Bài kiểm tra giữa kỳ môn thực hành KTMT

Họ và tên: Vũ Thành Trung (Câu 3 – 1 – 3)

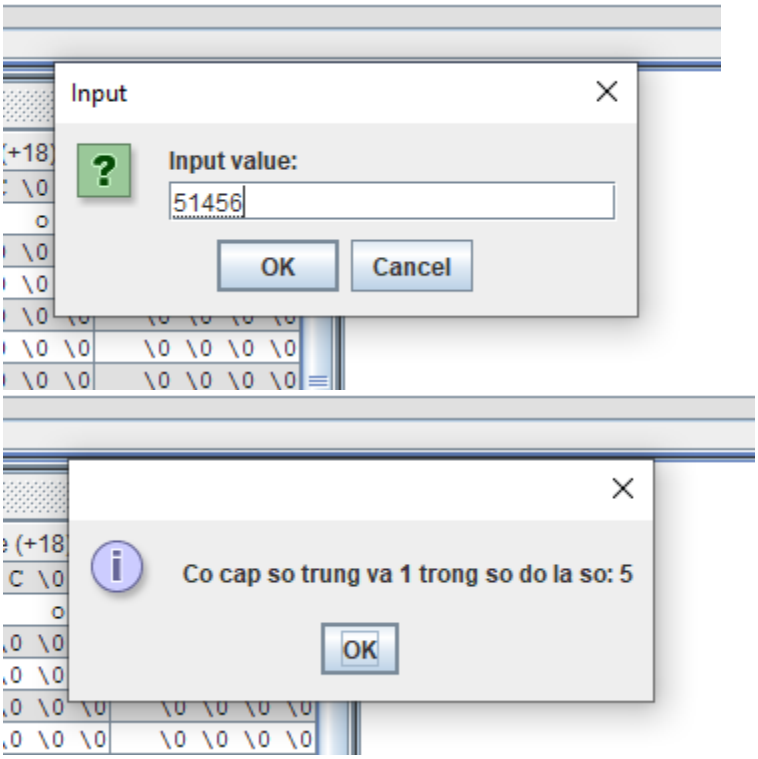
MSSV: 20200650

Bài A:

Bài 3: Nhập số nguyên dương N có từ 2 chữ số trở lên, kiểm tra N có chứa 2 chữ số trùng nhau không.

Giải thích thuật toán:

- Lấy số ban đầu (lưu ở s0) để chia 10 có số dư (lưu vào a2) và thương (lưu vào s0). Ta dùng 2 vòng loop đầu tiên check từng số 1 (là những số dư) với số dư a2, nếu k có số trùng thì chia số s0 để lấy dư và thương tiếp cho đến khi thương bằng 0 thì k thỏa mãn, nếu có trường hợp thỏa mãn thì rời vòng lặp và in ra kết quả là số tìm được



Có 2 chương trình con chính là 2 vòng lặp 1 (loop) để chia số ban đầu cho 10 (vòng lặp thứ 1) lấy thương và dư và vòng lặp thứ 2 (loop_2) để chia số thương lấy dư để so sánh với dư ban đầu.

```
.data
Message:    .asciiz "Input value:"
Error:      .asciiz     "Input error!"
Success:    .asciiz     "Co cap so trung va 1 trong so do la so: "
Failure:    .asciiz     "Value not found!"
```

```
.text
        li      $v0, 51                 # Get number
        la      $a0, Message
        syscall

        add     $s0, $a0, $zero         # Store value at $s0

        slt     $t1, $zero, $s0
        beq     $t1, $zero, error       # If $s0 is negative then show error message


        li      $a1, 10                 # We'll divide by 10 so load it first to a1
        li      $t2, 9                  # We'll divide by 9 to check whether it has more than 2
characters
        div     $s0, $a1                # Divide by 10
        mflo    $t1                     # quotient to $s0 (so chia)
        beqz    $t1, error              # If number inputted has less than 2 character than
error and exit

        j       loop

error:
        li      $v0, 55
        la      $a0, Error
        syscall

        j       end
loop:
        beqz    $s0, no_end             # end loop if quotient equals to 0
```

```
        div     $s0, $a1            # Divide by 10
        mfhi    $a2                 # remainder to $a2 (so du)
        mflo    $s0                 # quotient to $s0 (so chia)

        li      $t3, 0              # Index for 2nd loop
        add     $s1, $s0, $zero             # quotient store in $s1
        j       loop_2

loop_2:
        beqz    $s1, loop           # end loop if quotient equals to 0
        div     $s1, $a1            # divide quotient $s1 by 10
        mfhi    $s2                 # Remainder to $s2
        mflo    $s1                 # Quotient to $s1
        beq     $s2, $a2, yes_end   # If encountered, end loop
        j       loop_2              # If not, keep on looping


# If found value, print out with one of the character that is valid
yes_end:
        li      $v0, 56
        la      $a0, Success
        add     $a1, $s2, $zero
        syscall
        j       end

# If not, print out failure
no_end:
        li      $v0, 55
```

```
        la      $a0, Failure
        syscall


# End program
end:
        li      $v0, 10
        syscall
```

Bài B:

Bài 1: Nhập mảng số nguyên từ bàn phím. In ra màn hình cặp phần tử liền kề có tích nhỏ nhất.

Ý tưởng:

- Ta sẽ lưu số nguyên vào mảng array, lấy cặp phần tử liền kề đầu tiên làm mốc (start) và lưu 2 số vào s4, s5 sẽ so sánh với tích cặp phần tử sao cho thỏa mãn đề bài. Nếu thỏa mãn thì lưu kết quả và 2 số lại lần lượt vào t3, s4, s5 và tiếp tục vòng lặp cho đến khi kết thúc.

Giải thích

- Hàm start: Tạo hàm mốc và lưu 2 số đầu
- Hàm compare: Chạy vòng lặp để so sánh cho đến khi kết thúc

```
Input number of array's elements  5
Input: 2
3
4
5
6
Cap phan tu lien ke co tich be nhat la: 2 3
-- program is finished running --
```

```
.data
array: .word 0:100
Message:      .asciiz "Input number of array's elements  "
Message_2:    .asciiz "Input: "
space:                .asciiz " "
Out: .asciiz "Cap phan tu lien ke co tich be nhat la: "
.text
```

```
# Read input number of array's elements
        li      $v0, 4
        la      $a0, Message
        syscall

        li      $v0, 5
        syscall
        move    $t0, $v0        # Store value in $t0

# Insert array's elements

        li      $t1,0           # Use to count element's index
        la      $t2, array      # Load array's address into $t2

        li      $v0, 4
        la      $a0, Message_2
        syscall
Input:
        li      $v0, 5
        syscall
        sw      $v0, 0($t2)     # Save element's value into array

        addi    $t1 ,$t1 ,1     # Increment index
        addi    $t2 ,$t2 ,4     # Point to array's new address

        beq     $t1, $t0, start         # If index is equal to number of elements then we start
        j       Input

start:
```

```
        # subi  $t0, $t0, 1
        la      $t2 ,array      # Load array's address into $t2

        lw      $s1 ,0($t2)     # Load A[i]
        lw      $s2 ,4($t2)     # Load A[i+1]
        mul     $t3, $s1, $s2  # A[i]*A[i+1}
        addi    $t2, $t2, 4
        move  $s4, $s1        # Store value of first 2 integers (1)
        move  $s5, $s2        # Store value of first 2 integers (2)
        li      $t1, 1          # Increment (Equals to 1 because first one has already been
counted)
        beq     $t1, $t0 ,print         # If array has only 2 elements then return
compare:
# Same idea as start, with added comparison
        lw      $s1 ,0($t2)
        lw      $s2 ,4($t2)
        mul     $t4 ,$s1 ,$s2
        addi    $t2 ,$t2, 4
        addi    $t1 ,$t1 ,1

        slt     $s3 , $t4 , $t3
        beq     $t1 , $t0, print
        beq     $s3 , 0, compare      # If A[i]*A[i+1} new is lower than older one than swap
values
        move  $s4 ,$s1
        move  $s5 ,$s2
        move  $t3 ,$t4

        j       compare
```

```
print:
#----- Print output -----#
        la      $t3 , Out
        li      $v0, 4
        addi    $a0, $t3, 0
        syscall

        li      $v0, 1
        addi    $a0, $s4, 0
        syscall

        la      $t3 , space

        li      $v0, 4
        addi    $a0, $t3, 0
        syscall

        li      $v0, 1
        addi    $a0, $s5, 0
        syscall



        li      $v0, 4
        addi    $a0, $t3, 0
        syscall
#----- Print output -----#
        li      $v0, 10         # end program
        syscall
```

Bài C: (3)

- Ý tưởng: Check từng ký tự 1 rồi những ký tự là chữ Latinh thì sẽ viết hoa (hoặc viết thường). Chạy cho đến khi hết chuỗi rồi in

```
-- program is finished running --

Enter string: erajdkasl*()*)(312312ADASDASD
Transformed: ERAJDKASL*()*)(312312adasdasd

-- program is finished running --
```

Loop: Load từng byte một (Tương ứng với 1 ký tự) để check nếu lowercase thì đổi không thì về loop_2

Not_lower: nếu viết hoa thì viết thường, không thì về loop_2

Loop_2: tăng index, tiếp tục vòng lặp

```
.data
buffer:        .space        100
Message:       .asciiz "Enter string: "
Message_2:     .asciiz "Transformed: "

.text
main:
        la      $a0, Message        # Load and print string asking for string
        li      $v0, 4
        syscall

        li      $v0, 8              # take in input
        la      $a0, buffer         # load byte space into address
        li      $a1, 100            # allot the byte space for string
        syscall
        move    $s0, $a0            # save string to s0
```

```
        li      $v0, 4

#Loop to capitalize and to lowercase
loop:
        lb      $t1, ($a0)              #Load byte from 't0'th position in buffer into $t1
        beq     $t1, 0, exit            #If ends, exit
        blt     $t1, 'a', not_lower     #If less than a, exit
        bgt     $t1, 'z', not_lower     #If greater than z, exit
        sub     $t1, $t1, 32            #If lowercase, then subtract 32
        sb      $t1, ($a0)              #Store it back to nth position in buffer
        j       loop_2

#Lowercase letter
not_lower:
        blt     $t1, 'A', loop_2        #If less than A, exit
        bgt     $t1, 'Z', loop_2        #If greater than Z, exit
        add     $t1, $t1, 32            #If uppercase, then subtract 32
        sb      $t1, ($a0)              #Store it back to nth position in buffer

loop_2:
#if not alphabetical character, then increment $a0 and continue
        addi    $a0, $a0, 1
        j       loop

exit:
        la      $a0, Message_2      # load and print new string
        li      $v0, 4
        syscall
```

```
move   $a0, $s0              # primary address = s0 address (load pointer)
li     $v0, 4                # print string
syscall
li     $v0, 10               # end program
syscall
```