REPORT - 8

-SIDDHANT CHOUHDARY
-HARDIK AGGARWAL

The goal of the assignment was to implement the software for the pipeline model of a MIPS processor. In C++. The whole process could be divided into 5 stages – Instruction fetch, Instruction Decode, ALU, Memory access, Register write. For the implementation, we have used a loop for the process and each iteration in the the loop corresponds to one cycle. The instructions are fetched line by line from an input file and stored in a an array of strings.

1. In the instruction fetch state, an instruction is fetched from the block memory.

2. In the next iteration it goes to the decode stage where the opcodes and register values are updated. Two control signals mux_RegDst and mux_AluSrc are also updated in this stage.

3. In the ALU stage, the arithmetic and logical operations take place along with updating of mux_MemRead and mux_MemWrite signals.

4. In the memory stage, the data is either written to the memory or taken from the memory depending upon the value of control signals. mux_MemtoReg and mux_MemWrite control signals are updated here.

5. In the last stage, the data is written back to the destination register. Since the this happens in the first half of the cycle, the condition has also been placed near the start of the loop.
At the end of the loop the value of the pc is updated.

The final values stored in each of the registers is shown in the ouput along with the number of cycles involved as well as the IPC.

The pipeline model works quite efficiently but there may be some hazards depending upon the instruction types and register values which slow down the pipeline.

Data hazards occur when the new value for the register has not been updated by the previous instruction and the present instruction is using that register. Hence, this leads to a stall.
For data hazards, there is a stall of one cycle(1a and 1b type) or two cycle(2a and 2b type) and its been shown in the output along with the stage where the stall occurs during a particular cycle.

Control hazards occur when there is a prediction involved for instructions such as beq, bne etc. There has to be a stall one cycle minimum whenever the condition for branching is true. For this we have checked the condition for branching during the ID/EXE stage. The stall is represented as a bubble in the output.

TEST CASES -

```
bgtz $8 2 2
add $10 $9 $8 4
add $11 $8 $10 4
add $12 $9 $11 4
sll $13 $9 2 3
```

INTIAL VALUES -

```
reg[8]=8;
reg[9]=8;
reg[15]=15;
reg[10]=10;
reg[16]=16;
reg[18]=18;
ins[8]="13";
ins[9]="14";
ins[10]="10";
ins[14]="15"
```

OUTPUT -