程序的运行和汇编语言

计算机运行时，程序会显示为一整串二进制的机器语言，CPU 则会根据传入的语句进行断句和执行。由于机器语言是纯数字，我们通常会将其直接翻译成汇编语言加以理解。和其它高级语言不同，汇编语言反映了 CPU 执行程序的真实步骤和状态。

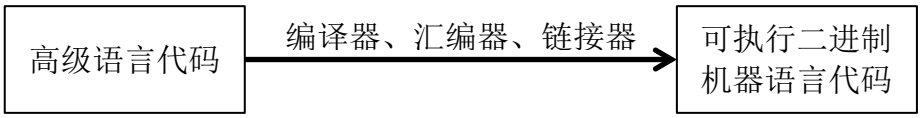

图 1 — 高级语言代码生成可执行代码

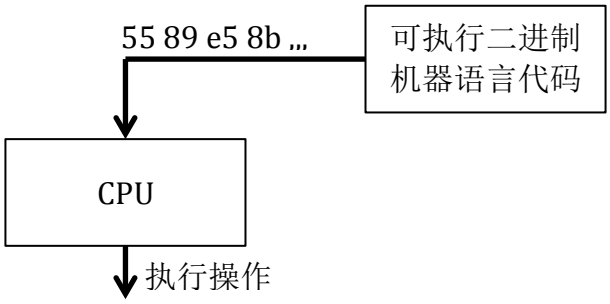

图 2 — CPU 执行程序

常用寄存器

通常情况下，常用的寄存器有%eax, %ebx, %ecx, %edx, %esi, %edi, %ebp, %esp 等等。前面 6 个寄存器为通用寄存器，可以用来保存数据，而后两个则指向了程序运行时比较重要的地址，如%ebp 指向当前运行的帧的地址，而%esp 则指向了当前所用的栈的地址。

程序运行实例

C 语言代码：

```
int simple (int * xp, int y) {
    int t = *xy + y;
    *xp = t;
    return t;
}
```

需要说明的汇编命令：

pushl, popl x: 推入栈和和弹出栈，以先入后出的顺序保存和拿回数据，l 意为后续参数为 32 字节

movl s, d: 将 d 地址的值修改为 s 地址上的值

addl s, d: 将 s 地址上的值和 d 地址上的值相加，并保存值 d 地址中
ret: 返回原函数
n(x): 括号意为取 x 地址上的值，括号前的 n 意为将后面的值加 n

汇编代码：

```
pushl %ebp              ; 进入方程，储存从前的帧地址
movl %esp, %ebp         ; 将帧指针移动到栈的位置
movl 8(%ebp), %eax      ; 将帧指针后 8 个字节的位置的值(xp)载入寄存器%eax
中
movl (%eax), %edx       ; 将%eax 的值(*xp)载入寄存器%edx
addl 12(%ebp), %edx     ; 将帧指针后 12 个字节的位置的值(y)与%edx 记录的值
                        ; (*xp)相加，并载入寄存器%edx 中
movl %edx, (%eax)       ; 将%edx 的值写入寄存器(%eax)记录的地址(xp)中，
                        ; 即修改*xp 的值
movl %edx, %eax         ; 将%edx 的值写入寄存器%eax 中（作为返回值）
movl %ebp, %esp         ; 将栈指针移动到现在的帧指针的位置上，即原来的栈
                        ; 指针的位置
popl %ebp               ; 将之前的帧地址重新载入
ret                     ; 返回原来的帧，重新运行
```

可知，汇编代码完成的是 C 语言代码要求的操作，而它是计算机运行时真实的运行步骤。汇编代码由 C 语言编译而来，平时一般储存在硬盘中，运行时会加载至内存，由 CPU 运行。