

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: 计算机与软件学院所有专业

指导教师: 罗秋明

报告人: 林浩晟 学号: 2022280310 班级: 01

实 验 时 间: 2024 年 9 月 29 日星期日

实验报告提交时间: 2024 年 10 月 10 日星期四

一、实验目标：

实际运用 WinMIPS64 进行试验，以期更了解 WinMIPS64 的操作；
更加深入地了解 MIPS 程序的语法；
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

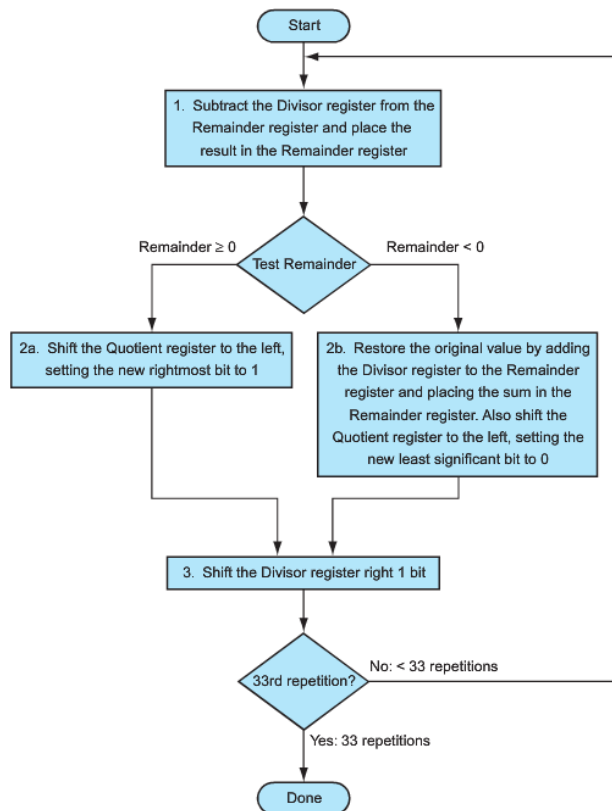
本次试验分为两个部分：第一部分、用加法器设计一个不考虑溢出的乘法器；第二部分、用加法器设计一个考虑溢出的乘法器（编程熟练的同学，也可以用除法器、浮点加法器等替代）。

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:
12
12
result:
144
  
```

第二种情况截图：

```

Terminal
please enter two numbers:
10000000
10000000
result:
276447232
  
```

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。

1.1 首先设计一个32位的乘法器： data字段

```
.data
CONTROL: .word 0x10000 #控制指令
DATA: .word 0x10008
STRING1: .asciiz "please enter two numbers:\n"
STRING2: .asciiz "result:\n"
STRING3: .asciiz "warning: result overflow\n"
NUM1: .word 0 #乘数1
NUM2: .word 0 #乘数2
SIZE: .space 20 # 栈大小
```

1.2 text字段

```
.text
main:
    daddi $sp, $zero, SIZE
    lw $a1, DATA($zero)
    lw $a2, CONTROL($zero)
    daddi $a0, $zero, STRING1
    jal coutstr

    daddi $a0, $zero, NUM1
    jal cinint
    daddi $a0, $zero, NUM2
    jal cinint
    daddi $t0, $zero, 32
    lw $t1, NUM1($zero)
    lw $t2, NUM2($zero)
loop1:
    beq $t0, $zero, loop2
    andi $t3, $t1, 1
    beq $t3, $zero, quit
    dadd $t4, $t4, $t2
quit:
    dsrl $t1, $t1, 1
    dsll $t2, $t2, 1
    daddi $t0, $t0, -1
    j loop1
loop2:
    daddi $a0, $zero, STRING2
    jal coutstr

    daddi $a0, $t4, 0
    jal coutint
```

```

coutstr:
    daddi $sp, $sp, -4
    sw $ra, ($sp)
    sw $a0, ($a1)
    daddi $t0, $zero, 4
    sw $t0, ($a2)

    lw $ra, ($sp)
    daddi $sp, $sp, 4
    jr $ra
coutint:
    daddi $sp, $sp, -4
    sw $ra, ($sp)

    sw $a0, ($a1)
    daddi $t0, $zero, 2
    sw $t0, ($a2)

    lw $ra, ($sp)
    daddi $sp, $sp, 4
    jr $ra

    halt
cinint:
    daddi $sp, $sp, -4
    sw $ra, ($sp)

    daddi $t0, $zero, 8
    sw $t0, ($a2)
    lw $t1, ($a1)
    sw $t1, ($a0)

    lw $ra, ($sp)
    daddi $sp, $sp, 4
    jr $ra

```

1.3 使用asm.exe检查程序是否正确，如下，发现没有错误

```

PS D:\WinMips\winmips64> ./asm.exe temp.s
Pass 1 completed with 0 errors
00000000      .data
00000000 0000000000010000 CONTROL: .word 0x10000 #控制步数
00000008 0000000000010008 DATA: .word 0x10008
00000010      STRING1: .asciiz "please enter two numbers:\n"
706c6561
73652065
6e746572
2074776f
206e756d
62657273

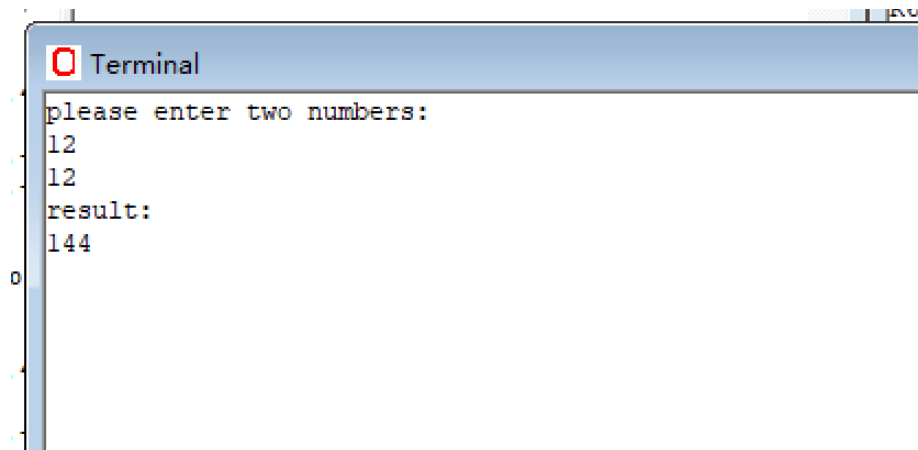
```

```

Pass 2 completed with 0 errors
Code Symbol Table
    main = 00000000
    loop1 = 00000030
    quit = 00000040
    loop2 = 00000050
    coutstr = 00000060
    coutint = 00000080
    cinint = 000000a4
Data Symbol Table
    CONTROL = 00000000
    DATA = 00000008
    STRING1 = 00000010
    STRING2 = 00000030
    STRING3 = 00000040
    NUM1 = 00000060
    NUM2 = 00000068
    SIZE = 00000070

```

1.4 实验结果



```

Terminal
please enter two numbers:
12
12
result:
144

```

```
Terminal
please enter two numbers:
10000000
10000000
result:
276447232
```

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:
12
12
result:
144
```

结果正确，其次，我们看溢出的情况结果如何：

```
Terminal
please enter two numbers:
1000000
1000000
result:
3567587328
warning: result overflow
```

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

2.1 更改上述程序，加入如下所示检测函数

```
test:
    daddi $a0, $zero, STRING2
    jal coutstr

    daddi $a0, $t4, 0
    jal coutint
```

```

dsrl $t4, $t4, 16
dsrl $t4, $t4, 16
beq $t4, $zero, halt

daddi $a0, $zero, STRING3
jal coutstr
halt:
halt

```

2.2 再使用 asm.exe 来检测程序是否正确，发现没有错误

```

PS D:\WinMips\winmips64> ./asm.exe temp1.s
Pass 1 completed with 0 errors
00000000      .data
00000000 0000000000010000 CONTROL: .word 0x10000 #繵y埜鑄困护
00000008 0000000000010008 DATA: .word 0x10008
00000010      STRING1: .asciiz "please enter two numbers:\n"
          706c6561
          73652065
          6e746572
          2074776f
          206e756d
          62657273
          3a0a00
00000030      STRING2: .asciiz "result:\n"
          72657375
          6c743a
          0a00

```

```

Pass 2 completed with 0 errors
Code Symbol Table
          main = 00000000
          loop1 = 00000030
          quit = 00000040
          loop2 = 00000050
          coutstr = 00000060
          coutint = 00000080
          test = 000000a0
          halt = 000000c4
          cinint = 000000c8
Data Symbol Table
          CONTROL = 00000000
          DATA = 00000008
          STRING1 = 00000010
          STRING2 = 00000030
          STRING3 = 00000040
          NUM1 = 00000060
          NUM2 = 00000068
          SIZE = 00000070

```

2.3 实验结果，发现成功输出溢出提示


```

R8= 00000000
Terminal
please enter two numbers:
12
12
result:
144
```

```

Terminal
please enter two numbers:
10000000
10000000
result:
3567587328
warning: result overflow
```

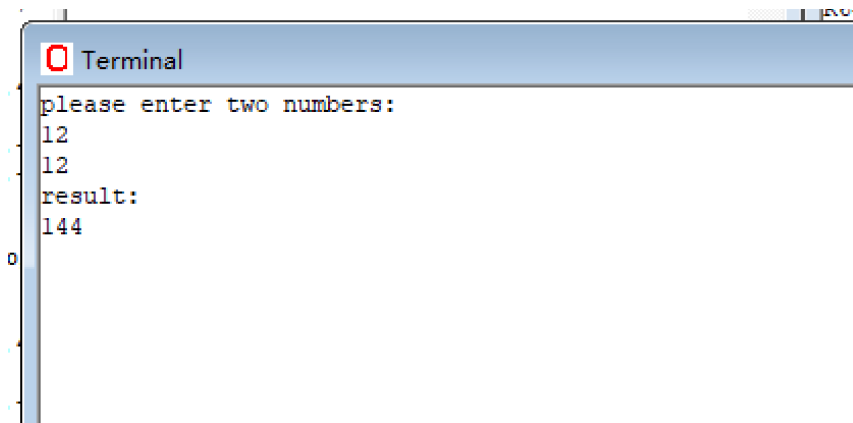
4 结束语

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。Perf 软件的使用让学生初步熟悉性能测评的主要工具。

五、实验结果

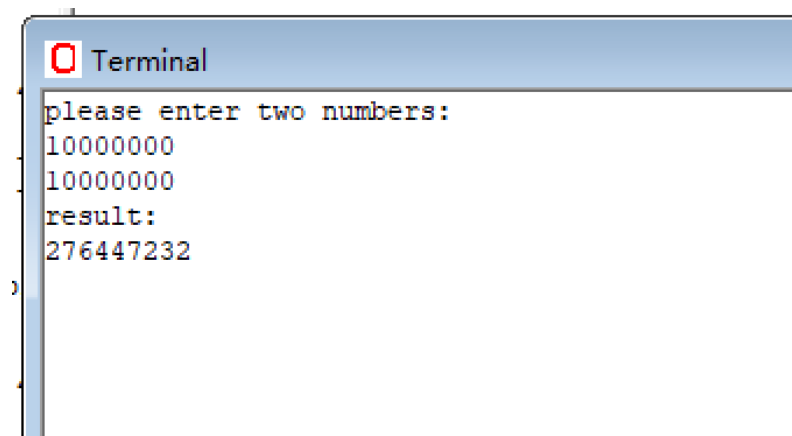
1. 无溢出检测的程序

① 无溢出的情况



```
Terminal
please enter two numbers:
12
12
result:
144
```

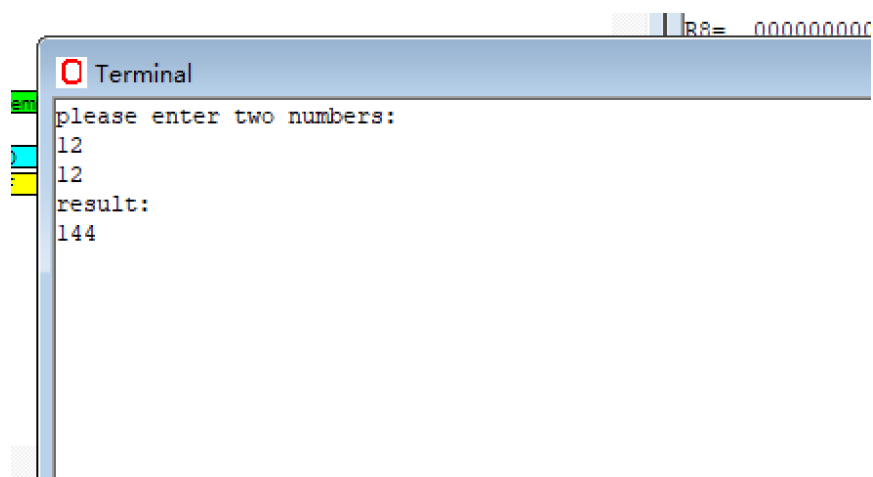
②有溢出的情况



```
Terminal
please enter two numbers:
100000000
100000000
result:
276447232
```

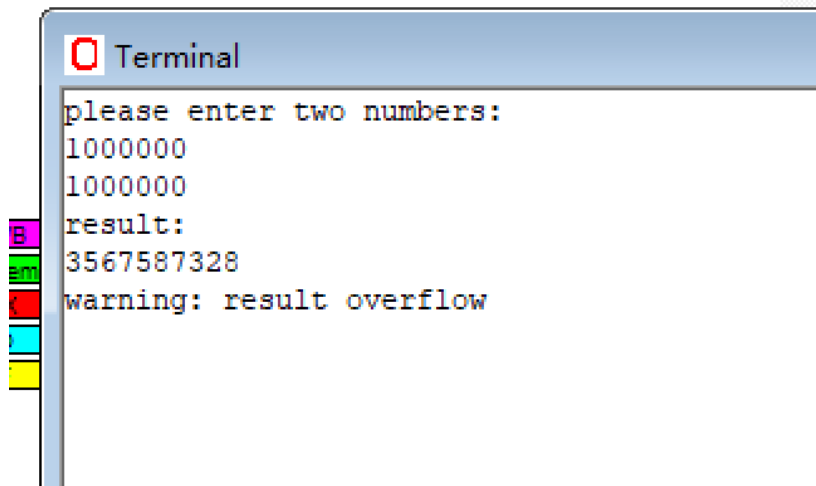
2.有溢出检测的程序

①无溢出的情况



```
Terminal
please enter two numbers:
12
12
result:
144
```

②有溢出的情况

A terminal window titled "Terminal" with a red close button. The text inside shows a simulation of MIPS64 multiplication. It prompts "please enter two numbers:", followed by the input "1000000" on two separate lines. The output shows "result:" followed by "3567587328", and then a "warning: result overflow". To the left of the terminal window is a vertical bar with five colored squares: purple, green, red, blue, and yellow.

```
Terminal
please enter two numbers:
1000000
1000000
result:
3567587328
warning: result overflow
```

五、实验总结与体会

本次 MIPS64 乘法器模拟实验旨在通过实践 WinMIPS64，深化对 MIPS64 程序语法及计算机乘法实现的理解。实验内容主要包括实现不检测溢出的乘法操作，并优化乘法操作以加入溢出检测。

首先，我们设计了一个简单的乘法操作，利用加法指令模拟乘法运算，不考虑溢出。通过编写 MIPS64 汇编程序，我们成功实现了乘法功能，并将结果存储在指定寄存器中。这一步帮助我们熟悉了 MIPS64 指令的使用及寄存器操作。

随后，我们优化了乘法操作，增加了溢出检测。通过引入条件分支指令，我们能够在乘法运算后判断结果是否溢出，并根据情况采取相应的处理措施。这一优化使我们更深入理解了乘法过程中可能出现的问题，并学会如何通过逻辑控制进行解决。

通过本次 MIPS64 乘法器模拟实验，我对计算机系统中乘法运算的实现方式有了更深的理解，也进一步熟悉了 MIPS64 汇编指令的使用和程序逻辑设计。整个实验不仅让我巩固了对 MIPS64 语法和指令的理解，还帮助我加深了对计算机硬件中基本运算原理的认识。这种理论与实践的结合，让我更加全面地看待计算机程序的执行过程，同时提高了我对底层编程的兴趣。

指导教师批阅意见:

成绩评定:

指导教师签字:

年 月 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。