# 实验3 C语言与鲲鹏920处理器汇编语言混合程序设计

本实验要求学生用C语言和鲲鹏920处理器汇编语言混合编程完成顺序、分支、循环程序设计。

## 3.1 题目

1. 求数组buf的最大值：数组buf中共有10个带符号字数据，请求出其最大值存入变量max中。要求数组buf的定义和初始化以及变量max的输出均在C程序中完成，鲲鹏汇编程序完成求最大值的过程。
2. 求数组buf的最小值：数组buf中共有10个带符号字数据，请求出其最小值存入变量min中。要求数组buf的定义和初始化以及变量min的输出均在C程序中完成，鲲鹏汇编程序完成求最小值的过程。
3. 求数组buf中负数、正数、零的个数：数组buf中共有10个带符号字数据，请分别求出负数、正数、零的个数，存入变量negtive、positive、zero中。数组buf的定义和初始化以及变量negtive、positive、zero的输出均在C程序中完成，鲲鹏汇编程序完成负数、正数、零的个数统计过程。
4. 在字符数组buf中查找一个特定字符：字符数组buf中共有若干个字符，请查找其中是否有字符“a”，若有则将其下标存入变量location中，若没有，则输出字符串“There is no that character.” 要求数组buf的定义和初始化、字符串的定义和输出、以及变量location的输出均在C程序中完成，鲲鹏汇编程序完成字符“a”的查找。
5. 求数组buf中数据元素的平均值：数组buf中共有10个无符号字数据，请求出其平均值存入变量aver中。数组buf的定义和初始化以及变量aver的输出均在C程序中完成，鲲鹏汇编程序完成求平均值的过程。

## 3.2 要求

1. 每位同学从上述5道题中任意选择一道题完成，要求分别用C语言程序调用鲲鹏汇编程序、C语言程序内嵌鲲鹏汇编语言程序段两种方式来实现。
2. 每位同学将C语言源程序、鲲鹏920汇编源程序、程序编译链接调试及执行结果这三部分内容分别截图上传学习通。
3. 每位同学独立完成实验报告。
4. 完成相同题目的同学可组成小组，但每组不得超过3人。每小组提交一份实验过程讲解、演示视频和一份汇报PPT。视频要求完整展示整个程序的编写、编译、链接、执行和调试过程以及程序的执行结果。PPT简要概括上述过程，着重分析混合编程过程及需要注意的事项、实验过程中遇到的问题和原因分析及解决办法、总结与体会等。
5. 互评：
   1. 同一组的几位同学根据实际完成情况分别给其他同学打分，满分为100分。
   2. 视频和PPT上传到QQ群，每组同学分别给其他组打分（客观、公正），满分为100分。

## 3.3 示例

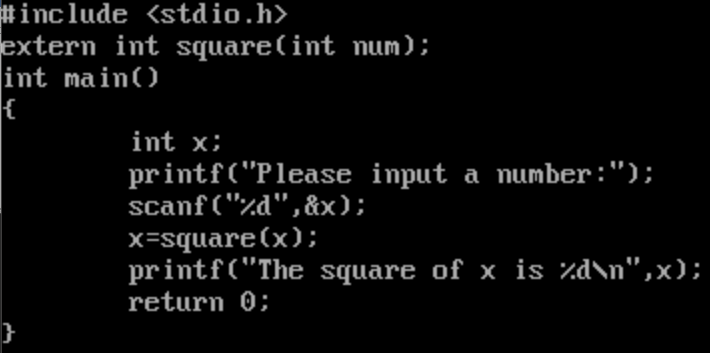
1. 从键盘接收变量x，求出x的平方后输出。
   1. C语言程序调用鲲鹏汇编程序：

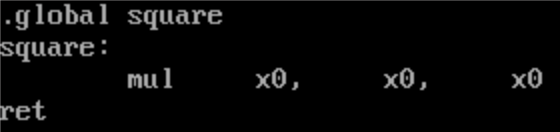
关于C与汇编的混合编程的参数传递，ARM64提供了31个通用寄存器，各自的用途见表3.1。参数传递用到的是x0~x7这8个寄存器，若参数个数大于8个则需要使用堆栈来传递参数。

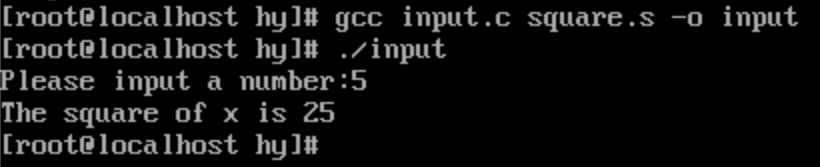
表3.1 ARM64通用寄存器用途

|  |  |
| --- | --- |
| 寄存器 | 用途 |
| x0~x7 | 传递参数和返回值，多余的参数用堆栈传递，64位的返回结果保存在x0中。 |
| X8 | 用于保存子程序的返回地址。 |
| x9~x15 | 临时寄存器，也叫可变寄存器，无需保存。 |
| x16~x17 | 子程序内部调用寄存器，使用时不需要保存，尽量不要使用。 |
| x18 | 平台寄存器，它的使用与平台相关，尽量不要使用。 |
| x19~x28 | 临时寄存器，子程序使用时必须保存。 |
| x29 | 帧指针寄存器（FP），用于连接栈帧，使用时必须保存。 |
| x30 | 链接寄存器（LR），用于保存子程序的返回地址。 |
| x31 | 堆栈指针寄存器（SP），用于指向每个函数的栈顶。 |

程序如下：







请注意：

* 在C程序中务必用“extern”将鲲鹏汇编子程序声明为外部函数；
* 在鲲鹏汇编子程序中务必用“.global”将子程序声明为全局函数。

1. C语言程序内嵌鲲鹏汇编程序：

C语言是无法完全代替汇编语言的，一方面是其效率比C要高，另一方面是某些特殊的指令在C语法中是没有等价的语法的。例如操作某些特殊的CPU寄存器如状态寄存器、操作主板上的某些IO端口或者对性能要求极其苛刻的场景等，我们都可以通过在C中内嵌汇编来满足要求。

在 C 代码中内嵌的汇编语句的基本格式为：

\_\_asm\_\_ \_\_volatile\_\_ (“asm code”

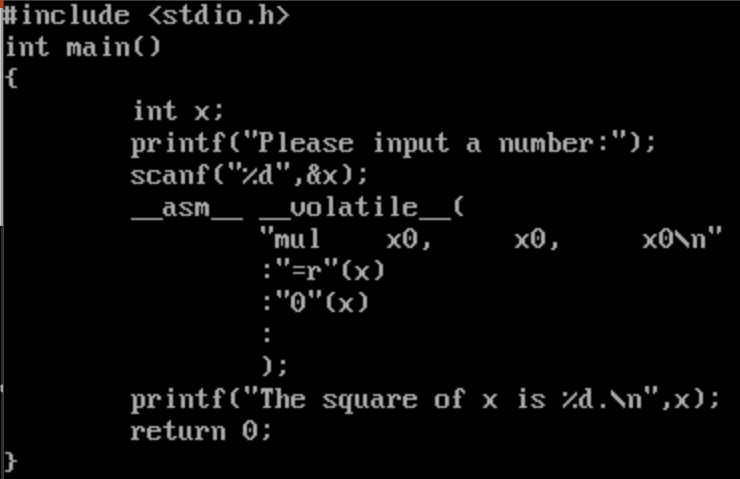
：输出操作数列表

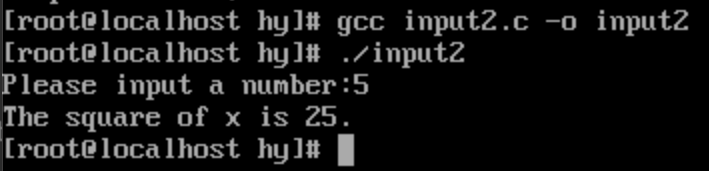
：输入操作数列表

：clobber 列表

);

程序如下：





说明：

①\_\_asm\_\_前后各两个下划线，并且两个下划线之间没有空格，用于声明这行代码是一个内嵌汇编表达式，是内嵌汇编代码时必不可少的关键字。

②关键字 volatile 前后各两个下划线，并且两个下划线之间没有空格。该关键字告诉编译器不要优化内嵌的汇编语句，如果想优化可以不加 volatile； 在很多时候，如果不使用该关键字的话，汇编语句有可能被编译器修改而无法达到预期的执行效果。

③括号里面包含四个部分：汇编代码（asm code）、输出操作数列表（output）、输入操作数列表（input）和 clobber 列表（破坏描述符）。这四个部分之间用“：”隔开。其中，输入操作数列表部分和 clobber 列表部分是可选的，如果不使用 clobber 列表部分，则格式可以简化为：

\_\_asm\_\_ \_\_volatile\_\_ (“asm code”：output：input);

如果不使用输入部分，则格式可以简化为：

\_\_asm\_\_ \_\_volatile\_\_ (“asm code”：output::changed);

此时，即使输入部分为空，输出部分之后的“：”也是不能省略的。另外，输入部分和 clobber 列表部分是可选的，如果都为空，则格式可以简化为：

\_\_asm\_\_ \_\_volatile\_\_ (“asm code”：output);

④括号之后要以“;”结尾。

⑤在上例中，输出操作数列表 :"=r" (x) 代表输出操作数存放在某个通用寄存器中，即在汇编代码里用一个寄存器代替()部分中定义的c变量x；=代表只写，即在汇编代码里只能改变 C 变量的值，而不能取它的值。输入操作数列表 :"0"(x) 0代表与第一个输出参数共用同一个寄存器。