TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Final Project
Computer Architecture

GVHD: ThS. Lê Bá Vui
Nhóm: 02
Thành Viên: Lê Anh Dũng - 20194522
Nguyễn Tiến Việt - 20194718
Mã Lớp: 130939

MỤC LỤC

LỜI MỞ ĐẦU

Nhóm 02 của bọn em gồm 2 thành viên Nguyễn Tiến Việt và Lê Anh Dũng

Bản báo cáo khái quát quá trình thực hiện 2 bài tập lớn là bài 3 và bài 6 gồm các nội dung chính:

- ❖ Đề bài.
- ❖ Phân tích cách làm.
- ❖ Mã nguồn.
- ❖ Hình ảnh kết quả minh họa

Bản báo cáo sẽ không tránh khỏi những sai sót. Nhóm rất mong nhận được ý kiến góp ý của thầy giáo và các bạn.

Chúng em xin chân thành cảm ơn.

I. Bài 6 : Hàm cấp phát bộ nhớ malloc()

Sinh viên thực hiện : Nguyễn Tiến Việt

1) Đề bài

Chương trình cho bên dưới là hàm malloc(), kèm theo đó là ví dụ minh họa, được viết bằng hợp ngữ MIPS,để cấp phát bộ nhớ cho một biến con trỏ nào đó. Hãy đọc chương trình và hiểu rõ nguyên tắc cấp phát bộ nhớ động.
Trên cơ sở đó, hãy hoàn thiện chương trình như sau: (Lưu ý, ngoài viết các hàm đó, cần viết thêm một số
ví dụ minh họa để thấy việc sử dụng hàm đó như thế nào)
1) Việc cấp phát bộ nhớ kiểu word/mảng kiểu word có 1 lỗi, đó là chưa bảo đảm qui tắc địa chỉ của
kiểu word phải chia hết cho 4. Hãy khắc phục lỗi này.
2) Viết hàm lấy giá trị của biến con trỏ.
3) Viết hàm lấy địa chỉ biến con trỏ.
4) Viết hàm thực hiện copy 2 con trỏ xâu kí tự.
5) Viết hàm giải phóng bộ nhớ đã cấp phát cho các biến con trỏ
6) Viết hàm tính toàn bộ lượng bộ nhớ đã cấp phát.
7) Hãy viết hàm malloc2 để cấp phát cho mảng 2 chiều kiểu .word với tham số vào gồm:
a. Địa chỉ đầu của mảng
b. Số dòng
c. Số cột
8) Tiếp theo câu 7, hãy viết 2 hàm getArray[i][j] và setArray[i][j] để lấy/thiết lập giá trị cho phần tử ở dòng i cột j của mảng.

Phân tích đề bài

- Hiển thị ra màn hình menu các yêu cầu của đề bài.
- Hiểu rõ nguyên tắc cấp phát bộ nhớ động.
- Viết các hàm cấp phát bộ nhớ động (malloc) cho các kiểu biến con trỏ.

2) Phân tích cách làm

1. Sửa lỗi cấp phát kiểu .word

   Khi người dùng lựa chọn cấp phát con trỏ kiểu word (WordPtr) thì sẽ kiểm tra xem địa chỉ còn trống đầu tiên có chia hết cho 4 không?
   Nếu chia hết thì thực hiện cấp phát như bình thường.
   Nếu dư thì cần cộng thêm vào địa chỉ này (4 – số dư) bytes, để lên địa chỉ mới chia hết cho 4
   VD: Địa chỉ bắt đầu cấp phát là 0x90000007 chia 4 dư 3. Cần cộng thêm (4-3) = 1 byte. Địa chỉ mới sẽ là 0x90000008. Thực hiện cấp phát tiếp như bình thường.

```
malloc:
        la      $t9, Sys_TheTopOfFree
        lw      $t8, 0($t9)                 # Lay dia chi dau tien con trong
        li      $t1, 4                      # Do dai 1 word nho
        bne     $a2, $t1, continue          # Neu khong phai cap phat kieu WORD thi OK
        divu    $t8, $t1                    # Kiem tra dia chi bat dau cap phat co chia het cho 4 khong
        mfhi    $t2                         # Luu phan du (remainder) vao $t2
        beqz    $t2, continue               # So du = 0 -> Kich thuoc hop le
        sub     $t3, $t1, $t2               # So du != 0, can cap phat them (4-sodu) bits
        add     $t8, $t8, $t3               # tang gia tri thanh ghi len so chia het cho 4
continue:
        sw      $t8, 0($a0)                 # Cat dia chi do vao bien con tro
        addi    $v0, $t8, 0                 # Dong thoi la ket qua tra ve cua ham
        mul     $t7, $a1, $a2               # Tinh kich thuoc cua mang can cap phat
        add     $t6, $t8, $t7               # Tinh dia chi dau tien con trong
        sw      $t6, 0($t9)                 # Luu tro lai dia chi dau tien do vao bien Sys_TheTopOfFree
```

Hàm malloc() sau khi sửa

2. Viết hàm lấy giá trị của biến con trỏ
   Truy cập địa chỉ của các biến con trỏ bằng lệnh la và thực hiện lệnh lw để lấy ra giá trị tương ứng. Gọi Syscall 34 để in giá trị đó dưới dạng mã Hex.
3. Viết hàm lấy địa chỉ của biến con trỏ:
   Chỉ cần truy cập địa chỉ các biến con trỏ bằng lệnh la và in ra màn hình với Syscall 34.
4. Viết hàm thực hiện copy 2 con trỏ xâu kí tự
   Cho phép người dùng nhập xâu kí tự từ bàn phím (giới hạn 100 kí tự), lưu vào vùng nhớ trỏ bởi con trỏ CharPtr1.
   Cho con trỏ CharPtr2 trỏ đến địa chỉ đầu tiên còn trống để thực hiện cấp phát.
   Lưu lần lượt từng kí tự trong xâu kí tự được trỏ bởi CharPtr1 sang vùng nhớ được trỏ bởi CharPtr2. Số kí tự đếm được (độ dài xâu) chính là lượng bộ nhớ cần cấp phát cho con trỏ CharPtr2.
5. Giải phóng bộ nhớ
   Đặt giá trị các con trỏ về 0 , đồng thời khởi động lại vùng cấp phát động

```
case5:                                  # Giai phong bo nho
        li      $s6 , 0
        la      $a0, CharPtr
        sw      $s6, 0($a0)
        la      $a0, BytePtr
        sw      $s6, 0($a0)
        la      $a0, WordPtr
        sw      $s6, 0($a0)
        la      $a0, TwoDArrayPtr
        sw      $s6, 0($a0)
        jal     SysInitMem                      # Khoi tao lai vung cap phat dong
        la      $a0, free_success
        li      $v0, 4
        syscall
        j       main
```

6. Viết hàm tính toàn bộ lượng bộ nhớ đã cấp phát cho các biến động
   Tổng lượng bộ nhớ đã sử dụng cho việc cấp phát = (Địa chỉ đầu tiên còn trống – địa chỉ vùng nhớ dùng để cấp phát)= Sys_TheTopOfFree – Sys_MyFreeSpace

```
MemoryCount:
        la      $t9, Sys_TheTopOfFree
        lw      $t9, 0($t9)                     # $t9 = Gia tri tai dia chi con trong dau tien
        la      $t8, Sys_MyFreeSpace            # Sys_MyFreeSpace luon co dinh la thanh ghi ngay sau Sys_TheTopOfFree
        sub     $v0, $t9, $t8                   # $v0 = luong bo nho da cap phat
        jr      $ra
```

7. Viết hàm Malloc2 để cấp phát cho mảng 2 chiều kiểu .word
   Có thể coi mảng 2 chiều kích thước m*n là mảng 1 chiều với (m*n) phần tử
   Bên trong hàm malloc2 chỉ cần gọi đến hàm malloc với tham số đầu vào là (m*n) phần tử, mỗi phần tử 4 bytes (1 word)
8. Viết 2 hàm Get / Set cho mảng 2 chiều trên
   Phần tử A[i][j] có thể được truy cập theo cộng thức: index = i * ncols + j
   nrows, ncols là số hàng, cột của mảng. Nếu i, j vượt quá giá trị này hay âm sẽ báo lỗi.
   Sau khi Set xong giá trị thì sẽ in ra vị trí.

3) Mã nguồn

```
# Bai 6 - Final Project
.data
#-----------Pointer-----------
    CharPtr:        .word 0         # Bien con tro, tro toi kieu asciiz
    BytePtr:        .word 0         # Bien con tro, tro toi kieu Byte
    WordPtr:        .word 0         # Bien con tro, tro toi kieu Word
    TwoDArrayPtr:   .word 0         # Bien con tro, tro toi mang hai chieu kieu Word
    CharPtr1:       .word 0         # Bien con tro, su dung trong copy xau
    CharPtr2:       .word 0         # Bien con tro, su dung trong copy xau
#-----------Menu String-----------
    menu:           .asciiz "               Menu\n Vui long chon tu 1->8.\n Chon nut bat ki khac de thoat.\n1. Cap phat bo nho.\n2. Tra ve gia tri cua cac bien con tro.\n3. Tra ve dia chi cua cac bien con tro.\n4. Copy 2 con tro xau ki tu.\n5. Giai phong bo nho.\n6. Tinh toan luong bo nho da cap phat.\n7. Malloc2 (2D Array).\n8. setArray[i][j] va getArray[i][j]."
    malloc_menu:    .asciiz "1. CharPtr\n2. BytePtr\n3. WordPtr\n4. Return main menu\n\nNhan nut khac: Exit"
    getset_menu:    .asciiz "1. SetArray[i][j]\n2. GetArray[i][j]\n3. Return main menu\n\nNhan nut khac: Exit"
    char_str:       .asciiz   "\nNhap so phan tu cua mang kieu Char :"
    byte_str:       .asciiz "\nNhap so phan tu cua mang kieu Byte :"
    word_str:       .asciiz "\nNhap so phan tu cua mang kieu Word :"
    copy_str:       .asciiz "\nXau da duoc copy la : "
    nb_row:         .asciiz "\nNhap so hang cua mang :"
    nb_col:         .asciiz "\nNhap so cot cua mang :"
    input_row:      .asciiz "\nNhap i (so thu tu hang) :"
    input_col:      .asciiz "\nNhap j (so thu tu cot) :"
    input_val:      .asciiz "\nNhap gia tri gan cho phan tu cua mang :"
    output_val:     .asciiz "\nGia tri tra ve: "
    charPtr_add:    .asciiz "\nCharPtr address: "
    bytePtr_add:    .asciiz "\nBytePtr address: "
    wordPtr_add:    .asciiz "\nWordPtr address: "
    arrayPtr_add:   .asciiz "\n2DArrayPtr address: "
    charPtr_val:    .asciiz "\nCharPtr value: "
    bytePtr_val:    .asciiz "\nBytePtr value: "
    wordPtr_val:    .asciiz "\nWordPtr value: "
    arrayPtr_val:   .asciiz "\n2DArrayPtr value: "
    malloc_str:     .asciiz "\nBo nho da cap phat(Tinh ca byte cap de chuyen dia chi): "
```

```
    bytes_str:          .asciiz " bytes"
    input_str:          .asciiz "\nNhap vao xau ky tu: "
#-----------Success Messages-----------
    malloc_success: .asciiz "\nCap phat bo nho thanh cong."
    free_success:       .asciiz "\nGiai phong bo nho thanh cong."
    set_success:        .asciiz "\nThem phan tu vao mang thanh cong.Vi tri : "
    address:                    .asciiz "\nAddress: "
#-----------Error Messages-----------
    bound_error:        .asciiz "\nError: Ngoai pham vi cua mang"
    null_error:         .asciiz "\nError: Chua khoi tao mang"
    overflow_error:  .asciiz "\nError: Gia tri input qua lon (> 2000)"
    negative_error:   .asciiz   "\nError: Gia tri input phai lon hon 0"
    zero_error:         .asciiz "\nError: Gia tri input phai khac 0"
#-----------Symbol Messages-----------
    arrow:              .asciiz " --> "
    left_bracket:       .asciiz "["
    right_bracket:      .asciiz "]"
    brackets:           .asciiz "]["
    string_copy:        .space  100     # Xau copy

.kdata
    # Luu gia tri la dia chi dau tien cua vung nho con trong
    Sys_TheTopOfFree:       .word 1
    # Vung khong gian tu do, dung de cap phat bo nho cho cac bien con tro
    Sys_MyFreeSpace:

.text
    # Khoi tao vung nho cap phat dong
    jal     SysInitMem

main:
print_menu:
    la      $a0, menu
    jal     integer_input               # Nhan tu ban phim
    move    $s0, $a0            # switch case
    beq     $s0, 1, case1
    beq     $s0, 2, case2
    beq     $s0, 3, case3
    beq     $s0, 4, case4
    beq     $s0, 5, case5
    beq     $s0, 6, case6
    beq     $s0, 7, case7
    beq     $s0, 8, case8
    j       end                         # Neu khac 1->11 => end
case1:
    la      $a0, malloc_menu        # 3 lua chon tuong ung voi yeu cau 1
    li      $v0, 51
    syscall
    move    $s0, $a0                    # switch case
    beq     $s0, 1, case1.1   # Malloc CharPtr
    beq     $s0, 2, case1.2   # Malloc BytePtr
    beq     $s0, 3, case1.3   # Malloc WordPtr
    beq     $s0, 4, main                # return menu
    j       end
```

```
case1.1:                                    # Cap phat bien con tro Char, moi phan tu 1 byte
    la      $a0, char_str
    jal     integer_input
    jal     check_input             # check_input (0 < input < 2000)
    move    $a1, $a0        # $a1 = so phan tu mang
    la      $a0, CharPtr            # $a0 = dia chi cua CharPtr
    li      $a2, 1                  # $a2 = kich thuoc Char = 1 byte
    jal     malloc                  # Cap phat bo nho
    j       main

case1.2:                                    # Cap phat bien con tro Byte, moi phan tu 1 byte
    la      $a0, byte_str
    jal     integer_input
    jal     check_input             # check_input (0 < input < 2000)
    move    $a1, $a0        # $a1 = so phan tu cua mang
    la      $a0, BytePtr            # $a0 = dia chi cua BytePtr
    li      $a2, 1                  # $a2 = kich thuoc Byte = 1 byte
    jal     malloc                  # Cap phat bo nho
    j       main


case1.3:                                    # Cap phat bien con tro Word, moi phan tu 4 byte
    la      $a0, word_str
    jal     integer_input
    jal     check_input             # check_input (0 < input < 2000)
    move    $a1, $a0        # $a1 = so phan tu mang
    la      $a0, WordPtr            # $a0 = dia chi cua WordPtr
    li      $a2, 4                  # $a2 = kich thuoc Word = 4 bytes
    jal     malloc                  # Cap phat bo nho
    j       main

case2:

    la      $a0, charPtr_val
    li      $v0, 4
    syscall
    la      $a0, CharPtr            # CharPtr value
    jal     Ptr_val

    la      $a0, bytePtr_val
    li      $v0, 4
    syscall
    la      $a0, BytePtr            # BytePtr value
    jal     Ptr_val

    la      $a0, wordPtr_val
    li      $v0, 4
    syscall
    la      $a0, WordPtr            # WordPtr value
    jal     Ptr_val

    la      $a0, arrayPtr_val
    li      $v0, 4
    syscall
```

```
    la      $a0, TwoDArrayPtr       # TwoDArrayPtr value
    jal     Ptr_val
    j       main

case3:

    la      $a0, charPtr_add
    li      $v0, 4
    syscall
    la      $a0, CharPtr            # CharPtr address
    jal     Ptr_add

    la      $a0, bytePtr_add
    li      $v0, 4
    syscall
    la      $a0, BytePtr            # BytePtr address
    jal     Ptr_add

    la      $a0, wordPtr_add
    li      $v0, 4
    syscall
    la      $a0, WordPtr            # WordPtr address
    jal     Ptr_add

    la      $a0, arrayPtr_add
    li      $v0, 4
    syscall
    la      $a0, TwoDArrayPtr       # TwoDArrayPtr address
    jal     Ptr_add
    j       main
case4:
input_string:
    li      $v0, 54                     # Input
    la      $a0, input_str
    la      $a1, string_copy        # Dia chi luu string dung de copy
    li      $a2, 100                    # So ki tu toi da = 100
    syscall
    la      $a1, string_copy
    la      $s1, CharPtr1               # Load dia chi cua CharPtr1
    sw      $a1, 0($s1)                 # Luu string vua nhap vao CharPtr1
    la      $a0, copy_str
    li      $v0, 4
    syscall
copy:
    la      $a0, CharPtr2               # Load dia chi cua CharPtr2
    la      $t9, Sys_TheTopOfFree
    lw      $t8, 0($t9)                 # Lay dia chi dau tien con trong
    sw      $t8, 0($a0)                 # Cat dia chi do vao bien con tro
    lw      $t4, 0($t9)                 # Dem so luong ki tu trong string
    lw      $t1, 0($s1)                 # Load gia tri con tro CharPtr1
    lw      $t2, 0($a0)                 # Load gia tri con tro CharPtr2
copy_loop:
    lb      $t3, 0($t1)             # Load 1 ki tu (tren cung) cua $t1 vao $t3
    sb      $t3, 0($t2)             # Luu 1 ki tu cua $t3 vao $t2
```

```
    addi    $t4, $t4, 1             # so luong ki tu trong string += 1
    addi    $t1, $t1, 1                 # Chuyen sang dia chi ki tu tiep theo cua CharPtr1
    addi    $t2, $t2, 1                 # Chuyen sang dia chi ki tu tiep theo cua CharPtr2
    beq     $t3, '\0', end_copy         # Check null = end string
    j       copy_loop
end_copy:
    sw      $t4, 0($t9)             # Kich thuoc cap phat = do dai string
    lw      $a0, 0($a0)                 # Lay noi dung con tro CharPtr2
    li      $v0, 4                  # In ra xau da copy
    syscall
    j       main

case5:                                  # Giai phong bo nho
    li      $s6 , 0
    la      $a0, CharPtr
    sw      $s6, 0($a0)
    la      $a0, BytePtr
    sw      $s6, 0($a0)
    la      $a0, WordPtr
    sw      $s6, 0($a0)
    la      $a0, TwoDArrayPtr
    sw      $s6, 0($a0)
    jal     SysInitMem              # Khoi tao lai vung cap phat dong
    la      $a0, free_success
    li      $v0, 4
    syscall
    j       main
case6:                                  # Tinh luong bo nho da cap phat
    la      $a0, malloc_str
    li      $v0, 4
    syscall
    jal     MemoryCount             # tinh luong bo nho da cap phat va luu vao $v0
    move    $a0, $v0
    li      $v0, 1                  # print integer
    syscall
    la      $a0, bytes_str
    li      $v0, 4
    syscall
    j       main

case7:                                  # Cap phat bo nho cho mang 2 chieu Malloc2
    la      $a0, nb_row
    jal     integer_input           # Nhap vao so hang
    jal     check_input
    move    $s4, $a0
    la      $a0, nb_col
    jal     integer_input           # Nhap vao so cot
    jal     check_input
    move    $s5, $a0
    move    $a1, $s4                # $a1 = so hang
    move    $a2, $s5                # $a2 = so cot
    la      $a0, TwoDArrayPtr
    jal     Malloc2                 # Cap phat bo nho cho mang 2 chieu
```

```
    j       main
case8:
    la      $a0, getset_menu        # 2 lua chon get/set tuong ung yeu cau 8
    li      $v0, 51
    syscall
    move    $s0, $a0                # switch case
    beq     $s0, 1, case8.1   # setArray[i][j]
    beq     $s0, 2, case8.2   # getArray[i][j]
    beq     $s0, 3, main            # return ti main menu
    j       end
case8.1:                                    # Set[i][j]
    la      $a0, TwoDArrayPtr
    lw      $s1, 0($a0)
    beqz    $s1, nullptr                    # if *ArrayPtr==0 => null error
    la      $a0, input_row
    jal     integer_input                   # Nhap vao hang
    move    $s2, $a0
    la      $a0, input_col
    jal     integer_input                   # Nhap vao cot
    move    $s3, $a0
    la      $a0, input_val
    jal     integer_input                   # Nhap gia tri can set
    move    $a3, $a0                # $a3 = gia tri can set
    move    $a1, $s2                # $a1 = so thu tu hang
    move    $a2, $s3                # $a2 = so thu tu cot
    move    $a0, $s1
    jal     SetArray
    la      $a0, set_success
    li      $v0, 4                          # In ra thong bao set thanh cong va vi tri
    syscall
    la      $a0, left_bracket
    li      $v0, 4
    syscall
    move    $a0, $a1
    li      $v0, 1
    syscall
    la      $a0, brackets
    li      $v0, 4
    syscall
    move    $a0, $a2
    li      $v0, 1
    syscall
    la      $a0, right_bracket
    li      $v0, 4
    syscall
    j       main

case8.2:                                    # Get[i][j]
    la      $a0, TwoDArrayPtr
    lw      $s1, 0($a0)
    beqz    $s1, nullptr                    # if *ArrayPtr == 0 return error null pointer
    la      $a0, input_row
    jal     integer_input                   # Nhap vao hang
    move    $s0, $a0
```

```
    la      $a0, input_col
    jal     integer_input                   # Nhap vao cot
    move    $a2, $a0                # $a2 = so cot
    move    $a1, $s0                # $a1 = so hang
    move    $a0, $s1                # $a0 = gia tri thanh ghi
    jal     GetArray
    move    $s0, $v0                        # $s0 = gia tri tra ve cua GetArray
    la      $a0, output_val
    li      $v0, 4
    syscall
    move   $a0, $s0
            li  $v0, 1
            syscall
    j       main




#---------------------------------------------------------------
# Ham khoi tao cho viec cap phat dong
# @param khong co
# @detail Danh dau vi tri bat dau cua vung nho co the cap phat duoc
#---------------------------------------------------------------
SysInitMem:
    la      $t9, Sys_TheTopOfFree           # Lay con tro chua dau tien con trong, khoi tao
    la      $t7, Sys_MyFreeSpace            # Lay dia chi dau tien con trong, khoi tao
    sw      $t7, 0($t9)                     # Luu lai
    jr      $ra

#-----------------------------------------------------------------------------
# Ham cap phat bo nho dong cho cac bien con tro
# @param [in/out] $a0 Chua dia chi cua bien con tro can cap phat
# Khi ham ket thuc, dia chi vung nho duoc cap phat se luu tru vao bien con tro
# @param [in] $a1 So phan tu can cap phat
# @param [in] $a2 Kich thuoc 1 phan tu, tinh theo byte
# @return $v0 Dia chi vung nho duoc cap phat
#-----------------------------------------------------------------------------
malloc:
    la      $t9, Sys_TheTopOfFree
    lw      $t8, 0($t9)                     # Lay dia chi dau tien con trong
    li      $t1, 4                          # Do dai 1 word nho
    bne     $a2, $t1, continue              # Neu khong phai cap phat kieu WORD thi OK
    divu    $t8, $t1                # Kiem tra dia chi bat dau cap phat co chia het cho 4 khong
    mfhi    $t2                             # Luu phan du (remainder) vao $t2
    beqz    $t2, continue                   # So du = 0 -> Kich thuoc hop le
    sub     $t3, $t1, $t2                   # So du != 0, can cap phat them (4-sodu) bits
    add     $t8, $t8, $t3                   # tang gia tri thanh ghi len so chia het cho 4
continue:
    sw      $t8, 0($a0)                     # Cat dia chi do vao bien con tro
    addi    $v0, $t8, 0                     # Dong thoi la ket qua tra ve cua ham
    mul     $t7, $a1, $a2                   # Tinh kich thuoc cua mang can cap phat
    add     $t6, $t8, $t7                   # Tinh dia chi dau tien con trong
    sw      $t6, 0($t9)                     # Luu tro lai dia chi dau tien do vao bien
Sys_TheTopOfFree
```

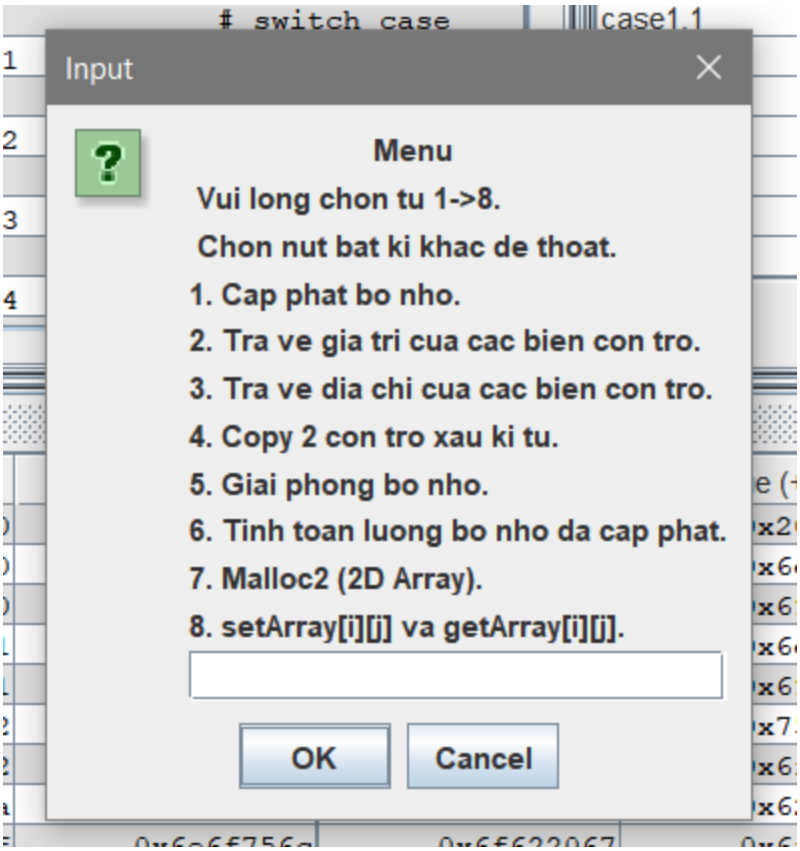
```
print_address:
    la      $a0, malloc_success         # Thong bao cap phat thanh cong
    li      $v0, 4
    syscall
    la      $a0, address
    li      $v0, 4
    syscall
    addi    $a0, $t8, 0                 # Malloc start address
    li      $v0, 34                     # In so integer ra man hinh duoi dang hexa
    syscall
    la      $a0, arrow                  # In ra man hinh " --> "
    li      $v0, 4
    syscall
    addi    $a0, $t6, 0                 # Malloc end address
    li      $v0, 34                     # In so integer ra man hinh duoi dang hexa
    syscall
    jr      $ra
#-----------------------------------------
#  2 ham in ra dia chi va gia tri cua pointer
#  @detail   ptr_val: in gia tri
#            ptr_add: in dia chi
#-----------------------------------------
Ptr_val:
    lw      $a0, 0($a0)                 # Lay gia tri luu trong trong con tro
Ptr_add:
    li      $v0, 34                     # In so integer ra man hinh duoi dang hexa
    syscall
    jr      $ra
#-----------------------------------------
# Tinh tong luong bo nho da cap phat
# @param: none
# @return $v0 chua luong bo nho da cap phat
#-----------------------------------------
MemoryCount:
    la      $t9, Sys_TheTopOfFree
    lw      $t9, 0($t9)                 # $t9 = Gia tri tai dia chi con trong dau tien
    la      $t8, Sys_MyFreeSpace        # Sys_MyFreeSpace luon co dinh la thanh ghi ngay sau
Sys_TheTopOfFree
    sub     $v0, $t9, $t8               # $v0 = luong bo nho da cap phat
    jr      $ra
#-----------------------------------------------------------------------------------
# Ham cap phat bo nho dong cho mang 2 chieu
# Idea: Dua ve cap phat bo nho cho mang 1 chieu co ROW * COL phan tu, su dung lai ham malloc
# @param [in/out] $a0 Chua dia chi cua bien con tro can cap phat
# Khi ham ket thuc, dia chi vung nho duoc cap phat se luu tru vao bien con tro
# @param [in] $a1 so hang
# @param [in] $a2 so cot
# @return $v0 Dia chi vung nho duoc cap phat
#-----------------------------------------------------------------------------------
Malloc2:
    addiu   $sp, $sp, -4                # them 1 phan tu vao stack
    sw      $ra, 4($sp)                 # push $ra
    mul     $a1, $a1, $a2               # tra ve so phan tu cua Array
    li      $a2, 4                      # kich thuoc kieu Word = 4 bytes
```

```
    jal     malloc
    lw      $ra, 4($sp)
    addiu   $sp, $sp, 4                 # pop $ra
    jr      $ra

#-----------------------------------------------------
# gan gia tri cua phan tu trong mang hai chieu
# @param [in] $a0 Chua dia chi bat dau mang
# @param [in] $a1 hang (i)    # @param [in] $a2 cot (j)
# @param [in] $a3 gia tri gan
#-----------------------------------------------------
SetArray:
    bge     $a1, $s4, bound_err         # So hang vuot qua pham vi => error
    bge     $a2, $s5, bound_err         # So cot vuot qua pham vi => error
    bltz    $a1, bound_err              # So hang < 0 => error
    bltz    $a2, bound_err              # So cot < 0 => error
    mul     $s0, $s5, $a1
    addu    $s0, $s0, $a2               # $s0 = i*col +j
    sll     $s0, $s0, 2
    addu    $s0, $s0, $a0               # $s0 = *array + (i*col +j)*4
    sw      $a3, 0($s0)
    jr      $ra

#----------------------------------------
# lay gia tri cua trong mang
# @param [in] $a0 Chua dia chi bat dau mang
# @param [in] $a1 hang (i)
# @param [in] $a2 cot (j)
# @return $v0 gia tri tai hang a1 cot a2 trong mang
#    ----------------------------------------
GetArray:
    bge     $a1, $s4, bound_err         # So hang vuot qua pham vi => error
    bge     $a2, $s5, bound_err         # So cot hang qua pham vi => error
    bltz    $a1, bound_err              # So hang < 0 => error
    bltz    $a2, bound_err              # So cot < 0 => error
    mul     $s0, $s5, $a1
    addu    $s0, $s0, $a2               # $s0= i*col +j
    sll     $s0, $s0, 2
    addu    $s0, $s0, $a0               # $s0 = *array + (i*col +j)*4
    lw      $v0, 0($s0)
    jr      $ra

integer_input:
    li      $v0, 51
    syscall
    beq     $a1, 0, end_input
    beq     $a1, -2, end
    j       integer_input
end_input:
    jr      $ra


check_input:
    bge     $a0, 2000, too_big          # Loi input >     2000
```

```
    beqz    $a0, zero_err           # Loi input =  0
    bltz    $a0, negative_err       # Loi input <  0
    jr      $ra
too_big:
    la      $a0, overflow_error
    j       error
zero_err:
    la      $a0, zero_error
    j       error
negative_err:
    la      $a0, negative_error
    j       error
bound_err:                          # Loi chi so vuot ngoai pham vi
    la      $a0, bound_error
    j       error

nullptr:                            # Loi null
    la      $a0, null_error
    j       error
error:
    li      $v0, 4                  # In ra thong bao loi
    syscall
    j       main

end:
    li      $v0, 10                 # Ket thuc
    syscall
```

4) Hình ảnh kết quả mô phỏng

1. Menu



Cấp phát cho 3 phần từ Char , 6 phần tử Byte và 5 phần tử Word .Kiểu word phải lên 3 byte để lên địa chỉ chia hết cho 4

```
Cap phat bo nho thanh cong.
Address: 0x90000004 --> 0x90000007
Cap phat bo nho thanh cong.
Address: 0x90000007 --> 0x9000000d
Cap phat bo nho thanh cong.
Address: 0x90000010 --> 0x90000024
```

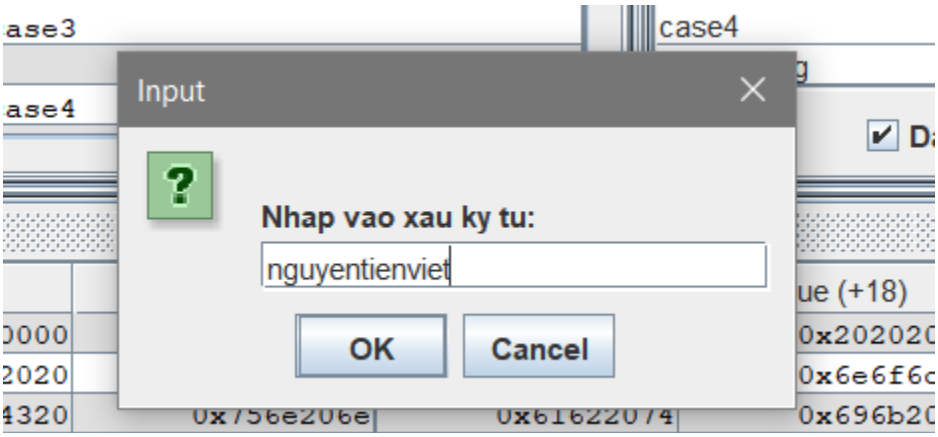
2. Trả về giá trị của các con trỏ trên

```
CharPtr value: 0x90000004
BytePtr value: 0x90000007
WordPtr value: 0x90000010
2DArrayPtr value: 0x00000000
```

3. Trả về địa chỉ của các con trỏ trên

```
CharPtr address: 0x10010000
BytePtr address: 0x10010004
WordPtr address: 0x10010008
2DArrayPtr address: 0x1001000c
```

4. Xâu nhập vào :



Xâu được copy ra :

```
Xau da duoc copy la : nguyentienviet
```

5. Giải phóng bộ nhớ

```
Giai phong bo nho thanh cong.
CharPtr value: 0x00000000
BytePtr value: 0x00000000
WordPtr value: 0x00000000
2DArrayPtr value: 0x00000000
Bo nho da cap phat(Tinh ca byte cap de chuyen dia chi): 0 bytes
```

6. Tính tổng bộ nhớ của câu 1 (tính cả 3 byte để sửa)

3 bytes Char + 6 bytes Bytes + 5*4 bytes Word + 3 bytes sửa = 32 bytes

```
Bo nho da cap phat(Tinh ca byte cap de chuyen dia chi): 32 bytes
```

7. Cấp phát cho mảng kích cỡ 5x5

```
Cap phat bo nho thanh cong.
Address: 0x90000004 --> 0x90000068
```

8. Set giá trị 9 ở vị trí [3][4]

```
Them phan tu vao mang thanh cong.Vi tri : [3][4]
```

Set ở vị trí ngoài phạm vi mảng sẽ báo lỗi

```
Error: Ngoai pham vi cua mang
```

Get giá trị ở vị trí [3][4]

```
Gia tri tra ve: 9
```

Get giá trị ở vị trí [1][1]

```
Gia tri tra ve: 0
```

II. Bài 3 : Kiểm tra tốc độ và độ chính xác khi gõ văn bản

Sinh viên thực hiện : Lê Anh Dũng

1) Đề bài

Thực hiện chương trình đo tốc độ gõ bàn phím và hiển thị kết quả bằng 2 đèn led 7 đoạn.
Nguyên tắc:
- Cho một đoạn văn bản mẫu, cố định sẵn trong mã nguồn. Ví dụ *"bo mon ky thuat may tinh"*
- Sử dụng bộ định thời Timer (trong bộ giả lập Digital Lab Sim) để tạo ra khoảng thời gian để đo. Đây là thời gian giữa 2 lần ngắt, chu kì ngắt.
- Người dùng nhập các kí tự từ bàn phím. Ví dụ nhập "*bo m**O**n ky **5**huat may tinh*". Chương trình cần phải đếm số kí tự đúng (trong ví dụ trên thì người dùng gõ sai chữ **O** và **5**) mà người dùng đã gõ và hiển thị lên các đèn led.
- Chương trình đồng thời cần tính được tốc độ gõ: thời gian hoàn thành và số từ trên một đơn vị thời gian.

Phân tích:

- Input: Cho sẵn một đoạn mã nguồn mẫu.
- Trong khoảng thời gian cố định, người dùng nhập kí tự từ bàn phím.
- Output: Hiện thị ra số lượng kí tự gõ đúng ra 2 đèn led 7 đoạn.

Tính được tốc độ gõ:
+ Thời gian hoàn thành
+ Số từ trên một đơn vị thời gian, số từ gõ được.

2) Phân tích cách làm

- Chương trình chính được chia làm 3 phần:

1.Nhập chuỗi từ MMIO Simulator -- > lưu vào biến buffer
2. Xử lý với số lượng chu kỳ ngắt.
3. Xử lý output tính số lượng ký tự đúng -- > hiện thị kết quả ra 2 đèn led 7 đoạn, thời gian hoàn thành, số từ trên đơn vị thời gian.
4.Tính số lượng từ trong chuỗi buffer, tính thời gian và số từ trên một đơn vị thời gian.

- Cách chạy chương trình:

1. Connect Digital Lab Sim và Keyboard and Display MMIO Simulator với MIPS.
2. Nhấn reset trong Keyboard and Display MMIO Simulator rồi chuyển sang bước 3.
3. Bắt đầu nhập ký tự từ bàn phím và nhấn enter để kết thúc việc nhập.
4. Xem tốc độ nhập ở Run I/O và số ký tự đúng ở 2 đèn led 7 đoạn (Digital Lab Sim)

- Cách tính thời gian:

1. Sử dụng thời gian giữa 2 lần ngắt, chu kì ngắt. Tính chu kì ngắt rồi tinh thời gian
2. Tính WPM số từ gõ trên một phút.

Wpm = 60*số từ/ thời gian gõ

- Cách làm kiểm tra kí tự đúng:

+ Sử dụng một mảng để lưu mã nguồn mẫu(mảng string), một mảng lưu các kí tự gõ vào(mảng buffer)

+ Khi nhập: dùng một biến để đếm số lượng các kí tự đã nhập vào. Nếu gặp phím xóa thì giảm biến này đi. Khi ấn enter thì kết thúc việc gõ.

+ So sánh 2 mảng string và mảng buffer. Dùng một biến để đếm số kí tự đúng.

Kiểm tra ký tự đúng trong chuỗi buffer

Mã C:

```
int count = 0;      // So ky tu dung
for(int i = 0; i < strlen(string); i++){
   if(string[i] == buffer[i])  count++;
}
```

Cách hoạt động: lần lượt so sánh các ký tự trong chuỗi string với chuỗi count nếu giống nhau tăng biến đếm lên 1.

Kiểm tra số từ trong chuỗi buffer

Mã C:

```
char str[20] = "   Hello   world    ";
int word = 1;
if(str[0] == ' ') word = 0;
for (int i = 0; i < strlen(str); i++)
{
   if (str[i] == ' ' && str[i + 1] != ' ')
   {
      word++;
   }
}
if(str[strlen(str)-1] == ' ') word--;
printf("So tu %d ", word);
```

Cách hoạt động:

- Nếu kí tự đầu tiên không là dấu cách thì ban đầu gán word = 1.
- Nếu kí tự đầu tiên là dấu cách thì ban đầu gán word = 0.
- Dùng vòng lặp để đếm nếu kí tự hiện tại là dấu cách và kí tự sau nó không phải là dấu cách thì tăng word lên 1
- Nếu kí tự cuối cùng là dấu cách thì giảm biến đếm đi 1
- Kết quả số từ trong chuỗi là word.

3) Mã nguồn

```
# MMIO Simulator
```

```
.eqv KEY_CODE		0xFFFF0004
.eqv KEY_READY			0xFFFF0000
.eqv DISPLAY_CODE	0xFFFF000C		# ASCII code to show, 1 byte
.eqv DISPLAY_READY	0xFFFF0008		# =1 if the display has already to do
						# Auto clear after sw
# Led 7 doan
.eqv SEVENSEG_RIGHT	0xFFFF0010		# Dia chi cua den led 7 doan phai
.eqv SEVENSEG_LEFT	0xFFFF0011		# Dia chi cua den led 7 doan trai

.eqv LIMIT_TIME			2000000			# Gioi han thoi gian 200000 chu ki ngat

.data
	string: .asciiz "Nhom ky thuat dien tu Nguyen Tien Viet yeu mon Kien truc may tinh"
	buffer: .space 200
	enter: .asciiz "\n\n"
	gach: .asciiz "/"
	ki_tu_dung: .asciiz "\nso ky tu dung la: "
	wpm: .asciiz "\nWPM(So tu tren mot phut): "
	speed: .asciiz "\nTime: "
	giay: .asciiz		" s"
	word: .asciiz "\nTong so tu la: "
	come_back: .asciiz "Ban co muon quay lai chuong trinh? "
	stringIsEmpty: .asciiz "\nERROR: Chuoi dau vao buffer bi rong\n Nhap lai chuoi de test toc do"
.text
	li $t7, 1
begin:

	li $t0, 1
	sb $t0, 0xFFFF0013($zero)	# Set bit tai 0xFFFF0013 khac 0 de kich hoat ngat
						# Sau 30 cau lenh se ngat 1 lan

	li $t0, 0				# Bien dem i = 0
	li $s0, 0				# $s0 Dem so chu ky ngat
	li $t5, 0				# Bien dieu kien dem so chu ky
						# $t5 = 0: Chua dem, $t5 = 1: Bat dau dem chu ky
#Hien thi chuoi string mau
dis_String:
WaitForDis:
	beq $t7,$zero,input_keyboard
	nop

	lw $t2, DISPLAY_READY			# Vong lap cho display san sang
	beq $t2, $zero, WaitForDis		#
	nop
	lb $t1, string($t0)			# Doc string[i]
	beq $t1, '\0', input_keyboard	# if string[i] == NULL then break
```

```
        nop

        sw $t1, DISPLAY_CODE                    # Hien thi string[i]
        add $t0, $t0, 1                 # i++
        j dis_String                            # Lap lai vong lap
#Lay du lieu tu keyboard
input_keyboard:
        li $t7, 0
        li $t0, 0                       # Bien dem i = 0

WaitForKey:
        lw $t4, KEY_READY    # Vong lap cho ban phim san sang
        beqz $t4, WaitForKey #
        nop
        li $t5, 1                  # Dat $t5 = 1, Bat dau dem cac chu ky ngat
        lw $t3, KEY_CODE       # Doc ky tu tu ban phim
        beq $t3, 8, backspace  # Kiem tra nut xoa
        nop
        beq $t3, 10, exit          # Ket thuc go neu nguoi dung nhan enter
        nop
        sb $t3, buffer($t0)        # Luu lai phim vua go

        add $t0, $t0, 1                 # i++
        sb $zero, buffer($t0)
        j WaitForKey
#Gap ky hieu xoa
backspace:
        beqz $t0, WaitForKey # if i == 0 {lap lai vong lap}
        nop
        sb $zero, buffer($t0)   # else{ buffer[i] = NULL;
        add $t0, $t0, -1                #             i--;}
        sb $zero, buffer($t0)   # Ky tu vua xoa cung gan bang null
        j WaitForKey                            # Lap lai vong lap

 #Ket thuc chuong trinh
exit:
        li $t5, 0                  # Khong dem chu ki nua
        move $t6, $t0              # luu so ky tu da go vao t6
        beqz $t6,error
        nop

        li $s4, 0                  # dung: $s4 dem so ky tu go dung
        li $t0, 0                  # i: $t0 bien dem

#Kiem tra ki tu dung
true_character:
```

```
        lb $t1, string($t0)             # $t1 = string[i]
        lb $t2, buffer($t0)             # $t2 = buffer[i]
        beqz $t2, showOutput            # if(buffer[i] == NULL) break;
        nop
        bne $t1, $t2, next_char                 # If (String[i] == buffer[i])
        nop

        add $s4, $s4, 1                 #       dung++;
next_char:
        add $t0, $t0, 1                 # i++;
        j true_character                # Lap lai vong lap
        nop

showOutput:

        li $t0, 0
        sb $t0, 0xFFFF0013($zero)       # Set bit tai 0xFFFF0013 khac 0 de kich hoat ngat
#in xau tu ban phim
        li $v0, 4
        la $a0, buffer
        syscall                         # In xau thu duoc tu ban phim

#in So ky tu dung
        li $v0, 4                               #
        la $a0, ki_tu_dung                      #
        syscall                                 # In message ki_tu_dung

        move $a0, $s4                           #
        li $v0, 1                               # In so ky tu go dung ra man hinh console
        syscall                                 #

        li $v0, 4                               #
        la $a0, gach                            #
        syscall                                 # In message ki_tu_dung

        move $a0, $t6                           #
        li $v0, 1                               # In so ky tu go dung ra man hinh console
        syscall                                 #

#Hien den led
        li $s2, 10
        div $s4, $s2                            # Chia so ki tu go dung cho 10

        mflo $t0                                # Lay so thuong ( Hang chuc)
        jal SET_DATA_FOR_7SEG       #
        move $a1, $a0                           # Dat du lieu hang chuc cho led
```

```
        mfhi $t0                                # Lay so du ( Hang don vi)
        jal SET_DATA_FOR_7SEG       # Dat du lieu hang don vi cho led

        jal SHOW_7SEG_RIGHT         # Hien thi hang don vi led phai
        jal SHOW_7SEG_LEFT          # Hien thi hang chuc led trai
#So go duoc
        li $v0, 4               #
        la $a0, word                    #
        syscall                 # In xau word
        jal CHECK_WORD

        addi $a0, $s6,0         #
        li $v0, 1               # In ra tong so tu go duoc
        syscall                 #

#In ra toc do hoan thanh
        li $v0, 4               #
        la $a0, speed           #In ra xau
        syscall                 #


        div $a0, $s0,54000#
        addi $t6, $a0,0
        li $v0, 1               #  Tong thoi gian go
        syscall                 #

        li $v0, 4               #
        la $a0, giay                    #
        syscall                 # In message


#       addi $a0, $s0,0         #
#       li $v0, 1               #  Tong so ky tu da go
#       syscall                 #

        #jal RESET_BUFFER
#Toc do go wpm
        li $v0, 4               #
        la $a0, wpm                     #
        syscall                 #




        mul $a0, $s6,60         # wpm = so tu * 60/ thoi gian go
```

```
        div $a0, $a0,$t6
        li $v0, 1               # In ra tong so tu go duoc
        syscall                 #




# In dau xuong dong
        li $v0, 4               #
        la $a0, enter                   #
        syscall                 #

        li $s0, 0                       # $s0 Dem so chu ky ngat
        li $t5, 0                       # Bien dieu kien dem so chu ky


        li $v0, 50
        la $a0, come_back
        syscall
        beq $a0,0, begin
        nop


        li $v0, 10              #
        syscall                         # Goi thu tuc ket thuc chuong trinh

#---------------------------------------------------------------------------------------------------
#---------------------------------------------------------------
# Function SHOW_7SEG_RIGHT : turn on/off the 7seg
# @param [in] $a0 value to shown
# remark $t0 changed
#---------------------------------------------------------------
SHOW_7SEG_RIGHT:
        sb $a0, SEVENSEG_RIGHT # assign new value
        jr $ra
#---------------------------------------------------------------
# Function SHOW_7SEG_LEFT : turn on/off the 7seg
# @param [in] $a1 value to shown
# remark $t0 changed
#---------------------------------------------------------------
SHOW_7SEG_LEFT:
        sb $a1, SEVENSEG_LEFT # assign new value
        jr $ra
#---------------------------------------------------------------
# Function SET_DATA_FOR_7SEG : Chuyen du lieu he 10 sang kieu ma hoa LED
```

```
# @param [in] $t0 gia tri he 10
# @return $a0 Ma hoa tung vung hien thi den led
#-------------------------------------------------------------
SET_DATA_FOR_7SEG:
        beq $t0, 0, CASE_0
        beq $t0, 1, CASE_1
        beq $t0, 2, CASE_2
        beq $t0, 3, CASE_3
        beq $t0, 4, CASE_4
        beq $t0, 5, CASE_5
        beq $t0, 6, CASE_6
        beq $t0, 7, CASE_7
        beq $t0, 8, CASE_8
        beq $t0, 9, CASE_9
        nop
CASE_0:         li $a0, 0x3f
        j END_SET_DATA
CASE_1:         li $a0, 0x06
        j END_SET_DATA
CASE_2:         li $a0, 0x5B
        j END_SET_DATA
CASE_3:         li $a0, 0x4f

        j END_SET_DATA
CASE_4:         li $a0, 0x66
        j END_SET_DATA
CASE_5:         li $a0, 0x6D
        j END_SET_DATA
CASE_6:         li $a0, 0x7d
        j END_SET_DATA
CASE_7:         li $a0, 0x07
        j END_SET_DATA
CASE_8:         li $a0, 0x7f
        j END_SET_DATA
CASE_9:         li $a0, 0x6f
        j END_SET_DATA
END_SET_DATA:
        jr $ra
#-----------------------------------------------------------------------------------------------------
# Kiem tra so tu
CHECK_WORD:
        li $t0, -1                      # bien dem i
        li $t6, 0                       # bien dem j
        li $s6, 1                       # So tu trong chuoi
check_first_space:
        lb $t1, buffer($t6)     # $t1 = buffer[0]
```

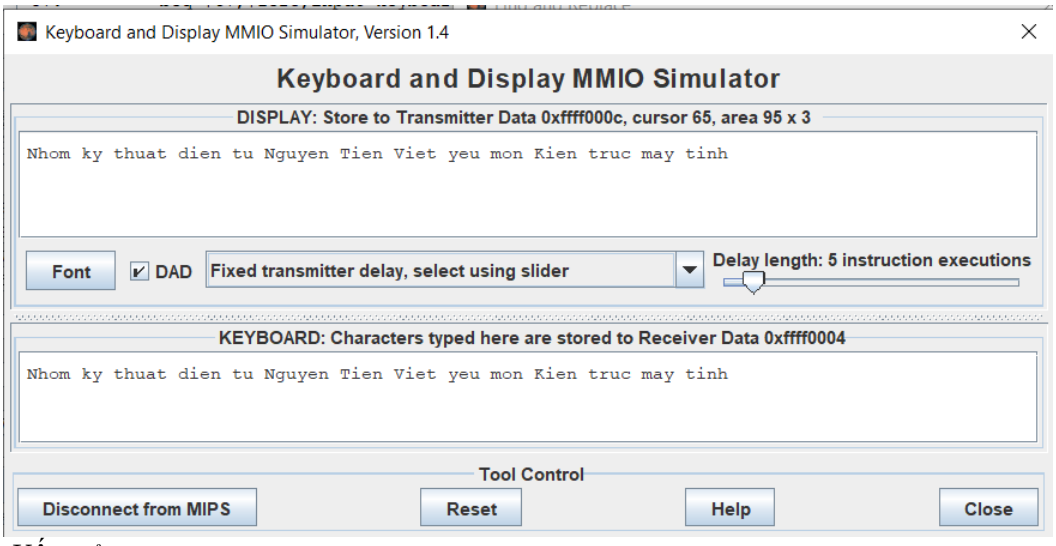
```
        bne $t1, ' ', for                   # if(buffer[i] != ' ' ) chay den vong lap for
        nop
        addi $s6, $s6,-1                    # So tu trong chuoi -1

for:
        addi $t0, $t0,1                     # i ++ ban dau i =0
        addi $t6, $t6,1                     # j ++ ban dau j =1

        lb $t1, buffer($t0)       # $t1 = buffer[i]
        beqz $t1, end_for                   # if(buffer[i] == NULL) break;
        nop
        bne $t1, ' ', for                   # if(buffer[i] != ' ' ) quay tro lai for
        nop

        lb $t2, buffer($t6)       # $t2 = buffer[i+1]
        bne $t2, ' ', count_word            # if( buffer[i+1] != ' ') nhay den count_word
        nop

        j for
        nop
count_word:
        addi $s6, $s6,1                     # Tang so tu len 1

        j for
        nop

end_for:
        addi $t0, $t0,-1                                    # t1 = t1 - 1 vi tri ki tu cuoi cung
        lb $t1, buffer($t0)                         # $t1 = buffer[i]

        bne $t1, ' ', END_CHECK_WORD            # if(buffer[i] != ' ' ) ket thuc chuong trinh con
        nop

        addi $s6, $s6,-1                                    #  if(buffer[i] != ' ' )  so tu -1

END_CHECK_WORD:
        jr $ra

# Thong bao loi
error:
        li          $v0, 4                                                  # "Error"
        la          $a0, stringIsEmpty
        syscall

        li $v0, 4                           #
        la $a0, enter                               #In dau xuong don
```

```
        syscall                         #

        j begin                         # Quay tro lai chuong trinh nhap
        nop

        li $v0, 10                      #
        syscall                         # Goi thu tuc ket thuc chuong trinh

#---------------------------------------------------------------------------------------------
# XU LY NGAT CHUONG TRINH
.ktext 0x80000180
IntSR:
 move $t9, $at                                  # Luu lai gia tri thanh ghi $at
 mfc0 $v0, $13                                  # Kiem tra ma nguyen nhan ngat
 bne $v0, 1024, exit                            # Ma ngat 1024, bo qua ma ngat do Counter cua Digit Lab
Sim
 nop                                            # Ma ngat khac, la Loi => Ket thuc chuong trinh

 add $s0, $s0, $t5                              # Tang bien dem so chu ky ngat, bien dem $s0 chi tang khi
$t5 == 1 (Khi bat dau go)

 sge $v0, $s0, LIMIT_TIME               # Thoat neu dat so chu ky ngat toi da
 bnez $v0, exit
 nop

 move $at, $t9                                  # Khoi phuc thanh ghi $at
return: eret                                            # Quay lai vi tri ngat
```

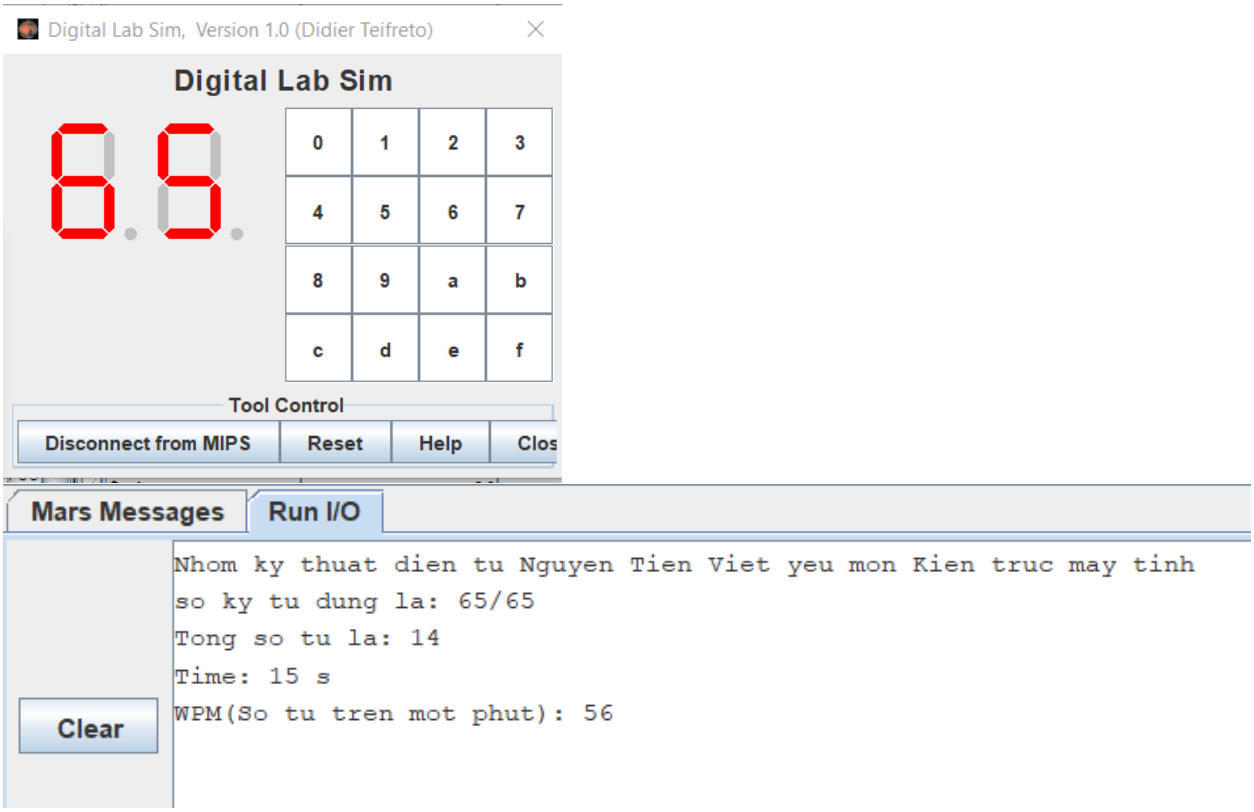
4) Hình ảnh kết quả mô phỏng

1. Gõ kí tự đúng:

-Đầu vào:

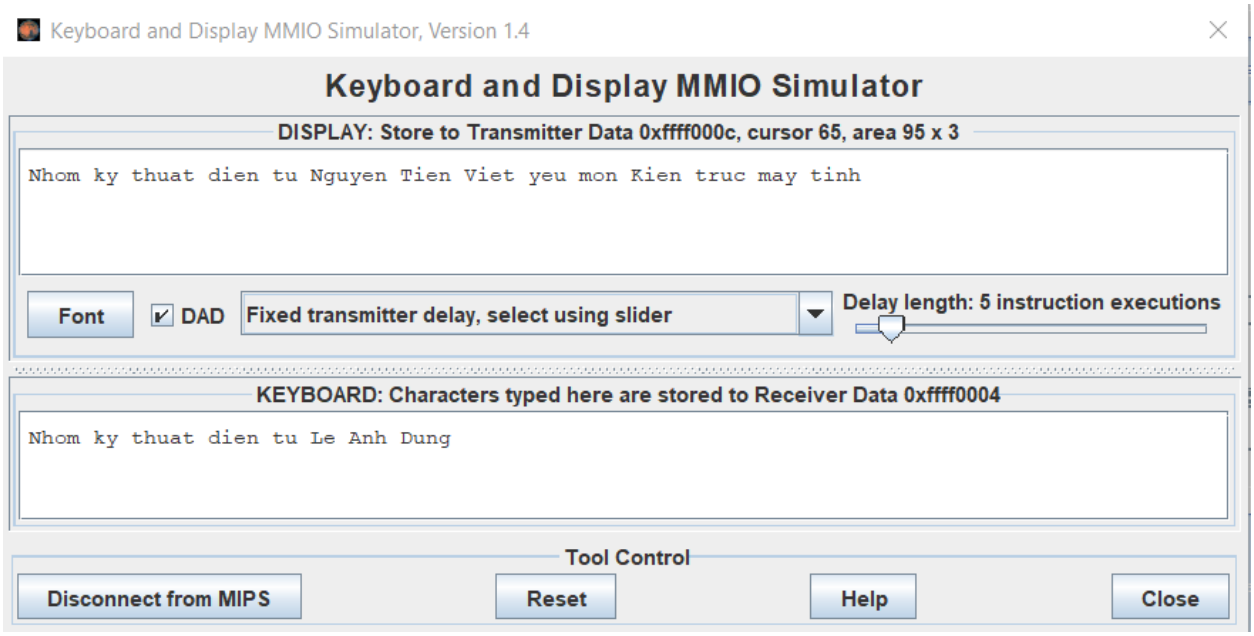


-Kết quả:

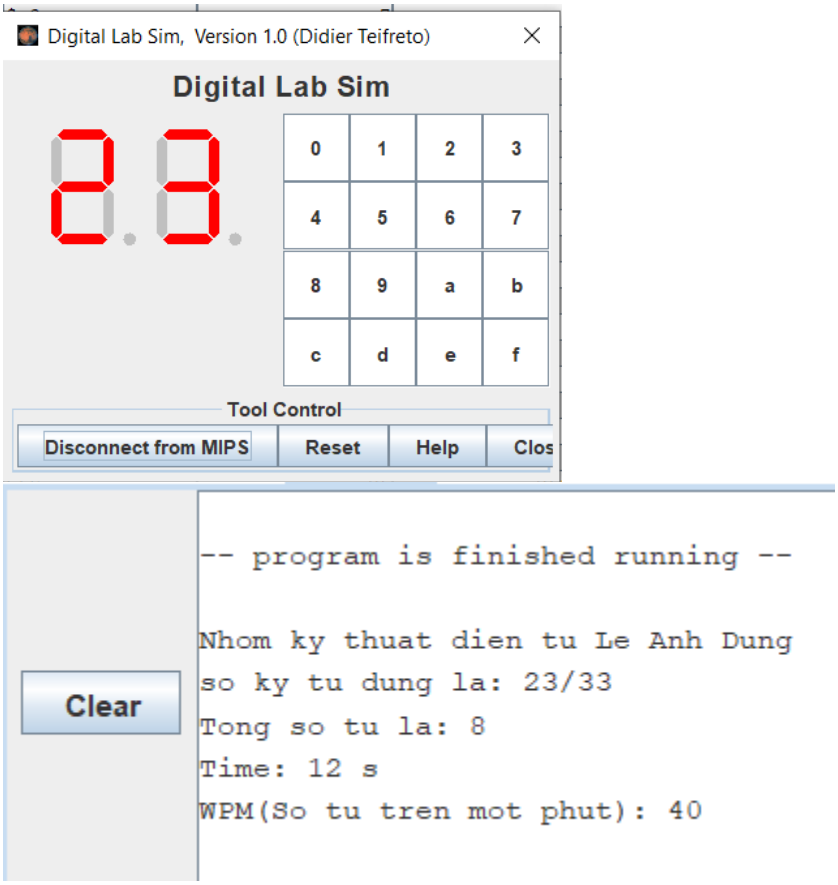


2. Gõ kí tự sai:

-Đầu vào:


Keyboard and Display MMIO Simulator, Version 1.4
Keyboard and Display MMIO Simulator
DISPLAY: Store to Transmitter Data 0xffff000c, cursor 65, area 95 x 3
Nhom ky thuat dien tu Nguyen Tien Viet yeu mon Kien truc may tinh
Font
DAD
Fixed transmitter delay, select using slider
Delay length: 5 instruction executions
KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004
Nhom ky thuat dien tu Le Anh Dung
Tool Control
Disconnect from MIPS
Reset
Help
Close

- Kết quả:


Digital Lab Sim, Version 1.0 (Didier Teifreto)
Digital Lab Sim
0 1 2 3
4 5 6 7
8 9 a b
c d e f
Tool Control
Disconnect from MIPS
Reset
Help
Clos
Clear
-- program is finished running --
Nhom ky thuat dien tu Le Anh Dung
so ky tu dung la: 23/33
Tong so tu la: 8
Time: 12 s
WPM(So tu tren mot phut): 40