计算机组成原理

翁睿

哈尔滨工业大学

第 5 章　输入输出系统

5.1 概述

5.2 外部设备

5.3 I/O接口

5.4 程序查询方式

5.5 程序中断方式

5.6 DMA方式

3. DMA 方式与程序中断方式的比较 5.6

	中断方式	DMA 方式
(1) 数据传送	程序	硬件
(2) 响应时间	指令执行结束	存取周期结束
(3) 处理异常情况	能	不能
(4) 中断请求	传送数据	后处理
(5) 优先级	低	高

关于DMA 周期窃取过程的一点补充

- **宏观** 看起来好像通过DMA一次性连续传送了一批数据
- **微观** 每与主存间传送一个字，设备发一次 **DMA请求**

➢ **一个字的传输过程分析如下**

1. **设备准备数据：** 设备接收到CPU发送的启动命令后，开始准备数据。
2. **设备发送DMA请求：** 数据准备好后，设备通过DREQ控制线向DMA接口发出DMA请求。
3. **DMA接口申请总线：** DMA接口收到DMA请求后，立即将HRQ(HOLD)信号置“1”，向CPU申请总线控制权。

必须让出，因为DMA比CPU优先极高

4. **总线授权：** CPU在每个机器周期结束后，响应总线使用申请并让出总线控制权，并发出总线授权信号HLDA通知DMA接口可以使用总线。
5. **数据传输：** DMA接口成功的偷走了一个访存周期，一个字的数据在设备和主存之间按约定的方向传送。

其中，3、4步骤虽然经过了CPU，但CPU所做的事只相当于：

HLDA=(HOLD && (CPU当前不访存))，这个过程几乎不耗时。

第 7 章　指 令 系 统

7.1　机器指令

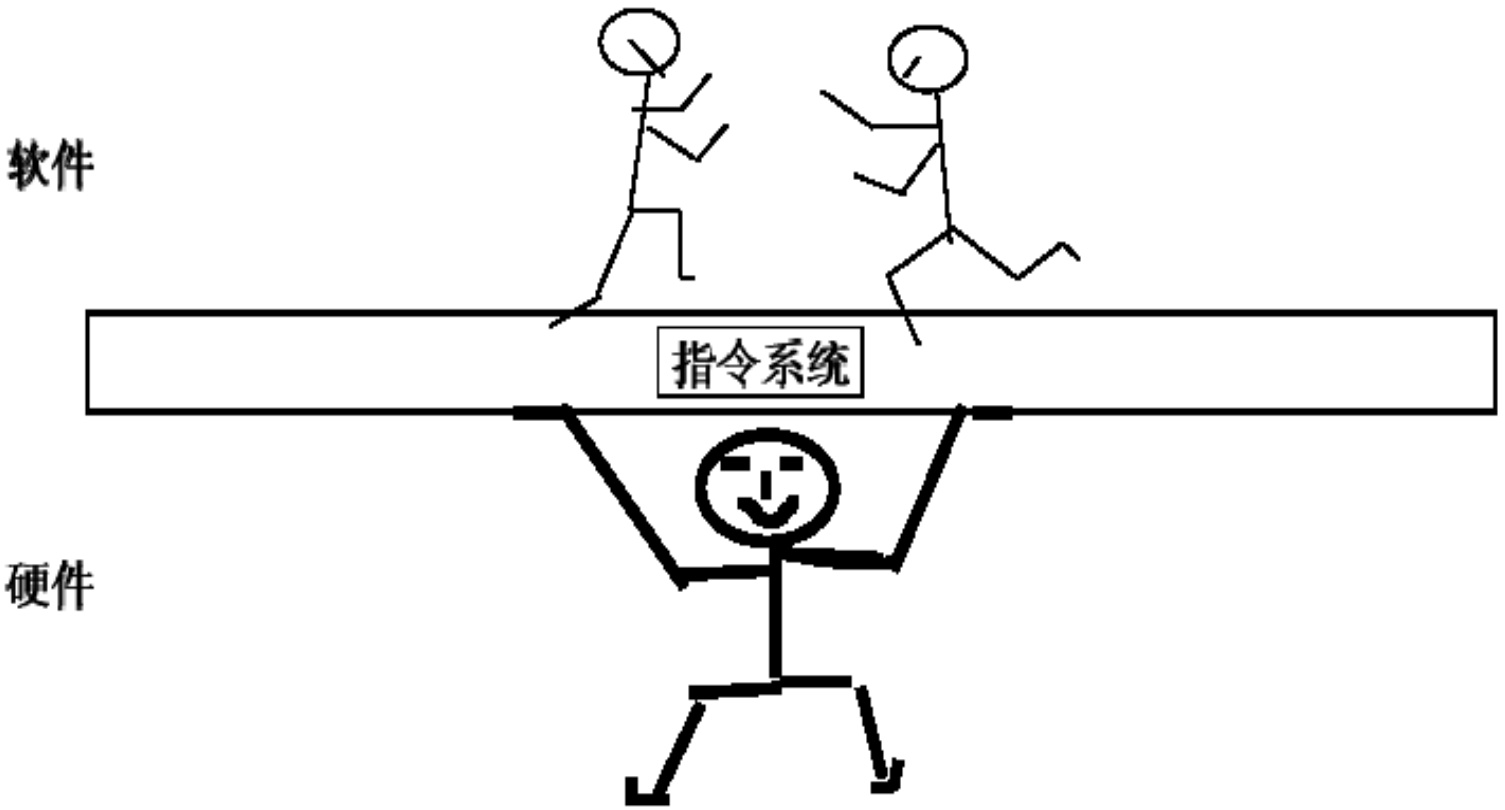
7.2　操作数类型和操作类型

7.3　寻址方式

7.4　指令格式举例

7.5　RISC 技术

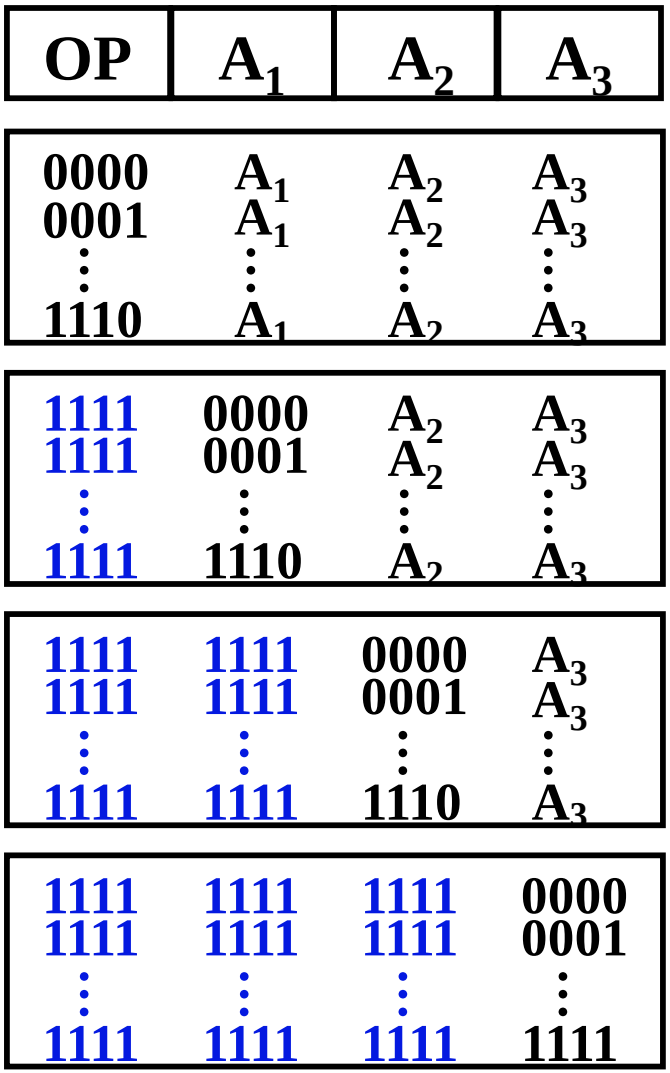
指令系统在计算机中的地位



7.1 机 器 指 令

一、指令的一般格式：操作码 + 地址码

1. 操作码　扩展操作码技术（变长操作码字段）：

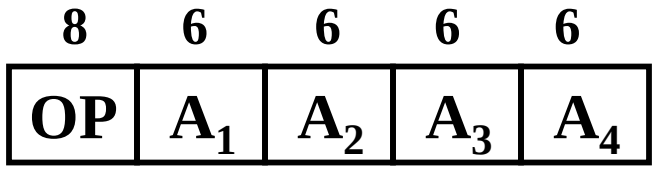


2. 地址码

设指令字长为 32 位
操作码固定为 8 位

7.1

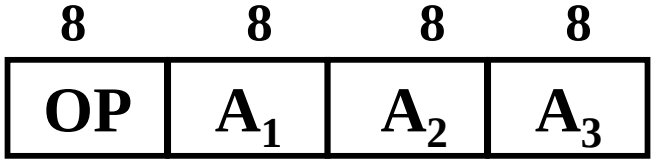
(1) 四地址



(A_1) OP (A_2) ⟶ A_3

4 次访存　寻址范围　$2^6 = 64$

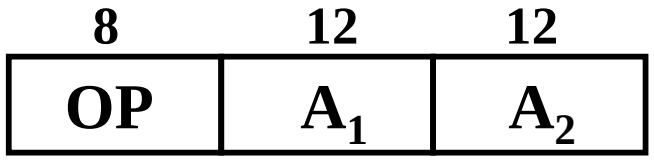
(2) 三地址



(A_1) OP (A_2) ⟶ A_3

4 次访存　寻址范围　$2^8 = 256$

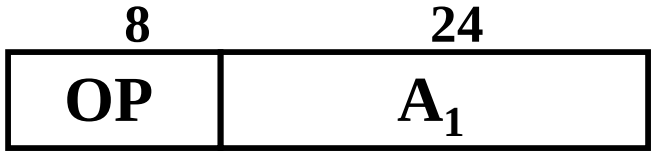
(3) 二地址



(A_1) OP (A_2) ⟶ $A_{1或2}$

4 次访存　寻址范围　$2^{12} = 4$ K

(4) 一地址



(ACC) OP (A_1) ⟶ ACC

2 次访存　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

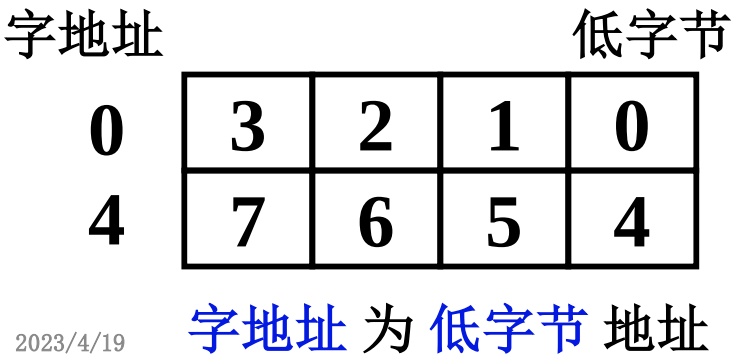
  一地址　OP　R_1
  - 可缩短指令字长
  - 指令执行阶段不访存

7.2 操作数类型和操作种类

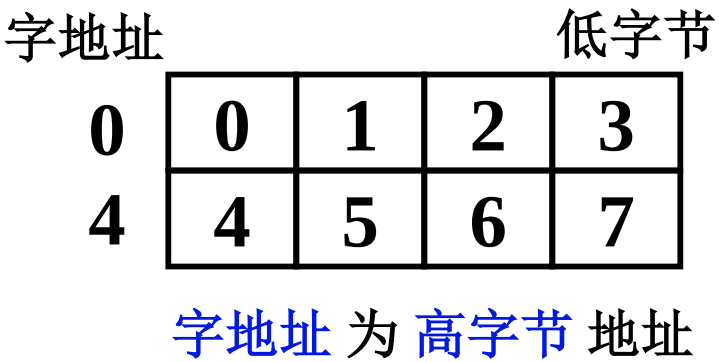
一、操作数类型

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



字地址 为 低字节 地址



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

寄存器 ➔ 寄存器　　寄存器 ➔ 主存　　置“1”，清“0”

主存 ➔ 主存　　主存 ➔ 寄存器　　压栈，弹栈

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

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

无条件转移　　条件转移

调用和返回　　陷阱/陷阱指令

5. 输入输出

输入指令

输出指令

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

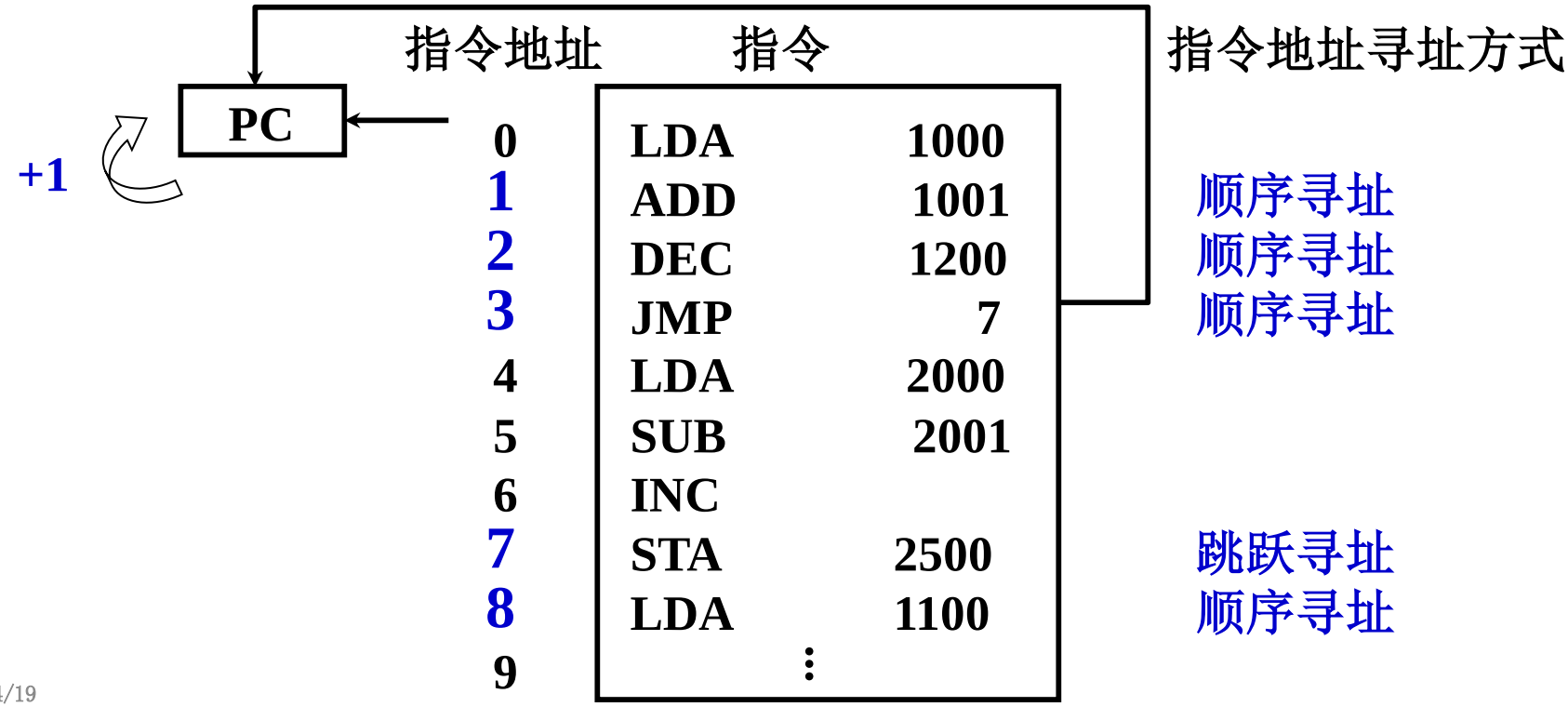
寻址方式
- 指令寻址
- 数据寻址

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

操作码	寻址特征	形式地址 A

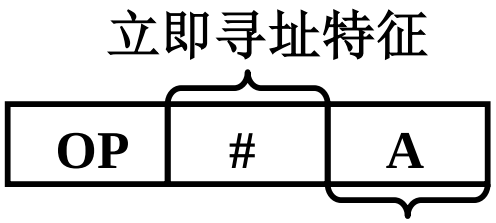
形式地址　　指令字中的地址

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

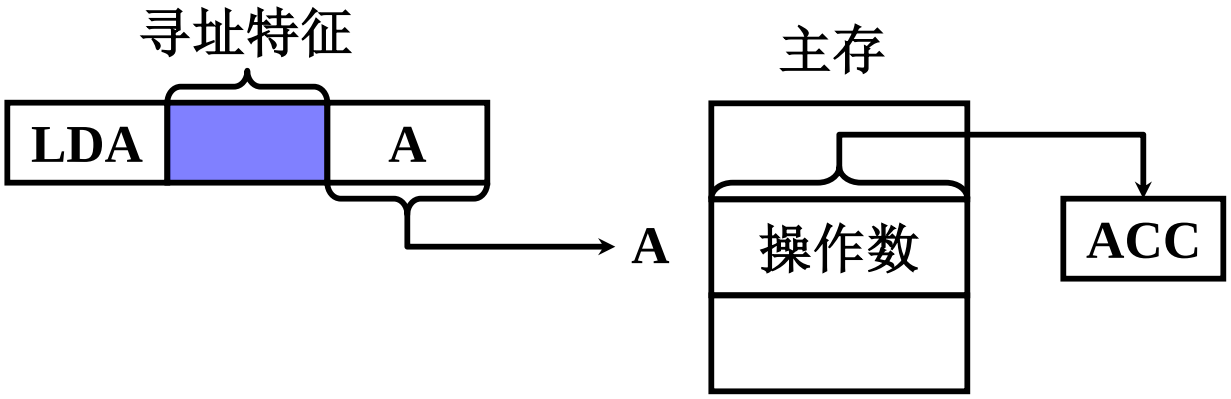


可正可负　补码

- 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

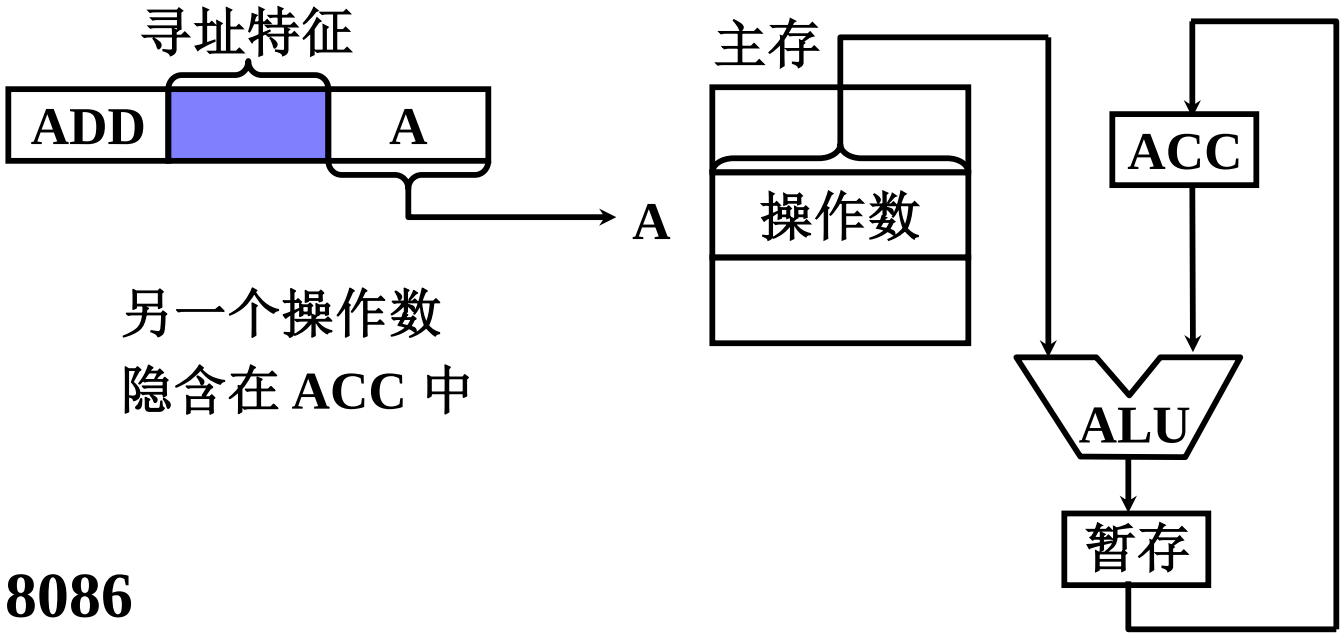
EA = A　　有效地址由形式地址直接给出



- 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

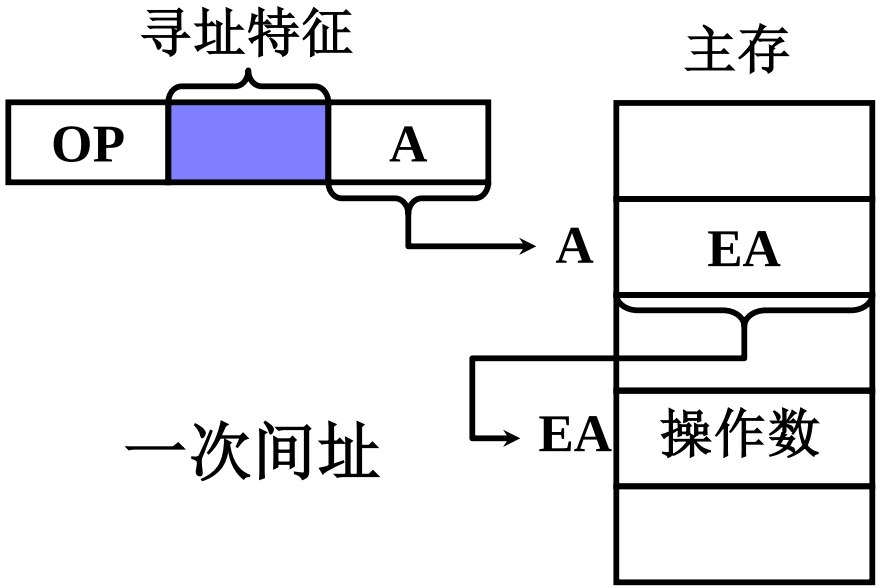
目的操作数的地址隐含在 DI 中

- 指令字中少了一个地址字段，可缩短指令字长

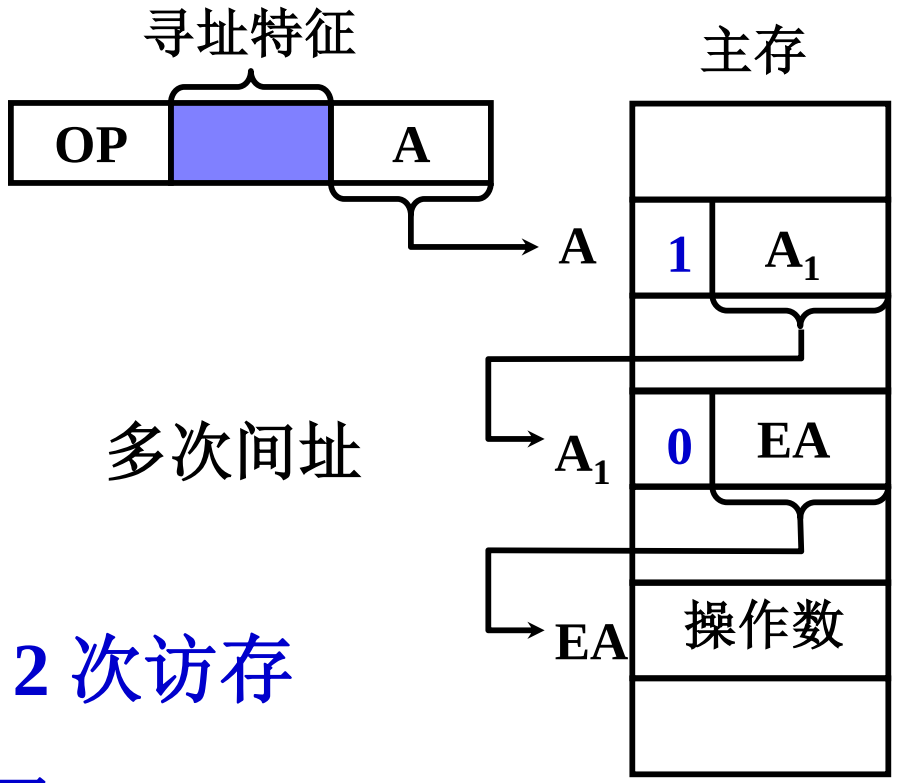
4. 间接寻址

7.3

EA =（A）　有效地址由形式地址间接提供



一次间址



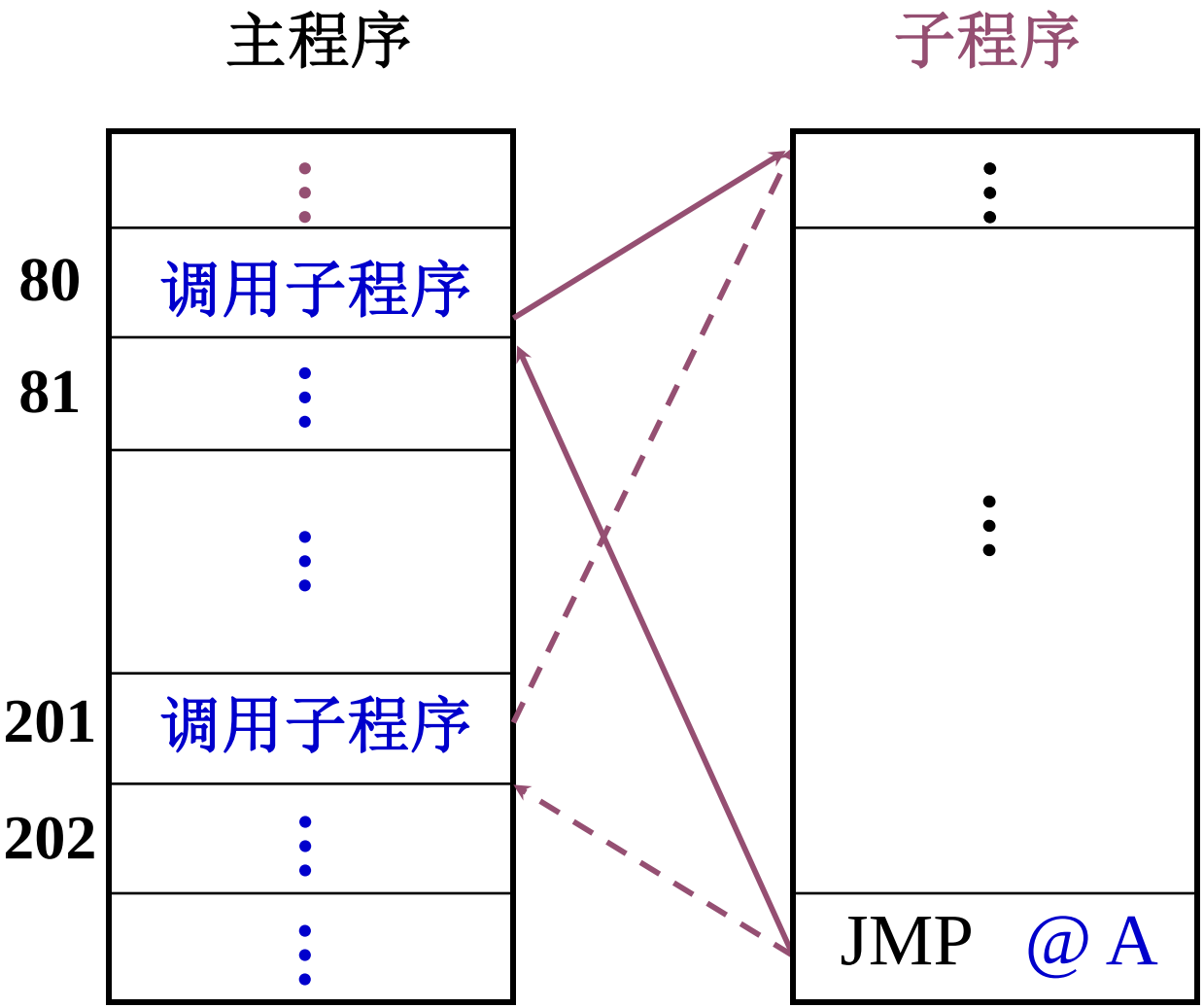
多次间址

- 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

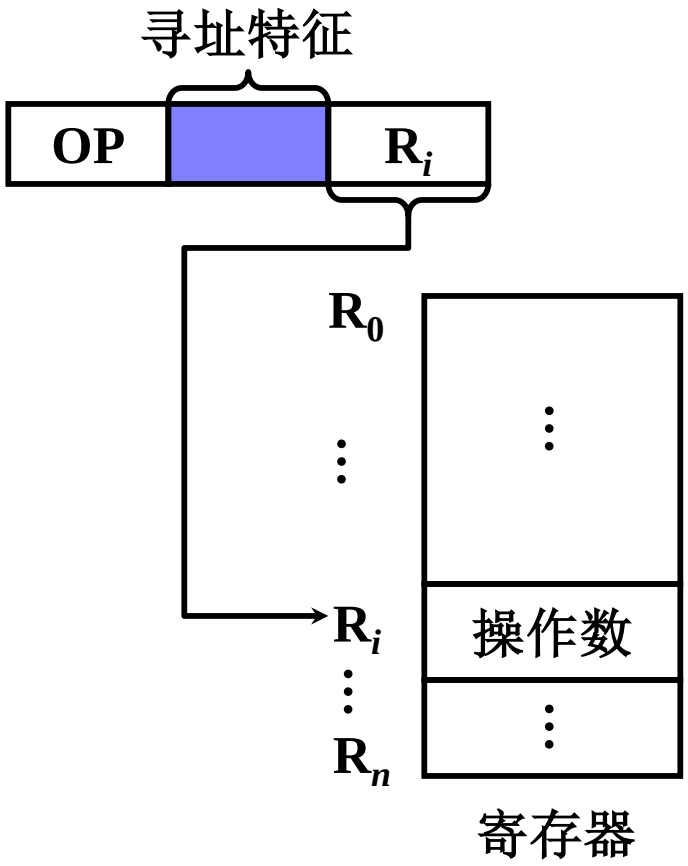


@ 间址特征

(A) = 202

5. 寄存器寻址

EA = R_i　　有效地址即为寄存器编号

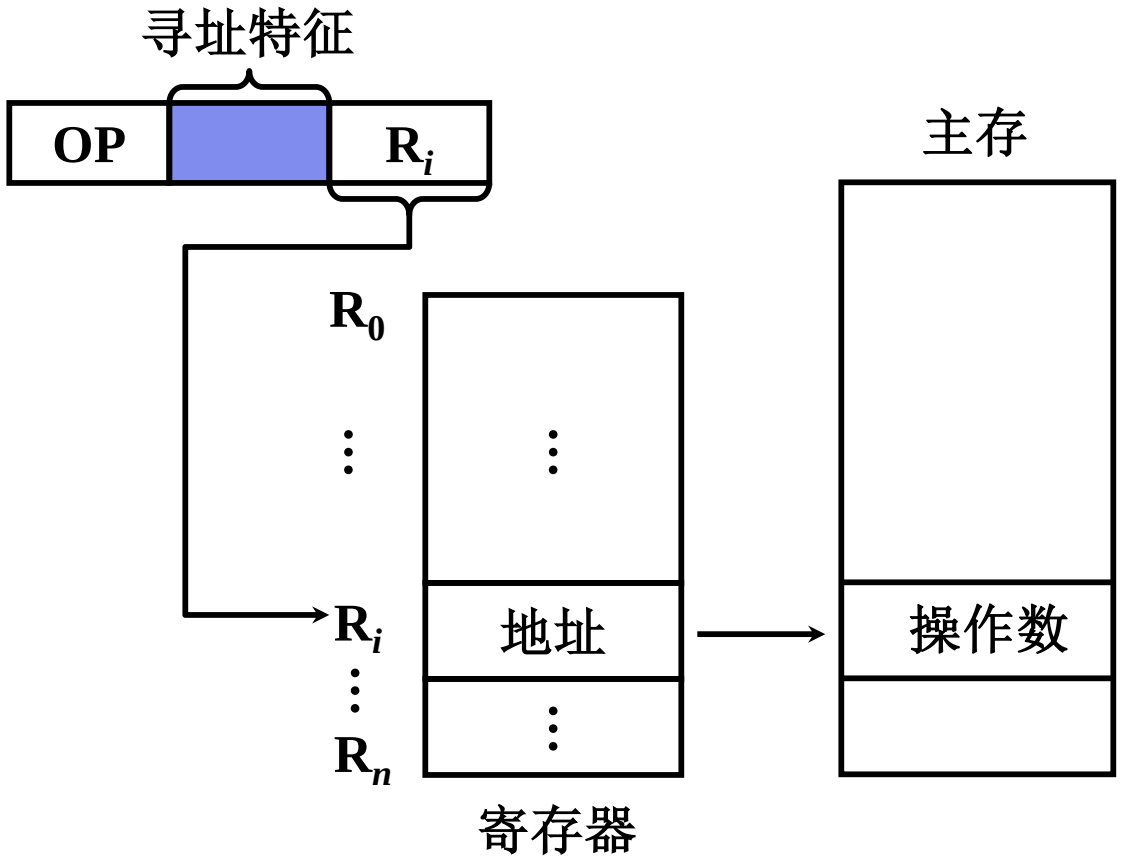


- 执行阶段不访存，只访问寄存器，执行速度快
- 寄存器个数有限，可缩短指令字长

6. 寄存器间接寻址

7.3

$EA = (R_i)$　　有效地址在寄存器中

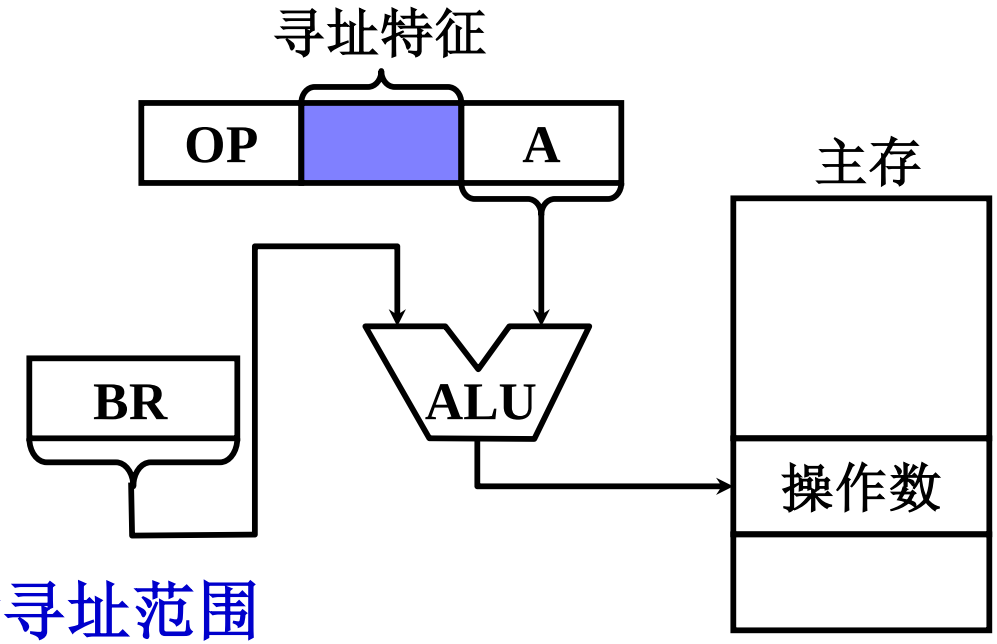


- 有效地址在寄存器中， 操作数在存储器中，执行阶段访存
- 便于编制循环程序

7. 基址寻址

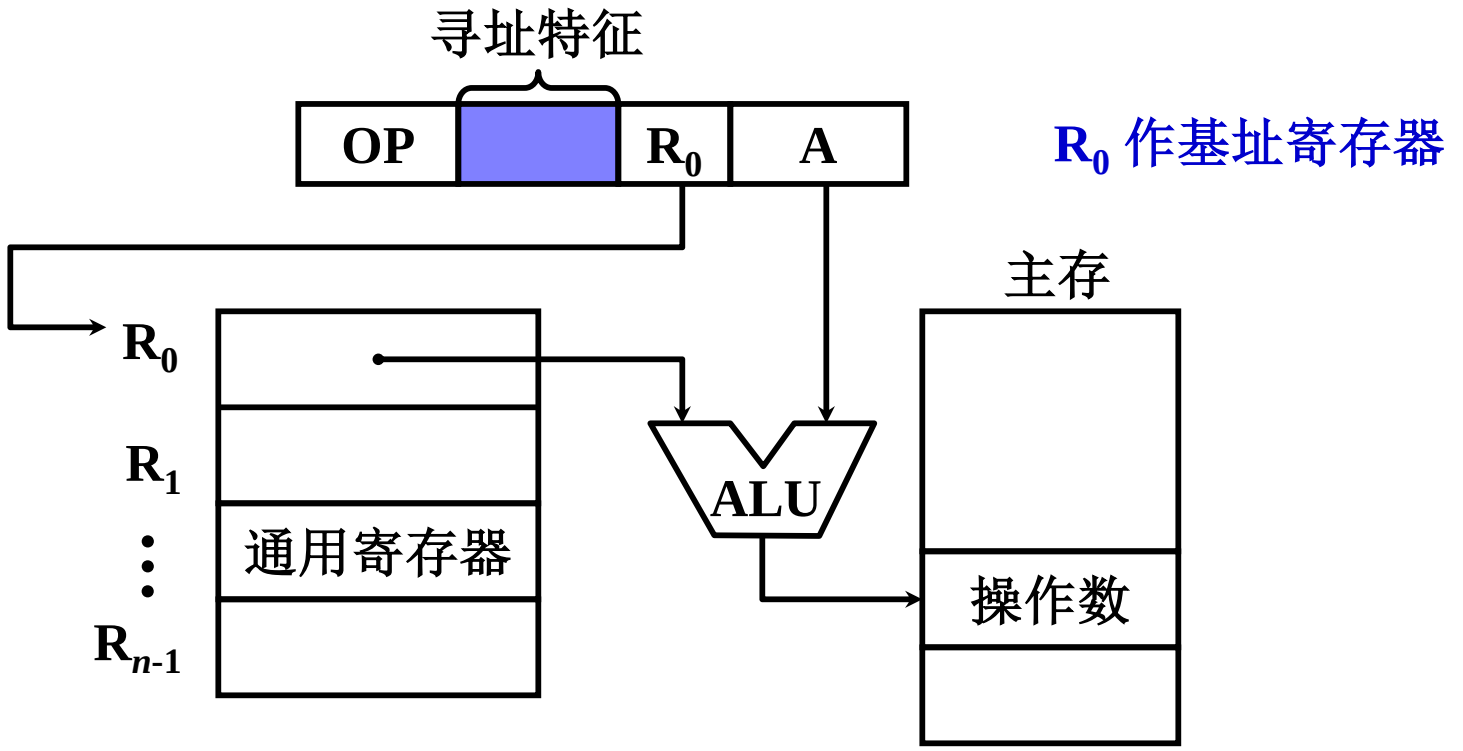
(1) 采用专用寄存器作基址寄存器

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



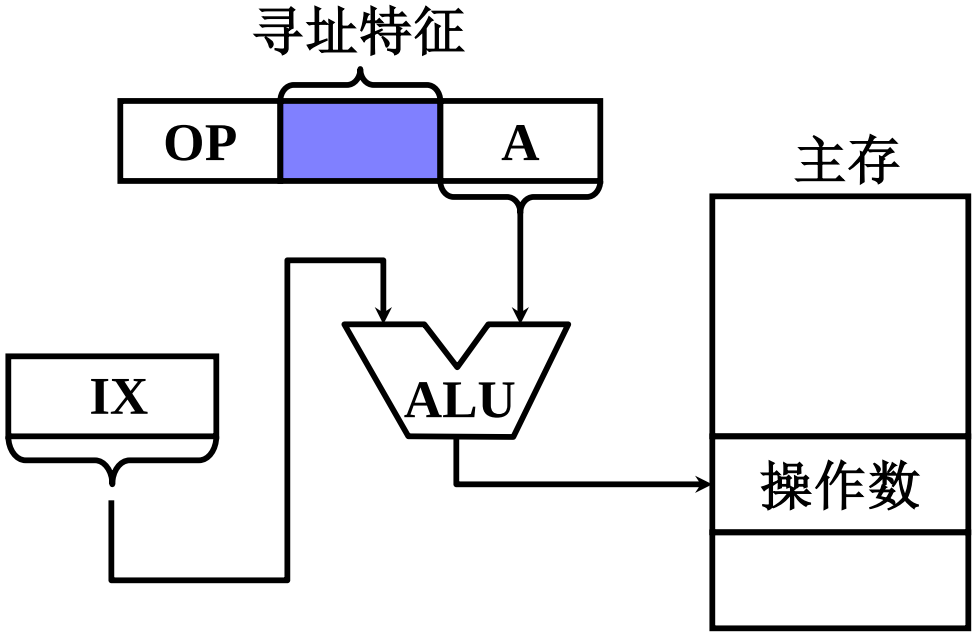
- 由用户指定哪个通用寄存器作为基址寄存器
- 基址寄存器的内容由操作系统确定
- 在程序的执行过程中 R_0 内容不变，形式地址 A 可变

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
```

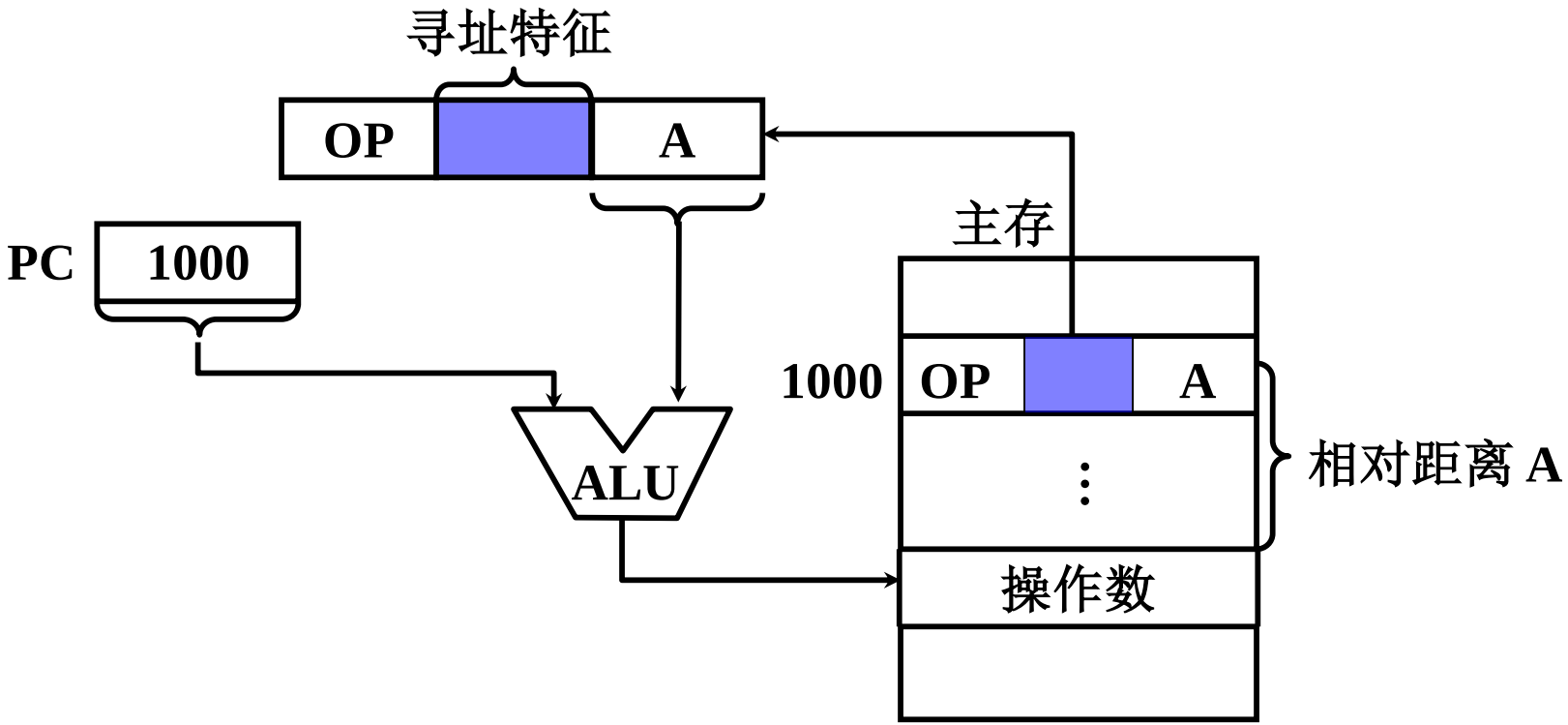
X 为变址寄存器

共 8 条指令

9. 相对寻址

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


```
          LDA    # 0
          LDX    # 0
M         ADD    X, D
M+1       INX
M+2       CPX    # N
M+3       BNE    M
          DIV    # N
          STA    ANS
```

∗ 相对寻址特征

M+3 BNE M → ∗ – 3

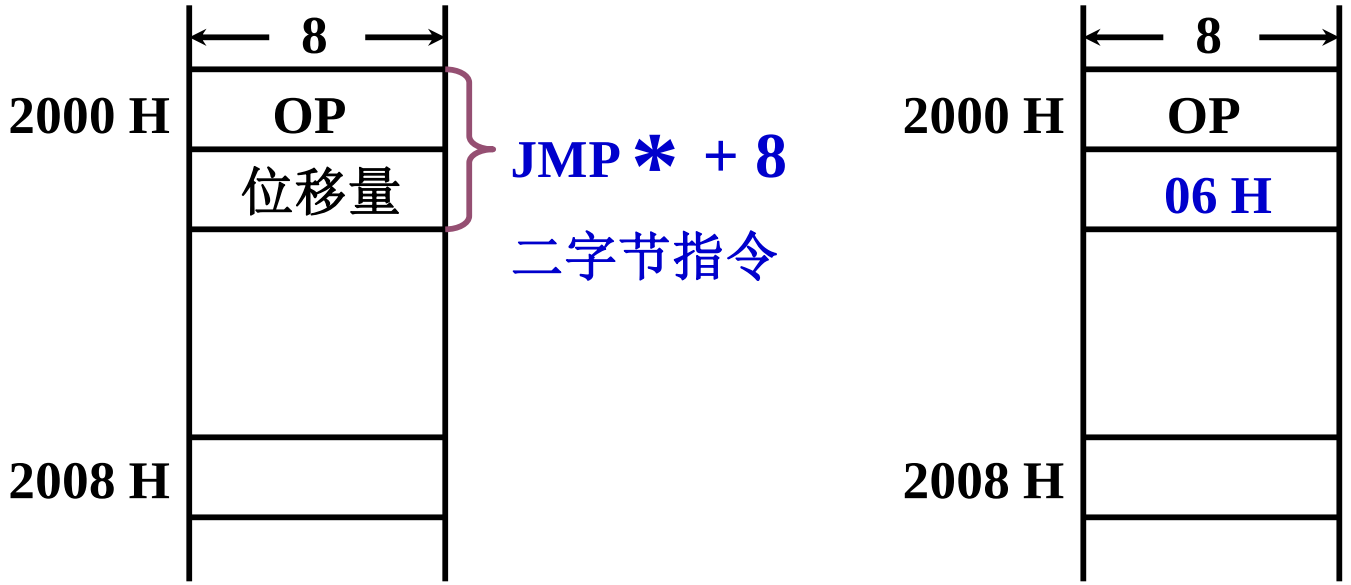
M 随程序所在存储空间的位置不同而不同

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**

10. 堆栈寻址

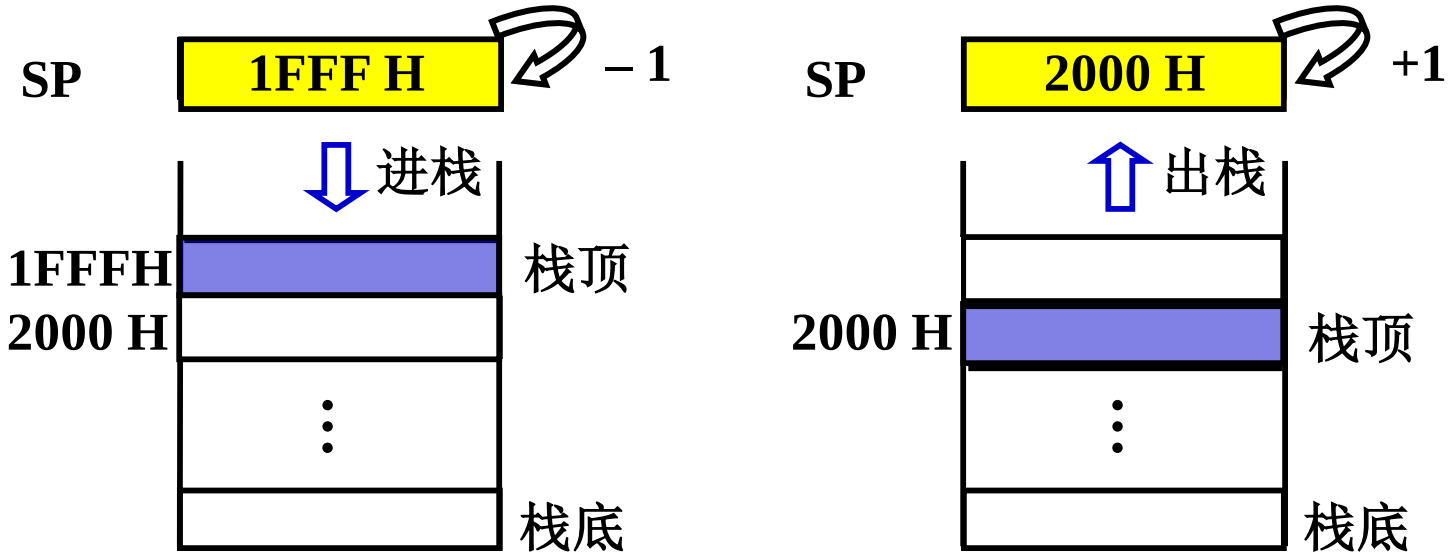
7.3

(1) 堆栈的特点

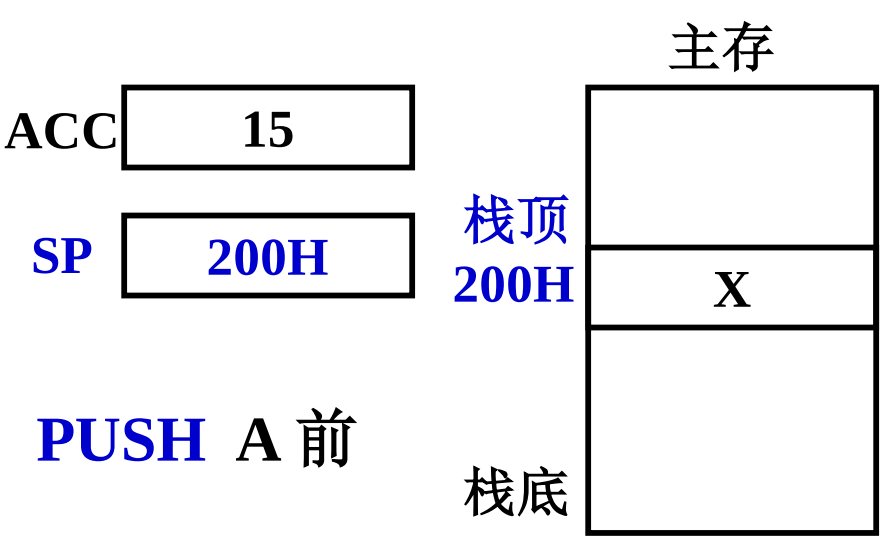
堆栈 { 硬堆栈　多个寄存器
　　 软堆栈　指定的存储空间

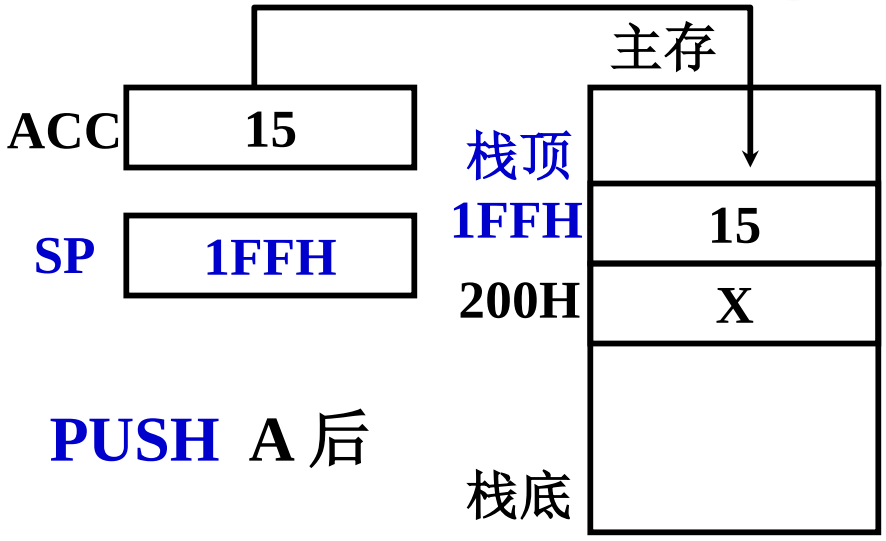
先进后出（一个入出口）　栈顶地址 由 SP 指出

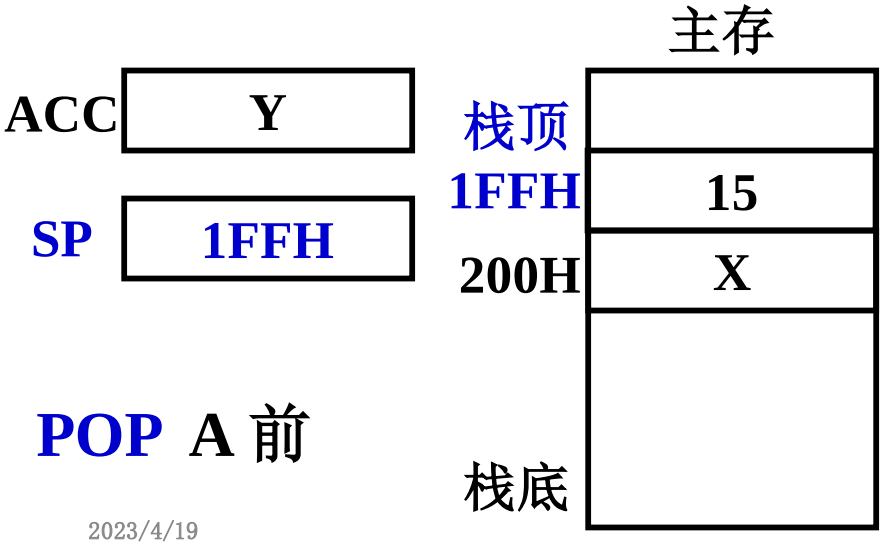
进栈 （SP）– 1 ⟶ SP　　出栈 （SP）+ 1 ⟶ SP

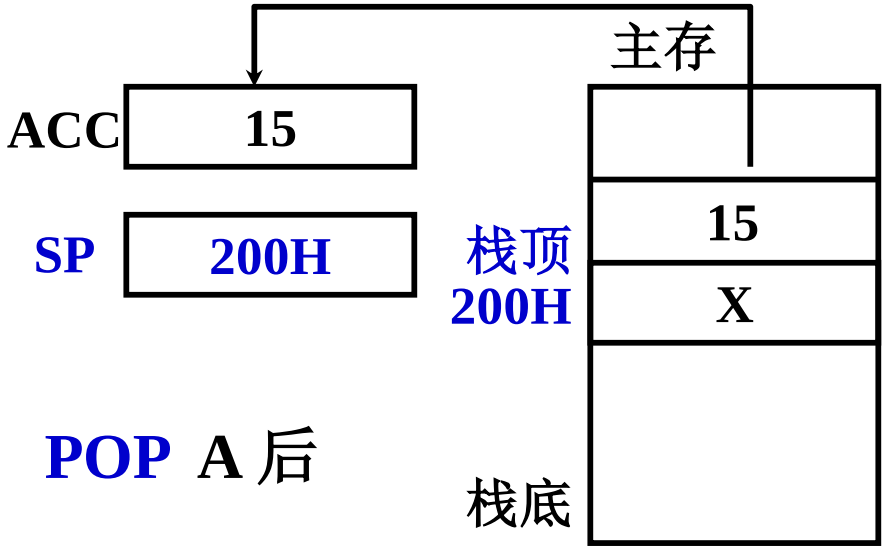


(2) 堆栈寻址举例


主存
ACC
15
SP
200H
栈顶
200H
X
PUSH A 前
栈底


主存
ACC
15
SP
1FFH
栈顶
1FFH
15
200H
X
PUSH A 后
栈底


主存
ACC
Y
SP
1FFH
栈顶
1FFH
15
200H
X
POP A 前
栈底


主存
ACC
15
SP
200H
15
栈顶
200H
X
POP A 后
栈底

(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

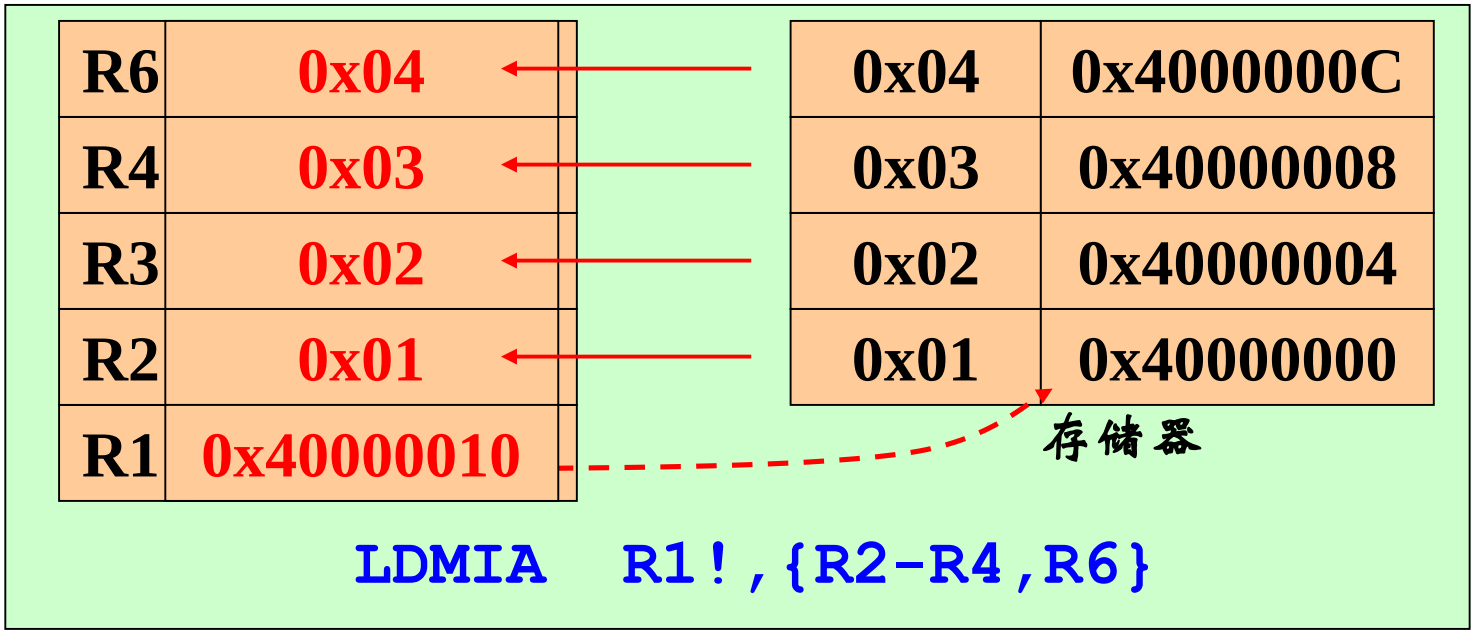
存储字长 32 位　进栈　（SP）– 4 ⟶ SP

出栈　（SP）+ 4 ⟶ SP

扩展知识：ARM处理器的多寄存器寻址

多寄存器寻址一次可传送几个寄存器值，允许一条指令传送16个寄存器的任何子集或所有寄存器。

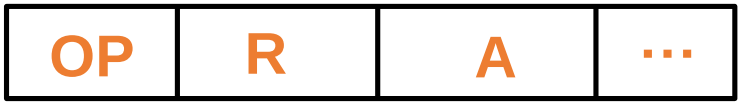
多寄存器寻址指令举例如下：



基本寻址方式及其优缺点总结

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	操作数范围有限
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

偏移寻址：将直接方式和寄存器间接方式结合起来。

具体包括：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. Intel 8086 指令格式（CISC）

(1) 指令字长　1 ~ 6 个字节 （ 整数个字节 ）

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

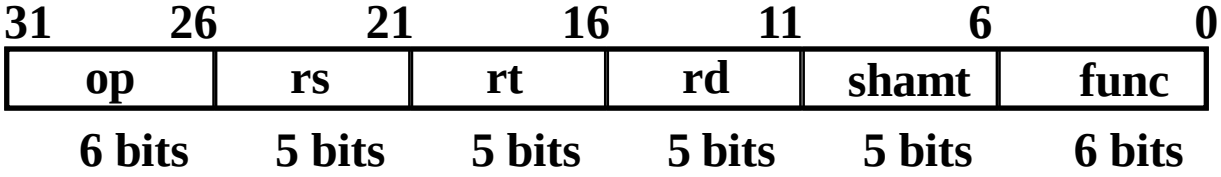
2. MIPS指令格式（RISC）

7.4

MIPS指令集有三种指令格式：**R型指令，I型指令，J型指令**

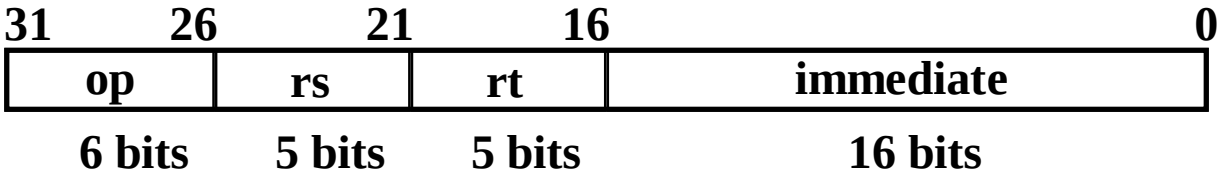
- **所有指令都是32位宽，须按字地址对齐，字地址为4的倍数！**

– R-Type



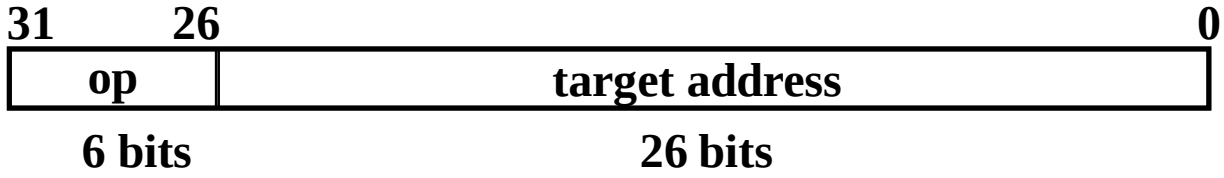
两个操作数和结果都在寄存器的运算指令。如：sub rd, rs, rt

– 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

– J-Type



跳转指令。如：j target

3. ARM® 指令系统

ARM处理器是基于精简指令集计算机(RISC)原理设计的，指令集和相关译码机制较为简单。

具有32位的ARM指令集和16位的Thumb指令集，ARM指令集效率高，但是代码密度低；而Thumb指令集具有较高的代码密度，却仍然保持ARM的大多数性能上的优势，它是ARM指令集的子集。

所有的ARM指令都是可以条件执行的，而Thumb指令仅有一条指令具备条件执行功能。ARM程序和Thumb程序可相互调用，相互之间的状态切换开销几乎为零。

•ARM指令集与Thumb指令集的关系

Thumb指令集具有灵活、小巧的特点

ARM指令集支持**ARM**核所有的特性，具有高效、快速的特点

•ARM指令集与Thumb指令集的比较

— Thumb代码所需的存储空间约为ARM代码的60%～70%

— Thumb代码使用的指令数比ARM代码多约30%～40%

— 若使用32位的存储器，ARM代码比Thumb代码快约40%

— 若使用16位的存储器，Thumb代码比ARM代码快约40%～50%

— 与ARM代码相比较，使用Thumb代码，存储器的功耗会降低约30%

•ARM指令集——指令格式

ARM指令的基本格式如下：

```
<opcode> {<cond>} {S}   <Rd> ,<Rn>{,<operand2>}
```

其中< >号内的项是必须的，{}号内的项是可选的。

各项的说明如下：

opcode：指令助记符；
cond：执行条件；
S：是否影响CPSR寄存器的值；
Rd：目标寄存器；
Rn：第1个操作数的寄存器；
operand2：第2个操作数；

二、指令格式举例：小结

7.4

1. Intel 8086 （CISC）

(1) 指令字长　1～6 个字节

(2) 地址格式　零地址　一地址　二地址

2. MIPS（RISC）

MIPS指令集有三种指令格式：**R型指令，I型指令，J型指令**

↑且只有　三地址　二地址　一地址

↑寄存器↑

3. ARM （RISC）

同时具有 32位的ARM指令集 和 16位的Thumb指令集

并能够无缝切换

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、RISC 的主要特征

- 只实现 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

- 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

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，大量采用 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

五、赠送小题一道

7.5

某RISC模型机支持39种操作，且操作码字段长度固定，支持直接、间接、相对、立即四种寻址方式。指令字长和存储字长均为16位，存储器按字编址，指令字采用单地址格式。

试问：

（1）该模型机的指令格式，解释指令字中各字段的作用；

（2）分别写出直接寻址、一次间址和多次间址的寻址范围；

（3）求该机立即数（补码）的表示范围。

某RISC模型机支持39种操作，且操作码字段长度固定，支持直接、间接、相对、立即四种寻址方式。指令字长和存储字长均为16位，存储器按字编址，指令字采用单地址格式。

试问：

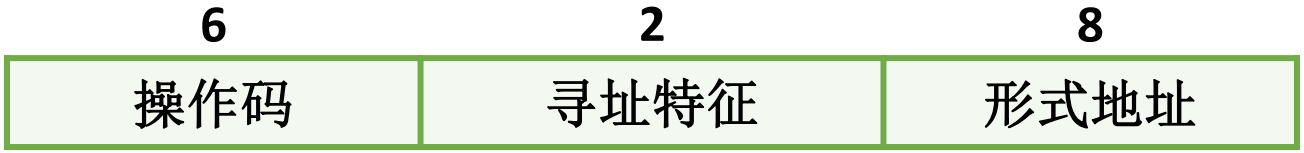
（1）该模型机的指令格式，解释指令字中各字段的作用；

（2）分别写出直接寻址、一次间址和多次间址的寻址范围；

（3）求该机立即数（补码）的表示范围。

答：

（1）该模型机的指令格式如下：



（2）直接寻址寻址范围为：$0 \sim (2^8-1)$ （或直接写2^8）

一次间址寻址范围为：$0 \sim (2^{16}-1)$ （或直接写2^{16}）

多次间址寻址范围为：$0 \sim (2^{15}-1)$ （或直接写2^{15}）

（3）该机立即数表示范围为：$-2^7 \sim +(2^7-1)$ （或 -128 ~ +127）