| ECE 2030 12:00pm | Computer Engineering | Spring 2009 |
|---|---|---|
| 5 problems, 7 pages | Final Exam | 1 May 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 | 5 | total |
|---|---|---|---|---|---|
| | | | | | |
| 32 | 32 | 24 | 28 | 40 | 156 |

Problem 1 (4 parts, 32 points) Implementation Bonanza

For each part implement the specified device. **Label all inputs and outputs**.

Part A (8 points) Implement the expression below using N and P type switches.

$$OUT_X = A \cdot \overline{B} \cdot C + \overline{D} + E$$

Part B (8 points) Implement the expression in mixed logic notation using NAND gates.

$$OUT_Y = \overline{A} \cdot B + (\overline{C + \overline{D}})$$

Part C (8 points) Implement a 2 to 4 decoder with enable using basic gates.

Part D (8 points) Write a POS expression for a two-input XOR (odd parity) using maxterms.

Problem 2 (6 parts, 32 points) Alien Software

SETI has just received an interesting message from deep space. While the comments are written in an alien tongue, they appear to write programs in MIPS assembly. Intergalactic scientists have only been able to decode the register assignments. Computer engineers must take it from there.

```
# INPUTS: $1= num elements, $2= array A pointer, $3= array B pointer,
# OUTPUT: $6=result, WORKING: $4= InA/diff, $5= InB/pred,
```

| # | label | instruction | comment |
|---|---|---|---|
| L1 | WhatsIt: | sub  $6, $6, $6 | |
| L2 | Loop: | lw   $4, ($2) | |
| L3 | | lw   $5, ($3) | |
| L4 | | sub  $4, $4, $5 | |
| L5 | | slt  $5, $4, $0 | |
| L6 | | beq  $5, $0, Skip1 | |
| L7 | | sub  $4, $0, $4 | |
| L8 | Skip1: | slt  $5, $6, $4 | |
| L9 | | beq  $5, $0, Skip2 | |
| L10 | | add  $6, $4, $0 | |
| L11 | Skip2: | addi $2, $2, 4 | |
| L12 | | addi $3, $3, 4 | |
| L13 | | addi $1, $1, -1 | |
| L14 | | bne  $1, $0, Loop | |
| L15 | | jr   $31 | |

Part A – E (26 points) Decode the abstract purpose of code in terms of the defined variable names. Don't transliterate instructions to words.

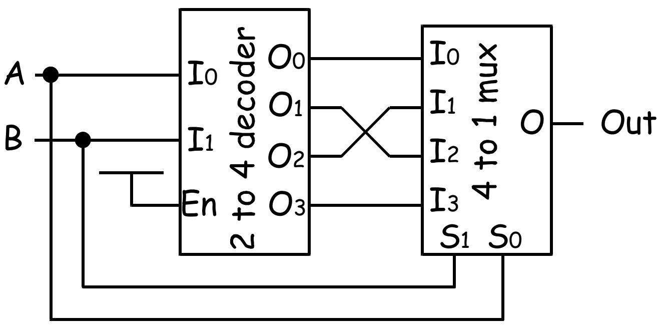
| | |
|---|---|
| A: What does L1 accomplish? | B: What math function do L5-L7 implement? |
| | |
| C: Why is Result updated (in terms of InA, InB)? | D: What is the branch offset in L14 (in bytes)? |
| | |
| E: What does the overall function compute? | |
| | |

Part F (6 points) Another routine calls WhatsIt below. Add missing instructions to preserve and restore its return address on the stack. Recall that $29 is the stack pointer.

| label | instruction | comment |
|---|---|---|
| | | # push return address … |
| | | # … on stack |
| | jal  WhatsIt | # call WhatsIt |
| | | # pop return address … |
| | | # … off stack |

Problem 3 (3 parts, 24 points) Agents changed the matrix

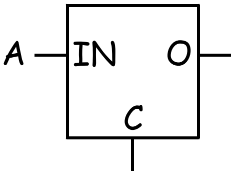
Part A (8 points) Consider the circuit below. Complete the truth table. Then state what logical function this circuit implements.



| A | B | Out |
|---|---|---|
| 0 | 0 | |
| 1 | 0 | |
| 0 | 1 | |
| 1 | 1 | |

This wacky circuit is a ________________________

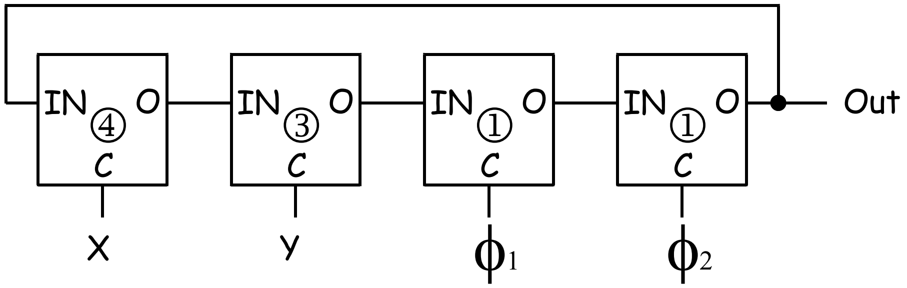
Part B (8 points) Consider four different function definitions below. The symbolic value A is presented at its input. The control input and resulting out are shown in the truth table. Name the *gate, building block, or storage device* that implements each definition.



| In | C | ① | ② | ③ | ④ |
|---|---|---|---|---|---|
| A | 0 | $Q_0$ | 1 | 0 | A |
| A | 1 | A | $\overline{A}$ | A | $\overline{A}$ |

① ② ③ ④

Part C (8 points) Blocks from part B are used to create a new module below. The symbolic values X and Y are presented at its inputs along with a two-phase clock. Complete the truth table and give its functional name.



| X | Y | Clk | Out |
|---|---|---|---|
| 0 | 0 | ↑↓ | |
| 1 | 0 | ↑↓ | |
| 0 | 1 | ↑↓ | |
| 1 | 1 | ↑↓ | |

It’s a: ________________________

Problem 4 (2 parts, 28 points) Microcode

Using the supplied datapath, write microcode fragments to accomplish the following procedures. Express all values in hexadecimal notation. Use ‘X’ when a value is don’t cared. For maximum credit, complete the description field.

Part A (14 points) $R_7 = \frac{3 \times R_5}{16} - 256 \times R_6$ **Modify only $R_5$, $R_6$ and $R_7$.**

| # | *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 | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | | |

Part B (14 points) Write a microcode sequence that loads a 32 bit word from memory location 0x4000, unpacks and averages two 15 bit unsigned values (A and B), and then stores the result back to memory location 0x4000. Assume the most significant two bits of the register are zero. **Modify only $R_1$, $R_2$, and $R_3$.**

| | | | |
|---|---|---|---|
| 0 | 0 | A | B |
| 31 | 30 | 29 15 | 14 0 |

| # | *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 | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | | |

Problem 5 (4 parts, 40 points) This and That

Part A (9 points) Consider the instruction set architecture below with fields containing zeros.

| | | | |
|---|---|---|---|
| 0 0000 | 000 0000 | 000 0000 | 000 0000 0000 0000 |
| opcode | dest. reg. | source 1 reg. | immediate value |

What is the maximum number of opcodes? ________________

What is the number of registers? ________________

What is the range of the signed immediate value? ________________

Part B (9 points) List three differences between a branch and a jump in the MIPS ISA.

1: ________________

2: ________________

3: ________________

Part C (12 points) For 32 bit representations below, determine the most positive value and the step size (difference between sequential values). **All answers should be expressed in decimal notation**. Fractions (e.g., 3/16ths) may be used. Signed representations are two's complement.

| representation | most positive value | step size |
|---|---|---|
| unsigned integer<br>(32 bits) . (0 bits) | | |
| signed fixed-point<br>(28 bits) . (4 bits) | | |
| signed fixed-point<br>(25 bits) . (7 bits) | | |
| signed fixed-point<br>(21 bits) . (11 bits) | | |

Part D (10 points) Consider a memory system with **256 million addresses** of **8 byte words** using DRAM chips organized as **16 million** addresses by **32 bit words.**

**word** address lines for memory system ________________

chips needed in one bank ________________

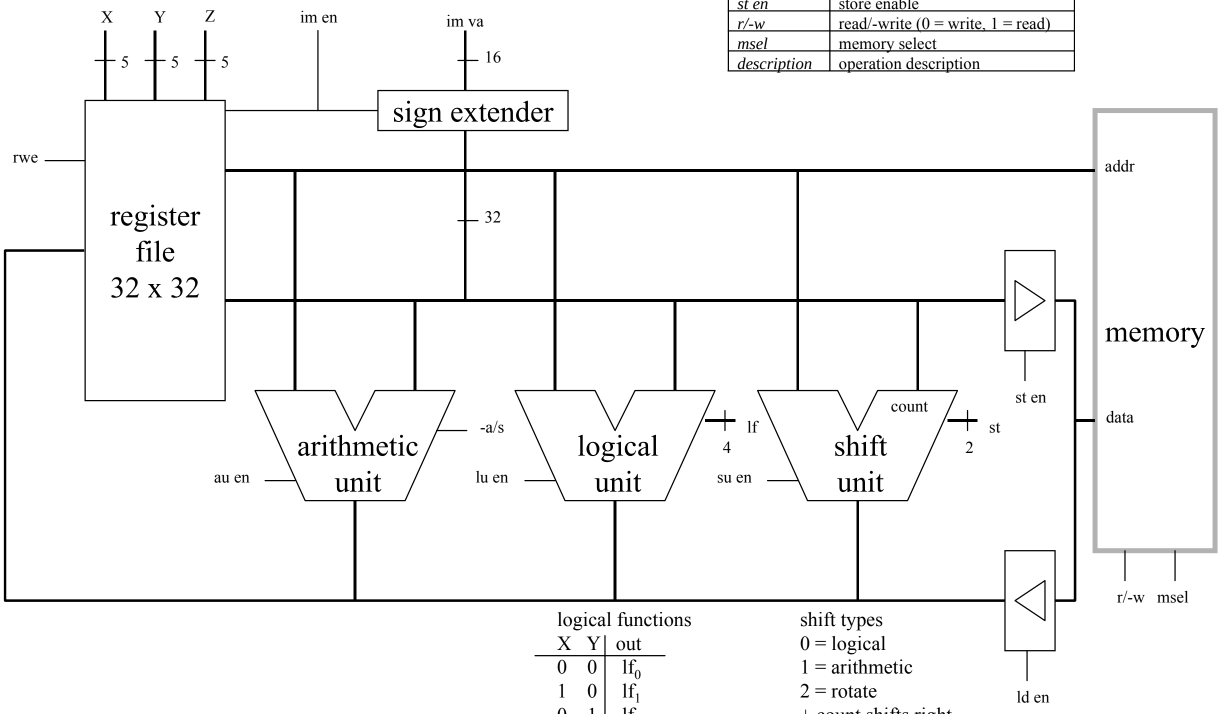
banks for memory system ________________

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

total memory system size (in bytes) ________________

| | |
|---|---|
| *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

| instruction | example | meaning |
|---|---|---|
| add | `add $1,$2,$3` | `$1 = $2 + $3` |
| subtract | `sub $1,$2,$3` | `$1 = $2 - $3` |
| add immediate | `addi $1,$2,100` | `$1 = $2 + 100` |
| multiply | `mul $1,$2,$3` | `$1 = $2 * $3` |
| divide | `div $1,$2,$3` | `$1 = $2 / $3` |
| and | `and $1,$2,$3` | `$1 = $2 & $3` |
| or | `or $1,$2,$3` | `$1 = $2 \| $3` |
| xor | `xor $1,$2,$3` | `$1 = $2 xor $3` |
| and immediate | `andi $1,$2,100` | `$1 = $2 & 100` |
| or immediate | `ori $1,$2,100` | `$1 = $2 \| 100` |
| xor immediate | `xori $1,$2,100` | `$1 = $2 xor 100` |
| shift left logical | `sll $1,$2,5` | `$1 = $2 << 5 (logical)` |
| shift right logical | `srl $1,$2,5` | `$1 = $2 >> 5 (logical)` |
| shift left arithmetic | `sla $1,$2,5` | `$1 = $2 << 5 (arithmetic)` |
| shift right arithmetic | `sra $1,$2,5` | `$1 = $2 >> 5 (arithmetic)` |
| load word | `lw $1, ($2)` | `$1 = memory [$2]` |
| store word | `sw $1, ($2)` | `memory [$2] = $1` |
| load upper immediate | `lui $1,100` | $\$1 = 100 \times 2^{16}$ |
| branch if equal | `beq $1,$2,100` | `if ($1 = $2), PC = PC + 4 + (100*4)` |
| branch if not equal | `bne $1,$2,100` | `if ($1 ≠ $2), PC = PC + 4 + (100*4)` |
| set if less than | `slt $1, $2, $3` | `if ($2 < $3), $1 = 1 else $1 = 0` |
| set if less than immediate | `slti $1, $2, 100` | `if ($2 < 100), $1 = 1 else $1 = 0` |
| jump | `j 10000` | `PC = 10000` |
| jump register | `jr $31` | `PC = $31` |
| jump and link | `jal 10000` | `$31 = PC + 4; PC = 10000` |