Chapter 4 Exercises

4.5 For the problems in this exercise, assume that there are no pipeline stalls and that the breakdown of the executed instructions is as follows:

add	addi	not	beq	lw	sw
20%	20%	0%	25%	25%	10%

4.5.1 [10] ⟨4.3⟩ In what fraction of cycles is the data memory used?

4.8 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 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 [5] ⟨4.5⟩ What is the clock cycle time in a pipelined and non-pipelined processor?

(a) All pipeline stages take a single clock cycle, so the clock cycle must be long enough to accommodate the slowest operation. In this case the slowest operation is the instruction decode (ID) at 350ps. Therefore the pipelined clock cycle time is 350ps.

(b) In a non-pipelined processor, each instruction executes after the other, one at a time. To complete one instruction, the time for each operation is added together, for the worst case instruction.

$250\text{ps} + 350\text{ps} + 150\text{ps} + 300\text{ps} + 200\text{ps} = 1250\text{ps}$

4.8.2 [10] ⟨4.5⟩ What is the total latency of an LW instruction in a pipelined and non-pipelined processor?

(a) The `lw` instruction uses every stage in the execution process (IF, ID, EX, MEM and WB). For a pipelined processor, each operation takes the same amount of time, in this case 350ps. So the total latency of the `lw` instruction is $350\text{ps} \times 5 = 1750\text{ps}$

(b) For a non-pipelined processor, stages are allowed to take different durations so in the non-pipelined example the total latency for `lw` is the sum of each stage's execution time.

$250\text{ps} + 350\text{ps} + 150\text{ps} + 300\text{ps} + 200\text{ps} = 1250\text{ps}$

4.11 Consider the following loop:

```
loop:
    lw r1, 0(r1)
    and r1, r1, r2
    lw r1, 0(r1)
    lw r1, 0(r1)
    beq r1,r0, loop
```

Assume that perfect branch prediction is used (no stalls due to control hazards), that there are no delay slots, and the pipeline has full forwarding support. Also, assume that many iterations of this loop are executed before the loop exists.

4.11.1 [10] ⟨4.6⟩ Show a pipeline execution diagram for the third iteration of this loop, from the cycle in which we fetch the first instruction up to (but not including) the cycle in which we can fetch the first instruction of the next iteration. Show all instructions that are in the pipeline during these cycles (not just those from the third iteration.

4.13 This exercise is intended to help you understand the relationship between forwarding, hazard detection, and ISA design. problems in this exercise refer to the following sequence of instructions and assume that it is executed of a 5-stage pipelined datapath:

```
add r5, r2, r1
lw r3, 4(r5)
lw r2, 0(r2)
or r3, r5, r3
sw r3, 0(r5)
```

4.13.1 [5] ⟨4.5⟩ If there is no forwarding or hazard detection, insert nops to ensure correct execution.

4.13.2 [10] ⟨4.5⟩ How many cycles in total are needed to complete those instructions (including the nops that you inserted?) If regular forwarding is implemented, how many cycles are in the revised code in total?