Assignment 4

4.8

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

IF	ID	EX	MEM	WB
250ps	350ps	150ps	300ps	200ps

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

alu	beq	lw	sw
45%	20%	20%	15%

4.8.1

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

Piplined version

The clock cycle time is equal to the slowest instruction. Therefore, the clock cycle time is `350ps`

Non-piplined version

The instruction will need all 5 stages to execute. The lw instruction needs all 5 stages to execute. Therefore, the clock cycle time of lw instruction is

$$250 + 350 + 150 + 300 + 200 = 1250ps$$

So the clock cycle time is `1250ps`

4.8.2

What is the total latency of an LW instruction in a pipelined and non-pipelined processor?

Piplined version

The latency is `1250ps`

Non-piplined version

The latency is 5 × 350 = `1750ps`

4.8.3

If we can split one stage of the pipelined datapath into two new stages, each with half the latency of the original stage, which stage would you split and what is the new clock cycle time of the processor?

Because the cycle time is due to the slowest instruction, if we split any stage other than ID, the cycle time will remain the same.

Thus, we split `ID stage`, so tach new stage takes 350 ÷ 2 = 170ps

The latencies will be:

IF	ID1	ID2	EX	MEM	WB
250ps	175ps	175ps	150ps	300ps	200ps

So now the new clock cycle time is `300ps` (The duration of MEM stage)

4.8.4

Assuming there are no stalls or hazards, what is the utilization of the data memory?

The data memory is used only for `sw` and `lw` instrucions.

Therefore, its utilization is `20% + 15% = 35%`

4.8.5

Assuming there are no stalls or hazards, what is the utilization of the write-register port of the “Registers” unit?

This port is used only for `alu` and `lw` instructions.

Therefore, its utilization is `45% + 20% = 65%`

4.8.6

Instead of a single-cycle organization, we can use a multi-cycle organization where each instruction takes multiple cycles but one instruction fnishes before another is fetched. In this organization, an instruction only goes through stages it actually needs (e.g., ST only takes 4 cycles because it does not need the WB stage). Compare clock cycle times and execution times with single-cycle, multi-cycle, and pipelined organization.

Suppose the instruction number is N, the execution time is T

Single-cycle version

The clock cycle time of single-cycle version is `1250ps`

$$T_{single-cycle} = 1250 \times N$$

Piplined version

The clock cycle time of single-cycle version is `350ps`

$$T_{pipline} = 350 \times N$$

Multi-cycle version

The clock cycle time of multi-cycle version is still `350ps`

Because `lw` instructions take 5 cycles, while all other instructions take 4 cycles.

So the execution time is

$$T_{multi} = 0.2 \times N \times 5 \times 350 + 0.8 \times N \times 4 \times 350 = 1470 \times N$$

So we have

	SINGLE-CYCLE VERSION	PIPLINED VERSION	MULTI-CYCLE VERSION
clock cycle time	1250ps	350ps	350ps
execution time	1250N	350N	1470N

Compare:

clock cycle time

$$\frac{Clock\ cycle_{single}}{Clock\ cycle_{piplined}} = \frac{1250}{350} = 3.57$$

$$\frac{Clock\ cycle_{multi}}{Clock\ cycle_{piplined}} = \frac{350}{350} = 1$$

execution time

$$\frac{T_{single}}{T_{piplined}} = \frac{1250N}{350N} = 3.57$$

$$\frac{T_{multi}}{T_{piplined}} = \frac{1470}{350N} = 4.2$$

4.9

In this exercise, we examine how data dependences af ect execution in the basic 5-stage pipeline described in Section 4.5. Problems in this exercise refer to the following sequence of instructions:

```
1 or r1,r2,r3
2 or r2,r1,r4
3 or r1,r1,r2
```

Also, assume the following cycle times for each of the options related to forwarding:

Without Forwarding	With Full Forwarding	With ALU-ALU Forwarding Only
250ps	300ps	290ps

4.9.1

Indicate dependences and their type.

---

1. The first dependency is that the first instruction changes `r1` while the second instrution need `r1` to execute. This is a type `1a-data hazard`
2. The second dependency is that `r1` is also needed for the third instruction, so this is a type `2a-data hazard`
3. The third dependency is that r2 iss needed for the third execution, and we set r2 in the second instruction, this is a type `1b-data hazard`

4.9.2

Assume there is no forwarding in this pipelined processor. Indicate hazards and add `nop` instructions to eliminate them.

Since the dependency between the first and the second instruction causes a type 1 hazard, we must add three nops between them (because the second instruction cannot enter the ID phase until the correct values of registers are put in place that is, after the first instruction finishes its WB phase).

The same as second and the third instruction. Notice that the addition of three nops between the second and the third instruction also takes care of the type 2 hazard caused by the dependency between the first and the third instruction.

The modified code is as follows

```
or r1, r2, r3
nop
nop
nop
or r2, r1, r4
nop
nop
nop
or r1, r1, r2
```

4.9.3

Assume there is full forwarding. Indicate hazards and add `nop` instructions to eliminate them.

We can dont'e need any `nop`

Forwording can solve the value passed in each instruction.

4.9.4

What is the total execution time of this instruction sequence without forwarding and with full forwarding? What is the speedup achieved by adding full forwarding to a pipeline that had no forwarding?

Without forwarding

The first instuction needs 5 cycles to complete, each subsequent instrucion need more 1 addition cycle.

Without forwarding, we use `5 + 8 = 13 cycles`

The execution time is `13 × 250 = 3250ps`

Full forwarding

With full forwarding, we use `5 + 2 = 7 cycles`

The execution time is `7 × 300 = 2100ps`

So the speedup is $\frac{3250}{2100} = 1.55$

4.9.5

Add `nop` instructions to this code to eliminate hazards if there is ALU-ALU forwarding only (no forwarding from the MEM to the EX stage).

From the explanation of 3., it is apparent that we must add at least two nop between the second and the third instruction since we cannot forward r1 anymore.

Therefore, the code is:

```
or r1, r2, r3
or r2, r1, r4
nop
nop
or r1, r1, r2
```

4.9.6

What is the total execution time of this instruction sequence with only ALU-ALU forwarding? What is the speedup over a no-forwarding pipeline?

We have `5 + 4 = 9 cycles` in this case.

The total execution time is `9 × 290 = 2610ps`

So the speedup is $\frac{3250}{2610} = 1.245$