CSI 504  
Homework 2  
Professor Chaiken  
Spring 2000  
Assigned 2/1/00  
Due 2/10/00

To submit your work from the computer exercises done at Academic Computing Services, print text files with the lpr command and retrieve the paper output from LC-4.

It is not really necessary for this assignment, but you can get more information on Java programming from www.javasoft.com, and more information on Postscript from the Adobe books I will place in Library Reserve under CS504.

1. At the bottom of PH figure 1.12, page 26 is the formula \[ w_i = \frac{1}{\text{Time}_i \times \sum_{j=1}^{n} \frac{1}{\text{Time}_j}} \]. Prove that the weights from this formula indeed do specify a job mix in which each job type runs for an equal amount of time on the normalizing machine. (PH unfortunately use \( T_i \) and \( \text{Time}_i \) for the same quantity.)

2. Read Chapter 2 of PH. As you read, write down every term that appears in italics, for example, “dynamic” (measurement). Then make sure you know what it means, either from knowledge you already have, careful reading of the sentences near the term, study of sections in PH you find by looking the term up in the index, other reference sources, or by bringing a question to class.

Finally, write a 1-2 sentence explanation of the term. For example, the term “dynamic”, when applied to an instruction or other event count, means “the number of times that instruction is executed or the other event actually occurs during the entire run of a program.” Dynamic frequencies are of course ratios to the dynamic count of one kind of event to the total dynamic count of all kinds of events. The opposite of “dynamic” is “static”: a static measurement pertains to program code without regard to execution, for example, the length of the program in bytes or the number of each type of instruction which you can count by scanning the assembly language listing.

3. PH page 120, exercise (2.3)

4. “Let the computer do the compiling,” (apologies to Yellow Pages): Code the calculations of exercise (2.3) in Java. Put your Java program in a file named Toy.java. Compile it with the command javac Toy.java. Successful compilation produces a Java Class File named Toy.class which contains a loadable program in Java Virtual Machine Language Class File Format. You can make the Java interpreter run the program with the command java Toy.

Use the command javap -c Toy to disassemble the class file. Figure out how the calculation of exercise (2.3) is expressed by the Java Virtual Machine instructions which belong to the stack family of instruction set architectures. Submit an annotated assembly language listing.

You can capture the output of javap or any other interactions with the shell by using the script command.

Here’s my toy Java program to help get you started if you are new to Java:

```java
public class Toy {
    public static void main(String[] args) {
        int A, B, C;
        A = 1;
        B = 2;
        C = A + B;
    }
}
```
System.out.println("A=");
System.out.println(A);
System.out.println(" B="+B + " C="+C + " \n");
}
}

5. You can give Postscript language commands and observe the text output of the Postscript interpreter gs when you start it (on eve.albany.edu or similar systems) with the command:
gs -DNODISPLAY
(If you have an X Window terminal, you can also observe the graphics output when you omit the -DNODISPLAY option and run graphics output commands.)
The Postscript language is stack oriented. See what happens when you type numbers, and
the operations:

```
psex
==
add, exch, div, sub, mul, dup, clear
```
A Postscript “variable” is a key associated with a value in the “current dictionary”. Try the sequences:

```
foo   (try to evaluate foo)
/fooo 3 def
foo psex
foo foo add ==
```
The Postscript command “quit” terminates the interpreter.

Use the Postscript interpreter to store values in variables named B and C and then perform the operations given in PH exercise 2.3, page 120.
Capture your work by doing it under the “script” command and submit a copy printed on paper.

6. Embed the code of PH exercise (2.6) on page 121 into a toy C program three times: First, make sure the arrays and C have static storage extent. Second, create an analogous loop with differently named variables but make sure they have automatic, i.e., stack storage extent. Finally, create a version in which they are allocated by malloc on in free-store memory. (The three pointers variables to hold the values returned by the three malloc calls should be static or automatic: Your choice.)

Compile the program into a SPARC assembly language twice using gcc: First with default (no) optimization and second with full optimization. Modify the program if necessary to force the compiler to actually generate the instructions indicated by the C code operations. (Simply code operations to print elements of A.)

What is the convention for the destination operand position (leftmost operand, rightmost operand, how does it vary) used by the Solaris SPARC assembler? It might be different from PH's dlx convention which always puts the destination operand leftmost.

Write and submit an annotated assembly language listing of the code for the loop generated by the compiler. Include an explanation of where each variable (index variable i, constant C and the two arrays. Then, find and write the solution to PH exercise (2.12 a only) of page 122.

7. PH exercise 2.8, page 121.