Your Name: ______________________________ netid:________________ Group #:
Name: ______________________________ netid:________________
Name: ______________________________ netid:________________
Name: ______________________________ netid:________________

ECE 120 Worksheet 11: LC-3 Instructions

In this discussion, you will be given a sequence of binary words that correspond to LC-3 instructions and you will be asked to convert each binary word to a corresponding LC-3 instruction. You will then explain the function performed by the sequence of the given instructions.

LC-3 Instruction Set

Instruction	15 14 13 12	11 10 9	8 7 6	5 4 3 2 1 0	Operation
ADD+	0001	DR	SR1	0 00 SR2	DR ← SR1 + SR2; set NZP
ADD+	0001	DR	SR1	1 imm5	DR ← SR1 + SEXT(imm5); set NZP
AND+	0101	DR	SR1	0 00 SR2	DR ← SR1 AND SR2; set NZP
AND+	0101	DR	SR1	1 imm5	DR ← SR1 AND SEXT(imm5); set NZP
BR	0000	n z p	PCoffset9		IF ((n·N)+(z·Z)+(p·P)) THEN PC ← PC + SEXT(PCoffset9)
JMP	1100	000	BaseR	000000	PC ← BaseR
JSR	0100	1 PCoffset11			R7 ← PC PC ← PC + SEXT(PCoffset11)
JSRR	0100	0 00	BaseR	000000	R7 ← PC PC ← BaseR
LD+	0010	DR	PCoffset9		DR ← M[PC + SEXT(PCoffset9)]; Set NZP
LDI+	1010	DR	PCoffset9		DR ← M[M[PC + SEXT(PCoffset9)]]; Set NZP
LDR+	0110	DR	BaseR	offset6	DR ← M[BaseR + SEXT(offset6)]; Set NZP
LEA+	1110	DR	PCoffset9		DR ← PC + SEXT(PCoffset9); Set NZP
NOT+	1001	DR	SR	111111	DR ← NOT(SR); Set NZP
RET	1100	000	111	000000	PC ← R7
RTI	1000	000000000000			IF (PSR[15]==0) THEN PC ← M[R6]; R6 ← R6 + 1; TEMP ← M[R6]; R6 ← R6 + 1; PSR ← TEMP
ST	0011	SR	PCoffset9		M[PC + SEXT(PCoffset9)] ← SR
STI	1011	SR	PCoffset9		M[M[PC + SEXT(PCoffset9)]] ← SR
STR	0111	SR	BaseR	offset6	M[BaseR + SEXT(offset6)] ← SR
TRAP	1111	0000	trapvect8		R7 ← PC PC ← M[ZEXT(trapvect8)]

superscript "+" denotes instructions that update the condition bits NZP

1. LC-3 Instructions

Shown on the right is a snapshot of a portion of the contents of the LC-3 memory for addresses x3000-x3009. The memory contains a short program and data the program operates on. The 16-bit addresses of, and data in, the RAM are encoded in hexadecimal representation. In this discussion, you will interpret the contents of the RAM, trace the program, and determine its functionality.

address	**value**
x3000	x2206
x3001	x2406
x3002	x94BF
x3003	x14A1
x3004	x1642
x3005	x3603
x3006	xF025
x3007	x0005
x3008	x0002
x3009	x0000

1) Translate the contents of the RAM from address x3000 to x3005 into its LC-3 instructions and write them in RTL notation. A copy of the encoding for the LC-3 instruction set appears on page 1. The first one has been done for you as an example.

Address	Binary Instruction (translate from hex above)	RTL (be specific to this instruction)
x3000	0010 001 000000110	R1 ← M[x3007], set CC
x3001		
x3002		
x3003		
x3004		
x3005		

2) Assuming PC is initially set to x3000, trace the execution of the given program segment for **six** instruction cycles, filling in the table below. The first one has been done for you as an example. Write down the values stored in the PC, IR, MAR, MDR, N, Z, P, R1, R2, and R3 registers at the end of the instruction cycle. Values for PC, IR, MAR, MDR, R1, R2, and R3 should be written in hexadecimal. Values for N, Z, and P should be written in binary. If a value cannot be determined, white "U" for *unknown*.

PC	IR	MAR	MDR	R1	R2	R3	N	Z	P
x3001	x2206	x3007	x0005	x0005	U	U	0	0	1

3) What hexadecimal value will be stored in R3 after the six instruction cycles?

Answer: ________________________

4) What is the address of the next instruction to be executed after the six instruction cycles?

Answer: ________________________

5) What is the address of the last memory access (read or write, whichever happened last) operation after six instruction cycles?

Answer: ________________________

6) Examine the sequence of your instructions from address x3000 to x3005. What function do they perform? Your description should explain the high level behavior of the program in a single sentence or formula and should not be a step-by-step description of what the program did. For example, "First the program adds R1 to R2 and stores it into R3…" is **unacceptable**.