CPSC 201: Introduction to Computer Science
Carsten Schürmann
Date: February 21, 2002

Homework 4

Due: 11:30am
Wednesday, February 27, 2002.

Problem 1: Linear Search (50 points)

Linear search is the simplest algorithm to search for an element in an list. It makes no assumptions regarding the organization of the list over which search is to be performed. It simply traverses a list step by step and compares every element with the one we are looking for.

Write an assembly language program for linear search for a microprocessor described below. You don't have to implement your program, but it should be "in working condition". Note, that in machine programs we don't have the convenient argument passing machinery as in high level languages. Arguments are passed through registers, and so should your implementation of the linear search routine. Our microprocessor has five registers labeled R_0, R_1, R_2, R_3, R_4 and understands the machine instructions given below.

The input arguments to your program will be stored in registers as follows. The byte we would like to find in memory is contained in register R_1, the address where the search is to start from in register R_2, the address where to stop in register R_3. Your machine program is expected to traverse the memory in between addresses R_2 and R_3 print `1` on the screen if the search was successful, `0` if it was not.

As example consider the contents of memory as follows:

```
2000   54
2001   92
2002   45
2003   40
2004   54
2005   *
```

and the registers set as follows R_1 =`40`, R_2 =`2000`, and R_3 =`2005`. Your program should print a `1` on the screen. Here is a summary of the commands you are allowed to use. We use A for addresses.

`LOAD` A R_i: Load the contents of memory at address A into register R_i.

`LOADI` R_i R_j : Load the contents of memory at the address contained in register R_i into register R_j.

`MOVE` R_i R_j: Move the contents of register R_i to register R_j.

`STORE` R_i A: Store the contents of register R_i into memory at address A.

`ADD` R_i R_j: Add contents of registers R_i and R_j and store the result in register R_0.

`MULT` R_i R_j: Multiply contents of registers R_i and R_j and store the result in register R_0.

`CONST` C R_i: Moves constant C to register R_i.

`JMP` A: Jump to memory address A and continue program execution from that location.

`CJMP` A: Jump to memory address A iff register R_0 is zero.

`OUT` R_i: Output contents of register R_i.

`HALT`: Halt the program.

Hints and Assumptions:

- Assume that memory has 1000 locations from address 0 to address 999. You might want to divide the memory into two portions, one to store the program and other to store the data.
- Each instructions requires just a single memory address.

Problem 2: Largest Element (10 points)

Write an SML function `max` to find the largest element in a list of real numbers.

1. What is the type of this function?
2. What is the invariant of this function?
3. Write the function itself.

Problem 3: Boolean Logic (40 Points)

A majority function $M_n(x_1, \ldots, x_n)$ is *true* if a majority of its inputs are *true* and *false* otherwise (assume n odd). In this assignment we study majority functions with *three* inputs.

1. Write the truth table for a majority function $M_3(x_1, x_2, x_3)$ with just three inputs.
2. Write the Boolean expression for this majority function.
3. Construct the circuit for $M_3(x1, x2, x3)$ using only AND and OR gates.
   Hint: A typical implementation uses three AND gates and three OR gates. Reformulate your answer from 2. using distributivity laws.
4. Does $M_3(x_1, x_2, x_3)$ and NOT gate form a complete Boolean basis? Justify your answer.
   Hint: A NAND Gate alone, a NOR Gate alone, or the set of AND, OR and NOT gate form a complete Boolean basis. Try to implement one of these (or possibly other basis) using the $M_3(x_1, x_2, x_3)$ and a NOT gate.