CSCE 3600: Systems Programming

Major Assignment 1 – GitLab and Bitwise Operators

Due: 11:59 PM on Friday, October 16, 2020

COLLABORATION

You should complete this assignment as a group assignment with the other members of your group as assigned on Canvas using our GitLab environment. If desired, you may submit only one program per group, but all source code *must* be committed in GitLab. Also, make sure that you list the names of all group members who participated in this assignment in order for each to get credit.

PROGRAM DESCRIPTION

In this introductory assignment, you will write a complete C program that will implement a set of operations supported by a menu that will prompt for the operation ("Count Leading Zeros", "Endian Swap", "Rotate Right", and "Parity") along with one integer between 1 and 4294967295, inclusively, and then perform that operation on the input integer to produce the result. Note: The "Rotate Right" operation will also take a second input integer between 0 and 31. The twist is that each operation must be performed using bitwise operators rather than the traditional operations.

This project will be organized in a header file called **`major1.h`**, a source code file called **`major1.c`**, which will contain the `main()` function, and four individual source code files called **`clz.c`**, **`endian.c`**, **`rotate.c`**, and **`parity.c`** that will contain the function definition for their respective operations. The entire group will be responsible in general for non-operation specific functionality in the **`major1.h`** and **`major1.c`** source code files while each individual group member is responsible for his/her own "function operation" in the appropriate files (including the header file and inside the `main()` function).

In particular, you are expected to have the following functionality for each file:

- **`major1.h`**

  This is the overall header file for the project that will contain any include directives and function prototypes. While the include directives are general for the team, each member is expected to add their own function prototype for the operation he/she is responsible for.

- **`major1.c`**

  This is the code file with the `main()` function that will do the following: (1) display the menu, (2) read in the user's response for the menu selection, (3) prompt for and read an integer operand between 1 and 4294967295, inclusively, (4) if the "Rotate Right" operation is selected, then prompt for a second integer between 0 and 31 (inclusively) to obtain the rotate amount, and finally, based on the menu selection, (5) call the appropriate function call for the specified operation, passing the integer operand(s) as parameter(s) to the function. This functionality will be contained in a loop that will continue to iterate until the user selects the option to end the program. If the user enters a valid outside of the 1 – 5 range, you will print a meaningful error message and re-display the menu. Additionally, you will continue to prompt for and read in the integer operand until the user enters an appropriate value (no error message is needed here). While the menu and integer operand is general for the team, each member is expected to add the function call for the operation he/she is responsible for.

- **`clz.c`**

  This code will contain a single function (and the include directive to your header file) to count the number of leading zeroes in the 32-bit input argument passed to the function (leading zeroes are the zeroes that occur in the most significant bit positions of the number until a bit value of '1' is found).

  One team member, and only one team member, will be responsible for the source code in this file, though collaboration with other team members may be done if needed.

- **`endian.c`**

  This code will contain a single function (and the include directive to your header file) to perform the endian swap operation of the 32-bit input argument passed to this function. Endianness refers to how the bytes are stored in memory. In a 32-bit word, there are 4 bytes – B0, B1, B2, and B3. Let us assume that B0 refers to the least significant byte and B3 the most significant byte.

  The function will swap (exchange) bytes B0 and B3, as well as bytes B1 and B2.

  One team member, and only one team member, will be responsible for the source code in this file, though collaboration with other team members may be done if needed.

- **`rotate.c`**

  This code will contain a single function (and the include directive to your header file) to perform the rotation operation using the two integer operands passed to this function. The first integer operand is the number to be rotated. The second integer operand is the number of bit positions that the first operand must be rotated by.

  One team member, and only one team member, will be responsible for the source code in this file, though collaboration with other team members may be done if needed.

- **`parity.c`**

  This code will contain a single function (and the include directive to your header file) to perform the parity computation on the input integer passed to this function. The output of the function is 0 if the input has even parity (that is, the number of 1s in the binary representation of the input is even) and 1 if the input has odd parity (that is, the number of 1s in the binary representation of the input is odd).

One team member, and only one team member, will be responsible for the source code in this file, though collaboration with other team members may be done if needed.

It is emphasized that no built-in functions or library functions must be used for these operations. Each operation must be implemented using ‘C’ language bitwise operators.

It is also emphasized that students should implement these operations on their own and not copy/reuse code from web sources (such as Stack Overflow).

Note that the expectation for this assignment assumes that a group contains 4 students, but if, for some reason, a team has only 3 students, then only 3 operations (i.e., five files) would need to be supported. This means that each team member is responsible for one and only one bitwise operation.

If you have any questions on what is acceptable, please discuss this with your instructor.

SAMPLE OUTPUT (user input shown in **bold**):

```
$ ./bitwisemenu
Enter the menu option for the operation to perform:
(1) Count Leading Zeroes
(2) Endian Swap
(3) Rotate-right
(4) Parity
(5) EXIT
--> -4
Error: Invalid option. Please try again.
Enter the menu option for the operation to perform:
(1) Count Leading Zeroes
(2) Endian Swap
(3) Rotate-right
(4) Parity
(5) EXIT
--> 1
Enter a 32-bit number (>= 1 and <= 4294967295, inclusively): -1
Enter a 32-bit number (>= 1 and <= 4294967295, inclusively): 2
The number of leading zeroes in 2 is 30
Enter the menu option for the operation to perform:
(1) Count Leading Zeroes
(2) Endian Swap
(3) Rotate-right
(4) Parity
(5) EXIT
--> 2
Enter a 32-bit number (>= 1 and <= 4294967295, inclusively): 1
Endian swap of 1 gives 16777216
```

```
Enter the menu option for the operation to perform:
(1) Count Leading Zeroes
(2) Endian Swap
(3) Rotate-right
(4) Parity
(5) EXIT
--> 3
Enter a 32-bit number (>= 1 and <= 4294967295, inclusively): 1
Enter the number of positions to rotate-right the input (between 0 and
31, inclusively): 1
1 rotated by 1 position gives: 2147483648
Enter the menu option for the operation to perform:
(1) Count Leading Zeroes
(2) Endian Swap
(3) Rotate-right
(4) Parity
(5) EXIT
--> 4
Enter a 32-bit number (>= 1 and <= 4294967295, inclusively): 123
Parity of 123 is 0
Enter the menu option for the operation to perform:
(1) Count Leading Zeroes
(2) Endian Swap
(3) Rotate-right
(4) Parity
(5) EXIT
--> 5
Program terminating. Goodbye...
```

REQUIREMENTS

Your header file, source code files, **`makefile`**, and README file will be committed to our GitLab repository as follows. No other files than those mentioned here should be present in the GitLab repository.

- Your source code and header files should be well documented in terms of comments. For example, good comments in general consist of a header (with your name(s), course section, date, and brief description), comments for each variable, and commented blocks of code.
- A **`makefile`** for compiling your source code, including a clean directive. To ensure that your C code is compiled correctly, you will need to create a simple **`makefile`**. This will allow our scripts to just run `make` to compile your code with the right libraries and flags. When using `gcc` to compile your code, please use the `–Wall` switch to ensure that all warnings are displayed. Do not be satisfied with code that merely compiles – it should compile with no warnings! You will lose points if your code produces warnings when compiled.
- A **`README`** file will contain some basic documentation about your code. Specifically, this file should contain the following four components:

    - Your name(s).
    - *Organization of the Project*: Identify the responsibilities for each team member, including which operation he/she was responsible for and what role(s) or expertise he/she served in. Note that this may be used in assessment of grades for this project.
    - Known Bugs or Problems: A list of any features that you did not implement or that you know are not working correctly.
- A completed group assessment evaluation (given at a later date) for each team member. Each team member will be asked to explicitly rate himself/herself along with every other team member. *A student receiving a poor evaluation with regards to their performance on the team will have his/her grading marks reduced by an appropriate amount, based on the evaluation. In addition, the rubric for this assignment allows modification of each individual's portion of the group grade to account for individual contribution to the group's submission, which may result in a member of the group receiving a much higher or much lower grade than other members of the group. A student found to not contribute to the team, will be removed from the team.*
- Your program will be graded based largely on whether it works correctly on the CSE machines (e.g., `cse01`, `cse02`, …, `cse06`), so you should make sure that your program compiles and runs on a CSE machine.

SUBMISSION

- Each team will ensure that all source code and header files, the **`makefile`**, and the **`README`** file are committed to our GitLab repository by the due date and time. If desired, one student may submit all applicable files to Canvas, but it is not required.
- Points in the grade (based on 100 points) will be allocated as follows: 30% of the points will be based on the common components (e.g., the menu, prompting for and reading in the integer operands, etc.) while 70% of the points will be based on the individual contribution specifically associated with the arithmetic operation each student is responsible for in the group.