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

Virtual Page Number:

TLB Index:

TLB Tag:

TLB Hit:

Page Fault:

Physical Page number:

Physical Page Offset:

Cache Offset:

Cache Index:

Cache Tag:

Hit:

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

Virtual Page Number:

TLB Index:

TLB Tag:

TLB Hit:

Page Fault:

Physical Page number:

Physical Page Offset:

Cache Offset:

Cache Index:

Cache Tag:

Hit:

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.

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			
2			
4			
8			
12			
1			
16			
5			
3			
8			

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.

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.

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).

Instruction Status

	Issue	Exec. Comp	Write CDB
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			

Reservation Stations

Timer

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

Memory Unit

Timer

	Busy?	Address
Load1		

Register Result Status

	F0	F2	F4	F6	F8	F10	F12
Fu							

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.

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

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

Question 3 Branch Prediction

In this example, assume that each loop uses a single branch instruction to decide where to go next. Consider the following loop nest:

```
Branch 1: while( true )                        // PC: 0000
Branch 2:    for( int i = 0; i < 2; i++ )      // PC: 0100
Branch 3:       for( int j = 0; j < 1; j++ )   // PC: 1000
                    <do something>
```

a) **[5 points]** Starting with "Branch 1", write the first 16 branch outcomes for the above code below.
Hint: The first three are given! Complete the sequence.

Number	**Branch**	**Outcome**
1	Branch 1	Taken
2	Branch 2	Taken
3	Branch 3	Taken
4	Branch 3	
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		

b) **[5 points]** Assume we run the above code on a machine implementing Smith's predictor. The predictor uses the two PC MSBs to index the Pattern History Table (PHT). Note that in this case the addresses are only 4b long, so it's skipping the 2 LSBs. You should assume the PHT entries are 2b saturating counters all initialized to Weakly Taken (WT).

How many entries does the PHT have and what is the predictors accuracy?

Branch Address	**Current State**	**Guess**	**Result**	**Next State**
00	WT	T	T	ST
01	WT	T	T	ST
10	WT	T	T	ST

c) **[20 points]** The CTO of Intel is calling you. They can't figure out how to design a branch predictor for the above branch pattern and can only afford to build an 8 entry PHT with 2b counters. They are convinced that the best option is to implement a static policy that guesses "Always Taken" and simply save the area the PHT would otherwise take up.

You don't believe this is true because you recently took ECE 6913 A at NYU and are a branch prediction expert. You have just enough time to evaluate two designs before the CPU microarchitecture is frozen and sent off for fabrication, which one do you choose?

1) BP1: A two-level branch predictor that uses a 2-bit Global History Register (GSR) and the 3rd (from the right) bit of the branch PC address to index an 8 entry PHT. (E.g., if a branch address is 0100 the address bit used would be 1.) Assume the PHT using 2b saturating counters that are initialized to Weakly Taken (WT).

2) BP2: A two-level branch predictor that uses a 3-bit GHR to directly index the PHT. In this scenario, the PC address is not used to index the PHT.

To evaluate the best design, simulate the 16 branch outcomes from part a) on both designs. To receive full credit, explain why each predictor is unable to do better by analyzing why mispredictions happen.

BP1:

GHR	**Branch PC Address**	**PHT Index**	**Current State**	**Guess**	**Result**	**Next State**
00	00	000	WT	T	T	ST
01	01	011	WT	T	T	ST
11	10	110	WT	T	T	ST

BP2:

GHR	PHT Index	Current State	Guess	Result	Next State
000	000	WT	T	T	ST
001	001	WT	T	T	ST
011	011	WT	T	T	ST

d) **[5 points]** What would happen if BP2 from part c) was allowed to use 4bits of global branch history and the PHT had 16 entries? Explain your answer, you don't need to show all the outcomes as above unless you want to.