ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO BÀI TẬP LỚN MÔN HỌC

THỰC HÀNH KIẾN TRÚC MÁY TÍNH

ĐỀ BÀI

Giảng viên hướng dẫn: ThS. Lê Bá Vui

Sinh viên: Võ Tá Hoan-20194568

Nguyễn Đức Trưởng-20194698

Hà Nội, 21 tháng 07 năm 2022

Nội dung

Bai 4 – Nguyễn Đức Trưởng-20194698

Yêu Cầu Đề Bài

Máy gia công cơ khí chính xác CNC Marsbot được dùng để cắt tấm kim loại theo các đường nét được qui
định trước. CNC Marsbot có một lưỡi cắt dịch chuyển trên tấm kim loại, với giả định rằng:
- Nếu lưỡi cắt dịch chuyển nhưng không cắt tấm kim loại, tức là Marsbot di chuyển nhưng không để lại vết (Track)
- Nếu lưỡi cắt dịch chuyển và cắt tấm kim loại, tức là Marsbot di chuyển và có để lại vết.
Để điều khiển Marsbot cắt đúng như hình dạng mong muốn, người ta nạp vào Marsbot một mảng cấu trúc gồm 3 phần tử:
- <Góc chuyển động>,<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)

Phân tích cách làm

1: Sau khi lấy được giá trị thông tin điều khiển từ pooling(giá trị lưu ở OUT_ADRESS_HEXA_KEYBOARD), lưu giá trị của nút bấm vào một thanh ghi

2:So sánh giá trị thanh ghi đó để xem cần thiết phái vẽ theo postscrip nào

3:đọc từ kí tự của postscrip và thực hiện lần lượt các việc sau:

- Đọc giá trị của rotate: giá trị của rotate được lưu vào thanh ghi t0, đọc kí tự tiếp theo trong postscrip, nếu không phải kí tự ‘,’ thì nhân giá trị của t0 với 10 và cộng nó với giá trị vừa đọc được, ngược lại thì thực hiện tiếp công việc tiếp theo
- Đọc giá trị của sleep time: làm tương tự như cách làm đọc giá trị của rotate và lưu vào thanh ghi t1
- Đọc giá trị ccuar Track: vẫn đọc 1 tí tự như đọc giá trị của rotate

4: sau khi đọc xong các giá trị cần thiết thì bắt đầu vẽ

Sau khi đã vẽ xong nét thì bắt đầu quay lại bước 3:

Mã Nguồn

```
# Mars bot

.eqv HEADING 0xffff8010

.eqv MOVING 0xffff8050

.eqv LEAVETRACK 0xffff8020

.eqv WHEREX 0xffff8030

.eqv WHEREY 0xffff8040
```

```
# Key matrix
.eqv OUT_ADRESS_HEXA_KEYBOARD 0xFFFF0014
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012

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

# postscript-G => numpad 4
pscript2: .asciiz
"90,5000,0;180,3000,0;270,500,1;260,500,1;250,500,1;240,500,1;230,500,1;220,500,1;210,500,1;200
,500,1;190,500,1;180,500,1;170,500,1;160,500,1;150,500,1;140,500,1;130,500,1;120,500,1;110,500,1
;100,500,1;0,2500,1;270,1000,1;90,2000,1;90,1000,0;"

# postscript-TRUONG => numpad 8
pscript3: .asciiz
"90,4000,0;180,3000,0;90,4000,1;270,2000,1;180,5790,1;90,3000,0;0,5790,1;100,500,1;120,500,1;14
0,500,1;160,500,1;180,500,1;200,500,1;220,500,1;240,500,1;260,500,1;270,50,1;145,3550,1;90,1000,
0;0,5790,0;180,5790,1;90,4000,1;0,5790,1;90,1000,0;180,5790,1;90,4000,1;0,5790,1;270,4000,1;90,
5000,0;180,5790,0;0,5790,1;145,6900,1;0,5790,1;90,4000,0;270,500,1;260,500,1;250,500,1;240,500,
1;230,500,1;220,500,1;210,500,1;200,500,1;190,500,1;180,500,1;170,500,1;160,500,1;150,500,1;140
,500,1;130,500,1;120,500,1;110,500,1;100,500,1;0,2500,1;270,1000,1;90,2000,1;180,1000,0;"

.text
# <--xu ly tren keymatrix-->
        li $t3, IN_ADRESS_HEXA_KEYBOARD
```

```
        li $t4, OUT_ADRESS_HEXA_KEYBOARD


# KEY MAXTRIX
    # $a0 gia tri key dang an

# MARBOT
        # $t0 rotate
        # $t1 time
        # $t5 ki tu hien tai dang doc trong prscrip
        # $a1 dia chi dau tien cua prscrip
        # $t6 vi tri ki tu dang doc trong prscrip



polling:
        li $t5, 0x01 # row-1 of key matrix
        sb $t5, 0($t3)
        lb $a0, 0($t4)
        bne $a0, 0x11, NOT_NUMPAD_0
        la $a1, pscript1
        j START
        NOT_NUMPAD_0:
        li $t5, 0x02 # row-2 of key matrix
        sb $t5, 0($t3)
        lb $a0, 0($t4)
        bne $a0, 0x12, NOT_NUMPAD_4
        la $a1, pscript2
        j START
        NOT_NUMPAD_4:
        li $t5, 0X04 # row-3 of key matrix
        sb $t5, 0($t3)
```

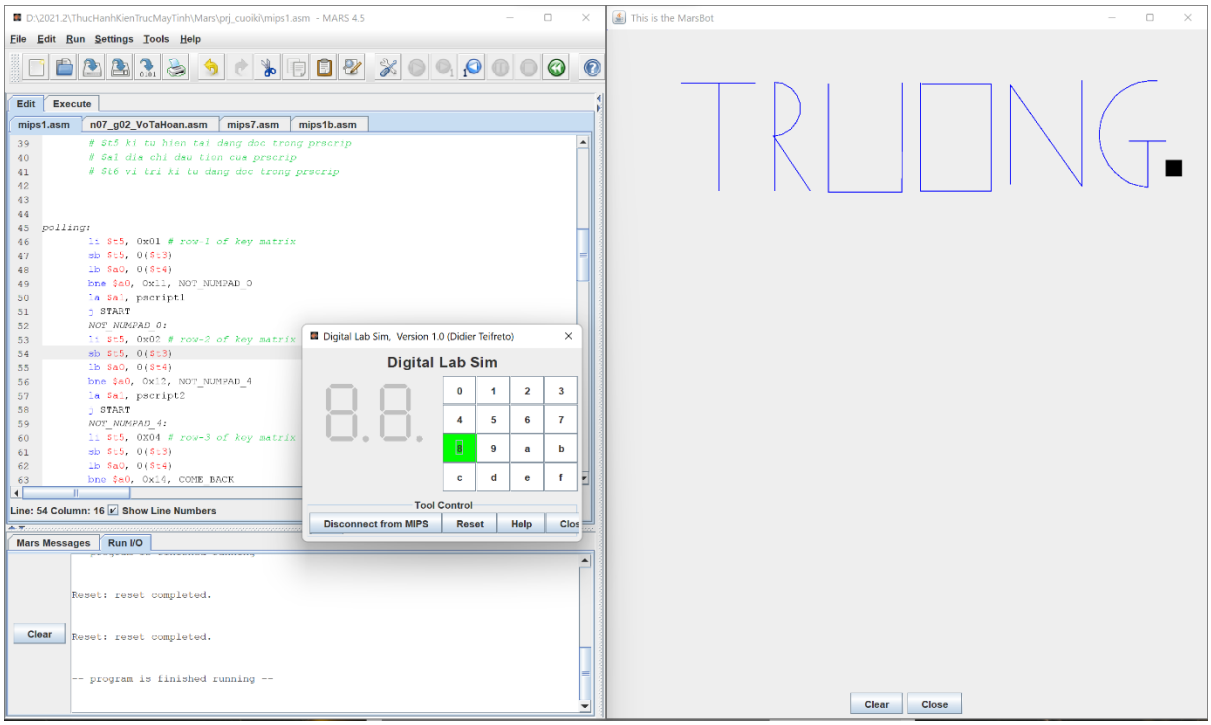
```
        lb $a0, 0($t4)
        bne $a0, 0x14, COME_BACK
        la $a1, pscript3
        j START
COME_BACK: j polling # khi cac so 0,4,8 khong duoc chon -> quay lai doc tiep
# <!--end xu ly key matrix-->

# <--xu li mars bot -->
START:
        jal GO
READ_PSCRIPT:
        addi $t0, $zero, 0 # luu gia tri rotate(goc quay)
        addi $t1, $zero, 0 # luu gia tri time

        READ_ROTATE:
        add $t7, $a1, $t6 # dich bit
        lb $t5, 0($t7)  # doc cac ki tu cua pscript
        beq $t5, 0, END # ket thuc pscript
        beq $t5, 44, READ_TIME # gap ki tu ','
        mul $t0, $t0, 10
        addi $t5, $t5, -48 # So 0 co thu tu 48 trong bang ascii.(ki tu doc dc la ma ascii cua ki tu so, nên phai tru di 48)
        add $t0, $t0, $t5  # cong cac chu so lai voi nhau.
        addi $t6, $t6, 1 # tang so bit can dich chuyen len 1
        j READ_ROTATE # quay lai doc tiep den khi gap dau ','

        READ_TIME: # doc thoi gian chuyen dong.

        addi $t6, $t6, 1
        add $t7, $a1, $t6 # ($a1 luu dia chi cua pscript)
        lb $t5, 0($t7)
        beq $t5, 44, READ_TRACK
        mul $t1, $t1, 10
```

```
        addi $t5, $t5, -48
        add $t1, $t1, $t5
        j READ_TIME # quay lai doc tiep den khi gap dau ','

        READ_TRACK:

        addi $t6, $t6, 1
        add $t7, $a1, $t6
        lb $t5, 0($t7)
        addi $t5, $t5, -48
        beq $t5, $zero, CHECK_UNTRACK # 1=track | 0=untrack
        jal UNTRACK
        jal TRACK
        j INCREAMENT

CHECK_UNTRACK:
        jal UNTRACK
INCREAMENT:
    add $a0, $t0, $zero
        jal ROTATE

    addi $v0,$zero,32 # Keep mars bot running by sleeping with time=$t1
        add $a0, $zero, $t1
        syscall
        addi $t6, $t6, 2 # bo qua dau ';'
        j READ_PSCRIPT

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

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

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

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

ROTATE:
        li $at, HEADING
        sw $a0, 0($at)
        jr $ra
END:
        jal STOP
        li $v0, 10
        syscall

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

Chạy và thử nghiệm

Chạy thử nghiệm 1 postscrip



Bai 7- VoTaHoan

Yêu Cầu 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 từbàn 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.

Phân tích cách làm

Xây dựng một mảng chứa khuôn dạng của từng lệnh với tên lệnh, kiểu của toán hạng 1, toán hạng 2, toán hạng 3.

"or***111;xor**111;lui**120;jr***100;jal**300;addi*112;add**111;sub**111;ori**112;and**111;beq**113;bne**113;j****300;nop**000;"

trong đó : mỗi tên lệnh sẽ có 5 byte bao gồm cả phần chứa dấu *

tiếp sẽ là kiểu của toán hang 1 , toán hạng 2, toán hạng 3

và kết thúc bằng dấu ;

Quy luật của kiểu toán hạng : thanh ghi = 1, hang so nguyen = 2, dinh danh = 3 hoac khong co = 0.

Sử dụng các thuật toán phân tích chuỗi để thực hiện project này

Thuật toan

1.Check Opcode Hợp Lệ

-Kiểm tra từng kí tự của chuỗi Opcode đã tách với từng kí tự của Library

-Nếu khác nhau tăng bước nhảy +9 sang vị trí chứa khuôn lệnh mới

-Nếu giống nhau tiếp tục so sánh kí tự tiếp theo

-Khi so sánh xuất hiện kí tự * trong chuỗi Library -> Kiểm tra thêm điều kiện Opcode

-Kiểm tra thêm opcode có kí tự đằng sau nữa không ? tránh trường hợp chỉ giống phần đầu ví dụ : nopp

2.Check Thanh Ghi Hợp Lệ

Tương tự checkOpcode nhưng thay chuỗi đích để so sánh từ Library bằng chuỗi tokenRegisters

và bước nhảy thay bằng 7 vì mỗi phần tử chứa trong 7 byte

3.Check hằng số nguyên hợp lệ

-Kiểm tra kí tự đầu tiên của token có phải '-' không nếu đúng thì đọc tới phần tử tiếp theo

-Kiểm tra toàn bộ phần tử , nếu xuất hiên kí tự không phải là số thì check =0 , nếu đúng toàn bộ check =1

4.Check label hợp lệ

-Kiểm tra kí tự đầu tiên bắt buộc phải không là chữ số

-Kiểm tra toàn bộ kí tự của token có nằm trong chuỗi charGroup không?

5.Check Toán tử rỗng

kiểm tra xem chuỗi toán hạng có rỗng hay không

Mã Nguồn

```
#Author : Vo Ta Hoan
.data
command: .asciiz "\n\nNhap vao mot dong lenh hop ngu: "
continueMessage: .asciiz "Ban muon tiep tuc chuong trinh?(0.Yes/1.No)"
errMessage: .asciiz "\n!!!Lenh hop ngu khong hop le. Loi cu phap!!!\n"
NF: .asciiz " :Khong hop le!\n"
endMess: .asciiz "\nHoan thanh! Lenh vua nhap vao phu hop voi cu phap!\n\n"
msg_Opcode: .asciiz "\nOpcode: "
msg_ToanHang: .asciiz "Toan hang: "
msg_HopLe: .asciiz " hop le.\n"
input: .space 100
token: .space 20
# quy luat cua library: opcode co do dai = 5 byte
# moi lenh co 3 toan hang va chi co 4 loai la: thanh ghi = 1, hang so nguyen =2, dinh danh = 3 hoac khong co = 0.
```

```
library: .asciiz
"or***111;xor**111;lui**120;jr***100;jal**300;addi*112;add**111;sub**111;ori**112;and**111;
beq**113;bne**113;j****300;nop**000;"

charGroup: .asciiz
"0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ_"

tokenRegisters: .asciiz "$zero ;$at ;$v0 ;$v1 ;$a0 ;$a1 ;$a2 ;$a3 ;$t0 ;$t1 ;$t2 ;$t3 ;$t4
;$t5 ;$t6 ;$t7 ;$s0 ;$s1 ;$s2 ;$s3 ;$s4 ;$s5 ;$s6 ;$s7 ;$t8 ;$t9 ;$k0 ;$k1 ;$gp ;$sp
;$fp ;$ra ;$0 ;$1 ;$2 ;$3 ;$4 ;$5 ;$7 ;$8 ;$9 ;$10 ;$11 ;$12 ;$13 ;$14 ;$15
;$16 ;$17 ;$18 ;$19 ;$20 ;$21 ;$22 ;$21 ;$22 ;$23 ;$24 ;$25 ;$26 ;$27 ;$28 ;$29
;$30 ;$31 ;"


#$k0=libraryIndex

#$k1=inputIndex

#$s4= checkOpcode

#$s5=checkToanHang1

#$t7= buocNhay


.text

j readData



#<--kiem tra opcode co dung hay khong ? dung: checkOpcode=1 sai : checkOpcode =0-->

checkOpcodeFunc:

#thanh ghi a2 : library

#thanh ghi a1: token

#$s4:checkOpcode

xuLyOpcode:

li $k0, 0                    # libraryIndex

li $k1, 0                    # inputIndex

li $s5, 0           # check = 0

addi $t7, $t7, 9  # buoc nhay = 9 de den vi tri opcode trong library

add  $k0, $k0, $t7          # cong buoc nhay


compare:

add $t3, $a2, $k0           # t3 tro thanh con tro cua library
```

```
lb  $s0, 0($t3)
beq $s0, 0, notFound      # khong tim thay opcode nao trong library
beq $s0, '*', check       # gap ki tu '*' -> check xem opcode co giong nhau tiep ko?.
add $t4, $a1, $k1
lb  $s1, 0($t4)    # s1= opcode[inputindex]
bne $s0, $s1,xuLyOpcode # so sanh 2 ki tu. dung thi so sanh tiep, sai thi nhay den phan tu chua khuon dang lenh tiep theo.
addi $k0, $k0, 1          # i+=1
addi $k1, $k1, 1          # j+=1
j compare
check:
lb $s1, 1($t4)     #kiem tra ki tu cuoi cung co phai = '\0'
beq $s1, '\0' ,check2 # neu ki tu tiep theo khong phai '\0' => lenh khong hop le. chi co doan dau giong.
li $s5, 0                 # checkOpcode = 0 lenh khong hop le
j endCheckOpcodeFunc
check2:
li $s5,1            # checkOpcode = 1 lenh hop le
endCheckOpcodeFunc:
jr $ra
#<-- Ket thuc kiem tra Opcode -->



#<-- kiem tra Toan Hang Rong dung hay sai ? dung : check =1 , sai : check =0 -->

checkNTFunc:
# $a1 : token
# $s5 : check
add $t1, $a1, 0
lb  $t2, 0($t1)    #t2=token[0]
bne $t2,'\0',emptyToken # if(token[0]== '\0') check = 1; else check = 0;
li $s5, 1                 #check = 1 toan hang hop le
j endCheckNTFunc
emptyToken:
```

```
li $s5, 0                    #check = 0 toan hang khong hop le
endCheckNTFunc:
jr $ra

#<-- Ket Thuc kiem tra toan hang rong -->


#<-- Kiem tra thanh ghi dung hay sai ? -->
checkTokenRegFunc:
#a1: token
#a3: tokenRegister
la $a3, tokenRegisters
li $s5, 0                    #check = 0

li $t7,-7                    # khoi tao buoc nhay = -7 de vao vong lap +7 thi = 0 la gia tri dau tien

xulyToken:
li $t0, 0 #i
li $t1, 0 #j
addi $t7, $t7, 7  # buoc nhay = 7 de den vi tri tokenRegister
add  $t0, $t0, $t7           # cong buoc nhay

compareToken:
add $t3, $a3, $t0            # t3 tro thanh con tro cua tokenRegister
lb  $s0, 0($t3)    # s0 = tokenRegister[i]
beq $s0, 0, notFound      # khong tim thay opcode nao trong library
beq $s0, ' ', checkToken # gap ki tu ' ' -> check xem co giong nhau tiep ko?.
add $t4, $a1, $t1
lb  $s1, 0($t4)    # s1 = token[j]
bne $s0,$s1,xulyToken # so sanh 2 ki tu. dung thi so sanh tiep, sai thi nhay den phan tu chua khuon danh lenh tiep theo.
addi $t0,$t0, 1   # i+=1
addi $t1,$t1, 1   # j+=1
```

```
j compareToken
checkToken:     # check co giong hoan toan hay khong
lb $s1,1($t4)     # kiem tra ki tu cuoi cung co phai = '\0'
beq $s1,'\0', checkToken2
li  $s5, 0                  # neu sai check=0
j endCheckTokenRegFunc
checkToken2:
li $s5, 1           # neu sai check=1
endCheckTokenRegFunc:
jr $ra
#<-- Ket thuc kiem tra thanh ghi -->



#<-- Kiem tra hang so nguyen -->
checkHSNFunc:

li $s5, 0 # check = 0
li $t0, 0  # i=0
li $t2, 48
li $t3, 57

add $t1, $a1, $t0
        lb  $s0, 0($t1)             # s0 = token[i]
        beq $s0, '\0', endCheckHSNFunc          # neu token[0] = null thi end function check = 0

        add $t1, $a1, $t0
        lb  $s0, 0($t1)
        bne $s0, '-', compareNum          # neu so am i++
        addi $t0, $t0, 1
        compareNum:
                add $t1, $a1, $t0
```

```
                lb  $s0, 0($t1)
                beq $s0, '\0', endCompareNum   # dung lai neu token[i] == '\0'
                beq $s0, ',', endCompareNum    # dung lai neu token[i] == ','
                beq $s0, '\n',endCompareNum    # dung lai neu token[i] == '\n'
        # neu 48< token[i] < 57 thi out chuong trinh check = 0
                slt $t5, $s0, $t2
bne $t5, $zero, endCheckHSNFunc
slt $t5, $t3, $s0
bne $t5, $zero, endCheckHSNFunc

                addi $t0, $t0, 1            # i++
                j compareNum                # quay lai vong while
        endCompareNum:
                li $s5, 1                   # neu dung het dk check = 1
endCheckHSNFunc:
jr $ra
#<-- Ket thuc kiem tra hang so nguyen -->


#<-- Kiem tra Label -->
checkIdentFunc:
li $s5, 0# check = 0
li $t0, 0 # i = 0
la $a3, charGroup #load ,
add $t3, $a1, $t0
        lb  $s0, 0($t3) # s0 = token[i]

        #ki tu dau khong duoc la so
        li $s2, 48
        li $s3, 57
slt $t5, $s2, $s0
slt $t6, $s0, $s3
```

```
and $t5, $t5, $t6           # token[0] > 48 && token[0] < 57
bne $t5, $zero, endCheckIdentFunc  # neu la so thi out func check = 0

loop1:                      # duyet tung ki tu trong token
                add $t3, $a1, $t0
                lb  $s0, 0($t3)             # s0 = token[i]
                beq $s0, '\0', endLoop1  # neu token[i] == '\0' thi out vong lap
                beq $s0, '\n',endLoop1   # neu token[i] == '\n' thi out vong lap

                li $t1, 0                           # j=0
                loop2:              # so sanh trong mang charGroup
                        add $t4, $a3, $t1
                        lb  $s1, 0($t4)    # s1 =  charGroup[j]
                        beq $s1,'\0', endCheckIdentFunc # neu khong tim thay ki tu cua token trong
charGroup -> ket thuc ham check = 0
                        beq $s0, $s1, endLoop2 #neu tim thay trong charGroup thi chuyen sang ki tu
tiep theo
                        addi $t1, $t1, 1  # j++
                        j loop2
                endLoop2:
                        addi $t0, $t0, 1            # i++
                j loop1

        endLoop1:
                li $s5, 1           # neu dung toan bo ky tu check = 1
endCheckIdentFunc:
jr $ra
#<-- ket thuc kiem tra label -->

readData: # Doc lenh nhap vao tu ban phim
li $v0, 4
la $a0, command #in ra man hinh
syscall
```

```
li $v0, 8           #readString
la $a0, input # chua dia chi cua lenh nhap vao
li $a1, 100
syscall

main:

#<--tach opcode tu chuoi input -->
la $a1, token # luu cac ki tu doc duoc vao token
readOpcode:
add $t3, $a0, $k1 # dich bit
add $t4, $a1, $k1
lb $t2, 0($t3) # doc tung ki tu cua input
sb $t2, 0($t4)
beq $t2, ' ', done # gap ki tu ' ' -> luu ki tu nay vao opcode de xu ly
beq $t2, '\0', done # ket thuc chuoi input
beq $t2, '\n', done
addi $k1, $k1, 1
j readOpcode

done:
addi  $t2, $0, '\0'
sb $t2, 0($t4) # xoa ky tu cuoi trong chuoi opcode ( '\n', ' ')


li $t7, -9                      # khoi tao buoc nhay -9
la $a2, library
jal checkOpcodeFunc             # kiem tra opcode co dung hay khong ?

beq $s5, 1, checkOpcode         # neu checkOpcode == 1 thi jump to checkOpcode
j notFound                      # neu checkOpcode != 1 thi jump to notFound
checkOpcode:            # in ra man hinh + readToanHang1
```

```
li $v0, 4
la $a0, msg_Opcode                  # opcode hop le
syscall
li $v0, 4
la $a0, token
syscall
li $v0, 4
la $a0, msg_HopLe
syscall
j readToanHang1

#<-- Bat dau xu ly toan hang 1 -->

readToanHang1:

addi $k1, $k1, 1            # tang inputIndex + 1

la $a0, input
li $t0, 0
li $t1, 0
newLibraryIndex:                    # tang libraryIndex den ma code cua Opcode trong Library
addi $t0,$k0, 3  #  3 so bieu dien dang toan hang cua lenh
add $t3, $a2, $t0
lb $t2, 0($t3)
beq $t2, ';', splitTH1
addi $k0, $k0, 1
j newLibraryIndex

#while (library[libraryIndex + 3] != ';')
            #{
            #libraryIndex++;
```

```
        #}

splitTH1:       #split Toan Hang thu 1

add $t3, $a0, $k1 # dich bit
add $t4, $a1, $t1
lb $t2, 0($t3) # doc tung ki tu cua input
sb $t2, 0($t4)
beq $t2, ',', doneSplitTH1
beq $t2,'\0', doneSplitTH1
beq $t2,'\n', doneSplitTH1
addi $k1, $k1, 1
addi $t1, $t1, 1
j splitTH1
doneSplitTH1:
addi  $t2, $0, '\0'
sb $t2, 0($t4)    # xoa ky tu cuoi trong chuoi token ( '\n', ' ')

add $t4, $a2, $k0
lb  $s7, 0($t4)
addi $s7,$s7,-48          # s7 = library[index -48]
TH1case0:
bne $s7, 0, TH1case1
jal checkNTFunc           # kiem tra Toan Hang Rong
j TH1done
TH1case1:
bne $s7, 1, TH1case2
jal checkTokenRegFunc # kiem tra Toan Hang co dang Thanh Ghi dung hay sai
j TH1done
TH1case2:
bne $s7, 2, TH1case3    # kiem tra Toan Hang co dang Hang So Nguyen dung hay sai
jal checkHSNFunc
```

```
j TH1done
TH1case3:
bne $s7, 3, TH1done
jal checkIdentFunc        # kiem tra Toan Hang co dang label dung hay sai
j TH1done
TH1done:

beq $s5, 1, checkToanHang1     # neu check == 1 thi jump to checkToanHang1
j notFound                     # else check != 1 thi jump to notFound
checkToanHang1:                # in ra man hinh + readToanHang2
beq $s7, 0, readToanHang2
li $v0, 4
la $a0, msg_ToanHang # toanHang hop le
syscall
li $v0, 4
la $a0, token
syscall
li $v0, 4
la $a0, msg_HopLe
syscall
#<-- Ket thuc xu ly Toan Hang 1 -->

#<-- Bat dau xu ly toan hang 2 -->
readToanHang2: #tuong tu xu ly Toan Hang 1
addi $k1, $k1, 1
addi $k0, $k0, 1
la $a1, token # luu cac ki tu doc duoc vao token
la $a0, input
li $t0, 0
li $t1, 0
```

```
splitTH2:

add $t3, $a0, $k1 # dich bit
add $t4, $a1, $t1
lb $t2, 0($t3) # doc tung ki tu cua input
sb $t2, 0($t4)
beq $t2, 44, doneSplitTH2 # gap ki tu ',' -> luu ki tu nay vao token de xu ly
beq $t2, 0, doneSplitTH2 # ket thuc chuoi input
beq $t2, 10, doneSplitTH2
addi $k1, $k1, 1
addi $t1, $t1, 1
j splitTH2
doneSplitTH2:
addi  $t2, $0, '\0'
sb $t2, 0($t4)

add $t4, $a2, $k0
lb  $s7, 0($t4)
addi $s7,$s7,-48
TH2case0:
bne $s7, 0, TH2case1
jal checkNTFunc
j TH2done
TH2case1:
bne $s7, 1, TH2case2
jal checkTokenRegFunc
j TH2done
TH2case2:
bne $s7, 2, TH2case3
jal checkHSNFunc
j TH2done
TH2case3:
```

```
bne $s7, 3, TH2done
jal checkIdentFunc
j TH2done
TH2done:
beq $s5, 1, checkToanHang2
j notFound
checkToanHang2:
beq $s7, 0, readToanHang3
li $v0, 4
la $a0, msg_ToanHang # opcode hop le
syscall
li $v0, 4
la $a0, token
syscall
li $v0, 4
la $a0, msg_HopLe
syscall
#<-- Ket Thuc xu ly toan hang 2 -->

#<-- bat dau xu ly toan hang 3 -->
readToanHang3: # tuong tu xu ly nhu toan hang 1,2
addi $k0, $k0, 1
addi $k1, $k1, 1
la $a1, token # luu cac ki tu doc duoc vao token
la $a0, input
li $t0, 0
li $t1, 0


splitTH3:

add $t3, $a0, $k1 # dich bit
```

```
add $t4, $a1, $t1
lb $t2, 0($t3) # doc tung ki tu cua input
sb $t2, 0($t4)
beq $t2, 44, doneSplitTH3 # gap ki tu ',' -> luu ki tu nay vao token de xu ly
beq $t2, 0, doneSplitTH3 # ket thuc chuoi input
beq $t2, 10, doneSplitTH3
addi $k1, $k1, 1
addi $t1, $t1, 1
j splitTH3
doneSplitTH3:
addi $t2, $0, '\0'
sb   $t2, 0($t4)

add  $t4, $a2, $k0
lb   $s7, 0($t4)
addi $s7, $s7,-48
TH3case0:
bne $s7, 0, TH3case1
jal checkNTFunc
j TH3done
TH3case1:
bne $s7, 1, TH3case2
jal checkTokenRegFunc
j TH3done
TH3case2:
bne $s7, 2, TH3case3
jal checkHSNFunc
j TH3done
TH3case3:
bne $s7, 3, TH3done
jal checkIdentFunc
j TH3done
```

```
TH3done:
beq $s5, 1, checkToanHang3
j notFound
checkToanHang3:
beq $s7, 0, end
li $v0, 4
la $a0, msg_ToanHang # opcode hop le
syscall
li $v0, 4
la $a0, token
syscall
li $v0, 4
la $a0, msg_HopLe
syscall
j end

#<-- Ket thuc xu ly toan hang 3

continue: # lap lai chuong trinh.
li $v0, 4
la $a0, continueMessage
syscall
li $v0, 5
syscall
add $t0, $v0, $zero
beq $t0, $zero, resetAll
j TheEnd
resetAll:
li $v0, 0
li $v1, 0
li $a0, 0
li $a1, 0
```

```
li $a2, 0
li $a3, 0
li $t0, 0
li $t1, 0
li $t2, 0
li $t3, 0
li $t4, 0
li $t5, 0
li $t6, 0
li $t7, 0
li $t8, 0
li $t9, 0
li $s0, 0
li $s1, 0
li $s2, 0
li $s3, 0
li $s4, 0
li $s5, 0
li $s6, 0
li $s7, 0
li $k0, 0
li $k1, 0
j readData
notFound:
li $v0, 4
la $a0, token
syscall

li $v0, 4
la $a0, NF
syscall
j error
```

```
error:
li $v0, 4
la $a0, errMessage
syscall
j continue
end:
li $v0, 4
la $a0, endMess
syscall
j continue
TheEnd:
```

Chạy và thử nghiệm

1 số trường hợp chạy thử