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 how to implement loop using MIPS.
2. Learn about the `$sp` and the stack implementation in assembly language.

二、 实验内容

1. Print out all the `t` value that satisfies `t= x*x+y*y+z*z+xy+xz+yz` (x,y,z is a positive integer) within 400.
2. Store a string (other characters except the string terminator) in reverse order, then print it out.

三、 实验步骤（阐述代码思路或操作步骤）

Problem 1

> Using array to remove duplicated numbers.
>
> Using loop to get all the combinations of `x`, `y` and `z`.
>
> If the number in the range of `[1, 400]` can be computed, then the `array[number] = -1`, else `array[number] = 0`

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

main:
    la      $s0, numbers
    li      $s1, 0          # count
    li      $s2, -1
    li      $a0, 1          # x: $a0 = 1
    li      $a1, 1          # y: $a1 = 1
    li      $a2, 1          # z: $a2 = 1
    loop_x:
        loop_y:
            loop_z:
                jal computeA
                bge     $v0, 400, end_loop_z    # if $a0 >= 400 then end_loop_z
                add    $t0, $s0, $v0           # $t0 = addr of numbers[sum]
                sb      $s2, 0($t0)
                addi    $s1, $s1, 1             # count++
                addi    $a2, $a2, 1             # z = z + 1
                j       loop_z                  # jump to loop_z
            end_loop_z:
            addi    $a1, $a1, 1                 # y = y + 1
            move    $a2, $a1                    # z = y
            jal     computeA
            bge     $v0, 400, end_loop_y        # if $a0 >= 400 then end_loop_y
            j       loop_y
        end_loop_y:
        addi    $a0, $a0, 1                     # x = x + 1
        move    $a1, $a0                        # y = x
        move    $a2, $a0                        # z = x
        jal     computeA
        bge     $v0, 400, end_loop_x            # if $a0 >= 400 then end_loop_x
        j       loop_x
    end_loop_x:
    jal     printResult
    end

computeA:
    mult    $a0, $a0            # $a0 * $a0 = Hi and Lo registers
    mflo    $t0                 # copy Lo to $t0 = x*x

    mult    $a1, $a1            # $a1 * $a1 = Hi and Lo registers
    mflo    $t2                 # copy Lo to $t2 = y*y
    add     $t0, $t0, $t2       # $t0 = x*x + y*y

    mult    $a2, $a2            # $a2 * $a2 = Hi and Lo registers
    mflo    $t2                 # copy Lo to $t2 = z*z
    add     $t0, $t0, $t2       # $t0 = x*x + y*y + z*z

    mult    $a0, $a1            # $a0 * $a1 = Hi and Lo registers
    mflo    $t2                 # copy Lo to $t2 = x*y
    add     $t0, $t0, $t2       # $t0 = x*x + y*y + z*z + x*y

    mult    $a1, $a2            # $a1 * $a2 = Hi and Lo registers
    mflo    $t2                 # copy Lo to $t2 = y*z
    add     $t0, $t0, $t2       # $t0 = x*x + y*y + z*z + x*y + y*z

    mult    $a0, $a2            # $a0 * $a2 = Hi and Lo registers
    mflo    $t2                 # copy Lo to $t2 = x*z
    add     $t0, $t0, $t2       # $t0 = x*x + y*y + z*z + x*y + y*z + x*z

    move    $v0, $t0            # $a0 = $t0 = x*x + y*y + z*z + x*y + y*z + x*z
    jr $ra

printResult:
    li      $s3,    0
    li      $t0,    0           # index of the array
    la      $t1,    numbers     #
    addi    $t0, $t0, -1
    addi    $t1, $t1, -1
    loop_array:
        addi    $t0, $t0, 1
        addi    $t1, $t1, 1
        beq     $t0, 400, end_loop_array    # prevent overflow and return
        lb      $t2, 0($t1)
        bne     $t2, $s2, loop_array        # if $t2 == $s2(-1) then target
        addi    $s3, $s3, 1
        move    $a0, $t0                    # $a0 = $t0
        li      $v0, 1
        syscall
        print_string("\n")
        j       loop_array      # jump to loop_array
    end_loop_array:
    jr      $ra                 # jump to $ra

    .data
space:      .ascii " "
x:          .word   1
y:          .word   1
z:          .word   1
numbers:    .space      400
```

Problem 2

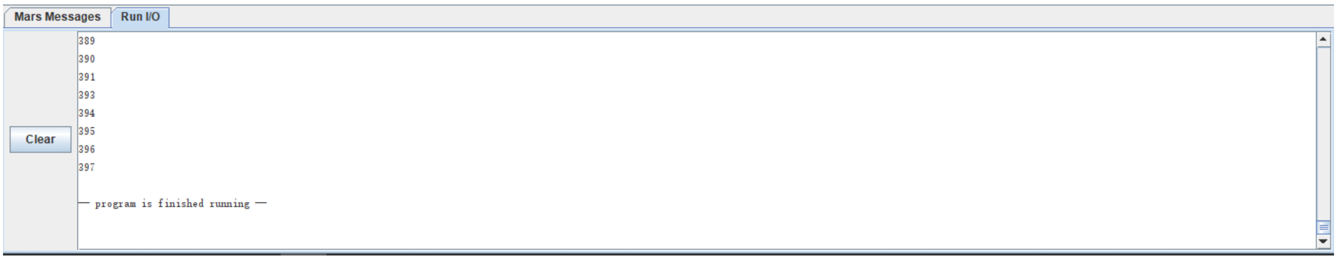
> Read the String, which length is limited and described on the screen, from `syscall` and store it in the buffer.
>
> Using iteration to store the characters of the string onto the stack until it ends.
>
> Restore the characters from the stack to the memory and we get the inverse order of the string.
>
> Using `syscall` to print the string.

```
.macro print_string(%str)
    .data
    pstr: .asciiz %str
    .text
    la $a0,pstr
    li $v0,4
    syscall
.end_macro

.macro end
    li $v0,10
    syscall
```

```
.end_macro

    .data
str:    .space 32
reverse_str:    .space  32
    .text
    .global main
    main:                           # execution starts here
        print_string("Please input a string with max length equals 30:\n>")
        li      $a1, 31             # length: $a1 = 31
        la      $a0, str            #
        li      $v0, 8              # system call to read_string
        syscall
        jal     reverse_string      # jump to reverse_string and save position to $ra
        move    $a0, $v0            # $a0 = $v0 len(str)
        jal     store_string        # jump to store_str and save position to $ra

        la      $a0, reverse_str
        li $v0,4
        syscall

        li      $v0, 4              # system call to print_string
        syscall

        end
    store_string:
        la      $t0, reverse_str    #
        loop_store_string:
        seq     $t2, $a0, $zero     # if $a0 == 0, then $t2 = 1
        bne     $t2, $zero, end_loop_store_string   # if 1 = $t2 != $zero then return
        lb      $t1, 0($sp)         # save the current byte at $t1
        addi    $sp, $sp, 4         # $sp = $sp + 4
        addi    $a0, $a0, -1        # $a0 = $a0 + -1
        sb      $t1, 0($t0)         # store the current byte
        addi    $t0, $t0, 1         # $t0 = $t0 + 1
        j loop_store_string
        end_loop_store_string:
        jr      $ra                 # jump to $ra

    reverse_string:
        li      $v0, 0              # len(str) $v0 = 0 the number of bytes
        la      $t0, str            # $t0 = addr(str)
        loop_reverse_string:
        lb      $t1, 0($t0)         # save the current at $t1 = the char at the addr($t0)
        seq     $t2, $t1, $zero     # if current byte is '\0', $t2 = 1
        bne     $t2, $zero, end_loop_reverse_string # if 1 = $t2 != $zero then return
        addi    $sp, $sp, -4        # $sp = $sp + -4
        sb      $t1, 0($sp)         # store the current byte at $sp
        addi    $v0, $v0, 1         # $v0 = $v0 + 1     len(str)++
        addi    $t0, $t0, 1         # $t0 = $t0 + 1     move to the next byte
        j       loop_reverse_string             # jump to loop_reverse_string
        end_loop_reverse_string:
        jr      $ra                 # jump to $ra
```

四、 实验结果（截图并配以适当的文字说明）

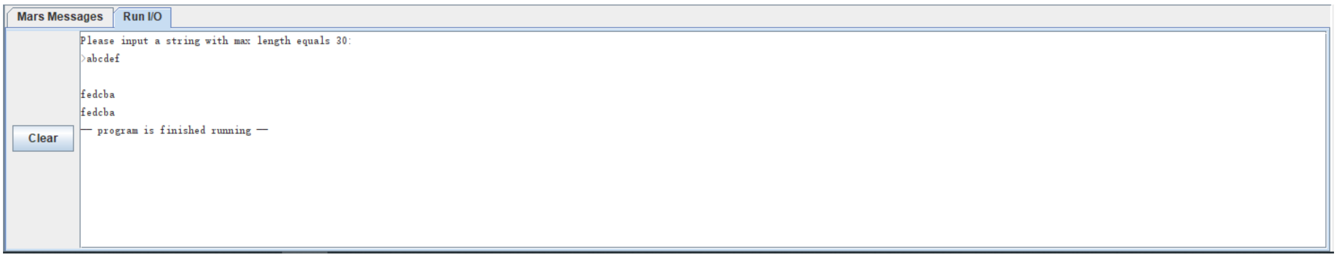
Problem 1



```
389
390
391
393
394
395
396
397

— program is finished running —
```

> I print all the summations in ascending order.

Problem 2



```
Please input a string with max length equals 30:
>abcdef

fedcba
fedcba
— program is finished running —
```

> Here I reverse the string and print it out.
>
> The string is printed "twice" because, the upper line is printed when I use `sb` instruction, and the next line is printed by system call.
>
> And there is a `\n` being printed as I used system call to read the string and it also read the `\n`.

五、 实验分析（遇到的问题以及解决方案）

> Not yet

六、 实验小结与体会

1. Using data segment should carefully malloc its size. It will be better if every data segment start with the address which is a multiple of 4.