Department of Electrical and Computer Engineering
University of Texas at Austin

EE460N Spring 2021
Y. N. Patt, Instructor
Siavash Zangeneh, Ben Lin, Juan Paez, TAs
Exam 2
April 21st, 2021

Name:

EID:

Problem 1: 20 points
Problem 2: 10 points
Problem 3: 20 points
Problem 4: 25 points
Problem 5: 25 points
Total: 100 points

Note: Please be sure that your answers to all questions (and all supporting work that is required) are contained in the space provided.

Please read the following sentence, and if you agree, sign/print your name where requested: **I have not given or received any unauthorized help on this exam.**

Signature:

Good Luck!

General Instructions:

1. You are free to use anything in the Handouts section of the course website that is listed under "Powerpoint Presentations", "Other Handouts", "LC-3b Handouts", and "Spring 2021 Exam 2 Buzzwords." In particular, Appendix A and Appendix C may be of use. Anything else from the course website is not allowed. Anything from any textbooks or the Internet is also not allowed and considered unauthorized access.
2. Use of a calculator is not required but is permitted.
3. If you have any questions, join the class Zoom link and ask a TA. Do not stay on the Zoom call during the exam unless you have questions.
4. Announcements will be posted here. Check this page periodically throughout the exam.
5. You will take the exam by editing a Google Doc. Unlike exam 1, we will not accept handwritten answers by either printing the exam or editing a pdf using a tablet.
6. Read the instructions below.
7. **You are required to stop working on the exam promptly at 6:30 PM.**

Editing a Google Doc:

1. Open the Google Doc version of the exam from here.
2. Save a copy of the document to your Google Drive. Click "File"-> "Make a copy."
3. While working on the exam, **DO NOT expand any boxes that are given to you.** Feel free to show your work in the available space. If you need more space, you are writing too much.
4. When you are ready to submit your exam, click "File"-> "Print" and select "Save as PDF". Save the edited PDF as "Exam 1 <your name>".
5. Upload the PDF to Gradescope **by 6:35 PM.**
6. If you fail to upload your exam to Gradescope on time, email your exam to the TAs as soon as possible. Late penalties may apply.

Problem 1 (20 points): Answer each question in 20 words or fewer. Note: For each of the four answers below, if you leave the box empty, you will receive one point of the five.

Part a (5 points): A program executes instructions 1,2,3,4,5,6,7,8 out-of-order. They produce their results in the cycles shown in the table below.

Instruction	Cycle
1	05
2	07
3	16
4	11
5	14
6	15
7	20
8	23

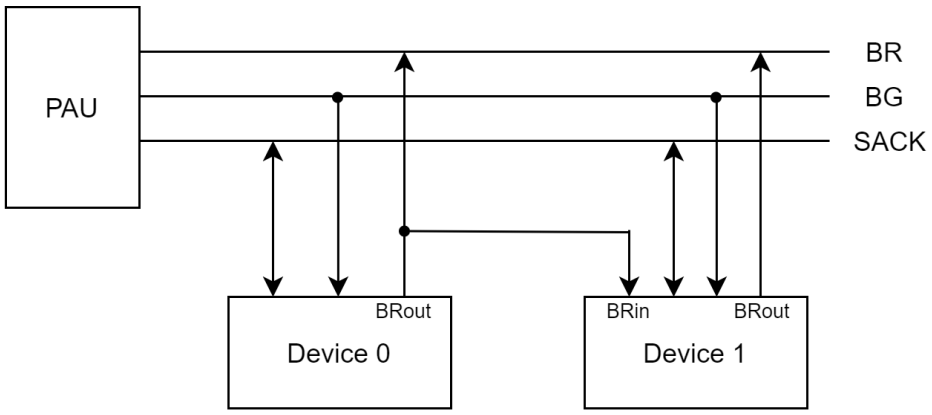
The microarchitecture allows up to 5 instructions to retire each cycle. If we insist on maintaining precise exceptions, when can instruction 5 retire? Explain.

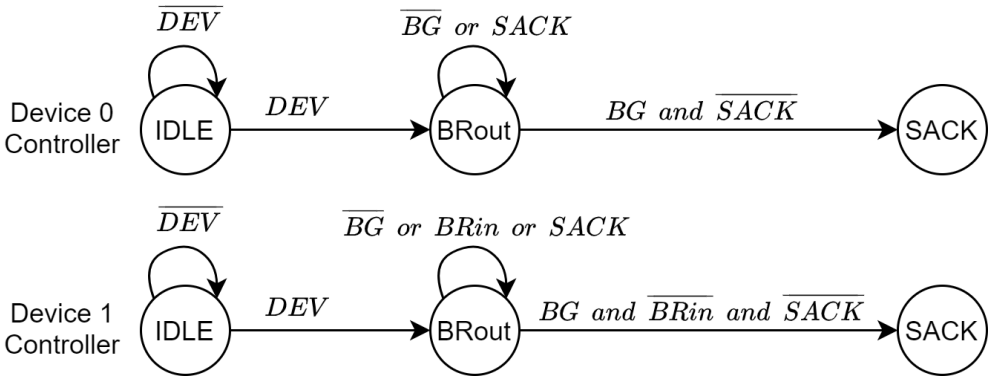
Part b (5 points): Every PTE contains an M bit. What information is contained in the M bit, and why is it useful?

Part c (5 points): In the asynchronous I/O system discussed in class, multiple devices requested the bus at the same priority. The PAU granted the bus in a daisy chain manner. Does that say anything about the relative priorities of the devices that requested the bus all at the identical priority level? Explain.

Part d (5 points): A user program generated a virtual address that was on page x83 of process space. The process page table consists of x81 PTEs. The hardware's job is to determine the frame of physical memory containing virtual page x83. In this case, what does the hardware do? Explain.

Problem 2 (10 points): A student decided to change the asynchronous bus arbitration protocol that we showed in class. Instead of daisy chaining the devices on the same priority, the student tried to solve the problem by letting Device 1 read the BR signal produced by Device 0. Only the relevant states in the state machine are shown.





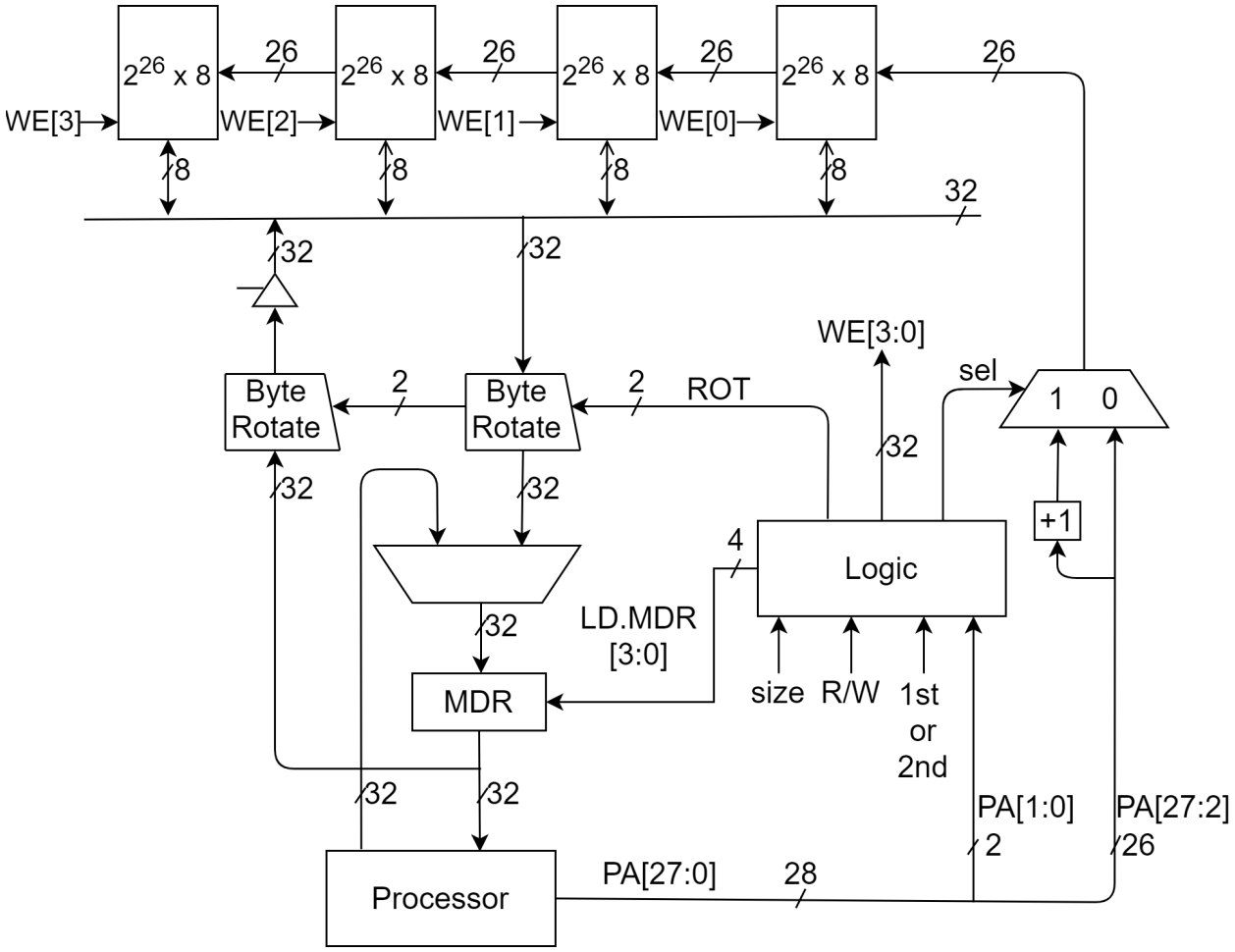
Unfortunately, this solution does not work because of a race condition.

Part a (3 points): What bad event will be caused as the result of this race condition?

Part b (7 points): What events must happen and in which order to cause the race condition? Please be precise but not wordy.

Problem 3 (20 points)

We have the following memory, similar to the one in Problem Set 3, which supports unaligned accesses. The ISA contains `LDByte`, `LDWord`, `STByte` , and `STWord` instructions, where a word is 32 bits.



Both byte rotators in the figure are right rotators.

Assume we have an array, `products_array`, with 2048 elements, where each element is a 5 byte struct of type `Product`, made up of a 4 byte field `PRICE`, and a 1 byte field `ON_SALE`, as shown below:

```
typedef struct Product_Struct {
    int  PRICE;                /* 4 bytes */
    char ON_SALE;              /* 1 byte  */
} Product;
Product products_array[2048];  /* begins at physical address 0x0  */
```

The array `products_array` begins at physical address 0x0.

We wish to read in the entire `products_array` array from memory. For each of the 2048 array elements, we will perform a `LDWord` on the 4 byte field `PRICE`, and a `LDByte` on the 1 byte field `ON_SALE`.

Part a (6 points):

1. Of the 2048 `LDWord` operations, how many will require a 2nd access?

2. Of the 2048 `LDByte` operations, how many will require a right rotate value of 2?

Part b (4 points):

Instead of an array of structs, suppose we stored the same information in two separate arrays `PRICE_ARRAY` and `ON_SALE_ARRAY`, as shown below:

```
int  PRICE_ARRAY[2048];              /* 4 bytes each element, 8192 bytes total*/
char ON_SALE_ARRAY[2048];            /* 1 byte each element,  2048 bytes total*/
```

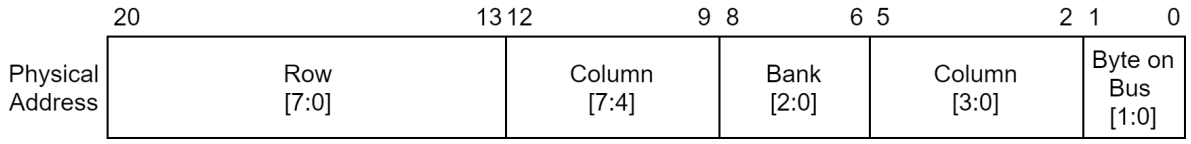
The two arrays are stored in contiguous physical memory, with the array `PRICE_ARRAY` located at physical address 0x0. We read in both arrays, performing a `LDWord` on each of the 2048 elements of `PRICE_ARRAY`, and a `LDByte` on each element of `ON_SALE_ARRAY`.

1. Of the 2048 `LDWord` operations, how many will require a 2nd access?

2. Of the 2048 `LDByte` operations, how many will require a right rotate value of 2?

Part c (10 points):

Suppose the array `PRICE_ARRAY` from part b is stored in DRAM memory which is organized as follows:



Note the 8 bits of column address are split into two parts, with the high 4 bits coming from bits [12:9] of the physical address, and the low 4 bits coming from bits [5:2] of the physical address.

Also note that this physical memory setup is completely independent from that of part a and b.

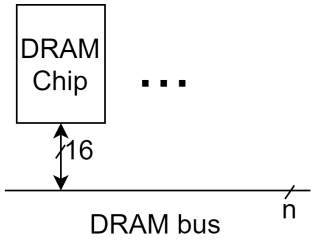
The array `PRICE_ARRAY` begins at physical address 0x0. We access all 2048 elements of the array in sequential order. **Assume all row buffers are initially empty**.

1. What is the address of the last array element that causes a row buffer miss?

2. What is the width of the bus in bytes?

3. What is the size of the row buffer (of a bank across all chips) in bytes?

4. If we used DRAM chips with 16 bit data interfaces, as shown in the figure below:



How many DRAM chips in total are there in the system?

What is the capacity per DRAM chip in bytes?

Problem 4 (25 points): We wish to enhance the LC-3b ISA by adding virtual memory support. Virtual memory will be implemented by a VAX-like scheme as we studied in class. The 16-bit addresses you are familiar with are virtual addresses. Physical memory is 16KB. Page size is 256 bytes.

The memory management system uses the two-level page table scheme. Virtual memory is partitioned into two halves. User space starts at x0000, System space starts at x8000. The TLB contains 8 entries and is fully-associative. The TLB is only used for storing user process PTEs. A PTE is 16 bits. For purposes of this question only, we will assume the PTE has the following form:

V	000000000	PFN

We wish to execute the following two instructions sequentially in a program fragment:

At address x3000: LDW R0, R1, #0 (instruction encoding: x6040)
At address x3002: STW R0, R2, #0 (instruction encoding: x7080)

Before the execution of the two instructions, R1 = x4000, R2 = x5002, and the TLB is initially empty. After the execution of the two instructions, R0 = xA000.

The execution of the two instructions leads to the following physical address accesses in an unspecified order. This means that the order of the list of physical addresses is not necessarily the order in which they are accessed. Note that address x0240 is accessed three times.

Physical addresses: x0240, x0240, x0240, x0360, x0380, x03A0, x0400, x0402, x0502, x0600

Assume SBR points to the starting address of a frame.

Part a (3 points): How many TLB hits do we observe?

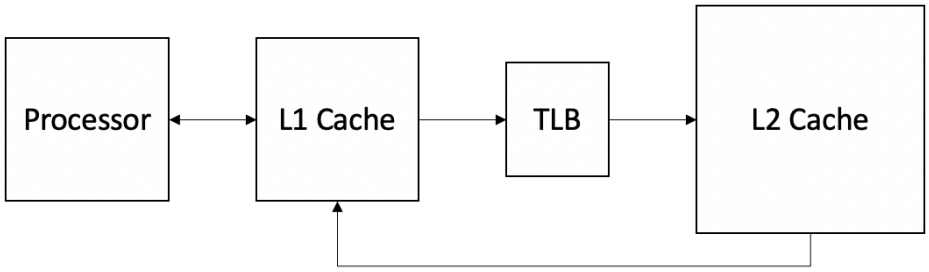
Part b (16 points): Specify the 2-byte word at each physical address after the two instructions are executed.

Address	x0240	x0360	x0380	x03A0	x0400	x0402	x0502	x0600
Content								

Part c (3 points): What is the value of the System Base Register (SBR)?

Part d (3 points): What is the value of the User Process Base Register (PBR)?

Problem 5 (25 points):
Consider the following two-level cache memory system.



We know the following:

- The memory is byte addressable.
- A virtual address is 16 bits, and a physical address is 14 bits.
- The size of a page is 256 bytes.
- The L1 cache has a capacity of 1 KB and is virtually indexed, virtually tagged.
- The L2 cache has a capacity of 4 KB and is physically indexed, physically tagged.
- The L1 and L2 caches have the same cache block size and the same number of sets.
- The L2 is inclusive of L1
- Both caches begin with a cold start.
- The TLB hit rate is 100%.

The processor is executing part of a program that needs to read five values from memory in the sequence shown below. Note that six cells in the L2 cache sequence table are left blank for you to fill in.

L1 Cache Access Sequence

Address	**Hit/Miss**
x3020	Miss
x3040	Miss
x3000	Hit
x3440	Miss
x3050	Miss

L2 Cache Access Sequence

Address	**Hit/Miss**
x0120	
x0940	

Part a (12 points):

1. What is the size of a cache block?

2. What is the associativity of the L1 cache?

3. What is the associativity of the L2 cache?

4. How many bits are required for tag, index, and offset?

Cache	# Tag Bits	# Index Bits	# Offset Bits
L1			
L2			

5. Is it possible to access the TLB in parallel with the L2 cache? Why or why not?

Part b (5 points): Fill out the six empty cells in the L2 Cache Access Sequence table.

Part c (8 points):
Now consider a **different** two-level cache system of a byte-addressable memory. In this system:

- The L1 cache is 1 KB and is 2-way set associative.
- The L2 cache is 2 KB and is 4-way set associative.
- Both caches are physically indexed, physically tagged; physical addresses are still 14 bits.
- The replacement policy is LRU.
- Both caches have 16-byte blocks.
- The L2 is inclusive of L1

We would like to execute the following C-code snippet. You may assume that the caches are empty before the program's execution, and that the values of integers `i` and `x` are always stored in a register. `A`, `B`, and `C` are arrays of 32-bit integers.

```
for (int i = 0; i < 128; i ++) {
    x += A[i]*B[i] + C[i];
}
```

In each iteration of the loop, the following operations occur in order: A[i] is read, then B[i], then C[i], then x and i are updated. Array `A` begins at physical address x0400, `B` begins at physical address x0800, and `C` begins at physical address x0C00.

For the questions below, consider only data accesses (i.e., ignore instruction accesses).

1. What is the L1 cache hit ratio when executing this snippet?

2. What is the L2 cache hit ratio when executing this snippet?

3. What is the L1 cache hit ratio if we execute this snippet again, immediately after executing it the first time, with a warm cache?

4. What is the L2 cache hit ratio if we execute this snippet again, immediately after executing it the first time, with a warm cache?