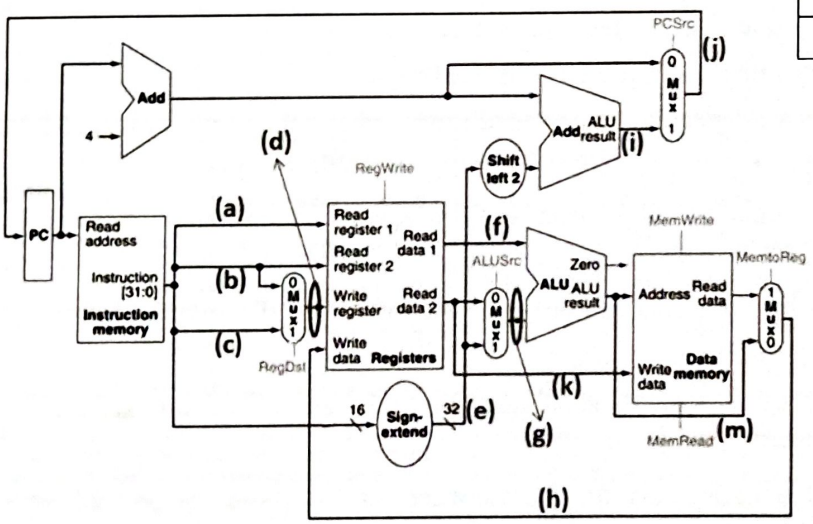
CO Exam2

1. (12%) Please answer the following questions briefly.
   (a) (2%) What is Anti-dependency?
   (b) (2%) What technique is typically used for solving the anti-dependency problem?
   (c) (2%) What is structure hazard?
   (d) (2%) What is the reservation station (in dynamically scheduled CPU) used for?
   (e) (4%) What's the difference between static multiple-issue and dynamic multiple-issue? (Please give your answer in terms of who and when is for the scheduling.)

2. (14%) There are six instructions supported by a single-cycle CPU: ***add, addi, lw, sw, slt,*** and ***beq***.
   (a) (3%) If path **(m)** is cut, which instruction(s) will not be able to run correctly?
   (b) (3%) If path **(e)** is cut, what instruction(s) will fail to run properly?
   (c) (3%) If the control signal **ALUSrc** is stuck on 0 (i.e., cannot switch to 1), what instruction(s) will fail to run properly?
   (d) (5%) What are the values on the paths **(a), (b), (e), (i), (j)** when the instruction ***beq*** below is executed. Assume the ***beq*** is located at address 0x6000000c, and the ***L*** is at 0x60000000. (Given reg. info. in the table.)

   beq $t1, $t2, ***L***

Register	ID	Initial values
$t1	0x09	0x11
$t2	0x0A	0x22



3. (8%) Assume the latencies for logic blocks are listed as table below. Please ignore the latency for control decoder, control signal delay, or other unspecified blocks. Assume only ***lw*** is supported.

I-Mem	Adder	Regs	ALU	D-Mem
100ps	30ps	50ps	80ps	120ps

   (a) For a single-cycle CPU where the instruction with the longest latency determines the clock cycle time, what's the minimum clock cycle time it can use?
   (b) For a classic MIPS CPU with a 5-stage pipeline, what's the minimum clock cycle time it can use?
   (c) What is the speedup of a pipelined processor versus a non-pipelined processor for executing 100 ***lw*** instructions? Assume there is no any data dependency in the code.
   (d) Explain why the speedup in (d) in much less than 5.

4. (5%) Consider the five-stage piplelined CPU introduced in the textbook. Please select the signals that are required to insert a stall cycle when data hazard is detected at ID stage.

   (a) IF.Flush (clear IF/ID)
   (b) ID.Flush (set 9 control signals (used in EX, MEM, WB stages) to 0
   (c) EX.Flush (set 5 control signals (used in MEM, WB stages) to 0
   (d) Set PCWrite = 0 (disable the write to Program Counter (PC))
   (e) Set IF/Dwrite = 0 (disable the write to IF/ID)

5. (5%) Consider the five-stage piplelined CPU introduced in the textbook. Please select the signals that are required when exception is detected in EX stage and need to flush the instructions currently in IF, ID and EX stages.

   (a) IF.Flush (clear IF/ID)
   (b) ID.Flush (set 9 control signals (used in EX, MEM, WB stages) to 0
   (c) EX.Flush (set 5 control signals (used in MEM, WB stages) to 0
   (d) Set PCWrite = 0 (disable the write to Program Counter (PC))
   (e) Set IF/Dwrite = 0 (disable the write to IF/ID)

6. (14%) Consider the following sequence of actual outcomes for a branch. T means the branch is taken. N means not taken. Assume both 1-bit and 2-bit predictors are initialized to **"strong taken"** state.

   Branch: **T-N-T-N-N-T-N**

   (a) (2%) What's the accuracy rate if 1-bit predictor is used?
   (b) (2%) What's the accuracy rate if 2-bit predictor is used?
   (c) (3%) If the same patterns are repeated thousands of times, what's the accuracy rate if 1-bit predictor is used?
   (d) (3%) If the same patterns are repeated thousands of times, what's the accuracy rate if 2-bit predictor is used?
   (e) (2%) For nested loops, what problems will occur for the inner loop if 1-bit predictor is used?
   (f) (2%) Is it possible to solve the problems in subquestion (e) by using the 2-bit predictor?

7. (10%) Given the control value definitions for the **forwarding MUXs** as below, where the **ForwardA** controls the MUX of first source operand of ALU and **ForwardB** controls the second one. For the code sequence below, which of following statements are correct? (Please copy the empty table to your answer sheet and fill with your answer)

Instr.	ForwardA	ForwardB
1	00	00
2		
3		
4		
5		
6		

```
1. add $t1, $t2, $t3
2. lw $t4, 4($t1)
3. add $t2, $t1, $t4
4. add $t1, $t4, $t3
5. add $t1, $t1, $t2
6. add $s2, $s1, $t1
```

Mux Control	Source
ForwardA = 00	ID/EX
ForwardA = 01	EX/MEM
ForwardA = 10	MEM/WB
ForwardB = 00	ID/EX
ForwardB = 01	EX/MEM
ForwardB = 10	MEM/WB

8. (12%) Consider the code sequence on the right side. For a 5-stage pipeline MIPS CPU, how many stall cycles would occur when executing this code with the 3 cases below, respectively? Assume that no prediction is made for branch instruction, and the two ***bne*** instructions are ***NOT taken***. Note: Register reads/writes can happen in the same cycle.

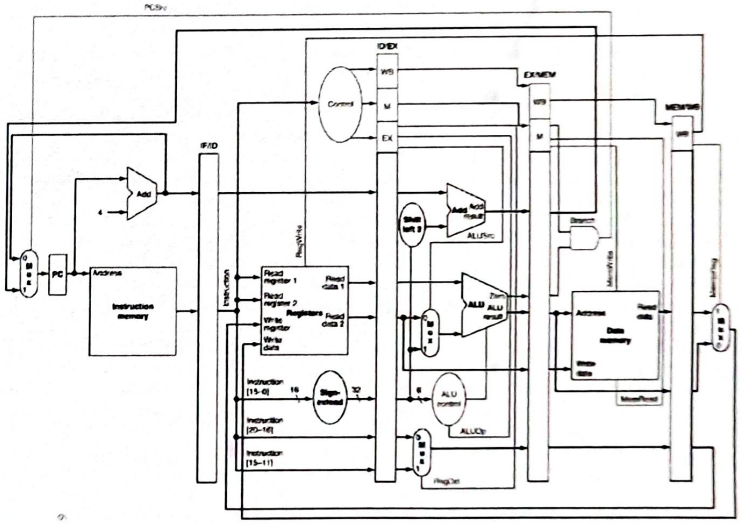
a) The CPU does not implement data forwarding and not modify h/w for branch (i.e., ***bne*** outcome is obtained in MEM stage)

b) The CPU has implemented data forwarding, but not modify h/w for branch (i.e., ***bne*** outcome is obtained in MEM stage)

c) The CPU has implemented data forwarding, and has modified h/w for branch (i.e., ***bne*** outcome is obtained in ID stage)

```
A. add $s2, $s0, $s1
B. add $s3, $s1, $s0
C. bne $s2, $s3, 10
D. lw $s0, 0($t0)
E. lw $s1, 0($t0)
F. bne $s0, $s1, 10
G. add $s2, $t1, $t2
H. add $s1, $t3, $t4
```

9. (10%) Given the pipelined CPU below and assume that stall and forwarding mechanism have been implemented. Consider the code sequence below. When the code runs to the **7th cycle**, what are the values of control signals: *RegWrite*, *RegDst*, *ALUsrc*, *MemRead* and *MemToReg*, respectively? (Please copy the table to your answer sheet and fill in your answer)



```
1. lw  $t2, 0($t2)
2. add $t3, $t2, $s2
3. sw  $t2, 0($t1)
4. lw  $t1, 0($t2)
5. add $t1, $t2, $s1
6. add $t1, $t1, $s2
7. add $t2, $s2, $t1
```

RegWrite	RegDst	ALUsrc	MemRead	MemToReg

10. (10%) For the static 2-issue pipelined processor given in the textbook and the code sequence below, answer the following questions. (Note: Assume *bne* prediction is always correct so there is no stall cycles for it. Please also ignore the clocks for the remaining stages of the last instruction)

```
Loop:  lw    $t0, 0($s1)
       addu  $t0, $t0, $s2
       sw    $t0, 0($s1)
       addi  $s1, $s1, -4
       bne   $s1, $zero, Loop
```

(a) (3%) Without using unrolling, if we schedule the code using minimum number of cycles, what's the IPC (instruction per cycle) of the scheduled code?

(b) (4%) If we unroll one more copies of the code (i.e., two times in total) and re-schedule it again using minimum number of cycles. What's the IPC of this scheduled code?

(c) (5%) Assume the loop sequence will be executed for 12 iterations. What's the speedup achieved by the scheduling in (b), compared to the scheduling in (a)?