Reading a character from Keyboard:

```
.ORIG   x3000
START   LDI     R1, KBSR
BRzp    START
LDI     R0, KBDR
OUT
HALT
KBSR    .FILL   xFE00
KBDR    .FILL   xFE02
.END
```

Outputting a character to Display/Console:

```
.ORIG   x3000
START   LD      R0, A
LOOP    LDI     R1, DSR
BRzp    LOOP
STI     R0, DDR
HALT
DSR     .FILL   xFE04
DDR     .FILL   xFE06
A       .FILL   65
.END
```

Device Drivers do not have direct user control. They run in Super User mode/Privileged mode.

**Solution:** Let the OS provide the I/O functionality. 1-C-3 does this using TRAP instructions.
The TRAP mechanism

1. Service call for using the TRAP routine
2. TRAP Service Routines
3. TRAP Vector Table in protected area of memory aka Service Control Block
4. A return from TRAP mechanism (RTI instruction in LC-3)

The TRAP mechanism

1. Service call for using the TRAP routine
2. TRAP Service Routines
3. TRAP Vector Table in protected area of memory aka Service Control Block
4. A return from TRAP mechanism (RTI instruction in LC-3)

20 I/O Service Routines - TRAPs

<table>
<thead>
<tr>
<th>Trap Vector</th>
<th>Assembler Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x21</td>
<td>OUT</td>
<td>Read a single character from the keyboard. The character is not echoed onto the console. Its ASCII code is copied into R0. The high eight bits of R0 are cleared.</td>
</tr>
<tr>
<td>x22</td>
<td>PUTS</td>
<td>Write a string of ASCII characters to the console display. The characters are contained in consecutive memory locations, one character per memory location, starting with the address specified in R0. Writing terminates with the occurrence of x0000 in a memory location.</td>
</tr>
<tr>
<td>x23</td>
<td>IN</td>
<td>Print a prompt on the screen and read a single character from the keyboard. The character is echoed onto the console monitor, and its ASCII code is copied into R0. The high eight bits of R0 are cleared. Write a string of ASCII characters to the console. The characters are contained in consecutive memory locations, two characters per memory location, starting with the address specified in R0. The ASCII code contained in bits [7:0] of a memory location is written to the console first. Then the ASCII code contained in bits [15:8] of that memory location is written to the console. (A character string consisting of an odd number of characters to be written will have x00 in bits [16:8] of the memory location containing the last character to be written.) Writing terminates with the occurrence of x0000 in a memory location. Hall execution and print a message on the console.</td>
</tr>
<tr>
<td>x25</td>
<td>HALT</td>
<td></td>
</tr>
<tr>
<td>x26</td>
<td>PRINTF</td>
<td></td>
</tr>
</tbody>
</table>

Content to McGraw-Hill Education. Permission required for reproduction or display.

Table A.3  Trap Service Routines

Copyright © McGraw-Hill Education. Permission required for reproduction or display.

User Program

Program for handling the I/O request

Operation system service routine

O/S module for doing I/O

BIOS

basic I/O system
20 TRAP & RTI Instructions
Friday, April 30, 2021  9:30 AM

1. If in user mode (PSR[$15] = 1)
   1. Save RC (USP) → Save USP
   2. Restore RC to point to SYSP
   3. Save PSR on system stack
   4. Save PC on system stack
   5. Set PSR[$15] = 0 [Supervisor Mode]
   6. PC ← M[S EXT (Trapvector)]

run next instruction pointed by PC

2. RTI

1. Restore PSR from system stack
2. PC ← from system stack.
3. Reset mode to user/supervisor by looking at PSR[$15]

I/O in LC-3 is done
using Polling
"Are we there yet?"

Alternative
Interrupt Direct I/O
Program A is executing instruction n
Program A is executing instruction n+1
Program A is executing instruction n+2

1: Interrupt signal is detected
1: Program A is put into suspended animation
1: PC is loaded with the starting address of Program B
2: Program B starts satisfying I/O device's needs
2: Program B continues satisfying I/O device's needs
2: Program B continues satisfying I/O device's needs
2: Program B finishes satisfying I/O device's needs
3: Program A is brought back to life

Program A is executing instruction n+3
Program A is executing instruction n+4