- 整数寄存器文件包含16个命名的位置，分别存储64位的值。这些寄存器可以存储地址(对应于C语言的指针)或整数数据。有的寄存器被用来记录某些重要的程序状态，而其他的寄存器用来保存临时数据，例如过程的参数和局部变量，以及函数的返回值。
- 条件码寄存器保存着最近执行的算术或逻辑指令的状态信息。它们用来实现控制或数据流中的条件变化，比如说用来实现 if 和 while 语句。
- 一组向量寄存器可以存放一个或多个整数或浮点数值。

虽然C语言提供了一种模型，可以在内存中声明和分配各种数据类型的对象，但是机器代码只是简单地将内存看成一个很大的、按字节寻址的数组。C语言中的聚合数据类型，例如数组和结构，在机器代码中用一组连续的字节来表示。即使是对标量数据类型，汇编代码也不区分有符号或无符号整数，不区分各种类型的指针，甚至于不区分指针和整数。

程序内存包含：程序的可执行机器代码，操作系统需要的一些信息，用来管理过程调用和返回的运行时栈，以及用户分配的内存块(比如说用 malloc 库函数分配的)。正如前面提到的，程序内存用虚拟地址来寻址。在任意给定的时刻，只有有限的一部分虚拟地址被认为是合法的。例如，x86-64 的虚拟地址是由 64 位的字来表示的。在目前的实现中，这些地址的高 16 位必须设置为 0，所以一个地址实际上能够指定的是 2^{48} 或 64TB 范围内的一个字节。较为典型的程序只会访问几兆字节或几千兆字节的数据。操作系统负责管理虚拟地址空间，将虚拟地址翻译成实际处理器内存中的物理地址。

一条机器指令只执行一个非常基本的操作。例如，将存放在寄存器中的两个数字相加，在存储器和寄存器之间传送数据，或是条件分支转移到新的指令地址。编译器必须产生这些指令的序列，从而实现(像算术表达式求值、循环或过程调用和返回这样的)程序结构。

旁注　不断变化的生成代码的格式

在本书的表述中，我们给出的代码是由特定版本的 GCC 在特定的命令行选项设置下产生的。如果你在自己的机器上编译代码，很有可能用到其他的编译器或者不同版本的 GCC，因而会产生不同的代码。支持 GCC 的开源社区一直在修改代码产生器，试图根据微处理器制造商提供的不断变化的代码规则，产生更有效的代码。

本书示例的目标是展示如何查看汇编代码，并将它反向映射到高级编程语言中的结构。你需要将这些技术应用到你的特定的编译器产生的代码格式上。

3.2.2　代码示例

假设我们写了一个C语言代码文件 mstore.c，包含如下的函数定义：

```
long mult2(long, long);

void multstore(long x, long y, long *dest) {
    long t = mult2(x, y);
    *dest = t;
}
```

在命令行上使用“-S”选项，就能看到C语言编译器产生的汇编代码：

```
linux> gcc -Og -S mstore.c
```

这会使 GCC 运行编译器，产生一个汇编文件 mstore.s，但是不做其他进一步的工作。(通常情况下，它还会继续调用汇编器产生目标代码文件)。