SNS COLLEGE OF TECHNOLOGY

An Autonomous Institution

Coimbatore-35

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

19ITT204 - MICROCONTROLLER AND EMBEDDED SYSTEMS

II YEAR/ IV SEMESTER

UNIT III EMBEDDED SYSTEM CONCEPTS AND PROCESSORS

TOPIC – Introduction to Embedded Systems

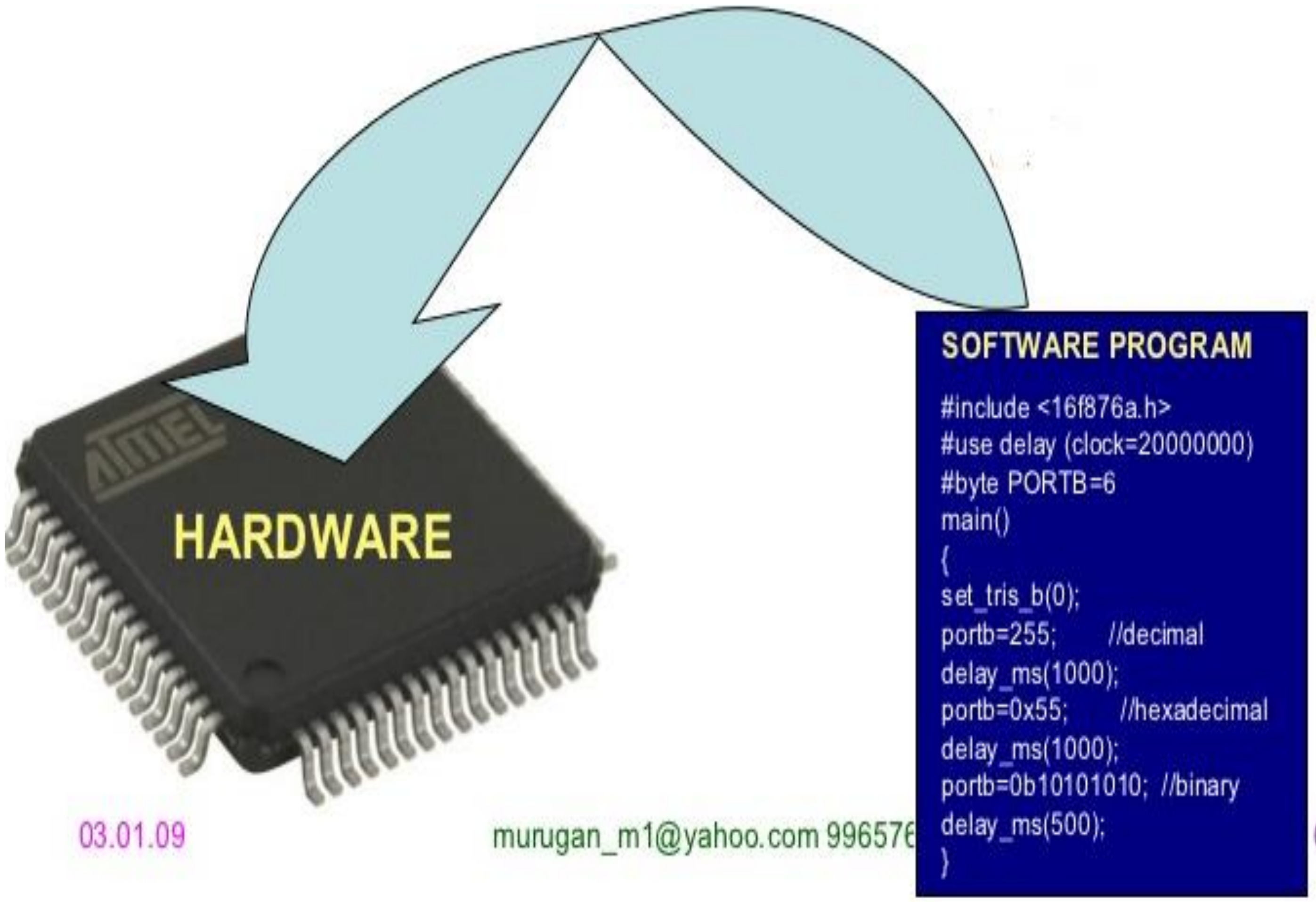
INTRODUCTION TO EMBEDDED SYSTEMS

- Introduction to Embedded Systems
- The Build process for embedded systems
- Structural units in Embedded processor
- Selection of processor
- Selection of Memory devices

INTRODUCTION TO EMBEDDED SYSTEMS

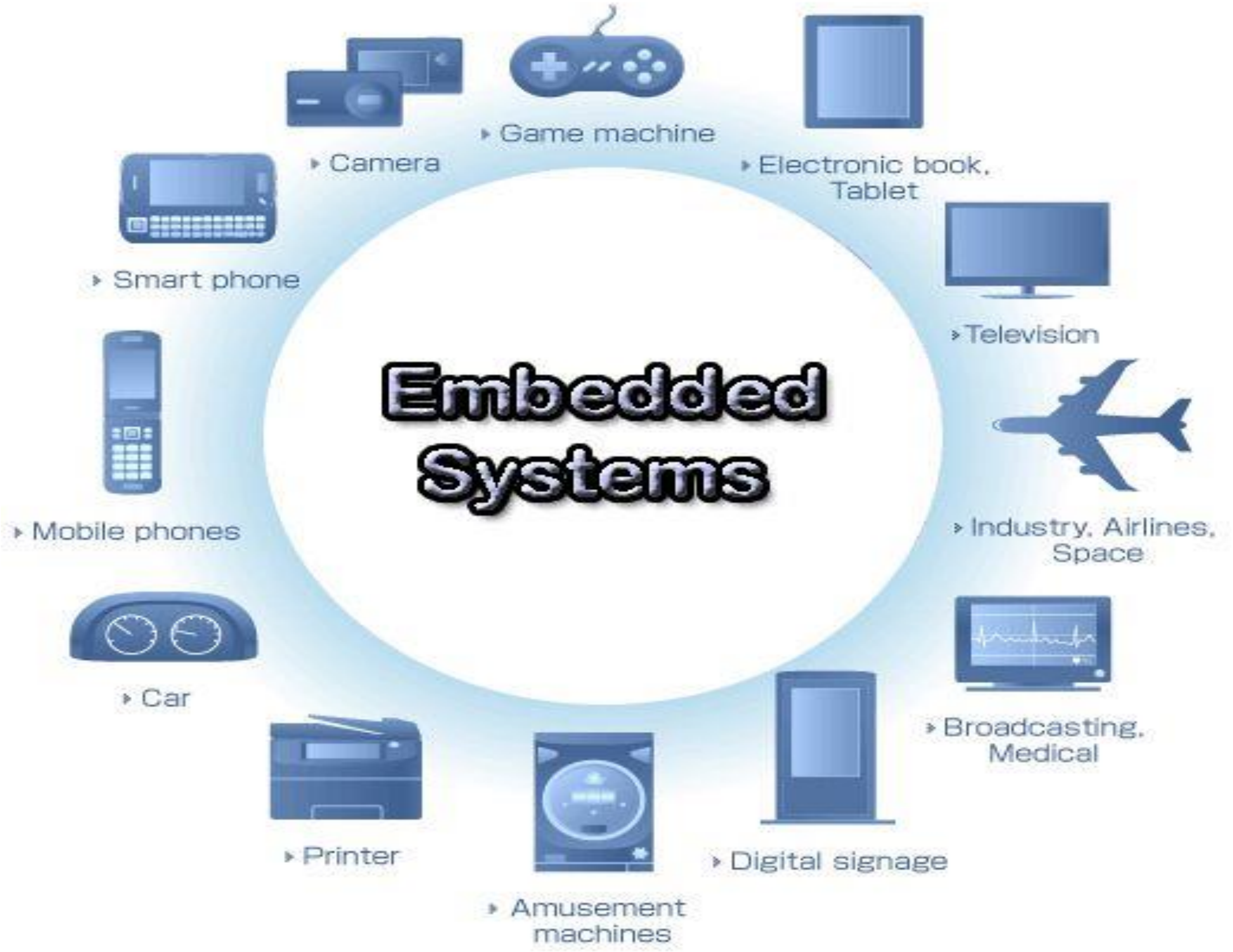
- DMA
- Memory management methods
- Timer and Counting devices
- Watchdog Timer
- Real Time Clock
- In circuit emulator
- Target Hardware Debugging.

What a Embedded System Is???

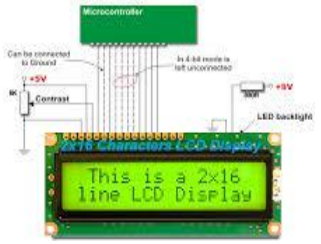


```
SOFTWARE PROGRAM
#include <16f876a.h>
#use delay (clock=20000000)
#byte PORTB=6
main()
{
set_tris_b(0);
portb=255;        //decimal
delay_ms(1000);
portb=0x55;       //hexadecimal
delay_ms(1000);
portb=0b10101010;  //binary
delay_ms(500);
}
```

Examples Embedded System



Components in Embedded System


Microcontroller
Can be connected to Ground
+5V
Contrast
In 4-bit mode is left unconnected
+5V
LED backlight
2x16 Characters LCD Display
This is a 2x16 line LCD Display

System

A system is a way of working, organizing or doing one or many tasks, which are performed in a system according to a fixed plan, program or set or rules

General Computing System Components

- *Microprocessor*
- *Large memory*
- *Input Units*
- *Out put Units*
- *Networking Units*
- *An Operating System*

Embedded System

An embedded system is a system that has ***embedded software in a computer hardware****. The system is dedicated for either an application(s) or specific part of an application or product or a component of a large system*

Eg.

- ✓ *Washing machine*
- ✓ *Cooking machine*
- ✓ Automative chocolate vending machines
- ✓ *Multitasking toys*

Classification of Embedded System

i. Small Scale Embedded Systems

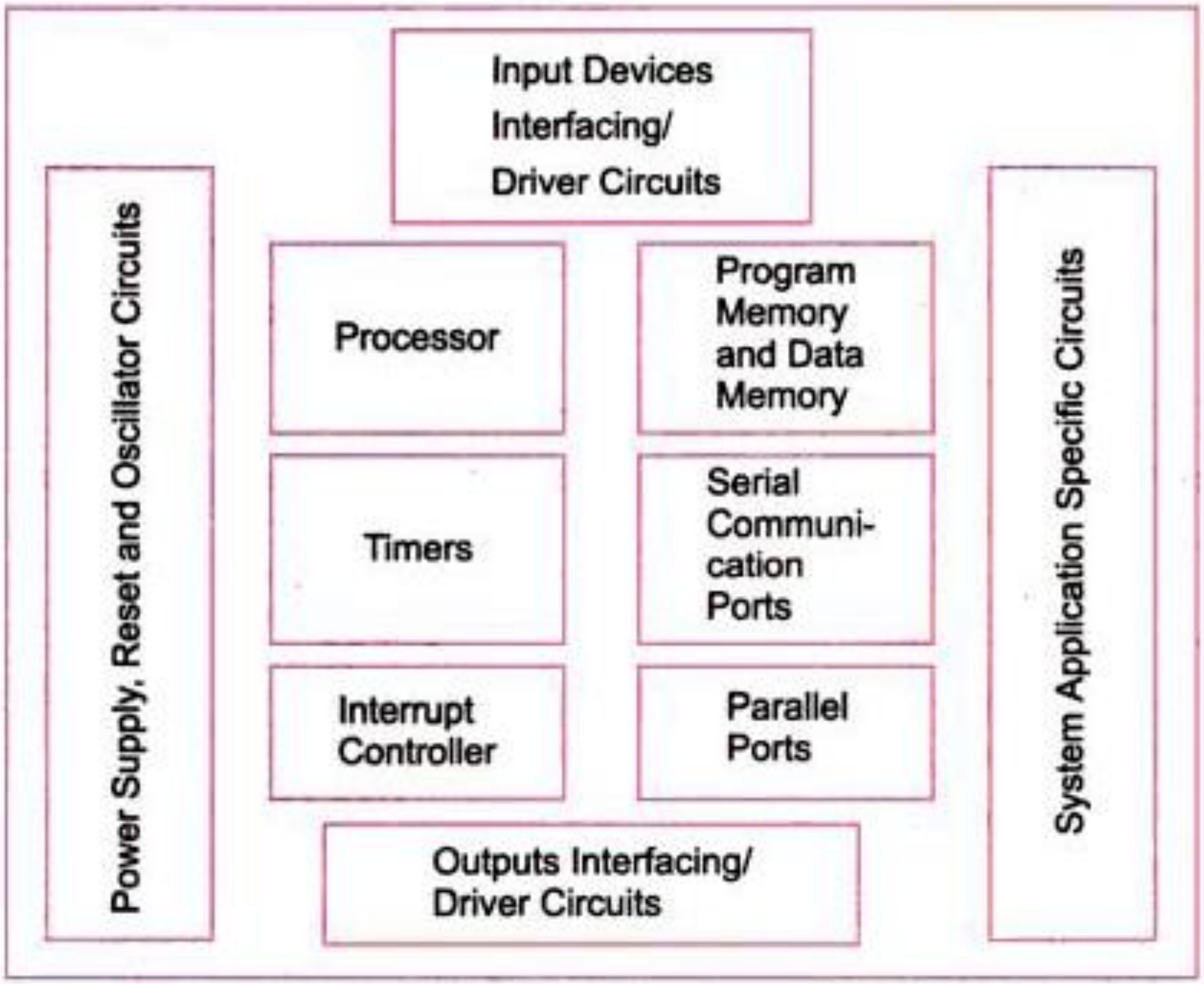
ii. Medium Scale Embedded Systems

iii. Sophisticated Embedded Systems

Embedded System Components

- *It embeds **Hardware** similar to a computer*
- ***Software** usually embeds in the ROM, flash memory or media card*
- *It embeds a Real Time Operation System (**RTOS**)*

Components of Embedded System Hardware


Input Devices Interfacing/ Driver Circuits
Power Supply, Reset and Oscillator Circuits
Processor
Program Memory and Data Memory
Timers
Serial Communi-cation Ports
Interrupt Controller
Parallel Ports
Outputs Interfacing/ Driver Circuits
System Application Specific Circuits

Building Blocks of Embedded System Hardware

- The Hardware consists of following building blocks and devices:
    - Power source
    - Clock Oscillator and Clocking Units
    - System Timer
    - Real Time Cock
    - Reset Circuit
    - Power Up Reset
    - Watchdog Timer Reset
    - Memory
    - I/O ports
    - I/O buses
    - I/O Interfaces
    - DAC
    - ADC

The Build Process

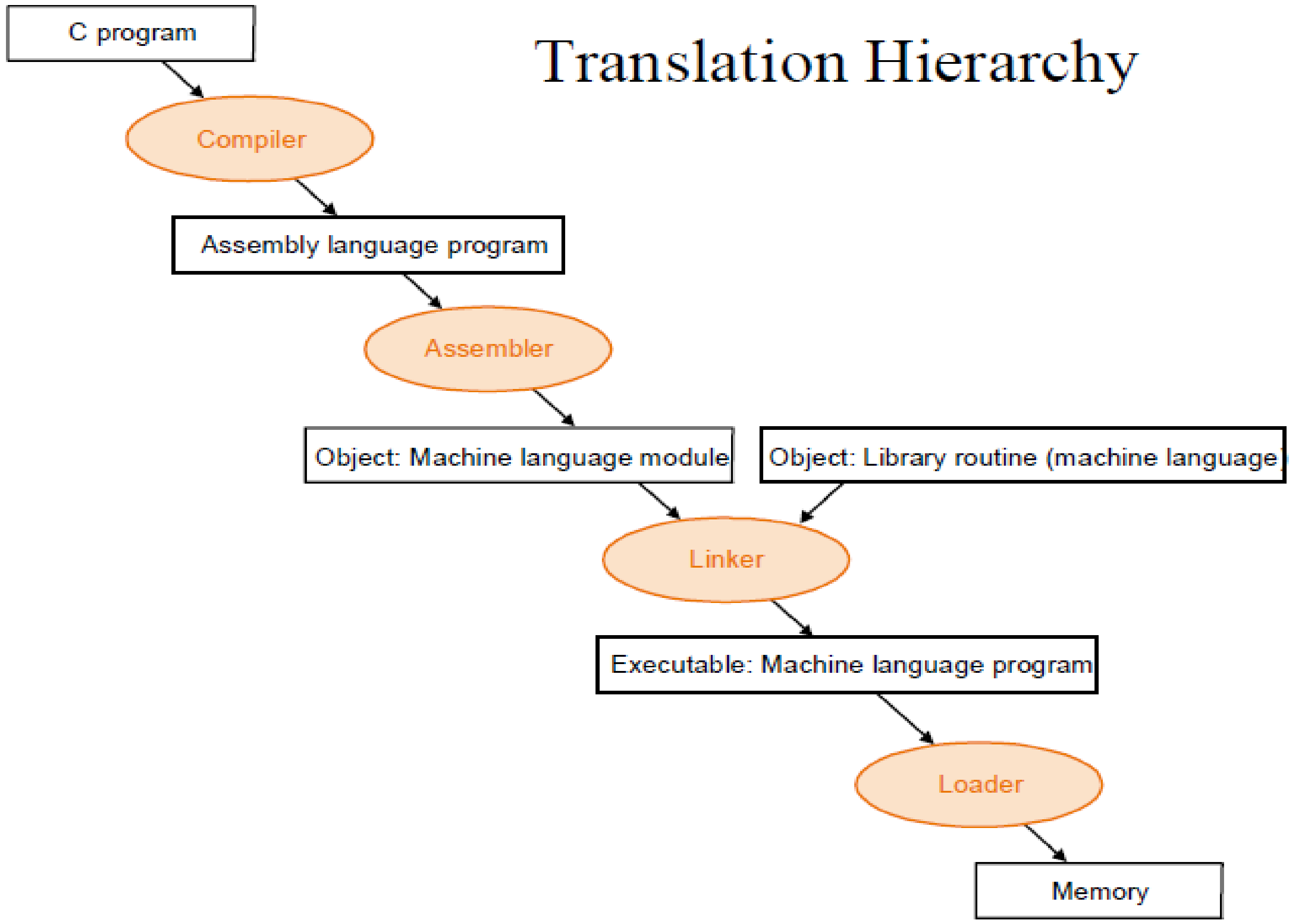
- The first step is selecting a processor
- The processor is selected from the following considerations
  - **Instruction Set**
  - **Maximum bits** in an operand in an operation
  - **Clock frequency in MHz or GHz** and the processing speed in Million Instructions per seconds
  - Processor **ability to solve the complex algorithms** used in meeting the deadlines for their processing

The Build Process

- The build process for embedding software consists of the following steps
  - Project file consisting of **source file and library files**
  - **Compilation** of the project files
  - **Linking** all object files and locating onto a single re-locatable object file
  - Converting the object file in a form called **hex-file to binary image**

The Build Process

Translation Hierarchy

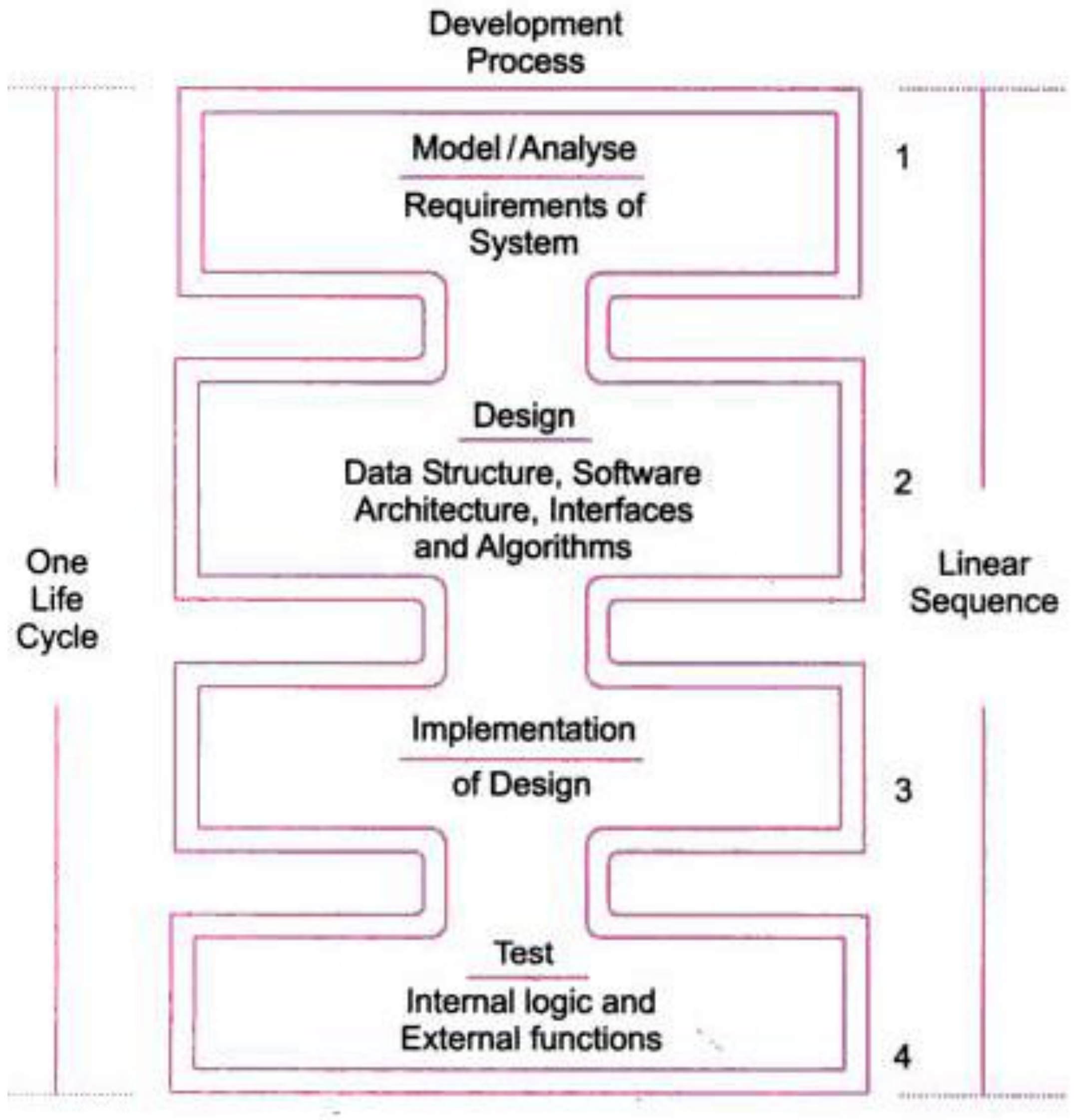

C program
Compiler
Assembly language program
Assembler
Object: Machine language module
Object: Library routine (machine language)
Linker
Executable: Machine language program
Loader
Memory

Design Process in Embedded System

- ***Concepts used during Design process***
  - Abstraction
  - Hardware and Software Architectures
  - Extra Functional Properties
  - System Related family of Designs
  - Modular Design
  - Mapping
  - User Interfaces Design
  - Refinements

Design Process in Embedded System

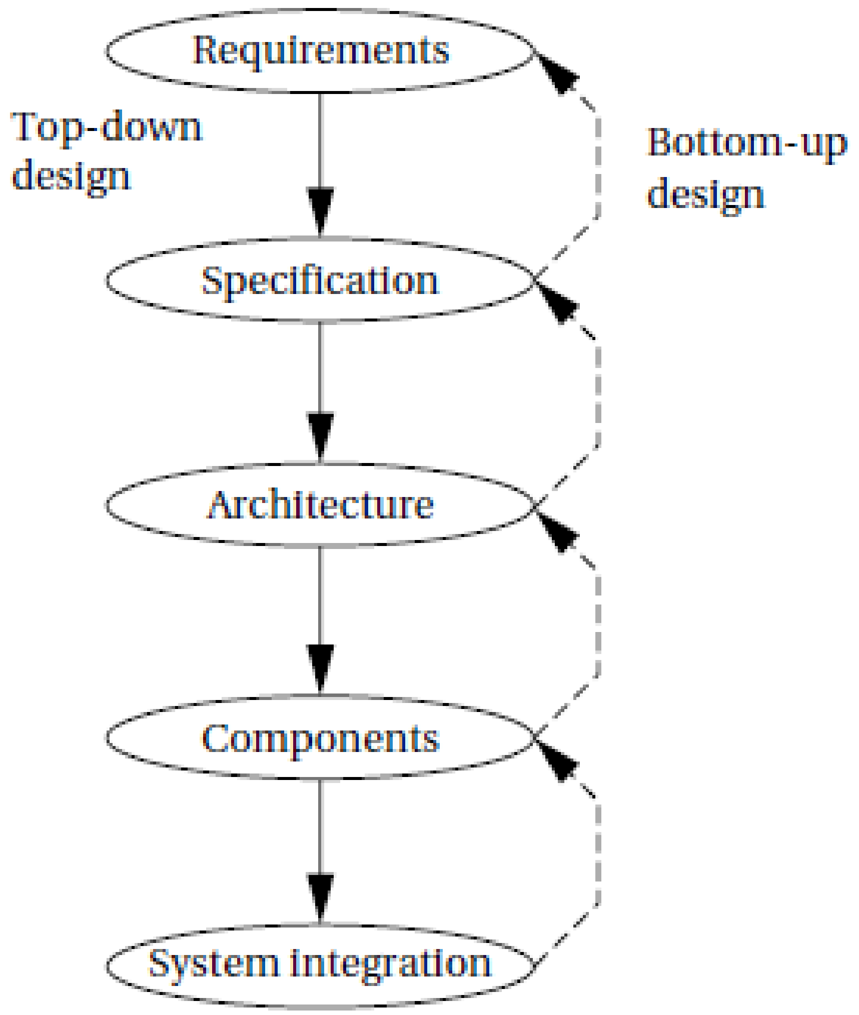
- *Software Design Process*



Design Process in Embedded System

- Design Metrics
  - Power Dissipation
  - Performance
  - Process Deadlines
  - User Interfaces
  - Size
  - Engineering Cost
  - Manufacturing Cost
  - Flexibility
  - Prototype Development Time
  - Time to Market
  - System and User Safety
  - Maintenance

Abstraction of Steps in Design Process



Structural Units in Embedded Processor

- **Internal Buses:**
  - It internally connects all the structural units inside the processor.
  - Its width can be 8,16,32 or 64 bits
- **Address Bus**
  - It is the external bus that carries the address from the MAR to the memory as well as the IO devices and the other units of the system
- **Data Bus**
  - It is an external bus that carries the data from or to the address
- **Control Bus**
  - It is an external bus to carry control signal between the processor and memory devices
- **Bus Interface Unit**
  - It is the interface unit between the processor’s internal units with the external buses
- **MAR:**
  - Memory Address Register
  - It holds the address of the byte or word to be fetched from external memories
- **MDR:**
  - It holds the byte or word fetched from/to external memory or I/O address

Structural Units in Embedded Processor

- **Program Counter**
  - PC holds the memory address of the next instruction that would be executed.
- **Stack Pointer**
  - It is a pointer for an address which corresponds to stack top in the memory
- **Instruction Register**
  - It takes sequentially the instruction codes to the execution unit of the processor
- **Instruction Decoder**
  - It decodes the instruction opcode received at the IR passes it to the processor CU
- **Instruction Queue**
  - It is the queue of instruction so that the IR does not have to wait for the next instruction after one has been carried out

Structural Units in Embedded Processor

- **Arithmetic and Logical Unit**
  - It is a unit to execute arithmetic and logical instruction according to the current instruction present in the IR
- **Floating Point Processing Unit (FPPU)**
  - A unit separate from ALU for floating point processing which is essential in processing mathematical functions fast in a microprocessor or DSP
- **Floating Point Register Set(FPRS):**
  - A register set dedicated for storing floating point number in a standard format and used by FPPU for its data and stack

Structural Units in Embedded Processor

- **Control Unit**
  - It controls all the bus activities and unit functions needed for processing
- **Memory management Unit (MMU)**
  - It manages the memory such that instruction and data are readily available for processing
- **Application Register Set**
  - It is set of on-chip registers used during processing the instruction of the **application program of the user**

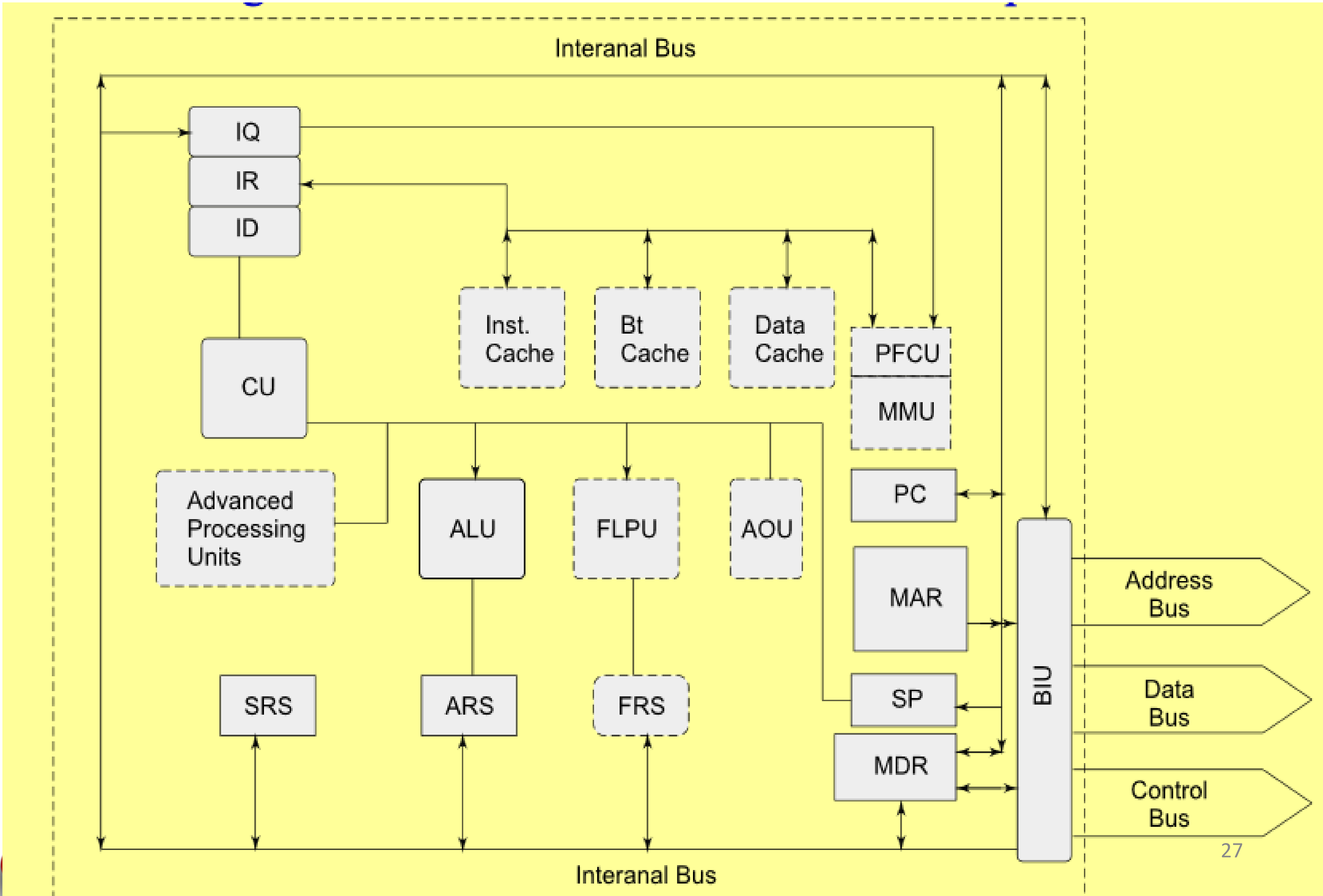
Structural Units in Embedded Processor

- **Instruction Cache**
  - It sequentially store s, like an Instruction queue, the instructions in FIFO.
- **Data Cache**
  - It stores the pre-fetched data from the external memory
  - The data cache generally holds both the key and the value together at the location
- **Pre-Fetch Control Unit**
  - It is the unit that controls the prefetching of data into the I-Cache and D-Cache in advance from the memory units
  - The instruction and data are delivered when needed to the processor's execution unit

Structural Units in Embedded Processor

- **Register Window**
  - A register window consists of a **subset of registers** with each subset storing static variables and status words of a task or program thread.
- **Advanced processing Unit**
  - These are units used for **multi stage pipeline** processing, multiline superscalar processing to **obtain processing speed higher that one instruction per clock cycle**
- **Atomic Operations Unit (AOU):**
  - It lets a user instruction when broken into number of processor instruction called atomic operation, finish before an interrupt of a process occurs.

Structural Units in Embedded Processor



Embedded System Characteristics

- System functions in real time
- Program is preloaded in the ROM(s) or flash memory
- Dedicated set of functions
- Complex dedicated purpose preprogrammed
  - algorithms
  - Hardware
  - Graphics and Other user interfaces
- Multirate operations with different predetermined time constraints to finish the different operations

Constraints of Embedded Systems

- Available System Memory
- Available Processor Speed
- Meeting Deadlines
- Performance
- Power
- Size
- Design and Manufacturing Cost

Selection of Processor

- Different systems require different processor features
- The processor is selected from the following considerations
  - **Instruction Set**
  - **Maximum bits** in an operand in an operation
  - Processing Speed
  - **Ability to solve the complex algorithms**
  - A processor gives high **computing performance** when it has
    - Pipeline and Superscalar architecture
    - Pre-fetch cache unit, caches, register files and MMU
    - RISC core architecture

Selection of Processor

- A processor with **register windows** provides fast **context switching** in a multitasking system
- Processor has **auto shut down** features for its units
- A processor with **burst mode** accesses external memories fast

Selection of Memory Devices

- Software designer coding is over and the ROM image file is ready, a hardware designer of a system is faced with the a questions, of
  - **what type of memory ?**
  - **what to use?**
  - **how much size of each, should be to used???**

Selection of Memory Devices

- Some of the selection process is,
  - Internal ROM
  - Internal EPROM
  - Internal EEPROM
  - Internal RAM
  - ROM Device
  - EPROM Device
  - EEPROM Device
  - Flash Device
  - RAM device
  - Parameterized Distributed RAM
  - Parameterized Block RAM

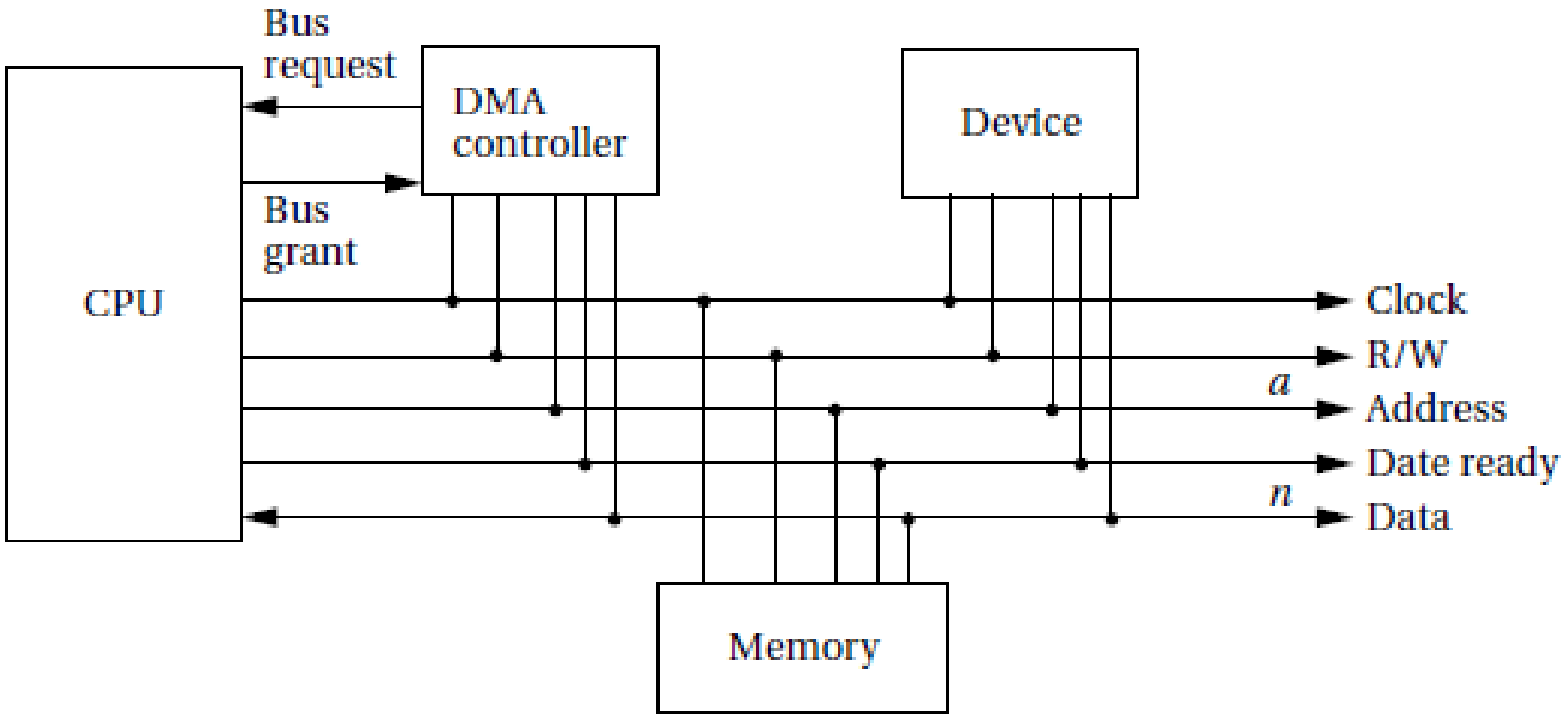
Selection of Memory Devices

- Masked **ROM** or **EPROM** stores the embedded software
- **EEPROM** is used for **testing and design stages** and also is used to store the **results during the system program run time**
- **Flash stores** the results **byte by byte** during a system run after a full sector erase
- **RAM** is mostly used in **SRAM** form in a system
- **Sophisticated** system use **RAM** in the form of **DRAM, SDRAM or RDRAM**
- **Parameterized distributed RAM** is used when the I/O devices and subunits require a memory buffer and a **fast write** by another system

DMA

- I/O devices need to transfer the data of other systems to the memory addresses in the system
- A system may also need to transfer data to the IO devices
- A Direct Memory Access is required when a block of data is to be transferred between two systems without the CPU intervening, except at the start and at the end

DMA


Bus request
Bus grant
CPU
DMA controller
Device
Memory
Clock
R/W
a
Address
Date ready
n
Data

DMA

- DMA supports 3 modes of operations
  - **Single transfer** at a time and then release of the hold on the system bus
  - **Burst transfer** at a time and then release of the hold on the system bus. A burst may be a few Kilo Bytes
  - **Bulk transfer** and then release of the hold on the system bus after the transfer is completed

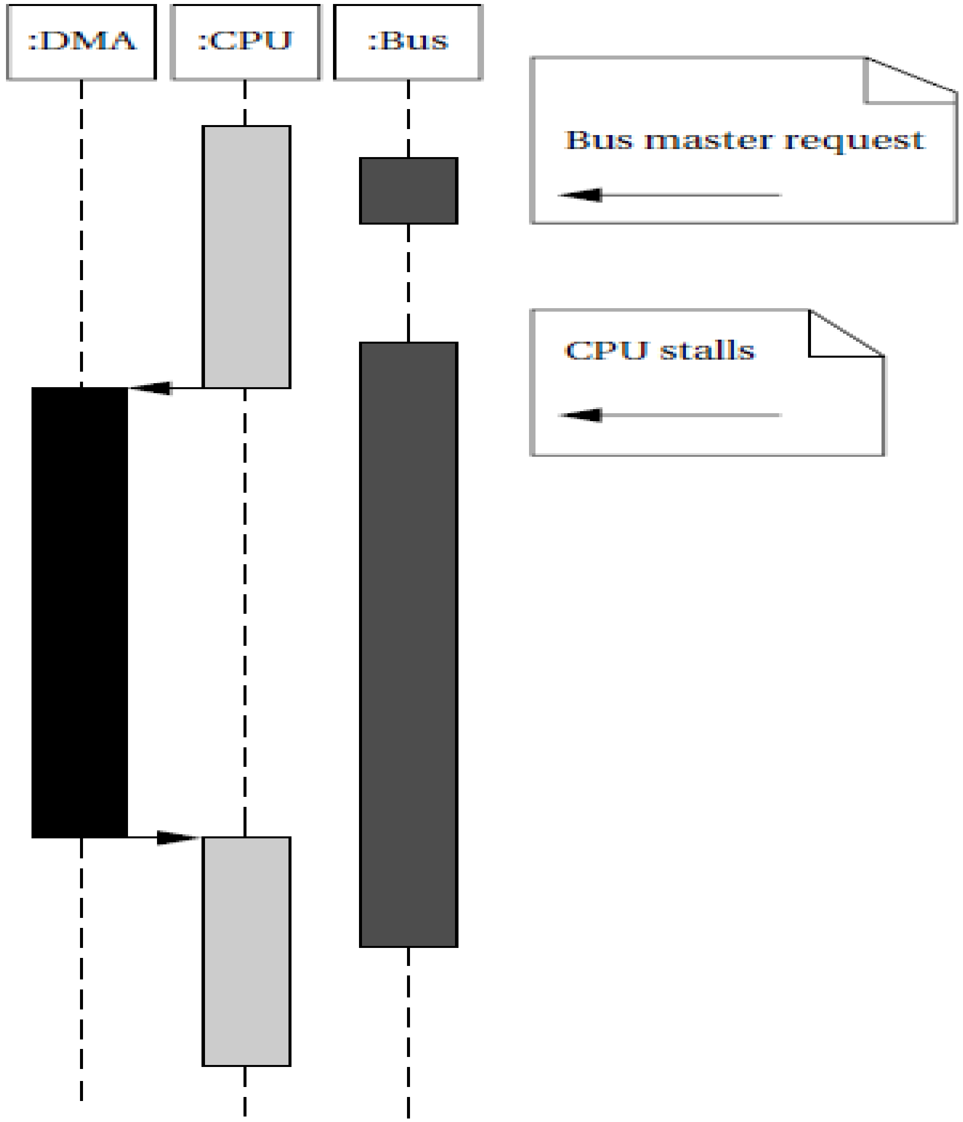
DMAC

- DMA transfer is facilitated by the DMAC (**DMA Controller**)
- Data transfer occurs efficiently between I/O devices and system memory with the least processor intervention using DMAC
- The system address and data buses become unavailable to the processor and available to the IO device that connects DMAC

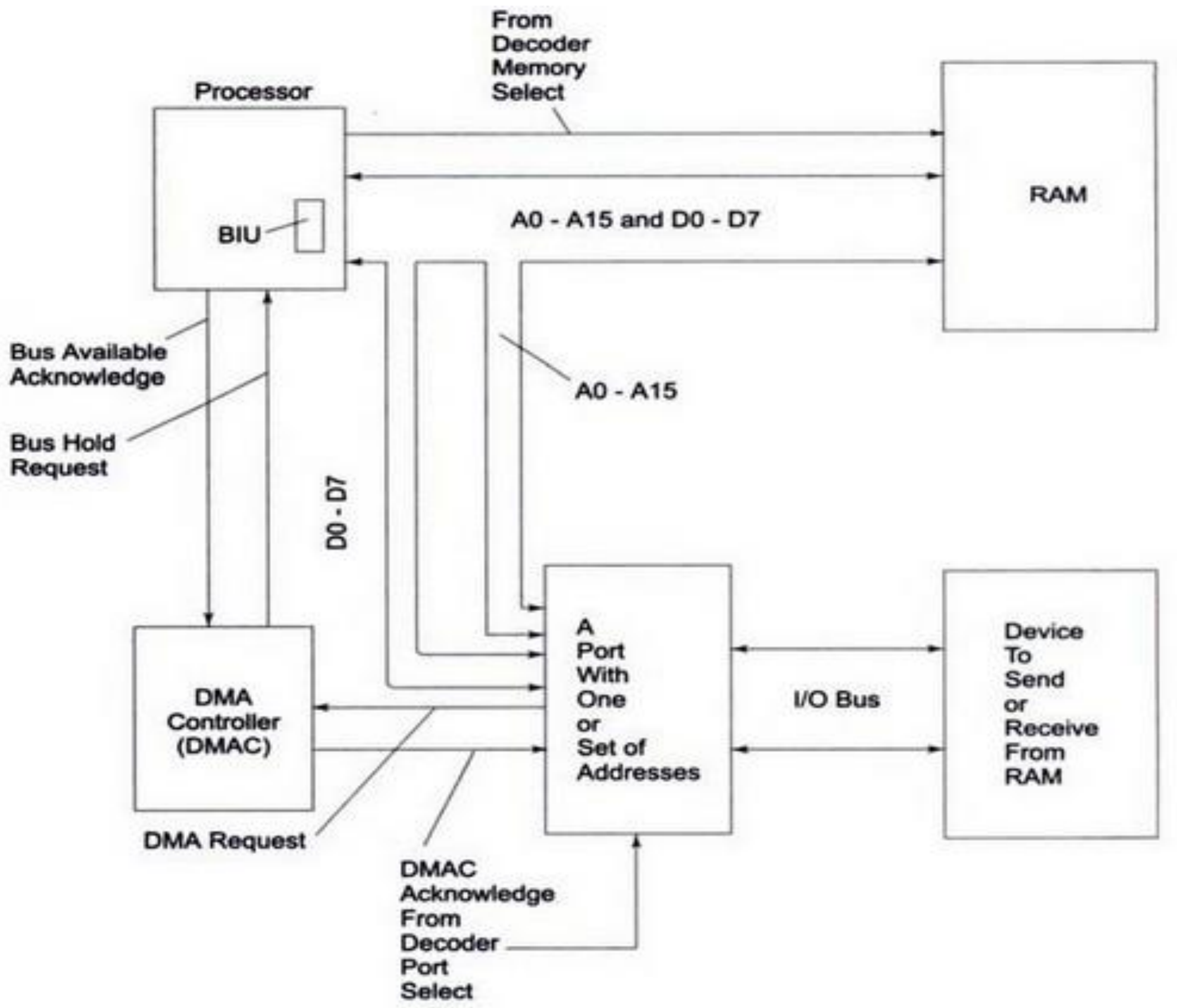
DMA

- A typical DMA controller includes three registers
    - *Starting Address Register*
    - *Length Register*
    - *Status Register*

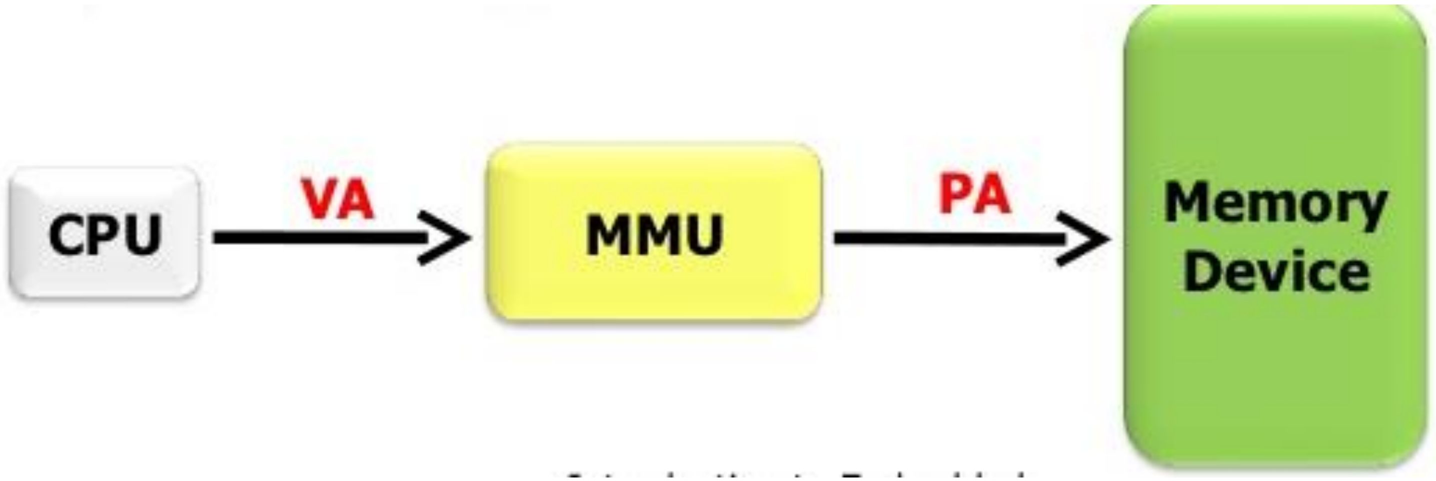
DMA



DMAC

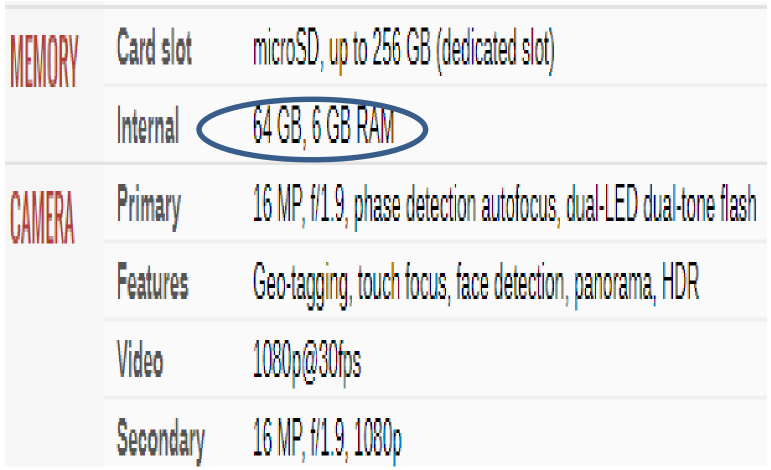

From Decoder Memory Select
Processor
BIU
RAM
A0 - A15 and D0 - D7
Bus Available Acknowledge
Bus Hold Request
A0 - A15
D0 - D7
DMA Controller (DMAC)
A Port With One or Set of Addresses
I/O Bus
Device To Send or Receive From RAM
DMA Request
DMAC Acknowledge From Decoder Port Select

Memory Management Methods



Apple iPhone 5s

- Display. 4.00-inch.
- Processor. 1.3GHz dual-core.
- Front Camera. 1.2-megapixel.
- Resolution. 640x1136 pixels.
- RAM. 1GB.
- OS. iOS 7.
- Storage. 16GB.
- Rear Camera. 8-megapixel.

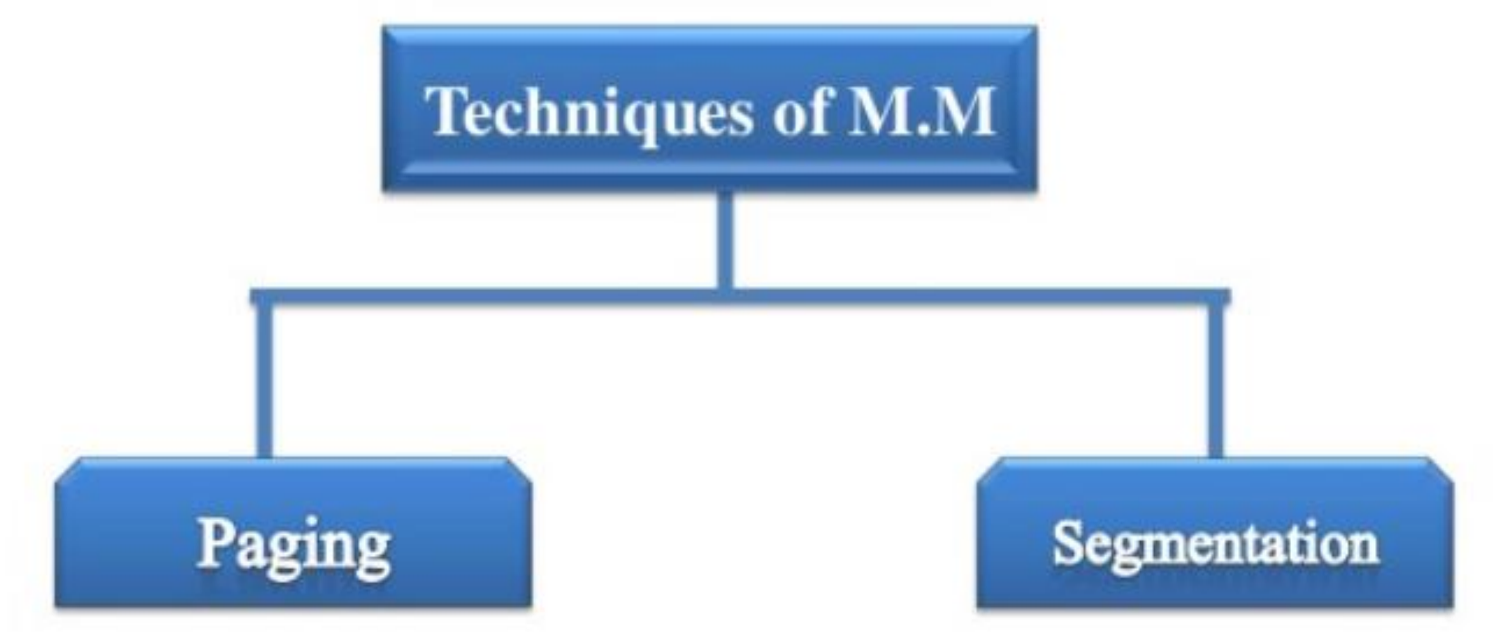


MEMORY	Card slot	microSD, up to 256 GB (dedicated slot)
	Internal	64 GB, 6 GB RAM
CAMERA	Primary	16 MP, f/1.9, phase detection autofocus, dual-LED dual-tone flash
	Features	Geo-tagging, touch focus, face detection, panorama, HDR
	Video	1080p@30fps
	Secondary	16 MP, f/1.9, 1080p

Memory Management Methods

- How MMU works
  - Keep track of **what part of memory are in use**
  - **Allocate memory to processes** when needed
  - **De-allocate** when processes are done
  - **Swapping or Paging** between main memory and disk

Memory Management Methods



Memory Management Methods

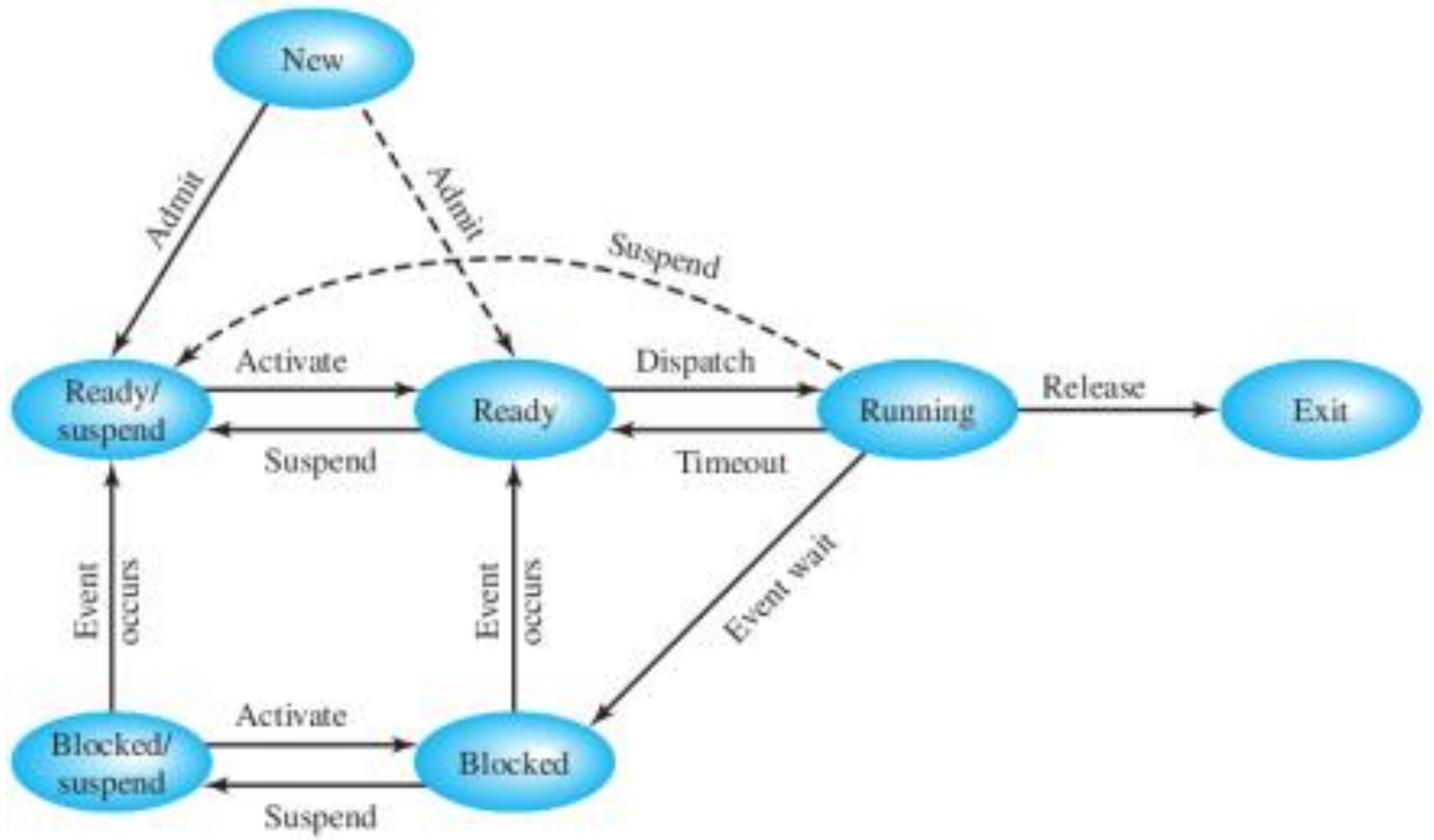
- In general, memory management responsibilities include:
  - Managing the mapping between **logical memory and task memory references**
  - Determining **which processes to load in to** the available memory space
  - **Allocating and de-allocating of memory** for processes that make up the system
  - **Tracking the memory usage** of system components
  - Ensuring **process memory protection**

User Memory Space

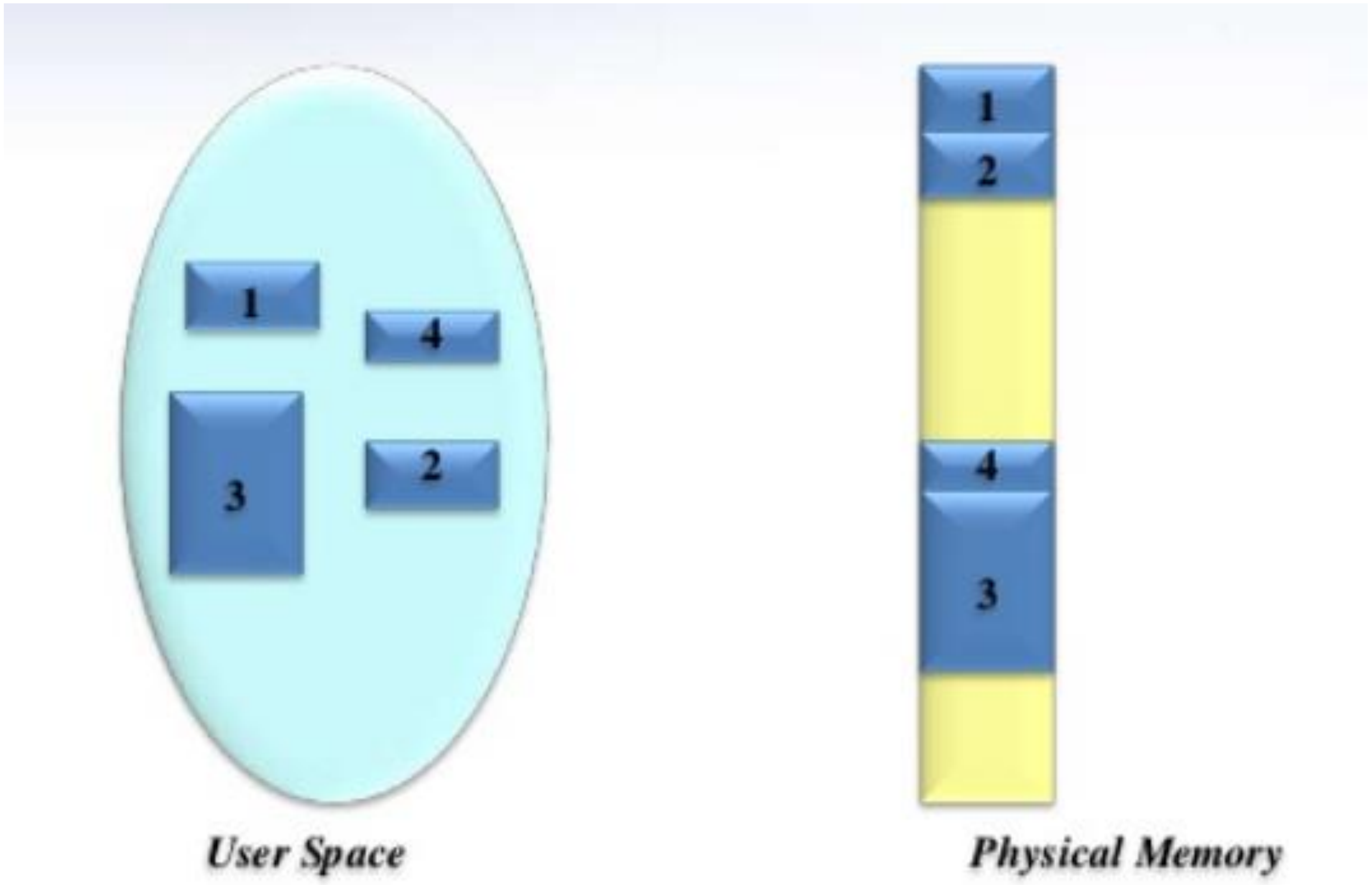
- Because multiple processes are sharing the same physical memory when being loaded into RAM for processing, there also must be some **protection mechanism** so processes cannot unintentionally affect each other when being swapped in and out of a single physical memory space
- These issues are typically resolved by the OS through "memory **swapping**" where partitions of memory are swapped at run time.

User Memory Space

- Memory **Swapping**



User Memory Space



Segmentation

- A process **encapsulates** all the information that is involved in executing a program, including **source code, stack and data.**
- All of the different types of information within a process are divided into logical memory units of variable sizes called **SEGMENTS**
- Most OS typically allow processes to have all or some combination of five types of information within segments such as
  - **Code Segment**
  - **Data Segment**
  - **Block started by Symbol**
  - **Stack Segment**
  - **Heap Segment**

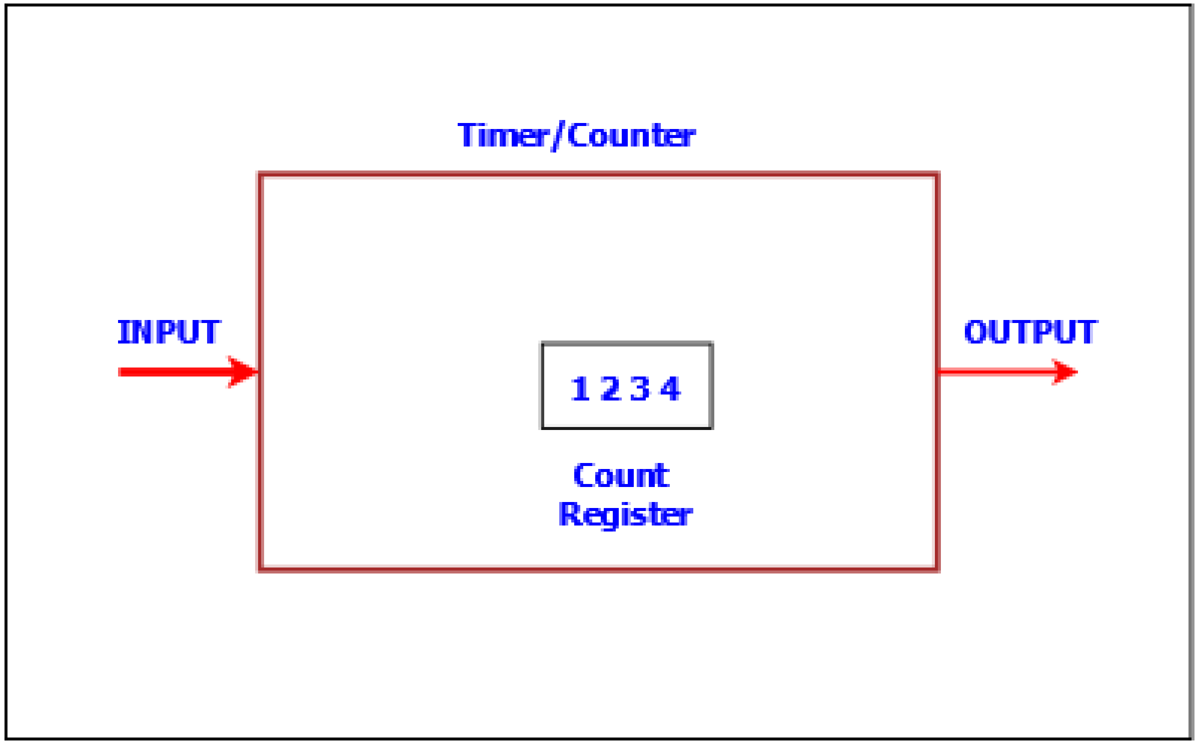
Typical Memory Allocation Schemes include:

- First Fit
  - It scans from the beginning for the first ‘Hole’ that is large enough
- Next Fit
  - It scans from where the last search ended for the next ‘Hole’ that is large enough
- Best Fit
  - It scans the entire list for the hole that best fits the new data
- Worst Fit
  - It places data in the largest available “hole”
- Quick Fit
  - Here, list is kept with memory sizes. Allocation is done from this information

Memory Management Functions

- Allocation and De-allocation
- Dynamic Blocks Allocation
- Memory Protection to the OS functions
- Memory Protection among the Tasks
- Multiprocessor Memory Allocation

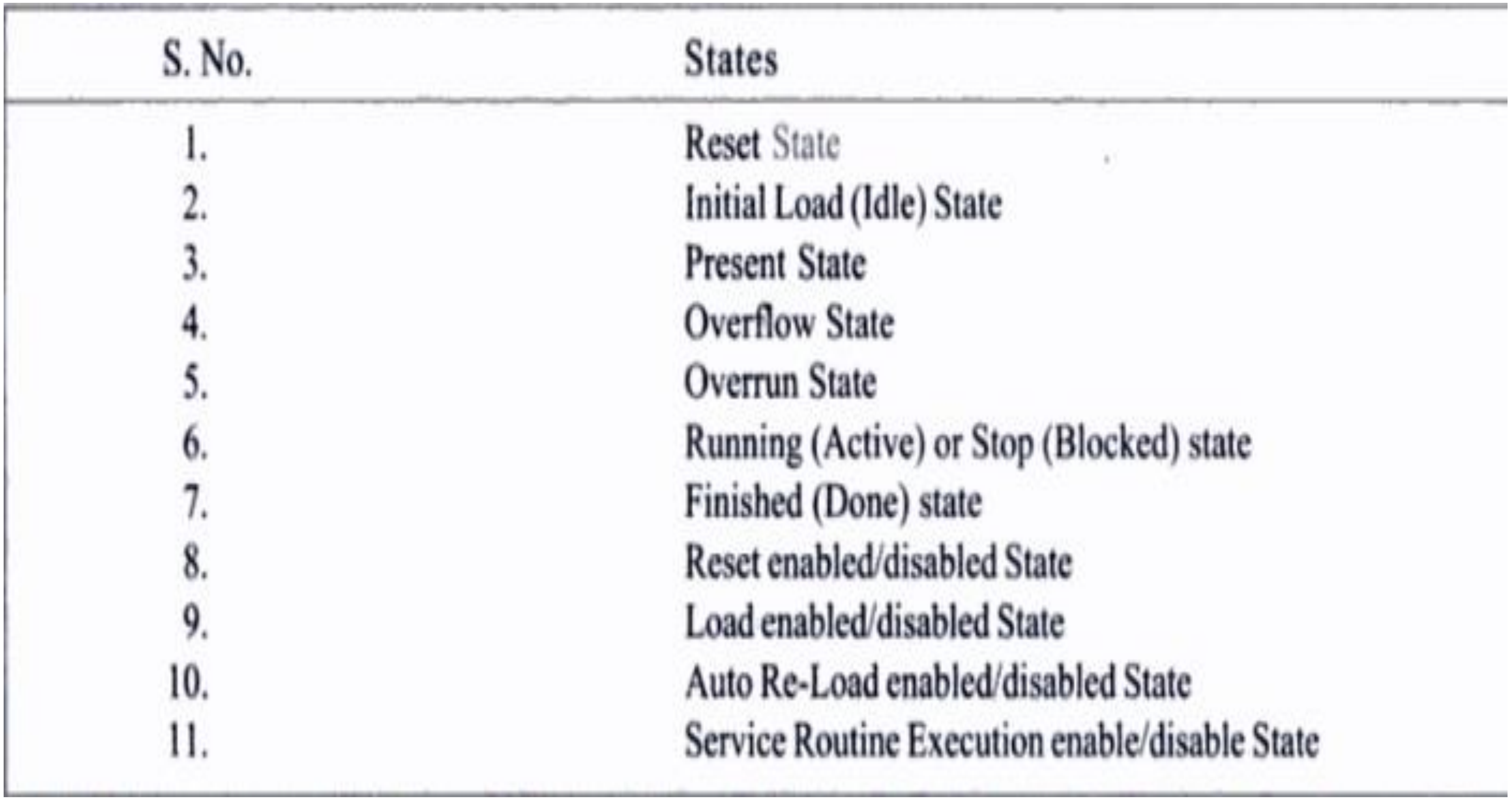
Timer and Counting Devices



Timer and Counting Devices

- **Timer Device**
  - A timer device is a device which **counts the input** at the regular interval of the clock pulses
  - The **count are stored and incremented** on each pulse
  - The timer gives **output bits** for the **present counts**
  - The **counts multiplied by the interval give the time**
  - It has a input pin for **resetting** it for all count bits = 0s
  - It has an **output pin** for output when all counts bits =0s after reaching the maximum value

Timer States

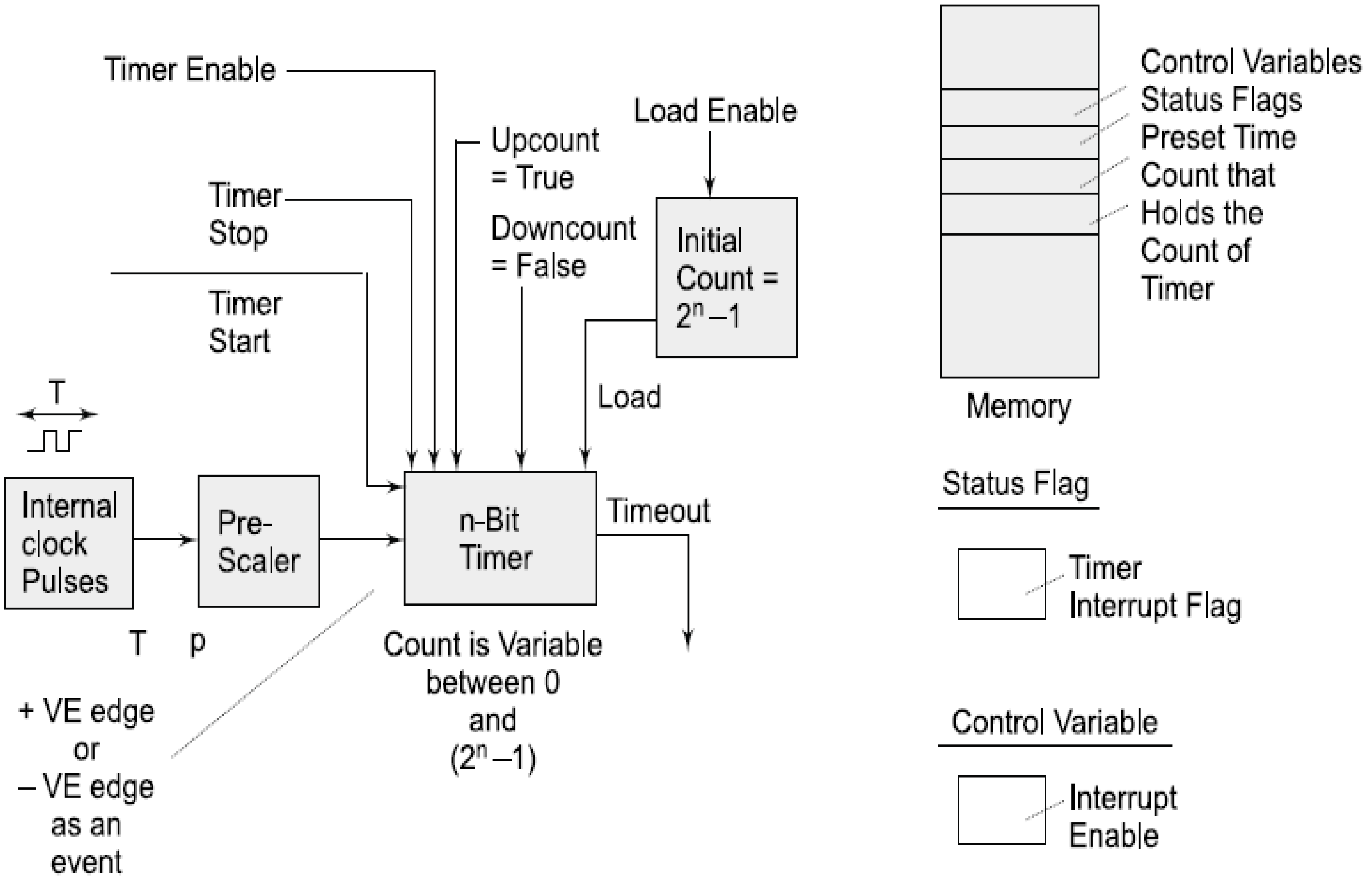


S. No.	States
1.	Reset State
2.	Initial Load (Idle) State
3.	Present State
4.	Overflow State
5.	Overrun State
6.	Running (Active) or Stop (Blocked) state
7.	Finished (Done) state
8.	Reset enabled/disabled State
9.	Load enabled/disabled State
10.	Auto Re-Load enabled/disabled State
11.	Service Routine Execution enable/disable State

Timer and Counting Devices

- Counting Devices
  - A counting device is a device which **counts the input due to the events at irregular or regular intervals**
  - The counts give the number of input events or pulses since it was last read
  - Has a register to enable read of present counts
  - Functions as timer when counting regular interval clock pulses
  - When a timer or counter becomes **0x00 or 0x0000** after **0xFF or 0xFFFF** (maximum value), it can generate an '**interrupt**', or an output '**Time-Out**' or set a status bit '**TOV**'

Hardware Timer

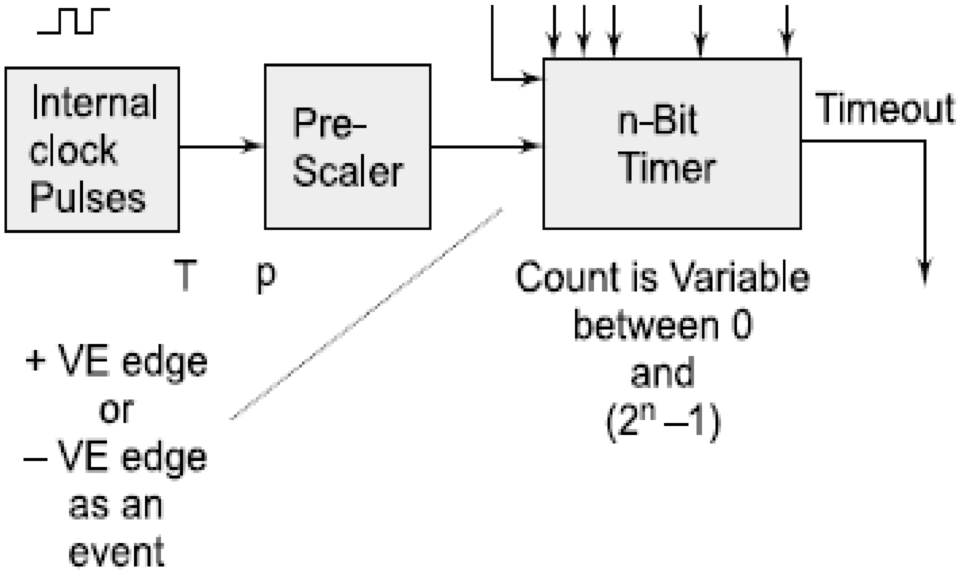


Hardware Timer

- Control Bits are of 9 types,
  I. Timer Enable (To activate a timer)
  II. Timer start (to start counting at each clock input)
  III. Timer stop (to stop counting)
  IV. Pre-scaling Bits (to divide the clock out frequency signal from the processor)
  V. Up count Enable (To enable counting up by incrementing the count value on each clock input)
  VI. Down count enable (to decrement on a clock input)
  VII. Load enable (To enable loading of a value at a register into the timer)
  VIII. Timer Interrupt Enable (To enable interrupt when the timer count value reaches 0)
  IX. Time out enable (To enable a signal when the timer overflows)

Free running Counter

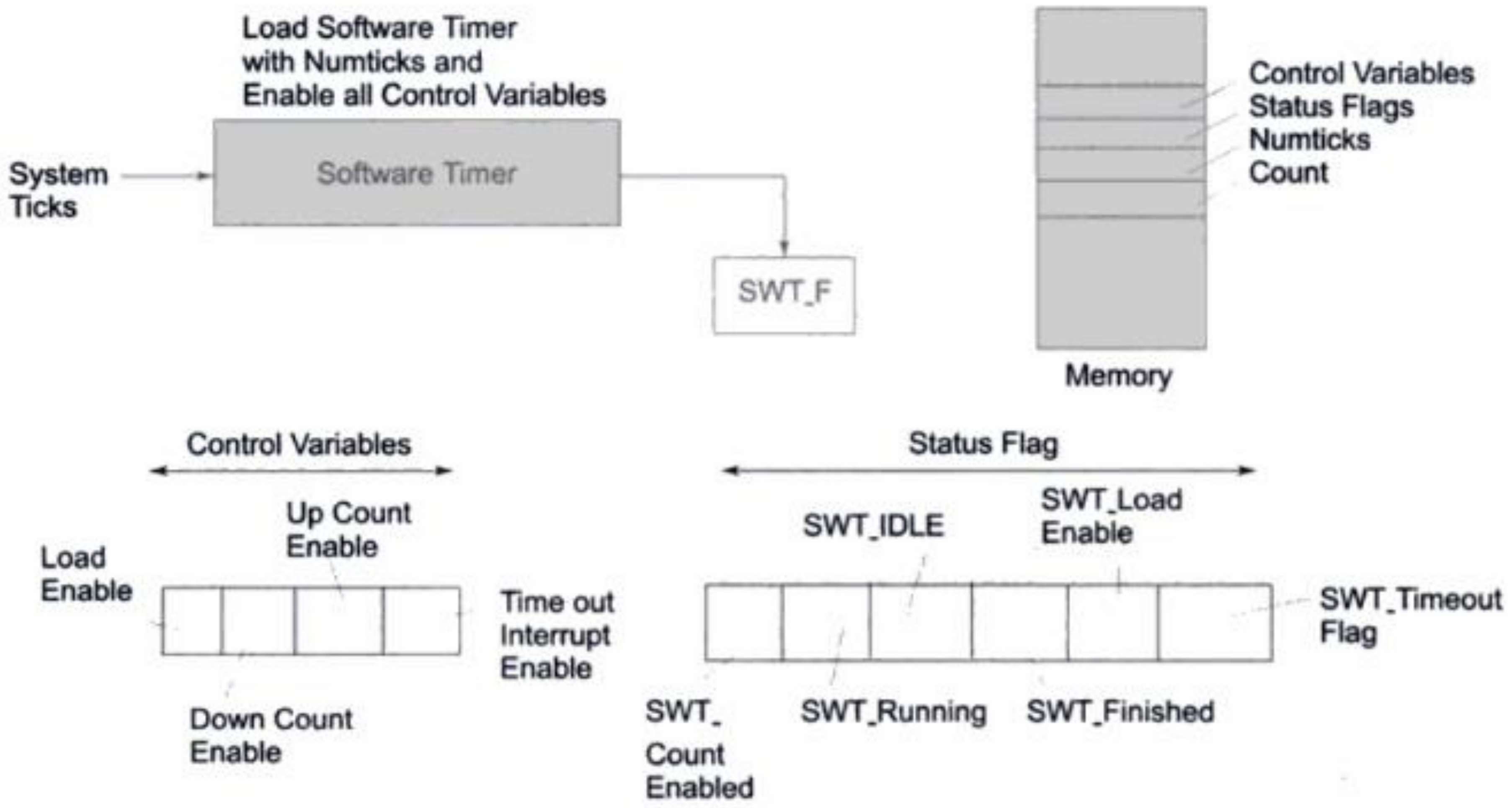
- A counting device may be a free running (blind counting) device giving overflow interrupts at fixed intervals
- A pre-scalar for the clock input pulses to fix the intervals



Internal clock Pulses
Pre-Scaler
n-Bit Timer
Timeout
T
p
Count is Variable between 0 and (2^n-1)
+ VE edge or – VE edge as an event

A Hardware Timer is a Counter that gets Inputs at Regular Intervals

Control bits, Status flags and variables of a software timer

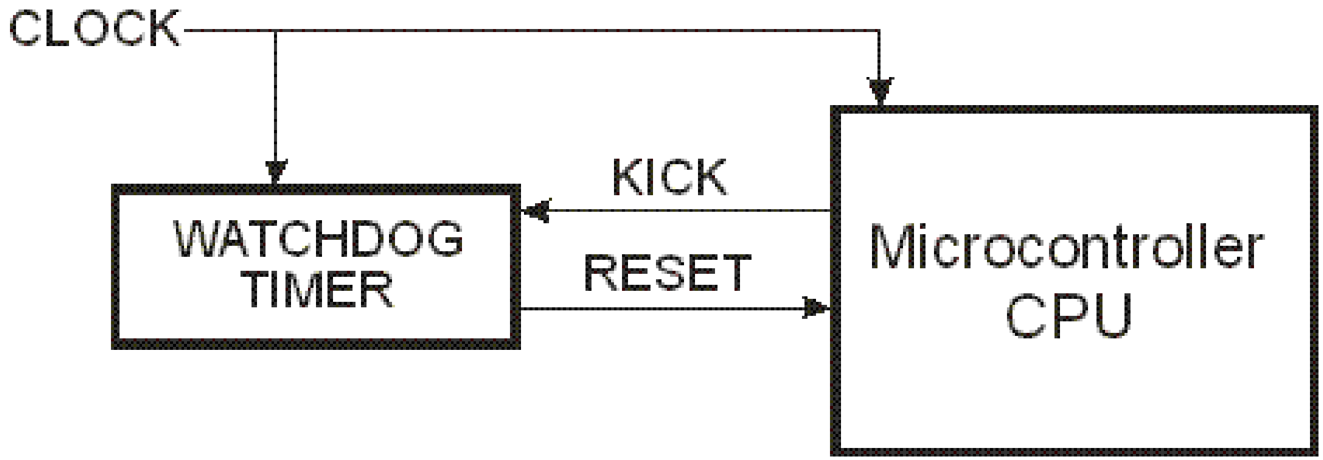


Uses of a timer device

- Initiating an event after a comparison(s) with between the pre-set time with counted value.
- Watchdog timer - It resets the system after a defined time
- Input pulse counting when using a timer, which is ticked by giving non-periodic inputs instead of the clock inputs.

Watchdog Timers

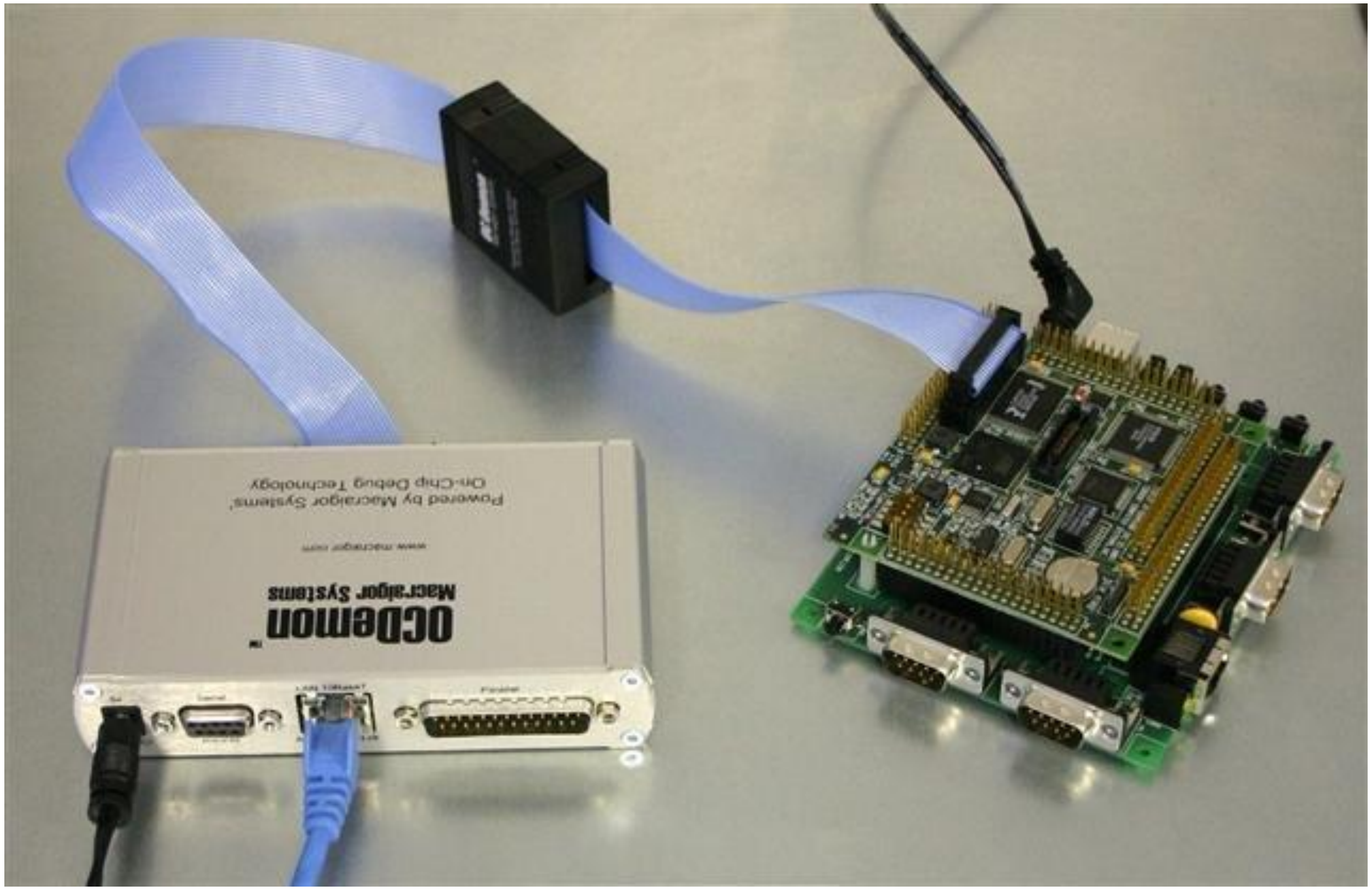
- A watchdog timer is a timing device that is set for a preset time interval and a set of task must finish during that intervals, or else the device will generate the timeout signal for the failure to finish the given task in the watched time interval.



Real Time Clocks

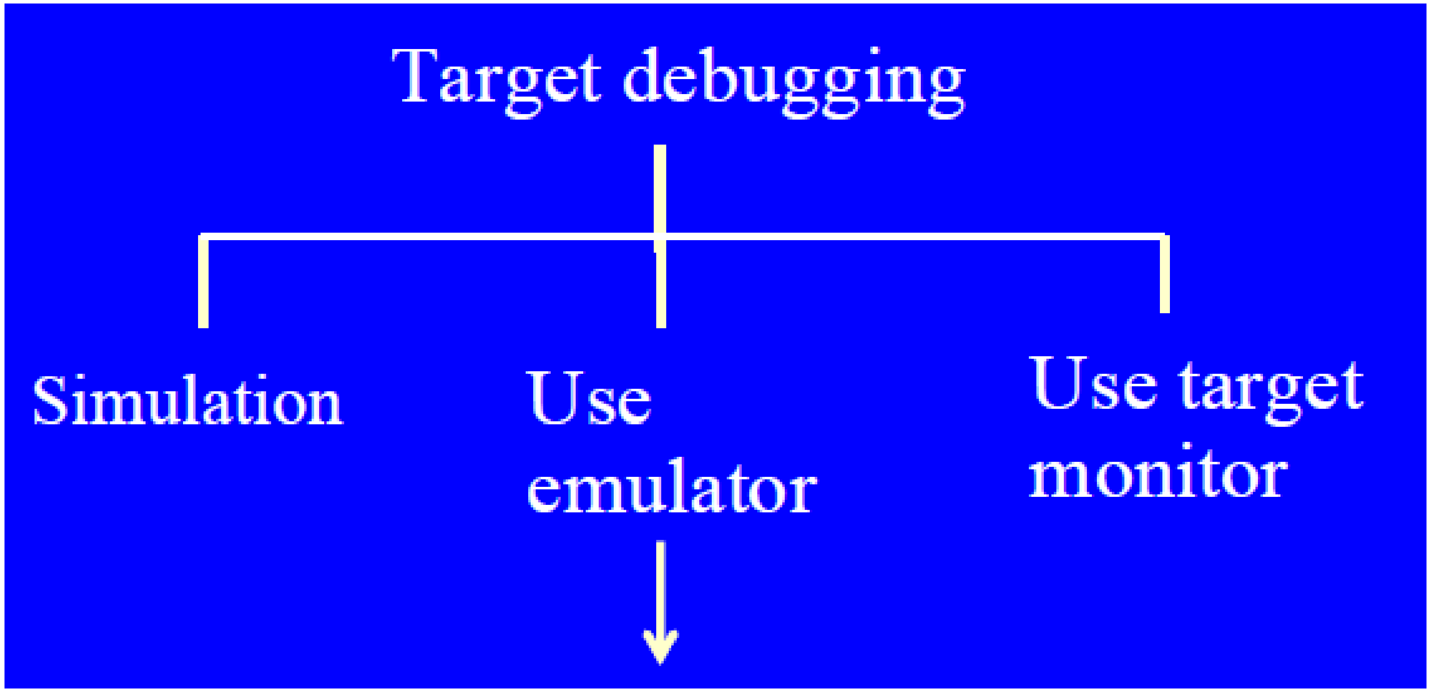
- A clock, which is based on the **interrupts at pre-set intervals.**
- An interrupt service routine executes on each **timeout (overflow)** of this clock.
- This **timing device once started never resets or never reloaded with another value**.
- **Once it is set, it is not modified later**.
- Used in a system to **save the time and date**.
- Used in a system to initiate return of control to the system (OS) after the set system clock periods

In Circuit Emulator



In Circuit Emulator

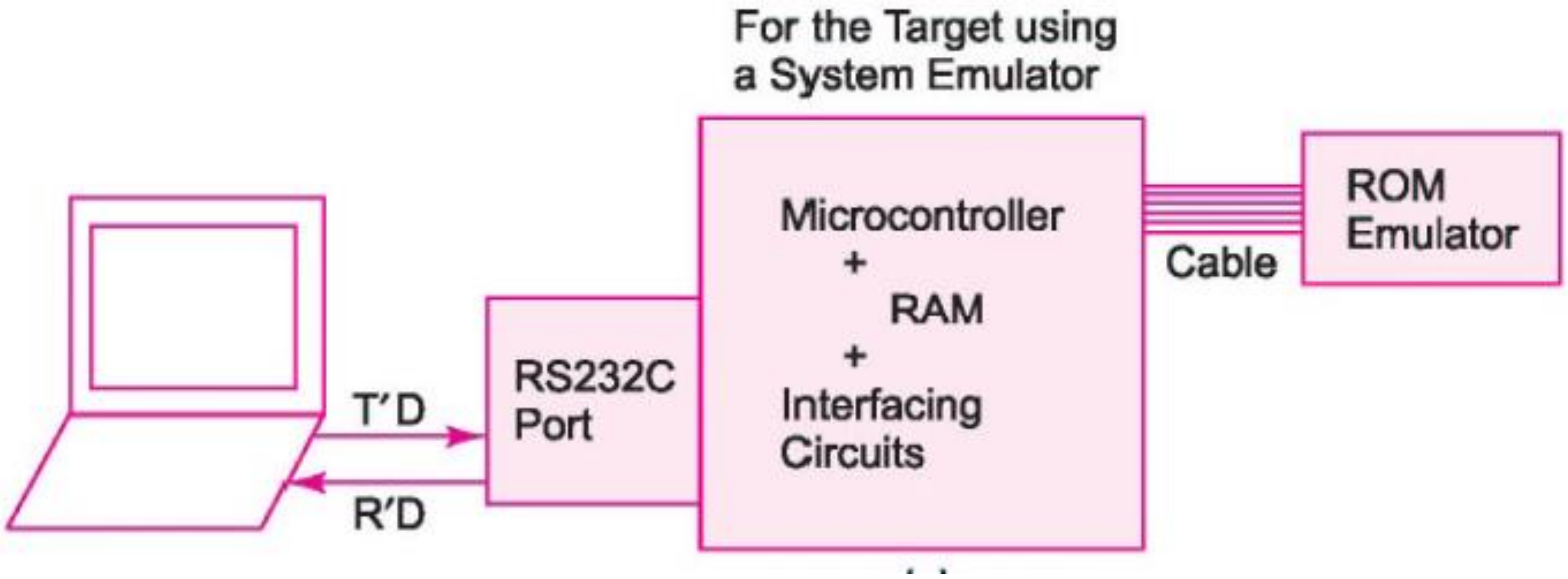
- Circuit for emulating target system remains independent of a particular targeted system and processor



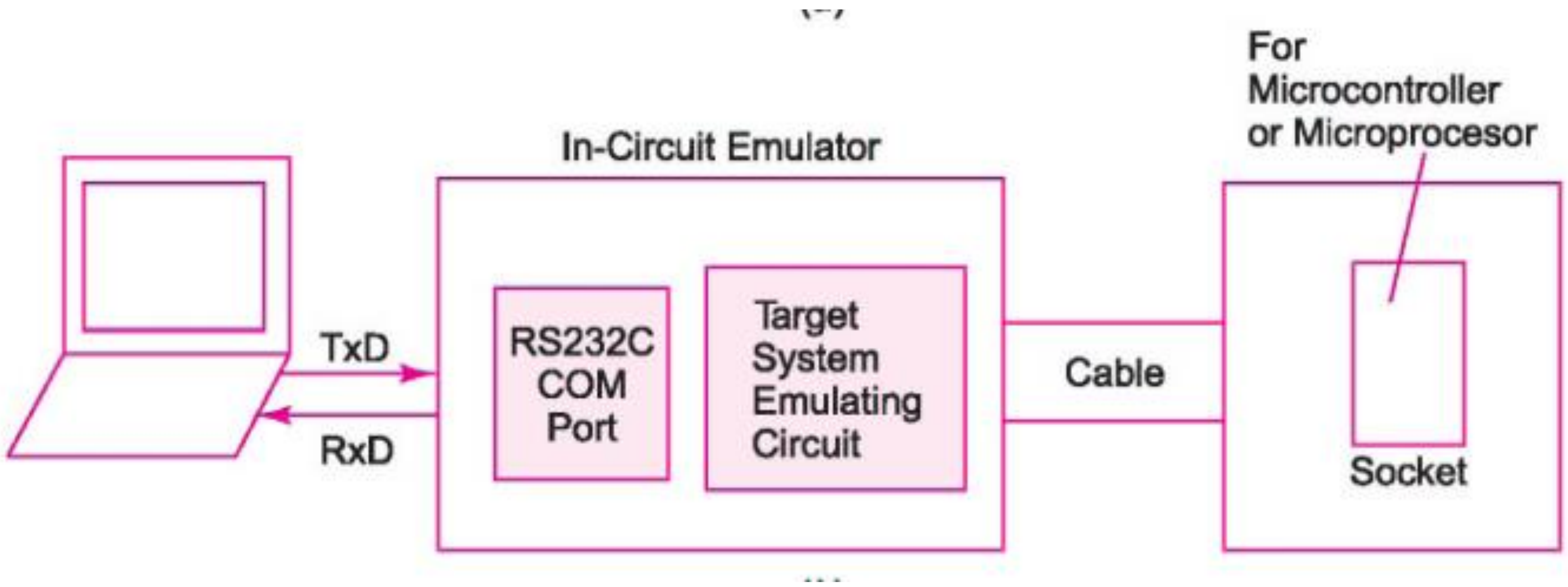
In Circuit Emulator

- A circuit for **emulating target system remains independent of a particular targeted system and processor**
- **Emulator or ICE** provides great flexibility and ease for developing various applications on a single system in place of testing that multiple targeted systems.

An Emulator


For the Target using a System Emulator
Microcontroller + RAM + Interfacing Circuits
Cable
ROM Emulator
RS232C Port
T'D
R'D

In Circuit Emulator


In-Circuit Emulator
For Microcontroller or Microprocesor
TxD
RxD
RS232C COM Port
Target System Emulating Circuit
Cable
Socket

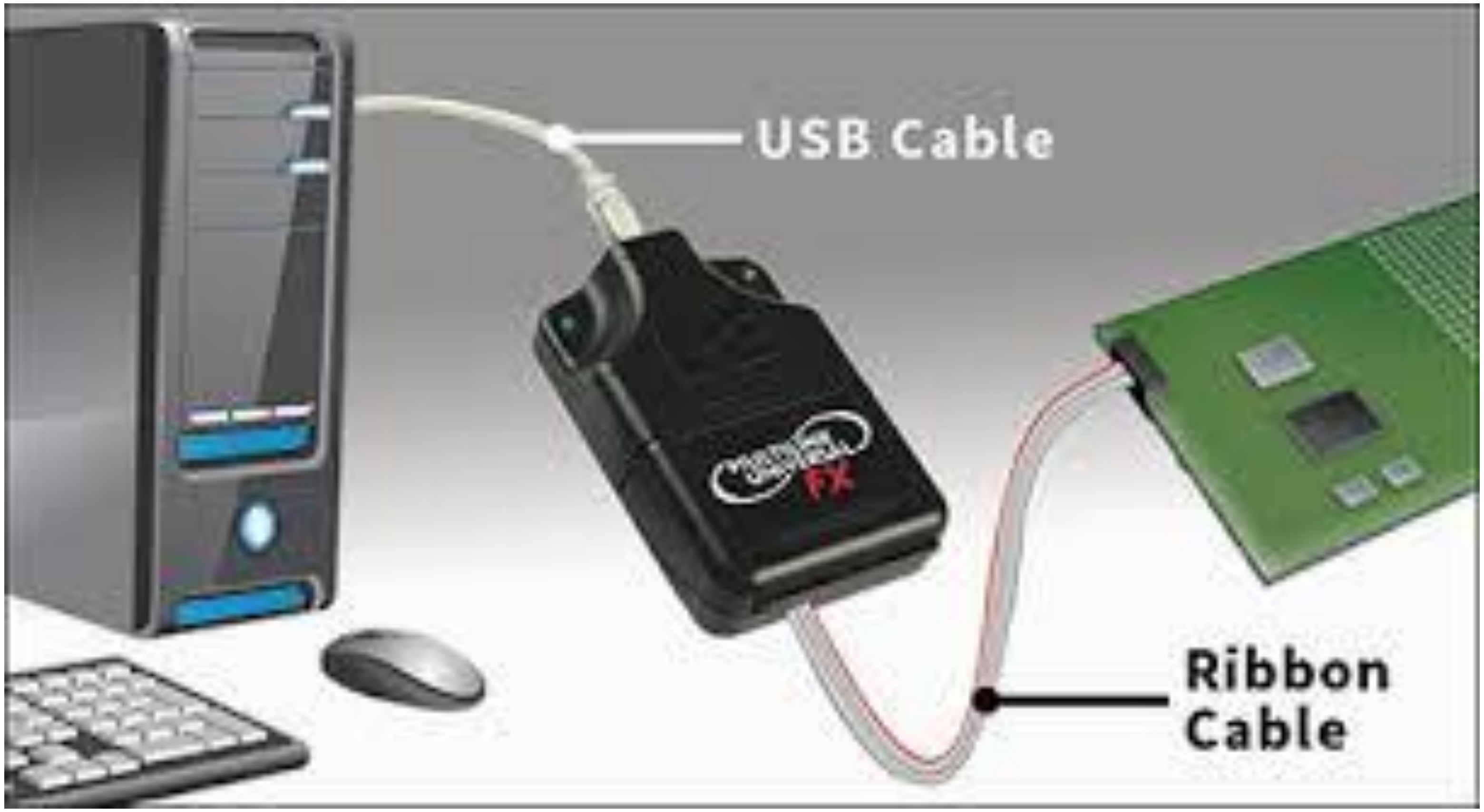
Emulator

- Emulates MCU inputs from sensors
- Emulates controlled outputs for the peripheral interfaces/systems
- Emulates target MCU IOs and socket to connect externally MCU

ICE

- Means In-Circuit Emulator
- Interface COM port of a computer
- Emulates target MCU IOs
- ICE socket connects MCU externally
- Uses computer developed object files and hex files for the MCU
- Uses debugger at the computer developed files for the MCU application

Target Hardware Debugging



Host-Target System Development Approach

- During development process, a host system is used
- Then locating and burning the codes in the target board.
- Target board hardware and software later copied to get the final embedded system
- Final system functions exactly as the one tested and debugged and finalized during the development process

Host System (PC or Laptop or Workstation)

- High performance **processor** with caches, large RAM memory
- **ROMBIOS** (read only memory basic input-output system)
- Very **large memory on disk**
- **Keyboard**
- **Display monitor**
- **Mice**
- **Network connection**
- **Program development kit** for a high level language program or **IDE**
- Host processor **compiler** and **Cross Compiler**
- **Cross assembler**

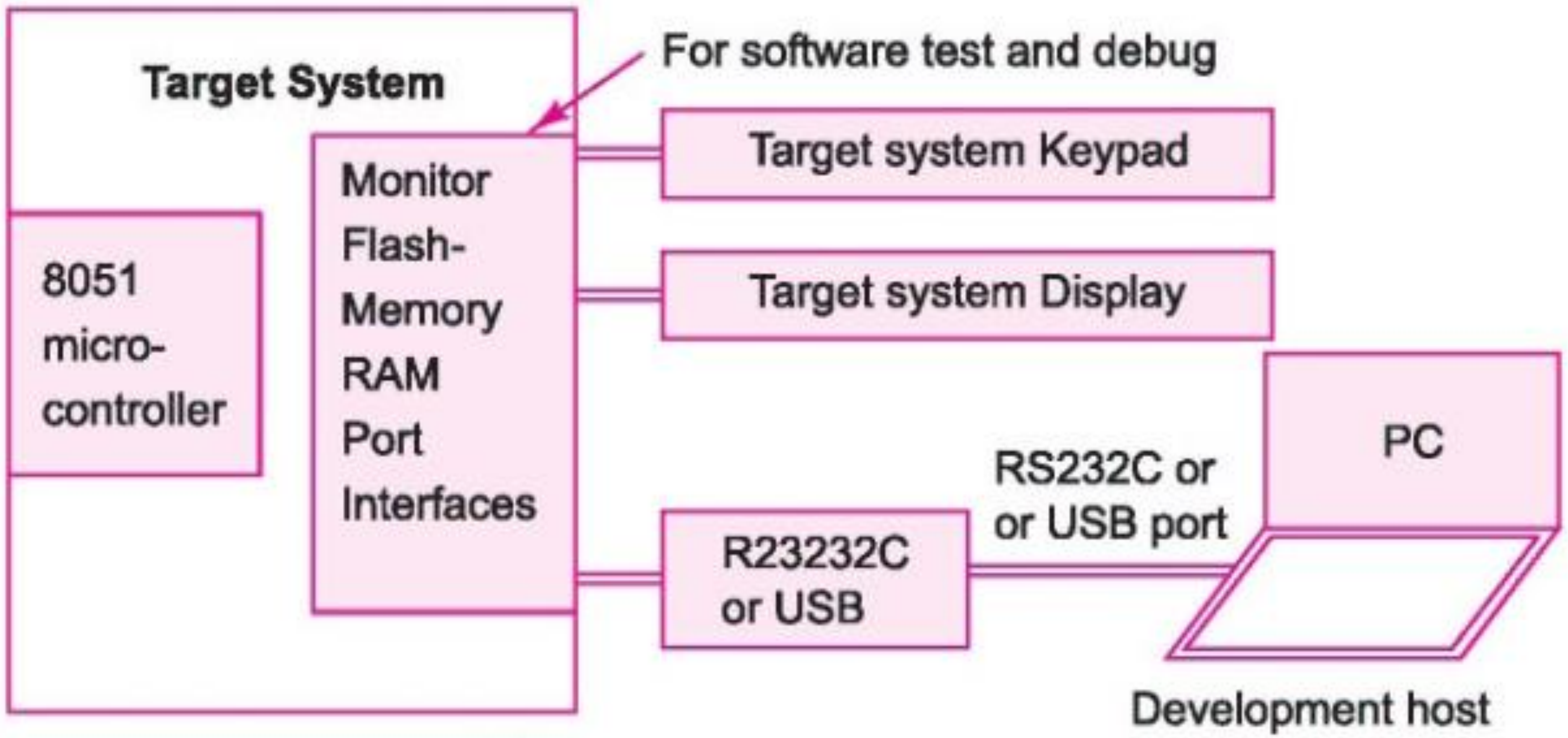
Program Development Tool Kit at host

- Program development tool kit or **IDE**
- **Editor**– used for writing C codes or assembly mnemonics or C++ or Java or Visual C++ using the keyboard of the host system (PC) for entering the program.
- Using **GUIs** for allowing the entry, addition, deletion, insert, appending previously written lines or files, merging record and files at the specific positions.

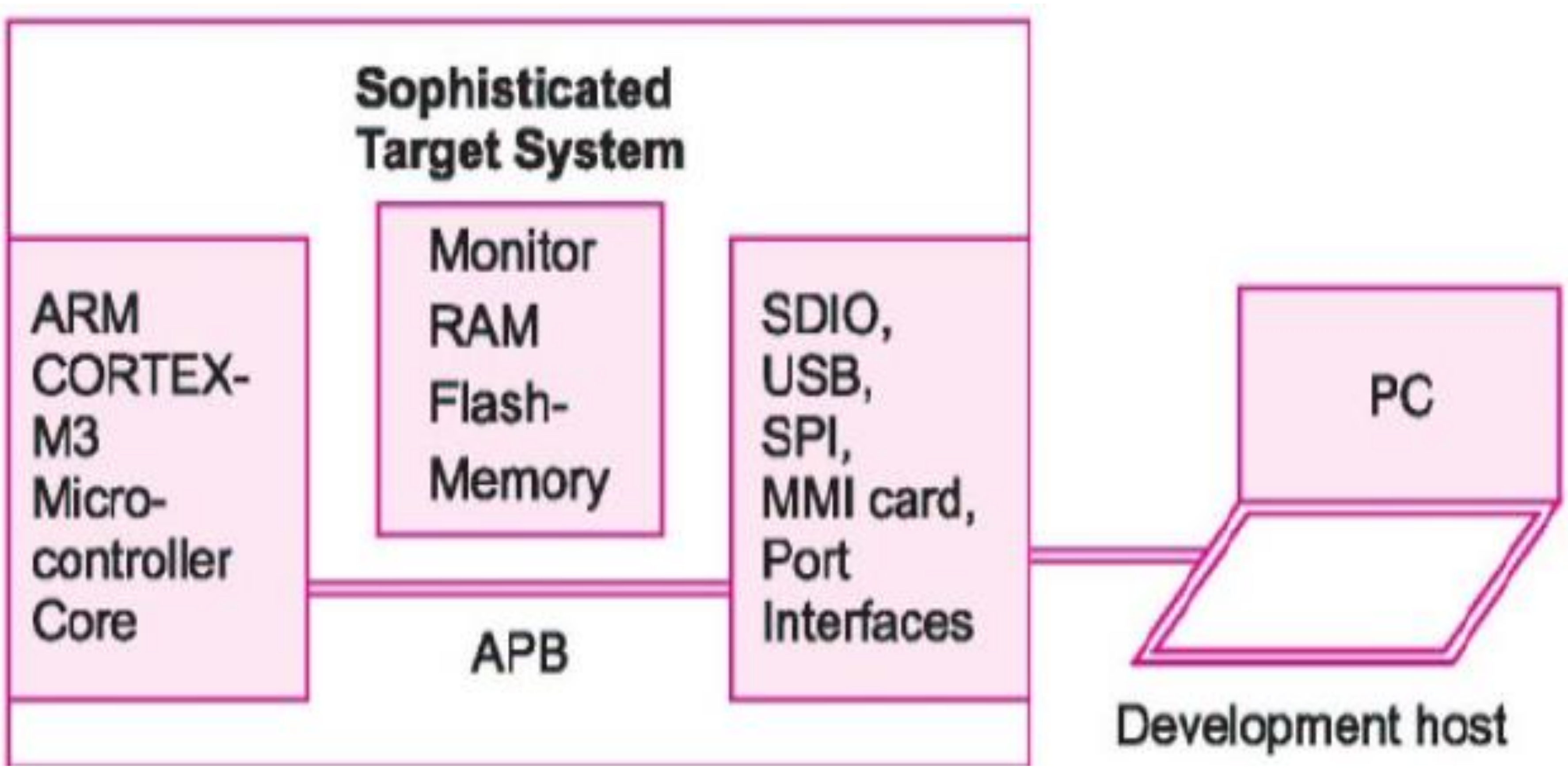
Program Development Tool Kit at host

- **Create source file** that stores the edited file.
- File given an appropriate name by the programmer
- Can **use previously created files**
- Can also **integrate the various source files**.
- Can save **different versions of the source files**.
- **Compiler, cross compiler, assembler, cross assembler**

Target System



Sophisticated Target System


Sophisticated Target System
ARM CORTEX-M3 Micro-controller Core
Monitor RAM Flash-Memory
SDIO, USB, SPI, MMI card, Port Interfaces
APB
PC
Development host

Sophisticated Target System

- Target system differs from a final system
- Target system interfaces with the computer as well as works as a standalone system
- In target system might be repeated downloading of the codes during the development phase.
- Target system copy made that later on functions as embedded system
- Designer later on simply copies it into final system or product.
- Final system may employs ROM in place of flash, EEPROM or EPROM in embedded system.

INTRODUCTION TO EMBEDDED SYSTEMS

- Introduction to Embedded Systems
- **The Build process for embedded systems** (2)
- **Structural units in Embedded processor**(2)
- Selection of processor(1)
- Selection of Memory devices (1)

Reference

1. Rajkamal, ‘Embedded System-Architecture, Programming, Design’, Mc Graw Hill, 2013.
2. Peckol, “Embedded system Design”, John Wiley & Sons, 2010.
3. Lyla B Das,” Embedded Systems-An Integrated Approach”, Pearson, 2013.