भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Computer Architecture

Assignment -3 - CS2323

Kartik Srinivas - ES20BTECH11015
November 7, 2022

Index

1 Problem 1

The total time is added for both the read and write onto any intermediate register is assumed to be 12ps. Then the time for each of the stages will increase by 12ps. This means that the cycle time(T_c) =max(T_i) + T_{reg} = 145 + 12 = 157ps.

1.1 Part a

$$1000 \times T_c \ s = 157 \times 10^{-9} \ s = 157ns$$

1.2 Part b

A small assumption, since 1000 instructions is large, the time for the setup region of the pipelining process has been ignored. In case there are hazards in 20 % of the instructions which incur a one cycle stall. We can see this in two ways, either the CPI increases to 1 + 0.2 or we can calculate the number of cycles as follows:

$$T_c \times 1000 \times cpi = 157 * (1 + 0.2 * 1) = 188.4 \ ns$$

The second way is to calculate the number of cycles.

$$1000 * 0.2 * 1 + 1000 = 1200 \text{ cycles}, T = 1200 * 157 \ ps = 188.4 \ ns$$

2 Problem 2

The time must be sufficient for the longest delay in the pipelined system, as the ALU latency increases by 250 ps the answer will be

$$800/5 + 250 = 250 + 160 \ ps = 410 \ ps$$

The processor P1 and P2 will excute x and $9x/10$ number of instructions. P1 would take 160 ps per cycle of instruction and P2 would take 410 ps , therefore

$$T_{p2} = 410 \times 9x/10 > 160 \times x = T_{p1}$$

So P2 would be slower than P1

If a piece of code has 5000 instructions on P1 , then the number of instructions on P2 = 4500 so the execution times are

$$T_{p1} = 160 \times 5000 \ ps = 80e4 \ ps$$

$$T_{p2} = 410 \times 4500 \ ps = 184.5e4 \ ps$$

3 Problem 3

The code after inserting nops is as follows, The **add instruction will add two nops and load use will add 3 nops.** However the load instruction has a conflict on the next to next instruction, so only 2 nops are required for the load one too. The total number of Nops = 6.

3.1 Without Forwarding

```
add x14 x12 x11
addi x0 x0 0 # nop
addi x0 x0 0 # nop
add x15 x14 x12
ld  x13 8(x13)
ld  x12  0(x14) # fine , no update 2 steps above
addi x0 x0 0 #nop because x13 has been updated above
addi x0 x0 0 #nop because x13 has been updated above
and x13 x15 x13
addi x0 x0 0 # nop
addi x0 x0 0 # nop
ld x11 4(x13)
sd x13 0(x15)
```

3.2 With Forwarding

In case I use forwarding then the number of nops required for simple data hazards is zero , while as a load use hazard will take a single nop, (btu that is already covered under an existing instruction)

```
add x14 x14 x11
add x15 x14 x12
ld x13 8(x13)
ld x12 0(x14) # effectively behaves like a nop to x13
and x13, x15, x13
ld x11, 4(x13) # no nop
sd x13, 0(x15)
```

So **NO Nops** will be required with forwarding in this case.