

CST8134: In-lab Exercise 10

StrUpper and StrLower

You should now be accustomed to using the stack to pass arguments from a calling to a called function and using a stack frame with saved registers and local variables as needed by your function.

In this exercise, you are to create functions that will change all characters in a string passed to it to upper/lower case. You know that 0x20 is the “case” bit for alphabetic characters, and that 'a' to 'z' is 0x61 to 0x7A and 'A' to 'Z' is 0x41 to 0x5A. Note also that these punctuation characters have 0x20 off (all other numeric and punctuation characters have it on,) as is true of the control characters from 0x00 to 0x1F (includes 0x0A or newline):

0x40	01000000	@	(commercial “at” sign)
0x5B	01011011	[(left bracket)
0x5C	01011100	\	(backslash)
0x5D	01011101]	(right bracket)
0x5E	01011110	^	(caret)
0x5F	01011111	_	(underscore or underbar)

There are lots of ASCII tables online; both <http://www.asciitable.com> and <http://www.neurophys.wisc.edu/comp/docs/ascii.html> both seem OK to use if you need to look at one. Check the column headings, though.

In each function, save the work registers you expect to use and create your frame pointer. Convert every alphabetic character in the line whose address was passed as the only argument to either upper case (0x20 off) or lower case (0x20 on), depending on which function was called. You can make the changes “in place” – pick up a character into a register, change it, and put it back. Be sure the case change will do what you expect it to do.

See the Lab 10 Preparation document for sample logic and a description of the driver program you will need.

Show your source code for the driver and both functions, a stack frame map, and the correct output of your program for at least these two sample lines plus an empty line (just press enter):

```
This is {not} all lower case
tHIS IS [MOSTLY] UPPER CASE
```

to your lab instructor before the end of your lab period.