16 – bit RISC Pipelined Processor

1. Introduction

A synthesizable 32-bit pipeline processor is designed in VHDL and simulated in Altera Modelsim for the given instruction set. Separate instruction and data memory is used.

2. Six stages of Pipeline

The following are the stages in the pipeline design:

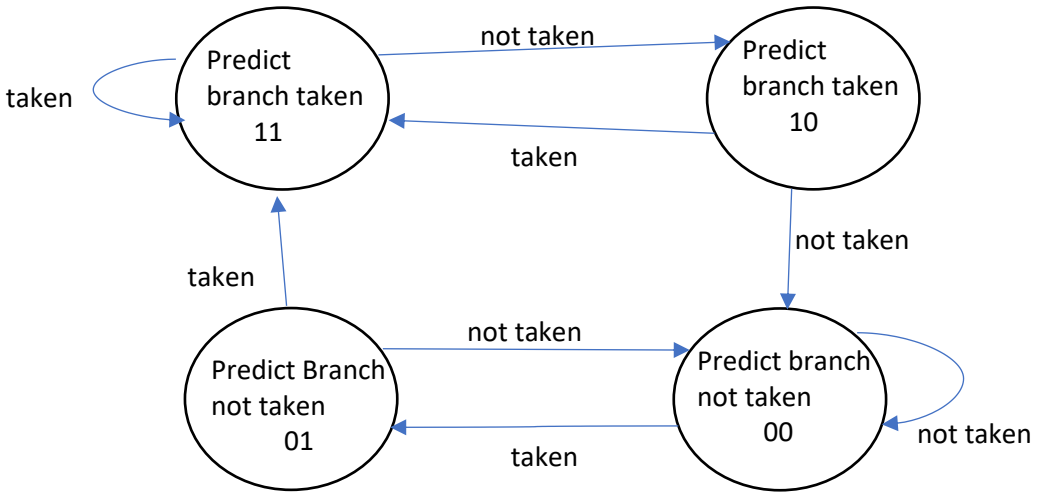
1. Instruction fetch
   It consists of ROM where instructions are stored. And a program counter which points to the next instruction address to be fetched.
2. Instruction decode
   This stage generates the control signals based on the instruction opcode received from the fetch stage. These signals are passed along the pipeline. It also consists of sign extender and an adder to add pc and the offset from instruction. Also generates stall signals in case of load multiple, store multiple or jump instruction.
3. Operand fetch
   It consists of a register file with eight 16-bit registers and register R7 i.e. 8^{th} register always holds the value of program counter. Register file has 2 ports to read the operands and one port to write to registers which is initiated by the writeback stage.
4. Execution
   All arithmetic and logical operations are done in this stage and updates the zero and carry flag based on the result generated. For load multiple and store multiple instructions we need to generate addresses of data memory to be accessed. An address generator module is implemented in this stage for this purpose.
5. Data memory read/write
   It uses a single port RAM to access data. If the instruction is not of memory access then it passes the result to writeback stage.
6. Writeback
   It consists of a mux which writes back to registers or to program counter in case for branch/jump instructions. It selects between the alu output, memory output or PC+1 according to the instruction.

3. Hazards prevention

To prevent data and control hazards, forwarding or bypassing along with stalls and flush signals is implemented.

4. Branch prediction

Branch prediction table is implemented to reduce the time for branching instructions (BEQ and JAL) by predicting the next address from the knowledge of history pattern. Initially history bit is set to 01. The table follows the below state diagram



Current address	Target address	History bit (2 bits)
x0001	x0005	01

5. Architecture of the processor

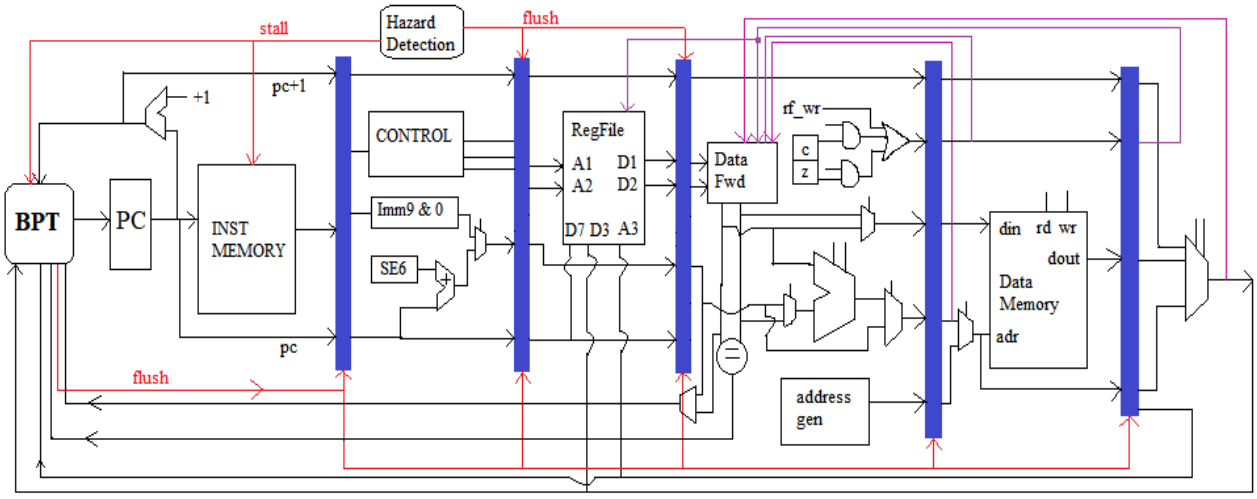


FIG 1. DATAPATH OF 16-BIT RISC PROCESSOR

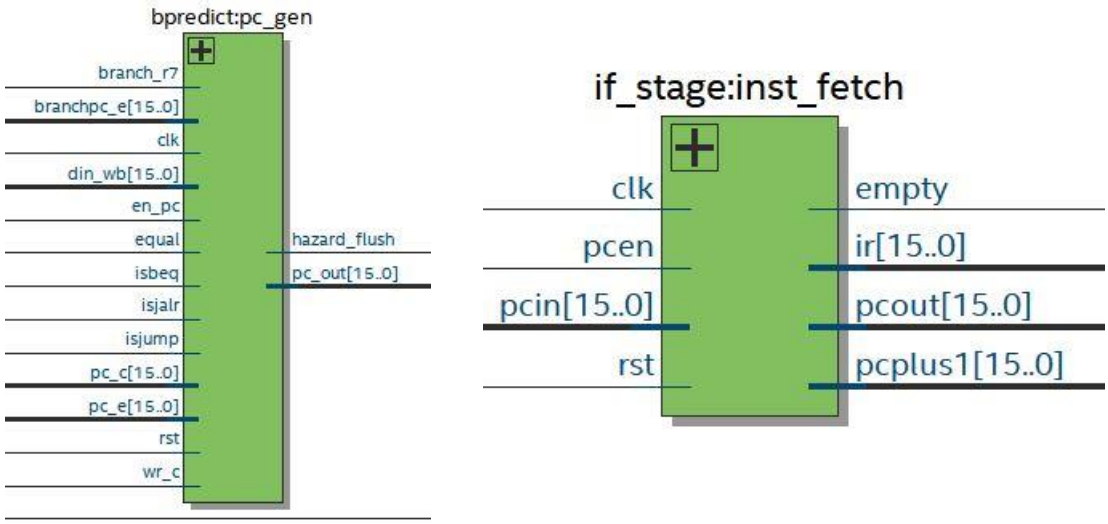


FIG 2. STAGE 1: BRANCH PREDICTION MODULE AND INSTRUCTION FETCH STAGE

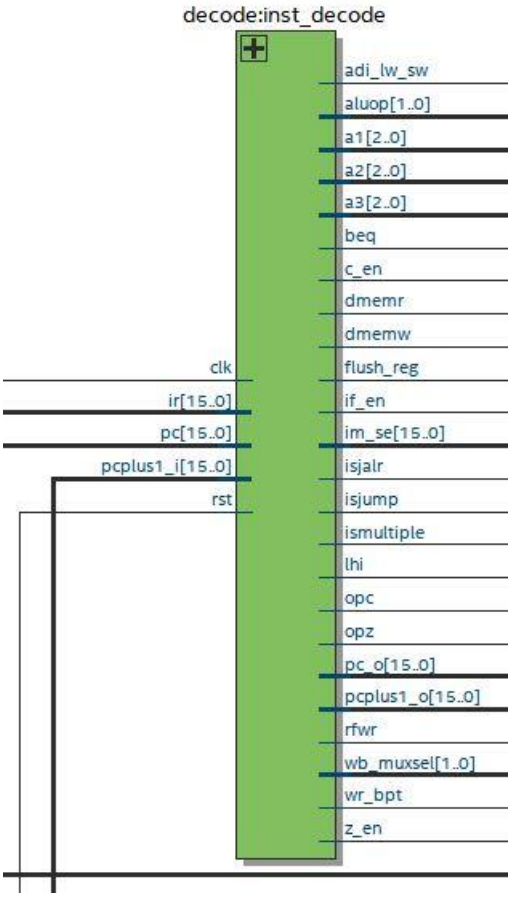


FIG 3. STAGE 2: CONTROL/DECODE STAGE

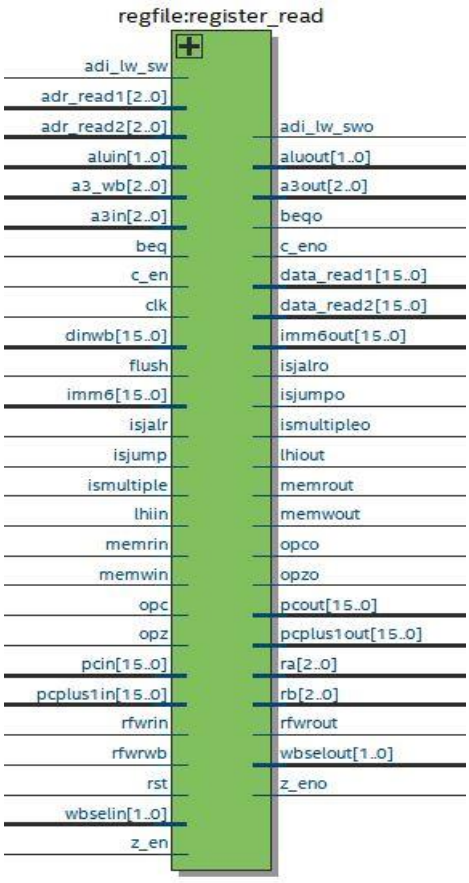


FIG 4. STAGE 3: REGISTER READ (OPERAND FETCH)

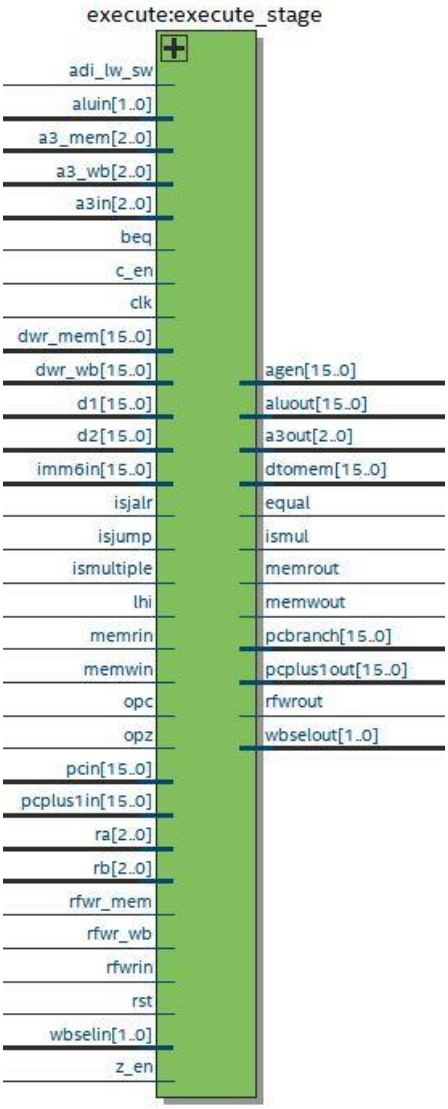


FIG 5. STAGE 4: EXECUTE MODULE

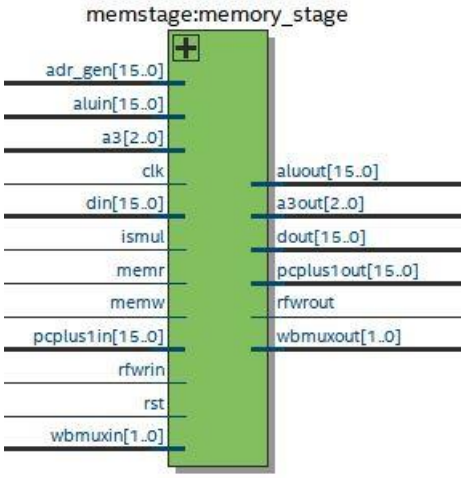


FIG 6. STAGE 5: DATA MEMORY READ

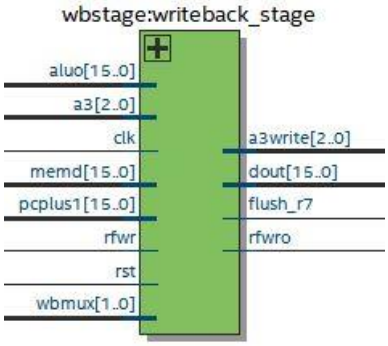


FIG 7. STAGE 6: WRITEBACK

6. Simulation results

FIG 8. SIMULATION WAVEFORMS FOR VARIOUS TEST CASES

FOLLOWING ARE THE CORRESPONDING TEST CASES FOR THE ABOVE FIGURES:

- 0 => "0011000000000001", --LHI R0 X0080
  1 => "0011001000000010", --LHI R1 X0100
  2 => "0011010000000100", --LHI R2 X0200
  3 => "0011011000001000", --LHI R3 X0400
  4 => "0011100000010000", --LHI R4 X0800
  5 => "0011101000100000", --LHI R5 X1000
  6 => "0011110001000000", --LHI R6 X2000

- 0 => "0011000000000001", --LHI R0 X0080
  1 => "1000001000100010", --JAL R1 X0020
  35 => "1100010011100000", --BEQ R2 R3 X0020

- 0 => "0101000010000000", --SW R0 R2+X00
  1 => "0000100101110000", --ADD R4 R5 R6
  2 => "0000001010011000", --ADD R1 R2 R3
  3 => "0000011110110010", --ADC R3 R6 R6
  4 => "0000001101110001", --ADZ R1 R5 R6
  5 => "0001100101000001", --ADI R4 R5 X01
  6 => "0100011010000000", --LW R3 R2+X00
  7 => "1100011100000000", --BEQ R3 R4 X00

- 0 => "0111000000111010", --SM R0 11111010

```
1 => "0000100101110000",      --ADD R4 R5 R6
2 => "0000001010011000",      --ADD R1 R2 R3
3 => "0000000110110010",      --ADC R3 R6 R6
4 => "0000001101110001",      --ADZ R1 R5 R6
5 => "0001100101000001",      --ADI R4 R5 X01
6 => "0100011010000000",      --LW R3 R2+X00
7 => "1100011100000000",      --BEQ R3 R4 X00
```

- ```
  x"3212",      --LHI
  x"3212",      --LHI
  x"3401",      --LHI
  x"029a",      --ADC
  x"5281",      --SW
  x"4881",      --LW
  x"0858",      --ADD
  x"8a05",      --JAL
  x"0000",      --ADD
  x"1234",      --ADI
  x"1234",      --ADI
  x"1234",      --ADI
  x"1d42",      --ADI
  x"7abf",      --SM
  x"7823",      --SM
  x"3001",      --LHI
  x"6a75",      --LM
  x"6c85",      --LM
  x"7a3a",      --SM
  x"c207",      --BEQ
  x"8af2",      --JAL
  x"3212",      --LHI
  ```
  ```