University of Windsor

School of Computer Science
Faculty of Science

COMP-2650: Computer Architecture I: Digital Design
Fall 2020

Lab#	Date	Title	Due Date	Grade Release Date
Lab 10	Nov 30-02, 2020	**L10: Shift Operators in C/C++**	Dec. 02, 2020 + Extension Wednesday Midnight AoE	Dec. 22, 2020

This the 10th and the last lab. This lab's objectives will be to master using your knowledge of sequential logic to speed up your programs with a programming language, herein, C/C++.

Deadline: According to the school's policy, no assignment can be due on the last week of class or after. Also, we cannot remove the assignments for the last two weeks since it changes the course outline for the marking schema, which is not possible at this time of the semester. This conflict of policies happens because we extend the labs' deadlines for two weeks to reduce the workload. So, here is the solution:

- The official due date is what is mentioned above: Dec. 02, 2020 Wednesday Midnight AoE.
- There is an extension of 1 week to the official deadline for all students. Therefore, we accept submission till Dec. 16, 2020 Wednesday Midnight AoE.
- The difficulty of this week and next week's assignments are in easy level that students are able to do the assignments within the official due date.

Step 1. Environment Setup

Our programming environment is the same as the first lab (Lab 01). In this lab, we want to explore operations related to our knowledge of Boolean logic obtained in this course. In C/C++, there are three categories of operations, aside from the normal arithmetic operations, dealing with Boolean values. They are:

1) Logical Operation: &&, ||, !
2) Relational Operation: <, <=, >, >=, ==, !=
3) Bitwise Operators: &, |, <<, >>, ~, ^

Bitwise operators are the focus of this lab, and, in particular, we want to work with the >> (shift right) and << (shift left) operators. We know that the binary numbers are stored in flip-flops that can store a binary digit 0 or 1. Additionally, we know that the digits of binary numbers have significance based on the powers of 2. For instance, 101 is $1\times2^2 + 0\times2^1 + 1\times2^0 = 5$ in decimal system. If we are given a byte, we have 8 flip-flops, e.g., D flip-flops, to store this number as 00000101. What happens if I shift the numbers in these flip-flops one time to the right:

1) Remove 1 in the lowest significant bit
2) Shift all other bits in the higher significant bits to the right by one time
3) Repeate the highest significant bit to preserve the sign of the number

This is called *arithmatic* shift right as it preserves the sign of the number after shift operation. So, 00000101 becomes 00000010. It means that we decreased the significance of bits by division by 2. The bit in 3rd position that has the significance of 2^2 now has 2^1 significance. Practically, we divided the number by 2 and kept the integer quotient, which is 5/2 = 2.

If we do another shift to the right, 00000010 becomes 00000001, which is 2/2=1.
If we do another shift to the right, 00000001 becomes 00000000, which is 1/2=0.

As seen, division by 2 can be done via shifting to the right one time. In general, division by 2^i is equal to shift the bits to the right *i* times, as shown below:

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
	setbuf(stdout, NULL);

	int a;

	printf("First Operand:");
	scanf("%d", &a);

	printf("%d/2 = %d \n", a, a >> 1 );
	printf("%d/4 = %d \n", a, a >> 2 );
	printf("%d/8 = %d \n", a, a >> 3 );

}
```

Sample runs would be:

```
First Operand:64
Normal divisions:
64/2 = 32
64/4 = 16
64/8 = 8
Divisions by arithmetic shifts to right:
64/2 = 32
64/4 = 16
64/8 = 8

First Operand:35
Normal divisions:
35/2 = 17
35/4 = 8
35/8 = 4
Divisions by arithmetic shifts to right:
35/2 = 17
35/4 = 8
35/8 = 4
```

How about negative numbers?

```
First Operand:-35
Normal divisions:
-35/2 = -17
-35/4 = -8
-35/8 = -4
Division by arithmetic shifts to right:
-35/2 = -18
-35/4 = -9
-35/8 = -5
```

```
Normal divisions:
-32/2 = -16
-32/4 = -8
-32/8 = -4
Divisions by arithmetic shifts to right:
-32/2 = -16
-32/4 = -8
-32/8 = -4
```

As seen, the results of divisions are the floor of the quotient, e.g., $-35/2 = \lfloor -17.5 \rfloor = -18$. The results are different by 1 unit for divisions that have remainder than when you normally divide a negative number by powers of 2. Why? This is not the case for -32, though.

The division operation is a costly action that requires multiple clock pulses. However, shifting to the right can be done in only 1 pulse. So, whenever a division by 2 or powers of 2 is needed, shifting is a preferable and wise choice to have an efficient (fast) program.

What happens if we do shift left (<<)?

Lab Assignment

You should investigate what happens when you shift the bits of a binary number to the left (<<) and in what arithmetic we can use it. Try for both positive and negative numbers. Write the main function with samples to illustrate your findings as we did above.

Deliverables

You will prepare and submit the program in one single zip file `COMP2650_Lab10_{UWinID}.zip` containing the following two items:

1. The entire project folder `COMP2650_Lab10_{UWinID}`, including the code (source) files and executable file.
2. The result of the commands in the file `COMP2650_Lab10_Results_{UWinID}.jpg/pdf`. Simply take a screenshot of the results and save it. If multiple images, please print them all into a single pdf file.
3. *[Optional and if necessary]* A lab report document in the PDF file `COMP2650_Lab08_Report_{UWinID}.pdf`. It should include:
    a. Your name, UWinID, and student number
    b. One paragraph describes the program that you attached, along with any prerequisites needed to build and run the program. *Please note that you will lose marks if your program cannot be built and run on our computer systems.*

In sum, your final zip file for the submission includes 1 folder (entire project folder), 1 image/pdf (results snapshot), and 1 pdf (report). *Please follow the naming convention as you lose marks otherwise.* Instead of {UWinID}, use your own UWindsor account name, e.g., mine is hfani@uwindsor.ca, so,

```
COMP2650_Lab10_hfani.zip
  - COMP2650_Lab10_hfani
      o src
          ▪ ... (whatever source/header files required to build the program)
          ▪ main.cpp
      o COMP2650_Lab10_hfani [.exe in MS-Windows]
  - COMP2650_Lab10_Report_hfani.pdf
  - COMP2650_Lab10_Results_hfani.jpg or COMP2650_Lab10_Results_hfani.pdf
```