

## LC3 Quick Reference Guide

Instruction Set			
Op	Format	Description	Example
ADD	ADD DR, SR1, SR2 ADD DR, SR1, imm5	Adds the values in SR1 and SR2/imm5 and sets DR to that value.	ADD R1, R2, #5 The value 5 is added to the value in R2 and stored in R1.
AND	AND DR, SR1, SR2 AND DR, SR1, imm5	Performs a bitwise and on the values in SR1 and SR2/imm5 and sets DR to the result.	AND R0, R1, R2 A bitwise and is performed on the values in R1 and R2 and the result stored in R0.
BR	BR(n/z/p) LABEL Note: (n/z/p) means any combination of those letters can appear there, but must be in that order.	Branch to the code section indicated by LABEL, if the bit indicated by (n/z/p) has been set by a previous instruction. n: negative bit, z: zero bit, p: positive bit. Note that some instructions do not set condition codes bits.	BRz LPBODY Branch to LPBODY if the last instruction that modified the condition codes resulted in zero. BRnp ALT1 Branch to ALT1 if last instruction that modified the condition codes resulted in a positive or negative (non-zero) number.
JMP	JMP SR1	Unconditionally jump to the instruction based upon the address in SR1.	JMP R1 Jump to the code indicated by the address in R1.
JSR	JSR LABEL	Put the address of the next instruction after the JSR instruction into R7 and jump to the subroutine indicated by LABEL.	JSR POP Store the address of the next instruction into R7 and jump to the subroutine POP.
JSSR	JSSR SR1	Similar to JSR except the address stored in SR1 is used instead of using a LABEL.	JSSR R3 Store the address of the next instruction into R7 and jump to the subroutine indicated by R3's value.
LD	LD DR, LABEL	Load the value indicated by LABEL into the DR register.	LD R2, VAR1 Load the value at VAR1 into R2.
LDI	LDI DR, LABEL	Load the value indicated by the address at LABEL's memory location into the DR register.	LDI R3, ADDR1 Suppose ADDR1 points to a memory location with the value x3100. Suppose also that memory location x3100 has the value 8. 8 then would be loaded into R3.
LDR	LDR DR, SR1, offset6	Load the value from the memory location found by adding the value of SR1 to offset6 into DR.	LDR R3, R4, #-2 Load the value found at the address (R4 -2) into R3.
LEA	LEA DR, LABEL	Load the address of LABEL into DR.	LEA R1, DATA1 Load the address of DATA1 into R1.
NOT	NOT DR, SR1	Performs a bitwise not on SR1 and stores the result in DR.	NOT R0, R1 A bitwise not is performed on R1 and the result is stored in R0.
RET	RET	Return from a subroutine using the value in R7 as the base address.	RET Equivalent to JMP R7.

RTI	RTI	Return from an interrupt to the code that was interrupted. The address to return to is obtained by popping it off the supervisor stack, which is automatically done by RTI.	RTI Note: RTI can only be used if the processor is in supervisor mode.
ST	ST SR1, LABEL	Store the value in SR1 into the memory location indicated by LABEL.	ST R1, VAR3 Store R1's value into the memory location of VAR3.
STI	STI SR1, LABEL	Store the value in SR1 into the memory location indicated by the value that LABEL's memory location contains.	STI R2, ADDR2 Suppose ADDR2's memory location contains the value x3101. R2's value would then be stored into memory location x3101.
STR	STR SR1, SR2, offset6	The value in SR1 is stored in the memory location found by adding SR2 and offset6 together.	STR R2, R1, #4 The value of R2 is stored in memory location (R1 + 4).
TRAP	TRAP trapvector8	Performs the trap service specified by trapvector8. Each trapvector8 service has its own assembly instruction that can replace the trap instruction.	TRAP x25 Calls a trap service to end the program. The assembly instruction HALT can also be used to replace TRAP x25.

## Symbol Legend

Symbol	Description	Symbol	Description
SR1, SR2	Source registers used by instruction.	LABEL	Label used by instruction.
DR	Destination register that will hold the instruction's result.	trapvector8	8 bit value that specifies trap service routine.
imm5	Immediate value with the size of 5 bits.	offset6	Offset value with the size of 6 bits.

## TRAP Routines

Trap Vector	Equivalent Assembly Instruction	Description
x20	GETC	Read one input character from the keyboard and store it into R0 without echoing the character to the console.
x21	OUT	Output character in R0 to the console.
x22	PUTS	Output null terminating string to the console starting at address contained in R0.
x23	IN	Read one input character from the keyboard and store it into R0 and echo the character to the console.
x24	PUTSP	Same as PUTS except that it outputs null terminated strings with two ASCII characters packed into a single memory location, with the low 8 bits outputted first then the high 8 bits.
x25	HALT	Ends a user's program.

## Pseudo-ops

Pseudo-op	Format	Description
.ORIG	.ORIG #	Tells the LC-3 simulator where it should place the segment of code starting at address #.
.FILL	.FILL #	Place value # at that code line.
.BLKW	.BLKW #	Reserve # memory locations for data at that line of code.
.STRINGZ	.STRINGZ "<String>"	Place a null terminating string <String> starting at that location.
.END	.END	Tells the LC-3 assembler to stop assembling your code.