1. (10 pts) True and False

1) Architectural techniques dominate performance improvement of processors.
2) Earlier branch resolution will decrease CPI.
3) A TLB miss causes a page fault.
4) Pipelined execution delivers better performance than single cycle execution because it improves both instruction throughput and latency.
5) Both RISC and CISC architectures can employ pipelined execution to improve the performance.
6) Displacement addressing is used for elements stored in the stack, whereas Pseudo-direct addressing is used for procedure calls.
7) Variable length op-code is used to simplify the decoding stage in a pipeline.
8) A delayed branch will necessitate adding an additional saved PC during an interrupt.
9) Larger cache block size will always decrease cache miss rate.
10) ROB (reorder buffer) holds all instructions until commit time and is responsible for tracking over-written registers.

2. (25 pts) Data Hazards

A. (10 pts) Consider the following execution sequence:

1. add $3, $1, $2
2. lw $1, 0($4)
3. and $5, $3, $4
4. and $6, $1, $2
5. or $1, $3, $6
6. sw $1, 4($4)
7. lw $2, 4($4)

Find the all the data hazards and fill the following table (you may leave some entries blank or add more rows to accommodate your answer).

Instruction #1	Instruction #2	Register	Hazard Type
add $3, $1, $2	and $5 $3 $4	$3	RAW
lw $1 0 $4	and $6 $1 $2	$1	RAW
and $6 $1, $2	or $1 $3 $6	$6	-
or $1, $3, $6	sw $1, 4($4)	$1	-

B. (3 pts) Suppose the code sequence in A is executed in a 5-stage pipelined in-order execution machine. How many *nops* do we need to insert into the pipeline **WITHOUT** a forwarding unit (your answer should be in the form of A+B+C+…+…=Z, where A, B, C….are the nops inserted in the order of the instruction sequence. E.g., A is the number of nops between instructions 1 and 2; B is the number of nops between instructions 2 and 3, etc)?

F D E m w
F x x D

0+1+0+2+2+0 =5

C. (3 pts) How many *nops* do we need to insert to the pipeline **WITH** a forwarding unit (**NO** mem to mem forwarding and the same answer format with B)?

F D E m w load use
F F D E m w

0+0+0+0+0+0 =0

D. (4 pts) For the code shown below executed on a **6 stage** MIPS pipelined datapath (Fetch, Decode, Execute, Memory-Tag, Memory-Access, Writeback), how many *nops* have to be inserted to ensure correct execution assuming **no** forwarding or hazard detection hardware? Fetching from the L1 Instruction cache takes one cycle, while loading or storing into the L1 data cache takes two cycles (Memory-Tag, Memory-Access). For a load, the data value is not available to the datapath until the end of Memory-Access.

lw $2, 40($6)

add $6, $1, $1

sw $6, 800($2)

3 nops

E. (5 pts) What are the fundamental differences between data dependencies and data hazards? Provide examples to highlight those differences.

3. (20 pts) Consider a superscalar architecture with two pipelined units, one for load/store and one for ALU instructions. The following two tables indicate the order of execution of two threads, A and B, and the latencies mandated by dependences between instructions. For example, A2 and A3 should execute after A1 and there should be at least four cycles between the execution of A3 and A5 and at least two cycles between A4 and A5. In otherwords, the tables indicate the schedule if the instructions of each of the threads are executed with no multithreading.

Note that an instruction that executes on one pipeline (for example A1 on the load/store pipeline) cannot execute on the other pipeline. Also you cannot issue instructions out of order.

time	Load/store pipeline	ALU pipeline
t	A1	
t+1	A3	A2
t+2		
t+3		A4
t+4		
t+5		
t+6	A5	A6
t+7	A7	

time	Load/store pipeline	ALU pipeline
t	B1	
t+1	B3	B2
t+2		B4
t+3		
t+4	B6	B5
t+5		B7
t+6	B8	
t+7		

Show the execution schedule for the two threads on the two pipelines assuming

(a) Fine grain multithreading (start from thread A, switch between thread A and B in every cycle)

Switch.

time	Load/store pipeline	ALU pipeline
t	A1	
t+1	B1	
t+2	A3	A2
t+3	B3	B2
t+4		A4
t+5		B4
t+6		
t+7	A5	A6
t+8	B6	B5
t+9	A7	
t+10		B7
t+11		
t+12	B8	
t+13		
t+14		

(b) Simultaneous multithreading (with priority always given to thread A)

time	Load/store pipeline	ALU pipeline
t	A1	
t+1	A3	A2
t+2	B1	
t+3	B3	A4
t+4		B2
t+5		B4
t+6	A5	A6
t+7	A7	B5
t+8	B6	
t+9		B7
t+10	B8	
t+11		
t+12		
t+13		
t+14		

4. **(10 pts)** Consider following MIPS instruction sequence:

```
lw $t0,0($s1)
add $t0,$t0,$s2
sw $t0,0($s1)
addi $s1,$s1,4
lw $t1,0($s1)
add $t1,$t1,$s2
sw $t1,0($s1)
addi $s1,$s1, 4
```

You are required to fill the bundles in the table below for a VLIW machine to have the minimum number of execution cycles. The **LEFT** part of the bundle (second column in the table) can be only ALU or branch instruction, whereas the **RIGHT** part (third column in the table) can be only load or store instruction. Note that, if there is a load-use hazard, there will be at least one cycle delay between the load instruction and the use instruction. You are allowed to reorder independent instructions and change the offset of addressing.

Cycle	ALU/Branch	Load/Store
1	addi $s1 $s1 8	lw $t0 0($s1)
2		lw $t1, -4($s1)
3	add $t0 $t0 $s2	
4	add $t1 $t1 $s2	sw $t0 -8($s1)
5	Invalid	sw $t1 -4($s1)
6		
7		
8		
9		

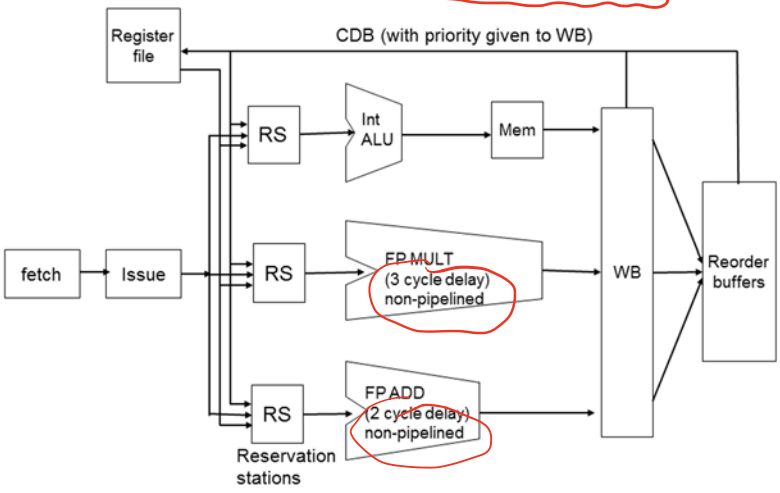
5 (10 pts). (e) What is the average memory access time (AMAT) when you have the following memory hierarchy? Assume that (i) the cache uses physical addresses and TLB lookup is before L1 cache access, (ii) the CPU stalls until the data is delivered, (iii) everything fits into the memory, and (iv) the hardware does the page table access and updates TLB.

Unit	Access Latency	Hit Rate
TLB	1 cycle	95%
L1	1 cycle	95%
L2	8 cycles	80%
L3	50 cycles	50%
Memory	100 cycles	100%
Page table access & TLB update	200 cycles	100%

6 (25 pts) In this question, you will trace the execution of the shown eight instructions on the dynamically scheduled (with speculation) out of order processor shown in thefigure assuming that:

- The cache is perfect (no misses),
- There is **NO** forwarding support.
- The number of reservation stations and reorder buffers is very large, with the ROBs used to rename the registers.
- Up to two instructions can be issued in the same cycle as long as they are not issued to the same execution unit,
- The architecture has only one CDB with priority given to the WB stage. In case of structural hazard on the WB stage, the instruction that issued first has higher priority.
- If an instruction is in WB while occupying the first entry in the ROB then it is committed in the **same** cycle (see the first two entries of the table where C indicates that the instruction has been committed and the ROB entry has been released).
- The "Int ALU" is used for integer operations as well as for memory address computation of load and store instructions.

I0: fld	F0, 0(R1)
I1: fld	F1, 8(R1)
I2: fmul.d	F1, F0, F1
I3: addi	R1, R1, 16
I4: fadd	F1, F0, F1
I5: fld	F2, 8(R1)
I6: fadd.d	F2, F1, F2
I7: bne	R1, R2, L



(a) Complete the table to indicate the state of each instruction in consecutive cycles (I=issued, EX=executing, M=accessed memory, WB=write back, ROB= in ROB, C=committed).

		Instruction status in cycle																	
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
fld	F0, 0(R1)	I	EX	M	WB	C													
fld	F1, 8(R1)		I	EX	M	WB	C												
fmul	F1, F0, F1		I																
addi	R1, R1, 16																		
fadd	F1, F0, F1																		
fld	F2, 8(R1)																		
fadd	F2, F1, F2																		
bne	R1, R2, L																		

(b) Complete the following table which specifies the relevant entries of the register status table at the end of cycles 1, 2, 3, 4 and 5.

	F0	F1	F2	…..	R1	…..	
Cycle 1	ROB0	--	--		--		
Cycle 2							
Cycle 3							
Cycle 4							
Cycle 5							