基础代码示例实验手册

版本1.0



华为技术有限公司

目录

[1 课程介绍 3](#_Toc55554575)

[1.1 简介 3](#_Toc55554576)

[1.2 内容描述 3](#_Toc55554577)

[1.3 读者知识背景 3](#_Toc55554578)

[1.4 实验环境说明 3](#_Toc55554579)

[2 实验1：hello-world示例程序 4](#_Toc55554580)

[2.1 实验介绍 4](#_Toc55554581)

[2.1.1 关于本实验 4](#_Toc55554582)

[2.1.2 教学目标 4](#_Toc55554583)

[2.1.3 实验内容介绍 4](#_Toc55554584)

[2.2 实验任务操作指导 5](#_Toc55554585)

[2.2.1 创建示例程序源码 5](#_Toc55554586)

[2.2.2 思考题及答案 6](#_Toc55554587)

[3 实验2：使用C语言代码调用汇编程序 7](#_Toc55554588)

[3.1 实验介绍 7](#_Toc55554589)

[3.1.1 关于本实验 7](#_Toc55554590)

[3.1.2 教学目标 7](#_Toc55554591)

[3.1.3 实验内容介绍 7](#_Toc55554592)

[3.2 实验任务操作指导 7](#_Toc55554593)

[3.2.1 创建示例程序源码 7](#_Toc55554594)

[3.2.2 进行编译运行 9](#_Toc55554595)

[3.3 思考题及答案 9](#_Toc55554596)

[4 实验3：使用C语言代码内嵌汇编程序 10](#_Toc55554597)

[4.1 实验介绍 10](#_Toc55554598)

[4.1.1 关于本实验 10](#_Toc55554599)

[4.1.2 教学目标 10](#_Toc55554600)

[4.1.3 内容介绍 10](#_Toc55554601)

[4.2 实验任务操作指导 10](#_Toc55554602)

[4.2.1 创建示例程序源码 10](#_Toc55554603)

[4.2.2 进行编译 12](#_Toc55554604)

[4.2.3 进行运行 12](#_Toc55554605)

[4.2.4 思考题及答案 12](#_Toc55554606)

# 课程介绍

## 简介

本手册适用于学习ARM平台汇编课程的学生进行实验练习，完成本实验手册后，您将能更加充分理解GNU ARM汇编代码运行环境的搭建、配置及编译运行，掌握在华为鲲鹏云服务器上进行环境配置。

## 内容描述

本实验指导书通过在华为鲲鹏云服务器上，编译运行3个不同功能的示例程序。完成实验操作后，读者会掌握基本的汇编程序编写，ARMv8开发编译环境的配置以及加深对ARM平台的了解。

## 读者知识背景

本课程为ARM平台汇编基础课程，为了更好地掌握本书内容，阅读本书的读者应首先具备以下基本条件：

* 具备基本的Linux命令能力。

## 实验环境说明

* 华为鲲鹏云主机、openEuler20.03操作系统；
* 安装gcc7.3+版本；
* 每套实验环境可供1名学员上机操作。

# 实验1：hello-world示例程序

## 实验介绍

### 关于本实验

实现ARM平台精简指令集（RISC）编写的hello-world程序的编译和运行。

### 教学目标

掌握GNU ARM平台汇编代码的编写以及编译运⾏方式。

### 实验内容介绍

在本例子中，两次使用软中断指令svc来进行系统调用，系统调用号通过x8寄存器传递。在第一次使用svc指令来在屏幕上打印一个字符串“Hello”：x0寄存器用于存放标准屏幕输出stdout描述符0，表明将向屏幕输出一些内容；x1寄存器用于存放待输出的字符串的首地址msg；x2寄存器用于存放待输出字符串的长度len；x8寄存器用于存放系统功能调用号64，即64号系统功能即系统写功能sys\_write()，写的目标在x0中定义；svc #0表示是一个系统功能调用。

第二次使用svc指令来退出当前程序：x0寄存器用于存放退出操作码123，不同的退出操作码将对应不同的退出操作；x8寄存器用于存放系统功能调用号93，即93号系统功能即系统退出功能sys\_exit()，退出操作码在x0中定义；svc #0表示是一个系统功能调用。

注意：像这种系统功能调用的方式和功能号，都是基于Arm64处理器体系结构以及之上所运行的linux kernel甚至BIOS来共同支持，而不仅仅是Arm64架构自身所能完成的。

在.data部分，加载msg和len实际上使用的是文字池的方法，即将变量地址放在代码段中不会执行到的位置（因为第二次使用svc指令来退出当前程序之后，是不可能将svc #0指令之后的内容来当做指令加以执行的），使用时先加载变量的地址，然后通过变量的地址得到变量的值。

本代码是Aarch64体系结构的汇编代码，需要在ArmV8处理器上运行。寄存器Xn都是Aarch64体系结构中的寄存器，svc是Aarch64体系结构中的指令。

## 实验任务操作指导

### 创建示例程序源码

以下步骤以在华为鲲鹏云服务器上执行为例。

创建hello目录

执行以下命令，创建hello目录，存放该程序的所有文件, 并进入hello目录。

mkdir hello

cd hello

创建示例程序源码hello.s

执行以下命令，创建示例程序源码hello.s。

vim hello.s

代码内容如下：

.text

.global tart1

tart1:

mov x0,#0

ldr x1,=msg

mov x2,len

mov x8,64

svc #0

mov x0,123

mov x8,93

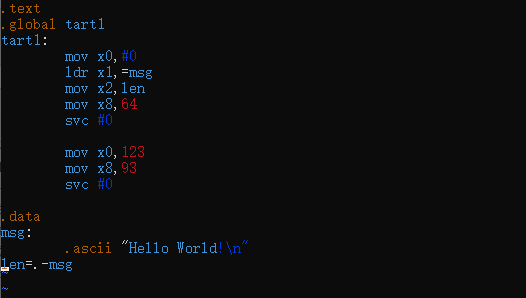
svc #0

.data

msg:

.ascii "Hello World!\n"

len=.-msg



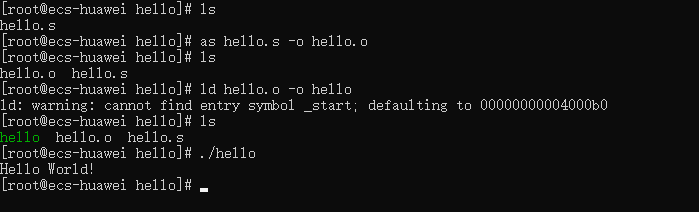
进行编译运行

保存示例源码文件，然后退出vim编辑器。在当前目录中依次执行以下命令，进行代码编译运行。

as hello.s –o hello.o

ld hello.o –o hello

./hello



通过上述代码运行，可以看出，编写的hello-wolrd示例程序已经在华为鲲鹏云服务器上通过编译和运行，并成功输出结果。

### 思考题及答案

* 思考下同样的代码在X86平台能否运行，为什么？

参考答案：

不能，因为X86平台使用的是复杂指令集(CISC)，而我们实验中使用到的华为鲲鹏云服务器是基于ARM平台的，使用的是精简指令集(RISC)，二者的汇编指令差异较大。

# 实验2：使用C语言代码调用汇编程序

## 实验介绍

### 关于本实验

实现ARM平台上通过C语言源码来调用汇编源码中的代码。

### 教学目标

掌握在ARM平台上使用C语言源码来调用汇编源码的方法。

### 实验内容介绍

该汇编代码是针对Aarch64架构的。在汇编程序中，用.global定义一个全局函数strcpy1，然后该函数就可以在C代码中用extern关键字加以声明，然后直接调用。

## 实验任务操作指导

### 创建示例程序源码

以下步骤以在华为鲲鹏云服务器上执行为例。

创建目录

执行以下命令，创建called目录存放该程序的所有文件, 并进入called目录。

mkdir called

cd called

创建globalCalling.c源代码

执行以下命令，创建示例调用C语言程序源码globalCalling.c。

vim globalCalling.c

代码内容如下：

/\* globalCalling.c\*/

#include <stdio.h>

extern void strcpy1(char \*d, const char \*s);

int main()

{

const char \*srcstring="Source string";

char dststring[]="Destination string";

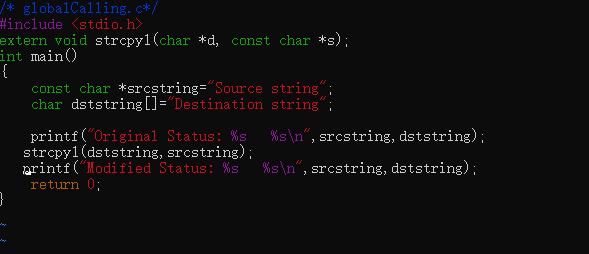
printf("Original Status: %s %s\n",srcstring,dststring);

strcpy1(dststring,srcstring);

printf("Modified Status: %s %s\n",srcstring,dststring);

return 0;

}



创建globalCalled.S源代码

执行以下代码命令，创建被调用的汇编语言程序源码globalCalled.S。

vim globalCalled.S

代码内容如下：

/\* globalCalled.S \*/

.global strcpy1

# Start the function: strcpy1

strcpy1:

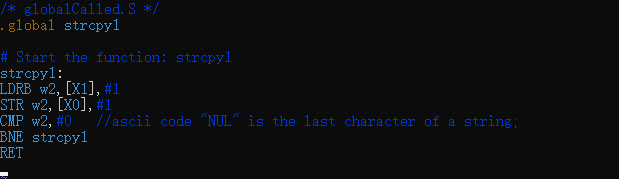
LDRB w2,[X1],#1

STR w2,[X0],#1

CMP w2,#0 //ascii code "NUL" is the last character of a string;

BNE strcpy1

RET

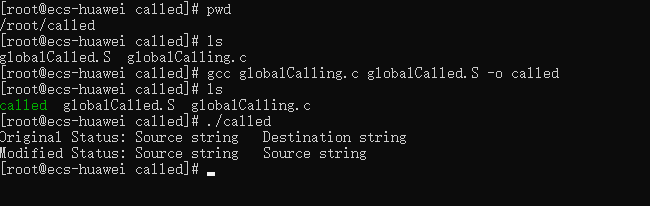


### 进行编译运行

保存示例源码文件，然后退出vim编辑器。在当前目录中依次执行以下命令，进行代码编译运行。

gcc globalCalling.c globalCalled.S -o called

./called



通过上述代码运行，可以看出，编写的使用C语言代码调用汇编程序已经在华为鲲鹏云服务器上通过编译和运行，并成功输出结果：

Original Status: Source string Destination string

Modified Status: Source string Source string

## 思考题及答案

* 除了使用调用汇编脚本的方式，还有哪些方法可以使用C语言调用汇编代码？

参考答案：

可以使用C语言中的asm关键字来在C语言源码中嵌入汇编代码。

# 实验3：使用C语言代码内嵌汇编程序

## 实验介绍

### 关于本实验

实现在ARM平台上通过C语言代码内嵌汇编代码的方式，将一个整数类型值，以字节为单位从小尾端转到大尾端或者相反的功能。

### 教学目标

掌握在ARM平台上实现C语言代码中内嵌汇编代码的方法。

### 内容介绍

通过C语言代码内嵌汇编代码，将一个整数类型值，以字节为单位从小尾端转到大尾端或者相反的功能。例如小尾端时32bit整数值用16进制表示为0x12345678，将其以字节为单位转换为大尾端存储后，该值为0x78563412。

## 实验任务操作指导

### 创建示例程序源码

以下步骤以在华为鲲鹏云服务器上执行为例。

创建目录

执行以下命令，创建builtin目录存放该程序的所有文件, 并进入bulitin目录。

mkdir builtin

cd builtin

创建C语言内嵌汇编程序源代码

执行以下命令，创建C语言内嵌汇编程序源码globalBuiltin.c。

vim globalBuiltin.c

代码内容如下：

/\* globalBuiltin.c\*/

#include <stdio.h>

int main()

{

int val=0x12345678;

\_\_asm\_\_ \_\_volatile\_\_(

"mov x3,%1\n"

"mov w3,w3, ror #8\n"

"bic w3,w3, #0x00ff00ff\n"

"mov x4,%1\n"

"mov w4,w4, ror #24\n"

"bic w4,w4, #0xff00ff00\n"

"add w3,w4,w3\n"

"mov %0,x3\n"

:"=r"(val)

:"0"(val)

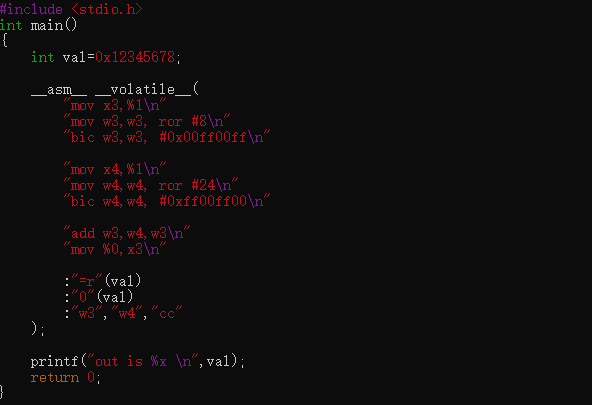
:"w3","w4","cc"

);

printf("out is %x \n",val);

return 0;

}



### 进行编译

保存示例源码文件，然后退出vim编辑器。在当前目录中依次执行以下命令，进行代码编译。

预处理

gcc -E globalBuiltin.c -o globalBuiltin.i

编译

gcc -S globalBuiltin.i -o globalBuiltin.s

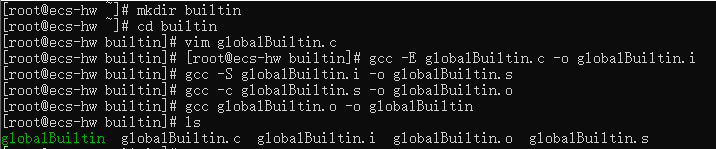
汇编

gcc -c globalBuiltin.s -o globalBuiltin.o

生成可执行文件

gcc globalBuiltin.o -o globalBuiltin

命令和生成的文件如下：



### 进行运行

运行生成的globalBuiltin文件，查看输出结果。

命令如下：

./globalBuiltin

结果如下：



通过上述代码运行，可以看出，编写的C语言代码内嵌汇编程序已经在华为鲲鹏云服务器上通过编译和运行，并成功输出结果：out is 78563412。

### 思考题及答案

* 在C代码中内嵌的汇编语句的基本格式为是什么？

参考答案：

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

：输出操作数列表

：输入操作数列表

：clobber列表

)

说明：《汇编与接口技术》课程配套实验手册中的实验内容由北京交通大学计算机与信息技术学院赵宏智老师提供，华为公司负责实验手册文档的编写。