ECE 6913 A
Final Exam

Released: 12/16 11:55 AM
Due: 12/18 11:55 PM

Question	Points	Possible
1		24
2		30
3		35
4		11
Total		100

You must work individually on the final. If you are caught cheating, you will receive an automatic zero and risk failing the course.

To receive full credit, show all your work and make sure answers are legible for grading (we suggest typing responses).

All test must be submitted in PDF format. Submitting in other formats will result in losing 10 points.

Please ask all questions through Piazza. There is a “Final” folder with subfolders for each question. We will do our best to respond quickly. I will not be holding a Zoom hour this time because I don’t think it’s fair for folks in other time zones who can’t call in; everyone should hear the same information.

Questions 1: Virtual memory (24 Points)

Addressing details:

- 14-bit virtual addresses
- 12-bit physical address
- Page size = 64 bytes

Assume the TLB, Page Table, and Cache have the following states:

TLB

Set	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid
0	03	–	0	09	0D	1	00	–	0	07	02	1
1	03	2D	1	02	–	0	04	–	0	0A	–	0
2	02	–	0	08	–	0	06	–	0	03	–	0
3	07	–	0	03	0D	1	0A	19	1	02	–	0

Page Table

VPN	PPN	Valid
00	28	1
01	–	0
02	33	1
03	02	1
04	–	0
05	16	1
06	–	0
07	–	0

VPN	PPN	Valid
08	13	1
09	17	1
0A	09	1
0B	–	0
0C	–	0
0D	2D	1
0E	11	1
0F	0D	1

Cache

Idx	Tag	Valid	B0	B1	B2	B3
0	19	1	99	11	23	11
1	15	0	–	–	–	–
2	1B	1	00	02	04	08
3	36	0	–	–	–	–
4	32	1	43	6D	8F	09
5	0D	1	36	72	F0	1D
6	31	0	–	–	–	–
7	16	1	11	C2	DF	03

Idx	Tag	Valid	B0	B1	B2	B3
8	24	1	3A	00	51	89
9	2D	0	–	–	–	–
A	2D	1	93	15	DA	3B
B	0B	0	–	–	–	–
C	12	0	–	–	–	–
D	16	1	04	96	34	15
E	13	1	83	77	1B	D3
F	14	0	–	–	–	–

A) [12 points] Complete the information using the given virtual address, addressing detail and TLB, Cache, and Page Table states given above.

Virtual address: 0x0AC0
001010 11 000000
Virtual Page Number: 0x2B

TLB Index: 3

TLB Tag: 0x0A

TLB Hit: Yes

Page Fault: No

Physical Page number: 0x19
011001 0000 00
Physical Page Offset: 000000

Cache Offset: 0

Cache Index: 0

Cache Tag: 0x19

Hit: Yes

B) [12 points] Complete the information using the given virtual address, addressing detail and TLB, Cache, and Page Table states given above.

Virtual address: 0x00EB
000000 11 101011
Virtual Page Number: 0x03

TLB Index: 3

TLB Tag: 0x00

TLB Hit: No

Page Fault: No

Physical Page number: 0x02
000010 1010 11
Physical Page Offset: 101011

Cache Offset: 3

Cache Index: 0xA

Cache Tag: 0x02

Hit: No

Question 2: Cache replacement and performance (30 Points)

A 16-Bytes, 4-way set-associative cache with 2-Byte blocks is used as the L1-cache for a processor with a Byte addressable main memory.

A) [5 points] Determine the number of offset bits, index bits, and tag bits for the cache.

Num_of_offsetBits = log2(blocksize) = log2(2) = 1;
Num_of_indexBits = log2(cacheSize/blockSize/way) = log2(16/2/4) = 1;
Num_of_tagBits = 32 – Num_of_offsetBits – Num_of_indexBits = 32 – 1 – 1 = 30;

B) [15 Points] Assume that the cache uses a true LRU replacement policy, i.e., it evicts the block that was least recently used/accessed.

The ways of the cache are labeled 0, 1, 2, 3. You can assume that when a cache block is fetched into a set with one or more empty ways, it is placed in the empty way with the lowest number label. For example, if all four ways are empty and a cache block is fetched, it would be placed in Way 0.

For the sequence of accesses below, determine if the access is a Hit or a Miss. Also, indicate the index of the set and which Way (Way 0, 1, 2, or 3) the block is placed in.

All accesses are assumed to be read accesses.

Access Address (Decimal)	Hit/Miss?	Set Index	Way # (0, 1, 2, 3)
0	miss	0	0
2	miss	1	0
4	miss	0	1
8	miss	0	2
12	miss	0	3
1	hit	0	0
16	miss	0	1
5	miss	0	2
3	Hit	1	0
8	miss	0	3

C) [10 Points] Determine the better cache design (Hint: remember AMAT from class). Both designs take 1000 cycles to access memory.

a. Design 1 is an inclusive cache. This means that all data in the L1 is also in the L2. The design is simple and makes it fast to move data between cache levels. The L1 has a 2-cycle access time and a miss rate or 7%. The L2 has a 20-cycle access time and a miss rate of 1%.
b. Design 2 is an exclusive cache. This means data can be in either L1 or L2 but not both. The benefit is that the effective capacity of the cache system is increased as when data is in L1, L2 does not waste space holding the same data; the drawback is that moving data between caches is now more complex. This cache's L1 has a 2-cycle access time and miss rate of 7%. The L2 now has a 22-cycle access time penalty and a miss rate of 0.55%.

Which design is better? Please provide a quantitative argument.

AMAT of a:
$T = 2 + 0.07(20 + 0.01(1000)) = 4.1$ cycles

AMAT of b:
$T = 2 + 0.07(22 + 0.0055(1000)) = 3.925$ cycles

Design B is better because it has a lower average memory access time.

Question 3: Tomasulo (35 Points)

Consider the following instruction sequence:

```
LD F0, 0(R1)
ADD.D F2, F0, F4
MUL.D F4, F2, F6
ADD.D F6, F8, F10
ADDI R1, R1, #8
LD F12, 0(R2)
MUL.D F12, F12, F8
ADD.D F6, F0, F6
ADDI R2, R2, #8
```

In this question you will simulate the instructions running out-of-order using Tomasulo's algorithm as we did in class.

You should assume the machine has 3 FP adders and 2 FP multipliers. One instruction is issued per cycle. Each functional unit has its own reservation station. There is also one reservation station for loads (this question ignores stores). FP additions takes 5 cycles, FP multiplication takes 8 cycles, loads take 1 cycle, and integer (ADDI Rs) take one cycle. You can assume that the integer unit has its own CDB and does not interfere with the FP unit.

Functional units are not pipelined and we will assume that completed instructions write their results to the CDB on the following cycle. During the CDB broadcast cycle, dependent instructions have their "timers" start. E.g., the first LD is issued in cycle 1, completes in cycle 2, and the result written to the CDB in cycle 3. The first add, dependent on the LD, would start its timer in cycle 3 and complete on cycle 8, writing the CDB in cycle 9. Only 1 instruction can write the CDB per cycle, the oldest instruction always has priority.

Your task is to simulate the remaining cycles. To help, the structures from class have been attached here and a Google sheet can be found here: https://docs.google.com/spreadsheets/d/1Cwp4_gLlZHQSZYdZsg6C0jY1NLmgvR08i3aaNWFwjws/edit?usp=sharing

You will need to use your NYU account to see the Template. You can copy the template to a private sheet and work there if you'd like.

A) [10 points] List all RAW, WAW, and WAR dependencies in the program. Don't assume anything about the underlying microarchitecture and report all possible dependencies over the entire program.

RAW:

LD F0, 0(R1)
ADD.D F2, F0, F4

LD F12, 0(R2)
MUL.D F12, F12, F8

WAR:

MUL.D F4, F2, F6
ADD.D F6, F8, F10

WAW:

ADD.D F6, F8, F10
ADD.D F6, F0, F6

LD F12, 0(R2)
MUL.D F12, F12, F8

B) [25 Points] Complete the “Instruction Status” table indicating how many cycles the program took to execute and when Issue, Execution Complete, and Write CDB happen.

You are welcome to link or submit a full simulation of the to maximize partial credit. At a minimum you must show the machine state (i.e., the values of “Instruction Status”, “Reservation Stations”, “Memory Unit”, and “Register Results Status”) at the cycle when each of the three ADD.D instruction complete execution (not write CDB).

Cycle 8

Instruction Status

	Issue	Exec. Comp	Write CDB
LD F0, 0(R1)	1	2	3
ADD.D F2, F0, F4	2	8	
MUL.D F4, F2, F6	3		
ADD.D F6, F8, F10	4		
ADDI R1, R1, #8	5	6	7
LD F12, 0(R2)	6	7	8
MUL.D F12, F12, F8	7		
ADD.D F6, F0, F6	8		
ADDI R2, R2, #8			

Reservation Stations

Timer	Name	Busy?	Op	Vj	Vk	Qj	Qk
0	Add 1	Add 1	yes	ADD.D	M(A1)	F4	
1	Add 2	Add 2	yes	ADD.D	F8	F10	
	Add 3	Add 3	yes	ADD.D	F0		
	Mult1	Mult 1	yes	MUL.D		F6	Add1
8	Mult2	Mult 2	yes	MUL.D	M(A2)	F8	

Memory Unit

Timer		Busy?	Address
	Load1	N	

Register Result Status

	F0	F2	F4	F6	F8	F10	F12

Fu		Add1	Mult1	Add3			M(A2)

Cycle 9

Instruction Status

	Issue	Exec. Comp	Write CDB
LD F0, 0(R1)	1	2	3
ADD.D F2, F0, F4	2	8	9
MUL.D F4, F2, F6	3		
ADD.D F6, F8, F10	4	9	
ADDI R1, R1, #8	5	6	7
LD F12, 0(R2)	6	7	8
MUL.D F12, F12, F8	7		
ADD.D F6, F0, F6	8		
ADDI R2, R2, #8	9		

Reservation Stations

Timer	Name	Busy?	Op	Vj	Vk	Qj	Qk
	Add 1						
0	Add 2	yes	ADD.D	F8	F10		
	Add 3	yes	ADD.D	F0			Add2
8	Mult1	yes	MUL.D	M-M	F6		
7	Mult2	yes	MUL.D	M(A2)	F8		

Memory Unit

Timer		Busy?	Address
	Load1	N	

Register Result Status

	F0	F2	F4	F6	F8	F10	F12
Fu		M-M	Mult1	Add3			Mult2

Cycle 15

Instruction Status

	Issue	Exec. Comp	Write CDB
LD F0, 0(R1)	1	2	3
ADD.D F2, F0, F4	2	8	9
MUL.D F4, F2, F6	3		
ADD.D F6, F8, F10	4	9	10
ADDI R1, R1, #8	5	6	7
LD F12, 0(R2)	6	7	8
MUL.D F12, F12, F8	7		
ADD.D F6, F0, F6	8	15	
ADDI R2, R2, #8	9	10	11

Reservation Stations

Timer	Name	Busy?	Op	Vj	Vk	Qj	Qk
	Add 1	Add 1					
	Add 2	Add 2					
0	Add 3	Add 3	yes	ADD.D	F0	M-M+M	
2	Mult1	Mult 1	yes	MUL.D	M-M	F6	
1	Mult2	Mult 2	yes	MUL.D	M(A2)	F8	

Memory Unit

Timer		Busy?	Address
	Load1	N	

Register Result Status

	F0	F2	F4	F6	F8	F10	F12
Fu			Mult1	Add3			Mult2

Cycle 18

Instruction Status

	Issue	Exec. Comp	Write CDB
LD F0, 0(R1)	1	2	3
ADD.D F2, F0, F4	2	8	9
MUL.D F4, F2, F6	3	17	18
ADD.D F6, F8, F10	4	9	10
ADDI R1, R1, #8	5	6	7
LD F12, 0(R2)	6	7	8
MUL.D F12, F12, F8	7	16	17
ADD.D F6, F0, F6	8	15	16
ADDI R2, R2, #8	9	10	11

Reservation Stations

Timer

Name	Busy?	Op	Vj	Vk	Qj	Qk
Add 1						
Add 2						
Add 3						
Mult1						
Mult2						

Memory Unit

Timer

	Busy?	Address
Load1	N	

Register Result Status

	F0	F2	F4	F6	F8	F10	F12
Fu			R(F4)				

Question 4: Short answer (11 Points)

A) [6 points] Consider two caches that have the same capacity but different block sizes. Cache-1 has a small block size while Cache-2 has a large block size. Name one advantage and one disadvantage of Cache-1 verses Cache-2.

Advantage:
Due to small block size C1 has smaller missing penalty.
Disadvantage:
C1 will have higher missing rate than C2.

B) [4 points] What is a limitation of Tomasulo's algorithm (something it does not support) that the Re-Order Buffer (ROB) solves?

Tomasulo's algorithm has out-of-order commitment. Re-Order Buffer allows Tomasulo's algorithm also have in-order commitment.

C) [1 point] Complete you course evaluation; did you submit it?
Yes.