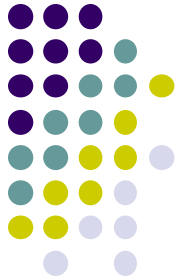


计算机组成与系统结构

期末复习（1）

吕昕晨

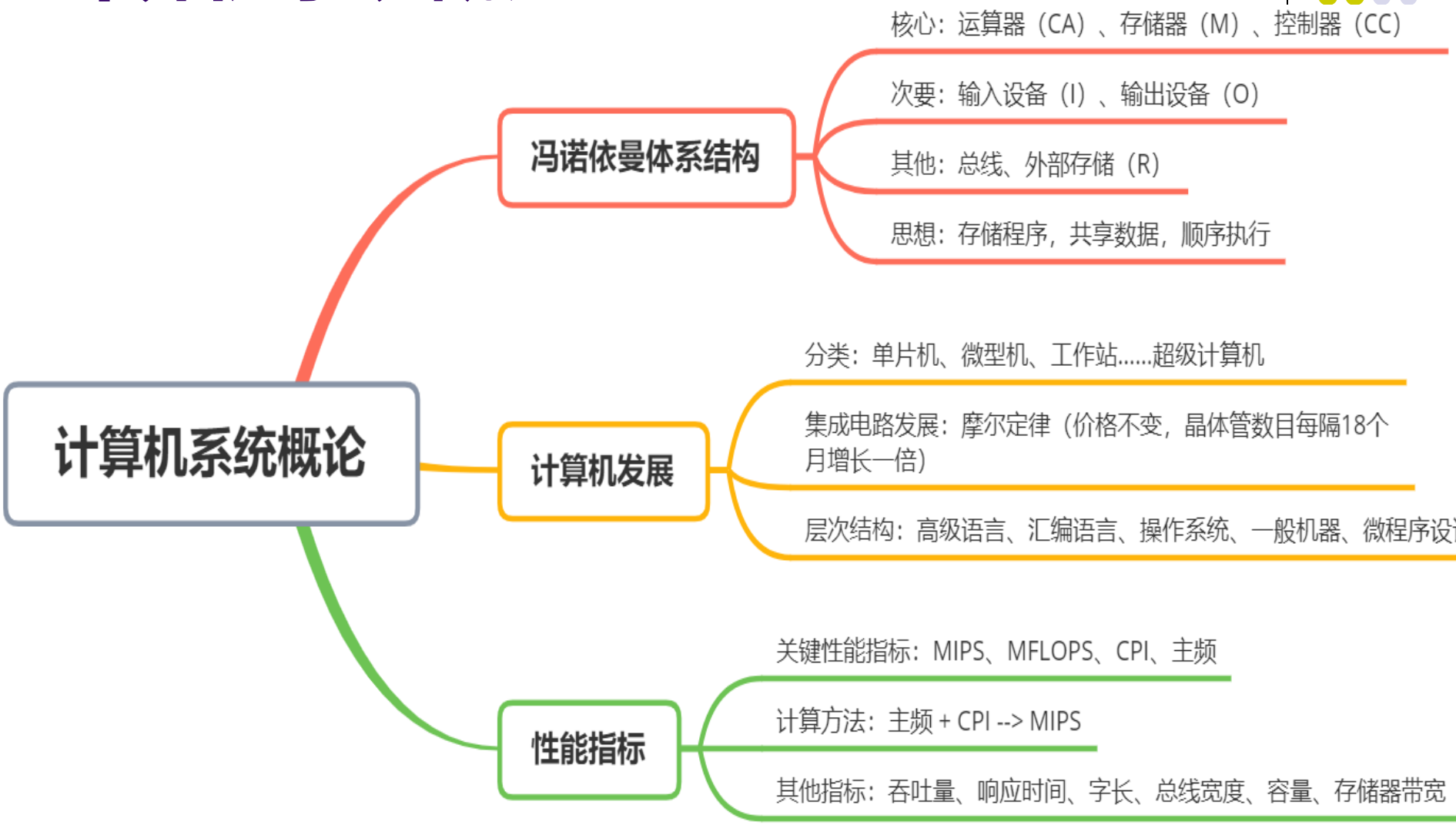
lvxinchen@bupt.edu.cn

网络空间安全学院

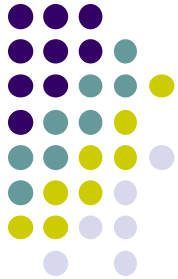
期末复习

- 第一章 计算机系统概论
  - 关键性能指标计算
- 第二章 运算方法与运算器
- 第三章 多层次存储器
- 第四章 指令系统

计算机系统概论


计算机系统概论
冯诺依曼体系结构
核心：运算器（CA）、存储器（M）、控制器（CC）
次要：输入设备（I）、输出设备（O）
其他：总线、外部存储（R）
思想：存储程序，共享数据，顺序执行
计算机发展
分类：单片机、微型机、工作站.....超级计算机
集成电路发展：摩尔定律（价格不变，晶体管数目每隔18个月增长一倍）
层次结构：高级语言、汇编语言、操作系统、一般机器、微程序设计
性能指标
关键性能指标：MIPS、MFLOPS、CPI、主频
计算方法：主频 + CPI --> MIPS
其他指标：吞吐量、响应时间、字长、总线宽度、容量、存储器带宽

计算机性能指标（1）



- 吞吐量

表征一台计算机在某一时间间隔内能够处理的信息量，单位是字节/秒（B/S）

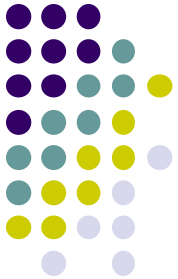
- 响应时间

表征从输入有效到系统产生响应之间的时间度量，用时间单位来度量，例如微秒（10^{-6}s）、纳秒（10^{-9}s）。

- 主频/时钟周期

CPU的工作节拍受主时钟控制，主时钟不断产生固定频率的时钟，主时钟的频率叫CPU的主频（f）。度量单位是MHz、GHz。例如Pentium系列机为60MHz～266MHz，而Pentium 4升至3.6GHz。

计算机的性能指标（2）

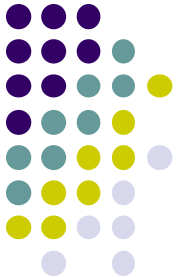


- 主频的倒数称为CPU时钟周期（T），即T=1/f，度量单位是纳秒
- CPI（Cycles per Instruction）表示每条指令周期数，即执行一条指令所需的平均时钟周期数：

$$CPI = \frac{\text{执行某段程序所需的CPU时钟周期数}}{\text{该程序包含的指令条数}}$$

- MIPS（Million Instructions per Second）表示每秒百万条（10^6）指令
- MFLOPS（Million Floating-point Operations per Second）表示每秒百万次浮点操作次数
  - MIPS是单位时间内的执行指令数，所以MIPS值越高说明机器速度越快
  - MFLOPS是基于操作而非指令的，用来衡量机器浮点操作的性能
- 程序执行时间Te = $\frac{\text{指令条数}}{\text{MIPS} \times 10^6}$

计算机性能指标（3）



- 处理机字长

指处理机运算器中一次能够完成二进制数运算的位数。当前处理机的字长有8位、16位、32位、64位。字长越长，表示计算的精度越高

- 总线宽度

一般指CPU中运算器与存储器之间进行互连的内部总线二进制位数。

- 存储器容量

存储器中所有存储单元的总数目，通常用KB、MB、GB、TB来表示。K=2^{10}，M=2^{20}，G=2^{30}，T=2^{40}，B=8位（1个字节）。

- 存储器带宽

存储器的速度指标，单位时间内从存储器读出的二进制数信息量，一般用字节数/秒表示。

关键性能指标计算习题

某计算机主频为1GHZ，共有A、B、C三类指令，每类指令的CPI分别为1、2、3。
现使用两个不同的编译器分别生成一大段测试软件的二进制代码
1）编译器1生成的代码含五百万条A类指令，一百万条B类指令和一百万条C类指令；
2）编译器2生成的代码还有10百万条A类指令，一百万条B类指令，一百万条C类指令。
问:
（1）根据MIPS计算，哪个编译器生成的代码执行速度更快?
（2）根据执行时间计算，哪个编译器生成的代码执行速度快?
解题思路:根据题中所给的条件，分别求出两种编译器得到代码的MIPS和执行一条指令需要的平均时间进行比较。

关键性能指标计算习题（1）

某计算机主频为1GHZ，共有A、B、C三类指令，每类指令的CPI分别为1、2、3。
现使用两个不同的编译器分别生成一大段测试软件的二进制代码
1）编译器1生成的代码含五百万条A类指令，一百万条B类指令和一百万条C类指令；
2）编译器2生成的代码还有10百万条A类指令，一百万条B类指令，一百万条C类指令。
问:
（1）根据MIPS计算，哪个编译器生成的代码执行速度更快?

$$MIPS_1 = \frac{f}{CPI * 10^6} = \frac{10^9}{(\frac{5}{7}*1+\frac{1}{7}*2+\frac{1}{7}*3)*10^6} = 700$$

$$MIPS_2 = \frac{10^9}{(\frac{10}{12}*1+\frac{1}{12}*2+\frac{1}{12}*3)*10^6} = 800$$

单从MIPS来看，编译器2编译出的代码具有较快的执行速度。

关键性能指标计算习题（2）

某计算机主频为1GHZ，共有A、B、C三类指令，每类指令的CPI分别为1、2、3。

现使用两个不同的编译器分别生成一大段测试软件的二进制代码

1）编译器1生成的代码含五百万条A类指令，一百万条B类指令和一百万条C类指令；

2）编译器2生成的代码还有10百万条A类指令，一百万条B类指令，一百万条C类指令。

问:

（2）根据执行时间计算，哪个编译器生成的代码执行速度快?

$Tcpu_1 = 10^{-9} * (5*1+2*1+3*1)*10^6 = 0.01$ s

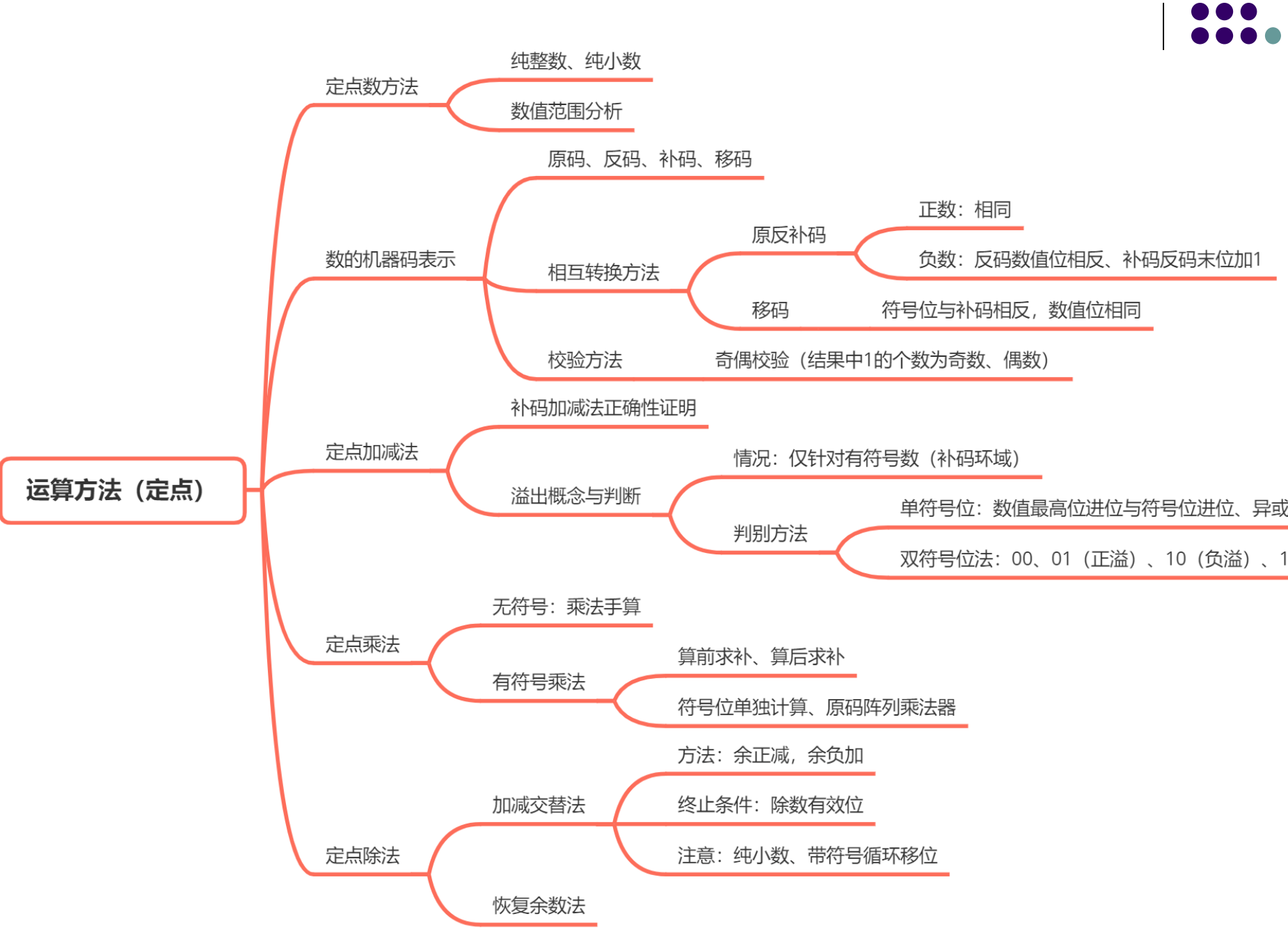
$Tcpu_2 = 10^{-9} * (10*1+2*1+3*1)*10^6 = 0.015$ s

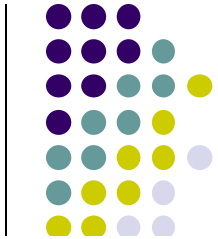
从CPU执行时间看，编译器1编译出来的代码执行速度快。

期末复习

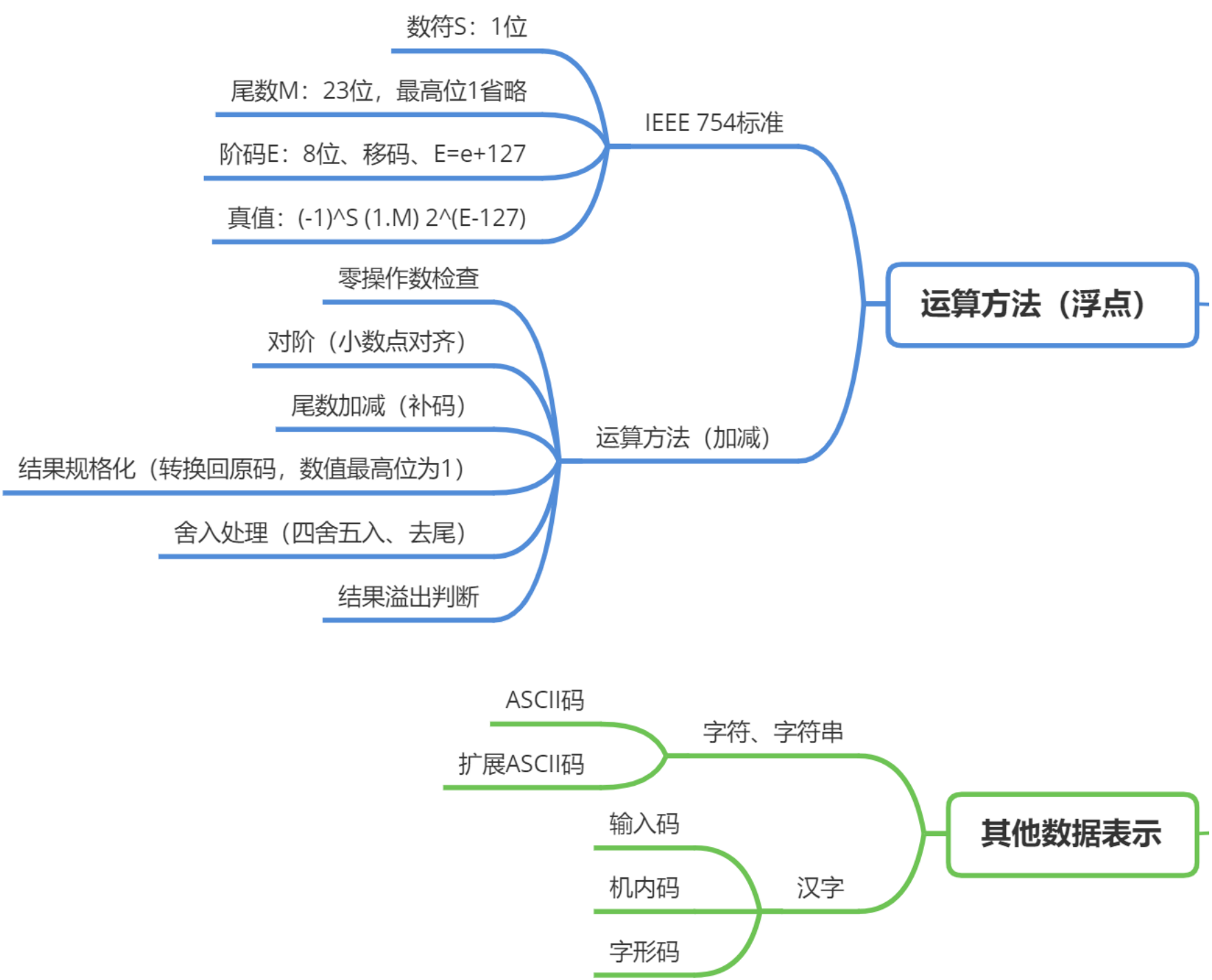
- 第一章 计算机系统概论
- 第二章 运算方法与运算器
  - 加减法器与溢出判断
  - 定点数乘除法
  - 浮点数加减法
- 第三章 多层次存储器
- 第四章 指令系统

运算方法与运算器


运算方法（定点）
定点数方法
纯整数、纯小数
数值范围分析
数的机器码表示
原码、反码、补码、移码
相互转换方法
原反补码
正数：相同
负数：反码数值位相反、补码反码末位加1
移码
符号位与补码相反，数值位相同
校验方法
奇偶校验（结果中1的个数为奇数、偶数）
定点加减法
补码加减法正确性证明
溢出概念与判断
情况：仅针对有符号数（补码环域）
判别方法
单符号位：数值最高位进位与符号位进位、异或
双符号位法：00、01（正溢）、10（负溢）、11
定点乘法
无符号：乘法手算
有符号乘法
算前求补、算后求补
符号位单独计算、原码阵列乘法器
定点除法
加减交替法
方法：余正减，余负加
终止条件：除数有效位
注意：纯小数、带符号循环移位
恢复余数法



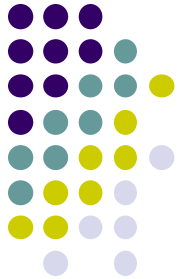

运算器（定点）
逻辑运算
算术运算（加减法器）
半加器、全加器
行波进位、超前进位（信号产生）
时延分析：关键路径
加减法统一：补码（异或）
溢出判断：异或（单符号位）
ALU
函数信号发生器、74181算数逻辑单元
内部总线方式：单总线、双总线、三总线
定点乘除法器
串行乘法、阵列乘法电路
对二求补电路：（负数）从右往左扫描，最低位1左侧（除符号位）全部取反
加减可控单元（CAS）、并行除法器


运算方法（浮点）
IEEE 754标准
数符S：1位
尾数M：23位，最高位1省略
阶码E：8位、移码、E=e+127
真值：(-1)^S (1.M) 2^(E-127)
运算方法（加减）
零操作数检查
对阶（小数点对齐）
尾数加减（补码）
结果规格化（转换回原码，数值最高位为1）
舍入处理（四舍五入、去尾）
结果溢出判断
其他数据表示
字符、字符串
ASCII码
扩展ASCII码
汉字
输入码
机内码
字形码

IEEE 754标准——32位浮点数

- 基数R=2，基数固定，采用隐含方式来表示它。
- 32位的浮点数：
  - S数的符号位，1位，在最高位，0表示正数，1表示负数
  - M是尾数， 23位，在低位部分，采用纯小数表示
  - E是阶码，8位，采用移码表示。移码比较大小方便
  - 注意：
    - 关于尾数：尾数域最左位(最高有效位)总是1，故这一位不予存储，而认为隐藏在小数点的左边。
    - 关于阶码：由于采用移码，将浮点数的指数真值e变成阶码E时，应将指数e加上一个固定的偏移值127(01111111)，即E=e+127。

IEEE 754标准——例题1



例1 若浮点数x的754标准存储格式为$(41360000)_{16}$，求其浮点数的十进制数值。

解：将16进制数展开后，可得二制数格式为

0　100/0001/0　011/0110/0000/0000/0000/0000

S　阶码(8位)　尾数(23位)

指数e=阶码-127=1000 0010 - 0111 1111=00000011=$(3)_{10}$

包括隐藏位1的尾数

1.M=1.011 0110 0000 0000 0000 0000=1.011011

于是有

$x=+(S==0)1.M\times 2^{e}=+(1.011011)\times 2^{3}=+1011.011=(11.375)_{10}$

IEEE 754标准——例题2

例2 将数$(20.59375)_{10}$转换成754标准的32位浮点数的二进制存储格式。

解:首先分别将整数和分数部分转换成二进制数:

20.59375=10100.10011

然后移动小数点，使其在第1，2位之间

10100.10011=1.010010011×2^4

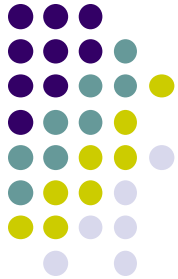
e=4于是得到: M

S=0, E=4+127=131, E=1000 0011

最后得到32位浮点数的二进制存储格式为:

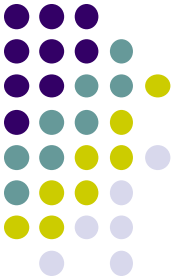
0 10000011 01001001100000000000=$(41A4C000)_{16}$

原码、反码、补码、移码（重要）



- 原码
  - 符号位加上真值的绝对值
- 反码
  - 正数的反码是其本身
  - 负数的反码是在其原码的基础上，符号位不变，其余各位取反
- 补码
  - 正数的补码就是其本身
  - 负数的补码是在反码的基础上+1
- 移码
  - 补码的符号位取反（无论正负）

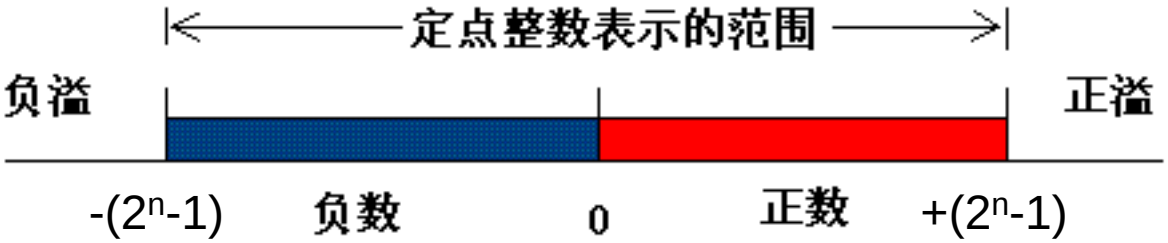
[例]将十进制真值(- 127, - 1,0, + 1, + 127)列表表示成二进制数及原码、反码、补码、移码值。

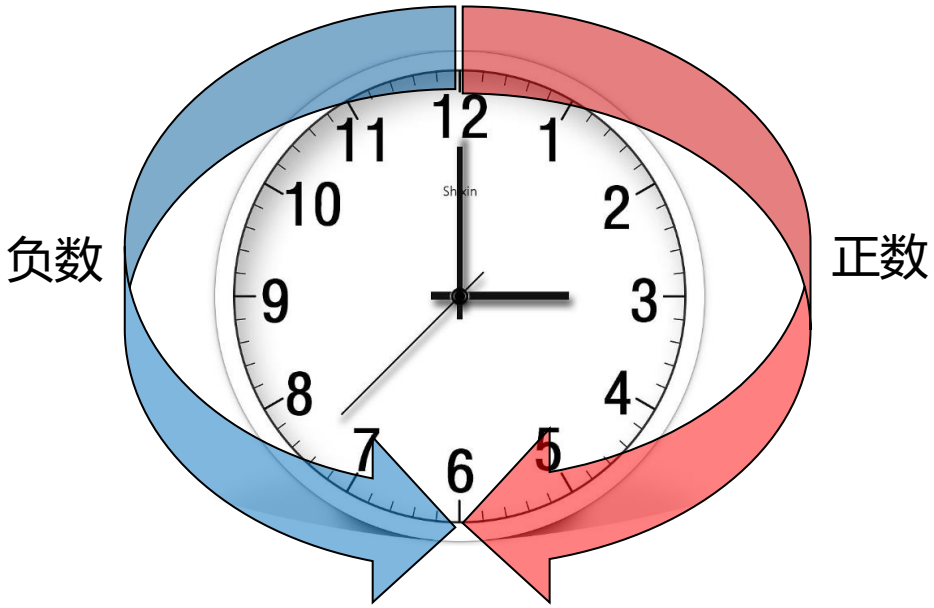


真值x（十进制）	真值x（二进制）	[x]原	[x]反	[x]补	[x]移
-127	-01111111	11111111	10000000	10000001	00000001
-1	-00000001	10000001	11111110	11111111	01111111
		00000000	00000000		
0	00000000			00000000	10000000
		10000000	11111111		
+1	+00000001	00000001	00000001	00000001	10000001
+127	+01111111	01111111	01111111	01111111	11111111

溢出概念

溢出的概念





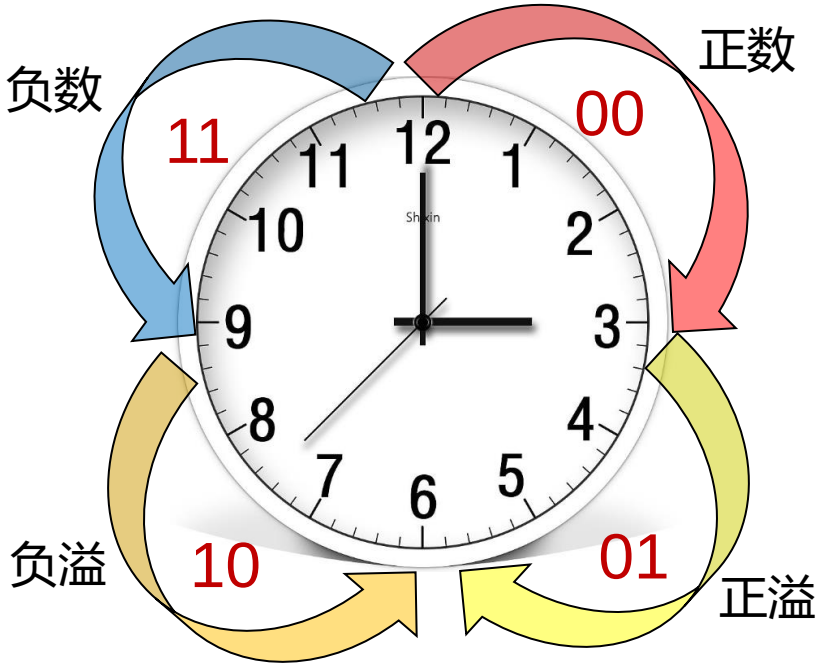
双符号位溢出检测——手算过程

1、双符号位法

（变形补码—扩大一倍范围）

$[x]_{补} = 2^{n+2} + x \quad (\bmod 2^{n+2})$

S_{f1}	S_{f2}	
0	0	正确（正数）
0	1	正溢
1	0	负溢
1	1	正确（负数）



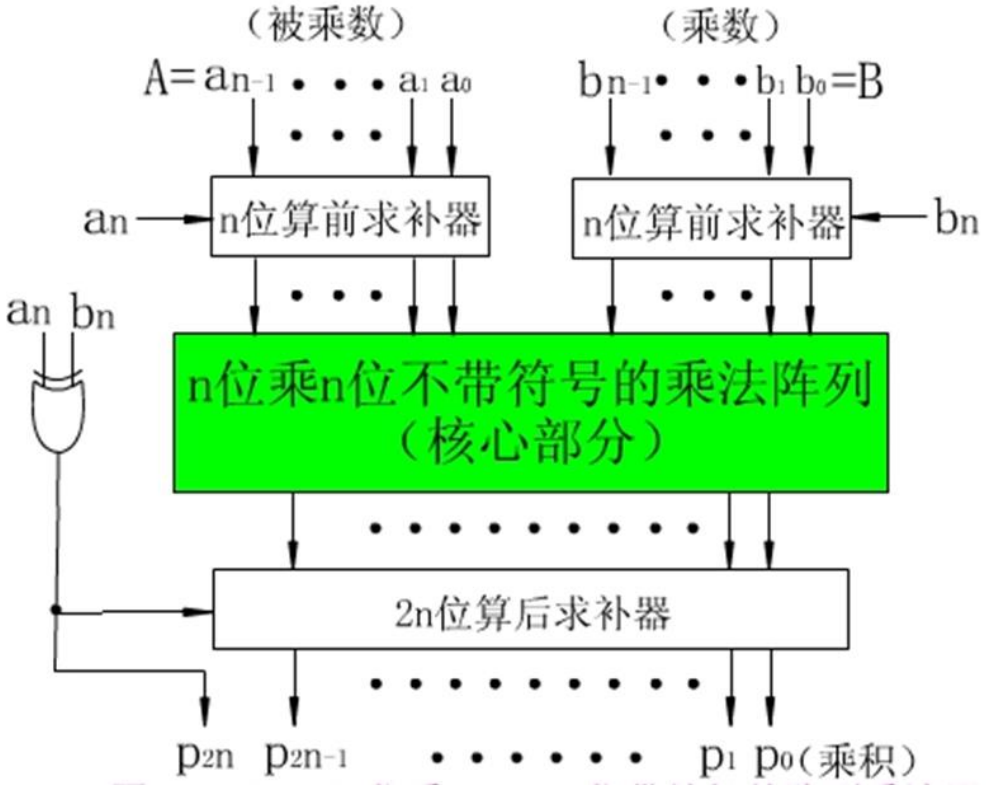
S_{f1} 表示正确的符号，逻辑表达式为$V=S_{f1} \oplus S_{f2}$，可以用异或门来实现

期末复习

- 第一章 计算机系统概论
- 第二章 运算方法与运算器
  - 加减法器与溢出判断
  - 定点数乘除法
  - 浮点数加减法
- 第三章 多层次存储器
- 第四章 指令系统

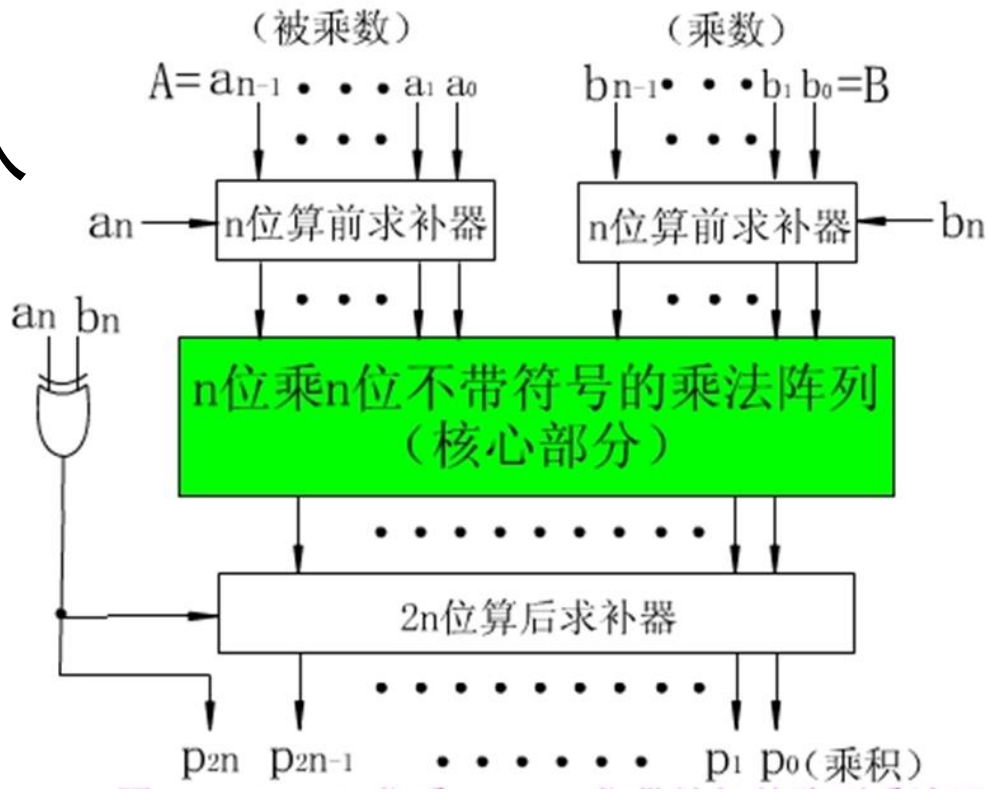
带符号乘法器——总结1

- 补码性质
  - $[\ [A]_{补}\ ]_{补} = [\ A\ ]_{原}$
- 带符号乘法器构思路
  - 算前求补（输入补码）
  - 乘法器
  - 算后求补（输出补码）
- 补码转换性质
  - 最右端往左边扫描，直到第一个1的时候，该位和右边各位保持不变，左边各数值位按位取反（扫描）



带符号乘法器——总结2

- 求解步骤
  - 判断原码/补码输入
  - 符号位计算
  - （算前求补）
  - 乘法（不带符号）
  - （算后求补）

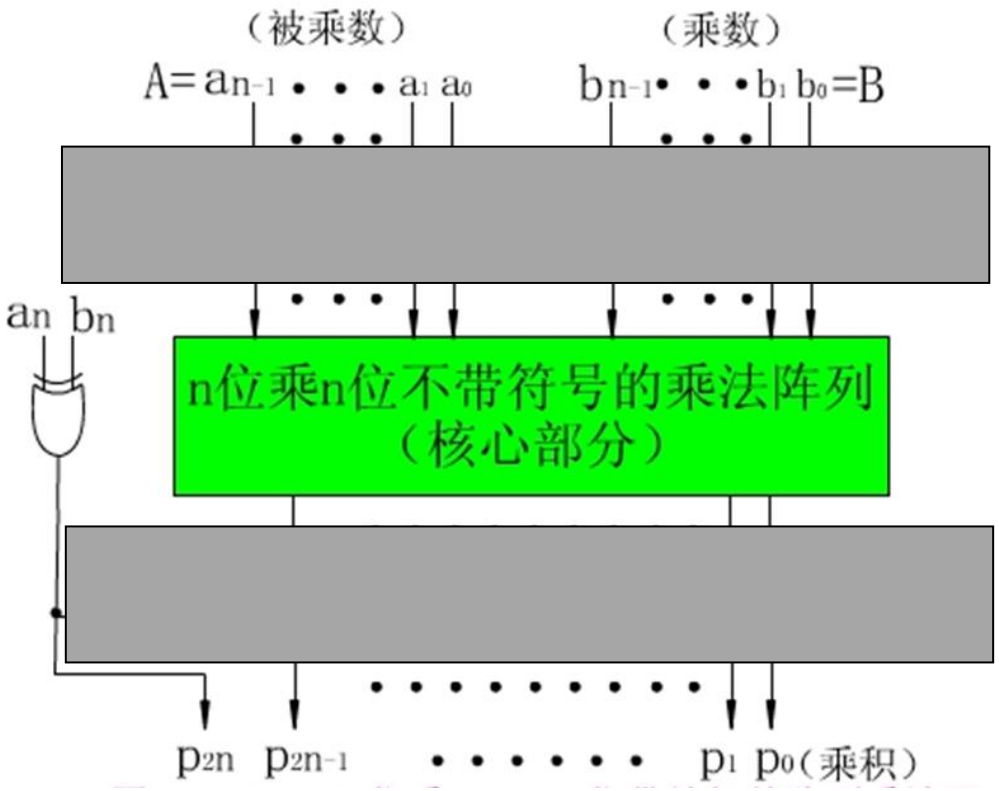


习题[2-7-1)] 已知x=11011，y=-11111
用原码阵列乘法器、补码阵列乘法器，计算x×y？（重点）

- 原码乘法（输入原码、输出原码）
- 符号位：0⊕1=1
- 无需算前算后求补
- |x|=11011，|y|=11111
- 乘法:

```
          1 1 0 1 1
    ×     1 1 1 1 1
    -----------------
          1 1 0 1 1
        1 1 0 1 1
      1 1 0 1 1
    1 1 0 1 1
+ 1 1 0 1 1
-----------------
  1 1 0 1 0 0 0 1 0 1
```

- $[x \times y]_{原}$ = 1 1101000101

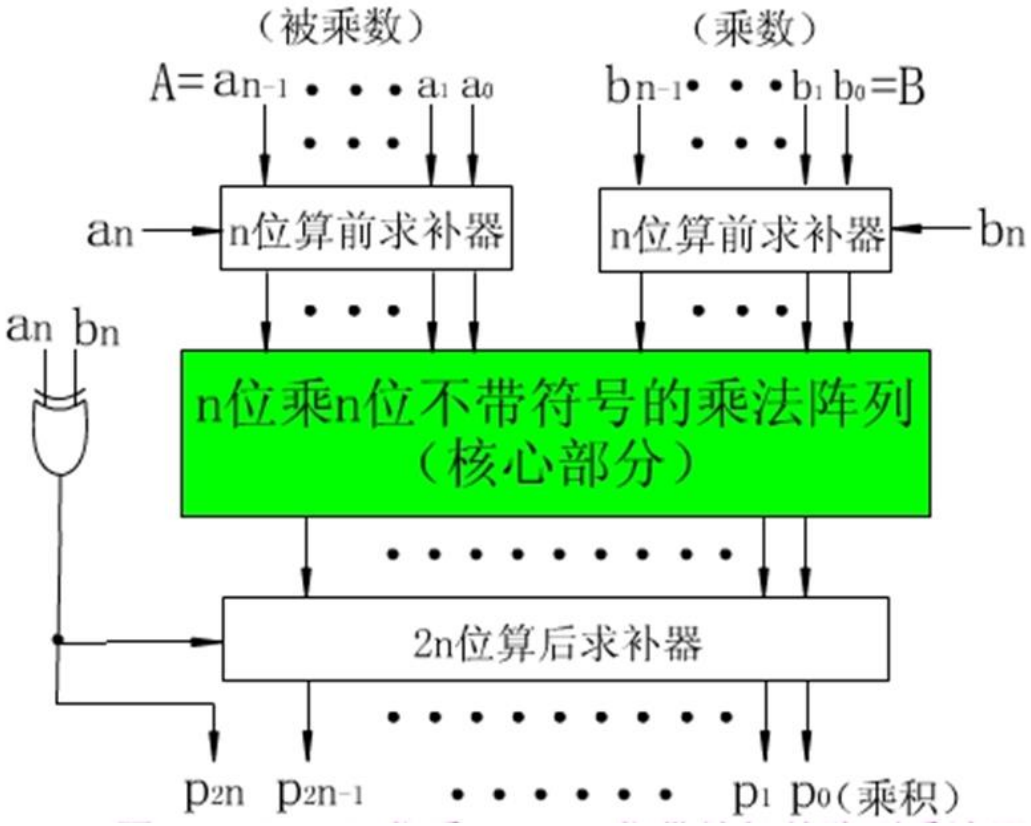


习题[2-7-1)] 已知x=11011，y=-11111
用原码阵列乘法器、补码阵列乘法器，计算x×y?

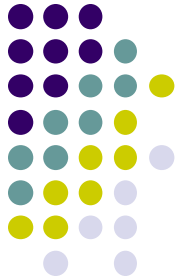
- 补码乘法（输入补码、输出补码）
- [x]补=0 11011，[y]补=100001
- 符号位：0⊕1=1
- 算前求补
- |x|=11011，|y|=11111
- 乘法:

```
          1 1 0 1 1
        × 1 1 1 1 1
        -------------
          1 1 0 1 1
        1 1 0 1 1
      1 1 0 1 1
    1 1 0 1 1
  1 1 0 1 1
+
  1 1 0 1 0 0 0 1 0 1
```

- $[x \times y]_{原}$ = 1 1101000101
- 算后求补 $[x \times y]_{补}$ = 1 0010111011



计算机除法流程



- 人工除法时，人可以比较被除数（余数）和除数的大小来确定商1（够减）或商0（不够减）
- 机器除法时，余数为正表示够减，余数为负表示不够减。不够减时必须恢复原来余数，才能继续向下运算。这种方法叫恢复余数法，控制比较复杂。
- 不恢复余数法（加减交替法——重点）
  - 余数为正，商1，下次除数右移做减法
  - 余数为负，商0，下次除数右移做加法

补码除法流程——加减交替法

[例23] $x = 0.101001,\ y = 0.111$, 求 $x \div y$。

[解:] $[x]_{补}$=**0.101001** , $[y]_{补}$=**0.111** , $[-y]_{补}$=**1.001**

起始位置		0.1 0 1 0 0 1	；	被除数
	+ $[-y]_{补}$	1.0 0 1	小数点后3位 y=0.111 ；	第一步减除数y
		1.1 1 0 0 0 1	<0 q_0=0 ；	余数为负,商0
	+ $[y]_{补}\rightarrow$	0.0 1 1 1	；	除数右移1位加
		0.0 0 1 1 0 1	>0 q_1=1 ；	余数为正,商1
	+ $[-y]_{补}\rightarrow$	1.1 1 0 0 1	；	除数右移2位减
		1.1 1 1 1 1 1	<0 q_2=0 ；	余数为负,商0
	+ $[y]_{补}\rightarrow$	0.0 0 0 1 1 1	；	除数右移3位加
	补码	0.0 0 0 1 1 0	>0 q_3=1 ；	余数为正,商1

真值

商q=$q_0.q_1q_2q_3$=0.101 , 余数r=0.000110

习题[2-8-1)] x = 11000，y = -11111, 用原码除法，求 $x \div y$。

- 符号位： 0⊕1=1
- |x|=11000，|y|=11111　　小数点后5位
- 纯小数表示，小数点左移5位，|x|=0.11000，|y|=0.11111
- [| x |]=0.11000, [| y |]补=0.11111 , [- | y |]补=1.00001

0.1 1 0 0 0　　；被除数

+ $[-y]_{补}$　1. 0 0 0 0 1　　；第一步减除数y

1.1 1 0 0 1　　<0　q_0=0　；余数为负,商0

+ $[y]_{补}\rightarrow$　0.0 1 1 1 1 1　　；除数右移1位加

0.0 1 0 0 0 1　　>0 q_1=1　；余数为正,商1

+ $[-y]_{补}\rightarrow$　1.1 1 0 0 0 0 1　　；除数右移2位减

0.0 0 0 0 0 1 1　>0 q_2=1　；余数为正, 商1

+ $[y]_{补}\rightarrow$　1.1 1 1 0 0 0 0 1　　；除数右移3位减

1.1 1 1 0 0 1 1 1 <0 q3=0 ；余数为负, 商0

习题[2-8-1)] $x = 11000$, $y = -11111$, 用原码除法, 求 $x \div y$。

1.1 1 1 0 0 1 1 1 <0 q3=0 ； 余数为负, 商0

+ $[y]_{补}\rightarrow$ 0.0 0 0 0 1 1 1 1 1 ； 除数右移4位加

1.1 1 1 1 0 1 1 0 1 <0 q4=0 ； 余数为负, 商0

+ $[y]_{补}\rightarrow$ 0.0 0 0 0 0 1 1 1 1 1 ； 除数右移5位加

---

1. 1 1 1 1 1 1 1 0 0 1 <0 q5=0 ； 余数为负，商0

- 商真值$|x \div y|$=0.11000，原码除法 $[x \div y]_{原}$=1.11000
- 余数：0.0000011
- 小数点右移5位（补偿）：0.11

0.0 1 0 0 0 1 >0 q_1=1 ； 余数为正,商1

+$[-y]_{补}\rightarrow$ 1. 1 1 0 0 0 0 1 ； 除数右移2位减

0.0 0 0 0 0 1 1 >0 q_2=1 ； 余数为正, 商1

0.0 0 0 0 0 1 1 >0 q_2=1 ； 余数为正, 商1

q0=0,
q1=1,
q2=1

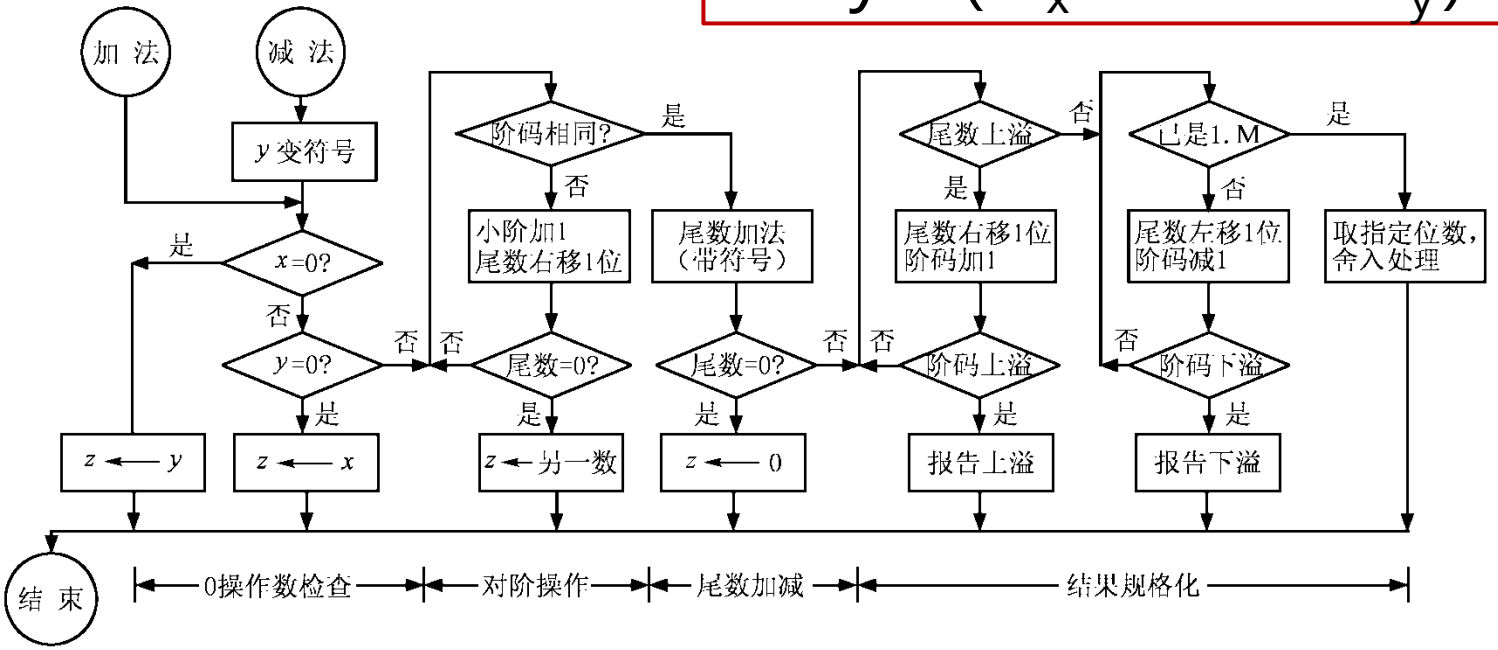
期末复习

- 第一章 计算机系统概论
- 第二章 运算方法与运算器
  - 加减法器与溢出判断
  - 定点数乘除法
  - 浮点数加减法
- 第三章 多层次存储器
- 第四章 指令系统

浮点加减法运算运算步骤

- 操作数检查（避免无效操作）；
- 比较阶码并完成对阶（小阶向大阶对齐）；
- 尾数加减运算;
- 结果规格化（溢出/规范化）；
- 舍入处理

$$x \pm y = (M_x 2^{Ex - Ey} \pm M_y) 2^{Ey}$$



浮点加减法示例——补码（1）

[例] 设$x=2^{010}\times0.11011011$,$y=-2^{100}\times0.10101100$，求x+y

（尾数用补码存储，尾数精度为8位）

1、0操作数检查（非0）

2、对阶：阶码对齐后才能加减。规则是阶码小的向阶码大的数对齐；

- 若△E＝0，表示两数阶码相等,即E x ＝E y；
- 若△E>0，表示E x >E y；
- 若△E<0，表示E x >E y 。
- 当E x ≠E y 时，要通过尾数的移动以改变E x 或E y ，使之相等。

- $[x]_{浮}$　Ex=00010，Mx=0.11011011
- $[y]_{浮}$　Ey=00100，My=1.01010100（补码，取反加一）；
- 阶差为-2，Mx右移两位，Ex加2。
- $[x]_{浮,对阶}$　Ex=00100，Mx=0.00110110(11)，尾数精度8位

浮点加减法示例——补码（2）

3、尾数相加（带符号位）

- 0. 0 0 1 1 0 1 1 0 (11)

  \+ 1. 0 1 0 1 0 1 0 0

  ---

  1. 1 0 0 0 1 0 1 0 (11)

4、规格化

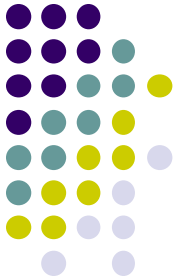
- 纯小数格式应为 (+/-) 0.1XXXXX
- 规则
  - 尾数右移1位，阶码加1
  - 尾数左移1位，阶码减1
- 左规处理，结果为1.00010101(10)，阶码为00011

$0.01*10^5=0.1*10^4$

正数1前；负数1后

Ex=00100

浮点加减法示例——补码（3）



- 舍入处理（对阶和向右规格化时）
  - 就近舍入(0舍1入)：类似”四舍五入”,丢弃的最高位为1，进1
  - 朝0舍入：截尾
  - 朝 + ∞舍入：正数多余位不全为0，进1；负数，截尾
  - 朝 - ∞ 舍入：负数多余位不全为0，进1；正数，截尾

  采用0舍1入法处理，得到1.00010110。 1.00010101(10)
- 溢出判断和处理
  - 阶码上溢，一般将其认为是 + ∞和 - ∞ 。
  - 阶码下溢，则数值为0。

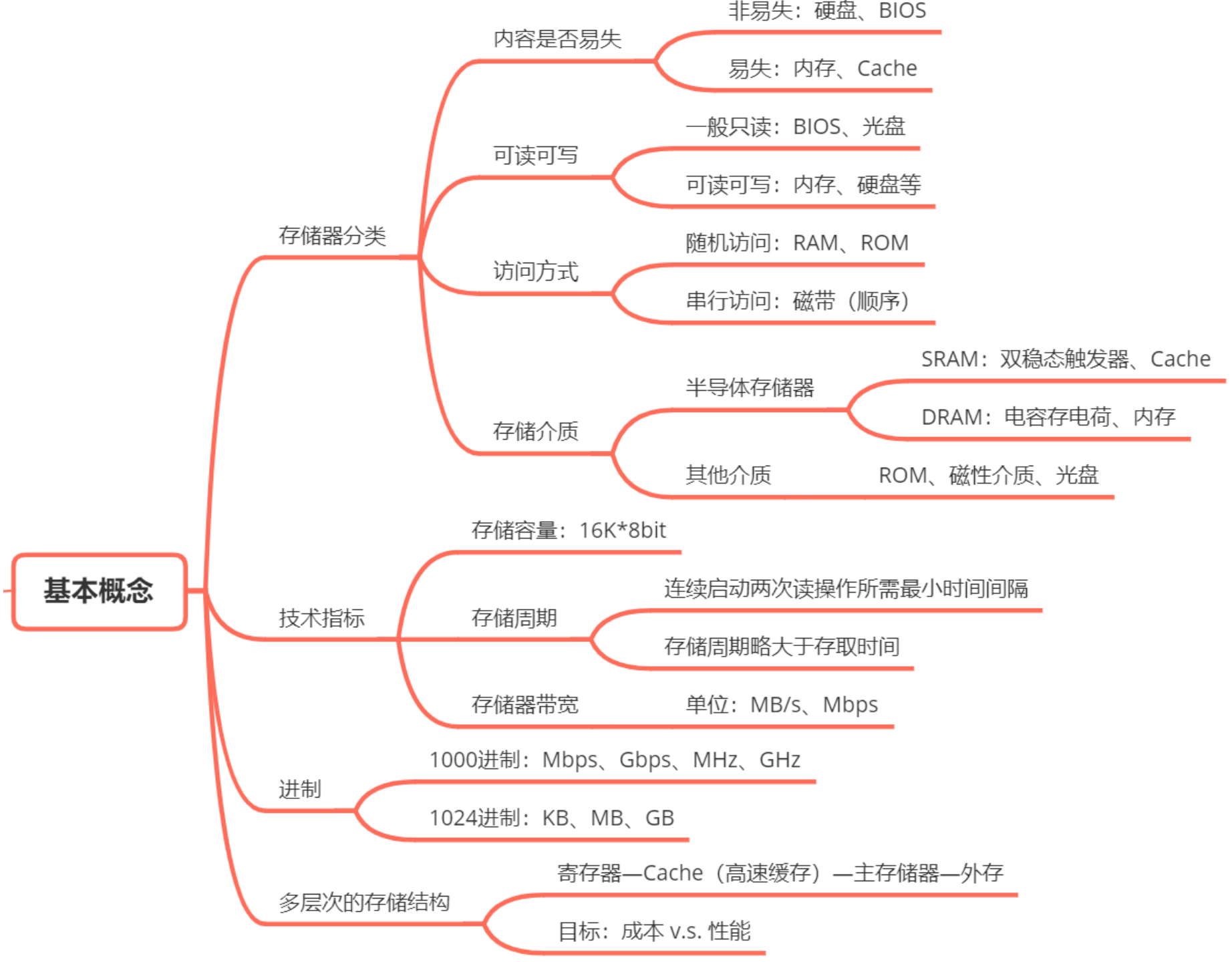
  阶码符号位为00，不溢出。得最终结果为

$$x+y = 2^{011} \times (-0.11101010)$$

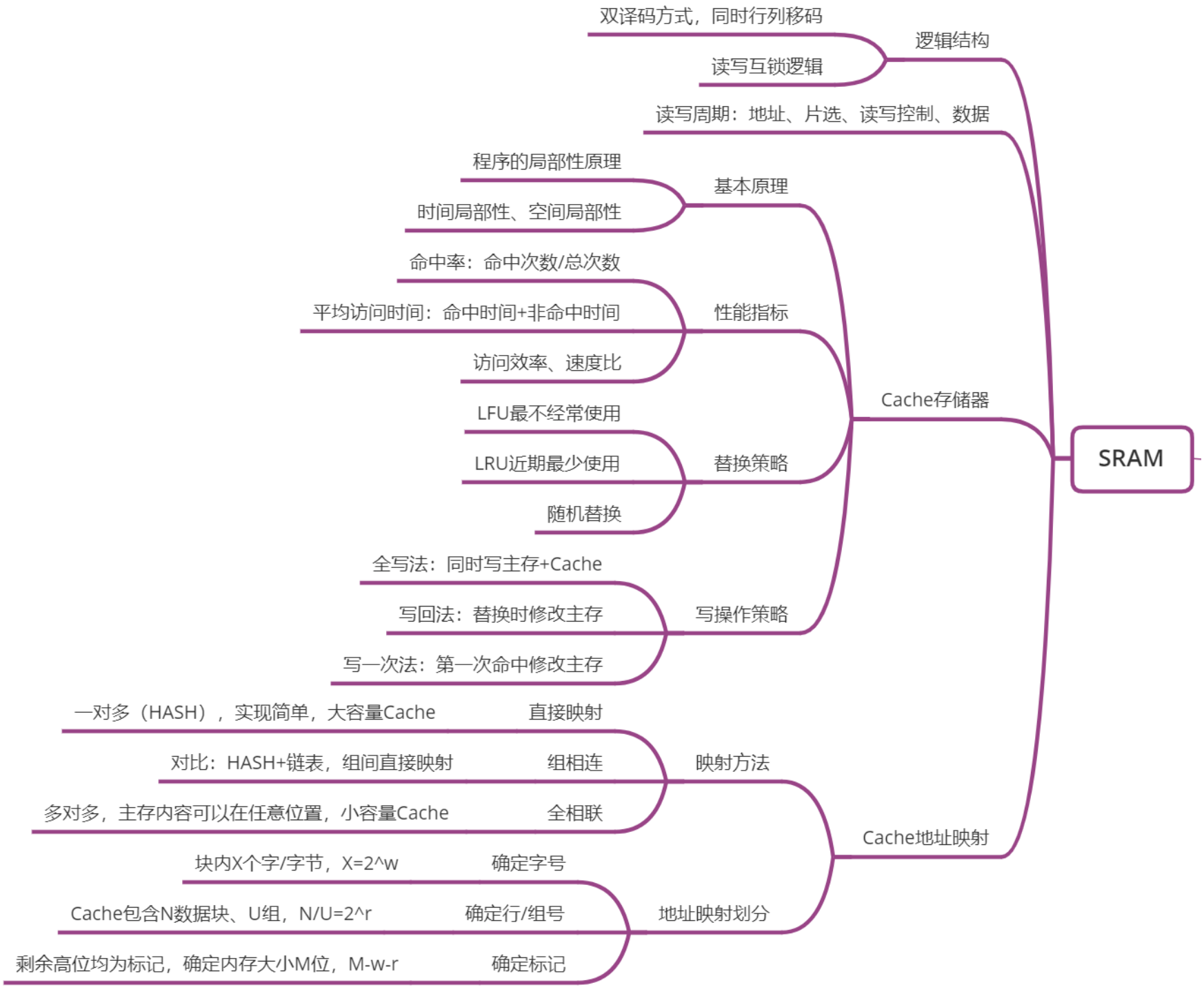
期末复习

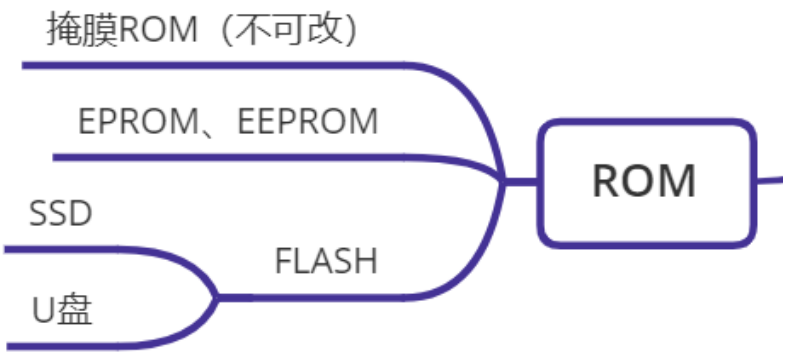
- 第一章 计算机系统概论
- 第二章 运算方法与运算器
- 第三章 多层次存储器
  - 存储器结构与扩展
  - Cache高速缓存映射
- 第四章 指令系统

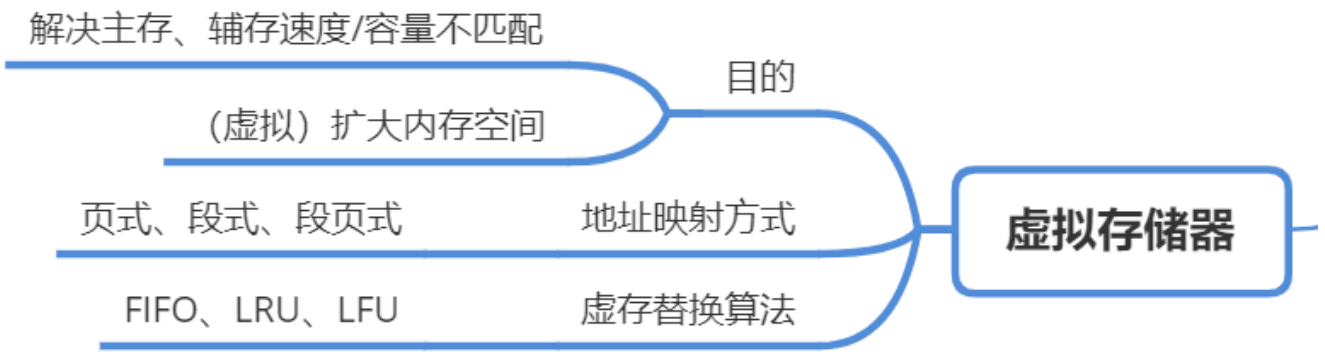
多层次的存储器

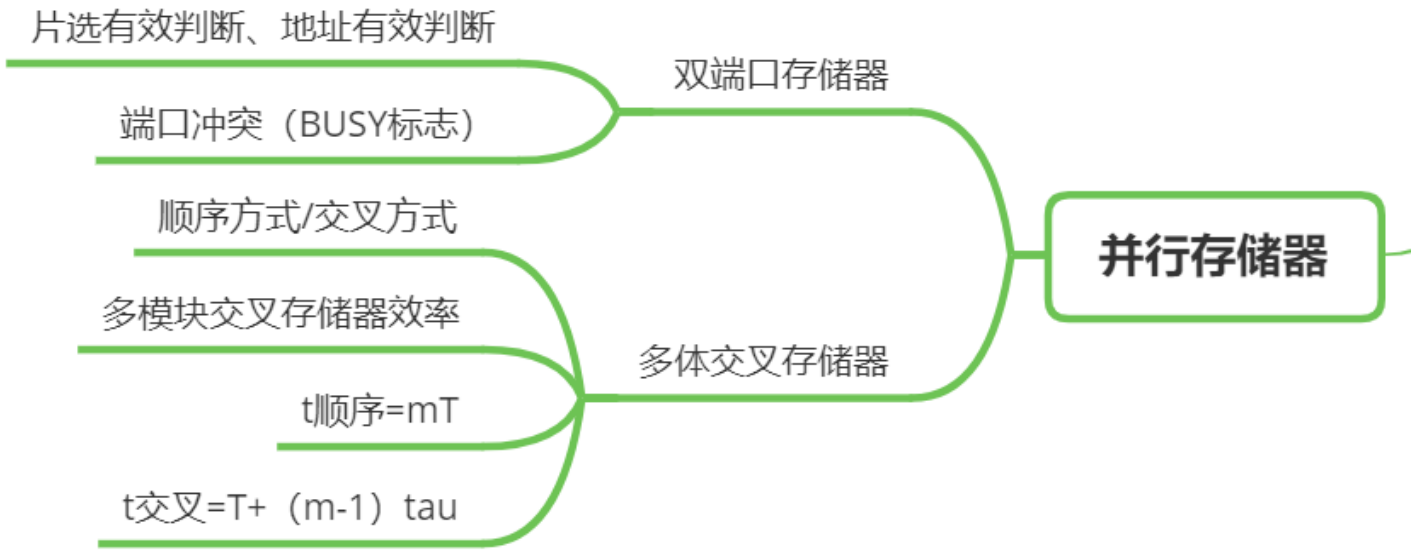

基本概念
存储器分类
内容是否易失
非易失：硬盘、BIOS
易失：内存、Cache
可读可写
一般只读：BIOS、光盘
可读可写：内存、硬盘等
访问方式
随机访问：RAM、ROM
串行访问：磁带（顺序）
存储介质
半导体存储器
SRAM：双稳态触发器、Cache
DRAM：电容存电荷、内存
其他介质
ROM、磁性介质、光盘
技术指标
存储容量：16K*8bit
存储周期
连续启动两次读操作所需最小时间间隔
存储周期略大于存取时间
存储器带宽
单位：MB/s、Mbps
进制
1000进制：Mbps、Gbps、MHz、GHz
1024进制：KB、MB、GB
多层次的存储结构
寄存器—Cache（高速缓存）—主存储器—外存
目标：成本 v.s. 性能


DRAM
逻辑结构
分时送地址码，扩大寻址范围
刷新计数器
输入输出缓冲放大器
读写时序
行地址、列地址分别使能
刷新
结构特点：按行刷新（一次一行）
刷新方法
集中刷新（连续对所有行进行刷新）
分散/异步刷新（刷新信号穿插）
刷新周期/行数、向下取整（按存储周期）
行数计算（单芯片）：2^17=2^9（行数）*2^8（列数）
存储容量扩展
位扩展
地址线、控制线公用，数据线分别连接至各芯片
多个存储芯片作为整体，当作一个新的存储体
字扩展
低位地址线、数据线公用；高位地址线用作译码（控制线端口）
多个芯片分时工作，每次仅一个芯片选中（控制）
混合扩展
高级DRAM方式
异步存储：FPM DRAM、CDRAM
同步存储：SDRAM、DDR


SRAM
逻辑结构
双译码方式，同时行列移码
读写互锁逻辑
读写周期：地址、片选、读写控制、数据
Cache存储器
基本原理
程序的局部性原理
时间局部性、空间局部性
性能指标
命中率：命中次数/总次数
平均访问时间：命中时间+非命中时间
访问效率、速度比
替换策略
LFU最不经常使用
LRU近期最少使用
随机替换
写操作策略
全写法：同时写主存+Cache
写回法：替换时修改主存
写一次法：第一次命中修改主存
Cache地址映射
映射方法
直接映射
一对多（HASH），实现简单，大容量Cache
组相连
对比：HASH+链表，组间直接映射
全相联
多对多，主存内容可以在任意位置，小容量Cache
地址映射划分
确定字号
块内X个字/字节，X=2^w
确定行/组号
Cache包含N数据块、U组，N/U=2^r
确定标记
剩余高位均为标记，确定内存大小M位，M-w-r


ROM
掩膜ROM（不可改）
EPROM、EEPROM
FLASH
SSD
U盘


虚拟存储器
目的
解决主存、辅存速度/容量不匹配
（虚拟）扩大内存空间
地址映射方式
页式、段式、段页式
虚存替换算法
FIFO、LRU、LFU


并行存储器
双端口存储器
片选有效判断、地址有效判断
端口冲突（BUSY标志）
多体交叉存储器
顺序方式/交叉方式
多模块交叉存储器效率
t顺序=mT
t交叉=T+（m-1）tau

字长位数扩展

- 字长扩展，即数据线扩展
  - 芯片8位，需求32位地址线
  - 多个芯片扩展字长位数
  - d=设计要求/芯片能力
- 设计方法
  - 共用地址线、控制线
  - 芯片数据线各位连接至数据线不同位

[例2] 利用1M×4位的SRAM芯片，设计一个存储容量为1M×8位的SRAM存储器。

- 位数不足，8位需求 v.s. 4位芯片
- 所需芯片数量=(1M×8)/(1M×4)=2片

存储容量扩展

- 存储容量扩展，即地址线扩展
  - 芯片1M（20位），需求2M（21位）
  - 多个芯片扩展地址线
  - d=设计要求/芯片能力
- 设计方法
  - 共用部分地址线（低20位）、控制线、数据线
  - 高位地址线用于生成芯片片选信号
    - 1位扩展（非门），多位扩展（译码器）

[例3]利用1M×8位的DRAM芯片设计2M×8位的DRAM存储器

- 容量不足：2M需求 v.s. 1M芯片
- 所需芯片数d=（2M×8）/（1M×8）=2(片)

习题3-4

- 由128K*8位的DRAM芯片构成1024K*32位存储器

1）总共需要多少DRAM芯片

2）画出存储器逻辑框图

3）存储器为读写周期0.5us，CPU在1us内至少访存一次，采用何种刷新方式?

4）刷新周期为8ms，刷新信号周期为?

习题3-4

- 由128K*8位的DRAM芯片构成1024K*32位存储器

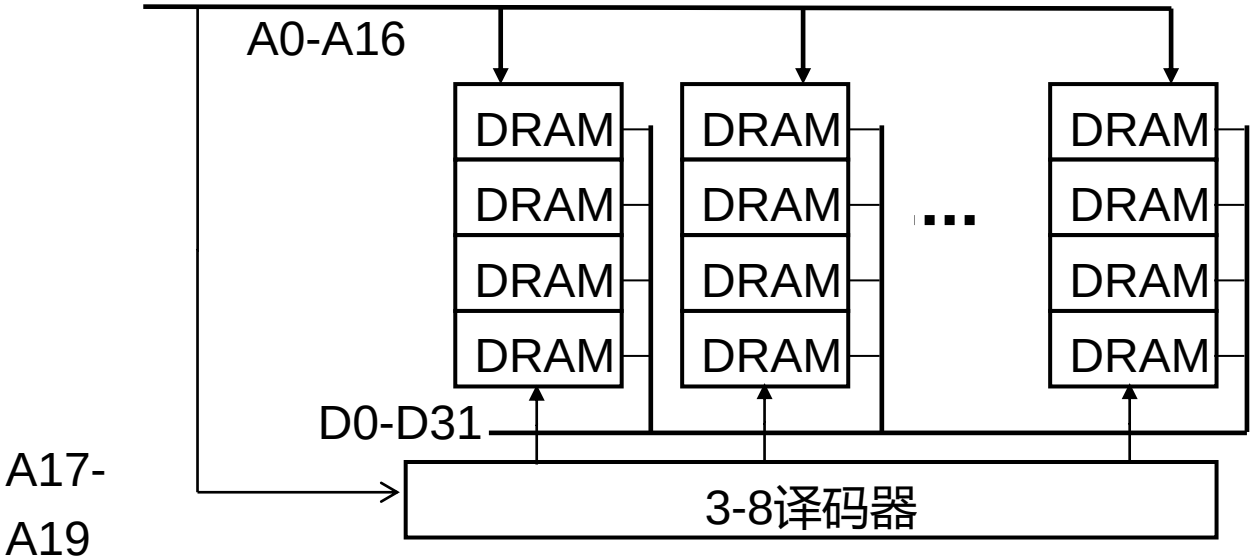
1）总共需要多少DRAM芯片

字长扩展 32/8=4 （4个DRAM组成32位）

容量扩展 1M/128K=8 （17位 20位，3-8译码器）

需要4*8=32片

2）画出存储器逻辑框图



习题3-4

- 由128K*8位的DRAM芯片构成1024K*32位存储器

3）存储器为读写周期0.5us，CPU在1us内至少访存一次，采用何种刷新方式?

假设存储器芯片为512*256*8bit（17位=9位+8位）

集中式刷新：0.5us*512=256us >> 1us，不可行

采用分散式刷新

4）刷新周期为8ms，刷新信号周期为?

刷新信号周期=刷新周期/行数

=8ms/512

=15.625us

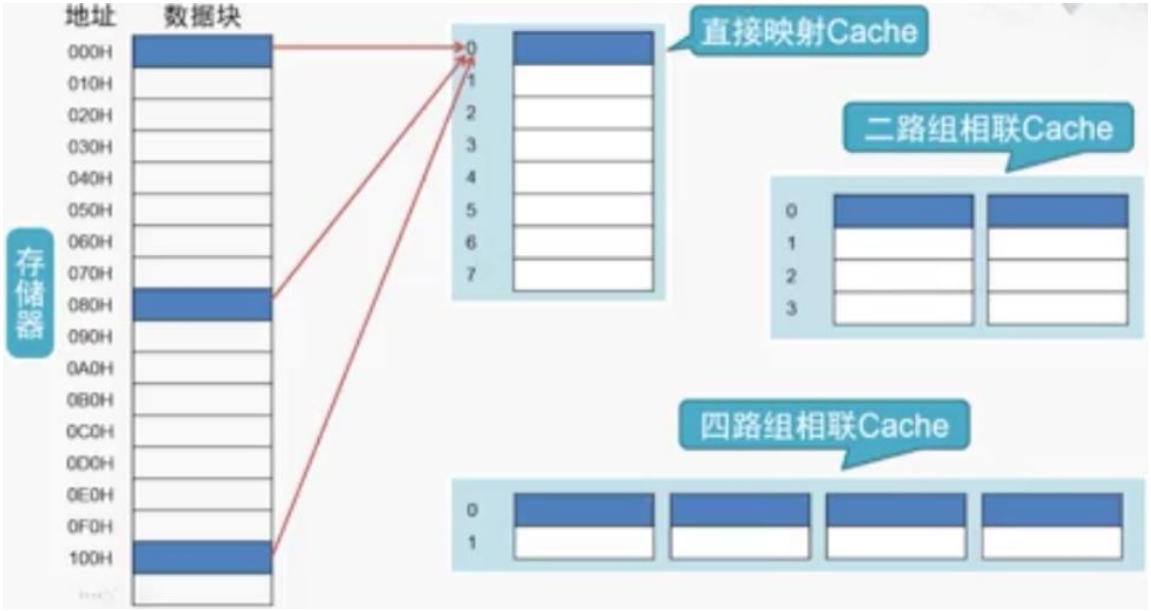
=15.5 us （以读写周期向下取整）

期末复习

- 第一章 计算机系统概论
- 第二章 运算方法与运算器
- 第三章 多层次存储器
  - 存储器结构与扩展
  - Cache高速缓存映射
- 第四章 指令系统

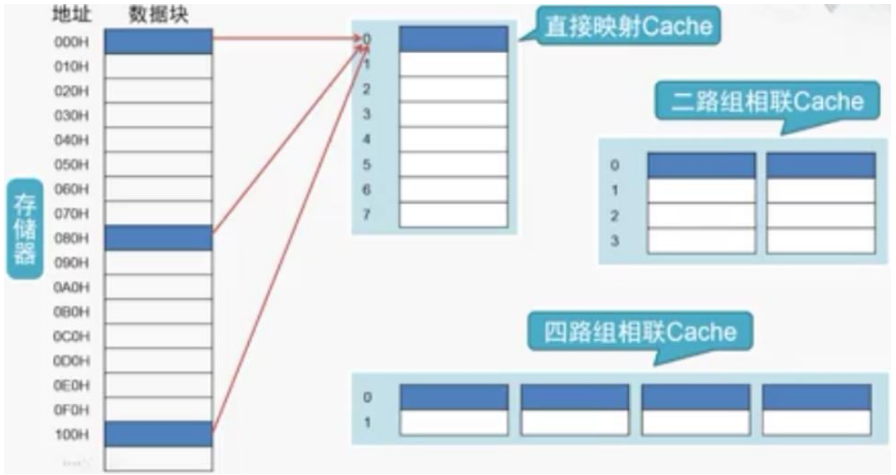
Cache地址映射

- Cache大小：N个数据块
  - 内存M数据块，M→N映射
  - 直接映射：N*1
  - 全相联：1*N
  - 组相联：U（组数）*V（组内块数）



Cache地址映射划分

- 求解方法
  - 确定字号w
    - 块内偏移量?
    - 数据块→X字，$X=2^w$
  - 确定行号r
    - 总共有多少行?
    - 直接映射N/组相联U，$N/U=2^r$
  - 确定标记s-r
    - 多少个数据块被映射到同一行?
    - 计算方法：内存总数据块地址位s-行号r
    - 内存大小→Y数据块，$2^s=Y$



地址映射方式例题

- 主存容量1MB，字长8位，块大小16B，Cache容量64KB
- 1）采用直接映射，给出[F0010H]对应标记为 [填空1] 、行号为[填空2]、字号为 [填空3] 。
  - 确定字号w
    - 数据块→16字，$X=2^w$　　∴ 字号w = 4
  - 确定行号r
    - 行数$N=64KB/16B=2^{12}$　∴ 行号r = 12
  - 确定标记s-r
    - 内存大小$1MB/16B=2^{16}$ ∴ s = 16
    - 标记位数：s-r=4
  - F0010H= 1111 0000 0000 0001 0000

标记　行号　字号

地址映射方式例题

- 主存容量1MB，字长8位，块大小16B，Cache容量64KB
- 2）采用二路组相联映射，给出[F0010H]对应标记为 [填空1] 、行号为[填空2]、字号为 [填空3] 。
  - 确定字号w
    - 数据块→16字，$X=2^w$　　∴ 字号w = 4
  - 确定行号r
    - 行数$N=64KB/16B/2=2^{11}$　∴ 行号r = 11
  - 确定标记
    - 内存大小$1MB/16B=2^{16}$ ∴ s = 16
    - 标记位数：s-r=5
  - F0010H= 1111 0000 0000 0001 0000

标记　　组号　　字号

地址映射方式例题

- 主存容量1MB，字长8位，块大小16B，Cache容量64KB
- 3）采用全相联映射，给出[F0010H]对应标记为 [填空1]、字号为 [填空2]。
  - 确定字号w
    - 数据块→16字，$X=2^w$　　∴ 字号w = 4
  - 确定标记（行号r=0）
    - 内存大小1MB/16B=2^{16} ∴ s = 16
  - F0010H= 1111 0000 0000 0001 0000

标记　　字号

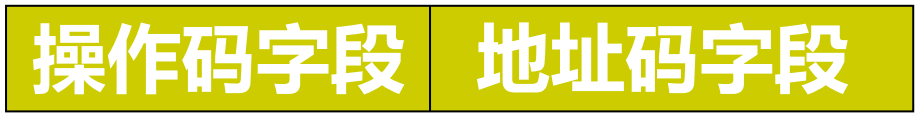
期末复习

- 第一章 计算机系统概论
- 第二章 运算方法与运算器
- 第三章 多层次存储器
- 第四章 指令系统
  - 指令格式
  - 寻址方式


指令系统
指令集分类
CISC
RISC
指令格式
操作码字段（OP）
OP与支持操作数关系，2^N
定长编码：操作码位数固定，指令长度不固定
变长编码：操作码长度不固定
指令操作码扩展
地址码字段
三地址、二地址、一地址、零地址
指令格式特点分析
指令字长度：半字长、单字长、双字长
指令类型：零地址、一地址、二地址（RR、RS、SS）、三地址
支持指令数目：OP码
支持寄存器数目：寄存器段位数
说明寻址方式
寻址方式总结表格
寻址方式
寻址方式
有效地址
访存次数

操作码（1）

- 操作码：OP（Operation code）
  - 表示该指令应进行什么性质的操作
  - 如进行加法、减法、乘法、除法、取数、存数等
  - 不同的指令用操作码字段的不同编码来表示，每一种编码代表一种指令
- 组成操作码字段的位数一取决于计算机指令系统的规模
- 较大的指令系统就需要更多的位数
- n位（操作码）→2^n
  - 32条指令：5位操作码

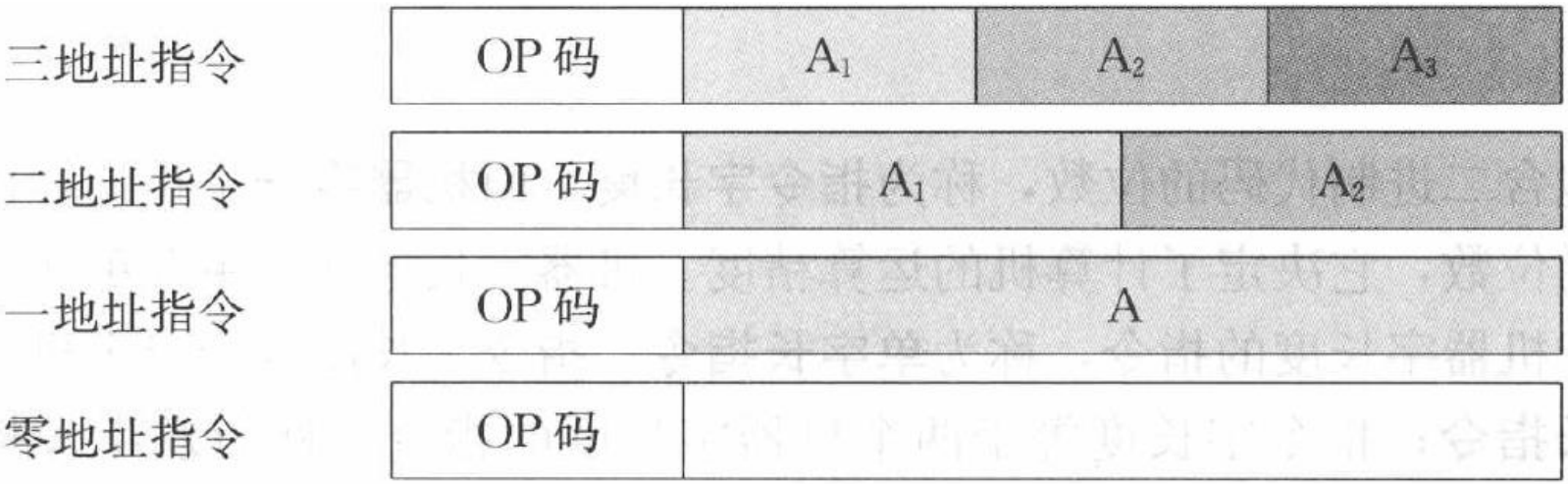


操作码——指令数目

- 编码特征：n位（操作码）→2^n
  - 5位操作码→32条指令
  - 6位操作码→ 64条指令
  - 7位操作码→ 128条指令
  - 8位操作码→ 256条指令 ……
  - 问题——编码存在浪费：130条指令，8位操作码
- 解决思路
  - 允许浪费，扩大指令长度——等长方法
  - 操作码变长，充分利用指令字——变长方法

指令分类

- 根据一条指令中有几个操作数地址，可将该指令称为几操作数指令或几地址指令
  - 三地址指令
  - 二地址指令
  - 单地址指令
  - 零地址指令



单选题 1分

设置

某指令长度为20位，具有双操作数、单操作数、无操作数三类指令，操作数地址6位、操作码长度8位，已设计出m条双操作数指令，n条无操作数指令，请问单操作数指令最多为?

A 128位

B 64位

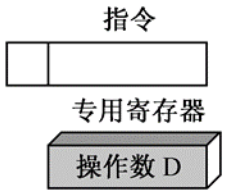
C 256-m-n位

D 128-m位

期中复习

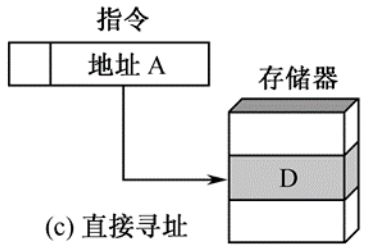
- 第一章 计算机系统概论
- 第二章 运算方法与运算器
- 第三章 多层次存储器
- 第四章 指令系统
  - 指令格式
  - 寻址方式

基本寻址方式

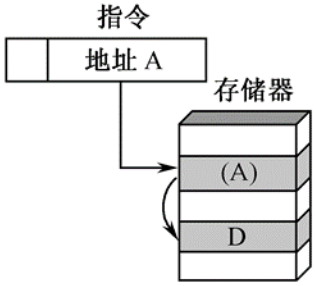


(a) 隐含寻址

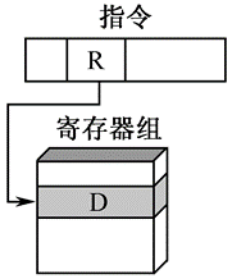
(b) 立即寻址



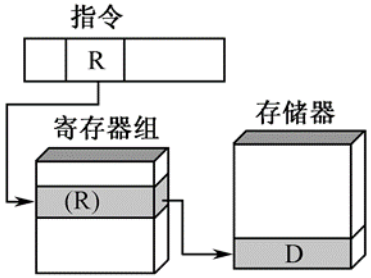
(c) 直接寻址



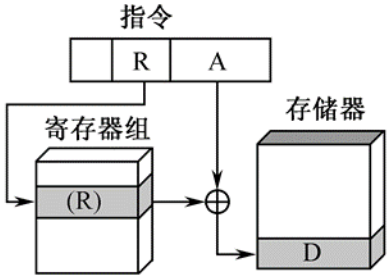
(d) 间接寻址



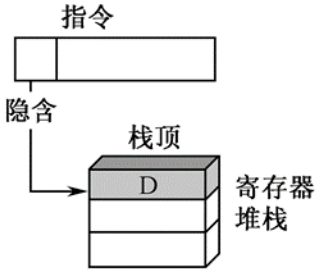
(e) 寄存器寻址



(f) 寄存器间接寻址

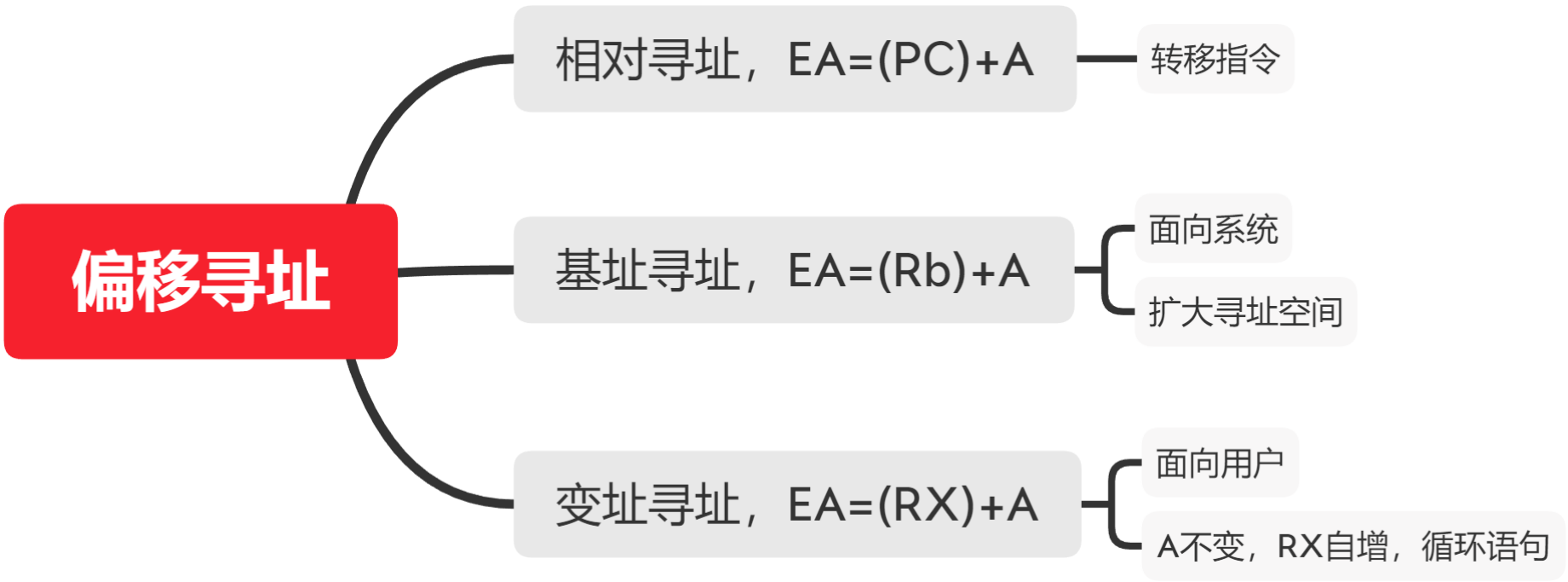


(g) 偏移寻址



(h) 堆栈寻址

偏移寻址——总结


偏移寻址
相对寻址，EA=(PC)+A
转移指令
基址寻址，EA=(Rb)+A
面向系统
扩大寻址空间
变址寻址，EA=(RX)+A
面向用户
A不变，RX自增，循环语句

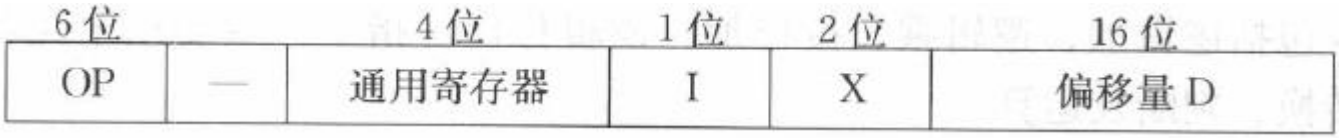
寻址方式总结（重要）

寻址方式	有效地址	访存次数	示例
隐含寻址	语句默认指定	0	STC、CLC
立即寻址	指令包含操作数	0	MOV AH，80H
寄存器寻址	EA=Ri	0	MOV AH，80H
直接寻址	EA=A	1	ADD R0, [6]
间接寻址	EA=(A)	2	
寄存器间接寻址	EA=(Ri)	1	MOV AX,[ESP]
相对寻址	EA=(PC)+A	1	转移指令
基址寻址	EA=(Bx)+A	1	扩大访存、段寻址
变址寻址	EA=(Rx)+A	1	循环语句
堆栈寻址	EA=(SS)+SP	1	POP、PUSH

偏移寻址

寻址例题

[例4] 一种二地址RS型指令的结构如下：



其中I为间接寻址标志位，X为寻址模式字段，D为偏移量字段。通过I，X，D的组合，可构成如下寻址方式：

寻址方式	I	X	有效地址E算法	说明
(1)	0	00	E=D	
(2)	0	01	E=(PC) ± D	PC为程序计数器
(3)	0	10	E=(R_2) ± D	R_2为变址寄存器
(4)	1	11	E=(R_3)	
(5)	1	00	E=(D)	
(6)	0	11	E=(R_1) ± D	R_1为基址寄存器

1）直接寻址
2）相对寻址
3）变址寻址
4）寄存器间接寻址
5）间接寻址
6）基址寻址