TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

KIẾN TRÚC MÁY TÍNH

****

FINAL TERM MINI PROJECT

GIÁO VIÊN HƯỚNG DẪN : ThS. LÊ BÁ VUI

SINH VIÊN THỰC HIỆN:

TRẦN MINH QUANG 20176858

PHẠM MINH HIẾU 20176758

Mục Lục

Nội dung **Trang**

Bài 4

Sinh viên thực hiện: Phạm Minh Hiếu

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>, <Thời gian>, <Cắt/Không cắt>
Trong đó

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

Hãy lập trình để CNC Marsbot có thể:

- Thực hiện cắt kim loại như đã mô tả
- Nội dung postscript được lưu trữ cố định bên trong mã nguồn
- Mã nguồn chứa 3 postscript và người dùng sử dụng 3 phím 0, 4, 8 trên bàn phím Key Matrix để chọn postscript nào sẽ được gia công.
- Một postscript chứa chữ DCE cần gia công. Hai script còn lại sinh viên tự đề xuất (tối thiểu 10 đường cắt)

Thuật toán:

1. Postscript đã thiết lập:

- Ứng với phím 0: DCE
- Ứng với phím 4: HIEU
- Ứng với phím 8: ELFSJ

2. Xử lý bài toán:

- Kiểm tra phím nào trên Key Matrix đã được bấm
- Đọc từng mảng cấu trúc trong Postscript

- Đọc giá trị góc

  + Đọc giá trị từng ký tự một trong Postscript cho đến khi gặp dấu phẩy bằng cách đọc giá trị ASCII của nó và trừ đi 48 (Trong ASCII số 0 có mã bằng 48)

  + Cứ một ký tự được đọc thêm thì nhân giá trị đã đọc với 10 và cộng với ký tự đã đọc để ra góc

- Đọc giá trị thời gian

  + Đọc từng ký tự một trong Postscript cho đến khi gặp dấu phẩy bằng cách đọc giá trị ASCII của nó và trừ đi 48 (Trong ASCII số 0 có mã bằng 48)

  + Cứ một ký tự được đọc thêm thì nhân giá trị đã đọc với 10 và cộng với ký tự đã đọc để ra thời gian

- Đọc giá trị Track
- Chuyển qua đọc mảng cấu trúc tiếp theo
- Cho Mars Bot chạy theo thông số ở trên

<u>Ý nghĩa của các thanh ghi</u>

$a1: Lưu địa chỉ của postscript được chọn

$t0: Lưu giá trị góc quay

$t1: Lưu giá trị thời gian

$t5: Lưu giá trị để so sánh

$a0: Lưu địa chỉ phím được bấm trong Key Matrix, đầu vào của chương trình con ROLATE

$t6: Lưu biến chạy i

$t7: Lưu vị trí đã đọc ở postscript hiện tại

<u>Ý nghĩa của các hàm:</u>

- Các hàm xử lý đầu vào:
  + An_0: Kiểm tra phím 0 trên Key matrix có được bấm hay không
  + An_4: Kiểm tra phím 4 trên Key matrix có được bấm hay không
  + An_8: Kiểm tra phím 8 trên Key matrix có được bấm hay không
  + COME_BACK: Tiếp tục chờ đến khi đầu vào là 0, 4, 8
- Các hàm để xử lý postscript:
  + START: Gọi chương trình con GO để Mars Bot bắt đầu chạy
  + READ_PSCRIPT: Khởi tạo giá trị ban đầu cho góc và thời gian bằng 0
  + READ_ROTATE: Đọc mảng cấu trúc và lưu giá trị góc
  + READ_TIME: Đọc mảng cấu trúc và lưu giá trị thời gian
  + READ_TRACK: Đọc mảng cấu trúc và lưu giá trị Track
  + CHECK_UNTRACK: Đổi trạng thái Mars Bot từ Track thành Untrack
  + INCREAMENT: Chạy Mars Bot với thông số ở trên và chuyển qua mảng cấu trúc mới

- Các chương trình con để Mars Bot di chuyển: (Sử dụng các chương trình con trong bài giảng)
  + GO: Cho Mars Bot chạy
  + STOP: Dừng Mars Bot
  + TRACK: Tạo vết cắt
  + UNTRACK: Không tạo vết cắt
  + ROTATE: Cho Mars Bot chạy theo góc

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;300,5
00,1;290,500,1;285,490,1;90,8000,0;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;14
0,500,1;130,500,1;120,500,1;110,500,1;100,500,1;90,900,1;90,5000,0;270,3000,1;0,5800
,1;90,3000,1;180,2900,0;270,2995,1;90,3500,0;"
# postscript-HIEU => numpad 4
pscript2: .asciiz
"90,2000,0;180,3000,0;180,5790,1;0,2900,0;90,3000,1;0,2900,0;180,5790,1;90,1500,0;0,
5790,1;90,4500,0;270,3000,1;180,5790,1;90,3000,1;0,2900,0;270,2995,1;90,4500,0;0,29
00,0;180,5000,1;170,200,1;160,200,1;150,200,1;140,200,1;130,200,1;90,2300,1;50,200,1
;40,200,1;30,200,1;20,200,1;10,200,1;0,5000,1;90,1000,0;"
# postscript-ELFSJ => numpad 8
pscript3: .asciiz
"180,3000,0;90,5000,0;270,3000,1;180,5790,1;90,3000,1;0,2900,0;270,2990,1;90,4500,0
;0,2900,0;180,5790,1;90,3000,1;90,1500,0;0,5790,1;90,3000,1;180,2900,0;270,2995,1;9
0,9000,0;0,2850,0;90,100,0;300,250,1;280,350,1;270,350,1;260,300,1;250,300,1;240,300
,1;220,300,1;210,500,1;200,500,1;180,300,1;160,500,1;140,500,1;120,500,1;100,500,1;9
0,300,1;110,300,1;120,300,1;140,500,1;160,500,1;180,500,1;190,300,1;200,300,1;210,30
0,1;220,300,1;230,300,1;240,300,1;250,300,1;270,500,1;280,500,1;90,4000,0;0,5790,0;9
0,3500,1;270,1700,0;180,4700,1;190,200,1;200,200,1;210,200,1;220,200,1;230,200,1;24
0,200,1;250,200,1;270,300,1;280,300,1;180,1000,0"

.text
# Nhan dau vao tu key matrix
        li $t3, IN_ADRESS_HEXA_KEYBOARD
        li $t4, OUT_ADRESS_HEXA_KEYBOARD
An_0:
        li $t5, 0x01          # row-1 of key matrix
        sb $t5, 0($t3)              # gan gia tri cua $t5 cho $t3
        lb $a0, 0($t4)              # lay dia chi thanh ghi $t4 load vao $a0
        bne $a0, 0x11, An_4     # Neu gia tri cua $a0 khac phim 0 thi chuyen qua kiem
```

```
tra phim 4
        la $a1, pscript1      # Neu la phim 0 thi chay postscript cua phim 0
        j START
An_4:
        li $t5, 0x02          # row-2 of key matrix
        sb $t5, 0($t3)              # gan gia tri cua $t5 cho $t3
        lb $a0, 0($t4)              # lay dia chi thanh ghi $t4 load vao $a0
        bne $a0, 0x12, An_8       # Neu gia tri cua $a0 khac phim 4 thi chuyen qua kiem
tra phim 8
        la $a1, pscript2      # Neu la phim 4 thi chay postscript cua phim 4
        j START
An_8:
        li $t5, 0X04          # row-3 of key matrix
        sb $t5, 0($t3)              # gan gia tri cua $t5 cho $t3
        lb $a0, 0($t4)              # lay dia chi thanh ghi $t4 load vao $a0
        bne $a0, 0x14, COME_BACK # khi cac so 0,4,8 khong duoc chon -> quay lai doc
tiep
        la $a1, pscript3
        j START                     # Neu la phim 8 thi chay postscript cua phim 8
COME_BACK:
        j An_0                      # khi cac so 0,4,8 khong duoc chon -> quay lai doc tiep

# xu li mars bot
START:
        jal GO
READ_PSCRIPT:
        addi $t0, $zero, 0   # luu gia tri rotate
        addi $t1, $zero, 0   # luu gia tri time

READ_ROTATE:
        add $t7, $a1, $t6    # dich bit, $t6 la i
        lb $t5, 0($t7)       # doc cac ki tu cua script
        beq $t5, 0, END      # Neu $t5 = NULL ket thuc pscript
        beq $t5, 44, READ_TIME        # gap ki tu ',' thi chuyen qua doc thoi gian
        mul $t0, $t0, 10     # Dich chu so doc duoc sang trai 1 hang
        addi $t5, $t5, -48   # So 0 co thu tu 48 trong bang ascii.
        add $t0, $t0, $t5    # cong cac chu so lai voi nhau de ra goc
        addi $t6, $t6, 1     # tang so bit can dich chuyen len 1, i =i+1
        j READ_ROTATE                 # quay lai doc tiep den khi gap dau ','
```

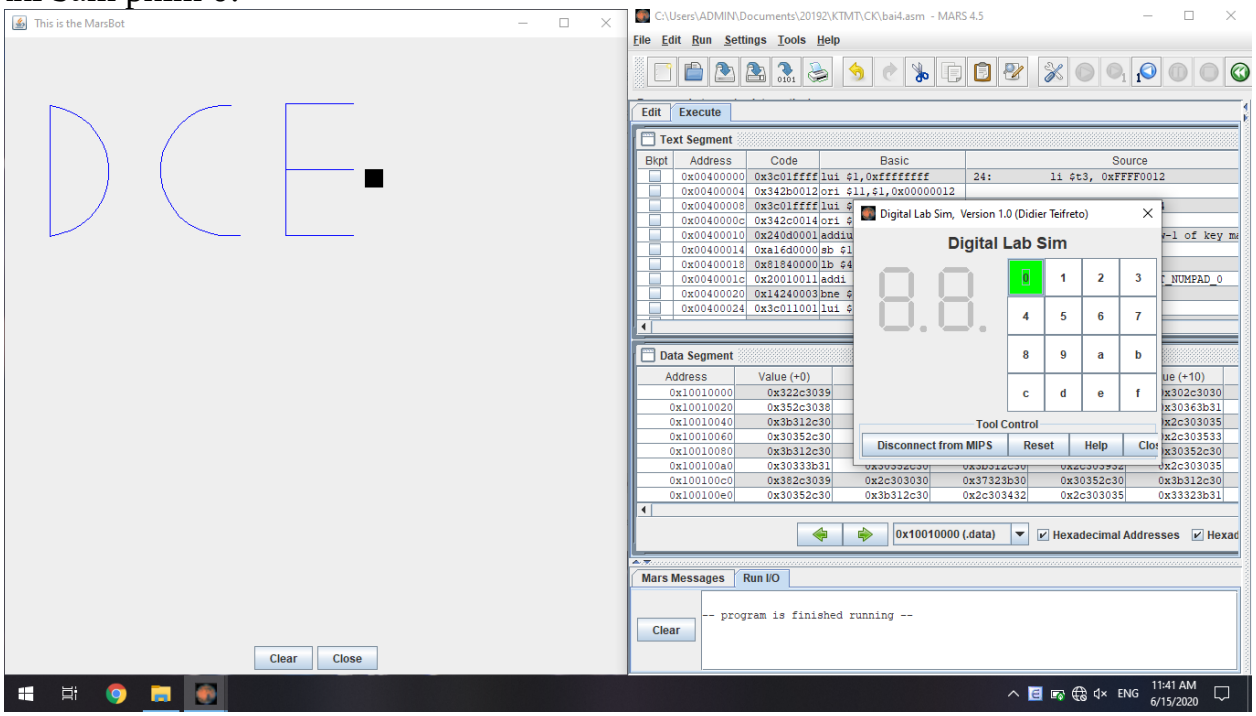
```
READ_TIME:                  # doc thoi gian chuyen dong.
        add $a0, $t0, $zero
        jal ROTATE
        addi $t6, $t6, 1
        add $t7, $a1, $t6    # dich bit
        lb $t5, 0($t7)
        beq $t5, 44, READ_TRACK #gap ki tu ',' thi chuyen qua kiem tra Track
        mul $t1, $t1, 10
        addi $t5, $t5, -48
        add $t1, $t1, $t5    # cong cac chu so lai voi nhau de ra thoi gian
        j READ_TIME          # quay lai doc tiep den khi gap dau ','

READ_TRACK:
        addi $v0,$zero,32  # Giu cho Mars bot chay
        add $a0, $zero, $t1# Gia tri thoi gian mars bot chay =$t1
        addi $t6, $t6, 1
        add $t7, $a1, $t6
        lb $t5, 0($t7)          # doc track
        addi $t5, $t5, -48
        beq $t5, $zero, CHECK_UNTRACK # 1=track | 0=untrack
        jal UNTRACK
        jal TRACK
        j INCREAMENT

CHECK_UNTRACK:
        jal UNTRACK

INCREAMENT:
        syscall                 # chay mars bot voi thong so o tren
        addi $t6, $t6, 2      # bo qua dau ';'
        j READ_PSCRIPT

# cac chuong trinh con cua MarsBot lay tu bai giang
#-----------------------------------------------------------
# GO procedure, to start running
# param[in]    none
#-----------------------------------------------------------
GO:
        li $at, MOVING
        addi $k0, $zero,1
```

```
        sb $k0, 0($at)
        jr $ra
#-----------------------------------------------------------
# STOP procedure, to stop running
# param[in]    none
#-----------------------------------------------------------
STOP:
        li $at, MOVING
        sb $zero, 0($at)
        jr $ra
#-----------------------------------------------------------
# TRACK procedure, to start drawing line
# param[in]    none
#-----------------------------------------------------------
TRACK:
        li $at, LEAVETRACK
        addi $k0, $zero,1
        sb $k0, 0($at)
        jr $ra
#-----------------------------------------------------------
# UNTRACK procedure, to stop drawing line
# param[in]    none
#-----------------------------------------------------------
UNTRACK:
        li $at, LEAVETRACK
        sb $zero, 0($at)
        jr $ra
#-----------------------------------------------------------
# ROTATE procedure, to rotate the robot
# param[in]    $a0, An angle between 0 and 359
#                0 : North (up)
#                90: East  (right)
#               180: South (down)
#               270: West  (left)
#-----------------------------------------------------------
ROTATE:
        li $at, HEADING
        sw $a0, 0($at)
        jr $ra
```

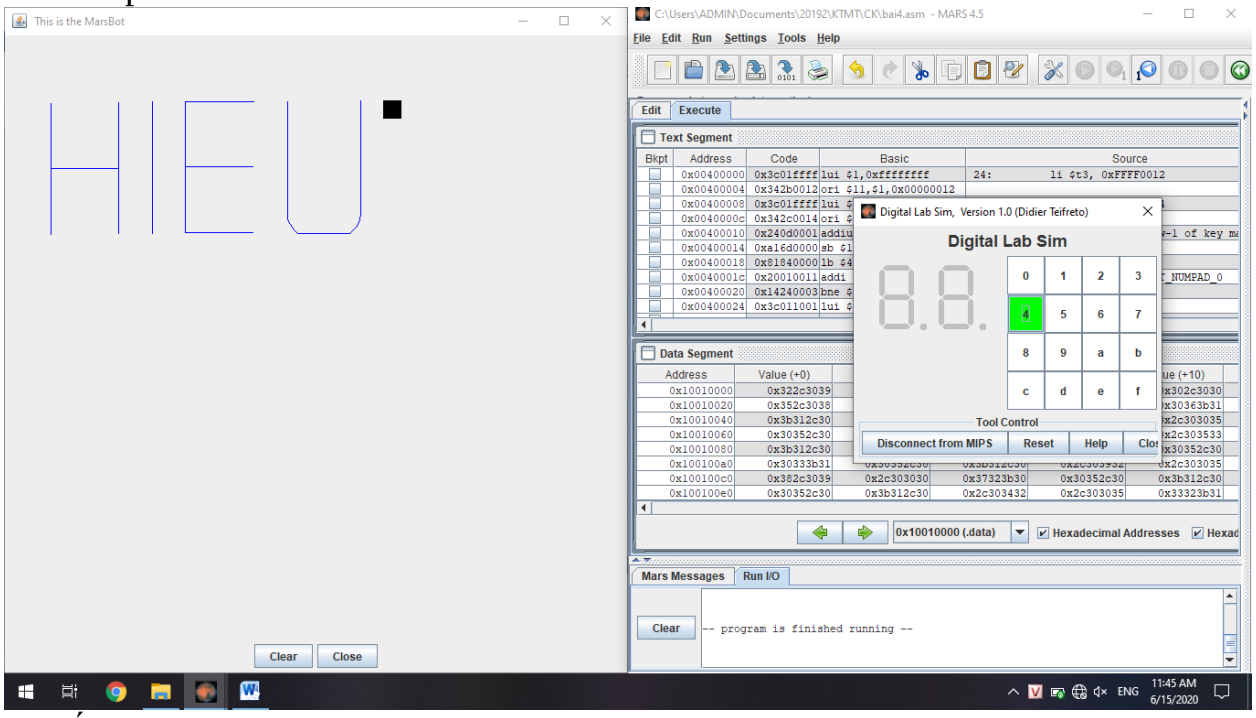
```
END:
        jal STOP
        li $v0, 10
        syscall
```

Kết quả:

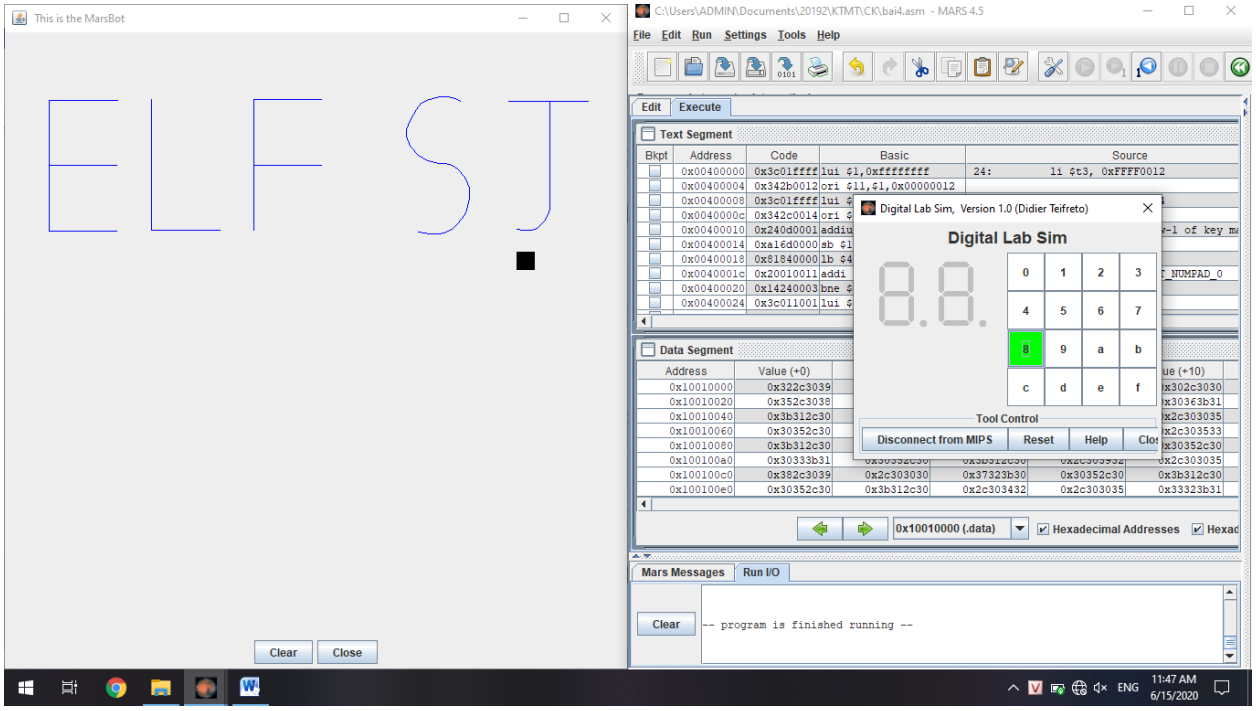
+ Khi bấm phím 0:



+ Khi bấm phím 4



+ Khi bấm phím 8


This is the MarsBot
Clear
Close
C:\Users\ADMIN\Documents\20192\KTMT\CK\bai4.asm - MARS 4.5
File Edit Run Settings Tools Help
Edit Execute
Text Segment
Bkpt Address Code Basic Source
Digital Lab Sim, Version 1.0 (Didier Teifreto)
Digital Lab Sim
Tool Control
Disconnect from MIPS
Reset
Help
Data Segment
Address Value (+0)
0x10010000 (.data)
Hexadecimal Addresses
Mars Messages
Run I/O
Clear
-- program is finished running --
ENG
11:47 AM
6/15/2020

BÀI 7

SINH VIÊN THỰC HIỆN: TRẦN MINH QUANG

Đề bài:

Chương trình kiểm tra cú pháp lệnh MIPS

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.
- Cho biết lệnh hợp ngữ đó cần bao nhiêu chu kì thì mới thực hiện xong.

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.

Thuật toán:

- Khởi tạo một mảng chứa khuôn dạng của các lệnh với tên lệnh, kiểu của các toán hàng và số chu kì thực hiện.

```
library: .asciiz "or***1111;xor**1111;lui**1201;jr***1001;jal**3002;addi*1121;add**1111;addiu1121;sub**1111;ori**1121;and**1111;
```

Trong đó, mỗi khuôn dạng của các lệnh sẽ được lưu 10 ký tự (5 ký tự đầu tiên sẽ lưu tên lệnh, nếu không đủ 5 thì sẽ bù vào bằng cách thêm các dấu '*', ký tự thứ 6 tới ký tự thứ 9 lần lượt lưu kiểu của toán hạng 1, toán hạng 2, toán hạng 3 và chu kì của lênh. Ký tự thứ 10 lưu trữ dấu ';')

- Khởi tạo một mảng chứa các ký tự của các nhãn trong chương trình, nếu không thuộc các ký tự trong mảng này thì nhãn không hợp lệ.

```
labelGroup: .asciiz "1234567890qwertyuiopasdfghjklmnbvcxzQWERTYUIOPASDFGHJKLZXCVBNM_"
```

- Khởi tạo một mảng để lưu trữ định dạng của các thanh ghi.

```
registersLib: .asciiz "$zero $at   $v0   $v1   $a0   $a1   $a2   $a3   $t0   $t1   $t2   $t3   $t4   $t5   $t6   $t7   $s0   $s1
```

- Sau khi nhận lệnh nhập từ bàn phím, sẽ tiến hành so sánh opcode của lệnh vừa nhập với các opcode trong thư viện lưu trữ khuôn dạng của các lệnh. Nếu tìm thấy sẽ tiếp tục chuyển đến so sánh các toán hạng thứ 1, toán hạng thứ 2, toán hạng thứ 3, ….
- So sánh các toán hạng thứ 1, thứ 2, thứ 3 sẽ lần lượt kiểm tra xem toán hạng đang xét là thanh ghi, là số, là nhãn trong lệnh nhảy, hay là trống.
- Nếu thỏa mãn toàn bộ khuôn dạng lệnh sẽ thông báo cho người dùng biết lệnh nhập vào là hợp lệ.
- Nếu không thỏa mãn ở bất kỳ thành phần nào sẽ dừng tại đấy, báo cho người dùng biết lệnh không hợp lệ.

Chức năng của các nhãn (label) trong chương trình:

- readData: In ra thông báo nhập lệnh nhập vào từ bàn phím.
- readOpcode: Đọc lệnh nhập vào từ bàn phím, lưu trữ vào mảng.
- xuLyOpcode: Thiết lập chỉ số trỏ đến phần từ đầu tiên của mảng lưu trữ khuôn dạng lệnh.
- compare: So sánh opcode của lệnh vừa nhập vào với opcode trong thư viện, nếu gặp dấu ‘*’ mà opcode chưa thỏa mãn sẽ chuyển qua so sánh các opcode tiếp theo.
- check: kiểm tra xem sau opcode có phải dấu cách (space) không, nếu không thì kết thúc.
- checkContinue: nếu opcode hợp lệ, in ra thông báo và tiếp tục đọc toán hạng 1
- check2: kiểm tra xem ký tự tiếp theo có phải là ‘\n’ hay không.
- readToanHang1: xác định kiểu của toán hạng thứ 1 trong library, sau đó đưa ra phương án xử lý (là thanh ghi / số / nhãn/ không có toán hạng)
- checkReg: bắt đầu xử lý kiểu thanh ghi.
- readToken: đọc toán hạng từ bàn phím và tiến hành lưu trữ. Nếu gặp dấu ‘,’ hoặc ký tự kết thúc thì sẽ dừng việc đọc.
- readTokenDone: kết thúc việc đọc toán hạng.
- compareToken: so sánh toán hạng với tập thanh ghi trong thư viện đã được khai báo từ đầu. Nếu như gặp dấu ‘,’ thì kết thúc việc so sánh.
- checkTokenLength: kiểm tra xem nếu trong toán hạng đang nhận vào có kí tự ‘,’ hoặc ‘/n’ thì dừng việc so sánh.
- CompareE: in ra thông báo opcode và toán hạng hợp lệ, đồng thời tiếp tục xử lý các toán hạng sau nó.

- CompareNE: toán hạng không phù hợp, kết thúc chương trình.
- checkImm: Kiểm tra toán hạng có phải hằng số nguyên hay không?
- readNumber: đọc số từ bàn phím và tiến hành lưu trữ. Nếu gặp dấu ‘,’ hoặc ký tự kết thúc thì sẽ dừng việc đọc.
- readNumberDone: Kết thúc việc đọc số.
- compareNumber: loại bỏ các dấu ‘-’, ‘,’, ký tự xuống dòng để so sánh trong khoảng từ 0 tới 9.
- checkLabel: kiểm tra các nhãn trong chương trình.
- ReadIdent: đọc định danh từ bàn phím và tiến hành lưu trữ, nếu gặp dấu ‘,’ hoắc ký tự kết thúc thì dừng việc đọc.
- readIdentDone:kết thúc việc đọc định danh và bắt đầu so sánh.
- compareIdent: So sánh từng ký tự của định danh với các ký tự trong thư viện được khai báo trước để kiểm tra tính hợp lệ.
- jumpIdent:
- compareIndentE: nếu tìm thấy toán hạng đúng là thanh ghi, thì in ra thông báo hợp lệ, tiếp tục đọc các toán hạng sau. ($v1 ghi số toán hạng đã đọc được)
- compareIndentNE: Toán hạng không phù hợp, kết thúc chương trình.
- checkNT: Kiểm tra khi vị trí của toán hạng không có gì.
- readToanHang2: tiến hành đọc toán hạng 2 và tiến hành kiểm tra kiểu của toán hạng 2.
- readToanHang3: tiến hành đọc toán hạng 3 và tiến hành kiểm tra kiểu của toán hạng 3.
- readChuKy: Chuyển đến vị trí của chữ ký trong chương trình và in ra.
- continue: In ra thông báo, hỏi người nhập có muốn tiếp tục chưng trình không. Nếu có sẽ đưa toàn bộ các thanh ghi về trạng thái ban đầu để tiếp tục đọc lệnh từ bàn phím. Nếu không thì sẽ kết thúc chương trình.
- resetAll: đưa các thanh ghi về trạng thái ban đầu.
- notFound: Hiện ra thông báo không tìm thấy khuôn dạng lệnh.
- error: Hiện ra thông báo lỗi.
- end: Kết thúc.

Mã Nguồn:

```
#---------------------------------------------------------------
# @brief      Nhap tu ban phim mot cau lenh Mips, kiem tra xem cau lenh ay
co dung hay khong
# @param[in]       User input Mips'command
# @param[out]      Correct or Not Conrrect Mips Symtax
#---------------------------------------------------------------
```

```
.data
	input: .asciiz "Input a Mips command: "
	continueMessage: .asciiz "Do you want to exit? (0. No / 1. Yes): "
	errMessage: .asciiz "\nSymtax error!\n"
	NF: .asciiz "Cann't found this command form!\n"
	endMess: .asciiz "\nThis command is conrrect\n"
	hopLe1: .asciiz "Opcode: "
	hopLe11: .asciiz "Toan Hang: "
	hopLe2: .asciiz "Conrect\n"
	khongHopLe: .asciiz "Inconrect.\n"
	chuKyMess: .asciiz "Command's cycle: "
	command: .space 100
	opcode: .space 10
	token: .space 15
	number: .space 15
	ident: .space 15
	# Register = 1; Label = 2; Imm = 3; null = 0
	library: .asciiz
"or***1111;xor**1111;lui**1201;jr***1001;jal**3002;addi*1121;add**111
1;addiu1121;sub**1111;ori**1121;and**1111;andi*1131;beq**1132;bne**1
132;j****3002;nop**0001;bgez*1202;bgtz*1202;xori*1131;sll**1131;slt**1
111"
	labelGroup: .asciiz
"1234567890qwertyuiopasdfghjklmnbvcxzQWERTYUIOPASDFGHJKLZX
CVBNM_"
	registersLib: .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"

.text
#readData
readData:
	li $v0, 4		# print string service
	la $a0, input		# print InputMessage
	syscall
	li $v0, 8		# read string service
	la $a0, command	# store input string in Command
	li $a1, 100		# max buffer
	syscall
# end readData
main:
	li $t2, 0		# $t2 = i = 0
readOpcode:
	la $a1, opcode		# luu cac ki tu doc duoc vao opcode
```

```
        add $t3, $a0, $t2    # truy cap vao dia chi cua Command o ki tu thu t1
        add $t4, $a1, $t2    # truy cap vao dia chi cua Opcode o ki tu thu t1
        lb $t1, 0($t3)              # lay ki tu Command[i]
        sb $t1, 0($t4)              # luu ki tu vao Opcode[i]
        beq $t1, 32, done    # gap ki tu ' ' -> luu ki tu nay vao opcode de xu ly
        beq $t1, 0, done     # ket thuc chuoi command
        addi $t2, $t2, 1     # i = i+1
        j readOpcode

#<--xu ly opcode-->
done:
        li $t7,-10           #Moi lenh cach nhau 10 byte nen bat dau tu -10
        la $a2, library             #Dia chi cua library
xuLyOpcode:
        li $t1, 0 # i
        li $t2, 0 # j
        addi $t7,$t7,10      # buoc nhay = 10 de den vi tri opcode dau tien trong
library
        add $t1,$t1,$t7      # lenh hien tai dang xet

        compare:
        add $t3, $a2, $t1    # t3 tro thanh con tro cua library
                        //Vi tri cua con tro hien tai trong library
        lb $s0, 0($t3)              # ki tu hien tai
        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?.                   //'*' = 42
        add $t4, $a1, $t2    # vi tri ki tu dang xet cua lenh duoc nhap
        lb $s1, 0($t4)
        bne $s0,$s1,xuLyOpcode          # so sanh 2 ki tu. dung thi so sanh tiep,
sai thi nhay den phan tu chua khuon danh lenh tiep theo.
        addi $t1,$t1,1              # i+=1
        addi $t2,$t2,1              # j+=1
        j compare
        # end compare
        check:
        add $t4, $a1, $t2
        lb $s1, 0($t4)
        bne $s1, 32, check2         # neu ki tu tiep theo khong phai 'space' =>
lenh khong hop le. chi co doan dau giong.

        checkContinue:
        add $t9,$t9,$t2      # t9 = luu vi tri de xu ly token trong command
        li $v0, 4
```

```
        la $a0, hopLe1          # Opcode:
        syscall
        la $a0, opcode          # opcode
        syscall
        la $a0, hopLe2          # correct
        syscall
        j readToanHang1

        check2:             # neu ki tu tiep theo khong phai '\n' => lenh khong
hop le. chi co doan dau giong.
        bne $s1, 10, notFound
                # ASCII \n = 10
        j checkContinue

# <!--ket thuc xu ly opcode -->

#<--xu li toan hang-->

#-----------------------------------------------------------------
# @brief      Doc cac toan hang va kiem tra kieu cua cac toan hang do
#-----------------------------------------------------------------
readToanHang1:
        # xac dinh kieu toan hang trong library
        # t7 dang chua vi tri khuon dang lenh trong library
        li $t1, 0
        addi $t7, $t7, 5      # chuyen den vi tri toan hang 1 trong library
        add $t1, $a2, $t7     # a2 chua dia chi library
        lb $s0, 0($t1)
        addi $s0,$s0,-48      # chuyen tu char -> int
        li $t8, 0             # khong co toan hang = 0
        beq $s0, $t8, checkNO
        li $t8, 1             # thanh ghi
        beq $s0, $t8, checkReg
        li $t8, 2             # hang so nguyen
        beq $s0, $t8, checkImm
        li $t8, 3             # dinh danh
        beq $s0, $t8, checkLabel
        j end

readToanHang2:
        # xac dinh kieu toan hang trong library
        # t7 dang chua vi tri khuon dang lenh trong library
        li $t1, 0
        la $a2, library
```

```
        addi $t7, $t7, 1      # chuyen den vi tri toan hang 2 trong library
        add $t1, $a2, $t7     # a2 chua dia chi library
        lb $s0, 0($t1)
        addi $s0,$s0,-48      # chuyen tu char -> int
        li $t8, 0             # khong co toan hang = 0
        beq $s0, $t8, checkNO
        li $t8, 1             # thanh ghi = 1
        beq $s0, $t8, checkReg
        li $t8, 2             # hang so nguyen = 2
        beq $s0, $t8, checkImm
        li $t8, 3             # dinh danh = 3
        beq $s0, $t8, checkLabel
        j end

readToanHang3:
        # xac dinh kieu toan hang trong library
        # t7 dang chua vi tri khuon dang lenh trong library
        li $t1, 0
        la $a2, library
        addi $t7, $t7, 1      # chuyen den vi tri toan hang 3 trong library
        add $t1, $a2, $t7     # a2 chua dia chi library
        lb $s0, 0($t1)
        addi $s0,$s0,-48      # chuyen tu char -> int
        li $t8, 0             # khong co toan hang = 0
        beq $s0, $t8, checkNO
        li $t8, 1             # thanh ghi = 1
        beq $s0, $t8, checkReg
        li $t8, 2             # hang so nguyen = 2
        beq $s0, $t8, checkImm
        li $t8, 3             # dinh danh = 3
        beq $s0, $t8, checkLabel
        j end

readChuKy:
        # xac dinh kieu toan hang trong library
        # t7 dang chua vi tri khuon dang lenh trong library
        li $t1, 0
        la $a2, library
        addi $t7, $t7, 1      # chuyen den vi tri chu ky  trong library
        add $t1, $a2, $t7     # a2 chua dia chi library
        lb $s0, 0($t1)
        addi $s0,$s0,-48      # chuyen tu char -> int
        li $v0, 4
        la $a0, chuKyMess
```

```
        syscall
        li $v0,1
        li $a0,0
        add $a0,$s0,$zero
        syscall
        j end

#-----------------------------------------------------------------
# @brief      Kiem tra neu toan hang la kieu thanh ghi
#-----------------------------------------------------------------
checkReg:
        la $a0, command          # load address of command to $a0
        la $a1, token            # load address of Token to $a0 => store
register token
        li $t1, 0
        li $t2, -1
        addi $t1, $t9, 0
        readToken:
                addi $t1, $t1, 1            # i
                addi $t2, $t2, 1            # j
                add $t3, $a0, $t1           # command[i]
                add $t4, $a1, $t2           # token[i]
                lb $s0, 0($t3)
                add $t9, $zero, $t1                 # vi tri toan hang tiep theo trong
command
                beq $s0, 44, readTokenDone  # gap dau ','
                beq $s0, 0, readTokenDone   # gap ki tu ket thuc
                sb $s0, 0($t4)
                j readToken

        readTokenDone:
                sb $s0, 0($t4)                      # luu them ',' vao de so sanh
                li $t1, -1                  # i
                li $t2, -1                  # j
                li $t4, 0
                li $t5, 0
                add $t2, $t2, $k1
                la $a1, token
                la $a2, registersLib
                j compareToken

compareToken:
        addi $t1,$t1,1
        addi $t2,$t2,1
```

```
        add $t4, $a1, $t1
        lb $s0, 0($t4)
        beq $s0, 0, notFound                    # neu o vi tri toan trong
command la Null thi bao loi

        add $t5, $a2, $t2
        lb $s1, 0($t5)
        beq $s1, 0, notFound
        beq $s1, 32, checkLengthToken           # neu gap dau space thi se nhay
den checkLengthToken
        bne $s0,$s1, jump
        j compareToken

        checkLengthToken:
                beq $s0, 10, compareE           #\n
                beq $s0, 44, compareE           #','
                j compareNE
        jump:
                addi $k1,$k1,6
                j readTokenDone
        compareE:
                la $a0, hopLe11             # opcode hop le
                syscall
                li $v0, 4
                la $a0, token
                syscall
                li $v0, 4
                la $a0, hopLe2
                syscall
                addi $v1, $v1, 1            # dem so toan hang da doc.
                li $k1, 0                   # reset buoc nhay
                beq $v1, 1, readToanHang2
                beq $v1, 2, readToanHang3
                beq $v1, 3, readChuKy
                j end
        compareNE:
                la $a0, hopLe11                 # Toan Hang:
                syscall
                li $v0, 4
                la $a0, token
                syscall
                li $v0, 4
                la $a0,khongHopLe
                syscall
```

```
		j notFound

#---------------------------------------------------------------
# @brief	Kiem tra neu kieu cua toan hang la mot hang so nguyen
#---------------------------------------------------------------
checkImm:
	la $a0, command
	la $a1, number				# luu day chu so vao number de
so sanh tung chu so co thuoc vao numberGroup hay khong.
	li $t1, 0
	li $t2, -1
	addi $t1, $t9, 0
	readNumber:
		addi $t1, $t1, 1		# i
		addi $t2, $t2, 1		# j
		add $t3, $a0, $t1
		add $t4, $a1, $t2
		lb $s0, 0($t3)
		add $t9, $zero, $t1		# vi tri toan hang tiep theo trong
command
		beq $s0, 44, readNumberDone	# gap dau ','
		beq $s0, 0, readNumberDone	# gap ki tu ket thuc
		sb $s0, 0($t4)
		j readNumber
	readNumberDone:
		sb $s0, 0($t4)			# luu them ',' vao de compare
		li $t1, -1			# i
		li $t4, 0
		la $a1, number
		j compareNumber
compareNumber:
	addi $t1, $t1, 1
	add $t4, $a1, $t1
	lb $s0, 0($t4)
	beq $s0, 0, notFound
	beq $s0, 45, compareNumber		# bo dau '-'
	beq $s0, 10, compareNumE
	beq $s0, 44, compareNumE
	li $t2, 48				#48 ASCII = 0
	li $t3, 57				#57 ASCII = 9
	slt $t5, $s0, $t2			#compare with 0
	bne $t5, $zero, compareNumNE
	slt $t5, $t3, $s0			#compare with 9
	bne $t5, $zero, compareNumNE
```

```
        j compareNumber

        compareNumE:
                la $a0, hopLe11
                syscall
                li $v0, 4
                la $a0, number
                syscall
                li $v0, 4
                la $a0, hopLe2
                syscall
                addi $v1, $v1, 1            # dem so toan hang da doc.
                li $k1, 0                   # reset buoc nhay
                beq $v1, 1, readToanHang2
                beq $v1, 2, readToanHang3
                beq $v1, 3, readChuKy
                j end
        compareNumNE:
                la $a0, hopLe11                 # Toan Hang:
                syscall
                li $v0, 4
                la $a0, number
                syscall
                li $v0, 4
                la $a0,khongHopLe
                syscall
                j notFound

#---------------------------------------------------------------
# @brief      Kiem tra neu kieu cua toan hang la mot nhan
#---------------------------------------------------------------
checkLabel:
        la $a0, command
        la $a1, ident                           # luu ten thanh ghi vao indent de
so sanh
        li $t1, 0
        li $t2, -1
        addi $t1, $t9, 0
        readIndent:
                addi $t1, $t1, 1            # i
                addi $t2, $t2, 1            # j
                add $t3, $a0, $t1
                add $t4, $a1, $t2
                lb $s0, 0($t3)
```

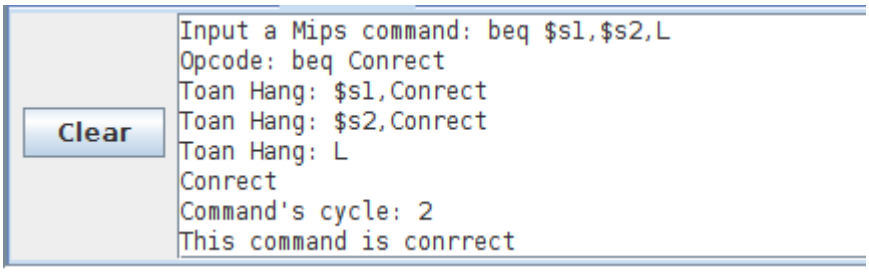
```
            add $t9, $zero, $t1             # vi tri toan hang tiep theo trong
command
            beq $s0, 44, readIdentDone      # gap dau ','
            beq $s0, 0, readIdentDone       # gap ki tu ket thuc
            sb $s0, 0($t4)
            j readIndent
        readIdentDone:
            sb $s0, 0($t4)                  # luu them ',' vao de compare
            loopj:
            li $t1, -1                  # i
            li $t2, -1                  # j
            li $t4, 0
            li $t5, 0
            add $t1, $t1, $k1
            la $a1, ident
            la $a2, labelGroup
            j compareIdent
compareIdent:
        addi $t1,$t1,1
        add $t4, $a1, $t1
        lb $s0, 0($t4)
        beq $s0, 0, notFound
        beq $s0, 10, compareIdentE
        beq $s0, 44, compareIdentE
        loop:
        addi $t2,$t2,1
        add $t5, $a2, $t2
        lb $s1, 0($t5)
        beq $s1, 0, compareIdentNE
        beq $s0, $s1, jumpIdent         # so sanh ki tu tiep theo trong ident
        j loop                          # tiep tuc so sanh ki tu tiep theo trong
labelGroup

        jumpIdent:
            addi $k1,$k1,1
            j loopj
        compareIdentE:
            la $a0, hopLe11             # opcode hop le
            syscall
            li $v0, 4
            la $a0, ident
            syscall
            li $v0, 4
            la $a0, hopLe2
```

```
                syscall
                addi $v1, $v1, 1            # dem so toan hang da doc.
                li $k1, 0                   # reset buoc nhay
                beq $v1, 1, readToanHang2
                beq $v1, 2, readToanHang3
                beq $v1, 3, readChuKy
                j end
        compareIdentNE:
                la $a0, hopLe11                 # Toan Hang:
                syscall
                li $v0, 4
                la $a0, ident
                syscall
                li $v0, 4
                la $a0,khongHopLe
                syscall
                j notFound
#---------------------------------------------------------------
# @brief      Kiem tra neu nguoi dung bo trong vi tri cua toan hang (NULL)
#---------------------------------------------------------------
checkNO:
        la $a0, command
        li $t1, 0
        li $t2, 0
        addi $t1, $t9, 0
        add $t2, $a0, $t1
        lb $s0, 0($t2)
        addi $v1, $v1, 1                # dem so toan hang da doc.
        li $k1, 0                       # reset buoc nhay
        beq $v1, 1, readToanHang2
        beq $v1, 2, readToanHang3
        beq $v1, 3, readChuKy

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
```

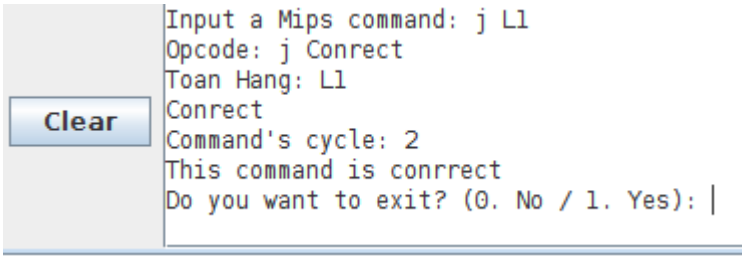
```
        li $v1, 0
        li $a0, 0
        li $a1, 0
        li $a2, 0
        li $a3, 0
        li $t0, 0
        li $t1, 0
        li $t2, 0
        li $t3, 0
        li $t4, 0
        li $t5, 0
        li $t6, 0
        li $t7, 0
        li $t8, 0
        li $t9, 0
        li $s0, 0
        li $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, NF
        syscall
        j continue
error:
        li $v0, 4
        la $a0, errMessage
        syscall
        j continue
end:
        li $v0, 4
        la $a0, endMess
        syscall
        j continue
TheEnd:
```

Kết Quả Hiển Thị:

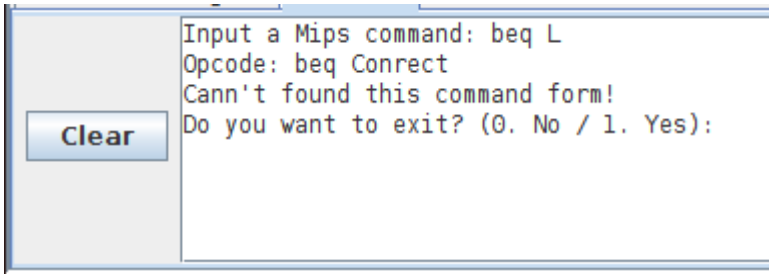
Câu lệnh beq:



Câu lệnh jump:



Trường hợp câu lệnh không chính xác:



-Hết-