# LC-3 Programmer's Reference

The PennSim Quick Start Guide appears on the back of this page

**Assembler Directives**: **.ORIG .END .FILL .BLKW .STRINGZ**

**PC⁺:** incremented **PC**
**mem[A]:** memory contents at address **A**

**SEXT(value):** sign-extend **value** to 16 bits
**ZEXT(value):** zero-extend **value** to 16 bits

**setcc():** set condition codes (**NZP**)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | DR | | | SR1 | | | 0 | 0 | 0 | SR2 | | | **ADD DR, SR1, SR2** ; **Addition**<br>*DR ← SR1 + SR2, setcc()* |
| 0 | 0 | 0 | 1 | DR | | | SR1 | | | 1 | imm5 | | | | | **ADD DR, SR1, imm5** ; **Addition with immediate**<br>*DR ← SR1 + SEXT(imm5), setcc()* |
| 0 | 1 | 0 | 1 | DR | | | SR1 | | | 0 | 0 | 0 | SR2 | | | **AND DR, SR1, SR2** ; **Bitwise AND**<br>*DR ← SR1 AND SR2, setcc()* |
| 0 | 1 | 0 | 1 | DR | | | SR1 | | | 1 | imm5 | | | | | **AND DR, SR1, imm5** ; **Bitwise AND with immediate**<br>*DR ← SR1 AND SEXT(imm5), setcc()* |
| 1 | 0 | 0 | 1 | DR | | | SR | | | 1 | 1 | 1 | 1 | 1 | 1 | **NOT DR, SR** ; **Bitwise complement**<br>*DR ← NOT(SR), setcc()* |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | DR | | | PCoffset9 | | | | | | | | | **LD DR, label** ; **Load from mem (PC-relative)**<br>*DR ← mem[PC⁺ + SEXT(PCoffset9)], setcc()* |
| 1 | 0 | 1 | 0 | DR | | | PCoffset9 | | | | | | | | | **LDI DR, label** ; **Load from mem (indirect)**<br>*DR ← mem[ mem[PC⁺ + SEXT(PCoffset9)] ], setcc()* |
| 0 | 1 | 1 | 0 | DR | | | BaseR | | | offset6 | | | | | | **LDR DR, BaseR, offset6** ; **Load from mem (base+offset)**<br>*DR ← mem[BaseR + SEXT(offset6)], setcc()* |
| 1 | 1 | 1 | 0 | DR | | | PCoffset9 | | | | | | | | | **LEA DR, label** ; **Put label address into reg**<br>*DR ← PC⁺ + SEXT(PCoffset9), setcc()* |
| 0 | 0 | 1 | 1 | SR | | | PCoffset9 | | | | | | | | | **ST SR, label** ; **Store to mem (PC-relative)**<br>*mem[PC⁺ + SEXT(PCoffset9)] ← SR* |
| 1 | 0 | 1 | 1 | SR | | | PCoffset9 | | | | | | | | | **STI SR, label** ; **Store to mem (indirect)**<br>*mem[ mem[PC⁺ + SEXT(PCoffset9)] ] ← SR* |
| 0 | 1 | 1 | 1 | SR | | | BaseR | | | offset6 | | | | | | **STR SR, BaseR, offset6** ; **Store to mem (base+offset)**<br>*mem[BaseR + SEXT(offset6)] ← SR* |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | n | z | p | PCoffset9 | | | | | | | | | **BRx label** ; **Conditional branch to label**<br>; **x is condition ∈{n,z,p,zp,np,nz,nzp}**<br>*if ( (n AND N) OR (z AND Z) OR (p AND P) )*<br>*then PC ← PC⁺ + SEXT(PCoffset9)* |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | BaseR | | | 0 | 0 | 0 | 0 | 0 | 0 | **JMP BaseR** ; **Jump to address in register**<br>*PC ← BaseR* |
| 0 | 1 | 0 | 0 | 1 | PCoffset11 | | | | | | | | | | | **JSR label** ; **Call subroutine at label**<br>*R7 ← PC⁺, PC ← PC⁺ + SEXT(PCoffset11)* |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | BaseR | | | 0 | 0 | 0 | 0 | 0 | 0 | **JSRR BaseR** ; **Call subroutine at address in reg**<br>*TEMP = PC⁺*<br>*PC ← BaseR, R7 ← TEMP* |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | **RET** ; **Return from subroutine**<br>*PC ← R7* |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **RTI** ; **Return from interrupt**<br>*Not used in ECE 252* |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | trapvect8 | | | | | | | | **TRAP trapvect8** ; **System call**<br>*R7 ← PC⁺, PC ← mem[ZEXT(trapvect8)]* |

**Values for trapvect8:** **GETC=x20 OUT=x21 PUTS=x22 IN=x23 PUTSP=x24 HALT=x25** (table of TRAP aliases on reverse)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | | | | | | | | | | | | | ; unused opcode |

# PennSim Quick Start Guide

The following steps will get you started loading an LC3 assembly program in PennSim. Each step includes two pieces of information:

> **the command you should type in the PennSim window**
> *the output you should expect to see in the PennSim window if it worked correctly.*

STEP 1: Reset simulator
> **reset**
> *System Reset.*

STEP 2: Assemble your program(s) (do this once per source code file for your program)
> **as my_program_name.asm**
> *Assembly of 'my_program_name.asm' completed without errors or warnings.*

STEP 3: Load your program(s) (do this once per object code file for your program)
Use **File→Open .obj File** and choose the file (e.g., **my_program_name.obj**)
> *Loaded binary object file 'my_program_name.obj'*
> *Loaded symbol file 'my_program_name.sym'*

STEP 4: Scroll the memory window directly to address x0200
> **list x0200**
> *x0200 : (data at address x0200 in hexadecimal) : (disassembled instruction at address x02000)*

***** ALSO FOLLOW BELOW STEPS IF YOU ARE USING THE LC-3 OS *****

STEP 4: Assemble OS (if you have not already—if `lc3os.obj` is present, you can skip this)
> **as lc3os.asm**
> *Assembly of 'lc3os.asm' completed without errors or warnings.*

STEP 5: Load OS (needs to be done every time after reset or starting simulator)
Use **File→Open .obj File** and choose **lc3os.obj**
> *Loaded binary object file 'lc3os.obj'*
> *Loaded symbol file 'lc3os.sym'*

STEP 6: Set a breakpoint at first memory address of your program (x3000)
> **break set x3000**
> *Breakpoint set at x3000*

STEP 7: Execute to the first breakpoint encountered (x3000 if breakpoint set as above)
> **continue**

Pay careful attention to the output you get, and make sure it matches the expected output for the command. If you don't see the expected output for any of these steps, **stop** and fix the problem before going to the next step. If you need to contact an instructor for help, make note of which step gave you different output than expected and what the actual output message you got was.

For more details on what to do after you have loaded your program, see the full PennSim tutorial and guide.

## LC-3 Memory-Mapped I/O

| Name | Address | Description |
|---|---|---|
| **KBSR**<br>Keyboard Status Register | **xFE00** | **KBSR[15]** is **1** when a new character is available. **KBSR[15]** is cleared by reading **KBDR**. |
| **KBDR**<br>Keyboard Data Register | **xFE02** | **KBDR[7:0]** has most recently-typed ASCII character. **KBDR[15:8]** are **0**. |
| **DSR**<br>Display Status Register | **xFE04** | **DSR[15]** is **1** when display is ready for a new character. **DSR[15]** is **0** while display is not ready. |
| **DDR**<br>Display Data Register | **xFE06** | **DDR[7:0]** is ASCII character to send to console display. **DDR[15:8]** must be **0**. |

## LC-3 TRAPs (require the OS!)

*Note: all TRAPs alter register **R7***

| Instruction | Alias | Description |
|---|---|---|
| **TRAP x20** | **GETC** | Waits for a character from keyboard. ASCII code returned in **R0[7:0]** but not echoed to console. |
| **TRAP x21** | **OUT** | Writes character in **R0[7:0]** to console display. **R0[15:8]** must be **0**. |
| **TRAP x22** | **PUTS** | Writes null-terminated ASCII string to console. String is one character per memory location (bits **[7:0]**), starting at address specified by **R0**. |
| **TRAP x23** | **IN** | Displays "`Input a character>`", then waits for a character from keyboard. ASCII code returned in **R0[7:0]** and echoed to console. |
| **TRAP x24** | **PUTSP** | *Not used in ECE 252.* |
| **TRAP x25** | **HALT** | Halts execution and prints console message. |