Programming for Engineers

Laboratory class 9

SECTION 1

This section introduces programming in machine code, using the Brookshear Machine language. A Quickstart guide that includes a summary of the instruction set can be downloaded on the **Study Direct *Resources* page**. You should have a look at this before you attempt the exercises below.

You can run Brookshear Machine language programs by downloading the program file

`JBrookshearMachine_1.5.1.jar`

from our Canvas page (Resources). Save this file in your personal documents / module folder and double-click to run, when needed.

The emulator main window contains a number of sub-panels, which show main memory, registers etc. For information about the emulator and the Brookshear machine, press ‘Help’ in the Information panel.

Note: all memory addresses, data and instructions below are in **hexadecimal**.

Programs are written with one instruction per line, but in the machine each instruction needs to be stored across **two** subsequent bytes of memory.

In all cases, the code is assumed to be loaded into memory starting from location 0, and the program counter (PC) points to 0 when execution starts.

1. What is the hexadecimal equivalent of the decimal number 20?

2. What is the binary equivalent of the hexadecimal number 74?

3. In the two's complement representation for integers, what bit pattern would be used to represent the decimal number -8?

4. Below is a program in Brookshear Machine language. Referring to the instruction set documentation, work out on paper what the program does. Do NOT enter the program into the Brookshear machine just yet.

   ```
   1110
   1211
   3111
   3210
   C000
   ```

   Next, start the simulator and enter the program in the simulator's main memory by editing the memory cells in the 'Hex' column. Remember that you need to split each instruction across two adjacent memory locations. The Return/Enter and down-arrow keys allow you to move down to the next location. Look at the '**Instruction**' field to check you worked out what the program does correctly.

   Finally, run the program, having first entered some data into memory cells 10 and 11 (HEX). You can run a program one instruction at a time with the '**Do Step**' button. To restart from the beginning, press '**Reset**' in the Registers panel. '**Reset & Run**' starts the program from the beginning and runs it at the speed controlled by the slider.

   What is the effect on memory of running the program twice?

5. Analyse the following program in the same way, and then enter and run it to check you were right.

```
1110
1211
9312
3312
C000
```

6. Modify the program of question 5 to add the data in memory locations 14 and 15 as integers, storing the result in location 16. Run it to check it works correctly.

SECTION 2

1. The program below is a solution for the earlier question 3. It adds two integers.

    ```
    1114
    1215
    5312
    3316
    C000
    ```

    Enter this program in the emulator, and run it with “FE” in location 14, and “06” in location 15. What is the result of this run? Explain the result in the cases where the input and result bit patterns are interpreted as (i) two’s complement integers, and (ii) unsigned integers.

    Do some more experiments with different values in locations 14 and 15 to check that you understand the behaviour of integer addition.

2. Referring to the slides, work out what floating point numbers are represented by the following bit patterns:

    ```
    00101000  01101010
    ```

    Check your answers by entering these bit patterns into the BM memory and looking in the ‘Float’ column.

    Modify the program of question 1 to add two floating point numbers together, and run it on these two numbers. What result do you get, and why? Experiment with different floating point numbers to get an understanding of the limitations of floating point arithmetic.

3. What are the largest and smallest positive Brookshear machine floating point numbers? What happens if you try to add the largest floating point number to itself?
   According to the IEEE 754 standard, how does the behavior of the emulator differ from what should happen when trying to add the largest IEEE 754 floating point number to itself? You will need to do a bit of research to find the answer.