

南開大學

## 汇编语言与逆向技术课程实验报告

### 实验四：ARM 平台 HelloWorld



学 院 网络空间安全学院  
专 业 信息安全  
学 号 2211044  
姓 名 陆皓喆  
班 级 信息安全

## 一、实验目的

- 1、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。
- 2、命令行输出“HelloWorld”。

## 二、实验内容

以下步骤以在华为鲲鹏云服务器上执行为例。

### 1.创建 hello 目录

```
mkdir hello  
cd hello
```

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。

### 2.创建示例程序代码 `hello.s`

```
vim hello.s
```

执行以下命令，创建示例程序源码 `hello.s`。

代码如下：

```
.text
.global _start
_start:
    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
```

### 3.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-world` 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。

### 三、代码解析

`.text;`标识后面有指令代码

`.global _start;` 声明了一个名称为“\_start”的全局符号

`_start::` 此处是“\_start”标签，为程序的入口点

`mov x0,#0;` 把立即数 0 存到寄存器 x0 里

`ldr x1,msg;` 将字符串 msg (“Hello World!\n”) 的首地址储存在寄存器 x1 里

`mov x2,len;` 将字符串 msg (“Hello World!\n”) 的长度储存在寄存器 x2 里

`mov x8,64;` 将立即数 64 储存在寄存器 x8 里

`svc #0;` 这是一个特殊操作码，终端程序的执行并转移控制权到操作系统。在这里是调用操作系统的系统调用打印字符串“Hello World!\n”

`mov x0,123;` 将立即数 123 存储到寄存器 x0 里

`mov x8,93;` 将立即数 93 存储到寄存器 x8 里

`svc #0;` 这里是再次调用操作系统的系统调用，用于退出程序

`.data ;`定义数据段

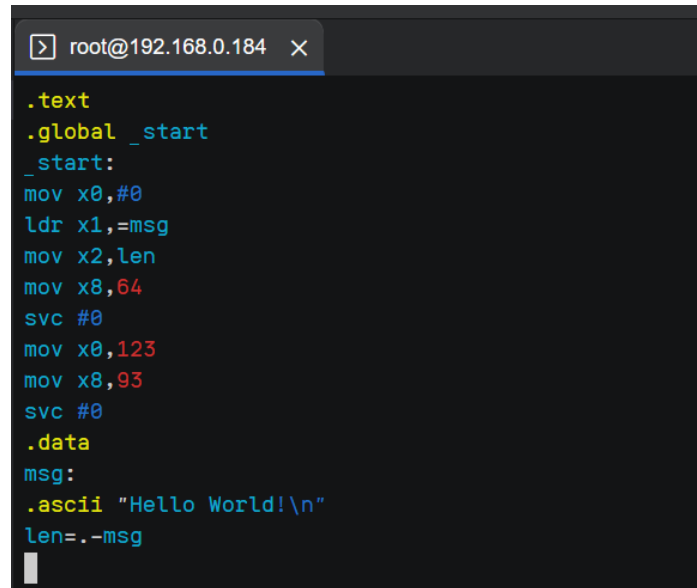
`msg: ;` 定义一个名为“msg”的字符串

`.ascii "Hello World!\n";`定义字符串常量“Hello World!\n”

`len=.-msg;` “.-”符号用于计算当前位置和 msg 标签之间的字节数，即字符串 msg 的长度

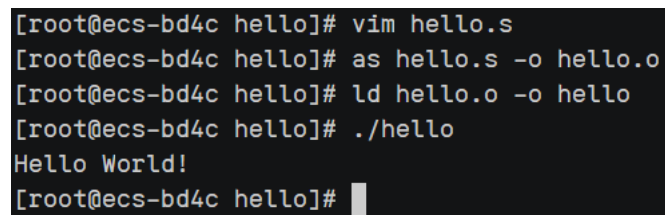
## 四、运行截图

### 1、“hello.s”的编写



```
root@192.168.0.184 x
.text
.global _start
_start:
mov x0,#0
ldr x1,msg
mov x2,len
mov x8,64
svc #0
mov x0,123
mov x8,93
svc #0
.data
msg:
.asciiz "Hello World!\n"
len=.-msg
```

### 2、运行结果截图



```
[root@ecs-bd4c hello]# vim hello.s
[root@ecs-bd4c hello]# as hello.s -o hello.o
[root@ecs-bd4c hello]# ld hello.o -o hello
[root@ecs-bd4c hello]# ./hello
Hello World!
[root@ecs-bd4c hello]#
```

## 五、思考题

同样的代码能否在 x86 平台运行，为什么？

答：不能在 x86 平台上运行。因为代码中使用了国产的鲲鹏处理器的特有的一些命令与寄存器，比如说 `svc`、`ldr` 等等。在 x86 平台中没有这些特殊的命令与寄存器，所以不能够运行。