Lecture 35: Assembly language programming: I/O

TRAP × 25   HALT
TRAP × 24   PUTSP
   × 23        IN
   × 22        PUTS
   × 21        OUT
   × 20        GETC

(i) R7 holds value of PC until control returns to the program
TRAP x20        GETC  Read one character from keyboard into R0
TRAP x21        OUT   Print character in R0 to display
TRAP x22        PUTS   Print string ending in NUL character from memory
to display; starting address in R0; each address holds 1 ASCII character
TRAP x23        IN     Print prompt "Input a character>", read character
                    into R0 and echo character to display
TRAP x24        PUTSP  Like PUTS, except characters read differently (Not
                    used in ECE120)
TRAP x25        HALT   Stops execution of program

Note: Careful with the use of R0;

Store to another register (copy)
OR
to memory (ST)
Example: Prompt the user to enter a character. Then print "The character you entered was: " and the character entered by the user on the screen.
Print the String

Print the character

HALT

• ORIG x 3000
IN

ADD R1, R0, #0

LEA R0, TEXT

Puts

ADD R0, R1, #0

OUT

HALT

TEXT .STRINGZ "The character you entered was: "

END
Example:

```
* * * * * * * *
*            *
*    ^   ^    *
*            *
* * * * * * * *
```

8 columns

8 rows

Question: how do we store a two-dimensional figure in a one-dimensional memory?
Answer: row after row, as if it were a one-dimensional array

```assembly
SPRITE
; row 0
.FILL x2A ; *
.FILL x2A ; *
.FILL x2A ; *
.FILL x2A ; *
.FILL x2A ; *
.FILL x2A ; *
.FILL x2A ; *
.FILL x2A ; *
; row 1
.FILL x2A ; *
.FILL x20 ; " "
.FILL x20 ; " "
.FILL x20 ; " "
.FILL x20 ; " "
.FILL x20 ; " "
.FILL x20 ; " "
.FILL x20 ; " "
.FILL x2A ; *
; row 2
 ...
```
**Algorithm:**

Register usage:
R2: memory location
R3: row counter
R4: column counter
.ORIG x3000

; load output string's address
LEA R2, SPRITE

; row counter set to 8
AND R3, R3, #0
ADD R3, R3, #8

NEXT_ROW : R3 != 0?
ADD R3, R3, #0
BRz DONE

; column counter set to 8
AND R4, R4, #0
ADD R4, R4, #8

NEXT_COLUMN : R4 != 0?
ADD R4, R4, #0
BRz DONE_ROW

; print next char
LDR R0, R2, #0 ; read next char
OUT
ADD R2, R2, #1 ; move to next char

; decrement column counter and move to next
ADD R4, R4, #-1
BRnzp NEXT_COLUMN

; print new line char
DONE_ROW LD R0, ASCII_NL ; load NewLine ASCII value
OUT

; move to next row
ADD R3, R3, #-1
BRnzp NEXT_ROW

DONE    HALT

ASCII_NL .FILL xA

; row 0
SPRITE  .FILL x2A ; *
         .FILL x2A ; *
         .FILL x2A ; *
         .FILL x2A ; *
         .FILL x2A ; *
         .FILL x2A ; *
         .FILL x2A ; *
         .FILL x2A ; *
         .FILL x2A ; *
; row 1
.FILL x2A  ; *
.FILL x20  ; " "
.FILL x20
.FILL x20
.FILL x20
.FILL x20
.FILL x20
.FILL x2A  ; *
; row 2
.FILL x2A  ; *
.FILL x20
.FILL x5E  ; ^

© Juan Jose Jaramillo. All rights reserved.