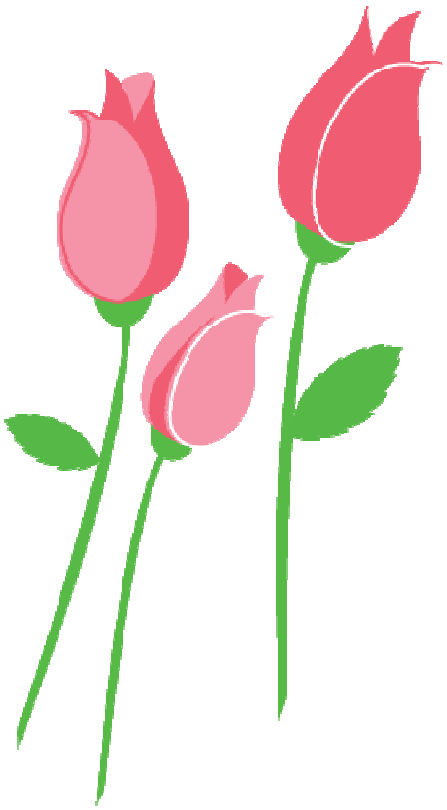
ECE 2030 1:00pm | Computer Engineering | Spring 2009

4 problems, 6 pages | Exam Three | 8 April 2009

*Instructions:* This is a closed book, closed note exam. Calculators are not permitted. If you have a question, raise your hand and I will come to you. Please work the exam in pencil and do not separate the pages of the exam. For maximum credit, show your work.
*Good Luck!*

Your Name (***please print***) _____

| 1 | 2 | 3 | 4 | total |
|---|---|---|---|---|
| | | | | |
| 30 | 24 | 26 | 20 | 100 |



Problem 1 (3 parts, 30 points) Memory Systems

**Part A** (12 points) Consider a DRAM chip organized as **4 million addresses** of **64-bit words**. Assume both the DRAM cell and the DRAM chip are square. The column number and offset concatenate to form the memory address. Using the organization approach discussed in class, answer the following questions about the chip. *Express all answers in decimal (not powers of two)*.

number of columns _____

number of words per column _____

column decoder required (*n* to *m*) _____

total number of bits in address _____

type of mux required (*n* to *m*) _____

number of address lines in column offset _____

**Part B** (10 points) Consider a memory system with **16 million addresses** of **32-bit words** using a **2 million** address by **8-bit word** memory DRAM chip.

**word** address lines for memory system _____

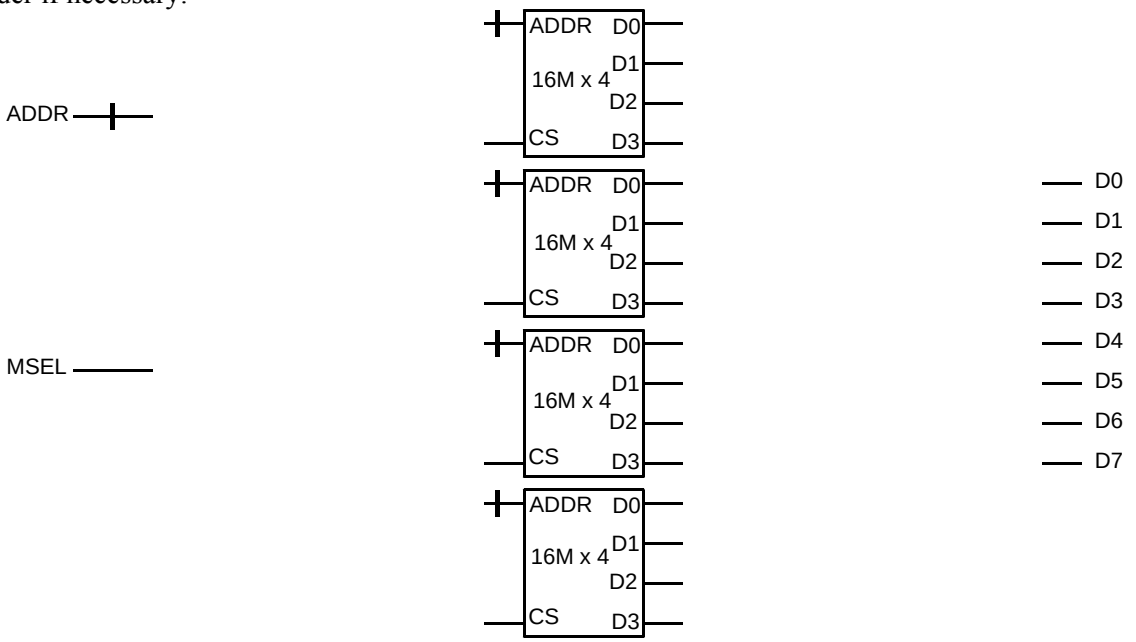
chips needed in one bank _____

banks for memory system _____

memory decoder required (*n* to *m*) _____

DRAM chips required _____

**Part C** (8 points) Design a 32 million address by 8 bit memory system with four 16M x 4 memory chips. *Label all busses and indicate bit width*. Assume R/W is connected and not shown here. Use a bank decoder if necessary.



Problem 2 (3 parts, 24 points) Datapath Elements

**Part A** (6 points) Suppose the following inputs (in `hexadecimal`) are applied to the 32-bit barrel shifter used in the datapath. Determine the output (in `hexadecimal`). Assume the shift amount is drawn from the 16-bit immediate value.

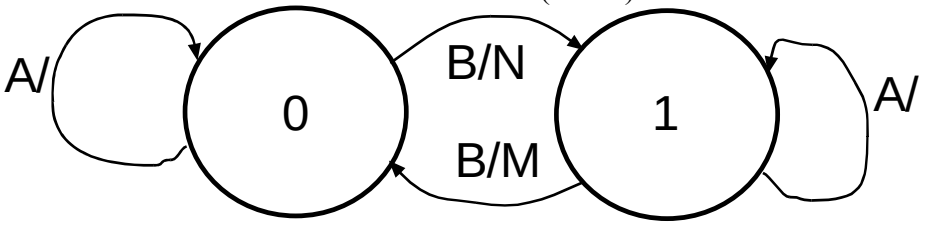
| Shift Type | Shift Amount | Input Value | Output Value |
|---|---|---|---|
| logical | `0xFFF4` | `EB25ACE7` | |
| arithmetic | `0x0008` | `CAB15317` | |
| rotate | `0x000C` | `DE2F1B36` | |

**Part B** (8 points) For each bitwise logical function specification below, determine the LF code (in hexadecimal) to correctly program the logical unit.

| X Y | Out |
|---|---|
| 0 0 | $LF_0$ |
| 1 0 | $LF_1$ |
| 0 1 | $LF_2$ |
| 1 1 | $LF_3$ |

| logical function | LF |
|---|---|
| $\overline{Y}$ | |
| $Y \cdot \overline{X}$ | |
| $X + \overline{Y}$ | |
| $X \cdot Y$ | |

**Part C** (10 points) Given the following finite state diagram, fill in the state table below. The current state variable is S and can be one of two states (0 or 1) and the next state variable is NS.




| $S$ | $A/\overline{B}$ | $NS$ | $M$ | $N$ |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |

Give the Boolean expression for computing NS in terms of the current state and the input.

NS = _____.

Problem 3 (3 parts, 26 points) Microcode

Using the supplied datapath, write microcode fragments to accomplish the following procedures. Express all values, except memory addresses, in hexadecimal notation. Use 'X' when a value is don't cared. For maximum credit, complete the description field. ⊕ means bitwise logical XOR.
**In each part, modify only registers 7 & 8.**

Part A (6 points) $R_7 = 15 \cdot R_8$

| # | X | Y | Z | rwe | im en | im va | au en | -a /s | lu en | lf | su en | st | ld en | st en | r/ -w | msel | description |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | |

Part B (12 points) Compute mem[3000] + 20 and store the result in mem[4000].

| # | X | Y | Z | rwe | im en | im va | au en | -a /s | lu en | lf | su en | st | ld en | st en | r/ -w | msel | description |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | | |

Part C (8 points) Register 7 holds two packed 16 bit binary strings A and B as illustrated below.

| A | B |
|---|---|

31 16 15 0

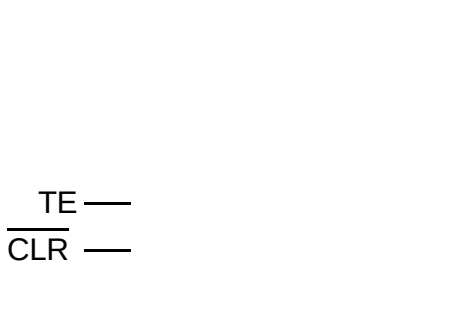
Write a microcode sequence that unpacks A and B and computes $R_8 = A \oplus B$.

| # | X | Y | Z | rwe | im en | im va | au en | -a /s | lu en | lf | su en | st | ld en | st en | r/ -w | msel | description |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | |

Assuming A and B are nonzero, what must be true about A and B for the result in $R_8$ to be zero?

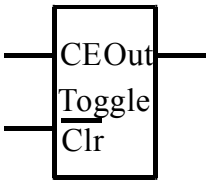
Problem 4 (2 parts, 20 points) Counters

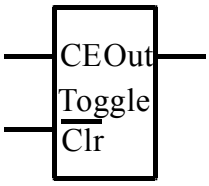
Part A (10 points) Design a toggle cell using two transparent latches, two 2 to 1 muxes, and one inverter. Your toggle cell should have an active high toggle enable input **TE**, and an active low clear input $\overline{CLR}$, clock inputs $\Phi_1$ and $\Phi_2$, and an output **Out**. The $\overline{CLR}$ signal has precedence over **TE**. Label all signals. Also complete the behavior table for the toggle cell.

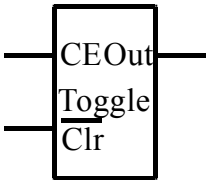



| TE | $\overline{CLR}$ | CLK | Out |
|---|---|---|---|
| 0 | 0 | ↑↓ | |
| 1 | 0 | ↑↓ | |
| 0 | 1 | ↑↓ | |
| 1 | 1 | ↑↓ | |

Part B (10 points) Now combine these toggle cells to build a **divide by seven** counter. Your counter should have an external clear, external count enable, and three count outputs $O_2$, $O_1$, $O_0$. Use any basic gates (AND, OR, NAND, NOR, XOR & NOT) you require. Assume clock inputs to the toggle cells are already connected. Your design must support multi-digit systems.

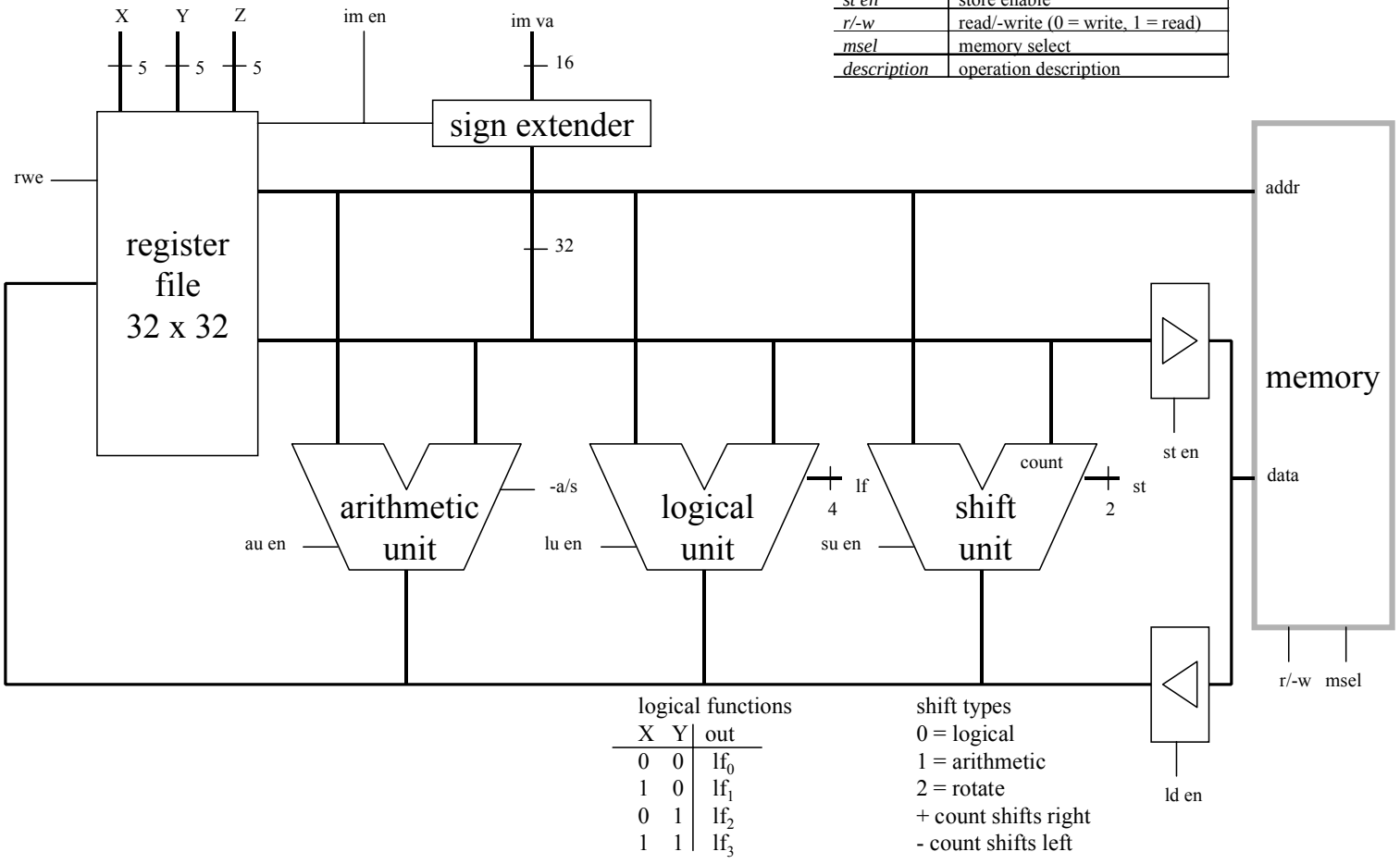


| | |
|---|---|
| *cycle* | cycle number |
| *X* | register driven onto X bus |
| *Y* | register driven onto Y bus |
| *Z* | register written from Z bus |
| *rwe* | register write enable |
| *im en* | immediate enable on Y bus |
| *im va* | immediate value |

| | |
|---|---|
| *au en* | arithmetic unit enable |
| *-a/s* | -add / sub (0 = add, 1 = subtract) |
| *lu en* | logical unit enable |
| *lf* | logical function |
| *su en* | shift unit enable |
| *st* | shift type |
| *ld en* | load enable |
| *st en* | store enable |
| *r/-w* | read/-write (0 = write, 1 = read) |
| *msel* | memory select |
| *description* | operation description |




logical functions

| X | Y | out |
|---|---|---|
| 0 | 0 | $lf_0$ |
| 1 | 0 | $lf_1$ |
| 0 | 1 | $lf_2$ |
| 1 | 1 | $lf_3$ |

shift types
0 = logical
1 = arithmetic
2 = rotate
+ count shifts right
- count shifts left