Ryan Somma

Week 7 Assignment
Using APA format, give short answers to the following:

1. What does a compiler do when it encounters data declarations in a source code file? Data (manipulation) operations? Control structures?

When the compiler encounters a data declaration, it allocates sufficient memory to store the data item based on the data type and number of bytes used to represent the data type for the target CPU. For a data operation, the compiler translates the data operation instructions into an equivalent sequence of data movement and data transformation instructions for the target CPU. When the compiler encounters a control structure, it must keep track of where CPU instructions for each source code instruction are located in memory, and retrieve the proper memory address based on the condition.

2. Compare and contrast the execution of compiled programs to interpreted programs in terms of CPU and memory utilization.

Interpreted programs require more memory and CPU instructions in execution. For instance, Interpreted programs require memory for the interpreter, source code, and executable code, while a compiled program only requires memory for the executable code. Interpreted programs also consume CPU instructions for translation operations, library linking, and the application program, while compiled programs only need CPU instructions for the application program.

3. Compare and contrast the error detection and correction facilities of interpreters and compilers.

Because the symbol table and program source code are always available to the interpreter at runtime, interpreted languages can report the most recent source code line translated along with the error, making them easier to debug than compiled languages. Compiled programs, in contrast, require a symbolic debugger, which tests executable programs using a symbol table, memory map, and source code files to track memory addresses to their source code statements and variables. Debugging compiled programs involves incorporating the symbolic debugger in a debugging version of the software, while an interpreted program often has access to symbolic debugging capabilities via the interpreter.

4. Describe the functions of the kernel, service, and command layers of the operating system.

The command layer of the operating system is the user's interface to the OS, either textual or graphical. The command layer accepts keyboard or mouse clicks from the user, which are translated into operating system commands. The service layer of the OS contains the set of functions that are executed by the application programs and the

command layer. The kernel manages resources and directly interacts with computer hardware using device drivers, allocating and interacting with these resources for service layer functions.

5. Describe the operation of virtual memory management.

Virtual memory management is a method operating systems use to minimize the amount of process code and data stored in memory at any one time, which frees memory for other processes. The method divides a program into partitions between one and four kilobytes and memory into portions of the same size. During execution, processes are given one or more portions of memory, with the rest of the program held in secondary storage. As portions of code are executed, more code is loaded from secondary storage into memory for execution. As code may reference memory addresses of programming code not currently loaded into memory, virtual memory management keeps page tables to store information about page locations. Because page sizes are fixed, memory references can be converted to corresponding page number and offset with the page.

Virtual memory can get loaded down with too many swapping operations, either from insufficient memory or poor programming, in a state known as thrashing (Hyde, 2001).

Randall Hyde, *The Art of Assembly*, Webster, 2001. Retrieved from the University of California website on February 25, 2009: http://webster.cs.ucr.edu/AoA/Windows/HTML/MemoryArchitecturea3.html