



合肥工业大学

HEFEI UNIVERSITY OF TECHNOLOGY

计算机组成原理

第七章 指令系统

阙夏

计算机与信息学院

2022/5/17



四、指令系统

(一)指令系统的基本概念

(二)指令格式

(三)寻址方式

(四) 数据的对齐和大/小端存放方式

(五) **CISC** 和 **RISC** 的基本概念

(六) 高级语言程序与机器级代码之间的对应

1.编译器，汇编器和链路器的基本概念

2.选择结构语句的机器级表示

3.循环结构语句的机器级表示

4.过程（函数）调用对应的机器级表示



1

7.1 机器指令

2

7.2 操作数类型和操作类型

3

7.3 寻址方式

4

7.4 指令格式举例

5

7.5 RISC 技术



一、指令格式

指令：就是让计算机识别并执行某种操作的**命令**。

指令系统：一台计算机中**所有机器指令的集合**。
称为这台计算机的**指令系统**。

系列计算机：指基本**指令系统相同**且**基本体系，结构相同**的一系列计算机。

系列机能解决软件兼容问题的**必要条件**是该系列的各种机种**有共同**的指令集，而且新推出的机种的指令系统一定包含旧机种的所有指令，因此在旧机种上运行的各种软件可以不加任何修改地在新机种上运行。

<https://baijiahao.baidu.com/s?id=1718417865583947761&wfr=spider&for=pc>



指令的一般格式:

n位操作码字段的指令系统
最多能够表示 2^n 条指令。

操作码字段	地址码字段
-------	-------

1. 操作码

指令应该执行什么性质的操作和具有何种功能。
反映机器做什么操作。

(1) 长度固定 (译码简单)

用于指令字长较长的情况, RISC

如 IBM 370 操作码 8 位

(2) 长度可变

操作码分散在指令字的不同字段中



操作码的位数随地址数的减少而增加

	OP	A ₁	A ₂	A ₃	
4 位操作码	0000	A ₁	A ₂	A ₃	最多15条三地址指令
	0001	A ₁	A ₂	A ₃	
	⋮	⋮	⋮	⋮	
	1110	A ₁	A ₂	A ₃	
8 位操作码	1111	0000	A ₂	A ₃	最多15条二地址指令
	1111	0001	A ₂	A ₃	
	⋮	⋮	⋮	⋮	
	1111	1110	A ₂	A ₃	
12 位操作码	1111	1111	0000	A ₃	最多15条一地址指令
	1111	1111	0001	A ₃	
	⋮	⋮	⋮	⋮	
	1111	1111	1110	A ₃	
16 位操作码	1111	1111	1111	0000	16条零地址指令
	1111	1111	1111	0001	
	⋮	⋮	⋮	⋮	
	1111	1111	1111	1111	



操作码的位数随地址数的减少而增加

4 位操作码

OP	A ₁	A ₂	A ₃
----	----------------	----------------	----------------

0000	A ₁	A ₂	A ₃
0001	A ₁	A ₂	A ₃
⋮	⋮	⋮	⋮
1110	A ₁	A ₂	A ₃

8 位操作码

1111	0000	A ₂	A ₃
1111	0001	A ₂	A ₃
⋮	⋮	⋮	⋮
1111	1110	A ₂	A ₃

12 位操作码

1111	1111	0000	A ₃
1111	1111	0001	A ₃
⋮	⋮	⋮	⋮
1111	1111	1110	A ₃

16 位操作码

1111	1111	1111	0000
1111	1111	1111	0001
⋮	⋮	⋮	⋮
1111	1111	1111	1111

两个原则:

1. 编码不能重复
2. 短码不能为长码前缀

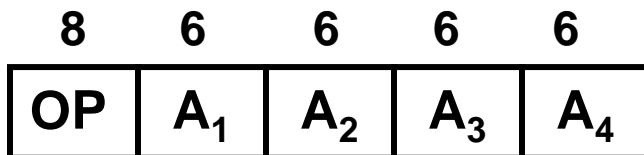
三地址指令操作码
每减少一种最多可多构成
2⁴ 种二地址指令

二地址指令操作码
每减少一种最多可多构成
2⁴ 种一地址指令

使用频率高用短码
使用频率低用长码



(1) 四地址



A₁ 第一操作数地址

A₂ 第二操作数地址

A₃ 结果的地址

A₄ 下一条指令地址

(A₁) OP (A₂) \longrightarrow A₃

设指令字长为 32 位

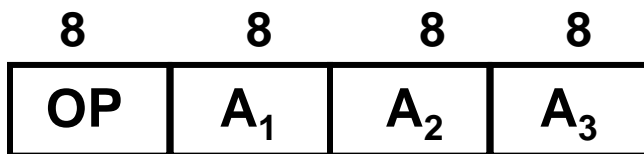
操作码固定为 8 位

4 次访存

寻址范围 **$2^6 = 64$**

若 PC 代替 A₄

(2) 三地址



(A₁) OP (A₂) \longrightarrow A₃

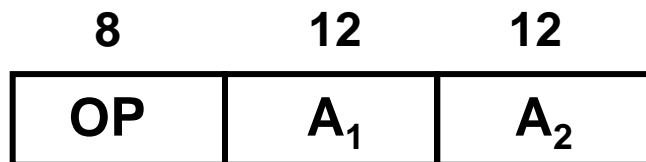
4 次访存

寻址范围 **$2^8 = 256$**

若 A₃ 用 A₁ 或 A₂ 代替



(3) 二地址



或 $(A_1) \text{ OP } (A_2) \longrightarrow A_1$

$(A_1) \text{ OP } (A_2) \longrightarrow A_2$

4 次访存

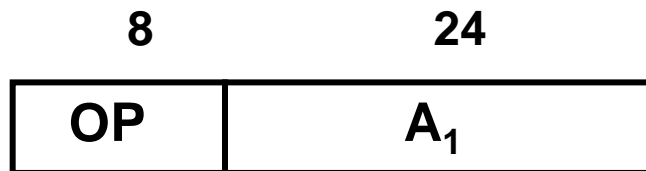
寻址范围 $2^{12} = 4 \text{ K}$

若结果存于 ACC

3次访存

若ACC 代替 A_1 (或 A_2)

(4) 一地址



$(\text{ACC}) \text{ OP } (A_1) \longrightarrow \text{ACC}$

2 次访存

寻址范围 $2^{24} = 16 \text{ M}$

(5) 零地址

无地址码

1. 无需任何操作数, 如空操作指令、停机指令等; 2. 所需操作数地址是默认的。



二、指令字长

指令字长决定于 {
操作码的长度
操作数地址的长度
操作数地址的个数

1. 指令字长 **固定**

指令字长 = 存储字长 = 机器字长（早期计算机）

2. 指令字长 **可变**

按字节的倍数变化



小结

➤ 当用一些硬件资源代替指令字中的地址码字段后

- 可扩大指令的寻址范围
- 可缩短指令字长
- 可减少访存次数

➤ 当指令的地址字段为寄存器时

三地址 OP R_1 , R_2 , R_3

二地址 OP R_1 , R_2

一地址 OP R_1

- 可缩短指令字长
- 指令执行阶段不访存



一、操作数类型

地址	绝对地址：无符号整数；相对地址：有符号数
数字	定点数、浮点数、十进制数
字符	ASCII
逻辑数	逻辑运算

二、数据在存储器中的存放方式

例7.1 12345678H，分别用大端小端方式存储
存储器中的数据存放：边界对齐

字地址	字节地址			
0	12H	34H	56H	78H
4	4	5	6	7
8	8	9	10	11

高位字节 地址为字地址

字地址	字节地址			
0	0	1	2	3
4	4	5	6	7
8	8	9	10	11

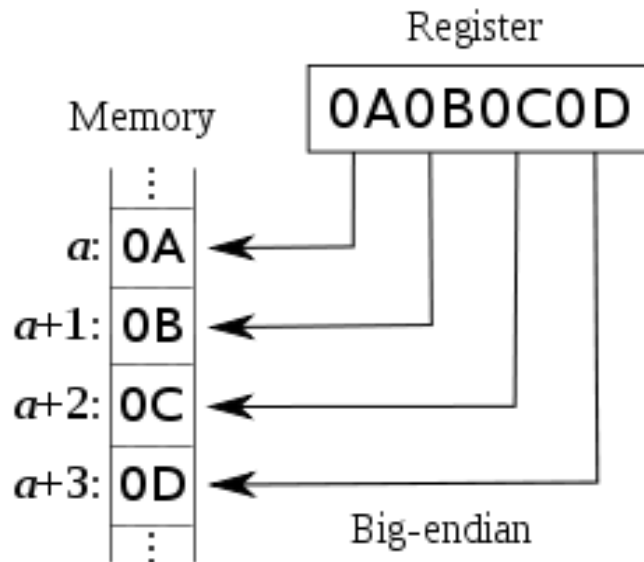
字地址	字节地址			
0	78H	56H	34H	12H
4	4	5	6	7
8	8	9	10	11

低位字节 地址为字地址



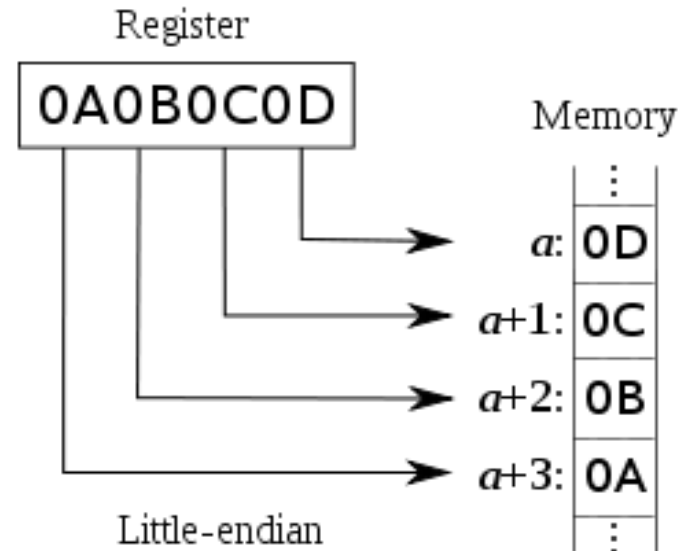
大端Big-Endian

低地址存放最高有效位
(**MSB**)，既高位字节排
放在内存的低地址端，低
位字节排放在内存的高地
址端。



小端Little-Endian

低地址存放最低有效
位 (LSB)，既低位字节
排放在内存的低地址端，
高位字节排放在内存的高
地址端。





三、操作类型

1. 数据传送

源

寄存器

寄存器

存储器

存储器

目的

寄存器

存储器

寄存器

存储器

例如

MOVE

STORE

LOAD

MOVE

MOVE

MOVE

PUSH

POP

置 "1" , 清 "0"

2. 算术逻辑操作

加、减、乘、除、增 1、减 1、求补、浮点运算、十进制运算

与、或、非、异或、位操作、位测试、位清除、位求反

如 8086 ADD SUB MUL DIV INC DEC CMP NEG
 AAA AAS AAM AAD
 AND OR NOT XOR TEST



3. 移位操作

算术移位

逻辑移位

循环移位 (带进位和不带进位)

4. 转移

(1) 无条件转移 **JMP**

(2) 条件转移

结果为零转 (Z = 1) **JZ**

结果溢出转 (O = 1) **JO**

结果有进位转 (C = 1) **JC**

跳过一条指令 **SKP**

如

300
⋮
305
306
→ 307

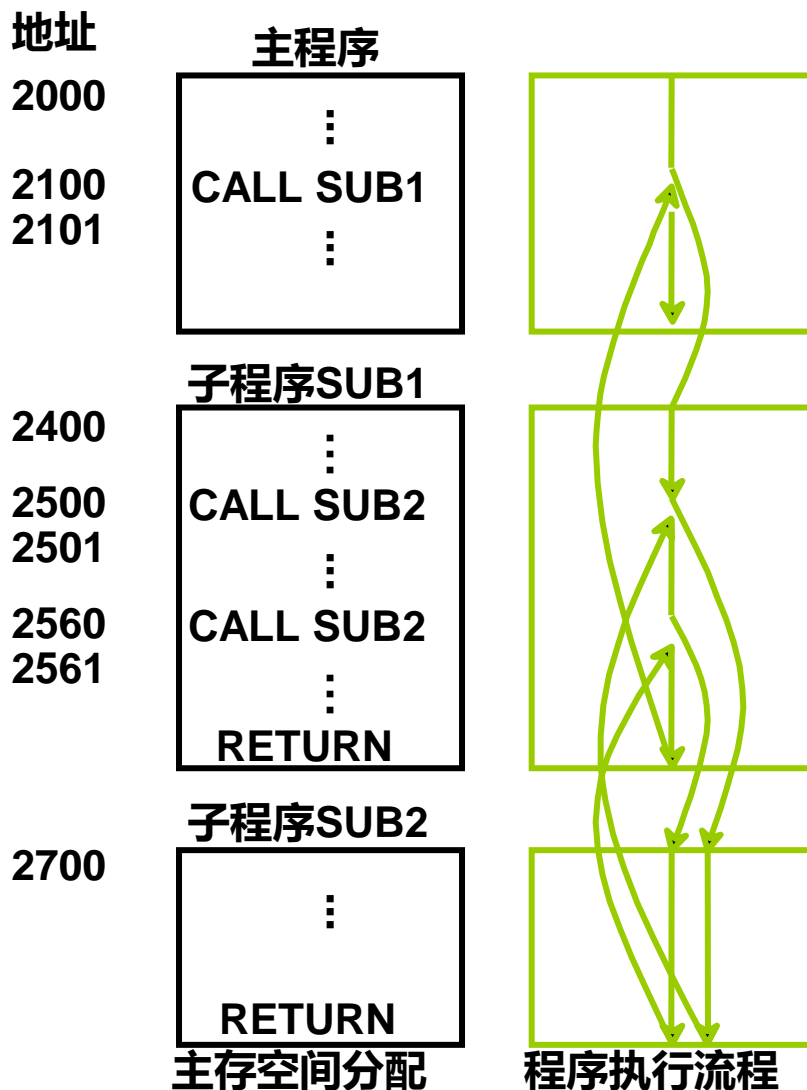
完成触发器

SKP DZ D = 0 则跳



7.2 操作数类型和操作种类

(3) 调用和返回





(4) 陷阱 (Trap) 与陷阱指令

陷阱: 计算机系统的意外事故 (如电压不稳。故障等)。

陷阱指令: 是处理陷阱的指令。

- 一般不提供给用户直接使用

在出现事故时, 由 CPU 自动产生并执行 (隐指令)

- 设置供用户使用的陷阱指令

如 8086 INT TYPE 软中断

提供给用户使用的陷阱指令, 完成系统调用

5. 输入输出

入 端口地址 \longrightarrow CPU 的寄存器

如 IN AX, m IN AX, DX

出 CPU 的寄存器 \longrightarrow 端口地址

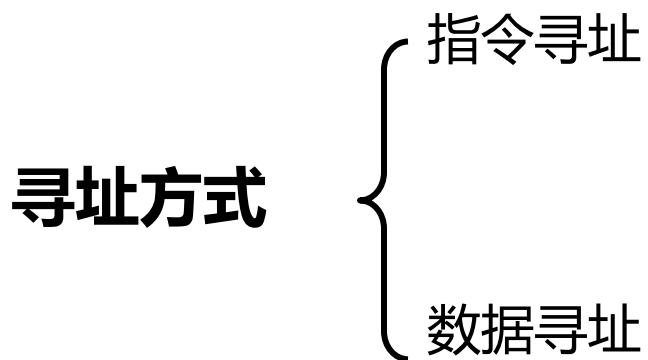
如 OUT n, AX OUT DX, AX



寻址方式

确定 本条指令 的 操作数地址

下一条 欲执行 指令 的 指令地址





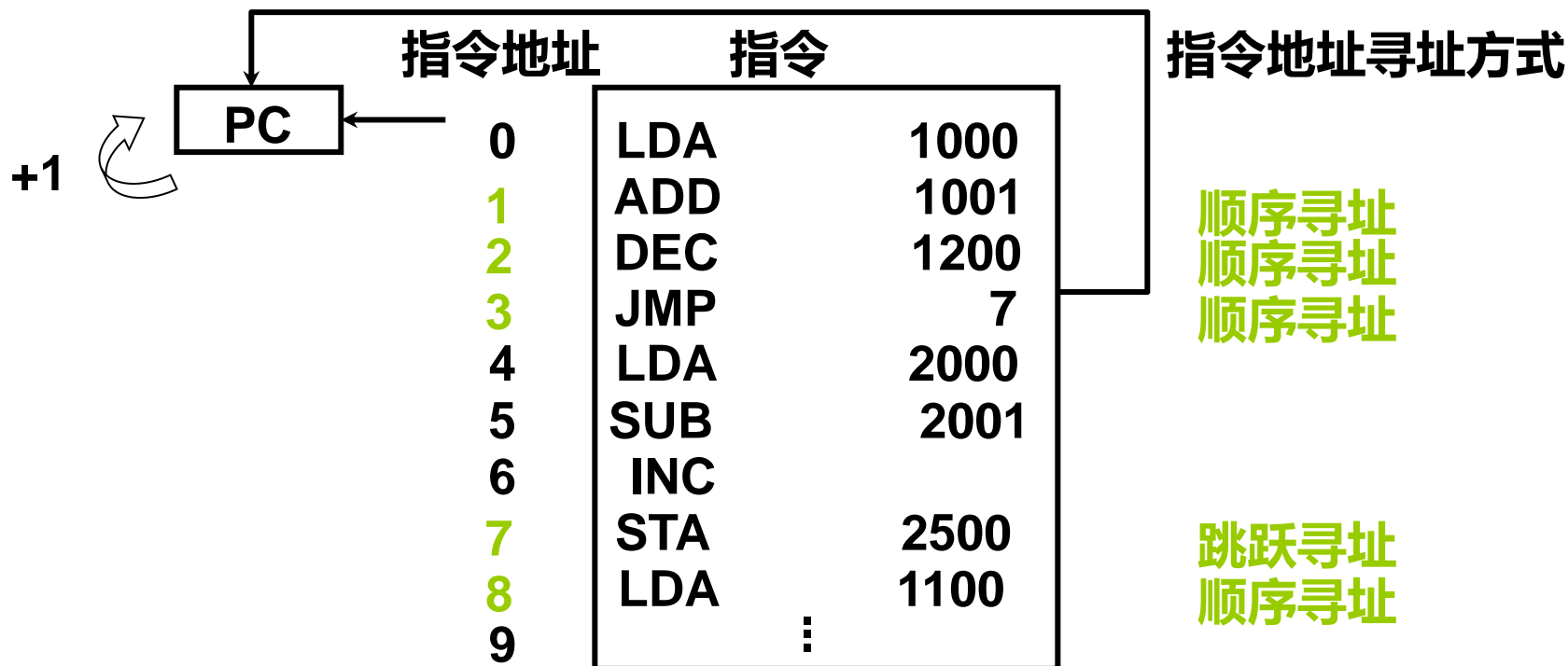
一、指令寻址

这里的+1是指下一条指令

顺序
跳跃

$(PC) + 1 \longrightarrow PC$

由转移指令指出, 如 JMP





二、数据寻址

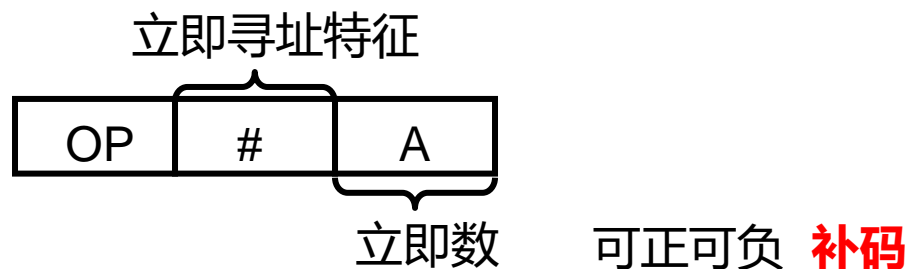
操作码	寻址特征	形式地址 A
-----	------	--------

形式地址 指令字中的地址 **A**
有效地址 操作数的真实地址 **EA**

约定 **指令字长 = 存储字长 = 机器字长**

1. 立即寻址

形式地址 A 就是操作数



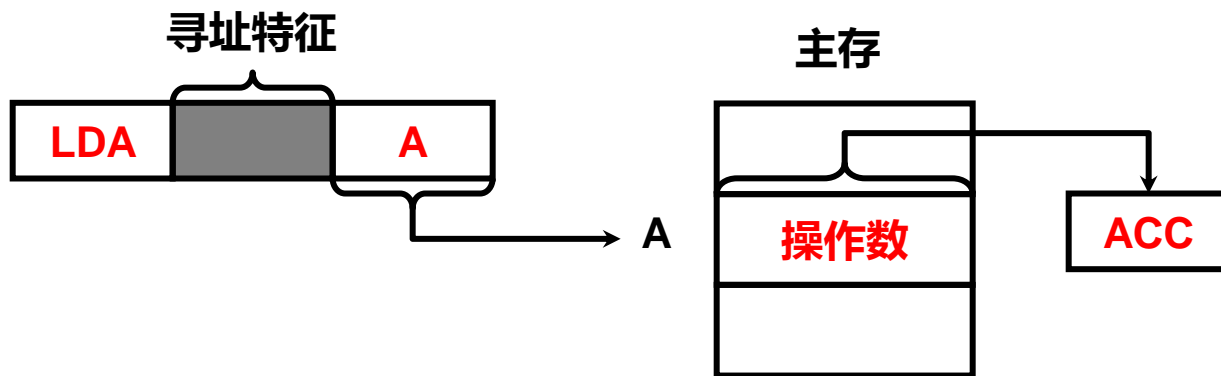
- **指令执行阶段不访存**
- **A 的位数限制了立即数的范围**



2. 直接寻址

$EA = A$

有效地址由形式地址直接给出

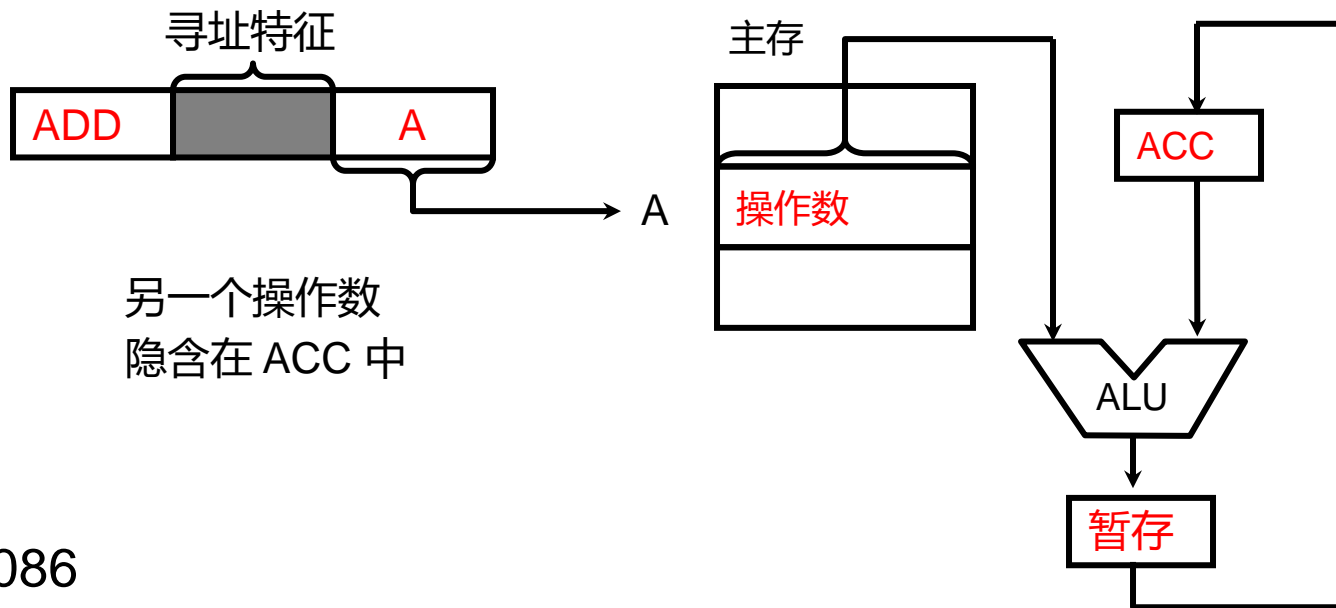


- 执行阶段访问一次存储器
- A 的位数决定了该指令操作数的寻址范围
- 操作数的地址不易修改（必须修改A）



3. 隐含寻址

操作数地址隐含在操作码中



如 8086

MUL 指令 被乘数隐含在 AX (16位) 或 AL (8位) 中

MOVS 指令 源操作数的地址隐含在 SI 中

目的操作数的地址隐含在 DI 中

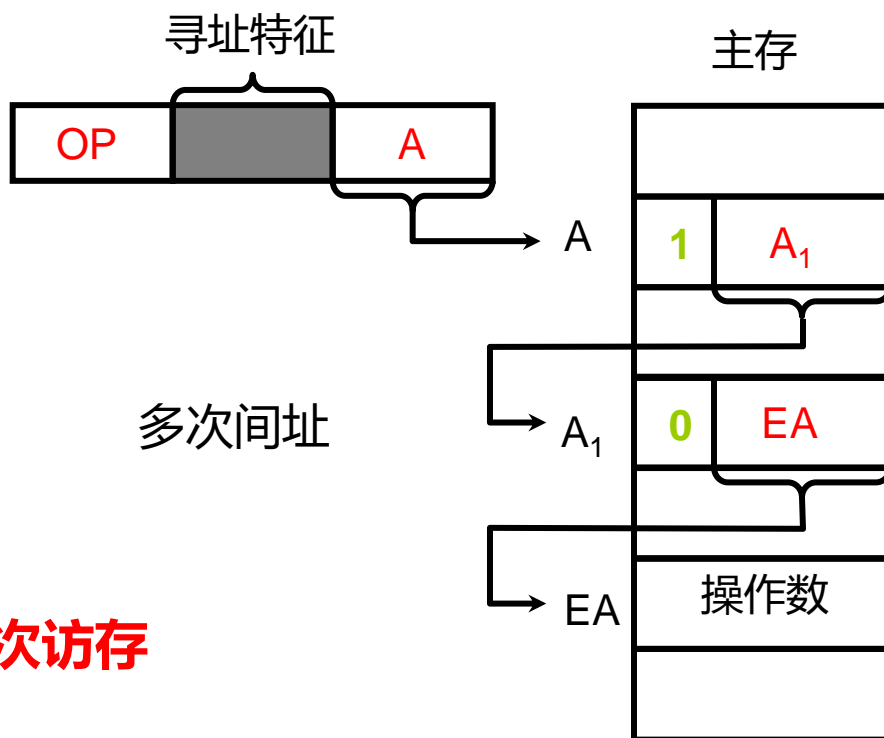
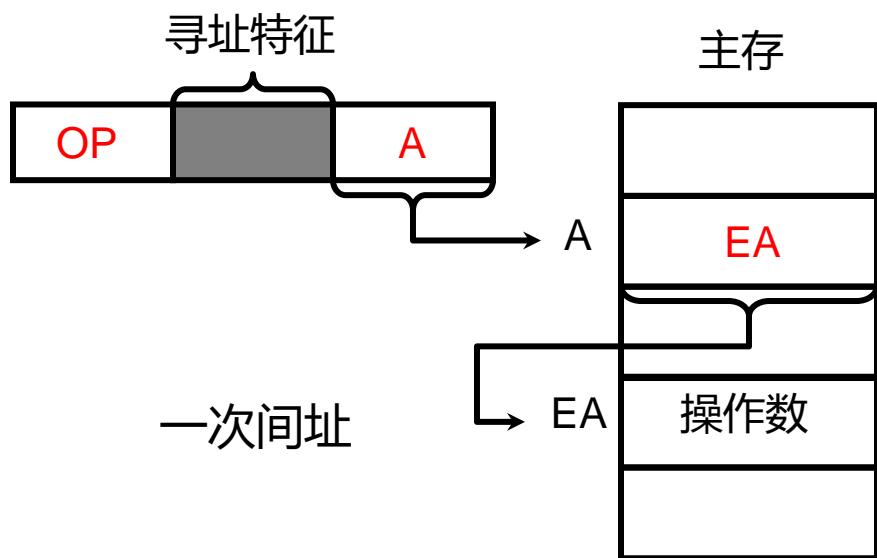
- 指令字中少了一个地址字段，可缩短指令字长



4. 间接寻址

$EA = (A)$

有效地址由形式地址间接提供

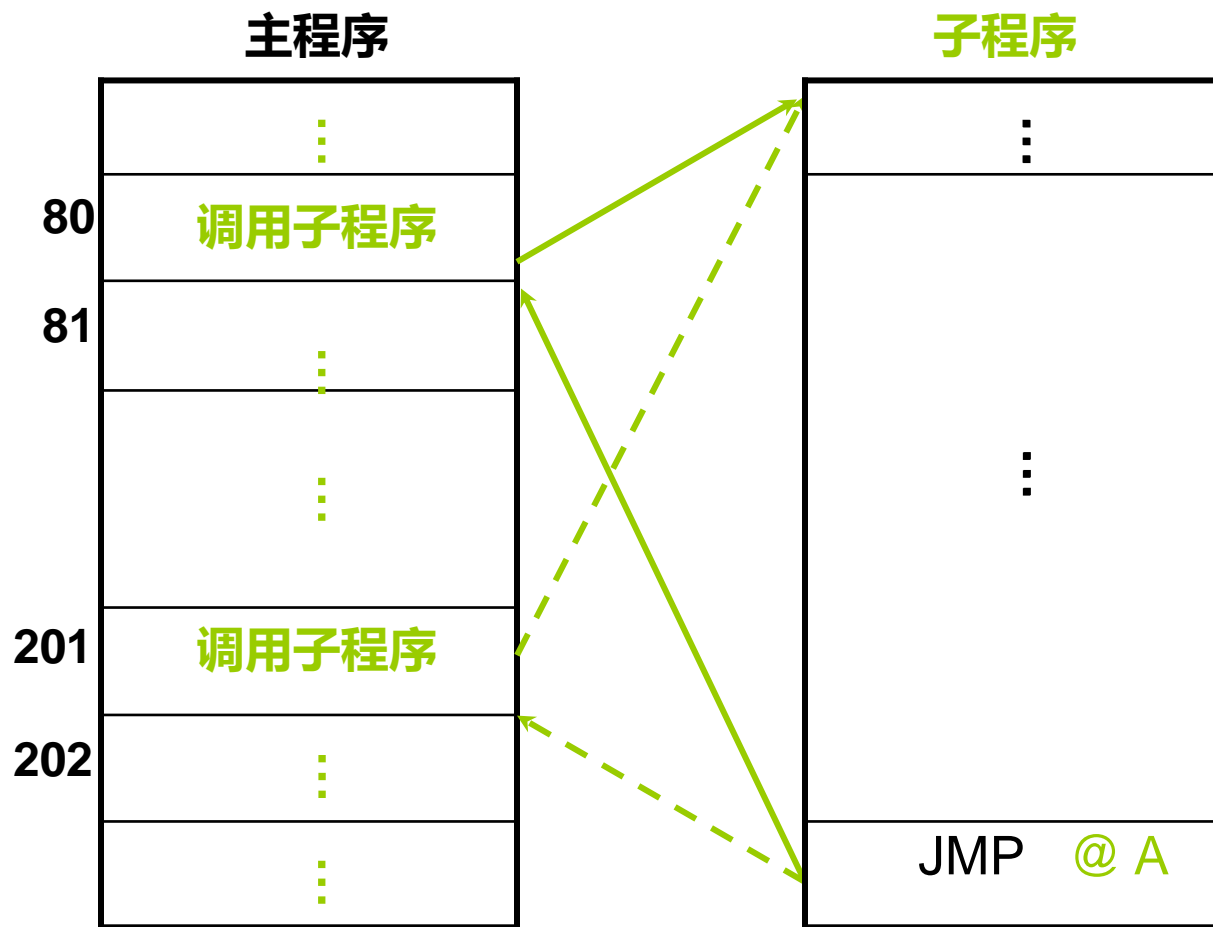


- 执行指令阶段 2 次访存
- 可扩大寻址范围
- 便于编制程序

多次访存



间接寻址编程举例



@ 间址特征

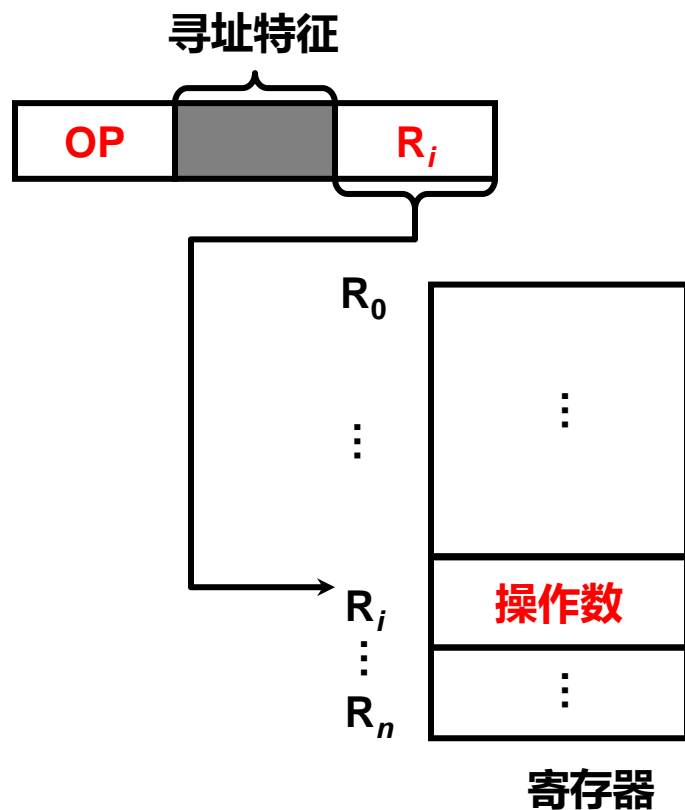
(A) = 202



5. 寄存器（直接）寻址

$$EA = R_i$$

有效地址即为寄存器编号



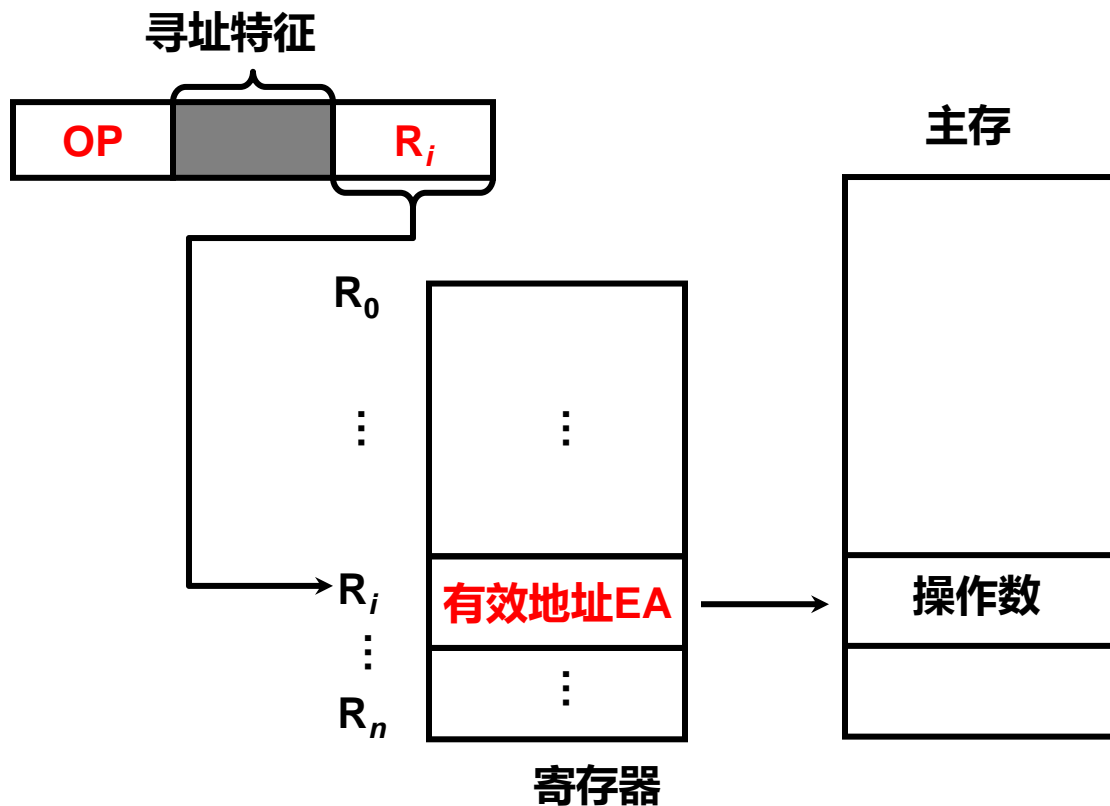
- 执行阶段不访存，只访问寄存器，执行速度快
- 寄存器个数有限，可缩短指令字长



6. 寄存器间接寻址

$$EA = (R_i)$$

有效地址在寄存器中



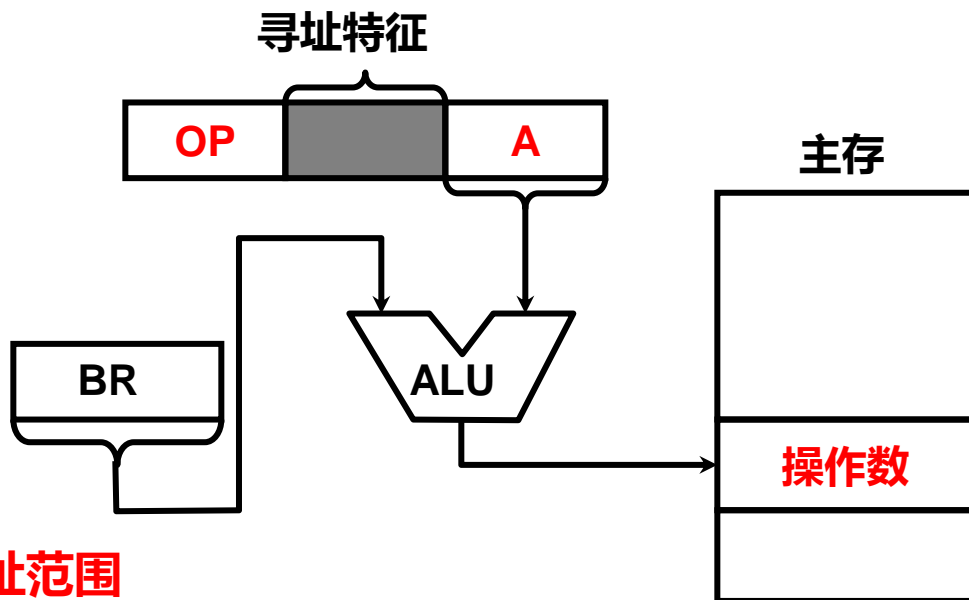
- 有效地址在寄存器中，操作数在存储器中，执行阶段访存
- 比一般间接寻址速度快



7. 基址寻址 (1) 采用专用寄存器作基址寄存器

$$EA = (BR) + A$$

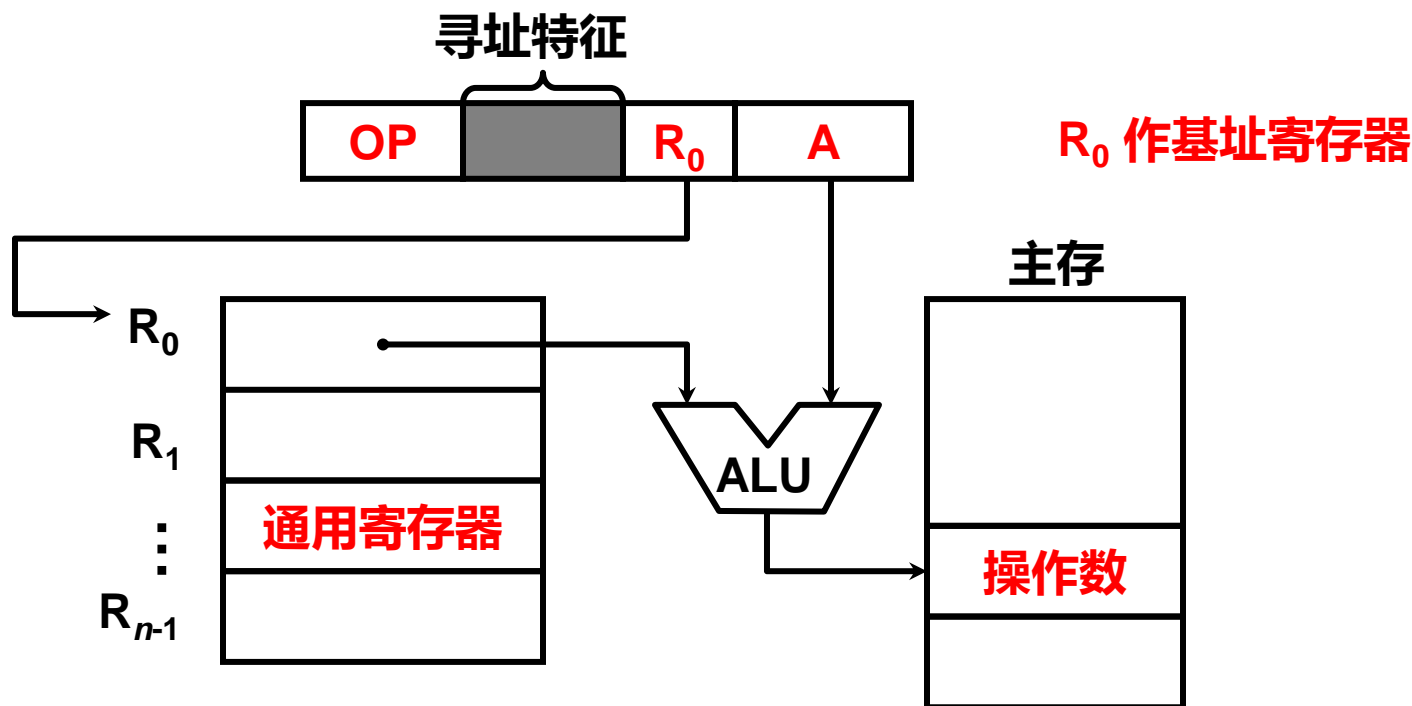
BR 为基址寄存器



- 可扩大寻址范围
- 有利于多道/浮动程序，但偏移量偏小
- BR 内容由操作系统或管理程序确定
- 在程序的执行过程中 BR 内容不变，形式地址 A 可变



(2) 采用通用寄存器作基址寄存器



- 由用户指定哪个通用寄存器作为基址寄存器
- 基址寄存器的内容由操作系统确定
- 在程序的执行过程中 **R₀** 内容不变，形式地址 **A** 可变

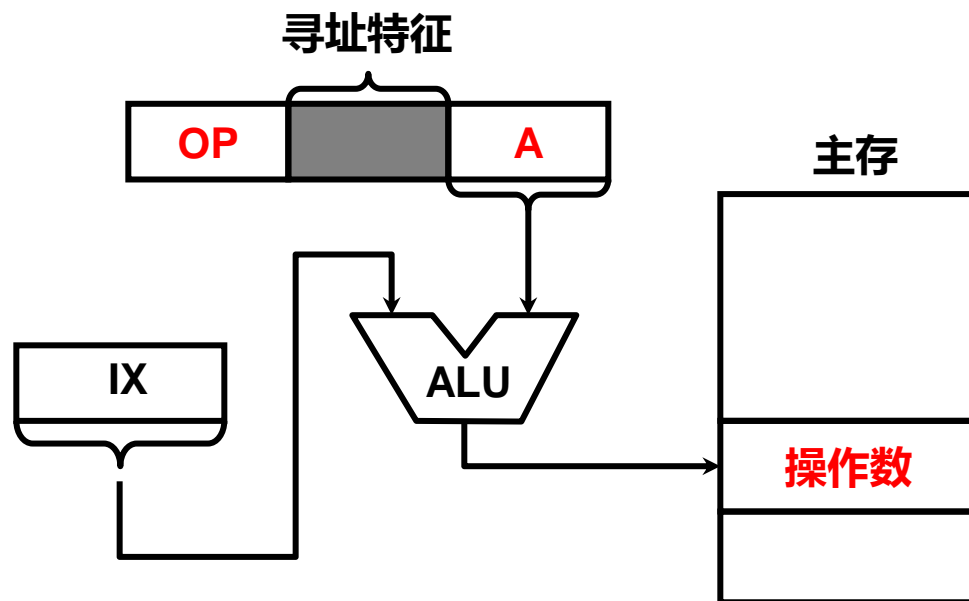


8. 变址寻址

$$EA = (IX) + A$$

IX 为变址寄存器 (专用)

通用寄存器也可以作为变址寄存器



- 可扩大寻址范围
- IX 的内容由用户给定
- 在程序的执行过程中 IX 内容可变, 形式地址 A 不变
- 便于处理循环/数组问题



例7.2 设数据块首地址为 D ，求 N 个数的平均值

直接寻址

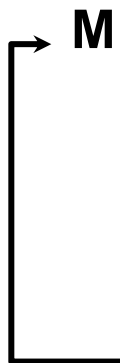
```
LDA  [D]
ADD  [D + 1]
ADD  [D + 2]
⋮
ADD  [D + ( N -1 )]
DIV  # N
STA  ANS
```

共 $N + 2$ 条指令

变址寻址

```
LDA  # 0           0 → ACC
LDX  # 0           X 为变址寄存器
ADD  X, [D]        D 为形式地址
INX                     (X) + 1 → X
CPX  # N           (X) 和 #N 比较
BNE  M             结果不为零则转
DIV  # N
STA  ANS
```

共 8 条指令



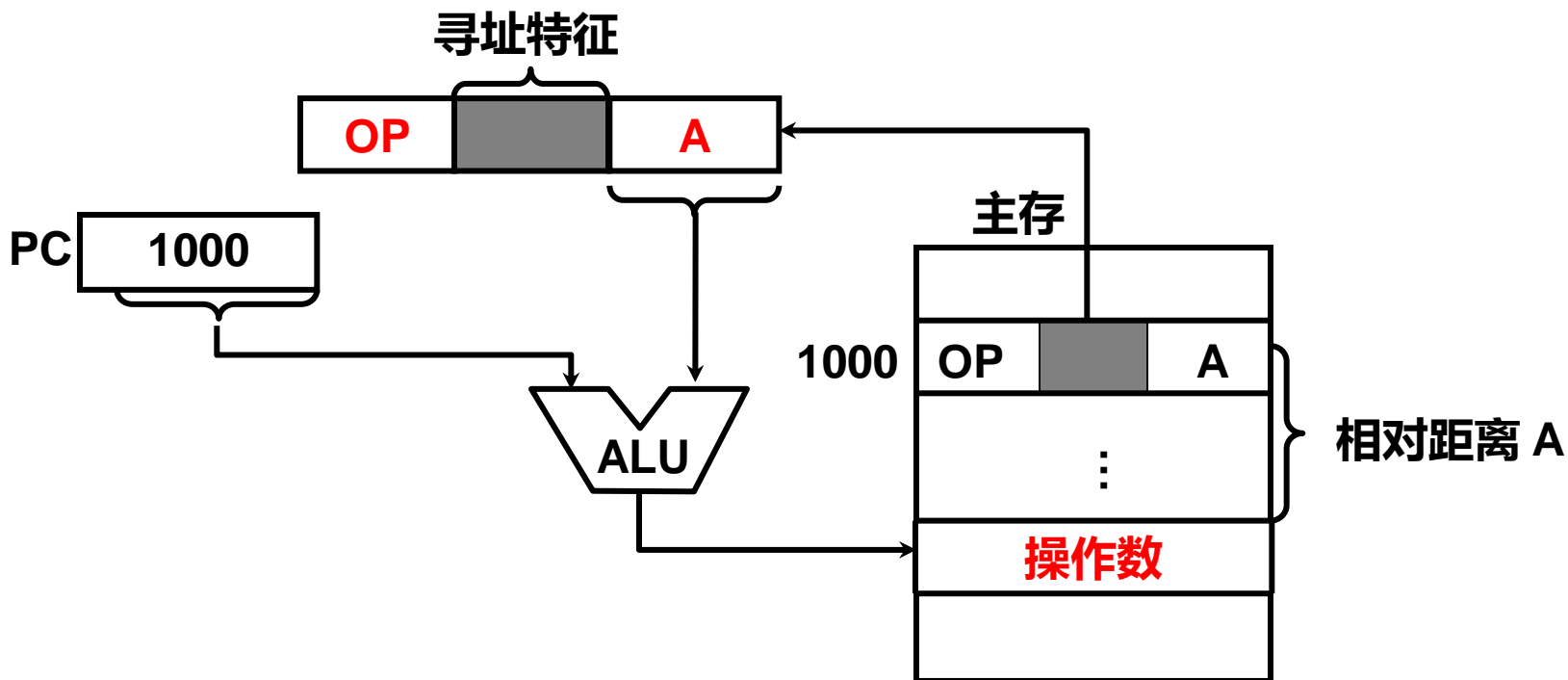


7.3 寻址方式

9. 相对寻址

$$EA = (PC) + A$$

A 是相对于当前指令的位移量 (可正可负, 补码)

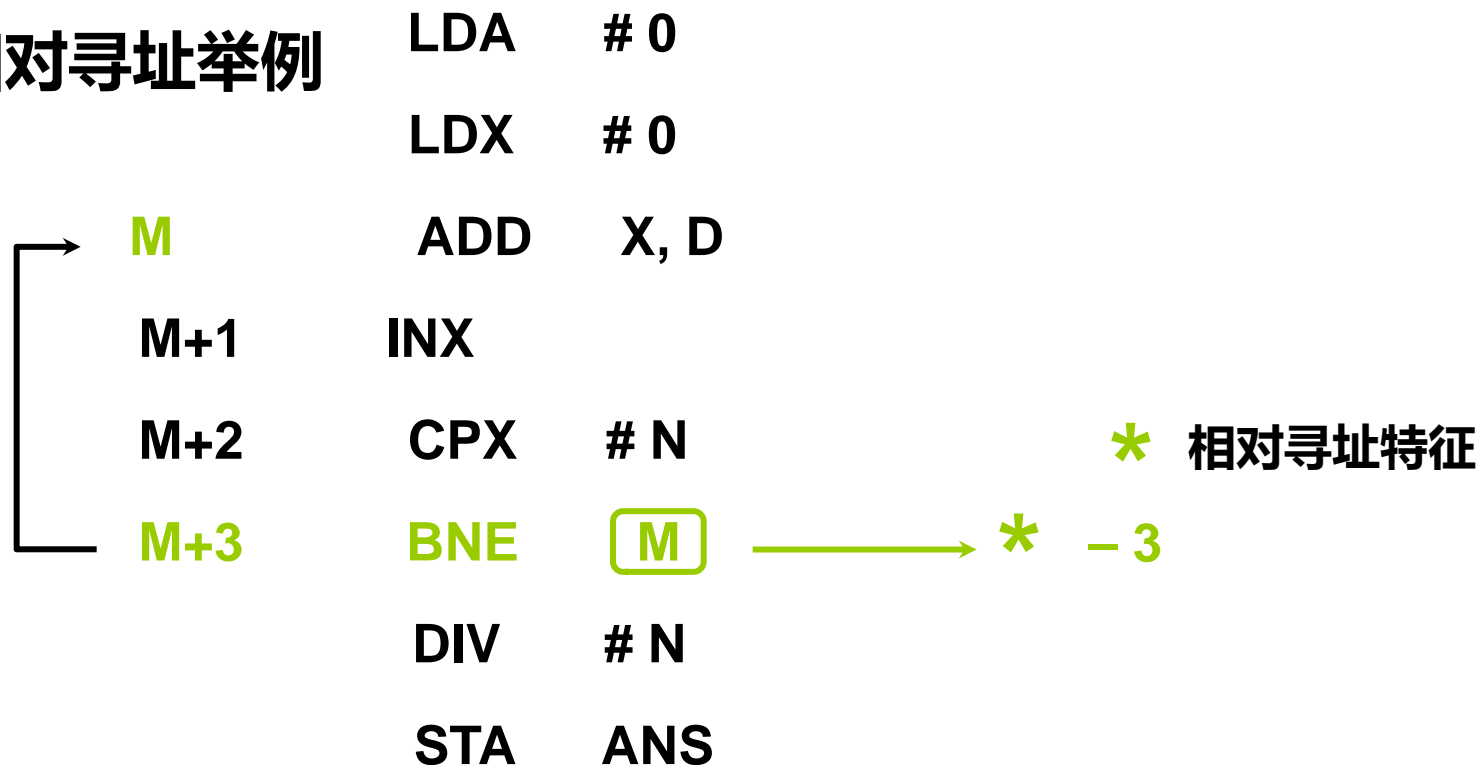


- A 的位数决定操作数的寻址范围
- 程序浮动
- 广泛用于转移指令



7.3 寻址方式

(1) 相对寻址举例



M 随程序所在存储空间的位置不同而不同

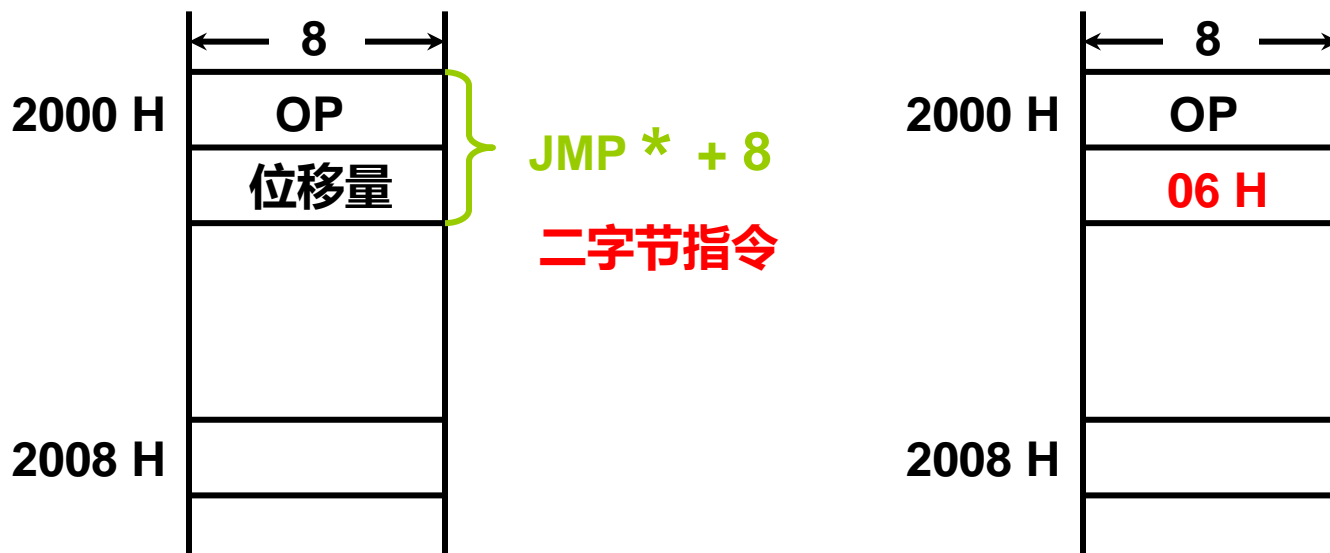
而指令 **BNE** $* - 3$ 与指令 **ADD** X, D 相对位移量不变

指令 **BNE** $* - 3$ 操作数的有效地址为

$$EA = (M+3) - 3 = M$$



(2) 按字节寻址的相对寻址举例



设 当前指令地址 $PC = 2000H$

转移后的目的地址为 $2008H$

因为 取出 $JMP * + 8$ 后 $PC = 2002H$

故 $JMP * + 8$ 指令 的第二字节为 $2008H - 2002H = 06H$



(1) 堆栈的特点

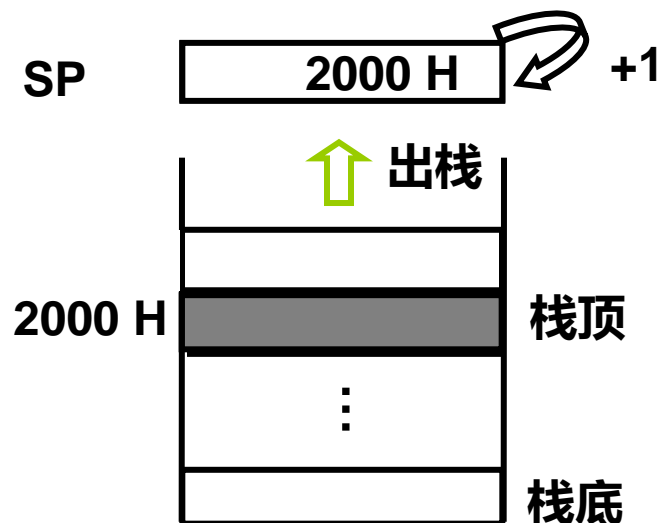
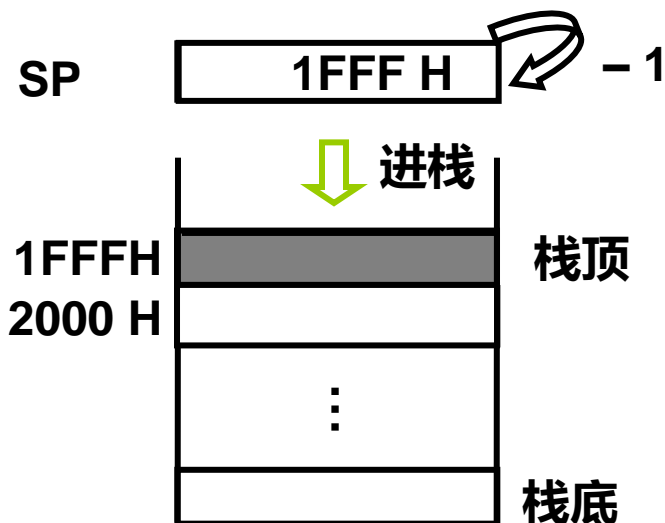
堆栈	硬堆栈	多个寄存器	堆栈计算机
	软堆栈	指定的存储空间	8086计算机

先进后出 (一个入出口)

栈顶地址 由 **SP** 指出

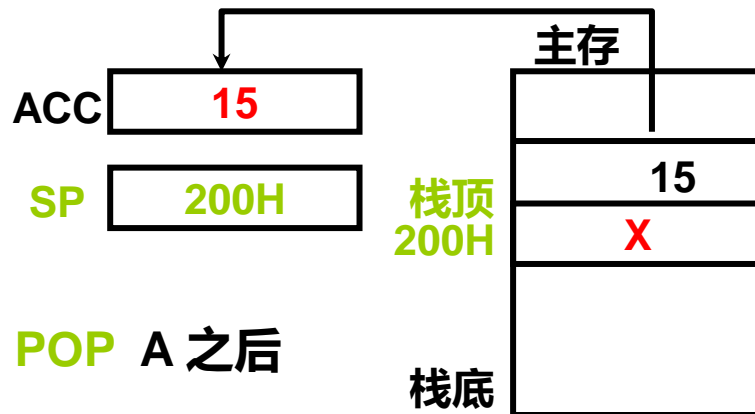
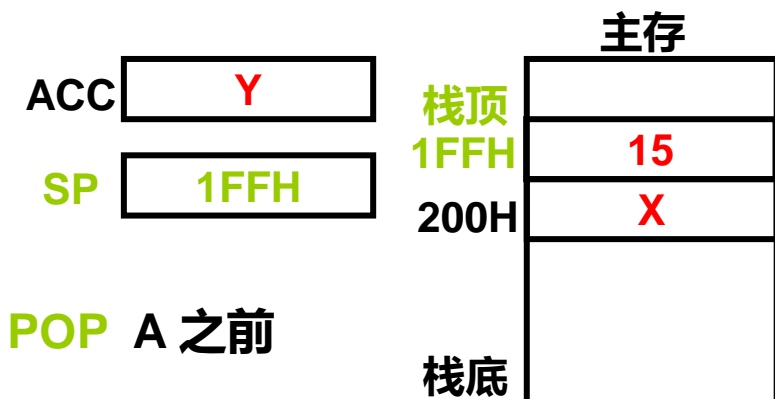
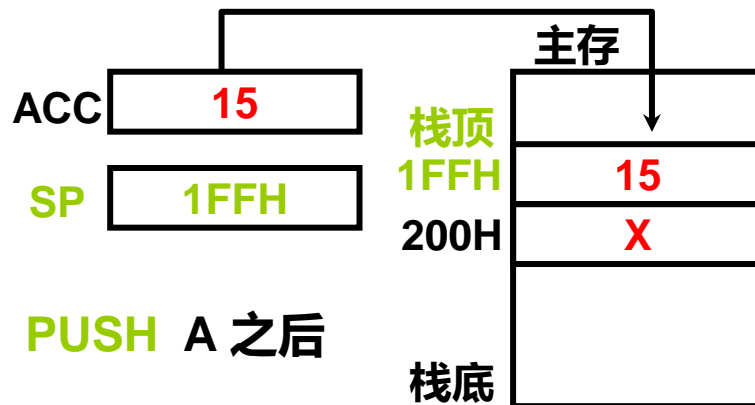
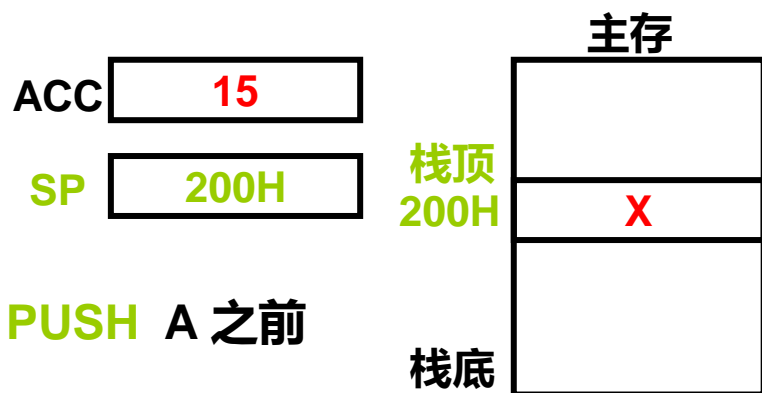
进栈 $(SP) - 1 \rightarrow SP$

出栈 $(SP) + 1 \rightarrow SP$





(2) 堆栈寻址举例





(3) SP 的修改与主存编址方法有关

① 按 字 编址

进栈 $(SP) - 1 \longrightarrow SP$

出栈 $(SP) + 1 \longrightarrow SP$

② 按 字节 编址

存储字长 16 位 进栈 $(SP) - 2 \longrightarrow SP$

出栈 $(SP) + 2 \longrightarrow SP$

存储字长 32 位 进栈 $(SP) - 4 \longrightarrow SP$

出栈 $(SP) + 4 \longrightarrow SP$



一、设计指令格式时应考虑的各种因素

1. 指令系统的 **兼容性** (向上兼容)

教材P320

2. 其他因素

操作类型

包括指令个数及操作的难易程度

数据类型

确定哪些数据类型可参与操作

指令格式

指令字长是否固定

操作码位数、是否采用扩展操作码技术,

地址码位数、地址个数、寻址方式类型

寻址方式

指令寻址、操作数寻址

寄存器个数

寄存器的多少直接影响指令的执行时间

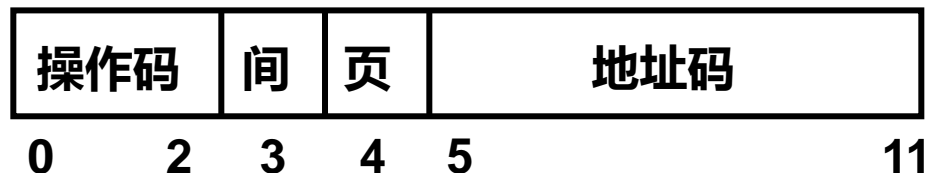


1. PDP – 8

指令字长固定 12 位

教材P320-321

访存类指令



I/O 类指令



寄存器类指令



采用扩展操作码技术



指令字长有 16 位、32 位、48 位三种

教材P321

扩展操作码技术



16

零地址 (16 位)



10

6

一地址 (16 位)



4

6

6

二地址 R – R (16 位)



10

6

16

二地址 R – M (32 位)



4

6

6

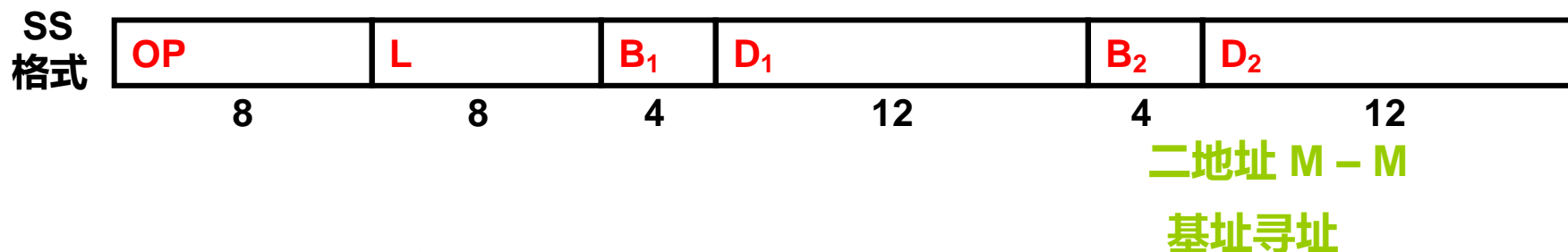
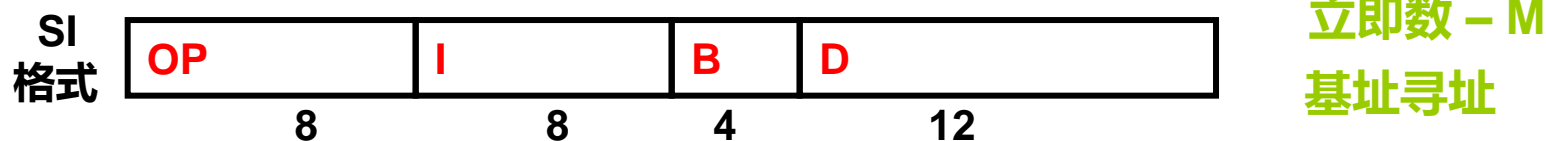
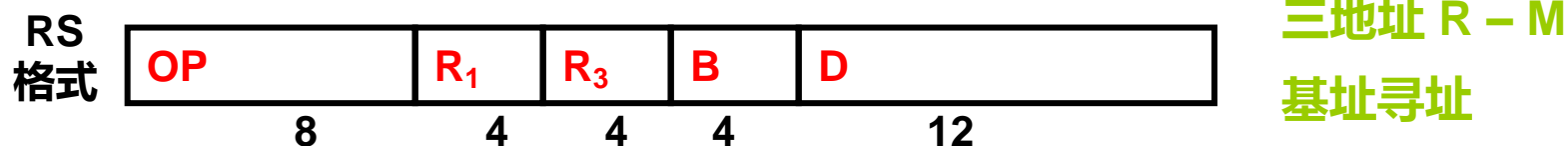
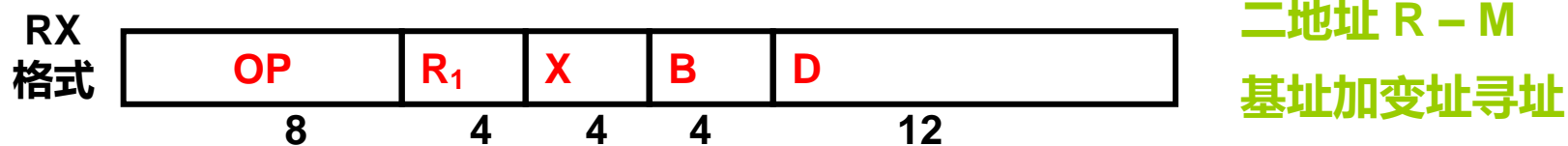
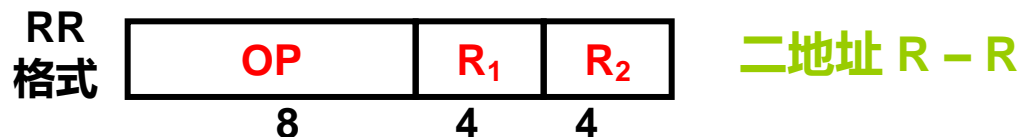
16

16

二地址 M – M (48 位)



教材P322





(1) 指令字长

1 ~ 6 个字节

INC AX 1 字节

MOV WORD PTR[0204], 0138H 6 字节

(2) 地址格式

零地址

NOP

1 字节

一地址

CALL

段间调用

5 字节

CALL

段内调用

3 字节

二地址

ADD AX, BX

2 字节

寄存器 – 寄存器

ADD AX, 3048H

3 字节

寄存器 – 立即数

ADD AX, [3048H]

4 字节

寄存器 – 存储器



合肥工业大学

HEFEI UNIVERSITY OF TECHNOLOGY

三、指令格式设计举例-自学* 7.4

教材P323-326



一、RISC 的产生和发展

RISC (Reduced Instruction Set Computer)

CISC (Complex Instruction Set Computer)

20—80规律

—— RISC技术

- 典型程序中 80% 的语句仅仅使用处理机中 20% 的指令
- 执行频度高的简单指令，因复杂指令的存在，执行速度无法提高
- ? 能否用 20% 的简单指令组合不常用的80% 的指令功能



- 选用使用频度较高的一些 **简单指令**，复杂指令的功能由简单指令来组合
- 指令 **长度固定、指令格式种类少、寻址方式少**
- 只有 **LOAD / STORE** 指令访存，其余指令的操作都在寄存器之间进行。
- CPU 中有 **多个通用寄存器**
- 采用 **流水技术**，大部分指令在一个 **时钟周期** 内完成
- 采用 **组合逻辑** 实现控制器
- 采用 **优化** 的 **编译** 程序



- 系统指令 **复杂庞大**，各种指令使用频度相差大
- 指令 **长度不固定、指令格式种类多、寻址方式多**
- **访存指令不受限制**
- CPU 中设有 **专用寄存器**
- 大多数指令需要 **多个时钟周期** 执行完毕
- 采用 **微程序** 控制器
- **难以用优化编译** 生成高效的代码



1. RISC更能 **充分利用 VLSI 芯片的面积**

2. RISC 更能 **提高计算机运算速度**

**指令数、指令格式、寻址方式少，
通用 寄存器多，采用 组合逻辑，
便于实现 指令流水**

3. RISC **便于设计，可 降低成本，提高 可靠性**

4. RISC **有利于编译程序代码优化**

5. RISC **不易 实现 指令系统兼容**



类别 对比项目	CISC	RISC
指令系统	复杂，庞大	简单，精简
指令数目	一般大于 200 条	一般小于 100 条
指令字长	不固定	定长
可访存指令	不加限制	只有 Load/Store 指令
各种指令执行时间	相差较大	绝大多数在一个周期内完成
各种指令使用频度	相差很大	都比较常用
通用寄存器数量	较少	多
目标代码	难以用优化编译生成高效的目标代码程序	采用优化的编译程序，生成代码较为高效
控制方式	绝大多数为微程序控制	绝大多数为组合逻辑控制
指令流水线	可以通过一定方式实现	必须实现



合肥工业大学

HEFEI UNIVERSITY OF TECHNOLOGY

锲而舍之，朽木不折；
锲而不舍，金石可镂。

计算机组成原理