॥ श्री ॥

__/__/____

Subject :- IoT. (Internet of Things).

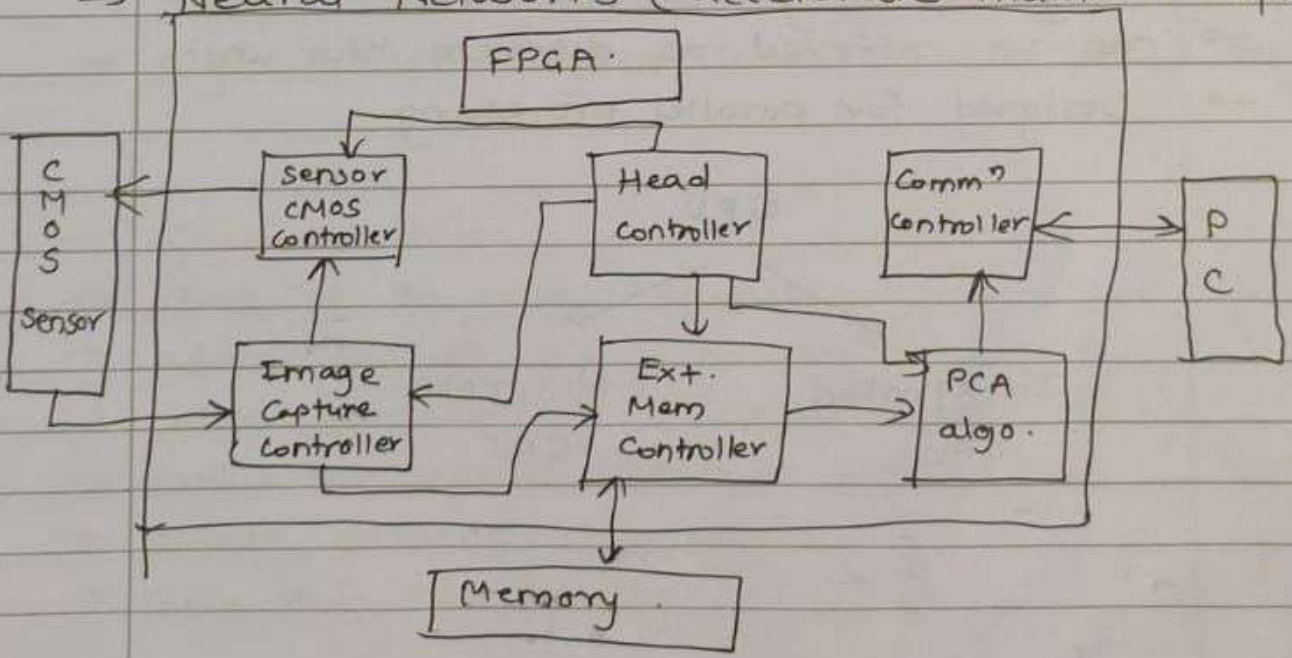
* SoC Products :-
  - ↳ FPGA
  - ↳ GPU
  - ↳ APU
  - ↳ Compute Units.

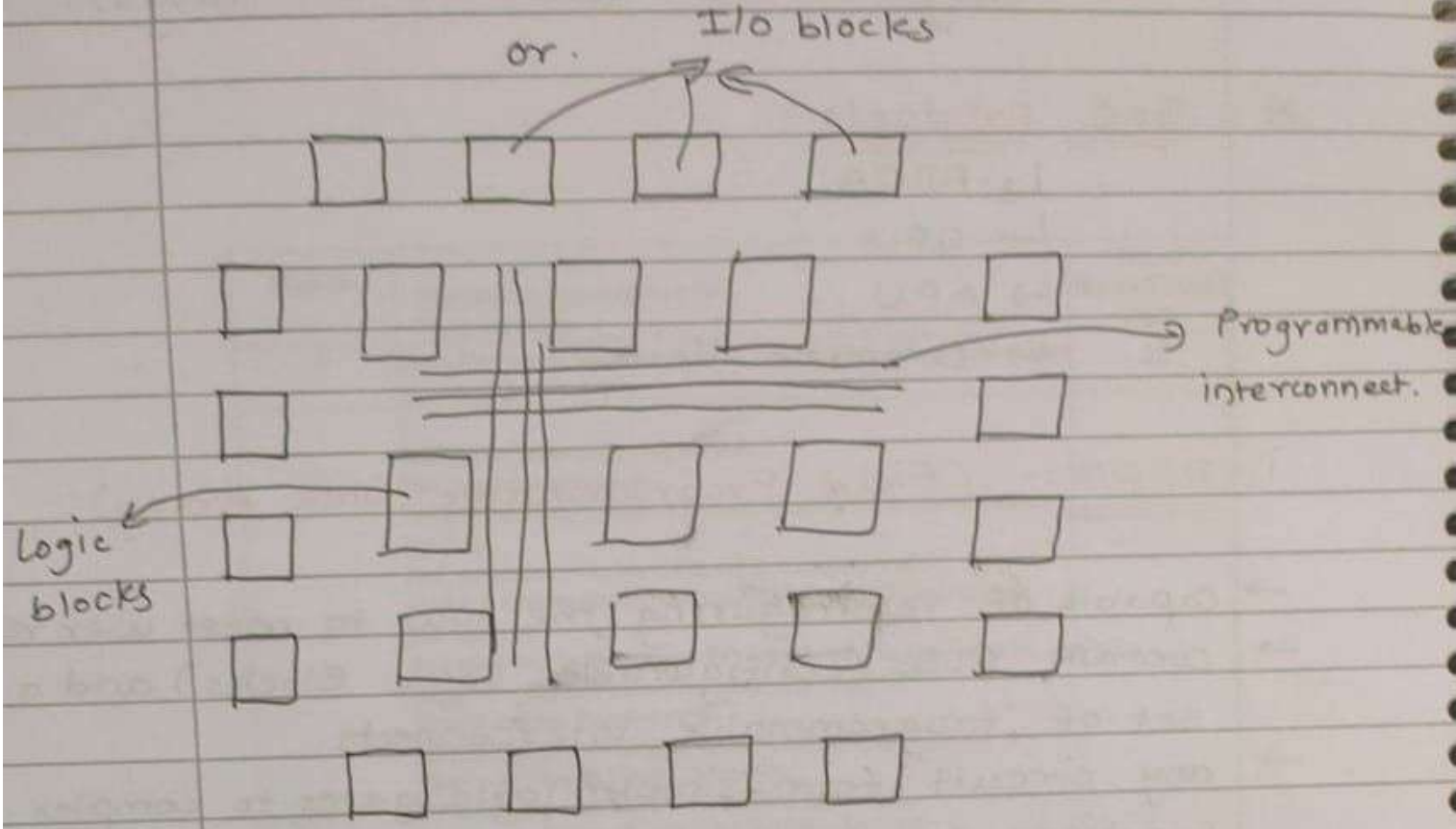
1) FPGA :- (Field Programmable Gate Array) :-

→ Capable of reconfiguring the h/w to meet user req.
→ Contain CLBs (configurable Logic Blocks) and a set of programmable interconnects.
→ any circuit from simple logic gates to complex functions can be implemented.
→ Full SoC containing multiple processors can be put on single FPGA device.

* Features of FPGA :-

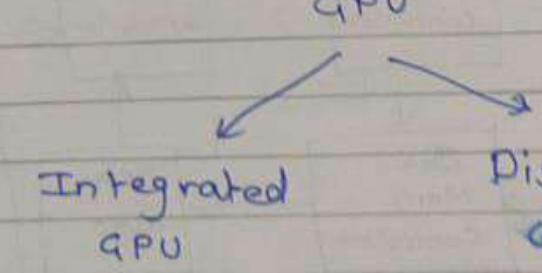
→ Hardware Prototyping (simulation)
→ Hardware acceleration
→ Space Avionics (Update, fix bugs without phy access)
→ Neural Networks (accelerate matrix multiplicat")





2) GPU :- (Graphics Processing Unit):

→ A specialised processor.
→ designed to accelerate graphics rendering
→ can process many pieces of data simultaneously.
→ Useful for ML, Video editing, gaming appln.
→ may be integrated into CPU.
→ can be offered as discrete h/w unit.
→ Designed for parallel processing.



* Uses of GPU:-

→ rendering graphics in 2D and 3D. } for gaming
→ higher resolution, faster frame rates. }

→ virtual streaming and enhance livestream } video editing & content creation.
→ support large displays }

→ high computational capability } for ML.
→ incredible acceleration in workload. }

APU (Accelerated Processing Unit):

→ Comb^n of CPU and GPU on a single chip.
→ can handle both general computing and graphics related task.
→ lower cost.
→ low power consumption.
→ used in laptops to increase Power factor. (form factor)
small space - small power consumption.

* Uses of APU:-

→ Budget gaming PCs
→ e.g. AMD's Ryzen series, Intel core with graphics

Compute Units.

→ similar to host groups.
→ added feature of granularity
→ allow to construct clusterwise structure.
→ useful for comm^n intensive parallel jobs.
→ encode cluster n/w topology.
→ help in ↓ n/w latency.

_/_/___

* Uses of compute units:

→ for the queue.

→ for the host on which jobs from queue can be run.

Differentiate between GPU and FPGA.

	GPU	FPGA.
No. of cores	High	High.
Serial/Parallel.	Parallel	Parallel
Clk freq.	High	Low
Dev. time	Avg.	Long.
Power eff.	Low	High.
Portability	Less challenging	challenging.

ARM8 Architechture:

* Overview of ARM8
  - ↳ Part of ARM family
    - ↙ ↓ ↘
    - Adv. RISC Machines.
  - ↳ 32 bit microprocessor.
  - ↳ Very low power consumption.
  - ↳ high performance.
  - ↳ Reduced Inst^n set Computer (RISC) Arch.
  - ↳ simple inst^n mechanism.
  - ↳ fully static CMOS implementation. (↑ clock speed)
  - ↳ Inclusion of branch predicting prefetch unit.
  - ↳ user optimizable.
  - ↳ Provides fast on-chip memory.
  - ↳ 32 bit address bus.

ARM-8 Instruction Set:-

→ 11 types of basic instructions.
→ Data operation inst^n (2)
→ Data transfer and control (3)
→ flow, privilege, system control inst^n (3)
→ co-processor control inst^n (3)

* Adv. of ARM8 inst^n set:-

→ straightforward to use.
→ good for compilers.
→ employs pretetch buffer, inst^n and data pipelining.

Multitasking when Inst^n 3 is executing

Fetch	Decode	Execute
		Inst^n 1
	Inst. 2	
Inst^n 1.		

read/
wr.

ARM-8 Architechture:-

→ consists of a core and a Prefetch Unit.
→ Mem. (on-chip) can provide two words of data/ Inst^n on each cycle.

* ARM 8 Prefetch Unit:-

→ prefetches and buffers instruction.
→ makes use of extra bandwidth.
→ decides if the branch will be taken or not.
→ If a branch is predicted, taken then its dest^n address is calculated.
→ further inst^n are fetched from there.
→ some branches can not be predicted and in that case, prefetch unit becomes empty.

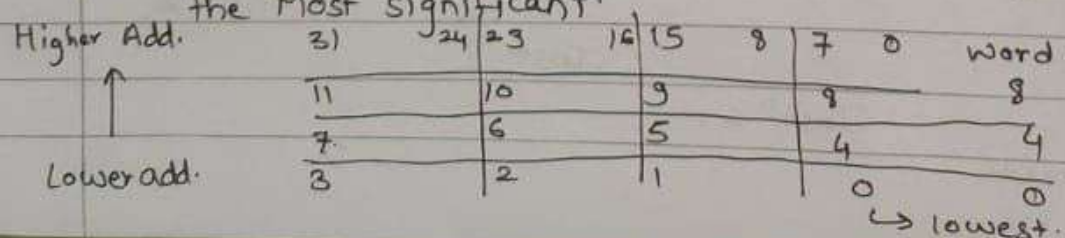
* ARM8 core:-

→ extension of data pipeline
→ 5 stage pipeline.
→ adder and shifter operate in parallel.
→ bigger multiplier operates on 8 bits per cycle.

** Programmer's Model.

1) Hardware configurations
   ↳ Big & Little Endian Memory.

Little Endian format:-
→ The lowest numbered byte in a word is considered LSB of the word and highest no. byte the Most significant.



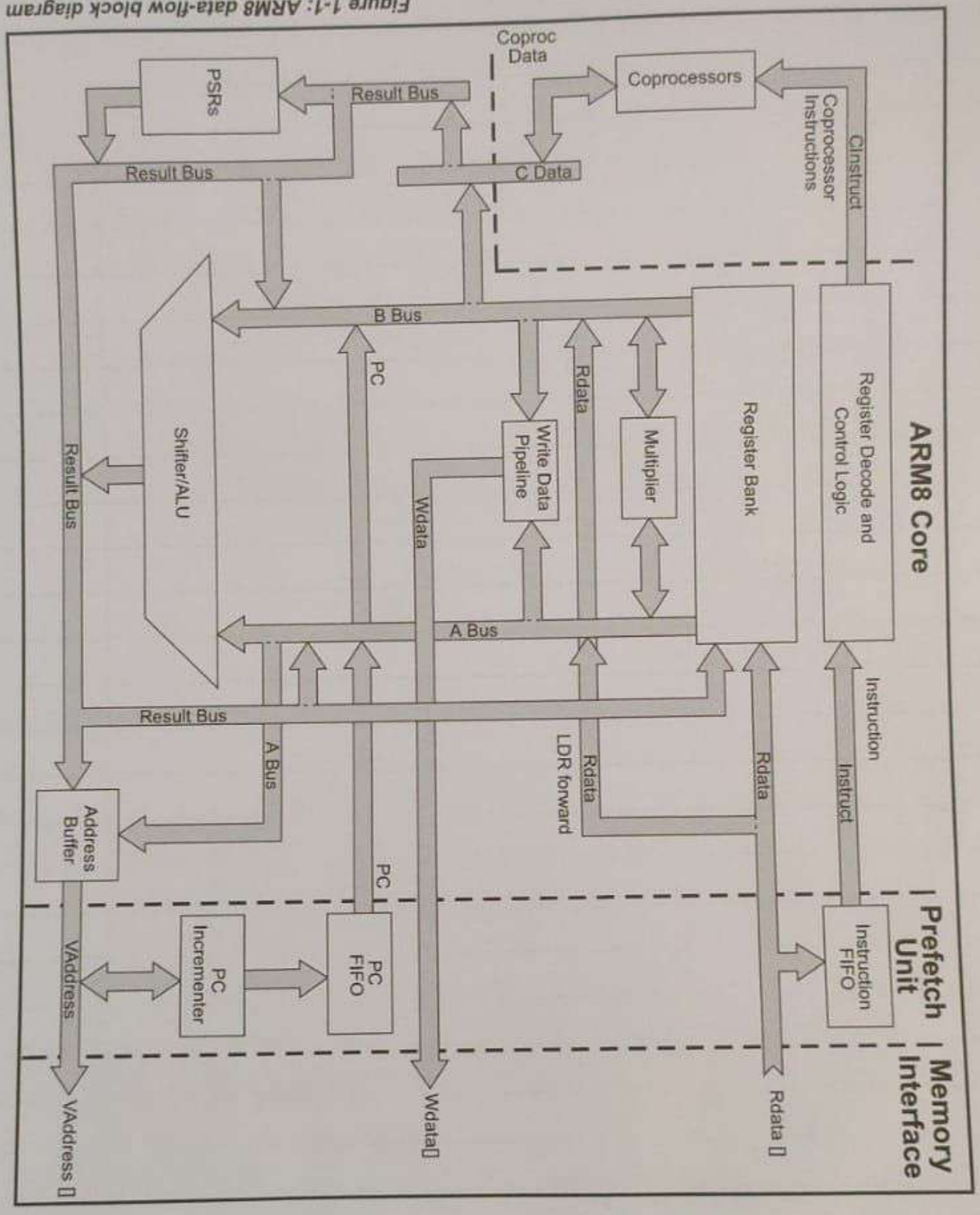
Big-endian format:-

- In this format, most significant byte is stored

8	9	10	11
4	5	6	7
0	1	2	3

MSB at lowest add.

Figure 1-1: ARM8 data-flow block diagram



ARM 8 consists of following blocks:

1) ARM 8 core
   - Reg. decode & control logic
   - Reg. Bank
   - Multiplier
   - Write data pipeline.
   - Shifter / ALU.
   - PSRs
   - coprocessors.
   - Address buffer.
2) Prefetch Unit
   - Instruction FIFO
   - PC FIFO
   - PC Incrementer.

* Operating modes:-
  → Supports byte (8 bits), half word (16 bit) and word (32 bits) data types.

7 modes of operation:-

	Mode	Description.
1)	User Mode (Usr)	normal program execution state.
2)	FIQ mode	used for fast (high priority) interrupt handling.
3)	IRQ mode	used for gen. purpose intr. reg. handling.
4)	Supervisor mode	A protected mode for O.S.
5)	System Mode	privileged user mode for O.S.
6)	Abort Mode	After data / inst^n prefetch Abort. entered when an undef. inst^n is executed.
7)	Undefined	

Intro. to Raspberry Pi.

high performance, low cost micro controller.

key features :-

1) Dual ARM Cortex-M0
2) 264 KB of on-chip SRAM in 6 banks.
3) 16 MB off chip mem. support.
4) DMA controller.
5) Fully connected AHB crossbar. (Adv. High perf. Bus)
6) Interpolator and integer divider peripherals.
7) On-chip programmable Low Drop Out (LDO) reg. to generate core voltage.
8) 2 on-chip PLLs.
9) 30 GPIO pins (4 can be used as analog i/p)
10) Peripherals
    - → 2 UARTs. (Universal Asynch. Receiver transmitter).
    - → 2 SPI controllers.
    - → 2 I2C controllers.
    - → 16 PWM channels.
    - → 8 PIO state machines. (Prog. I/O).

Raspberry-Pi Hardware :-

→ Internal SRAM for code/data storage.
  264 KB → 6 banks.
→ DMA controller for data transfer op^n.
→ GPIO pins with various logic fun^n can be driven directly.
→ dedicated hardware for specific functions. (SPI, I2C, UART)
→ PIO controller for various I/O functions.
→ USB controller.
→ 4 ADC i/Ps.
→ 2 PLLs
→ An internal voltage reg.

__/__/____

* Prepare Rasp- Pi for programming :-

1) Install the O.S. (Raspbian with desktop).
2) Configure Pi. (basic steps on terminal).
3) Run the config tool.
4)
    1) change User Pwd.
    2) Network options
    3) Boot options
    4) Localization options
    5) Interfacing options
    6) Overclock
    7) Advanced options
    8) Update
    9) About R-pi.
4) Use Python 3.
5) Install PIP
6) Write program.
7) Connect hardware as per requirement.
8) Run the program.

* Intro to Node.js :-

↳ open source, runtime javascript environment
↳ app runs in a single process
↳ serverside platform

Features of node.js :-
→ Asynchronous and event driven
→ Very fast.
→ Single threaded but highly scalable.
→ No buffering
→ Licence.

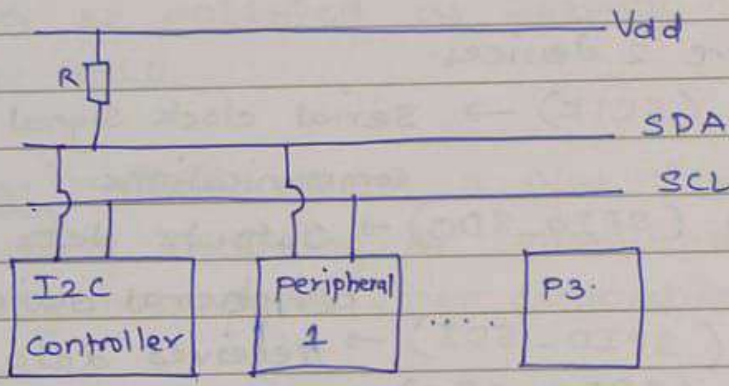
Raspberry-Pi Interfaces:-

1) UART :- (Universal Asynchronous Receiver/Tx).
- Used as serial console.
- UART Tx - GPIO 14
  UART Rx - GPIO 15

→ R-Pi has 2 UARTs.

2) GPIO :-

3) I2C :-
- It is a protocol for communicating with low speed peripherals.



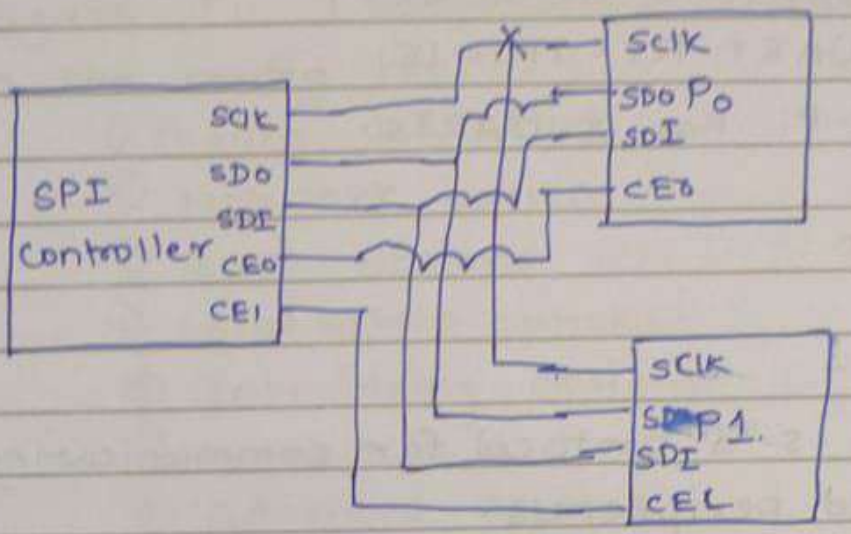
→ R-Pi may have 1 or 2 I2C buses.
- each bus has an I2C central connected to 2 bidirectional lines.
  SDA → Serial data line. (SDA)
  SCL → Serial data clock.

→ These lines are connected to GPIO pins.
→ It is possible to connect multiple I2C devices (ADC, LCD, Sensors etc). to I2C.
→ each device on I2C must have unique address.
→ R-Pi supports 7 bit add.
→ 128 unique devices can be connected.

__/__/____

SPI:-

Serial Peripheral Interface (SPI) is a full duplex serial protocol for communicating with high-speed peripherals.



Can drive 2 devices.

1) GPIO 11 (SCLK) → Serial clock signal to synchronize communications.
2) GPIO 10 (SPIO_SDO) → Outputs data to SPI peripheral device.
3) GPIO 9 (SPIO_SDI) → receives data from SPI peri
4) GPIO 8 (SPIO_CE0) → enables first SPI dev.
5) GPIO 7 (SPIO_CE1) → enables other SPI dev.

SPI on h/w R-Pi supports:-

1) Mode 0, 1, 2, 3
2) 8 bits per word.
3) Data speeds (Hz) → 0.5 M, 1M, 2M, 4M, 8M, 16M, 32M.

Mode	CPOL	CPHA	
0	0	0	↓ sclk, active $\overline{CS}$
1	0	1	↑ sclk
2	1	0	↑ sclk, active $\overline{CS}$
3	1	1	↓ sclk.