COMPS267F Chapter 1

Overview of Operating Systems

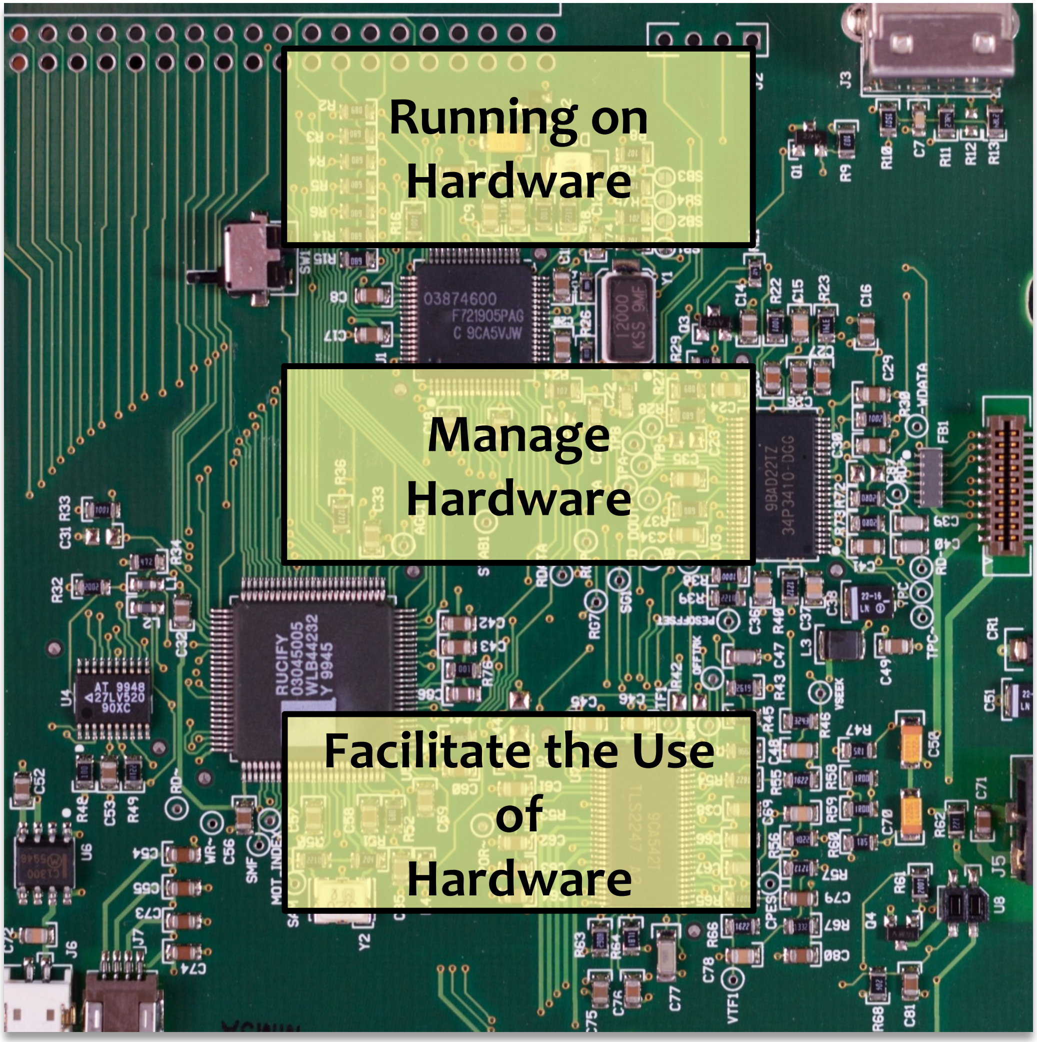
Dr. Andrew Kwok-Fai LUI

why studying OS?

My Short Answers

- A good time to ask this question
- A few answer are provided for you
  - One most important software
    - Hardware is nothing
  - A major achievement in computing history
    - Effective use of computing resources
  - Techniques developed for OS are general and useful
  - Analysis of techniques and critical thinking

What are OS?



Running on Hardware

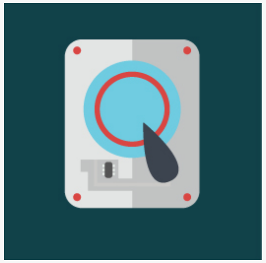
Manage Hardware

Facilitate the Use of Hardware

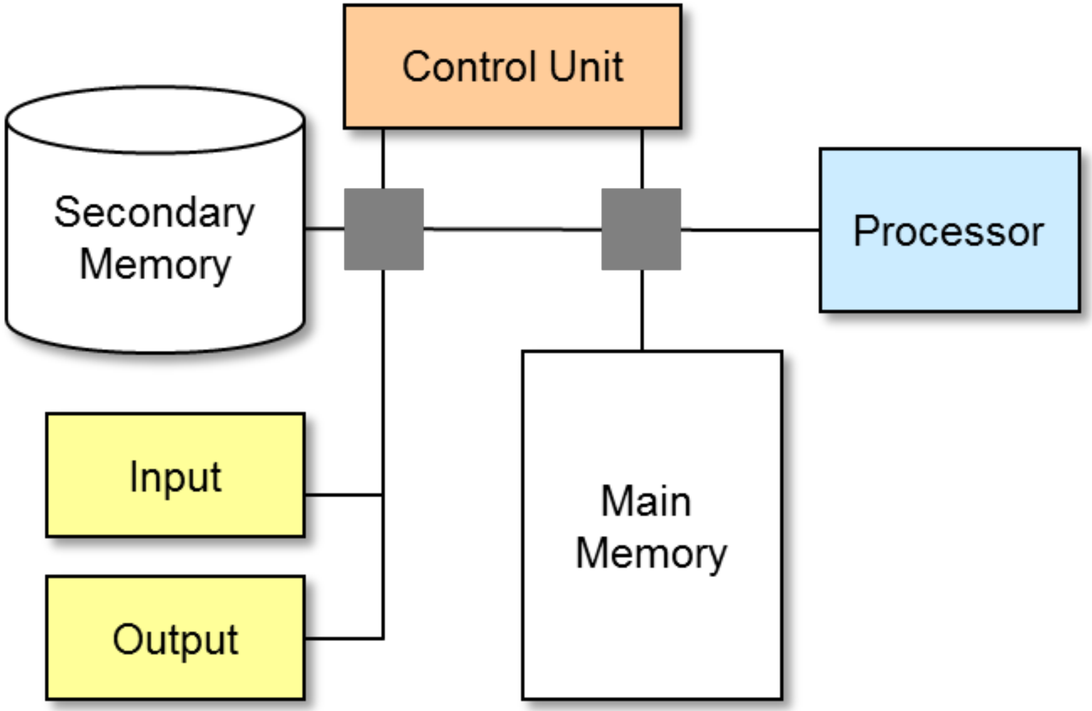
programmable computer systems

Programmability

- A computer that can be re-purposed
  - Game platform, word processor, slide presenter, …
  - The program running determines the purposes



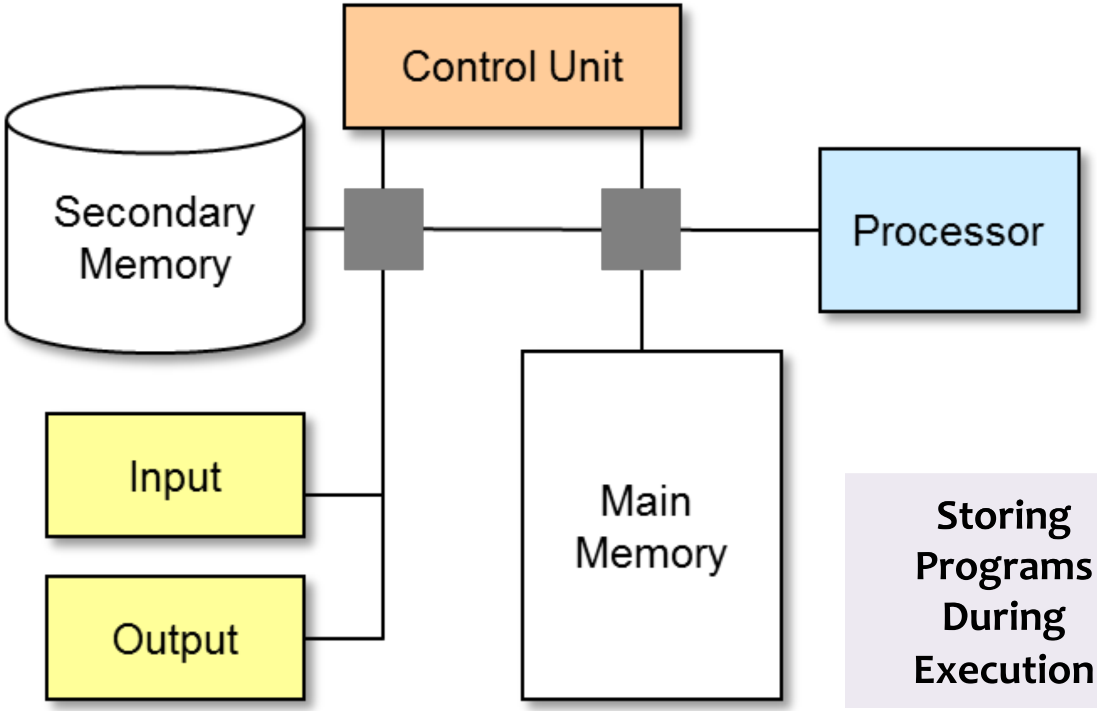
A Programmable Computer Systems



A Programmable Computer Systems

Storing Programs Long-Term

Loading Programs



Running Programs

Storing Programs During Execution

What is a Computer Program?

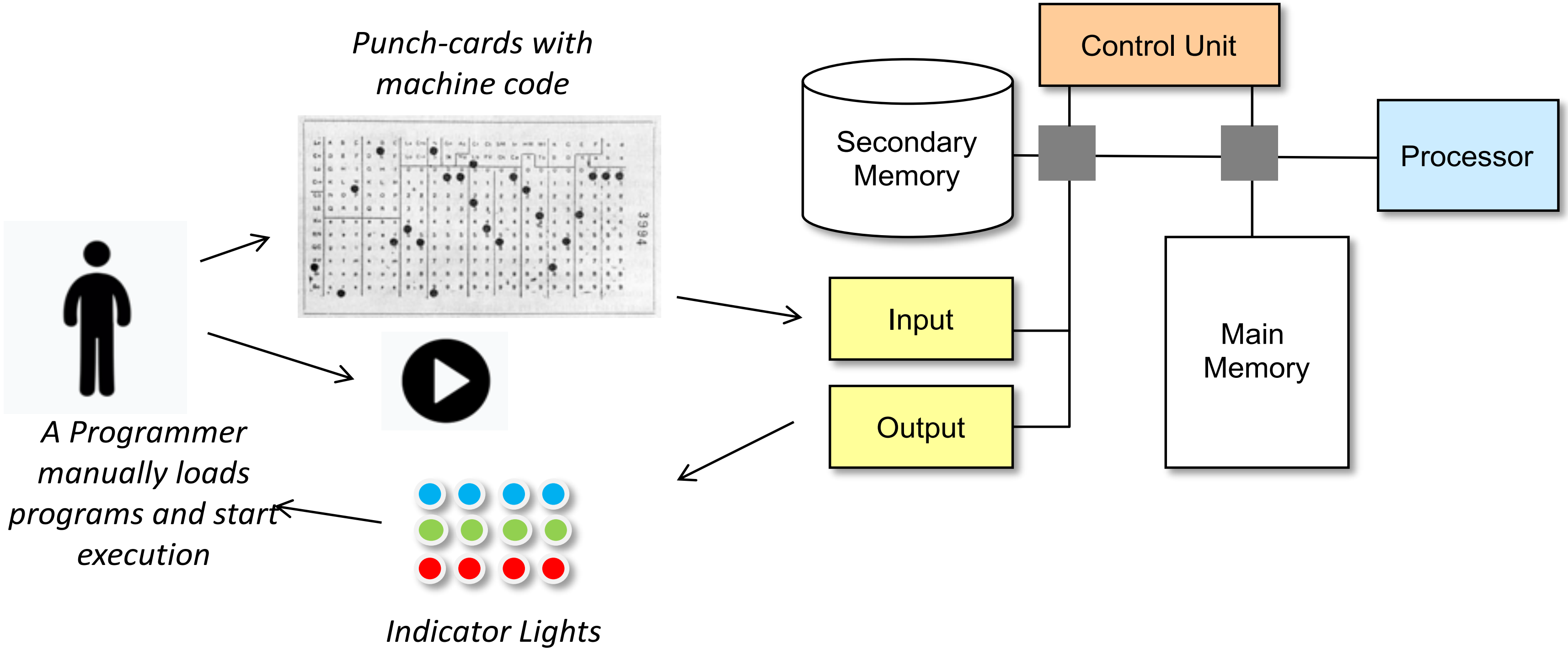
- Artefacts performs a purposefully designed sequence of operations
  - A programmable computer can do purposeful things
  - The purpose requires carefully designed sequence of operations
- What is a program?
  - Take over the control of a computer
  - The computer executes the instructions in programs
  - A program made up of instructions, each causing one or more computer operations

the need for OS

An OS-less Computer System

- How to use an OS-less computer system?

Serial Processing



Punch-cards with machine code

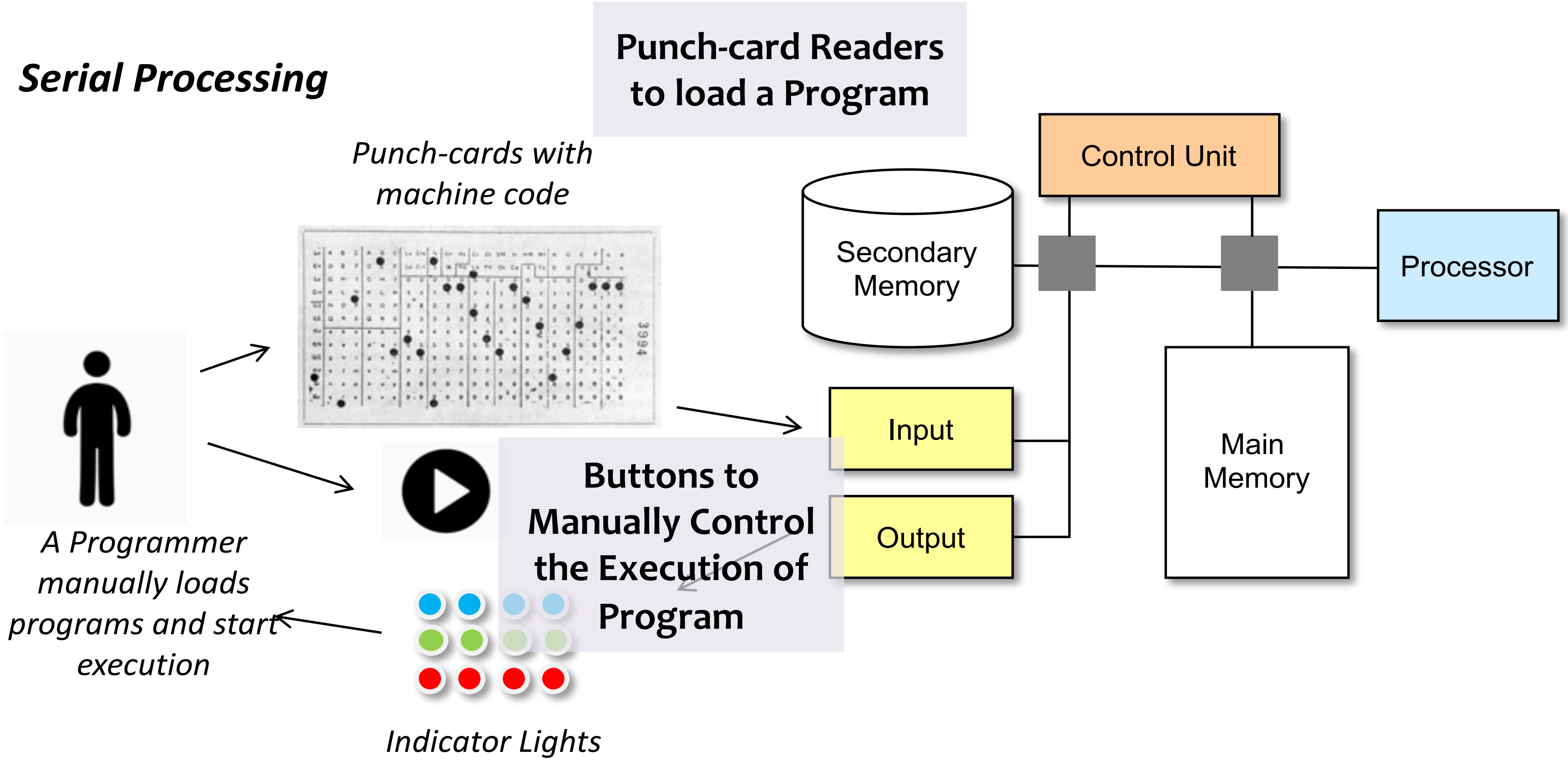
A Programmer manually loads programs and start execution

Indicator Lights

An OS-less Computer System

- It works well enough for making atomic bombs

Serial Processing



Punch-card Readers to load a Program

Punch-cards with machine code

Buttons to Manually Control the Execution of Program

A Programmer manually loads programs and start execution

Indicator Lights

Problems

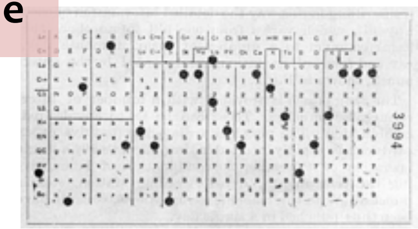
- Scheduling of usage and program setup time

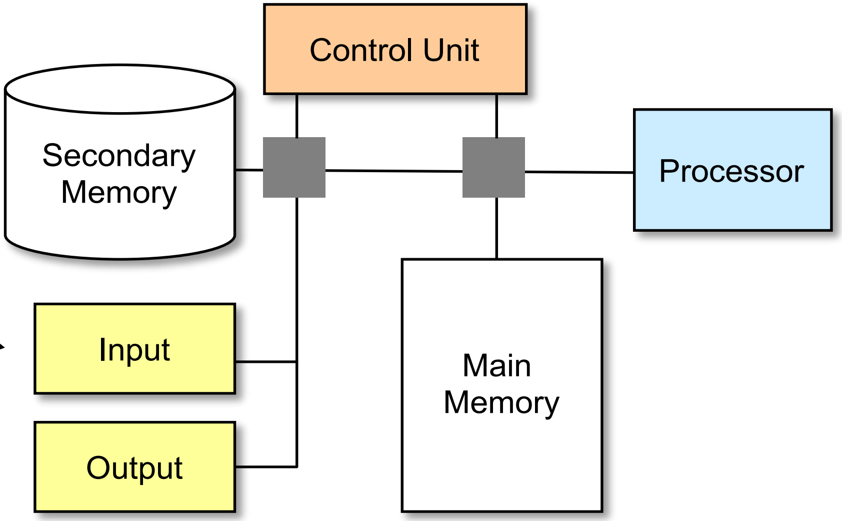
Some Scheduling is needed

Serial Processing

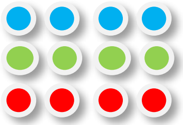
One Programmer can use it at a time

Punch-cards with machine code





A Programmer manually loads programs and start execution



Indicator Lights

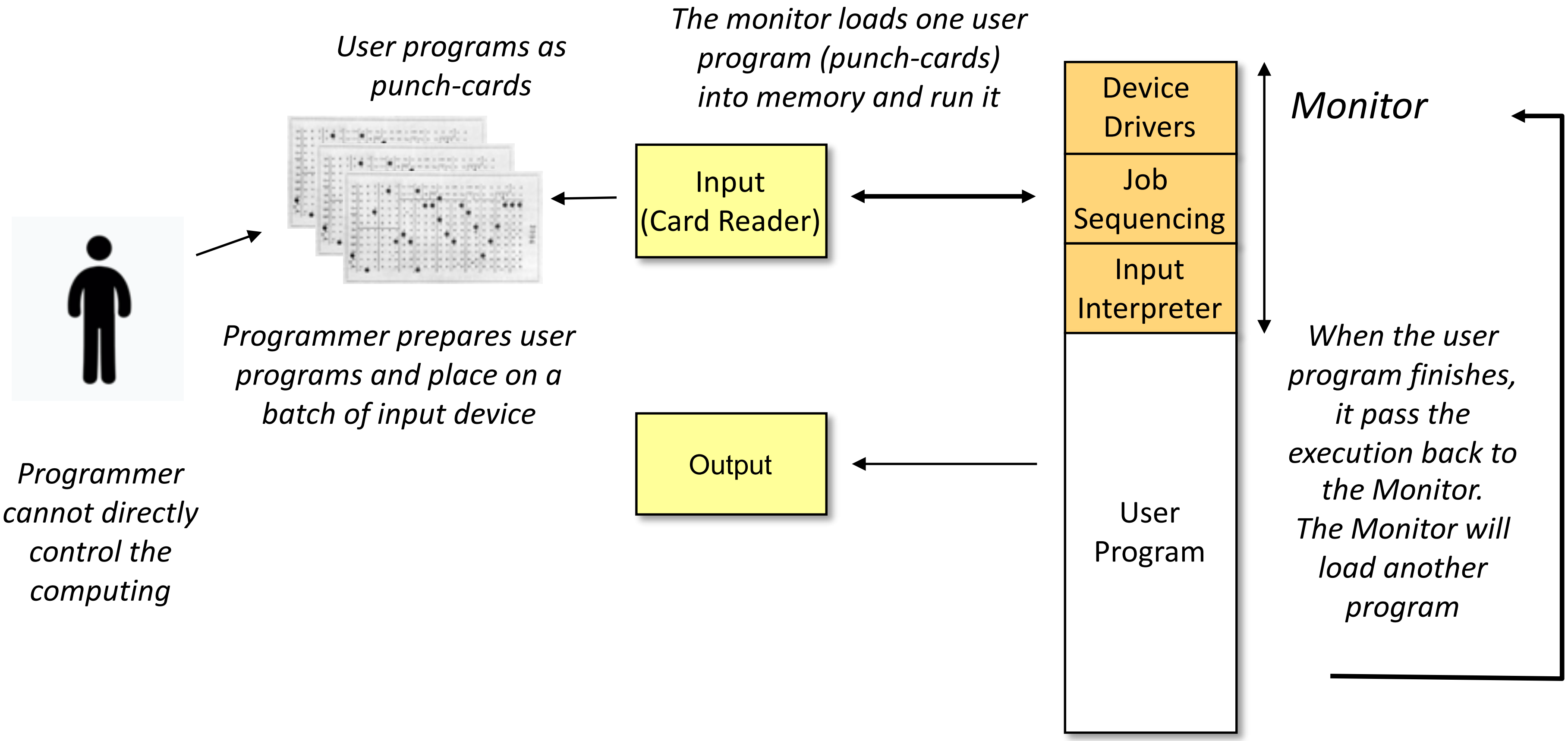
Loading programs and tidying up can take a lot of time

The computer is expensive, so reducing the setup time is important

Batch Systems

- Use automation to solve the two problems

Batch Processing



User programs as punch-cards

The monitor loads one user program (punch-cards) into memory and run it

Input (Card Reader)

Output

Device Drivers

Job Sequencing

Input Interpreter

User Program

Monitor

Programmer prepares user programs and place on a batch of input device

Programmer cannot directly control the computing

When the user program finishes, it pass the execution back to the Monitor. The Monitor will load another program

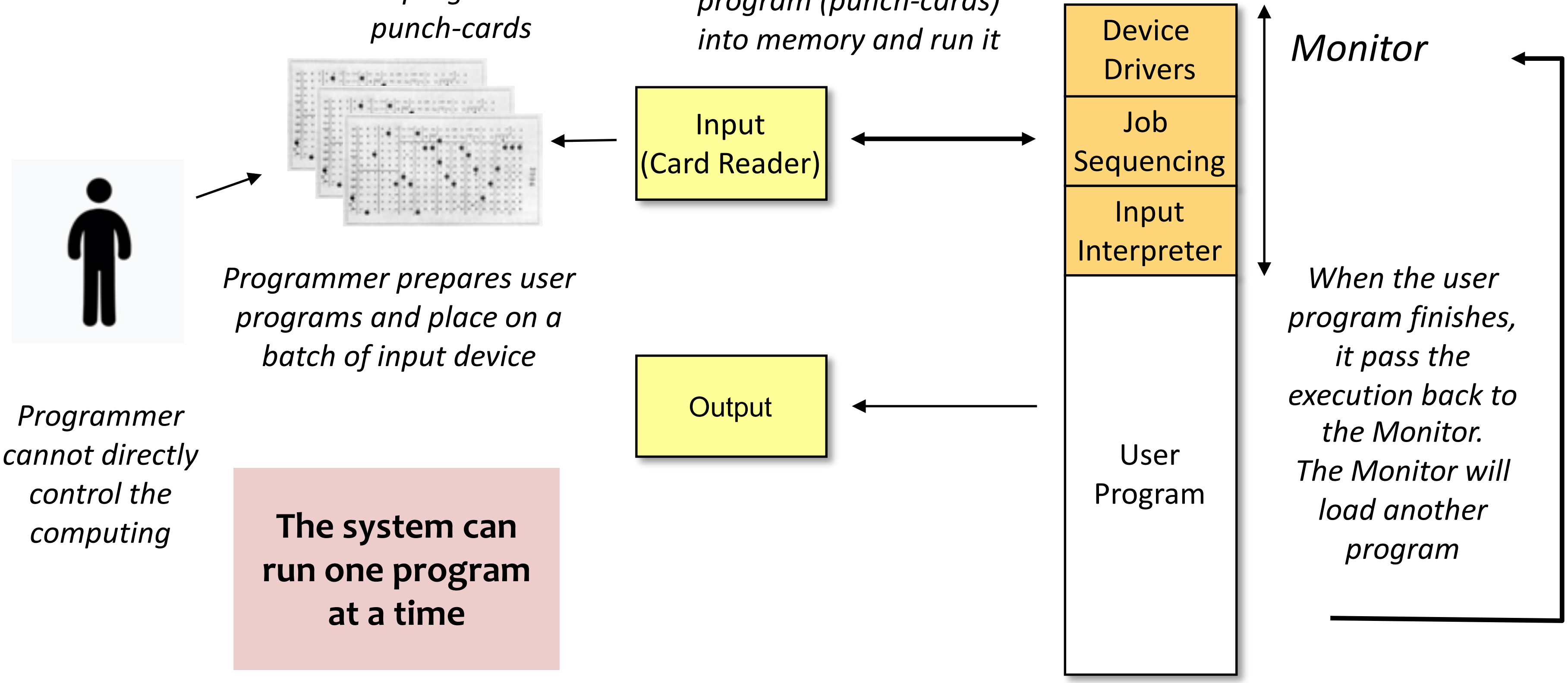
The Monitor

- A program residing in the computer memory
- Automates the setup and execution of programs
- Called batch processing
  - The Monitor is a early form of OS called Batch OS

Batch Systems

- Use automation to solve the two problems

Batch Processing



User programs as punch-cards

The monitor loads one user program (punch-cards) into memory and run it

Programmer prepares user programs and place on a batch of input device

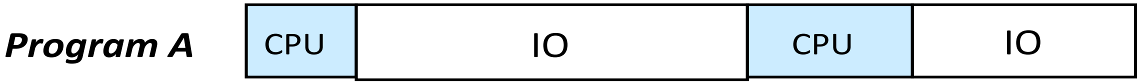
Programmer cannot directly control the computing

The system can run one program at a time

When the user program finishes, it pass the execution back to the Monitor. The Monitor will load another program

Uni-Programming

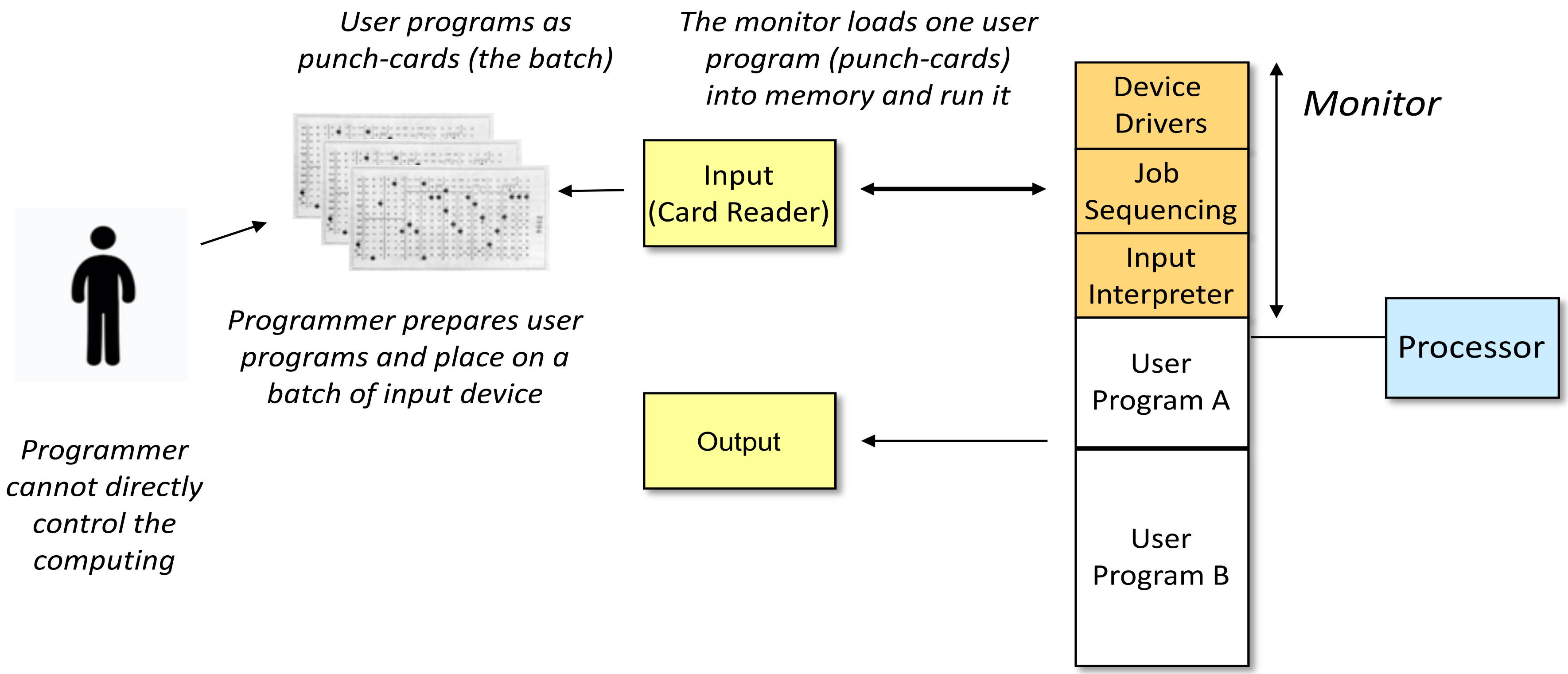
- Single programming (another name)
  - Only one program is running
  - Another program cannot run until the first program has finished
- Problems
  - The processor often got nothing to do
  - Most programs do not use the processor all the time
    - Doing I/O, reading data or writing data
  - The process should be made to do more work



Multi-Programmed Batch Systems

- Use automation to solve the two problems

Multiprogrammed Batch Processing



User programs as punch-cards (the batch)

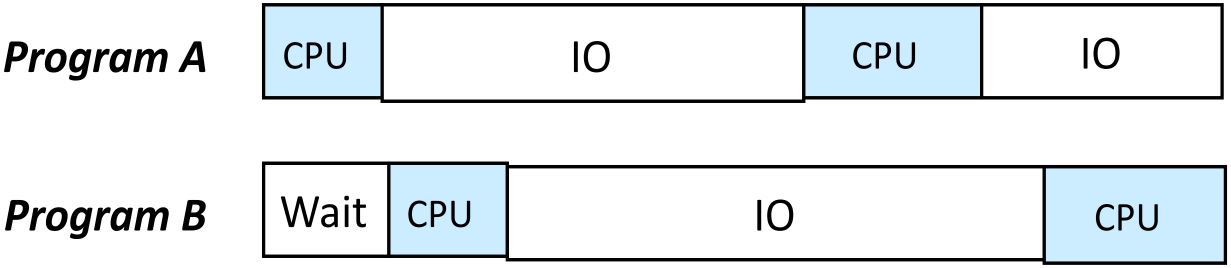
The monitor loads one user program (punch-cards) into memory and run it

Programmer prepares user programs and place on a batch of input device

Programmer cannot directly control the computing

Multi-Programming

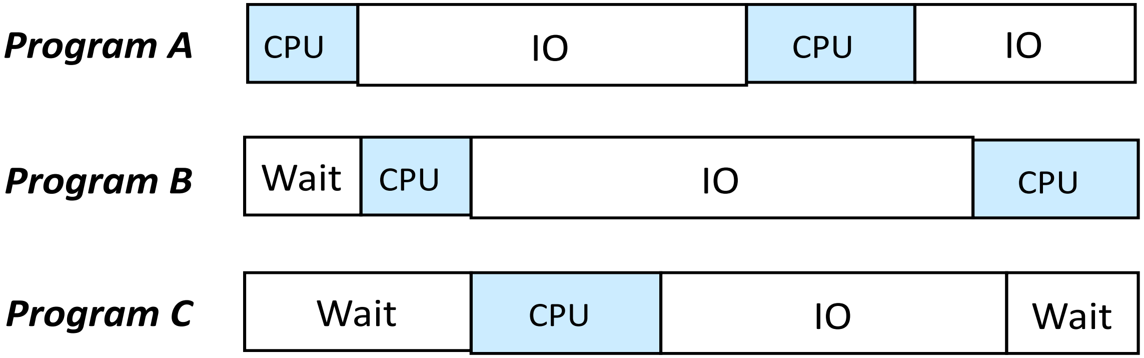
- The processor can execute another program if the current program is doing I/O



Program A: CPU | IO | CPU | IO

Program B: Wait | CPU | IO | CPU

Processor utilization can be kept at a higher level when there are more programs



Program A: CPU | IO | CPU | IO

Program B: Wait | CPU | IO | CPU

Program C: Wait | CPU | IO | Wait

Multi-Programmed Batch Systems

- Technical challenges

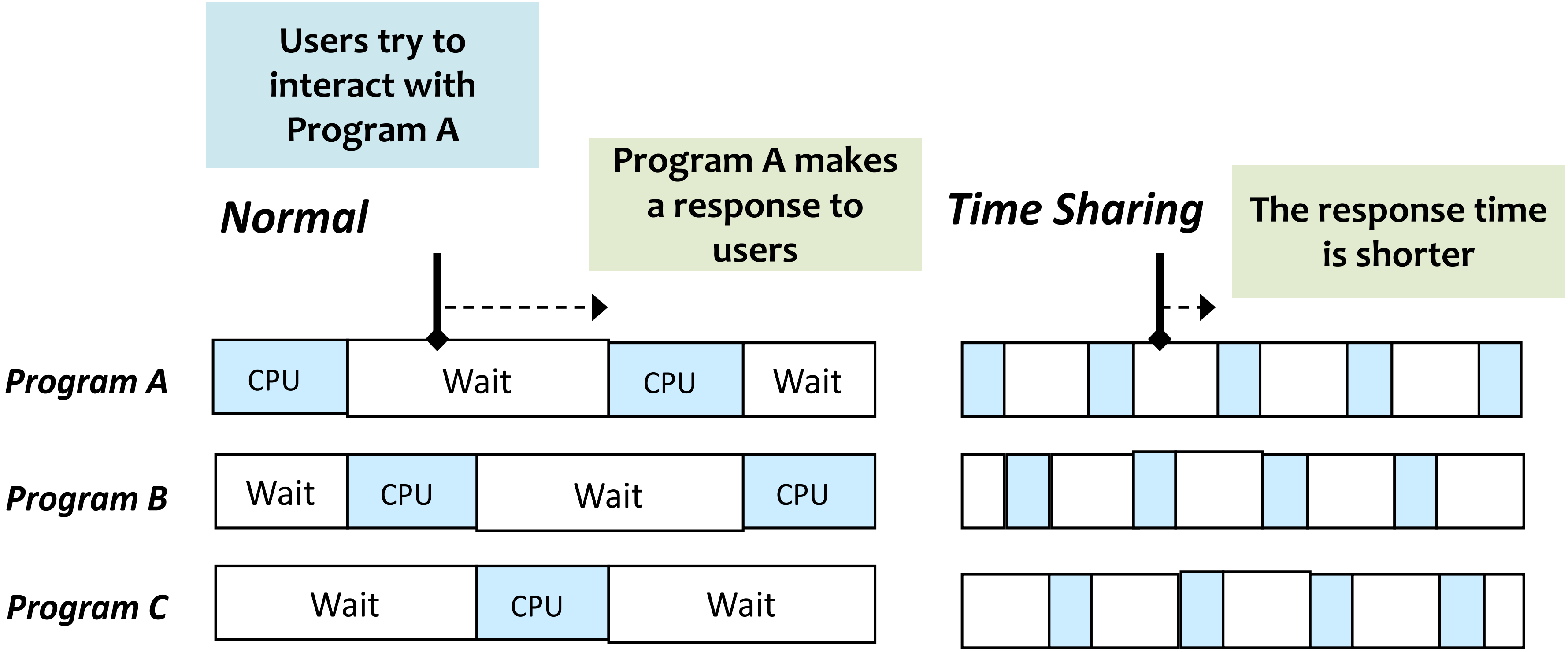
Benefits of Multiprogramming	Challenges to Overcome
Convenience to users. Able to do multiple tasks at the same time (multi-tasking)	Running multiple programs require a memory that is large enough to load multiple programs.
Increases the utilization of processors	Memory has to be managed so that each program can have its own space
Efficient in process execution	Memory of each program has to be secured to prevent other programs from attacking
	A large of memory is needed for effective multi-programming
	The monitor needs to manage the job scheduling effectively

Time Sharing Systems

- Interactions between users and computers are necessary
  - Batch processing does not allow user interaction
  - Response time
  - Users are unhappy if computer response is slow
- Time sharing systems allow for quicker responses
  - Now the standard for modern computing

Time Sharing Systems

- Shorter response time due to more frequent but shorter running of Processor for each program



Users try to interact with Program A

Program A makes a response to users

Normal

Time Sharing

The response time is shorter

Program A

Program B

Program C

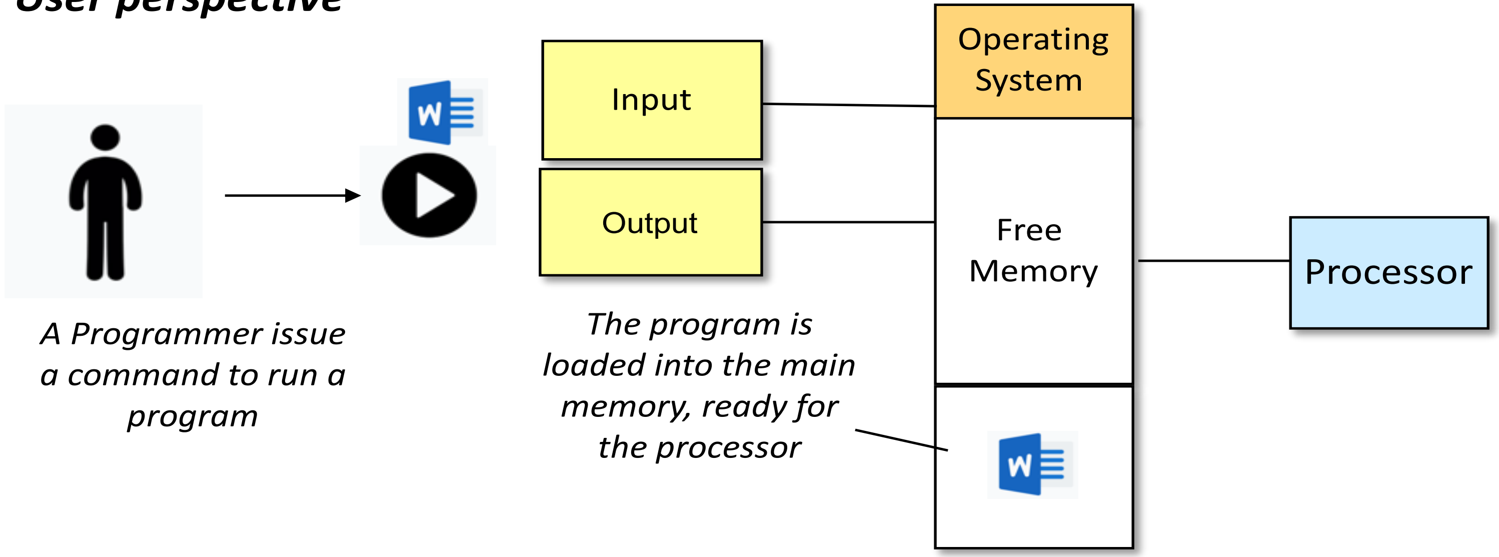
what is running a program

What is Running a Program?

- Can you give me an answer now?
- There are two different answers actually
  - From two different perspective
    - User perspective: Issued a command to execute a program
    - Processor perspective: Executing the instruction of a program
- They are not the same
  - The OS understands the difference between the two perspectives

What is Running a Program?

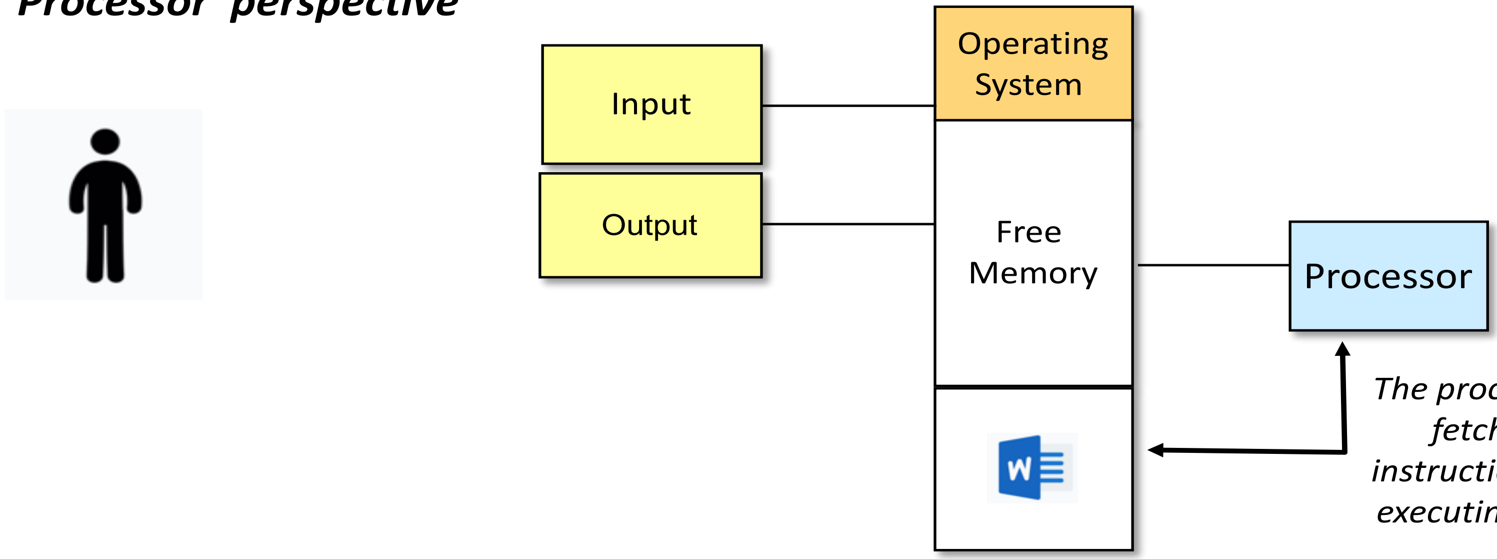
User perspective



A Programmer issue a command to run a program

The program is loaded into the main memory, ready for the processor

Processor perspective



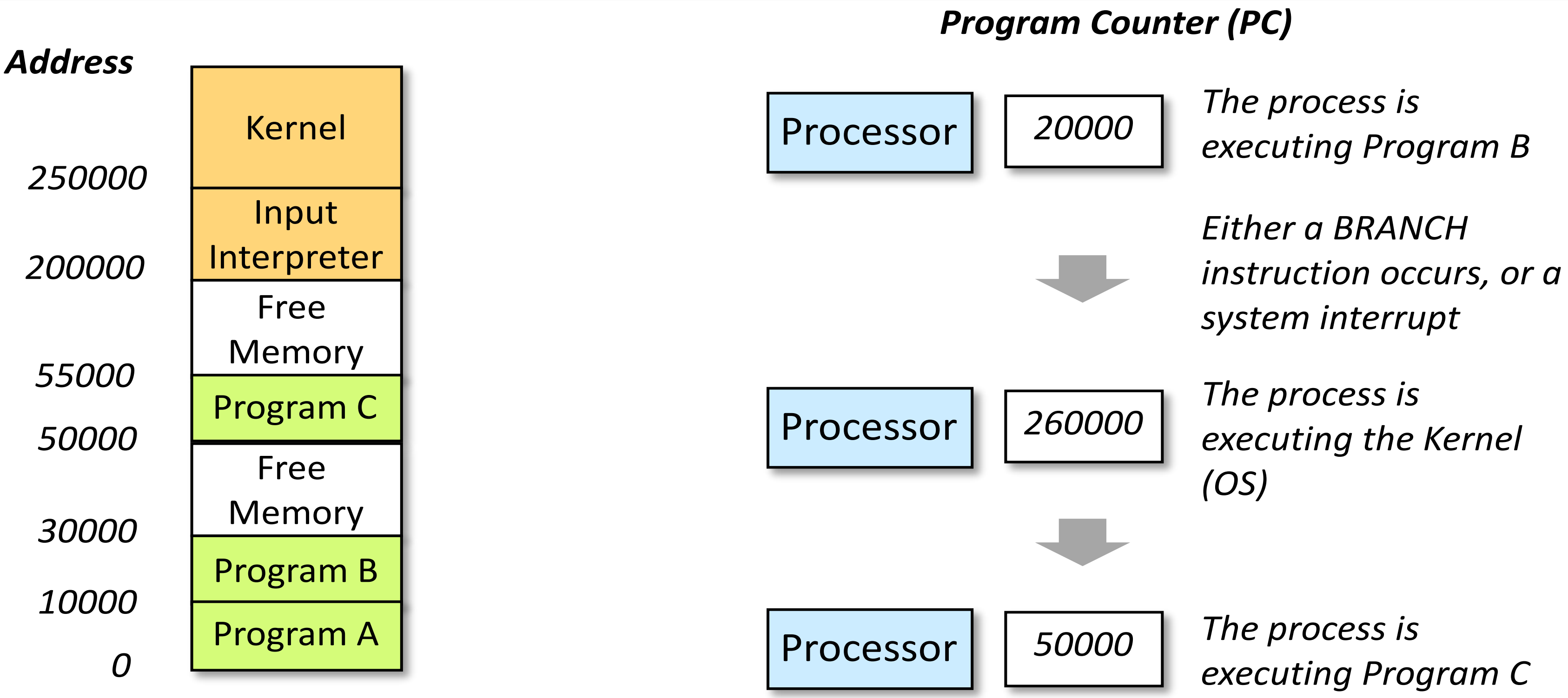
The processor is fetching instructions and executing them

What is Running a Program by the Processor?

- Executes the instructions of difference programs
  - In a multi-programming systems
  - This moment, it executes a program
  - Next moment, it executes another program

What is Running a Program by the Processor?

- How the processor changes the program it is executing?
  - The Program Counter (PC)



Address

Address	
	Kernel
250000	Input Interpreter
200000	Free Memory
55000	Program C
50000	Free Memory
30000	Program B
10000	Program A
0	

Program Counter (PC)

Processor 20000 — *The process is executing Program B*

Either a BRANCH instruction occurs, or a system interrupt

Processor 260000 — *The process is executing the Kernel (OS)*

Processor 50000 — *The process is executing Program C*

The Hazards of the BRANCH Instruction

- The PC can only be changed by the BRANCH instructions (and similar instructions)
  - Any other possible way for the PC to change?
- Must be restricted due to security reasons
  - Depends on who you are (what program you are)
  - OS can freely change the programs
    - Start running a user program after loading
    - Resume running a user program after being suspended
  - Other user programs do not enjoy the same privilege
    - Can only jump to addresses of the program itself
    - The address of OS function calls

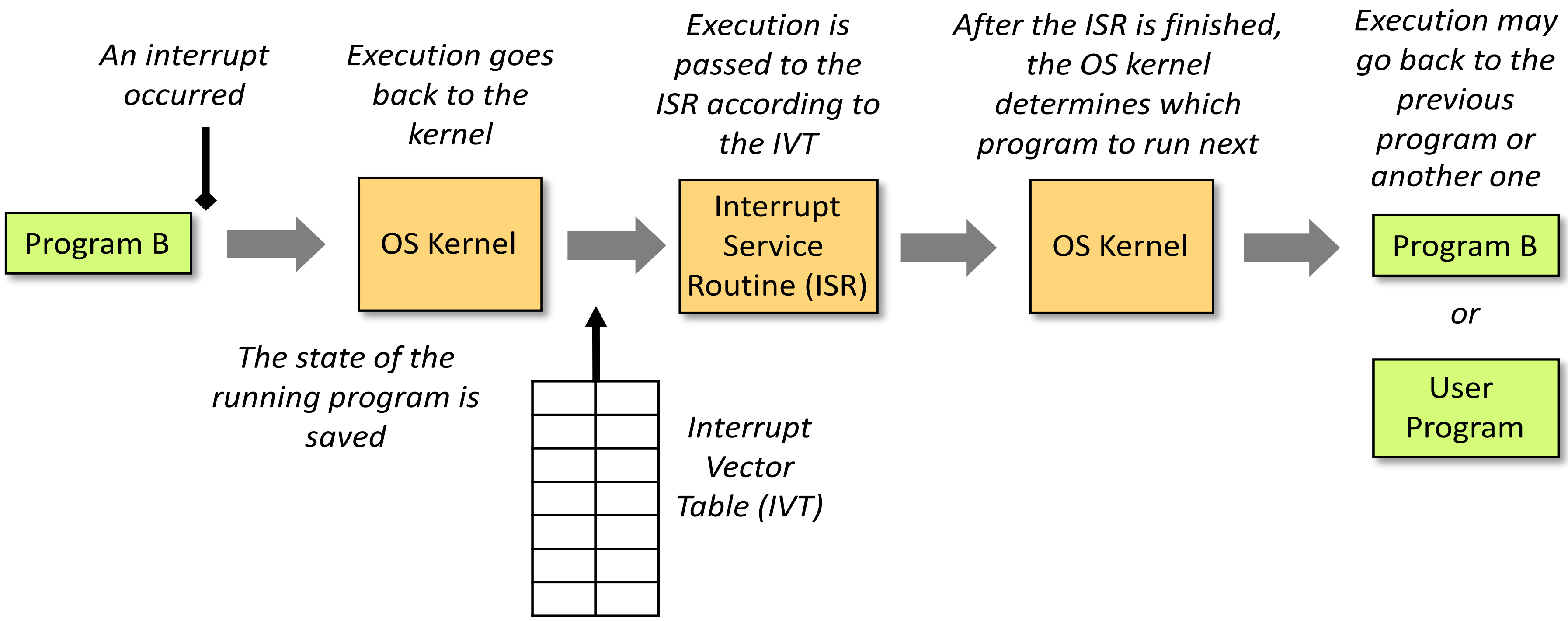
system interrupts

System Interrupts

- The most important mechanism in a running computer system
  - It is an event or signal
  - The system control goes back to the OS
    - The program counter is set to an address of the OS program

System Interrupts

Handling of Interrupts



An interrupt occurred

Program B

Execution goes back to the kernel

OS Kernel

The state of the running program is saved

Execution is passed to the ISR according to the IVT

Interrupt Service Routine (ISR)

Interrupt Vector Table (IVT)

After the ISR is finished, the OS kernel determines which program to run next

OS Kernel

Execution may go back to the previous program or another one

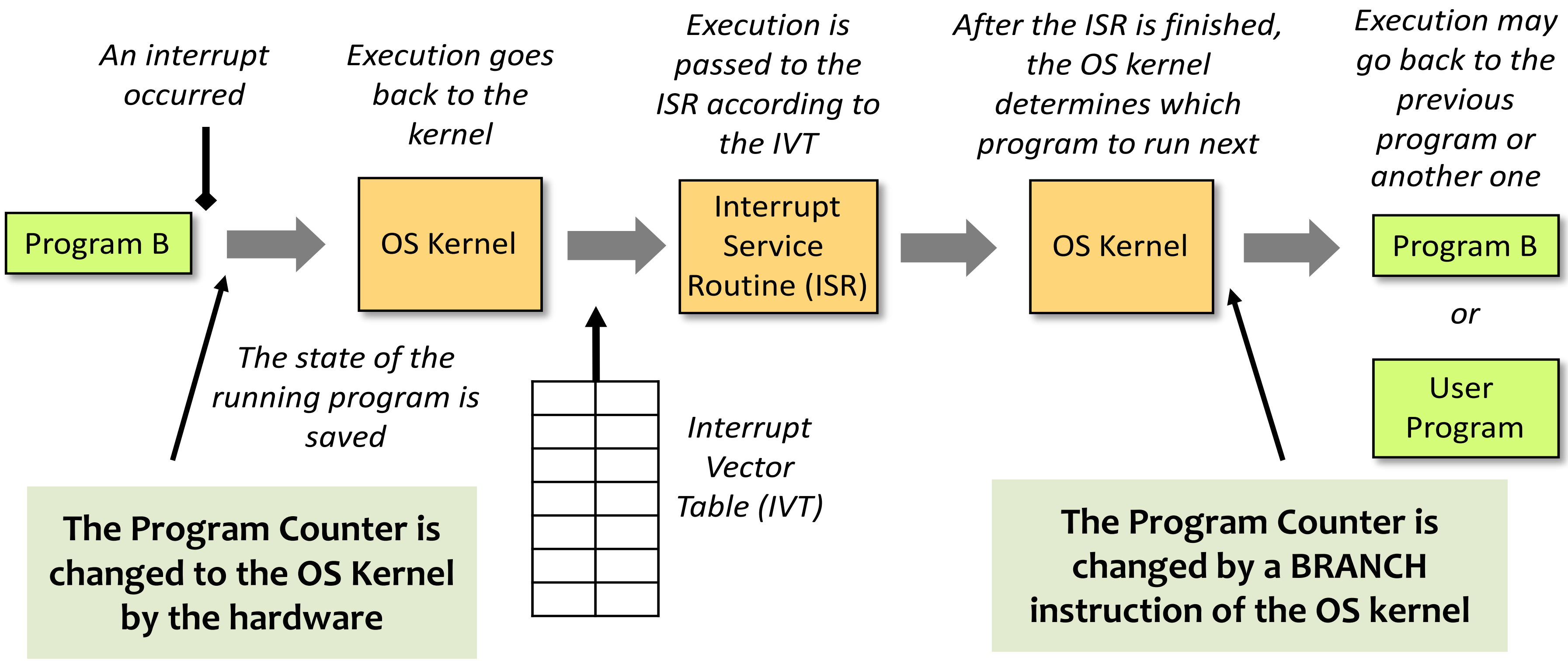
Program B

or

User Program

System Interrupts

Handling of Interrupts



Program B

An interrupt occurred

The state of the running program is saved

The Program Counter is changed to the OS Kernel by the hardware

Execution goes back to the kernel

OS Kernel

Interrupt Vector Table (IVT)

Execution is passed to the ISR according to the IVT

Interrupt Service Routine (ISR)

After the ISR is finished, the OS kernel determines which program to run next

OS Kernel

The Program Counter is changed by a BRANCH instruction of the OS kernel

Execution may go back to the previous program or another one

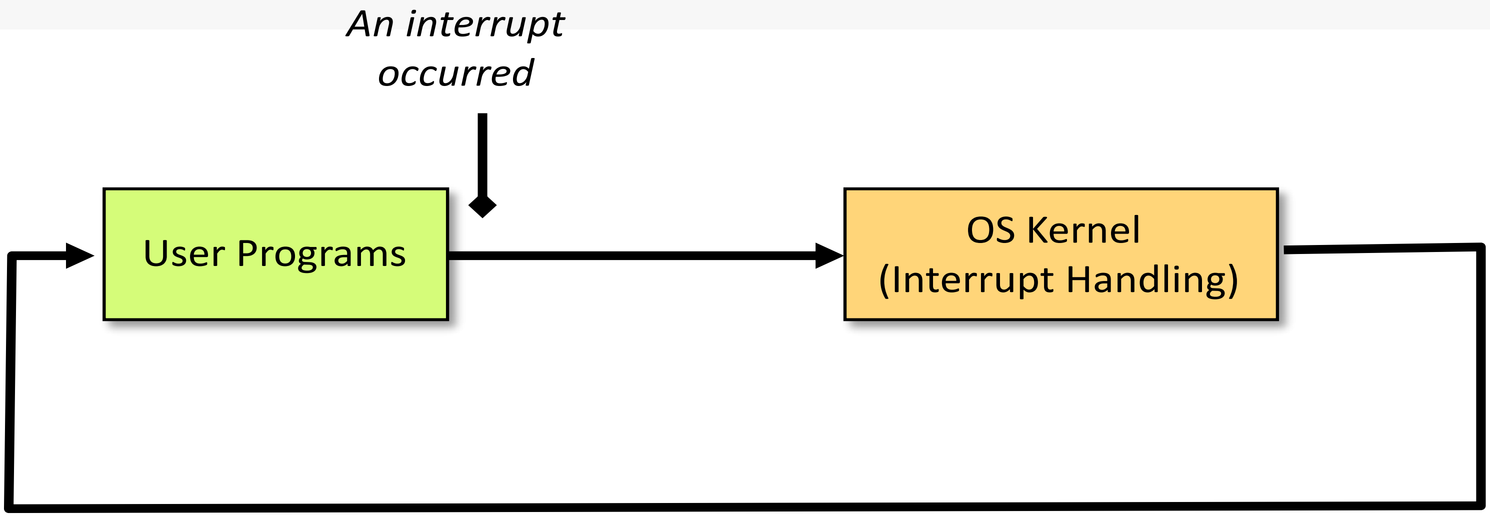
Program B

or

User Program

Importance of System Interrupts for Computer Systems

- The OS can regain control
  - Not possible to know how long a user program will run
  - Interrupts force suspension of user programs
- OS can deal with any issue
- Modern computer systems run in a cycle



Types of Interrupts

- Indicates something is happening or has happened
- Three types of interrupts
  - IO Interrupts or external interrupts
  - Internal or hardware interrupts
    - System clocks, serious errors
  - Software interrupts
    - User programs

Interrupt Service Routine (ISR)

- Interrupt Service Routine is a function that is designed to deal with a particular type of interrupts.
- The addresses of each ISR are stored in the Interrupt Vector Table (IVT)
  - Maps the interrupt type to the address of ISR.

security issues in multi-programming systems

Security Issues

- Multiprogramming systems execute many user programs at a time
  - User programs should not be trusted
  - May attack the OS
  - May attack other user programs

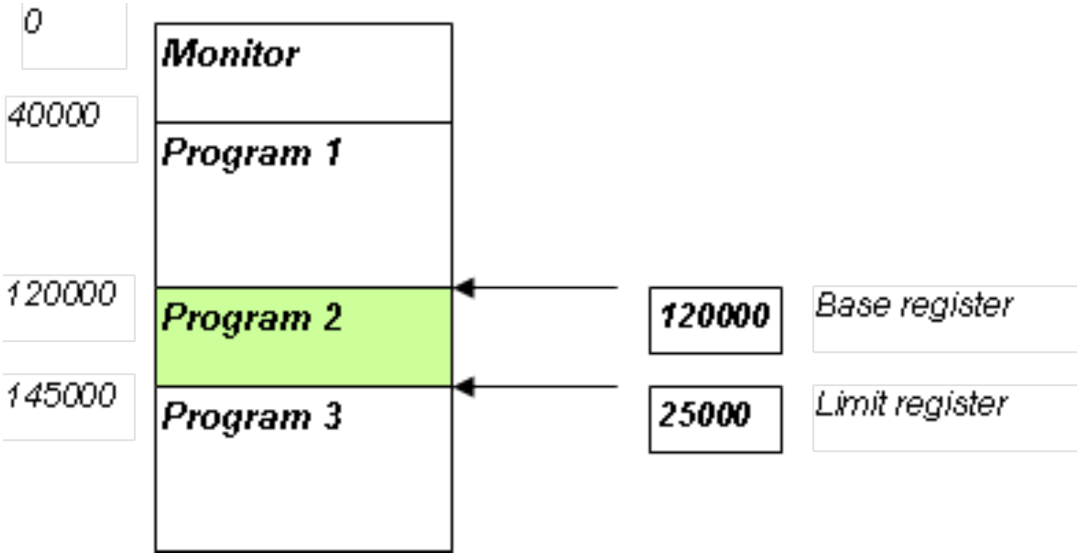
Dual-Mode Protection

- The system can be in one of two modes
  - User mode
    - Running user programs – limited privileges
  - Monitor mode
    - Can execute privileged instructions that can cause harm
- A monitor bit in the hardware that indicates the state
  - Hardware is safer and prevent attack
  - OS is in control of the computer system - monitor mode
  - Before OS pass control to a user program - switches to user mode
  - System call allows a user program asking the OS to perform privileged instructions in monitor mode

Memory Protection

- Preventing mishaps
  - Examples: accessing another program’s data or modifying the interrupt vector
  - Define legal memory addresses
    - Range of must be managed by the OS
    - A base register and a limit register

Memory Protection



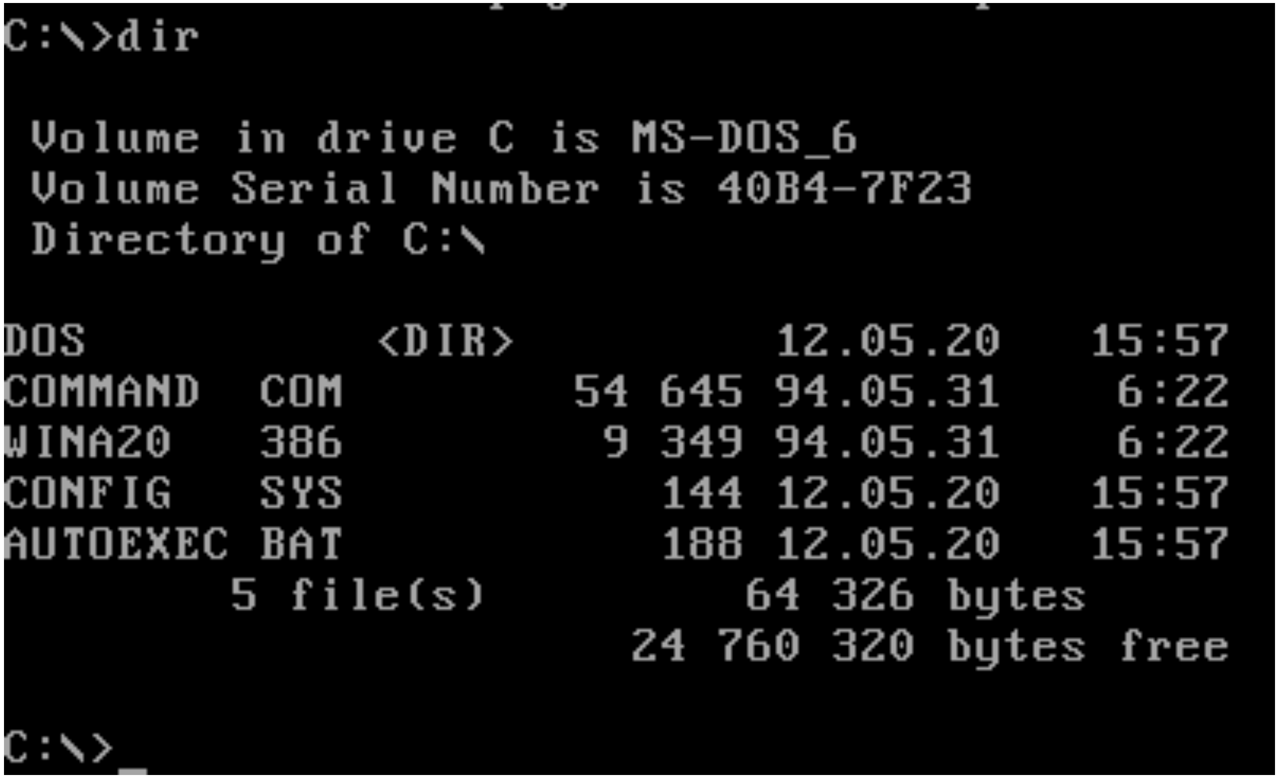
CPU Protection

- OS maintaining control of the computer systems
  - OS to regain control in regularly time intervals
  - A timer for user program
    - Time is up - interrupt is generated - OS regains control.
    - OS assigns a timer when a program is first executed.
  - The timer interrupts every second and the counter is decreased

case study: pseudo-multitasking in DOS

DOS

- Disk Operating System (DOS) is a famous operating system for IBM PC compatibles



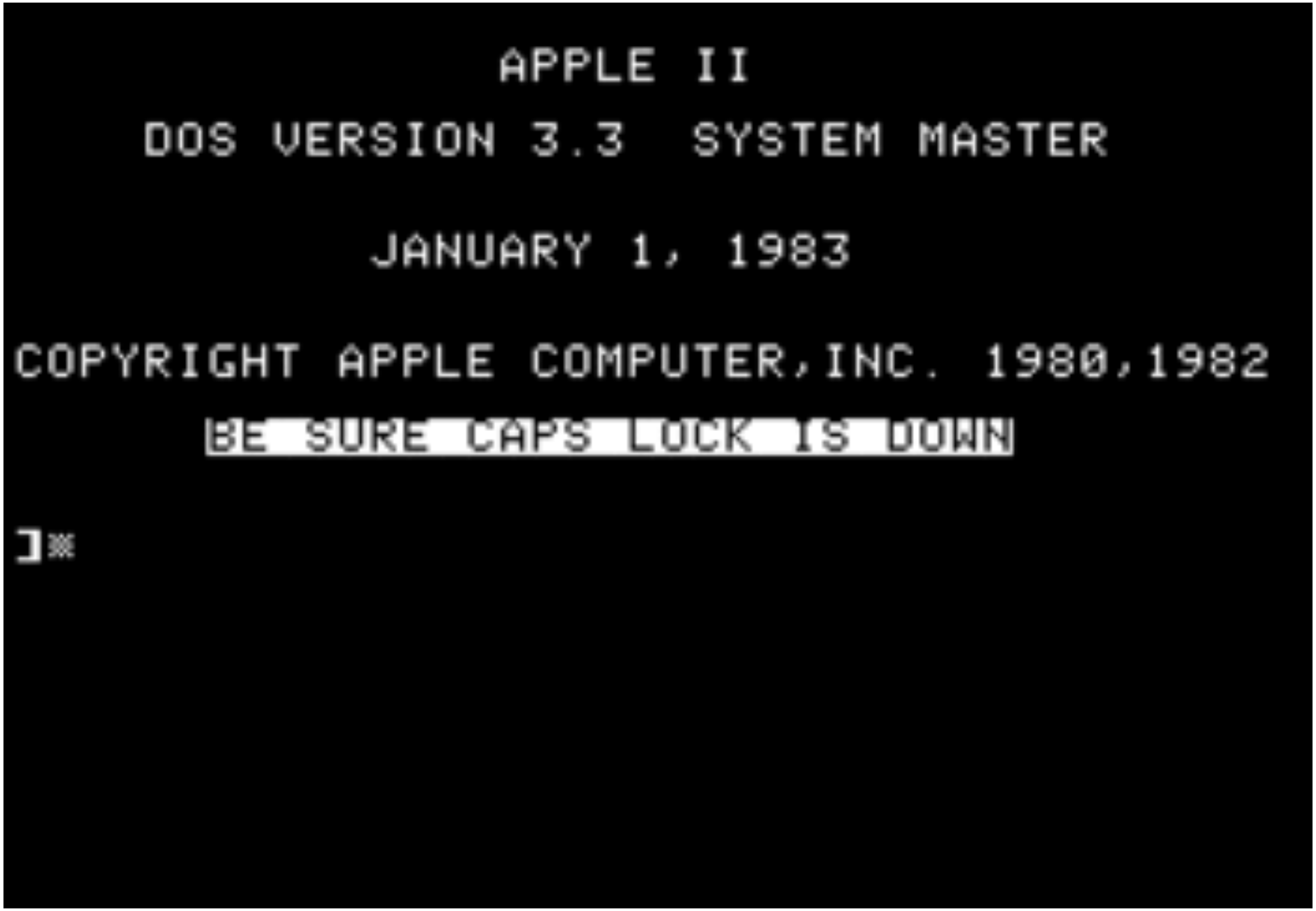
```
C:\>dir

 Volume in drive C is MS-DOS_6
 Volume Serial Number is 40B4-7F23
 Directory of C:\

DOS            <DIR>          12.05.20   15:57
COMMAND  COM           54 645 94.05.31    6:22
WINA20   386            9 349 94.05.31    6:22
CONFIG   SYS              144 12.05.20   15:57
AUTOEXEC BAT              188 12.05.20   15:57
        5 file(s)          64 326 bytes
                       24 760 320 bytes free

C:\>_
```

DOS

- Adopted by other personal computers
  - MS-DOS
  - Apple DOS
  - Included in Windows OS series

- Released as freeware by Microsoft
  - https://github.com/microsoft/ms-dos
  - Over 2000 games are also available in the Internet Archive

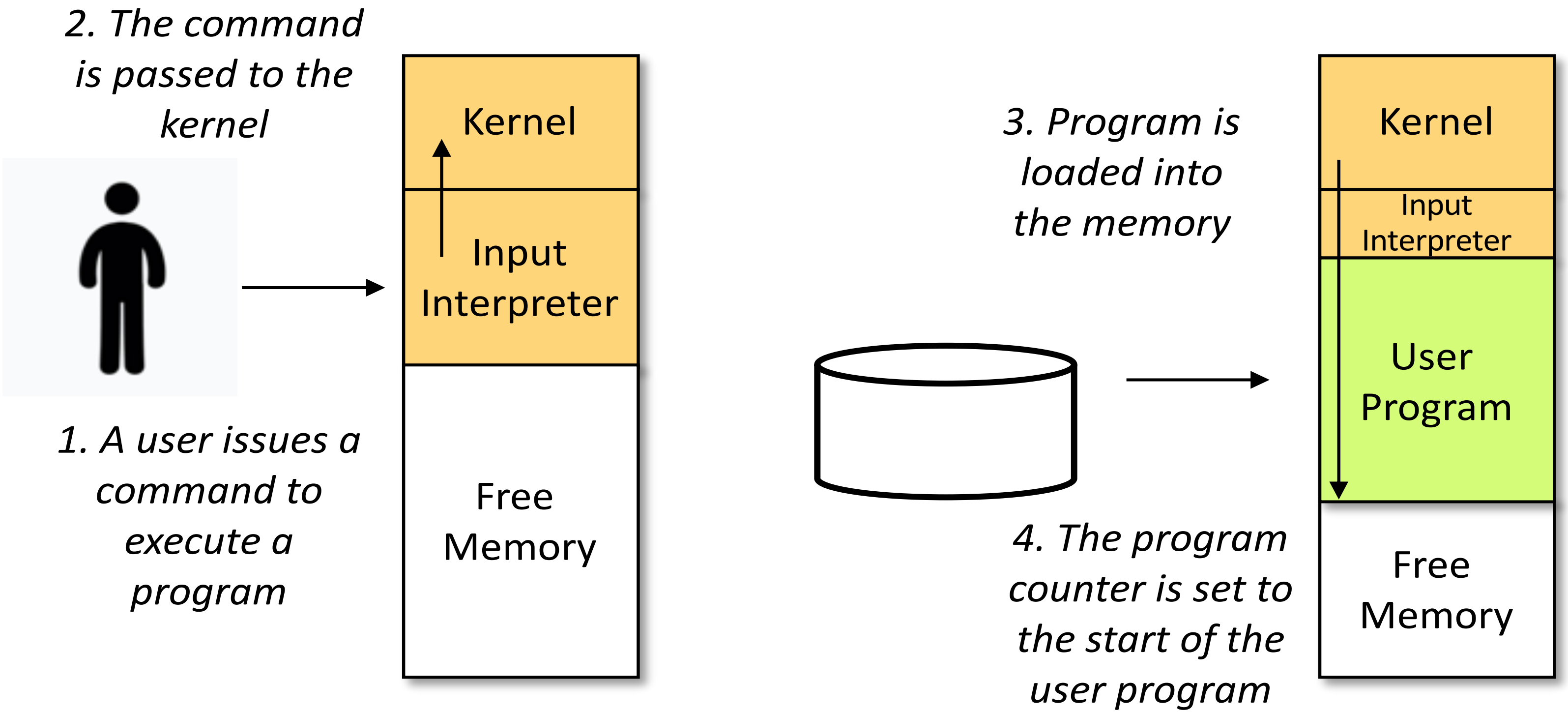
Apple II DOS



```
              APPLE II
    DOS VERSION 3.3  SYSTEM MASTER

           JANUARY 1, 1983

COPYRIGHT APPLE COMPUTER,INC. 1980,1982
      BE SURE CAPS LOCK IS DOWN

]▒
```

Execution Model of MSDOS

- Early versions did not support multi-tasking
  - One process is loaded into the memory and executed at one time
  - The OS is also loaded in the memory

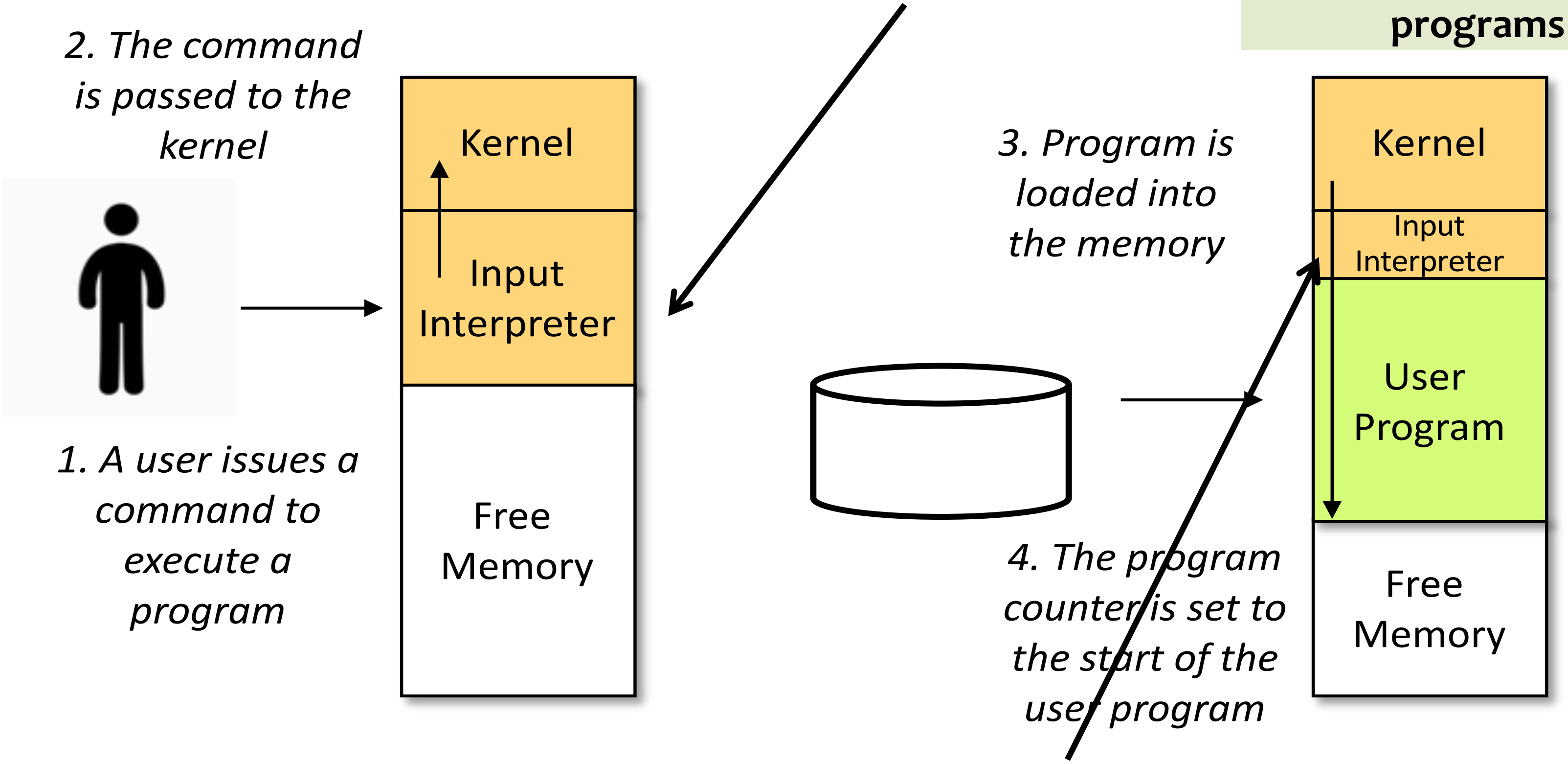

2. The command is passed to the kernel

1. A user issues a command to execute a program

3. Program is loaded into the memory

4. The program counter is set to the start of the user program

Execution Model of MSDOS



2. The command is passed to the kernel

1. A user issues a command to execute a program

A part of the Input Interpreter is overwritten

The Input Interpreter is not needed while executing user programs

3. Program is loaded into the memory

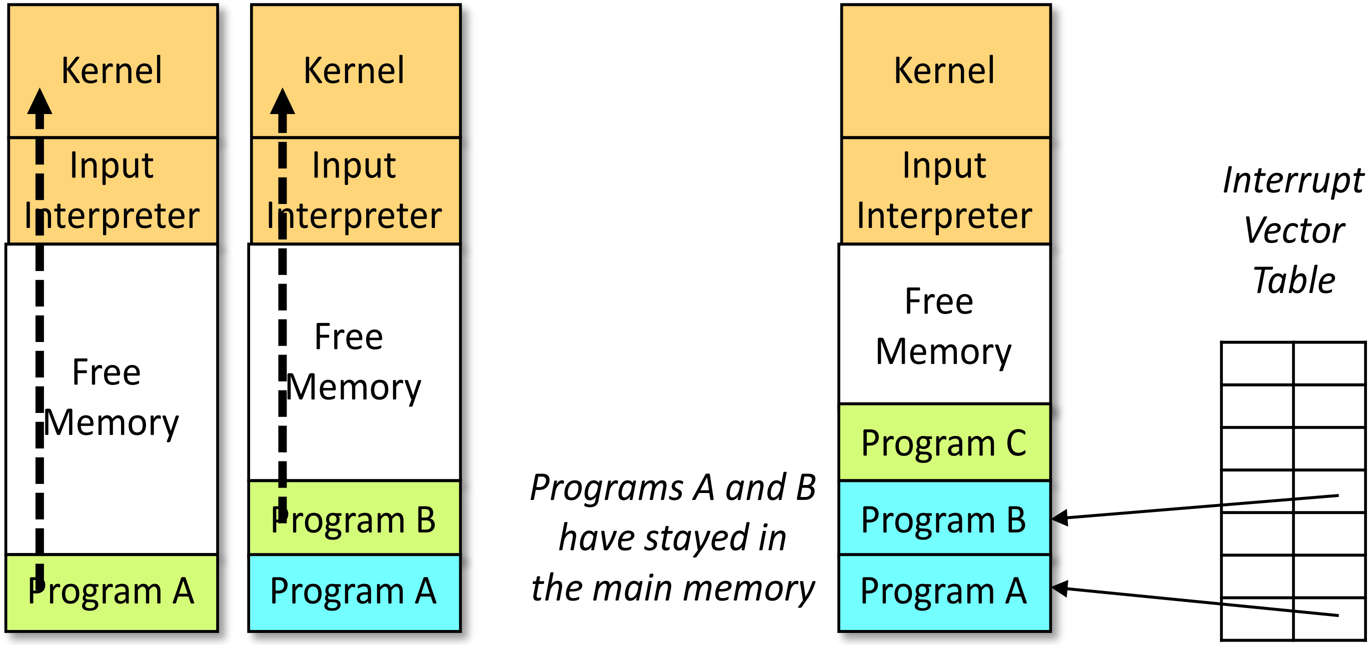
4. The program counter is set to the start of the user program

This remaining part is kept for handling the end of user programs

Terminate and Stay Resident (TSR)

- Multi-tasking is simulated with the Terminate and Stay Resident (TSR) mechanism
  - It is a system call that can be called by user programs
  - Modify the interrupt vector table so that one of the vector points to the user program

Terminate and Stay Resident (TSR)



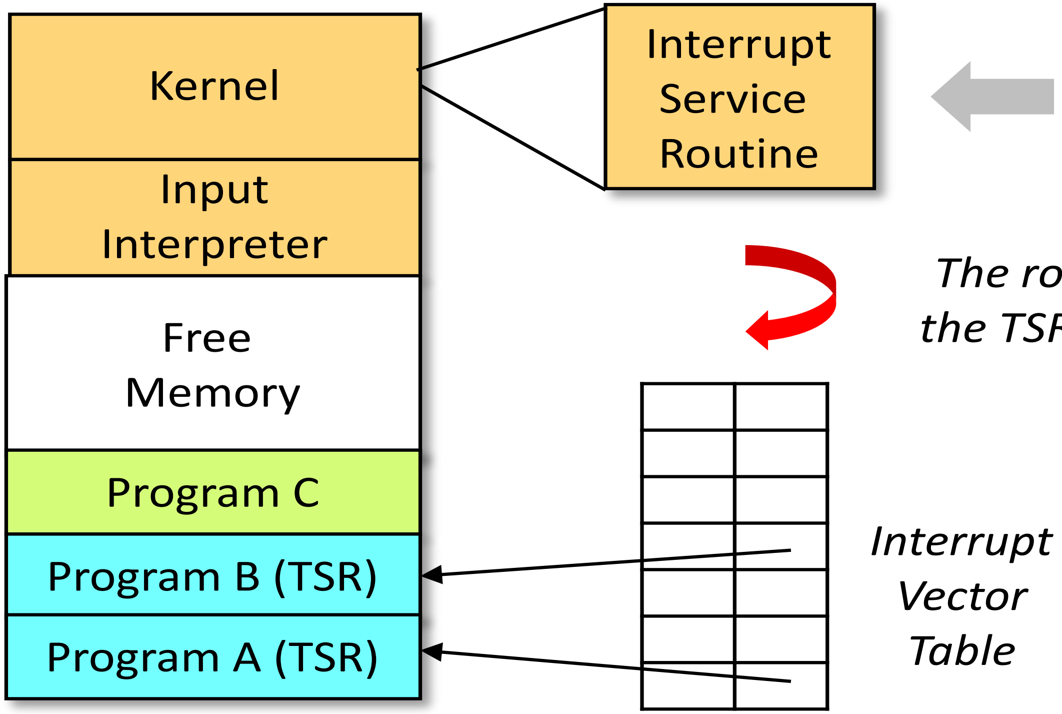
Programs A and B makes Terminate and Stay Resident (TSR) call

Programs A and B have stayed in the main memory

Terminate and Stay Resident (TSR)

- The system clock is involved
  - An interrupt is sent by system clock regularly (several times a second or more)
  - Every user program registered in the interrupt vector table is executed

Terminate and Stay Resident (TSR)



A regular system interrupt occurs and executes this interrupt handler

The routine executes the TSR programs one by one

Interrupt Vector Table