CS214 Computer Organization HW#3

Problem 1.

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%

1. What is the clock cycle time in a pipelined and non-pipelined processor?
2. What is the total latency of an LW instruction in a pipelined and non-pipelined processor?
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?
4. Assuming there are no stalls or hazards, what is the utilization of the data memory? The utilization of a resource is defined as the duration the resource is used over the total time.
5. Assuming there are no stalls or hazards, what is the utilization of the write-register port of the “Registers” unit?
6. 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., 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.

Problem 2.

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

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

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

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

3. 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?

4. 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).

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

Problem 3.

In this exercise, we examine how resource hazards, control hazards, and Instruction Set Architecture (ISA) design can affect pipelined execution. Problems in this exercise refer to the following fragment of MIPS code:

```
sw r16,12(r6)
lw r16,8(r6)
beq r5,r4,Label # Assume r5!=r4
add r5,r1,r4
slt r5,r15,r4
```

Assume that individual pipeline stages have the following latencies:

IF	ID	EX	MEM	WB
200ps	120ps	150ps	190ps	100ps

1. For this problem, assume that all branches are perfectly predicted (this eliminates all control hazards) and that no delay slots are used. If we only have one memory (for both instructions and data), there is a structural hazard every time we need to fetch an instruction in the same cycle in which another instruction accesses data. To guarantee forward progress, this hazard must always be resolved in favor of the instruction that accesses data. What is the total execution time of this instruction sequence in the 5-stage

pipeline that only has one memory? We have seen that data hazards can be eliminated by adding nops to the code. Can you do the same with this structural hazard? Why?

2. For this problem, assume that all branches are perfectly predicted (this eliminates all control hazards) and that no delay slots are used. If we change load/store instructions to use a register (without an of set) as the address, these instructions no longer need to use the ALU. As a result, MEM and EX stages can be overlapped and the pipeline has only 4 stages. Change this code to accommodate this changed ISA. Assuming this change does not affect clock cycle time, what speedup is achieved in this instruction sequence?

3. Assuming stall-on-branch and no delay slots, what speedup is achieved on this code if branch outcomes are determined in the ID stage, relative to the execution where branch outcomes are determined in the EX stage?

4. Given these pipeline stage latencies, repeat the speedup calculation from question 2, but take into account the (possible) change in clock cycle time. When EX and MEM are done in a single stage, most of their work can be done in parallel. As a result, the resulting EX/MEM stage has a latency that is the larger of the original two, plus 20 ps needed for the work that could not be done in parallel.

5. Given these pipeline stage latencies, repeat the speedup calculation from question 3, taking into account the (possible) change in clock cycle time. Assume that the latency ID stage increases by 50% and the latency of the EX stage decreases by 10ps when branch outcome resolution is moved from EX to ID.

6. Assuming stall-on-branch and no delay slots, what is the new clock cycle time and execution time of this instruction sequence if beq address computation is moved to the MEM stage? What is the speedup from this change? Assume that the latency of the EX stage is reduced by 20 ps and the latency of the MEM stage is unchanged when branch outcome resolution is moved from EX to MEM.

Problem 4.

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];
```

When writing MIPS code, assume that variables are kept in registers as follows, and that all registers except those indicated as Free are used to keep various variables, so they cannot be used for anything else.

i	j	a	b	c	Free
R5	R6	R1	R2	R3	R10, R11, R12

1. Translate this C code into MIPS instructions. Your translation should be direct, without rearranging instructions to achieve better performance.
2. If the loop exits after executing only two iterations, draw a pipeline diagram for your MIPS code from question 1 executed on a 2-issue processor shown in Figure 4.69. Assume the processor has perfect branch prediction and can fetch any two instructions (not just consecutive instructions) in the same cycle.
3. Rearrange your code from question 1 to achieve better performance on a 2-issue statically scheduled processor from Figure 4.69.
4. Repeat question 2, but this time use your MIPS code from question 3.
5. What is the speedup of going from a 1-issue processor to a 2-issue processor from Figure 4.69?