CPSC 201: Introduction to Computer Science
Carsten Schürmann
Date: February 21, 2003

Homework 5

Due: Friday, February 28, 2003.

Guidelines

While we acknowledge that beauty is in the eye of the beholder, you should nonetheless strive for elegance in your code. Not every program which runs deserves full credit. Make sure to state invariants in comments which are sometimes implicit in the informal presentation of an exercise. If auxiliary functions are required, describe concisely what they implement. Do not reinvent wheels, and try to make your functions small and easy to understand. Use tasteful layout and avoid long winded and contorted code. None of the problems requires more than a few lines of SML code.

Exercise 1 An operation that is often used in computer programs is a function that copies the content of a block of memory addresses starting at a given address to the another block of memory addresses at a given destination address. It traverses the original block address by address and copies the content to the destination, gradually increasing the destination address.

Write an assembly language program for copy in the language described below. Implement it on our simulator we have discussed in class. 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 copy function. Our microprocessor has five registers labeled R_0, R_1, R_2, R_3, R_4.

The input arguments to your program will be stored in registers as follows. The start and end address of the original memory block is given in register R_1 and R_2, respectively, and register R_3 contains the destination address.

As example consider the contents of memory as follows:

```
2000  54
2001  92
2002  45
2003  40
2004  54
2005  *
2006  *
2007  *
2008  *
2009  *
```

and the registers set as follows R_1 =2001, R_2 =2003, and R_3 =2006. Upon termination, your program should leave the memory in the following state

```
2000  54
2001  92
2002  45
2003  40
2004  54
2005  *
2006  92
2007  45
2008  40
2009  *
```

`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:

- Your program should be stored at memory address 100.
- Each instructions requires just a single memory address.