Comp 255 – Traps/Interrupts Coding Examples

ARC:

! A simple ARC program to demonstrate traps!

.begin
.org 2048
main: addcc %r0, 1, %r1  ! store 1 in %r1
    subcc %r0, %r1, %r2 ! store -1 in %r2 and set n & c
    ta 0    ! trap 0
    halt
.org 0xff000800  ! address for trap 0
addcc %r2, 2, %r3 ! add 2 to %r2 and put in %r3
addcc %r0, 1, %r0 ! clear cc
rett %r16, 4   ! restore PSR; rett %r16+4 won't work
.end

The above program executes instructions to set the n and c condition codes before invoking the ta 0 instruction which saves the return address in %r16 and the program status word or psw (i.e. the condition codes) in %r18.

The trap handler begins at address 0xff000800 (this is fixed by the ARC simulator).

The trap handler executes instructions to clear the n and c condition codes but when the rett (return from trap) is execute note that the old values of the psw are restored!

PDP-8 Example:

The PDP-8 example on the following page endlessly generates the alphabet from ‘a’ to ‘z’. However when any key is hit, an interrupt is generated which rings the bell (ASCII code 07) resets the alphabet display back to the beginning. If you hit (Enter) it stops the program; hitting Enter again will restart it where it left off.

Note that if you do this enough time and you’re lucky you might notice that the initial ‘a’ in the alphabet display is missing! This is because the interrupt occurs in the part of the code that either prints the character or increments the character.

The solution to prevent this to insert iof and ion instructions to disable and re-enable the interrupt system around critical code areas where the occurrence and handling of an interrupt might cause problems.
PDP-8 Example
/
/ File: Inttest1.pal
/ Date: April 2010
/
/ Program displays alphabet continuously. When a key is hit
/ it sounds the bell and restarts the alphabet display
/ Program terminates on CR
/
/ General Interrupt Handler
/
*0000
  0       / Return Address from Interrupt
  jms i IntH  / Jump to Handler
  ion       / Turn on Interrupt System
  jmp i 0   / Return from Interrupt
IntH,   XIntH           / Address of Interrupt Handler

*5000
XIntH,  0               / Interrupt Handler
dca XHold       / Save AC
krb             / Get character
tad XConst      / Is it a CR?
sna             /   No  - continue
   /   Yes - halt
   cla cll
tad XConst+1    / Get Bell
tsfl               / Sound It
cla cll
tad XConst+2    / Get carriage Return
tsfl         / and display It
cla cll
tad XConst+3    / Get Line Feed
tsfl         / and display it
cla cll
tad A
dca CH       / reset character to a
   tad XHold   / Restore AC
   jmp i XIntH  / Return

XHold,  0
XConst, -13d; 07; 13d; 10d  / -CR, Bell, CR, LF characters
/  / Basic I/O Routine Vectors - Page 0
/  
*0050
GetChar, XGetChar
Type, XType
CRLF, XCRLF
PrtStr, XPrtStr
/
/  Global Variables
/
*0100
A, 'a'
Z, 'z'
CH, 0
/
/  Code Segment - Page 1
/
*0200
Start, cla cll         / Clear AC and Link
ksf             / Wake up keyboard
tls             / Wake up printer
ion             / Turn on Interrupt System
cla cll
  tad A           / Get 'a'
dca CH
Loop,  cla cll
  tad CH
    jms i Type      / Display letter
  cla cll
  tad Z           / check if 'z'
cia
  tad CH
  sna
    jmp Reset       / If 'z' reset sequence
   isz CH          / Else increment ASCII code
    jmp Loop
Reset,  tad A           / Get 'a'
dca CH
  jms i CRLF      / Do CR-LF
    jmp Loop
XGetChar, 0  / store return address here
  ksf    / is keyboard flag raised?
  jmp .-1    / no - loop
  krb    / yes - read character to AC
  jmp i XGetChar    / return

XType, 0  / store return address here
  tsf    / is printer ready?
  jmp .-1    / no - loop
  tls    / yes - print character
  cla cl1    / clear AC and Link
  jmp i XType    / return

XCRLF, 0  / store return address here
  cla cl1
  tad .+5    / get CR
  jms XType    / and type it
  tad .+4    / get LF
  jms XType    / and type it
  jmp i XCRLF    / return
  13d;10d;    / carriage return; line feed

XPrtStr,0
  dca 10    / store address at autoindex register 10
  tad i 10    / get character
  sna    / is it null?
  jmp i XPrtStr    / yes - return
  jms XType    / no - type it
  jmp .-4    / get next character

$Start