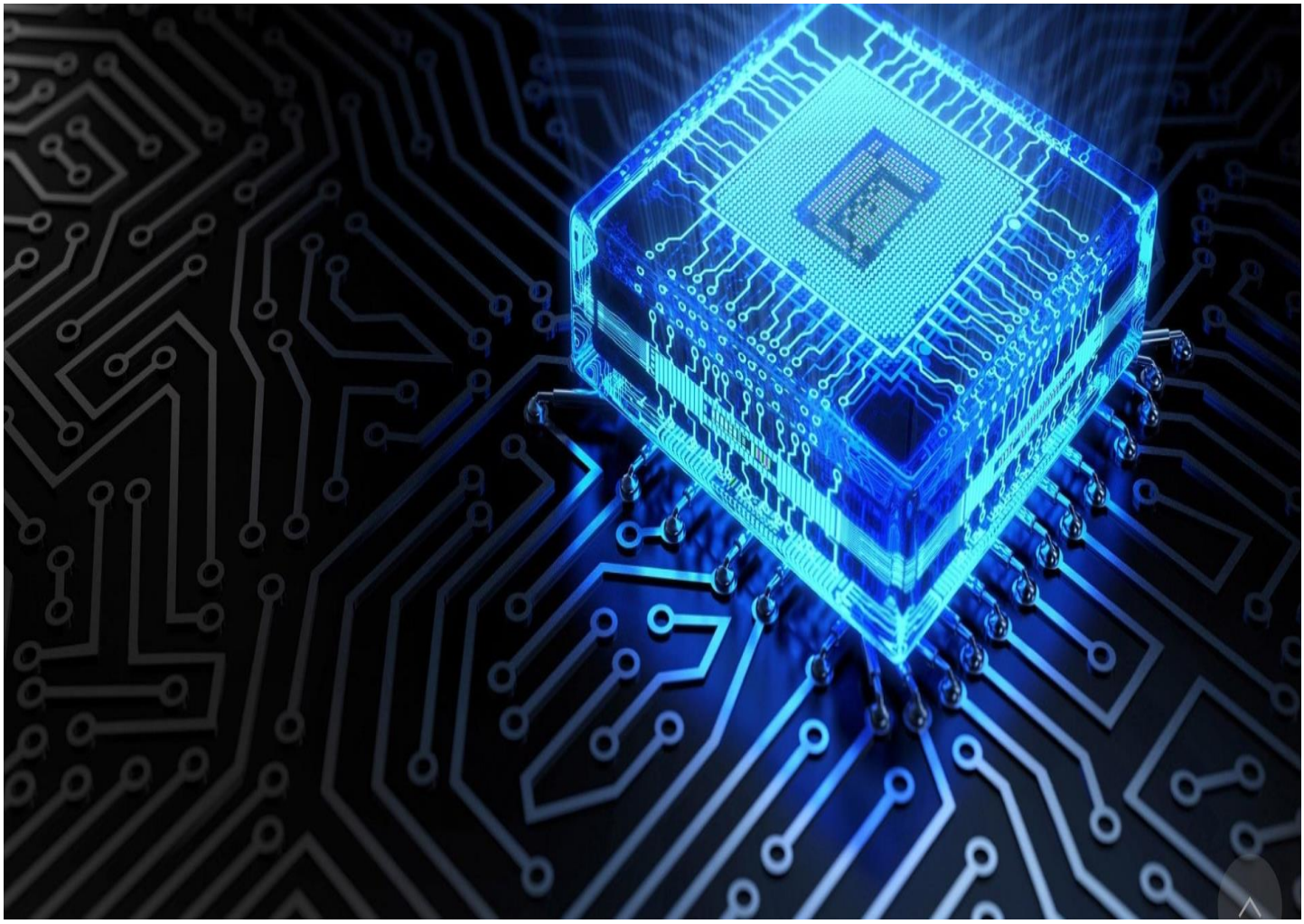
Final Project BBQM



Team Members: -

1. **Ahmed Nasser Emam Ibrahim**
2. **Ahmed Mohamed Mahmoud Taha**
3. **Adham Ibrahim**

Supervisors: -

Dr. Ahmed Shalaby

Eng. Mostafa Samy

Contents

- Inputs

Inputs	Description
Tcount	Describes the number of available tellers
Reset	Used to reset system
UpDown	When it equals “1” it pushes up the queue of system When it equals “0” it pushes down the queue of system

- Outputs

Outputs	Description
Pcount	Describes the number of people in the queue
Wtime	Describes the waiting time of people in the queue
Full	Show if the queue is full or not
Empty	Show if the queue is empty or not
Alarm	When pushing up while queue is full or When pushing down while queue is empty it becomes “1”
Error	Show the error happened during the run of the system

- Truth table

Empty Full Empty & Alarm Full & Alarm

Tcount	Pcount	UpDown	Wtime	Empty	Full	Alarm
0	0	X	0	1	0	1
1	0	0	0	1	0	1
1	0	1	0	1	0	0
1	1	X	3	0	0	0
1	2	X	6	0	0	0
1	3	X	9	0	0	0
1	4	X	12	0	0	0
1	5	X	15	0	0	0
1	6	X	18	0	0	0
1	7	0	21	0	1	0
1	7	1	21	0	1	1
2	0	0	0	1	0	1
2	0	1	0	1	0	0
2	1	X	3	0	0	0
2	2	X	4	0	0	0
2	3	X	6	0	0	0
2	4	X	7	0	0	0
2	5	X	9	0	0	0
2	6	X	10	0	0	0
2	7	0	12	0	1	0
2	7	1	12	0	1	1
3	0	0	0	1	0	1
3	0	1	0	1	0	0
3	1	X	3	0	0	0
3	2	X	4	0	0	0
3	3	X	5	0	0	0
3	4	X	6	0	0	0
3	5	X	7	0	0	0
3	6	X	8	0	0	0
3	7	0	9	0	1	0
3	7	1	9	0	1	1

- BBqM C++ Code

1) Flags Function

```
void Flags()
{
    if (Pcount == 0)
    {
        Empty = 1;
        Full = 0;
    }
    else if (Pcount == 7)
    {
        Empty = 0;
        Full = 1;
    }
    else
    {
        Empty = 0;
        Full = 0;
    }
    Alarm = 0;
    Error = "There is No Errors in the System to show";
}
```

Description: it used to set the values of Empty, Full, Alarm and Error according to the change in value of Pcount.
Empty: describe if Pcount reaches the minimum number or not
Full: describe if Pcount reaches the maximum number or not
Alarm: describe if error happened to the system or not
Error: show the error happened during execution

2) UpDown_Counter Function

```
void Up_Down_Counter(bool Reset, bool UpDown)
{
    if (Reset)
    {
        Pcount = 0;
        Tcount = 0;
        Flags();
    }
    else if (Tcount == 0)
    {
        Pcount = 0;
        Flags();
        Alarm = 1;
        if (UpDown)
        {
            Error = "Pushing Up when there is no Tellers Avaliable ";
        }
        else if (!UpDown)
        {
            Error = "Pushing Down when there is no Tellers Avaliable ";
        }
    }
    else if (UpDown)
    {

        if (Pcount != 7)
        {
            Pcount++;
            Flags();
        }
        else //(Pcount == 7)
        {
            Flags();
            Alarm = 1;
            Error = "(Overflow Error) Pushing Up when the Queue is already Full of 7";
        }
    }
    else if (!UpDown)
    {
        if (Pcount != 0)
        {
            Pcount--;
            Flags();
        }
        else  //(Pcount == 0)
        {
            Flags();
            Alarm = 1;
            Error = "(Underflow Error) Pushing Down when the Queue is already Empty";
        }
    }
```

Description: it takes Reset and UpDown as arguments and used them to reset the system, push up or push down the system, here we call Flags function due to there is the change in Pcount Happened and describe the error happened.

3) Setting_Tcount Function

```
void setting_Tcount()
{
    cout << "Setting Tcount " << endl
        << "Enter the Number of Available Tellers between 0 - 3 : ";
    cin >> Tcount;
    while (Tcount < 0 || Tcount > 3)
    {
        cout << "Wrong Number of Available Tellers " << endl
            << "Enter a number between  0 - 3 : ";
        cin >> Tcount;
    }
}
```

Description: it used to set the value of Tcount and avoid choosing value < 0 Tellers and > 3 Tellers

4) Setting_Pcount Function

```
void setting_Pcount()
{
    if (Tcount == 0)
    {
        Pcount = 0;
        Flags();
        Alarm = 1;
        Error = "Setting Pcount when there is no Tellers Avaliable ";
    }
    else
    {
        cout << "Setting a manual value for Pcount " << endl
            << "Enter the Number of People in the Queue between 0 - 7 : ";
        cin >> Pcount;
        while (Pcount < 0 || Pcount > 7)
        {
            cout << "Wrong Number of People in the Queue" << endl
                << "Enter a number between  0 - 7 : ";
            cin >> Pcount;
        }
        Flags();
    }
}
```

Description: it used to set the value of Pcount and avoid choosing value < 0 Persons and > 7 Persons
Due to changing value of Pcount we call Flag Function
When tellers are not available so no service then Pcount becomes 0.

5) Wtime_Calculations Function

```
int Wtime_Calculations(int P, int T)
{
    int Wtime = 0;
    int total = 0;
    int Multicounts = 3;
    if (P == 0)
    {
        Wtime = 0;
    }
    else
    {
        while (Multicounts != 0)
        {
            total = total + (P + T - 1);
            Multicounts--;
        }
        while (total > 0)
        {
            total = total - T;
            Wtime++;
        }
        if (total < 0)
        {
            Wtime--;
        }
    }
    return Wtime;
}
```

Description: it used to calculate the value of Wtime according to the arguments (Pcount, Tcount) and to the law

Wtime (Pcount = 0) = 0,

Wtime (Pcount ≠ 0, Tcount) = 3(Pcount+Tcount-1)/Tcount*

But this Function later we will assemble (Changing to MIPS Assembly), and we don't have the needed instructions (mult, div) on the single cycle MIPS processor that we have. So, we solve the problem by using the concepts of Multiplications is the process of repeating addition and Division is the process of repeating subtraction.

6) Display Function

```
void display()
{
    cout << "Tcount (Number of Avalible Tellers) = " << Tcount << endl
        << "Pcount (Number of People in the Queue) = " << Pcount << endl
        << "Wtime (Waiting Time) = " << Wtime_Calculations(Pcount, Tcount) << " Sec. " << endl
        << "Empty Flag = " << Empty << endl
        << "Full Flag = " << Full << endl
        << "Alarm Flag = " << Alarm << endl;
    if (Empty)
    {
        cout << "The Queue is Empty " << endl;
    }
    else if (Full)
    {
        cout << "The Queue is Full " << endl;
    }
    cout << "Error Happened during Operation : " << Error << endl;
}
```

Description: it used to show all data we have in the system values of Tcount, Pcount, Wtime, Empty, Full, Alarm, Error.

7) Menu Function

```
void Menu()
{
    int Operation;
    cout << "----------------------------------------------------------------" << endl
        << "Choose Operation number That you want to be executed " << endl
        << "1 - Reset the System " << endl
        << "2 - Push Up the Queue " << endl
        << "3 - Push Down the Queue " << endl
        << "4 - Set a Manual value for Pcount " << endl
        << "5 - Set Tcount " << endl
        << "6 - Get the Data Saved in System " << endl
        << "Operation no. : ";
    cin >> Operation;
    do
    {
        switch (Operation)
        {
        case 1:
            cout << "Resetting the System " << endl;
            Up_Down_Counter(true, true);// or (tru , X )
            Operation = -1;
            break;
        case 2:
            cout << "Pushing Up the Queue " << endl;
            Up_Down_Counter(false, true);
            Operation = -1;
            break;
        case 3:
            cout << "Pushing Down the Queue " << endl;
            Up_Down_Counter(false, false);
            Operation = -1;
            break;
        case 4:
            setting_Pcount();
            Operation = -1;
            break;
        case 5:
            setting_Tcount();
            Operation = -1;
            break;
        case 6:
            cout << "Getting the Data Saved in System " << endl;
            display();
            Operation = -1;
            break;
        default:
            do
            {
                cout << "Wrong Operation number Please Enter a Correct Operation number" << endl
                    << "Operation no. : ";
                cin >> Operation;
            } while (Operation < 1 || Operation > 6);
            break;
        }
    } while (Operation >= 1 && Operation <= 6);
}
```

Description: it used to make a menu of operation the user can choose from it to handle the system; it asks the user to choose operation number then execute it.
Operations are

1. Reset the System
2. Push Up the Queue
3. Push Down the Queue
4. Set a Manual value for Pcount
5. Set Tcount
6. Get the Data Saved in System

After choosing the operation number it executes it using the calling of all previous Functions.

8) Main Function

```
int main()
{
    char Again = 'x';
    setting_Tcount();
    cout << "Number of People in the Queue = " << Pcount << endl;
    do
    {
        Menu();
        cout << "-----------------------------------------------------------------" << endl
            << "Current Data after Execution of Operation " << endl;
        display();
        cout << "-----------------------------------------------------------------" << endl
            << "If you want to make another Operation Press \'Y\' or \'y\'" << endl
            << "OR " << endl
            << "Press another key to Exit " << endl;
        cin >> Again;
    } while (Again == 'y' || Again == 'Y');
    cout << "End of Operations " << endl;
    system("pause");
    return 0;
}
```

Description: it's the function here all functions are called there by calling Menu function and choosing the operation Then ask the user if he want another operation or not.

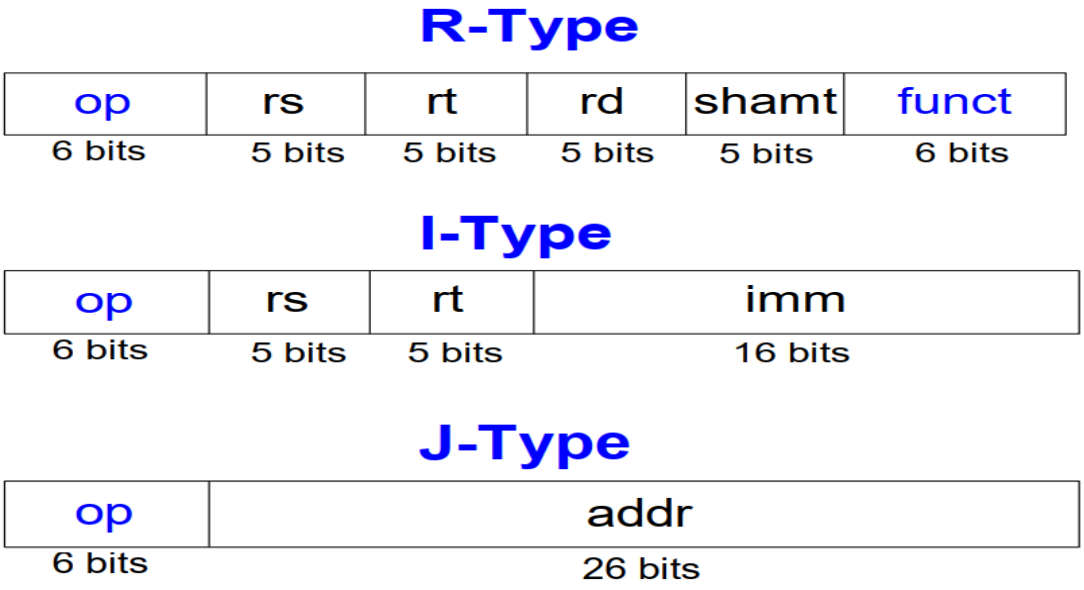
- Wtime Calculations Assembly code

```
#P = $a0 , T = $a1
# Wtime = $t0
# total = $s0 , Multicounts = $s1 , P+T-1=$t3 ,0<total =$t4
addi $a0,$0,7 #P = 7
addi $a1,$0,1  #T = 1
addi $s0,$0,0  #total = 0
addi $s1,$0,3 #Multicounts = 3
add $t3,$a0,$a1  #P+T
addi $t3,$t3,-1  #P+T-1
addi $t0,$0,0 #Wtime = 0
beq $a0,$0,done #if(P==0)
j else
else :
beq $s1,$0,while #while(Multicounts != 0)
add $s0,$s0,$t3  #total = total + (P + T - 1)
addi $s1,$s1,-1  #Multicounts--
j else
while :
slt $t4,$0,$s0  #total>0
beq $t4,$0,if  #if (total >0)
sub $s0,$s0,$a1  #total=total-T
addi $t0,$t0,1   #Wtime++
j while
if :
slt $t5,$s0,$0 #total < 0
beq $t5,$0,done
addi $t0,$t0,-1   #Wtime--
done :
sw $t0,8($0)
```

- Wtime Calculations Machine code
  - MIPS Register Set

Name	Register Number	Usage
`$0`	0	the constant value 0
`$at`	1	assembler temporary
`$v0-$v1`	2-3	Function return values
`$a0-$a3`	4-7	Function arguments
`$t0-$t7`	8-15	temporaries
`$s0-$s7`	16-23	saved variables
`$t8-$t9`	24-25	more temporaries
`$k0-$k1`	26-27	OS temporaries
`$gp`	28	global pointer
`$sp`	29	stack pointer
`$fp`	30	frame pointer
`$ra`	31	Function return address

  - Instruction Formats



○ MIPS Instructions

R-Type Instructions

Funct	Name	Description	Operation
100000 (32)	add rd, rs, rt	add	[rd] = [rs] + [rt]
100010 (34)	sub rd, rs, rt	subtract	[rd] = [rs] – [rt]
101010 (42)	slt rd, rs, rt	set less than	[rs] < [rt] ? [rd] = 1 : [rd] = 0

I-Type Instructions

Opcode	Name	Description	Operation
000100 (4)	beq rs, rt, label	branch if equal	if ([rs] == [rt]) PC = BTA
001000 (8)	addi rt, rs, imm	add immediate	[rt] = [rs] + SignImm
101011 (43)	sw rt, imm(rs)	store word	[Address] = [rt]

J-Type Instructions

Opcode	Name	Description	Operation
000010 (2)	j label	jump	PC = JTA

○ Converting Assembly to Machine code

1. addi $a0,$0,7 -> I-Type

8	0	4	7
6 bits	5 bits	5 bits	16 bits
0010 00	00 000	0 0100	0000 0000 0000 0111

Machine code in hexadecimal format-> 0x20040007

2. addi $a1,$0,1 -> I-Type

8	0	5	1
6 bits	5 bits	5 bits	16 bits
0010 00	00 000	0 0101	0000 0000 0000 0001

Machine code in hexadecimal format-> 0x20050001

3. addi $s0,$0,0 ->I-Type

8	0	16	0
6 bits	5 bits	5 bits	16 bits
0010 00	00 000	1 0000	0000 0000 0000 0000

Machine code in hexadecimal format-> 0x20100000

4. addi $s1,$0,3 -> I-Type

8	0	17	3
6 bits	5 bits	5 bits	16 bits
0010 00	00 000	1 0001	0000 0000 0000 0011

Machine code in hexadecimal format-> 0x20110003

5. add $t3,$a0,$a1 -> R-Type

0	4	5	11	0	32
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
0000 00	00 100	0 0101	0101 1	000 00	10 0000

Machine code in hexadecimal format-> 0x00855820

6. addi $t3,$t3,-1 -> I-Type

8	11	11	-1
6 bits	5 bits	5 bits	16 bits
0010 00	01 011	0 1011	1111 1111 1111 1111

Machine code in hexadecimal format-> 0x216BFFFF

7. addi $t0,$0,0 -> I-Type

8	0	8	0
6 bits	5 bits	5 bits	16 bits
0010 00	00 000	0 1000	0000 0000 0000 0000

Machine code in hexadecimal format-> 0x20080000

8. beq $a0,$0,done -> I-Type

4	4	0	13
6 bits	5 bits	5 bits	16 bits
0001 00	00 100	0 0000	0000 0000 0000 1101

Machine code in hexadecimal format-> 0x1080000D

9. j else -> J-Type

2	Address = 0x00400024
6 bits	26 bits

0000 10	00 0001 0000 0000 0000 0000 1001

Machine code in hexadecimal format-> 0x08100009

10. else: beq $s1,$0,while -> I-Type

4	17	0	3
6 bits	5 bits	5 bits	16 bits

0001 00	10 001	0 0000	0000 0000 0000 0011

Machine code in hexadecimal format-> 0x12200003

11. add $s0,$s0,$t3 -> R-Type

0	16	11	16	0	32
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

0000 00	10 000	0 1011	1000 0	000 00	10 0000

Machine code in hexadecimal format-> 0x020B8020

12. addi $s1,$s1,-1 -> I-Type

8	17	17	-1
6 bits	5 bits	5 bits	16 bits

0010 00	10 001	1 0001	1111 1111 1111 1111

Machine code in hexadecimal format-> 0x2231FFFF

13. j else -> J-Type

2	Address = 0x00400024
6 bits	26 bits

0000 10	00 0001 0000 0000 0000 0000 1001

Machine code in hexadecimal format-> 0x08100009

14. while: slt $t4,$0,$s0 -> R-Type

0	0	16	12	0	42
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
0000 00	00 000	1 0000	0110 0	000 00	10 1010

Machine code in hexadecimal format-> 0x0010602A

15. beq $t4,$0,if -> I-Type

4	12	0	3
6 bits	5 bits	5 bits	16 bits
0001 00	01 100	0 0000	0000 0000 0000 0011

Machine code in hexadecimal format-> 0x11800003

16. sub $s0,$s0,$a1 -> R-Type

0	16	5	16	0	34
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
0000 00	10 000	0 0101	1000 0	000 00	10 0010

Machine code in hexadecimal format-> 0x02058022

17. addi $t0,$t0,1 -> I-Type

8	8	8	1
6 bits	5 bits	5 bits	16 bits
0010 00	01 000	0 1000	0000 0000 0000 0001

Machine code in hexadecimal format-> 0x21080001

18. j while -> J-Type

2	Address = 0x00400034
6 bits	26 bits
0000 10	00 0001 0000 0000 0000 0000 1101

Machine code in hexadecimal format-> 0x0810000D

19. if: slt $t5,$s0,$0 -> R-Type

0	16	0	13	0	42
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
0000 00	10 000	0 0000	0110 1	000 00	10 1010

Machine code in hexadecimal format-> 0x0200682A

20. beq $t5,$0,done -> I-Type

4	13	0	1
6 bits	5 bits	5 bits	16 bits
0001 00	01 101	0 0000	0000 0000 0000 0001

Machine code in hexadecimal format-> 0x11A00001

21. addi $t0,$t0,-1 -> I-Type

8	8	8	1
6 bits	5 bits	5 bits	16 bits
0010 00	01 000	0 1000	1111 1111 1111 1111

Machine code in hexadecimal format-> 0x2108FFFF

22. done: sw $t0,8($0)

43	8	8	8
6 bits	5 bits	5 bits	16 bits
1010 11	00 000	0 1000	0000 0000 0000 1000

Machine code in hexadecimal format-> 0xAC080008

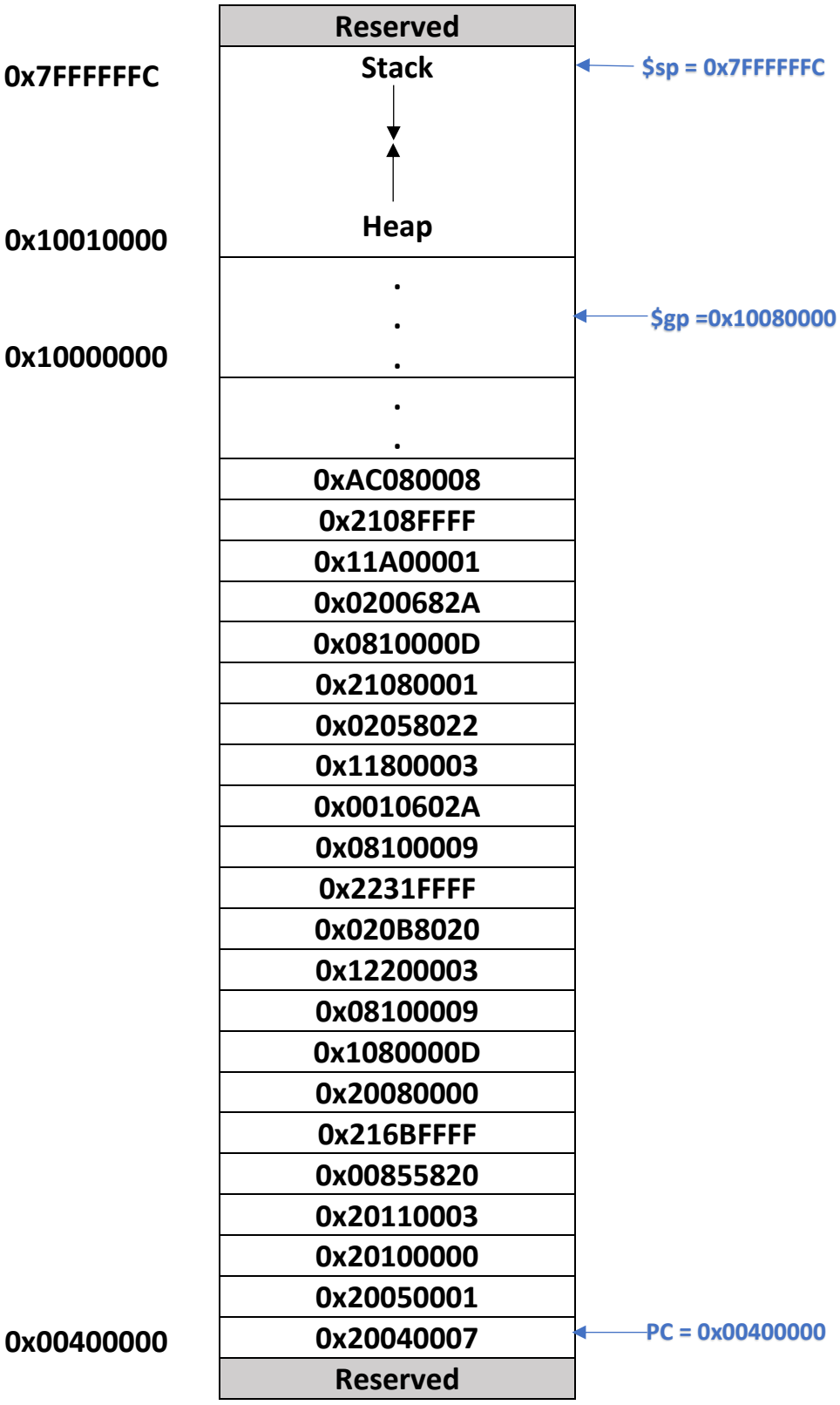
- Machine code && Assembly

Address	Machine	Assembly	Description
0x00400000	0x20040007	addi $a0,$0,7	#P = 7
0x00400004	0x20050001	addi $a1,$0,1	#T = 1
0x00400008	0x20100000	addi $s0,$0,0	#total = 0
0x0040000c	0x20110003	addi $s1,$0,3	#Multicounts = 3
0x00400010	0x00855820	add $t3,$a0,$a1	#P+T
0x00400014	0x216BFFFF	addi $t3,$t3,-1	#P+T-1
0x00400018	0x20080000	addi $t0,$0,0	#Wtime = 0
0x0040001c	0x1080000D	beq $a0,$0,done	#if(P==0)
0x00400020	0x08100009	j else	
0x00400024	0x12200003	else: beq $s1,$0,while	#while(Multicounts != 0)
0x00400028	0x020B8020	add $s0,$s0,$t3	#total = total + (P + T - 1)
0x0040002c	0x2231FFFF	addi $s1,$s1,-1	#Multicounts—
0x00400030	0x08100009	j else	
0x00400034	0x0010602A	while: slt $t4,$0,$s0	#total>0
0x00400038	0x11800003	beq $t4,$0,if	#if (total >0)
0x0040003c	0x02058022	sub $s0,$s0,$a1	#total=total-T
0x00400040	0x21080001	addi $t0,$t0,1	#Wtime++
0x00400044	0x0810000D	j while	
0x00400048	0x0200682A	if: slt $t5,$s0,$0	#total < 0
0x0040004c	0x11A00001	beq $t5,$0,done	#if (total <0)
0x00400050	0x2108FFFF	addi $t0,$t0,-1	#Wtime—
0x00400054	0xAC080008	done: sw $t0,8($0)	#return Wtime (By writing it in Memory)

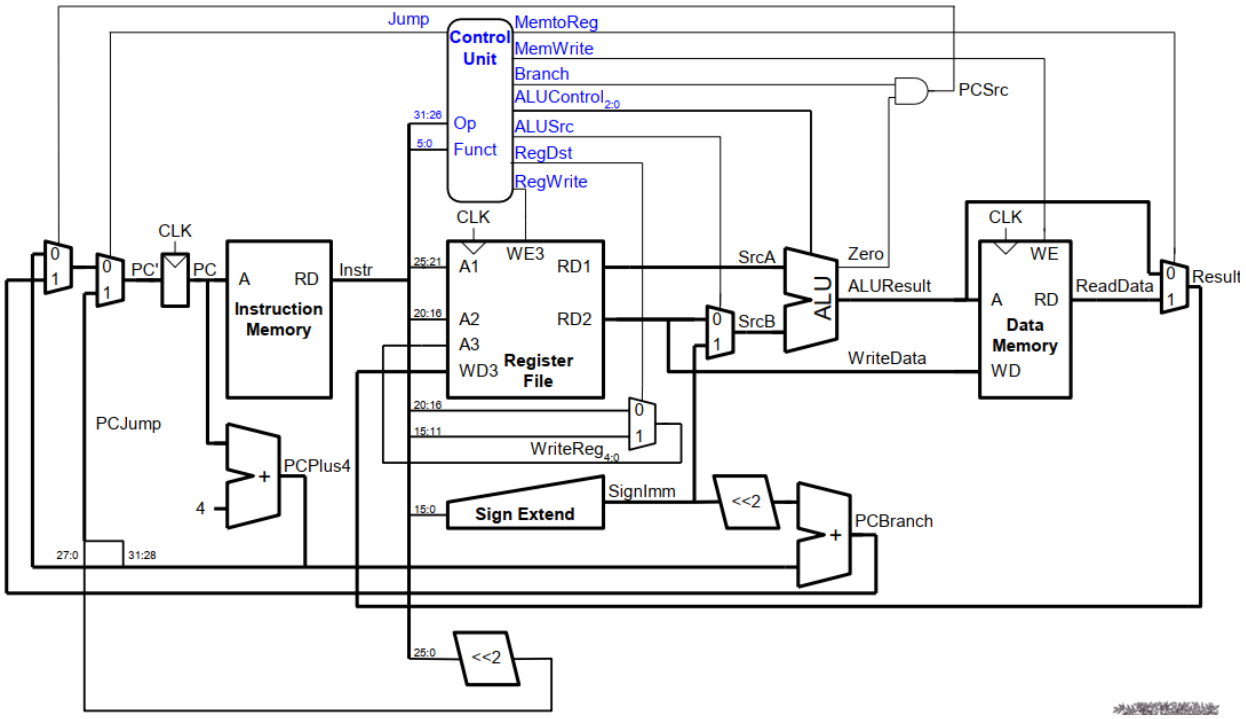
- Executable File Header

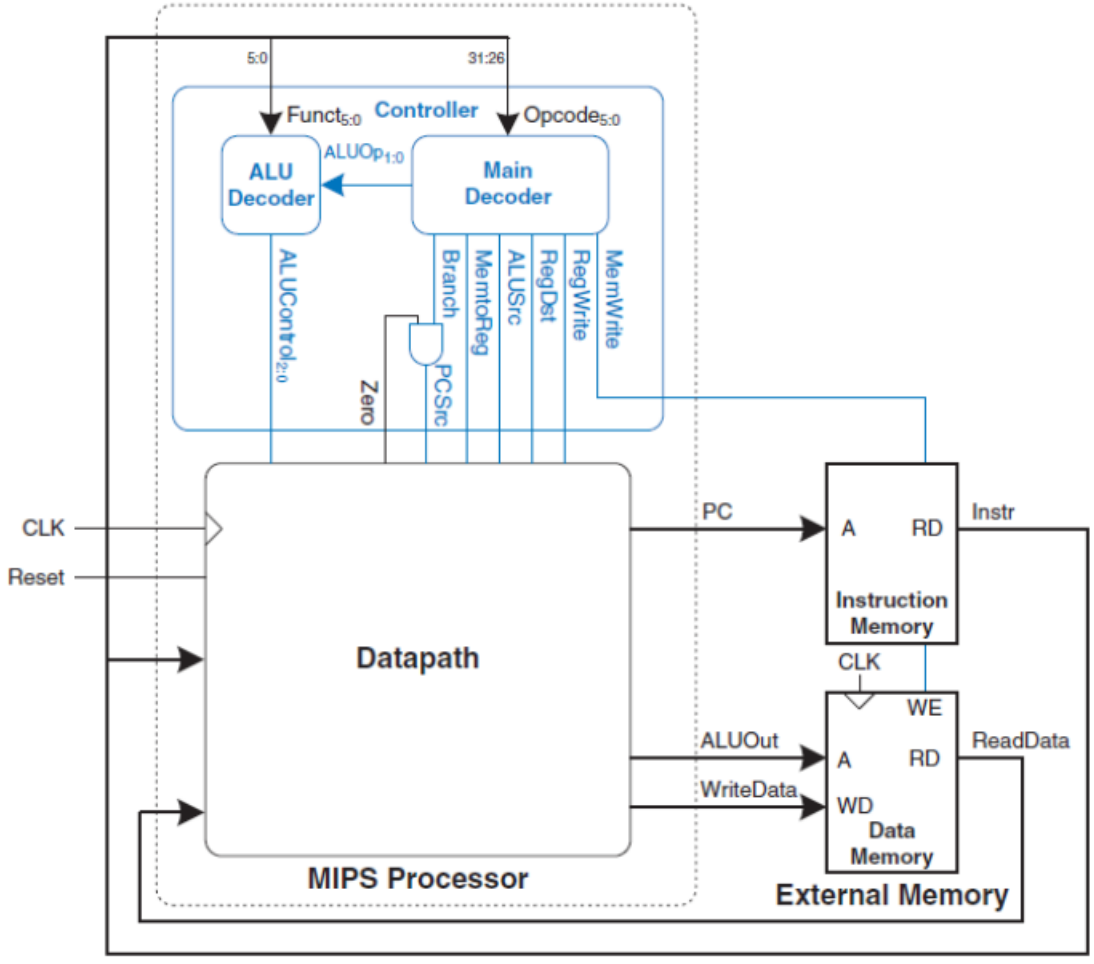
Executable File Header	Text Size	Data Size
	0x58 (88 bytes)	0x0(0 bytes)
Text Segment	**Address**	**Instruction**
	0x00400000	0x20040007
	0x00400004	0x20050001
	0x00400008	0x20100000
	0x0040000c	0x20110003
	0x00400010	0x00855820
	0x00400014	0x216BFFFF
	0x00400018	0x20080000
	0x0040001c	0x1080000D
	0x00400020	0x08100009
	0x00400024	0x12200003
	0x00400028	0x020B8020
	0x0040002c	0x2231FFFF
	0x00400030	0x08100009
	0x00400034	0x0010602A
	0x00400038	0x11800003
	0x0040003c	0x02058022
	0x00400040	0x21080001
	0x00400044	0x0810000D
	0x00400048	0x0200682A
	0x0040004c	0x11A00001
	0x00400050	0x2108FFFF
	0x00400054	0xAC080008
Data Segment	**Address**	**Data**
	-	-

- Memory

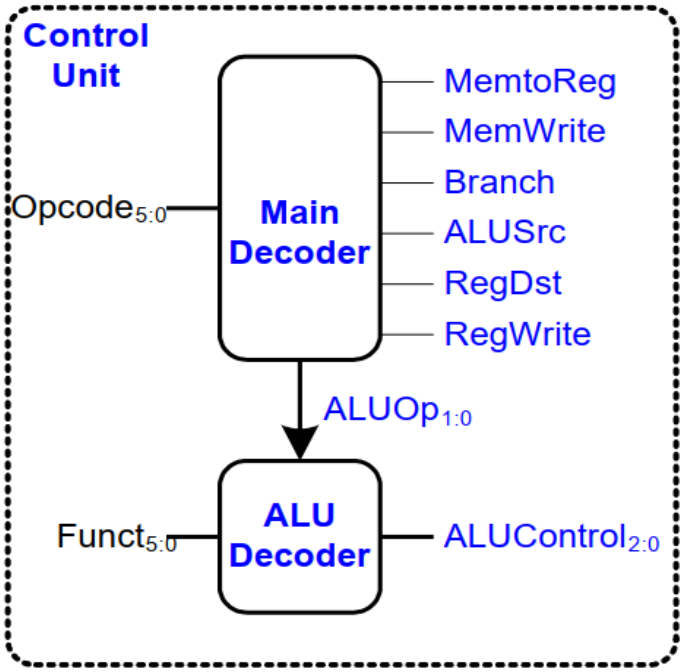

Reserved
0x7FFFFFFC
Stack
$sp = 0x7FFFFFFC
Heap
0x10010000
$gp =0x10080000
0x10000000
0xAC080008
0x2108FFFF
0x11A00001
0x0200682A
0x0810000D
0x21080001
0x02058022
0x11800003
0x0010602A
0x08100009
0x2231FFFF
0x020B8020
0x12200003
0x08100009
0x1080000D
0x20080000
0x216BFFFF
0x00855820
0x20110003
0x20100000
0x20050001
0x00400000
0x20040007
PC = 0x00400000
Reserved

- Testcases with Their testbenches
  - Single Cycle Processor





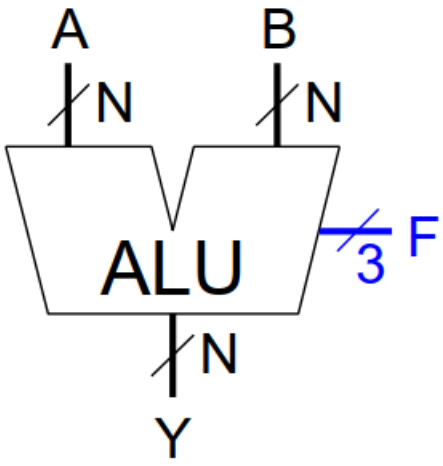
○ Single Cycle Control



$ALUOp_{1:0}$	Meaning
00	Add
01	Subtract
10	Look at Funct
11	Not Used

$ALUOp_{1:0}$	Funct	$ALUControl_{2:0}$
00	X	010 (Add)
X1	X	110 (Subtract)
1X	100000 (add)	010 (Add)
1X	100010 (sub)	110 (Subtract)
1X	100100 (and)	000 (And)
1X	100101 (or)	001 (Or)
1X	101010 (slt)	111 (SLT)

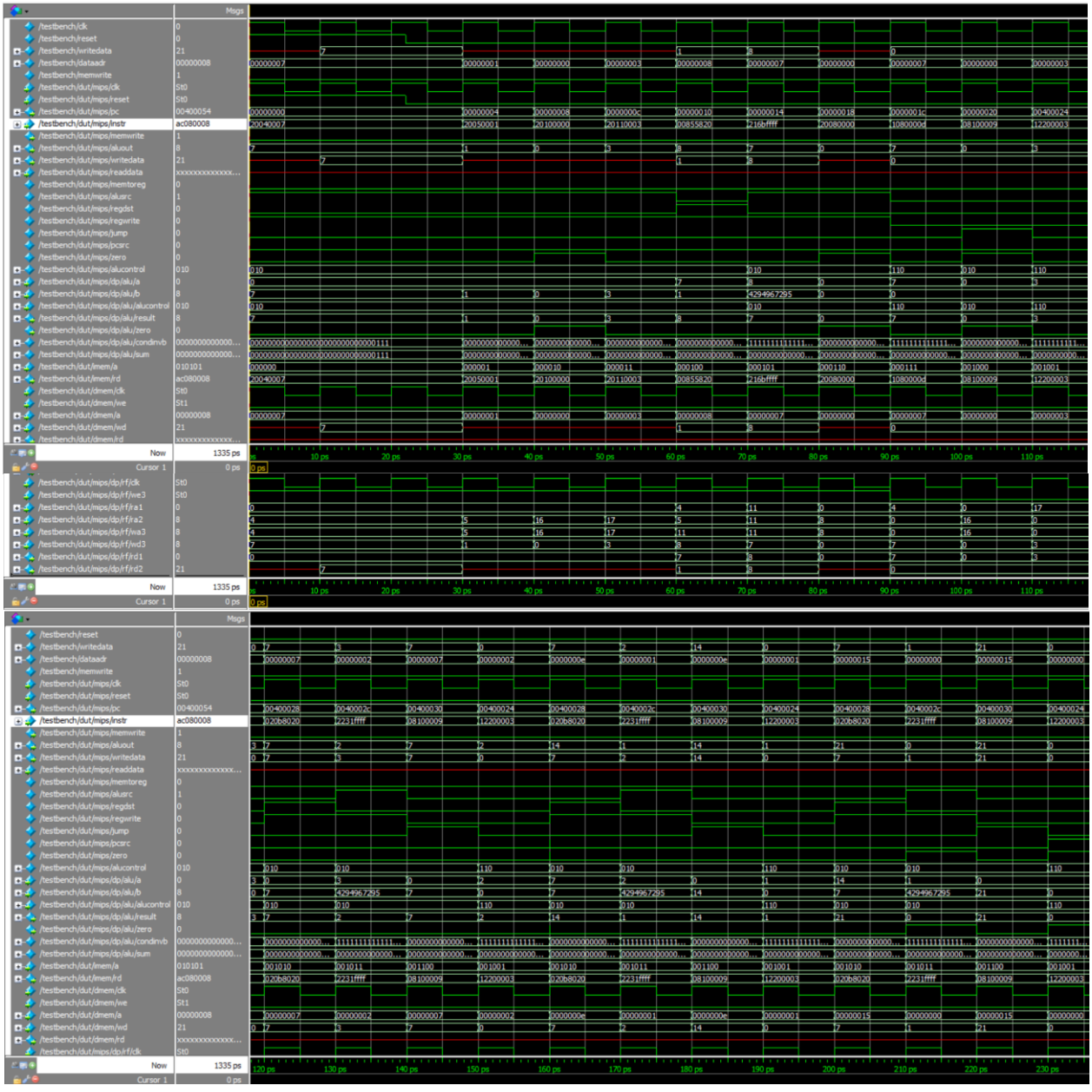
○ ALU

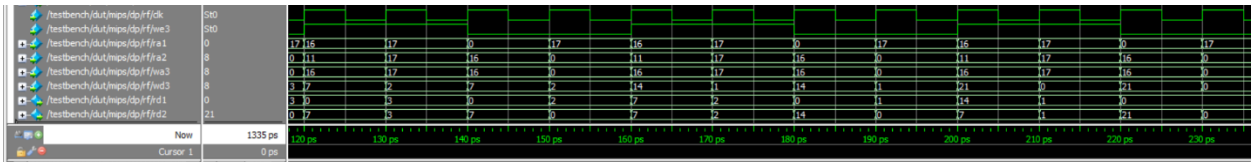


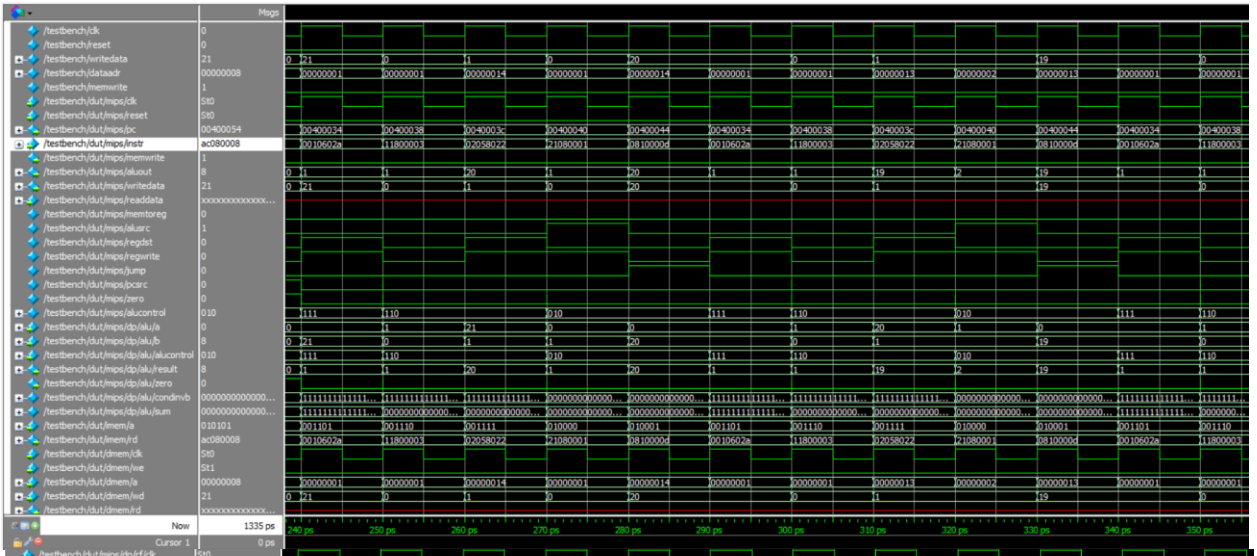
$F_{2:0}$	Function
000	A & B
001	A | B
010	A + B
011	not used
100	A & ~B
101	A | ~B
110	A - B
111	SLT

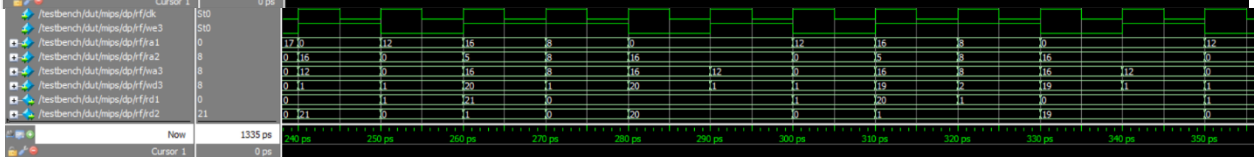
- Case 1 At Tcount = 1, Pcount = 7 Expected Wtime = 21

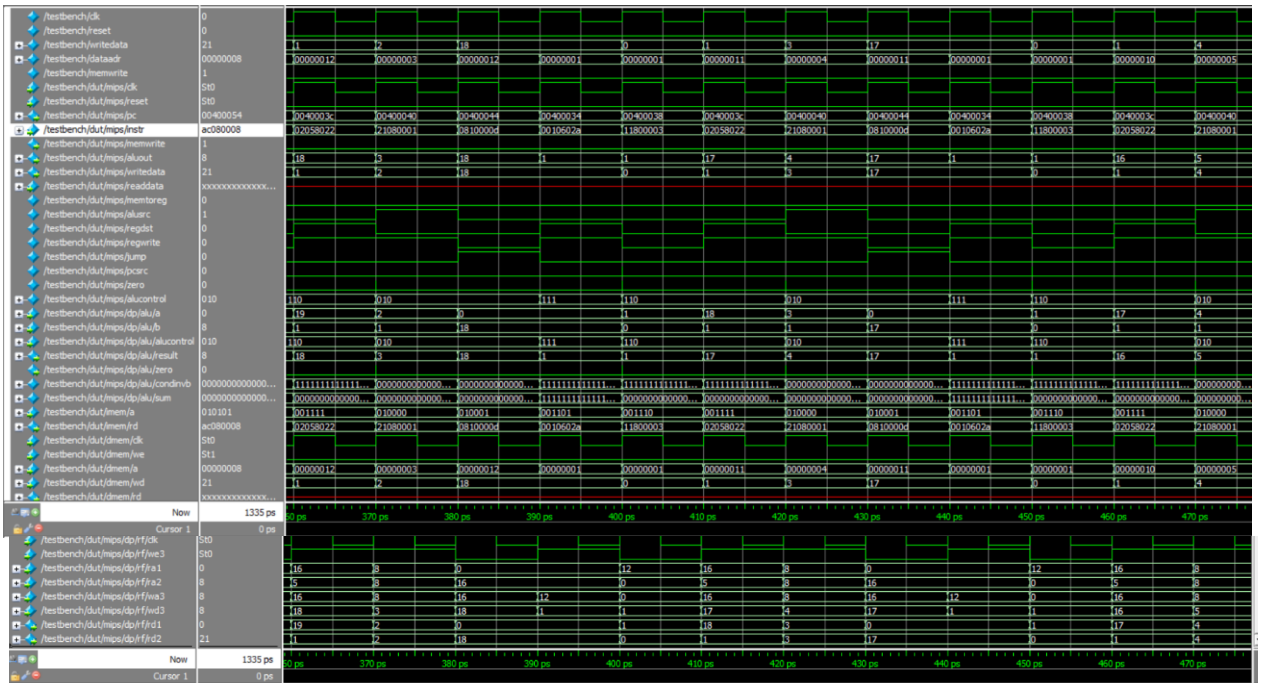
```
#                1220          20          19
#                1230           1           1
#                1250           1           0
#                1260           0           1
#                1270          21          20
#                1280           0           0
#                1330           8          21
# Simulation succeeded
# Break in Module testbench at mipssingle.sv line 37
# Simulation Breakpoint: Break in Module testbench at mipssingle.sv line 37
```



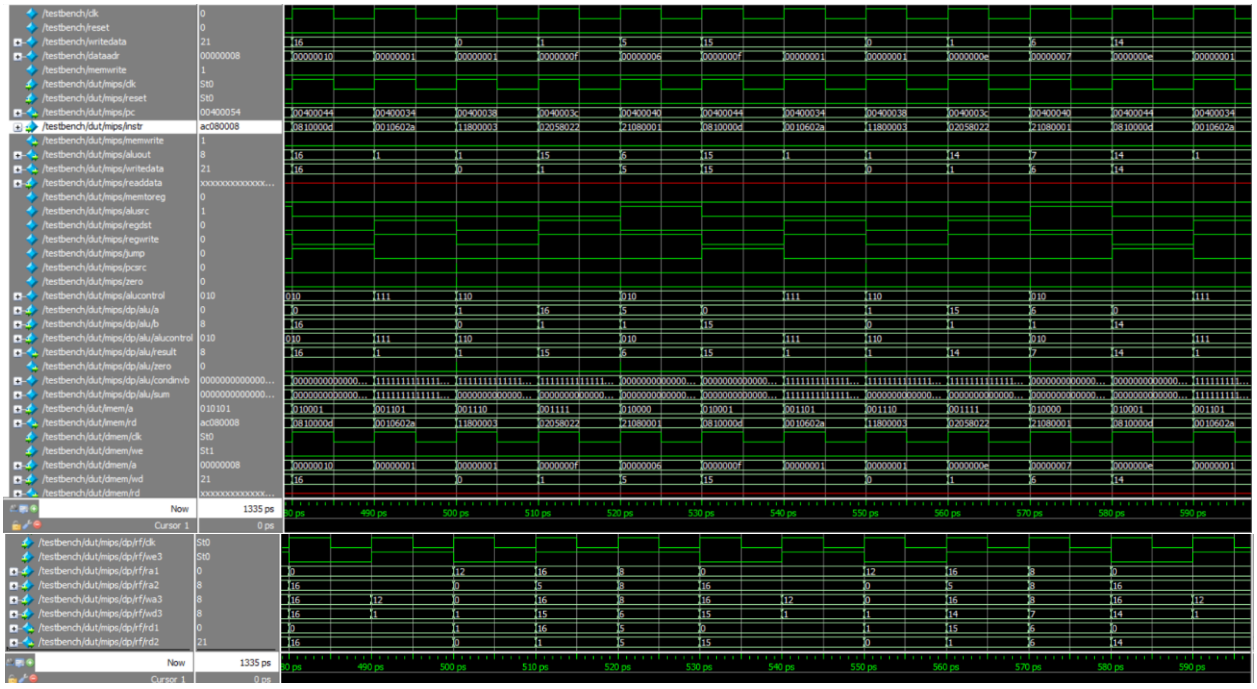

/testbench/dut/mips/dp/rf/clk
/testbench/dut/mips/dp/rf/we3
/testbench/dut/mips/dp/rf/ra1
/testbench/dut/mips/dp/rf/ra2
/testbench/dut/mips/dp/rf/wa3
/testbench/dut/mips/dp/rf/wd3
/testbench/dut/mips/dp/rf/rd1
/testbench/dut/mips/dp/rf/rd2
Now
Cursor 1

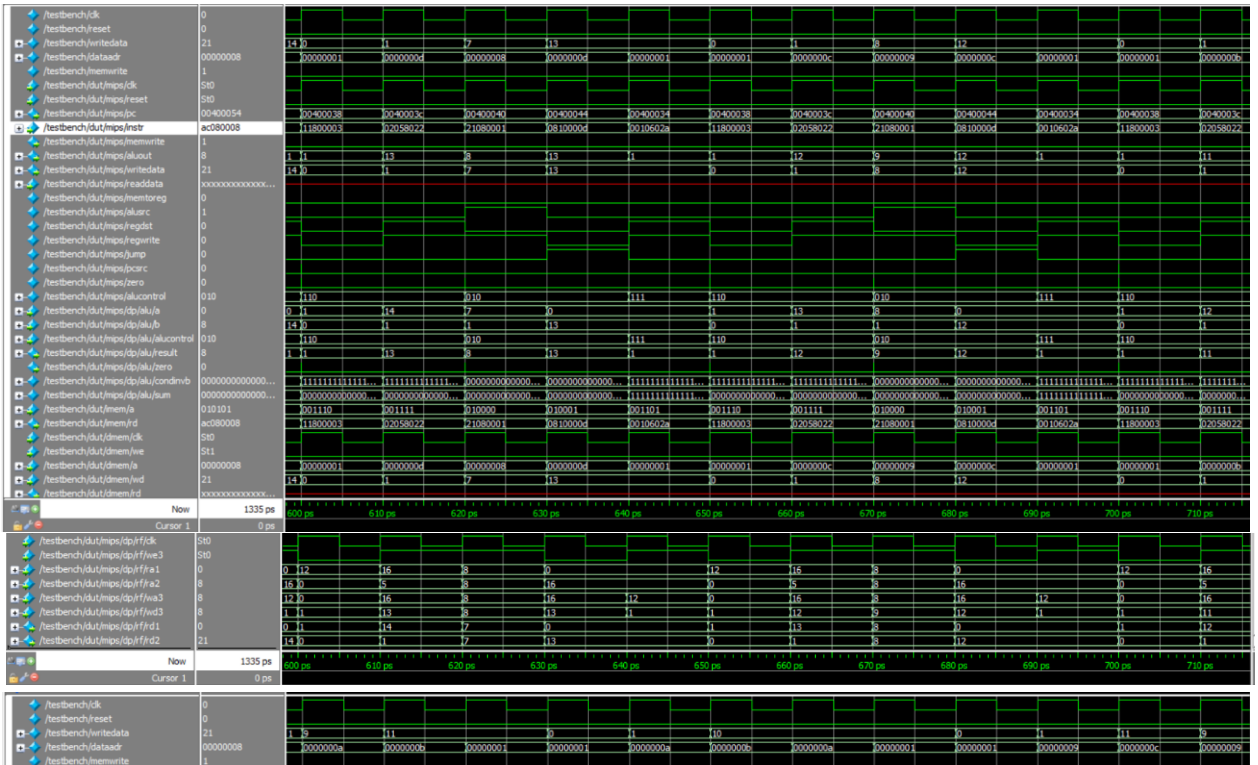

/testbench/clk
/testbench/reset
/testbench/writedata
/testbench/dataadr
/testbench/memwrite
/testbench/dut/mips/clk
/testbench/dut/mips/reset
/testbench/dut/mips/pc
/testbench/dut/mips/instr
/testbench/dut/mips/memwrite
/testbench/dut/mips/aluout
/testbench/dut/mips/writedata
/testbench/dut/mips/readdata
/testbench/dut/mips/memtoreg
/testbench/dut/mips/alusrc
/testbench/dut/mips/regdst
/testbench/dut/mips/regwrite
/testbench/dut/mips/jump
/testbench/dut/mips/pcsrc
/testbench/dut/mips/zero
/testbench/dut/mips/alucontrol
/testbench/dut/mips/dp/alu/a
/testbench/dut/mips/dp/alu/b
/testbench/dut/mips/dp/alu/alucontrol
/testbench/dut/mips/dp/alu/result
/testbench/dut/mips/dp/alu/zero
/testbench/dut/mips/dp/alu/condinvb
/testbench/dut/mips/dp/alu/sum
/testbench/dut/imem/a
/testbench/dut/imem/rd
/testbench/dut/dmem/clk
/testbench/dut/dmem/we
/testbench/dut/dmem/a
/testbench/dut/dmem/wd
/testbench/dut/dmem/rd
Now
Cursor 1

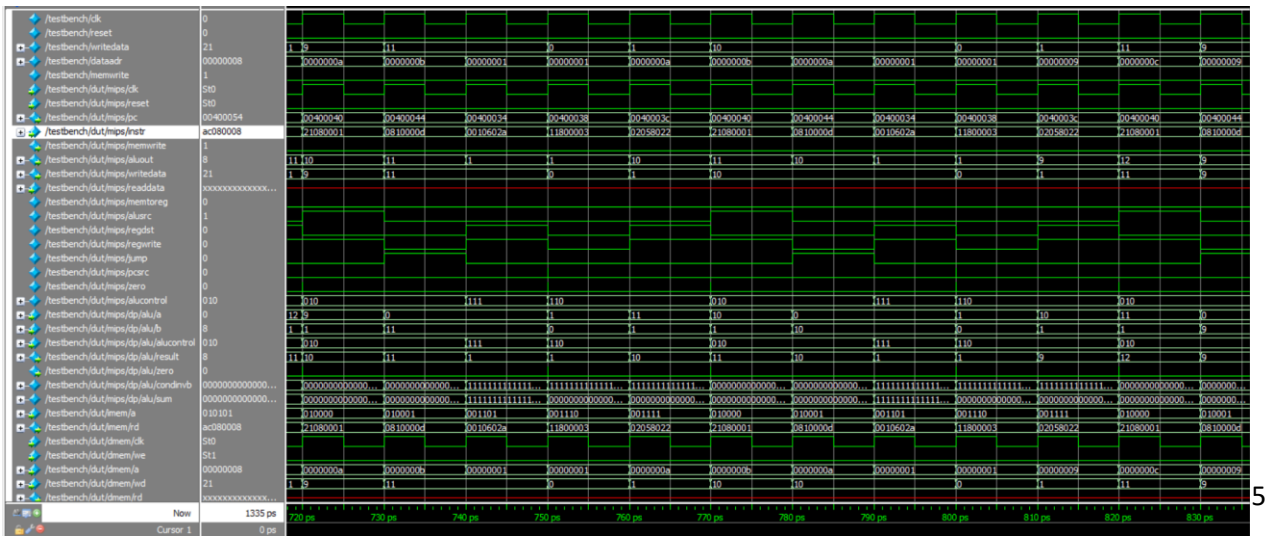

/testbench/dut/mips/dp/rf/clk
/testbench/dut/mips/dp/rf/we3
/testbench/dut/mips/dp/rf/ra1
/testbench/dut/mips/dp/rf/ra2
/testbench/dut/mips/dp/rf/wa3
/testbench/dut/mips/dp/rf/wd3
/testbench/dut/mips/dp/rf/rd1
/testbench/dut/mips/dp/rf/rd2
Now
Cursor 1

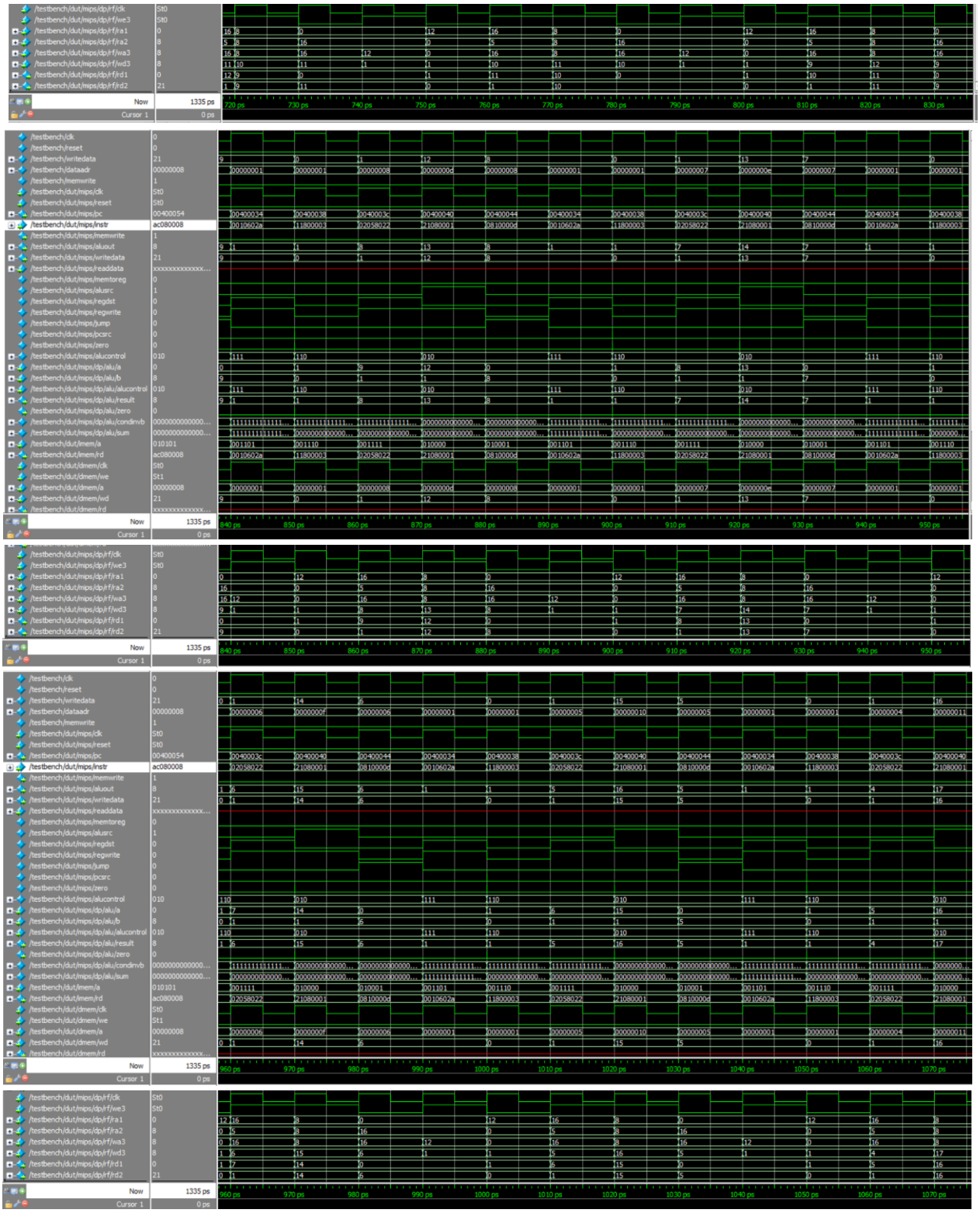

/testbench/clk
/testbench/reset
/testbench/writedata
/testbench/dataadr
/testbench/memwrite
/testbench/dut/mips/clk
/testbench/dut/mips/reset
/testbench/dut/mips/pc
/testbench/dut/mips/instr
/testbench/dut/mips/memwrite
/testbench/dut/mips/aluout
/testbench/dut/mips/writedata
/testbench/dut/mips/readdata
/testbench/dut/mips/memtoreg
/testbench/dut/mips/alusrc
/testbench/dut/mips/regdst
/testbench/dut/mips/regwrite
/testbench/dut/mips/jump
/testbench/dut/mips/pcsrc
/testbench/dut/mips/zero
/testbench/dut/mips/alucontrol
/testbench/dut/mips/dp/alu/a
/testbench/dut/mips/dp/alu/b
/testbench/dut/mips/dp/alu/alucontrol
/testbench/dut/mips/dp/alu/result
/testbench/dut/mips/dp/alu/zero
/testbench/dut/mips/dp/alu/condinvb
/testbench/dut/mips/dp/alu/sum
/testbench/dut/imem/a
/testbench/dut/imem/rd
/testbench/dut/dmem/clk
/testbench/dut/dmem/we
/testbench/dut/dmem/a
/testbench/dut/dmem/wd
/testbench/dut/dmem/rd
Now
Cursor 1

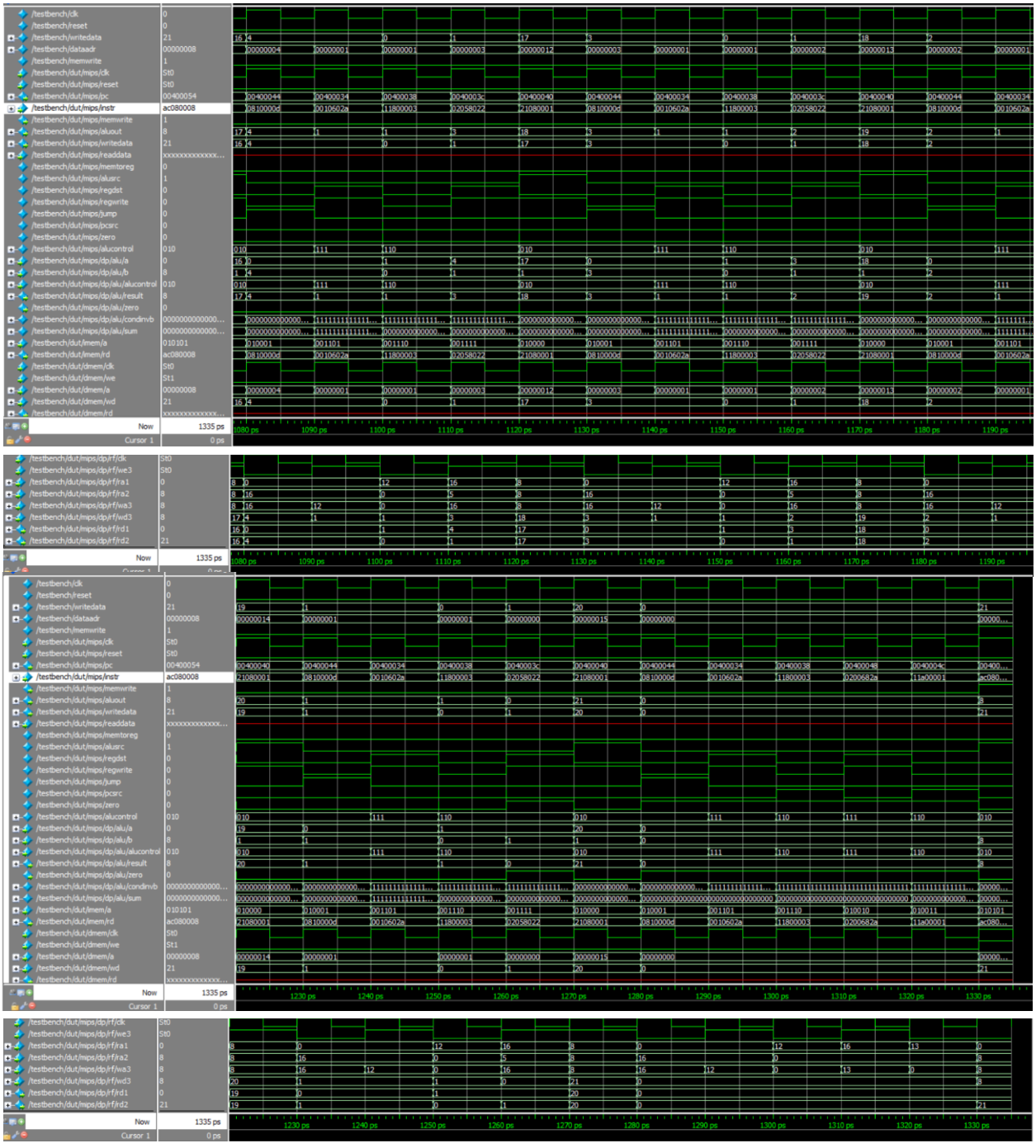
/testbench/dut/mips/dp/rf/clk
/testbench/dut/mips/dp/rf/we3
/testbench/dut/mips/dp/rf/ra1
/testbench/dut/mips/dp/rf/ra2
/testbench/dut/mips/dp/rf/wa3
/testbench/dut/mips/dp/rf/wd3
/testbench/dut/mips/dp/rf/rd1
/testbench/dut/mips/dp/rf/rd2
Now
Cursor 1


/testbench/clk
/testbench/reset
/testbench/writedata
/testbench/dataadr
/testbench/memwrite
/testbench/dut/mips/clk
/testbench/dut/mips/reset
/testbench/dut/mips/pc
/testbench/dut/mips/instr
/testbench/dut/mips/memwrite
/testbench/dut/mips/aluout
/testbench/dut/mips/writedata
/testbench/dut/mips/readdata
/testbench/dut/mips/memtoreg
/testbench/dut/mips/alusrc
/testbench/dut/mips/regdst
/testbench/dut/mips/regwrite
/testbench/dut/mips/jump
/testbench/dut/mips/pcsrc
/testbench/dut/mips/zero
/testbench/dut/mips/alucontrol
/testbench/dut/mips/dp/alu/a
/testbench/dut/mips/dp/alu/b
/testbench/dut/mips/dp/alu/alucontrol
/testbench/dut/mips/dp/alu/result
/testbench/dut/mips/dp/alu/zero
/testbench/dut/mips/dp/alu/condinvb
/testbench/dut/mips/dp/alu/sum
/testbench/dut/imem/a
/testbench/dut/imem/rd
/testbench/dut/dmem/clk
/testbench/dut/dmem/we
/testbench/dut/dmem/a
/testbench/dut/dmem/wd
/testbench/dut/dmem/rd
Now
Cursor 1
/testbench/dut/mips/dp/rf/clk
/testbench/dut/mips/dp/rf/we3
/testbench/dut/mips/dp/rf/ra1
/testbench/dut/mips/dp/rf/ra2
/testbench/dut/mips/dp/rf/wa3
/testbench/dut/mips/dp/rf/wd3
/testbench/dut/mips/dp/rf/rd1
/testbench/dut/mips/dp/rf/rd2


/testbench/clk
/testbench/reset
/testbench/writedata
/testbench/dataadr
/testbench/memwrite
/testbench/dut/mips/clk
/testbench/dut/mips/reset
/testbench/dut/mips/pc
/testbench/dut/mips/instr
/testbench/dut/mips/memwrite
/testbench/dut/mips/aluout
/testbench/dut/mips/writedata
/testbench/dut/mips/readdata
/testbench/dut/mips/memtoreg
/testbench/dut/mips/alusrc
/testbench/dut/mips/regdst
/testbench/dut/mips/regwrite
/testbench/dut/mips/jump
/testbench/dut/mips/pcsrc
/testbench/dut/mips/zero
/testbench/dut/mips/alucontrol
/testbench/dut/mips/dp/alu/a
/testbench/dut/mips/dp/alu/b
/testbench/dut/mips/dp/alu/alucontrol
/testbench/dut/mips/dp/alu/result
/testbench/dut/mips/dp/alu/zero
/testbench/dut/mips/dp/alu/condinvb
/testbench/dut/mips/dp/alu/sum
/testbench/dut/imem/a
/testbench/dut/imem/rd
/testbench/dut/dmem/clk
/testbench/dut/dmem/we
/testbench/dut/dmem/a
/testbench/dut/dmem/wd
/testbench/dut/dmem/rd
Now
Cursor 1
/testbench/dut/mips/dp/rf/clk
/testbench/dut/mips/dp/rf/we3
/testbench/dut/mips/dp/rf/ra1
/testbench/dut/mips/dp/rf/ra2
/testbench/dut/mips/dp/rf/wa3
/testbench/dut/mips/dp/rf/wd3
/testbench/dut/mips/dp/rf/rd1
/testbench/dut/mips/dp/rf/rd2


/testbench/clk
/testbench/reset
/testbench/writedata
/testbench/dataadr
/testbench/memwrite
/testbench/dut/mips/clk
/testbench/dut/mips/reset
/testbench/dut/mips/pc
/testbench/dut/mips/instr
/testbench/dut/mips/memwrite
/testbench/dut/mips/aluout
/testbench/dut/mips/writedata
/testbench/dut/mips/readdata
/testbench/dut/mips/memtoreg
/testbench/dut/mips/alusrc
/testbench/dut/mips/regdst
/testbench/dut/mips/regwrite
/testbench/dut/mips/jump
/testbench/dut/mips/pcsrc
/testbench/dut/mips/zero
/testbench/dut/mips/alucontrol
/testbench/dut/mips/dp/alu/a
/testbench/dut/mips/dp/alu/b
/testbench/dut/mips/dp/alu/alucontrol
/testbench/dut/mips/dp/alu/result
/testbench/dut/mips/dp/alu/zero
/testbench/dut/mips/dp/alu/condinvb
/testbench/dut/mips/dp/alu/sum
/testbench/dut/imem/a
/testbench/dut/imem/rd
/testbench/dut/dmem/clk
/testbench/dut/dmem/we
/testbench/dut/dmem/a
/testbench/dut/dmem/wd
/testbench/dut/dmem/rd
Now
Cursor 1




/testbench/clk
/testbench/reset
/testbench/writedata
/testbench/dataadr
/testbench/memwrite
/testbench/dut/mips/clk
/testbench/dut/mips/reset
/testbench/dut/mips/pc
/testbench/dut/mips/instr
/testbench/dut/mips/memwrite
/testbench/dut/mips/aluout
/testbench/dut/mips/writedata
/testbench/dut/mips/readdata
/testbench/dut/mips/memtoreg
/testbench/dut/mips/alusrc
/testbench/dut/mips/regdst
/testbench/dut/mips/regwrite
/testbench/dut/mips/jump
/testbench/dut/mips/pcsrc
/testbench/dut/mips/zero
/testbench/dut/mips/aluControl
/testbench/dut/mips/dp/alu/a
/testbench/dut/mips/dp/alu/b
/testbench/dut/mips/dp/alu/aluControl
/testbench/dut/mips/dp/alu/result
/testbench/dut/mips/dp/alu/zero
/testbench/dut/mips/dp/alu/condinvb
/testbench/dut/mips/dp/alu/sum
/testbench/dut/mem/a
/testbench/dut/mem/rd
/testbench/dut/dmem/clk
/testbench/dut/dmem/we
/testbench/dut/dmem/a
/testbench/dut/dmem/wd
/testbench/dut/dmem/rd
Now
1335 ps
Cursor 1
0 ps
1080 ps
1090 ps
1100 ps
1110 ps
1120 ps
1130 ps
1140 ps
1150 ps
1160 ps
1170 ps
1180 ps
1190 ps

/testbench/dut/mips/dp/rf/clk
/testbench/dut/mips/dp/rf/we3
/testbench/dut/mips/dp/rf/ra1
/testbench/dut/mips/dp/rf/ra2
/testbench/dut/mips/dp/rf/wa3
/testbench/dut/mips/dp/rf/wd3
/testbench/dut/mips/dp/rf/rd1
/testbench/dut/mips/dp/rf/rd2
Now
1335 ps
1080 ps
1090 ps
1100 ps
1110 ps
1120 ps
1130 ps
1140 ps
1150 ps
1160 ps
1170 ps
1180 ps
1190 ps

/testbench/clk
/testbench/reset
/testbench/writedata
/testbench/dataadr
/testbench/memwrite
/testbench/dut/mips/clk
/testbench/dut/mips/reset
/testbench/dut/mips/pc
/testbench/dut/mips/instr
/testbench/dut/mips/memwrite
/testbench/dut/mips/aluout
/testbench/dut/mips/writedata
/testbench/dut/mips/readdata
/testbench/dut/mips/memtoreg
/testbench/dut/mips/alusrc
/testbench/dut/mips/regdst
/testbench/dut/mips/regwrite
/testbench/dut/mips/jump
/testbench/dut/mips/pcsrc
/testbench/dut/mips/zero
/testbench/dut/mips/aluControl
/testbench/dut/mips/dp/alu/a
/testbench/dut/mips/dp/alu/b
/testbench/dut/mips/dp/alu/aluControl
/testbench/dut/mips/dp/alu/result
/testbench/dut/mips/dp/alu/zero
/testbench/dut/mips/dp/alu/condinvb
/testbench/dut/mips/dp/alu/sum
/testbench/dut/mem/a
/testbench/dut/mem/rd
/testbench/dut/dmem/clk
/testbench/dut/dmem/we
/testbench/dut/dmem/a
/testbench/dut/dmem/wd
/testbench/dut/dmem/rd
Now
1335 ps
Cursor 1
0 ps
1230 ps
1240 ps
1250 ps
1260 ps
1270 ps
1280 ps
1290 ps
1300 ps
1310 ps
1320 ps
1330 ps

/testbench/dut/mips/dp/rf/clk
/testbench/dut/mips/dp/rf/we3
/testbench/dut/mips/dp/rf/ra1
/testbench/dut/mips/dp/rf/ra2
/testbench/dut/mips/dp/rf/wa3
/testbench/dut/mips/dp/rf/wd3
/testbench/dut/mips/dp/rf/rd1
/testbench/dut/mips/dp/rf/rd2
Now
1335 ps
1230 ps
1240 ps
1250 ps
1260 ps
1270 ps
1280 ps
1290 ps
1300 ps
1310 ps
1320 ps
1330 ps

- Case 2 At Tcount = 1, Pcount = 1 Expected Wtime = 3

```
#                  290             1              2
#                  300             1              0
#                  310             1              1
#                  320             2              1
#                  330             1              1
#                  350             1              0
#                  360             0              1
#                  370             3              2
#                  380             0              0
#                  430             8              3
# Simulation succeeded
# Break in Module testbench at mipssingle.sv line 37
# Simulation Breakpoint: Break in Module testbench at mipssingle.sv line 37
```