Worksheet 2 (Lab)

This lab is based on the material on assembly language from lectures 3 and 4. You will use a software emulator for the IT Machine to experiment with some small assembly language programs. The work is not assessed, but you should hand in your program from the section **Writing a Program** at the end of the lab. Your tutor's comments will help you to understand this section of the course.

Loading and Executing a Program

1. Go into AMS and set up the CS1Q exercise "Week17".

2. In your folder Workspace/CS1Q/Week17, you should see a file called IT_Machine. Double-click on this file. You should see a display which looks similar to the illustrations of the IT Machine in lectures: there are boxes showing the registers, the memory, the current instruction, etc. All of these boxes will be blank.

3. Use the Programmers' File Editor to open the file "Program1" (it's in the folder Week17/programs). Try to arrange your desktop so that you can see the assembly language program and part of the IT Machine display.

4. Select "Open" from the "File" menu in the IT Machine, and open the file "Program1". The display will not change as a result of opening the file.

5. Select "Assemble" from the "Run" menu. This converts the instructions from assembly language to machine language. Select "Assembler Listing" from the "View" menu. You will see a window containing the assembly language program and its machine language equivalent, in hexadecimal. In general, any errors in the assembly language program will be shown in the assembler listing. In this case, we know that the program is correct so there are no errors.

6. Select "Load" from the "Run" menu. This loads the assembled program into the memory of the IT Machine. The "Memory" display has two scrollable views of the memory. Each view lists addresses on the left, and contents on the right.

7. Just below the "View" menu there is a button with a right arrow on it. Clicking on this button executes one instruction. Click on it to execute the first instruction in the program. The instruction being executed is highlighted in white in the memory display. The instruction is also shown in the "Current Instruction" display in two forms: machine language and assembly language. The "Assembly Statement" display shows the corresponding line from the original assembly language program.

   The first instruction in this program loads a value into register R6. The new value of R6 is highlighted in red in the register display area.

8. Execute the next instruction. Again you will see the instruction highlighted in the memory display. This instruction stores a value into the memory location labelled "x". In the right hand side of the memory display, scroll down so that you can see the new contents of location "x" (which is actually location 0014) highlighted in red.

9. Step through the rest of the program. Notice that the ADD and MUL instructions show which registers are being used, and cause the "ALU" display to indicate the operation being carried out.

Modifying a Program

1. Open the file "Program2" in your editor. This is the program from Lecture 4 which calculates the sum of the integers up to some value n.

2. The program is incomplete because the value of n has not been specified. When designing the program we decided to use register R1 for n. Add an instruction at the beginning of the program which sets R1 to a particular value (try something fairly small). Save the file.

3. Load the file into the IT Machine and assemble it. This time it is important to check for errors by viewing the assembler listing. After correcting any errors (and saving the file), you must open the file again in the IT Machine, and assemble it again.

4. When the program assembles without errors, load it and run it. You can either step through the program one instruction at a time, or click on the double arrow button to run it all the way to the end.

5. When the program terminates, register R2 should contain the sum of the integers up to n. Is the result what you expected? (Remember that the IT Machine uses hexadecimal.)

6. Open the file "Program3" in your editor. This is another copy of the same program. Modify the program so that it corresponds to the following Ada code:

```
s := 1;
while n > 0 loop
  s := s * n;
  n := n - 1;
end loop;
```

   Again, you will also need to add an instruction to set n to a particular value.

7. Open, assemble and load this file into the IT Machine as before, remembering to check for errors after assembling. Run the program. What does it calculate?

Writing a Program

Using the example programs from lectures as a guide (especially the program for finding the largest element of an array, from Lecture 4), write a program which calculates the sum of the elements of an array. The file "Program4" is provided for you to put this program in. Assume that the end of the array is indicated by the value -1. You should go through the following steps, on paper, before entering your program into a file and testing it.

1. Write the algorithm in Ada.
2. Decide whether to use registers or memory locations for the variables. (The array will be in memory, of course.)
3. Translate the algorithm from Ada into assembly language. The overall structure of the program should be similar to the largest element program.
4. Remember to declare memory space for the array and any variables which you are storing in memory. Remember to put some specific values into the array, and end the array with a value of -1.
5. Include comments in your program, so that you can keep track of what each instruction means in relation to the Ada code.
6. Remember to end your program with the instruction `CALL exit[R0]`.

Hand in, on paper, your design work and program for this part of the exercise, at the end of the lab.

Supplementary

1. Try out some of the optimizations mentioned in Lecture 4.
2. Try writing some other programs.
