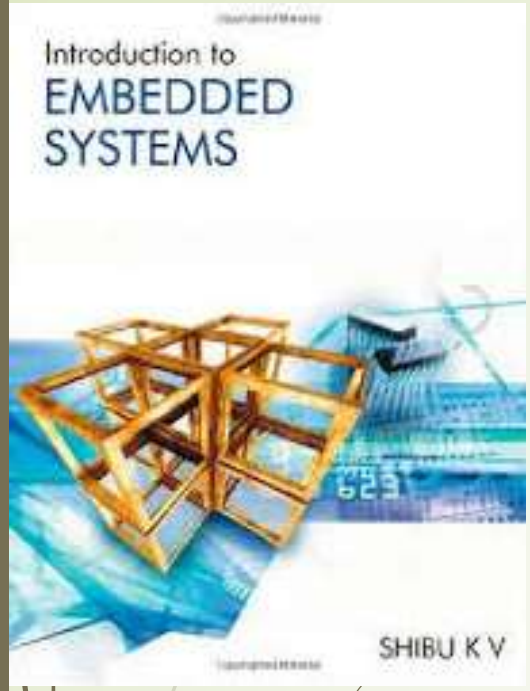


Introduction to

EMBEDDED SYSTEM

(2nd Edition)

SHIBU K V

Dr Moe Moe Myint

Department of Computer Engineering & Information Technology

Mandalay Technological University

www.slideshare.net/MoeMoeMyint

moemoemyint@moemyanmar.ml

moe2myint.mdy@gmail.com

drmoemoemyint.blogspot.com

Agenda

Learning Objectives

- Learn the building blocks of a typical Embedded System
- Learn about General Purpose Processors (GPPs), Application Specific Instruction Set Processors (ASIPs), Microprocessors, Microcontrollers, Digital Signal Processors, RISC & CISC processors, Harvad and Von-Neumann Processor Architecture, Big-endian v/s Little endian processors, Load Store operation and Instruction pipelining
- Learn about different PLDs like Complex Programmable Logic Devices (CPLDs), Field Programmable Gate Arrays (FPGAs), etc.

Cont'd

- Learn about the different memory technologies and memory types used in embedded system development
- Learn about Masked ROM (MROM), PROM, OTP, EPROM, EEPROM, and FLASH memory for embedded firmware storage
- Learn about Serial Access Memory (SAM), Static Random Access Memory (SRAM), Dynamic Random Access Memory (DRAM) and Nonvolatile SRAM (NVRAM)
- Understand the different factors to be considered in the selection of memory for embedded systems
- Understand the role of sensors, actuators and their interfacing with the I/O subsystems of an embedded system

Cont'd

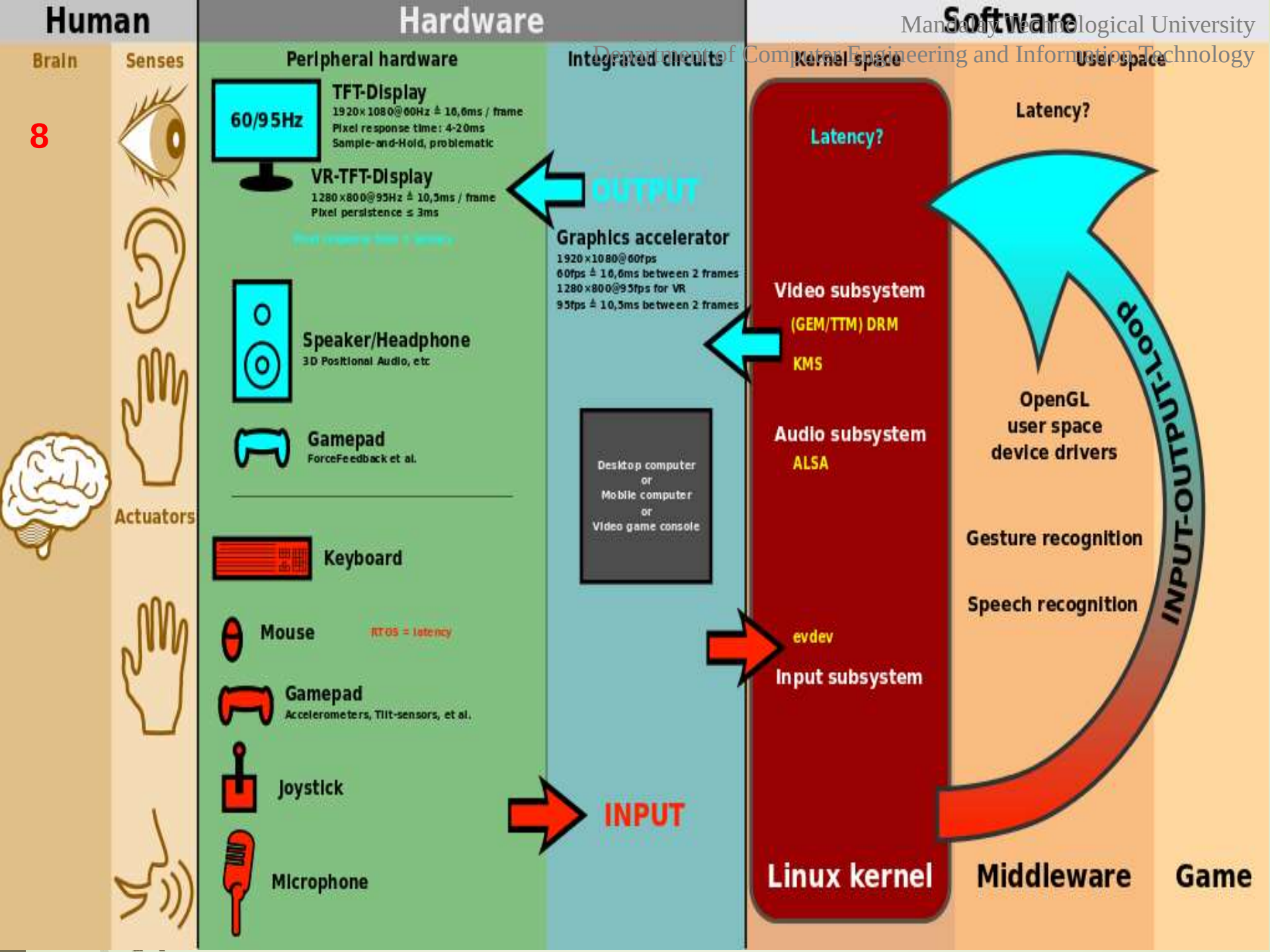
- Learn about the interfacing of LEDs, 7-segment LED Displays, Piezo Buzzer, Stepper Motor, Relays, Optocouplers, Matrix keyboard, Push button switches, Programmable Peripheral Interface Device (e.g. 8255 PPI), etc. with the I/O subsystem of the embedded system
- Learn about the different communication interfaces of an embedded system
- Understand the various chip level communication interfaces like I2C, SPI, UART, 1-wire, parallel bus, etc
- Understand the different wired and wireless external communication interfaces like RS-232C, RS-485, Parallel Port, USB, IEEE1394, Infrared (IrDA), Bluetooth, Wifi, ZigBee, GPRS, etc.
- Know what embedded firmware is and its role in embedded systems

Cont'd

- Understand the different system components like Reset Circuit, Brown-out protection circuit, Oscillator Unit, Real-Time Clock (RTC) and Watchdog Timer unit
- Understand the role of PCB in embedded systems

Typical Embedded System

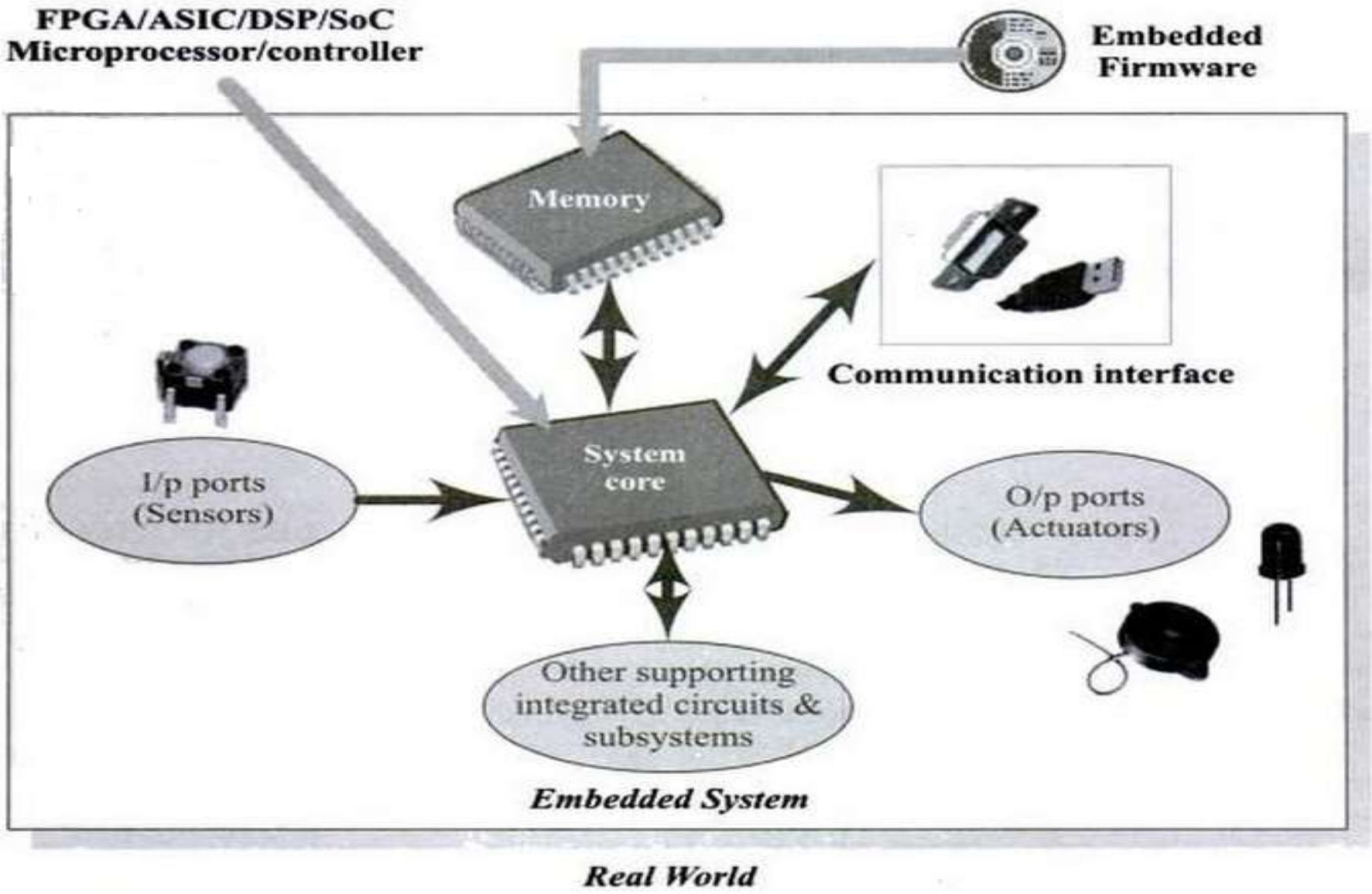
- A typical embedded system contains a single chip controller, which acts as the master brain of the system.
- The controller can be a Microprocessor or a microcontroller or a Field Programmable Gate Array (FPGA) device or a Digital Signal Processor (DSP) or an Application Specific Integrated Circuit (ASIC)/ Application Specific Standard Product (ASSP).
- Embedded hardware/software systems are basically designed to **regulate a physical variable** or to manipulate the state of some devices by sending some control signals to the Actuators or devices connected to the o/p ports of the system, in response to the input signals provided by the end users or Sensors which are connected to the input ports.


Human
Hardware
Software
Brain
Senses
Peripheral hardware
Integrated circuits
Kernel space
User space
8
60/95Hz
TFT-Display
1920x1080@60Hz ≙ 16,6ms / frame
Pixel response time: 4-20ms
Sample-and-Hold, problematic
VR-TFT-Display
1280x800@95Hz ≙ 10,3ms / frame
Pixel persistence ≤ 3ms
Latency?
Latency?
OUTPUT
Graphics accelerator
1920x1080@60fps
60fps ≙ 16,6ms between 2 frames
1280x800@95fps for VR
95fps ≙ 10,3ms between 2 frames
Video subsystem
(GEM/TTM) DRM
KMS
Speaker/Headphone
3D Positional Audio, etc
Gamepad
ForceFeedback et al.
Audio subsystem
ALSA
OpenGL
user space
device drivers
Desktop computer
or
Mobile computer
or
Video game console
Actuators
Keyboard
Gesture recognition
Speech recognition
Mouse
RTOS = latency
evdev
Input subsystem
Gamepad
Accelerometers, Tilt-sensors, et al.
Joystick
INPUT
INPUT-OUTPUT-LOOP
Microphone
Linux kernel
Middleware
Game

Cont'd

- Keyboards, push button switches, etc. are examples for common user interface **input** devices whereas LEDs, liquid crystal displays, piezoelectric buzzers, etc. are examples for common user interface **output** devices for a typical embedded system.
- **For example**, if the embedded system is designed for any handheld application, such as a mobile handset application, then the system should contain user interfaces like a keyboard for performing input operations and display unit for providing users the status of various activities in progress.

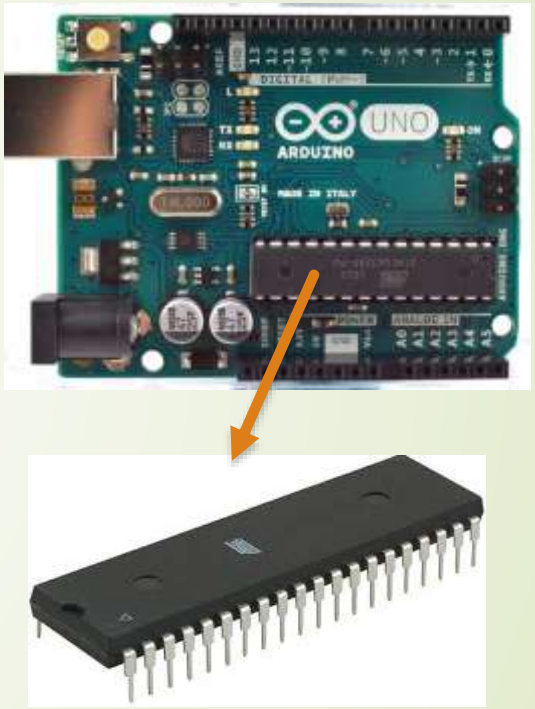
Elements of an Embedded System



2.1 Core of Embedded Systems

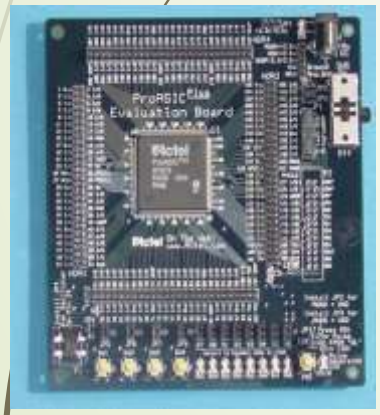
- The core of the embedded system falls into any of the following categories:

1. General Purpose and Domain Specific Processors
   i. Microprocessors
   ii. Microcontrollers
   iii. Digital Signal Processors



Cont'd

2. Application Specific Integrated Circuits (ASICs)
3. Programmable Logic Devices (PLDs)
4. Commercial off-the-shelf Components (COTS)



Merits, Drawbacks and Application Areas of Microcontrollers and Microprocessors

- **Microcontrollers** are designed to perform **specific tasks**. However, **Microprocessors** are designed to perform **unspecific tasks** like developing software, games, website, photo editing, creating documents, etc.

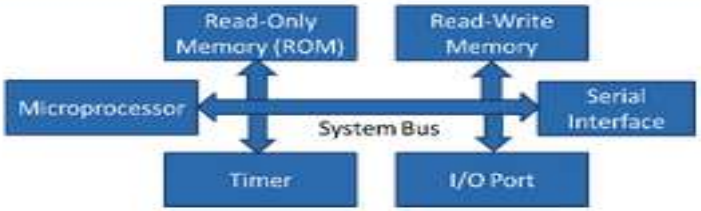
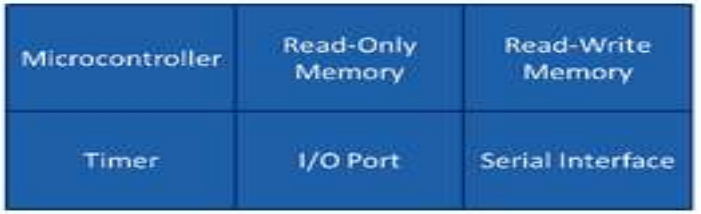
- Depending on the input, some processing for **microcontroller** needs to be done and output is **defined**. However, the relationship between input and output for **microprocessor** is **not defined**.

- Since the applications of **microcontroller** are very specific, they need **small resources** like RAM, ROM, I/O ports etc. and hence can be embedded on **a single chip.** **Microprocessors** need **high amount of resources** like RAM, ROM, I/O ports etc.

- The clock speed of Microprocessor is **quite high** as compared to the microcontroller. Whereas the **microcontrollers** operate from a **few MHz** (from 30 to 50 MHz), today's **microprocessor** operate above **1 GHz** as they perform **complex tasks**.

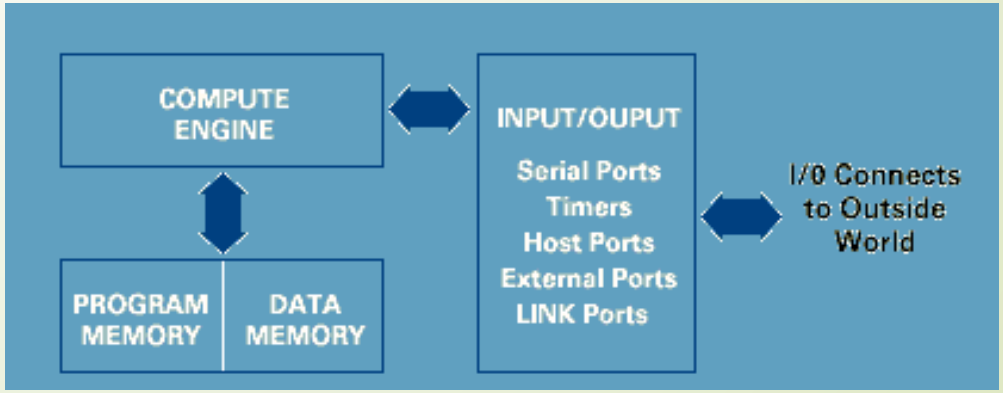
Cont'd

- **Microprocessor cannot be used stand alone**. They need other peripherals like RAM, ROM, buffer, I/O ports etc and hence a system designed around a microprocessor is quite costly.
- **Application areas of microcontroller**: Mobile phones, CD/DVD players, Washing machines, Cameras, Security alarms, microwave oven, etc.
- **Application areas of microprocessor**: Calculators, Accounting Systems, Games Machine, Complex Industrial Controllers, Data Acquisition Systems, Military applications, Communication systems, etc.

Microprocessor	Micro Controller
	
Microprocessor is heart of Computer system.	Micro Controller is a heart of embedded system.
It is just a processor. Memory and I/O components have to be connected externally	Micro controller has external processor along with internal memory and i/O components
Since memory and I/O has to be connected externally, the circuit becomes large.	Since memory and I/O are present internally, the circuit is small.
Cannot be used in compact systems and hence inefficient	Can be used in compact systems and hence it is an efficient technique
Cost of the entire system increases	Cost of the entire system is low
Due to external components, the entire power consumption is high. Hence it is not suitable to used with devices running on stored power like batteries.	Since external components are low, total power consumption is less and can be used with devices running on stored power like batteries.
Most of the microprocessors do not have power saving features.	Most of the micro controllers have power saving modes like idle mode and power saving mode. This helps to reduce power consumption even further.
Since memory and I/O components are all external, each instruction will need external operation, hence it is relatively slower.	Since components are internal, most of the operations are internal instruction, hence speed is fast.
Microprocessor have less number of registers, hence more operations are memory based.	Micro controller have more number of registers, hence the programs are easier to write.
Microprocessors are based on von Neumann model/architecture where program and data are stored in same memory module	Micro controllers are based on Harvard architecture where program memory and Data memory are separate
Mainly used in personal computers	Used mainly in washing machine, MP3 players

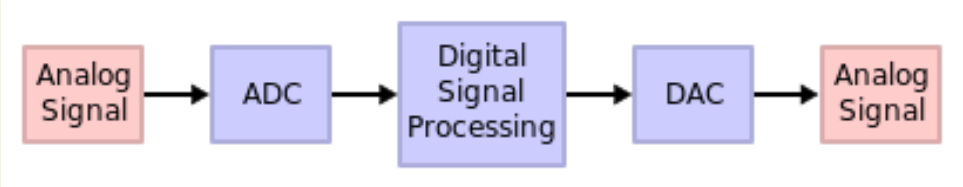
Digital Signal Processors

- DSPs are **powerful special purpose 8/16/32 bit microprocessors** designed specifically to meet the computational demands and power constraints of today's embedded audio, video, and communications **applications**.
- Digital signal processors are **2 to 3 times faster** than the general purpose microprocessors in signal processing applications.
- A typical digital signal processor incorporates the following key units:
  - i. **Program Memory** : Memory **for storing the program** required by DSP to process the data.
  - ii. **Data Memory** : Working memory **for storing temporary variables/information and data/signal** to be processed.
  - iii. **Computational Engine** : **Performs the signal/math processing ,** accessing the program from the Program Memory and the data from the Data Memory.
  - iv. **I/O Unit** : Acts as **an interface between the outside world and DSP**. It is responsible for capturing signals to be processed and delivering the processed signals.



Cont'd

- **Application areas** : Audio video signal processing, telecommunication and multimedia applications.
- DSP employs **a large amount of real-time calculations, Sum of products (SOP) calculation, convolution, fast Fourier transform (FFT), discrete Fourier transform (DFT),** etc. are some of the operations performed by digital signal processors.

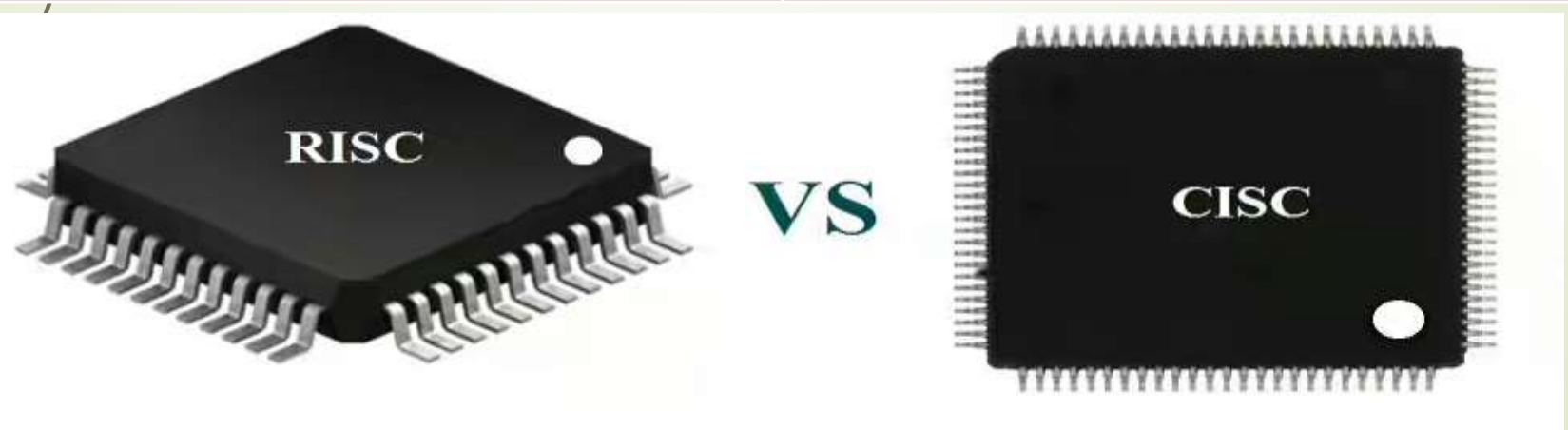


RISC vs CISC Processors/Controllers

RISC	CISC
Lesser number of instructions	**Greater number of Instructions**
Instruction pipelining and increased execution speed	Generally **no instruction pipelining feature**
Orthogonal instruction set	**Non-orthogonal** instruction set
Operations are performed **on registers only**, the only memory operations are load and store.	Operations are performed **on registers or memory** depending on the instruction.
A large number of registers are available.	**Limited** number of general purpose registers.
Programmer **needs to write more code to execute a task** since **the instructions are simpler ones**.	Instructions are like macros in **C language**. A programmer can achieve the desired functionality with a single instruction which in turn provides the effect of using **more simpler single instructions** in RISC.

Cont'd

RISC	CISC
Single, fixed length instructions	**Variable length instructions**
Less silicon usage and pin count	**More** silicon usage since more additional decoder logic is required to implement the complex instruction decoding.
With **Harvard Architecture**	Can be **Harvard or Von-Neumann Architecture**



Big-Endian vs. Little-Endian Processors/Controllers

- **Endianness** specifies the order in which a sequence of bytes are stored in computer memory.
- **Little-endian** is an order in which the "little end"/ the lower-order byte of the data (least significant value in the sequence) is stored in memory at the lowest address. (The little end comes first.)
- For example, a 4 byte long integer **Byte3, Byte2, Byte1, Byte0** will be stored in the memory as shown below:

Cont'd

Base Address+0	Byte 0	Byte 0	0x20000 (Base Address)
Base Address+1	Byte 1	Byte 1	0x20001 (Base Address+1)
Base Address+2	Byte 2	Byte 2	0x20002 (Base Address+2)
Base Address+3	Byte 3	Byte 3	0x20003 (Base Address+3)

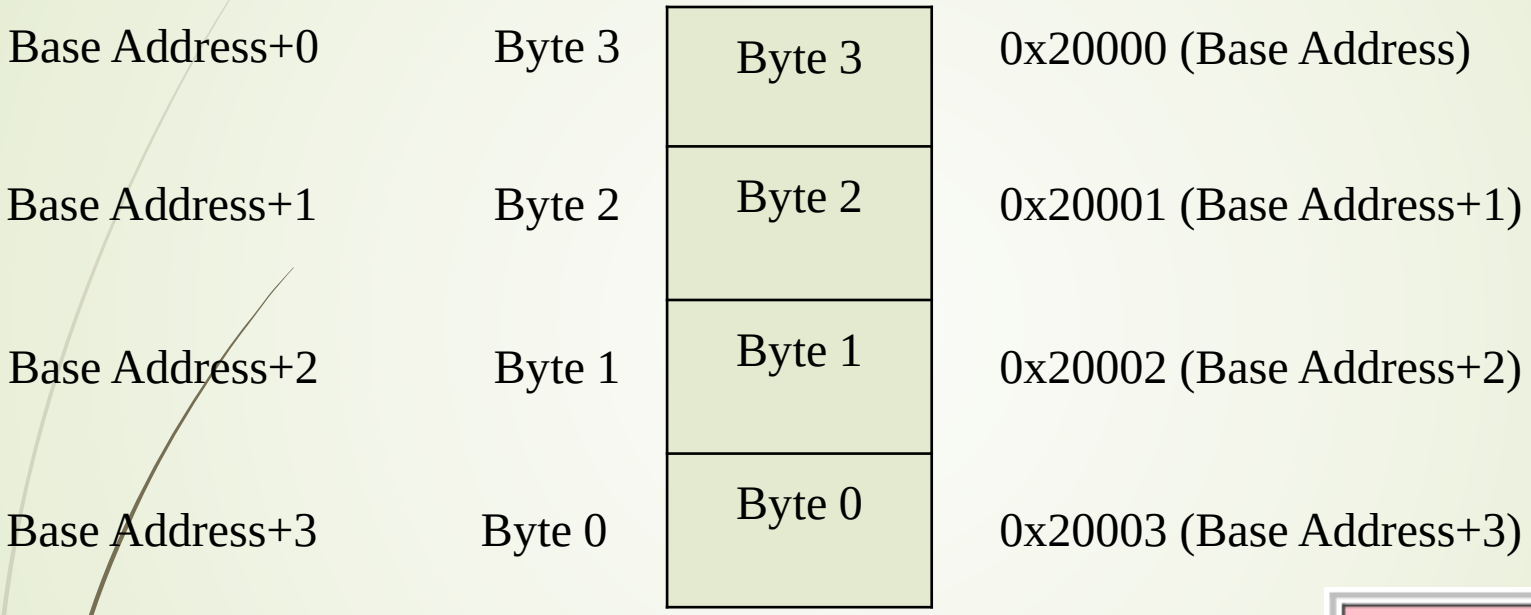
Example : **90AB12CD** (Hexadecimal)

Address	Value
1000	CD
1001	12
1002	AB
1003	90

Cont'd

- **Big-endian** is an order in which the "big end" / the higher-order byte of the data (most significant value in sequence) is stored in memory at the lowest address. (The big end comes first.)
- For example, a 4 byte long integer **Byte3, Byte2, Byte1, Byte0** will be stored in the memory as shown below:

Cont'd



Example : **90AB12CD** (Hexadecimal)

Address	Value
1000	90
1001	AB
1002	12
1003	CD

Understanding Test Questions II

1. Embedded hardware/software systems are basically designed to

   (a) Regulate a physical variable (b) Change the state of some devices

   (c) Measure/Read the state of a variable/device (d) Any/All of these

2. Little Endian processors

   (a) Store the lower-order byte of the data at the lowest address and the higher-order byte of the data at the highest address of memory

   (b) Store the higher-order byte of the data at the lowest address and the lower-order byte of the data at the highest address of memory

   (c) Store both higher order and lower order byte of the data at the same address of memory

   (d) None of these

3. The instruction set of RISC processor is

   (a) Simple and lesser in number (b) Complex and lesser in number

   (c) Simple and larger in number (d) Complex and larger in number

Cont'd

4. Which of the following is true about CISC processors?

   (a) The instruction set is non-orthogonal

   (b) The number of general purpose registers is limited.

   (c) Instructions are like macros in C language

   (d) Variable length instructions

   (e) All of these (f) None of these

5. Which of the following processor architecture supports easier instruction pipelining?

   (a) Harvard (b) Von Neumann (c) Both of them (d) None of these

6. An integer variable with value 255 is stored in memory location at 0x8000. The processor word length is 8 bits and the processor is a big endian processor. The size of integer is considered as 4 bytes in the system. What is the value held by the memory location 0x8000?

   (a) 0 x FF (b) 0 x 00 (c) 0 x 01 (d) None of these

Reviewed Questions II

1. Explain the components of a typical embedded system in detail.
2. Which are the components used as the core of an embedded system? Explain the merits, drawbacks, if any, and the applications/domains where they are commonly used.
3. What is the difference between microprocessor and microcontroller? Explain the role of microprocessors and controllers in embedded system design?
4. What is Digital Signal Processor (DSP)? Explain the role of DSP in embedded system design?
5. What is the difference between RISC and CISC processors? Give an example for each.
6. What is the difference between big-endian and little-endian processors? Give an example of each.

Assignment II

What are the components used as the core of an embedded system? Among them, explain the merits, drawbacks, if any, and the applications/ domains of Application Specific Integrated Circuits (ASICs), Programmable Logic Devices and Commercial Off-the-Shelf Components (COTS) where they are commonly used.

Remark: for all groups

Deadline : 26.12.17 (Coming Tuesday)

- Only Original Owner has full rights reserved for copied images.
  - This PPT is only for fair academic use.
  - Coming soon for chapter 2 (2^{nd} portion)

Thank you !!!