Intro to Processor Architecture:

Project Report (Pipeline):

NAME- ADITYA RAJ SINGH

ROLL No. 2022102067

Teammate: VENKATA RAMANA M N

INTRODUCTION:

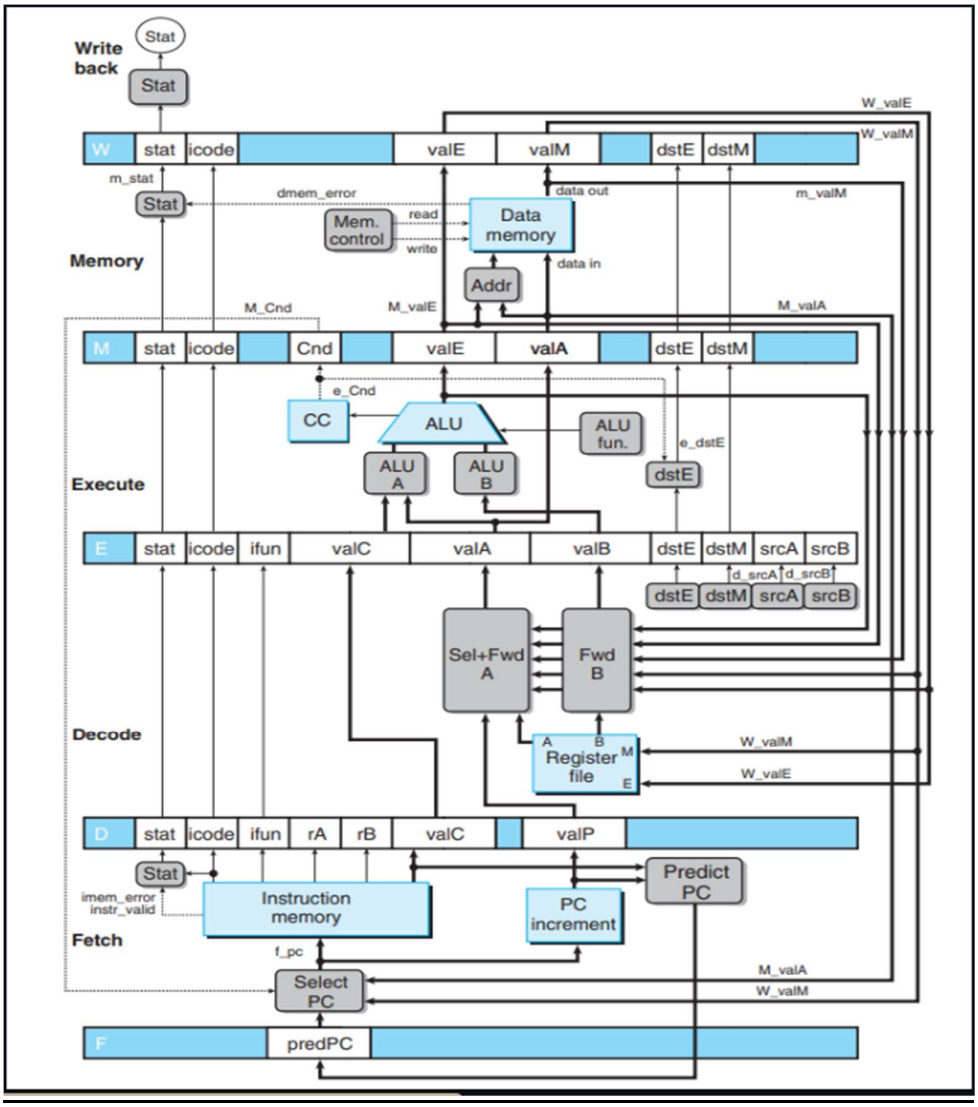
This project is an attempt to write a functional processor in verilog which supports the Y86-64 ISA with pipelining. Most of the reference is taken from the textbook Computer Systems A Programmer's Perspective by Randal E. Bryant and David R. O'Hallaron.

The implementation can be looked at in these major stages:

- *Fetch & PC_select*
- *Decode*
- *Execute & ALU implementation*
- *Memory & writeback*

Our final implementation would look something like this:

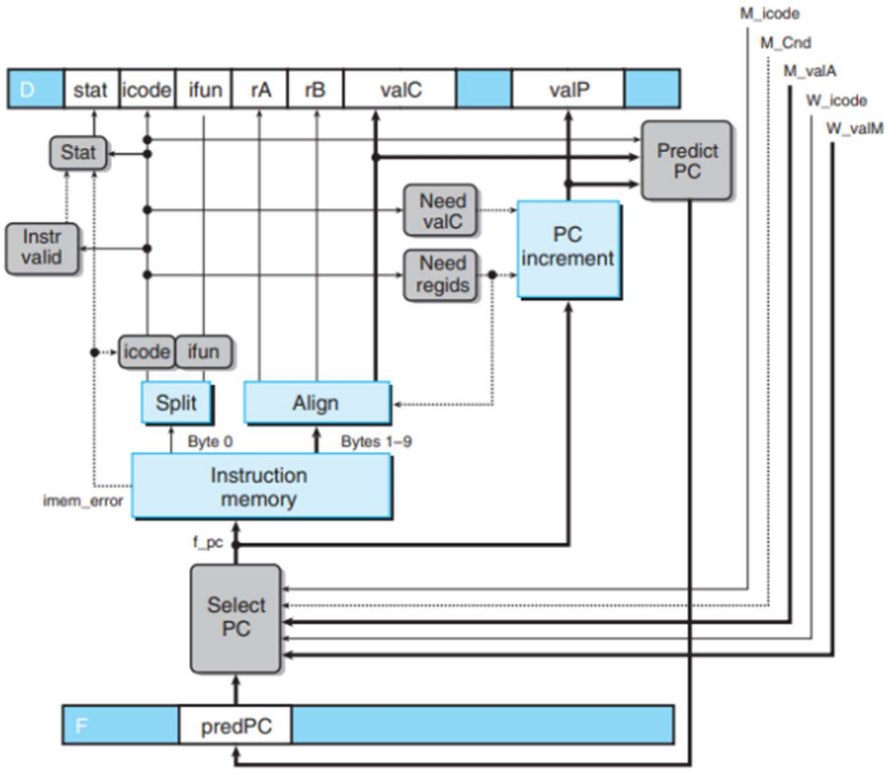
P.T.O.



Fetch & PC prediction:

In this stage we fetch the instruction from main memory and then also predict PC. Within the one cycle time limit, the processor can only predict the address of the next instruction.

Our implementation looks like this:



The fetch stage in a Y86 64-bit processor is responsible for retrieving the next instruction from memory and preparing it for execution. The fetch stage works as follows:

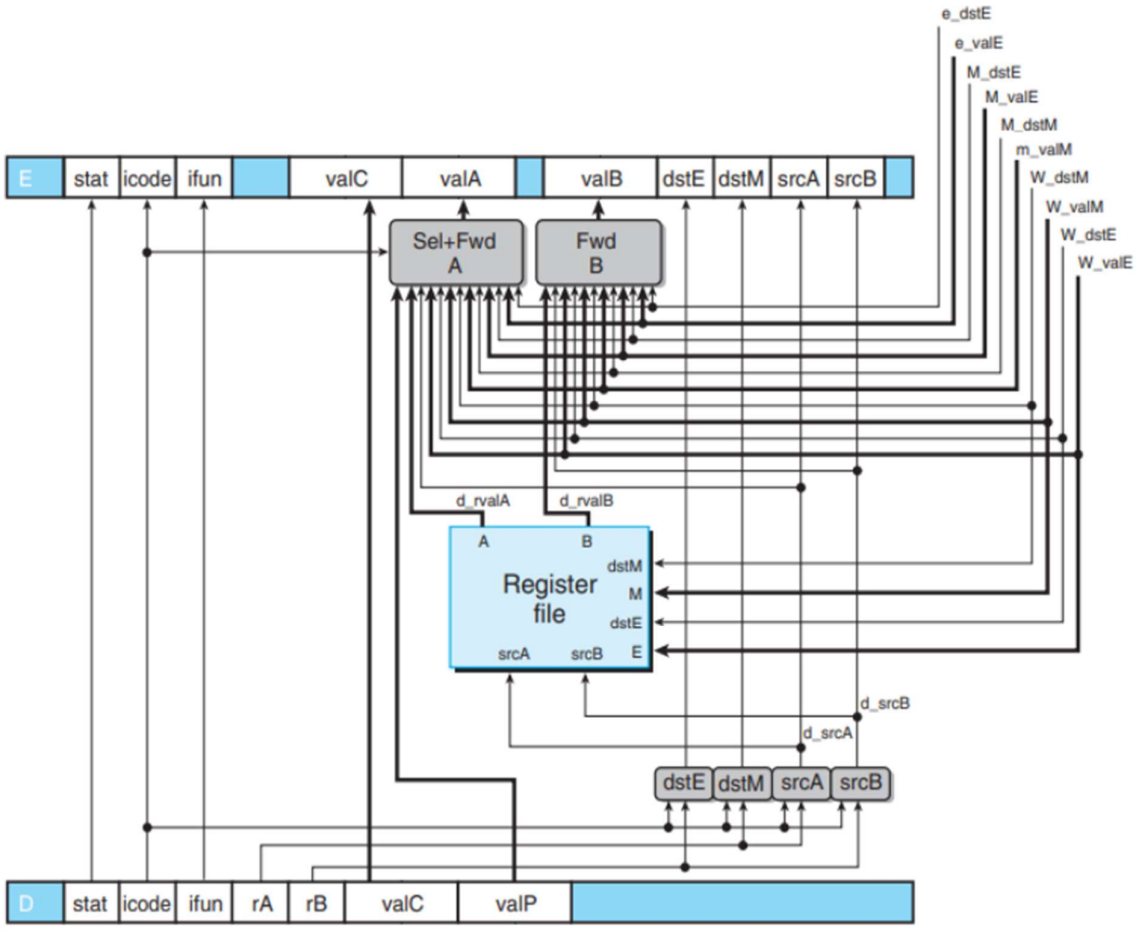
- *The program counter (PC) holds the address of the next instruction to be fetched.*
- *The fetch stage sends a read request to the memory subsystem at the address held by the PC.*
- *The instruction is retrieved from memory and stored in a buffer called the instruction register (IR).*
- *The PC is incremented to point to the next instruction.*
- *The fetched instruction is then passed on to the next stage of the pipeline, which is typically the decode stage.*

*The PC selection logic chooses between three program counter sources. As a mispredicted branch enters the memory stage, the value of **valP** for this instruction (indicating the address of the following instruction) is read from pipeline register M (signal **M_valA**). When a ret instruction enters the write-back stage, the return address is read from pipeline register W*

*(signal **W_valM**). All other cases use the predicted value of the PC, stored in pipeline register F (signal **F_predPC**).*

Overall, the fetch stage is responsible for reading instructions from memory and making them available for execution by the processor. Overall one of the major difference between fetch stage execution in a sequential model and this pipeline model is that in this model we move the PC update stage so that its logic is active at the beginning of the clock cycle by making it compute the PC value for the current instruction.

DECODE:



The decode stage is responsible for decoding the instruction fetched in the previous cycle and preparing the operands for the execution stage. The decode stage works as follows:

- *The instruction that was fetched in the previous cycle is loaded from the instruction register (IR) into the decode register (DR).*

- *The opcode of the instruction is extracted from the instruction and decoded to determine the type of instruction and the operands that are needed.*
- *The registers that are specified as operands in the instruction are read from the register file and their values are passed on to the execution stage.*
- *If the instruction involves a memory access, the address of the memory location is computed based on the operands and passed on to the execution stage.*
- *Control signals are generated based on the instruction type and passed on to the execution stage to enable the appropriate functional units.*
- *The decoded instruction and its operands are then passed on to the execution stage to be executed.*

Pipelining's decode stage is crucial because it gets the operands ready for the execution stage so that it may start processing the instruction as soon as it becomes available. The fetch, decode, and execute phases of the CPU can be combined to boost throughput and overall performance.

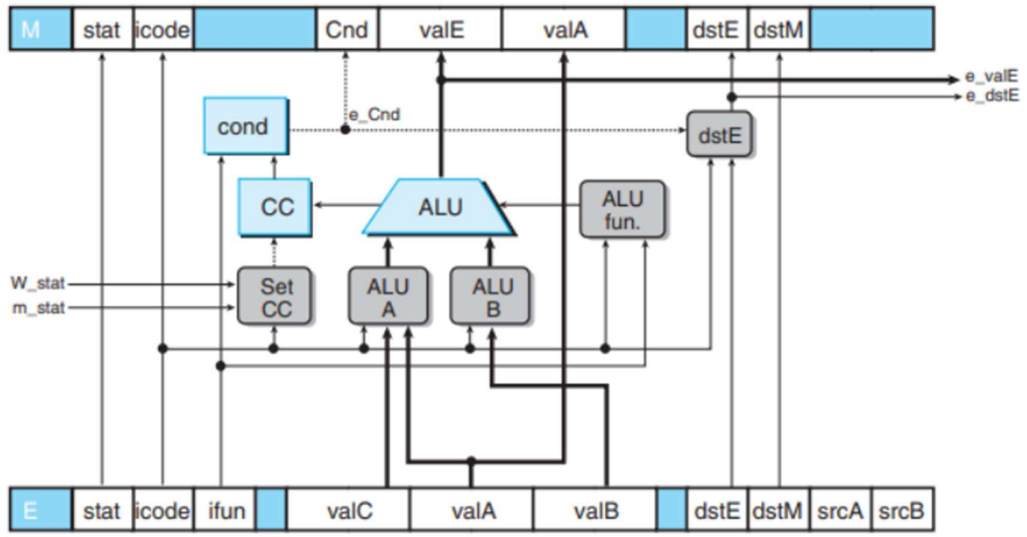
Execute:

The execute stage in a Y86 64-bit processor with pipelining is responsible for carrying out the operation specified by the instruction, using the operands that were fetched and prepared in the previous stages. The execute stage works as follows:

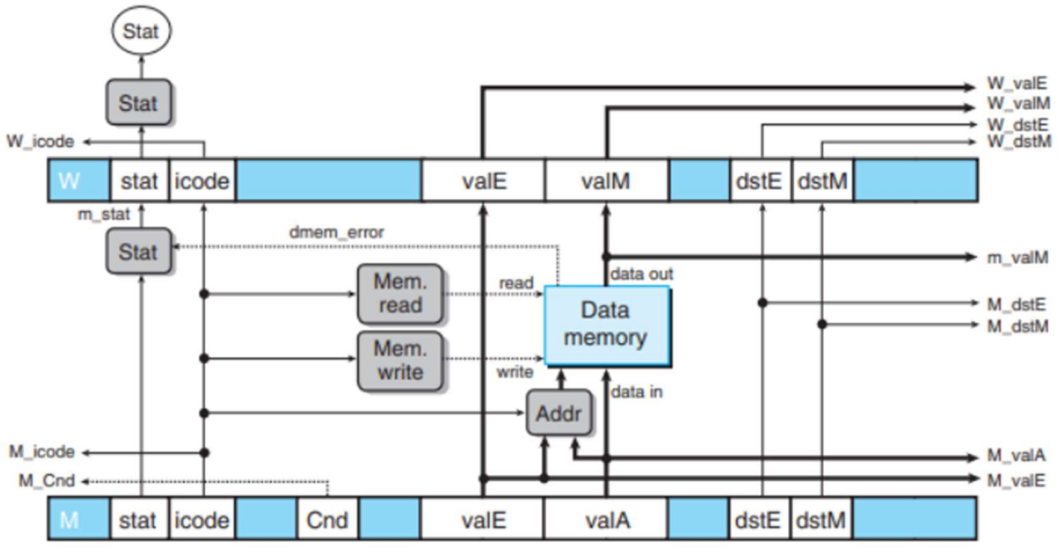
- *The instruction and its operands are received from the previous stage, typically the decode stage. The operation specified by the instruction is performed on the operands.*
- *If the instruction involves a memory access, the memory subsystem is accessed to read or write the data.*
- *If the instruction is a branch instruction, the branch condition is evaluated, and the program counter (PC) is updated accordingly.*
- *The result of the operation is then passed on to the next stage of the pipeline, typically the memory stage or the write-back stage.*

Hazard detection and handling techniques, such as forwarding, stalling, and branch prediction, may also be used during the execution stage to ensure correct program execution in the presence of hazards.

Our implementation looks like this:



Memory:



The memory stage in a Y86 64-bit processor with pipelining is responsible for accessing memory to read or write data, and also for handling any memory-related hazards that may occur in the pipeline. The memory stage works as follows:

- *If the instruction involves a memory access, the memory address is computed based on the operands received from the previous stage, typically the execute stage.*
- *A read or write request is sent to the memory subsystem to access the data at the memory location specified by the address.*
- *If the instruction is a load instruction, the data that is read from memory is passed on to the next stage of the pipeline, typically the write-back stage.*
- *If the instruction is a store instruction, the data to be written to memory is passed on to the memory subsystem.*
- *If a memory-related hazard occurs, such as a load-use hazard, where a later instruction depends on the data loaded by an earlier instruction, the pipeline may need to be stalled or data forwarding techniques may need to be used to resolve the hazard.*

Because memory accesses can significantly impede processor performance, the memory step is a crucial one in the pipeline. Techniques like caching, prefetching, and speculative execution may be utilised to boost performance by reducing memory-related delays, this is common in modern processors but it's far beyond the scope of this project.

Write-back:

Registers are updated in this stage. Often, instructions require registered values from previous instructions but this cannot be done until the previous instructions have gone through writeback stage. Thus it is not uncommon to stall instructions to avoid such control hazards.

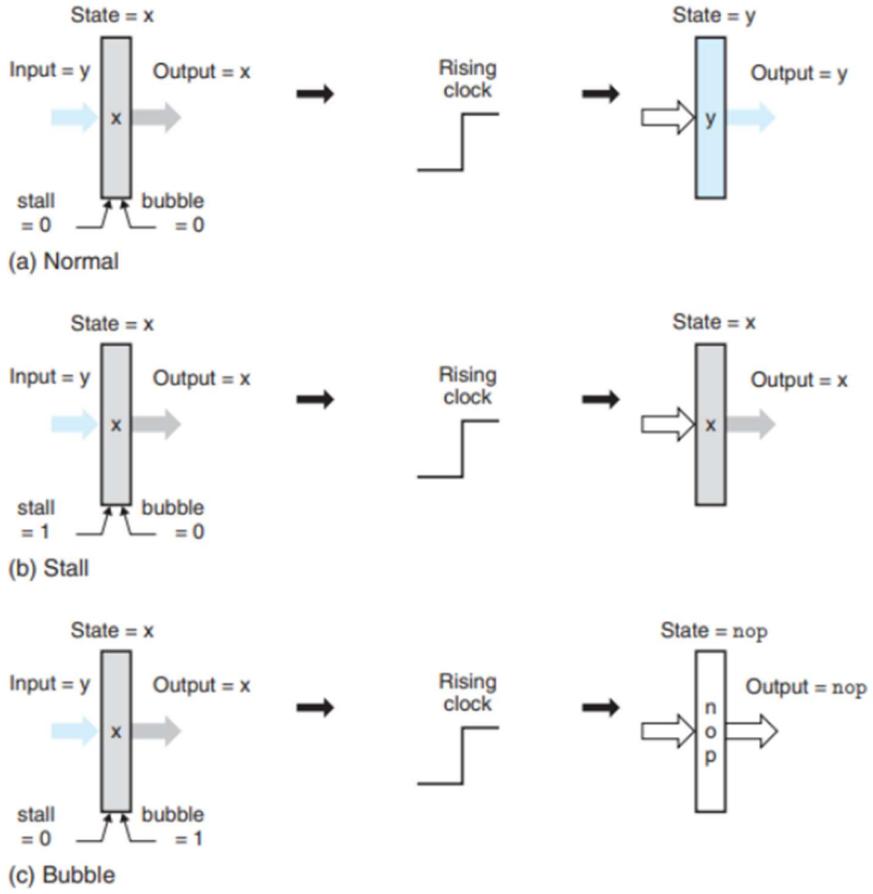
PIPELINE Control Logic, DATA forwording and Hazards:

We are now prepared to create the pipeline control logic to finish our design. The following control scenarios, for which alternative mechanisms like data forwarding and branch prediction are insufficient, must be handled using this logic:

- *Load/use hazards: The pipeline must stall for one cycle between an instruction that reads a value from memory and an instruction that uses this value.*

- *Processing ret: The pipeline must stall until the ret instruction reaches the write-back stage.*
- *Mispredicted branches. By the time the branch logic detects that a jump should not have been taken, several instructions at the branch target will have started down the pipeline. These instructions must be canceled, and fetching should begin at the instruction following the jump instruction.*

We will use bubbling and stalling to overcome these hazzards.



- *Bubbling is a technique used to insert a "bubble" or a no-operation (NOP) instruction into the pipeline to stall the pipeline and allow the preceding instructions to complete execution. This is typically used to resolve control hazards, where a branch instruction has not yet resolved and the pipeline needs to wait until the correct instruction path is determined.*
- *Stalling, on the other hand, is a technique used to hold a stage of the pipeline in place and prevent it from advancing until the preceding stage has completed its work. This is typically used to resolve data hazards,*

where a later instruction depends on data produced by an earlier instruction that has not yet completed execution. The stalled instruction remains in the pipeline, but no further progress is made until the data dependency is resolved.

Both techniques are used to ensure correct program execution and avoid hazards that can arise in pipelined processors. However, they can also introduce additional delays and reduce performance if used excessively, so they need to be used judiciously. Techniques such as forwarding and speculation can be used to reduce the need for bubbling and stalling in the pipeline.

RESULTS (PIPELINE PROCESSOR WORKING):

We gave the processor model a series of different instructions to check it's functionality and robustness. For example below is a set of instructions given and the result obtained:

```
 // Instruction memory
hlt = 0;
   // nop instruction
   memory[0]  = 8'b00010000;
   memory[1]  = 8'b01100000; // Opq add
   memory[2]  = 8'b00000001; // rA = 0, rB = 1;

   // irmovq instruction
   memory[3]  = 8'b00110000;
   memory[4]  = 8'b11110010; // F, rB = 2;
   memory[5]  = 8'b11111111; // 1st byte of V = 255, rest all bytes will be zero
   memory[6]  = 8'b00000000; // 2nd byte
   memory[7]  = 8'b00000000; // 3rd byte
   memory[8]  = 8'b00000000; // 4th byte
   memory[9]  = 8'b00000000; // 5th byte
   memory[10] = 8'b00000000; // 6th byte
   memory[11] = 8'b00000000; // 7th byte
   memory[12] = 8'b00000000; // 8th byte (This completes irmovq)

   memory[13]  = 8'b00010000; // nop instruction
   memory[14]  = 8'b00010000; // nop instruction
   memory[15]  = 8'b00010000; // nop instruction
   memory[16]  = 8'b00010000; // nop instruction

   // irmovq instruction
   memory[17] = 8'b00110000;
   memory[18] = 8'b11110011; // F, rB = 3;
   memory[19] = 8'b00000101; // 1st byte of V = 5, rest all bytes will be zero
   memory[20] = 8'b00000000; // 2nd byte
   memory[21] = 8'b00000000; // 3rd byte
   memory[22] = 8'b00000000; // 4th byte
   memory[23] = 8'b00000000; // 5th byte
   memory[24] = 8'b00000000; // 6th byte
   memory[25] = 8'b00000000; // 7th byte
   memory[26] = 8'b00000000; // 8th byte (This completes irmovq)
```

```
memory[27]  = 8'b00010000; // nop instruction
memory[28]  = 8'b00010000; // nop instruction
memory[29]  = 8'b00010000; // nop instruction
memory[30]  = 8'b00010000; // nop instruction

// irmovq instruction
memory[31] = 8'b00110000;
memory[32] = 8'b11110100; // F, rB = 4;
memory[33] = 8'b00000101; // 1st byte of V = 5, rest all bytes will be zero
memory[34] = 8'b00000000; // 2nd byte
memory[35] = 8'b00000000; // 3rd byte
memory[36] = 8'b00000000; // 4th byte
memory[37] = 8'b00000000; // 5th byte
memory[38] = 8'b00000000; // 6th byte
memory[39] = 8'b00000000; // 7th byte
memory[40] = 8'b00000000; // 8th byte (This completes irmovq)

memory[41]  = 8'b00010000; // nop instruction
memory[42]  = 8'b00010000; // nop instruction
memory[43]  = 8'b00010000; // nop instruction
memory[44]  = 8'b00010000; // nop instruction

memory[45] = 8'b00100000; // rrmovq
memory[46] = 8'b01000101; // rA = 4; rB = 5;

memory[47] = 8'b01100000; // Opq add
memory[48] = 8'b00110100; // rA = 3 and rB = 4, final value in rB(4) = 10;

memory[49] = 8'b00000000; // halt
```

Output:

```
VCD info: dumpfile processor.vcd opened for output.
time=0,
 F_stall=0, W_stall=0,
clk=1, f_icode=1, f_ifun=0, f_rA=x, f_rB=x, f_valP=1, f_valC=x, D_icode=1, E_icode=x,
 M_icode=x, W_icode=x, mem_err=0, halt=0, 0=0, 1=1, 2=2, 3=3, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=x,m_valE=x,e_valE=x,M_valE=x

time=5,
 F_stall=0, W_stall=0,
clk=0, f_icode=1, f_ifun=0, f_rA=x, f_rB=x, f_valP=1, f_valC=x, D_icode=1, E_icode=x,
 M_icode=x, W_icode=x, mem_err=0, halt=0, 0=0, 1=1, 2=2, 3=3, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=x,m_valE=x,e_valE=x,M_valE=x

time=10,
 F_stall=0, W_stall=0,
clk=1, f_icode=6, f_ifun=0, f_rA=0, f_rB=1, f_valP=3, f_valC=x, D_icode=1, E_icode=1,
 M_icode=x, W_icode=x, mem_err=0, halt=0, 0=0, 1=1, 2=2, 3=3, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=x,m_valE=x,e_valE=x,M_valE=x

time=15,
 F_stall=0, W_stall=0,
clk=0, f_icode=6, f_ifun=0, f_rA=0, f_rB=1, f_valP=3, f_valC=x, D_icode=1, E_icode=1,
 M_icode=x, W_icode=x, mem_err=0, halt=0, 0=0, 1=1, 2=2, 3=3, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=x,m_valE=x,e_valE=x,M_valE=x

time=20,
 F_stall=0, W_stall=0,
clk=1, f_icode=3, f_ifun=0, f_rA=15, f_rB=2, f_valP=13, f_valC=255, D_icode=6, E_icode=1,
 M_icode=1, W_icode=x, mem_err=0, halt=0, 0=0, 1=1, 2=2, 3=3, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=x,m_valE=x,e_valE=x,M_valE=x

time=25,
 F_stall=0, W_stall=0,
clk=0, f_icode=3, f_ifun=0, f_rA=15, f_rB=2, f_valP=13, f_valC=255, D_icode=6, E_icode=1,
 M_icode=1, W_icode=x, mem_err=0, halt=0, 0=0, 1=1, 2=2, 3=3, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=x,m_valE=x,e_valE=x,M_valE=x

time=30,
 F_stall=0, W_stall=0,
clk=1, f_icode=1, f_ifun=0, f_rA=15, f_rB=2, f_valP=14, f_valC=255, D_icode=3, E_icode=6,
 M_icode=1, W_icode=1, mem_err=0, halt=0, 0=0, 1=1, 2=2, 3=3, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=x,m_valE=x,e_valE=1,M_valE=x

time=35,
 F_stall=0, W_stall=0,
clk=0, f_icode=1, f_ifun=0, f_rA=15, f_rB=2, f_valP=14, f_valC=255, D_icode=3, E_icode=6,
 M_icode=1, W_icode=1, mem_err=0, halt=0, 0=0, 1=1, 2=2, 3=3, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=x,m_valE=x,e_valE=1,M_valE=x

time=40,
 F_stall=0, W_stall=0,
clk=1, f_icode=1, f_ifun=0, f_rA=15, f_rB=2, f_valP=15, f_valC=255, D_icode=1, E_icode=3,
 M_icode=6, W_icode=1, mem_err=0, halt=0, 0=0, 1=1, 2=2, 3=3, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=x,m_valE=1,e_valE=255,M_valE=1

time=45,
 F_stall=0, W_stall=0,
clk=0, f_icode=1, f_ifun=0, f_rA=15, f_rB=2, f_valP=15, f_valC=255, D_icode=1, E_icode=3,
 M_icode=6, W_icode=1, mem_err=0, halt=0, 0=0, 1=1, 2=2, 3=3, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=x,m_valE=1,e_valE=255,M_valE=1
```

```
time=50,
 F_stall=0, W_stall=0,
clk=1, f_icode=1, f_ifun=0, f_rA=15, f_rB=2, f_valP=16, f_valC=255, D_icode=1, E_icode=1,
 M_icode=3, W_icode=6, mem_err=0, halt=0, 0=0, 1=1, 2=2, 3=3, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=1,m_valE=255,e_valE=255,M_valE=255

time=55,
 F_stall=0, W_stall=0,
clk=0, f_icode=1, f_ifun=0, f_rA=15, f_rB=2, f_valP=16, f_valC=255, D_icode=1, E_icode=1,
 M_icode=3, W_icode=6, mem_err=0, halt=0, 0=0, 1=1, 2=2, 3=3, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=1,m_valE=255,e_valE=255,M_valE=255

time=60,
 F_stall=0, W_stall=0,
clk=1, f_icode=1, f_ifun=0, f_rA=15, f_rB=2, f_valP=17, f_valC=255, D_icode=1, E_icode=1,
 M_icode=1, W_icode=3, mem_err=0, halt=0, 0=0, 1=1, 2=2, 3=3, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=255,m_valE=255,e_valE=255,M_valE=255

time=65,
 F_stall=0, W_stall=0,
clk=0, f_icode=1, f_ifun=0, f_rA=15, f_rB=2, f_valP=17, f_valC=255, D_icode=1, E_icode=1,
 M_icode=1, W_icode=3, mem_err=0, halt=0, 0=0, 1=1, 2=2, 3=3, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=255,m_valE=255,e_valE=255,M_valE=255

time=70,
 F_stall=0, W_stall=0,
clk=1, f_icode=3, f_ifun=0, f_rA=15, f_rB=3, f_valP=27, f_valC=5, D_icode=1, E_icode=1,
 M_icode=1, W_icode=1, mem_err=0, halt=0, 0=0, 1=1, 2=255, 3=3, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=255,m_valE=255,e_valE=255,M_valE=255

time=75,
 F_stall=0, W_stall=0,
clk=0, f_icode=3, f_ifun=0, f_rA=15, f_rB=3, f_valP=27, f_valC=5, D_icode=1, E_icode=1,
 M_icode=1, W_icode=1, mem_err=0, halt=0, 0=0, 1=1, 2=255, 3=3, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=255,m_valE=255,e_valE=255,M_valE=255

time=80,
 F_stall=0, W_stall=0,
clk=1, f_icode=1, f_ifun=0, f_rA=15, f_rB=3, f_valP=28, f_valC=5, D_icode=3, E_icode=1,
 M_icode=1, W_icode=1, mem_err=0, halt=0, 0=0, 1=1, 2=255, 3=3, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=255,m_valE=255,e_valE=255,M_valE=255

time=85,
 F_stall=0, W_stall=0,
clk=0, f_icode=1, f_ifun=0, f_rA=15, f_rB=3, f_valP=28, f_valC=5, D_icode=3, E_icode=1,
 M_icode=1, W_icode=1, mem_err=0, halt=0, 0=0, 1=1, 2=255, 3=3, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=255,m_valE=255,e_valE=255,M_valE=255

time=90,
 F_stall=0, W_stall=0,
clk=1, f_icode=1, f_ifun=0, f_rA=15, f_rB=3, f_valP=29, f_valC=5, D_icode=1, E_icode=3,
 M_icode=1, W_icode=1, mem_err=0, halt=0, 0=0, 1=1, 2=255, 3=3, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=255,m_valE=255,e_valE=5,M_valE=255

time=95,
 F_stall=0, W_stall=0,
clk=0, f_icode=1, f_ifun=0, f_rA=15, f_rB=3, f_valP=29, f_valC=5, D_icode=1, E_icode=3,
 M_icode=1, W_icode=1, mem_err=0, halt=0, 0=0, 1=1, 2=255, 3=3, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=255,m_valE=255,e_valE=5,M_valE=255

time=100,
 F_stall=0, W_stall=0,
clk=1, f_icode=1, f_ifun=0, f_rA=15, f_rB=3, f_valP=30, f_valC=5, D_icode=1, E_icode=1,
 M_icode=3, W_icode=1, mem_err=0, halt=0, 0=0, 1=1, 2=255, 3=3, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=255,m_valE=5,e_valE=5,M_valE=5
```

```
time=105,
 F_stall=0, W_stall=0,
clk=0, f_icode=1, f_ifun=0, f_rA=15, f_rB=3, f_valP=30, f_valC=5, D_icode=1, E_icode=1,
 M_icode=3, W_icode=1, mem_err=0, halt=0, 0=0, 1=1, 2=255, 3=3, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=255,m_valE=5,e_valE=5,M_valE=5

time=110,
 F_stall=0, W_stall=0,
clk=1, f_icode=1, f_ifun=0, f_rA=15, f_rB=3, f_valP=31, f_valC=5, D_icode=1, E_icode=1,
 M_icode=1, W_icode=3, mem_err=0, halt=0, 0=0, 1=1, 2=255, 3=3, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=5,m_valE=5,e_valE=5,M_valE=5

time=115,
 F_stall=0, W_stall=0,
clk=0, f_icode=1, f_ifun=0, f_rA=15, f_rB=3, f_valP=31, f_valC=5, D_icode=1, E_icode=1,
 M_icode=1, W_icode=3, mem_err=0, halt=0, 0=0, 1=1, 2=255, 3=3, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=5,m_valE=5,e_valE=5,M_valE=5

time=120,
 F_stall=0, W_stall=0,
clk=1, f_icode=3, f_ifun=0, f_rA=15, f_rB=4, f_valP=41, f_valC=5, D_icode=1, E_icode=1,
 M_icode=1, W_icode=1, mem_err=0, halt=0, 0=0, 1=1, 2=255, 3=5, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=5,m_valE=5,e_valE=5,M_valE=5

time=125,
 F_stall=0, W_stall=0,
clk=0, f_icode=3, f_ifun=0, f_rA=15, f_rB=4, f_valP=41, f_valC=5, D_icode=1, E_icode=1,
 M_icode=1, W_icode=1, mem_err=0, halt=0, 0=0, 1=1, 2=255, 3=5, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=5,m_valE=5,e_valE=5,M_valE=5

time=130,
 F_stall=0, W_stall=0,
clk=1, f_icode=1, f_ifun=0, f_rA=15, f_rB=4, f_valP=42, f_valC=5, D_icode=3, E_icode=1,
 M_icode=1, W_icode=1, mem_err=0, halt=0, 0=0, 1=1, 2=255, 3=5, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=5,m_valE=5,e_valE=5,M_valE=5

time=135,
 F_stall=0, W_stall=0,
clk=0, f_icode=1, f_ifun=0, f_rA=15, f_rB=4, f_valP=42, f_valC=5, D_icode=3, E_icode=1,
 M_icode=1, W_icode=1, mem_err=0, halt=0, 0=0, 1=1, 2=255, 3=5, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=5,m_valE=5,e_valE=5,M_valE=5

time=140,
 F_stall=0, W_stall=0,
clk=1, f_icode=1, f_ifun=0, f_rA=15, f_rB=4, f_valP=43, f_valC=5, D_icode=1, E_icode=3,
 M_icode=1, W_icode=1, mem_err=0, halt=0, 0=0, 1=1, 2=255, 3=5, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=5,m_valE=5,e_valE=5,M_valE=5

time=145,
 F_stall=0, W_stall=0,
clk=0, f_icode=1, f_ifun=0, f_rA=15, f_rB=4, f_valP=43, f_valC=5, D_icode=1, E_icode=3,
 M_icode=1, W_icode=1, mem_err=0, halt=0, 0=0, 1=1, 2=255, 3=5, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=5,m_valE=5,e_valE=5,M_valE=5
```

```
time=150,
 F_stall=0, W_stall=0,
clk=1, f_icode=1, f_ifun=0, f_rA=15, f_rB=4, f_valP=44, f_valC=5, D_icode=1, E_icode=1,
 M_icode=3, W_icode=1, mem_err=0, halt=0, 0=0, 1=1, 2=255, 3=5, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=5,m_valE=5,e_valE=5,M_valE=5

time=155,
 F_stall=0, W_stall=0,
clk=0, f_icode=1, f_ifun=0, f_rA=15, f_rB=4, f_valP=44, f_valC=5, D_icode=1, E_icode=1,
 M_icode=3, W_icode=1, mem_err=0, halt=0, 0=0, 1=1, 2=255, 3=5, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=5,m_valE=5,e_valE=5,M_valE=5

time=160,
 F_stall=0, W_stall=0,
clk=1, f_icode=1, f_ifun=0, f_rA=15, f_rB=4, f_valP=45, f_valC=5, D_icode=1, E_icode=1,
 M_icode=1, W_icode=3, mem_err=0, halt=0, 0=0, 1=1, 2=255, 3=5, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=5,m_valE=5,e_valE=5,M_valE=5

time=165,
 F_stall=0, W_stall=0,
clk=0, f_icode=1, f_ifun=0, f_rA=15, f_rB=4, f_valP=45, f_valC=5, D_icode=1, E_icode=1,
 M_icode=1, W_icode=3, mem_err=0, halt=0, 0=0, 1=1, 2=255, 3=5, 4=4, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=5,m_valE=5,e_valE=5,M_valE=5

time=170,
 F_stall=0, W_stall=0,
clk=1, f_icode=2, f_ifun=0, f_rA=4, f_rB=5, f_valP=47, f_valC=5, D_icode=1, E_icode=1,
 M_icode=1, W_icode=1, mem_err=0, halt=0, 0=0, 1=1, 2=255, 3=5, 4=5, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=5,m_valE=5,e_valE=5,M_valE=5

time=175,
 F_stall=0, W_stall=0,
clk=0, f_icode=2, f_ifun=0, f_rA=4, f_rB=5, f_valP=47, f_valC=5, D_icode=1, E_icode=1,
 M_icode=1, W_icode=1, mem_err=0, halt=0, 0=0, 1=1, 2=255, 3=5, 4=5, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=5,m_valE=5,e_valE=5,M_valE=5

time=180,
 F_stall=0, W_stall=0,
clk=1, f_icode=6, f_ifun=0, f_rA=3, f_rB=4, f_valP=49, f_valC=5, D_icode=2, E_icode=1,
 M_icode=1, W_icode=1, mem_err=0, halt=0, 0=0, 1=1, 2=255, 3=5, 4=5, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=5,m_valE=5,e_valE=5,M_valE=5

time=185,
 F_stall=0, W_stall=0,
clk=0, f_icode=6, f_ifun=0, f_rA=3, f_rB=4, f_valP=49, f_valC=5, D_icode=2, E_icode=1,
 M_icode=1, W_icode=1, mem_err=0, halt=0, 0=0, 1=1, 2=255, 3=5, 4=5, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=5,m_valE=5,e_valE=5,M_valE=5

processor.v:116: $finish called at 190 (1s)
time=190,
 F_stall=0, W_stall=0,
clk=1, f_icode=0, f_ifun=0, f_rA=3, f_rB=4, f_valP=50, f_valC=5, D_icode=6, E_icode=2,
 M_icode=1, W_icode=1, mem_err=0, halt=1, 0=0, 1=1, 2=255, 3=5, 4=5, 5=5, 6=6, 7=7, 8=8, 9=9, 10=10, 11=11, 12=12, 13=13, 14=14,W_valE=5,m_valE=5,e_valE=5,M_valE=5
```

Thank You