Project4 Report

Liu Yuheng

120090263

1. Big Pictures

The pipelined 5-stage CPU includes instruction fetch, instruction decode and register read, executing operation or calculate address, access memory operand, write the result back to register.

IF: in this stage finishes 2 operations. First, fetch instruction in memory to the instruction register. Second, plus program counter with 4 and store it into next program counter.

ID: in this stage finishes three operations. First, decode the instruction into different parts including operation code, function code and registers. Second, take values from the registers. Third, do sign extension which extends to 32 bits.

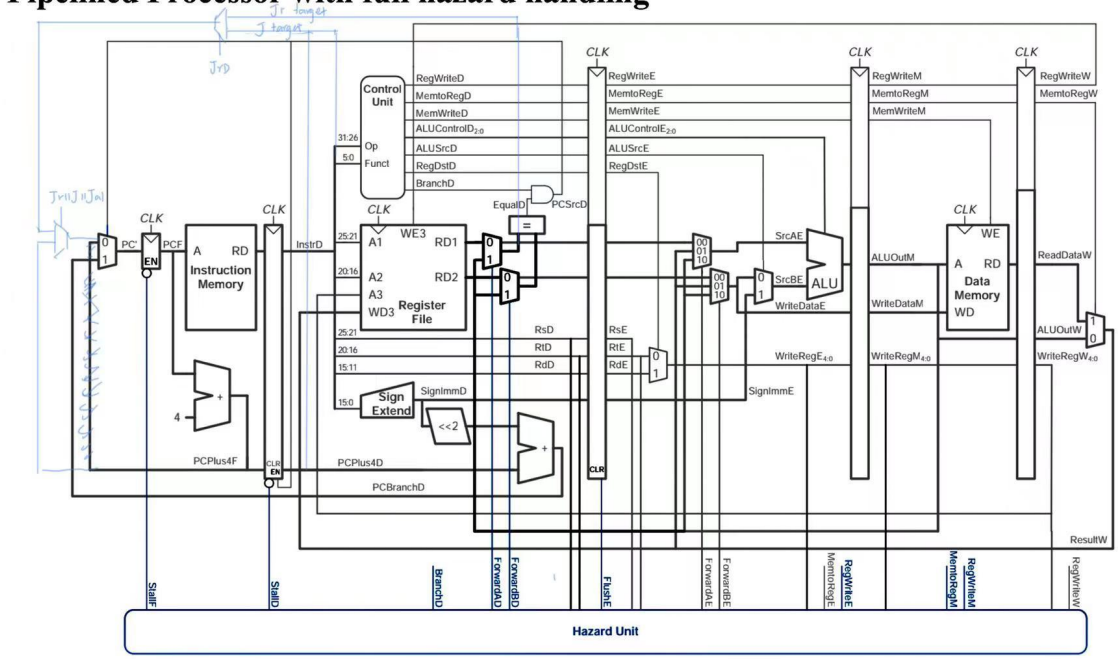
EX: in this stage, three operations may be conducted. First, do arithmetic calculating including addition, subtraction, multiplication, division. Or it calculates the memory address. If it is branch, it will calculate the target address and check the jump condition.

MEM: in this stage, three operations may be conducted. It gives the next PC to PC. It may conduct read and write operations according to the memory address calculated in the EX stage. If it is branch, it will assign the target address to PC according to conditions.

WB: in this stage, two operations may be conducted. It will write the arithmetic result back to registers. It writes back the data read from the memory.

2. Extension

Pipelined Processor with full hazard handling



3. High-Level Implementations

I break down the whole problem mainly according to different stages. In my program, the main files are IF_ID.v, ID_EX.v, EX_MEM.v, MEM_WB.v and PCReg.v. Each of these parts includes the pipeline register. I write some parts separately. For instance, ALU.v implements functions for arithmetic operations. Controlunit.v implements the control signals for different registers. There are three multiplexers separately dealing 1 bit, 5 bits, and 32 bits. Signextend,v implements extension to 32 bits. Then I wrote CPU.v which connects all parts together and defined wires to build the data path.

4. Implementation Details

About hazards. I tried to deal with branch and jump hazards in my program. At the start of CPU.v, I used one multiplexer to choose PC+4 and target address from EX stage. If it is a jump instruction, the next PC will be assigned with the target address. I also implemented PCBranch signals in control unit to indicate when encountering branches.

About negedge. In the RegisterFile.v, I used a negative edge in the always block for writing registers, simulating writing the data into registers in the second half of the clock.

5. Result

This program passed the first test successfully. It passed half of the fifth test and half of the sixth test case. There are some bugs in dealing with stall and flush, therefore the result still needs to improve to deal with hazards.

END OF REPORT