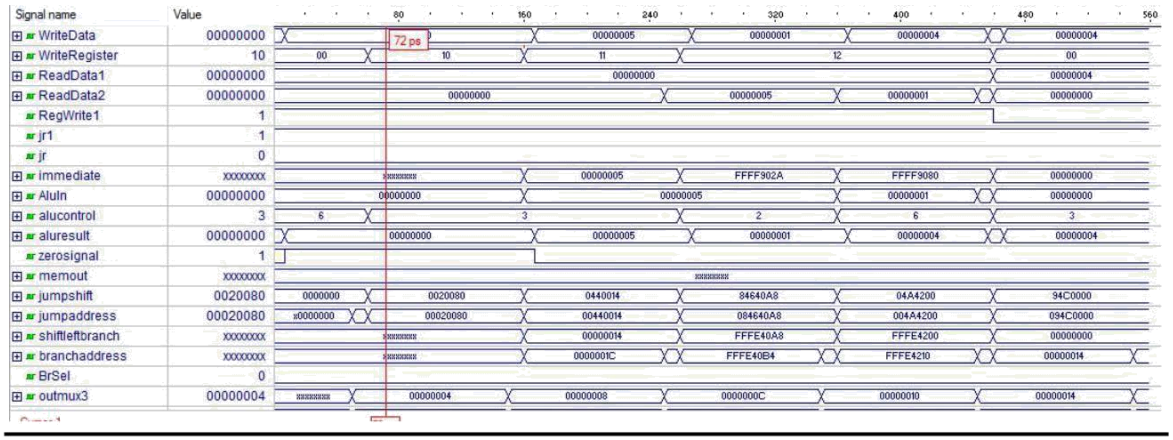
00000010000100011001000000101010

00000000000100101001000010000000

10101110010100110000000000000000

Simulation of the output:





12) Assembly code:

addi $s0, $0, 6 #$s0=00000110

andi $s1, $s0, 4 #$s1=00000100

nor $s1,$s1,$s0 #$s1=11111001

sw $s2, 0($s1)

No of cycles: 4

Expected value: FFFFFFF9

Code in binary:

00100000000100000000000000000110

00100010000100010000000000000100

00000010001100001000100000100111

10101110001100100000000000000000

Simulation of the output: