Name: ____________________________.

Computer Science and Engineering 431 (CMPEN 431)

Exam 2

Fall 2022

10/27/2022

Name: ____.

Q1	15	
Q2	20	
Q3	10	
Q4	10	
Q5	20	
Q6	15	
Total	90	

Note:

You have **1 hour and 15 minutes** to complete the exam.

This exam is **closed notes and book**.

Please **write down solution steps** for partial credits to be considered.

Please also **write you name on top of each page**.

I strongly recommend **solving easy problems first**, as problems are not in the order of complexity.

Name: ___________________________.

Q1. Select True or False (15 pt, 1pt each).

- (True / [✓ False]) a=3; b=a is a WAR dependence.
- ([✓ True] / False) WAR and WAW dependences caused by registers can be resolved by register renaming.
- (True / [✓ False]) VLIW has good backward code compatibility.
- (True / [✓ False]) In an OoO processor, commit of ROB entries can happen out-of-order.
- ([✓ True] / False) Conservative load-store reordering allows reordering between loads, but loads cannot be reordered with a preceding store.
- ([✓ True] / False) ILP is the best possible IPC of a hypothetical hardware with infinite resources.
- (True / [✓ False]) Coarse-grain multithreading (CGMT) can efficiently fill load-use stalls.
- ([✓ True] / False) In fine-grain multithreading (FGMT), multiple threads share the ALU.
- ([✓ True] / False) If you immediately load from an address that you just stored to in an OoO processor, the load value may come from an LSQ.
- (True / [✓ False]) SRAM is slower than DRAM but is more energy-efficient.
- (True / [✓ False]) Write-through cache only updates the lower-level cache or memory when the cache block is evicted.
- ([✓ True] / False) When the program first starts, the cache is empty and will cause a cache miss for any accesses. This is called a compulsory miss.
- ([✓ True] / False) If a cache hit is 1 cycle, miss penalty is 100 cycles, and 2% of the access misses, AMAT is 3.
- ([✓ True] / False) Using a set-associative cache instead of a direct-mapped cache can reduce conflict misses.
- (True / [✓ False]) As the processor improves, the memory stall due to cache misses becomes less of a problem.

Name: ____________________________.

Q2. Consider the MIPS assembly program shown below. Answer the following questions (20pt total).

```
L: lw $5, 128($1)
   lw $6, 132($1)
   add $5, $5, $6
   sw $5, 0($1)
   addi $1, $1, -4
   bne $1, $0, L
```

- Q2-1. Schedule the instructions into VLIW packets to achieve **minimum** number of execution cycles. Assume a 2-issue VLIW, where the **first slot can only do ALU/branch** operation and the **second slot can only do load/store**. Put **nop** for empty slots. You may reorder independent instructions, rename registers, and change the offset of load/store (5pt).

Cycle	ALU/Branch	Load/Store
1	L: This can go here instead (other offsets may change)	lw $5, 128($1)
2		lw $6, 132($1)
3	addi $1, $1, -4	nop
4	add $5, $5, $6	nop
5	bne $1, $0, L	sw $5, 4($1)
6		
7		
8		

- Q2-2. Loop unrolling has many benefits. However, it also has its downside. State one **disadvantage** of loop unrolling (Hint: compiler complexity or compilation time increase is NOT the right answer – it is not that hard for the compiler to do loop unrolling). (5pt)

Increased code size (partial credit answer: needs more registers)

Name: Was not part of the exam

- Q2-3. Assuming the loop from above can be perfectly unrolled by 2. Write down the MIPS assembly program where the above loop is unrolled by 2. You should merge equivalent instructions or multiple `addi` instructions to the same register (5pt).

```
L: lw $5, 128($1)
   lw $6, 132($1)
   lw $7, 124($1)
   add $5, $5, $6
   add $7, $7, $5
   sw $5, 0($1)
   sw $7, -4($1)
   addi $1, $1, -8
   bne $1, $0, L
```

- Q2-4. Schedule the instructions you wrote in Q2-3 into VLIW packets to achieve **minimum** number of execution cycles. Assume the same constraints as Q2-1 (5pt).

Cycle	ALU/Branch	Load/Store
1	L: this can go here	lw $5, 128($1)
2		lw $6, 132($1)
3	addi $1, $1, -8	lw $7, 124($1)
4	add $5, $5, $6	
5	add $7, $7, $5	sw $5, 8($1)
6	bne $1, $0, L	sw $7, 4($1)
7		
8		

Name: ______________________________.

Q3. Consider a Tomasulo's machine shown below. In the instruction queue, `add` is the first instruction to be issued, and `mul` is the last instruction. Reservation station RS1—3 can only hold add/sub instructions, and RS4—6 can only hold mul/div instructions (10pt total).

Instruction queue

mul \$2, \$2, \$3
add \$1, \$1, \$3
sub \$4, \$1, \$4
div \$4, \$1, \$3
add \$1, \$2, \$3

Register files

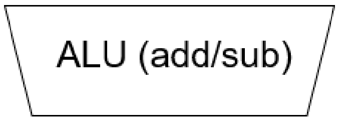
\$1	3
\$2	-1
\$3	2
\$4	7

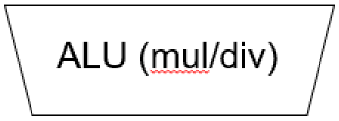
\$1	RS3
\$2	RS5
\$3	
\$4	RS2

RAT

	op	src1	src2
RS1	add	-1	2
RS2	sub	RS1	RS4
RS3	add	RS1	2

	op	src1	src2
RS4	div	RS1	2
RS5	mul	-1	2
RS6			





Reservation station & ALUs

- Q3-1. If all the instructions were issued without being dispatched, what would the final reservation station and the RAT would look like? Fill the RAT and the reservation station entries above. If multiple slots in the reservation station is available, always select the slot with the smallest number (Hint: RAT should be filled with RS1—6, the op column of the reservation station should be filled with add/sub/mul/div, and the src1/2 columns of the reservation station should be filled with either an integer or RS1—6). (5pt)

- Q3-2. Indicate the order each instruction will be executed (dispatched) in the below table. Write 1 for the instruction that will execute first, 2 for the instruction that will execute second, etc. Note that at most two instructions (one from add/sub, one from mul/div) may execute simultaneously and have the same order number. (5pt)

add \$1, \$2, \$3	1
div \$4, \$1, \$3	2
sub \$4, \$1, \$4	3
add \$1, \$1, \$3	2
mul \$2, \$2, \$3	1

Name: ______________________________.

Q4. Consider the OoO processor with ROB below. (10pt total)

- Q4-1. At a certain point, div and add instruction is ready to be executed. Indicate how the ROB, register files, RAT, and the reservation station will be updated after the two instructions are executed and the results are broadcasted. If something needs to be freed, cross it out. If something needs to be added, write it at the correct slot. If something needs to be replaced, cross it out and write the updated value. Assume nothing is committed, and the sub instruction is not executed. (5pt)

ROB

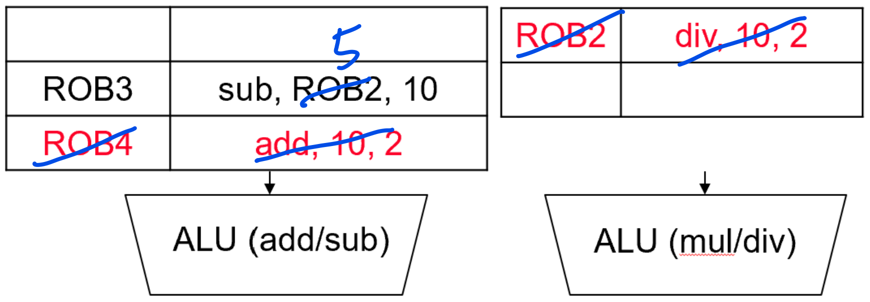
\$2	10	Done!
\$1	5	Done!
\$4		
\$1	12	Done!

Register files

\$1	3
\$2	-1
\$3	2
\$4	7

\$1	ROB4
\$2	ROB1
\$3	
\$4	ROB3

RAT



	5
ROB3	sub, ~~ROB2~~, 10
~~ROB4~~	~~add, 10, 2~~

~~ROB2~~	~~div, 10, 2~~

- Q4-2. After you broadcast the two instructions from Q4-1, assume you perform a series of commits, as much as possible. Which of the four slots in the ROB can be committed? If there are slots that are filled but cannot be committed in the ROB, explain why. Assume that the sub instruction is still not executed and is in the reservation station. (5pt)

ROB1 and ROB2.

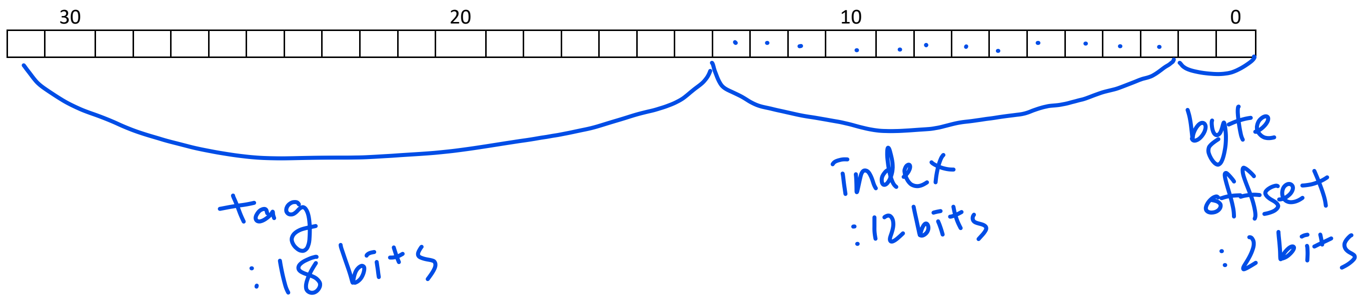
ROB4 cannot be committed because commit must happen in-order and ROB3 is not ready

Name: ______________________________.

$\frac{\overset{4}{\cancel{16}}\cancel{K}\cancel{B}}{\cancel{4}\cancel{B}}$ 4K

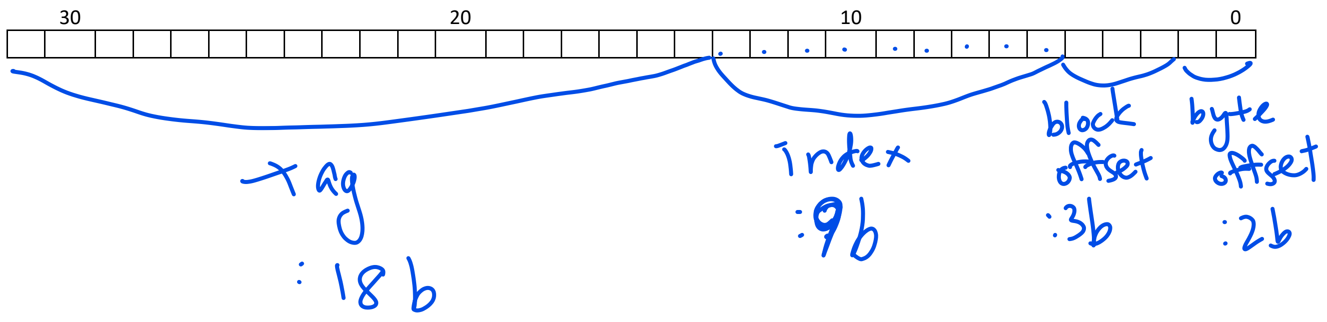
Q5. Consider a direct-mapped cache with 16KB of data. Answer the following questions. Assume a 32-bit address. (20pt total)

- Q5-1. If each cache block is **one word**, indicate how many bits are needed for byte offset, block offset, index, and tag. Indicate the position of each field below. (5pt)



$\frac{\overset{2}{\cancel{16}}K\cancel{B}}{\cancel{8}\times\cancel{4}\cancel{B}}$ 512

- Q5-2. If each cache block is **8 words**, indicate how many bits are needed for byte offset, block offset, index, and tag. Indicate the position of each field below. (5pt)



Name: ____________________________.

- Q5-3. How many bits are needed for the tag and the valid bit to implement the cache from Q5-1 and Q5-2? (5pt)

Q5-2: $512 \times (1b + 18b + 256b)$

$= 275 \times 512b = \boxed{137.5Kb}$

- Q5-4. If we keep increasing the block size when the cache capacity is fixed, what will happen to the cache miss rate and why? (Hint: the answer may not be simply "will go down" or "will go up") (5pt)

It will first go down due to spatial locality, and then go up because of the reduced number of blocks (increased conflict misses).

Name: ______________________________.

Q6. Assume a 2-way set-associative cache shown below. There are 4 sets, and each block is 2 words. (15pt total)

valid?	tag	word1	word0
1	~~00~~ 01	Mem(~~1~~) 9	Mem(~~0~~) 8
1	~~10~~ 01	Mem(~~19~~) 11	Mem(~~18~~) 10

Way 0

valid?	tag	word1	word0
1	~~01~~ 10	Mem(~~9~~) 17	Mem(~~8~~) 16
1	00	Mem(3)	Mem(2)

Way 1

- Q6-1. If memory access pattern is shown as below, indicate whether each access is a cache hit or a cache miss. If it is a cache miss, indicate into which set of which way it needs to be inserted. **If both ways are available, always insert to way 0**. Assume the top of each way is set 0, and the bottom is set 3. Assume a **perfect LRU** replacement policy. (10pt)

Access	address in bits	hit/miss ?	where to insert (only for cache misses)
Mem(0)	0…00000xx	miss	Way 0, set 0
Mem(8)	0…01000xx	miss	Way 1, set 0
Mem(1)	0…00001xx	hit	Way ____, set ____
Mem(17)	0…10001xx	miss	Way 1, set 0
Mem(18)	0…10010xx	miss	Way 0, set 1
Mem(2)	0…00010xx	miss	Way 1, set 1
Mem(19)	0…10011xx	hit	Way ____, set ____
Mem(3)	0…00011xx	hit	Way ____, set ____
Mem(10)	0…01010xx	miss	Way 0, set 1
Mem(8)	0…01000xx	miss	Way 0, set 0

tag

set index

Name: ______________________________.

- Q6-2. Below, draw the final state of the cache. You only need to **write the lower 2 bits of the tag**. For data, write in the form of Mem(0), Mem(1), etc. (5pt)

valid?	tag	word1	word0
1	01	Mem(9)	Mem(8)
1	01	Mem(11)	Mem(10)

Way 0

valid?	tag	word1	word0
1	10	Mem(17)	Mem(16)
1	00	Mem(3)	Mem(2)

Way 1

(end of the exam)