CS 352 – Prof. Mark **Assignment #1** **Due Jan 30, 2007**

Reminder: Homeworks are due at the *start* of class.

Problem #1

This is a trivial programming assignment; the purpose is to make sure that you have a UTCS unix computer account, know how to use the compilers on our system, and know how to use the UTCS "turnin" program. This problem should be done individually.

a) Choose a language, C, C++, or Java.
b) Write a "Hello World" program – that is, a program which prints the string "Hello World!" to the console. If you have made to CS 352, we hope this is easy.
c) Compile the program:
   for C: use `gcc` to compile `helloworld.c` into `helloworld` executable
   for C++: use `g++` to compile `helloworld.C` into `helloworld` executable
   for Java: use `javac` to compile `helloworld.java` into `helloworld.class`
d) Verify that your program works.
e) Use "turnin" to submit both the source code file and the executable file.
   e.g. for Java, you would type:

```
/p/bin/turnin –submit impjdi hw1 helloworld.java helloworld.class
```

Problems #2-#12:

P&H 4.1

P&H 4.2

P&H 4.3

P&H 4.10

P&H 4.12

P&H 4.15

P&H 4.17

P&H 4.19 (on CD, but attached for convenience)

P&H 4.20 (on CD, but attached for convenience)

P&H 4.21 (on CD, but attached for convenience)

P&H 4.22 (on CD, but attached for convenience)

In More Depth: Amdahl's Law

Amdahl's law is sometimes given in another form that yields the speedup. *Speedup* is the measure of how a computer performs after some enhancement relative to how it performed previously. Thus, if some feature yields a speedup ratio of 2, performance with the enhancement is twice what it was before the enhancement. Hence, we can write

$$\text{Speedup} = \frac{\text{Performance after improvement}}{\text{Performance before improvement}}$$

$$= \frac{\text{Execution time before improvement}}{\text{Execution time after improvement}}$$

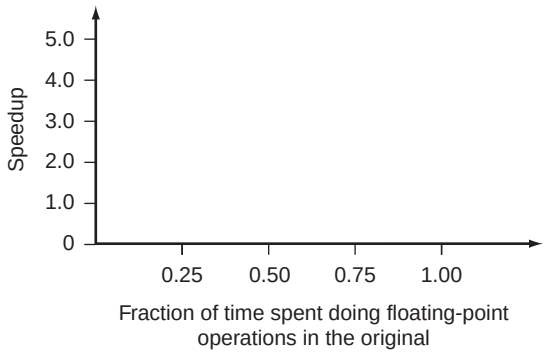
The earlier version of Amdahl's law was given as

$$\text{Execution time after improvement} = \frac{\text{Execution time affected by improvement}}{\text{Amount of improvement}} + \text{Execution time unaffected}$$

4.19 [5] <§4.3> Suppose we enhance a computer to make all floating-point instructions run five times faster. Let's look at how speedup behaves when we incorporate the faster floating-point hardware. If the execution time of some benchmark before the floating-point enhancement is 10 seconds, what will the speedup be if half of the 10 seconds is spent executing floating-point instructions?

4.20 [10] <§4.3> We are looking for a benchmark to show off the new floating-point unit described in Exercise , and we want the overall benchmark to show a speedup of 3. One benchmark we are considering runs for 100 seconds with the old floating-point hardware. How much of the initial execution time would floating-point instructions have to account for to show an overall speedup of 3 on this benchmark?

4.21 [10] <§4.3> Assuming that we enhance the floating-point unit as described in Exercise , plot the speedup obtained, versus the fraction of time in the original program spent doing floating-point operations, on a graph of the following form:



4.22 [5] <§4.3> You are going to enhance a computer, and there are two possible improvements: either make multiply instructions run four times faster than before, or make memory access instructions run two times faster than before. You repeatedly run a program that takes 100 seconds to execute. Of this time, 20% is used for multiplication, 50% for memory access instructions, and 30% for other tasks. What will the speedup be if you improve only multiplication? What will the speedup be if you improve only memory access? What will the speedup be if both improvements are made?

4.23 [5] <§4.3> You are going to change the program described in Exercise so that the percentages are not 20%, 50%, and 30% anymore. Assuming that none of the new percentages is 0, what sort of program would result in a tie (with regard to speedup) between the two individual improvements? Provide both a formula and some examples.

4.24 [20] <§4.3> Amdahl's law is often written in terms of overall speedup as a function of two variables: the size of the enhancement (or amount of improvement) and the fraction of the original execution time that the enhanced feature is being used. Derive this form of the equation from the two equations above.