CSE 140 HW #2

Solve the following problems and place your answers in a text document for submission.

1. In this exercise we examine in detail how an instruction is executed in a single-cycle datapath. Problems in this exercise refer to a clock cycle in which the processor fetches the following instruction word:

   10101100011000100000000000010100

   Assume that data memory contains all zeros and that the processor's registers have the following values at the beginning of the cycle in which the above instruction word is fetched:

R0	R1	R2	R3	R4	R5	R6	R8	R12	R31
0	-1	2	-3	-4	10	6	8	2	-16

   a. What are the outputs of the sign-extend and the jump "Shift left 2" unit (near the top of Figure 4.4.11 (The simple control and datapath are extended to handle the jump instruction)) for this instruction word?

   b. What are the values of the ALU control unit's inputs for this instruction?

   c. What is the new PC address after this instruction is executed? Highlight the path through which this value is determined.

   d. For each Mux, show the values of its data output during the execution of this instruction and these register values.

   e. For the ALU and the two add units, what are their data input values?

   f. What are the values of all inputs for the "Registers" unit?

2. 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%

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., 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.

3. In this exercise, we examine how data dependences affect execution in the basic 5-stage pipeline described in Chapter 4.5 (An overview of pipelining). 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

a. Indicate dependences and their type.

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

c. Assume there is full forwarding. Indicate hazards and add NOP instructions to eliminate them.

d. 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?

e. 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).

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

4. 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

a. 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?

b. 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 offset) 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?

c. 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?

d. Given these pipeline stage latencies, repeat the speedup calculation from **question b**, 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 20ps needed for the work that could not be done in parallel.

e. Given these pipeline stage latencies, repeat the speedup calculation from **question c**, 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.

f. 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 20ps and the latency of the MEM stage is unchanged when branch outcome resolution is moved from EX to MEM.