Eric Ponce

3/10/24

CSE 140

**HW #4**

2. Consider the data path and the component latency on page 4 of “CSE140\_Lecture-4\_Processor 3”.  
What is the execution time of add?

You would first fetch the instruction (I-Mem): 50ps

Then you would read the registers (Regs Rd): 35ps

Then you would use mux for “read data 2” (Mux): 5ps

Then you would use the ALU (ALU): 70ps

Then you would use mux again (Mux): 5ps

Lastly, you would write back to the destination register (Regs Wr): 35ps

The total would be 200ps

4. (zyBook) Exercise 4.19.16 (a)-(b)

In this exercise, we examine how pipelining affects the clock cycle time of the processor. Problems in this exercise assume that individual stages of the data path have the following latencies:

Latencies for IF, ID, EX, MEM, and WB stages. IF is 250 ps. ID is 350 ps. EX is 150 ps. MEM is 300 ps. WB is 200 ps.

Also, assume that instructions executed by the processor are broken down as follows:

Percentage of ALU/Logic, Jump/Branch, Load, and Store instructions. ALU/Logic are 45 percent. Jump/Branch are 20 percent. Load are 20 percent. Store are 15 percent.

(a) What is the clock cycle time in a pipelined and non-pipelined processor?

- non-pipelined clock cycle time = IF (250ps) + ID (350ps) + EX (150ps) + MEM (300ps) + WB (200ps) : 1250ps

- Pipelined clock cycle time = the stage with the highest latency which is ID: 350ps

(b) What is the total latency of a lw instruction in a pipelined and non-pipelined processor?

- If we are just running the single instruction, we still have to go through all 5 stages so the total latency would be 1250ps.

5. (zyBook) Exercise 4.19.20

Add NOP instructions to the code below so that it will run correctly on a pipeline that does not handle data hazards.

(I1) addi x11, x12, 5 (Adding 5 to register 12 and storing it in register 11)

NOP

NOP

NOP

NOP

(We need to updated value of x11 in the text instruction)

(I2) add x13, x11, x12 (Adding register 11 and register 12 together, storing in register 13)

(I3) addi x14, x11, 15 (Adding 15 to register 11 and storing in register 14)

NOP

NOP

NOP

(I4) add x15, x13, x12 (Adding registers 12 and 13 and storing in register 15)

Problem 6

Consider the following code.

addi x2, x2, 4 // I1  
lw x1, 0(x2) // I2  
addi x1, x1, 1 // I3  
sw x1, 0(x6) // I4  
addi x6, x6, 4 // I5

Assume that we have data forwarding paths EX/MEM to EXE and from MEM/WB to EXE. Fill  
the following timing diagram for the above code.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **CC1** | **CC2** | **CC3** | **CC4** | **CC5** | **CC6** | **CC7** | **CC8** | **CC9** | **CC10** | **CC11** | **CC12** |
| **I1** | IF | ID | EXE | MEM | WB |  |  |  |  |  |  |  |
| **I2** |  | IF | NOP | ID | EXE | MEM | WB |  |  |  |  |  |
| **I3** |  |  |  |  | IF | ID | EXE | MEM | WB |  |  |  |
| **I4** |  |  |  |  |  |  | IF | ID | EXE | MEM | WB |  |
| **I5** |  |  |  |  |  |  |  | IF | ID | EXE | MEM | WB |

Problem 7

Consider the following code.

I1: LOOP: lw x10, 0(x5)  
I2: slti x11, x10, 1  
I3: beq x11, zero, L1  
I4: addi x7, zero, 1  
I5: addi x5, x5, 8  
I6: sub x4, x10, x5  
I7: L1: bne x4, zero, LOOP  
I8: lw x9, 4(x5)  
I9: addi x8, x9, 8  
I10: sw x8, 0($t1)  
I11: add x8, x8, x6

Assume that we have data forwarding paths from EX/MEM to EXE stage and from  
MEM/WB to EXE stage. We do not use early branch determination. Branches are predicted  
untaken but the actual outcomes are as shown in the table. Show the first 18-cycles of  
execution of the code in a timing diagram shown below. When an instruction is stalled in  
a pipeline stage, fill the pipeline stage’s name for the instruction until the end of stall (as  
shown in the slide “CSE140\_Lecture-4\_Processor-4”). Draw additional rows if needed.  
Show instruction id to the first column.  
ID Branch instructions Branch outcome  
(NT: not taken, T: taken)  
I3 beq x11, zero, L1 NT at the first  
iteration  
T from the second  
iteration  
I9 bne x4, zero, LOOP T always

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **CC1** | **CC2** | **CC3** | **CC4** | **CC5** | **CC6** | **CC7** | **CC8** | **CC9** | **CC10** | **CC11** | **CC12** | **CC13** | **CC14** | **CC15** | **CC16** | **CC17** | **CC18** |
| **I1** | IF | ID | EXE | MEM | WB |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **I2** |  | IF | ID | EXE | MEM | WB |  |  |  |  |  |  |  |  |  |  |  |  |
| **I3** |  |  | IF | NOP | NOP | ID | EXE | MEM | WB |  |  |  |  |  |  |  |  |  |
| **I4** |  |  |  | IF | ID | EXE | MEM | WB |  |  |  |  |  |  |  |  |  |  |
| **I5** |  |  |  |  | IF | ID | EXE | MEM | WB |  |  |  |  |  |  |  |  |  |
| **I6** |  |  |  |  |  | IF | ID | EXE | MEM | WB |  |  |  |  |  |  |  |  |
| **I7** |  |  |  |  |  |  | IF | ID | EXE | MEM | WB |  |  |  |  |  |  |  |
| **I8** |  |  |  |  |  |  |  | IF | ID | EXE | MEM | WB |  |  |  |  |  |  |
| **I9** |  |  |  |  |  |  |  |  | IF | NOP | NOP | ID | EXE | MEM | WB |  |  |  |
| **I10** |  |  |  |  |  |  |  |  |  | IF | ID | EXE | MEM | WB |  |  |  |  |
| **I11** |  |  |  |  |  |  |  |  |  |  | IF | ID | EXE | MEM | WB |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |