Introduction to Computer Architecture - Final Exam Reference Answers (2023 Spring Semester)

Total 10 questions, 100 pts

1. Floating point number format (2 sub-questions, 5 pts. each):
   a. Convert "128_{10}" to the IEEE-754 single-precision floating-point format. Write the answer in hexadecimal.

      0x43000000

   b. In the IEEE-754 single-precision floating-point format, 10000_{10} is 0x461C4000, 20000_{10} is 0x469C4000, and 40000_{10} is 0x471C4000. How 80000_{10} is represented in the single-precision format? Write the answer in hexadecimal.

      0x479c4000

2. The following statements describe the expected changes when designing a CPU with an increased number of pipeline stages. Choose the appropriate description that aligns with the general trends of the changes. Keep in mind that while some statements may not always hold true, try to select the description that reflects the expected trend (5 sub-questions, 2 pts. each):

   a. The area of the CPU would be (increased / decreased / the same) **increased**

      Adding pipeline stages would require more hardware resources → larger area

   b. The clock period would be (increased / decreased / the same) **decreased**

      The reason for adding pipeline stage is to decrease the clock period (clock frequency is increased)

   c. The branch misprediction penalty cycles would be (increased / decreased / the same) **increased**

      With more pipeline stages, the number of stages between the instruction fetch stage and the stage where the branch outcome gets determined becomes larger.

   d. The cache hit time in terms of clock cycles would be (increased / decreased / the same) **increased**

      The cache hit time is determined by the SRAM size. With the same hit "time", the number of cycles for the hit time would be increased since the clock period gets decreased (b).

   e. The address range of the branch target would be (increased / decreased / the same) **the same**

      The address range of the branch target is determined by the ISA. The number of pipeline stage does not change what is defined in the ISA specification.

The following questions, 2 and 3, are based on the basic five-stage pipeline we covered in class. Note: Executing one instruction on this CPU requires 5 cycles. In the absence of hazards, executing two instructions takes 6 cycles in total. With 10 instructions, the execution time would be 14 cycles, still assuming no hazards are present.

3. Data hazard (2 sub-questions, 5 pts. each):

```
A: add x1, x2, x3
B: sub x3, x1, x10
C: sw  x3, 0(x10)
D: lw  x3, 0(x10)
E: and x4, x1, x3
F: sub x4, x1, x3
```

a. Assume the CPU has forwarding and register bypassing to handle data hazards. In the case of a load-use data hazard, the CPU will incur a one-cycle stall. How many cycles are required to execute the instructions above?

11 cycles

6 instructions → 10 cycles if there is no additional penalty

+1 cycle for load-use hazard stall between D-E

b. How many cycles would be required to execute the same instructions if **forwarding logic in the EX stage is not implemented**? Despite the absence of the forwarding logic, assume register bypassing in the ID stage is available.

16 cycles

6 instructions → 10 cycles if there is no additional penalty

+ 2 stall cycles each for B, C, and E. Since no forwarding logic is available, the data hazard can only be resolved by the register bypassing betwwen the WB stage and the ID stage. This add 2 cycles if there is data hazard. This 2 cycles of additional dealy is the same for the load-use data hazard.

4. Control hazard (2 sub-questions, 5 pts. each):

```
    ori  x10, x0,  1
TT: slli x10, x10, 1
    bne  x10, x0,  TT
    add  x0,  x0,  x0
```

a. Your CPU does not have any form of branch prediction, meaning it must stall on every branch instruction. Also, the branch outcome is resolved at the EX stage, not at the ID stage. However, it does have both forwarding and register bypassing mechanisms. How many cycles are required to execute the instructions above?

134 cycles

The loop runs for 32 times, with 31 taken branches and 1 not taken branch. Total executed instructions = 66 instructions (1 ori, 32 slli, 32 bne, 1 add). Without stalls, it would take 70 cycles.

Because there is no branch prediciton, every bne instruction adds 2 cycles. 70 + 32*2 = 134

b. Your CPU uses an "always taken" branch prediction strategy. Same as the previous subquestion, the branch outcome is available at the EX stage instead of the ID stage. How many cycles are required to execute the instructions above?

72 cycles

31 taken branches won't add any penalty. Only the last bne instruction adds 2 cycles.

5. Branch prediction (2 sub-questions, 5 pts each):

- You have a branch predictor with a branch prediction buffer (BPB) that contains only 2 entries.
- In your program, there are four branch instructions: A, B, C, and D. Branch instructions A and B utilize entry 0 in the branch prediction buffer (BPB), while branch instructions C and D use entry 1 in the BPB.
- The execution order and actual outcomes of the branches are provided in the following table (T: taken, N: not taken).

Execution Order	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Instruction	A	C	B	A	A	C	C	A	B	C	D	A	B	B	A	B	C	D	C	A
Branch Outcome	T	T	T	T	T	N	N	T	N	N	T	T	T	N	T	N	N	N	T	T

How many branches were accurately predicted by the branch predictor? If all branches are correctly predicted, the answer would be 20. Please provide the answers for the two different configurations mentioned below.

a. Each entry in the BPB is a one-bit predictor, and all entries are initially set to "0" (not taken). **8**

b. Each entry in the BPB is a two-bit predictor that transitions according to the patterns we covered in the lecture. All entries are initially set to "01" (weakly predict not taken). **12**

Execution Order		1		2		3		4		5		6		7		8		9		10	
Instruction		A		C		B		A		A		C		C		A		B		C	
Branch Outcome		T		T		T		T		T		N		N		T		N		N	
BPB		0		1		0		0		0		1		1		0		0		1	
BPB status	0		1		1		1		1		1		1		1		1		0		0
	0		0		1		1		1		1		0		0		0		0		0
Prediction		N		N		T		T		T		T		N		T		T		N	
Correct?		X		X		O		O		O		X		O		O		X		O	

Execution Order	11		12		13		14		15		16		17		18		19		20	
Instruction	D		A		B		B		A		B		C		D		C		A	
Branch Outcome	T		T		T		N		T		N		N		N		T		T	
BPB	1		0		0		0		0		0		1		1		1		0	
BPB status		0		1		1		0		1		0		0		0		0		1
		1		1		1		1		1		1		0		0		1		1
Prediction	N		N		T		T		N		T		T		N		N		N	
Correct?	X		X		O		X		X		X		X		O		X		X	

8

Execution Order		1		2		3		4		5		6		7		8		9		10	
Instruction		A		C		B		A		A		C		C		A		B		C	
Branch Outcome		T		T		T		T		T		N		N		T		N		N	
BPB		0		1		0		0		0		1		1		0		0		1	
BPB status	01		10		10		11		11		11		11		11		11		10		10
	01		01		10		10		10		10		01		00		00		00		00
Prediction		N		N		T		T		T		T		N		T		T		N	
Correct?		X		X		O		O		O		X		O		O		X		O	

Execution Order	11		12		13		14		15		16		17		18		19		20	
Instruction	D		A		B		B		A		B		C		D		C		A	
Branch Outcome	T		T		T		N		T		N		N		N		T		T	
BPB	1		0		0		0		0		0		1		1		1		0	
BPB status		10		11		11		10		11		10		10		10		10		11
		01		01		01		01		01		01		00		00		01		01
Prediction	N		T		T		T		T		T		N		N		N		T	
Correct?	X		O		O		X		O		X		O		O		X		O	

12

6. Cache memory (2 sub-questions, 5 pts. each):

- 32-bit address size. The size of a cache block is 64 bytes.
- Data capacity is 192KB (=192*1024 bytes) → 192*1024/64 = 3072 blocks
- 12-way set associative cache → 3072/12 = 256 sets
- Write-back, write-allocate scheme. → Needs dirty bit

a. How many bits are needed to represent the "tag" field of a single cache block in this cache?
32bits – 6 bits(offset bits) – 8 bits(set index bits) = **18** bits

b. This cache utilizes a pseudo LRU algorithm for cache block replacement. This pseudo LRU algorithm selects a random block, excluding the most recently used block in a given set. In order to implement this cache, what is the minimum number of bits (not bytes) required, excluding the data array size? Hint: The answer is a 5-digit integer number.
There are 3072 blocks and 256 sets. Each block needs valid bit, dirty bit, and tag. Also, each set needs to remember the most recentely used (MRU) block, which take $[log_2 12]$=4 bits.
3072 * (1+1+18) + 256*4 = 61440 + 1024 = **62464** bits

7. Cache hit/miss (10 sub-questions, 1 pts. each):

- 16-bit address size. The size of a cache block is 8 bytes.
- Data capacity is 16KB (=16*1024 bytes) → 16* 1024 / 8 = 2048 blocks
- 2-way set associative cache, LRU replacement algorithm. = 2048/2 = 1024 sets
- Write-back, no write allocate scheme.

Please indicate **H** (hit) or **M** (miss) for each memory access listed below. Assume the cache is empty (all valid bits are 0) at the beginning.

(tag: 3 bits, index: 10 bits, offset: 3 bits)

Order	Access Type	Address	Tag	Index	
1	Load	0xAFF0	101	**0**111111110	M
2	Store	0xFFF0	111	**1**111111110	M
3	Load	0xAFF4	101	**0**111111110	H
4	Load	0xFFF4	111	**1**111111110	M
5	Store	0xFFF0	111	**1**111111110	H
6	Load	0xAFF0	101	**0**111111110	H
7	Load	0xEFF4	111	**0**111111110	M
8	Store	0xAFF4	101	**0**111111110	H
9	Load	0x6FF0	011	**0**111111110	M
10	Load	0xEFF0	111	**0**111111110	M

Order	Access Type	Address	Tag	Index	Cache state after access (Bold is LRU)	Result
1	Load	0xAFF0	101	**0**111111110	0xAFF0	M
3	Load	0xAFF4	101	**0**111111110	0xAFF0	H
6	Load	0xAFF0	101	**0**111111110	0xAFF0	H
7	Load	0xEFF4	111	**0**111111110	**0xAFF0**, 0xEFF0	M
8	Store	0xAFF4	101	**0**111111110	0xAFF0, **0xEFF0**	H
9	Load	0x6FF0	011	**0**111111110	**0xAFF0**, 0x6FF0	M
10	Load	0xEFF0	111	**0**111111110	0xEFF0, **0x6FF0**	M

Order	Access Type	Address	Tag	Index	Cache state after access (Bold is LRU)	Result
2	Store	0xFFF0	111	**1**111111110	- (no write allocate)	M
4	Load	0xFFF4	111	**1**111111110	0xFFF4	M
5	Store	0xFFF0	111	**1**111111110	0xFFF4	H

8. Calculate the CPI of the following processors (2 sub-questions, 5 pts each):

   Common conditions:

   - Base CPI is 1
   - Instruction mix: 40% load/store, 10% branch, 50% arithmetic/logic
   - Forwarding and register bypassing available. No load-use data hazard stalls

   a. What would be the average CPI of the following CPU A configuration?
   - Cache:
     - L1 D-cache miss rate = 5%
     - L1 I-cache miss rate = 10%
     - L1 cache miss penalty (same for all miss types) = 40 cycles
   - No virtual address → physical address translation
   - No branch prediction. Branch penalty of 1 cycle on every branch.

   Cache miss penalty: (100%*10 % + 50%* 5%) = 4.8

   Branch missprediction penamty: 10%*100%*1 = 0.1

   Total CPI: 1 + 4.8 + 0.1 = **5.9**

   b. What would be the average CPI of the following CPU B configuration?
   - Cache:
     - L1 D-cache miss rate = 8%
     - L1 I-cache miss rate = 8%
     - L1 cache miss penalty (same for all miss types) = 10 cycles
     - L2 cache miss rate (local) = 15%
     - L2 cache miss penalty (same for all miss types) = 40 cycles
   - TLB (TLB is used for load, store, and instruction fetch. Assume no page fault):
     - TLB miss rate = 1%
     - TLB miss penalty is the same as executing one load instruction (i.e., even when there is a TLB miss, if the corresponding PTE can be loaded from the data cache, no penalty is given.)
   - Branch prediction:
     - Misprediction rate = 3%
     - Misprediction penalty = 4 cycles

   Cache miss penalty per instruciton fetch: 8%*(10+15%*40) = 1.28

   Cache miss penalty per data access: 8%*(10+15%*40) = 1.28

   Cache miss penalty per instrction: 100%*1.28 + 40%*1.28 = 1.28 + 0.512 = 1.792

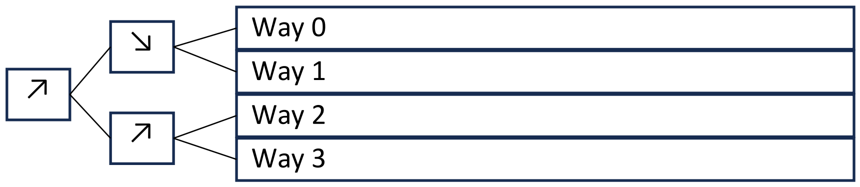
   TLB miss penalty per instruction: (100%*1% + 40%*1%) *1.28 = 0.01792

   Branch missprediction penamty: 10%*3%*4 = 0.012

   Total CPI: 1 + 1.782 + 0.01792 + 0.012 = **2.82192**

9. Floating-point True/False questions (5 sub-questions, 2 pts. each):
   a. By allocating additional bits for the exponent field, we can expand the range of numbers that can be represented. **T**
      The expressible range of numbers is determined by the exponent field.
   b. Numbers represented in floating-point format has higher precision compared to numbers represented in integer format. **F**
      Counter example: 200000001_{10} can be preciesely represented in 32-bit integer format, but this number cannot be correctly represented in the single-precision floating pointe format.
   c. If a number cannot be precisely represented in the binary floating-point format, the resulting number represented in the binary floating-point format is larger than the original number. **F**
      The represented number may be smaller than the original depending on the rounding operation.
   d. If the CPU does not support floating-point operations, we can utilize software logic (i.e., software functions) to emulate the functionality of the floating-point hardware components. **T**
      Although it can be slow, floating-point operations can be handled by software. This is how floating-point numbers are handled on small, embedded systems without a dedicated floating-point hardware.
   e. By examining the number represented in binary format, we can distinguish whether it is represented in the integer format or in the floating-point format. **F**
      The 32-bit binary data itself does not say anything about the format. For example, "0x3f800000" may be "1.0" in single-precision floating-point format, but it can also be 1065353216_{10} represented in integer format.

10. You have a 4-way set-associative cache that implements a pseudo-LRU replacement algorithm. The pseudo-LRU algorithm used in this cache employs a tree-like structure, assigning an LRU bit per group, where each group contains two cache blocks. Additionally, there is another LRU bit that selects the least recently used group. In total, there are three LRU bits per set.



When using this LRU mechanism, the selectied victim may be or may not be the true LRU entry. For exmaple, suppose the following case. After accessing ways in the ordr of 0 → 1 → 2 → 3, if the algorithm selects a victim for a new block, the true LRU entry, 0, will be selected.

However, depending on the access order, the selected victim may not be the true LRU entry. Please give an example access order where the following replacement operation on this cache would choose an entry that is not the true LRU entry.
Basically, if the oldest one and the youngest one comes from the same group, the oldest one is never selected as the victim here. The following are possible cases:
0 → 2 → 3 → 1
0 → 3 → 2 → 1
1 → 2 → 3 → 0
1 → 3 → 2 → 0
2 → 0 → 1 → 3
2 → 1 → 0 → 3
3 → 0 → 1 → 2
3 → 1 → 0 → 2