第四章 指令系统

- 4.1 指令系统的发展与性能要求
- 4.2 指令格式
- 4.3 操作数类型
- 4.4 指令和数据的寻址方式
- 4.5 典型指令
- 4.6 ARM汇编语言

4.1 指令系统的发展与性能要求

- 计算机的程序是由一系列的指令组成的。指令就是要计算机执行某种操作的命令。从计算机组成的层次结构来说，计算机的指令有微指令、机器指令和宏指令之分。
  - *计算机的程序*：是解决某一实际问题的指令序列；
  - *微指令*：包含若干微操作命令，属于硬件；
  - *机器指令*（指令）：每条指令可完成一个独立的算术运算或逻辑运算，属于硬件和软件的接口；
  - *宏指令*：由若干条机器指令组成的软件指令，属于软件。
  - **本章所讨论的指令，是机器指令。**
- 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。
  - 指令系统是表征一台计算机性能的重要因素，它的格式与功能不仅直接影响到机器的硬件结构，而且也直接影响到系统软件，影响到机器的适用范围。

4.1.1 指令系统的发展

- 发展过程：
  - ***50年代***：指令系统只有定点加减、逻辑运算、数据传送、转移等十几至几十条指令。
  - ***60年代后期***：增加了乘除运算、浮点运算、十进制运算、字符串处理等指令，指令数目达一二百条，寻址方式多样化。
  - 60年代后期开始出现系列计算机。
  - *系列计算机*：是指基本指令系统相同、基本体系结构相同的一系列计算机。其必要条件是同一系列的各机种有共同的指令集．而且新推出的机种指令系统一定包含所有旧机种的全部指令，即实现一个“向上兼容”。因此旧机种上运行的各种软件可以不加任何修改便可在新机种上运行，大大减少了软件开发费用。系列机解决了各机种的软件兼容问题

4.1.1 指令系统的发展

- 发展过程：
  - ***CISC*** ：复杂指令系统计算机，指令系统多达几百条指令。但是如此庞大的指令系统不但使计算机的研制周期变长，难以保证正确性，不易调试维护，而且由于采用了大量使用频率很低的复杂指令而造成硬件资源浪费。
  - ***RISC***：精简指令系统计算机，人们又提出了便于VLSI技术实现的精简指令系统计算机。
    - 指令集百分比20：80%规律：最常使用的简单指令占指令总数20%，在程序出现的频率是80%。
    - ***VLSI技术***：超大规模集成电路技术。

4.1.2 指令系统的性能要求

- 完善的指令系统要满足的4个要求：

1.*完备性* 完备性要求指令系统丰富、功能齐全、使用方便。

2.*有效性* 利用该指令系统所编写的程序能够高效率的运行。高效率主要表现在程序占据存储空间小、执行速度快。一般一个功能更强、更完善的指令系统，必定有更好的有效性。

3.*规整性* 规整性包括指令系统的对称性、匀齐性、指令格式和数据格式的一致性。

- 对称性指在指令系统中所有的寄存器和存储器单元都可同等对待，所有的指令都可使用各种寻址方式；
- 匀齐性是指一种操作性质的指令可以支持各种数据类型；
- 指令格式和数据格式的一致性是指指令长度和数据长度有一定的关系，以方便处理和存取。

4.*兼容性* 系列机各机种之间具有相同的基本结构和共同的基本指令集，因而指令系统是兼容的，即各机种上基本软件可以通用。但由于不同机种推出的时间不同，在结构和性能上有差异，做到所有软件都完全兼容是不可能的，只能做到“向上兼容”，即低档机上运行的软件可以在高档机上运行。

4.1.3 低级语言与硬件结构的关系

- 计算机语言分为高级语言和低级语言：P116 表4.1对比
  - *高级语言*：如C，其语句和用法与具体机器的指令系统无关。
    - 高级语言与计算机的硬件结构及指令系统无关，在编写程序方面比汇编语言优越。但是高级语言程序“看不见”机器的硬件结构，不能用于编写直接访问机器硬件资源的系统软件或设备控制软件。
  - *低级语言*：分机器语言（二进制语言）和汇编语言（符号语言），是面向机器的语言，和具体机器的指令系统密切相关。
    - 机器语言：用 指令代码/编码 编写程序，计算机能够直接识别和执行的唯一语言。
    - 符号语言/汇编语言：用 指令助记符 编写程序，依赖于计算机的硬件结构和指令系统。不同的机器有不同的指令，不同汇编语言编写的程序不能在其他类型的机器上运行。
    - 符号语言程序编译：人们通常采用符号语言或高级语言编写程序，但计算机却不能识别。必须借助汇编程序或编译程序，把符号语言或高级语言翻译成二进制机器语言。

4.2 指令格式

- 每指令中包含以下信息：
  - 做什么操作
  - 如果需要操作数，从哪里取
  - 结果送哪里
  - 下一条指令从哪里取
- 指令字：简称指令，即表示一条指令的机器字/0,1编码。
- 指令格式：是指令字用二进制代码表示的结构形式，由操作码字段和地址码字段组成。
  - 操作码字段：表征指令的操作特性与功能；
  - 地址码字段：通常指定参与操作的操作数的地址。

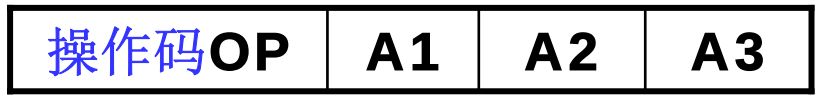
操作码字段OP	地址码字段A

4.2.1 操作码

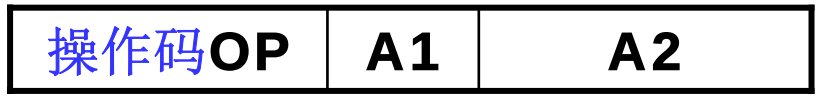
- 指令的操作码**OP**：表示该指令应进行什么性质的操作
  - 如：加法、减法、乘法、除法、取数、存数等等。
  - 不同的指令用操作码字段的不同编码来表示，每一种编码代表一种指令。
- 操作码**OP**的位数：组成操作码字段的位数一般取决于计算机指令系统的规模。
  - 若操作码字段的位数固定为n位，则指令系统最多可表示2^n条指令
  - 有固定位数和可变位数操作码；
  - 例：IBM 370机，该机字长32位，16个通用寄存器R0～R15，共有183条指令；指令的长度可以分为16位、32位和48位等几种，所有指令的操作码都是8位固定长度。
  - 固定长度编码的主要缺点是：信息的冗余极大，使程序的总长度增加。

4.2.2 地址码

- 地址码字段通常指定参与操作的操作数的地址或操作数本身。
- 根据一条指令中有几个操作数地址，可将该指令称为几操作数指令或几地址指令。
- 一般指令的操作数有被操作数、操作数及操作结果这三种数：
  - 三地址指令



  - 二地址指令



  - 单地址指令

  - 零地址指令

4.2.2 地址码

- 根据地址码提供的操作数的保存位置，有4种形式的操作数：
  - 立即数操作数：
    - 指令中地址码位置是操作数本身，操作数保存在指令编码中。
    - **mov** ax,**1234H**
  - 寄存器操作数：
    - 指令中地址码位置是**cpu内部寄存器的名字**，操作数保存在寄存器中。
    - **mov ax**,1234H
  - 内存操作数：
    - 指令中地址码位置是内存地址，操作数保存在地址指定的内存单元中。
    - **mov** ax,**[1234H]**
  - 端口操作数：
    - 指令中地址码位置是**I/O端口编号**，操作数保存在I/O端口中。
    - **in** ax,**12H**

4.2.2 地址码

- 三地址指令：

操作码OP	A1	A2	A3

  - 指令功能：
    - (A1) op (A2) → A3
    - (PC)+1→PC，下一条指令的保存位置
    - PC：程序计数器，当前指令的保存位置。
    - 在80X86cpu中，IP寄存器实现PC的功能。
  - 特点：
    - 指令长度仍比较长，所以只在字长较长的大、中型机中使用，而小型、微型机中很少使用。
- 二地址指令：

操作码OP	A1	A2

  - 指令功能：
    - (A1) op (A2) → A1
    - (PC)+1→PC，下一条指令的保存位置
  - 特点：
    - 二地址指令在计算机中得到了广泛的应用，但是在使用时有一点必须注意：指令执行之后，A1中原存的内容已经被新的运算结果替换了。

4.2.2 地址码

- 一地址指令：

操作码OP	A1

  - 指令功能：
    - (AC) op (A1) → A1
      - AC是cpu中的累加器AL / AX / EAX，隐含操作数
    - (PC) + 1→PC，下一条指令的保存位置
  - 一地址指令，如“+1”、“-1”、“求反”
- 零地址指令：

操作码OP	

  - 指令功能：
    - (PC) + 1→PC，下一条指令的保存位置
  - 零地址指令，“停机”、“空操作”、“清除”等控制类指令。

4.2.3 指令字长度

- 指令字长度：一个指令字中包含二进制代码的位数。有等长和变长两种。
  - *等长指令字结构*：各种指令字长度是相等的。这种指令字结构简单，且指令字长度是不变的。
  - *变长指令字结构*：各种指令字长度随指令功能而异。结构灵活，能充分利用指令长度，但指令的控制较复杂。
- 机器字长：计算机能直接处理的二进制数据的位数，它决定了计算机的运算精度。操作数可以包含的二进制最大位数。
  - 单字长指令：指令字长度=**1个**机器字长度
  - 半字长指令：指令字长度=半个机器字长度
  - 双字长指令：指令字长度=**2个**机器字长度
  - 单字长、半字长指令：长度有限，功能简单，能力有限
  - 多字长指令的优缺点
    - 优点提供足够的地址位来解决访问内存任何单元的寻址问题；
    - 缺点必须两次或多次访问内存以取出一整条指令，降低了CPU的运算速度，又占用了更多的存储空间。

4.2.4 指令助记符

- 指令助记符：机器语言程序的操作码使用0/1的不同组合表示，难记忆，难理解；在汇编语言程序中，为了便于书写和阅读程序，每条指令通常用3个或4个英文缩写字母来表示。
- 注意：
  - 不同的计算机系统，助记符的规定不一样
  - 汇编：使用汇编工具，将助记符形式的操作码翻译成二进制形式的操作码；
  - 例：“将寄存器BX的内容送到AX中”
    - 机器码是：000011 01110 11000
    - 操作码 2个寄存器地址
    - 汇编指令是：MOV AX,BX

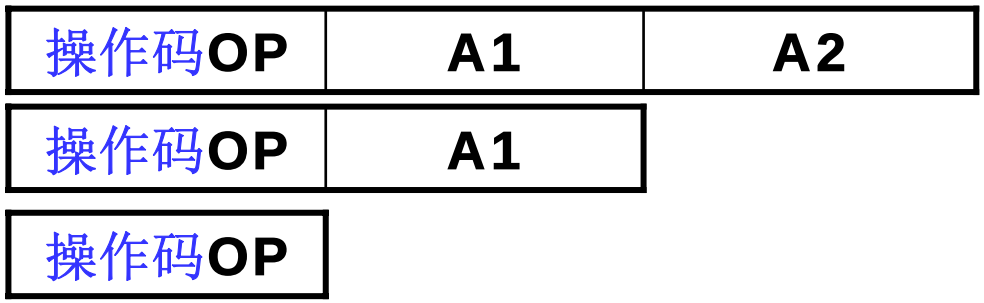
4.2.4 指令助记符

- 常见指令助记符：

典型指令	指令助记符	二进制操作码
加　法	ADD	001
减　法	SUB	010
传　送	MOV	011
跳　转	JMP	100
转　子	JSR	101
存　储	STR	110
读　数	LDA	111

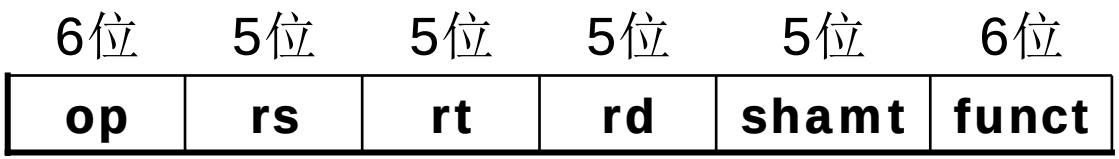
4.2.5 指令格式举例

- 八位微型计算机的指令格式：如8088，字长8位，内存字节寻址，指令结构可变，包括：
  - 单字长指令：指令字长度=1个字，只有操作码，没操作数；
  - 双字长指令：指令字长度=2个字，有操作码，有1个操作数；
  - 三字长指令：指令字长度=3个字，有操作码，有2个操作数；
  - 操作码长度固定使用0/1的不同组合表示；



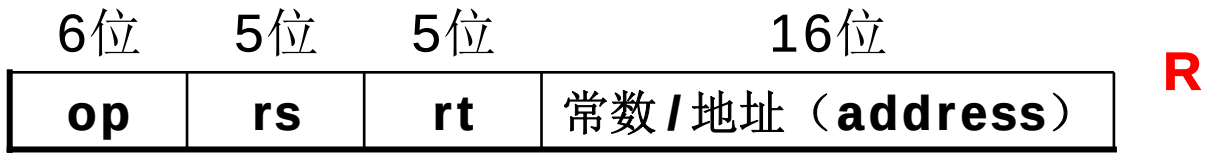
4.2.5 指令格式举例

- **MIPS R4000**指令格式：RISC计算机系统，字长32位，内存字节寻址，指令结构固定，1个字长，有R型和I型两类指令：
  - **R型**：运算指令，所有运算数据要保存在通用寄存器中，通用寄存器共32个。
    - op字段：6位，操作码，区分指令类型(R/I)；
    - rs字段：5位，提供第1个操作数的寄存器；
    - rt字段：5位，提供第2个操作数的寄存器；
    - rd字段：5位，保存运算结果的寄存器；
    - shamt字段：5位，移位取值，实现跳转指令；
    - funct字段：6位，函数码，指定要实现的特定运算功能；



4.2.5 指令格式举例

- **MIPS R4000**指令格式：
  - **I型**：访存指令，提供取字和存字。
    - **op**字段/**rs**字段/**rt**字段：含义同R型；
    - 常数/address字段：16位，提供常数/要访问内存单元的地址；
    - op字段：R型一般=0，I型可以是8,35,43；P120，表4.3



R

4.2.5 指令格式举例

- **ARM**的指令格式：P120
  - opcode：操作码，要实现基本操作。
  - Rn：源操作数寄存器。
  - Rd：目的操作数寄存器。
  - operand2：第2个源操作数。
  - I：立即数。
  - S：状态控制。
  - F：指令类型控制。

4位	2位	1位	4位	1位	4位	4位	12位
cond	F	I	opcode	S	Rn	Rd	operand2

4.2.5 指令格式举例

- **Pentium**的指令格式：P120
  - 变长指令字：1B~12B；
  - 指令字组成：操作码+Mod-R/M+SIB+位移量+立即数，共5个字段，后4个可选，不需要置为0；

1/2B		0/1B			0/1B		0/1/2/4	0/1/2/4
op	**Mod**	**Reg/op**	**R/M**	比例**S**	变址**I**	基址**B**	位移量	立即数
	2位	**3**位	**3**位	**2**位	**3**位	**3**位		

4.2.5 指令格式举例

- 例：分析指令格式特点，字长16位。

15~9		7~4	3~0
op	------	源寄存器	目标寄存器

  - 单字长2地址指令；
  - 操作码op=7位：$2^7=128$条指令；
  - 2地址提供寄存器操作数， RR型指令，4位，16个寄存器选1个。

15~10		7~4	3~0
op	------	源寄存器	变址寄存器
位移量16位			

  - 双字长2地址指令；
  - 操作码op=6位：$2^6=64$条指令；
  - 2地址提供寄存器和内存操作数， RS型指令
  - 寄存器，源寄存器提供，4位，16选1。
  - 内存数据：变址寄存器+位移量提供保存位置

4.3 操作数类型

- 一般的数据类型：
  - 地址数据：地址实际上也是一种形式的数据，需要经过某种运算(映射算法)，才能确定其对应的在主存的有效地址。无符号整数。
  - 数值数据：三种类型的数值数据：
    - 定点数；
    - 浮点数；
    - 10进制数：压缩BCD与非压缩BCD
  - 字符数据：文本数据或字符串，目前广泛使用ASCII码。
    - ASCII码：8位2进制编码表示1个字符，最高位是校验位，可以表示128个字符数据。
  - 逻辑数据：一个单元中有几位二进制bit项组成，每个bit的值可以是1或0。当数据以这种方式看待时，称为逻辑性数据。

4.3 操作数类型

- **Pentium数据类型**：
  - 数据类型：8位，16位，32位，64位。
  - 见表4.6 P122
- **Power PC数据类型**：精简指令系统计算机
  - 数据类型：8位，16位，32位，64位。
    - 无符号字节
      无符号半字
      有符号半字
      无符号字
      有符号字
      无符号双字
      字节串
      浮点数

4.4 指令和数据的寻址方式

- 本节要解决的问题
  - 确定当前指令中各操作数的地址
  - 下一条指令的地址
- 操作数或指令在存储器中的地址：存储器中保存某个操作数或某条指令的存储单元，其存储单元的编号。
- 寻址方式：CPU根据指令中给出的地址码字段寻找相应的操作数的方式，或根据当前指令的保存位置与指令功能计算下一条指令的保存位置。寻址方式分为两类
  - 指令寻址方式：计算下一条指令的在内存中的保存位置；
  - 数据寻址方式：计算机指令中操作数的保存位置；

4.4.1 指令的寻址方式

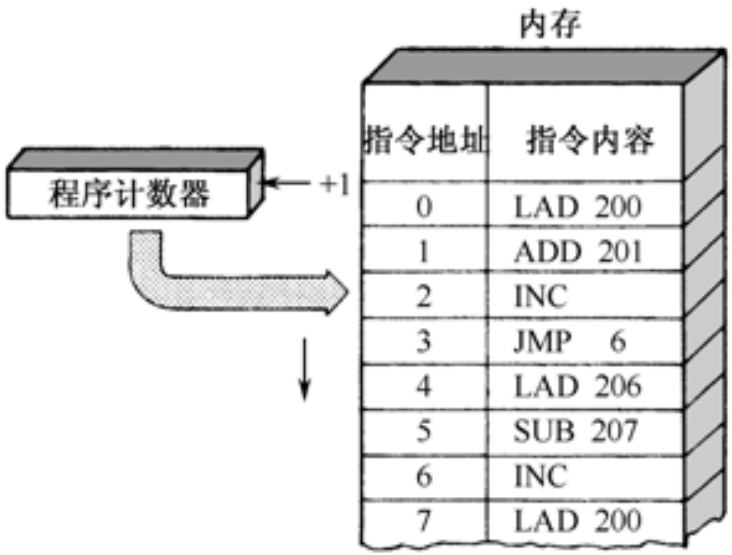
- 程序中的各条指令，依据在源程序中的位置，依次保存在一段连续的内存空间中，执行程序的时候，在获取程序保存位置入口点的条件下，就可以支持程序中的各条指令，根据当前执行指令的保存位置，获取下一条指令保存位置的方法有两种：
  - 顺序寻址方式。
  - 跳跃寻址方式。

内存

指令地址	指令内容	
0	LDA	200
1	ADD	201
2	INC	
3	JMP	6
4	LDA	206
5	SVB	207
6	INC	
7	LDA	200

4.4.1 指令的寻址方式

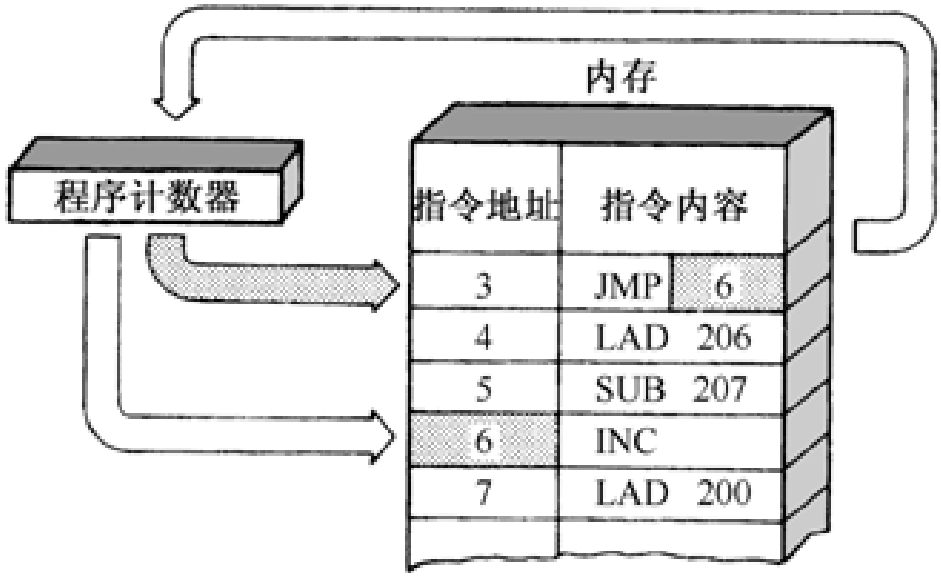
- 顺序寻址方式：
  - 当前执行的指令不是跳转指令，要计算下一条要执行指令的保存位置，使用顺序寻址方式。
  - 寻址方式：（PC）＋1 －＞ PC
    - （PC）＋指令字长度 －＞ PC
  - PC：程序计数器，指令指针寄存器，保存当前指令在内存中的保存位置。



(a) 指令的顺序寻址方式

4.4.1 指令的寻址方式

- 跳跃寻址方式：
  - 当前执行的指令是跳转指令(循环，选择等)，要计算下一条要执行指令的保存位置，使用顺序寻址方式。
  - 寻址方式：目标地址 －＞ PC
  - 目标地址：下一条指令在内存中的保存位置。使用当前指令的操作数计算得到目标地址。



(b) 指令的跳跃寻址方式

4.4.2 操作数基本寻址方式

- 操作数的保存位置：
  - 操作数包含在指令中；
  - 操作数包含在CPU的某一个内部寄存器中；
  - 操作数包含在主存储器中；
  - 操作数包含在I/O设备的端口中；
- 操作数的寻址方式：根据指令提供的操作数的地址码形成操作数有效地址EA的方法，称为寻址方式。
  - 地址码：包括形式地址和寻址方式的特征位，不是操作数的真正保存位置；
  - EA：effective address，规定操作数的真正保存位置；
    - 有效地址EA=***f***(形式地址，特征位)；

4.4.2 操作数基本寻址方式

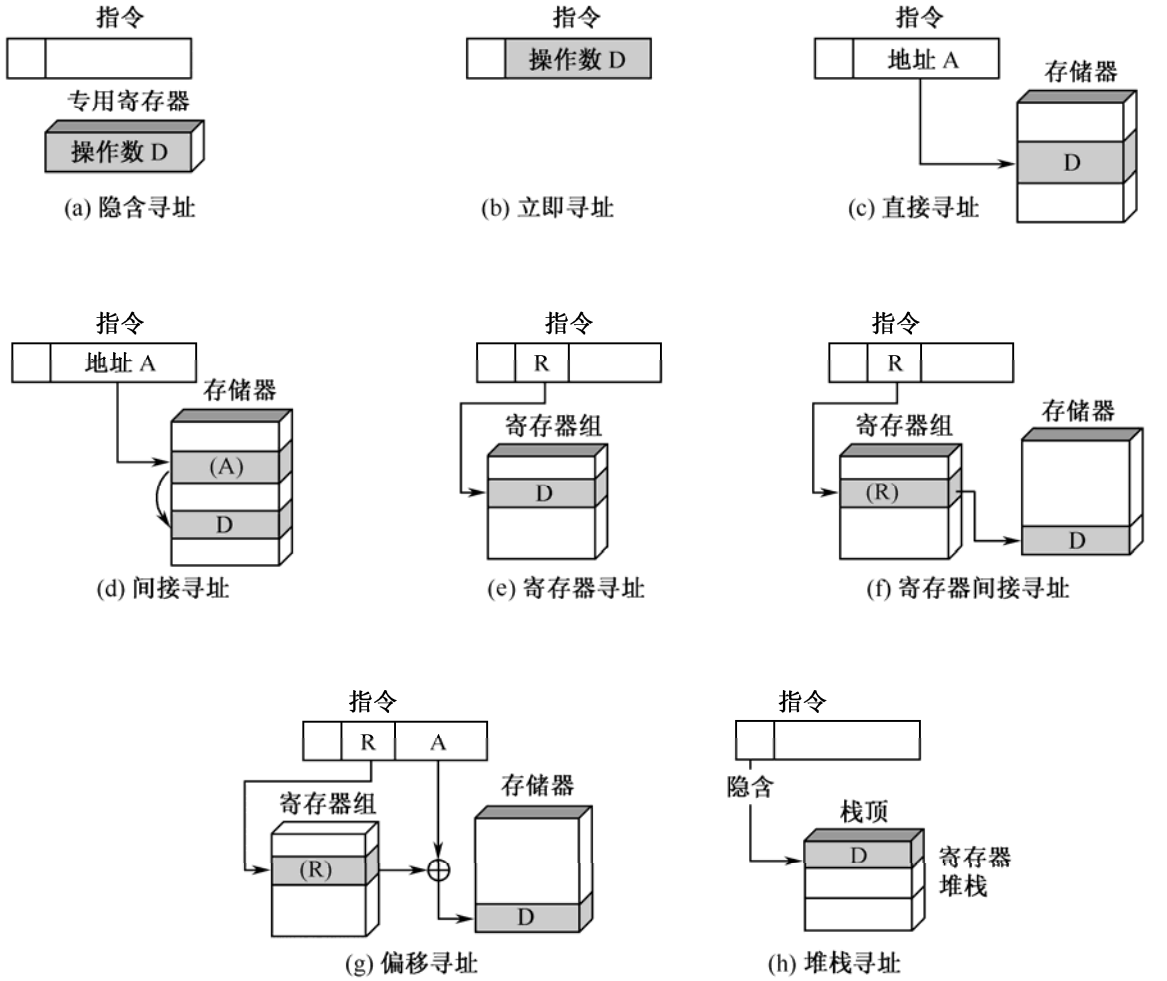
- 例：
  - **OP**：操作码
  - 地址码：X，I，A；
  - **X**：变址特征位；
  - **I**：间址特征位；
  - **A**：形式地址，地址偏移量；
  - **EA**：= ***f*** (X，I，A)；

指令编码

OP	变址X	间址I	形式地址A

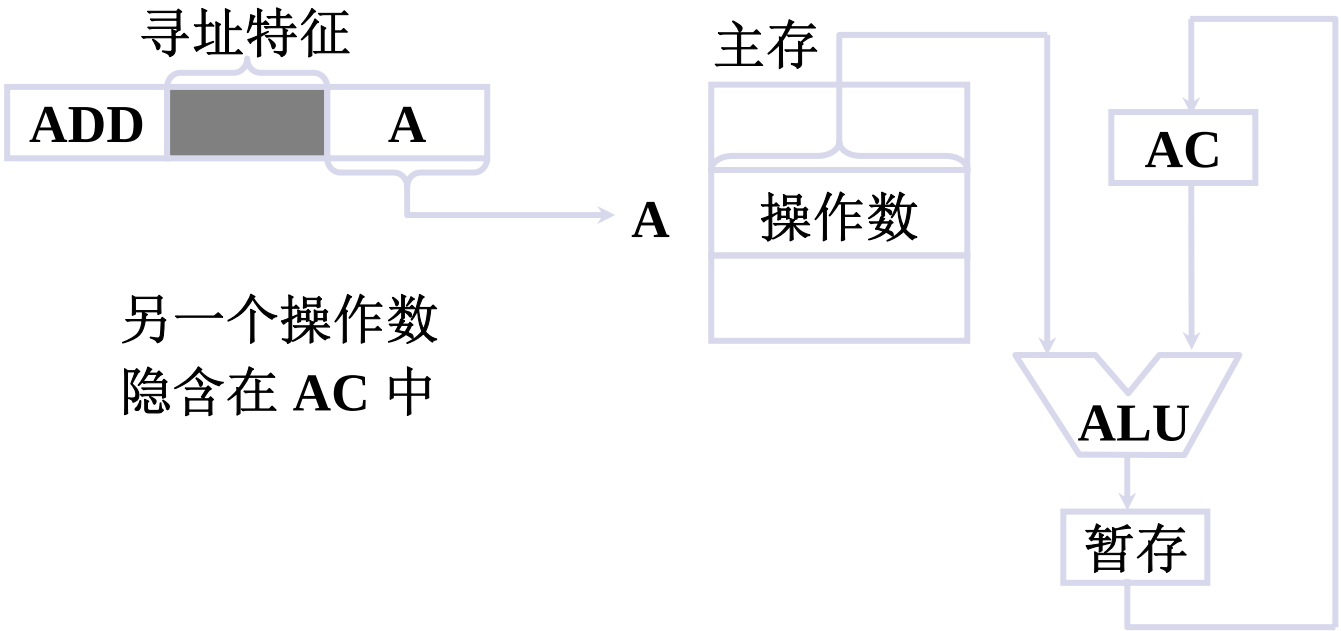
4.4.2 操作数基本寻址方式

- 例：由于不同机型的内部结构不同，从而形成了各种不同的操作数寻址方式，比较典型且常用的寻址方式如表4.7 P124，基本寻址示意图：



4.4.2 操作数基本寻址方式

- 隐含寻址：
  - 指令格式：**op**
  - **EA**：不需要。指令中无地址码。
  - 操作数：默认/隐含规定保存在某个特定的位置。
    - 运算类型的指令一般隐含保存在AC累加器中；
    - clc ; mul bx



4.4.2 操作数基本寻址方式

- 立即寻址：
  - 指令格式：**op** 地址码**A**
  - **EA**：不需要
  - 操作数：指令中地址码A直接提供操作数本身，也就是说数据就包含在指令中，只要取出指令，就取出了可以立即使用的操作数。
    - 操作数就保存在指令字中。
    - mov ax, 1234H

4.4.2 操作数基本寻址方式

- 直接/间接寻址：
  - 指令格式： op 间址I 地址码A
  - 直接寻址：
    - I=0;
    - EA=地址码A，指针？
    - 操作数：在地址EA指定的内存单元中；
      - 指令中的地址码A直接就是EA地址，不需要转换；
      - mov ax,[10H]
  - 间接寻址：
    - I=1;
    - EA=(地址码A)=地址码A指定位置保存的数，指针的指针？
    - 操作数：在地址EA指定的内存单元中；
      - 指令中的地址码A间接提供EA地址；
      - 访问两次存储器，很少使用；
      - les ax,[10H]

4.4.2 操作数基本寻址方式

- 寄存器寻址：
  - 指令格式： **op 间址I 地址码A→寄存器编号R**
  - 寄存器直接寻址：
    - **I=0;**
    - **EA**=寄存器编号R
    - 操作数：在地址EA指定的寄存器中；
      - 指令中的寄存器编号R直接就是EA地址，不需要转换；
      - mov ax,[10H]
  - 寄存器间接寻址：
    - **I=1;**
    - **EA**=(寄存器编号R)=编号指定的寄存器中的数
    - 操作数：在地址EA指定的内存单元中；
      - 指令中的寄存器编号R间接提供EA地址；
      - 寄存器编号在指令中使用寄存器名字表示；
      - mov[bx], ax

4.4.2 操作数基本寻址方式

- 偏移寻址：
  - 直接寻址和寄存器间接寻址的组合；
  - EA=A+(寄存器编号R)
  - 操作数：在EA指定的内存单元中。
  - 要求指令提供两个地址，地址A是显示的，地址R隐含规定在某个寄存器R中；或者两个都是显示的。
  - 3种偏移寻址：
    - 相对寻址
    - 基址寻址
    - 变址寻址

4.4.2 操作数基本寻址方式

- 偏移寻址：相对寻址
  - 指令格式： **op** 地址码**A**
  - 指令的地址码提供A地址，地址R隐含在程序计数器PC中；
  - **EA**=A+(PC)
  - 操作数；在EA指定的内存单元中。
    - jmp 10H
- 偏移寻址：基址寻址
  - **op** 地址码**A** 寄存器编号R
  - 指令的地址码提供A地址，地址R可以显示或者隐含在某个寄存器R中；
  - R地址是一个基地址/起始位置，A地址是一个相对于R地址的偏移量。R称为基址寄存器。
    - R是数组名，A是数组下标。
  - **EA**=A+(R)
  - 操作数；在EA指定的内存单元中。
    - mov ax,[bx+10H]

4.4.2 操作数基本寻址方式

- 偏移寻址：变址寻址
  - **op 地址码A** 寄存器编号R
  - 指令的地址码提供A地址，地址R可以显示或者隐含在某个寄存器R中；
  - A地址是一个基地址/起始位置，R地址是一个相对于A地址的偏移量。R称为变址寄存器。
    - A是数组名，R是数组下标。
  - **EA**=A+(R)
  - 操作数：在EA指定的内存单元中。
    - mov ax,[ARRAY+BX]

4.4.2 操作数基本寻址方式

- 段寻址：
  - 内存分段：在实模式中，将1M内存空间分为段为单位的空间进行使用。每个段的大小≤64K。
  - 寻址的问题：2^{20}=1M，2^{16}=64K，1M空间中的20位的地址=段起始位置地址(段基址)/20位+段内偏移量/16位，规定20位的段基址最低4为=0，高16位保存在cpu内部的段寄存器中。则20位地址=段基址×16+段内偏移地址
  - 段内偏移地址可以使用直接寻址，间接寻址，偏移寻址提供。
  - **op 地址码A /** 寄存器编号R 段寄存器编号SR
  - 指令的地址码提供A地址/R地址提供偏移地址，SR地址可以显示或者隐含在某个段寄存器SR中；
  - **EA**=SR×16+A+(R)
  - 操作数；在EA指定的内存单元中。
    - mov ax,ES:[10H]

4.4.2 操作数基本寻址方式

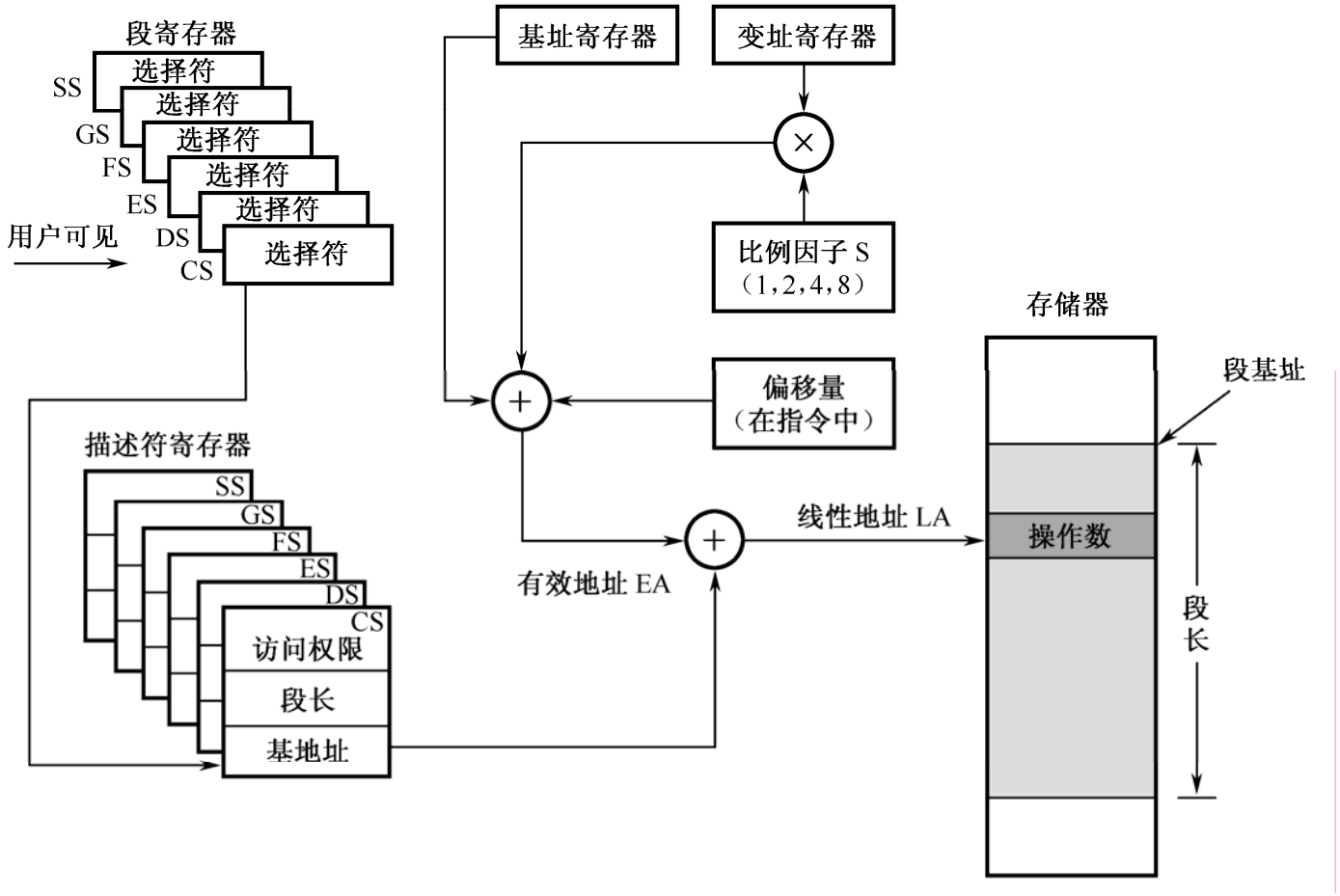
- 堆栈寻址：
  - 堆栈：先进后出/FILO/LIFO。区别队列：FIFO/LILO
  - 栈顶：保存最后进栈数据的位置。栈底：保存第一个进栈数据的位置。栈顶地址≤栈底位置。
  - 在堆栈中，数据的进出都在栈顶位置，设置一个栈顶指针寄存器/堆栈指示器R(SP/ESP)，保存栈顶位置。
  - 进栈PUSH：R减1，数据→(R)；
  - 出栈POP： (R) →数据，R加1；
  - 分类：寄存器堆栈和存储器堆栈。
  - push ax
  - pop bx

4.4.3 寻址方式举例

- **Pentium**的寻址方式：
  - Pentium外部地址总线宽度是36位，支持32位物理地址空间。按字节寻址。
  - 实地址模式，使用1M内存空间，兼容8086CPU。
    - 使用段寻址方式，逻辑地址=段基址(16位)：段内偏移地址(16位)。
    - 物理地址(20位) =段基址×16+偏移地址
  - 保护模式：32位段基地址加上段内偏移得到32位线性地址。由存储管理部件MMU将线性地址转换成32位的物理地址。
  - 段基址使用段寄存器提供，一般隐含。
  - 偏移地址在指令中使用地址码提供。

4.4.3 寻址方式举例

- **Pentium**的寻址方式：



4.4.3 寻址方式举例

- **Pentium**的寻址方式：

序号	寻址方式	有效地址EA	说明
1	立即寻址	操作数=A	操作数在指令中
2	寄存器寻址	EA=R	操作数在寄存器中
3	直接寻址	EA=A	操作数在内存中，A是内存地址
4	基址寻址	EA=(B)	B是基址寄存器，提供内存地址
5	基址+偏移量	EA=(B)+A	
6	比例变址+偏移量	EA=(I) ×S+A	B是基址寄存器，S是比例因子(1,2,4,8)
7	基址+变址+偏移量	EA=(B)+(I)+A	
8	基址+比例变址+偏移量	EA=(B)+(I) ×S+A	
9	相对寻址	PC=PC+A	PC程序计数器，当前指令地址

4.4.3 寻址方式举例

- **Pentium**的32位寻址方式说明：
  - 立即数可以是8位，16位，32位。
  - 寄存器地址：
    - 8位通用寄存器：AL,AH,BL,BH,CL,CH,DL,DH;
    - 16位通用寄存器：AX,BX,CX,DX,SI,DI,BP,SP;
    - 32位通用寄存器：EAX,EBX,ECX,EDX,ESI,EDI,EBP,ESP;
  - 段寄存器：CS,DS,ES,SS,GS,FS;
  - 直接寻址：偏移量寻址方式，偏移量长度可以是8位，16位，32位。
  - 基址寻址/基址＋偏移量寻址：基址寄存器B可以是上述通用寄存器中任何一个。
  - 比例地址＋偏移量寻址：变址寻址方式，变址寄存器I是32位通用寄存器中除ESP外的任何一个。
  - **7，8**两种寻址方式是4，6两种寻址方式的组合，偏移量可有可无。
  - 相对寻址：适用于转移控制类指令。用当前指令指针寄存器EIP或IP的内容（下一条指令地址）加上一个有符号的偏移量，形成跳转的目的位置。

4.4.3 寻址方式举例

- 例：一种二地址RS型指令的结构如下：
  - OP：操作码
  - R：通用寄存器
  - I：间接寻址标志位
  - X：寻址模式字段
  - D：偏移量字段
  - 通过I，X，D的组合，可构成如下源操作数的寻址方式：

6位	4位	1位	2位	16位
OP	R	I	X	D

寻址方式	I	X	有效地址EA	说明
直接寻址	0	00	EA=D	
相对寻址	0	01	EA=(PC)±D	PC程序计数器
变址寻址	0	10	EA=(R_2) ±D	R_2变址寄存器
寄存器间接	1	11	EA=(R_3)	R_3寄存器
间接寻址	1	00	EA=(D)	
基址寻址	0	11	EA=(R_1) ±D	R_1基址寄存器

4.5 典型指令

- 指令的分类：一个完善的指令系统应该具有数据处理、数据存储、数据传送、程序控制4大类指令：
  - 数据传送类指令
    - 一般传送指令： MOV…
    - 数据交换指令： XCHG …
    - 堆栈操作指令： PUSH，POP …
  - 运算类指令
    - 算术运算指令： ADD,SUB,MUL,DIV …
    - 逻辑运算指令：AND,OR,NOT,XOR …
    - 移位指令：SHR,SHL,SAR,SAL …
  - 程序控制类指令
    - 程序控制类指令用于控制程序的执行方向，并使程序具有测试、分析与判断的能力。实现程序的选择，循环结构。
    - JMP,LOOP…

4.5 典型指令

- 指令的分类：一个完善的指令系统应该具有数据处理、数据存储、数据传送、程序控制4大类指令：
  - 输入和输出指令
    - 实现CPU与计算机端口数据交换：IN,OUT…
  - 字符串处理指令
    - 运算数据是字符串数据：MOVS,CMPS…
  - 特权指令
    - 实现某些特权操作：LOCK…
  - 其他指令：NOP,HLT…
- 基本指令系统
  - P132表4.11

4.6 ARM汇编语言

- 汇编语言是计算机机器语言（二进制指令代码）进行符号化的一种表示方法，每一个基本汇编语句对应一条机器指令。
- 表4.13P134，列出了嵌入式处理机ARM的汇编语言。其中操作数使用16个寄存器（r0 , r1～r12 , sp , lr , pc）,2^{30}个存储字（字节编址，连续的字的地址间相差4）。