Computer Organization HW3

1. (ref: textbook Exercise 4.16) 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
250 ps	350 ps	150 ps	300 ps	200 ps

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

ALU/Logic	Branch	Load	Store
45%	20%	20%	15%

   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?
   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.

2. (ref: textbook Exercise 4.21) 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 4*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 4*n to 0.05*n, but increase the cycle time to 300 ps. What is the speedup of this new pipeline compared to the one without forwarding?
   b) Different programs will require different amounts of NOPs. How many NOPs (as a percentage of code instructions) can remain in the typical program before that program runs slower on the pipeline with forwarding? (题目意思为：在不影响程序性能的条件下，在带有 forwarding 的 pipeline 中执行程序，程序中的 NOP 指令比例最多可以为多少?即执行时间仍能少于没有 forwarding 的 pipeline)

3. (ref: textbook Exercise 4.22) Consider the fragment of RISC-V assembly below:

```
sw    x29, 12(x16)
lw    x29, 8(x16)
sub   x17, x15, x14
beqz  x17, label
add   x15, x11, x14
sub   x15, x30, x14
```

   Suppose we modify the pipeline so that it has only one memory (that handles both instructions and data). In this case, there will be a structural hazard every time a program needs to fetch an instruction during the same cycle in which another instruction accesses data.

a) Draw a pipeline diagram to show were the code above will stall.

b) In general, is it possible to reduce the number of stalls/NOPs resulting from this structural hazard by reordering code?

c) Must this structural hazard be handled in hardware? We have seen that data hazards can be eliminated by adding NOPs to the code. Can you do the same with this structural hazard? If so, explain how. If not, explain why not.

4. (ref: textbook Exercise 4.31) 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):

```
for(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:

```
        li x12, 0
        jal ENT
TOP:    slli x5, x12, 2
        add x6, x10, x5
        lw x7, 0(x6)
        lw x29, 4(x6)
        sub x30, x7, x29
        add x31, x11, x5
        sw x30, 0(x31)
        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, x29–x31

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.

a) Draw a pipeline diagram 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 speedup of going from a one-issue to a two issue processor? (Assume the loop runs thousands of iterations.)

c) Rearrange/rewrite the RISC-V code given above to achieve better performance on the one-issue processor. Hint: Use the instruction “beqz x13,DONE” to skip the loop entirely if j = 0.

d) Rearrange/rewrite the RISC-V code given above to achieve better performance on the two-issue processor. (Do not unroll the loop, however.)

e) What is the speedup of going from a one-issue processor to a two-issue processor when running the optimized code from the previous question.