计算机组成原理实验报告

姓名： 张佳晨　　学号：11713020

一、 实验目的

1. Learn about input and output of the floating point numbers.
2. Know the difference of storing for single and double precision numbers.
3. Learn to use the instructions for floating point numbers.

二、 实验内容

1. There are 5 students in a class , every students attend 10 lab classes and got its score(integer from 0 to 10). All the scores are stored in a two-dimensional （5*10） array. Print out the index of the lab class whose performance is not so good(the average score of the lab is lower than the total average score)
2. Calculate the square root of an integer number without using “sqrt.s” and “sqrt.d” • Get the input data and the precision value from input device • If the input data is a negative number, print out the warning message and exit • If the input data is a positive number, calculate its square root value which can satisfy the accuracy requirement and print it out

三、 实验步骤（阐述代码思路或操作步骤）

Problem 1

> - Got the total score for each lab and store in a 1-D array.
> - Got the average total score of the total labs.
> - Compare and output the labs whose performance is not so good.

```
.macro print_string(%str)
    .data
    pstr: .asciiz %str
    .text
    add $t0, $a0, $zero
    la $a0,pstr
    li $v0,4
    syscall
    add $a0, $t0, $zero
.end_macro
.macro end
    li $v0,10
    syscall
.end_macro
    .text
    .global main
    main:
    la      $a0, stu        # $s0 = stu
```

```
    jal     getScore            # get the average score of each lab
    la      $a0, score  #
    move    $a1, $v1
    jal     findLab
    end

    findLab:
        mtc1    $v1, $f1
        cvt.s.w $f1, $f1
        li  $t0, 10
        mtc1    $t0, $f2
        cvt.s.w $f2, $f2
        div.s   $f0, $f1, $f2       # $f0 is the average score of the sum of one lab

        la  $a1, score
        li  $t1, 11
        li  $a0, 0
        loop_findLab:
        addi    $t1, $t1, -1
        beq $t1, $zero  end_loop_findLab
        addi    $a0, $a0, 1
        lw  $t0, ($a1)
        addi    $a1, $a1, 4
        mtc1    $t0, $f1
        cvt.s.w $f1, $f1
        c.le.s  $f0, $f1        # #if(f0 < f1), this lab is ok, set condition flag=1
        bc1t    loop_findLab
        li  $v0, 1
        syscall
        print_string(" ")
        j   loop_findLab
    end_loop_findLab:
    jr  $ra


    getScore:                       # store the sum of each lab in the memory, store the
sum of the total score in $v1
        la      $a1, score
        addi    $sp, $sp, -8        # $sp = $sp + -4
        sw      $ra, 0($sp) #
        sw      $s1, 4($sp)

        li      $s1, 10         # $s1 = 10
        li      $v1, 0
        loop_getScore:
            jal     getLabScore # jump to getLabScore and save position to $ra
            sw      $v0, 0($a1)
            add     $v1, $v1, $v0
            addi    $a1, $a1, 4
            addi    $a0, $a0, 4
            addi    $s1, $s1, -1
            beq     $s1, $zero  end_loop_getScore
            j       loop_getScore       # jump to loop_getScore
```

```
        end_loop_getScore:
        lw      $s1, 4($sp)
        lw      $ra, 0($sp)     #
        addi    $sp, $sp, 8         # $sp = $sp + 8
        jr      $ra         # jump to $ra

    getLabScore:
        li      $t0, 5
        move    $v0, $zero
        move    $t2, $a0
        loop:
            lw      $t1, 0($t2)
            addi    $t0, $t0, -1
            add     $v0, $v0, $t1
            addi    $t2, $t2, 40        # $t0 = $t0 + 4
            beq     $t0, $zero, end_loop    # if $t2 == $zero then target
            j       loop            # jump to loop
        end_loop:
        jr      $ra         # jump to $ra

    .data
    .align  2
    stu:        .word       1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
                            1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
                            1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
                            1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
                            1, 2, 3, 4, 5, 6, 7, 8, 9, 10
    .align      2
    score:      .space       40
```

Problem 2

> - Read the precision using system call
> - Or use the default value 0.0000005, which is commented in this code.

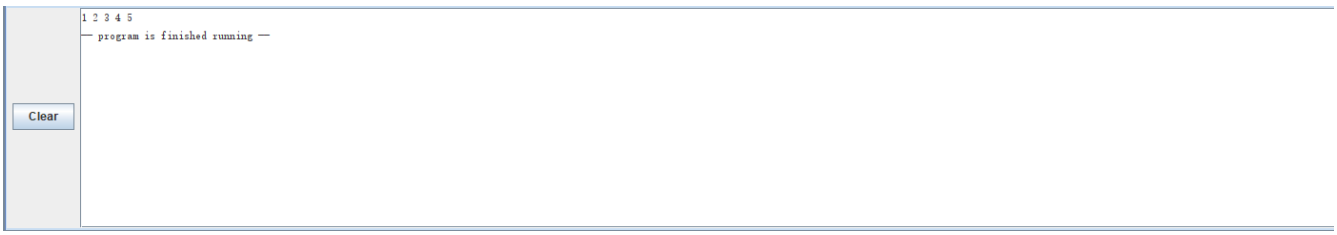
```
.macro print_string(%str)
    .data
    pstr: .asciiz %str
    .text
    add $t0, $a0, $zero
    la $a0,pstr
    li $v0,4
    syscall
    add $a0, $t0, $zero
.end_macro
.macro end
    li $v0,10
    syscall
.end_macro
    .text
    .global main
    main:
```

```
        ############################ get the precision ################################
        print_string(" > Please input the precision:\n > ")
        li  $v0, 6      # syscall to read float in $f0
        syscall
        # get the minimum difference (in $f0)  (which is the precision of float value) #
        # li    $t0, 5
        # mtc1  $t0, $f0
        # cvt.s.w   $f0, $f0
        # li    $t0, 10000000
        # mtc1  $t0, $f1
        # cvt.s.w   $f1, $f1
        # div.s $f0, $f0, $f1
        ###############################################################################

        ################## get the integer to compute square root #################
        print_string(" > Please input an integer which you want to get the square root of:
\n > ")
        li      $v0, 5
        syscall                         # get an integer number from syscall
        blt $v0, $zero, negative_error  # if $v0 < $zero then negative_error
        beq $v0, $zero, zero_or_one     # if $v0 == $zero then zero_value
        li  $t0, 1
        beq $v0, $t0, zero_or_one
        ###################### change to float value storing in $f1 ################
        mtc1    $v0, $f1
        cvt.s.w $f1, $f1
        ###############################################################################

        ############################ Using binary search##############################
        li  $t0, 1          # lo
        mtc1    $t0, $f2
        cvt.s.w $f2, $f2
        move    $t0, $v0        # hi
        mtc1    $t0, $f3
        cvt.s.w $f3, $f3
        li  $t0, 2        # hi
        mtc1    $t0, $f4
        cvt.s.w $f4, $f4
        ################# $f2 = lo, $f3 = hi, $f4 = 2.0, $f5 = mid #################
        binary_search:
            add.s $f5, $f2, $f3
            div.s $f5, $f5, $f4             # mid = (lo + hi)/2

            mul.s $f6, $f5, $f5         # $f6 = mid^2
            sub.s $f7, $f6, $f1         # $f7 = mid^2 - aim
            abs.s $f7, $f7              # $f7 is the absolute difference
            c.le.s  $f7, $f0            # if $f7 <= $f0, find the root
            bc1t    end_binary_search
            c.lt.s  $f6, $f1            # if $f6 < $f1, root may small
            bc1t    root_small
            # root_large
            mov.s  $f3, $f5
            j       binary_search
```

```
        root_small:
            mov.s  $f2, $f5
            j       binary_search               # jump to binarg_search
        end_binary_search:
            mov.s  $f12, $f5        # $f0 =$f5
            li    $v0, 2        # syscall to print float
            syscall
            end
        zero_or_one:
            move      $a0, $v0
            li      $v0, 1
            syscall
        negative_error:
            print_string(" > Error: Negative value has no real square root.\n" )
            end
```

四、 实验结果（截图并配以适当的文字说明）

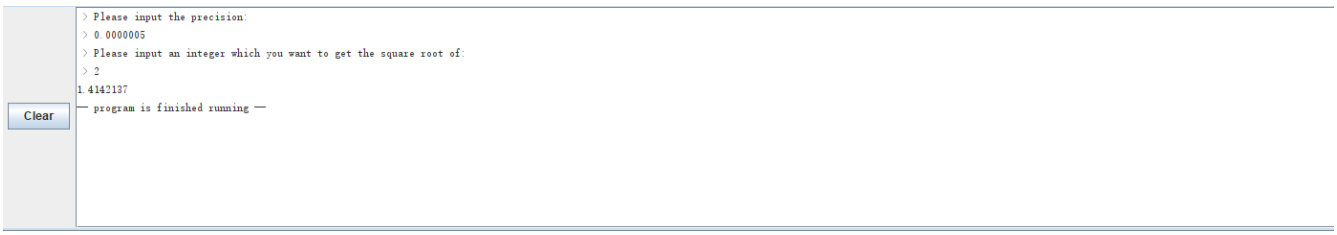
Problem 1



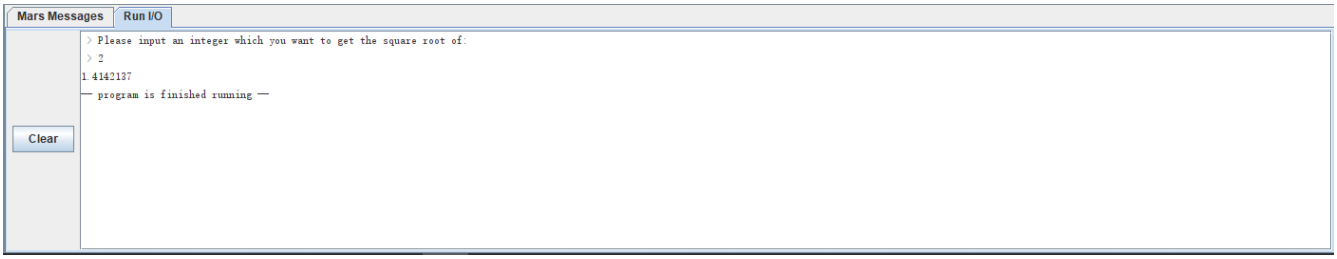
> The scores of each student got for different lab is stored in a two-dimensional （5*10）array.

```
stu:        .word       1,  2,  3,  4,  5,  6,  7,  8,  9,  10,
                        1,  2,  3,  4,  5,  6,  7,  8,  9,  10,
                        1,  2,  3,  4,  5,  6,  7,  8,  9,  10,
                        1,  2,  3,  4,  5,  6,  7,  8,  9,  10,
                        1,  2,  3,  4,  5,  6,  7,  8,  9,  10
```

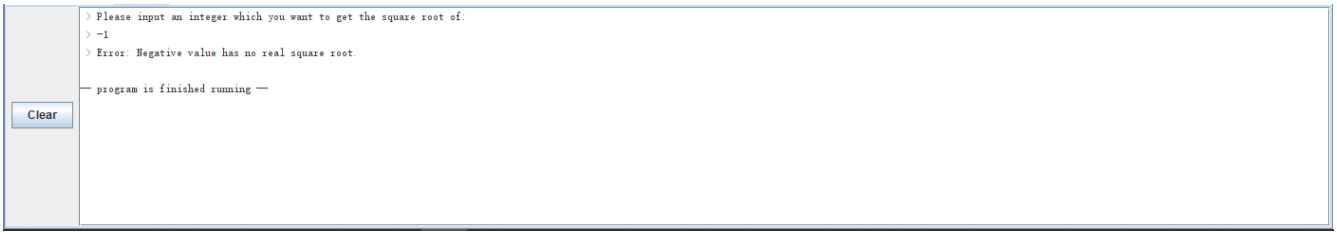
Problem 2



> Using `syscall` to get the precision.



> Using default precision (= 0.0000005).



> Error when input is negative.

五、 实验分析（遇到的问题以及解决方案）

1. It's required to use coprocessor to load the float point number without `syscall`.
2. Using the 2-D array is similar to which in C.

六、 实验小结与体会

1. It's required to use coprocessor to load the float point number without `syscall`.