P123-127

- 数据传送指令将数据从原位置复制到目的位置，不作任何变化。

MOV 类指令： XXX S，D

- movb：传送字节
- movw: 传送字
- movl: 传送双字
- movq ：传送四字
- movabsq： 传送绝对四字

其他扩展指令：

指令	效果	描述
MOVZ S，R	R←零扩展(S)	以零扩展进行传送
movzbw		将做了零扩展的字节传送到字
movzbl		将做了零扩展的字节传送到双字
movzwl		将做了零扩展的字传送到双字
movzbq		将做了零扩展的字节传送到四字
movzwq		将做了零扩展的字传送到四字

图 3-5 零扩展数据传送指令。这些指令以寄存器或内存地址作为源，以寄存器作为目的

指令	效果	描述
MOVS S，R	R←符号扩展(S)	传送符号扩展的字节
movsbw		将做了符号扩展的字节传送到字
movsbl		将做了符号扩展的字节传送到双字
movswl		将做了符号扩展的字传送到双字
movsbq		将做了符号扩展的字节传送到四字
movswq		将做了符号扩展的字传送到四字
movslq		将做了符号扩展的双字传送到四字
cltq	%rax ←符号扩展(%eax)	把%eax 符号扩展到%rax

图 3-6 符号扩展数据传送指令。MOVS 指令以寄存器或内存地址作为源，以寄存器作为目的。cltq 指令只作用于寄存器%eax 和%rax

原操作数 s 指定的是一个立即数，存储在寄存器或者内存中。

目的操作数 d 指定一个位置。

因此将一个值从内存位置复制到另一个内存位置需要两条指令：先加载到寄存器，再写入目的位置。

数据传送示例：

```
long exchange(long *xp, long y)
{
    long x = *xp;
    *xp = y;
    return x;
}
```

a）C语言代码

```
    long exchange(long *xp, long y)
    xp in %rdi, y in %rsi
1   exchange:
2     movq    (%rdi), %rax    Get x at xp. Set as return value.
3     movq    %rsi, (%rdi)    Store y at xp.
4     ret                     Return.
```

b）汇编代码

图 3-7　exchange 函数的 C 语言和汇编代码。寄存器%rdi 和%rsi 分别存放参数 xp 和 y

第 2 行：从内存中读值到寄存器

第 3 行：从寄存器中写入内存。

c 语言中*所谓的指针其实就是地址，引用指针就是将该指针放入寄存器，然后在内存引用中使用这个寄存器。局部变量通常也就是存放在寄存器中，因为访问速度快。

- 压入和弹出栈数据
  - pushq S
  - popq D