Microcontroller & Embedded Systems-Module-1

ESTD : 2001
An Institute with a Difference

Dr. Girijamma H A
Professor
Department of Computer Science and Engineering
RNS Institute of Technology
Bangalore -560098
Email: girijakasal@gmail.com
Contact: 9480031494

18CS44

ESTD : 2001
An Institute with a Difference

MICROCONTROLLER AND EMBEDDED SYSTEMS

PREREQUISITES

LEVELS OF PROGRAMMING

ESTD : 2001
An Institute with a Difference

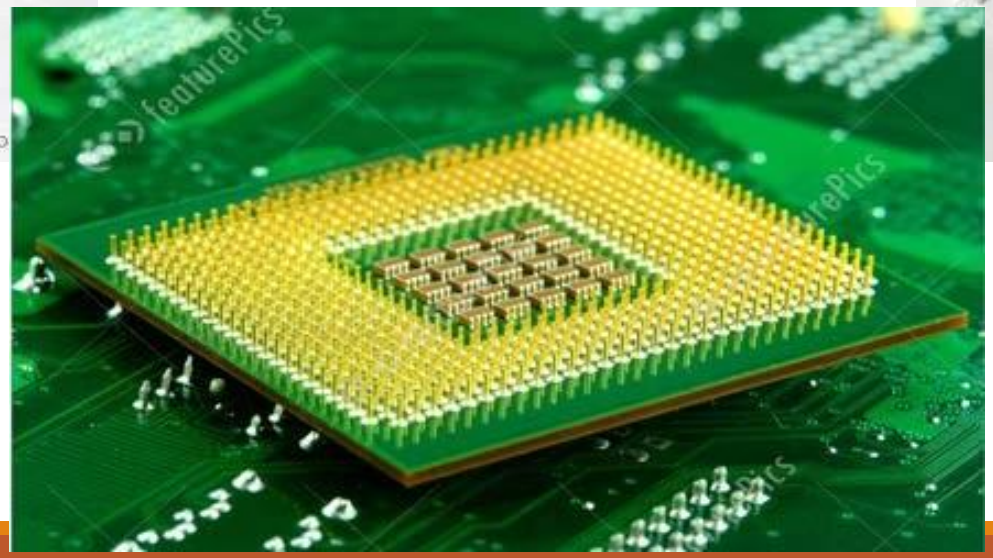
There are three levels of programming –

1. Machine language
2. Assembler language
3. High level language.

- A translator converts assembly or high-level language to machine language.
  - High-level language – Compiler.
  - Assembly language – Assembler.

A **microprocessor** is an electronic component that is used by a computer to do its work.

It is a central processing unit on a single integrated circuit chip containing millions of very small components including transistors, resistors, and diodes that work together.



The Microprocessor

- In December 1970, Gilbert Hyatt filed a patent application entitled "*Single Chip Integrated Circuit Computer Architecture*", the first basic patent on the microprocessor. The microprocessor was invented in the year 1971 in the Intel Labs. The first processor was a 4 bit processor and was called 4004.

The Modern Microprocessor

- In 1978, Intel released the **8086** microprocessor - a 16-bit microprocessors
- Addresses 1M bytes (1M byte = 1024K bytes = 1024 * 1024 bytes = 1,048,576 bytes) of memory
- Executes 2.5 MIPs (millions of instructions per second)
- A small 6-byte instruction cache or queue that pre-fetched a few instructions before they were executed.
- Its instruction set contained over 20,000 instructions.

☺ The Microprocessor (sometime called CPU)

ESTD : 2001
An Institute with a Difference

- ☺ Heart of microprocessor based computer systems
- ☺ Controls the memory and I/O through buses
- ☺ Transfers address, data, and control information between an I/O device or memory and the microprocessor via buses
- ☺ Three buses exist for the transfer of following information–
    1. Address,
    2. Data, and
    3. Control.
- ☺ Memory and I/O are controlled through instructions
- ☺ Instructions are stored in the memory and executed by the microprocessor.

- The microprocessor performs *three main tasks* for the computer system –
  1. Data transfer between itself and the memory or I/O systems,
  2. Simple arithmetic and logic operations, and
  3. Program flow via simple decisions.
- The *power of microprocessor* is –
  - Its capability to execute hundreds of millions of instructions per second
  - Also, a microprocessor can make simple decisions based upon numerical facts.

- The ***applications of microprocessors*** can be sub-divided into ***three*** categories.
  - The first and most important one is the computer applications.
  - The second one is the control application (micro-controllers, embedded controllers etc.).
  - The third is in communication (DSP processors, Cell phones etc.).

A **microcontroller** is a compact integrated circuit designed to govern a specific operation in an embedded system.

A typical microcontroller includes a processor, memory and input/output (I/O) peripherals on a single chip.

ESTD : 2001
An Institute with a Difference

18CS44

MICROCONTROLLER AND EMBEDDED SYSTEMS

MODULE 1

ARM EMBEDDED SYSTEMS

Module-1 Contents

ESTD : 2001
An Institute with a Difference

Microprocessors versus Microcontrollers, ARM Embedded Systems: The RISC design philosophy, The ARM Design Philosophy, Embedded System Hardware, Embedded System Software.
ARM Processor Fundamentals: Registers, Current Program Status Register, Pipeline, Exceptions, Interrupts, and the Vector Table , Core Extensions

Text book 1: Chapter 1 - 1.1 to 1.4, Chapter 2 - 2.1 to 2.5

Module-1 Contents

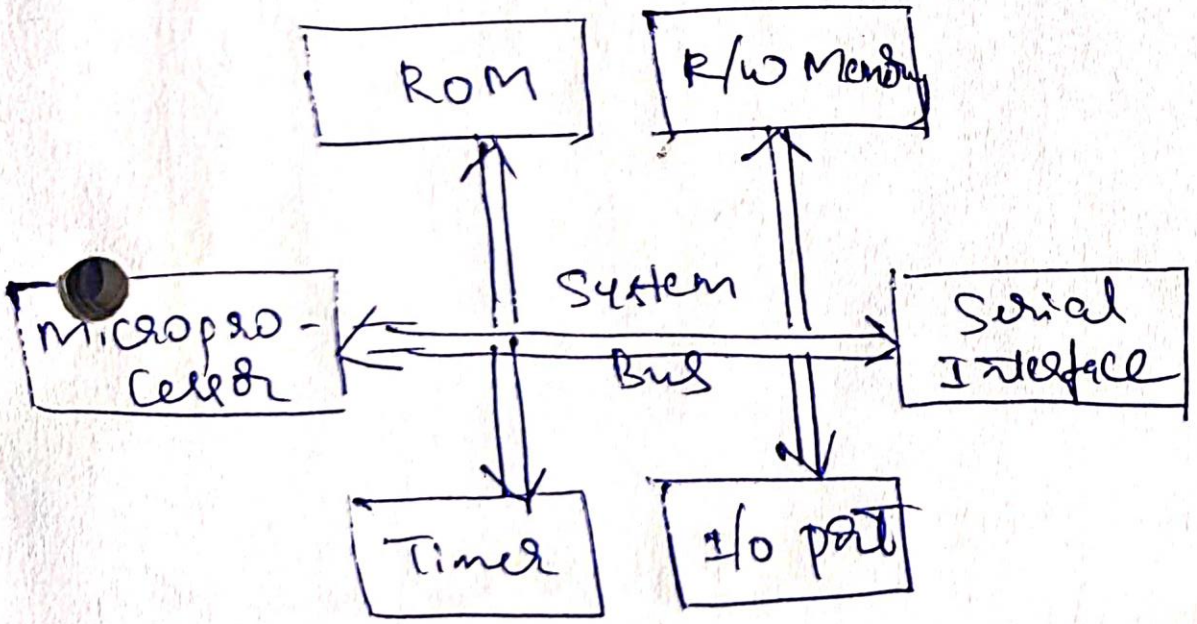
ESTD : 2001
An Institute with a Difference

1.1 Microprocessors versus Microcontrollers
1.2 ARM Embedded Systems
    1.2.1 The RISC design philosophy
    1.2.2 The ARM Design Philosophy
    1.2.3 Embedded System Hardware
    1.2.4 Embedded System Software
1.3 ARM Processor Fundamentals
    1.3.1 Registers
    1.3.2 Current Program Status Register
    1.3.3 Pipeline
    1.3.4 Exceptions, Interrupts,

1.1 Microprocessors versus Microcontrollers

Microprocessors Versus Microcontrollers

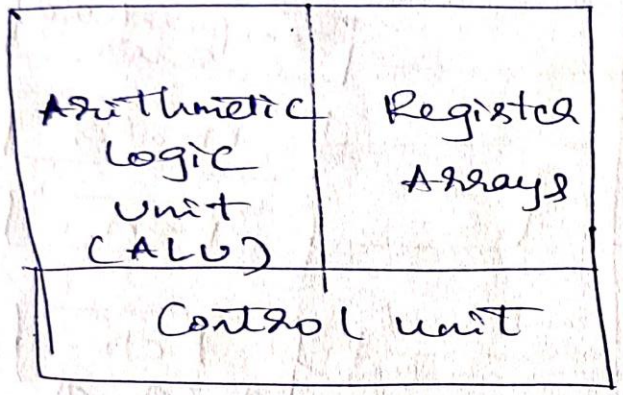
Microprocessor



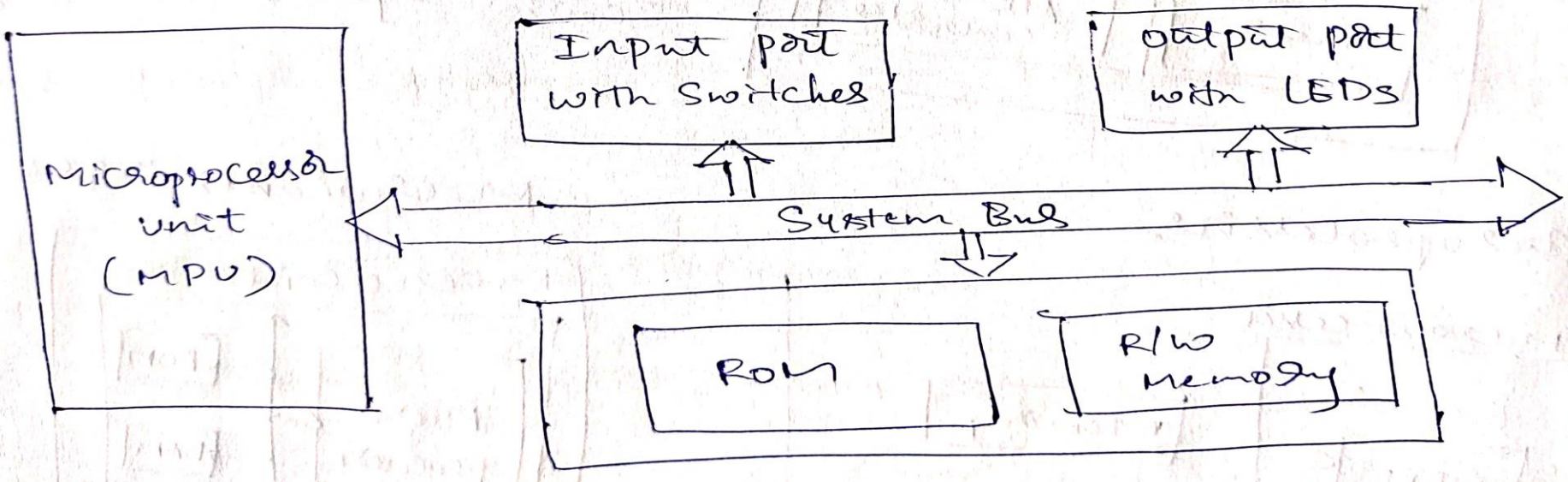
Microcontroller

Micro-controller	Read only Memory	Read-write memory
Timer	I/o port	Serial Interface

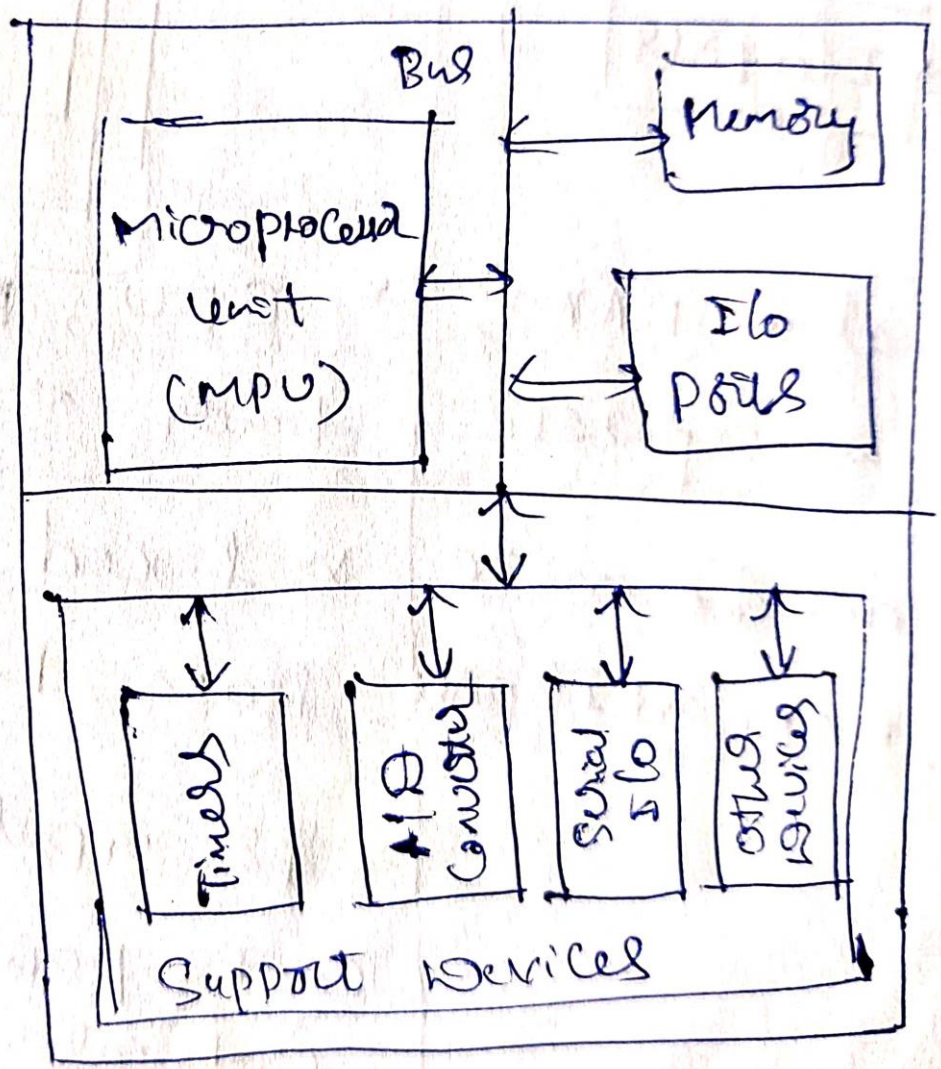
* Microprocessor



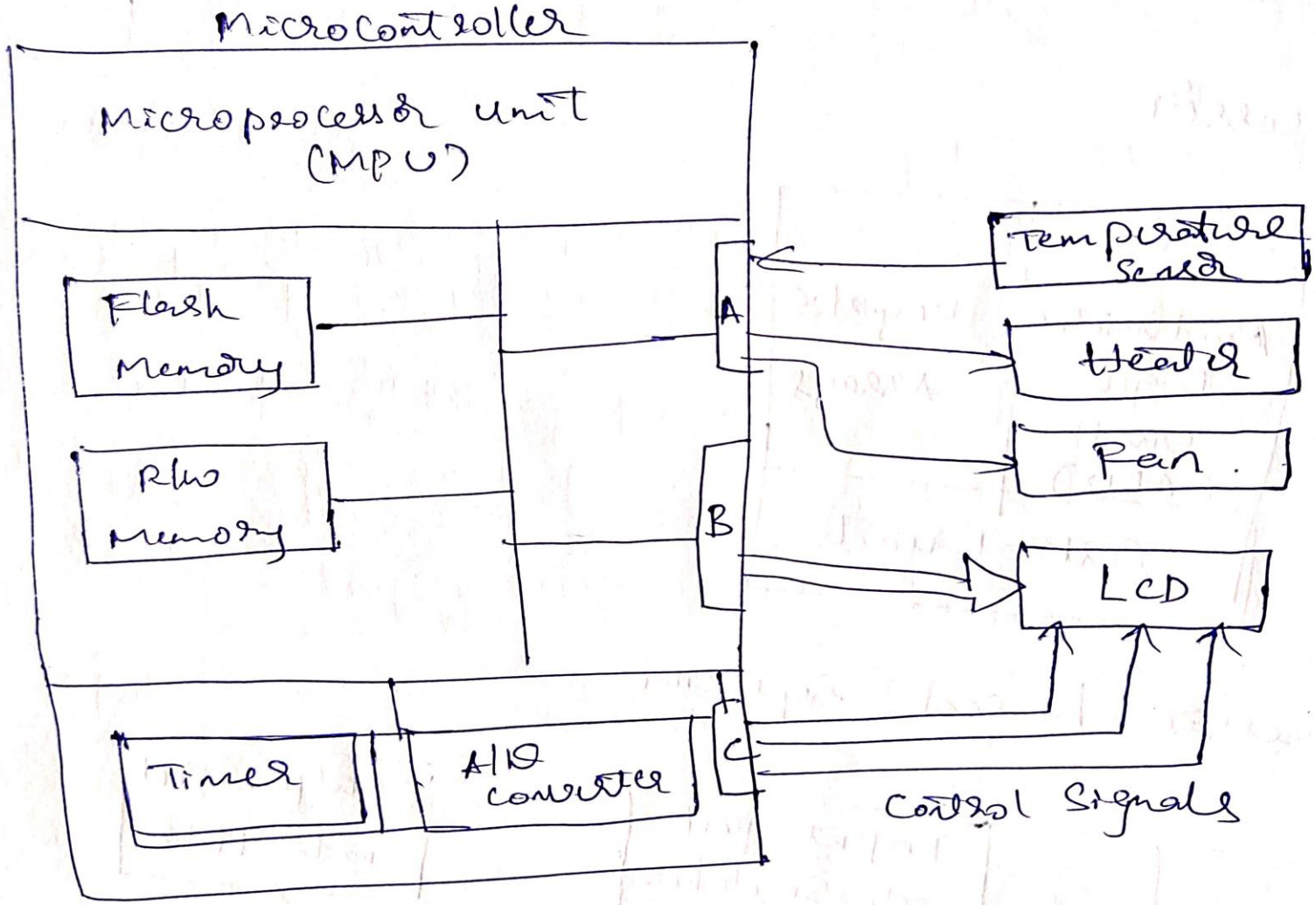
* Microprocessor based system



Microcontroller



* Microcontroller based system



MICROPROCESSORS versus MICROCONTROLLERS

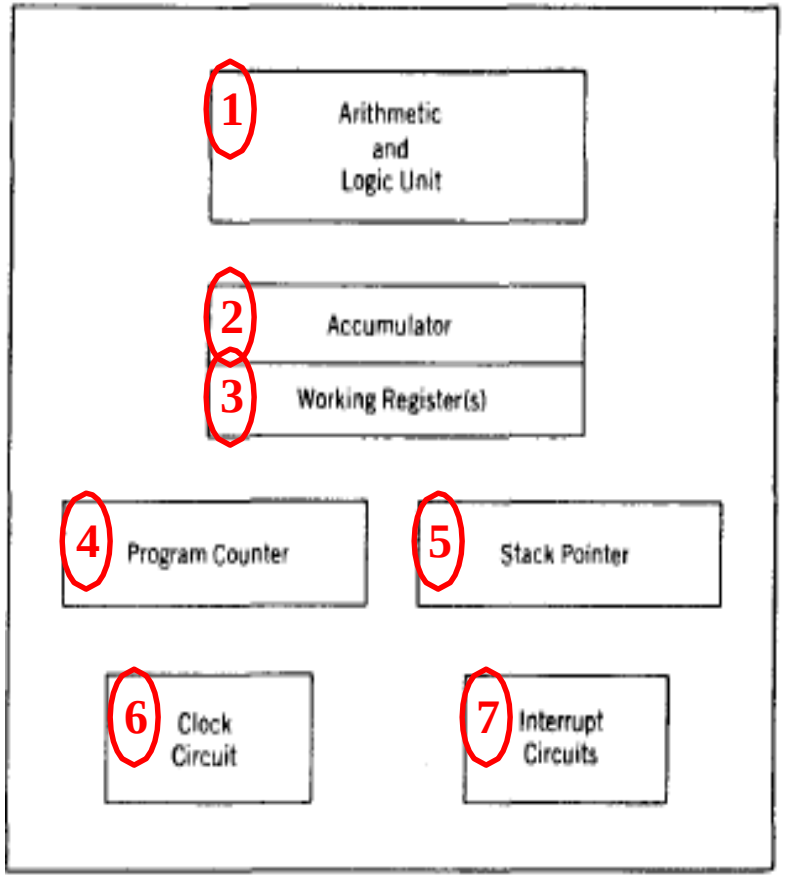
1) Microprocessors generally does not have RAM, ROM and I/O pins.
2) Microprocessors usually uses its pins as a bus to interface to RAM, ROM, and peripheral devices. Hence, the controlling bus is expandable at the board level.
3) Microprocessors are generally capable of being built into bigger general purpose applications.
4) Microprocessors, generally do not have power saving system.

1) Microcontroller is 'all in one' processor, with RAM, I/O ports, all on the chip.
2) Controlling bus is internal and not available to the board designer.
3) Microcontrollers are usually used for more dedicated applications.
4) Microcontrollers have power saving system, like idle mode or power saving; mode so overall it uses less power.

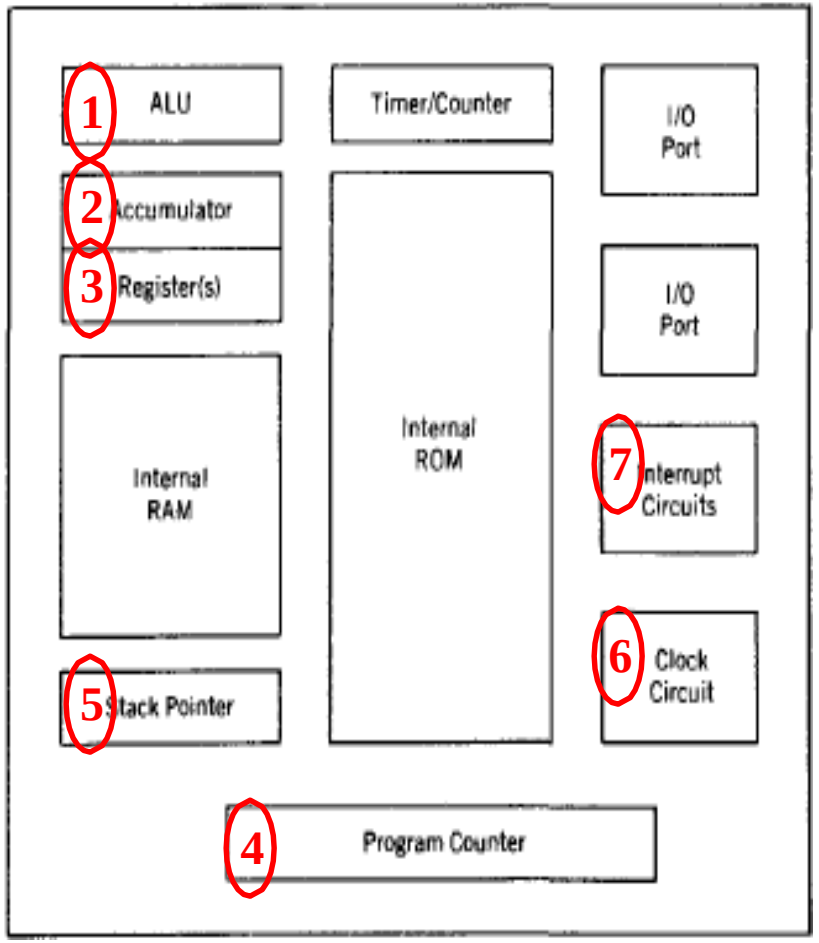
Cont…

5) The overall cost of systems made with Microprocessors are high, because of the high number of external components required.	5) Microcontrollers are made by using complementary metal oxide semiconductor technology; so they are far cheaper than Microprocessors.
6) Processing speed of general microprocessors is above 1 GHz ; so it works much faster than Microcontrollers.	6) Processing speed of Microcontrollers is about 8 MHz to 50 MHz.
7) Microprocessors are based on von-Neumann model; where, program and data are stored in same memory module.	7) Microcontrollers are based on Harvard architecture; where , program memory and data memory are separate.

A Block Diagram of a Microprocessor



A Block Diagram of a Microcontroller



1.2 ARM Embedded Systems

- The ARM processor core is a key component of many successful 32-bit embedded systems.
- ARM cores are widely used in mobile phones, handheld organizers, and a multitude of other everyday portable consumer devices.
- The first ARM prototype name ARM1 was designed in 1985 and continues to improve through constant technical innovation leading to ARM2, ARM3, ARM4, ARM5, ARM6, ARM7, ARM8, ARM9… ARM Cortex.

ARM History

- 1983 developed by Acorn computers

  1990 ARM (Advanced RISC

  Machine), owned by Acorn, Apple

  and VLSI

  •

Nomenclature of ARM

ARMxyzTDMIEJFS

- x: series
- y: MMU
- z: cache
- T: Thumb
- D: debugger
- M: Multiplier
- I: EmbeddedICE (built-in debugger hardware)
- E: Enhanced instruction
- J: Jazelle (JVM)
- F: Floating-point
- S: Synthesizible version (source code version for EDA tools)

Popular ARM Architectures

- ARM7TDMI
  - 3 pipeline stages (fetch/decode/execute)
  - High code density/low power consumption
  - One of the most used ARM-version (for low-end systems)
  - All ARM cores after ARM7TDMI include TDMI even if they do not include TDMI in their labels
- ARM9TDMI
  - Compatible with ARM7
  - 5 stages (fetch/decode/execute/memory/write)
  - Separate instruction and data cache
- ARM11

ARM Embedded Systems

- ARM cores are widely used in mobile phones, handheld organizers, and a multitude of other everyday portable consumer devices.
- The first ARM1 prototype was designed in 1985.
  - Success – a simple and powerful original design, which continues to improve today through constant technical innovation.
    - Example – ARM7TDMI:
    - provides up to 120 Dhrystone MIPS
    - high code density and low power consumption
    - ideal for mobile embedded devices

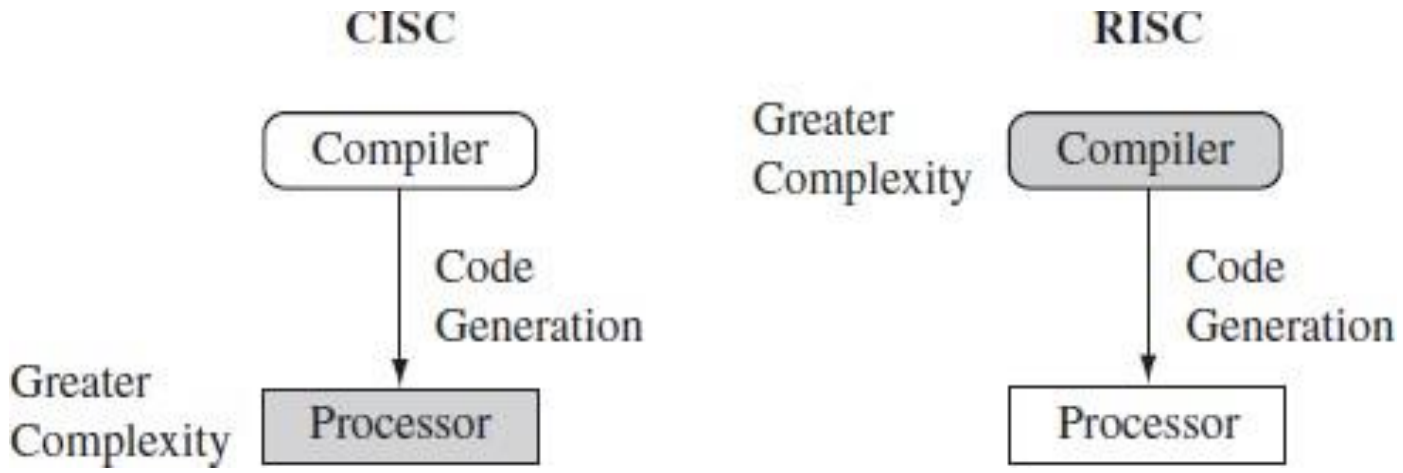
Dhrystone is a benchmark program written in *C* or *Pascal* (and now in *Java*) that tests a system's performance.

1.2.1THE RISC DESIGN PHYLOSOPHY

ESTD : 2001
An Institute with a Difference

- The ARM core uses *reduced instruction set computer (RISC)* architecture.
- RISC
  - a design philosophy
  - delivering simple but powerful instructions, that execute within a single cycle at a high clock speed
  - reducing the complexity of instructions performed by the hardware
    - it is easier to provide greater flexibility and intelligence in software rather than hardware
  - places greater demands on the compiler

- The ***complex instruction set computer (CISC)***
  - relies more on the hardware for instruction functionality
  - instructions are more complicated



CISC	RISC
1. Complex instructions, taking multiple clock	1. Simple instructions, taking single clock
2. Emphasis on hardware, complexity is in the micro-program/processor	2. Emphasis on software, complexity is in the complier
3. Complex instructions, instructions executed by micro-program/processor	3. Reduced instructions, instructions executed by hardware
4. Variable format instructions, single register set and many instructions	4. Fixed format instructions, multiple register sets and few instructions
5. Many instructions and many addressing modes	5. Fixed instructions and few addressing modes
6. Conditional jump is usually based on status register bit	6. Conditional jump can be based on a bit anywhere in memory
7. Memory reference is embedded in many instructions	7. Memory reference is embedded in LOAD/STORE instructions

RISC Design Rules:

Instructions, Pipeline, Registers, Load-Store Architecture

1. Instructions –
   - » Reduced number of instruction classes
   - » Simple operations
   - » Each instruction can execute in a single clock cycle
   - » Compiler/programmer synthesizes complicated operations
   - » Each instruction is having fixed length – allows pipeline to fetch future instructions
     - ○ In CISC processors, the instructions are often of variable size and take many cycles to execute

2. Pipelines –

- Processing of instructions is broken down into smaller units
- Instructions are executed in parallel by pipelines
- Pipeline advances by one step on each cycle for maximum throughput
  - There is no need for an instruction to be executed by a mini-program called microcode as on CISC processors

3. Registers –

- Large general-purpose register set
- Any register can contain either data or an address
- Registers act as the fast local memory store for all data processing operations
  - CISC processors have dedicated registers for specific purposes

4. Load-Store Architecture –

» Processor operates on data held in registers

» Separate load and store instructions transfer data between the register bank and external memory

» Separating memory accesses from data processing provides an advantage because you can use data items held in the register bank multiple times without needing multiple memory accesses.

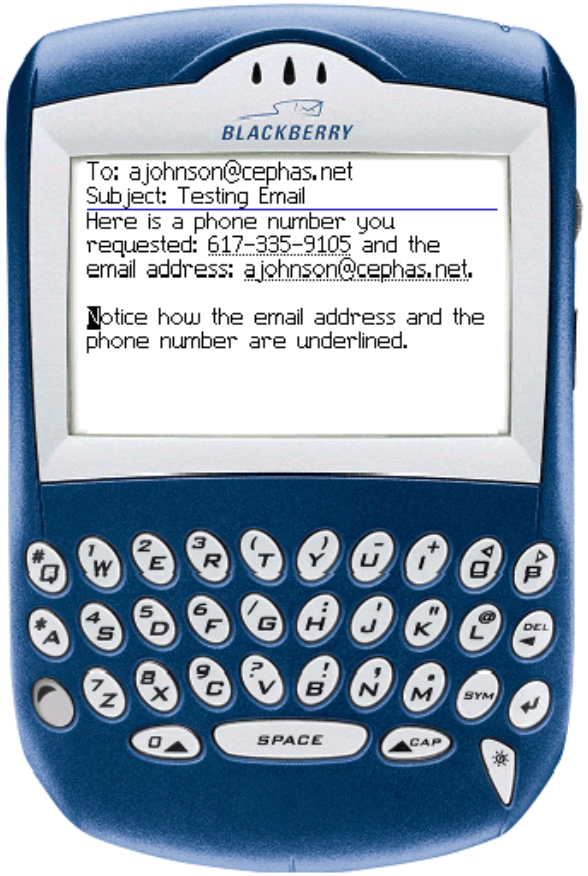
- In a CISC design the data processing operations can act on memory directly

- Design rules allow a RISC processor to be simpler, and the core can operate at higher clock frequencies
- In contrast, traditional CISC processors are more complex and operate at lower clock frequencies

ARM Based Products

Apple iPhone ARM11

Motorola Z8 Smart phone
ARM11



Blackberry ARM11

Nokia E90 Communicator ARM11

iPOD ARM7TDMI

Juice Box
Low cost Multimedia
player ARM7TDMI

Lego Mindstrome
Robot ARM7

Paison Series game
consoles ARM7TDMI

1.2.2 THE ARM DESIGN PHYLOSOPHY

ESTD : 2001
An Institute with a Difference

- Physical features that have driven the ARM processor design –
- Portable embedded systems require ***battery power***.
  - The ARM processor specially designed to be small –
    - to reduce power consumption and extend battery operation
      - —essential for applications such as mobile phones and personal digital assistants (PDAs)
- ***High code density***
  - Embedded systems have limited memory
    - due to cost and/or physical size restrictions
      - —useful for applications that have limited on-board memory, such as mobile phones and mass storage devices

Embedded systems are ***price sensitive***

- *Use slow and low-cost memory devices*
  - to get substantial savings
    - —essential for high-volume applications like digital cameras
- *Reduce the area of the die* taken up by the embedded processor
  - to reduced cost of the design and manufacturing for the end product

- ARM has incorporated ***hardware debug technology***
  - so that software engineers can view what is happening while the processor is executing code (visibility)
  - software engineers can resolve issues faster
- The ARM core is ***not a pure RISC architecture***
  - because of the constraints of its primary application—the embedded system
  - In some sense, the strength of the ARM core is that it does not take the RISC concept too far.

Instruction Set for Embedded Systems:

» The ARM instruction set differs from the pure RISC definition, in several ways, that make the ARM instruction set suitable for embedded applications

- ***Variable cycle execution***
- ***Inline barrel shifter***
- ***Thumb 16-bit instruction set***
- ***Conditional execution***
- ***Enhanced instructions***

Variable cycle execution

- *Variable cycle execution for certain instructions*—Not every ARM instruction executes in a single cycle
  - For example, load-store-multiple instructions vary in the number of execution cycles depending upon the number of registers being transferred
  - The transfer can occur on sequential memory addresses
  - Code density is also improved since multiple register transfers are common operations at the start and end of functions

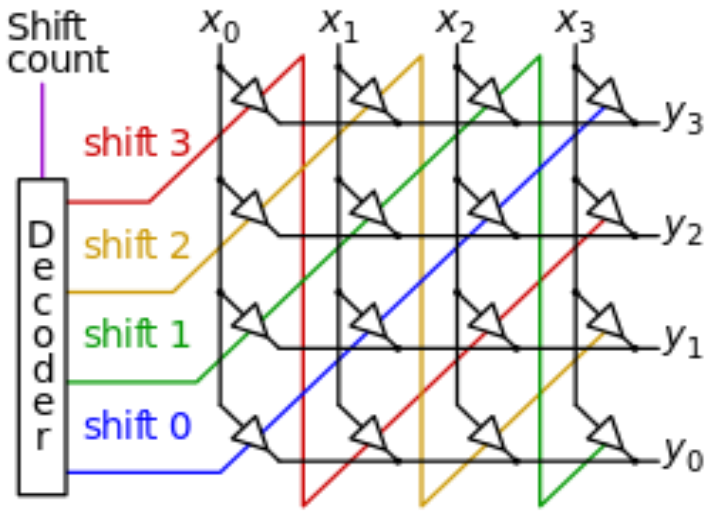
Inline barrel shifter

» *Inline barrel shifter leading to more complex instructions*—The inline barrel shifter is a hardware component that pre-processes one of the input registers before it is used by an instruction

This expands the capability of many instructions

» to improve core performance and code density

Barrel Shifter is a digital circuit that can shift a data word by a specified number of bits without the use of any sequential logic, only pure combinatorial logic



Thumb 16-bit instruction set

- *Thumb 16-bit instruction set*—ARM enhanced the processor core by adding a second 16-bit instruction set called Thumb
  - Thumb permits the ARM core to execute either 16- or 32-bit instructions
    - The 16-bit instructions improve code density by about 30% over 32-bit fixed-length instructions

Conditional execution

- *Conditional execution*—An instruction is only executed when a specific condition has been satisfied
  - This feature improves performance and code density by reducing branch instructions.

Enhanced instructions

- *Enhanced instructions*—The enhanced digital signal processor (DSP) instructions were added to the standard ARM instruction set
  - to support fast 16×16-bit multiplier operations
  - These instructions allow a faster-performing ARM processor

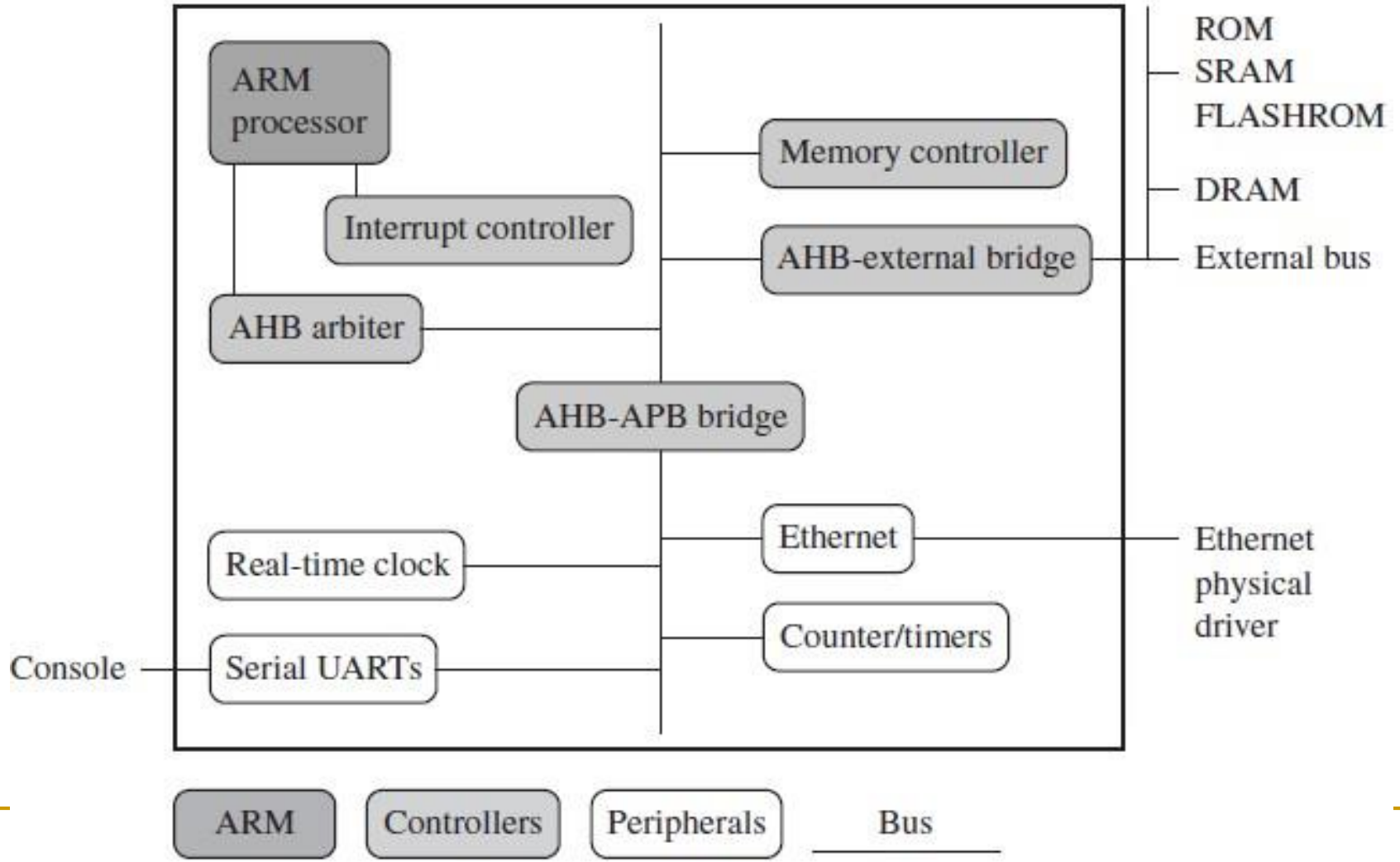
- These ***additional features*** have made the ARM processor one of the most commonly used 32-bit embedded processor cores

1.2.3 EMBEDDED SYSTEM HARDWARE

ESTD : 2001
An Institute with a Difference

- Embedded systems can control many different devices
  - from small sensors to real-time control systems
- All these devices use a combination of software and hardware components

An example of an ARM based embedded device, a Microcontroller



ESTD : 2001
An Institute with a Difference

Four Main Hardware Components –

ARM Processor, Controllers, Peripherals, Bus

1. The *ARM processor*
   - » Comprises a core
     - » the execution engine that processes instructions and manipulates data
     - » plus the surrounding components(memory management and caches) that interface it with a bus.
   - » Controls the embedded device
   - » Different versions of the ARM processor are available
     - » to suit the desired operating characteristics

2. ***Controllers***
   - » Coordinate important functional blocks of the system
   - » Two commonly found controllers are interrupt and memory controllers
3. The *peripherals*
   - » Provide all the input-output capability external to the chip
   - » Responsible for the uniqueness of the embedded device
4. A *bus* is used to
   - » Communicate between different parts of the device.

ARM Bus Technology

- Embedded devices use an on-chip bus that is internal to the chip
- allows different peripheral devices to be interconnected with an ARM core

- There are ***two different classes of devices*** attached to the bus:

1. The *ARM processor core* is a bus master—a logical device capable of initiating a data transfer with another device across the same bus.
2. *Peripherals* tend to be bus slaves—logical devices capable only of responding to a transfer request from a bus master device.

- A bus has ***two architecture*** levels:
  - A *physical level*—covers the electrical characteristics and bus width (16, 32, or 64 bits).
  - The *protocol*—the logical rules that govern the communication between the processor and a peripheral.

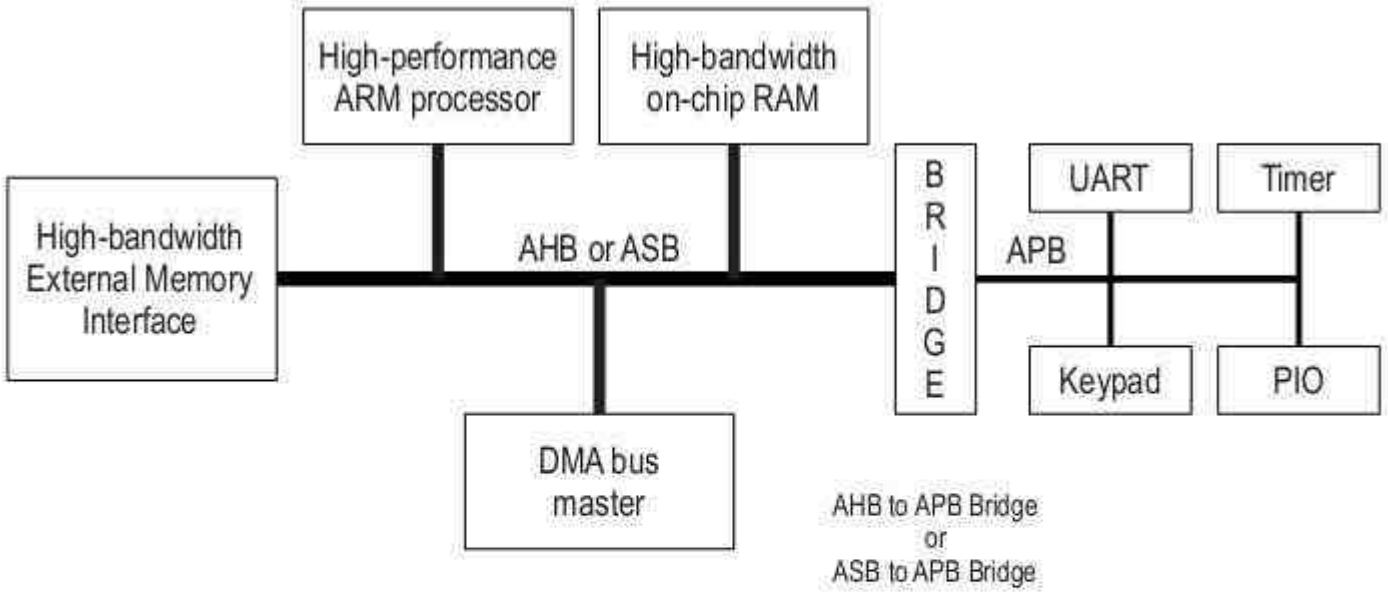
AMBA Bus Protocol

- The *Advanced Microcontroller Bus Architecture (AMBA)* was introduced in 1996 and has been widely adopted as the on-chip bus architecture used for ARM processors
- The first AMBA buses introduced were
  - the *ARM System Bus (ASB)* and
  - the *ARM Peripheral Bus (APB)*
  - Later ARM introduced the *ARM High Performance Bus* (AHB)

- Using AMBA,
  - peripheral designers can reuse the same design on multiple projects.
    - A peripheral can simply be bolted onto the on-chip bus without having to redesign an interface for each different processor architecture
    - This plug-and-play interface for hardware developers improves availability and time to market
- AHB provides higher data throughput than ASB because it is based on a centralized multiplexed bus scheme rather than the ASB bidirectional bus design. This change allows the AHB bus to run at higher clock speeds.

- ARM has introduced ***two variations*** on the AHB bus: ***Multi-layer AHB*** and ***AHB-Lite***
- The Multi-layer AHB bus allows multiple active bus masters
- AHB-Lite is a subset of the AHB bus and it is limited to a single bus master
- The example device shown in the above Figure has three buses:
  - an *AHB bus* for the high- performance peripherals
  - an *APB bus* for the slower peripherals
  - a third *bus for external peripherals*

ARM Bus Technology



AMBA AHB

* High performance
* Pipelined operation
* Multiple bus masters
* Burst transfers

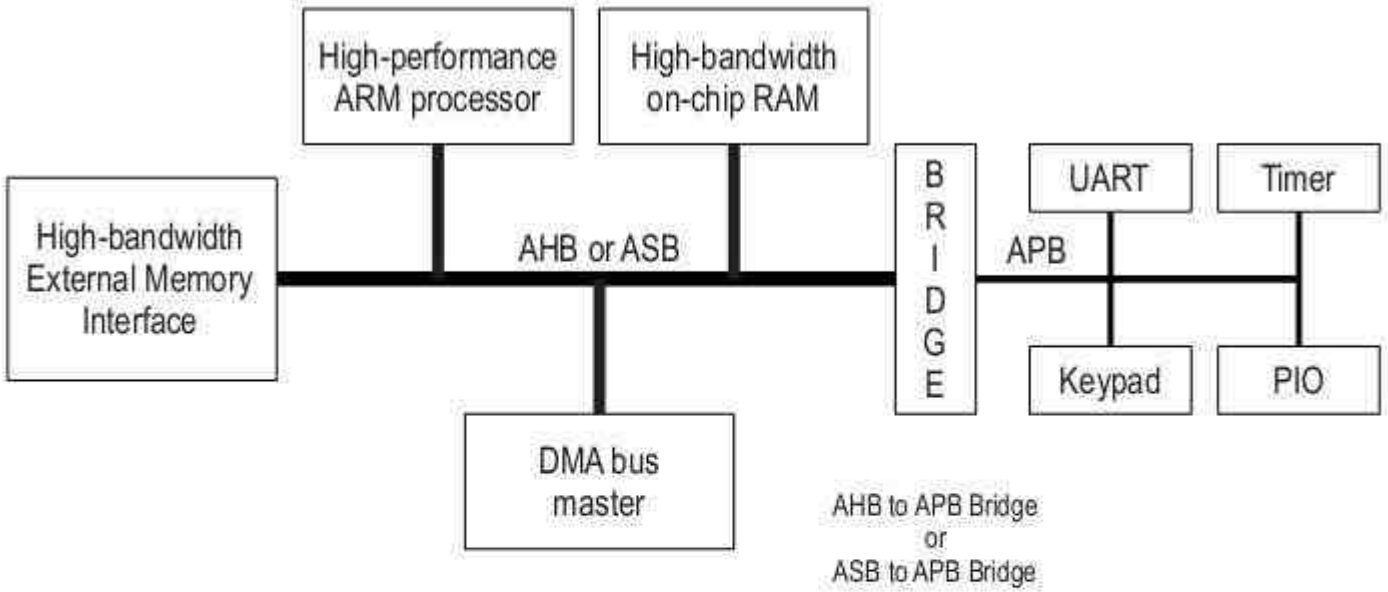
AMBA ASB

* High performance
* Pipelined operation
* Multiple bus masters

AMBA APB

* Low power
* Latched address and control
* Simple interface
* Suitable for many peripherals

ARM Bus Technology



AMBA AHB

* High performance
* Pipelined operation
* Multiple bus masters
* Burst transfers

AMBA ASB

* High performance
* Pipelined operation
* Multiple bus masters

AMBA APB

* Low power
* Latched address and control
* Simple interface
* Suitable for many peripherals

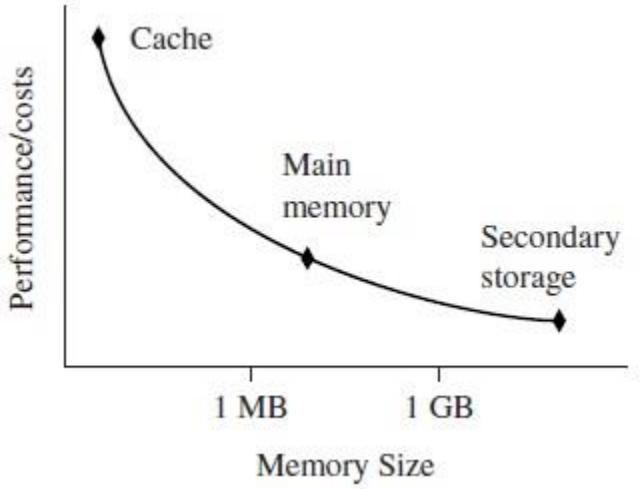
Memory Hierarchy

ESTD : 2001
An Institute with a Difference

» All computer systems have memory arranged in some form of hierarchy

» The following Figure shows the memory trade-offs: the fastest memory cache is physically located nearer the ARM processor core and the slowest secondary memory is set further away

» Generally the closer memory is to the processor core, the more it costs and the smaller its capacity

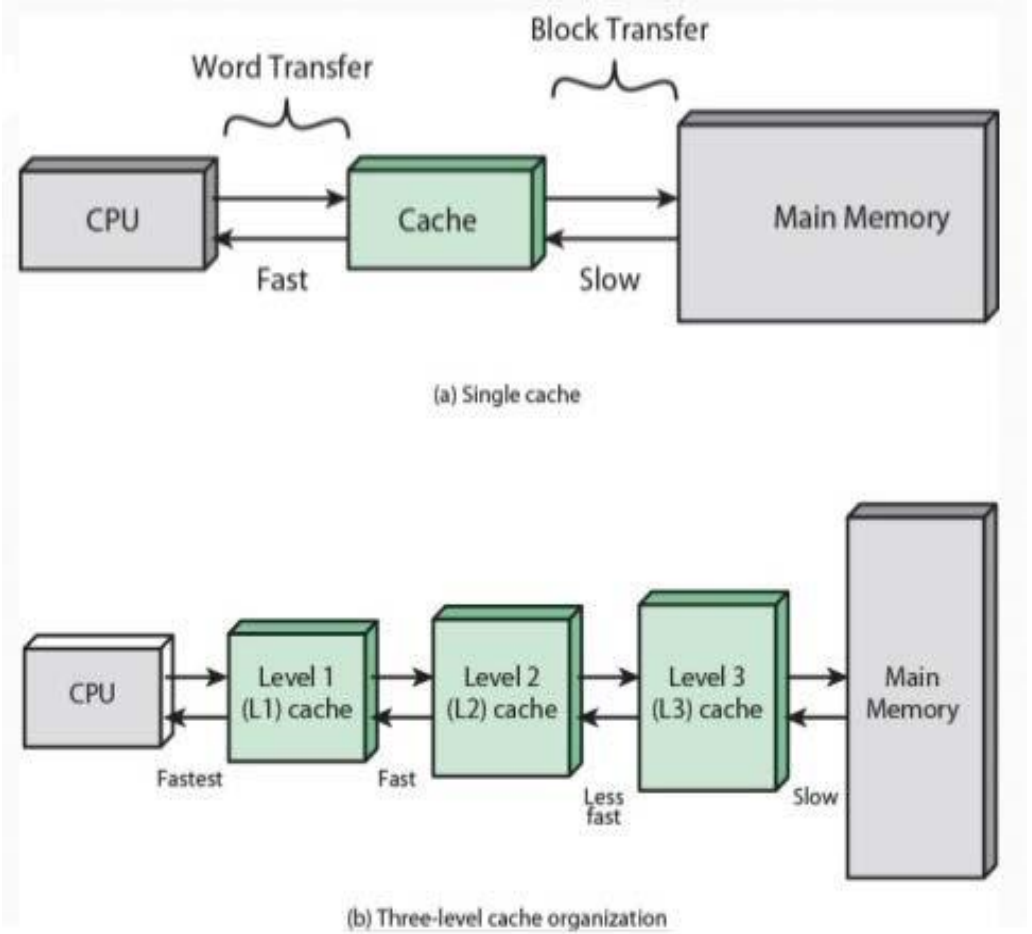


- The *cache* is
  - Placed between main memory and the core
  - Used to speed up data transfer between the processor and main memory
  - Provides an overall increase in performance but with a loss of predictable execution time
- Although the cache increases the general performance of the system, it does not help real-time system response
- The *main memory* is large—
  - Around 256 KB to 256 MB (or even greater), depending on the application—Generally stored in separate chips

Load and store instructions access the main memory unless the values have been stored in the cache for fast access

Secondary storage is the largest and slowest form of memory

» Examples – Hard disk drives and CD- ROM drives



Memory Width

ESTD : 2001
An Institute with a Difference

- The memory width is the number of bits the memory returns on each access—typically 8, 16, 32, or 64 bits
- The memory width has a direct effect on the overall performance and cost ratio
  - Lower bit memories are less expensive, but reduce the system performance
- The following Table summarizes theoretical cycle times on an ARM processor using different memory width devices

Instruction Size	8-bit Memory	16-bit Memory	32-bit Memory
ARM 32-bit	4 cycles	2 cycles	1 cycles
Thumb 16-bit	2 cycles	1 cycles	1 cycles

Memory Types – There are many ***different types of memory***:

» *Read-only memory (ROM)* is the least flexible of all memory types because it contains an image that is permanently set at production time and cannot be reprogrammed

» ROMs are used in high-volume devices that require no updates or corrections. Many devices also use a ROM to hold boot code

» *Flash ROM* can be written to as well as read, but it is slow to write so you shouldn't use it for holding dynamic data

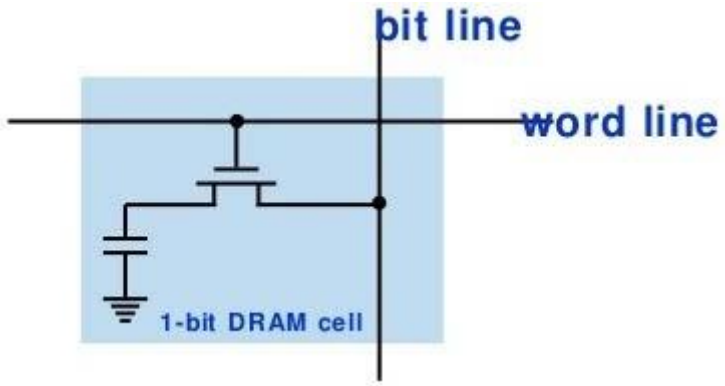
» Its main use is for holding the device firmware or storing long-term data that needs to be preserved after power is off.

» The erasing and writing of flash ROM are completely software controlled with no additional hardware circuitry required, which reduces the manufacturing costs

» *Dynamic random access memory (DRAM)* is the most commonly used RAM for devices

» It has the lowest cost per megabyte compared with other types of RAM

» DRAM is dynamic—it needs to have its storage cells refreshed and given a new electronic charge every few milliseconds, so you need to set up a DRAM controller before using the memory



- *Static random access memory (SRAM)* is faster than the more traditional DRAM, but requires more silicon area
- SRAM is static—the RAM does not require refreshing
- The access time for SRAM is considerably shorter than the equivalent DRAM because SRAM does not require a pause between data accesses
- But cost of SRAM is high
- *Synchronous dynamic random access memory (SDRAM)* is one of many subcategories of DRAM
- It can run at much higher clock speeds than conventional memory
- SDRAM synchronizes itself with the processor bus, because it is clocked
- Internally the data is fetched from memory cells, pipelined, and finally brought out on the bus in a burst

Dynamic RAM	Static RAM
1. Made up of Capacitors	1. Made up of Flip-flops
2. Data storage in the form of charge	2. Data storage in the form of voltage
3. Small in size and less expensive	3. Large in size and much expensive
4. High storage capacity	4. Low storage capacity
5. Slow, but consume less power	5. Fast, but consume more power
6. Data loses with time, so needs refreshing circuitry to maintain the charge in the capacitors for data	6. No data loses, does not require periodic refreshment to maintain data, data sustain with time
7. DRAM are used in Main memory	7. SRAM are used in Cache memory

Peripherals

» Embedded systems that interact with the outside world need some form of peripheral device.

» A ***peripheral device*** performs input and output functions for the chip by connecting to other devices or sensors that are off-chip

»Each peripheral device usually performs a single function and may reside on-chip

»Peripherals range from a simple serial communication device to a more complex 802.11 wireless device

- All ARM peripherals are *memory mapped*—the programming interface is a set of memory-addressed registers
- The address of these registers is an offset from a specific peripheral base address
- *Controllers* are specialized peripherals that implement higher levels of functionality within an embedded system
- Two important types of controllers are memory controllers and interrupt controllers

Memory Controllers:

- Memory controllers connect different types of memory to the processor bus
- On power-up, a memory controller is configured in hardware to allow certain memory devices to be active
  - These memory devices allow the initialization code to be executed
- Some memory devices must be set up by software; for example, when using DRAM, you first have to set up the memory timings and refresh rate before it can be accessed

Interrupt Controllers:

» When a peripheral or device requires attention, it raises an *interrupt* to the processor.

» An *interrupt controller* provides a programmable governing policy that allows software to determine which peripheral or device can interrupt the processor at any specific time by setting the appropriate bits in the interrupt controller registers

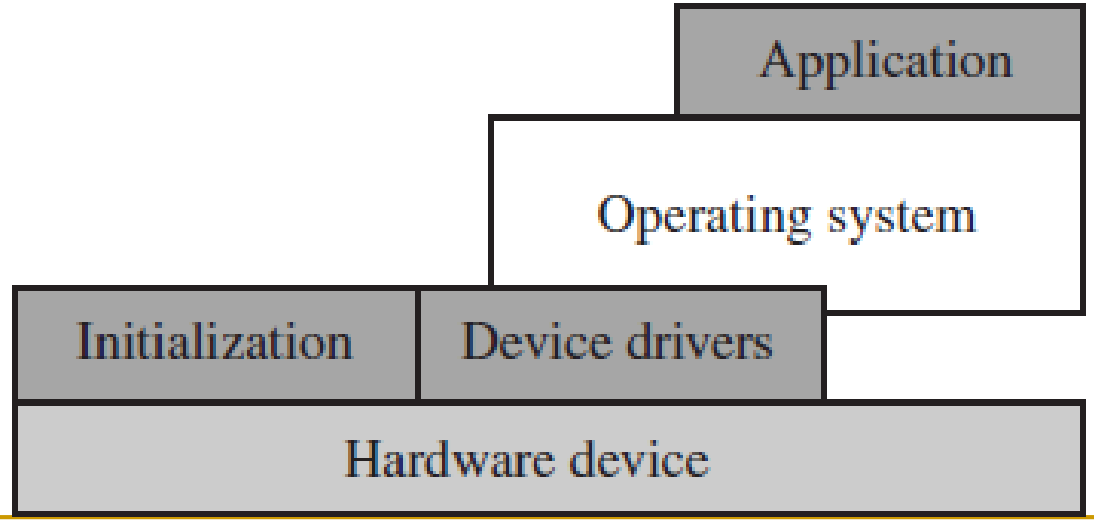
» There are *two types of interrupt controller* available for the ARM processor: the standard interrupt controller and the vector interrupt controller

- The ***standard interrupt controller*** sends an interrupt signal to the processor core when an external device requests servicing
  - It can be programmed to ignore or mask an individual device or set of devices.
    - The *interrupt handler* determines which device requires servicing by reading a device bitmap register in the interrupt controller
  - The *vector interrupt controller (VIC)* is more powerful than the standard interrupt controller, because it prioritizes interrupts and simplifies the determination of which device caused the interrupt
    - Depending on the type, the VIC will either call the standard interrupt exception handler, which can load the address of the handler

EMBEDDED SYSTEM SOFTWARE

ESTD : 2001
An Institute with a Difference

- An embedded system needs software to drive it
- The following Figure shows four typical software components required to control an embedded device



- The *initialization code*
  - first code executed on the board and is specific to a particular target or group of targets
  - sets up the minimum parts of the board before handing control over to the operating system
- The *operating system*
  - provides an infrastructure to control applications and manage hardware system resources
- The *device drivers*
  - provide a consistent software interface to the peripherals on the hardware device
- An *application*
  - performs one of the tasks required for a device
    - For example, a mobile phone might have a diary application
- There may be multiple applications running on the same device, controlled by the operating system

Initialization (Boot) Code

ESTD : 2001
An Institute with a Difference

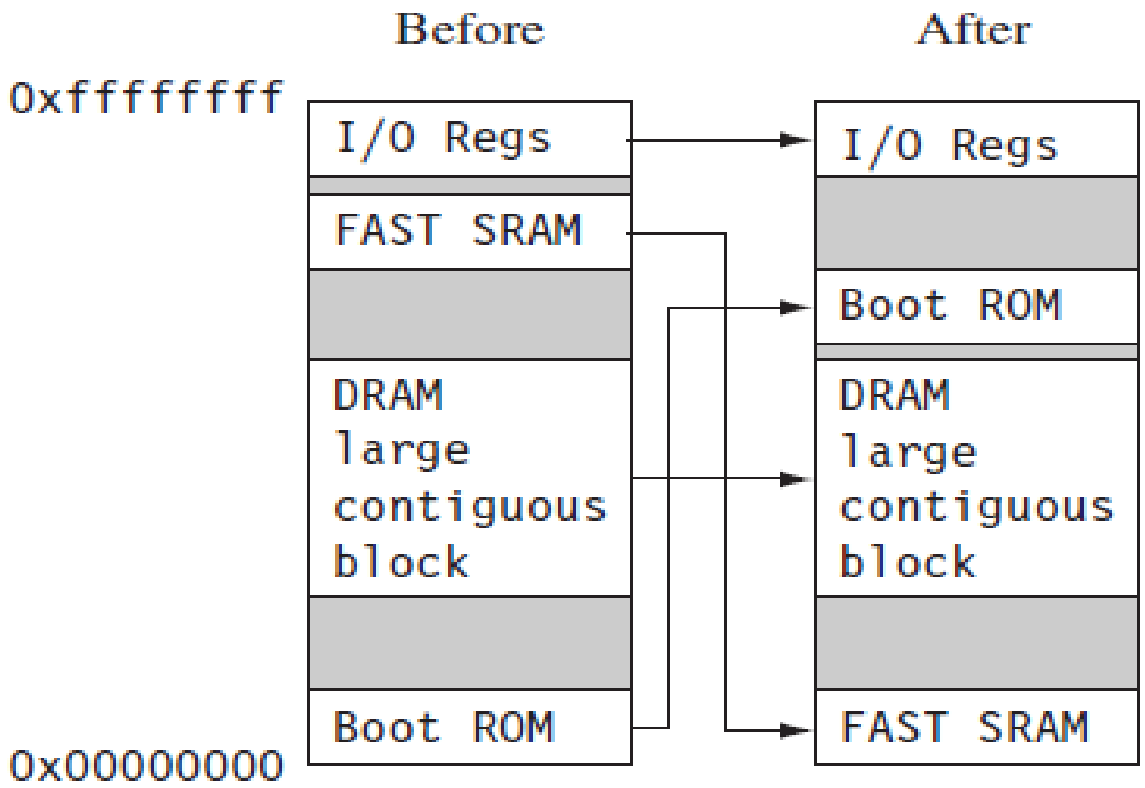
- Initialization code (or boot code) takes the processor from the reset state to a state where the operating system can run
- It usually configures the memory controller and processor caches and initializes some devices
- The initialization code handles a number of administrative tasks prior to handing control over to an operating system
  - We can group these different tasks into *three phases*: initial hardware configuration, diagnostics, and booting

1. ***Initial hardware configuration*** involves setting up the target platform, so that it can boot an image

» The target platform comes up in a standard configuration; but, this configuration normally requires modification to satisfy the requirements of the booted image

» For example, the memory system normally requires reorganization of the memory map

» *Initializing or organizing memory is an important part of the initialization code, because many operating systems expect a known memory layout before they can start*



- *Memory remapping allows the system to start the initialization code from ROM at power-up*
- *The initialization code then redefines or remaps the memory map to place RAM at address 0x00000000—an important step because then the exception vector table can be in RAM and thus can be reprogrammed*

2. ***Diagnostic***s are often embedded in the initialization code

» Diagnostic code tests the system by exercising the hardware target to check if the target is in working order

» It also tracks down standard system-related issues

» The primary purpose of diagnostic code is fault identification and isolation

3. *Booting* involves loading an image and handing control over to that image

» The boot process itself can be complicated if the system must boot different operating systems or different versions of the same operating system

» Booting an image is the final phase, but first you must load the image

» Loading an image involves anything from copying an entire program including code and data into RAM, to just copying a data area containing volatile variables into RAM

» Once booted, the system hands over control by modifying the program counter to point into the start of the image

Operating System

- The initialization process prepares the hardware for an operating system to take control.

- An operating system organizes the system resources: the peripherals, memory, and processing time

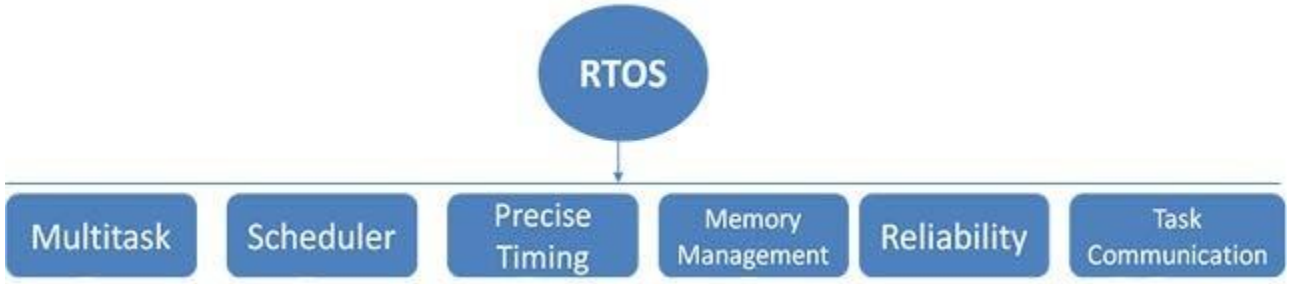
- ARM processors support over 50 operating systems
  - We can divide operating systems into *two main categories*:
    - real-time operating systems (RTOSs)
    - platform operating systems

1. *RTOSs* provide guaranteed response times to events

» Different operating systems have different amounts of control over the system response time

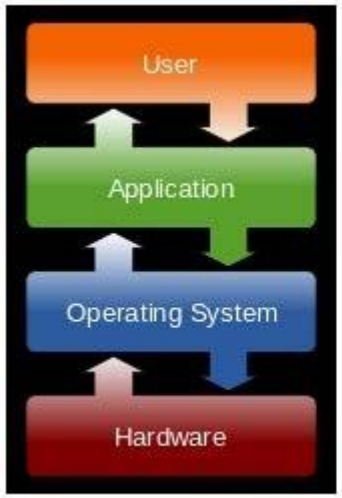
» A *hard real-time* application requires a guaranteed response to work at all

» A *soft real-time* application requires a good response time, but the performance degrades more gracefully if the response time overruns



» ***Platform operating systems*** require a memory management unit to manage large, non-real-time applications and tend to have secondary storage

» The Linux operating system is a typical example of a platform operating system



Applications

- The operating system schedules *applications*—code dedicated to handle a particular task
- An application implements a processing task; the operating system controls the environment
  - An embedded system can have one active application or several applications running simultaneously
- ARM processors are found in numerous market segments, including networking, auto-motive, mobile and consumer devices, mass storage, and imaging

» ARM processor is found in networking applications like home gateways, DSL modems for high-speed Internet communication, and 802.11 wireless communications

» The mobile device segment is the largest application area for ARM processors, because of mobile phones

» ARM processors are also found in mass storage devices such as hard drives and imaging products such as inkjet printers— applications that are cost sensitive and high volume.

» In contrast, ARM processors are not found in applications thatrequire leading-edge high performance. Because these applications tend to be low volume and high cost, ARM has decided not to focus designs on these types of applications.

Thank You