Department of Electrical and Computer Engineering

The University of Texas at Austin

Name: Xinyuan (Allen) Pan xp572
EE 460N, Spring 2017
Problem Set 4
Due: April 3, before class
Yale N. Patt, Instructor
Chirag Sakhuja, Sarbartha Banerjee, Jon Dahm, Arjun Teh, TAs

Instructions

You are encouraged to work on the problem set in groups and turn in one problem set for the entire group. The problem sets are to be submitted on Canvas. Only one student should submit the problem set on behalf of the group. The only acceptable file format is PDF. Include the name of all students in the group in the file.

You will need to refer to the assembly language handouts and the LC-3b ISA on the course website.

/* This problem has been moved here from the previous problem set */
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?

A. Size of PFN is 4 bits
B. Size of PTE = 1+1+2+4 = 8 bits
So number of virtual pages is $(3 \times 2^{12}) / 2^{9} = 24$
Size of System Page Table = 24 * 8 = 192 bits

/* This problem has been moved here from the previous problem set */

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?

(1) TLB contains entries for page 3, 4, 13.
(2) Frame 0 Page 6
Frame 1 Page 13
Frame 2 Page 3
Frame 3 Page 2
Frame 4 Page 14
Frame 5 Page 4
Frame 6 Page 15
Frame 7 Page Table

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`?

(1) X825C
(2) X80000009
(3) X0342

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 this correspond to?
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

Address	Data
x118	x81
x119	x72
x11A	x65
x11B	x34

x1FC	xBE	x11C	x97
x1FD	xB8	x11D	x83
x1FE	xB7	x11E	xC6
x1FF	xB6	x11F	xB2

1) Number of virtual pages = 2^{12} / s^{5} = 2^{7}
2) Number of physical frames = 2^{9} / 2^{5} = 2^{4} frames
   So PFN size = 4 bits
   PFE size = 1 + 1 + 2 + 4 = 8 bits
3) Size for user page table = 96 bytes
   Correspond to 3 pages
4) x6C

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?

5) X is on page x1A2B3C of P0
   VA of PTE of the page containing X is x8B2CACF0
   Virtual page of system space this PTE is on is x59656
   PA of PTE of this page of system space is x22D958
   PA of PTE of this page containing X is x1EBA6EF0
   Thus physical memory of X is x14E49A

Problem 6

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

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.

Every instruction can possibly generate a page fault.
RTI can generate the maximum number of faults – 3 faults.

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?

8KB / (4 x 8B) = 256 = 2^8 sets in cache
so 3 bits for cache line index and 8 bits for cache set index
24-8-3 = 13 bits   13+2+1 = 16 bits tag store for each cache line
size of tag store = 16 x 256 x 4 = 16384 bits

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**.

Size of tag store = 2^{12} + 2^8 = 17 * 2^8 = number of sets x bits per set
Address space is 16-bit = a (tag bits) + b (index bits) + c (bits for block)
So number of sets = 2^8 = 256 bits (b = 8)
Bits per set = 17
Number of bits per set = 1 for LRU + 2 for valid bits + 2 for dirty bits + 2*a = 5 + 2a = 17bits
So a = 6, c = 16-8-6 = 2
So Cache block size = 2^2 = 4

Problem 9

Based on Hamacher et al., p. 255, question 5.18. You are working with a computer that has a first level cache that we call L1 and a second level cache that we call L2. Use the following information to answer the questions.

- The L1 hit rate is 0.95 for instruction references and 0.90 for data references.
- The L2 hit rate is 0.85 for instruction references and 0.75 for data references.
- 30% of all instructions are loads and stores.
- The size of each cache block is 8 words.
- The time needed to access a cache block in L1 is 1 cycle and the time needed to access a cache block in L2 is 6 cycles.
- The accesses to the caches and memory are done sequentially. If there is a miss in the L1 and a hit in the L2 then the total latency is 7 cycles.
- Memory is accessed only if there is a miss in both caches.
- The width of the memory bus is one word.
- It takes one clock cycle to send an address to main memory.
- It takes 20 cycles to access the main memory.
- It takes one cycle to send one word from the memory to the processor. Thus the total latency to get a word from memory to the processor is 22 cycles.
- The bus allows sending a new address to memory in the same cycle that data is sent from memory to the processor.
- Assume the data is accessible to the processor only AFTER the whole cache block has been brought in from the memory, and buffered on the processor chip. The processor can then access the data independent of and during the cache fill.

1. What is the average access time per instruction (assume no interleaving)?
2. What is the average access time per instruction if the main memory is 4-way interleaved?
3. What is the average access time per instruction if the main memory is 8-way interleaved?
4. What is the improvement obtained with interleaving?

1) 4.32 cycles
2) 2.47 cycles
3) 2.22 cycles
4) 4 way: (4.32-2.47)/4.32 = 42.7%
   8 way: (4.32-2.22)/4.32 = 48.6%

Problem 10

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.

1)

Valid	**Tag**	**Data**			
1	**0010000**	**203**	**202**	**201**	**200**
1	**0010000**	**207**	**206**	**205**	**204**
1	**0010000**	**20B**	**20A**	**209**	**208**
1	**0010010**	**24F**	**24E**	**24D**	**24C**
1	**0010111**	**2F3**	**2F2**	**2F1**	**2F0**
1	**0010111**	**2F7**	**2F6**	**2F5**	**2F4**
1	**0010000**	**21B**	**21A**	**219**	**218**
1	**0010000**	**21F**	**21E**	**21D**	**21C**

Hit rate = 33/48

Problem 11

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

Associativity: 4 way
Block size: 8 bytes
Cache size: 256B
Replacement policy: LRU

Problem 12

In class, we discussed the asynchronous finite state machine for the device controller of an input-output device within the context of a priority arbitration system. Draw the state diagram for this device controller (as drawn in lecture), identify the input and output signals, and briefly explain the function of each input and output signal.

In class, we mentioned two race conditions that existed in the finite state machine. Describe the race conditions and show what simple modifications can be made to eliminate them.