Homework 5 Solution

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 time more efficient.**
   - **i) Entire program in C**
   - **ii) Entire program in assembler.**
   - **iii) First all in C, then the key 2% in assembly.**

Solution:

i) 100 man month, execution time = X (let)

ii) 100 * 10 = 1000 man month, execution time = X/4

iii) 100 + 2*10 = 120 man month, execution time = 0.5X + (0.5/4)X = 0.625X.

2. **Do the considerations that hold for two pass assembler also hold for compilers?**
   - **i) Assume that the compilers produce object modules, not assembly code.**
   - **ii) Assume that the compilers produce symbolic assembly language.**

   Solution:

   a) If the compilers produce object modules, the compilers have the same problems as the assemblers. For example if the first statement is a branch to L, the compiler doesn't know the absolute address of L yet.

   b) If assembly code is produced, the compiler does not have to deal with the problem. It is pushed off onto the assembler.

3. **Can the following program be assembled in two passes? EQU is pseudo-instruction that equates the label to the expression in the operand field.**

```
P EQU Q
Q EQU R
R EQU S
S EQU 4
```

Solution:

No. It requires 4 passes.

6. **What is the difference between an instruction and pseudoinstruction?**
   Solution:
   A pseudo instruction is a command by the assembler to itself. It doesn't produce any executable code in the object file. On the other hand, an instruction produce executable code in the object file.

7. **What is the difference between instruction location counter and program counter if any? After all both keep track of the next instruction in a program.**
   Solution:
   Assembly counter used during the assembly time to point where to put the next instruction in the memory. On the other hand, PC keeps track of which instruction to execute next during the run time.

8. **Show the symbol tables after the following Pentium 4 statements have been encountered. The first statement is assigned to address 1000.**

a.	Everest:	POP BX	(1 byte)
b.	K2:	PUSH BP	(1 byte)
c.	WHITNEY:	MOV BP, SP	(2 bytes)
d.	MCKINLEY:	PUSH X	(3 bytes)
e.	FUJI:	PUSH SI	(1 byte)
f.	KIBO:	SUB SI, 300	(3 bytes)

   Solution:

a.	Everest:	POP BX	(1 byte)	1000
b.	K2:	PUSH BP	(1 byte)	1001
c.	WHITNEY:	MOV BP, SP	(2 bytes)	1002
d.	MCKINLEY:	PUSH X	(3 bytes)	1004
e.	FUJI:	PUSH SI	(1 byte)	1007
f.	KIBO:	SUB SI, 300	(3 bytes)	1008

10. **Show the steps needed to look up Berkeley using binary search on the following list: Ann Arbor, Berkeley, Cambridge, Eugene, Madison, New Haven, Palo Alto, Pasadena, Santa Cruz, Stony Brook, Westwood and Yellow Springs. When computing the middle element of a list with an even number of elements, use the element just after the middle index.**
    Solution:
    Step 1: List size = 12. So middle index = 7, the element is Palo Alto
    Berkeley < Palo Alto, so we will search in the left list
    Step 2: List size = 6, so middle index = 4, element is Eugene
    Berkeley < Eugene, so we will search in the left half
    Step 3: List size = 3, so middle index = 2, element is Berkeley. So the key is found.

11. **Compute the hash code for each of the following symbols by adding up the letters (A=1, B = 2, etc.) and taking the result module the hash table size. The hash table has 19 slots, numbered 0 to 18.**

    els, jan, jelle, maaike

    Does each of them generate unique hash code? If not, how to deal with the collision?

    Solution:

    els = (5 + 12 + 19) mod 19 = 17

    jan = (10 + 1 + 14) mod 19 = 6

    jelle = (10 + 5 + 12 + 12 + 5) mod 19 = 6

    maaike = (13 + 1 + 1 + 9 + 11 + 5) mod 19 = 2

    jan and jelle hash to the same value. We can maintain a linked list in the slots that contain all the elements. (e,g. 6 will contain both jan and jelle).

16. 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?

Solution: If any part of a program will not be used by any other program except this, it would be more efficient. But in practice, different parts of the program are used from other programs too. Suppose a binary search method might a part of a program X. If we have a dll for the binary search, then it can be also used from other program. Otherwise, only X can use it.

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

Solution: They are 0, 200, 1000, 1600, 2100.