**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**\*\*\*\*\*****\*\*\*\*\***

A red and yellow logo

Description automatically generated

**BÁO CÁO BÀI TẬP LỚN**

**THỰC HÀNH KIẾN TRÚC MÁY TÍNH**

Giảng viên: ThS. Lê Bá Vui

Mã lớp học: 147789

Nhóm: 13

  Danh sách sinh viên thực hiện:

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **Mã sinh viên** | **Đề tài số** |
| 01 | Nguyễn Đức Trường | 20225675 | **05** |
| 02 | Nguyễn Mạnh Tuấn | 20225679 | **07** |

Bài 5 : Biểu thức trung tố hậu tố

Thực hiện: Nguyễn Đức Trường - 20225675

1. 1 Phân tích bài toán

Đề bài:

Viết chương trình tính giá trị biểu thức bất kỳ bằng phương pháp duyệt biểu thức hậu tố. Các yêu cầu cụ thể: 1. Nhập vào biểu thức trung tố, ví dụ: 9 + 2 + 8 \* 6 2. In ra biểu thức ở dạng hậu tố, ví dụ: 9 2 + 8 6 \* + 3. Tính ra giá trị của biểu thức vừa nhập Các hằng số là số nguyên, trong phạm vi từ 0 → 99. Toán tử bao gồm các phép toán cộng, trừ, nhân, chia lấy thương (/), chia lấy dư (%), đóng mở ngoặc

Yêu cầu bài toán: Nhập một biểu thức trung tố (infix). Chuyển đổi biểu thức trung tố thành biểu thức hậu tố (postfix). Tính giá trị của biểu thức hậu tố.

1. 2 Ý tưởng thuật toán

Chuyển đổi biểu thức trung tố sang biểu thức hậu tố: Sử dụng ngăn xếp để lưu các toán tử và một danh sách để lưu kết quả. Duyệt qua từng ký tự của biểu thức trung tố, nếu gặp toán hạng (số) thì đưa vào danh sách kết quả, nếu gặp toán tử thì xử lý theo mức độ ưu tiên của các toán tử.

Tính toán giá trị của biểu thức hậu tố: Sử dụng ngăn xếp để tính toán giá trị của biểu thức hậu tố. Duyệt qua từng ký tự của biểu thức hậu tố, nếu gặp toán hạng (số) thì đưa vào ngăn xếp, nếu gặp toán tử thì lấy hai số từ ngăn xếp ra để tính toán và đưa kết quả trở lại ngăn xếp.

1. 3 Source Code

.data

Bttrungto: .space 256

Bthauto: .space 256

Nganxep: .space 256

Arr: .space 256

String: .asciiz “\n”

Message1: .asciiz “Bieu thuc trung to: “

Message2: .asciiz “Bieu thuc hau to: “

Message3: .asciiz “Ket qua bieu thuc vua nhap: “

Message4: .asciiz “Nhap vao bieu thuc trung to: “

Message5: .asciiz “MENU\n1.Chay chuong trinh\n2.Thoat chuong trinh\nBan chon?\n “

Message6: .asciiz “Bieu thuc khong hop le.\n”

.text

Main:la $a0, message5

Li $v0, 4

Syscall

Li $v0, 5

Syscall

Beq $v0, 2, end\_main

Jal INPUT

Nop

Jal CHECK

Nop

Beq $v0, 0, next1

Nop

Jal OUTPUT1

Nop

Jal CONVERT

Nop

Jal OUTPUT2

Jal CALCULATE

Jal OUTPUT3

J main

Nop

Next1:la $a0, message6

Li $v0, 4

Syscall

J main

End\_main:

Li $v0, 10

Syscall

#Nhap bieu thuc trung to

INPUT:la $a0, message4

Li $v0, 4

Syscall

Li $v0, 8

La $a0, bttrungto

La $a1, 256

Syscall

Jr $ra

Nop

#In bieu thuc trung to ra man hinh

OUTPUT1:la $a0, message1

Li $v0, 4

Syscall

La $a0, bttrungto

Li $v0, 4

Syscall

Jr $ra

Nop

#In bieu thuc hau to ra man hinh

OUTPUT2: la $a0, message2

Li $v0, 4

Syscall

La $a0, bthauto

Li $v0, 4

Syscall

La $a0, string

Li $v0, 4

Syscall

Jr $ra

Nop

#In ket qua ra man hinh

OUTPUT3:la $a0, message3

Li $v0, 4

Syscall

Lb $a0, -1($s3)

Li $v0, 1

Syscall

La $a0, string

Li $v0, 4

Syscall

Jr $ra

Nop

#Kiem tra tinh hop le cua bieu thuc vua nhap

CHECK:addi $sp, $sp, -4

Sw $ra, 0($sp)

La $a0, bttrungto

Li $t1, 0

Load:add $t2, $t1, $a0

Lb $t3, 0($t2)

Beq $t3, 10, end\_check

Jal check\_number

Beq $v0, 1, continue

Jal check\_operator

Beq $v0, 1, continue

Jal check\_space

Beq $v0, 1, continue

J end\_check

Continue:

Addi $t1, $t1, 1

J load

End\_check:

Lw $ra, 0($sp)

Addi $sp, $sp, 4

Jr $ra

Nop

#CONVERT : chuyen bieu thuc trung to thanh bieu thuc hau to

CONVERT:addi $sp, $sp, -4

Sw $ra, 0($sp)

La $s1, bttrungto

La $s2, bthauto

La $s3, nganxep

Li $s6, 0

Li $s7, 0

Li $t7, -1

While1:add $s4, $s1, $s6

Lb $t3, 0($s4)

Beq $t3, 10, end\_while1

Jal check\_number

Beq $v0, 1, push\_number

Next3:beq $t3, 10, end\_while1

Jal check\_operator

Beq $v0, 1, check\_before\_push\_operator

Next2:addi $s6, $s6, 1

J while1

Push\_number:add $s4, $s2, $s7

Sb $t3, 0($s4)

Addi $s7, $s7, 1

Addi $s6, $s6, 1

Add $s4, $s1, $s6

Lb $t3, 0($s4)

Jal check\_number

Beq $v0, 1, push\_number

Li $t2, ‘ ‘

Add $s4, $s2, $s7

Sb $t2, 0($s4)

Addi $s7, $s7, 1

J next3

Check\_before\_push\_operator:beq $s0, 0, push\_operator1

Beq $t7, -1, push\_operator1

Addi $t1, $t3, 0

Addi $a0, $s0, 0

After\_pop:beq $t7, -1, push\_operator2

Add $s4, $t7, $s3

Lb $t3, 0($s4)

Jal check\_operator

Ble $a0, $s0, pop\_operator

J push\_operator2

Pop\_operator:beq $t3, ‘(‘, before\_pop\_operator

Add $s4, $s2, $s7

Sb $t3, 0($s4)

Addi $s7, $s7, 1

Li $t2, ‘ ‘

Add $s4, $s2, $s7

Sb $t2, 0($s4)

Addi $s7, $s7, 1

Addi $t7, $t7, -1

J after\_pop

Before\_pop\_operator:addi $t7, $t7, -1

J push\_operator2

Push\_operator1:addi $t7, $t7, 1

Add $s4, $s3, $t7

Sb $t3, 0($s4)

J next2

Push\_operator2:beq $t1, ‘)’, after\_pop2

Addi $t7, $t7, 1

Add $s4, $s3, $t7

Sb $t1, 0($s4)

J next2

After\_pop2:

Add $s4, $t7, $s3

Lb $t3, 0($s4)

Addi $t7, $t7, -1

Beq $t3, ‘(‘, push\_operator3

Add $s4, $s2, $s7

Sb $t3, 0($s4)

Addi $s7, $s7, 1

Li $t2, ‘ ‘

Add $s4, $s2, $s7

Sb $t2, 0($s4)

Addi $s7, $s7, 1

J after\_pop2

Push\_operator3:

J next2

End\_while1:beq $t7, -1, rt3

Add $s4, $s3, $t7

Lb $t3, 0($s4)

Addi $t7, $t7, -1

Add $s4, $s2, $s7

Sb $t3, 0($s4)

Addi $s7, $s7, 1

Li $t2, ‘ ‘

Add $s4, $s2, $s7

Sb $t2, 0($s4)

Addi $s7, $s7, 1

J end\_while1

Rt3: li $t2, ‘\0’

Add $s4, $s2, $s7

Sb $t2, 0($s4)

Lw $ra, 0($sp)

Addi $sp, $sp, 4

Jr $ra

Nop

#CACULATE

# $s2 : bieu thuc hau to

# $s3 : ngan xep

CALCULATE:addi $sp, $sp, -4

Sw $ra, 0($sp)

Li $s6, 0

Li $s7, -4

La $t2, arr

While2:add $s4, $s2, $s6

Lb $t3, 0($s4)

Beq $t3, 0, end\_while2

Jal check\_number

Li $t9, -1

Li $s5, 0

Beq $v0, 1, before\_convert\_num

Next4:beq $t3, 0, end\_while2

Jal check\_operator

Beq $v0, 1, ccl

Next5:addi $s6, $s6, 1

J while2

Before\_convert\_num:add $s4, $t2, $s5

Sb $t3, 0($s4)

Addi $s5, $s5, 1

Addi $t9, $t9, 1

Addi $s6, $s6, 1

Add $s4, $s2, $s6

Lb $t3, 0($s4)

Jal check\_number

Beq $v0, 1, before\_convert\_num

Jal convert\_num

J next4

Ccl: addi $s3, $s3, -2

Lb $a0, ($s3)

Lb $a1, 1($s3)

Jal CAL

J next5

End\_while2:

Lw $ra, ($sp)

Addi $sp, $sp, 4

Jr $ra

#Check xem co phai ki tu space khong?

# $v0 = 0 -> khong la space

# $v0 = 1 -> la space

Check\_space:

Bne $t3, ‘ ‘, check\_space\_false

Check\_space\_true:

Li $v0, 1

Jr $ra

Nop

Check\_space\_false:

Li $v0, 0

Jr $ra

Nop

#Check xem co phai toan tu khong?

# $v0 = 0 -> khong phai la toan tu

# $v0 = 1 -> la toan tu, $s0 luu tru thu tu uu tien cua toan tu

Check\_operator:

Addi $sp, $sp, -28

Sw $s2, 24($sp)

Sw $s3, 20($sp)

Sw $s4, 16($sp)

Sw $s5, 12($sp)

Sw $s6, 8($sp)

Sw $s7, 4($sp)

Sw $t8, ($sp)

Li $s2, ‘+’

Li $s3, ‘-‘

Li $s4, ‘\*’

Li $s5, ‘/’

Li $s6, ‘%’

Li $s7, ‘(‘

Li $t8, ‘)’

Beq $t3, $s2, operator1

Beq $t3, $s3, operator1

Beq $t3, $s4, operator2

Beq $t3, $s5, operator2

Beq $t3, $s6, operator2

Beq $t3, $s7, operator0

Beq $t3, $t8, operator3

J check\_operator\_false

Operator0:li $s0,0

J check\_operator\_true

Operator1:li $s0, 1

J check\_operator\_true

Operator2:li $s0, 2

J check\_operator\_true

Operator3:li $s0, 3

Check\_operator\_true:

Li $v0, 1

J rt1

Nop

Check\_operator\_false:

Li $v0, 0

Rt1: lw $s2, 24($sp)

Lw $s3, 20($sp)

Lw $s4, 16($sp)

Lw $s5, 12($sp)

Lw $s6, 8($sp)

Lw $s7, 4($sp)

Lw $t8, ($sp)

Addi $sp, $sp, 28

Jr $ra

Nop

#Check xem co phai so khong?

# $v0 = 0 -> khong la so

# $v0 = 1 -> la so

Check\_number:

Addi $sp, $sp, -8

Sw $t8, 4($sp)

Sw $t9, ($sp)

Li $t8, ‘0’

Li $t9, ‘9’

Beq $t8, $t3, check\_number\_true

Beq $t9, $t3, check\_number\_true

Bgt $t8, $t3, check\_number\_false

Bgt $t3, $t9, check\_number\_false

Check\_number\_true:

Li $v0, 1

J rt2

Check\_number\_false:

Li $v0, 0

Rt2:

Lw $t9, ($sp)

Lw $t8, 4($sp)

Addi $sp, $sp, 8

Jr $ra

#Tinh gia tri bieu thuc

# $a0 : phan tu truoc

# $a1 : phan tu sau

# $t3 : op

# $a0 op $a1

CAL:

Add\_op: bne $t3, ‘+’, minus\_op

Add $v0, $a0, $a1

J rt\_cal

Minus\_op: bne $t3, ‘-‘, mul\_op

Sub $v0, $a0, $a1

J rt\_cal

Mul\_op: bne $t3, ‘\*’, div\_op

Mult $a0, $a1

Mflo $v0

J rt\_cal

Div\_op: bne $t3, ‘/’, divr\_op

Div $a0, $a1

Mflo $v0

J rt\_cal

Divr\_op: bne $t3, ‘%’, rt\_cal

Div $a0, $a1

Mfhi $v0

Rt\_cal:

Sb $v0, ($s3)

Addi $s3, $s3, 1

Jr $ra

#Chuyen doi ky tu thanh so

# $t2 : luu ki tu

# $s3 : lu so tuong ung

# $t9 : bien dem cua mang luu ki tu

Convert\_num:

Addi $sp, $sp, -8

Sw $t1, ($sp)

Sw $t5, 4($sp)

Bne $t9, $0, twoNum

Lb $t1, 0($t2)

Addi $t1, $t1, -48

J push

twoNum:

lb $t1, 0($t2)

addi $t1, $t1, -48

addi $t5, $0, 10

mult $t1, $t5

mflo $t1

lb $t5, 1($t2)

addi $t5, $t5, -48

add $t1, $t1, $t5

push:sb $t1, 0($s3)

addi $s3, $s3, 1

lw $t5, 4($sp)

lw $t1, ($sp)

addi $sp, $sp, 8

j rt4

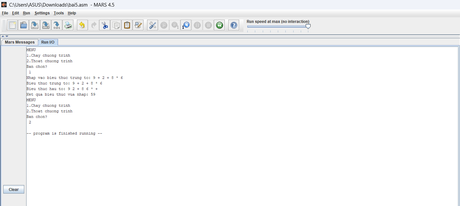
end\_loop2:

end\_loop1:

rt4:jr $ra

nop

1.4 Kết quả chương trình



Bài 7 : Chương trình kiểm tra cú pháp lệnh MIPS

Thực hiện:Nguyễn Mạnh Tuấn

MSSV:20225679

2.1Phân tích bài toán

Đề bài:

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

- Nhập vào từ bàn phím một dòng lệnh hợp ngữ. Ví dụ beq s1,31,t4

- Kiểm tra xem mã opcode có đúng hay không? Trong ví dụ trên, opcode là beq là hợp lệ thì hiện thị thông báo “opcode: beq, hợp lệ”

- Kiểm tra xem tên các toán hạng phía sau có hợp lệ hay không? Trong ví dụ trên, toán hạng s1 là hợp lệ, 31 là không hợp lệ, t4 thì khỏi phải kiểm tra nữa vì toán hạng trước đã bị sai rồi.

Yêu cầu bài toán : Hiện ra thông báo lệnh nhập vào có đúng cú pháp hay không.

2.2 Ý tưởng thuật toán

Cú pháp là: ‘opcode toán\_hạng\_1,toán\_hạng\_2,toán\_hạng 3 (có thể có hoặc không)’

Từ cú pháp ta có ý tưởng sẽ duyệt các ký từ trong lệnh nhập vào đến khi gặp dấu ‘space’ sẽ là phần opcode, sẽ dùng nó so sánh với các lệnh có sẵn trong cấu trúc các lệnh . Ta sẽ đánh dấu vị trí opcode và từ đó sẽ duyệt các toán hạng gạp các dấu “,” để nhận biết các toán hạng và so sánh với các toán hạng có sẵn.

2.3 Source code

.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:

2.4 Kết quả chạy chương trình

