Intro To Processor Architecture

Course Project: Building a Y-86 Processor

Abhinav Marri
2021112015

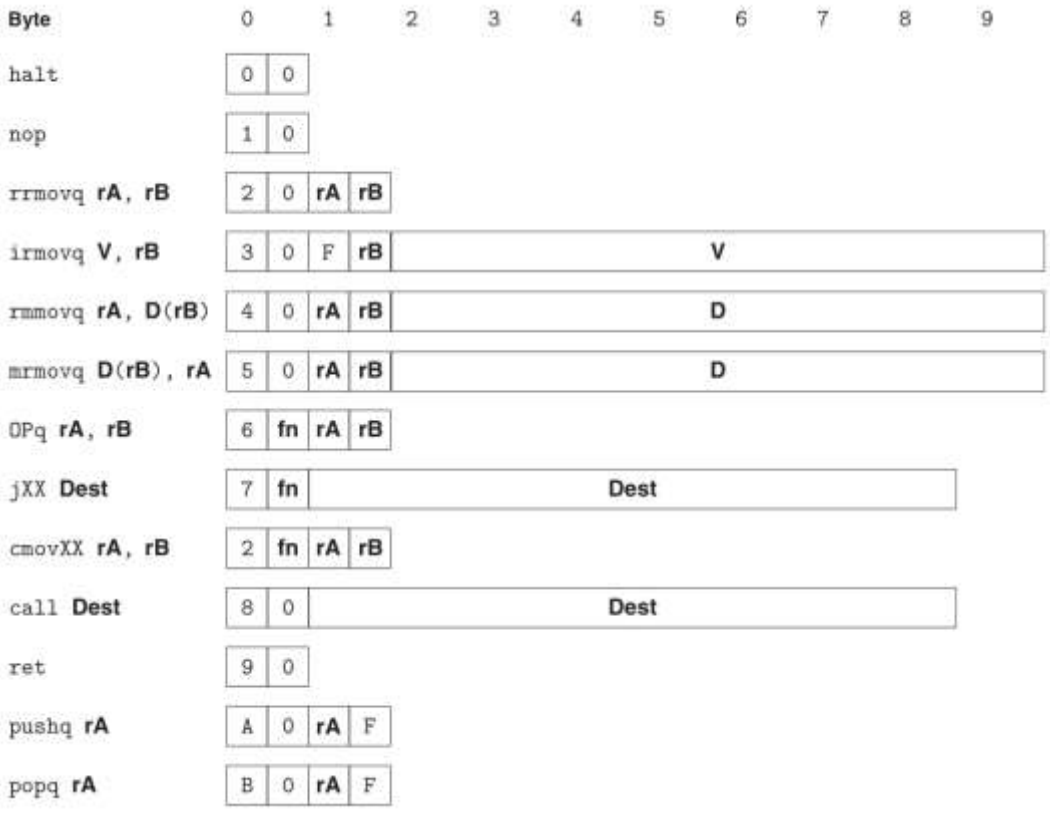
Vedansh Agrawal
2021112010

OBJECTIVE :

Our Goal is to build a Y-86 processor that makes use of pipeline architecture. We will do this by first implementing a sequential processor. We will be using a modular approach to build a 5-stage pipelined processor. Namely the 5 stages will be Fetch, Decode, Execute, Memory and Writeback stages. The HDL iverilog is used to achieve this implementation.

Y86 Instruction set Architecture :

The Y86 ISA is a simplified version of the X86 ISA that is used it most modern-day intel processors. An Instruction Set Architecture (ISA) is part of the abstract model of a computer that defines how the CPU is controlled by the software. It consists of the instructions supported by a particular processor and their byte-level encodings.



The Y86 Instruction Set Architecture

The instructions cmovxx, OPq, jxx have multiple functions that is dictated by the ifun as shown :

Operations

addq	6	0
subq	6	1
andq	6	2
xorq	6	3

Branches

jmp	7	0	jne	7	4
jle	7	1	jge	7	5
jl	7	2	jg	7	6
je	7	3			

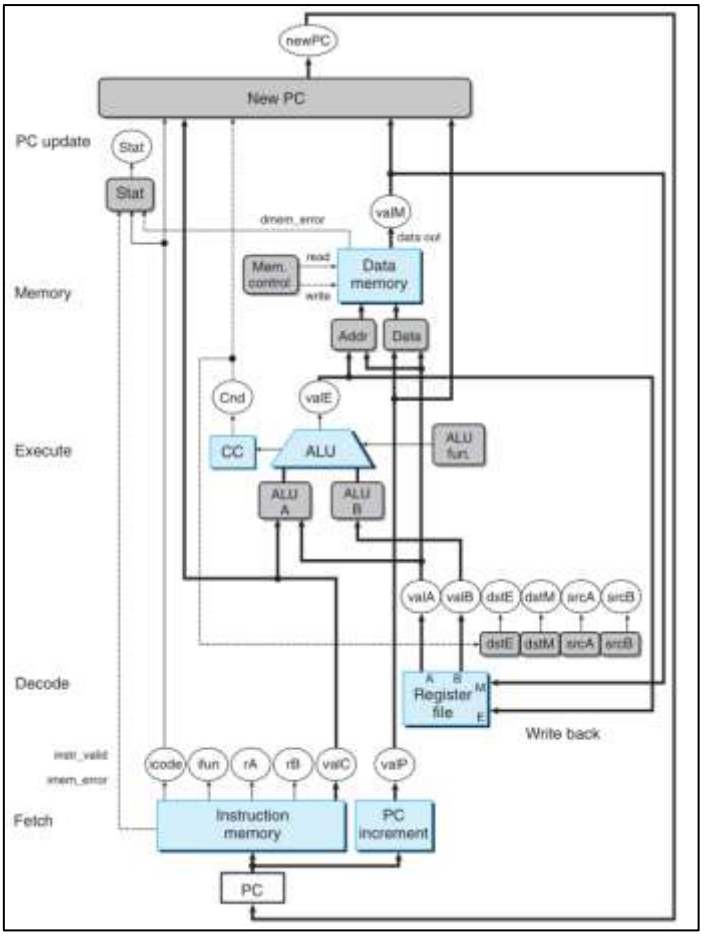
Moves

rrmovq	2	0	cmovne	2	4
cmovle	2	1	cmovge	2	5
cmovl	2	2	cmovg	2	6
cmove	2	3			

This ISA makes use of 15 registers labelled from %rax to %r14.

SEQUENTIAL DESIGN :

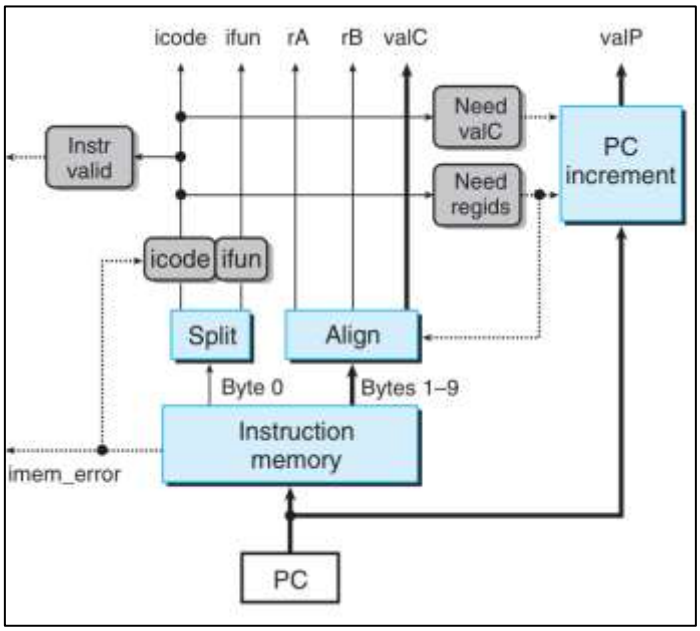
The sequential design of a processor is an architecture that processes instructions one after the other. An instruction is processed all the way from fetching it and then its execution is completed before fetching the next instruction.



The following stages are present in the SEQ processor.

- **FETCH :**

The fetch stage fetches instructions from the instruction memory, and splits the encoded instructions into '*icode*' , '*ifun'* and '*align*'. It also sets the instr_valid bit which dictates whether the instruction read is valid or invalid. It computes the value of '*valP*' and also stores a constant '*valC'* that is fetched.



Sequential fetch stage

We made use of two modules, one called instruction_memory that stores all the input instructions and also splits the encoding into split and align, the fetch module that receives them and further obtains '*icode'* and '*ifun'*.

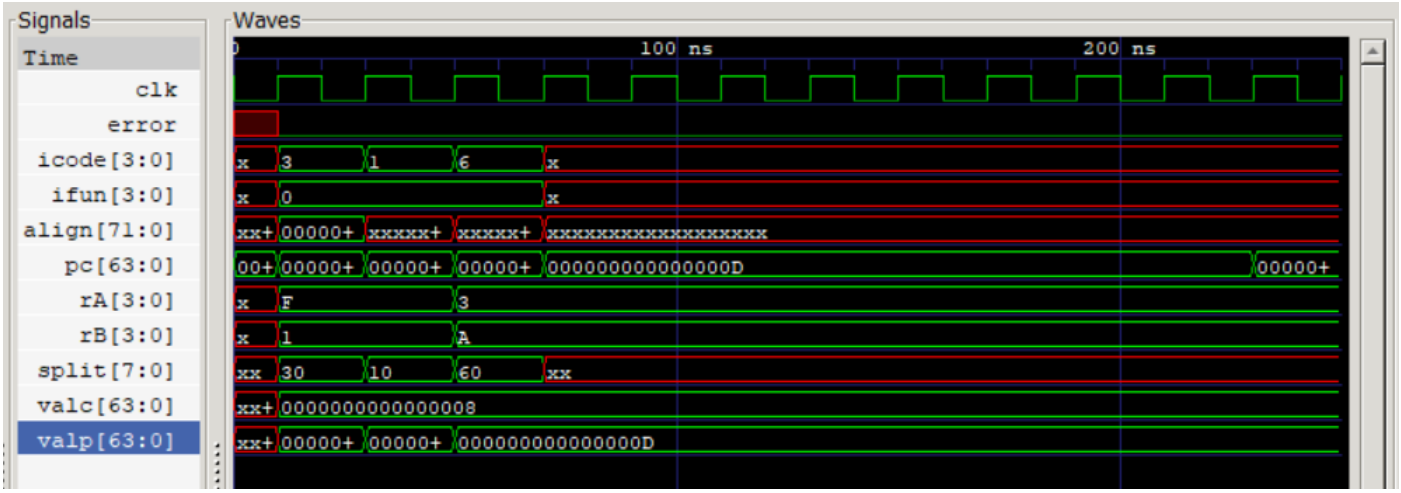
Testing of the Fetch stage :

We used a particular set of inputs and tested if the encodings inputted are getting read properly and if they are being split into icode, ifun etc properly.

Inputs : (opcodes)

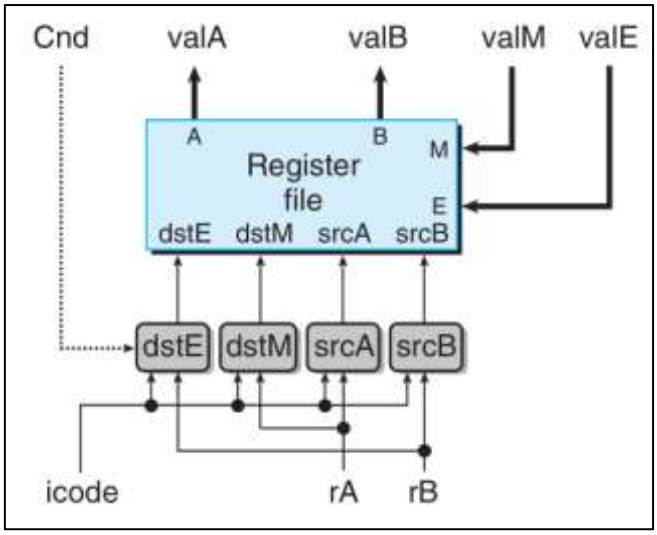
```
30
F1
08
00
00
00
00
00
00
00
10
60
3a
```

```
icode = 3, ifun = 0, rA = f, rB = 1, valc = 0000000000000008, valp = 000000000000000a,
 error = 0, instr_valid = 1
icode = 1, ifun = 0, rA = f, rB = 1, valc = 0000000000000008, valp = 000000000000000b,
 error = 0, instr_valid = 1
icode = 6, ifun = 0, rA = 3, rB = a, valc = 0000000000000008, valp = 000000000000000d,
 error = 0, instr_valid = 1
```



- **DECODE / WRITE-BACK :**

The decode stage reads values from the respective registers and updates valA and valB. We make use of a file that reads and writes registers. We implement the decode and the writeback stages in one verilog file for convenience.

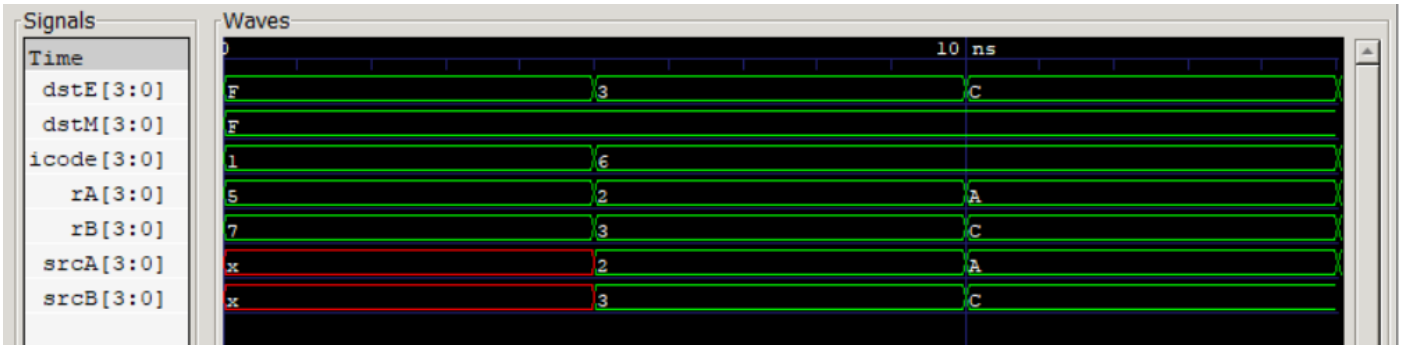


SEQ decode and writeback

Testing of the Decode stage :

We input icode, registers rA and rB, the decode stage outputs dstA, dstB, dstE and dstM based on our inputs.

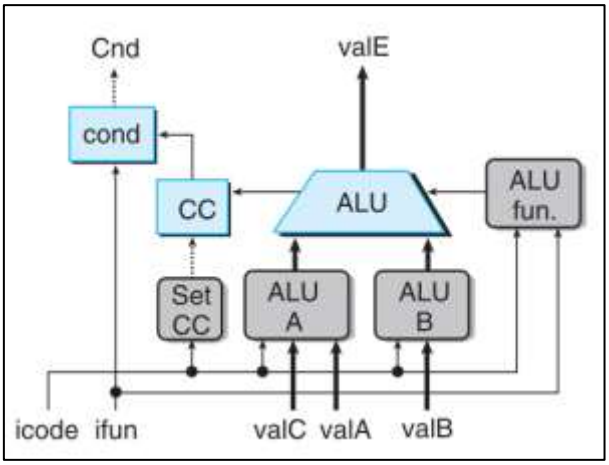
```
icode = 1,rA = 5,rB = 7,
srcA = x, srcB = x, dstE = f, dstM = f
icode = 6,rA = 2,rB = 3,
srcA = 2, srcB = 3, dstE = 3, dstM = f
icode = 6,rA = a,rB = c,
srcA = a, srcB = c, dstE = c, dstM = f
icode = 2,rA = 5,rB = 7,
srcA = 5, srcB = c, dstE = f, dstM = f
```



- **EXECUTE :**

The execute stage is the stage where operations are performed by the arithmetic and logic unit (ALU). The ALU can perform '*add*', '*sub*', '*and*' and '*xor*'. The ALU chooses which of these to perform based on a control signal that is given to the ALU. The control signal is based on the ifun value. The ALU outputs the result 'valE' and sets the condition code registers. These registers tell us about the most recent operation that has taken place, these are the following condition codes :

- Overflow Flag
- Zero flag
- Sign flag



SEQ execute stage

Let us investigate the working of the ALU :

Approach to build ALU:

- The ALU is an arithmetic block which performs and outputs the required operation based on the control signal.
- The ALU can perform the following operations :
  1. ADD
  2. SUBTRACT
  3. AND
  4. XOR
- We develop separate blocks for all the above functions and call them using the ALU based on the control signal. The control signal is as given for the ALU designed :

i. 2’b00 = ADD
ii. 2’b01 = SUB
iii. 2’b10 = AND
iv. 2’b11 = XOR

Let us now check out the working of the 4 blocks individually.

➢ ADDER BLOCK

The adder block consists of using full adders and a half adder . They are looped to perform the addition operation for 64 bits. We are implementing a ripple adder which adds the bits using a loop in Verilog by using generate function.

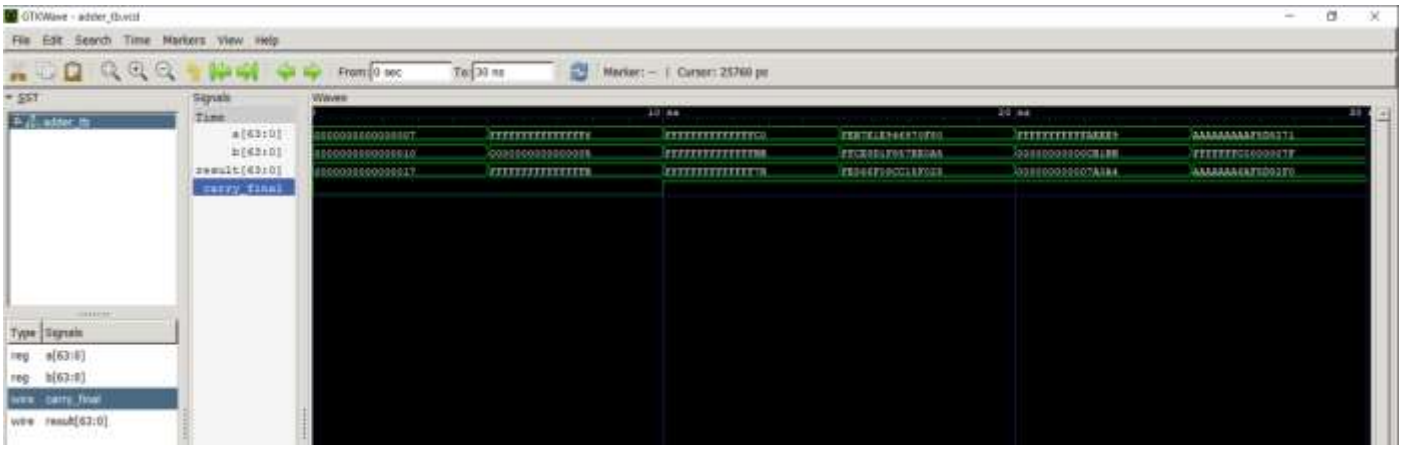
I/O format :

Input two 64-bit binary numbers a and b. The adder will output the result after performing ‘ a + b ’ and the final carry bit (also known as the overflow bit).

Testing :

The inputs in the testbenches can be given as decimal numbers, the following inputs have been given to try to cover all possible cases of I/O combinations.

GTKWAVE PLOTS :



Observations :

We can observe that the shown plots are as expected. The carry bit is produced whenever the result of the addition operation is a number too big to be represented by 64 bits. We have chosen the following cases to show the working of our adder block :

- Positive positive without carry
- Positive positive with carry
- Positive negative (no carry possible)
- Negative negative (always gives carry)

➢ SUBTRACTER BLOCK

The subtracter block uses the concept of 2's compliment of a number and the adder block to perform subtraction of 2 64-bit numbers. The 2's compliment of the second input is generated which is then added to the first input. This is equivalent to performing in1 – in2. The way we find the 2's compliment of the input is by taking the compliment of every bit by running a for loop and then using the adder to add 1 to the complimented number. This will give the 2's compliment of the number.

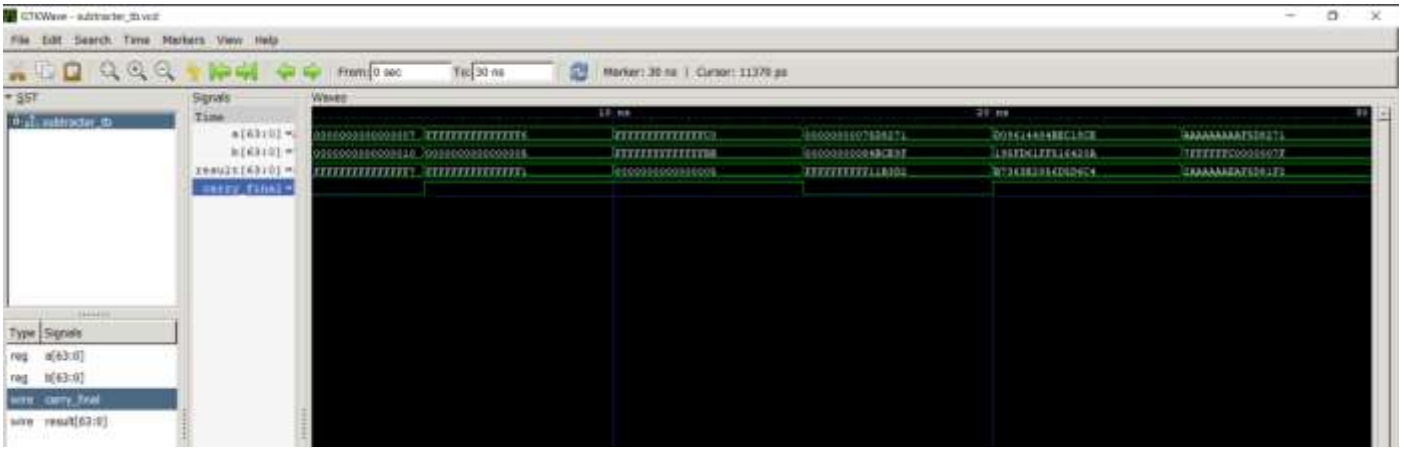
I/O format :

Input two 64-bit binary numbers a and b. The subtracter will output the result after performing ' a - b ' and the final carry bit (also known as the overflow bit).

Testing :

The inputs in the testbenches can be given as decimal numbers, the following inputs have been given to try to cover all possible cases of I/O combinations.

GTKWAVE PLOTS :



Observations :

We can observe that the shown plots are as expected. We have chosen the following cases to show the working of our subtracter block :

- Positive positive without carry
- Positive negative with carry
- Positive negative without carry
- negative positive with carry
- Negative negative with carry

➢ AND BLOCK

The and block performs bitwise and for 2 64-bit inputs. This is done by calling the *and* function for the corresponding bits by running a loop through every bit and performing AND on them one by one to achieve the output.

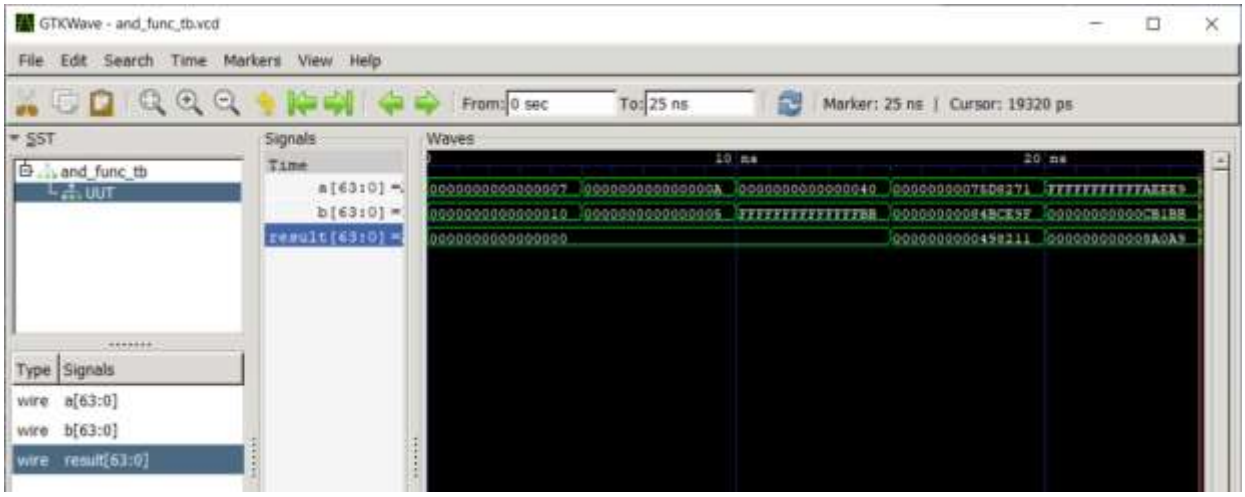
I/O format :

Input two 64-bit binary numbers a and b. The and block will output the result after performing ‘ a & b (bitwise)’ and a 0 (overflow bit).

Testing :

The inputs in the testbenches can be given as decimal numbers, the following inputs have been given to try to cover all possible cases of I/O combinations.

GTKWAVE PLOTS :



vvp output:

```
PS C:\iverilog\bin> ./iverilog.exe -o and_tb and_tb.v and.v
PS C:\iverilog\bin> ./vvp and_tb
VCD info: dumpfile and_func_tb.vcd opened for output.
a = 0000000000000000000000000000000000000000000000000000000000000111
b = 0000000000000000000000000000000000000000000000000000000000010000
result = 0000000000000000000000000000000000000000000000000000000000000000
a = 0000000000000000000000000000000000000000000000000000000000001010
b = 0000000000000000000000000000000000000000000000000000000000000101
result = 0000000000000000000000000000000000000000000000000000000000000000
a = 0000000000000000000000000000000000000000000000000000000001000000
b = 1111111111111111111111111111111111111111111111111111111110111011
result = 0000000000000000000000000000000000000000000000000000000000000000
a = 0000000000000000000000000000000011101011101100000100111000 1
b = 0000000000000000000000000000000010000100101111001110100111 11
result = 0000000000000000000000000000000000000000000000000000000000000000
a = 0000000000000000000000000000000011101011101100000100111000 1
b = 0000000000000000000000000000000010000100101111001110100111 11
result = 0000000000000000000000000000000000000000000000001001001100000100001000 1
a = 1111111111111111111111111111111111111111111111101011101110111010 1
b = 0000000000000000000000000000000000000000000000001100101100011011 1011
result = 0000000000000000000000000000000000000000000000001001001100000100001000 1
a = 1111111111111111111111111111111111111111111111101011101110111010 1
b = 0000000000000000000000000000000000000000000000001100101100011011 1011
result = 0000000000000000000000000000000000000000000000001000101000001010 1001
a = 1010101010101010101010101010101010111010111011000001001110001
b = 1111111111111111111111111111111100000000000000000000000001111111
result = 0000000000000000000000000000000000000000000000001000101000001010 1001
a = 1010101010101010101010101010101010111010111011000001001110001
b = 1111111111111111111111111111111100000000000000000000000001111111
result = 1010101010101010101010101010100000000000000000000000000001110001
PS C:\iverilog\bin> ./gtkwave.exe and_func_tb.vcd
```

- XOR BLOCK

The XOR block performs bitwise XOR for 2 64-bit inputs. This is done by calling the *xor* function for the corresponding bits by running a loop through every bit and performing AND on them one by one to achieve the output.

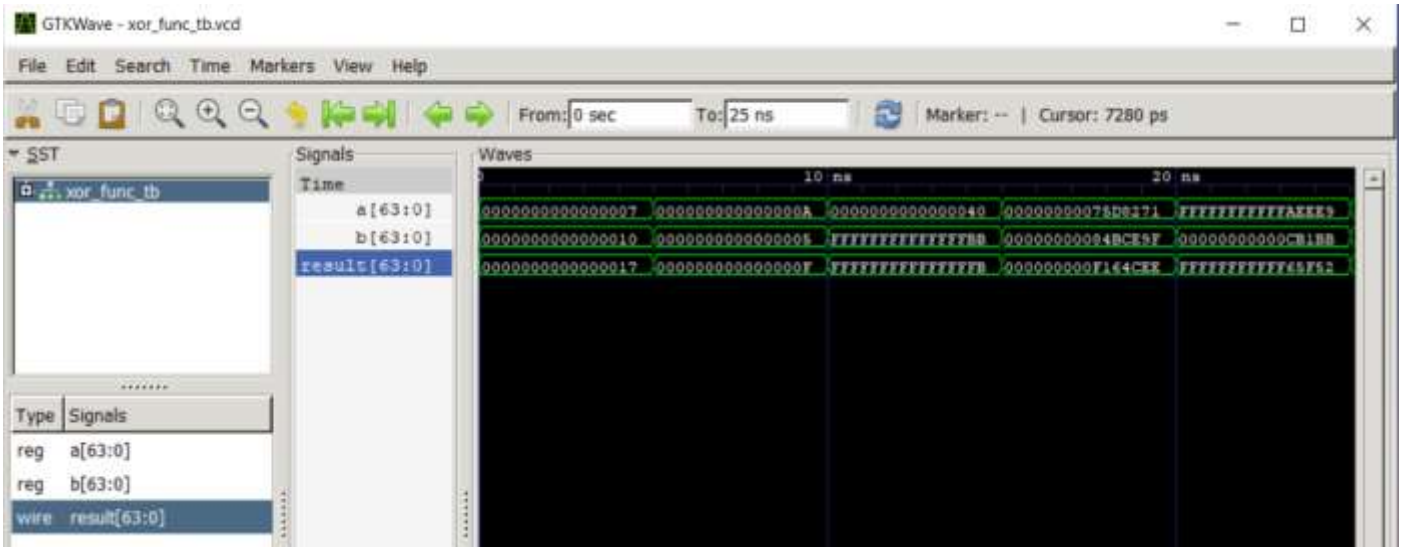
I/O format :

Input two 64-bit binary numbers a and b. The XOR block will output the result after performing ‘ a ^ b (bitwise)’ and a 0 (overflow bit).

Testing :

The inputs in the testbenches can be given as decimal numbers, the following inputs have been given to try to cover all possible cases of I/O combinations.

GTKWAVE PLOTS :



vvp output :

```
VCD info: dumpfile xor_func_tb.vcd opened for output.
a = 0000000000000000000000000000000000000000000000000000000000000111
b = 0000000000000000000000000000000000000000000000000000000000010000
result = 0000000000000000000000000000000000000000000000000000000000010111
a = 0000000000000000000000000000000000000000000000000000000000001010
b = 0000000000000000000000000000000000000000000000000000000000000101
result = 0000000000000000000000000000000000000000000000000000000000010111
a = 0000000000000000000000000000000000000000000000000000000000001010
b = 0000000000000000000000000000000000000000000000000000000000000101
result = 0000000000000000000000000000000000000000000000000000000000001111
a = 0000000000000000000000000000000000000000000000000000000001000000
b = 1111111111111111111111111111111111111111111111111111111110111011
result = 0000000000000000000000000000000000000000000000000000000001000000
a = 0000000000000000000000000000000000000000000000000000000001000000
b = 1111111111111111111111111111111111111111111111111111111110111011
result = 1111111111111111111111111111111111111111111111111111111111111011
a = 0000000000000000000000000000000000000000011101011101100000001110001
b = 0000000000000000000000000000000000000000100001001011110011101011111
result = 1111111111111111111111111111111111111111111111111111111111111011
a = 0000000000000000000000000000000000000000011101011101100000001110001
b = 0000000000000000000000000000000000000000100001001011110011101011111
result = 0000000000000000000000000000000000000000111100010110010011001110110
a = 1111111111111111111111111111111111111111111111111111111101011101111011101001
b = 0000000000000000000000000000000000000000000000000000000011001011000110111011
result = 0000000000000000000000000000000000000000111100010110010011001110110
a = 1111111111111111111111111111111111111111111111111111111101011101111011101001
b = 0000000000000000000000000000000000000000000000000000000011001011000110111011
result = 1111111111111111111111111111111111111111111111111111111101100101111101010010
a = 1010101010101010101010101010101010101111010111010110000001001110001
b = 1111111111111111111111111111111100000000000000000000000000001111111
result = 1111111111111111111111111111111111111111111111111111111101100101111101010010
a = 1010101010101010101010101010101010101111010111010110000001001110001
b = 1111111111111111111111111111111100000000000000000000000000001111111
result = 0101010101010101010101010101011010101111010111011000001000001110
```

THE ALU (WRAPPER) BLOCK :

The ALU block is a module that will take the input of a control signal and choose which operation to perform out of the above 4 and then outputs it through the implementation of a mux.

Implementation :

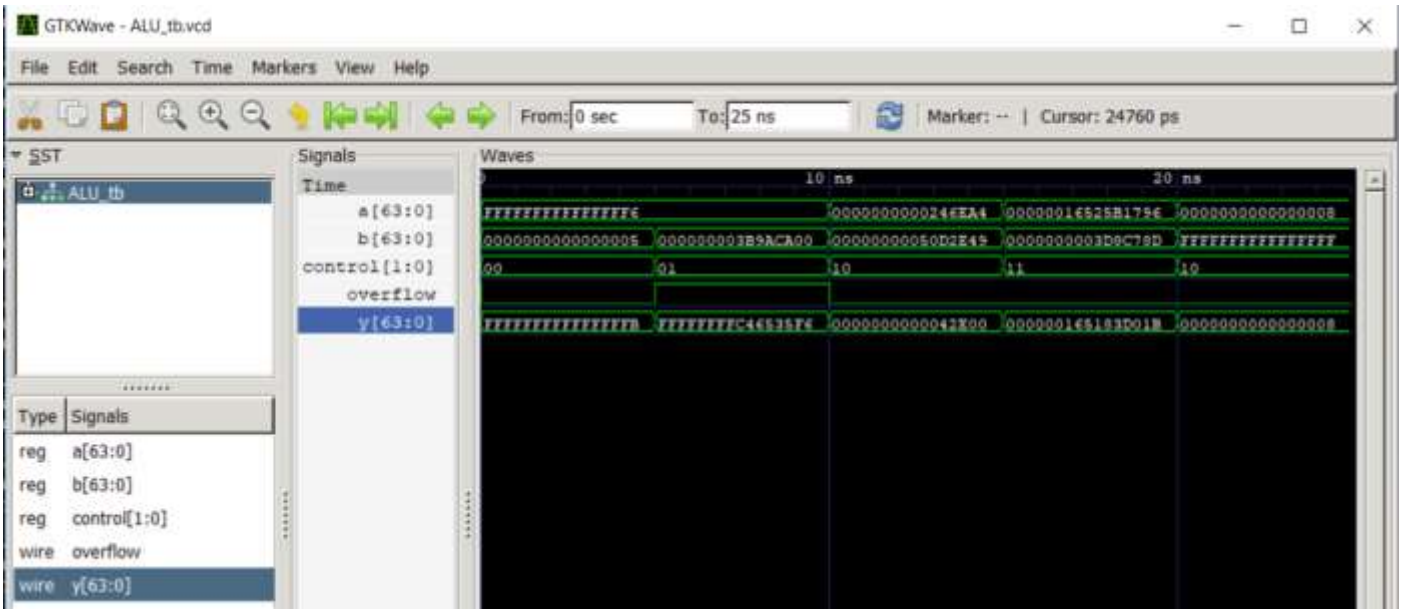
ALU.v

```
1   `include "subtracter.v"
2   `include "xor.v"
3   `include "and.v"
4
5   module  ALU(control,a,b,y,overflow);
6
7   input [63:0] a,b;
8   input [1:0] control;
9   output reg signed overflow;
10  wire [63:0] result1,result2,result3,result4;
11  output reg signed [63:0] y;
12
13  adder ADD_2 (a,b,result1,carry1);
14  subtracter SUB (a,b,result2,carry2);
15  and_func AND_1 (a,b,result3,carry3);
16  xor_func XOR_1 (a,b,result4);
17
18  always@(*) begin
19
20  case(control)
21
22      2'b00 :
23      begin y <= result1;
24      overflow = carry1;
25      end
26      2'b01 :
27      begin y <= result2;
28      overflow = carry2;
29      end
30      2'b10 :
31      begin y <= result3;
32      overflow = carry3;
33      end
34      2'b11 : y <= result4;
35  endcase
36  end
37
38  endmodule
```

Testing :

Various control signals along with inputs are given and the output is observed using GTKwave.

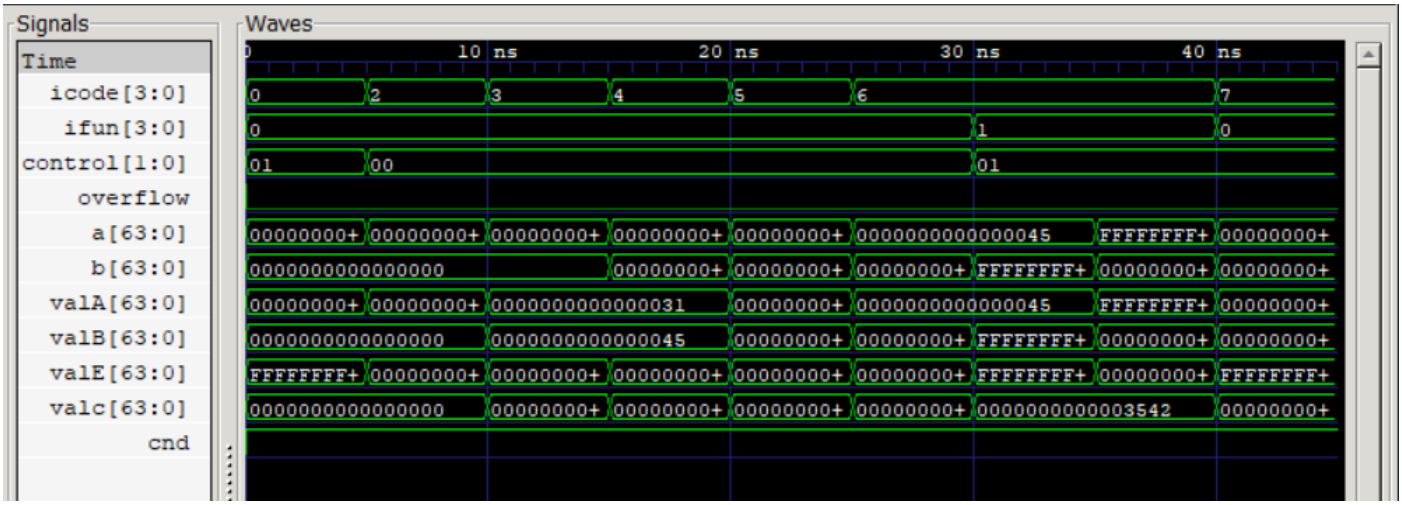
GTKwave plots:



Testing of the execute stage :

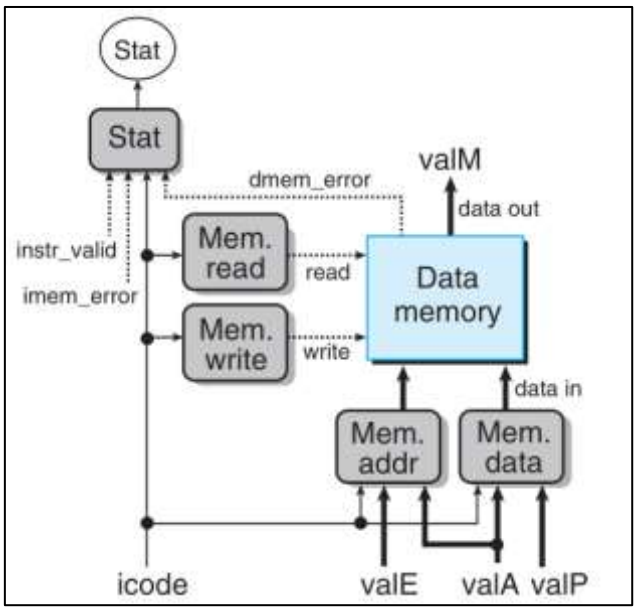
we can observe the output valE based on the inputs A, B and the control signal. We can observe that the condition codes are also being set.

```
icode = 0, ifun = 0
a = xxxxxxxxxxxxxxx, b = xxxxxxxxxxxxxxx, valE = xxxxxxxxxxxxxxx, control = 1
cnd = 1
of = 0, sf = 0, zf = 0
icode = 2, ifun = 0
a = 0000000000000008, b = 0000000000000000, valE = fffffffffffff8, control = 0
cnd = 1
of = 0, sf = 0, zf = 0
icode = 3, ifun = 0
a = 0000000000000006, b = 0000000000000000, valE = 0000000000000006, control = 0
cnd = 1
of = 0, sf = 0, zf = 0
icode = 4, ifun = 0
a = 0000000000007878, b = 0000000000000000, valE = 0000000000007878, control = 0
cnd = 1
of = 0, sf = 0, zf = 0
icode = 5, ifun = 0
a = 0000000000000011, b = 0000000000000045, valE = 0000000000000056, control = 0
cnd = 1
of = 0, sf = 0, zf = 0
icode = 6, ifun = 0
a = 0000000000000001, b = 0000000000000004, valE = 0000000000000005, control = 0
cnd = 1
of = 0, sf = 0, zf = 0
icode = 7, ifun = 0
a = fffffffffffffffb, b = 0000000000000045, valE = 00000000000004a, control = 1
cnd = 1
of = 0, sf = 1, zf = 1
```



- **MEMORY :**

The memory stage deals with reading from and writing to the main memory. When data is read, the value is referred to as valM.

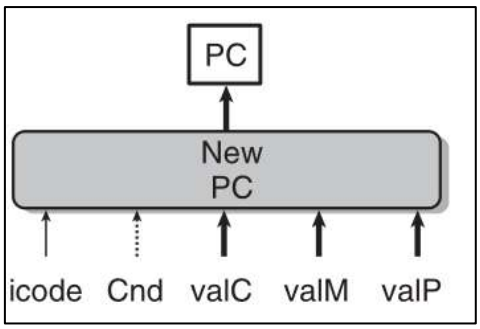


SEQ memory stage

The memory stage also deals with setting the status code.

- **PC UPDATE :**

This stage generates the new value of the program counter. This sets the value of the new PC to either valP, valC or valM depending on the type of instruction encountered.



SEQ PC update

Testing of PC update stage :

Inputs :

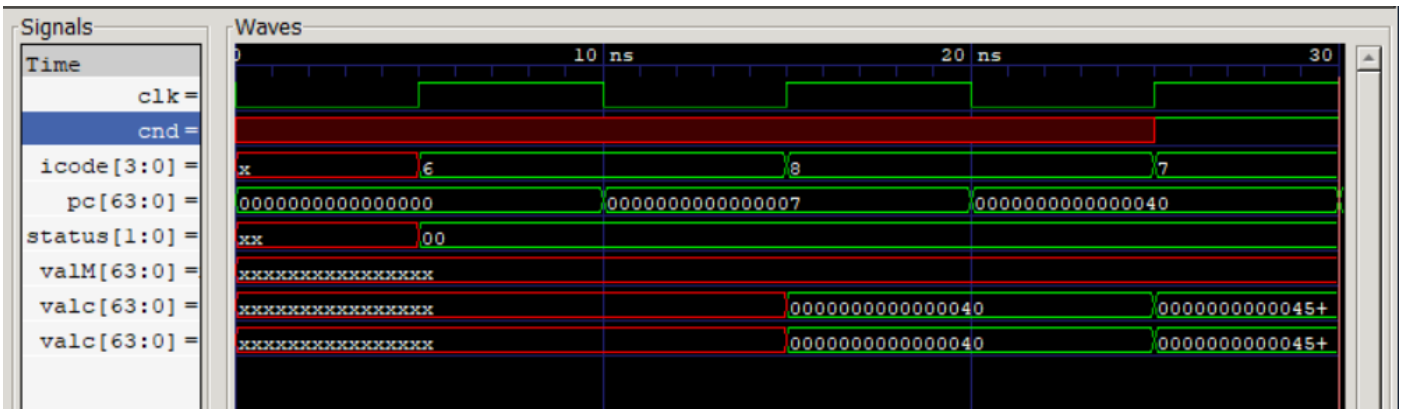
```
// opq
#5
icode = 4'h6 ;
valp = 64'h7 ;
status = 2'h0;
clk = ~clk ;
#5
$display("new pc = %h" , pc );
```

```
// call
clk = ~clk ;
#5
icode = 4'h8 ;
valc = 64'h40 ;
// valp = 64'h7 ;
status = 2'h0;
clk = ~clk ;
#5
$display("new pc = %h" , pc );


// conditional jmp
clk = ~clk ;
#5
icode = 4'h7 ;
valc = 64'h45677 ;
status = 2'h0;
cnd = 1 ;
valp = 64'd9 ;
clk = ~clk ;
#5
clk = ~clk ;
```

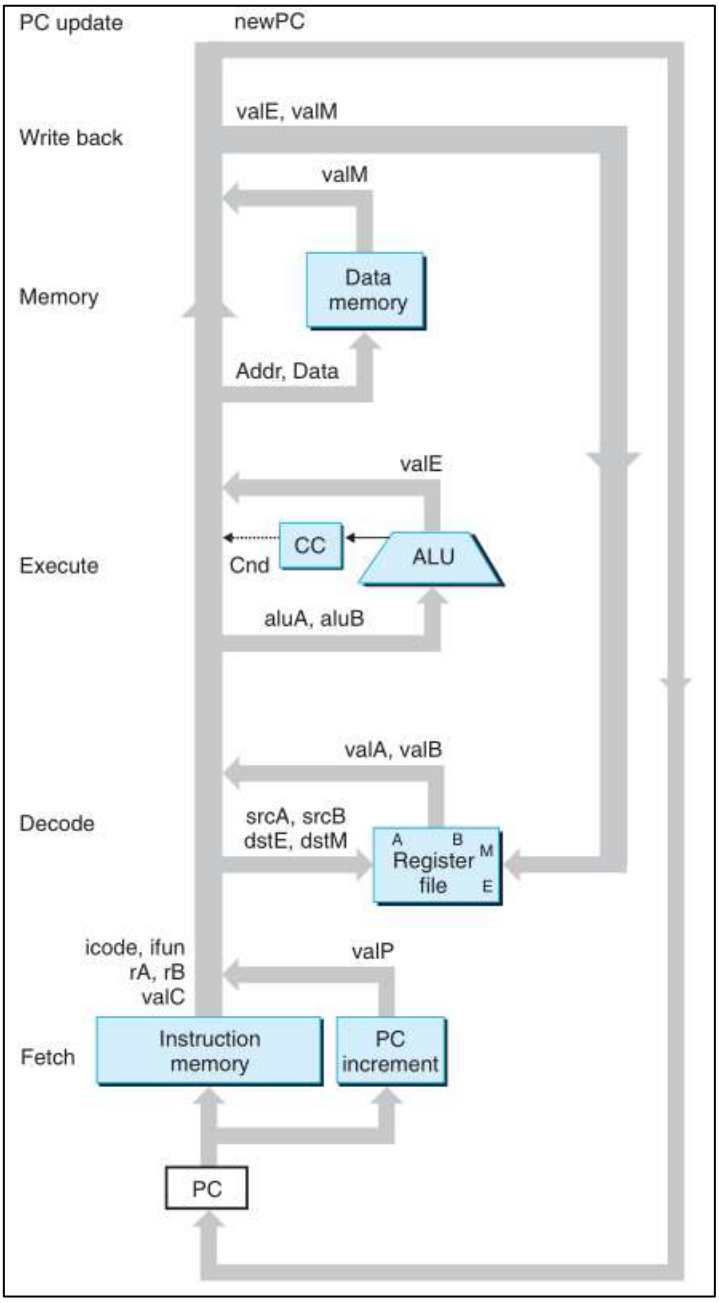
Outputs :

```
pc_old = xxxxxxxxxxxxxxx
new pc = 0000000000000000
pc_old = 0000000000000000
new pc = 0000000000000007
pc_old = 0000000000000007
new pc = 0000000000000040
pc_old = 0000000000000040
```



As we can observe our PC update block is working and is changing the program counter based on the instruction inputted.

FINAL SEQUENTIAL PROCESSOR :



Abstract view of SEQ processor

Code :

```
`include "fetch.v"
`include "decodewb.v"
`include "registerfile.v"
`include "execute.v"
`include "pcupdate.v"
`include "memory.v"

module processor(clk, status);  // the final processor/wrapper module.

input clk;
output reg [1:0] status;

wire signed [63:0] pc, valA, valB, valc, valp, valE, valM, memory_data;
wire [3:0] icode, ifun, rA, rB, dstE, dstM, srcA, srcB;
```

```
wire cnd,instr_valid, error;

// calling every module to perform similar function to that of a processor. this works as
a Y86 Sequential processor.

fetch module1
(.error(error),.instr_valid(instr_valid),.icode(icode),.ifun(ifun),.rA(rA),.rB(rB),.valc(v
alc),.valp(valp),.pc(pc),.clk(clk)); //fetching instructions

decodewb module2
(.dstE(dstE), .dstM(dstM), .srcA(srcA), .srcB(srcB), .cnd(cnd), .icode(icode), .rA(rA), .r
B(rB), .clk(clk)); // decoding the instructions that had been fetched

registerfile module3
(.icode(icode),.srcA(srcA), .srcB(srcB),.dstE(dstE),.dstM(dstM),.valA(valA),.valB(valB),.v
alM(valM),.valE(valE),.clk(clk));  // changing registers.

execute module4
(.valA(valA),.valB(valB),.valc(valc),.valE(valE), .icode(icode),.ifun(ifun), .cnd(cnd), .f
lags(flags)); // performing nescessary caculations using the ALU module.

memory module5
(.icode(icode), .valA(valA),.valB(valB),.valE(valE), .valp(valp), .clk(clk), .memory_data(
memory_data), .valM(valM)); // this deals with writing and reading our memory stack.

pcupdate module6
(.pc(pc), .clk(clk), .valp(valp),.valc(valc),.valM(valM),.status(status),.cnd(cnd),.icode(
icode)); // this block updates the program counter.

initial begin
    status = 0;
end

always@(posedge(clk))
begin
    // $display("icode = ",icode);
    // checking status condition and deciding on whether to halt the processor or not.
    if(icode == 2'h0)  // halt
    begin
        status = 2'h1;
    end

    else if (instr_valid == 2'h0)  // invalid_instruction
    begin
        status = 2'h2;
    end

    else if (error == 1)  // processor has run into an error.
    begin
        status = 2'h3;
    end

    else
    begin
```

```
        status = 0;
    end

    $display("icode = %h, ifun = %h, rA = %h, rB = %h\nvalA = %h, valB = %h, valc = %h,
valE = %h, valM = %h, valp = %h\npc = %h, instr_valid = %h, status = %h, cnd = %h\n of,
sf, zf = %b\n*****************\n",icode,ifun,rA , rB ,valA , valB , valc , valE ,
valM ,valp,pc , instr_valid , status , cnd , flags );


    if (status != 0) //stopping (halting) the process if status is not set to 0.
    begin
        $finish;
    end
end

endmodule
```

Testing the Sequential processor :

The following is the input sequence given to the processor :

```
    irmov 0x20 %rsp ;
    irmov 0x7 %rdx ;
    irmov %5 %r8 ;
    addq %rdx %r8 ;
    rmmov %r8 0x7(%rdx) ;
    rmmov %rdx (%r8) ;
    mrmov (%rax) %r8 ;
    nop ;
    call 0x61 ;
    halt;

    0x61 :  //call location
    irmov 0x7 %r12 ;
    irmov 0x5 %r13 ;
    addq %r12 %r13 ;
    ret ;
```

Output :

```
VCD info: dumpfile processor.vcd opened for output.
srcA = x, srcB = x
icode = 3, ifun = 0, rA = f, rB = 4
valA = xxxxxxxxxxxxxxxx, valB = xxxxxxxxxxxxxxxx, valc = 0000000000000020, valE = 000000
0000000020, valM = xxxxxxxxxxxxxxxx, valp = 000000000000000a
pc = 0000000000000000, instr_valid = 1, status = 0, cnd = 1
 of, sf, zf = 000
*******

clk = 0

WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
srcA = x, srcB = x
icode = 3, ifun = 0, rA = f, rB = 3
valA = xxxxxxxxxxxxxxxx, valB = xxxxxxxxxxxxxxxx, valc = 0000000000000007, valE = 000000
0000000007, valM = xxxxxxxxxxxxxxxx, valp = 0000000000000014
pc = 000000000000000a, instr_valid = 1, status = 0, cnd = 1
 of, sf, zf = 000
*******

clk = 0

WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
srcA = x, srcB = x
icode = 3, ifun = 0, rA = f, rB = 8
valA = xxxxxxxxxxxxxxxx, valB = xxxxxxxxxxxxxxxx, valc = 0000000000000005, valE = 000000
0000000005, valM = xxxxxxxxxxxxxxxx, valp = 000000000000001e
pc = 0000000000000014, instr_valid = 1, status = 0, cnd = 1
 of, sf, zf = 000
*******

clk = 0
```

```
WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
srcA = 3, srcB = 8
icode = 6, ifun = 0, rA = 3, rB = 8
valA = 0000000000000007, valB = 0000000000000005, valc = 0000000000000005, valE = 000000
000000000c, valM = xxxxxxxxxxxxxxxx, valp = 0000000000000020
pc = 000000000000001e, instr_valid = 1, status = 0, cnd = 1
 of, sf, zf = 000
*******

clk = 0

WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
srcA = 8, srcB = 3
icode = 4, ifun = 0, rA = 8, rB = 3
valA = 000000000000000c, valB = 0000000000000007, valc = 0000000000000007, valE = 000000
000000000e, valM = xxxxxxxxxxxxxxxx, valp = 000000000000002a
pc = 0000000000000020, instr_valid = 1, status = 0, cnd = 1
 of, sf, zf = 000
*******

clk = 0

srcA = 3, srcB = 8
icode = 4, ifun = 0, rA = 3, rB = 8
valA = 0000000000000007, valB = 000000000000000c, valc = 0000000000000000, valE = 000000
000000000c, valM = xxxxxxxxxxxxxxxx, valp = 0000000000000034
pc = 000000000000002a, instr_valid = 1, status = 0, cnd = 1
 of, sf, zf = 000
*******

clk = 0
```

```
WARNING: ./registerfile.v:60: $writememh: Standard inconsistency, following 1364-2005.
srcA = 3, srcB = 8
icode = 1, ifun = 0, rA = a, rB = 8
valA = 0000000000000007, valB = 000000000000000c, valc = 0000000000000000, valE = 000000
000000000c, valM = 0000000000000007, valp = 000000000000003f
pc = 000000000000003e, instr_valid = 1, status = 0, cnd = 1
 of, sf, zf = 000
*******

clk = 0

srcA = 3, srcB = 4
icode = 8, ifun = 0, rA = a, rB = 8
valA = 0000000000000007, valB = 0000000000000020, valc = 0000000000000061, valE = 000000
0000000018, valM = 0000000000000007, valp = 0000000000000048
pc = 000000000000003f, instr_valid = 1, status = 0, cnd = 1
 of, sf, zf = 000
*******

clk = 0

WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
srcA = 3, srcB = 4
icode = 3, ifun = 0, rA = f, rB = c
valA = 0000000000000007, valB = 0000000000000018, valc = 0000000000000007, valE = 000000
0000000007, valM = 0000000000000007, valp = 000000000000006b
pc = 0000000000000061, instr_valid = 1, status = 0, cnd = 1
 of, sf, zf = 000
*******

clk = 0
```

```
WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
srcA = 3, srcB = 4
icode = 3, ifun = 0, rA = f, rB = d
valA = 0000000000000007, valB = 0000000000000018, valc = 0000000000000005, valE = 000000
0000000005, valM = 0000000000000007, valp = 0000000000000075
pc = 000000000000006b, instr_valid = 1, status = 0, cnd = 1
 of, sf, zf = 000
*******

clk = 0

WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
srcA = c, srcB = d
icode = 6, ifun = 0, rA = c, rB = d
valA = 0000000000000007, valB = 0000000000000005, valc = 0000000000000005, valE = 000000
000000000c, valM = 0000000000000007, valp = 0000000000000077
pc = 0000000000000075, instr_valid = 1, status = 0, cnd = 1
 of, sf, zf = 000
*******

clk = 0

WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
srcA = 4, srcB = 4
icode = 9, ifun = 0, rA = c, rB = d
valA = 0000000000000018, valB = 0000000000000018, valc = 0000000000000005, valE = 000000
0000000020, valM = 0000000000000048, valp = 0000000000000078
pc = 0000000000000077, instr_valid = 1, status = 0, cnd = 1
 of, sf, zf = 000
*******

clk = 0
```

```
WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
srcA = 4, srcB = 4
icode = 0, ifun = 0, rA = c, rB = d
valA = 0000000000000020, valB = 0000000000000020, valc = 0000000000000005, valE = ffffff
fffffffff8, valM = 0000000000000048, valp = 0000000000000049
pc = 0000000000000048, instr_valid = 1, status = 1, cnd = 1
 of, sf, zf = 000
*******
```

Limitations / Drawbacks of Sequential Architecture :

- A sequential processor is way too slow.
- We have to wait for an instructions execution to be completed before fetching another, this not only wastes time but also resources as the other stages of a processor sit idle when the instruction is at one particular stage.

PIPELINED DESIGN :

Principle of pipelining : In a pipelined system, the task to be performed is divided

into a series of discrete stages. We break up the instructions into parts that can be executed by certain specified hardware parts of the processor. Each stage is independent in terms that is can process a part of an instruction while other stages are processing other instructions.

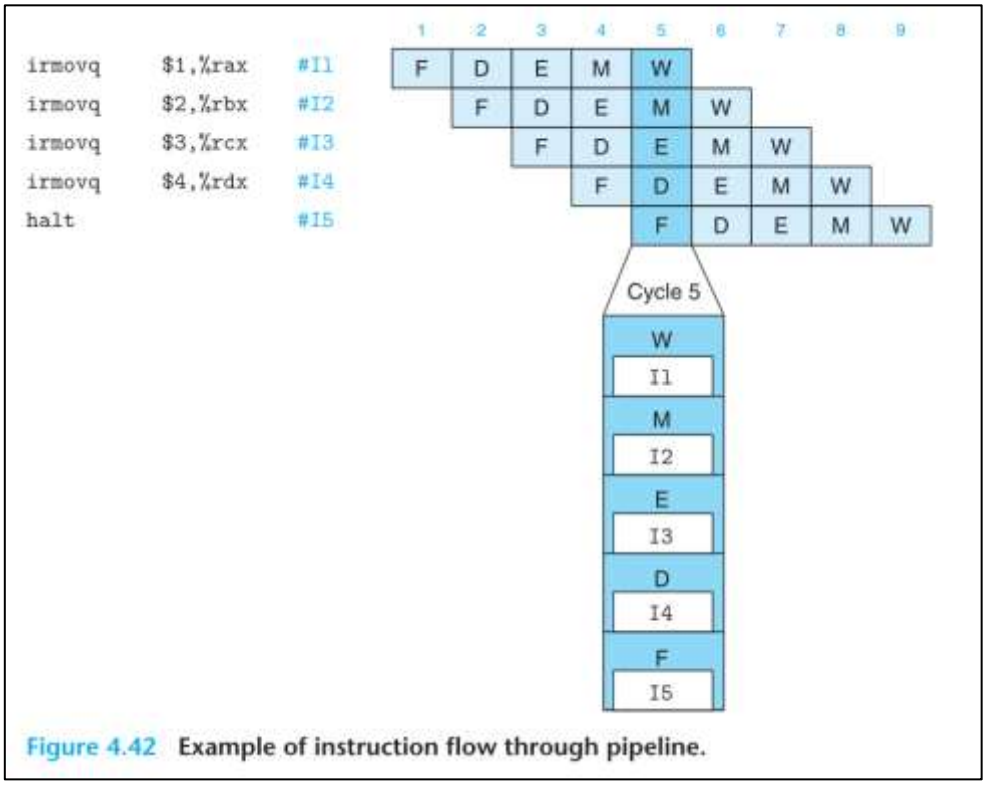
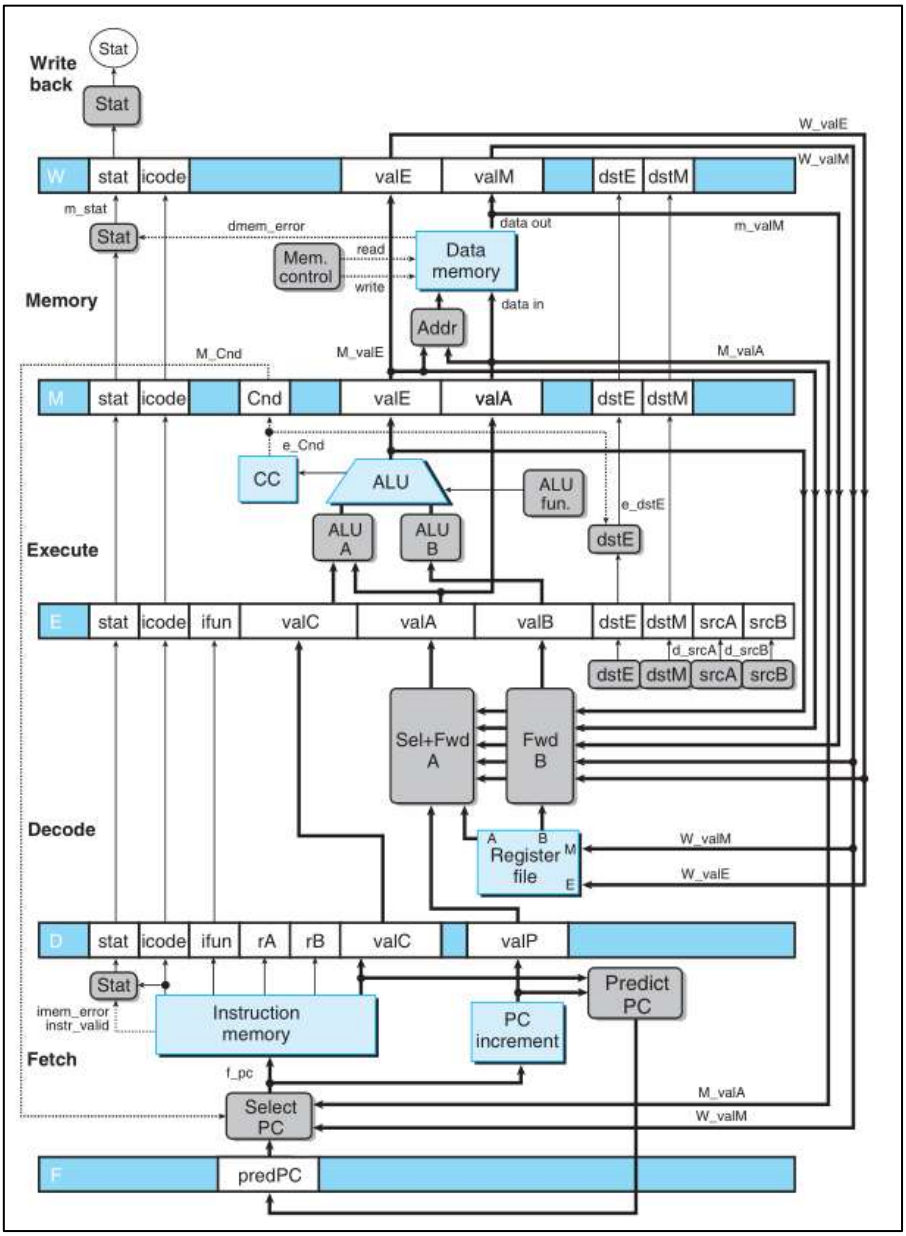


Figure 4.42 Example of instruction flow through pipeline.

PIPELINED ARCHITECTURE :



We insert registers known as pipeline registers which hold the outputs of each stage. We combine the fetch and the pc update blocks. This sits at the beginning of the 5 stages of our processor. We implement the following :

- Prediction of Program counter pointer
- Data forwarding
- Insertion of bubbles and stalling

1) PC Prediction :

We make predictions on where the instruction to process is present, generally we make pc = f_valP, but in cases where we read a constant value for jump and call instructions, we equate pc to f_valC. And if we encounter a ret instruction, we update pc to W_valM.

2) Data Forwarding :

Introducing pipelining into a system with feedback can lead to problems when there are dependencies between successive instructions. data dependencies are where the results computed by one instruction are used as the data for a following instruction. This can be solved by data forwarding. The technique of passing a result value directly from one pipeline stage to an earlier one known as data forwarding. This is done by letting our hardware access the stage registers present.

3) Insertion of bubbles and stalling :

One way to deal with data and control hazards is to stall the processor. This means do nothing for that clock cycle where the stage is stalled. This gives time for other stages to update and removes the data/control hazard. Or else when we need to remove a process mid stage we can use something called a bubble, inserting a bubble is equivalent to dynamically inserting a nop instruction. This also helps deal with hazards.

The following are the methods used to deal with particular hazards :

	Pipeline register				
Condition	F	D	E	M	W
Processing `ret`	stall	bubble	normal	normal	normal
Load/use hazard	stall	stall	bubble	normal	normal
Mispredicted branch	normal	bubble	bubble	normal	normal

Pipeline control logic

Y86 PIPELINED PROCESSOR CODE :

```
`include "fetch.v"
`include "predictPC.v"
`include "Freg.v"
`include "Dreg.v"
`include "decodewb.v"
`include "registerfile.v"
`include "selfwd.v"
`include "Ereg.v"
`include "execute.v"
`include "Mreg.v"
`include "memory.v"
`include "Wreg.v"
`include "control.v"
```

```
module processor(clk, M_stat);

input clk;
wire [63:0] f_PC;
output [1:0] M_stat;

wire [2:0] flags;

wire error,instr_valid, F_stall;
wire [3:0] f_icode, f_ifun, f_rA, f_rB;
wire [1:0] f_stat;
wire [63:0] f_valc, f_valp;
wire [63:0] F_PC,predictPC;

wire D_bubble, D_stall;
wire [1:0] D_stat;
wire [3:0] D_icode, D_ifun, D_rA, D_rB, d_dstE, d_dstM, d_srcA, d_srcB;
wire [63:0] D_valc, D_valp, d_rvalA, d_rvalB, d_valA, d_valB;

wire  E_bubble, e_cnd, set_cc;
wire [63:0] e_valE, E_valc, E_valA, E_valB;
wire [3:0] e_dstE, E_icode, E_ifun, E_dstE, E_dstM, E_srcA, E_srcB;
wire [1:0] E_stat;

wire M_cnd, M_bubble, dmem_error;
wire [1:0] W_stat, m_stat;
wire [3:0] M_icode, M_dstM, M_dstE;
wire [63:0] M_valA, m_valM, M_valE, memory_data;

wire W_stall;
wire [3:0] W_icode, W_dstE, W_dstM;
wire [63:0] W_valM, W_valE;

control module16
(.clk(clk), .D_icode(D_icode), .E_icode(E_icode), .M_icode(M_icode), .E_dstM(E_dstM), .d_s
rcA(d_srcA), .d_srcB(d_srcB), .e_cnd(e_cnd), .m_stat(m_stat), .W_stat(W_stat), .F_stall(F_
stall), .D_bubble(D_bubble), .D_stall(D_stall), .E_bubble(E_bubble), .M_bubble(M_bubble),
.W_stall(W_stall), .set_cc(set_cc));

Freg module5 (.clk(clk), .predictPC(predictPC), .F_PC(F_PC), .F_stall(F_stall));
fetch module1
(.error(error), .instr_valid(instr_valid), .icode(f_icode), .ifun(f_ifun), .rA(f_rA), .rB(
f_rB), .valc(f_valc), .valp(f_valp), .pc(f_PC), .clk(clk));
predictPC module2
(.clk(clk),.f_valp(f_valp), .f_valc(f_valc), .f_icode(f_icode), .predictPC(predictPC));
selectPC module3
(.clk(clk), .F_PC(F_PC), .M_icode(M_icode), .M_cnd(M_cnd), .M_valA(M_valA), .W_icode(W_ico
de), .W_valM(W_valM), .f_PC(f_PC));
Fstat module4
(.clk(clk),.instr_valid(instr_valid), .f_icode(f_icode), .error(error), .f_stat(f_stat));

Dreg module6
(.clk(clk), .f_stat(f_stat), .f_icode(f_icode), .f_ifun(f_ifun), .f_rA(f_rA), .f_rB(f_rB),
```

```
.f_valc(f_valc), .f_valp(f_valp), .D_stat(D_stat), .D_icode(D_icode), .D_ifun(D_ifun), .D_
rA(D_rA), .D_rB(D_rB), .D_valc(D_valc), .D_valp(D_valp), .D_bubble(D_bubble), .D_stall(D_s
tall));
decodewb module7
(.dstE(d_dstE), .dstM(d_dstM), .srcA(d_srcA), .srcB(d_srcB), .icode(D_icode), .ifun(D_ifun
), .rA(D_rA), .rB(D_rB), .clk(clk));
registerfile module8
(.icode(D_icode),.ifun(D_ifun),.srcA(d_srcA),.srcB(d_srcB),.dstE(W_dstE),.dstM(W_dstM),.va
lA(d_rvalA),.valB(d_rvalB),.valM(W_valM),.valE(W_valE),.clk(clk));
selfwdA module9
(.d_rvalA(d_rvalA), .D_valp(D_valp), .W_valE(W_valE), .W_valM(W_valM), .m_valM(m_valM), .e
_valE(e_valE), .d_srcA(d_srcA), .W_dstE(W_dstE), .W_dstM(W_dstM), .M_dstM(M_dstM), .M_dstE
(M_dstE), .e_dstE(e_dstE), .M_valE(M_valE), .D_icode(D_icode), .d_valA(d_valA));
fwdB module10
(.d_rvalB(d_rvalB), .W_valE(W_valE), .W_valM(W_valM), .m_valM(m_valM), .e_valE(e_valE), .d
_srcB(d_srcB), .W_dstE(W_dstE), .W_dstM(W_dstM), .M_dstM(M_dstM), .M_dstE(M_dstE), .e_dstE
(e_dstE), .M_valE(M_valE), .d_valB(d_valB));

Ereg module11
(.clk(clk), .D_stat(D_stat),.D_icode(D_icode),.D_ifun(D_ifun), .D_valc(D_valc), .d_valA(d_
valA), .d_valB(d_valB),.d_dstE(d_dstE), .d_dstM(d_dstM), .d_srcA(d_srcA), .d_srcB(d_srcB),
.E_stat(E_stat),.E_icode(E_icode),.E_ifun(E_ifun), .E_valc(E_valc), .E_valA(E_valA), .E_va
lB(E_valB),.E_dstE(E_dstE), .E_dstM(E_dstM), .E_srcA(E_srcA), .E_srcB(E_srcB), .E_bubble(E
_bubble));
execute moule12
(.clk(clk),.valA(E_valA), .valB(E_valB), .valc(E_valc), .valE(e_valE), .icode(E_icode), .i
fun(E_ifun), .cnd(e_cnd), .set_cc(set_cc), .dstE(E_dstE), .e_dstE(e_dstE), .flags(flags));

Mreg module13
(.clk(clk), .E_stat(E_stat), .E_icode(E_icode), .e_cnd(e_cnd), .e_valE(e_valE), .E_valA(E_
valA), .e_dstE(e_dstE), .E_dstM(E_dstM), .M_stat(M_stat), .M_icode(M_icode), .M_cnd(M_cnd)
, .M_valE(M_valE), .M_valA(M_valA), .M_dstE(M_dstE), .M_dstM(M_dstM), .M_bubble(M_bubble))
;
memory module14
(.M_stat(M_stat), .m_stat(m_stat), .dmem_error(dmem_error), .icode(M_icode), .valA(M_valA)
, .valE(M_valE), .clk(clk), .memory_data(memory_data), .valM(m_valM));

Wreg module15
(.clk(clk), .m_stat(m_stat), .M_icode(M_icode), .M_valE(M_valE), .m_valM(m_valM), .M_dstE(
M_dstE), .M_dstM(M_dstM), .W_stat(W_stat), .W_icode(W_icode), .W_valE(W_valE), .W_valM(W_v
alM), .W_dstE(W_dstE), .W_dstM(W_dstM), .W_stall(W_stall));



always@(posedge(clk))
begin
    $display("F_PC = %h\nD_stat = %h, D_icode = %h, D_ifun = %h, D_rA = %h, D_rB = %h,
D_valc = %h, D_valp = %h\ne_valE = %h,E_stat = %h, E_icode = %h, E_valc = %h, E_valA = %h,
E_valB = %h, E_dstE = %h, E_dstM = %h, EsrcA = %h, E_srcB = %h\nM_stat = %h, M_icode = %h,
M_cnd = %h, M_valE = %h, M_valA = %h, M_dstE = %h, M_dstM = %h\nW_stat = %h, W_icode = %h,
W_valE = %h, W_valM = %h, W_dstE = %h, W_dstM = %h",F_PC, D_stat, D_icode, D_ifun, D_rA,
D_rB, D_valc, D_valp,e_valE,E_stat, E_icode, E_valc, E_valA, E_valB, E_dstE, E_dstM,
E_srcA, E_srcB,M_stat, M_icode, M_cnd, M_valE, M_valA, M_dstE, M_dstM,W_stat, W_icode,
W_valE, W_valM, W_dstE, W_dstM);
```

```
    if (W_stat != 2'b00)
    begin
        $finish;
    end
end

endmodule
```

Testing the Pipelined Processor :

The following is the input sequence given to the processor :

```
irmov 0x20 %rsp ;
irmov 0x7 %rdx ;
irmov %5 %r8 ;
addq %rdx %r8 ;
rmmov %r8 0x7(%rdx) ;
rmmov %rdx (%r8) ;
mrmov (%rax) %r8 ;
nop ;
call 0x61 ;
halt;

0x61 :  //call location
irmov 0x7 %r12 ;
irmov 0x5 %r13 ;
addq %r12 %r13 ;
ret ;
```

Initial register status :

reg_pre - Notepad

File Edit Format View Help

```
// 0x00000000
0000000000000001
0000000000000002
0000000000000003
0000000000000004
0000000000000005
0000000000000006
0000000000000007
0000000000000008
0000000000000009
000000000000000a
000000000000000b
000000000000000c
000000000000000d
000000000000000e
000000000000000f
```

Output :

C Copy Caption

```
VCD info: dumpfile processor.vcd opened for output.
F_PC = 0000000000000000
D_stat = x, D_icode = x, D_ifun = x, D_rA = x, D_rB = x, D_valc = xxxxxxxxxxxxxxxx, D_va
lp = xxxxxxxxxxxxxxxx
e_valE = xxxxxxxxxxxxxxxx,E_stat = x, E_icode = x, E_valc = xxxxxxxxxxxxxxxx, E_valA = x
xxxxxxxxxxxxxxx, E_valB = xxxxxxxxxxxxxxxx, E_dstE = x, E_dstM = x, EsrcA = x, E_srcB =
x
M_stat = x, M_icode = x, M_cnd = x, M_valE = xxxxxxxxxxxxxxxx, M_valA = xxxxxxxxxxxxxxx
x, M_dstE = x, M_dstM = x
W_stat = x, W_icode = x, W_valE = xxxxxxxxxxxxxxxx, W_valM = xxxxxxxxxxxxxxxx, W_dstE =
x, W_dstM = x
clk = 0

F_PC = 000000000000000a
D_stat = 0, D_icode = 3, D_ifun = 0, D_rA = f, D_rB = 4, D_valc = 0000000000000020, D_va
lp = 000000000000000a
e_valE = xxxxxxxxxxxxxxxx,E_stat = x, E_icode = x, E_valc = xxxxxxxxxxxxxxxx, E_valA = x
xxxxxxxxxxxxxxx, E_valB = xxxxxxxxxxxxxxxx, E_dstE = x, E_dstM = x, EsrcA = x, E_srcB =
x
M_stat = x, M_icode = x, M_cnd = 1, M_valE = xxxxxxxxxxxxxxxx, M_valA = xxxxxxxxxxxxxxx
x, M_dstE = x, M_dstM = x
W_stat = x, W_icode = x, W_valE = xxxxxxxxxxxxxxxx, W_valM = xxxxxxxxxxxxxxxx, W_dstE =
x, W_dstM = x
clk = 0

F_PC = 000000000000000a
D_stat = 0, D_icode = 3, D_ifun = 0, D_rA = f, D_rB = 4, D_valc = 0000000000000020, D_va
lp = 000000000000000a
e_valE = 0000000000000020,E_stat = 0, E_icode = 3, E_valc = 0000000000000020, E_valA = x
xxxxxxxxxxxxxxx, E_valB = xxxxxxxxxxxxxxxx, E_dstE = 4, E_dstM = f, EsrcA = x, E_srcB =
```

```
x
M_stat = x, M_icode = x, M_cnd = 1, M_valE = xxxxxxxxxxxxxxx, M_valA = xxxxxxxxxxxxxxx
x, M_dstE = x, M_dstM = x
W_stat = x, W_icode = x, W_valE = xxxxxxxxxxxxxxx, W_valM = xxxxxxxxxxxxxxx, W_dstE =
x, W_dstM = x
clk = 0

F_PC = 0000000000000014
D_stat = 0, D_icode = 3, D_ifun = 0, D_rA = f, D_rB = 3, D_valc = 0000000000000007, D_va
lp = 0000000000000014
e_valE = 0000000000000020,E_stat = 0, E_icode = 3, E_valc = 0000000000000020, E_valA = x
xxxxxxxxxxxxxx, E_valB = xxxxxxxxxxxxxxx, E_dstE = 4, E_dstM = f, EsrcA = x, E_srcB =
x
M_stat = 0, M_icode = 3, M_cnd = 1, M_valE = 0000000000000020, M_valA = xxxxxxxxxxxxxx
x, M_dstE = 4, M_dstM = f
W_stat = x, W_icode = x, W_valE = xxxxxxxxxxxxxxx, W_valM = xxxxxxxxxxxxxxx, W_dstE =
x, W_dstM = x
clk = 0

WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 0000000000000014
D_stat = 0, D_icode = 3, D_ifun = 0, D_rA = f, D_rB = 3, D_valc = 0000000000000007, D_va
lp = 0000000000000014
e_valE = 0000000000000007,E_stat = 0, E_icode = 3, E_valc = 0000000000000007, E_valA = x
xxxxxxxxxxxxxx, E_valB = xxxxxxxxxxxxxxx, E_dstE = 3, E_dstM = f, EsrcA = x, E_srcB =
x
M_stat = 0, M_icode = 3, M_cnd = 1, M_valE = 0000000000000020, M_valA = xxxxxxxxxxxxxx
x, M_dstE = 4, M_dstM = f
W_stat = 0, W_icode = 3, W_valE = 0000000000000020, W_valM = xxxxxxxxxxxxxxx, W_dstE =
4, W_dstM = f
clk = 0
```

```
WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 000000000000001e
D_stat = 0, D_icode = 3, D_ifun = 0, D_rA = f, D_rB = 8, D_valc = 0000000000000005, D_va
lp = 000000000000001e
e_valE = 0000000000000007,E_stat = 0, E_icode = 3, E_valc = 0000000000000007, E_valA = x
xxxxxxxxxxxxxx, E_valB = xxxxxxxxxxxxxxx, E_dstE = 3, E_dstM = f, EsrcA = x, E_srcB =
x
M_stat = 0, M_icode = 3, M_cnd = 1, M_valE = 0000000000000007, M_valA = xxxxxxxxxxxxxx
x, M_dstE = 3, M_dstM = f
W_stat = 0, W_icode = 3, W_valE = 0000000000000020, W_valM = xxxxxxxxxxxxxxx, W_dstE =
4, W_dstM = f
clk = 0

WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 000000000000001e
D_stat = 0, D_icode = 3, D_ifun = 0, D_rA = f, D_rB = 8, D_valc = 0000000000000005, D_va
lp = 000000000000001e
e_valE = 0000000000000005,E_stat = 0, E_icode = 3, E_valc = 0000000000000005, E_valA = x
xxxxxxxxxxxxxx, E_valB = xxxxxxxxxxxxxxx, E_dstE = 8, E_dstM = f, EsrcA = x, E_srcB =
x
M_stat = 0, M_icode = 3, M_cnd = 1, M_valE = 0000000000000007, M_valA = xxxxxxxxxxxxxx
x, M_dstE = 3, M_dstM = f
W_stat = 0, W_icode = 3, W_valE = 0000000000000007, W_valM = xxxxxxxxxxxxxxx, W_dstE =
3, W_dstM = f
clk = 0

WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 0000000000000020
D_stat = 0, D_icode = 6, D_ifun = 0, D_rA = 3, D_rB = 8, D_valc = 0000000000000005, D_va
lp = 0000000000000020
e_valE = 0000000000000005,E_stat = 0, E_icode = 3, E_valc = 0000000000000005, E_valA = x
xxxxxxxxxxxxxx, E_valB = xxxxxxxxxxxxxxx, E_dstE = 8, E_dstM = f, EsrcA = x, E_srcB =
```

```
x
M_stat = 0, M_icode = 3, M_cnd = 1, M_valE = 0000000000000005, M_valA = xxxxxxxxxxxxxxx
x, M_dstE = 8, M_dstM = f
W_stat = 0, W_icode = 3, W_valE = 0000000000000007, W_valM = xxxxxxxxxxxxxxxx, W_dstE =
3, W_dstM = f
clk = 0

WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 0000000000000020
D_stat = 0, D_icode = 6, D_ifun = 0, D_rA = 3, D_rB = 8, D_valc = 0000000000000005, D_va
lp = 0000000000000020
e_valE = 000000000000000c,E_stat = 0, E_icode = 6, E_valc = 0000000000000005, E_valA = 0
000000000000007, E_valB = 0000000000000005, E_dstE = 8, E_dstM = f, EsrcA = 3, E_srcB =
8
M_stat = 0, M_icode = 3, M_cnd = 1, M_valE = 0000000000000005, M_valA = xxxxxxxxxxxxxxx
x, M_dstE = 8, M_dstM = f
W_stat = 0, W_icode = 3, W_valE = 0000000000000005, W_valM = xxxxxxxxxxxxxxxx, W_dstE =
8, W_dstM = f
clk = 0

WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 000000000000002a
D_stat = 0, D_icode = 4, D_ifun = 0, D_rA = 8, D_rB = 3, D_valc = 0000000000000007, D_va
lp = 000000000000002a
e_valE = 000000000000000c,E_stat = 0, E_icode = 6, E_valc = 0000000000000005, E_valA = 0
000000000000007, E_valB = 000000000000000c, E_dstE = 8, E_dstM = f, EsrcA = 3, E_srcB =
8
M_stat = 0, M_icode = 6, M_cnd = 1, M_valE = 000000000000000c, M_valA = 000000000000000
7, M_dstE = 8, M_dstM = f
W_stat = 0, W_icode = 3, W_valE = 0000000000000005, W_valM = xxxxxxxxxxxxxxxx, W_dstE =
8, W_dstM = f
clk = 0
```

```
WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 000000000000002a
D_stat = 0, D_icode = 4, D_ifun = 0, D_rA = 8, D_rB = 3, D_valc = 0000000000000007, D_va
lp = 000000000000002a
e_valE = 000000000000000e,E_stat = 0, E_icode = 4, E_valc = 0000000000000007, E_valA = 0
000000000000c, E_valB = 0000000000000007, E_dstE = f, E_dstM = f, EsrcA = 8, E_srcB =
3
M_stat = 0, M_icode = 6, M_cnd = 1, M_valE = 000000000000000c, M_valA = 000000000000000
7, M_dstE = 8, M_dstM = f
W_stat = 0, W_icode = 6, W_valE = 000000000000000c, W_valM = xxxxxxxxxxxxxxxx, W_dstE =
8, W_dstM = f
clk = 0

WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 0000000000000034
D_stat = 0, D_icode = 4, D_ifun = 0, D_rA = 3, D_rB = 8, D_valc = 0000000000000000, D_va
lp = 0000000000000034
e_valE = 000000000000000e,E_stat = 0, E_icode = 4, E_valc = 0000000000000007, E_valA = 0
000000000000c, E_valB = 0000000000000007, E_dstE = f, E_dstM = f, EsrcA = 8, E_srcB =
3
M_stat = 0, M_icode = 4, M_cnd = 1, M_valE = 000000000000000e, M_valA = 000000000000000
c, M_dstE = f, M_dstM = f
W_stat = 0, W_icode = 6, W_valE = 000000000000000c, W_valM = xxxxxxxxxxxxxxxx, W_dstE =
8, W_dstM = f
clk = 0

F_PC = 0000000000000034
D_stat = 0, D_icode = 4, D_ifun = 0, D_rA = 3, D_rB = 8, D_valc = 0000000000000000, D_va
lp = 0000000000000034
e_valE = 000000000000000c,E_stat = 0, E_icode = 4, E_valc = 0000000000000000, E_valA = 0
000000000000007, E_valB = 000000000000000c, E_dstE = f, E_dstM = f, EsrcA = 3, E_srcB =
8
```

```
M_stat = 0, M_icode = 4, M_cnd = 1, M_valE = 000000000000000e, M_valA = 000000000000000
c, M_dstE = f, M_dstM = f
W_stat = 0, W_icode = 4, W_valE = 000000000000000e, W_valM = xxxxxxxxxxxxxxxx, W_dstE =
f, W_dstM = f
clk = 0

F_PC = 000000000000003e
D_stat = 0, D_icode = 5, D_ifun = 0, D_rA = a, D_rB = 8, D_valc = 0000000000000000, D_va
lp = 000000000000003e
e_valE = 000000000000000c,E_stat = 0, E_icode = 4, E_valc = 0000000000000000, E_valA = 0
000000000000007, E_valB = 000000000000000c, E_dstE = f, E_dstM = f, EsrcA = 3, E_srcB =
8
M_stat = 0, M_icode = 4, M_cnd = 1, M_valE = 000000000000000c, M_valA = 000000000000000
7, M_dstE = f, M_dstM = f
W_stat = 0, W_icode = 4, W_valE = 000000000000000e, W_valM = xxxxxxxxxxxxxxxx, W_dstE =
f, W_dstM = f
clk = 0

F_PC = 000000000000003e
D_stat = 0, D_icode = 5, D_ifun = 0, D_rA = a, D_rB = 8, D_valc = 0000000000000000, D_va
lp = 000000000000003e
e_valE = 000000000000000c,E_stat = 0, E_icode = 5, E_valc = 0000000000000000, E_valA = 0
000000000000007, E_valB = 000000000000000c, E_dstE = f, E_dstM = a, EsrcA = 3, E_srcB =
8
M_stat = 0, M_icode = 4, M_cnd = 1, M_valE = 000000000000000c, M_valA = 000000000000000
7, M_dstE = f, M_dstM = f
W_stat = 0, W_icode = 4, W_valE = 000000000000000c, W_valM = xxxxxxxxxxxxxxxx, W_dstE =
f, W_dstM = f
clk = 0

F_PC = 000000000000003f
D_stat = 0, D_icode = 1, D_ifun = 0, D_rA = a, D_rB = 8, D_valc = 0000000000000000, D_va
lp = 000000000000003f
```

```
e_valE = 000000000000000c,E_stat = 0, E_icode = 5, E_valc = 0000000000000000, E_valA = 0
000000000000007, E_valB = 000000000000000c, E_dstE = f, E_dstM = a, EsrcA = 3, E_srcB =
8
M_stat = 0, M_icode = 5, M_cnd = 1, M_valE = 000000000000000c, M_valA = 000000000000000
7, M_dstE = f, M_dstM = a
W_stat = 0, W_icode = 4, W_valE = 000000000000000c, W_valM = xxxxxxxxxxxxxxxx, W_dstE =
f, W_dstM = f
clk = 0

WARNING: ./registerfile.v:60: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 000000000000003f
D_stat = 0, D_icode = 1, D_ifun = 0, D_rA = a, D_rB = 8, D_valc = 0000000000000000, D_va
lp = 000000000000003f
e_valE = 000000000000000c,E_stat = 0, E_icode = 1, E_valc = 0000000000000000, E_valA = 0
000000000000007, E_valB = 000000000000000c, E_dstE = f, E_dstM = f, EsrcA = 3, E_srcB =
8
M_stat = 0, M_icode = 5, M_cnd = 1, M_valE = 000000000000000c, M_valA = 000000000000000
7, M_dstE = f, M_dstM = a
W_stat = 0, W_icode = 5, W_valE = 000000000000000c, W_valM = 0000000000000007, W_dstE =
f, W_dstM = a
clk = 0

WARNING: ./registerfile.v:60: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 0000000000000061
D_stat = 0, D_icode = 8, D_ifun = 0, D_rA = a, D_rB = 8, D_valc = 0000000000000061, D_va
lp = 0000000000000048
e_valE = 000000000000000c,E_stat = 0, E_icode = 1, E_valc = 0000000000000000, E_valA = 0
000000000000007, E_valB = 000000000000000c, E_dstE = f, E_dstM = f, EsrcA = 3, E_srcB =
8
M_stat = 0, M_icode = 1, M_cnd = 1, M_valE = 000000000000000c, M_valA = 000000000000000
7, M_dstE = f, M_dstM = f
W_stat = 0, W_icode = 5, W_valE = 000000000000000c, W_valM = 0000000000000007, W_dstE =
f, W_dstM = a
```

```
clk = 0

F_PC = 0000000000000061
D_stat = 0, D_icode = 8, D_ifun = 0, D_rA = a, D_rB = 8, D_valc = 0000000000000061, D_va
lp = 0000000000000048
e_valE = 0000000000000018,E_stat = 0, E_icode = 8, E_valc = 0000000000000061, E_valA = 0
000000000000048, E_valB = 0000000000000020, E_dstE = 4, E_dstM = f, EsrcA = 3, E_srcB =
4
M_stat = 0, M_icode = 1, M_cnd = 1, M_valE = 000000000000000c, M_valA = 000000000000000
7, M_dstE = f, M_dstM = f
W_stat = 0, W_icode = 1, W_valE = 000000000000000c, W_valM = 0000000000000007, W_dstE =
f, W_dstM = f
clk = 0

F_PC = 000000000000006b
D_stat = 0, D_icode = 3, D_ifun = 0, D_rA = f, D_rB = c, D_valc = 0000000000000007, D_va
lp = 000000000000006b
e_valE = 0000000000000018,E_stat = 0, E_icode = 8, E_valc = 0000000000000061, E_valA = 0
000000000000048, E_valB = 0000000000000018, E_dstE = 4, E_dstM = f, EsrcA = 3, E_srcB =
4
M_stat = 0, M_icode = 8, M_cnd = 1, M_valE = 0000000000000018, M_valA = 000000000000004
8, M_dstE = 4, M_dstM = f
W_stat = 0, W_icode = 1, W_valE = 000000000000000c, W_valM = 0000000000000007, W_dstE =
f, W_dstM = f
clk = 0

WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 000000000000006b
D_stat = 0, D_icode = 3, D_ifun = 0, D_rA = f, D_rB = c, D_valc = 0000000000000007, D_va
lp = 000000000000006b
e_valE = 0000000000000007,E_stat = 0, E_icode = 3, E_valc = 0000000000000007, E_valA = 0
000000000000007, E_valB = 0000000000000018, E_dstE = c, E_dstM = f, EsrcA = 3, E_srcB =
4
M_stat = 0, M_icode = 8, M_cnd = 1, M_valE = 0000000000000018, M_valA = 000000000000004
8, M_dstE = 4, M_dstM = f
W_stat = 0, W_icode = 8, W_valE = 0000000000000018, W_valM = 0000000000000007, W_dstE =
4, W_dstM = f
clk = 0

WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 0000000000000075
D_stat = 0, D_icode = 3, D_ifun = 0, D_rA = f, D_rB = d, D_valc = 0000000000000005, D_va
lp = 0000000000000075
e_valE = 0000000000000007,E_stat = 0, E_icode = 3, E_valc = 0000000000000007, E_valA = 0
000000000000007, E_valB = 0000000000000018, E_dstE = c, E_dstM = f, EsrcA = 3, E_srcB =
4
M_stat = 0, M_icode = 3, M_cnd = 1, M_valE = 0000000000000007, M_valA = 000000000000000
7, M_dstE = c, M_dstM = f
W_stat = 0, W_icode = 8, W_valE = 0000000000000018, W_valM = 0000000000000007, W_dstE =
4, W_dstM = f
clk = 0

WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 0000000000000075
D_stat = 0, D_icode = 3, D_ifun = 0, D_rA = f, D_rB = d, D_valc = 0000000000000005, D_va
lp = 0000000000000075
e_valE = 0000000000000005,E_stat = 0, E_icode = 3, E_valc = 0000000000000005, E_valA = 0
000000000000007, E_valB = 0000000000000018, E_dstE = d, E_dstM = f, EsrcA = 3, E_srcB =
4
M_stat = 0, M_icode = 3, M_cnd = 1, M_valE = 0000000000000007, M_valA = 000000000000000
7, M_dstE = c, M_dstM = f
W_stat = 0, W_icode = 3, W_valE = 0000000000000007, W_valM = 0000000000000007, W_dstE =
c, W_dstM = f
clk = 0
```

```
WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 0000000000000077
D_stat = 0, D_icode = 6, D_ifun = 0, D_rA = c, D_rB = d, D_valc = 0000000000000005, D_va
lp = 0000000000000077
e_valE = 0000000000000005,E_stat = 0, E_icode = 3, E_valc = 0000000000000005, E_valA = 0
000000000000007, E_valB = 0000000000000020, E_dstE = d, E_dstM = f, EsrcA = 3, E_srcB =
4
M_stat = 0, M_icode = 3, M_cnd = 1, M_valE = 0000000000000005, M_valA = 000000000000000
7, M_dstE = d, M_dstM = f
W_stat = 0, W_icode = 3, W_valE = 0000000000000007, W_valM = 0000000000000007, W_dstE =
c, W_dstM = f
clk = 0

WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 0000000000000077
D_stat = 0, D_icode = 6, D_ifun = 0, D_rA = c, D_rB = d, D_valc = 0000000000000005, D_va
lp = 0000000000000077
e_valE = 0000000000000005,E_stat = 0, E_icode = 6, E_valc = 0000000000000005, E_valA = 0
000000000000007, E_valB = 0000000000000005, E_dstE = d, E_dstM = f, EsrcA = c, E_srcB =
d
M_stat = 0, M_icode = 3, M_cnd = 1, M_valE = 0000000000000005, M_valA = 000000000000000
7, M_dstE = d, M_dstM = f
W_stat = 0, W_icode = 3, W_valE = 0000000000000005, W_valM = 0000000000000007, W_dstE =
d, W_dstM = f
clk = 0
```

```
WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 0000000000000078
D_stat = 0, D_icode = 9, D_ifun = 0, D_rA = c, D_rB = d, D_valc = 0000000000000005, D_va
lp = 0000000000000078
e_valE = 0000000000000005,E_stat = 0, E_icode = 6, E_valc = 0000000000000005, E_valA = 0
000000000000007, E_valB = 0000000000000005, E_dstE = d, E_dstM = f, EsrcA = c, E_srcB =
d
M_stat = 0, M_icode = 6, M_cnd = 1, M_valE = 0000000000000005, M_valA = 000000000000000
7, M_dstE = d, M_dstM = f
W_stat = 0, W_icode = 3, W_valE = 0000000000000005, W_valM = 0000000000000007, W_dstE =
d, W_dstM = f
clk = 0

WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 0000000000000078
D_stat = 0, D_icode = 1, D_ifun = 0, D_rA = c, D_rB = d, D_valc = 0000000000000005, D_va
lp = 0000000000000078
e_valE = 0000000000000020,E_stat = 0, E_icode = 9, E_valc = 0000000000000005, E_valA = 0
000000000000018, E_valB = 0000000000000018, E_dstE = 4, E_dstM = f, EsrcA = 4, E_srcB =
4
M_stat = 0, M_icode = 6, M_cnd = 1, M_valE = 0000000000000005, M_valA = 000000000000000
7, M_dstE = d, M_dstM = f
W_stat = 0, W_icode = 6, W_valE = 0000000000000005, W_valM = 0000000000000007, W_dstE =
d, W_dstM = f
clk = 0

WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 0000000000000078
D_stat = 0, D_icode = 1, D_ifun = 0, D_rA = c, D_rB = d, D_valc = 0000000000000005, D_va
lp = 0000000000000078
e_valE = fffffffffffffff8,E_stat = 0, E_icode = 1, E_valc = 0000000000000005, E_valA = 0
000000000000020, E_valB = 0000000000000020, E_dstE = f, E_dstM = f, EsrcA = 4, E_srcB =
```

```
M_stat = 0, M_icode = 9, M_cnd = 1, M_valE = 0000000000000020, M_valA = 000000000000001
8, M_dstE = 4, M_dstM = f
W_stat = 0, W_icode = 6, W_valE = 0000000000000005, W_valM = 0000000000000007, W_dstE =
d, W_dstM = f
clk = 0

WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 0000000000000078
D_stat = 0, D_icode = 1, D_ifun = 0, D_rA = c, D_rB = d, D_valc = 0000000000000005, D_va
lp = 0000000000000078
e_valE = fffffffffffffff8,E_stat = 0, E_icode = 1, E_valc = 0000000000000005, E_valA = 0
000000000000020, E_valB = 0000000000000020, E_dstE = f, E_dstM = f, EsrcA = 4, E_srcB =
4
M_stat = 0, M_icode = 1, M_cnd = 1, M_valE = fffffffffffffff8, M_valA = 000000000000002
0, M_dstE = f, M_dstM = f
W_stat = 0, W_icode = 9, W_valE = 0000000000000020, W_valM = 0000000000000048, W_dstE =
4, W_dstM = f
clk = 0

F_PC = 0000000000000078
D_stat = 0, D_icode = 1, D_ifun = 0, D_rA = c, D_rB = d, D_valc = 0000000000000005, D_va
lp = 0000000000000078
e_valE = fffffffffffffff8,E_stat = 0, E_icode = 1, E_valc = 0000000000000005, E_valA = 0
000000000000020, E_valB = 0000000000000020, E_dstE = f, E_dstM = f, EsrcA = 4, E_srcB =
4
M_stat = 0, M_icode = 1, M_cnd = 1, M_valE = fffffffffffffff8, M_valA = 000000000000002
0, M_dstE = f, M_dstM = f
W_stat = 0, W_icode = 1, W_valE = fffffffffffffff8, W_valM = 0000000000000048, W_dstE =
f, W_dstM = f
clk = 0

F_PC = 0000000000000079
D_stat = 1, D_icode = 0, D_ifun = 0, D_rA = c, D_rB = d, D_valc = 0000000000000005, D_va
lp = 0000000000000049
e_valE = fffffffffffffff8,E_stat = 0, E_icode = 1, E_valc = 0000000000000005, E_valA = 0
000000000000018, E_valB = 0000000000000018, E_dstE = f, E_dstM = f, EsrcA = 4, E_srcB =
4
M_stat = 0, M_icode = 1, M_cnd = 1, M_valE = fffffffffffffff8, M_valA = 000000000000002
0, M_dstE = f, M_dstM = f
W_stat = 0, W_icode = 1, W_valE = fffffffffffffff8, W_valM = 0000000000000048, W_dstE =
f, W_dstM = f
clk = 0

F_PC = 0000000000000079
D_stat = 0, D_icode = 1, D_ifun = 0, D_rA = c, D_rB = d, D_valc = 0000000000000005, D_va
lp = 0000000000000079
e_valE = fffffffffffffff8,E_stat = 1, E_icode = 0, E_valc = 0000000000000005, E_valA = 0
000000000000020, E_valB = 0000000000000020, E_dstE = f, E_dstM = f, EsrcA = 4, E_srcB =
4
M_stat = 0, M_icode = 1, M_cnd = 1, M_valE = fffffffffffffff8, M_valA = 000000000000001
8, M_dstE = f, M_dstM = f
W_stat = 0, W_icode = 1, W_valE = fffffffffffffff8, W_valM = 0000000000000048, W_dstE =
f, W_dstM = f
clk = 0

F_PC = 0000000000000079
D_stat = 0, D_icode = 1, D_ifun = 0, D_rA = c, D_rB = d, D_valc = 0000000000000005, D_va
lp = 000000000000007a
e_valE = fffffffffffffff8,E_stat = 0, E_icode = 1, E_valc = 0000000000000005, E_valA = 0
000000000000020, E_valB = 0000000000000020, E_dstE = f, E_dstM = f, EsrcA = 4, E_srcB =
4
M_stat = 1, M_icode = 0, M_cnd = 1, M_valE = fffffffffffffff8, M_valA = 000000000000002
0, M_dstE = f, M_dstM = f
W_stat = 0, W_icode = 1, W_valE = fffffffffffffff8, W_valM = 0000000000000048, W_dstE =
f, W_dstM = f
clk = 0
```

```
F_PC = 000000000000007a
D_stat = 0, D_icode = 1, D_ifun = 0, D_rA = c, D_rB = d, D_valc = 0000000000000005, D_va
lp = 000000000000007a
e_valE = fffffffffffffff8,E_stat = 0, E_icode = 1, E_valc = 0000000000000005, E_valA = 0
000000000000020, E_valB = 0000000000000020, E_dstE = f, E_dstM = f, EsrcA = 4, E_srcB =
4
M_stat = 0, M_icode = 1, M_cnd = 1, M_valE = fffffffffffffff8, M_valA = 000000000000002
0, M_dstE = f, M_dstM = f
W_stat = 1, W_icode = 0, W_valE = fffffffffffffff8, W_valM = 0000000000000048, W_dstE =
f, W_dstM = f
```

Final register status :

*reg_post - Notepad

File Edit Format View Help

```
// 0x00000000
0000000000000001
0000000000000002
0000000000000003
0000000000000007
0000000000000020
0000000000000006
0000000000000007
0000000000000008
000000000000000c
000000000000000a
0000000000000007
000000000000000c
0000000000000007
0000000000000005
000000000000000f
```

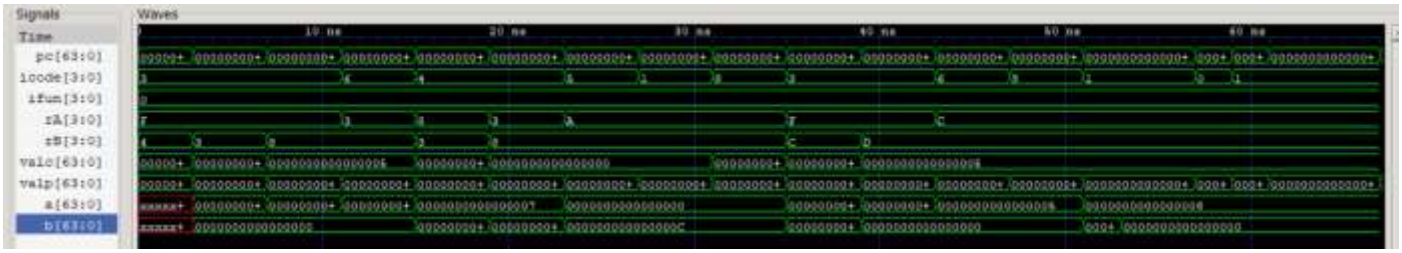
MEMORY STATUS :

mem post - Notepad

File Edit Format View Help

```
// 0x00000000
xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx
0000000000000008
0000000000000008
0000000000000008
xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx
0000000000000004
xxxxxxxxxxxxxxxx
0000000000000007
xxxxxxxxxxxxxxxx
000000000000000c
xxxxxxxxxxxxxxxx
// 0x00000010
xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx
```

Gtkwave plots :



Testing Condition jump with condition satisfied :

Input :

```
addq %rcx %rcx
jne L2
irmovq %r14 $33
L2:
irmovq %r14 $55
```

OUTPUT :

```
VCD info: dumpfile processor.vcd opened for output.
F_PC = 0000000000000000
D_stat = x, D_icode = x, D_ifun = x, D_rA = x, D_rB = x, D_valc = xxxxxxxxxxxxxxxx, D_valp =
xxxxxxxxxxxxxxxx
e_valE = xxxxxxxxxxxxxxxx,E_stat = x, E_icode = x, E_valc = xxxxxxxxxxxxxxxx, E_valA = xxxxxxxxxxxxxxxx,
E_valB = xxxxxxxxxxxxxxxx, E_dstE = x, E_dstM = x, EsrcA = x, E_srcB = x
M_stat = x, M_icode = x, M_cnd = x, M_valE = xxxxxxxxxxxxxxxx, M_valA = xxxxxxxxxxxxxxxx, M_dstE = x,
M_dstM = x
W_stat = x, W_icode = x, W_valE = xxxxxxxxxxxxxxxx, W_valM = xxxxxxxxxxxxxxxx, W_dstE = x, W_dstM = x
clk = 0

F_PC = 0000000000000002
D_stat = 0, D_icode = 6, D_ifun = 0, D_rA = 1, D_rB = 1, D_valc = xxxxxxxxxxxxxxxx, D_valp =
0000000000000002
e_valE = xxxxxxxxxxxxxxxx,E_stat = x, E_icode = x, E_valc = xxxxxxxxxxxxxxxx, E_valA = xxxxxxxxxxxxxxxx,
E_valB = xxxxxxxxxxxxxxxx, E_dstE = x, E_dstM = x, EsrcA = x, E_srcB = x
M_stat = x, M_icode = x, M_cnd = 1, M_valE = xxxxxxxxxxxxxxxx, M_valA = xxxxxxxxxxxxxxxx, M_dstE = x,
M_dstM = x
W_stat = x, W_icode = x, W_valE = xxxxxxxxxxxxxxxx, W_valM = xxxxxxxxxxxxxxxx, W_dstE = x, W_dstM = x
clk = 0

F_PC = 0000000000000002
D_stat = 0, D_icode = 6, D_ifun = 0, D_rA = 1, D_rB = 1, D_valc = xxxxxxxxxxxxxxxx, D_valp =
0000000000000002
e_valE = 0000000000000004,E_stat = 0, E_icode = 6, E_valc = xxxxxxxxxxxxxxxx, E_valA = 0000000000000002,
E_valB = 0000000000000002, E_dstE = 1, E_dstM = f, EsrcA = 1, E_srcB = 1
M_stat = x, M_icode = x, M_cnd = 1, M_valE = xxxxxxxxxxxxxxxx, M_valA = xxxxxxxxxxxxxxxx, M_dstE = x,
M_dstM = x
W_stat = x, W_icode = x, W_valE = xxxxxxxxxxxxxxxx, W_valM = xxxxxxxxxxxxxxxx, W_dstE = x, W_dstM = x
clk = 0

F_PC = 0000000000000016
D_stat = 0, D_icode = 7, D_ifun = 4, D_rA = 1, D_rB = 1, D_valc = 0000000000000016, D_valp =
000000000000000b
e_valE = 0000000000000008,E_stat = 0, E_icode = 6, E_valc = xxxxxxxxxxxxxxxx, E_valA = 0000000000000004,
E_valB = 0000000000000004, E_dstE = 1, E_dstM = f, EsrcA = 1, E_srcB = 1
M_stat = 0, M_icode = 6, M_cnd = 1, M_valE = 0000000000000004, M_valA = 0000000000000002, M_dstE = 1,
M_dstM = f
W_stat = x, W_icode = x, W_valE = xxxxxxxxxxxxxxxx, W_valM = xxxxxxxxxxxxxxxx, W_dstE = x, W_dstM = x
clk = 0
```

```
WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 0000000000000016
D_stat = 0, D_icode = 7, D_ifun = 4, D_rA = 1, D_rB = 1, D_valc = 0000000000000016, D_valp =
000000000000000b
e_valE = fffffffffffffff8,E_stat = 0, E_icode = 7, E_valc = 0000000000000016, E_valA = 000000000000000b,
E_valB = 0000000000000008, E_dstE = f, E_dstM = f, EsrcA = 1, E_srcB = 1
M_stat = 0, M_icode = 6, M_cnd = 1, M_valE = 0000000000000008, M_valA = 0000000000000004, M_dstE = 1,
M_dstM = f
W_stat = 0, W_icode = 6, W_valE = 0000000000000004, W_valM = xxxxxxxxxxxxxxxx, W_dstE = 1, W_dstM = f
clk = 0

WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 0000000000000020
D_stat = 0, D_icode = 3, D_ifun = 0, D_rA = f, D_rB = e, D_valc = 0000000000000055, D_valp =
0000000000000020
e_valE = fffffffffffffff8,E_stat = 0, E_icode = 7, E_valc = 0000000000000016, E_valA = 000000000000000b,
E_valB = 0000000000000008, E_dstE = f, E_dstM = f, EsrcA = 1, E_srcB = 1
M_stat = 0, M_icode = 7, M_cnd = 1, M_valE = fffffffffffffff8, M_valA = 000000000000000b, M_dstE = f,
M_dstM = f
W_stat = 0, W_icode = 6, W_valE = 0000000000000008, W_valM = xxxxxxxxxxxxxxxx, W_dstE = 1, W_dstM = f
clk = 0

F_PC = 0000000000000020
D_stat = 0, D_icode = 3, D_ifun = 0, D_rA = f, D_rB = e, D_valc = 0000000000000055, D_valp =
0000000000000020
e_valE = 0000000000000055,E_stat = 0, E_icode = 3, E_valc = 0000000000000055, E_valA = 0000000000000008,
E_valB = 0000000000000008, E_dstE = e, E_dstM = f, EsrcA = 1, E_srcB = 1
M_stat = 0, M_icode = 7, M_cnd = 1, M_valE = fffffffffffffff8, M_valA = 000000000000000b, M_dstE = f,
M_dstM = f
W_stat = 0, W_icode = 7, W_valE = fffffffffffffff8, W_valM = xxxxxxxxxxxxxxxx, W_dstE = f, W_dstM = f
clk = 0

F_PC = 0000000000000021
D_stat = 1, D_icode = 0, D_ifun = 0, D_rA = f, D_rB = e, D_valc = 0000000000000055, D_valp =
0000000000000021
e_valE = 0000000000000055,E_stat = 0, E_icode = 3, E_valc = 0000000000000055, E_valA = 0000000000000004,
E_valB = 0000000000000004, E_dstE = e, E_dstM = f, EsrcA = 1, E_srcB = 1
M_stat = 0, M_icode = 3, M_cnd = 1, M_valE = 0000000000000055, M_valA = 0000000000000008, M_dstE = e,
M_dstM = f
W_stat = 0, W_icode = 7, W_valE = fffffffffffffff8, W_valM = xxxxxxxxxxxxxxxx, W_dstE = f, W_dstM = f
clk = 0
```

```
WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 0000000000000021
D_stat = 1, D_icode = 0, D_ifun = 0, D_rA = f, D_rB = e, D_valc = 0000000000000055, D_valp =
0000000000000021
e_valE = fffffffffffffff8,E_stat = 1, E_icode = 0, E_valc = 0000000000000055, E_valA = 0000000000000008,
E_valB = 0000000000000008, E_dstE = f, E_dstM = f, EsrcA = 1, E_srcB = 1
M_stat = 0, M_icode = 3, M_cnd = 1, M_valE = 0000000000000055, M_valA = 0000000000000004, M_dstE = e,
M_dstM = f
W_stat = 0, W_icode = 3, W_valE = 0000000000000055, W_valM = xxxxxxxxxxxxxxxx, W_dstE = e, W_dstM = f
clk = 0

WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 0000000000000022
D_stat = 1, D_icode = 0, D_ifun = 0, D_rA = f, D_rB = e, D_valc = 0000000000000055, D_valp =
0000000000000022
e_valE = fffffffffffffff8,E_stat = 1, E_icode = 0, E_valc = 0000000000000055, E_valA = 0000000000000008,
E_valB = 0000000000000008, E_dstE = f, E_dstM = f, EsrcA = 1, E_srcB = 1
M_stat = 1, M_icode = 0, M_cnd = 1, M_valE = fffffffffffffff8, M_valA = 0000000000000008, M_dstE = f,
M_dstM = f
W_stat = 0, W_icode = 3, W_valE = 0000000000000055, W_valM = xxxxxxxxxxxxxxxx, W_dstE = e, W_dstM = f
clk = 0

F_PC = 0000000000000022
D_stat = 1, D_icode = 0, D_ifun = 0, D_rA = f, D_rB = e, D_valc = 0000000000000055, D_valp =
0000000000000022
e_valE = fffffffffffffff8,E_stat = 1, E_icode = 0, E_valc = 0000000000000055, E_valA = 0000000000000008,
E_valB = 0000000000000008, E_dstE = f, E_dstM = f, EsrcA = 1, E_srcB = 1
M_stat = 1, M_icode = 0, M_cnd = 1, M_valE = fffffffffffffff8, M_valA = 0000000000000008, M_dstE = f,
M_dstM = f
W_stat = 1, W_icode = 0, W_valE = fffffffffffffff8, W_valM = xxxxxxxxxxxxxxxx, W_dstE = f, W_dstM = f
```

Testing Condition jump with condition NOT satisfied :

INPUTS :

```
addq %rcx %rcx
je L2
irmovq %r14 $33
L2:
irmovq %r14 $55
```

Our jump instruction condition is not satisfied and thus comes back to the irmov function above L12.

OUTPUT :

```
VCD info: dumpfile processor.vcd opened for output.
F_PC = 0000000000000000
D_stat = x, D_icode = x, D_ifun = x, D_rA = x, D_rB = x, D_valc = xxxxxxxxxxxxxxxx, D_valp =
xxxxxxxxxxxxxxxx
e_valE = xxxxxxxxxxxxxxxx,E_stat = x, E_icode = x, E_valc = xxxxxxxxxxxxxxxx, E_valA = xxxxxxxxxxxxxxxx,
E_valB = xxxxxxxxxxxxxxxx, E_dstE = x, E_dstM = x, EsrcA = x, E_srcB = x
M_stat = x, M_icode = x, M_cnd = x, M_valE = xxxxxxxxxxxxxxxx, M_valA = xxxxxxxxxxxxxxxx, M_dstE = x,
M_dstM = x
W_stat = x, W_icode = x, W_valE = xxxxxxxxxxxxxxxx, W_valM = xxxxxxxxxxxxxxxx, W_dstE = x, W_dstM = x
clk = 0

F_PC = 0000000000000002
D_stat = 0, D_icode = 6, D_ifun = 0, D_rA = 1, D_rB = 1, D_valc = xxxxxxxxxxxxxxxx, D_valp =
0000000000000002
e_valE = xxxxxxxxxxxxxxxx,E_stat = x, E_icode = x, E_valc = xxxxxxxxxxxxxxxx, E_valA = xxxxxxxxxxxxxxxx,
E_valB = xxxxxxxxxxxxxxxx, E_dstE = x, E_dstM = x, EsrcA = x, E_srcB = x
M_stat = x, M_icode = x, M_cnd = 1, M_valE = xxxxxxxxxxxxxxxx, M_valA = xxxxxxxxxxxxxxxx, M_dstE = x,
M_dstM = x
W_stat = x, W_icode = x, W_valE = xxxxxxxxxxxxxxxx, W_valM = xxxxxxxxxxxxxxxx, W_dstE = x, W_dstM = x
clk = 0

F_PC = 0000000000000002
D_stat = 0, D_icode = 6, D_ifun = 0, D_rA = 1, D_rB = 1, D_valc = xxxxxxxxxxxxxxxx, D_valp =
0000000000000002
e_valE = 0000000000000010,E_stat = 0, E_icode = 6, E_valc = xxxxxxxxxxxxxxxx, E_valA = 0000000000000008,
E_valB = 0000000000000008, E_dstE = 1, E_dstM = f, EsrcA = 1, E_srcB = 1
M_stat = x, M_icode = x, M_cnd = 1, M_valE = xxxxxxxxxxxxxxxx, M_valA = xxxxxxxxxxxxxxxx, M_dstE = x,
M_dstM = x
W_stat = x, W_icode = x, W_valE = xxxxxxxxxxxxxxxx, W_valM = xxxxxxxxxxxxxxxx, W_dstE = x, W_dstM = x
clk = 0

F_PC = 0000000000000016
D_stat = 0, D_icode = 7, D_ifun = 3, D_rA = 1, D_rB = 1, D_valc = 0000000000000016, D_valp =
000000000000000b
e_valE = 0000000000000020,E_stat = 0, E_icode = 6, E_valc = xxxxxxxxxxxxxxxx, E_valA = 0000000000000010,
E_valB = 0000000000000010, E_dstE = 1, E_dstM = f, EsrcA = 1, E_srcB = 1
M_stat = 0, M_icode = 6, M_cnd = 1, M_valE = 0000000000000010, M_valA = 0000000000000008, M_dstE = 1,
M_dstM = f
W_stat = x, W_icode = x, W_valE = xxxxxxxxxxxxxxxx, W_valM = xxxxxxxxxxxxxxxx, W_dstE = x, W_dstM = x
clk = 0
```

```
WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 0000000000000016
D_stat = 0, D_icode = 7, D_ifun = 3, D_rA = 1, D_rB = 1, D_valc = 0000000000000016, D_valp =
000000000000000b
e_valE = ffffffffffffffff8,E_stat = 0, E_icode = 7, E_valc = 0000000000000016, E_valA = 000000000000000b,
E_valB = 0000000000000020, E_dstE = f, E_dstM = f, EsrcA = 1, E_srcB = 1
M_stat = 0, M_icode = 6, M_cnd = 1, M_valE = 0000000000000020, M_valA = 0000000000000010, M_dstE = 1,
M_dstM = f
W_stat = 0, W_icode = 6, W_valE = 0000000000000010, W_valM = xxxxxxxxxxxxxxxx, W_dstE = 1, W_dstM = f
clk = 0

WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 0000000000000020
D_stat = 0, D_icode = 3, D_ifun = 0, D_rA = f, D_rB = e, D_valc = 0000000000000055, D_valp =
0000000000000020
e_valE = ffffffffffffffff8,E_stat = 0, E_icode = 7, E_valc = 0000000000000016, E_valA = 000000000000000b,
E_valB = 0000000000000020, E_dstE = f, E_dstM = f, EsrcA = 1, E_srcB = 1
M_stat = 0, M_icode = 7, M_cnd = 0, M_valE = ffffffffffffffff8, M_valA = 000000000000000b, M_dstE = f,
M_dstM = f
W_stat = 0, W_icode = 6, W_valE = 0000000000000020, W_valM = xxxxxxxxxxxxxxxx, W_dstE = 1, W_dstM = f
clk = 0

F_PC = 0000000000000020
D_stat = 0, D_icode = 1, D_ifun = 0, D_rA = f, D_rB = e, D_valc = 0000000000000055, D_valp =
0000000000000020
e_valE = ffffffffffffffff8,E_stat = 0, E_icode = 1, E_valc = 0000000000000016, E_valA = 000000000000000b,
E_valB = 0000000000000020, E_dstE = f, E_dstM = f, EsrcA = 1, E_srcB = 1
M_stat = 0, M_icode = 7, M_cnd = 0, M_valE = ffffffffffffffff8, M_valA = 000000000000000b, M_dstE = f,
M_dstM = f
W_stat = 0, W_icode = 7, W_valE = ffffffffffffffff8, W_valM = xxxxxxxxxxxxxxxx, W_dstE = f, W_dstM = f
clk = 0

F_PC = 0000000000000015
D_stat = 0, D_icode = 1, D_ifun = 0, D_rA = f, D_rB = e, D_valc = 0000000000000055, D_valp =
0000000000000020
e_valE = ffffffffffffffff8,E_stat = 0, E_icode = 1, E_valc = 0000000000000016, E_valA = 000000000000000b,
E_valB = 0000000000000020, E_dstE = f, E_dstM = f, EsrcA = 1, E_srcB = 1
M_stat = 0, M_icode = 1, M_cnd = 0, M_valE = ffffffffffffffff8, M_valA = 000000000000000b, M_dstE = f,
M_dstM = f
W_stat = 0, W_icode = 7, W_valE = ffffffffffffffff8, W_valM = xxxxxxxxxxxxxxxx, W_dstE = f, W_dstM = f
clk = 0
```

```
F_PC = 0000000000000015
D_stat = 0, D_icode = 3, D_ifun = 0, D_rA = f, D_rB = e, D_valc = 0000000000000033, D_valp =
0000000000000015
e_valE = ffffffffffffffff8,E_stat = 0, E_icode = 1, E_valc = 0000000000000055, E_valA = 0000000000000020,
E_valB = 0000000000000020, E_dstE = f, E_dstM = f, EsrcA = 1, E_srcB = 1
M_stat = 0, M_icode = 1, M_cnd = 0, M_valE = ffffffffffffffff8, M_valA = 000000000000000b, M_dstE = f,
M_dstM = f
W_stat = 0, W_icode = 1, W_valE = ffffffffffffffff8, W_valM = xxxxxxxxxxxxxxxx, W_dstE = f, W_dstM = f
clk = 0

F_PC = 0000000000000016
D_stat = 1, D_icode = 0, D_ifun = 0, D_rA = f, D_rB = e, D_valc = 0000000000000033, D_valp =
0000000000000016
e_valE = 0000000000000033,E_stat = 0, E_icode = 3, E_valc = 0000000000000033, E_valA = 0000000000000020,
E_valB = 0000000000000020, E_dstE = e, E_dstM = f, EsrcA = 1, E_srcB = 1
M_stat = 0, M_icode = 1, M_cnd = 0, M_valE = ffffffffffffffff8, M_valA = 0000000000000020, M_dstE = f,
M_dstM = f
W_stat = 0, W_icode = 1, W_valE = ffffffffffffffff8, W_valM = xxxxxxxxxxxxxxxx, W_dstE = f, W_dstM = f
clk = 0

F_PC = 0000000000000016
D_stat = 1, D_icode = 0, D_ifun = 0, D_rA = f, D_rB = e, D_valc = 0000000000000033, D_valp =
0000000000000016
e_valE = fffffffffffffcd,E_stat = 1, E_icode = 0, E_valc = 0000000000000033, E_valA = 0000000000000020,
E_valB = 0000000000000020, E_dstE = f, E_dstM = f, EsrcA = 1, E_srcB = 1
M_stat = 0, M_icode = 3, M_cnd = 0, M_valE = 0000000000000033, M_valA = 0000000000000020, M_dstE = e,
M_dstM = f
W_stat = 0, W_icode = 1, W_valE = ffffffffffffffff8, W_valM = xxxxxxxxxxxxxxxx, W_dstE = f, W_dstM = f
clk = 0

WARNING: ./registerfile.v:51: $writememh: Standard inconsistency, following 1364-2005.
F_PC = 0000000000000020
D_stat = 0, D_icode = 3, D_ifun = 0, D_rA = f, D_rB = e, D_valc = 0000000000000055, D_valp =
0000000000000020
e_valE = fffffffffffffcd,E_stat = 1, E_icode = 0, E_valc = 0000000000000033, E_valA = 0000000000000020,
E_valB = 0000000000000020, E_dstE = f, E_dstM = f, EsrcA = 1, E_srcB = 1
M_stat = 1, M_icode = 0, M_cnd = 0, M_valE = fffffffffffffcd, M_valA = 0000000000000020, M_dstE = f,
M_dstM = f
W_stat = 0, W_icode = 3, W_valE = 0000000000000033, W_valM = xxxxxxxxxxxxxxxx, W_dstE = e, W_dstM = f
clk = 0
```

```
F_PC = 0000000000000020
D_stat = 0, D_icode = 3, D_ifun = 0, D_rA = f, D_rB = e, D_valc = 0000000000000055, D_valp =
0000000000000020
e_valE = 0000000000000055,E_stat = 0, E_icode = 3, E_valc = 0000000000000055, E_valA = 0000000000000020,
E_valB = 0000000000000020, E_dstE = e, E_dstM = f, EsrcA = 1, E_srcB = 1
M_stat = 1, M_icode = 0, M_cnd = 0, M_valE = fffffffffffffcd, M_valA = 0000000000000020, M_dstE = f,
M_dstM = f
W_stat = 1, W_icode = 0, W_valE = fffffffffffffcd, W_valM = xxxxxxxxxxxxxxxx, W_dstE = f, W_dstM = f
```

Processor Frequency **:**

The clock cycle time for our processor is 1ns

$$freq(Hz) = \frac{1}{(clock\ cycle\ time)}$$

$$freq\ (Hz) = \frac{1}{10^{-9}} s^{-1}$$

$$freq = 10^{9} Hz\ or\ 1GHz$$

Challenges Encountered :

- We faced difficulties with matching timings of all blocks but figured it out with the help of a textbook. Referring to ‘Computer Systems: A Programmer's Perspective’ by Randal Bryant helped a lot.
- Figuring how to implement memory and registers as text files and using them through our Verilog modules.