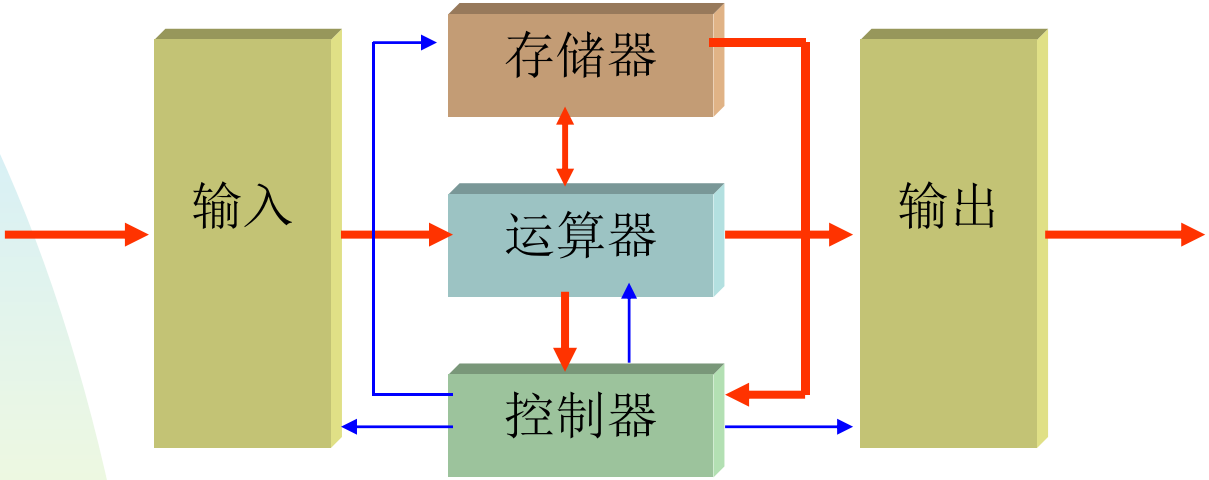
第1章 计算机系统概论

- 计算机的基本构成
- 计算机的系统结构
- 计算机的发展历史
- 数据编码和数据运算
- 嵌入式系统简介

1、 计算机系统的基本构成

- 1.1 计算机的基本构成
  - 1. 运算器
  - 2. 存储器
  - 3. 控制器
  - 4. 输入输出设备
  - 5. 系统连接
- 1.2 计算机软件概述
  - 1. 软件的分类
  - 2. 操作系统的概念
  - 3 计算机语言及其编译
- 1.3 计算机系统的历史与发展

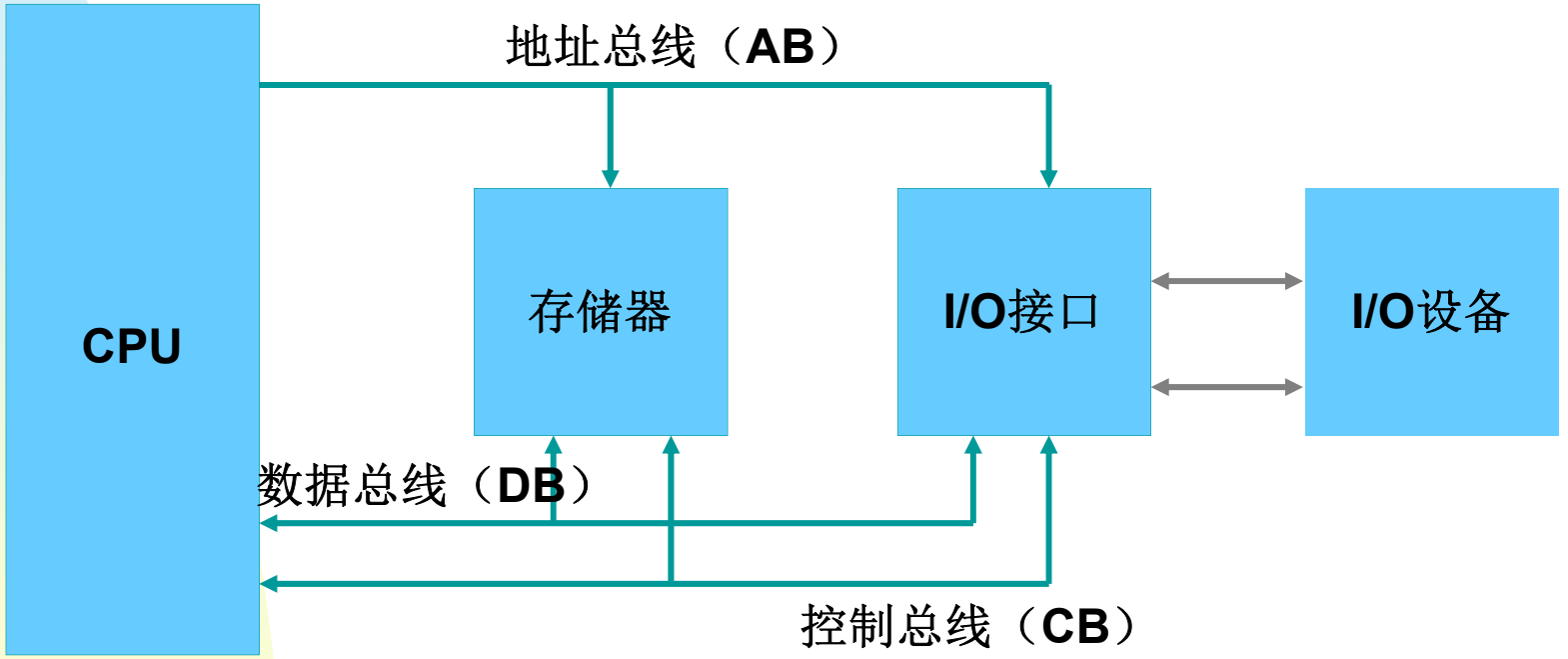
1.1 计算机的基本构成



微型计算机

- 以微处理器为核心
- 配上由大规模集成电路的存储器（**ROM/RAM**）、输入／输出接口（**I/O**）电路及系统总线（**BUS**）等所组成的计算机。
- 将这些组成部分集成在一片超大规模集成电路芯片上则构成单片微型计算机
  - 单片机
  - 嵌入式计算机

微型计算机的构成



ROM： 只读存储器

RAM： 读写存储器

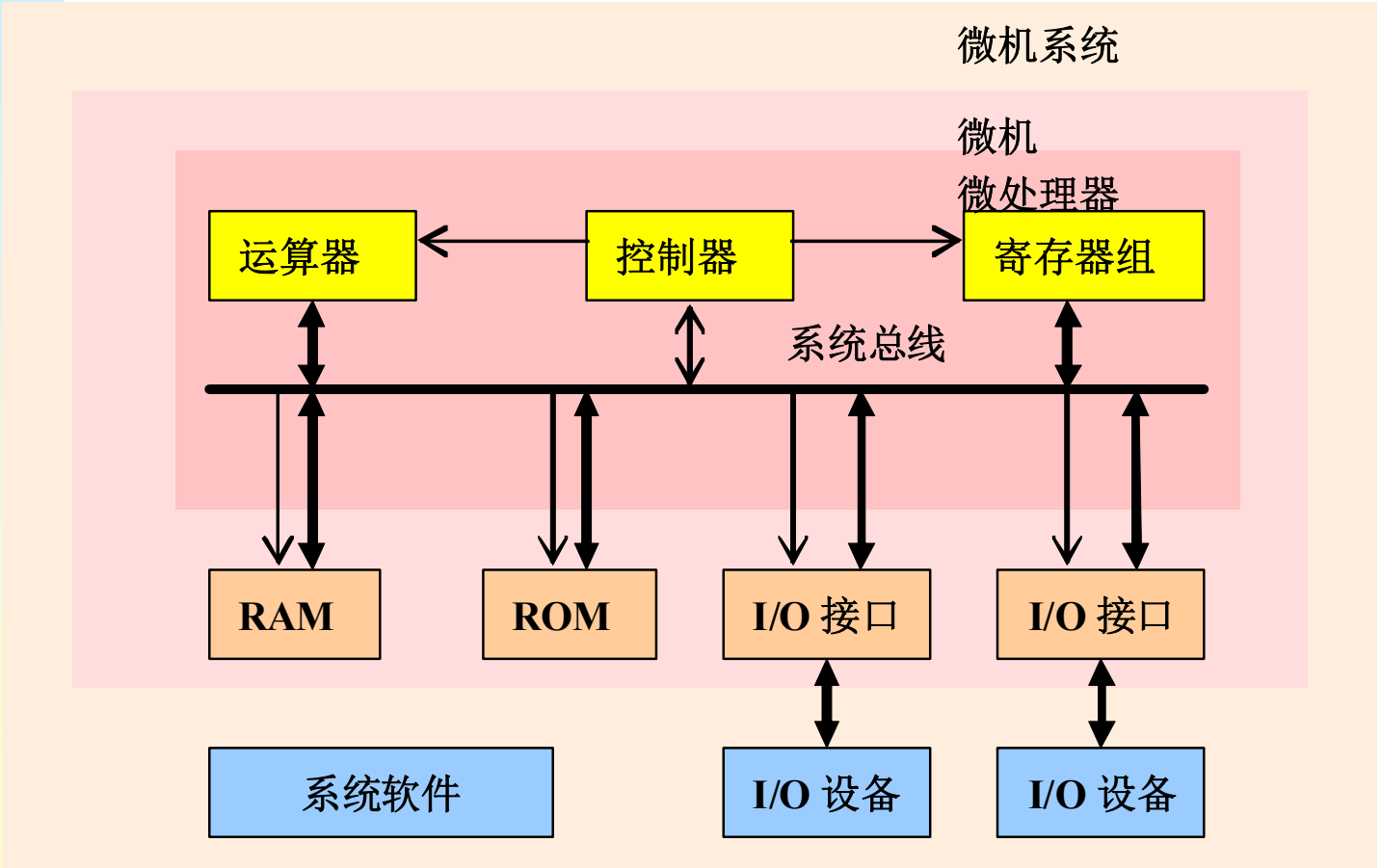
冯.诺依曼结构和哈佛结构

- 冯.诺依曼
  - 五大组成部分
    - 单存储器
  - 二进制
  - 存储程序
  - 控制器根据存放在存储器中的指令序列工作
  - 普林斯顿结构
- 哈佛结构
  - 双存储器
    - 指令存储器
    - 数据存储器

微型计算机系统

- 以微型计算机为中心
- 配以相应的外围设备以及控制微型计算机工作的软件
  - 系统软件
  - 应用软件

微型计算机系统的构成


微机系统
微机
微处理器
运算器
控制器
寄存器组
系统总线
RAM
ROM
I/O 接口
I/O 接口
系统软件
I/O 设备
I/O 设备

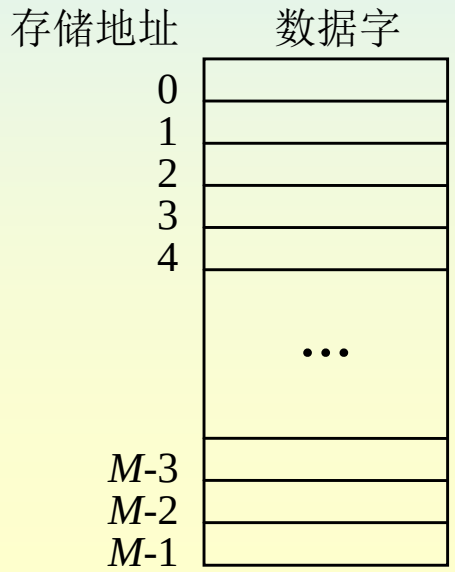
微型计算机系统的构成

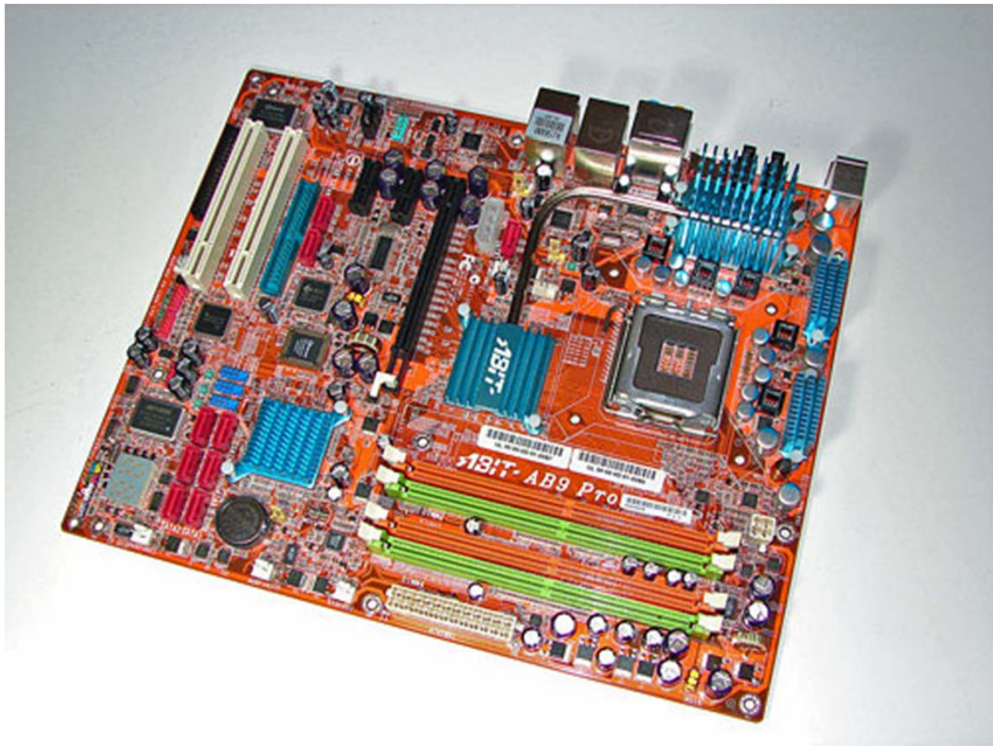
微型计算机系统（μCS）

- 微型计算机（μC）
  - 微处理器（μP）
    - 运算器（ALU）
    - 控制器
  - 内存储器
    - 寄存器
  - 输入/输出接口
- 输入/输出设备及外存储器
- 系统软件
- 应用软件
- 电源、面板、机架等

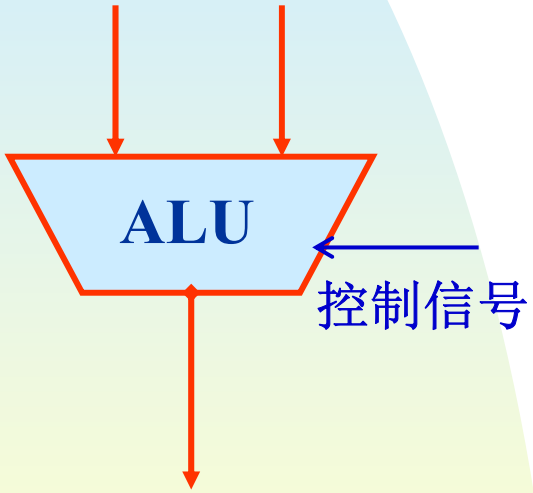
计算机的硬件结构

- **ALU**（运算器）
- 存储器（读、写、访问）
- 容量（字、字节）
- 字长
- 指令
- 程序
- **CPU**
- 主机
- 总线



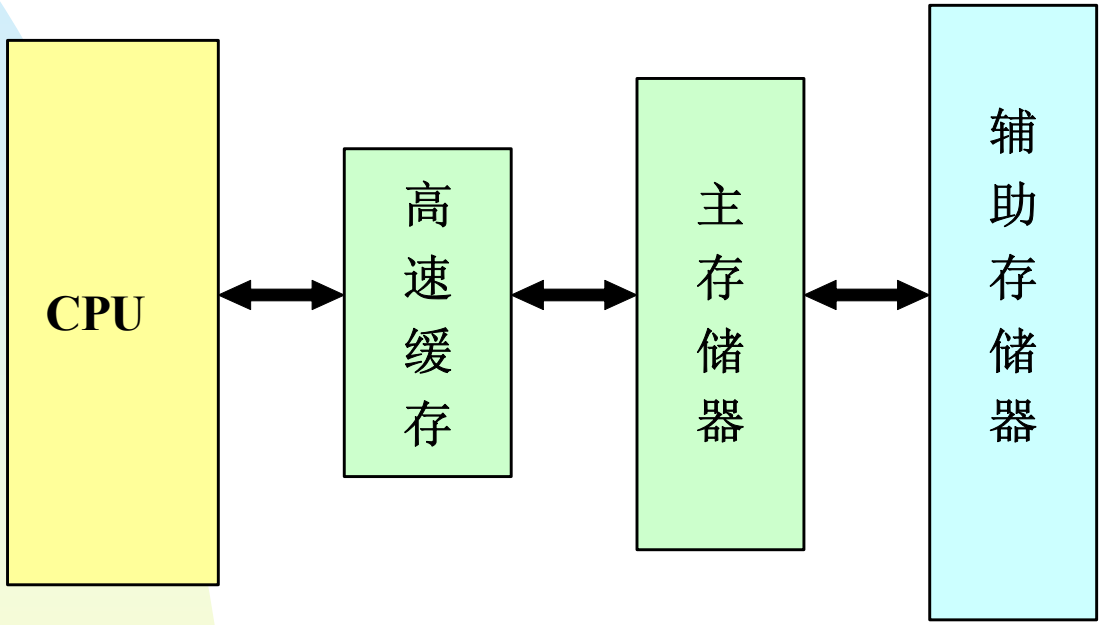


运算器



- 运算器是完成运算功能的部件。
- 运算器中有一个算术逻辑单元（ALU），它执行各种数据运算操作。
  - 算术运算：加、减、乘、除、数据格式转换。
  - 逻辑运算：按位对数据进行与、或、非、移位等运算。
- ALU是一个多功能的运算电路，进行何种运算取决于由控制器发出的控制信号。
- ALU由2个输入端，可同时输入两个参加运算的数据。
- 在运算器中有若干个临时存放数据的寄存器，由于存储最频繁使用的数据。

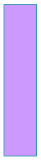
存储器

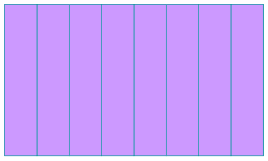


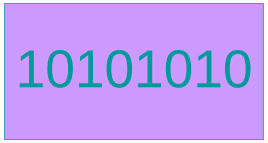
- 高速缓存：Cache
- 主存储器（内存）：（RAM+ROM）
- 辅助存储器（外存）：磁盘、U盘、磁带、光盘等。

存储器内的数据

- 位**bit**
  - 计算机所能表示的最小最基本的数据单位
  - 取值只能为**0**或**1**的一个二进制数值位
  - 记作**b**
- 字节**byte**
  - 由**8**个位二进制位组成
  - 用作计算存储容量的单位
  - 是所有存储器的基本存储数据单元。
  - 记作**B**
- **1KB = 1024B**
- **1MB = 1024KB**
- **1GB = 1024MB**
- **1TB = 1024GB**







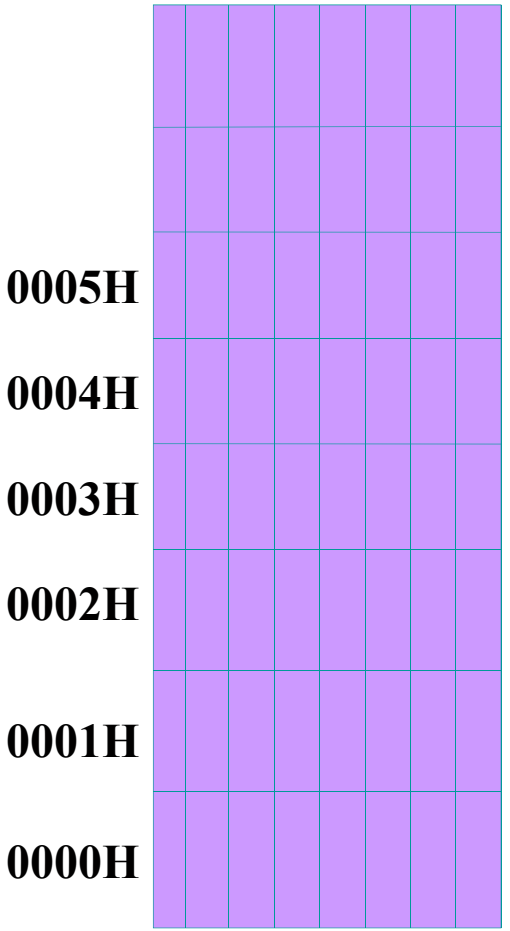
存储器内的数据

- 字（word）
  - 一次可以直接处理的二进制数码的位数
  - 通常取决于微处理器内部通用寄存器的位数和数据总线的宽度
    - 如CPU的数据总线是16位的，
      - 1word=16bit
    - 如CPU的数据总线是32位的，
      - 1word=32bit
    - 字长反映了计算机中并行运算的能力。
- 双字
- 四倍字

存储器

- 字数：存储器的地址范围
  - 所需要的地址总线
- 位数：存储器的数据范围
  - 所需要的数据总线

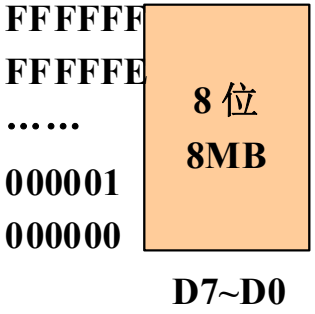
通常带地址的存储器基本单元
都是**1Byte（8bit）**的数据容量



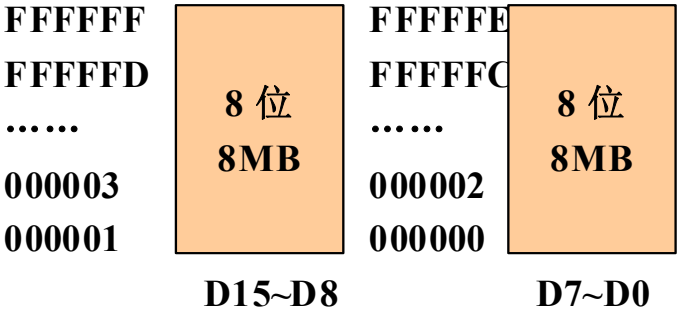
存储器组织

- 如系统所需的存储器：
  - 数据位数超过存储芯片的数据总线位数，则需要进行位扩展。
  - 地址范围超过存储芯片的地址表示范围，则需要进行字扩展。

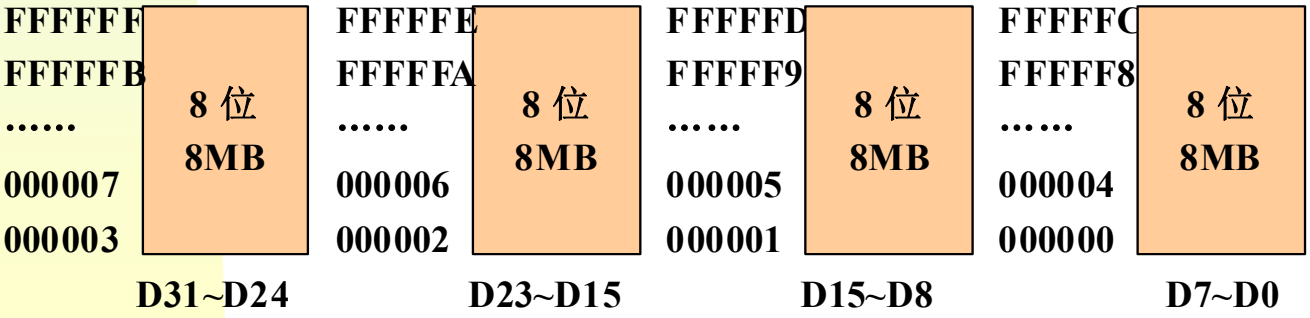
- 8位



- 16位



- 32位



控制器—指令

- 控制器在计算机指令的控制下进行工作。
- 计算机指令是一种经过编码的操作命令。
  - 它指定需要进行的操作，支配计算机中信息的传递以及计算机与I/O设备间的传递。
- 控制器对指令进行译码，并根据指令的操作要求指挥所有其它部件的工作，为此它根据指令生成一系列时序控制信号，控制其它单元工作。
- 控制器不断地从存储器中读取指令，然后分析指令的含义（译码），并执行该指令的操作，执行完成以后又从存储器中读取下一条指令。

控制器—程序

- 一条计算机指令的功能是有限的，完成复杂的运算功能需要将多条指令组合起来构成一个指令序列。这样的一个完成某种功能的指令序列成为程序。
- 指令在计算机中用二进制的代码（机器码）表示，以便于硬件的识别。
- 程序在执行前存储在主存储器中，控制器通常按指令的顺序自动地从存储器中取出指令并依次执行，或者根据指令决定执行的顺序（如跳转指令等）。

CPU、主机

- 运算器和控制器一起构成了计算机的中央处理器（**Central Processing Unit，CPU**）。它是计算机的核心部件。
- 通常还把**CPU**、存储器和输入/输出接口电路和在一起构成的电路系统称为主机（也即微型计算机）
- 连接计算机各个部分的方式可以采用总线的方式。

总线（BUS）

- 总线是计算机中连接各个功能模块的纽带，是计算机各模块之间进行信息传输的公共线路。
  - 连接在总线上的模块分为发送模块和接收模块，构成信息的接收方和发送方。
  - 总线上的设备可分为主设备和从设备两大类。
    - 总线主设备：能够启动总线服务的设备（如**CPU**）。
    - 总线从设备：只能等待启动命令的被动型设备。

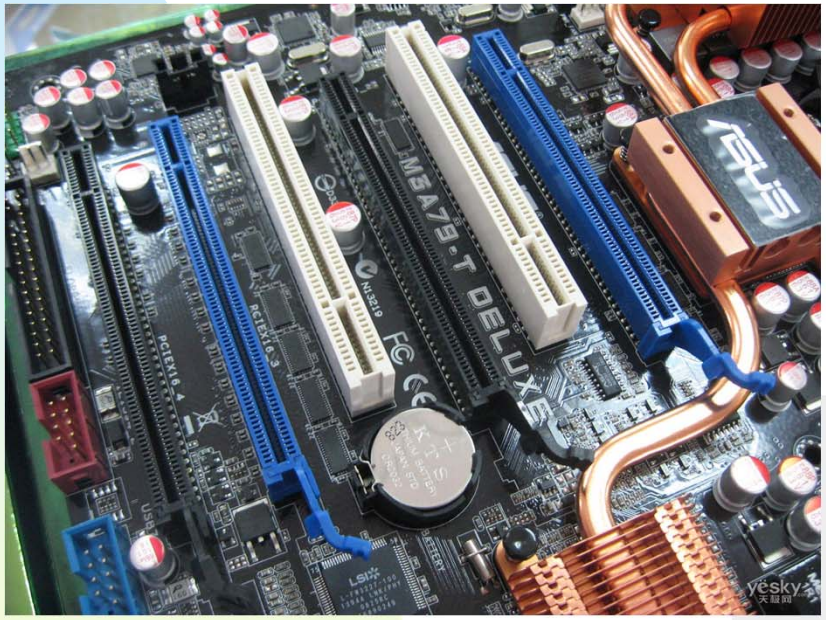
总线的分类

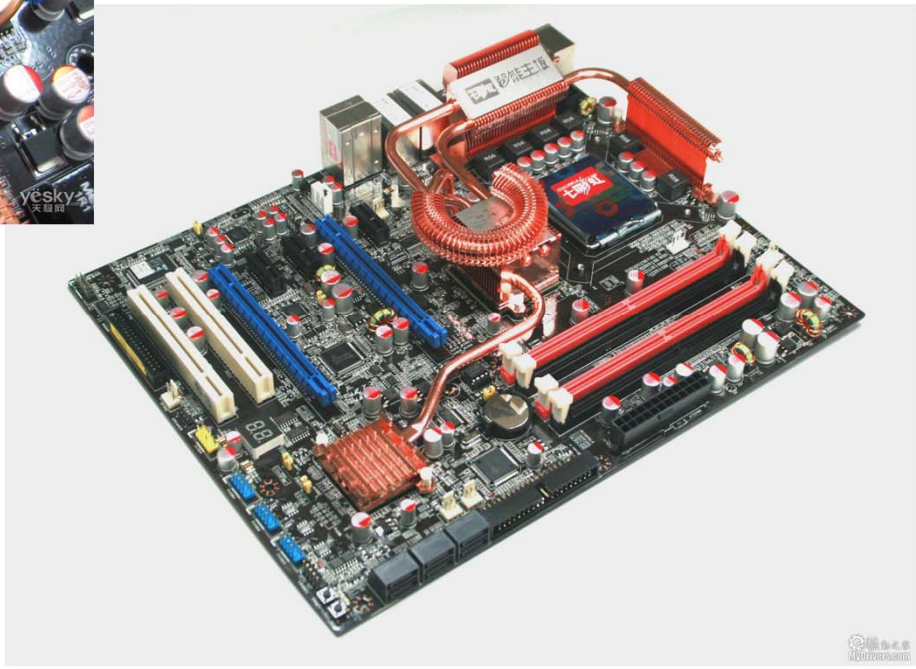
- 可按以下特性来对总线进行分类：
  - 物理特性
  - 功能特性
  - 电气特性

物理特性

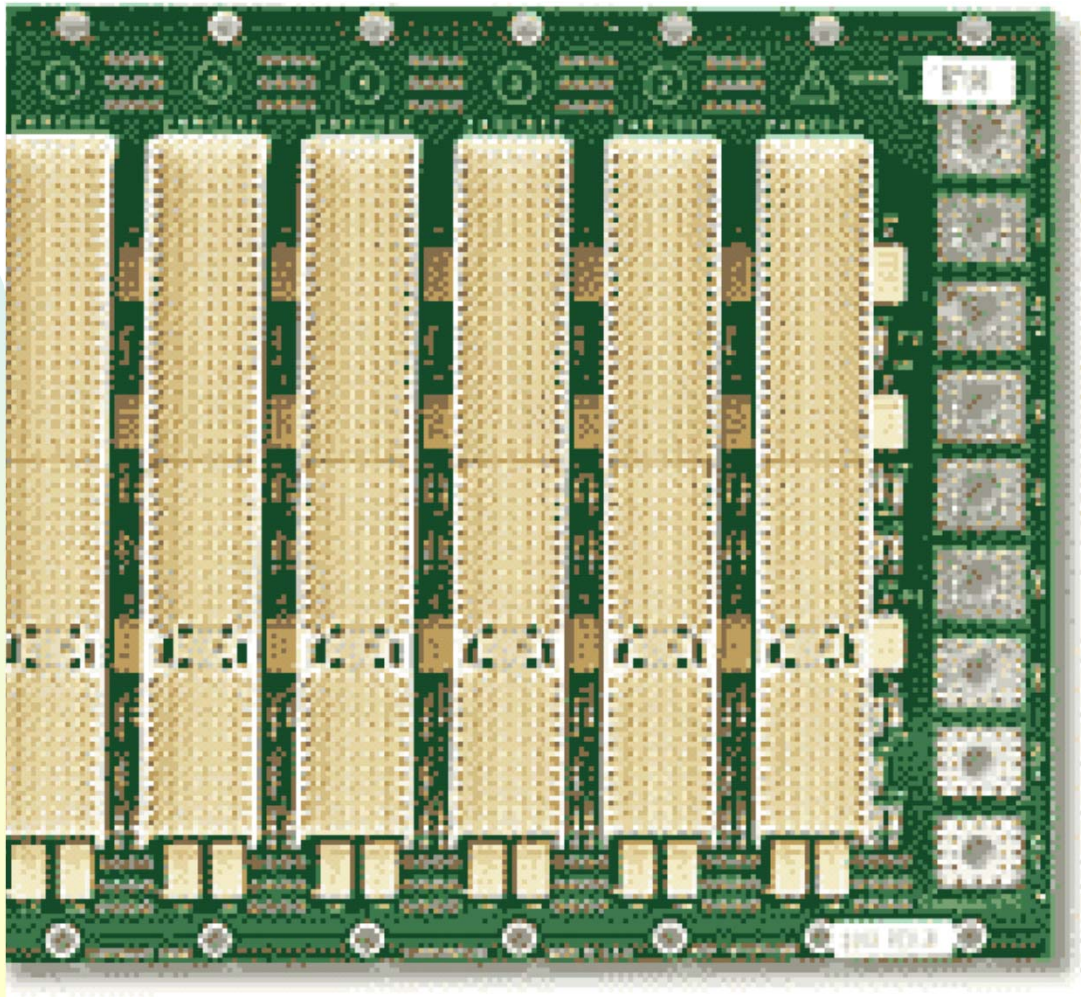
- 按连线的类型
  - 电缆式：通常采用扁平电缆连接线路板
  - 主板式：在主机板上采用插槽方式供功能板插入。
  - 背板式：在机箱中设置一个专门的总线插槽板。
- 按连线的数量
  - 串行总线：用一条数据线进行数据传输。
  - 并行总线：一般有8位、16位、32位、64位总线。

主板

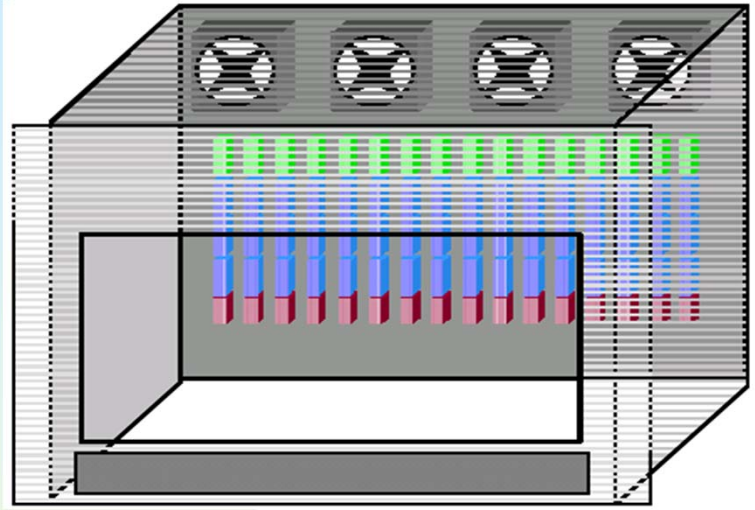


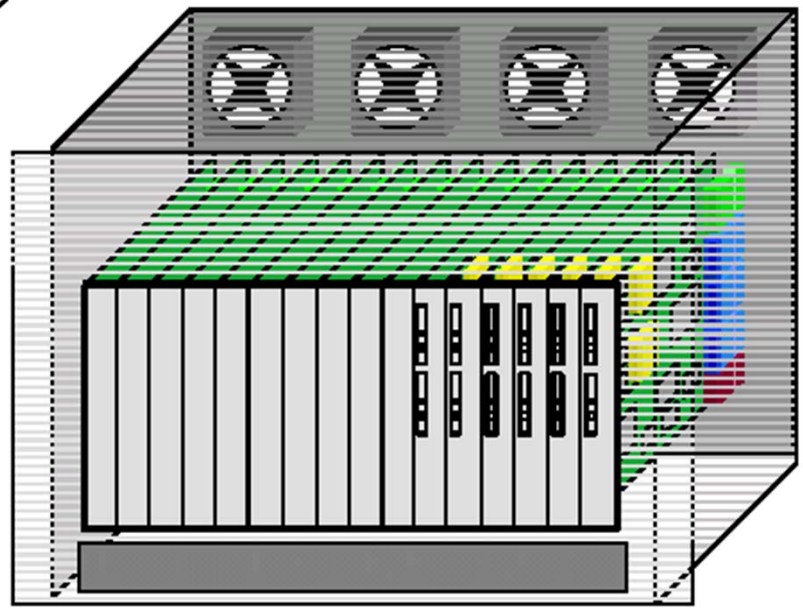


背板



背板总线与机箱





功能特性

- 按功能层次
  - 芯片级总线：CPU芯片内部的总线，也称内部总线。
  - 板级总线：连接CPU、主存和I/O接口等模块，也称局部总线。
  - 系统级总线：连接系统中的各个功能模块。
- 按资源类型
  - 处理器总线（系统总线）：连接处理器、主存与外设。
    - 面向单处理器的和面向多处理器的。
  - 输入输出总线：连接主机与外围设备。

电气特性

- 数据传输方向
  - 单工：单向传输总线。
  - 双工：双向传输总线。
    - 半双工：只能在两个方向上轮流传输信息。
    - 全双工：可在两个方向上同时传输信息。
- 定时特征(clocking)
  - 同步：数据传输速率是固定的。
  - 异步：数据传输速率是可变的。

系统总线-简单的总线结构

地址总线 Address Bus

- CPU用来向存储器或I/O端口传送地址
- 单向
  - 由CPU发出
- 位数n决定了CPU可直接寻址的内存容量
  - 2^n

数据总线 Data Bus

- **CPU**与存储器及外设交换数据的通路
- 双向、三态
- 位数与微处理器的位数相同

控制总线 Control Bus

- 用来传输控制信号
- 由两种方向的单向控制信号组成
  - 命令信号线（**CPU→MEM/IO**）
  - 状态信号线（**MEM/IO →CPU**）

系统总线

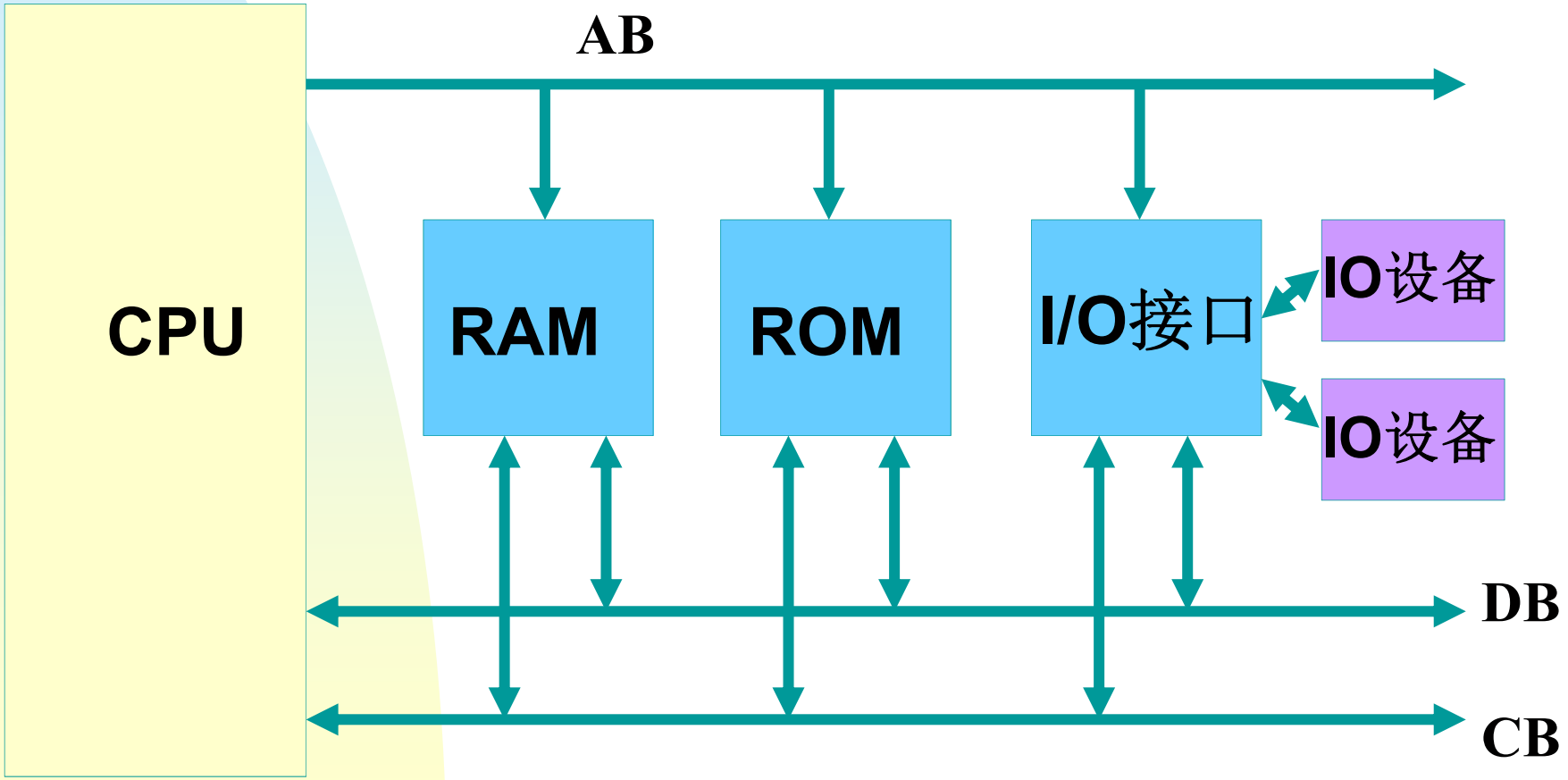
- PC总线
  - 62芯 8位 1M/s
- ISA总线
  - 62+36芯 16位 8M/s
- VESA总线
  - 116芯 32/64位
- PCI总线
  - 124芯 32/64位 132-264M/s

总线结构

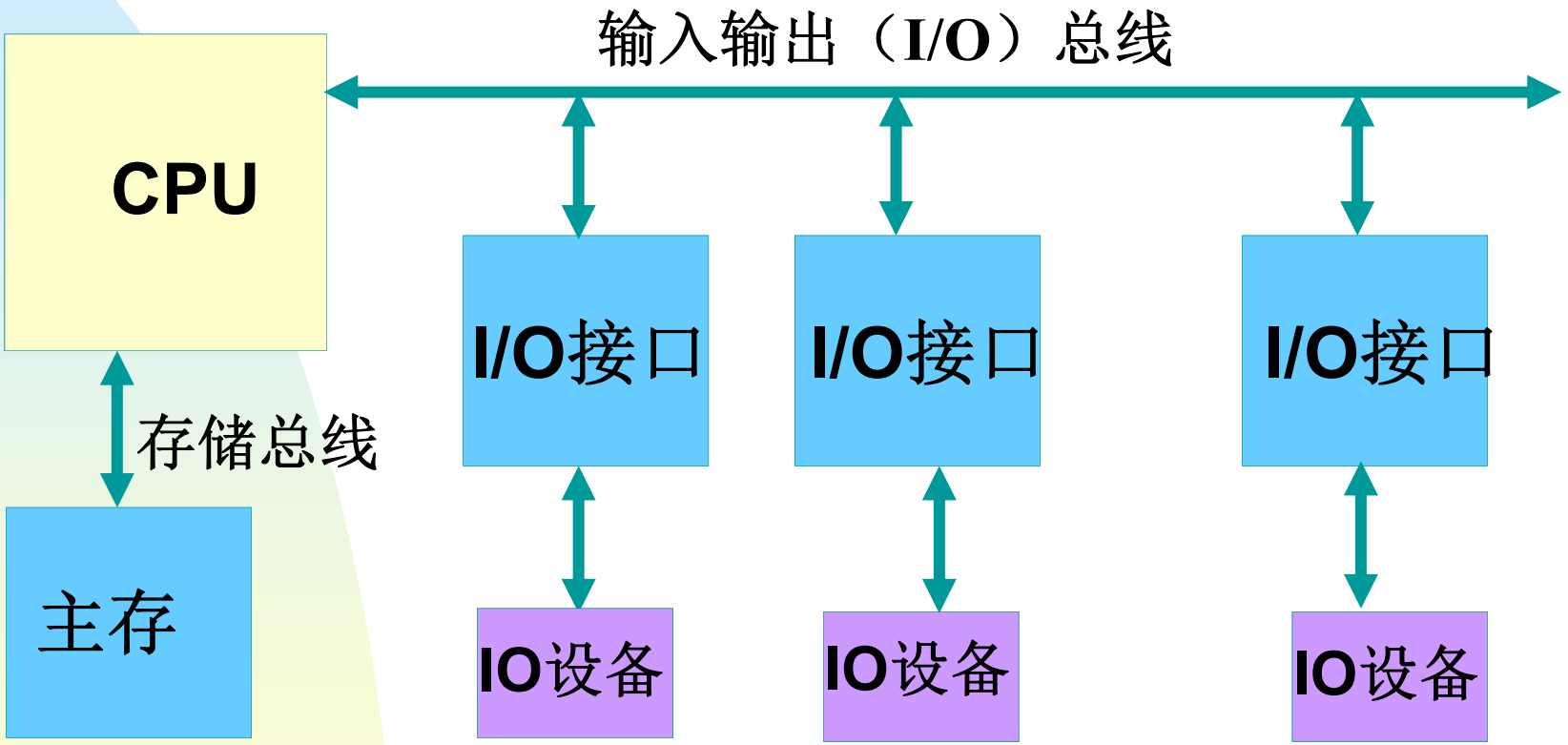
- 单总线结构
- 双总线结构
  - 面向CPU
  - 面向主存

单总线结构

- 各模块之间的信息传递都通过单总线进行。
- 优点：控制简单，易于扩充配置I/O设备。
- 缺点：所有设备都连在一组总线上，总线只能分时工作，使数据传输量受限。

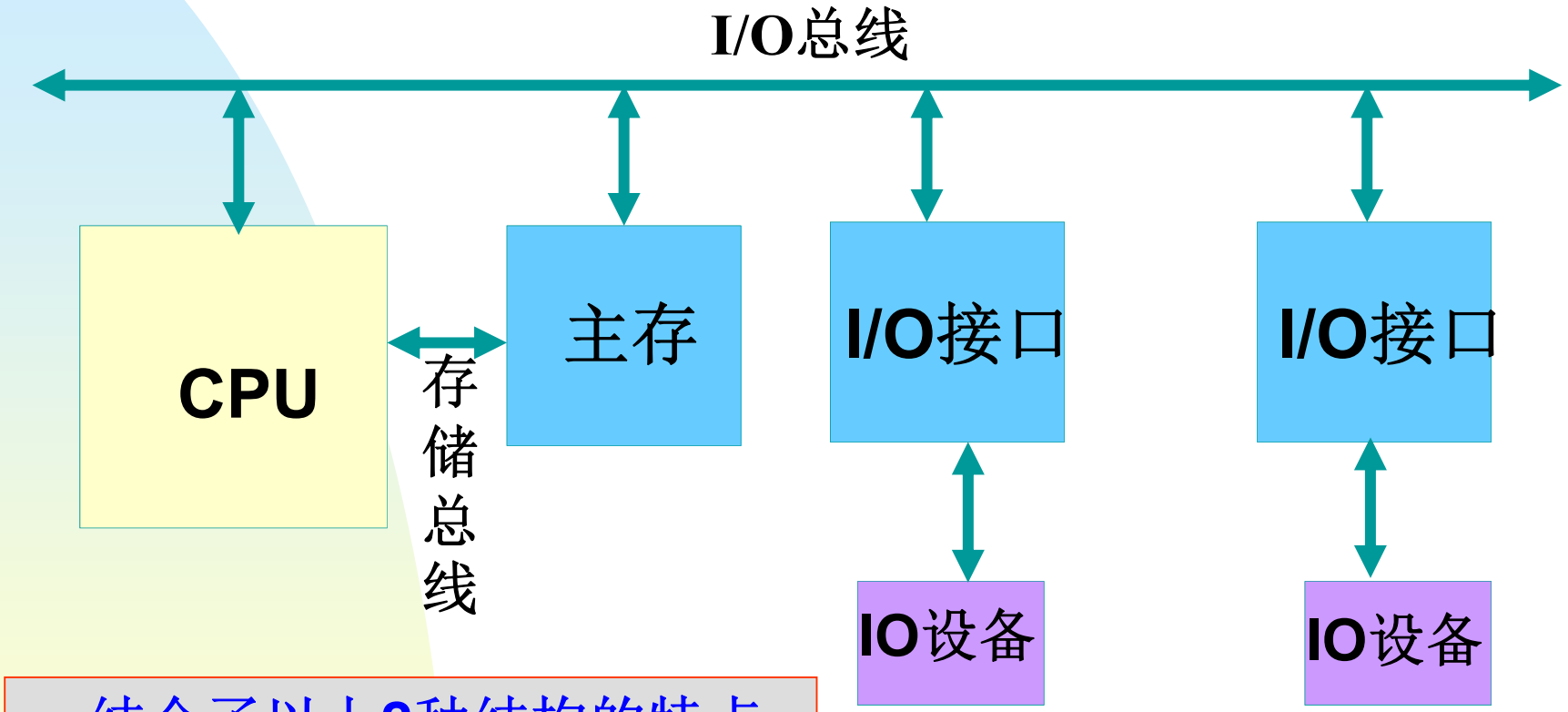


面向CPU的双总线结构



- 通过存储总线使CPU对主存进行读/写操作。通过I/O总线让CPU与I/O设备进行数据交换。
- 优点：提高了微机系统数据传输效率。
- 缺点：外设和主存之间没有直接通路，要通过CPU进行信息交换，降低了CPU的工作效率

面向主存的双总线结构



- 结合了以上2种结构的特点。提高了信息传送效率，同时也不降低CPU的工作效率。

微型计算机的性能指标

- 主频：计算机的晶振，反映时钟周期的大小。
- 字长：**CPU**的数据位数，反映**CPU**并行处理能力。
- 内存容量
- 存取周期
- 响应时间：用户向计算机系统发出一个请求后，到系统对该请求做出响应并获得其结果所需的等待时间。
- 吞吐率：系统响应用户请求的速率。
- 运算速度
  - **MIPS（Million Instruction Per second)**：反映计算机每秒可执行的指令数。

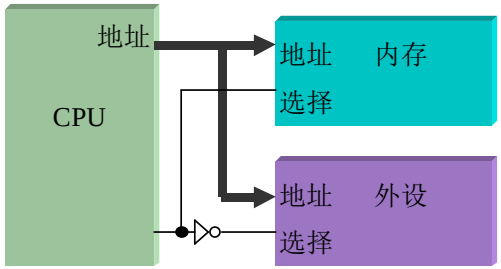
输入输出设备

- 输出设备
  - 1. 显示器
  - 2. 打印设备
  - 3. 绘图仪
- 输入设备
  - 键盘
  - 鼠标器
- 外存储设备

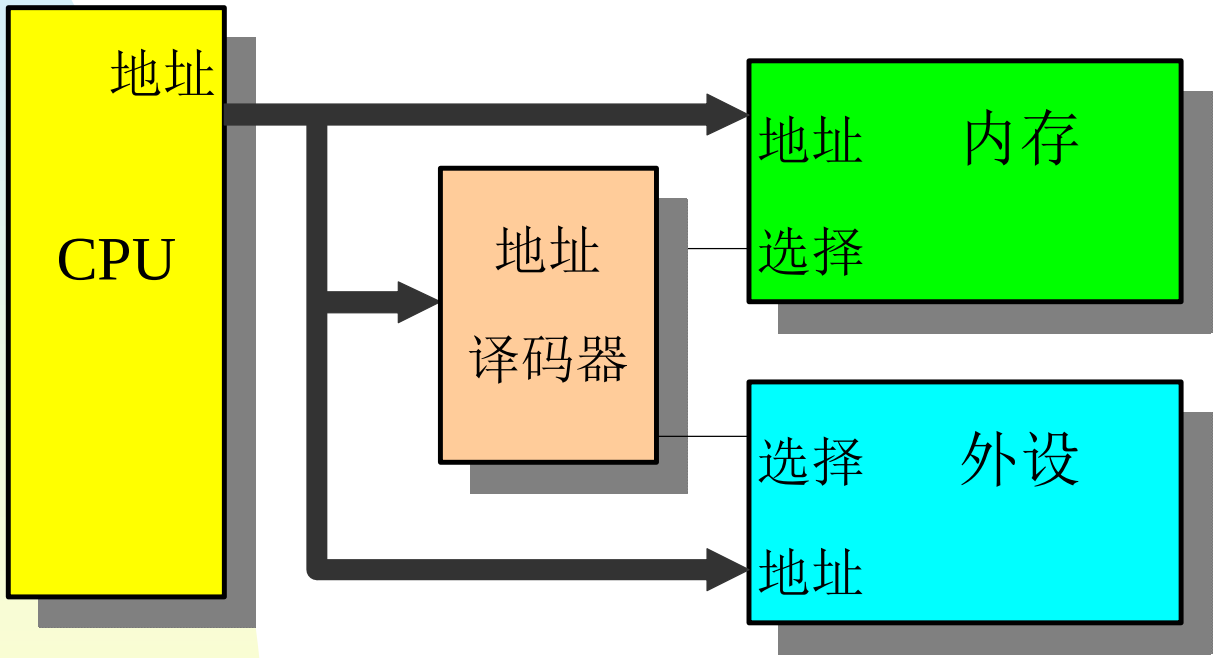
基本的输入输出方式

输入输出设备的寻址

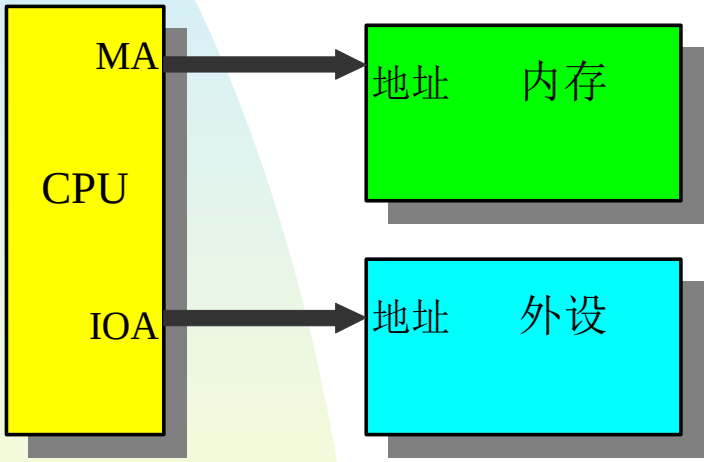
- 统一编址法
  - 存储器映像的外设寻址
  - 将接口中的控制寄存器、数据寄存器、状态寄存器和内存单元一样看待
  - 接口与存储器采用不同的地址
  - 可以利用访存指令进行输入输出操作
- 单独编址法
  - 两个地址空间
  - 访问存储器和访问外围设备采用不同的指令



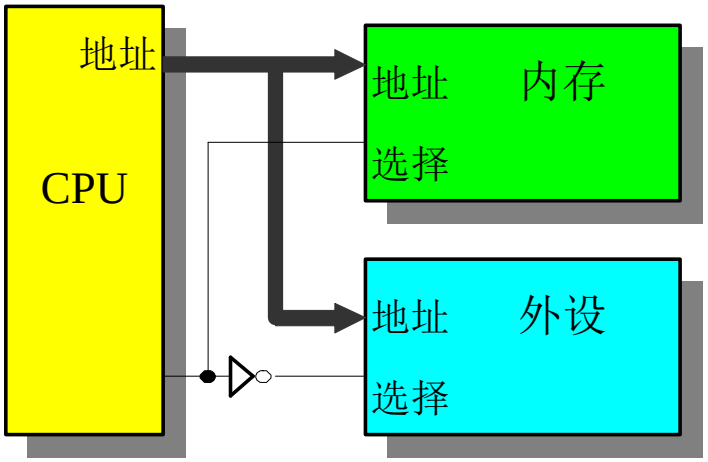
统一编址法



单独编址法



两套地址线



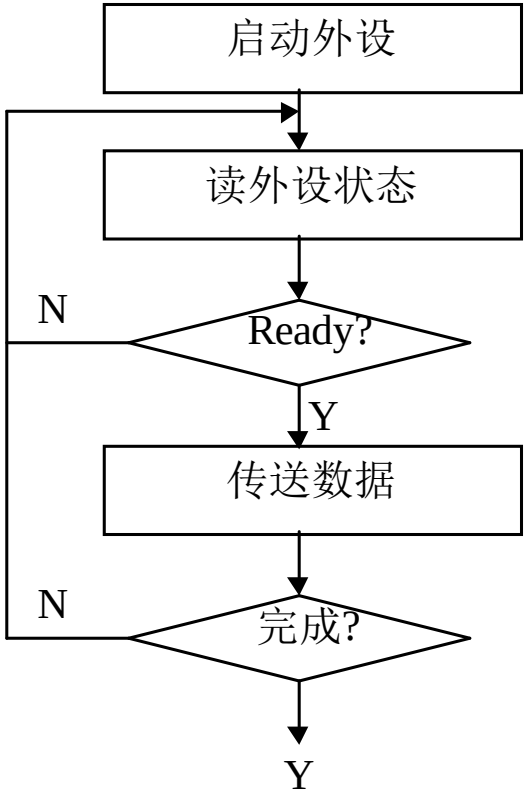
一套地址线

实现输入输出数据传送的方式

- 程序控制方式
  - 程序查询
  - 中断
- 直接存储器访问(DMA)方式
- 通道方式

程序控制方式

- 程序查询



中断方式

基本概念

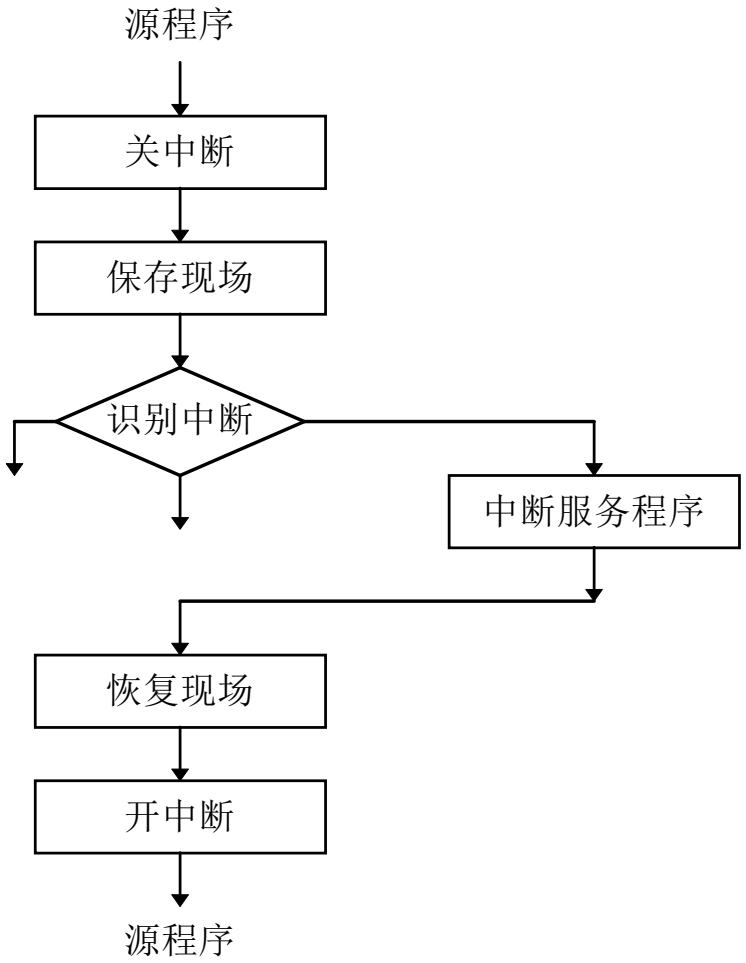
- 在发生了一个外部的事件时调用相应的处理程序的过程
  - 中断服务程序
- 中断服务程序与中断时CPU正在运行的程序是相互独立的
  - 相互不传递数据。

中断处理中的问题

- 设备识别方式
- 中断处理程序入口地址的形成
- 中断裁决
- 中断屏蔽

中断响应过程

- 简单的中断处理过程
  - ① 关中断
  - ② 保存现场
  - ③ 识别中断
  - ④ 形成服务程序入口地址
  - ⑤ 执行服务程序
  - ⑥ 恢复现场
  - ⑦ 开中断



中断屏蔽

- 多重中断
  - 中断服务程序也可以被中断
  - 中断嵌套
- 实现方法
  - 给CPU及中断请求都设置优先级
  - 多级中断

多重响应过程

① 关中断
② 保存现场
③ 识别中断
④ 形成服务程序入口地址
⑤ 开中断
⑥ 执行服务程序
⑦ 关中断
⑧ 恢复现场
⑨ 开中断

1.2 计算机软件

- 系统软件：使计算机系统功能完整，并为应用提供了一个平台。
  - 操作系统
  - 编译程序
  - 解释程序
- 应用软件：面向用户应用的功能软件，专门为解决某个应用领域中的具体任务而开发。
  - 多媒体软件、印刷排版软件、数据处理软件、自动控制软件等。
- 虚拟机(virtual machine)：指通过软件模拟的具有完整硬件系统功能的、运行在一个完全隔离环境中的完整计算机系统。
- 软件与硬件的等效性
- 固件

层次化结构

- 应用软件、系统软件和硬件构成了计算机系统的三个层次。
  - 应用软件为用户提供了应用系统的界面，使用户方便地使用计算机解决具体问题。
  - 系统软件则向用户提供了一个基本的操作界面，并向应用软件提供功能上的支持。
  - 硬件系统是整个计算机系统的基础和核心，所有的功能最终由硬件完成。

层次化结构

- 计算机系统按功能可划分成多层次结构。
  - 在硬件之上有操作系统级、汇编语言级、高级语言级和应用语言级。
- 计算机硬件也是一个层次化结构。
  - 可分为微系统结构级、寄存器级和电路级。
  - 在实际硬件以上所有的机器层次都成为虚拟机，它们都是由软件构成的外部特性。
- 一般将软件也分成几个层次。
  - 从而使得从某一较高层次上观察计算机时看不到较低层次的细节，这样可比较方便地了解计算机某一方面的特点。
  - 这种分层就形成了计算机系统的不同虚拟机。处于某一级虚拟机层次的程序员只需知道这一级的虚拟机特点，其下层的特性无需知道，即下层特性对该层程序员是透明的。

虚拟机

- 计算机硬件的特征对操作系统用户和应用程序用户是透明的。
  - 用户看不到计算机的硬件特征，也无需关心它们。
  - 安装了某个操作系统后，用户不必关心**CPU**是什么厂家的，有什么特点，因为操作都是一样的。
- 操作系统虚拟机对应用程序用户是透明的。
  - 应用程序使得用户无需关心操作系统虚拟机的特征。
  - 安装了某种应用软件后，不管它运行在什么操作系统上，用户的操作都是一样的。

软件与硬件的等效性

- 计算机系统的大部分功能既可以用硬件实现，又可以用软件实现。
  - 如64位数据运算、浮点数据运算、图形处理等功能在某些计算机中可用硬件实现，在另一些计算机中则可用软件实现。
  - 计算机主机的功能的这两种实现在逻辑上是等效的，其区别在于速度、成本、可靠性、存储容量、变更周期等因素。
    - 用硬件实现的功能性能较高，成本也高，而且硬件不易改变，灵活性较差。
    - 具有相同功能的计算机系统，它们的软、硬件之间的功能分配，可在很宽的范围内变化，没有固定的界限。
    - 从功能上看，软件是硬件的扩充。软件和硬件之间的界面是计算机的指令系统。
    - 随着大规模集成电路技术的发展，器件的功能越来越强，硬件实现的功能在逐步增加。

固件（Firmware）

- 通常可把固定不变的常用软件固化在硬件中，如写入只读存储器（**ROM**）中，成为固件。
- 固件是介于硬件和软件之间的实体。
  - 其设计方法类似于软件，而实现形态上则类似于硬件。
- 固件的应用例子：
  - 固化在**PC**机中**ROM-BIOS**的启动软件。
  - 固化在**ARM**、**DSP**、**FPGA**等的软件。
  - 固化在各类电子设备（机顶盒、各类智能仪器等）的软件。

操作系统

- 操作系统是最主要的系统软件，它管理系统资源，为应用程序提供运行环境并为用户提供操作界面。
  - 存储管理：
    - 内存管理：管理内存的分配。
    - 外存管理：管理磁盘存储区和文件结构。
  - 命令处理：用户给操作系统命令启动一个程序的运行，以完成某一项系统操作和应用操作。
  - 进程管理：计算机可以同时启动多个进程（任务）。
  - 设备管理：管理各种I/O设备（磁盘、鼠标、打印机等），提供统一的程序设计界面，为设备的共享使用和管理提供方便。
  - 网络通信管理：实现某种网络通信协议，管理通信方式，为计算机之间的操作和程序设计提供方便的界面。
  - 新型操作系统还提供病毒防护、数据加密等安全性能。

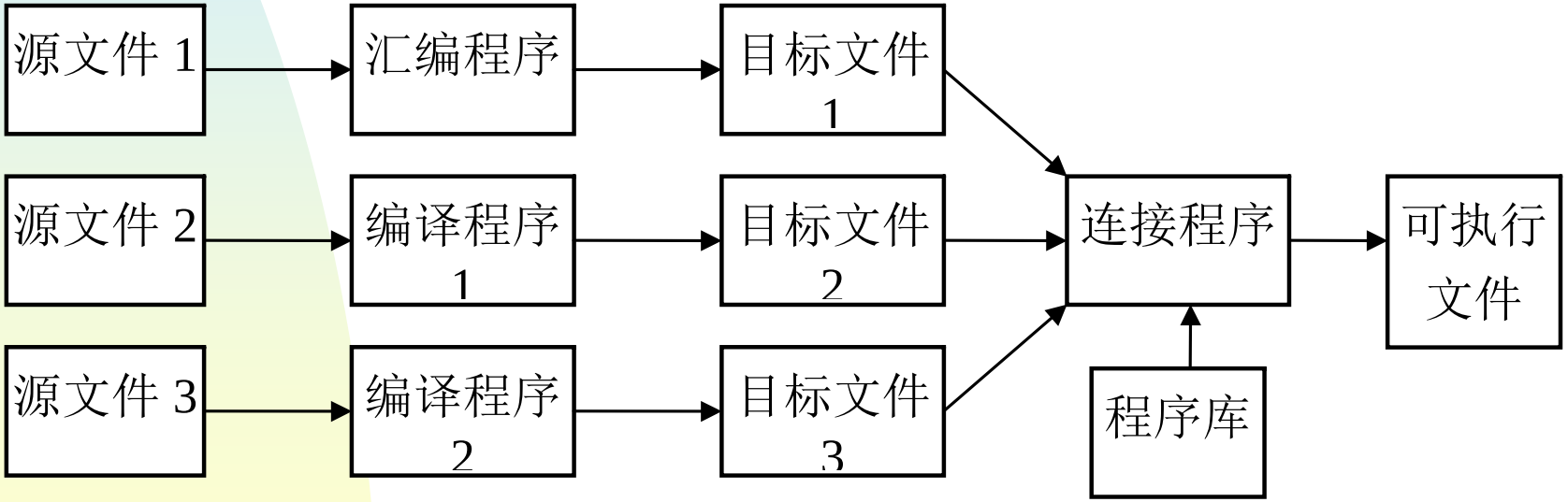
操作系统

- 交互操作系统：用于PC机和服务器（DOS、Windows等）
  - 是一个应用程序的运行平台，为应用程序的运行提供基础设施和环境。
  - 应用程序可以随时加在到计算机系统中，并作为一个进程运行。运行完成后退出系统，释放所有资源。
  - 在PC机中，操作系统分为2个层次：
    - 内核：提供最基础的机制。
    - 外围：提供与应用程序的接口。
- 实时操作系统：通常用于嵌入式系统中（VxWorks、RT-Linux等）
  - 响应的及时性：要求能对外部的事件做出及时的反应，要求系统响应事件短。
  - 响应时间的确定性：要能够确保响应时间的上限。
  - 运行稳定、低成本、系统规模小、根据具体需求可裁剪。

计算机语言及其编译

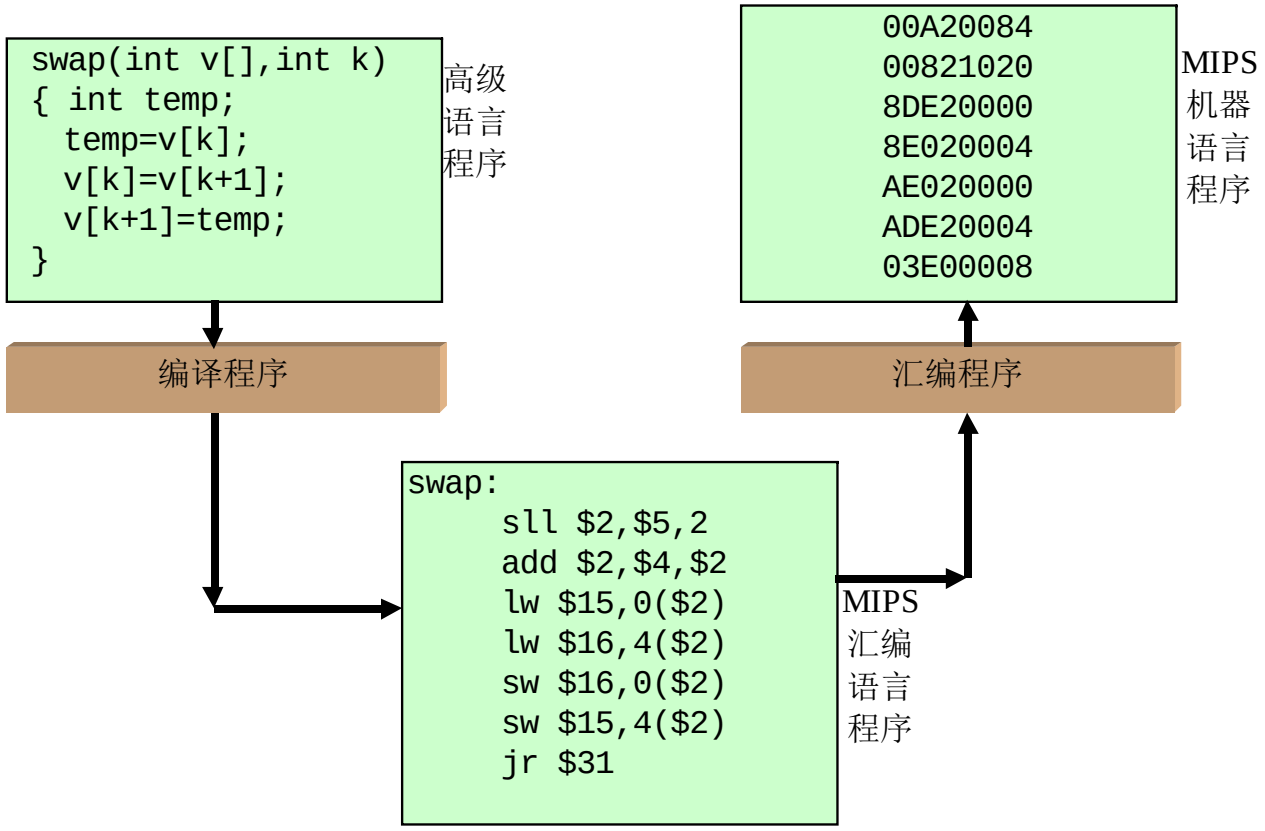
- 机器语言：二进制代码表示、能被计算机硬件直接识别的语言。
  - 000 001 010
- 汇编语言：采用文字符号（助记符）表示的机器语言，便于程序员记忆。
  - ADD R1, R2
- 高级语言：与计算机结构无关的程序设计语言，具有更强的表达能力。（C、Pascal、Fortran、Basic语言等）
  - A=A+B
- 应用语言：各种应用程序中使用的语言。
  - SQL、HTML等。

程序设计语言的编译


源文件 1
汇编程序
目标文件 1
源文件 2
编译程序 1
目标文件 2
连接程序
可执行文件
源文件 3
编译程序 2
目标文件 3
程序库

编译过程

- 词法分析
- 语法分析
- 生成中间代码
- 代码优化
- 生成目标代码



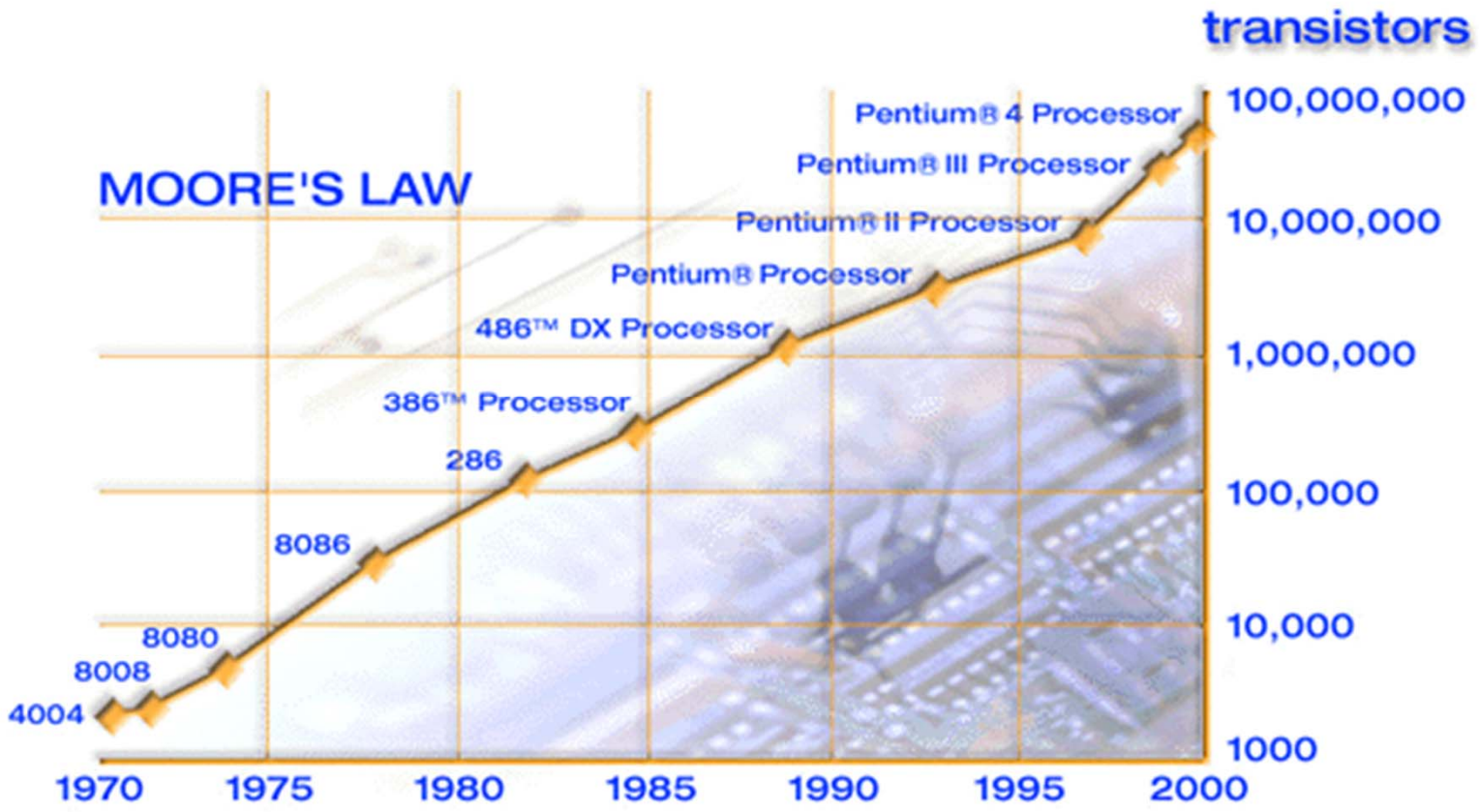
1.3 计算机系统的历史与发展

1.3.1 计算机的发展历史

- **ENIAC**
  - 美国第一台由程序控制的电子数字计算机
  - 全电子，不存储程序，十进制
- **EDVAC**
  - 二进制，存储程序，冯•诺依曼结构
- **IBM、DEC、Cray、Intel**
- 电子管、晶体管、集成电路、大规模集成电路和超大规模集成电路

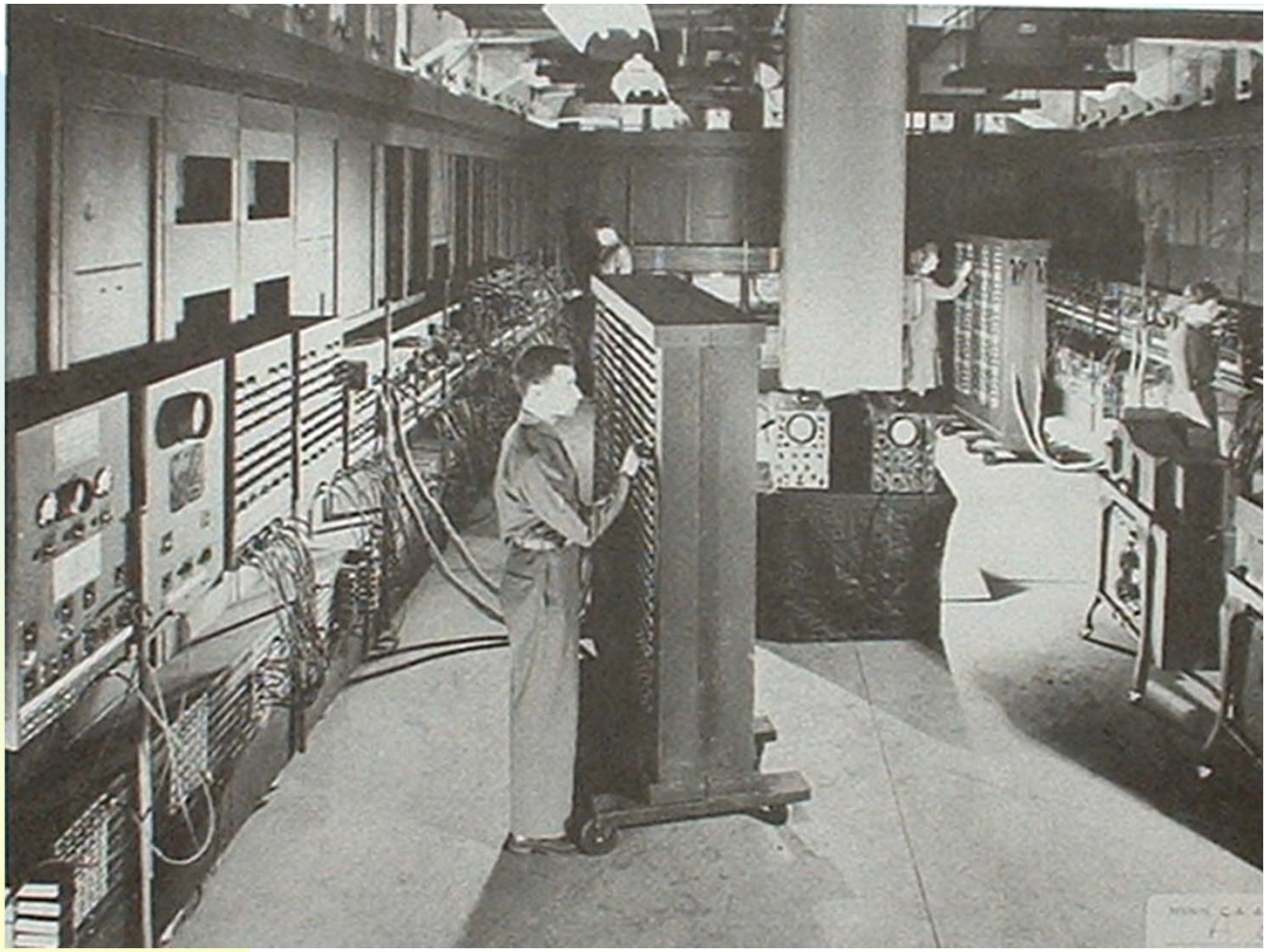
摩尔定律

- 芯片的容量每**18**个月增加一倍

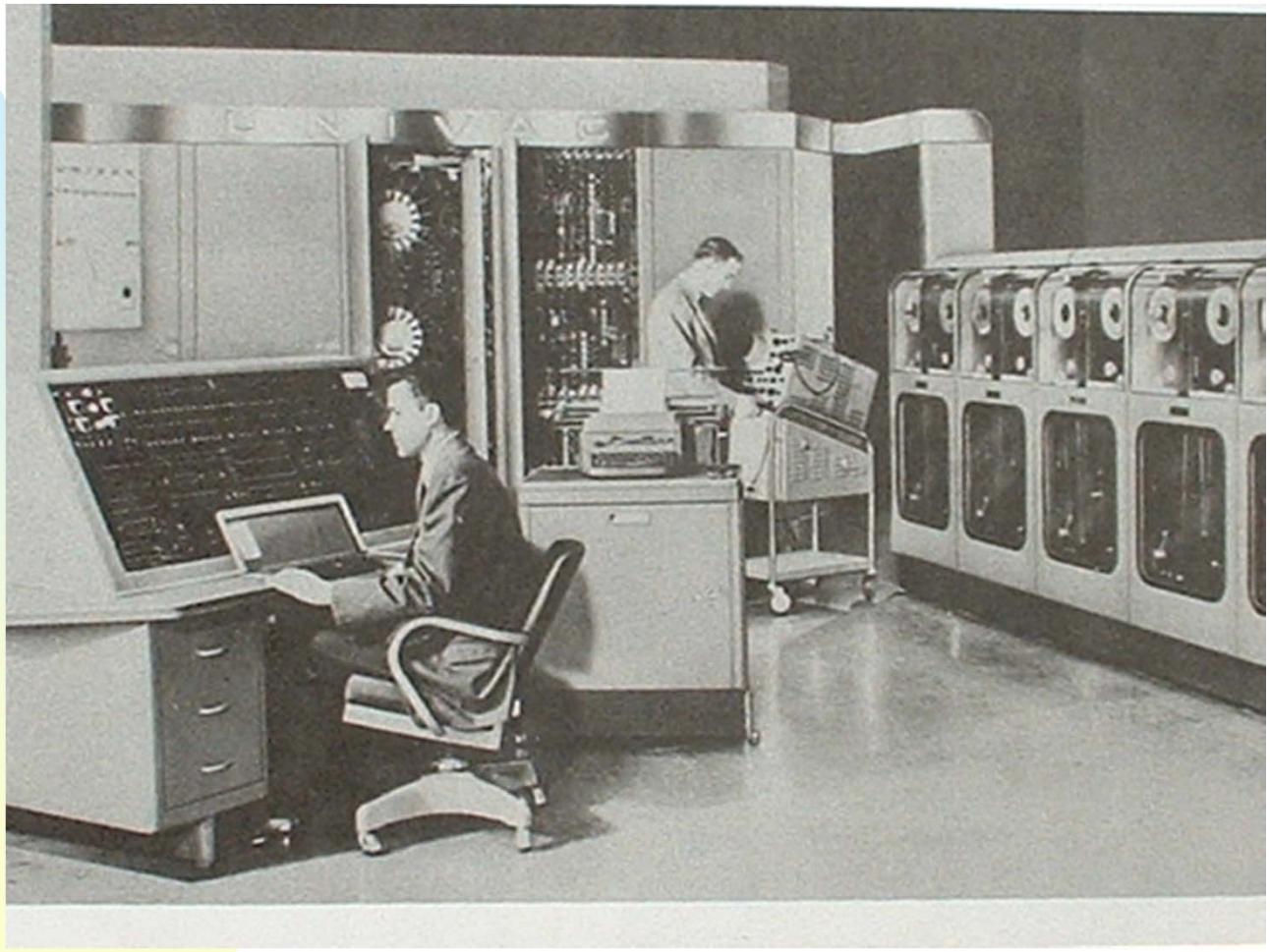


年份	大事
1938	Konrad Zuse 建成了第一台二进制的机电式通用计算机 Z-1
1943	Alan Turing 等建成了一台真空管计算机
1945	J.W.Mauchley 教授等研制成 ENIAC
1947	由 IBM 公司和哈佛大学共同制成自动机电式哈佛 Mark I 计算机
1948	曼彻斯特 Mark I 成为第一台存储程序的数字计算机
1952	EDVAC 研制成功
1952	IBM 制成第一台军用的存储程序电子计算机 IBM 701
1954	Univac1103A 成为第一台商业计算机，采用磁芯存储器
1956	采用晶体管的 Univac 商用计算机开发成功
1960	DEC 公司 11 月研制成 PDP-1，第一台具有显示器和键盘的商用计算机
1961	IBM 公司研制成 7030，号称超级计算机
1962	英国研制成 Atlas 计算机，首次采用虚拟存储器和流水操作
1964	IBM 宣布 System/360
1964	CDC 6600 研制成功，第一台商用超级计算机
1965	DEC 推出 PDP-8，采用晶体管线路
1968	Seymour Cray 设计成功 CDC 7600 超级计算机，40MFLOPS
1971	Intel 推出第一个微处理器芯片 4004
1972	DEC 推出 PDP-11
1975	第一台微型机 Altair8800 研制成功
1976	Cray-1 研制成功，第一台向量结构超级计算机
1977	Tony 和 Commodore 推出商品微型机
1980	Apollo 公司研制成第一台工程工作站
1981	IBM 推出 PC 机
1982	Cray X-MP 推出，将两台 Cray-1 链接在一起
1982	日本启动”第五代”计算机项目
1985	Cray-2 和 Connection Machine 研制成功，性能均达每秒十亿次运算
1989	Cray-3 研制成功，采用砷化镓芯片
1991	Cray Y-MP C90 研制成功，采用 16 个处理机

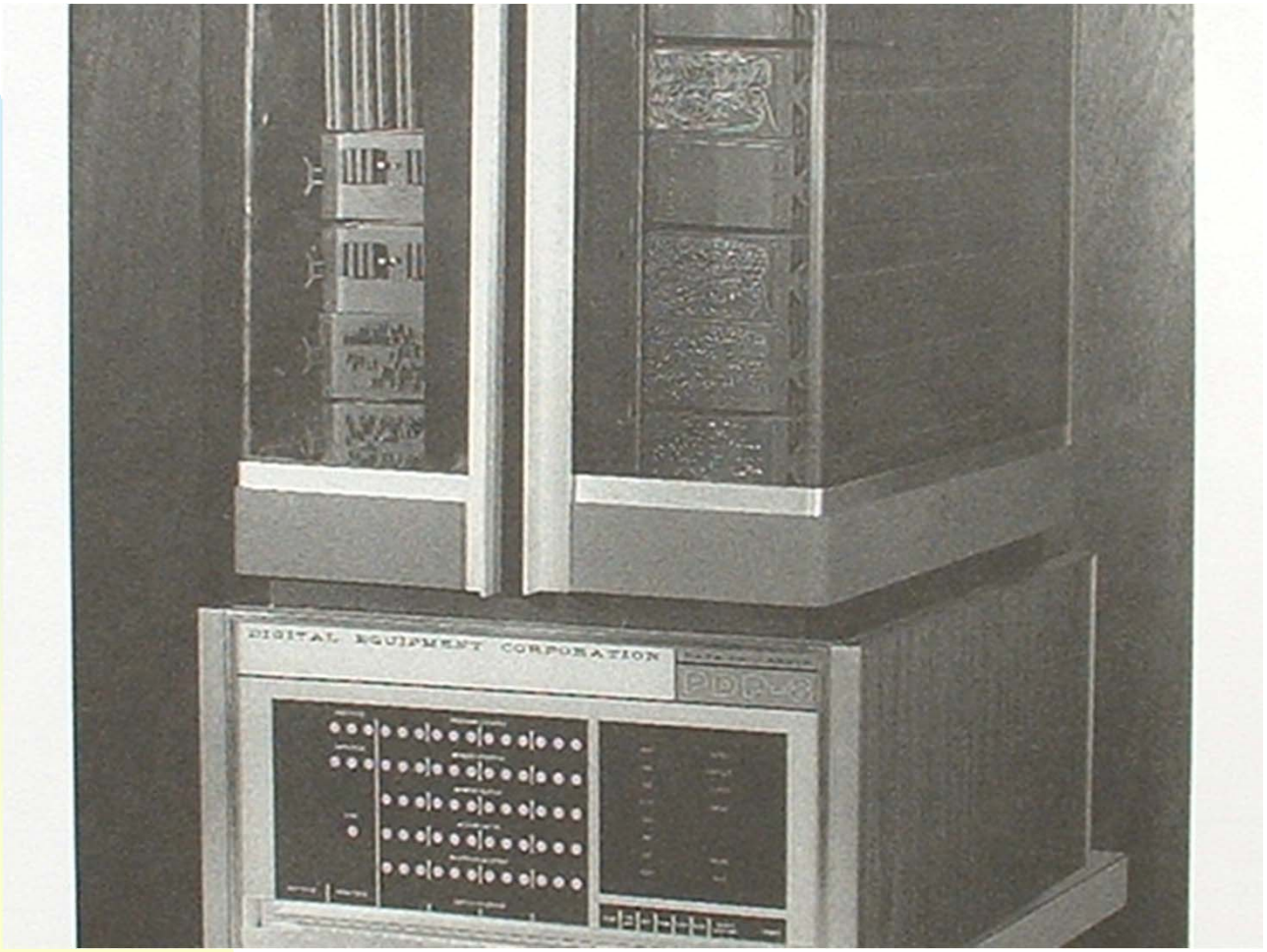
ENIAC



UNIVAC



DEC PDP-8



1.3.2 计算机的分类

- 超级计算机
  - 用于科学计算领域
- 大型计算机
  - 多用户的通用计算机
- 小型计算机
  - 体积小、成本低，通用性强

服务器

- 桌上型计算机->微机->PC机
  - 强大的图形功能,成本低、应用广
- 嵌入式计算机
  - 成为其它设备的一部分

Intel Architecture Processors

Intel Processor	Date of Product Intro-duction	Perform-ance_ in MIPs[1]	Max. CPU Frequency at Intro-duction	No. of Transis-tors on the Die	Main CPU Register Size[2]	Extern. Data Bus Size[2]	Max. Extern. Addr. Space	Caches in CPU Pack-age[3]
8086	1978	0.8	8 MHz	29 K	16	16	1 MB	None
Intel 286	1982	2.7	12.5 MHz	134 K	16	16	16 MB	Note 3
Intel386™ DX	1985	6.0	20 MHz	275 K	32	32	4 GB	Note 3
Intel486™ DX	1989	20	25 MHz	1.2 M	32	32	4 GB	8KB L1
Pentium®	1993	100	60 MHz	3.1 M	32	64	4 GB	16KB L1
Pentium® Pro	1995	440	200 MHz	5.5 M	32	64	64 GB	16KB L1; 256KB or 512KB L2
Pentium II®	1997	466	266	7 M	32	64	64 GB	32KB L1; 256KB or 512KB L2
Pentium® III	1999	1000	500	8.2 M	32 GP 128 SIMD-FP	64	64 GB	32KB L1; 512KB L2

嵌入式系统

- 定义：以应用为中心、以计算机技术为基础、软件硬件可裁剪、适应应用系统对功能、可靠性、成本、体积、功耗严格要求的专用计算机系统。

- 与嵌入式计算机系统相对立的是通用计算机系统，例如**PC**机. 这类计算机的软件和硬件系统功能通常较丰富，可以满足用户的不同的需求。

- 嵌入式系统已经渗透到我们生活中的每个角落，工业、服务业、消费电子，例子：手持的**MP3/MP4**、智能手机、家用电器、变频器、检测仪器等。

嵌入式系统的特点

- 面向特定应用
  - 以通用微处理器/微控制器作为内核
  - 集成其他接口电路和存储器接口
- 成本低
  - **SoC**技术
- 低功耗
- 实时性
  - 强实时性**vs**弱实时性
- 高可靠
  - **Watchdog**技术
- 免维护

1.3.3 计算机的应用领域

- 科学计算
  - 气象学、天文学、量子化学、空气动力学、核物理学、图像学、模式识别、基因工程学、分子生物学、医药学
- 工程计算
  - 工程设计、自动控制和自动测量
- 信息处理
  - 事务处理、信息管理、通信
- 信息电器（嵌入式系统为主）
  - 可视电话、电子书籍、网络游戏、个人数字助理、家用电器控制

嵌入式计算机的应用领域

- 无线通信领域：手机、PDA
- 消费类电子产品：数字媒体播放器、游戏机
- 网络应用：语音及视频处理、数字机顶盒、VoIP
- 成像和安全产品：数码相机、打印机、SIM智能卡
- 工业控制与仪器仪表：
- 其他领域

嵌入式系统产品

Samsung ML5100A

JVC "Pixstar" GC-X1

Diamond Multimedia Rio 600

Alba Bush
Internet TV

3Com
10/100 PCI NIC

Lexmark Z52 Color Jetprinter

Nintendo
Gameboy
Advance

Iomega HipZip

Sony MZ-R90 MiniDisc

Ericsson
R380

HP CapShare

HP Jornado 820

Psion Revo Plus

Nokia 8810

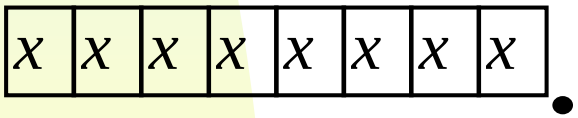
Nokia Mediamaster

2、 数据编码和数据运算

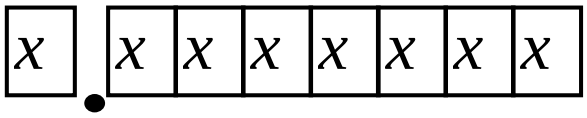
- 定点数的编码与运算
- 浮点数的编码与运算
- 逻辑运算

2.1 定点数的编码和运算

- 定点数：小数点位置固定不变的数
  - 定点整数：小数点定在最低位数的右面
  - 定点小数：小数点固定在最高位数的后面，即纯小数表示



(a) 定点整数



(b) 定点小数

机器数

- 无符号数
  - 半字、字、双倍字、四倍字
- 有符号数
  - 原码、反码、补码、移码
  - 1位符号位

2.1.1 无符号数的编码

- 定点整数

  二进制数值表示：

  $$x = x_0x_1x_2\ldots x_n \qquad x_i=\{0,1\},\ 0\le i\le n$$

  $$x_0 2^n + x_1 2^{n-1} + \ldots + x_{n-1} 2^1 + x_n$$

  数值范围

- $$0\le x\le 2^{n+1}-1$$

- 例如：

  x=010101

  其数值=$2^4+2^2+2^0=21$

定点小数

数值表示

$x = x_0 . x_1 x_2 \ldots x_n$　　$x_0=0, x_i=\{0,1\}, 0 \le i \le n$

$x_1 2^{-1} + \ldots + x_{n-1} 2^{-n+1} + x_n 2^{-n}$

数值范围

- $0 \le x \le 1-2^{-n}$
- 例如：

$x=0.10101$

其数值$=2^{-1}+2^{-3}+2^{-5}=21/32$

其他数制

- 八进制数
  - 记数符号用0到7
  - 计数的方法：“逢八进一”

- 例如，八进制数123_8

  $1\times8^2+2\times8^1+3\times8^0=83$

  $123_8=1010011_2$

其他数制

- 十六进制数
  - 记数符号用0到9以及A到F
  - 计数的方法：“逢十六进一”。

- 例如
- $123_{16}=1\times16^2+2\times16^1+3\times16^0$

  $=256+32+3=291$

其他数制

- 二—十进制数（BCD编码）
  - 用4位二进制编码表示一位十进制数

- 例如:

  123表示成0001 0010 0011

ASCII码

b6b5b4 / b3b2b1b0	000	001	010	011	100	101	110	111
0 0 0 0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	STX	DC2	"	2	B	R	b	r
0 0 1 1	ETX	DC3	#	3	C	S	c	s
0 1 0 0	EOT	DC4	$	4	D	T	d	t
0 1 0 1	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	ACK	SYN	&	6	F	V	f	v
0 1 1 1	BEL	ETB	'	7	G	W	g	w
1 0 0 0	BS	CAN	(	8	H	X	h	x
1 0 0 1	HT	EM	)	9	I	Y	i	y
1 0 1 0	LF	SUB	*	:	J	Z	j	z
1 0 1 1	VT	ESC	+	;	K	[	k	{
1 1 0 0	FF	FS	,	<	L	\	l	|
1 1 0 1	CR	GS	-	=	M	]	m	}
1 1 1 0	SO	RS	.	>	N	^	n	~
1 1 1 1	SI	US	/	?	O	_	o	DEL

2.1.2 有符号数的编码

- 原码
- 反码
- 补码

1. 原码表示法

定义（编码规则）

$$[x]_{原} = \begin{cases} x, & 0 \le x < 2^n \\ 2^n - x = 2^n + |x|, & -2^n < x \le 0 \end{cases}$$

数值（求值方法）

$$x = (-1)^{x_0}（x_1 2^{n-1} + \cdots x_{n-1}2 + x_n）$$

数值范围

$$-2^n + 1 \leqslant x \leqslant 2^n - 1$$

$$[x]_{原} = \begin{cases} x, & 0 \le x < 1 \\ 1 - x = 1 + |x|, & -1 < x \le 0 \end{cases}$$

$$x = (-1)^{x_0}（x_1 2^{-1} + \cdots x_{n-1}2^{-(n-1)} + x_n 2^{-n}）$$

$$-1 + 2^{-n} \leqslant x \leqslant 1 - 2^{-n}$$

特点：简便编码方法（加符号位）

例：$[3]_{原} = 00000011$

$[-3]_{原} = 10000011$

1. 原码表示法

- 零有两种表示方式
  - **00000000**
  - **10000000**

例 设x=101010，y= -101010，求$[x]_{原}$和$[y]_{原}$

解：　$[x]_{原}$=00101010

　　$[y]_{原}$=10101010

例 设x=0.1010，y= -0.1010，求$[x]_{原}$和$[y]_{原}$

解：　$[x]_{原}$=0.101010

　　$[y]_{原}$=1.101010

2. 反码表示法

$$[x]_{反} = \begin{cases} x, & 0 \le x < 2^n \\ 2^{n+1} - 1 + x, & -2^n < x \le 0 \end{cases}$$

$$x = -x_0 (2^n - 1) + x_1 2^{n-1} + \cdots + x_{n-1} 2 + x_n$$

$$-2^n + 1 \leqslant x \leqslant 2^n - 1$$

例：$[1010]_{反} = 01010$

$[-1010]_{反} = 10101$

1‘s complement coding

2. 反码表示法

- 编码方法
  - 正数的反码与原码相同
  - 负数的反码是将二进制位按位取反
- 数值范围

$$-2^n+1 \leq x \leq 2^n-1$$

  - 定点小数

$$-1+2^{-n} \leq x \leq 1-2^{-n}$$

- 零有两个编码：000…0和111…1

3. 补码表示法

编码规则

$$[x]_{补} = \begin{cases} x, & 0 \le x < 2^n \\ 2^{n+1} + x, & -2^n \le x < 0 \end{cases}$$

求值方法

$$x = -x_0 2^n + x_1 2^{n-1} + \cdots + x_{n-1} 2 + x_n$$

数值范围

$$-2^n \leqslant x \leqslant 2^n - 1$$

特点：便于运算

例： $[3]_{补} = 00000011$

$[-3]_{补} = 11111101$

2‘s complement coding

3. 补码表示法

- 方法1
  - 正数：直接取其原来的二进制码（加符号位0）
  - 负数：对其二进制码按位取反之后再在最低位加1

    例： $[010101]_{补}$= 00010101

    $[-010101]_{补}$= 11101010+1 = 11101011
- 方法2
  - 正数：直接取其原来的二进制码
  - 负数：从二进制码的最低位开始，对遇到的0和第一个1取其原来的二进制编码，从第一个1以后开始直到最高位均取其相反编码。

    例：$[101010]_{补}$= 00101010

    $[-101010]_{补}$= 11010110

定点小数的补码编码

- $x = x_0.x_1 \cdots x_n$

$$[x]_{补} = \begin{cases} x, & 0 \le x < 1 \\ 2 + x, & -1 \le x < 0 \end{cases}$$

- 数值范围：

$$-1 \le x \le 1 - 2^{-n}$$

例 设x=0.101010，y=-0.101010， 求$[x]_{补}$和$[y]_{补}$。

解：

$[x]_{补}$=0.101010

$[y]_{补}$=1.010110

补码求值的方法

- 公式法

  $x = -x_0 2^n + x_1 2^{n-1} + \ldots + x_{n-1} 2 + x_n$

  例如：10000100的真值为−128+4=−124

- 求补法

  $[x]_{补}$与$[-x]_{补}$的关系

  例: $[x]_{补}$=11111100

  $[-x]_{补}$=00000100

  $-x=4$

  $x=-4$

模4补码（双符号位）

$$[x]_{补} = \begin{cases} x, & 0 \le x < 2^n \\ 2^{n+2} + x, & -2^n \le x < 0 \end{cases}$$

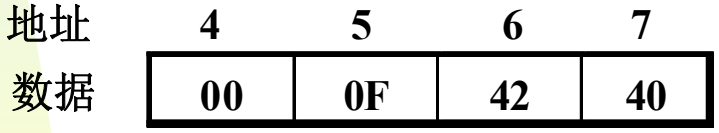
$$[x]_{补} = \begin{cases} x, & 0 \le x < 1 \\ 4 + x, & -1 \le x < 0 \end{cases}$$

例： $[0.1010110]_{补}=00.1010110$

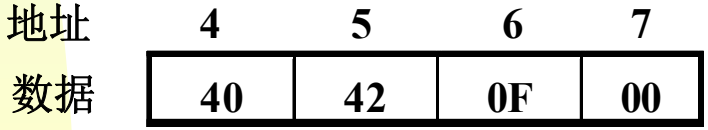
$[-0.1010110]_{补}=11.0101010$

2.1.3 数据的存储与访问

- 数据类型
  - 整型数、单精度和双精度浮点数、字符型
- 数据长度
  - 单字节、双字节、字、双字、四倍字
- 字节存储顺序
  - 大数端(big Endian)和小数端(little Endian)。



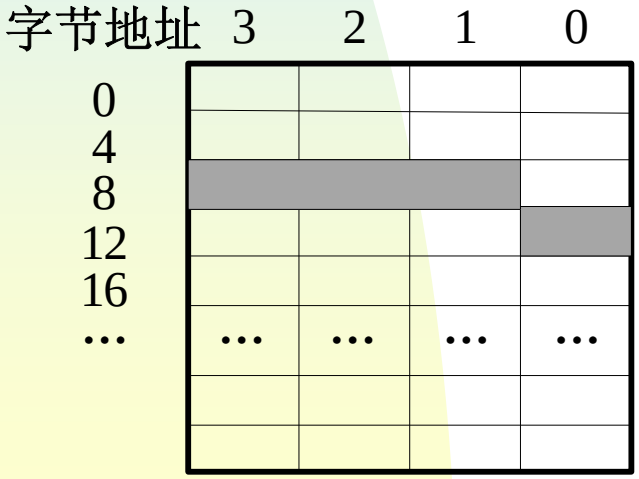
(a) 大数端存储方式



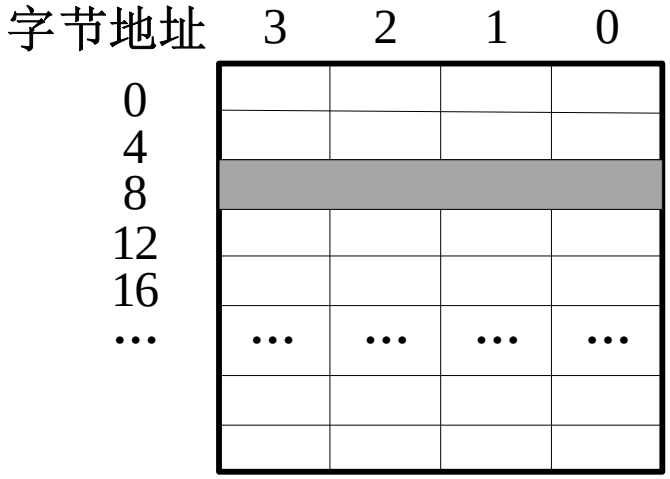
(b) 小数端存储方式

数据的存储方式

- 对齐的方式
- 非对齐的方式。



(a) 字不对齐



(b) 字对齐

2.1.4 定点数的加减运算

一、补码加法

- 根据补码加法公式，补码可以直接相加。

$$[x]_{补} + [y]_{补} = [x+y]_{补} \quad (\text{mod } 2)$$

二、补码减法

- 根据补码减法公式，补码可以直接相减。

$$[x-y]_{补} = [x]_{补} - [y]_{补} = [x]_{补} + [-y]_{补} \quad (\text{mod } 2)$$

定点数的加减运算例子

例2-4 $x=0.1010$，$y=-0.0011$，用补码的加法求$x+y$。

解：$[x]_{补}=0.1010$，$[y]_{补}=1.1101$

$[x]_{补}+[y]_{补}=0.1010+1.1101=0.0111$

$x+y=0.0111$

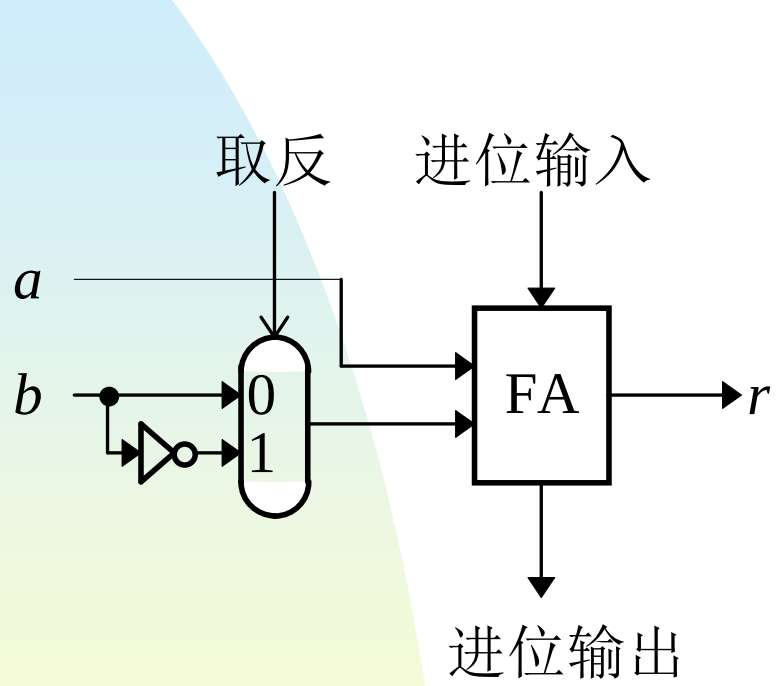
例2-5 $x=0.1001$，$y=-0.0011$，用补码的减法求$x-y$。

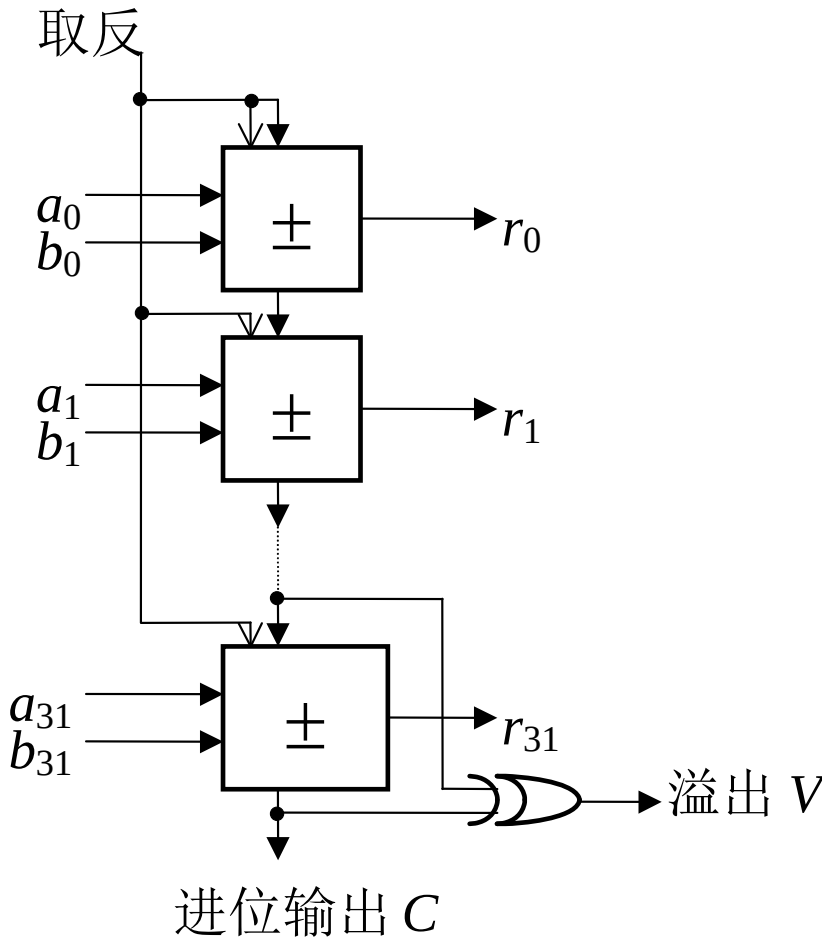
解：$[x]_{补}=0.1001$，$[y]_{补}=1.1101$，$[-y]_{补}=0.0011$

$[x]_{补}-[y]_{补}=[x]_{补}+[-y]_{补}=0.1001+0.0011=0.1100$

$x-y=0.1100$

加减运算电路





数据溢出及其检测

方法一——符号位判断

$$V = x_0\ y_0\ \overline{z_0} + \overline{x_0}\ \overline{y_0}\ z_0$$

- x_0、y_0为加数与被加数的符号位
- z_0为和的符号位。
- V=1，则溢出。
- 含义：
  - 如果两个负数相加得到正数，则溢出
  - 如果两个正数相加得到负数，则溢出

方法二——双符号位判断

$$V = z_0' \overline{z_0} + \overline{z'_0}\, z_0 = z_0' \oplus z_0$$

	00.xxxx		11.xxxx
+	00.xxxx	或 +	11_1.xxxx
	00.xxxx		11.xxxx

- **00**为符号**+**、**11**为符号**-**
- z_0'、z_0为和的双符号位
- **V=1**，则溢出。
- 含义：
  - 如果和的双符号位**01**，则溢出
  - 如果和的双符号位**10**，则溢出

例 2-6 设 $x = +1100$，$y = +1000$，求 6 位双符号位补码之和$[x+y]_{补}$。

解： $[x]_{补} = 001100$，$[y]_{补} = 001000$

$$\begin{array}{r} 001100 \\ +\ 001000 \\ \hline 010100 \end{array}$$

$[x+y]_{补} = 010100$，其中两个符号位出现 01，表示已溢出。

例 2-7 设 $x = -1100$，$y = -1000$，求 6 位双符号位补码之和 $[x+y]_{补}$。

解： $[x]_{补} = 110100$，$[y]_{补} = 111000$

$$\begin{array}{r} 110100 \\ +\ 111000 \\ \hline 101100 \end{array}$$

$[x+y]_{补}=101100$，其中两个符号位出现 10，表示已溢出。

方法三——判断最高位和次高位的进位

$$V = C_0 \overline{C_1} + \overline{C_0} C_1$$

- C_0：相加后最高位（符号位）的进位位
- C_1：相加后次高位（最高数值位）的进位位
- V=1，则溢出。
- 含义：
  - 如果最高位进位而次高位无进位，则溢出
  - 如果次高位进位而最高位无进位，则溢出

避免数据的溢出的方法

- 增加数据的表示位数
  - 例如数据6
    - 在8位的计算机中表示为00000110，
    - 在16位计算机中表示为0000000000000110。
  - 例如用补码表示-2时
    - 在8位计算机中是1111 1110，
    - 在16位计算机中是1111 1111 1111 1110。
- 符号扩展

数据溢出的概念与数据取模时的丢弃

- 数据运算中最高位的进位被丢弃并不一定是溢出
- 例如，两个负数的补码相加，

  设x= -0110，即-6_{10}；y= -0101，即-5_{10}。

  则$[x]_{补}$=11010，$[y]_{补}$=11011。

  $[x+y]_{补}=10101 \pmod{2^5}$，即$-11_{10}$

  运算结果正确，没有发生溢出

2.1.5 定点数的乘除运算

二进制乘法

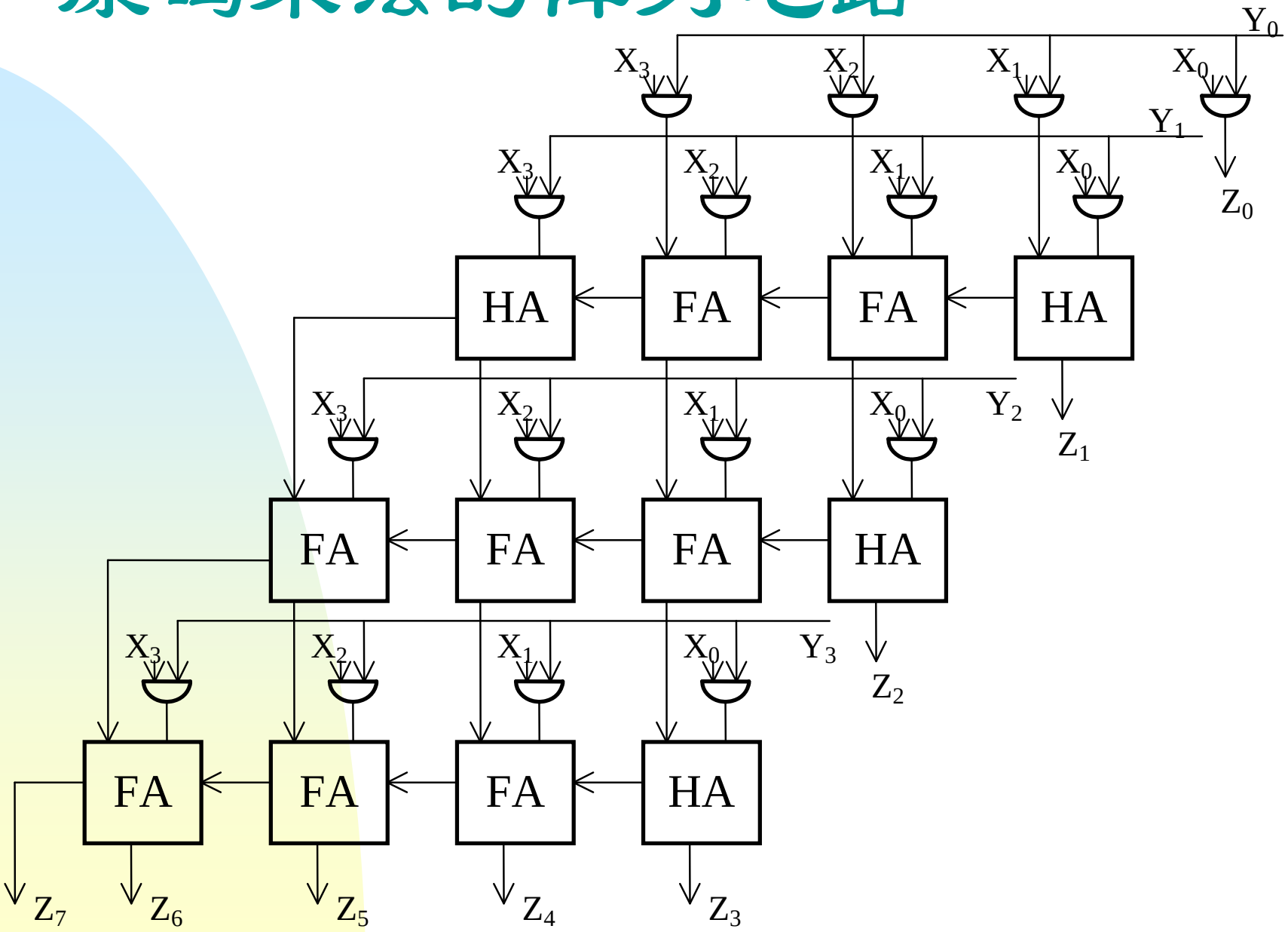
$$
\begin{array}{r}
0010 \\
\times\quad 0011 \\
\hline
0010 \\
0010 \\
0000 \\
0000 \\
\hline
0000110
\end{array}
$$

$$Z = X \times Y = \left(\sum_{i=0}^{M-1} x_i 2^i\right)\left(\sum_{j=0}^{N-1} y_j 2^j\right) = \sum_{i=0}^{M-1}\left(\sum_{j=0}^{N-1} x_i y_j 2^{i+j}\right) = \sum_{k=0}^{M+N-1} z_k 2^k$$

原码乘法

- 将符号位与数值位分开进行运算
- 运算结果的数值部分是乘数和被乘数数值位的乘积
- 运算结果的符号位是乘数和被乘数符号位的异或

原码乘法的阵列电路


Y0
X3
X2
X1
X0
Y1
Z0
HA
FA
FA
HA
Y2
Z1
FA
FA
FA
HA
Y3
Z2
FA
FA
FA
HA
Z7
Z6
Z5
Z4
Z3

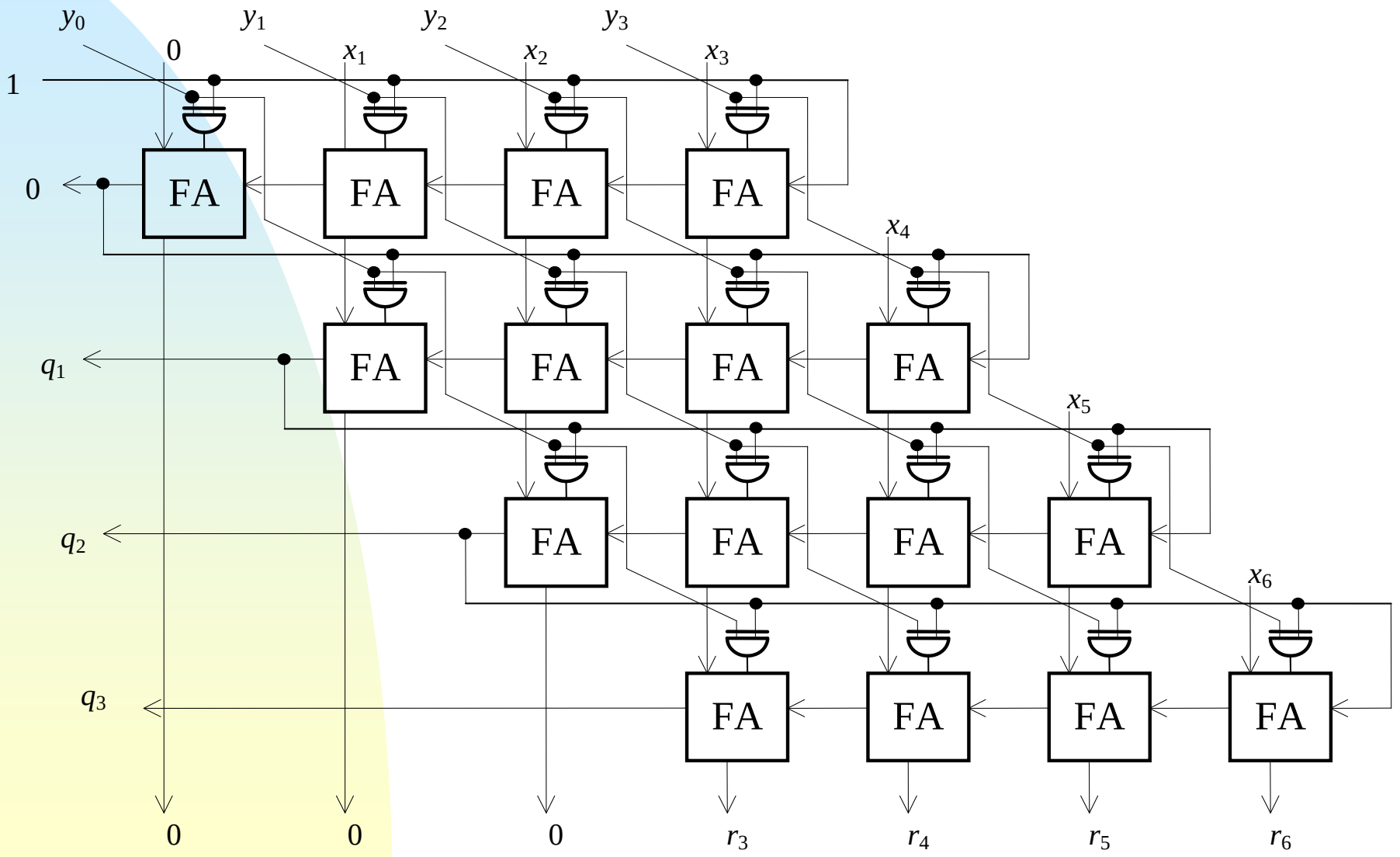
二进制除法

$$
\begin{array}{r}
1010 \\
1101\overline{)10001001} \\
1101 \\
\hline
0010000 \\
1101 \\
\hline
000111
\end{array}
$$

比较被除数（或部分余数）和除数大小的方法

- 恢复余数法
  - 通过减法实现，不够减时再用加法恢复原来的部分余数。

实现加减交替法的电路



2.2 浮点数的编码和运算

构成：阶码 E，尾数 M，符号位 S，基数 R

$$N = (-1)^S \times M \times R^E$$

- **E的编码：** 移码或补码
  - **移码：** 将数据的二进制编码加上一个常数(通常是2^n)后的二进制代码作为该数据的编码。
  - **移码的特点：** 保持了数据原有的大小顺序，便于数据比较。
- **S与M的编码：** 原码或补码
- **R进制的含义：** 多个二进制位构成一组，代表一个R进制位

浮点数的编码

- 规格化：
  - $0.1_2 = 0.1 \times 2^0 = 0.01 \times 2^1$
  - 为了在尾数中表示最多的有效数据位
  - 为了数据表示的唯一性。
- 机器零：
  - 全部为0，特殊的数据编码

规格化的编码

- 原码
  - 数据位的最高位为1　　1xxxxxx
- 补码
  - 小数点前后两位互不相同。
  - 例如，尾数0.1010和1.0101是规格化的，而尾数0.0101和1.1010是非规格化的。
- 基数为2的浮点数规格化后，其尾数的绝对值在1/2到1之间
- 基数为*R*的浮点数规格化后，其尾数的绝对值在1/*R*到1之间。

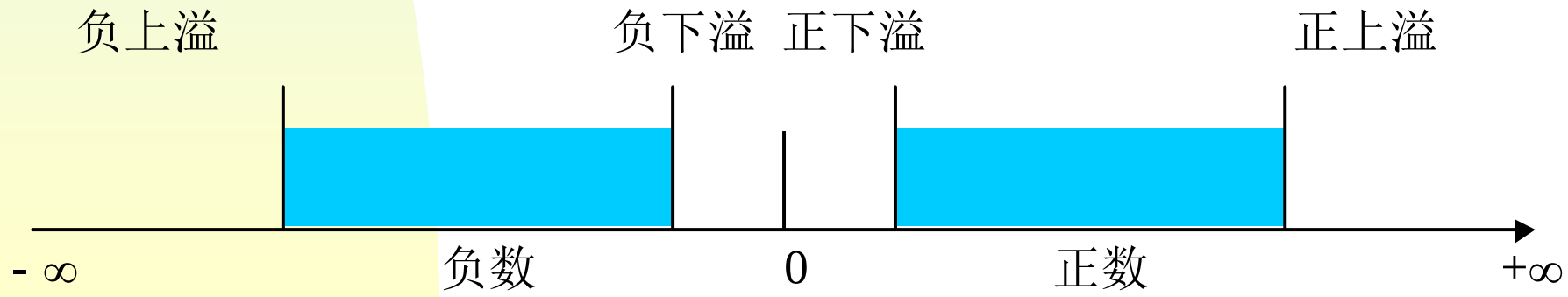
例2-11 对数据123_{10}作规格化浮点数的编码，假定1位符号位，基数为2，阶码5位，采用移码，尾数10位，采用补码。

解：先将数据表示成$M\times R^E$的形式

- $123_{10} = 1111011 = 0.1111011000\times 2^7$
- 分别对M和E按照题目要求进行编码，阶码采用5位移码，尾数采用10位补码。
- $[7]_{移}=10000+00111 = 10111$
- $[0.1111011000]_{补}=0.1111011000$
- 将编码的结果按浮点数格式表示出来。因为符号位为0，所以浮点格式为：
- 0 10111 1111011000

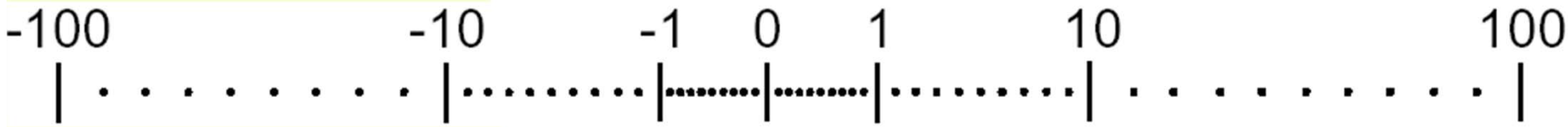
浮点数的表示范围

- 浮点数的溢出表现为阶码的溢出
- 浮点数的上溢(overflow)
  - 数据太大，以至于大于阶码所能表示的数据
- 浮点数的下溢(underflow)
  - 数据太小，以至于小于阶码所能表示的数值时，



浮点数的表示范围与表示精度

- 浮点数表示法可以扩大数值表示的范围
- 浮点数表示法未增加表示数值的个数
- 绝对值越大，浮点数分布越稀疏
- 阶码位数越多，数据表示的范围就越大
- 尾数位数越多，数据表示的精度越高



浮点数的表示范围与表示精度

例：浮点数基数为2，有1位符号位、4位阶码和4位尾数代码，阶码采用移码表示，求数值表示范围及可表示的数据个数。

解：

最小规格化正尾数：

最大规格化正尾数：

最大阶码：

最小阶码：

最小正值：

最大正值：

尾数个数：

数值个数：

浮点数的表示范围与表示精度

例：浮点数基数为8，有1位符号位、4位阶码和6位二进制尾数代码，阶码采用移码表示，求数值表示范围及可表示的数据个数。

解：

最小规格化正尾数：

最大规格化正尾数：

最大阶码：

最小阶码：

最小正值：

最大正值：

尾数个数：

数值个数：

浮点数标准(IEEE754)

- 三种格式：短实数、长实数、临时实数
- 无定义数据(NaN)：
  - 发信号的NaN，不发信号的NaN
- 无穷大：+INF, -INF
- 单精度规格化数：

$$(-1)^s \times 1.f \times 2^{e-127}$$

- 非规格化数：

$$(-1)^s \times 0.f \times 2^{e-126}$$

浮点数标准(IEEE754)

表 2-3　IEEE 单精度浮点数编码格式

符号位	阶码	尾数	表示
0/1	255	非 0	S
0/1	255	非 0	Q
0	255	0	+INF
1	255	0	-INF
0/1	1-254	f	$(-1)^s \times 1.f \times 2^{e-127}$
0/1	0	f(非 0)	$(-1)^s \times 0.f \times 2^{e-126}$
0/1	0	0	+0/-0

IEEE754浮点数的范围

格式	最小值	最大值
单精度	E=1, M=0, $1.0\times2^{1-127}=2^{-126}$	E=254, f=.1111…, $1.111\cdots1\times2^{254-127}$ $=2^{127}\times(2-2^{-23})$
双精度	E=1, M=0, $1.0\times2^{1-1023}=2^{-1022}$	E=2046, f=.1111…, $1.111\cdots1\times2^{2046-1023}$ $=2^{1023}\times(2-2^{-52})$

浮点数标准

例3-7 将十进制数-0.75表示成单精度的IEEE754标准代码。

解：$-0.75=-3/4=-0.11_2=-1.1\times2^{-1}$

$=(-1)^1\times(1+0.1000\ 0000\ 0000\ 0000\ 0000\ 000)\times2^{-1}$

$=(-1)^1\times(1+0.1000\ 0000\ 0000\ 0000\ 0000\ 000)\times2^{126-127}$

$s=1$，$e=126_{10}=01111110_2$，$f=1000\ \ldots\ 000$。

1 01111110 10000000000000000000000

浮点数标准

例3-8 将如下IEEE754 单精度浮点数用十进制数表示：

1 10000001 01000000000000000000000

解：$s=1$，$e=129$，$f = 1/4 = 0.25$，

$(-1)^1 \times (1+0.25) \times 2^{129-127}$

$= -1 \times 1.25 \times 2^2$

$= -1.25 \times 4$

$= -5.0$

2.2.2 浮点数的运算

——浮点数加减法

- 五个基本步骤
  - 对阶
  - 尾数加减
  - 规格化（左规，右规）
  - 舍入（截去、0舍1入、冯•诺依曼舍入）
  - 检查溢出

例2-13 两浮点数$x = 2^{01} \times 0.1101$，$y = 2^{11} \times (-0.1010)$。假设尾数在计算机中以补码表示，可存储4位尾数，2位保护位，阶码以原码表示，求$x+y$。

解：将x,y转换成浮点数据格式

$[x]_{浮}$ = 00 01, 00.1101

$[y]_{浮}$ = 00 11, 11.0110

步骤1：对阶，阶差为11-01=10，即2，因此将x的尾数右移两位，得

$[x]_{浮}$ = 00 11, 00.001101

步骤2：对尾数求和，得:

$[x+y]_{浮}$ = 00 11, 11.100101

步骤3：由于符号位和第一位数相等，不是规格化数，向左规格化，得

$[x+y]_{浮}$ = 00 10, 11.001010

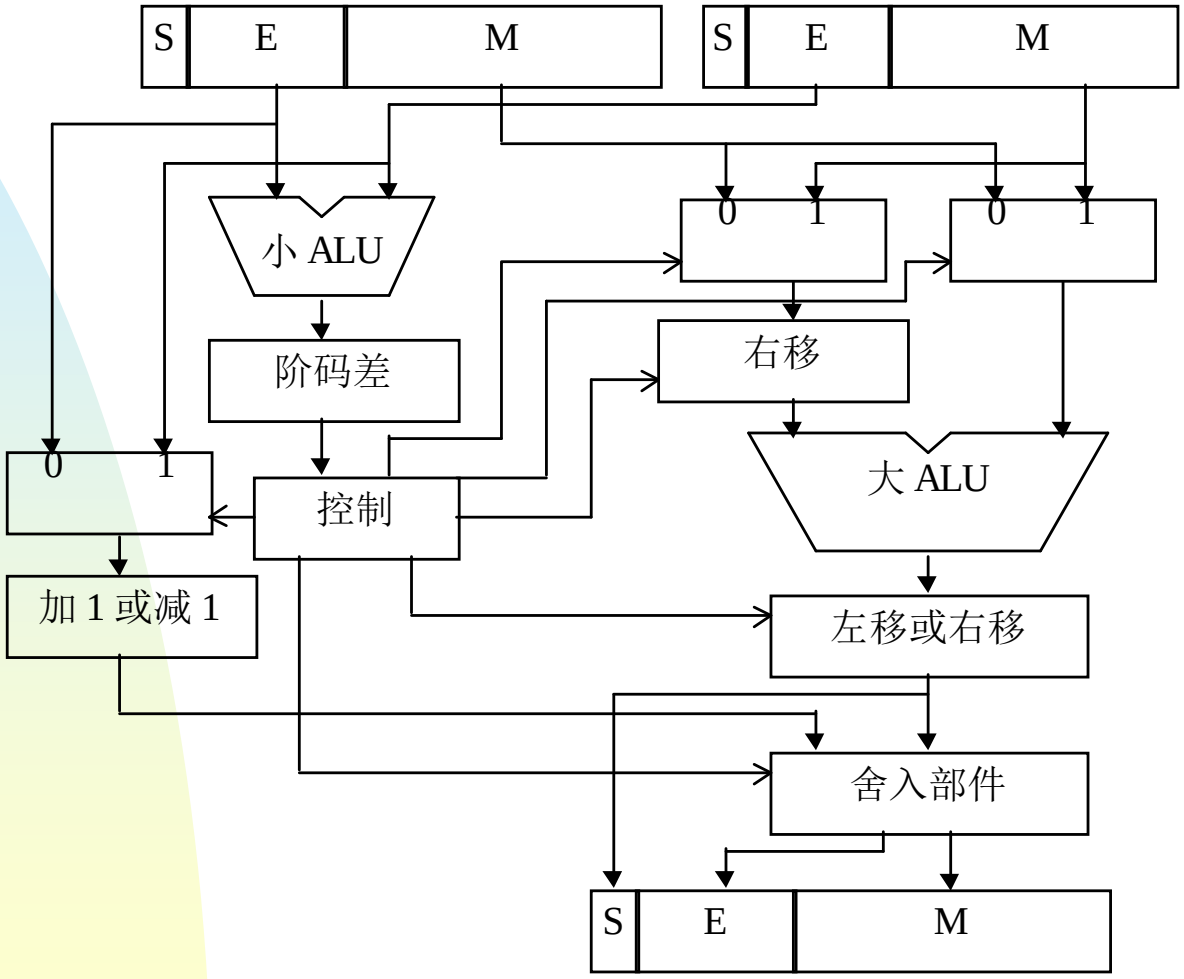
步骤4：截去。

$[x+y]_{浮}$ = 00 10, 11.0010

步骤5: 数据无溢出，因此结果为

$x+y = 2^{10} \times (-0.1110)$

浮点运算电路



浮点数乘除法

- 五个基本步骤
  - 阶码加减
  - 尾数乘除
  - 规格化
  - 舍入
  - 检查溢出

2.3 逻辑运算

- 按位运算
  - 分别考虑每一位信息
- 基本的逻辑运算
  - 逻辑与、逻辑或、逻辑非
  - 移位操作

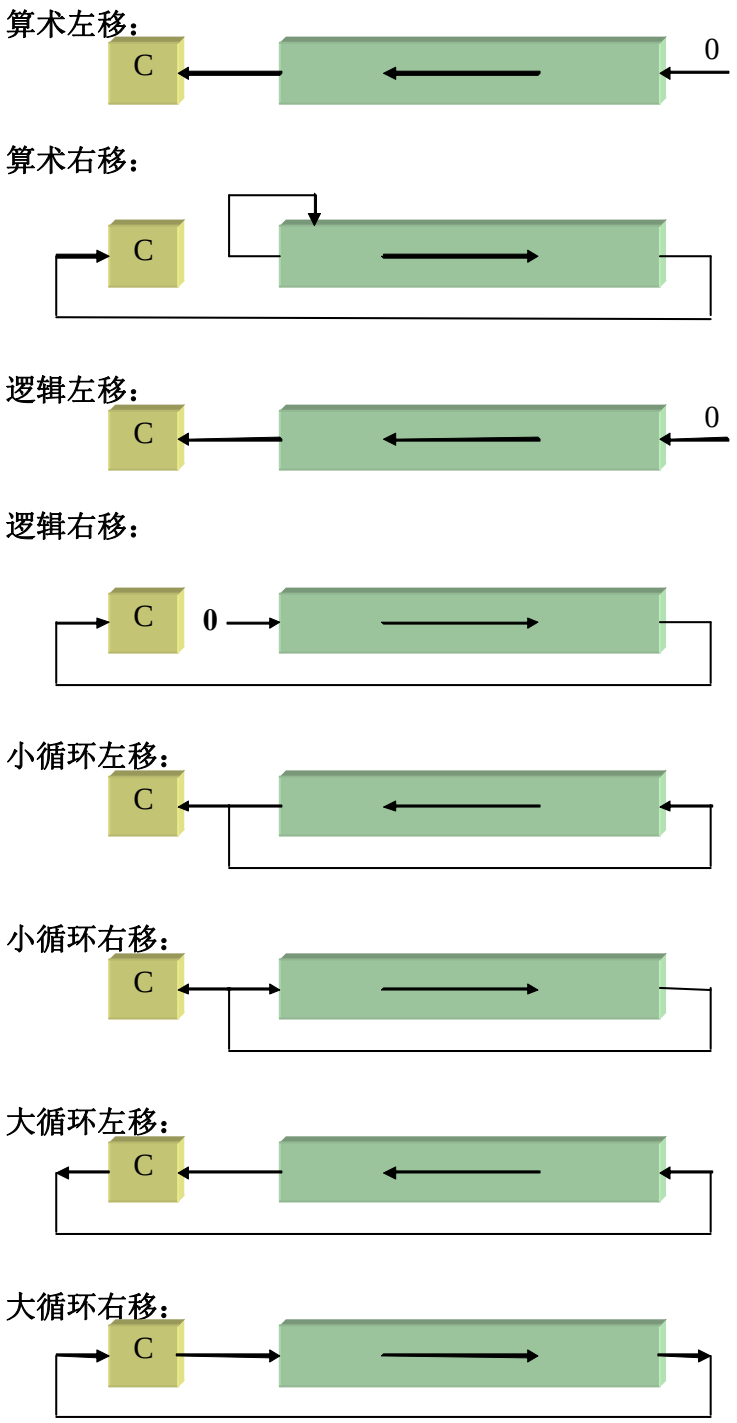
逻辑运算的例子

x=001101，则 $\overline{x}$ =110010

- **x=10100001，y=10011011，**
  则
  x OR y=10111011
  x AND y=10000001

移位运算

- 算术移位
- 逻辑移位
- 左移
- 右移
- 循环移位
  - 小循环
  - 大循环



例2-15 已知一个8位寄存器中的数据为11001011，求以下移位运算后寄存器的结果及标志位C的值。

(1) 算术左移，

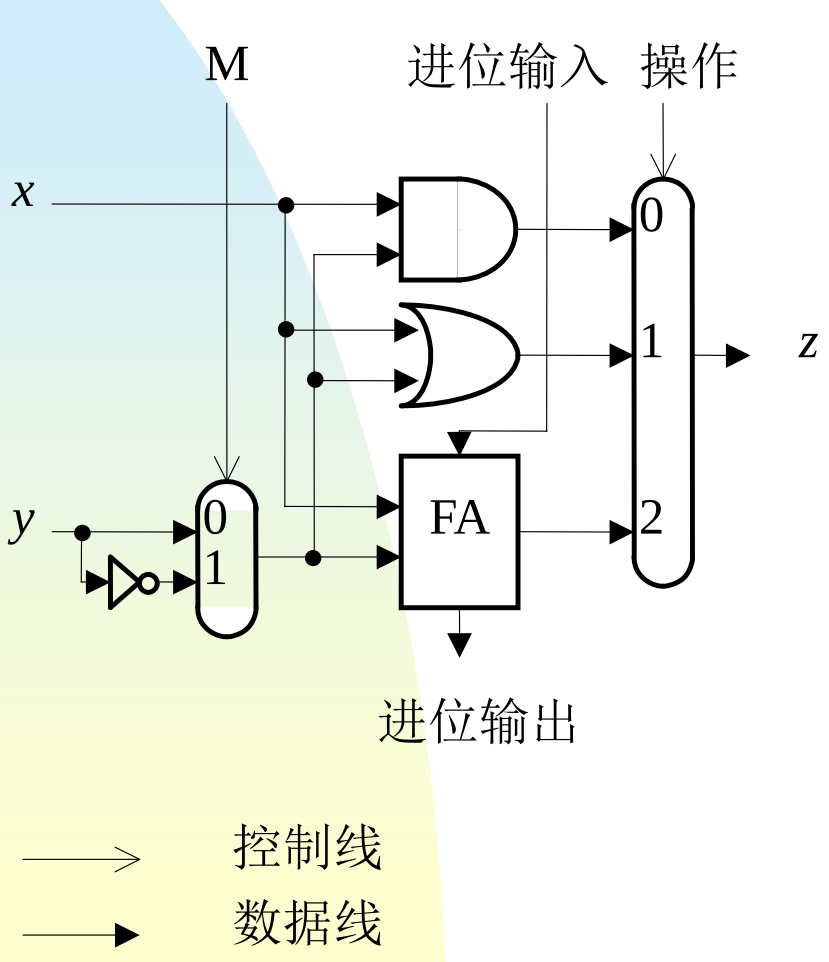
(2) 算术右移，

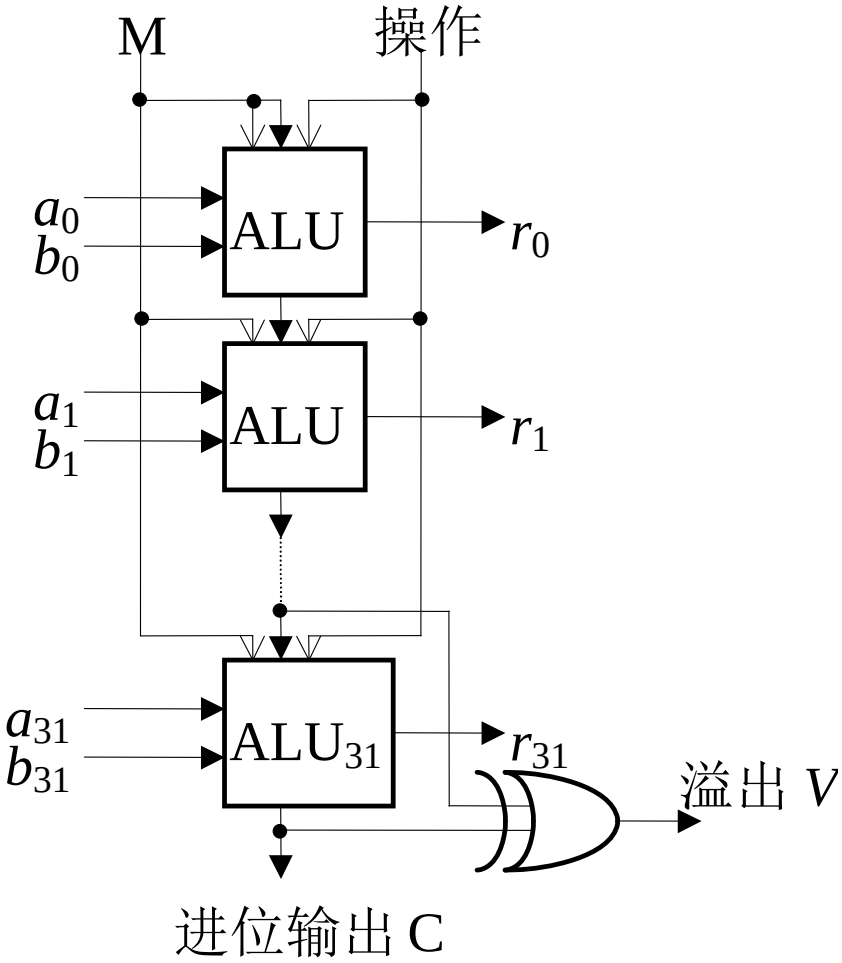
(3) 循环右移。

解：

- (1) 10010110，C=1。
- (2) 11100101，C=1。
- (3) 11100101，C=1。

算术逻辑运算单元


M
进位输入
操作
x
y
0
1
FA
0
1
2
z
进位输出
控制线
数据线


M
操作
a_0
b_0
ALU
r_0
a_1
b_1
ALU
r_1
a_{31}
b_{31}
ALU_{31}
r_{31}
溢出 V
进位输出 C

移位电路