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

ECE 460N/382N.1 Fall 2024
Instructor: Yale N. Patt
TAs: Anna Guo, Nadia Houston, Logan Liberty, Luke Mason, Abhay Mittal, Asher Nederveld, Edgar Turcotte
Exam 2 KEY
November 13, 2024

Name: ____

Problem 1 (20 points): _______

Problem 2 (25 points): _______

Problem 3 (25 points): _______

Problem 4 (30 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: ______________________________

Question 1 (20 points): Answer the following questions.
Note: For each of the four answers, **if you leave the box empty, you will receive one point**.

Part a (5 points): We have discussed several access methods to load and store values in storage. One was Content Addressable Memory (CAM). Give an example of storage that uses CAM for access and how that works, in 15 words or fewer.

One example: TLB. VPN identifies the location, and is part of the contents.
Another example is Tag Store. The tag bits specify the location (way), and are part of the way's tag store entry.

Part b (5 points): Process A is running when it encounters an exception. Control goes to the operating system which executes the relevant exception service routine. At what priority does the service routine execute:

1. The same priority that Process A has been running at.
2. A priority higher than what Process A is running at.
3. A priority lower than what Process A is running at.

Explain in 15 words or fewer.

Same priority. Urgency is not affected by exceptions occurring during execution.

Part c (5 points): G-share modified my index to the pattern history tables by XORing each bit of the n-bit history register with n-bits from somewhere else. What supplied the additional n-bits, and why did it improve prediction accuracy? …in 15 words, of course!

PC bits. Different indexes for the same history but different behavior indexes to different counters.

PROBLEM CONTINUES ON THE NEXT PAGE

Name: ________________________________

Part d (5 points): On a context switch, some state associated with the process losing control of the computer is saved. Which of the following are never saved: PC, condition codes, PBR, PLR, SBR, SLR. Explain in 15 words or fewer.

SBR and SLR. They are part of the OS and are identical for all processes, they always remain and are not saved. I should have given them 25 words and I will take that into account in grading problem 1. - Dr. Patt

Name: ________________________________

Question 2 (25 points):

A computer has physical memory with the following characteristics:

- The memory is byte-addressable
- There is one channel, one rank, two banks, 64 rows, and 16 columns
- There is a 32-bit memory data bus
- Row buffer hits take 10 cycles to return the desired data
- Row buffer misses take 50 cycles to return the desired data
- In the case of a bank conflict, the next access starts 1 cycle after the data is returned from the previous access to that bank
- In the case of no bank conflict, the next access starts 1 cycle after the previous access was started
- All row buffers start empty and there are no outstanding memory operations from before cycle 0
- Memory requests are sent in order
- Data corresponding to a later memory access can arrive before the data corresponding to an earlier memory access

Part a (4 points):

Given the above specification, how many bits of the physical address are required to specify each of the following?

Channel	0 bits
Rank	0 bits
Bank	1 bit
Row	6 bits
Column	4 bits
Byte-on-Bus	2 bits

PROBLEM CONTINUES ON THE NEXT PAGE

Name: ______________________________

Part b (21 points):
The table below holds information about 6 memory accesses that are handled in order. Each row contains the physical address of the access, the cycles during which the access started and ended, and whether the access had a row buffer hit or bank conflict. Some of the entries have been filled in for you.

Your job: fill in the missing entries, and the physical address format.

Physical Address	Cycle Start	Cycle End	Row Buffer Hit	Bank Conflict
0000_0010_1000_0100	0	50	N	N
0000_1100_0100_1001	1	51	N	N
0000_1100_0101_1000	52	62	Y	Y
0001_1100_0110_0110	63	113	N	Y
0000_0010_1000_0111	64	74	Y	N
0001_0101_0100_0110	114	164	N	Y

Note: Each bit field in the physical address is contiguous (i.e. all the row bits are next to each other, and all the column bits are next to each other). The byte-on-bus bits are the least significant bits.

Use Ch for channel, Rk for rank, Bk for bank, R for row, C for column, and BoB for byte-on-bus. You may not need to use all the provided bits or symbols. If so, place any unused bits as X at the highest positions of the address.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	R	R	R	R	R	R	Bk	C	C	C	C	BoB	BoB

Name: ________________________________

Virtual Memory
Problem 3 (25 points):

This question requires you to solve several problems dealing with virtual memory.
All of them are on the next page. Shown below is all the information you will need to solve them

- The machine is the LC-3b that has VAX-like 2-level virtual memory
  - Byte-addressable, virtual addresses range from x0000 to xFFFF, little-endian
- System space ranges from x0000 to x7FFF; user space ranges from x8000 to xFFFF
- The computer is augmented to support unaligned accesses
- The machine has a 4-entry fully associative TLB with LRU replacement policy
- The TLB is initially empty and is not necessarily filled top-to-bottom.
- The virtual address is of the following format with a page size you must figure out.

Region	VPN	Page Offset

- The PTE is 2 bytes, and is zero-padded at the lower end.

V	000	PFN	0's

The machine is running the code shown below:

```
        .ORIG x8000
        LEA R0, DATA
        LDW R1, R0, #1
        LDW R0, R0, #0
        STW R1, R0, #0
        HALT
        .END
```

```
        .ORIG x8080
DATA    .FILL xCFFF
        .FILL xEE00
        .END
```

PROBLEM CONTINUES ON THE NEXT PAGE

Name: ________________________________

Below is the state of the TLB after the program at x8000 halts. Note: the TLB does not map system pages.

V	Region	VPN	PTE
1	1	**x00**	**x8600**
1	1	**x50**	x8580
1	1	**x4F**	x8480
0	1	x26	x8200

Some memory locations after the program halts are shown below:

VA	PA	Data
———	x07B0	x8040
x5800	x0100	x8600
x8000	x1800	xE03F
x8080	**x1880**	xCFFF
xCFFE	x12FE	x0066
xD000	x1600	x88EE

Part a (5 points): What is the page size?

Page Size **2^8 or 256B**

Part b (8 points): During the execution of the first instruction, 3 memory accesses were made. One at physical address 0x07B0, one at physical address 0x0100, and another one at physical address 0x1800. Given this, what are the PBR and SBR?

PBR **x5800**

SBR **x0700**

Part c (12 points): Fill out the missing entries in the program, the TLB, and memory locations.

Name: ________________________________

Question 4 (30 points):
An Aggie has been asked to write a program segment to multiply two 3 by 3 matrices A and B. However, instead of producing a 3 by 3 matrix where each element C[i, j] is formed by computing the dot product of row i of matrix A and column j of matrix B, he incorrectly produces a 3-element vector by computing the dot product of row i of matrix A and row i of matrix B.

Matrix A, Matrix B, and the result of his incorrect process are shown below.

Matrix A				Matrix B				Array results
a	b	c	●	j	k	l	=	aj+bk+cl
d	e	f	●	m	n	o	=	dm+en+fo
g	h	i	●	p	q	r	=	gp+hq+ir

The Aggie stores Matrices A and B in memory in column-major order and his result vector in three consecutive memory locations. The base address of matrix A is xD540, the base address of matrix B is x3D80, and the base address of his result vector is xED10. All other variables are stored in registers. All three arrays begin at the start of a cache line. His program segment is shown below:

```
int A[9] = generate_array(); // Generates an array for A
int B[9] = generate_array(); // Generates an array for B
int results[3] = {0, 0, 0};
clear_cache(); // Function that guarantees the cache is empty

for (int i = 0; i < 3; i++) {
     int temp = 0;
     for (int j = 0; j < 3; j++) {
          int A_val = A[(j * 3) + i]; // Memory Access
          int B_val = B[(j * 3) + i]; // Memory Access
          temp += A_val * B_val;
     }
     results[i] = temp; // Memory Access
 }
```

PROBLEM CONTINUES ON THE NEXT PAGE

Name: ________________________________

He executes the program segment on a computer having the following characteristics:

- Memory is byte-addressable.
- The integer data type is 32 bits.
- The computer has a 1KB (1024 bytes) data cache with perfect LRU replacement.

To get the 3-element vector he has to execute the loop body of his code three times. The following table shows the result (HIT or MISS) of each access to the cache for each iteration of the loop body.

Iteration 0 (i=0)	Iteration 1 (i=1)	Iteration 2 (i=2)
MISS	HIT	HIT
MISS	HIT	HIT
HIT	MISS	MISS
HIT	MISS	MISS
MISS	HIT	MISS
MISS	HIT	MISS
MISS	MISS	MISS

Hint: The index and tag bits may not be contiguous (the line offset bits are still contiguous). The next page contains a table for scratch work.

Part a (2 points): Given the information above, what is the size of a cache line?

16 Bytes

Part b (4 points): What is the associativity of the cache?

2 Way

Part c (4 points): How many sets does the cache have?

32

Part d (20 points): Indicate which bits of the address are used for tag, index, and offset

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Address:	T	T	T	T	T	I	I	I	T	T	I	I	O	O	O	O

PROBLEM CONTINUES ON THE NEXT PAGE

Name: ________________________________

This page is only meant for scratch work and **will not be graded. You should not have to fill out the entire table to solve this problem**

Iteration 0 (i=0)	**Memory Address (in binary)**			
MISS MISS HIT HIT MISS MISS MISS	1 1 0 1 0 0 1 1 1 1 1 0	0 1 0 1 1 1 0 1 1 1 0 1	0 1 0 0 1 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0 0 0 0 0
Iteration 1 (i=1)	**Memory Address (in binary)**			
HIT HIT MISS MISS HIT HIT MISS				
Iteration 2 (i=2)	**Memory Address (in binary)**			
HIT HIT MISS MISS MISS MISS MISS				