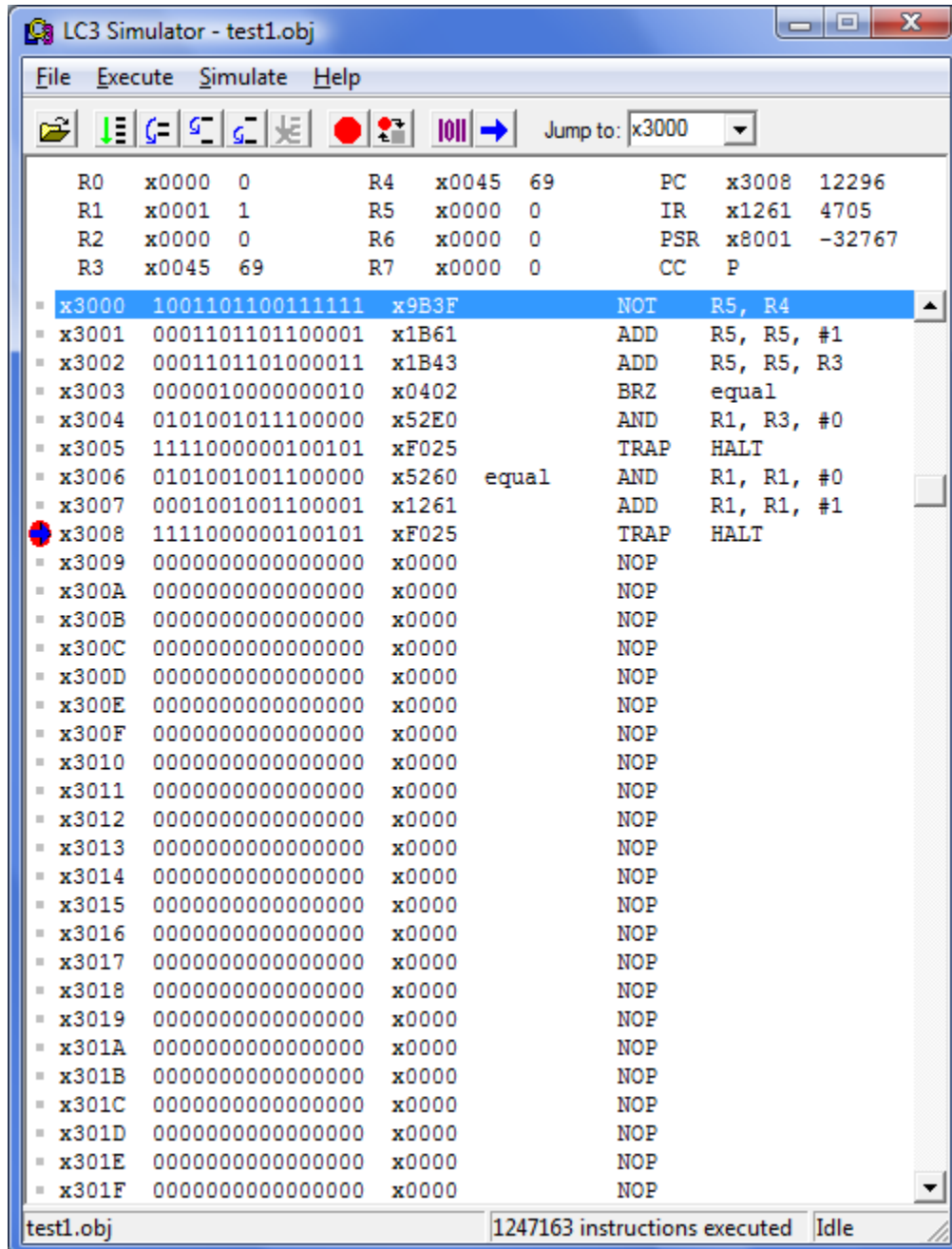


Solution for Homework -6

Problem 1

Write a LC3 program which checks for the equality of two numbers in R3 and R4. If the numbers are equal, register R1 must contain the value 1. Else, it must contain the value 0. For the screenshot, make R3 = x0045 and R4 = x0045



```
LC3 Simulator - test1.obj
File  Execute  Simulate  Help
Jump to: x3000
R0  x0000  0      R4  x0045  69      PC  x3008  12296
R1  x0001  1      R5  x0000  0      IR  x1261  4705
R2  x0000  0      R6  x0000  0      PSR x8001  -32767
R3  x0045  69      R7  x0000  0      CC  P
x3000 1001101100111111 x9B3F NOT R5, R4
x3001 0001101101100001 x1B61 ADD R5, R5, #1
x3002 0001101101000011 x1B43 ADD R5, R5, R3
x3003 0000010000000010 x0402 BRZ equal
x3004 0101001011100000 x52E0 AND R1, R3, #0
x3005 1111000000100101 xF025 TRAP HALT
x3006 0101001001100000 x5260 equal AND R1, R1, #0
x3007 0001001001100001 x1261 ADD R1, R1, #1
x3008 1111000000100101 xF025 TRAP HALT
x3009 0000000000000000 x0000 NOP
x300A 0000000000000000 x0000 NOP
x300B 0000000000000000 x0000 NOP
x300C 0000000000000000 x0000 NOP
x300D 0000000000000000 x0000 NOP
x300E 0000000000000000 x0000 NOP
x300F 0000000000000000 x0000 NOP
x3010 0000000000000000 x0000 NOP
x3011 0000000000000000 x0000 NOP
x3012 0000000000000000 x0000 NOP
x3013 0000000000000000 x0000 NOP
x3014 0000000000000000 x0000 NOP
x3015 0000000000000000 x0000 NOP
x3016 0000000000000000 x0000 NOP
x3017 0000000000000000 x0000 NOP
x3018 0000000000000000 x0000 NOP
x3019 0000000000000000 x0000 NOP
x301A 0000000000000000 x0000 NOP
x301B 0000000000000000 x0000 NOP
x301C 0000000000000000 x0000 NOP
x301D 0000000000000000 x0000 NOP
x301E 0000000000000000 x0000 NOP
x301F 0000000000000000 x0000 NOP
test1.obj 1247163 instructions executed Idle
```

Problem 2

Write a LC3 program which compares two numbers in memory locations 0x301D and 0x301E and puts the smaller number in memory location 0x301F. For the screenshot, make the value in memory location x301D to be x0027 and the value in memory location x301E to be x0036. Your screenshot should show memory location x301F.

The screenshot shows the LC3 Simulator interface with the following state:

Register	Value	PC	IR	PSR	CC
R0	x7FFF 32767	x300B	x3214 12820	x8001 -32767	P
R1	x0027 39				
R2	x0000 0				
R3	x0027 39				
R4	x0036 54				
R5	xFFF1 -15				
R6	x0000 0				
R7	xFD75 -651				

Address	Binary	Hex	Instruction	Comment
x3000	0010011000011100	x261C	LD R3, val1	
x3001	0010100000011100	x281C	LD R4, val2	
x3002	1001101100111111	x9B3F	NOT R5, R4	
x3003	0001101101100001	x1B61	ADD R5, R5, #1	
x3004	0001101101000011	x1B43	ADD R5, R5, R3	
x3005	0000100000000011	x0803	BRN r4smaller	
x3006	0101001100111111	x533F	AND R1, R4, #-1	
x3007	0011001000010111	x3217	ST R1, result	
x3008	111100000100101	xF025	TRAP HALT	
x3009	0101001011111111	x52FF	AND R1, R3, #-1	r4smalle
x300A	0011001000010100	x3214	ST R1, result	
x300B	111100000100101	xF025	TRAP HALT	
x300C	0000000000000000	x0000	NOP	
x300D	0000000000000000	x0000	NOP	
x300E	0000000000000000	x0000	NOP	
x300F	0000000000000000	x0000	NOP	
x3010	0000000000000000	x0000	NOP	
x3011	0000000000000000	x0000	NOP	
x3012	0000000000000000	x0000	NOP	
x3013	0000000000000000	x0000	NOP	
x3014	0000000000000000	x0000	NOP	
x3015	0000000000000000	x0000	NOP	
x3016	0000000000000000	x0000	NOP	
x3017	0000000000000000	x0000	NOP	
x3018	0000000000000000	x0000	NOP	
x3019	0000000000000000	x0000	NOP	
x301A	0000000000000000	x0000	NOP	
x301B	0000000000000000	x0000	NOP	
x301C	0000000000000000	x0000	NOP	
x301D	000000000100111	x0027 val1	NOP	
x301E	000000000110110	x0036 val2	NOP	
x301F	000000000100111	x0027 result	NOP	

test2.obj | 1248881 instructions executed | Idle

Problem 3

Write a LC3 program which multiplies two numbers in memory locations 0x301C and 0x301D and puts the result in memory location 0x301E. For the screenshot, make the value in memory location x301C to be x0012 and the value in memory location x301D to be x0020. Your screenshot should show memory location x301E.

```
LC3 Simulator - test3.obj
File  Execute  Simulate  Help
Jump to: x3000

R0  x7FFF  32767   R4  x0000   0       PC  x3007  12295
R1  x0240   576       R5  xFFF1  -15     IR  x3217  12823
R2  x0000   0         R6  x0000   0       PSR x8002  -32766
R3  x0012   18         R7  xFD75  -651    CC  Z

x3000  0010011000011011  x261B      LD  R3, val1
x3001  0010100000011011  x281B      LD  R4, val2
x3002  0101001001100000  x5260      AND R1, R1, #0
x3003  0001001001000011  x1243      ADD R1, R1, R3
x3004  0001100100111111  x193F      ADD R4, R4, #-1
x3005  0000001111111101  x03FD      BRP loop
x3006  0011001000010111  x3217      ST  R1, result
x3007  1111000000100101  xF025      TRAP HALT
x3008  0000000000000000  x0000      NOP
x3009  0000000000000000  x0000      NOP
x300A  0000000000000000  x0000      NOP
x300B  0000000000000000  x0000      NOP
x300C  0000000000000000  x0000      NOP
x300D  0000000000000000  x0000      NOP
x300E  0000000000000000  x0000      NOP
x300F  0000000000000000  x0000      NOP
x3010  0000000000000000  x0000      NOP
x3011  0000000000000000  x0000      NOP
x3012  0000000000000000  x0000      NOP
x3013  0000000000000000  x0000      NOP
x3014  0000000000000000  x0000      NOP
x3015  0000000000000000  x0000      NOP
x3016  0000000000000000  x0000      NOP
x3017  0000000000000000  x0000      NOP
x3018  0000000000000000  x0000      NOP
x3019  0000000000000000  x0000      NOP
x301A  0000000000000000  x0000      NOP
x301B  0000000000000000  x0000      NOP
x301C  000000000010010    x0012  val1  NOP
x301D  000000000100000    x0020  val2  NOP
x301E  000001001000000    x0240  result BRP  x305F

test3.obj 1249816 instructions executed Idle
```

Problem 4

Assume R1 has the value 0 initially. R4 has the value 2. Say what will be in the register R1 after the execution of the following program.

(i) Program 1

0001001001101010

0001100100111111

0000001111111101

1111000000100101

Answer: 20

(ii) Program 2

0001001001101010

0001100100111111

0000101111111101

1111000000100101

Answer: 20

(iii) Program 3

0001001001101010

0001100100111111

0000011111111101

1111000000100101

Answer: 30

(iv) Program 4

0001001001101010

0001100100111111

0000111111111101

1111000000100101

Answer: Infinite Loop

Problem 5

- i. Do all operate instructions require two source operands? If not, mention the instruction(s) that require(s) one source operand.
- ii. Given instructions AND, LD, STI identify whether the instructions are operate instructions, data movement instructions or control instructions. For each instruction list the addressing modes that can be used with the instruction.

Solution:

- i. NOT instruction is the only operate instruction that performs a unary operation i.e. it requires one source operand

AND: operate instruction

- register addressing for destination and source 1
- register or immediate addressing for source 2

LD: data movement instruction

- PC relative addressing

STI: data movement instruction

- Indirect addressing

Problem 6

Debugged code with corrections highlighted.

.orig x3000

AND R0,R0,#0

AND R1,R1,#0

AND R2,R2,#0

AND R3,R3,#0

AND R4,R4,#0

AND R5,R5,#0

AND R6,R6,#0

AND R7,R7,#0

ADD R0,R0,#9

O_loop

AND R1,R1,#0

ADD R1,R1,~~#9~~ #8

LD R7,val

ADD R0,R0,#-1

BRZ end

I_loop

LDR R3,R7,#0

LDR R4,R7,#1

ADD R6,R4,#0

NOT R6,R6

ADD R6,R6,#1

ADD R5,R3,R6

~~*BRN BRP swap*~~

ADD R7,R7,#1

ADD R1,R1,#-1

BRZ ~~I_loop~~ O_loop

BRP ~~O_loop~~ I_loop

swap

STR R3,R7,#1

STR R4,R7,#0

ADD R7,R7,#1

ADD R1,R1,#-1

BRZ O_loop

BRP I_loop

end

HALT

val

.fill x3030

.end

LC3 Simulator - input.obj

File Execute Simulate Help

Jump to: x3000

R0	x0000	0	R4	x0096	150	PC	x301F	12319
R1	x0008	8	R5	xFFFF1	-15	IR	x0411	1041
R2	x0000	0	R6	xFF6A	-150	PSR	x8002	-32766
R3	x0087	135	R7	x3030	12336	CC	Z	

```

x3000 0101000000100000 x5020 AND R0, R0, #0
x3001 0101001001100000 x5260 AND R1, R1, #0
x3002 0101010010100000 x54A0 AND R2, R2, #0
x3003 0101011011100000 x56E0 AND R3, R3, #0
x3004 0101100100100000 x5920 AND R4, R4, #0
x3005 0101101101100000 x5B60 AND R5, R5, #0
x3006 0101110110100000 x5DA0 AND R6, R6, #0
x3007 0101111111100000 x5FE0 AND R7, R7, #0
x3008 0001000000101001 x1029 ADD R0, R0, #9
x3009 0101001001100000 x5260 AND R1, R1, #0
x300A 0001001001101000 x1268 ADD R1, R1, #8
x300B 0010111000010100 x2E14 LD R7, x3020
x300C 0001000000111111 x103F ADD R0, R0, #-1
x300D 0000010000010001 x0411 BRZ x301F
x300E 0110011111000000 x67C0 LDR R3, R7, #0
x300F 0110100111000001 x69C1 LDR R4, R7, #1
x3010 0001110100100000 x1D20 ADD R6, R4, #0
x3011 1001110110111111 x9DBF NOT R6, R6
x3012 0001110110100001 x1DA1 ADD R6, R6, #1
x3013 0001101011000110 x1AC6 ADD R5, R3, R6
x3014 0000001000000100 x0204 BRP x3019
x3015 0001111111100001 x1FE1 ADD R7, R7, #1
x3016 0001001001111111 x127F ADD R1, R1, #-1
x3017 0000010111110001 x05F1 BRZ x3009
x3018 0000001111110101 x03F5 BRP x300E
x3019 0111011111000001 x77C1 STR R3, R7, #1
x301A 0111100111000000 x79C0 STR R4, R7, #0
x301B 0001111111100001 x1FE1 ADD R7, R7, #1
x301C 0001001001111111 x127F ADD R1, R1, #-1
x301D 0000010111101011 x05EB BRZ x3009
x301E 0000001111101111 x03EF BRP x300E
x301F 1111000000100101 xF025 TRAP HALT
x3020 0011000000110000 x3030 ST R0, x3051
x3021 0000000000000000 x0000 NOP

```

input.obj 2490 instructions executed Idle