Problem Set 4

Due: November 5th @ 4:59 pm (Before class)

Department of Electrical and Computer Engineering
The University of Texas at Austin
EE 382N.1, Fall 2018
Instructor: Dam Sunwoo
TA: Pritesh Chhajed

Instructions

You are encouraged to work on the problem set in small groups (3 or 4 per group) and turn in one problem set for the entire group on Gradescope. Create a copy of this file to fill in your answers within the spaces provided. If you need more space than given box, make a comment there and use extra work space at end of file. There are extra pages provided for the same.

Before submitting, convert it into a PDF file. Remember to put all your names and eid in the box below. The person submitting must choose everyone in the group.

You will need to refer to the assembly language handout and the LC-3b ISA, microarchitecture, and state diagram documents on the course website.

Student names and EID

Questions

Problem 1

We have been referring to the LC-3b memory as 2^16 bytes of memory, byte-addressable. This is the memory that the user sees, and may bear no relationship to the actual physical memory.

A. Suppose that the actual physical address space is 8K bytes, and our page size is 512 bytes. What is the size of the PFN?

B. Suppose we have a virtual memory system in which virtual memory is divided into User Space and System Space, and System Page Table remains resident in physical memory. System space includes trap vector table, interrupt vector table, operating system and supervisor stack as shown in Figure A.1 in Appendix A. The rest of the address space in Figure A.1 is user space. If each PTE contained, in addition to the PFN, a Valid bit, a modified bit, and two bits of access control, how many bits of physical memory would be required to store the System Page Table?

Problem 2

An ISA supports an 8-bit, byte-addressable virtual address space. The corresponding physical memory has only 128 bytes. Each page contains 16 bytes. A simple, one-level translation scheme is used and the page table resides in physical memory. The initial contents of the frames of physical memory are shown below.

Frame Number	Frame Contents
0	empty
1	Page 13
2	Page 5
3	Page 2
4	empty
5	Page 0
6	empty

7	Page Table

A three-entry Translation Lookaside Buffer that uses LRU replacement is added to this system. Initially, this TLB contains the entries for pages 0, 2, and 13. For the following sequence of references, put a circle around those that generate a TLB hit and put a rectangle around those that generate a page fault. What is the hit rate of the TLB for this sequence of references? (Note: LRU policy is used to select pages for replacement in physical memory.)

References (to pages): 0, 13, 5, 2, 14, 14, 13, 6, 6, 13, 15, 14, 15, 13, 4, 3.

1. At the end of this sequence, what three entries are contained in the TLB?

2. What are the contents of the 8 physical frames?

Frame Number	**Frame Contents**
0	
1	
2	
3	
4	
5	
6	
7	

Problem 3

A little-endian machine with 64KB, byte addressable virtual memory and 4KB physical memory has two-level virtual address translation similar to the VAX. The page size of this machine is 256 bytes. Virtual address space is partitioned into the P0 space, P1 space, system space and reserved space. The space a virtual address belongs to is specified by the most significant two bits of the virtual address, with 00 indicating P0 space, 01 indicating P1 space, and 10 indicating system space. Assume that the PTE is 32 bits and contains only the Valid bit and the PFN in the format V0000000..000PFN.

For a single load instruction the physical memory was accessed three times, *excluding instruction fetch*. The first access was at location x108 and the value read from that location (`x10B`,`x10A`,`x109`,`x108`) was `x80000004`. Hint: What does this value mean?

The second access was at location x45C and the third access was at location x942.
If SBR = `x100`, P0BR = `x8250` and P1BR = `x8350`,

1. What is the virtual address address corresponding to physical address `x45C`?

2. What is 32 bit value read from location `x45C`?

3. What is the virtual address corresponding to physical address `x942`?

Problem 4

Note: In this problem, the user and system virtual address spaces are not sized equally (the system virtual address space is 1/4 of the total virtual address space, and the user virtual address space makes up the other 3/4). Thus you need to include the address region bits in your calculation of the user space virtual page number. To make it easier for the machine to index into the user space page table, PTBR points to 0x380, which is at an offset of -0x20 from the actual first entry in the user space page table at 0x3A0. To index into the user space page table, add (user space virtual page number * PTE size) to the PTBR. (Why does this work?)

Consider a processor that supports a 9-bit physical address space with byte addressable memory. We would like the processor to support a virtual memory system. The features of the virtual memory system are:

```
Virtual Memory Size : 4 Kbytes (12 bit address-space)
 Page Size          : 32 bytes
 PTBR               : 0x380
 SBR                : 0x1E0
```

The virtual memory is divided into two spaces: system space and user space. System space is the first kilobyte of the virtual address space (i.e., most significant two bits of the virtual address are 00). The rest of the virtual memory is user space. The system page table remains resident in physical memory. Each PTE contains, in addition to the PFN, a Valid bit, a modified bit and 2 bits for access control. The format of the PTE is

Valid	Modified	Access Control	PFN

(Valid bit is the most significant bit of the PTE and the PFN is stored in the least significant bits.)

1. How many virtual pages does the system accommodate?

2. What is the size of the PFN? How big is the PTE?

3. How many bytes are required for storing the entire user space pagetable? How many pages does user space page table occupy?

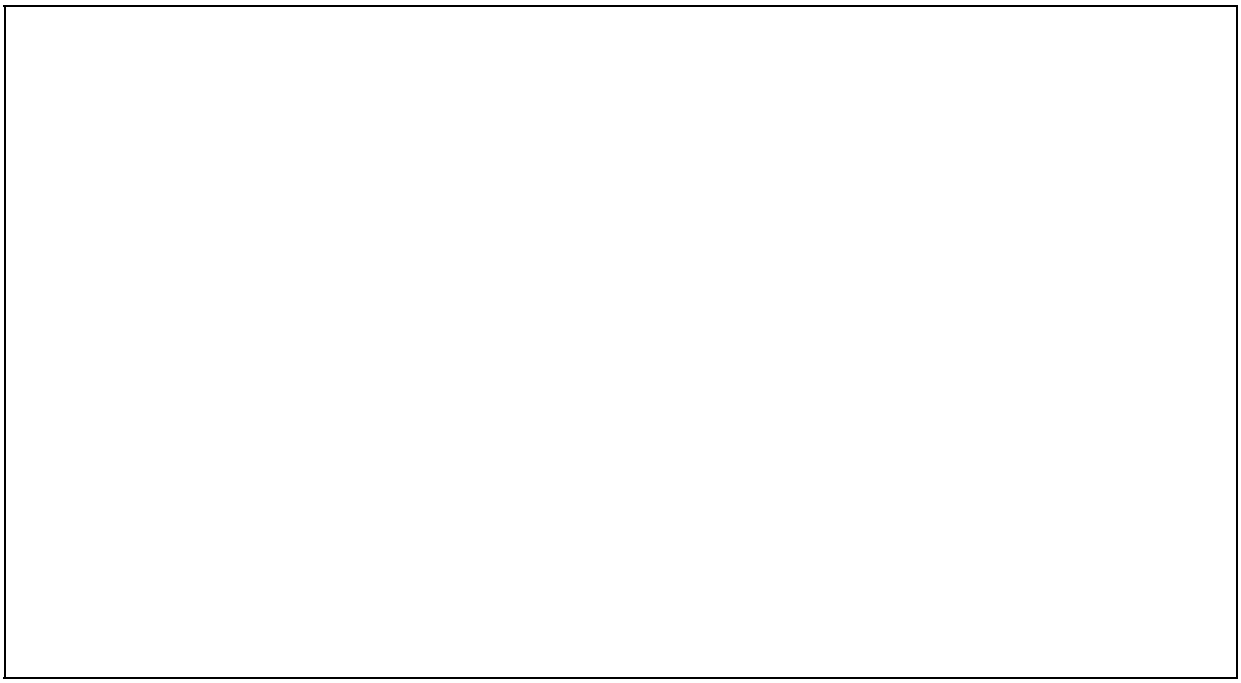
4. Since the user space page table can occupy a significant portion of the the physical memory, this system uses a 2 level address translation scheme, by storing the user space Page Table in virtual memory (similar to VAX).

   Given the virtual address 0x7AC what is the Physical address?

   The following table shows the contents of the physical memory that you may need to do the translation:

Address	Data
x1F8	xBA
x1F9	xBB
x1FA	xBC
x1FB	xBD
x1FC	xBE
x1FD	xB8
x1FE	xB7
x1FF	xB6

Address	Data
x118	x81
x119	x72
x11A	x65
x11B	x34
x11C	x97
x11D	x83
x11E	xC6
x11F	xB2



Problem 5

The virtual address of variable X is `x3456789A`. Find the physical address of X. Assume a Virtual Memory model similar to VAX.

Remember that in VAX each Virtual Address consists of:

- 2 bits to specify the Address Space
- 21 bits to specify Virtual Page Number
- 9 bits to specify the byte on the page

You will need to know the contents of P0BR: `x8AC40000` and SBR: `x000C8000`.

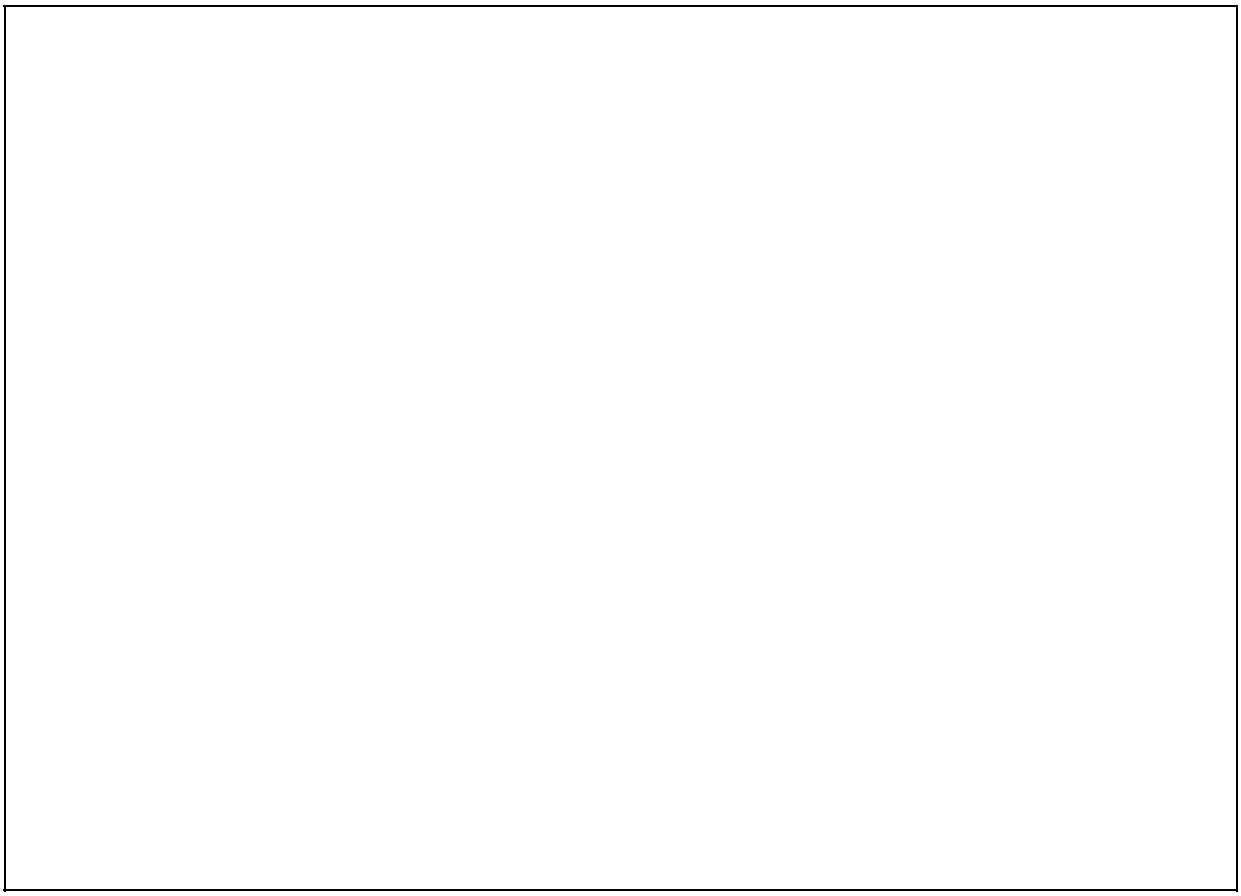
You will also need to know the contents of the following physical memory locations:

```
x1EBA6EF0: x80000A72
x0022D958: x800F5D37
```

Some intermediate questions to help you:

- What virtual page of P0 Space is X on?
- What is VA of the PTE of the page containing X?

- What virtual page of System Space is this PTE on?
- What is the PA of the PTE of this page of System Space?
- What is the PA of the PTE of the page containing X?



Problem 6

Given a VAX-like virtual memory with 8b virtual addresses where the most significant bit indicates P0 (MSB == 0) or system segment (MSB == 1), the next 4 bits are the virtual page number and the last 3 bits of address are the page offset. The machine is byte-addressable.

The format of a page table entry (PTE) is shown below.

7	6	5	4	3	2	1	0
Valid	0	Protection		Physical Frame Number			

Protection bits:
00: none
01: read-only
10: read-write
11: (invalid)

The following single instruction is executed
MEM[R0] = 3;
The state of the physical memory after executing that instruction is shown below. The blank frames represent frames to which no pages are mapped. No page faults occurred during the program run. P0LR and SLR are 1, and P1LR is 0.

a. What is the physical address containing 3?

b. What is the PTE of the page containing 3?

c. What is SBR?

d. What is P0BR?

e. What is the virtual address of the location containing 3?

Frame	Addr	Dec	Hex	Bin	Frame	Addr	Dec	Hex	Bin
0	0				4	32	30	0x1e	0b0001 1110
	1					33	163	0xa3	0b1010 0011
	2					34	82	0x52	0b0101 0010
	3					35	10	0x0a	0b0000 1010
	4					36	105	0x69	0b0110 1001
	5					37	93	0x5d	0b0101 1101
	6					38	56	0x38	0b0011 1000
	7					39	4	0x04	0b0000 0100
2	16	37	0x25	0b0010 0101	6	48	28	0x1c	0b0001 1100
	17	123	0x7b	0b0111 1011		49	148	0x94	0b1001 0100
	18	96	0x60	0b0110 0000		50	21	0x15	0b0001 0101
	19	31	0x1f	0b0001 1111		51	86	0x56	0b0101 0110
	20	36	0x24	0b0010 0100		52	93	0x5d	0b0101 1101
	21	11	0x0b	0b0000 1011		53	80	0x50	0b0101 0000
	22	19	0x13	0b0001 0011		54	11	0x0b	0b0000 1011
	23	121	0x79	0b0111 1001		55	88	0x58	0b0101 1000
3	24	20	0x14	0b0001 0100	7				
	25	90	0x5a	0b0101 1010					
	26	3	0x03	0b0000 0011					
	27	81	0x51	0b0101 0001					
	28	18	0x12	0b0001 0010					
	29	31	0x1f	0b0001 1111					
	30	27	0x1b	0b0001 1011					
	31	24	0x18	0b0001 1000					

Problem 7

A computer has an 8KB write-through cache. Each cache block is 64 bits, the cache is 4-way set associative and uses a victim/next-victim pair of bits for each block for its replacement policy. Assume a 24-bit address space and byte-addressable memory. How big (in bits) is the tag store?

Problem 8

An LC-3b system ships with a two-way set associative, write back cache with perfect LRU replacement. The tag store requires a total of 4352 bits of storage. What is the block size of the cache? Please show all your work.

Hint: **$4352 = 2^{12} + 2^{8}$**.

Problem 9

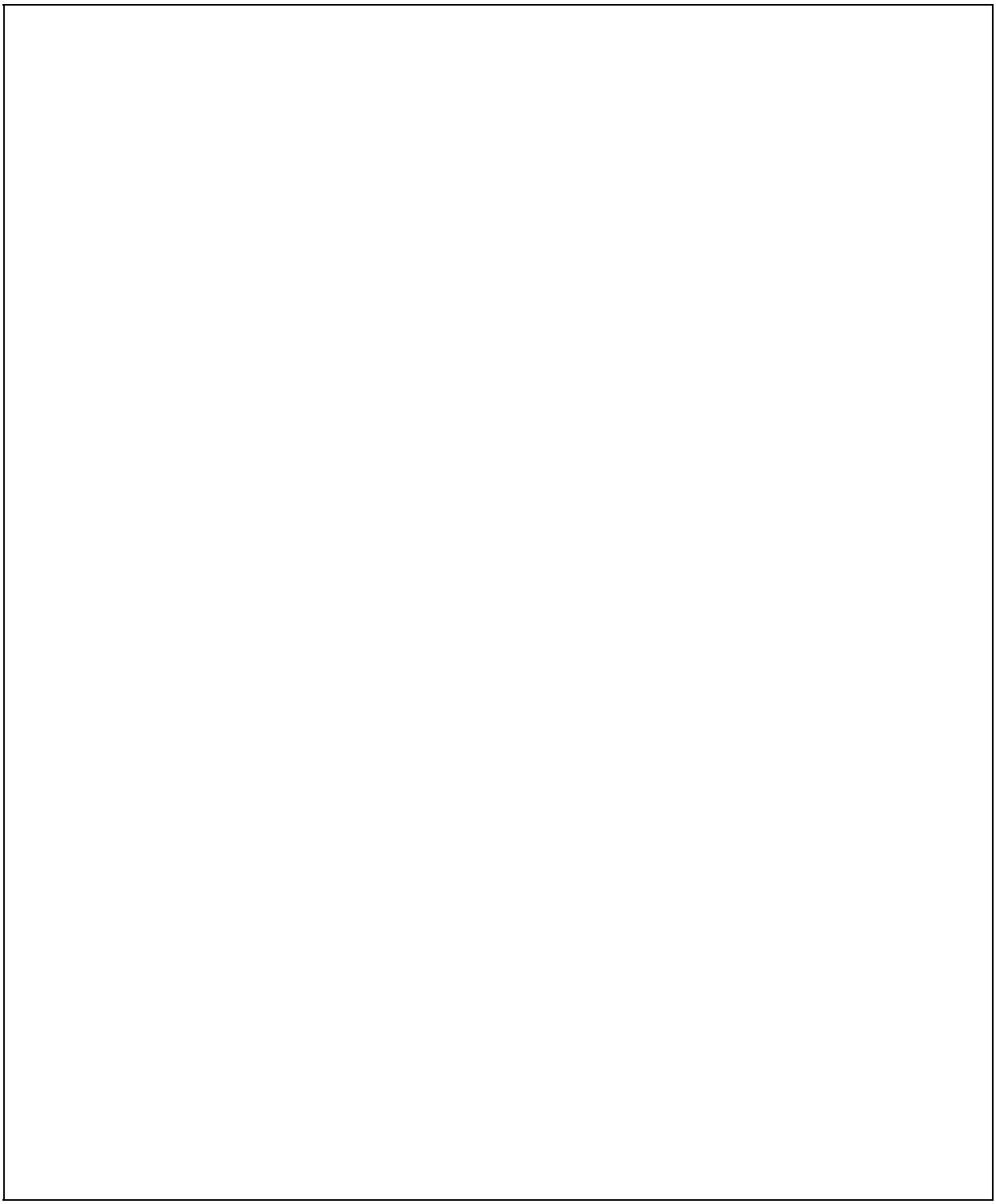
Hamacher, pg.255, question 5.13. A byte-addressable computer has a small data cache capable of holding eight 32-bit words. Each cache block consists of one 32-bit word. When a given program is executed, the processor reads data from the following sequence of hex addresses:

`200, 204, 208, 20C, 2F4, 2F0, 200, 204, 218, 21C, 24C, 2F4`

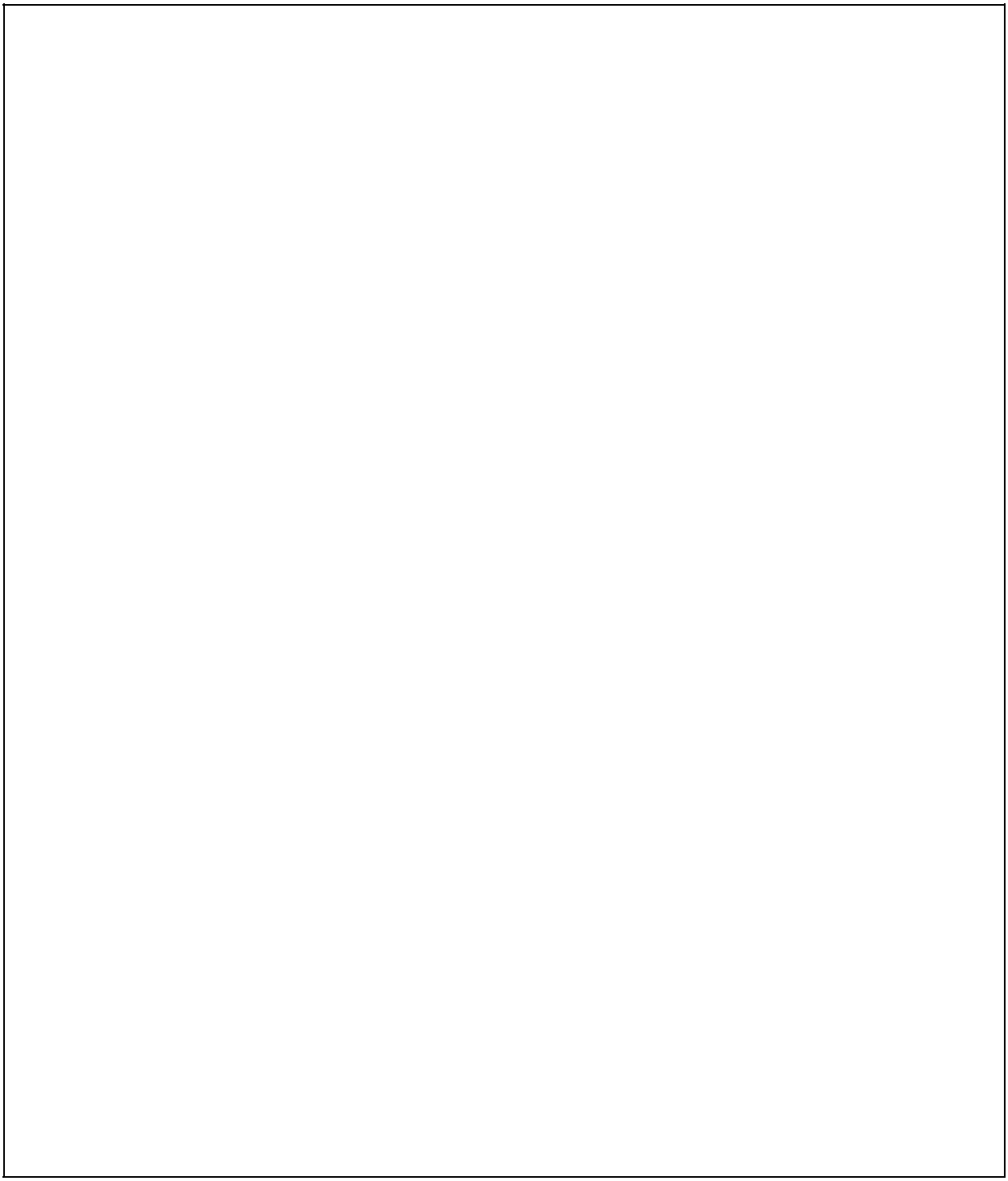
This pattern is repeated four times.

1. Show the contents of the cache at the end of each pass throughout this loop if a direct-mapped cache is used. Compute the hit rate for this example. Assume that the cache is initially empty.

2. Repeat part (a) for a fully-associative cache that uses the LRU-replacement algorithm.



3. Repeat part (a) for a four-way set-associative cache that uses the LRU replacement algorithm.



Problem 10

Below, we have given you four different sequences of addresses generated by a program running on a processor with a data cache. Cache hit ratio for each sequence is also shown below. Assuming that the cache is initially empty at the beginning of each sequence, find out the following parameters of the processor's data cache:

- Associativity (1, 2, or 4 ways)
- Block size (1, 2, 4, 8, 16, or 32 bytes)
- Total cache size (256B, or 512B)
- Replacement policy (LRU or FIFO)

Assumptions: all memory accesses are one byte accesses. All addresses are byte addresses.

Number	Address Sequence	Hit Ratio
1	0, 2, 4, 8, 16, 32	0.33
2	0, 512, 1024, 1536, 2048, 1536, 1024, 512, 0	0.33
3	0, 64, 128, 256, 512, 256, 128, 64, 0	0.33
4	0, 512, 1024, 0, 1536, 0, 2048, 512	0.25

The following problems are meant to help you study for the test. These problems do NOT need to be turned in.

Problem 1 (Ungraded)

Let's say we added a virtual memory system to the LC-3b. Which instructions can possibly generate a page fault? What is the maximum number of page faults an instruction can possibly generate while it is being processed? Which instructions can possibly generate that maximum number of page faults?

Assume that the virtual memory system added uses a one-level translation scheme and the page table is always resident in physical memory.

An instruction is said to generate a page fault if a page fault occurs at any time during the processing of that instruction.

Problem 2 (Ungraded)

You will be given a cache simulator (just the executable) with a hard-coded configuration. Your job is to determine the configuration of the cache. The simulator takes a trace of memory addresses as input and provides a hit ratio as output. Find the following:

* Associativity (1, 2, 4, or 8 ways)
* Block size (1, 2, 4, 8, 16, or 32 bytes)
* Total cache size (256B, 512B, or 1024B)
* Replacement policy (LRU or Pseudo-LRU)

Show the traces you used to determine each parameter of the cache. Assumptions:All memory accesses are one byte accesses. All addresses are byte addresses.

Simulator
The syntax for running the program is:
./cachesim.linux <trace.txt>
The traces are just text files with one integer memory address per line. For example, the following trace would cause conflict misses in a direct-mapped, 256B cache:
0
256
512
0
Simulator for Linux
After downloading the file, please do **"chmod 700 cachesim.linux"**

Space for Extra work