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

Final Project
Computer Architecture

GVHD: ThS. Lê Bá Vui

Nhóm: 13

Thành Viên: Đặng Lê Duy - 20194538

Đỗ Bùi Quang Anh - 20194415

Mã Lớp: 130939

Mục lục

Assignment 7:

1. Đề bài

Trình biên dịch của bộ xử lý MIPS sẽ tiến hành kiểm tra cú pháp các lệnh hợp ngữ trong mã nguồn, xem có phù hợp về cú pháp hay không, rồi mới tiến hành dịch các lệnh ra mã máy. Hãy viết một chương trình kiểm tra cú pháp của 1 lệnh hợp ngữ MIPS bất kì (không làm với giả lệnh) như sau:

- Nhập vào từ bàn phím một dòng lệnh hợp ngữ. Ví dụ beq s1,31,t4
- Kiểm tra xem mã opcode có đúng hay không? Trong ví dụ trên, opcode là beq là hợp lệ thì hiện thị thông báo "opcode: beq, hợp lệ"
- Kiểm tra xem tên các toán hạng phía sau có hợp lệ hay không? Trong ví dụ trên, toán hạng s1 là hợp lệ, 31 là không hợp lệ, t4 thì khỏi phải kiểm tra nữa vì toán hạng trước đã bị sai rồi.

Gợi ý: nên xây dựng một cấu trúc chứa khuôn dạng của từng lệnh với tên lệnh, kiểu của toán hạng 1, toán hạng 2, toán hạng 3, số chu kì thực hiện.

2. Phân tích cách thực hiện

- Yêu cầu người dùng nhập vào một câu lệnh, lưu vào một chuỗi data.

- Coi một câu lệnh hợp ngữ gồm 4 phần: mã opcode, toán hạng 1, toán hạng 2, toán hạng 3.
- Tách từng thành phần của câu lệnh và so sánh với khuôn dạng lệnh đã xây dựng sẵn xem nó là kiểu gì và đưa vào các hàm check kiểu hợp lệ tương ứng.
- Chuỗi cấu trúc câu lệnh chuẩn:

  + library: .asciiz "lw****140-lb****140-sw****140-sb****140-addi**112add***111-addiu*112-and***111-andi**112-beq***113-bne***113-div***110divu**110-j*****300-jal***300-lui***120-mfhi**100-mflo**100-mul***111nop***000-nor***111-or****111-ori***112-sll***111-slt***111-slti**112sub***111-subu**111-xor***111-xori**112-"

  + Mỗi câu lệnh có độ dài là 10, ngăn cách nhau bởi ký tự "-", bao gồm mã opcode và khuôn dạng toán hạng tương ứng với mã opcode đó.

  + Quy ước các toán hạng: 1 – register, 2 – constant, 3 – label, 4 – imm(rs), 0 – null.

- Đầu chương trình, khởi tạo một vòng lặp thực hiện việc đọc từng thành phần một của chuỗi data và đưa đi so sánh với kiểu dữ liệu quy ước. Nếu phù hợp thì tiếp tục so sánh các thành phần phía sau. Nếu không phù hợp thì báo cho người dùng thành phần đó không hợp lệ. • Để lấy từng thành phần của câu lệnh nhập vào, đọc từng kí tự một của chuỗi data nhập vào và lưu vào một biến temp, cho đến khi gặp các kí tự là '\t'(tab), ' '(space), ','(dấu phẩy), '\0' thì dừng lại và trở về chương trình chính để thực hiện việc check thành phần vừa lấy được có hợp lệ hay không.
- Để kiểm tra chuỗi opcode sau khi lấy được thì sẽ so sánh với chuỗi này với chuỗi cấu trúc câu lệnh chuẩn bằng cách so sánh từng ký tự, nếu ký tự khác nhau thì so sánh với opcode tiếp theo trong chuỗi chuẩn cho đến khi kết thúc.
- Nếu không có opcode phù hợp -> in ra màn hình run I/O opcode không hợp lệ và kết thúc chương trình. Nếu có opcode hợp lệ -> Nếu opcode phù hợp thì sẽ lấy được khuôn dạng

toán hạng tương ứng với opcode đó và tiến hành kiểm tra lần lượt các toán hạng tiếp theo. Nếu có 1 toán hạng không hợp lệ -> báo ra màn hình run I/O và kết thúc chương trình.

- Để kiểm tra toán hạng là thanh ghi, tương tự opcode chuẩn thì cũng có một chuỗi register chuẩn được tạo ở đầu chương trình để so sánh
- Để kiểm tra toán hạng là hằng số: hằng số hợp lệ có thể có kí tự đầu là dấu trừ '- ' hoặc dấu cộng '+', các ký tự phía sau bắt buộc phải từ 0 đến 9 (bài này em chỉ áp dụng với hệ 10)
- Để kiểm tra toán hạng là label: label hợp lệ có thể có kí tự đầu là dấu gach dưới '_' hoặc chữ thường, chữ hoa, nếu có các ký tự tiếp theo, thì các ký tự này bắt buộc phải là chữ cái, chữ số hoặc dấu gạch dưới.
- Để kiểm tra toán hạng là null: kiểm tra xem chuỗi vừa lấy được có kí tự nào hay không, nếu có -> không hợp lệ, nếu không có kí tự nào -> hợp lệ
- Để kiểm tra toán hạng là một cụm imm(rs): kiểm tra xem dấu mở ngoặc '(' và đóng ngoặc ')' có hợp lệ không?

  + Có đủ 2 dấu đóng và mở ngoặc hay không.

  + Có đúng thứ tự '(' rồi mới đến ')' hay không

  + Sau dấu ')' còn ký tự thừa nào hay không

  + Nếu có hợp lệ -> xử lý chuỗi bằng cách thay ký tự '(' bằng kí tự ' ' và ký tự ')' bằng kí tự '\0', sau đó đưa vào hàm tách chuỗi để tách thành 2 thành phần là hằng số và thanh ghi -> đưa vào các hàm kiểm tra từng thành phần tương ứng bên trên. Nếu có bất kì một thành phần nào sai -> Không hợp lệ. Ngược lại -> hợp lệ.

- Sau khi kiểm tra xong 3 toán hạng, chương trình sẽ kiểm tra xem còn kí tự thừa nào khác hay không. Nếu còn -> câu lệnh không hợp lệ và báo kết quả của câu lệnh vừa nhập
- Sau khi kiểm tra xong 1 câu lệnh -> Hỏi người dùng có muốn kiểm tra tiếp một câu lệnh khác hay không?
- Nếu người dùng chọn không -> Kết thúc chương trình.

3. Kết quả:

- Nhập đúng cú pháp:

```
>>>>>>>>>>MENU<<<<<<<<<<
1. Kiem tra code
2. Thoat
Lua chon: 1
Nhap vao mot dong lenh hop ngu: add $s1,$s2,$s2
opcode: add hop le
toan hang: $s1 hop le
toan hang: $s2 hop le
toan hang: $s2 hop le
cau lenh:  hop le

Tiep tuc chuong trinh (0.no||1.yes):
```

- Nhập sai cú pháp:

```
Tiep tuc chuong trinh (0.no||1.yes): 1
Nhap vao mot dong lenh hop ngu: add $s1,$s2,4
opcode: add hop le
toan hang: $s1 hop le
toan hang: $s2 hop le
toan hang: 4 khong hop le

Tiep tuc chuong trinh (0.no||1.yes):
```

```
Tiep tuc chuong trinh (0.no||1.yes): 1
Nhap vao mot dong lenh hop ngu: j 1
opcode: j hop le
toan hang: 1 khong hop le

Tiep tuc chuong trinh (0.no||1.yes):
```

```
Tiep tuc chuong trinh (0.no||1.yes): 1
Nhap vao mot dong lenh hop ngu: lw $t1,12($sp
opcode: lw hop le
toan hang: $t1 hop le
toan hang: 12 $sp khong hop le

Tiep tuc chuong trinh (0.no||1.yes):
```

- Nhập thừa toán hạng:

```
Tiep tuc chuong trinh (0.no||1.yes): 1
Nhap vao mot dong lenh hop ngu: add $s1,$s2,$s3,$s3
opcode: add hop le
toan hang: $s1 hop le
toan hang: $s2 hop le
toan hang: $s3 hop le
cau lenh:  khong hop le

Tiep tuc chuong trinh (0.no||1.yes):
```

4. Mã nguồn.

.data

#cau truc cau lenh: 'addi $t1 $t2 0'

library: .asciiz "lw****140-lb****140-sw****140-sb****140-addi**112-add***111-addiu*112-and***111-andi**112-beq***113-bne***113-div***110-divu**110-j*****300-jal***300-lui***120-mfhi**100-mflo**100-mul***111-nop***000-nor***111-or****111-ori***112-sll***111-slt***111-slti**112-sub***111-subu**111-xor***111-xori**112-"

register: .asciiz "$zero-$at-$v0-$v1-$a0-$a1-$a2-$a3-$t0-$t1-$t2-$t3-$t4-$t5-$t6-$t7-$t8-$t9-$s1-$s2-$s3-$s4-$s5-$s6-$s7-$k0-$k1-$gp-$sp-$fp-$ra-$0-$1-$2-$3-$4-$5-$6-$7-$8-$9-$10-$11-$12-$13-$14-$15-$16-$17-$18-$19-$20-$21-$22-$23-$24-$25-$26-$27-$28-$29-$30-$31-"

#quy uoc cac toan hang: 1 registers, 2 constant, 3 label, 4 imm(rs)

```
	menu_msg:	.asciiz "\n>>>>>>>>>>MENU<<<<<<<<<<\n1. Kiem tra code\n2. Thoat \nLua chon: "
	menu_error_msg: .asciiz "\nKhong co trong menu. Hay chon lai!\n"
	msg1: "Nhap vao mot dong lenh hop ngu: "
	msg2: "opcode: "
	msg3: "toan hang: "
	msg4: "cau lenh: "
	msg5: "\nTiep tuc chuong trinh (0.no||1.yes): "
	msg_valid:" hop le\n"
	msg_invalid:" khong hop le\n"
	data: .space 200		# luu input
	temp: .space 50		# luu cac thanh phan cau lenh sau khi cat
	temp2: .space 10		# luu khuon dang cau lenh
	temp3: .space 50		# luu thanh phan sau khi cat duoc o offset(base)

.text
m_menu_start:
	li $v0, 4
	la $a0, menu_msg #display Menu
	syscall

	# Read number input menu
	li $v0, 5
	syscall

	beq $v0, 2, end_main		# 2: exit
	beq $v0, 1, m_menu_end		# 1: jump to examination

	li $v0, 4
```

```
        la $a0, menu_error_msg                  # Wrong input
        syscall

        j m_menu_start

m_menu_end:
# >>>>>>>>>> READ INPUT <<<<<<<<<<
m_input:
        jal  readData
        nop

# >>>>>>>>>> CHECK <<<<<<<<<<

        j m_menu_start #return to menu after checking

end_main:
        li $v0, 10 #exit
        syscall

readData:       # doc lenh nhap vao tu ban phim
        li      $v0, 4          # in msg ra run i/o
        la      $a0, msg1
        syscall
        li      $v0, 8
        la      $a0, data       # luu chuoi nhap vao -> data
        li      $a1, 200        # so ky tu toi da doc vao
        syscall

main:
```

```
        la      $s0, data               # dia chi data
        la      $s1, temp               # dia chi chuoi sau khi cat
        add     $s2, $zero, $zero       # i = 0 data
        add     $s3, $zero, $zero       # dem thanh phan: 1 - opcode, 2, 3, 4 - toan hang, 5 check ki tu thua
        la      $s4, temp2              # dia chi temp2

# lap 5 lan de lay cac thanh phan cau lenh
# 1 - lay opcode
# 2, 3, 4 - lay toan hang
# 5 - check ki tu thua
getComponent:
        addi    $s3, $s3, 1
        beq     $s3, 6, exit            # dem >= 6 -> exit
        add     $a0, $s0, $zero  # truyen bien vao cutComponent
        add     $a1, $s1, $zero
        add     $a2, $s2, $zero
        jal     cutComponent
        add     $s2, $v0, $zero
        beq     $s3, 5, endGetComponent # dem = 5 -> check ket thuc
        beq     $s3, 1, opcode          # dem = 1 -> check ket thuc
        j       checkToanHang                   # dem = 2, 3, 4 -> check toan hang

opcode:
        add     $a0, $s1, $zero  # truyen vao temp
        la      $a1, library            # opcode chuan
        jal     checkOpcode
        add     $s5, $v0, $zero  # check
        li      $v0, 4
```

```
        la      $a0, msg2
        syscall
        li      $v0, 4
        la      $a0, temp
        syscall
        j       check

checkToanHang:
        addi    $t0, $s3, -2            # k - stt cua toan hang
        add     $t0, $s4, $t0           # t0 = dia chi temp2[k]
        lb      $t1, 0($t0)             # t1 = temp2[k]
        beq     $t1, 48, null           # null - 0
        beq     $t1, 49, reg            # register - 1
        beq     $t1, 50, const          # constant - 2
        beq     $t1, 51, label          # label - 3
        beq     $t1, 52, immrs          # imm(rs) - 4

reg:
        add     $a0, $s1, $zero     # i
        jal     checkReg
        add     $s5, $v0, $zero
        j       print

const:
        add     $a0, $s1, $zero
        jal     checkConstant
        add     $s5, $v0, $zero
        j       print
```

```
label:
        add     $a0, $s1, $zero
        jal     checkLabel
        add     $s5, $v0, $zero
        j       print

immrs:
        add     $a0, $s1, $zero
        jal     checkImmRs
        add     $s5, $v0, $zero
        j       print

print:
        li      $v0, 4
        la      $a0, msg3
        syscall
        li      $v0, 4
        la      $a0, temp
        syscall
        j       check

null:
endGetComponent:
        add     $s3, $zero, 5
        li      $v0, 4
        la      $a0, msg4
        syscall
        add     $t0, $s1, $zero         # check xem con ki tu thua hay ko
        lb      $t2, 0($t0)
```

```
        bne     $t2, $zero, invalid
        j       valid

check:
        beq     $s5, $zero, invalid

valid:
        li      $v0, 4
        la      $a0, msg_valid
        syscall
        j       getComponent

invalid:
        li      $v0, 4
        la      $a0, msg_invalid
        syscall

exit:
repeatMain:
        li      $v0, 4
        la      $a0, msg5
        syscall
        li      $v0, 8
        la      $a0, data
        li      $a1, 200
        syscall

checkRepeat:
        add     $t0, $a0, $zero  # ki tu dau tien
```

```
        lb      $t0, 0($t0)
        beq     $t0, 48, out            # = 0
        beq     $t0, 49, readData       # = 1 -> lap lai
        j       repeatMain

out:
        li $v0, 10 #exit
        syscall

#---------------------------------------------------------------
# cutComponent: cat cac thanh phan cau lenh (bo qua cac ky tu space, dau phay, \t)
# a0 -> dia chi chuoi data
# a1 -> dia chi temp - chuoi chua ket qua sau cut
# a2 -> vi tri bat dau cat
# v0 -> vi tri ket thuc cat
#---------------------------------------------------------------
cutComponent:
        addi    $sp, $sp, -8
        sw      $ra, 0($sp)
        sw      $s0, 4($sp)                     # j dem temp

init1:
        add     $s0, $zero, $zero               # j = 0

_X:
        add     $t0, $a0, $a2                   # dia chi data[i]
        lb      $t1, 0($t0)                     # t1 = data[i]
        beq     $t1, 9, update1         # \t
        beq     $t1, 32, update1                # ' '
```

```
        beq     $t1, 44, update1        # ','
        beq     $t1, 10, endF1          # \n
        j       loadChar

update1:
        addi    $a2, $a2, 1
        j       _X

loadChar:
        beq     $t1, 9, endF1
        beq     $t1, 32, endF1
        beq     $t1, 44, endF1
        beq     $t1, 10, endF1
        beq     $t1, $zero, endF1       # ki tu ket thuc
        add     $t0, $a1, $s0           # dia chi temp
        sb      $t1, 0($t0)             # temp[j]
        addi    $s0, $s0, 1             # j++
        addi    $a2, $a2, 1             # i++
        add     $t0, $a0, $a2           # dia chi data[i]
        lb      $t1, 0($t0)             # data[i]
        j       loadChar

endF1:
        add     $t0, $a1, $s0
        sb      $zero, 0($t0)
        add     $v0, $a2, $zero
        lw      $ra, 0($sp)
        lw      $s0, 4($sp)
        addi    $sp, $sp, 8
```

```
        jr      $ra

#-----------------------------------------------------------------
# checkOpcode: check opcode co hop le ko -> lay khuon dang toan hang
# a0 -> dia chi opcode cat duoc
# a1 -> dia chi chuoi opcode chuan
# v0 -> 0|1 -> ko hop le | hop le
#-----------------------------------------------------------------

checkOpcode:
        addi    $sp, $sp, -16
        sw      $ra, 0($sp)
        sw      $s0, 4($sp)             # j temp
        sw      $s1, 8($sp)             # i opcode
        sw      $s2, 12($sp)            # check = 0

initcheckOpcode:
        add     $s0, $zero, $zero       # j = 0
        add     $s1, $zero, $zero       # i = 0
        add     $s2, $zero, $zero       # flag danh dau opcode hop le hay khong

loopCheckOpcode:
        add     $t0, $a1, $s1           # dia chi chuoi opcode chuan
        lb      $t1, 0($t0)             # opcode[i]
        beq     $t1, $zero, endFCheckOpcode # ket thuc opcode nhung ko co opcode phu hop
        add     $t0, $a0, $s0           # dia chi chuoi opcode
        lb      $t2, 0($t0)             # temp[j]
        addi    $s0, $s0, 1             # j = j+1
        addi    $s1, $s1, 1             # i = i+1
```

```
        beq     $t1, $t2, loopCheckOpcode       # temp[j] = opcode[i] -> loop
        bne     $t1, 42, updateNext             # neu opcode[i] != '*' -> sai tu nay -> check tu tiep theo
        bne     $t2, $zero, updateNext          # neu temp[j] != '\0' -> check tu tiep theo
        j       assign

updateNext:
        add     $s0, $zero, $zero               # xet tu temp[0]
        addi    $t0, $zero, 10                  # so ki tu trong 1 opcode
        div     $s1, $t0                        # i/10
        mflo    $t1                             # lay phan thuong
        addi    $t1, $t1, 1
        mul     $s1, $t1, $t0
        j       loopCheckOpcode

assign: # opcode chuan
        addi    $s2, $zero, 1                   # opcode hop le -> flag = 1
        la      $a0, temp2                      # gan khuon dang
        add     $s0, $zero, $zero               # dem temp2[j]

_X42:   # bo qua *
        addi    $s1, $s1, 1                     # i = i+1
        add     $t0, $a1, $s1
        lb      $t1, 0($t0)                     # opcode[i]
        beq     $t1, 42, _X42                   # opcode[i] = '*' -> bo qua

loopAssign: # gan khuon dang
        add     $t0, $a0, $s0                   # dia chi temp2[i]
        sb      $t1, 0($t0)
        addi    $s1, $s1, 1                     # i = i+1
```

```
        addi    $s0, $s0, 1                     # j = j+1
        add     $t0, $a1, $s1
        lb      $t1, 0($t0)                     # opcode[i]
        bne     $t1, 45, loopAssign             # opcode[i] != '-' -> tiep tuc gan
        add     $t0, $a0, $s0
        sb      $zero, 0($t0)                   # gan '\0' cho temp2[i]

endFCheckOpcode:
        add     $v0, $s2, $zero
        lw      $ra, 0($sp)
        lw      $s0, 4($sp)                     # j temp
        lw      $s1, 8($sp)                     # i opcode
        lw      $s2, 12($sp)                    # check = 0
        addi    $sp, $sp, 16
        jr      $ra

#---------------------------------------------------------------
# checkReg: ktra register co hop le hay ko?
# a0 -> dia chi temp - chuoi chua gia tri can check
# v0 -> 0|1: ko hop le | hop le
#---------------------------------------------------------------

checkReg:
        addi    $sp, $sp, -20
        sw      $ra, 0($sp)
        sw      $s0, 4($sp)             # dia chiopcode chuan
        sw      $s1, 8($sp)             # i
        sw      $s2, 12($sp)            # j temp[j]
        sw      $s3, 16($sp)            #check
```

```
initCheckReg:
        la      $s0, register
        add     $s1, $zero, $zero       # i
        add     $s2, $zero, $zero       # j
        add     $s3, $zero, $zero       # flag = 0

loopCheckReg:
        add     $t0, $s0, $s1           # dia chi register[i]
        lb      $t1, 0($t0)             # register[i]
        beq     $t1, $zero, endCheckReg # ket thuc chuoi register
        add     $t0, $a0, $s2           # dia chi temp[j]
        lb      $t2, 0($t0)             # temp[j]
        addi    $s1, $s1, 1
        addi    $s2, $s2, 1
        beq     $t1, $t2, loopCheckReg  # bang nhau -> loop
        bne     $t1, 45, updateNextReg # reg[i] = '-'
        beq     $t2, $zero, true # temp[j] = '\0'

updateNextReg:
_X45:
        add     $t0, $s0, $s1           # dia chi reg[i]
        lb      $t1, 0($t0)             # reg[i]
        beq     $t1, 45, false
        add     $s1, $s1, 1
        j       _X45

false:
        addi    $s1, $s1, 1
```

```
        add     $s2, $zero, $zero
        j       loopCheckReg

true:
        addi    $s3, $zero, 1           # flag = 1

endCheckReg:
        add     $v0, $s3, $zero
        lw      $ra, 0($sp)
        lw      $s0, 4($sp)
        lw      $s1, 8($sp)
        lw      $s2, 12($sp)
        lw      $s3, 16($sp)
        addi    $sp, $sp, 20
        jr      $ra

#---------------------------------------------------------------
# checkConstant: ktra 1 hang so co hop le hay ko
# a0 -> temp: dia chi chuoi chua gia tri can check
# v0 -> 0|1 -> ko hop le | hop le
#---------------------------------------------------------------

checkConstant:
        addi    $sp, $sp, -12
        sw      $ra, 0($sp)
        sw      $s0, 4($sp)
        sw      $s1, 8($sp)

initCheckConstant:
```

```
        add     $s0, $zero, $zero       # i
        addi    $s1, $zero, 1           # flag = 1

firstChar: # isNumber
        add     $t0, $a0, $s0           # dia chi tenp[i]
        lb      $t1, 0($t0)             # temp[i]
        beq     $t1, 43, dkien          # +
        beq     $t1, 45, dkien          # -

j       isNumber

dkien: # -
        add     $s0, $s0, 1             # i++
        add     $t0, $a0, $s0           # dia chi temp[i]
        lb      $t1, 0($t0)     # temp[i]
        bne     $t1, $zero, isNumber
        j       falseCheckConstant

loopCheckConstant:
        add     $s0, $s0, 1                     # i++
        add     $t0, $a0, $s0                   # dia chi temp[i]
        lb      $t1, 0($t0)                     # temp[i]
        bne     $t1, $zero, isNumber
        beq     $s0, $zero, falseCheckConstant  # co ky tu hay ko
        j       endCheckConstant

isNumber:
        slti    $t0, $t1, 48
        bne     $t0, $zero, falseCheckConstant  # temp[i] < 48 -> sai
```

```
        slti    $t0, $t1, 58
        beq     $t0, $zero, falseCheckConstant  # temp[i] >= 58 -> sai
        j       loopCheckConstant

falseCheckConstant:
        add     $s1, $zero, $zero               # flag = 0

endCheckConstant:
        add     $v0, $s1, $zero
        lw      $ra, 0($sp)
        lw      $s0, 4($sp)             # i
        lw      $s1, 8($sp)             # check
        addi    $sp, $sp, 12
        jr      $ra

#---------------------------------------------------------------
# checkLabel: ktra 1 label co hop le hay ko
# a0 -> temp -> dia chi chuoi chua gia tri can ktra
# v0 -> 0|1 -> ko hop le | hop le
#---------------------------------------------------------------

checkLabel:
        addi    $sp, $sp, -12
        sw      $ra, 0($sp)
        sw      $s0, 4($sp)
        sw      $s1, 8($sp)

initCheckLabel:
        add     $s0, $zero, $zero       # i
```

```
        addi    $s1, $zero, 1           # flag = 1

firstCharLabel:
        add     $t0, $a0, $s0
        lb      $t1, 0($t0)             # temp[i]
        j       isUpcase

loopCheckLabel:
        add     $s0, $s0, 1
        add     $t0, $a0, $s0
        lb      $t1, 0($t0)             # temp[i]
        beq     $t1, $zero, endCheckLabel

isNumberL: # 48 -> 57
        slti    $t0, $t1, 48
        bne     $t0, $zero, falseCheckLabel
        slti    $t0, $t1, 58
        beq     $t0, $zero, isUpcase
        j       loopCheckLabel

isUpcase: # 65 -> 90
        slti    $t0, $t1, 65
        bne     $t0, $zero, falseCheckLabel     # temp[i] < 65 -> sai
        slti    $t0, $t1, 91
        beq     $t0, $zero, _                   # temp[i] >= 91 -> check xem co = '_'
        j       loopCheckLabel

_:
        bne     $t1, 95, isLowcase
```

```
        j       loopCheckLabel

isLowcase: # 97  -> 122
        slti    $t0, $t1, 97
        bne     $t0, $zero, falseCheckLabel     # temp[i] < 97 -> sai
        slti    $t0, $t1, 123
        beq     $t0, $zero, falseCheckLabel     # temp[i] >= 123 -> sai
        j       loopCheckLabel

falseCheckLabel:
        add     $s1, $zero, $zero               # flag = 0

endCheckLabel:
        add     $v0, $s1, $zero
        lw      $ra, 0($sp)
        lw      $s0, 4($sp) # i
        lw      $s1, 8($sp) # check
        addi    $sp, $sp, 12
        jr      $ra

#---------------------------------------------------------------
# checkImmRs: kiem tra 1 cum offset(base) xem co hop le hay ko? vd: 12($t0)
# a0 -> dia chi temp - chuoi chua gia tri can ktra
# v0 -> 0|1 -> ko hop le | hop le
#---------------------------------------------------------------

checkImmRs:
        addi    $sp, $sp, -28
        sw      $ra, 0($sp)
```

```
        sw      $s0, 4($sp)                 # temp
        sw      $s1, 8($sp)                 # i
        sw      $s2, 12($sp)                # temp3
        sw      $s3, 16($sp)                # check
        sw      $s4, 20($sp)                # space 32

initCheckImmRs:
        add     $s0, $a0, $zero         # luu dia chi temp -> s0
        add     $s1, $zero, $zero           # i
        la      $s2, temp3                  # cum offset(base) sau khi cat
        add     $s3, $zero, $zero           # flag = 0
        addi    $s4, $zero, 32              # s4 = ' '

test:
        add     $t0, $s0, $s1               # dia chi temp[i]
        lb      $t1, 0($t0)                 # temp[i]
        addi    $s1, $s1, 1                 # i++
        beq     $t1, $zero, endTest
        beq     $t1, 40, open               # '('
        beq     $t1, 41, close              # ')'
        j       test

open:
        bne     $s3, $zero, falseCheckImmRs # flag != 0 -> sai
        sb      $s4, 0($t0)                 # luu ' ' thay '('
        addi    $s3, $s3, 1                 # flag = 1
        j       test

close:
```

```
        bne     $s3, 1, falseCheckImmRs                 # flag != 1 -> sai
        sb      $zero, 0($t0)                   # luu '\0' thay ')'
        addi    $s3, $s3, 1                     # s3 = 2
        add     $t0, $s0, $s1                   # dia chi temp[i]
        lb      $t1, 0($t0)                     # temp[i]
        bne     $t1, $zero, falseCheckImmRs     # temp[i] != '\0' -> sai

endTest:
        bne     $s3, 2, falseCheckImmRs                 # flag != 2 -> sai

init2:
        add     $s3, $zero, $zero               # flag = 0
        add     $s1, $zero, $zero               # i = 0

mainCheck:
        add     $a0, $s0, $zero         # truyen bien vao cutComponent
        add     $a1, $s2, $zero
        add     $a2, $s1, $zero
        jal     cutComponent
        add     $s1, $v0, $zero         # i ket thuc cat
        add     $a0, $s2, $zero         # truyen temp3
        jal     checkConstant
        add     $s3, $v0, $zero         # lay ket qua 0|1
        beq     $s3, $zero, falseCheckImmRs
        add     $a0, $s0, $zero         # truyen bien vao cutComponent
        add     $a1, $s2, $zero
        add     $a2, $s1, $zero
        jal     cutComponent
        add     $s1, $v0, $zero
```

```
        add     $a0, $s2, $zero
        jal     checkReg
        add     $s3, $v0, $zero
        bne     $s3, $zero, trueCheckImmRs

falseCheckImmRs:
        add     $s3, $zero, $zero
        j       endCheckImmRs

trueCheckImmRs:
        addi    $s3, $zero, 1

endCheckImmRs:
        add     $v0, $s3, $zero
        lw      $ra, 0($sp)
        lw      $s0, 4($sp)
        lw      $s1, 8($sp)
        lw      $s2, 12($sp)
        lw      $s3, 16($sp)
        lw      $s4, 20($sp) # space 32
        addi    $sp, $sp, 24
        jr      $ra
```

Assignment 4:

1. Đề bài: Postscript CNC Marsbot

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>,<Cắt/Không cắt>,<Thời gian>
- Trong đó <Góc chuyển động> là góc của hàm HEADING của Marsbot
- <Cắt/Không cắt> thiết lập lưu vết/không lưu vết
- <Thời gian> là thời gian duy trì quá trình vận hành hiện tại

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)

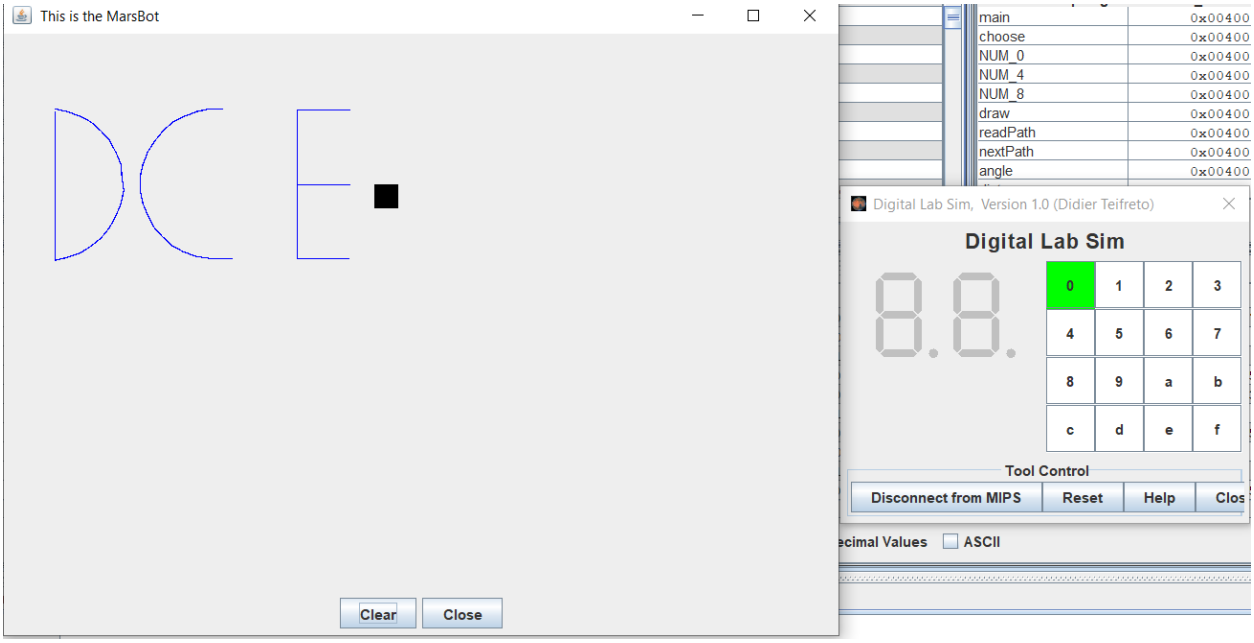
2. Cách thực hiện:

- Lấy các địa chỉ controller HEADING, MOVING, LEAVETRACK để có thể điều khiển được Marsbot
- Lấy địa chỉ IN_KEYBOARD và OUT_KEYBOARD để nhận dữ liệu số input từ Key Matrix
- Sau đó khởi tạo 3 mảng word kết thúc lần lượt bởi zero1, zero2, zero3 dùng cho postscript để chạy MarsBot
- Sau đó vào main, đi lần lượt qua choose (lấy dữ liệu từ Key Matrix và xử lí lựa chọn), draw (đọc postscript và vẽ hình) và cuối cùng là quit. Chạy qua lần lượt các chức năng trên rồi quay lại qua jr $ra để chạy chức năng tiếp theo. Khởi tạo t6 để loop qua script, và t7 làm kho chứa nội dung script[t6]
- Bắt đầu choose với lấy dữ liệu từ Key Matrix như đã học, nhưng chỉ xét xem nó có lần lượt là NUM_0, nếu không thì check NUM_4, nếu không thì check NUM_8, và nếu không thì loop lại check từ đầu.
- Check được đúng trường hợp nào, thì load vào 2 arguments a1 và a2 địa chỉ của script và kết thúc của script tương ứng. Nhảy sang draw để bắt đầu đọc postscript và cắt
- Khởi tạo draw bằng bắt đầu đổi MOVING sang 1 để MarsBot bắt đầu chạy, sau đó cho t0 và t1 bằng 0, 2 biến tạm thời này sẽ được dùng cho việc chạy qua loop và kiểm tra điều kiện
- Lần lượt lấy các số từ script bằng cách nhân t6 với 4 và cộng vào địa chỉ a1, sau đó lưu các số lần lượt vào angle (góc), distance (độ dài đi – thời gian đi), và cut (check xem con đường đến địa chỉ tiếp theo là có in track hay không track). Mỗi lần lưu số liệu vào các mục increment t6.
- Sau khi lấy được angle, đi vào ROTATE để xoay góc của Marsbot

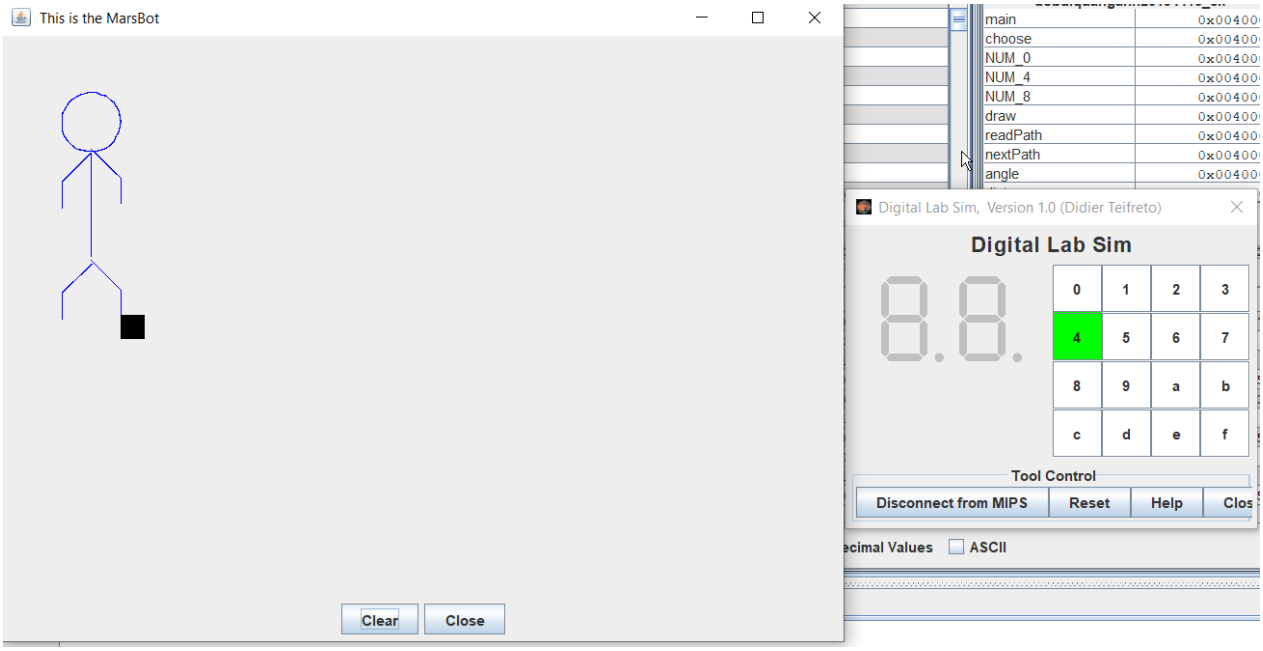
- Sau khi lấy được distance, cho vào t1 để sử dụng sau trong SLEEP (bot trong quá trình chạy đến điểm tiếp theo)
- Lấy được cut, check là cut = 1 hoặc = 0 để chuyển LEAVETRACK là 1 hoặc 0 trong cut hoặc noCut, sau đó đi vào SLEEP
- SLEEP giữ cho bot chạy bằng cách làm chương trình sleep trong khoảng thời gian tỉ lệ thuận với độ dài. Sau khi đã sleep xong, tiến hành kiểm tra xem địa chỉ xét script bây giờ có trùng với địa chỉ kết thúc script không, nếu có, thì đổi MOVING thành 0 trong END, và rồi quit chương trình bằng syscall với v0 bằng 10
- Nếu không thì sẽ quay lại draw và đọc tiếp script để lấy hướng tiếp theo bot sẽ đi.

3. Kết quả:

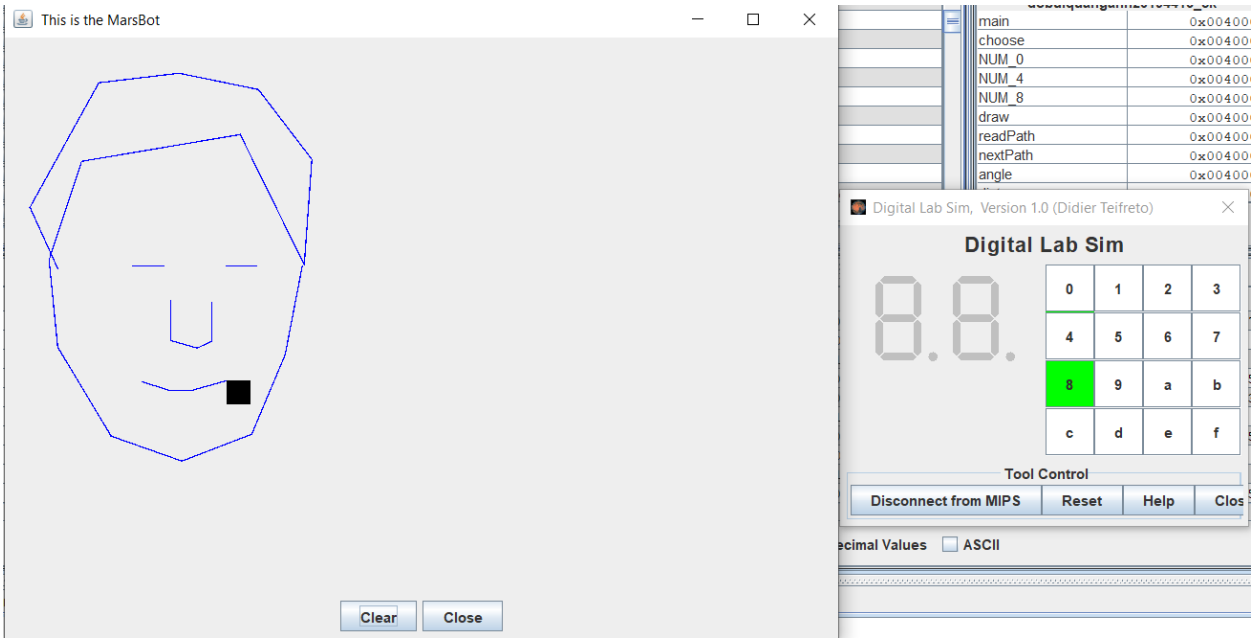
- Nếu người dùng ấn số 0: Hiện ra DCE



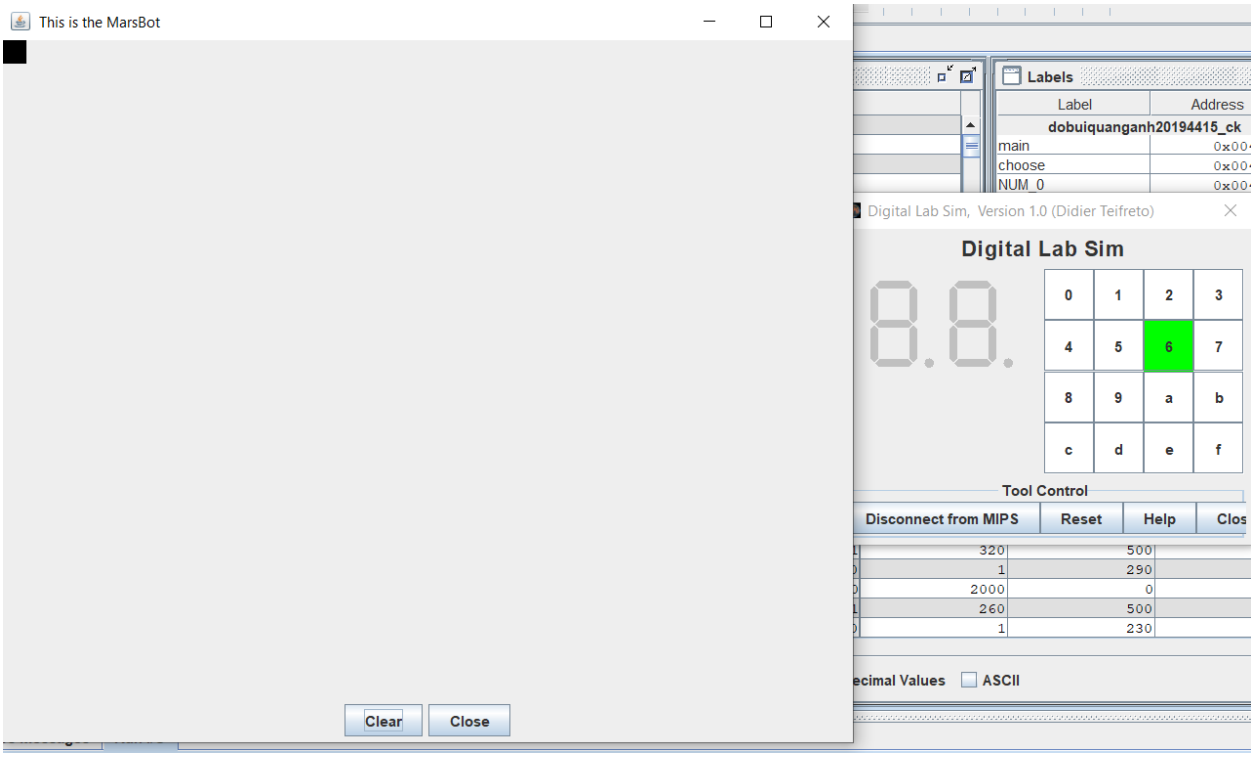
- Nếu người dùng ấn số 4: Hiện ra hình người gậy



- Nếu người dùng ấn số 8: Hiện ra chân dung mặt thầy Lê Bá Vui



- Nếu người dùng ấn số không phải 3 số trên: Không hiện gì và đợi người dùng nhập đúng số



4. Mã assembly:

```
.eqv HEADING 0xffff8010
.eqv MOVING 0xffff8050
.eqv LEAVETRACK 0xffff8020
.eqv IN_KEYBOARD 0xFFFF0012
.eqv OUT_KEYBOARD 0xFFFF0014

.data
# DCE
script1: .word
90,2000,0,180,3000,0,180,5790,1,80,500,1,70,500,1,60,500,1,50,500,1,40,500,1,30,500,1,20,50
0,1,10,500,1,0,500,1,350,500,1,340,500,1,330,500,1,320,500,1,310,500,1,300,500,1,290,500,1,2
80,490,1,90,2000,0,90,4500,0,270,500,1,260,500,1,250,500,1,240,500,1,230,500,1,220,500,1,21
0,500,1,200,500,1,190,500,1,180,500,1,170,500,1,160,500,1,150,500,1,140,500,1,130,500,1,120
,500,1,110,500,1,100,500,1,90,900,1,90,4500,0,270,2000,1,0,5800,1,90,2000,1,180,2900,0,270,
2000,1,90,3000,0
zero1: .word
# Nguoi que
script2: .word
146,3605,0,0,0,1,4,174,1,15,174,1,24,174,1,35,174,1,45,174,1,54,174,1,65,174,1,74,174,1,84,17
4,1,94,174,1,105,174,1,114,174,1,125,174,1,135,174,1,145,174,1,155,174,1,164,174,1,175,174,
1,184,174,1,194,174,1,204,174,1,215,174,1,225,174,1,235,174,1,245,174,1,254,174,1,265,174,1
,274,174,1,284,174,1,294,174,1,305,174,1,315,174,1,324,174,1,335,174,1,345,174,1,355,174,1,
```

```
135,1414,0,180,4000,1,0,4000,0,225,1414,1,180,1000,1,26,2236,0,135,1414,1,180,1000,1,206,2
236,0,225,1414,1,180,1000,1,26,2236,0,135,1414,1,180,1000,1
zero2: .word
# Chan dung thay Le Ba Vui
script3: .word
167,8102,0,336,2284,1,29,4837,1,83,2716,1,102,2765,1,142,3420,1,184,3612,1,334,4965,1,260,
5474,1,199,3511,1,174,3014,1,149,3498,1,110,2563,1,69,2563,1,23,2954,1,11,3059,1,270,6600,
0,90,1200,1,90,2400,0,90,1200,1,248,3231,0,180,1500,1,108,948,1,63,670,1,0,1500,1,221,3612,
0,108,948,1,90,900,1,75,1236,1
zero3: .word

.text
main:
        jal     choose
        nop
        jal     draw
        nop
        jal     quit
        nop
choose:
        li      $t3, IN_KEYBOARD
        li      $t4, OUT_KEYBOARD
        addi    $t6, $zero, 0 # Su dung de loop qua mang
        addi    $t7, $zero, 0
        NUM_0:
                li      $t5, 0x01
                sb      $t5, 0($t3)
                lb      $a0, 0($t4)
                bne     $a0, 0x11, NUM_4
                nop
                la      $a1, script1
                la      $a2, zero1
                jr      $ra
        NUM_4:
                li      $t5, 0x02
                sb      $t5, 0($t3)
                lb      $a0, 0($t4)
                bne     $a0, 0x12, NUM_8
                nop
                la      $a1, script2
                la      $a2, zero2
                jr      $ra
        NUM_8:
                li      $t5, 0X04
```

```
                sb      $t5, 0($t3)
                lb      $a0, 0($t4)
                bne     $a0, 0x14, main
                nop
                la      $a1, script3
                la      $a2, zero3
                jr      $ra

draw:
        li      $at, MOVING
        addi    $k0, $zero,1
        sb      $k0, 0($at)
readPath:
        nextPath:
                addi    $t0, $zero, 0
                addi    $t1, $zero, 0
                angle:          # Extract goc di chuyen
                        sll     $t0, $t6, 2
                        add     $t7, $a1, $t0
                        lw      $t5, 0($t7)
                        add     $a0, $t5, $zero
                        jal     ROTATE
                        nop
                distance:
                        addi    $t6, $t6, 1
                        sll     $t0, $t6, 2
                        add     $t7, $a1, $t0
                        lw      $t5, 0($t7)
                        add     $t1, $zero, $t5
                willCut: # Lay boolean xet co cat hay khong
                        jal     UNTRACK                 # Set the previous point
                        nop
                        addi    $t6, $t6, 1
                        sll     $t0, $t6, 2
                        add     $t7, $a1, $t0
                        lw      $t5, 0($t7)
                        beq     $t5, 1, cut
                        nop
                        j       noCut
                        nop
        cut:
                jal     TRACK           # Cho track tuong duong voi cat
                nop
                j       SLEEP
```

```
        nop
    noCut:
        jal UNTRACK             # Bo track tuong duong voi khong cat
        nop
        j       SLEEP
        nop
    TRACK:
        li      $at, LEAVETRACK #  Dat LEAVETRACK = 1 de ve track
        li      $k0, 1
        sb      $k0, 0($at)
        jr      $ra

    UNTRACK:
        li      $at, LEAVETRACK #  Dat LEAVETRACK = 0 de khong ve track
        sb      $0, 0($at)
        jr      $ra

    ROTATE:
        li      $at, HEADING    # Dat HEADING la goc trong a0
        sw      $a0, 0($at)
        jr      $ra
    SLEEP:
        li      $v0, 32
        move    $a0, $t1        # Ngu de cho Marsbot cat trong khoang thoi gian $t1
        syscall
        addi    $t6, $t6, 1
        sll     $t0, $t6, 2
        add     $t7, $a1, $t0
        bne     $t7, $a2, draw  # Sau khi ket thuc 1 net, quay lai ve tiep, tru khi gap end
    END:
        li      $at, MOVING     # Dat dia chi MOVING thanh 0 va ket thuc
        sb      $0, 0($at)
        add     $at, $zero, 0

quit:
    li      $v0, 10
    syscall
```