Final Exam

CSC 252
8 May 2019
Computer Science Department
University of Rochester

Instructor: Yuhao Zhu
TAs: Jessica Ervin, Yu Feng, Max Kimmelman, Olivia Morton, Yawo Alphonse Siatitse, Yiyang Su, Amir Taherin, Samuel Triest, Minh Tran

Name: _____

Problem 0 (3 points):	_____
Problem 1 (17 points):	_____
Problem 2 (10 points):	_____
Problem 3 (20 points):	_____
Problem 4 (20 points):	_____
Problem 5 (30 points):	_____
Total (100 points):	_____

Remember "**I don't know**" is given 15% partial credit, but you must erase everything else. This does not apply to extra credit questions.

Your answers to all questions must be contained in the given boxes. The lengths of the boxes should be more or less indicative of the lengths of your answers. Use spare space to show all supporting work to earn partial credit.

You have 2 hours 45 minutes to work.

Please sign the following. I have not given nor received any unauthorized help on this exam.

Signature:_____

GOOD LUCK!!!
And have a great summer break.

Problem 0: Warm-up (3 Points)

Some people say 252 shouldn't be required for the BS. What say you? (Hint: the correct answer is YES, IT SHOULD BE, but we are not necessarily looking for the correct answer here.)

Problem 1: Floating-Point Arithmetics (17 points)

In this problem, we assume that IEEE decided to add a new N-bit representation, with its main characteristics consistent with the other IEEE standards. This N-bit representation could represent the value $12\frac{1}{8}$ exactly, but cannot represent the value $34\frac{1}{4}$ exactly. The smallest positive normalized value it can represent is 2^{-62}.

Part a) (3 points) Put $12\frac{1}{8}$ in binary normalized form.

1.100001 x 2^3

Part b) (3 points) Of the N bits, how many bits are fraction bits?

6

Part c) (3 points) Of the N bits, how many bits are exponent bits?

7

Part d) (4 points) What is N?

14

Part e) (4 points) What is the bias?

63

Problem 2: Pipelining (10 points)

A pipelined processor has 5 stages with delays as follows:

Stage 1	28 ns
Stage 2	59 ns
Stage 3	23 ns
Stage 4	34 ns
Stage 5	36 ns

The delay of pipeline registers between two stage is 1 ns.

Part a) (4 points) What is the cycle time of this processor? Recall the cycle time refers to the delay of a single clock cycle.

60 ns

Part b) (6 points) Now we execute 8 instructions on this pipelined processor. What is the speedup that the pipeline achieves compared to a non-pipelined design? Assume that there are no pipeline stalls. Show your work to earn partial credit.

Without pipeline: Delay is (28 + 59 + 23 + 34 + 36) * 8
With pipeline: Delay is 60 * 5 + 7 * 60

Speedup = 2

Problem 3: Assembly Programming (20 points)

Clark Kent has taken CSC 252 and is working on research project on ISA. Clark defines Kryptonian numbers and Xenonian numbers as follows.

- A binary number is said to be Kryptonian if and only if there are more 0's than 1's when we discard the leading zeros. For example, 100100_2 is Kryptonian and 000001111_2 is not.
- A binary number is said to be Xenonian if and only if there are exactly 4 1's. For example, 110110_2 is Xenonian and 11111_2 is not.

Part a) (8 points) Consider the 32-bit two's complement representation. Is 18_{10} a Kryptonian, and is -18_{10} a Xenonian? Show your work to earn partial credit.

18 is a Kryptonian -18 is not a Xenonian

Part b) (12 points) Clark wants to add two new flags to the standard x86-64 ISA.

- Krypton Flag: set if a number is Kryptonian.
- Xenon Flag: set if a number is Xenonian.

According to Clark's ISA specification, the `addl` instruction sets these two flags according to the addition result, and the `movl` instruction sets these two flags according to the content of the source operand, i.e., the content that is being moved. No other instructions change these two flags. Other flags are set as in the standard x86-64 ISA.

Further, Clark implemented the following two instructions:

- `jkr addr`: jump to the address specified by `addr` if the Krypton flag is set and the Overflow flag is not set.
- `jxn addr`: jump to the address specified by `addr` if the Xenon flag is set and the Overflow flag is not set.

He wrote the following function in assembly language to test his work. Recall from the programming assignments that `movl` instructions move a 4 byte integer to the destination, and the size of the data moved by the `mov` instructions is implicit in the operands. We assume an assembly syntax where the source is the first operand and the destination is the second operand.

```
00000000000005fa <foo>:
 5fa: 55                    push   %rbp
 5fb: 48 89 e5              mov    %rsp,%rbp
```

```
5fe: 89 7d ec              mov    %edi,-0x14(%rbp)
601: 48 89 75 e0           mov    %rsi,-0x20(%rbp)
605: c7 45 f8 00 00 00 00  movl   $0x0,-0x8(%rbp)
60c: c7 45 fc 12 00 00 00  movl   $0x12,-0x4(%rbp)
613: eb 0e                 jmp    61f
615: 7e 04                 jkr    61b
617: 83 45 f8 01           addl   $0x1,-0x8(%rbp)
61b: 83 45 fc 01           addl   $0x1,-0x4(%rbp)
61f: 83 7d fc 35           cmpl   $0x35,-0x4(%rbp)
624: 7e ec                 jle    615 <foo+0x1b>
626: b8 00 00 00 00        mov    $0x0,%eax
62b: 5d                    pop    %rbp
62c: c3                    retq
```

Unfortunately, he made a small mistake in his implementation such that the machine will jump to the designated address on any `jkr`/`jxn` instruction regardless of the flags.

(4 points) On this faulty machine, what is the value of `-0x18(%rsp)` after the `foo` function returns? Hint: what do `pop` and `retq` do to `%rsp`?

0

(4 points) He then fixes the mistake, and runs the same program again on this correct machine. What is the value of `-0x18(%rsp)` now after the `foo` function returns?

26

(4 points) To test the Xenon flag and the `jxn` instruction, Clark replaces the instruction at `0x61f` with four `nop` instructions, and replaces the instruction at `0x624` with `jxn 615`. What is the value of `-0x18(%rsp)` after the `foo` function returns?

3

Problem 4: Cache (20 points)

Solar radiation can randomly flip bits in the computer system. Therefore, a cache on a space-faring vehicle, which is exposed to solar radiation, utilizes error-correcting codes (ECC) *for each of its cache blocks* to detect if bits have been flipped. These ECC bits add to the overhead of the cache, in addition to the usual overhead bits such as valid bits and tags, etc.

On a memory access, the cache operates as normal, but in addition to checking hit/miss it will also check if the content in the cache block has been corrupted. This is done by checking the ECC bits. How exactly ECC bits are used to detect corruption is irrelevant to this problem. If the ECC bits associated with a block indicate that the data in the block is corrupted, that cache access is regarded as a cache miss. For the sake of the problem, assume that the memory is incorruptible.

The physical memory is byte addressable, and is 64 KB in size. Each cache block is 4B, and requires 6 extra bits for the error-correcting codes. The cache is 2-way associative with the LRU replacement policy. The entire cache has an overhead of 3712 bits.

Part a) (4 points) For this cache to function properly, should it use a write-back policy or a write-through policy upon a write hit?

Write through

Part b) (4 points) Determine the number of offset bits.

2

Part c) (4 points) Determine the number of tag bits.

7

Hint: how many bits does each set have to have to implement the LRU replacement policy? Use the box below to show your work to earn partial credit.

Part d) (8 points) Given the following sequence of 9 cache accesses; some cache accesses result in loads from the physical memory. Assume that the initial state of the cache is empty.

Address	**Load From Memory?**
a) 0x3420	Yes
b) 0x3423	Yes
c) 0x062e	Yes
d) 0x1e2f	Yes
e) 0x73ec	Yes
f) 0x062f	No
g) 0x0e2f	Yes
h) 0x1e2e	Yes
i) 0x0e2e	Yes

Determine which accesses were necessarily a result of cache corruption due to solar radiation. Write the letters corresponding to the memory addresses below:

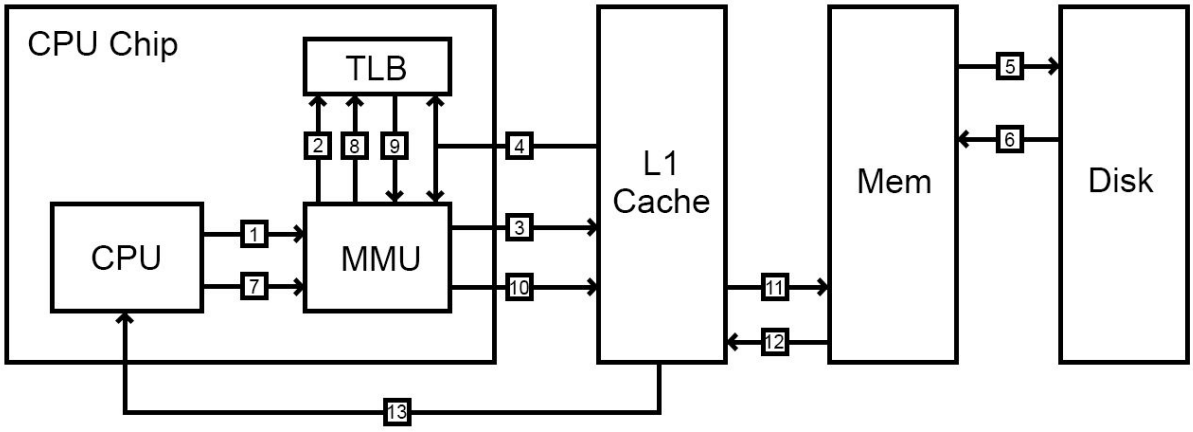
b) and i)

Problem 5: Virtual Memory (30 points)

The diagram below shows the interactions between components of a computer system resulting from a *single* virtual memory access from the CPU. You are given the following information:

- Assume a single-level virtual memory system.
- The virtual address space is 128 KB.
- The physical memory size is 32 KB.
- The L1 cache block size is 2 B.
- The value stored in the Page Table Base Register (PTBR) is 0x260.
- Each Page Table Entry (PTE) takes 2 B and has the following structure:

Valid <1-bit>	Padding of zeros	Physical page number



Part a) (22 points) Fill in the right column of the table below with answers to the questions about each step in the diagram above. "I don't know" is accepted at each entry.

1	CPU reads virtual address 0x_____	0x488
2	MMU checks TLB. Is this a hit or a miss?	miss
3	MMU accesses memory at physical address 0x2F0. Is this a hit or a miss?	hit
4	What is the most significant bit of the data returned to the MMU and TLB?	0
5	What happens at this step? (Answer in 15 words or fewer)	Page fault handler evicts a victim page from physical memory

6	16 bytes are returned.	(no question)
7	CPU reads virtual address 0x_____	0x488
8	MMU checks TLB. Is this a hit or a miss?	hit
9	Data returned from TLB to MMU is 0x_____	0x80A4
10	MMU accesses memory at physical address 0xA48. Is this a hit or a miss?	miss
11	Is this an access to the page table?	no
12	How many bytes of data are returned here?	2B
13	Requested data is returned to the CPU.	(no question)

Part b) (3 points) How many pages does the entire page table occupy?

2^10 = 1024

Part c) (3 points) How many physical memory accesses were made during this single virtual memory access?

2 or 3 (depending on how you count)

Part d) (2 points) What would the system do differently at step 3 and 4 if the L1 cache block size is 1 B?

The system would request and receive two cache blocks from the L1 cache and concatenate them instead of just accessing one cache block in order to get the PTE