BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH

(Final Project)

Nhóm: 1

Thành viên: Đinh Quốc Cường – 20184054

Phạm Tiến Đạt – 20184064

1. **Bài finalproject 4: Postscript CNC Marsbot**

**A.Đề bài :**

Máy gia công cơ khí chính xác CNC Marsbot được dùng để cắt tấm kim loại theo các đường nét được qui định trước. CNC Marsbot có một lưỡi cắt dịch chuyển trên tấm kim loại, với giả định rằng:

Nếu lưỡi cắt dịch chuyển nhưng không cắt tấm kim loại, tức là Marsbot  di chuyển nhưng không để lại vết (Track).

Nếu lưỡi cắt dịch chuyển và cắt tấm kim loại, tức là Marsbot  di chuyển và có để lại vết.

Để điều khiển Marsbot cắt đúng như hình dạng mong muốn, người ta  nạp vào Marsbot một mảng cấu trúc gồm 3 phần tử:

<Góc chuyển động>, <Thời gian>, <Cắt/Không cắt>

Trong đó <Góc chuyển động> là góc của hàm HEADING của Marsbot.

<Thời gian> là thời gian duy trì quá trình vận hành hiện tại.

<Cắt/Không cắt> thiết lập lưu vết/không lưu vết.

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).

**B.Cách thực hiện:**

- Thiết kế 3 postscript.

- Tạo hàm để check các phím 0,4,8. Nếu nhập vào 0, 4 hoặc 8 thì sẽ tiến hành lưu địa chỉ của postscript tương ứng.

- Xử lí Mars Bot , đọc postscript, đọc góc chuyển động, thời gian, cắt< không cắt>. Sử dụng 3 phím 0, 4, 8 trên bàn phím Key Matrix để chọn postscript nào sẽ được gia công.

- Phím 0 : postscript chứa chữ DCE

- Phím 4 : postscript chứa chữ CUONG

- Phím 8 : postscript chứa chữ TDAT

**C: Mã nguồn:**

# Mars bot

.eqv HEADING 0xffff8010 #Integer: 1 goc giua 0-359

.eqv MOVING 0xffff8050 #Boolean: co di chuyen hay khong di chuyen

.eqv LEAVETRACK 0xffff8020 #Boolean (1/0): co hay khong de lai track

# .eqv thay gia tri bang bien

# Key matrix

.eqv OUT\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0014

.eqv IN\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0012

.data

#.asciiz Luu chuoi gia tri

# postscript-DCE => num 0

# (rotate,time,0=untrack | 1=track;)

postscript1: .asciiz "90,3000,0;180,4000,0;180,5800,1;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;90,6000,0;260,500,1;260,500,1;260,500,1;200,500,1;200,500,1;200,500,1;200,500,1;200,500,1;160,500,1;160,500,1;160,500,1;160,500,1;160,500,1;160,500,1;90,500,1;90,500,1;90,500,1;90,500,1;90,3800,0;270,2400,1;0,5600,1;90,2400,1;180,2600,0;270,2400,1;90,2400,0;"

# postscript-CUONG => num 4

postscript2: .asciiz "90,3000,0;180,4000,0;260,500,1;260,500,1;260,500,1;200,500,1;200,500,1;200,500,1;200,500,1;200,500,1;160,500,1;160,500,1;160,500,1;160,500,1;160,500,1;160,500,1;90,500,1;90,500,1;90,500,1;90,500,1;90,900,0;0,5600,1;180,5600,0;90,2800,1;0,5600,1;90,3700,0;290,500,1;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;90,500,1;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;350,500,1;330,500,1;320,500,1;310,500,1;300,500,1;290,500,1;280,500,1;270,500,1;90,3800,0;180,5600,1;0,5600,0;150,6200,1;0,5600,1;90,3600,0;290,500,1;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;90,500,1;80,500,1;70,500,1;60,500,1;50,500,1;0,500,1;0,500,1;0,500,1;0,500,1;270,900,1;90,900,0;90,900,1;90,100,0;"

# postscript-TDAT => num 8

postscript3: .asciiz "90,3000,0;180,4000,0;180,5800,1;0,5800,1;270,1200,1;90,1200,0;90,1200,1;90,1000,0;180,5800,1;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;90,2100,0;180,5800,0;20,6050,1;160,6050,1;340,3100,0;270,1950,1;90,1200,0;180,3000,0;90,2050,0;0,5800,1;270,1200,1;90,1200,0;90,1200,1;"

.text

# key matrix

li $t3, IN\_ADDRESS\_HEXA\_KEYBOARD #gan 2 cai t3, t4

li $t4, OUT\_ADDRESS\_HEXA\_KEYBOARD

polling:

li $t5, 0x01 # dong1 key matrix

        sb $t5, 0($t3)

lb $a0, 0($t4)                        # lay phim 0

bne $a0, 0x11, NOT\_NUM\_0              # 0 thi doc ps1 / not 0 -> check num #

la $a1, postscript1

j START

NOT\_NUM\_0:

li $t5, 0x02 # dong2 key matrix

sb $t5, 0($t3)                         #luu tru bit

lb $a0, 0($t4)                         # lay phim 4

bne $a0, 0x12, NOT\_NUM\_4               # 4 thi doc ps2 / not 4 -> check num #

la $a1, postscript2

j START

NOT\_NUM\_4:

li $t5, 0X04 # dong3 key matrix

sb $t5, 0($t3)

lb $a0, 0($t4)                         # lay phim 8

bne $a0, 0x14, COME\_BACK               #

la $a1, postscript3

j START

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

# xu li mars bot

START:

jal GO

READ\_POSTSCRIPT:

addi $t0, $zero, 0 # luu gia tri goc quay

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

READ\_ROTATE:                                   # doc goc quay

  add $t7, $a1, $t6 # dich tung ki tu    # start t6=0    #$a1: dia chi postscript

        lb $t5, 0($t7)  # doc cac ki tu cua postscript   #$t5: gia tri so nhap vao

beq $t5, 0, END # khong gap ki tu nao, ket thuc postscript

  beq $t5, 44, READ\_TIME # gap ki tu ','

  mul $t0, $t0, 10                       # x10

  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 ky tu 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                    # gan a0=t0

jal ROTATE

  addi $t6, $t6, 1

  add $t7, $a1, $t6 # ($a1 luu dia chi cua postscript)

lb $t5, 0($t7) # doc cac ki tu cua postscript #$t5: gia tri so nhap vao

beq $t5, 44, READ\_TRACK # gap ki tu ','

mul $t1, $t1, 10

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

  add $t1, $t1, $t5 # cong cac chu so lai voi nhau.

  j READ\_TIME # quay lai doc tiep den khi gap dau ','

READ\_TRACK:                                    # cat

  addi $v0,$zero,32 # Keep mars bot running by sleeping with time=$t1

  add $a0, $zero, $t1                    #t1 time

  addi $t6, $t6, 1

  add $t7, $a1, $t6 # ($a1 luu dia chi cua postscript)

lb $t5, 0($t7) # doc cac ki tu cua postscript #$t5: gia tri so nhap vao

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

  beq $t5, $zero, CHECK\_UNTRACK # 1=track | 0=untrack

        jal UNTRACK

jal TRACK

j INCREAMENT

CHECK\_UNTRACK:

jal UNTRACK

INCREAMENT:

syscall

  addi $t6, $t6, 2 # bo qua dau ';'

  j READ\_POSTSCRIPT

#-----------------------------------------------------------

# GO procedure, to start running

# param[in] none

#-----------------------------------------------------------

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 procedure, to stop running

# param[in] none

#-----------------------------------------------------------

STOP:

li $at, MOVING # change MOVING port to 0

  sb $zero, 0($at) # to stop

  nop

  jr $ra

nop

#-----------------------------------------------------------

# TRACK procedure, to start drawing line

# param[in] none

#-----------------------------------------------------------

TRACK:

li $at, LEAVETRACK # change LEAVETRACK port

  addi $k0, $zero,1 # to logic 1,

sb $k0, 0($at) # to start tracking

  nop

  jr $ra

  nop

#-----------------------------------------------------------

# UNTRACK procedure, to stop drawing line

# param[in] none

#-----------------------------------------------------------

UNTRACK:

li $at, LEAVETRACK # change LEAVETRACK port to 0

  sb $zero, 0($at) # to stop drawing tail

  nop

  jr $ra

  nop

#-----------------------------------------------------------

# ROTATE procedure, to rotate the robot

# param[in] $a0, An angle between 0 and 359

# 0 : North (up)

# 90: East (right)

# 180: South (down)

# 270: West (left)

#-----------------------------------------------------------

ROTATE:

li $at, HEADING # change LEAVETRACK port to 0

  sw $a0, 0($at) # to stop drawing tail

  nop

  jr $ra

  nop

END:

jal STOP

li $v0, 10

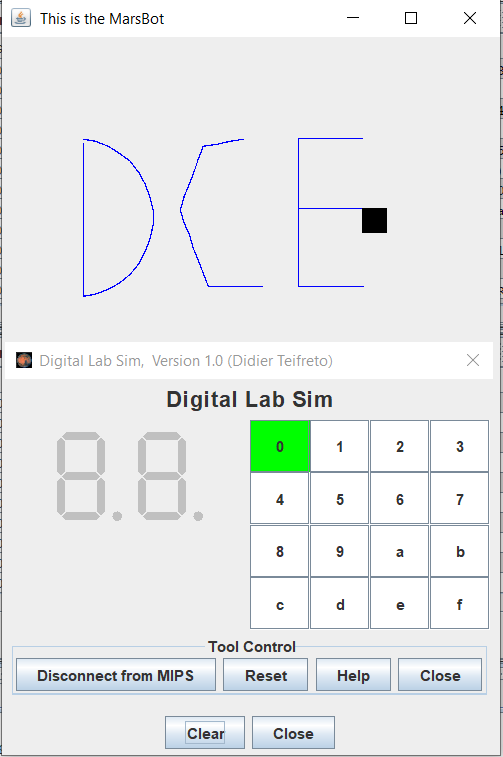
syscall

j polling

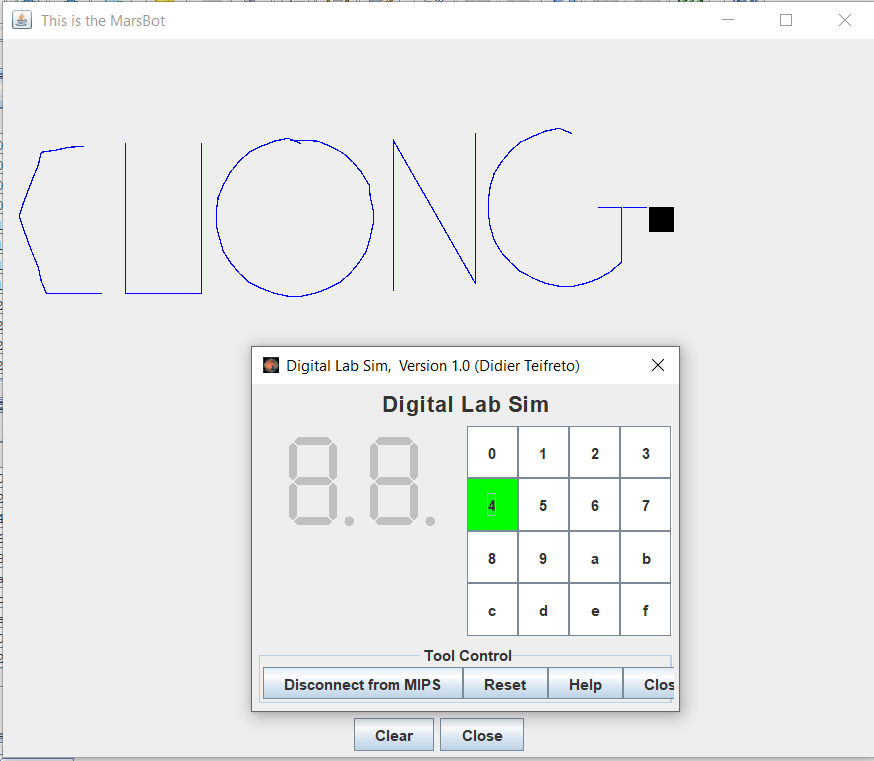
#end

**D.Kết quả:**

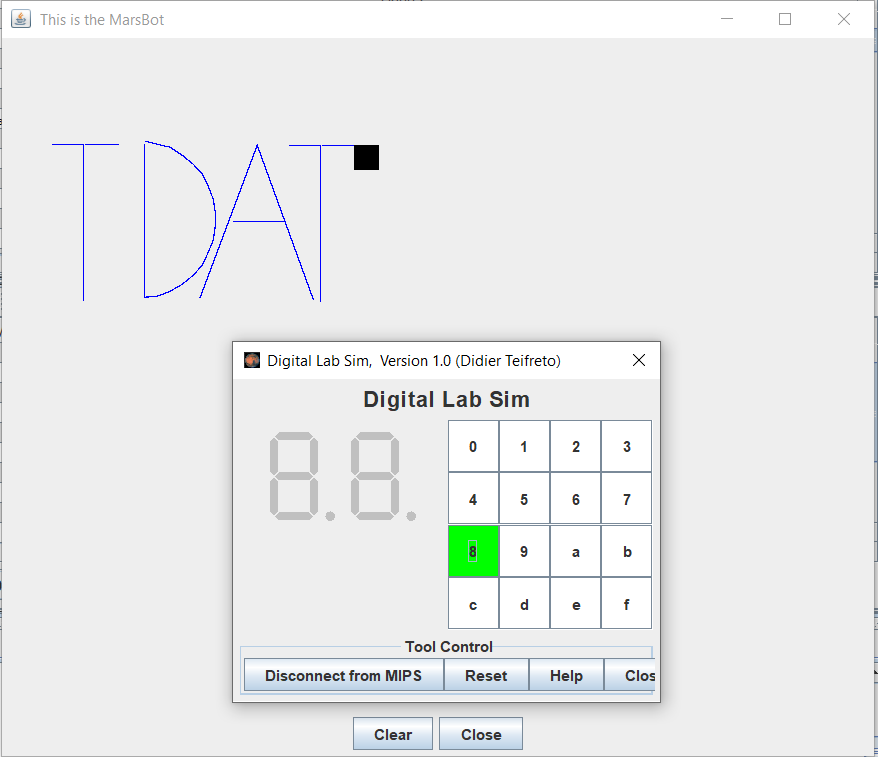
- Phím 0 : postscript chứa chữ DCE



- Phím 4 : postscript chứa chữ CUONG



- Phím 8 : postscript chứa chữ TDAT



1. **Bài finalproject 6: Hàm cấp phát bộnhớmalloc()**
2. **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 tựnhư \*CharPtr, \*BytePtr, \*WordPtr)

3)Viết hàm lấy địa chỉ biến con trỏ(tương tựnhư &CharPtr, &BytePtr, \*WordPtr)

4)Viết hàm thực hiện copy 2 con trỏ xâu kí 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:

a.Địa chỉ đầu của mảng

b.Số dòng

c.Số cột

7)Tiếp theo câu 6, hãy viết 2 hàm GetArray[i][j] và SetArray[i][j] đểlấy/thiết lập giá trị cho phần tử ở dòng I cột j của mảng.

**B.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.

**C.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
* Đối với từng hàm riêng biệt cần chú ý lượng bộ nhớ cần cấp phát cho mỗi kiểu biến con trỏ là khác nhau.
* Sử dụng cách thao tác stack của asembly để thao tác với mảng theo yêu cầuđề bài.  (trong hàm Malloc2).
* Để thực hiện copy xâu ký tự ta sẽ copy từng ký tự từ xâu 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 và sử dụng hàm malloc.

**D.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 mang kieu Word

ArrayPtr: .word 0 # Bien con tro, tro toi mang hai chieu

CharPtr1: .word 0 # Bien con tro, dung trong yeu cau copy xau

CharPtr2: .word 0 # Bien con tro, dung trong yeu cau copy xau

Newline: .asciiz "\n" # Ky tu xuong dong

row: .word 1

col: .word 1

menu: .asciiz "\n1. Cap phat bo nho kieu CharPtr.\n2. Cap phat bo nho kieu BytePtr.\n3. Cap phat bo nho kieu WordPtr.\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. Tinh toan luong bo nho da cap phat cho cac bien dong (malloc).\n8. Cap phat mang hai chieu kieu .word.\n9. Set Array[i][j].\n10. Get Array[i][j].\nThoat neu khac 1-10"

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: "

nb\_row: .asciiz "\nNhap so hang cua mang: "

nb\_col: .asciiz "\nNhap so cot cua mang: "

input\_row: .asciiz "\nNhap i (so thu tu cua dong): "

input\_col: .asciiz "\nNhap j (so thu tu cua 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 || ArrayPtr la: "

value\_str: .asciiz "\nGia tri cua bien con tro CharPtr | BytePtr | WordPtr | ArrayPtr la: "

malloc\_str: .asciiz "\nBo nho da cap phat: "

bytes\_str: .asciiz " bytes"

input\_str: .asciiz "\nNhap vao mot xau ky tu: "

malloc\_success: .asciiz "\nCap phat bo nho thanh cong.\n Mang bat dau tai dia chi: "

mal\_error: .asciiz "\nError: So hang hoac so cot phai nho hon 1000"

bound\_error: .asciiz "\nError: Ngoai vung bo nho cho phep cua mang"

null\_error: .asciiz "\nError: Chua khoi tao mang"

overflow\_error: .asciiz "\nError: Gia tri input qua lon (> 2000)"

zero\_error: .asciiz "\nError: Gia tri input phai khac 0"

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 # 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\_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 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 nhay sang skip

addi $t8, $t8, 3

andi $t8, $t8, 0xfffffffc # 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 tru 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

bgt $a2, 1000, mal\_err # phan tu hang (cot) qua lon

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\_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 # 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

beq $a1, -2, end

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

jr $ra

too\_big:

la $a0, overflow\_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

bound\_err: # In ra thong bao loi chi so vuot ngoai pham vi

la $a0, bound\_error

j error

nullptr: # In ra thong bao loi con tro rong ( null)

la $a0, null\_error

j error

error:

li $v0, 4 # In ra thong bao loi

syscall

j main

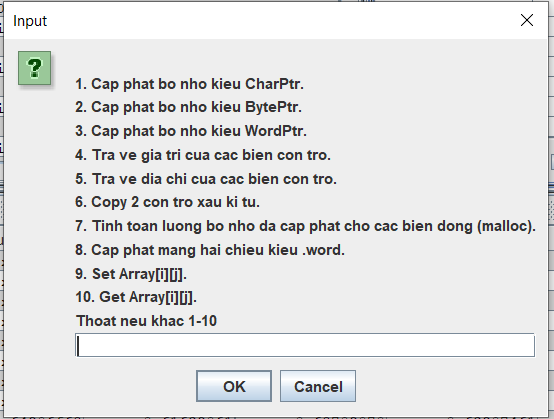
end:

li $v0, 10 # Terminate

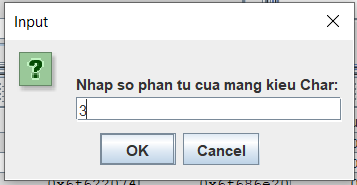
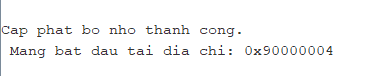
syscall

**E.Kết quả thực hiện:**

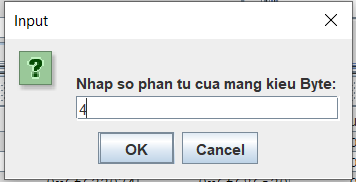
* Menu:

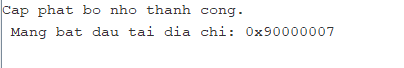


* Cấp phát dữ liệu:
* Kiểu Char:

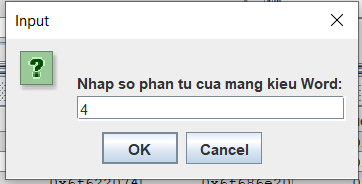
 

* Kiểu Byte:





* Kiểu Word:

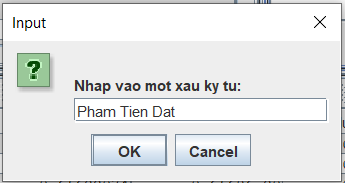


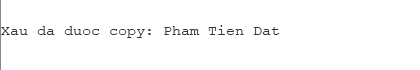


* Địa chỉ và giá trị của biến con trỏ:

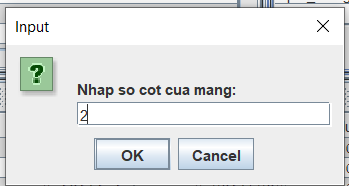
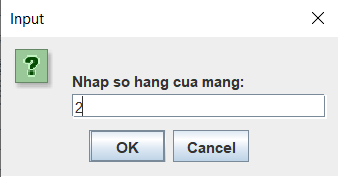


* Coppy 2 xâu kí tự:

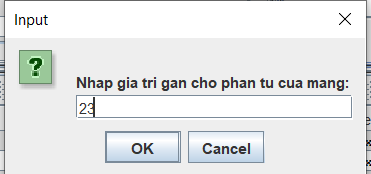
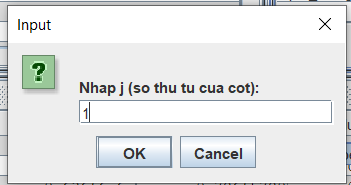
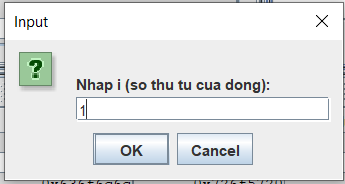




* Mảng 2 chiều:
* Cấp phát:



* SetArray[1][1]=23:



* GetArrray[1][1]:

