Generals:

16-bit addresses

64k memory locations (16-bit word @ each loc.)

2’sC integers, 8 data reisters (3bits to name reg)

3 condition code bits, 4bit opcodes(16 intsructs)

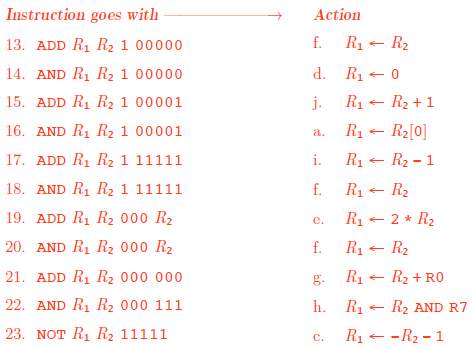
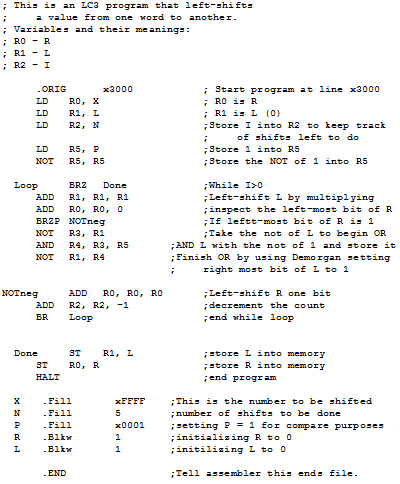
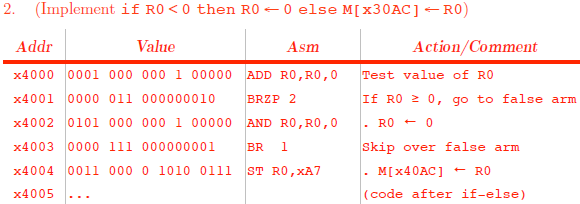
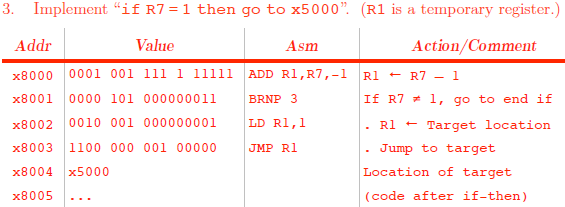
Words and addresses are totally unrelated

5 ways to specify operand (addressing modes):

Immediate: contained in instruction

Register: number -> 000, 001, …, 110, 111

3ways to specify mem locs: Base-offset, PC-offset, and Indirect.



Differences from SDC:

Address size, Word size, radix, #of Register, CondCode, #of opcodes, SDC uses Absolute addr. (while addr is part of instruct.)

LC3 has 16bit addr and 16bit instruc

3kinds of instructions:

Data movement: Load (value into register)

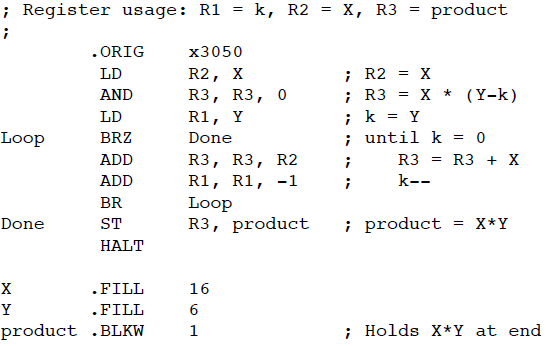
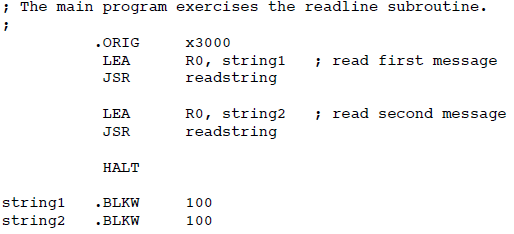
Store (value from register)

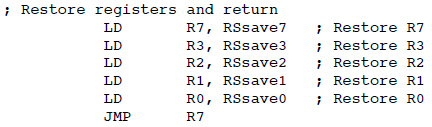
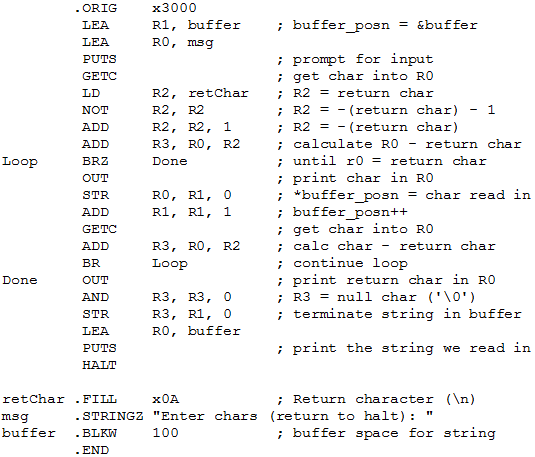
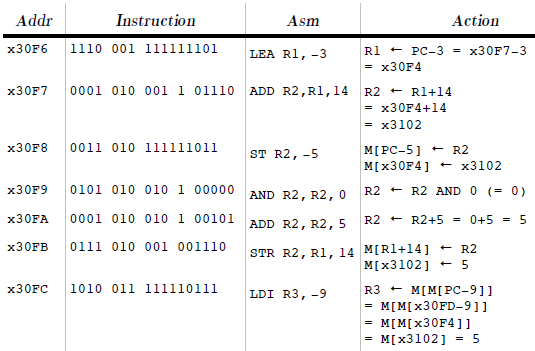
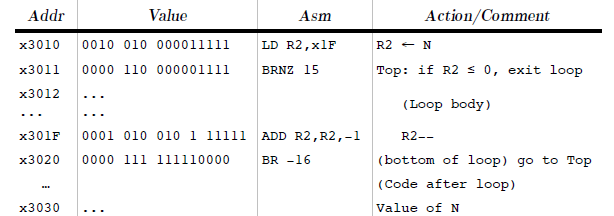
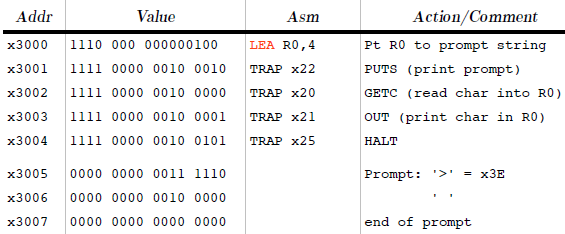
Calculation: ADD

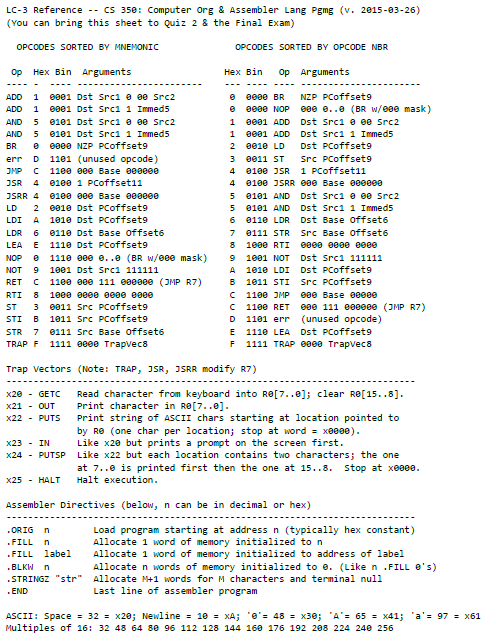
Control: Branch, Jump – modify the PC during execute instruction phase of cycle (otherwise PC is incremented during Fetch instuc., after reading instruct. From memory) – (IR – decodes instruction)

Instruc. Cycle for all LC3 instructs: (not orthog)

Fetch instruction 🡪 Decode Instruction 🡪 Evaluate instruction 🡪 Fetch op from Mem 🡪 Execute operation 🡪 Store result.







PCoffset9 range: -256 ≤ PC ≤ 255

ADD & AND Immed5 range -16 ≤ val ≤ 15

CC: N100 or Z010 or P001

LD: R# <- M[PC+offset]

ST: M[PC+offset] <- R#

LEA: R# <- PC+offset

LDR: R# <- M[RB#+offset]

STR: M[RB#+offset] <- R#

LDI: R# <- M[M[PC+offset]]

STI: M[M[PC+offset]] <- R#

JMP: PC <- RB#

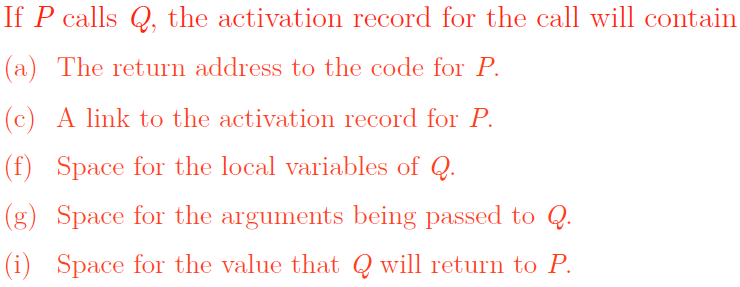
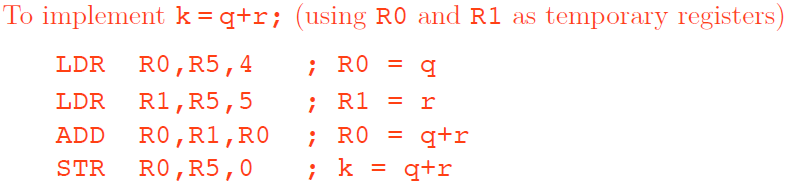
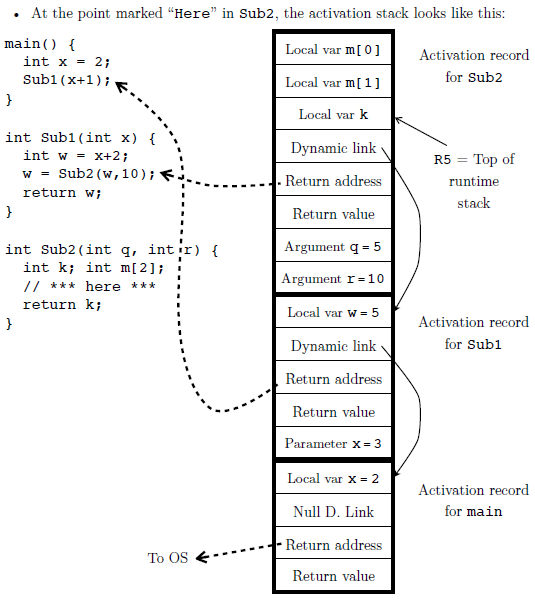
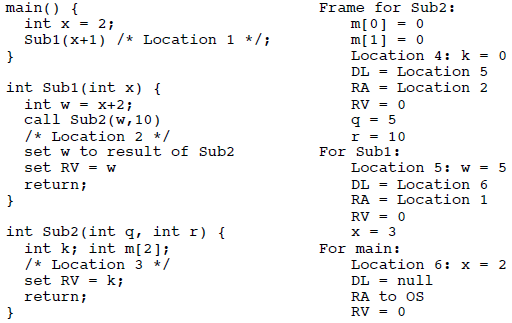
BR: if (CC&Mask≠000) then PC <- PC+offset

TRAP: R7 <- PC <- M[TrapVec8]

JSR: R7 <- PC; PC <- PC+Sext(PCoffset11)

JSRR: target(goto)<- RB#; R7 <- PC; PC <- target

RET = JMP R7



Directives begin with period

Only Labels are case sensitive

Unsupported Operations:

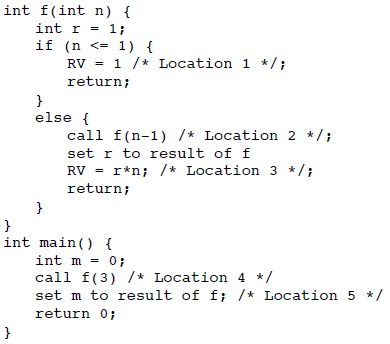
Subtraction: for X-Y 🡪 use ADD like X+NOT Y+1

OR: for X OR Y 🡪 NOT(NOT X AND NOT Y)

Setting R to 0: AND with 0 🡪 AND R1, R1, 0

Copy R to R: 3 ways: ADD R0, R1, 0;

AND R0, R1, R1; AND R0, R1, xFFFF



======== Problem 2 ========

[Note: this is the non-tail-recursive solution]

At Location 1:

For g:

Location 9: DL = Location 8

RA = Location 2

RV = 6

n = 1

r = 6

For g:

Location 8: DL = Location 7

RA = Location 2

RV = 0

n = 2

r = 3

For g:

Location 7: DL = Location 6

RA = Location 4

RV = 0

n = 3

r = 1

For main:

Location 6: m = 0;

DL = null

RA to OS

RV = 0

--------------------

At Location 3:

For g:

Location 8: DL = Location 7

RA = Location 2

RV = 6

n = 2

r = 3

For g:

Location 7: DL = Location 6

RA = Location 4

RV = 0

n = 3

r = 1

For main:

Location 6: m = 0;

DL = null

RA to OS

RV = 0

--------------------

At Location 3:

For g:

Location 7: DL = Location 6

RA = Location 4

RV = 6

n = 3

r = 1

For main:

Location 6: m = 0;

DL = null

RA to OS

RV = 0

--------------------

At Location 5:

For main:

Location 6: m = 6;

DL = null

RA to OS

RV = 0

======== Problem 1 =========

At location 1:

For f:

Location 9: r = 0

DL = Location 8

RA = Location 2

RV = 1

n = 1

For f:

Location 8: r = 0

DL = Location 7

RA = Location 2

RV = 0

n = 2

For f:

Location 7: r = 0

DL = Location 6

RA = Location 4

RV = 0

n = 3

For main:

Location 6: m = 0;

DL = null

RA to OS

RV = 0

--------------------

At location 3:

For f:

Location 8: r = 1

DL = Location 7

RA = Location 2

RV = 2

n = 2

For f:

Location 7: r = 0

DL = Location 6

RA = Location 4

RV = 0

n = 3

For main:

Location 6: m = 0;

DL = null

RA to OS

RV = 0

--------------------

At location 3:

For f:

Location 7: r = 2

DL = Location 6

RA = Location 4

RV = 6

n = 3

For main:

Location 6: m = 0;

DL = null

RA to OS

RV = 0

--------------------

At location 5:

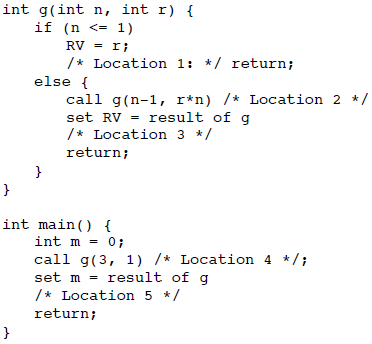
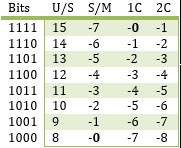
For main:

Location 6: m = 6;

DL = null

RA to OS

RV = 0



**Conversions From:**

S/M: Replace leftmost with a sign.

1’sC: **Flip** -7(1000) to get 7(0111).

2’sC: **Flip except last 1 or 0**

-3(1101) to 3(0011).

**Conversions To:**

S/M: Leftmost used for sign. Has (±0).

1’sC: **Flip** 7(0111) to get -7(1000). Has (±0).

2’sC: **Flip+1** 7(0111) to (1000+1)= -7(1001).

Has more neg. than pos.

;////~~~ Location 1 ~~~////

For g:

DL = Location 20

RA = Location 4

RV = 6

n = 1

r = 6

For main:

Location 20: m = 0

DL = null

RA to OS

RV = 0

;////~~~ Location 3 ~~~////

; -- not accessed because of tail recursion --

; ignores Location 3 instruction

;////~~~ Location 5 ~~~////

For main:

Location 20: m = 6

DL = null

RA to OS

RV = 0