CS130 Lab Exercise #9 – Procedure Call & Stack Operation

Overview

In this exercise, students will create an assembly-language program to copy an array of twenty-seven ASCII characters into another array using a so-called "procedure call". As part of the copying process, uppercase letters will be stored as lowercase. More specifically, both character arrays will be a so-called "C-Strings" each containing twenty-seven characters (A → Z), and an ending period, plus a NULL terminating character '**\0**'. Your program will be tested by me on the *zyBooks* simulator of Section 2.24.

Storing Your Program

You can store as a text file your entire program – including all of the values that you choose to store as preset values. Simply use the "Export" feature of the simulator.

Requirements

1. Using the so-called "Preset Dialog Boxes", initialize the simulator with the following values:
   a. **X0** = **4000** (Decimal)
   b. **X1** = **8000** (Decimal)
   c. **X3 = 4000** (Decimal)
   d. **X28 = 12000** (Decimal) Will be used as the "Stack Pointer", **SP**
   e. **X19 = 1919** (Decimal) This value may be used by other parts of the program but will be available for use in the called procedure as an index value; remember to save it on the Stack before the procedure is called.
   f. **X20 = 2020** (Decimal) This value may used by other parts of the program but may be used in the called procedure; remember to save it on the Stack before the procedure is called.
   g. Memory Address **8000** = **0x 47 4645 4443 4241** (ASCII **GFEDCBA**) Note: Hexadecimal values shown for (g) through (j); use the *Windows* 'Calculator' to convert to decimal for entry.
   h. Memory Address **8008** = **0x 4E 4D4C 4B4A 4948** (ASCII **NMLKJIH**)
   i. Memory Address **8016** = **0x 55 5453 5251 504F** (ASCII **UTSRQPO**)
   j. Memory Address **8024** = **0x 2E5A 5958 5756** (ASCII **.ZYXWV**)
2. On Line 1 of your code, enter the comment **// Call StrCpy**
3. At this point, I recommend that you save this initializing configuration. Locate and then click the down arrow (∨) just to the right of the text "**More options**". A dialog box will appear with choices '**EXPORT** ↓' and '**IMPORT** ↑'; click '**EXPORT** ↓' to save this configuration. Click anywhere inside the dialog box. Hold down the **Ctrl** key and press the letter '**C**' on your keyboard. Paste this stored-on-the-*Clipboard* text into *Notepad* and save the file, named as you choose.
3. The procedure you are to create will be called will be named **StrCpy**. On Line 1 of your program, enter the Branch/Link command to call procedure **StrCpy**.
5. Immediately following the call to **StrCpy**, will be the remainder of the program, NOT the procedure itself. However, for this lab exercise, the program will be a simple endless loop. Use Lines 2 and 3 of your code to create an endless loop. On Line 2, name this loop **Halt** and, on Line 3, enter the Unconditional Branch command pointing to **Halt**.
6. Skip Line 4.
7. On Line 5, enter a label for your procedure: **StrCpy:**.
8. On Line 6, "adjust" (i.e. "push down") the Stack (**X28**) to accept two more doublewords.
9. On Line 7, save value of **X19** on the Stack.
10. On Line 8, save value of **X20** on the Stack.
11. You will use Register **X19** as a Loop Counter. Initialize it to zero on Line 9.
12. Although not appearing in the program, we will refer the two arrays as **Upper[]** and **Lower[]**.

13. Starting with Line 9, create a procedure to copy the entire Upper[] array that is stored in Memory addresses 8000 through 8031 as a result of Steps 1(g) through 1(j) above.
14. As part of the copying process of Step 13 above, convert all uppercase letters to lowercase. Note that the period symbol ('**.**') has no 'case' and, therefore should be copied "as is".
15. Store this now-lowercase-array in addresses **4000** to **4031**.
16. Following your procedure, restore (i.e. "pop off of the Stack") the saved value for Register **X20**.
17. Following your procedure, restore (i.e. "pop off of the Stack") the saved value for Register **X19**.
18. Restore the Stack Pointer (i.e. "pop 2 doublewords off of the Stack)
19. The last line of your code should be a command to return from the procedure. Here you will use the Branch Return command, referencing Register **X30**.

Running your program

As you step through you program, observe Register **X30** as the Branch/Link command on Line 1 is executed. "What?" you say! We did not initialize Register **X30** but it suddenly appeared in the list of register in the simulator? Remember that when a procedure is called by a Branch/Link command on Line 1, the assembler saves the address ("Program Counter" some would say) and calculates a return address. This address is 4 bytes greater than the address of the current instruction. In this way, when we return from the procedure, we will be at the address of the first instruction that follows the procedure. Cool?