

# 深圳大学实验报告

课程名称: 计算机系统(3)

实验项目名称: MIPS64 乘法器模拟实验

学 院: 计算机与软件学院

专 业: 计算机科学与技术/软件工程

指导教师: 王 毅

报告人: 郑雨婷 学号: 2021150122 班级: 高性能

实 验 时 间: 2023 年 10 月 6 日

实验报告提交时间: 2023 年 10 月 19 日

## 一、实验目标：

1. 实际运用 WinMIPS64 进行实验，以期更了解 WinMIPS64 的操作；
2. 更加深入地了解 MIPS 程序的语法；
3. 深入地了解在计算机中乘法的实现以及加法与乘法之间的关系。

## 二、实验内容

按照下面的实验步骤及说明，完成相关操作记录实验过程的截图：

首先，我们使用加法操作设计一个不检测溢出的乘法操作；完成后，我们对此进行优化，以期获得一个可以对溢出进行检测的乘法操作。（100 分）

## 三、实验环境

硬件：桌面 PC

软件：Windows，WinMIPS64 仿真器

## 四、实验步骤及说明

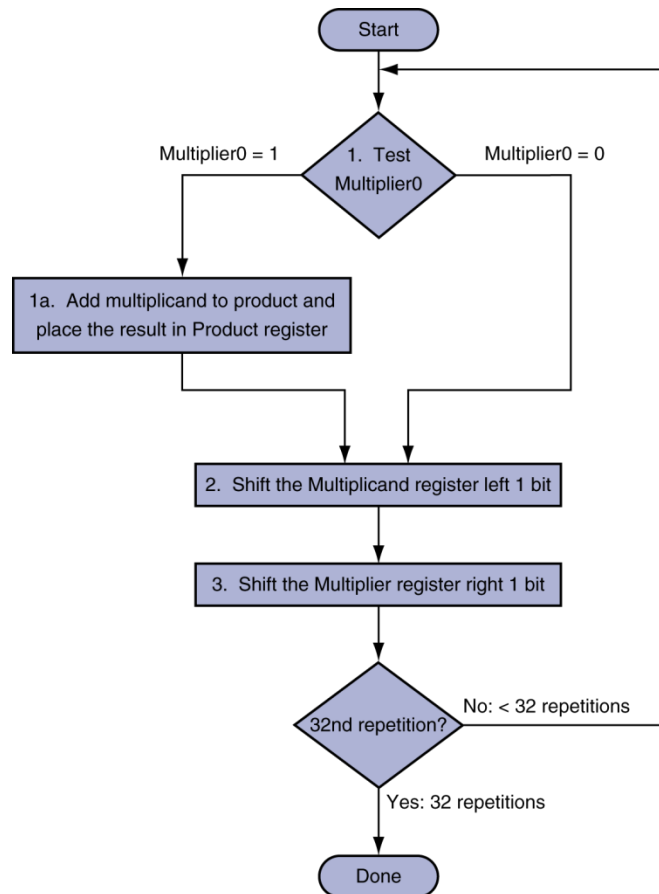
本次试验分为两个部分：第一部分、用加法器设计一个不考虑溢出的乘法器；第二部分、用加法器设计一个考虑溢出的乘法器（编程熟练的同学，也可以用除法器、浮点加法器等替代）。

### 1、忽略溢出的乘法器

首先，我们得了解乘法器如何由加法器设计得到，此处，我们以 32 位乘法为例。

总共分为 4 步：

1. 测试乘数最低位是否为 1，是则给乘积加上被乘数，将结果写入乘积寄存器；
2. 被乘数寄存器左移 1 位；
3. 乘数寄存器右移一位；
4. 判断是否循环了 32 次，如果是，则结束，否则返回步骤 1。



运行显示运行结果的例子如下，由于我们这里展示的是忽略了溢出的乘法，所以结果有两种：1、小于 32 位；2、大于 32 位。

第一种情况截图：

```

Terminal
please enter two numbers:
13
13
result:
169
  
```

第二种情况截图：

```

Terminal
please enter two numbers:
1000001
10000002
result:
1328134914
  
```

根据上面的程序代码和截图，我们可以很清楚的看出，当结果小于32位时，结果正常；当结果大于32位时，结果只截取了低32位的结果，而高32位的结果直接忽略掉了。

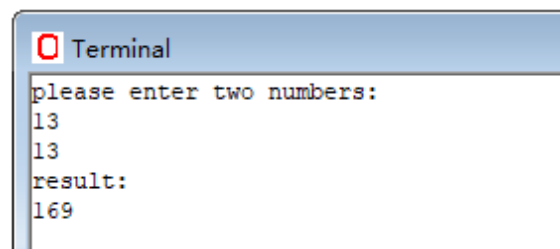
## 2、溢出提示的乘法器

上述的程序，用加法实现了 32 位乘法，但是，其中，对溢出情况没有进行考虑是其中的弊端。这里，我们来完善上述的乘法器，使得该乘法器会在结果溢出时候提示。

其实，这个小优化是十分简单的，只需要对 64 位的寄存器中的高 32 位进行检测即可。当高 32 位为 0 时，说明结果没有溢出，否则，结果溢出。

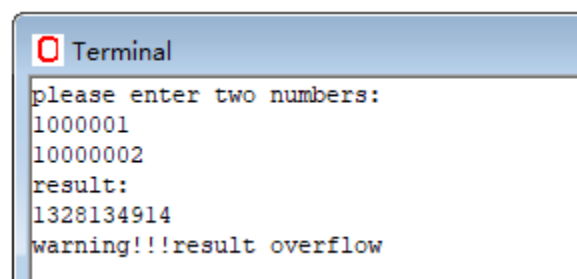
上述代码运行结果也有两个，一个是没有溢出的情况下的结果，一个是溢出了的情况下的结果。

首先，我们看没有溢出的情况结果：



```
Terminal
please enter two numbers:
13
13
result:
169
```

结果正确，其次，我们看溢出的情况结果如何：



```
Terminal
please enter two numbers:
1000001
1000002
result:
1328134914
warning!!!result overflow
```

可以看到，当结果溢出时，程序会给出提示“warning!! 1result overflow”。

## 4 结束语

本实验介绍了通过加法器来设计乘法器的原理，并且在编写该实验程序的时候，我们更加了解了：1、计算机乘法器工作原理的内容；2、进一步熟练 MIPS 的编程方法；3、WinMIPS64 的使用方法。当然，如果想要更加深入的学习，我们也可以课外继续编写对除法的模拟。

## 五、实验结果

### 1.不检测溢出的乘法器：

先新建一个 myUnlimitedMult.s 文件，用来实现用加法器实现的不考虑溢出的乘法器。

先定义数据区，要实现数据的输出，需要定义 CONTROL 和 DATA 这两个数据区，用来控制输入或者输出数据的格式还有数据的内容。还有定义要输出的两个字符串的内容，数据格式的.asciiz 的。

```
.data
CONTROL: .word 0x10000
DATA: .word 0x10008
mes: .asciiz "please enter two numbers:\n"
result: .asciiz "result:\n"
w: .word 0
```

接着开始写代码。开始用\$t0 读取 DATA 指针指向的地址，用\$t1 读取 CONTROL 指针指向的内容。然后用\$t3 读取输出提示信息的 mes 字符串地址，将输出内容用 sd 指令写到 DATA 区，在往 CONTROL 控制区写入 4，表示输出的数据类型是字符串，这样就完成提示信息的输出。

```
ld $t0,DATA($zero)      ; get data
ld $t1,CONTROL($zero)   ; and control registers
daddi $t2,$zero,4       ; set for ascii output
daddi $t3,$zero,mes
sd $t3,0($t0)           ; write address of message to DATA register
sd $t2,0($t1)           ; make it happen
```

接下来完成数据的输入，先往 CONTROL 控制区写入数值 8，表示要输入数据，然后就可以用 terminal 控制台输入数据了，数据会被写入到 DATA 数据区，再用 ld 指令读取 DATA 数据区的内容。重复编写两次，完成对两个数据的输入。

```
daddi $t2,$zero,8       ; set for input
sd $t2,0($t1)
ld $t4,0($t0)           ;t4 = multiplicand
sd $t2,0($t1)
ld $t5,0($t0)           ;t5 = multiplier
```

要使用加法器实现 32 位乘法器的功能，那我们就需要控制累加的次数是 32 次。那我们就将\$t6 初始化为 32，用来控制循环的次数，再将\$t8 初始化为 0，用来累加，作为乘法的结果。然后那用 andi 指令获取乘数最低位，然后如果低位是 0 的话，不累加，如果低位是 1 的话，将乘数累加到\$t8 中。因此用\$t7 记录低位信息，乘数与 1 进行按位与即可得到最低位。

最后还需要实现乘数和被乘数的移位操作，将被乘数左移一位，将乘数右移一位，再将\$t6 的数值减一，如果\$t6 的值不为 0 的话，也就是循环次数还没到达 32 次，那就要跳转接着计算。

```
daddi $t6,$zero,32      ;i=32
daddi $t8,$zero,0
loop:
    andi $t7,$t5,1      ;t7是乘数的最低位
    beq $t7,$zero,weiyi
    dadd $t8,$t8,$t4
weiyi:
    dsll $t4,$t4,1
    dsrl $t5,$t5,1

    daddi $t6,$t6,-1
    beq $t6,$zero,end
    j loop
end:
```

最后，需要将结果输出到 terminal 屏幕上，还是将要输出的提示信息读取后写入到 DATA 数据区中，再往 CONTROL 写入 4，表示输出的字符串类型的数据。再将\$t8 结果截取前 32 位出来，写入数据区 DATA，再往 CONTROL 写入 2，表示输出的是有符号整形数据。

```
daddi $t2,$zero,4       ; set for ascii output
daddi $t3,$zero,result
sd $t3,0($t0)           ; write address of message to DATA register
sd $t2,0($t1)           ; make it happen
```

```

daddi $t9,$zero,-1
dsrl $t9,$t9,16
dsrl $t9,$t9,16
and $t9,$t8,$t9

daddi $t2,$zero,2      ; set for ascii output
sd $t9,0($t0)          ; write address of message to DATA register
sd $t2,0($t1)          ; make it happen

```

结果如下图所示：

无溢出情况的乘法

```

Terminal
please enter two numbers:
13
13
result:
169

```

溢出的情况：

```

Terminal
please enter two numbers:
1000001
10000002
result:
1328134914

```

## 2.检测溢出的乘法器：

新建一个 `Mmult.s` 文件，用来实现检测溢出乘法器。优化算法就是在输出结果之后，检测结果的高 32 位是否全都是 0，如果是，则表示乘法没有溢出，如果不是，则表示乘法溢出了。

那在未实现检测溢出乘法器基础上进行改进。首先是在 `.DATA` 的数据区新添加一个 `warning` 的输出提示信息，用于提示乘法溢出的情况。

```

.data
CONTROL: .word 0x10000
DATA: .word 0x10008
mes: .asciiz "please enter two numbers:\n"
result: .asciiz "result:\n"
w: .word 0
warning: .asciiz "warning!!!result overflow\n"

```

在跳出循环之后，先是把 `$t8` 的数值备份下来，再进行低 32 位的截断输出，再将原来的结果逻辑右移 32 位，看结果是否是 0，如果是的话，则表示结果的高 32 位都是 0，如果不是，则表示高 32 位不全都是 0，这就是有溢出的情况。那如果结果是 0 的话，直接跳转到结束语句 `finish`，如果非零的话，将溢出的 `warning` 信息输出之后再结束。

```

daddi $t9,$zero,-1
dsrl $t9,$t9,16
dsrl $t9,$t9,16
and $t9,$t8,$t9      ;t9 32bit ans,—t8 real ans

daddi $t2,$zero,2      ; set for ascii output
sd $t9,0($t0)          ; write address of message to DATA register
sd $t2,0($t1)          ; make it happen

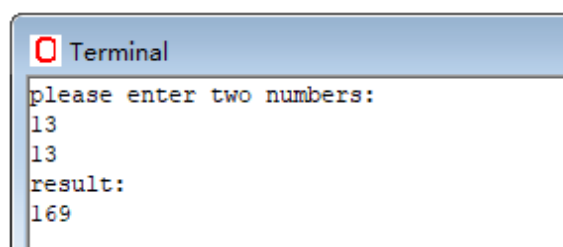
dsrl $t9,$t8,16
dsrl $t9,$t9,16
beqz $t9,finish

daddi $t2,$zero,4
daddi $t3,$zero,warning
sd $t3,($t0)
sd $t2,($t1)
finish:
halt

```

结果如下图所示：

不溢出的情况：

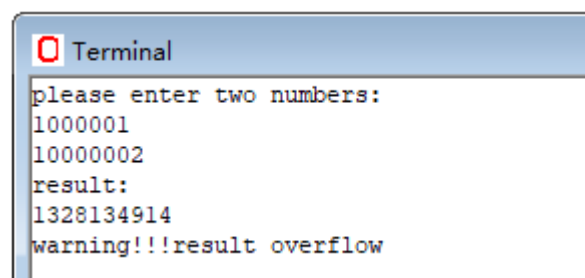


```

Terminal
please enter two numbers:
13
13
result:
169

```

溢出的情况：



```

Terminal
please enter two numbers:
1000001
10000002
result:
1328134914
warning!!!result overflow

```

## 六、实验总结与体会

本次实验分为两个部分，首先，我们设计了一个不考虑溢出的乘法器，然后改进它，设计了一个考虑溢出的乘法器。以下是对这两部分的总结与体会：

在第一部分中，我们学习了如何使用加法器来设计一个不考虑溢出的乘法器。这个乘法器执行了一个简单的 32 位乘法操作，其基本原理是将乘数的每一位与被乘数相乘，然后将结果相加。虽然这个乘法器功能简单，但它为我们提供了一个重要的概念，即乘法可以通过加法来实现。

在第二部分中，我们对第一部分的乘法器进行了改进，以考虑结果溢出的情况。这是一个重要的改进，因为在实际计算中，溢出可能会导致不正确的结果或错误。通过在计算的过程中检查高 32 位是否为 0，我们能够检测结果是否溢出，从而更安全地使用这个乘法器。

指导教师批阅意见:

成绩评定:

指导教师签字： 王毅

2023 年 11 月 1 日

备注:

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。