RAIK 284H Foundations of Computer Systems

Spring 2010
Steve Goddard

Homework 0, January 14
(Total of 50 points)

Evaluation of prerequisite knowledge

Due: 9:00pm Thursday, January 21

Individual Assignment. This is NOT a Team assignment.

50 points: This is a "simple" C programming problem to help familiarize you with some of the basic C language functions and the binary representation of numbers. The requirements are simple:

a. Your C executable program will be named `BitCalc`. Since this is a C program, you should not have any C++ files. Files should be named with the extension .c, not .cpp. In addition, you should use the gcc compiler. Use of the c++ compiler is not allowed.

b. No input parameters on the command line will be required.

c. No C++ functions are to be used (e.g., cout, cin). Use C functions instead (e.g., printf, scanf).

d. The program will first prompt the user for an operation. The possible operations are `|`, &, and ^ (or, and, xor). If `q` is entered as the operation, then the program should quit. If any other operation is input, the program should print the message "Please enter |, &, ^, or q". The program should then output the Enter operation prompt again on the next line. After the operation has been entered, the user will be prompted for a number of integers. The user must enter a valid integer greater than one. Then prompt the user for that many integers in the manner shown below. The integers must be formatted in hexadecimal, and must have a maximum of 8 hexadecimal digits. Input integers with less than 8 digits should be assumed to have leading zeros. Any invalid input should result in the message "Please enter an 8-digit hexadecimal integer", followed by the prompt for that integer on the next line.

   After receiving valid input, the program should then perform the operation on the numbers in the form `firstnum operator secondnum [operator thirdnum etc]`. The program should then output the entire operation (including the operands, operation, and result) in both binary and hexadecimal. The hexadecimal values should be padded so that they have 8 digits. The binary values should be padded so that there are 32 total digits, grouped into clusters of 8 digits. Below is the format for a sample operation, where red text indicates user-entered values.

```
Enter operation: |
Enter number of integers: 2
Enter integer 1: 000000FF
Enter integer 2: 30C01101

Hexadecimal operation:
  000000FF
| 30C01101
= 30C011FF

Binary operation:
  00000000  00000000  00000000  11111111
| 00110000  11000000  00010001  00000001
= 00110000  11000000  00010001  11111111

Enter operation: q
```

e. Here is another output of a possible execution of the program.

```
Enter operation: @
Please enter |, &, ^, or q
Enter operation: &
Enter number of integers: 3
Enter integer 1: DF5383
Enter integer 2: Z0238984
Please enter an 8-digit hexadecimal integer
Enter integer 2: 908070605040
Please enter an 8-digit hexadecimal integer
Enter integer 2: C5204
Enter integer 3: F13FB

Hexadecimal operation:
  00DF5383
& 000C5204
& 000F13FB
= 000C1200

Binary operation:
  00000000  11011111  01010011  10000011
& 00000000  00001100  01010010  00000100
& 00000000  00001111  00010011  11111011
= 00000000  00001100  00010010  00000000

Enter operation: q
```

f. A major goal of this assignment is to test your attention to detail. **Therefore, you must follow the output format shown above exactly**. Labels in your program must exactly match the labels shown above. The same number of spaces that are used above must be used in your program. In particular, there should be one space after the labels before the user input. The operation results should be indented by two spaces, and there should be two spaces between each group of 8 binary digits. There will be no spaces on the right-hand side of any line after the output. Your program will be diffed against expected output, and all formatting errors will result in a loss of points.

g. The coding standard must be followed.

h. Create and turn in a Makefile that will build your executable from your source file(s).

The program will be scored at follows:

Make File	5% (2.5 pts)
README.TXT file	5% (2.5 pts)
Program correctness	50% (25 pts)
Quality of design/readability	20% (10 pts)
In-line documentation/coding standard	20% (10 pts)

No analysis report is required, but a README.TXT file is required to explain program compilation and execution. Program correctness will be determined by the correct functionality of the program as well as adherence to format specifications.

Hints:

1. The C language has several important differences from the C++ language. One of these differences is that single-line comments (comments after //) are not allowed in C. Only comments of the form /* */ are allowed. Also, variable declarations in C must come at the beginning of the code block. The following declaration would be invalid:

   ```
   int alpha;
   alpha = getAlpha();
   float beta = 2.3; /* not valid in C, float beta should be declared before getAlpha() is called */
   ```

   For a fairly complete list of differences between C and C++, see http://david.tribble.com/text/cdiffs.htm.

2. The printf and scanf functions have some very powerful flags to help with formatting input and output, including laying out fields with specified widths and reading/writing hexadecimal integers.

Talk to the teaching assistant if you have questions about how to use the C input/output functions or have questions about the C language.