Internet of Things
Autumn 2021-22

Prof. Suchismita Chinara

Dept. of Computer Science Engg.

National Institute of Technology Rourkela-769008

Email: suchismita@nitrkl.ac.in

Books:

- Internet of Things by S K Vasudevan, A S Nagarajan, and RMD Sundaram, Wiley
- Internet of Things :Principles and Paradigms by Rajkumar Buyaa and Amir Vahid Dastjerdi, Elsevier.
- Internet of Things: A Hands-on approach by A Bahga and V Madisetti, University press.
- The Internet of Things: Enabling Technologies, Platforms, and Use cases by P Raj and A C Raman, CRC Press.

Outline

- 1. Introduction to IoT
- 2. Introduction to Sensors, Microcontrollers, and their Interfacing
- 3. Protocols for IoT- Messaging and Transport
- 4. Protocols for IoT- Addressing and Identification
- 5. Data Analytics – Visualizing the Power of Data from IoT

Introduction to IoT

Definition of IoT

- IoT refers to the interconnection via the Internet of computing devices embedded in everyday objects, enabling them to send and receive data.
- IoT can also be defined as the analysis of data to generate a meaningful action.
- The scope of IoT is not just limited to getting the devices connected or networked, but its more about the exchange of meaningful information from one device to another device for the accurate interpretation of raw data.
- IoT is not a single technology, but a combination of technologies and domain knowledge. So IoT is not owned by one engineering branch. It is a reality when multiple domains come together.

Year 2020

- Things like refrigerators, kettles, water heaters, home equipment, and other electronic devices will be connected to the internet and there would be 15 billion device connections. About 500 million wearables, watches, shoes, etc. and 100 millions of implants in human / animals would talk to internet.
- Father of IoT: kevin Ashton

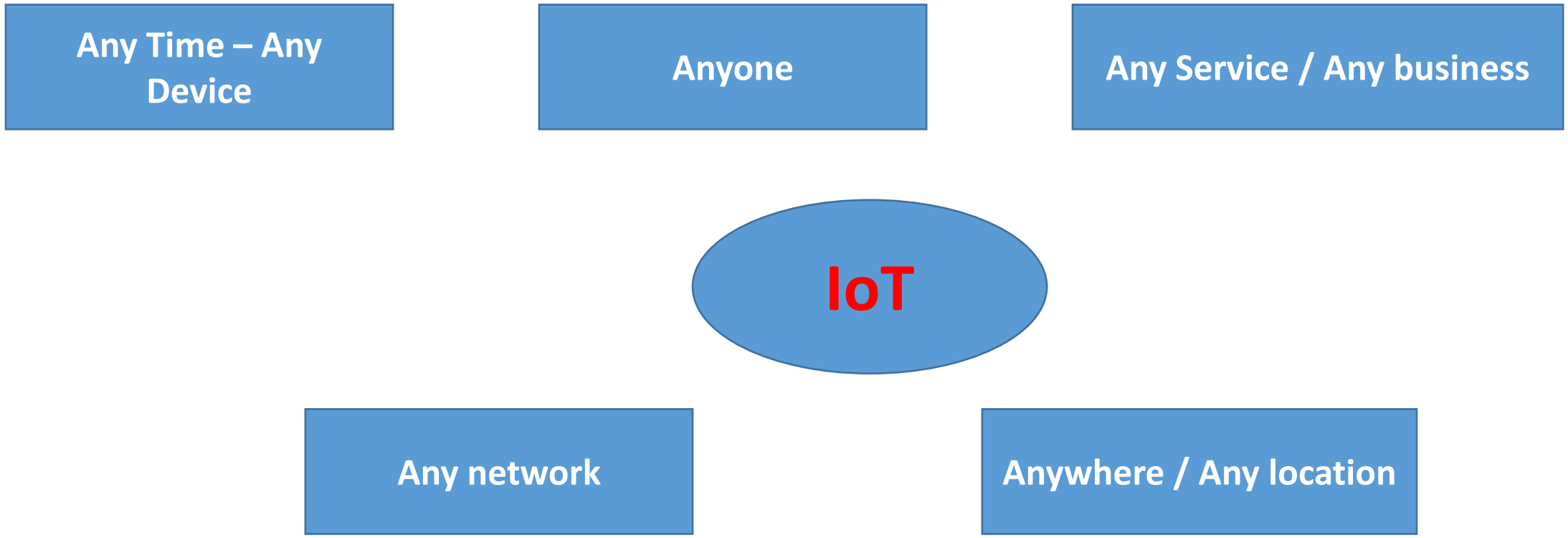
Life in an IoT ecosystem

- When you wakeup, the water heater would have hot water ready
- During your bath, the microwave would have cooked your food.
- When you lock the door of your house, the car doors would open for you.
- Your AC would adjust based on your body temperature/ room temperature
- The pill box would you at the time of your medicine intake.
- Your refrigerator would order milk / egg on need
- While nearing your house, the house lights would turn ON with appropriate brightness.

Pillars of IoT

- Based of two Important pillars: Internet & Things
- Internet (well known term)
  - Application need to support a diverse set of devices for data analysis &knowledge extraction
  - Communication protocols
  - Integration of mobile devices, edge devices, and humans as controllers
- Things
  - Smart devices / Sensors / Human beings
  - Any device that is able to communicate with other entities making it accessible at any time and anywhere.

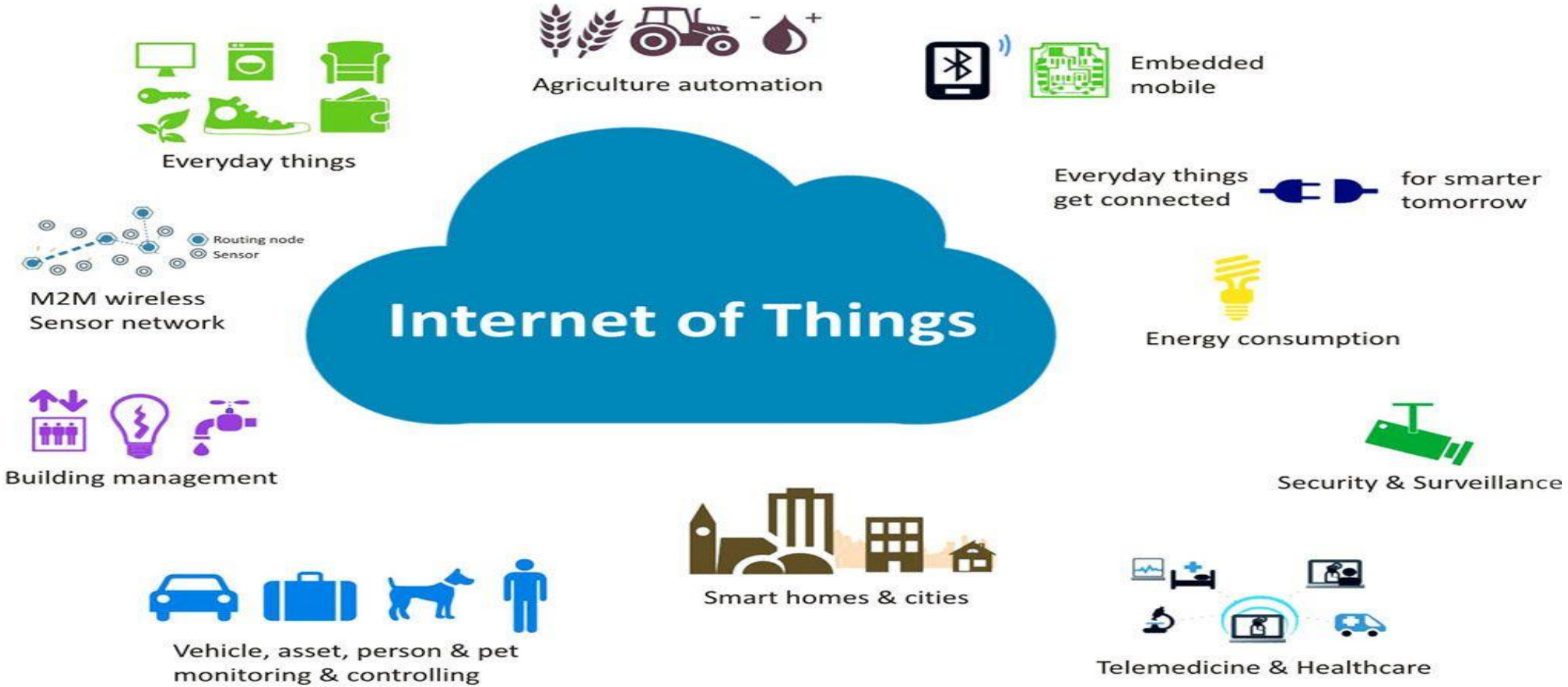
IoT Design goals



Why IoT ?

- Want to automate
- Want to control everything
- Want to receive more data
- Want to make things faster

Application Area of IoT



Characteristics of IoT

1. Connectivity
2. Intelligence and identity
3. Scalability
4. Dynamic and self-adapting
5. Architecture
6. safety

Connectivity

- One of the most important requirement of IoT infrastructure
- THINGS should be connected to IoT infrastructure (wireless connection)
- Connection should be guaranteed anytime, anywhere, with anyone
- No connectivity – No IoT exists

Intelligence & identity

- The extraction of knowledge from the generated data is more important.
- Ex: the sensor generated data will be useful only when it is interpreted properly. Each IoT device has an unique identifier called IP address.
- The identification is helpful in tracking the object.

Scalability

- The number of devices connected to IoT eco system is increasing day by day.
- Any IoT setup should be capable of handling the massive growth in number.
- The data generated is also enormous.

Dynamic and self adapting

- IoT devices should dynamically adapt themselves to the changing contexts or scenarios.
- Ex: a camera working in an IoT environment should be adaptable to work in different condition and different intensity of light.

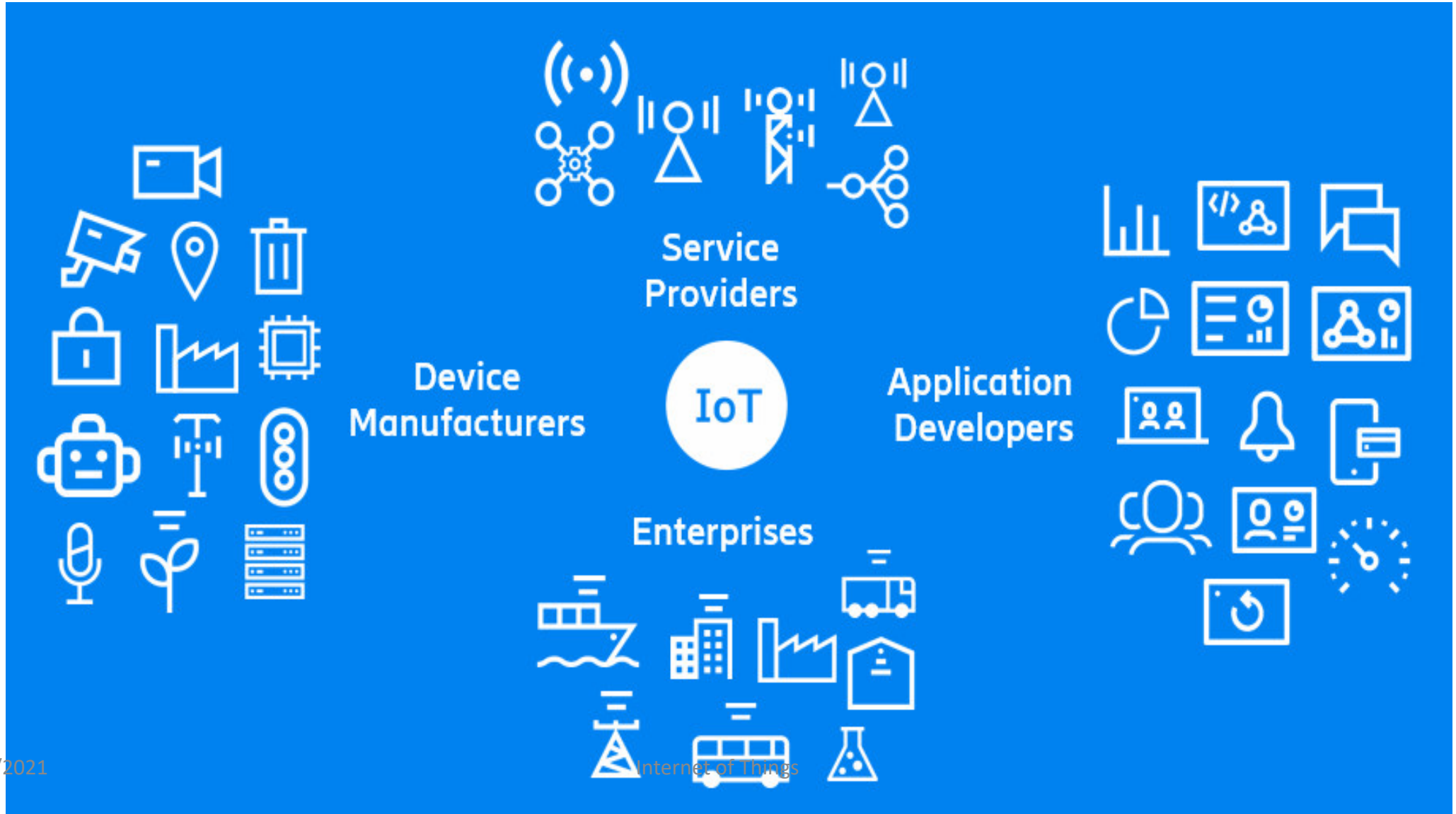
Architecture

- IoT architecture can not be homogeneous in nature.
- It should by hybrid, supporting different manufacturer’s product to function in the IoT network.

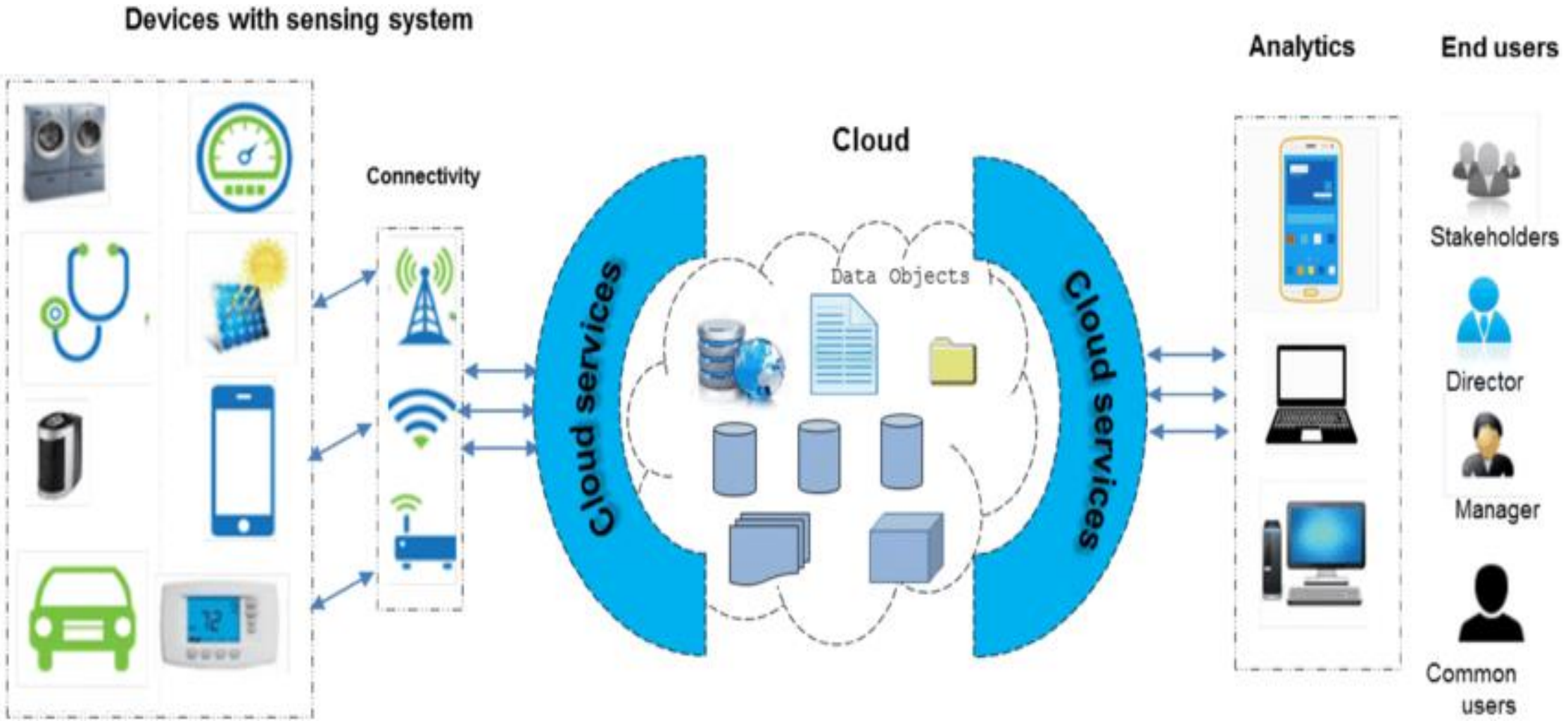
safety

- There is a threat of sensitive personal details of an user getting compromised when all his devices are connected to the internet.
- This could cause a loss to the user.
- The equipment safety is also critical.

Example IoT Ecosystem



IoT Ecosystem



Internet of Things
Lect#2

Prof. Suchismita Chinara

Dept. of Computer Science Engg.

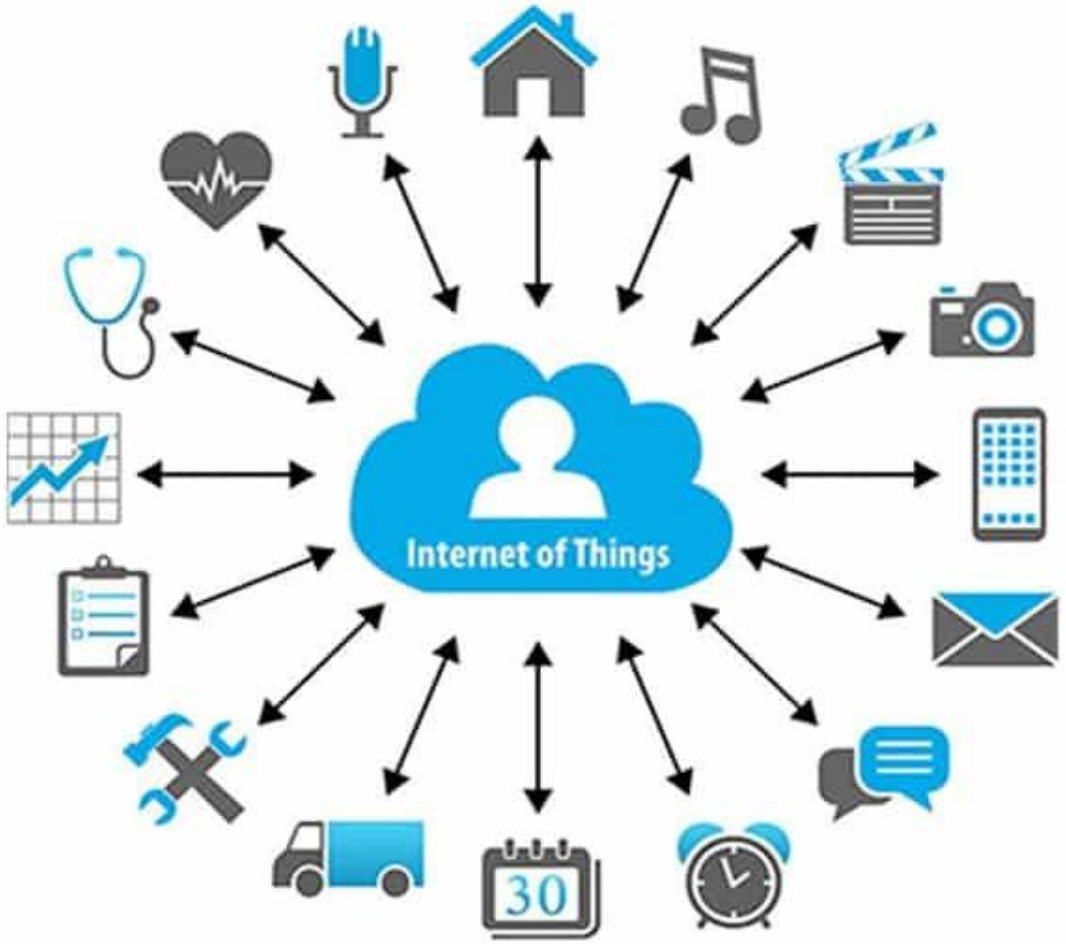
National Institute of Technology Rourkela-769008

THINGS of IoT

- The “thing” in an IoT system is something that has an identity of existence. (IP address)
- The thing can measure / monitor / exchange data
- Example of THINGS are – temp sensor, pressure sensor, human / server / vehicles/ shoes/ wearables/ pacemaker/ food quality measuring/ biochip transponders for animals in farms etc.

- “THINGS” = HARDWARE + SOFTWARE +SERVICE

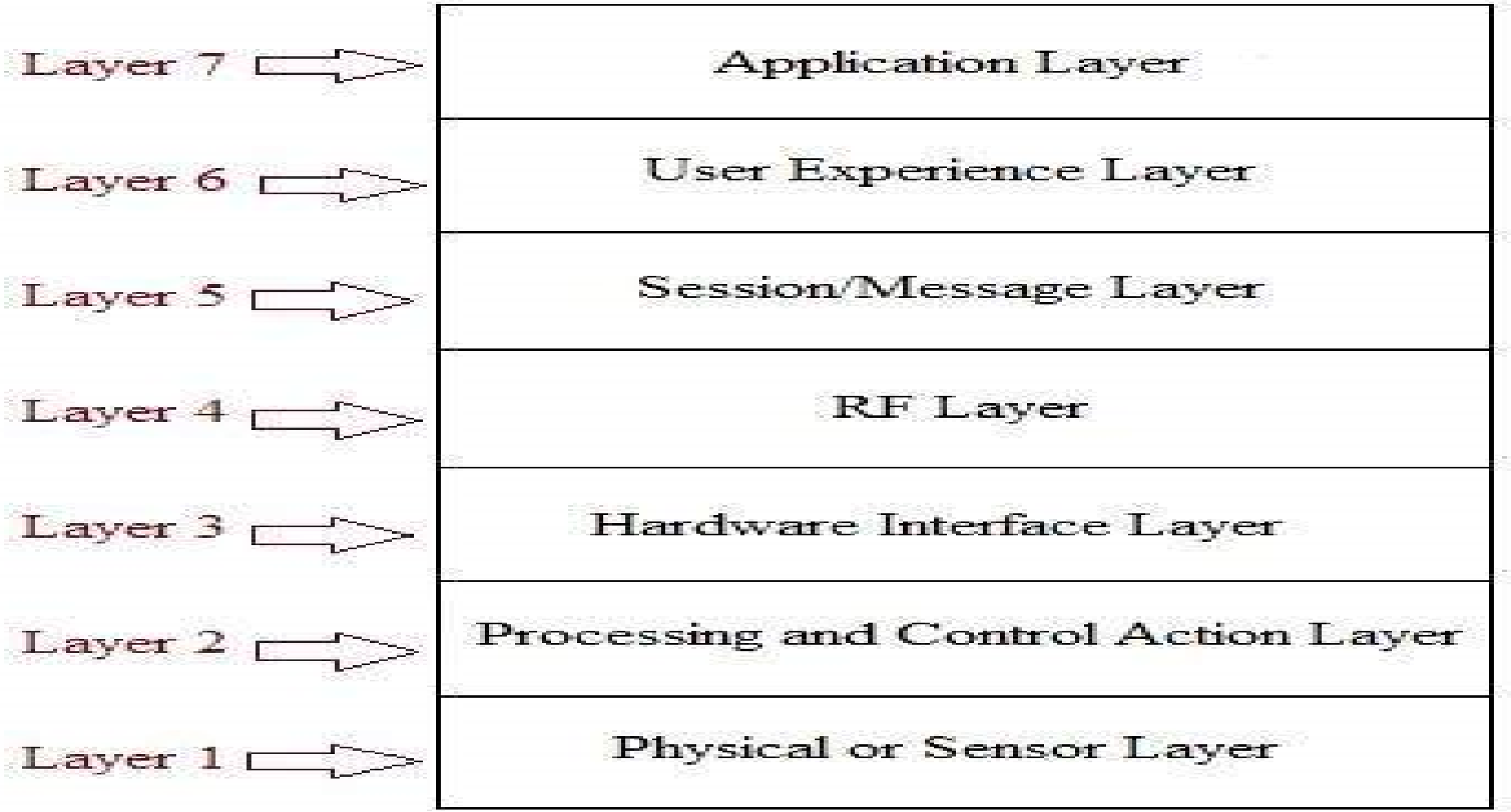
Example: THINGS



What could be the possible things for home automation

- Alexa
- Echo
- Smart bulb
- Camera
- Controllers
- Refrigerator
- TV
- Microwave
- Washing machine

IoT Protocol Stack Layers

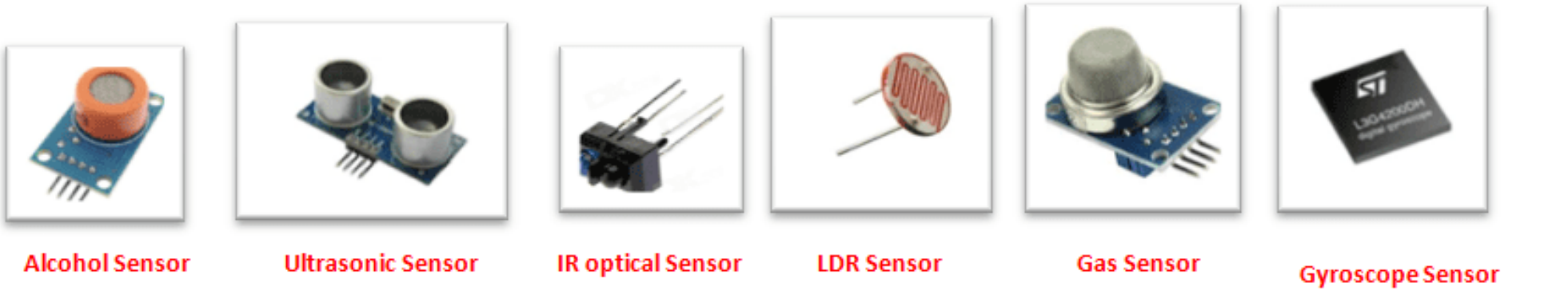


IoT Protocol Stack Layers

Layer 1: Physical or Sensor Layer

- This layer is concerned about the physical components, mainly includes sensors.
- This layer is responsible for data collection (mostly sensing).
- Choosing the appropriate sensor is the major challenge, as many sensors are available that can do the same task but at different cost.
- Ex: temp sensor, humidity sensor etc. are major components for this layer. For industrial automation (IIoT) PLC, actuators are the major components of layer 1.

Example of Layer 1 devices



Layer 2 : Processing and Control action layer

- This layer comprises of core components like microcontrollers or processors.
- The data is received by the microcontrollers from the sensors.
- Example microcontrollers are: Arduino, NodeMCU, PIC , ARM etc.
- Operating systems, Android, IOS, Linux play a major role to execute the task.
- The data collected from the sensors is processed in this layer.
- To determine if the data is meaningful a microcontroller should be present.

Examples

- Microcontrollers:
  - PIC / ARM/ INTEL boards
- DEV kits:
  - Arduino, Rasberry
- OS:
  - RTOS, Linux, Android, IoSR

Arduino Uno

Raspberry pi

Layer 3: Hardware interface layer

- This layer include components or interfaces used for communication such as RS232, RS485, SPI, I2C, CAN, SCI etc.
- These interfaces are used for serial or parallel communication at various baud rates in synchronous/asynchronous modes.
- The above mentioned interface protocols ensure flawless communication.

HW Interface Layer

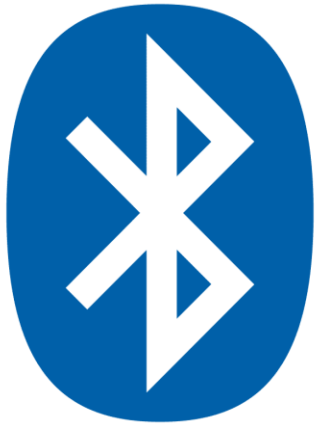
- Components for communication:
- Serial / Parallel standards, RS 232, USB, I^2C, SPI, CAN, routers etc.
- Layer responsible for handshake

Layer 4: RF Layer

- This radio frequency layer houses RF technologies based on short range or long range and data rate desired by the application of use.
- The common indoor RF/wireless technologies include Wifi, Bluetooth, Zigbee, Zwave, NFC, RFID etc.
- The common outdoor RF cellular technologies include GSM/GPRS, CDMA, LTE-M, NB-IoT, 5G etc.
- RF layer does communication of data using radio frequency based EM waves.
- There is another technology which uses light waves for data communication.
- This light based data communication is referred as LiFi.

Example: RF Layer

Short Range: Low bandwidth, High bandwidth
Long Range: Low bandwidth, High bandwidth



BLE

Wi-Fi

Layer 5: Session / Message Layer

- This layer deals with various messaging protocols such as MQTT, CoAP, HTTP, FTP (or Secured FTP), SSH etc.
- It defines how messages are broadcasted to the cloud.

- Example Protocols: MQTT, CoAP, HTTP, FTP, SSH
- Content: Messaging

Layer 5: User experience layer

- This layer deals with providing best experience to the end users of IoT products.
- To fulfill this, this layer takes care of rich UI designs with lots of features.
- Various languages and tools are developed for the design of GUI interface software.
- These include object oriented and procedure oriented technologies as well database languages (DBMS, SQL) in addition to analytics tools.
- Technologies: Object oriented, Procedure Oriented
- DBMS , SQL: Analytics Tools / Software from vendors

Layer 7: Application Layer

- This layer talks about the possible applications that can be built with the support of the rest of the layers.
- It can range from a simple automation application to smart city application.
- Example: Smart home, smart city, smart parking, smart energy, smart retail etc.

Internet of Things: Enabling Technologies Lect#3

Prof. Suchismita Chinara

Dept. of Computer Science Engg.

National Institute of Technology Rourkela-769008

IoT Enabling Technologies

- Technologies that help in acquiring / sensing data
- Technologies that help in analysing / processing data
- Technologies that help in taking control action
- Technologies that help in enhancing security / privacy

Sensors

- Sensors sense the environment and retrieve data.
- They are the starting point in any IoT application
- Ex: for a temperature monitoring application, the temperature sensor fetches data for us to operate on.
- Sensors could be analog or digital.
- Digital Sensor: LED, Push button switches, Alarm etc.
- Analog Sensor: Pressure, Temperature etc.

Sensors

- A sensor converts the physical action to be measured into an electrical equivalent and processes it so that the electrical signals can be easily sent and further processed.
- A sensor consists of three main components:
  - **(1)** The sensing section contains the sensor itself which is based on a particular technology. The variety of technologies means you can select a sensor technology which fits your application.
  - **(2)** The processing circuitry converts the physical variable into an electrical variable.
  - **(3)** The signal output contains the electronics connected to a control system.

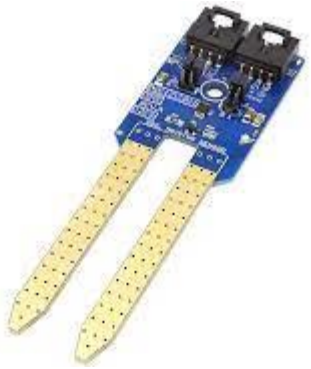
Sensors

- Example of sensors that are regarded as enabling technologies:
  - Camera used in security system
  - Temp/humidity/moisture sensors used for weather monitoring
  - Vehicle health monitoring sensors to keep track of speed / tyre pressure etc
  - OBDs (ELM327) used for collecting all critical information from an automobile to detect anomalies
  - Vibration sensors to track the quality of structures (building / bridge etc.)
  - Water quality monitoring sensors to measure PH, turbidity, chloride level etc.
  - PIR sensor to sense the presence of a human or any object (Ex: pedestrian signal operation)

Sensors

Vibration sensor

Temp sensor



Moisture sensor

Camera

OBD: ELM327

PIR sensor

PH sensor

Pressure, Temp, humidity sensor

Cloud Computing

- Cloud computing is a significant technology in IoT.
- As data storage plays a major role in IoT, cloud has grown much more popular as it serves as an affordable, effective and efficient medium for data storage.
- Different popular cloud services are:
  - IaaS (Infrastructure as a Service)
  - PaaS (Platform as a Service)
  - SaaS (Software as a Service)

Cloud Computing Services (IaaS)

- IaaS (Infrastructure as a Service)
  - This service replaces physical machines by virtual machines. Meaning, it provides virtualized computing resources over the internet.
  - The users manage the machines, select the OS and underlying applications, and pay as per their use.
  - In short, IaaS is a type of cloud computing service that offers essential compute, storage and networking resources on demand, on a pay-as-you-go basis.
  - Example IaaS are:
    - Microsoft Azure
    - Amazon Web Service (AWS)
    - Cisco Metacloud

Cloud Computing Services(PaaS)

- PaaS (Platform as a Service)
  - In this model, the cloud service provider delivers hardware and software tools needed for application development to users over the internet.
  - A PaaS provider hosts the hardware and software on its own infrastructure.
  - Users have to build, manage and maintain the applications as per their requirement.
- Computing platforms which typically includes operating system, programming language execution environment, database, web server etc.
- Example PaaS are:
  - AWS Elastic Beanstalk, Windows Azure, Google App Engine, Apache Stratos.

Cloud Computing Services(SaaS)

- SaaS (Software as a Service)
  - In this model, a complete software application is provided to the user.
  - It can also be called as application as a service.
- In short, software distribution model in which a cloud provider hosts applications and makes them available to end users over the internet.
- In this web-based model, software vendors host and maintain the servers, databases, and the code that makes up an application.

Big Data Analytics

- The biggest challenge with big data is its volume, variety, speed (velocity) at which it comes and its veracity. (4Vs of big data)
- Big data is majorly governed by :
  - Scale (Volume)
    - Storage has become inexpensive and hence, cost-related challenges have reduced. Cloud storage and hardware storage both have become affordable because of the tremendous growth in the semiconductor industry.
  - Complexity (Varity)
    - Data comes from different formats(audio / video/ text/ image) and has to be interpreted systematically.
  - Speed (Velocity)
    - It is the rate at which the new data is generated / changed. All the data pours in at a very high speed, which makes it very challenging to not miss and oversee the actual data from the noise.
  - Data in doubt (Veracity)
    - It says how accurate is the data? The data's nature changes dynamically and ambiguity is often seen (incomplete data).

Embedded Computing Boards

- It brings IoT design to reality.
- Most of the computing boards available in market are driven by microcontrollers or processors.
- Example computing boards are:
  - Raspberry Pi
  - Arduino
  - Node MCU

Example computing boards

Node MCU

Arduino Uno

Raspberry Pi 4

Communication Protocols

- Data exchange is possible because of the protocols.
- Protocols take care of the following:
  - Addressing
  - Format of the message
  - Message security (encryption & decryption)
  - Routing
  - Flow control
  - Error monitoring
  - Sequencing
  - Retransmission guidelines
  - Segmentation of the data packet

User Interfaces

- UIs are designed for accessing and handling the services are easier and comfortable for the end user.

IoT challenges

- Security / Personnel safety
- Privacy
- Data extraction with consistency from complex environments
- Connectivity
- Power requirements
- Complexity involved
- Storage

IoT Architecture

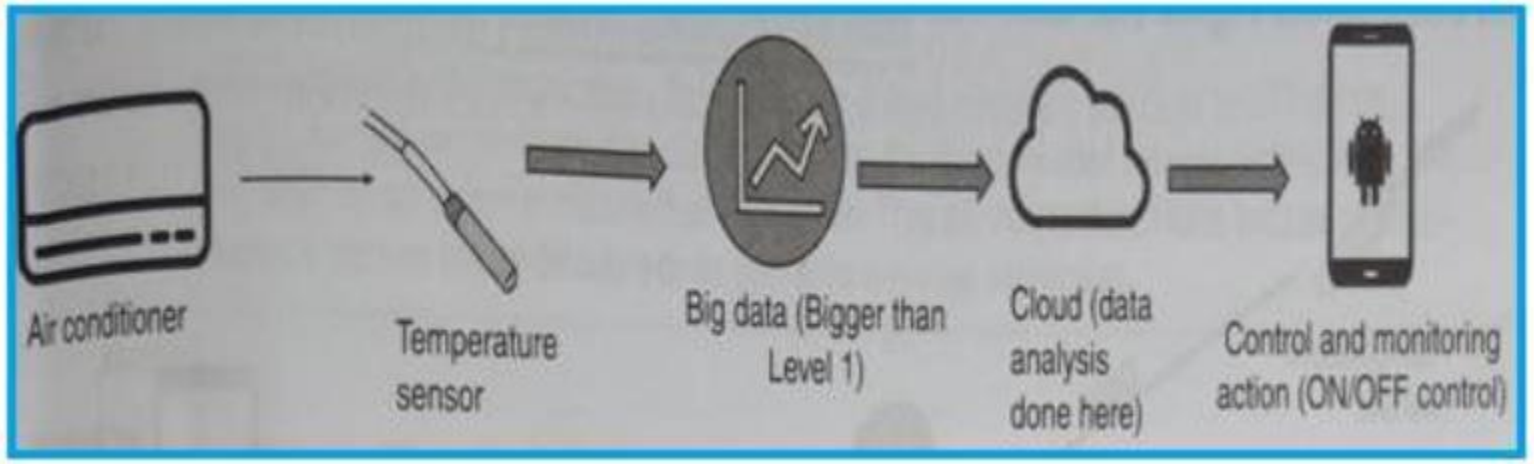
- IoT architecture elements vary based on applications of use.
- Based on this fact, various levels are defined for IoT system.
- Let us take example of air conditioner whose temperature has to be monitored to understand IoT levels.

IoT Level 1

- This level consists of air conditioner, temperature sensor, data collection and analysis and control & monitoring app.
  - The data sensed in stored locally.
  - The data analysis is done locally.
  - Monitoring & Control is done using Mobile app or web app.
  - The data generated in this level application is not huge.
  - All the control actions are performed through internet.
- **Example:** Room temperature is monitored using temperature sensor and data is stored/analysed locally. Based on analysis made, control action is triggered using mobile app or it can just help in status monitoring.

IoT Level 2

- This level consists of air conditioner, temperature sensor, Big data (Bigger than level -1, data analysis done here) , cloud and control & monitoring app.
  - This level-2 is complex compare to level-1. Moreover rate of sensing is faster compare to level-1.
  - This level has voluminous size of data. Hence cloud storage is used.
  - Data analysis is carried out locally. Cloud is used for only storage purpose.
  - Based on data analysis, control action is triggered using web app or mobile app.
  - **Examples:** Agriculture applications, room freshening solutions based on odour sensors etc.


Air conditioner
Temperature sensor
Big data (Bigger than Level 1)
Cloud (data analysis done here)
Control and monitoring action (ON/OFF control)

IoT Level 3

- As shown in the figure, this level consists of air conditioner, temperature sensor, big data collection (Bigger than level-1) , cloud (for data analysis) and control & monitoring app.
  - Data here is voluminous i.e. big data. Frequency of data sensing is fast and collected sensed data is stored on cloud as it is big.
  - Data analysis is done on the cloud side and based on analysis control action is triggered using mobile app or web app.
  - **Examples:** Agriculture applications, room freshening solutions based on odour sensors etc.

IoT Level 4

- This level consists of multiple sensors, data collection and analysis and control & monitoring app.
  - At this level-4, multiple sensors are used which are independent of the others.
  - The data collected using these sensors are uploaded to the cloud separately. The cloud storage is used in this level due to requirement of huge data storage.
  - The data analysis is performed on the cloud and based on which control action is triggered either using web app or mobile app.

IoT Level 5

- This level consists of multiple sensors, coordinator node, data collection and analysis and control & monitoring app.
  - This level is similar to level-4 which also has huge data and hence they are sensed using multiple sensors at much faster rate and simultaneously.
  - The data collection and data analysis is performed at the cloud level.
  - Based on analysis,control action is performed using mobile app or web app.

Internet of Things: Introduction to Sensors, Microcontrollers, and their Interfacing
Lect #4

Prof. Suchismita Chinara

Dept. of Computer Science Engg.

National Institute of Technology Rourkela-769008

Monitor the Heart Rate using Pulse Sensor and Arduino

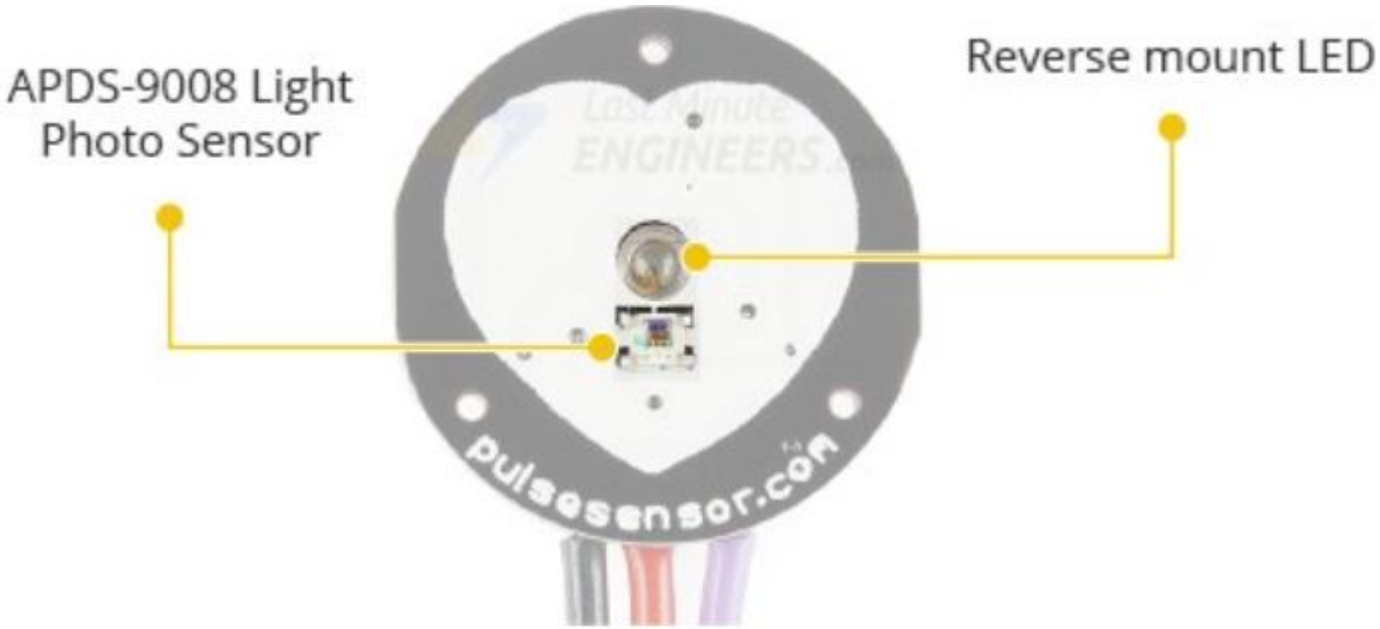
- The Pulse Sensor is a well-designed low-power plug-and-play heart-rate sensor for the Arduino.
- This sensor plugs right into Arduino and easily clips onto a fingertip or earlobe.
- It is super small (button-shaped) with holes,

so it can be sewn into fabric.

Hardware overview of the pulse sensor

- The front of the sensor is the side with the heart logo. This is where you place your finger. On the front side you will see a small round hole, from where the reverse mounted green LED shines.
- Just below the LED is a small ambient light photo sensor – APDS-9008 from Avago, similar to that used in cell phones, tablets and laptops, to adjust the screen brightness in different light conditions.

Hardware overview of the pulse sensor

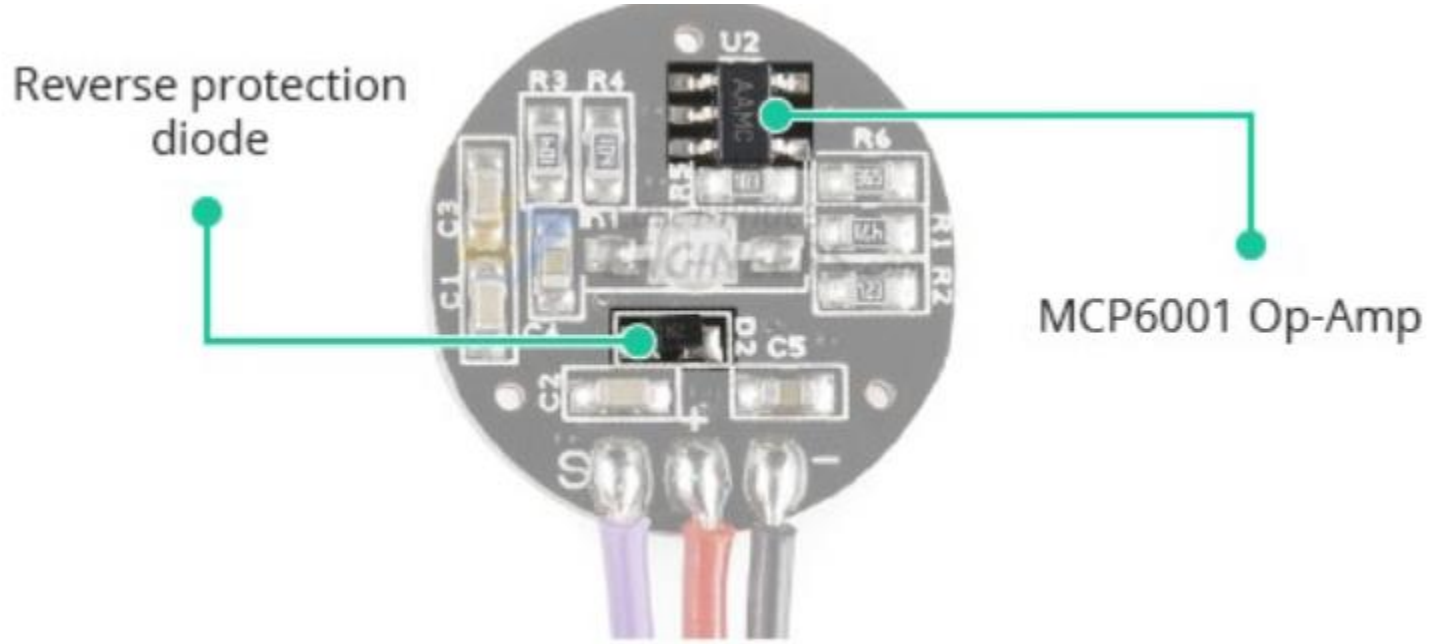


Front view of the Pulse sensor

Hardware overview of the pulse sensor

- On the back of the module you will find the rest of the components including a microchip's MCP6001 Op-Amp and a bunch of resistors and capacitors that make up the R/C filter network. There is also a reverse protection diode to prevent damage if the power leads are accidentally reversed.
- The module operates from a 3.3 to 5V DC Voltage supply with a operating current of < 4mA.

Hardware overview of the pulse sensor

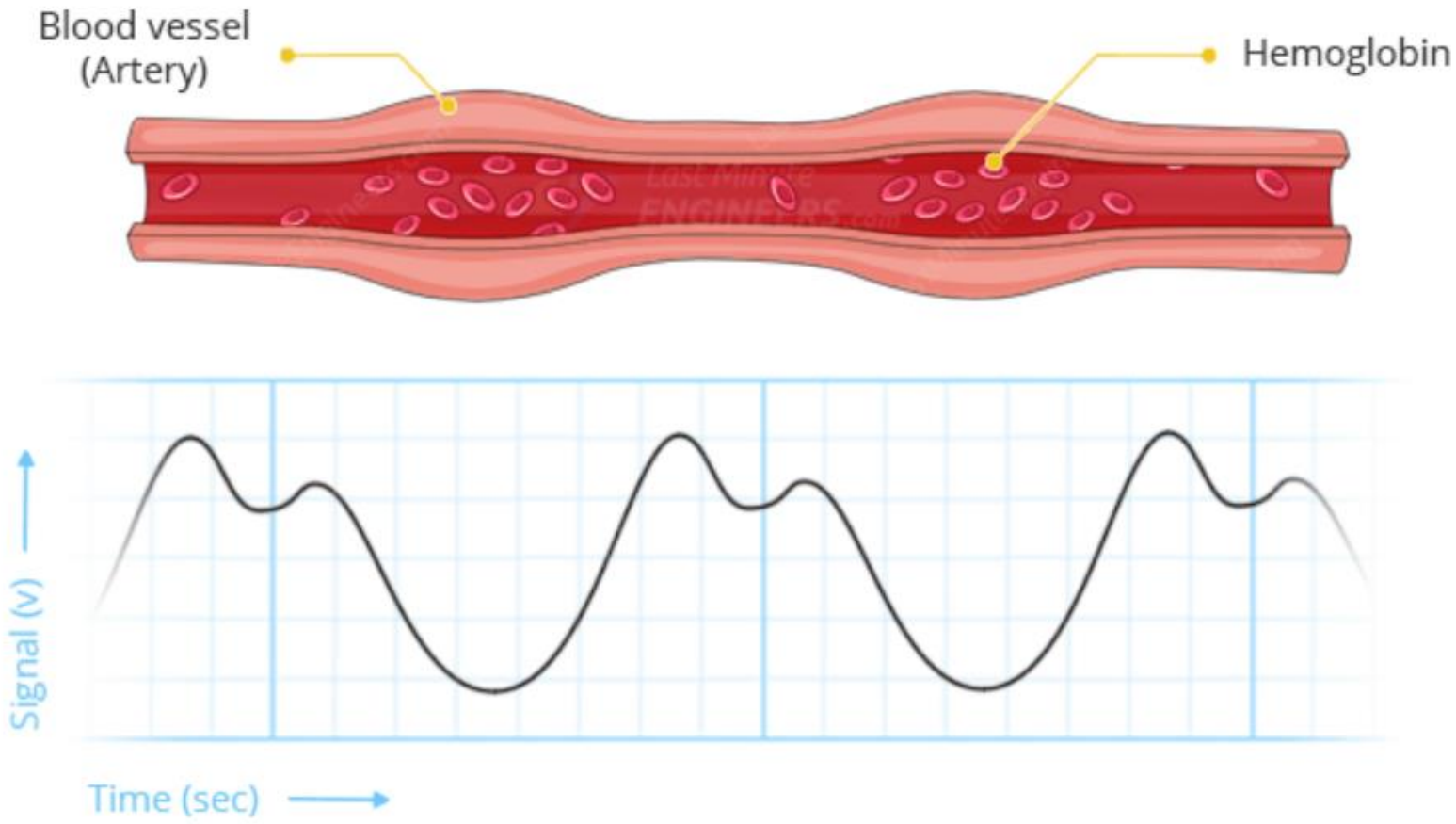


Back view of the Pulse sensor

How Pulse Sensor Works

- A pulse sensor or any optical heart-rate sensor, works by shining a green light (~ 550nm) on the finger and measuring the amount of reflected light using a photosensor.
- This method of pulse detection through light is called Photoplethysmogram.
- The oxygenated hemoglobin in the arterial blood has the characteristic of absorbing green light.
- The redder the blood (the higher the hemoglobin), the more green light is absorbed.

How Pulse Sensor Works



How Pulse Sensor Works

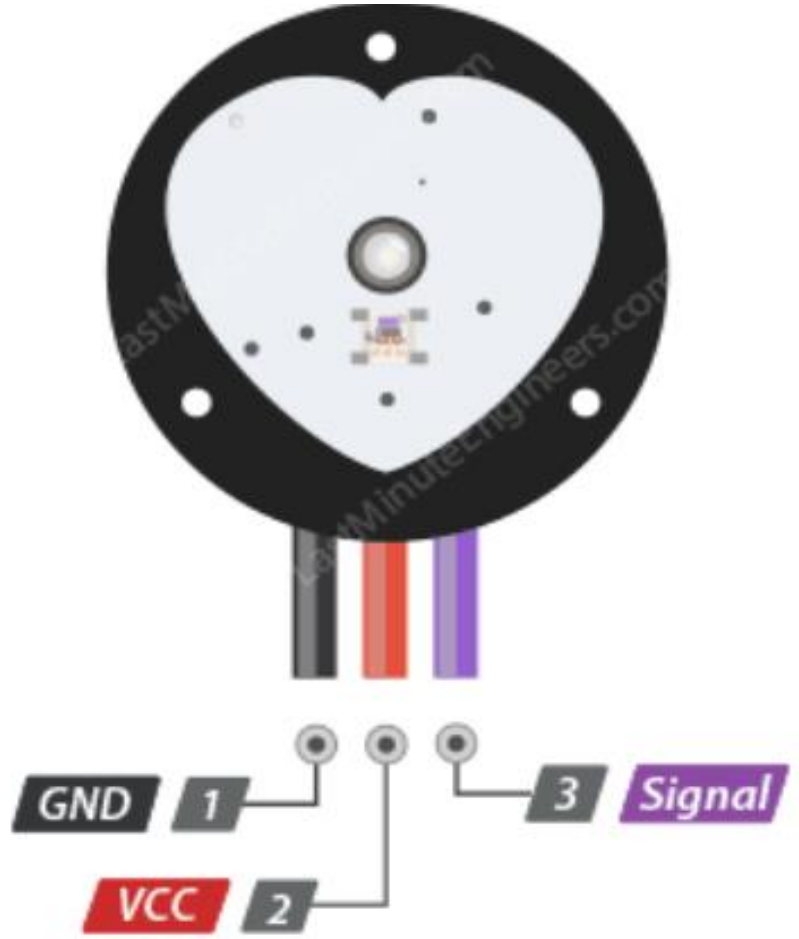
- As the blood is pumped through the finger with each heartbeat, the amount of reflected light changes, creating a changing waveform at the output of the photosensor.
- As you continue to shine light and take photosensor readings, you quickly start to get a heart-beat pulse reading.
- This signal from the photosensor is generally small and noisy, therefore the signal is passed through an R/C filter network and then amplified using an Op Amp to create a signal that is much larger, cleaner and easier to detect.

Pulse Sensor Pinout

S (Signal) is the signal output. Connects to analog input of an Arduino.

+ (VCC) is the VCC pin. Connects to 3.3 or 5V.

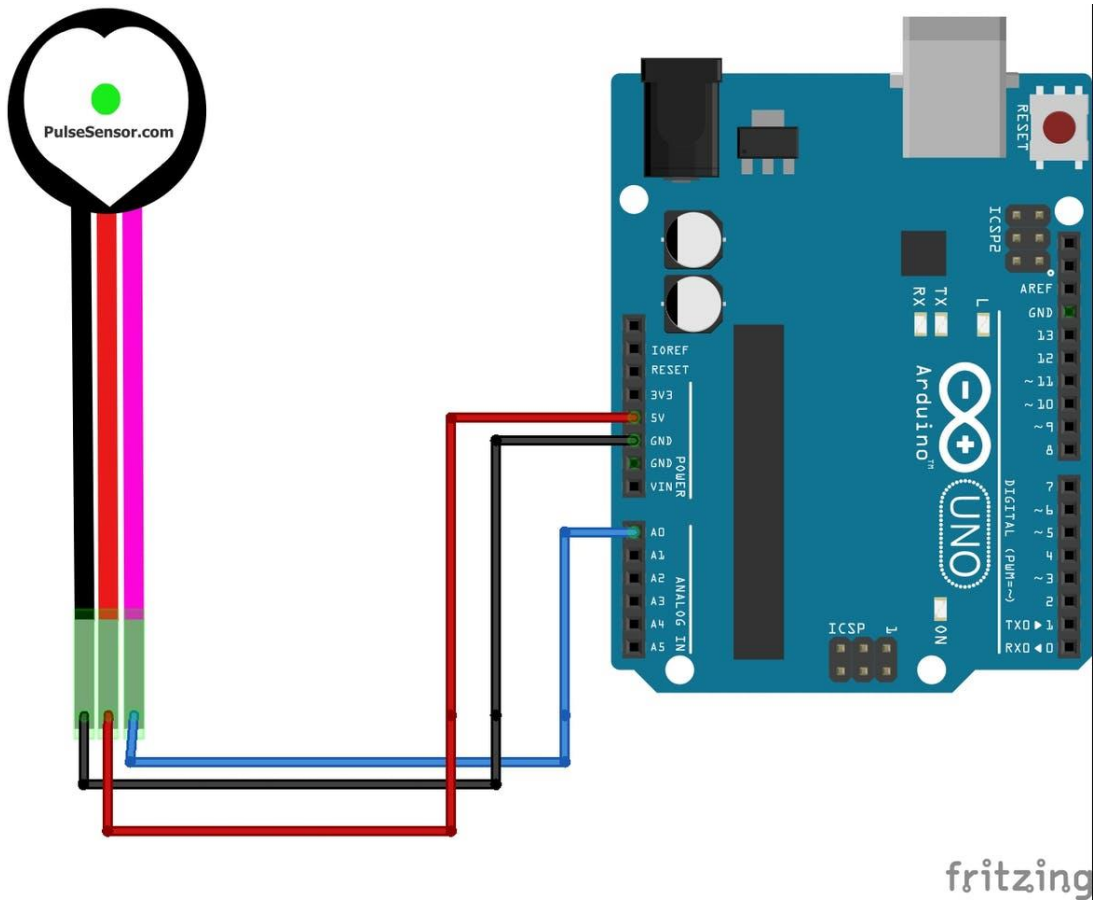
- (GND) is the Ground pin.

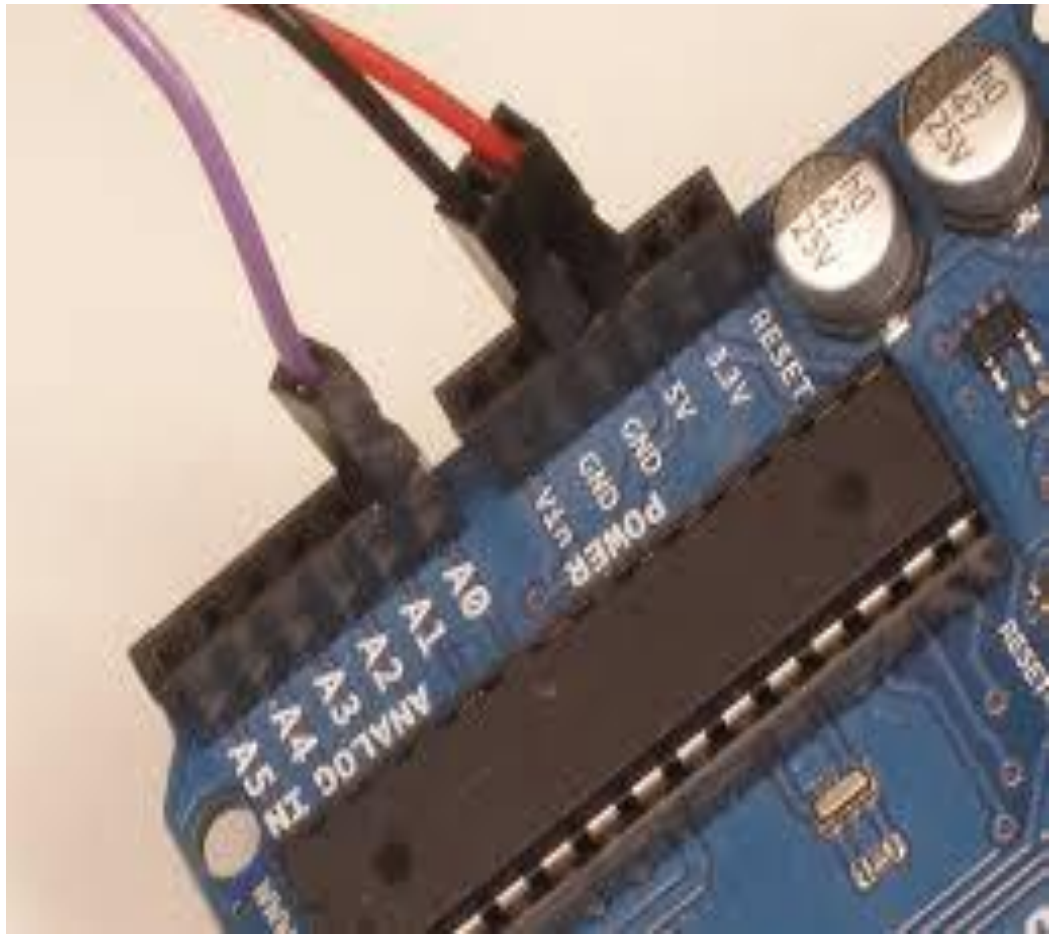


Wiring Pulse Sensor with Arduino

- Hooking up the Pulse Sensor to an Arduino is simple. You only need to connect three wires:
  - two for power and one for reading the sensor value.
- The module can be powered from 3.3 or 5V. The positive voltage connects to ‘+’ and ground connects to ‘-‘.
- The 3rd ‘S’ wire is the analog signal output from the sensor and this will connect to the A0 analog input of an Arduino.

Wiring Pulse Sensor with Arduino





Basic Arduino Code – Blink with the Heartbeat

- It begins with defining the pins used to connect the Pulse Sensor.
- Two variables are also defined; the Signal holds the incoming ADC data and the Threshold determines which signal to “count as a beat” and which to ignore.

```
int const PULSE_SENSOR_PIN = 0;

int Signal;
int Threshold = 550;
```

Basic Arduino Code – Blink with the Heartbeat

- In the setup, the Built-in LED pin (pin 13) can be defined as the output and set up the serial monitor as below:

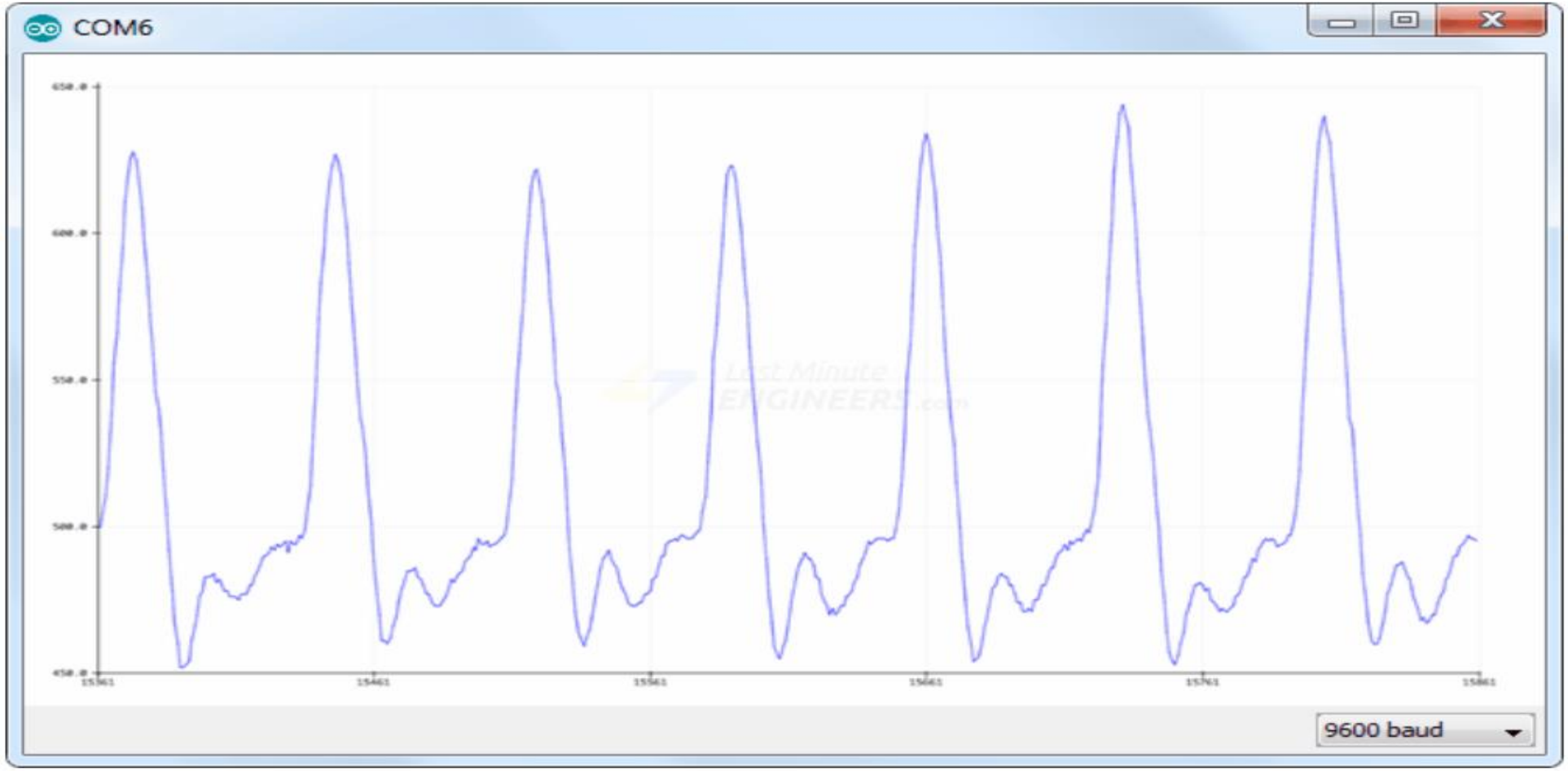
```
void setup() {
    pinMode(LED_BUILTIN,OUTPUT);
    Serial.begin(9600);
}
```

Basic Arduino Code – Blink with the Heartbeat

- In the loop, the analog signal is read from the Pulse Sensor, and when the signal exceeds a threshold value, the Built-in LED turned ON. The complete code is as:

```
int const PULSE_SENSOR_PIN = 0;    // 'S' Signal pin connected to A0

int Signal;                // Store incoming ADC data. Value can range from 0-1024
int Threshold = 550;       // Determine which Signal to "count as a beat" and which

void setup() {
    pinMode(LED_BUILTIN,OUTPUT);  // Built-in LED will blink to your heartbeat
    Serial.begin(9600);           // Set comm speed for serial plotter window
}

void loop() {

    Signal = analogRead(PULSE_SENSOR_PIN); // Read the sensor value

    Serial.println(Signal);                // Send the signal value to serial plotte

    if(Signal > Threshold){                // If the signal is above threshold, turn
        digitalWrite(LED_BUILTIN,HIGH);
    } else {
        digitalWrite(LED_BUILTIN,LOW);     // Else turn off the LED
    }
    delay(10);
}
```

Arduino Code: Heartbeat Plotting



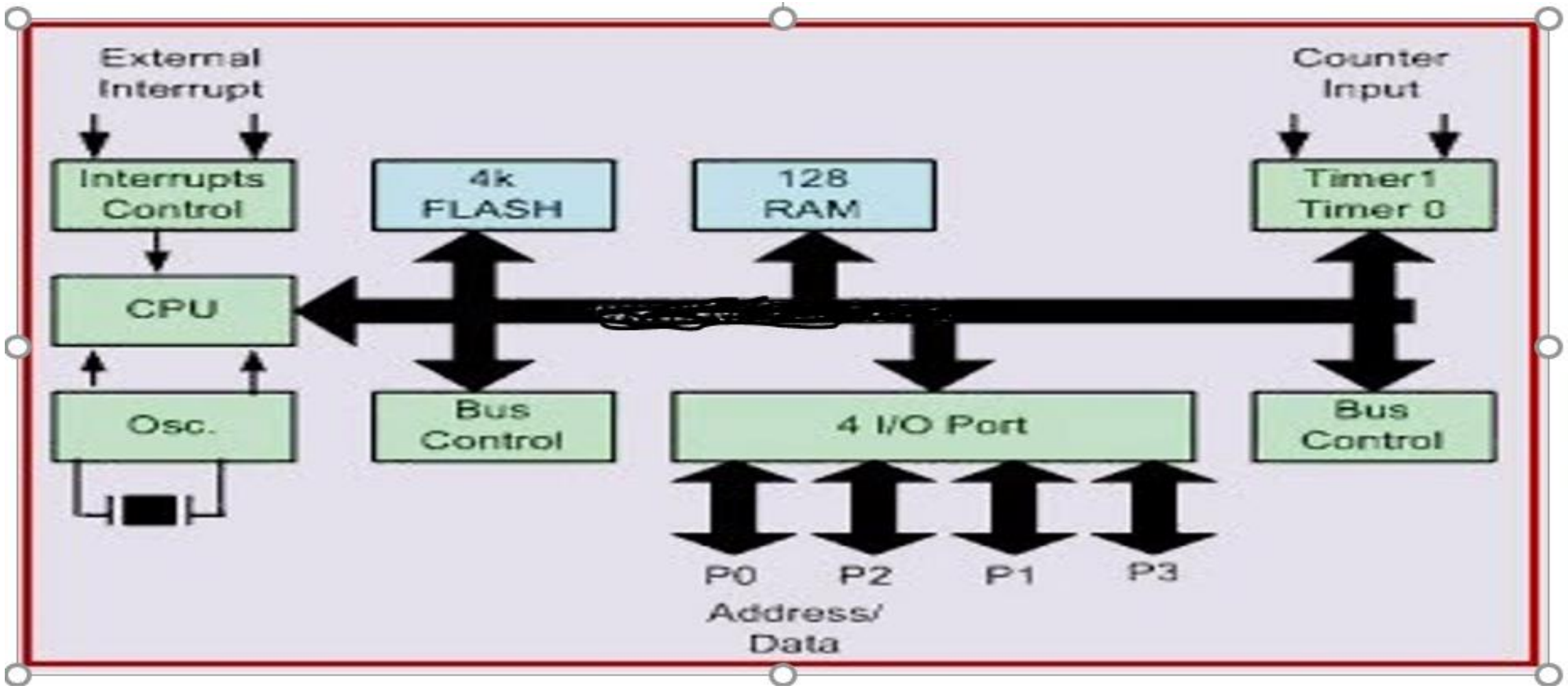
Microcontrollers

- Evolution of microprocessors
  - 8085
  - 8086
  - 8088
  - 80186
  - 80286
  - 80386
  - 80486
  - Intel Pentium
  - Intel Celeron
  - Intel Core Processor etc.

Microcontrollers

- Evolution of Microcontrollers
  - 8051
  - 80151
  - 80251
  - MCS96
  - LPC2148
  - PIC Microcontrollers

8051 Microcontroller Architectural Layout



8051 Microcontroller

- 8051 microcontroller has got everything inbuilt.

Internet of Things: Introduction to Microcontrollers 8051
Lect #5

Prof. Suchismita Chinara

Dept. of Computer Science Engg.

National Institute of Technology Rourkela-769008

Microcontroller

- A **microcontroller** is a small and low-cost microcomputer, which is designed to perform the specific tasks of embedded systems like displaying microwave's information, receiving remote signals, etc.
- The general microcontroller consists of the processor, the memory (RAM, ROM, EPROM), Serial ports, peripherals (timers, counters), etc.
- Microcontrollers are used to execute a single task within an application.
- Its designing and hardware cost is low.
- It is built with CMOS technology, which requires less power to operate.
- It consists of CPU, RAM, ROM, I/O ports.

Types of Microcontrollers

- Microcontrollers are divided into various categories based on:
  - ❖ memory
  - ❖architecture
  - ❖bits
  - ❖instruction sets.

Types based on Bit configuration

- **8-bit microcontroller** − This type of microcontroller is used to execute arithmetic and logical operations like addition, subtraction, multiplication division, etc. For example, Intel 8031 and 8051 are 8 bits microcontroller.
- **16-bit microcontroller** − This type of microcontroller is used to perform arithmetic and logical operations where higher accuracy and performance is required. For example, Intel 8096 is a 16-bit microcontroller.
- **32-bit microcontroller** − This type of microcontroller is generally used in automatically controlled appliances like automatic operational machines, medical appliances, etc.

Types based on memory configuration

- **External memory microcontroller** − This type of microcontroller is designed in such a way that they do not have a program memory on the chip. Hence, it is named as external memory microcontroller. For example: Intel 8031 microcontroller.
- **Embedded memory microcontroller** − This type of microcontroller is designed in such a way that the microcontroller has all programs and data memory, counters and timers, interrupts, I/O ports are embedded on the chip. For example: Intel 8051 microcontroller.

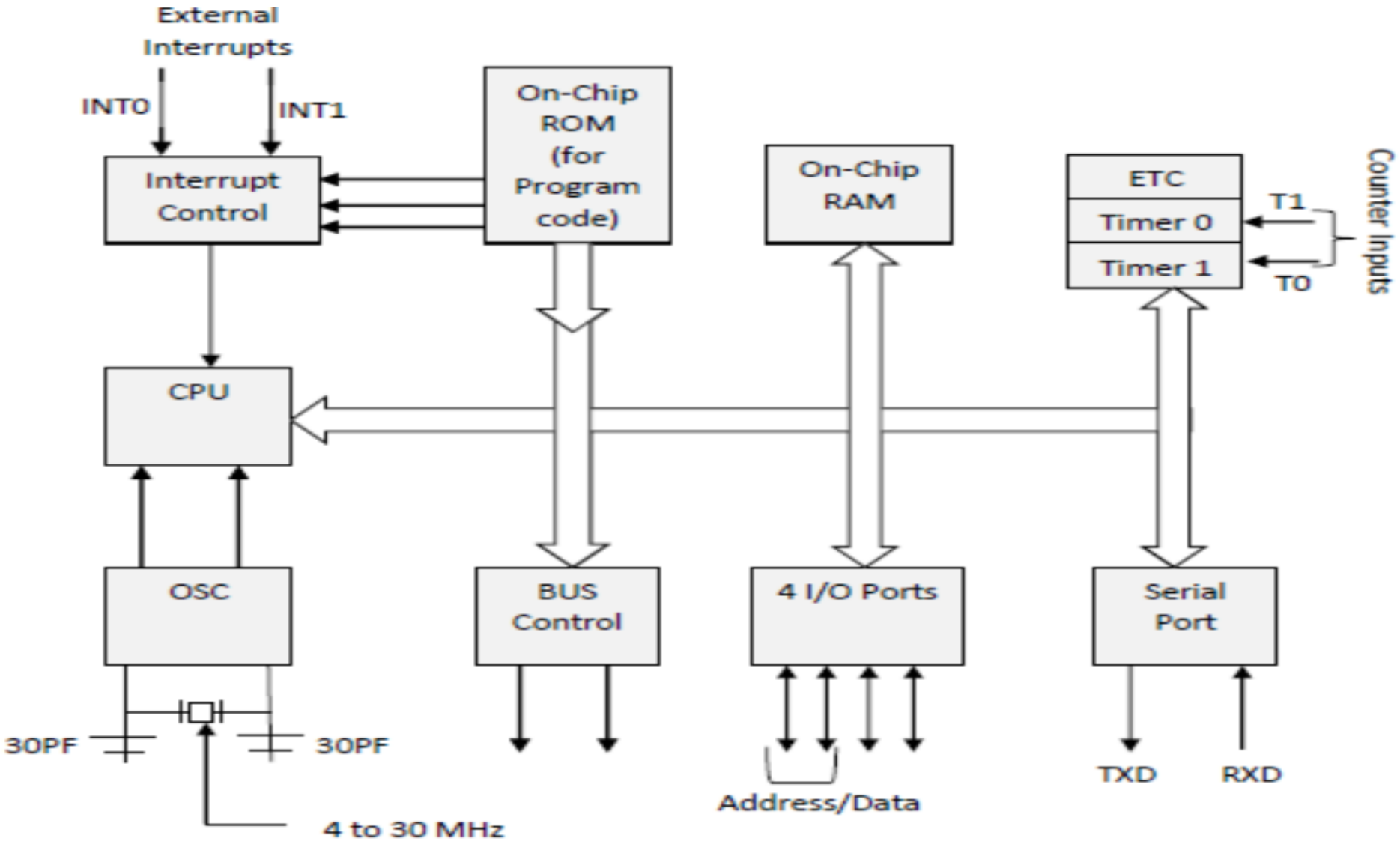
Types based on Instruction sets

- **CISC** − CISC stands for Complex Instruction Set Computer. It allows the user to insert a single instruction as an alternative to many simple instructions.
- **RISC** − RISC stands for Reduced Instruction Set Computers. It reduces the operational time by shortening the clock cycle per instruction.

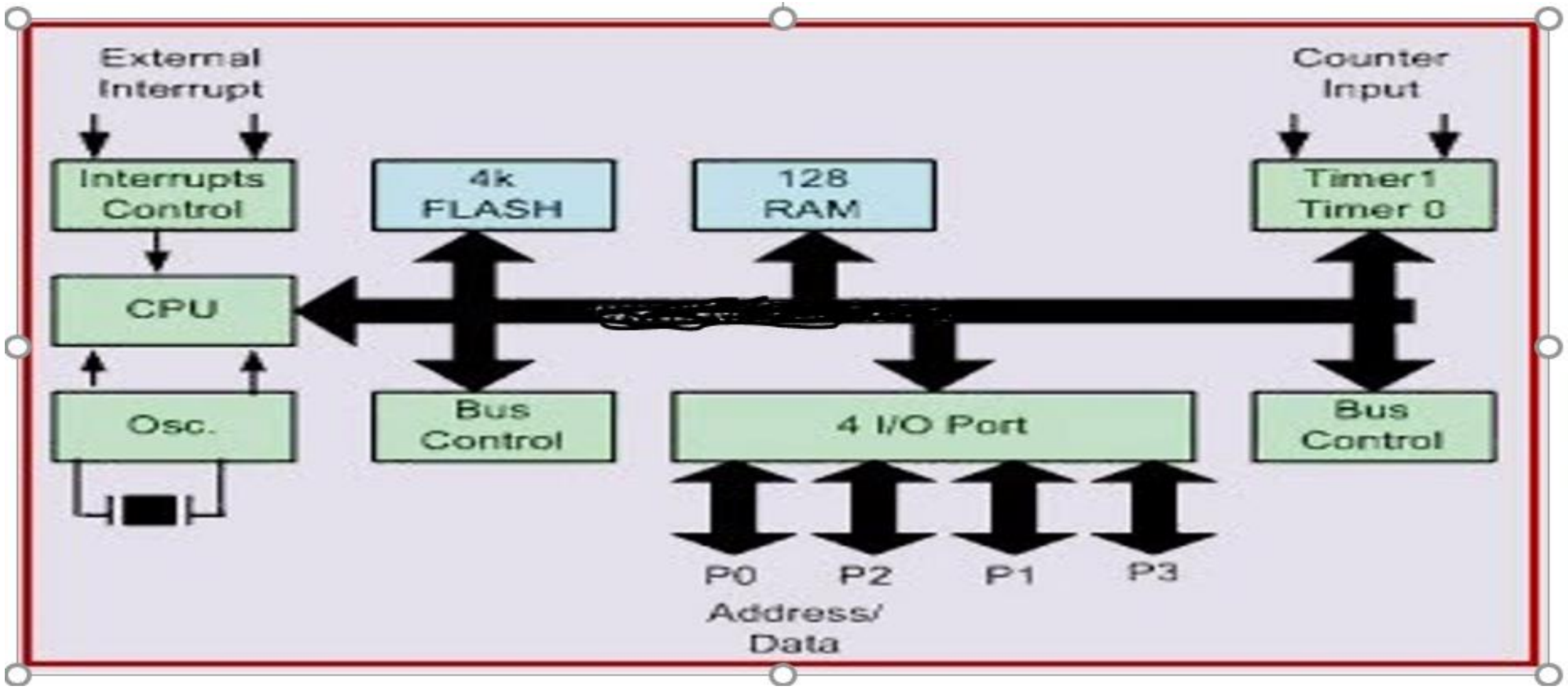
Applications of Microcontrollers

- Light sensing and controlling devices like LED.
- Temperature sensing and controlling devices like microwave oven, chimneys.
- Fire detection and safety devices like Fire alarm.
- Measuring devices like Volt Meter.

8051 Microcontroller Architectural Layout



8051 Microcontroller Architectural Layout



8051 Microcontroller : Features supported

- 8-bit microcontroller supporting 8-bit operations.
- Many general purpose (GPR) and few special function registers (SFRs)
  - Two major 8-bit registers for arithmetical operations and other applications- A and B. A is the accumulator register and it is a bit addressable register
  - 21 SFRs.
  - Few registers are bit addressable.
  - Control registers for timer, serial communication, and interrupts fall under SFR category.

8051 Microcontroller : Features supported

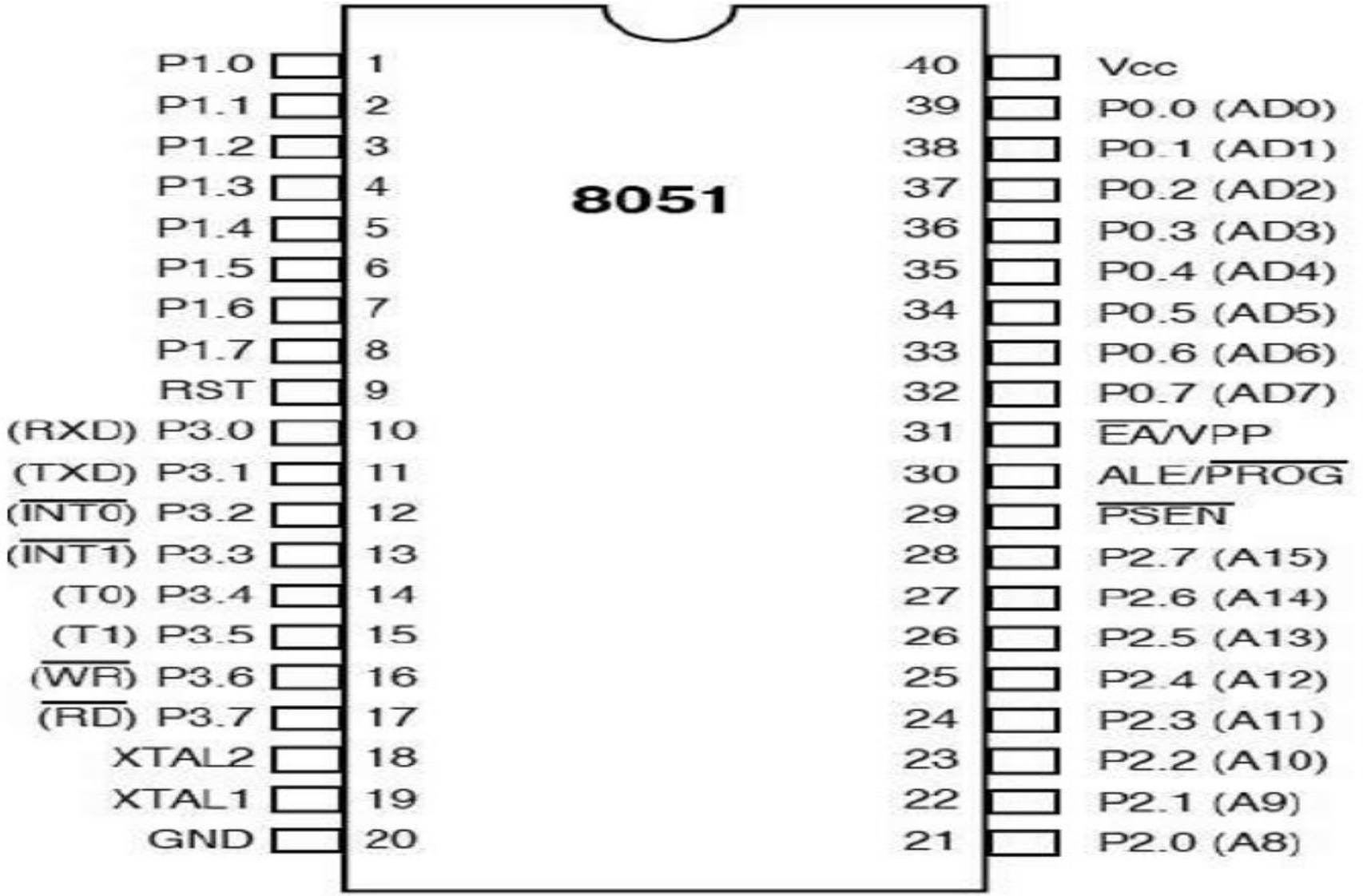
- Four register banks having 8 registers in each bank.
  - These 32 registers can be used by the programmers
  - Stack pointer register: points to the stack (8-bit register)

- Data pointer: 16 bit register (combination of two 8-bit registers (DPL & DPH)
- 16-bit program counter (PC) – holds the address of the next instruction to be executed.
- 8- bit Program Status Word (PSW) – It is the flag register and helps in selection of register banks as well.

8051 Microcontroller : Features supported

- Internal ROM or EPROM: size 4K
- Internal user accessible RAM 128 bytes
- 40 pins package: four ports P0, P1, P2, and P3 (can be configured as input or output ports)
- Two 16 bits timer / counter: T0 and T1
- Full duplex serial data receiver / transmitter
- Support for interrupt programming
- Oscillator and clock circuit
- Easier and simpler instruction set

8051 pin description



8051 pin description

- **Pins 1 to 8 −** These pins are known as Port 1. This port doesn't serve any other functions. It is internally pulled up, bi-directional I/O port.
- **Pin 9 −** It is a RESET pin, which is used to reset the microcontroller to its initial values.
- **Pins 10 to 17 −** These pins are known as Port 3. This port serves some functions like interrupts, timer input, control signals, serial communication signals RxD and TxD, etc.
- **Pins 18 & 19 −** These pins are used for interfacing an external crystal to get the system clock.
- **Pin 20 −** This pin provides the power supply to the circuit.
- **Pins 21 to 28 −** These pins are known as Port 2. It serves as I/O port. Higher order address bus signals are also multiplexed using this port.
- **Pin 29 −** This is PSEN pin which stands for Program Store Enable. It is used to read a signal from the external program memory.
- **Pin 30 −** This is EA pin which stands for External Access input. It is used to enable/disable the external memory interfacing.
- **Pin 31 −** This is ALE pin which stands for Address Latch Enable. It is used to demultiplex the address-data signal of port.
- **Pins 32 to 39 −** These pins are known as Port 0. It serves as I/O port. Lower order address and data bus signals are multiplexed using this port.
- **Pin 40 −** This pin is used to provide power supply to the circuit.

8051 input output ports

- 8051 microcontrollers have 4 I/O ports each of 8-bit, which can be configured as input or output. Hence, total 32 input/output pins allow the microcontroller to be connected with the peripheral devices.
- **Pin configuration**, i.e. the pin can be configured as 1 for input and 0 for output as per the logic state.
  - **Input/Output (I/O) pin** − All the circuits within the microcontroller must be connected to one of its pins except P0 port because it does not have pull-up resistors built-in.
  - **Input pin** − Logic 1 is applied to a bit of the P register. The output FE transistor is turned off and the other pin remains connected to the power supply voltage over a pull-up resistor of high resistance.

8051 input output ports

- **Port 0** – The P0 (zero) port is characterized by two functions –
  - When the external memory is used then the lower address byte (addresses A0A7) is applied on it, else all bits of this port are configured as input/output.
  - When P0 port is configured as an output then other ports consisting of pins with built-in pull-up resistor connected by its end to 5V power supply, the pins of this port have this resistor left out.
- Port 1 -
  - P1 is a true I/O port as it doesn't have any alternative functions as in P0, but this port can be configured as general I/O only. It has a built-in pull-up resistor and is completely compatible with TTL circuits.

8051 input output ports

- Port 2 -
  - P2 is similar to P0 when the external memory is used. Pins of this port occupy addresses intended for the external memory chip. This port can be used for higher address byte with addresses A8-A15. When no memory is added then this port can be used as a general input/output port similar to Port 1.
- Port 3
  - In this port, functions are similar to other ports except that the logic 1 must be applied to appropriate bit of the P3 register.

8051 Interrupts

- Interrupts are the events that temporarily suspend the main program, pass the control to the external sources and execute their task. It then passes the control to the main program where it had left off.
- 8051 has 5 interrupt signals, i.e. INT0, TFO, INT1, TF1, RI/TI. Each interrupt can be enabled or disabled by setting bits of the IE register and the whole interrupt system can be disabled by clearing the EA bit of the same register.

IE (Interrupt Enable) Register

- This register is responsible for enabling and disabling the interrupt. EA register is set to one for enabling interrupts and set to 0 for disabling the interrupts.

EA	-	-	ES	ET1	EX1	ET0	EX0

IE (Interrupt Enable) Register

EA	IE.7	It disables all interrupts. When EA = 0 no interrupt will be acknowledged and EA = 1 enables the interrupt individually.
-	IE.6	Reserved for future use.
-	IE.5	Reserved for future use.
ES	IE.4	Enables/disables serial port interrupt.
ET1	IE.3	Enables/disables timer1 overflow interrupt.
EX1	IE.2	Enables/disables external interrupt1.
ET0	IE.1	Enables/disables timer0 overflow interrupt.
EX0	IE.0	Enables/disables external interrupt0.

IP (Interrupt Priority) Register

- We can change the priority levels of the interrupts by changing the corresponding bit in the Interrupt Priority (IP) register:
  - A low priority interrupt can only be interrupted by the high priority interrupt, but not interrupted by another low priority interrupt.
  - If two interrupts of different priority levels are received simultaneously, the request of higher priority level is served.
  - If the requests of the same priority levels are received simultaneously, then the internal polling sequence determines which request is to be serviced.

IP (Interrupt Priority) Register

-	-	PT2	PS	PT1	PX1	PT0	PX0
bit7	bit6	bit5	bit4	bit3	bit2	bit1	

-	IP.6	Reserved for future use.
-	IP.5	Reserved for future use.
PS	IP.4	It defines the serial port interrupt priority level.
PT1	IP.3	It defines the timer interrupt of 1 priority.
PX1	IP.2	It defines the external interrupt priority level.
PT0	IP.1	It defines the timer0 interrupt priority level.
PX0	IP.0	It defines the external interrupt of 0 priority level.

Internet of Things: Introduction to Microcontrollers 8051
Lect #6

Prof. Suchismita Chinara

Dept. of Computer Science Engg.

National Institute of Technology Rourkela-769008

Summary of 8051 Architecture features

- 8 bit CPU optimized for control applications
- 64K Program memory address space
- 64K Data memory address space
- 4KB bits on chip program memory
- 128 bytes on chip data RAM / Memory
- 32 bidirectional I/O lines
- Two 16 bit timers / counters
- 8051 is housed in a 40 pin DIP

8051 Family and Specifications

Features	8051	8052	8031
ROM	4K	8K	0K
RAM	128 Bytes	256 Bytes	128 Bytes
Timers	2	3	2
Input/output Pins	32	32	32
Serial Port	1	1	1
Interrupt Sources	6	8	6

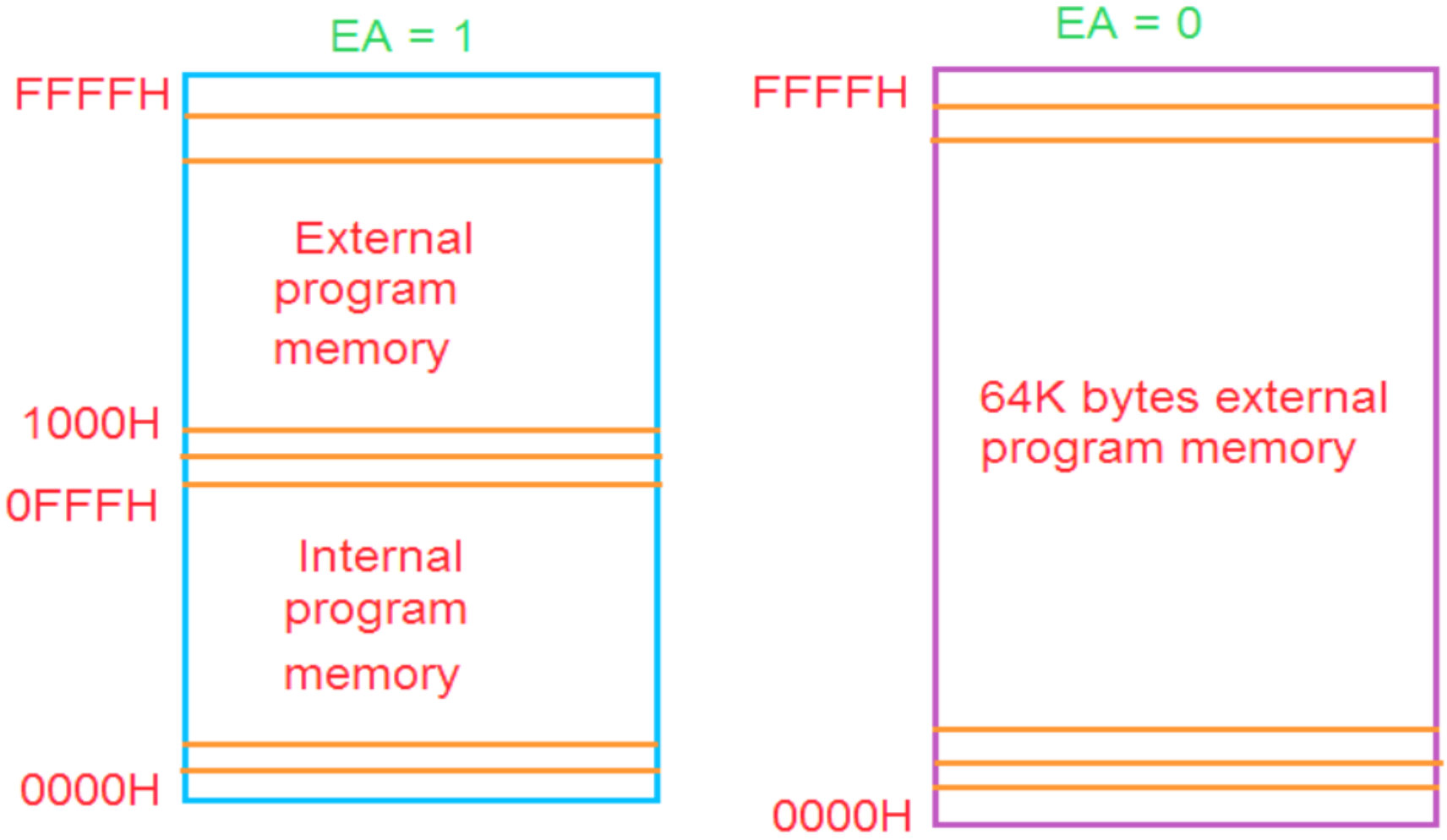
8051 Memory Organization of Microcontroller

- There are two broad categories of memory
  - Program Memory
  - Data Memory

8051 Program Memory

- Program memory is normally referred as ROM. (non-volatile)
- It is accessed through EA pin
- If EA is high, internal program memory is accessed from 0000H to 0FFFH memory location and external program memory accessed from 1000H to FFFFH memory locations.
- If EA is low, only external program memory accessed from 0000H to FFFFH memory locations.
- 8051 has a 4K byte internal program memory available
- It is used to store
  - Boot up program
  - Interrupt service routines (ISR)
  - Macro functions

8051 Program Memory

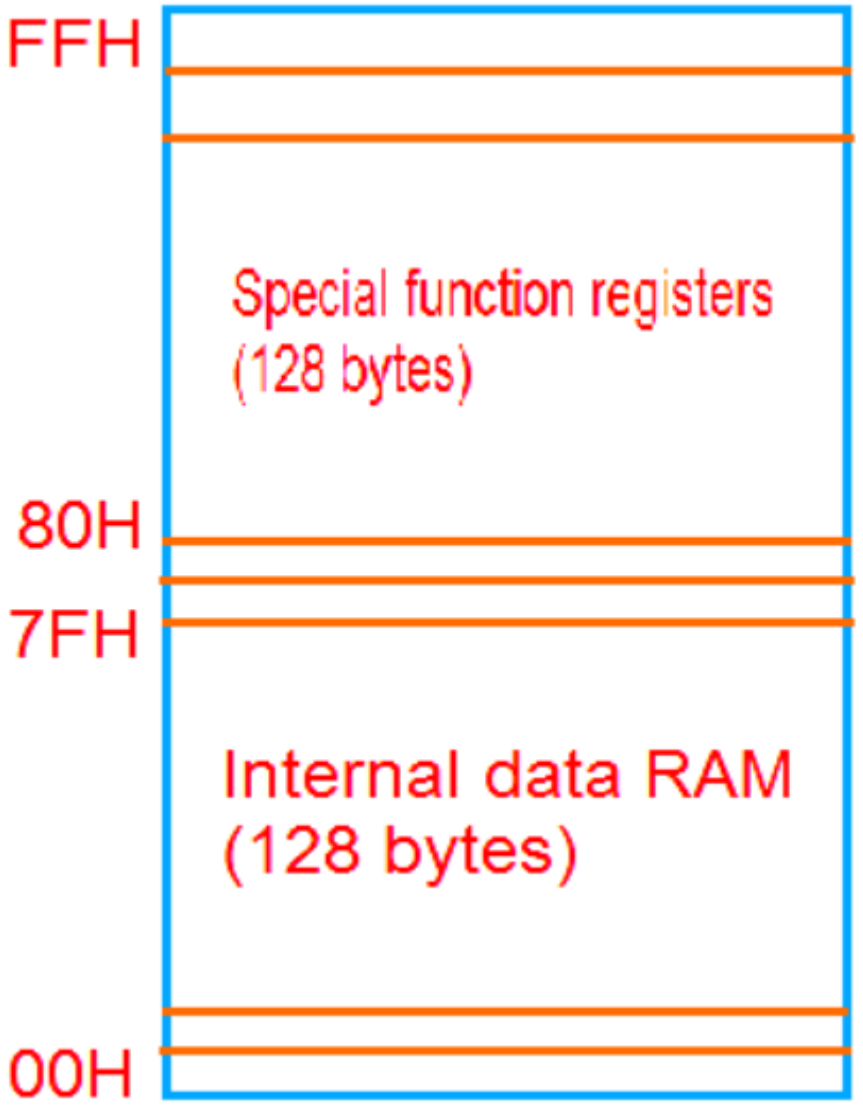


8051 Data memory

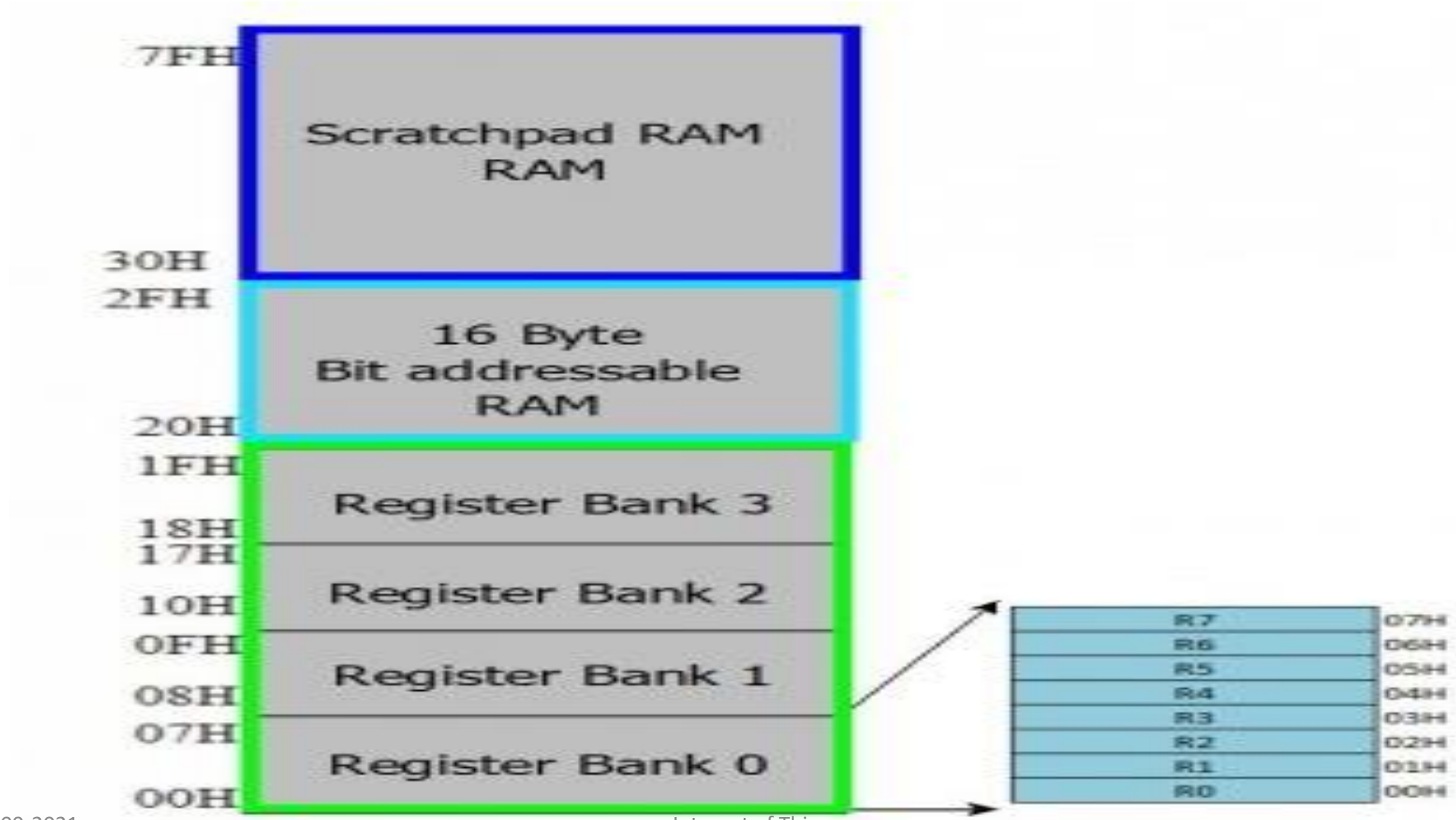
- Data memory is used to store the memory in the registers each of 64k bytes size.
- To access the data memory instruction MOVX is used.
- Data memory is of two types
  - Internal
  - External.

8051 Internal Data Memory

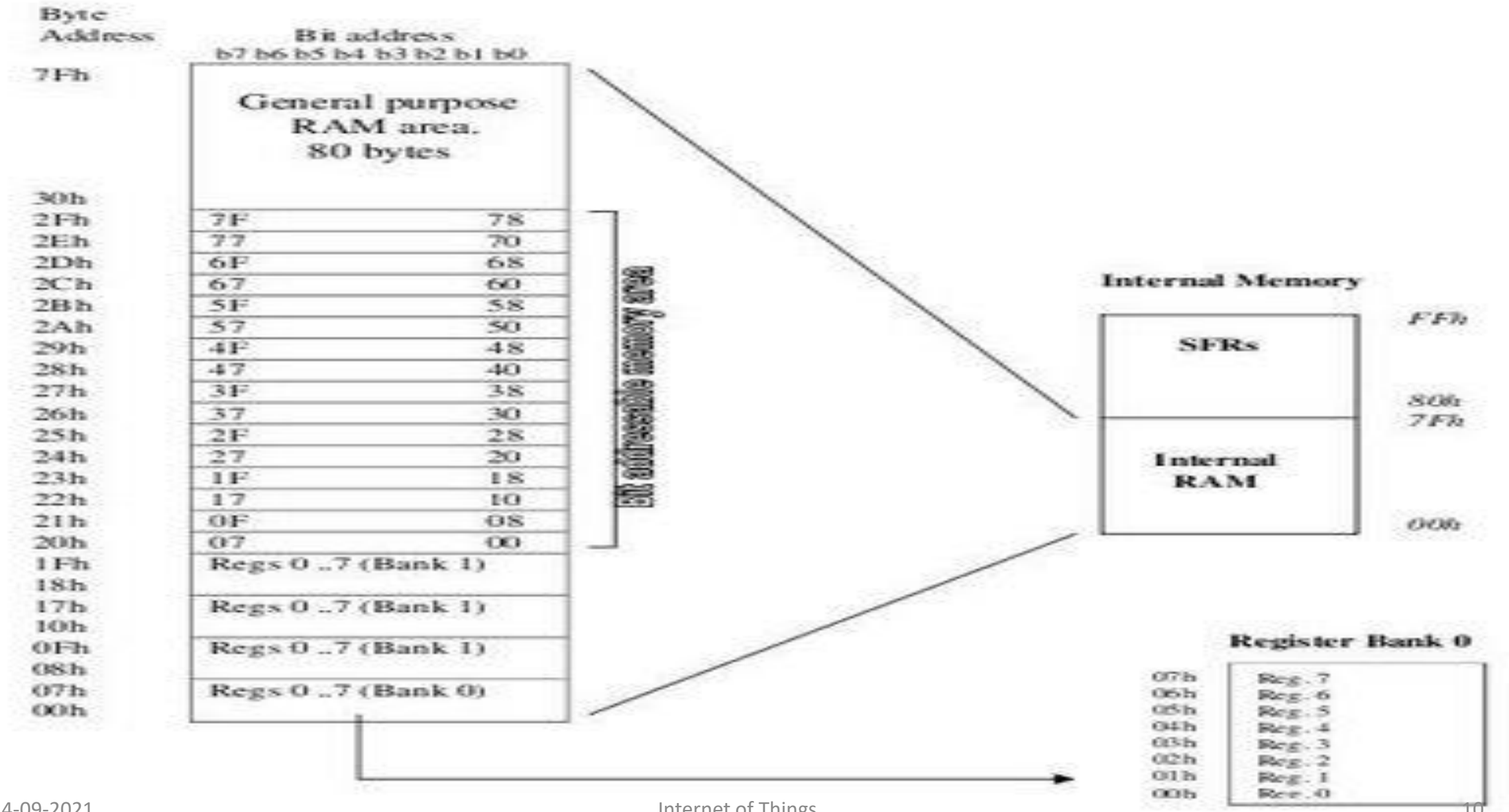
- The internal data memory consists of 256 bytes, these are divided into two parts:
- 00H-7FH for internal data RAM (128 bytes)
- 80H-FFH for special function registers (128 bytes)

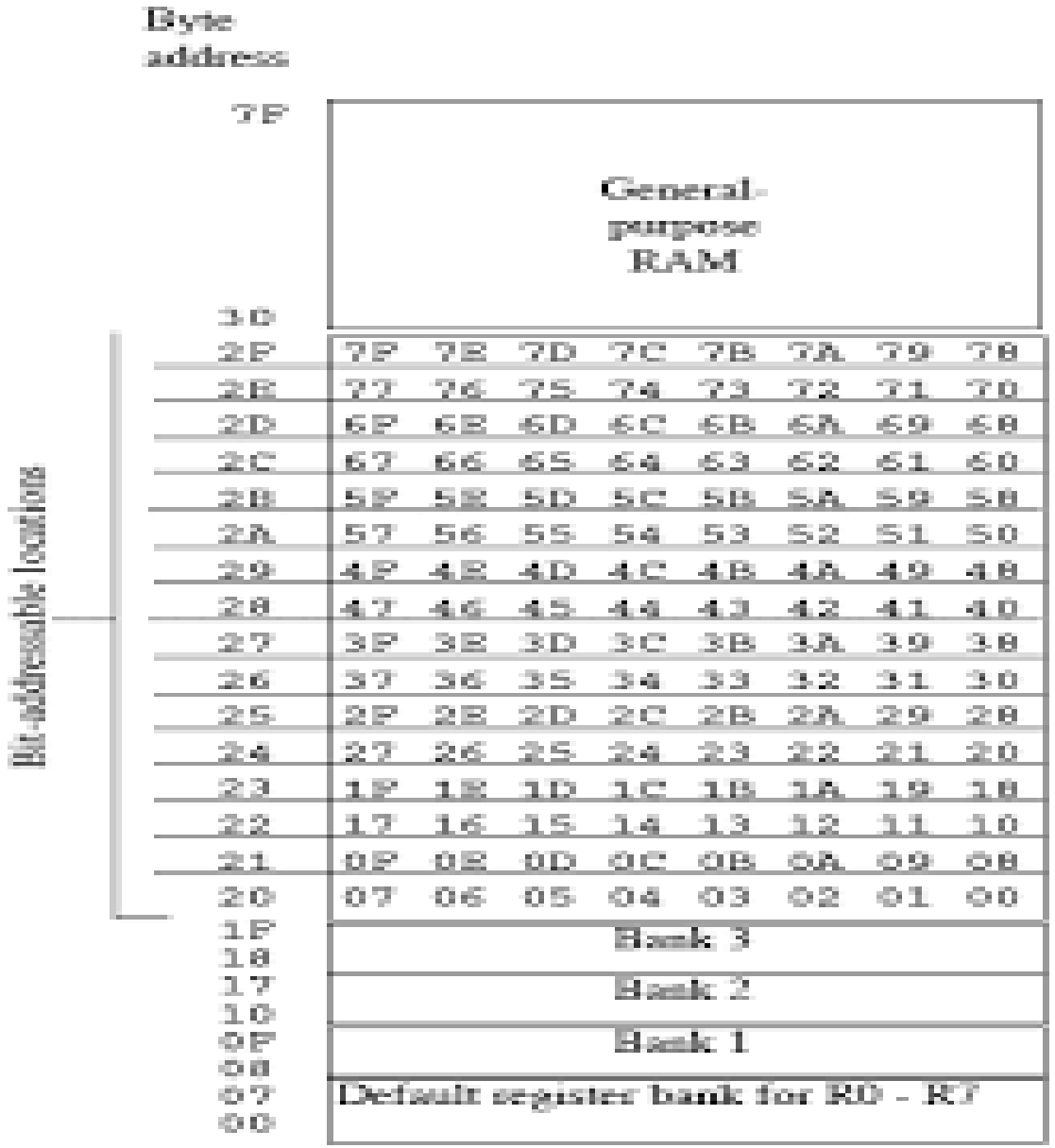


8051 Internal Data Memory Organization



8051 Internal Data Memory Organization





Detailed memory organization of Bit addressable locations

8051 – Register Banks

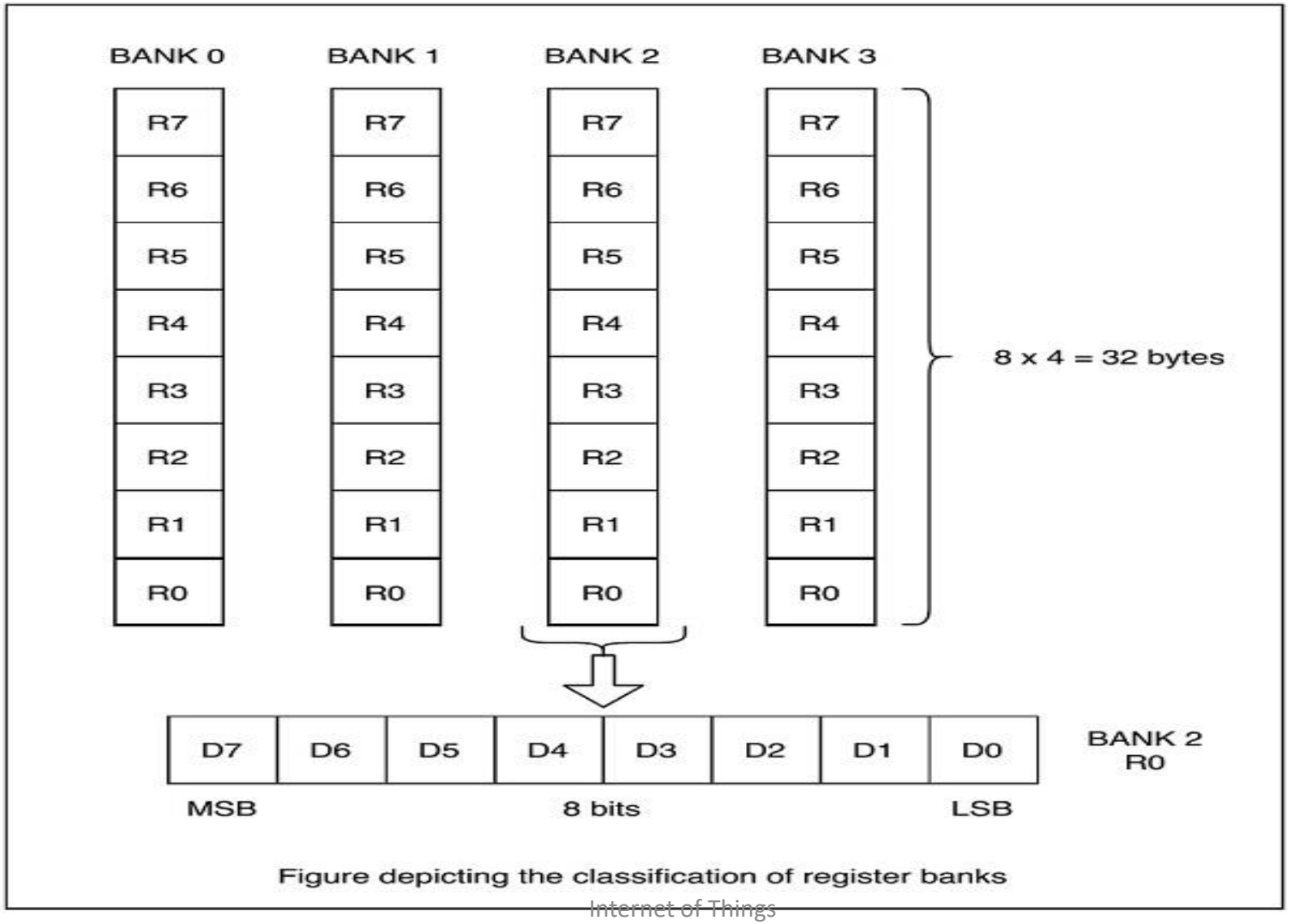


Figure depicting the classification of register banks

8051 External Data Memory

- The 8051 gives the facility to interface external RAM and ROM.
- External RAM is accessed by DPTR and up to 64KB of RAM can be interfaced.
- External data memory interfacing is of two types i.e. RAM and ROM interfacing.

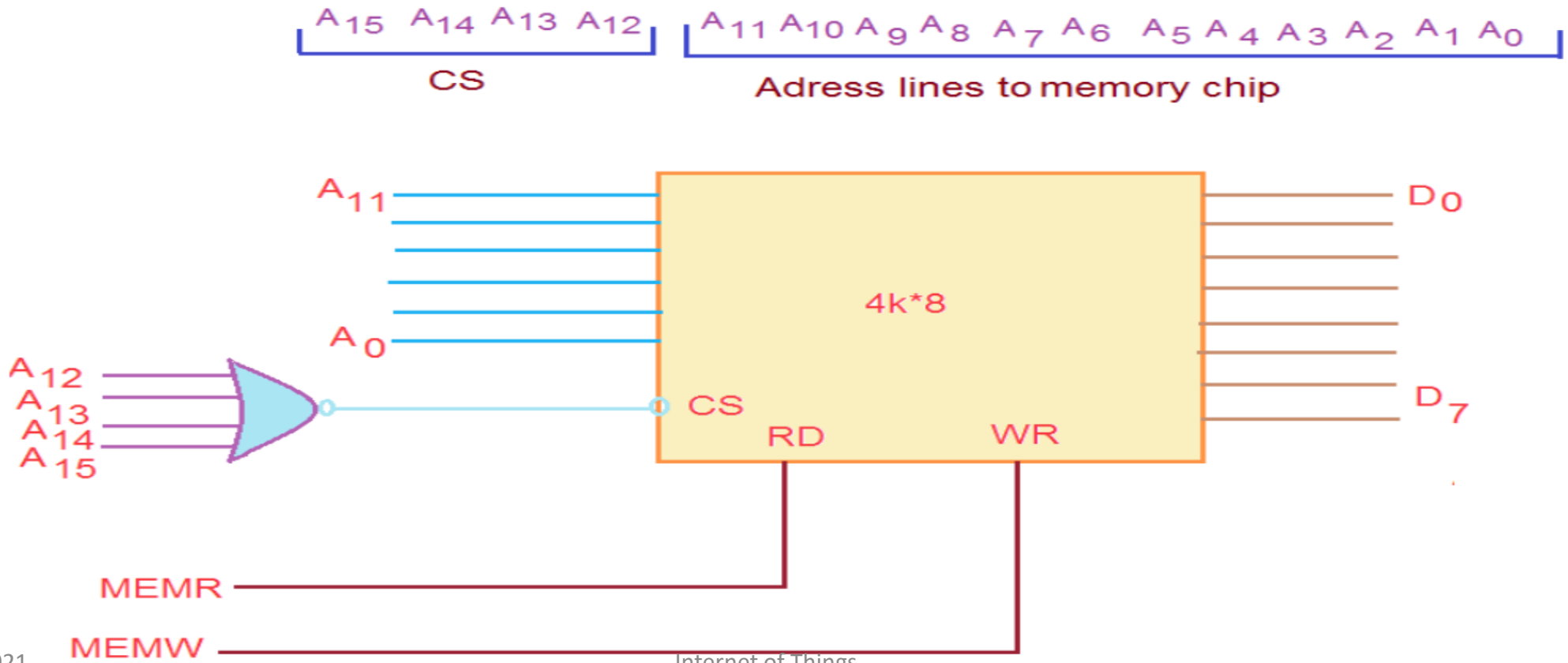
- DPTR (Data Pointer) – Contains current Data pointer's value.

8051 External Data Memory: RAM Interfacing

- The interfacing of memory chip with microcontroller has some regulations to follow:
  - The memory data bus is directly connected to memory chip data pins
  - Control signal connection
    - RD(Read Memory) connected to OE (Output Enable)
    - WR(Write Memory) connected to WE(Write Enable)
  - The CPU address lines are directly connected to memory chip addressing lines.

Interfacing 4KB RAM

- The memory chip consists of Chipset (CS) and Chip enable (CE) address lines varies based on memory capacity
- The accessing of memory is done when chip is activated.



8051 External Data Memory: ROM Interfacing

- In many systems the on chip ROM of 8051 is not sufficient.
- To Indicate the program code is stored in microcontroller on chip ROM, EA pin is connected to Vcc.
- To indicate program code is stored in external ROM, EA pin is connected to ground.

Internet of Things: Microcontroller Lect #7

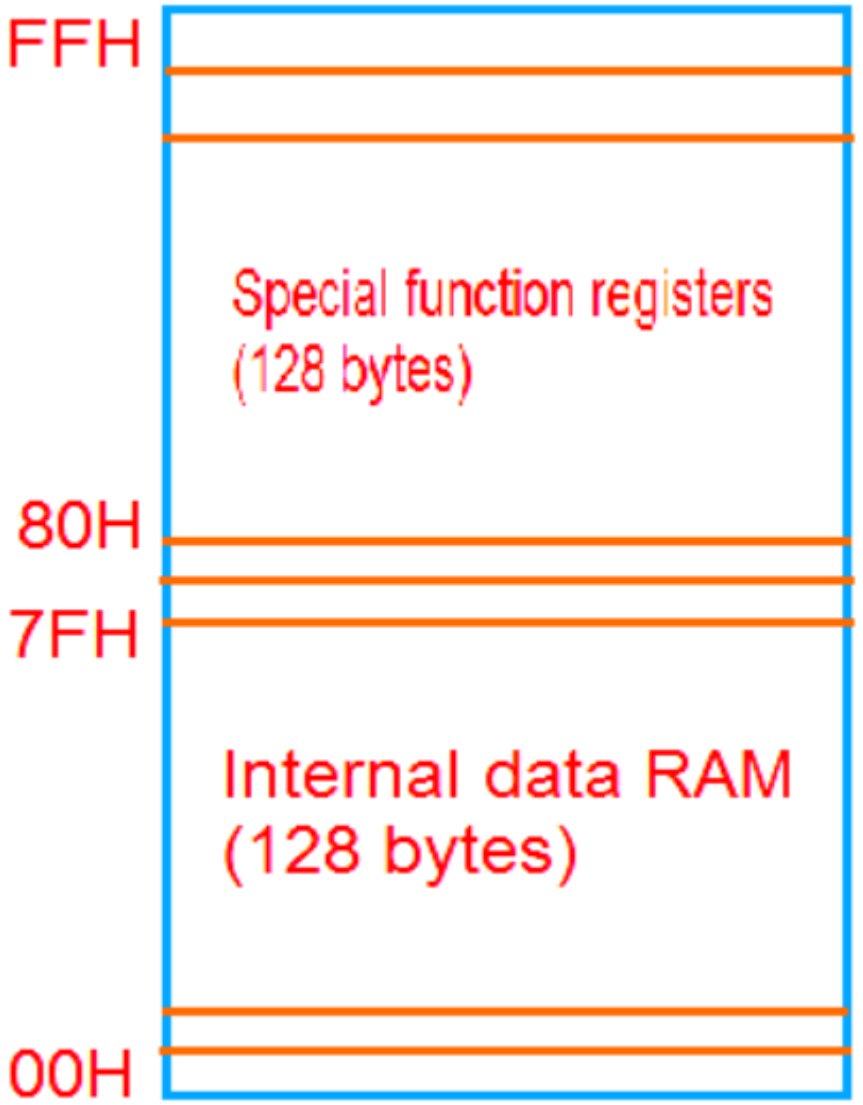
Prof. Suchismita Chinara

Dept. of Computer Science Engg.

National Institute of Technology Rourkela-769008

8051 Internal Data Memory

- The internal data memory consists of 256 bytes, these are divided into two parts:
- 00H-7FH for internal data RAM (128 bytes)
- 80H-FFH for special function registers (128 bytes)



Special Function Registers of 8051

- There are 21 SFRs available in 8051 microcontroller.

	SFR MEMORY MAP								
F8									FF
F0	B								F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW								D7
C8	T2CON		RCAP2L	RCAP2H	TL2	TH2			CF
C0									C7
B8	IP								BF
B0	P3								B7
A8	IE								AF
A0	P2								A7
98	SCON	SBUF							9F
90	P1								97
88	TCON	TMOD	TL0	TL1	THO	TH1			8F
80	P0	SP	DPL	DPH				PCON	87
MEMORY LOCATIONS	← 8 BYTES →								MEMORY LOCATIONS

Special Function Registers (SFRs): Accumulator (A)

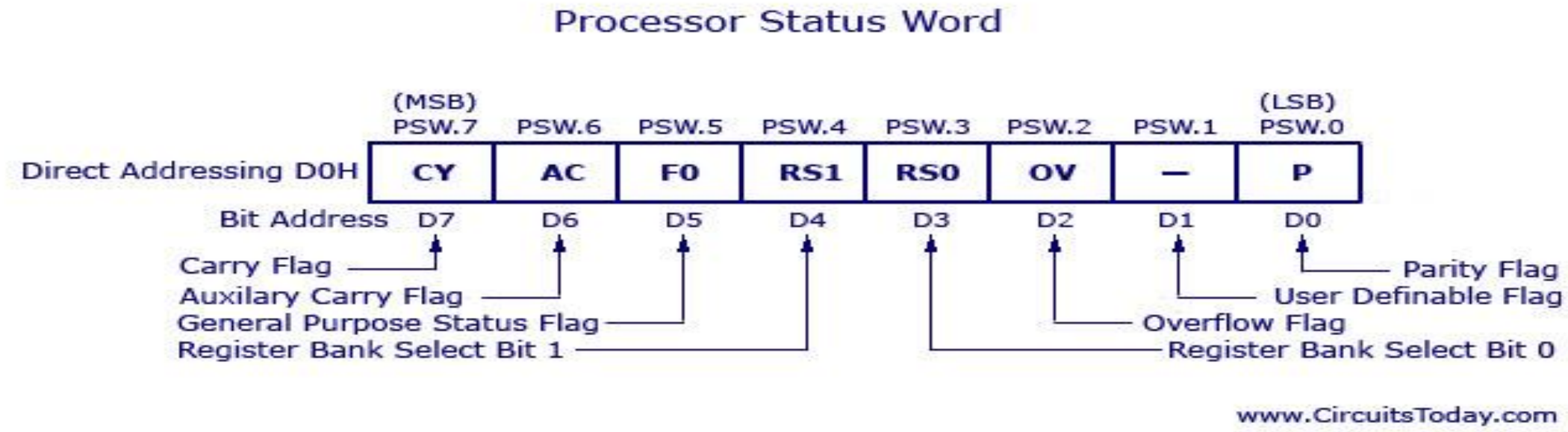
- It is an 8 bit register, used in all mathematical and arithmetical operations.
- It is bit addressable and by default, it has 0 in all bits.
- The results obtained from all arithmetical and mathematical operations are stored in this register.
- In operations like addition, subtraction etc. one of the operands remains in A.

SFR: B

- It is very significant when multiplication and divisions are done.
- B must be one of the operands in these operations.
- B is 8-bit register and is bit addressable.

SFR: Program Status Word (PSW)

- Microcontrollers have the provision of indicators to know if something is going wrong.
- These indicators are called FLAGs.
- All these FLAGs are accommodated in as SFR called PSW.
- PSW is bit accessible.



PSW Register

- P- Parity Flag (PSW.0): It will be set or reset based on the value in the accumulator. If it has odd number of 1s then it is set to 1 or it is set to 0.
- PSW.1 : This field is not used and is reserved for future use
- PSW.2: OV (Overflow Flag): When result of a signed operation is very large, OV flag is used. It will be set in that case, else it is reset.
- PSW.3 & PSW.4: RS0 & RS1 (Register bank select) : There are four register banks supported in 8051 and each of them has 8 registers from R0 to R7. The user has the flexibility of selecting one of those banks depending on the values in two bits.

PSW Register

- PSW.5: FO: This is a general purpose bit that can be used by a user as per the need.
- PSW.6: Auxiliary Carry: It is used in case of BCD related operation.
- PSW.7: Carry: This FLAG is used by the processor in arithmetic and shift operations.

SFR: Registers for Timer & counter operations

- TCON
- TMOD
- TL1
- TL0
- TH1
- TH0

SFR

- Registers useful for interfacing with peripherals:
  - PO
  - P1
  - P2
  - P3
- Registers useful for Interrupt handling:
  - IP
  - IE

SFR

- PCON: Used for controlling power consumption
- SCON & SBUF: Used for serial communication
- Data Pointer Register (DPH & DPL): This is an imaginary register and actually does not exist. It's a combination of DPH and DPL registers. When a 16-bit data needs to be stored, DPTR can be used as a register and the instructions used for this are as:
  - MOV DPTR, #DATA ;{ will move the 16 bit data to DPTR}
  - INC DPTR; { Increment DPTR by 1}
  - Example:
    - MOV DPTR, #2312 { after execution DPH will have 23H and DPL will have 12H}

SFR

- Stack pointer: The stack is accessed by the stack pointer register. This is a 8-bit register. The instructions are POP, PUSH, MOV SP, #XX, where XX is the address where stack pointer should point.
- Program Counter: The purpose of PC is to keep a book mark; i.e the address of the next instruction to be executed will be kept in PC.

Advanced RISC Machine (ARM) Microcontroller

- ARM micro-controller was introduced by Acron computer organization.
- Manufactured by Apple, Nvidia, Qualcomm, Motorola, ST Microelectronics, Samsung Electronics, and TI etc.
- ARM processor belongs to the family of CPUs which are based on Reduced Instruction Set Computer (RISC) and ARM microprocessor with RAM, ROM and other peripherals in one single chip.
- Example ARM micro-controller is LPC2148.
- It is based on RISC Instruction set Architecture (ISA) and also called as Advanced RISC machine.

Advanced RISC Machine (ARM) Microcontroller

- It is most popular micro controller and most industries use it for embedded systems as it provides large set of features and is good to produce devices with excellent appearances.
- The architecture of **ARM processor** is created by **Advanced RISC Machines**, hence name ARM. This needs very few instruction sets and transistors. It has very small size. This is reason that it is perfect fit for small size devices (IoT). It has less power consumption along with reduced complexity in its circuits.

ARM Evolution

- The first version supported 26-bit addressing scheme with no support being offered for multiply operation (1983 – 1985).
- The second and third version include 32 bit addressing with multiply operation and coprocessor. (The low-end microcontrollers support only add and subtract operations)
- The fourth version was enhanced with the availability of signed instructions and the version 4T-Thumb, which is a 16-bit compressed form of instruction , was introduced.
- The functionality of the Thumb instruction set is a subset of the functionality of the 32-bit ARM instruction set. The Thumb instruction set:
  - imposes some limitations on register access
  - does not allow conditional execution except for branch instructions
  - does not allow access to the barrel shifter except as a separate instruction.

ARM Features

- **Multiprocessing Systems –** ARM processors are designed so that they can be used in cases of multiprocessing systems where more than one processors are used to process information.

- **Tightly Coupled Memory –** Memory of ARM processors is tightly coupled. This has very fast response time. It has low latency (quick response) that can also be used in cases of cache memory being unpredictable.

ARM Features

- **Memory Management –** ARM processor has management section. This includes Memory Management Unit and Memory Protection Unit. These management systems become very important in managing memory efficiently.

- **Thumb-2 Technology –** Thumb-2 Technology was introduced in 2003 and was used to create variable length instruction set. It extends 16-bit instructions of initial Thumb technology to 32-bit instructions.

ARM Features

- **One cycle execution time –** ARM processor is optimized for each instruction on CPU. Each instruction is of fixed length that allows time for fetching future instructions before executing present instruction. ARM has CPI (Clock Per Instruction) of one cycle.
- **Pipelining –** Processing of instructions is done in parallel using pipelines. Instructions are broken down and decoded in one pipeline stage. The pipeline advances one step at a time to increase throughput (rate of processing).

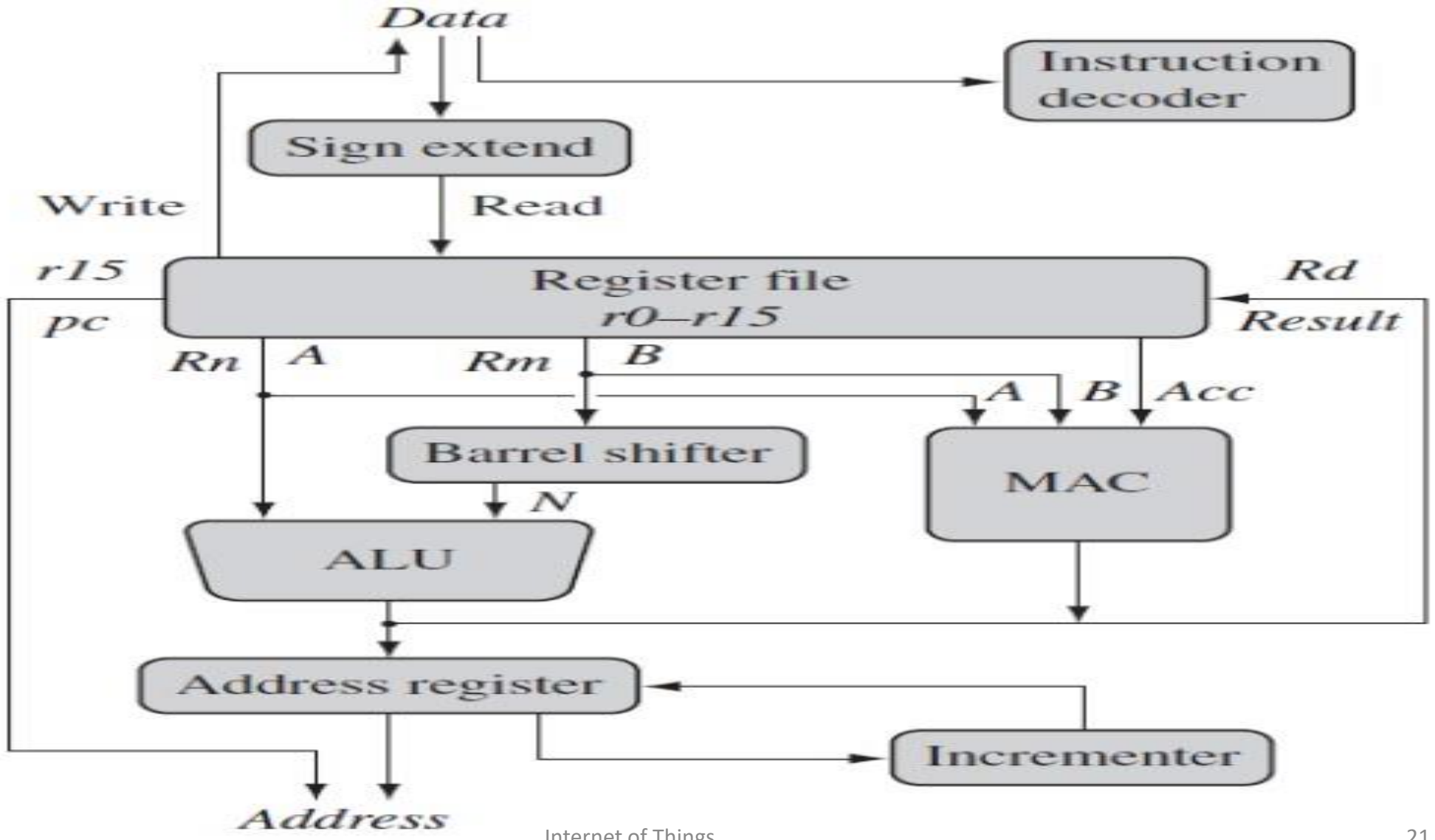
ARM Features

- **Large number of registers –** Large number of registers are used in ARM processor to prevent large amount of memory interactions. Registers contain data and addresses. These act as local memory store for all operations.

Difference between 8051 and ARM

Sl No	8051 Microcontroller	ARM Microcontroller
	8 bit for standard core bus width is present in 8051 micro-controller.	Mostly 32 bit bus width is present in ARM micro-controller and also 64-bit is available.
	Its speed is 12 clock cycles per machine cycle.	Its speed is 1 clock cycle per machine cycle.
	UART, USART, I2C, SPI, communication protocols are used.	UART, USART, Ethernet, I2S, DSP, SPI, CAN, LIN, I2C communication protocols are used.
	Flash, ROM, SRAM memory is used in 8051 micro-controller.	Flash, EEPROM, SDRAM memory is used in ARM micro-controller.
	It is based on CISC Instruction set Architecture.	It is based on RISC Instruction Set Architecture.
	8051 micro-controller is a Harvard-based architecture, but it allows us to connect external memory and simulate von Neumann's architecture.	PIC micro-controller is based on Havard architecture.
	Power consumption of 8051 micro-controller is average.	Power consumption of ARM micro-controller is low.

ARM Organization Core Data Flow Model



ARM Organization Core Data Flow Model

- The purpose of each building block in the data flow model is:
  - Instruction Decoder
  - Load and Store Architecture
- Instruction Decoder:
  - It decodes the instruction before execution.
  - There are three types of instruction sets supported in ARM core.
  - ARM instruction set
  - Jazelle instruction set
  - Thumb instruction set
  -

ARM Organization Core Data Flow Model: Load and store architecture:

- ARM falls in the RISC category, which follows the load and store architecture.

Internet of Things: Lect #8

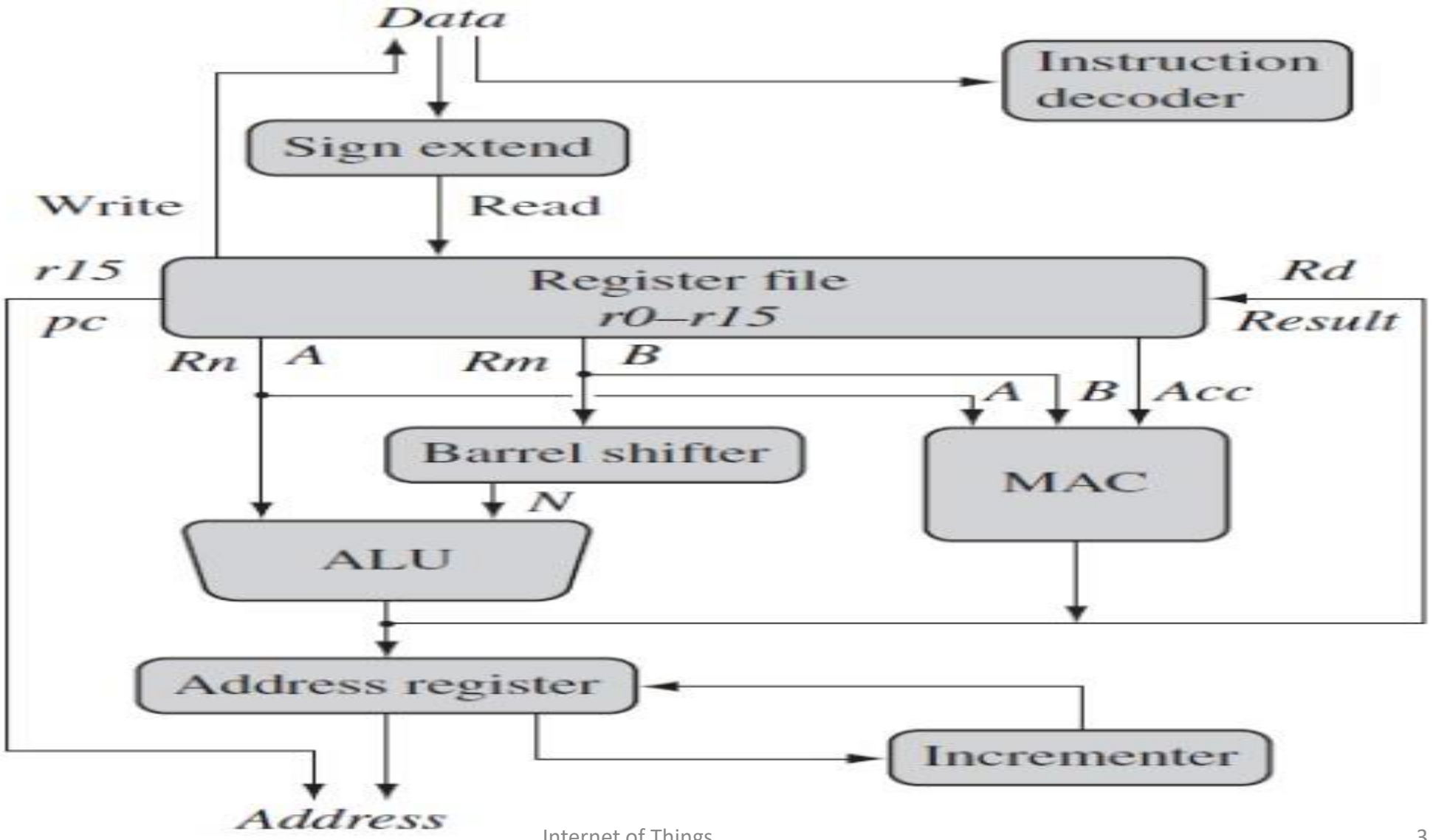
Prof. Suchismita Chinara

Dept. of Computer Science Engg.

National Institute of Technology Rourkela-769008

ARM Microcontroller

ARM Organization Core Data Flow Model



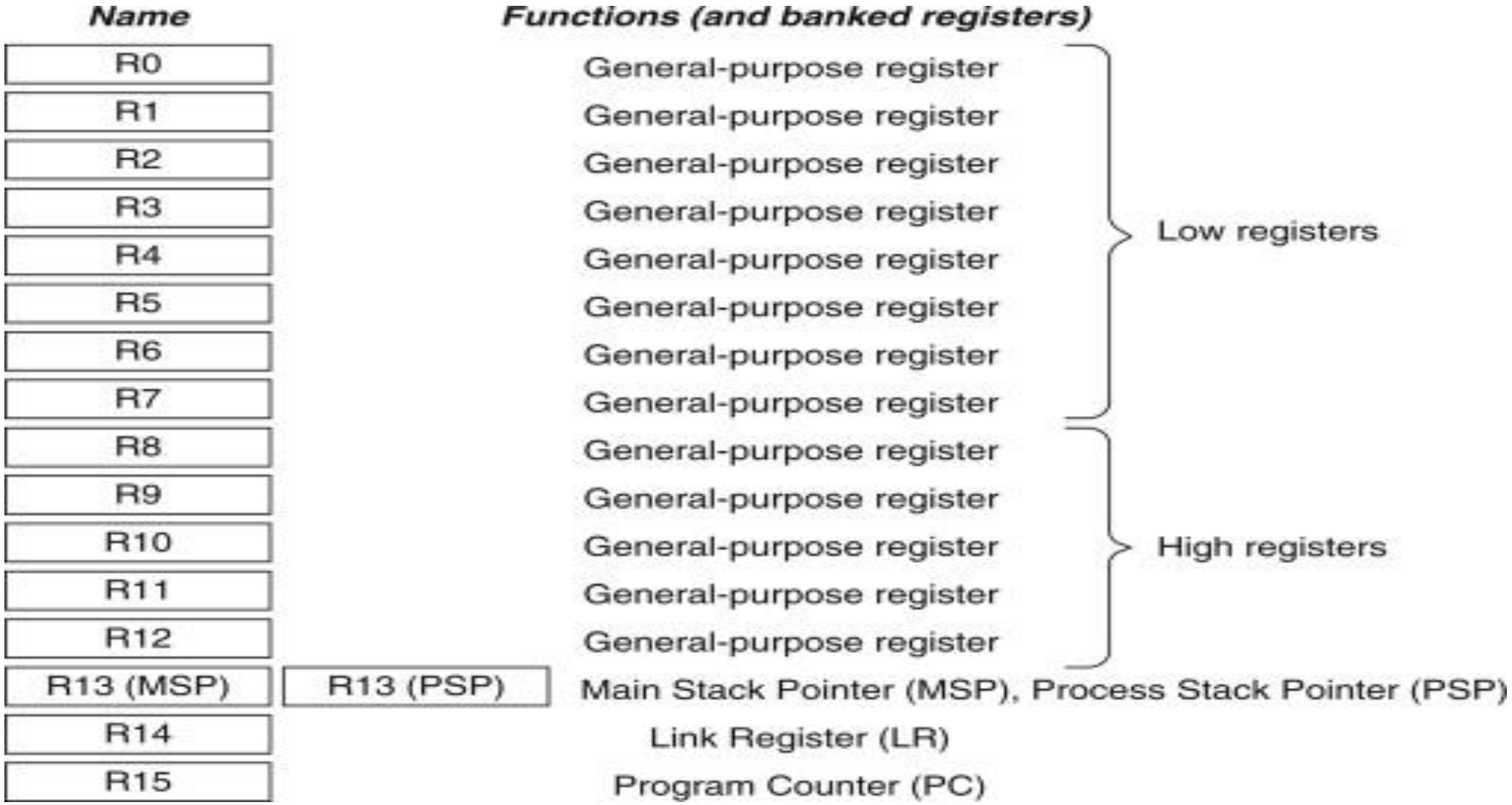
ARM Organization Core Data Flow Model

- The purpose of each building block in the data flow model is:
  - Instruction Decoder
  - Load and Store Architecture
- Instruction Decoder:
  - It decodes the instruction before execution.
  - There are three types of instruction sets supported in ARM core.
    - ARM instruction set
    - Jazelle instruction set
    - Thumb instruction set

Load and store architecture

- ARM falls in the RISC category, which follows the load and store architecture. No operation is possible in ARM architecture without the registers.
- ARM has two source registers R_m and R_n, and one destination register R_d that carries the result. A and B are the two buses that help in reading the source operands. R_m and R_n values will be fetched from buses A and B and the computation is carried out in ALU or multiply accumulator (MAC).
- Barrel shifter is a kind of support that is very useful in association with ALU for expression evaluation and address calculation.
- After going through the steps and sequences, the result will be moved to the register R_d.

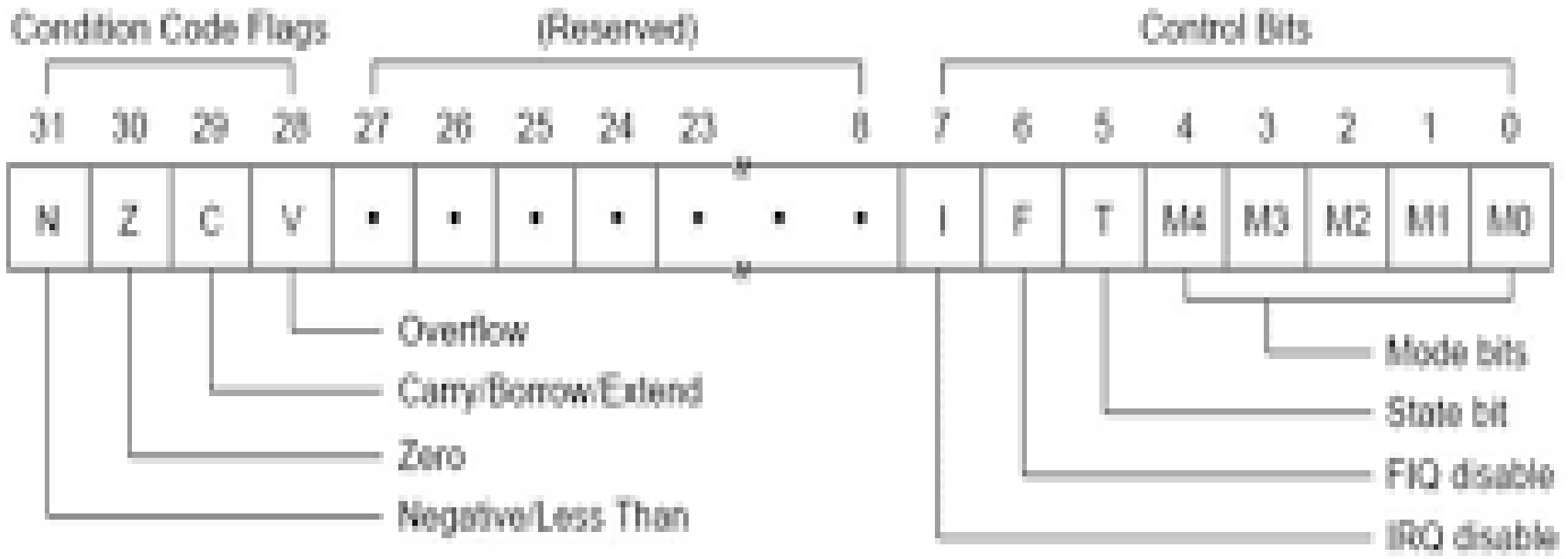
ARM Register organisation



ARM Register organisation

- There are 13 GPRs (R0 – R12)
- All the registers are 32 bits in size
- SFR: Stack Pointer (SP, R13)
- SFR: Link Register (LR, R14): When a subroutine is called, the return address will be stored in this register. When the subroutine execution is over, this register will be used for fetching the address from where the control of execution was initiated.
- SFR: Program Counter (PC, R15)

Current Program Status Register (CPSR)



Operating Modes of ARM

- User mode is the usual ARM program execution state, and is used for executing most application programs.
- *Fast Interrupt* (FIQ) mode supports a data transfer or channel process.
- *Interrupt* (IRQ) mode is used for general-purpose interrupt handling.
- Supervisor mode is a protected mode for the operating system.
- Abort mode is entered after a data or instruction Prefetch Abort.
- System mode is a privileged user mode for the operating system.
- Undefined mode is entered when an undefined instruction is executed.

Protocols for IoT- Addressing & Identification

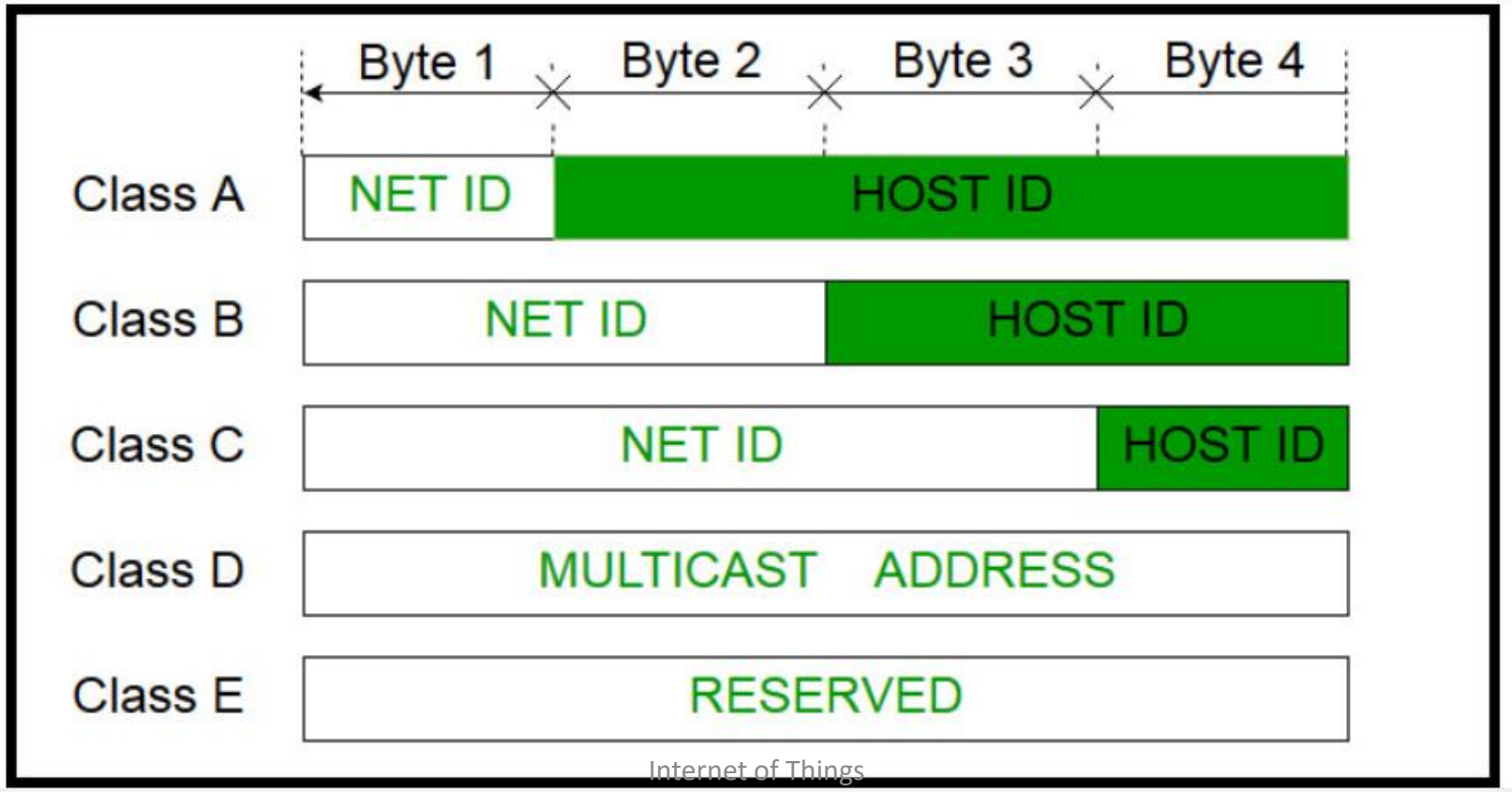
IP Address

- An Internet Protocol (IP) address is the unique identifying number assigned to every device connected to the internet.
- An IP address definition is a numeric label assigned to devices that use the internet to communicate.
- Computers that communicate over the internet or via local networks share information to a specific location using IP addresses.
- IP addresses have two distinct versions or standards.
- The Internet Protocol version 4 (IPv4) address is the older of the two, which has space for up to 4 billion IP addresses and is assigned to all computers.
- The more recent Internet Protocol version 6 (IPv6) has space for trillions of IP addresses, which accounts for the new breed of devices in addition to computers.
- There are also several types of IP addresses, including public, private, static, and dynamic IP addresses.

IPV4

- IPv4 is the fourth version of the IP.
- The protocol was first deployed on the Atlantic Packet Satellite Network (SATNET), which was a satellite network that formed a segment of the initial stages of the internet, in 1982.
- It is still used to route most internet traffic despite the existence of IPv6.
- IPv4 is currently assigned to all computers.
- An IPv4 address uses 32-bit binary numbers to form a unique IP address.
- It takes the format of four sets of numbers, each of which ranges from 0 to 255 and represents an eight-digit binary number, separated by a period point. Ex: A.B.C.D

- IPv4 address is divided into two parts:
  - **Network ID**
  - **Host ID**
- Length of NID and HID are variable
-



IPV4 address classes

1.Class A:

- The higher order bits of the first octet are always set to 0
- Ex: 0 followed by netid. hostid
- It has total 126 networks (2 addresses 0.0.0.0 and 127.x.y.z are special addresses)
- $2^{24} – 2$(=16,777,214) host ID
- IP addresses belonging to class A ranges from 1.x.x.x – 126.x.x.x

2.Class B:

- The higher order bits of the first octet are always set to 10
- Ex: 10 followed by netid. hostid
- Total $2^{14} -2$(=16384) networks and $2^{16} -2$(=65534)hosts in each network
- IP addresses belonging to class B ranges from 128.0.x.x – 191.255.x.x.

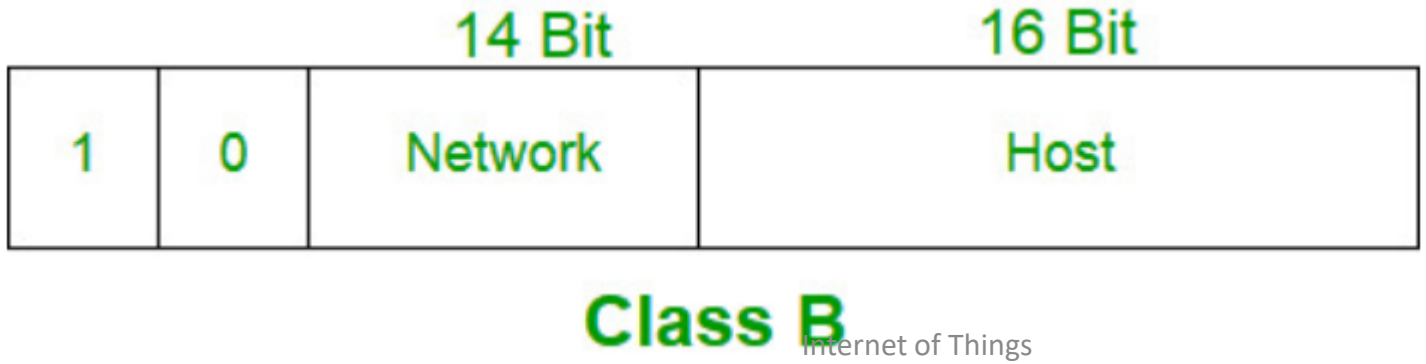
3. Class C:

- The higher order bits of the first octet always set to 110
- Ex: 110 followed by netid. hosted
- Total 2^{21} (=2097152) networks and 2^{8} -2(=254)hosts in each network
- IP addresses belonging to class C ranges from 192.0.0.x – 223.255.255.x.

4. Class D or multicast:

- IP address belonging to class D are reserved for multi-casting
- The higher order bits of the first octet of IP addresses belonging to class D are always set to 1110
- The remaining bits are for the address that interested hosts recognize.
- IP addresses belonging to class D ranges from 224.0.0.0 – 239.255.255.255.

- **Class E:**
  - IP addresses belonging to class E are reserved for experimental and research purposes
  - IP addresses of class E ranges from 240.0.0.0 – 255.255.255.254
  - The higher order bits of first octet of class E are always set to 1111.



Example of Class A and Class B addresses

Rules for assigning Host ID:

- Within any network, the host ID must be unique to that network.
- Host ID in which all bits are set to 0 cannot be assigned because this host ID is used to represent the network ID of the IP address.
- Host ID in which all bits are set to 1 cannot be assigned because this host ID is reserved as a broadcast address to send packets to all the hosts present on that particular network.

Rules for assigning Network ID:

- The network ID cannot start with 127 because 127 belongs to class A address and is reserved for internal loop-back functions.
- All bits of network ID set to 1 are reserved for use as an IP broadcast address and therefore, cannot be used.
- All bits of network ID set to 0 are used to denote a specific host on the local network and are not routed and therefore, aren’t used.

Summary of classful addresses

CLASS	LEADING BITS	NET ID BITS	HOST ID BITS	NO. OF NETWORKS	ADDRESSES PER NETWORK	START ADDRESS	END ADDRESS
CLASS A	0	8	24	2^7 (128)	2^{24} (16,777,216)	0.0.0.0	127.255.255.255
CLASS B	10	16	16	2^{14} (16,384)	2^{16} (65,536)	128.0.0.0	191.255.255.255
CLASS C	110	24	8	2^{21} (2,097,152)	2^8 (256)	192.0.0.0	223.255.255.255
CLASS D	1110	NOT DEFINED	NOT DEFINED	NOT DEFINED	NOT DEFINED	224.0.0.0	239.255.255.255
CLASS E	1111	NOT DEFINED	NOT DEFINED	NOT DEFINED	NOT DEFINED	240.0.0.0	255.255.255.255

Special Addresses

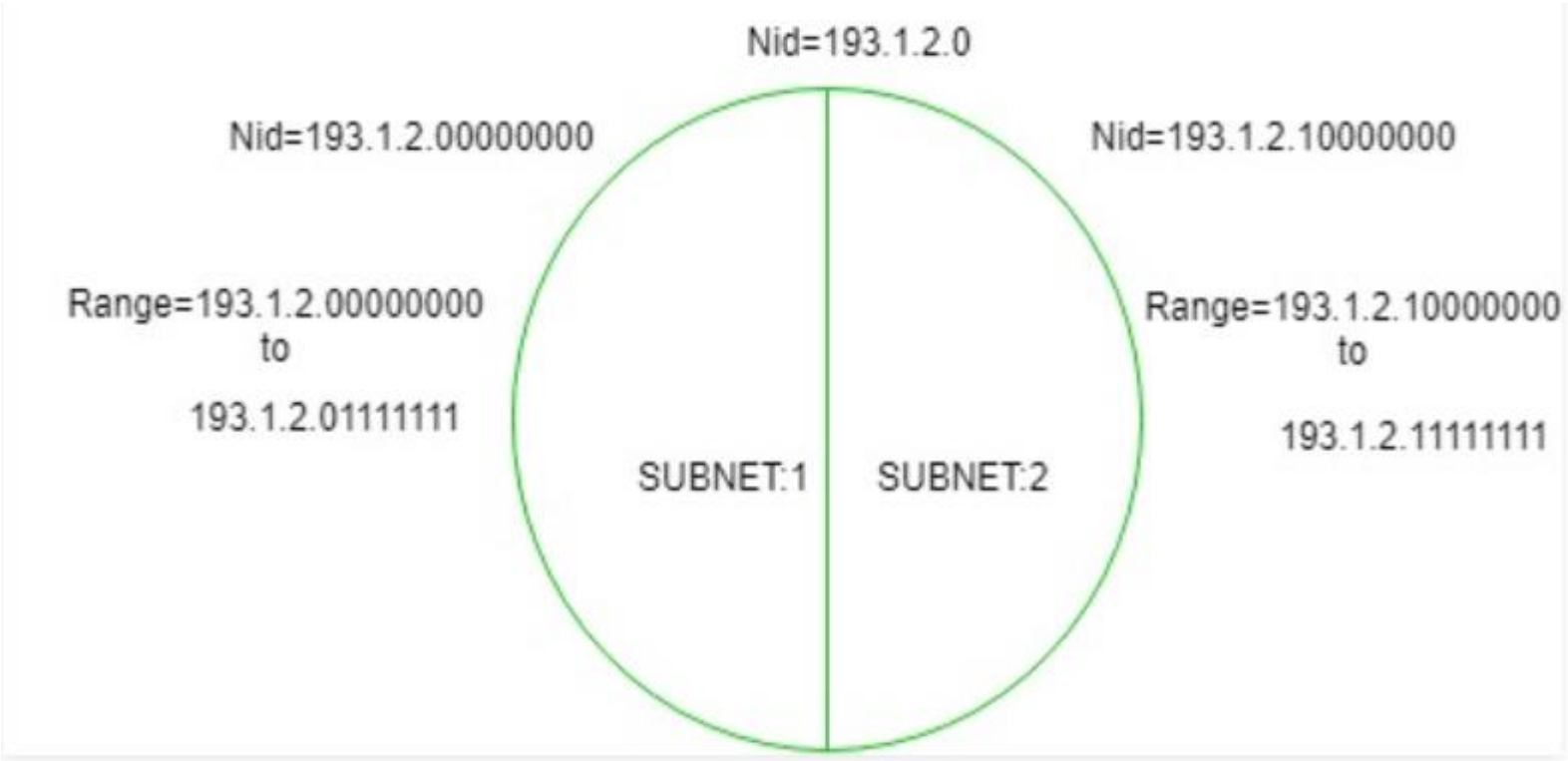
- Network address
  - Netid. Hostid (All 0’s)
  - Ex: Class A: X.0.0.0 (Ex: 44.0.0.0)
  - Class B: X.Y. 0.0 (Ex: 130.55.0.0)
  - Class C: X.Y.Z.0 (Ex: 192.168.40.0) 192.168.40.1 ----- 192.168.40.254
- Direct broadcast address
  - Netid. Hostid (All 1’s)
  - Ex: Class A: X.255.255.255
  - Class B: X.Y. 255.255
  - Class C: X.Y.Z.255

Special addresses contd..

- This host on this network
  - Used as a source address (during bootstrap)
  - Ex: 0.0.0.0
- Specific host on this network
  - NetID remains same
  - The data does not go through the router
  - Src and Dest remains in the same network
  - Ex: src 144.14.x.y sends packet to dest 144.14.p.q
- Loop back address
  - Local Loopback Address is used to let a system send a message to itself to make sure that TCP/IP stack is installed correctly on the machine.
  - Typically 127.0.0.1 is used as the local loopback address.

Subnetting

- When a bigger network is divided into smaller networks, then that is known as Subnetting.



Subnetting Contd..

- so to divide a network into two parts, you need to choose one bit for each Subnet from the host ID part.
- **For Subnet-1:**
  The first bit which is chosen from the host id part is zero and the range will be from (193.1.2.00000000 till you get all 1's in the host ID part i.e, 193.1.2.01111111) except for the first bit which is chosen zero for subnet id part.
- Thus, the range of subnet-1:
  - 193.1.2.0 to 193.1.2.127

- **For Subnet-2:**
  The first bit chosen from the host id part is one and the range will be from (193.1.2.100000000 till you get all 1's in the host ID part i.e, 193.1.2.11111111).
- Thus, the range of subnet-2:
  - 193.1.2.128 to 193.1.2.255

- **Note:**

1. To divide a network into four (2^2) parts you need to choose two bits from host id part for each subnet i.e, (00, 01, 10, 11).
2. To divide a network into eight (2^3) parts you need to choose three bits from host id part for each subnet i.e, (000, 001, 010, 011, 100, 101, 110, 111) and so on.

Internet of Things: Addressing & Identification
Lect #9

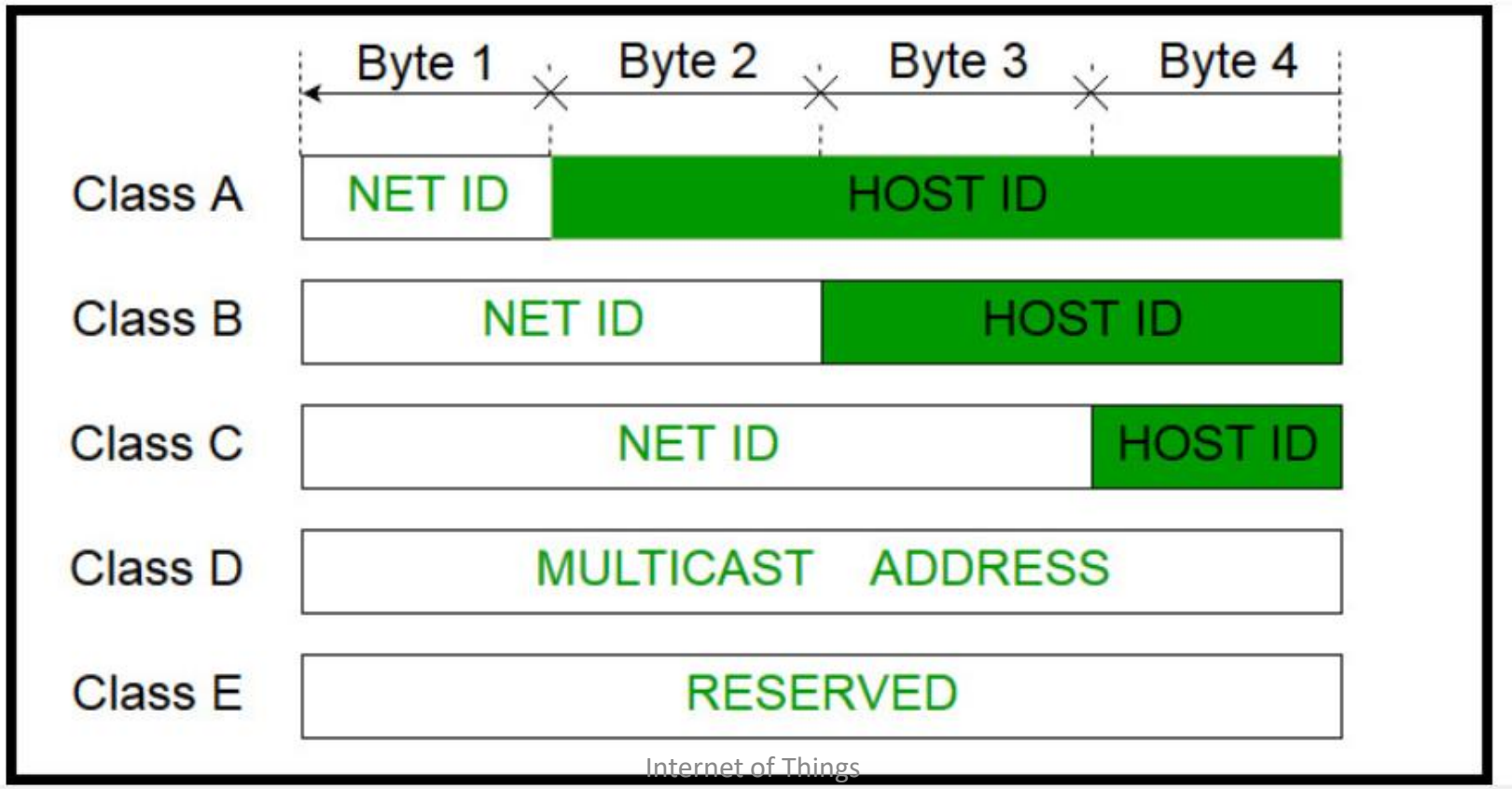
Prof. Suchismita Chinara

Dept. of Computer Science Engg.

National Institute of Technology Rourkela-769008

IPV4 addressing

- IPv4 address is divided into two parts:
  - **Network ID**
  - **Host ID**
- Length of NID and HID are variable
-



Special Addresses

- Network address
  - Netid. Hostid (All 0’s)
  - Ex: Class A: X.0.0.0 (Ex: 44.0.0.0)
  - Class B: X.Y. 0.0 (Ex: 130.55.0.0)
  - Class C: X.Y.Z.0 (Ex: 192.168.40.0) 192.168.40.1 ----- 192.168.40.254
- Direct broadcast address
  - Netid. Hostid (All 1’s)
  - Ex: Class A: X.255.255.255
  - Class B: X.Y. 255.255
  - Class C: X.Y.Z.255

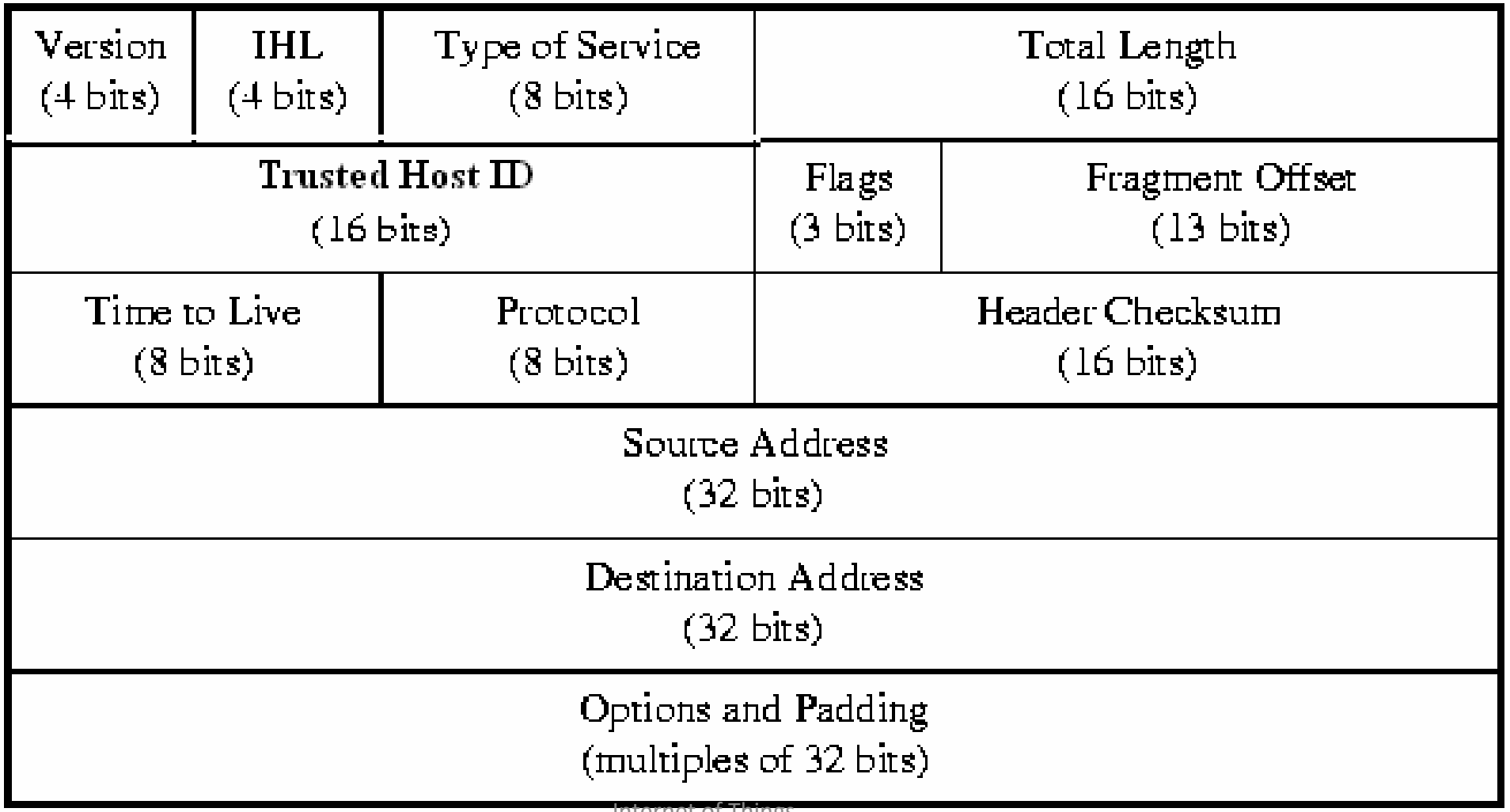
Special addresses

- This host on this network
  - Used as a source address (during bootstrap)
  - Ex: 0.0.0.0
- Specific host on this network
  - NetID remains same
  - The data does not go through the router
  - Src and Dest remains in the same network
  - Ex: src 144.14.x.y sends packet to dest 144.14.p.q
- Loop back address
  - Local Loopback Address is used to let a system send a message to itself to make sure that TCP/IP stack is installed correctly on the machine.
  - Typically 127.0.0.1 is used as the local loopback address.

IPV4 Packet header

- Length of IP Header – 20 bytes – 60 bytes (40 bytes optional)

IP Header Format



IP Header details

- **Version** − Version no. of Internet Protocol used (e.g. IPv4).
- **IHL** − Internet Header Length HLEN; Length of entire IP header.

Ex: for 20 bytes IP header HLEN= 0101

- **ToS** – Type of Service – 03 bits (precedence fm 000 to 111) – 04 bits (DTRC) – 01 bit (reserved)
  - D : Delay to be minimized
  - T: Throughput to be maximized
  - R: Reliability to be maximized
  - C: Cost to be minimized
- **Total Length** − Length of entire IP Packet (including IP header and IP Payload). Data = TLEN - HLEN

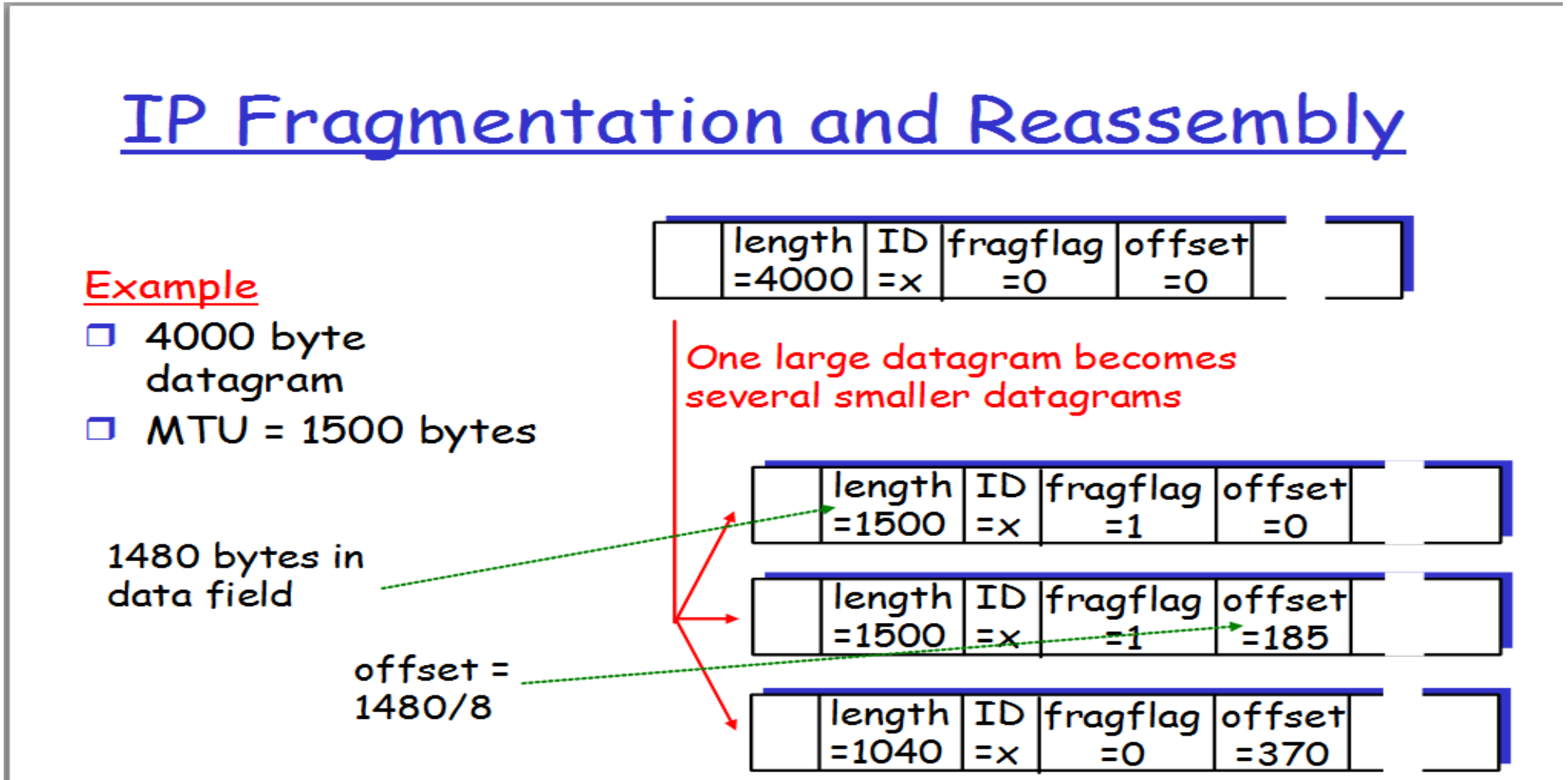
- **Identification** − If IP packet is fragmented during the transmission, all the fragments contain same identification number. to identify original IP packet they belong to.

- **Flags** − As required by the network resources, if IP Packet is too large to handle, these ‘flags’ tells if they can be fragmented or not. In this 3-bit flag, the MSB is always set to ‘0’.
  - D – Do not fragment
  - M – More fragment

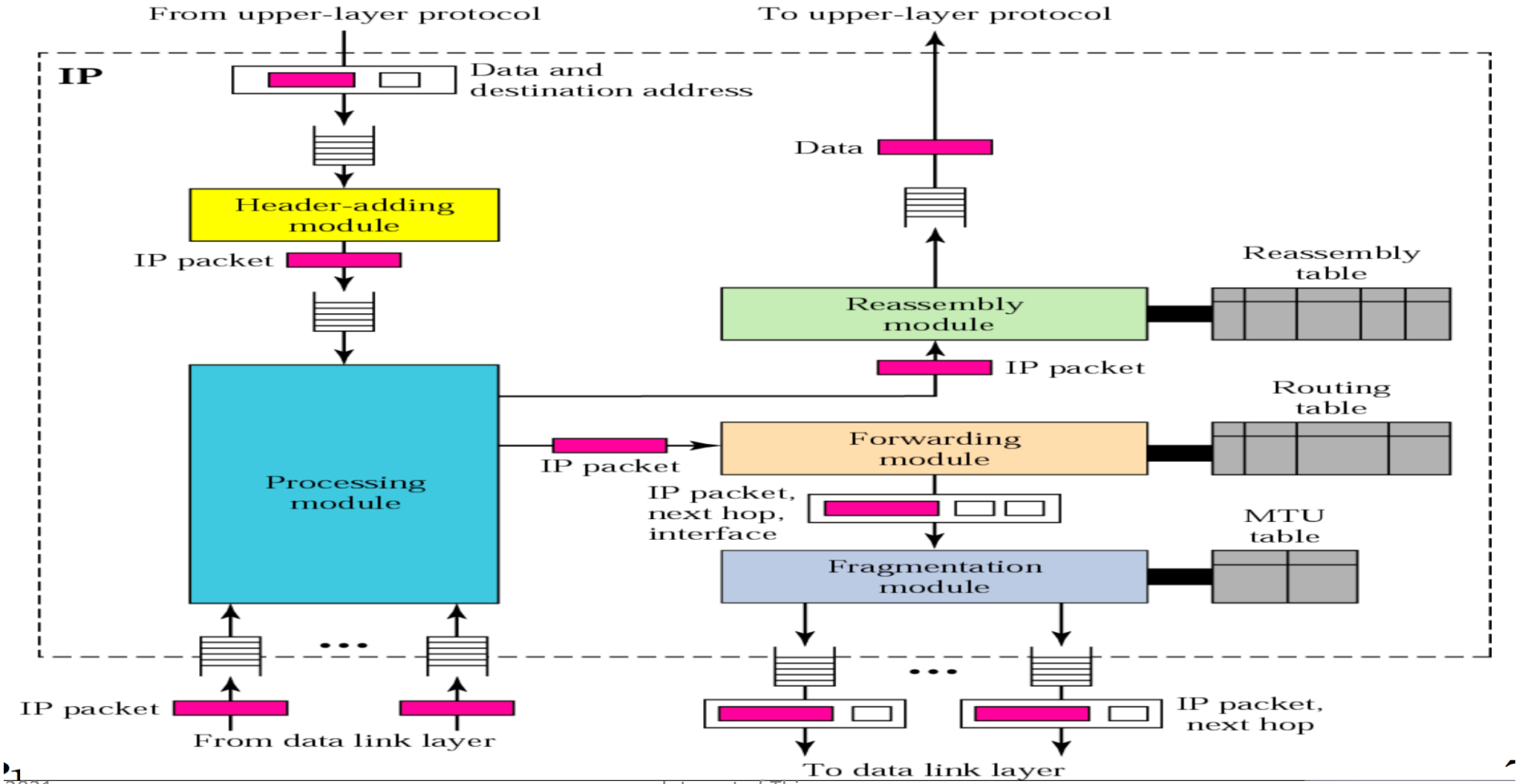
- **Fragment Offset** − This offset tells the exact position of the fragment in the original IP Packet.

- **Time to Live** − To avoid looping in the network, every packet is sent with some TTL value set, which tells the network how many routers (hops) this packet can cross. At each hop, its value is decremented by one and when the value reaches zero, the packet is discarded.
- **Protocol** − Tells the Network layer at the destination host, to which Protocol this packet belongs to, i.e. the next level Protocol. For example protocol number of ICMP is 1, TCP is 6 and UDP is 17.
- **Header Checksum** − This field is used to keep checksum value of entire header which is then used to check if the packet is received error-free.
- **Source Address** − 32-bit address of the Sender (or source) of the packet.
- **Destination Address** − 32-bit address of the Receiver (or destination) of the packet.
- **Options** − This is optional field, which is used if the value of IHL is greater than 5. These options may contain values for options such as Security, Record Route, Time Stamp, etc.

IP Fragments & Reassembly



Internet Protocol Design



The IP involves the following components:

- Header-adding module
- Processing module
- Forwarding module
- Reassembly module
- Reassembly table
- Routing table
- MTU table

Header Adding Module

- The Header Adding Module receives data from an upper layer protocol along with the destination IP address.
- It encapsulates the data in an IP datagram by adding the IP header.

Processing Module

- The processing module receives the datagram from an Interface or from the Header-adding module and does the following:

1. Remove a datagram from one of the input queues.
2. If the destination address matches loopback address or local address, Send the packet to Reassembly module.
3. If the machine is a router,
   - decrease TTL.
4. If TTL is less than or equal to 0,
   - discard the datagram.
   - else send the datagram to the forwarding module

The input queue / The output queue

- The input queues store the datagrams coming from the datalink layer or the upper layer protocols.
- The output queues store the datagrams going to the datalink layer or the upper layer protocols.
- The processing module removes datagrams from the input queues.
- The fragmentation and reassembly modules add the datagram into the output queues.

Forwarding module

- The forwarding module receives an IP packet from the processing module.
- The module finds the IP address of the next-hop along with the interface number to which the packet should be sent.

MTU Table

- The MTU table has two columns Interface number and MTU.

- The MTU table is used by the fragmentation module to find the maximum transfer unit of a particular interface.
- Fragmentation module consults the MTU table to find the MTU of the specific interface number.
- If the length of the datagram is larger than the MTU, the fragmentation module fragments the datagrams, adds a header to each fragment and sends them to the ARP package for address resolution and delivery.

Fragmentation Module

1. Extract the size of the datagram.
2. If size is greater than the MTU of the corresponding network.
3. If “D” bit is set, a discard the datagram. Else, divided the datagram into fragments, add header to each fragment. Add required options to each fragment. Send the datagram.
4. Else send the datagram.

Reassembly Module

1. If offset value is 0 and the “more fragment” bit is 0, send the datagram to the appropriate queue.
2. Else search the reassembly table for the corresponding entry.(Same D.I. and S.A.)
3. If not found, create a new entry.
4. Insert the fragment at the appropriate place in the linked list.(based on the offset)
5. If all fragments had arrived (“M”-bit and offset), reassembly the fragments, deliver the datagram to the upper layer protocol.
6. Check the T.O., if the T.O. expired discard all fragments.

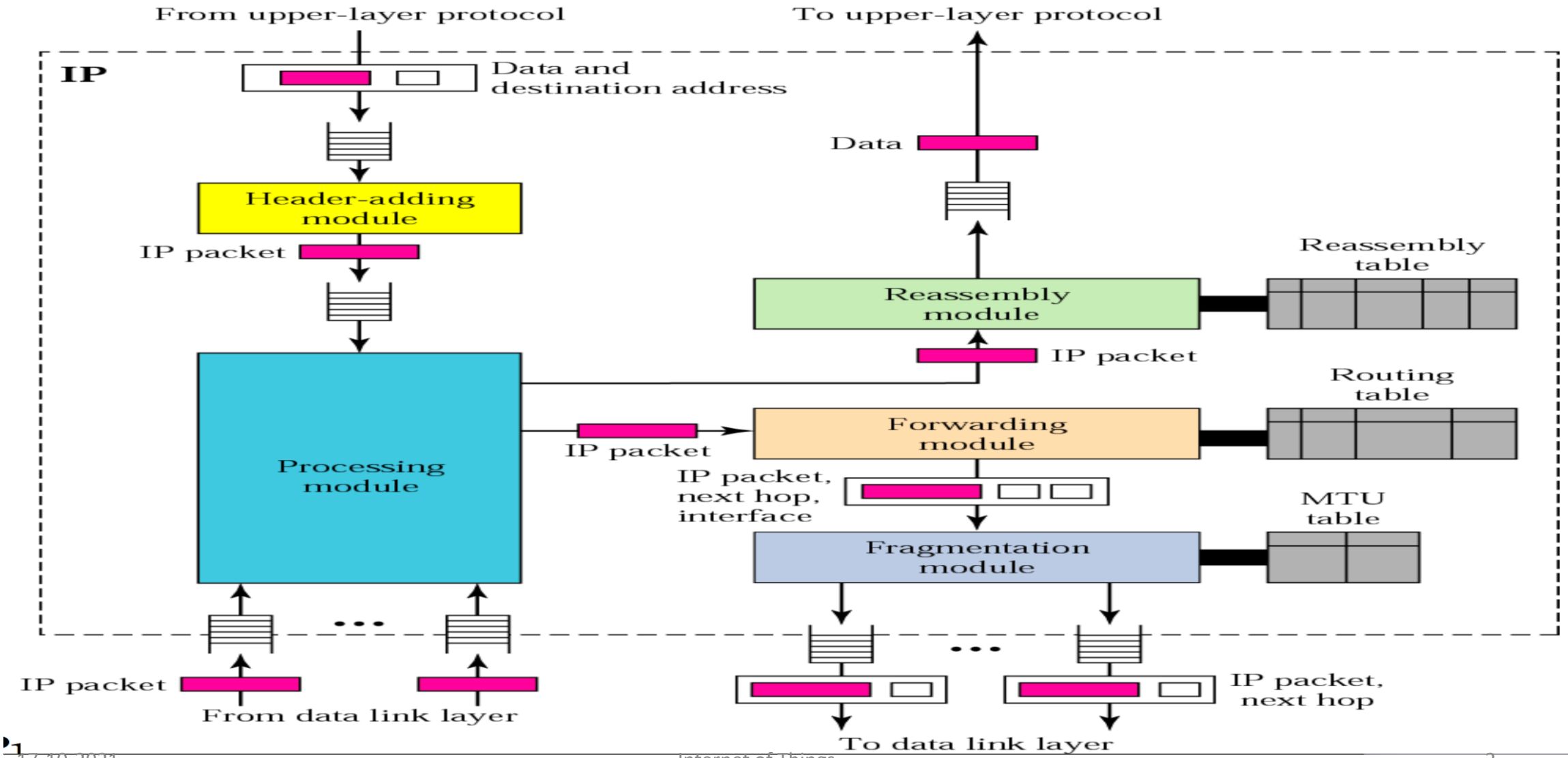
Internet of Things: Addressing & Identification
Lect #10

Prof. Suchismita Chinara

Dept. of Computer Science Engg.

National Institute of Technology Rourkela-769008

Internet Protocol Design



The IP involves the following components:

- Header-adding module
- Processing module
- Forwarding module
- Reassembly module
- Reassembly table
- Routing table
- MTU table

Header Adding Module

- The Header Adding Module receives data from an upper layer protocol along with the destination IP address.
- It encapsulates the data in an IP datagram by adding the IP header.

Processing Module

- The processing module receives the datagram from an Interface or from the Header-adding module and does the following:

1. Remove a datagram from one of the input queues.
2. If the destination address matches loopback address or local address, Send the packet to Reassembly module.
3. If the machine is a router,
   - decrease TTL.
4. If TTL is less than or equal to 0,
   - discard the datagram.
   - else send the datagram to the forwarding module

The input queue / The output queue

- The input queues store the datagrams coming from the datalink layer or the upper layer protocols.
- The output queues store the datagrams going to the datalink layer or the upper layer protocols.
- The processing module removes datagrams from the input queues.
- The fragmentation and reassembly modules add the datagram into the output queues.

Forwarding module

- The forwarding module receives an IP packet from the processing module.
- The module finds the IP address of the next-hop along with the interface number to which the packet should be sent.

MTU Table

- The MTU table has two columns Interface number and MTU.

- The MTU table is used by the fragmentation module to find the maximum transfer unit of a particular interface.
- Fragmentation module consults the MTU table to find the MTU of the specific interface number.
- If the length of the datagram is larger than the MTU, the fragmentation module fragments the datagrams, adds a header to each fragment and sends them to the ARP package for address resolution and delivery.

Fragmentation Module

1. Extract the size of the datagram.
2. If size is greater than the MTU of the corresponding network.
3. If “D” bit is set, a discard the datagram. Else, divided the datagram into fragments, add header to each fragment. Add required options to each fragment. Send the datagram.
4. Else send the datagram.

Reassembly Module

1. If offset value is 0 and the “more fragment” bit is 0, send the datagram to the appropriate queue.
2. Else search the reassembly table for the corresponding entry.(Same D.I. and S.A.)
3. If not found, create a new entry.
4. Insert the fragment at the appropriate place in the linked list.(based on the offset)
5. If all fragments had arrived (“M”-bit and offset), reassembly the fragments, deliver the datagram to the upper layer protocol.
6. Check the T.O., if the T.O. expired discard all fragments.

IPV6 Overview

- there is a growing shortage of IPv4 addresses, which are needed for all new devices added to the Internet.
- The key to IPv6 enhancement is the expansion of the IP address space from 32 bits to 128 bits, enabling virtually unlimited, unique IP addresses.
- The new IPv6 address text format is:
  xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx

Where each x is a hexadecimal digit representing 4 bits.

Simpler IP Configuration

- IPv6 provides new functions that simplify the tasks of configuring and managing the addresses on the network.
- IPv6 reduces some of the workload by automating several of the network administrator's tasks.
- The IPv6 autoconfiguration feature, for example, automatically configures interface addresses and default routes.
- In stateless autoconfiguration, IPv6 takes the Media Access Control (MAC) address of the machine and a network prefix provided by a local router and combines these two addresses to create a new, unique IPv6 address.
- This feature eliminates the need for a Dynamic Host Configuration Protocol (DHCP) server.

IPv6 address formats

- The size and format of the IPv6 address expand addressing capability.
- The IPv6 address size is 128 bits.
- The preferred IPv6 address representation is: `x:x:x:x:x:x:x:x,Where each x` is the hexadecimal values of the eight 16-bit pieces of the address.
- IPv6 addresses range from

`0000:0000:0000:0000:0000:0000:0000:0000 to`

`ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff`

IPv6 address formats

- In addition to this preferred format, IPv6 addresses might be specified in two other shortened formats:
- **Omit leading zeros**
  - Specify IPv6 addresses by omitting leading zeros. For example, IPv6 address

  `1050:0000:0000:0000:0005:0600:300c:326b` **can be written as**

  `1050:0:0:0:5:600:300c:326b`
- **Double colon**
  - Specify IPv6 addresses by using double colons **(::)** in place of a series of zeros. For example, IPv6 address

  `ff06:0:0:0:0:0:0:c3` can be written as

  `ff06::c3`. Double colons can be used only once in an IP address

Embedding IPV4 into IPV6

- An alternative format for IPv6 addresses combines the colon and dotted notation, so the IPv4 address can be embedded in the IPv6 address.
- Hexadecimal values are specified for the left-most 96 bits, and decimal values are specified for the right-most 32 bits indicating the embedded IPv4 address.
- This format ensures compatibility between IPv6 nodes and IPv4 nodes when you are working in a mixed network environment.

Embedding IPV4 into IPV6

- IPv4-mapped IPv6 address uses this alternative format.
- This type of address is used to represent IPv4 nodes as IPv6 addresses.
- It allows IPv6 applications to communicate directly with IPv4 applications.
- `For example, 0:0:0:0:0:ffff:192.1.56.10 and ::ffff:192.1.56.10 / 96 are the same.`

IPv6 address types

- IPv6 addresses are categorized into these basic types:
- **Unicast address**
- **Anycast address**
- **Multicast address**

Unicast address

- The unicast address specifies a single interface.
- A packet sent to a unicast address destination travels from one host to the destination host.
- The two regular types of unicast addresses include:
  - **Link-local address**
    - Link-local addresses are designed for use on a single local link (local network).
    - Link-local addresses are automatically configured on all interfaces.
    - The prefix used for a link-local address is `fe80::/10`.
    - Routers do not forward packets with a destination or source address containing a link-local address.
  - **Global address**
    - Global addresses are designed for use on any network.
    - The prefix used for a global address begins with binary `001`

Special unicast address

- **Unspecified address**
  - The unspecified address is `0:0:0:0:0:0:0:0`**.**
  - You can abbreviate the address with two colons **(::).**
  - The unspecified address indicates the absence of an address, and it can never be assigned to a host.
  - It can be used by an IPv6 host that does not yet have an address assigned to it.
  - For example, when the host sends a packet to discover if an address is used by another node, the host uses the unspecified address as its source address.
- **Loopback address**
  - The loopback address is `0:0:0:0:0:0:0:1`**.**
  - You can abbreviate the address as `::1`.
  - The loopback address is used by a node to send a packet to itself.

Internet of Things: Addressing & Identification
Lect #11

Prof. Suchismita Chinara

Dept. of Computer Science Engg.

National Institute of Technology Rourkela-769008

IPV6: Anycast Address

- An anycast address specifies a set of interfaces, possibly at different locations, that all share a single address.
- A packet sent to an anycast address goes only to the nearest member of the anycast group.

IPV6: Multicast Address

- The multicast address specifies a set of interfaces, possibly at multiple locations.
- The prefix used for a multicast address is FF.
- If a packet is sent to a multicast address, one copy of the packet is delivered to each member of the group.

IPV6: **Neighbour discovery**

- Neighbor discovery allows hosts and routers to communicate with one another.
- Neighbor discovery functions are used by IPv6 nodes (hosts or routers) to discover the presence of other IPv6 nodes, to determine the link-layer addresses of nodes, to find routers that are capable of forwarding IPv6 packets, and to maintain a cache of active IPv6 neighbors.

IPV6: **Neighbour discovery**

- IPv6 nodes use these five Internet Control Message Protocol version 6 (ICMPv6) messages to communicate with other nodes:
- **Router solicitation**
- **Router advertisement**
- **Neighbour solicitation**
- **Neighbour advertisement**
- **Redirect**

IPV6: **Neighbour discovery**

- **Router solicitation**
  - Hosts send these messages to request routers to generate router advertisements.
  - A host sends an initial router solicitation when the host first becomes available on the network.
- **Router advertisement**
  - Routers send these messages either periodically or in response to a router solicitation.
  - The information provided by router advertisements is used by hosts to automatically create global interfaces, and associated routes.
  - Router advertisements also contain other configuration information used by a host such as maximum transmission unit and hop limit.

IPV6: **Neighbour discovery**

- **Neighbour solicitation**
  - Nodes send these messages to determine the link-layer address of a neighbor, or to verify that a neighbor is still reachable.
- **Neighbour advertisement**
  - Nodes send these messages in response to a neighbor solicitation or as an unsolicited message to announce an address change.
- **Redirect**
  - Routers use these messages to inform hosts of a better first hop for a destination.

IPV6 Packet

- An **IPv6 packet** is the smallest message entity exchanged using Internet Protocol version 6 (IPv6).
- Packets consist of control information for addressing and routing and a payload of user data.
- The control information in IPv6 packets is subdivided into a mandatory fixed header and optional extension headers.
- The payload of an IPv6 packet is typically a datagram or segment of the higher-level transport layer protocol, but may be data for an internet layer (e.g., ICMPv6) or link layer instead.

IPV6 Packet

- IPv6 packets are typically transmitted over the link layer (i.e., over Ethernet or Wi-Fi), which encapsulates each packet in a frame. Packets may also be transported over a higher-layer protocol, such as IPv4 when using 6 to 4 transition technologies.
- In contrast to IPv4, routers do not fragment IPv6 packets larger than the maximum transmission unit (MTU), it is the sole responsibility of the originating node. A minimum MTU of 1,280 octets is mandated by IPv6, but hosts are "strongly recommended" to use Path MTU Discovery to take advantage of MTUs greater than the minimum.