BÁO CÁO BÀI TẬP LỚN CUỐI KỲ

Mã học phần: IT3208

Tên học phần: Thực hành Kiến trúc máy tính

Giảng viên hướng dẫn: TS. Lê Bá Vui

Sinh Viên

Ngô Quang An Khánh - 20204991

Lê Thế Kỳ - 20205092

Ngày 22 tháng 7 năm 2022

MỤC LỤC

I. Bài 6: Hàm cấp phát bộ nhớ malloc() (Sinh viên thực hiện: Ngô Quang An Khánh)

1. Cách làm

- Xây dựng các hàm theo yêu cầu của đề bài.
- Viết chương trình cho phép kiểm tra các hàm vừa viết.

2. Thuật toán

- Câu 1:
  + Với trường hợp số byte bằng 4, kiểm tra địa chỉ đầu tiên trống có chia hết cho 4 hay không. Nếu có, chuyển qua bước cấp phát bộ nhớ.
  + Nếu không, di chuyển tới địa chỉ chia hết cho 4 lớn hơn và gần nhất bằng cách tìm số dư khi chia địa chỉ cho 4 bằng cách thực hiện phép logic AND với số nhị phân 11 (3 ở hệ thập phân), lấy địa chỉ cộng thêm 4 và trừ đi số dư vừa tìm được.ký
- Câu 2: Thực hiện load word với địa chỉ chính là biến con trỏ đang xét. Trả về giá trị thu được.
- Câu 3: Load address biến con trỏ. Trả về giá trị này.
- Câu 4: Mỗi lần con trỏ xâu ký tự nguồn ở vị trí một phần tử, sao chép giá trị của phần tử đó sang cho vị trí của con trỏ xâu ký tự đích, sau đó tăng giá trị từng con trỏ thêm 1 đơn vị. Lặp lại đến khi ký tự vừa được sao chép là ký tự '\0'.
- Câu 5: Đặt địa chỉ đầu tiên được cấp phát vào con trỏ trỏ đến địa chỉ đầu tiên còn trống
- Câu 6: Lấy địa chỉ đầu tiên còn trống (lưu trong con trỏ) trừ đi địa chỉ đầu tiên được cấp phát. Trả về hiệu của phép tính vừa rồi.
- Câu 7: Tính số phần tử bằng cách lấy số hàng nhân với số cột, sau đó thực hiện hàm malloc với số byte mỗi phần tử bằng 4 (kiểu word).
- Câu 8: Để tìm phần tử, ta lấy chỉ số i nhân với số cột của mảng, sau đó cộng thêm chỉ số j và địa chỉ của phần tử đầu.

+ getArray[i][j]: thực hiện load word tại địa chỉ tìm được. Trả về giá trị thu được.
+ setArray[i][j]: thực hiện save word giá trị cho trước vào địa chỉ tìm được.

3. Mã nguồn:

```
.data
CharPtr1:       .word   0       # Bien con tro, tro toi kieu asciiz
CharPtr2:       .word   0       # Bien con tro, tro toi kieu asciiz
ArrayPtr:       .word   0       # Bien con tro mang 1 chieu
Array2Ptr:      .word   0       # Bien con tro mang 2 chieu
message1:       .asciiz "\n\n1. Xu ly mang mot chieu\n"
message2:       .asciiz "2. Sao chep mang ky tu\n"
message3:       .asciiz "3. Xu ly mang hai chieu\n"
message4:       .asciiz "4. Giai phong bo nho\n"
message0.1:     .asciiz "So phan tu: "
message0.2:     .asciiz "So byte moi phan tu (1 hoac 4): "
message0.3:     .asciiz "Nhap phan tu: "
message1.1:     .asciiz "Gia tri cua con tro: "
message1.2:     .asciiz "\nDia chi cua con tro: "
message1.3:     .asciiz "\nTong dia chi da cap phat: "
message2.1:     .asciiz "So ky tu toi da: "
message2.2:     .asciiz "\nNhap chuoi ky tu: "
message2.3:     .asciiz "\nChuoi ky tu duoc copy: "
message3.1:     .asciiz "\nSo hang: "
message3.2:     .asciiz "\nSo cot: "
message3.3:     .asciiz "\n1. getArray[i][j]\n"
message3.4:     .asciiz "2. setArray[i][j]\n"
message3.5:     .asciiz "3. Thoat\n"
message3.6:     .asciiz "\nGia tri cua phan tu: "
message3.01:    .asciiz "i = "
message3.02:    .asciiz "j = "
message4.1:     .asciiz "Da giai phong toan bo bo nho cap phat.\n"
select:         .asciiz "Lua chon: "
errmessage:     .asciiz "\nSo vua nhap khong hop le.\n"

.kdata
# Bien chua dia chi dau tien cua vung nho con trong
```

```
Sys_TheTopOfFree:       .word   1
# Vung khong gian tu do, dung de cap bo nho cho cac bien con tro
Sys_MyFreeSpace:

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

        menu:
        li      $v0, 4
        la      $a0, message1
        syscall
        la      $a0, message2
        syscall
        la      $a0, message3
        syscall
        la      $a0, message4
        syscall
        la      $a0, select
        syscall
        li      $v0, 5
        syscall
case_1:
        bne     $v0, 1, case_2
        li      $v0, 4
        la      $a0, message0.1
        syscall
        li      $v0, 5
        syscall
        bltz    $v0, error
        move    $a1, $v0
        li      $v0, 4
        la      $a0, message0.2
```

```
        syscall
        li      $v0, 5
        syscall
is1:    beq     $v0, 1, ready
is4:    beq     $v0, 4, ready
        j       error
ready:  move    $a2, $v0
        la      $a0, ArrayPtr
        jal     malloc
        move    $t0, $v0
        li      $v0, 4
        la      $a0, message0.3
        syscall
        move    $a0, $t0
        add     $t0, $0, $0
input_loop:
        beq     $t0, $a1, input_end
        li      $v0, 5
        syscall
        bne     $a2, 1, byte_4
byte_1:
        sb      $v0, 0($a0)
        addi    $a0, $a0, 1
        addi    $t0, $t0, 1
        j       input_loop
byte_4:
        sw      $v0, 0($a0)
        addi    $a0, $a0, 4
        addi    $t0, $t0, 1
        j       input_loop
input_end:
        li      $v0, 4
        la      $a0, message1.1
```

```
        syscall
        la      $a0, ArrayPtr
        jal     getValue
        move    $a0, $v0
        li      $v0, 34
        syscall
        li      $v0, 4
        la      $a0, message1.2
        syscall
        la      $a0, ArrayPtr
        jal     getAddress
        move    $a0, $v0
        li      $v0, 34
        syscall
        li      $v0, 4
        la      $a0, message1.3
        syscall
        jal     memoryCalculate
        move    $a0, $v0
        li      $v0, 1
        syscall
        j       menu
case_2:
        bne     $v0, 2, case_3
        li      $v0, 4
        la      $a0, message2.1
        syscall
        li      $v0, 5
        syscall
        move    $a1, $v0
        addi    $a2, $0, 1
        la      $a0, CharPtr1
        jal     malloc
```

```
        move    $s0, $v0
        la      $a0, CharPtr2
        jal     malloc
        move    $s1, $v0
        li      $v0, 4
        la      $a0, message2.2
        syscall
        move    $a0, $s0
        li      $v0, 8
        syscall
        move    $a1, $s1
        jal     strcpy
        li      $v0, 4
        la      $a0, message2.3
        syscall
        move    $a0, $s1
        syscall
        j       menu
case_3:
        bne     $v0, 3, case_4
        li      $v0, 4
        la      $a0, message3.1
        syscall
        li      $v0, 5
        syscall
        move    $a1, $v0
        li      $v0, 4
        la      $a0, message3.2
        syscall
        li      $v0, 5
        syscall
        move    $a2, $v0
        la      $a0, Array2Ptr
```

```
        jal     malloc2
        move    $t0, $v0
        li      $v0, 4
        la      $a0, message0.3
        syscall
        move    $a0, $t0
        add     $t0, $0, $0
        move    $t1, $a1
        mul     $a1, $a1, $a2
input_loop2:
        beq     $t0, $a1, input_end2
        li      $v0, 5
        syscall
        sw      $v0, 0($a0)
        addi    $a0, $a0, 4
        addi    $t0, $t0, 1
        j       input_loop2
input_end2:
        move    $a1, $t1
sub_menu:
        li      $v0, 4
        la      $a0, message3.3
        syscall
        la      $a0, message3.4
        syscall
        la      $a0, message3.5
        syscall
        la      $a0, select
        syscall
        li      $v0, 5
        syscall
sub_case_1:
        bne     $v0, 1, sub_case_2
```

```
        li      $v0, 4
        la      $a0, message3.01
        syscall
        li      $v0, 5
        syscall
        move    $s0, $v0
        li      $v0, 4
        la      $a0, message3.02
        syscall
        li      $v0, 5
        syscall
        move    $s1, $v0
        la      $t0, Array2Ptr
        lw      $a0, 0($t0)
        jal     getArray
        move    $s2, $v0
        li      $v0, 4
        la      $a0, message3.6
        syscall
        li      $v0, 1
        move    $a0, $s2
        syscall
        j       sub_menu
sub_case_2:
        bne     $v0, 2, sub_case_3
        li      $v0, 4
        la      $a0, message3.01
        syscall
        li      $v0, 5
        syscall
        move    $s0, $v0
        li      $v0, 4
        la      $a0, message3.02
```

```
        syscall
        li      $v0, 5
        syscall
        move    $s1, $v0
        move    $s2, $v0
        li      $v0, 4
        la      $a0, message0.3
        syscall
        li      $v0, 5
        syscall
        la      $t0, Array2Ptr
        lw      $a0, 0($t0)
        jal     setArray
        j       sub_menu
sub_case_3:
        bne     $v0, 3, error
        j       menu
case_4:
        bne     $v0, 4, error
        jal     free
        li      $v0, 4
        la      $a0, message4.1
        syscall
        li      $v0, 4
        la      $a0, message1.3
        syscall
        jal     memoryCalculate
        move    $a0, $v0
        li      $v0, 1
        syscall
        j       menu
error:
        li      $v0, 4
```

```
        la      $a0, errmessage
        syscall
        j       menu
#-----------------------------------------
# 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, initialize  # Neu mang khoi tao co kieu Word, kiem tra dia chi dau co dam bao quy tac
khong
        andi    $t0, $t8, 0x03          # Lay so du khi chia dia chi trong cho 4
        beq     $t0, 0, initialize  # Neu khong co du, bo qua
        addi    $t8, $t8, 4             # Neu co, tien toi dia chi chia het cho 4 tiep theo
        subu    $t8, $t8, $t0
initialize:
        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 lay gia tri cua bien con tro
# @param	[in]		$a0: Chua dia chi cua bien con tro can lay gia tri
# @return			$v0: Gia tri cua bien con tro
#-----------------------------------------
getValue:
	lw	$v0, 0($a0)	# Lay gia tri cua bien con tro trong o nho co dia chi luu trong $a0
	jr	$ra
#-----------------------------------------
# Ham lay dia chi cua bien con tro
# @param	[in]		$a0: Chua dia chi cua bien con tro can lay dia chi
# @return			$v0: Dia chi cua bien con tro
#-----------------------------------------
getAddress:
	add	$v0, $0, $a0	# Lay dia chi tu $a0
	jr	$ra
#-----------------------------------------
# Ham copy 2 con tro xau ky tu
# @param	[in]	$a0: Chua dia chi cua bien con tro xau ky tu nguon
# @param	[in]	$a1: Chua dia chi cua bien con tro xau ky tu dich
#-----------------------------------------
strcpy:
	add	$t0, $0, $a0	# Khoi tao $t0 o dau xau ky tu nguon
	add	$t1, $0, $a1	# Khoi tao $t1 o dau xau ky tu dich
	addi	$t2, $0, 1	# Khoi tao $t2 la ky tu khac '\0' de chay vong lap
cpyLoop:
	beq	$t2, 0, cpyLoopEnd	# Neu ky tu duoc copy trong vong lap truoc la '\0', dung vong lap
	lb	$t2, 0($t0)		# Doc ky tu o xau ky tu nguon
	sb	$t2, 0($t1)		# Luu ky tu vua doc vao xau ky tu dich
	addi	$t0, $t0, 1		# Chuyen $t0 tro sang vi tri cua phan tu tiep theo trong xau ky tu nguon
```

```
        addi    $t1, $t1, 1             # Chuyen $t1 tro sang vi tri cua phan tu tiep theo trong xau ky tu dich
        j       cpyLoop
cpyLoopEnd:
        jr      $ra
#-----------------------------------------
# Ham giai phong bo nho da cap phat
# @param        khong co
#-----------------------------------------
free:
        addi    $sp, $sp, -4        # Khoi tao 1 vi tri trong stack
        sw      $ra, 0($sp)         # Luu $ra vao stack
        jal     SysInitMem          # Tai lap lai vi tri cua con tro luu dia chi dau tien con trong
        lw      $ra, 0($sp)         # Tra gia tri cho $ra
        addi    $sp, $sp, 4         # Xoa stack
#-----------------------------------------
# Ham tinh toan bo nho da cap phat
# @param        khong co
# @return       $v0: so byte da cap phat
#-----------------------------------------
memoryCalculate:
        la      $t0, Sys_MyFreeSpace    # Lay dia chi dau tien duoc cap phat
        la      $t1, Sys_TheTopOfFree   # Lay dia chi luu dia chi dau tien con trong
        lw      $t2, 0($t1)             # Lay dia chi dau tien con trong
        sub     $v0, $t2, $t0           # Tru hai dia chi cho nhau
        jr      $ra
#-----------------------------------------
# Ham cap phat bo nho cho mang word 2 chieu
# @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 can cap phat
# @param        [in]            $a2: So cot can cap phat
# @return                       $v0: Dia chi vung nho duoc cap phat
#-----------------------------------------
```

```
malloc2:
        addi    $sp, $sp, -12       # Luu cac gia tri can thiet de thuc hien 1 chuong trinh con malloc trong chuong
trinh con nay
        sw      $ra, 8($sp)
        sw      $a1, 4($sp)
        sw      $a2, 0($sp)
        mul     $a1, $a1, $a2       # $a1 = so phan tu = so hang * so cot
        addi    $a2, $0, 4          # $a2 = so byte cua 1 phan tu kieu word = 4
        jal     malloc              # Chuyen mang 2 chieu thanh mang 1 chieu, khoi tao
        lw      $ra, 8($sp)         # Tra lai gia tri cho cac thanh ghi
        lw      $a1, 4($sp)
        lw      $a2, 0($sp)
        addi    $sp, $sp, 12
        jr      $ra
#-----------------------------------------
# Ham lay gia tri cua phan tu trong mang 2 chieu
# @param        [in]            $a0: Chua dia chi cua bien con tro
# @param        [in]            $a1: So hang cua mang
# @param        [in]            $a2: So cot cua mang
# @param        [in]            $s0: Chi so i cua phan tu
# @param        [in]            $s1: Chi so j cua phan tu
# @return                       $v0: Gia tri cua phan tu
#-----------------------------------------
getArray:
        mul     $t0, $s0, $a2       # Vi tri cua phan tu = i * so cot + j
        add     $t0, $t0, $s1
        sll     $t0, $t0, 2         # Do phan tu co kieu word nen phai * 4 de ra khoang cach dia chi tuong doi so
voi dia chi dau
        add     $t0, $t0, $a0       # Cong dia chi dau de ra dia chi phan tu
        lw      $v0, 0($t0)         # Lay gia tri phan tu
        jr      $ra
#-----------------------------------------
# Ham dat gia tri cua phan tu trong mang 2 chieu
# @param        [in]            $a0: Chua dia chi cua bien con tro
```

```
# @param        [in]        $a1: So hang cua mang
# @param        [in]        $a2: So cot cua mang
# @param        [in]        $s0: Chi so i cua phan tu
# @param        [in]        $s1: Chi so j cua phan tu
# @param        [in]        $v0: Gia tri can dat vao phan tu
#-----------------------------------------
setArray:
        mul     $t0, $s0, $a2   # Vi tri cua phan tu = i * so cot + j
        add     $t0, $t0, $s1
        sll     $t0, $t0, 2     # Do phan tu co kieu word nen phai * 4 de ra khoang cach dia chi tuong doi so voi dia chi dau
        add     $t0, $t0, $a0   # Cong dia chi dau de ra dia chi phan tu
        sw      $v0, 0($t0)     # Dat gia tri phan tu
        jr      $ra
```

4. Kết quả:

- Câu 1, 2, 3, 6 (mảng mẫu kiểu Word, giá trị phần tử: 1, 7, 9, 11):

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x90000000	-1879048172	1	7	9	11	0	0	0
0x90000020	0	0	0	0	0	0	0	0
0x90000040	0	0	0	0	0	0	0	0
0x90000060	0	0	0	0	0	0	0	0
0x90000080	0	0	0	0	0	0	0	0
0x900000a0	0	0	0	0	0	0	0	0
0x900000c0	0	0	0	0	0	0	0	0
0x900000e0	0	0	0	0	0	0	0	0
0x90000100	0	0	0	0	0	0	0	0
0x90000120	0	0	0	0	0	0	0	0

```
So phan tu: 4
So byte moi phan tu (1 hoac 4): 4
Nhap phan tu: 1
7
9
11
Gia tri cua con tro: 0x90000004
Dia chi cua con tro: 0x10010008
Tong dia chi da cap phat: 16
```

- Câu 4 (xâu ký tự mẫu: "Thuc hanh kien truc may tinh"):

```
So ky tu toi da: 30

Nhap chuoi ky tu: Thuc hanh kien truc may tinh

Chuoi ky tu duoc copy: Thuc hanh kien truc may tinh
```

- Câu 7, 8 (mảng mẫu 2 dòng, 3 cột, giá trị phần tử: 1, 2, 3, 4, 5, 6):
  + getArray (lấy giá trị tại hàng 1, cột 1):

```
i = 1
j = 1

Gia tri cua phan tu: 5
```

  + setArray (thay đổi giá trị tại hàng 0, cột 2, thay đổi thành giá trị -4):

```
i = 0
j = 2
Nhap phan tu: -4

i = 0
j = 2

Gia tri cua phan tu: -4
```

- Câu 5:

Address	Value (+0)
0x90000000	0x90000004

II. Bài 4: Postscript CNC Marsbot (Sinh viên thực hiện: Lê Thế Kỳ)

1. Cách làm:

- Chuyển đổi postscript từ dạng xâu ký tự sang dạng word, tạo thành mảng để lưu vào bộ nhớ và lưu địa chỉ đầu vào các con trỏ theo cơ chế giống hàm malloc(), 3 số cuối của mảng được đặt bằng 0 để làm sentinel.
- Với mỗi lựa chọn postscript, lấy bộ 3 giá trị của bộ nhớ, truyền vào các địa chỉ của Marsbot.

2. Thuật toán:

- Bước setup: chuyển từng chỉ số của postscript thành số, sau đó lưu vào bộ nhớ, tạo thành 1 mảng. Đánh dấu địa chỉ đầu bằng 1 con trỏ. Kết thúc chuyển đổi khi gặp ký tự '\0' ở xâu ký tự.
- Kiểm tra nút được bấm trên bảng 4 x 4 bằng phương thức polling. Với mỗi nút, truyền con trỏ vào hàm đọc.
- Ở hàm đọc, đọc từng bộ 3 chỉ số được trỏ bởi con trỏ. Sau đó đặt 2 chỉ số đầu vào địa chỉ HEADING và LEAVETRACK, chỉ số thứ 3 lưu vào thanh ghi $a0 để chuẩn bị thực hiện lệnh syscall sleep.
- Kích hoạt di chuyển Marsbot bằng cách truyền giá trị 1 vào địa chỉ MOVING, sau đó thực hiện lệnh syscall sleep.
- Sau lệnh sleep, truyền giá trị 0 vào MOVING và LEAVETRACK.
- Di chuyển con trỏ lên 3 phần tử.
- Kết thúc khi 3 chỉ số cùng bằng 0 (kết quả phép logic AND của 3 chỉ số bằng 0).

3. Mã nguồn:

```
.eqv     HEADING                 0xffff8010      # Integer: An angle between 0 and 359
.eqv     MOVING                  0xffff8050      # Boolean: whether MarsBot move or not
.eqv     LEAVETRACK  0xffff8020          # Boolean (0 or non-0): whether MarsBot leave a track or not
.eqv     IN_ADDRESS_HEXA_KEYBOARD        0xFFFF0012
.eqv     OUT_ADDRESS_HEXA_KEYBOARD       0xFFFF0014


.data
ptr1:    .word    0          # Pointer to the data of postscript 1
ptr2:    .word    0          # Pointer to the data of postscript 2
ptr3:    .word    0          # Pointer to the data of postscript 3
ps1:     .asciiz
        "90,0,5000,180,0,8000,180,1,10000,75,1,3000,50,1,1500,25,1,1500,360,1,3000,350,1,1500,330,1,1500,310
,1,1500,270,1,2700,90,0,14000,270,1,4000,200,1,5000,160,1,5000,90,1,4000,90,0,3000,90,1,4000,270,0,4000,360,1
,9500,90,1,4000,270,0,4000,180,0,4750,90,1,4000"
#---------------------------------------------------------------
# Press 0 in Key Matrix to choose postscript 1 : draw 'DCE'
#---------------------------------------------------------------
ps2:     .asciiz
        "90,0,5000,180,0,8000,180,1,5000,90,1,10000,360,1,5000,270,0,5000,180,1,15000,90,0,15000,360,0,3000,
360,1,8000,135,1,4000,45,1,4000,180,1,8000,90,0,2000,90,1,4000,360,1,8000,270,0,4000,180,1,8000"
#---------------------------------------------------------------
# Press 4 in Key Matrix to choose postscript 2 : draw 'MU'
#---------------------------------------------------------------
ps3:     .asciiz
        "90,0,10000,180,0,8000,90,1,10000,270,0,5000,360,1,1000,180,0,2000,270,1,4000,180,1,1000,90,1,8000,3
60,1,1000,270,1,4000,180,0,2000,270,1,5000,180,1,3000,360,0,3000,90,0,5000,90,1,5000,180,1,4000,270,1,1000,9
0,0,1000,360,0,4000,270,0,5000,180,0,1000,90,1,4000,180,1,1000,270,1,8000,360,1,1000,90,1,4000,90,0,10000"
#---------------------------------------------------------------
# Press 8 in Key Matrix to choose postscript 3 : draw 'takai'
#---------------------------------------------------------------


.kdata
initptr: .word    0          # Pointer to initialize the postscript data array
firstaddress:    .word              # The first address of the array
```

```
.text
main:

setup:
        jal     initialize

        la      $a0, ps1
        jal     store_data
        la      $t0, ptr1
        sw      $v0, 0($t0)

        la      $a0, ps2
        jal     store_data
        la      $t0, ptr2
        sw      $v0, 0($t0)

        la      $a0, ps3
        jal     store_data
        la      $t0, ptr3
        sw      $v0, 0($t0)

main_loop:
        jal     press
        nop
        j       main_loop

end_main:

initialize:
        la      $t0, initptr
        la      $t1, firstaddress
        sw      $t1, 0($t0)
        jr      $ra
```

```
store_data:
        la      $t0, initptr
        lw      $v0, 0($t0)
        add     $t0, $0, $v0
        add     $t1, $0, $a0
script_read_loop:
        add     $t2, $0, $0
number_read_loop:
        lbu     $t3, 0($t1)
        beq     $t3, ',', number_read_end
        beq     $t3, '\0', script_read_end
        subi    $t3, $t3, '0'
        mul     $t2, $t2, 10
        add     $t2, $t2, $t3
        addi    $t1, $t1, 1
        j       number_read_loop
number_read_end:
        sw      $t2, 0($t0)
        addi    $t0, $t0, 4
        addi    $t1, $t1, 1
        j       script_read_loop
script_read_end:
        sw      $t2, 0($t0)
        sw      $0, 4($t0)
        sw      $0, 8($t0)
        sw      $0, 12($t0)
        addi    $t0, $t0, 16
        la      $t1, initptr
        sw      $t0, 0($t1)
        jr      $ra

press:
```

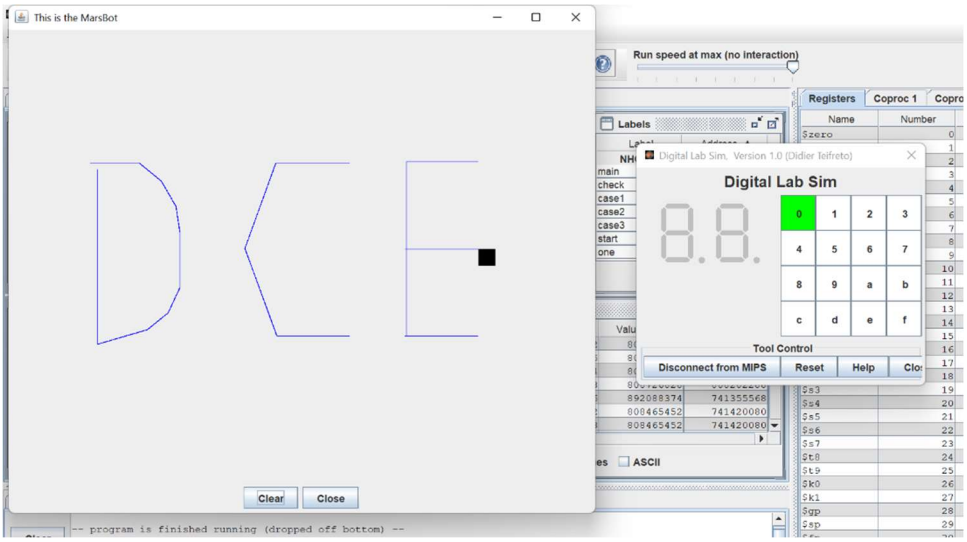
```
        li      $t0, IN_ADDRESS_HEXA_KEYBOARD
        li      $t1, OUT_ADDRESS_HEXA_KEYBOARD
reset_button:
        lb      $t3, 0($t1)
        bne     $t3, 0, finish
case_0:
        addi    $t2, $0, 0x01
        sb      $t2, 0($t0)
        lb      $t3, 0($t1)
        bne     $t3, 0x11, case_4
        la      $t0, ptr1
        lw      $a3, 0($t0)
        j       start
case_4:
        addi    $t2, $0, 0x02
        sb      $t2, 0($t0)
        lb      $t3, 0($t1)
        bne     $t3, 0x12, case_8
        la      $t0, ptr2
        lw      $a3, 0($t0)
        j       start
case_8:
        addi    $t2, $0, 0x04
        sb      $t2, 0($t0)
        lb      $t3, 0($t1)
        bne     $t3, 0x14, finish
        la      $t0, ptr3
        lw      $a3, 0($t0)
        j       start
finish:
        jr      $ra

start:
```

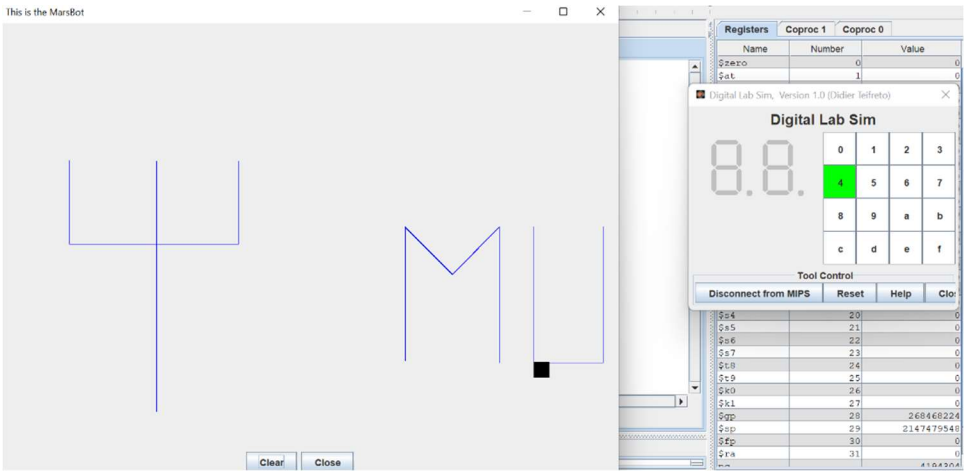
```
        addi    $t3, $0, 1
        addi    $v0, $0, 32
draw_loop:
        lw      $a1, 0($a3)
        lw      $a2, 4($a3)
        lw      $a0, 8($a3)
        li      $t1, HEADING
        li      $t2, LEAVETRACK
        sw      $a1, 0($t1)
        sw      $a2, 0($t2)
        or      $t0, $a1, $a2
        or      $t0, $t0, $a0
        beq     $t0, 0, draw_end
        li      $t0, MOVING
        sw      $t3, 0($t0)
        syscall
        sw      $0, 0($t0)
        sw      $0, 0($t2)
        addi    $a3, $a3, 12
        j       draw_loop
draw_end:
        j       finish
```

4. Kết quả:

- Postscript 1:



- Postscript 2:



– Postscript 3: