计算机组成原理

第九讲

刘松波

哈尔滨工业大学计算学部

模式识别与智能系统计算研究中心

第 7 章　指 令 系 统

7.1 机器指令

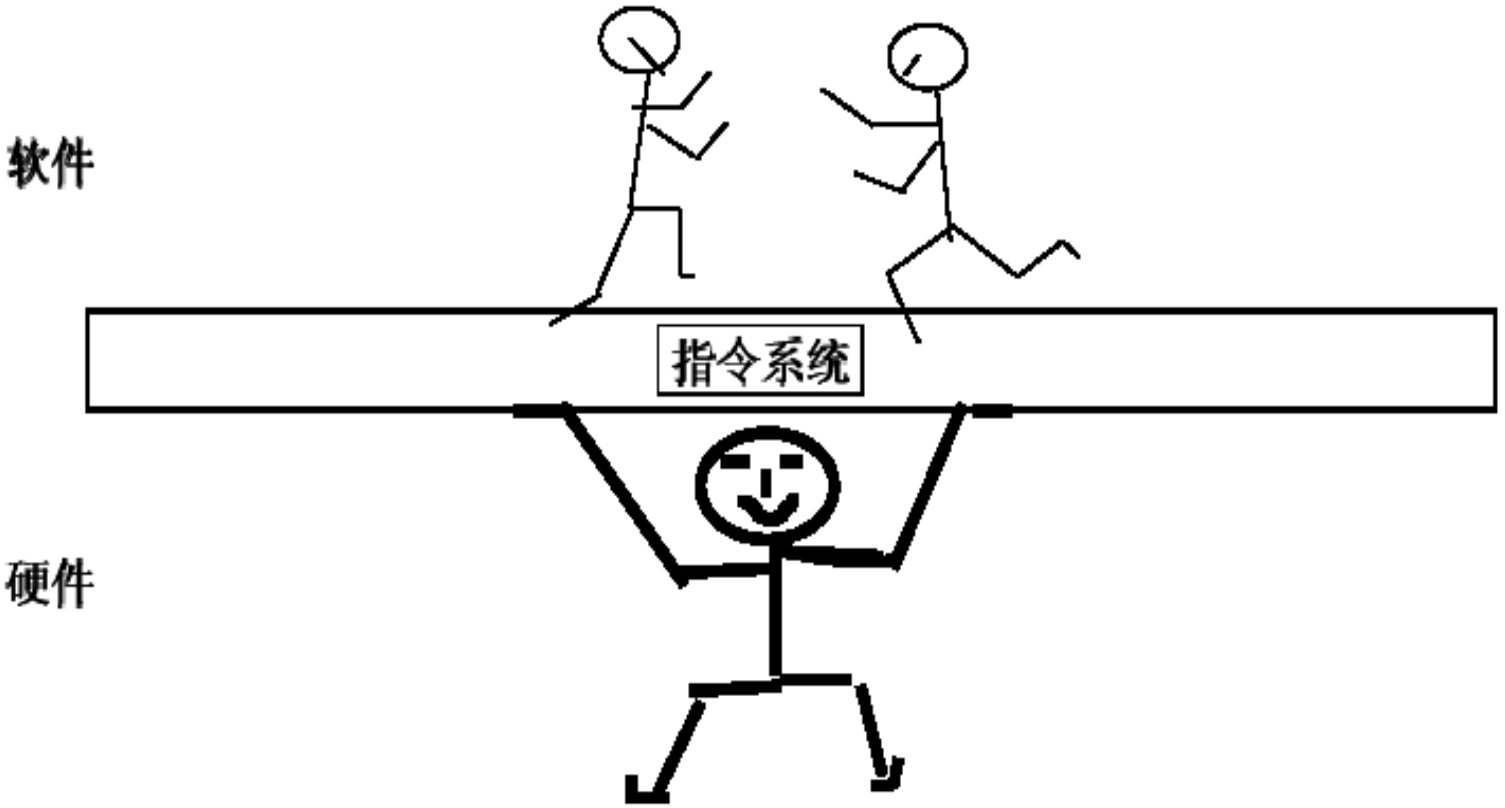
7.2 操作数类型和操作类型

7.3 寻址方式

7.4 指令格式举例

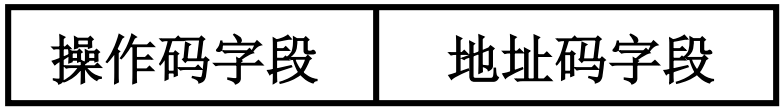
7.5 RISC 技术

指令系统在计算机中的地位



7.1 机 器 指 令

一、指令的一般格式



1. 操作码　反映机器做什么操作

(1) 长度固定

用于指令字长较长的情况 ， **RISC**

如 **IBM 370** 操作码 **8** 位

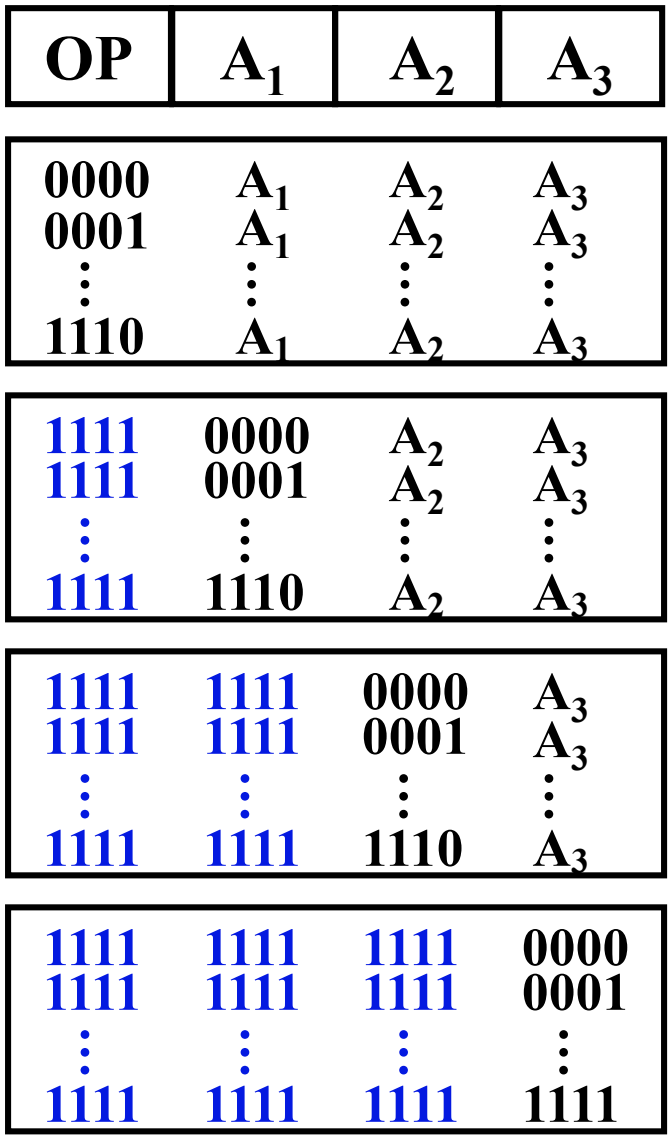
(2) 长度可变

操作码分散在指令字的不同字段中

(3) 扩展操作码技术

7.1

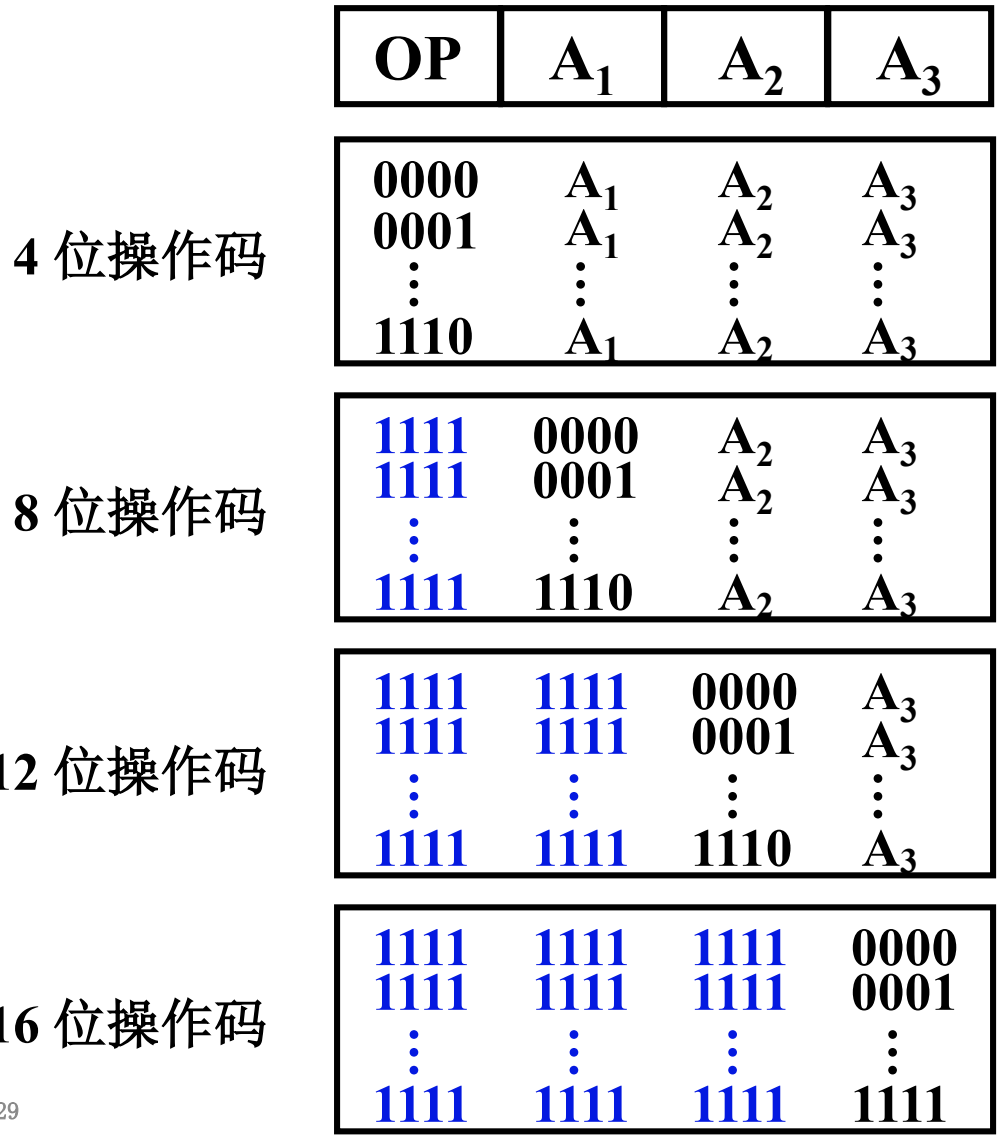
操作码的位数随地址数的减少而增加



(3) 扩展操作码技术

7.1

操作码的位数随地址数的减少而增加

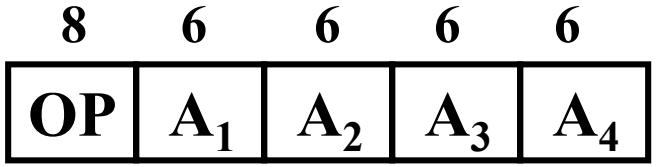


三地址指令操作码
每减少一种最多可多构成
2^4 种二地址指令

二地址指令操作码
每减少一种最多可多
构成2^4 种一地址指令

2. 地址码

(1) 四地址



A_1 第一操作数地址

A_2 第二操作数地址

A_3 结果的地址

A_4 下一条指令地址

(A_1) OP $(A_2) \longrightarrow A_3$

设指令字长为 32 位

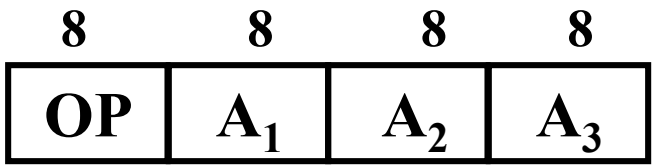
操作码固定为 8 位

4 次访存

寻址范围 $2^6 = 64$

若 PC 代替 A_4

(2) 三地址



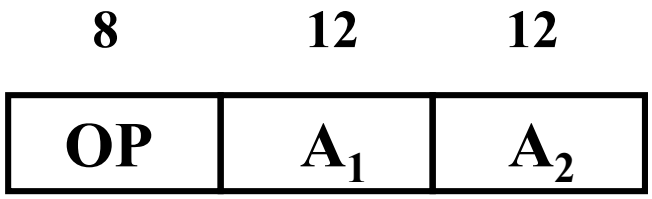
(A_1) OP $(A_2) \longrightarrow A_3$

4 次访存

寻址范围 $2^8 = 256$

若 A_3 用 A_1 或 A_2 代替

(3) 二地址



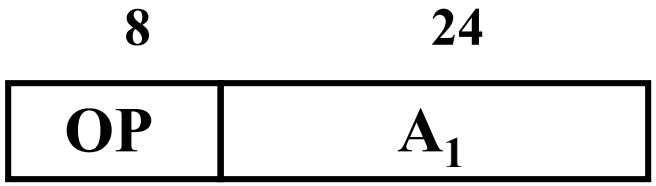
(A_1) OP $(A_2) \longrightarrow A_1$　　4 次访存

或

(A_1) OP $(A_2) \longrightarrow A_2$　　寻址范围 $2^{12} = 4$ K

若结果存于 ACC　3次访存　　若ACC 代替 A_1（或A_2）

(4) 一地址



(ACC) OP $(A_1) \longrightarrow$ ACC　　寻址范围 $2^{24} = 16$ M

(5) 零地址　　无地址码

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

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

7.2 操作数类型和操作种类

一、操作数类型

地址	无符号整数
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01234567

字地址

0	3	2	1	0
4	7	6	5	4

字地址 为 低字节 地址

字地址

0	0	1	2	3
4	4	5	6	7

字地址 为 高字节 地址

存储器中的数据存放（存储字长为32位）

7.2

边界对准

地址（十进制）

字（地址 0）				0
字（地址 4）				4
字节（地址11）	字节（地址10）	字节（地址 9）	字节（地址 8）	8
字节（地址15）	字节（地址14）	字节（地址13）	字节（地址12）	12
半字（地址18）✓		半字（地址16）✓		16
半字（地址22）✓		半字（地址20）✓		20
双字（地址24）▲				24
双字				28
双字（地址32）▲				32
双字				36

边界未对准

地址（十进制）

字(地址2)		半字(地址0)	0
字节(地址7)	字节(地址6)	字(地址4)	4
半字(地址10)		半字(地址8)	8

三、操作类型

7.2

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 MOVE PUSH	LOAD MOVE POP	MOVE

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

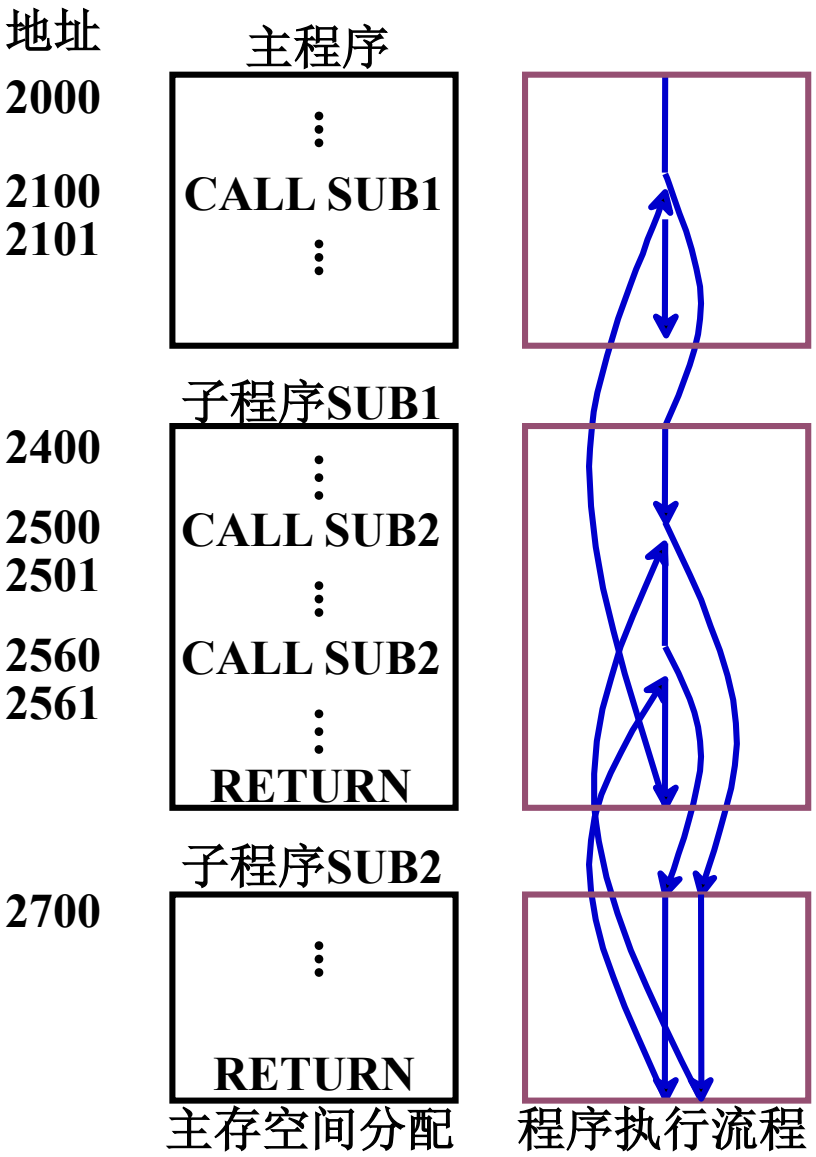
跳过一条指令　SKP

如

```
300
 ⋮
305  SKP  DZ;  D = 0 则跳
306
307
```

完成触发器

(3) 调用和返回


地址
主程序
2000
2100
CALL SUB1
2101
子程序SUB1
2400
2500
CALL SUB2
2501
2560
CALL SUB2
2561
RETURN
子程序SUB2
2700
RETURN
主存空间分配
程序执行流程

(4) 陷阱（Trap）与陷阱指令

7.2

意外事故的中断

- 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 ⟶ CPU 的寄存器

如　　IN AX, *m*　　IN AX, DX

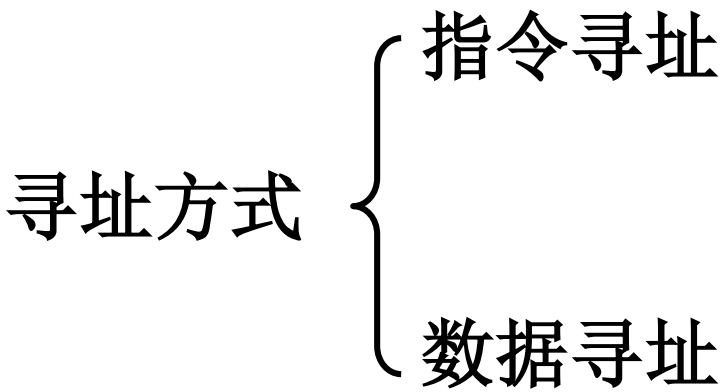
出　　CPU 的寄存器 ⟶ 端口地址

如　　OUT *n*, AX　　OUT DX, AX

7.3 寻址方式

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

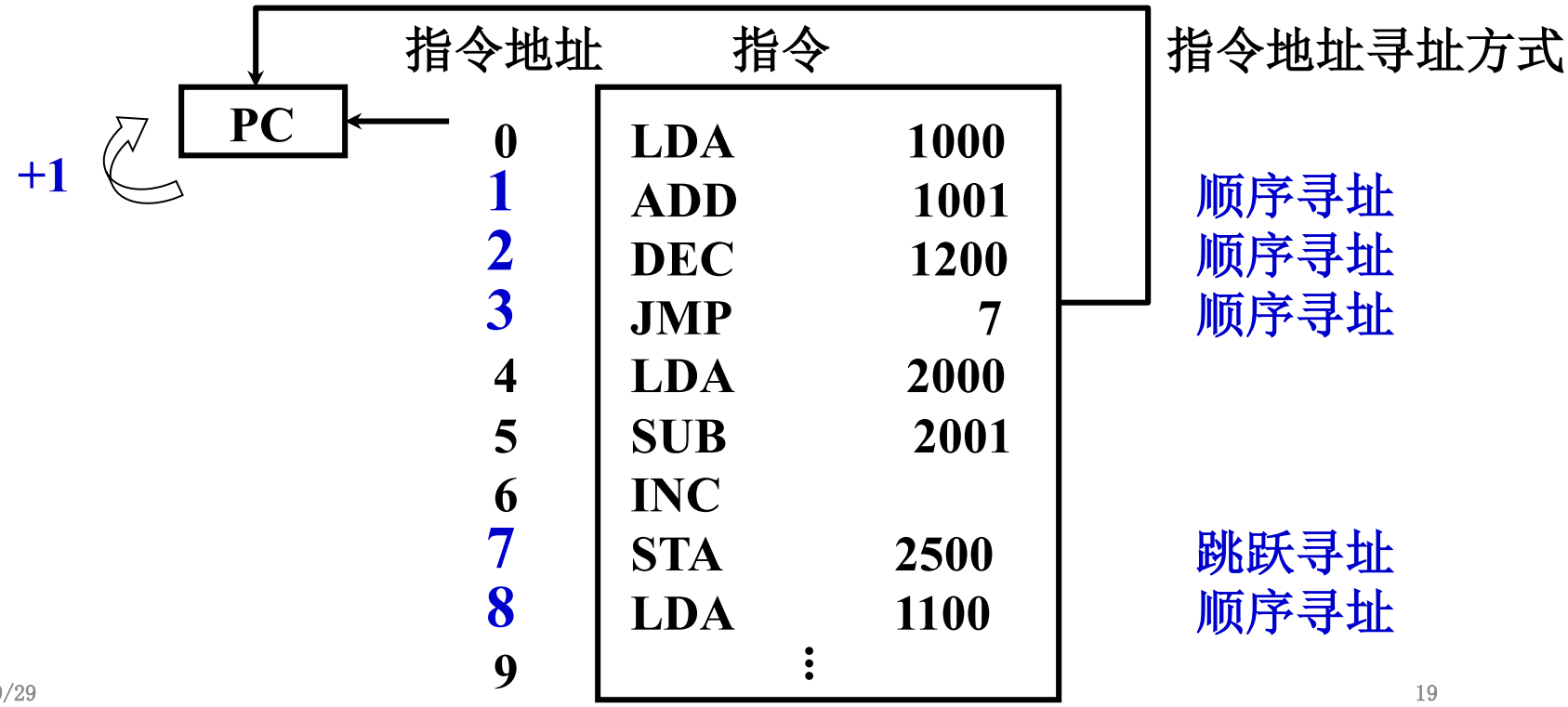


7.3 寻址方式

一、指令寻址

顺序　　(PC) + 1 ⟶ PC

跳跃　　由转移指令指出



二、数据寻址

7.3

操作码	寻址特征	形式地址 A

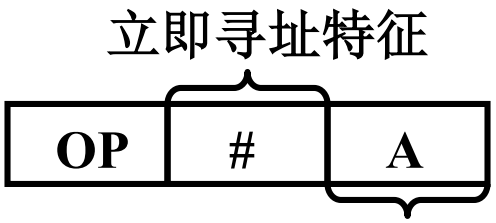
形式地址　　指令字中的地址

有效地址　　操作数的真实地址

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



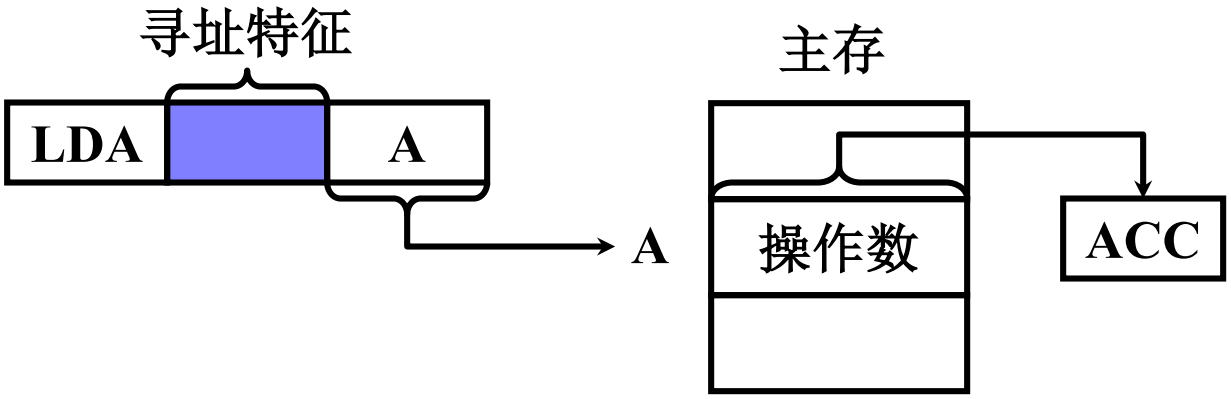
可正可负　补码

- 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

7.3

EA = A　　有效地址由形式地址直接给出

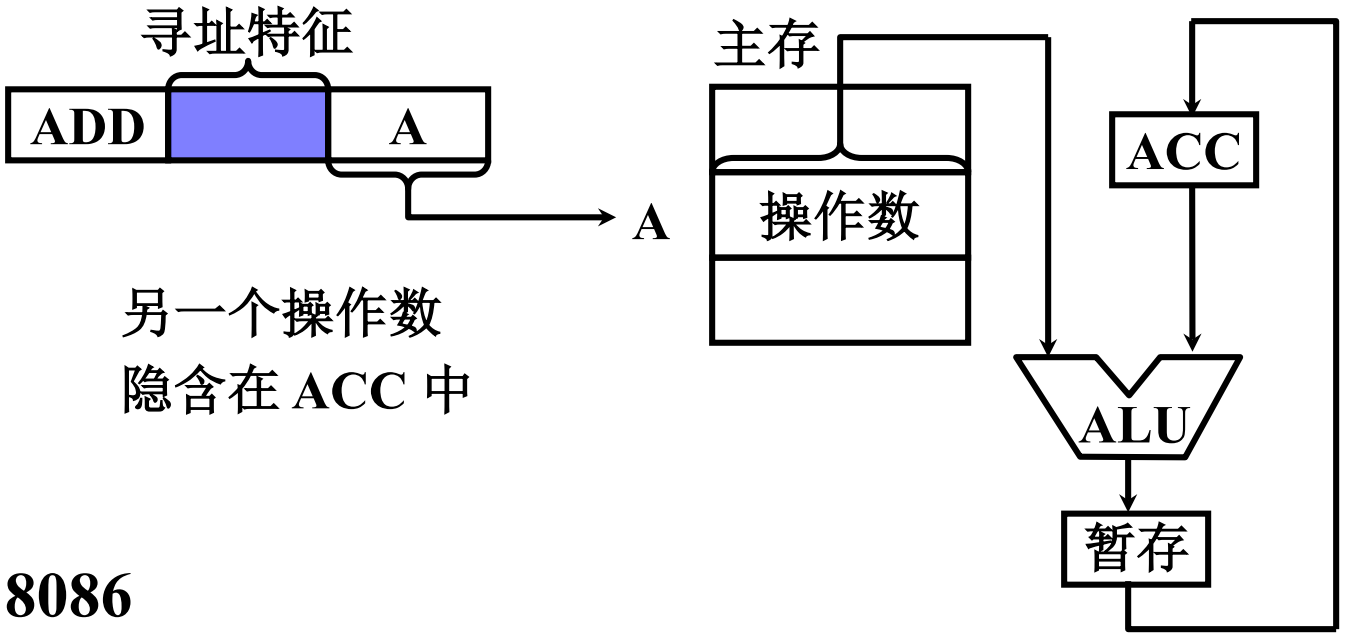


- 执行阶段访问一次存储器
- A 的位数决定了该指令操作数的寻址范围
- 操作数的地址不易修改（必须修改A）

3. 隐含寻址

7.3

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

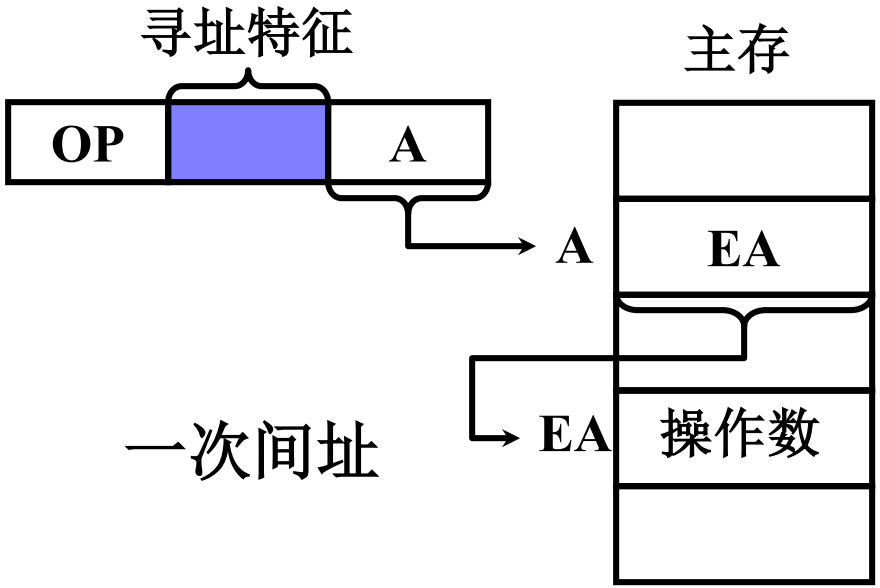
目的操作数的地址隐含在 DI 中

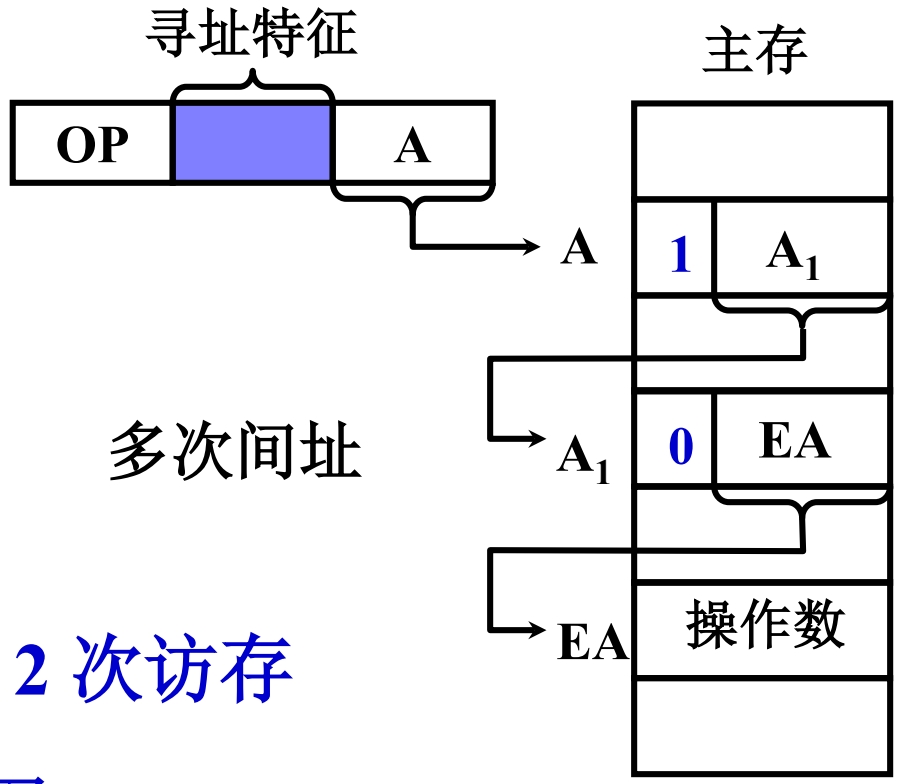
- 指令字中少了一个地址字段，可缩短指令字长

4. 间接寻址

7.3

EA＝（A）　有效地址由形式地址间接提供



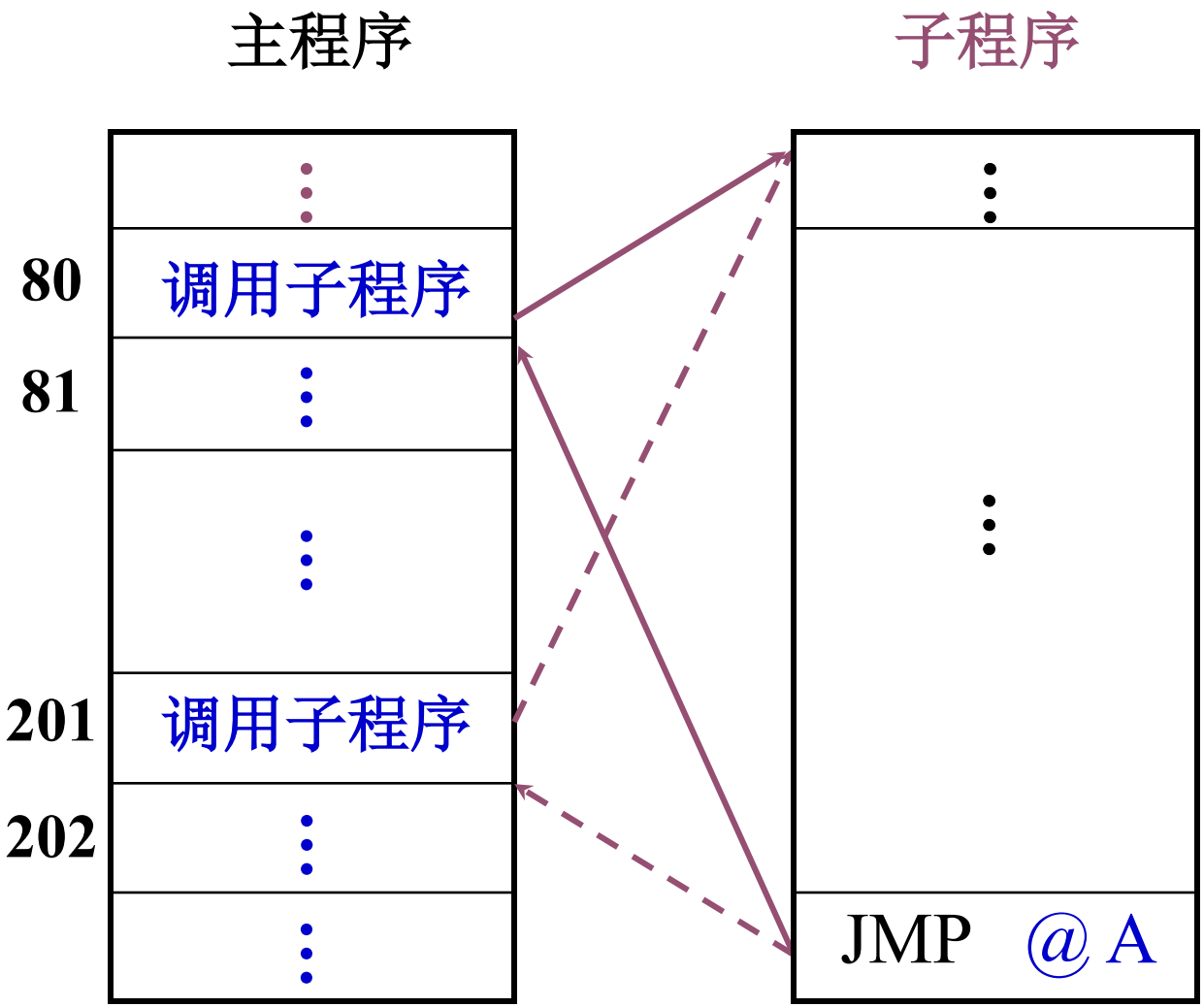


- 执行指令阶段 2 次访存
- 可扩大寻址范围
- 便于编制程序

多次访存

间接寻址编程举例

7.3



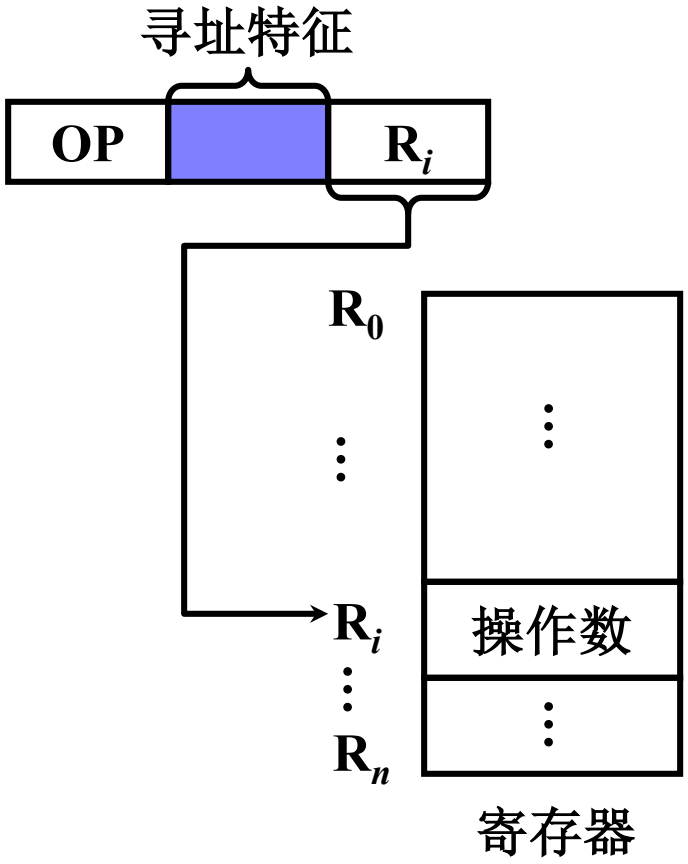
@ 间址特征

(A) = 202

5. 寄存器寻址

7.3

$EA = R_i$　　有效地址即为寄存器编号

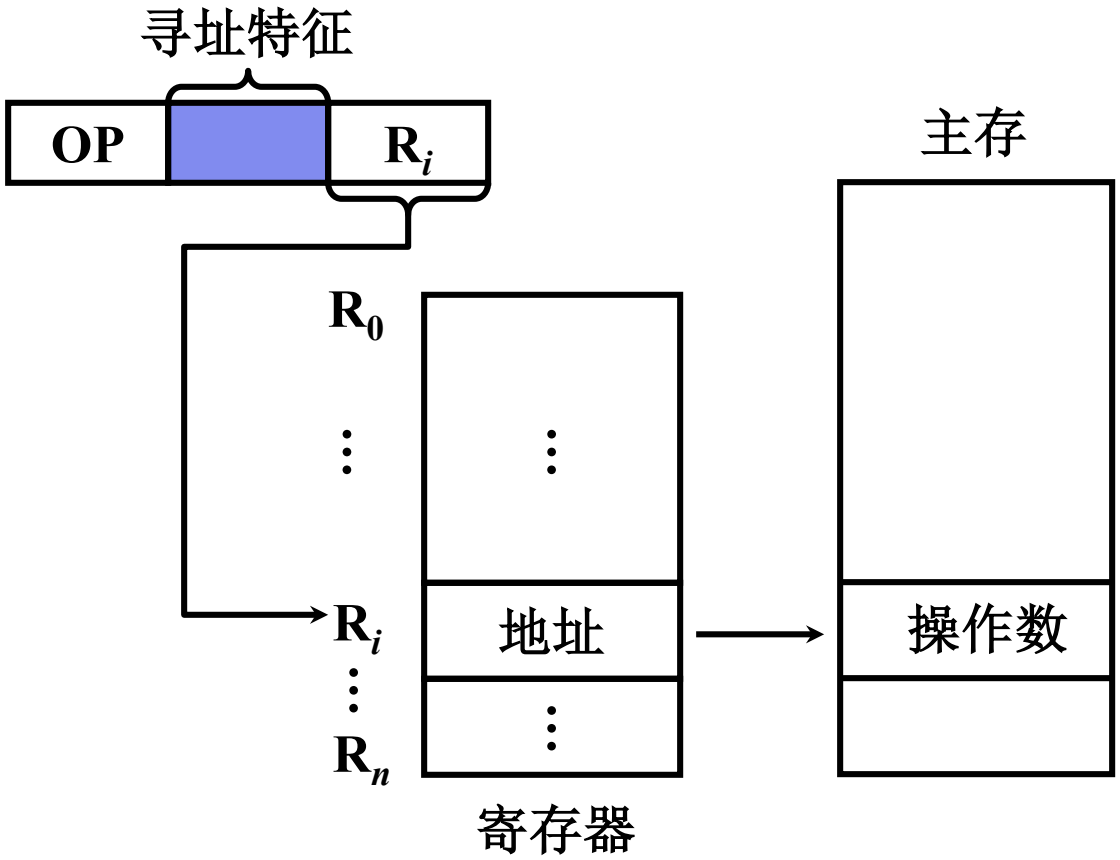


- 执行阶段不访存，只访问寄存器，执行速度快
- 寄存器个数有限，可缩短指令字长

6. 寄存器间接寻址

7.3

$EA = (R_i)$　　有效地址在寄存器中



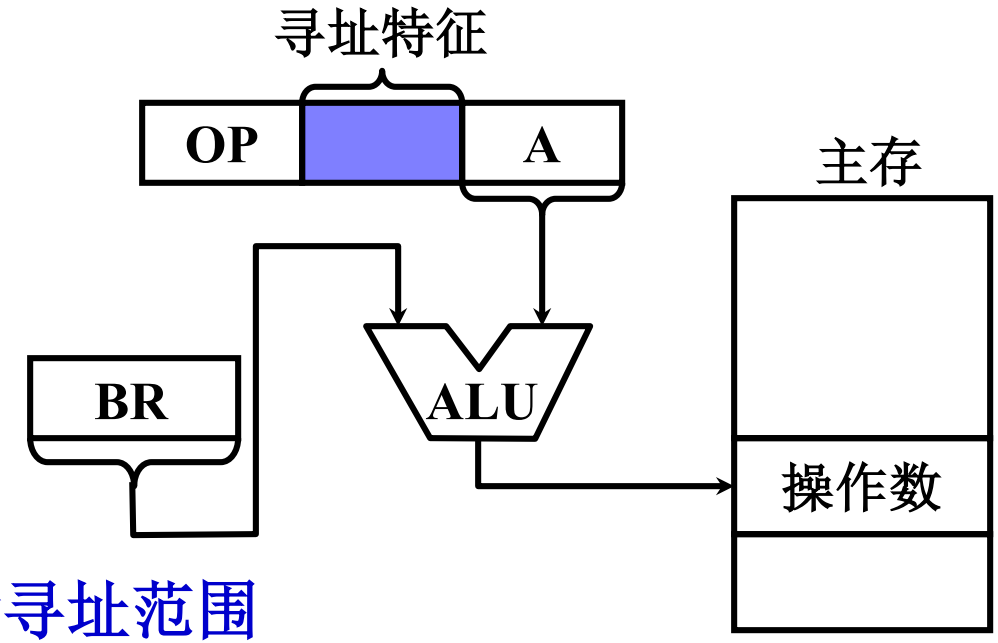
- 有效地址在寄存器中， 操作数在存储器中，执行阶段访存
- 便于编制循环程序

7. 基址寻址

7.3

(1) 采用专用寄存器作基址寄存器

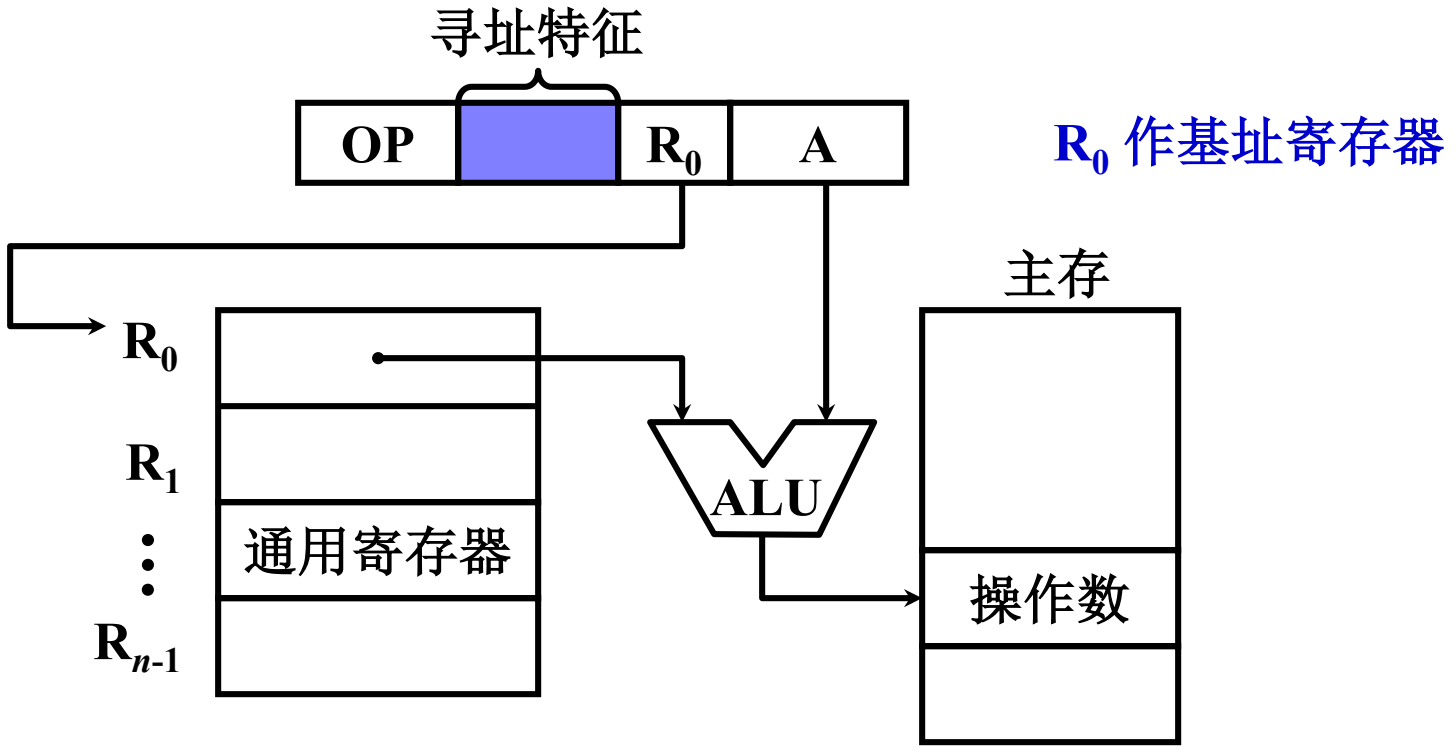
EA = (BR) + A　　BR 为基址寄存器



- 可扩大寻址范围
- 有利于多道程序
- 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

7.3



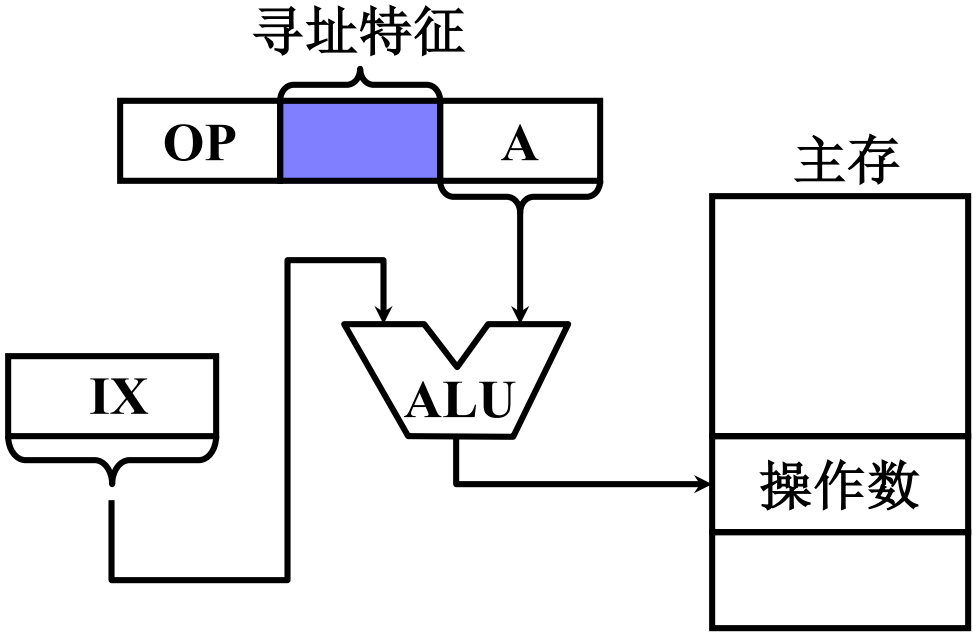
- 由用户指定哪个通用寄存器作为基址寄存器
- 基址寄存器的内容由操作系统确定
- 在程序的执行过程中 R_0 内容不变，形式地址 A 可变

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
- 便于处理数组问题

例　设数据块首地址为 D，求 *N* 个数的平均值　7.3

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
       LDX    # 0
M      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
```

X 为变址寄存器

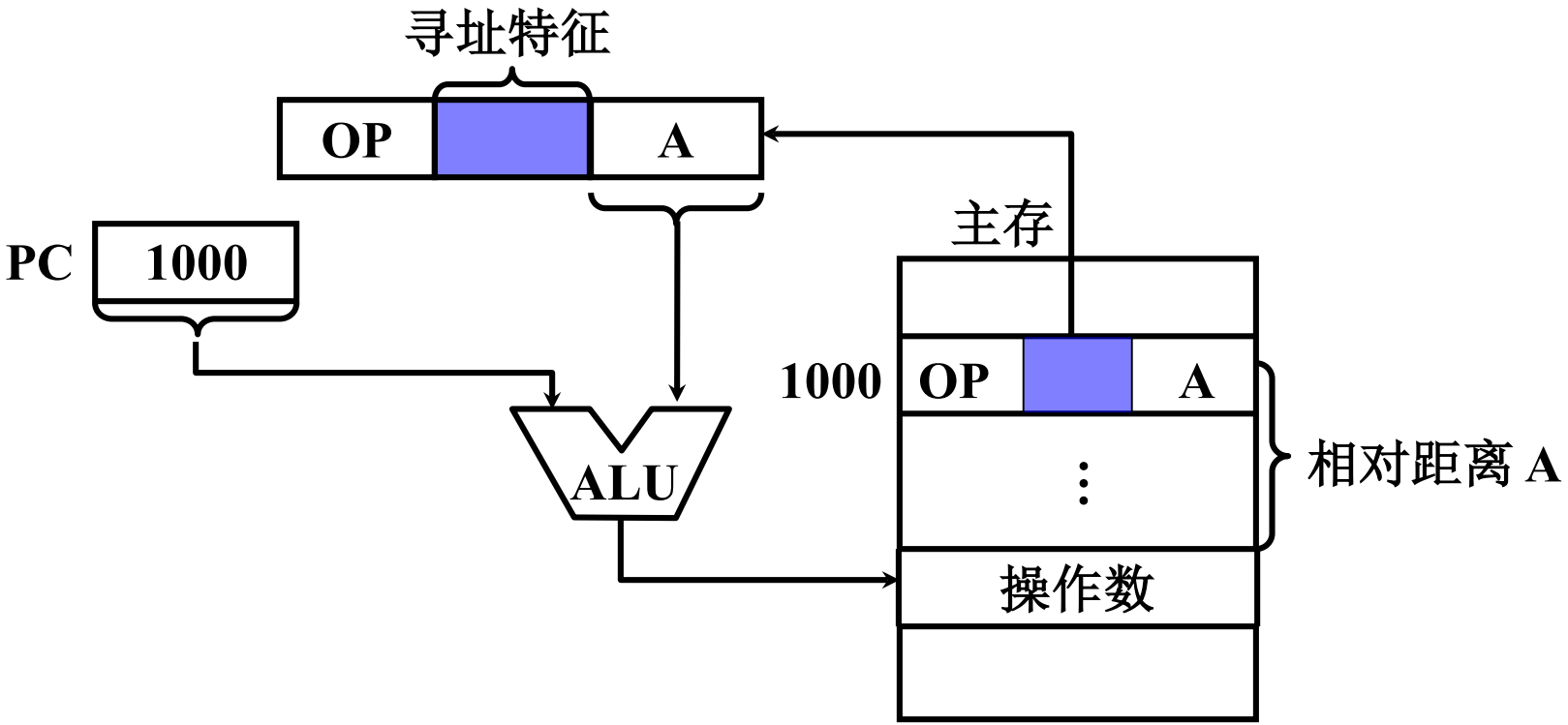
共 8 条指令

9. 相对寻址

7.3

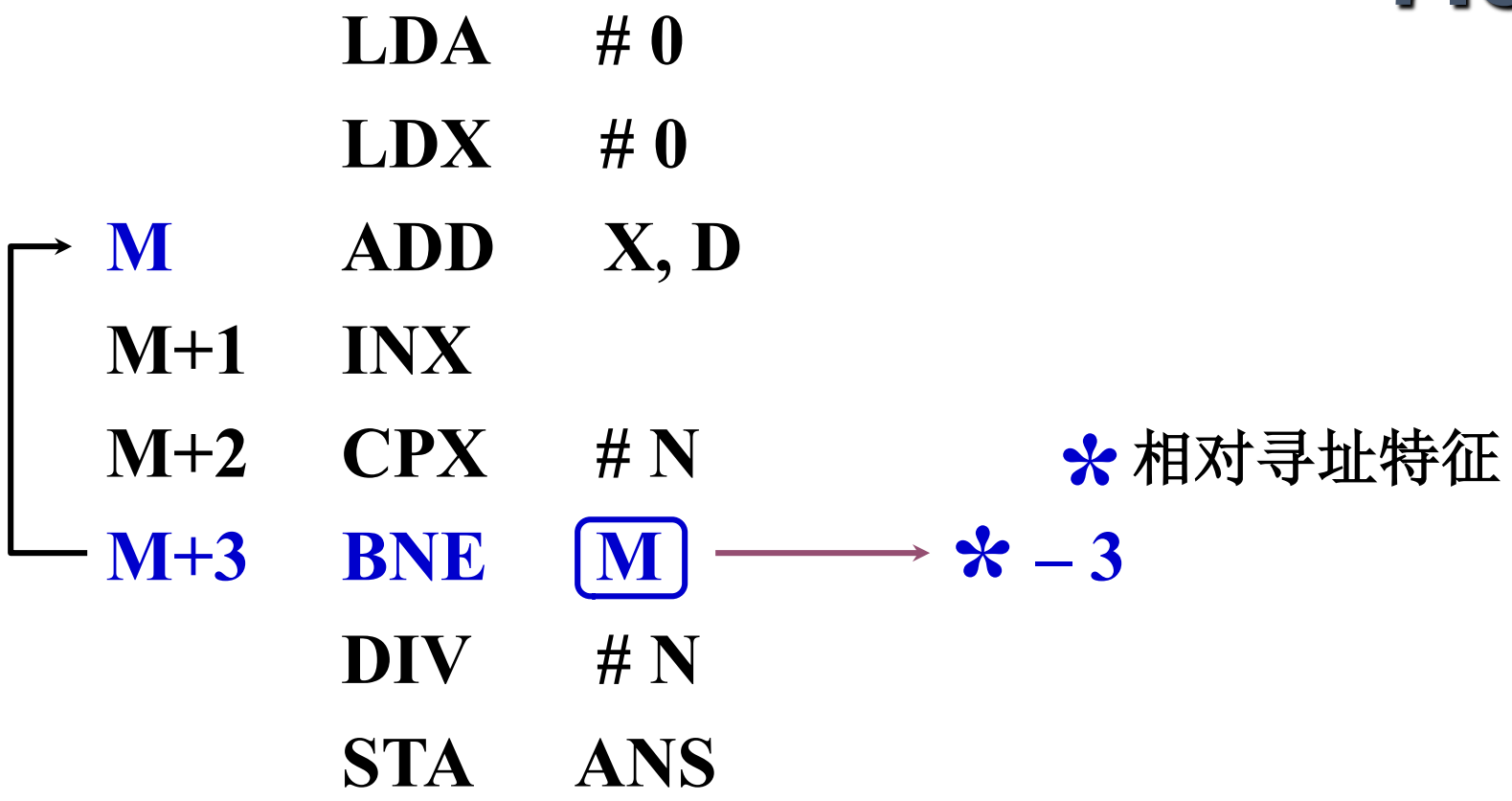
$EA = (PC) + A$

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

(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

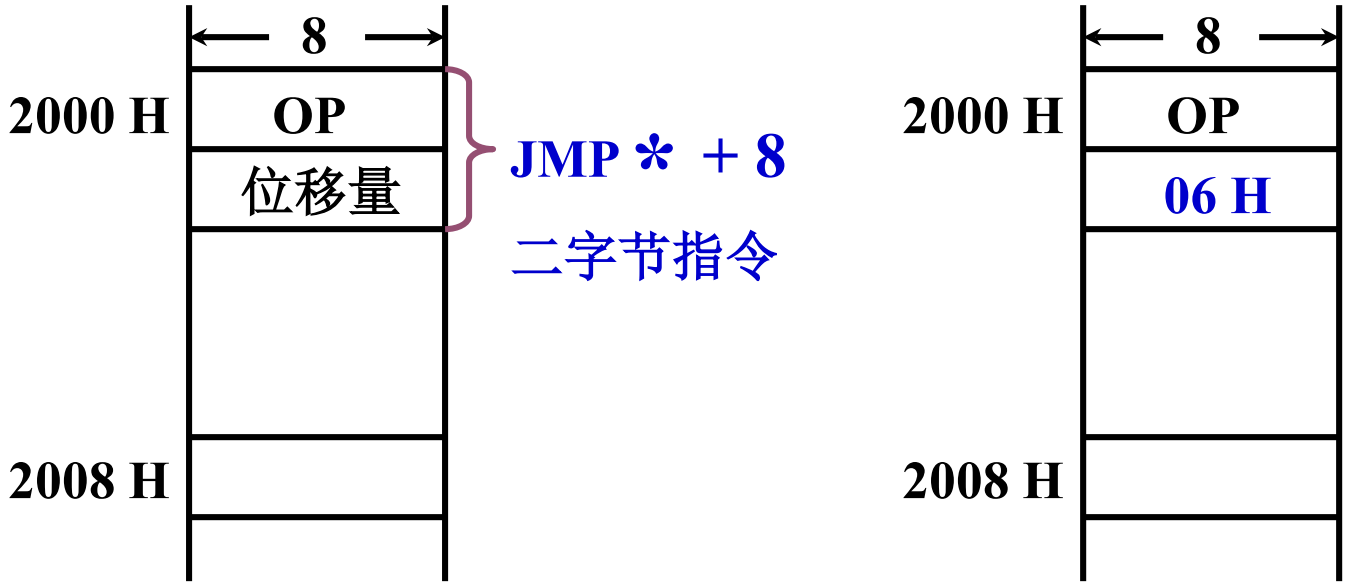
而指令 BNE * – 3 与 指令 ADD X, D 相对位移量不变

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

7.3



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

10. 堆栈寻址

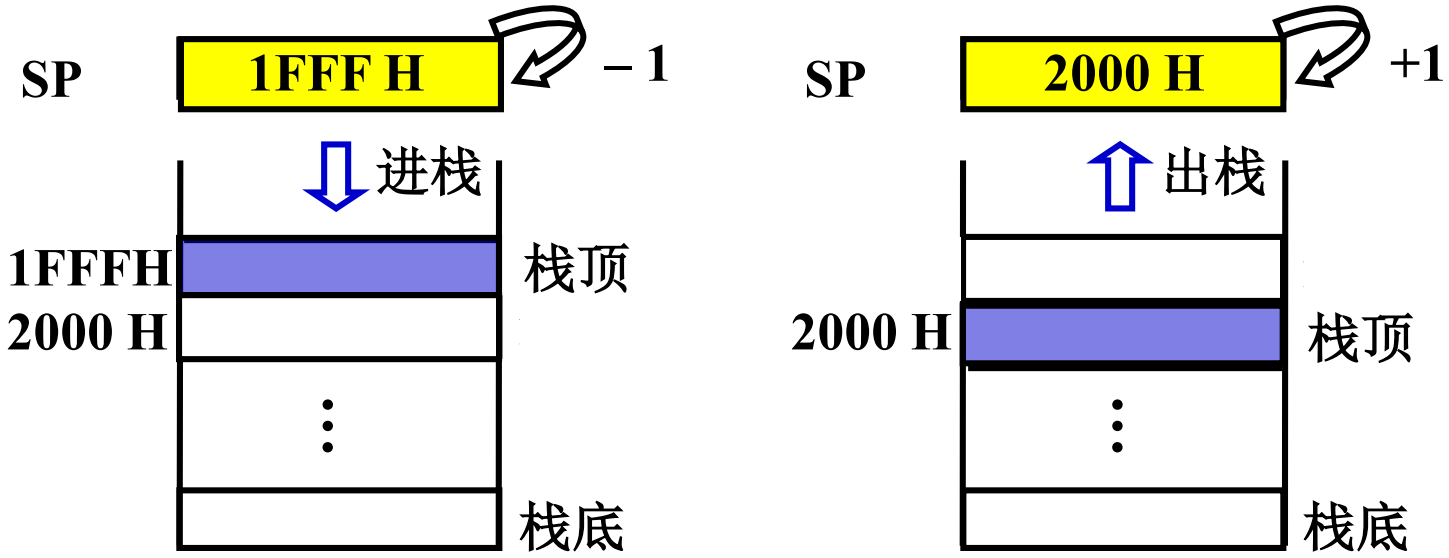
7.3

(1) 堆栈的特点

堆栈 { 硬堆栈　多个寄存器
　　 软堆栈　指定的存储空间

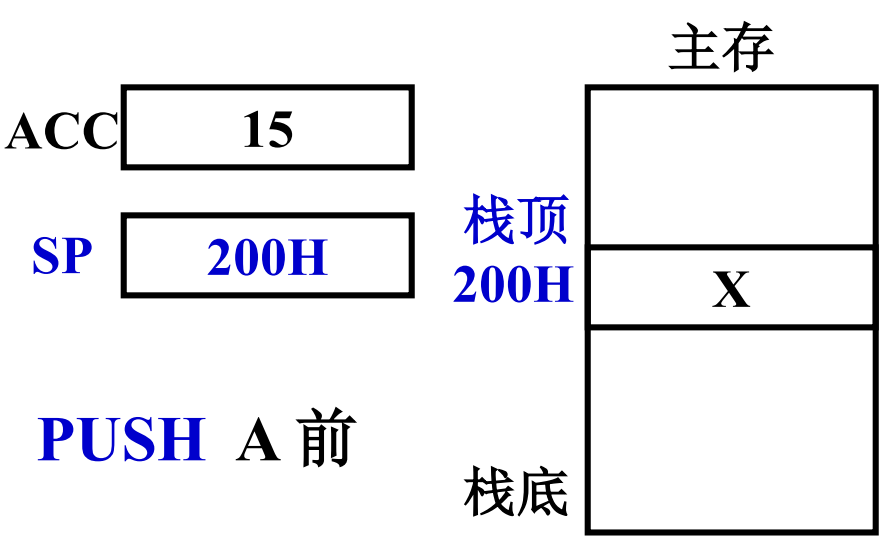
先进后出（一个入出口）　栈顶地址 由 SP 指出

进栈 （SP）– 1 ⟶ SP　　出栈 （SP）+ 1 ⟶ SP

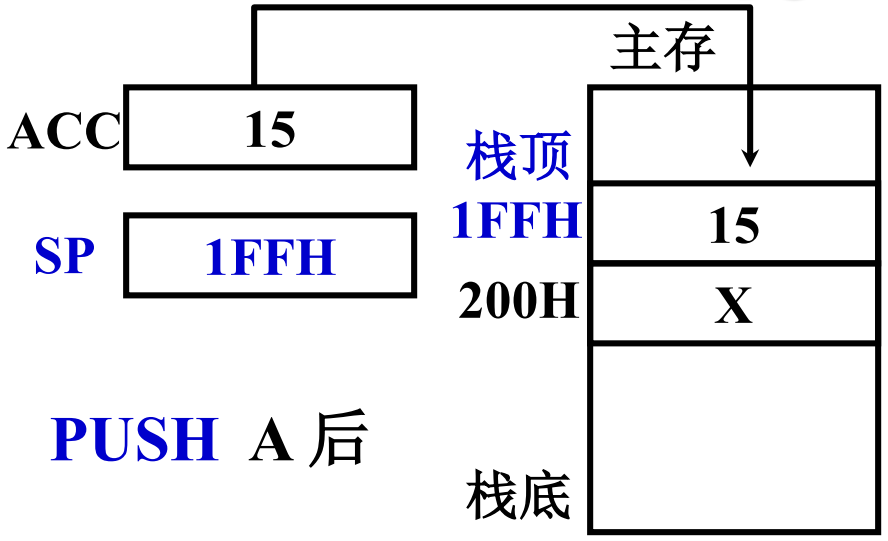


(2) 堆栈寻址举例

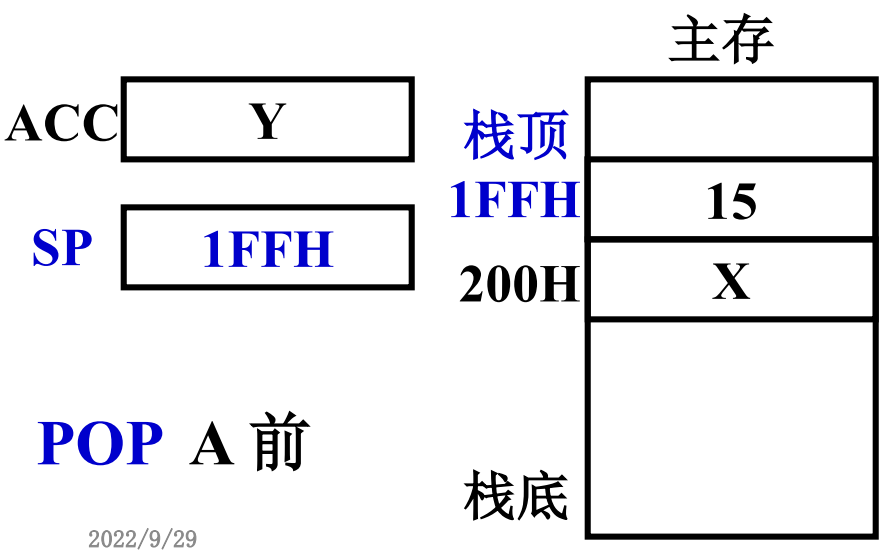
7.3



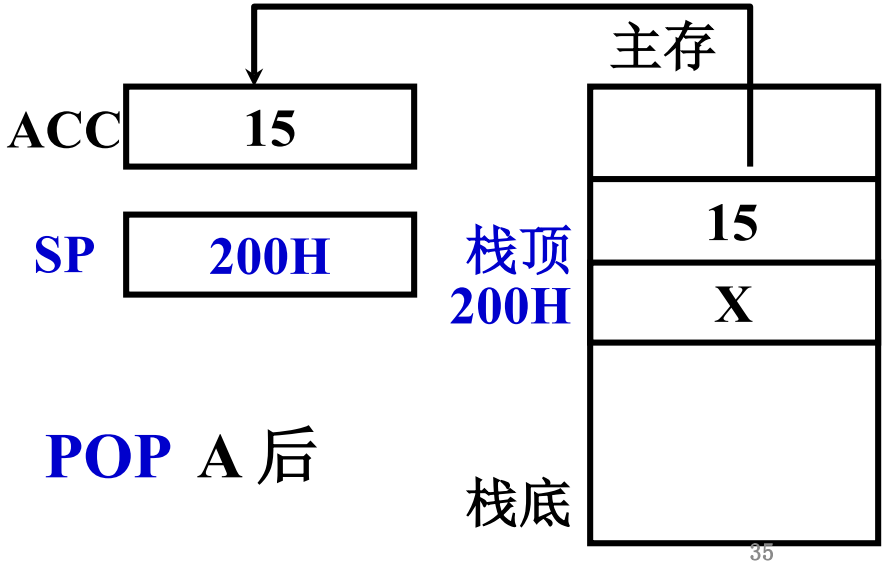
PUSH A 前



PUSH A 后



POP A 前



POP A 后

(3) SP 的修改与主存编址方法有关

① 按 字 编址

进栈　（SP）– 1 ⟶ SP

出栈　（SP）+ 1 ⟶ SP

② 按 字节 编址

存储字长 16 位　进栈　（SP）– 2 ⟶ SP

出栈　（SP）+ 2 ⟶ SP

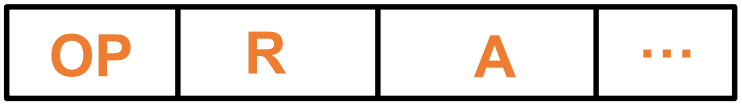
存储字长 32 位　进栈　（SP）– 4 ⟶ SP

出栈　（SP）+ 4 ⟶ SP

基本寻址方式的算法和优缺点

7.3

假设：A=地址字段值，R=寄存器编号，
EA=有效地址， (X)=X中的内容



方式	算法	主要优点	主要缺点
立即	操作数=A	指令执行速度快	操作数幅值有限
直接	EA=A	有效地址计算简单	地址范围有限
间接	EA=(A)	有效地址范围大	多次存储器访问
寄存器	操作数=(R)	指令执行快，指令短	地址范围有限
寄间接	EA=(R)	地址范围大	额外存储器访问
偏移	EA=A+(R)	灵活	复杂
堆栈	EA=栈顶	指令短	应用有限

偏移方式：将直接方式和寄存器间接方式结合起来。
有：相对 / 基址 / 变址三种

7.4 指令格式举例

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）

2. 其他因素

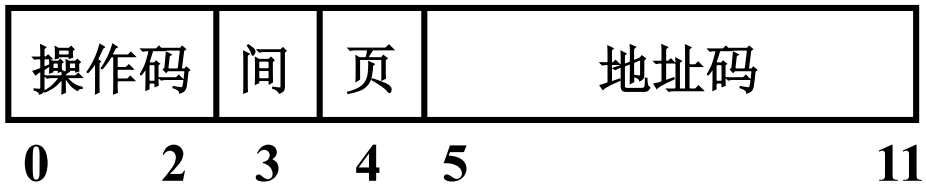
操作类型	包括指令个数及操作的难易程度
数据类型	确定哪些数据类型可参与操作
指令格式	指令字长是否固定
	操作码位数、是否采用扩展操作码技术，
	地址码位数、地址个数、寻址方式类型
寻址方式	指令寻址、操作数寻址
寄存器个数	寄存器的多少直接影响指令的执行时间

二、指令格式举例

7.4

1. PDP – 8　　指令字长固定 12 位

访存类指令



I/O 类指令

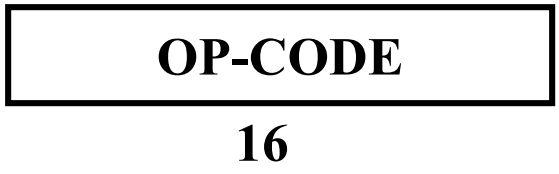
寄存器类指令

采用扩展操作码技术

2. PDP – 11

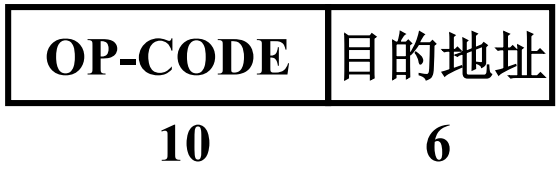
7.4

指令字长有 16 位、32 位、48 位三种

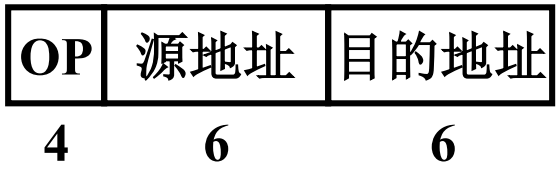


零地址 (16 位)

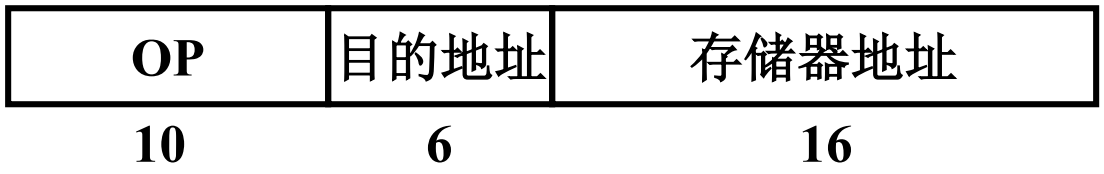
扩展操作码技术



一地址 (16 位)



二地址 R – R (16 位)



二地址 R – M (32 位)

二地址 M – M (48 位)

3. MIPS指令格式

7.4

32个MIPS寄存器

寄存器#	助记符	释义
0	$zero	固定值为0　硬件置位
1	$at	汇编器保留，临时变量
2~3	$v0~$v1	函数调用返回值
4~7	$a0~$a3	4个函数调用参数
8~15	$t0~$t7	暂存寄存器，调用者按需保存
16~23	$s0~$s7	save寄存器，被调用者按需保存
24~25	$t8~$t9	暂存寄存器，同上
26~27	$k0~$k1	操作系统保留，中断异常处理
28	$gp	全局指针 (Global Pointer)
29	$sp	堆栈指针 (Stack Pointer)
30	$fp	帧指针　(Frame Pointer)
31	$ra	函数返回地址 (Return Address)

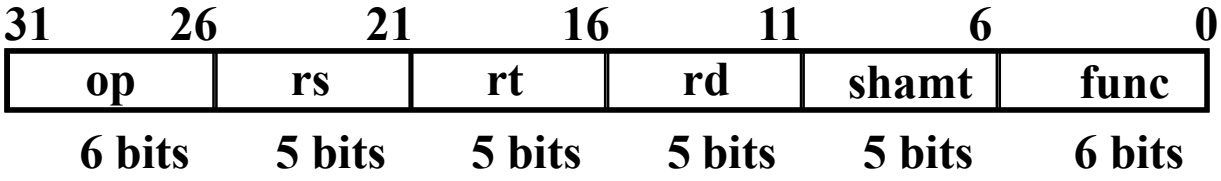
- **32个32位通用寄存器$0~$31**
- **32个32位单精度浮点寄存器f_0-f_{31}**
- **2个32位乘、商寄存器 H_i和L_0**
- **程序寄存器PC是单独的寄存器**
- **无程序状态寄存器**
- **RISC-V也有类似的32个寄存器设置**

3. MIPS指令格式

7.4

- 所有指令都是32位宽，须按字地址对齐
  字地址为4的倍数！
- 有三种指令格式
  - R-Type

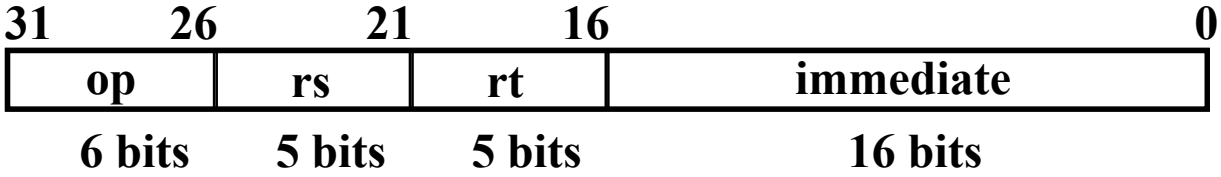
R-Type指令



两个操作数和结果都在寄存器的运算指令。如：sub rd, rs, rt

  - I-Type
    - 运算指令：一个寄存器、一个立即数。如：ori rt, rs, imm16
    - LOAD和STORE指令。如：lw rt, rs, imm16
    - 条件分支指令。如：beq rs, rt, imm16

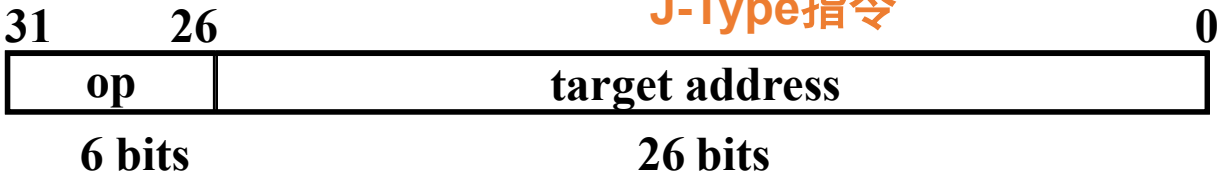
I-Type指令



  - J-Type

无条件跳转指令。如：j target

J-Type指令



MIPS指令字段含义

7.4

OP：操作码

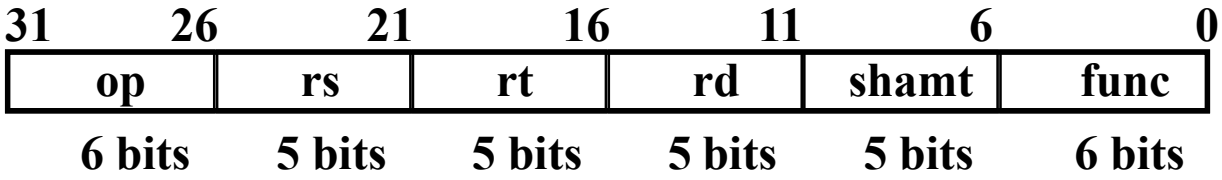
rs：第一个源操作数寄存器

rt：第二个源操作数寄存器

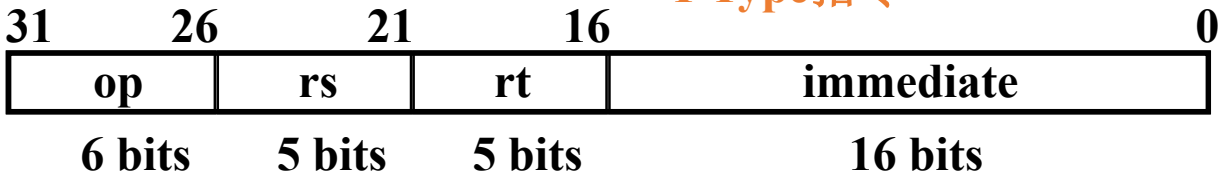
rd：结果寄存器

shamt：移位指令的位移量

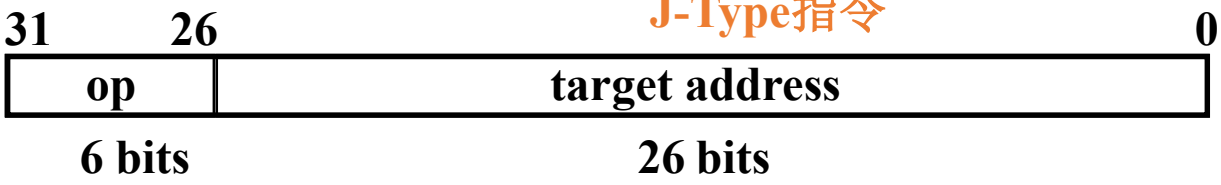
R-Type指令



I-Type指令



J-Type指令



func：R-Type指令的OP字段是特定的“000000”，具体操作由func字段给定。例如：func=“100000”时，表示“加法”运算。

操作码的不同编码定义不同的含义，操作码相同时，再由功能码定义不同的含义!

immediate：立即数或load/store指令和分支指令的偏移地址

target address：无条件转移地址的低26位。将PC高4位拼上26位直接地址，最后添2个“0”就是32位目标地址。为何最后两位要添“0”？

指令按字地址对齐，所以每条指令的地址都是4的倍数（最后两位为0）。

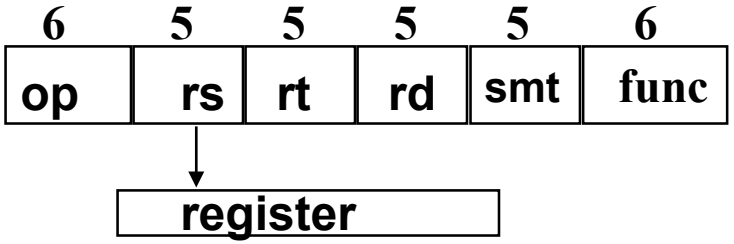
MIPS Addressing Modes（寻址方式） 7.4

有专门的寻址方式字段（Mod）吗？

没有！由指令格式来确定，而指令格式由op来确定！

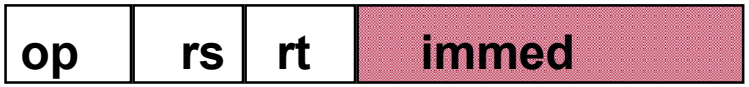
R-format: OP=000000H

Register



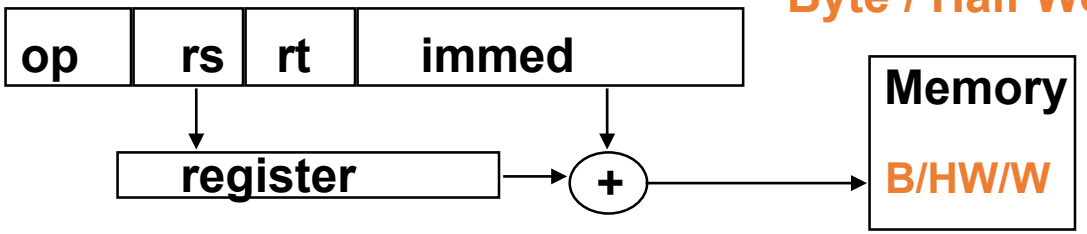
I-format:

Immediate

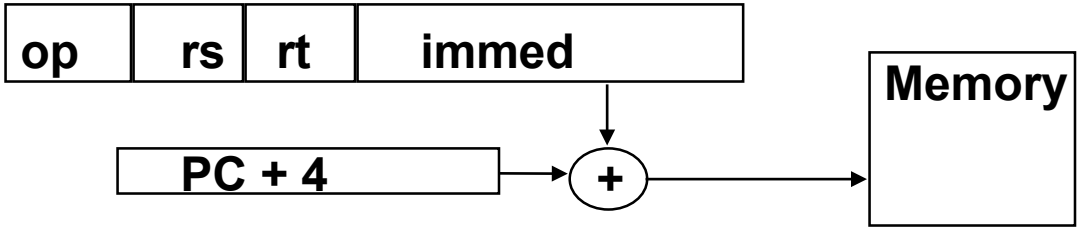


Base或index
基址或变址

Byte / Half Word / Word

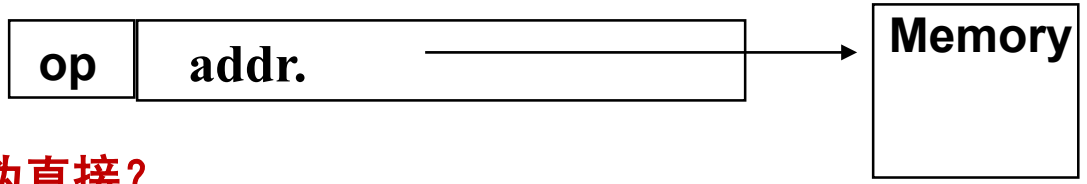


PC-relative
相对寻址



J-format: OP=000010H or 000011H

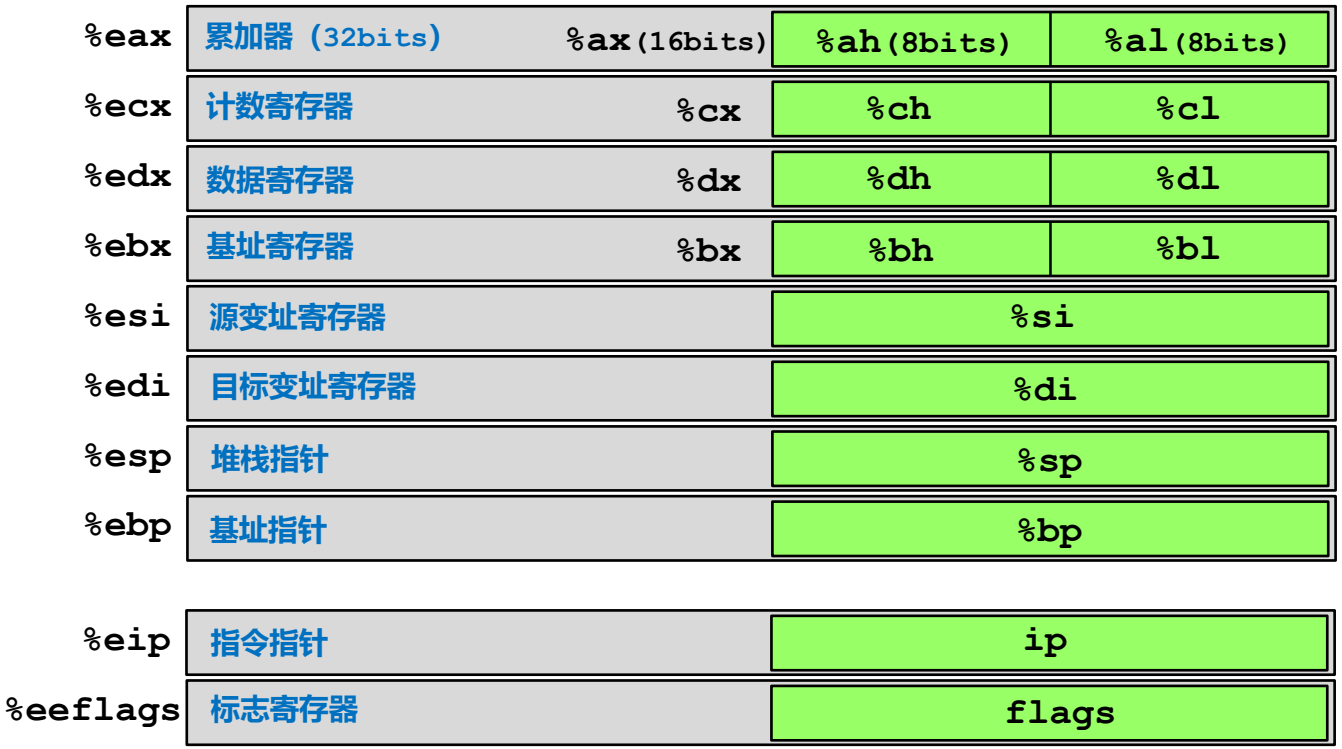
Pseudodirect
伪直接寻址



为什么称伪直接？

最终地址=$PC_{31\sim28}$||addr.||00　位数：4+26+2=32

4.X86指令系统举例-- IA-32的寄存器组织 7.4



- 8个通用寄存器
- 两个专用寄存器
- 6个段寄存器

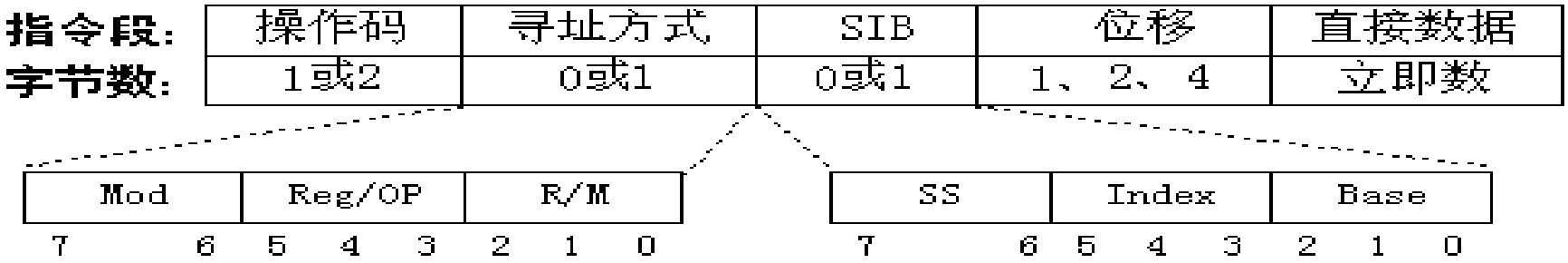
X86指令系统举例：Pentium指令格式

7.4

前缀：包括指令、段、操作数长度、地址长度四种类型

前缀类型：	指令前缀	段前缀	操作数长度	地址长度
字节数：	0或1	0或1	0或1	0或1

指令：含操作码、寻址方式、SIB、位移量和直接数据五部分，位移量和立即数都可是1/2/4B。SIB中基址B和变址I都可是8个GRS中任一个。SS给出比例因子。操作码：opcode; w：与机器模式（16 / 32位）一起确定寄存器位数（AL / AX / EAX）；d：操作方向；寻址方式： mod、r/m、 reg/op三个字段与w字段和机器模式一起确定操作数所在的寄存器编号或有效地址计算方式



变长指令字：1B~17B
变长操作码：4b / 5b / 6b / 7b / 8b /……
变长操作数：Byte / Word / DW / QW
变长寄存器：8位 / 16位 /32位

调用指令自动把返回地址压栈
专门的push/pop指令，自动修改栈指针
ALU指令在Flags中隐含生成条件码
ALU指令中的一个操作数可来自存储器
提供基址加比例索引寻址

Pentium处理器的寻址方式

7.4

操作数的来源：

立即数(立即寻址)：直接来自指令

寄存器(寄存器寻址)：来自32位 / 16位 / 8位通用寄存器

存储单元(其他寻址)：需进行地址转换

虚拟地址 => 线性地址LA (=> 内存地址)

分段　　分页

指令中的信息：

(1) 段寄存器SR（隐含或显式给出）

(2) 8/16/32位偏移量A （显式给出）

(2) 基址寄存器B （明显给出，任意通用寄存器皆可）

(3) 变址寄存器I （明显给出，除ESP外的任意通用寄存器皆可。）

- 有比例变址和非比例变址
- 比例变址时要乘以比例因子S (1:8位 / 2:16位 / 4:32位 / 8:64位)

指令“ MOV EAX, ES: [EBP+ESI*8+100]”的含义是什么？（作业）

保护模式下的寻址方式

7.4

寻址方式	说明
立即寻址	指令直接给出操作数
寄存器寻址	指定的寄存器R的内容为操作数
位移	LA=(SR)+A
基址寻址	LA=(SR)+(B)
基址加位移	LA=(SR)+(B)+A
比例变址加位移	LA=(SR)+(I)×S + A
基址加变址加位移	LA=(SR)+(B)+(I)+A
基址加比例变址加位移	LA=(SR)+(B)+(I)×S + A
相对寻址	LA=(PC)+A

存储器操作数

跳转目标指令地址

注：LA:线性地址　(X):X的内容　SR:段寄存器　PC:程序计数器　R:寄存器
A:指令中给定地址段的位移量　B:基址寄存器　I:变址寄存器　S:比例系数

- SR段寄存器（间接）确定操作数所在段的段基址
- 有效地址给出操作数在所在段的偏移地址
- 寻址过程涉及到“分段虚拟管理方式”，

MIPS X86 差异

7.4

#	X86	MIPS
1	变长（1-15bytes）	定长指令
2	指令数多 CISC	指令数少 RISC
3	8个通用寄存器	32个通用寄存器
4	寻址方式复杂	寻址方式简单
5	有标志寄存器	无标志寄存器
6	最多两地址指令	三地址指令
7	无限制	只有Load/store能访问存储器
8	有堆栈指令 push，pop	无堆栈指令（访存指令代替）
9	有I/O指令	无I/O指令(设备统一编址)

7.5 RISC 技术

一、RISC 的产生和发展

RISC（Reduced Instruction Set Computer）

CISC（Complex Instruction Set Computer）

80 — 20 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**

二、RISC 的主要特征

7.5

- 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
- **CPU** 中有多个 通用 寄存器
- 采用 流水技术 一个时钟周期 内完成一条指令
- 采用 组合逻辑 实现控制器
- 采用 优化 的 编译 程序

三、CISC 的主要特征

7.5

- 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 生成高效的目的代码

四、RISC和CISC 的比较

7.5

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