CS2104 – Programming Languages
School of Computing
National University of Singapore

August 26, 2012

Problem Set 1

Semester 1, 2012/13

Due: ***September 9, 23:59***

12 marks

Submission: In IVLE, in the cs2104 workbin, you will find a folder called "`Homework submissions`". In that folder, there are currently ***5 subfolders:*** **`PS1P01,…,PS1P05`**. The last two digits of the folder name indicate the solution that is to be submitted into that folder: the solution to *Question 1* into **`PS1P01`**, and so on (that is, you need to submit 5 separate solutions to 5 questions). A solution should consist of a ***single text file*** that can be compiled, or loaded into the interpreter of interest and executed. You should provide as much supplementary information about your solution as you can, ***in the form of program comments***.

Problem 1 [B] [1 marks, submit to `PS1P01`]

Write a VAL program that tests whether the sum of all the registers `eax`, `ebx`, `ecx`, `edx`, `esi`, and `edi`, is odd or even. At the end of the program, register `eax` should contain the result: `0` if the sum is *even*, and `1` if the sum is *odd*.

Problem 2 [C] [2 marks, submit to `PS1P02`]

Write a VAL program that converts a string made up of digit characters ('0'..'9') into the 4-byte unsigned numeric value corresponding to that string. Your program should assume that the string resides in memory at the address pointed to by register **`esi`**, and should return the result in register **`eax`**.

Problem 3 [A] [4 marks, submit to PS1P03]

Write an interpreter for the following language, which we shall call *L*. Programs of the language *L* are sequences of simple instructions (no whitespaces are allowed). Each instruction has the form

$$variable \otimes = operand;$$

where *variable* is a variable name, following the syntax of C identifiers, $\otimes$ is any of the operators +, -, *, or /, and *operand* is either a variable, or an unsigned integer. The instruction is terminated by semicolon. All the variables are considered to be initialized with 0 before the program starts execution. The interpreter should print the values of all the

variables in the program at the end of the execution. Your interpreter should be implemented in C.

Execution example:

For the input string: "**`x+=3;y+=5;x*=y;`**"
Interpreter output: **`x = 15  y = 5`**
Hint: Implement your interpreter as a deterministic finite automaton (slide 15 in `cs2104-L02-pre.pdf`).

Problem 4 [C] [2 marks, submit to PS1P04]

Write an VAL program that computes the first N prime integers (each taking up a space of 4 bytes), where N is a parameter of the program that is made available in register ecx. The prime numbers you compute should be placed in increasing order in the memory area with the address range 0 … 4*N.

Problem 5 [A] [3 marks, submit to PS1P05]

In VAL, a linked list is implemented in the following way. Each list element is an 8-byte entity, whereby the first group of 4 bytes represents the data field of the element, and the second group of 4 bytes represents the pointer to the next element. In our case, the 4 data field bytes shall be interpreted as an integer. Thus, list elements are sequenced together by interpreting the second group of 4 bytes in an element as the address of the next element in the list. The last element in the list has this "next element" field set to 0 (implying that no list element can reside at address 0).

Write a VAL program that receives the address of a list in register **`esi`**, and **computes the sum of the elements in the list**. To test your program, write a **`main`** function that sets up a linked list with the elements placed in memory in as much a random fashion as possible (but remember not to place a list element at address 0). Your **`main`** function should then call **`exec()`**, and then finally print the value of **`eax`**, which should be the desired result of the VAL program.