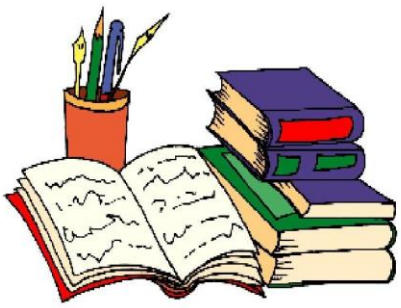
ĐẠI HỌC BÁCH KHOA HÀ NỘI

BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH

GIẢNG VIÊN HƯỚNG DẪN : **Ths. LÊ BÁ VUI**

SINH VIÊN : **HOÀNG ĐỨC ANH** **MSSV: 20176688**

VŨ MINH HOÀNG ANH **MSSV: 20176689**



Hà Nội, Ngày 21 Tháng 6 Năm 2020

Mục Lục

Bài 4: Postscript CNC Marsbot

Sinh viên thực hiện: Vũ Minh Hoàng Anh - 20176689

A. **Đề bài**: Máy gia công cơ khí chính xác CNC Marsbot được dùng để cắt tấm kim loại theo các đường nét được qui định trước. CNC Marsbot có một lưỡi cắt dịch chuyển trên tấm kim loại, với giả định rằng:

- Nếu lưỡi cắt dịch chuyển nhưng không cắt tấm kim loại, tức là Marsbot di chuyển nhưng không để lại vết (Track)
- Nếu lưỡi cắt dịch chuyển và cắt tấm kim loại, tức là Marsbot di chuyển và có để lại vết.

Để điều khiển Marsbot cắt đúng như hình dạng mong muốn, người ta nạp vào Marsbot một mảng cấu trúc gồm 3 phần tử:

- <Góc chuyển động>, <Thời gian>, <Cắt/Không cắt>

- Trong đó <Góc chuyển động> là góc của hàm HEADING của Marsbot

- <Thời gian> là thời gian duy trì quá trình vận hành hiện tại

- <Cắt/Không cắt> thiết lập lưu vết/không lưu vết

Hãy lập trình để CNC Marsbot có thể:

- Thực hiện cắt kim loại như đã mô tả
- Nội dung postscript được lưu trữ cố định bên trong mã nguồn
- Mã nguồn chứa 3 postscript và người dùng sử dụng 3 phím 0, 4, 8 trên bàn phím Key Matrix để chọn postscript nào sẽ được gia công.

Một postscript chứa chữ DCE cần gia công. Hai script còn lại sinh viên tự đề xuất (tối thiểu 10 đường cắt) .

B. Phân tích yêu cầu

- Cho sẵn một đoạn mã nguồn

- Trong một khoảng thời gian cố định, người dùng nhập số từ bàn phím Digital Lab Sim.

- Kết quả: Mars Bot chạy theo đúng postscript.

C. Cách làm

- Thiết kế postscript
- Tạo hàm để check các phím 0,4,8. Nếu nhập vào 0, 4 hoặc 8 thì sẽ tiến hành lưu địa chỉ của postscript tương ứng.
- Xử lí Mars Bot , đọc postscript, đọc góc chuyển động, thời gian, cắt< không cắt>.

D. Mã nguồn

```
# Mars bot

.eqv HEADING 0xffff8010                     #Integer: 1 goc giua 0-359

.eqv MOVING 0xffff8050                             #Boolean: co di chuyen hay khong di chuyen

.eqv LEAVETRACK 0xffff8020                  #Boolean (1/0): co hay khong de lai track

# .eqv thay gia tri bang bien

# Key matrix

.eqv OUT_ADDRESS_HEXA_KEYBOARD 0xFFFF0014

.eqv IN_ADDRESS_HEXA_KEYBOARD 0xFFFF0012

.data

#.asciiz Luu chuoi gia tri

# postscript-DCE => numpad 0

# (rotate,time,0=untrack | 1=track;)

pscript1: .asciiz
"90,3000,0;180,4000,0;180,5790,1;80,500,1;70,500,1;60,500,1;50,500,1;40,500,1;30,500,1;20,500,
1;10,500,1;0,500,1;350,500,1;340,500,1;330,500,1;320,500,1;310,500,1;300,500,1;290,500,1;280,4
90,1;90,7000,0;270,300,1;260,500,1;250,500,1;240,500,1;230,500,1;220,500,1;210,500,1;200,500,1
;190,500,1;180,500,1;170,500,1;160,500,1;150,500,1;140,500,1;130,500,1;120,500,1;110,500,1;100
,500,1;90,800,1;90,5000,0;270,3000,1;0,5800,1;90,3000,1;180,2900,0;270,3000,1;90,3000,0;"

# postscript-HA => numpad 4

pscript2: .asciiz
"90,3000,0;180,5000,0;180,6000,1;0,3000,0;90,3000,1;0,3000,0;180,6000,1;90,1500,0;20,6385,1;16
0,6395,1;340,3198,0;270,2184,1;90,3000,0;180,3198,0;"

# postscript-DUY => numpad 8

pscript3: .asciiz
"90,2000,0;180,3000,0;180,5790,1;80,500,1;70,500,1;60,500,1;50,500,1;40,500,1;30,500,1;20,500,
1;10,500,1;0,500,1;350,500,1;340,500,1;330,500,1;320,500,1;310,500,1;300,500,1;290,500,1;280,5
00,1;90,5000,0;180,5500,1;90,3000,1;0,5500,1;90,3000,0;150,2500,1;30,2500,1;210,2500,0;180,310
0,1;180,1000,0;"

.text

# <--xu ly tren keymatrix-->

       li $t3, IN_ADDRESS_HEXA_KEYBOARD    #gan 2 cai t3, t4

       li $t4, OUT_ADDRESS_HEXA_KEYBOARD

polling:

       li $t5, 0x01                        # row-1 of key matrix
```

```
        sb $t5, 0($t3)
        lb $a0, 0($t4)
        bne $a0, 0x11, NOT_NUMPAD_0
        la $a1, pscript1
        j START
        NOT_NUMPAD_0:
        li $t5, 0x02                        # row-2 of key matrix
        sb $t5, 0($t3)
        lb $a0, 0($t4)
        bne $a0, 0x12, NOT_NUMPAD_4
        la $a1, pscript2
        j START
        NOT_NUMPAD_4:
        li $t5, 0X04                        # row-3 of key matrix
        sb $t5, 0($t3)
        lb $a0, 0($t4)
        bne $a0, 0x14, COME_BACK
        la $a1, pscript3
        j START
COME_BACK: j polling                        # khi cac so 0,4,8 khong duoc chon -> quay lai doc
tiep
# <!--end-->
# <--xu li mars bot -->
START:
        jal GO
READ_PSCRIPT:
        addi $t0, $zero, 0                  # luu gia tri rotate
        addi $t1, $zero, 0                  # luu gia tri time


        READ_ROTATE:
        add $t7, $a1, $t6                   # dich bit    #$a1: dia chi pscript
```

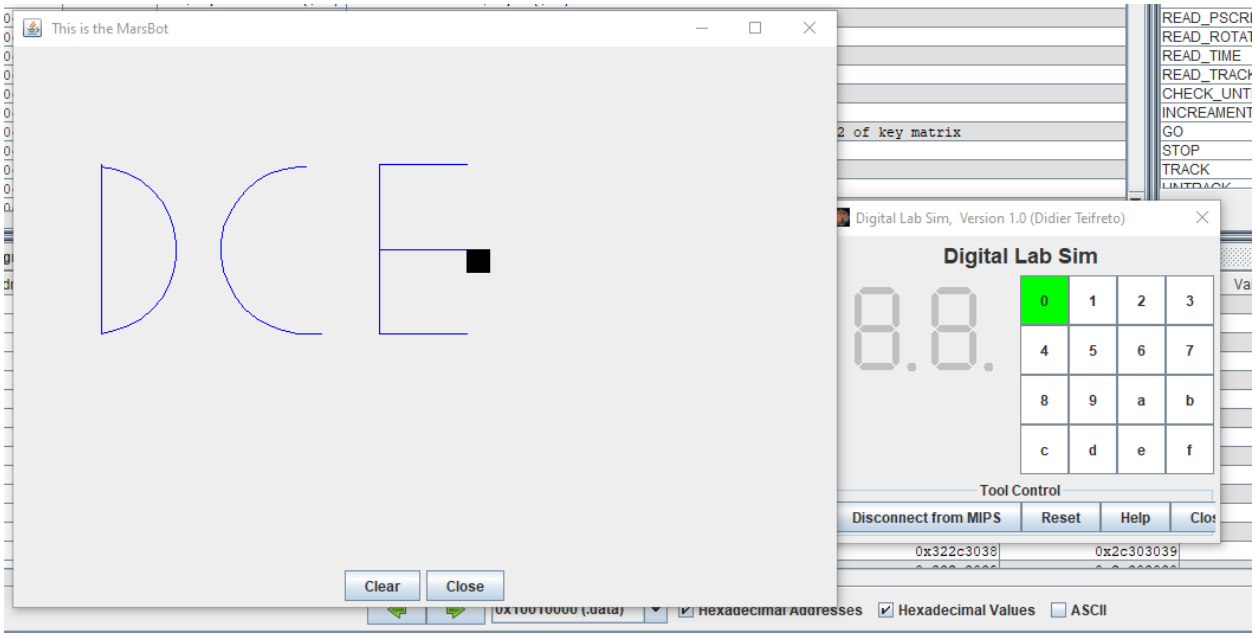
```
	lb $t5, 0($t7)			# doc cac ki tu cua pscript #$t5: gia tri so nhap
vao

	beq $t5, 0, END			# khong gap ki tu nao, ket thuc pscript

	beq $t5, 44, READ_TIME			# gap ki tu ','

	mul $t0, $t0, 10

	addi $t5, $t5, -48		# So 0 co thu tu 48 trong bang ascii.

	add $t0, $t0, $t5		# cong cac chu so lai voi nhau.

	addi $t6, $t6, 1		# tang so bit can dich chuyen len 1

	j READ_ROTATE			# quay lai doc tiep den khi gap dau ','


	READ_TIME:			# doc thoi gian chuyen dong.

	add $a0, $t0, $zero

	jal ROTATE

	addi $t6, $t6, 1

	add $t7, $a1, $t6		# ($a1 luu dia chi cua pscript)

	lb $t5, 0($t7)			# doc cac ki tu cua pscript #$t5: gia tri
so nhap vao

	beq $t5, 44, READ_TRACK			# gap ki tu ','

	mul $t1, $t1, 10

	addi $t5, $t5, -48		# So 0 co thu tu 48 trong bang ascii.

	add $t1, $t1, $t5		# cong cac chu so lai voi nhau.

	j READ_TIME			# quay lai doc tiep den khi gap dau ','


	READ_TRACK:

	addi $v0,$zero,32		# Keep mars bot running by sleeping with time=$t1

	add $a0, $zero, $t1

	addi $t6, $t6, 1

	add $t7, $a1, $t6		# ($a1 luu dia chi cua pscript)

	lb $t5, 0($t7)			# doc cac ki tu cua pscript #$t5: gia tri
so nhap vao

	addi $t5, $t5, -48		# So 0 co thu tu 48 trong bang ascii.

	beq $t5, $zero, CHECK_UNTRACK		# 1=track | 0=untrack
```

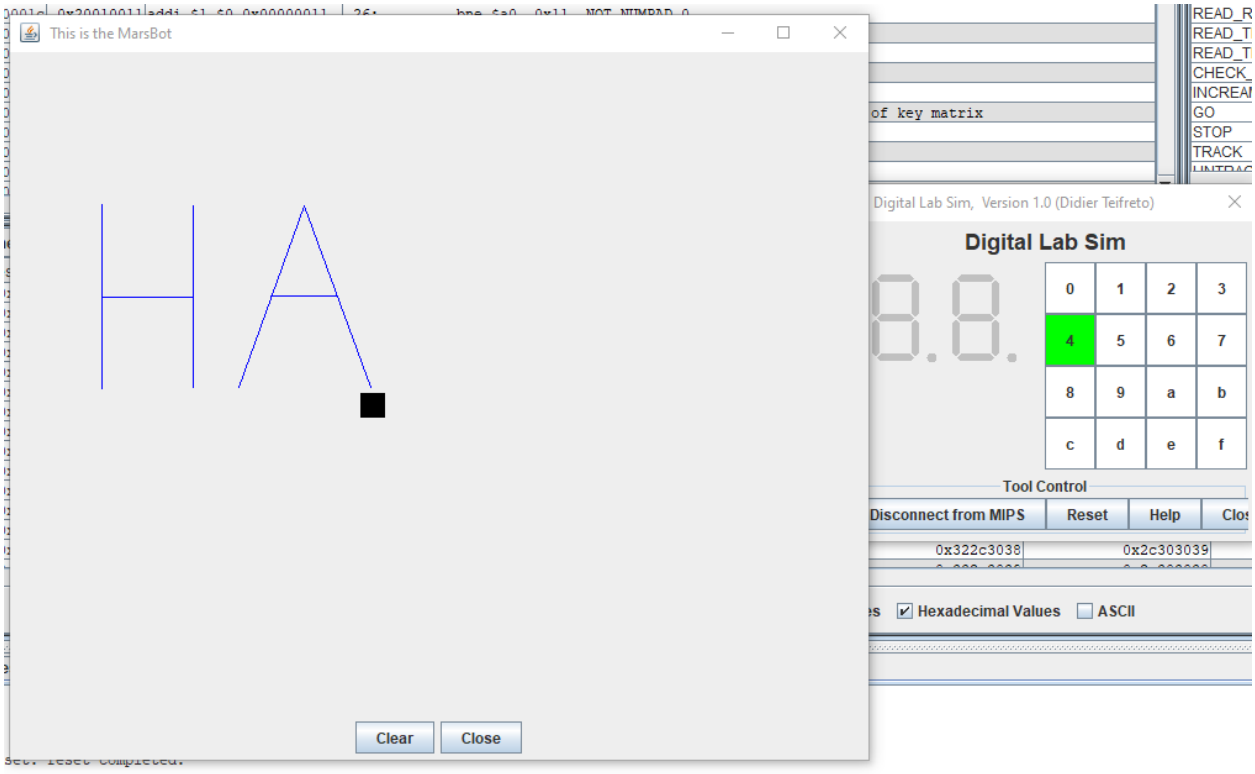
```
        jal UNTRACK
        jal TRACK
        j INCREAMENT

CHECK_UNTRACK:
        jal UNTRACK
INCREAMENT:
        syscall
        addi $t6, $t6, 2                    # bo qua dau ';'
        j READ_PSCRIPT
#-------------------------------------------------------------
# GO procedure, to start running
# param[in] none
#-------------------------------------------------------------
GO:
        li $at, MOVING              # change MOVING port
        addi $k0, $zero,1     # to logic 1,
        sb $k0, 0($at)              # to start running
        nop
        jr $ra
        nop
#-------------------------------------------------------------
# STOP procedure, to stop running
# param[in] none
#-------------------------------------------------------------
STOP:
        li $at, MOVING              # change MOVING port to 0
        sb $zero, 0($at)      # to stop
        nop
        jr $ra
```

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

```
ROTATE:
	li $at, HEADING		# change LEAVETRACK port to 0
	sw $a0, 0($at)		# to stop drawing tail
	nop
	jr $ra
	nop
END:
	jal STOP
	li $v0, 10
	syscall
	j polling
# <!—end
```

E. Hình ảnh kết quả mô phỏng





Bài 5: Biểu thức trung tố hậu tố

Sinh viên thực hiện: Hoàng Đức Anh - 20176688

A. 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 ởdạng hậu tố, ví dụ: 9 2 + 8 6* +
3. Tính ra giá trị của biểu thức vừa nhập

Các hằng số là sốnguyên, trong phạm vi từ 0 →99. Toán tử bao gồm phép cộng, trừ, nhân, chia lấy thương

B. Phân tích yêu cầu :

1. Nhập vào biểu thức trung tố
2. In ra biểu thức ởdạng hậu tố
3. Tính ra giá trị của biểu thức vừa nhập
4. Các hằng số là số nguyên, trong phạm vi từ 0 →99.
5. Toán tử bao gồm phép cộng, trừ, nhân, chia lấy thương (+, -, *, /) .

C. Cách làm

Mô tả thuật toán

Thuật toán sử dụng một Stack để chứa các toán tử và dấu ngoặc mở. Thủ tục Push(V) để đẩy một phần tử vào Stack, hàm Pop để lấy ra một phần tử từ Stack, hàm Get để đọc giá trị phần tử nằm ở đỉnh Stack mà không lấy phần tử đó ra. Mức độ ưu tiên của các toán tử được quy định bằng hàm Priority như sau:

+ Ưu tiên cao nhất là dấu "*" và dấu "/" với priority là 2
+ Dấu "+" và dấu "-" có độ ưu tiên thấp hơn dấu "*" và "/" với Priority là 1
+ Dấu "(" có độ ưu tiên thấp nhất với Priority là 0

Ví dụ: với biểu thức trung tố (2*3 + 7/8)*(5-1) ta có quá trình như sau

Đọc	Xử lý	Stack	Output
(	Đẩy vào Stack	(	
2	Hiển thị	(	2
*	phép "*" được ưu tiên hơn phần tử ở đỉnh Stack là "(", đẩy "*" vào Stack	(*	
3	Hiển thị	(*	2 3
+	phép "+" ưu tiên không cao hơn phần tử ở đỉnh Stack là "*", lấy ra và hiển thị "*". So sánh tiếp, thấy phép "+" được ưu tiên cao hơn phần tử ở đỉnh Stack là "(", đẩy "+" vào Stack	(+	2 3 *
7	Hiển thị	(+	2 3 * 7
/	phép "/" được ưu tiên hơn phần tử ở đỉnh Stack là "+", đẩy "/" vào Stack	(+/	
8	Hiển thị	(+/	2 3 * 7 8
)	Lấy ra và hiển thị các phần tử trong Stack tới khi lấy phải dấu ngoặc mở	ø	2 3 * 7 8 / +
*	Stac đang là rỗng, đẩy * vào Stack	*	
(	Đẩy vào Stack	*(	
5	Hiển thị	*(	2 3 * 7 8 / + 5
-	phép "-" được ưu tiên hơn phần tử ở đỉnh Stack là "(", đẩy "-" vào Stack	*(-	
1	Hiển thị	*(-	2 3 * 7 8 / + 5 1
)	Lấy ra và hiển thị các phần tử ở đỉnh Stack cho đến khi lấy phải dấu ngoặc mở	*	2 3 * 7 8 / + 5 1 -
Hết	Lấy ra và hiển thị hết các phần tử còn lại trong Stack		2 3 * 7 8 / + 5 1 - *

D. Mã nguồn:

```
.data
	infix: .space 256
	postfix: .space 256
	operator: .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: "
	converter: .word 1
	wordToConvert: .word 1
	stack: .float
.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"  nhan vao so (0 -> 99)
					# trang thai "2"  nhan vao toan tu(* / + -)
					# trang thai "3"  nhan vao dau "("
					# trang thai "4"  nhan vao dau ")"
```

```
	li $t9,0 # dem chu so?
	li $t5,-1 # Postfix top offset
	li $t6,-1 # Operator top offset
	la $t1, infix # Sau sau moi vong lap dia chi infix hien tai tang
len 1
	la $t2, postfix
	la $t3, operator
	addi $t1,$t1,-1 # Set dia chi khoi tao infix la -1
# chuyen sang postfix
scanInfix: # Loop for each moi ki tu trong postfix
# kiem tra dau vao
	addi $t1, $t1, 1 # tang vi tri con tro infix len 1 don vii =
i + 1
	lb $t4, ($t1) # lay gia tri cua con tro infix hien tai
	beq $t4, ' ', scanInfix # neu la space tiep tuc scan
	beq $t4, '\n', EOF # Scan ket thuc pop tat ca cac toan tu sang
postfix
	beq $t9, 0, digit1 # Neu trang thai là 0 => co 1 chu so
	beq $t9, 1, digit2 # Neu trang thai là 1 => co 2 chu so
	beq $t9, 2, digit3 # neu trang thai là 2 => co 3 chu so
	continueScan:
	beq $t4, '+', plusMinus
	beq $t4, '-', plusMinus
	beq $t4, '*', multiplyDivide
	beq $t4, '/', multiplyDivide
	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 là -1 $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 --> tính ket qua
	bgt $t7,99,printOp		# 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
	printOp:
	li $v0, 11
	addi $t7,$t7,-100		# Decode toán tu
	add $a0,$t7,$zero
	syscall
	li $v0, 11
	li $a0, ' '
	syscall
	j printPostfix			# Loop
finishPrint:
	li $v0, 11
	li $a0, '\n'
	syscall
# tính toán ket	qua
	li $t9,-4			# set offset cua dinh stack là -4
	la $t3,stack			# Load dia chi dinh stack
	li $t6,-1			# Load offset cua Postfix hien tai là -1
	l.s $f0,converter		# Load converter
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		# tính toán ket qua va in ra
	bgt $t7,99,calculate		# neu gia tri postfix hien tai > 99 --> toan tu --
> lay ra 2 toan hang va tính toán
	# 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,wordToConvert
	l.s $f10,wordToConvert		# Load so sang coproc1 roi lai chuyen sang
float
	div.s $f10,$f10,$f0
```

```
	s.s $f10,($t4)			# day so vào stack
	sub.s $f10,$f10,$f10		# Reset f10
	j CalculatorPost			# Loop
	calculate:
		# Pop 1 so
		add $t4,$t3,$t9
		l.s $f3,($t4)
		# pop so tiep theo
		addi $t9,$t9,-4
		add $t4,$t3,$t9
		l.s $f2,($t4)
		# Decode toán tu
		beq $t7,143,plus
		beq $t7,145,minus
		beq $t7,142,multiply
		beq $t7,147,divide
		plus:
			add.s $f1,$f2,$f3	# tinh tong gia tri cua 2 con tro dang luu gia tri
toan hang
			s.s $f1,($t4)		# luu gia tri cua con tro ra $t4
			sub.s $f2,$f2,$f2	# Reset f2 f3
			sub.s $f3,$f3,$f3
			j CalculatorPost
		minus:
			sub.s $f1,$f2,$f3
			s.s $f1,($t4)
			sub.s $f2,$f2,$f2	# Reset f2 f3
			sub.s $f3,$f3,$f3
			j CalculatorPost
		multiply:
			mul.s $f1,$f2,$f3
			s.s $f1,($t4)
			sub.s $f2,$f2,$f2	# Reset f2 f3
			sub.s $f3,$f3,$f3
			j CalculatorPost
		divide:
			div.s $f1,$f2,$f3
			s.s $f1,($t4)
			sub.s $f2,$f2,$f2	# Reset f2 f3
			sub.s $f3,$f3,$f3
			j CalculatorPost

printResult:
```

```
	li $v0, 4
	la $a0, prompt_result
	syscall
	li $v0, 2
	l.s $f12,($t4)			# load gia tri cua $t4 ra con tro $f12
	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, 55
	#la $a0, byeMsg
	#li $a1, 1
	#syscall
	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 thì không có 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 toán tu sau toán 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  vào thành 2
	continuePlusMinus:
	beq $t6,-1,inputOperatorToStack         # Không có gì trong  stack -> day vao
	add $t8,$t6,$t3                         # Load dia chi cua toan tu o dinh
	lb $t7,($t8)                    # Load byte giá tri cua toan tu o dinh
	beq $t7,'(',inputOperatorToStack        # neu dinh là ( --> day vào
	beq $t7,'+',equalPrecedence     # neu dinh là + - --> day vào
	beq $t7,'-',equalPrecedence
	beq $t7,'*',lowerPrecedence     # neu dinh là * / thi lay * / ra roi day vao
	beq $t7,'/',lowerPrecedence
```

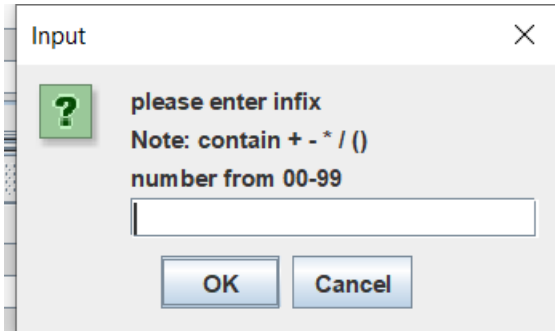
```
multiplyDivide:                                 # dau vào là * /
       beq $s7,2,wrongInput                     # Nhan toán tu sau toán tu hoac "("
       beq $s7,3,wrongInput
       beq $s7,0,wrongInput                     # Nhan toán tu truoc bat ki so nào
       li $s7,2                                 # Thay doi trang thai dau vào thành 2
       beq $t6,-1,inputOperatorToStack          # Không có gì trong  stack -> day vào
       add $t8,$t6,$t3                          # Load dia chi cua toan tu o dinh
       lb $t7,($t8)                             # Load bytegiá tri cua toan tu o dinh
       beq $t7,'(',inputOperatorToStack         # neu dinh là ( --> day vào
       beq $t7,'+',inputOperatorToStack         # neu dinh là + - --> day vào
       beq $t7,'-',inputOperatorToStack
       beq $t7,'*',equalPrecedence              # neu dinh là * / day vao
       beq $t7,'/',equalPrecedence
openBracket:                                    # dau vào là (
       beq $s7,1,wrongInput                     # Nhan "(" sau mot so hoac dau ")"
       beq $s7,4,wrongInput
       li $s7,3                                 # Thay doi trang thai dau vào thành 3
       j inputOperatorToStack
closeBracket:                                   # dau vao la  ")"
       beq $s7,2,wrongInput                     # Nhan ")" sau mot toán tu hoac toán tu
       beq $s7,3,wrongInput
       li $s7,4                                 # Thay doi trang thai dau vào thành 4
       add $t8,$t6,$t3                          # Load dia chi toán tu dinh
       lb $t7,($t8)                             # Load giá tri cua toan tu o dinh
       beq $t7,'(',wrongInput                   # Input bao gom () không có gì o giua  -->
error
       continueCloseBracket:
       beq $t6,-1,wrongInput                    # không tìm duoc dau "(" --> error
       add $t8,$t6,$t3                          # Load dia chi cua toan tu o dinh
       lb $t7,($t8)                             # Load gia tri cua toan tu o dinh
       beq $t7,'(',matchBracket                 # Tìm ngoac phu hop
       jal PopOpeatorToPostfix                  # day toan tu o dinh vào postfix
       j continueCloseBracket                   # tiep tuc vong lap cho den khi tim duoc
ngoac phu hop
equalPrecedence:                                #  nhan + - và dinh stack là + - || nhan *
/ và dinh stack là * /
       jal PopOpeatorToPostfix                  # lay toan tu dinh stack ra Postfix
       j inputOperatorToStack                   # day toan tu moi vào stack
lowerPrecedence:                                # nhan + - và dinh stack * /
       jal PopOpeatorToPostfix                  #  lay  toan  tu  dinh  stack  ra  va  day  vào
postfix
       j continuePlusMinus                      # tiep tuc vong lap
inputOperatorToStack:                           # day dau vào cho toán tu
```

```
        add $t6,$t6,1                   # tang offset cua toan tu o dinh lên 1
        add $t8,$t6,$t3                         # load dia chi cua  toán tu o dinh
        sb $t4,($t8)                    # luu toan tu nhap vao stack
        j scanInfix
PopOpeatorToPostfix:                    # lay toan tu o dinh va luu vào postfix
        addi $t5,$t5,1                  # tang offet cua toan tu o dinh stack lên 1
        add $t8,$t5,$t2                         # load dia chi cua toan tu o dinh stack
        addi $t7,$t7,100                        # mã hóa toán tu + 100
        sb $t7,($t8)                    # luu toan tu vào postfix
        addi $t6,$t6,-1                         # giam offset cua toan tu o dinh stack di 1
        jr $ra
matchBracket:                           # xóa 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 PopOpeatorToPostfix
        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
        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 thái thành 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
```

```
        add $t9,$zero,$zero                 # thay doi trang thai ve 0
        endnumberToPostfix:
        jr $ra
```

E. Kết quả hiển thi :

Nhập dữ liệu

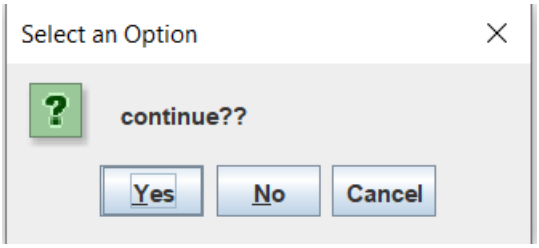


Input: 3+4*(10+99)

Kết quả:

```
infix expression: 3+4*(10+99)

postfix expression: 3 4 10 99 + * +
result: 439.0
```

Tiếp tục sử dụng bằng cách ấn continue



Input: 3+5 +120

Báo lỗi do đầu vào phải từ 0→99, tương tự các lỗi khác cũng sẽ báo lỗi như thế