IMT2022552-Shannon Muthanna IMT2022089-Aayush Bhargav

Non Pipelined and Pipelined Processor Implementation-Computer Architecture Report

Non Pipelined Processor:

We ask the user to enter 1 if he wants to sort numbers or 2 to find factorial.

We create dictionaries for opcodes, function fields , register file and register values.

We create a data memory to hold data and initialize output address as 50.($s3)

Depending on the user input, we read the corresponding machine code.

We initialize all the variables to be used by us for the program.

Clock_cycles holds the number of clock cycles taken by the program.

In the instruction fetch phase, we just fetch the instruction from the instruction memory location pointed to by the pc(program counter).

Next , in the instruction decode function, we decode the current instruction in the same way MIPS assembler does. We check if the instruction is r -type or I -type or j-type from the opcode of the instruction and set rs, rt, rd, imm values ,shift amount, function fields etc respectively.

In the execute function, we execute the decoded function. Depending on what type of instruction it is, we set the alu sources and perform the asked operation.In this function we handle branches and jumps too(by setting branch and jump flags to 1).(clearly commented in the code)

Next , in the memory access function, we write to the memory if it's a store word instruction or read from the memory if it's a load word instruction .

Next , in the write back phase , we write to the register file and the location we write to rt or rd depends on whether the instruction is an r type instruction or an I type instruction.

We run a while loop while pc(starting at 0) is less than the number of instructions present in the instruction memory. We call all our methods one by one in sequence and in each of those methods we increment out clock cycles by one.

After the write back function is called, we need to update our pc. So, we update it by depending on whether it's a branch or a jump or neither.

Then we output the data.

Images:

For sorting

```
"C:\New folder\python.exe" C:\Users\IIITB\PycharmProjects\pythonProject\main.py
Enter 1 for sorting,2 for factorial:
1
Enter the number of numbers to sort:
5
Enter the numbers:
1
2
3
4
5
Output sorted in descending order is:
5
4
3
2
1
Total number of clock cycles is:
1080

Process finished with exit code 0
```

For factorial:

```
"C:\New folder\python.exe" C:\Users\IIITB\PycharmProj
Enter 1 for sorting,2 for factorial:
2
Enter the number whose factorial has to be found:
7
Factorial of the number is:
5040
Total number of clock cycles is:
655
```

Pipelined Processor:

The non-pipelined processor takes one clock-cycle for each instruction, where as in the pipelined processor each phase takes one clock cycle. For the pipelined processor we have modified the non-pipelined code. Firstly we have made 5 pipelined registers to store the information we want in the next phase. Then we maintain 5 control signals ie IF,ID,EX,MEM and IF_status to check if each phase happend. Based on these signals we check if the next phase should happen or not. Example: Writeback should happen if Mem access has happend,Mem acces should happen if Ex has happend and so on.the variables a,b,c,d are flags that that tell these phases happend. Here are loop exits if an no-op instruction is read four times, four times because the previous instruction should have finished its WB phase.We also update the contents of the pipelined register in each cycle.

We have handeled structural hazards by writing back first and then reading. We have installed a forwardng unit and a hazard detection unit to handel data hazard. The forwardng unit handels data dependencies between two instruction. If the dependency is between two immediate instruction we forward from the EX phase. If the dependenciesis between load and some other instruction, we forward from MEM phase. If the dependencies is between one instruction and its next-to-next instruction, we forward from WB phase. Hazard detection unit detects the load hazard by creating a stall. We implement the stall by retaining the same pc,performing the same ID and IF, and making all control signals zero for the other phases. We have handeled Control hazards by flushing the instruction which was wrongly

fetched. Flushing is done by changing the IF and IF_status so that they never enter the if blocks of ID and IF in that cycle.As we check beq in EX stage, ID and IF should be flushed if wrogn instruction is fetched. Then in the next cycle we fetch the correct instruction. The pipelined processor is very efficient considered to the non-pipelined processor. The pipelined processor reduces the clock cycles by nearly four times.

Images:

Sorting

```
"C:\New folder\python.exe" C:\Users\IIITB\PycharmProjects\pythonProject\mips_pipeline.py
Enter 1 for sorting,2 for factorial:
1
Enter the number of numbers to sort:
5
Enter the numbers:
1
2
3
4
5
Output sorted in descending order is:
5
4
3
2
1
Total number of clock cycles is:
247

Process finished with exit code 0
```

Factorial:

```
"C:\New folder\python.exe" C:\Users\IIITB\PycharmProjects\pythonProject\mips_pipeline.py
Enter 1 for sorting,2 for factorial:
2
Enter the number whose factorial has to be found:
7
Factorial of the number is:
5040
Total number of clock cycles is:
151
```

You can clearly see the decrease in the number of clock cycles taken to complete the entire process.

Thank You