

1. 怎么设计出指令集（指令格式），指令集怎么实现？



### 指令格式举例（3）

某机指令字长度为16位，包括基本操作码4位和3个地址段，每个地址段长4位，其格式为：

OP	A <sub>1</sub>	A <sub>2</sub>	A <sub>3</sub>
----	----------------	----------------	----------------

- 1) 4位基本操作码若全部用于表示三地址指令，则共有多少条？
- 2) 若三地址指令15条，二地址指令最多可有多少条？
- 3) 若三地址指令、二地址指令和一地址指令各有15条，零地址指令16条，则共有61条指令。



### 指令格式举例（3）

0 0 0 0	X X X X	X X X X	X X X X	} 15条 三地址指令
0 0 0 1	X X X X	X X X X	X X X X	
⋮	⋮	⋮	⋮	
1 1 1 0	X X X X	X X X X	X X X X	
1 1 1 1	0 0 0 0	X X X X	X X X X	} 15条 二地址指令
1 1 1 1	0 0 0 1	X X X X	X X X X	
⋮	⋮	⋮	⋮	
1 1 1 1	1 1 1 0	X X X X	X X X X	



## 指令格式举例（3）

1 1 1 1	1 1 1 1	0 0 0 0	X X X X	} 15条 一地址指令
1 1 1 1	1 1 1 1	0 0 0 1	X X X X	
⋮	⋮	⋮	⋮	
1 1 1 1	1 1 1 1	1 1 1 0	X X X X	

1 1 1 1	1 1 1 1	1 1 1 1	0 0 0 0	} 16条 零地址指令
1 1 1 1	1 1 1 1	1 1 1 1	0 0 0 1	
⋮	⋮	⋮	⋮	
1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	



## 指令格式举例（4）

设某指令系统指令字长16位，每个地址码为6位。若二地址指令15条，一地址指令34条，则剩下零地址指令最多有多少条？

OP(4)	A1(6)	A2(6)
-------	-------	-------

解:操作码按短到长进行扩展编码

二地址指令: (0000 – 1110) 共15条 (不扩展时为16条)

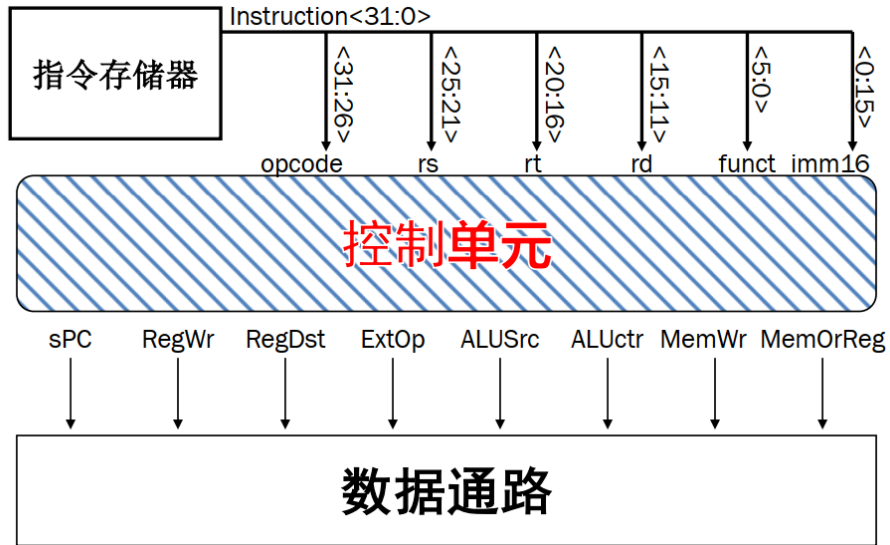
一地址指令: 1111 (000000 – 100001); (34条)

零地址指令: 1111 (100010 – 111111) (000000 – 111111)

(30种扩展标志)

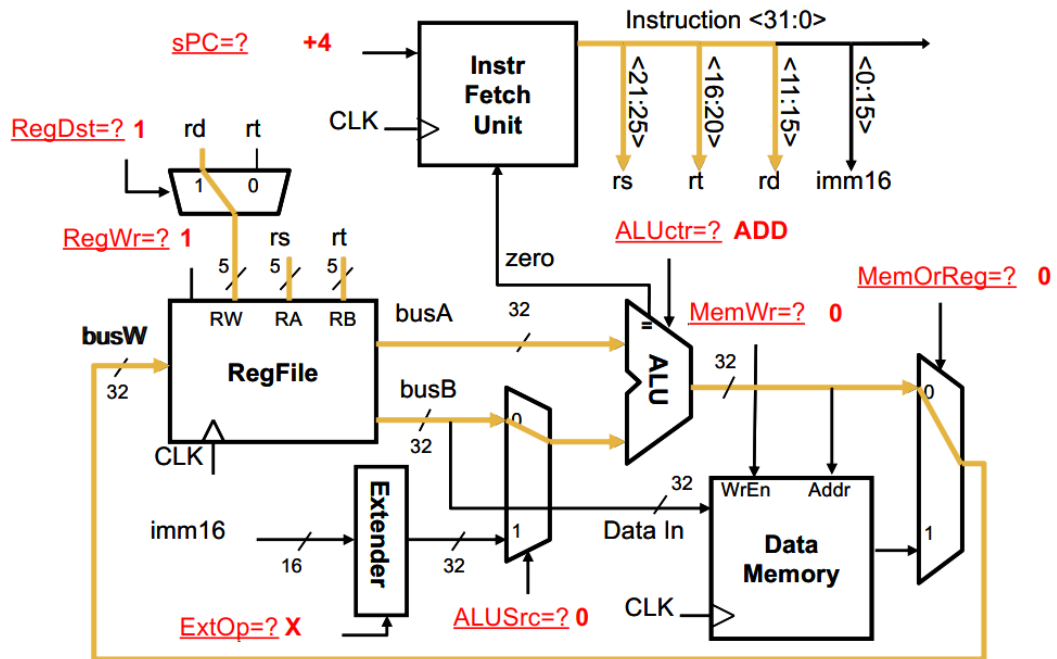
故零地址指令最多有  $30 \times 2^6 = 15 \times 2^7$  种

- 取指、译码、执行、存储访问、写回，给几条指令，分析数据通路；控制器生成控制信号电路（与逻辑、或逻辑结合）



第 4 讲\_部分 3-单周期 CPU 设计

$R[rd] \leftarrow R[rs] + R[rt];$





- 实现方式多种多样，这里介绍一种简单直观方式
- 思路
  - 指令的OP和FUNC编码唯一确定指令——译码 译码（ID）阶段
  - 指令使能各控制信号
- 实现方式
  - 指令译码
    - I型和J型指令，只与OP相关
      - $BEQ = \overline{op[5]} \wedge \overline{op[4]} \wedge \overline{op[3]} \wedge op[2] \wedge \overline{op[1]} \wedge \overline{op[0]}$
    - R型指令
      - $Rtype = \overline{op[5]} \wedge \overline{op[4]} \wedge \overline{op[3]} \wedge \overline{op[2]} \wedge \overline{op[1]} \wedge op[0]$
      - $ADD = Rtype \wedge (op[5] \wedge \overline{op[4]} \wedge \overline{op[3]} \wedge \overline{op[2]})$
  - 指令控制
    - $MemWrite = SW$
    - $RegWrite = ADD \vee SUB \vee ORI \vee LW$

与逻辑

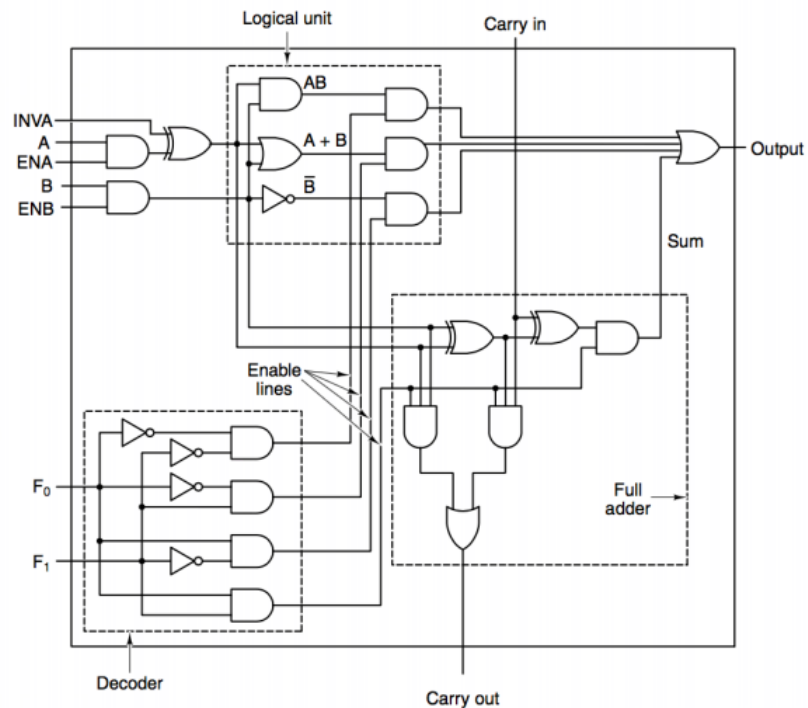
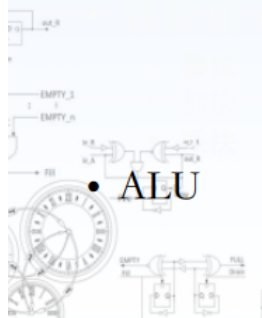
为什么公式中不用考虑其他变量？

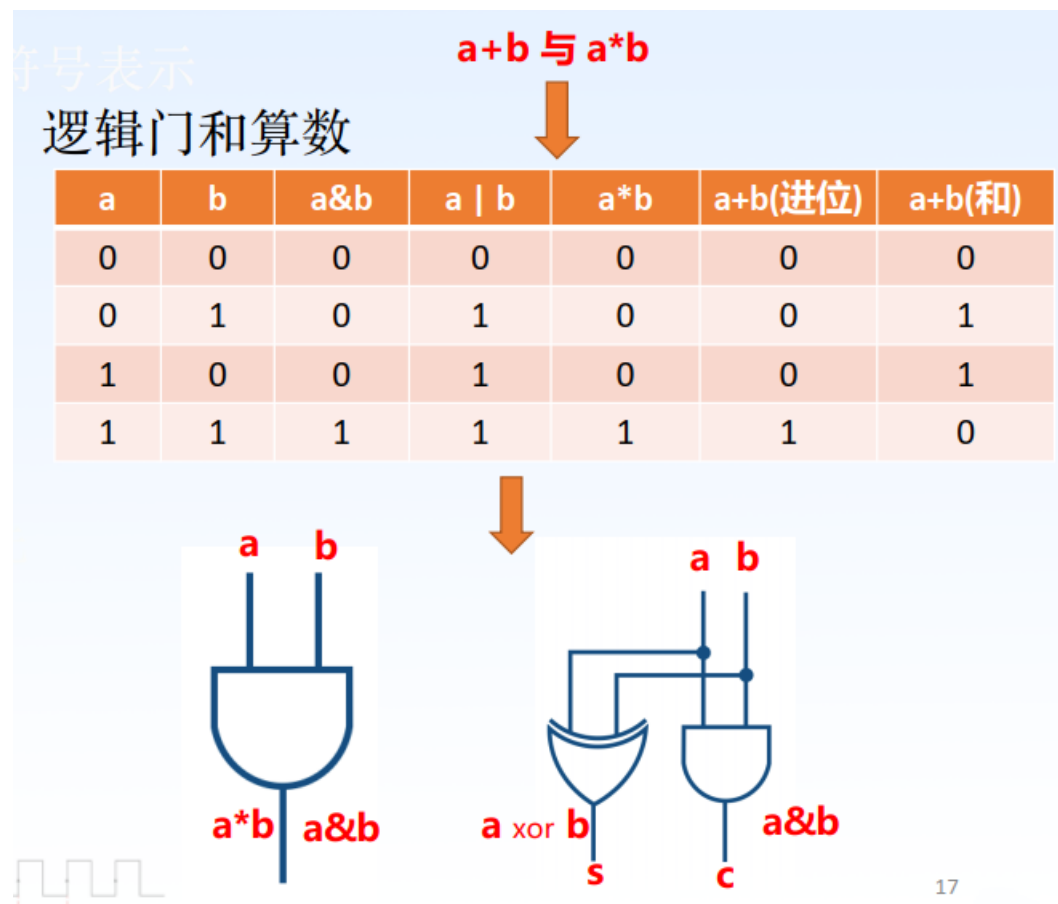
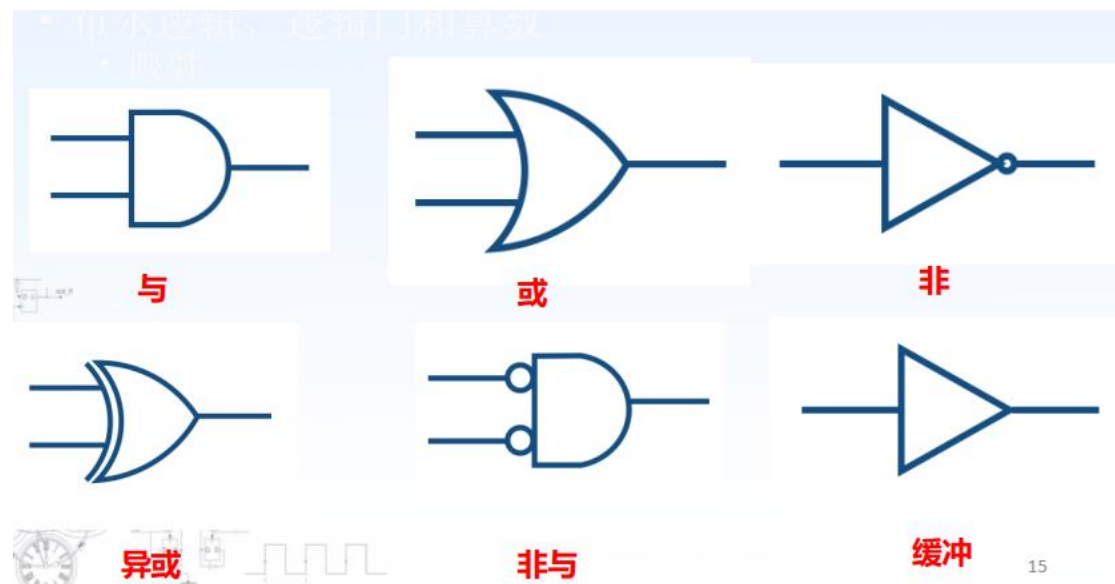
或逻辑

3. 分析 ALU 功能，进行的什么运算（逻辑，算数）（给一张图，输入一个编码）

## 从布尔逻辑到电路实现——逻辑门

- 逻辑门的符号
- 布尔逻辑
- 控制
- 运算





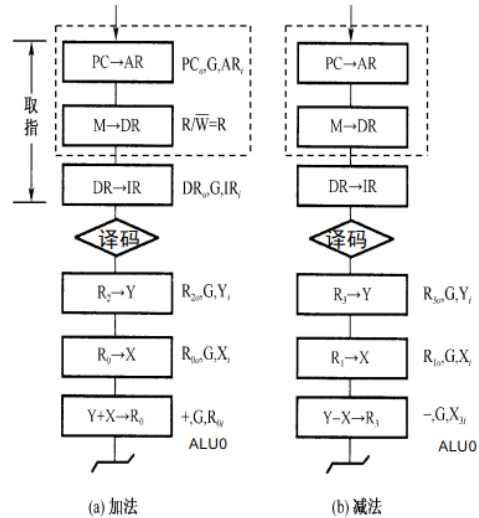
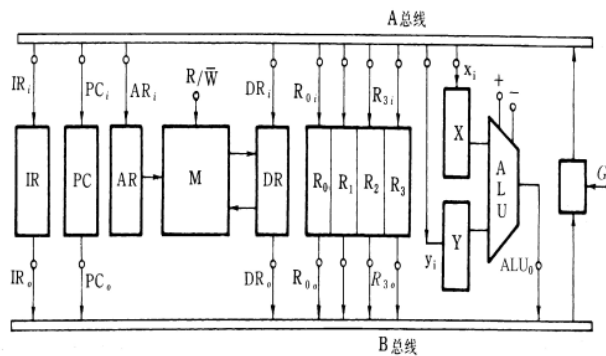
4. 总线型 CPU (单总线, 双总线, 多总线) 一个周期内分布操作流程图。



# 总线型CPU

## 分步操作流程图

### 简单双总线CPU的数据通路



如果是单总线结构有何优劣？

41

### 5. 补码的计算

-128 的补码是 10000000。

一个正整数的补码和该数值的原码（即该数值的二进制形式）相同。

求负整数的补码的方法如下：

- 取该负整数的绝对值，并将该绝对值以二进制形式（原码）表示；
- 将上一步骤的二进制按位取反（包括符号位，0 变 1，1 变 0）；
- 对上一步骤的结果加 1。

例如，求 -10 的补码的方法是：

- 取 -10 的绝对值 10，并以二进制形式，即 10 的原码为 0000000000001010（此处按一个整数占 16 位计算，下同）；
- 对 1010 按位取反，得到二进制 1111111111110101；
- 对上一步骤的结果再加 1，得到 -10 的补码 1111111111110110。

### 6. 浮点数的计算

#### 32位单精度浮点数（floating point）的规格化形式



- 第 0 位为符号位（数符），记为 **S**
- 第 1~8 位为8位**移码**，表示指数，记为 **E**
- 第 9~31 位为23位二进制**原码**小数表示的尾数**M**

浮点数： $\pm 1.\text{xxxxxxxxxx}_{\text{two}} \times 2^{\text{Exponent}}$



S: 0表示正, 1表示负

E: 单精度数偏置量为127, 双精度为1023

- 单精度规格化数的阶码范围为0000 0001 (-126) ~ 1111 1110 (127)
- 全0和全1用于表示特殊值

M: 规格化尾数最高位总是1, 所以隐含表示, 省1位

- 尾数长度: 1 + 23 bits (单精度)
- 尾数长度: 1 + 52 bits (双精度)

值的范围:

- 单精度:  $(-1)^S \times (1 + \text{Significand}) \times 2^{(\text{Exponent}-127)}$
- 双精度:  $(-1)^S \times (1 + \text{Significand}) \times 2^{(\text{Exponent}-1023)}$

如何表示0?

- 指数: 均为0
- 尾数: 均为0
- 符号位: 0/1均可
- +0: 0 00000000 000000000000000000000000
- -0: 1 00000000 000000000000000000000000

如何表示无穷大 $+\infty/-\infty$ ?

- 指数: 均为1
- 尾数: 均为0
- 符号位: 0/1均可
- $+\infty$ : 0 11111111 000000000000000000000000
- $-\infty$ : 1 11111111 000000000000000000000000

如何表示非数 (NaN)?

- 指数: 均为1
- 尾数: 非0
- 符号位: 0/1均可

Sqrt (- 4.0) = ?          0/0 = ?

➤ 称为非数 ( NaN, Not a Number)





## 浮点数加减法基本要点

(假定： $X_m$ 、 $Y_m$ 分别是X和Y的尾数， $X_e$ 和 $Y_e$ 分别是X和Y的阶码)

- (1) 求阶差： $\Delta e = Y_e - X_e$  (若 $Y_e > X_e$ ，则结果的阶码为 $Y_e$ )
- (2) 对阶：将 $X_m$ 右移 $\Delta e$ 位，尾数变为  $X_m * 2^{X_e - Y_e}$  (保留右移部分**附加位**)
- (3) 尾数加减： $X_m * 2^{X_e - Y_e} \pm Y_m$
- (4) 规格化：
  - 当尾数高位为0，则需左规：尾数左移一次，阶码减1，直到MSB为1
  - 每次阶码减1后要判断阶码是否下溢 (比最小可表示的阶码还要小)
  - 当尾数最高位有进位，需右规：尾数右移一次，阶码加1，直到MSB为1
  - 每次阶码加1后要判断阶码是否上溢 (比最大可表示的阶码还要大)
  - 阶码溢出异常处理：阶码上溢，则结果溢出；阶码下溢到无法用非规格化数表示，则结果为0
- (5) 如果尾数比规定位数长 (有附加位)，则需考虑舍入 (有4种舍入方式)
- (6) 若**运算结果尾数是0**，则需要将阶码也置0。为什么？

尾数为0说明结果应该为0 (阶码和尾数为全0)。

35



## 浮点数加法运算举例

例：用二进制浮点数形式计算  $0.5 + (-0.4375) = ?$

$$0.4375 = 0.25 + 0.125 + 0.0625 = 0.0111B$$

解： $0.5 = 1.000 \times 2^{-1}$ ， $-0.4375 = -1.110 \times 2^{-2}$

对 阶： $-1.110 \times 2^{-2} \rightarrow -0.111 \times 2^{-1}$

加 减： $1.000 \times 2^{-1} + (-0.111 \times 2^{-1}) = 0.001 \times 2^{-1}$

左 规： $0.001 \times 2^{-1} \rightarrow 1.000 \times 2^{-4}$

判溢出：无

结果为： $1.000 \times 2^{-4} = 0.0001000 = 1/16 = 0.0625$

问题：为何IEEE 754 加减运算右规时最多只需一次？

因为即使是两个最大的尾数相加，得到的和的尾数也不会达到4，故尾数的整数部分最多有两位，保留一个隐含的“1”后，最多只有一位被右移到小数部分。

36

1. 计算  $X+Y$ ，其中  $X=0.1010 \times 2^{11}$ ； $Y=0.1101 \times 2^{10}$ ；要求：阶码、尾数以双符号位法表示，另阶码、尾数数值位分别取3位和6位，计算结果采用0舍1入法，尾数保留6位数字。

1解：

$$[X]_{\text{补}} = 00\ 011 ; 00.101000 \quad [Y]_{\text{补}} = 00\ 010 ; 00.110100$$

①对阶：

$$[\Delta]_{\text{补}} = [j_x]_{\text{补}} - [j_y]_{\text{补}} = 00\ 011$$

$$+ \quad \underline{11\ 101}$$

$$= \quad 1\ 00\ 001$$

阶差为+1，大于0

$M_y$ 右移1位， $E_y+1$

$$[Y]_{\text{补}} = 00\ 011 ; 00.011010 (0)$$

②尾数求和：

$$M_x = \quad 00.101000$$

$$+ \quad \underline{M_y = \quad 00.011010 (0)}$$

$$\quad \quad \quad 01.000010 (0)$$

尾数和溢出，需要右规

③右规：

$$[X+Y]_{\text{补}} = 00\ 011 ; 01.000010 (0)$$

$$= 00\ 100 ; 00.100001 (0)$$

$$\text{即：} X+Y = 00\ 100 ; 00.100001 (0) = 0.100001 (0) \times 2^{100}$$

④舍入：0舍1入，保留6位

$$M_{X+Y} = 00.100001 (0) = 00.100001$$

$$X+Y = 00\ 100 ; 00.100001 = 0.100001 \times 2^{100}$$

计算  $X \cdot Y$ ，其中  $X = (3/8) \times 2^{-3}$ ； $Y = (-5/8) \times 2^{-4}$ ；要求：阶码、尾数以双符号位法表示，另阶码、尾数数值位分别取3位和4位，计算结果采用0舍1入法，尾数保留4位数字。

2解：

$$[X]_{\text{补}} = 11\ 101 ; 00.0110 \quad [Y]_{\text{补}} = 11\ 100 ; 11.0110$$

①对阶：

$$[\Delta]_{\text{补}} = [j_x]_{\text{补}} - [j_y]_{\text{补}} = 11\ 101$$

$$+ \quad \underline{00\ 100}$$

$$= \quad 1\ 00\ 001$$

阶差为+1，大于0

$M_y$ 右移1位， $E_y+1$

$$[Y]_{\text{补}} = 11\ 101 ; 11.1011 (0)$$

②尾数求差：

$$[-M_y]_{\text{补}} = 00.0101 (0)$$

$$M_x = \quad 00.0110$$

$$+ \quad \underline{-M_y = \quad 00.0101 (0)}$$

$$\quad \quad \quad 00.1011 (0)$$

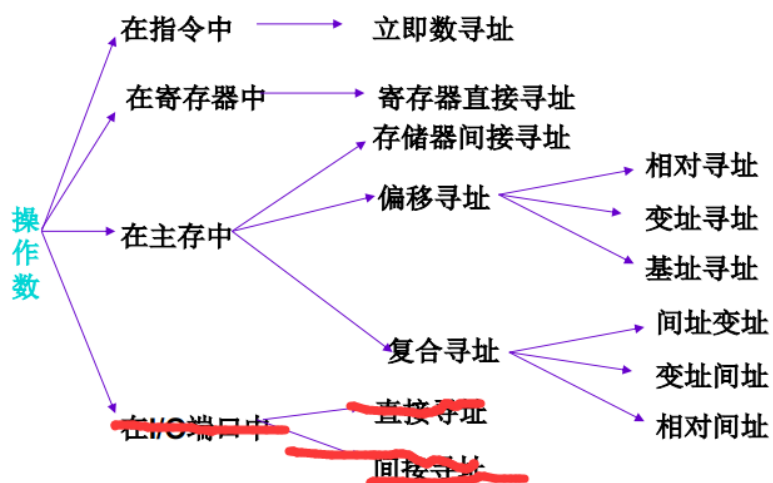
尾数和未溢出，且无需规格化

7. 操作数寻址

第4讲\_部分2-指令系统.pdf



# 操作数寻址



## 操作数寻址实例（1）

主存数据分布如图所示,若A为单元地址 (A)为A的内容,求

$((7)) - (N) + ((N)) + (((N)))$ 的值

0	9
1	11
2	22
3	53
4	44
5	3
6	2
7	0
.	.
N	5

$$((7)) = 9$$

$$(N) = 5$$

$$((N)) = 3$$

$$(((N))) = 53$$

$$\text{结果} = 60$$



## 操作数寻址实例（2）

字长16位，主存64K，指令单字长单地址码，80条指令。寻址方式有直接、间接、相对、变址。请设计指令格式

解： 80条指令  $\Rightarrow$  OP字段需要7位(  $2^7=128$  )

4种寻址方式  $\Rightarrow$  寻址方式特征位2位

16位字长指令还余7位作为地址位，由于是单地址指令，所以地址位的长度也是7位。

指令格式如下：

7	2	7
OP	X	A

PC为16位

变址寄存器16位

- 相对寻址  $E = (PC) + D$  ,寻址范围为： 64K
- 变址寻址  $E = (R) + D$  , 寻址范围为： 64K
- 直接寻址  $E = D$  , 寻址范围为128
- 间接寻址  $E = (D)$  , 寻址范围为64K

52

8. CPU 访存实例，给地址变换，怎么映射

作业三

9. 命中关系，判断机制，是否都可命中



## CPU访存：访存次数

TLB	Page table	Cache	Possible? If so, under what circumstance?
hit	hit	miss	
miss	hit	hit	
miss	hit	miss	
miss	miss	miss	
hit	miss	miss	
hit	miss	hit	
miss	miss	hit	

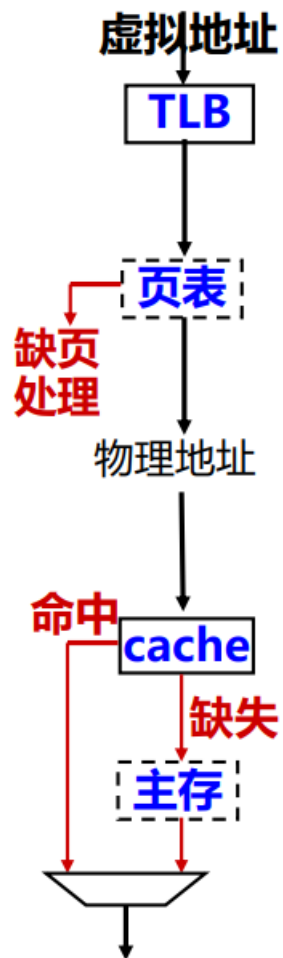
最好的情况是hit、 hit、 hit，此时，访问主存几次？ 不需要访问主存！

以上组合中，最好的情况是？ hit、 hit、 miss和miss、 hit、 hit 访存1次

以上组合中，最坏的情况是？ miss、 miss、 miss 需访问磁盘、并访存至少2次

介于最坏和最好之间的是？ miss、 hit、 miss 不需访问磁盘、但访存至少2次

16



10. 描述题（系统互连）总线 工作原理、查询方式、仲裁、同步定时、异步定时
11. I/O 子系统（机制） 中断（多重中断屏蔽字，程序执行顺序（作业四）） DMA（简答 DMA 过程）  
DMA 过程 启动 DMA，建立虚拟通道，I/O 传输。。。（中断）
12. 第六题，从这学期学的内容判断分析某种 CPU 结构的好处，有无缺陷（怎样让 CPU 更快）开放题