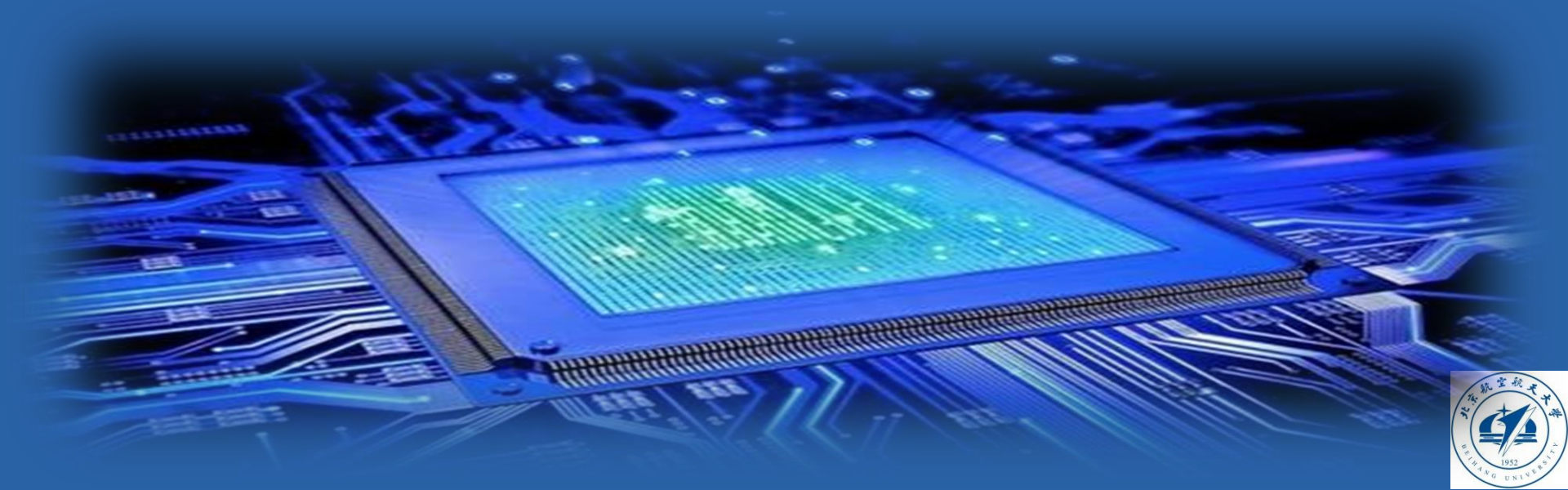
仪器科学与光电工程学院
School of Instrumentation Science and Opto-electronics Engineering

微机原理与接口技术

冯仁剑
rjfeng@buaa.edu.cn
新主楼B534



微机原理与接口技术

----基于嵌入式芯片

第一章 微型计算机概述

1.4 微型计算机的基本组成结构

微型计算机指令系统:

按照指令的执行方式和指令集的复杂程度来划分可以分为两类:

(1)复杂指令集（Complex Instruction Set Computer , CISC)

(2) 精简指令集（Reduce Instruction Set Computer , RISC)

1.4 微型计算机的基本组成结构

(1)复杂指令集（CISC）结构和技术特征

Intel、AMD的x86系列处理器是典型的CISC处理器。

CISC指令集的处理器主要特点：指令集庞大、指令的长度不同，指令译码步骤比较复杂。

随着技术的发展CISC指令集的执行效率和计算机硬件资源利用率低下等问题越来越突出。

1.4 微型计算机的基本组成结构

(2)精简指令集（RISC）结构和技术特征

RISC指令集从如何使计算机的结构更加合理，如何提高处理器的运行速度出发，优选使用频率搞的简单指令，避免复杂指令；将指令长度固定、指令格式和寻址方式种类减少；CPU的控制功能改由硬件逻辑电路实现，取代CISC结构中用软件程序实现控制功能方式。从而提高执行效率。

RISC具有以下特点：指令集有限、简单；单周期指令；大量使用寄存器

RISC的设计应当遵循以下五个原则：

①指令条数少，格式简单，易于译码；

②提供足够的寄存器，只允许load 和store指令访问内存；

③指令由硬件直接执行，在单个周期内完成；

④充分利用流水线；

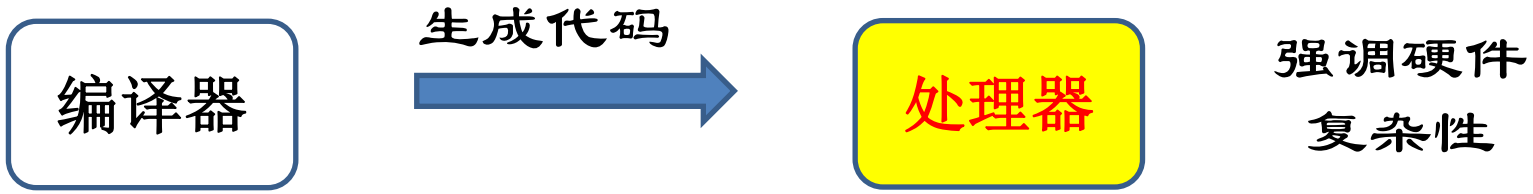
⑤依赖优化编译器的作用；

CISC和RISC的指令集特征对比

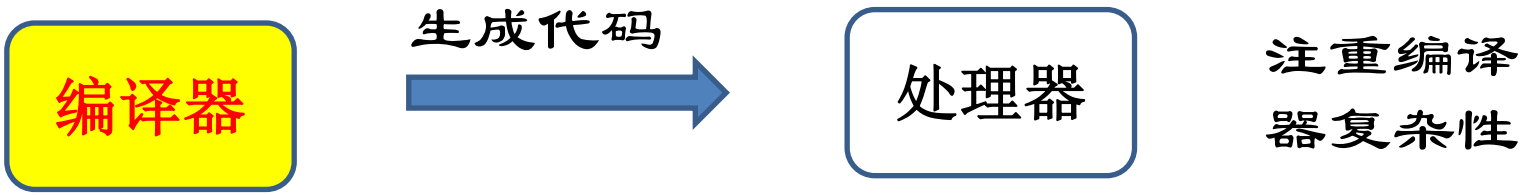
	CISC	RISC
指令集数目	一般大于250条	常小于100
指令长度	不定（3~6字节）	基本固定（2、4字节）
寻址方式	多样、复杂	少，简单
存储访问指令	比较多	只有load/store指令
指令使用频率	差异大	相差不大
指令执行时间	差异大	基本一个周期完成
程序代码长度	较短	较长

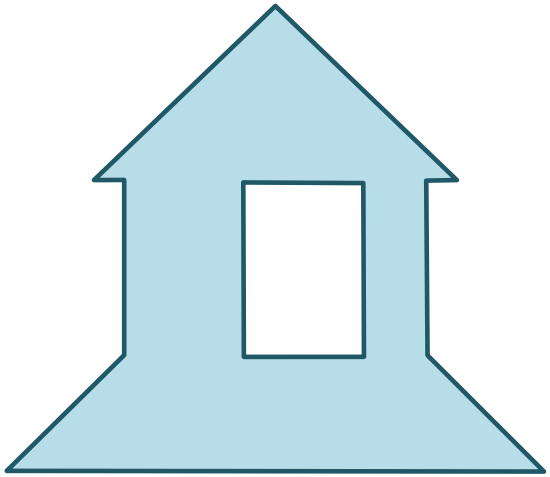
CISC和RISC的不同

CISC：



RISC：



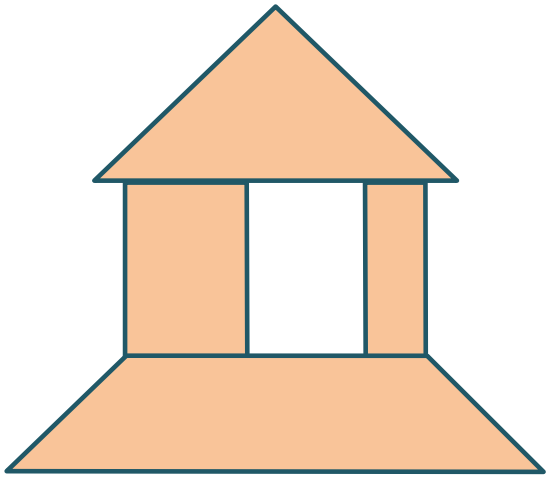


目标或任务

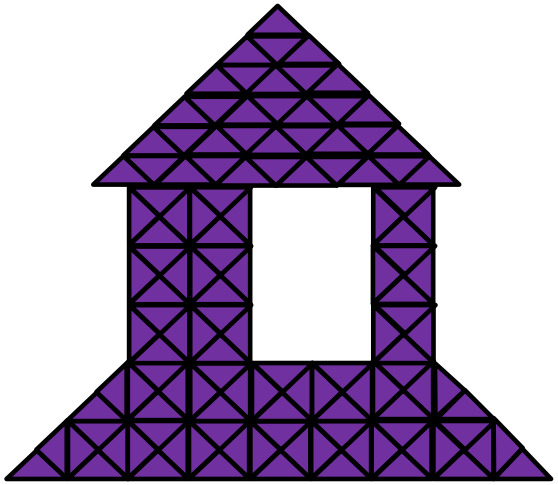
John I. Hennessy

David Patterson

CISC

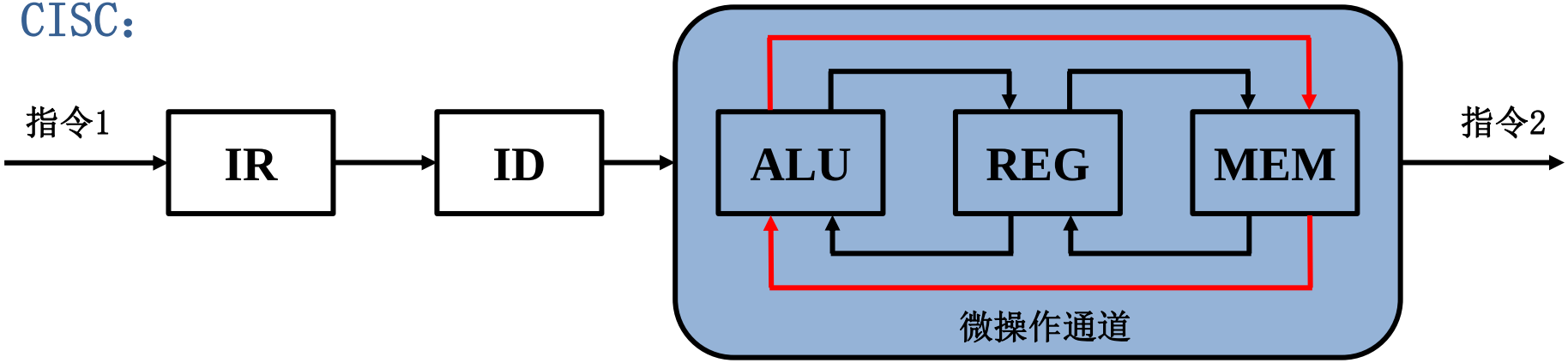


RISC

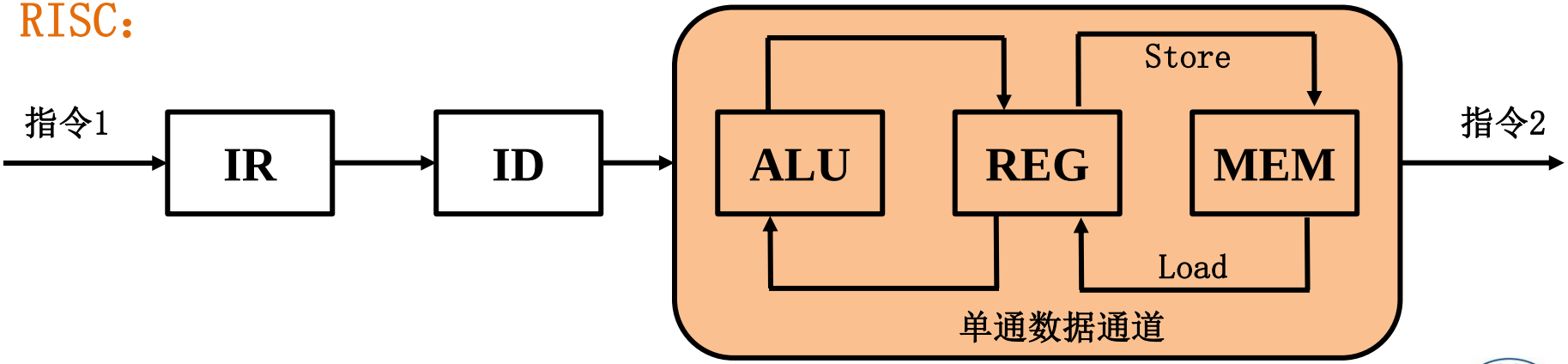


CISC和RISC指令处理过程的差异

CISC：

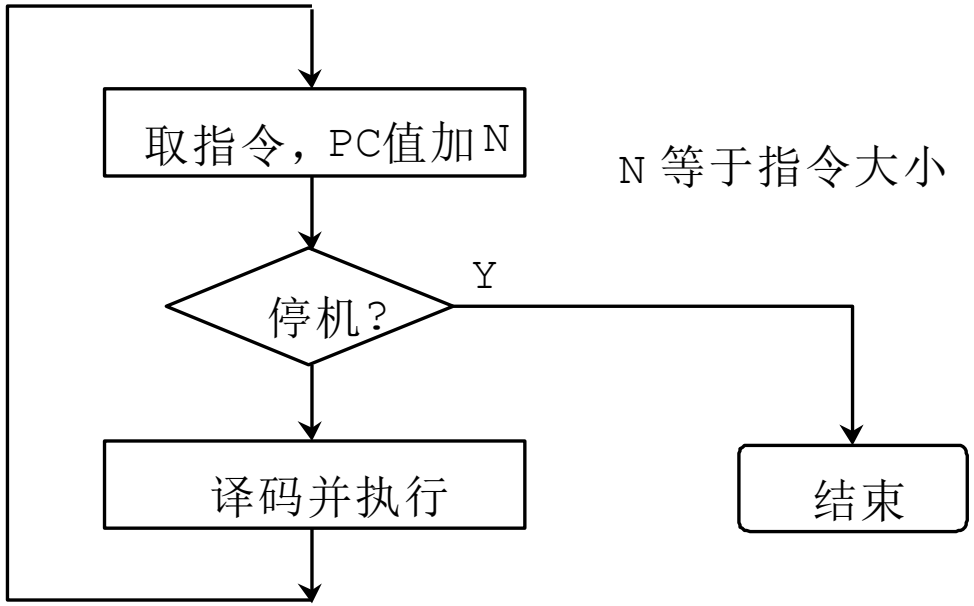


RISC：



1.5 微型计算机的基本工作原理

- *存储程序工作原理*



存储程序工作原理：先存储，再执行。即先把程序和数据送到具有记忆功能的存储器中保存，然后将程序第一条指令的地址送到程序计数器（PC），控制器依次取出存储器中的指令，结合相应的数据，执行每条指令，直至结束。

1.5 微型计算机的基本工作原理

指令执行过程：

- 微型计算机每执行一条指令都分成N个阶段进行：比如N=3

  取指令、译码指令、执行指令。

- 程序运行的过程，实际上就是周而复始地完成这3个阶段操作的过程，直至遇到停机指令时才结束整个机器的运行。

取指令1	译码指令1	执行指令1	取指令2	译码指令2	执行指令2

1.5 微型计算机的基本工作原理

- 指令流水线工作原理
  - 指令流水线类似于工厂的装配线。

    一个产品有若干个部件，不同的部件在装配线上的不同阶段同时装配，不同部件的装配在时间上具有重叠性。

    同样的，完整执行一条指令可分为多个阶段，由不同的部件来同时完成指令执行的不同阶段
  - 有了流水线结构，不同指令的取指、分析、执行3个阶段可并行处理。

1.5 微型计算机的基本工作原理

- 指令流水线工作原理

程序执行是一个周而复始的重复过程，早期计算机的指令执行是一个串行过程：取指令、译码指令和执行指令。

取指令1	译码指令1	执行指令1	取指令2	译码指令2	执行指令2

目前指令主要采用3级流水的工厂装配模式：由不同的部件来同时执行指令的不同阶段，从而提高执行效率。

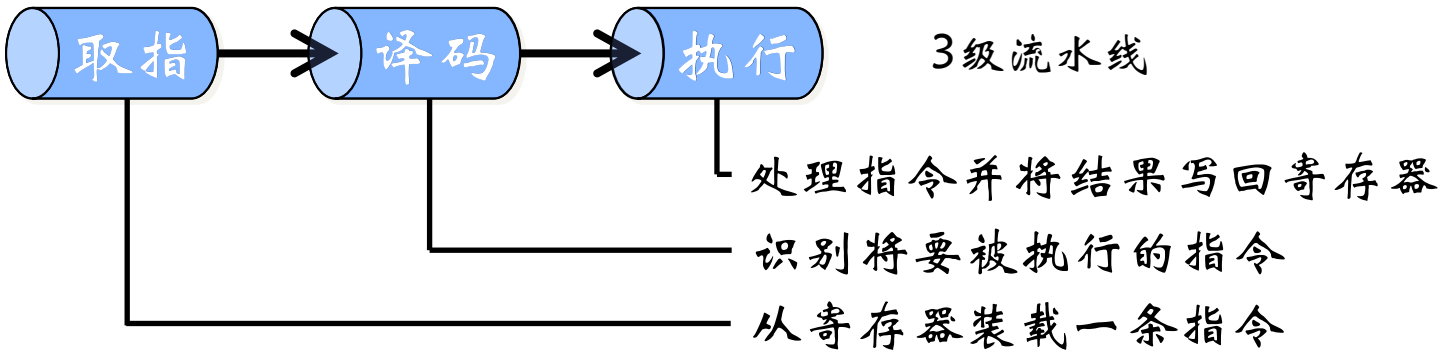
取指令1	译码指令1	执行指令1						
	取指令2	译码指令2	执行指令2					
		取指令3	译码指令3	执行指令3				

1.5 微型计算机的基本工作原理

- **流水线**

ARM处理器使用流水线来增加处理器指令流的速度，这样可使几个操作同时进行，并使处理和存储器系统连续操作，能提供XX MIPS/MHz的指令执行速度。

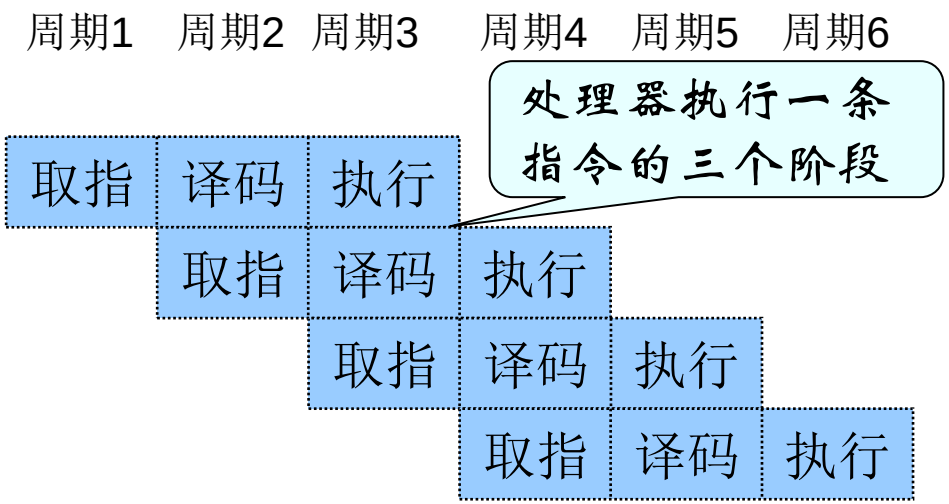
正常操作过程中，在执行一条指令的同时对下一条(第二条)指令进行译码，并将第三条指令从存储器中取出。



1.5 微型计算机的基本工作原理

- 三级流水线结构的指令执行顺序

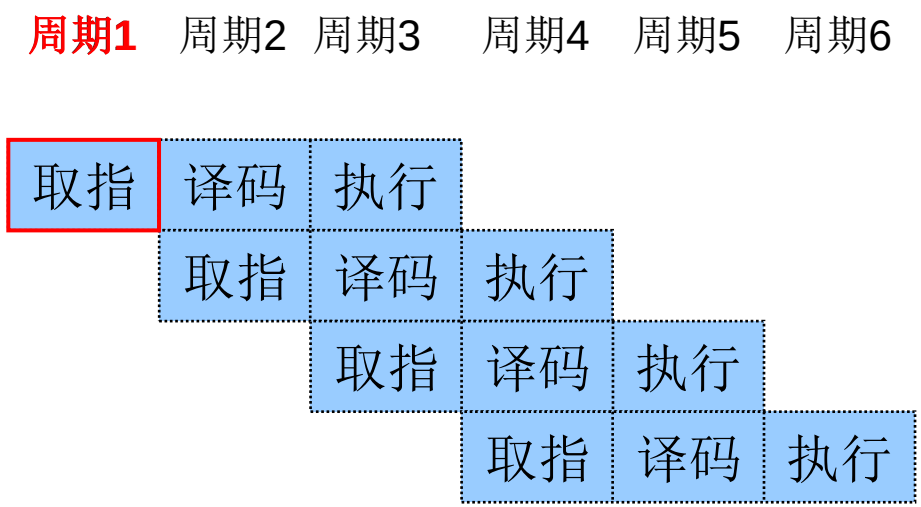
程序存储器



1.5 微型计算机的基本工作原理

- 三级流水线结构的指令执行顺序

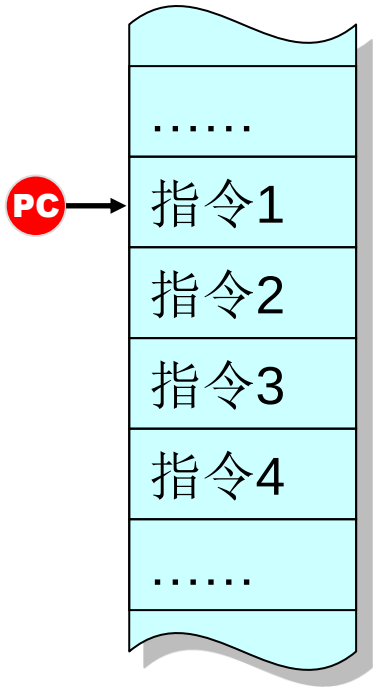
程序存储器



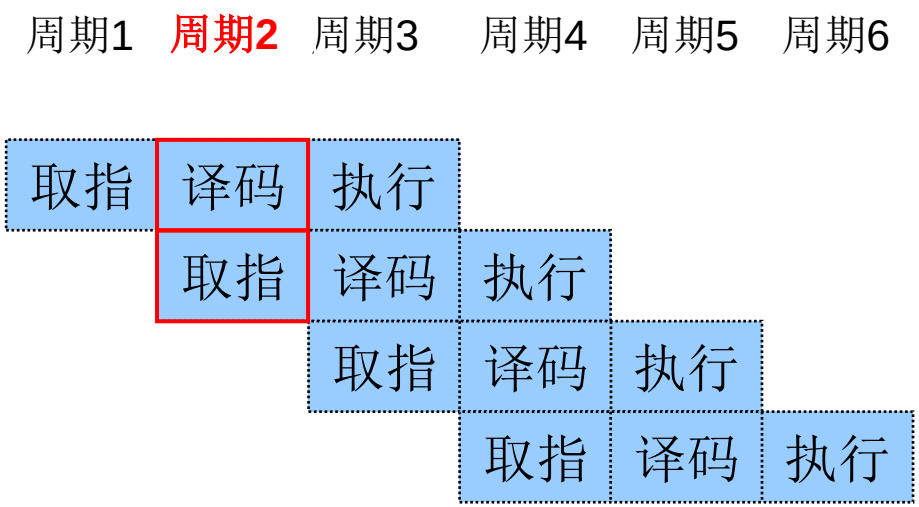
1 在第1个周期，PC指向指令1，此时指令1进入三级流水线的取指阶段。

1.5 微型计算机的基本工作原理

- 三级流水线结构的指令执行顺序



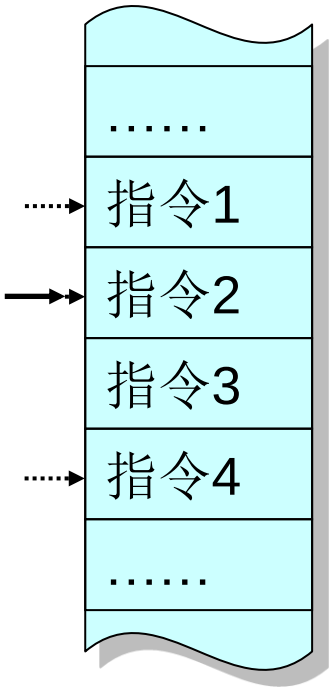
程序存储器



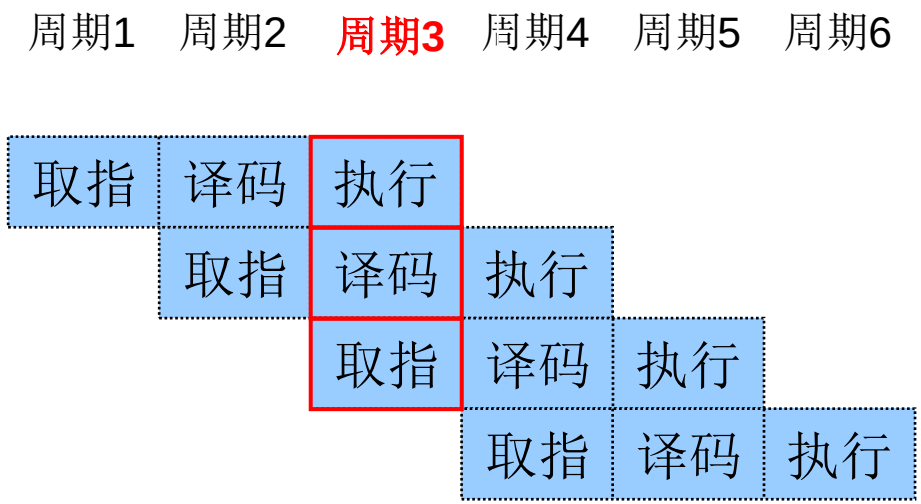
2 在第2个周期，PC指向指令2，此时指令1进入三级流水线的译码阶段，同时取出指令2。

1.5 微型计算机的基本工作原理

- 三级流水线结构的指令执行顺序



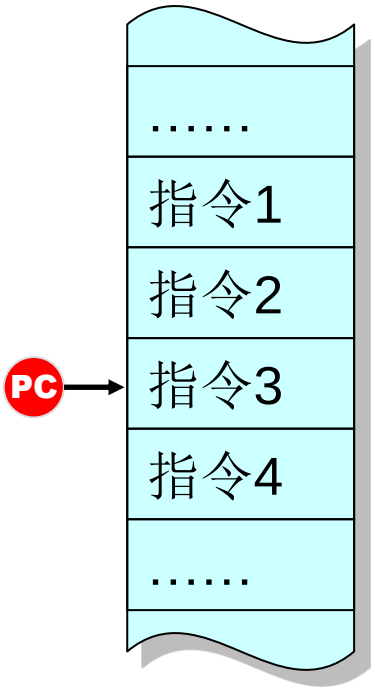
程序存储器



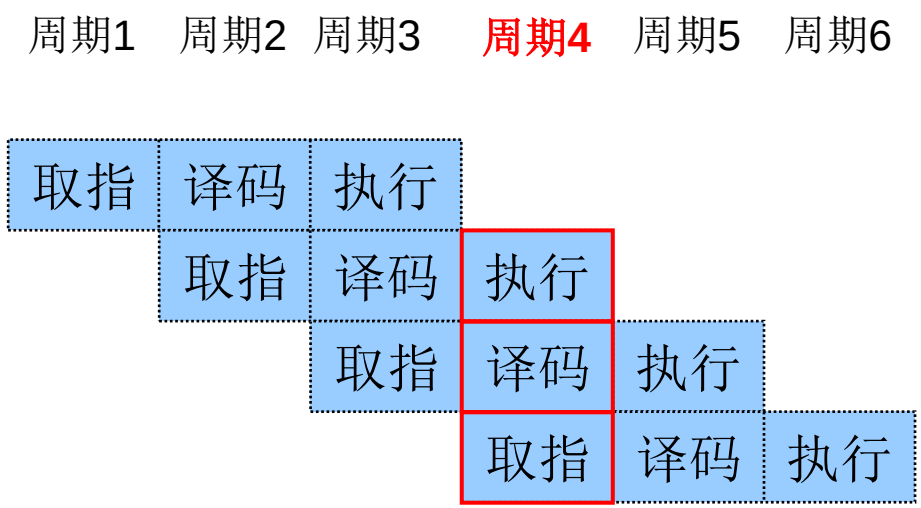
③ 在第3个周期，PC指向指令3，此时指令1进入三级流水线的执行阶段，指令2进入译码阶段，取出指令3。

1.5 微型计算机的基本工作原理

- 三级流水线结构的指令执行顺序



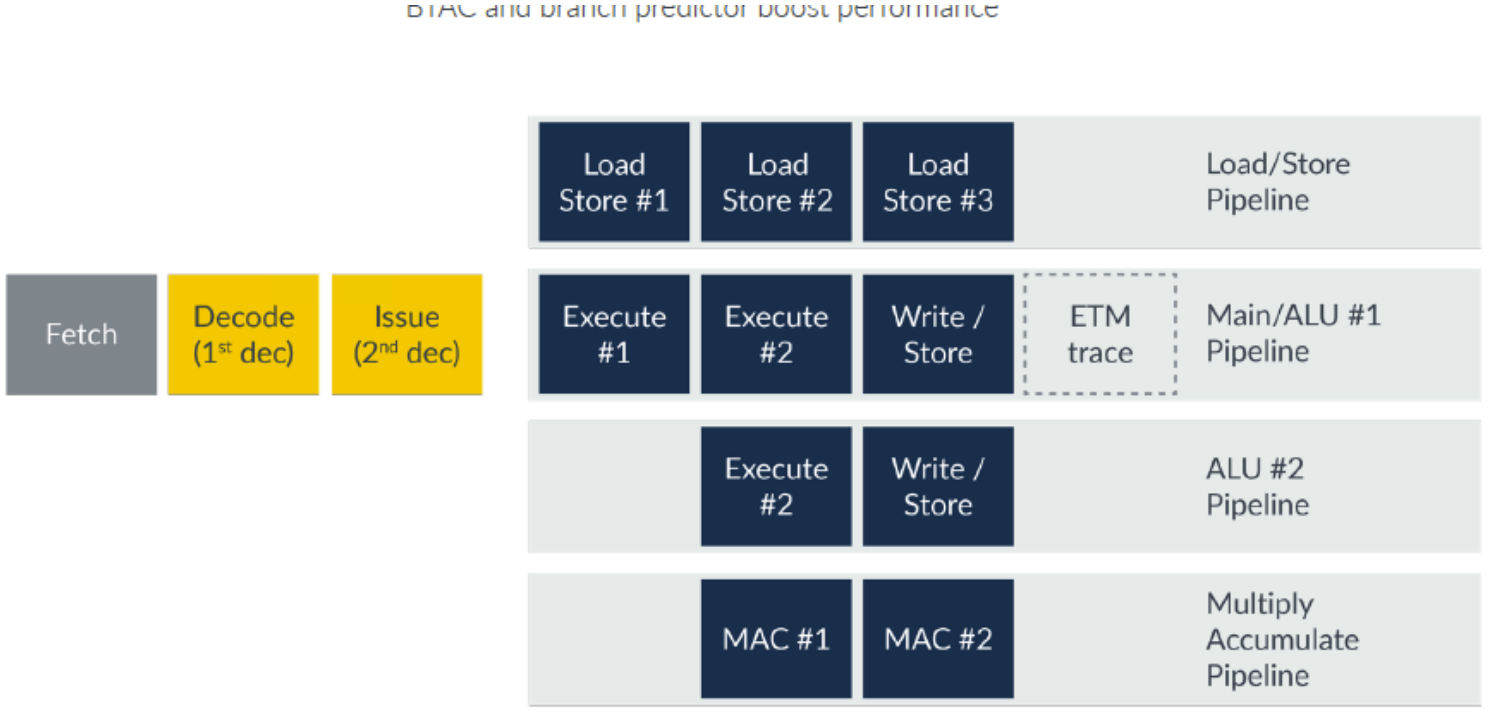
程序存储器



4 在第4个周期，指令1执行完成，指令2和指令3流水线推进一级，同时开始指令4的取指处理。

1.5 微型计算机的基本工作原理

- **Cortex-M7的6级超标量流水线**



概论小结

- 1.1 引言
- 1.2 微型计算机的发展概况
- 1.3 微型计算机分类
- 1.4 微型计算机的基本组成结构
- 1.5 微型计算机的基本工作原理

信息的表示

1. 计算机中数的表示及运算

1.1 无符号整数的表示

1.2 有符号整数的表示

1.3 符号位扩展

1.4 整数的运算

1.5 小数的表示

1.6 字符的表示

1.7 其他数据

2. 存储器中的字

1.计算机中数的表示及运算

计算机：首先需要解决的信息（数据）的表示——“人机交互”。

计算机需要表示哪些数?

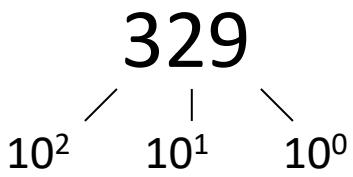
- **Numbers** – signed, unsigned, integers, floating point, complex, rational, irrational, …
- **Text** – characters, strings, …
- **Images** – pixels, colors, shapes, …
- **Sound**
- **Logical** – true, false
- **Instructions**
- …

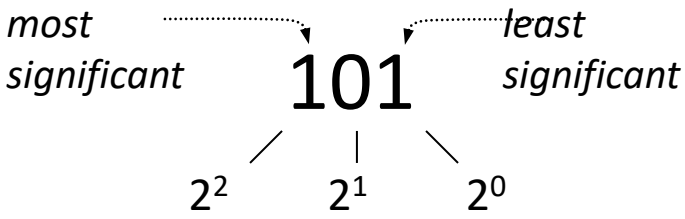
1.计算机中整数的表示及运算

1.1 无符号整数

日常生活中：十进制，十二进制，六十进制等。

计算机中：　采用二进制数表示。





3x100 + 2x10 + 9x1 = 329

1x4 + 0x2 + 1x1 = 5

$Y = \text{“abc”} = a \cdot 2^2 + b \cdot 2^1 + c \cdot 2^0$

（这里的a, b, c的值只能是0或1）

表示范围:

$0 \le Y \le 2^W - 1$

（*W = number of bits*）

1.1 无符号整数的表示

原理：无符号数编码的定义

对向量 $\vec{x}=[x_{w-1}, x_{w-2}, \cdots, x_0]$：

$$B2U_w(\vec{x}) \doteq \sum_{i=0}^{w-1} x_i 2^i$$

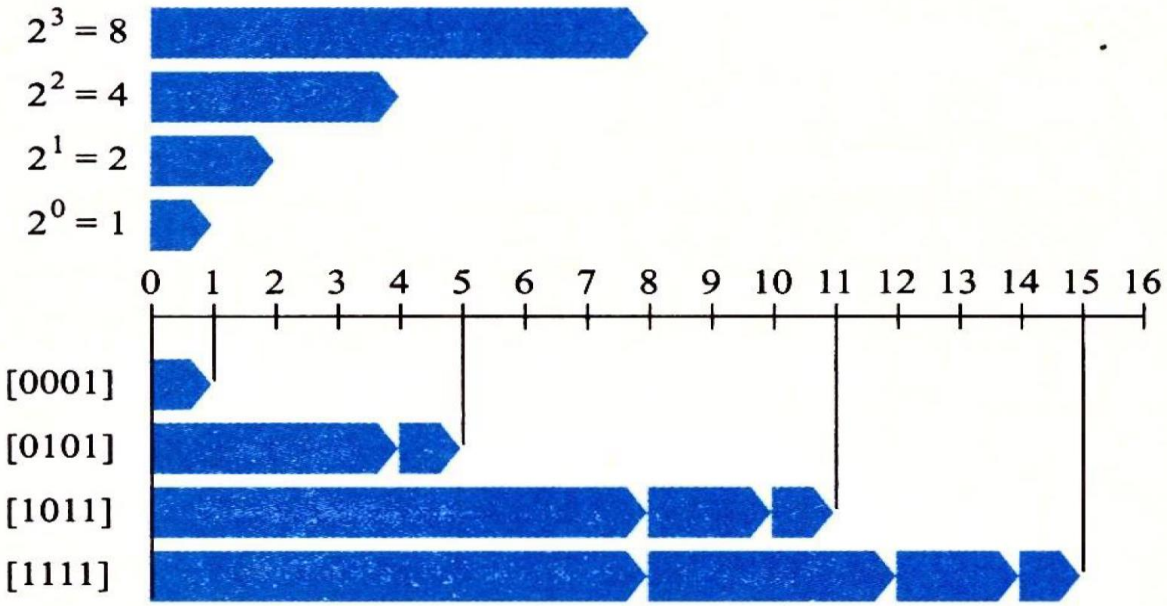


图 2-12　$w=4$ 的无符号数示例。当二进制表示中位 i 为 1，数值就会相应加上 2^i

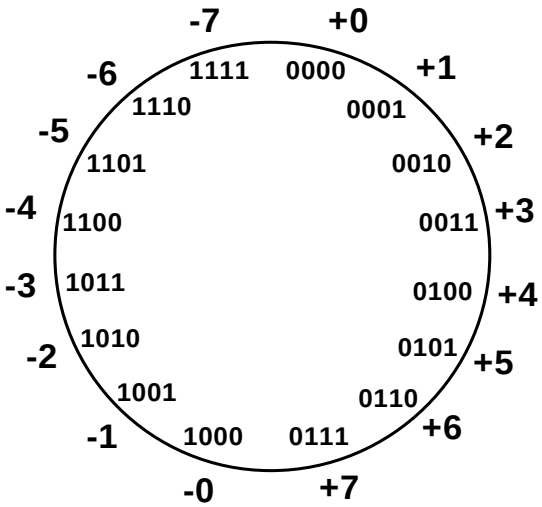
1.2 有符号整数的表示

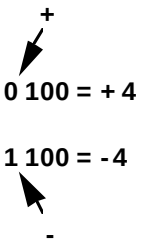
- 有符号整数的表示

  最高位表示符号位。

$$Y = \text{“abc”} = (-1)^a (b \cdot 2^1 + c \cdot 2^0)$$

Range is: $-2^{W-1} + 1 \le i \le 2^{W-1} - 1$

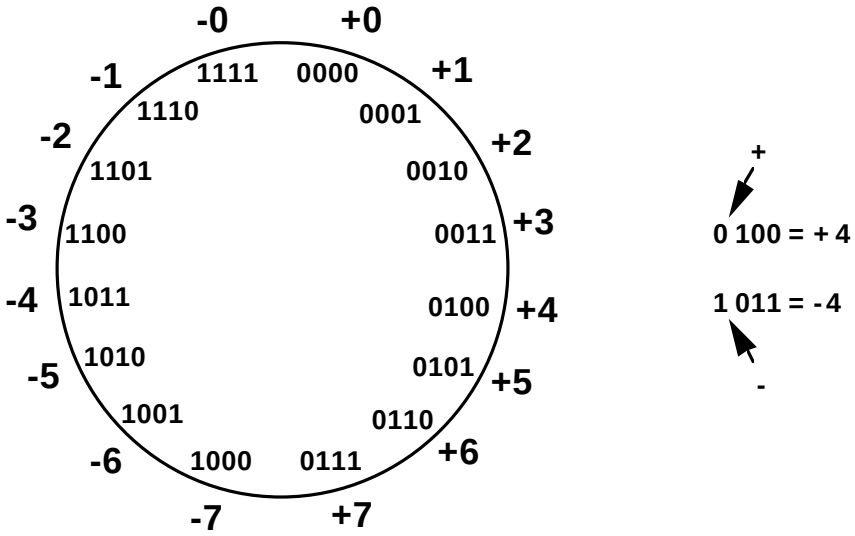




Problems:

- How do we do addition/subtraction?
  - e.g., try 2 + (-3)
- We have two numbers for zero (+/-)!

1.2 有符号整数的表示

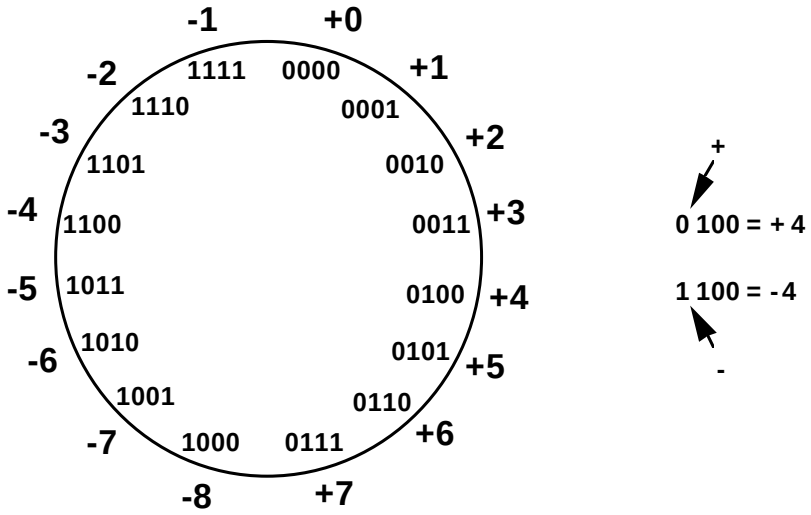


反码(1's complement)表示法

- Subtraction implemented by addition & 1's complement
- How do we do addition/subtraction?
- Still two representations of 0!
- arithmetic circuits are complex, e.g., try 4 + (-3)

1.2 有符号整数的表示

与补码表示法相比，负数仅顺时针移动了一位！



补码(2's complement)表示法

Advantages:

- Operations need not check the sign
- Only one representation for 0
- Efficient use of all the bits

补码表示方式的优点，使计算机的运算更为简单

补码:数字系统中常用的整数表示方式。

1.2 有符号整数的表示

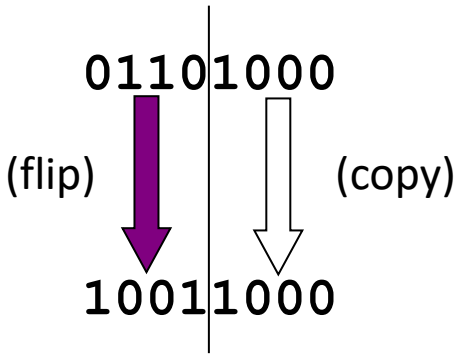
总之：计算机中有符号数表示均采用二进制补码形式表示。具体如下：

1、0和正整数的补码还是本身。

2、负数的补码等于对应正整数各位取反+1。

或：从右往左复制正整数的所有位，直至第一个‘1’位为止，剩下位取反。

```
  01101000
  10010111
+         1
----------
  10011000
```



原理：补码编码的定义

对向量 $\vec{x}=[x_{w-1}, x_{w-2}, \cdots, x_0]$：

$$B2T_w(\vec{x}) \doteq -x_{w-1}2^{w-1} + \sum_{i}^{w-2} x_i 2^i$$

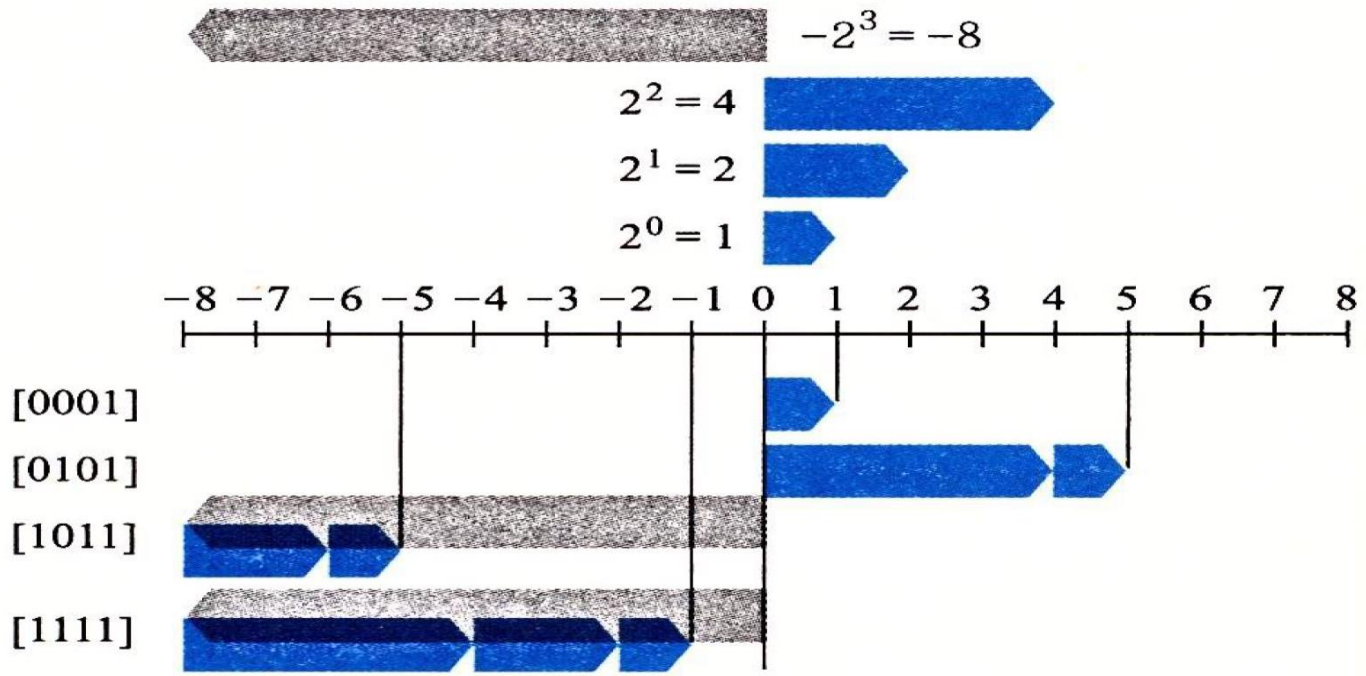


图 2-13　$w=4$ 的补码示例。把位 3 作为符号位，因此当它为 1 时，对数值的影响是 $-2^3=-8$。这个权重在图中用带向左箭头的条表示

数的转换

1、二进制→十进制

展开求和

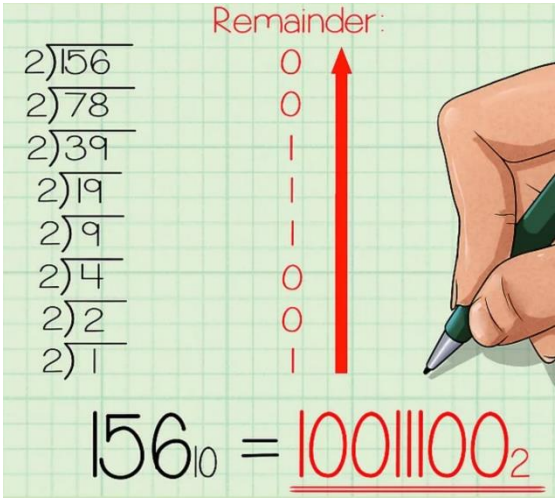
例子：4位二进制 $A = a_3a_2a_1a_0$ 对应:

$$a_3.2^3 + a_2.2^2 + a_1.2^1 + a_0.2^0 = a_3.8 + a_2.4 + a_1.2 + a_0.1$$

$$(a_i = 0 \text{ 、 } 1)$$

2、十进制→二进制

余数短除法除以二



十六进制表示

- **Base 16 (hexadecimal)**
  - **More a convenience for us humans than a true computer data type**
  - **0 to 9 represented as such**
  - **10, 11, 12, 13, 14, 15 represented by A, B, C, D, E, F**
  - **16 = 2^4: i.e. every hexadecimal digit can be represented by a 4-bit binary (unsigned) and vice-versa.**

Example

$$16AB_{16} = 0x16AB$$
$$= 1x16^3 + 6x16^2 + 10x16 + 11$$
$$= 5803_{10} = \#5803$$
$$= 0001\ 0110\ 1010\ 1011_2$$

典型长度整数的范围

数	字长w			
	8	16	32	64
$UMax_w$	0xFF 255	0xFFFF 65 535	0xFFFFFFFF 4 294 967 295	0xFFFFFFFFFFFFFFFF 18 446 744 073 709 551 615
$TMin_w$	0x80 −128	0x8000 −32 768	0x80000000 −2 147 483 648	0x8000000000000000 −9 223 372 036 854 775 808
$TMax_w$	0x7F 127	0x7FFF 32 767	0x7FFFFFFF 2 147 483 647	0x7FFFFFFFFFFFFFFF 9 223 372 036 854 775 807
−1	0xFF	0xFFFF	0xFFFFFFFF	0xFFFFFFFFFFFFFFFF
0	0x00	0x0000	0x00000000	0x0000000000000000

图 2-14　重要的数字。图中给出了数值和十六进制表示

设置

-200，用16位二进制补码表示为（）

A -200

B 1000 0000 1100 1000

C 1111 1111 0011 1000

D 1011 1000

E 以上都不对

提交