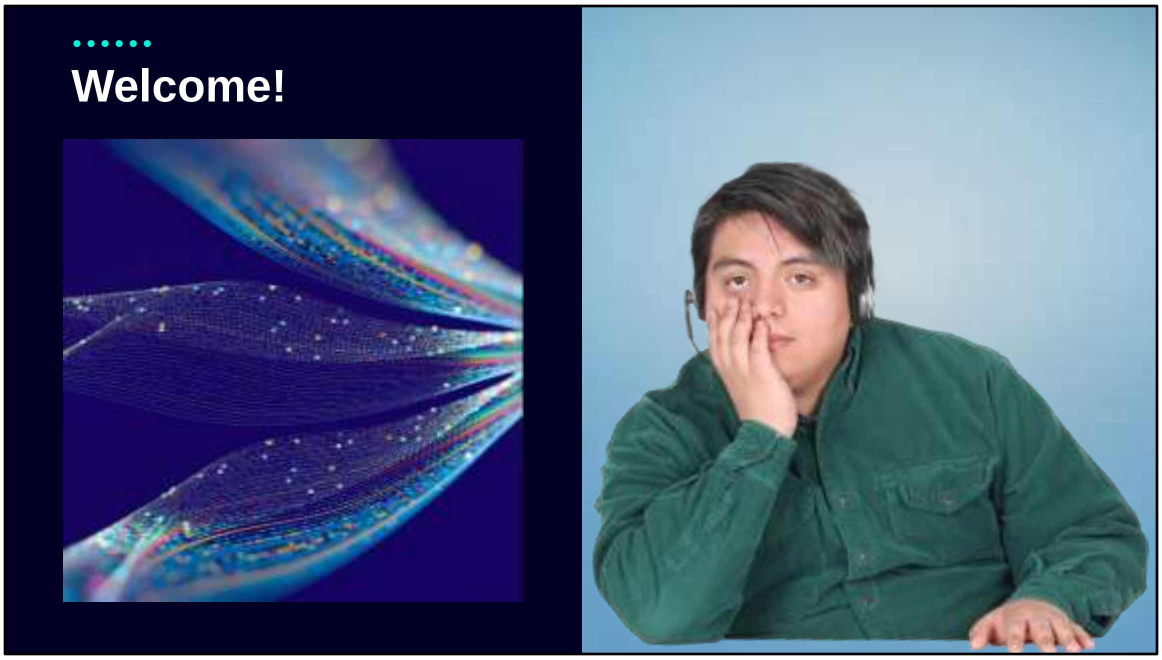
Introduction to Operating Systems

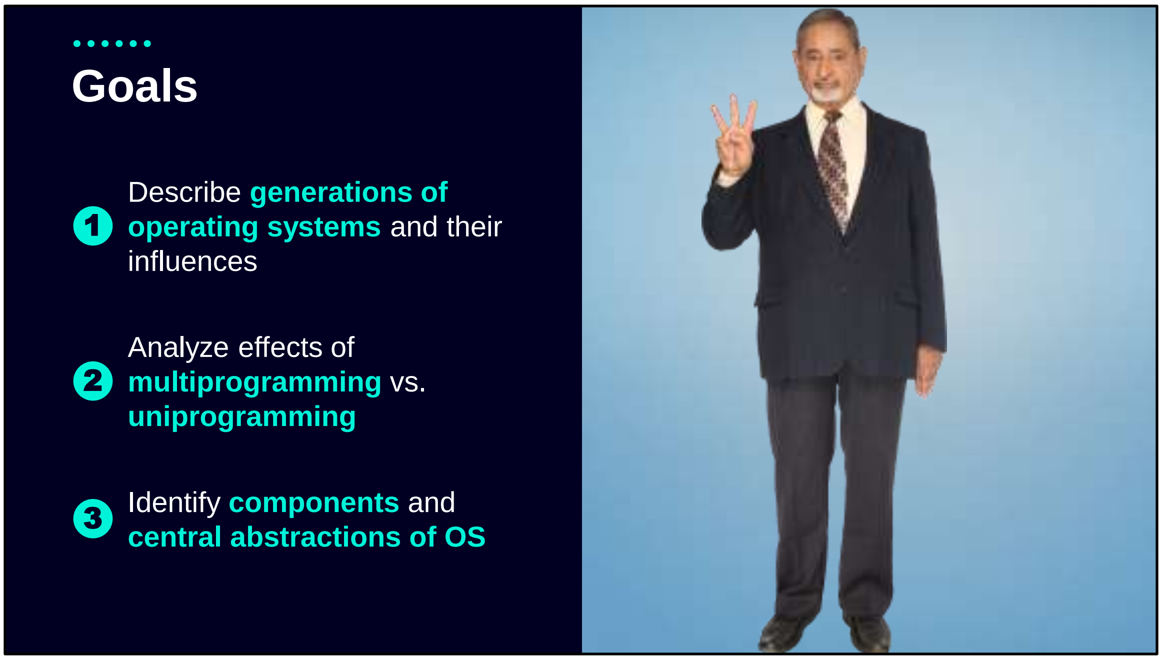


Welcome!

ON SCREEN

Welcome! In this lecture, we will look at a brief history of operating systems. We will look at how they evolved over the years in parallel to the evolution of the hardware that they run on.

[CLICK – Next Slide]



Goals

1. Describe **generations of operating systems** and their influences
2. Analyze effects of **multiprogramming** vs. **uniprogramming**
3. Identify **components** and **central abstractions of OS**

ON SCREEN

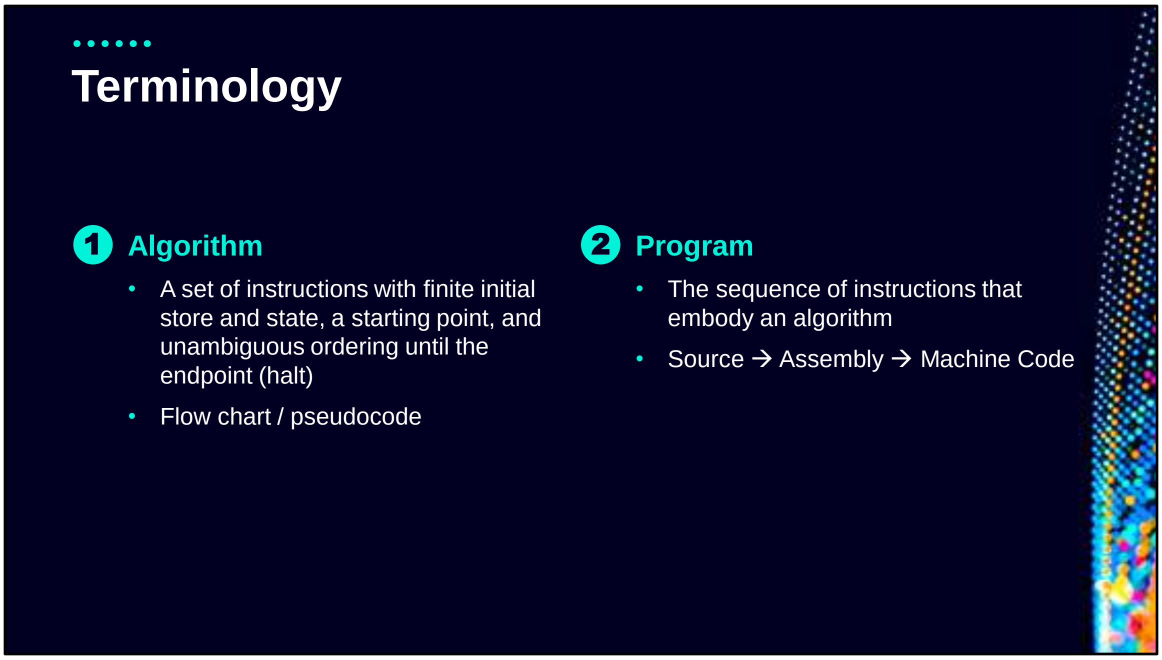
By the end of this lecture, you should be able to: **[CLICK]**

Describe the generations of operating systems and their influences; **[CLICK]**

Analyze effects of multiprogramming vs. uniprogramming; and **[CLICK]**

Identify components and central abstractions of OS

[CLICK – Next Slide]



Terminology

1 **Algorithm**

- A set of instructions with finite initial store and state, a starting point, and unambiguous ordering until the endpoint (halt)
- Flow chart / pseudocode

2 **Program**

- The sequence of instructions that embody an algorithm
- Source → Assembly → Machine Code

OFF SCREEN

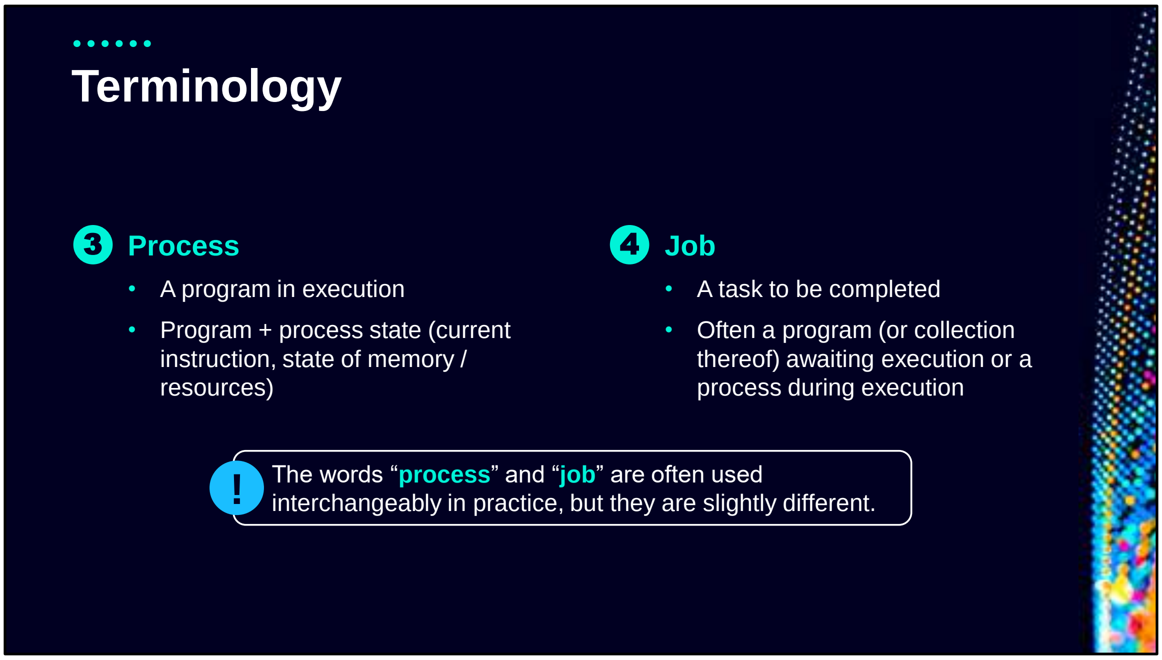
But first, let’s look at some terminology. **[CLICK]**

Algorithm: A set of instructions with finite initial store and state, a starting point, and unambiguous ordering until the endpoint (halt point).

Can be represented by a flow chart or a pseudocode. **[CLICK]**

Program: The sequence of instructions that embody an algorithm. It can be the Source code or Machine Code.

[CLICK – Next Slide]



Terminology

3 Process

- A program in execution
- Program + process state (current instruction, state of memory / resources)

4 Job

- A task to be completed
- Often a program (or collection thereof) awaiting execution or a process during execution

! The words "**process**" and "**job**" are often used interchangeably in practice, but they are slightly different.

OFF SCREEN

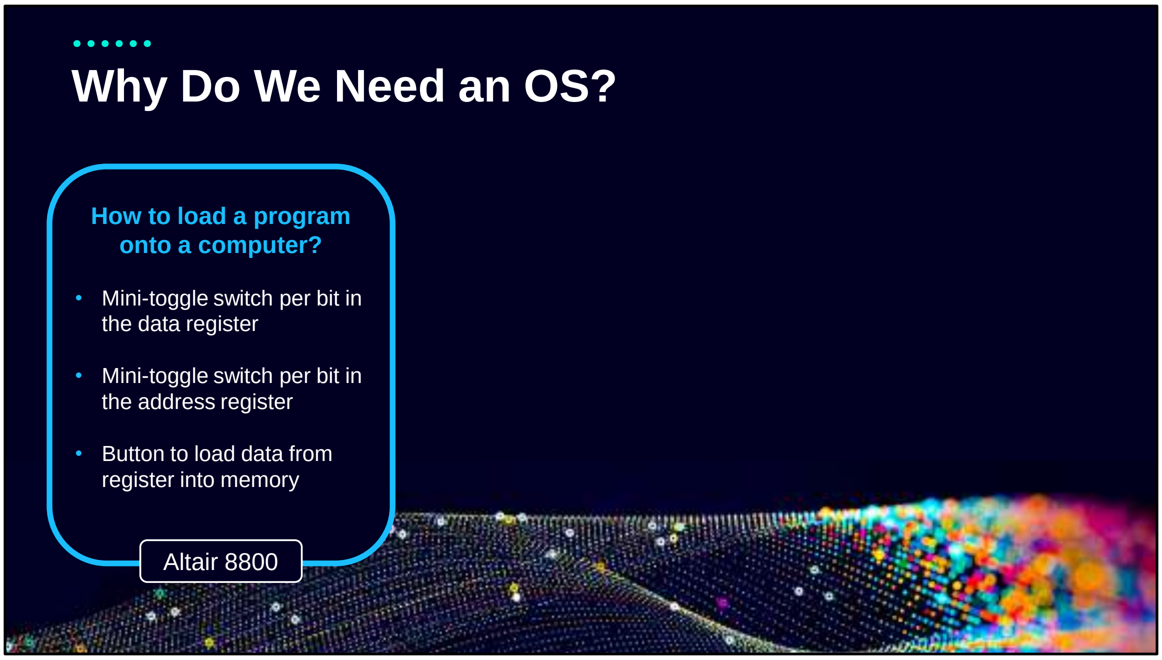
Process: a program in execution Program plus, its Process State. These are represented by the current instruction, state of memory and resources. **[CLICK]**

Job: a task to be completed.

Often a program or a collection of programs awaiting execution. A "job" can also be a process during execution. **[CLICK]**

The words 'process' and 'job' are often used interchangeably in practice, but they are slightly different.

[CLICK – Next Slide]



Why Do We Need an OS?

How to load a program onto a computer?

- Mini-toggle switch per bit in the data register
- Mini-toggle switch per bit in the address register
- Button to load data from register into memory

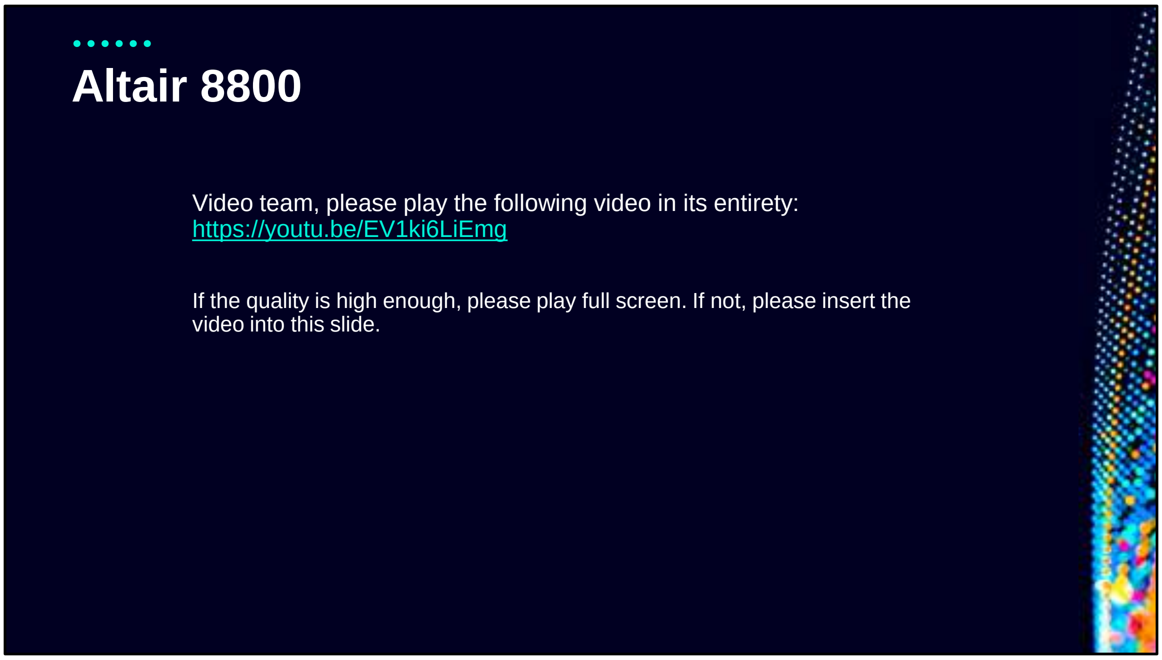
Altair 8800

OFF SCREEN
Why do we need an OS? For example, to load a program onto a computer.

So, how to load a program onto a computer?
The OS helps us with repetitive or complex tasks that we want to achieve with a computer. For example, loading a program can involve a series of precise but repetitive tasks such as loading data from main memory to a register. **[CLICK]**

The Alteir 8800 is an example of an old computer that did not have an Operating System. Programs had to be loaded manually before execution. Even to run a simple program involved a series of tasks to be accomplished manually by the operator/programmer. Let’s watch a video of how the Altair 8800 works.

[CLICK – Next Slide]



Altair 8800

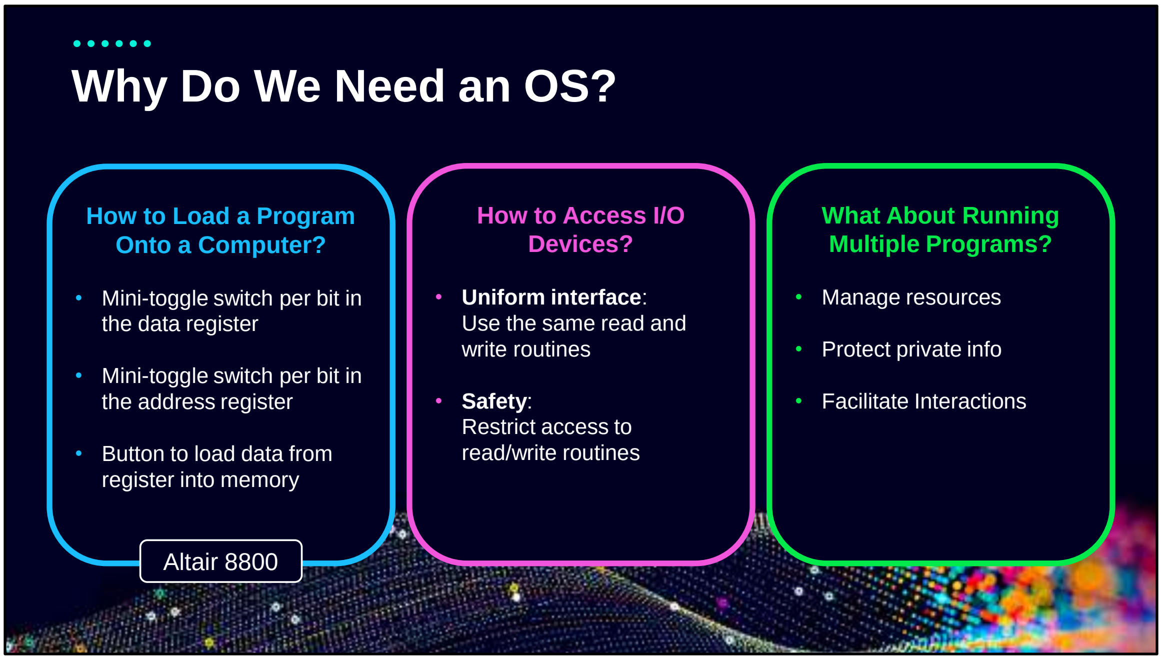
Video team, please play the following video in its entirety:
https://youtu.be/EV1ki6LiEmg

If the quality is high enough, please play full screen. If not, please insert the video into this slide.

OFF SCREEN

[PAUSE FOR A COUPLE OF SECONDS THEN CLICK TO NEXT SLIDE]

[CLICK – Next Slide]



Why Do We Need an OS?

How to Load a Program Onto a Computer?

- Mini-toggle switch per bit in the data register
- Mini-toggle switch per bit in the address register
- Button to load data from register into memory

Altair 8800

How to Access I/O Devices?

- **Uniform interface**: Use the same read and write routines
- **Safety**: Restrict access to read/write routines

What About Running Multiple Programs?

- Manage resources
- Protect private info
- Facilitate Interactions

OFF SCREEN

The OS also helps accessing I/O devices

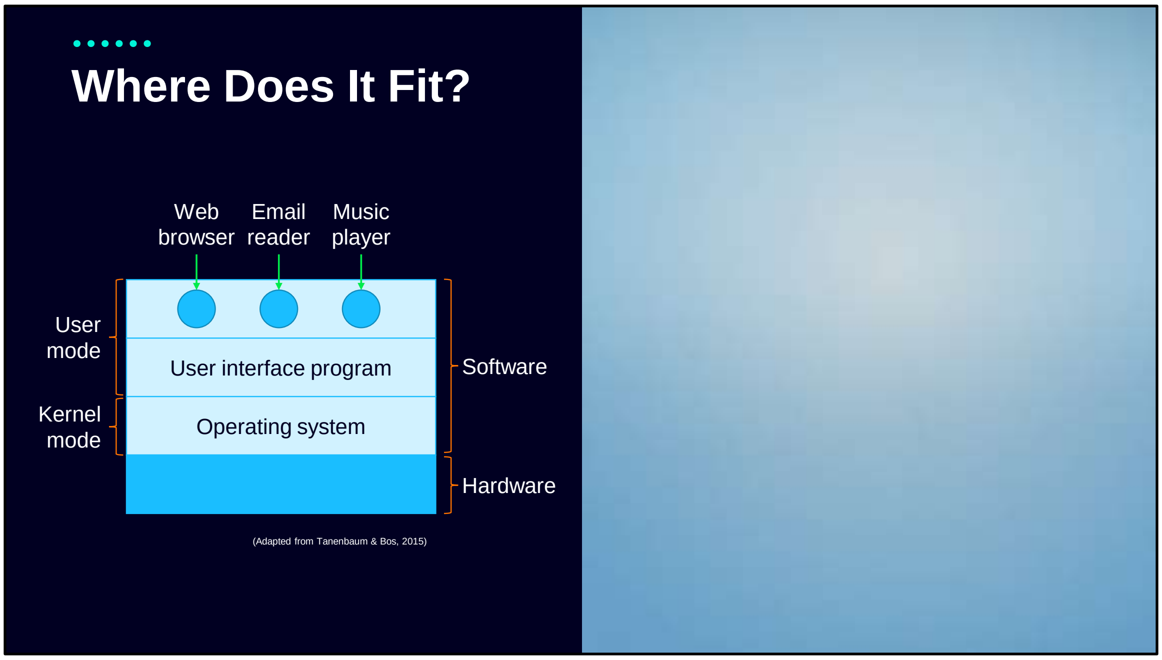
The OS provides the user with a uniform interface to access input and output devices. It often standardizes input and output routines. **[CLICK]**

The OS also provides a layer of safety. It restricts access to read/write routines to users and process. This keeps users or processes from accessing resources that they shouldn’t, which can disrupt the function of the entire operating system. **[CLICK]**

What about running multiple programs?

The OS also Manages resources, Protects private information, and Facilitates Interactions between multiple users, processes, and programs. In modern OSs, we are used to multi-tasking: write an email, while glancing at the web browser and listening to some music. The OS makes sure all these programs and tasks are accomplished for the user.

[CLICK – Next Slide]



Where Does It Fit?

(Adapted from Tanenbaum & Bos, 2015)

ON SCREEN

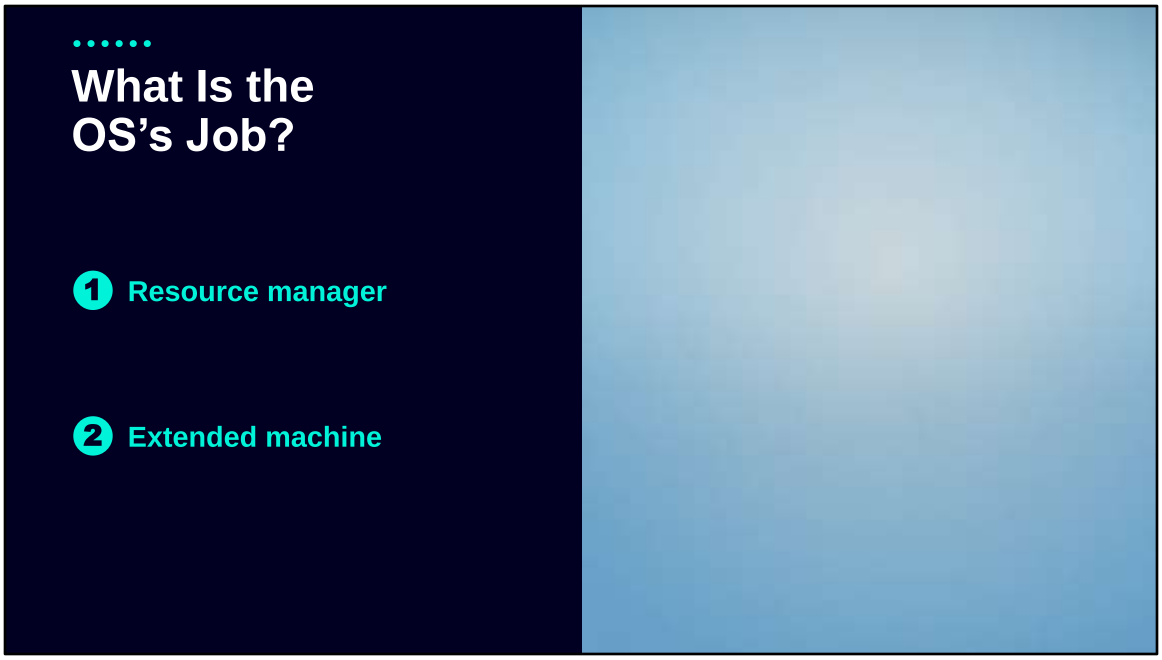
But where does the OS fit in the organization of a typical personal computer system?

The OS is the layer of software that exists between the hardware and the user-interface program.

While the user-interface often seems to be part of the OS, it is not. It is one of the programs that it helps to execute for the user.

This is more transparent, for example, to Unix/Linux users that are used to changing user interfaces, while it may seem stranger to Windows or Mac OS users, that cannot choose between different user interface programs.

[CLICK – Next Slide]



What Is the OS's Job?

1. Resource manager
2. Extended machine

ON SCREEN

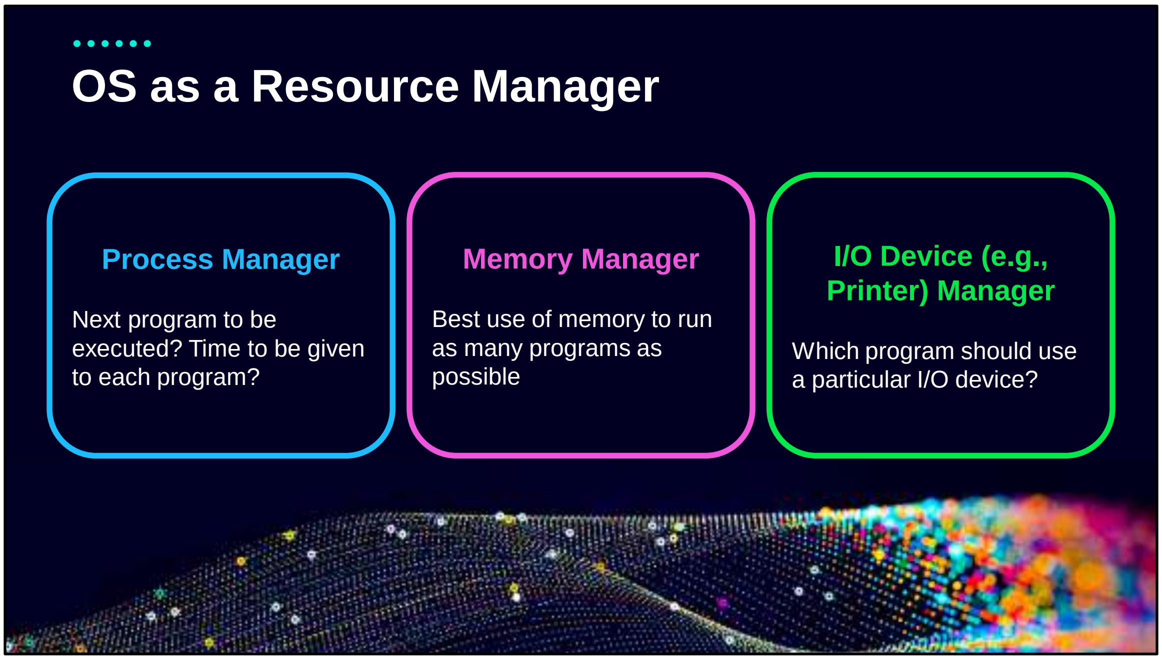
The OS has two primary jobs: **[CLICK]**

Resource Manager and **[CLICK]**

To provide an extended machine to the user. **[CLICK]**

Let's begin with OS as a resource manager.

[CLICK – Next Slide]



OS as a Resource Manager

Process Manager

Next program to be executed? Time to be given to each program?

Memory Manager

Best use of memory to run as many programs as possible

I/O Device (e.g., Printer) Manager

Which program should use a particular I/O device?

OFF SCREEN

As a Resource Manager, the OS works as: **[CLICK]**

A Process Manager: Typically, considering modern interactive OSs, such as the ones we use in our laptops and desktops, multiple programs and processes alternate execution every second. The OS switches between them continuously, giving the user the illusion that those programs are executing at the same time, in parallel. In reality, each program runs for a fraction of a second and lets the next program run. **[CLICK]**

One of the jobs of the OS is to define the next program to be executed, which programs have priority, and which programs can wait. **[CLICK]**

As a Memory Manager, the OS decides what is the best use of the memory space available. It decides which programs to load to main memory, which programs to keep on fixed storage (e.g., SSD or hard drive), and which processes or programs to load into cache memory. **[CLICK]**

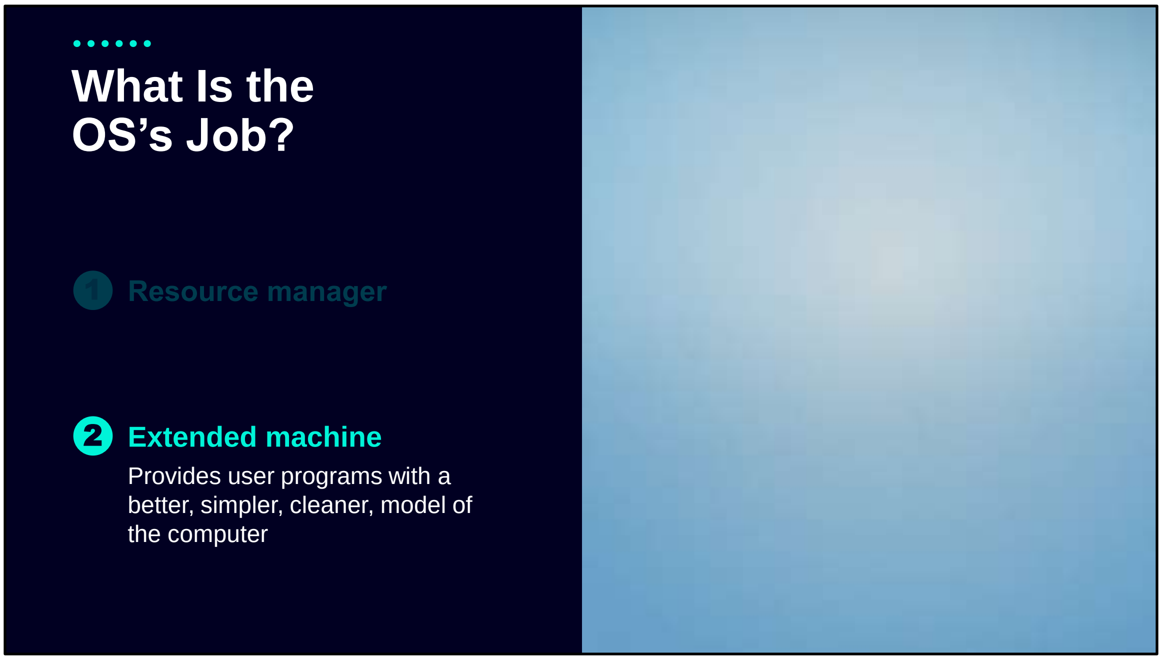
The OS also manages the I/O devices. It communicates to output devices such as disks, printers, and displays, and manages input such as from mouse, keyboard, camera, etc.

[CLICK – Next Slide]

ON SCREEN

In many ways, the OS can be compared to a one-man band that must perform each task correctly and the right time so that it can provide a good user-experience to the user.

[CLICK – Next Slide]



What Is the OS’s Job?

1 Resource manager

2 **Extended machine**

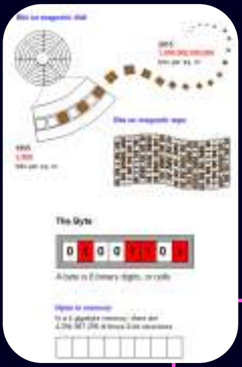
Provides user programs with a better, simpler, cleaner, model of the computer

ON SCREEN

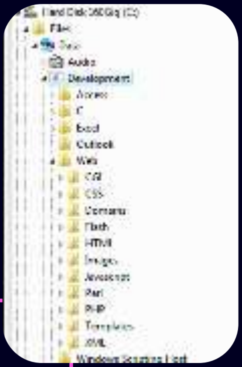
The OS also provides an extended machine to the user. This means that it hides the complexities of the underlying hardware and software, and provides users with a better, simpler, cleaner,
model of the computer.

[CLICK – Next Slide]

OS as an Extended Machine



(Freedman & Morrison, 2022)



(Virtue, 2017)

OS

OFF SCREEN

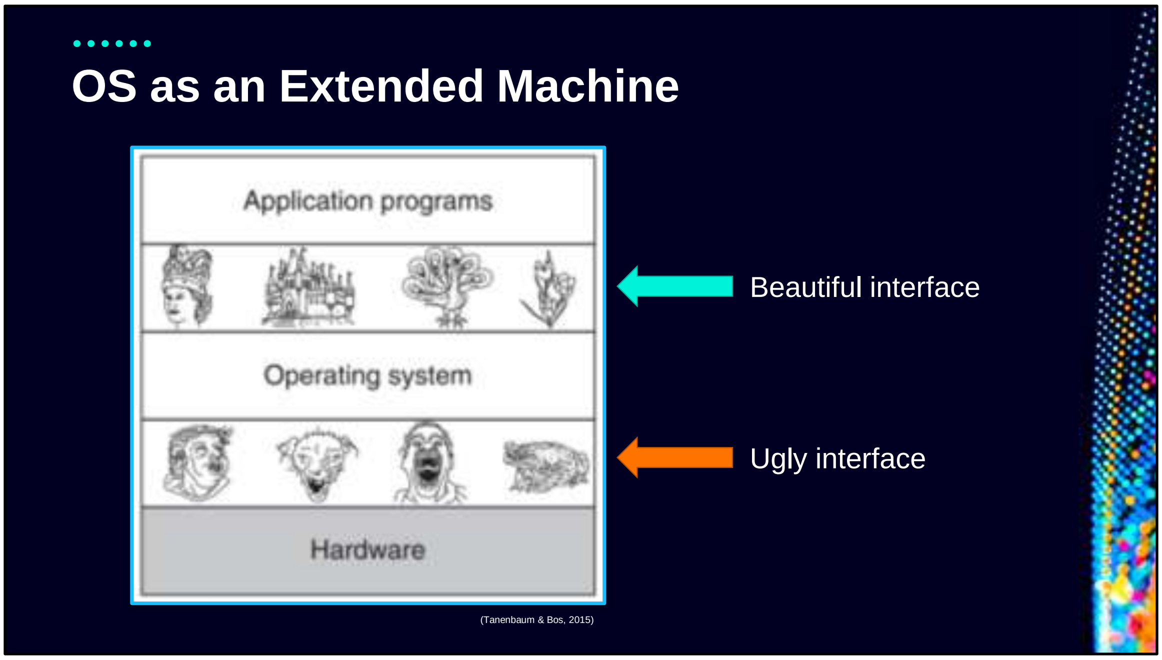
For example, when we create a text file, an image, or a video, **[CLICK]**

we typically don't think about how those bits and bytes will be stored. **[CLICK]**

All we think about is perhaps in which folder we will save the file so that we can retrieve later. **[CLICK]**

All the complexity involved with creating, naming, storing, and retrieving the file is managed by the OS.

[CLICK – Next Slide]



OS as an Extended Machine

OFF SCREEN

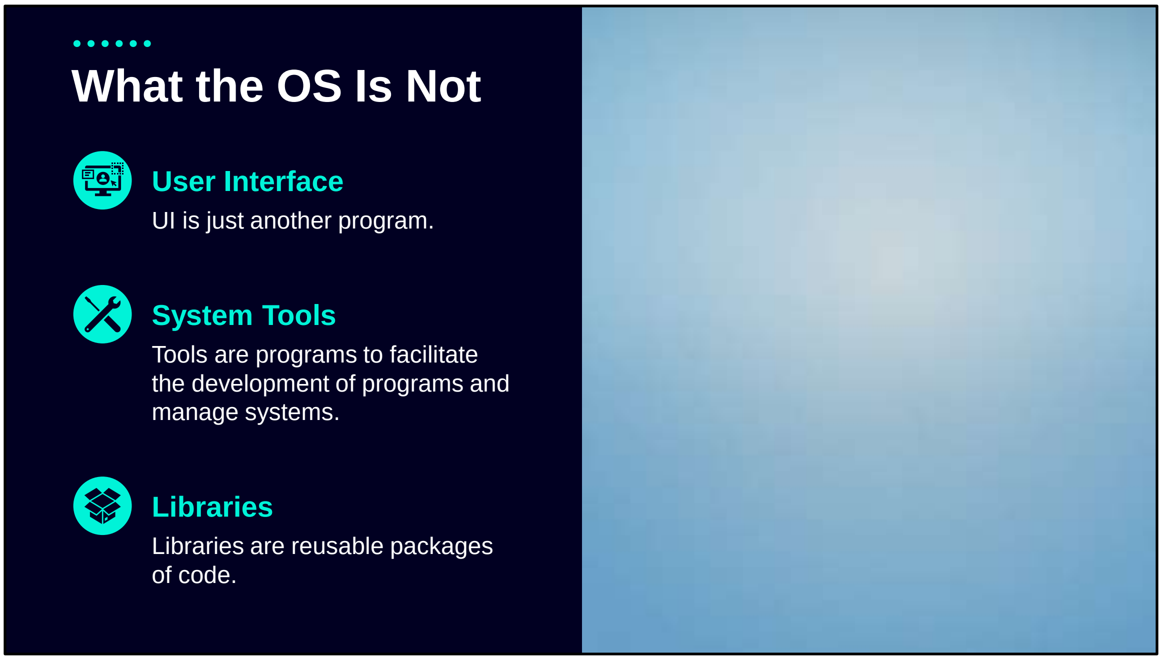
A good analogy is provided by Tanenbaum and Bos in the Modern Operating Systems book. **[CLICK]**

We can think about the OS as providing a “beautiful” interface to the users and user programs, **[CLICK]**

while dealing with the “ugly” interfaces provided by hardware components.

Beautiful here is related to interfaces that are easy to use and effective to humans.

[CLICK – Next Slide]



What the OS Is Not

User Interface
UI is just another program.

System Tools
Tools are programs to facilitate the development of programs and manage systems.

Libraries
Libraries are reusable packages of code.

ON SCREEN

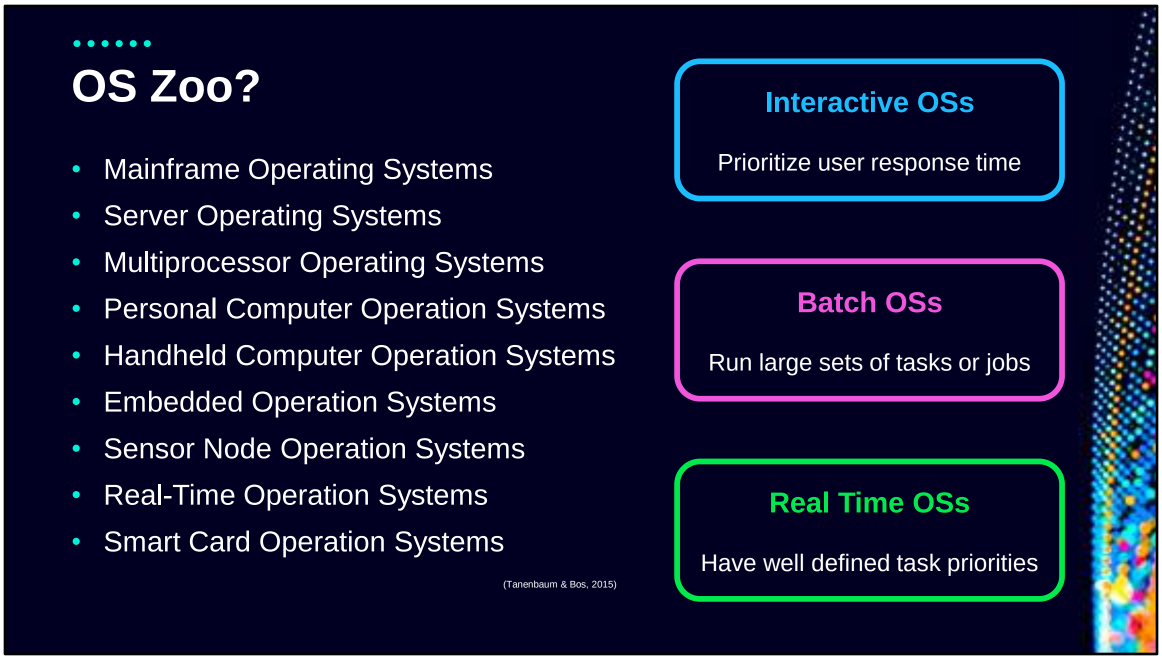
We have talked a lot about what the OS is. Here are somethings that the OS is not: **[CLICK]**

The OS is not the user interface. We mentioned OSs such as Linux, that allow the user to switch between different operating systems. **[CLICK]**

The OS is not the System Tools. Tools that facilitate the development of programs are not part of the OS. **[CLICK]**

The OS is also not the Libraries it utilizes. Such libraries may allow the OS to function in different ways but are not part of the OS itself.

[CLICK – Next Slide]



OS Zoo?

- Mainframe Operating Systems
- Server Operating Systems
- Multiprocessor Operating Systems
- Personal Computer Operation Systems
- Handheld Computer Operation Systems
- Embedded Operation Systems
- Sensor Node Operation Systems
- Real-Time Operation Systems
- Smart Card Operation Systems

(Tanenbaum & Bos, 2015)

Interactive OSs

Prioritize user response time

Batch OSs

Run large sets of tasks or jobs

Real Time OSs

Have well defined task priorities

OFF SCREEN

While we, most of the time, refer to OSs as the software that runs on our laptops and desktops, the term refers to a family of OSs that perform similar jobs.

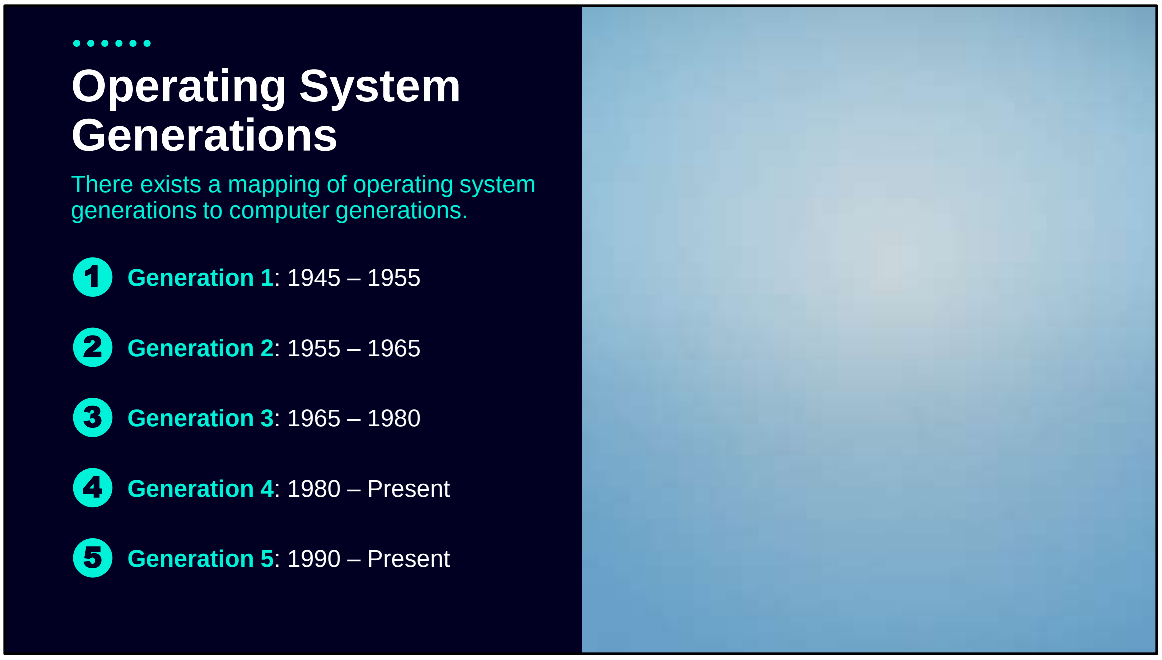
While the list is vast, consisting of OSs for mainframes, servers, handheld, real-time systems, etc., we can classify OSs into three large groups: **[CLICK]**

Interactive OSs: such as the ones we use in our desktops and laptops. These systems prioritize user response time. If the user interacts, the OS must give immediate, friendly feedback. **[CLICK]**

Batch OS: These systems are optimized to run typically large sets of tasks or jobs. They prioritize the completion of pre-defined queued tasks instead of user interaction. The user may have to wait for a job to finish to be able to interact or receive feedback from the system. **[CLICK]**

Real-time OSs: These specialized operating systems typically have well defined task priorities. The avionics system of an airplane, or safety systems of a car may run real-time OSs, that primarily must react/provide feedback within well defined time constraints.

[CLICK – Next Slide]



Operating System Generations

There exists a mapping of operating system generations to computer generations.

1. **Generation 1**: 1945 – 1955
2. **Generation 2**: 1955 – 1965
3. **Generation 3**: 1965 – 1980
4. **Generation 4**: 1980 – Present
5. **Generation 5**: 1990 – Present

ON SCREEN

Toward understanding how OSs evolved overtime, classifications have been created.

OSs have been classified into 5 Generations. These generations are often closely related to the evolution of the software of OSs and the hardware they ran on. **[CLICK]**

Let’s look at the first three generations in detail.

[CLICK – Next Slide]

Generation 1: Vacuum Tubes and Plugboards

1945 – 1955

Programmer / user = operator

Single application program at a time – no OS

(Weik, 1961)

Remmington Rand 409 (Univac)

(Columbia University, 2001)

Plugboard

OFF SCREEN

The first generation goes from 1945 until 1955.

While there were computers prior to 1945, this date was chosen since computers from this era share similarities with hardware and software we still use to this date. **[CLICK]**

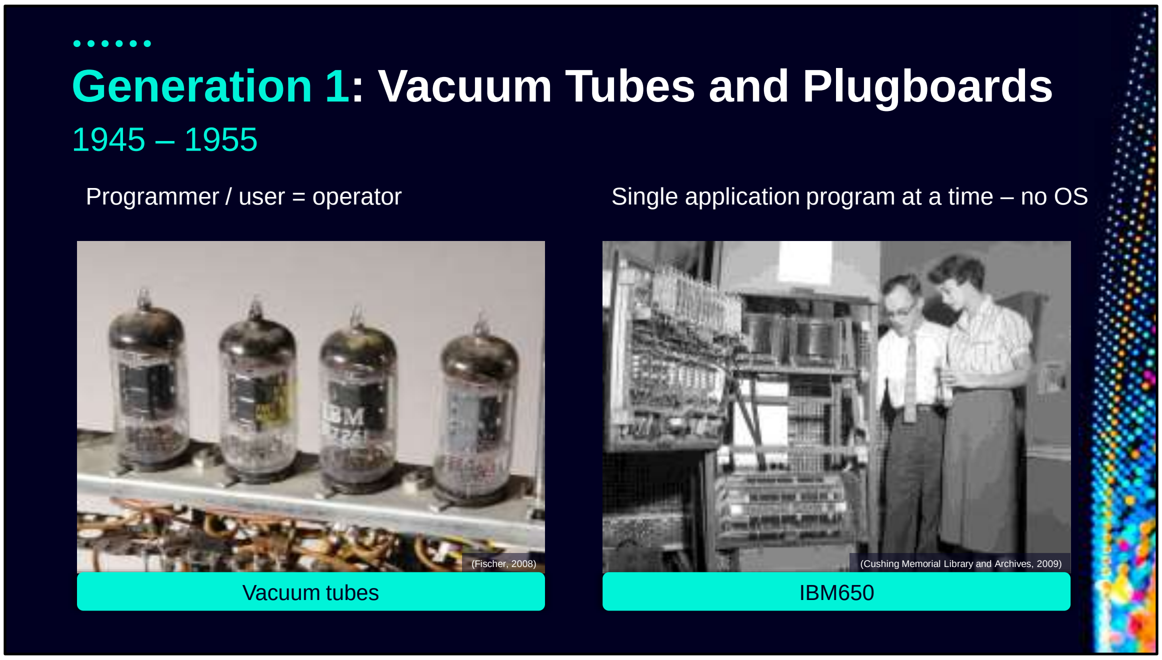
Computers of this era were not reliable. Besides knowledge about programming, users had also to learn how to keep the machine running. In later generations, these two tasks were divided between the “programmer”, who knew how to create software for the computer, and the “operator”, who knew how to load programs and maintain the machine working. **[CLICK]**

So, Generation 1 is said to be characterized by having the users of the computers also be their operators (users wrote the code, loaded into the machines, and maintained the machines). **[CLICK]**

To program computers of this era, users/programmers used plugboards. Programs were represented by wiring instructions into physical boards that were than connected to the computer. **[CLICK]**

Another characteristic of Gen 1 computers is that they ran a single program at a time, that were small, and simple compared to today's standards.

[CLICK – Next Slide]



Generation 1: Vacuum Tubes and Plugboards

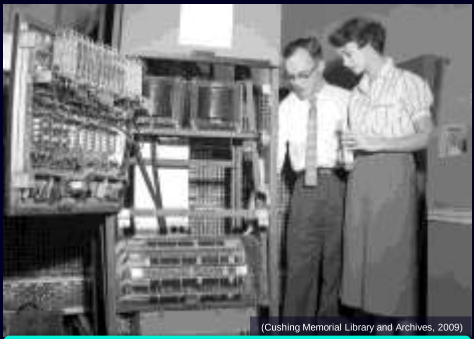
1945 – 1955

Programmer / user = operator

Single application program at a time – no OS

(Fischer, 2008)

Vacuum tubes


(Cushing Memorial Library and Archives, 2009)

IBM650

OFF SCREEN

One reason why computers were unreliable is that the hardware was primitive. **[CLICK]**

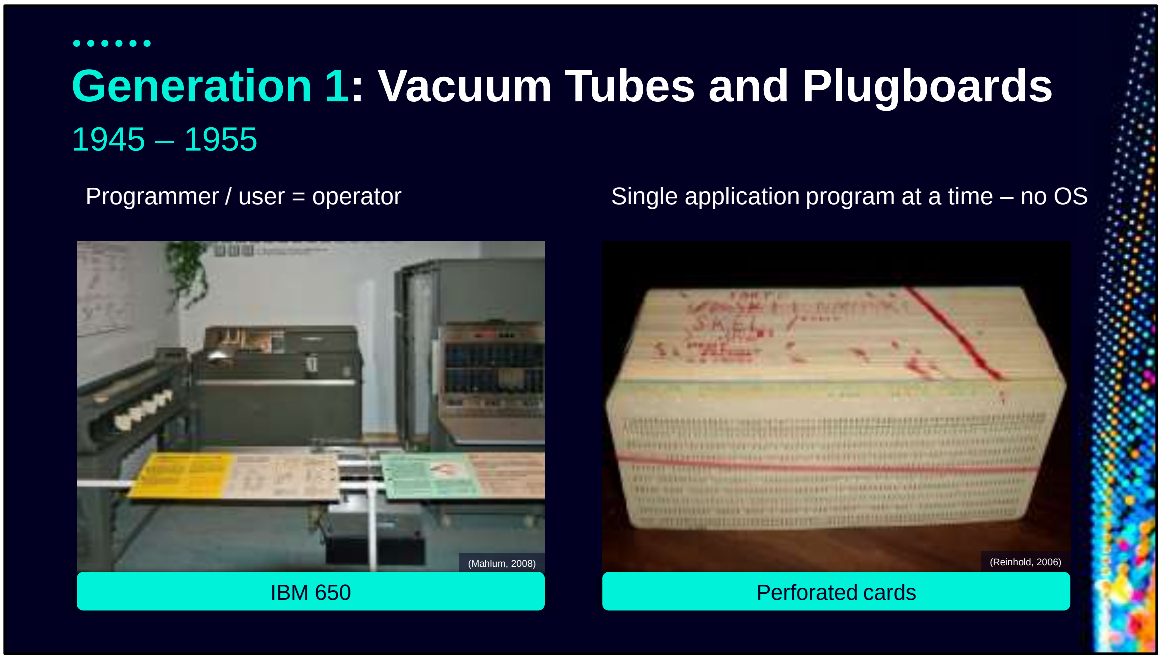
For example, Logic ports were implemented using Vacuum tubes.

Vacuum tubes required constant maintenance, work within narrow temperature ranges and are prone to failure.

A big part of an operator's job was to replace or fix vacuum tubes when they failed. **[CLICK]**

Because of the low reliability, commercial computers were rare. They were mostly deployed in military facilities and research institutes. A famous exemplar of that era is the IBM 650 computer.

[CLICK – Next Slide]



Generation 1: Vacuum Tubes and Plugboards

1945 – 1955

Programmer / user = operator

Single application program at a time – no OS

(Mahlum, 2008)

IBM 650

(Reinhold, 2006)

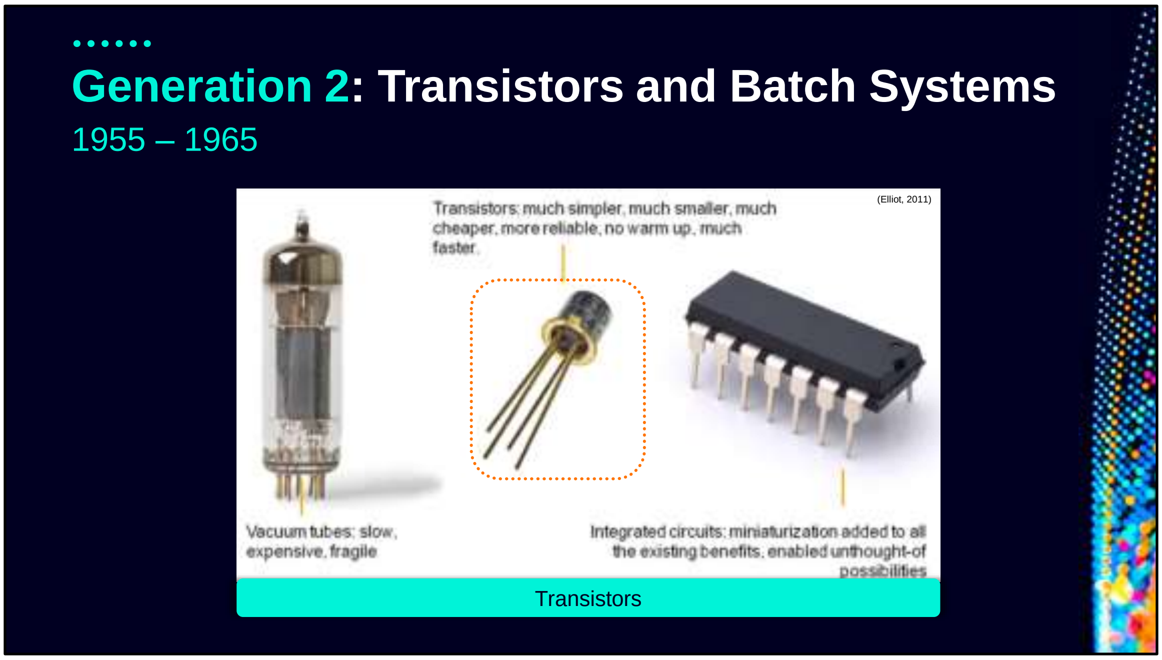
Perforated cards

OFF SCREEN

By the end of this generation, Perforated cards began to be used. While they do not characterize this generation, perforated cards allowed in increase in program complexity since they replaced the plugboards. **[CLICK]**

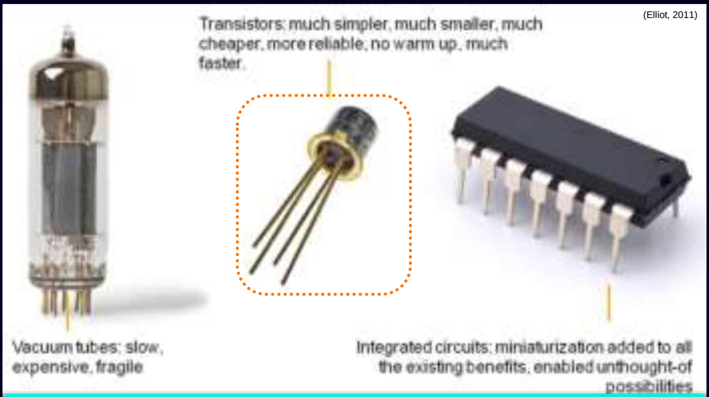
It was now possible to write programs on cards and have machines read them in instead of using plugboards.

[CLICK – Next Slide]



Generation 2: Transistors and Batch Systems

1955 – 1965


(Elliot, 2011)
Transistors: much simpler, much smaller, much cheaper, more reliable, no warm up, much faster
Vacuum tubes: slow, expensive, fragile
Integrated circuits: miniaturization added to all the existing benefits, enabled unthought-of possibilities

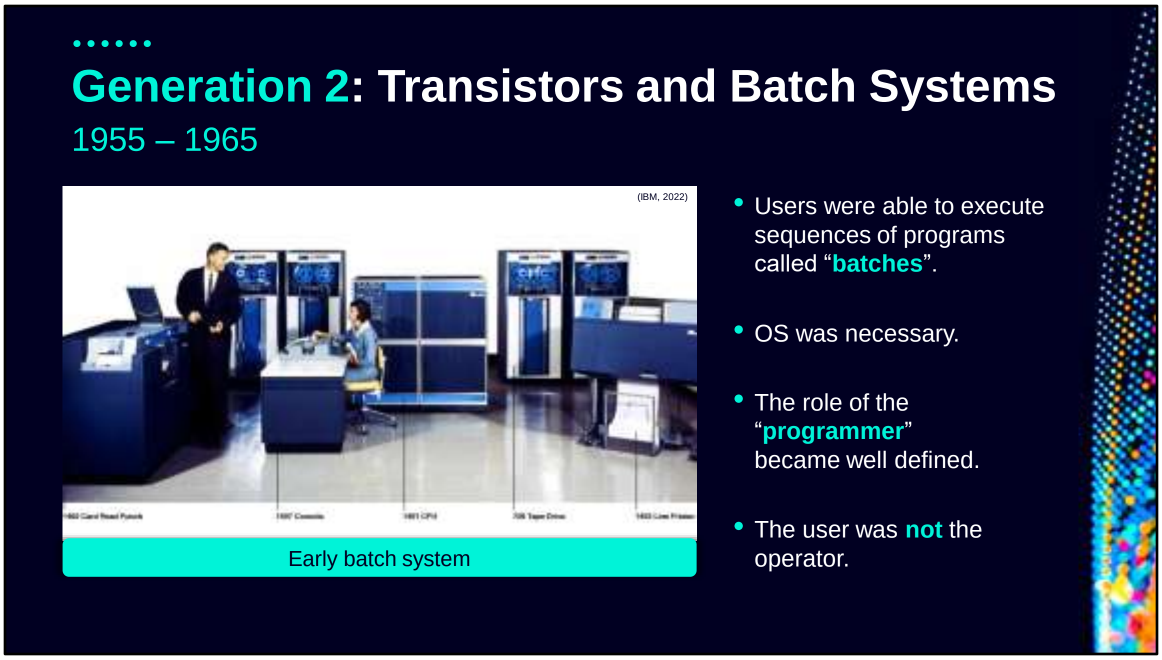
Transistors

OFF SCREEN

Generation 2 (1955 – 65): Transistors and batch systems **[CLICK]**

Hardware-wise, computers began be constructed using Transistors. Compared to vacuum tubes, transistors are reliable and much faster. Commercial mainframes became more popular and were purchased by large companies.

[CLICK – Next Slide]



Generation 2: Transistors and Batch Systems

1955 – 1965

(IBM, 2022)

Early batch system

- Users were able to execute sequences of programs called “**batches**”.
- OS was necessary.
- The role of the “**programmer**” became well defined.
- The user was **not** the operator.

OFF SCREEN

The image shows a typical system of the era. **[CLICK]**

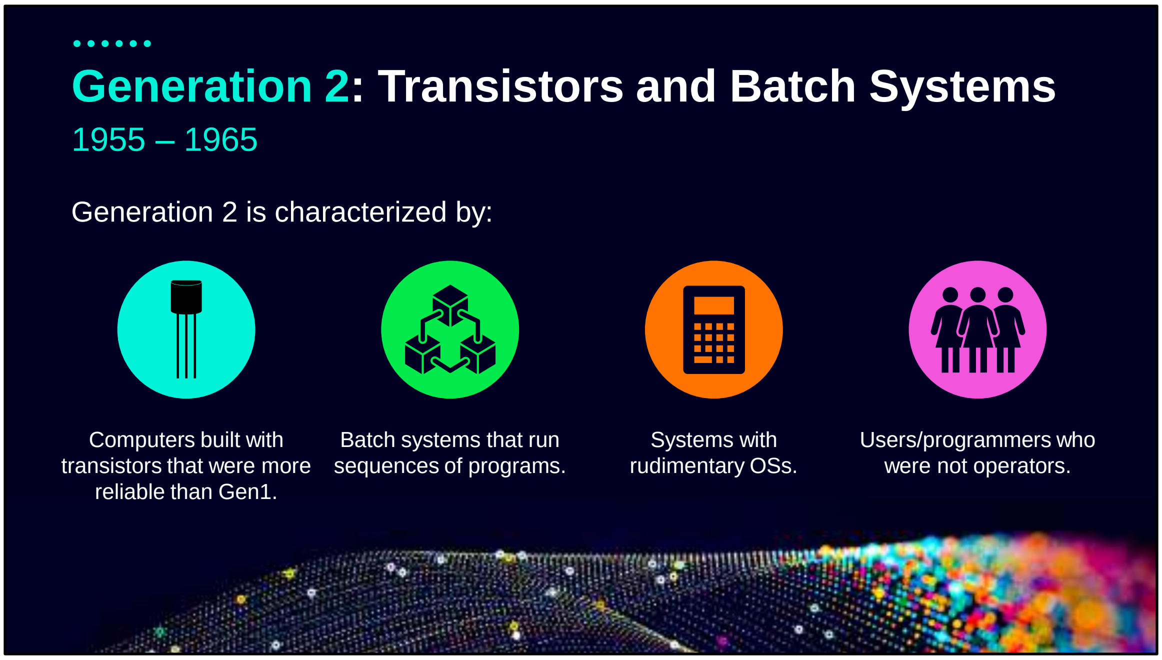
With the evolution in hardware, users were able to execute sequences of programs in the computers. These sequences of programs were called “batches.” **[CLICK]**

Operating Systems were necessary. Their job was to load, execute and save the results of one program and do the same for the next until the batch job was finished. **[CLICK]**

The role of the “programmer” became well defined. Programmers did not know how to “fix” the machine or keep it running and were not typically responsible for loading programs into it or operating the computer. **[CLICK]**

Operators would receive perforated cards from programmers and load them into the machine for execution. It may sound strange to us, but a programmer of that era might not interact with the computer at all.

[CLICK – Next Slide]



Generation 2: Transistors and Batch Systems

1955 – 1965

Generation 2 is characterized by:

- Computers built with transistors that were more reliable than Gen1.
- Batch systems that run sequences of programs.
- Systems with rudimentary OSs.
- Users/programmers who were not operators.

OFF SCREEN

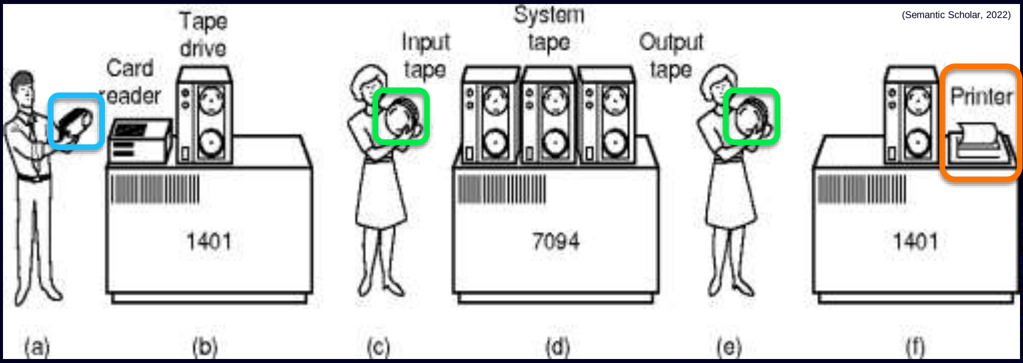
So, Generation 2 is characterized by: **[CLICK]**

- Computers built with transistors were more reliable than Gen1. **[CLICK]**
- Batch systems that run sequences of programs. **[CLICK]**
- Systems had rudimentary OSs. **[CLICK]**
- Users/programmers were not Operators

[CLICK – Next Slide]

Generation 2: Transistors and Batch Systems

1955 – 1965



(Semantic Scholar, 2022)

(a) Programmers bring cards to 1401

(b) 1401 reads batch of jobs onto tape

(c) Operator carries input tape to 7094

(d) 7094 does computing

(e) Operator carries output tape to 1401

(f) 1401 prints output

OFF SCREEN

A typical system of this generation would be composed of input unity, processing unity, and output unity. **[CLICK]**

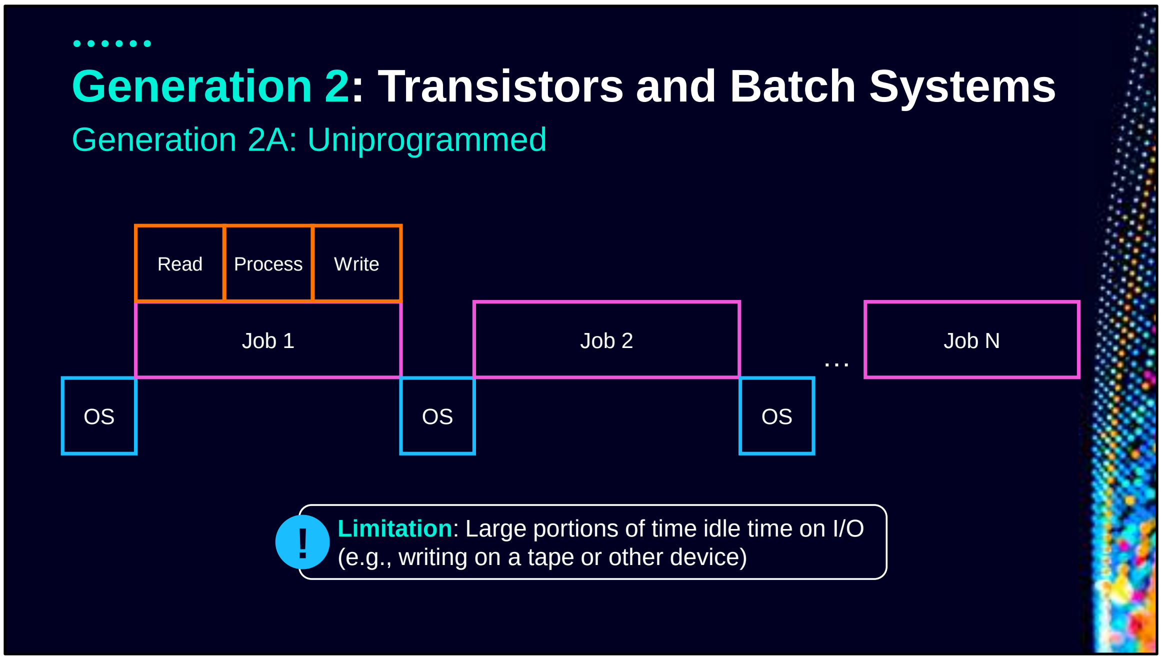
Magnetic tapes were typically used for storage. **[CLICK]**

Perforated cards were typically used for input, **[CLICK]**

while a printing unit would generate the output. **[CLICK]**

Once the batches of programs were loaded, there was no interaction with users. Users had to wait until the execution finished to read the output.

[CLICK – Next Slide]



Generation 2: Transistors and Batch Systems

Generation 2A: Uniprogrammed

Limitation: Large portions of time idle time on I/O (e.g., writing on a tape or other device)

OFF SCREEN

Generation 2 is divided into 2a and 2b. **[CLICK]**

Generation 2a: Each program on a batch (set of programs to execute), would complete before another one was loaded into the system for execution.

Programs had typically 3 steps: **[CLICK]**

Read data from tapes, **[CLICK]**

Process the data using the CPU, **[CLICK]**

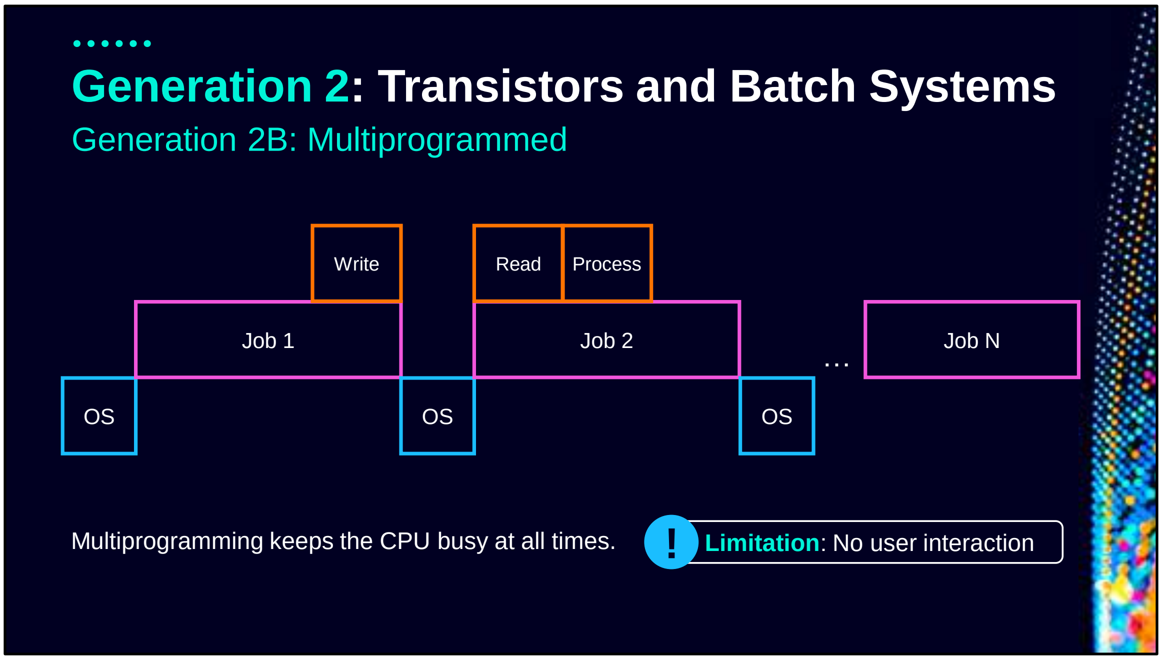
and Write the results. **[CLICK]**

The majority of the time was spent either reading data or writing data, with large portions of time in which the CPU was idle, not performing any useful task.

This setup is defined as “Uniprogrammed Batch Systems” since there is a single program running at any given time.

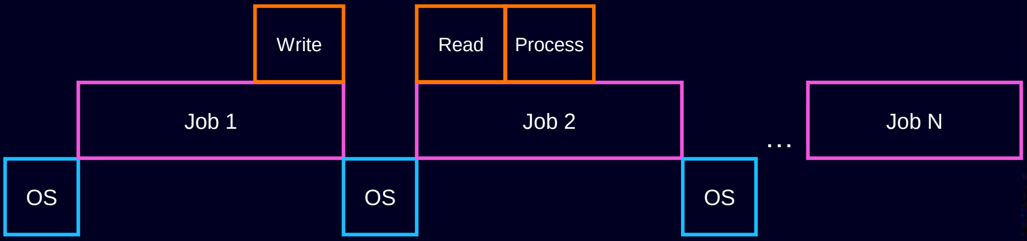
A program in execution is either reading, processing or writing. The other programs in the batch must wait the completion of the previous program to begin execution.

[CLICK – Next Slide]



Generation 2: Transistors and Batch Systems

Generation 2B: Multiprogrammed



Multiprogramming keeps the CPU busy at all times.

! **Limitation**: No user interaction

OFF SCREEN

Generation 2b is characterized by Multiprogrammed systems. **[CLICK]**

The aim is to keep the CPU busy at all times.

This is accomplished by utilizing a pipeline approach. **[CLICK]**

While one program is writing, **[CLICK]**

another may be executing and yet another may be read. So, a pipeline is formed.

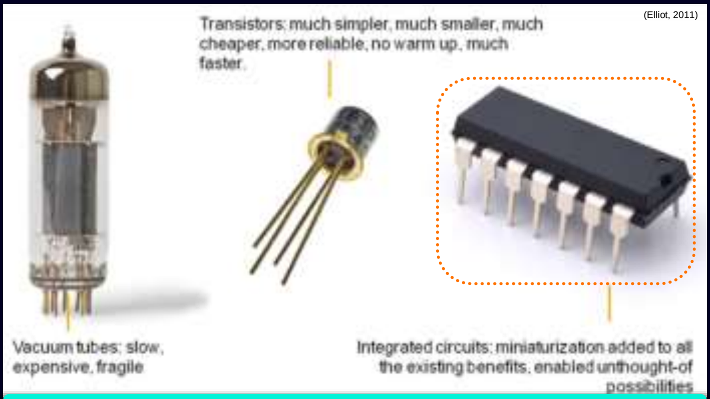
The OS is a little more complex, since it must manage multiple programs in different phases of execution. But overall performance of the system is increased. **[CLICK]**

One limitation of this system is that there was no interaction with the user. After loading a batch, all users could do was to wait to see the printed results of the execution.

[CLICK – Next Slide]

Generation 3: Ics and Multiprogramming

1965 – 1980



(Elliot, 2011)

Transistors: much simpler, much smaller, much cheaper, more reliable, no warm up, much faster.

Vacuum tubes: slow, expensive, fragile

Integrated circuits: miniaturization added to all the existing benefits, enabled unthought-of possibilities

Integrated circuits

- Allowed for miniaturization, performance and even greater reliability
- Software developed directly using terminals

OFF SCREEN

Generation 3 (1965-80) **[CLICK]**

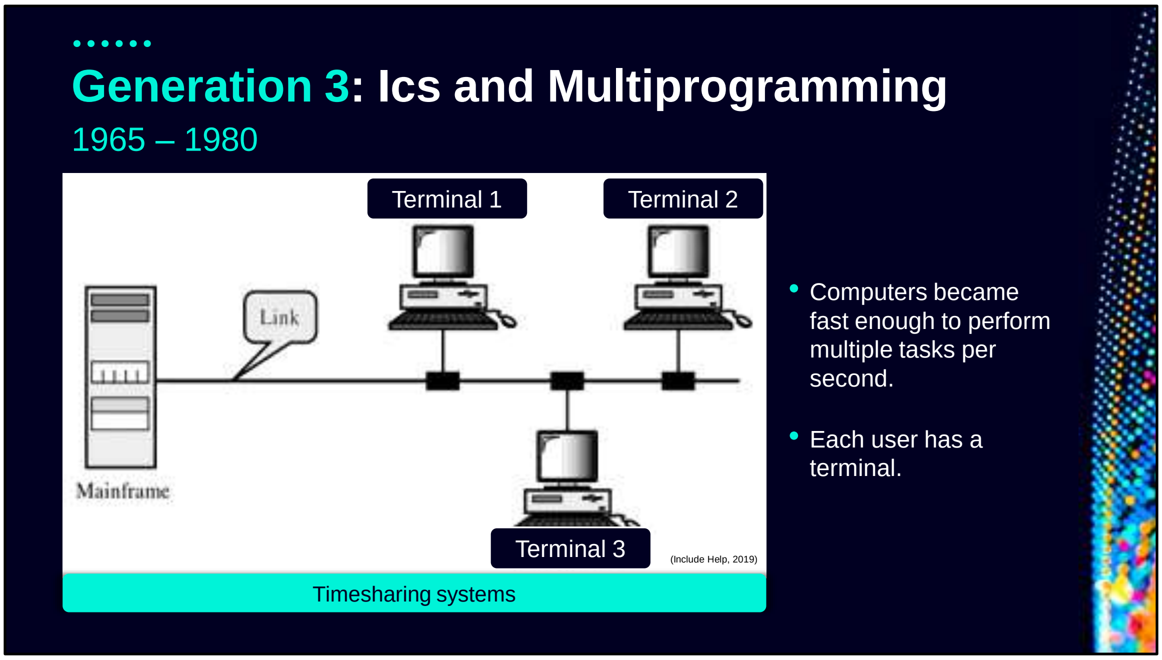
This generation is characterized by computers built with Integrated Circuits and users that were operators. **[CLICK]**

Integrated Circuits allowed for miniaturization, performance and even grater reliability. **[CLICK]**

Users/programmers developed software directly using terminals.

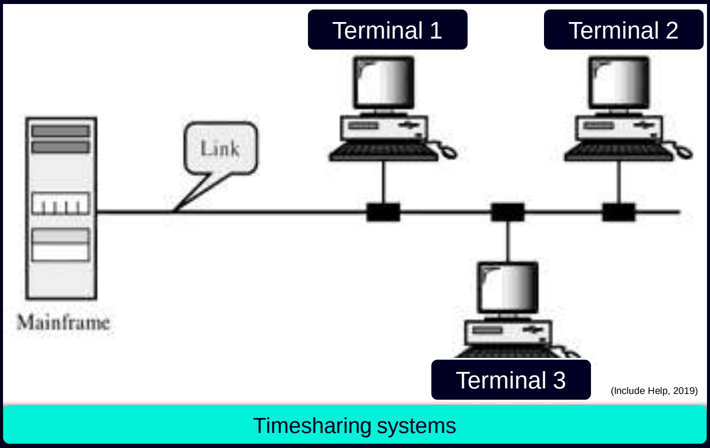
So, the figure of the operator, loading batches of perforated cards into the systems disappeared (the user is the operator!)

[CLICK – Next Slide]



Generation 3: Ics and Multiprogramming

1965 – 1980



- Computers became fast enough to perform multiple tasks per second.
- Each user has a terminal.

OFF SCREEN

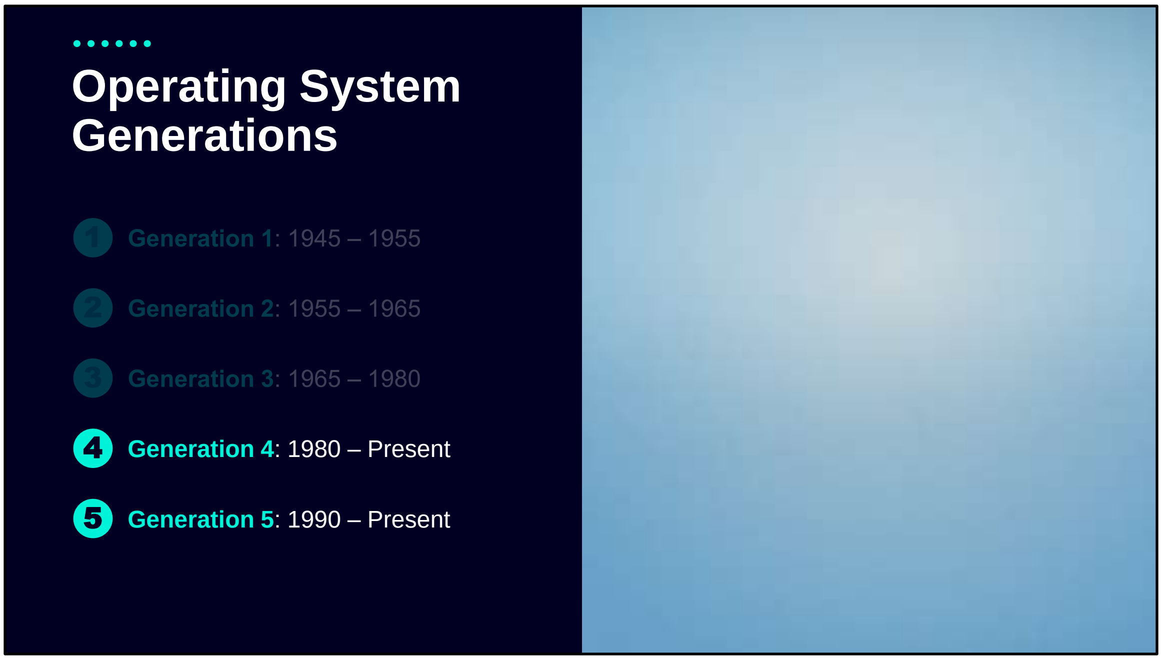
During this generation, Timesharing systems appeared. **[CLICK]**

Computers became fast enough to perform multiple tasks per second. This speed allowed large systems (mainframes) to be connected to multiple terminals where users could interact with the system, for example developing and running programs. **[CLICK]**

Each terminal/user would receive a small amount of processing time sequentially, however, that would happen multiple times per second.

As a result, users had the illusion of real-time interaction with the system, and the illusion that they were they were alone interacting with the system.

[CLICK – Next Slide]



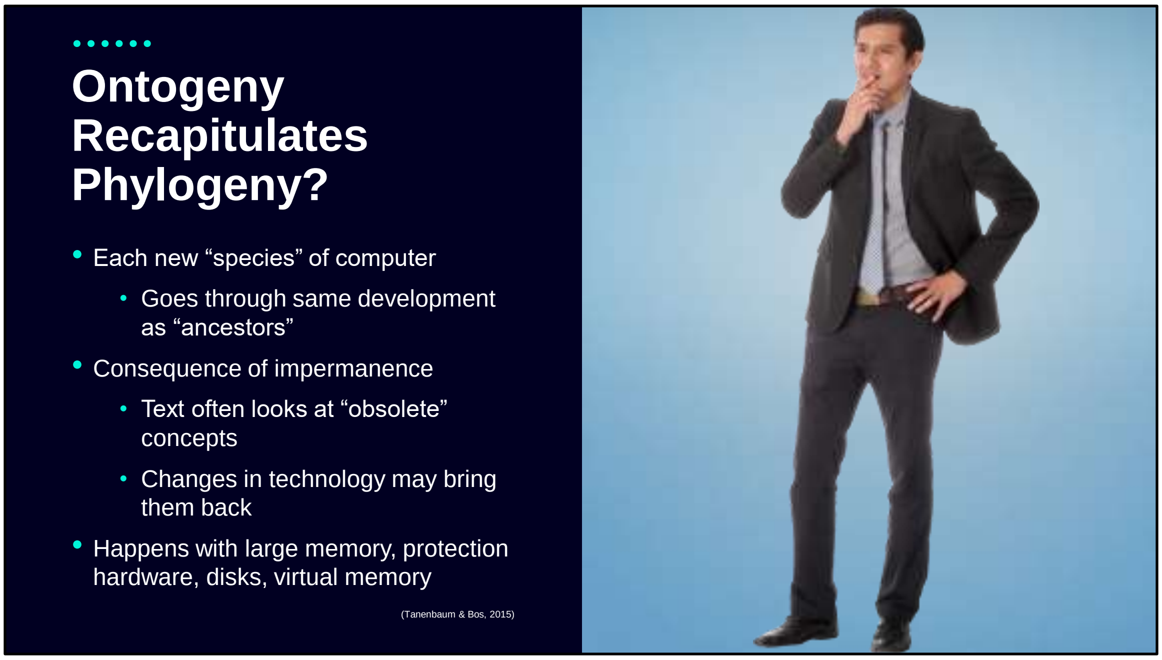
Operating System Generations

1. **Generation 1**: 1945 – 1955
2. **Generation 2**: 1955 – 1965
3. **Generation 3**: 1965 – 1980
4. **Generation 4**: 1980 – Present
5. **Generation 5**: 1990 – Present

ON SCREEN

We will not look into the details of the last two generations since we are still living them. They are characterized by personal computers and mobile computers. Additional generations have also been proposed such as a generation of Wearable computers.

[CLICK – Next Slide]



Ontogeny Recapitulates Phylogeny?

- Each new "species" of computer
  - Goes through same development as "ancestors"
- Consequence of impermanence
  - Text often looks at "obsolete" concepts
  - Changes in technology may bring them back
- Happens with large memory, protection hardware, disks, virtual memory

(Tanenbaum & Bos, 2015)

ON SCREEN

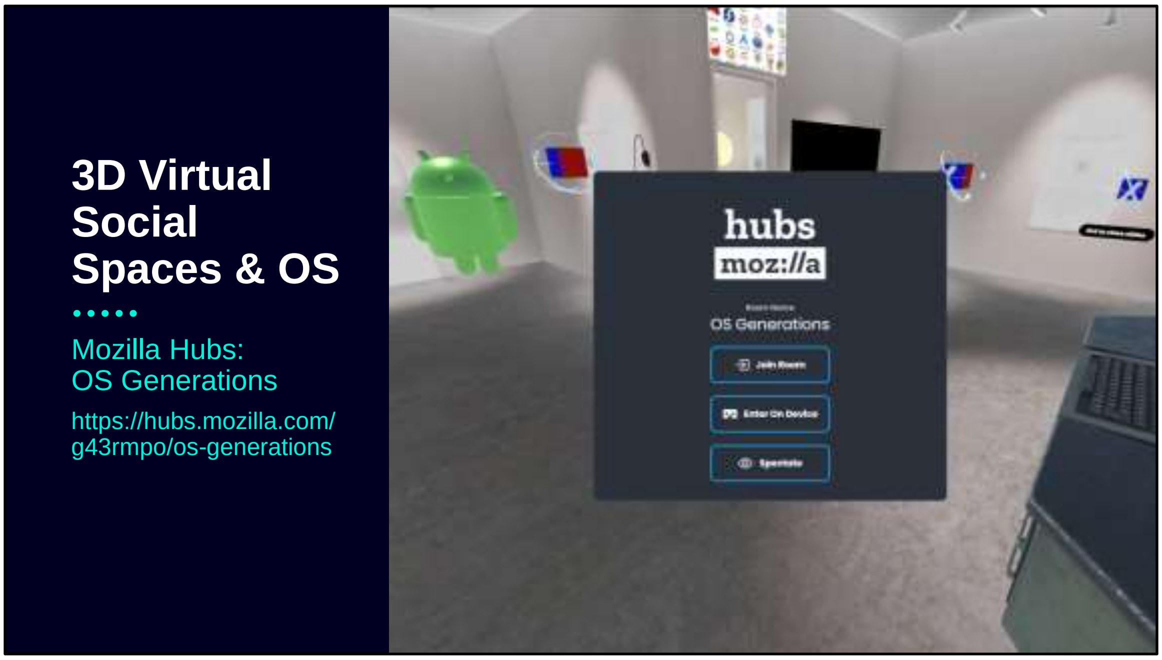
By looking at the generations of computers and operating systems we see a pattern in which limitations and constraints periodically emerge and are overcome. **[CLICK]**

Old computers had very limited memory and storage. Programmers of that era had to learn how to deal with those constraints. **[CLICK]**

As computers evolved, large amounts of memory and storage became available at lower cost. Those "old" techniques seamed obsolete. Until wearables and embedded computers became popular and those limitations and the strategies to overcome them became relevant again.

Put simply, it is important to learn about the strategies and approaches that surround OSs. Even if they don't seem immediately relevant anymore, they might become in the near future.

[CLICK – Next Slide]


3D Virtual Social Spaces & OS

Mozilla Hubs:
OS Generations

https://hubs.mozilla.com/g43rmpo/os-generations

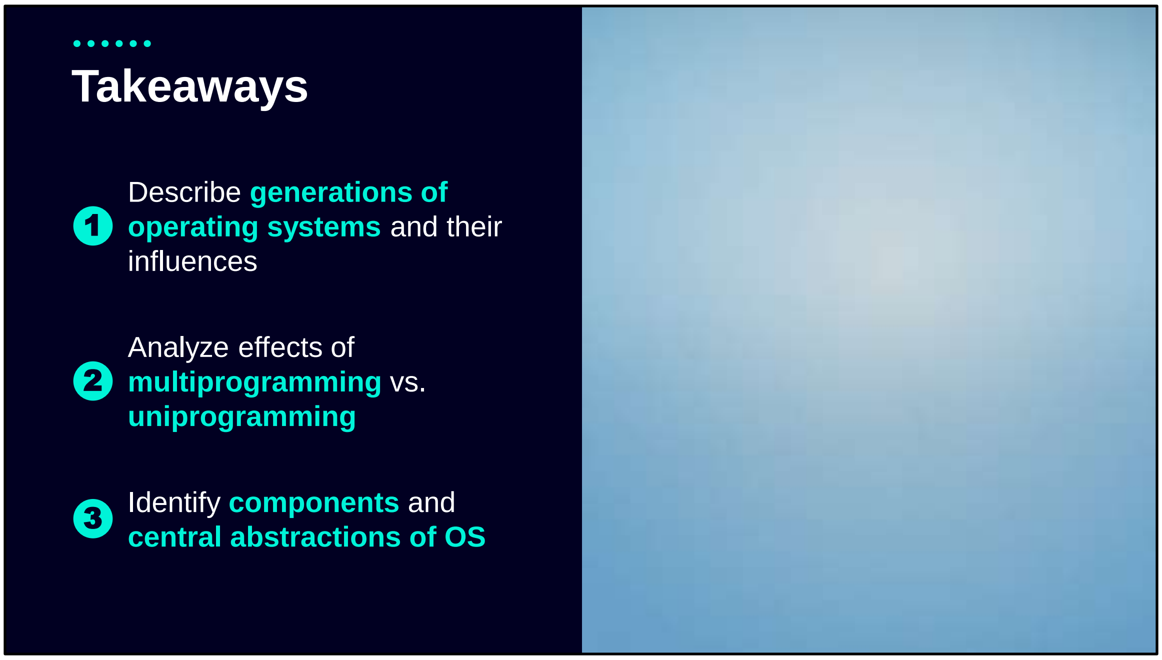
OFF SCREEN

Here is a Virtual Reality room where you can see some additional content.

You don’t need a VR headset. All you need is your computer.

This is additional material, not required. But it has some interesting content regarding the history of operating systems and the 5 generations.

[CLICK – Next Slide]



Takeaways

1. Describe **generations of operating systems** and their influences
2. Analyze effects of **multiprogramming** vs. **uniprogramming**
3. Identify **components** and **central abstractions of OS**

ON SCREEN

In this lecture, we discussed some of the history of operating systems. Now, you should be able to: **[CLICK]**

Describe the generations of operating systems and their influences; **[CLICK]**

Analyze effects of multiprogramming vs. uniprogramming; and **[CLICK]**

Identify components and central abstractions of OS.

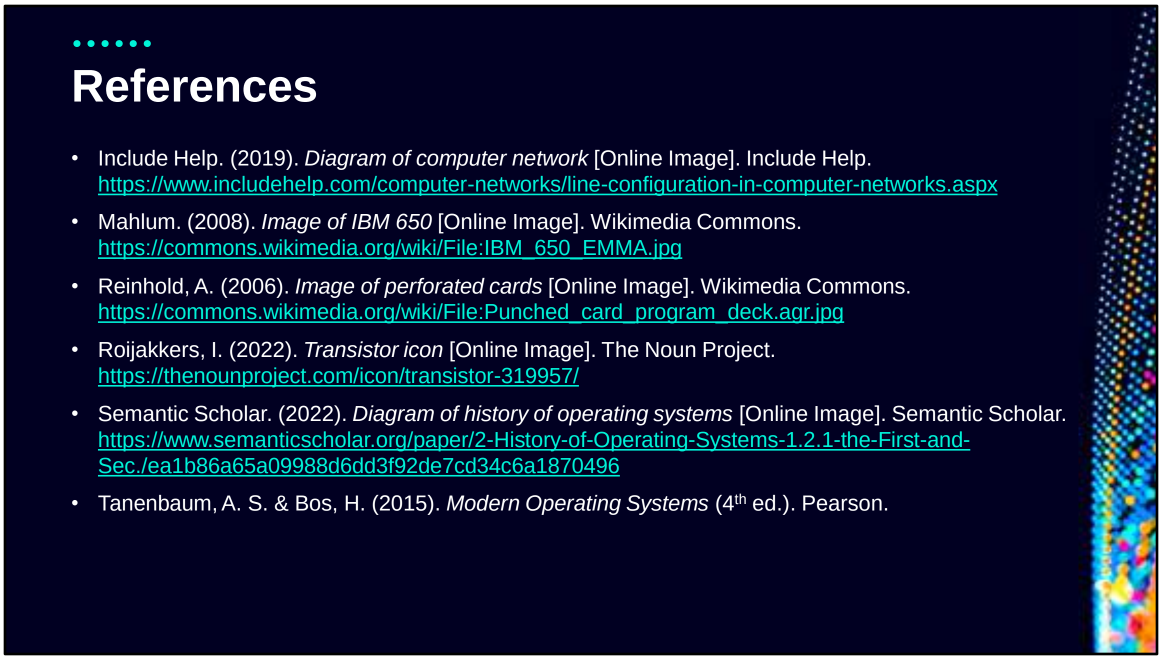
Thanks for watching! I'll see you next time.

[CLICK – Next Slide]

Thank You

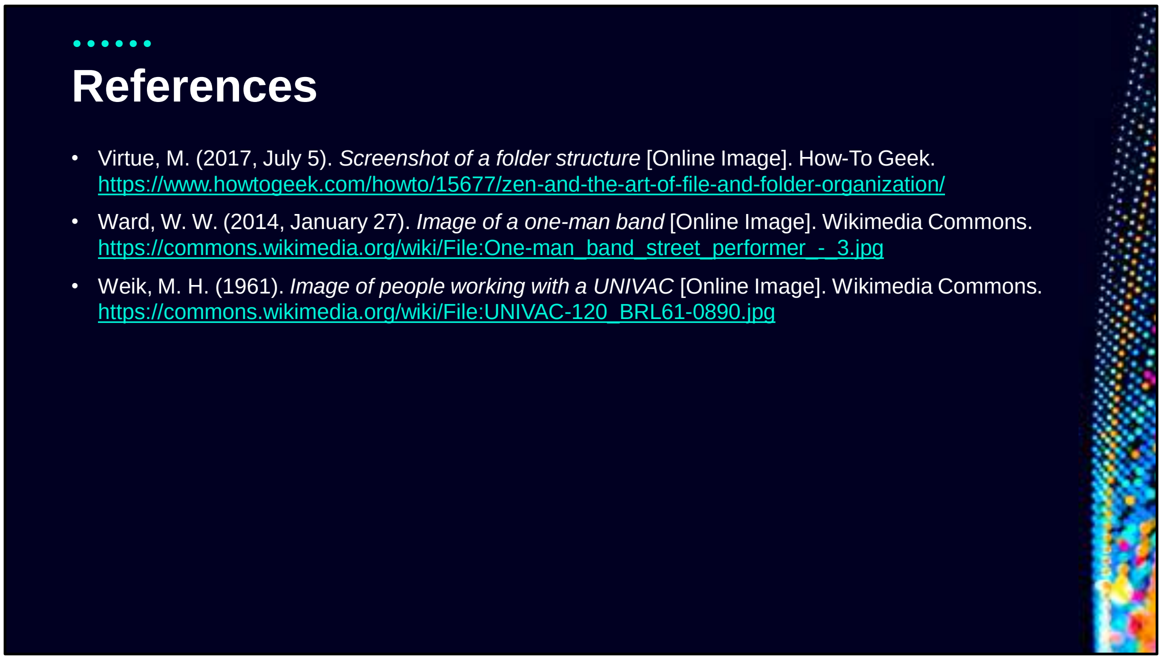
References

- Columbia University. (2001). *Image of a plugboard* [Online Image]. Columbia University. http://www.columbia.edu/cu/computinghistory/plugboard.html
- Cushing Memorial Library and Archives, Texas A&M. (2009). *Image of IBM 650* [Online Image]. Wikimedia Commons. https://commons.wikimedia.org/wiki/File:IBM_650_with_front_open.jpg
- Elliot, T. (2011). *Image of vacuum tubes, transistors, and circuits* [Online Image]. Timo Elliot. https://timoelliott.com/blog/2011/03/why-the-last-decade-of-bi-best-practice-architecture-is-rapidly-becoming-obsolete.html
- Fisher, D. (2008). *Image of IBM vacuum tubes* [Online Image]. Wikimedia Commons. https://commons.wikimedia.org/wiki/File:Ibm-tube.jpg
- Freedman, A. & Morrison, I. (2022). *Graphic illustration of how a computer stores bits* [Online Image]. Computer Language Company. https://www.computerlanguage.com/results.php?definition=bit
- IBM. (2022). *Drawing of early batch systems* [Online Image]. IBM. https://www.ibm.com/ibm/history/ibm100/us/en/icons/mainframe/breakthroughs/



References

- Include Help. (2019). *Diagram of computer network* [Online Image]. Include Help. https://www.includehelp.com/computer-networks/line-configuration-in-computer-networks.aspx
- Mahlum. (2008). *Image of IBM 650* [Online Image]. Wikimedia Commons. https://commons.wikimedia.org/wiki/File:IBM_650_EMMA.jpg
- Reinhold, A. (2006). *Image of perforated cards* [Online Image]. Wikimedia Commons. https://commons.wikimedia.org/wiki/File:Punched_card_program_deck.agr.jpg
- Roijakkers, I. (2022). *Transistor icon* [Online Image]. The Noun Project. https://thenounproject.com/icon/transistor-319957/
- Semantic Scholar. (2022). *Diagram of history of operating systems* [Online Image]. Semantic Scholar. https://www.semanticscholar.org/paper/2-History-of-Operating-Systems-1.2.1-the-First-and-Sec./ea1b86a65a09988d6dd3f92de7cd34c6a1870496
- Tanenbaum, A. S. & Bos, H. (2015). *Modern Operating Systems* (4th ed.). Pearson.



References

- Virtue, M. (2017, July 5). *Screenshot of a folder structure* [Online Image]. How-To Geek. https://www.howtogeek.com/howto/15677/zen-and-the-art-of-file-and-folder-organization/
- Ward, W. W. (2014, January 27). *Image of a one-man band* [Online Image]. Wikimedia Commons. https://commons.wikimedia.org/wiki/File:One-man_band_street_performer_-_3.jpg
- Weik, M. H. (1961). *Image of people working with a UNIVAC* [Online Image]. Wikimedia Commons. https://commons.wikimedia.org/wiki/File:UNIVAC-120_BRL61-0890.jpg