ECE 6913A
Final Exam

Released:	12/14/2021	5:00 PM EST
Due:	12/21/2021	4:59 PM EST

You must upload a PDF of your solutions to Brightspace by the deadline. Solutions not in PDF format will not be considered.

No extensions will be given, nor late submissions accepted. If you do not submit your exam on time no credit will be given. There's plenty of time to complete the test, we highly suggest getting in in early to avoid issues.

You must work independently on the exam. Working together is considered cheating. If you are suspected of cheating, you will be **reported to NYU and risk failing the class.**

To receive full credit and maximize partial credit you must show your work. Any answers that are not legible will not be graded and you will lose points accordingly.

You are welcome to post questions on Piazza, but only ask general or clarifying ones. Any questions about specific answers will not be answered and subject to deletion.

Any assumptions made must be explicitly stated.

Name:

Question	Points	Possible
1		**35**
2		**34**
3		**20**
4		**11**
Total		**100**

Question 1: Tomasulo's Algorithm (35 Points)

Consider the following instruction sequence:

```
LD F0, 0(R1)
LD F2, 0(R2)
ADDD F8, F2, F0
MULTD F4, F0, F8
LD F10, 4(R1)
SUBD F10, F10, F6
ADDD F4, F4, F10
SUBD R1, R1, 1024
```

In this question you will simulate the instruction sequence on an out-of-order processor that implements Tomasulo's algorithm, just as we did in class.

This question makes the same assumptions as we did in class.

The machine can Issue one instruction per cycle.

There are two functional units. One supports addition and subtraction and has 3 reservation stations. This functional unit is pipelined and has a latency of 2 cycles. The other functional unit processes multiplication and division. It is also pipelined, has a latency of 10 cycles and has 2 reservation stations. Integer operations are assumed to take 1 cycle.

The load-store unit is also pipelined and there are three reservation stations for both loads and stores. You can assume that there are no memory ambiguation issues in this problem. Loads and stores are assumed to take 4 cycles and are broadcast over the same CDB as the functional units.

There is a single CDB shared by all units in the machine. Assume that only one instruction can write to the CDB at a time and that the earliest issued instruction is always given priority. We assume the CDB is fast, and that dependent instructions getting values from the CDB can start their execution if they are able. Thus, the execution timers start when the values are received (this is what we assumed in the class examples).

A) [5 Points]
List all RAW, WAW, and WAR dependencies in the program (copied here for convenience). You should not assume anything about the underlying microarchitecture in this subproblem.

```
LD F0, 0(R1)
LD F2, 0(R2)
ADDD F8, F2, F0
MULTD F4, F0, F8
LD F10, 4(R1)
SUBD F10, F10, F6
ADDD F4, F4, F10
SUBD R1, R1, 1024
```

B) [25 Points]
Complete the “Instruction Status” table indicating how many cycles the program took to execute and when instructions were Issued, their Execution Completed, and Write CDB happens.

You are welcome to submit a full, cycle-by-cycle simulation of the machine to maximize partial credit. At a minimum, you must show the state of the machine (i.e., the state of Reservation Stations, Memory Unit, and Register Result Status, as given below) at cycles 7 and 15.

You can replicate the structures below to simulate as many cycles as needed, only the two mentioned above must be included for full credit.

Instruction Status

	Issue	Exec. Comp	Write CDB
LD F0, 0(R1)			
LD F2, 0(R2)			
ADDD F8, F2, F0			
MULTD F4, F0, F8			
LD F10, 4(R1)			
SUBD F10, F10, F6			
ADDD F4, F4, F10			
SUBD R1, R1, 1024			

Reservation Stations

Timer

Name	Busy?	Op	Vj	Vk	Qj	Qk
Add1						
Add2						
Add3						
Mult1						
Mult2						

Memory Unit

Timer

	Busy?	Address
Load1		
Load2		
Load3		

Register Result Status

	F0	F2	F4	F6	F8	F10	F12
FU							

C) [5 Points]
Why does Tomasulo's algorithm outperform an in-order pipeline? Explain your answer using the multiply instruction in this problem.

Hint – think about how the Multiply is handled in an in-order machine verses here using out-of-order... where is the speedup coming from?

Please keep your answer short, 2-4 sentences is plenty.

Question 2: Branch Prediction [34 Points]

In this example, consider the following snippet of code, which constitutes three conditionals.

```
for(i=0; i<4; i++)       // branch1; address = 1100
  if(i % 2 == 1)         // branch2; address = 0100
    for(j=0; j<3; j++)   // branch3; address = 1000
      < do something unrelated to i and j and without branches >

// NOTE: for this example, we don't include addresses for the
"other" instructions in the brackets (< >), only branches.
// The "%" operator is the true modulus operation.
```

In assembly, this is:

```
branch1_label:
   Branch_if_i_mod_2_is_1(branch2_label) // Address = 0100
branch3_label:
   < do something unrelated to i and j and without branches >
   Branch_if_j<3(branch_3_label) // Address = 1000
branch2_label:
   Branch_if_i<4(branch1_label) // Address = 1100
```

A) [4 Points]
Write out all the branches for the above code snippet in program order (i.e., in the order in which they are encountered). Include whether each branch was taken (T) or not taken (NT).

Throughout this question you may grow or shrink this table as needed; the number of branches is not given.

Number	Branch	Outcome
1	branch2	NT
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		

B) [7 Points]

Assume you have limited space and can only store a PHT of 4 entries. Assume that you're using the standard 2b saturating counter for each PHT entry and that all counters are initialized to the weakly taken (WT) state. You should assume that the LSBs of the branch addresses are used to index the table and that the 2 LSBs of the addresses (i.e., the 00s) are ignored.

For full credit you need to show the state of the PHT after branch 9 and at the end of the program as well as report the correct prediction rate by completing the table below.

PHT

Entry	Current State
0	WT
1	WT
2	WT
3	WT

Number	Branch	Prediction	Outcome	Correct?
1	branch2		NT	
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				

Prediction Table

C) [10 points]

Recall that two-level predictors typically outperform simple Smith predictors.
In this question, you'll evaluate how well a global history, two-level predictor works. You should assume that the branch history register (BHR) is only two entries, and it is used (entirely) to index the PHT. Here you can again assume that the PHT has four entries each using 2b saturating counters initialized to weakly taken. You should assume that the BHR is initialized to 00.

For full credit you must show the BRH and PHT state after branch 9 as well as report the correct prediction rate by completing the prediction table.

PHT

Entry	Current State
0	WT
1	WT
2	WT
3	WT

BHR
00

Number	Branch	Prediction	Outcome	Correct?
1	branch2		NT	
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				

13				
14				

Prediction table

D) [10 Points]

Now consider using a local history two-level predictor.

In this design the branch history table is indexed using the two non-zero LSBs of the branch address (i.e., the 3rd and 4th bits from the right of the branch addresses above). The PHT is indexed using only the value of the BHRs. We'll assume that all BHT entries are initialized to "00" and all PHT entries start in start weakly taken.

For full credit you must show the BHT and PHT state after branches 9 as well as report the correct prediction rate by completing the prediction table.

BHT
00
00
00
00

PHT

Entry	Current State
0	WT
1	WT
2	WT
3	WT

Number	Branch	Prediction	Outcome	Correct?
1	branch2		NT	
2				
3				
4				
5				
6				
7				
8				
9				
10				

11				
12				
13				
14				

Prediction table

E) [3 Points]
Which of the two-level predictors worked best and why?
For full credit you must provide one reason that either the local or global predictor performs better in this example.

Question 3: Virtual Memory [20 Points]

Addressing details: Memory is Byte addressable, has a 14-bit virtual addresses, a 12-bit physical address, and 64 Byte Page size.

The TLB has 16 entries and is 4-way set associative. The cache is direct mapped and has 16 blocks each containing 4 Bytes. (The cache is physically addressed.)

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	1F	1	0A	34	1	02	12	1

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	13	1	3A	00	51	89
9	2D	0	–	–	–	–
A	2D	1	93	15	DA	3B
B	0B	0	–	–	–	–
C	12	1	95	82	10	03
D	16	1	04	96	34	15
E	13	1	83	77	1B	D3
F	14	0	–	–	–	–

A) [9 Points]
Complete the following fields for the data access using the provided virtual address.

Virtual address: 0x02F3

Virtual Page Number:

TLB Index:

TLB Tag:

TLB Hit:

Page Fault:

Physical Page number:

Physical Page Offset:

Cache Offset:

Cache Index:

Cache Tag:

Hit:

Data:

B) [9 Points]
Complete the following fields for the data access using the provided virtual address.

Virtual address: 0x0220

Virtual Page Number:

TLB Index:

TLB Tag:

TLB Hit:

Page Fault:

Physical Page number:

Physical Page Offset:

Cache Offset:

Cache Index:

Cache Tag:

Hit:

Data:

C) [2 Points]
Virtual memory solves a lot of problems, as we discussed in detail during lectures. What is one drawback? (Hint: recall that the Cray-1 did not use virtual memory.)

Question 4: **Short** answer [11 Points]

A) [1 Point]
Please complete your course evaluation on Albert; did you submit it?

A:

B) [5 Points]
Most of class focused on branch prediction. The second challenge of branches is knowing there to go when the branch is predicted takes. The Branch Target Buffer (BTB) is a cache-like structure that is commonly used to predict addresses.
For this question answer:
Does the BTB run in parallel or after branch prediction and why?
How is the BTB populated (i.e., when and what values are put into the BTB)?
What happens on a BTB miss?
Bonus: [2 points] What are "phantom branches"?

A:

C) [5 Points]
In Lab3 you implemented an Inclusive cache, which is a cache that guarantees any data resident in L1 is also in L2. An alternative design is an Explicit cache, where any data in L1 is guaranteed not to be in the L2.

Compare and contrast these two cache designs by answering:
Which cache has a larger effective capacity(i.e., how many unique bits can an inclusive L1/L2 hold verses an exclusive L1/L2 with the same sized data array)?
What happens when the two caches evict a clean block form the L1?
What happens when the two caches evict a dirty block from the L2?
Name one disadvantage of exclusive caches.

A: