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　2313546
姓　　名　蒋枘言
班　　级　信息安全班

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“Hello World”。

二、实验环境

华为鲲鹏云主机、openEuler20.03 操作系统。

三、实验过程

1.创建 hello 目录并进入，用于存放该程序所有文件：执行以下命令

```
mkdir hello
cd hello
```

2.创建示例程序源码 hello.s：执行以下命令

```
vim hello.s
```

3.代码解析

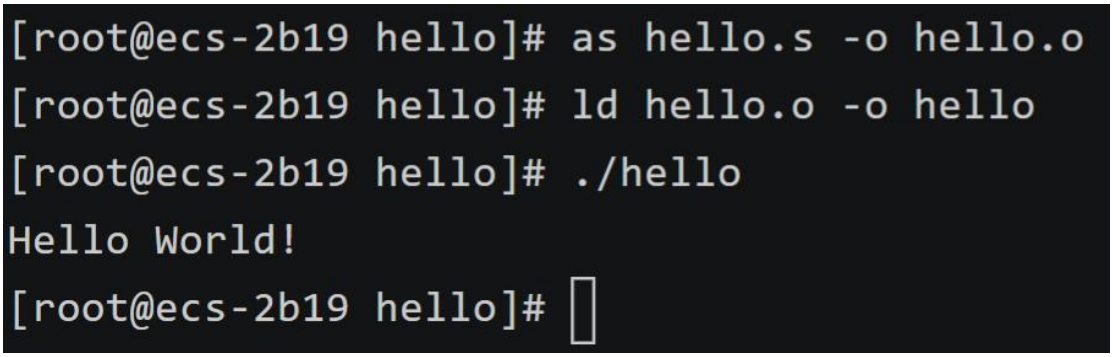
```
.text
;.text 段通常用于存放程序的执行指令
.global _start
;.global 指令将符号（在这个例子中是 start）标记为全局。
;_start 是大多数 UNIX-like 系统上的程序入口点，即程序开始执行的地方。
_start:
;这是一个标签，标记了程序执行的开始位置。
mov x0,#0
;这条指令将立即数 0 移动到寄存器 x0 中。
ldr x1,=msg
;这条指令将 msg 标签的地址加载到寄存器 x1 中。
;ldr 指令用于加载数据，而=msg 用于获取标签 msg 的地址。
mov x2,len
;这条指令将 len 的值移动到寄存器 x2 中。len 是一个符号，表示 msg 字符串的长度。
mov x8,64
;将立即数 64 移动到寄存器 x8 中。
svc #0
;执行一个系统调用。svc（Supervisor Call）指令用于触发系统调用。
;#0 通常表示一个立即数参数，但在 svc 指令的上下文中，它并不直接指定系统调用号。
;系统调用号是通过 x8 寄存器传递的。
;因此，这行代码实际上执行了之前通过 x8 寄存器指定的系统调用。
mov x0,123
;将立即数 123 移动到寄存器 x0 中。
mov x8,93
;将立即数 93 移动到寄存器 x8 中。
svc #0
;再次执行一个系统调用，这次使用的是 x8 寄存器中的新值（93）。
.data
;这行指令告诉汇编器接下来的部分包含数据段。
msg:
;这是一个标签，标记了字符串 Hello World!的开始位置。
.ascii "Hello World!\n"
;这条指令定义了一个 ASCII 字符串 Hello World!，并将其存放在数据段中。
len=.-msg
;这是一个计算 msg 字符串长度的表达式。.表示当前位置，msg 是字符串开始的标签。
;因此，.-msg 计算当前位置与 msg 标签之间的字节差，这个差值就是字符串的长度。
;这个值被赋给符号 len。
```

4.编译与运行：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

```
as hello.s -o hello.o
```

```
ld hello.o -o hello
./hello
```

4.结果：结果如下



四、思考题

1.同样的代码能否在 x86 平台运行？为什么？

答：同样的代码不能在 x86 平台上直接运行，主要原因有

①指令集差异：ARM 和 x86 是两种完全不同的处理器架构，它们有各自的指令集。

②寄存器命名和用途：ARM 和 x86 架构中的寄存器命名和用途也不同。