Assignment 3

Problem 1

1.What is the clock cycle time in a pipelined and non-pipelined processor?

A: pipelined clock cycle time: 350 ps, non-pipelined clock cycle time:
$250 + 350 + 150 + 300 + 200 = 1250$ ps.

2.What is the total latency of an LW instruction in a pipelined and non-pipelined processor?

A: total latency of LW in pipelined: $5 * 350 = 1750$ ps, total latency of LW in non-pipelined:1250 ps.

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?

A: I would split **ID** stage, because the clock cycle time in pipelined processor depends on the longest cycle time of all stages. So, after spliting ID stage into two parts, the clock cycle time will decrease to 300 ps.

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.

A: Only LW and SW use the data source , so the utilization of data memory is $20\% + 15\% = 35\%$.

5.Assuming there are no stalls or hazards, what is the utilization of the write register port of the “Registers” unit?

A: Only LW and ALU use the write register port of the "Registers" unit, so the utilization of the write register port of the “Registers” unit is $45\% + 20\% = 65\%$.

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.

A: clock cycle time of single-cycle is 1250 ps and clock cycle time of pipelined is 350 ps, which have been calculated before. And according to the definition of multi-cycle, it has the same clock cycle time as pipelined, that is 350 ps. In pipelined organization without hazards, each cycle completes one instruction. In single-cycle organization, each cycle also completes one instruction. So, the execution time of single-cycle is $1250/350 \approx 3.6$ times as that of pipelined. According to the definition, we find that only LW needs 5 cycles, and all of other type instructions require only 4 cycles(alu without MEM, beq without WB, sw without WB). So, the execution time of multi-cycle is $(45\% + 20\% + 15\%) \times 4 + 20\% \times 5 = 4.2$ times as that of pipelined, as they share the same clock cycle time.

Problem 2

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

A: The destination register of $I1$ is used as the resource register of $I2$,and the destination register of $I2$ is used as the resource register of $I3$. So, there are data hazards between $I1$ and $I2$, $I2$ and $I3$. So, add nop instructions to eliminate them. Like this, $I1\ nop\ nop\ I2\ nop\ nop\ I3$.

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

A: With full forwarding, we can add bypassing line between EX of $I1$ to EX of $I2$ and EX of $I2$ to EX of $I3$. Therefore, there is no hazard and no need to add nop instructions.

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?

A: Without forwarding: $(5+1+1+2\times 2)\times 250=2750$ ps. With full forwarding: $(5+1+1)\times 300=2100$ ps. The speedup is $2750/2100\approx 1.31$.

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).

A: According the analysis in sub-problem 2, ALU-ALU forwarding is just what we want. There is also no hazard and no need to add nop instructions.

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?

A: With only ALU-ALU forwarding: $(5+1+1)\times 290=2030$ ps. The speedup over no-forwarding pipeline:$2750/2030\approx 1.35$.

Problem 3

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?

A: The clock cycle time in this pipeline is 200 ps.

Instruction	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
sw	IF	ID	EX	**MEM**	WB						
lw		IF	ID	EX	**MEM**	WB					
beq			IF	ID	EX	MEM	WB				
add						IF	ID	EX	MEM	WB	
slt							IF	ID	EX	MEM	WB

So, to avoid MEM structural hazards, we need totally 11 cycles to complete these instructions. And the total execution time is $11 \times 200 = 2200$ ps. We can't use nops to eliminate the hazard as nop also needs to be fetched which will use memory.

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?

A: The clock cycle is also 200 ps.

Original with five stages:

Instruction	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
sw	IF	ID	EX	MEM	WB				
lw		IF	ID	EX	MEM	WB			
beq			IF	ID	EX	MEM	WB		
add				IF	ID	EX	MEM	WB	
slt					IF	ID	EX	MEM	WB

After change:

Instruction	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
sw	IF	ID	EX/MEM	WB				
lw		IF	ID	EX/MEM	WB			
beq			IF	ID	EX/MEM	WB		
add				IF	ID	EX/MEM	WB	
slt					IF	ID	EX/MEM	WB

Therefore the speedup is $(9 \times 200)/(8 \times 200) \approx 1.125$.

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?

A: Branch outcomes determined in the ID stage:

Instruction	1	2	3	4	5	6	7	8	9	10
sw	IF	ID	EX	MEM	WB					
lw		IF	ID	EX	MEM	WB				
beq			IF	ID	EX	MEM	WB			
add					IF	ID	EX	MEM	WB	
slt						IF	ID	EX	MEM	WB

Branch outcomes determined in the EX stage:

Instruction	1	2	3	4	5	6	7	8	9	10	11
sw	IF	ID	EX	MEM	WB						
lw		IF	ID	EX	MEM	WB					
beq			IF	ID	EX	MEM	WB				
add						IF	ID	EX	MEM	WB	
slt							IF	ID	EX	MEM	WB

Therefore, the speedup is $11/10 \approx 1.1$.

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.

A: So, the latency of EX/MEM part is $max\{150, 190\} + 20 = 210$ ps. Therefore, after change, the clock cycle time alters to $max\{200, 210\} = 210$ ps. The speedup changes to $(9 \times 200)/(8 \times 210) \approx 1.07$.

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.

A: The latency of ID changes to $120 \times (1 + 50\%) = 180$ ps and latency of EX changes to $150 - 10 = 140$ ps. Therefore, the clock cycle time is still 200 ps. Thus the speedup keeps same as 1.1.

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.

A: The latency of EX is $150 - 20 = 130$ ps. The new clock cycle time is still 200 ps.

Branch outcomes determined by MEM:

Instruction	1	2	3	4	5	6	7	8	9	10	11	12
sw	IF	ID	EX	MEM	WB							
lw		IF	ID	EX	MEM	WB						
beq			IF	ID	EX	MEM	WB					
add							IF	ID	EX	MEM	WB	
slt								IF	ID	EX	MEM	WB

Execution time: $200 \times 12 = 2400$ ps.

Branch outcomes determined by EX:

Instruction	1	2	3	4	5	6	7	8	9	10	11
sw	IF	ID	EX	MEM	WB						
lw		IF	ID	EX	MEM	WB					
beq			IF	ID	EX	MEM	WB				
add						IF	ID	EX	MEM	WB	
slt							IF	ID	EX	MEM	WB

Execution time:$200 \times 11 = 2200$ ps.

So, the speedup is $2200/2400 \approx 0.917$.

Problem 4

1.Translate this C code into MIPS instructions. Your translation should be direct, without rearranging instructions to achieve better performance.

A: For convenience, assuming the type of array is $char$.

```
        add R5,R0,R0
Loop:   beq R5, R6, End
        add R10,R1,R5
        lw R11,(R10)
        lw R12,1(R10)
        sub R11,R11,R12
        add R10,R2,R5
        sw R11,(R10)
        addi R5,R5,2
        j Loop
End:
```

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.

A: (supposing with full forwarding)

INS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
add R5,R0,R0	IF	ID	EX	MEM	WB																
Loop: beq R5, R6, End	IF	ID		EX	MEM	WB															
add R10,R1,R5		IF		ID	EX	MEM	WB														
lw R11,(R10)		IF		ID		EX	MEM	WB													
lw R12,1(R10)				IF		ID	EX	MEM	WB												
sub R11,R11,R12				IF		ID			EX	MEM	WB										
add R10,R2,R5						IF			ID	EX	MEM	WB									
sw R11, (R10)						IF			ID		EX	MEM	WB								
addi R5,R5,2									IF		ID	EX	MEM	WB							
j Loop									IF		ID	EX	MEM	WB							
Loop: beq R5, R6, End											IF	ID	EX	MEM	WB						
add R10,R1,R5											IF	ID	EX	MEM	WB						
lw R11,(R10)												IF	ID	EX	MEM	WB					
lw R12,1(R10)												IF	ID	EX	MEM	WB					
sub R11,R11,R12													IF	ID		EX	MEM	WB			
add R10,R2,R5													IF	ID		EX	MEM	WB			
sw R11, (R10)														IF		ID	EX	MEM	WB		
addi R5,R5,2														IF		ID	EX	MEM	WB		
j Loop																IF	ID	EX	MEM	WB	
Loop: beq R5, R6, End																	IF	ID	EX	MEM	WB

3.Rearrange your code from question 1 to achieve better performance on a 2-issue statically scheduled processor from Figure 4.69.

A: For convenience, assuming the type of array is $char$.

```
        add R5,R0,R0
Loop:   add R10,R1,R5
        beq R5, R6, End
        lw R11,(R10)
        lw R12,1(R10)
        add R10,R2,R5
        sub R11,R11,R12
        sw R11,(R10)
        addi R5,R5,2
        j Loop
End:
```

4.Repeat question 2, but this time use your MIPS code from question 3.

A: (supposing with full forwarding)

INS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
add R5,R0,R0	IF	ID	EX	MEM	WB												
Loop:add R10,R1,R5	IF	ID		EX	MEM	WB											
beq R5, R6, End		IF		ID	EX	MEM	WB										
lw R11,(R10)		IF		ID	EX	MEM	WB										
lw R12,1(R10)				IF	ID	EX	MEM	WB									
add R10,R2,R5				IF	ID		EX	MEM	WB								
sub R11,R11,R12					IF		ID	EX	MEM	WB							
sw R11, (R10)					IF		ID	EX	MEM	WB							
addi R5,R5,2							IF	ID	EX	MEM	WB						
j Loop							IF	ID	EX	MEM	WB						
Loop:add R10,R1,R5								IF	ID	EX	MEM	WB					
beq R5, R6, End								IF	ID	EX	MEM	WB					
lw R11,(R10)									IF	ID	EX	MEM	WB				
lw R12,1(R10)									IF	ID	EX	MEM	WB				
add R10,R2,R5										IF	ID	EX	MEM	WB			
sub R11,R11,R12										IF	ID	EX	MEM	WB			
sw R11, (R10)											IF	ID	EX	MEM	WB		
addi R5,R5,2											IF	ID	EX	MEM	WB		
j Loop												IF	ID	EX	MEM	WB	
Loop:add R10,R1,R5													IF	ID	EX	MEM	WB
beq R5, R6, End													IF	ID	EX	MEM	WB

5.What is the speedup of going from a 1-issue processor to a 2-issue processor from Figure 4.69?

A: In 1-issue processor, 9 cycles per loop. In 2-issue processor within sub-problem 4, 7 cycles per loop. So, the speedup is $9/7 \approx 1.29$.