Evolutions of Operating Systems

Nitant Suhgaiya

Computer Science

University of Regina

September 10, 2020

Abstract

This paper describes a change in the operating systems over the time between three operating systems. Here, I gathered some of the operating systems from the very first operating systems to the current operating systems, and how the companies updated the level and their architecture to satisfy the needs of the customers. I am going to talk about the very first operating system of the world based on CLI, which led a step ahead in the computer field, and the company named Xerox PARC released their personal computer Alto in which they used Alto OS, which was the first operating system based on GUI. With both these operating systems the revolution was about to come and in the 1980's two companies made their debut in the computer world and they changed history. Both the company introduced its operating system, "Windows 1.0" and "Apple Lisa Os" respectively.

Introduction

An operating system (OS) is system software that manages, Computer Hardware and Software resources, and provides common services for computer programs. Usually, resources include the central processing unit (CPU), the memory of the computer (RAM), storage of the files, input/output (I/O) devices, and network-based connection. It has a special ability to run indefinitely so it terminates only when the computer is turned off. The first digital computers had no operating systems, they can run only one program at a time. The first operating system was developed after 1955. The initial operating systems only provided basic I/O operations such as controlling punch card readers and printers. 1960's was the golden decade in the technology industry when John Backus first introduced the concept of “Time-Sharing” in 1954 in the summer session at MIT, and later in December 1958 by W. F. Bauer published a paper in which he wrote that "The computers would handle several problems simultaneously." It was a new era for technology after these statements and concepts. Implementing such a system was quite tough at that time, however, in 1959, John McCarthy implemented the first time-sharing

project at MIT and was implemented on IBM 709 (Made by IBM in January 1954 and was the first computer that can compute floating-point numbers.)

(IBM 709 front panel)

Then the operating system made by IBM in 1964 called IBM System/360 which was the first major operating system. It was considered as the biggest change in the history of computers, and also the most complex software package of the 1960s. It was based on the command line interface (CLI). After achieving the big success in the first operating system more and more people came with their new ideas to bring a completely new and innovative concept to make the operating system better than ever. To follow this chain company named Xerox released the first graphical user interface (GUI) based operating system in mid 1970's around 1973. It was a huge success for them and the industry to have a graphical user interface (GUI) on a computer, that set new standards for the industry and also new challenges for the other competitors.

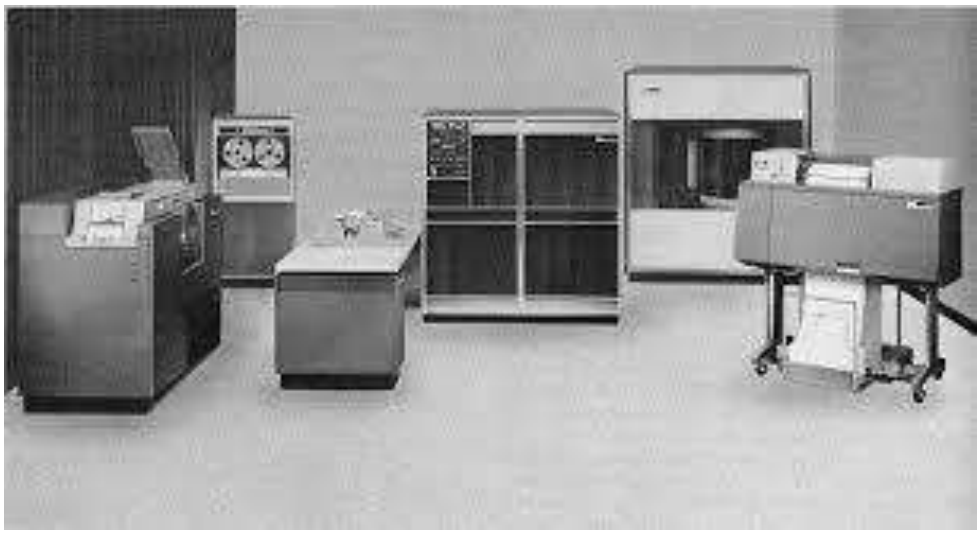
To support the chain and advance the technology two companies started their research and projects to make their operating system, Apple and Microsoft Windows. Microsoft came up with its first operating system "Windows 1.0" and a year after Apple came up with its operating system Alto OS in their personal Computer Alto. All of these operating systems are quite amazing and innovative on their own especially considering the time in which they came. Here I am going to write about all three of these operating system about their Architecture, Processing Management, Memory Management, and the components that used in the personal computers and operating systems.

How they evolve with time and trying to be better than others over time and it helped us a lot as well in the technological field. How they improved the architecture, they also improved the hard drive architecture and size overtime to speed up the processes, considering the size became more compact over time. How they learned from their mistakes from the past about scheduling the processes and they applied new algorithms to resolve the errors and to protect the system as well as the performance.

IBM's System/360

The Beginning

1950's was the age of Data processing. There were already plenty of Data Processing machines in existence at that time, but nobody had considered as a programmed device. However, the idea of computers came out and once the idea had sprouted, demand for computers as data processing machines flourished, and new machines, such as IBM 1401, were built to meet this demand.

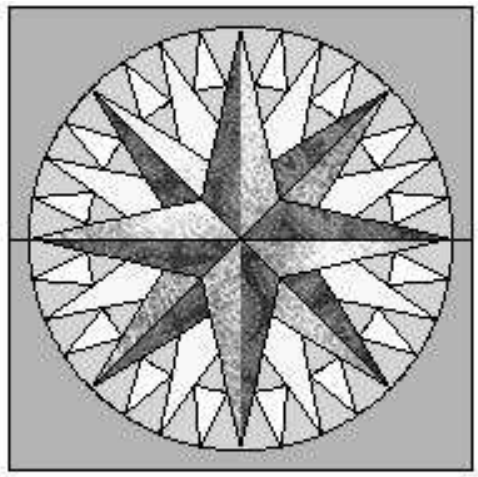


(IBM 1401 Data Processing System)

These early "Data Processors" proved that they were primordially high performers than the previous versions. With the previous experience of 1401, IBM decided to implement a completely new architecture primarily designed for both data processing and to be compatible with a wide range of performance levels. One of the key factors of their architecture was the ability to move data. It was so powerful that even the earliest System/360's could move data in and out of main storage at a speed comparable to today's workstations. The System/360 was designed, such that it can be also implemented in the future. The architecture of the system could have plenty amount of storage for extra storage capacity, such as the main storage of the architecture can address up to as high as 16 Megabytes and It could also have up to 7 data channels. With this big idea and concept IBM is going

to achieve new heights. The architecture of the System/360 was so fast, accurate, and compact (compare to previous devices) that changed the history of the whole decade, and considered as the golden decade of the century for computers and technology industries.

System/360 was announced by IBM on April 7, 1964, and was first delivered to customers in 1965. Dr. Fredrick Brooks from North Carolina was in charge of IBM's System/360 project, whereas Dr. Gene Amdahl was the head of the architecture of the System/360. The System/360 project was so huge that it adhibited over 60,000 people to complete this project. The project was so huge and unique that IBM cost over $5 billion, and it was the second biggest project after the Apollo Moon program at that time.



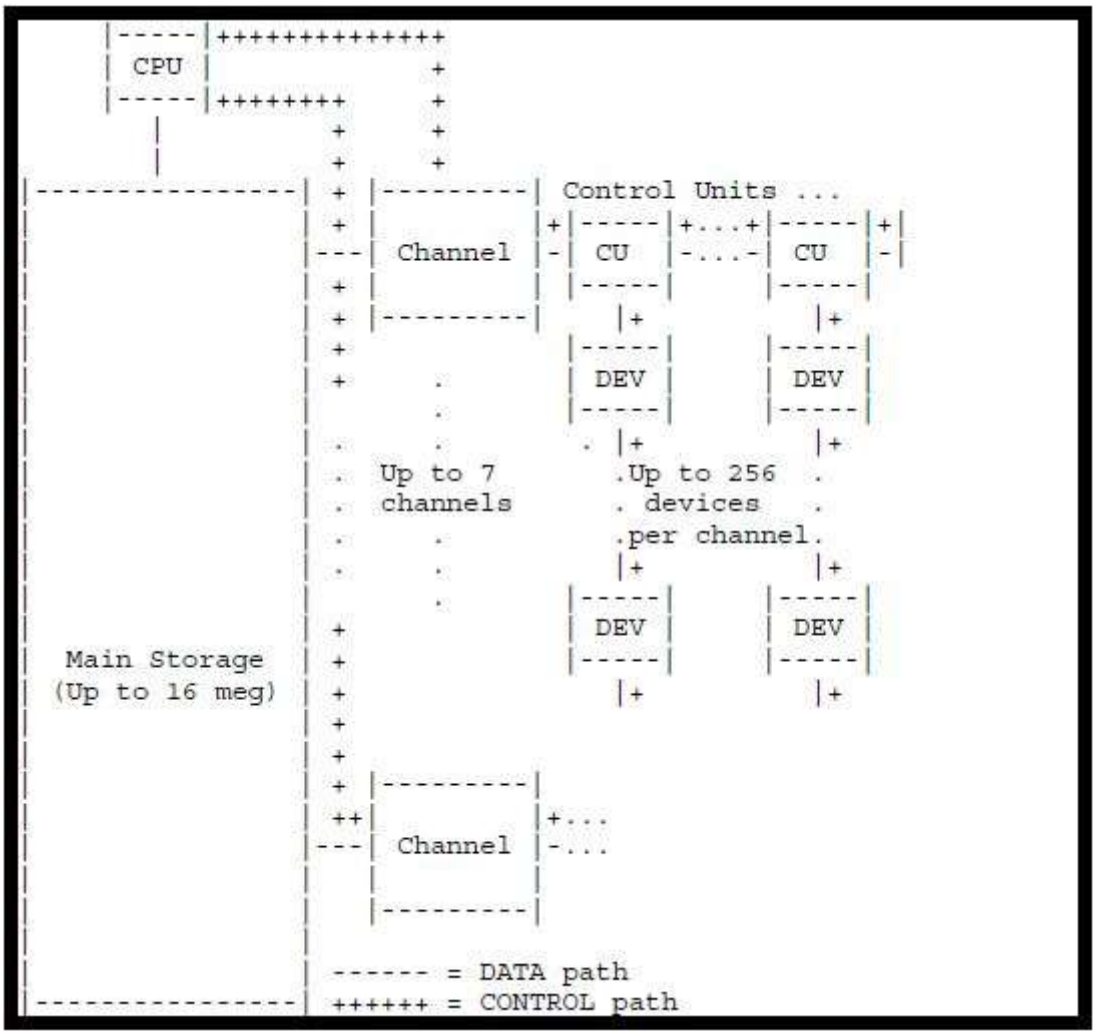
(The official IBM system/360 compass logo)

The logo defines "These are general-purpose computers usable by all industries, academic institutions, and government departments alike, and not tailored to one niche. They cover 360 degrees of the computing compass."

Architecture

IBM’s System/360 has five major components in it :

1. One main huge storage, as big as up to 16 Megabytes.
2. It could have one and even two Central Processors, with these it could do multitasking.
3. It also had channels in it, for extra capacity and performance it could be extended up to 7 channels.
4. Control units, which are connected to the channels.
5. I/O devices to connect control units.



(Basic Components of a System/360)

Main Storage

Main storage is a part of the architecture. It was housed with CPU in the smaller models and housed separately in the larger ones. It provides the system with directly-addressable, and fast-access storage of data. Both data and programs must be loaded into main storage by any I/O devices before they can be processed. Main storage is an array of data locations, where each location contains eight binary bits, each 8-bit location is known as a byte, and each of them has their unique address, starting from zero to the last byte of the main storage. However, there are some restrictions on data to store, which depends on whether a data field is variable-length or fixed-length. Fixed-length fields (halfwords, words, and doublewords) must be located in the main storage on integral boundaries. To protect the system from any damage by the application programs, the Main storage is divided into 2048-byte blocks, where each block can be assigned by a protection key. If a program lies inside the blocks of the main storage, and have the same key as the current execution key then only it's allowed to modify data on the system. It has a special ability with the supervisor software in charge of the machine can assign and change the key dynamically using privileged instructions.

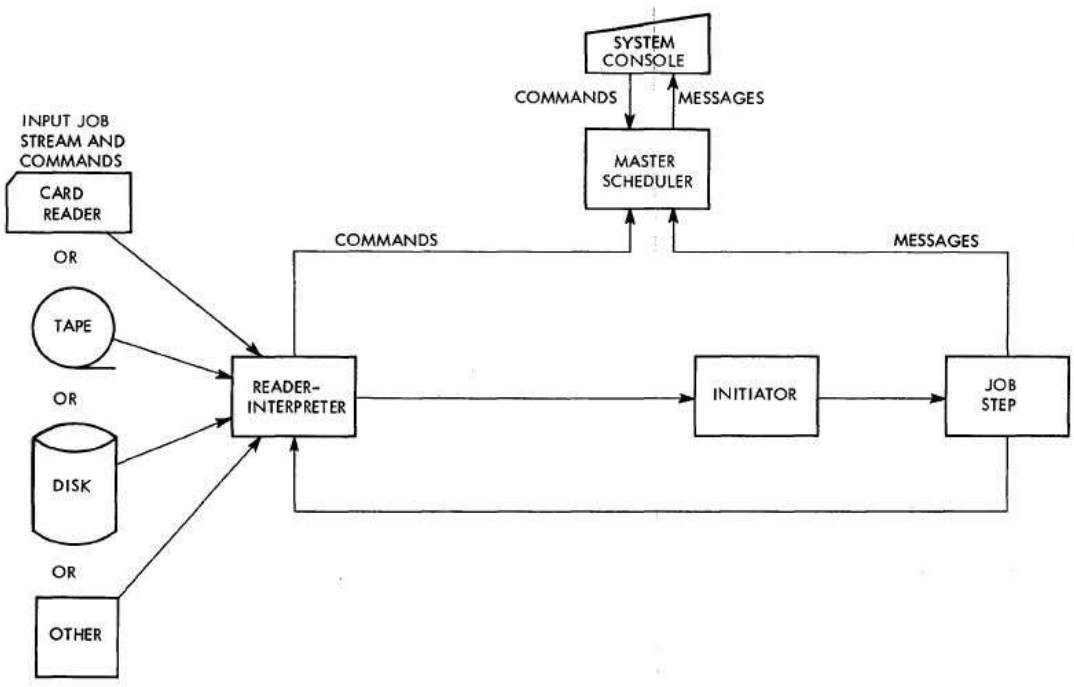
The Central Processing Unit

The central processing unit (CPU) is the controlling center of System/360. It provides various facilities for :

- Addressing main storage.
- Fetching and storing data.
- Arithmetic and logical processing data.
- Executing instruction in the desired sequence.
- Initiating communication between main storage and input/output (I/O) devices.

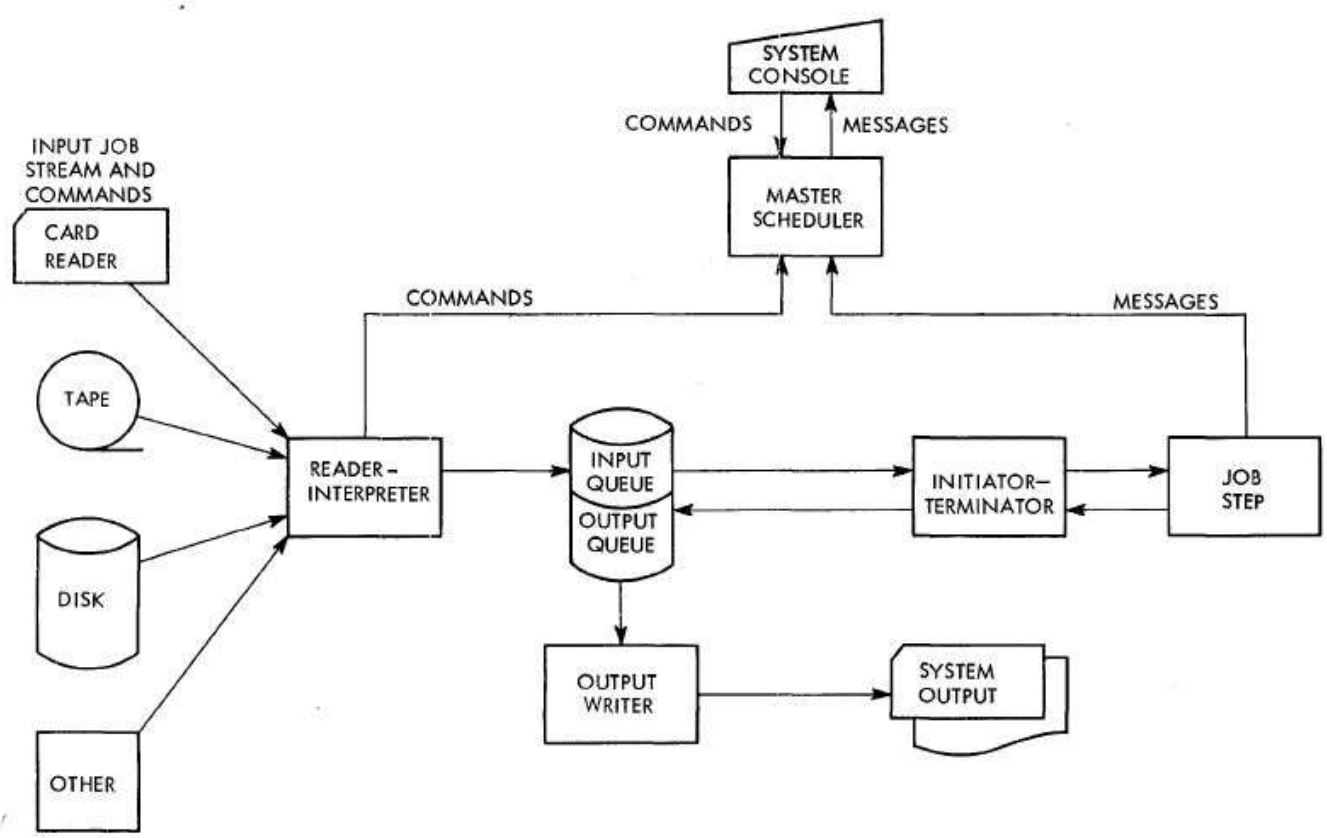
The CPU also provides 16 general registers and 4 floating-point registers, but these are accessible to the programmer and are capable of receiving data, holding it, permitting it to be operated on, and transferring it. The general registers in the CPU are used primarily for fixed-point, logic, and addressing operations, whereas the floating-point registers are used only for floating-point arithmetic.

Job scheduling on this system was done so precisely that the job scheduler allocates the input/output (I/O) units needed and then requests the supervisor program to initiate the execution of the programs specified in the control statements. One of these is the Sequential Scheduling System (SSS), this consists of a reader/ interpreter, an initiator, and a master schedular. The reader/interpreter reads in job control statements for a single job step. The initiator then allocates the required input/output (I/O) devices, notifies the operator of volumes to be mounted, and when all required volumes are mounted, requests the supervisor to execute the named program. The master scheduler program carries out operator commands that control or inquire about system functions. It also relays messages to the operator, such as the volume mounting instructions. The variety of commands depends on the control program configurations.



(Sequential Scheduling System (SSS))

The second type is the Priority Scheduling System (PSS), in which jobs are not executed as soon as they are encountered in an input job stream. Instead, a summary of the control information associated with each job is placed on a direct-access device from which it may later be selected. The set of summary information is called the input work queue. More than one input stream can feed this queue.



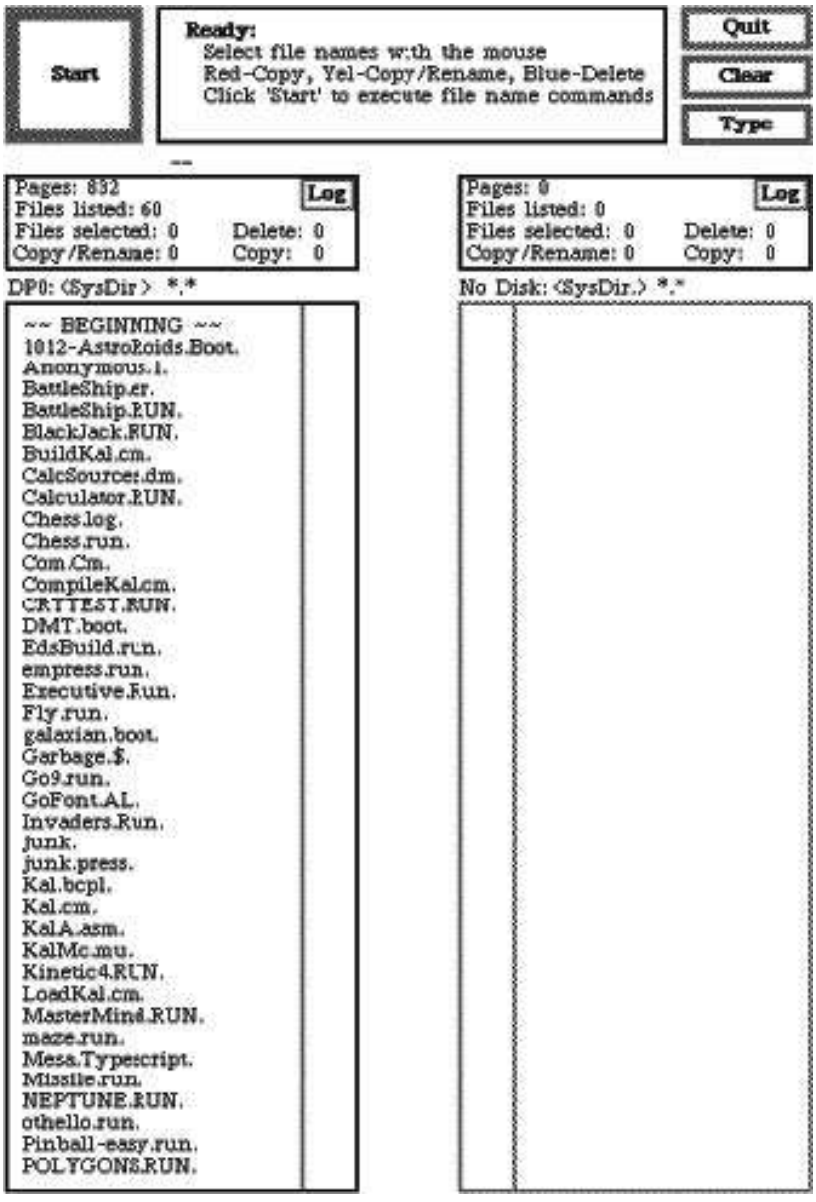
(Priority Scheduling System (PSS))

The use of the input work queue permits greater flexibility in the sequence in which jobs are selected for execution. The delays are caused by the mounting and demounting of input/output (I/O) volumes. The initiator/terminator can look ahead to future job steps (within the same job) and issue volume-mounting instructions to the operator to mount volumes for them in advance. Jobs that can be run without having the designated as non-setup. An optional non-setup padding feature lets the initiator/terminator ignore the usual priorities whenever a job step is delayed waiting for volumes to be mounted. It uses such a delay to select a step from a non- setup job in the input work queue, and run it. It also has other job management options :

- Job Account Log.
- Multi job Initiation.
- Remote Stacked Job Processing.

Xerox Alto

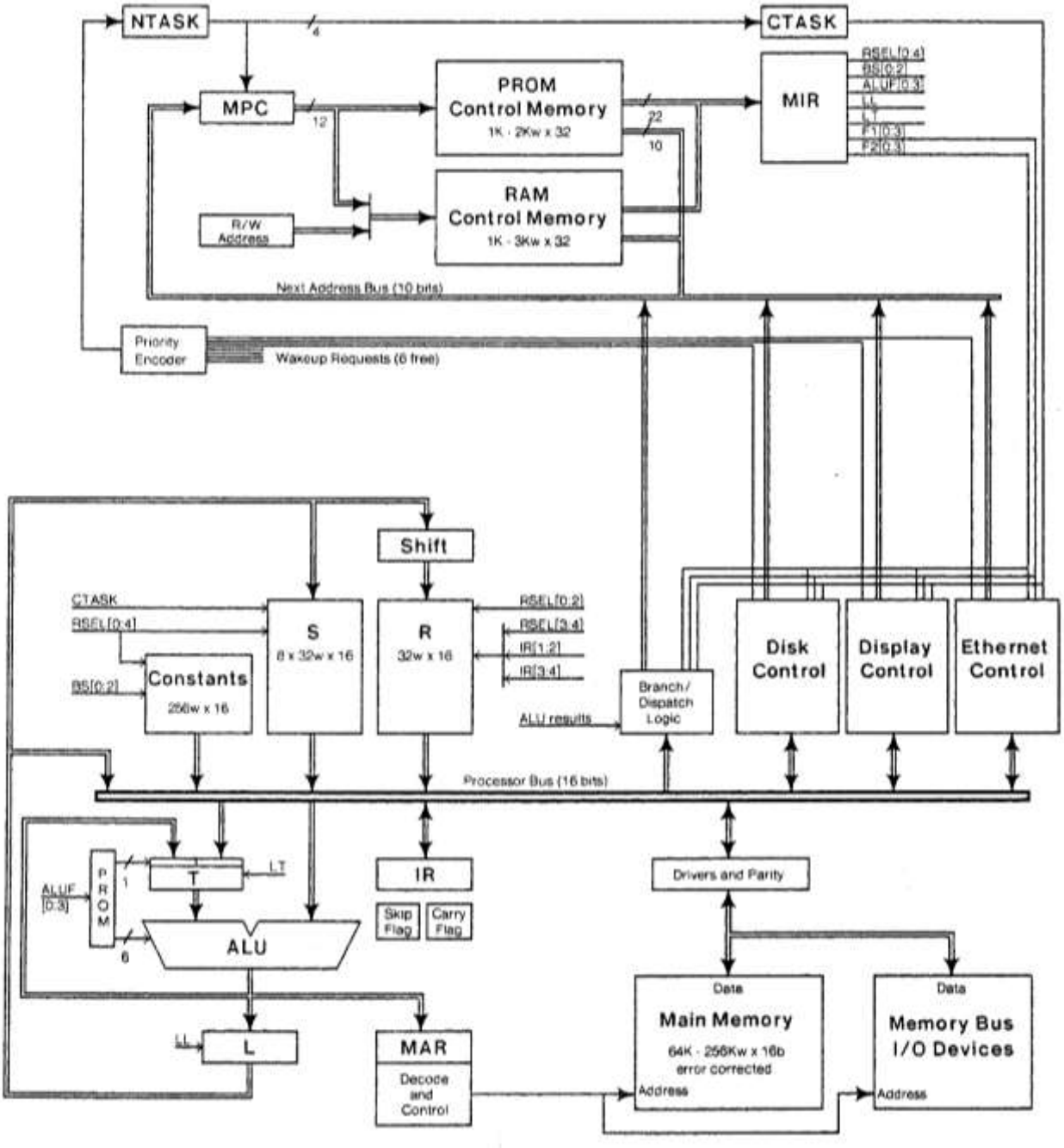
As the 1960's decade went on people came up with new ideas and concepts. They tried new things and they failed as well. Finally, Jacob E. Goldman came up with an idea and he started a company called Xerox PARC (Xerox Corporation Palo Alto Research Center), this division was established in 1969. The company's main motive was to explore new information technologies that were not necessarily related to the company's core. Many innovations in the computer design were developed by PARC researchers, Alto was one of them. The concept of the Alto was first introduced by Charles P. Thacker, he led the project of Alto, and on March 1, 1973, they made the history by releasing the first Graphical User Interface (GUI) based Operating System named Alto Operating System, implemented in theirs first Xerox Alto computer. Alto used a bitmap display in which everything on the screen was, in effect, a picture in which programs were shown in windows that could be manipulated by using the mouse. They finally made a computer-based on GUI and they knew that the world is going to change from now, and they were true after a decade in 1980's first Windows Operating System came out and the rest is history. At first, Alto cost them $12,000, and it would be too expensive to market to the private and small-business.



(The Alto Neptune File Management Program)

Microprocessor

A principal design goal in this system was to achieve the simplest structure adequate for the required task. As a result, the central portion of the processor contains very little application-specific logic and no specialized data paths. The entire system is synchronous with a clock interval of 170 nanoseconds. Microinstruction requires one cycle for its execution. A second design goal was to minimize the amount of hardware in the input/output (I/O) controllers. This is achieved by doing most of the processing associated with input/output (I/O) transfers with microprograms. To allow devices to proceed in parallel with each other and with CPU activity, a control structure was devised which allows the microprocessor to be shared among up to 16 fixed-priority tasks. Switching between tasks requires very little overhead, and occurs typically every few microseconds.



(Alto's CPU)

Control of the Alto microprocessor is shared among 16 "tasks" arranged in priority order. The tasks are numbered 0 to 15: 0 is the lowest priority task and 15 is the highest. The lowest priority task is the emulator task which fetches instructions and executes them. The only state saved for each task is a "microprogram counter," MPC. The current task number, saved in the current task register, addresses a 16 by l2 MPC RAM. The result is an MPC for the current task; it is used to address a 1 K by 32-bit microinstruction memory (MI ROM). The microinstruction memory produces an instruction and the address of its successor NEXT[0 – 9], This successor address may be modified by merging bits into it under control of the function fields of the current microinstruction. This limited branching capability makes coding more difficult than with a more general scheme, but not seriously so, as examples of microcode demonstrate. The amount of memory available for microinstructions is often extended by an additional l K of control memory implemented with RAM. Because the MPC RAM produces 12 bits, enough are available (11) to address both the microinstruction ROM and RAM. The microinstruction RAM may be loaded or read by special CPU instructions, and provisions exist for causing any of the 16 tasks to execute instructions from it (see section 8). At the end of each cycle, the microinstruction register (MIR) and the MPC are loaded, and the cycle repeats. There is only one phase of the system clock. It is true during the last 25 ns. Of every instruction.

The basic OS

The Alto Operating System (OS), a program that provides a set of basic facilities for control and communication with the Alto, is written in BCPL ("Basic Combined Programming Language"), a language similar to C. Most programs, BCPL or otherwise, run under the direction of the Alto OS. Since the address space of an Alto is small, a technique called a "Junta" is used to permit BCPL programs to shed unwanted sections of the Alto OS during execution. If those positions are needed later, they may be restored by performing a "Counterjunta". Alto has a 16- bit processor, 64k words of 800 ns memory, and or two moving-head disk drives, each of which can store 2.5 megabytes on a single removable pack and can transfer 64k words in about one second. As it was written in BCPL this language is considered to be one of the standard ways of programming the machine. The compiler generates ordinary machine instructions and uses no runtime support routines except for a small body of code that extends the instruction set slightly.

One BCPL program that runs on top of the operating system is called the Alto Executive. This Program speaks to the user directly and makes facilities available for file manipulation and program execution. An interesting feature of the Executive is that of escape expansion and file-name completion. Typing a partial file or program name followed by an escape, in the same fashion that an ESC (escape) or ALT (alternate mode) might be sent from an ASCII (American Standard Code for Information Interchange) terminal, causes the Executive to complete the typing of the name on the screen. This allows a programmer to name a file in a descriptive manner (such as a GatewayInformation.press), rather than typing in a long name. The Executive program will recognize it as soon as it has read enough characters to determine the file uniquely. By typing a question mark instead of an escape, the Executive will list all file names that are valid matches for the string types thus far.

Communication

A key objective of most operating systems is to foster communication between separate programs, often written in different programming languages or environments. But how can communication be provided by an open operating system that allows the programmer to reject all facilities of the system? The most conservative solution is to allow communication only through disk files since the file structure must be observed by all programs. Ordinary disk files can be used in this way to pass data from one program to another. The disk file structure must also serve as a way to invoke an arbitrary program, that is, to "transfer control" from one program to another. Because of the openness of the operating system, the called and calling programs may have little or nothing in common. The inter-program communication mechanism has found many uses. Examples are;

- Bootstrapping: A hardware bootstrap button causes the state of the machine to be restored from a disk file page is kept at a fixed location on the disk.
- Debugging: When a breakpoint is encountered, the state of the machine is written on a disk file, and the machine state is restored from a file that contains the debugger.

Lisa OS

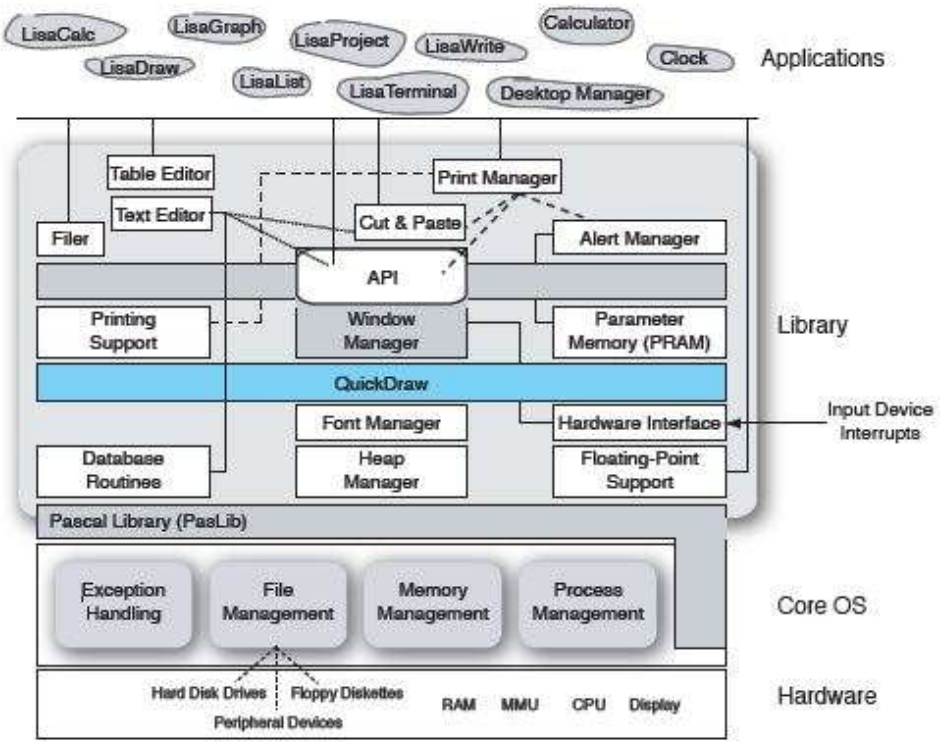
1980's was the decade in which the world saw the emerging technologies inspired by the previous projects and also to improve the previous generations. In this decade two big companies (comparing to the current date) started a revolution in the computer industry. After it, the face of computer tech was completely changed. On April 4, 1975, Bill Gates and Paul Allen established Microsoft to develop and sell BASIC interpreters for the Altair 8800 (Microcomputer designed in 1974 by MITS).

(Altair 8800 with 8-inch floppy disk)

Microsoft dominated the personal computer operating system market with MS-DOS in the mid-1980's. They first introduced their operating system (OS) Called Windows 1.0 on November 20, 1985, it was based on the graphical user interface (GUI), and they again dominated the personal computer market with over 90% market share (At that time). On the other side, Apple was founded in April 1976, by Steve Jobs, Steve Wozniak, and Renal Wayne to develop "Wozniak's Apple 1" a personal computer. They launched Apple 2 in January 1977, by the arrival of Alto around this time they also started to make their operating system based on the graphical user interface (GUI). They started a project to make a new personal computer with their operating system they started working on the project, Lisa, in 1978, and they finally came up with a personal computer "Lisa (Locally Integrated Software Architecture)" on January 19, 1983, It was quite costly at that around US$9,995 with a five-megabyte hard drive. It was a huge success for both the company to make their operating system, however, because of its' price both of them went flop.

The Basic OS

Lisa was introduced to a proprietary operating system and a suite of office applications. Several aspects of Lisa's software would become part of Apple's system to come, and in fact, many such concepts exist in MAC OS X in some form. Lisa was not tied to a particular operating system. Unlike its successor, the Macintosh, Lisa did not have portions of the operating system in ROM. This allowed it to support multiple operating systems. It presented the user with an interactive screen called the Environments Window if multiple bootable environments were found on the attached storage devices. Most of Lisa's system and application software was written in an extended version of Pascal (Lisa Pascal). During Lisa's development, Apple even considered using a p-Chip that would run p-Code natively. Lisa OS was designed to support Lisa's graphical user interface.



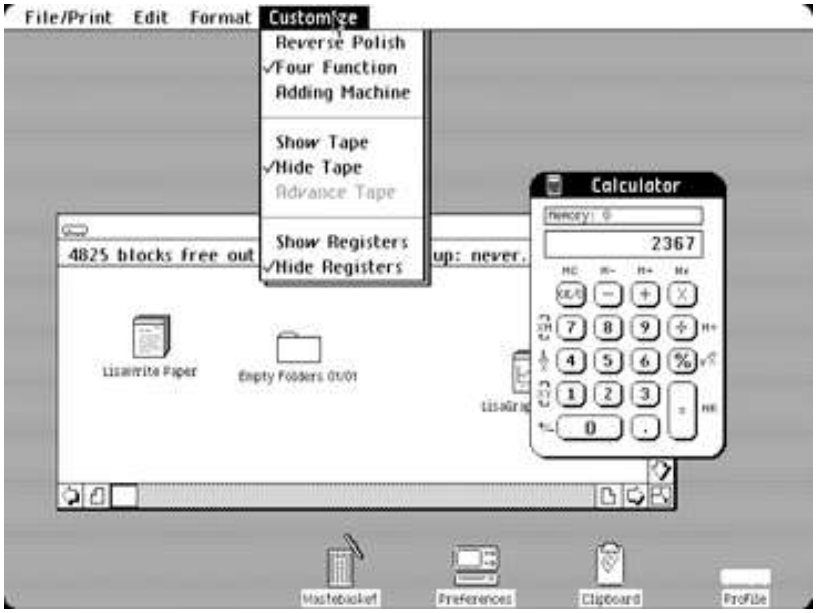
(Lisa's Software Architecture)

Process Management

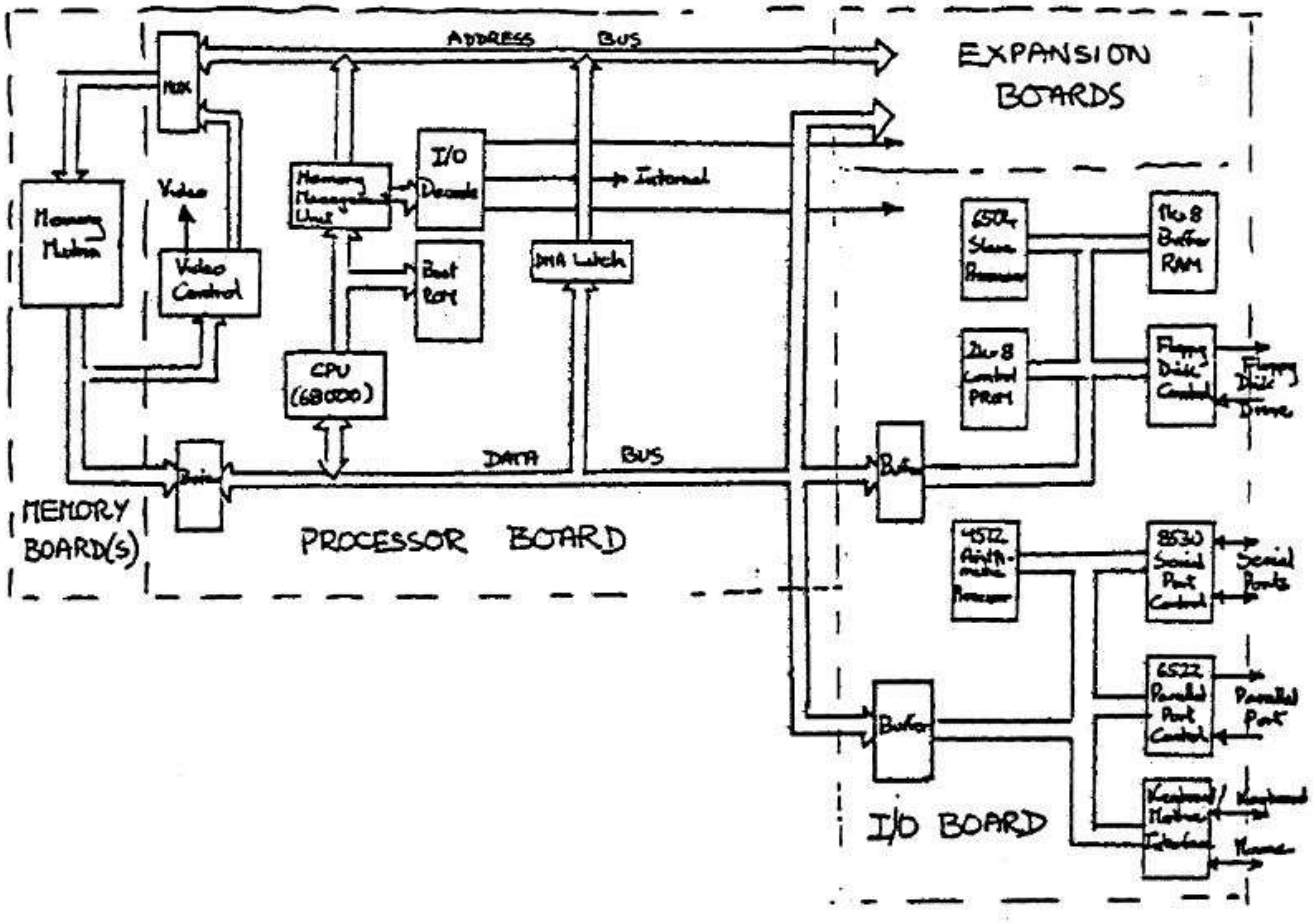
A process is an executing program and the data associated with it. Several processes can exist at one time, and they appear to run simultaneously because the CPU Is multiplexed among them. The scheduler decides what process should use the CPU at any one time. It uses a generally nonpreemptive scheduling algorithm. This means that a process will not lose the CPU unless It blocks. The blocked state is explained later In this section. A process can lose the CPU when one of the following happens :

- The process calls an Operating System procedure of function.
- The process reference one of its code segments that are not currently in memory.

If neither of these occurs, the process will not lose the CPU. Every process is started by another process. The newly started process is called the son process. The process that started it is called its father process. The resulting structure is a tree of processes. When any process terminates, all its son processes and their descendants are also terminated. When the OS is booted, it starts a shell process. The shell process starts any other process desired by the user. Every newly created process has the same system-standers attributes and capabilities. These can be changed by using system calls. Any processes can suspend, activate, or kill any other process for which the global ID is known, as long as the other process does not protect itself. The memory accesses of an executing process are restricted to its own memory address space. Processes can communicate with other processes by using shared files, pipes, event channels, or shared data segments. A process can be in one three states: ready, running, or blocked. A ready process is waiting for the scheduler to select it to run. A running process is currently using the CPU to execute its code. A blocked process is waiting for some event, such as the completion of an input/output (I/O) operation. It will not be scheduled until the event occurs, at which point becomes ready. A terminated process has finished executing. Each process has a priority from 1 to 255. The higher the number, the higher the priority of the process. Priorities 226 to 255 are reserved for system processes. The schedular always runs the ready process with the highest priority. A process can change its Priority, or the priority of any other process, while it is executing.



(Apple Lisa Office System)



(Process Management Brad of Lisa)

Conclusion

My study represents all the facts I gathered about the evolution of these three operating systems over time and how they improved their hardware according to the need, such as hard drive, disks, especially the architecture and the change in conditions of operating system and the formulas over the time.

References

1. Case Study: IBM's System/360-370 Architecture, Number 4, Volume 30, April 1987.
2. IBM Mainframe Operating Systems: Timeline and Brief Explanation For the IBM System/360 and Beyond, by Dave Morton, Version 33, September 2013.
3. IBM Operating System/360 Concepts and Facilities, File Number: S360-36, Form C28-6535-0.
4. IBM Operating System/360 Summary, File Number: S360-00, Form GA22-6810-12.
5. Development of 360/370 Architecture – A Plain Man's View, By P.J. Gribbin, February 10, 1989.
6. ALTO OPERATING SYSTEM REFERENCE MANUAL, version 2, March 17, 1977 (Copyright by Xerox).

7. Alto: A personal computer, by C.P. Thacker, E.M. McMreight, B.W. Lampson, R.F. Sproull, and D.R. Boggs, CSL-79-11, August 7, 1979 (Copyright by Xerox).
8. The Xerox Alto Computer, by Thomas A Wadlow, September 1981.
9. Dealers Of Lightning Xerox Parc And The Dawn Of The Computer Age, by Michael A. Hiltzik, ISBN 978-0-88730-989-2, the Year 2000.
10. A History of the GUI, by Jeremy Reimer.
11. An Open Operating System for a Single-User Machine, Butler W. Lampson, Robert F. Sproull.
12. ALTO: A Personal Computer System Hardware Manual, August 1976 (Copyright by Xerox).
13. A Technical History of Apple's Operating Systems, by Amit Singh, Version 2.0.1, July 28, 2006 (Copyright by Amit Singh).
14. Lisa Pascal 3.0 System Software, 620-6149-B, the Year 1983 (Copyright by Apple Inc.).

Citations

1. https://www.britannica.com/topic/Xerox-PARC
2. http://www.columbia.edu/cu/computinghistory/650.html
3. https://www.ibm.com/ibm/history/ibm100/us/en/icons/system360/
4. http://www.columbia.edu/cu/computinghistory/36091.html
5. http://www.columbia.edu/cu/computinghistory/36075.html
6. https://en.wikipedia.org/wiki/IBM_System/360
7. https://en.wikipedia.org/wiki/IBM_1401
8. http://www.digibarn.com/friends/curbow/star/index.html
9. https://www.britannica.com/technology/graphical-user-interface#ref665410
10. https://en.wikipedia.org/wiki/Xerox_Alto
11. https://www.britannica.com/biography/Charles-P-Thacker#ref1092161
12. https://books.google.ca/books?id=ti4EAAAAMBAJ&pg=PA52&redir_esc=y#v=onepage&q&f=false
13. All the Pictures used in this Research Paper is Copyrighted by IBM, Xerox, and Apple Inc.