# Mips学习总结

## 函数调用传参和系统响应

寄存器$a0-$a3用于函数值的传递，但是对于syscall的使用也常常需要对于$a0,$a1进行赋值操作，如果这两个情况在同一个函数中发生，如果对于原来的$a0-$a3不做保护，就会造成数据丢失导致错误。

**在使用宏定义的时候更是如此，因为可能会忘了这个指令中包含对于系统的调用，没有对$a0进行保护**

```
.macro printString(%src)
    push($a0)
    la $a0,%src
    li $v0,4
    syscall
    pop($a0)
.end_macro

.macro printInt(%src)
    push($a0)
    move $a0,%src
    li $v0,1
    syscall
    pop($a0)
.end_macro
```

以这两个为例，要注意对于$a0的保护！！！

**在循环里面如果进行了跳转，那就一定要注意对i的push**

## 标签名字对应正确

如果有类似的代码部分，**最好不要复制粘贴**，如果用了标签，真的很容易忘了改跳转时的标签。。。

## 连续字符串的读入

在读入一个数字后，连续读入一些列字符的情况，有以下几种处理方式

### 1.使用$v0=12一行读入

```
li $v0,5
syscall      # 读入数字
li $v0,12
syscall      # 读入单个字符
   ……
```

在这里不用考虑回车的影响，直接读入即可

在mars**的输入里要注意在一行里输入字符**

```
5        # 有回车
abcde    # 一行读入
```

## 2.使用$v0=12读入，但是分行读

```
li $v0,5
syscall     # 读入数字
li $v0,12
syscall     # 读字符
li $v0,12
syscall     # 吃掉回车
```

在mars中输入

```
5       # 有回车
a       # 分行读入
b
c
d
e
```

## 3.使用$v0=8读(容易出bug)

```
.macro getapl(%src)   # 读入字母，返回相对于a的距离
    la $a0,str
    li $a1,3         # 重要的地方，$a1只能为3，否则就会读不进去
    li $v0,8
    syscall
    lb %src,str
    addi %src,%src,-97
.end_macro

.data
    str: .space 3        # 用于每次存放读的那一串
.text
    li $v0,5
    syscall     # 读入数字
for:
    ……
    getapl($t1) # 读入字母
    ……
end_for:
```

**对于字符串str的长度没有限制，但是对于输入的时候的$a1，必须设置为3，否则就会读着读着报错**

**其实这也就是程设中的CRLF坑**(助教告诉我的)

在mars中输入

```
5       # 有回车
a       # 分行读入
b
c
d
e
```

## if else

要注意if，else的每个分支，运行结束后要跳到最后，不要流入到下一个else if分支。

## 矩阵相关

对于寻址，当前行要乘列的总数加上当前列，要注意一定是乘n