**System Software & Programming Techniques**

**Computer Systems: Software, Code, & Memory**

**By Odiscious Dozier**

**Trident University**

INTRODUCTION

Often, many assumptions are made when computing. Things like powering up your computer and the expectation that there will be colorful images with convenient tiles or icons that can be double clicked, to open items of choice. With these type of assumptions, it is difficult to discern what the computer system is and how to make best uses of the software that it comes with or is later purchased.

This discussion is designed to help obtain a better understanding of the overall and inner workings of a computer system by answering a few questions. What are some of the differences in programs and how they can be used in conjunction with other programs, users, and the computer system itself? What does BIOS do and what is it used for? What are some different programming languages and how does their level-identification define how they are received/interpreted by a computer?

SYSTEM SOFTWARE

Computers are machines that are limited to performing the commands they are given which specify operations to be completed on given data. These commands are known as, *instructions*. When a programmer writes a sequence of instructions, she creates what is known as a *program*. A set of programs, written for a computer, is known as, *software* (ROM, 2007).

The term *software* can mean either, system software or a user’s program.  *System software* is the term given to the software that is needed to execute a user’s program. A user’s program is a program that is prepared by a programmer to solve a problem or to accomplish a task. While the user’s program is stored in RAM, programs that rarely or never change are known as *firmware* and are stored in non-volatile memory.

OPERATING SYSTEM

“An operating system is a collection of programs that controls the overall operation of a computer” (Ram, 2007). It allows users to navigate and access directories on their machines by using a mouse and clicking on various images. Additionally, it enables users to perform tasks like, manage memory, read from files, write to files, delete files, store files, manage input and output devices, track system usage, connect to the internet via visual displays, and execute programs.

Windows, Mac iOS, and Linux are very popular operating systems. Linux is an open source product and as such is free to use. Windows and Mac iOS are long term market competitors.

"There are about 60,000 viruses known for Windows, 40 or so for the Macintosh, about 5 for commercial Unix versions, and perhaps 40 for Linux. Most of the Windows viruses are not important, but many hundreds have caused widespread damage. Two or three of the Macintosh viruses were widespread enough to be of importance. None of the Unix or Linux viruses became widespread - most were confined to the laboratory" (Granneman, 2003).

FIRMWARE

Firmware is software that is stored in non-volatile memory; as, it rarely changes or cannot be changed at all – depending on the manufacturer of a given device. “Some system programs that manage the computer’s operations are stored in ROM as firmware” (Ram, 2007). They perform fundamental types of supervisory and support work that programmers need and are known as subroutines, but referred to as Basic Input/Output Services or BIOS or ROM-BIOS.

BIOS is what boots up a given machine. Typically, you can adjust the hardware settings of the machine through the BIOS during the boot-up sequence. This is typically done using a Setup Utility which acts as a user interface.

UTILITY PROGRAMS

“The software that are helpful to users in developing, writing, debugging and documenting programs are referred to as utility programs” (Ram, 2007). There are two types of utility programs, file management utilities and program development utilities.

File management utilities are tools that are built to help the user perform tasks such as, cutting and pasting documents from one file path to another, deleting text from a file, and printing documents or images. Some common programs are, notepad, Microsoft Word, and File Explorer.

Program development utilities are tools that are built to help the user, architect, design, code, build, debug, test, and deploy programs. Some common programs are, Microsoft’s Visual Studios, Jet Brain’s IntelliJ, and Eclipse.

PROGRAMMING LANGUAGES

ASSEMBLY LANGUAGES

Machine language is written and understood with the use of 1s and 0s. It is a tedious language to write, by a developer, and as such it is error prone. To address the issues of speed and accuracy in program development, assembly languages have been created and put into practice. Assembly languages are machine specific (cannot be used on other machines) and use easily memorable symbols like, ADD, SUB, and CMP, which stand for, addition, subtraction, and comparison, successively and are known as mnemonics. Assembly languages advantages over machine language include, being better for human understanding, the use of mnemonics for operation codes, the use of symbolic names for operand addresses, they’re easier to write, they’re easier to modify, and they’re easier to debug.

Assembly languages must be converted from mnemonics to 1s and 0s before the computer can interpret/understand the instructions given it. This interpretation is done using an assembler.

There are two different types of assemblers, the one-pass assembler and the two-pass assembler. As the names infer, each assembler is read a number of times in accordance with their names. One-pass assemblers come equipped with the ability to assign addresses to the labels that are used. The two-pass assembler, collects all the labels on the first read, assigns addresses to the labels, and counts their positions from the starting address. The machine code is produced for each instruction and addresses are assigned to each on the second pass.

HIGH-LEVEL LANGUAGES

Because high-level languages are not computer/machine dependent, programs written in these languages are not subject to the same constraints assembly languages are. They can be executed on any machine with a compiler. Some examples of high-level programming languages are, C++, Java, and C#.

In 1983 Bjarne Stroustrup, a Danish computer scientist, created the C++ language (History of C, 2015). It’s a superset of the C language, but not the same.

In 1991, James Gosling led a small group of engineers at Sun Microsystems known as the green team (The History of Java Technology, n.d.). Working ‘round the clock the team developed a new programming language called, Java.

In 2000, Borland Hejlsberg, about 4 years after joining Microsoft, built a team and operated as the lead architect developing the language C# (Hashimi, 2014). C# is a part of the .NET Framework and has been a long time market competitor with Java for a majority of software engineers.

ASSEMLBY VS. HIGH-LEVEL LANGUAGES

The primary advantage of assembly languages over high-level languages is speed. It is faster for assembly languages to compute and run code to produce desired results.

“The basic problem of an assembly language is that it is more closely related to the structure of a computer than to the performance of the task which is to be performed” (Ram, 2007). The developer ends up spending more time focusing on considering instruction sequences and manipulating registers rather than solving problems. For this reason, high-level or procedure oriented languages have been created which moves the developer from computer oriented development to object oriented development.

There are several additional disadvantages to assembly languages. Being that these types of languages are machine oriented, they require the developer to be very skilled at the structure of the computer that the code is being written on or for. The software engineer will also need to have an in-depth knowledge of, the structure of the computer, the registers and instruction set, and connections of ports to the peripherals – to name a few. Additionally, programs written in assembly language for one machine cannot be used on other machines because, each processor has its own instruction set; hence, its own assembly language.

REFERENCES

Granneman, S. (2003, October 6). Linux vs. Windows Viruses • The Register. Retrieved September 9, 2015, from <http://www.theregister.co.uk/2003/10/06/linux_vs_windows_viruses/>

Hashimi, J. (2014, July 30). History of C# language | JAMSHID HASHIMI. Retrieved September 11, 2015, from <http://jamshidhashimi.com/2014/07/30/history-of-csharp-language/>

History of C. (2015). Retrieved September 11, 2015, from <http://www.cplusplus.com/info/history/>

Linux Does What Win Don't! (2013, June 12). Retrieved September 9, 2015, from <https://www.youtube.com/watch?v=HD6nqQrJx78>

Munteanu, I. (2014, March 21). Simple Questions: What is the BIOS & What Does It Do? Retrieved September 9, 2015, from <http://www.7tutorials.com/simple-questions-what-bios-what-does-it-> do

Ram, B. (2007). Chapter 8: System Software and Programming Techniques. Computer Fundamentals, Architecture and Organization, New Age International, New Delhi

The History of Java Technology. (n.d.). Retrieved September 11, 2015, from <http://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html>