Simple Register Machine

CSE4102 1, Spring 2018

Sarthak Bhatnagar

2/4/18

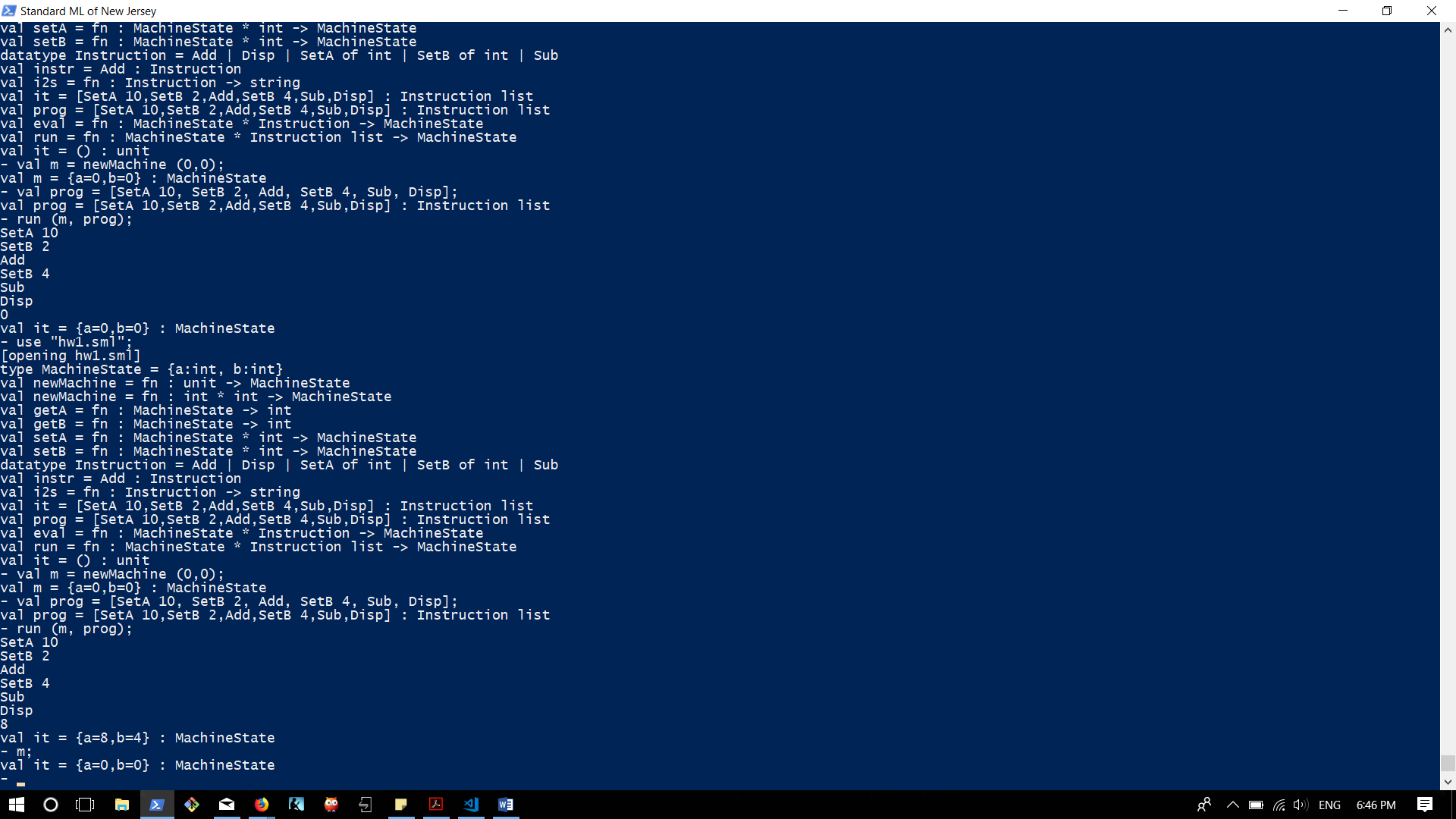
Section: 001

Instructor: Jeffrey A. Meunier

Introduction

In this assignment we will create an interpreter for a very small imperative language that in many ways resembles an assembly language. This language will run on a two-register machine (computer). The machine will be simulated by a record data structure and several functions. The end product is a program that we will enter in a single .sml file.

Output



Source Code

(\* {a=0, b=0}; \*)

type MachineState = {a:int, b:int};

fun newMachine () : MachineState = {a=0, b=0};

fun newMachine (a, b) : MachineState = {a=a, b=b};

fun getA (m : MachineState) = #a m;

fun getB (m : MachineState) = #b m;

(\* takes a MachineState and returns a new MachineState with the register changed to a new value \*)

fun setA (m: MachineState, newA) = newMachine(newA, #b m);

fun setB (m: MachineState, newB) = newMachine(#a m, newB);

datatype Instruction = SetA of int | SetB of int | Add | Sub | Disp;

val instr = Add;

fun i2s (SetA a) = "SetA " ^ Int.toString(a)

|i2s (SetB b) = "SetB " ^ Int.toString(b)

|i2s Add = "Add"

|i2s Sub = "Sub"

|i2s Disp = "Disp"

;

[SetA 10, SetB 2, Add, SetB 4, Sub, Disp];

val prog = it;

fun eval (m : MachineState, SetA a) : MachineState = setA(m, a)

| eval (m : MachineState, SetB b) : MachineState = setB(m, b)

| eval (m : MachineState, Add) : MachineState = setA(m, getA(m) + getB(m))

| eval (m : MachineState, Sub) : MachineState = setA(m, getA(m) - getB(m))

| eval (m : MachineState, Disp) : MachineState = (print (Int.toString (#a m) ^ "\n"); m)

;

fun run (m: MachineState, [] : Instruction list) = m

| run (m : MachineState, prog : Instruction list) =

let val instr = hd prog

val instrs = tl prog

val \_ = print (i2s instr ^ "\n");

val m1 = eval (m, instr);

in

(\* (print (i2s instr ^ "\n"); \*)

run (m1, instrs)

end;

;