

Homework 2

These questions cover part 7.1. of the textbook.

1. (10%) Describe the difference between:
 - a. Assembler and compiler.
 - b. Assembly language and high-level language.

2. (10%) What is the difference between:
 - a. Assembler directives and opcodes?
 - b. PUBLIC and EXTERN?

3. (10%) Why is that coding in assembly language result in smaller codes?

4. (10%) Which one should perform their job faster: compiler or assembler? Why?

5. (10%) The textbook says that it is common to have 10% of the program be responsible for 90% of the execution time. Explain. Can you find some examples? *Hint:* Remember the facts that many programs have repetitions.

6. (10%) Suppose you are asked to write some portion of the program into assembly language. Of course you would look into the portions that determine most of the execution time. How do you know such portions when you look into the program? *Hint:* See number 5 and 7.

7. A program is written under C++ to run under a 32-bit processor. It generates 20,000 bytes of executable codes. After profiling, the result indicates that 3,000 bytes of the program is executed 10 times more often than the others. The author then rewrite this portion of the code into assembly language. Now, that portion is four times smaller and is executed five times less often than the original. If all instructions are of the same length and each instruction takes 1 cycle time, then:
 - a. (10%) How many cycle times are saved?
 - b. (10%) How fast is the program in whole (in percentage)?

8. Suppose there is an algorithm, written in C, that takes n^2 cycle times for n data. This algorithm is then rewritten into assembly language, resulting 1000% speedup. There is a new algorithm, written in C, doing the same task. It takes only $n \log_2 n$ cycle times (rounded up) for n data.
 - a. (10%) Determine which one is better: the previous algorithm written in assembly language or the new algorithm in C? *Hint: Compare the value of n between algorithms*
 - b. (10%) Regarding your answer in part a, which one do you think it is better: Invent a new algorithm or just convert the existing algorithm into assembly language if the algorithm is then used for a huge amount of data? *Hint: Look into the n^2 and $n \log_2 n$.*