METRO STATE
UNIVERSITY

ICS 232 Computer Organization & Architecture
Homework 12 - Chapter 8 - 10 points
Due Date: 11/15/2023

Name: Key

Note: Please post your homework to ICS232 D2L on or before the due date.

Chapter 8 – System Software

Essential Terms and Concepts

12. What is meant by preemptive scheduling?

Preemptive scheduling does not allow one job to use all CPU resources. After a time slot is consumed a new job is scheduled for execution. Scheduling of the CPU may be also adjusted by job priorities.

16. Besides process management what are the other two important functions of an operating system?

Resource management
Security and protection

24. Describe the purpose of each phase of a compiler.

Lexical analysis – split source code into symbols
Syntax analysis – verify correct syntax
Semantic analysis – verify semantics
Intermediate code generator – generate independent code (three address code)
Code optimizer – optimize code
Code generator – generate machine code

25. How does an interpreter differ from a compiler.

A compiler generates machine code that can be directly executed by the CPU. An interpreter uses an intermediate language and executes this intermediate language. This is usually much slower than direct execution.

METRO STATE
UNIVERSITY

ICS 232 Computer Organization & Architecture
Homework 12 - Chapter 8 - 10 points
Due Date: 11/15/2023

Exercises

4. What is the difference between multiprogramming and multiprocessing? Multiprogramming and multithreading?

Multiprogramming which is also called multitasking means having multiple programs (processes) running on one processor, whereas multiprocessing means having multiple processors. Multiprogramming typically refers to multiple processes from multiple users, whereas multithreading generally refers to multiple threads from one user process.

9. Discuss the advantages and disadvantages of dynamic linking.

Dynamic linking saves disk space, results in fewer system errors, and allows for code sharing. However, dynamic linking can cause load-time delays, and if dynamic link library routines are changed, others using modified libraries could end up with difficult bugs to track down.

10. What problems does an assembler have to overcome in order to produce complete binary code in one pass over the source file? How would code written for a one-pass assembler be different from code written for a two-pass assembler?

The key problem that has to be addressed is forward referencing (an address is referenced before it is defined). One solution would be to require all data values to be declared before being used. This won't help with forward references to labels for such things as jumps. Another would be to use some sort of list (or other mechanism) to handle forward references for jumps and subroutine calls.

13. What are the advantages of using a compiled language over an interpreted one? Under what circumstances would you choose to use an interpreted language?

Interpreters do not optimize as nicely as compilers do. In addition, compiled code is generally faster than interpreted code. Interpreted languages are also less in control of the hardware as a general rule. Compilers are better in production environments (for many reasons, not the least of which is that with interpreted languages, errors may go unnoticed, since an error is not detected unless that particular statement is actually executed), situations in which the programs are repetitive (will be run many times), and situations in which execution speed and size matter. So, if you want to save memory space, increase speed, and have enhanced control over the hardware, use a compiled language. Interpreters are useful in a development and testing phase (since large programs can take a long time to compile, simple changes in compiled languages can result in long waits, whereas in with interpreted languages programs can easily be debugged and modified), for programs that are run once and then discarded, or where the execution speed is secondary to another factor better handled by an interpreted language (for example, communication costs versus execution speed). Interpreted code is often more portable (as long as the correct machine-specific interpreter is installed, the same source code can be run on different platforms). In addition, everyone can read and modify the source code for interpreted programs, which may be a plus or a minus. Interpreted languages are typically used for scripting.

15. Why is the execution environment of a Java class called a virtual machine? How does this virtual machine compare to a real machine running code written in C?

The JVM is a virtual computer, implemented in software on top of a "real" hardware platform. Code written in C on a real machine is first compiled to machine language, and then this machine language is run on the real machine. Java, on the other hand, is first compiled to bytecode; this bytecode is then run on the virtual machine. Java bytecode can be thought of as the machine language for the JVM, but it is not real machine language.

17. We stated that only one method at a time can be active within each thread running in the JVM. Why do you think that this is the case?

When a thread invokes a method, it creates a memory frame for the method, where part of this frame is used for the method's local variables, and another part for its private stack. After the thread has defined the method stack, it pushes the method's parameters and points its program counter to the first executable statement of the method. If more than one method were active, there would be multiple copies of these entities, which would imply synchronization and consistency issues.

Prepare for next class by reading Chapter 9 – Alternative Architectures

Continue working on Project 2

Continue working on Your Group Project