Example

* Problem statement:
  Given a sequence of 100 numbers stored in memory starting at $x4000$, count the negative values

* Flowchart:
* Program:

Start

Initialization

Get number

Negative?  No  Yes

Count number

Move to next number

Done?  No  Yes

Stop

Register usage:
R0: address of current number
R1: numbers yet to be checked
R2: negative numbers found
R5: value of current number
count100.asm

; Counter of negative numbers in a sequence
;
; Given a sequence of 100 values stored in 2's complement format,
; starting at x4000, count the number of negative values.
;
; Register usage:
; R0 address of current number
; R1 numbers yet to be checked
; R2 negative numbers found
; R3
; R4
; R5 temp register, used to setcc
; R6
; R7
;
; Start
.ORIG x3000
;
; Initialization
LD R0, FIRST
LD R1, TOTAL
AND R2, R2, #0
;
; Get number
LOOP  LDR R5, R0, #0
;
; Negative?
BRzp SKIP
;
; Count number
ADD R2, R2, #1
;
; Move to next number
SKIP  ADD R0, R0, #1
;
; Done?
ADD R1, R1, #-1
BRp LOOP
;
; Stop
HALT
;
; Symbol table

count100.sym

// Symbol table
// Scope level 0:
// Symbol Name   Page Address
// -------------- ------------
// LOOP          3003
// SKIP          3006
// FIRST         300A
// TOTAL         300B

©2017 Juan Jose Jaramillo. All rights reserved.

Programming in LC-5 assembly language Page 5
`; Upper
ADD R1, R1, #\-1
BRp LOOP
; Stop
HALT
;
FIRST .FILL x4000
TOTAL .FILL #100
.END`
LC-3 TRAP mechanism

* I/O operations require specialized knowledge, a mistake could affect lots of users
* Service routines are therefore provided by the Operative System (OS) to safely and conveniently perform low-level, privileged operations
* In LC-3, service routines implemented in TRAP mechanism
* Elements of TRAP mechanism:

1) Set of service routines: executed on behalf of user program by OS. Up to 256 in LC-3

2) Table of starting addresses: for routines stored in memory from x0000 to x00FF

3) TRAP instruction: way for user to call routine

4) Linkage: mechanism for returning control back to user program
* TRAP instruction:

```
1111 0000 0010 0101
```

Opcodex

* Trap vector: zero-extend to identify index in table of starting addresses (trap vector table)
Complete mechanism when invoking TRAP instruction:

\[
\begin{align*}
\text{MAR} &\leftarrow \text{ZEXT} \ (\text{trapvect } 8) \\
\text{MDR} &\leftarrow M \ [\text{MAR}] \\
R7 &\leftarrow \text{PC} \\
\text{PC} &\leftarrow \text{MDR}
\end{align*}
\]

(Service routine gets executed)

\[
\text{JMP R7}
\]

Mnemonic for JMP R7: RET (not really a new instruction)
* On saving/restoring registers before/after executing service routines

1) Caller-save:
   Calling program saves all necessary registers before calling service routine

2) Callee-save:
   Called program (service routine) saves all registers before executing routine
* TRAP vectors:

HALT (x25): print message on display, then stop execution

IN (x23): get character from keyboard, echo to display

GETC (x20): get character from keyboard, don't echo to display

OUT (x21): write character to display

POTS (x22): write null-terminated string to display

PUTSP (x24): similar to POTS but get two characters per memory location