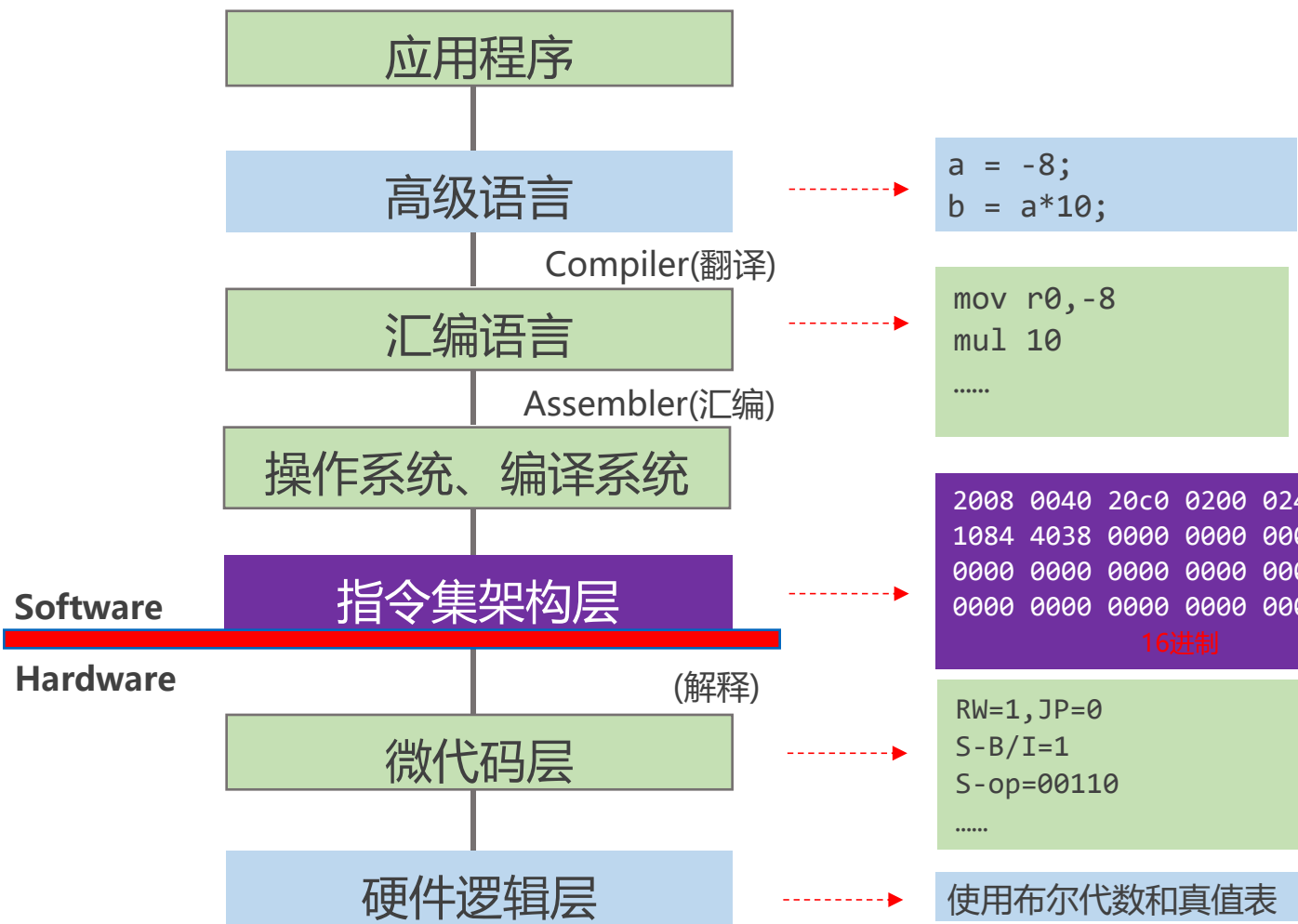


四、指令系统

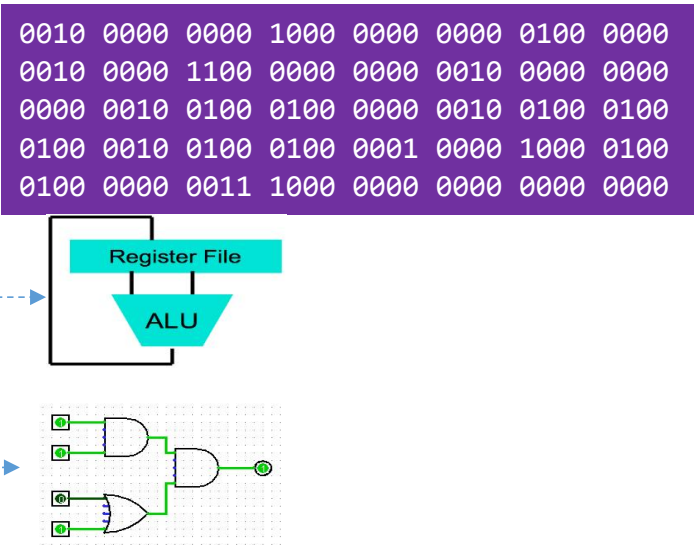
指令系统基本概念

- 机器指令（指令）
 - 计算机能直接识别、执行的某种操作命令
- 指令系统（指令集）
 - 一台计算机中所有机器指令的集合

计算机指令系统层次



- 指令系统是CPU设计的依据
- 软件设计的基础
- 硬件和软件间的界面
- 直接影响计算机系统性能



硬件设计四原则

- 简单性来自规则性
 - Simplicity favors regularity
 - 指令越规整设计越简单
- 越小越快
 - Smaller is faster
 - 面积小，传播路径小，门延迟少
- 加快经常性事件
 - Make the common case fast
- 好的设计需要适度的折衷
 - Good design demands good compromises

指令格式

■ 用二进制代码表示的指令结构形式

□ 指令要处理什么数据?

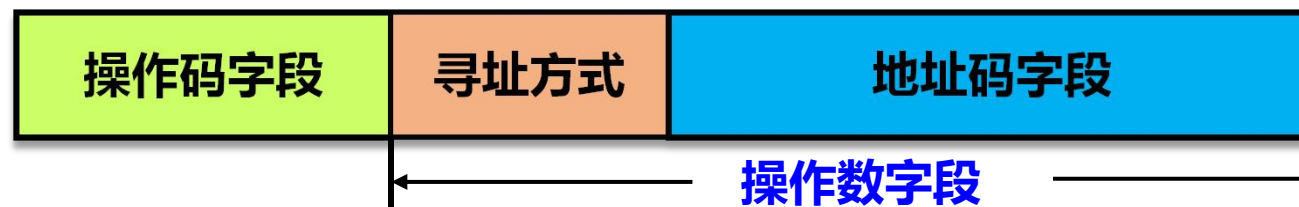
指令的操作数需要解决的问题

□ 指令要做什么样的处理?

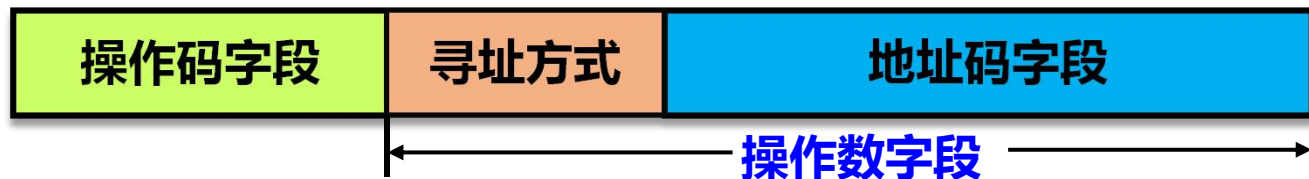
□ 怎样才能得到要处理的数据?

指令的操作码需要解决的问题

指令的寻址方式需要解决的问题



指令格式



■ 操作码字段长度决定指令系统规模

- 每条指令对应一个操作码
- 定长操作码 $\text{Length}_{\text{Op}} = \lceil \log_2 n \rceil$
- 变长操作码 操作码向不用的地址码字段扩展

S3	S2	S1	S0(Sub)	Cin	
1	0	0	0	0	A+B (ADD)
1	0	0	0	1	A+B+1 (ADC)

变长操作码举例：32位指令



- 3种指令操作码部分不得重叠，否则无法区分和译码，可以设置特征位（可能会有浪费）
- 设双操作数指令数为 k ，显然 $k < 2^8$
- $2^8 - k$ 为多余状态，可用于表示其他类型指令
- 可用于单操作数指令的条数 = $(2^8 - k) * 2^{12}$ ， 2^{12} 是多余12位组合

指令系统举例

- 设某指令系统指令字长16位，每个地址码为6位。若要求设计二地址指令15条、一地址指令34条，问最多还可设计多少条（无操作数）零地址指令？

双操作数15



2^4

单操作数34



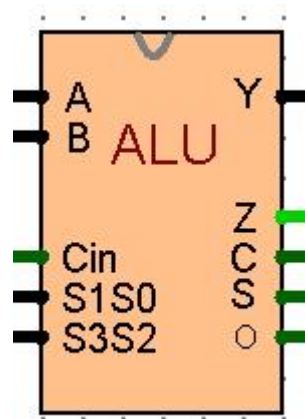
$(2^4 - 15) * 2^6$

无操作数?



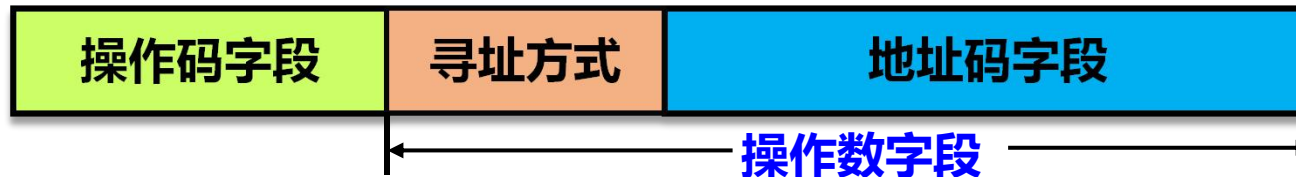
$((2^4 - 15) * 2^6 - 34) * 2^6$

our指令设计



ALU 操作	S3	S2	S1	S0(Sub)	Cin	解释
移位操作	0	0	0	0	x	MOV B
	0	0	0	1	x	逻辑/算术左移 B (SHL/SAL)
	0	0	1	0	x	逻辑右移 B (SHR)
	0	0	1	1	x	算术右移 B (SAR)
比较操作	0	1	0	0	x	A>B (CMP)
	0	1	0	1	x	A=B (CMP)
	0	1	1	0	x	A<B (CMP)
算术运算	1	0	0	0	0	A+B (ADD)
	1	0	0	0	1	A+B+1 (ADC)
	1	0	0	1	0	A-B (SUB)
	1	0	0	1	1	A-B-1 (SBB)
	1	0	1	0	1	A+1
	1	0	1	1	1	A-1
逻辑运算	1	1	0	0	x	A and B (AND)
	1	1	0	1	x	A or B (OR)
	1	1	1	0	x	~A (NOT)
	1	1	1	1	x	A xor B (XOR)

指令格式



■ 操作数字段可能有多个

- **寻址方式字段** 长度与寻址方式种类有关，也可能隐含在操作码字段
- **地址码字段** 作用及长度都与寻址方式有关

寻址方式

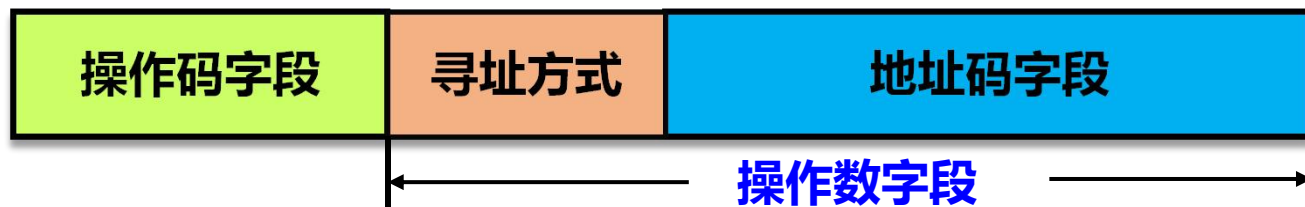
■ 寻找指令或操作数有效地址的方式

□ 指令寻址

- ◆ 顺序寻址
- ◆ 跳跃寻址

□ 操作数寻址

- ◆ 立即寻址、直接寻址
- ◆ 间接寻址、寄存器寻址
- ◆ 寄存器间接寻址、相对寻址
- ◆ 基址\变址寻址、复合寻址



顺序寻址

■ 顺序寻址方式

- 程序对应的机器指令序列在主存顺序存放
- 执行时从第一条指令开始，逐条取出并执行

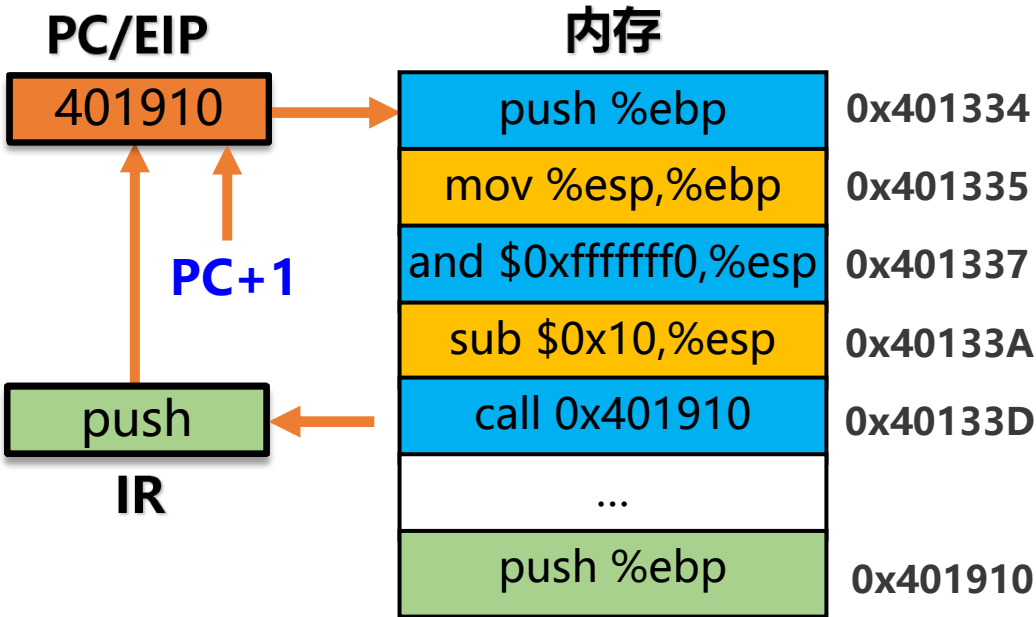
■ 实现方式

- **程序计数器**（PC）对指令序号进行计数
- PC存放下条指令地址，初始值为程序首址
- 执行一条指令， **$PC = PC + \text{当前指令字节长度}$**

$\text{Mem}[pc++]\rightarrow \text{IR}$

跳跃寻址

- **跳跃寻址方式**：当程序中出现分支或循环时，就会改变程序的执行顺序
 - 下条指令地址不是PC++得到，而是由指令本身给出
 - 跳跃的处理方式是重新修改PC的内容，然后进入取指令阶段
 - 绝对位移和相对位移



IR(A) → IR

操作数的寻址方式

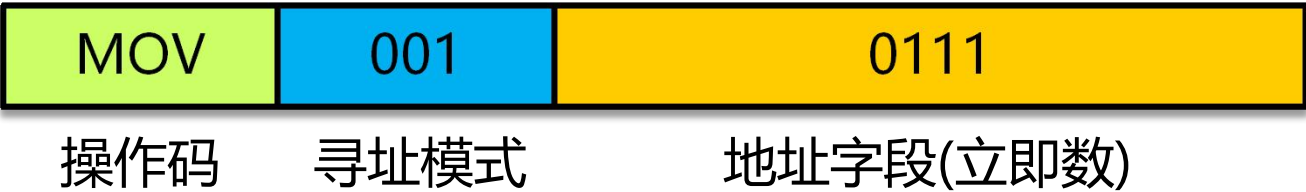
■ 如何找到操作数

- 立即数寻址
- 寄存器寻址
- 内存直接寻址
- 内存间接寻址
- 寄存器间接寻址
- 相对寻址
- 变址寻址
- 堆栈寻址
-

立即寻址

■ 地址码字段是操作数本身

□ 例: MOV R0, 7



寻址方式 和控制功能	3 位组合	R/W	S-B/I	RM	WM	JP	JZ	JC
选择两个寄存器进行运 算（RW，可读可写）	000	1	0	0	0	0	0	0
寄存器和立即数进行运 算	001	1	1	0	0	0	0	0
写主存	010	0	0	0	1	0	0	0
读主存	011	1	0	1	0	0	0	0
无条件转移 JB	100	0	1	0	0	1	0	0

寄存器寻址(Register Addressing)

■ 操作数在CPU的内部寄存器中

□ MOV R1, R0

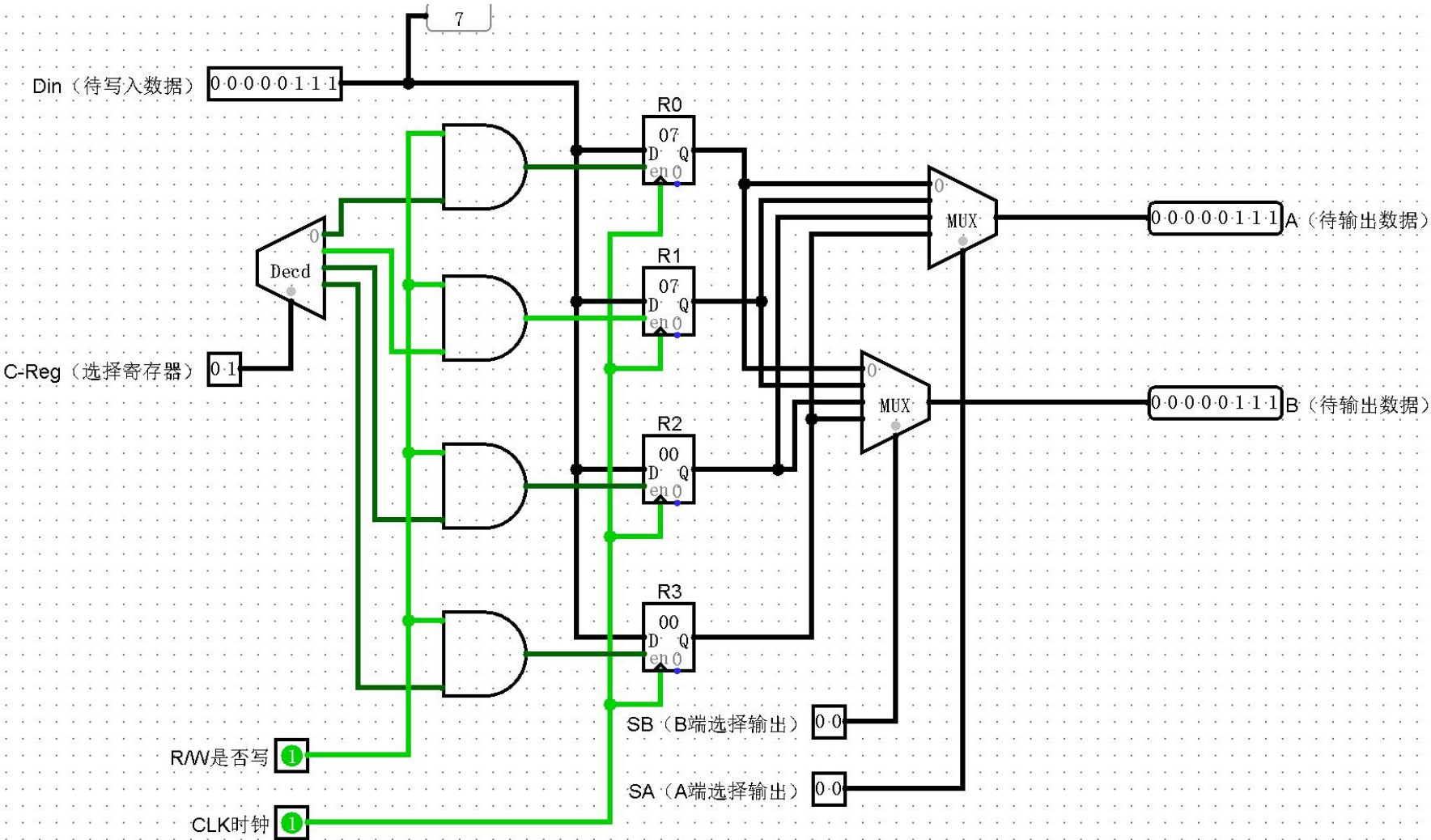


寻址方式 和控制功能	3 位组合	R/W	S-B/I	RM	WM	JP	JZ	JC
选择两个寄存器进行运算（RW，可读可写）	000	1	0	0	0	0	0	0
寄存器和立即数进行运算	001	1	1	0	0	0	0	0
写主存	010	0	0	0	1	0	0	0
读主存	011	1	0	1	0	0	0	0



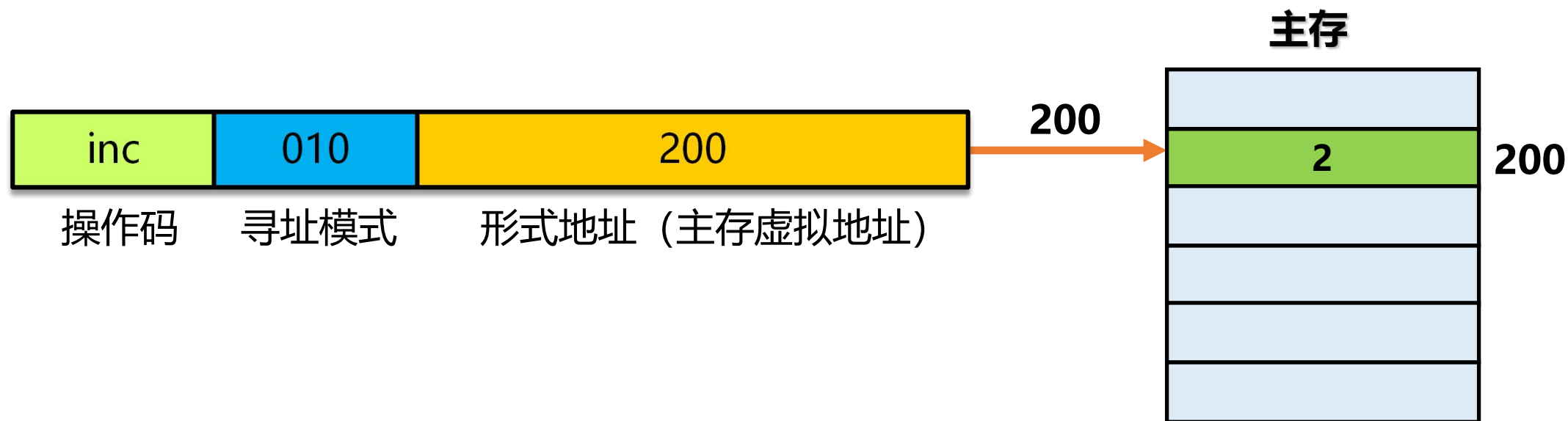
MOV R1, R0

操作码 寻址模式 形式地址（寄存器编号）



直接寻址(Direct Addressing)

- 地址码字段直接给出操作数在内存的地址.
- inc [200]



间接寻址(Indirect Addressing)

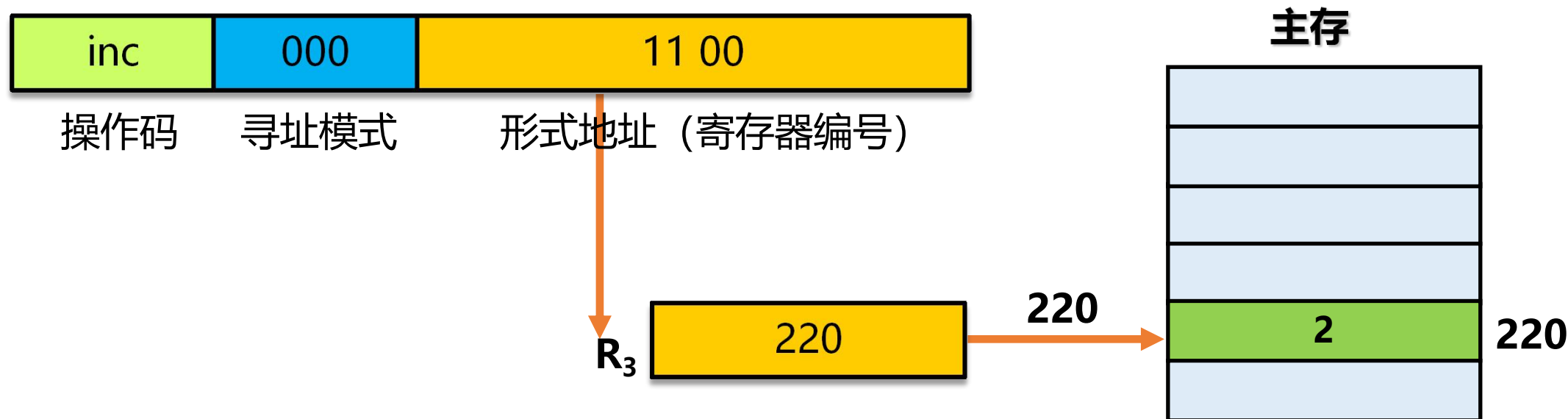
- D是操作数地址的地址



需两次访存，速度慢，已淘汰

寄存器间接寻址 (Register Indirect Addressing)

- 寄存器中是操作数在内存中的地址
- store R3, R0



our指令设计

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
寻址方式和控制功能			S-op					S-Regs							
			S3	S2	S1	S0	Cin	C-Reg	SA		SB				
											4 位立即数				

寻址方式 和控制功能	3 位组合	R/W	S-B/I	RM	WM	JP	JZ	JC
选择两个寄存器进行运算（RW，可读可写）	000	1	0	0	0	0	0	0
寄存器和立即数进行运算	001	1	1	0	0	0	0	0
写主存	010	0	0	0	1	0	0	0
读主存	011	1	0	1	0	0	0	0
无条件转移 JP	100	0	1	0	0	1	0	0
条件转移(相等转) JZ	101	0	1	0	0	0	1	0
条件转移(小于转) JC	110	0	1	0	0	0	0	1
选择两个寄存器进行运算（RO，只读）	111	0	0	0	0	0	0	0

指令系统设计原则

- 完备性：指令丰富，基本功能齐全，使用方便
- 有效性：程序占空间小，执行速度快
- 规整性：
 - 对称性 （对不同寻址方式的支持）
 - 匀齐性 （对不同数据类型的支持）
 - 一致性 （指令长度和数据长度与设定相一致）
- 兼容性：系列机软件向上兼容

指令格式设计举例




■ **例1.** 字长16位，主存64K，指令单字长单地址，80条指令。寻址方式有直接、间接、相对、变址。请设计指令格式。

- 80条指令 \Rightarrow OP字段需7位 ($2^7=128$)
- 4种寻址方式 \Rightarrow 寻址方式位需2位
- 单字长单地址 \Rightarrow 地址码长度 = $16 - 7 - 2 = 7$ 位



指令集体系结构 Instruction Set Architecture (ISA)

■ 不同类型的CPU执行不同指令集，是设计CPU的依据

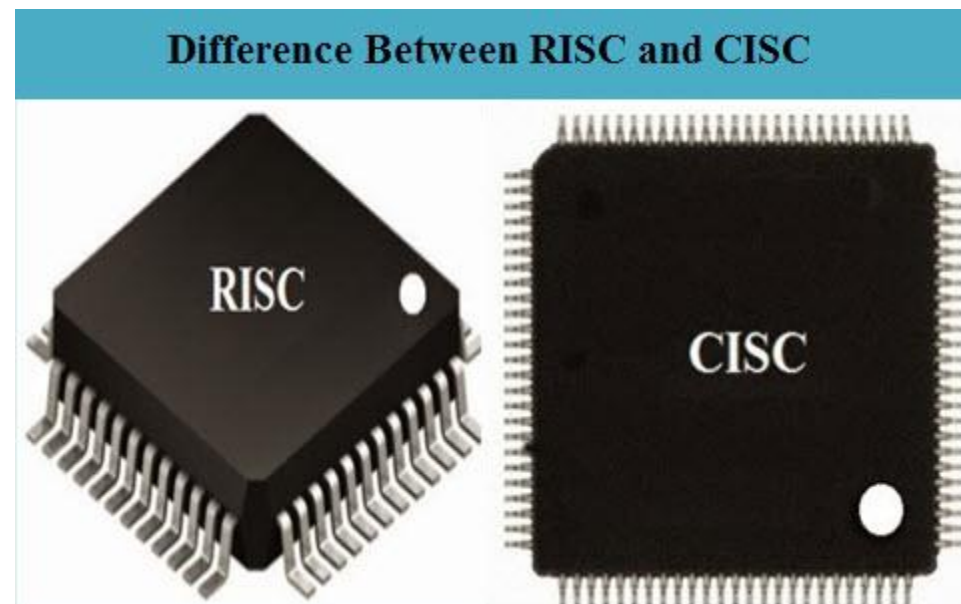
- ◆  1970 DEC **PDP-11** 1992 **ALPHA**(64位)
- ◆  1978 **x86**, 2001 **IA64**
- ◆  1980 **PowerPC**
- ◆  1981 **MIPS**
- ◆  1985 **SPARC**
- ◆  1991 **arm**
- ◆  2016 **RISC-V**

■ 指令集优劣

- 方便硬件设计，方便编译器实现，性能更优，成本功耗更低

指令系统发展方向

- CISC---复杂指令系统计算机
 - Complex Instruction System Computer
 - 指令数量多，指令功能复杂的计算机。
 - Intel x86
- RISC---精简指令系统计算机
 - Reduced Instruction System Computer
 - 指令数量少，指令功能单一的计算机。
 - MIPS、RISC-V
- CSIC、RISC互相融合



精减指令系统(RISC)

- 指令条数少，只保留使用频率最高的简单指令，指令定长
 - 便于硬件实现，用软件实现复杂指令功能
- Load/Store架构
 - 只有存/取数指令才能访问存储器，其余指令的操作都在寄存器之间进行
 - 便于硬件实现
- 指令长度固定，指令格式简单、寻址方式简单
 - 便于硬件实现
- CPU设置大量寄存器（32~192）
 - 便于编译器实现
- 一个机器周期完成一条机器指令
- RISC CPU采用硬布线控制，CISC采用微程序

指令分类方法

■ 按计算机系统的层次结构分类

- 微指令、机器指令、宏指令

■ 按操作数物理位置分类

- 存储器 - 存储器 (SS) 型、寄存器 - 寄存器 (RR) 型、寄存器 - 存储器 (RS) 型

■ 按指令长度分类

- 定长指令，变长指令

■ 按操作数个数分类

- 四地址、三地址、二地址、单地址、零地址

■ 按指令功能分类

按操作数个数分类

三地址指令



$(A1) \text{ OP } (A2) \rightarrow A3$

二地址指令



$(A1) \text{ OP } (A2) \rightarrow A1$

单地址指令



$(AC) \text{ OP } (A1) \rightarrow AC$

零地址指令



如停机、空操作、开关中断等



程序设计

our指令设计

设计指令

R0+R1→R2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
寻址方式和控制功能			S-op					S-Regs							
			S3	S2	S1	S0	Cin	C-Reg		SA		SB			
											4 位立即数				

寻址方式 和控制功能	3 位组合	R/W	S-B/I	RM	WM	JP	JZ	JC
选择两个寄存器进行运算（RW，可读可写）	000	1	0	0	0	0	0	0
寄存器和立即数进行运算	001	1	1	0	0	0	0	0
写主存	010	0	0	0	1	0	0	0
读主存	011	1	0	1	0	0	0	0
无条件转移 JP	100	0	1	0	0	1	0	0
条件转移(相等转) JZ	101	0	1	0	0	0	1	0
条件转移(小于转) JC	110	0	1	0	0	0	0	1
选择两个寄存器进行运算（RO，只读）	111	0	0	0	0	0	0	0

ALU 操作	S3	S2	S1	S0(Sub)	Cin	解释
移位操作	0	0	0	0	x	MOV B
	0	0	0	1	x	逻辑/算术左移 B（SHL/SAL）
	0	0	1	0	x	逻辑右移 B（SHR）
	0	0	1	1	x	算术右移 B（SAR）
比较操作	0	1	0	0	x	A>B（CMP）
	0	1	0	1	x	A=B（CMP）
	0	1	1	0	x	A<B（CMP）
算术运算	1	0	0	0	0	A+B（ADD）
	1	0	0	0	1	A+B+1（ADC）
	1	0	0	1	0	A-B（SUB）
	1	0	0	1	1	A-B-1（SBB）
	1	0	1	0	1	A+1
	1	0	1	1	1	A-1
	1	1	0	0	0	A+1B（AND）

设计程序

■ -8*10

高级语言

```
int a, m;  
a=-8;  
m=a*10;
```

自然语言

```
-8→r0  
r0→r1  
r0左移1位  
r1左移3位  
r0+r1→r2  
r2→r3 (存储到内存)
```

汇编语言

```
mov r0, -8  
mov r1, r0  
mov r3, 0  
sll r0  
sll r1  
sll r1  
sll r1  
add r2,r0,r1  
store r3,r2
```

机器语言 (16进制)

```
2008  
0040  
20c0  
0200  
0244  
0244  
0244  
1084  
4038
```