Cyber Physical Systems (CPS)
UNIT-2

Prof. Narasimha Swamy S
Department of AIML
RV College of Engineering
Bengaluru-59

UNIT-II

Outline

→ Introduction

→ Concepts

- Processors
- Basic System Architecture
- Buses (System Buses)
- Processor Operation
- ALU
- Interrupts
- CISC and RISC
- Digital Signal Processor

→ Memory

- RAM
- ROM
- EPROM
- EEROM
- Flash

→ Input/ Output

- Programmed I/O
- Interrupt-driven I/O
- Direct Memory Access (DMA)

→ Direct Memory Access

→ Embedded Computer Architecture

Introduction

What is Computer ?

Computer is a electronic machine, which holds the data and manipulate the data

- Types of Computers
  - Desktop Computers/ Laptops etc.
  - Embedded Computers
  - Smart/ mobile Phones

Introduction (Contd.)

Desktop Computer

A desktop computer is a general-purpose programmable computer, where the user can choose what applications to run on that computer

Embedded Computer

- Typically, Embedded computer is a Single-Function Device, whose hardware and software is designed for one specific purpose
- Ask yourself How many computers we have in our Homes ?
  - We may get the answer as One or Two
  - In fact, we may have 30 or more, hidden inside his TVs, VCRs, DVD players, remote controls, washing machines, cell phones, air conditioners, game consoles, ovens, toys, and other devices

Introduction (Contd.)

Computer	Embedded Computer
Desktop computers can run a variety of application programs	Embedded computer is normally dedicated to a specific task
Desktop computers generally comprise of Operating system, and allows to install application software's	The embedded computer may or may not have an operating system, and rarely does it provide the user with the ability to arbitrarily install new software
Architecture is complex	Embedded hardware is often much simpler
Desktop computers need the multiple chips	An embedded computer may be implemented in a single chip with just a few support components
Flexibility is high	Lower Flexibility
Higher Computing and Processing power	Less computing and processing power

Concepts

- A computer is a machine designed to process, store, and retrieve data
- Data may be numbers in a spreadsheet, characters of text in a document, dots of color in an image, waveforms of sound, or the state of some system, such as an air conditioner or a CD player
- All data is stored in the computer as numbers (0's and 1's)
- The computer manipulates the data by performing operations on the numbers.
  - Displaying an image on a screen is accomplished by moving an array of numbers to the video memory, each number representing a pixel of color
  - To play an MP3 audio file, the computer reads an array of numbers from disk and into memory, manipulates those numbers to convert the compressed audio data into raw audio data, and then outputs the new set of numbers (the raw audio data) to the audio chip
- Everything that a computer does, from web browsing to printing, involves moving and processing numbers

Concepts (Contd.)

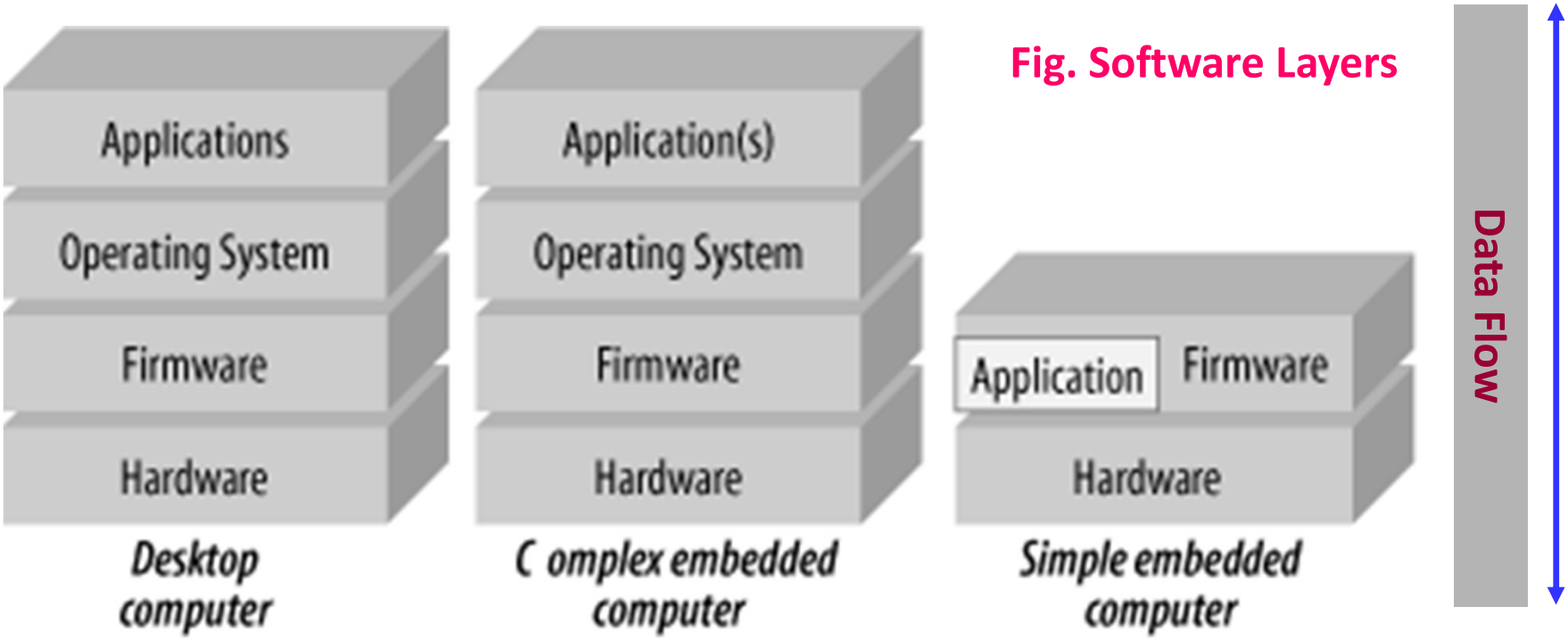


Fig. Software Layers

Windows Linux MAC	RaspberryPi Jetson Nano	NodeMCU Arduino RaspberryPi PICO

Concepts (Contd.)

- At the lowest level, there are programs that are run by the processor when the computer first powers up.
  - These programs initialize the other hardware subsystems to a known state and configure the computer for correct operation. This software, because it is permanently stored in the computer's memory, is known as firmware

- The bootloader is in the firmware.
  - The bootloader is a special program run by the processor that reads the operating system from disk (or nonvolatile memory or network interface) and places it in memory so that the processor may then run it. The bootloader is present in desktop computers and workstations, and may be present in some embedded computers

Concepts (Contd.)

- The operating system controls the operation of the computer. It organizes the use of memory and controls devices such as the keyboard, mouse, screen, disk drives, and so on
- It (OS) is also the software that often provides an interface to the user, enabling her to run application programs and access her files on disk
- The operating system typically provides a set of software tools for application programs, providing a mechanism by which they too can access the screen, disk drives, and so on
- Not all embedded systems use or even need an operating system. Often, an embedded system will simply run code dedicated to its task, and the presence of an operating system
- In other instances, such as network routers, an operating system provides necessary software integration and greatly simplifies the development process

Concepts (Contd.)

- At the highest level, the application software constitutes the programs that provide the functionality of the computer. Everything below the application is considered system software
- For embedded computers, the boundary between application and system software is often blurred

Concepts (Contd.)

Processors

- The processor is the most important part of a computer, the component around which everything else is centered. In essence, the processor is the computing part of the computer
- A processor is an electronic device capable of manipulating data (information) in a way specified by a sequence of instructions. The instructions are also known as opcodes or machine code
- A sequence of instructions is a machine-code program. Each type of processor has a different instruction set (SISC and RISC), meaning that the functionality of the instructions (and the bit patterns that activate them) varies.
- Processor instructions are often quite simple, such as "add two numbers" or "call this function"

Concepts (Contd.)

Basic System Architecture

- The processor alone is incapable of successfully performing any tasks.
- Processor requires memory (for program and data storage), support logic, and at least one I/O device
- Input/ Output devices are used to transfer data between the computer and the outside world

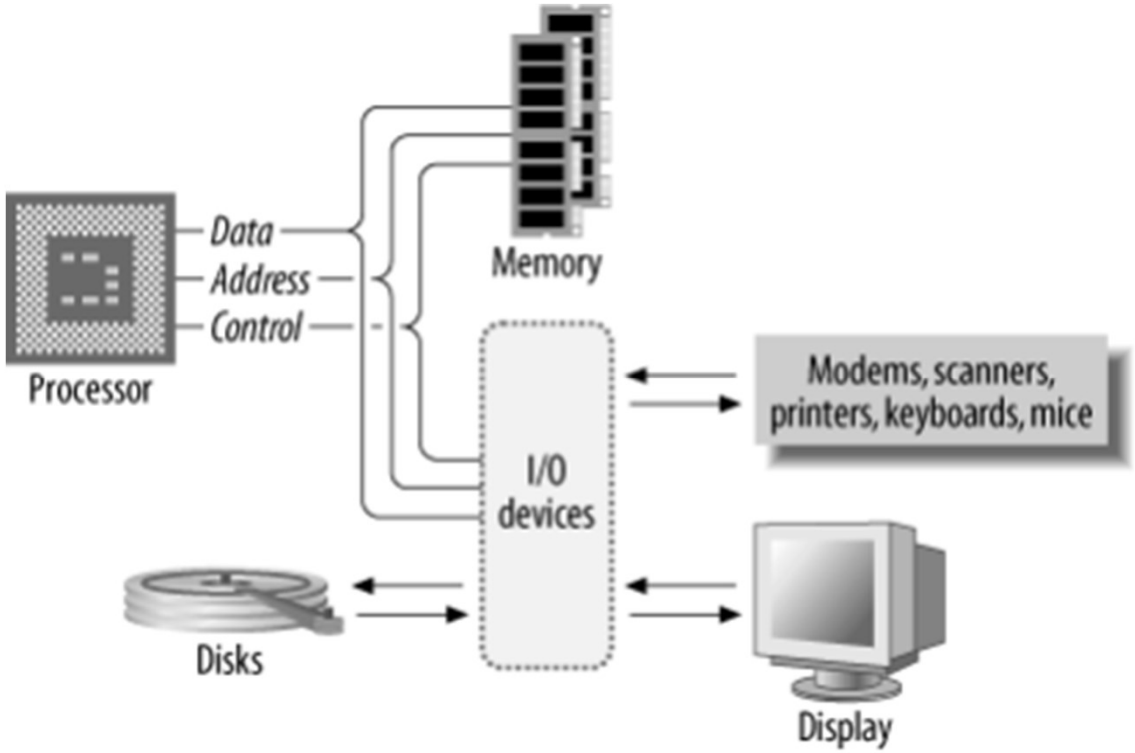


Fig. Basic Computer System Architecture

Concepts (Contd.)

Basic System Architecture

- A **microprocessor** is a processor implemented (usually) on a single, integrated circuit
  - Common microprocessors in use today are the Intel Pentium series, Freescale/IBM PowerPC, MIPS, ARM, and the Sun SPARC, among others. A microprocessor is sometimes also known as a CPU (Central Processing Unit)
- A **microcontroller** is a processor, memory, and some I/O devices contained within a single, integrated circuit, and intended for use in embedded systems
  - The buses that interconnect the processor with its I/O exist within the same integrated circuit
  - The range of available microcontrollers is very broad. They range from the tiny Peripheral Interface Controllers (PICs) and Advanced Virtual RISC micro-controller (AVRs) to PowerPC processors with inbuilt I/O, intended for embedded applications

Concepts (Contd.)

Basic System Architecture

- The memory of the computer system contains both the instructions that the processor will execute and the data it will manipulate
- The memory of a computer system is never empty. It always contains something, whether it be instructions, meaningful data, or just the random garbage that appeared in the memory when the system powered up
- The Instructions are read (fetched) from memory, while data is both read from and written to memory

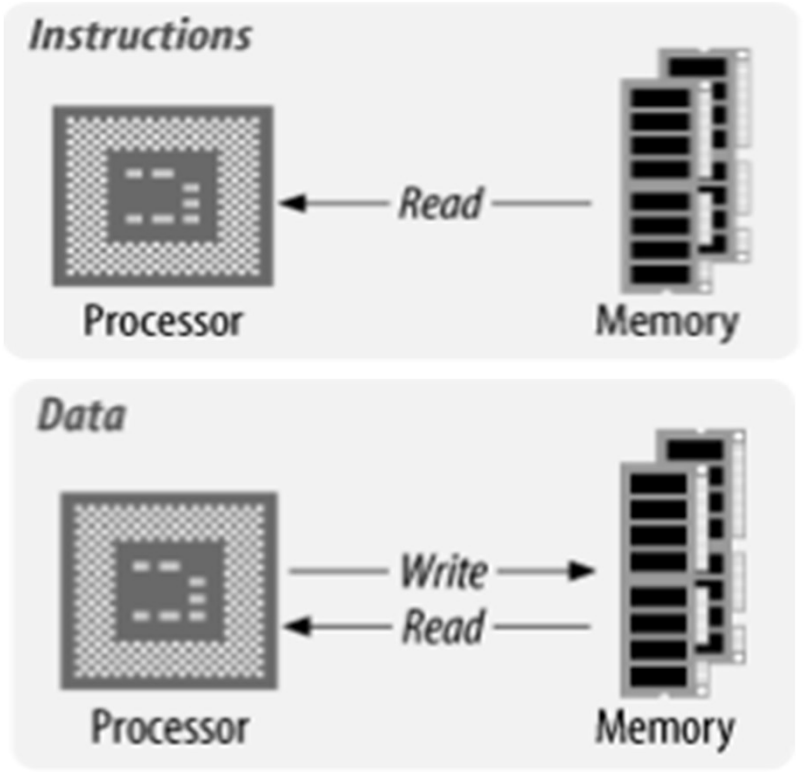
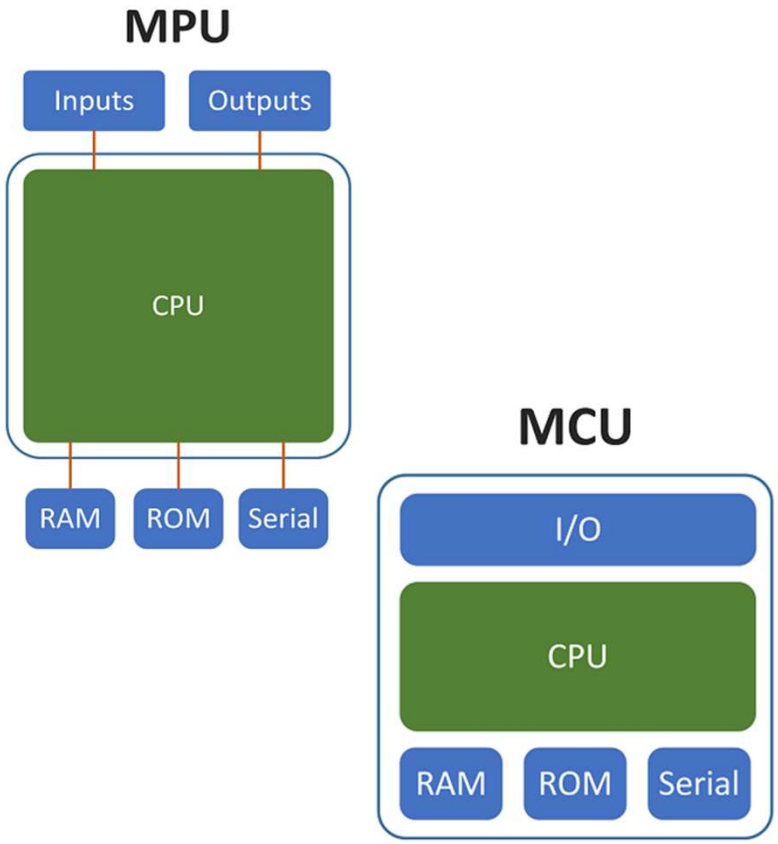


Fig. Data Flow

Concepts (Contd.)

Difference Between MPU and MCU



Microprocessor	Microcontroller
Contains only CPU; RAM, ROM, I/O, timers are separately Interfaced	CPU, RAM, ROM, I/O and Timers are on Single Chip
Designer decides on the amount of ROM, RAM and I/O Ports	Fix amount of on-chip ROM, RAM, I/O Ports
High Cost	Low Cost
Multi-purpose	Single Purpose
High Speed	Low Speed
High Power consumption	Low Power consumption
Ex: 8085, 8086	Ex: 8051, AVR

Concepts (Contd.)

Processor Operation

- There are six basic types of access that a processor can perform with external chips. The processor can write data to memory or write data to an I/O device, read data from memory or read data from an I/O device, read instructions from memory, and perform internal manipulation of data within the processor
- In many systems, writing data to memory is functionally identical to writing data to an I/O device
- Similarly, reading data from memory constitutes the same external operation as reading data from an I/O device, or reading an instruction from memory
- The processor makes no distinction between memory and I/O
- The internal data storage of the processor is known as its registers
- The processor has a limited number of registers, and these are used to hold the current data/operands that the processor is manipulating

Concepts (Contd.)

Buses / System Buses

- A bus is a physical group of signal lines that have a related function. Buses allow for the transfer of electrical signals between different parts of the computer system and thereby transfer information from one device to another
- The "width" of a bus is the number of signal lines dedicated to transferring information. For example, an 8-bit-wide bus transfers 8 bits of data in parallel
- The majority of microprocessors available today (with some exceptions) use the three-bus system Architecture
- The three buses are the address bus, the data bus, and the control bus

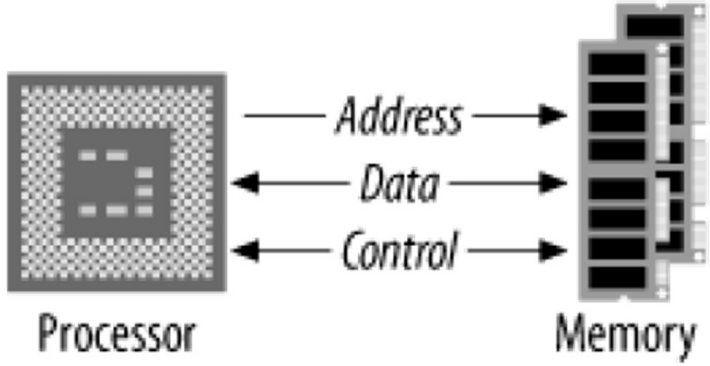


Fig. Three-bus system Architecture

Concepts (Contd.)

Buses / System Buses

- The data bus is bidirectional, the direction of transfer being determined by the processor
- The address bus carries the address, which points to the location in memory that the processor is attempting to access
- The control bus carries information from the processor about the state of the current access, such as whether it is a write or a read operation
  - The control bus can also carry information back to the processor regarding the current access, such as an address error
- The control bus may consist of output signals such as read, write, valid address, etc. A processor usually has several input control lines too, such as reset, one or more interrupt lines, and a clock input

Concepts (Contd.)

Von Neumann Machine/ Architecture

- In this architecture where the data and programs are subjected to shared memory i.e., are stored in the same memory block
- The Von Neumann architecture is also known as Princeton Architecture was proposed by John Von Neumann in the year 1945
- All the modern computers follow Von Neumann architecture
- Von Neumann computers are what can be termed control-flow computers. The steps taken by the computer are governed by the sequential control of a program.
- Typical Characteristics of Von Neumann Architecture

1. No real difference between data and instructions
2. Data has no inherent meaning
3. Data and instructions share the same memory
4. Memory is a linear (one-dimensional) array of storage locations

Concepts (Contd.)

Von Neumann Machine/ Architecture

1. No real difference between data and instructions

   A processor can be directed to begin execution at a given point in memory, and it has no way of knowing whether the sequence of numbers beginning at that point is data or instructions

2. Data has no inherent meaning

   There is nothing to distinguish between a number that represents a dot of color in an image and a number that represents a character in a text document. Meaning comes from how these numbers are treated under the execution of a program.

Concepts (Contd.)

Von Neumann Machine/ Architecture

3. Data and instructions share the same memory

To the compiler, the compiled program is just data, and it is treated as such. It is a program only when the processor begins execution. Similarly, an operating system loading an application program from disk does so by treating the sequence of instructions of that program as data

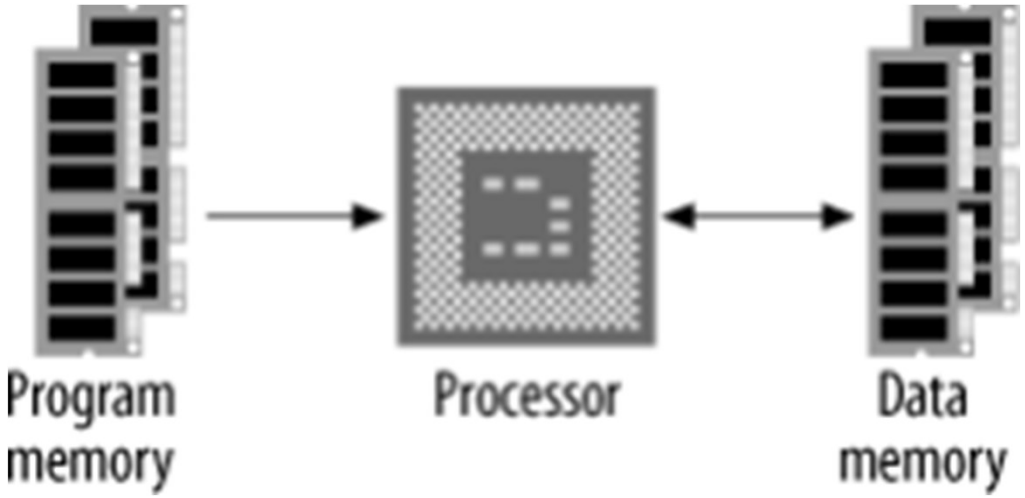
4. Memory is a linear (one-dimensional) array of storage locations

The processor's memory space may contain the operating system, various programs, and their associated data, all within the same linear space. Each location in the memory space has a unique, sequential address.

Concepts (Contd.)

Harvard Architectures

- In this architecture, instructions and data have different memory spaces with separate address, data, and control buses for each memory space



- This has a number of advantages in that instruction and data fetches can occur concurrently, and the size of an instruction is not set by the size of the standard data unit

Concepts (Contd.)

Harvard Architecture

- The main deviation from this is the Harvard architecture, in which instructions and data have different memory spaces with separate address, data, and control buses for each memory space

Concepts (Contd.)

Difference between Von Neumann and Harvard Architecture

Von – Neumann Architecture	Harvard Architecture.
• Codes and data in the same memory	• Codes and data in separate memory
• Require less space	• Require more space
• CPU cannot access data and instructions in same time	• CPU can access data and instructions in same time
• Comparatively slower	• Comparatively faster
• Only a single memory connected to the CPU	• Data memory (RAM) and Program memory (ROM) are separately connected to the CPU
• Using only one channel/bus	• Using separate busses (two busses)

Concepts (Contd.)

Arithmetic Logic Units

- The Arithmetic Logic Unit (ALU) performs the internal arithmetic manipulation of data in the processor
- The instructions that are read and executed by the processor control the data flow between the registers and the ALU
- The instructions also control the arithmetic operations performed by the ALU via the ALU's control inputs
- A symbolic representation of an ALU is as follows
- Whenever instructed by the processor, the ALU performs an operation (typically one of addition, subtraction, NOT, AND, OR, XOR, shift left/right, or rotate left/right) on one or more values

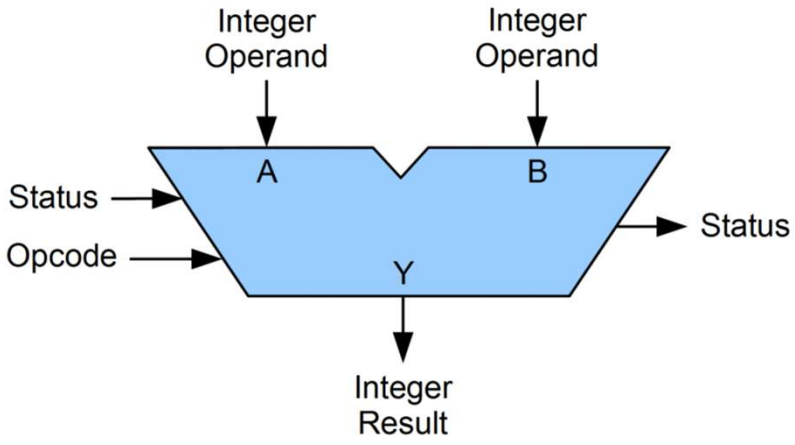


Fig. ALU Diagram

Concepts (Contd.)

Arithmetic Logic Units

- These values, called operands, are typically obtained from two registers, or from one register and a memory location
- The result of the operation is then placed back into a given destination register or memory location
- The status outputs indicate any special attributes about the operation, such as whether the result was zero, negative, or if an overflow or carry occurred
- Some processors have separate units for multiplication and division, and for bit shifting, providing faster operation and increased throughput

Concepts (Contd.)

Arithmetic Logic Units

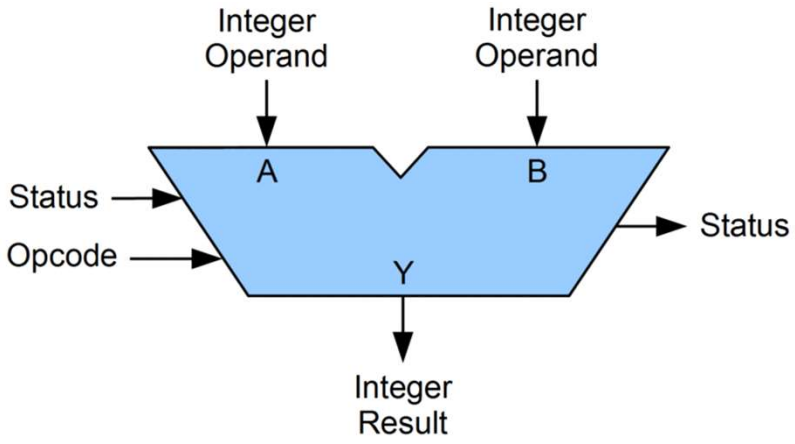


Fig. ALU Diagram

Memory	Mnemonics	Operands
2000	MOV	CX, 0000
2003	MOV	AX, [3000]
2007	MOV	BX, [3002]
200B	ADD	AX, BX
2010	MOV	[3004], AX

Fig. Machine Code to ADD Two Numbers Diagram

Concepts (Contd.)

Interrupts

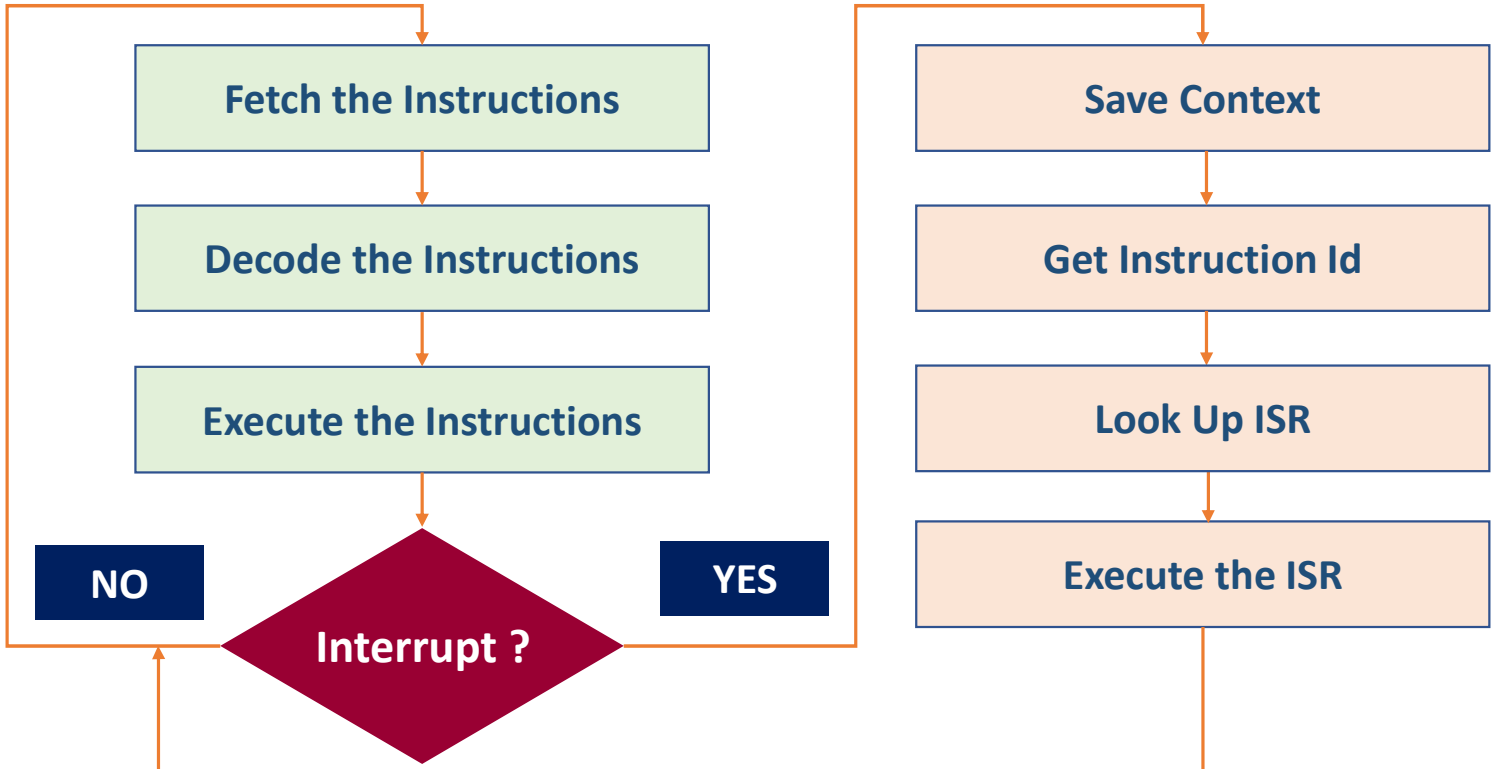
- Interrupts (also known as traps or exceptions in some processors) are a technique of diverting the processor from the execution of the current program so that it may deal with some event that has occurred
- Interrupt is an event may be an error from a peripheral, or simply that an I/O device has finished the last task it was given and is now ready for another
- An interrupt is generated in your computer every time you type a key or move the mouse. You can think of it as a hardware-generated function call
- Interrupts free the processor from having to continuously check the I/O devices to determine whether they require service
- Instead, the processor may continue with other tasks. The I/O devices will notify it when they require attention by asserting one of the processor's interrupt inputs
- Interrupts can be of varying priorities in some processors, thereby assigning differing importance to the events that can interrupt the processor

Concepts (Contd.)

Interrupts

Instructions

1002	ADD
1003	SUB
1004	MUL
1005	DIV
1006	STORE



Interrupt Service Routine (ISR)

Concepts (Contd.)

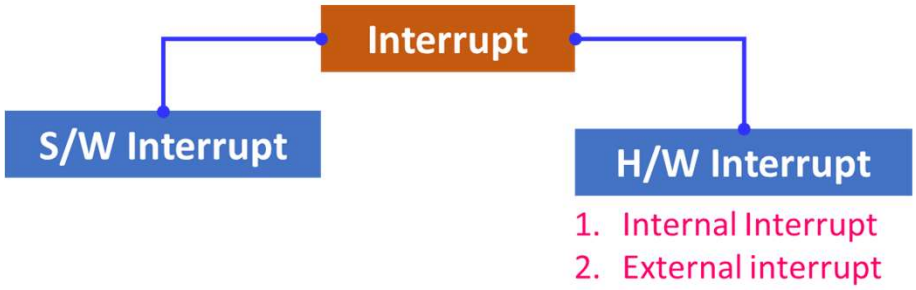
Interrupts

- If the processor is servicing a low-priority interrupt, it will pause it in order to service a higher-priority interrupt. However, if the processor is servicing an interrupt and a second, lower-priority interrupt occurs, the processor will ignore that interrupt until it has finished the higher-priority service.
- When an interrupt occurs, the usual procedure is for the processor to save its state by pushing its registers and program counter onto the stack
- The processor then loads an Interrupt Vector into the program counter. The interrupt vector is the address at which an Interrupt Service Routine (ISR) lies.
- Thus, loading the vector into the program counter causes the processor to begin execution of the ISR, performing whatever service the interrupting device required. The last instruction of an ISR is always a Return from Interrupt instruction.
- This causes the processor to reload its saved state (registers and program counter) from the stack and resume its original program. Interrupts are largely transparent to the original program.

Concepts (Contd.)

Interrupts

- This means that the original program is completely "unaware" that the processor was interrupted, save for a lost interval of time
- Processors with shadow registers use these to save their current state, rather than pushing their register back onto the stack. This saves considerable memory accesses (and therefore time) when processing an interrupt
- However, since only one set of shadow registers exists, a processor servicing multiple interrupts must "manually" preserve the state of the registers before servicing the higher interrupt. If it does not, important state information will be lost
- Upon returning from an ISR, the contents of the shadow registers are swapped back into the main register array
- Internal Interrupts: Generated by I/O Devices
- External Interrupts: Generated as result of outside interference



Concepts (Contd.)

Interrupts: Hardware Interrupt

- A hardware interrupt is an electronic alerting signal sent to the processor from an external device, like a disk controller or an external peripheral
- For example, when we press a key on the keyboard or move the mouse, they trigger hardware interrupts which cause the processor to read the keystroke or mouse position
- Small, simple processors may only have one (or two) interrupt inputs, so several external devices may have to share the interrupt lines of the processor
- When an interrupt occurs, the processor must check each device to determine which one generated the interrupt (This can also be considered a form of Polling.)
- The advantage of interrupt polling over ordinary polling is that the polling occurs only when there is a need to service a device
- The Polling interrupts is suitable only in systems that have a small number of devices; otherwise, the processor will spend too long trying to determine the source of the interrupt.

Concepts (Contd.)

Interrupts: Hardware Interrupt

- The other technique of servicing an interrupt is by using Vectored Interrupts, by which the interrupting device provides the interrupt vector that the processor is to take
- The Vectored Interrupts reduce considerably the time it takes the processor to determine the source of the interrupt
- If an interrupt request can be generated from more than one source, it is therefore necessary to assign priorities (levels) to the different interrupts
- This can be done in either hardware or software, depending on the particular application
- So, for example, when an interrupt of priority 7 occurs (interrupt lines corresponding to "7" are declared), the processor loads vector 7 into its program counter and starts executing the service routine specific to interrupt 7.

Concepts (Contd.)

Interrupts: Hardware Interrupt

- Some processors have a feature known as a fast hardware interrupt
- With this interrupt, only the program counter is saved
- It assumes that the ISR will protect the contents of the registers by manually saving their state as required
- Fast interrupts are useful when an I/O device requires a very fast response from a processor and cannot wait for the processor to save all its registers to the stack
- A special (and separate) interrupt line is used to generate fast interrupts

Concepts (Contd.)

Interrupts: Software Interrupt

- A software interrupt is caused either by an exceptional condition or a special instruction in the instruction set which causes an interrupt when it is executed by the processor
- For example, if the processor's arithmetic logic unit runs a command to divide a number by zero, to cause a divide-by-zero exception, thus causing the computer to abandon the calculation or display an error message. Software interrupt instructions work similar to subroutine calls
- It is the lowest-priority interrupt and is generally used by programs to request a service to be performed by the system software

Concepts (Contd.)

CISC and RISC

- There are two major approaches to **Processor Architecture**: Complex Instruction Set Computer (CISC, pronounced "Sisk") processors and Reduced Instruction Set Computer (RISC) processors
- CISC was developed by **Intel**, and the RISC was developed by **IBM**
- Classic CISC processors are the Intel x86, Motorola 68xxx, and National Semiconductor 32xxx processors, and, to a lesser degree, the Intel Pentium
- Common RISC architectures are the Freescale/IBM PowerPC, the MIPS architecture, Sun's SPARC, the ARM, the Atmel AVR, and the Microchip PIC

Concepts (Contd.)

CISC and RISC

- CISC processors have a single processing unit, external memory, and a relatively small register set and many hundreds of different instructions
- RISC processors have a number of distinguishing characteristics. They have large register sets (in some architectures numbering over 1,000), thereby reducing the number of times the processor must access main memory
- In RISC variables can be left inside the processor, reducing the number of accesses to (slow) external memory
- By having smaller and simpler instruction decode units, RISC processors have fast instruction execution, and this also reduces the size and power consumption of the processing unit.
- This is in contrast to instructions for a CISC processor, whose instructions may take many tens of cycles to execute. For example, one instruction (integer multiplication) on an 80486 CISC processor takes 42 cycles to complete. The same instruction on a RISC processor may take just one cycle

Concepts (Contd.)

CISC and RISC

- RISC processors implement what is known as a "load/store" architecture. This means that the only instructions that actually reference memory are load and store. In contrast, many (most) instructions on a CISC processor may access or manipulate memory
- On a RISC processor, all other instructions (aside from load and store) work on the registers only
- RISC processors also often have pipelined instruction execution. This means that while one instruction is being executed, the next instruction in the sequence is being decoded, while the third one is being fetched
- Due to their low power consumption and computing power, RISC processors are becoming widely used, particularly in embedded computer systems, and many RISC attributes are appearing in what are traditionally CISC architectures (such as with the Intel Pentium)

Concepts (Contd.)

CISC and RISC

- Instructions on a RISC processor have a simple format. All instructions are generally the same length

So, which is better for embedded and industrial applications, RISC or CISC?

- If power consumption needs to be low, then RISC is probably the better architecture to use
- However, if the available space for program storage is small, then a CISC processor may be a better alternative

Ironically, many RISC architectures are adding some CISC-like features, and so the distinction between RISC and CISC is blurring

Concepts (Contd.)

Difference Between RISC and SISC	
RISC	**CISC**
Some of the instructions refer to memory.	Most of the instructions refer to Memory
There are few addressing modes. Most instructions have register to register addressing modes.	There are many addressing modes
There are few instructions.	There are many instructions
It can include simple instructions and takes one cycle.	It can include complex instructions and takes multiple cycles.
Hardware executes the instructions.	Micro-program executes the instructions
There are Fixed format instructions.	There are Variable format instructions
It can be easier to decode as instructions have a fixed format.	It can be complex to decode as instructions have variable format
There are multiple register sets are used.	A single register set is used
It can load and store functions are separate instructions.	It can load and store functions are found in a single instruction

Concepts (Contd.)

Digital Signal Processors (DSP)s

- A special type of processor architecture is that of the Digital Signal Processor (DSP). These processors have instruction sets and architectures optimized for numerical processing of array data
- They often extend the Harvard architecture concept further by not only having separate data and code spaces, but also by splitting the data spaces into two or more banks
- This allows concurrent instruction fetch and data accesses for multiple operands. As such, DSPs can have very high throughput and can outperform both CISC and RISC processors
- DSPs have special hardware well suited to numerical processing of arrays
- They often have hardware looping, whereby special registers allow for and control the repeated execution of an instruction sequence. This is also often known as zero-overhead looping, since no conditions need to be explicitly tested by the software as part of the looping process
- DSPs often have dedicated hardware for increasing the speed of arithmetic operations. High-speed multipliers, Multiply-And-Accumulate (MAC) units, and barrel shifters are common features

Memory

- Memory is used to hold data
- There is a variety of memory types, and often a mix is used within a single system. Some memory will retain its contents while there is no power (Non-Volatile), yet will be slow to access
- Other memory devices will be high-capacity, yet will require additional support circuitry and will be slower to access
- Still other memory devices will trade capacity for speed, yielding relatively small devices, yet will be capable of keeping up with the fastest of processors
- Memory chips can be organized in two ways, either in word-organized or bit-organized schemes
- In the word-organized scheme, complete nibbles, bytes, or words are stored within a single component
- In bit-organized memory, each bit of a byte or word is allocated to a separate component

Memory (Contd.)

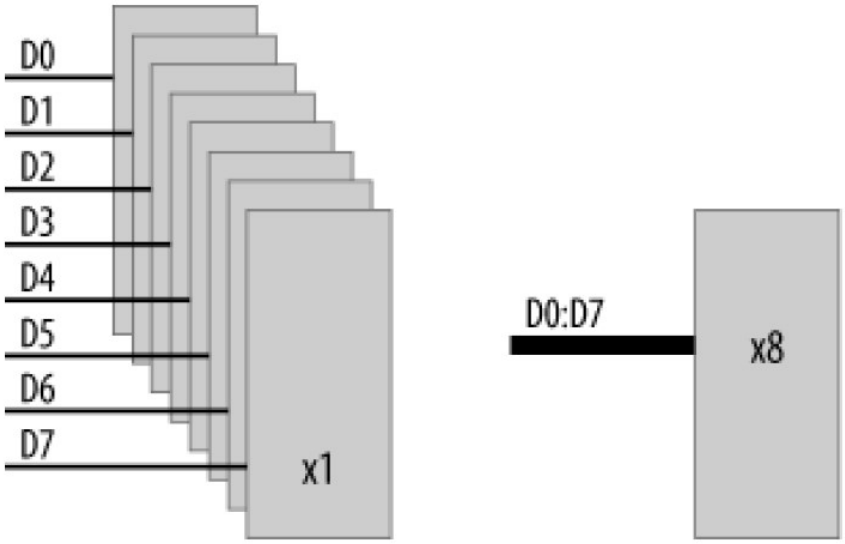
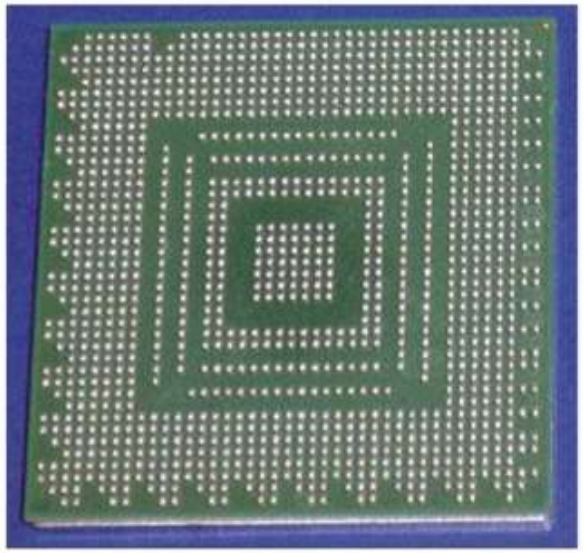
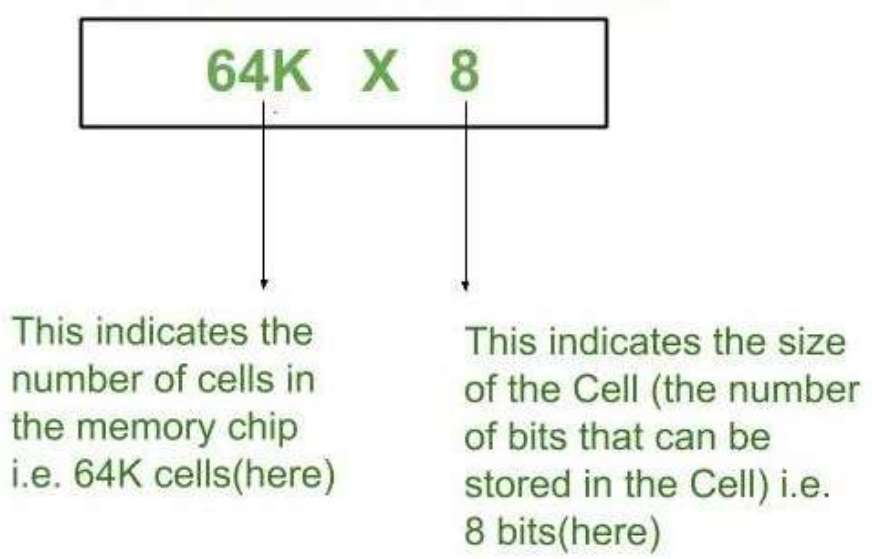


Fig. Eight bit-organized 8x1 devices and one word-organized 8x8 device

- Memory chips come in different sizes, with the width specified as part of the size description
- For instance, a DRAM (Dynamic RAM) chip might be described as being 4Mx1 (bit-organized), whereas a SRAM (Static RAM) may be 512Kx8 (word-organized)
- In both cases, each chip has exactly the same storage capacity, but organized in different ways

Memory (Contd.)





- In the DRAM case, it would take eight chips to complete a memory block for an 8-bit data bus, whereas the SRAM would only require one chip. However, because the DRAMs are organized in parallel, they are accessed simultaneously

Memory (Contd.)

RAM

- RAM stands for Random Access Memory
- RAM is the "working memory" in the computer system
- RAM is the place where the processor may easily write data for temporary storage
- RAM is generally volatile, losing its contents when the system loses power
- Any information stored in RAM that must be retained must be written to some form of permanent storage before the system powers down
- There are special nonvolatile RAMs that integrate a battery-backup system, such that the RAM remains powered even when the rest of the computer system has shut down
- RAMs generally fall into two categories: Static RAM (also known as SRAM) and Dynamic RAM (also known as DRAM)

Memory (Contd.)

RAM: SRAM

- SRAMs use pairs of logic gates to hold each bit of data
- SRAMs are the fastest form of RAM available, require little external support circuitry, and have relatively low power consumption
- Their drawbacks are that their capacity is considerably less than DRAM, while being much more expensive
- Their relatively low capacity requires more chips to be used to implement the same amount of memory

Memory (Contd.)

RAM: DRAM

- DRAM uses arrays of what are essentially capacitors to hold individual bits of data
- The capacitor arrays will hold their charge only for a short period before it begins to diminish. Therefore, DRAMs need continuous refreshing, every few milliseconds or so
- This perpetual need for refreshing requires additional support and can delay processor access to the memory
- If a processor access conflicts with the need to refresh the array, the refresh cycle must take precedence
- Interfacing DRAMs to small microcontrollers is generally not possible, and certainly not practical. Most processors with large address spaces include support for DRAMs

Memory (Contd.)

Difference Between SRAM and DRAM

Static RAM	Dynamic RAM
Uses Transistors to Store Information	Uses capacitors to Store the Information
Architecture is Complex	Architecture is Simple
Faster than DRAM	Slower than SRAM
Used in Cache Memory	Used in Main Memory

Memory (Contd.)

ROM

- ROM stands for Read-Only Memory
- ROMs are non-volatile memory, requiring no power to retain their contents
- They are generally slower than RAM, and considerably slower than fast static RAM
- The primary purpose of ROM within a system is to hold the code (and sometimes data) that needs to be present at power-up
- Such software is generally known as firmware and contains software to initialize the computer by placing I/O devices into a known state
- It may contain either a bootloader program to load an operating system off disk or network or, in the case of an embedded system, it may contain the application itself
- Many microcontrollers contain on-chip ROM, thereby reducing component count and simplifying system design

Memory (Contd.)

ROM

- Standard ROM is fabricated (in a simplistic sense) from a large array of diodes
- The unwritten bit state for a ROM is all 1s
- The process of loading software into a ROM is known as burning the ROM
- This term comes from the fact that the programming process is performed by passing a sufficiently large current through the appropriate diodes to "blow them," or burn them, thereby creating a zero at that bit location
- A device known as a ROM burner can accomplish this, or, if the system supports it, the ROM may be programmed in-circuit. This is known as In-System Programming (ISP) or, sometimes, In-Circuit Programming (ICP)
- One-Time Programmable (OTP) ROMs, as the name implies, can be burned once only Computer manufacturers typically use them in systems where the firmware is stable and the product is shipping in bulk to customers

Memory (Contd.)

ROM

- Mask-programmable ROMs are also one-time programmable, but unlike OTPs, they are burned by the chip manufacturer prior to shipping
- Like OTPs, they are used once the software is known to be stable and have the advantage of lowering production costs for large shipments.

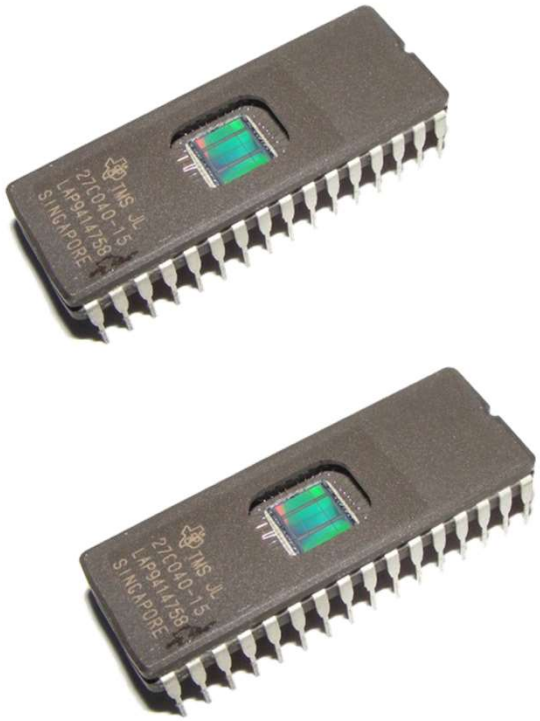
Memory (Contd.)

ROM

- OTP ROMs are great for shipping in final products, but they are wasteful for debugging, since with each iteration of code change, a new chip must be burned and the old one thrown away
- As such, OTPs make for a very expensive development option
- A (slightly) better choice for system development and debugging is the Erasable Programmable Read-Only Memory, or EPROM. Shining ultraviolet light through a small window on the top of the chip can erase the EPROM, allowing it to be reprogrammed and reused
- They are pin-and signal-compatible with comparable OTP and mask devices. Thus, an EPROM can be used during development, while OTPs can be used in production with no change to the rest of the system.

Memory (Contd.)

ROM: EPROM



- EPROMs (**Erasable Programmable read-only Memory**) and their equivalent OTP cousins range in capacity from a few kilobytes (exceedingly rare these days) to a megabyte or more
- The drawback with EPROM technology is that the chip must be removed from the circuit to be erased, and the erasure can take many minutes to complete
- The chip is then inserted into the burner, loaded with software, and then placed back in-circuit. This can lead to very slow debug cycles
- Further, it makes the device useless for storing changeable system parameters
- EPROMs are relatively rare these days

Memory (Contd.)

ROM: EEROM

- EEROM is Electrically Erasable Read-Only Memory, also known as EEPROM
- Very rarely, it is also called Electrically Alterable
- EEROM can be pronounced as either "e-e ROM" or "e-squared. ROM, or sometimes just "e-squared" for short
- EEROMs can be erased and reprogrammed in-circuit. Their capacity is significantly smaller than standard ROM (typically only a few kilobytes), and so they are not suited to holding firmware
- Instead, they are typically used for holding mode information to be retained during power-off
- It is common for many microcontrollers to incorporate a small EEROM on-chip for holding system parameters
- This is especially useful in embedded systems and may be used for storing network addresses, configuration settings, serial numbers, servicing records, and so on.

Memory (Contd.)

ROM: Flash

- Flash is the newest ROM technology and is now dominant
- Flash memory has the re-programmability of EEROM and the large capacity of standard ROMs
- The Flash chips are sometimes referred to as "flash ROMs" or "flash RAMs."
- Flash is normally organized as sectors and has the advantage that individual sectors may be erased and rewritten without affecting the contents of the rest of the device
- Typically, before a sector can be written, it must be erased. It can't just be written over as with a RAM
- There are several different flash technologies, and the erasing and programming requirements of flash devices vary from manufacturer to manufacturer.

Input/ Output

- The address space of the processor can contain devices other than memory
- These are input/output devices (I/O devices, also known as peripherals) are used by the processor to communicate with the external world
- Some examples are serial controllers that communicate with keyboards, mice, modems, etc.;
- Parallel I/O devices that control some external subsystem; or disk-drive controllers, video and audio controllers, or network interfaces

Input/ Output (Contd.)

Programmed I/O

The processor accepts or delivers data at times convenient to it (the processor). It is one of the simplest forms of I/O where the Processor has to do all the work

Interrupt-Driven I/O

External events control the processor by requesting the current program be suspended and the external event be serviced. An external device will interrupt the processor (assert an interrupt control line into the processor), at which time the processor will suspend the current task (program) and begin executing an interrupt service routine. The service of an interrupt may involve transferring data from input to memory, or from memory to output.

Direct Memory Access (DMA)

DMA allows data to be transferred from I/O devices to memory directly without the continuous involvement of the processor. DMA is used in high-speed systems, where the rate of data transfer is important. Not all processors support DMA.

Direct Memory Access

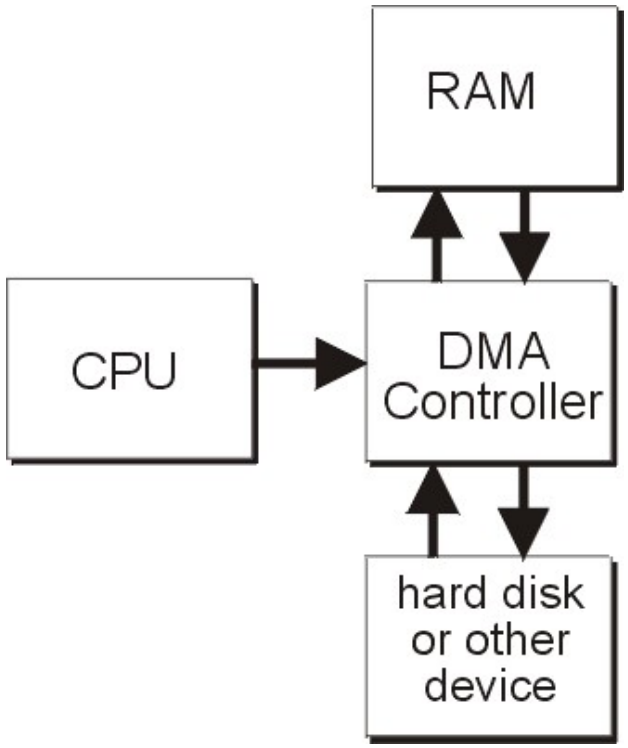
- DMA is a way of streamlining transfers of large blocks of data between two sections of memory, or between memory and an I/O device
- Let's say you want to read in 100M from disk and store it in memory. There are two options
- Option 1

  The processor read one byte at a time from the disk controller into a register and then store the contents of the register to the appropriate memory location. For each byte transferred, the processor must read an instruction, decode the instruction, read the data, read the next instruction, decode the instruction, and then store the data. Then the process starts over again for the next byte.

- Option 2

  Move the large amounts of data to the DMA Systems. Using DMA bypasses the processor. A special device, called a DMA Controller (DMAC), performs high-speed transfers between memory and I/O devices. by setting up a channel between the I/O device and the memory. Thus, data is read from the I/O device and written into memory without the need to execute code to perform the transfer on a byte-by-byte

Direct Memory Access (Contd.)



Types of DMA

- **Standard Block Transfer**

  The transfers involve a load operation from a source address followed by a store operation to a destination address. Standard block transfers are initiated under software control and are used for moving data structures from one region of memory to another

- **Demand-mode transfers**

  Similar to standard mode except that the transfer is controlled by an external device. Demand-mode transfers are used to move data between memory and I/O or vice versa. The I/O device requests and synchronizes the movement of data

Direct Memory Access (Contd.)

- **Fly-by transfer**

  Provides high-speed data movement in the system. Instead of using multiple bus accesses as with conventional DMA transfers, fly-by transfers move data from source to destination in a single access. During a fly-by transfer, memory and I/O are given different bus control signals. For example, an I/O device is given a read request at the same time that memory is given a write request. Data moves from the I/O device straight into the memory device.

- **Data-chaining transfers**

  Allow DMA transfers to be performed as specified by a linked-list in memory. Data chaining is started by specifying a pointer to a descriptor in memory. The descriptor is a table specifying byte count, source address, destination address, and a pointer to the next descriptor. The DMAC loads the relevant information about the transfer from this table and begins moving data. The transfer continues until the number of bytes transferred is equal to the entry in the byte-count field. On completion, the pointer to the next descriptor is loaded. This continues until a null pointer is found.

Direct Memory Access (Contd.)

Parallel and Distributed Computers

- Some embedded applications require greater performance than is achievable from a single processor
- It may make more sense to use a fleet of lower-cost processors, distributed throughout the installation
- It is becoming increasingly common to see embedded systems implemented using parallel processors

Direct Memory Access (Contd.)

Introduction to Parallel Architectures

- The traditional architecture for computers follows the conventional, Von Neumann serial architecture
- Computers based on this form usually have a single, sequential processor. The main limitation of this form of computing architecture is that the conventional processor is able to execute only one instruction at a time
- The Algorithms that run on these machines must therefore be expressed as a sequential problem
- A given task must be broken down into a series of sequential steps, each to be executed in order, one at a time.

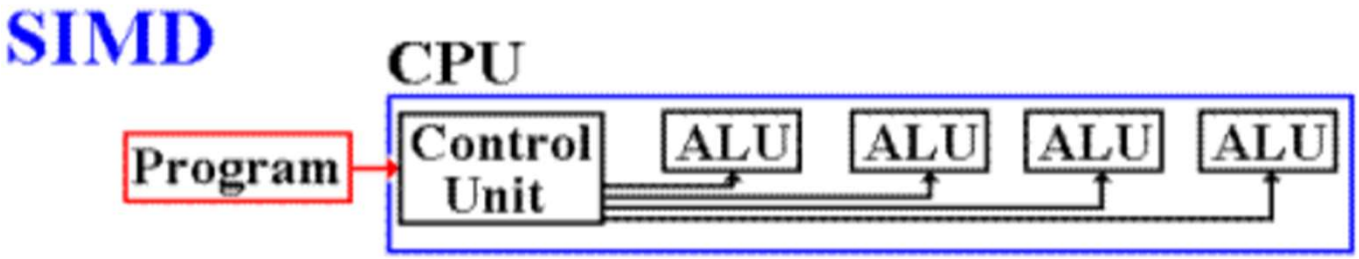
Direct Memory Access (Contd.)

SIMD Computers

- A Single-Instruction Multiple-Data (SIMD) computers are highly parallel machines, employing large arrays of simple processing elements
- In an SIMD machine, each processing element has a small amount of local memory
- The instructions executed by the SIMD computer are broadcast from a central instruction server to every processing element within the machine
- In this way, each processor executes the same instruction as all other processing elements within the machine
- Since each processor executes the instruction on its local data, all elements within the data structure are worked upon simultaneously
- The SIMD machine is generally used in conjunction with a conventional computer

Direct Memory Access (Contd.)

SIMD Computers



Direct Memory Access (Contd.)

SIMD Computers

- An example of this was the Connection Machine (CM-1) by Thinking Machines Corporation that used either a VAX minicomputer or a Silicon Graphics or Sun workstation as the "host" computer
- The CM-1 was a finely grained SIMD computer with up to 64K of processing elements that appeared as a block of 64K of "intelligent memory" to the host system
- An application running on the host downloaded a data set into the processor array of the CM-1, each processor within the CM-1 acting as a single memory unit
- The host then issued instructions to each processing element of the CM-1 simultaneously. After the computations were completed, the host then read back the result from the CM-1 as though it were conventional memory

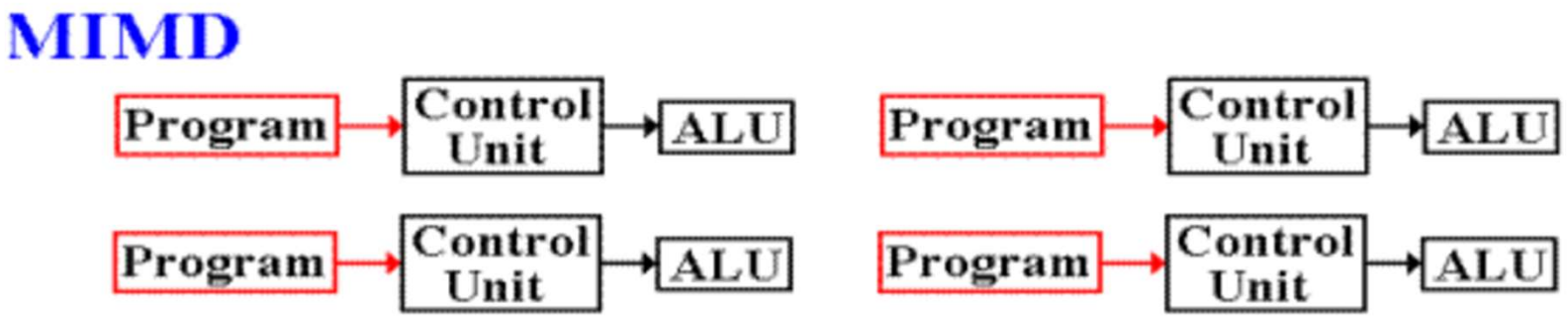
Direct Memory Access (Contd.)

MIMD Computers

- The other major form of parallel machine is the Multiple-Instruction Multiple-Data (MIMD) computer
- These machines are typically coarsely grained collections of semi-autonomous processors, each with their own local memory and local programs
- An algorithm being executed on an MIMD computer is typically broken up into a series of smaller sub-problems, each executed on a processor of the MIMD machine
- By giving each processing element in the MIMD machine identical programs to execute, the MIMD machine may be treated as an SIMD computer
- MIMD computers tend to use a smaller number of very powerful processors, rather than a large number of less powerful ones
- MIMD computers can be of one of two types: Shared-memory MIMD and Message-passing MIMD

Direct Memory Access (Contd.)

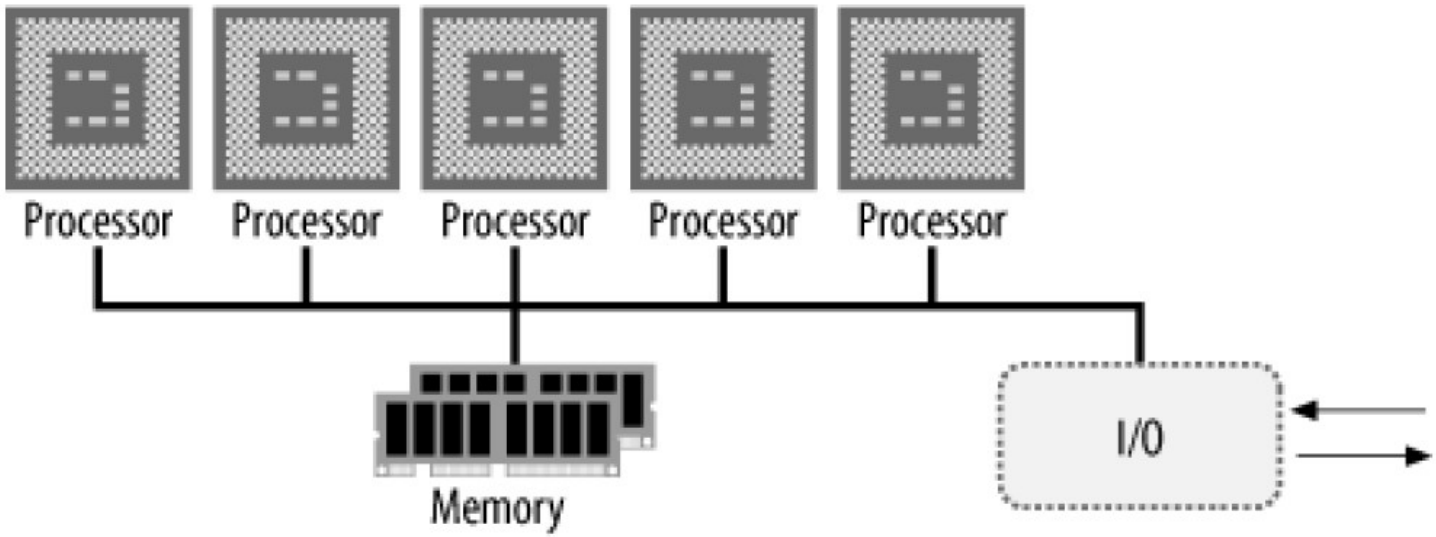
MIMD Computers



Direct Memory Access (Contd.)

MIMD Computers

- Shared-memory MIMD systems have an array of high-speed processors, each with local memory or cache, and each with access to a large, global memory



Direct Memory Access (Contd.)

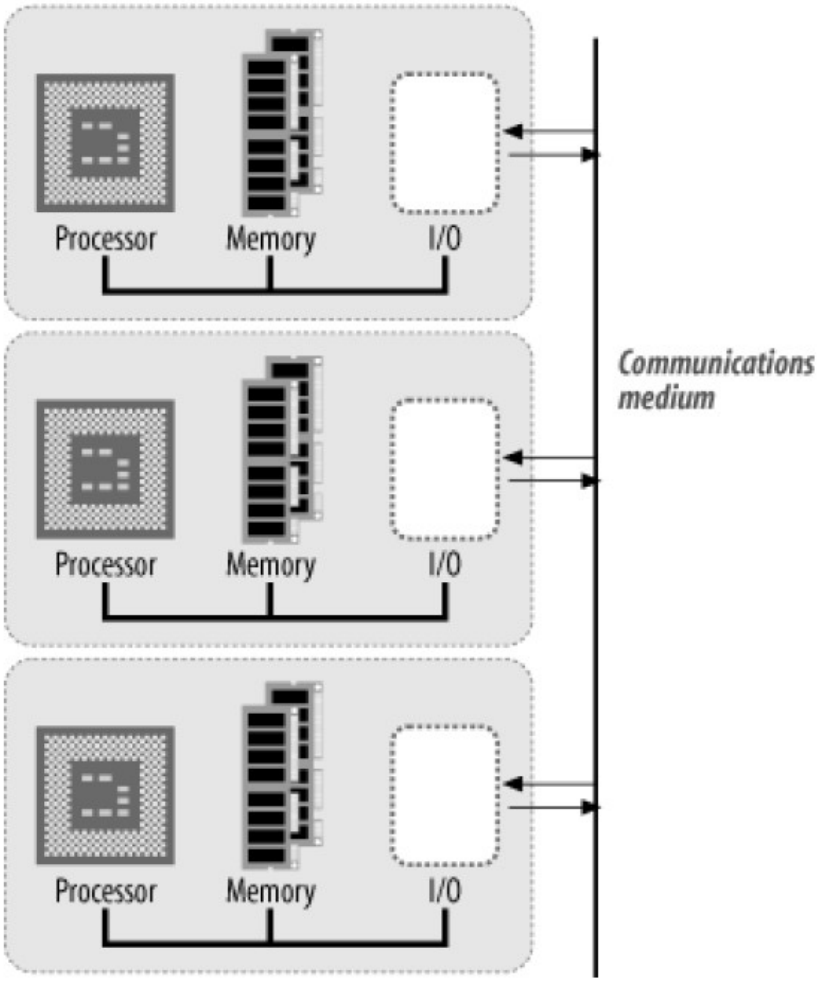
MIMD Computers

- The global memory contains the data and programs to be executed by the machine. Also in this memory is a table of processes (or sub-programs) awaiting execution
- Each processor will fetch a process and associated data into its local memory or cache and will run semi-autonomously of the other processors in the system
- The process communication also takes place through the global memory
- A speed advantage is gained by sharing the program among several, powerful processors
- However, logic within the system must arbitrate between processors for access to the shared memory and associated shared buses of the system
- In addition, allowances must be made for a processor attempting to access data in global memory that is out of date.

Direct Memory Access (Contd.)

MIMD Computers

- If processor A reads a process and data structure into its local memory and subsequently modifies that data structure, processor B attempting to access the same data structure in main memory must be notified that a more recent version of the data structure exists. Such arbitration is implemented in processors like the (now extinct) Motorola MC88110, which was intended for use in shared-memory MIMD machines.



Direct Memory Access (Contd.)

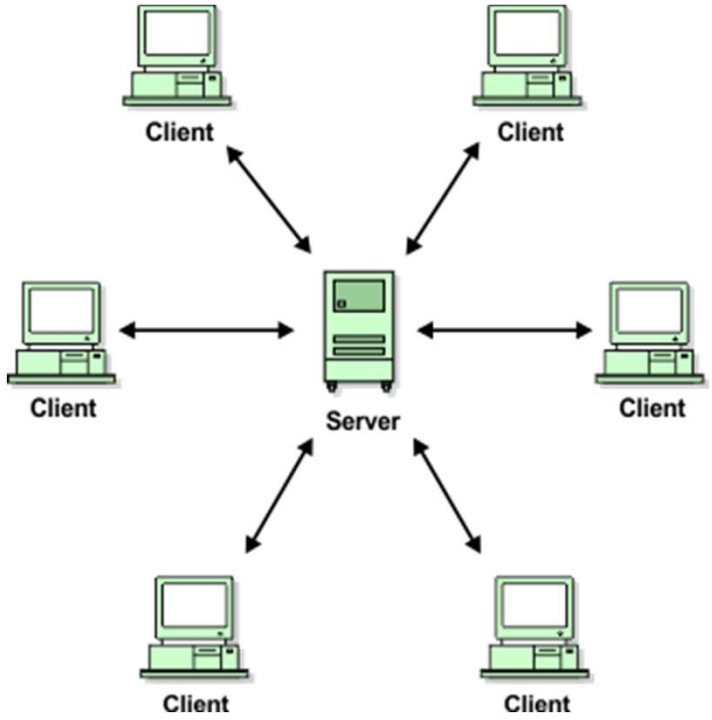
MIMD Computers

- The processors may communicate through a single, shared bus (such as Ethernet, CAN, or SCSI) or by using a more elaborate inter-processor connection architecture, such as 2-D arrays, N-dimensional hypercubes, rings, stars, trees, or fully interconnected systems
- MIMD machines do not suffer the bus-contention problems of shared-memory machines. However, the most effective and efficient means of interconnecting the processing nodes of a message-passing MIMD machine is still a major area of research
- The ideal interconnection architecture is that of the fully interconnected system, where every processing node has a direct communications link with every other processing node. However, this is not always practical, due to the costs and logistics of such a high degree of interconnectivity.
- A solution to this problem is to provide each processing element in the machine with a limited number of connections, based on the assumption that a processing element will not need or be able to communicate with every other processing element in the machine simultaneously

Direct Memory Access (Contd.)

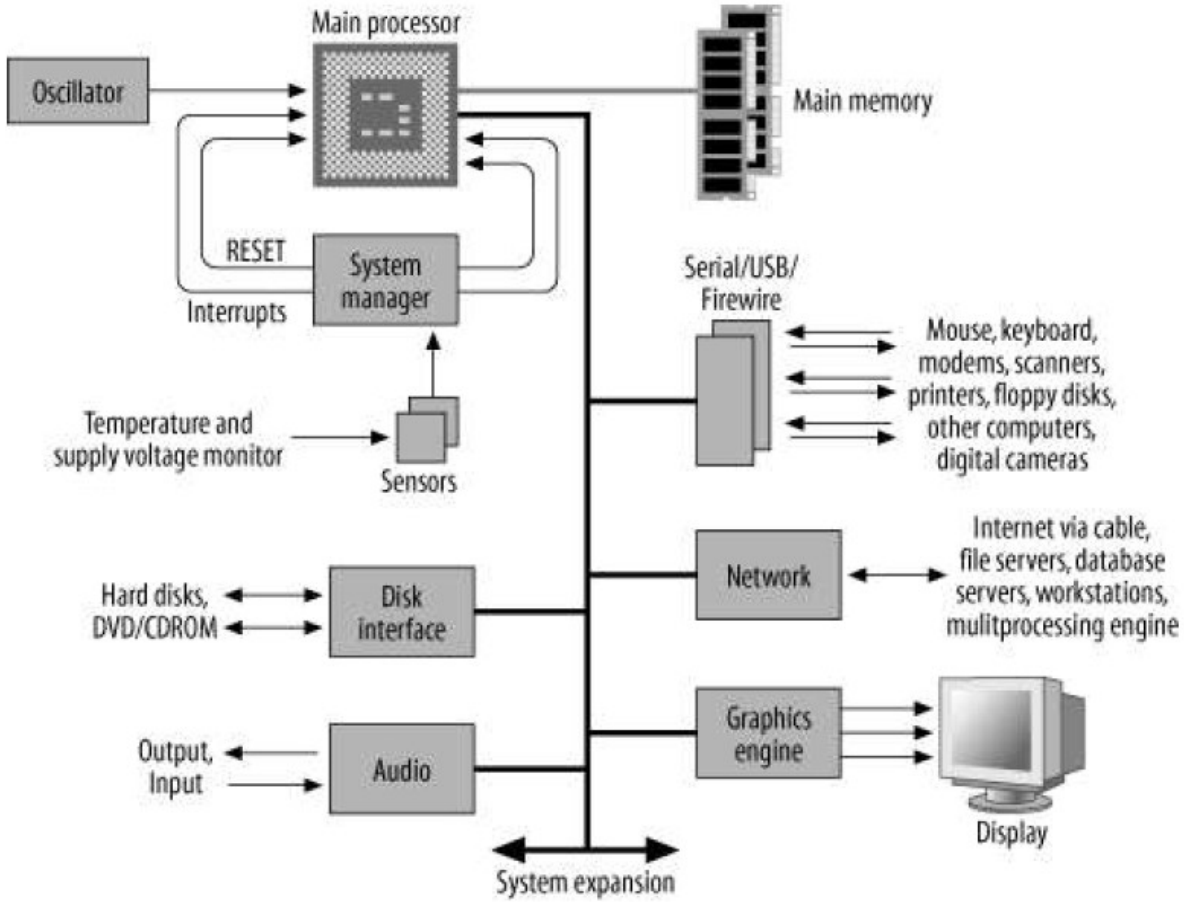
Distributed Computers

- A distributed machine is composed of individual computers networked together as a loosely coupled MIMD parallel machine
- Distributed machines are common in the embedded world. A collection of small processing nodes may be distributed across a factory, providing local monitoring and control, and together forming a parallel machine executing the global control algorithm
- The avionics of commercial and military aircraft are also distributed parallel computers



Embedded Computer Architecture

- Desktop computer is shown in



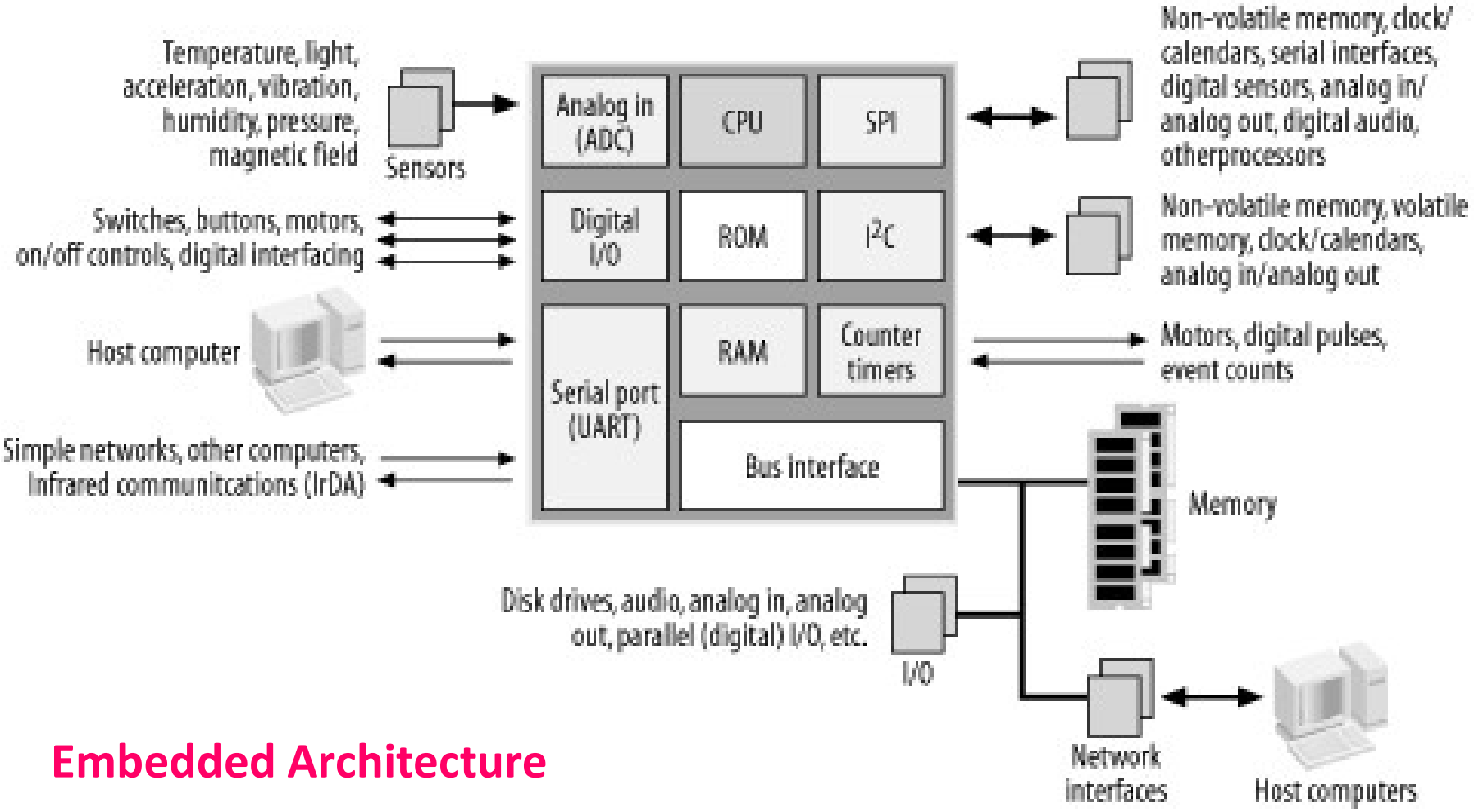
Embedded Computer Architecture (Contd.)

- It has a large main memory to hold the operating system, applications, and data, and an interface to mass storage devices (disks and DVD/CD-ROMs)
- It has a variety of I/O devices for user input (keyboard, mouse, and audio), user output (display interface and audio), and connectivity (networking and peripherals)
- The fast processor requires a system manager to monitor its core temperature and supply voltages, and to generate a system reset.

Embedded Computer Architecture (Contd.)

- The Large-scale embedded computers may also take the same form as Computers
- For example, they may act as a network router or gateway, and so will require one or more network interfaces, large memory, and fast operation
- In terms of hardware, many high-performance embedded systems are not that much different from a conventional desktop machine
- Smaller embedded systems use microcontrollers as their processor, with the advantage that this processor will incorporate much of the computer's functionality on a single chip
- The microcontroller has, at a minimum, a CPU, a small amount of internal memory (ROM and/or RAM), and some form of I/O, which is implemented within a microcontroller as subsystem blocks.

Embedded Computer Architecture (Contd.)



Embedded Architecture

Embedded Computer Architecture (Contd.)

- The most common I/O is digital I/O, commonly called general-purpose I/O, or GPIO
- These are ports that may be configured by software, on a pin-by-pin basis, as either a digital input or digital output
- As digital inputs, they may be used to read the state of switches or push buttons, or to read the digital status of another device
- For example, a digital output may be used to activate the control circuitry for a motor, turn a light on or off, or perhaps activate some other device such as a water valve for a garden-watering system
- Many microcontrollers also have analog inputs, allowing sensors to be sampled for monitoring or recording purposes. Thus, an embedded computer may measure light levels, temperature, vibration or acceleration, air or water pressure, humidity, or magnetic field, to name just some
- Some microcontrollers have serial ports, which enable the embedded computer to be interfaced to a host computer, a modem, another embedded system, or perhaps a simple network

Embedded Computer Architecture (Contd.)

- Specialized forms of serial interface, such as SPI and I2C, provide a simple way of expanding the microcontroller's functionality
- They allow peripherals to be interfaced to the microcontroller, providing access to such devices as off-chip memories (for data or parameter storage), clock/calendar chips (for timekeeping)
- Sensors with digital interfaces, external analog input or output, and even audio chips and other processors. Most microcontrollers have timers and counters
- These may be used to generate internal interrupts at regular intervals for multitasking, to generate external triggers for off-chip systems, or to provide control pulses for motors
- Alternatively, they may be used to count external triggers (pulses) from another system. A few microcontrollers also include network interfaces, such as USB, Ethernet, or CAN


Thank
you