Example

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

* Flowchart:
* Program:

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

- Start
- Initialization
- Get number
  - Negative?
    - No: Count number
    - Yes: Move to next number
  - Done?
    - No: Move to next number
    - Yes: Stop

© 2017-2018 Juan Jose Jaramillo. All rights reserved.
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
;
FIRST .FILL x4000
TOTAL .FILL #100
.END

count100.sym

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

© 2017-2018 Juan Jose Jaramillo. All rights reserved.
LC-3 TRAP mechanism

* I/O operations require specialized knowledge, a mistake could affect lots of users
* Service routines are therefore provided by the Operating 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
```

Opcode

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

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

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