《计算机组成与结构》实验报告

年级、专业、班级	2021 级计算机科学与技术(卓越)02 班	**姓名**	文红兵
实验题目	MIPS 汇编程序设计		
实验时间	2023.5.15	**实验地点**	DS1410
实验成绩		**实验性质**	□验证性 □设计性 ☑综合性
评价教师签名： □算法/实验过程正确; □源程序/实验内容提交; □程序结构/实验步骤合理; □实验结果正确; □语法、语义正确; □报告规范; 其他: 评价教师:			

报告完成时间: 2023 年 5 月 27 日

1 实验内容

假设没有浮点表示和计算的硬件，用软件方法采用仿真方式实现 IEEE 754 单精度浮点数的表示及运算功能，具体要求如下：

(1) 程序需要提供人机交互方式（字符界面）供用户选择相应的功能；

(2) 可接受十进制实数形式的输入，在内存中以 IEEE 754 单精度方式表示，支持以二进制和十六进制的方式显示输出；

(3) 可实现浮点数的加减 (或者乘除) 运算；

(4) 使用 MIPS 汇编指令，但是不能直接使用浮点指令，只能利用整数运算指令来编写软件完成。

（5）设计报告中给出程序的需求分析和关键算法的流程图，提交的代码注释比例 >40%，注释语义清晰。

2 实验设计

首先定义了一些常量，包括 IEEE754 标准下的浮点数位数、指数范围、尾数范围等等，以及一些字符串常量。

- 定义了一个 main 函数，用于接收用户输入，根据用户输入的操作符进行相应的计算，并输出结果。具体的计算过程是在 Add_function 和 Sub_function 函数中完成的。
- Add_function 函数用于实现加法运算。首先调用 Add_control 函数对符号、阶和尾数进行处理；然后根据符号位是否相同，分别进行处理：如果相同，则直接将两个数的尾数相加；如果不同，则需要进行绝对值大小的比较，然后进行尾数的加减操作。最后，将结果转化成不同进制的数值进行输出。
- Sub_function 函数用于实现减法运算。它将 num2 的符号位取反，然后调用
- Add_control 函数来处理符号、阶和尾数。最后，将结果转化成不同进制的数值进行输出。
- Add_control 函数用于处理符号、阶和尾数。首先将两个数的阶值进行比较，然后进行尾数的移位，使其小数点对齐。然后根据阶值差异进行尾数的移位和阶值的调整。最后，处理符号位，将结果存入对应寄存器中。
- Print_ret 函数用于输出不同进制的结果。首先判断是否下溢或上溢，如果是，则输出相应的异常信息；否则，将结果还原成 IEEE754 标准的浮点数表示，然后将其转化成不同进制的数值进行输出。
- binary 函数用于将结果转化成二进制。循环 32 次，每次取出结果的一个二进制位，然后将其输出。
- hex 函数用于将结果转化成十六进制。循环 8 次，每次取出结果的 4 个二进制位，然后将其转化成一个十六进制字符进行输出。
- Print_0 函数用于处理结果为零的情况，直接输出”0”。

- Again 标号用于转化为指定进制输出后回到调用函数前的指令。
- Input_error 标号用于处理输入不符合规范的情况，输出”Input error!!”，然后回到 main 函数重新输入操作符。
- Exit_function 标号用于处理用户输入操作符为 0 的情况，输出”Goodbye!”，然后结束程序。

3 实验结果

3.1 加法

```
Input num1:3.9
Input num2:4.5
Function: 0 for exit, 1 for add, 2 for sub: 1
Decimal result:8.4
Binary result:01000001000001100110011001100110
Hexadecimal result:41066666
```

```
Input num1:9.4
Input num2:6.8
Function: 0 for exit, 1 for add, 2 for sub: 1
Decimal result:16.199999
Binary result:01000001100000011001100110011001
Hexadecimal result:41819999
```

```
Input num1:0
Input num2:0
Function: 0 for exit, 1 for add, 2 for sub: 1
Decimal result:1.17549435E-38
Binary result:00000000100000000000000000000000
Hexadecimal result:00800000
```

```
Input num1:-2.3
Input num2:3
Function: 0 for exit, 1 for add, 2 for sub: 1
Decimal result:0.70000005
Binary result:00111111001100110011001100110100
Hexadecimal result:3F333334
```

```
Input num1:6.1
Input num2:-3.9
Function: 0 for exit, 1 for add, 2 for sub: 1
Decimal result:2.1999998
Binary result:01000000000011001100110011001100
Hexadecimal result:400CCCCC
```

3.2 减法

```
Input num1:6.2
Input num2:3.1
Function: 0 for exit, 1 for add, 2 for sub: 2
Decimal result:3.1
Binary result:01000000010001100110011001100110
Hexadecimal result:40466666
```

```
Input num1:6.2
Input num2:-3.5
Function: 0 for exit, 1 for add, 2 for sub: 2
Decimal result:9.7
Binary result:01000001000110110011001100110011
Hexadecimal result:411B3333
```

```
Input num1:-3.6
Input num2:-6.1
Function: 0 for exit, 1 for add, 2 for sub: 2
Decimal result:2.5
Binary result:01000000001000000000000000000000
Hexadecimal result:40200000
```

```
Input num1:3.5
Input num2:-1.2
Function: 0 for exit, 1 for add, 2 for sub: 2
Decimal result:4.7
Binary result:01000000100101100110011001100110
Hexadecimal result:40966666
```

4 MIPS 代码

```
#主函数流程：
    #读入浮点数和运算符号
    #解析符号、指数、尾数以及带偏阶指数
    #进行运算
    #输出不同进制表示的结果

# ********************************************************************************************************************************
.data       #数据段（存放于内存中）
    #数据声明，声明代码中使用的变量名：
    num1:       .space 20                           # num1数组，存放浮点数1，符号，指数，尾数，偏阶
    num2:       .space 20                           # num2数组，存放浮点数2，符号，指数，尾数，偏阶
    result:     .space 16

    String_1:       .asciiz "Input num1:\0"
    String_2:       .asciiz "Input num2:\0"
    String_3:       .asciiz "Function: 0 for exit, 1 for add, 2 for sub: \0"
    String_4:       .asciiz "Input error!!\n"
    String_5:       .asciiz "Exit\0"
    String_6:       .asciiz "Up Overflow!\n"
    String_7:       .asciiz "Down Overflow\n"
    String_8:       .asciiz "String_8 loss!\n"
    String_9:       .asciiz "Binary result:\0"
    String_10:      .asciiz "Hexadecimal result:\0"
    String_11:      .asciiz "Decimal result:\0"
    endl:           .asciiz "\n"

# ********************************************************************************************************************************
.text       #代码段

main:   #开始执行
    la      $s5,    num1                            #保存将num1的首地址
    la      $s6,    num2                            #保存将num2的首地址
    jal     Input                                   #输入函数，输入操作数并分解操作数存入num1和num2
    la      $a0,    String_3
    li      $v0,    4
    syscall                                         #输入计算功能（0退出、1加法、2减法）
    li      $v0,    5
    syscall
    li      $t0,    1
    beq     $v0,    $t0,    Add_function            #加法
    li      $t0,    2
    beq     $v0,    $t0,    Sub_function            #减法
    li      $t0,    0
    beq     $v0,    $t0,    Exit_function           #退出
    bne     $v0,    $t0,    Input_error             #输入的不是0-4

# ********************************************************************************************************************************
#输入函数，输入操作数并分解操作数存入num1和num2
Input:
    #打印"Input num1:\0"
    la      $a0,    String_1
    li      $v0,    4
    syscall
    #系统调用读取输入的浮点数，存入$f0
    li      $v0,    6
    syscall
    #将$f0中的数据存入$s1并放入内存
    mfc1    $s1,    $f0
    sw      $s1,    0($s5)
    #打印"Input num2:\0"
    la      $a0,    String_2
    li      $v0,    4
    syscall
    #系统调用读取输入的浮点数，存入$f0
    li      $v0,    6
    syscall
    #将$f0中的数据存入$s2并放入内存
    mfc1    $s2,    $f0
    sw      $s2,    0($s6)
    #将符号位存入
    andi    $t1,    $s1,    2147483648              # $s1中的num1按位与得到num1符号位（31位）
    srl     $t1,    $t1,    31
    sw      $t1,    4($s5)
    andi    $t1,    $s2,    2147483648
    srl     $t1,    $t1,    31
    sw      $t1,    4($s6)
    #将指数存入
    andi    $t1,    $s1,    2139095040              # $s1中的num1按位与得到num1指数（23~30位）
    srl     $t2,    $t1,23
    sw      $t2,    8($s5)
    andi    $t1,    $s2,    2139095040
    srl     $t3,    $t1,    23
    sw      $t3,    8($s6)
    #将尾数存入
    andi    $t1,    $s1,    8388607                 # $s1中的num1按位与得到num1尾数（0~22位）
    sw      $t1,    12($s5)
    andi    $t1,    $s2,    8388607
    sw      $t1,    12($s6)
    #将带偏阶指数存入
    addi    $t4,    $0,     127                     #偏阶127
    sub     $t1,    $t2,    $t4
```

```
    sw      $t1,    16($s5)
    sub     $t1,    $t3,    $t4
    sw      $t1,    16($s6)
    jr      $ra

# ********************************************************************************************************************************
# 加法流程：取num1、num2的符号位、阶、尾数  ->  补全尾数的整数位  ->  对阶  ->  执行加法运算  ->  输出
Add_function:
    jal     Add_control
    jal     binary
    jal     hex
    j       main                                    #本次执行完毕，跳回主函数开头

# ********************************************************************************************************************************
Add_control:
    #取num1和num2的符号位
    lw      $s0,    4($s5)                          #$s0是num1的符号位，$s1是num2的符号位
    lw      $s1,    4($s6)
    #取num1和num2的阶
    lw      $s2,    8($s5)                          #$s2是num1的阶，$s3是num2的阶
    lw      $s3,    8($s6)
    #取num1和num2的尾数
    lw      $s4,    12($s5)                         #$s4是num1的尾数，$s5是num2的尾数
    lw      $s5,    12($s6)
    #补全尾数的整数位1
    ori     $s4,    $s4,    8388608                 #将整数位1补全
    ori     $s5,    $s5,    8388608
    #对阶
    sub     $t0,    $s2,    $s3
    bltz    $t0,    Dui_jie_1
    bgtz    $t0,    Dui_jie_2
    beqz    $t0,    Start_add

# ********************************************************************************************************************************
#对阶
Dui_jie_1:  #num1的阶小于num2的阶，s2 < s3
    sub     $t0,    $s3,    $s2                     # s2 与 s3 相差的阶数
    add     $s2,    $s2,    $t0
    srlv    $s4,    $s4,    $t0
    sub     $t0,    $s2,    $s3
    beqz    $t0,    Start_add
Dui_jie_2:  #num1的阶大于num2的阶，s2 > s3
    sub     $t0,    $s2,    $s3                     # s3 与 s2 相差的阶数
    add     $s3,    $s3,    $t0
    srlv    $s5,    $s5,    $t0
    sub     $t0,    $s2,    $s3
    beqz    $t0,    Start_add

# ********************************************************************************************************************************
#此时num1、num2阶数相同，判断符号后才能相加
Start_add:
    xor     $t1,    $s0,    $s1                     #按位异或判断num1、num2符号是否相同（相同则$t1存32'b0，不同存32'b1）
    beq     $t1,    $zero,  Same_sign
    j       Diff_sign

# ********************************************************************************************************************************
#num1、num2符号相同相加
Same_sign:
    add     $t2,    $s4,    $s5                     #尾数相加后的结果即为输出的尾数，但需要先判断是否溢出
    sge     $t3,    $t2,    16777216
    bgtz    $t3,    Number_srl                      #上溢则尾数右移
    j       Print_ret
Number_srl:
    srl     $t2,    $t2,    1                       #尾数右移
    addi    $s2,    $s2,    1                       #阶数+1
    j       Print_ret

# ********************************************************************************************************************************
#num1、num2符号不同相加
Diff_sign:
    sub     $t2,    $s4,    $s5                     #符号不同的数相加相当于先相减再加符号，但可能出现尾数过大（上溢）或过小（下溢）
的情况
    bgtz    $t2,    Diff_sign1
    bltz    $t2,    Diff_sign2
    j   Print_0                                     #如果它们的绝对值相等，则结果为0，可以跳转到特殊结果输出
Diff_sign1:
    blt     $t2,    8388608,    Diff_sign11         # 尾数太小
    bge     $t2,    16777216,   Diff_sign12         # 尾数过大
    j       Print_ret                               # 既不上溢也不下溢的结果过可以直接输出
#num1、num2符号不同相加，num1尾数比num2小
Diff_sign2:
    sub     $t2,    $s5,    $s4                     #将$t2中数化为正
    xori    $s0,    $s0,    1                       #结果与num2同号
    j       Diff_sign1

# ********************************************************************************************************************************
#num1、num2符号不同相加，num1尾数比num2大，结果尾数太小
Diff_sign11:
    sll     $t2,    $t2,    1                       # 左移扩大尾数
    subi    $s2,    $s2,    1                       # 阶数-1
    blt     $t2,    8388608,    Diff_sign11
```

```
#num1、num2符号不同相加，num1尾数比num2大，结果尾数太大
Diff_sign12:
    srl     $t2,    $t2,    1                       #左移缩小尾数
    addi    $s2,    $s2,    1                       #阶数+1
    bge     $t2,    16777216,   Diff_sign12
    j       Print_ret


# ********************************************************************************************************************************
#减法可以复用加法模块（将num2符号取反即可）
Sub_function:
    lw      $t1,    4($s6)                          # num2的符号位存入$t1
    xori    $t1,    $t1,    1                       # 将num2符号位按位异或（取反）
    sw      $t1,    4($s6)
    jal     Add_control
    jal     binary
    jal     hex
    j       main

# ********************************************************************************************************************************
#op输入0时退出
Exit_function:
    la      $a0,    String_5
    li      $v0,    4
    syscall
    li      $v0,    10                              #结束程序
    syscall

# ********************************************************************************************************************************
#op不符合规范时回到main开头重新输入
Input_error:
    la      $a0,    String_4
    li      $v0,    4
    syscall
    j       main

# ********************************************************************************************************************************
# 打印不同进制的结果
Print_ret:
    # 打印十进制结果
    li      $v0,    4
    la      $a0,    String_11
    syscall
    # 判断是否下溢
    # 若小于0则原指数小于-128，即结果下溢；
    # 若大于255则原指数大于127，即结果上溢出
    blt     $s3,    0,      Down_overflow           #下溢，跳转到Down_overflow打印"Down Overflow Excption!"
    # 判断是上溢
    bgt     $s2,    255,    Up_overflow             #上溢，跳转到Up_overflow打印"Up Overflow Excption!"
    # 将结果还原回31位数据
    sll     $s0,    $s0,    31                      # 前面的处理已经将结果的符号位存入$s0，直接左移至最高位
    sll     $s2,    $s2,    23                      # 将指数位移动至相应位置
    sll     $t2,    $t2,    9                       # $t2中存放了输出的尾数，为防止尾数23位，采用先左移再右移的方式只留下0~22位的
数值
    srl     $t2,    $t2,    9
    add     $s2,    $s2,    $t2                     #符号位+指数+尾数=结果
    add     $s0,    $s0,    $s2
    mtc1    $s0,    $f12
    #输出
    li      $v0,    2
    syscall
    li      $v0,    4
    la      $a0,    endl
    syscall
    jr      $ra

#
**********************************************************************************************************************************
*****
#最终结果下溢
Down_overflow:
    la      $a0,    String_7
    li      $v0,    4
    syscall                                         #打印"Down Overflow Excption!"
    jr      $ra

# ********************************************************************************************************************************
#最终结果上溢
Up_overflow:
    la      $a0,    String_6
    li      $v0,    4
    syscall                                         # 打印"Up Overflow Excption!"
    jr      $ra

# ********************************************************************************************************************************
# 转化成二进制
binary:
    li      $v0,    4
    la      $a0,    String_9
    syscall                                         # 打印"The binary result of calculation is:"
    addu    $t5,    $s0,    $0                      # $s0中存放的IEEE754标准的计算结果
```

```
    addi    $t9,    $0,     0
bin_For:                                        # 循环32次，每次打印一个字符
    subi    $t7,    $t7,    1
    and     $t9,    $t6,    $t8
    srl     $t8,    $t8,    1
    srlv    $t9,    $t9,    $t7
    add     $a0,    $t9,    $0
    li      $v0,    1
    syscall
    beq     $t7,    $t0,    Again
    j       bin_For

# ***************************************************************************************************************************
#转化成十六进制（用4位二进制转1位十六进制即可）
hex:
    li      $v0,    4
    la      $a0,    String_10
    syscall
    addi    $t7,    $0,     8
    add     $t6,    $t5,    $0
    add     $t9,    $t5,    $0
hex_For:                                        # 循环8次，每次打印一个字符
    beq     $t7,    $0,     Again
    subi    $t7,    $t7,    1
    srl     $t9,    $t6,    28
    sll     $t6,    $t6,    4
    bgt     $t9,    9,      Print_char
    li      $v0,    1
    addi    $a0,    $t9,    0
    syscall
    j       hex_For

# ***************************************************************************************************************************
#转变为ascii码
Print_char:
    addi    $t9,    $t9,    55
    li      $v0,    11
    add     $a0,    $t9,    $0
    syscall
    j       hex_For

# ***************************************************************************************************************************
#计算结果为0的输出
Print_0:
    mtc1    $zero,  $f12
    li      $v0,    2
    syscall
    jr      $ra

# ***************************************************************************************************************************
#转化为指定进制输出后回到调用函数前的指令
Again:
    la      $a0,    endl
    li      $v0,    4
    syscall
    jr      $ra
```