BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC BÁCH KHOA HÀ NỘI

Trường Công nghệ thông tin và truyền thông

----- o0o -----

BÁO CÁO BÀI TẬP LỚN

Môn Học: Thực hành Kiến trúc Máy Tính

Mã học phần: IT3280

Sinh viên thực hiện:

Nguyễn Tuấn Nam	MSSV:	20194629
Trần Quang Thưởng	MSSV:	20194684

Giảng viên hướng dẫn: Thầy Lê Bá Vui

A. ĐỀ 10

1. Đề bài : Máy tính bỏ túi

Sử dụng 2 ngoại vi là bàn phím keypad và led 7 thanh để xây dựng một máy tính bỏ túi đơn giản. Hỗ trợ các phép toán +, -, *, /,% với các toán hạng là số nguyên. Do trên bàn phím không có các phím trên nên sẽ dùng các phím:

- Bấm phím a để nhập phép tính +
- Bấm phím b để nhập phép tính –
- Bấm phím c để nhập phép tính *
- Bấm phím d để nhập phép tính /
- Bấm phím e để nhập phép tính %
- Bấm phím f để nhập phép =

Yêu cầu cụ thể như sau:

- Khi nhấn các phím số, hiển thị lên LED, do chỉ có 2 LED nên chỉ hiển thị 2 số cuối cùng. Ví dụ khi nhấn phím 1 → hiển thị 01. Khi nhấn thêm phím 2 → hiển thị 12. Khi nhấn thêm phím 3 → hiển thị 23.
- Sau khi nhập số, sẽ nhập phép tính + - * / %
- Sau khi nhấn phím f (dấu =) , tính toán và hiển thị kết quả lên LED.
- Có thể thực hiện các phép tính liên tiếp (tham khảo ứng dụng Calculator trên hệ điều hành Windows)

Chú ý: Do bài toán sẽ có rất nhiều trường hợp xảy ra, yêu cầu cơ bản là thực hiện được phép tính và hiển thị lên LED. Các yêu cầu về bắt lỗi, các trường hợp tràn số, ... là tùy chọn.

2. Phân tích cách làm

- Khởi tạo chương trình chính là 1 vòng lặp vô hạn . Khi có một Exception xảy ra, MIPS sẽ luôn nhảy tới địa chỉ cố định 0x80000180 để thực hiện chương trình con phục vụ ngắt. Để viết chương trình con phục vụ ngắt, ta sẽ sử dụng chỉ thị .ktext để viết code ở địa chỉ 0x80000180 nói trên.
- Trong .ktext sẽ thực hiện các công việc:
  - Quét bàn phim trên Digital Lab Sim và lấy địa chỉ vị trí của phím đã được nhập.
  - Xác định cụ thể phim đó thuộc loại gì:

  Nếu phím đó là số hạng thứ 1:
  + Update lại số thứ 1 hiện tại (thông qua update)
  + Hiển thị ra LED

  Nếu phím đó là các toán tử +,-,*,/,% :
  + Thực hiện lưu lại số hạng thứ 1 (thông qua hàm savefirstnum)
  + Reset lại thanh LED bên trái về giá trị 0 (thông qua reset_left_7seg)

  Nếu phím đó là số hạng thứ 2:
  + Update lại số hạng thứ 2 hiện tại
  + Hiển thị ra LED

  Nếu phím đó là toán tử = :
  + Thực hiện lưu lại số hạng thứ 2 (thông qua hàm savesecondnum)

+ Gán thanh ghi $k0 =1 để có thể tai sử dụng kết quả phép tính
+ Kiểm tra xem phép toán này thuộc loại nào (thông qua thanh ghi $s1)
+ Thực hiện phép toán và in kết quả ra cửa sổ RunI/O cũng như hiển thị ra LED

3. Thuật toán

- Xử lý khi có phím được nhập vào

```
.ktext 0x80000180
process:
        jal checkrow1                   # check tung hang de biet hang do co phim duoc bam khong
        bnez $t3,row1           # Neu co thi tim ra ky tu duoc nhap de hien thi ra led
        nop
        jal checkrow2
        bnez $t3,row2
        nop
        jal checkrow3
        bnez $t3,row3
        nop
        jal checkrow4
        bnez $t3,row4
```

Ta sử dụng thủ tục checkrow để kiểm tra xem phím đó có thuộc trong hàng đang kiểm tra hay không. Nếu thuộc ($t3 != 0) thì sẽ nhảy sang thủ tục row để kiểm tra cụ thể là phím gì.

- Cập nhất lại số khi nhập phím mới vào

```
update:         # update lai so sau khi nhap them 1 ki tu so vao
          mul $t0, $t0, 10
          add $t0, $t0, $t7
```

Biến tạm thời $t0 = 10 * $t0 + $ t7 ($t7 là giá trị của số hiện trên LED phải)

- Hiển thị chữ số bên trái và phải ra LED 7segment

```
SHOW_7SEG_LEFT:    # Hien thi so len led trai
        lb $t8,0($sp)         # gia tri cua byte hien thi led ben phai tu stack (luc dau la zero)
        add $sp,$sp,-4
        lb $t9,0($sp)         #load gia tri so hien tren led phai (luc dau la 0)
        add $sp,$sp,-4
        add $s3,$t9,$zero     #s2 = gia tri so o led trai
        sb $t8,0($t5)         # hien thi len led trai
SHOW_7SEG_RIGHT:        # Hien thi so len led phai
        sb $t6,0($t4)         # hien thi bit len led phai
        add $sp,$sp,4
        sb $t7,0($sp)        #day gia tri so hien tren led phai vao stack
        add $sp,$sp,4
        sb $t6,0($sp)         #day gia tri cua byte hien thi led ben phai vao stack
        add $s2,$t7,$zero     #s1 = gia tri bit led phai
        j finish
```

Giải thuật của quá trình này là sử dụng ngăn xếp (stack). Mỗi khi có số mới được nhập vào từ bàn phim , ta **lấy lần lượt 2 phần tử đầu tiên trên ngăn xếp** (tương đương với giá trị và byte) **gán cho**

các thanh ghi hiển thị bên LED trái rồi sau đó lại **gán giá trị các thanh ghi liên quan đến phím vừa nhập cho các thanh ghi hiển thị bên LED phải** và **đẩy 2 phần tử mới đó vào ngăn xếp**. Quá trình này sẽ liên tục lặp lại .

- Hiển thị kết quả phép tinh ra LED 7seg

```
show_result_7seg:       #ham chia ket qua thanh 2 chu so de hien thi len tung led
        li $t9,10
        div $s6,$t9     # s6/10

        mflo $t7        # t7 = chu so ben trai
        jal convert_7seg     # convert t7 thanh bit hien thi len led
        sb $t6,0($t5)   # hien thi len led trai
        add $sp,$sp,4
        sb $t7,0($sp)       #day gia tri bit nay vao stack
        add $sp,$sp,4
        sb $t6,0($sp)        #day bit nay vao stack
        add $s3,$t7,$zero    #s1 = gia tri bit led phai

        mfhi $t7        # t7= chu so ben phai
        jal convert_7seg     # convert t7 thanh bit hien thi len led
        sb $t6,0($t4)   #hien thi len led phai
        add $sp,$sp,4
        sb $t7,0($sp)       # day gia tri bit nay vao stack
        add $sp,$sp,4
        sb $t6,0($sp)        # day bit nay vao stack
        add $s2,$t7,$zero    # s1 = gia tri bit led phai
        j reset_left_7seg      # ham reset lai led
```

Để hiển thị kết quả phép tinh ra LED 7seg sao cho chỉ hiện ra 2 chữ số cuối của kết quả, ta chia kết quả phép tinh cho 100 rồi lấy phần dư là 2 chữ số cuối. Sau đó tiếp tục chia phần dư này cho 10 . Thương của phép tính trên chinh là chữ số bên trai, còn phần dư chính là chữ số bên phải.

4. Mã nguồn

```
# n10_g15_NguyenTuanNam

.data

.eqv IN_ADDRESS_HEXA_KEYBOARD 0xFFFF0012

.eqv OUT_ADDRESS_HEXA_KEYBOARD 0xFFFF0014

#-----------------------------------------------------

# col 0x1 col 0x2 col 0x4 col 0x8

#
```

```
# row 0x1 0 1 2 3

# 0x11 0x21 0x41 0x81

#

# row 0x2 4 5 6 7

# 0x12 0x22 0x42 0x82

#

# row 0x4 8 9 a b

# 0x14 0x24 0x44 0x84

#

# row 0x8 c d e f

# 0x18 0x28 0x48 0x88

#

#-----------------------------------------------------

# command row number of hexadecimal keyboard (bit 0 to 3)

# Eg. assign 0x1, to get key button 0,1,2,3

# assign 0x2, to get key button 4,5,6,7

# NOTE must reassign value for this address before reading,

# eventhough you only want to scan 1 row

# receive row and column of the key pressed, 0 if not key pressed

# Eg. equal 0x11, means that key button 0 pressed.

# Eg. equal 0x28, means that key button D pressed.



.eqv SEVENSEG_LEFT 0xFFFF0011                # Dia chi cua den led 7 doan trai.

.eqv SEVENSEG_RIGHT 0xFFFF0010               # Dia chi cua den led 7 doan phai

zero:  .byte 0x3f    #gia tri bit tuong ung

one:   .byte 0x6

two:   .byte 0x5b
```

```
three: .byte 0x4f
four:  .byte 0x66
five:  .byte 0x6d
six:   .byte 0x7d
seven: .byte 0x7
eight: .byte 0x7f
nine:  .byte 0x6f
warning: .asciiz "khong the thuc hien phep chia cho 0 \n"



.text
main:
declare:
        li $t5,SEVENSEG_LEFT        # $t5: Bien gia tri so cua den LED trai
    li $t4,SEVENSEG_RIGHT           # $t1: Bien gia tri so cua den LED phai
    li $s0,0                        # bien kiem tra loai bien nhap vao, 0: so, 1 :toan tu
    li $s1,0                        # bien kiem tra loai toan tu, 1:cong, 2:tru, 3:nhan, 4:chia, 5 : %
    li $s2,0                        # so dang hien thi o led phai
    li $s3,0                        # so dang hien thi o led trai
    li $s4,0                        # so thu nhat
    li $s5,0                        # so thu hai
    li $s6,0                        # ket qua 2 so, cong ,tru, nhan, chia

    li $t0,0                        # gia tri so tam thoi
        li $t1, IN_ADDRESS_HEXA_KEYBOARD  #bien dieu khien hang keyboard va enable keyboard interrupt
        li $t2, OUT_ADDRESS_HEXA_KEYBOARD #bien chua vi tri key nhap vao the hang va cot
        li $t3, 0x80                      # bit 7 of = 1 to enable interrupt
```

```
        sb $t3, 0($t1)
        li $t6,0                    #byte hien thi len led ben phai
        li $t7,0                        #gia tri cua so hien tren led ben phai
        li $t8,0                    #byte hien thi len led ben trai
        li $t9,0                        #gia tri cua so hien tren led ben trai


storefirstvalue:
        li $t9,0                    #gia tri cua bit can hien thi ban dau :0
        addi $sp,$sp,4                  #day vao stack
    sb $t9,0($sp)
        lb $t8,zero                     #bit dau tien can hien thi :0
        addi $sp,$sp,4              #day vao stack
    sb $t8 ,0($sp)
loop:   #loop de doi nhap phim tu digital lab sim
        nop
        b loop
endloop:


end_main:
        li $v0,10
        syscall
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#  GENERAL INTERRUPT SERVED ROUTINE for all interrupts
# Xu ly khi co interupt va hien thi so vua bam len den led 7 doan
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.ktext 0x80000180
process:
```

```
            jal checkrow1                       # check tung hang de biet hang do co phim duoc bam khong
            bnez $t3,row1               # Neu co thi tim ra ky tu duoc nhap de hien thi ra led
            nop
            jal checkrow2
            bnez $t3,row2
            nop
            jal checkrow3
            bnez $t3,row3
            nop
            jal checkrow4
            bnez $t3,row4
checkrow1:
            addi $sp,$sp,4
        sw $ra,0($sp)                       # luu ra lai vi ve sau co the doi
        li $t3,0x81             # Kich hoat interrupt, cho phep bam phim o hang 1
        sb $t3,0($t1)

        li $t2,OUT_ADDRESS_HEXA_KEYBOARD  # gan dia chi chua vi tri cua phim duoc nhap cho t2
        lb $t3,0($t2)                       # load byte cua vi tri phim duoc nhap

        lw $ra,0($sp)
        addi $sp,$sp,-4
        jr $ra
checkrow2:
            addi $sp,$sp,4
        sw $ra,0($sp)
            li $t3,0x82         # kich hoat interrupt, cho phep bam phim o hang 2
```

```
        sb $t3,0($t1)


        li $t2,OUT_ADDRESS_HEXA_KEYBOARD  # gan dia chi chua vi tri cua phim duoc nhap cho t2
        lb $t3,0($t2)                     # load byte cua vi tri phim duoc nhap


        lw $ra,0($sp)
        addi $sp,$sp,-4
        jr $ra
checkrow3:
            addi $sp,$sp,4
        sw $ra,0($sp)
            li $t3,0x84             # Kich hoat interrupt, cho phep bam phim o hang 3
        sb $t3,0($t1)


        li $t2,OUT_ADDRESS_HEXA_KEYBOARD  # gan dia chi chua vi tri cua phim duoc nhap cho t2
        lb $t3,0($t2)                     # load byte cua vi tri phim duoc nhap


        lw $ra,0($sp)
        addi $sp,$sp,-4
        jr $ra
checkrow4:
            addi $sp,$sp,4
        sw $ra,0($sp)
            li $t3,0x88             # Kich hoat interrupt, cho phep bam phim o hang 4
        sb $t3,0($t1)


        li $t2,OUT_ADDRESS_HEXA_KEYBOARD  # gan dia chi chua vi tri cua phim duoc nhap cho t2
```

```
        lb $t3,0($t2)                   # load byte cua vi tri phim duoc nhap



        lw $ra,0($sp)
        addi $sp,$sp,-4
        jr $ra


row1:                                           # check tung ky tu trong row
            beq $t3,0x11,key_0
            beq $t3,0x21,key_1
            beq $t3,0x41,key_2
            beq $t3,0xffffff81,key_3
key_0:
            li $k0,0    # nhap hep tinh moi ( khong su dung lai ket qu phep tinh cu )
            lb $t6,zero                 #t4=zero (tuc = 0x3f, tong cac bit thanh ghi de tao thanh so 0 tren led)
            li $t7,0            #t7= 0
            j update
key_1:
            li $k0,0
            lb $t6,one
            li $t7,1
            j update
key_2:
            li $k0,0
            lb $t6,two
            li $t7,2
            j update
key_3:
```

```
        li $k0,0
        lb $t6,three
        li $t7,3
        j update
row2:
        beq $t3,0x12,key_4
        beq $t3,0x22,key_5
        beq $t3,0x42,key_6
        beq $t3,0xffffff82,key_7
key_4:
        li $k0,0
        lb $t6,four
        li $t7,4
        j update
key_5:
        li $k0,0
        lb $t6,five
        li $t7,5
        j update
key_6:
        li $k0,0
        lb $t6,six
        li $t7,6
        j update
key_7:
        li $k0,0
        lb $t6,seven
```

```
            li $t7,7
            j update
row3:
            beq $t3,0x14,key_8
            beq $t3,0x24,key_9
            beq $t3 0x44,key_a
            beq $t3 0xffffff84,key_b
key_8:
            li $k0,0
            lb $t6,eight
            li $t7,8
            j update
key_9:
            li $k0,0
            lb $t6,nine
            li $t7,9
            j update
key_a:    #bam phim cong
            addi $a3,$zero,1
            addi $s0,$s0,1          # bien check phim nhap vao la 1 toan tu
            addi $s1,$zero,1   # bien check the loai toan tu
            j savefirstnum        # Luu lai so thu 1
key_b: #bam phim tru
            addi $a3,$zero,2
            addi $s0,$s0,1
            addi $s1,$zero,2
            j savefirstnum
```

```
row4:
        beq $t3,0x18,key_c
        beq $t3,0x28,key_d
        beq $t3,0x48,key_e
        beq $t3 0xffffff88,key_f
key_c: #bam phim nhan
        addi $a3,$zero,3
        addi $s0,$s0,1
        addi $s1,$zero,3
        j savefirstnum
key_d: #bam phim chia
        addi $a3,$zero,4
        addi $s0,$s0,1
        addi $s1,$zero,4
        j savefirstnum

key_e:  #bam phim lay so du
        addi $a3,$zero,5
        addi $s0,$s0,1
        addi $s1,$zero,5
        j savefirstnum

savefirstnum:                   # Luu lai so thu 1 sau khi co phim nhan vao la 1 toan tu
        bne $k0,$zero,savefirstnumtoRecalculate
        addi $s4, $t0, 0
        li $t0, 0
        j checkoperator
```

```
savefirstnumtoRecalculate :# check neu nhu co bien su dung lai kq phep tinh
            add $t0,$zero,$s6
            addi $s4, $t0, 0
            li $t0, 0
            j checkoperator

key_f:  # bam phim =
            li $k0,1    # co the su dung ket qua phep tinh nay o lan sau
            jal savesecondnumber
            beq $s1,1,sum         # s3=1--> cong
            beq $s1,2,subtract
            beq $s1,3,multiply
            beq $s1,4,divide
            beq $s1,5,surplus




savesecondnumber:  #ham tinh so thu 2 dang hien thi tren led trong 2 so
            addi $s5, $t0, 0     # Luu lai so thu 2 khi co toan tu nhap vao la =
            jr $ra

sum:
            add $s6,$s5,$s4
            li $s1,0
            li $t0, 0
```

```
        j printsum
        nop                 # s6=s5+s4

printsum:
        li $v0, 1
        move $a0, $s4
        syscall
        li $s4, 0

        li $v0, 11
        li $a0, ' '
        syscall

        li $v0, 11
        li $a0, '+'
        syscall

        li $v0, 11
        li $a0, ' '
        syscall

        li $v0, 1
        move $a0, $s5
        syscall
        li $s5, 0           #reset $s5

        li $v0, 11
```

```
        li $a0, ' '
        syscall

        li $v0, 11
        li $a0, '='
        syscall

        li $v0, 11
        li $a0, ' '
        syscall

        li $v0, 1
        move $a0, $s6
        syscall
        nop

        li $v0, 11
        li $a0, '\n'
        syscall
        li $s7,100
        div $s6,$s7
        mfhi $s6    # chi lay 2 chu so cuoi cua ket qua de in ra led
        j show_result_7seg      # chuyen den ham chia ket qua thanh 2 chu so de hien thi len tung led
        nop

subtract:
        sub $s6,$s4,$s5    # s6=s4-s5
```

```
        li $s1,0
        li $t0, 0
        j printsubtract
        nop
printsubtract:
        li $v0, 1
        move $a0, $s4
        syscall

        li $v0, 11
        li $a0, ' '
        syscall

        li $v0, 11
        li $a0, '-'
        syscall

        li $v0, 11
        li $a0, ' '
        syscall

        li $v0, 1
        move $a0, $s5
        syscall

        li $v0, 11
        li $a0, ' '
```

```
        syscall
        li $v0, 11
        li $a0, '='
        syscall

        li $v0, 11
        li $a0, ' '
        syscall

        li $v0, 1
        move $a0, $s6
        syscall

        li $v0, 11
        li $a0, '\n'
        syscall
        j show_result_7seg      # chuyen den ham chia ket qua thanh 2 chu so de hien thi len tung led
        nop

multiply:
        mul $s6,$s4,$s5     # s6=s4*s5
        li $s1,0
        li $t0, 0
        j printmultiply
        nop
printmultiply:
        li $v0, 1
```

```
move $a0, $s4
syscall

li $v0, 11
li $a0, ' '
syscall

li $v0, 11
li $a0, '*'
syscall

li $v0, 11
li $a0, ' '
syscall

li $v0, 1
move $a0, $s5
syscall

li $v0, 11
li $a0, ' '
syscall

li $v0, 11
li $a0, '='
syscall
```

```
		li $v0, 11
		li $a0, ' '
		syscall

		li $v0, 1
		move $a0, $s6
		syscall

		li $v0, 11
		li $a0, '\n'
		syscall
		li $s7,100
		div $s6,$s7
		mfhi $s6	# chi lay phan du la 2 chu so sau cung cua ket qua de in ra
		j show_result_7seg	# chuyen den ham chia ket qua thanh 2 chu so de hien thi len tung led
		nop

divide:
		beq $s5,0,divide0
		li $s1,0
		div $s4,$s5		# s6=s4/s5
		mflo $s6
		mfhi $s7
		li $t0, 0
		j printdivide
		nop
printdivide:
```

```
li $v0, 1
move $a0, $s4
syscall

li $v0, 11
li $a0, ' '
syscall

li $v0, 11
li $a0, '/'
syscall

li $v0, 11
li $a0, ' '
syscall

li $v0, 1
move $a0, $s5
syscall

li $v0, 11
li $a0, ' '
syscall
li $v0, 11
li $a0, '='
syscall
```

```
li $v0, 11
li $a0, ' '
syscall

li $v0, 1
move $a0, $s6
syscall

li $v0, 11
li $a0, ' '
syscall

li $v0, 11
li $a0, 'r'
syscall

li $v0, 11
li $a0, '='
syscall

li $v0, 1
move $a0, $s7
syscall

li $v0, 11
li $a0, '\n'
syscall
```

```
        j show_result_7seg      # chuyen den ham chia ket qua thanh 2 chu so de hien thi len tung led
        nop
divide0:
        li $v0, 55
        la $a0, warning
        li $a1, 0
        syscall
        j reset_left_7seg

surplus:
        beq $s5,0,divide0
        li $s1,0
        div $s4,$s5             # s6=s4/s5
        mfhi $s6
        li $t0, 0
        j printsurplus
        nop
printsurplus:
        li $v0, 1
        move $a0, $s4
        syscall

        li $v0, 11
        li $a0, ' '
        syscall

        li $v0, 11
```

```
li $a0, '%'
syscall

li $v0, 11
li $a0, ' '
syscall

li $v0, 1
move $a0, $s5
syscall

li $v0, 11
li $a0, ' '
syscall

li $v0, 11
li $a0, '='
syscall

li $v0, 11
li $a0, ' '
syscall

li $v0, 1
move $a0, $s6
syscall
```

```
            li $v0, 11
            li $a0, '\n'
            syscall
            j show_result_7seg       # chuyen den ham chia ket qua thanh 2 chu so de hien thi len tung led
            nop



show_result_7seg: #ham chia ket qua thanh 2 chu so de hien thi len tung led
            li $t9,10
            div $s6,$t9    # s6/10

            mflo $t7       # t7 = chu so ben trai
            jal convert_7seg    # convert t7 thanh bit hien thi len led
        sb $t6,0($t5)  # hien thi len led trai
            add $sp,$sp,4
            sb $t7,0($sp)            #day gia tri bit nay vao stack
            add $sp,$sp,4
            sb $t6,0($sp)       #day bit nay vao stack
            add $s3,$t7,$zero   #s1 = gia tri bit led phai

            mfhi $t7       # t7= chu so ben phai
            jal convert_7seg    # convert t7 thanh bit hien thi len led
        sb $t6,0($t4)  #hien thi len led phai
            add $sp,$sp,4
            sb $t7,0($sp)            # day gia tri bit nay vao stack
            add $sp,$sp,4
            sb $t6,0($sp)       # day bit nay vao stack
```

```
            add $s2,$t7,$zero   # s1 = gia tri bit led phai
        j reset_left_7seg     # ham reset lai led
convert_7seg:
            addi $sp,$sp,4
        sw $ra,0($sp)
        beq $t7,0,case_0    # t7=0 -->ham chuyen 0 thanh bit zero hien thi len led
        beq $t7,1,case_1
        beq $t7,2,case_2
        beq $t7,3,case_3
        beq $t7,4,case_4
        beq $t7,5,case_5
        beq $t7,6,case_6
        beq $t7,7,case_7
        beq $t7,8,case_8
        beq $t7,9,case_9
case_0:   #ham chuyen 0 thanh bit zero hien thi len led
            lb $t6,zero    #t4=zero
            lw $ra,0($sp)
            addi $sp,$sp,-4
            jr $ra
case_1:
            lb $t6,one
            lw $ra,0($sp)
            addi $sp,$sp,-4
            jr $ra
case_2:
            lb $t6,two
```

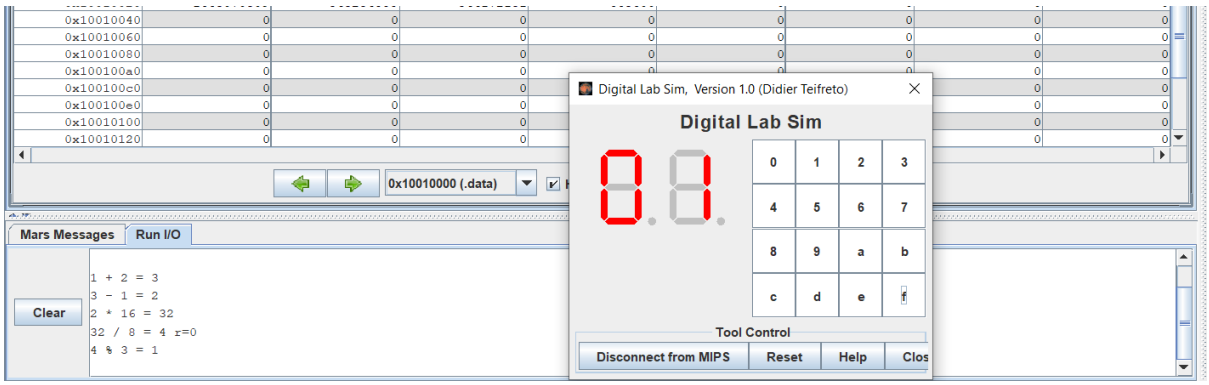
```
        lw $ra,0($sp)
        addi $sp,$sp,-4
        jr $ra
case_3:
        lb $t6,three
        lw $ra,0($sp)
        addi $sp,$sp,-4
        jr $ra
case_4:
        lb $t6,four
        lw $ra,0($sp)
        addi $sp,$sp,-4
        jr $ra
case_5:
        lb $t6,five
        lw $ra,0($sp)
        addi $sp,$sp,-4
        jr $ra
case_6:
        lb $t6,six
        lw $ra,0($sp)
        addi $sp,$sp,-4
        jr $ra
case_7:
        lb $t6,seven
        lw $ra,0($sp)
        addi $sp,$sp,-4
```

```
            jr $ra
case_8:
            lb $t6,eight
            lw $ra,0($sp)
            addi $sp,$sp,-4
            jr $ra
case_9:
            lb $t6,nine
            lw $ra,0($sp)
            addi $sp,$sp,-4
            jr $ra




update:             # update lai so sau khi nhap them 1 ki tu so vao
             mul $t0, $t0, 10
             add $t0, $t0, $t7

checkoperator:
            beq $s0,1,reset_left_7seg  # reset lai so o ben leg ben trai ve 0 khi phim vua nhap vao la 1 toan tu
            nop

SHOW_7SEG_LEFT:   # Hien thi so len led trai
            lb $t8,0($sp)       # gia tri cua byte hien thi led ben phai tu stack (luc dau la zero)
            add $sp,$sp,-4
            lb $t9,0($sp)       #load gia tri so hien tren led phai (luc dau la 0)
```

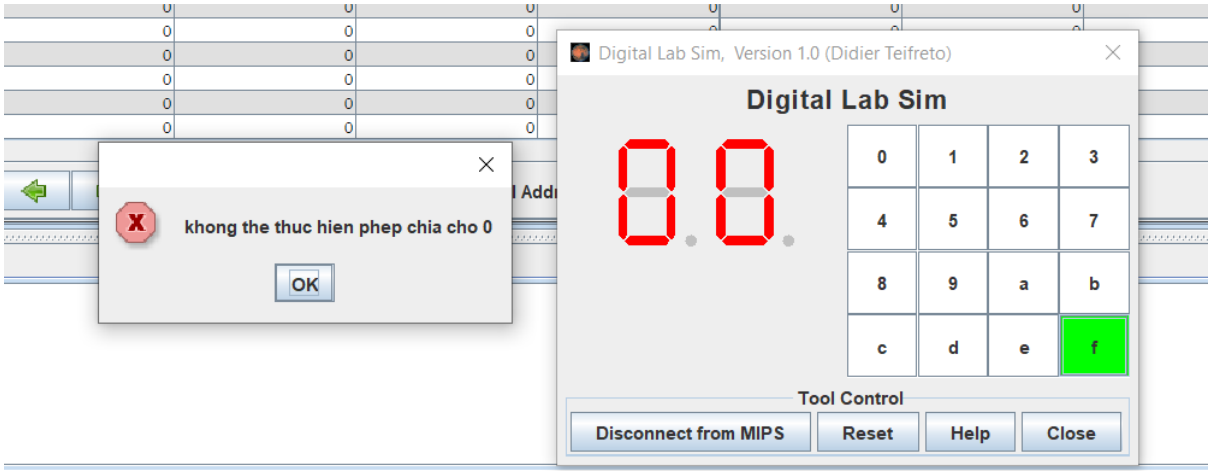
```
            add $sp,$sp,-4
            add $s3,$t9,$zero   #s2 = gia tri so o led trai
            sb $t8,0($t5)       # hien thi len led trai
SHOW_7SEG_RIGHT:            # Hien thi so len led phai
            sb $t6,0($t4)       # hien thi bit len led phai
            add $sp,$sp,4
            sb $t7,0($sp)             #day gia tri so hien tren led phai vao stack
            add $sp,$sp,4
            sb $t6,0($sp)       #day gia tri cua byte hien thi led ben phai vao stack
            add $s2,$t7,$zero   #s1 = gia tri bit led phai
            j finish


reset_left_7seg:       # reset lai so o ben tahi leg ben trai ve 0 khi phim vua nhap vao la 1 toan tu
            li $s0,0           #s0=0--> doi nhap so tiep theo trong 2 so
        li $t9,0
            addi $sp,$sp,4
        sb $t9,0($sp)
        lb $t8,zero        # day bit zero vao stack
            addi $sp,$sp,4
        sb $t8,0($sp)


finish:
            la $a3, loop
            mtc0 $a3, $14
            eret
```

5. Kết quả chạy mô phỏng

- **Các phép toán thường**



- **Phép toán chia cho 0**



B. ĐỀ 7

Chương trình kiểm tra cú pháp lệnh MIPS

Trình biên dịch của bộ xử lý MIPS sẽtiến hành kiểm tra cú pháp các lệnh hợp ngữtrong mã nguồn, xem có phù hợp vềcú 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 dụbeq s1,31,t4
-Kiểm tra xem mã opcode có đúng hay không? Trong ví dụ trên, opcode là beq là hợp lệ thì hiện thịthông 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.

1. Phân tích đề bài Input:

Nhập vào một lệnh cần kiểm tra Output : Kiểm tra từng thành toán hạng và opcode của câu lệnh, nếu hợp lệ thì in ra còn không thì sẽ báo lỗi

Phân tích cách thực hiện : Khai báo thư viện các opcode, mã lệnh, toán hạng hợp lệ có trong mips 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 – không có, 1 – thanh ghi, 2 – hằng số nguyên, 3 – định danh.

Sau đó tách từng phần của chuỗi tương ứng với thành phần của câu lệnh hợp ngữ. Đưa từng phần đã cắt được đi so sánh với kiểu dữ liệu quy ước. Nếu phù hợp thì in ra hợp lệ và tiếp tục so sánh các phần ở phía sau. Nếu không phù hợp thì báo cho người dùng opcode hoặc toán hạng không hợp lệ.

+ Đầu chương trình, đã 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 thì báo opcode không hợp lệ. 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 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 vàsẽ so sánh với chuỗi các register chuẩn.

+ Trường hợp toán hạng là hằng số nguyên: bắt buộc phải từ 0 đến 9.

+ Trường hợp không có: kiểm tra xem chuỗi vừa cắt được có kí tự nào hay không, nếu có thì không hợp lệ, nếu không có kí tự nào hợp lệ và thông báo câu lệnh hợp lệ. Thoát chương trình. Sau khi kiểm tra xong 3 toán hạng, chương trình sẽ kiểm tra xem còn kí tự nào khác hay không. Nếu còn thì câu lệnh không hợp lệ. Và báo kết quả của câu lệnh vừa nhập. Tiếp tục hỏi người dùng có muốn kiểm tra tiếp một câu lệnh khác hay không? Nếu người dùng chọn không, kết thúc chương trình

2.2. Source code

```
.data
    Nhap: .asciiz "Nhap vao mot dong lenh hop ngu: "
    continueMessage: .asciiz "Ban muon tiep tuc chuong trinh khong? (0.Yes/1.No)"
    errMessage: .asciiz "Lenh hop ngu khong hop le. Loi cu phap!\n"
    NF: .asciiz "Khong tim duoc khuon dang lenh nay!\n"
    endMessage: .asciiz "\nHoan thanh! Lenh vua nhap vao phu hop voi cu phap!\n"
    hopLe1: .asciiz "Opcode: "
    hopLe11: .asciiz "Toan hang: "
    hopLe2: .asciiz "hop le.\n"
    command: .space 100
    opcode: .space 10
    token: .space 20
    number: .space 15
    ident: .space 30
    # 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***1111;xor**1111;lui**1201;jr***1001;jal**3002;addi*1121;add**1111;sub**1111;ori**1121;and**1111;beq**1132;bne**1132;j****3002;nop**0001;"
    charGroup: .asciiz "qwertyuiopasdfghjklmnbvcxzQWERTYUIOPASDFGHJKLZXCVBNM_"
    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  "

.text
```

```
readData: # Doc lenh nhap vao tu ban phim
    li $v0, 4
    la $a0, Nhap
    syscall
    li $v0, 8
    la $a0, command # chua dia chi cua lenh nhap vao
    li $a1, 100 # so ki tu toi da
    syscall

main:

    li $t2, 0 # i
readOpcode:
    la $a1, opcode # luu cac ki tu doc duoc vao opcode
    add $t3, $a0, $t2 # t3 tro vao a i
    add $t4, $a1, $t2 # t4 tro vao opcode
    lb $t1, 0($t3) # doc tung ki tu cua command
    sb $t1, 0($t4)
    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
    j readOpcode

#<--xu ly opcode-->
done:
    li $t7,-10
    la $a2, library # load thu vien vao lai
xuLyOpcode:
    li $t1, 0 # i
    li $t2, 0 # j
    addi $t7,$t7,10 # buoc nhay = 10 de den vi tri opcode trong library
    add $t1,$t1,$t7 # cong buoc nhay
```

```
compare:
add $t3, $a2, $t1 # t3 tro thanh con tro cua library
lb $s0, 0($t3)
beq $s0, 0, notFound # khong tim thay opcode nao trong library
beq $s0, 42, check # gap ki tu '*' -> check xem opcode co giong nhau tiep ko?.
add $t4, $a1, $t2
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


check:
add $t4, $a1, $t2
lb $s1, 0($t4)
bne $s1, 32, check2 # neu ki tu tiep theo khong phai ' ' => 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 hop le
syscall
li $v0, 4
la $a0, opcode
syscall
li $v0, 4
la $a0, hopLe2
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
    j checkContinue

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

#<--xu li toan hang-->
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, 1 # thanh ghi = 1
    beq $s0, $t8, checkTokenReg
    li $t8, 2 # hang so nguyen = 2
    beq $s0, $t8, checkHSN
    li $t8, 3 # dinh danh = 3
    beq $s0, $t8, checkIdent
    li $t8, 0 # khong co toan hang = 0
    beq $s0, $t8, checkNT
    j end

#<--check Token Register-->
checkTokenReg:
    la $a0, command
    la $a1, token # luu ten thanh ghi vao token de so sanh
    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
            add $t4, $a1, $t2
            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 compare
            li $t1, -1 # i
            li $t2, -1 # j
            li $t4, 0
            li $t5, 0
            add $t2, $t2, $k1
            la $a1, token
            la $a2, tokenRegisters
            j compareToken

compareToken:
    addi $t1,$t1,1
    addi $t2,$t2,1
    add $t4, $a1, $t1
    lb $s0, 0($t4)
    beq $s0, 0, end
    add $t5, $a2, $t2
    lb $s1, 0($t5)
    beq $s1, 0, notFound
```

```
    beq $s1, 32, checkLengthToken
    bne $s0,$s1, jump
    j compareToken

    checkLengthToken:
        beq $s0, 44, compareE
        beq $s0, 10, 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
        j end
    compareNE:
        j notFound
#<!--ket thuc check Token Register-->

#<--check toan hang la hang so nguyen-->
checkHSN: # kiem tra co phai hang so nguyen hay ko
    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, end
    beq $s0, 45, compareNumber # bo dau '-'
    beq $s0, 10, compareNumE
    beq $s0, 44, compareNumE
    li $t2, 48
    li $t3, 57
    slt $t5, $s0, $t2
```

```
        bne $t5, $zero, compareNumNE
        slt $t5, $t3, $s0
        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
                j end
        compareNumNE:
                j notFound
#<!--ket thuc check toan hang la hang so nguyen-->

#<--check Indent-->
checkIdent:
        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, charGroup
        j compareIdent
compareIdent:
    addi $t1,$t1,1
    add $t4, $a1, $t1
    lb $s0, 0($t4)
    beq $s0, 0, end
    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 charGroup

    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
        j end
    compareIdentNE:
        j notFound
#<!--ket thuc check Indent-->

#<--kiem tra khong co toan hang-->
checkNT:
    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
#<!--ket thuc kiem tra khong co toan hang-->

#<--check Token Register 2-->
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, 1 # thanh ghi = 1
    beq $s0, $t8, checkTokenReg
    li $t8, 2 # hang so nguyen = 2
    beq $s0, $t8, checkHSN
    li $t8, 3 # dinh danh = 3
    beq $s0, $t8, checkIdent
    li $t8, 0 # khong co toan hang = 0
    beq $s0, $t8, checkNT
    j end
#<!--ket thuc check Token Register 2-->

#<--check Token Register 3-->
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, 1 # thanh ghi = 1
        beq $s0, $t8, checkTokenReg
        li $t8, 2 # hang so nguyen = 2
        beq $s0, $t8, checkHSN
        li $t8, 3 # dinh danh = 3
        beq $s0, $t8, checkIdent
        li $t8, 0 # khong co toan hang = 0
        beq $s0, $t8, checkNT
        j end
    #<!--ket thuc check Token Register 3-->

    #<--ket thuc xu li toan hang-->

    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 TheEnd
error:
    li $v0, 4
    la $a0, errMessage
```

```
    syscall
    j TheEnd
end:
    li $v0, 4
    la $a0, endMessage
    syscall
    j continue
TheEnd:
```

3. Kết quả chạy chương trình Câu lệnh hợp lệ

Khuôn lệnh hợp lệ:

```
Nhap vao mot dong lenh hop ngu: add $t1,$t2,$t3
Opcode: add hop le.
Toan hang: $t1,hop le.
Toan hang: $t2,hop le.
Toan hang: $t3
hop le.

Hoan thanh! Lenh vua nhap vao phu hop voi cu phap!
Ban muon tiep tuc chuong trinh khong? (0.Yes/1.No)
```

Khuôn lệnh không hợp lệ

Messages | Run I/O

```
Nhap vao mot dong lenh hop ngu: addi $t1,$t2,$t3
Opcode: addi hop le.
Toan hang: $t1,hop le.
Toan hang: $t2,hop le.
Khong tim duoc khuon dang lenh nay!

-- program is finished running (dropped off bottom) --
```