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

ECE 460N Spring 2023
Instructor: Yale N. Patt
TAs: Michael Chen, Ali Mansoorshahi
Exam 2
April 12, 2023

Name: _____

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.

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 below, if you leave the box empty, you will receive one point.

Part a (5 points): An integer, expressed as a 16 bit 2's complement number is 0000000100110000. What is its 16 bit representation in BCD?

0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0

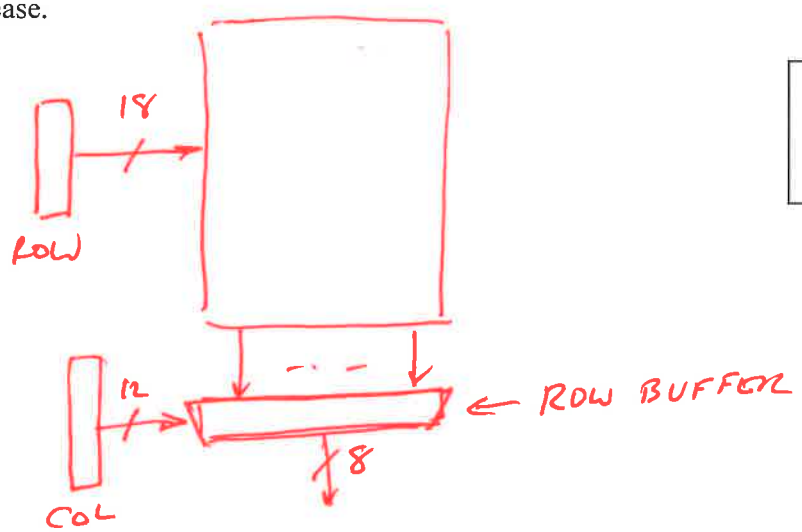
Part b (5 points): Synchronous logic assumes a clock, and the result is latched at the end of the clock cycle, even if the result was available long before that. Asynchronous logic implies no clock, which means the machine does not have to wait and the result can be used as soon as it is generated. Therefore, which provides higher performance? Explain in fewer than 15 words.

SYNCHRONOUS BECAUSE IT DOES NOT REQUIRE HANDSHAKING

Part c (5 points): Most ISAs have a "reference bit." In 15 words or fewer, what is the purpose of the reference bit?

TO IDENTIFY PAGES THAT HAVE NOT BEEN ACCESS AS TARGETS FOR EVICTION

Part d (5 points): The physical address space is 1 GB, with 18 bits of row address and 12 bits of column address. Assuming one bank, the chip contains one row buffer. How big is that row buffer? Just the size, please.

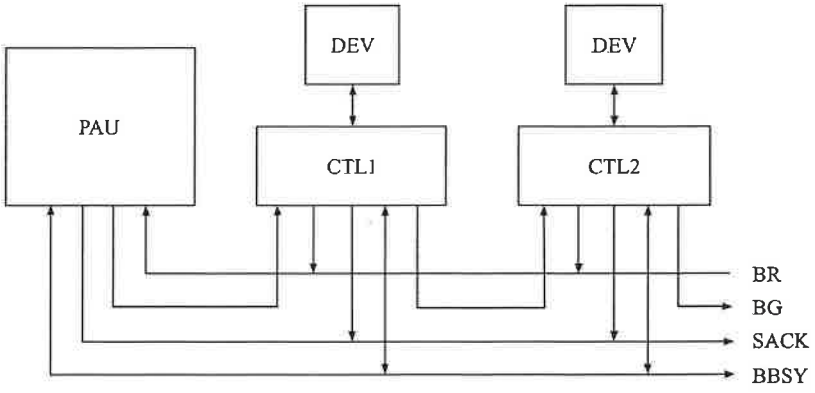


2^{12} B

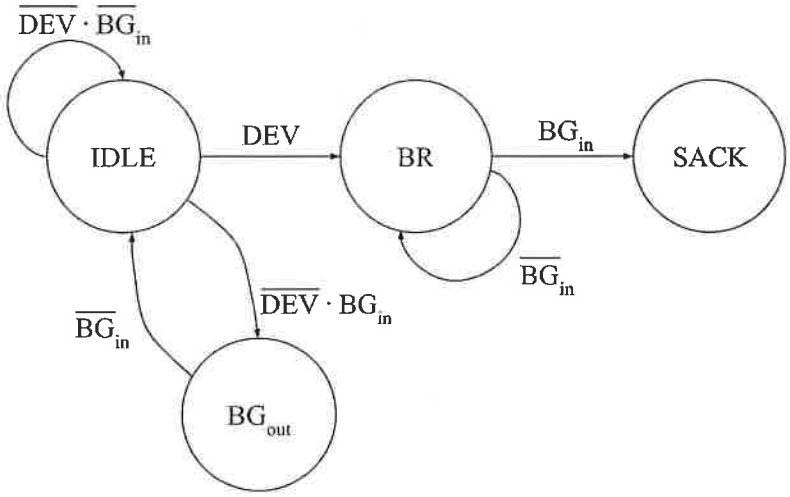
Name: _____

Question 2 (10 Points): We want to specify the behavior of the Priority Arbitration Unit (PAU) in an asynchronous IO system.

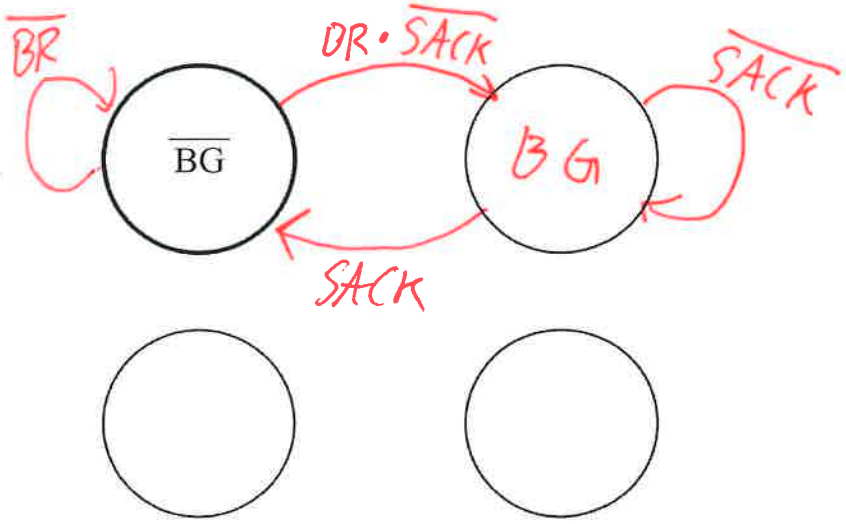
Note: If you leave the question blank, you will receive one point.



Below is the state machine for each device's controller.



Draw the state machine for the PAU. Start in the bold state. Use as many states as you need.



Name: _____

Question 3 (20 Points): Access to the DRAMs that make up physical memory is controlled by a memory controller. The memory controller has the following properties:

- The Memory Controller can send one request per cycle
- The Memory Controller can receive data in the same cycle as it sends a request
- The Memory Controller sends requests to the DRAM in the order specified by its scheduling algorithm

The DRAM Memory has the following characteristics:

- There is one channel, one rank, two banks, m rows and n columns
- Page_mode_access (row buffer hit) returns the data in the next cycle after the memory controller sent the request.
- Non-page_mode_access (row buffer miss) takes a fixed number of cycles
- All rows start closed

For example, if non-page_mode_access is 14 cycles, and the first access on the right is non-page_mode_access, a request sent in cycle 0 receives the data in cycle 14. If the second access is page_mode_access to a different bank, the request can be sent in cycle 1, and the data received in cycle 2.

Send Cycle	Receive Cycle
0	14
1	2

Part a (8 points). The table below shows the start and end cycle for 7 memory accesses that are serviced in the order received (in this case, A,B,C,D...). This scheduling mechanism is referred to as First Come First Served (FCFS) scheduling.

Your job: Fill in the missing entries.

Access	Send Cycle	Receive Cycle	Bank	Row Address	Page Mode
A	0	20	0	0	miss
B	20	21	0	0	hit
C	21	41	1	1	miss
D	22	23	0	0	hit
E	41	61	1	0	miss
F	42	62	0	1	miss
G	62	82	0	0	miss

PROBLEM CONTINUES ON NEXT PAGE

Name: _____

Part b (3 point). How many bank conflicts occur using FCFS?

3

An alternative scheduling algorithm is Open Row Priority (ORP). ORP sends page_mode_accesses first. If there is a tie between two accesses, then they are scheduled in FCFS order.

Part c (6 points). What is the order of accesses under the Open Row Priority scheduling algorithm? Does this new order improve, degrade, or not change performance? (Empty tables are provided for scratch work on the next page)

A B D G C E F

improves performance

Part d (3 point). How many bank conflicts occur using ORP?

2

Name: _____

Access	Send Cycle	Receive Cycle	Bank	Row Address	Page Mode

Access	Send Cycle	Receive Cycle	Bank	Row Address	Page Mode

Name: _____

Question 4 (25 Points): Question 4 requires you to solve several problems dealing with virtual memory. All of them are on the next page. Shown below is all the structure you will need to solve those problems.

- The machine is the LC3-b that has VAX-like 2 level virtual memory
- 16KB of Physical Memory
- We partition memory into 2 regions: system space starts at x0000, user space starts at x8000
- SBR points to the start of a frame
- The machine has a 2 entry TLB that starts initially empty.

The virtual address is of the following format with a page size that you must figure out.

Region	VPN	Offset

The PTE is 2 Bytes and has the following format shown below.

0	0	PFN	0's	V	Ref	Mod

We execute instruction LDW R1, R0, #0. Afterward, we are given a snapshot of the TLB. The TLB does not map system pages.

Page No.		PTE
Region	VPN	
1	x34	x2B06
----	----	----------

We are also given a portion of physical memory. The contents of physical address x1468, x2368, x3168, x0852, x084A, and x085A are PTEs. Some are used during translation.

Virtual Address	Physical Address	Physical Address (bits)	Contents	Contents (bits)
0x0268	x1468	0 1 0 1 0 0 0 1 1 0 1 0 0 0	x2B06	0 0 1 0 1 0 1 1 0 0 0 0 0 1 1 0
0x0168	x2368	1 0 0 0 1 1 0 1 1 0 1 0 0 0	x2B06	0 0 1 0 1 0 1 1 0 0 0 0 0 1 1 0
0x0368	x3168	1 1 0 0 0 1 0 1 1 0 1 0 0 0	x2B06	0 0 1 0 1 0 1 1 0 0 0 0 0 1 1 0
---------	x0852	0 0 1 0 0 0 0 1 0 0 0 0 1 0	x1447	0 0 0 1 0 1 0 0 0 1 0 0 0 1 1 1
---------	x084A	0 0 1 0 0 0 0 1 0 0 1 0 1 0	x2207	0 0 1 0 0 0 1 0 0 0 0 0 0 1 1 1
---------	x085A	0 0 1 0 0 0 0 1 0 1 1 0 1 0	x3003	0 0 1 1 0 0 0 0 0 0 0 0 0 1 1
x8D3E	x2B3E	_____ 0 0 1 1 1 1 1 0	x5007	0 1 0 1 0 0 0 0 0 0 0 0 0 1 1 1
x8D30	x2B30	_____ 0 0 1 1 0 0 0 0	x6200	0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0

PROBLEM CONTINUES ON NEXT PAGE

Name: _____

Part a (3 points): LDW accesses memory twice (FETCH and LOAD). Why is the TLB only populated with 1 entry? Use 15 words or fewer.

Same page.

Part b (3 points). What is the physical address of the PTE in the process page table used in the access?

x1468

Part c (6 points). What is the page size?

2^6

Part d (5 points). Fill in the bold boxes on the previous page.

Part e (3 points). What is the SBR?

x840

Part f (3 points). What is the PBR?

x200

Part g (2 points). What are the contents on R0 and R1 after the instruction executes?

R0 x283E

R1 x5007

Name: _____

Question 5 (25 Points): An Aggie lost the data sheet with the specifications for a cache. It's your job to recover this information. You are given a subset of the cache specifications and the results of some code execution.

- The cache is Physically Indexed Physically Tagged
- The TLB, L1 tag store, and L1 data store are accessed at the same time
- L1 and L2 use LRU replacement
- L1 and L2 have the same number of sets
- The caches are inclusive
- The caches are designed to minimize associativity
- The L1, L2, and memory are accessed sequentially (i.e. L2 is accessed on an L1 miss. Memory is accessed on an L2 miss)
- Block size = 8B
- Page size = 256B
- 16KB of physical memory

Part a (3 points) Show the address layout. Use "I" for index, "T" for tag, and "O" for offset into the cache line.

The 4 code snippets are executed with the results shown in bold. In all the snippets below:

- `A` is an array of `int` that starts at 0x3000
- `sum` and `i` are kept in registers
- `int` is 32-bits

```
for (int i = 0; i < 3; i++) {
    sum += A[128*i] + A[(128*i) + 1]
}
for (int i = 0; i < 3; i++) {
    sum += A[128*i] + A[(128*i) + 1]
}
```

3 accesses to DRAM, 9 hits in L1, 0 hits times in L2

```
for (int i = 0; i < 6; i++) {
    sum += A[128*i] + A[(128*i) + 1]
}
sum += A[128] + A[128 + 1]
```

6 accesses to DRAM, 7 hits in L1, 1 hit time in L2

PROBLEM CONTINUES ON NEXT PAGE

Name: _____

```
for (int i = 0; i < 6; i++) {
    sum += A[128*i] + A[(128*i) + 1]
}
for (int i = 0; i < 6; i++) {
    sum += A[128*i] + A[(128*i) + 1]
}
```

6 accesses to DRAM, 12 hits in L1, 6 hits in L2 6 times

```
for (int i = 0; i < 7; i++) {
    sum += A[128*i] + A[(128*i) + 1]
}
for (int i = 0; i < 7; i++) {
    sum += A[128*i] + A[(128*i) + 1]
}
```

14 accesses to DRAM, 14 hits in L1, 0 hits in L2

Part b (8 points). What is the associativity of the L1 and L2 cache?

L1 [3] L2 [6]

Part c (4 points). What is the size of the L1 and L2 cache?

L1 [$2^3 \cdot 2^5 \cdot 3$] L2 [$2^3 \cdot 2^5 \cdot 6$]

Part d (10 points). Can the L2 maintain the same total size but have double the number of sets? Would we see an increase in performance on the code snippets above? Explain.

Yes it can double sets by ½ associativity it will decrease performance.

Performance will decrease since all accesses still goto the same set.