Department of Electrical and Computer Engineering
The University of Texas at Austin

ECE 460N Spring 2025
Instructor: Yale N. Patt
TAs: Luke Mason, Jenna May, Roy Mor, Rathna Sivakumar, Margaret Lee
Exam 2
April 16th, 2025

Name: _____

Problem 1 (15 points): _____

Problem 2 (10 points): _____

Problem 3 (20 points): _____

Problem 4 (30 points): _____

Problem 5 (20 points): _____

Legibility (5 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.

Note: Please be sure your name is recorded on each sheet of the exam.

Please read the following sentence, and if you agree, sign where requested:
I have not given nor received any unauthorized help on this exam.

Signature: _____

GOOD LUCK!

Name: _____

Problem 1 (15 pts): Answer the following questions in 20 words or fewer.
For each answer, if you leave the box empty, you will receive one point of the five.

Part a (5 pts): Dr. Patt wants to design a 32KB 4-way set associative physically accessed cache to support a computer system containing 4MB of physical memory. He wants the cache line size to be 8 bytes. Furthermore, he wants to be able to access the TLB, Tag Store, and Data Store concurrently. Are there any constraints on the virtual page size in order to make this possible? If so, what is the constraint?

Part b (5 pts): We have discussed two approaches to handling virtual memory: putting the process page table in physical memory and putting the process page table in System virtual memory. Putting the process page table in system virtual memory requires much more complicated handling than putting it in physical memory. Why then would we put it in system virtual memory?

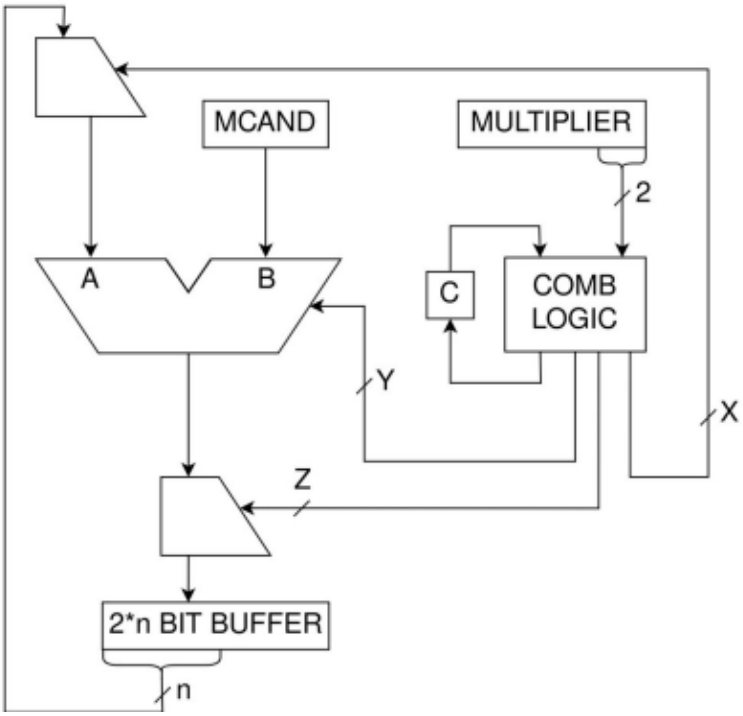
Part c (5 pts): A receiver receives 8 bits of data from a sender. Consider two approaches to dealing with errors:

1. Transmit one parity bit alongside the message, ensuring parity of all 8 bits.
2. Use Hamming codes; i.e. transmit four parity bits, each corresponding to a different combination of data bits.

What is one benefit of applying Hamming code compared to a single parity bit?

Name: _____

Problem 2 (10 pts): The following diagram shows hardware logic for an implementation of Booth's algorithm. If you leave **both** parts blank, you will receive **1 point**.



For an instance of Booth's Algorithm, bits [13:8] of the multiplier are 011101. These are used in iterations 5, 6, and 7.

Part a (3 pts): What is the value of C after iteration 5?

Part b (7 pts): In iteration 7, what are the values of the control signals for X, Y, and Z? Circle the correct options below.

X:	**Y:**	**Z:**
SHF0 SHF1 SHF2	ADD SUB PASSA	SHF0 SHF1 SHF2

Name: _____

Problem 3 (20 pts): You've been asked to optimize the DRAM address mapping of a system with these parameters for the given address stream:

- Memory has an 8-bit address space, and is byte addressable. The bus size is 1 byte.
- It takes 11 cycles to close an open row, 16 cycles to open a row, and 8 cycles to read column data from an open row.
- There are 2 channels, 1 rank/channel, 2 banks/rank, 16 rows and 4 cols/bank.
- Requested data may arrive out of order.
- Requests are sent in order.
- All row buffers begin unopened (empty).
- Each channel can send at most one request and receive one byte of data per cycle.

Note: You will need the information in the table below to complete part a and b, but the empty boxes in the table will not be graded.

Address (Hex)	Address (Binary)	Dispatch Cycle
0x00	0 0 0 0 0 0 0 0	0
0x40	0 1 0 0 0 0 0 0	0
0x03	0 0 0 0 0 0 1 1	
0x02	0 0 0 0 0 0 1 0	
0x46	0 1 0 0 0 1 1 0	
0x05	0 0 0 0 0 1 0 1	
0x04	0 0 0 0 0 1 0 0	

Part a (14 pts): What's the best address mapping to **optimize cycle time** for the access sequence above? Label bits Ch for channel, Bk for bank, R for row, and C for column.

7	6	5	4	3	2	1	0

Part b (6 pts): How many row buffer hits are there?

#Hits:

Name: _____

Problem 4 (30 pts): For this problem, consider the following PTE format:

PFN	0 Padding*	V	P

V is 1 if the PTE is valid, 0 otherwise. P is 0 if the page requires privilege to access, 1 otherwise.
***The PTEs are 0 padded to the next byte. There may be no 0 padding.**

Part a (9 pts): System 1 uses a one-level translation scheme where virtual addresses are 32 bits and physical addresses are 20 bits, byte addressable. There is one page table containing both system and user pages, similar to lab 5, so the VA has no region bit. The page table must be resident in memory with additional space for at least one user page and system page. What is the smallest page size possible, and what is the corresponding PTE size in this case?

Minimum page size:	Show work:
PTE size:	

Part b (6 pts): System 2 has the same address space and PTE format as System 1. However, it uses a VAX-like two-level translation scheme. Like before, the entire System page table must be resident in memory with additional space for at least one user page and one system page. User space begins at 0x00000000 and system space begins at x80000000. What is the smallest page size possible, and what is the corresponding PTE size?

Minimum page size:	Show work:
PTE size:	

PROBLEM CONTINUES ON NEXT PAGE

Name: _____

The following information and tables correspond to part c, d, and e:
You wish to execute the instruction STW R2, R0, #0 in **user mode** on the processor.

- You do not know if the instruction is executed using system 1 or system 2.
- The processor is using the LC-3b ISA and is **little-endian.**
- User mode does not support self-modifying code.

The tables below contain some **physical** memory locations after the instruction is executed, including all locations (both data and PTE) accessed during fetch and execution of the instruction.

Physical Address	Content
0x00000	0x02
0x00001	0x84
0x10072	0x02
0x10073	0x90
0x70102	0x01
0x70103	0xC0
0x83060	0x03
0x83061	0xC0

Physical Address	Content
0x84006	0x03
0x84007	0xD0
0x90080	0x03
0x90081	0xC0
0xC0194	0x00
0xC0195	0x74
0xD0174	0x03
0xD0175	0x11

Part c (5 pts): Is the program run on System 1 or System 2? How do you know?

Circle one: **System 1 System 2**	Explain (30 character limit):

Part d (5 pts): What is the 16-bit word that was stored?

Part e (5 pts): What is the value of R0, the base register?
Note: registers are large enough to contain a 32-bit VA.

Name: _____

Problem 5: (20 pts): Consider a processor with byte-addressable memory and a single-level data cache. The processor executes the following C code:

```
void CopyArray(int[] A, int[] B) {
   for(int i= 0; i< 6; i++) {
        int temp = B[i];
       A[i] = temp;
   }
}
```

The cache contains four sets. You don't know the block size, nor the associativity of the cache. To achieve better distribution of set accesses, this cache uses an interesting indexing scheme. The 8 leftmost bits of the address are used for the tag. However, to generate the 2-bit set index, bits W and X are XORed with bits Y and Z. So, **Set# = (W⊕Y)'(X⊕Z)**. See the format below:

						W	X	Y	Z	
T	T	T	T	T	T	T	T			Block Offset

You also know the following:

- Only accesses to A and B will access the cache.
- Cache blocks are **at least** large enough to contain an int data type, i.e. 32 bits.
- Both arrays start at the beginning of a cache block.
- There were 4 cache evictions during code execution.
- The cache is initially empty.
- The cache uses LRU replacement.

You are given the tags stored in way 0 of each set after program execution. The tags **0x3A and 0x75** shown below correspond to the cache blocks containing the fifth element of A and B, respectively (i.e. when i = 4). **Answer the questions on the next page.**

Set	**V**	**Tag stored in way 0**	**Tag Bits**
0	1	0x39	0 0 1 1 1 0 0 1
1	1	0x39	0 0 1 1 1 0 0 1
2	1	0x3A	0 0 1 1 1 0 1 0
3	1	0x75	0 1 1 1 0 1 0 1

PROBLEM CONTINUES ON NEXT PAGE

Name: _____

Part a (5 pts):
Is this cache function valid? In other words, would using this cache avoid correctness problems for all programs? Explain your answer briefly. If your answer is no, do not solve the rest of this problem.

Part b (6 pts):
What is the cache block size? *Hint: figure out the Y and Z bits of each cache block.*

Part c (4 pts):
What is the starting address of A and B?

A:	B:

Part d (5 pts):
Recall that there were 4 cache evictions during the execution of the program. What is the total size of the cache?