TRƯỜNG Đại Học Bách Khoa Hà Nội

--ଓ📖ଓ--

Bài Báo Cáo

Thực hành KTMT

Giáo viên hướng dẫn : Lê Bá Vui

Sinh viên thực hiện :

Nguyễn Thế Thương – 20205031 – Bài 4

Phạm Xuân Duy – 20200114 – Bài 6

Bài 4 .Postscript CNC Marsbot – Nguyễn Thế Thương

A, Mã nguồn :

```
.eqv HEADING 0xffff8010      #Integer : An angle between 0 and 359
.eqv MOVING 0xffff8050       #Boolean : whether or not to move
.eqv LEAVETRACK 0xffff8020   #Boolean ( 0 or not 0 )
.eqv WHEREX 0xffff8030       #current x-location of Marsbot
.eqv WHEREY 0xffff8040       #current y-location of Marsbot

.eqv IN_ADDRESS_HEXA_KEYBOARD 0xFFFF0012
.eqv OUT_ADDRESS_HEXA_KEYBOARD 0xFFFF0014



.data
#script- DCE , number 0
script1:     .word  135,3000,0, 180, 6000, 1, 60, 3000, 1, 0, 3000, 1, 300, 3000, 1, 90, 6000, 0, 270, 2600, 1, 180, 6000, 1, 90, 2600, 1, 90, 3500, 0, 270, 2600, 1, 0, 3000, 1,90, 3000, 1, 270, 3000, 0, 0, 3000, 1, 90, 3000, 1
script1_len :  .word 17
#postscript-0531 => numpad 4
script2:     .word          135, 2000, 0, 180, 4000, 1, 90, 2000, 1, 0, 4000, 1, 270, 2000, 1, 90, 3000, 0, 180, 2000, 1, 90, 2000, 1, 180, 2000, 1, 270, 2000, 1, 0, 4000, 0,90, 2000, 1,90, 1000, 0, 90, 2000, 1, 180, 2000, 1, 270, 2000, 1, 90, 2000, 0, 180, 2000, 1, 270, 2000, 1, 90, 4000 , 0, 0, 4000, 1
script2_len:  .word 21
# postscript-2020 => numpad 8
script3: .word  135, 2000, 0, 90, 2000, 1, 180, 2000, 1, 270, 2000, 1, 180, 2000, 1, 90, 2000, 1, 90, 1000, 0, 90, 2000, 1, 0, 4000, 1, 270, 2000, 1, 180, 4000, 1, 90, 5000, 0, 270, 2000, 1, 0, 2000, 1, 90, 2000, 1, 0, 2000, 1, 270, 2000, 1, 90, 3000, 0,90, 2000, 1, 180, 4000, 1, 270, 2000, 1, 0 , 4000, 1
script3_len: .word 22


.text
```

```
main:
    li      $t1, IN_ADDRESS_HEXA_KEYBOARD
    li      $t2, OUT_ADDRESS_HEXA_KEYBOARD


POLLING:
    li      $t3, 0x1       # check row1
    sb      $t3, 0($t1)     # Must reasign axpected row
    lb      $a0, 0($t2)     # read scan code of key bytton
    bne     $a0, 0x11, NUMBER_4 # $a0 != 0x11 => NUMBER_4
    la      $s0, script1    # gan $s0 = value script1
    la      $s1, script1_len
    lw      $s1, 0($s1)
    j       START          # thuc thi ham START

NUMBER_4:
    li      $t3, 0x2    # check row2
    sb      $t3, 0($t1)     # Must reasign axpected row
    lb      $a0, 0($t2)    # read scan code of key bytton
    bne     $a0, 0x12, NUMBER_8 # $a0 != 0x11 => NUMBER 8
    la      $s0, script2    # gan $s0 = value script2
    la      $s1, script2_len
    lw      $s1, 0($s1)
    j       START # thuc thi ham START
    nop
NUMBER_8:
    li      $t3, 0x4    # check row3
    sb      $t3, 0($t1)    # Must reasign axpected row
    lb      $a0, 0($t2)   # read scan code of key bytton
```

```
    bne     $a0, 0x14, RETURN   # $a0 != 0x11 => return
    la      $s0, script3     # gan $s0 = value script3
    la      $s1, script3_len
    lw      $s1, 0($s1)
    j       START        # thuc thi ham START
    nop
RETURN: j       POLLING      # $a0 != 0,4,8 => polling
    nop

START:
    addi    $t9, $zero, 0 # j =0 dem so luong duong cat
    addi    $t0, $zero, 0 # i =0 vi tri

FOR:
    add         $t1, $t0, $s0  #$t1 gia tri phan tu
        lw      $t2, 0($t1)     #rotate
        lw      $t3, 4($t1)    #time
        lw      $t4, 8($t1)    #track

        add     $a0, $zero, $t4  #$a= $zero + $s4
        jal     TRACK
        nop

        add     $a0, $zero, $t2
        jal     ROTATE
        nop

        jal     GO
        nop
```

```
SLEEP:  addi    $v0, $zero, 32
        add     $a0, $zero, $t3
        syscall


IF_TRACK:
        beq     $t4, $zero, END_IF_TRACK
        jal     UNTRACK
        nop

END_IF_TRACK:
        addi    $t0, $t0, 12
        addi    $t9, $t9, 1
        slt     $t5, $t9, $s1
        beq     $t5, $zero, END_FOR
        j       FOR
        nop
END_FOR:
        j       END
        nop


GO:
        li    $at, MOVING  # change MOVING port
        addi   $k0, $zero, 1  # to logic 1
        sb     $k0, 0($at)  # to start running
        nop
        jr    $ra
```

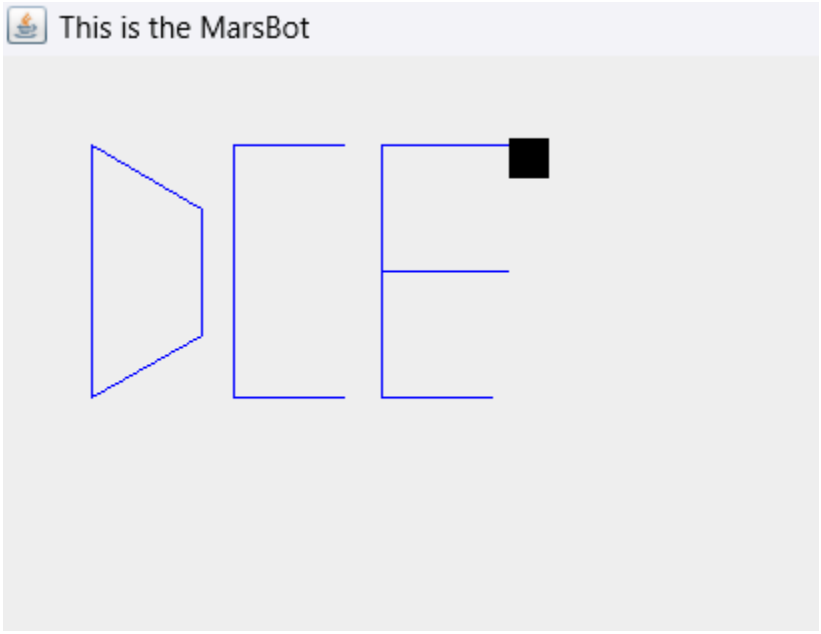
```
        nop

STOP:
        li      $at, MOVING # change MOVING port to 0
        sb      $zero, 0($at) # to stop
        nop
        jr      $ra
        nop

TRACK:
        li      $at, LEAVETRACK # change LEAVETRACK port
        sb      $a0, 0($at)     # to start tracking
        nop
        jr      $ra
        nop

UNTRACK:
        li      $at, LEAVETRACK # change LEAVETRACK port to 0
        sb      $zero, 0($at) # to stop drawing tail
        nop
        jr      $ra
        nop

ROTATE:
        li      $at, HEADING # change HEADING port
        sw      $a0, 0($at) # to rotate robot
        nop
        jr $ra
        nop
```

END:

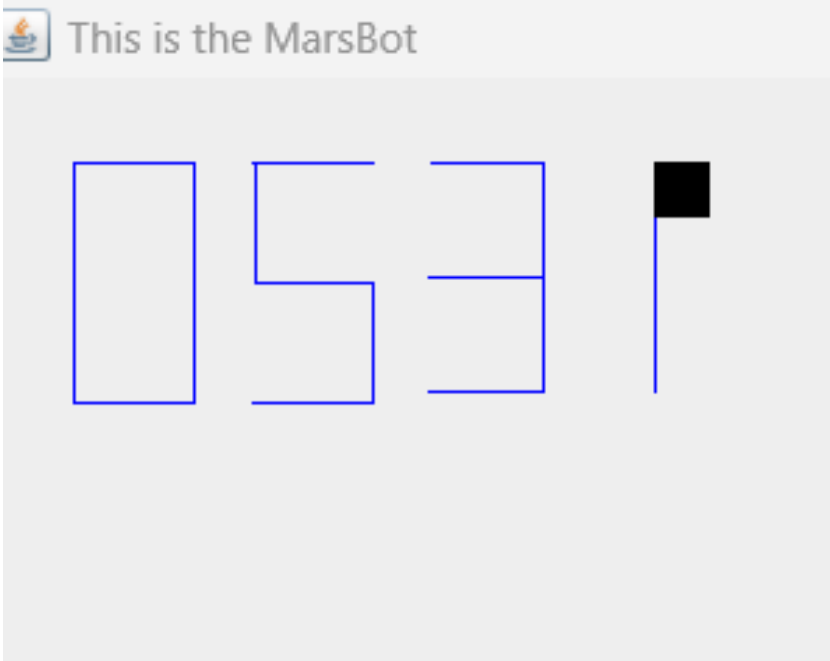
```
jal    STOP
li     $v0, 10
syscall
nop
```

B,Kết quả

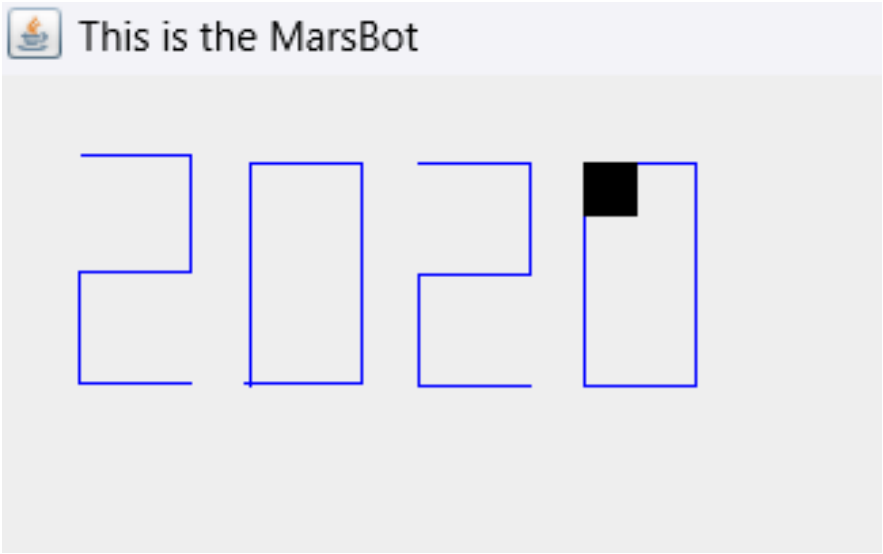
- Script 1: chữ DCE



-Script 2: số 5031



-Script 3: số 2020



C, Phân tích cách làm :

-Phân tích cách thực hiện

+Thanh ghi $s0 : lưu các giá trị phần tử của mảng

+Thanh ghi $s1 : lưu số đường cắt và không cắt

+Thanh ghi $t1 : lưu địa chỉ 0xFFFF0012

+Thanh ghi $t2 : lưu địa chỉ 0xFFFF0014

+Chọn Script từ matrix

Thanh ghi $t3: lưu giá trị để lưu byte vào 0xFFFF0012, chọn hàng để nhận diện trong 0xFFFF0014. Nếu đúng hàng, byte tại địa chỉ 0xFFFF0014 sẽ trả về phím đang bấm ở Digital LabSim.

Thực hiện bước gán $t3 bằng giá trị hàng 1, nếu nút đang nhấn ở hàng 1, tiếp tục kiểm tra giá trị có khớp với 0x11(Nút 0) hay không.

Nếu đúng, lựa chọn Script 1 bằng cách: lấy địa chỉ script_1 và script_1_len đưa vào $s0 và $s1. Nếu khác 0x11(Nút 0) thì sang nút 4 và tương tụ nút 8

Nếu không phải nút 0, 4, 8 thì quay lại bước ban đầu cho đến khi nhập đúng 1 trong 3 nút yêu cầu.

+*Đọc dữ liệu từ postscript và chạy Marsbot*

Các thủ tục điều khiển Marsbot:

GO: không có tham số đầu vào, cho Marsbot thực hiện di chuyển

STOP: không có tham số đầu vào, dừng Marsbot

TRACK: tham số đầu vào **$a0,** = 1 thì thực hiện in và không thì ngược lại

UNTRACK: không có tham số đầu vào, Marsbot không để lại dấu trên đường đi

ROTATE: Tham số đầu vào **$a0**, góc quay Marsbot sẽ đi chuyển

D,Thuật toán

-Sử dụng vòng lặp lấy giá trị lần lượt là I,i+1,i+2 gán vào ba thanh ghi $t2,$t3,$t4

-Truyền giá trị $t4 vài $a0 , gọi tới track

- Tương tự $t2 = $a0 , gọi tới rotate

-Thực hiện Go để di chuyển và SLEEP để duy trì di chuyển

-Kiểm tra $t4 = 1 gọi tới UNTRACK để lưu vết cũ

-Nếu $t9 = $s1 gọi END

-Nếu $t9< $s1 quay lại

Bài 6 –Phạm Xuân Duy :

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.

A. Mã nguồn:

```
.data
        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
        row:            .word 1
        col:            .word 1
        menu:           .asciiz "               Menu\n Vui long chon tu 1->11.\n Chon nut bat ki khac de thoat.\n1. Malloc Char Pointer.\n2. Malloc Byte Pointer.\n3. Malloc Word Pointer.\n4. Tra ve gia tri cua cac bien con tro.\n5. Tra ve dia chi cua cac bien con tro.\n6. Copy 2 con tro xau
```

```
ki tu.\n7. Giai phong bo nho.\n8. Tinh toan luong bo nho da cap phat.\n9. Malloc2 (2D Array).\n10. setArray[i][j].\n11. getArray[i][j]."
        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: "
        address_str:    .asciiz "\nDia chi cua bien con tro CharPtr | BytePtr | WordPtr | TwoDArrayPtr la: "
        value_str:      .asciiz "\nGia tri cua bien con tro CharPtr | BytePtr | WordPtr | TwoDArrayPtr la: "
        malloc_str:     .asciiz "\nBo nho da cap phat: "
        bytes_str:      .asciiz " bytes"
        input_str:      .asciiz "\nNhap vao xau ky tu: "
        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 : "
        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"
        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
        beq     $s0, 9, case9
        beq     $s0, 10, case10
        beq     $s0, 11, case11
        j       end                     # Neu khac 1->11 => end

case1:                                  # 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 < 1000)
        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
        move    $s0, $v0                # $s0 = gia tri tra ve ham malloc
        la      $a0, malloc_success     # Thong bao cap phat thanh cong
        li      $v0, 4
        syscall
        j       main

case2:                                  # 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 < 1000)
        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
        move    $s0, $v0                # $s0 = gia tri tra ve ham malloc
        la      $a0, malloc_success     # Thong bao cap phat thanh cong
        li      $v0, 4
        syscall
        j       main
```

```
case3:                                  # 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 < 1000)
        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
        move    $s0, $v0                # $s0 = gia tri tra ve ham malloc
        la      $a0, malloc_success     # Thong bao cap phat thanh cong
        li      $v0, 4
        syscall
        j       main

case4:
        la      $a0, value_str
        li      $v0, 4
        syscall
        li      $a0, 0
        jal     Ptr_val                 # Lay gia tri cua CharPtr
        jal     print_value

        li      $a0, 1
        jal     Ptr_val         # Lay gia tri cua BytePtr
        jal     print_value

        li      $a0, 2
        jal     Ptr_val         # Lay gia tri cua WordPtr
        jal     print_value

        li      $a0, 3
        jal     Ptr_val         # Lay gia tri cua TwoDArrayPtr
        jal     print_value

        j       main

case5:
        la      $a0, address_str
        li      $v0, 4                  # print string service
        syscall

        li      $a0, 0                  # Lay dia chi cua CharPtr
        jal     Ptr_addr
```

```
        jal     print_value

        li      $a0, 1              # Lay dia chi cua BytePtr
        jal     Ptr_addr

        jal     print_value

        li      $a0, 2              # Lay dia chi cua WordPtr
        jal     Ptr_addr

        jal     print_value

        li      $a0, 3              # Lay dia chi cua TwoDArrayPtr
        jal     Ptr_addr

        jal     print_value

        j       main

case6:
input_string:
        li      $v0, 54                 # InputDialogString
        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
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 CharPtr2
        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      $t3, ($t1)              # Load ki tu tren cung tai $t1 vao $t3
        sb      $t3, ($t2)              # Luu 1 ki tu cua $t3 vao o nho tai dia chi $t2
        addi    $t4, $t4, 1             # $t4 : dem so luong ki tu string
        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', exit_copy        # Check null => end string
        j       copy_loop
exit_copy:
        la      $a0, copy_str
        li      $v0, 4
        syscall
        sw      $t4,($a0)                   # Luu so byte(s) dung de luu string
        la      $a2, CharPtr2               # Load dia chi CharPtr2 vao $a2
        lw      $a0, ($a2)                  # Luu xau da copy tu $a0 vao CharPtr2
        li      $v0, 4                      # In ra gia tri CharPtr2
        syscall
        j       main

case7:                                      # 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

case8:                                      # Giai phong bo nho
        la      $t0, CharPtr

        jal     SysInitMem                  # Khoi tao lai vung cap phat dong
        la      $a0, free_success
        li      $v0, 4
        syscall
        j       case8                       # jump toi case8 de check xem bo nho da duoc
giai phong

case9:                                      # Cap phat bo nho cho mang 2 chieu Malloc2
        la      $a0, nb_row
        jal     integer_input               # Nhap vao so hang
        jal     check_input
        move    $s0, $a0
        la      $a0, nb_col
```

```
        jal     integer_input           # Nhap vao so cot
        jal     check_input
        move    $a1, $s0                # $a1 = so hang
        move    $a2, $a0                # $a2 = so cot
        la      $a0, TwoDArrayPtr
        jal     Malloc2                 # Cap phat bo nho cho mang 2 chieu
        move    $s0, $v0                # $s0 = gia tri tra ve cua malloc2
        la      $a0, malloc_success
        li      $v0, 4
        syscall
        j       main

case10:                                 # Set[i][j]
        la      $a0, TwoDArrayPtr
        lw      $s7, 0($a0)
        beqz    $s7, nullptr            # if *ArrayPtr==0 => null error
        la      $a0, input_row
        jal     integer_input           # Nhap vao hang
        move    $s0, $a0                # $s0 = so thu tu hang
        la      $a0, input_col
        jal     integer_input           # Nhap vao cot
        move    $a2, $a0                # $a2 = so thu tu cot
        la      $a0, input_val
        jal     integer_input           # Nhap gia tri can set
        move    $a3, $a0                # $a3 = gia tri can set
        move    $a1, $s0                # $a1 = so thu tu hang
        move    $a0, $s7
        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

case11:                             # 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                    # $0 = so hang
        la      $a0, input_col
        jal     integer_input               # get col
        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
        bne     $a2, 4, continue        # Neu khong phai kieu Word thi continue
        addi    $t8, $t8, 3
        andi    $t8, $t8, 0xfffffffc    # 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
        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
        la      $s0, row
        sw      $a1, 0($s0)             # luu so hang vao row
        sw      $a2, 4($s0)             # luu so cot vao col
        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
```

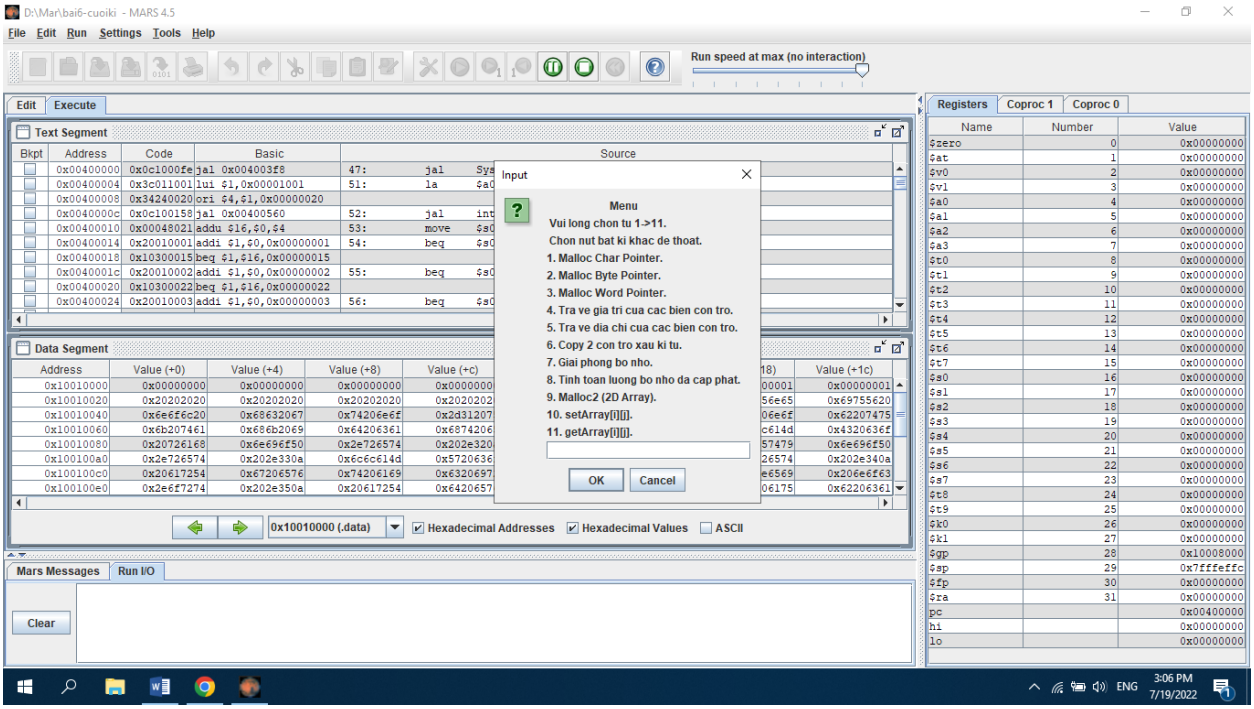
```
#-----------------------------------------------------
# gan gia tri cua phan tu trong mang hai chieu
# @param [in] $a0 Chua dia chi bat dau mang
# @param [in] $a1 hang (i)        # @param [in] $a2 cot (j)
# @param [in] $a3 gia tri gan
#-----------------------------------------------------
SetArray:
        la      $s0, row                        # $s0 = dia chi so hang
        lw      $s1, 0($s0)                     # $s1 so hang
        lw      $s2, 4($s0)                     # $s2 so cot
        bge     $a1, $s1, bound_err             # So hang vuot qua pham vi => error
        bge     $a2, $s2, 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, $s2, $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:
        la      $s0, row                        # $s0 = dia chi so hang
        lw      $s1, 0($s0)                     # $s1 so hang
        lw      $s2, 4($s0)                     # $s2 so cot
        bge     $a1, $s1, bound_err             # So hang vuot qua pham vi => error
        bge     $a2, $s2, 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, $s2, $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

#-----------------------------------------------------
```

```
# Ham lay gia tri cua cac bien con tro
# @param [in] $a0 {0: char ; 1: byte ; 2: word ; 3: 2Darray}
# @return $v0 gia tri bien con tro
#-------------------------------------------------------
Ptr_val:
        la      $t0, CharPtr            # Luu dia chi bien con tro CharPtr vao $t0
        sll     $t1, $a0, 2             # CharPtr, BytePtr, WordPtr nam lien tiep nhau
        add     $t0, $t0, $t1           # $t0 luu dia chi cua
CharPtr/BytePtr/WordPtr/TwoDArrayPtr
        lw      $v0, 0($t0)             # lay gia tri luu tai bien con tro va luu vao $v0 (gia tri tra
ve)
        jr      $ra

#-------------------------------------------------------
# Ham lay dia chi cua cac bien con tro
# @param [in] $a0 {0: char ; 1: byte ; 2: word ; 3: 2Darray}
# @return $v0 dia chi bien con tro
#-------------------------------------------------------
Ptr_addr:
        la      $t0, CharPtr            # Luu dia chi bien con tro CharPtr vao $t0
        sll     $t1, $a0, 2             # CharPtr, BytePtr, WordPtr nam lien tiep nhau
        add     $v0, $t0, $t1           # $v0 luu dia chi cua
CharPtr/BytePtr/WordPtr/TwoDArrayPtr
        jr      $ra

print_value:                            # in ra gia tri $v0
        move    $a0, $v0
        li      $v0, 34                 # print integer in hexadecimal service
        syscall
        li      $a0, ';'
        li      $v0, 11                 # print character service
        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
```

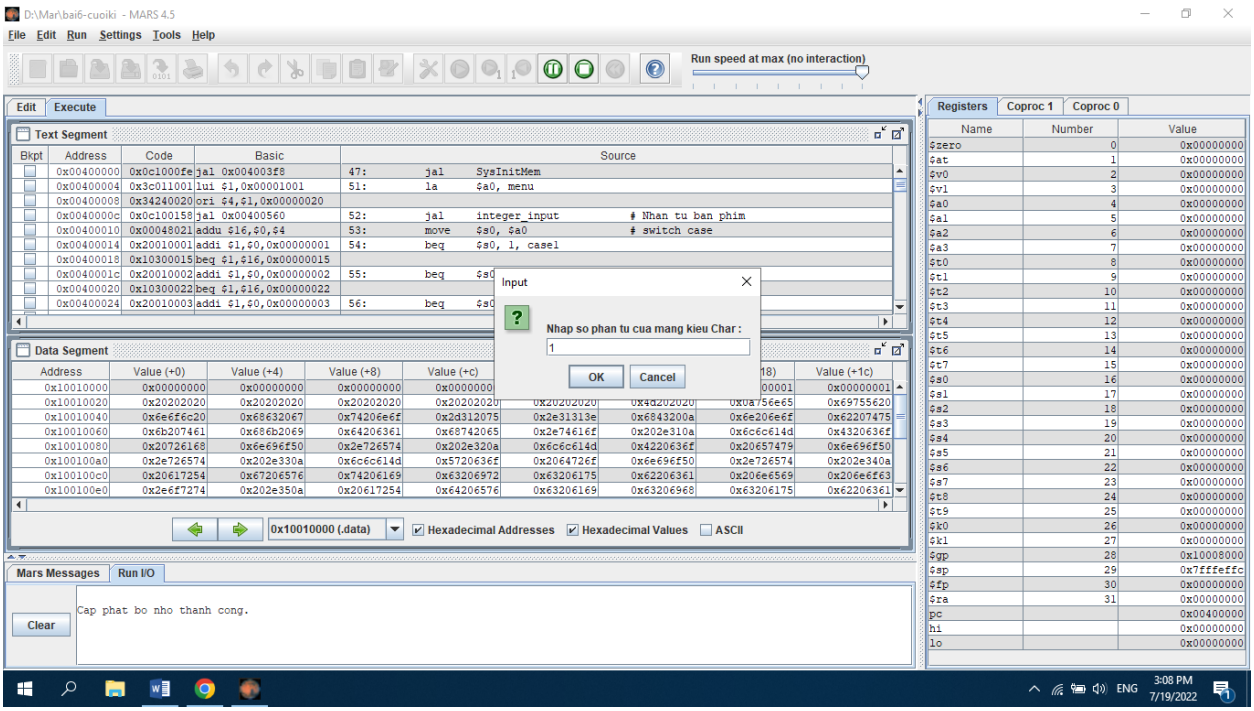
```
        sub     $v0, $t9, $t8                   # $v0 = luong bo nho da cap phat
        jr      $ra



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, 1000, too_big      # Loi input >     1000
        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

B. Kết quả:

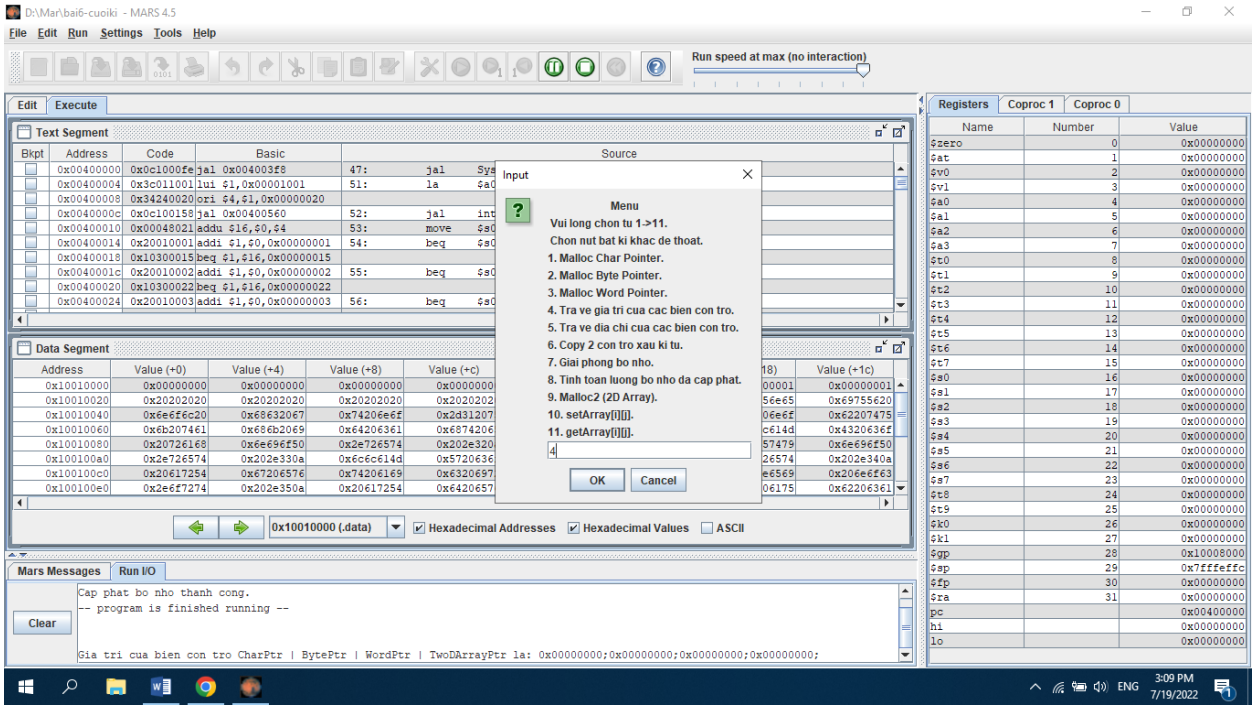


- Với các case 1,2,3 có tác dụng cung cấp bộ nhớ động cho kiểu char, byte, word

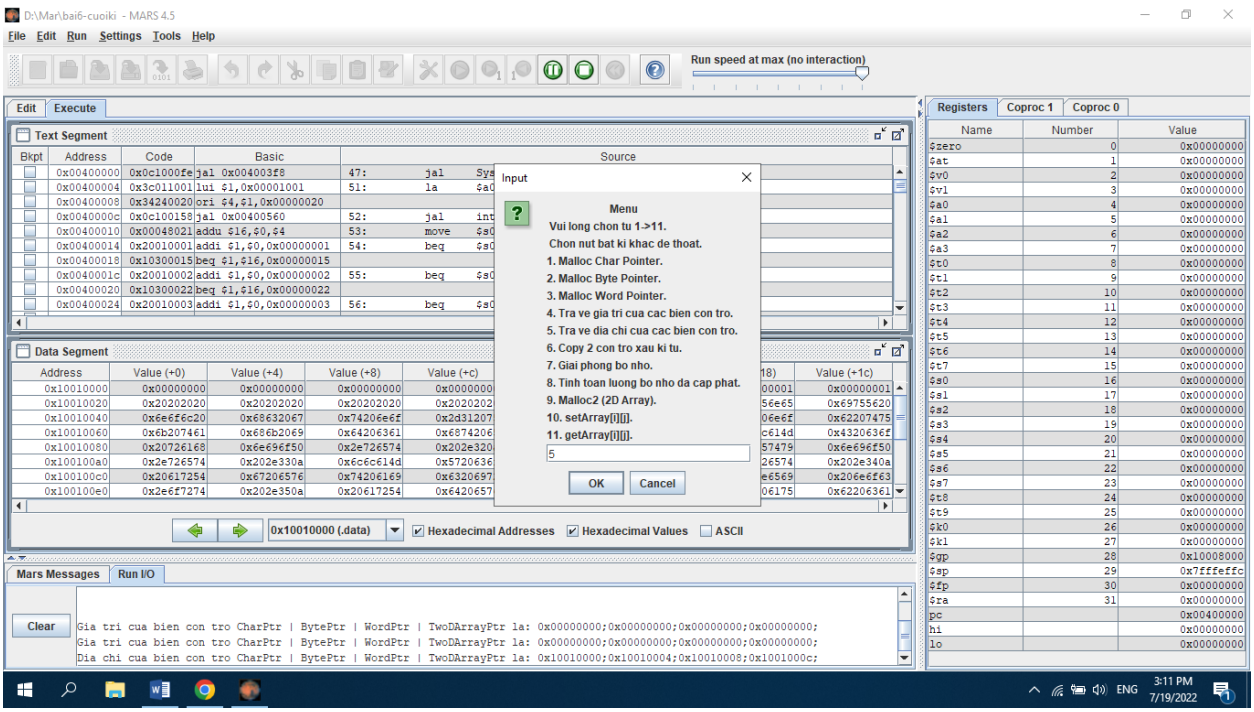


Kết quả: Cấp phát bộ nhớ thành công

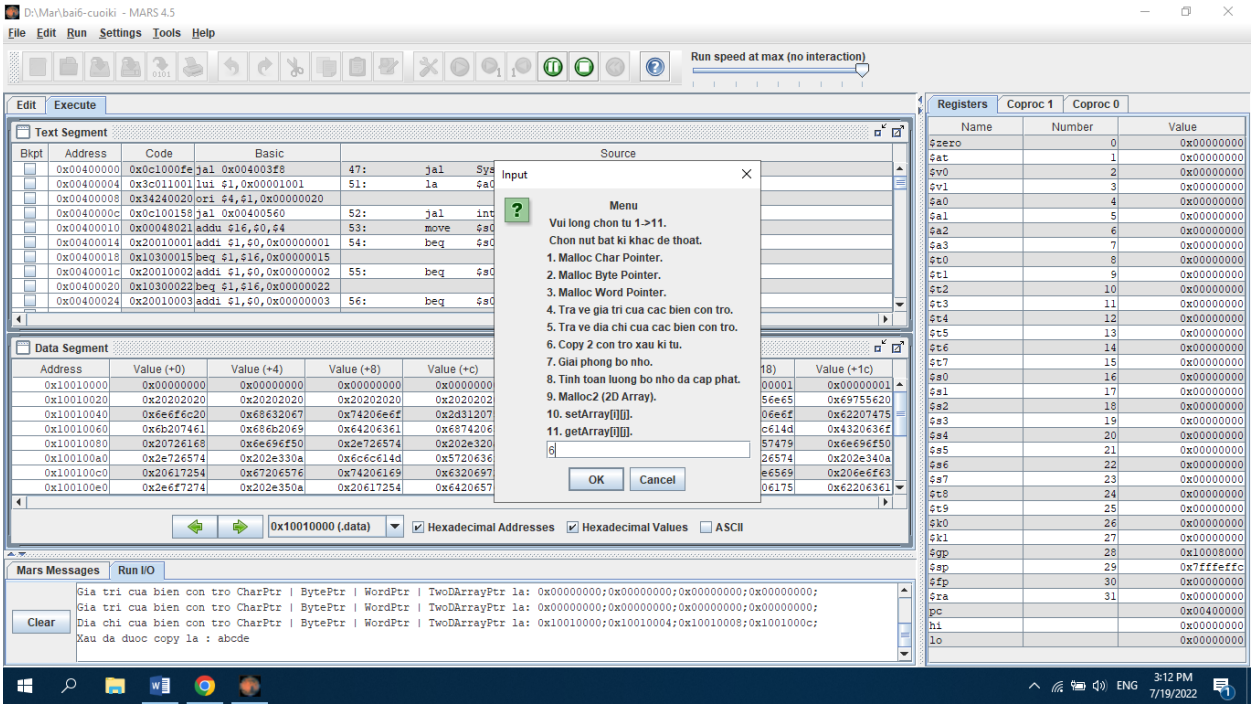
- Với case 4: hiển thị giá trị của biến con trỏ



- Với case 5: Hiển thị địa chỉ của biến con trỏ



- Với case 6 : dù để copy 1 xâu kí tự

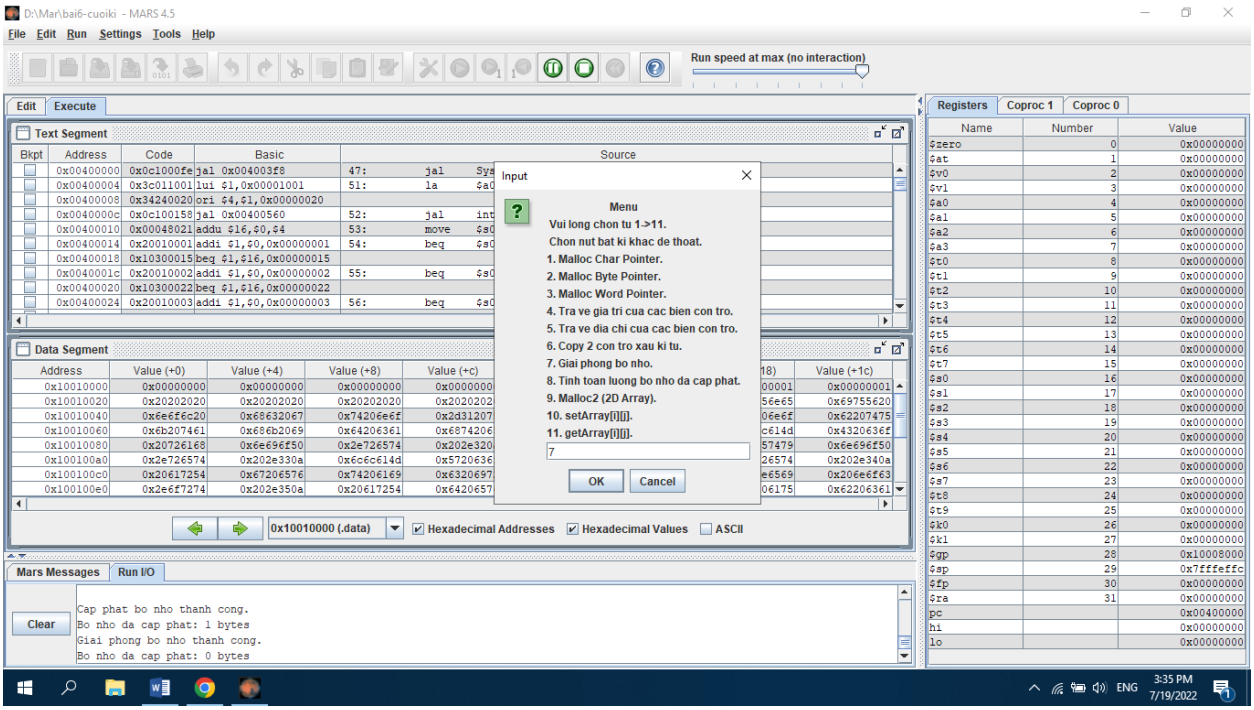


Khi ấn sẽ hiện ra một cửa sổ để điền 1 xâu cần copy (ở đây là abcde) xâu kết quả là “abcde”

-Với case 7: hiển thị bộ nhớ đã cấp phát cho từng kiểu dữ liệu:

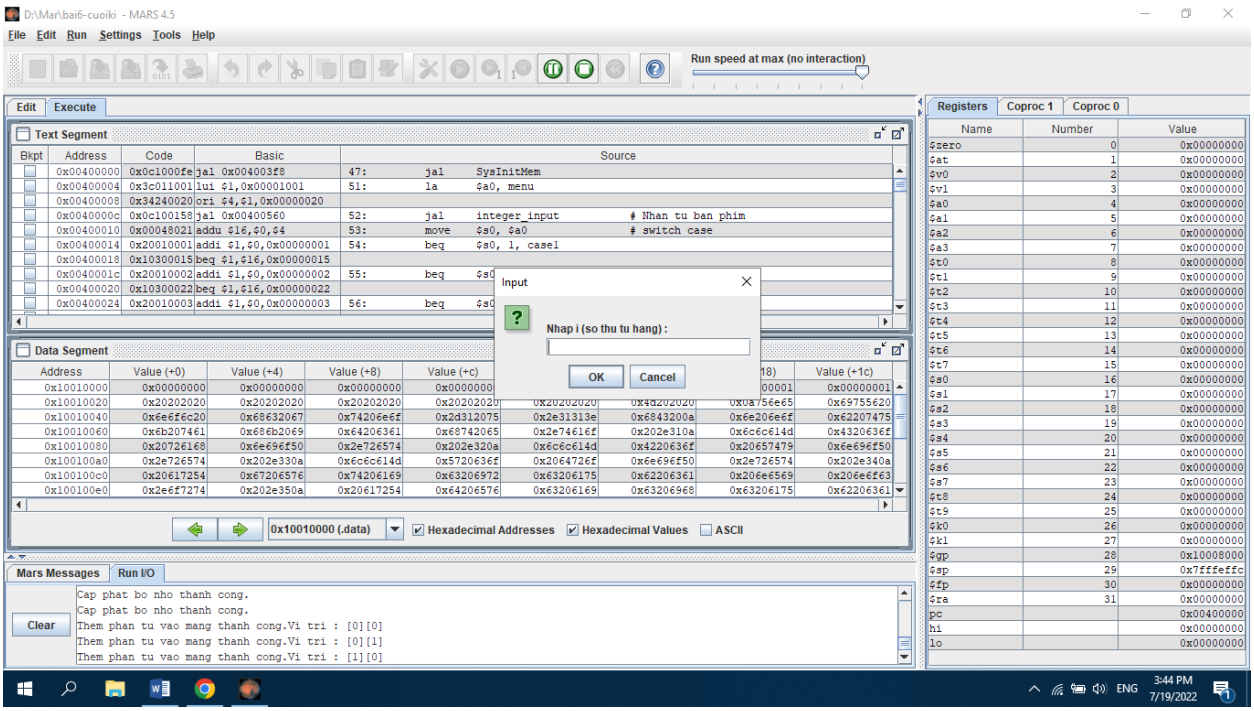
Với kiểu word: 4*n với n là số phần tử con trỏ
Với kiểu char: n với n là số phần tử con trỏ
Với kiểu byte: 2*n với n là số phần tử con trỏ

- Với case 8: giải phóng bộ nhớ



- Với case 9: Tạo con trỏ 2 chiều
  Kết quả hiện ra yêu cầu nhập số hàng và cột để cung cấp bộ nhớ

- Với case 10: nhập giá trị vào mảng 2 chiều vừa được cung cấp bộ nhớ ở case 9
  Kết quả yêu cầu nhập vào vị trí cần điền (bao gồm số hàng số cột bắt đầu từ hàng 0, cột 0) sau đó nhập giá trị của vị trí đó để tiếp tục cho case 11



- Với case 11: đưa ra giá trị của vị trí đã chọn
  Yêu cầu nhập địa chỉ của ô cần tìm (số hàng số cột)
  Kết quả: đưa ra giá trị của vị trí cần tìm (nếu không tồn tại địa chỉ đó hay giá trị sẽ hiển thị ra lỗi)