

汇编语言与逆向技术实验报告

Lab8 - ARM 平台-HelloWorld

学号：2112492 姓名：刘修铭 专业：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、实验环境

1.华为鲲鹏云主机

2.openEuler20.03 操作系统

三、实验过程及运行结果

（一）搭建华为云环境，利用 **temius** 的 SSH 工具远程登录

1.创建 hello 目录

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
```

2.创建程序代码 hello.s

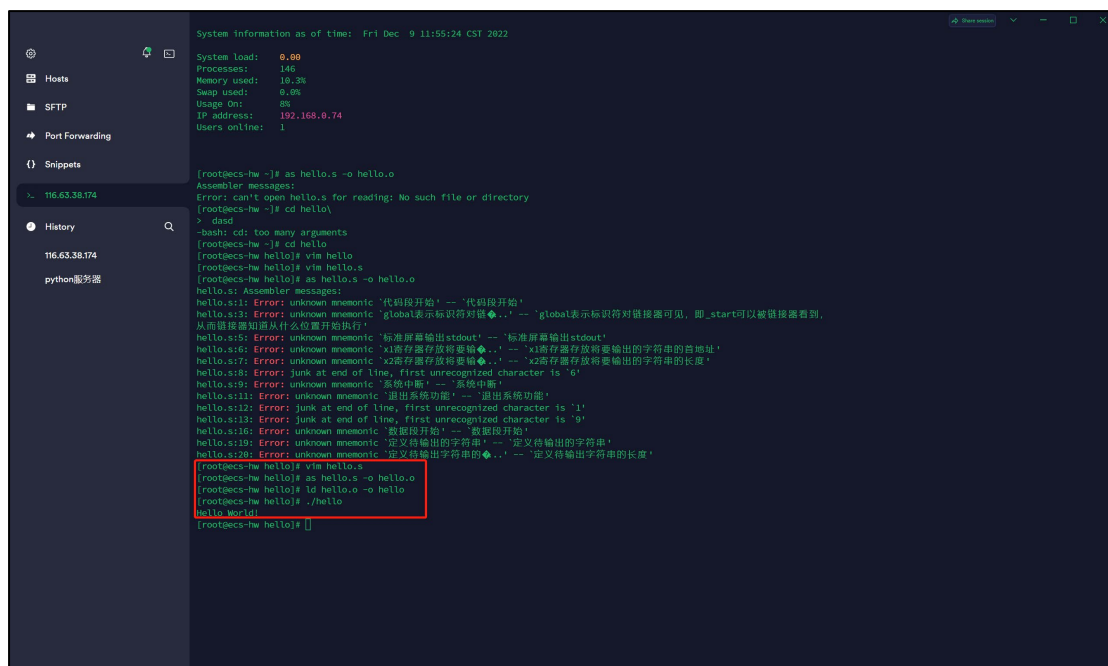
```
vim hello.s
```

3.编译运行

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

```
as hello.s -o hello.o  
ld hello.o -o hello  
./hello
```

运行结果如图



```
System information as of time: Fri Dec 9 11:55:24 CST 2022
System load: 0.00
Processes: 146
Memory used: 10.3%
Swap used: 0.0%
Usage On: 8%
IP address: 192.168.0.74
Users online: 1

[root@ecs-hw ~]# as hello.s -o hello.o
Assembler messages:
Error: can't open hello.s for reading: No such file or directory
[root@ecs-hw ~]# cd hello\
> dad
-bash: cd: too many arguments
[root@ecs-hw ~]# cd hello
[root@ecs-hw hello]# vim hello
[root@ecs-hw hello]# vim hello.s
[root@ecs-hw hello]# as hello.s -o hello.o
hello.s: Assembler messages:
hello.s:1: Error: unknown mnemonic '代码段开始' -- '代码段开始'
hello.s:2: Error: unknown mnemonic 'global标识符对链接器可见，即_start可以被链接器看到，从而链接器知道从什么位置开始执行' -- 'global标识符对链接器可见，即_start可以被链接器看到，从而链接器知道从什么位置开始执行'
hello.s:3: Error: unknown mnemonic '标准屏幕输出stdout' -- '标准屏幕输出stdout'
hello.s:4: Error: unknown mnemonic 'x1寄存器存放将要输出的字符串的首地址' -- 'x1寄存器存放将要输出的字符串的首地址'
hello.s:7: Error: unknown mnemonic 'x2寄存器存放将要输出的字符串的长度' -- 'x2寄存器存放将要输出的字符串的长度'
hello.s:8: Error: junk at end of line, first unrecognized character is 'q'
hello.s:9: Error: unknown mnemonic '系统中断' -- '系统中断'
hello.s:11: Error: unknown mnemonic '退出系统功能' -- '退出系统功能'
hello.s:12: Error: junk at end of line, first unrecognized character is 'l'
hello.s:13: Error: junk at end of line, first unrecognized character is '9'
hello.s:16: Error: unknown mnemonic '数据段开始' -- '数据段开始'
hello.s:19: Error: unknown mnemonic '定义待输出的字符串' -- '定义待输出的字符串'
hello.s:20: Error: unknown mnemonic '定义待输出字符串的长度' -- '定义待输出字符串的长度'
[root@ecs-hw hello]# vim hello.s
[root@ecs-hw hello]# as hello.s -o hello.o
[root@ecs-hw hello]# ld hello.o -o hello
[root@ecs-hw hello]# ./hello
Hello World!
[root@ecs-hw hello]#
```

四、源代码及语句解析

;代码段开始

.text

.global _start.global ;global 表示标识符对链接器可见，即_start 可以被链接器看到，从而链接器知道从什么位置开始执行

_start:

mov x0,#0 ;标准屏幕输出 stdout

ldr x1,msg ;x1 寄存器存放将要输出的字符串的首地址

mov x2,len ;x2 寄存器存放将要输出的字符串的长度

mov x8,64 ;64 是系统调用号，对应系统输出，输出目标存在 x0 寄存器

svc #0 ;系统中断

;退出系统功能

mov x0,123 ;123 是调用号，对应一个退出操作

mov x8,93 ;93 是系统调用号，对应系统退出功能

svc #0

;数据段开始

.data

msg:

.ascii "Hello World!\n" ;定义待输出的字符串

len=.-msg ;定义待输出字符串的长度

五、思考题

同样的代码能否在 x86 平台运行，为什么？

不能。因为 x86 架构和 ARM 架构的指令集不同，寄存器数目也不同。所以相同的代码在 x86 平台无法运行，因为其指令集无法被另一种架构识别。