TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

----o0o----

BÁO CÁO CUỐI KÌ

THỰC HÀNH KIẾN TRÚC MÁY TÍNH

Đề bài: Đề bài 4 và đề bài 7

Nhóm sinh viên (nhóm 6):

1. Nguyễn Văn Hồng 20184113
2. Dương Quốc Khánh 20194592

Hà Nội 2022

Mục lục

Phần 1: Postscript CNC Marsbot (Đề tài 4)

1. Yêu cầu thực hiện

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. Các postscript là:

- Postscript 1 : chữ DCE được gia công
- Postscript 2 : chữ VAN được gia công
- Postscript 3 : chữ HONG được gia công

2. Ý tưởng

Cần nạp vào Marsbot một mảng cấu trúc gồm 3 phần tử là <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. Mỗi phần tử ngăn cách nhau bởi dấu “,”. Một bộ 3 phần tử cách nhau bởi dấu “;”. Các phần tử được lưu cố định trong mã nguồn.

Thực hiện ấn nút , nếu ấn nút 0 thì đọc postscript 1, nếu nút ấn là 4 thì đọc postscript 2, nếu nút ấn là 8 thì đọc postscript 3

3. Phân tích cách thực hiện

3.1 Xử lý trên key matrix

Nhãn polling dùng để kiểm tra nút ấn. Nếu nút ấn là một trong các số 0,4,8 thì nhảy đến nhãn START để bắt đầu chương trình, nếu không thì nhảy đến nhãn COME_BACK để quay lại đến khi ấn được các nút trên.

3.2 Xử lý trên marsbot:

Trước tiên nhảy đến nhãn GO bật chế độ di chuyển MOVING của robot. Sau đó nhảy vào nhãn READ_PSCRIPT để đọc chuỗi mã nguồn đã chọn, nhãn READ_DOC để đọc giá trị góc được lưu vào $t0, đọc từng

chữ số(từng ký tự) cho tới khi gặp “ , “ thì vào nhãn READ_TIME để đọc giá trị thời gian được lưu vào $t1.

Trước khi đọc giá trị thời gian, cần nhảy đến nhãn GOC để xác định phương hướng di chuyển của robot.

Sau đó đọc từng chữ số cho tới khi gặp “ , ” thì nhảy đến nhãn READ_TRACK.

Nhãn READ_TRACK để đọc các số 0, 1 tương ứng không lưu vết và lưu vết. Nếu là 0 thì tới nhãn UNTRACK để kích hoạt không truy vết, ngược lại thì tới nhãn TRACK để kích hoạt lưu vết.

Nếu gặp “ ;” bỏ qua ký tự này và tiếp tục đọc tiếp. Đọc cho đến khi hết postscript

4. Ý nghĩa của các thanh ghi được sử dụng:

STT	Tên thanh ghi	Ý nghĩa thanh ghi
1	$t0	Chứa giá trị góc chuyển động
2	$t1	Chứa giá trị thời gian.
3	$t2	Chứa biến chạy (dịch vị trí để đọc chữ số (ký tự) tiếp theo).
4	$t3	IN_ADRESS_HEXA_KEYBOARD (trên key martrix , kiểm tra xem hàng (cột) nào).
5	$t4	OUT_ADRESS_HEXA_KEYBOARD (trên key matrix, đọc địa chỉ).
6	$t5	Lưu địa chỉ hàng 1(2/3), chứa giá trị khi đọc từng chữ số trong chuỗi mã nguồn.
7	$t7	Chứa địa chỉ chuỗi postscript
8	$a0	Chứa giá trị so sánh.
9	$a1	Chứa địa chỉ chuỗi postscript.

10	$zero	Chứa giá trị 0.
11	$v0 (= 32)	Chứa giá trị giữ cho con robot di chuyển trong khoảng thời gian $t1 sau đó chuyển hướng.
12	$at	Chưa giá trị địa chỉ mars bot.
13	$k0	Chứa giá trị 0 (1).
14	$ra	Chứa địa chỉ trả về.

5. Ý nghĩa các chương trình con, các hàm được sử dụng

STT	Tên chương trình con	Ý nghĩa chương trình con
1	polling	Kiểm tra nút ấn.
2	NOT_NUMPAD_0	Nếu nút ấn không phải là 0.
3	NOT_NUMPAD_4	Nếu nút ấn không phải là 4.
4	COME_BACK	Nếu nút ấn không phải là 0, 4, 8 thì quay lại kiểm tra lại.
5	START	Bắt đầu khởi tạo cho con robot di chuyển.
6	READ_PSCRIPT	Đọc mã nguồn.
7	READ_GOC	Đọc góc chuyển động.
8	XAC_DINH_HUONG	Nhảy đến nhãn GOC để xác định hướng.
9	READ_TIME	Đọc thời gian.
10	READ_TRACK	Đọc có truy vết hay không.
11	TIEP_TUC	Quay lại vòng lặp đọc pscript đến khi nào gặp “;”.

12	GO	Xác định robot di chuyển.
13	STOP	Dừng robot (kết thúc chương trình).
14	TRACK	Có lưu vết.
15	UNTRACK	Không lưu vết.
16	END	Kết thúc chương trình.

6. Mã nguồn

```
.data

#  an nut numpad 0 thuc hien gia cong DCE

postscript1: .asciiz
"90,0,2000;180,0,3000;180,1,5790;80,1,500;70,1,500;60,1,500;50,1,500
;40,1,500;30,1,500;20,1,500;10,1,500;0,1,500;350,1,500;340,1,500;330,
1,500;320,1,500;310,1,500;300,1,500;290,1,500;280,1,490;90,0,7000;27
0,1,500;260,1,500;250,1,500;240,1,500;230,1,500;220,1,500;210,1,500;
200,1,500;190,1,500;180,1,500;170,1,500;160,1,500;150,1,500;140,1,50
0;130,1,500;120,1,500;110,1,500;100,1,500;90,1,1000;90,0,5000;270,1,
3000;0,1,5800;90,1,3000;180,0,2900;270,1,3000;90,0,3000;"

#  an nut numpad 4 thuc hien gia cong VAN

postscript2: .asciiz
"90,0,2000;180,0,2000;160,1,5321;20,1,5321;90,0,2500;200,1,5321;20,
0,2341;90,1,2039;340,1,3300;160,0,3300;160,1,2500;90,0,1500;0,1,530
0;150,1,5474;0,1,5300;90,0,3000;"

#  an nut numpad 8 thuc hien gia cong HONG

postscript3: .asciiz
"90,0,2000;180,0,2000;180,1,6000;0,0,3000;90,1,3500;0,1,3000;180,1,6
000;90,0,5000;80,1,500;70,1,500;60,1,500;50,1,500;40,1,500;30,1,500;2
0,1,500;10,1,500;0,1,500;350,1,500;340,1,500;330,1,500;320,1,500;310,
1,500;300,1,500;290,1,500;280,1,500;270,1,500;260,1,500;250,1,500;24
0,1,500;230,1,500;220,1,500;210,1,500;200,1,500;190,1,500;180,1,500;
170,1,500;160,1,500;150,1,500;140,1,500;130,1,500;120,1,500;110,1,50
```

```
0;100,1,500;90,1,500;90,0,5000;0,1,6000;150,1,6228;0,1,6000;90,0,500
0;270,1,500;260,1,500;250,1,500;240,1,500;230,1,500;220,1,500;210,1,
500;200,1,500;190,1,500;180,1,500;170,1,500;160,1,500;150,1,500;140,
1,500;130,1,500;120,1,500;110,1,500;100,1,500;90,1,1000;0,1,2500;270
,1,2000;90,0,5000;"


.eqv HEADING 0xffff8010      # Integer: An angle between 0 and 359

.eqv MOVING 0xffff8050       # Boolean: whether or not to move

.eqv LEAVETRACK 0xffff8020       # Boolean (0 or non-0):

.eqv WHEREX 0xffff8030       # Integer: Current x-location of MarsBot

.eqv WHEREY 0xffff8040       # Integer: Current y-location of MarsBot

# Key matrix

.eqv OUT_ADRESS_HEXA_KEYBOARD 0xFFFF0014

.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012


.text

# xu ly tren keymatrix

        li $t3, IN_ADRESS_HEXA_KEYBOARD

        li $t4, OUT_ADRESS_HEXA_KEYBOARD

polling:

        li $t5, 0x01                            # check row 1 with key 0, 1, 2, 3

        sb $t5, 0($t3)                              # must reassign expected
row

        lb $a0, 0($t4)                              # $a0 = read scan code of
key button
```

```
        bne $a0, 0x11, NOT_NUMPAD_0  # if not numpad 0
        la $a1, postscript1                     # a1 = address of
postsciprt 1
        j START


        NOT_NUMPAD_0:
        li $t5, 0x02                            # check row 2 with key 4,
5, 6, 7
        sb $t5, 0($t3)                          # must reassign expected
row
        lb $a0, 0($t4)                          # read scan code of key
button
        bne $a0, 0x12, NOT_NUMPAD_4  # if not numpad 4
        la $a1, postscript2                     # a1 = address of
postsciprt 2
        j START


        NOT_NUMPAD_4:
        li $t5, 0X04                            # check row 3 with key 8,
9, a, b
        sb $t5, 0($t3)                          # must reassign expected
row
        lb $a0, 0($t4)                          # read scan code of key
button
        bne $a0, 0x14, back_to_polling      # if not numpad 8
```

```
        la $a1, postscript3                 # a1 = address of
postsciprt 3

        j START

 back_to_polling: j polling                 # neu numpad 0, 4, 8 khong
duoc chon -> quay lai doc tiep



# xu li tren marsbot

START:

        jal GO


READ_POSTCRIPT:

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

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


READ_ROTATE:                                # doc goc quay

                                            # $t6 = i = 0 , i la chi so
duyet mang xau

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

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

        beq $t5, 0, END_PROGRAM             # if $t5 == null -> ket thuc
doc pscript

        beq $t5, 44, READ_TRACK             # gap dau "," chuyen den
READ_TRACK
```

```
        mul $t0, $t0, 10                # $t0 = $t0 * 10
        addi $t5, $t5, -48              # $t5 = $t5 - 48 -> $t5 =
ma ASCII -48 = gia tri ki tu can tim
        add $t0, $t0, $t5               # $t0 = $t0 + $t5
        addi $t6, $t6, 1                # $t6 = $t6 +1 ->  tang so bit
can dich chuyen len 1
        j READ_ROTATE                   # quay lai doc tiep den khi
nao gap ','

READ_TRACK:
        add $a0, $zero, $t0             # Marsbot rotates $t0* and start
running
        jal ROTATE

        addi $t6, $t6, 1                # $t6 = $t6 + 1 -> tang so bit
can dich chuyen len 1
        add $t7, $a1, $t6               # $t7= $a1 + $t6 =
postscript[0] + i = address of postscript[i], dich bit
        lb $t9, 0($t7)                  # t9  (1 OR 0)
        addi $t9, $t9, -48              # $t9 = $t9 - 48 -> $t5 = ma
ASCII -48 = gia tri ki tu can tim
        addi $t6, $t6, 1                # $t6 = $t6 + 1 -> tang so bit
can dich chuyen len 1
READ_TIME:                              # doc thoi gian chuyen
dong.
        addi $t6, $t6, 1                # $t6 = $t6 + 1 -> tang so bit
can dich chuyen len 1
```

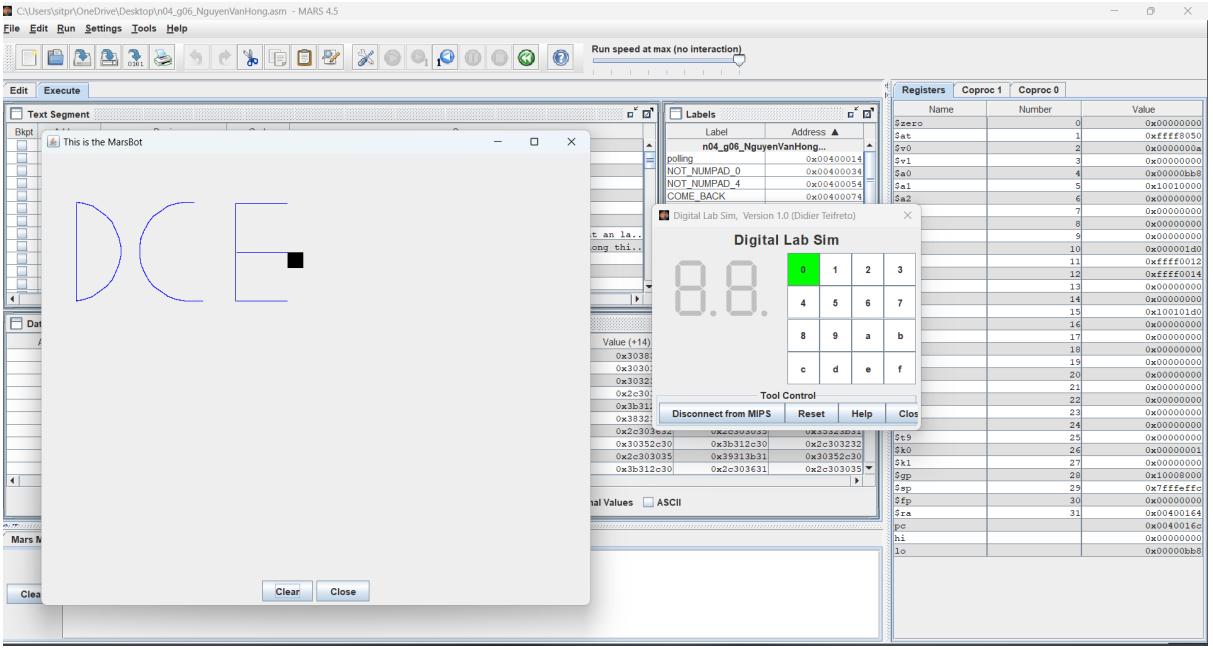
```
        add $t7, $a1, $t6                   # $t7= $a1 + $t6 =
postscript[0] + i = address of postscript[i]

        lb $t5, 0($t7)                      # $t5= value at $t7 =
postscript[i], doc cac ki tu cua pscript

        beq $t5, 59, RUNNING                # gap dau ";" chuyen den
cau truc moi

        mul $t1, $t1, 10                # $t1 = $t1 * 10

        addi $t5, $t5, -48              # $t5 = $t5 - 48 -> $t5 = ma
ASCII -48 = gia tri ki tu can tim

        add $t1, $t1, $t5               # $t1 = $t1 + $t5

        j READ_TIME                         # quay lai doc tiep den khi
gap dau ','


RUNNING:

        addi $v0,$zero,32                   # Keep running by
sleeping in $t1 ms

        add $a0, $zero, $t1             # $a0 = $t1

        beq $t9, $zero, NON_CUT         # 1=cat | 0=khongcat

        jal UNTRACK                         # keep old track

        jal TRACK                       # draw new track

        j READ_NEXT_PHASE

NON_CUT:

        jal UNTRACK                         # keep old track


READ_NEXT_PHASE:
```

```
        syscall
        addi $t6, $t6, 1                # tang so bit can dich chuyen
len 1, bo qua dau ';'
        j READ_POSTCRIPT                # quay lai doc tiep postscript



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
```

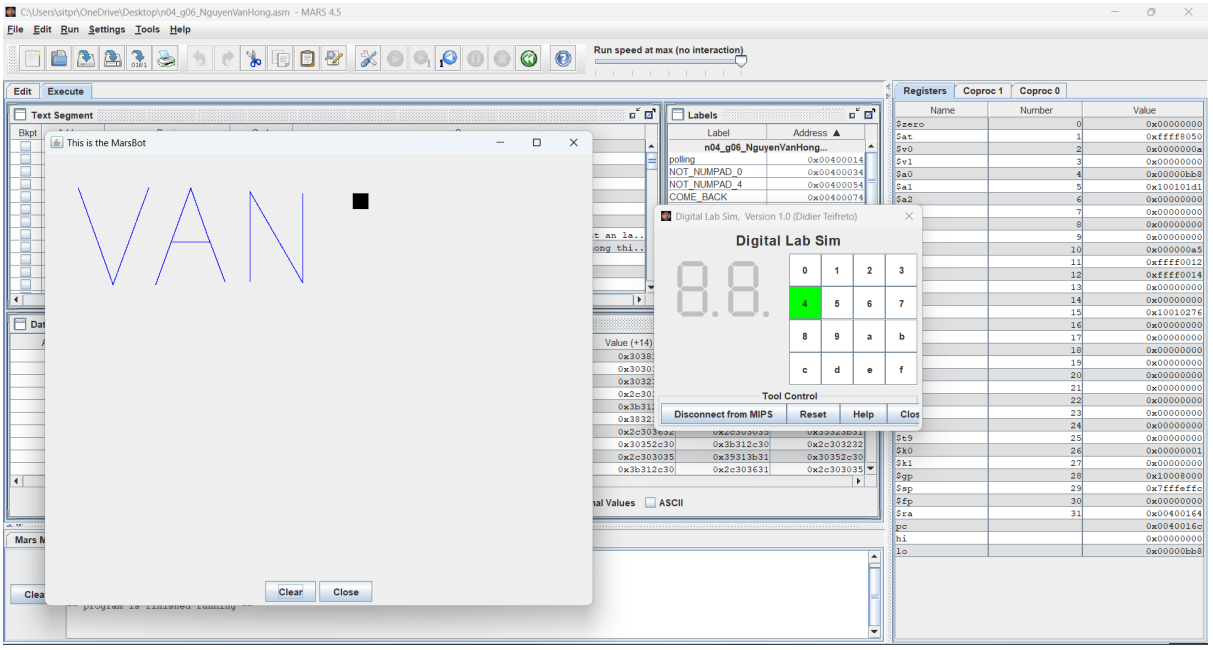
```
UNTRACK:
        li $at, LEAVETRACK
        sb $zero, 0($at)
        jr $ra

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

END_PROGRAM:
        jal STOP
        li $v0, 10
        syscall
        j polling
```

7. Ảnh chụp màn hình kết quả

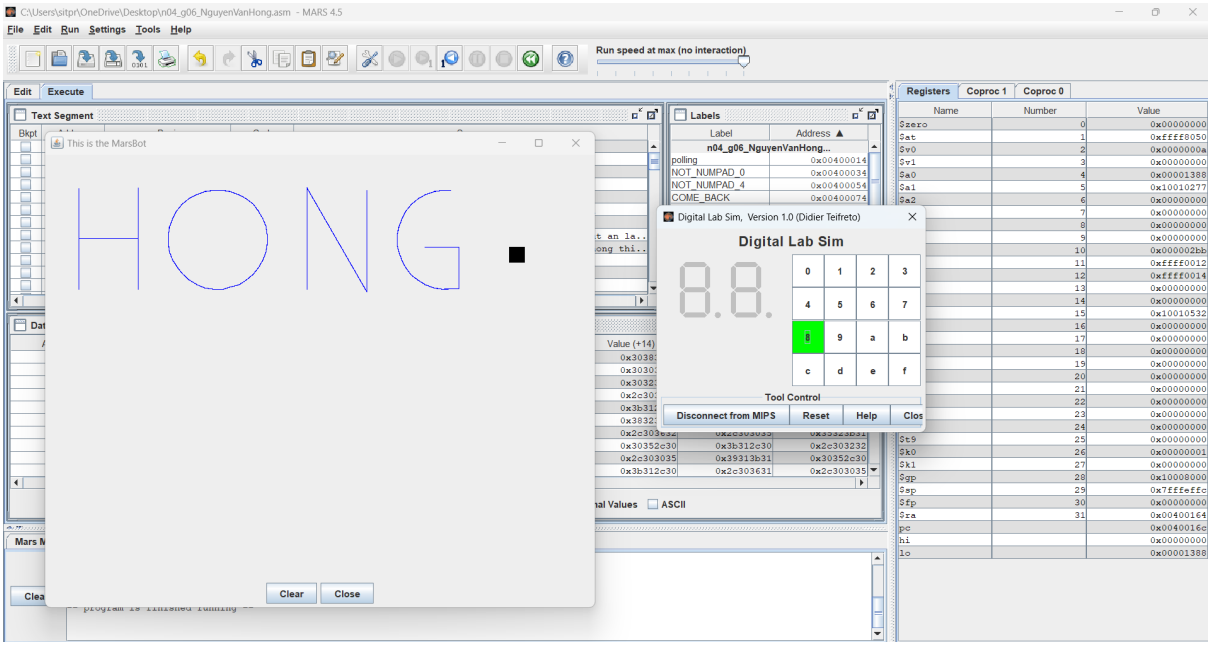
Kết quả chạy khi đọc postscript 1:



Kết quả chạy khi đọc postscript 2:



Kết quả chạy khi đọc postscript 3:



Phần 2: Chương trình kiểm tra cú pháp lệnh MIPS (Đề tài 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 cần kiểm tra, lưu vào một chuỗi input .
• Một câu lệnh hợp ngữ gồm 4 phần: mã opcode, toán hạng 1, toán hạng 2, toán hạng 3. Quy ước các dạng của toán hạng: 0 – null, 1 – register, 2 – constant, 3 – label, 4 - cụm offset(base).
• Sau khi tách một thành phần của chuỗi tương ứng với thành phần của câu lệnh hợp ngữ. Đưa từng phần cắt được đi so sánh với kiểu dữ liệu quy ước. Nếu phù hợp thì tiếp tục so sánh các phần phía sau. Nếu không phù hợp thì báo cho người dùng opcode hoặc toán hạng không hợp lệ.
+ Đầu chương trình, khởi tạo một chuỗi opcode chuẩn, opcode sau khi cắt được thì sẽ so sánh với chuỗi này bằng cách so sánh từng ký tự, nếu ký tự khác nhau thì so sánh với opcode tiếp theo trong chuỗi cho đến khi kết thúc.
+ Nếu không có opcode phù hợp -> opcode nhập vào không hợp lệ, -> báo opcode không hợp lệ và exit. Nếu có opcode hợp lệ -> tìm khuôn

dạng các toán hạng phù hợp với opcode. Đầu chương trình khi tạo chuỗi opcode chuẩn thì sẽ tạo một chuỗi khuôn dạng tương ứng với từng lệnh bằng cách so vị trí lệnh và khuôn lệnh. Nếu opcode phù hợp thì sẽ lấy được khuôn dạng lệnh và tiến hành kiểm tra lần lượt các toán hạng. Nếu có 1 toán hạng không hợp lệ -> báo ra màn hình run I/O và exit
+ Tương tự opcode chuẩn thì cũng có một chuỗi register chuẩn được tạo ở đầu chương trình. Nếu toán hạng có mã là 1 -> nó là thanh ghi và sẽ so sánh với chuỗi các regisster chuẩn.
+ Trường hợp toán hạng là constant: một constant 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.
+ Trường hợp 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.
+ Trường hợp toán hạng null: kiểm tra xem chuỗi vừa cắt được có kí tự nào hay không, nếu có -> không hợp lệ, nếu không có kí tự nào -> hợp lệ và thông báo câu lệnh hợp lệ -> Thoát chương trình.
+ Trường hợp toán hạng là một cụm offset(base): kiểm tra xem dấu mở ngoặc '(' và đóng ngoặc ')' có hợp lệ không? Có đủ 2 dấu đóng và mở ngoặc hay không. Nếu không -> không hợp lệ. Nếu có hợp lệ -> cut chuỗi này thành 2 thành phần là constant và register -> kiểm tra từng thành phần. 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 nào khác hay không. Nếu còn -> câu lệnh không hợp lệ. Và báo kết quả của câu lệnh vừa nhập -> Hỏi người dùng có muốn kiểm tra tiếp một câu lệnh khác hay không?
• Nếu người dùng chọn không -> Kết thúc chương trình.

3. Mã nguồn

```
.data
#cau lenh mips gom opcode va 3 toan hang.
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-"
#ma opcode hop le:
opcode: .asciiz
"lw-lb-sw-sb-addi-add-addiu-addu-and-andi-beq-bne-div-divu-j-jal-lui-mfh
i-mflo-mul-nop-nor-or-ori-sll-slt-slti-sub-subu-syscall-xor-xori-"
#quy uoc toan hang: 1 - thanh ghi, 2 - so, 3 - Label, 4 - offset(base):
number(register), 0 - null
#toan hang tuong ung voi cac opcode tren:
operand: .asciiz
"140-140-140-140-112-111-112-111-111-112-113-113-110-110-300-300-1
20-100-100-111-000-111-111-112-112-111-112-111-111-000-111-112-"

msg1: .asciiz "Nhap lenh can kiem tra: "
msg2: .asciiz "\nOpcode: "
msg21: .asciiz ", hop le.\n"
msg22: .asciiz ", khong hop le.\n"
msg3: .asciiz "Toan hang: "
msg4: .asciiz "Lenh can kiem tra"
msg5: .asciiz "1) Kiem tra them lenh \n0) Thoat \n "
msg6: .asciiz "Khuan dang lenh: "
msg7: .asciiz "\n"
input: .space 100 # chuoi dau vao
tmp: .space 20 #bien tmp luu thanh phan cat duoc cua opcode
tmp2: .space 20 # luu khuon dang code
tmp3:.space 20 # thanh phan cat duoc o offset(base)
.text
main:
Input: # lay dau vao
        li      $v0, 4
        la      $a0, msg1 # in ra thong bao "Nhap lenh can kiem tra"
        syscall
        li      $v0, 8
        la      $a0, input # nhap vao lenh can kiem tra
        li      $a1, 100 # toi da 100 ki tu
        syscall
        #tach chu va so sanh
```

```
	la	$s0, input #dia chi input
	add	$s1, $zero, $zero # i -> dem tung ky tu trong tmp
readOpcode:
	add	$a0, $s0, $zero # truyen tham so vao cutComponent
	add	$a1, $s1, $zero
	la	$a2, tmp
	jal	cutComponent
	add	$s1, $v0, $zero # so ky tu cat duoc
	add	$s7, $v1, $zero #so ky tu cat duoc
checkOpcode:
	la	$a0, tmp
	add	$a1, $s7, $zero
	la	$a2, opcode
	jal	compareOpcode
	add	$s2, $v0, $zero #check opcode
	add	$s3, $v1, $zero #count matching voi khuon dang toan hang
	li	$v0, 4
	la	$a0, msg2
	syscall
	li	$v0, 4
	la	$a0, tmp
	syscall
	bne	$s2, $zero, validOpcode # neu opcode hop le -> valid
invalidOpcode: #opcode ko hop le
	li	$v0, 4
	la	$a0, msg22
	syscall
	j	exit
validOpcode:
	li	$v0, 4
	la	$a0, msg21
	syscall
	#-------lay khuon dang tuong ung voi opcode
	la	$a0, operand
	add	$a1, $s3, $zero #truyen vao count
	jal	getOperand #tra ve tmp2 - khuon dang
```

```
        li $v0,4
        la $a0,msg6
        syscall

        li      $v0, 4
        la      $a0, tmp2
        syscall

        li $v0,4
        la $a0,msg7
        syscall


        la      $s4, tmp2   #khuon dang
        add   $s5, $zero, $zero #toan hang 1 2 3  - dem
        add   $t9, $zero, 48 #0
        addi  $t8, $zero, 49 #1
        addi  $t7, $zero, 50 #2
        addi  $t6, $zero, 51 #3
        addi  $t5, $zero, 52 #4
Cmp: # kiem tra dang cua tung toan hang và check
        slti    $t0, $s5, 3
        beq   $t0, $zero, end
        #-----------lay toan hang 1
        add   $a0, $s0, $zero
        add   $a1, $s1, $zero
        la      $a2, tmp
        jal     cutComponent
        add   $s1, $v0, $zero
        add   $s7, $v1, $zero #so ky tu co trong tmp
        #--so sanh toan hang 1
        add   $t0, $s5, $s4
        lb      $s6, 0($t0) #dang cua toan hang i
        beq   $s6, $t8, reg
        beq   $s6, $t7, number
        beq   $s6, $t6, label
        beq   $s6, $t5, offsetbase
```

```
        beq   $s6, $t9, null
reg:
        la    $a0, tmp
        add   $a1, $s7, $zero
        la    $a2, register
        #tra ve 0 -> error, 1 -> ok
        jal   compareOpcode
        j     checkValid
number:
        la    $a0, tmp
        add   $a1, $s7, $zero
        jal   checkNumber
        j     checkValid
label:
        la    $a0, tmp
        add   $a1, $s7, $zero
        jal   checkLabel
        j     checkValid
offsetbase:
        la    $a0, tmp
        add   $a1, $s7, $zero
        jal   checkOffsetBase
        j     checkValid
null:
        j     print
checkValid:
        add   $s2, $v0, $zero
        li    $v0, 4
        la    $a0, msg3
        syscall
        li    $v0, 4
        la    $a0, tmp
        syscall
        beq   $s2, $zero, error
        j     ok
updateCheck:     #buoc lap
        addi  $s5, $s5, 1
```

```
        j       Cmp

error:
        li      $v0, 4
        la      $a0, msg22
        syscall
        j       exit
ok:
        li      $v0, 4
        la      $a0, msg21
        syscall
        j       updateCheck
end:
        add   $a0, $s0, $zero
        add   $a1, $s1, $zero
        jal     cutComponent
        add   $s1, $v0, $zero #i hien tai
        add   $s7, $v1, $zero #so ky tu co trong tmp
print:
        li      $v0, 4
        la      $a0, msg4
        syscall
        bne   $s7, $zero, error
        li      $v0, 4
        la      $a0, msg21
        syscall
exit:
        repaetMain:
                li      $v0, 4
                la      $a0, msg5
                syscall
                li      $v0, 8
                la      $a0, input
                li      $a1, 100
                syscall
                checkRepeat:
                        addi   $t2, $zero, 48
```

```
                addi  $t3, $zero, 49
                add   $t0, $a0, $zero #ki tu dau tien
                lb    $t0, 0($t0)
                beq   $t0, $t2, out# =0
                beq   $t0, $t3, main
                j     repaetMain
        out:
                li $v0, 10 #exit
                syscall
#-----------------------------------------------------
# tach toan hang,, opcode tu chuoi dau vao
# a0 address input, a1 i-> dem tmp. a2 address tmp
# v0 i = i+ strlen(tmp), v1 strlen(tmp)
#-----------------------------------------------------
cutComponent:
         addi $sp, $sp, -20 # Ngan nho stack
        sw    $ra, 16($sp) # Luu gia tri thanh ra luc nhay vao ram
        sw    $s0, 12($sp) # Dia chi cua input luu vao thanh ram
        sw    $s2, 8($sp)#j
        sw    $s3, 4($sp) #input[i]
        sw    $s4, 0($sp) #dau phay

        addi  $s0, $zero, 32 #space
        addi  $t2, $zero, 10 #\n xuong dong moi
        addi  $s4, $zero, 44 #dau phay
        addi  $t3, $zero, 9 #\tab
checkSpace: #bo qua , \t space
        add   $t0, $a0, $a1 #dia chi input[i]
        lb    $s3, 0($t0) #Luu gia cua input[i] vao thanh ghi $s3
        beq   $s3, $s0 cutSpace # so sanh ki tu voi space neu phai nhay
den cutSpace
        beq   $s3, $t3 cutSpace #  so sanh voi tab neu phai nhay den
cutSpace
        beq   $s3, $s4 cutSpace # so sanh voi dau phay neu phai nhay
den cutSpace
        j     cut
```

```
cutSpace: # neu nguoi nhap vao dau dong la ("space","xuong dong",",")
se bi bo qua
        addi  $a1, $a1, 1 # dia chi cong them 1
        j     checkSpace
cut:
        add   $s2, $zero, $zero #j = 0
loopCut:
        beq   $s3, $s0 endCut # space
        beq   $s3, $s4, endCut # dau phay
        beq   $s3, $zero, endCut # 0
        beq   $s3, $t2, endCut # xuong dong moi
        beq   $s3, $t3 endCut # tab
        add   $t0, $a2, $s2 #dia chi tmp[j]
        sb    $s3, 0($t0) #luu gia tri thanh ghi s3 vao tmp[j]
        addi  $a1, $a1, 1 # tang dia chi cua input len 1
        add   $t0, $a0, $a1 #dia chi input[i]
        lb    $s3, 0($t0) #Luu gia tri cua input[i] vao thanh ghi $s3

        addi  $s2, $s2, 1 # tang dia chi cua tmp len 1
        j     loopCut
endCut:
        add   $t0, $a2, $s2 #dia chi tmp[j]
        sb    $zero, 0($t0) #luu tmp[j] = '\0'
        add   $v0, $a1, $zero # so ki tu cat duoc
        add   $v1, $s2, $zero # so ki tu cat duoc
        lw    $ra, 16($sp) # Luu gia tri tu ram vao thanh ghi $ra de nhay
quay lai
        lw    $s0, 12($sp) # Luu lai dia chi input vao thanh ghi $s0
        lw    $s2, 8($sp) # Tra lai gia tri cho thanh ghi $s2
        lw    $s3, 4($sp) # Tra lai gia tri cho thanh ghi $s3
        lw    $s4, 0($sp) # Tra lai gia tri cho thanh ghi $s4
        addi  $sp, $sp, 20 # Tra lai dia chi ban dau cho thanh ghi $sp
        jr    $ra

#-----------------------------------------------------
# so sanh toan hang, opcode voi toan hang, opcode chuan
```

```
# a0 address tmp, a1 strlen(tmp), a2 adress cua chuoi opcode, register
chu?n
# v0 0|1, v1 count vi tri cua opcode
#-----------------------------------------------------
compareOpcode:
        addi  $sp, $sp, -24
        sw    $ra, 20($sp)
        sw    $s1, 16($sp) #i -> opcode
        sw    $s2, 12($sp) #j -> tmp
        sw    $s3, 8($sp) #tmp[j]
        sw    $s4, 4($sp)#luu opcode[i]
        sw    $s5, 0($sp) # - 45

        beq   $a1, $zero, endCmp

        add   $s1, $zero, $zero
        add   $s2, $zero, $zero
        addi  $s5, $zero, 45
        addi  $v0, $zero, 1
        addi  $v1, $zero, 1
loopCmp:
        add   $t0, $a2, $s1 #dia chi opcode[i]
        lb    $s4, 0($t0) #luu opcode[i]
        beq   $s4, $s5, checkCmp
        beq   $s4, $zero, endCmp
        add   $t0, $a0, $s2 #dia chi tmp[j]
        lb    $s3, 0($t0) #luu tmp[j]
        bne   $s3, $s4, falseCmp # Khac nhau se nhay den falseCmp

        addi  $s1, $s1, 1
        addi  $s2, $s2, 1
        j     loopCmp
checkCmp:
        bne   $a1, $s2, falseCmp
trueCmp:
        addi  $v0, $zero, 1
        j     endF
```

```
falseCmp:
        addi  $v0, $zero, 0
        addi  $s2, $zero, 0
        loopXspace:
                beq   $s4, $s5, Xspace
                addi  $s1, $s1, 1
                add   $t0, $a2, $s1 #dia chi opcode[i]
                lb    $s4, 0($t0) #luu opcode[i]
                j     loopXspace
                Xspace:
                        add   $v1, $v1, 1
                        addi  $s1, $s1, 1
                        j     loopCmp
endCmp:
        addi  $v0, $zero, 0
endF:
        addi  $v1, $v1, -1
        lw    $ra, 20($sp)
        lw    $s1, 16($sp)
        lw    $s2, 12($sp)
        lw    $s3, 8($sp)
        lw    $s4, 4($sp)
        lw    $s5, 0($sp)
        addi  $sp, $sp, 24
        jr    $ra
#-----------------------------------------------------
# lay khuon dang toan hang ung voi opcode
# a0 address chuoi operand - vi tri tuong ung voi opcode, a1 count
# tra ve chuoi opcode tuong ung o tmp2 =
#-----------------------------------------------------
getOperand:
        addi  $sp, $sp, -20
        sw    $s0, 16($sp) #i
        sw    $s1, 12($sp) #op[i]
        sw    $s2, 8($sp) # 45
        sw    $s3, 4($sp) # address tmp2
```

```
        sw      $s4, 0($sp)# j

        addi    $t0, $zero, 4 #moi khuon dang chiem 4 byte
        mul     $s0, $a1, $t0 # i hien tai
        addi    $s2, $zero, 45 # -
        la      $s3, tmp2
        add     $s4, $zero, $zero #j
loopGet:
        add     $t0, $a0, $s0 #dia chi op
        lb      $s1, 0($t0)
        beq     $s1, $s2, endGet #gap - -> out
        add     $t0, $s3, $s4 #dia chi tmp[i]
        sb      $s1, 0($t0)

        addi    $s0, $s0, 1
        addi    $s4, $s4, 1
        j       loopGet
endGet:
        add     $t0, $s3, $s4 #dia chi tmp[i]
        sb      $zero, 0($t0)
        lw      $s0, 16($sp) #i
        lw      $s1, 12($sp) #op[i]
        lw      $s2, 8($sp) #
        lw      $s3, 4($sp) #
        lw      $s4, 0($sp)#
        addi    $sp, $sp, 20
        jr $ra
#-----------------------------------------------------
# kiem tra chuoi tmp co la so hay ko -> 0 -> sai, 1-> dung
# a0 address tmp, a1 strlen(tmp)
# v0 0|1
#-----------------------------------------------------
checkNumber:
        add     $sp, $sp, -24
        sw      $ra, 20($sp)
        sw      $s4, 16($sp) #+
        sw      $s3, 12($sp) #-
```

```
        sw    $s0, 8($sp)
        sw    $s1, 4($sp)
        sw    $s2, 0($sp) #1
        add   $v0, $zero, 0
        add   $s0, $zero, $zero #dem i

        beq   $a1, $zero, endCheckN
checkFirstN:
        addi  $s3, $zero, 45 # -
        addi  $s4, $zero, 43 # +
        addi  $s2, $zero, 1
        add   $t0, $a0, $s0 #toanhang[i]
        lb    $s1, 0($t0)
        #check - +  -> 123
checkMinus:
        bne   $s1, $s3, checkPlus
        beq   $a1, $s2, endCheckN
        j     update
checkPlus:
        bne   $s1, $s4, _123
        beq   $a1, $s2, endCheckN
        j     update
#lb    $t2, 0($s1)

checkI:
        slt   $t0, $s0, $a1
        beq   $t0, $zero, trueN
        add   $t0, $a0, $s0 #toanhang[i]
        lb    $s1, 0($t0)
_123: #48 -> 57
        slti  $t0, $s1, 48
        bne   $t0, $zero, endCheckN
        slti  $t0, $s1, 58
        beq   $t0, $zero, endCheckN
update:
        addi  $s0, $s0, 1
        j     checkI
```

```
trueN:
	addi	$v0, $v0, 1
endCheckN:
	lw	$ra, 20($sp)
	lw	$s4, 16($sp) #+
	lw	$s3, 12($sp) #-
	lw	$s0, 8($sp)
	lw	$s1, 4($sp)
	lw	$s2, 0($sp)
	add	$sp, $sp, 24
	jr	$ra
#-----------------------------------------------------
# kiem tra chuoi tmp co la Label hay ko, ki tu dau tien: _ | A -> _ | A | 1
# a0 -> address tmp, a1 strlen(tmp)
# v0 0|1
#-----------------------------------------------------
checkLabel:
	add	$sp, $sp, -12
	sw	$ra, 8($sp)
	sw	$s1, 4($sp)
	sw	$s0, 0($sp)
	add	$v0, $zero, 0
	add	$s0, $zero, $zero #dem i

	beq	$a1, $zero, endCheckL

checkFirstChar:
	add	$t0, $a0, $s0 #toanhang[i]
	lb	$s1, 0($t0)
	j	ABC

checkIL:
	slt	$t0, $s0, $a1
	beq	$t0, $zero, trueL
	add	$t0, $a0, $s0 #toanhang[i]
	lb	$s1, 0($t0)
_123L: #48 -> 57
```

```
        slti    $t0, $s1, 48
        bne     $t0, $zero, endCheckL
        slti    $t0, $s1, 58
        beq     $t0, $zero, ABC
        addi    $s0, $s0, 1
        j       checkIL
ABC: #65 -> 90
        slti    $t0, $s1, 65
        bne     $t0, $zero, endCheckL
        slti    $t0, $s1, 91
        beq     $t0, $zero, _
        addi    $s0, $s0, 1
        j       checkIL
_:
        add     $t0, $zero, 95
        bne     $s1, $t0, abc
        addi    $s0, $s0, 1
        j       checkIL
abc: #97  -> 122
        slti    $t0, $s1, 97
        bne     $t0, $zero, endCheckL
        slti    $t0, $s1, 123
        beq     $t0, $zero, endCheckL
        addi    $s0, $s0, 1
        j       checkIL
trueL:
        addi    $v0, $v0, 1
endCheckL:
        sw      $ra, 8($sp)
        lw      $s1, 4($sp)
        lw      $s0, 0($sp)
        add     $sp, $sp, 12
        jr      $ra
#-----------------------------------------------------
# kiem tra chuoi tmp co dung cau truc offset(base) hay khong
# a0 -> address tmp, a1 strlen(tmp)
# v0 0|1
```

```
#-----------------------------------------------------
#a0, tmp a1 strlen
checkOffsetBase:
#0($s1) -> 0_$s1_
        add   $sp, $sp, -28
        sw    $ra, 24($sp)
        sw    $s5, 20($sp) #so ki cut dk
        sw    $s4, 16($sp) # )
        sw    $s3, 12($sp) # (
        sw    $s2, 8($sp) #check
        sw    $s1, 4($sp)# tmp[i]
        sw    $s0, 0($sp) # dem i
checkO:
        slti  $t0, $a1, 5 #it nhat 5 kis tu, vd: 0($s1)
        bne   $t0, $zero, falseCheck
        addi  $s3, $zero, 40
        addi  $s4, $zero, 41
        add   $s0, $zero, $zero #i
        add   $s2, $zero, $zero #check
        addi  $t2, $zero, 1
loopCheck:
        add   $t0, $a0, $s0 #dia chi tmp[i]
        lb    $s1, 0($t0)
        beq   $s1, $zero, endLoopO
        beq   $s1, $s3, open_
        beq   $s1, $s4, close_
        j     updateO
open_:
        bne   $s2, $zero, falseCheck
        addi  $s2, $s2, 1
        addi  $t1, $zero, 32
        sb    $t1, 0($t0)
        j     updateO
close_:
        bne   $s2, $t2, falseCheck
        addi  $s2, $s2, 1
        sb    $zero, 0($t0)
```

```
        addi  $s0, $s0, 1
        bne   $s0, $a1, falseCheck

updateO:
        addi  $s0, $s0, 1
        j     loopCheck
endLoopO:
        addi  $t2, $t2, 1 # =2
        bne   $s2, $t2, falseCheck
#----
trueCheck:
        add   $s0, $zero, $zero #i
        #cut
        sw    $a0, -8($sp)
        sw    $a1, -4($sp)

        la    $a0, tmp
        add   $a1, $s0, $zero
        la    $a2, tmp3
        jal   cutComponent
        add   $s0, $v0, $zero
        add   $s5, $v1, $zero #so ky tu co trong cutword

        lw    $a0, -8($sp)
        lw    $a1, -4($sp)
        #check number
        sw    $a0, -8($sp)
        sw    $a1, -4($sp)
        la    $a0, tmp3
        add   $a1, $s5, $zero
        jal   checkNumber
        add   $s2, $v0, $zero
        lw    $a0, -8($sp)
        lw    $a1, -4($sp)
        beq   $s2, $zero, falseCheck
        #cut
```

```
        sw      $a0, -8($sp)
        sw      $a1, -4($sp)

        la      $a0, tmp
        add     $a1, $s0, $zero
        la      $a2, tmp3
        jal     cutComponent
        add     $s0, $v0, $zero
        add     $s5, $v1, $zero #so ky tu co trong cutword

        lw      $a0, -8($sp)
        lw      $a1, -4($sp)
        #checkReg
        sw      $a0, -8($sp)
        sw      $a1, -4($sp)
        sw      $a2, -16($sp)
        la      $a0, tmp3
        add     $a1, $s5, $zero
        la      $a2, register
        #tra ve 0 -> error, 1 -> ok
        jal     compareOpcode
        add     $s2, $v0, $zero
        lw      $a0, -8($sp)
        lw      $a1, -4($sp)
        lw      $a2, -12($sp)
        beq     $s2, $zero, falseCheck
        #->ket luan
        addi    $v0, $zero, 1
        j       endO
falseCheck:
        add     $v0, $zero, $zero
        j       endO
endO:
        lw      $ra, 24($sp)
        lw      $s5, 20($sp) #so ki cut dk
        lw      $s4, 16($sp) #
        lw      $s3, 12($sp) #
```

```
lw      $s2, 8($sp)
lw      $s1, 4($sp)
lw      $s0, 0($sp) #
add     $sp, $sp, 28
jr      $ra
```

4. Kết quả chạy thử

```
Nhap lenh can kiem tra: **** user input : addi $s4,s5,100

Opcode: addi, hop le.
Khuan dang lenh: 112
Toan hang: $s4, hop le.
Toan hang: s5, khong hop le.
1) Kiem tra them lenh
0) Thoat
```

```
Nhap lenh can kiem tra: **** user input : sw $s1,0($2)

Opcode: sw, hop le.
Khuan dang lenh: 140
Toan hang: $s1, hop le.
Toan hang: 0 $2, hop le.
Lenh can kiem tra, hop le.
```

```
Nhap lenh can kiem tra: **** user input : j L

Opcode: j, hop le.
Khuan dang lenh: 300
Toan hang: L, hop le.
Lenh can kiem tra, hop le.
1) Kiem tra them lenh
0) Thoat
```