only appear on MCQ

Cx1106
Computer Organization and Architecture

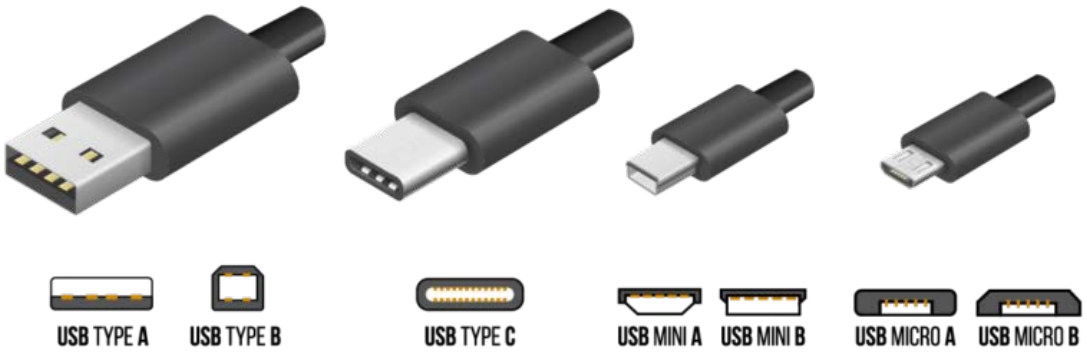
Communication and User Interface

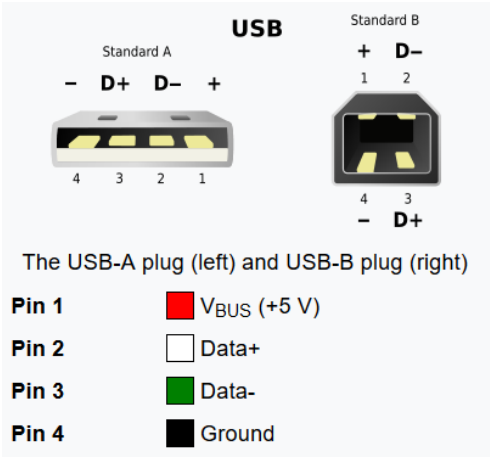
Oh Hong Lye
Lecturer
School of Computer Science and Engineering, Nanyang Technological University.
Email: hloh@ntu.edu.sg

Introduction

- This chapter introduce some of the common
  - Wired and wireless communication technology used in computers.
    - USB and HDMI
    - WiFi and Bluetooth
  - User interface devices used to transfer real world data.
    - Keyboard/Mouse
    - Capacitive Touch devices
    - Camera
    - Microphone
    - Display
    - Audio Speakers
- Only a general overview of the above will be discussed, technical details will not be covered.

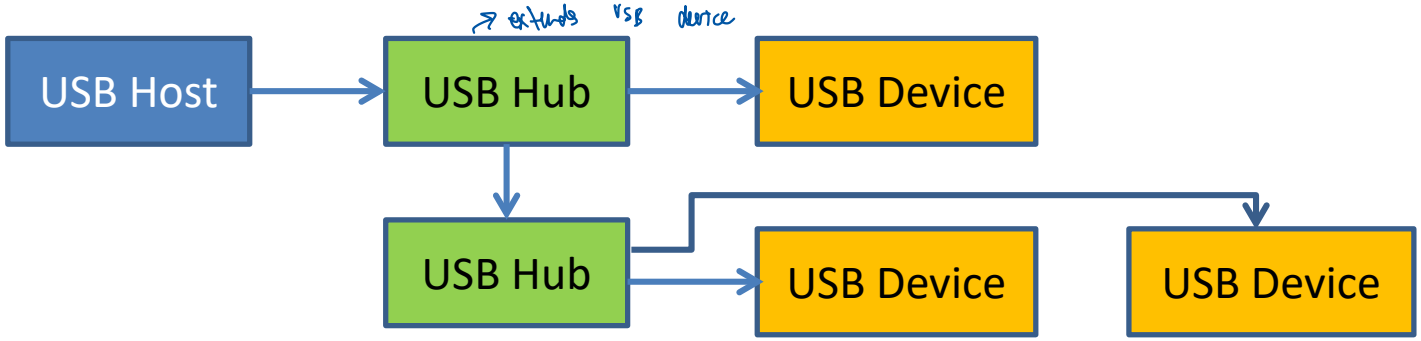
Universal Serial Bus (USB)





- Serial Bus designed to standardize connection of computer peripheral devices (Keyboard, Mouse, Printers, Disk Drives, Network Adapters, Digital Cameras etc).
- Effectively replaced many interface buses e.g. Serial Bus (COM Port), Parallel Port etc
- USB1.0 standard only supports a transfer rate of 12Mbps but has progressed over the years and the latest USB3.2 standard supports up to 20Gbps transfer.
- 4 basic pinout: VBUS (+5V Power supply), Data+/Data- (Data pins) and Ground pin.

USB Topology and interface



- All data transactions are initiated by the USB Host (typically resided on the computer). USB Peripherals (Mouse, Keyboard etc) can be connected directly to the Host or indirectly via the USB Hub devices.
- Host will assign address to devices connected to it to enable proper communication.
- All data transaction is with respect to the USB Host, i.e. data going into the Host is known as IN transaction and data going out of the Host is known as OUT transaction.
- Power to the devices can be supplied by the Host or Hub, known a bus-powered, or the device could have its own power source (self-powered).

USB Enumeration and Device Class

- When the USB device is first connected to the Host, it will undergo an enumeration process where the device and the host will exchange information on their capability and requirement. E.g. a USB mouse when connected with inform the Host its Vendor and Product ID, the device class it support, bandwidth required etc.
- The host cannot communicate with the device until it is properly enumerated.
- Once all the device information is transferred to the Host, the host will check if it has the required device driver to support the USB device according to the USB device class that the device belongs to.
- There are many USB Device Classes, common ones are
  - Human Interface Device (HID) used in Mouse/Keyboard.
  - Communication Device Class (CDC) used to implement Virtual COM Port e.g. Arduino Board, MSP432 Launchpad used in the lab.
  - Mass Storage Class (MSC) used to interface to external USB HDD/SSD.
  - USB Audio Class used to stream audio to USB headset/Microphone.

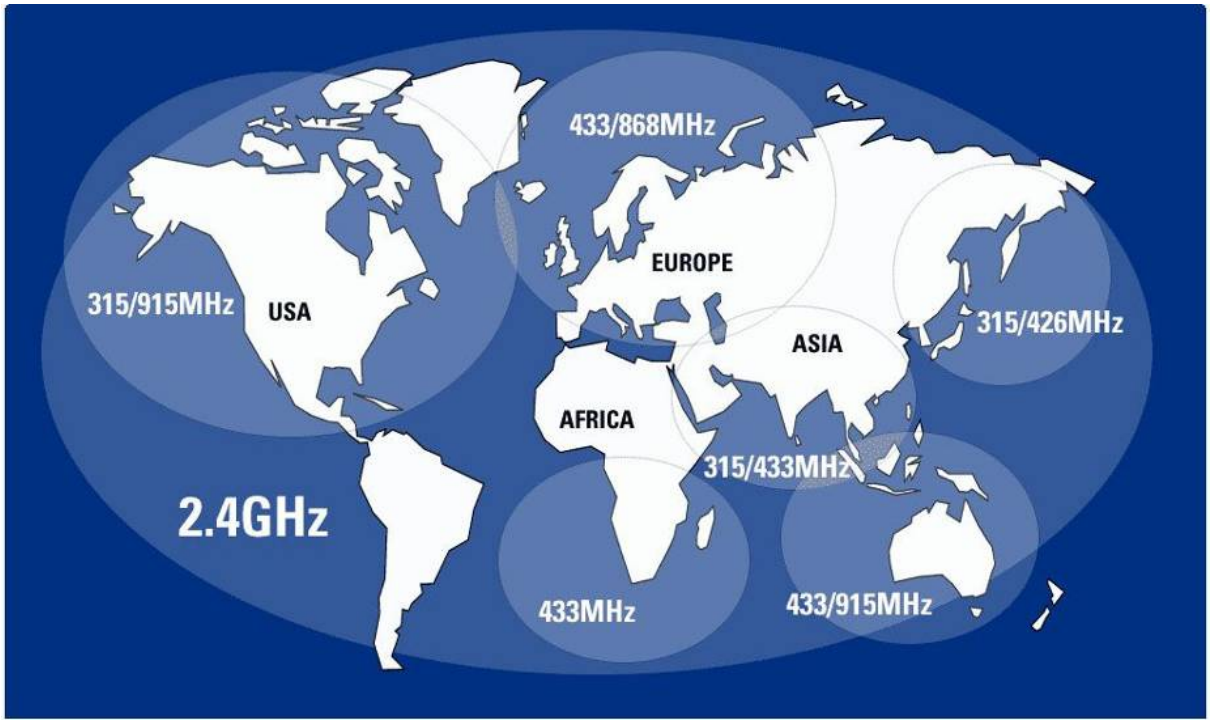
High-Definition Multimedia Interface (HDMI)

- HDMI is a proprietary audio/video interface for transmitting uncompressed video data and compressed/uncompressed digital audio data from a source device, such as a display controller in a computer, to a compatible HDMI receiver such as computer monitor, digital television, or digital audio device.
- In addition to transferring audio/video data, the CEC (Consumer Electronics Control) capability allows HDMI devices to control each other when necessary and allows the user to operate multiple devices with one handheld remote control device.
- Three commonly used type of HDMI connector type are shown in the figure on the right. Starting from the left: Type D (micro), Type C (Mini) and Type A.

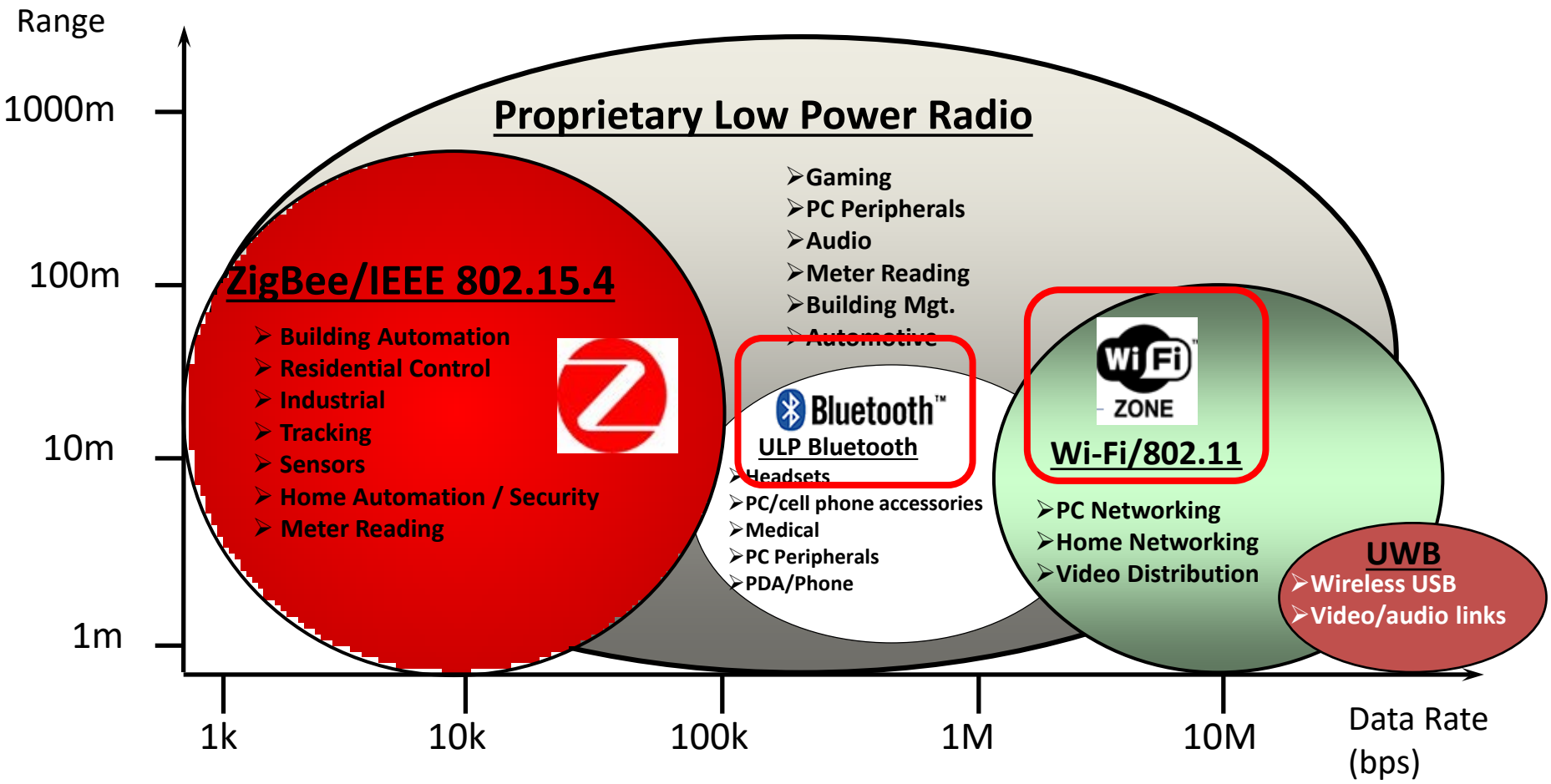
- The original HDMI v1.0/1.1 can only support a transfer rate of 3.96Gbps, allowing video format up to 1080p 60fps.
- The latest HDMI v2.1 can achieve 42.6Gbps, allowing 8K resolution video to be displayed.

Industrial, Scientific and Medical (ISM) RF Band

- A range of "Royalty Free" Radio Frequency Bandwidth
- Some are applicable world wide while some are restricted to certain geographical regions.
- The two commonly known world wide ISM bands are 2.4Ghz and 5.8Ghz.



Wireless Standards in ISM



Source: Texas Instruments

WiFi

- A family of wireless networking technologies, based on the IEEE 802.11 family of standards, commonly known as “Wireless LAN”.
- Operates in the 2.4Ghz and 5.8Ghz RF range.
- Two common topologies: Infrastructure and Adhoc.
- Infrastructure Mode
  - Uses Star topology, at the center of the network is an Access Point or Router, connected to devices at the end.
- Adhoc Mode
  - Peer to Peer connection
- Transmission Range generally between 20m to 150m, factors affecting the range include transmission frequency, transmission power and interference.
- Bandwidth up to ~10Gbps range for the latest 802.11ax (WiFi 6).

Bluetooth

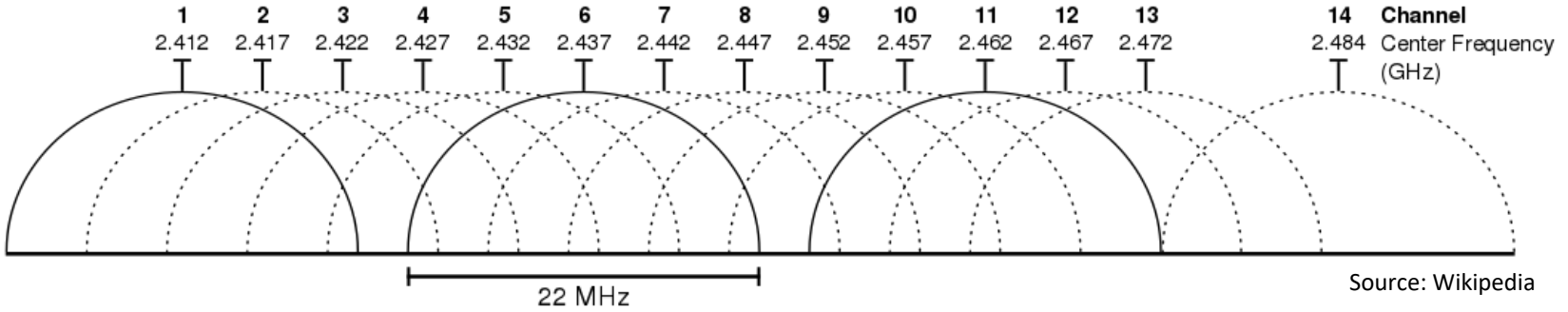
- Mainly used for low data rate wireless transmission with focus on low power consumption.
- Example: bluetooth headset, smart wearables, Bluetooth mouse etc.
- Operates in 2.4Ghz range
- Transmission range up to 100m but typically kept to 10-20m to keep power consumption low. Factors affecting transmission range is similar to that of WiFi.
- Star topology
- Transfer rate in order of Kbps and Mbps.
- The Bluetooth standard defined two different Bluetooth protocols: Bluetooth Classic (BT Classic) and Bluetooth Low Energy (BLE).
- These are based on two completely different network protocols and are not compatible. But most Bluetooth Hosts today are "Dual Mode" host so is able to connect to both BT Classic and BLE devices.

Factors affecting transmission range

- Transmission power
  - Transmission range increase as transmission power increase.
- Transmission frequency
  - Higher frequency signal experience higher attenuation when propagating through the air or other medium.
  - All things equal, higher frequency signal has lower range than lower frequency signal.
- Interference
  - Many commonly adopted standards such as Wifi and Bluetooth works in the same 2.4Ghz ISM band. Their transmission will interfere with each other.
  - The closer the transmitter is to each other, the stronger the interference.
  - Even the micro oven in your kitchen operates in the 2.4Ghz range!

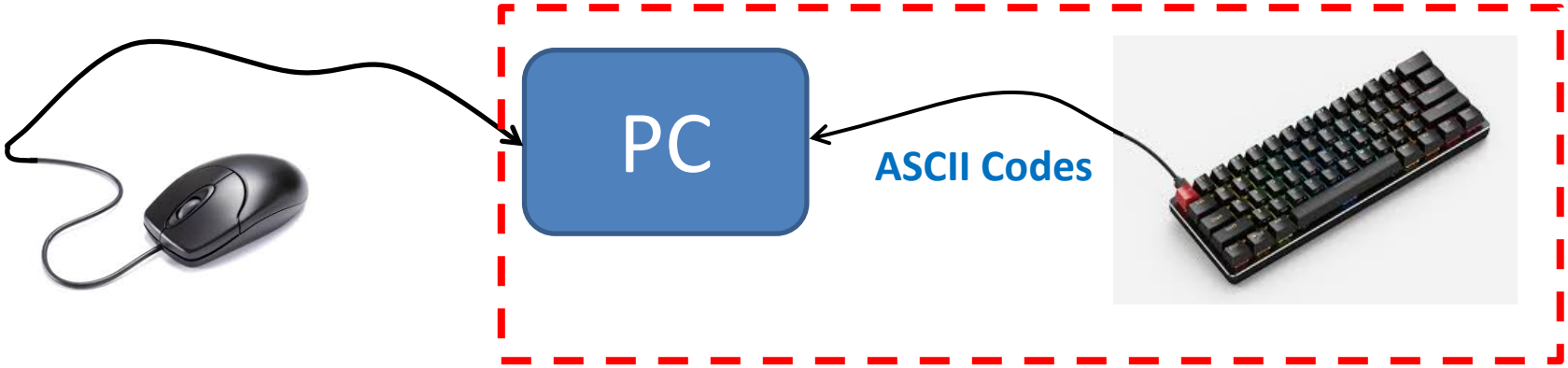
9 chances at 5.8 Gig wifi dropping off would actually be higher than 2.4 gHz, if all things are equal

Mitigating Interference (some methods)



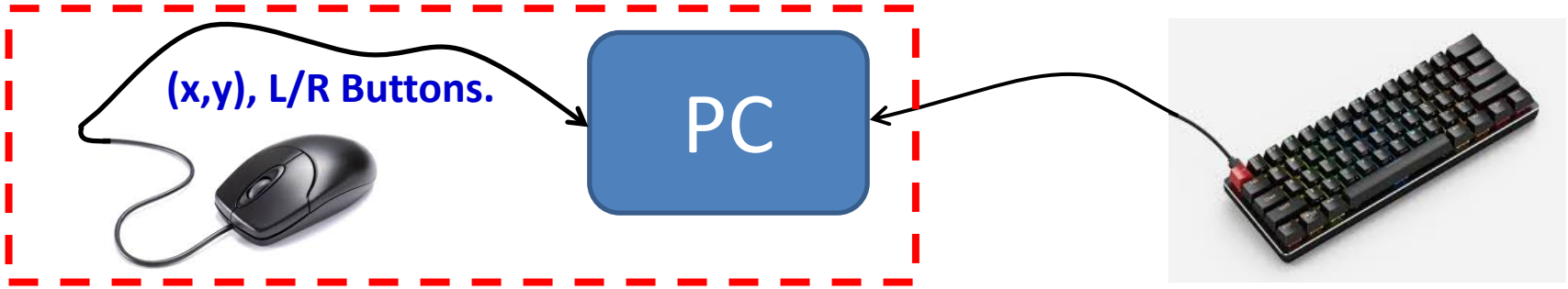
- "2.4Ghz ISM band" really consist of a band of frequencies between 2.4 and 2.5Ghz. Similarly, "5.8Ghz band" also span across a certain frequency range.
- Figure above shows the Wifi standard dividing the given frequency band into sub-bands known as channels.
- One common way to mitigate Wifi interference is to select different channels to use from your neighbours.
- For scenario of different wireless standard such as Bluetooth and Wifi, another common way is to use frequency hopping, i.e. to constantly hop from one channel to another so that transmission will eventually succeed. BT uses frequency hopping.
- There are other methodologies and techniques employed to further mitigate interference between transmitters.

Keyboard and Mouse



- **Keyboard**
- User input device for transmitting characters (alphabets, numbers, special symbols etc).
- Has a small micro-controller on board to detect the key press and sent their corresponding ASCII codes to the computer.
- Connection to PC is via wired or wireless means.
- Wired keyboard today mainly uses USB interface, keyboard will be enumerated as a HID Keyboard device.
- Wireless keyboard mainly operate in the 2.4Ghz ISM, technology could be Bluetooth or proprietary 2.4Ghz RF protocol.

Keyboard and Mouse



- **Mouse**
- User pointing device, tracks its position by mechanical or optical means.
- Information reported are the (x,y) coordinates and the left/right mouse button.
- More information may be reported for mouse with more advanced features.
- Information reported back to PC periodically and is known as the scan/polling rate of the mouse.
- Typical mouse has a scan rate of 125Hz, gaming mouse scan rate could be up to 1000Hz or higher. Higher scan rate typically implies better response.
- Another parameter is the Dot-Per-Inch (DPI), which measures how fine the mouse could track the physical movement, higher DPI implies higher sensitivity to small physical mouse movements.
- Connection to PC utilise similar technology as Keyboard.

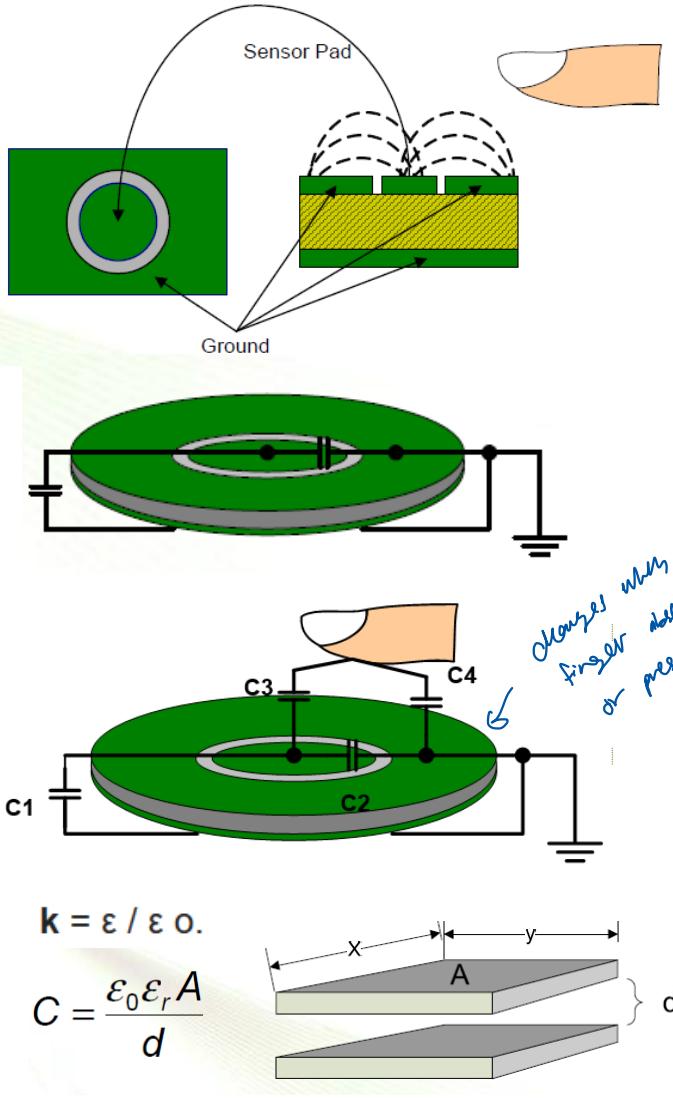
ASCII Table

LS \ MS	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	'	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	
F	SI	US	/	?	O	_	o	DEL

ASCII Character Set (7-Bit Code)

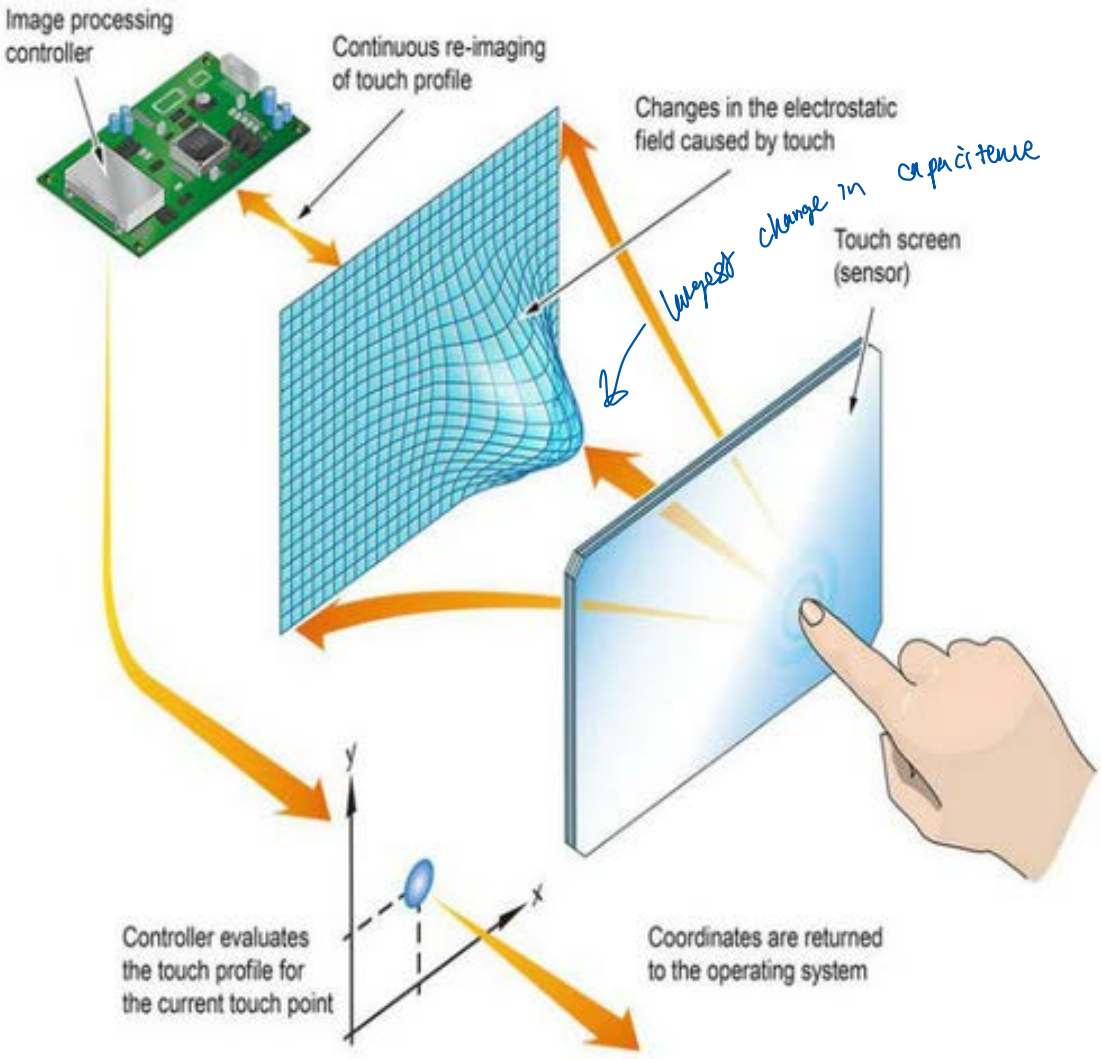
Capacitive Touch Interface

- A capacitive touch pad/button on the right is seen as a capacitor to the processor it is connected to.
- When a conductive element is present, e.g. finger, the effective capacitance of the setup increases.
- Capacitance is also affected by any dielectric e.g. gloves, plastics, liquid between the finger and the pad.
- Capacitance is directly proportional to dielectric constant (k) and air typically has a smaller dielectric constant (~1) compare to all other materials (>1).
- That's why your phone touch screen, which is typically capacitive touch based, don't work as well if you have a glove on or the screen is wet.
- Calibration is done under assumption of human finger touch.



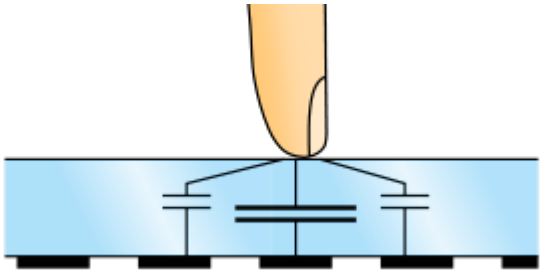
Source: Texas Instruments

Capacitive Touch Screen

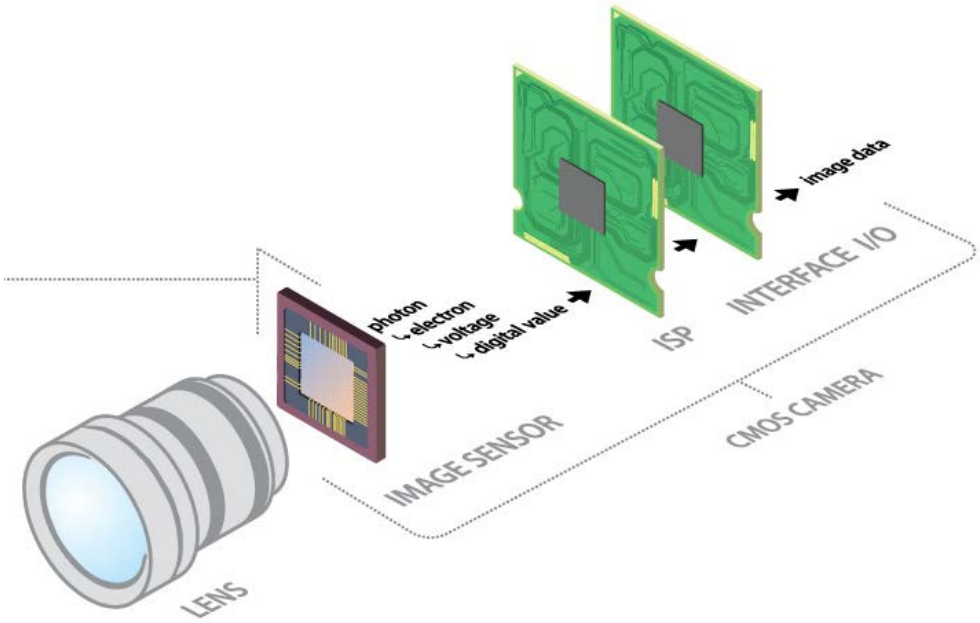


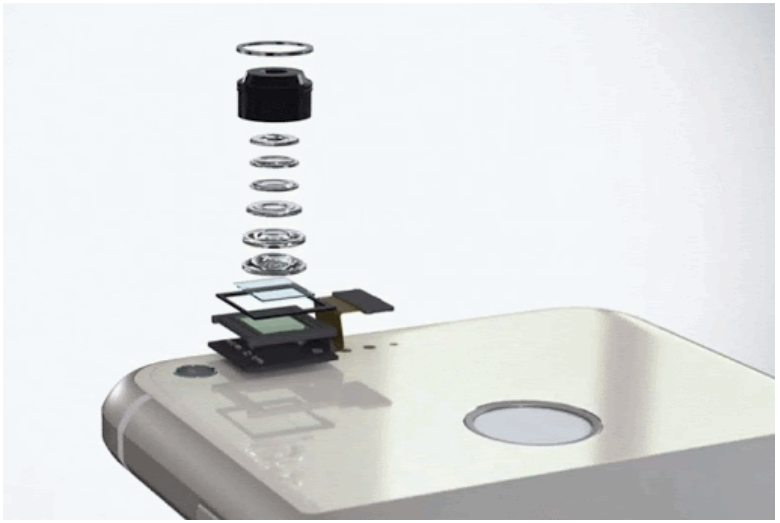
Source: Wikipedia, Arrow.

- Capacitive touch screen consist of an array of electrodes functioning as capacitive touch sensors.
- Finger touching any part of the screen will affect one or more electrodes, with the one closest having the most change to its nominal capacitance.
- A controller is used to process the information to derive the exact position of contact.



Camera





Source: Thinklucid.com, wired.com

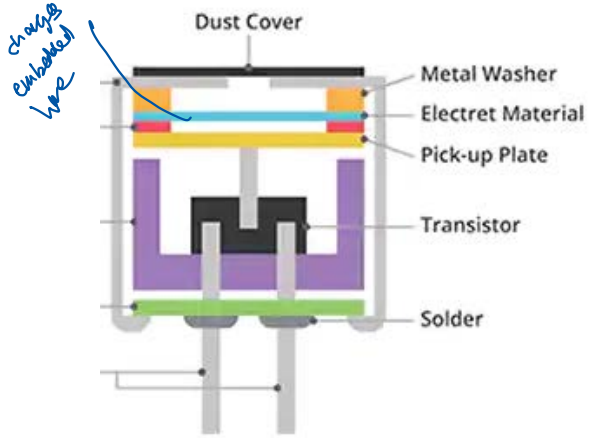
- Typical camera on phones or PC today uses CMOS camera module, it consist of the sub-modules shown in the figure above.
- Sub-modules: Lens, Image Sensor, Image Signal Processor and Interface I/O.

Camera Sub-Modules

- Lens
  - Optical lens to focus the real world images onto the image sensor.
- Image Sensor
  - Mostly CMOS based these days.
  - Array of sensors that transduce photons (light information) to electrical signals (analog).
  - Analog-to-Digital conversion of the analog electrical signal to its digital equivalent for processing in the ISP.
- Image Signal Processor (ISP)
  - Processor designed to specifically handle image processing of collected image data from the sensor.
  - Processing done include signal conditioning, image format conversion, image post-processing/compression etc.
  - Common image format are YUV (luminance and chrominance) and RGB (Red, Green, Blue).
- Interface I/O
  - Re-formatting of image data from the ISP for delivery to the host processor.

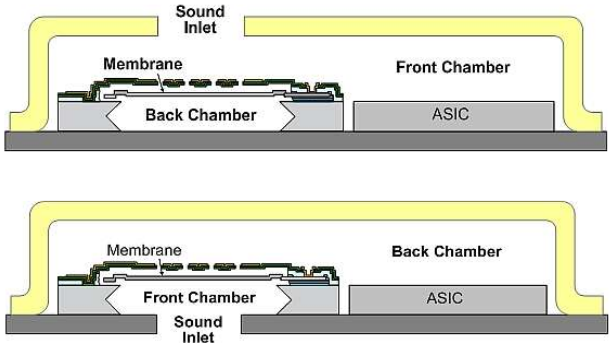
Microphones

ECM



MEMS Microphone

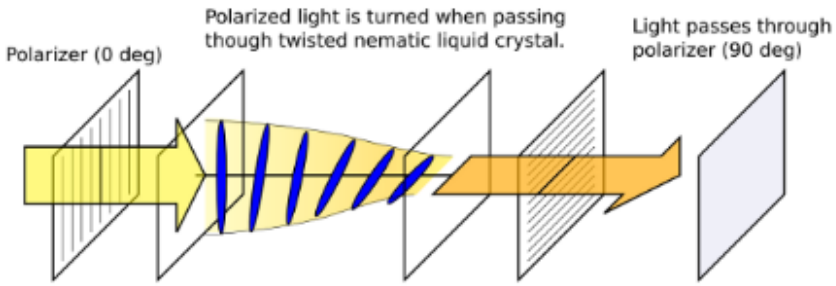
Source: EDN, DigiKey



- Two of the most popular types of microphones are micro-electro-mechanical system (MEMS) microphones and electret condenser microphones (ECM).
- Both MEMS microphone and ECM uses the variation in capacitance when the diaphragm is displaced by the sound pressure to transduce sound wave to electrical signals.
- Difference is the in ECM, the electrical charges needed to measure the change in capacitance is provided by the charges stored on the electret.
- In MEMS Microphone case, the electrical charges needed is provided by a charge pump instead.
- The transduced signal is analog in nature but an ADC could be added to enable a digital output.

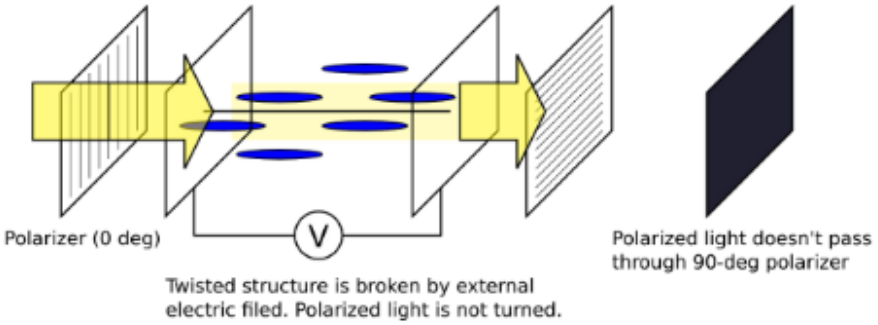
Liquid Crystal Display (LCD) Basics

Polarised light twisted (Light Passes)



Source: awawa.hariko.com

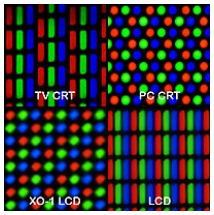
Polarised light not twisted (Light Blocked)



- Passive display technology, i.e. they don't emit light. Instead, they need a backlight in order for user to see the image.
- Liquid crystal is an organic substance that has both a liquid form and a crystal molecular structure. The rod-shaped molecules are able to kept their order in a particular direction or orientation although they are in liquid state.
- An electric field can be used to control the molecules orientation.
- Depending on their orientation, these molecules is able to twist the light passing through them.
- The amount of light that is able to pass through the polariser depends on its orientation with respect to the polariser.
- This result in light passing or being blocked from user point of view.

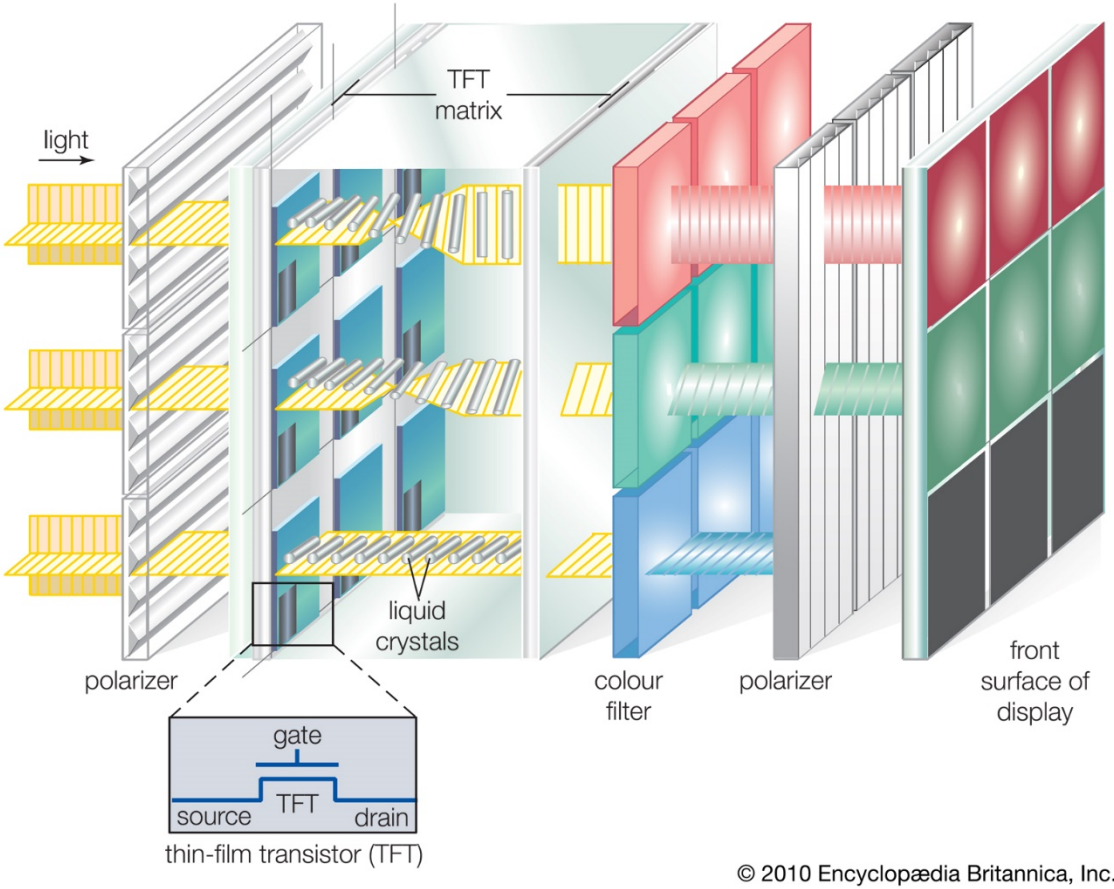
Colour LCD

Source: Wikipedia



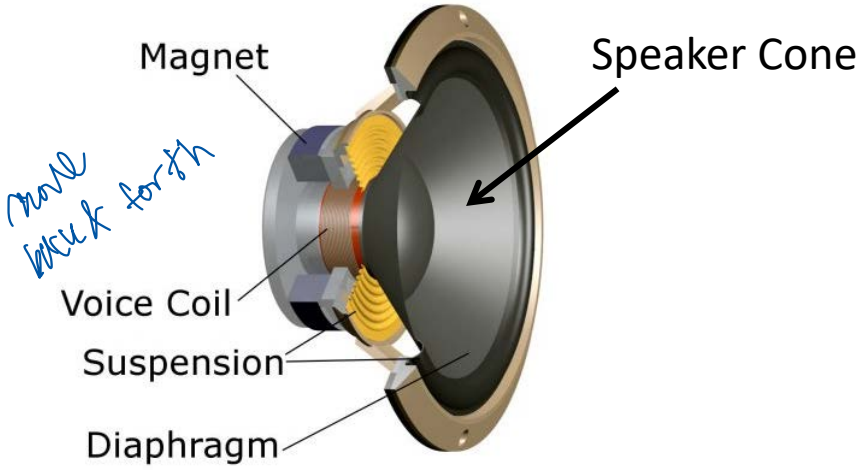
RGB Pixels Pattern on Colour LCD

Colour LCD



- Process for colour LCD is similar to that discussed in previous slide.
- Each pixel is associated with three colour filters to project the RGB colours.
- A varying orientation of molecules in the liquid crystal suspension. varies the amount of light allowed to pass through to the colour filter, thereby changing the colour picture on the display screen.

Audio Speaker



Source: Wikipedia

"basic operating principle"

"lecture notes are sufficient"

- The device transduces electrical energy to sound energy.
- The speaker cone vibrates, pushing and pulling the air to create sound waves.
- The conversion from electrical to mechanical energy occurs through an electromagnetic coil and magnet combination attached to the cone. This coil moves the speaker cone back and forth as its electromagnetic field changes with the electrical current passing through it, converting the mechanical energy to sound energy.

Cx1106
Computer Organization and Architecture

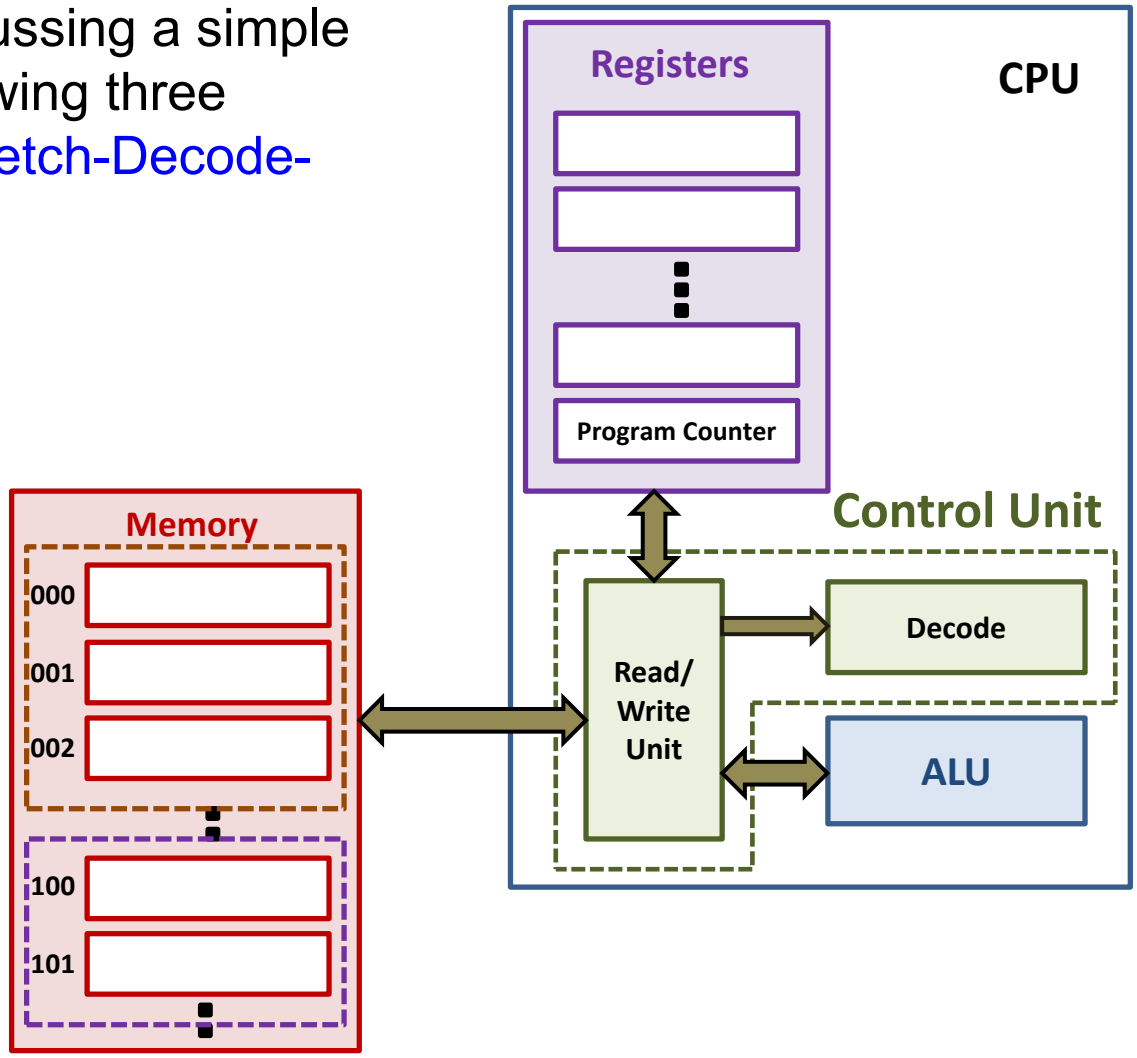
Processor Pipeline

Oh Hong Lye
Lecturer
School of Computer Science and Engineering, Nanyang Technological University.
Email: hloh@ntu.edu.sg

Review of a Simple CPU

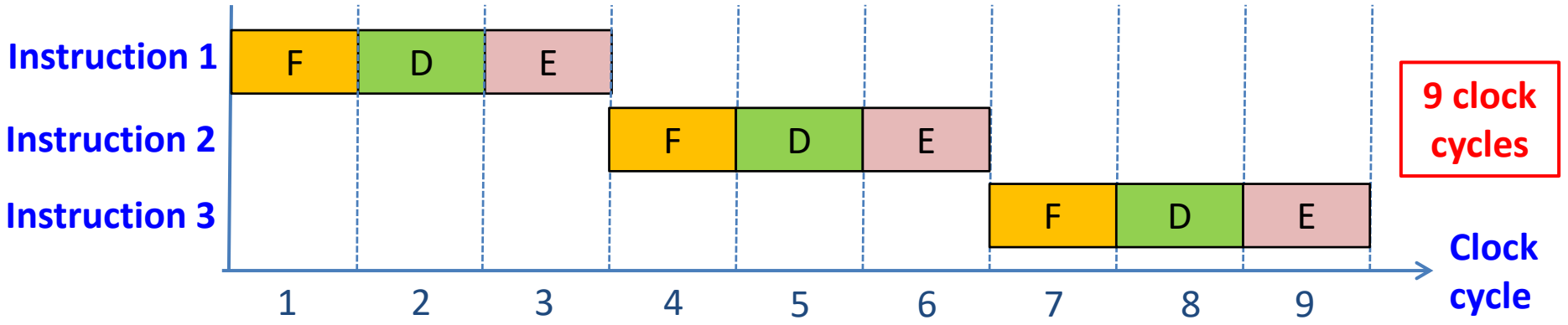
- So far we have been discussing a simple CPU which does the following three operations sequentially: Fetch-Decode-Execute

- In the simple CPU, the Fetch-Decode-Execute cycle of an instruction must complete before the next instruction is fetched



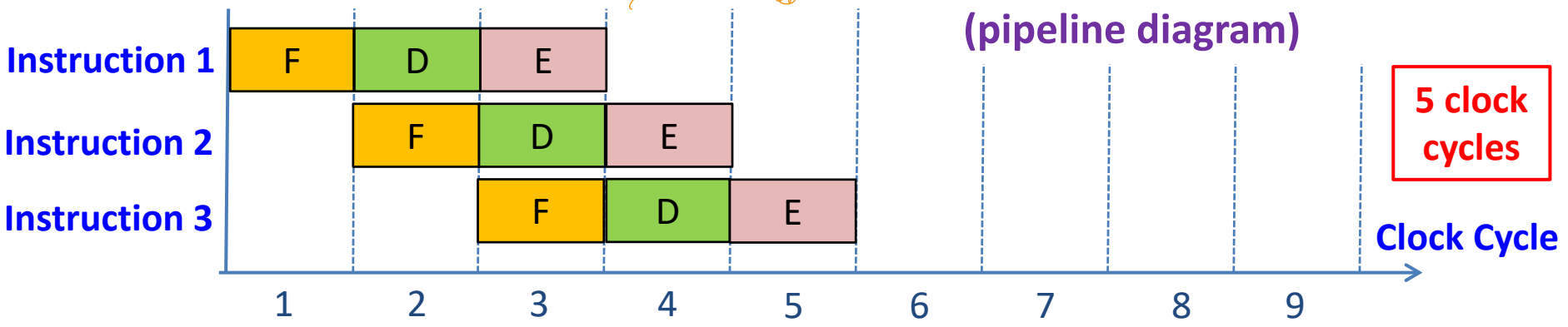
Instruction Execution – Simple CPU

- Consider the simple CPU discussed in previous slide.
- Each instruction is broken down into the following stages
  - Fetch Instruction (F),
  - Decode (D)
  - Execute (E)
- Executing each instruction is equivalent to cycling through the three stages F, D and E. If we assume that each stage takes one clock cycle, the time taken to execute 3 instruction will be 9 cycles.



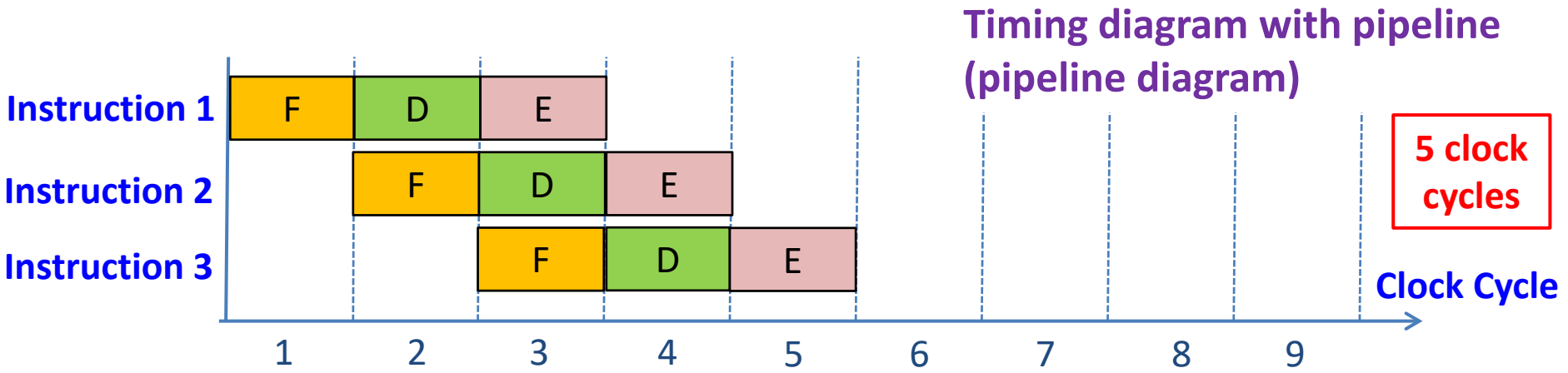
Executing each stages in parallel

- Now, consider a processor which can execute the individual stages simultaneously.
- When instruction1 is in the Decode (D) stage, the processor is able to fetch the machine code of instruction2 from the memory.
- And when instruction3 in the Execution (E) stage, the processor is able to Decode instruction2 and Fetch the machine code of instruction3.
- Total time taken to execute the same three instructions has reduced to just 5 clock cycles.



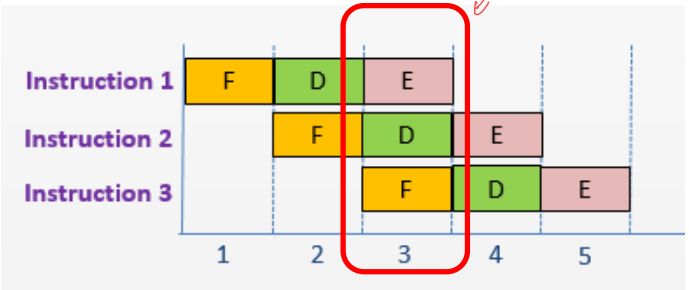
Pipeline Architecture

- This is an example of a Pipeline Processor Architecture.
- Pipelining is about partitioning an instruction into simpler stages and assigning resources to allow these stages to be executed simultaneously.
- There are many ways to partition an instruction, the example shown here is one of the simplest. Its not uncommon for more complex application processors to have more than 10 pipeline stages.



Pipeline – A peek into Processor Internal

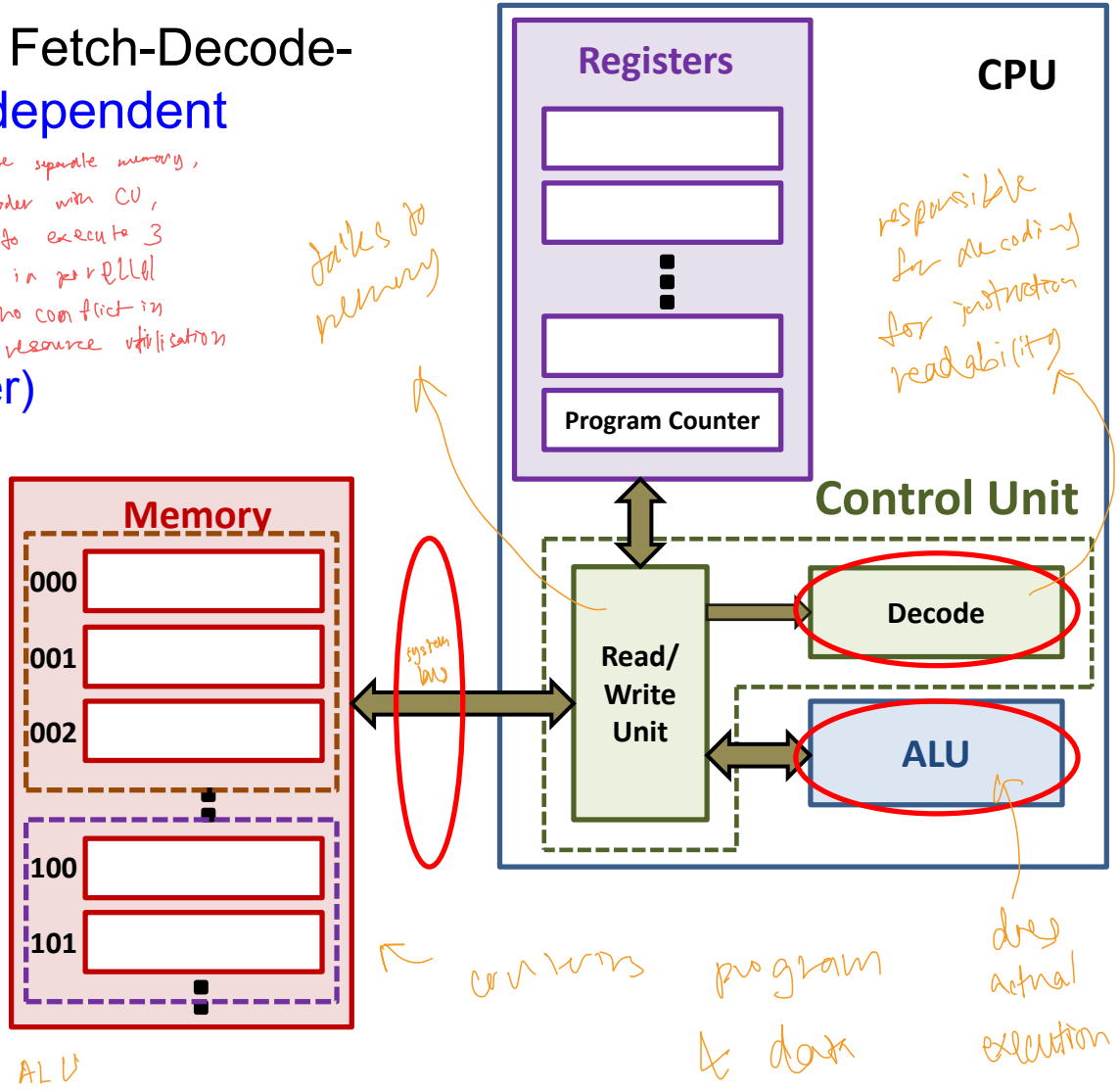
- Pipelining is possible if the Fetch-Decode-Execute operations use independent resources
- For example
  - Fetch (External buses)
  - Decode (Instruction Decoder)
  - Execute (ALU)



Fetch Instruction

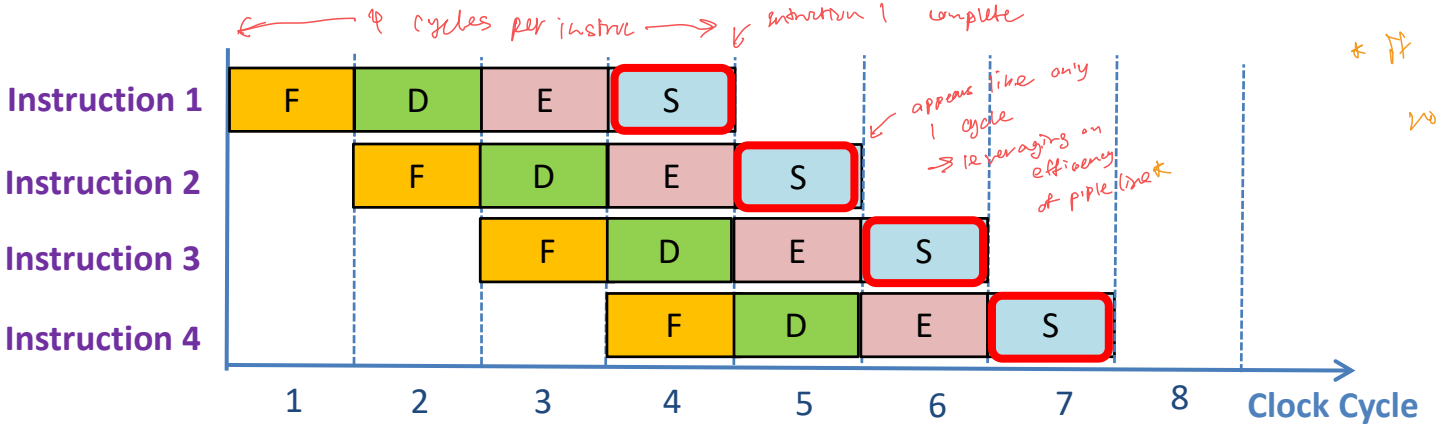
Decode Instruction

Execute Instruction



Pipeline Efficiency

- The efficiency of a Pipeline architecture is maximised when the pipeline is filled
- Supposed we have a 4-stage pipeline processor
  - Instruction Fetch (F)
  - Instruction Decode (D)
  - Instruction Execution (E)
  - Data Store (S)
- It takes 4 cycles to fill the pipeline and release the first instruction from the pipeline.
- But after the pipeline is filled, it is able to release one instruction every cycle, giving the effect of 'executing' one instruction every clock cycle.



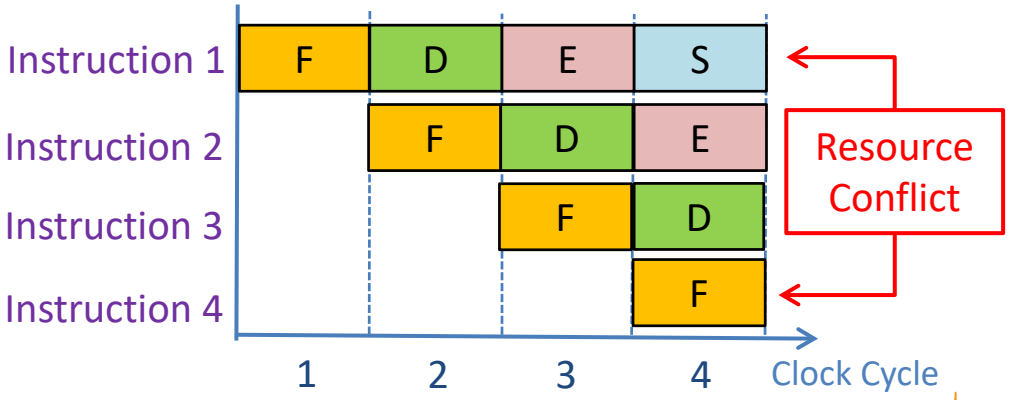
Pipeline Conflicts

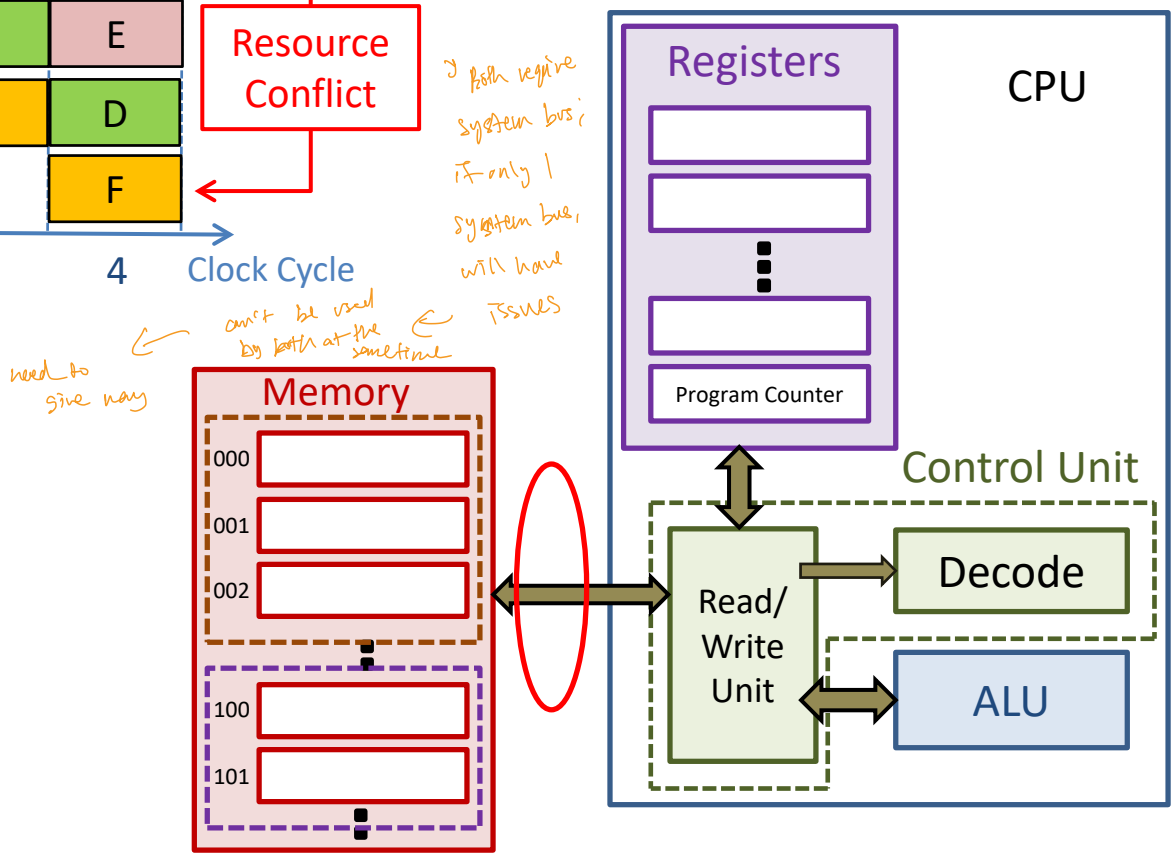
- Pipeline efficiency mentioned in the previous slide will be reduced drastically if there are disruption to the pipeline.
- Events that disrupt the pipeline is known as pipeline conflicts.
- Pipeline conflicts typically cause a temporary halt to the pipeline or in some cases, the processor has to flush the instructions in the pipeline and reload with a fresh set of instructions.
- These actions result in additional time to execute a particular set of instructions.
- Since the total time taken to execute the instruction is longer, the number of instructions that can be released from the pipeline is reduced, which means the effective performance of the processor is lowered.
- In this module, we will look at three sources of pipeline conflict
  - Insufficient Resource
  - Data Dependency between instruction
  - Pipeline flushing due to Branch Instruction

Resource Conflicts

- Consider a processor with 4 pipeline stages: Fetch Instruction (F), Decode (D), Execute (E), Store Result (S)

Example: Instruction 1 is storing result in memory when instruction 4 needs to fetch a new instruction





- Resource conflict occurs when two instructions attempt to access the same resource in the same cycle.

Resolving Resource Conflicts

- You need sufficient resources for a pipeline processor to work efficiently
- A pipeline processor with insufficient resources to operate each pipeline stage simultaneously will result in constant halting and flushing of pipeline.
- Resultant processor performance may even be worst than that of a processor with a simple CPU architecture.
- Its common for pipeline processors to have multiple resources: internal buses (data, instruction, peripherals), control and processing units etc to allow simultaneous operations in any instance.

Use of ARM instructions in Pipeline Examples

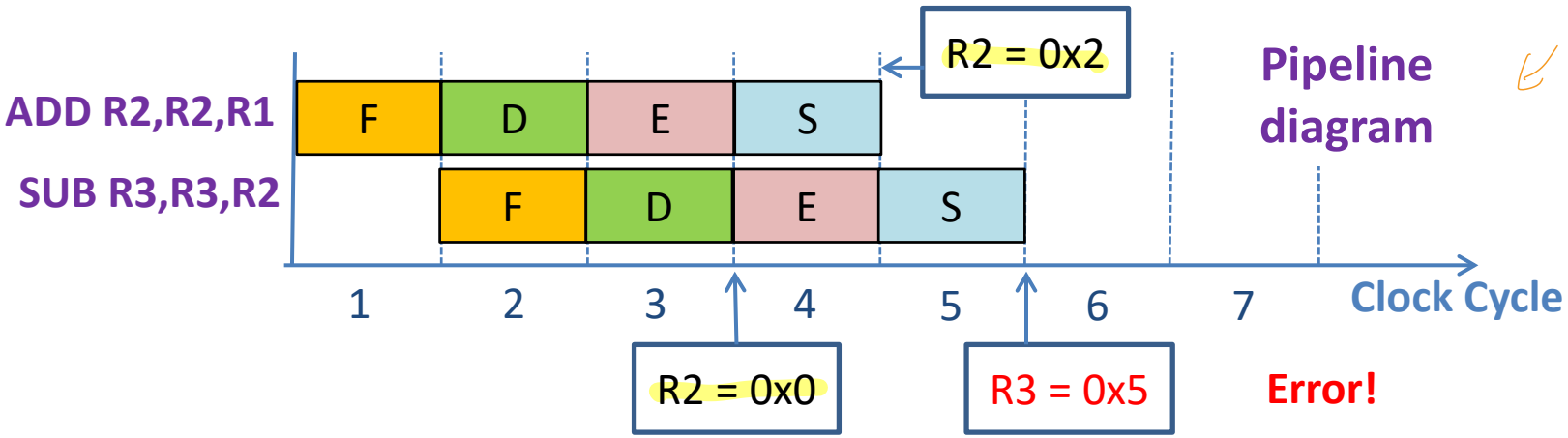
- ARM instructions are used in the Pipeline examples but note that the following key differences
  - The pipeline structure discussed used is NOT that of the ARM pipeline
  - For simplification of analysis, ALL the ARM instructions used are assumed to occupy only one word and each pipeline stage take one clock cycle to execute.

Data Dependency Conflict

→ pipeline conflict that happens when current instructions required some information that have not been updated yet by the previous instructions.

- Consider a 4-stage pipeline **(**FDES**).**
- Actual operation of each instruction e.g. ADD, SUB, is done during the Execute (E) stage.
- The resultant value of the execution is transferred to the destination during the Store (S) stage.
- In the example below, the old value (0x0) is still in `R2` when SUB instruction is in Execute (E) stage, so R3 will have the wrong value (0x5) instead of the correct value (0x3).

Assume initial register states: R1= 0x2, R2 = 0x0, R3 = 0x5

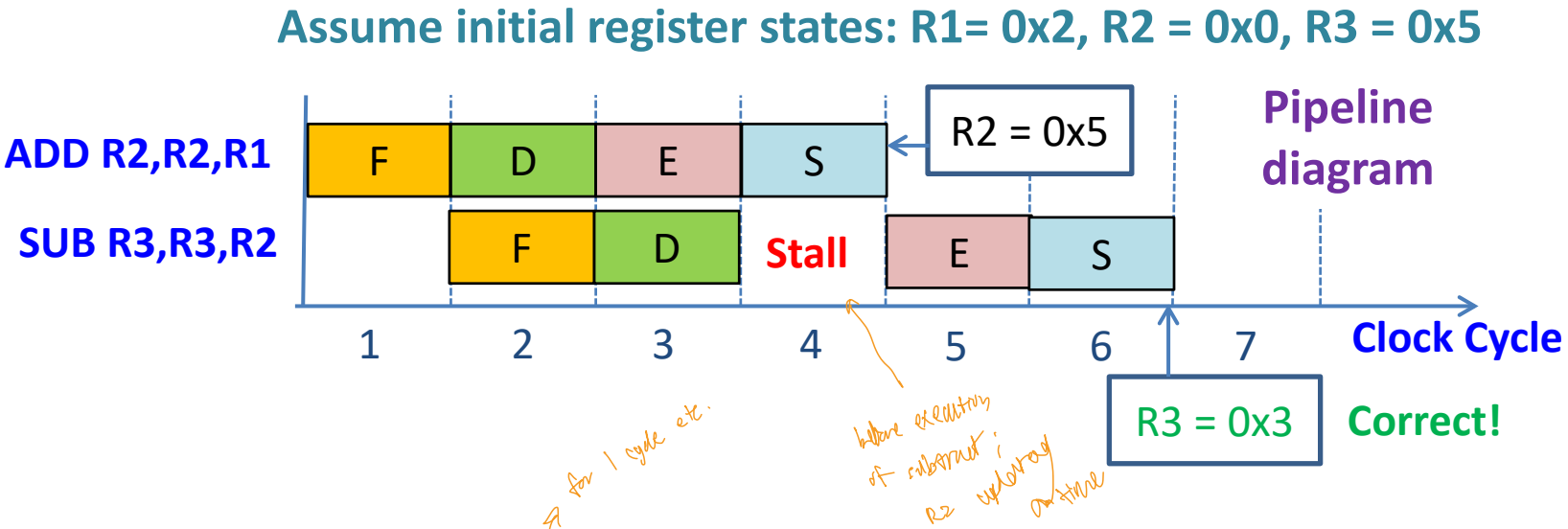


Data Dependency Conflict

- For the example in the previous slide, it will not be an issue if the processor has a simple CPU architecture where instruction execution is sequential. The current instruction will start its execution only after the previous instruction has complete. So the most updated copy of the memory/register value is always available to the current instruction.
- Due to the overlapping nature between instructions within a pipeline, there will be instances where a data required for the proper execution of the current instruction has not been updated yet.
- This is known as data dependency issue/conflict. It arises when the source operand of the current instruction is also the destination operand of a prior instruction.
- The exact separation between the two instructions in order for the issue to occur depends on the pipeline structure and design.

Resolving Data Conflict – Stall the Pipeline

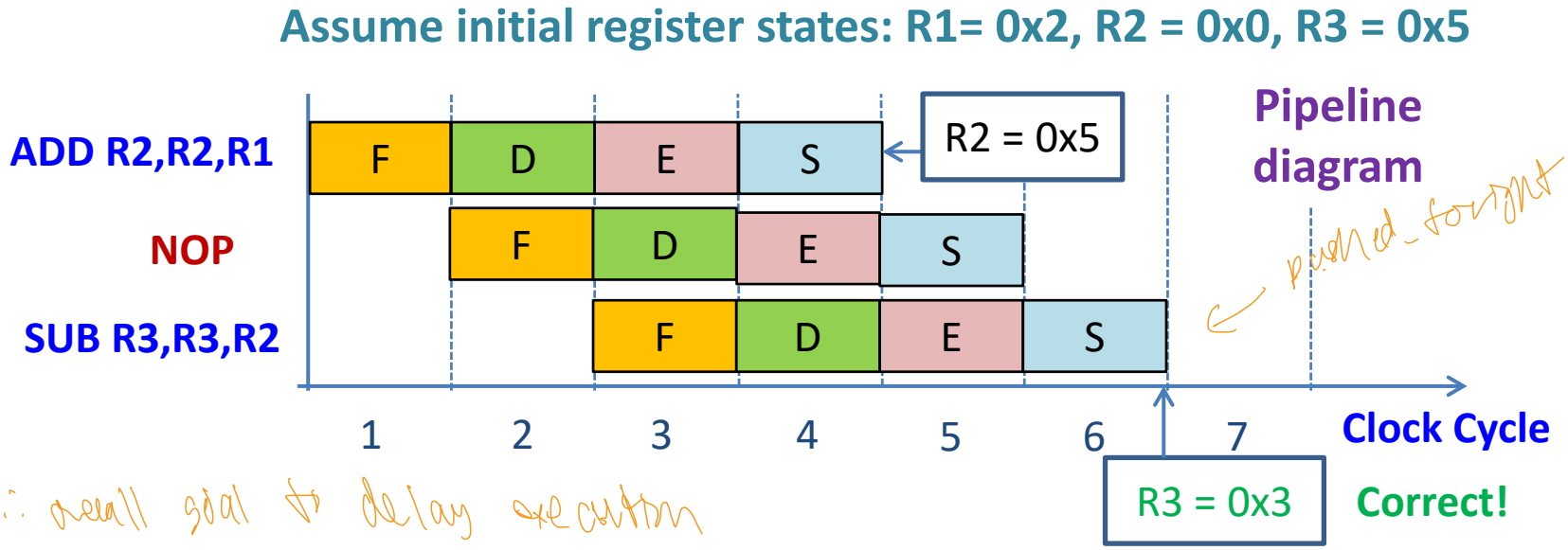
- Hardware circuitry can be used to detect data dependency between instructions
  - Compare destination identifier in the Execute Stage with source(s) in the Decode stage.



- If data dependency is detected (i.e. R2 matches), allow ADD to continue normally, but stall the Decode stage of SUB.
- After ADD completes, SUB is allowed to resume.

Resolving Data Conflict – Insert NOP Instructions

- Compiler can analyze and insert redundant instructions to reduce data conflict
  - Data dependencies are evident in instructions during compilation
  - Compiler inserts explicit NOP (No Operation) instructions between instructions with data dependencies
- Delay ensures new value is available in register but causes total execution time to increase

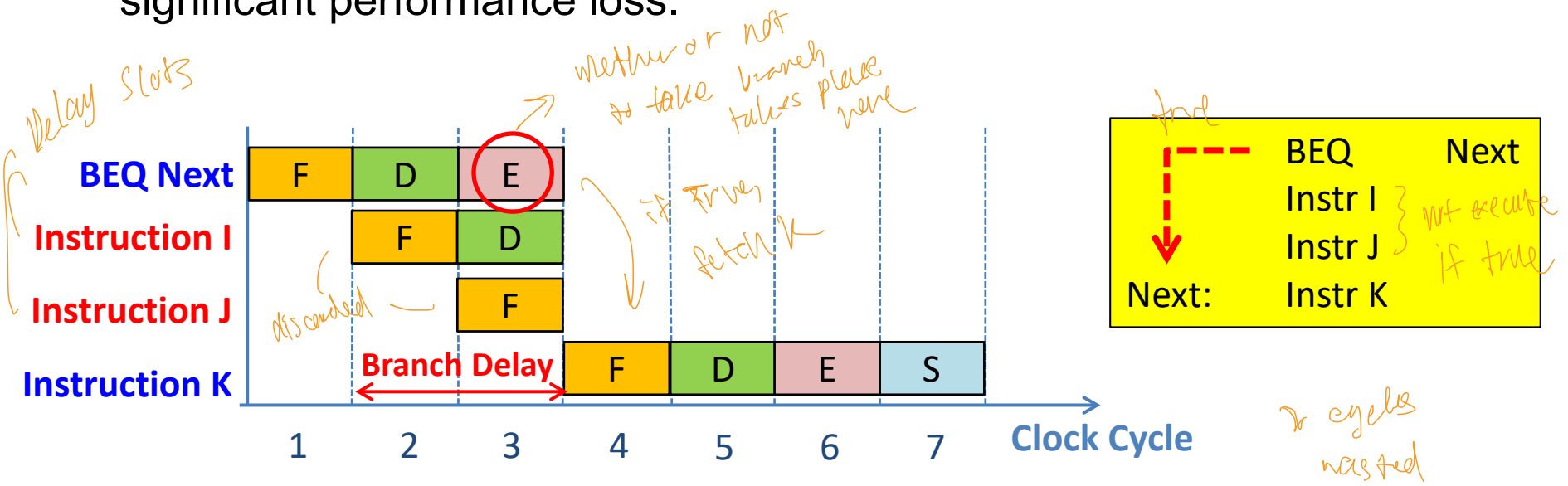


Branch Instruction

- Branch instruction usually need to perform two operations
  - Evaluate condition to determine if branch should be taken/not taken
  - If branch is taken, calculate branch target using adder in ALU
- Both operation requires an ALU to perform some computation and processing so a natural stage to do this is in the Execute (E) Stage of the pipeline.
- However, due to overlapping operations between instructions in a pipeline, the unnecessary instructions may already been introduced into the pipeline before the branch decision had been made.
- The processor would need to flush the pipeline to reload correct instructions according to the branch decision. Flushing of pipeline is equated to wastage in cycles for instruction execution.
- The number of cycles wasted/lost is known as Branch Delay.

Branch Delay

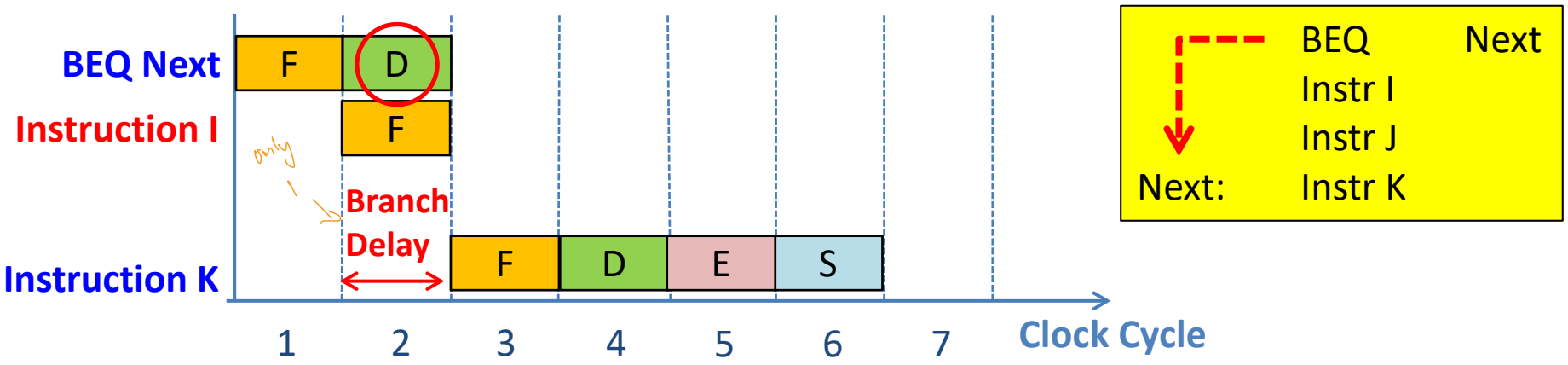
- Branch statements in pipelining can lead to Branch Delay which cause significant performance loss.



- The branch target is only known after the Execute stage, but by this time, Instructions I and J have already been fetched.
- Instructions I and J will be discarded, resulting in two-cycle branch delay.
- The two slots that are discarded is known as the Delay Slots
- Instruction I and J is known as the Delay Slot Instruction

Reducing Branch Delay

- Branch delay can be reduced by making the branch decision and calculating the branch target earlier at the Decode stage.
- An additional adder is introduced to be used in the Decode stage to enable earlier calculation of branch target.



- After the Decode stage, the branch decision and the branch target is known.
- Hence if branch is taken, only Instruction I needs to be discarded.
- Branch delay is reduced to one cycle.

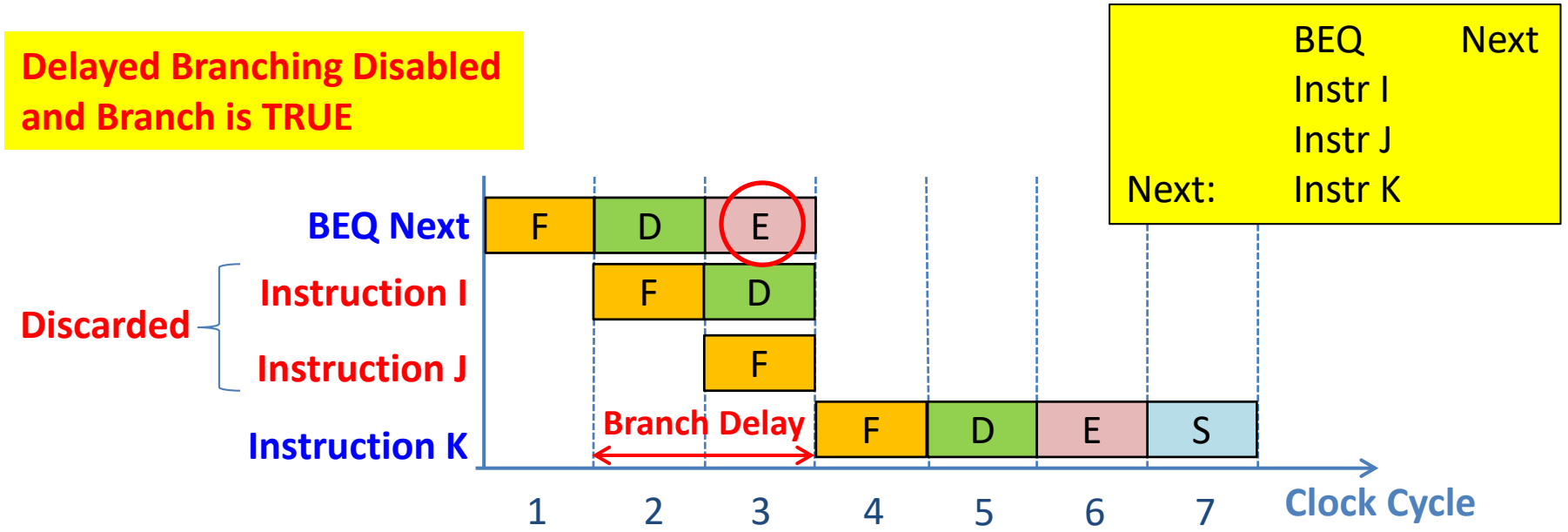
Delayed Branching

- In the previous implementation, the instruction(s) immediately following a branch is always fetched, regardless of the branch decision.
- If branch is taken, these instruction(s) will be discarded resulting in branch delay.
- Delayed Branching is a method that ensures no instructions are discarded after the branch. That means delay slot instructions are always executed.
- User or Compiler can schedule independent instructions to be filled in the delay slots after the branch.
- If such independent instruction exists, they will always be executed, leading to zero branch delay, since no cycles is wasted.
- If an independent instruction cannot be found or if there are insufficient number of independent instructions to fill the delay slots, NOP instruction(s) should be used to populate the delay slots to preserve the correctness of the original program logics.

Delayed Branching

can be enabled/disabled by user

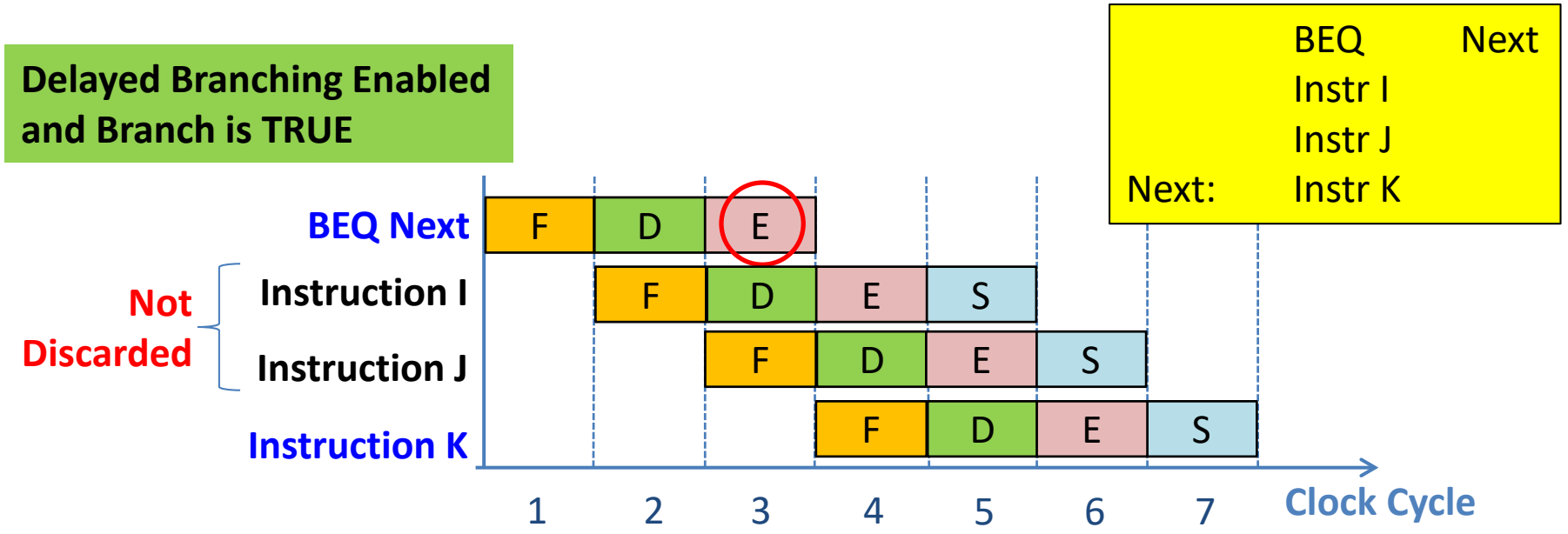
- For simplification sake, we can view Delayed branching as a feature in the processor
- If Delayed branching is disabled, the processor pipeline will discard the delay slot instructions if the branch is True to preserve the correctness of the program logic, at the expense of cycle wastage.



Delayed Branching

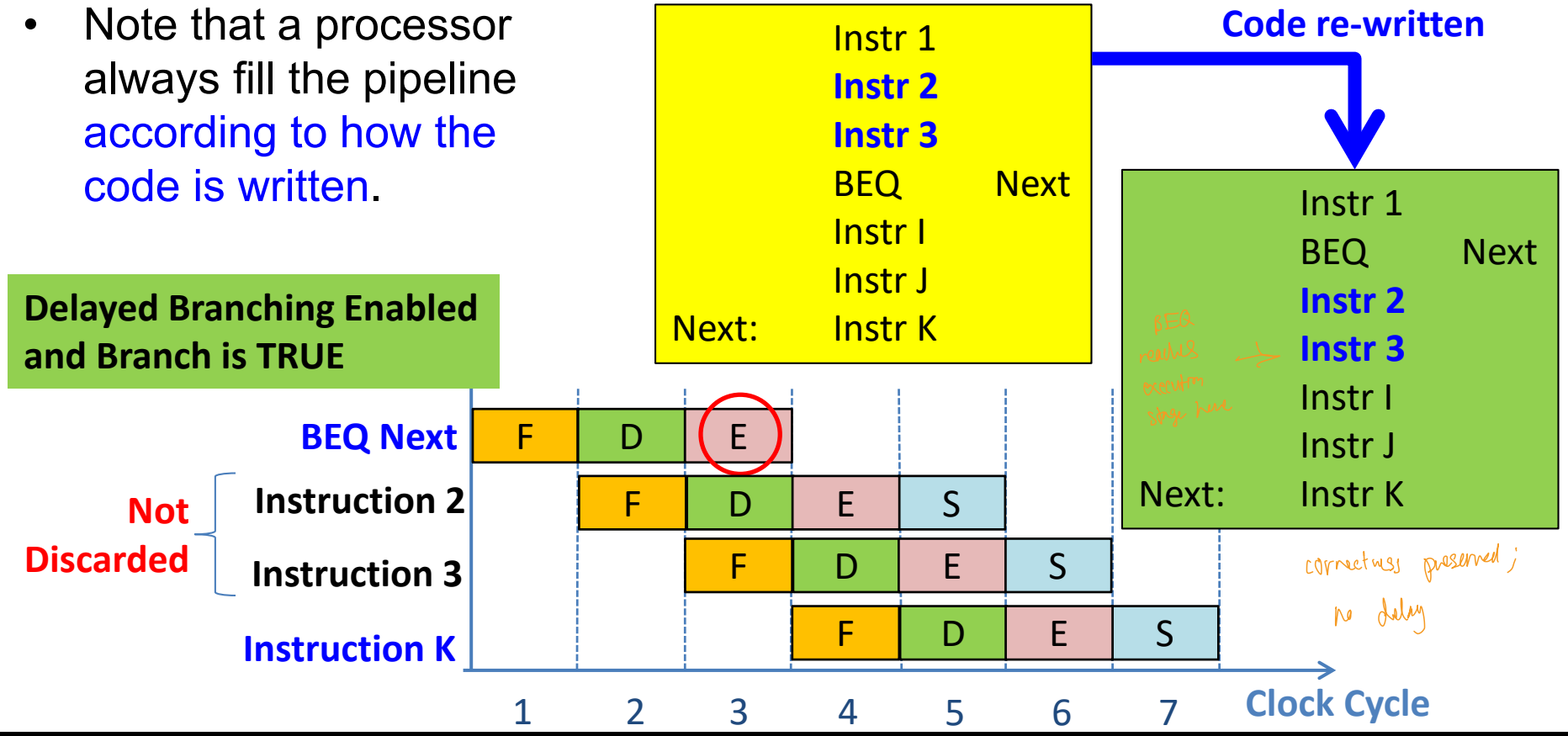
- If the Delayed branching is enabled, the processor will execute the delay slots instructions regardless of true or false branch decision so no cycles are wasted. ⇒ addresses reduction in efficiency problem cause by branch delay.
- However, the program logic would be wrong if instruction I and J are executed, when the branch is True.
- So, some other instructions (other than I and J) need to fill the delay slots.

Delayed Branching Enabled and Branch is TRUE



Delayed Branching

- The user or compiler needs to fill the delay slots with Independent instructions.
- For example, if instruction 2 and 3 are independent instructions, they can be made to occupy the delay slots by re-writing the program code.
- Note that a processor always fill the pipeline according to how the code is written.

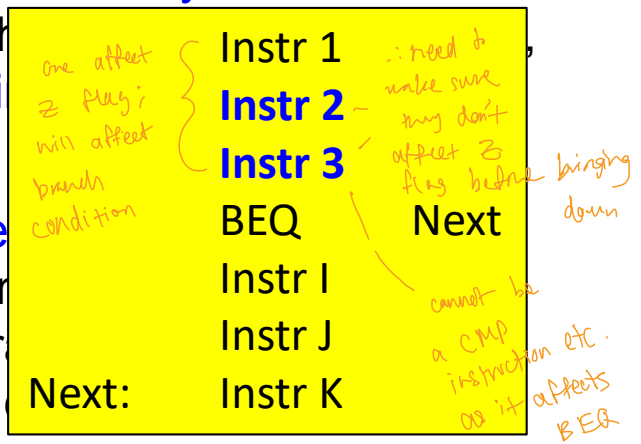


Independent Instructions

- Independent instructions refer to instructions which will always be executed in the original program flow, regardless of whether the branch is taken or not, but does not play a part in the branch decision making process.
- There are a few general rule of thumb
- It should be some instructions that are earlier sequence wise in the original program, compared to the branch instruction. Any instruction that occurs later than the branch instruction in the original program would incidentally be executed or not executed depending on the branch decision aka not independent.
- Its operation should not have effect on status of any registers that would in turn affect the branch decision. e.g. if a BEQ is used, the independent instruction should not affect the Z flag which BEQ instruction used for decision making.
- This is because delay slot instructions gets fully executed only after the branch decision had been made, which means delay slots instruction had no effect on the branch decision.
- If the Branch is part of a loop, independent instructions needs to be sourced from instructions within the loop as delay slot instruction will go through the same number of iterations as well.

Independent Instructions

- Independent instructions refer to instructions which will always be executed in the original program flow, regardless of whether th
- There are a few general rule of thumb
- It should be some instructions that are earlier seque
- Its operation should not have effect on status of any registers that would in turn affect the branch decision. e.g. if a BEQ is used, the independent instruction should not affect the Z flag which BEQ instruction used for decision making.
- This is because delay slot instructions gets fully exec
- If the Branch is part of a loop, independent instructio



Dynamic Branch Prediction

- Hardware circuitry to guess outcome of a conditional branch
  - **Branch history table is implemented to store target addresses of taken branches**
- If prediction is **correct**
  - **Continue normal execution – no wasted cycles**

- If prediction is **incorrect**
  - **Flush instructions that were incorrectly fetched – wasted cycles**
  - **Update prediction bit and target address for future use**

Address of Branch Instruction	Predicted Target address	Prediction Result (T/F)
Address of "BEQ Next"	Next	T

Connecting to the Real World

- ARM Cortex M3/M4 Processor
  - 3-stage pipeline. Instruction Fetch, Instruction Decode and Instruction Execute)
- Branch speculation.
  - When a branch instruction is encountered, the decode stage also includes a speculative instruction fetch that could lead to faster execution.
  - The processor fetches the branch destination instruction during the decode stage itself.
  - During the execute stage, the branch is resolved and it is known which instruction is to be executed next.
  - If the branch is not taken, the next sequential instruction is already available.
  - If the branch is taken, the branch instruction is made available at the same time as the decision is made.

Which pin does USB Host / Hub use to supply power to the USB devices?

A. Data + } Differential signal
B. Data −
C. VBUS → 5 volt ; power Arduino, etc.
D. Ground

4 Pins → Basic USB connectors

connected at its port?

A. Information is pre-fixed before connecting
B. It enumerates the USB device by polling the logic states of the ID pins on the device
✓ C. There is a handshaking process after connection where the USB Host and device will exchange information to let the other party knows of each other's capability and requirement.
D. Host and device exchange ID → not just ID during enumeration to understand each other's capability and requirement

- There are many USB Device Classes, common ones are
  - Human Interface Device (HID) used in Mouse/Keyboard.
  - Communication Device Class (CDC) used to implement Virtual COM Port e.g. Arduino Board, MSP432 Launchpad used in the lab.
  - Mass Storage Class (MSC) used to interface to external USB HDD/SSD.
  - USB Audio Class used to stream audio to USB headset/Microphone.

Why is it that many wireless standards chose to use 2.4GHz as its operating frequency?

A. It was used in the first version of Wifi and developers continued to used that since the characteristics of that frequency range has been well researched and understood.

✓B. The frequency range is free for use world wide.

C. Transmission range of this frequency is one of the best among the various frequency range available for use by consumers.

D. It's the resonance frequency of water, which is available in abundance in nature.

Assuming other factors equal, lower frequency, longer transmission range.
⇒ but still depends on many other factors other than frequency

Which of the following factor does not affect wireless transmission range?

A. Transmission Power

B. Transmission Frequency

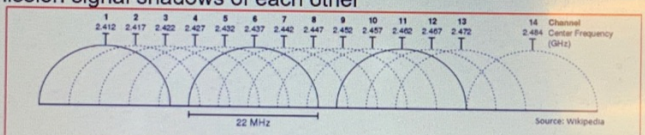
C. Interference from other wireless sources

✓D. Number of repetitions in transmission → affect success rate

- Transmission power
  - Transmission range increase as transmission power increase.
- Transmission frequency
  - Higher frequency signal experience higher attenuation when propagating through the air or other medium.
  - All things equal, higher frequency signal has lower range than lower frequency signal.
- Interference
  - Many commonly adopted standards such as Wifi and Bluetooth works in the same 2.4Ghz ISM band. Their transmission will interfere with each other.
  - The closer the transmitter is to each other, the stronger the interference.
  - Even the micro oven in your kitchen operates in the 2.4Ghz range!

Which of the following would not be able to mitigate wireless interference?

A. Select different Wifi Channels
B. Use frequency hopping
C. Increase Transmission power
✓D. Place various wireless devices near each other so that it'll be in the transmission signal shadows of each other



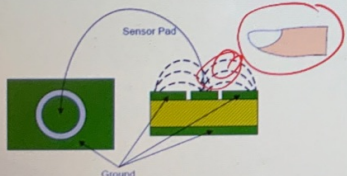
- "2.4Ghz ISM band" really consist of a band of frequencies between 2.4 and 2.5Ghz. Similarly, "5.8Ghz band" also span across a certain frequency range.
- Figure above shows the Wifi standard dividing the given frequency band into sub-bands known as channels.
- One common way to mitigate Wifi interference is to select different channels to use from your neighbours.
- For scenario of different wireless standard such as Bluetooth and Wifi, another common way is to use frequency hopping, i.e. to constantly hop from one channel to another so that transmission will eventually succeed. BT uses frequency hopping.
- There are other methodologies and techniques employed to further mitigate interference between transmitters.

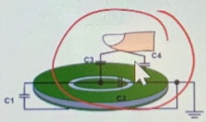
CE/CZ 1006

What is the basic operating principle of the capacitive touch screen?

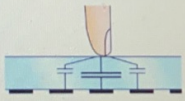
A. Magnetic object interacting with the electric field generated by the screen, affecting the system's capacitance
✓B. Electrical conductive object interacting with the electric field of the system setup, affecting the system's capacitance.
C. Pressure from the object touching the screen changes the capacitance of the system → can't change distance with pressure in this case
D. Obstruction of light from opaque objects affecting the capacitance of the light sensitive setup of the screen.

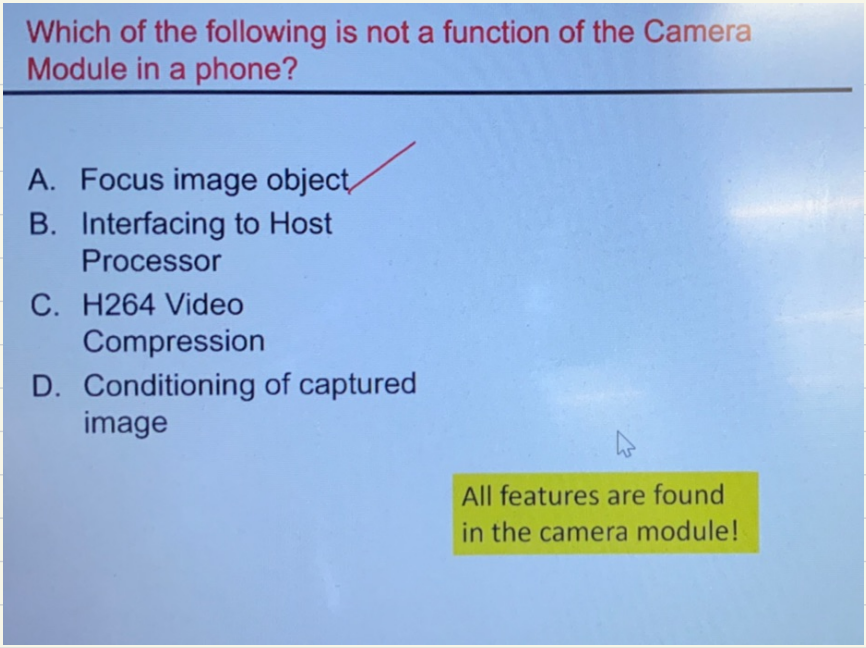
Diagram: resistive touch screen → pressure involved





fingernail conducts electricity




Which of the following is not a function of the Camera Module in a phone?
A. Focus image object
B. Interfacing to Host Processor
C. H264 Video Compression
D. Conditioning of captured image
All features are found in the camera module!