

1. (10 pts) **We're going to play Final (exam) Jeopardy!** Associate the following answers with the appropriate question. (You are given the "answers": Pick the "question" that goes best with each "answer".) The first one has been done for you.

"Answers:"

- z. Anywhere but here. **v**
- 1a. A type of memory that uses capacitors to store data.
- 1b. A technique in which the occurrence of one event on a bus follows and depends only on the occurrence of a previous event.
- 1c. A structure that holds recent mappings of virtual to physical addresses.
- 1d. To guarantee that some part of the Page table will remain resident in Memory.
- 1e. Memory that goes away when the power is turned off.
- 1f. What happens when a page is referenced but not in memory.
- 1g. A technique used in CD-ROM Drives to increase storage density.
- 1h. To maximize the efficient use of an expensive system.
- 1i. To make Memory appear to cost as much as the cheapest element and perform as well as the fastest element.
- 1j. An unscheduled subroutine call.

"Questions:"

- a) What is synchronous timing?
- b) What is asynchronous timing?
- c) What is a circuit that exhibits purely sequential behavior?
- d) What is the Data Bus?
- e) What is a Parity error?
- f) What is a Register?
- g) What is a Cache?
- h) What is DMA?
- i) What is an Address Translation Lookaside Buffer (or TLB)?
- j) What is an interrupt?
- k) What is the goal of the Memory Hierarchy?
- l) What is a Page?
- m) What's the point to using a Base Page Table?
- n) What is Volatile memory?
- o) What is a Page Fault?
- p) What is Non-Volatile memory?
- q) What is polling?
- r) What is Constant Linear Velocity?
- s) What is Constant Angular Velocity?
- t) What is the goal of multiprogramming?
- u) What is Dynamic RAM?
- v) Where would I rather be right now than where I am?

2. (6) In the ALU you designed in the homework, how did you differentiate between an operation being an "add" and an operation being a "subtract"? In other words, what bit/bits were set/cleared in order to indicate that the values were to be added instead of subtracted? Why did this work so well?
3. (6) A computer has a cache, main memory, and a disk. If a reference to the cache is a hit, it takes 10 ns to retrieve the data. If a reference misses in the cache, it takes 70 ns to fetch the item from memory and put it in the cache, at which point the request is reissued to the cache. If the required item is not in main memory, it takes 10 ms to fetch the word from the disk, followed by 75 ns to copy the word to the cache, and then the reference is reissued to the cache. The cache hit ratio is .90 and the main memory hit ratio is .70. Write down the equation you would use to calculate the average time in nanoseconds to access a data item on this system.
4. (5) Explain the principle of Locality and why it is important.

- 3 -

8. (6) Given a logical 23-bit address and a 512K-byte physical memory for a byte-addressable machine,

How big is the physical address space?

How big is the virtual address space?

Assuming 64K-byte pages, how many page frames are there? How many pages?

Assuming 2K-byte pages, how many page frames are there? How many pages?

9. (12 pts) Here is a 12-bit Error Correction code format (same one used in class):

$d_8 \ d_7 \ d_6 \ d_5 \ C_4 \ d_4 \ d_3 \ d_2 \ C_3 \ d_1 \ C_2 \ C_1$

- a. Given the *data* bit pattern

0 0 1 1 0 1 1 0

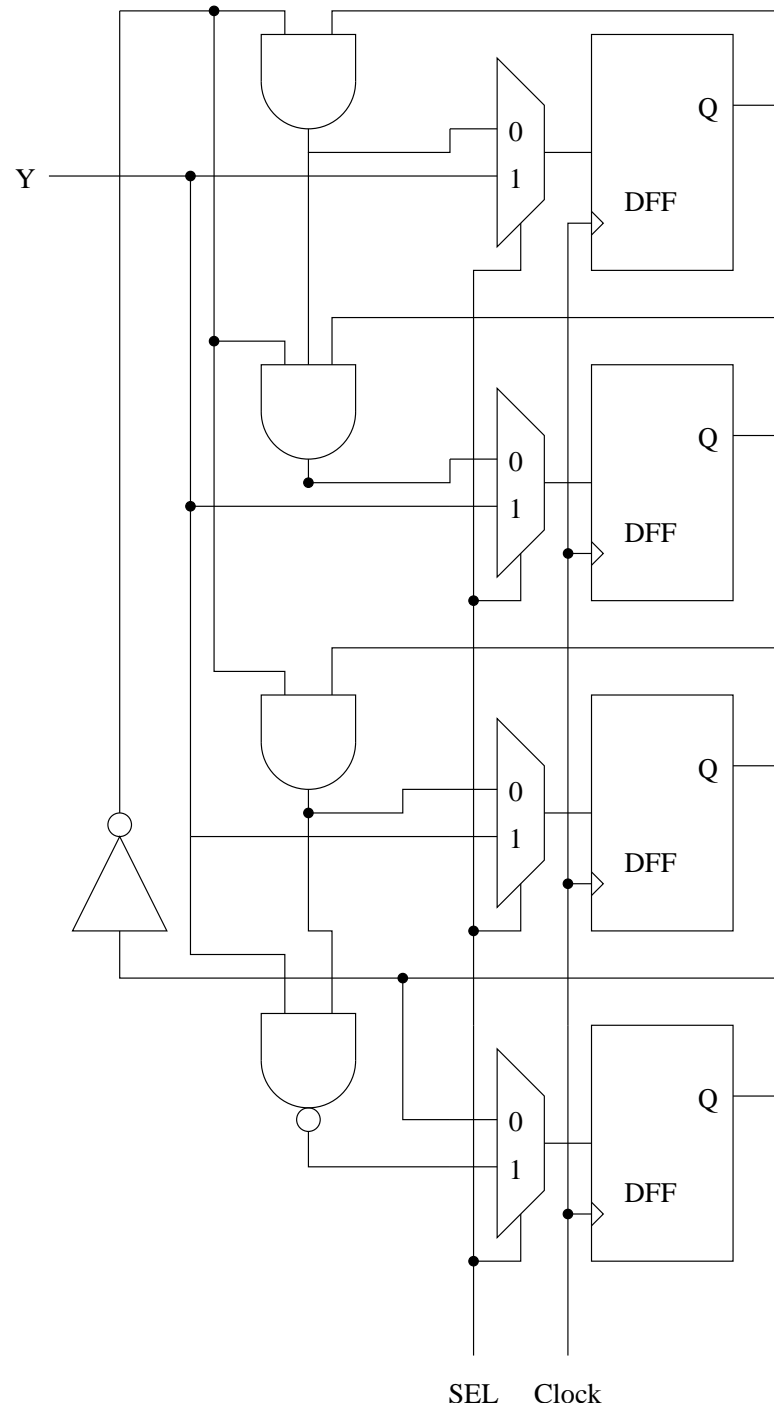
in a machine using the above ECC code, what bit pattern gets sent to memory? (No credit will be given without work being shown.)

- b. In this same machine, the following bit pattern is retrieved from memory:

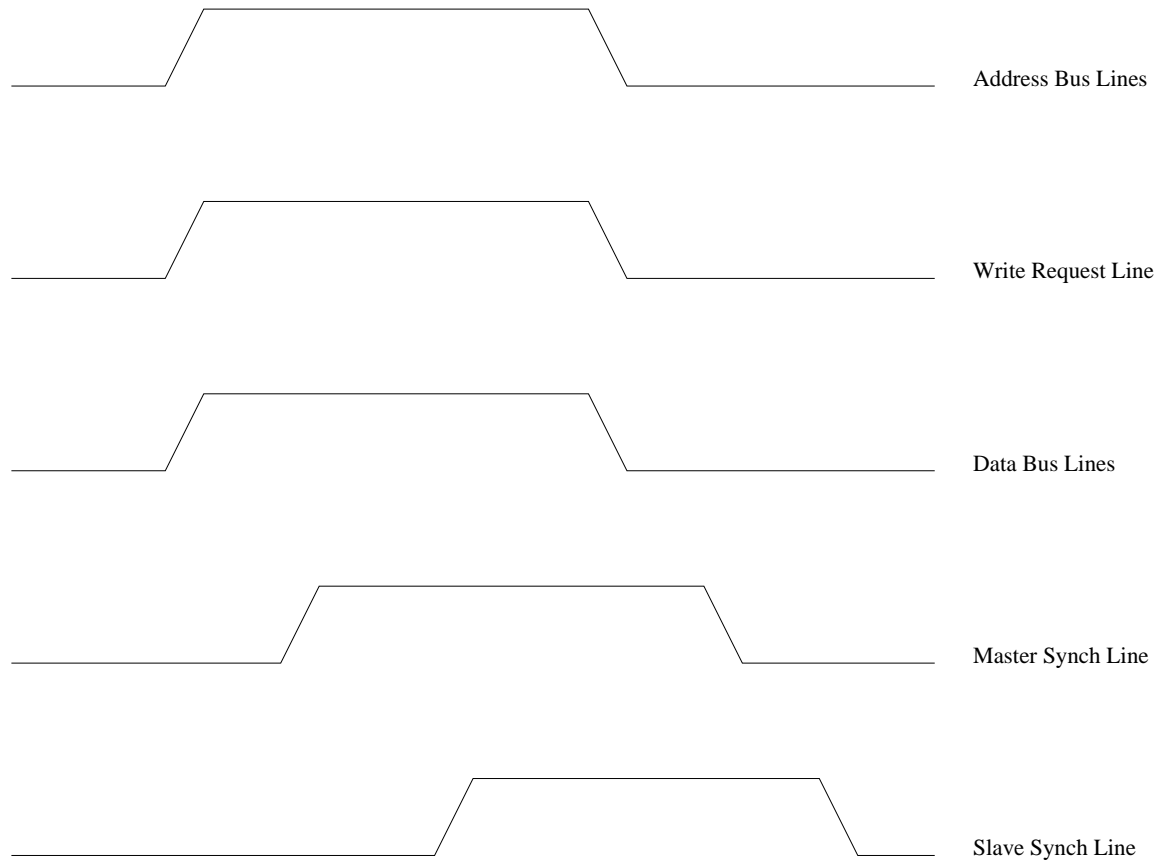
0 0 1 0 1 0 0 0 1 0 0 1

Assuming the above Error Correction code format, identify and correct any errors that may have occurred during transmission or storage. (No credit will be given without work being shown.)

10. (10 pts) What is the maximum clock frequency possible for the following circuit? (In other words, what is the maximum clock frequency that will still guarantee correct behavior?) Use the following delay values, and assume all input signals become valid at time 0:
- AND: 3ns NAND: 4ns NOT: 2ns MUX: 8ns Tprop: 9ns Tsetup: 2ns Thold: 1ns



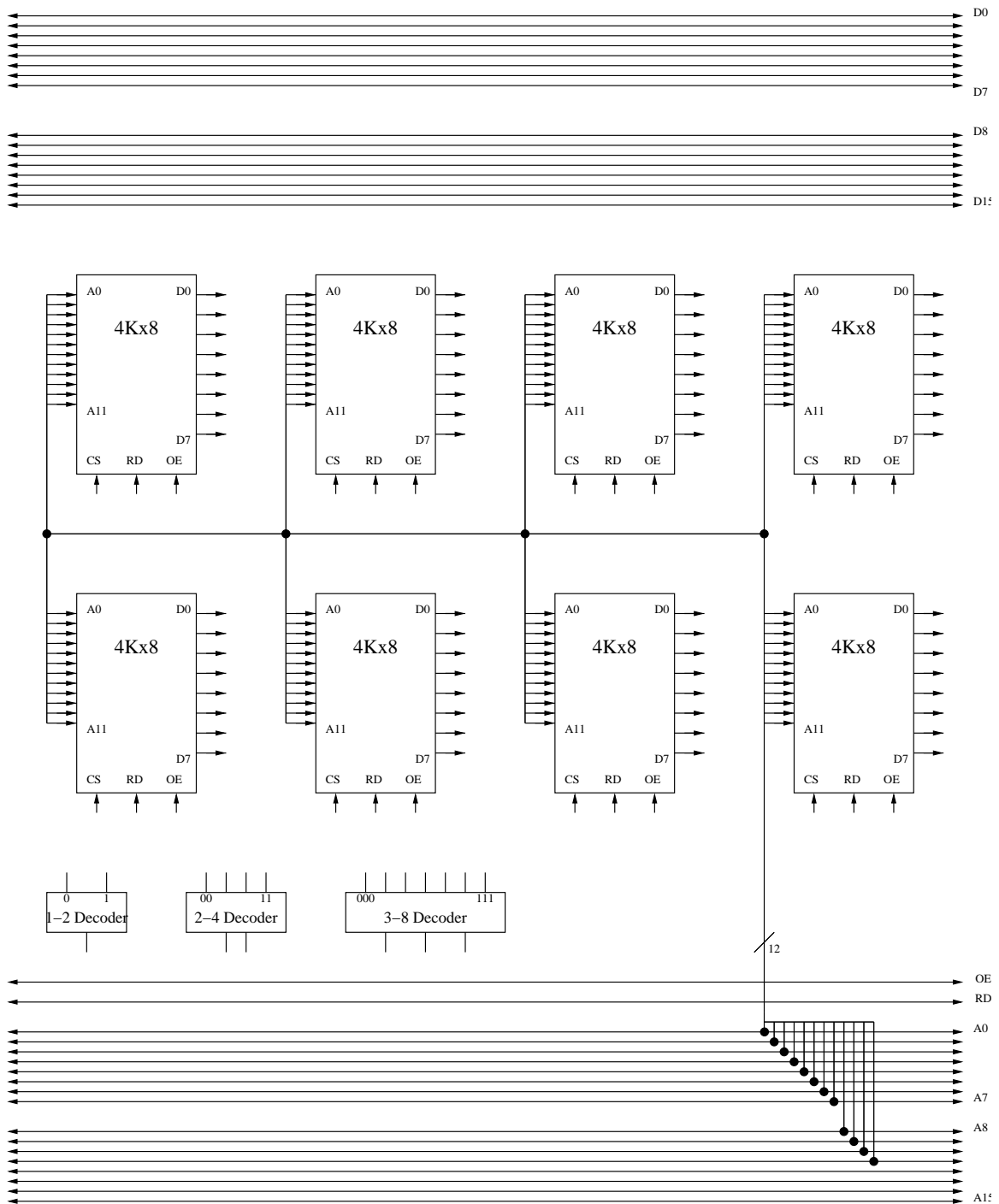
11. (5 pts) Add arrows to the following diagram to indicate which signal transitions **cause** other transitions to occur.



Now explain in words (briefly) what the arrows you have added are doing. (What is the sequence of events in words?)

12. (10 pts) Add the connections to the following diagram necessary to create a 24Kx8 memory. (This might be done in a machine with memory-mapped I/O, for example.) Not all of the hardware shown is required to perform this task.

CS - Chip Select
 OE - Output Enable
 RD - Read (Read/Write, technically)



13. (15) Moon Nanosystems features the "Crater", a byte-addressable computer with a 32-bit word size and 256 bytes of memory. In this machine accessing main memory takes 5 clock cycles (in addition to the time necessary to do a cache lookup), and the bus between main memory and the processor is 8-bits wide. In order to improve performance, they are considering adding a 32-byte physically addressed Direct-Mapped cache with a line size of 1 word and an access time of 1 cycle. Given the following address reference sequence (in Hex):

0x35,0x36,0x38,0xC3,0x3A

- a) Write down how you are partitioning each address (which bits are the Tag, offset, etc.)
- b) In the table below, fill in the proposed Cache's Tag values after each memory reference has been processed. If it is a hit, mark the entry number to indicate this, and if it is a miss enter what the new tag should be. (X indicates the entry is invalid). There may be more Tag Array entries than you need.

| Tag Array | | Contents of Tag Array after processing address (Time ->) | | | | |
|------------------|------------------|---|--------------------|--------------------|--------------------|--------------------|
| Entry Number | Initial Contents | 0x35 (00110101) | 0x36 (00110110) | 0x38 (00111000) | 0xC3 (11000011) | 0x3A (00111010) |
| 0 | X | | | | | |
| 1 | X | | | | | |
| 2 | X | | | | | |
| 3 | X | | | | | |
| 4 | X | | | | | |
| 5 | X | | | | | |
| 6 | X | | | | | |
| 7 | X | | | | | |
| 8 | X | | | | | |
| 9 | X | | | | | |
| 10 | X | | | | | |
| 11 | X | | | | | |
| 12 | X | | | | | |
| 13 | X | | | | | |
| 14 | X | | | | | |
| 15 | X | | | | | |

What is the Average Memory access time for this sequence of references?

Now fill in the contents of the Data array after processing the given address reference.
Write down only the ones that change.

| Data Array | Data Array Contents after processing address |
|--------------|--|
| Entry Number | 0x35 |
| 0 | |
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |
| 10 | |
| 11 | |
| 12 | |
| 13 | |
| 14 | |
| 15 | |

| Memory Contents at Hex Address XY | | | | | | | | | | | | | | | | |
|--|------------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Most Signifi cant Digit (X) | Least Signifi cant Digit (Y) | | | | | | | | | | | | | | | |
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
| 0 | 23 | 20 | 6d | 61 | 74 | 74 | 27 | 73 | 20 | 67 | 76 | 69 | 65 | 77 | 20 | 73 |
| 1 | 68 | 65 | 6c | 6c | 20 | 73 | 63 | 72 | 69 | 70 | 74 | 0a | 73 | 65 | 74 | 20 |
| 2 | 67 | 66 | 69 | 6c | 65 | 20 | 3d | 20 | 24 | 31 | 0a | 09 | 65 | 63 | 68 | 6f |
| 3 | 20 | 2d | 6e | 20 | 22 | 67 | 67 | 72 | 61 | 70 | 68 | 20 | 24 | 67 | 66 | 69 |
| 4 | 6c | 65 | 2e | 2e | 2e | 22 | 20 | 0a | 09 | 65 | 63 | 68 | 6f | 20 | 22 | 2e |
| 5 | 73 | 70 | 20 | 32 | 22 | 20 | 3e | 21 | 20 | 2f | 74 | 6d | 70 | 2f | 74 | 65 |
| 6 | 6d | 70 | 24 | 24 | 2e | 6e | 72 | 0a | 09 | 65 | 63 | 68 | 6f | 20 | 22 | 2e |
| 7 | 70 | 6f | 20 | 2b | 30 | 2e | 35 | 69 | 22 | 20 | 3e | 3e | 20 | 2f | 74 | 6d |
| 8 | 23 | 7b | 92 | 08 | 22 | 41 | 85 | 32 | 69 | 73 | 11 | 35 | 97 | 54 | 31 | 48 |
| 9 | 88 | 73 | 48 | 72 | 98 | 21 | 42 | 85 | 62 | 65 | 90 | 84 | 31 | 56 | 55 | 83 |
| a | 43 | 64 | 84 | 36 | 59 | 3c | 8a | 95 | 3b | 8f | 0e | 41 | 7a | 40 | 2b | 3c |
| b | 4c | d4 | c7 | 82 | a0 | 38 | f9 | c6 | 29 | a3 | d0 | 9c | 7d | 41 | 2b | 75 |
| c | 54 | 69 | 9c | 3b | b0 | 2a | d9 | 3e | 45 | 72 | 6e | f0 | f9 | 3f | a0 | 0a |
| d | 60 | 89 | 43 | d8 | c0 | e7 | 49 | 76 | 59 | 21 | 2c | c8 | a8 | f2 | 87 | 43 |
| e | 76 | 8f | 2e | a9 | ff | 38 | ae | 65 | dd | cf | 21 | 84 | ce | e4 | 34 | 51 |
| f | 8a | 65 | 30 | 2f | c9 | 3a | 58 | 72 | 3e | a0 | 4f | 38 | 96 | 47 | 21 | 80 |

14. (15) Moon decided to experiment with using a smaller, 24-byte 3-way Set Associative Cache (instead of the Direct-mapped Cache) with a line size of 1 word. Remember, the Crater is a byte-addressable machine with a 32-bit word size, an 8-bit bus between processor and memory, and a Main Memory access time of 5 cycles (in addition to the time necessary to to a cache lookup). The Cache access time is still 1 cycle. Given the same address reference sequence (in Hex) as before:

0x35,0x36,0x38,0xC3,0x3A

a) Write down how you are partitioning each address (which bits are the Tag, offset, etc.)

b) In the table below, fill in the proposed Cache's Tag values after each memory reference has been processed. If it is a hit, put an "H" in the tag field, and if it is a miss write down what the new tag should be. Use an LRU replacement scheme, and after each address is processed be sure to indicate the age of the references. There may be more entries than you need. MRU = Most Recently Used, LRU = Least Recently Used.

| Tag Array | | | | Contents of Tag Array after processing address (Time ->) | | | | | | | | | |
|-----------|---------|------------------|-------------|---|-----|--------------------|-----|--------------------|-----|--------------------|-----|--------------------|-----|
| Set # | Entry # | Initial contents | | 0x35 (00110101) | | 0x36 (00110110) | | 0x38 (00111000) | | 0xC3 (11000011) | | 0x3A (00111010) | |
| | | Age | Tag | Age | Tag | Age | Tag | Age | Tag | Age | Tag | Age | Tag |
| 0 | 0 | MRU | 0 1 1 0 0 | | | | | | | | | | |
| | 1 | LRU | 0 1 1 1 0 | | | | | | | | | | |
| | 2 | | 1 1 0 0 0 | | | | | | | | | | |
| 1 | 0 | | 0 0 1 0 0 | | | | | | | | | | |
| | 1 | MRU | 0 0 0 0 1 | | | | | | | | | | |
| | 2 | LRU | 0 1 1 0 1 | | | | | | | | | | |
| 2 | 0 | LRU | 0 0 1 0 0 | | | | | | | | | | |
| | 1 | | 0 1 0 0 1 | | | | | | | | | | |
| | 2 | MRU | 0 0 1 1 0 | | | | | | | | | | |
| 3 | 0 | LRU | 1 0 0 1 0 | | | | | | | | | | |
| | 1 | | 1 0 0 1 0 1 | | | | | | | | | | |
| | 2 | MRU | 1 0 1 1 0 | | | | | | | | | | |

What is the Average Memory access time for this sequence of references?

15. (9 pts) The following tables contain some of the information about a segmented, paged virtual memory system and certain select memory locations. Total physical memory size is 16K bytes, and the page size is 1024 bytes. All numbers in this table are in Hex unless otherwise noted.

| Segment Table | | |
|---------------|--------------|------------|
| Entry Number | Presence Bit | Page Table |
| 0 | 1 | 5 |
| 1 | 0 | 0 |
| 2 | 1 | 0 |
| 3 | 1 | 7 |
| 4 | 1 | 2 |
| 5 | 1 | 2 |
| 6 | 1 | 2 |
| 7 | 1 | 2 |

| Page Table 0 | | | |
|--------------|------------------|-----------|--------------|
| Entry Number | Present? (1=Yes) | Disk Addr | Frame Number |
| 0 | 1 | 1234123 | 0x4 |
| 1 | 0 | 0893748 | 0x7 |
| 2 | 1 | 2489567 | 0x1 |
| 3 | 1 | 9623873 | 0x5 |
| 4 | 1 | B0F6BD3 | 0x2 |
| 5 | 0 | 32829AA | 0x1 |
| 6 | 1 | 56D87AC | 0x0 |
| 7 | 1 | 10A876D | 0x6 |

| Page Table 2 | | | |
|--------------|------------------|-----------|--------------|
| Entry Number | Present? (1=Yes) | Disk Addr | Frame Number |
| 0 | 1 | 1234123 | 0x1 |
| 1 | 0 | 0893748 | 0x3 |
| 2 | 1 | 2489567 | 0x5 |
| 3 | 1 | 9623873 | 0x7 |
| 4 | 1 | BC56BD3 | 0x9 |
| 5 | 0 | 832759E | 0x2 |
| 6 | 1 | 46B37AC | 0x4 |
| 7 | 1 | 810476D | 0x6 |

| Memory | |
|---------|----------|
| Address | Contents |
| 0x00A4 | 0x76 |
| 0x01A4 | 0x73 |
| 0x02A4 | 0x32 |
| 0x03A4 | 0x46 |
| 0x14A4 | 0x30 |
| 0x2AA4 | 0x29 |
| 0x05A4 | 0xa9 |
| 0x09A4 | 0x74 |
| 0x1AA4 | 0x05 |
| 0x3CA4 | 0x23 |
| 0x0DA4 | 0xE3 |
| 0x17A4 | 0xAE |
| 0x26A4 | 0x92 |

| Page Table 5 | | | |
|--------------|------------------|-----------|--------------|
| Entry Number | Present? (1=Yes) | Disk Addr | Frame Number |
| 0 | 1 | 1234123 | 0x2 |
| 1 | 0 | 0893748 | 0x3 |
| 2 | 0 | 2489567 | 0x4 |
| 3 | 1 | 9623873 | 0x4 |
| 4 | 1 | AE76BD3 | 0x6 |
| 5 | 0 | 328759A | 0x7 |
| 6 | 1 | 11D87BE | 0x1 |
| 7 | 1 | 91C875D | 0x2 |

| Page Table 7 | | | |
|--------------|------------------|-----------|--------------|
| Entry Number | Present? (1=Yes) | Disk Addr | Frame Number |
| 0 | 1 | 1234123 | 0x5 |
| 1 | 0 | 0893748 | 0x6 |
| 2 | 1 | 2489567 | 0x1 |
| 3 | 1 | 9623873 | 0x2 |
| 4 | 1 | AE76BD3 | 0x4 |
| 5 | 0 | 328759A | 0x2 |
| 6 | 1 | 56D87AC | 0x5 |
| 7 | 1 | 10A876D | 0x6 |

For each of the following convert the virtual address into a physical address (if possible) and write down the value of the memory location corresponding to the address. If it is not possible to do so, explain why.

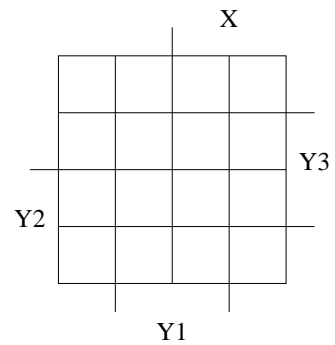
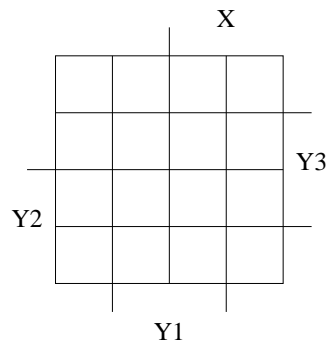
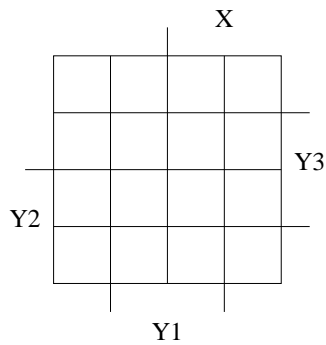
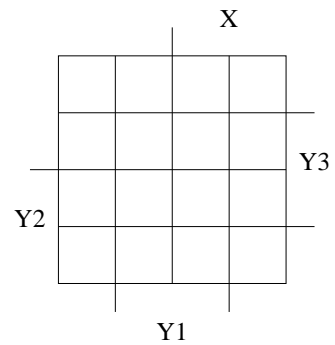
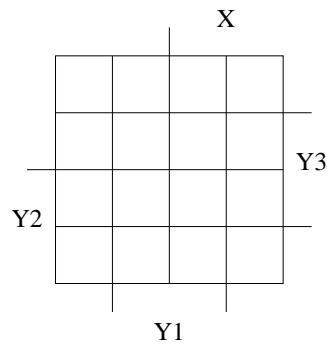
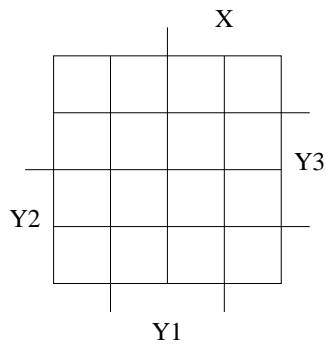
0xB2A4 (1 0 1 1 0 0 1 0 1 0 1 0 0 1 0 0 in binary).

0x5EA4 (0 1 0 1 1 1 1 0 1 0 1 0 0 1 0 0 in binary).

0x28A4 (0 0 1 0 1 0 0 0 1 0 1 0 0 1 0 0 in binary).

16. (16) Given the following table, draw the Karnaugh maps for $Y1'$, $Y2'$, and $Y3'$ and Z in terms of X, Y1, Y2 and Y3, and then write **minimum** boolean equations for each.

| Present State (Y1 Y2 Y3) | Next State | | Output | |
|-----------------------------|----------------------|----------------------|--------|-----|
| | X=0 (Y1' Y2' Y3') | X=1 (Y1' Y2' Y3') | X=0 | X=1 |
| 000 | 111 | 011 | 0 | 0 |
| 001 | 101 | 001 | 0 | 0 |
| 011 | 100 | 000 | 0 | 1 |
| 100 | 001 | 001 | 0 | 0 |
| 101 | 001 | 001 | 1 | 1 |
| 110 | 100 | 000 | 0 | 1 |
| 111 | 100 | 000 | 1 | 1 |



17. (10 pts) Given the following Karnaugh maps, implement the sequential machine using a JK FF for Y1, an SR FF for Y2, and a T FF for Y3. You do not need to draw the gates, but you do need to write down the **minimized** input equations for each of the inputs of each of the Flip Flops in the circuit.

Y1'

| | | | | | |
|----|----|---|---|---|----|
| | | X | | | |
| | 1 | | 1 | d | |
| | | 1 | 1 | 1 | Y3 |
| | | 1 | 1 | | |
| Y2 | 1 | | 1 | d | |
| | Y1 | | | | |

Y2'

| | | | | | |
|----|----|---|---|---|----|
| | | X | | | |
| | 1 | 1 | 1 | d | |
| | | | | | Y3 |
| | | | | | |
| Y2 | 1 | | | d | |
| | Y1 | | | | |

Y3'

| | | | | | |
|----|----|---|---|---|----|
| | | X | | | |
| | 1 | 1 | 1 | d | |
| | 1 | | | 1 | Y3 |
| | 1 | 1 | 1 | 1 | |
| Y2 | 1 | | | d | |
| | Y1 | | | | |

| | | | | | |
|----|----|---|--|--|----|
| | | X | | | |
| | | | | | Y3 |
| | | | | | |
| Y2 | | | | | |
| | Y1 | | | | |

| | | | | | |
|----|----|---|--|--|----|
| | | X | | | |
| | | | | | Y3 |
| | | | | | |
| Y2 | | | | | |
| | Y1 | | | | |

| | | | | | |
|----|----|---|--|--|----|
| | | X | | | |
| | | | | | Y3 |
| | | | | | |
| Y2 | | | | | |
| | Y1 | | | | |

| | | | | | |
|----|----|---|--|--|----|
| | | X | | | |
| | | | | | Y3 |
| | | | | | |
| Y2 | | | | | |
| | Y1 | | | | |

| | | | | | |
|----|----|---|--|--|----|
| | | X | | | |
| | | | | | Y3 |
| | | | | | |
| Y2 | | | | | |
| | Y1 | | | | |

| | | | | | |
|----|----|---|--|--|----|
| | | X | | | |
| | | | | | Y3 |
| | | | | | |
| Y2 | | | | | |
| | Y1 | | | | |