

Write your answers on these pages. Additional pages may be attached (with staple) if necessary. Please ensure that your answers are legible and *show your work*. Write your name at the top of each page. Due at the *beginning of class*. Total points: 62

1. [6 Points] **Instruction Encoding.** Suppose a machine encodes instructions in 32 bits according to the following format. Also, suppose the encoding must accommodate 110 opcodes and 28 registers.

OPCODE	SR	DR	IMM
--------	----	----	-----

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

Answer: Six bits can represent 64 values ($2^6 = 64$), so (in this case) six isn't enough. Seven bits can represent 128 values ($2^7 = 128$), so we need seven bits to represent 110 values.

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

Answer: First, it is important to observe that both register fields (*i.e.*, SR, and DR) can name all registers, so they will all be the same size. Four bits isn't enough, because $2^4 = 16 < 28$. Five bits is just enough, because $2^5 = 32 > 28$.

- (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?

Answer: We know the whole instruction is 32 bits and the opcode consumes seven bits and each register field consumes five bits (for a total of $5 \times 2 = 10$ bits). We are then left with $32 - (7 + 10) = 15$ bits for the immediate field. In 15 bits we can represent $2^{15} = 32768$ different values, specifically the range of integers from $-2^{14} = -16384$ to $2^{14} - 1 = 16383$.

2. [12 Points] **LC-3 Instruction Encoding.** For these questions assume the LC-3 instruction encoding (inside the back cover of your textbook). You may also want to consult Appendix A.

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

Answer: The immediate field in an AND instruction is five bits. This immediate represents a 2's complement number, so integer values from -16 to 15 may be represented.

(b) What is the range of values (in decimal) that may be specified by the PCoffset field in a BR instruction?

Answer: The PCoffset field in a BR instruction is a 9-bit 2's complement number. Therefore, it may represent integer values from -256 to 255.

(c) What is the range of values (in decimal) that may be specified by the offset field in an LDR instruction?

Answer: The offset field in an LDR instruction is a 6-bit 2's complement number. Therefore, it may represent integer values from -32 to 31.

(d) What is the relationship between JMP and RET? In particular, why do they have the same bits in the opcode field?

Answer: Both instructions impact control flow by updating the PC. JMP places the contents of any register into the PC, while RET places the contents of R7 in the PC. In fact, if you examine the instruction encodings of both (p. 529) you will see that RET is encoded just like a JMP except the BaseR field is set to R7. From this we can conclude that there is not a special RET machine instruction. Rather, it is a special case of JMP.

(e) Give the encoding of two LC-3 instructions that together increment register R3 by 20. Complete the following table.

Answer: Only five bits are available to encode an immediate operand in an ADD instruction. With a 5-bit 2's complement value we can represent values from -16 to 15, so we can not increment by 20 in one instruction. Instead, we can increment by 15 in the first instruction and 5 in the next.

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Instruction
x3001	0	0	0	1	0	1	1	0	1	1	1	0	1	1	1	1	R3 <- R3 + 15
x3002	0	0	0	1	0	1	1	0	1	1	1	0	0	1	0	1	R3 <- R3 + 5

3. [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	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	R3 <- M[R1+1]
x3003	1	0	0	1	1	0	0	0	1	1	1	1	1	1	1	1	R4 <- NOT R3
x3004	0	0	0	1	1	0	0	1	0	0	1	0	0	0	0	1	R4 <- R4 + 1
x3005	0	0	0	1	1	0	1	0	1	0	0	0	0	1	0	0	R5 <- R2 + R4
x3006	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	BRzp x3009
x3007	0	1	1	1	0	1	1	0	0	1	0	0	0	0	1	0	M[R1+2] <- R3
x3008	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	BRnzp x3010
x3009	0	1	1	1	0	1	0	0	0	1	0	0	0	0	1	0	M[R1+2] <- R2

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

Answer: See 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 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.

Answer:

PC	Operation	R0	R1	R2	R3	R4	R5	R6	R7	CCs	M[x3102]
	<i>initial state</i> ⇒	0	x3100	0	3	4	5	6	7		0
x3001	R2 <- M[R1+0]			14						P	
x3002	R3 <- M[R1+1]				27					(P)	
x3003	R4 <- NOT R3					-28				N	
x3004	R4 <- R4 + 1					-27				(N)	
x3005	R5 <- R2 + R4						-13			(N)	
x3006	BRzp x3009										
x3007	M[R1+2] <- R3										27
x3008	BRnzp x3010										

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

Answer: This code determines the greater value between M[R1+0] and M[R1+1], and stores it to M[R1+2].

4. [16 Points] **LC-3**. 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	Instruction
x3001	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	R1 <- M[R0+0]
x3002	0	1	0	1	0	1	0	0	1	0	1	0	0	0	0	0	R2 <- R2 AND 0
x3003	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	1	R3 <- M[R0+1]
x3004	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	1	R2 <- R2 + R1
x3005	0	0	0	1	0	1	1	0	1	1	1	1	1	1	1	1	R3 <- R3 + -1
x3006	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	1	BRp x3004
x3007	0	1	1	1	0	1	0	0	0	0	0	0	0	0	1	0	M[R0+2] <- R2

- (a) First, determine what each instruction does. Write this next to each instruction (above) 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 instruction 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 has not been changed by the instruction**). The initial state and the effect of the first instruction are given in the first two rows. Assume memory locations x3100 and x3101 contain 3 and 2, respectively.

Answer:

PC	Instruction	R0	R1	R2	R3	R4	R5	R6	R7	CCs	M[x3102]
	<i>initial state</i> ⇒	x3100	1	2	3	4	5	6	7		0
x3001	R1 <- M[R0+0]		3							P	
x3002	R2 <- R2 AND 0			0						Z	
x3003	R3 <- M[R0+1]				2					P	
x3004	R2 <- R2 + R1			3						(P)	
x3005	R3 <- R3 + -1				1					(P)	
x3006	BRp x3004										
x3004	R2 <- R2 + R1			6						(P)	
x3005	R3 <- R3 + -1				0					Z	
x3006	BRp x3004										
x3007	M[R0+2] <- R2										6

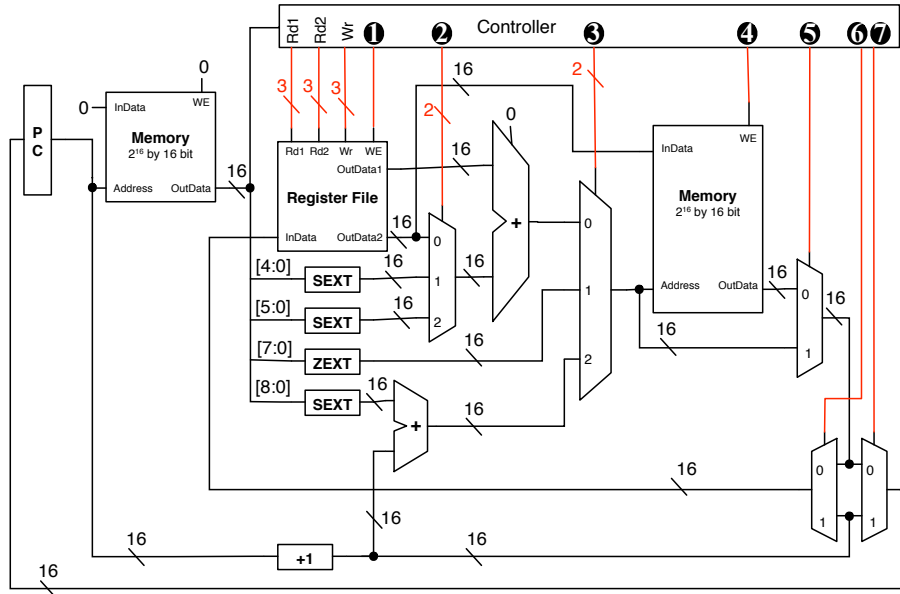
- (c) Describe what this code does, assuming execution starts at address x3001. What registers or memory serve as input to this code? And what registers or memory serve as output? Be very careful in determining the input and output (*i.e.*, just because a register appears in the code does not mean that it is input or output).

Answer: This code performs multiplication. R0 is an input register containing the address of the input and output in memory. The memory at addresses R0+0 and R0+1 contain the two operands to multiply. The result is stored in memory at address R0+2.

- (d) Under what circumstances will this program fail to perform its principal task?

Answer: This program will fail to multiply when the second operand (at R0+1) is 0 or negative. Note that it *will* work when the first operand (at R0+0) is 0 or negative.

5. [12 Points] **LC-3 Data-path.** Consider the single-cycle LC-3 data-path from lecture.



Give the control lines for the instructions in the table, below. In each box, place either a constant (*e.g.*, 2), one or more bits from the instruction (*e.g.*, I[8:6]), or “X” indicating that the value on the control line does not matter.

Name	Opcode		Registers			Control						
	I[15:12]	I[5]	Rd1	Rd2	Wr	①	②	③	④	⑤	⑥	⑦
ADD	0001	0	I[8:6]	I[2:0]	I[11:9]	1	0	0	0	1	0	1
ADD Immed	0001	1	I[8:6]	x	I[11:9]	1	1	0	0	1	0	1
LDR	0110	—	I[8:6]	x	I[11:9]	1	2	0	0	0	0	1
LD	0010	—	x	x	I[11:9]	1	x	2	0	0	0	1
STR	0111	—	I[8:6]	I[11:9]	x	0	2	0	1	x	x	1
ST	0011	—	x	I[11:9]	x	0	x	2	1	x	x	1
LEA	1110	—	x	x	I[11:9]	1	x	2	0	1	0	1
JMP	1100	—	I[8:6]	x	x	0	2*	0	0	1	x	0
JSRR	0100	—	I[8:6]	x	7	1	2*	0	0	1	1	0

*Because the low-order 6 bits of the JMP instruction encoding are all zeros, this control signal causes the mux to select 0.