澳 門 理 工 學 院
Instituto Politécnico de Macau
Macao Polytechnic Institute

School of Applied Science (BSc. in Computing)

Notes #1: Computer System Review

COMP 213 (211/212)
Operating Systems

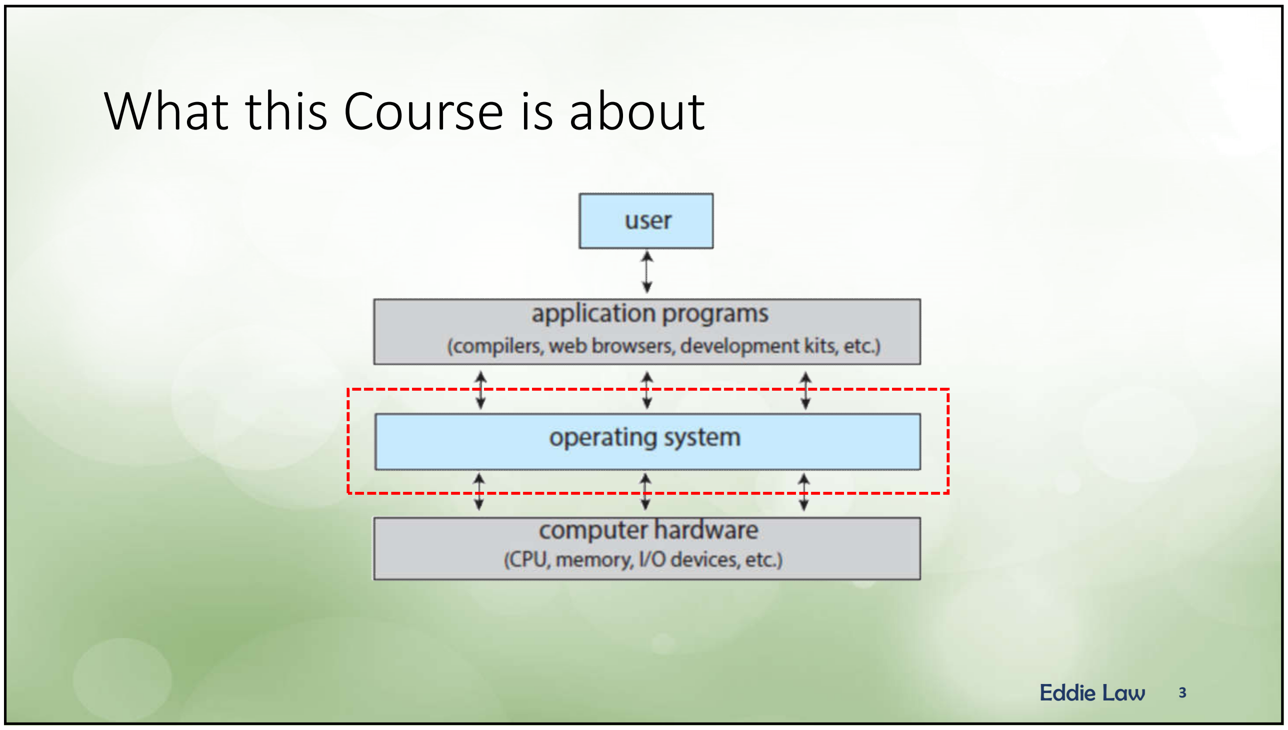
2019-2020 1st Semester

1

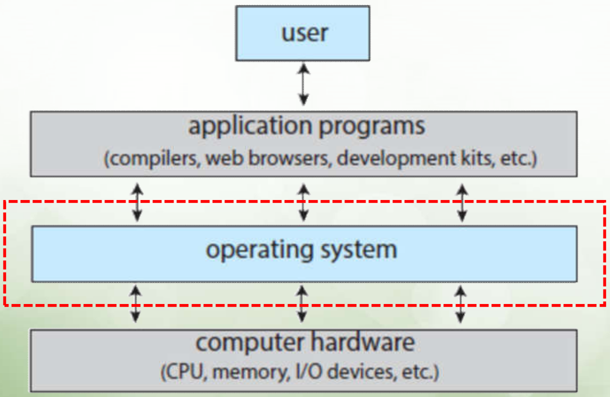
Topics to Cover

- History and development of computing systems
- Evolution of operating systems
- Textbook
  - Stallings's : Chapter 1.1-1.7
- References
  - Tanenbaum's : Chapter 1
  - Silberschatz's : Chapter 1

Eddie Law 2

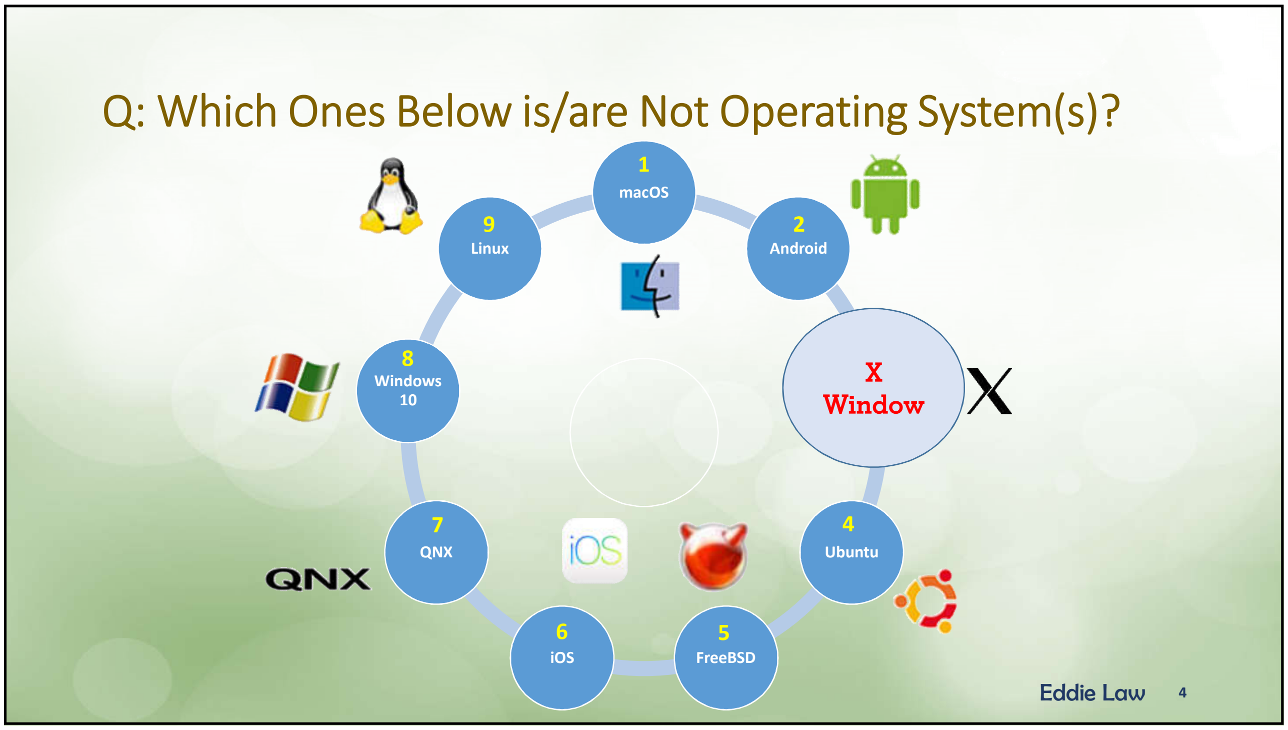


What this Course is about

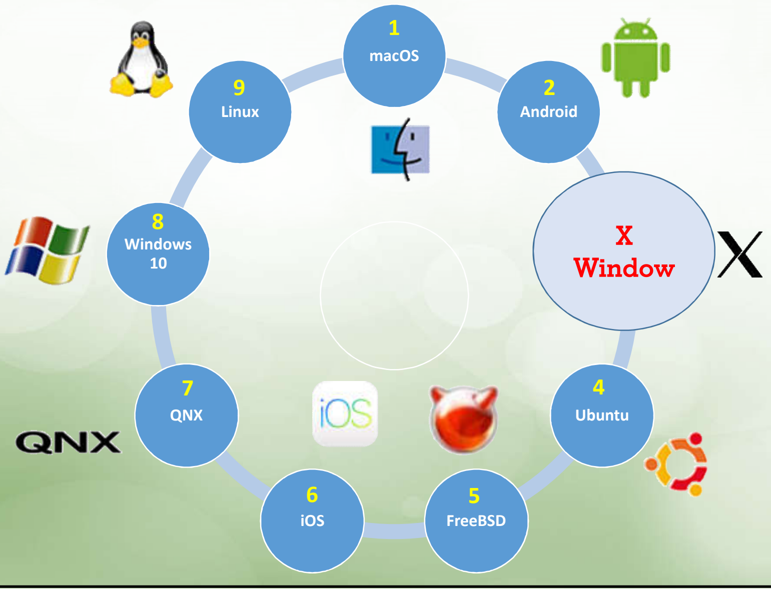


user
application programs
(compilers, web browsers, development kits, etc.)
operating system
computer hardware
(CPU, memory, I/O devices, etc.)

Eddie Law 3



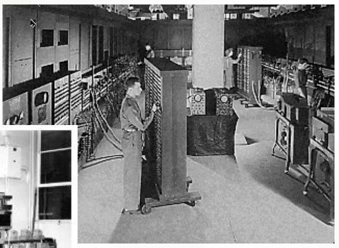
Q: Which Ones Below is/are Not Operating System(s)?



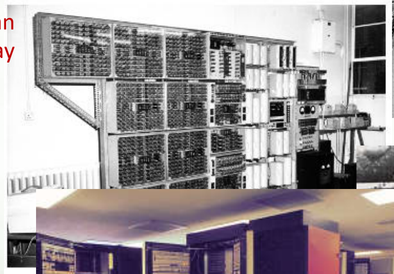
Eddie Law 4

Short History of Computers

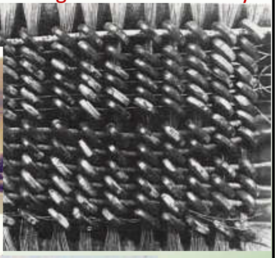
ENIAC: weighted 30 tons, consumed 200 kilowatts



Harwell: can still run today



Magnetic core memory



Multics System

- Dawn of Time (1945 – 55): ENIAC (Electronic Numerical Integrator And Computer)
  - Harwell Computer at National Museum of Computing (TNMOC) at Bletchley Park (1949)
- Core Memories (1950s - 60s)
- Multics System: MIT & Bell (1975)
- Disk drives (early 1970s)

1.7 Mbit/sq. in
140 MBytes

7.7 Mbit/sq. in
2,300 MBytes

5

Products and Solutions

- Nowadays, the world is a large parallel system
  - Microprocessors in everything
  - Vast infrastructure behind them

Internet Connectivity

Scalable, reliable, Secure services

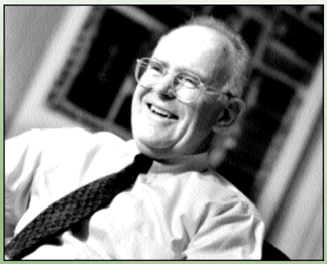
Databases
Information Collection
Remote Storage
Online Games
Commerce
…

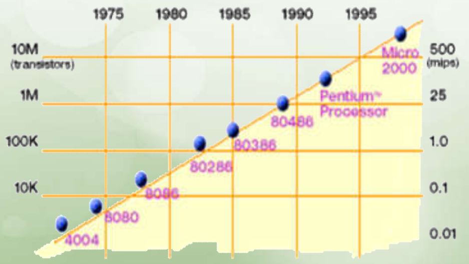
MEMS sensor

Eddie Law 6

Moore's Law

- Gordon Moore (co-founder of Intel) predicted in 1965 that the ***transistor density*** of semiconductor chips would double roughly every 18 months
- 2 × transistors/chip every 1.5 years - called "Moore's Law"
- i.e., microprocessors have become smaller, denser, and more powerful





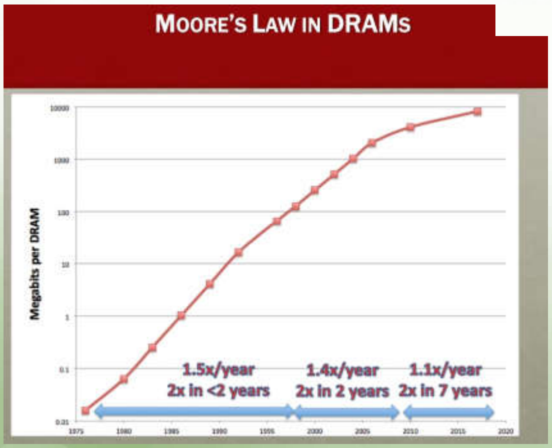
Eddie Law 7

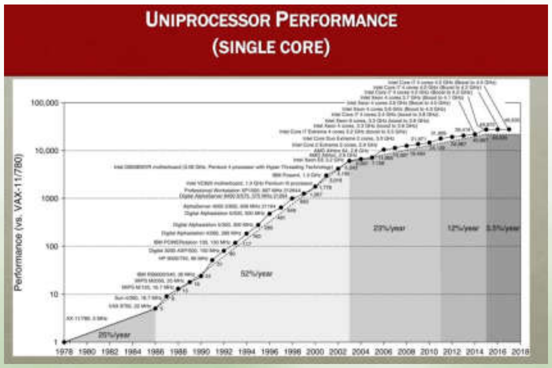
Joy's Law (1984) - peak computer speed doubled every year

Death of the Moore's Law

Then how to further improve the system performance in future?

Parallelism, e.g., multi-core, discuss later

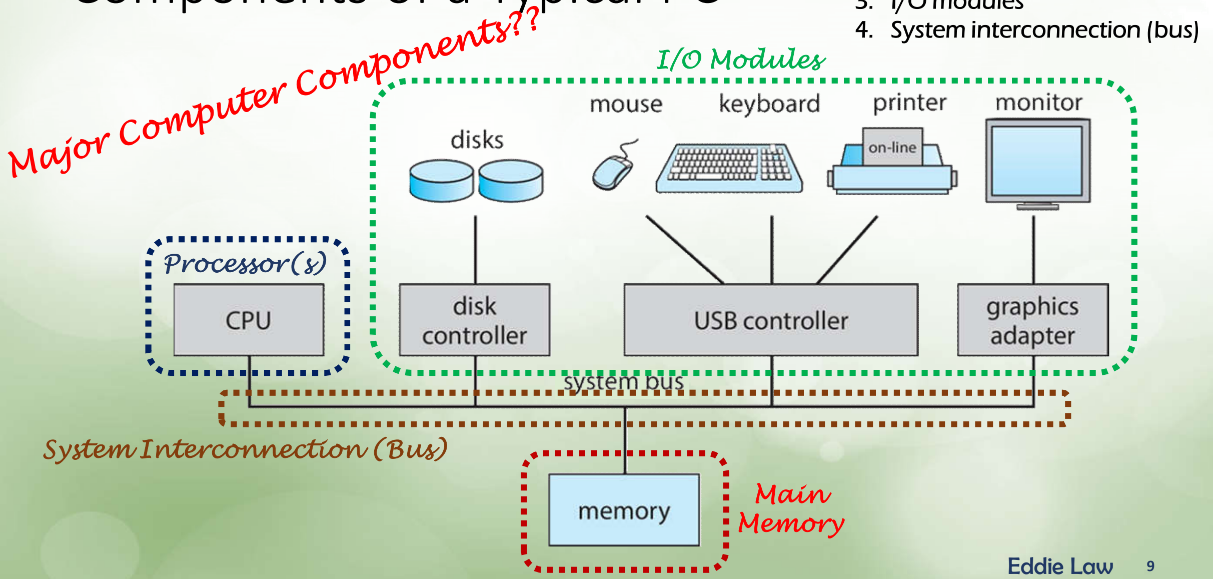




Eddie Law 8

Components of a Typical PC

Typical components:
1. Processor(s)
2. Main memory
3. I/O modules
4. System interconnection (bus)



Eddie Law 9

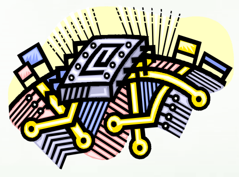
Roles of Processor(s)

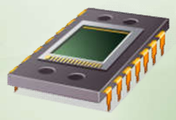
Controls the operation of a computer

Performs data processing functions

The chip is called ***Central Processing Unit*** (**CPU**)

The heart and soul of a computer





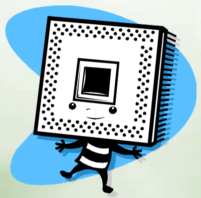
Eddie Law 10

Main Memory

- Holds data and instructions temporarily which the CPU will process and execute
- Volatile (RAM: Random Access Memory)
- Contents of the memory is lost when the computer is shut down
- Also referred to as real, or primary memory

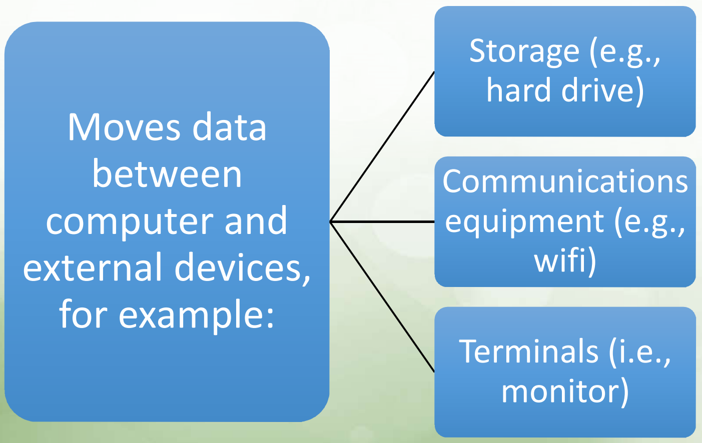
RAM in market:
- 3200 MT/s DDR4
- 6000+ MT/s DDR5 (2019)

DDR (double data rate)
MT/s (mega transfers per second)



Eddie Law 11

I/O Modules

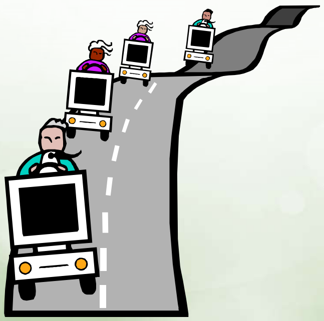

Moves data between computer and external devices, for example:
Storage (e.g., hard drive)
Communications equipment (e.g., wifi)
Terminals (i.e., monitor)

Eddie Law 12

System Bus

- Provides for communications among processors, main memory, and I/O modules

Examples of buses in PC:
1. ISA (industry standard architecture)
2. PCI (peripheral component interconnect)
3. PCIe (PCI express)



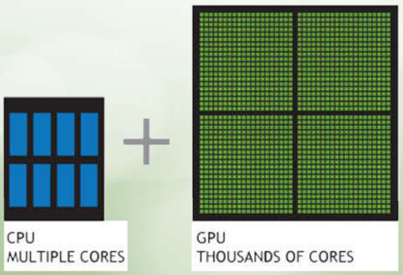
Eddie Law 13

Microprocessor(s)

AMD Ryzen 9 3950X:
16 cores, 32 threads, 3.5-4.7GHz
L1 64KB/core, L2 512KB/core, L3 64MB
Thermal design power (TDP) 105W

- Enabled desktop, handheld computing, small board computer
- Processor(s) on a single chip
- Fastest general purpose processor
- Multiprocessors
  - SMP (Symmetric), AMP (Asymmetric)
- Multi-core: a chip (socket) contains multiple processors (cores)
  - “ManyCore” for GPU refers to many processors/chip, e.g., 128 cores
  - Parallelism must be exploited - OS facilitates it

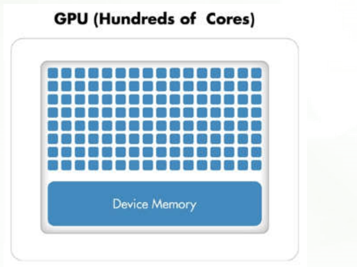
Examples
CPU: Intel, AMD, ARM, Atmel, etc.
GPU: ATI, NVidia

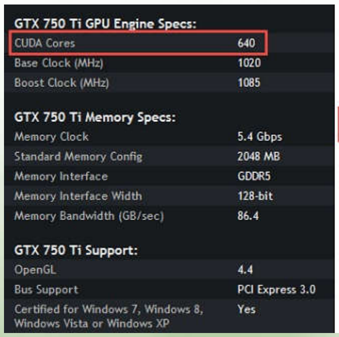


Eddie Law 14

Graphical Processing Units (GPUs)

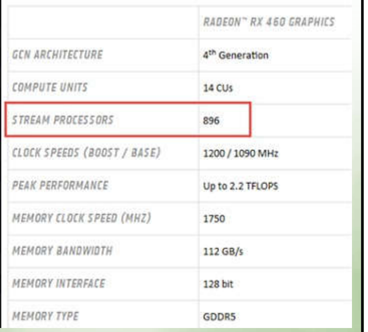
- Arrays of **small computing cores**
- Provide efficient computation on arrays of data
- Good for general numerical processing, such as
  - Physics simulations for games
  - Computations on large spreadsheets





GTX 750 Ti GPU Engine Specs:	
CUDA Cores	640
Base Clock (MHz)	1020
Boost Clock (MHz)	1085
GTX 750 Ti Memory Specs:	
Memory Clock	5.4 Gbps
Standard Memory Config	2048 MB
Memory Interface	GDDR5
Memory Interface Width	128-bit
Memory Bandwidth (GB/sec)	86.4
GTX 750 Ti Support:	
OpenGL	4.4
Bus Support	PCI Express 3.0
Certified for Windows 7, Windows 8, Windows Vista or Windows XP	Yes

CUDA (Compute United Device Architecture), the Nvidia's proprietory compute platform on GPU



	RADEON™ RX 460 GRAPHICS
GCN ARCHITECTURE	4th Generation
COMPUTE UNITS	14 CUs
STREAM PROCESSORS	896
CLOCK SPEEDS (BOOST / BASE)	1200 / 1090 MHz
PEAK PERFORMANCE	Up to 2.2 TFLOPS
MEMORY CLOCK SPEED (MHZ)	1750
MEMORY BANDWIDTH	112 GB/s
MEMORY INTERFACE	128 bit
MEMORY TYPE	GDDR5

OpenCL (Open Computing Language) On ATI's Radeon GPU

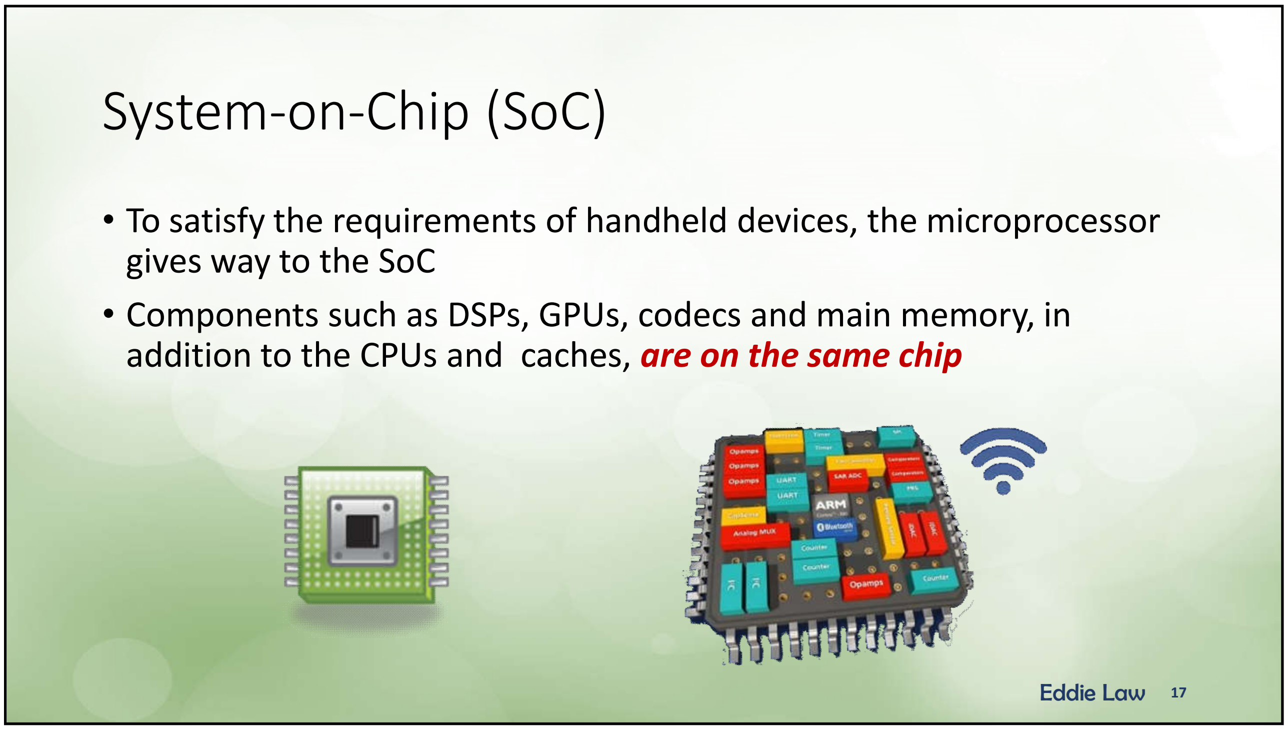
Eddie Law 15

Digital Signal Processors (DSPs)

- Used to be embedded in devices like modems
- Effective for ***streaming signals*** such as audio or video
- **Encoding/decoding** speech and video (codecs)
- Support for **encryption** and **security**

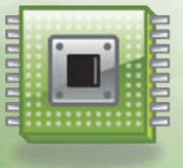
1.25GHz, 8-core, TMS320C6678 DSP

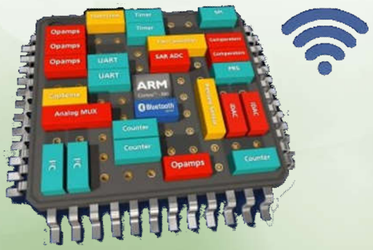
Eddie Law 16



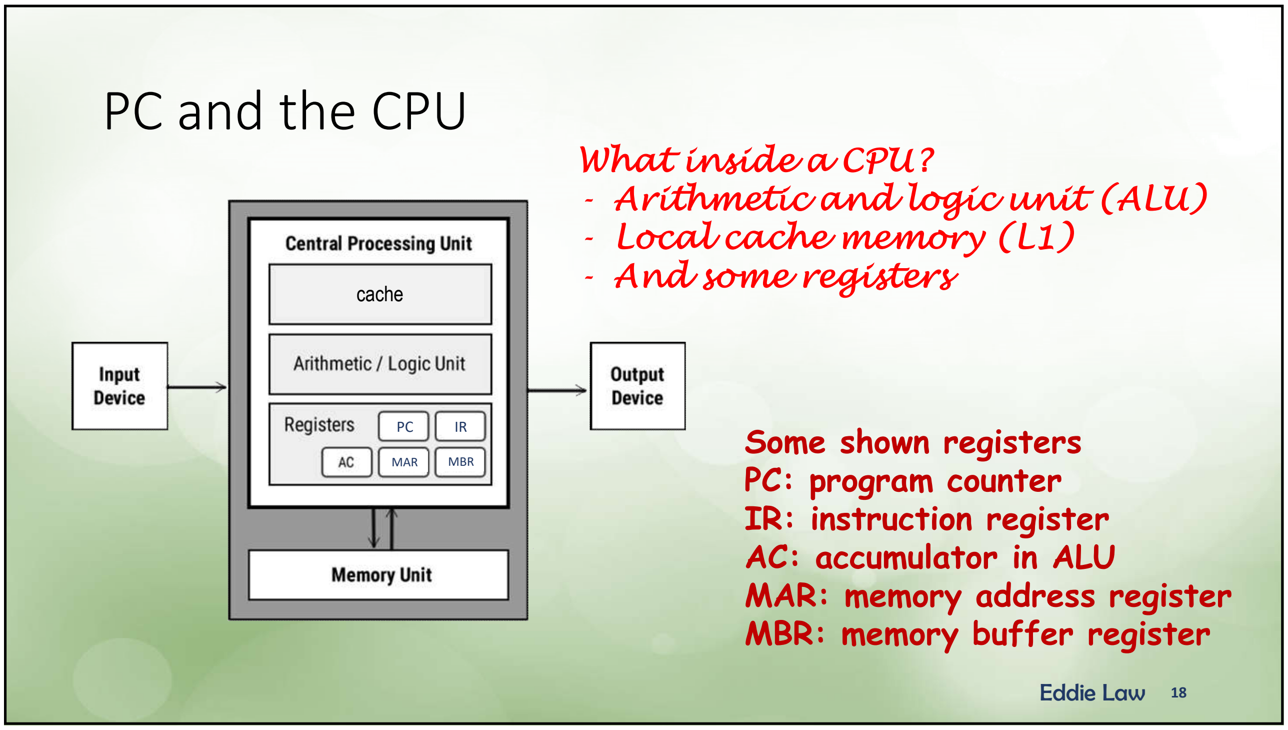
System-on-Chip (SoC)

- To satisfy the requirements of handheld devices, the microprocessor gives way to the SoC
- Components such as DSPs, GPUs, codecs and main memory, in addition to the CPUs and caches, ***are on the same chip***



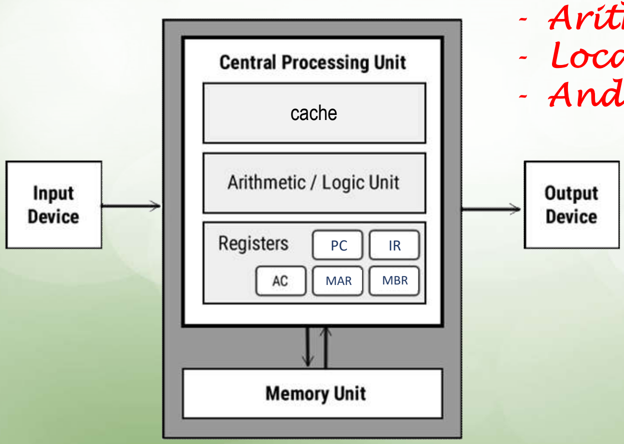


Eddie Law 17



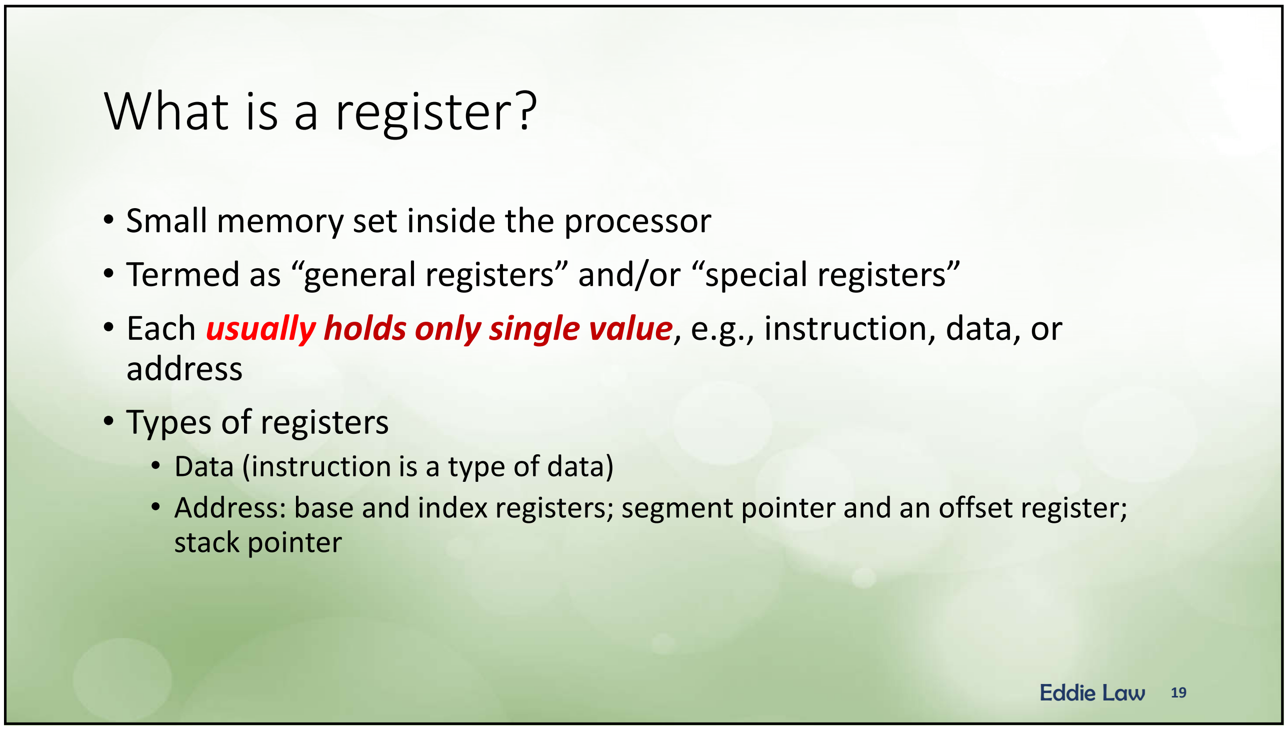
PC and the CPU

What inside a CPU?
- *Arithmetic and logic unit (ALU)*
- *Local cache memory (L1)*
- *And some registers*



Some shown registers
PC: program counter
IR: instruction register
AC: accumulator in ALU
MAR: memory address register
MBR: memory buffer register

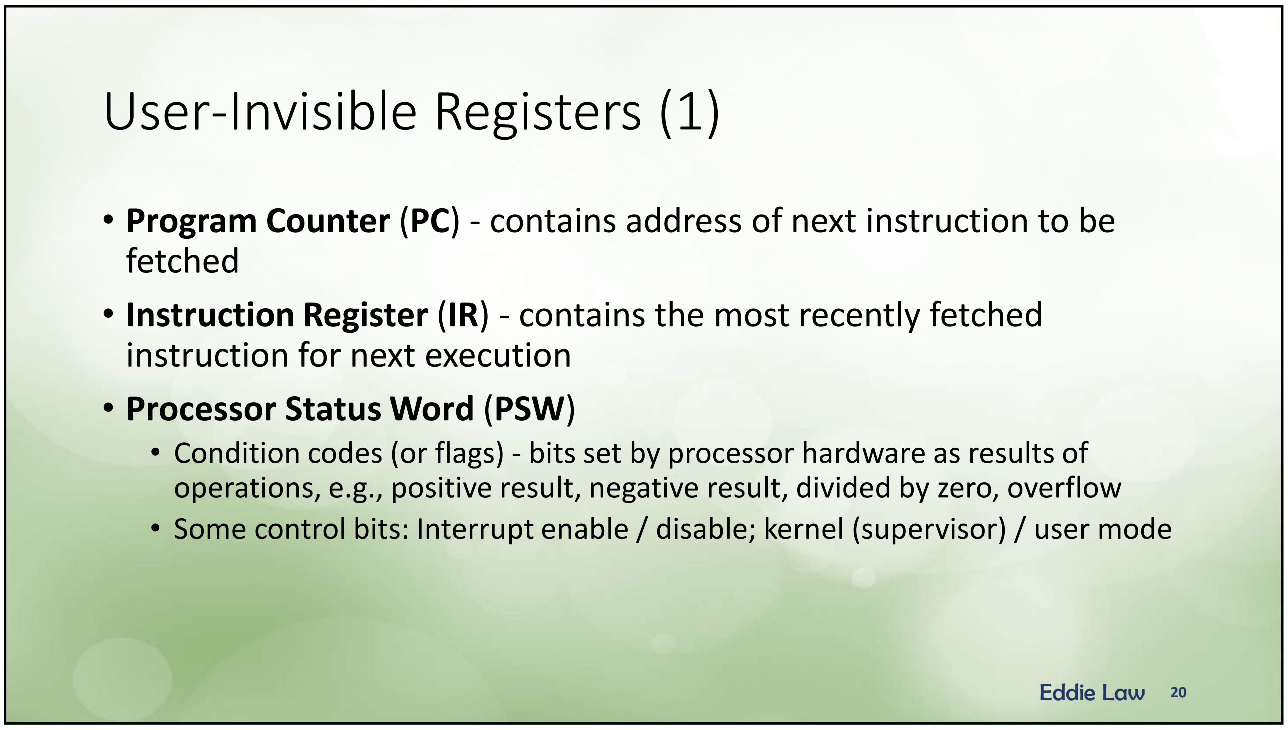
Eddie Law 18



What is a register?

- Small memory set inside the processor
- Termed as “general registers” and/or “special registers”
- Each ***usually holds only single value***, e.g., instruction, data, or address
- Types of registers
  - Data (instruction is a type of data)
  - Address: base and index registers; segment pointer and an offset register; stack pointer

Eddie Law 19



User-Invisible Registers (1)

- **Program Counter** (**PC**) - contains address of next instruction to be fetched
- **Instruction Register** (**IR**) - contains the most recently fetched instruction for next execution
- **Processor Status Word** (**PSW**)
  - Condition codes (or flags) - bits set by processor hardware as results of operations, e.g., positive result, negative result, divided by zero, overflow
  - Some control bits: Interrupt enable / disable; kernel (supervisor) / user mode

Eddie Law 20

User-Visible Registers (2)

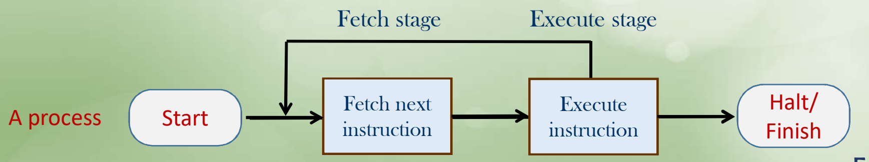
- Enable programmer to minimize main-memory references by optimizing register use
- Available to all applications and system programmes
- **Data registers**
  - For calculation
  - Minimize references to main memory
- **Address registers**
  - Contain main memory address of data and instructions
  - May contain a portion of an address used to calculate the complete address
- Settings of registers would be vendor-specific and chip-specific

Control and Status Registers

- Used by processor to control the operations of a processor
- Used by privileged operating-system routines to control the execution of programmes
- E.g., the PSW

An Instruction Cycle (1)

- In its simplest form, instruction processing consists of ***two*** steps:
  1. The processor **reads** (*fetches*) instructions from memory one at a time
  2. It then **executes** the instruction
- Program execution consists of repeating the process of instruction fetch and instruction execution
- Instruction execution may involve several operations and depends on the nature of the instruction



Eddie Law 23

Registers and Instruction Cycles

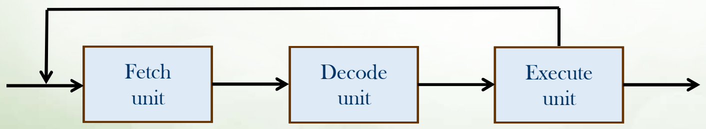
- In an instruction cycle
  - PC holds the address of next instruction to be fetched
  - Fetched instruction is placed in IR
  - Then PC is incremented after the fetch
  - Then execution of instruction

Eddie Law

An Instruction Cycle (2)

(a) Some books called it a three-stage pipeline

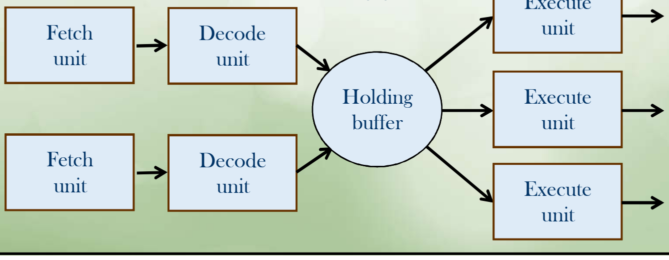
- i.e., “fetch-and-decode” for the “fetch” stage



(a)

(b) A superscalar CPU

(b)



Short Notes on Superscalar and Multicore (1)

- Super-scalar processors
  - Multiple instructions can be dispatched during a single clock cycle
  - Keeping track of multiple instructions in-flight, but all instructions are from a single program; that is, it is still just ***one process***
  - Hence, there is only one instruction counter (different from multi-core)
  - For only "one instruction counter," and it is technically true that the code will run and experience no disparity unless using some branch prediction schemes (speculative execution: simultaneously execute both branches and throw away the "wrong" prediction's result)

Short Notes on Superscalar and Multicore (2)

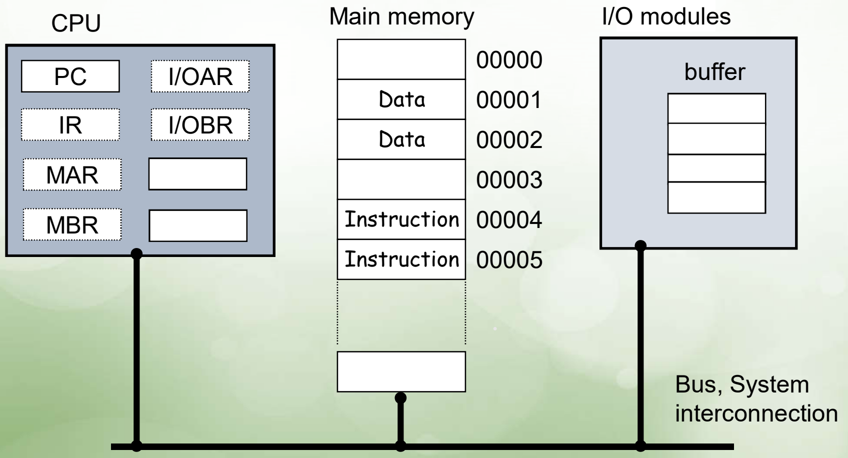
- Multi-core
  - Multiple instruction streams can execute simultaneously
  - The important part is that each core (executing with its own instruction counter) can also be super-scalar in order to execute each single process more quickly!

A Typical Set of Registers

- PC - Program Counter
- IR - Instruction Register
- MAR - Memory Address Register
- MBR - Memory Buffer Register
- I/OAR - I/O Address Register
- I/OBR - I/O Buffer Register

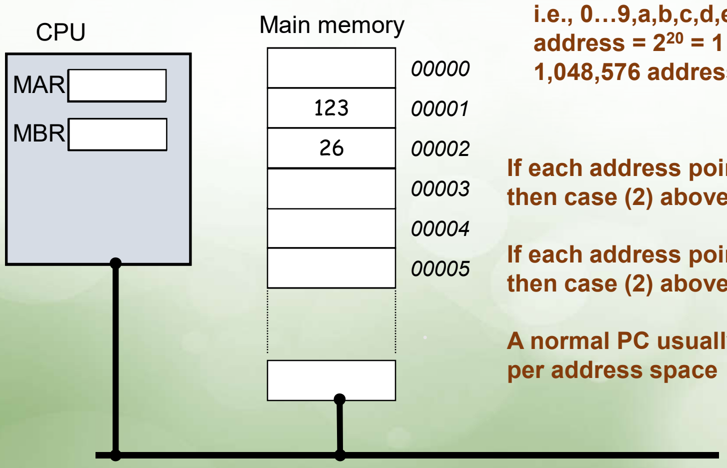
Usually, some of these registers are directly visible to system programmers, but most of them are **not**

Computer Components: Top Level View



Eddie Law 29

Reading/Writing Memory



A side notes on addressing:
1. If using decimal addressing, then 5-digit address gives 10000 address space
2. If using hexadecimal addressing, i.e., 0…9,a,b,c,d,e,f, then 5-digit address = 2^{20} = 1 M = 1,048,576 address space

If each address points to 1 byte, then case (2) above has 1 MBytes RAM

If each address points to 2 bytes, then case (2) above has 2 MBytes RAM

A normal PC usually points to only 1 byte per address space

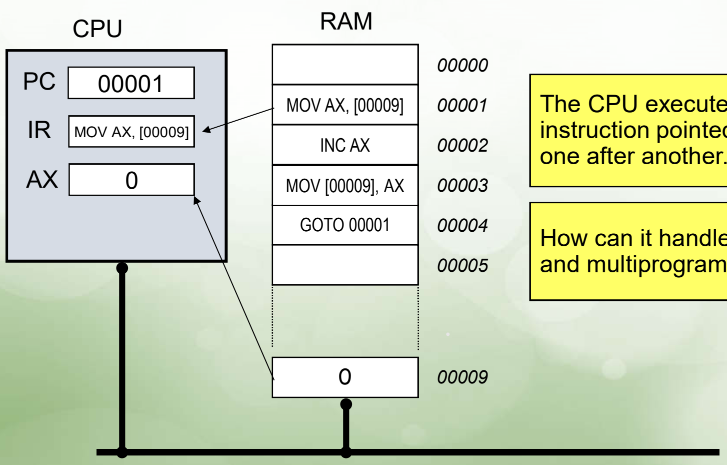
Eddie Law 30

Categories of Instructions

- **Processor-memory**: transfer data between processor and memory
- **Processor-I/O**: transfer data between processor and I/O module
- **Data processing**: perform arithmetic or logic operation on data
- **Control**: change the sequence of execution

Eddie Law

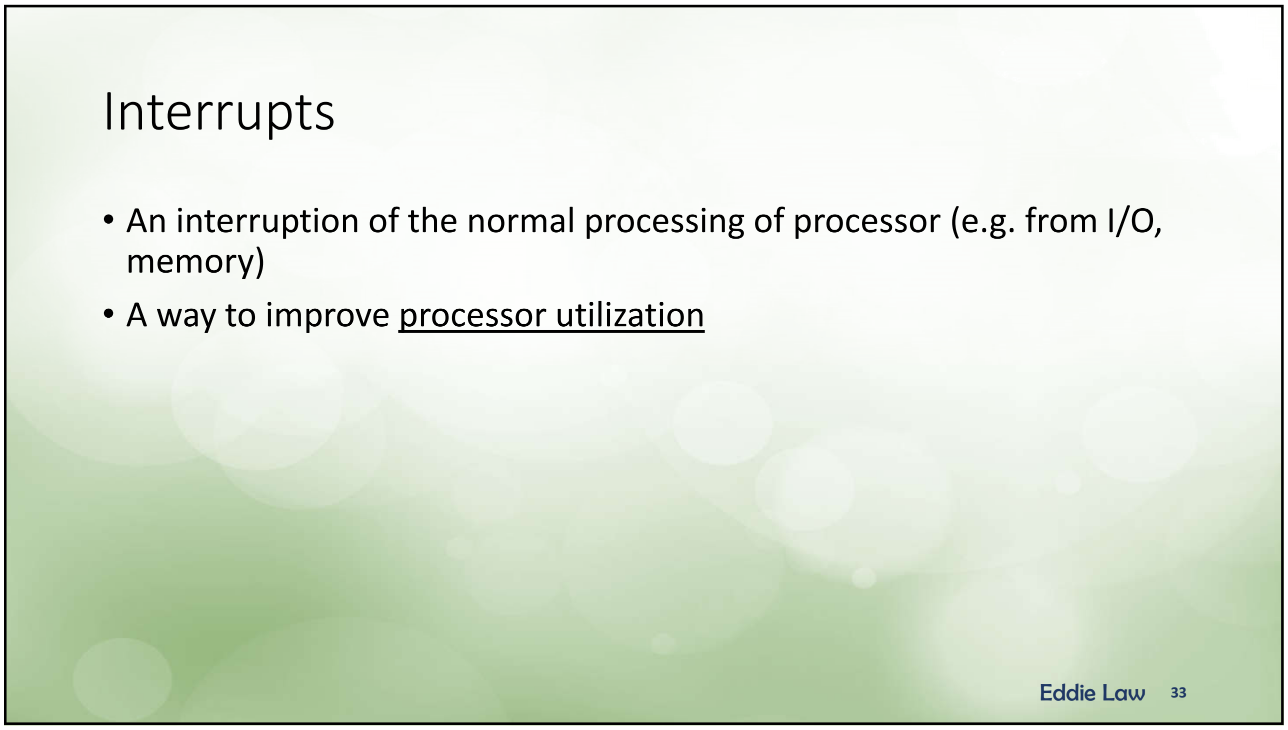
Instruction Execution



The CPU executes the instruction pointed by PC, one after another.

How can it handle I/O and multiprogramming?

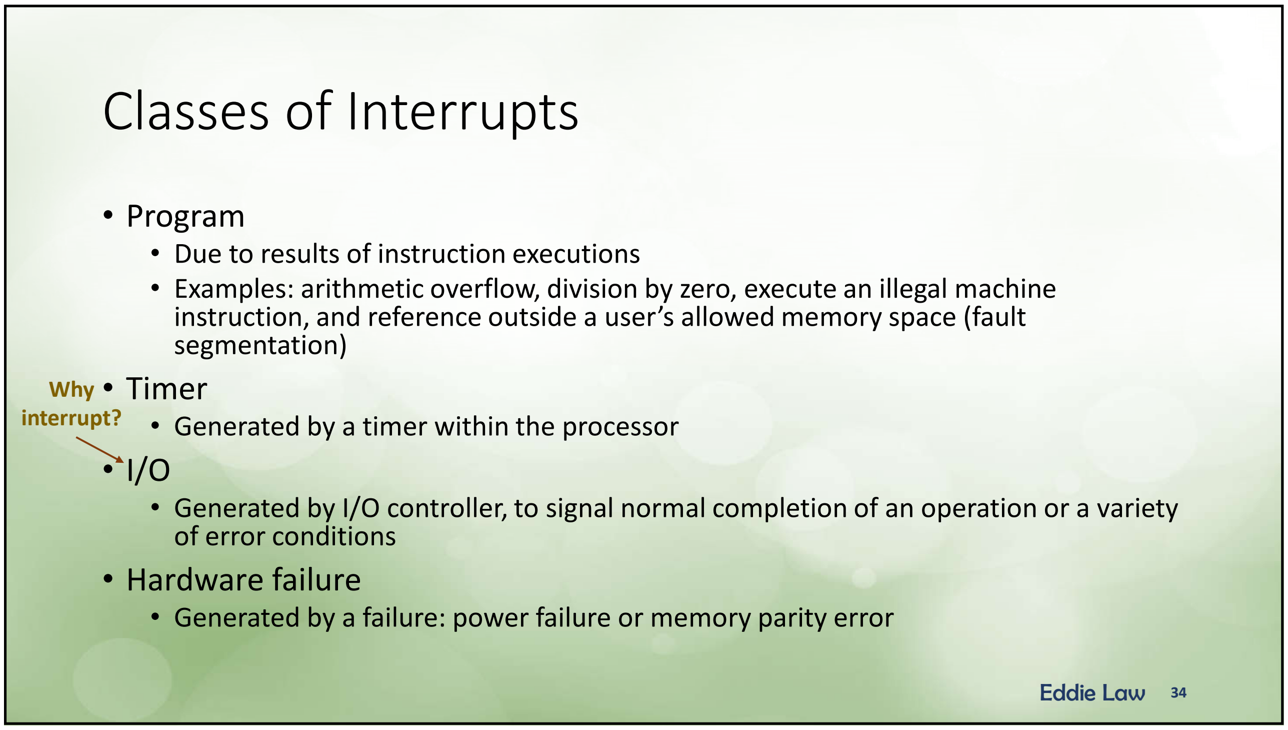
Eddie Law 32



Interrupts

- An interruption of the normal processing of processor (e.g. from I/O, memory)
- A way to improve processor utilization

Eddie Law 33



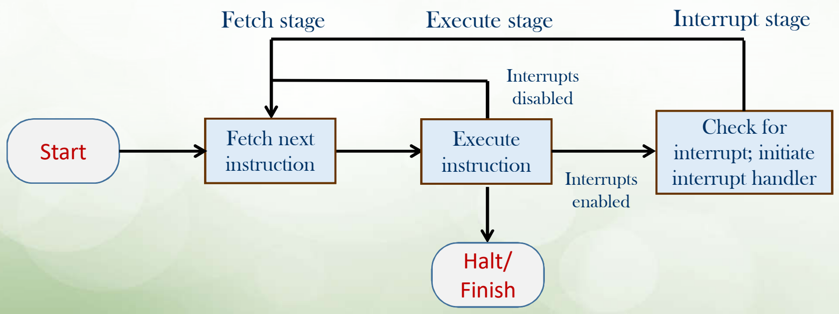
Classes of Interrupts

- Program
  - Due to results of instruction executions
  - Examples: arithmetic overflow, division by zero, execute an illegal machine instruction, and reference outside a user's allowed memory space (fault segmentation)
- Timer
  - Generated by a timer within the processor
- I/O
  - Generated by I/O controller, to signal normal completion of an operation or a variety of error conditions
- Hardware failure
  - Generated by a failure: power failure or memory parity error

Why interrupt?

Eddie Law 34

Interrupt Cycles with Interrupts



Eddie Law 35

Interrupt Stage

- Processor checks for interrupts
- If no interrupts, fetch the next instruction from the current process
- If an interrupt is pending, suspend execution of the current process, and execute the interrupt handler

Eddie Law 36

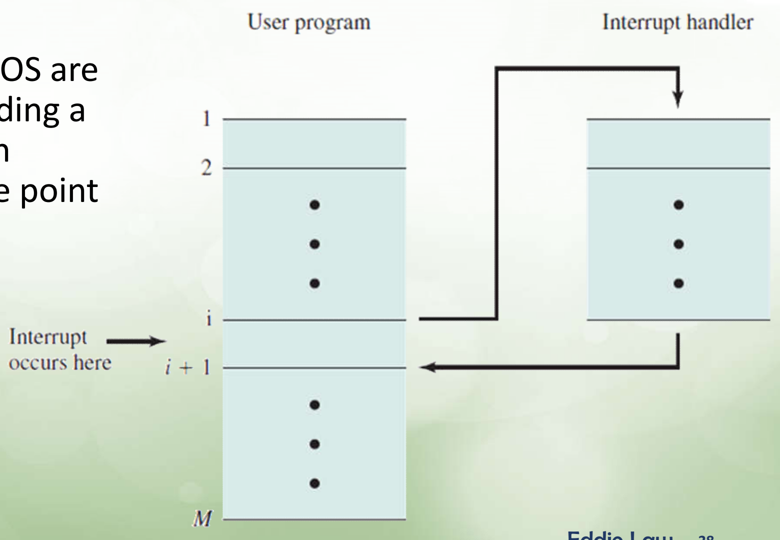
Interrupt Handler or Interrupt Service Routine

- Usually a small process which determines the nature of an interrupt and performs whatever actions are needed
- When the CPU is interrupted, control is transferred to this interrupt handler process
- Upon finishing, control is transferred back to the interrupted process
- It could generally be a part of an operating system, e.g., mouse click

Eddie Law 37

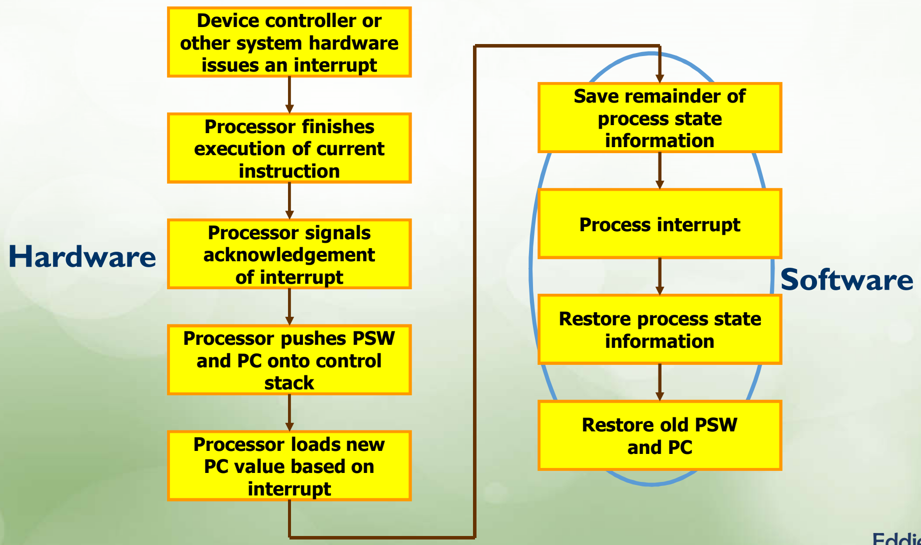
Transfer of Control via Interrupts

- The processor and the OS are responsible for suspending a user program, and then resuming it at the same point



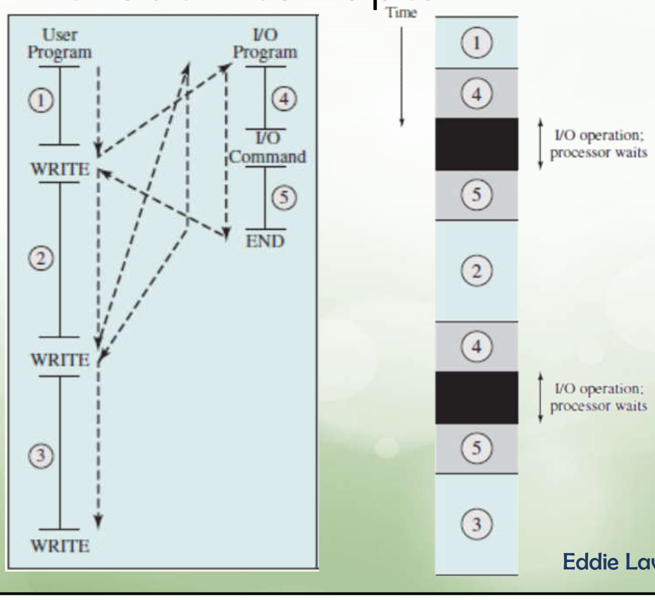
Eddie Law 38

Procedure of a Simple Interrupt

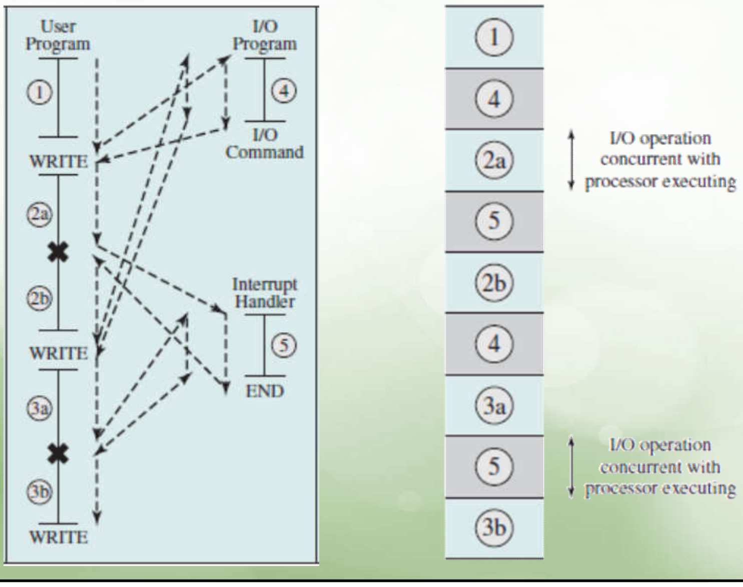

Device controller or other system hardware issues an interrupt
Processor finishes execution of current instruction
Processor signals acknowledgement of interrupt
Processor pushes PSW and PC onto control stack
Processor loads new PC value based on interrupt
Save remainder of process state information
Process interrupt
Restore process state information
Restore old PSW and PC
Hardware
Software

A Program Flow without Interrupts

- Short I/O waiting time



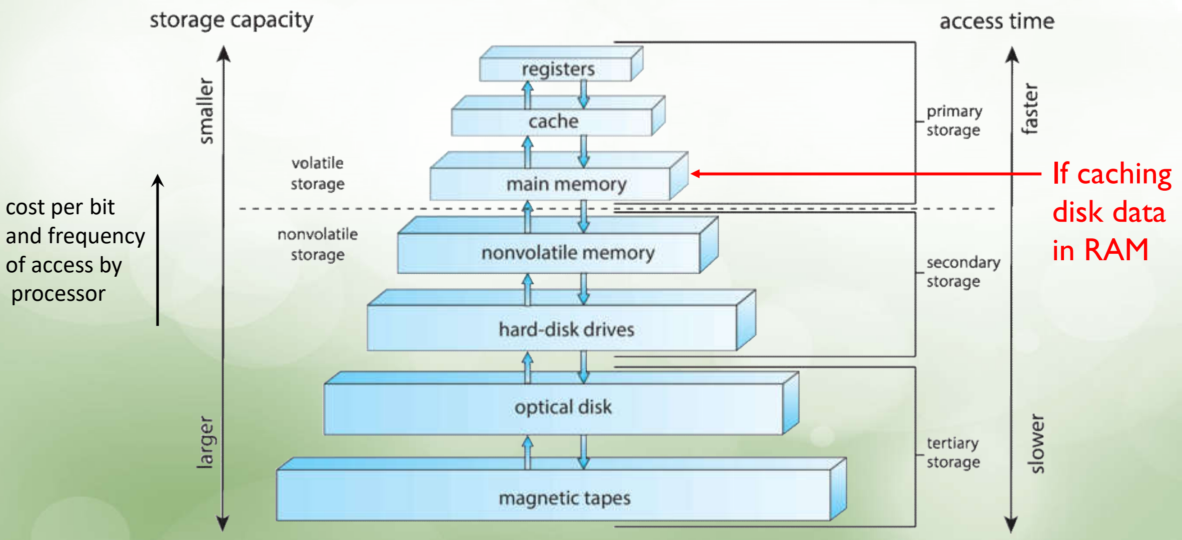
A Program Flow with Interrupts



✖ Interrupt occurs

Eddie Law 41

Memory Hierarchy



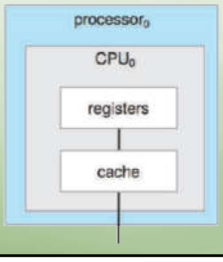
Eddie Law 42

Disk Cache

- Could be some proprietary implementations
- A portion of main memory used as a buffer to temporarily hold data for the disk
- Disk writes are clustered
- Some data written out may be referenced again
- The data are retrieved rapidly from the software cache instead of slowly from disk

Cache Memory

- Invisible to operating system
- Processor speed is much faster than memory access speed
- Shorten the data retrieval time, in general
- Virtually increase the access speed of memory on average
- Small caches have a significant impact on performance



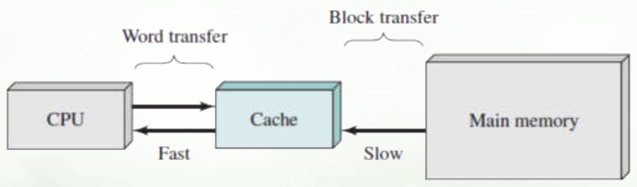
Cache Memory: How It Works

- Operation of cache: saves a portion of data from main memory
- How to do it?
- When data is requested, processor first checks cache(s)
- If not found in cache(s), the block of memory containing the needed information is moved to the cache

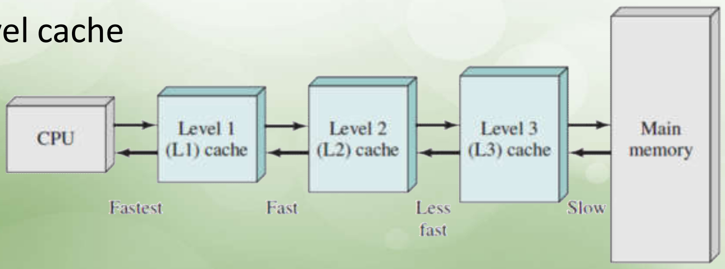
Eddie Law 45

Caches and Main Memory

- Single cache



- Three-level cache

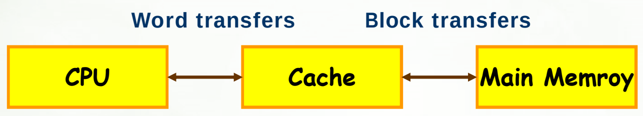


Eddie Law 46

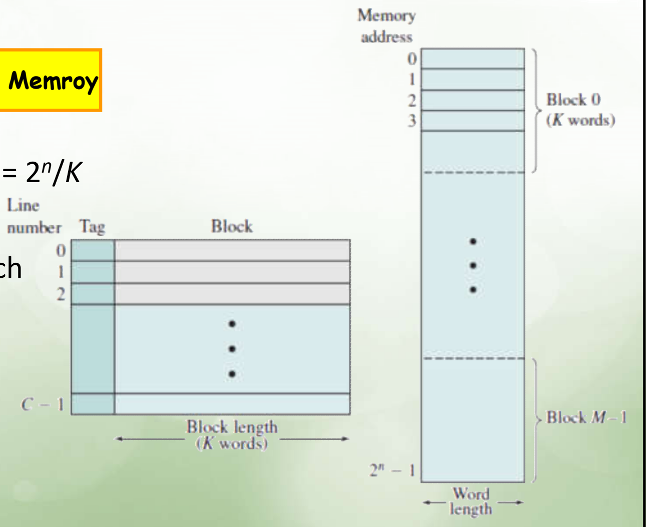
Basis of Validity

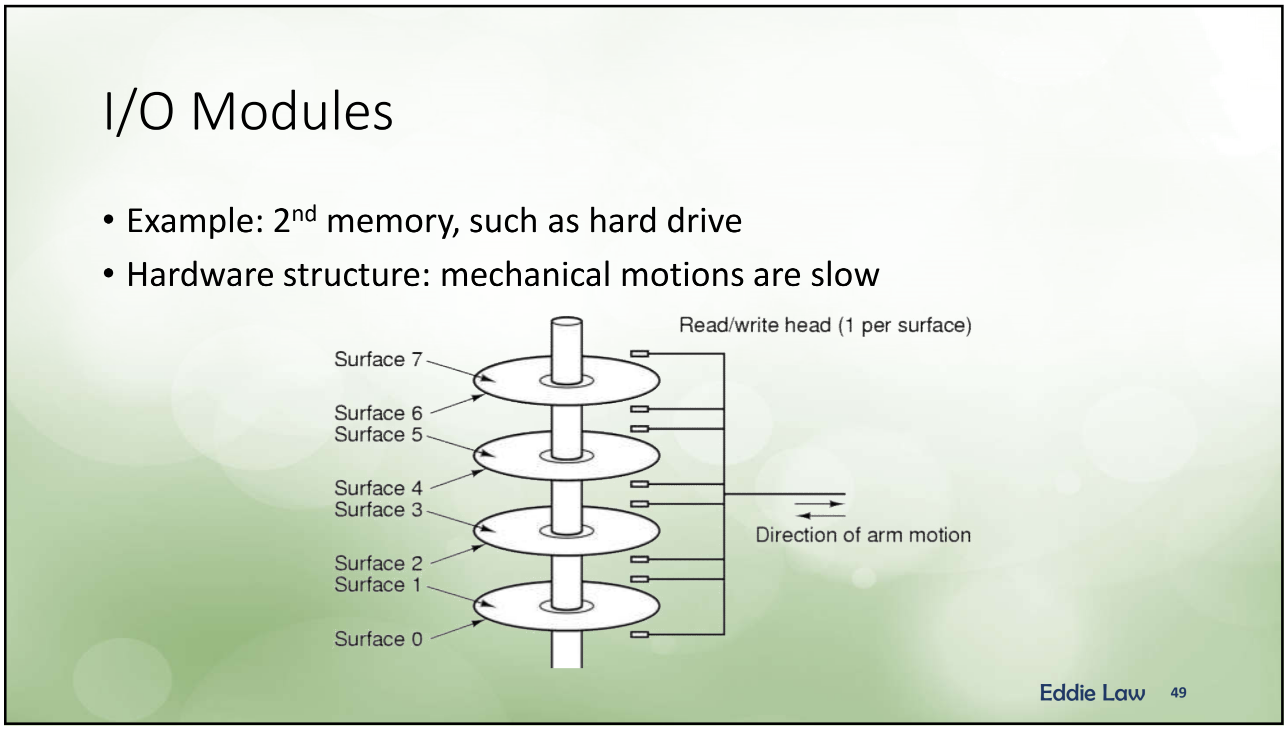
- Based on the principle known as the **locality of reference**

Cache/Main Memory Example



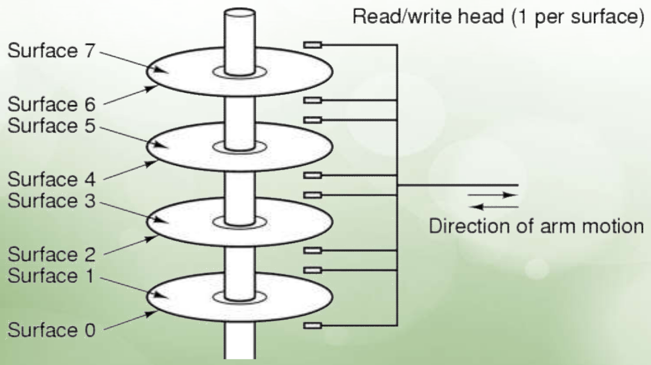
- Number of blocks in main memory = $M = 2^n/K$
- Number of slots in cache = $C << M$
- Size of a tag is small to allow quick search
  - But a tag maps to a number of blocks



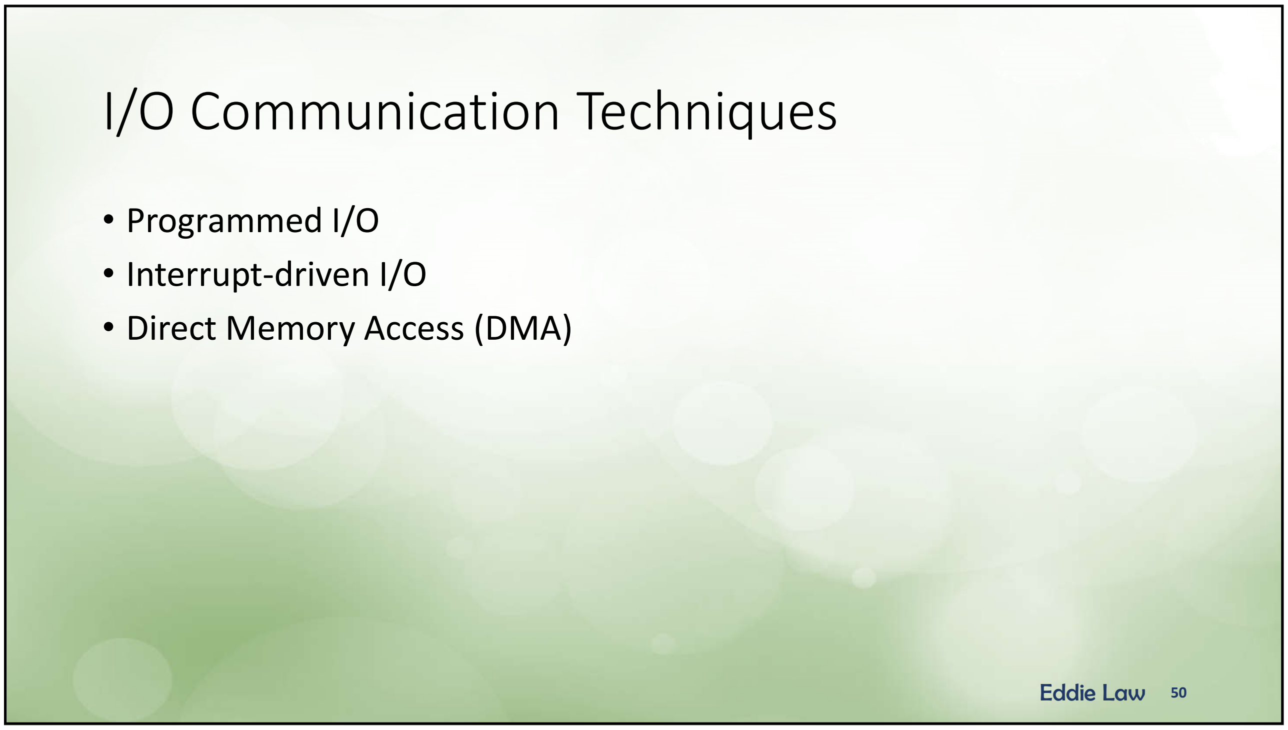


I/O Modules

- Example: 2nd memory, such as hard drive
- Hardware structure: mechanical motions are slow



Eddie Law 49

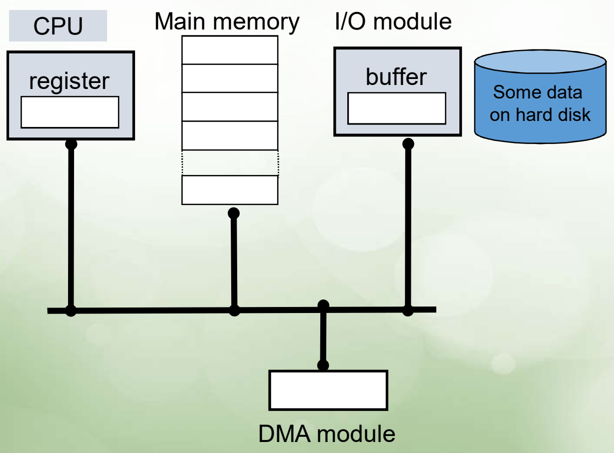


I/O Communication Techniques

- Programmed I/O
- Interrupt-driven I/O
- Direct Memory Access (DMA)

Eddie Law 50

I/O Communication Techniques



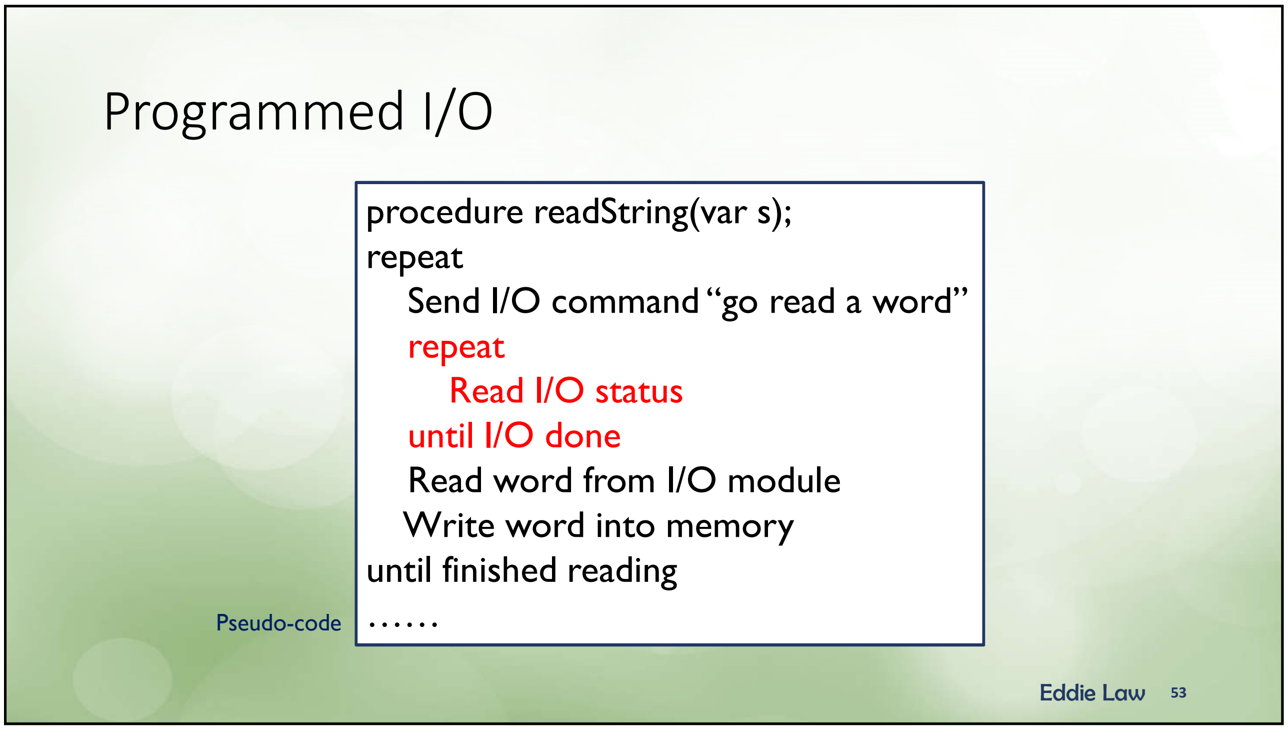
Eddie Law 51

Programmed I/O

- No interrupts occur
- Processor is kept busy checking status
- Polling

I/O access speeds are always much slower than the CPU. It is inefficient for the CPU to wait for I/O completion in a tight loop. (busy waiting). More on this in later chapters.

Eddie Law 52

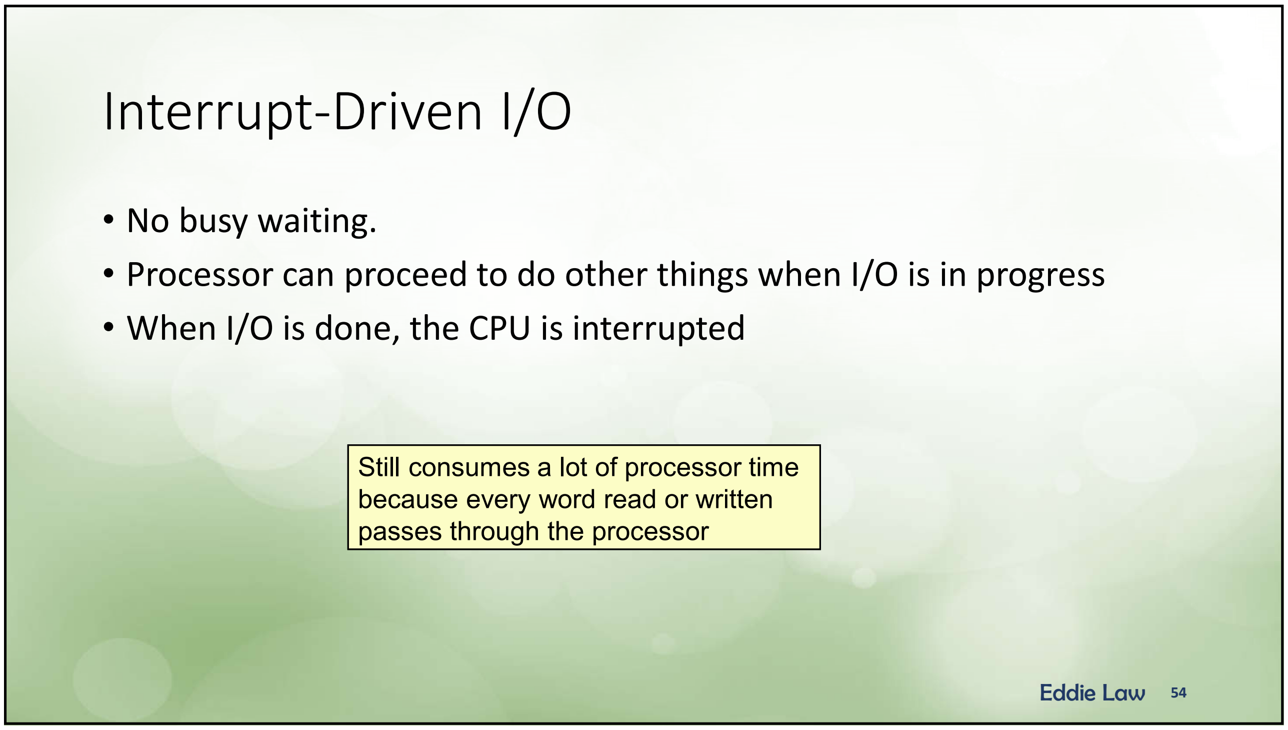


Programmed I/O

```
procedure readString(var s);
repeat
    Send I/O command "go read a word"
    repeat
        Read I/O status
    until I/O done
    Read word from I/O module
    Write word into memory
until finished reading
……
```

Pseudo-code

Eddie Law 53



Interrupt-Driven I/O

- No busy waiting.
- Processor can proceed to do other things when I/O is in progress
- When I/O is done, the CPU is interrupted

Still consumes a lot of processor time because every word read or written passes through the processor

Eddie Law 54

Interrupt-Driven I/O

```
Install interrupt handler
…
procedure readString(var s);
repeat
    Send I/O command "go read a word"
    …
    Now CPU does something else
    No need to check I/O status
    …
until finished reading
……
```

/* interrupt handler */
Read word from I/O module
Write word into memory
return

When the I/O module finished reading the word, it sends an interrupt to the CPU. The CPU runs an interrupt handler which moves a character(word) to memory

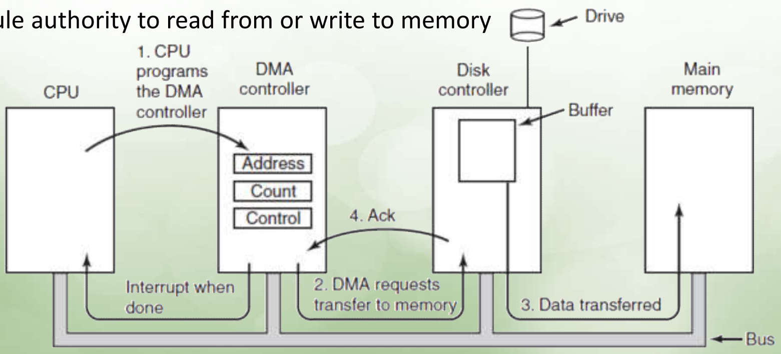
Interrupt-driven I/O is still inefficient in data transfer of large amount because the CPU has to transfer the data word by word between I/O module and memory

Afterwards, control is returned to the program which continues to read the next character/word

Eddie Law 55

Direct Memory Access

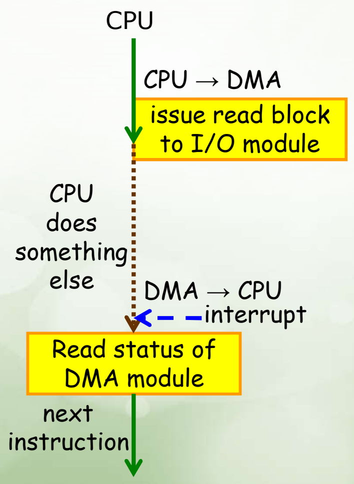
- Goal: relieves CPU's responsibility for data exchange with main memory
- DMA
  - Control exchanges of data directly between I/O device and memory
  - CPU grants I/O module authority to read from or write to memory



Eddie Law 56

Direct Memory Access: Operations

- Operations
  - Processor is only involved at the beginning and end of the transfer, continues with other work
  - Data block transfers to or from memory directly
  - An interrupt is sent when the transfer of an entire block is complete
- Any limitations?



Eddie Law 57

DMA: The Example

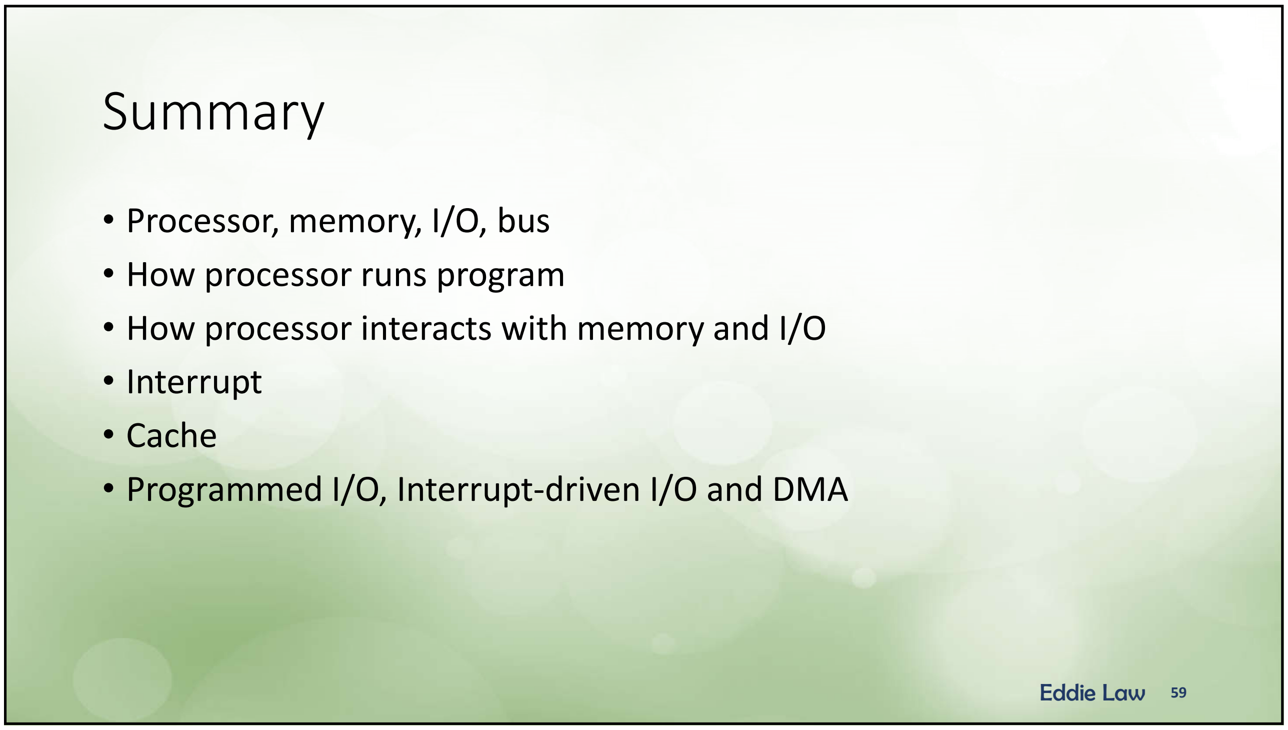
```
procedure readString(var s);
  [CPU requests DMA module to
    read some data]
  …
  Now, CPU does something else
  No checking on I/O status
  …
```

The DMA module starts reading each word of the data and save them in the memory; transparent to the process

DMA I/O is more efficient in large data transfer because the interaction with the I/O module and data transfer between I/O module and memory are performed by the DMA module.

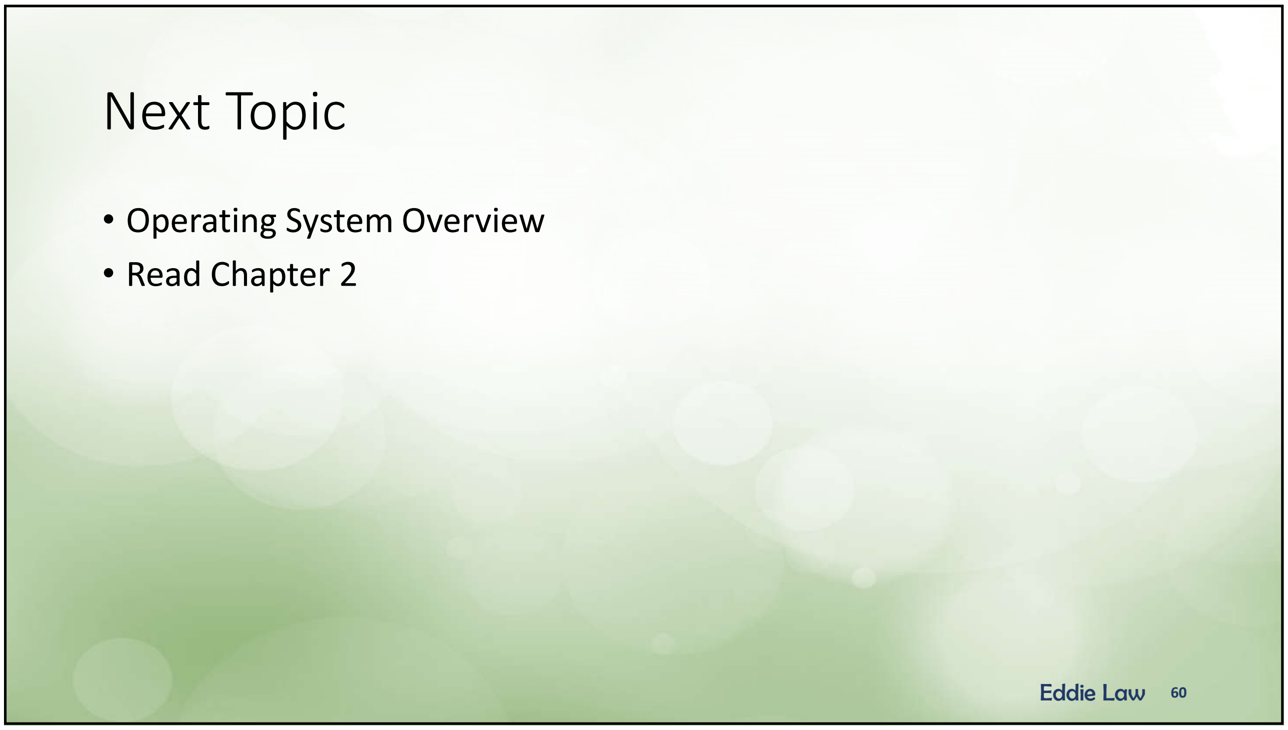
After the DMA has transferred all the data requested to the memory, it notifies that it has finished the I/O by sending the CPU an interrupt

Eddie Law 58



Summary

- Processor, memory, I/O, bus
- How processor runs program
- How processor interacts with memory and I/O
- Interrupt
- Cache
- Programmed I/O, Interrupt-driven I/O and DMA



Next Topic

- Operating System Overview
- Read Chapter 2