CS202 HW3

Problem 1 20points

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
200ps	170ps	220ps	210ps	150ps

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

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

(c) 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?

The remaining problems in this exercise assume that instructions executed by the processor are broken down as follows:

ALU	BEQ	LW	SW
55%	15%	15%	15%

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

(e) Assuming there are no stalls or hazards, what is the utilization of the write-register port of the "Registers" unit?

(f) Instead of a single-cycle organization, we can use a multi-cycle organization, where each instruction takes multiple cycles but one instruction finishes before another is fetched. In this organization, an instruction only goes through stages it actually needs (e.g., Branch only takes 3 cycles because it does not need the Mem/WB stage). Compare clock cycle times and execution times with single-cycle, multi-cycle, and pipelined organization.

Problem 2 15points

Consider a version of the pipeline that does not handle data hazards (i.e., the programmer is responsible for addressing data hazards by inserting NOP instructions where necessary). Suppose that (after optimization) a typical n-instruction program requires an additional 5*n NOP instructions to correctly handle data hazards.

(a) Suppose that the cycle time of this pipeline without forwarding is 250 ps. Suppose also that adding forwarding hardware will reduce the number of NOPs from 5*n to 0.05*n, but increase the cycle time to 275 ps. What is the speedup of this new pipeline compared to the one without forwarding?

(b)Under the conditions specified in part (a), assume the following:

- The **pipeline without forwarding** contains `n` instructions and `5n` NOP instructions, with a cycle time of **250 ps**.
- The **pipeline with forwarding** contains `n` instructions (no initial NOPs), with a cycle time of **275 ps**.

How many **additional NOP instructions** (expressed as a multiple of `n`, e.g., `k*n`) can be introduced in the pipeline with forwarding such that its total execution time is still **less than or equal to** the original pipeline without forwarding?

Problem 3 20points

Given the following instruction sequence, answer the questions below. Assume the instructions are executed on a **5-stage pipelined datapath** that **does NOT handle data hazards** but **does handle structural hazards**.

```
add x5, x2, x1
lw x6, 4(x5)
lw x2, 0(x2)
add x3, x5, x1
sw x3, 0(x5)
```

(a) Insert NOP instructions into the code below to ensure correct execution on a pipeline that does NOT handle data hazards. Finally, calculate the total number of clock cycles required to execute the modified code. You need to refer to the example format below.

```
inst xx
NOP
inst xx
inst xx
NOP
NOP
...
```

(b) Beside adding NOP, Re-solve the problem by reordering the instructions to minimize stalls. Calculate the total number of clock cycles required after reordering.

Problem 4 15points

Given the following code sequence, assume it is executed on a **5-stage pipelined datapath** and the processor **has all possible forwarding paths** between stages.

```
add x15, x12, x11
lw x13, 4(x15)
lw x12, 0(x2)
or x13, x15, x13
sw x12, 0(x15)
```

Draw a pipeline diagram (using the table format shown below) to indicate :

- **Stalls** (if any) with bubbles (`*`).
- **Forwarding paths** by annotating the relevant stages (e.g., `Fwd EX→EX` or `Fwd MEM→EX`).

Then **Count the number of times each type of forwarding path is used** (e.g., EX→EX, MEM→EX).

	t1	t2	t3	t4	t5	t6	t7	t8
inst xx	IF	ID	EX	MEM	WB			
inst xx		*	IF	ID	EX	MEM	WB	
inst xx				IF	ID	EX	MEM	WB
...								

Problem 5 30Points

In this exercise we compare the performance of 1-issue and 2-issue processors, taking into account program transformations that can be made to optimize for 2-issue execution. Problems in this exercise refer to the following loop (written in C), where a and b are char array:

```
for(int i=0;i!=j;i+=2){
    b[i] = a[i] - a[i+1]
}
```

A compiler doing little or no optimization might produce the following RISC-V assembly code:

```
    addi x12, x0, 0
    jal ENT
TOP:add x6, x10, x12
    lbu x7, 0(x6)
    lbu x19, 1(x6)
    sub x20, x7, x19
    add x21, x11, x12
    sb x20, 0(x21)
    addi x12, x12, 2
ENT:bne x12, x13, TOP
```

The code above uses the following registers:

i	j	a	b	Temporary values
x12	x13	x10	x11	x5-x7, x19-x21

Assume the two-issue, statically scheduled processor for this exercise has the following properties:

- One instruction must be a memory operation; the other must be an arithmetic/logic instruction or a branch.

- The processor has all possible forwarding paths between stages (including paths to the ID stage for branch resolution).
- The processor has perfect branch prediction.
- Two instruction may not issue together in a packet if one depends on the other.
- If a stall is necessary, **both instructions** in the issue packet must stall.
- For speedup calculations (parts b and e), ignore `addi x12, x0, 0` and `jal ENT`. Assume `j` is very large, and count the cycles between two consecutive `bne` instructions.

(a) Using the table format shown in Figure 4.67 in textbook to draw a code scheduling, showing how RISC-V code given above executes on the two-issue processor. Assume that the loop exits after two iterations.

(b) What is the approximate speedup of going from a one-issue to a two issue processor? (Assume the loop runs a large amount of iterations.)

(c) Rearrange the provided RISC-V code to optimize performance on the two-issue processor (Do not unroll the loop) . Assume that the loop exits after two iterations. Using the table format shown in Figure 4.67 in textbook to draw a code scheduling for the reordered code.

(d) What is the approximate speedup of going from a one-issue processor to a two-issue processor, when running the rearranged code from the previous question. (Assume the loop runs a large amount of iterations.