

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BỘ MÔN THỰC HÀNH KIẾN TRÚC MÁY TÍNH

---o0o---



BÁO CÁO FINAL PROJECT
THỰC HÀNH KIẾN TRÚC MÁY TÍNH

Giáo viên hướng dẫn : ThS. Lê Bá Vui

Lớp : 130938

Nhóm sinh viên thực hiện:

Phạm Thị Ánh 20194480

Lê Đình Tuyên 20194715

Hà Nội, 07-2022

Mục lục

1. Project 5	1
1.1. Phân tích cách thực hiện, thuật toán.....	1
*Phân tích cách thực hiện:	1
*Ý tưởng và thuật toán:	1
1.2. Mã nguồn.....	1
1.3. Kết quả chạy.....	13
2. Project 7	13
2.1. Phân tích cách thực hiện, thuật toán	14
*Phân tích cách thực hiện:	14
*Ý tưởng:	14
*Thuật toán:	14
2.2. Mã nguồn	14
2.3. Kết quả chạy	32

NỘI DUNG

1. Project 5

Sinh viên thực hiện: Phạm Thị Ánh - 20194480

1.1. Phân tích cách thực hiện, thuật toán

*Phân tích cách thực hiện:

- Đầu vào: Biểu thức trung tố
- Đầu ra: In ra biểu thức ở dạng hậu tố và tính giá trị biểu thức hậu tố

*Ý tưởng và thuật toán:

a) Đổi biểu thức hậu tố sang trung tố

Để đổi biểu thức trung tố sang hậu tố, ta sẽ dùng ngăn xếp và xâu.

Bước 1: Đưa 1 biểu thức trung tố vào 1 xâu kí tự và đặt tên là infix.

Bước 2: Tạo ra 1 xâu mới để lưu biểu thức hậu tố, đặt tên là postfix.

Bước 3: Sắp xếp lại câu:

- Nếu kí tự là số thì lưu vào postfix.
- Nếu kí tự là toán tử, nếu ngăn xếp trống thì đẩy vào ngăn xếp.
- Nếu kí tự là dấu '(' thì cho vào ngăn xếp.
- Nếu gặp kí tự ')' thì sẽ lấy hết kí tự sau dấu '(' cho vào postfix.
- Nếu toán tử đang xét có bậc cao hơn toán tử ở đỉnh ngăn xếp thì đẩy toán tử vào ngăn xếp.
- Nếu toán tử đang xét có bậc bằng toán tử ở đỉnh ngăn xếp thì lấy toán tử đỉnh ngăn xếp ra, xếp vào postfix và đẩy toán tử đang xét vào ngăn xếp.
- Nếu toán tử đang xét có bậc nhỏ hơn toán tử ở đỉnh ngăn xếp thì lấy toán tử đang xét và xếp vào postfix.

Bước 4: Thực hiện bước 3 cho đến khi kết thúc biểu thức và tất cả các toán tử toán hạng được xếp vào postfix, khi đó ta có biểu thức hậu tố.

b) Tính giá trị biểu thức

Bước 1: Quét toàn bộ biểu thức từ trái sang phải.

Bước 2: Tạo 1 ngăn xếp mới.

Bước 3: Nếu phần tử được quét là toán hạng thì đưa vào ngăn xếp.

Bước 4: Nếu phần tử được quét là toán tử thì lấy 2 toán hạng trong ngăn xếp ra, sau đó tính toán giá trị của chúng dựa vào toán tử này, sau đó đẩy lại vào ngăn xếp.

Bước 5: Thực hiện bước 3 và bước 4 cho đến khi kết thúc biểu thức và trong ngăn xếp còn 1 giá trị duy nhất. Đó chính là giá trị của biểu thức.

1.2. Mã nguồn

```
#project 5: Chương trình nhập vào một biểu thức trung tố,  
in ra biểu thức hậu tố và giá trị biểu thức  
#-----
```

```

.data
    infix: .space 256
    postfix: .space 256
    operator: .space 256
    stack: .space 256
    endMsg: .asciiz "continue??"
    errorMsg: .asciiz "input not correct"
    startMsg: .asciiz "please enter infix\nNote: contain +
- * / % ()\nnumber from 00-99"
    prompt_postfix: .asciiz "postfix expression: "
    prompt_result: .asciiz "result: "
    prompt_infix: .asciiz "infix expression: "

```

```

.text
start:
# nhap vao bieu thuc trung to
    li $v0, 54
    la $a0, startMsg
    la $a1, infix
    la $a2, 256
    syscall
    beq $a1,-2,end          # if cancel then end
    beq $a1,-3,start        # if enter then start
# in bieu thuc trung to
    li $v0, 4
    la $a0, prompt_infix
    syscall
    li $v0, 4
    la $a0, infix
    syscall
    li $v0, 11
    li $a0, '\n'
    syscall
# khoi tao cac trang thai
    li $s7,0                # bien trang thai $s7

                                # trang thai "1" khi nhan
vao so (0 -> 99)

```

vao toan tu * / + - %	# trang thai "2" khi nhan
vao dau "("	# trang thai "3" khi nhan
vao dau ")"	# trang thai "4" khi nhan
li \$t9,0	# dem so chu so?
li \$t5,-1	# luu dinh cua offset
postfix	
li \$t6,-1	# luu dinh cua offset toan
tu	
la \$t1, infix	# load cac dia chi cua cac
offset	
la \$t2, postfix	
la \$t3, operator	
addi \$t1,\$t1,-1	# Set dia chi khoi tao
infix la -1	
# chuyen sang postfix	
scanInfix:	# For each moi ki tu trong
postfix	
# kiem tra dau vao	
addi \$t1, \$t1, 1	# tang vi tri con tro
infix len 1 don vi i = i + 1	
lb \$t4, 0(\$t1)	# lay gia tri cua con
tro infix hien tai	
beq \$t4, ' ', scanInfix	# neu la space tiep tục
scan	
beq \$t4, '\n', EOF	# Scan ket thuc pop tat
ca cac toan tu sang postfix	
beq \$t9, 0, digit1	# Neu trang thai la 0
=> co 1 chu so	
beq \$t9, 1, digit2	# Neu trang thai la 1
=> co 2 chu so	
beq \$t9, 2, digit3	# neu trang thai la 2
=> co 3 chu so	
continueScan:	
beq \$t4, '+', plusMinus	# kiem tra ki tu hien
tai \$t4	

```

    beq $t4, '-', plusMinus
    beq $t4, '*', multiplyDivideModulo
    beq $t4, '/', multiplyDivideModulo
    beq $t4, '%', multiplyDivideModulo
    beq $t4, '(', openBracket
    beq $t4, ')', closeBracket
wrongInput:                                # dau vao loi
    li $v0, 55
    la $a0, errorMsg
    li $a1, 2
    syscall
    j ask
finishScan:
# in bieu thuc infix
    # Print prompt:
    li $v0, 4
    la $a0, prompt_postfix
    syscall
    li $t6, -1                                # set gia tri infix
hien tai la $s6 = -1
printPostfix:
    addi $t6, $t6, 1                            # tang offset cua
postfix hien tai
    add $t8, $t2, $t6                            # load dia chi cua
postfix hien tai
    lbu $t7, ($t8)                                # Load gia tri cua
postfix hien tai
    bgt $t6, $t5, finishPrint                    # in ra postfix xong
roi tinh ket qua
    bgt $t7, 99, printOperator                    # neu postfix hien tai
> 99 --> la mot toan tu
    # Neu khong thi la mot toan hang
    li $v0, 1
    add $a0, $t7, $zero
    syscall
    li $v0, 11
    li $a0, ' '
    syscall

```

```

j printPostfix                    # Loop
printOperator:
li $v0, 11
addi $t7,$t7,-100                # Decode toan tu
add $a0,$t7,$zero
syscall
li $v0, 11
li $a0, ' '
syscall
j printPostfix                    # Loop
finishPrint:
li $v0, 11
li $a0, '\n'
syscall
# tinh toan ket qua
li $t9,-4                        # set offset cua dinh stack
la -4
la $t3,stack                     # Load dia chi dinh stack
li $t6,-1                        # Dat offset cua Postfix
hien tai la -1
CalculatorPost:
addi $t6,$t6,1                   # tang offset hien tai cua
Postfix
add $t8,$t2,$t6                  # Load dia chi cua postfix
hien tai
lbu $t7,($t8)                    # Load gia tri cua postfix
hien tai
bgt $t6,$t5,printResult          # tinh toan ket qua va
in ra
bgt $t7,99,calculate             # neu gia tri postfix
hien tai > 99 --> toan tu --> lay ra 2 toan hang va tinh
toan
# neu khong thi la toan hang
addi $t9,$t9,4                   # tang offset dinh stack
len
add $t4,$t3,$t9                  # tang dia chi cua dinh
stack
sw $t7, ($t4)                    # day so vao stack

```

```

j CalculatorPost                                # Loop
calculate:
    # Pop 1 so
    add $t4,$t3,$t9
    lw $t0,($t4)
    # pop so tiep theo
    addi $t9,$t9,-4
    add $t4,$t3,$t9
    lw $t1,($t4)
    # Decode toan tu
    beq $t7,143,plus
    beq $t7,145,minus
    beq $t7,142,multiply
    beq $t7,147,divide
    beq $t7, 137, modulo
    plus:
        add $t0,$t0,$t1        # tinh tong gia tri cua 2
con tro dang luu gia tri toan hang
        sw $t0,($t4)          # luu gia tri cua con tro
ra $t4
#         li $t0, 0                # Reset t0, t1
#         li $t1, 0
        j CalculatorPost
    minus:
        sub $t0, $t1,$t0
        sw $t0,($t4)
#         li $t0, 0                # Reset t0, t1
#         li $t1, 0
        j CalculatorPost
    multiply:
        mul $t0, $t1,$t0
        sw $t0,($t4)
#         li $t0, 0                # Reset t0, t1
#         li $t1, 0
        j CalculatorPost
    divide:
        div $t1, $t0
        mflo $t0

```



```

        sw $t0,($t4)
#        li $t0, 0                # Reset t0, t1
#        li $t1, 0
        j CalculatorPost
modulo:
        div $t1, $t0
        mfhi $t0
        sw $t0,($t4)
#        li $t0, 0                # Reset t0, t1
#        li $t1, 0
        j CalculatorPost
printResult:
        li $v0, 4
        la $a0, prompt_result
        syscall
        li $v0, 1
        lw $a0,($t4)                # load gia tri cua $t4 ra
con tro $t0
        syscall
        li $v0, 11
        li $a0, '\n'
        syscall
ask:                                # tiep tuc khong??
        li $v0, 50
        la $a0, endMsg
        syscall
        beq $a0,0,start
        beq $a0,2,ask
# End program
end:
        li $v0, 10
        syscall

# Sub program
EOF:
        beq $s7,2,wrongInput        # ket thuc khi gap toan
tu hoac dau ngoac mo
        beq $s7,3,wrongInput

```

```

    beq $t5,-1,wrongInput          # -1 thi khong co dau
vao
    j popAllOperatorInStack
digit1:
    beq $t4,'0',storeDigit1
    beq $t4,'1',storeDigit1
    beq $t4,'2',storeDigit1
    beq $t4,'3',storeDigit1
    beq $t4,'4',storeDigit1
    beq $t4,'5',storeDigit1
    beq $t4,'6',storeDigit1
    beq $t4,'7',storeDigit1
    beq $t4,'8',storeDigit1
    beq $t4,'9',storeDigit1
    j continueScan

digit2:
    beq $t4,'0',storeDigit2
    beq $t4,'1',storeDigit2
    beq $t4,'2',storeDigit2
    beq $t4,'3',storeDigit2
    beq $t4,'4',storeDigit2
    beq $t4,'5',storeDigit2
    beq $t4,'6',storeDigit2
    beq $t4,'7',storeDigit2
    beq $t4,'8',storeDigit2
    beq $t4,'9',storeDigit2
    # neu khong nhap vao chu so thu 2
    jal numberToPostfix
    j continueScan

digit3:
    # neu scan ra chu so thu 3 --> error
    beq $t4,'0',wrongInput
    beq $t4,'1',wrongInput
    beq $t4,'2',wrongInput
    beq $t4,'3',wrongInput
    beq $t4,'4',wrongInput
    beq $t4,'5',wrongInput

```

```

    beq $t4,'6',wrongInput
    beq $t4,'7',wrongInput
    beq $t4,'8',wrongInput
    beq $t4,'9',wrongInput
    # neu khong co chu so thu 3
    jal numberToPostfix
    j continueScan
plusMinus:                                # Input is + -
    beq $s7,2,wrongInput                  # Nhan toan tu sau toan
    tu hoac "("
    beq $s7,3,wrongInput
    beq $s7,0,wrongInput                  # nhan toan tu truoc
    bat ki so nao
    li $s7,2                              # Thay doi trang thai
    dau vao thanh 2
    continuePlusMinus:
    beq $t6,-1,inputOperatorToStack      # Khong co gi trong
    stack -> day vao
    add $t8,$t6,$t3                       # Load dia chi cua
    toan tu o dinh
    lb $t7,($t8)                          # Load byte gia tri
    cua toan tu o dinh
    beq $t7,'(',inputOperatorToStack      # neu dinh la ( -->
    day vao
    beq $t7,'+',equalPrecedence           # neu dinh la + - -
    -> day vao
    beq $t7,'-',equalPrecedence
    beq $t7,'*',lowerPrecedence           # neu dinh la * / %
    thi lay * / % ra roi day vao
    beq $t7,'/',lowerPrecedence
    beq $t7,'%',lowerPrecedence
multiplyDivideModulo:                    # dau vao la * / %
    beq $s7,2,wrongInput                  # Nhan toan tu sau
    toan tu hoac "("
    beq $s7,3,wrongInput
    beq $s7,0,wrongInput                  # Nhan toan tu
    truoc bat ki so nao

```

li \$s7,2	# Thay doi trang
thai dau vao thanh 2	
beq \$t6,-1,inputOperatorToStack	# Khong co gi trong
stack -> day vao	
add \$t8,\$t6,\$t3	# Load dia chi cua
toan tu o dinh	
lb \$t7,(\$t8)	# Load byte gia tri
cua toan tu o dinh	
beq \$t7,'(',inputOperatorToStack	# neu dinh la (-->
day vao	
beq \$t7,'+',inputOperatorToStack	# neu dinh la + - -
-> day vao	
beq \$t7,'-',inputOperatorToStack	
beq \$t7,'*',equalPrecedence	# neu dinh la * / %
day vao	
beq \$t7,'/',equalPrecedence	
beq \$t7,'%',equalPrecedence	
openBracket:	# dau vao la (
beq \$s7,1,wrongInput	# Nhan "(" sau mot
so hoac dau ")"	
beq \$s7,4,wrongInput	
li \$s7,3	# Thay doi trang
thai dau vao thanh 3	
j inputOperatorToStack	
closeBracket:	# dau vao la ")"
beq \$s7,2,wrongInput	# Nhan ")" sau mot
toan tu hoac toan tu	
beq \$s7,3,wrongInput	
li \$s7,4	# Thay doi trang
thai dau vao thanh 4	
add \$t8,\$t6,\$t3	# Load dia chi toan
tu dinh	
lb \$t7,(\$t8)	# Load gia tri cua
toan tu o dinh	
beq \$t7,'(',wrongInput	# Input bao gom (
khong co gi o giua --> error	
continueCloseBracket:	

```

    beq $t6,-1,wrongInput          # không tìm được
dau "(" --> error
    add $t8,$t6,$t3                # Load địa chỉ của
toan tu o dinh
    lb $t7,($t8)                   # Load giá trị của
toan tu o dinh
    beq $t7,'(',matchBracket       # Tìm ngoặc phù hợp
    jal PopOperatorToPostfix       # đẩy toán tử vào
dinh vào postfix
    j continueCloseBracket        # tiếp tục vòng lặp cho
den khi tìm được ngoặc phù hợp
equalPrecedence:                  # nhân + - vào dinh
stack la + - || nhân * / % vào dinh stack la * / %
    jal PopOperatorToPostfix       # lấy toán tử dinh
stack ra Postfix
    j inputOperatorToStack        # đẩy toán tử mới vào
stack
lowerPrecedence:                  # nhân + - vào dinh
stack * / %
    jal PopOperatorToPostfix       # lấy toán tử dinh
stack ra và đẩy vào postfix
    j continuePlusMinus           # tiếp tục vòng lặp
inputOperatorToStack:            # đẩy dấu vào cho toán tử
    add $t6,$t6,1                 # tăng offset của toán tử vào
dinh lên 1
    add $t8,$t6,$t3                # load địa chỉ của toán tử
o dinh
    sb $t4,($t8)                   # lưu toán tử nhập vào
stack
    j scanInfix
PopOperatorToPostfix:             # lấy toán tử o dinh và lưu
vào postfix
    addi $t5,$t5,1                 # tăng offset của toán tử vào
dinh stack lên 1
    add $t8,$t5,$t2                # load địa chỉ của toán tử
o dinh stack
    addi $t7,$t7,100               # mã hóa toán tử + 100
    sb $t7,($t8)                   # lưu toán tử vào postfix

```

```

    addi $t6,$t6,-1          # giam offset cua toan tu o
dinh stack di 1
    jr $ra
matchBracket:                # xoa cap dau ngoac
    addi $t6,$t6,-1          # giam offset cua toan tu o
dinh stack di 1
    j scanInfix
popAllOperatorInStack:       # lay het toan tu vao
postfix
    jal numberToPostfix
    beq $t6,-1,finishScan    # stack rong --> ket
thuc
    add $t8,$t6,$t3          # lay dia chi cua toan
tu o dinh stack
    lb $t7,($t8)             # lay gia tri cua toan
tu o dinh stack
    beq $t7,'(',wrongInput   # ngoac khong phu hop -
-> error
    beq $t7,')',wrongInput
    jal PopOperatorToPostfix
    j popAllOperatorInStack   # lap cho den
khi stack rong
storeDigit1:
    beq $s7,4,wrongInput     # nhan vao so sau ")"
    addi $s4,$t4,-48          # luu chu so dau tien
duoi dang so ma ascii cua chu so 0 la 48
    add $t9,$zero,1          # Thay doi trang thai
thanh 1
    li $s7,1
    j scanInfix
storeDigit2:
    beq $s7,4,wrongInput     # nhan vao so sau ")"
    addi $s5,$t4,-48          # luu chu so thu hai
duoi dang so
    mul $s4,$s4,10
    add $s4,$s4,$s5           # luu number = first
digit * 10 + second digit

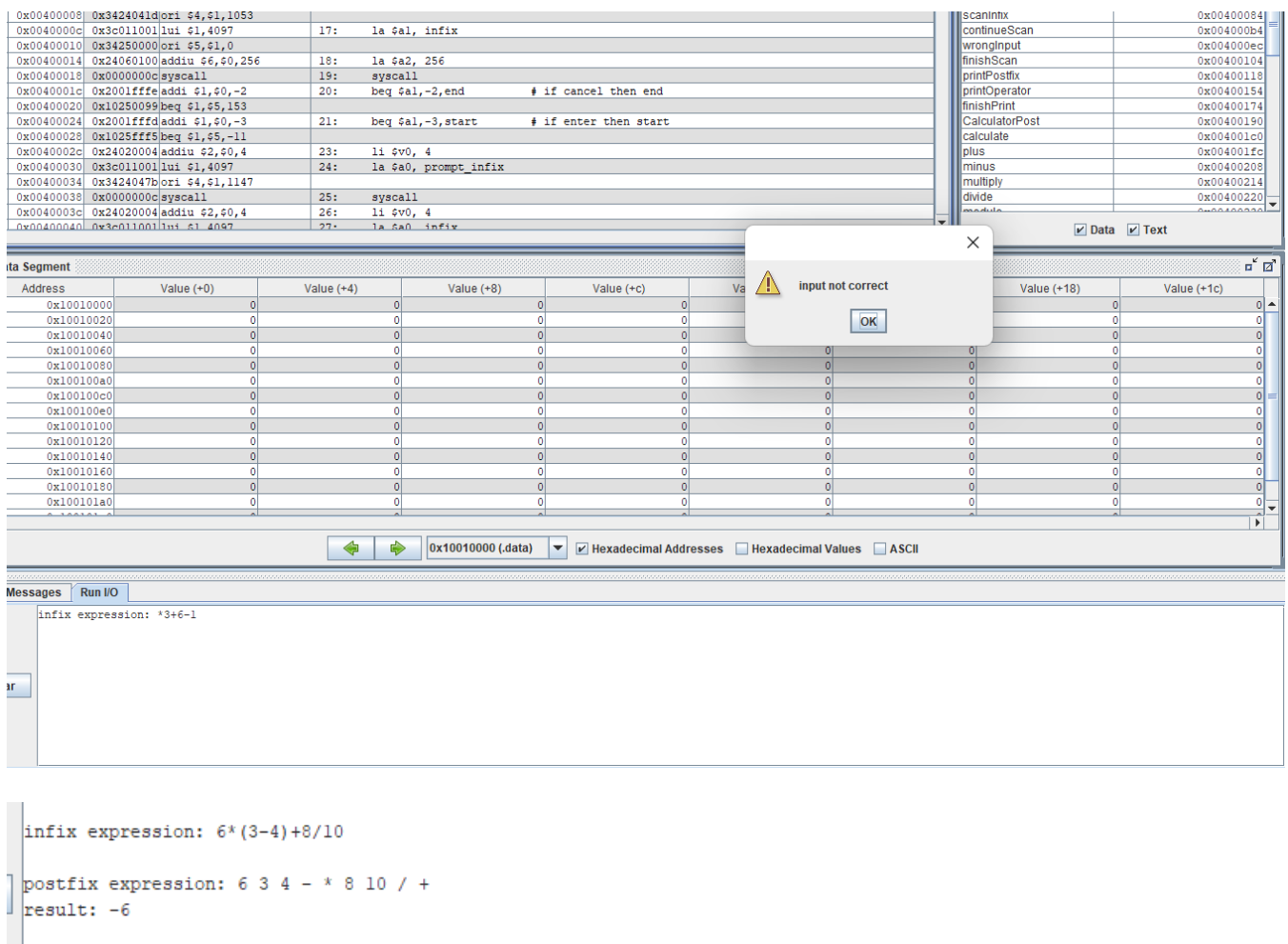
```

```

        add $t9,$zero,2                                # thay doi trang thai
thanh 2
        li $s7,1
        j scanInfix
numberToPostfix:
        beq $t9,0,endnumberToPostfix
        addi $t5,$t5,1
        add $t8,$t5,$t2
        sb $s4,($t8)                                # luu so vao postfix
        li $t9, 0                                    # thay doi trang thai ve 0
        endnumberToPostfix:
        jr $ra

```

1.3. Kết quả chạy



The screenshot shows a debugger window with assembly code on the left, a memory dump in the middle, and a list of functions on the right. A message box with the text "input not correct" is overlaid on the memory dump. Below the debugger, there is a text area showing the following text:

```

infix expression: *3+6-1

infix expression: 6*(3-4)+8/10
postfix expression: 6 3 4 - * 8 10 / +
result: -6

```

2. Project 7

Sinh viên thực hiện: Lê Đình Tuyên - 20194715

2.1. Phân tích cách thực hiện, thuật toán

*Phân tích cách thực hiện:

- Đầu vào: Câu lệnh chương trình MIPS
- Đầu ra: Kiểm tra xem lệnh đã nhập vào có đúng lệnh MIPS hay không?

*Ý tưởng:

- Xây dựng 1 Library có chứa cấu trúc của các lệnh hợp ngữ (Cấu trúc của Library gồm: opcode - toán hạng).
- Các ký hiệu trong Library: 1 - thanh ghi; 2 - hằng số nguyên; 3 - định danh; 4 - dành cho các lệnh lw, sw,... có cấu trúc ví dụ như imm(\$rs); 0 - không có.
- Tạo các khu vực chứa "character", "number", "register" để thực hiện kiểm tra.
- Duyệt câu lệnh nhập vào từ trái sang phải. Nếu opcode đúng, duyệt tiếp tới các toán hạng. Nếu Opcode sai thì thông báo sai (tương tự với các toán hạng cũng duyệt dần như vậy).

*Thuật toán:

Bước 1: Menu thực hiện nhập lệnh hoặc thoát.

- Nhập lệnh MIPS từ bàn phím.

Bước 2: Kiểm tra câu lệnh:

- 1, Kiểm tra opcode (add, and, or,...).
- 2, Sau khi kiểm tra, opcode kiểm tra dần tới các toán hạng. Nếu toán hạng đó là '0' trong Library thì thực hiện kiểm tra đằng sau đó có thừa ký tự gì không.
- 3, Kiểm tra giữa 2 toán hạng cần phải có dấu ','.
- 4, Khi kiểm tra các toán hạng cần xem trong Library toán hạng đó là thanh ghi, số nguyên, định danh hay imm(\$rs). Rồi thực hiện đi đến so sánh từng giá trị này.

Bước 3: Kiểm duyệt thành công in thông báo lệnh đúng cấu trúc và ngược lại.

Sau đó quay trở lại Menu.

2.2. Mã nguồn

```
# Project 7: Chương trình kiểm tra cú pháp lệnh Mips
#-----

.data
    menu_mess:      .ascii "\n----- MENU ----- \n1. Kiểm tra
cú pháp lệnh \n2. Thoát \nChọn: "
    menu_error_mess: .ascii "\nNhập sai, vui lòng nhập
lại! \n"
    input_mess:     .ascii "\nNhập vào lệnh Mips: "

    opcode_mess:    .ascii "Opcode: "
    toanHang_mess:  .ascii "Toán hạng: "
    hopLe_mess:     .ascii " - hop le. \n"
```



```

    error_mess:      .asciiiz "\nLenh hop ngu khong hop le, sai
khuon dang lenh !\n"
    completed_mess: .asciiiz "\nLenh hop ngu chinh xac !\n"
    command:        .space 100 # Luu cau lenh
    opcode:         .space 30  # Luu ma lenh, vi du: add, and,...
    ident:          .space 30  # nhan | hoac number
    token:          .space 30  # cac thanh ghi, vi du: $zero,
$at,...

    # Cau truc cua library:
    # opcode (7) - operation (3)
    # Trong so luong operation: 1 - thanh ghi; 2 - hang so
nguyen; 3 - dinh danh (ident); 4 - imm($rs); 0 - khong co
    library:        .asciiiz
"add***111;sub***111;addi***112;addu***111;addiu**112;subu**
*111;mfc0***110;mult***110;multu**110;div***110;mfhi***100;mf
lo***100;and***111;or*****111;andi***112;ori***112;sll***11
2;srl***112;lw*****140;sw*****140;lbu*****140;sb*****140;lui**
**120;beq***113;bne***113;slt***111;slti***112;sltiu**112;j
*****300;jal***300;jr*****100;nop***000"
    numberGroup:    .asciiiz "0123456789-"
    characterGroup: .asciiiz
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ_"
    # Moi thanh ghi cach nhau 6 byte
    tokenRegisters: .asciiiz "$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   $6   $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
main:
# ----- MENU -----
m_menu_start:
    li $v0, 4
    la $a0, menu_mess

```

```

syscall

# Read number input menu
li $v0, 5
syscall

beq $v0, 2, end_main          # 2: ket thuc
beq $v0, 1, m_menu_end        # 1: thuc hien kiem tra

li $v0, 4
la $a0, menu_error_mess       # Nhap sai
syscall

j m_menu_start
m_menu_end:

# ----- READ INPUT -----
m_input:
    jal input
    nop

# ----- START CHECK -----

m_check:
    jal check
    nop

    j m_menu_start

end_main:
    li $v0, 10
    syscall

#-----
# 1. @input: Nhap vao lenh Mips tu ban phim
#-----
input:
    li $v0, 4

```

```

    la $a0, input_mess
    syscall

    li $v0, 8
    la $a0, command
    li $a1, 100
    syscall

    jr $ra

#-----
# 2. @check: Kiem tra cau lenh
# - Buoc 1: Kiem tra opcode (add, and, or,...) ten lenh
# - Buoc 2: Kiem tra Operand lan luot cac operand (Toan hang)
# - Giua 2 toan hang can kiem tra xem co dau ',' hay khong.
# $s7: Luu index cua command
# $s3: Vi tri cua tung toan hang trong Library
#-----
check:
    # Luu $ra de tro ve main
    addi $sp, $sp, -4
    sw    $ra, 0($sp)

    addi $s7, $zero, 0      # Thanh ghi $s7 luu index cua
command

    # START CHECK OPCODE
    jal check_opcode
    nop

    # START CHECK OPERAND 1
    li    $s3, 7            # Vi tri operand trong Library
    jal check_operand
    nop

    # START CHECK OPERAND 2      # Neu khong co dau ',' ngay
cach giua operand_1 va operand_2 => FALSE
    li    $s3, 8            # Vi tri operand trong Library

```

```

    add $t0, $s5, $s3
    lb  $t0, 0($t0)
    beq $t0, 48, check_none      # Kiem tra neu operand = 0 ->
ket thuc; ky tu 0 trong ASCII

    la  $a0, command
    add $t0, $a0, $s7           # tro toi vi tri tiep tuc cua
command
    lb  $t1, 0($t0)
    bne $t1, 44, not_found      # Dau ', '
    add $s7, $s7, 1

    jal check_operand
    nop

    # START CHECK OPERAND 3      # Neu khong co dau ', ' ngan
cach giua operand_1 va operand_2 => FALSE
    li  $s3, 9                  # Vi tri operand trong Library
    add $t0, $s5, $s3
    lb  $t0, 0($t0)
    beq $t0, 48, check_none      # Kiem tra neu operand = 0 ->
ket thuc; ky tu 0 trong ASCII

    la  $a0, command
    add $t0, $a0, $s7           # tro toi vi tri tiep tuc cua
command
    lb  $t1, 0($t0)
    bne $t1, 44, not_found      # Dau ', '
    add $s7, $s7, 1

    jal check_operand
    nop

    # KIEM TRA KY TU THUA
    j  check_none

    # Tra lai $ra de tro ve main
    lw  $ra, 0($sp)

```

```

    addi $sp, $sp, 4
    jr   $ra

#-----
# 2.1 @check_opcode: Kiem tra cau lenh
# - Buoc 1: Lay cac opcode trong command da nhap
#           Xoa cac dau cach thua phia truoc
# - Buoc 2: So sanh voi trong bo tu dien xem co opcode do
#           khong
#           - Neu khong co ket thuc va quay lai menu
#           - Meu co, luu lai dia chi opcode trong library va tiep
#           tuc kiem tra
# $a0: command
# $a1: opcode
# $s7: index of command
# $t9: index of opcode
#-----

check_opcode:
    la    $a0, command                # Dia chi cua command
    la    $a1, opcode                # Dia chi cua opcode
    li    $t0, 0

remove_space_command:                # Xoa cac dau cach
    phia truoc lenh
    add $t1, $a0, $t0
    lb  $t2, 0($t1)
    bne $t2, 32, end_remove_space_command    # Neu khong
    phai ' ' -> Ket thuc
    addi $t0, $t0, 1
    j remove_space_command
end_remove_space_command:

    li    $t9, 0                    # index for opcode
    li    $s6, 0                    # so luong cac ki tu cua
opcode = 0
read_opcode:
    add $t1, $a0, $t0                # Dich bit cua command
    add $t2, $a1, $t9                # Dich bit cua opcode

```

```

    lb $t3, 0($t1)

    beq $t3, 32, read_opcode_done      # Neu co dau cach
    ' ' ket thuc read opcode
    beq $t3, 10, read_opcode_done     # Neu dau '\n' ket
    thuc read opcode
    beq $t3, 0, read_opcode_done      # Ket thuc chuoai

    sb $t3, 0($t2)
    addi $t9, $t9, 1
    addi $t0, $t0, 1
    j read_opcode
read_opcode_done:

    addi $s6, $t9, 0                  # $s6: So luong ki tu cua
    opcode
    add $s7, $s7, $t0                 # luu index cua command
    la $a2, library
    li $t0, -11

check_opcode_inlib:
    addi $t0, $t0, 11                 # Buoc nay bang 10 de
    nay den tung Instruction
    li $t1, 0                          # i = 0
    li $t2, 0                          # j = 0
    add $t1, $t1, $t0                 # Cong buoc nay

    compare_opcode:
        add $t3, $a2, $t1             # t3 tro thanh vi tri tro
    den dau cua tung Instruction
        lb $t4, 0($t3)
        beq $t4, 0, not_found
        beq $t4, 42, check_len_opcode # Neu gap ky tu
    `*` => Kiem tra do dai
        add $t5, $a1, $t2             # Load opcode
        lb $t6, 0($t5)
        bne $t4, $t6, check_opcode_inlib # So sanh 2 ki tu,
    neu khong bang nhau thi tinh den Instruction tiep theo.

```

```

        addi $t1, $t1, 1          # i = i + 1
        addi $t2, $t2, 1          # j = j + 1
        j compare_opcode
check_len_opcode:
        bne $t2, $s6, check_opcode_inlib
end_check_opcode_inlib:

        add $s5, $t0, $a2          # Luu lai vi tri
Instruction trong Library.

```

```

# ----- In thong tin ra man hinh -----
li $v0, 4
la $a0, opcode_mess
syscall

```

```

la $a3, opcode
li $t0, 0
print_opcode:
        beq $t0, $t9, end_print_opcode
        add $t1, $a3, $t0
        lb  $t2, 0($t1)
        li $v0, 11
        add $a0, $t2, $zero
        syscall
        addi $t0, $t0, 1
        j print_opcode
end_print_opcode:

```

```

li $v0, 4
la $a0, hopLe_mess
syscall

```

```

jr $ra

```

```

#-----
# 2.2 @check_operand:
# $a0: command.

```

```

# $s7: Luu index cua command.
# $s5: vi tri cua instruction trong library.
# $t9: Gia tri cua toan hang trong Library.
#-----

check_operand:
    # Luu $ra de tro ve check_operand
    addi $sp, $sp, -4
    sw    $ra, 0($sp)

    add $t9, $s5, $s3                # Tro toi operand trong
Library                               Library
    lb    $t9, 0($t9)
    addi $t9, $t9, -48                # Char -> Number

    la    $a0, command
    add $t0, $a0, $s7

    li $t1, 0                        # i = 0
    space_remove:                    # Xoa cac khoang trang thua
        add $t2, $t0, $t1
        lb    $t2, 0($t2)            # Lay ky tu tiep theo
        bne $t2, 32, end_space_remove # Ky tu ' '
        addi $t1, $t1, 1            # i = i + 1
        j    space_remove
    end_space_remove:

    add $s7, $s7, $t1                # Cap nhat lai index command

    li $s2, 0                        # Tat kich hoat check
number_register
    li $t8, 0                        # Khong co
    beq $t8, $t9, check_none
    li $t8, 1                        # Thanh ghi
    beq $t8, $t9, go_register
    li $t8, 2                        # So hang nguyen
    beq $t8, $t9, go_number
    li $t8, 3                        # Ident

```



```

    beq $t8, $t9, go_ident
    li $t8, 4 # Check number & register
    beq $t8, $t9, go_number_register

end_check_operand:
    # Tra lai $ra de tro ve check_operand
    lw $ra, 0($sp)
    addi $sp, $sp, 4
    jr $ra

#-----
# jal toi cac ham check de kiem tra
#-----

    go_register: # Check register
        jal check_register
        nop
    j end_check_operand

    go_number: # Check number
        la $a2, numberGroup
        jal check_ident
        nop
    j end_check_operand

    go_ident: # Check Ident
        la $a2, characterGroup
        jal check_ident
        nop
    j end_check_operand

    go_number_register: # Check number-register
        jal check_number_register
        nop
    j end_check_operand

#-----
# @check_none: Kiem tra xem con ky tu nao o cuoi khong
#-----

```

```

check_none:
    la $a0, command
    add $t0, $a0, $s7

    lb $t1, 0($t0)

    beq $t1, 10, none_ok    # Ky tu '\n'
    beq $t1, 0, none_ok    # Ket thuc chuoì

    j not_found

none_ok:
    li $v0, 4
    la $a0, completed_mess
    syscall
    j m_menu_start

#-----
# @check_register: Kiem tra xem register co hop le hay khong
# $a0: command (vi tri luu command)
# $a1: token (vi tri luu thanh ghi)
# $a2: tokenRegisters
# $s7: Luu index cua command
# $t9: index cua token
#-----

check_register:
    la $a0, command
    la $a1, token
    la $a2, tokenRegisters
    add $t0, $a0, $s7                # Tro den vi tri cac
instruction

    li $t1, 0                        # i = 0
    li $t9, 0                        # index cua token

read_token_register:
    add $t2, $t0, $t1                # command

```

```

add $t3, $a1, $t1          # token
lb $t4, 0($t2)

beq $t4, 41, end_read_token # Gap ky tu ')'
beq $t4, 44, end_read_token # Gap ky tu ', '
beq $t4, 10, end_read_token # Gap ky tu '\n'
beq $t4, 0, end_read_token  # Ket thuc

addi $t1, $t1, 1
beq $t4, 32, read_token_register # Neu gap dau ' '
thi tiep tuc

sb $t4, 0($t3)
addi $t9, $t9, 1
j read_token_register

end_read_token:
    add $s7, $s7, $t1          # Cap nhat lai gia tri
index

    li $t0, -6
compare_token_register:
    addi $t0, $t0, 6          # Buoc nhay bang 6 de nhay
den tung Register

    li $t1, 0                 # i = 0
    li $t2, 0                 # j = 0

    add $t1, $t1, $t0          # Cong buoc nhay

compare_reg:
    add $t3, $a2, $t1          # t3 tro thanh vi tri tro
den dau cua tung Register
    lb $t4, 0($t3)
    beq $t4, 0, not_found
    beq $t4, 32, check_len_reg # Neu gap ky tu `` =>
Kiem tra do dai

```

```

    add $t5, $a1, $t2          # Load token
    lb  $t6, 0($t5)

    bne $t4, $t6, compare_token_register    # So sanh 2 ki
tu, neu khong bang nhau thi tinh den Register tiep theo.
    addi $t1, $t1, 1          # i = i + 1
    addi $t2, $t2, 1          # j = j + 1
    j compare_reg

check_len_reg:
    bne $t2, $t9, compare_token_register    # Neu do dai
khong bang nhau di den register tiep theo

end_compare_token_register:

# >>>>>>>> In thong tin ra man hinh <<<<<<<<<
beq $s2, 1, on_token_number_register
li $v0, 4
la $a0, toanHang_mess
syscall

la $a3, token
li $t0, 0
print_token_register:
    beq $t0, $t9, end_print_token_register
    add $t1, $a3, $t0
    lb  $t2, 0($t1)
    li $v0, 11
    add $a0, $t2, $zero
    syscall
    addi $t0, $t0, 1
    j print_token_register
end_print_token_register:

li $v0, 4
la $a0, hopLe_mess
syscall
jr $ra

```

on_token_number_register:

```
    la $a3, token
    li $t0, 0
    print_on_token_register:
        beq $t0, $t9, end_print_on_token_register
        add $t1, $a3, $t0
        lb $t2, 0($t1)
        li $v0, 11
        add $a0, $t2, $zero
        syscall
        addi $t0, $t0, 1
        j print_on_token_register
    end_print_on_token_register:
```

```
    li $v0, 11
    li $a0, 41
    syscall
    li $v0, 4
    la $a0, hopLe_mess
    syscall
    jr $ra
```

```
#-----
# @check_ident: Kiem tra ident (label) HOAC number
# $a0: command (vi tri luu command)
# $a1: ident (vi tri luu ident)
# $a2: characterGroup | numberGroup
# $s7: luu index cua command
# $t9: index cua ident
#-----
```

check_ident:

```
    la $a0, command
    la $a1, ident
```

```
    add $t0, $a0, $s7                # Tro den vi tri cac
```

instruction

```

    li $t1, 0                # i = 0
    li $t9, 0                # index của ident

read_ident:
    add $t2, $t0, $t1        # command
    add $t3, $a1, $t1        # ident
    lb $t4, 0($t2)

    beq $t4, 40, end_read_ident # Gap ky tu '('
    beq $t4, 44, end_read_ident # Gap ky tu ', '
    beq $t4, 10, end_read_ident # Gap ky tu '\n'
    beq $t4, 0, end_read_ident  # Ket thuc

    addi $t1, $t1, 1
    beq $t4, 32, read_ident      # Neu gap dau ' ' thi tiep
tuc

    sb $t4, 0($t3)
    addi $t9, $t9, 1
    j read_ident

end_read_ident:
    add $s7, $s7, $t1          # Cap nhat lai gia tri index
    beq $t9, 0, not_found      # Khong co label

    #li $v0, 10
    #syscall

    li $t2, 0                  # index cho Ident
compare_ident:
    beq $t2, $t9, end_compare_ident # ket thuc chuoï
    li $t1, 0                  # index cho characterGroup

    add $t3, $a1, $t2
    lb $t3, 0($t3)             # Tung char trong Ident

```

```

    loop_Group:                                # Kiem tra tung ky tu Ident co
trong Group hay khong
    add $t4, $a2, $t1
    lb $t4, 0($t4)
    beq $t4, 0, not_found                    # Khong co -> Khong tim
thay
    beq $t4, $t3, end_loop_Group

    addi $t1, $t1, 1
    j loop_Group

end_loop_Group:

    addi $t2, $t2, 1

    j compare_ident

end_compare_ident:

    beq $s2, 1, on_number_register

    # ----- In thong tin ra man hinh -----
    li $v0, 4
    la $a0, toanHang_mess
    syscall

    la $a3, ident
    li $t0, 0
    print_ident:
        beq $t0, $t9, end_print_ident
        add $t1, $a3, $t0
        lb $t2, 0($t1)
        li $v0, 11
        add $a0, $t2, $zero
        syscall
        addi $t0, $t0, 1
        j print_ident
    end_print_ident:

```

```

    li $v0, 4
    la $a0, hopLe_mess
    syscall
    jr $ra

```

on_number_register:

```

    li $v0, 4
    la $a0, toanHang_mess
    syscall

```

```

    la $a3, ident

```

```

    li $t0, 0

```

print_on_ident:

```

    beq $t0, $t9, end_print_on_ident

```

```

    add $t1, $a3, $t0

```

```

    lb $t2, 0($t1)

```

```

    li $v0, 11

```

```

    add $a0, $t2, $zero

```

```

    syscall

```

```

    addi $t0, $t0, 1

```

```

    j print_on_ident

```

end_print_on_ident:

```

    li $v0, 11

```

```

    li $a0, 40

```

```

    syscall

```

```

    jr $ra

```

```

#-----
# @check_number_register: Kiem tra number - ident
# $a0: command (vi tri luu command)
# $s7: luu index cua command
# $s2: Luu kich hoat check number register
#-----

```

check_number_register:

```

    # Luu $ra de tro ve

```



```

addi $sp, $sp, -4
sw   $ra, 0($sp)

li   $s2, 1                # Bat kich hoat number_register

# Check number
la   $a2, numberGroup
jal  check_ident
nop

la   $a0, command
add  $t0, $a0, $s7         # Tro den vi tri cac
instruction
lb   $t0, 0($t0)
bne  $t0, 40, not_found    # Neu ki tu khong phai la
dau '('
addi $s7, $s7, 1

# Check register
jal  check_register
nop
la   $a0, command
add  $t0, $a0, $s7         # Tro den vi tri cac
instruction
lb   $t0, 0($t0)
bne  $t0, 41, not_found    # Neu ki tu khong phai la
dau ')'
addi $s7, $s7, 1

# Tra lai $ra de tro ve
lw   $ra, 0($sp)
addi $sp, $sp, 4
jr   $ra

#-----
# @not_found: Khong tim thay khon dang lenh
#-----
not_found:

```

```
li $v0, 4
la $a0, error_mess
syscall
j m_menu_start
```

```
#-----
#  END
#-----
```

2.3. Kết quả chạy

```
----- MENU -----
1. Kiem tra cu phap lenh
2. Thoat
Chon: 1

Nhap vao lenh Mips: add $5, $6, $7
Opcode: add - hop le.
Ioan hang: $5 - hop le.
Ioan hang: $6 - hop le.
Ioan hang: $7 - hop le.

Lenh hop ngu chinh xac !

----- MENU -----
1. Kiem tra cu phap lenh
2. Thoat
Chon: 1

Nhap vao lenh Mips: j loop
Opcode: j - hop le.
Ioan hang: loop - hop le.

Lenh hop ngu chinh xac !

----- MENU -----
1. Kiem tra cu phap lenh
2. Thoat
Chon: 1

Nhap vao lenh Mips: addi $t3, $t4, -100
Opcode: addi - hop le.
Ioan hang: $t3 - hop le.
Ioan hang: $t4 - hop le.
Ioan hang: -100 - hop le.

Lenh hop ngu chinh xac !

----- MENU -----
1. Kiem tra cu phap lenh
2. Thoat
Chon: 2

-- program is finished running --
```

Hết