OUTLINES

- Microcontrollers and embedded processors
- Overview of the 8051 family

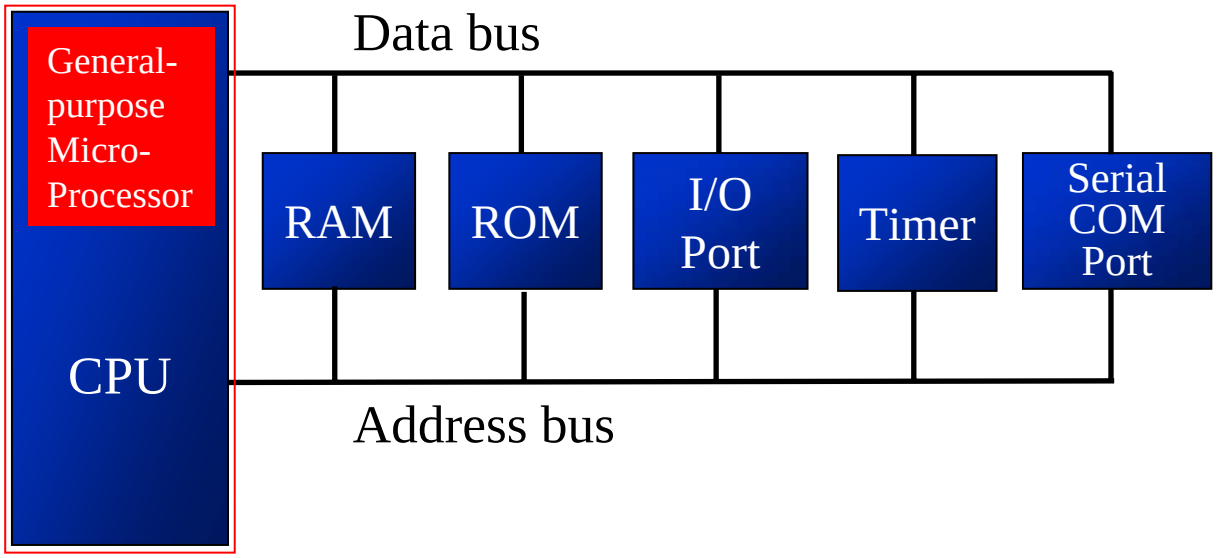
MICRO-CONTROLLERS AND EMBEDDED PROCESSORS

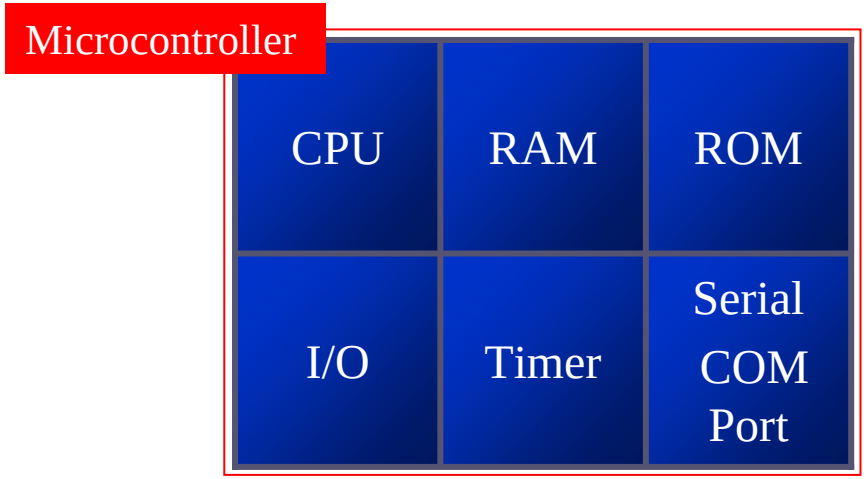
Microcontroller vs. General-Purpose Microprocessor

- General-purpose microprocessors contains
  - No RAM
  - No ROM
  - No I/O ports
- Microcontroller has
  - CPU (microprocessor)
  - RAM
  - ROM
  - I/O ports
  - Timer
  - ADC and other peripherals

MICRO-CONTROLLERS AND EMBEDDED PROCESSORS

Microcontroller vs. General-Purpose Microprocessor (cont')





MICRO-CONTROLLERS AND EMBEDDED PROCESSORS

Microcontroller vs. General-Purpose Microprocessor (cont')

- General-purpose microprocessors
  - Must add RAM, ROM, I/O ports, and timers externally to make them functional
  - Make the system bulkier and much more expensive
  - Have the advantage of versatility on the amount of RAM, ROM, and I/O ports
- Microcontroller
  - The fixed amount of on-chip ROM, RAM, and number of I/O ports makes them ideal for many applications in which cost and space are critical
  - In many applications, the space it takes, the power it consumes, and the price per unit are much more critical considerations than the computing power

MICRO-CONTROLLERS AND EMBEDDED PROCESSORS

Microcontrollers for Embedded Systems

- An embedded product uses a microprocessor (or microcontroller) to do one task and one task only
  - There is only one application software that is typically burned into ROM
- A PC, in contrast with the embedded system, can be used for any number of applications
  - It has RAM memory and an operating system that loads a variety of applications into RAM and lets the CPU run them
  - A PC contains or is connected to various embedded products
    - Each one peripheral has a microcontroller inside it that performs only one task

MICRO-CONTROLLERS AND EMBEDDED PROCESSORS

Microcontrollers for Embedded Systems (cont')

- Home
  - Appliances, intercom, telephones, security systems, garage door openers, answering machines, fax machines, home computers, TVs, cable TV tuner, VCR, camcorder, remote controls, video games, cellular phones, musical instruments, sewing machines, lighting control, paging, camera, pinball machines, toys, exercise equipment
- Office
  - Telephones, computers, security systems, fax machines, microwave, copier, laser printer, color printer, paging
- Auto
  - Trip computer, engine control, air bag, ABS, instrumentation, security system, transmission control, entertainment, climate control, cellular phone, keyless entry

MICRO-CONTROLLERS AND EMBEDDED PROCESSORS

x86 PC Embedded Applications

- Many manufactures of general-purpose microprocessors have targeted their microprocessor for the high end of the embedded market
  - There are times that a microcontroller is inadequate for the task
- When a company targets a general-purpose microprocessor for the embedded market, it optimizes the processor used for embedded systems
- Very often the terms *embedded processor* and *microcontroller* are used interchangeably

MICRO-CONTROLLERS AND EMBEDDED PROCESSORS

x86 PC Embedded Applications (cont')

- One of the most critical needs of an embedded system is to decrease power consumption and space
- In high-performance embedded processors, the trend is to integrate more functions on the CPU chip and let designer decide which features he/she wants to use
- In many cases using x86 PCs for the high-end embedded applications
  - Saves money and shortens development time
    - A vast library of software already written
    - Windows is a widely used and well understood platform

MICRO-CONTROLLERS AND EMBEDDED PROCESSORS

Choosing a Microcontroller

- 8-bit microcontrollers
  - Motorola's 6811
  - Intel's 8051
  - Zilog's Z8
  - Microchip's PIC
- There are also 16-bit and 32-bit microcontrollers made by various chip makers

MICRO-CONTROLLERS AND EMBEDDED PROCESSORS

Criteria for Choosing a Microcontroller

- Meeting the computing needs of the task at hand efficiently and cost effectively
  - Speed
  - Packaging
  - Power consumption
  - The amount of RAM and ROM on chip
  - The number of I/O pins and the timer on chip
  - How easy to upgrade to higher-performance or lower power-consumption versions
  - Cost per unit

MICRO-CONTROLLERS AND EMBEDDED PROCESSORS

Criteria for Choosing a Microcontroller (cont')

- Availability of software development tools, such as compilers, assemblers, and debuggers
- Wide availability and reliable sources of the microcontroller
  - The 8051 family has the largest number of diversified (multiple source) suppliers
    - Intel (original)
    - Atmel
    - Philips/Signetics
    - AMD
    - Infineon (formerly Siemens)
    - Matra
    - Dallas Semiconductor/Maxim

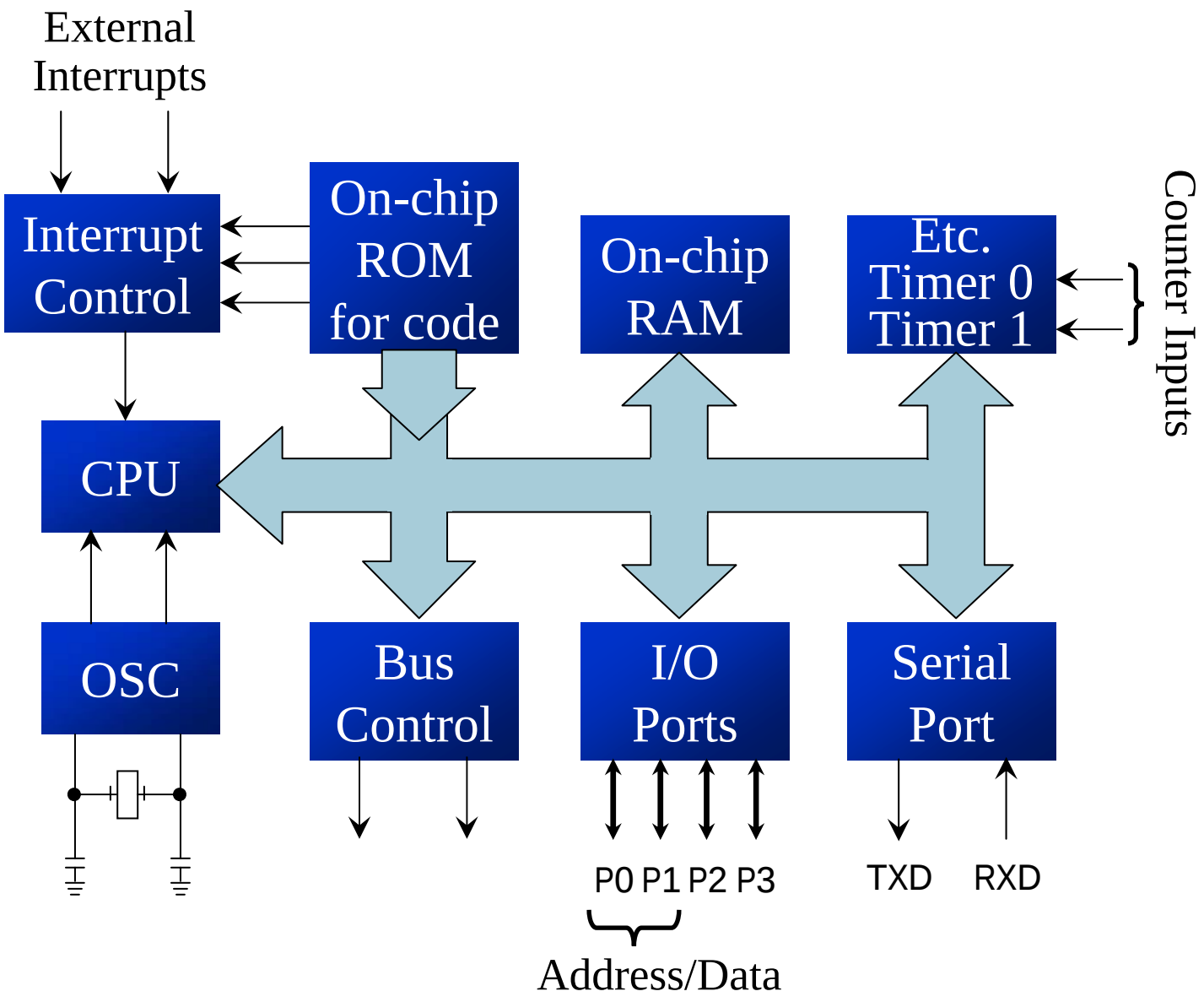
OVERVIEW OF 8051 FAMILY

8051 Microcontroller

- Intel introduced 8051, referred as MCS-51, in 1981
  - The 8051 is an 8-bit processor
    - The CPU can work on only 8 bits of data at a time
  - The 8051 had
    - 128 bytes of RAM
    - 4K bytes of on-chip ROM
    - Two timers
    - One serial port
    - Four I/O ports, each 8 bits wide
    - 6 interrupt sources
- The 8051 became widely popular after allowing other manufactures to make and market any flavor of the 8051, but remaining code-compatible

OVERVIEW OF 8051 FAMILY

8051 Microcontroller (cont')



- The 8051 is a subset of the 8052
- The 8031 is a ROM-less 8051
  - Add external ROM to it
  - You lose two ports, and leave only 2 ports for I/O operations

Feature	8051	8052	8031
ROM (on-chip program space in bytes)	4K	8K	0K
RAM (bytes)	128	256	128
Timers	2	3	2
I/O pins	32	32	32
Serial port	1	1	1
Interrupt sources	6	8	6

OVERVIEW OF 8051 FAMILY

Various 8051 Microcontrollers

- 8751 microcontroller
  - UV-EPROM
    - PROM burner
    - UV-EPROM eraser takes 20 min to erase
- AT89C51 from *Atmel Corporation*
  - Flash (erase before write)
    - ROM burner that supports flash
    - A separate eraser is not needed
- DS89C4x0 from *Dallas Semiconductor*, now part of *Maxim Corp.*
  - Flash
    - Comes with on-chip loader, loading program to on-chip flash via PC COM port

OVERVIEW OF 8051 FAMILY

Various 8051 Microcontrollers (cont')

- DS5000 from *Dallas Semiconductor*
  - NV-RAM (changed one byte at a time), RTC (real-time clock)
    - Also comes with on-chip loader
- OTP (one-time-programmable) version of 8051
- 8051 family from *Philips*
  - ADC, DAC, extended I/O, and both OTP and flash

INSIDE THE 8051

Registers

- Register are used to store information temporarily, while the information could be
  - a byte of data to be processed, or
  - an address pointing to the data to be fetched
- The vast majority of 8051 register are 8-bit registers
  - There is only one data type, 8 bits

INSIDE THE 8051

Registers (cont')

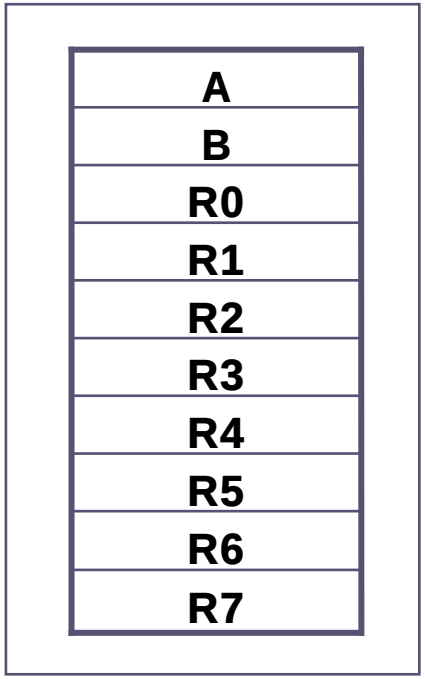
- The 8 bits of a register are shown from MSB D7 to the LSB D0
  - With an 8-bit data type, any data larger than 8 bits must be broken into 8-bit chunks before it is processed

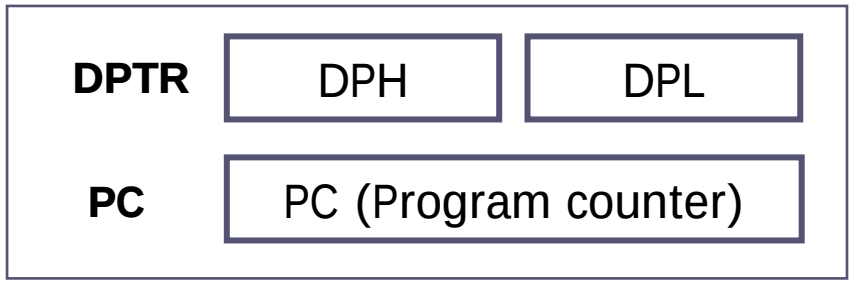
8 bit Registers

INSIDE THE 8051

Registers (cont')

- The most widely used registers
  - A (Accumulator)
    - For all arithmetic and logic instructions
  - B, R0, R1, R2, R3, R4, R5, R6, R7
  - DPTR (data pointer), and PC (program counter)





INSIDE THE 8051

MOV Instruction

MOV destination, source ;copy source to dest.

- The instruction tells the CPU to move (in reality, COPY) the source operand to the destination operand

"#" signifies that it is a value

```
MOV   A,#55H      ;load value 55H into reg. A
MOV   R0,A        ;copy contents of A into R0
                  ;(now A=R0=55H)
MOV   R1,A        ;copy contents of A into R1
                  ;(now A=R0=R1=55H)
MOV   R2,A        ;copy contents of A into R2
                  ;(now A=R0=R1=R2=55H)
MOV   R3,#95H     ;load value 95H into R3
                  ;(now R3=95H)
MOV   A,R3        ;copy contents of R3 into A
                  ;now A=R3=95H
```

INSIDE THE 8051

MOV Instruction (cont')

- Notes on programming
  - Value (proceeded with #) can be loaded directly to registers A, B, or R0 – R7
    - `MOV A, #23H`
    - `MOV R5, #0F9H`

Add a 0 to indicate that F is a hex number and not a letter

If it's not preceded with #, it means to load from a memory location

  - If values 0 to F moved into an 8-bit register, the rest of the bits are assumed all zeros
    - "`MOV A, #5`", the result will be A=05; i.e., A = 00000101 in binary
  - Moving a value that is too large into a register will cause an error
    - `MOV A, #7F2H ; ILLEGAL: 7F2H>8 bits (FFH)`

INSIDE THE 8051

ADD Instruction

```
ADD A, source     ;ADD the source operand
                  ;to the  accumulator
```

- The ADD instruction tells the CPU to add the source byte to register A and put the result in register A
- Source operand can be either a register or immediate data, but the destination must always be register A
  - "`ADD R4, A`" and "`ADD R2, #12H`" are invalid since A must be the destination of any arithmetic operation

```
MOV A, #25H       ;load 25H into A
MOV R2, #34H      ;load 34H into R2
ADD A, R2 ;add R2 to Accumulator
                  ;(A = A + R2)
```

There are always many ways to write the same program, depending on the registers used

```
MOV A, #25H       ;load one operand
            ;into A (A=25H)
ADD A, #34H       ;add the second
                  ;operand 34H to A
```

8051 ASSEMBLY PROGRAMMING

Structure of Assembly Language

- In the early days of the computer, programmers coded in *machine language*, consisting of 0s and 1s
  - Tedious, slow and prone to error
- *Assembly languages*, which provided mnemonics for the machine code instructions, plus other features, were developed
  - An Assembly language program consist of a series of lines of Assembly language instructions
- Assembly language is referred to as a *low-level language*
  - It deals directly with the internal structure of the CPU

8051 ASSEMBLY PROGRAMMING

Structure of Assembly Language

- Assembly language instruction includes
  - a mnemonic (abbreviation easy to remember)
    - the commands to the CPU, telling it what those to do with those items
  - optionally followed by one or two operands
    - the data items being manipulated
- A given Assembly language program is a series of statements, or lines
  - Assembly language instructions
    - Tell the CPU what to do
  - Directives (or pseudo-instructions)
    - Give directions to the assembler