重庆大学课程设计报告

课程设计题目:	MIPS 浮点运算的整数实现
学　　院:	计算机学院
专 业 班 级:	计算机科学与技术卓越 02 班、01 班
年　　级:	2019
学　　生:	李燕琴　　杨思怡
学　　号:	20195633　　20195217
完 成 时 间:	2021 年　06 月　23 日
成　　绩:	90
指 导 教 师:	钟将

重庆大学教务处制

项目	分值	优秀 $100 > x \geq 90$	良好 $90 > x \geq 70$	中等 $80 > x \geq 70$	及格 $70 > x \geq 60$	不及格 $x < 60$	评分
		参考标准					
学习态度	15	学习态度认真，科学作风严谨，严格保证设计时间并按任务书中规定的进度开展各项工作	学习态度比较认真，科学作风良好，能按期圆满完成任务书规定的任务	学习态度尚好，遵守组织纪律，基本保证设计时间，按期完成各项工作	学习态度尚可，能遵守组织纪律，能按期完成任务	学习马虎，纪律涣散，工作作风不严谨，不能保证设计时间和进度	
技术水平与实际能力	25	设计合理、理论分析与计算正确，实验数据准确，有很强的实际动手能力、经济分析能力和计算机应用能力，文献查阅能力强、引用合理、调查调研非常合理、可信	设计合理、理论分析与计算正确，实验数据比较准确，有较强的实际动手能力、经济分析能力和计算机应用能力，文献引用、调查调研比较合理、可信	设计合理，理论分析与计算基本正确，实验数据比较准确，有一定的实际动手能力，主要文献引用、调查调研比较可信	设计基本合理，理论分析与计算无大错，实验数据无大错	设计不合理，理论分析与计算有原则错误，实验数据不可靠，实际动手能力差，文献引用、调查调研有较大的问题	
创新	10	有重大改进或独特见解，有一定实用价值	有较大改进或新颖的见解，实用性尚可	有一定改进或新的见解	有一定见解	观念陈旧	
论文(计算书、图纸)撰写质量	50	结构严谨，逻辑性强，层次清晰，语言准确，文字流畅，完全符合规范化要求，书写工整或用计算机打印成文；图纸非常工整、清晰	结构合理，符合逻辑，文章层次分明，语言准确，文字流畅，符合规范化要求，书写工整或用计算机打印成文；图纸工整、清晰	结构合理，层次较为分明，文理通顺，基本达到规范化要求，书写比较工整；图纸比较工整、清晰	结构基本合理，逻辑基本清楚，文字尚通顺，勉强达到规范化要求；图纸比较工整	内容空泛，结构混乱，文字表达不清，错别字较多，达不到规范化要求；图纸不工整或不清晰	

指导教师评定成绩：

指导教师签名：

MIPS 浮点加减运算的整数实现

李燕琴、杨思怡

1 小组分工说明

李燕琴：负责方案设计和使用 Mars 编写汇编。

杨思怡：负责文献的查找和结果的整理。

2 设计方案

2.1 总体设计思路

输入两个 32 位浮点数，按照 IEEE 754[(1)] 中单精度浮点数的表示格式设计符号位、指数位、尾数位的掩码，分别与浮点数进行“与”操作，得到对应的符号位、指数位、尾数位；按照浮点数加法原理，经过指数对齐、符号位判断加减操作、结果规格化并判断指数溢出情况得到加法计算结果，最后去掉结果前导 1 并通过“或”操作组合符号位、指数位、尾数位得到最终结果二进制表达数。具体流程如图1。

2.2 主要运算模块

(1) 浮点数分解模块

IEEE 754 中单精度浮点数的表示格式如图2。通过掩码取出需要的位数，具体有 符号位：掩码需要将第 31 位设置为 1，剩余位设置为 0，即 0x80000000，通过与对应浮点数执行 and 操作，即可取出符号位。指数位：掩码设置为 0x7f800000。取出指数位后需要将其右移 23 位，便于后续计算和比较大小。尾数位：掩码设置为 0x7fffff。取出尾数位后需要恢复前导 1。代码如图3。

(2) 指数对齐模块

先比较指数大小，进而计算二者之差（设为 x），将指数小的数的位数右移 x 位，并保存大的指数，作为最终结果的指数部分。代码如图4。

(3) 有效数计算模块

本模块重点在于同号相加，异号相减，故需要先判断符号位异同（代码中的 xor 指令）。加法规格化时需要判断指数上溢情况，减法规格化时需要判断指数下溢情况后者，且减法若差为 0，按照 IEEE 754 浮点数编码格式需要特殊处理，即结果直接返回 0x00000000。代码如5所示。

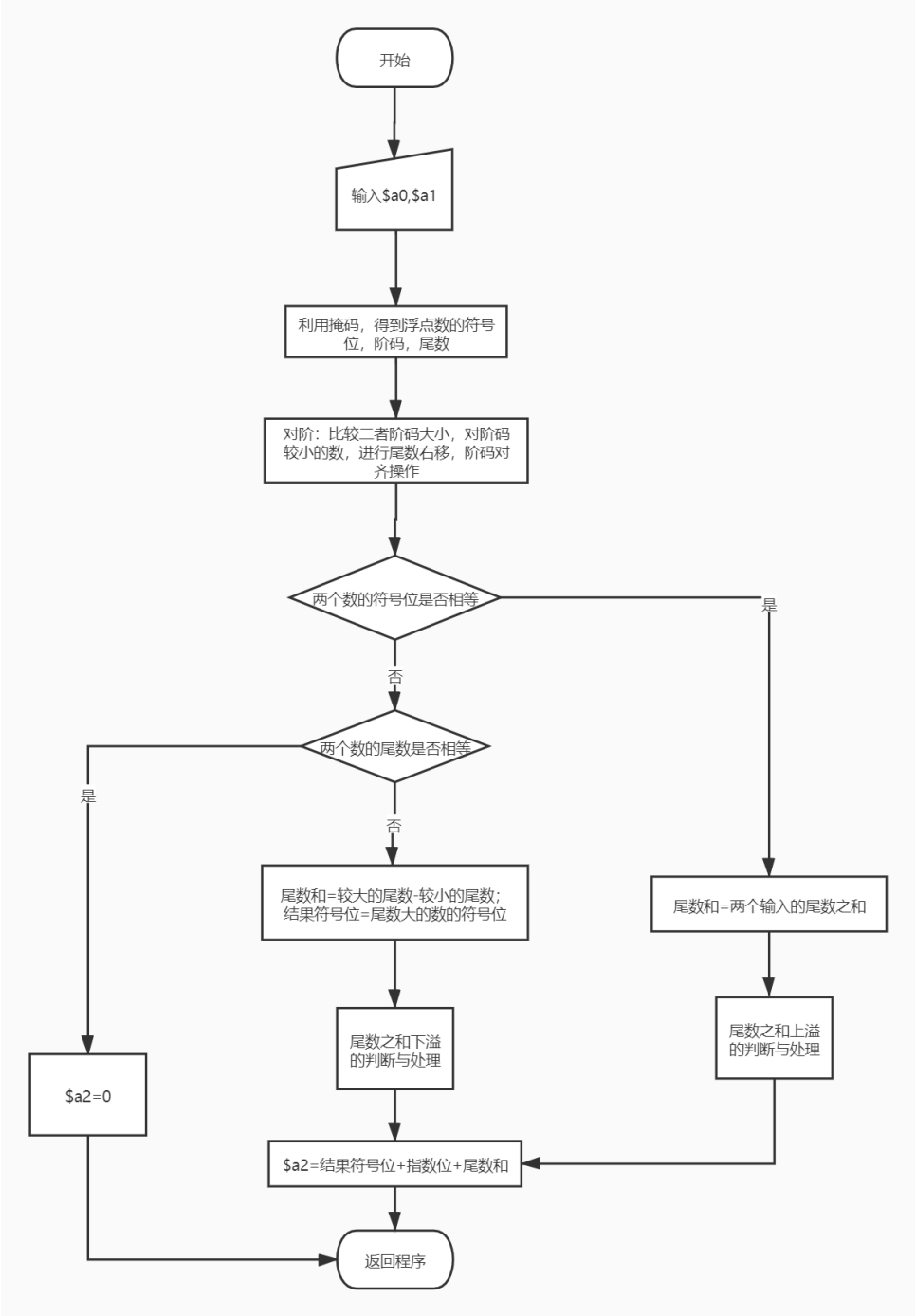

图 1: 计算流程图

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
s	指数								尾数																						
1位	8位								23位																						

图 2: 32 位浮点数格式表示

```
# 1.取指数尾数（32bit浮点数）
andi $s0,$a0,0x80000000     # a0 符号位   掩膜
andi $s1,$a0,0x7f800000     # a0指数     掩膜
srl  $s1,$s1,23             # 归右
andi $s2,$a0,0x7fffff       # a0尾数     掩膜
addi $s2,$s2,0x00800000     # 恢复前导1
```

图 3: 浮点数分解模块

```
# 2.指数对齐
move $t1,$s1                    # t1 初始化
beq  $s4,$s1,sum_branch
blt  $s1,$s4,align_a0           # $s1<$s4，处理a0
blt  $s4,$s1,align_a1           # $s4<$s1，处理a1

# a0指数对齐
align_a0:
sub  $t7,$s4,$s1
srlv $s2,$s2,$t7                # a0尾数右移
move $t1,$s4                    # a1 对齐的指数
j    sum_branch
```

图 4: 指数对齐模块

```
# 3.有效数相加
sum_branch:
xor  $t7,$s0,$s3              # 符号位判断，异或
beq  $t7,$0,sum_same          # 符号相同，直接加减
j    sum_differ

# 异号相加，即作加法
sum_same:
add  $t2,$s2,$s5               # 符号相同，绝对值直接相加 s2+s5
move $t0,$s0
j    judge_overflow

# 异号相加，即作减法
sum_differ:
beq  $s2,$s5,result_zero
blt  $s2,$s5,little_add_big
sub  $t2,$s2,$s5                # s2>=$s5
move $t0,$s0                   # a0的符号位
j    judge_underflow
little_add_big:
sub  $t2,$s5,$s2                # s2 < $s5
move $t0,$s3                   # a1的符号
j    judge_underflow
```

图 5: 有效数计算模块

(4) 尾数规格化模块

规格化时,需要保证尾数结果第 25 以上为 0,即尾数小于 0x01000000,否则尾数需要右移对应指数加 1 ;且需要保证尾数结果第 24 位为 1,即尾数大于等于 0x00800000 ,否则尾数需要左移,对应指数减 1 [(2)]。由于加法结果必然比原始标准格式内的数大,故只需进行上溢判断;同理,减法结果必然比原始标准格式内的数小,故只需进行下溢判断[(3)]。具体实现代码如图6所示。

```
# 尾数计算结果判断上溢
judge_overflow:
blt $t2,0x01000000,result  # 保证第25位以上为0
srl $t2,$t2,1    # t2 计算的结果
addi $t1,$t1,1   # t1 对齐的指数
j judge_overflow

# 尾数计算结果判断下溢
judge_underflow:
bge $t2,0x00800000,result # 保证第24位=1
sll $t2,$t2,1    # t2 计算的结果
subi $t1,$t1,1   # t1 对齐的指数
j judge_underflow
```

图 6: 尾数规格化并判断指数溢出模块

(5) 指数溢出错误模块

按照 IEEE 754 规定,指数需要在 1 至 254 之间,故如果不在该范围需要输出溢出结果。代码如图7所示。

2.3 输入模块设计

输入时,十进制输入中涉及到小数点的处理,故这里直接采用的系统[(4)] 读浮点数,再将结果转入常见的 32 位寄存器中。代码如图8所示。

2.4 输出模块设计

(1) 十进制

涉及到小数点输出的处理,这里将结果寄存器中的值转入浮点寄存器,并调用系统浮点数输出指令输出结果。代码如图9所示。

```
# 指数上溢处理
error_overflow:
la $a0,print_overflow
li $v0,4
syscall
j result_zero

# 指数下溢处理
error_underflow:
la $a0,print_underflow
li $v0,4
syscall
j result_zero
```

图 7: 指数溢出错误模块

```
# 输入num1
li $v0,6
syscall
mfc1 $t5,$f0  # 直接以IEEE754格式存储到$a0
```

图 8: 输入模块设计

```
# 十进制输出
function_outDec:
    move $t0,$s0
    la $a0,print_dec
    li $v0,4
    syscall

    mtc1 $t0,$f12    # 使用了浮点指令
    li $v0,2
    syscall
    jr $ra
```

图 9: 输出十进制

(2) 二进制

二进制输出,即从高到低逐位输出寄存器中 32 位中的各个位数。这里设置了从第 31 位开始的掩码 0x80000000,以取出该位值,并右移到最低位以 int 格式输出,此后右移掩码 1 位,继续循环输出,直至掩码为 0,代码如10所示。

(3) 十六进制

十六进制输出时,与二进制类似,不同的是每次掩码需要取出 4 位,若这四位在 0 至 9 之间,则之间按以 int 格式输出,否则需要按照 char 格式输出,其中字母 A 的 ascci 码为 65,对于数字 10 来说,需要加上偏移量 55,才能得到其字母表示,其余数字同理。此后右移掩码 4 位,继续循环输出,直至掩码为 0,代码如11所示。

3 设计结果

(1) 加法执行

如加法执行结果图12示,程序根据输入正确的计算了 1.25 与 0.5 相加,并输出了结果的十进制,二进制,以及十六进制形式。

(2) 减法执行

如减法执行结果13图示,程序根据输入正确的计算了 1.25 与 1.75 相减,并输出了结

```
# 二进制输出
function_outBinary:
    addi $sp,$sp,-4
    sw $ra,0($sp)

    move $t0,$s0
    la $a0,print_bin
    li $v0,4
    syscall

    addi $t7,$0,0           # $t7 掩膜结果数
    addi $t1,$0,32          # $t1 移位
    addi $t2,$0,0x80000000  # 1000_0000_... $t2做掩膜

    # 逐1位输出
    binaryLoop:
    and  $t7,$t0,$t2        # 掩膜结果
    srl  $t2,$t2,1          # 掩膜右移
    addi $t1,$t1,-1         # 移位数
    srlv $t7,$t7,$t1        # 结果位右移
    add  $a0,$t7,$zero      # 传参
    li   $v0,1              # 输出int
    syscall
    beq  $t1,$0,return_outBinary
    j    binaryLoop

    return_outBinary:
    lw   $ra,0($sp)
    addi $sp,$sp,4
    jr   $ra
```

图 10: 输出二进制

```
# 十六进制
function_outHex:
    addi $sp,$sp,-4
    sw $ra,0($sp)

    move $t0,$s0
    la $a0,print_hex
    li $v0,4
    syscall

    li $t7,0 # $t7 掩膜结果数
    addi $t1,$0,32 # $t1 移位
    addi $t2,$0,0xf0000000  # 1111_0000_... $t2做掩膜

    # 逐4位输出
    hexLoop:
    beq $t1,$0,return_outHex
    and  $t7,$t0,$t2 # 掩膜结果
    srl  $t2,$t2,4   # 掩膜右移
    addi $t1,$t1,-4  # 移位数
    srlv  $t7,$t7,$t1      # 结果位右移
    bgt $t7,9,outChar # 超过9需要输出char
    add  $a0,$t7,$zero     # 传参
    li $v0,1              # 输出int
    syscall
    j hexLoop

    # 输出字符串
    outChar:
    add $a0,$t7,55            #  ASCZII码表示字母 = 数字
    li $v0,11
    syscall
    j    hexLoop

    return_outHex:                  # 输出结束
    lw    $ra,0($sp)
    addi $sp,$sp,4
    jr $ra
```

图 11: 输出十六进制

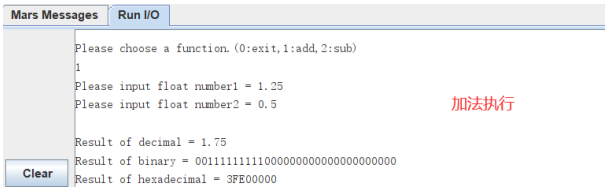

图 12: 加法执行结果

果的十进制,浮点二进制,以及十六进制形式。

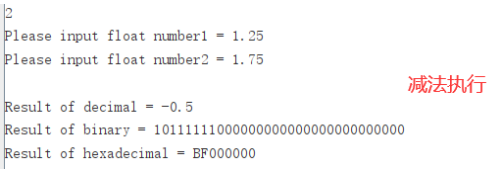

图 13: 减法执行结果

(3) 测试程序运行结果

如测试结果图14和15示,控制台输出与要求一致,加法和减法都通过了验证,设计正确,程序运行无误。

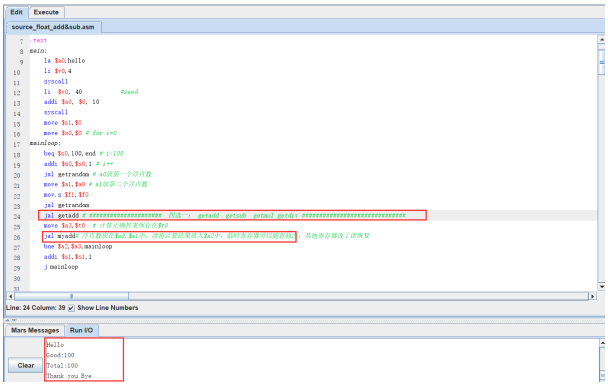

图 14: 加法验证结果图

(4) 退出程序操作

如图16示,程序根据设计预期成功退出。

(5) 非法输入处理

如图17示,程序能正确判断输入的合理性并做出提示。

4 总结

(1) 组员:李燕琴

过了一段时间再看感觉有很多冗余的代码,就像是幼儿园的过家家一样幼稚。

(2) 组员:杨思怡

文献好少好难找,浮点数好棒好奇妙。

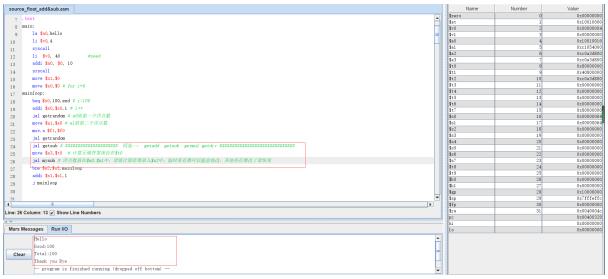

图 15: 减法验证结果图

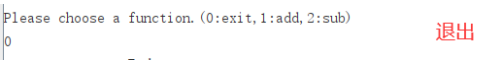

图 16: 退出程序操作

参考文献

[1] 戴维 A. 帕特森, 约翰 L. 亨尼斯著, 王党辉等译. 计算机组成与设计∶硬件/软件接口 [M]. 北京: 机械工业出版社, 2015.
[2] 谢文彬. 基于 Verilog HDL 语言的 FPGA 浮点数加减法运算的实现 [J]. 机电信息,2018(24):92-93+95.
[3] 胡同瑞, 王孝贤. 计算机规格化浮点加减法运算的研究 [J]. 齐齐哈尔医学院学报,2007(10):1232-1233.
[4] 杨羽频. 指导书 [EB/OL]. 2019[2021-06-23]. yyp@cqu.edu.cn.

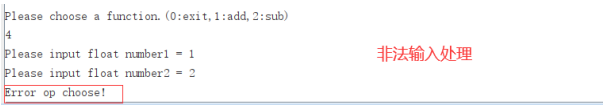
Please choose a function.(0:exit,1:add,2:sub)
4
Please input float number1 = 1
Please input float number2 = 2
Error op choose!
非法输入处理

图 17: 非法输入处理