CPE 225
Assignment 1

Name:

Objectives:

Review binary, hexadecimal, and logical operations
Navigate the BASH shell
Start LC-3 Simulator

This is an individual lab. You may **not** have a lab partner, but you may talk to one and compare answers.

Save this document as with answers to an unformatted txt file named "225asgn1.txt".

1 Binary Arithmetic

Complete the worksheet on binary arithmetic on the following pages. **Each section must be 100% correct to get each point.**

For each of the sections below, you must get the answers completely correct to get the point.

1.1 Unsigned 8-bit arithmetic (1 pt)

For the following **unsigned**, 8-bit numbers, what is the value (in decimal) of each of the following numbers:

Binary Value : Decimal Value

00000000 : ________________
00101100 : ________________
11111111 : ________________
00010000 : ________________
10000000 : ________________
00010001 : ________________

For **unsigned** numbers, what is the range of values that can fit in 8 bits? I have done the first one for you to show you what I expect.

Minimum Values
binary: 10000000
decimal: 0

Maximum Values
binary: ________________
decimal: ________________

1.2 Signed 2's Compliment 8-bit arithmetic (1 pt)

For signed, 8-bit numbers, what is the value (in decimal) of each of the following 2's complement binary numbers:

Binary Value : Decimal Value

00000000 : ________________
00011100 : ________________
11111111 : ________________
00010101 : ________________
10000000 : ________________
10111001 : ________________

For **2's complement signed** numbers, what is the range of values that can fit in 8 bits?

Minimum Values
binary: ________________
decimal: ________________

Maximum Values
binary: ________________
decimal: ________________

1.3 Carry and Overflow (1 pt)

In binary arithmetic, an ***unsigned*** operation whose result does not lie in the range of valid numbers produces a **carry overflow**. (A bit was carried out the left side of the addition.)

Write two 8-bit ***unsigned*** numbers that, if added, result in carry. Write the same numbers in decimal as well as binary - truncate the binary to 8 bits, *write the translation of the binary answer in decimal*. **Note that the answer to the decimal addition will be incorrect!**

Binary:	___________	+	___________	=	____________
Decimal:	___________	+	___________	=	____________

In binary arithmetic, a ***signed*** operation can have a result with a sign that is incorrect. This is called an overflow. For example adding two negative numbers and getting a positive result produces an **signed overflow** (and also produces a carry).

Write two 8-bit ***signed*** numbers that, if added, produce a sign overflow, **but not a carry overflow**. Write the same numbers in decimal as well as binary - truncate the binary to 8 bits, *write the translation of the binary answer in decimal.* . **Note that the answer to the decimal addition will be incorrect!**

Binary:	___________	+	___________	=	____________
Decimal:	___________	+	___________	=	____________

1.4 Hexadecimal (1 pt)

Translate the following values into hexadecimal:

00000000 : ________________
00001011 : ________________
11111111 : ________________
00010101 : ________________
10001000 : ________________
11010110 : ________________

Add the following 16-bit hexadecimal values:

x24A3 + x5678 = ______

xABCD + x1EEE = ______

Subtract the following 16-bit hexadecimal values:

x94AA - x5678 = ______

xFBCD - xEEEE = ______

1.5 Bitwise Logic Operations (1 pt)

Perform the following bitwise logical operations.

x24A3 & x5087 = ______

xABCD | x1EEE = ______

Determine both the operation (ADD/AND/OR/NOT) and the value needed to set bit 5 of the following value to zero (clear bit 5) in a single operation. Leave the rest of the bits unchanged. (Hint: expand the values to binary.)

x947A ___ ______ = ______

Determine both the operation (ADD/AND/OR/NOT) and the value needed to set bit 4 of the following value to one (set bit 4) in a single operation. Leave the rest of the bits unchanged.

x76CA ___ ______ = ______

2 Navigate the BASH shell in the Linux Environment

This section is a tutorial only and has no questions to complete.

2.1 Get connected

2.1.1 Find a lab machine. It doesn't matter which

Log in to Linux using your Calpoly username and password. We will be using Linux throughout the quarter, so becoming familiar with it is a priority.

Log in from home. Using any SSH terminal program, log into unix1.csc.calpoly.edu from home using your login ID and password.

> Trouble logging in? Try again. Capitalization matters.
> If you are a non-CSC major and not yet enrolled in the course, you will not yet have an account. See the lab staff in or around the open lab downstairs (14-235)

2.1.2 Get started

Feel free to look around the computer a little bit. Try to find the command shell (the application is likely called “Terminal” or “Shell”). As an icon, it looks like a computer monitor with text. When launched, it will open a small window where you type commands for immediate execution.

For helpful tips, browse these pages on basic Unix commands:

http://www.math.utah.edu/lab/unix/unix-commands.html

http://www.washington.edu/computing/unix/unixqr.html

Make a note of these locations (or bookmark them!). You’ll want to refer to them throughout the quarter.

When the terminal is first opened, you will likely see an odd bit of text, followed by a cursor. That text will likely contain your username (the name you logged in with), machine name (your computer's name), and current working directory. A directory is what a folder is called in Linux. The working directory is the directory you currently have open. This bit of text may look like gibberish right now, but will become very handy when you begin navigating through the computer using the command line. This is called the prompt.

2.2 Basic Commands

Read this part carefully and do this!! First, I want to make sure you do all your work on the same computer that I will be testing your work on. Right now you are working on the "local machine", the one sitting on the desk next to you. You need to compile, test, and handin your programs from one of the unix1-unix4 machines (it doesn't matter which). I will use unix1-4 to test your assignments. If your submittal does not work on one of these machines, it will be graded as not working. To work on one of those machines you need to ssh to it. The following command shows how you would ssh to unix1. You will need to re-enter your password. When you type your password, you will not see any characters printed to the screen. This is normal. Be careful – if you enter an incorrect password 3 times in a row, your account will become locked for 30 minutes.

```
ssh unix1.csc.calpoly.edu
```

Now that we're on unix1, let's make a directory for your CPE 225 coursework called "cpe225 ". Look at the list of Unix commands at the link above to find the right command. (This is the command-line equivalent of creating a new folder through Microsoft Explorer, Mac Finder, Gnome Nautilus, etc.)

Now you want to change your working directory to the directory you just created. What is the command to do this?

Once you are in the cpe225 directory, make a "asgn01" directory, then change to that directory. This is where we will keep all the files for this lab.

To make sure you're in the right place, use the "pwd" command to print (display) the present working directory. It should look similar to:
/home/*username*/cpe225/asgn01

Here is the terminal output when I run each of the commands that perform the actions above. I've noted the commands in bold. Your colors and prompt will vary:

```
jplanck@unix1:~ $ mkdir cpe225

jplanck@unix1:~ $ cd cpe225

jplanck@unix1:~/cpe225 $ mkdir asgn1

jplanck@unix1:~/cpe225 $ cd asgn1

jplanck@unix1:~/cpe225/asgn1$ pwd

//home/jplanck/cpe225 /asgn1

jplanck@unix1:~/cpe225 /asgn1$
```

2.3 Running command-line programs:

We will be creating and running many programs in this class, and most of what we do will be through the command line. To familiarize yourself with running programs from the command line, we will run each of the below programs a few times. Run these from the command line by typing the command as you see it. The first word in the command is the name of the program; the pieces that follow are “arguments” that the program needs to run (more on this later in the quarter).

- cal 1 2018

The program “cal” displays calendars for given dates, in this case January 2010. You can enter “cal” with both a month and year, or only a year. Try a few different values.

- who

The program “who” displays the user id for all users currently logged into that particular server. Note: This tool can be useful if you notice that your session is running slowly. There may be too many users logged in and you can log into a different server for better response times.

There are no deliverables for Part 2 of this lab.

3 Using the LC-3 Simulator

Refer to the UNIX LC3 Simulator quickstart link on the class PolyLearn page for use of the LC-3 simulator. You can also get help by typing "help" after starting the LC-3 simulator. You will have a great deal of difficulty completing this portion of the assignment without referring to the LC-3 PolyLearn page.

3.1 Starting the LC-3 Simulator (1 pt)

The LC3 Simulator is a command line unix tool. Start the LC-3 Simulator by executing "lc3sim" from the command line on unix3.

3.1.1 What is the Program Counter ***Register*** **set to by default? __________**

3.1.2 What is the content of memory location x0023 in binary? __________

3.1.3 What is the content of memory location xFFFE in hex? __________

3.1.4 What is the opcode for the instruction at memory location x0590? (How do you get a "listing" of the instructions at an address?) Write out the binary digits of the opcode as well as the name of the instruction. __________

3.2 Let's make the LC-3 DO something… (1 pt)

a. Set the value of Register 0 (R0) to decimal 100.
b. Set the value of Register 1 (R1) to hexadecimal x23.

3.2.1 What is the hexadecimal value of the data in R0? __________

3.2.2 What is the decimal value of the data in R1? __________

3.2.3 Modifying Memory

a. Set the Program Counter Register to x3000.
b. List the contents of memory at the PC (Program Counter)
c. Change the data in memory location x3000 to be an ADD instruction to add R0 to R1 and put the result in R2. Hint: Write out the command in binary. Then figure out the value of that command in hexadecimal. **Set the value in *memory* at location x3000 to the hexadecimal number that you came up with**.
d. Now we're almost ready to run the command. Set a break point on the next line at the address x3001.

e. List the contents of memory at the PC (Program Counter) again. There should be a "B" on the left of x3001 in the listing to indicate that a breakpoint is set.
f. Now you can run your code by typing "c" to continue running program. Note that there is no distinction between starting your program and running it in the LC-3 simulator. This is an important distinction when you start using real simulators.
g. R2 should now contain the sum of R0 and R1. Note that the Program Counter was advanced to x3001.

3.2.4 What is the hexadecimal value of the data in R2? __________

If you are unsure if you have done this correctly, or if you have any questions about what you have done so far, or if you have any questions about the simulator after reviewing the PolyLearn quick-start guide, call your instructor over for questions.

3.3 Let's do more complicated things with ADD, AND, and NOT… (1 pt)

The LC-3 only has three Operate (math) instructions, but we can do a lot of more complicated things with them. Figure out how to do the following things using only ADD, AND, and NOT. I have done the first one for you to show you what notation to use. Verify your answers using the LC-3 simulator like we did in Part 2. Be sure to put a break point after your code!

Example Problem: **R2 = R0 – R1** (3 instructions max)

Example Solution:

NOT R1 R1$\longrightarrow$

R1 ADD #1 R1 $\longrightarrow$ (the # means immediate mode)

R0 ADD R1 R2$\longrightarrow$

In my solution I first change R1 to be negative. Then I add it to R0. Note that I use an arrow to show the destination register for each operation. Also, use a # sign if you are indicating a number.

For the questions below, **write the instructions in hexadecimal**. Write instructions that will always accomplish the goal, no matter what values are in any of the registers. In other words, **don't rely on one of the registers being 0** for your instructions to work.

3.3.1 Copy R0 to R1 (1 instruction max) __________

3.3.2 Clear R0 to zero (1 instruction max) __________

3.3.3 Clear bit 2 (set it to 0) of R0 but leave the rest of the register unchanged. Note how the bits are numbered starting at 0. (1 instruction max) __________

Test your answer to Q3.3 by setting the value of R0 to all ones (xFFFF) and then running your command. The result should be 1111111111111011 (xFFFB). Test with other values as well. Bit 2 should be 0 (no matter what it was before), and all other bits must remain unchanged.

4 Handin – This part must be done individually (each student must do this).

Save this completed document as a textfile named “225asgn1.txt”. The file 225asgn1.txt is your homework submission. Don’t forget to put your name where indicated at the top of the document. Transfer the document (using SecureFTP, Cyberduck, Filezilla, or something similar) to your Cal Poly Unix account. The “SSH and SFTP Info” link in PolyLearn has more information on how to use SFTP.

Handin 225asgn1.txt **by the end of lab** on the day it is due using the command below on the SSH terminal. You must handin to your lecture section. Replace the XX with your section number of the lecture. **Handin will CLOSE at the end of the lab period.**

```
handin jplanck 225-XX-asgn1 225asgn1.txt
```