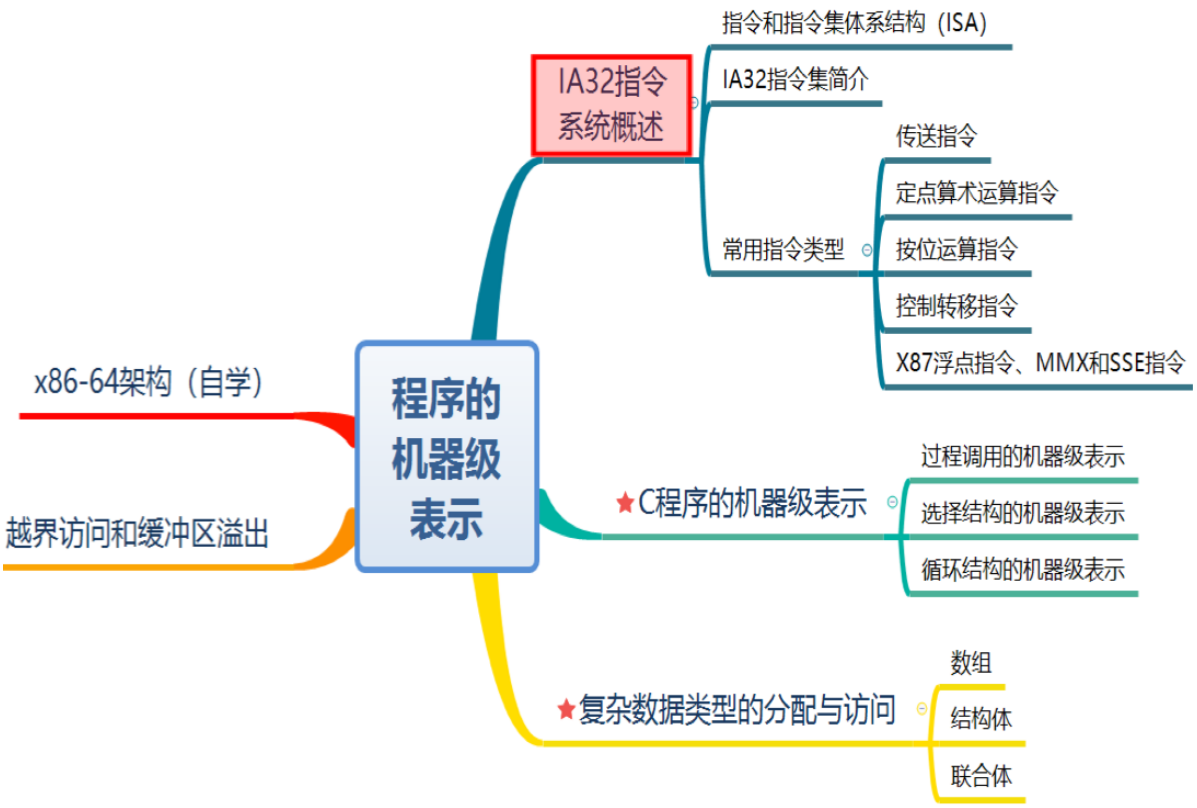
No.3 程序的转换与机器级表示



(一) IA32指令

- 微指令：微程序级命令，硬件范畴
- 伪指令：机器指令构成的指令序列，软件范畴
- 机器指令：以上二者之间，本章内容

1、汇编指令是机器指令的汇编表示形式，即符号表示

2、机器指令和汇编指令一一对应，都属于机器级指令

(二)

指令格式设计原则

- 操作码（操作性质）
  - 操作码的编码必须有唯一的解释
  - 要有足够的操作码位数
- 操作数（操作对象）
  - 源操作数1 或/和 2（立即数、寄存器编号、存储地址）
  - 目的操作数地址 （寄存器编号、存储地址）
  - 合理选择地址字段的个数
- 其他原则
  - 指令应尽量短。
  - 指令长度应是字节的整倍数。
  - 指令应尽量规整。

指令集体系结构（ISA）

- ISA是一种规约（Specification）
  - 可执行的指令集合，包括指令格式、操作种类以及每种操作对应的操作数的相应规定（寻址方式、数据宽度等）
  - 操作数所能存放的寄存器组的结构、存储空间的大小、编址方式
  - 操作数在存储空间的存放方式（大端or小端）
  - 指令执行过程的控制方式，包括程序计数器、条件码定义等
- ISA设计的好坏直接决定计算机的性能和成本
- 指令系统是ISA中最核心的部分

(四) 数据类型

支持的数据类型

C语言声明	Intel操作数类型	汇编指令长度后缀	存储长度（位）
(unsigned) char	整数 / 字节	b	8
(unsigned) short	整数 / 字	w	16
(unsigned) int	整数 / 双字	l	32
(unsigned) long int	整数 / 双字	l	32
(unsigned) long long int	-	-	2×32
char *	整数 / 双字	l	32
float	单精度浮点数	s	32
double	双精度浮点数	l	64
long double	扩展精度浮点数	t	80 / 96

(五) 寄存器组织

1、8个通用寄存器（编号占3位）

2、2个专用寄存器

3、6个段寄存器（只有16位）

(六) 寻址方式

- **操作数所在的位置**
  - 指令中：立即寻址（立即数）
  - 寄存器中：寄存器寻址
  - 存储单元中（属于存储器操作数，按字节编址）：其他寻址方式

实地址模式 20位 1MB

保护模式（正使用）32 4G

保护模式下的寻址方式

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

EIP+位移

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 段内偏移

0

存储器操作数的寻址方式

```
int x;
float a[100];
short b[4][4];
char c;
double d[10];
```

比例因子 int为4 sizeof()

各变量应采用什么寻址方式?

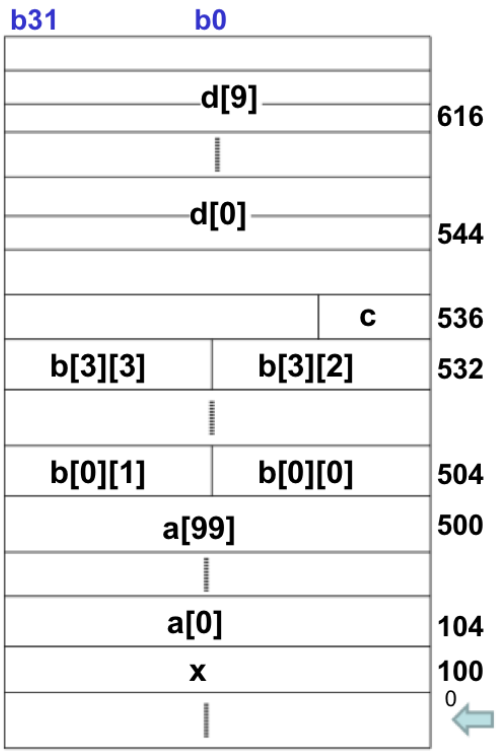
x、c：位移 / 基址
a[i]：104+i×4，比例变址+位移
d[i]：544+i×8，比例变址+位移
b[i][j]： 504+i×8+j×2，
基址+比例变址+位移

将b[i][j]取到AX中的指令可以是：

“movw 504(%ebp,%esi,2), %ax”

其中， i×8在EBP中，j在ESI中，

有效地址（段内偏移）



0

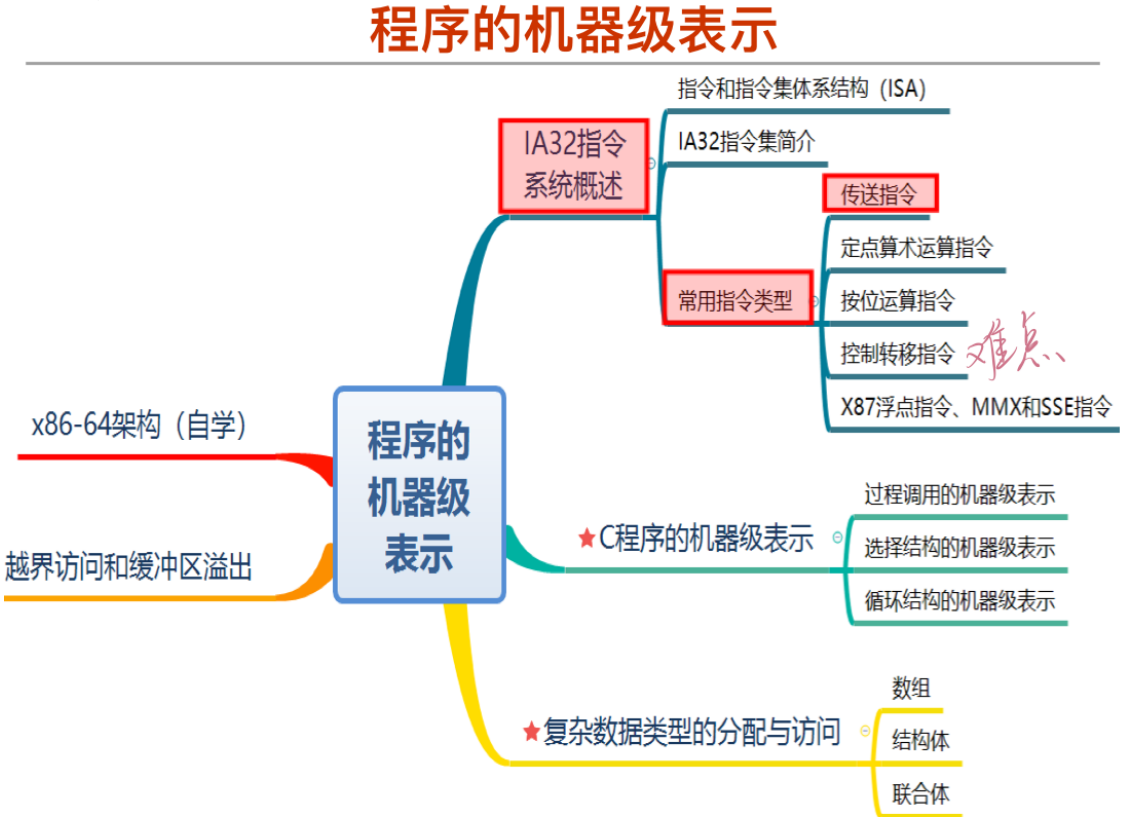
(4). 指令表示.(操作码、寻址方式、寄存器编号)

1. { Intel格式
   { AT&T格式(课本采用)

2、 movb %cl, -6(%bx, %di)

表示 M[R(bx)+ R(di)-6] ← R[cl]

常用指令类型

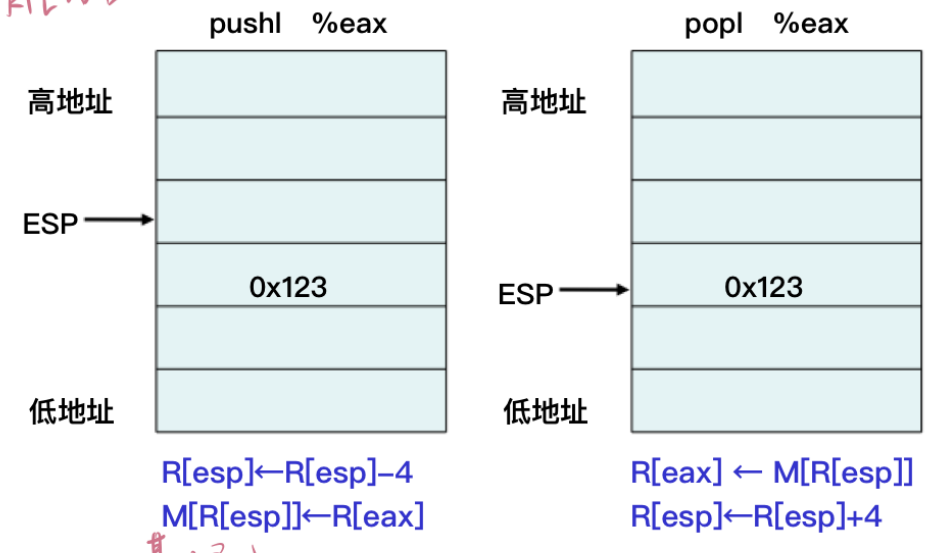


传送指令

- ① 通用数据传送指令 ——寄存器之间或寄存器和存储器之间交换数据
  MOV：一般传送，包括movb、movw和movl等
  MOVS：符号扩展传送，如movsbw、movswl等
  MOVZ：零扩展传送，如movzwl、movzbl等
  XCHG：数据交换
  PUSH/POP：pushl,popl，入栈/出栈
- ② 地址传送指令
  LEA：加载有效地址，如leal (%edx,%eax), %eax”的功能为
  R[eax]←R[edx]+R[eax]，执行前，若R[edx]=*i*，R[eax]=*j*，则指令执行后，R[eax]=*i*+*j*
- ③ 输入输出指令
  IN和OUT
  ——I/O端口与寄存器之间交换数据
- ④ 标志传送指令
  PUSHF、POPF：将EFLAG压栈，或将栈顶内容送EFLAG
  ——标志寄存器和栈存储区之间交换数据

“入栈”和“出栈”操作

- 利用栈（Stack）“先进后出”的特点，可以实现过程调用机制
- 注意：栈存储区是从高地址向低地址增长的



传送指令举例

阅读以下AT&T格式汇编指令，请用RTL语言描述每条指令的功能。

```
pushl    %ebp
                    //R[esp]←R[esp]-4，M[R[esp]] ←R[ebp]，双字
movl     %esp, %ebp
                    //R[ebp] ←R[esp]，双字
movl    8(%ebp), %edx
                    //R[edx] ←M[R[ebp]+8]，双字
movb    $255, %bl
                    //R[bl]←255，字节
movw    8(%ebp,%edx,4), %ax
                    //R[ax]←M[R[ebp]+R[edx]×4+8]，字
movw    %dx, 20(%ebp)
                    //M[R[ebp]+20]←R[dx]，字
leal    8(%ecx,%edx,4), %eax
                    //R[eax]←R[ecx]+R[edx]×4+8，双字
```