



第1章 计算机基础



1.1 计算机中的数制 (p.11)

解决微型机领域中数的不同表示方法

一. 常用计数制

1. 十进制数：编程时使用(D)
2. 二进制数：计算机内部信息存储，运算，
输入/输出都是二进制数(B)



二进制数特点：

$$\begin{array}{cccccc} (& \boxed{1} & \boxed{1} & \boxed{0} & \boxed{1} & . & \boxed{1} & \boxed{1} &)_2 \\ \text{权:} & 2^3 & 2^2 & 2^1 & 2^0 & . & 2^{-1} & 2^{-2} & \end{array}$$

特点：

- 每位代码非0即1
- 高位权是低位权的2倍
- 加减运算法则：逢二进一，借一当二

$$\begin{array}{r} 101 \\ + 111 \\ \hline 1100 \end{array}$$

$$\begin{array}{r} 1101 \\ - 1010 \\ \hline 0011 \end{array}$$



3. 十六进制数：

- 人们最常用的是十进制，但在计算机中为了物理实现的方便，采用的是二进制。
- 人们为了书写阅读方便，又常常采用十六进制数来表示二进制数。
- 十六进制的基数是16，权值为 16^0 、 16^1 、...，数码有0、1、...、9、A、B、C、D、E、F。
- 十六进制用H表示，二进制数用B表示



➤十六进制与二进制的关系：

每4位二进制数用1位十六进制数来表示

4位二进制数	等值的一位十六进制数	4位二进制数	等值的一位十六进制数
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F



所以, $(10, 1001, 1010, 1111)_2 = (29AF)_{16}$

十六进制数特点:

权: $(\begin{array}{|c|} \hline 2 \\ \hline 16^3 \\ \hline \end{array} \begin{array}{|c|} \hline 9 \\ \hline 16^2 \\ \hline \end{array} \begin{array}{|c|} \hline A \\ \hline 16^1 \\ \hline \end{array} \begin{array}{|c|} \hline F \\ \hline 16^0 \\ \hline \end{array})_{16}$

- 特点:
- 每位代码0~9, A ~ F
 - 高位权是低位权的16倍
 - 加减运算法则: 逢十六进一, 借一当十六

$$\begin{array}{r} (39)_{16} \\ + (7A)_{16} \\ \hline (B3)_{16} \end{array}$$

$$\begin{array}{r} (45)_{16} \\ - (26)_{16} \\ \hline (1F)_{16} \end{array}$$



4. BCD码 (p.16):

- 计算机中采用二进制，但二进制书写、阅读不便，所以在输入输出时人们仍习惯使用十进制。
- 采用二进制数对每一位十进制数字进行编码来表示一个十进制数，这种数叫做BCD码。
- BCD码有多种形式，最常用的是8421BCD码，它是用4位二进制数对十进制数的每一位进行编码，这4位二进制码的值就是被编码的一位十进制数的值。



四位二进制数	等值的一位BCD码数	等值的一位十进制数
0000	0000	0
0001	0001	1
0010	0010	2
0011	0011	3
0100	0100	4



0101	0101	5
0110	0110	6
0111	0111	7
1000	1000	8
1001	1001	9
1010	非法BCD码	
1011		
1100		
1101		
1110		
1111		



- **BCD码在计算机中的存储分为紧凑型和非紧凑型两种：**

紧凑型BCD码：

$$(37)_D = 0011, 0111B$$

非紧凑型BCD码：

$$(37)_D = 0000, 0011B$$
$$0000, 0111B$$



二. 数制转换 (p.10)

1. 二、十六进制数→十进制数

算法: 每位的代码和该位的权值相乘, 再求累加和

如: $(1101.11)_2 = (\quad ? \quad)_{10}$

$$\begin{aligned} \text{解: } & 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} \\ &= 8 + 4 + 0 + 1 + 0.5 + 0.25 \\ &= (13.75)_{10} \end{aligned}$$

如: $(29AF)_{16} = (\quad ? \quad)_{10}$

$$\begin{aligned} \text{解: } & 2 \times 16^3 + 9 \times 16^2 + 10 \times 16^1 + 15 \times 16^0 \\ &= 8192 + 2304 + 160 + 15 \\ &= (10671)_{10} \end{aligned}$$



2. 二进制数→十六进制数

四位二进制数为一组，每组用等值的十六进制代换

如： $(101011.11)_2 = (10, 1011.1100)_2 = (2B.C)_{16}$

3. 十六进制数→二进制数

一位十六进制数用等值的四位二进制数代换

如： $(17E.58)_{16} = (0001, 0111, 1110.0101, 1000)_2$


4. 十进制数→二进制数

① 十进制整数→二进制数

算法：除2取整，直到商为零，倒排余数



2		11	余1
2		5	余1
2		2	余0
2		1	余1
		0		




所以, $(11)_{10} = (1011)_2$



② 十进制数纯小数→二进制数

算法：小数部分乘2取整数部分，直到乘积的小数部分为0时止，顺排整数部分



0.	8125
	2
<hr/>	
1.	625
	2
<hr/>	
1.	25
	2
<hr/>	
0.	5
	2
<hr/>	
1.	0

所以， $(0.8125)_{10} = (0.1101)_2$



③ 十进制带小数 \rightarrow 二进制数
整数、纯小数分别计算, 再合并

$$\therefore (11.8125)_{10} = (1011.1101)_2$$

不同数制的数比大小, 可看它等值的十进制数谁大。

例: 设 $X = (01010110)_2$

$$Y = (5A)_{16}$$

问: X, Y 谁大?

解: 转换成同一数制比, 等值的十进制数谁最大。

$$\because X = (01010110)_2 = 2^6 + 2^4 + 2^2 + 2^1 = (86)_{10}$$

$$Y = (5A)_{16} = 5 \times 16^1 + 10 \times 16^0 = (90)_{10}$$

$\therefore Y$ 大



1.2 计算机中数据的编码

解决不同信息在计算机中的具体表示

计算机处理的信息 { 数值数据
非数值数据

数值数据： 有符号数、无符号数

非数值数据： 字符、图像



一.字符的编码——ASCII码 (p.17)

- 在计算机中除了数值之外，有一类非常重要的数据，那就是字符，计算机常用的输入 / 输出设备有键盘、显示器、打印机，它们处理的数都是人熟悉的字符，有英文的大小写字母，数字符号(0, 1, ..., 9)以及其他常用符号(如：%、+等)。
- 在计算机中，这些符号都是用二进制编码的形式表示，每一个字符被赋予一个惟一固定的二进制编码。目前，一般都是采用美国标准信息交换码(ASCII)，它使用七位二进制编码来表示一个符号。由于用七位码来表示一个符号，故该编码方案中共有128个符号($2^7=128$)。



如：键入“1”，实际写入键盘存储区的是31H
即 00110001B

键入“A”，实际写入键盘存储区的是41H
即 01000001B

又如：欲显示“0”，应把 30H
即 00110000B → 显示存储区
欲显示“F”，应把 46H
即 01000110B → 显示存储区



要求牢记以下18个字符的ASCII码：

0~9的ASCII码为 **30H ~ 39H**

A~F的ASCII码为 **41H ~ 46H**

回车符的ASCII码为 **0DH**

换行符的ASCII码为 **0AH**



二. 码制 (p.13)

解决在微型机领域中如何表示有符号数？

- 计算机只能识别0和1组成的数或代码，所以有符号数的符号也只能用0和1来表示，

(一) 真值和机器数的概念



1. **真值**：一个数的数值。

用“+”表示正数，用“-”表示负数

如：+101 -101

2. **机器数**：在计算机中如何表示正负？

把符号数值化，用0表示“+”，用1表示“-”，这样，连同符号位在一起作为一个数，称为机器数。



正数



负数



3. **字长**：包括符号位在内，一个二进制数占有的位数
如：字长 $n=8$ 的二进制数，除了符号位，数值部分为7位
- 由于数值部分的表示方法不同，在微机系统中有符号数可有三种表示方法，即机器数有三种形式，分别叫做原码、反码和补码。



1. 原码:

- 原码表示的有符号数，最高位为符号位，数值位部分就是该数的绝对值。
- 例如：假设某机器为8位机，即一个数据用8位(二进制)来表示，则：

+23(17H)的原码机器数为 **0**0010111

-23(-17H)的原码机器数为 **1**0010111

其中最高位是符号位，后7位是数值位。



2. 反码:

- 反码表示的有符号数，也是把最高位规定为符号位，但数值部分对于正数是其绝对值，而对于负数则是其绝对值按位取反(即1变0，0变1)。

例如：+23的反码机器数为 **00010111**

- 23的反码机器数为 **11101000**

- 数字 ‘0’的反码有2种表示:

$$(+0)_{10} = (00000000)_2$$

$$(-0)_{10} = (11111111)_2$$



3. 补码:

- 补码表示的有符号数，对于正数来说同原码、反码一样，但负数的数值位部分为其绝对值**按位取反后末位加1**所得。

例如：+23的反码为 **0**0010111

-23的反码为 **1**1101000

- 23的补码为 **1**1101001



小结:

- ① 机器数比真值数多一个符号位。
- ② 正数的原、反、补码与真值数相同。
- ③ 负数原码的数值部分与真值相同；负数反码的数值部分为真值数按位取反；负数补码的数值部分为真值数按位取反末位加1。
- ④ 没有负零的补码，或者说负零的补码与正零的补码相同。



⑤ 由于补码表示的机器数更适合运算，为此，计算机系统中负数一律用补码表示。

⑥ 补码机器数的数值范围

设机器数字长为 n 位, 用来表示整数, 则 n 位补码数, 其真值范围为

$$-2^{n-1} \sim +2^{n-1} - 1$$

设：8位补码数为1000,0000 ~ 0111,1111

则：十进制真值数为 $-128 \sim +127$

设：16位补码数为

1000,0000,0000,0000 ~ 0111,1111,1111,1111

则：十进制真值数为 $-32768 \sim +32767$



➤不同编码方式下，相同0/1代码对应的不同真值。

0/1代码	真值(原码)	真值(反码)	真值(补码)
0000	+0	+0	+0
0001	+1	+1	+1
0010	+2	+2	+2
0011	+3	+3	+3
0100	+4	+4	+4
0101	+5	+5	+5
0110	+6	+6	+6
0111	+7	+7	+7
1000	-0	-7	-8
1001	-1	-6	-7
1010	-2	-5	-6
1011	-3	-4	-5
1100	-4	-3	-4
1101	-5	-2	-3
1110	-6	-1	-2
1111	-7	-0	-1



□ 为何补码1000 0000B的真值是-128

补码与同余

给定一个正整数 m ，如果两个整数 a 和 b 满足 $a-b$ 能被 m 整除，即 $m|(a-b)$ ，那么就称整数 a 与 b 对模 m 同余，记作 $a \equiv b(\text{mod } m)$ 。

$$-12 \equiv 244 (\text{mod } 256) \quad 1111 \ 0100$$

$$-120 \equiv 136 (\text{mod } 256) \quad 1000 \ 1000$$

$$-128 \equiv 128 (\text{mod } 256) \quad 1000 \ 0000$$

补码最高位为1其余位0的数的真值建议特殊记忆。



真值与机器数的转换(设字长 $n=8$)

(1) 设 $[X]_{\text{原}}=(96)_{16}$, 则 $x=(?)_{10}$

解: $[x]_{\text{原}}=(96)_{16}=1001\ 0110$

则 $x=-001\ 0110=(-22)_{10}$

(2) 设 $x=(-120)_{10}$, 则 $[x]_{\text{补}}=(?)_{16}$

解: $x=(-120)_{10}=(-1111000)_2$

则 $[x]_{\text{补}}=(1000,1000)=(88)_{16}$

(3) 设 $x=(100)_{10}$, 则 $[x]_{\text{补}}=(?)_{16}$

解: $x=(100)_{10}=(+110,0100)_2$

则 $[x]_{\text{补}}=(0110,0100)_2=(64)_{16}$



若已知一个负数的补码，再取一次补，则

$$\{[x]_{\text{补}}\}_{\text{补}} = [x]_{\text{原}}$$

(4) 设 $[X]_{\text{补}} = (96)_{16}$ ，则 $x = (?)_{10}$

解： $[x]_{\text{补}} = (96)_{16} =$

$$\begin{array}{r} 1001\ 0110 \\ 1110\ 1001 \\ + \quad 1 \\ \hline [x]_{\text{原}} = 1110\ 1010 \end{array}$$

则 $x = -1101010 = (-106)_{10}$



(二) 整数补码的运算

1. 关于“模”的概念

一个计量器的最大容量称为该计量器的“模”

四位计数器能存0000~1111共十六个数,

$$\therefore \text{模} = 2^4$$

八位计数器能存0000,0000~1111,1111共256个数,

$$\therefore \text{模} = 2^8$$

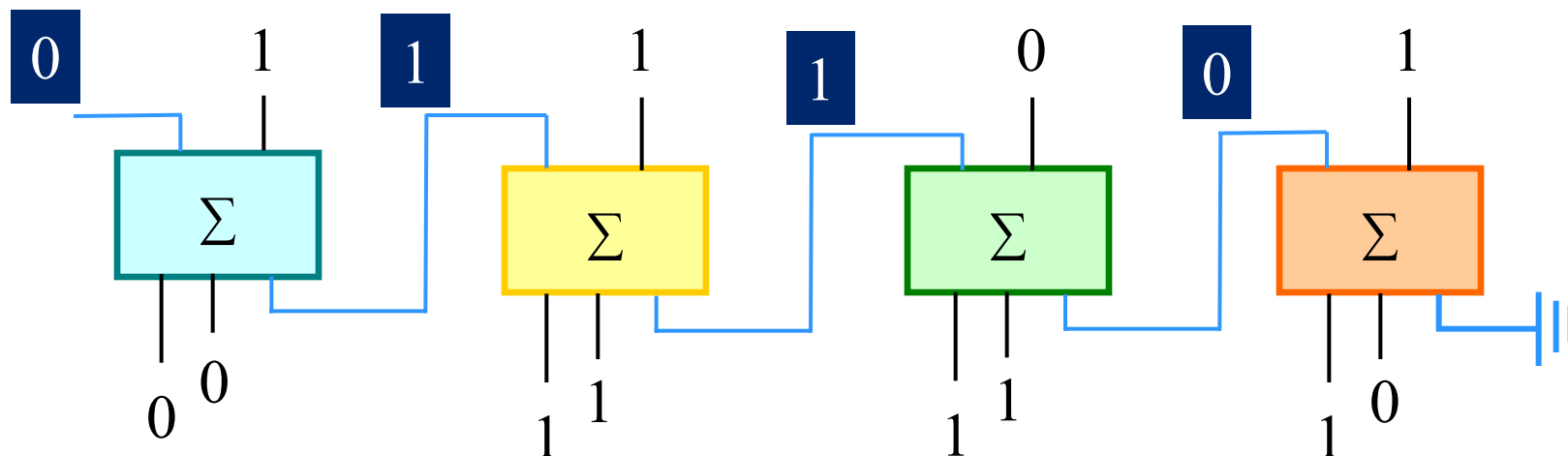
十六位计数器能存

0000,0000,0000,0000~1111,1111,1111,1111共65536个数,

$$\therefore \text{模} = 2^{16}$$



2. 四位的加法器(由四个全加器组成)模 = $2^4 = 16$

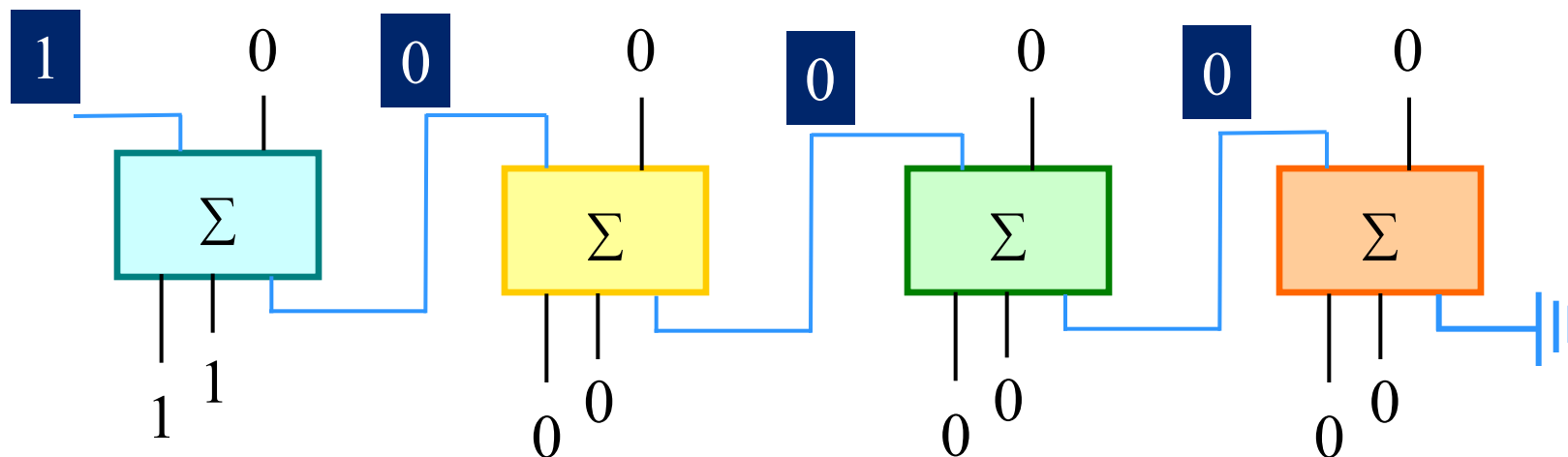


一位全加器有3个输入端(加数、被加数、低位向本位的进位)
2个输出端(本位和、本位向高位的进位)

在上述加法器上进行:
 $7+6=13$, 进位为0



2. 四位的加法器(由四个全加器组成)模 = $2^4 = 16$



在上述加法器上进行8+8:

8+8=16, 进位为1

进位为“1”，其值为16，就是四位加法器的“模”，它被运算器丢失了。



3. 整数补码的加减运算

$$[x + y]_{\text{补}} = [x]_{\text{补}} + [y]_{\text{补}}$$

$$[x - y]_{\text{补}} = [x]_{\text{补}} + [-y]_{\text{补}}$$

条件: (1) 符号位参加运算

(2) 以 2^n 为模(n 为字长)

(3) 当真值满足下列条件时, 结果是正确的,
否则结果错误

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



例1. 设 $x=(66)_{10}$, $y=(51)_{10}$, 以 2^8 为模, 补码运算 $x \pm y$

解: $x=(66)_{10}=+100\ 0010$, $y=(51)_{10}=+011\ 0011$

$$\begin{array}{r} [x]_{\text{补}} = 0100\ 0010 \\ +) [y]_{\text{补}} = 0011\ 0011 \\ \hline [x+y]_{\text{补}} = 0\ 0111\ 0101 \end{array}$$



被运算器丢失, 保存在进位标志寄存器中

$$\begin{array}{r} [x]_{\text{补}} = 0100\ 0010 \\ +) [-y]_{\text{补}} = 1100\ 1101 \\ \hline [x-y]_{\text{补}} = 1\ 0000\ 1111 \end{array}$$



被运算器丢失, 保存在进位标志寄存器中

$\therefore x+y=+117$, 进位=0, $\therefore x-y=+15$, 进位=1



例2. 以 2^8 为模，补码运算，求 $66+99$ ， $-66-99$

解: $[66]_{\text{补}} = 0100\ 0010$

+ $[99]_{\text{补}} = 0110\ 0011$

$[66+99]_{\text{补}} = 0\ 1010\ 0101$



被运算器丢失，保存在进位标志寄存器中

$\therefore 66+99 = -101\ 1011 = -91$

$-66-99 = +101\ 1011 = +91$

$[-66]_{\text{补}} = 1011\ 1110$

+ $[-99]_{\text{补}} = 1001\ 1101$

$[-66-99]_{\text{补}} = 1\ 0101\ 1011$



被运算器丢失，保存在进位标志寄存器中

结果都是错的？



错误原因：

因为字长 $n=8$ ，8位字长的补码数，
其真值 范围是： **$-128 \sim +127$**

而 $66+99 = 165$ ， 真值超过127，

$-66-99 = -165$ ， 真值小于-128

总之， \because 运算器位数不够，不能表示165和-165，
 \therefore 出错。



(三) 无符号数的概念

计算机处理的数值数据，包括有符号数和无符号数两类。
有符号数用补码表示，其最高位代表符号。

什么是无符号数？

即数的最高位不代表符号，而是数值的一部分。

某数是无符号数，还是有符号数，其物理意义是由程序员定义

如：编程统计某班级单科的及格人数。

学生成绩没有负数，所以成绩应视为无符号数。

如：编程统计某科室工资总额...

工资是无符号数

如：数 $N=(1111, 1111)_2$

若它是有符号补码数，则其值 $=-1$

若它是无符号数，则其值 $=255$



(四) 溢出和进位的概念

1. 进位：运算后，最高位向更高位的进位值。

溢出：运算结果超出了运算器所能表示的范围。

下列情况就发生了溢出：

8位加法器，运算无符号数，结果 ≥ 256

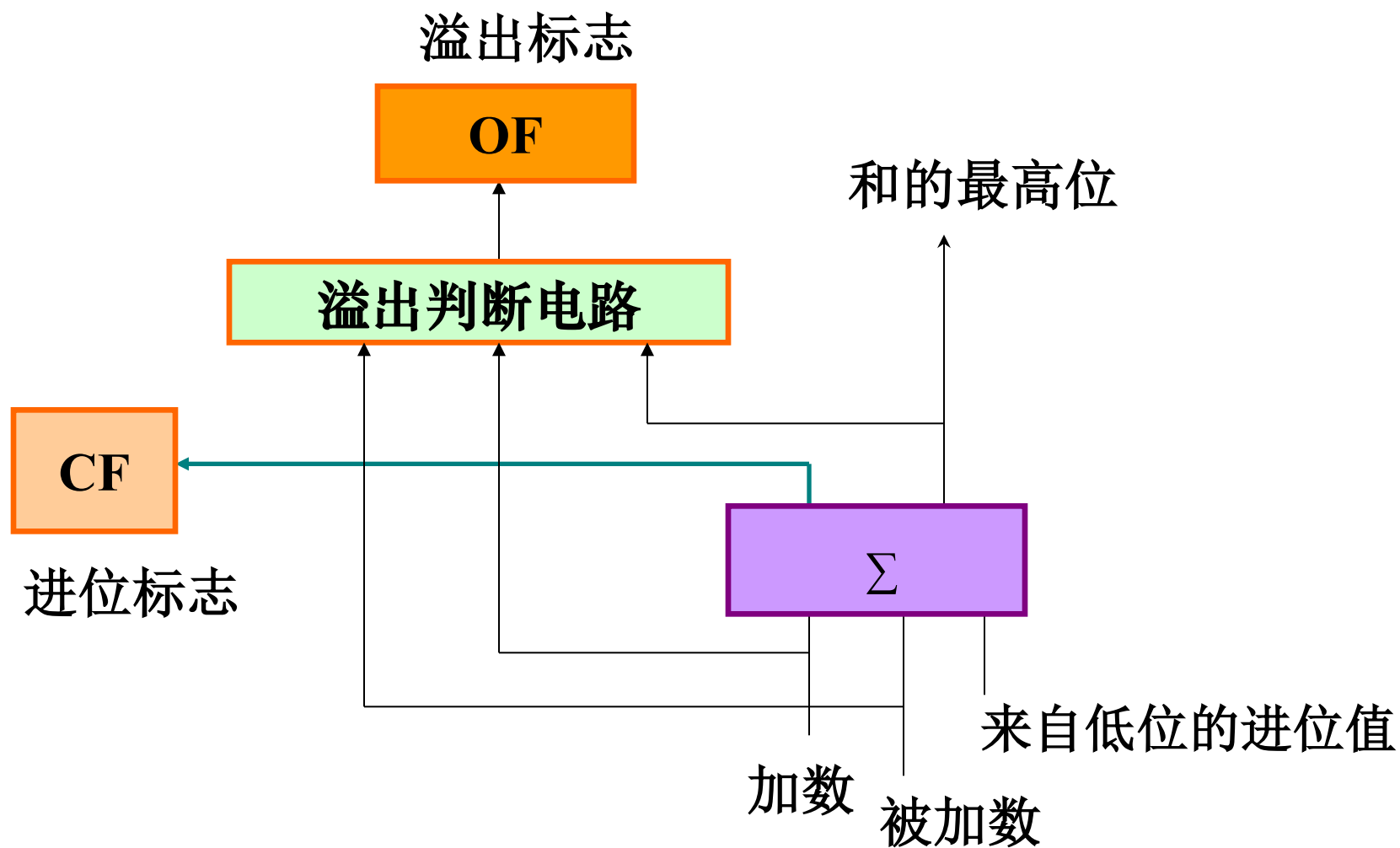
8位加法器，运算有符号数，结果 $> +127$ ， < -128

16位加法器，运算无符号数，结果 ≥ 65536

16位加法器，运算有符号数，结果 $> 2^{15}-1, < -2^{15}$



2. 计算机怎样表示进位和溢出





- ① 运算器对有符号数和无符号数同样对待
- ② 最高位的进位值保存在“进位标志寄存器”中
- ③ 如加数与被加数的最高位相同，却与结果的最高位相异，则溢出标志置1；其他情况下溢出标志为0

3. 程序员如何判断溢出错？

如果参与运算的数是无符号数，则判进位标志，进位标志=1，表示溢出错。

如果参与运算的数是有符号数，则判溢出标志，溢出标志=1，表示溢出错。



例： 加数= 01000010
被加数= 01100011 (+

和= 010100101

CF=0, OF=1

若加数、被加数为无符号数，则结果=10100101=165

若加数、被加数有符号数，则结果=-91 有符号数运算
出错

再如： 加数= 10111110
被加数= 10011101 (+

和= 101011011

CF=1, OF=1

若它们是无符号数，结果=+91 (CF=1)

它们是有符号数，结果=+91 (OF=1)

结果都错



重点！ 以 2^8 为模，补码运算 $(-85)+(-70)$ 的值，要求有计算过程，写出加数和被加数的补码、结果的补码和真值，O标志和C标志的值，并判断结果是否正确。

解： $[-85-70]_{\text{补}} = [-85]_{\text{补}} + [-70]_{\text{补}}$

$$[-85]_{\text{补}} = 1010\ 1011$$

$$+) [-70]_{\text{补}} = 1011\ 1010$$

$$\hline [(-85)+(-70)]_{\text{补}} = 1\ 0110\ 0101$$



被运算器丢失，保存
在进位标志寄存器中

可得，运算结果补码为：01100101，真值为+101，且
CF=1，OF=1。

∴ 进行有符号运算，

∴ 运算结果溢出，结果不正确。



1.3 计算机系统的基本组成

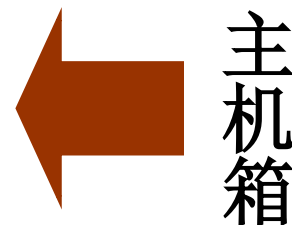
一. 计算机系统组成:

硬件：泛指设备而言

软件：泛指程序而言

硬件：

中央处理器CPU
(运算器、控制器)
存储器系统
I/O接口
电源系统
I/O设备

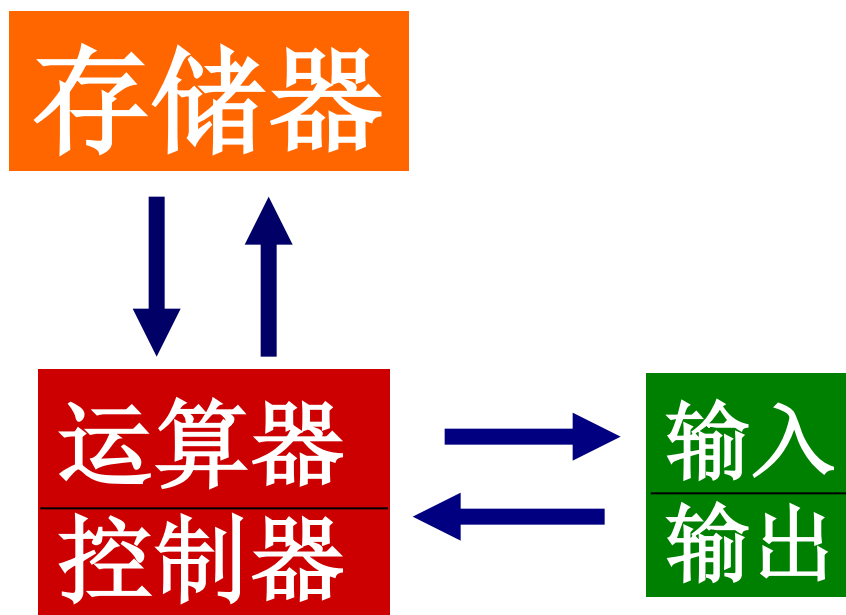


主机箱

软件： 系统软件 应用软件



冯·诺依曼计算机体系结构图



存储程序、程序控制



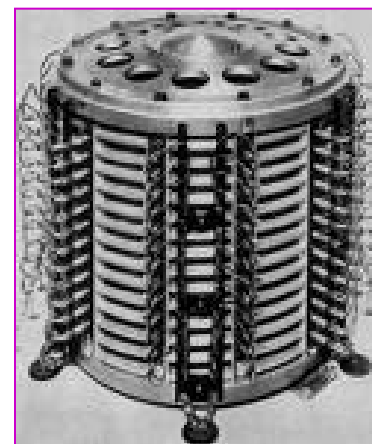
□ 计算机发展

- 第一代(1946~1957)——采用**电子管**作为逻辑部件
- 第二代(1957~1965)——采用**晶体管**作为逻辑部件
- 第三代(1965~1971)——采用**中、小规模集成电路**为主要部件
- 第四代(1971~现在)——采用**大、超大规模集成电路**为主要部件



第1代数字电子计算机

- 时间：约1946-1957
- 使用的元器件：电子管
 - 速度：几十～几万次/秒
 - 内存：磁鼓，千字
 - 外设：磁带
 - 机器语言或汇编语言编程



磁鼓存储器



美国于20世纪50年代生产的IBM704型电子管计算机



第2代数字电子计算机

- 时间：约1957-1964
- 使用的元器件：**晶体管**
 - 速度：几十万次/秒，
 - 内存：磁芯，十万字
 - 外设：磁盘
 - 高级语言编程



电子管

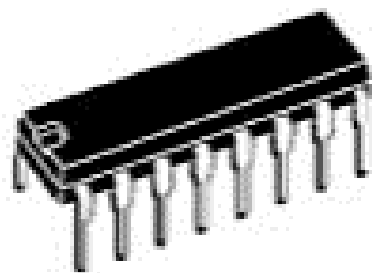
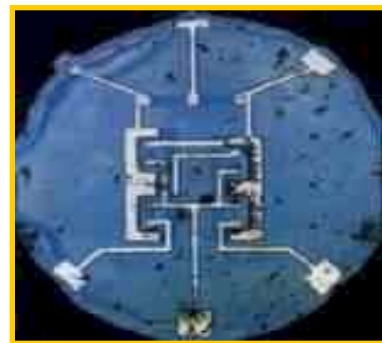


晶体管



第3代数字电子计算机

- 时间：约1965-1973
- 使用的元器件：**中小规模集成电路**
 - 速度：几十万次~几百万次/秒
 - 内存：半导体存储器
 - 软件：高级编程语言



操作系统, 数据库



第4代数字电子计算机

- 时间：从1974年起
- 使用的元器件：**大规模和超大规模集成电路**

(VLSI)

- 速度：几百万次~亿次/秒
- 内存：半导体存储器
- 软件工程，分布式处理等





第1~4代计算机的对比

微型计算机原理与接口技术

代 别	年 代	使用的元器件	使用的软件类型	应用领域
第1代	20世纪40年代中期~50年代末期	CPU: 电子管 内存: 磁鼓	使用机器语言和汇编语言编写程序	科学和工程计算
第2代	20世纪50年代中、后期~60年代中期	CPU: 晶体管 内存: 磁芯	使用FORTRAN等高级程序设计语言	开始广泛应用于数据处理领域
第3代	20世纪60年代中期~70年代初期	CPU: SSI, MSI 内存: SSI, MSI的半导体存储器	操作系统、数据库管理系统等开始使用	在科学计算、数据处理、工业控制等领域得到广泛应用
第4代	20世纪70年代中期以来	CPU: LSI, VLSI 内存: LSI, VLSI的半导体存储器	软件开发工具和平台、分布式计算、网络软件等开始广泛使用	深入到各行各业, 家庭和个人开始使用计算机



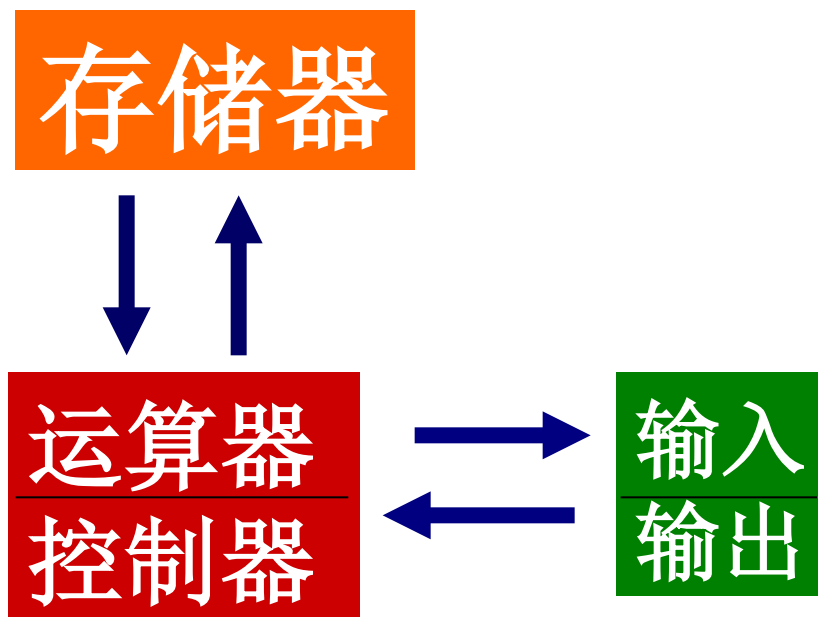
最新的英特尔(Intel) CPUi9-14900K 酷睿14代处理器 24核32线程 睿频至高可达6.0Ghz 36M三级缓存

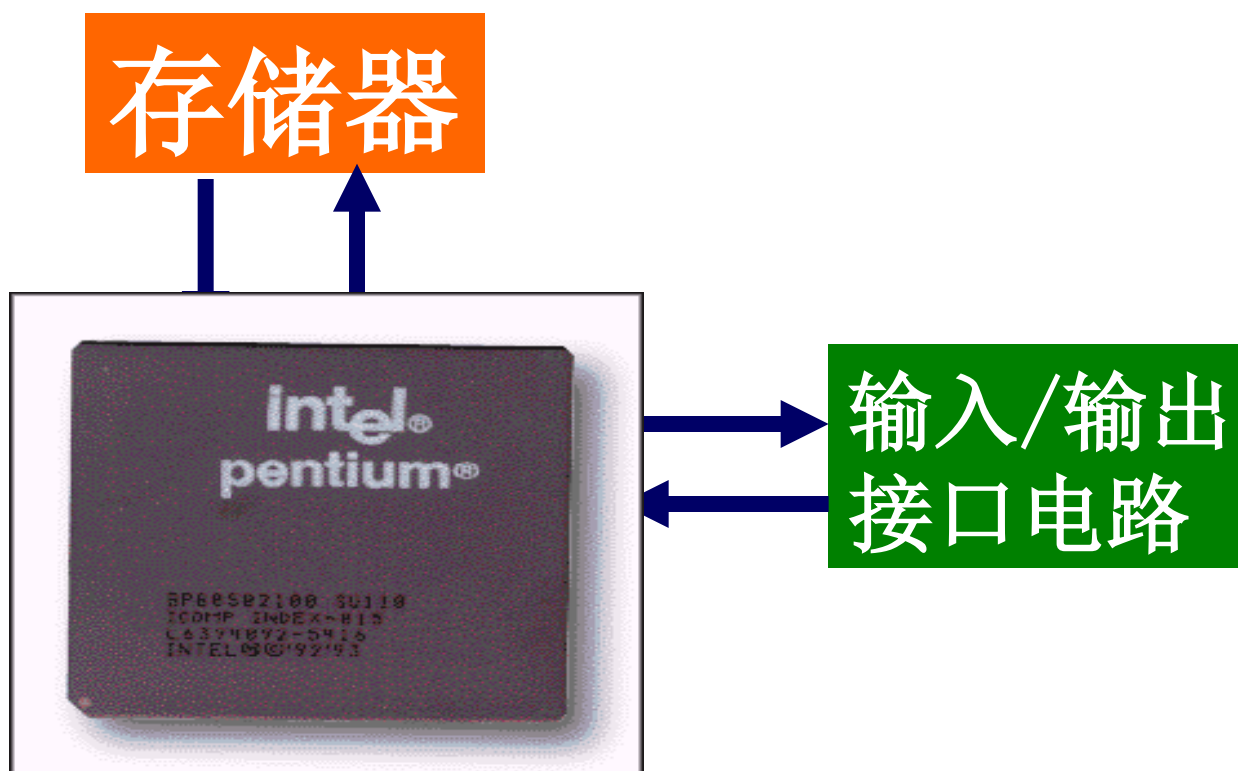
英特尔® 酷睿™ 配置表

处理器型号	处理器内核 (P核+E核)	处理器 线程	英特尔® 智能高速 缓存 (L3)	处理器睿频频率			未锁频	处理器显卡	内存 容量 高达
				英特尔®温度 自适应睿频 加速技术 (英特尔® TVB)	英特尔® 睿频 加速Max技术 3.0频率	单 P 核/ E核 睿频频率 (GHz)			
i9-14900K	24 (8+16)	32	36MB	高达 6.0	高达 5.8	高达 5.6/4.4	√	英特尔® 超核芯 显卡770	192
i9-14900KF	24 (8+16)	32	36MB	高达 6.0	高达 5.8	高达 5.6/4.4	√	不适用	192
i7-14700K	20 (8+12)	28	33MB	-	高达 5.6	高达 5.5/4.3	√	英特尔® 超核芯 显卡770	192
i7-14700KF	20 (8+12)	28	33MB	-	高达 5.6	高达 5.5/4.3	√	不适用	192
i5-14600K	14 (6+8)	20	24MB	-	-	高达 5.3/4.0	√	英特尔® 超核芯 显卡770	192
i5-14600KF	14 (6+8)	20	24MB	-	-	高达 5.3/4.0	√	不适用	192
i9-13900KS	24 (8+16)	32	36MB	高达 6.0	高达 5.8	高达 5.4/4.3	√	英特尔® 超核芯 显卡770	128
i9-13900K	24 (8+16)	32	36MB	高达 5.8	高达 5.7	高达 5.4/4.3	√	英特尔® 超核芯 显卡770	128
i9-13900KF	24	32	36MB	高达	高达	高达	√	不适用	128



按照冯·诺依曼的计算机体系结构思想
存储程序、程序控制/存储程序控制







计算机按体积、性能和价格等分类，可分为：

➤ **巨型机、大型机、中型机、小型机、微型机**

什么是微型计算机？

以微处理器为基础，配以内存存储器及输入输出(I/O)接口电路和相应的辅助电路而构成的裸机。

由微型计算机配以相应的外围设备(如打印机)及其他专用电路、电源、面板、机架以及足够的软件构成的系统叫做微型计算机系统。



微型计算机的发展

微型计算机——以大规模、超大规模集成电路为主要部件，以集成了计算机主要部件-控制器和运算器的微处理器为核心所构成的计算机系统。

- 第一代(1971~1972) ——4位和低档8位微机
4004、8008
- 第二代(1973~1977) ——中、高档8位微机
8080/8085
- 第三代(1978~1984) ——16位微机
8086/8088
- 第四代(1985~1999) ——32位微机
80386、80486、Pentium、Pentium Pro
MMX Pentium、Pentium II、Pentium III
Pentium 4
- 第五代(2000~至今)——Itanium



微型计算机的特点：

- **体积小、重量轻、价格低廉**
- **简单灵活、可靠性高**
- **功耗低、对使用环境要求不高**
- **结构灵活、应用面广**

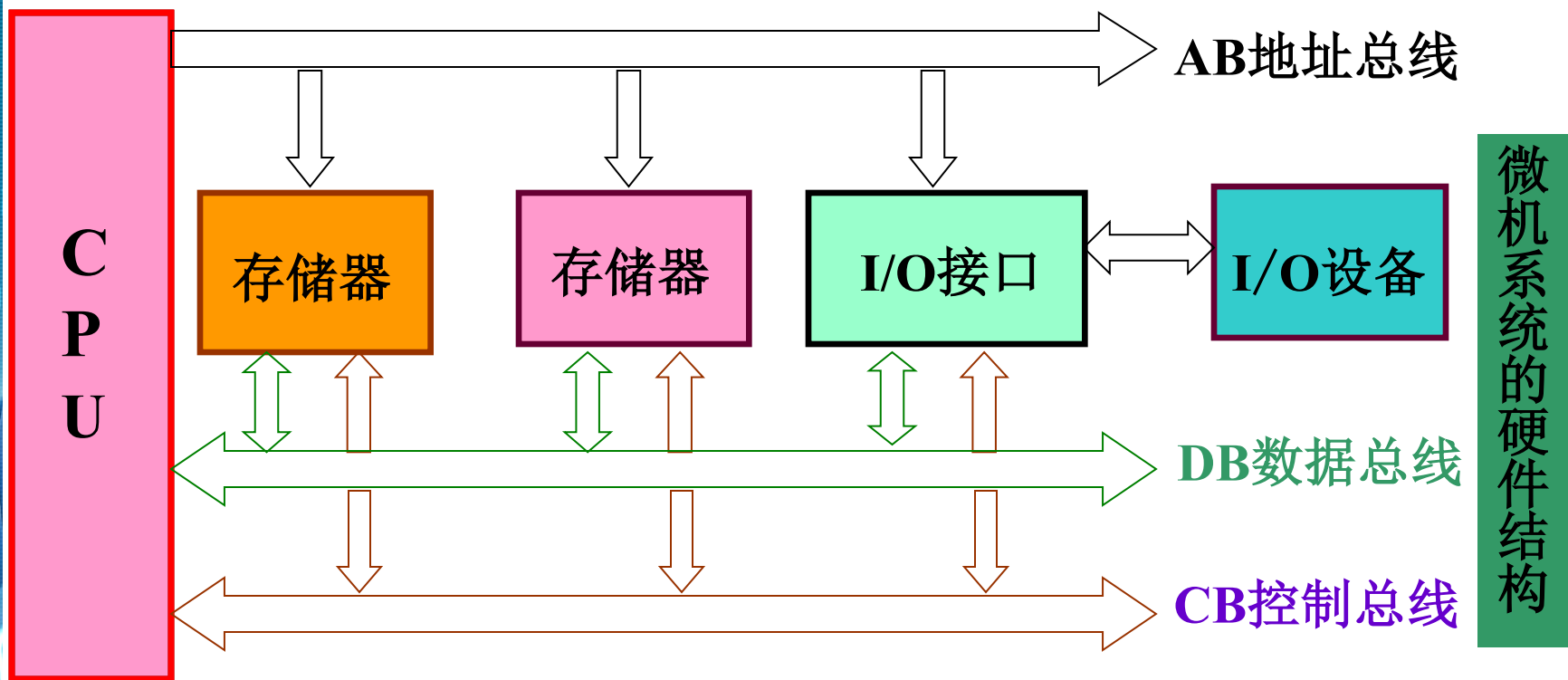


微型计算机的发展方向

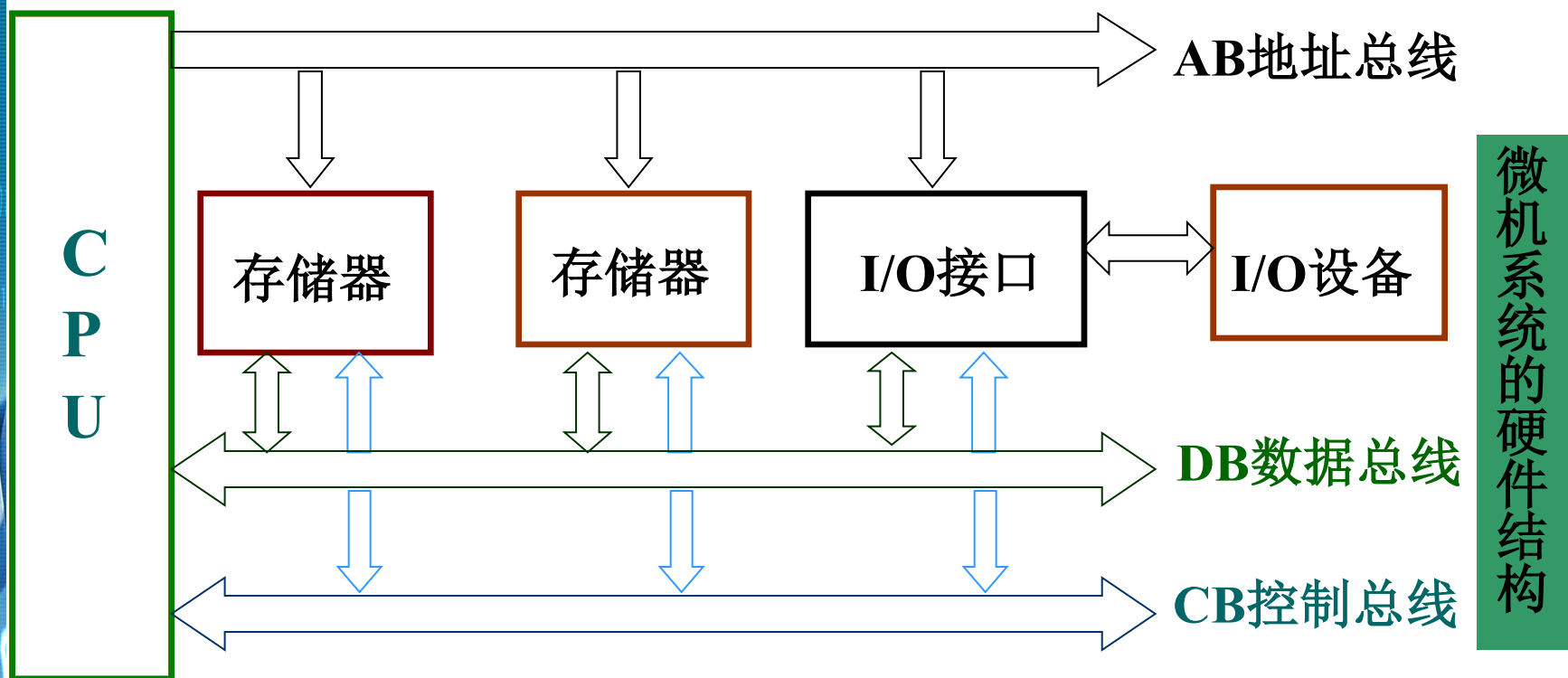
- **并行化**—运算速度更高、存储容量更大、功能更强、并行处理。
- **微型化**—减小体积、重量、价格，便于携带。
- **网络化**—将分布在各区域的计算机和外部设备连成一个功能强大的网络系统，共享软硬件和数据信息资源。
- **多媒体化**—具有处理文本、图形图像、音频、视频及网络等功能，实现电脑、电视、电话的“三电一体”。
- **智能化**—模拟人的感觉和思维，具有逻辑推理和学习能力，能会“看、听、说、想、做”。



一. 微型机硬件结构

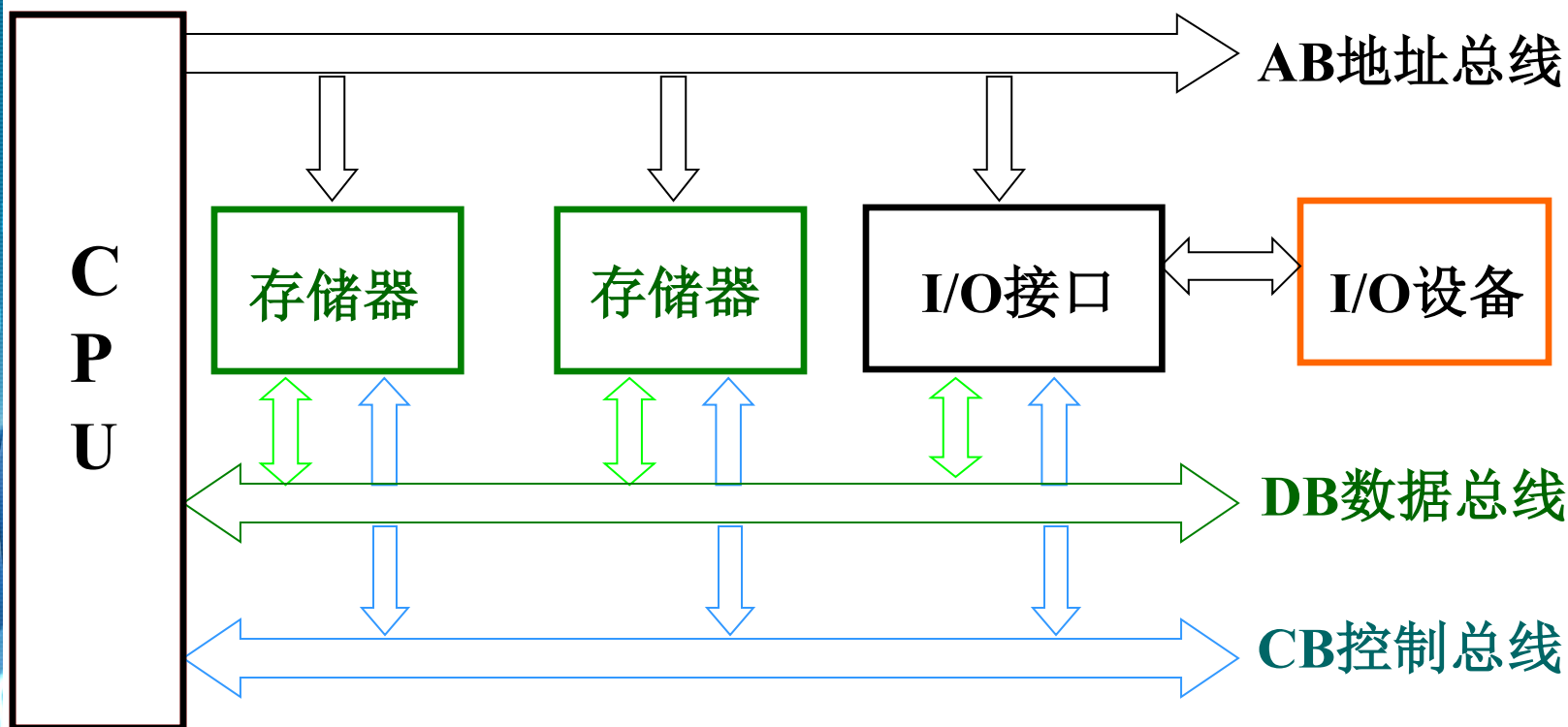


① 以**CPU**为核心通过3条总线连接存储器、I/O接口



① 以CPU为核心通过3条总线连接存储器、I/O接口

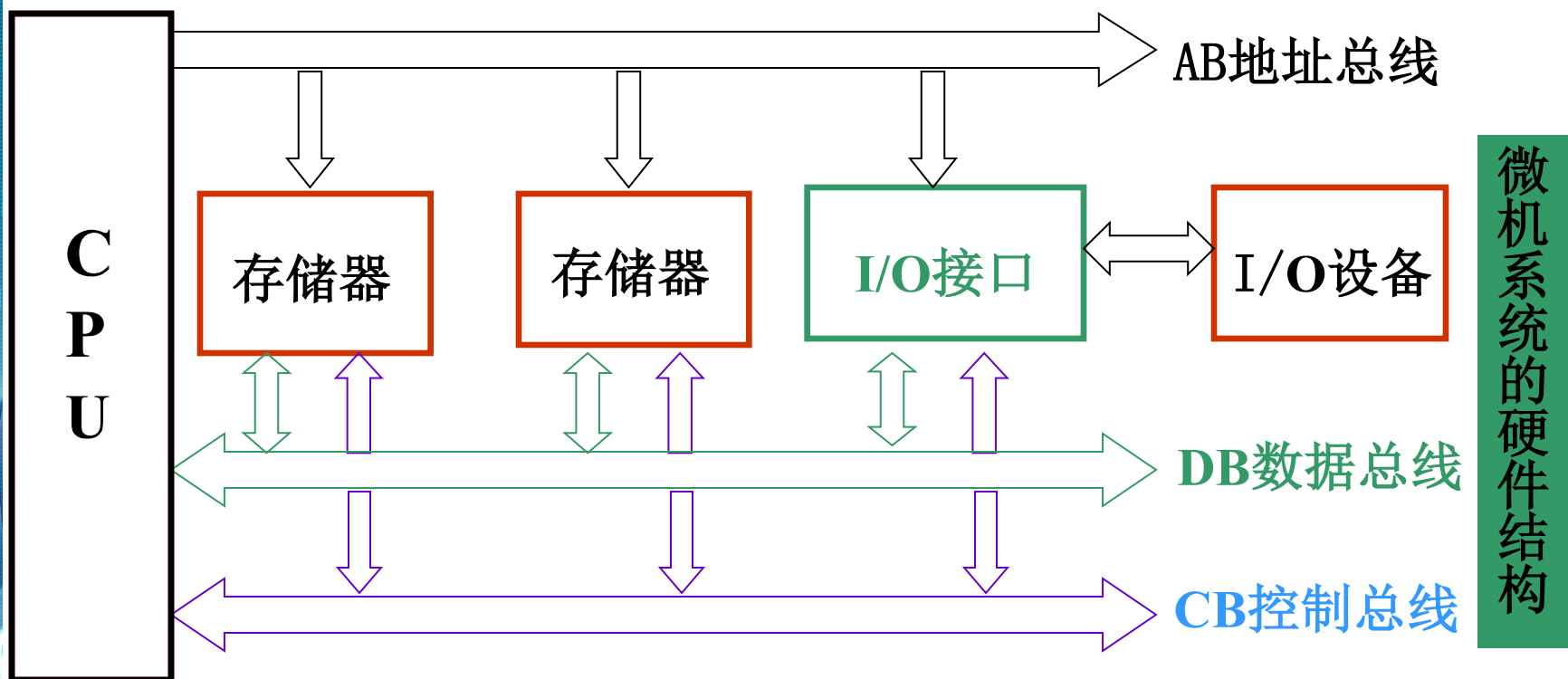
□ **CPU**: 即微处理器，是超大规模集成电路，内部集成了运算器、控制器、存储器管理部件.....



① 以CPU为核心通过3条总线连接存储器、I/O接口

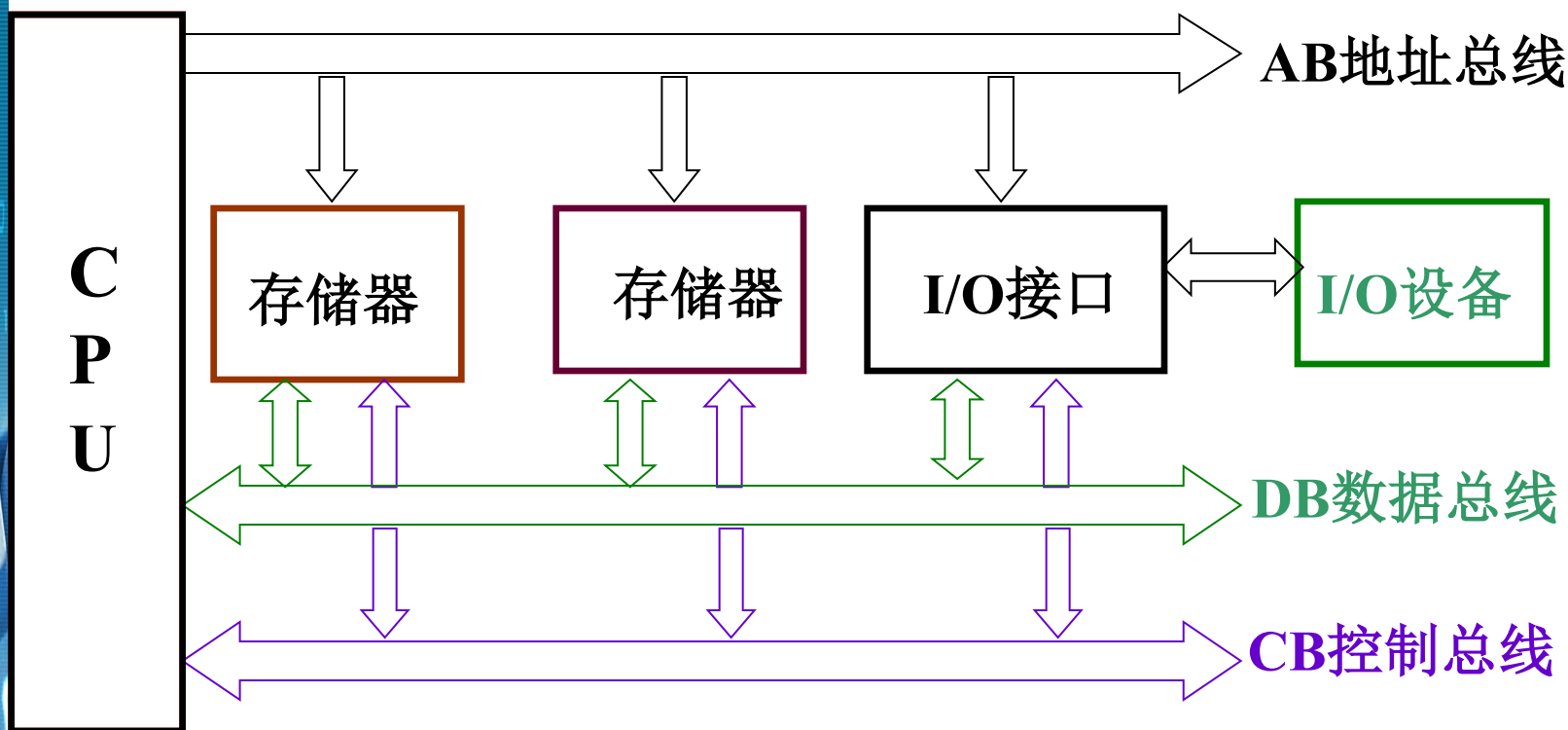
□ 存储器：指系统的主存储器，简称为内存。

用来存放程序、数据



① 以CPU为核心通过3条总线连接存储器、I/O接口

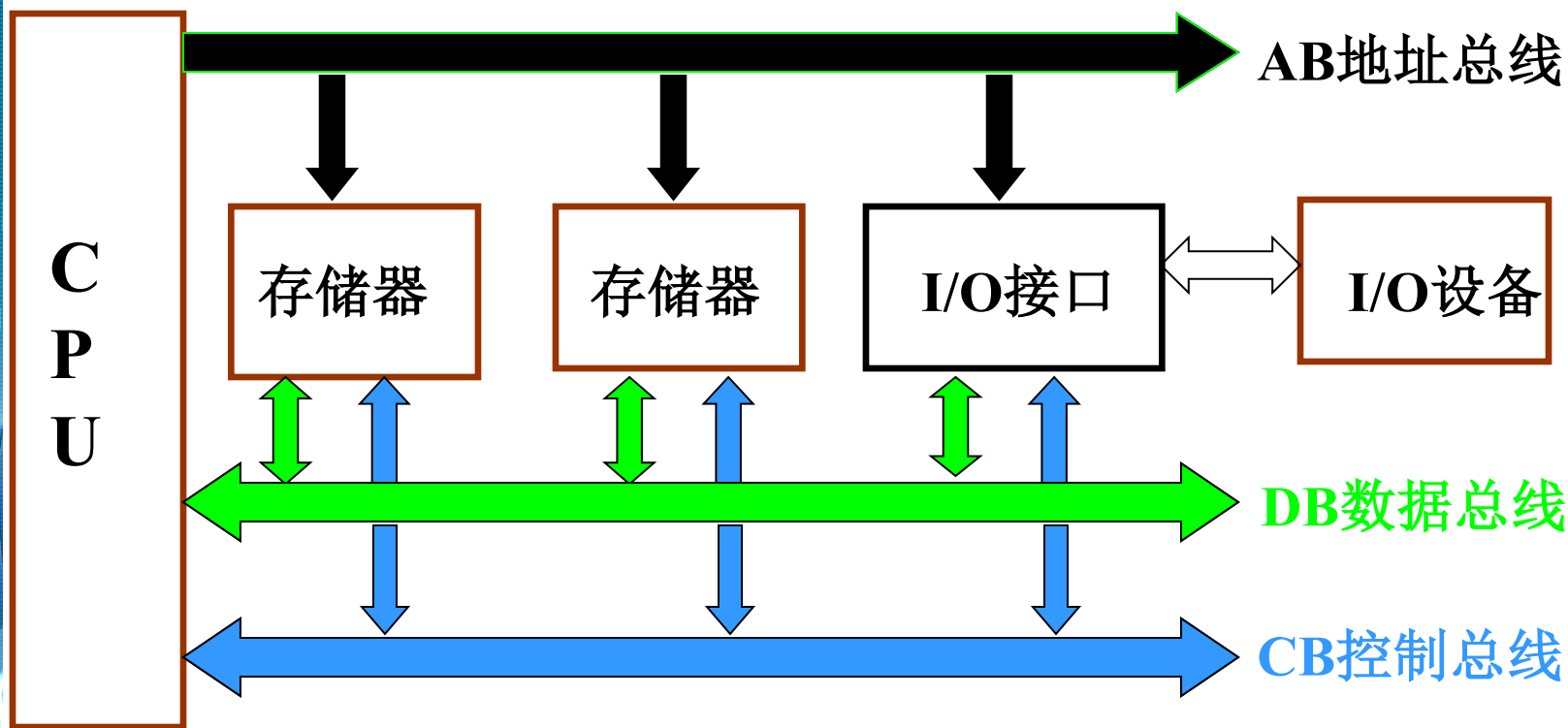
★ **I/O接口**: 是CPU和外部设备交换信息的“中转站”



① 以CPU为核心通过3条总线连接存储器、I/O接口

□ **I/O设备**: 如键盘、显示器、打印机……

注意: 硬盘(外存储器)也是I/O设备!



② **总线**:总线是CPU与存储器、I/O接口交换信息的公共通道。



按总线上信息传输的物理意义，总线分为：

- 地址总线:传输CPU访问存储器，
访问I/O端口的地址信号。
- 数据总线:传输CPU读/写内存，
读写I/O端口时的数据。
- 控制总线:CPU发出的控制命令，
或外部向CPU提出的请求。

地址总线通常是单向总线，

数据总线通常是双向总线，

大部分控制线是单向，少数是双向。

“3条”是习惯说法,其实每一条都有若干根。



□ 术语:

“读”：即输入，信息从外部→CPU

“写”：即输出，信息从CPU→外部

“读内存”：从存储器取信息→CPU

“写内存”：信息写入存储器



存储器基础知识

一.分类:

存储器

辅助存储器:磁盘、光盘

主存储器:RAM、ROM (EPROM)

高速缓冲存储器:静态RAM

存贮容量
存贮周期

大
↑
小

1. RAM: 随机存储器,习惯上称为“内存”, CPU 执行指令可对其进行“读”、“写”操作。

- 静态RAM: 集成度低, 信息稳定, 读写速度快。
- 动态RAM: 集成度高, 容量大, 缺点是信息存储不稳定, 只能保持几个毫秒, 为此要不断进行“信息再生”, 即进行“刷新”操作。



- 内存条:由于动态RAM集成度高, 价格较便宜, 在微机系统中使用的动态RAM组装在一个条状的印刷板上。系统配有动态RAM刷新控制电路, 不断对所存信息进行“再生”。

2. ROM:只读存储器

只读存储器是指: 所存信息只能读出,不能写入。

- 掩模式ROM: 初始信息是在芯片制造时写入的。
- EPROM: 初始信息是在专门的写入器上写入的。



3. ROM / EPROM在微机系统中的应用:

- 存放“基本输入/输出系统程序”(简称BIOS)。
- BIOS是计算机最底层的系统管理程序，操作系统和用户程序均可调用。

4. 高速缓冲存储器Cache:

Cache位于CPU与主存储器之间，由高速静态RAM组成。容量较小，为提高整机的运行速度而设置，应用程序不能访问Cache，CPU内部也有Cache。



二. 存储器容量:

□ 存储器由若干“存储单元”组成，每一单元存放一个“字节”的信息。

1字节即为8位二进制数

2字节即为1个“字”

4字节即为1个“双字”

□ 1K容量为1024个单元

$1\text{M}=1024\text{K}=1024*1024\text{单元}$

$1\text{G}=1024\text{M}$

$1\text{T}=1024\text{G}$



三. 存储器地址与读写操作:

系统为每一单元编排一个地址，地址码为二进制数，习惯上写成16进制。

1. 存储器容量由地址线“宽度”决定:

□ 1M容量的存储器

地址范围: **00000H~FFFFFH**

由20根地址线提供地址码。

□ 16M容量的存储器

地址范围: **000000H~FFFFFFFH**

由24根地址线提供地址码。



□ 4G容量的存储器

地址范围：**0000,0000H~FFFF,FFFFH**

由32根地址线提供地址码。

注意操作系统位数与微机系统位数关系！

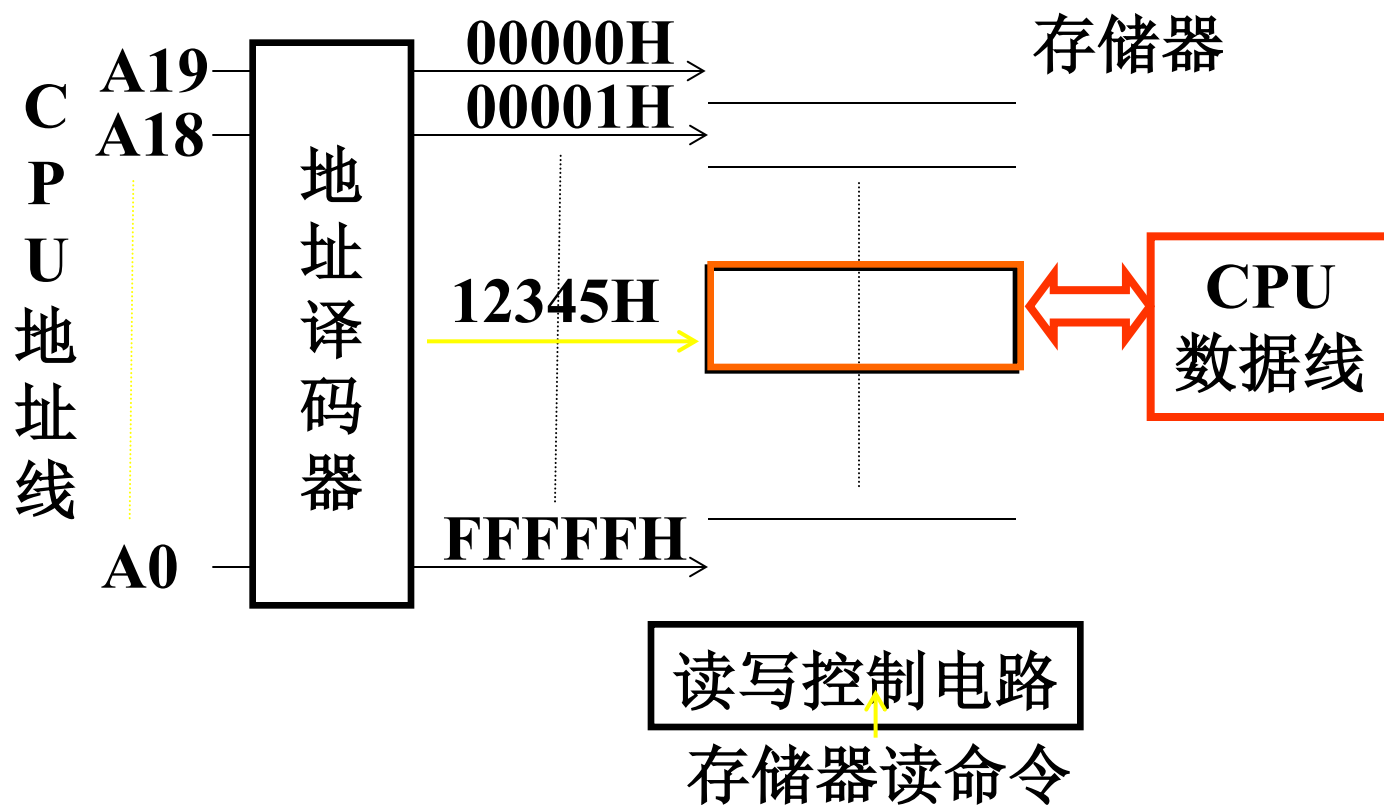
2.存储器读写示意：

为了读写存储器，由地址译码电路对地址码进行“翻译”，从而“选中”某一单元，在CPU的存储器读命令的控制下读出某一单元的内容→数据线。在存储器写命令的控制下把数据线信息→某一个存储单元。下面以动画方式演示读写过程：



读存储器：

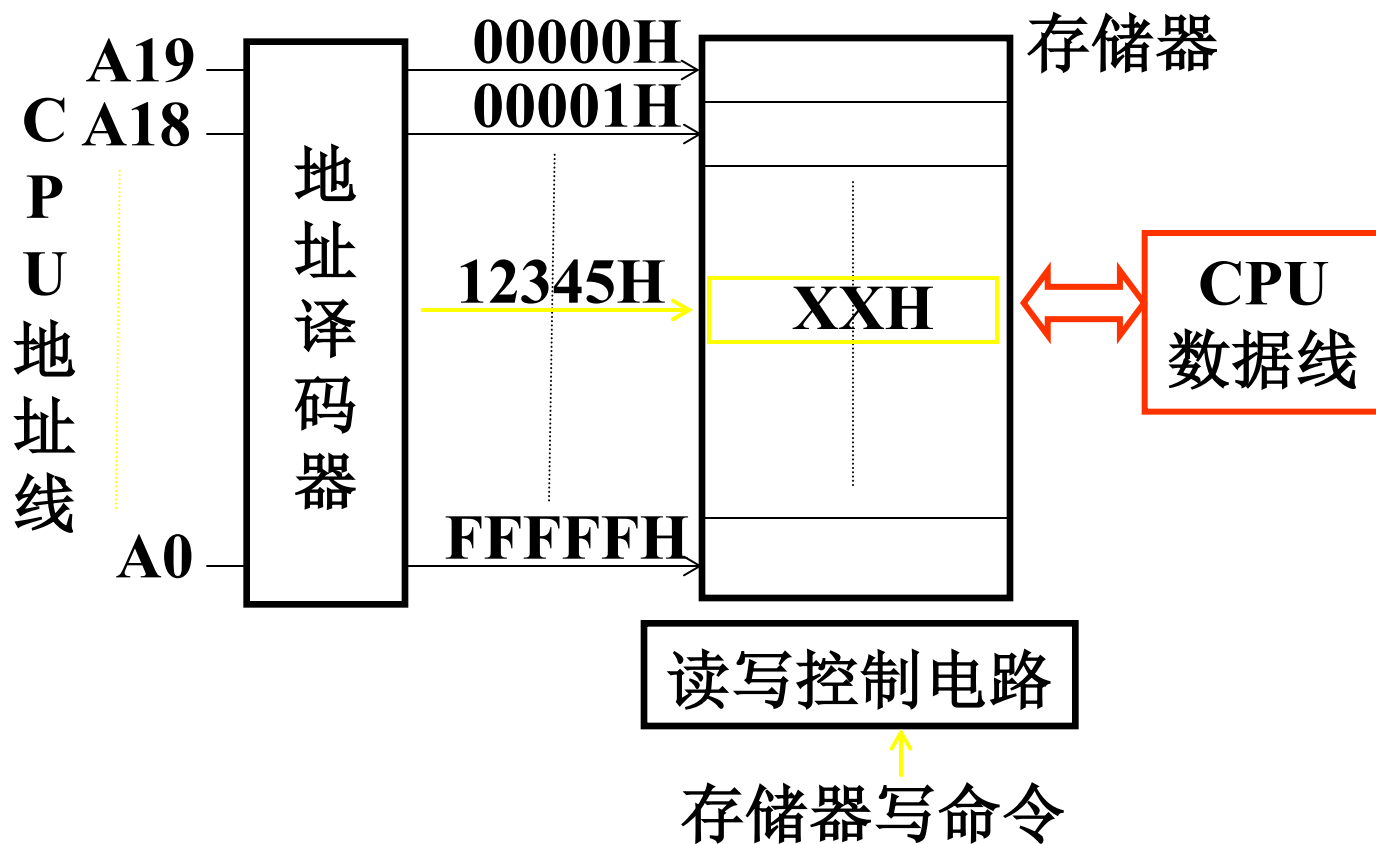
由地址译码电路对地址码进行“翻译”，从而“选中”某一单元，在CPU的存储器读命令的控制下，读出某一单元的内容→数据线。





写存储器：

由地址译码电路对地址码进行“翻译”，从而“选中”某一单元，在CPU的存储器写命令的控制下，把数据线信息→某存储单元。





第1章 学习重点

- 熟练掌握二进制数，十进制数，十六进制数和BCD码数之间的转换方法。
- 熟练掌握真值数和补码数之间的转换方法。
- 牢记0~9，A~F，回车符，换行符的ASCII码。
- 熟练掌握整数补码的运算方法，并对结果进行分析，深入理解有关进位和溢出的概念。
- 掌握微型计算机的硬件基本结构。



□ 今后，在用汇编语言进行程序设计的时候，二进制数用后缀“B”表示，十六进制数及BCD码数用后缀“H”表示。

如：(1010)₂ 应写成1010B

(5A)₁₆ 应写成5AH

(0111 1000)_{BCD} 应写成78H

(123)₁₀ 应写成123