

# 南开大学

Huawei Kunpeng

(汇编语言与逆向技术实验 4)



姓名: 申宗尚

学号: 2213924

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

华为鲲鹏云主机、openEuler20.03 操作系统；

## 三．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

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
```

### 2. 创建示例程序代码 hello.s

执行以下命令，创建示例程序源码 hello.s。

```
vim hello.s

.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
```

代码内容如下：

### 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
```

```
[root@ecs-huawei hello]# ls
hello.s
[root@ecs-huawei hello]# as hello.s -o hello.o
[root@ecs-huawei hello]# ls
hello.o  hello.s
[root@ecs-huawei hello]# ld hello.o -o hello
ld: warning: cannot find entry symbol _start; defaulting to 00000000004000b0
[root@ecs-huawei hello]# ls
hello  hello.o  hello.s
[root@ecs-huawei hello]# ./hello
Hello World!
[root@ecs-huawei 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。

## 四. 实验代码及注释解析

```
.text
.global _start          ; 声明 _start 为全局标签
_start:
    mov x0, #0           ; 将寄存器 x0 设置为 0，通常用作程序的返回值
    ldr x1, =msg          ; 将 msg 的地址加载到寄存器 x1
    mov x2, len           ; 将 len 的值加载到寄存器 x2
    mov x8, 64           ; 将系统调用号 64 (write) 加载到寄存器 x8
    svc #0               ; 触发系统调用
    mov x0, 123          ; 将寄存器 x0 设置为 123
    mov x8, 93           ; 将系统调用号 93 (exit) 加载到寄存器 x8
    svc #0               ; 触发系统调用

.data
msg:
.ascii "Hello World!\n"  ; 存储字符串 "Hello World!\n"
len = . - msg            ; 计算字符串长度
```

终端代码：

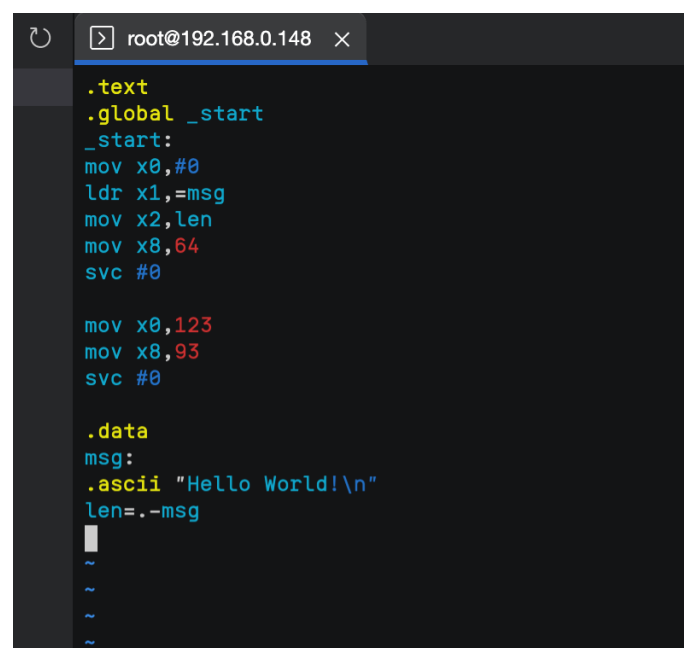
mkdir hello	创造 hello 文件夹
cd hello	转换路径
vim hello.s	用 vim 编辑器创建 hello.s 文件
as hello.s -o hello.o	编译文件

ld hello.o -o hello	链接文件
./hello	运行文件

## 五. 实验程序测试

```
[root@ecs-fb01 ~]# ls
[root@ecs-fb01 ~]# ls
[root@ecs-fb01 ~]# pwd
/root
[root@ecs-fb01 ~]# mkdir hello
[root@ecs-fb01 ~]# cd hello
[root@ecs-fb01 hello]# vim hello.s
```

如图进入终端后，先进行文件夹的创造，再进入文件夹，用 vim 编辑器创造 hello.s 文件编写代码



```
root@192.168.0.148 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:
.ascii "Hello World!\n"
len=.-msg
~
~
~
~
```

如图，为代码内容

```
[root@ecs-fb01 hello]# as hello.s -o hello.o
[root@ecs-fb01 hello]# ld hello.o -o hello
[root@ecs-fb01 hello]# ./hello
Hello World!
[root@ecs-fb01 hello]#
```

如图，在进行了编译和链接后，执行文件，输出了

“Hello World! ”

## 六、思考题

答：这段 ARM 架构下的代码不能直接在 x86 架构下运行。

不同的处理器架构使用不同的指令集和体系结构，因此编写的汇编代码是特定于架构的。

本次实验代码中，使用的是 ARM 指令集的指令，比如 `mov`、`ldr`、`svc` 等。这些指令专门设计用于 ARM 架构。

若想要在 x86 架构下运行，需要编写使用 x86 指令集的汇编代码。例如，x86 架构中的指令 `mov`、`lea`、`int` 等。