THE ASSEMBLY LANGUAGE LEVEL

Andrei León Salas

Date 10/11/20

Contents

1 For a certain program, 2% of the code accounts for 50% of the execution time. Compare the following three strategies with respect to programming time and execution time. Assume that it would take 100 man-months to write it in C, and that assembly code is 10 times slower to write and four times more efficient

1.1 Entire program in C

It would take 100 man-months to write it. It is going to be 4 times slower than assembly

1.2 Entire program in assembler

It would take 100 man-months to write it, It is going to be 4 times faster than c

1.3 First all in C, then the key 2% rewritten in assembler

It would take 100 man-months to write it. It is going to be 4 times slower than assembly. it is going to take 2 man-months to rewrite the 2% part of the code (for a total of 102 man-months) but it is going to be only 2 times slower than assembly, and 50% faster than in c

2 What is the difference between an instruction and a pseudoinstruction?

An instruction is a command to the assembler to execute something, like a representation of a machine instruction. A pseudoinstruction is a command to the assembler itself, also called assembler directives. It can be an instructions to tell the assembler to allocate some storage or to eject to a new page on the listing. In other words, a pseudoinstruction cannot be processed directly into machine code, as it represent something, but a instruction has a specific command equivalent to machine code

3 Programs often link to multiple DLLs. Would it not be more efficient just to put all the procedures in one big DLL and then link to it?

The lecture points out that one advantage is to save memory on disk and memory, so asking for all the dll's if i only needed one or 2, would be a waste of memory and disk. Also it makes it easy to update library procedures, even after the programs using them have been compiled and linked.

4 Can a DLL be mapped into two process' virtual address spaces at different virtual addresses? If so, what problems arise? Can they be solved? If not, what can be done to eliminate them?

yes it can be mapped into two process's virtual address at various virtual address. The problem is relocation problem, where if an object change their address, affects all the others. The fix is done via explicit linking, where the dll's makes the call during run time, and makes additional calls to get the addresses of procedures it needs.

5 One way to do (static) linking is as follows. Before scanning the library, the linker builds a list of procedures needed, that is, names defined as EXTERN in the modules being linked. Then the linker goes through the library linearly, extracting every procedure that is in the list of names needed. Does this scheme work? If not, why not and how can it be remedied?

No as the instructions would be in different positions, it is the relocation problem, and in the translation step, the assembler may not know what address the procedure is called in another function, this is the external reference problem. It is fixed as the linker merges the separate address spaces of the object modules into a single linear address space using these steps

1. It constructs a table of all the object modules and their lengths.
2. Based on this table, it assigns a base address to each object module.

3. It finds all the instructions that reference memory and adds to each a relocation constant equal to the starting address of its module.

4. It finds all the instructions that reference other procedures and inserts the address of these procedures in place

6 A linker reads five modules, whose lengths are 200, 800, 600, 500, and 700 words, respectively. If they are loaded in that order, what are the relocation constants?

0,200,1000,1600,2100