计算机组成原理与系统结构

第六章 指令系统

http://jpkc.hdu.edu.cn/computer/zcyl/dzkjdx/

杭州電子科技大學
HANGZHOU DIANZI UNIVERSITY

第六章 指令系统

6.1 指令格式

机器指令是指能被计算机硬件识别并执行的0、1代码串。

指令系统是一台计算机中所有机器指令的集合，它体现了计算机的性能。

一、指令操作码与地址码

二、指令字长和操作码扩展

一、指令操作码与地址码

指令是由操作码和地址码两部分组成的：

操作码字段（OP）	地址码字段（A）

1. **操作码：**用来指明该指令所要完成的操作，如加法、减法、传送、移位、转移等等。
   - 位数反映了机器的操作种类，也即机器允许的指令条数，如果操作码有n位二进制数，则最多可表示2^n种指令。

一、指令操作码与地址码

指令是由操作码和地址码两部分组成的：

操作码字段（OP）	地址码字段（A）

1. **地址码：** 用来寻找运算所需要的操作数（源操作数和目的操作数）。
   - 地址码包括：源操作数地址、目的操作数地址和下一条指令的地址。
   - 地址含义：主存的地址、寄存器地址或者I/O设备地址。

一、指令操作码与地址码

1、操作码

- **操作码长度固定：**将操作码集中放在指令字的一个字段内。

  这种格式便于硬件设计，指令译码时间短，广泛应用于字长较长的、大中型计算机和超级小型计算机以及RISC（Reduced Instruction Set Computer）中。如IBM370和VAX-11系列机，操作码长度均为8位。

- **操作码长度不固定：**指令操作码分散在指令字的不同字段中。

  这种格式可有效地压缩操作码的平均长度，在字长较短的微机中被广泛采用。如PDP-11，Intel8086/80386等。

2、地址码

（1）三地址指令：

- （A1）OP（A2）→A3

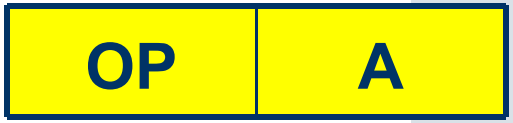
（2）二地址指令：

- （A1）OP（A2）→A1
- A1：目的操作数
- A2：源操作数

（3）单地址指令：

- （ACC）OP（A）→ACC
- OP（A）→A

  单目操作：如NEG、INC等指令



2、地址码

（4）零地址指令

- 不涉及操作数：如NOP、HLT指令
- 操作数隐含：如PUSH、POP指令

OP

对于寄存器类型的操作数，地址A指寄存器编号。

3、操作数类型

1. **按照指令处理的操作数存放位置分：**
   - 存储器类型：操作数存放在主存中，A为其地址信息
   - 寄存器类型：操作数存放在CPU的通用寄存器中，A为寄存器号
   - 立即数类型：操作数存放在指令（地址字段）中
2. **按照指令处理的操作数性质分：**
   - 地址（addresses）：存储器地址，是无符号整数。
   - 数字（numbers）：整数、浮点数、十进制数。
   - 字符（characters）
   - 逻辑数据：真假两种状态

二、指令字长和操作码扩展

1、指令字长度

2、指令操作码扩展

1、指令字长度

- 机器指令是用二进制机器字来表示的，表示一条指令的机器字，就称为指令字。一条指令中所包含的二进制码的位数，称为指令字长度或指令字长。它主要取决于操作码的长度、操作数地址的长度和操作数地址的个数。不同机器的指令字长是不相同的。
- 按指令长度固定与否可以分为：

  ①固定指令字长的指令：所有指令的字长均相等，一般等于机器字长。

  ②可变指令字长的指令：指令字长不固定，通常取字节的整数倍。

1、指令字长度

- 按照指令字长与机器字长的关系分类：

①短格式指令：指令字长小于或等于机器字长。

②长格式指令：指令字长大于机器字长。

2、指令操作码扩展

- 固定操作码长度的格式和可变操作码长度格式
- 在设计操作码不固定的指令系统时，应安排指令使用频度高的指令占用短的操作码，对使用频度低的指令可占用较长的操作码，这样可以缩短经常使用的指令的译码时间。
- 采用扩展操作码技术，使操作码的长度随地址数的减少而增加，即不同地址数的指令可以具有不同长度的操作码，从而可以有效地缩短指令字长。
- 优缺点：指令操作码扩展技术是一种重要的指令优化技术，它可以缩短指令的平均长度，增加指令字所能表示的操作信息。但指令操作码扩展技术需要更多的硬件支持，它的指令译码更加复杂，使控制器设计难度增大。

举例

4位操作码,15条三地址指令

8位操作码,15条二地址指令

12位操作码,15条一地址指令

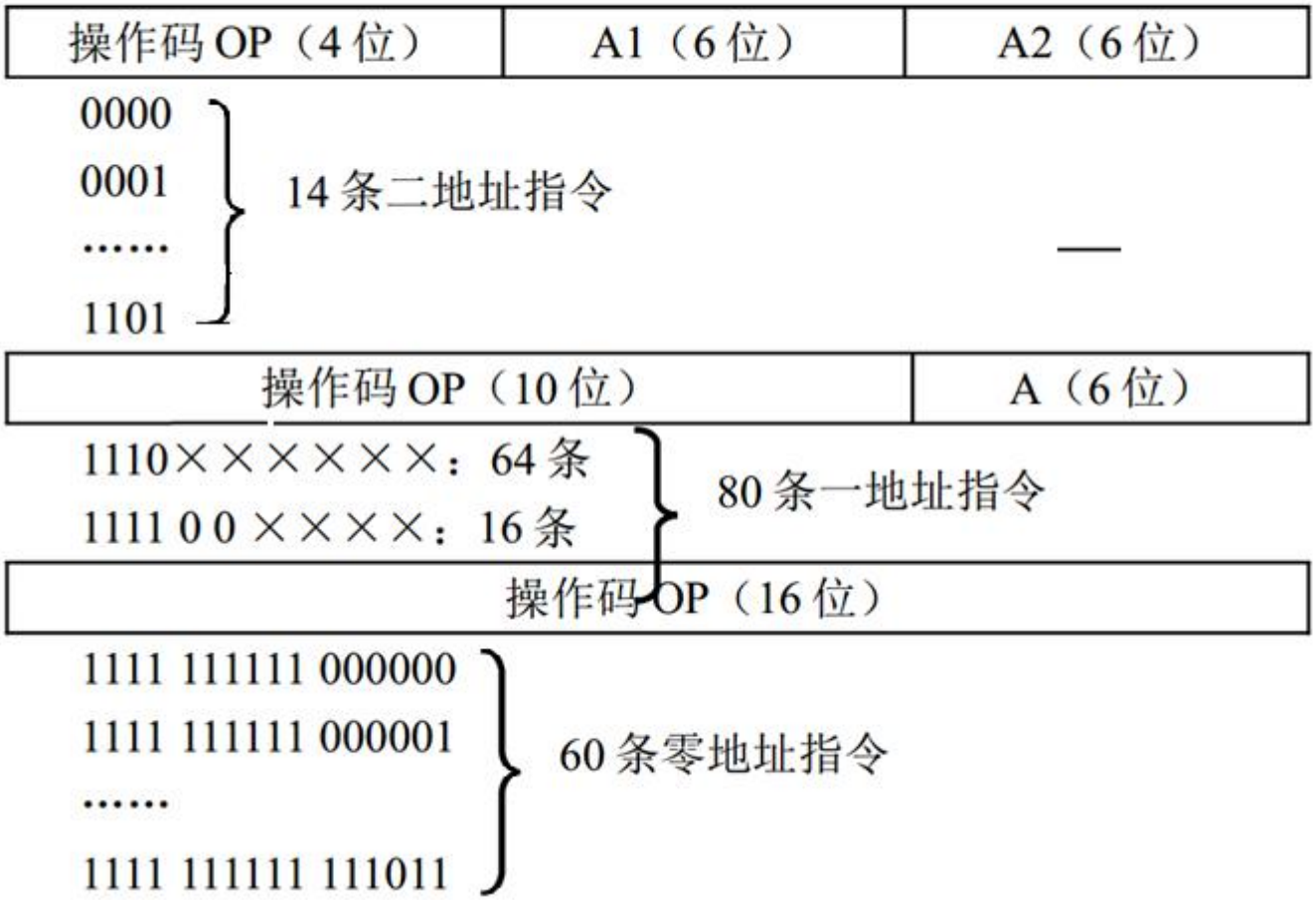
16位操作码

OP	A_1	A_2	A_3
0000 : 1110	A_1 : A_1	A_2 : A_2	A_3 : A_3
1111 : 1111	0000 : 1110	A_2 : A_2	A_3 : A_3
1111 : 1111	1111 : 1111	0000 : 1110	A_3 : A_3
1111 : 1111	1111 : 1111	1111 : 1111	0000 : 1111

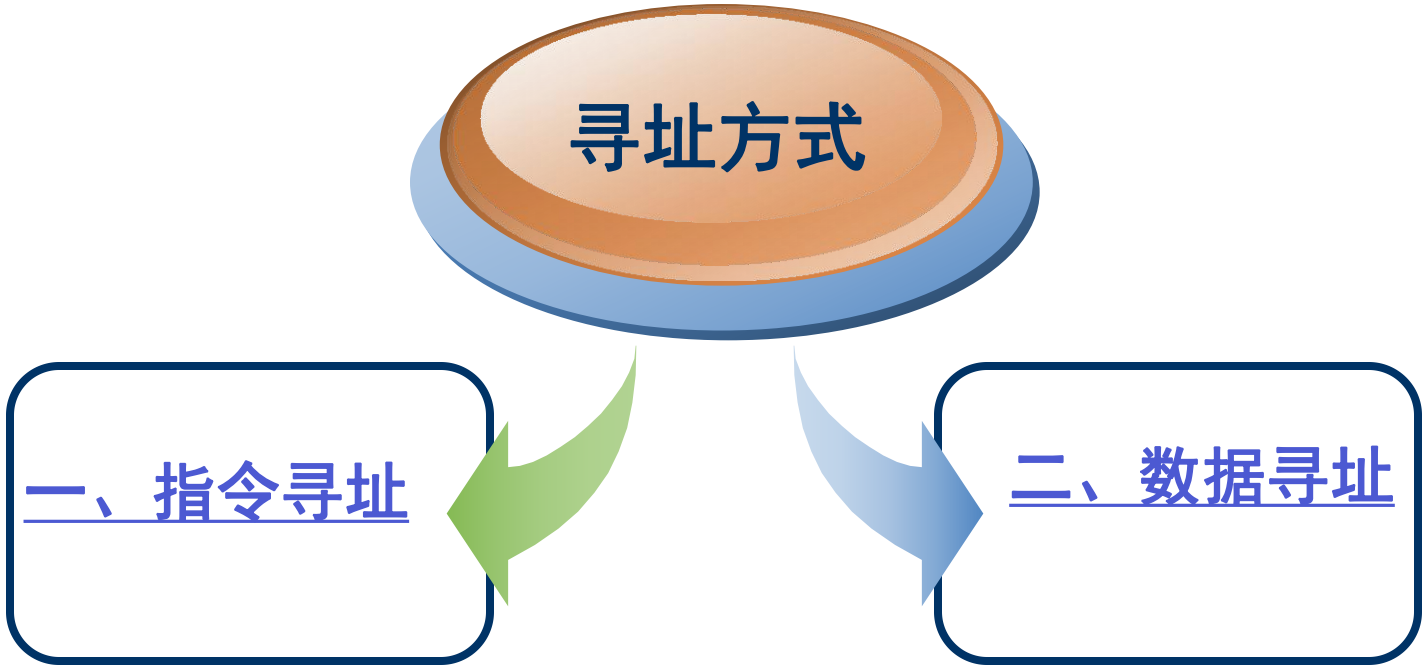
课堂练习

1. 某机器字长 16 位，采用单字长指令，每个地址码 6 位。试采用操作码扩展技术，设计14 条二地址指令，80 条一地址指令，60 条零地址指令。请给出指令编码示意图。

不唯一，其他不冲突和重复的编码方式也是正确的。



6.2 寻址方式



一、指令寻址

1. 顺序寻址方式

- 控制器中使用程序计数器PC来指示指令在内存中的地址。在程序顺序执行时，指令的地址码由PC自加1得出。
- 指令在内存中按顺序存放，当顺序执行一段程序时，根据PC从存储器取出当前指令， PC自动＋1，然后执行这条指令；接着又根据PC指示从存储器取出下一条指令， PC自动＋1， 执行……。

2. 跳跃寻址方式

- 当程序执行转移指令时，程序不再顺序执行，而是跳转到另一个地址去执行，此时，由该条转移指令的地址码字段可以得到新指令地址，然后将其置入PC中。

指令寻址

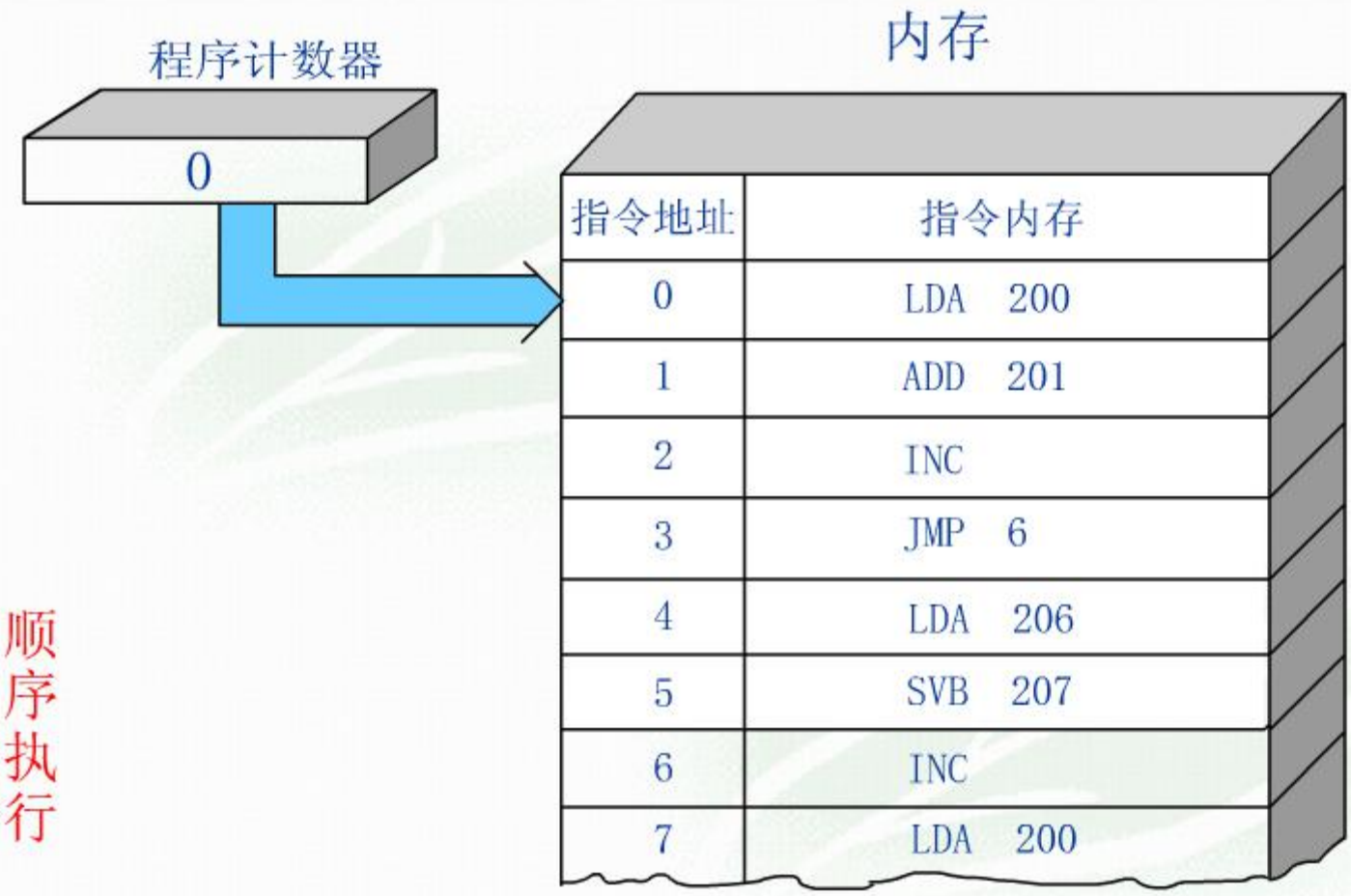
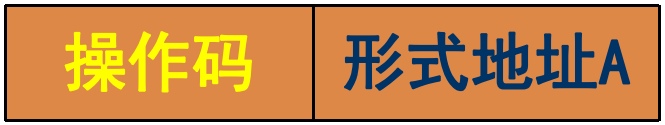


图4.1 指令的寻址方式

二、数据寻址

- **形式地址**：指指令的地址码字段，通常都不代表操作数的真实地址，记为A。
- **有效地址**：指操作数的真实地址，记作EA，它是由寻址方式和形式地址共同来确定的。
  - 问题：在实地址模式和虚拟地址模式下，EA是？
- 常见的有9种基本的寻址方式
  - 复合寻址？
- 所有的计算机CPU均采用多种寻址方式
  - 问题：如何识别？



预备知识

常用汇编助记符

- ADD：加法
- SUB：减法
- JMP：无条件转移
- MOV：传送字或字节
- STA：存数
- IN：I/O端口输入
- OUT： I/O端口输出

- INC：加1
- DEC：减1
- CMP：比较
- HLT：处理器暂停

1、立即寻址(Immediate Addressing)

- 操作数在指令的地址码字段，即：

  DATA＝A

- 例如：

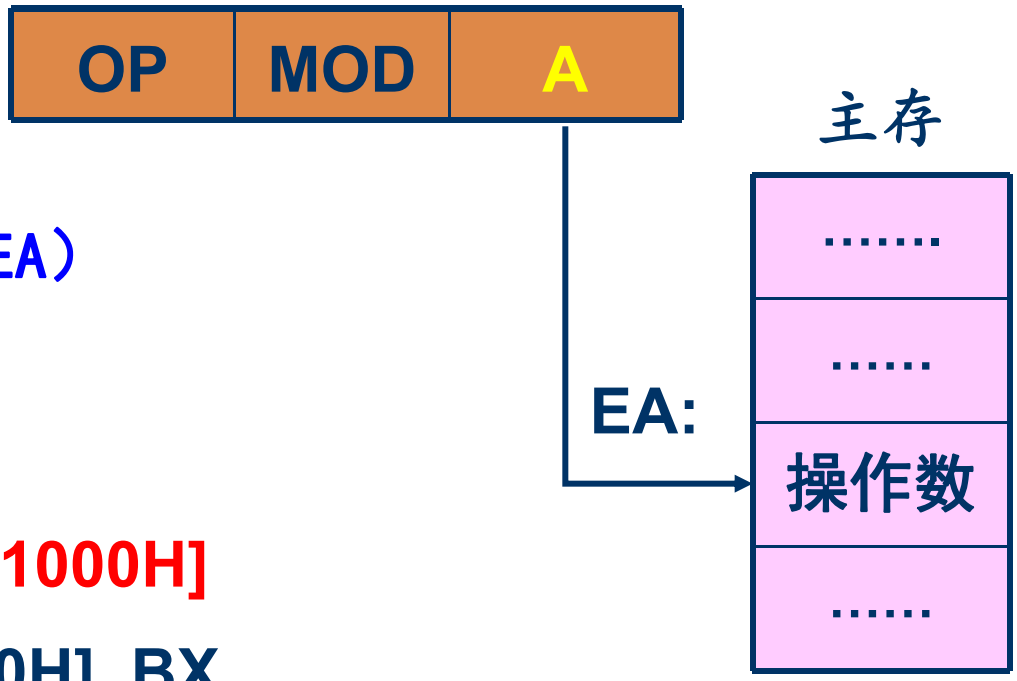
  MOV　AL，5

  MOV　AX，3064H

  MOV　AL，‘A’

2. 直接寻址(Direct Addressing)

- 操作数位于存储器中，操作数所在的存储器单元的地址存放在指令的地址字段A中，即：



DATA＝（EA）

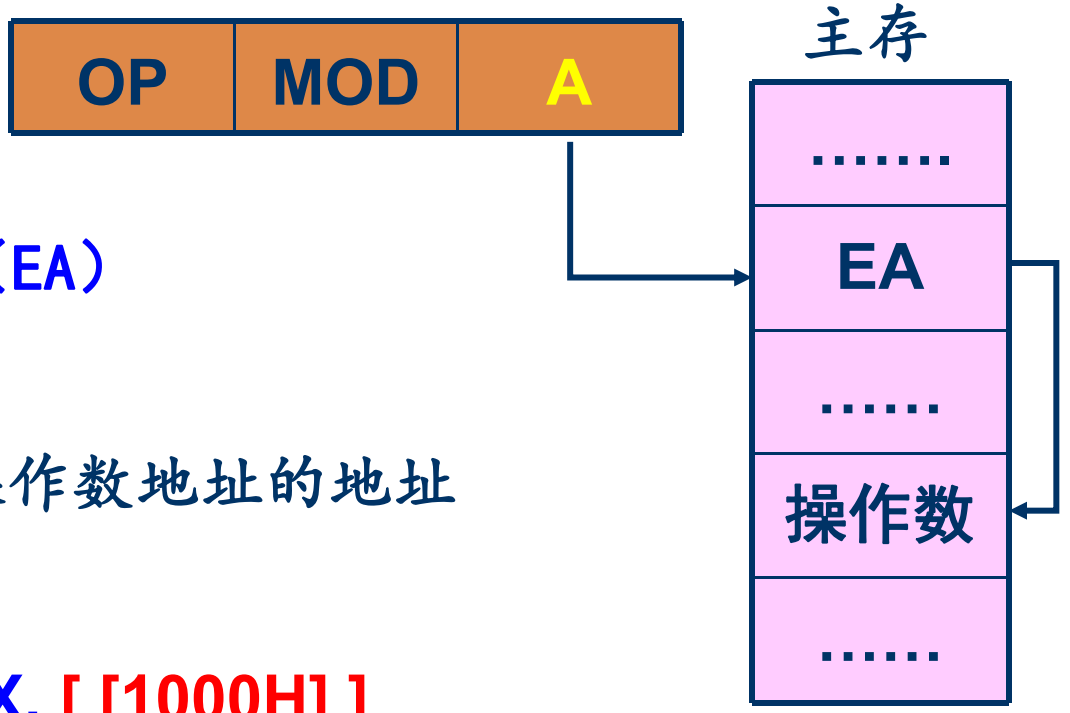
EA=A

例如：

- MOV AX, [1000H]
- ADD [2000H], BX

3、间接寻址（ Indirect Addressing ）

- 操作数位于存储器中，操作数所在的存储器单元地址也存放在存储器中，该存储器地址则存放在指令的地址字段中，即：



DATA＝（EA）

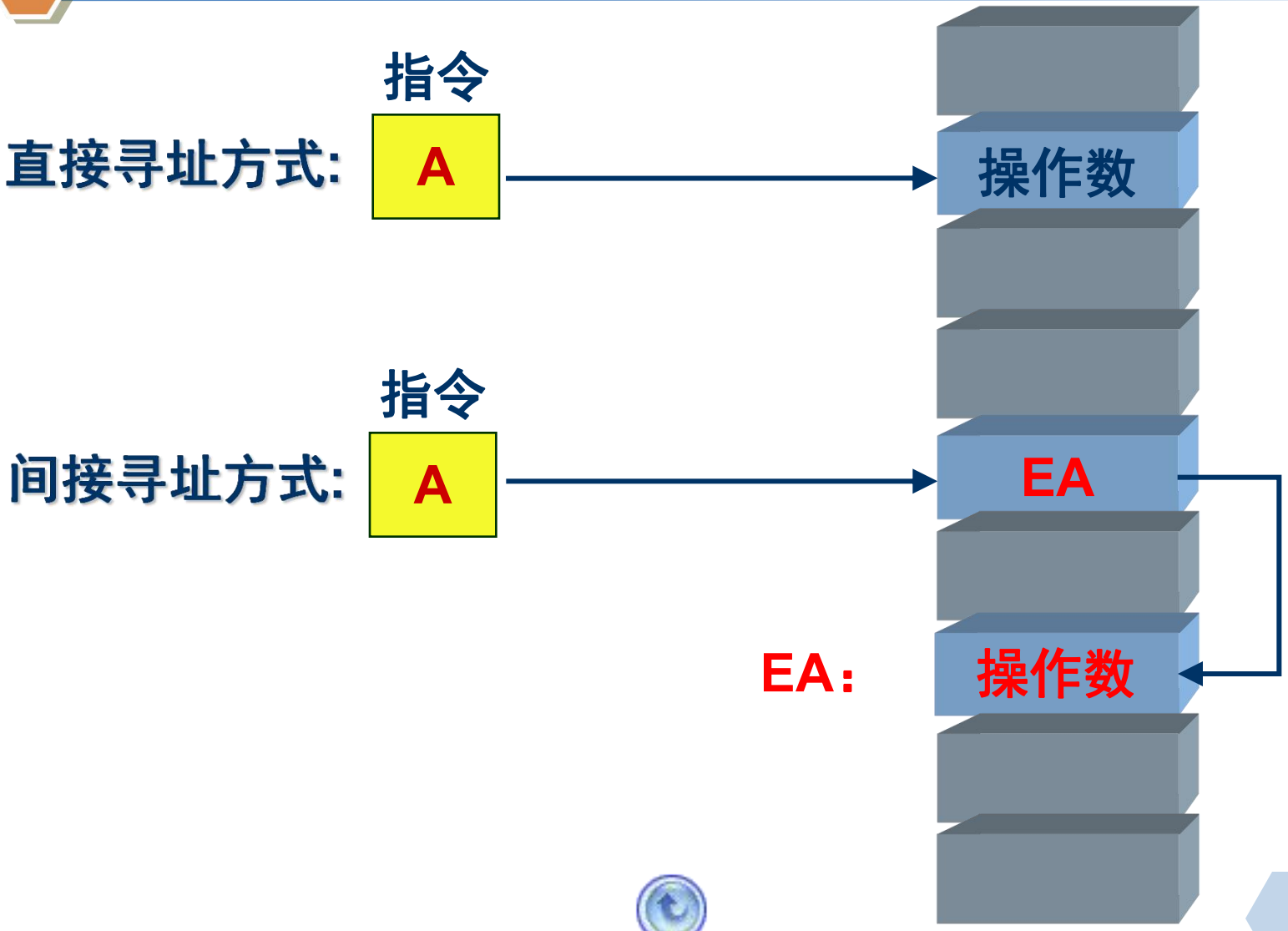
EA=（A）

- 即：A为操作数地址的地址

例如：

MOV AX,[[1000H]]

存储器



4、寄存器寻址方式（ Register Addressing ）

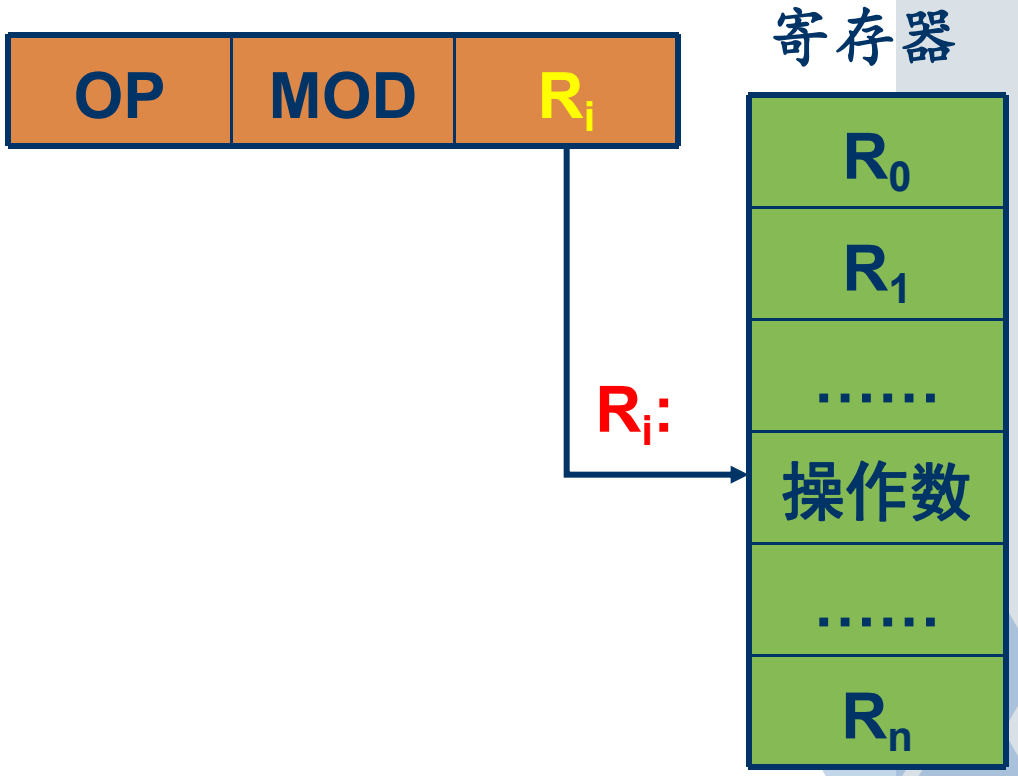
- 操作数位于寄存器中，操作数所在的寄存器编号存放在指令的地址字段A中，即：

$DATA = (R_i)$

例如：

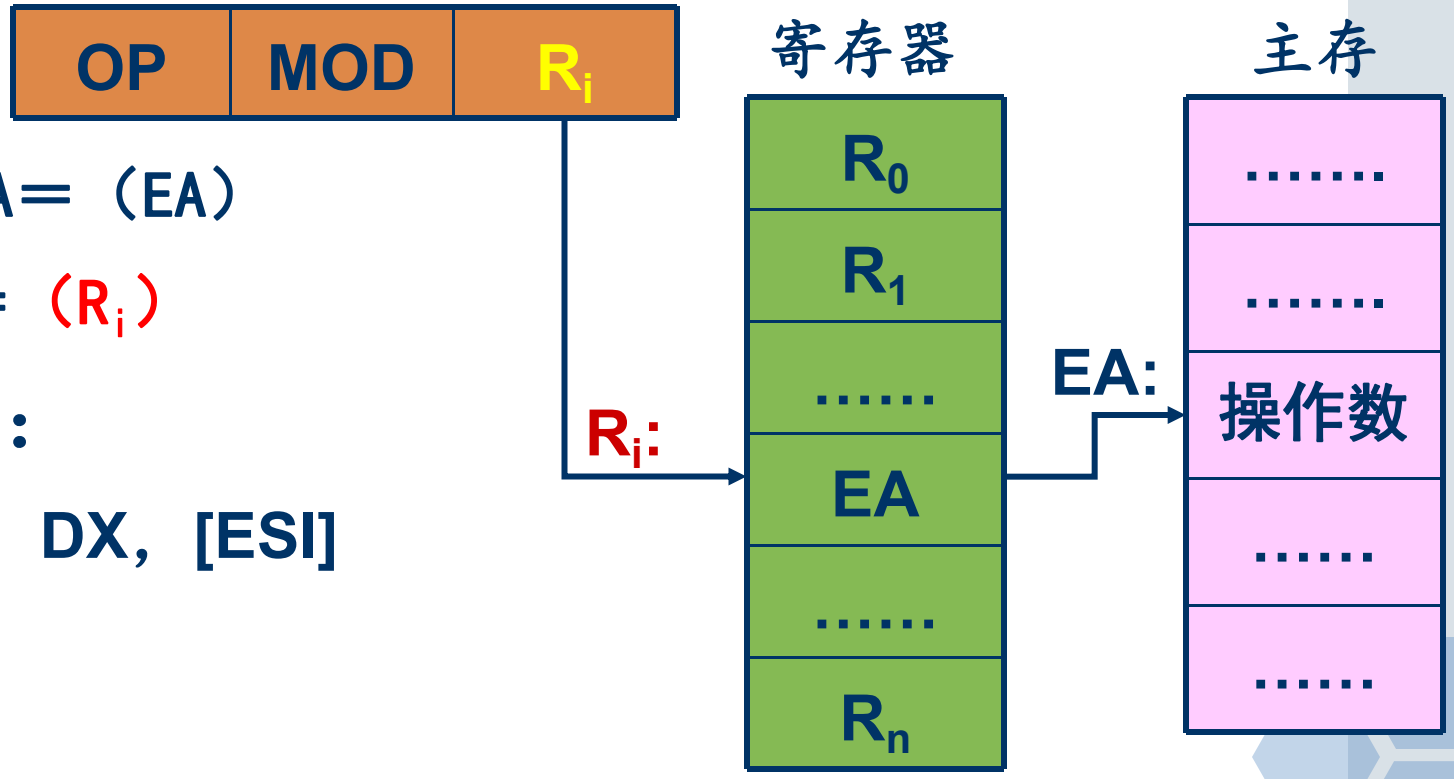
MOV AX, BX ;

MOV AL, BH ;



5、寄存器间接寻址方式

- 操作数位于存储器中，操作数所在的存储器地址存放在寄存器中，而该寄存器编号存放在指令的地址字段A中，即：

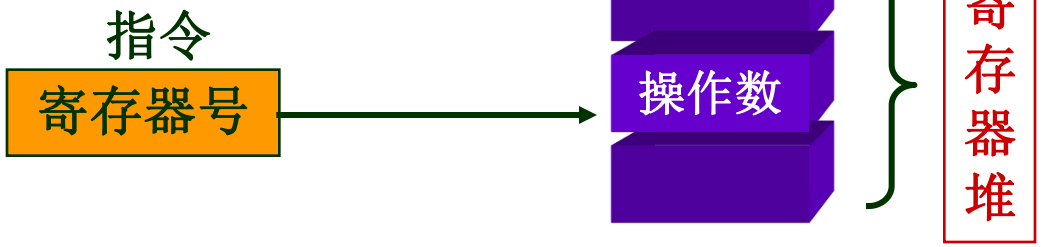


- DATA＝（EA）
- EA＝（R_i）

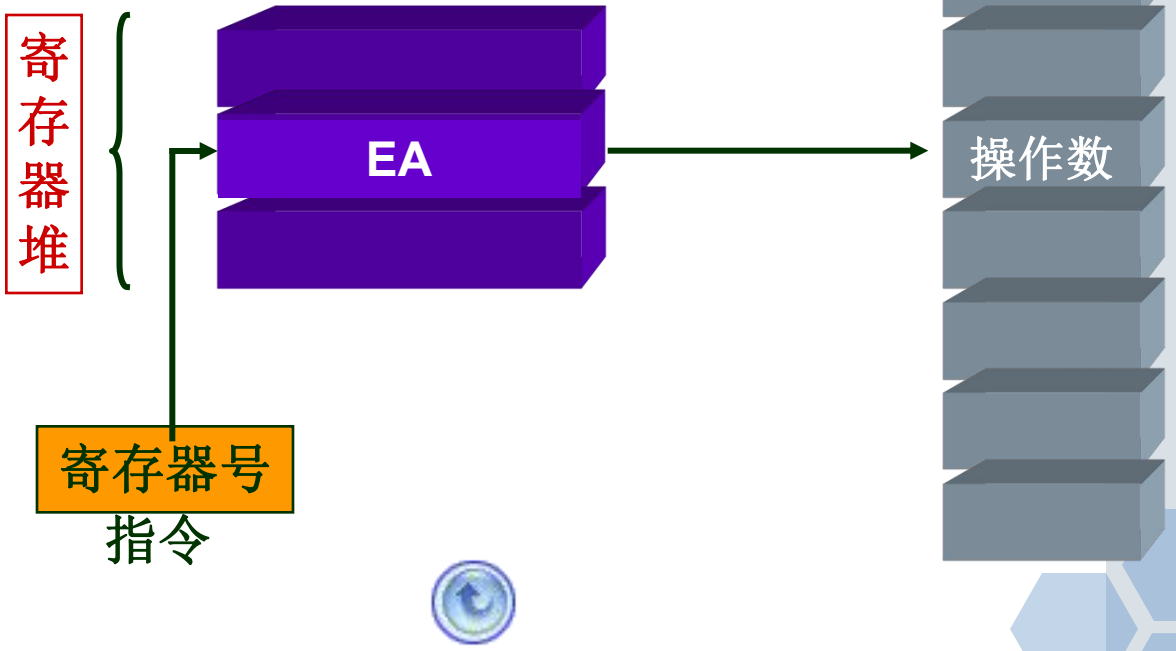
例如：

SUB　DX，[ESI]

寄存器寻址方式：



寄存器间接寻址方式：



6、变址寻址（Indexed Addressing）

- 操作数位于存储器中，操作数所在的存储器地址EA由变址寄存器R_i和指令的地址字段A指出：

$$DATA=(EA)$$

$$EA=(R_i)+A$$

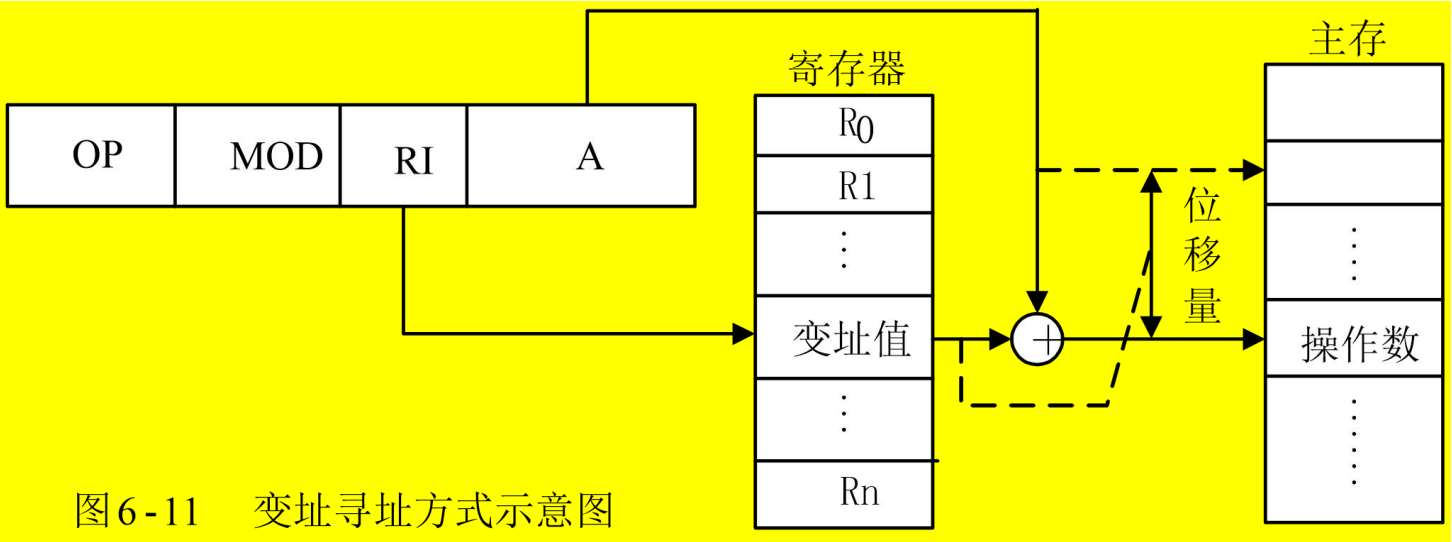


图6-11　变址寻址方式示意图

6、变址寻址（Indexed Addressing）

❖ 例如：

```
.data
   str_tb db     ‘Abort，Retry？’，0
.code
   MOV   ESI,0
   MOV   AL, str_tb[ESI]
   ……
   INC    ESI
   MOV   AL, str_tb[ESI]
```

串首址

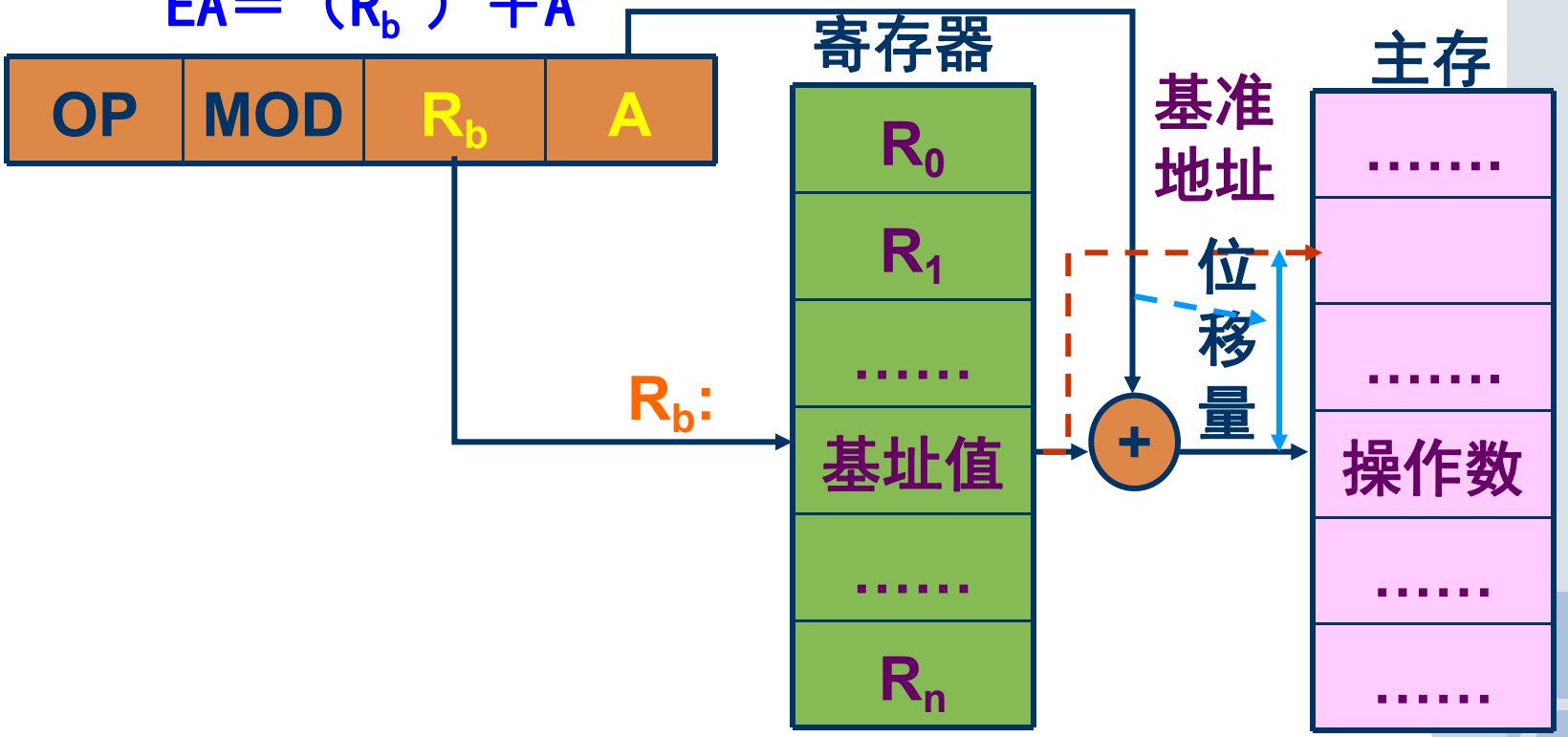
变址寄存器

7、基址寻址（Based Addressing）

- 操作数位于存储器中，操作数所在的存储器地址EA由基址寄存器Rb和指令的地址字段A指出：

  DATA＝（EA）

  EA＝（R_b）＋A

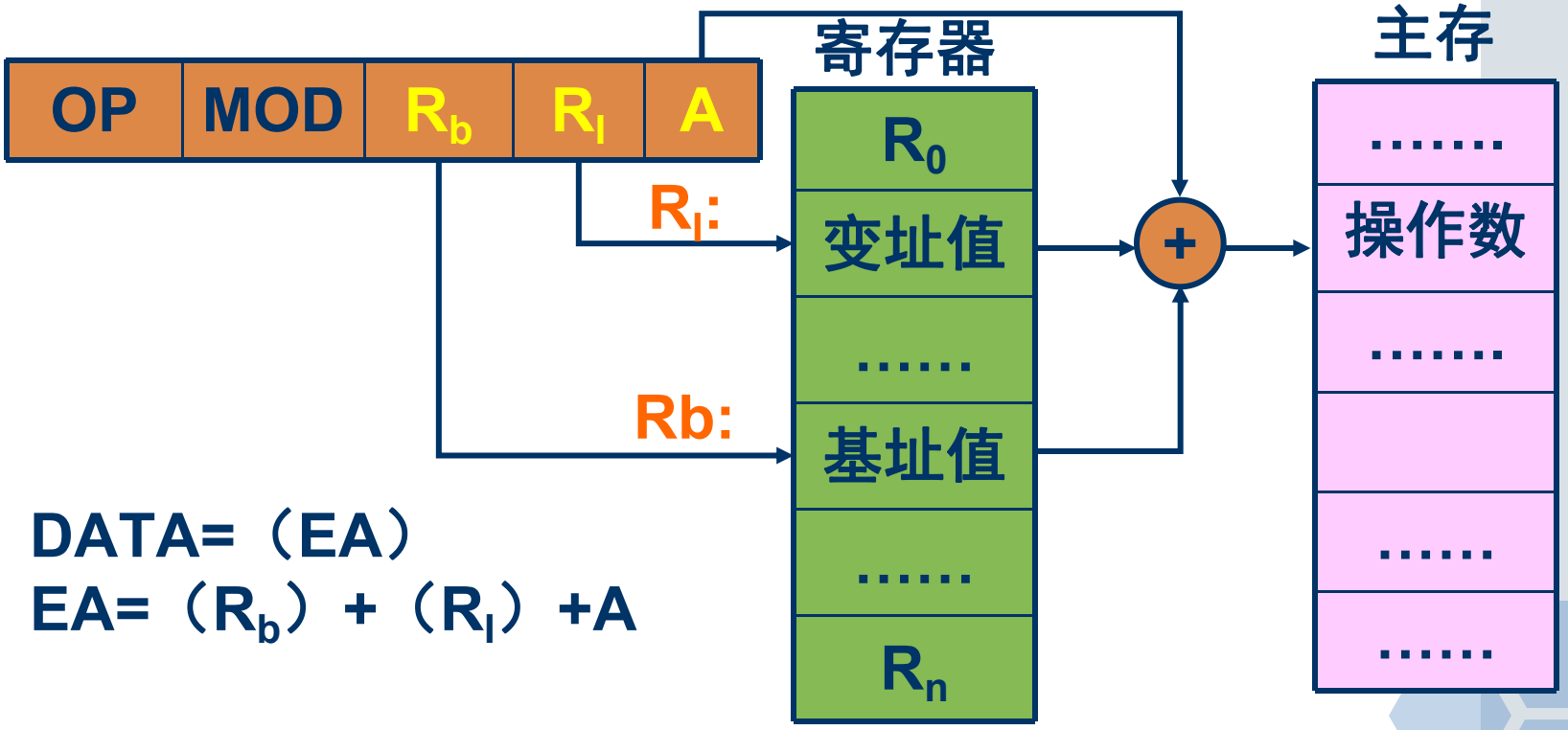


7、基址寻址（Based Addressing）

- 基址寻址适合于多用户计算机系统，当操作系统为多道程序分配主存空间，将用户程序装入主存时，需要进行逻辑地址到物理地址的变换。
- 操作系统给每个用户一个基地址并放入其相应的基址寄存器，在程序执行时，以基址为基准自动进行逻辑地址到物理地址的变换。
- 在应用场合上，基址寻址面向系统，可以用来解决程序在主存中的重定位和扩大寻址空间等问题。而变址寻址是面向用户编程，用来访问字符串、向量和成批数据。

8、基址变址寻址

- 在指令中指定一个基址寄存器和一个变址寄存器，指令中的地址码给出位移量。有效地址是由基址寄存器中的值、变址寄存器中的值和位移量三者相加而成。



- DATA=（EA）
- EA=（R_b）+（R_I）+A

9、相对寻址（Relative Addressing）

- 操作数位于存储器中，操作数所在的存储器地址EA由程序计数器PC和指令的地址字段A指出：

  DATA＝（EA）

  EA＝（PC）＋A

  ① A 通常称作相对偏移量DISP。

  ②相对寻址主要用于转移指令，执行之后，程序将转移到（PC）＋偏移量为地址的指令去执行。

  ③偏移量可正、可负，通常用补码表示，即可相对PC值向后或向前转移。

10、堆栈寻址（ Stack Addressing ）

- 操作数位于存储器中，操作数所在的存储器地址 EA 由堆栈指针寄存器 SP 隐含指出，通常用于堆栈指令。
- 堆栈是由若干个连续主存单元组成的先进后出（first in last out，即FILO）存储区
- 第一个放入堆栈的数据存放在栈底，栈底是固定不变的。
- 最近放入的数据存放在栈顶，栈顶随着数据的入栈和出栈在时刻变化。
- 栈顶的地址由堆栈指针SP指明。

10、堆栈寻址（ Stack Addressing ）

- 计算机中，堆栈从高地址向低地址扩展，即栈底的地址总是大于或等于栈顶的地址，称为上推堆栈。也有少数计算机相反，称为下推堆栈。
- 堆栈寻址主要用来暂存中断和子程序调用时现场数据及返回地址。
- 堆栈指针的管理：

  ①SP总是指向最后压入的有效数据

  ②SP总是指向栈顶的空单元

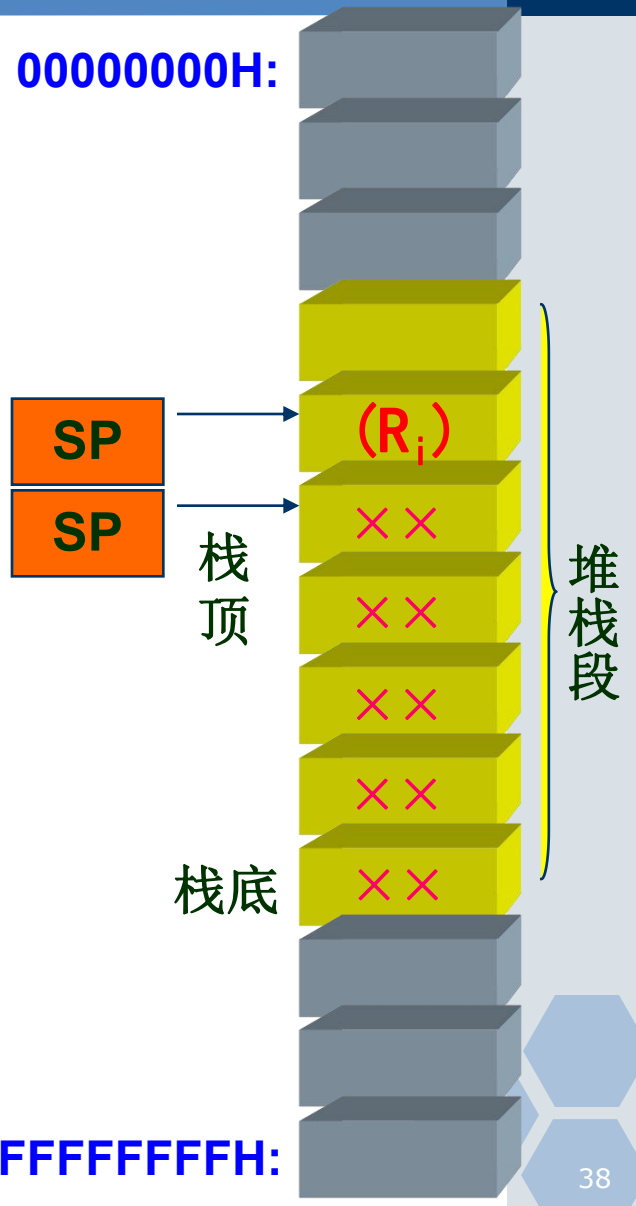
堆栈的结构

- 堆栈的操作：压入（PUSH）和弹出（POP），对应PUSH和POP指令，假设数据字长为1B

①压入指令 PUSH R_i：将R_i寄存器内容压入堆栈。其操作是：

$(SP)-1\rightarrow SP$,

$(R_i)\rightarrow (SP)$



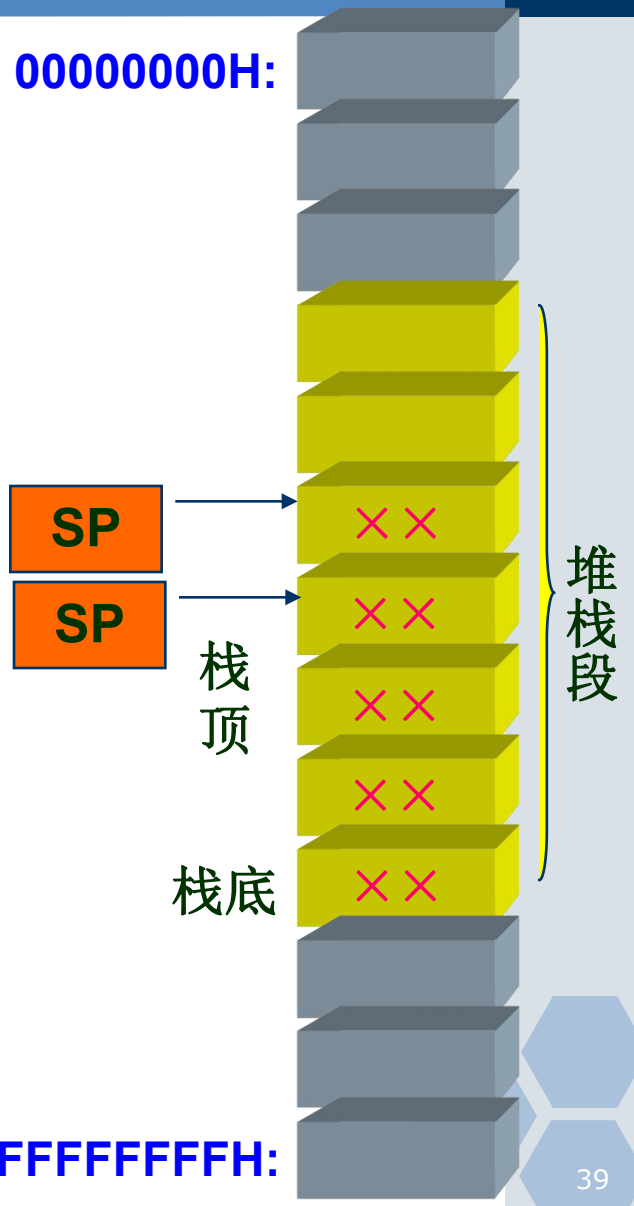
堆栈的结构

②弹出指令POP Ri：从堆栈中弹出1个数据送Ri寄存器，其操作是：

$((SP)) \rightarrow R_i$，

$(SP)+1 \rightarrow SP$

③(SP)表示堆栈指针SP的内容；((SP))表示SP所指的栈顶的内容。



总结

寻址方式	操作数类型	地址码字段A	EA
立即寻址	立即数	立即数Data	——
直接寻址	存储器	直接地址A	EA=A
间接寻址	存储器	间接地址A	EA=(A)
寄存器寻址	寄存器	寄存器号n	——
寄存器间接寻址	存储器	寄存器号n	EA=（R_n）
变址寻址	存储器	形式地址X	EA=（R_i）+X
基址寻址	存储器	相对偏移量Disp	EA=（R_b）+Disp
相对寻址	存储器	相对偏移量Disp	EA=（PC）+Disp

课堂练习

请指出对应的寻址方式，并解释其意义

- ADD R_0, R_1
- ADD R_0, #DATA
- ADD R_0, [ADDR]
- ADD R_0, [R_1]
- ADD R_0, [[ADDR]]
- ADD [ADDR], R_1
- ADD R_0, [PC+DISP] （DISP表示为偏移量）

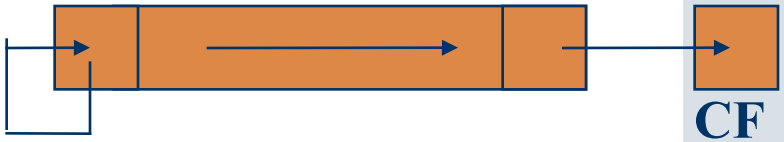
6.3 指令类型

- 数据传送指令
  - 包括寄存器与寄存器、寄存器与存储单元、存储单元与存储单元之间的传送。
- 算术逻辑运算指令
  - 实现算术运算（加、减、乘、除等）和逻辑运算（与、或、非、异或）。有些计算机还设置有位操作指令，如位测试（测试指定位的值）、位清零、位求反指令等。
- 移位操作指令
  - 可分为算术移位、逻辑移位和循环移位。

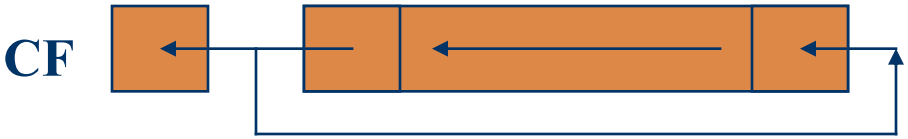
逻辑左移、算术左移

逻辑右移

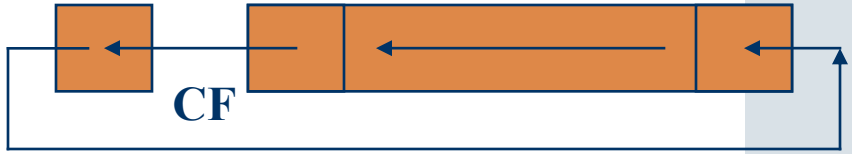
算术右移



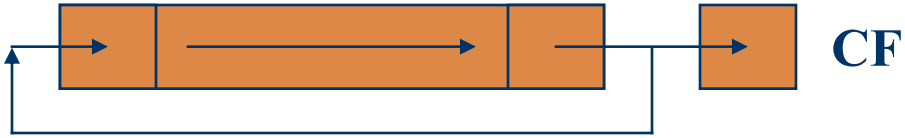
循环左移



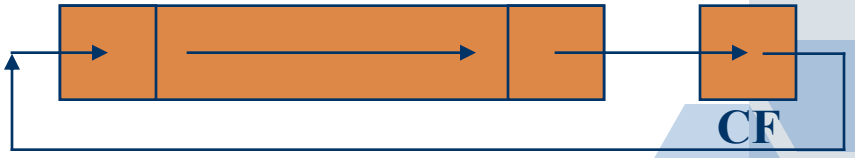
带进位循环左移



循环右移



带进位循环右移



6.3 指令类型

4. 程序控制类指令

- 无条件转移指令：无条件转至目的地址处执行，例如JMP 。
- 条件转移指令：条件满足转至目的地址处执行，否则顺序执行，例如JA（≥）、JB（≤）、JC（CF=1）等指令。
- 调用与返回指令：
  - 调用指令CALL用于从当前的程序位置转至子程序的入口；
  - 返回指令RETURN用于子程序执行完后重新返回到原程序的断点。
- 陷阱指令

  陷阱其实是一种意外事故的中断。

6.3 指令类型

5. 堆栈操作指令

6. 输入输出指令：

它完成从外设端口读入一个数据到CPU的寄存器内(IN)，或将数据从CPU的寄存器输出到某外设的端口中（OUT）。

7. 处理器控制指令：

包括等待指令WAIT、停机指令HOT、空操作指令NOP、开中断指令EI等

8. 特权指令:

特权指令只能给操作系统或其他系统软件，而不能提供给用户使用，以防止破坏系统或其他用户信息。

6.4 指令系统的设计技术

一 指令系统的要求

二 CISC及其特点

三 RISC及其特点

四 指令系统举例

一、指令系统的要求

- **完备性**：指令系统直接提供的指令足够使用，而不必用软件来实现。
- **有效性**：是指利用该指令系统所编写的程序能够高效地运行。程序占据存储空间小、执行速度快。
- **规整性**：
  - 对称性：所有的指令都可使用各种寻址方式；
  - 匀齐性：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：指令长度和数据长度有一定的关系，以方便处理和存取。
- **兼容性**：“向上兼容”，即低档机上运行的软件可以在高档机上运行。

二、CISC及其特点

- “复杂指令系统计算机”，简称CISC（Complex Instruction Set Computer）
  - 指令格式不固定，寻址方式丰富，功能复杂
  - 一些比较简单的指令，在程序中仅占指令系统中指令总数的20%，但出现的频率却占80%；占指令总数20%的最复杂的指令，却占用了控制存储器容量的80%，且使用频率却不高。

二、CISC及其特点

- 早期CISC指令系统的主要特点是：
  - 指令系统复杂。具体表现为指令条数多、寻址方式多、指令格式多。指令串行执行，大多数指令需要多个时钟周期完成。
  - 采用微程序控制，因为微程序控制器适合于实现CISC指令执行过程的控制。
  - 有较多的专用寄存器，大部分运算所需的数据均需访问存储器获取。
  - 编译程序难以用优化措施生成高效的目标代码程序。

二、CISC及其特点

2. CISC主要在以下方面来对增强指令的功能

- 面向目标程序增强指令功能具体方法有：

  ① 提高运算类指令的功能

  ② 提高传送类指令的功能

  ③ 增强程序控制指令功能

- 面向编译程序目标代码生成优化的改进
- 提供面向操作系统优化的指令

三、RISC指令及其特点

- 精简指令系统计算机（Reduced Instruction Set Computer，简称RISC）
- RISC体系结构的芯片经历了三代：
  - 第一代以32位数据通路为代表，支持Cache,软件支持较少，性能与CISC体系结构的产品相当，如RISC Ⅰ、MIPS、IBM801等。
  - 第二代产品提高了集成度，增加了对多处理机系统的支持，提高了时钟频率，建立了完善的存储管理体系，软件支持系统也逐渐完善。它们已具有单指令流水线，可同时执行多条指令。
  - 第三代RISC产品为64位微处理器，采用了超级流水线技术和超标量技术，提高了指令级的并行处理能力，使RISC处理器的整体性能更好。如MIPS的R4000处理器。

三、RISC指令及其特点

- 大部分RISC机具有以下特点：
  - （1） 指令系统设计时选择一些使用频率较高的简单指令，且选择一些很有用但不复杂的指令。
  - （2） 指令长度固定，指令格式种类少，寻址方式种类少。
  - （3） 只有取数/存数指令访问存储器，其余指令的操作都在寄存器之间进行。

三、RISC指令及其特点

- （4） 采用流水线技术。超级标量及超级流水线技术，增加了指令执行的并行度，使得一条指令的平均指令执行时间小于一个机器周期。
- （5） CPU中通用寄存器数量相当多，可以减少访存次数。
- （6） 以硬布线控制逻辑为主，不用或少用微码控制。
- （7） 采用优化的编译程序，力求有效地支持高级语言程序。

三、RISC指令及其特点

同CISC比较，RISC的优点

- 可以充分利用VLSI芯片面积
- 可以提高计算机运算速度

① 指令数、寻址方式和指令格式的种类都较少，且指令的编码很有规律，使指令译码加快。

② 在简化指令的情况下，硬布线连接比微程序控制的延迟小，可缩短CPU的周期。

③ CPU的通用寄存器多，减少了访存次数，加快了速度。

三、RISC指令及其特点

④大部分指令能在一个周期内完成，特别适合于流水线工作。

⑤有的RISC机采用寄存器窗口重叠技术，程序嵌套时不必将寄存器内容保存到存储器中，加快了速度。

- 设计容易，可降低成本，提高可靠性。

三、RISC指令及其特点

- 能有效支持高级语言程序
  - RISC靠编译程序的优化来支持高级语言程序。
  - 指令少，寻址方式少，反而使编译程序容易选择更有效的指令和寻址方式。
  - 通用寄存器多，可尽量安排快速的寄存器操作，使编译程序的代码优化效率较高。
  - 有的RISC机采用寄存器窗口重叠技术，使过程间的参数传送快，且不必保存与恢复现场，因而能直接支持调用子程序和过程的高级语言程序。
  - 在编译时尽量做好程序优化工作，而减少程序执行时间。

五、指令系统举例

.1、Pentium指令系统

2、模型机上8位字长的指令系统设计

1、Pentium 指令系统

- 指令类型
  - 算术逻辑操作指令
  - 串操作/转移控制指令
  - 标志控制/高级语言支持指令
  - 数据传送指令
  - 系统控制/段寄存器操作指令
  - 保护/CACHE管理指令
- Pentium 的指令格式

1、Pentium 指令系统

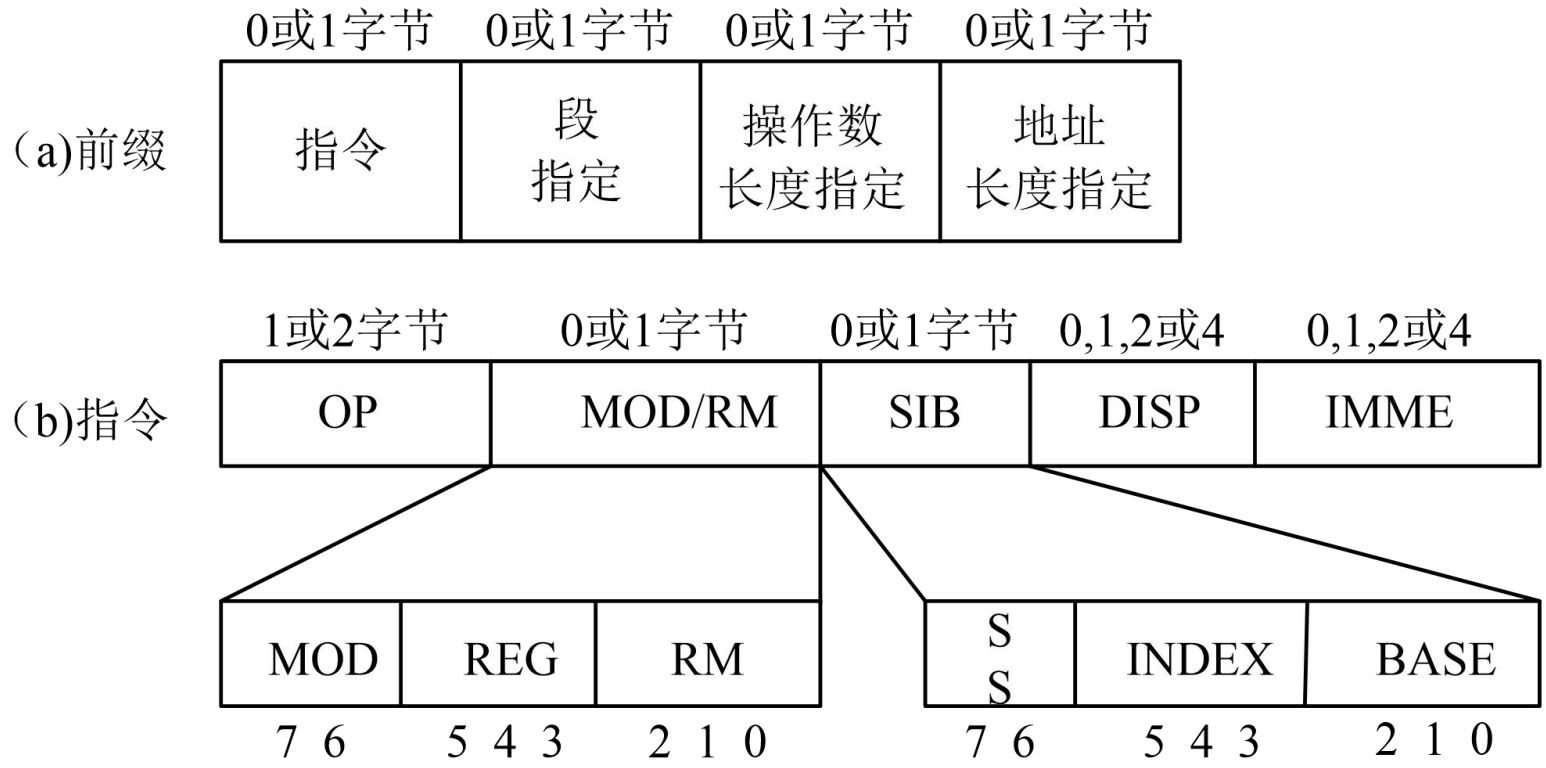


图6－15　Pentium 指令格式

主要由两部分组成：指令前缀，指令本身。

指令前缀为可选。

1、Pentium 指令系统

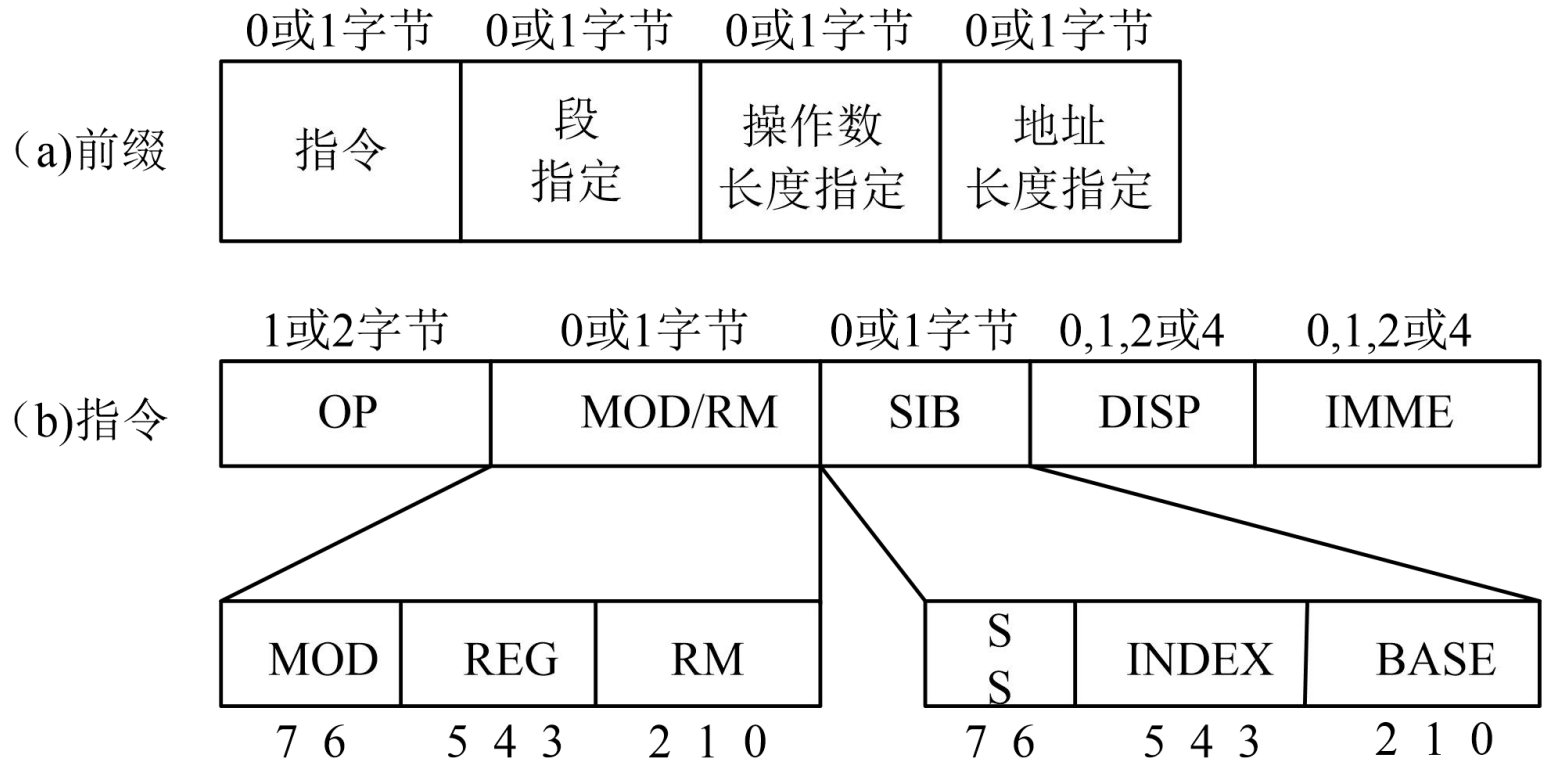


图6－15　Pentium 指令格式

主要由两部分组成：指令前缀，指令本身。

指令前缀为可选。

1、Pentium 指令系统

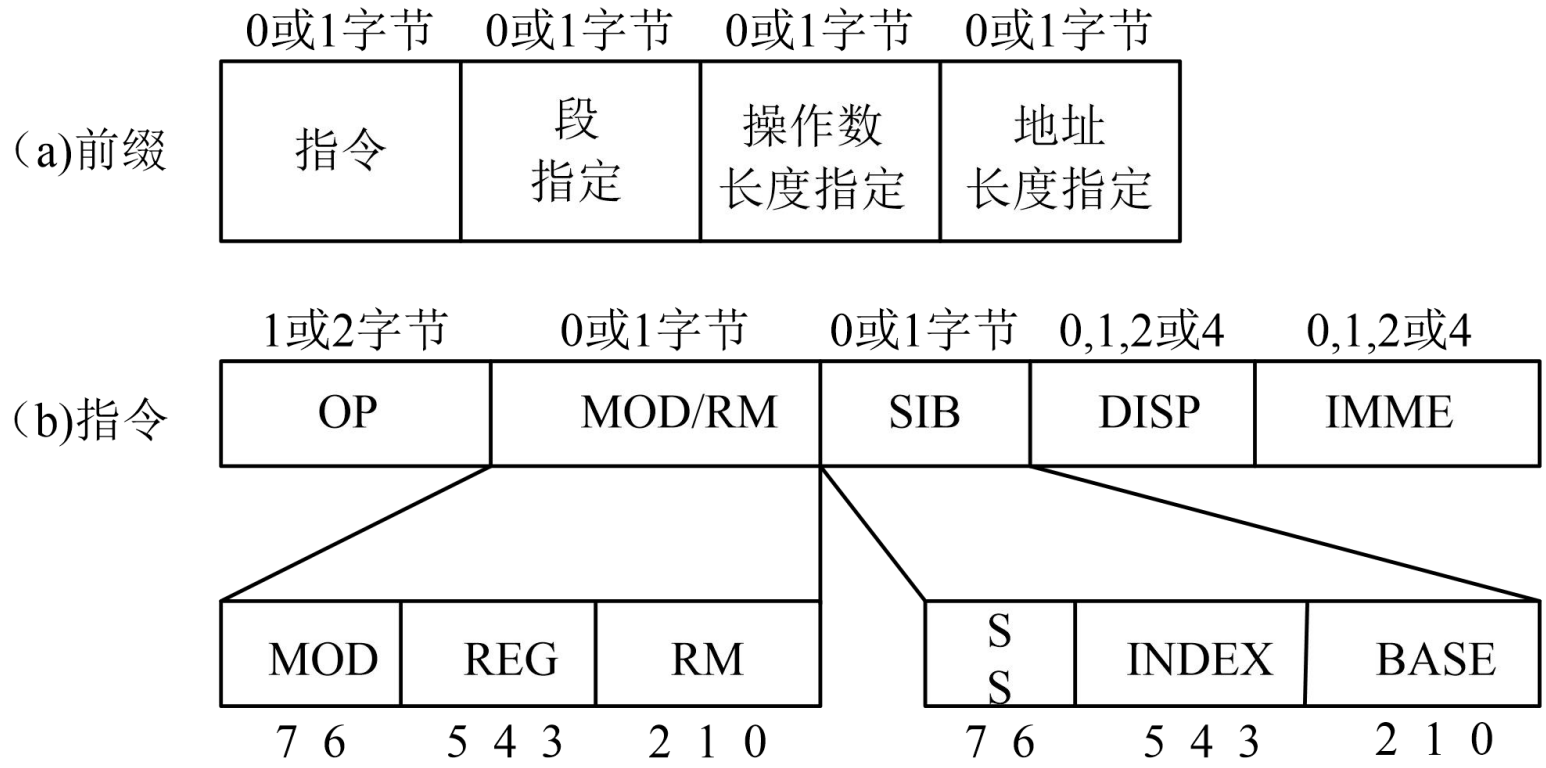


图6－15　Pentium 指令格式

主要由两部分组成：指令前缀，指令本身。

指令前缀为可选。

2、模型机上8位字长的指令系统设计

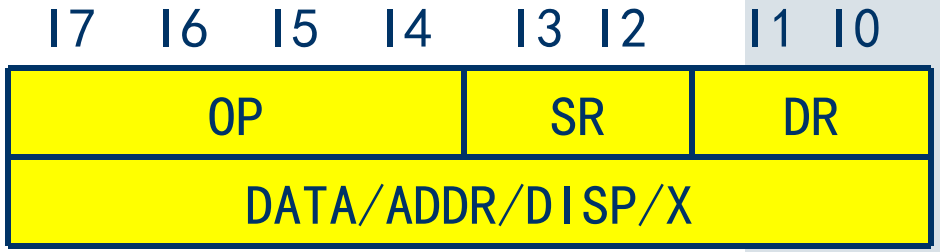
① 模型机指令格式

② 模型机寻址方式

③ 模型机指令系统设计

① 模型机指令格式

- 格式1：一般指令格式



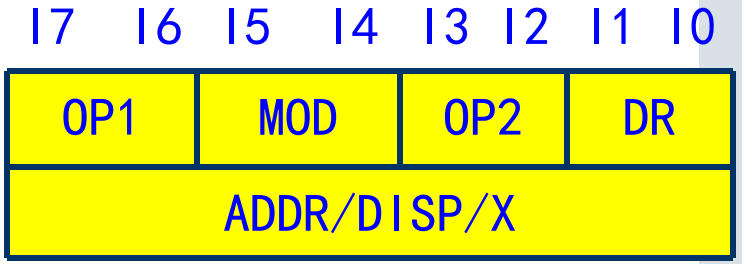
OP：指令操作码，4位，用于对12条机器指令进行编码，是识别指令的标志。

SR：源寄存器号，2位，用于对4个通用寄存器R0、R1、R2、R3的选择，其内容送总线，作为源操作数之一。

DR：目的寄存器号，2位，用于对4个通用寄存器R0、R1、R2、R3的选择，其内容可送总线，也可以从总线上接收数据，通常作为目的操作数。

DATA/ADDR/DISP/X：指令的第二个字，可有可无，其含义也可以由用户自定义，可以是立即数，可以是直接/间接地址，也可以是其它寻址方式用到的地址信息，如相对偏移量、形式地址等等。

格式2：带寻址方式码的指令格式



OP1：第一指令操作码，
2位，是带寻址方式码
的指令（4条）的特征位。

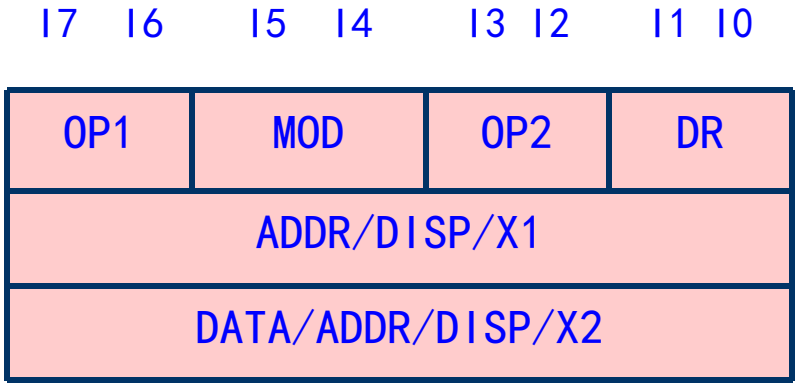
MOD：寻址方式码，2位，用于对4种寻址方式的编码，至于4种寻址方式的定义，可以自行设计，通常为直接、间接、变址、相对寻址。

OP2：第二指令操作码，2位，是4条带寻址方式码的指令本身的编码。

DR：目的寄存器号，同格式一。

ADDR/DISP/X：指令的第二个字，为寻址方式中所用到的直接/间接地址ADDR，或者是相对寻址的偏移量DISP，或者是变址寻址的形式地址X

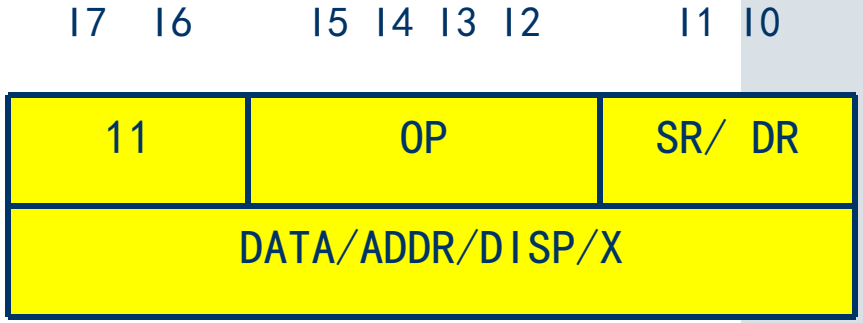
格式3：三字指令



- **指令包含三字：**
  - 指令第一字：包含操作码、寻址方式、寄存器号
  - 指令第二,三字：为寻址方式中所用到的直接/间接地址ADDR，或者是相对寻址的偏移量DISP，或者是变址寻址的形式地址X，也可以是立即数DATA
- **双存储器操作数的指令：**既指令的两个操作数均在存储器内。其余同格式2。

格式4：操作码扩展指令格式

OP—指令操作码，4位，是单寄存器地址指令（16条）的操作码，可通过I_7 I_6为11方式实现散转。



SR/DR —源/目的寄存器，2位，含义同上。

ADDR/DISP/X—指令的第二个字，为寻址方式中所用到的立即数DATA、直接/间接地址ADDR，或者是相对寻址的偏移量DISP，或者是变址寻址的形式地址X。

② 模型机寻址方式

- 模型机的指令系统可实现7种基本寻址方式：寄存器直接、寄存器间接、直接、间接、相对、变址、立即数。
- 对于其中相对复杂的寻址方式（直接、间接、相对、变址），可以由指令中的MOD字段来定义。
- 简单的寻址方式可以直接由指令操作码指定。
- 注意：任何一种寻址方式，均可以直接由指令操作码隐含指定。
- 用户也可以根据需要，自行设计一些特殊的寻址方式，例如相对SR的偏移量寻址方法，即EA=（SR）+ADDR。

带寻址方式MOD的指令格式（格式2）

- 对于指令格式2，假设定义：
  - MOD=00：直接寻址，则有效地址EA=ADDR，操作数=（ADDR）；
  - MOD=01：间接寻址，则有效地址EA=（ADDR），操作数=（（ADDR））；
  - MOD=10：变址寻址，则有效地址EA=（SI）+X，操作数=（（SI）+X）；其中SI为变址寄存器，隐含为R2；
  - MOD=11：相对寻址，则有效地址EA=（PC）+DISP，操作数=（（PC）+DISP）；

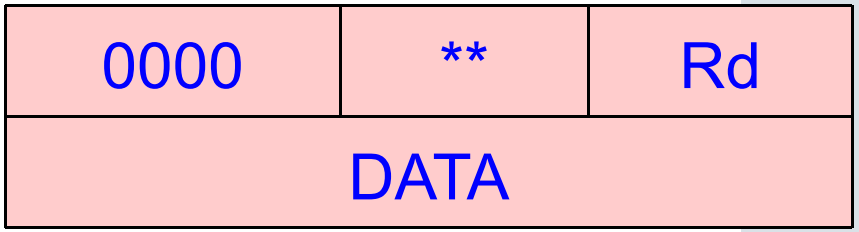
③ 模型机指令系统设计

- 指令设计原则
  - 指令的格式必须按照规定的格式设计，即操作码OP、源寄存器号SR、目的寄存器号DR必须按格式规定固定长度和位置，若按照格式2设计指令，则操作码OP分为两段（OP1和OP2）。
  - 寻址方式的设计，可以根据需要，或由MOD字段定义，或由操作码隐含指定。
  - 指令类型及功能的设计，只需满足程序设计的要求和需求即可。
  - 指令操作码的分配设计，要注意规整性。
- 模型机指令设计举例1
- 模型机指令设计举例2

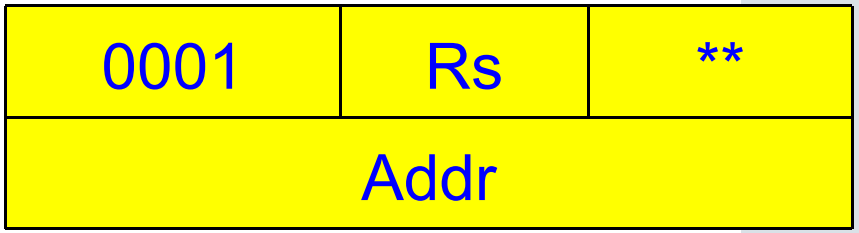
指令系统1举例

- 不用专门的MOD字段指出寻址方式，寻址方式由指令码定义。

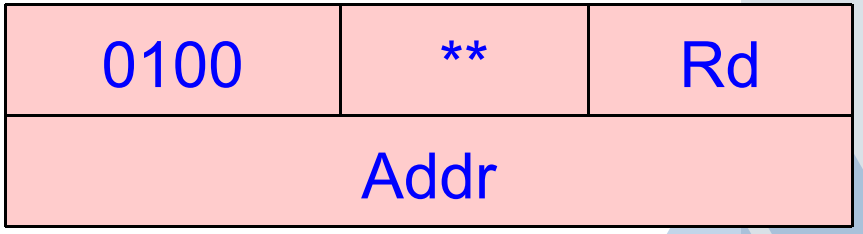
1. MOV1 #DATA , Rd ;
   DATA→Rd



2. MOV2 Rs, [Addr];
   (Rs)→Addr



3. ADD [Addr], Rd;
   (Addr)+(Rd)→Rd



返回

4. IN Rd,[Addr];
(Port Addr)→Rd

11	0000	Rd
Port Addr		

5. OUT [Addr],Rs;
(Rs)→ Port Addr

11	0001	Rs
Port Addr		

6. Jmp [Addr];
(Addr)→ PC

11	0010	**
Addr		

7. HLT

11	0011	**

8. DEC Rd；(Rd)-1 →Rd

11	0100	Rd

9. INC Rd；(Rd)+1 →Rd

11	0101	Rd

10. JZ Addr；FZ=1,则
Addr→PC; 否则结束

11	0110	**
Addr		

返回

练习

1. **MOV1 #DATA , Rd ; DATA→Rd**

0000	**	Rd
DATA		

2. **MOV2 Rs, [Addr]; (Rs)→Addr**

0001	Rs	**
Addr		

- 请写出（1）MOV1 #04H,R1 和 （2）MOV2 R1,[11H]的功能和机器指令
- （1）功能：04H→R1

  机器指令：00000001，00000100
- （2）功能：(R1)→11H

  机器指令：00010100，00010001

练习

3. ADD [Addr], Rd;
(Addr)+(Rd)→Rd

0100	**	Rd
Addr		

- 请ADD [10H],R1对应的功能和机器指令

  功能：(10H)+(R1)→R1

  指令：00100001，00010000

练习

4. IN Rd,[Addr];
   (Port Addr)→Rd

11	0000	Rd
Port Addr		

5. OUT [Addr],Rs;
(Rs)→ Port Addr

11	0001	Rs
Port Addr		

- **请写出（1）IN [01H],R1和（2）OUT R1,[02H]对应的功能和机器指令**
- （1）功能：(Port Addr)→R1
  机器指令： 11000001， 00000001
- （2）功能：(R1)→Port Addr
  机器指令： 11000101， 00000010

练习

6. Jmp [Addr];
   (Addr)→ PC

11	0010	**
Addr		

7. HLT

11	0011	**

❖请写出（1）JMP [11H]和（2）HLT对应的功能和机器指令。

❖（1）功能：(11H)→PC

机器指令： 11001000， 00010001

❖（2）功能：停止

机器指令：11001100

程序	功能	汇编结果 （M地址：机器指令）
MOV1 #04H,R1		
MOV2 R1,[11H]		
IN [01H],R1		
MOV2 R1,[10H]		
IN [01H],R1		
ADD [10H],R1		
OUT R1,[02H]		
JMP [11H]		

程序	功能	汇编结果 （M地址：机器指令）
		00H:00000001
		01H:00000100
		02H:00010100
		03H:00010001
		04H:11000001
		05H:00000001
		06H:00010100
		07H:00010000
		08H:11000001
		09H:00000001
		0AH:00100001
		0BH:00010000
		0CH:11000101
		0DH:00000010
		0EH:11001000
		0FH:00010001

指令系统2举例

共有12条指令，分为：

- 5条双寄存器算术逻辑运算类指令
- 3条单寄存器指令
- 4条存储器访问类指令
- 2条I/O指令
- 2条过程控制类指令
- 程序设计

5条双寄存器算术逻辑运算类指令

格式：

操作码及功能：

助记符	操作码OP	功能
MOV DR，SR	0000	(SR)→DR
ADD DR，SR	0001	(SR)+(DR)→DR
SUB DR，SR	0010	(DR)-(SR)→DR
AND DR，SR	0011	(SR)∧(DR)→DR
RRC DR，SR	0100	(SR)进行带进位循环右移→DR

3条单寄存器指令

格式：

操作码及功能：

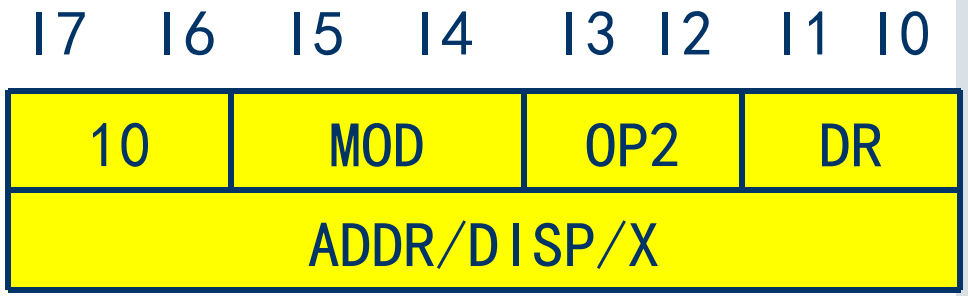
助记符	操作码OP	功能
INC DR	1101	(DR)+1→DR
DEC DR	1110	(DR)-1→DR
CLR DR	1111	0→DR

4条存储器访问类指令

格式：

操作码及功能：

SI隐含为R2

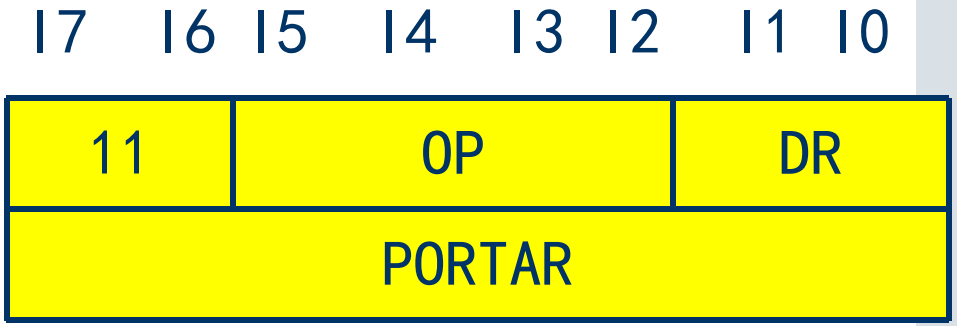


MOD	寻址方式	EA	助记符	OP_2	功能
00	直接寻址	ADDR	LDA	00	[EA]→DR
01	间接寻址	[ADDR]	STA	01	（DR）→EA
10	变址寻址	（SI）+X	JMP	10	EA→PC
11	相对寻址	（PC）+DISP	JZC	11	若FC+FZ=1，则EA→PC，否则，结束指令

2条I/O指令

格式：

操作码及功能：

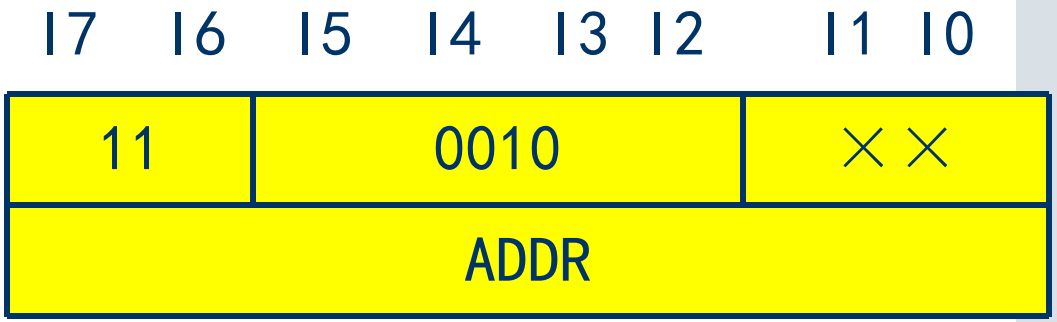


助记符	操作码OP	功能
IN DR, [PORTAR]	0000	(PORTAR)→DR
OUT DR, [PORTAR]	0001	(DR)→ PORTAR

2条过程控制类指令

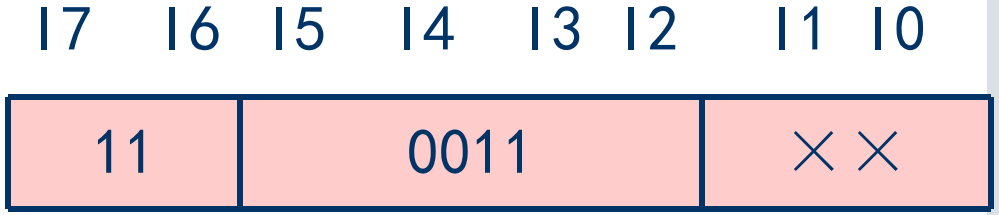
格式：

- CALL ADDR
- (PC) →(SP)，
- (SP)－1→SP，ADDR→PC



格式：

- RET
- (SP)＋1→SP，((SP))→PC



地址	机器码	助记符	备注
00H	?	CLR R_0	R_0当作累加器。
01H 02H	? ?	LDA R_2，[2BH] 直接地址2BH	R_2当作计数器/变址寄存器;其初值0AH存放在单元2BH中。
03H 04H	? ?	L1：LDA R_1， [SI+1FH]	取出需要累加的数据;采用变址寻址方式; 第1次地址=29H。
05H	?	ADD R_0，R_1	累加。
06H	?	DEC R_2	计数器递减；并影响标志FZ、FC
07H 08H	? ?	JZC L2 相对位移03H	FC+FZ=1循环，FC+FZ=0（无借位不为0）退出循环。
09H 0AH	? ?	STA [2AH],R_0 直接地址2AH	存储累加和；采用直接寻址方式。
0BH 0CH	? ?	JMP L1 直接地址03H	无条件转移；采用直接寻址方式。
0DH 0EH	?	L2: JMP [00H]	转移至00H地址所保存的指令地址。

地址	机器码	助记符	备注
00H	11_1111_00	CLR R_0	R_0当作累加器。
01H 02H	10_00_00_10 0010_1011	LDA R_2，[2BH] 直接地址2BH	R_2当作计数器/变址寄存器;其初值0AH存放在单元2BH中。
03H 04H	10_10_00_01 0001_1111	L1：LDA R_1，[SI+1FH]	取出需要累加的数据;采用变址寻址方式;第1次地址=29H。
05H	0001_01_00	ADD R_0，R_1	累加。
06H	11_1110_10	DEC R_2	计数器递减；并影响标志FZ、FC
07H 08H	10_11_11_00 0000_0011	JZC L2 相对位移03H	FC+FZ=1循环，FC+FZ=0（无借位不为0）退出循环。
09H 0AH	10_00_01_00 0010_1010	STA [2AH],R_0 直接地址2AH	存储累加和；采用直接寻址方式。
0BH 0CH	10_00_10_00 0000_0011	JMP L1 直接地址03H	无条件转移；采用直接寻址方式。
0DH 0EH	10_01_10_00 0000_0000	L2: JMP [00H]	转移至00H地址所保存的指令地址。

地址	机器码	助记符	备注
…	……		
20H	?	N1	数据1
21H	?	N2	数据2
…	……		
29H	?	N10	数据10
2AH	?	N1+N2+……+N10	累加和
2BH	?	计数值0AH	

作业：请仿照前例，写出例2给出的程序中每条机器指令对应的二进制代码（用存储器地址：机器语言程序的形式写出来）

第六章作业

- 作业： 1，2，4，10，12，16
- 可通过拍摄解题过程的照片提交。
- 限本章结束后一周内提交，过期无效

End