

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



Bài báo cáo project cuối kì

GVHD: Lê Bá Vui

Nhóm: 10

Thành Viên: Phạm Huy Hoàng – 20194574

Lê Tuấn Hưng – 20194584

Mã Lớp: 130939

MỤC LỤC

LỜI MỞ ĐẦU3

I. Bài 6:4

- 1. Đề bài**4
- 2. Phân tích đề bài**4
- 3. Phân tích cách làm**5
- 4. Mã nguồn**5
- 5. Hình ảnh kết quả mô phỏng**15
 - o Menu lựa chọn:15
 - o Cấp phát bộ nhớ cho biến con trỏ CharPtr15
 - o Cấp phát bộ nhớ cho biến con trỏ BytePtr16
 - o Cấp phát bộ nhớ cho biến con trỏ WordPtr17
 - o Lấy giá trị của các biến con trỏ18
 - o Lấy địa chỉ của các biến con trỏ19
 - o Copy xâu19
 - o Tính toán lượng bộ nhớ đã cấp phát20
 - o Cấp phát bộ nhớ cho mảng 2 chiều21
 - o Thiết lập giá trị các phần tử cho mảng 2 chiều22
 - o Lấy giá trị của các phần tử mảng 2 chiều23

II. Bài 4:Lỗi! Thẻ đánh dấu không được xác định.

- 1. Đề bài**Lỗi! Thẻ đánh dấu không được xác định.
- 2. Phân tích đề bài**Lỗi! Thẻ đánh dấu không được xác định.
- 3. Phân tích cách thực hiện**Lỗi! Thẻ đánh dấu không được xác định.
- 4. Mã nguồn**Lỗi! Thẻ đánh dấu không được xác định.
- 5. Hình ảnh kết quả mô phỏng**Lỗi! Thẻ đánh dấu không được xác định.
 - o Marsbot ở chế độ chờ:15
 - o Marsbot thực hiện DCE khi nhập 0:15
 - o Marsbot thực hiện DAT khi nhập 4:15
 - o Marsbot thực hiện DUY khi nhập 8:15

LỜI MỞ ĐẦU

Nhóm gồm 2 thành viên:

- Phạm Huy Hoàng : Thực hiện bài 6
- Lê Tuấn Hưng: Thực hiện bài 4

Bản báo cáo khái quát quá trình thực hiện 2 bài tập lớn là bài 6 và bài 4 gồm các nội dung chính:

- ❖ Đề bài.
- ❖ Phân tích đề bài.
- ❖ Cách làm.
- ❖ Mã nguồn.
- ❖ Hình ảnh kết quả.

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 chân thành cảm ơn.

BÁO CÁO PROJECT MÔN THỰC HÀNH KIẾN TRÚC MÁY TÍNH

I. Bài 6:

Hàm cấp phát bộ nhớ Malloc()

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 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 Word /Byte của biến con trỏ (tương tự như *CharPtr, *BytePtr, *WordPtr).
3. Viết hàm lấy địa chỉ biến con trỏ (tương tự như &CharPtr, &BytePtr, *WordPtr).
4. Viết hàm thực hiện copy 2 con trỏ chuỗi ký tự (Xem ví dụ về CharPtr).
5. Viết hàm tính toàn bộ lượng bộ nhớ đã cấp phát cho các biến động.
6. Hãy viết hàm Malloc2 để cấp phát cho mảng 2 chiều kiểu .word với tham số vào gồm:
 1. Địa chỉ đầu của mảng
 2. Số dòng
 3. Số cột
7. Tiếp theo câu 6, 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.

2. Phân tích đề bài

- Hiện 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.

3. Phân tích cách làm

- Sử dụng 1 thanh ghi để lưu địa chỉ của các biến con trỏ, nếu không phải kiểu word, sẽ ép giá trị thanh ghi đó tăng lên số chia hết cho 4 (addi với 3, andi với 0xfffffc). Đối với từng hàm riêng biệt cần chú ý lượng bộ nhớ cần cấp phát cho mỗi kiểu biến con trỏ là khác nhau.
- Để thực hiện copy chuỗi ký tự ta sẽ copy từng ký tự từ chuỗi mà người dùng nhập vào kết hợp với việc cấp phát bộ nhớ cho biến con trỏ kiểu Char.
- Để cấp phát bộ nhớ cho mảng hai chiều ta sẽ đưa về dạng cấp phát bộ nhớ cho mảng một chiều (malloc2) và sử dụng hàm malloc cho từng phần tử.

4. Mã nguồn

```
.data
CharPtr: .word 0 # Biến con trỏ, trỏ tới kiểu ascii
BytePtr: .word 0 # Biến con trỏ, trỏ tới kiểu Byte
WordPtr: .word 0 # Biến con trỏ, trỏ tới mảng kiểu Word
ArrayPtr: .word 0 # Biến con trỏ, trỏ tới mảng hai chiều
CharPtr1: .word 0 # Biến con trỏ, dùng trong yêu cầu copy chuỗi
CharPtr2: .word 0 # Biến con trỏ, dùng trong yêu cầu copy chuỗi
Newline: .ascii "\n" # Ký tự xuống dòng
row: .word 1
col: .word 1

menu: .ascii "\n1. Malloc CharPtr.\n2. Malloc BytePtr.\n3. Malloc WordPtr.\n4. Tra về giá trị của các biến con trỏ.\n5. Tra về địa chỉ của các biến con trỏ.\n6. Copy 2 con trỏ chuỗi ký tự.\n7. Tính toán lượng bộ nhớ đã cấp phát cho các biến mảng (malloc).\n8. Malloc2 (Mảng hai chiều kiểu .word).\n9. Set Array[i][j].\n10. Get Array[i][j].\nThoát nếu khác 1-10"

char_str: .ascii "\nNhập số phần tử của mảng kiểu Char: "
byte_str: .ascii "\nNhập số phần tử của mảng kiểu Byte: "
word_str: .ascii "\nNhập số phần tử của mảng kiểu Word: "
copy_str: .ascii "\nChuỗi đã được copy: "
nb_row: .ascii "\nNhập số hàng của mảng: "
nb_col: .ascii "\nNhập số cột của mảng: "
input_row: .ascii "\nNhập i (số thứ tự của dòng): "
input_col: .ascii "\nNhập j (số thứ tự của cột): "
input_val: .ascii "\nNhập giá trị gán cho phần tử của mảng: "
output_val: .ascii "\nGiá trị trả về: "
address_str: .ascii "\nĐịa chỉ của biến con trỏ CharPtr | BytePtr | WordPtr | ArrayPtr là: "
value_str: .ascii "\nGiá trị của biến con trỏ CharPtr | BytePtr | WordPtr | ArrayPtr là: "
malloc_str: .ascii "\nBộ nhớ đã cấp phát: "
bytes_str: .ascii " bytes"
input_str: .ascii "\nNhập vào một chuỗi ký tự: "
malloc_success: .ascii "\nCấp phát bộ nhớ thành công. Mảng bắt đầu tại địa chỉ: "
mal_error: .ascii "\nError: Số hàng hoặc số cột phải nhỏ hơn 1000"
big_error: .ascii "\nError: Số hàng hoặc số cột phải lớn hơn 0"
bound_error: .ascii "\nError: Ngoài vùng bộ nhớ cho phép của mảng"
null_error: .ascii "\nError: Chưa khởi tạo mảng"
overflow_error: .ascii "\nError: Giá trị input quá lớn (> 2000)"
toosmall_error: .ascii "\nError: Giá trị input quá nhỏ (< 0)"
zero_error: .ascii "\nError: Giá trị input phải khác 0"
```

```

string_copy: .space 100      # Xau copy

.kdata
# Luu gia tri la dia chi dau tien cua vung nho con trong
Sys_TopOfFree: .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 # get integer input value from user
    move $s0, $a0 # switch option
    beq $s0, 1, option1
    beq $s0, 2, option2
    beq $s0, 3, option3
    beq $s0, 4, option4
    beq $s0, 5, option5
    beq $s0, 6, option6
    beq $s0, 7, option7
    beq $s0, 8, option8
    beq $s0, 9, option9
    beq $s0, 10, option10
    j end

option1:      # Malloc Char
    la $a0, char_str
    jal integer_input
    jal check_input # kiem tra gia tri input (0 < input < 2000)
    move $a1, $a0 # Luu input (so phan tu cua mang) vao $a1
    la $a0, CharPtr # Luu dia chi cua CharPtr vao $a0
    li $a2, 1 # Kich thuoc Char = 1 byte
    jal malloc # Cap phat bo nho
    move $s0, $v0 # Luu gia tri tra ve cua ham malloc vao $s0
    la $a0, malloc_success # Thong bao cap phat thanh cong
    li $v0, 4 # print string service
    syscall
    move $a0, $s0 # Chuyen gia tri tu $s0 vao $a0
    li $v0, 34 # print integer in hexadecimal service
    syscall # in ra gia tri integer cua $a0
    j main

```

```

option2:          # Malloc Byte
    la $a0, byte_str
    jal integer_input
    jal check_input
    move $a1, $a0    # Luu input (so phan tu cua mang) vao $a1
    la $a0, BytePtr    # Luu dia chi cua BytePtr vao $a0
    li $a2, 1        # Kich thuoc Byte = 1 byte
    jal malloc        # Cap phat bo nho
    move $s0, $v0      # Luu gia tri tra ve cua ham malloc vao $s0
    la $a0, malloc_success # Thong bao cap phat thanh cong
    li $v0, 4         # print string service
    syscall
    move $a0, $s0      # Chuyen gia tri tu $s0 vao $a0
    li $v0, 34        # print integer in hexadecimal service
    syscall
    j main

```

```

option3:          # Malloc Word
    la $a0, word_str
    jal integer_input
    jal check_input
    move $a1, $a0    # Luu input (so phan tu cua mang) vao $a1
    la $a0, WordPtr    # Luu dia chi cua WordPtr vao $a0
    li $a2, 4        # Kich thuoc Word = 4 bytes
    jal malloc        # Cap phat bo nho
    move $s0, $v0      # Luu gia tri tra ve cua ham malloc vao $s0
    la $a0, malloc_success # Thong bao cap phat thanh cong
    li $v0, 4         # print string service
    syscall
    move $a0, $s0      # Chuyen gia tri tu $s0 vao $a0
    li $v0, 34        # print integer in hexadecimal service
    syscall
    j main

```

```

option4:
    la $a0, value_str
    li $v0, 4         # print string service
    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 ArrayPtr
    jal  print_value

j   main

option5:
    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 ArrayPtr
    jal  Ptr_addr
    jal  print_value

j   main

option6:
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 co the doc duoc = 100
    syscall
    la $a1, string_copy    # Load lai 1 lan
    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_TopOfFree

```



```

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 1 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                # print string service
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
la     $a0, Newline
syscall
j      main

option7:                     # Tinh luong bo nho da cap phat
la     $a0, malloc_str
li     $v0, 4                # print string service
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                # print string service
syscall
j      main

option8:                     # Cap phat bo nho cho mang 2 chieu Malloc2
la     $a0, nb_row
jal     integer_input        # Nhap vao so hang
move    $s0, $a0
la     $a0, nb_col
jal     integer_input        # Nhap vao so cot
move    $a1, $s0             # malloc2 input_row parameter

```

```

move  $a2, $a0          # malloc2 input_col parameter
la    $a0, ArrayPtr
jal   Malloc2           # Cap phat bo nho
move  $s0, $v0          # save return value of malloc
la    $a0, malloc_success
li    $v0, 4
syscall
move  $a0, $s0
li    $v0, 34
syscall                # In ra gia tri mang vua nhap
j     main

option9:                # Set[i][j]
la    $a0, ArrayPtr
lw    $s7, 0($a0)
beqz  $s7, nullptr      # if *ArrayPtr==0 error null pointer
la    $a0, input_row
jal   integer_input     # get row
move  $s0, $a0
la    $a0, input_col
jal   integer_input     # get col
move  $s1, $a0
la    $a0, input_val
jal   integer_input     # get val
move  $a3, $a0
move  $a1, $s0
move  $a2, $s1
move  $a0, $s7
jal   SetArray
j     main

option10:               # Get[i][j]
la    $a0, ArrayPtr
lw    $s1, 0($a0)
beqz  $s1, nullptr      # if *ArrayPtr == 0 return error null pointer
la    $a0, input_row
jal   integer_input     # get row
move  $s0, $a0          # $s0 = 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          # save return value of GetArray
la    $a0, output_val

```

```

li $v0, 4
syscall
move $a0, $s0
li $v0, 34
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, skip             # Neu khong phai kieu Word thi nay sang skip
    addi $t8, $t8, 3
    andi $t8, $t8, 0xffffffff    # gia tri luu tai $t8 luon la 1 so chia het cho 4
skip:
    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
    bgt $a1, 1000, mal_err  # kiem tra loi so luong
    bltz $a1, big_err       # kiem tra hang be hon 0
    bgt $a2, 1000, mal_err  # phan tu hang (cot) qua lon
    bltz $a2, big_err       # kiem tra cot be hon 0
    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 # Neu so cot vuot qua gioi han => error
    bge $a2, $s2, bound_err # Neu so hang vuot qua gioi han => 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 # Neu so cot vuot qua gioi han => error
    bge $a2, $s2, bound_err # Neu so hang vuot qua gioi han => 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: array}
# @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/ArrayPtr
    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: array}
# @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/ArrayPtr
    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_TopOfFree
    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_TopOfFree
    sub $v0, $t9, $t8    # Tra ve gia tri $v0 = luong bo nho da cap phat
    jr $ra

#-----
# Wrapper for syscall 51 (InputDialogInt)
# repeat if status value !=0
#-----
integer_input:
    move $t9, $a0
    li $v0, 51
    syscall
    beq $a1, 0, doneIn    # OK
    beq $a1, -2, end    # Cancel
    move $a0, $t9
    j integer_input
doneIn:
    jr $ra

#-----
# kiem tra gia tri nhap vao >0 va <2000
#-----
check_input:
    bge $a0, 2000, too_big
    beqz $a0, zero_err
    bltz $a0, too_small
    jr $ra
too_big:
    la $a0, overflow_error
    j error
too_small:
    la $a0, toosmall_error
    j error
zero_err:
    la $a0, zero_error
    j error

mal_err:    # In ra thong bao loi so luong malloc
    la $a0, mal_error

```

```

j  error

big_err:      # In ra thông báo lỗi số lượng malloc
    la $a0, big_error
j  error

bound_err:    # In ra thông báo lỗi chỉ số vượt ngoài phạm vi
    la $a0, bound_error
j  error

nullptr:      # In ra thông báo lỗi con trỏ rỗng ( null)
    la $a0, null_error
j  error

error:
    li $v0, 4    # In ra thông báo lỗi
    syscall
j  main

end:
    li $v0, 10   # Terminate
    syscall

```

5. Hình ảnh kết quả mô phỏng

- Menu lựa chọn:



- Cấp phát bộ nhớ cho biến con trỏ CharPtr

Lựa chọn 1:

Input ×




1. Malloc CharPtr.
2. Malloc BytePtr.
3. Malloc WordPtr.
4. Tra ve gia tri cua cac bien con tro.
5. Tra ve dia chi cua cac bien con tro.
6. Copy 2 con tro xau ki tu.
7. Tinh toan luong bo nho da cap phat cho cac bien dong (malloc).
8. Malloc2 (Mang hai chieu kieu .word).
9. Set Array[i][j].
10. Get Array[i][j].

Thoat neu khac 1-10

Cấp phát bộ nhớ cho 3 phần tử:

Input ×



Nhap so phan tu cua mang kieu Char:

Kết quả:

Cấp phát bộ nhớ thành công. Mạng bắt đầu tại địa chỉ: 0x90000004

- Cấp phát bộ nhớ cho biến con trỏ BytePtr


Input ×



1. Malloc CharPtr.
2. Malloc BytePtr.
3. Malloc WordPtr.
4. Tra ve gia tri cua cac bien con tro.
5. Tra ve dia chi cua cac bien con tro.
6. Copy 2 con tro xau ki tu.
7. Tinh toan luong bo nho da cap phat cho cac bien dong (malloc).
8. Malloc2 (Mang hai chieu kieu .word).
9. Set Array[i][j].
10. Get Array[i][j].

Thoat neu khac 1-10

Cấp phát bộ nhớ cho 4 phần tử
Input ×




Nhap so phan tu cua mang kieu Byte:

Kết quả:

Cấp phát bộ nhớ thành công. Mạng bắt đầu tại địa chỉ: 0x90000007

- Cấp phát bộ nhớ cho biến con trỏ WordPtr

Input ×




1. Malloc CharPtr.
2. Malloc BytePtr.
3. Malloc WordPtr.
4. Tra ve gia tri cua cac bien con tro.
5. Tra ve dia chi cua cac bien con tro.
6. Copy 2 con tro xau ki tu.
7. Tinh toan luong bo nho da cap phat cho cac bien dong (malloc).
8. Malloc2 (Mang hai chieu kieu .word).
9. Set Array[i][j].
10. Get Array[i][j].

Thoat neu khac 1-10

Cấp phát bộ nhớ cho 2 phần tử

Input ×



Nhap so phan tu cua mang kieu Word:

Kết quả:

Cấp phát bộ nhớ thành công. Mạng bắt đầu tại địa chỉ: 0x9000000c

- Lấy giá trị của các biến con trỏ

Input X

?

1. Malloc CharPtr.
2. Malloc BytePtr.
3. Malloc WordPtr.
4. Tra ve gia tri cua cac bien con tro.
5. Tra ve dia chi cua cac bien con tro.
6. Copy 2 con tro xau ki tu.
7. Tinh toan luong bo nho da cap phat cho cac bien dong (malloc).
8. Malloc2 (Mang hai chieu kieu .word).
9. Set Array[i][j].
10. Get Array[i][j].

Thoat neu khac 1-10

4

OK

Cancel

Kết quả:

Gia tri cua bien con tro CharPtr | BytePtr | WordPtr | ArrayPtr la: 0x90000004;0x90000007;0x9000000c;0x00000000;

- Lấy địa chỉ của các biến con trỏ

Input X

?

1. Malloc CharPtr.
2. Malloc BytePtr.
3. Malloc WordPtr.
4. Tra ve gia tri cua cac bien con tro.
5. Tra ve dia chi cua cac bien con tro.
6. Copy 2 con tro xau ki tu.
7. Tinh toan luong bo nho da cap phat cho cac bien dong (malloc).
8. Malloc2 (Mang hai chieu kieu .word).
9. Set Array[i][j].
10. Get Array[i][j].

Thoat neu khac 1-10

5

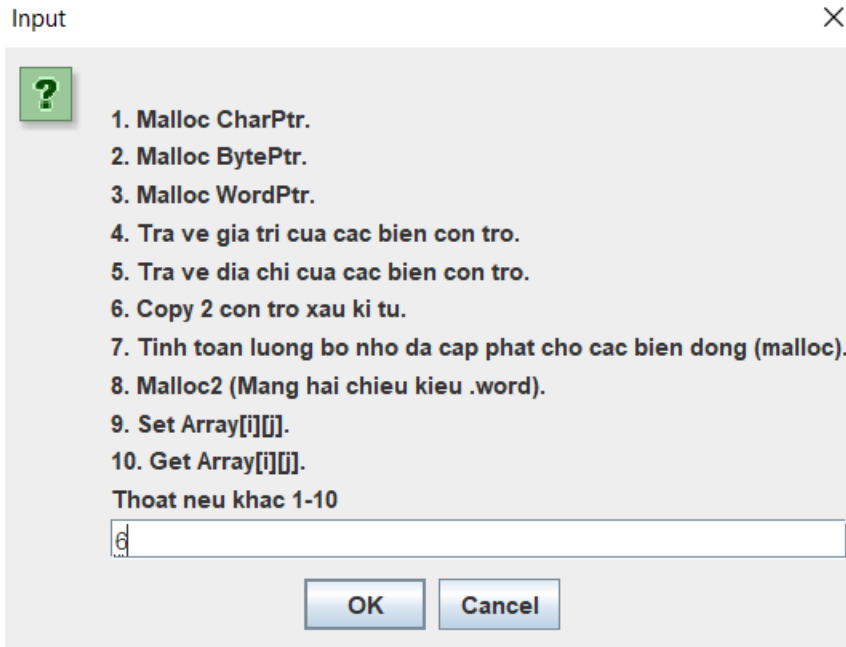
OK

Cancel

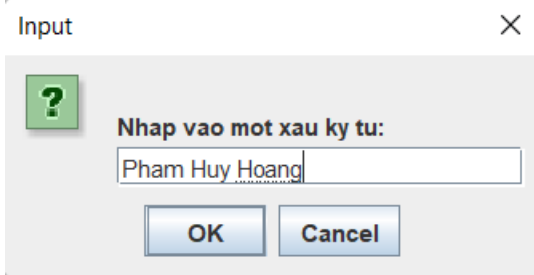
kết quả:

Địa chỉ của biến con trỏ CharPtr | BytePtr | WordPtr | ArrayPtr la: 0x10010000;0x10010004;0x10010008;0x1001000c;

- Copy xâu



Nhập xâu đầu vào: Pham Huy Hoang



Kết quả:

Xau da duoc copy: Pham Huy Hoang

- Tính toán lượng bộ nhớ đã cấp phát

Input ×



1. Malloc CharPtr.
2. Malloc BytePtr.
3. Malloc WordPtr.
4. Tra ve gia tri cua cac bien con tro.
5. Tra ve dia chi cua cac bien con tro.
6. Copy 2 con tro xau ki tu.
7. Tinh toan luong bo nho da cap phat cho cac bien dong (malloc).
8. Malloc2 (Mang hai chieu kieu .word).
9. Set Array[i][j].
10. Get Array[i][j].

Thoat neu khac 1-10

Kết quả: Do ta cấp phát 3 bytes cho 3 phần tử Char, 4 bytes cho 4 phần tử Byte thì thành 7 bytes nhưng vì khi cấp phát cho phần tử Word cần địa chỉ chia hết cho 4 nên tăng lên thành 8 và 8 bytes cho 2 phần tử Word nên tổng cộng bộ nhớ được cấp phát là 16 bytes.

Bộ nhớ đã cấp phát: 16 bytes

- Cấp phát bộ nhớ cho mảng 2 chiều

Input ×



1. Malloc CharPtr.
2. Malloc BytePtr.
3. Malloc WordPtr.
4. Tra ve gia tri cua cac bien con tro.
5. Tra ve dia chi cua cac bien con tro.
6. Copy 2 con tro xau ki tu.
7. Tinh toan luong bo nho da cap phat cho cac bien dong (malloc).
8. Malloc2 (Mang hai chieu kieu .word).
9. Set Array[i][j].
10. Get Array[i][j].

Thoat neu khac 1-10

Nhập số hàng là 5

Nhập số cột là 5

Input
X

?
Nhập số hàng của mảng:

OK Cancel

Input
X

?
Nhập số cột của mảng:

OK Cancel

Kết quả:

Cấp phát bộ nhớ thành công. Mảng bắt đầu tại địa chỉ: 0x90000014

- Thiết lập giá trị các phần tử cho mảng 2 chiều

Input
X

?

1. Malloc CharPtr.
2. Malloc BytePtr.
3. Malloc WordPtr.
4. Tra về giá trị của các biến con trỏ.
5. Tra về địa chỉ của các biến con trỏ.
6. Copy 2 con trỏ vào kí tự.
7. Tính toán lượng bộ nhớ đã cấp phát cho các biến động (malloc).
8. Malloc2 (Mảng hai chiều kiểu .word).
9. Set Array[i][j].
10. Get Array[i][j].

Thoát nếu khác 1-10

OK Cancel

Nhập số thứ tự của dòng (i)

Input
X

?
Nhập i (số thứ tự của dòng):

OK Cancel

Nhập số thứ tự của cột (j)

Input
X

?
Nhập j (số thứ tự của cột):

OK Cancel

Nhập giá trị cho phần tử Array[3][3] bằng 12:

Input

?

Nhap gia tri gan cho phan tu cua mang:

12

OK Cancel

- Lấy giá trị của các phần tử mảng 2 chiều

Input

?

1. Malloc CharPtr.
2. Malloc BytePtr.
3. Malloc WordPtr.
4. Tra ve gia tri cua cac bien con tro.
5. Tra ve dia chi cua cac bien con tro.
6. Copy 2 con tro xau ki tu.
7. Tinh toan luong bo nho da cap phat cho cac bien dong (malloc).
8. Malloc2 (Mang hai chieu kieu .word).
9. Set Array[i][j].
10. Get Array[i][j].

Thoat neu khac 1-10

10

OK Cancel

Input

?

Nhap i (so thu tu cua dong):

3

OK Cancel

Input

?

Nhap j (so thu tu cua cot):

3

OK Cancel

Kết quả:

Gia tri tra ve: 0x0000000c

II. Bài 4:

CNC Marsbot

6. Đề bài

Hãy lập trình để CNC Marsbot có thể:

- Thực hiện cắt kim loại như đã mô tả
- Nội dung postscript được lưu trữ cố định bên trong mã nguồn
- Mã nguồn chứa 3 postscript và người dùng sử dụng 3 phím 0, 4, 8 trên bàn phím Key Matrix để chọn postscript nào sẽ được gia công.
- Một postscript chứa chữ DCE cần gia công. Hai script còn lại sinh viên tự đề xuất (tối thiểu 10 đường cắt)

7. Phân tích đề bài

- o Khi nhập từ bàn phím vào các phím đã cài đặt trước để marsbot chạy
- o Nếu không phải các phím đã quy định trước thì sẽ không chạy
- o 0 là “DCE”; 4 là “DAT”; 8 là “DUY”

8. Phân tích cách làm

- o Xây dựng tọa độ cho postscript muốn dùng marsbot để cắt ra
- o Sử dụng HEXA_KEYBOARD để nhận lệnh người dùng
- o Đọc các giá trị có trong postscript và tiến hành thực hiện
- o Dừng lại khi đã gia công xong

9. Mã nguồn

```
.
# Mars bot
.eqv HEADING 0xffff8010
.eqv MOVING 0xffff8050
.eqv LEAVETRACK 0xffff8020
.eqv WHEREX 0xffff8030
.eqv WHEREY 0xffff8040
# Key matrix
.eqv OUT_ADRESS_HEXa_KEYBOARD 0xFFFF0014
.eqv IN_ADRESS_HEXa_KEYBOARD 0xFFFF0012

.data
# postscript-DCE => numpad 0
# (rotate,time,0=untrack | 1=track;
pscript1: .asciiz
"90,2000,0;180,3000,0;180,5790,1;80,500,1;70,500,1;60,500,1;50,500,1;40,500,1;30,500,1;20,500,1;10,500,1;0,500,1;350,500,1;340,500,1;330,500,1;320,500,1;310,500,1;300,500,1;290,500,1;280,490,1;90,7000,0;270,500,1;260,500,1;250,500,1;240,500,1;230,500,1;220,500,1;210,500,1;200,500,1;190,500,1;180,500,1;170,500,1;160,500,1;150,500,1;140,500,1;130,500,1;120,500,1;110,500,1;100,500,1;90,1000,1;90,5000,0;270,2000,1;0,5800,1;90,2000,1;180,2900,0;270,2000,1;90,3000,0;"
# postscript-DAT => numpad 4
pscript2: .asciiz
"90,2000,0;180,3000,0;180,5790,1;80,500,1;70,500,1;60,500,1;50,500,1;40,500,1;30,500,1;20,500,1;10,500,1;0,500,1;350,500,1;340,500,1;330,500,1;320,500,1;310,500,1;300,500,1;290,500,1;280,490,1;90,7000,0;200,6020,1;90,4160,0;340,6020,1;200,3000,0;90,2000,1;90,5000,0;180,2900,0;0,5500,1;270,2500,0;90,5000,1;90,1000,0;"
# postscript-DUY => numpad 8
pscript3: .asciiz
"90,2000,0;180,3000,0;180,5790,1;80,500,1;70,500,1;60,500,1;50,500,1;40,500,1;30,500,1;20,500,1;10,500,1;0
```



```
,500,1;350,500,1;340,500,1;330,500,1;320,500,1;310,500,1;300,500,1;290,500,1;280,490,1;90,5000,0;180,550
0,1;90,3000,1;0,5500,1;90,3000,0;150,2500,1;30,2500,1;210,2600,0;180,3100,1;180,1000,0;"
```

```
.text
```

```
# <--xu ly tren keymatrix-->
```

```
li $t3, IN_ADRESS_HEXА_KEYBOARD
```

```
li $t4, OUT_ADRESS_HEXА_KEYBOARD
```

```
polling:
```

```
li $t5, 0x01 # row-1 of key matrix
```

```
sb $t5, 0($t3)
```

```
lb $a0, 0($t4)
```

```
bne $a0, 0x11, NOT_NUMPAD_0
```

```
la $a1, pscript1
```

```
j START
```

```
NOT_NUMPAD_0:
```

```
li $t5, 0x02 # row-2 of key matrix
```

```
sb $t5, 0($t3)
```

```
lb $a0, 0($t4)
```

```
bne $a0, 0x12, NOT_NUMPAD_4
```

```
la $a1, pscript2
```

```
j START
```

```
NOT_NUMPAD_4:
```

```
li $t5, 0x04 # row-3 of key matrix
```

```
sb $t5, 0($t3)
```

```
lb $a0, 0($t4)
```

```
bne $a0, 0x14, COME_BACK
```

```
la $a1, pscript3
```

```
j START
```

```
COME_BACK: j polling # khi cac so 0,4,8 khong duoc chon -> quay lai doc tiep
```

```
# <!--end-->
```

```
# <--xu li mars bot -->
```

```
START:
```

```
jal GO
```

```
READ_PSCRIPT:
```

```
addi $t0, $zero, 0 # luu gia tri rotate
```

```
addi $t1, $zero, 0 # luu gia tri time
```

```
READ_ROTATE:
```

```
add $t7, $a1, $t6 # dich bit
```

```
lb $t5, 0($t7) # doc cac ki tu cua pscript
```

```
beq $t5, 0, END # ket thuc pscript
```

```
beq $t5, 44, READ_TIME # gap ki tu ','
```

```
mul $t0, $t0, 10
```

```
addi $t5, $t5, -48 # So 0 co thu tu 48 trong bang ascii.
```

```
add $t0, $t0, $t5 # cong cac chu so lai voi nhau.
```

```

addi $t6, $t6, 1 # tang so bit can dich chuyen len 1
j READ_ROTATE # quay lai doc tiep den khi gap dau ';'

```

```

READ_TIME: # doc thoi gian chuyen dong.
add $a0, $t0, $zero
jal ROTATE
addi $t6, $t6, 1
add $t7, $a1, $t6 # ($a1 luu dia chi cua pscript)
lb $t5, 0($t7)
beq $t5, 44, READ_TRACK
mul $t1, $t1, 10
addi $t5, $t5, -48
add $t1, $t1, $t5
j READ_TIME # quay lai doc tiep den khi gap dau ';'

```

```

READ_TRACK:
addi $v0, $zero, 32 # Keep mars bot running by sleeping with time=$t1
add $a0, $zero, $t1
addi $t6, $t6, 1
add $t7, $a1, $t6
lb $t5, 0($t7)
addi $t5, $t5, -48
beq $t5, $zero, CHECK_UNTRACK # 1=track | 0=untrack
jal UNTRACK
jal TRACK
j INCREMENT

```

CHECK_UNTRACK:

```

jal UNTRACK

```

INCREMENT:

```

syscall
addi $t6, $t6, 2 # bo qua dau ';'
j READ_PSCRIPT

```

GO:

```

li $at, MOVING
addi $k0, $zero, 1
sb $k0, 0($at)
jr $ra

```

STOP:

```

li $at, MOVING
sb $zero, 0($at)
jr $ra

```

TRACK:

```

li $at, LEAVETRACK
addi $k0, $zero, 1
sb $k0, 0($at)
jr $ra

```

UNTRACK:

```

li $at, LEAVETRACK
sb $zero, 0($at)
jr $ra

```

ROTATE:

```

li $at, HEADING
sw $a0, 0($at)
jr $ra

```

END:

```

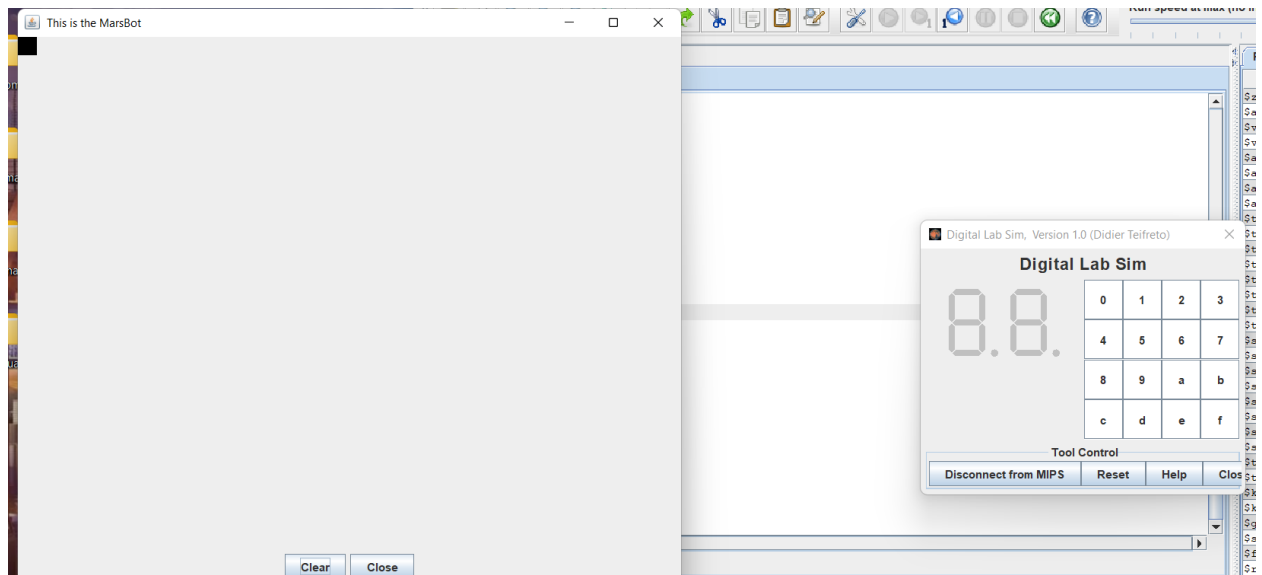
jal STOP
li $v0, 10
syscall
j polling

```

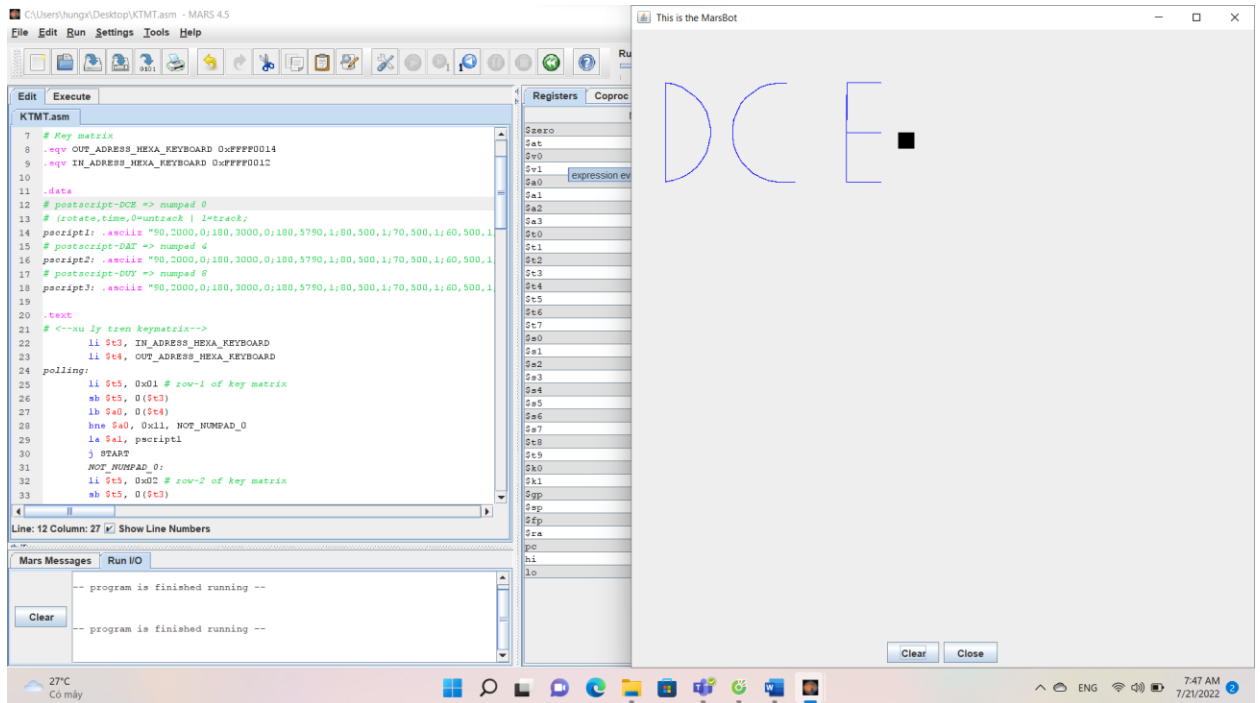
<!--end-->

10. Hình ảnh kết quả mô phỏng

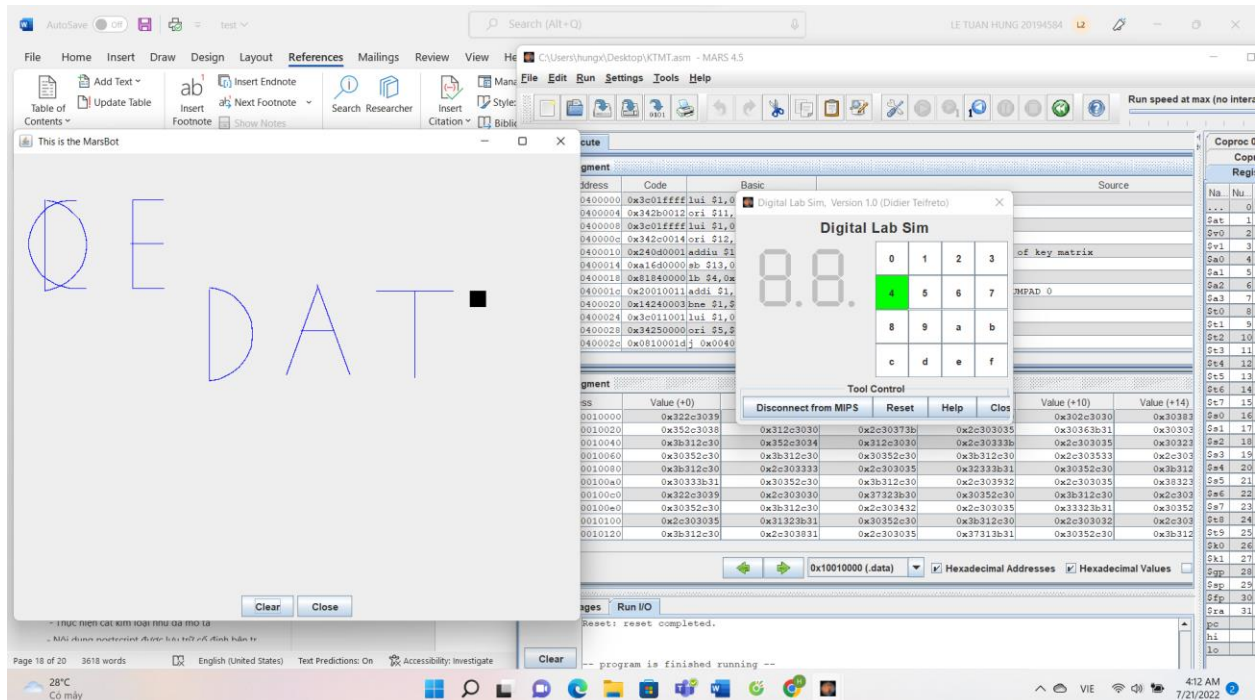
- Đầu tiên Marbos sẽ ở chế độ chờ khi đợi chọn số từ người dùng:



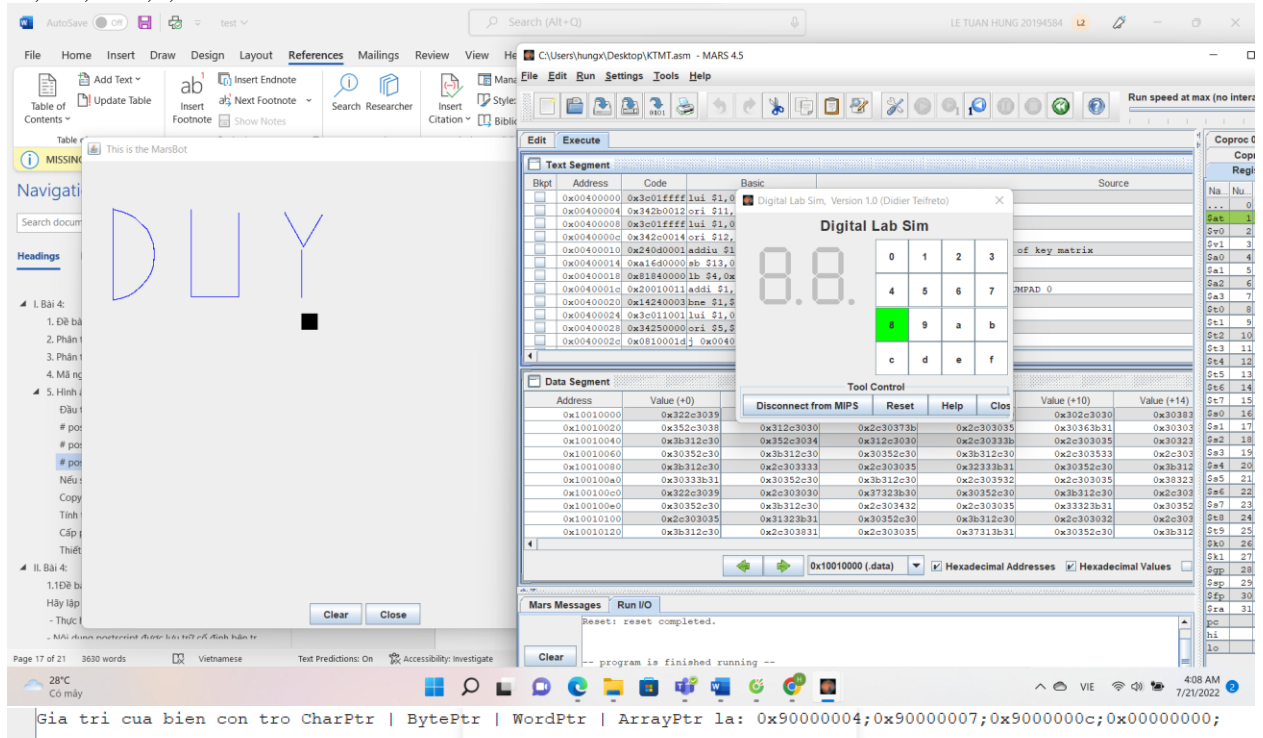
- # postscript-DCE => numpad 0
 postscript1: .asciiz"90,2000,0;180,3000,0;180,5790,1;80,500,1;70,500,1;60,500,1;50,500,1;40,500,1;30,500,1;20,500,1;10,500,1;0,500,1;350,500,1;340,500,1;330,500,1;320,500,1;310,500,1;300,500,1;290,500,1;280,490,1;90,2000,0;270,500,1;260,500,1;250,500,1;240,500,1;230,500,1;220,500,1;210,500,1;200,500,1;190,500,1;180,500,1;170,500,1;160,500,1;150,500,1;140,500,1;130,500,1;120,500,1;110,500,1;100,500,1;90,1000,1;90,5000,0;270,2000,1;0,5800,1;90,2000,1;180,2900,0;270,2000,1;90,3000,0;"



- # postscript-DAT => numpad 4
 poscript2: .asciii"90,2000,0;180,3000,0;180,5790,1;80,500,1;70,500,1;60,500,1;50,500,1;40,500,1;30,500,1;20,500,1;10,500,1;0,500,1;350,500,1;340,500,1;330,500,1;320,500,1;310,500,1;300,500,1;290,500,1;280,490,1;90,7000,0;200,6020,1;90,4160,0;340,6020,1;200,3000,0;90,2000,1;90,5000,0;180,2900,0;0,5500,1;270,2500,0;90,5000,1;90,1000,0;"



- # postscript-DUY -numpad8 pscript3: .asciiz
 "90,2000,0;1,80,3000,0;180,5790,1;80,500,1;70,500,1;60,500,1;50,500,1;40,500,1;30,500,1;20,500,1;
 10,500,1;0,500,1;350,500,1;340,500,1;330,500,1;320,500,1;310,500,1;300,500,1;290,500,1;280,490
 ,1;90,5000,0;180,5500,1;90,3000,1;0,5500,1;90,3000,0;150,2500,1;30,2500,1;210,2600,0;180,3100,
 1;180,1000,0;"



- Nếu số được chọn không phải 0,4,8

