The Trap Instruction Subroutines

The High Level
The **Trap Instruction Subroutines** are a highly useful and relatively simple to use library of **subroutines** that allow for handling Input / Output with the user.

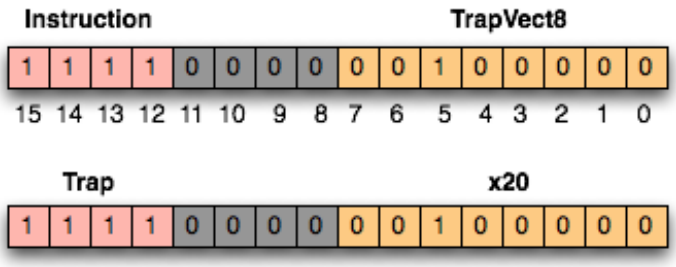

Figure 1: The Trap Instruction (details)

Trap x20 Gets a single character of input from the user and stores the ASCII code for that character in R0 (Note: The input character is NOT printed/echoed to the console).

Trap x21 Prints to the console whatever ASCII-coded value is in R0.

Trap x22 Treats R0 as an address and goes to MEM[R0]. Then, it prints every ASCII-coded character it finds from MEM[R0] until it encounters a 0.

Trap x23 Prompts the user to enter a single character of input, prints/echoes it to the console when the user types it, and then stores the ASCII code for the input character in R0 (Note: This is like a composite of Trap x20 and Trap x21),

The Breakdown:

- Bits [15-12] specify what instruction to execute (1111 is Trap)
- Bits [11-8] are unused
- Bits [7-0] specify the Trap Vector, which is a numerical code that indicates which Trap subroutine to call

The Examples!

```
.orig x3000
   ;---------------
   ; Instructions
   ;---------------
   Trap x20           ; R0 <-- ASCII[input character]
   Trap x21           ; Echo the character just typed to the console
   ADD R1, R0, #0     ; R1 <-- R0

   Trap x20           ; R0 <-- ASCII[input character]
   Trap x21           ; Echo the character just typed to the console
   ADD R2, R0, #0     ; R2 <-- R0
   HALT
.end
```

Result: The result of the above program is that two single characters of input have been obtained from the user. The ASCII code for the first input character is stored in R1 and the ASCII code for the second input character is stored in R2.

```
.orig x3000
   ;---------------
   ; Instructions
   ;---------------
   LEA R0, MSG_1
   Trap x22     ; Print the string found at MEM[R0] until a 0 is found

   HALT
   ;-------
   ; Data
   ;-------
   MSG_1     .STRINGZ     "I like hotdogs!!"
.end
```

Result:
The result of the above program is that the string "I like hotdogs!!" is printed to the console. YAY HOTDOGS!!

Pitfalls... (aka: Erroneous code makes baby Dr. Linards cry)

The only real pitfall that you can encounter with the Trap is forgetting that calling a Trap Subroutine will overwrite the value currently in R0. So if you have a value in R0 and it is important, make sure to **ST**ore it somewhere or copy it to another register for safe-keeping.

How to copy a value to another register:
As illustrated in an example above, you can copy a value from one register directly to another in a single command:

```
ADD DestReg, SourceReg, #0
```