**Shell Commands and Scripting**

In this assignment you will gain experience with the bash shell scripting language. You may work on this project as a team. Each of you should turn in your soluiton to your own repository, and edit "README.txt" to list the names an CCIS ids of your team members.

The GitHub repository 2019SPCS5600SV/assignment-2 contains the bash script **word**. This script implements the functions of the "word.c" functions from assignment 1: *getBitOfWord(word, bitno, bitval)* and *setBitOfWord(word, bitno)*

A word is implemented as a string of *wordsize* 0's and 1's., with *wordtopbit* as the top bit of the word. This implementation uses the built-in bash string functions to set and get bit characters from the word. The word is in *wordendian* format (either **bigendian** or **littleendian**) the default is a 16-bit word in **bigendian** format.

This script accepts one, two, or three arguments.

* If one argument is specified, it is a *wordsize* binary word or an unsigned hex (0xA35F) or decimal value. The script echoes the word as a *wordsize* binary word.
* If two arguments, the first is a *wordsize* binary word or an unsigned hex (0xA35F) or decimal value, and the second is an number between 0 and *wordtopbit*. The script echoes the specified bit of the word.
* If three arguments, the first is a  *wordsize* binary word or an unsigned hex (0xA35F) or decimal value, the second is a number between 0 and wordtopbits, and the third is the bit value either 0 or 1. The program echoes the word with the specified bit set to the bit value.

If no arguments are specified or the arguments are not as expected, the program writes a usage message to stderr and exists with status 1. Otherwise, the program echoes its result and exits with the status 0.

**Implement the ALU Functions**

The first part of the assignment is to create bash shell script **alu**. This script should provide all the "alu.c" functions using the **word** command to implement the functions. The script accepts parameters that represent an operation, and the appropriate number and type of argument required. The script calls the appropriate function, and echoes the word result of the function. You must document the script and the functions similar to the way you did for "alu.c". See the **word** script for an example.

If no arguments are specified or the arguments are not as expected, the program writes a usage message to stderr and exists with status 1. Otherwise, the program echoes its result and exits with the status 0.

The program accepts the following commands and arguments.

|  |  |  |
| --- | --- | --- |
| **command** | **arguments** | **description** |
| testlt | *word* | returns 1 if word is < 0, else 0 |
| testge | *word* | returns 1 if word is >= 0, else 0 |
| testeq | *word* | returns 1 if word is == 0, else 0 |
| ash | *word count* | arithmetic shift of word by count: +: left, -: right |
| csh | *word count* | circular shift of word by count: +: left, -: right |
| lsh | *word count* | logical shift of word by count: +: left, -: right |
| mask | *word count* | masks word with count bits: +: left, -: right |
| and | *word1* *word2* | ands the two words |
| or | *word1* *word2* | ors the two words |
| xor | *word1* *word2* | xors the two words |
| not | *word* | negates the word |
| add | *word1* *word2* | adds the two words |
| sub | *word1* *word2* | subtracts the two words |
| mul | *word1* *word2* | multiplies the two words |
| div | *word1* *word2* | divides the first word by the second |
| rmdr | *word1* *word2* | the remainder of the first word by the second |
| minus | *word* | negative of the word |

**Test Script**

The second part of the assignment is to create bash shell script **testalu**. This script should perform similar tests on the ALU functions to what you did for assignment 1. It will call the **alu** script with test values and validate the returned values. Provide the same level of documentation that you did for your tests in assignment 1.