Intro. to Computer Architecture	**Homework 4**
CSE 240 Autumn 2005	**DUE: Mon. 10 October 2005**

Write your answers on these pages. Additional pages may be attached (with staple) if necessary. Please ensure that your answers are legible. Please show your work. Due at the ***beginning of class***. Total points: 80.

1. [6 Points] **Instruction Encoding.** Suppose a machine encodes instructions in 32 bits according to the following format. Also, suppose the encoding must accommodate 164 opcodes and 50 registers.

OPCODE	SR	DR	IMM

   (a) What is the minimum number of bits required to represent the OPCODE field?

   (b) What is the minimum number of bits required to represent each of the register fields (*e.g.*, DR)?

   (c) What is the greatest number of bits that are left for the IMM field? If the IMM field encodes a 2's complement integer, what range of values can be represented with these bits?

2. [8 Points] **Instruction Processing.** The PC, IR, MAR, MDR, and RF (register file) are structures written (and read) in various phases of the instruction processing cycle, depending on the instruction being executed. In each table cell, below, enter the instructions that result in *writes* to the corresponding structure (row) during the corresponding phase (column) of the instruction processing cycle. For example, place LDR in the PC row of the FETCH column if LDR causes the PC to be written during the FETCH phase. To make this simpler, let's only consider the following instructions: ADD, LDR, LEA, ST, and TRAP. The textbook doesn't directly answer this question (*i.e.*, you can't just "look it up"), so your answers will need to be based on your understanding of what should happen in each execution phase.

	FETCH	DECODE	EVAL ADDR	FETCH OPERANDS	EXECUTE	STORE
PC						
IR						
MAR						
MDR						
RF						

3. [20 Points] **LC-3 Instruction Encoding.** For these questions assume the LC-3 instruction encoding (summarized on the inside back cover of your textbook). You may also find Appendix A helpful.

   (a) What is the range of values (in decimal) that may be specified by the immediate field of an ADD instruction?

   (b) Give the encoding of two LC-3 instructions that together decrement register R3 by 31 and place the result back in R3. Complete the following table.

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Instruction
x3001																	
x3002																	

(c) Different kinds of store instructions (*e.g.*, ST, STI, and STR) are able to access different parts of the address space. For each kind of store instruction, below, give the possible range of addresses to which the instruction can store (assuming the instruction resides in memory at address x2345 and that the rest of memory can contain any possible values). Express your answers as hexadecimal address ranges (*e.g.*, x100 to x200).

i. ST

ii. STI

iii. STR (with BaseR=R1=xCEA3)

(d) Like store instructions, different kinds of control instructions are able to transfer control to different parts of the address space. For each of the following control instructions, give the possible range of values for the next PC (assuming the instruction resides in memory at address x3456 and the registers have the values below). Express your answers as hexadecimal address ranges (*e.g.*, x100 to x200).

R0	R1	R2	R3	R4	R5	R6	R7
x0	x1111	x2222	x3333	x4444	x5555	x6666	x7777

i. BRn

ii. BRnzp

iii. JMP(with BaseR=R3)

iv. RET

4. [4 Points] **Simple LC-3 Code.** Consider the following LC-3 program. Register R1 serves as the input value for this code, and R2 holds the output value when the loop terminates. If R1 originally contains some positive integer, *n*, express the output, R2, in terms of *n*. Assume the values manipulated by this program are small enough that overflow does not occur.

Address	Operation	Operation
x3001	0101 0100 1010 0000	R2 <- 0
x3002	0001 0100 1010 0001	R2 <- R2+1
x3003	0001 0100 1000 0010	R3 <- R2+R2
x3004	0001 0100 1000 0011	R2 <- R2+R3
x3005	0001 0010 0111 1111	R1 <- R1-1
x3006	0000 0011 1111 1100	BRp x3003

5. [15 Points] **LC-3 Code.** Suppose you want to write a program consisting of instructions with the behavior described by the operation in the final column of the following table.

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Operation
x3001																	R2 <- M[R1+0]
x3002																	R3 <- M[R1+1]
x3003																	R4 <- NOT R3
x3004																	R4 <- R4 + 1
x3005																	R5 <- R2 + R4
x3006																	BRzp x3009
x3007																	M[R1+2] <- R3
x3008																	BRnzp x3010
x3009																	M[R1+2] <- R2

(a) Give the binary encoding of each instruction in the table. Write your answers in the table, above.

(b) Trace the execution of the above program, starting at x3001, by completing the following table. Give the PC and operation to execute in the first two columns, and give the state of the registers and condition codes *after* the execution of that instruction (**leave an entry blank if it is not changed by the instruction**). The initial register state and the effect of the first instruction are given in the first two rows. Assume memory locations x3100 and x3101 contain 14 and 27, respectively.

PC	Operation	R0	R1	R2	R3	R4	R5	R6	R7	CCs
initial register state ⇒		0	x3100	0	3	4	5	6	7	
x3001	R2 <- M[R1+0]			14						P

(c) In a sentence, what does this code compute?

6. [15 Points] **LC-3 Code.** The following (bit-level) memory contents represent an LC-3 program.

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Operation
x3001	0	1	1	0	0	1	0	0	0	1	0	0	0	0	0	0	R2 <- M[R1+0]
x3002	0	1	1	0	0	1	1	0	0	1	0	0	0	0	0	1	
x3003	1	0	0	1	0	1	1	0	1	1	1	1	1	1	1	1	
x3004	0	0	0	1	0	1	1	0	1	1	1	0	0	0	0	1	
x3005	0	0	0	1	0	1	0	0	1	0	0	0	0	0	1	1	
x3006	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	
x3007	1	0	0	1	0	1	0	0	1	0	1	1	1	1	1	1	
x3008	0	0	0	1	0	1	0	0	1	0	1	0	0	0	0	1	
x3009	0	1	1	1	0	1	0	0	0	1	0	0	0	0	1	0	

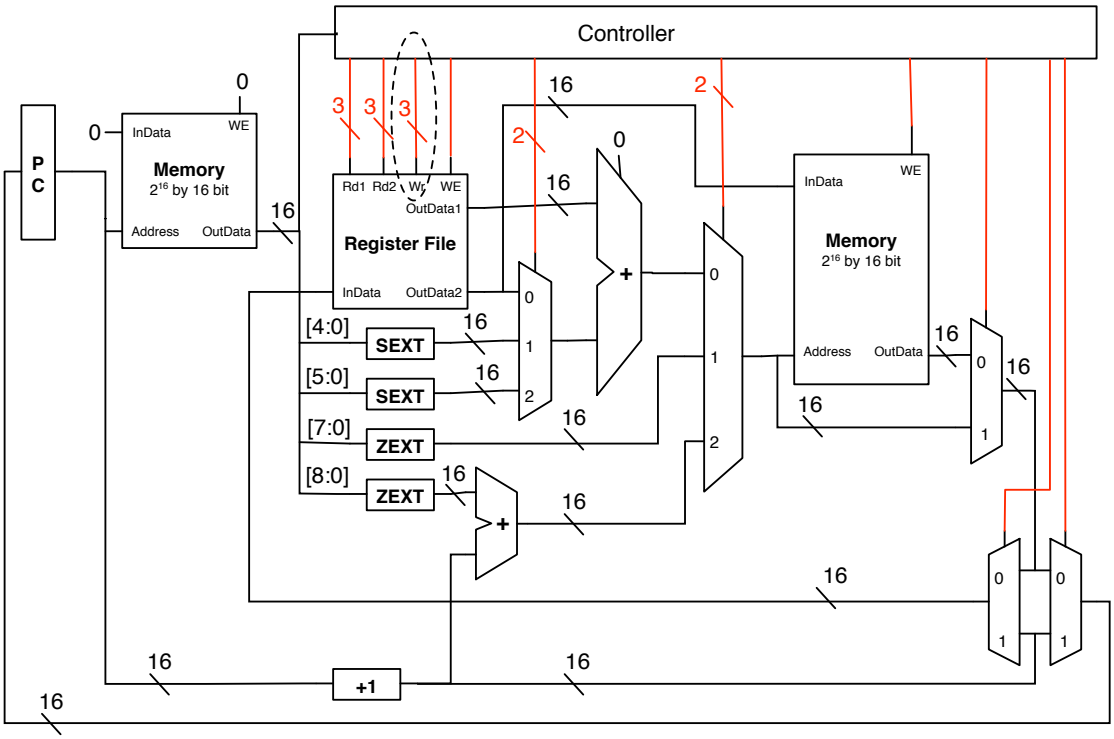
(a) First, determine what each instruction does. Write this next to each instruction (above, in the final column) in a manner similar to that of the previous problem.

(b) Next, trace the execution of the above program, starting at x3001, by completing the following table. Give the PC and operation to execute in the first two columns, and give the state of the registers and condition codes *after* the execution of that instruction (**leave an entry blank if it is not changed by the instruction**). The initial register state and the effect of the first instruction are given in the first two rows. Assume memory locations x3100 and x3101 contain -34 and -20, respectively.

PC	Operation	R0	R1	R2	R3	R4	R5	R6	R7	CCs
initial register state $\Rightarrow$		0	x3100	2	3	4	5	6	7	
x3001	R2 <- M[R1+0]			-34						N

(c) In a sentence, what does this code do?

7. [12 Points] **LC-3 Implementation.** Consider the LC-3 datapath, below, from lecture. We are going to build the control logic (*i.e.*, part of the contents of the "Controller" box) for the write register control lines (circled, below). Based on the currently executing instruction, the Controller generates a value (from 0 to 7, encoded as a 3-bit unsigned binary integer) on the write register control lines (WR) to specify what register should be written by this instruction. For instructions that do not write to the register file (*e.g.*, JMP), the Controller can generate any values on the WR because they will be ignored anyway.



(a) Before we build the circuit to generate the WR, let's determine what these control lines should be for each opcode by completing the following table. When the value of the WR come from bits of the instruction, please specify exactly what bits are to be used. For example, I[8:6] specifies the first operand of an ADD instruction. When the WR need to have a constant value, specify that value as a decimal (e.g., 7). When it doesn't matter what the WR is (because write will not be enabled), leave the box for that opcode blank.

Instruction	*Opcode*	*Register write control lines*
ADD	0001	
AND	0101	
BR	0000	
JMP	1100	
JSR/JSRR	0100	
LD	0010	
LDI	1010	
LDR	0110	
LEA	1110	
NOT	1001	
RTI	1000	
ST	0011	
STI	1011	
STR	0111	
TRAP	1111	

(b) Build a gate-level circuit that generates the bits of the WR control lines. The input to this circuit will be all the bits of the instruction (I[15] - I[0]). You may use as many or few of these bits as you need. The output will be the 3-bit WR control lines. Please structure your circuit around a 3-bit 2-to-1 mux. The inputs to the mux should be determined by your answers for part (a), above; and the output will be the WR. You will need to build a simple circuit to generate the 1-bit selector line based on the instruction opcode.

8. [No Points] **Last and Most Important Question!** Please complete this question, and give us your feedback!

   (a) How many hours did you spend on this assignment?

   (b) On a scale of 1-5, how difficult did you find this assignment? (1-easiest, 5- most difficult)

   (c) Do you have any other comments on your experience completing this assignment? What are they?