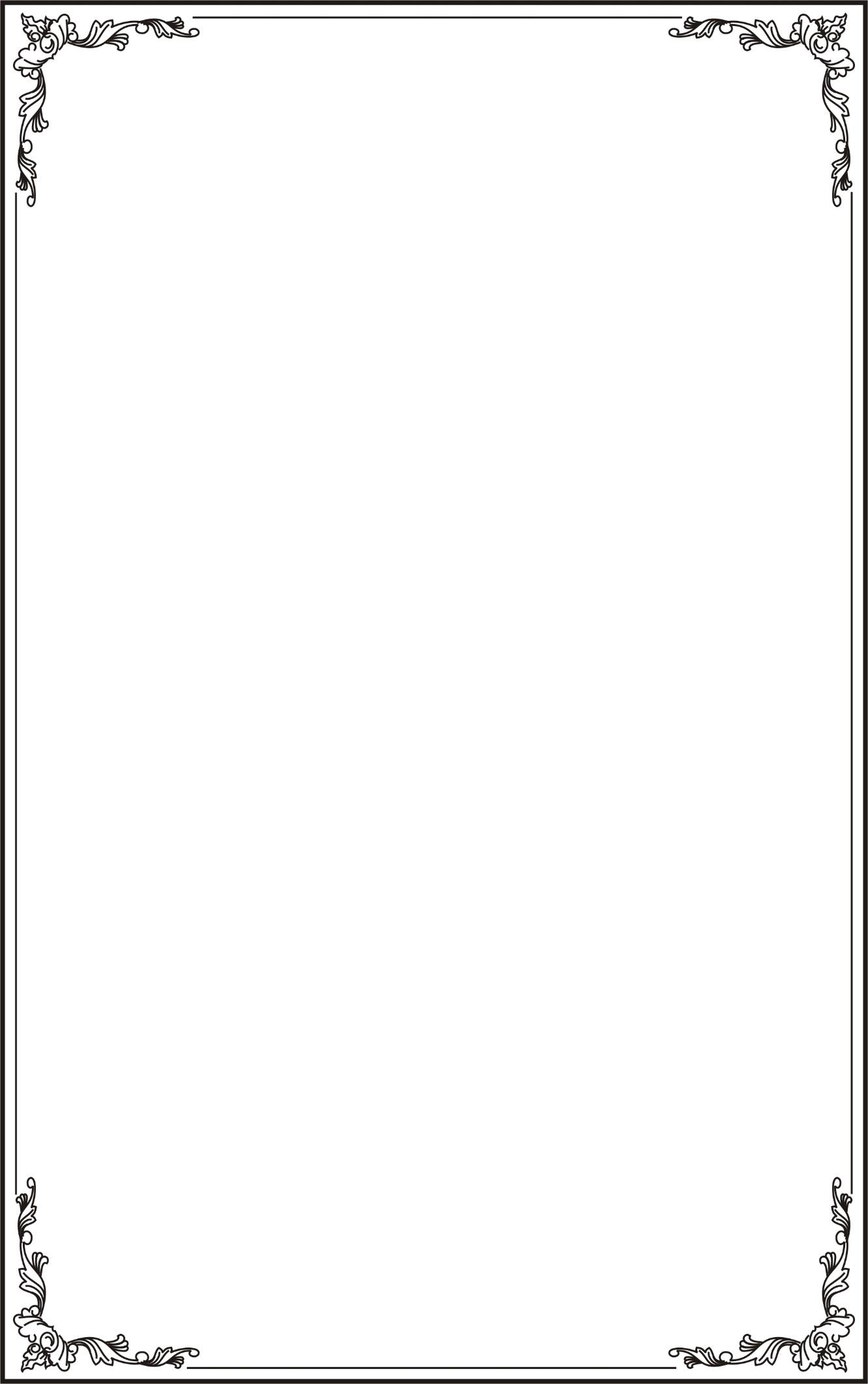
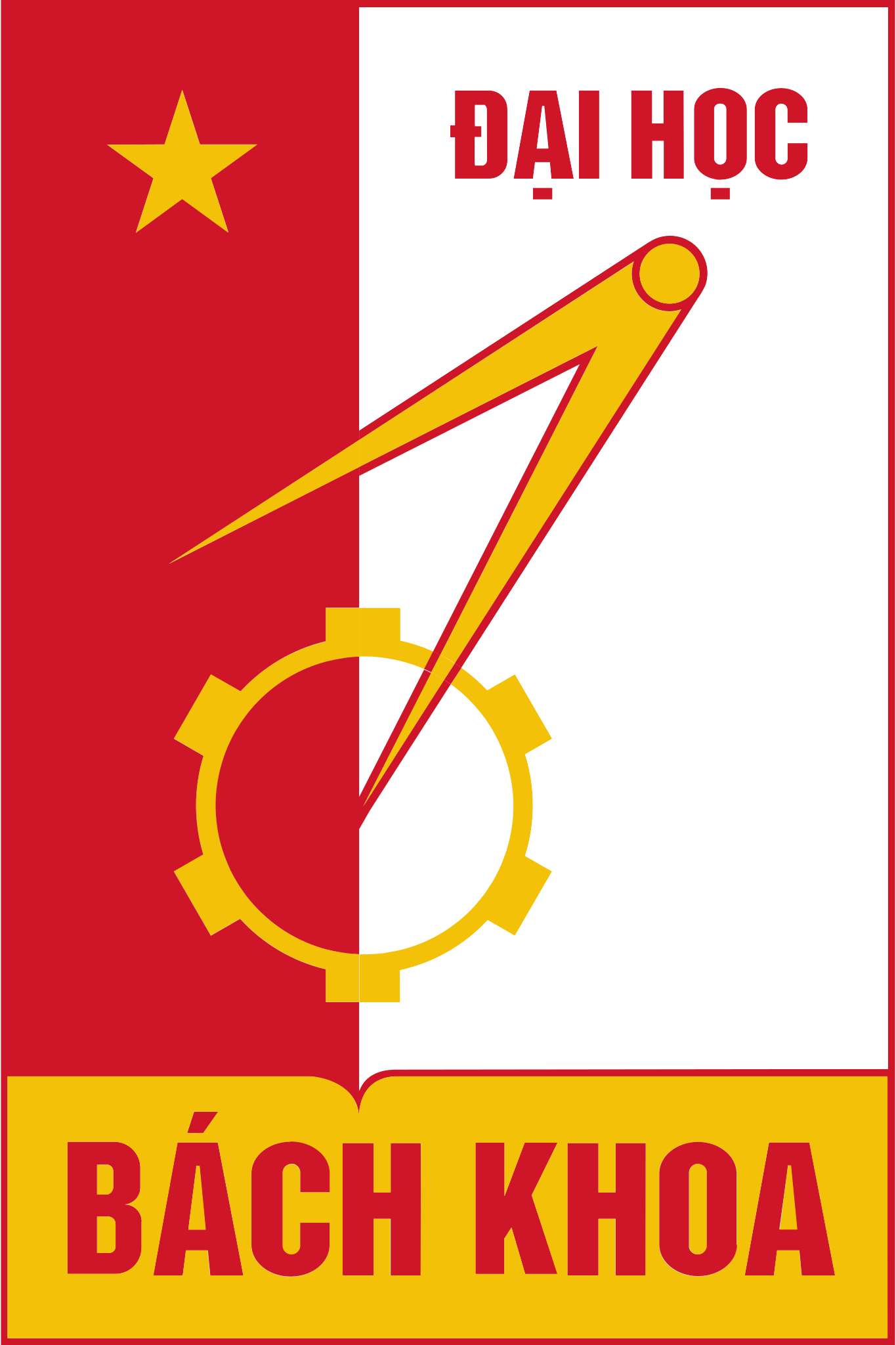
ĐẠI HỌC BÁCH KHOA HÀ NỘI

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

**~~~~~** 🙟🕮🙝 **~~~~~**



**BÁO CÁO PROJECT CUỐI KỲ MÔN THỰC HÀNH KIẾN TRÚC MÁY TÍNH**

Sinh viên thực hiện: -Nguyễn Duy Khánh 20204992

-Lê Đức Anh Duy 20200111

Giảng viên hướng dẫn: ThS. Lê Bá Vui

Hà Nội, 07/2022

Project được thực hiện bởi nhóm 8, gồm các thành viên:

* *Sinh viên: Nguyễn Duy Khánh. MSSV: 20204992. Bài 7*
* *Sinh viên: Lê Đức Anh Duy. MSSV: 20200111. Bài 8*

Hướng dẫn: *Giảng viên Lê Bá Vui*

**Bài 7: CHƯƠNG TÌNH KIỂM TRA CÚ PHÁP LỆNH MIPS**

Đề bài:

Chương trình kiểm tra cú pháp lệnh MIPS

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

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

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

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

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.

**I, Phân tích cách làm**

Yêu cầu của Project là kiểm tra xem câu lệnh hợp ngữ có cú pháp đúng hay không.

-Để có thể check xem câu lệnh có đúng hay không thì ta cần lưu các opcode thỏa mãn, các thanh ghi thỏa mãn vào một chỗ nào đó trước rồi khi kiểm tra ta lại lấy ra để so sánh xem opcode nhập vào và thanh ghi nhập vào đã đúng chưa. Ở đây mình sẽ sử dụng mảng ký tự để lưu các opcode mẫu cũng như là các thanh ghi mẫu. và các từ cách nhau bởi dấu ‘/’. Và kết thúc xâu bởi dấu Space

Opcode\_R\_Check: .asciiz "/add/sub/addu/subu/and/or/slt/sltu/nor/srav/srlv/movn/movz/mul/ "

Opcode\_R\_Check\_1: .asciiz "/beq/bne/ "

Opcode\_R\_Check\_2: .asciiz "/div/divu/mfc0/mult/multu/clo/clz/move/negu/not/madd/maddu/msub/msubu/ "

Opcode\_I\_Check: .asciiz "/addi/addiu/andi/ori/slti/sltiu/sll/srl/sra/ "

Opcode\_I\_Check\_1: .asciiz "/li/lui/ "

Opcode\_J\_Check: .asciiz "/j/jal/ "

Opcode\_J\_Check\_1: .asciiz "/jr/mfhi/mthi/mflo/mtlo/ "

Opcode\_L\_Check: .asciiz "/lb/lbu/lhu/ll/lw/sb/sc/sh/sw/lwc1/ldc1/swc1/sdc1/ "

Opcode\_L\_Check\_1: .asciiz "/la/ "

Special\_command: .asciiz "/syscall/nop/ "

Register\_Check: .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/$sp/$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/$23/$24/$25/$26/$27/$28/$29/$30/$31/ "

Việc chia Opcode thành các nhóm như trên không hẳn dựa trên Opcode này thuộc khuôn dạng R, I hay J mà phụ thuộc vào khuôn dạng lệnh khi ta code.

Ví dụ: add và beq đều là thuộc R nhưng khuôn dạng code là khác hẳn nhau:

add $s1, $s2, $s3

beq $s1, $s2, LoopA

-Sau khi lưu các opcode, thanh ghi thỏa mãn như trên rồi thì ta sẽ dựng các hàm để tách và check riêng từng bộ phận của câu lệnh như: Opcode, thanh ghi, số, label, và kiểu dạng như 0($s2).

Sau khi tách được một phần thì ta phải đưa phần đấy đi kiểm tra xem có thỏa mãn hay không. Vậy rõ ràng ta cần phải có một vùng để lưu dữ liệu sau khi tách và đưa dữ liệu này đi kiểm tra. Ở đây mình sử dụng cấu trúc hàng đợi để kiểm soát vùng dữ liệu này. Địa chỉ đầu của hàng đợi được khai báo là ‘chain\_check’ và được lưu vào s2 và vị trí phần tử cuối cùng của hàng đợi được lưu vào s0 (Lưu ý: ví dụ hàng đợi có lưu “beq” thì s0 = 2 vì phần tử đầu tiên được đánh số là 0).

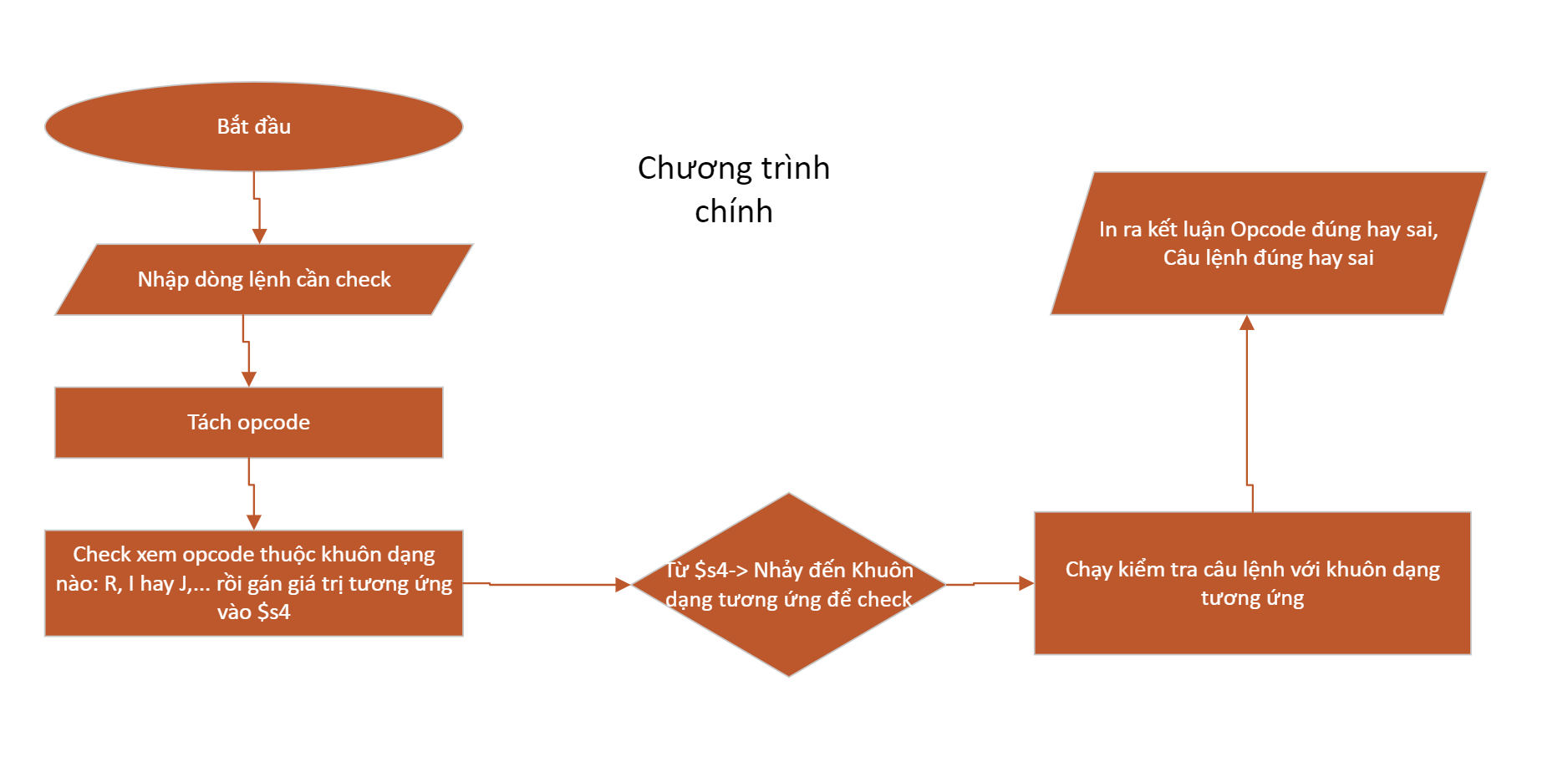
Các hàm trong bài là:

* Split\_opcode: Tách phần opcode -> Lưu vào hàng đợi
* Split\_Register\_and\_Number: Tách phần thanh ghi hoặc số -> Lưu vào hàng đợi
* Split\_Sign\_ExtImm: Tách kí tự dùng để tách khuôn dạng 0($s2) -> Lưu vào hàng đợi
* Check\_opcode: Kiểm tra Opcode có thỏa mãn không và thuộc khuôn dạng nào và gán $s4 với giá trị tương ứng với khuôn dạng lệnh của nó: R (s4 = 1), R\_1 (s4=2), R\_2(s4=6), I(s4=3), I\_1(s4=7), J(s4=4), J\_1(s4=8), L(s4=9), L\_1(s4=10), Đặc biệt (syscall, nop -> s4=5), Opcode sai (s4=0)

-Và ứng với từng kiểu khuôn dạng (với từng giá trị s4) thì ta sẽ nhảy để hàm check tương ứng của khuôn dạng đó (khuôn dạng R, R1, I, ….)

-Sau khi check xong thì ta sẽ in ra kết quả rằng câu lệnh nhập vào có cấu trúc đúng hay sai.

Để có thể dễ hình dung về các bước xây dựng thuật toán thì dưới đây là lưu đồ thuật toán.



Code:

#main

jal Split\_opcode

jal Check\_opcode

beq $s4, $zero, False\_opcode #Opcode false

addi $t0, $zero, 5 #Syscall, nop->Right code

beq $s4, $t0, Right\_code

addi $t5, $zero, 1

beq $s4, $t5, R\_Check\_Register\_and\_Number

addi $t5, $zero, 2

beq $s4, $t5, R\_1\_Check\_Register\_and\_Number

addi $t5, $zero, 3

beq $s4, $t5, I\_Check\_Register\_and\_Number

addi $t5, $zero, 4

beq $s4, $t5, J\_Check\_Register\_and\_Number

addi $t5, $zero, 6

beq $s4, $t5, R\_2\_Check\_Register\_and\_Number

addi $t5, $zero, 7

beq $s4, $t5, I\_1\_Check\_Register\_and\_Number

addi $t5, $zero, 8

beq $s4, $t5, J\_1\_Check\_Register\_and\_Number

addi $t5, $zero, 9

beq $s4, $t5, L\_Check\_Register\_and\_Number

addi $t5, $zero, 10

beq $s4, $t5, L\_1\_Check\_Register\_and\_Number

j End\_main

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

**II, Thuật toán và mã nguồn từng bước.**

**1.Chức năng của các thanh ghi**

-Để có thể dễ theo dõi thuật toán và code thì chúng ta sẽ thống nhất một số thanh ghi sẽ thực hiện các chức năng cụ thể trong suốt chương trình

-s0: vị trí phần tử cuối của hàng đợi

-s1: địa chỉ của lệnh vào

-s2: địa chỉ hàng đợi (phần tử đầu)

-s4: lưu giá trị tương ứng với opcode của lệnh như đã nói ở trên

-s5: vị trí đang load s1. (Ví dụ: “beq $s1, $s2, ABC” thì s5=4)

-t9: kí tự cuối cùng được load

-s6 = 32 là kí tự space

-s7 = 47 là kí tự ‘/’

**2. Tách Opcode và Check opcode**

Tách opcode:

#Tach ma opcode

Split\_opcode:

li $s0, 0 #Vi tri phan tu cuoi cua mang chain\_check

li $t1, 0 #i=0

Loop1:

add $a2, $s1, $t1 #a2 = Dia chi cua ky tu dang load

add $a3, $s2, $s0 #a3 = Dia chi dang nap vao hang doi

lb $t0, 0($a2)

beq $t0, $zero, EndLoop #Gap null => ket thuc vong lap 1

beq $t0, $s6, Loop1\_them

sb $t0, 0($a3) #Nap ky tu vao hang doi

addi $s0, $s0, 1 #Dich chuyen vi tri cuoi cua hang doi sang phai

addi $t1, $t1, 1

Loop2:

add $a2, $s1, $t1 #a2 = Dia chi cua ky tu dang load

add $a3, $s2, $s0 #a3 = Dia chi dang nap vao hang doi

lb $t0, 0($a2)

beq $t0, $zero, EndLoop #Gap null => ket thuc vong lap 1

beq $t0, $s6, EndLoop #Gap space => ket thuc vong lap 1

li $t5, 10 #t5=newline

beq $t0, $t5, EndLoop #Gap newline => ket thuc vong lap 1

sb $t0, 0($a3) #Nap ky tu vao hang doi

addi $s0, $s0, 1 #Dich chuyen vi tri cuoi cua hang doi sang phai

addi $t1, $t1, 1

j Loop2

EndLoop:

#Chen ky tu NULL cho hang doi

sb $zero, 0($a3)

add $s5, $s0, $zero #Luu vi tri ki tu dang doc vao s5

addi $s0, $s0, -1

jr $ra

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

Giải thích:

-Vòng Loop1 để load kí tự đầu tiên đồng thời để loại bỏ các kí tự space thừa ở trước opcode

-Vòng Loop2 để load các kí tự tiếp theo của opcode và Loop 2 kết thúc khi gặp space hoặc new line hoặc null.

Check opcode:

Chức năng của các thanh ghi:

-s2: địa chỉ kí tự đầu của hàng đợi

-s3: địa chỉ xâu opcode mẫu (đã lưu từ trước)

-s0: vị trí cuối của hàng đợi

-a2: địa chỉ kí tự opcode đang load từ hàng đợi đang cần kiểm tra

-a3: địa chỉ kí tự đang load của opcode mẫu

-s4: chứa giá trị của khuôn dạng lệnh (mặc định ban đầu = 0)

-t1: i (mặc định ban đầu = 0)

-t2: kí tự load từ a2

-t3: kí tự load từ a3

-t0: số kí tự của opcode mẫu

Giải thích:

-Hàm check opcode sẽ gồm các hàm check nhỏ theo từng khuôn dạng lệnh và được sắp xếp liên tiếp nhau.

-Chương trình sẽ kiểm tra khuôn dạng lệnh đầu tiên xem có opcode có thuộc khuôn dạng lệnh đó không. Nếu thuộc thì gán s4 tương ứng và nhảy về chương trình chính (main). Nếu không thuộc thì sẽ tới khuôn dạng tiếp theo để xét.

-Nếu xét hết tất cả các khuôn dạng lệnh trong hàm check opcode mà opcode vẫn chưa tìm được khuôn dạng thỏa mãn thì nhảy về chương trình chính -> Lúc này s4 = 0 -> Opcode sai!

Sau đây là 1 đoạn code để check khuôn dạng lệnh R nằm trong hàm Check opcode:

#Check\_R

la $s3, Opcode\_R\_Check

li $t1, 0 #i=0

Loop1\_R:

add $a3, $s3, $t1 #load byte cua opcode mau

lb $t3, 0($a3)

addi $t1, $t1, 1

bne $t3, $s7, Loop1\_R

li $t0, 0 #So ki tu cua opcode mau

Loop2\_R:

add $a3, $s3, $t1 #load byte cua opcode mau

lb $t3, 0($a3)

add $a2, $s2, $t0 #Load byte cua opcode can check

lb $t2, 0($a2)

beq $t3, $s7, Check\_R

beq $t3, $s6, End\_Loop\_R

bne $t2, $t3, Loop1\_R\_them #Kiem tra xem opcode check va opcode mau co giong nhau khong

beq $t2, $t3, Loop2\_R\_them

End\_Loop\_R:

**3. Kiểm tra lệnh theo từng khuôn dạng code**

**3.1. Hàm kiểm tra lệnh theo từng khuôn dạng**

Sau khi ta biết được khuôn dạng lệnh thông qua opcode thì s4 sẽ có giá trị tương ứng => Ta nhảy đến hàm kiểm tra code theo từng khuôn dạng tương ứng với giá trị s4.

Ở đây mình sẽ đưa trước một số opcode của một số khuôn dạng đặc trưng và phân tích:

Hàm kiểm tra khuôn dạng R: (add $s1, $s2, $s3)

R\_Check\_Register\_and\_Number:

jal Right\_opcode

jal Split\_Register\_and\_Number

jal Check\_Register

jal Split\_Register\_and\_Number

jal Check\_Register

jal Split\_Register\_and\_Number

jal Check\_Register

addi $t5, $zero, 10

beq $t9, $t5, Right\_code

addi $t5, $zero, 0

beq $t9, $t5, Right\_code

j False\_code

Giải thích:

-Đầu tiên khi đã tới bước này thì tất nhiên opcode phải thỏa mãn nên ta phải in ra thông báo rằng opcode trong câu lệnh là hợp lệ! Ta sử dụng hàm Right\_opcode để in ra thông báo

-Sau đó ta cần kiểm tra 3 thanh ghi trong câu lệnh có đúng hay không bằng cách tách thanh ghi (Split\_Register\_and\_Number) và kiểm tra xem thanh ghi đó có thỏa mãn hay không (Check\_Register). Vì có3 thanh ghi nên ta phải thực hiện lại thao tác vừa rồi ba lần.

Hàm kiểm tra khuôn dạng lệnh I: (addi $s1, $s2, 1)

I\_Check\_Register\_and\_Number:

jal Right\_opcode

jal Split\_Register\_and\_Number

jal Check\_Register

jal Split\_Register\_and\_Number

jal Check\_Register

jal Split\_Register\_and\_Number

jal Check\_Number

addi $t5, $zero, 10

beq $t9, $t5, Right\_code

addi $t5, $zero, 0

beq $t9, $t5, Right\_code

j False\_code

Giải thích:

-Đầu tiên ta tách và kiểm tra đối với 2 thanh ghi đầu

-Đối với bộ phân cuối cùng là số thì ta sẽ cần tách số (Split\_Register\_and\_Number) và kiểm tra số(Check\_Number) xem số có thỏa mãn hay không.

Hàm kiểm tra khuôn dạng lệnh R\_1: (beq $s1, $s2, Loop\_ABCD)

R\_1\_Check\_Register\_and\_Number:

jal Right\_opcode

jal Split\_Register\_and\_Number

jal Check\_Register

jal Split\_Register\_and\_Number

jal Check\_Register

jal Split\_Register\_and\_Number

addi $t5, $zero, 10

beq $t9, $t5, R\_1\_Check\_Label

addi $t5, $zero, 0

beq $t9, $t5, R\_1\_Check\_Label

j False\_code

R\_1\_Check\_Label:

jal Check\_Label

Giải thích:

-Cách hoạt động của hàm này cũng tương tự như với khuôn dạng lệnh R nhưng phần cuối có nhãn (Label) nên ta phải thực hiện 2 lần tách và kiểm tra thanh ghi và sau đó là tách Label(Split\_Register\_and\_Number) và kiểm tra Label (Check\_Label)

Hàm kiểm tra khuôn dạng lệnh L: ( lb $s1, 12($s2) )

L\_Check\_Register\_and\_Number:

jal Right\_opcode

jal Split\_Register\_and\_Number

jal Check\_Register

jal Check\_Sign\_ExtImm

Giải thích:

-Đầu tiên thì ta vẫn tách và kiểm tra thanh ghi

-Với khuôn dạng như loại này thì nó có cái cụm phía sau khá đặc biệt nên ta sẽ xây dựng một hàm riêng để vừa cả tách đồng thời kiểm tra xem có thỏa mãn hay không (Check\_Sign\_ExtImm)

**3.2. Các hàm tách và kiểm tra từng bộ phận**

**3.2.1. Hàm tách thanh ghi hoặc số**

Code:

Split\_Register\_and\_Number:

li $s0, 0 #Vi tri phan tu cuoi cua mang chain\_check

add $t1, $s5, $zero #i=vi tri dang doc trong cau lenh=s5

Loop1\_Split:

add $a2, $s1, $t1 #a2 = Dia chi cua ky tu dang load

add $a3, $s2, $s0 #a3 = Dia chi dang nap vao hang doi

lb $t0, 0($a2) #t0 = Ky tu dang Load

add $t9, $zero, $t0 #t9 = Ky tu cuoi cung duoc load

beq $t0, $zero, EndLoop\_Split#Check\_Reg\_and\_Num #Gap null => ket thuc vong lap 1

beq $t0, $s6, Loop1\_Split\_them #Gap Space -> Chay qua Space

li $t5, 44 #t5=44~'dau phay,'

beq $t0, $t5, False\_code

sb $t0, 0($a3) #Nap ky tu vao hang doi

addi $s0, $s0, 1 #Dich chuyen vi tri cuoi cua hang doi sang phai

addi $t1, $t1, 1

Loop2\_Split:

add $a2, $s1, $t1 #a2 = Dia chi cua ky tu dang load

add $a3, $s2, $s0 #a3 = Dia chi dang nap vao hang doi

lb $t0, 0($a2)

add $t9, $zero, $t0 #t9 = Ky tu cuoi cung duoc load

beq $t0, $zero, EndLoop\_Split#Check\_Reg\_and\_Num #Gap null => ket thuc vong lap 1

beq $t0, $s6, Loop3\_Split #Gap space => Chay qua Space

li $t5, 10 #t5=newline

beq $t0, $t5, EndLoop\_Split #Check\_Reg\_and\_Num #Gap newline => ket thuc vong lap 1

li $t5, 44 #t5=44~'dau phay,'

beq $t0, $t5, EndLoop\_Split #Gap dau phay => ket thuc vong lap 1

sb $t0, 0($a3) #Nap ky tu vao hang doi

addi $s0, $s0, 1 #Dich chuyen vi tri cuoi cua hang doi sang phai

addi $t1, $t1, 1

j Loop2\_Split

Loop3\_Split:

add $a2, $s1, $t1 #a2 = Dia chi cua ky tu dang load

add $a3, $s2, $s0 #a3 = Dia chi dang nap vao hang doi

lb $t0, 0($a2) #t0 = Ky tu dang Load

add $t9, $zero, $t0 #t9 = Ky tu cuoi cung duoc load

beq $t0, $zero, EndLoop\_Split#Check\_Reg\_and\_Num #Gap null => ket thuc vong lap 1

beq $t0, $s6, Loop3\_Split\_them #Gap Space -> Chay qua Space

li $t5, 44 #t5=44~'dau phay,'

beq $t0, $t5, EndLoop\_Split

li $t5, 10 #t5=10~'New line'

beq $t0, $t5, EndLoop\_Split

j False\_code

EndLoop\_Split:

#Chen ky tu NULL cho hang doi

sb $zero, 0($a3)

addi $s5, $t1, 1 #Luu vi tri ki tu dang doc vao s5

addi $s0, $s0, -1

jr $ra

Giải thích:

-Loop1\_Split: Dùng để chạy bỏ đi các dấu space đứng trước thanh ghi/số và load kí tự đầu vào hàng đợi.

-Loop2\_Split: Dùng để load các kí tự vào hàng đợi, thoát vòng lặp khi gặp space hoặc gặp null, newline, dấu phẩy.

-Loop3\_Split: sẽ được chạy khi gặp space ở Loop2\_Split. Dùng để loại bỏ các dấu space đứng sau thanh ghi/số

-EndLoop\_Spliy: Chèn NULL sau kí tự cuối cùng của hàng đợi, cập nhật s5 và quay về chương trình trước khi gọi hàm.

**3.2.2. Hàm kiểm tra thanh ghi**

Code:

#Check Register

Check\_Register:

la $s3, Register\_Check

li $t1, 0 #i=0

Loop1\_Reg:

add $a3, $s3, $t1 #load byte cua thanh ghi mau

lb $t3, 0($a3)

addi $t1, $t1, 1

bne $t3, $s7, Loop1\_Reg

li $t0, 0 #So ki tu cua thanh ghi mau

Loop2\_Reg:

add $a3, $s3, $t1 #load byte cua thanh ghi mau

lb $t3, 0($a3)

add $a2, $s2, $t0 #Load byte cua thanh ghi can check

lb $t2, 0($a2)

beq $t3, $s7, Check\_Reg

beq $t3, $s6, False\_code

bne $t2, $t3, Loop1\_Reg\_them #Kiem tra xem thanh ghi check va thanh ghi mau co giong nhau khong

beq $t2, $t3, Loop2\_Reg\_them

End\_Loop\_Reg:

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

Check\_Reg:

addi $t0, $t0, -1

beq $s0, $t0, Reg\_True

j Loop1\_Reg

Loop1\_Reg\_them:

addi $t1, $t1, 1

j Loop1\_Reg

Loop2\_Reg\_them:

addi $t1, $t1, 1

addi $t0, $t0, 1

j Loop2\_Reg

Reg\_True:

jr $ra

Chức năng các thanh ghi:

-s2: địa chỉ kí tự đầu của hàng đợi

-s3: địa chỉ xâu thanh ghi mẫu (đã lưu từ trước)

-s0: vị trí cuối của hàng đợi

-a2: địa chỉ kí tự thanh ghi đang load từ hàng đợi đang cần kiểm tra

-a3: địa chỉ kí tự đang load của thanh ghi mẫu

-t1: i (mặc định ban đầu = 0)

-t2: kí tự load từ a2

-t3: kí tự load từ a3

-t0: số kí tự của thanh ghi mẫu

Giải thích:

-Loop1\_Reg: Load kí tự từ xâu thanh ghi mẫu tới khi gặp dấu ‘/’

-Loop2\_Reg: Load kí tự từ xâu thanh ghi mẫu vào t3 và Load kí tự từ hàng đợi vào t2. So sánh t2 và t3.

+Nếu t3 = ‘/’ thì kiểm tra xem t0 =s0 hay không. Bằng thì kết luận thanh ghi cần check thỏa mãn, nếu không thì nhảy về Loop1\_Reg để kiểm tra tiếp.

+Nếu t3 = space -> Kiểm tra hết thanh ghi mẫu rồi mà vẫn chưa thỏa mãn -> thanh ghi sai -> Câu lệnh sai -> Nhảy đến False\_code

+ Nếu t2=t3 thì chạy về Loop2\_Reg

+Nếu t2 != t3 thì chạy về Loop1\_Reg

**3.2.3. Hàm kiểm tra số**

Code:

#Check Number

Check\_Number:

li $t1, 0 #i = 0

j Check\_Mark

Check\_Mark\_done:

add $a2, $s2, $t1 #Kiem tra so dau tien

lb $t2, 0($a2)

li $t5, 10 #t5 = newline

beq $t2, $t5, False\_code

beq $t2, $zero, False\_code

li $t5, 48 #t5 = zero

bne $t2, $t5, Loop\_Number

slti $t4, $t2, 48

bne $t4, $zero, False\_code

slti $t4, $t2, 58

beq $t4, $zero, False\_code

addi $t1, $t1, 1 #Kiem tra so thu hai(co the la x trong so hexa)

add $a2, $s2, $t1

lb $t2, 0($a2)

beq $t2, $zero, Right\_Number

li $t5, 120

beq $t2, $t5, Loop\_Number\_them

li $t5, 88

beq $t2, $t5, Loop\_Number\_them

slti $t4, $t2, 48

bne $t4, $zero, False\_code

slti $t4, $t2, 58

beq $t4, $zero, False\_code

Loop\_Number:

add $a2, $s2, $t1

lb $t2, 0($a2)

beq $t2, $zero, Right\_Number

li $t5, 48

beq $t2, $t5, Loop\_Number\_them

li $t5, 49

beq $t2, $t5, Loop\_Number\_them

li $t5, 50

beq $t2, $t5, Loop\_Number\_them

li $t5, 51

beq $t2, $t5, Loop\_Number\_them

li $t5, 52

beq $t2, $t5, Loop\_Number\_them

li $t5, 53

beq $t2, $t5, Loop\_Number\_them

li $t5, 54

beq $t2, $t5, Loop\_Number\_them

li $t5, 55

beq $t2, $t5, Loop\_Number\_them

li $t5, 56

beq $t2, $t5, Loop\_Number\_them

li $t5, 57

beq $t2, $t5, Loop\_Number\_them

li $t5, 65

beq $t2, $t5, Loop\_Number\_them

li $t5, 66

beq $t2, $t5, Loop\_Number\_them

li $t5, 67

beq $t2, $t5, Loop\_Number\_them

li $t5, 68

beq $t2, $t5, Loop\_Number\_them

li $t5, 69

beq $t2, $t5, Loop\_Number\_them

li $t5, 70

beq $t2, $t5, Loop\_Number\_them

li $t5, 97

beq $t2, $t5, Loop\_Number\_them

li $t5, 98

beq $t2, $t5, Loop\_Number\_them

li $t5, 99

beq $t2, $t5, Loop\_Number\_them

li $t5, 100

beq $t2, $t5, Loop\_Number\_them

li $t5, 101

beq $t2, $t5, Loop\_Number\_them

li $t5, 102

beq $t2, $t5, Loop\_Number\_them

j False\_code

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

Right\_Number:

jr $ra

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

Check\_Mark: #Ham kiem tra dau cua imm

add $a2, $s2, $t1 #Kiem tra xem ki tu dau tien cua Imm co phai dau + hay - khong?

lb $t2, 0($a2)

li $t5, 43 #t5 =43 ~ '+'

beq $t2, $t5, Check\_Mark\_them

li $t5, 45 #t5 =45 ~ '-'

beq $t2, $t5, Check\_Mark\_them

j Check\_Mark\_done

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

Chức năng của các thanh ghi:

-s2: địa chỉ kí tự đầu của hàng đợi

-s0: vị trí cuối của hàng đợi

-a2: địa chỉ kí tự thanh ghi đang load từ hàng đợi đang cần kiểm tra

-t1: i (mặc định ban đầu = 0)

-t2: kí tự load từ a2

-t3: kí tự load từ a3

-t5: kí tự tạm thời dùng để check tạm thời.

Giải thích:

-Check\_Mark\_done: Kiểm tra xem có phải có dấu hay không

-Loop\_Number: load kí tự tiếp theo trong hàng đợi và kiểm tra xem kí tự có hợp lệ hay không. Trong quá trình kiểm tra nếu không hợp lệ -> Số không hợp lê -> Code bị sai -> Nhảy đến False\_code

-Right\_Number: nhảy về chương trình chính.

**3.2.4. Hàm kiểm tra Label.**

#Check Label

Check\_Label:

li $t1, 0 #i = 0

add $a2, $s2, $t1

lb $t2, 0($a2)

beq $t2, $zero, False\_code

li $t5, 10 #t5 = 'New line'

beq $t2, $t5, False\_code

slti $t4, $t2, 48

bne $t4, $zero, False\_code

li $t5, 58

beq $t2, $t5, False\_code

li $t5, 59

beq $t2, $t5, False\_code

li $t5, 60

beq $t2, $t5, False\_code

li $t5, 61

beq $t2, $t5, False\_code

li $t5, 62

beq $t2, $t5, False\_code

li $t5, 63

beq $t2, $t5, False\_code

li $t5, 64

beq $t2, $t5, False\_code

li $t5, 91

beq $t2, $t5, False\_code

li $t5, 92

beq $t2, $t5, False\_code

li $t5, 93

beq $t2, $t5, False\_code

li $t5, 94

beq $t2, $t5, False\_code

li $t5, 96

beq $t2, $t5, False\_code

slti $t4, $t2, 123

beq $t4, $zero, False\_code

addi $t1, $t1, 1

Loop\_Label:

add $a2, $s2, $t1

lb $t2, 0($a2)

beq $t2, $zero, Right\_code

li $t5, 10 #t5 = 'New line'

beq $t2, $t5, Right\_code

slti $t4, $t2, 48

bne $t4, $zero, False\_code

li $t5, 58

beq $t2, $t5, False\_code

li $t5, 59

beq $t2, $t5, False\_code

li $t5, 60

beq $t2, $t5, False\_code

li $t5, 61

beq $t2, $t5, False\_code

li $t5, 62

beq $t2, $t5, False\_code

li $t5, 63

beq $t2, $t5, False\_code

li $t5, 64

beq $t2, $t5, False\_code

li $t5, 91

beq $t2, $t5, False\_code

li $t5, 92

beq $t2, $t5, False\_code

li $t5, 93

beq $t2, $t5, False\_code

li $t5, 94

beq $t2, $t5, False\_code

li $t5, 96

beq $t2, $t5, False\_code

slti $t4, $t2, 123

beq $t4, $zero, False\_code

addi $t1, $t1, 1

j Loop\_Label

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

Giải thích:

-Đầu tiên ta load kí tự đầu và kiểm tra xem nó có hợp lệ không (gặp new line hoặc null -> False\_code)

-Loop\_Label: Xét các kí tự tiếp theo xem có thỏa mãn không.

Gặp new line hoặc null -> Right\_code

**3.2.5. Tách và kiểm tra cấu trúc đặc biệt {0($s2)}**

Code:

#Tach Sign ExtImm

Split\_Sign\_ExtImm:

li $s0, 0 #Vi tri phan tu cuoi cua mang chain\_check

add $t1, $s5, $zero #i=vi tri dang doc trong cau lenh=s5

Loop1\_Sign:

add $a2, $s1, $t1 #a2 = Dia chi cua ky tu dang load

add $a3, $s2, $s0 #a3 = Dia chi dang nap vao hang doi

lb $t0, 0($a2) #t0 = Ky tu dang Load

add $t9, $zero, $t0 #t9 = Ky tu cuoi cung duoc load

beq $t0, $zero, EndLoop\_Sign\_them\_2#Check\_Reg\_and\_Num #Gap null => ket thuc vong lap 1

li $t5, 10 #t5=10~'New line'

beq $t0, $t5, EndLoop\_Sign\_them\_2

beq $t0, $s6, Loop1\_Sign\_them #Gap Space -> Chay qua Space

li $t5, 44 #t5=44~'dau phay,'

beq $t0, $t5, False\_code

sb $t0, 0($a3) #Nap ky tu vao hang doi

li $t5, 40 #Thay dau ( thi ket thuc

beq $t0, $t5, EndLoop\_Sign\_them

li $t5, 41 #Thay dau ) thi ket thuc

beq $t0, $t5, EndLoop\_Sign\_them

addi $s0, $s0, 1 #Dich chuyen vi tri cuoi cua hang doi sang phai

addi $t1, $t1, 1

Loop2\_Sign:

add $a2, $s1, $t1 #a2 = Dia chi cua ky tu dang load

add $a3, $s2, $s0 #a3 = Dia chi dang nap vao hang doi

lb $t0, 0($a2)

add $t9, $zero, $t0 #t9 = Ky tu cuoi cung duoc load

beq $t0, $zero, EndLoop\_Sign\_them\_2#Check\_Reg\_and\_Num #Gap null => ket thuc vong lap 1

li $t5, 10 #t5=10~'New line'

beq $t0, $t5, EndLoop\_Sign\_them\_2

beq $t0, $s6, EndLoop\_Sign #Gap space => Chay qua Space

li $t5, 10 #t5=newline

beq $t0, $t5, EndLoop\_Sign #Check\_Reg\_and\_Num #Gap newline => ket thuc vong lap 1

li $t5, 44 #t5=44~'dau phay,'

beq $t0, $t5, EndLoop\_Sign #Gap dau phay => ket thuc vong lap 1

li $t5, 40 #Thay dau ( thi ket thuc

beq $t0, $t5, EndLoop\_Sign\_them\_1

li $t5, 41 #Thay dau ) thi ket thuc

beq $t0, $t5, EndLoop\_Sign\_them\_1

sb $t0, 0($a3) #Nap ky tu vao hang doi

addi $s0, $s0, 1 #Dich chuyen vi tri cuoi cua hang doi sang phai

addi $t1, $t1, 1

j Loop2\_Sign

EndLoop\_Sign:

#Chen ky tu NULL cho hang doi

sb $zero, 0($a3)

addi $s5, $t1, 0

addi $s0, $s0, -1

jr $ra

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

#Check Sign\_ExtImm

Check\_Sign\_ExtImm:

add $t8, $zero, $ra #Luu dia chi tro ve chuong trinh vao -> t8

jal Split\_Sign\_ExtImm

jal Check\_Number

jal Split\_Sign\_ExtImm

jal Check\_Parentheses\_1

jal Split\_Sign\_ExtImm

jal Check\_Register

jal Split\_Sign\_ExtImm

jal Check\_Parentheses\_2

addi $t5, $zero, 10

beq $t9, $t5, Right\_code

addi $t5, $zero, 0

beq $t9, $t5, Right\_code

addi $t5, $zero, 41 #t5 ~ ')'

beq $t9, $t5, Right\_code

j False\_code

**4. In ra kết luận**

Ta lưu một số xâu kí tự để in ra thông báo.

Code:

.data

Message1: .asciiz "Nhap dong lenh can check: "

Message2: .asciiz "\nOpcode: "

Message3: .asciiz ", hop le!"

Message4: .asciiz " khong hop le!"

Message5: .asciiz " \nCau lenh dung!"

Message6: .asciiz " \nCau lenh sai!"

Message7: .asciiZ " \n"

**4.1. Trường hợp opcode sai**

-Ta phải in ra thông báo opcode sai và câu lệnh sai

Code:

False\_opcode:

#Print "Opcode"

li $v0, 4

la $a0, Message2

syscall

nop

#Print Opcode Input

li $v0, 4

add $a0, $zero, $s2

syscall

nop

#Print "Khong hop le!"

li $v0, 4

la $a0, Message4

syscall

nop

jal False\_code

j End\_main

**4.2. Trường hợp câu lệnh đúng**

-Ta phải in ra thông báo opcode đúng và câu lệnh đúng

Code:

Right\_opcode:

#Print "Opcode"

li $v0, 4

la $a0, Message2

syscall

nop

#Print Opcode Input

li $v0, 4

add $a0, $zero, $s2

syscall

nop

#Print ", hop le!"

li $v0, 4

la $a0, Message3

syscall

nop

jr $ra

Right\_code:

#Print "Right code"

li $v0, 4

la $a0, Message5

syscall

nop

j End\_main

**4.3. Trường hợp câu lệnh sai**

Code:

False\_code:

#Print "False code"

li $v0, 4

la $a0, Message6

syscall

nop

j End\_main

**III, Source code**

#Nguyen Duy Khanh 20204992

#Project 07

.data

Message1: .asciiz "Nhap dong lenh can check: "

Message2: .asciiz "\nOpcode: "

Message3: .asciiz ", hop le!"

Message4: .asciiz " khong hop le!"

Message5: .asciiz " \nCau lenh dung!\n-------------------\n"

Message6: .asciiz " \nCau lenh sai!\n-------------------\n"

Message7: .asciiz " \n"

Message8: .asciiz "Thanh ghi "

Message9: .asciiz "So "

Message10: .asciiz "Nhan "

Message11: .asciiz "Ban muon kiem tra tiep khong?"

string: .space 100

#Luu cac opcode can check vao mang

Opcode\_R\_Check: .asciiz "/add/sub/addu/subu/and/or/slt/sltu/nor/srav/srlv/movn/movz/mul/ "

Opcode\_R\_Check\_1: .asciiz "/beq/bne/ "

Opcode\_R\_Check\_2: .asciiz "/div/divu/mfc0/mult/multu/clo/clz/move/negu/not/madd/maddu/msub/msubu/ "

Opcode\_I\_Check: .asciiz "/addi/addiu/andi/ori/slti/sltiu/sll/srl/sra/ "

Opcode\_I\_Check\_1: .asciiz "/li/lui/ "

Opcode\_J\_Check: .asciiz "/j/jal/ "

Opcode\_J\_Check\_1: .asciiz "/jr/mfhi/mthi/mflo/mtlo/ "

Opcode\_L\_Check: .asciiz "/lb/lbu/lhu/ll/lw/sb/sc/sh/sw/lwc1/ldc1/swc1/sdc1/ "

Opcode\_L\_Check\_1: .asciiz "/la/ "

Special\_command: .asciiz "/syscall/nop/ "

Register\_Check: .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/$sp/$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/$23/$24/$25/$26/$27/$28/$29/$30/$31/ "

chain\_check: .word #Chua xau ki tu đang xet

.text

start:

la $s2, chain\_check #Dia chi chua chain\_check

li $s6, 32 #s6=space

li $s7, 47 #s7 = '/'

#Nhap dong lenh can check

li $v0, 54

la $a0, Message1

la $a1, string

la $a2, 100

syscall

la $s1, string

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

#main

jal Split\_opcode

jal Check\_opcode

beq $s4, $zero, False\_opcode #Opcode false

addi $t0, $zero, 5 #Syscall, nop->Right code

beq $s4, $t0, Right\_code

addi $t5, $zero, 1

beq $s4, $t5, R\_Check\_Register\_and\_Number

addi $t5, $zero, 2

beq $s4, $t5, R\_1\_Check\_Register\_and\_Number

addi $t5, $zero, 3

beq $s4, $t5, I\_Check\_Register\_and\_Number

addi $t5, $zero, 4

beq $s4, $t5, J\_Check\_Register\_and\_Number

addi $t5, $zero, 6

beq $s4, $t5, R\_2\_Check\_Register\_and\_Number

addi $t5, $zero, 7

beq $s4, $t5, I\_1\_Check\_Register\_and\_Number

addi $t5, $zero, 8

beq $s4, $t5, J\_1\_Check\_Register\_and\_Number

addi $t5, $zero, 9

beq $s4, $t5, L\_Check\_Register\_and\_Number

addi $t5, $zero, 10

beq $s4, $t5, L\_1\_Check\_Register\_and\_Number

j End\_main

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

#Tach ma opcode

Split\_opcode:

li $s5, 0 #Vi tri load ban dau cua lenh nap vao

li $s0, 0 #Vi tri phan tu cuoi cua mang chain\_check

li $t1, 0 #i=0

Loop1:

add $a2, $s1, $t1 #a2 = Dia chi cua ky tu dang load

add $a3, $s2, $s0 #a3 = Dia chi dang nap vao hang doi

lb $t0, 0($a2)

beq $t0, $zero, EndLoop #Gap null => ket thuc vong lap 1

beq $t0, $s6, Loop1\_them

sb $t0, 0($a3) #Nap ky tu vao hang doi

addi $s0, $s0, 1 #Dich chuyen vi tri cuoi cua hang doi sang phai

addi $t1, $t1, 1

addi $s5, $s5, 1

Loop2:

add $a2, $s1, $t1 #a2 = Dia chi cua ky tu dang load

add $a3, $s2, $s0 #a3 = Dia chi dang nap vao hang doi

lb $t0, 0($a2)

beq $t0, $zero, EndLoop #Gap null => ket thuc vong lap 1

beq $t0, $s6, EndLoop #Gap space => ket thuc vong lap 1

li $t5, 10 #t5=newline

beq $t0, $t5, EndLoop #Gap newline => ket thuc vong lap 1

sb $t0, 0($a3) #Nap ky tu vao hang doi

addi $s0, $s0, 1 #Dich chuyen vi tri cuoi cua hang doi sang phai

addi $t1, $t1, 1

addi $s5, $s5, 1

j Loop2

EndLoop:

#Chen ky tu NULL cho hang doi

sb $zero, 0($a3)

#add $s5, $s0, $zero #Luu vi tri ki tu dang doc vao s5

addi $s0, $s0, -1

jr $ra

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

#Tach ma thanh ghi va so

Split\_Register\_and\_Number:

li $s0, 0 #Vi tri phan tu cuoi cua mang chain\_check

add $t1, $s5, $zero #i=vi tri dang doc trong cau lenh=s5

Loop1\_Split:

add $a2, $s1, $t1 #a2 = Dia chi cua ky tu dang load

add $a3, $s2, $s0 #a3 = Dia chi dang nap vao hang doi

lb $t0, 0($a2) #t0 = Ky tu dang Load

add $t9, $zero, $t0 #t9 = Ky tu cuoi cung duoc load

beq $t0, $zero, EndLoop\_Split#Check\_Reg\_and\_Num #Gap null => ket thuc vong lap 1

beq $t0, $s6, Loop1\_Split\_them #Gap Space -> Chay qua Space

li $t5, 44 #t5=44~'dau phay,'

beq $t0, $t5, False\_code

sb $t0, 0($a3) #Nap ky tu vao hang doi

addi $s0, $s0, 1 #Dich chuyen vi tri cuoi cua hang doi sang phai

addi $t1, $t1, 1

Loop2\_Split:

add $a2, $s1, $t1 #a2 = Dia chi cua ky tu dang load

add $a3, $s2, $s0 #a3 = Dia chi dang nap vao hang doi

lb $t0, 0($a2)

add $t9, $zero, $t0 #t9 = Ky tu cuoi cung duoc load

beq $t0, $zero, EndLoop\_Split#Check\_Reg\_and\_Num #Gap null => ket thuc vong lap 1

beq $t0, $s6, Loop3\_Split #Gap space => Chay qua Space

li $t5, 10 #t5=newline

beq $t0, $t5, EndLoop\_Split #Check\_Reg\_and\_Num #Gap newline => ket thuc vong lap 1

li $t5, 44 #t5=44~'dau phay,'

beq $t0, $t5, EndLoop\_Split #Gap dau phay => ket thuc vong lap 1

sb $t0, 0($a3) #Nap ky tu vao hang doi

addi $s0, $s0, 1 #Dich chuyen vi tri cuoi cua hang doi sang phai

addi $t1, $t1, 1

j Loop2\_Split

Loop3\_Split:

add $a2, $s1, $t1 #a2 = Dia chi cua ky tu dang load

add $a3, $s2, $s0 #a3 = Dia chi dang nap vao hang doi

lb $t0, 0($a2) #t0 = Ky tu dang Load

add $t9, $zero, $t0 #t9 = Ky tu cuoi cung duoc load

beq $t0, $zero, EndLoop\_Split#Check\_Reg\_and\_Num #Gap null => ket thuc vong lap 1

beq $t0, $s6, Loop3\_Split\_them #Gap Space -> Chay qua Space

li $t5, 44 #t5=44~'dau phay,'

beq $t0, $t5, EndLoop\_Split

li $t5, 10 #t5=10~'New line'

beq $t0, $t5, EndLoop\_Split

j False\_code

EndLoop\_Split:

#Chen ky tu NULL cho hang doi

sb $zero, 0($a3)

addi $s5, $t1, 1 #Luu vi tri ki tu dang doc vao s5

addi $s0, $s0, -1

jr $ra

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

#Tach Sign ExtImm

Split\_Sign\_ExtImm:

li $s0, 0 #Vi tri phan tu cuoi cua mang chain\_check

add $t1, $s5, $zero #i=vi tri dang doc trong cau lenh=s5

Loop1\_Sign:

add $a2, $s1, $t1 #a2 = Dia chi cua ky tu dang load

add $a3, $s2, $s0 #a3 = Dia chi dang nap vao hang doi

lb $t0, 0($a2) #t0 = Ky tu dang Load

add $t9, $zero, $t0 #t9 = Ky tu cuoi cung duoc load

beq $t0, $zero, EndLoop\_Sign\_them\_2#Check\_Reg\_and\_Num #Gap null => ket thuc vong lap 1

li $t5, 10 #t5=10~'New line'

beq $t0, $t5, EndLoop\_Sign\_them\_2

beq $t0, $s6, Loop1\_Sign\_them #Gap Space -> Chay qua Space

li $t5, 44 #t5=44~'dau phay,'

beq $t0, $t5, False\_code

sb $t0, 0($a3) #Nap ky tu vao hang doi

li $t5, 40 #Thay dau ( thi ket thuc

beq $t0, $t5, EndLoop\_Sign\_them

li $t5, 41 #Thay dau ) thi ket thuc

beq $t0, $t5, EndLoop\_Sign\_them

addi $s0, $s0, 1 #Dich chuyen vi tri cuoi cua hang doi sang phai

addi $t1, $t1, 1

Loop2\_Sign:

add $a2, $s1, $t1 #a2 = Dia chi cua ky tu dang load

add $a3, $s2, $s0 #a3 = Dia chi dang nap vao hang doi

lb $t0, 0($a2)

add $t9, $zero, $t0 #t9 = Ky tu cuoi cung duoc load

beq $t0, $zero, EndLoop\_Sign\_them\_2#Check\_Reg\_and\_Num #Gap null => ket thuc vong lap 1

li $t5, 10 #t5=10~'New line'

beq $t0, $t5, EndLoop\_Sign\_them\_2

beq $t0, $s6, EndLoop\_Sign #Gap space => Chay qua Space

li $t5, 10 #t5=newline

beq $t0, $t5, EndLoop\_Sign #Check\_Reg\_and\_Num #Gap newline => ket thuc vong lap 1

li $t5, 44 #t5=44~'dau phay,'

beq $t0, $t5, EndLoop\_Sign #Gap dau phay => ket thuc vong lap 1

li $t5, 40 #Thay dau ( thi ket thuc

beq $t0, $t5, EndLoop\_Sign\_them\_1

li $t5, 41 #Thay dau ) thi ket thuc

beq $t0, $t5, EndLoop\_Sign\_them\_1

sb $t0, 0($a3) #Nap ky tu vao hang doi

addi $s0, $s0, 1 #Dich chuyen vi tri cuoi cua hang doi sang phai

addi $t1, $t1, 1

j Loop2\_Sign

EndLoop\_Sign:

#Chen ky tu NULL cho hang doi

sb $zero, 0($a3)

addi $s5, $t1, 0

addi $s0, $s0, -1

jr $ra

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

#Check Opcode

Check\_opcode:

li $s4, 0 #s4 bieu thi cho khuon dang lenh: Saiopcode: 0, R: 1, R\_1: 2, I: 3, J: 4, Dac biet: 5

#Check\_R

la $s3, Opcode\_R\_Check

li $t1, 0 #i=0

Loop1\_R:

add $a3, $s3, $t1 #load byte cua opcode mau

lb $t3, 0($a3)

addi $t1, $t1, 1

bne $t3, $s7, Loop1\_R

li $t0, 0 #So ki tu cua opcode mau

Loop2\_R:

add $a3, $s3, $t1 #load byte cua opcode mau

lb $t3, 0($a3)

add $a2, $s2, $t0 #Load byte cua opcode can check

lb $t2, 0($a2)

beq $t3, $s7, Check\_R

beq $t3, $s6, End\_Loop\_R

bne $t2, $t3, Loop1\_R\_them #Kiem tra xem opcode check va opcode mau co giong nhau khong

beq $t2, $t3, Loop2\_R\_them

End\_Loop\_R:

#Check\_R\_2

la $s3, Opcode\_R\_Check\_2

li $t1, 0 #i=0

Loop1\_R\_2:

add $a3, $s3, $t1 #load byte cua opcode mau

lb $t3, 0($a3)

addi $t1, $t1, 1

bne $t3, $s7, Loop1\_R\_2

li $t0, 0 #So ki tu cua opcode mau

Loop2\_R\_2:

add $a3, $s3, $t1 #load byte cua opcode mau

lb $t3, 0($a3)

add $a2, $s2, $t0 #Load byte cua opcode can check

lb $t2, 0($a2)

beq $t3, $s7, Check\_R\_2

beq $t3, $s6, End\_Loop\_R\_2

bne $t2, $t3, Loop1\_R\_2\_them #Kiem tra xem opcode check va opcode mau co giong nhau khong

beq $t2, $t3, Loop2\_R\_2\_them

End\_Loop\_R\_2:

#Check\_I

la $s3, Opcode\_I\_Check

li $t1, 0 #i=0

Loop1\_I:

add $a3, $s3, $t1 #load byte cua opcode mau

lb $t3, 0($a3)

addi $t1, $t1, 1

bne $t3, $s7, Loop1\_I

li $t0, 0 #So ki tu cua opcode mau

Loop2\_I:

add $a3, $s3, $t1 #load byte cua opcode mau

lb $t3, 0($a3)

add $a2, $s2, $t0 #Load byte cua opcode can check

lb $t2, 0($a2)

beq $t3, $s7, Check\_I

beq $t3, $s6, End\_Loop\_I

bne $t2, $t3, Loop1\_I\_them #Kiem tra xem opcode check va opcode mau co giong nhau khong

beq $t2, $t3, Loop2\_I\_them

End\_Loop\_I:

#Check\_I\_1

la $s3, Opcode\_I\_Check\_1

li $t1, 0 #i=0

Loop1\_I\_1:

add $a3, $s3, $t1 #load byte cua opcode mau

lb $t3, 0($a3)

addi $t1, $t1, 1

bne $t3, $s7, Loop1\_I\_1

li $t0, 0 #So ki tu cua opcode mau

Loop2\_I\_1:

add $a3, $s3, $t1 #load byte cua opcode mau

lb $t3, 0($a3)

add $a2, $s2, $t0 #Load byte cua opcode can check

lb $t2, 0($a2)

beq $t3, $s7, Check\_I\_1

beq $t3, $s6, End\_Loop\_I\_1

bne $t2, $t3, Loop1\_I\_1\_them #Kiem tra xem opcode check va opcode mau co giong nhau khong

beq $t2, $t3, Loop2\_I\_1\_them

End\_Loop\_I\_1:

#Check\_J

la $s3, Opcode\_J\_Check

li $t1, 0 #i=0

Loop1\_J:

add $a3, $s3, $t1 #load byte cua opcode mau

lb $t3, 0($a3)

addi $t1, $t1, 1

bne $t3, $s7, Loop1\_J

li $t0, 0 #So ki tu cua opcode mau

Loop2\_J:

add $a3, $s3, $t1 #load byte cua opcode mau

lb $t3, 0($a3)

add $a2, $s2, $t0 #Load byte cua opcode can check

lb $t2, 0($a2)

beq $t3, $s7, Check\_J

beq $t3, $s6, End\_Loop\_J

bne $t2, $t3, Loop1\_J\_them #Kiem tra xem opcode check va opcode mau co giong nhau khong

beq $t2, $t3, Loop2\_J\_them

End\_Loop\_J:

#Check\_J\_1

la $s3, Opcode\_J\_Check\_1

li $t1, 0 #i=0

Loop1\_J\_1:

add $a3, $s3, $t1 #load byte cua opcode mau

lb $t3, 0($a3)

addi $t1, $t1, 1

bne $t3, $s7, Loop1\_J\_1

li $t0, 0 #So ki tu cua opcode mau

Loop2\_J\_1:

add $a3, $s3, $t1 #load byte cua opcode mau

lb $t3, 0($a3)

add $a2, $s2, $t0 #Load byte cua opcode can check

lb $t2, 0($a2)

beq $t3, $s7, Check\_J\_1

beq $t3, $s6, End\_Loop\_J\_1

bne $t2, $t3, Loop1\_J\_1\_them #Kiem tra xem opcode check va opcode mau co giong nhau khong

beq $t2, $t3, Loop2\_J\_1\_them

End\_Loop\_J\_1:

#Check Special Command

la $s3, Special\_command

li $t1, 0 #i=0

Loop1\_Sc:

add $a3, $s3, $t1 #load byte cua opcode mau

lb $t3, 0($a3)

addi $t1, $t1, 1

bne $t3, $s7, Loop1\_Sc

li $t0, 0 #So ki tu cua opcode mau

Loop2\_Sc:

add $a3, $s3, $t1 #load byte cua opcode mau

lb $t3, 0($a3)

add $a2, $s2, $t0 #Load byte cua opcode can check

lb $t2, 0($a2)

beq $t3, $s7, Check\_Sc

beq $t3, $s6, End\_Loop\_Sc

bne $t2, $t3, Loop1\_Sc\_them #Kiem tra xem opcode check va opcode mau co giong nhau khong

beq $t2, $t3, Loop2\_Sc\_them

End\_Loop\_Sc:

#Check\_L

la $s3, Opcode\_L\_Check

li $t1, 0 #i=0

Loop1\_L:

add $a3, $s3, $t1 #load byte cua opcode mau

lb $t3, 0($a3)

addi $t1, $t1, 1

bne $t3, $s7, Loop1\_L

li $t0, 0 #So ki tu cua opcode mau

Loop2\_L:

add $a3, $s3, $t1 #load byte cua opcode mau

lb $t3, 0($a3)

add $a2, $s2, $t0 #Load byte cua opcode can check

lb $t2, 0($a2)

beq $t3, $s7, Check\_L

beq $t3, $s6, End\_Loop\_L

bne $t2, $t3, Loop1\_L\_them #Kiem tra xem opcode check va opcode mau co giong nhau khong

beq $t2, $t3, Loop2\_L\_them

End\_Loop\_L:

#Check\_L\_1

la $s3, Opcode\_L\_Check\_1

li $t1, 0 #i=0

Loop1\_L\_1:

add $a3, $s3, $t1 #load byte cua opcode mau

lb $t3, 0($a3)

addi $t1, $t1, 1

bne $t3, $s7, Loop1\_L\_1

li $t0, 0 #So ki tu cua opcode mau

Loop2\_L\_1:

add $a3, $s3, $t1 #load byte cua opcode mau

lb $t3, 0($a3)

add $a2, $s2, $t0 #Load byte cua opcode can check

lb $t2, 0($a2)

beq $t3, $s7, Check\_L\_1

beq $t3, $s6, End\_Loop\_L\_1

bne $t2, $t3, Loop1\_L\_1\_them #Kiem tra xem opcode check va opcode mau co giong nhau khong

beq $t2, $t3, Loop2\_L\_1\_them

End\_Loop\_L\_1:

#Check\_R\_1

la $s3, Opcode\_R\_Check\_1

li $t1, 0 #i=0

Loop1\_R\_1:

add $a3, $s3, $t1 #load byte cua opcode mau

lb $t3, 0($a3)

addi $t1, $t1, 1

bne $t3, $s7, Loop1\_R\_1

li $t0, 0 #So ki tu cua opcode mau

Loop2\_R\_1:

add $a3, $s3, $t1 #load byte cua opcode mau

lb $t3, 0($a3)

add $a2, $s2, $t0 #Load byte cua opcode can check

lb $t2, 0($a2)

beq $t3, $s7, Check\_R\_1

beq $t3, $s6, End\_Loop\_R\_1

bne $t2, $t3, Loop1\_R\_1\_them #Kiem tra xem opcode check va opcode mau co giong nhau khong

beq $t2, $t3, Loop2\_R\_1\_them

End\_Loop\_R\_1:

jr $ra

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

#Check cac thanh ghi va so

R\_Check\_Register\_and\_Number:

jal Right\_opcode

jal Split\_Register\_and\_Number

jal Check\_Register

#jal Check\_Number

jal Split\_Register\_and\_Number

jal Check\_Register

jal Split\_Register\_and\_Number

jal Check\_Register

addi $t5, $zero, 10

beq $t9, $t5, Right\_code

addi $t5, $zero, 0

beq $t9, $t5, Right\_code

j False\_code

R\_1\_Check\_Register\_and\_Number:

jal Right\_opcode

jal Split\_Register\_and\_Number

jal Check\_Register

jal Split\_Register\_and\_Number

jal Check\_Register

jal Split\_Register\_and\_Number

addi $t5, $zero, 10

beq $t9, $t5, R\_1\_Check\_Label

addi $t5, $zero, 0

beq $t9, $t5, R\_1\_Check\_Label

j False\_code

R\_1\_Check\_Label:

jal Check\_Label

R\_2\_Check\_Register\_and\_Number:

jal Right\_opcode

jal Split\_Register\_and\_Number

jal Check\_Register

jal Split\_Register\_and\_Number

jal Check\_Register

addi $t5, $zero, 10

beq $t9, $t5, Right\_code

addi $t5, $zero, 0

beq $t9, $t5, Right\_code

j False\_code

I\_Check\_Register\_and\_Number:

jal Right\_opcode

jal Split\_Register\_and\_Number

jal Check\_Register

#jal Check\_Number

jal Split\_Register\_and\_Number

jal Check\_Register

jal Split\_Register\_and\_Number

jal Check\_Number

addi $t5, $zero, 10

beq $t9, $t5, Right\_code

addi $t5, $zero, 0

beq $t9, $t5, Right\_code

j False\_code

I\_1\_Check\_Register\_and\_Number:

jal Right\_opcode

jal Split\_Register\_and\_Number

jal Check\_Register

#jal Check\_Number

jal Split\_Register\_and\_Number

jal Check\_Number

addi $t5, $zero, 10

beq $t9, $t5, Right\_code

addi $t5, $zero, 0

beq $t9, $t5, Right\_code

j False\_code

J\_Check\_Register\_and\_Number:

jal Right\_opcode

jal Split\_Register\_and\_Number

addi $t5, $zero, 10

beq $t9, $t5, J\_Check\_Label

addi $t5, $zero, 0

beq $t9, $t5, J\_Check\_Label

j False\_code

J\_Check\_Label:

jal Check\_Label

J\_1\_Check\_Register\_and\_Number:

jal Right\_opcode

jal Split\_Register\_and\_Number

jal Check\_Register

addi $t5, $zero, 10

beq $t9, $t5, Right\_code

addi $t5, $zero, 0

beq $t9, $t5, Right\_code

j False\_code

L\_Check\_Register\_and\_Number:

jal Right\_opcode

jal Split\_Register\_and\_Number

jal Check\_Register

jal Check\_Sign\_ExtImm

L\_1\_Check\_Register\_and\_Number:

jal Right\_opcode

jal Split\_Register\_and\_Number

jal Check\_Register

jal Split\_Register\_and\_Number

addi $t5, $zero, 10

beq $t9, $t5, L\_1\_Check\_Label

addi $t5, $zero, 0

beq $t9, $t5, L\_1\_Check\_Label

j False\_code

L\_1\_Check\_Label:

jal Check\_Label

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

Loop1\_them:

addi $t1, $t1, 1

addi $s5, $s5, 1

j Loop1

Loop1\_Split\_them:

addi $t1, $t1, 1

j Loop1\_Split

Loop2\_Split\_them:

addi $t1, $t1, 1

j Loop2\_Split

Loop3\_Split\_them:

addi $t1, $t1, 1

j Loop3\_Split

Loop1\_Sign\_them:

addi $s5, $s5, 1

addi $t1, $t1, 1

j Loop1\_Sign

Loop2\_Sign\_them:

addi $t1, $t1, 1

j Loop2\_Sign

EndLoop\_Sign\_them:

addi $a3, $a3, 1

sb $zero, 0($a3)

addi $s5, $s5, 1

jr $ra

#addi $s0, $s0, -1

#addi $t1, $t1, 1

#add $a3, $s2, $s0 #Cap nhat moi dia chi dang load cua hang doi

#j EndLoop\_Sign

EndLoop\_Sign\_them\_1:

add $a3, $s2, $s0 #Cap nhat moi dia chi dang load cua hang doi

j EndLoop\_Sign

EndLoop\_Sign\_them\_2:

add $s0, $s0, 1

j EndLoop\_Sign

Loop\_Number\_them:

addi $t1, $t1, 1

j Loop\_Number

Loop\_Number\_them\_1:

addi $t1, $t1, 1

j Loop\_Number\_1

Check\_Mark\_them:

addi $t1, $t1, 1

j Check\_Mark\_done

#Check thanh ghi R

Check\_R:

addi $t0, $t0, -1

beq $s0, $t0, R\_True

j Loop1\_R

Loop1\_R\_them:

addi $t1, $t1, 1

j Loop1\_R

Loop2\_R\_them:

addi $t1, $t1, 1

addi $t0, $t0, 1

j Loop2\_R

R\_True:

li $s4, 1

jr $ra

#Check thanh ghi R\_2

Check\_R\_2:

addi $t0, $t0, -1

beq $s0, $t0, R\_2\_True

j Loop1\_R\_2

Loop1\_R\_2\_them:

addi $t1, $t1, 1

j Loop1\_R\_2

Loop2\_R\_2\_them:

addi $t1, $t1, 1

addi $t0, $t0, 1

j Loop2\_R\_2

R\_2\_True:

li $s4, 6

jr $ra

#Check thanh ghi I

Check\_I:

addi $t0, $t0, -1

beq $s0, $t0, I\_True

j Loop1\_I

Loop1\_I\_them:

addi $t1, $t1, 1

j Loop1\_I

Loop2\_I\_them:

addi $t1, $t1, 1

addi $t0, $t0, 1

j Loop2\_I

I\_True:

li $s4, 3

jr $ra

#Check thanh ghi I\_1

Check\_I\_1:

addi $t0, $t0, -1

beq $s0, $t0, I\_1\_True

j Loop1\_I\_1

Loop1\_I\_1\_them:

addi $t1, $t1, 1

j Loop1\_I\_1

Loop2\_I\_1\_them:

addi $t1, $t1, 1

addi $t0, $t0, 1

j Loop2\_I\_1

I\_1\_True:

li $s4, 7

jr $ra

#Check thanh ghi J

Check\_J:

addi $t0, $t0, -1

beq $s0, $t0, J\_True

j Loop1\_J

Loop1\_J\_them:

addi $t1, $t1, 1

j Loop1\_J

Loop2\_J\_them:

addi $t1, $t1, 1

addi $t0, $t0, 1

j Loop2\_J

J\_True:

li $s4, 4

jr $ra

#Check thanh ghi J\_1

Check\_J\_1:

addi $t0, $t0, -1

beq $s0, $t0, J\_1\_True

j Loop1\_J\_1

Loop1\_J\_1\_them:

addi $t1, $t1, 1

j Loop1\_J\_1

Loop2\_J\_1\_them:

addi $t1, $t1, 1

addi $t0, $t0, 1

j Loop2\_J\_1

J\_1\_True:

li $s4, 8

jr $ra

#Check thanh ghi Sc - Special Command

Check\_Sc:

addi $t0, $t0, -1

beq $s0, $t0, Sc\_True

j Loop1\_Sc

Loop1\_Sc\_them:

addi $t1, $t1, 1

j Loop1\_Sc

Loop2\_Sc\_them:

addi $t1, $t1, 1

addi $t0, $t0, 1

j Loop2\_Sc

Sc\_True:

li $s4, 5

jr $ra

#Check thanh ghi R\_1

Check\_R\_1:

addi $t0, $t0, -1

beq $s0, $t0, R\_1\_True

j Loop1\_R\_1

Loop1\_R\_1\_them:

addi $t1, $t1, 1

j Loop1\_R\_1

Loop2\_R\_1\_them:

addi $t1, $t1, 1

addi $t0, $t0, 1

j Loop2\_R\_1

R\_1\_True:

li $s4, 2

jr $ra

#Check thanh ghi L

Check\_L:

addi $t0, $t0, -1

beq $s0, $t0, L\_True

j Loop1\_L

Loop1\_L\_them:

addi $t1, $t1, 1

j Loop1\_L

Loop2\_L\_them:

addi $t1, $t1, 1

addi $t0, $t0, 1

j Loop2\_L

L\_True:

li $s4, 9

jr $ra

#Check thanh ghi L\_1

Check\_L\_1:

addi $t0, $t0, -1

beq $s0, $t0, L\_1\_True

j Loop1\_L\_1

Loop1\_L\_1\_them:

addi $t1, $t1, 1

j Loop1\_L\_1

Loop2\_L\_1\_them:

addi $t1, $t1, 1

addi $t0, $t0, 1

j Loop2\_L\_1

L\_1\_True:

li $s4, 10

jr $ra

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

#Check Register

Check\_Register:

la $s3, Register\_Check

li $t1, 0 #i=0

Loop1\_Reg:

add $a3, $s3, $t1 #load byte cua thanh ghi mau

lb $t3, 0($a3)

addi $t1, $t1, 1

bne $t3, $s7, Loop1\_Reg

li $t0, 0 #So ki tu cua thanh ghi mau

Loop2\_Reg:

add $a3, $s3, $t1 #load byte cua thanh ghi mau

lb $t3, 0($a3)

add $a2, $s2, $t0 #Load byte cua thanh ghi can check

lb $t2, 0($a2)

beq $t3, $s7, Check\_Reg

beq $t3, $s6, False\_code

bne $t2, $t3, Loop1\_Reg\_them #Kiem tra xem thanh ghi check va thanh ghi mau co giong nhau khong

beq $t2, $t3, Loop2\_Reg\_them

End\_Loop\_Reg:

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

Check\_Reg:

addi $t0, $t0, -1

beq $s0, $t0, Reg\_True

j Loop1\_Reg

Loop1\_Reg\_them:

addi $t1, $t1, 1

j Loop1\_Reg

Loop2\_Reg\_them:

addi $t1, $t1, 1

addi $t0, $t0, 1

j Loop2\_Reg

Reg\_True:

add $t8, $zero, $ra

jal Right\_Register

jr $t8

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

#Check Number

Check\_Number:

li $t1, 0 #i = 0

j Check\_Mark

Check\_Mark\_done:

add $a2, $s2, $t1 #Kiem tra so dau tien

lb $t2, 0($a2)

li $t5, 10 #t5 = newline

beq $t2, $t5, False\_code

beq $t2, $zero, False\_code

li $t5, 48 #t5 = zero

bne $t2, $t5, Loop\_Number\_1

slti $t4, $t2, 48

bne $t4, $zero, False\_code

slti $t4, $t2, 58

beq $t4, $zero, False\_code

addi $t1, $t1, 1 #Kiem tra so thu hai(co the la x trong so hexa)

add $a2, $s2, $t1

lb $t2, 0($a2)

beq $t2, $zero, Right\_Number

li $t5, 120

beq $t2, $t5, Loop\_Number\_them

li $t5, 88

beq $t2, $t5, Loop\_Number\_them

slti $t4, $t2, 48

bne $t4, $zero, False\_code

slti $t4, $t2, 58

beq $t4, $zero, False\_code

Loop\_Number:

add $a2, $s2, $t1

lb $t2, 0($a2)

beq $t2, $zero, Right\_Number

li $t5, 48

beq $t2, $t5, Loop\_Number\_them

li $t5, 49

beq $t2, $t5, Loop\_Number\_them

li $t5, 50

beq $t2, $t5, Loop\_Number\_them

li $t5, 51

beq $t2, $t5, Loop\_Number\_them

li $t5, 52

beq $t2, $t5, Loop\_Number\_them

li $t5, 53

beq $t2, $t5, Loop\_Number\_them

li $t5, 54

beq $t2, $t5, Loop\_Number\_them

li $t5, 55

beq $t2, $t5, Loop\_Number\_them

li $t5, 56

beq $t2, $t5, Loop\_Number\_them

li $t5, 57

beq $t2, $t5, Loop\_Number\_them

li $t5, 65

beq $t2, $t5, Loop\_Number\_them

li $t5, 66

beq $t2, $t5, Loop\_Number\_them

li $t5, 67

beq $t2, $t5, Loop\_Number\_them

li $t5, 68

beq $t2, $t5, Loop\_Number\_them

li $t5, 69

beq $t2, $t5, Loop\_Number\_them

li $t5, 70

beq $t2, $t5, Loop\_Number\_them

li $t5, 97

beq $t2, $t5, Loop\_Number\_them

li $t5, 98

beq $t2, $t5, Loop\_Number\_them

li $t5, 99

beq $t2, $t5, Loop\_Number\_them

li $t5, 100

beq $t2, $t5, Loop\_Number\_them

li $t5, 101

beq $t2, $t5, Loop\_Number\_them

li $t5, 102

beq $t2, $t5, Loop\_Number\_them

j False\_code

Loop\_Number\_1:

add $a2, $s2, $t1

lb $t2, 0($a2)

beq $t2, $zero, Right\_Number

li $t5, 48

beq $t2, $t5, Loop\_Number\_them\_1

li $t5, 49

beq $t2, $t5, Loop\_Number\_them\_1

li $t5, 50

beq $t2, $t5, Loop\_Number\_them\_1

li $t5, 51

beq $t2, $t5, Loop\_Number\_them\_1

li $t5, 52

beq $t2, $t5, Loop\_Number\_them\_1

li $t5, 53

beq $t2, $t5, Loop\_Number\_them\_1

li $t5, 54

beq $t2, $t5, Loop\_Number\_them\_1

li $t5, 55

beq $t2, $t5, Loop\_Number\_them\_1

li $t5, 56

beq $t2, $t5, Loop\_Number\_them\_1

li $t5, 57

j False\_code

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

Right\_Number:

add $t8, $zero, $ra

jal Print\_Right\_Number

jr $t8

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

Check\_Mark: #Ham kiem tra dau cua imm

add $a2, $s2, $t1 #Kiem tra xem ki tu dau tien cua Imm co phai dau + hay - khong?

lb $t2, 0($a2)

li $t5, 43 #t5 =43 ~ '+'

beq $t2, $t5, Check\_Mark\_them

li $t5, 45 #t5 =45 ~ '-'

beq $t2, $t5, Check\_Mark\_them

j Check\_Mark\_done

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

#Check Sign\_ExtImm

Check\_Sign\_ExtImm:

add $t8, $zero, $ra #Luu dia chi tro ve chuong trinh vao -> t8

jal Split\_Sign\_ExtImm

jal Check\_Number

jal Split\_Sign\_ExtImm

jal Check\_Parentheses\_1

jal Split\_Sign\_ExtImm

jal Check\_Register

jal Split\_Sign\_ExtImm

jal Check\_Parentheses\_2

addi $t5, $zero, 10

beq $t9, $t5, Right\_code

addi $t5, $zero, 0

beq $t9, $t5, Right\_code

addi $t5, $zero, 41 #t5 ~ ')'

beq $t9, $t5, Right\_code

j False\_code

#Check\_Parentheses\_1 Kiem tra dau (

Check\_Parentheses\_1:

li $t1, 0 #i = 0

add $a2, $s2, $t1

lb $t2, 0($a2)

li $t5, 40

bne $t2, $t5, False\_code

addi $t1, $t1, 1

add $a2, $s2, $t1

lb $t2, 0($a2)

bne $zero, $t2, False\_code

jr $ra

#Check\_Parentheses\_2 Kiem tra dau )

Check\_Parentheses\_2:

li $t1, 0 #i = 0

add $a2, $s2, $t1

lb $t2, 0($a2)

li $t5, 41

bne $t2, $t5, False\_code

addi $t1, $t1, 1

add $a2, $s2, $t1

lb $t2, 0($a2)

bne $zero, $t2, False\_code

jr $ra

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

#Check Label

Check\_Label:

li $t1, 0 #i = 0

add $a2, $s2, $t1

lb $t2, 0($a2)

beq $t2, $zero, False\_code

li $t5, 10 #t5 = 'New line'

beq $t2, $t5, False\_code

slti $t4, $t2, 48

bne $t4, $zero, False\_code

li $t5, 58

beq $t2, $t5, False\_code

li $t5, 59

beq $t2, $t5, False\_code

li $t5, 60

beq $t2, $t5, False\_code

li $t5, 61

beq $t2, $t5, False\_code

li $t5, 62

beq $t2, $t5, False\_code

li $t5, 63

beq $t2, $t5, False\_code

li $t5, 64

beq $t2, $t5, False\_code

li $t5, 91

beq $t2, $t5, False\_code

li $t5, 92

beq $t2, $t5, False\_code

li $t5, 93

beq $t2, $t5, False\_code

li $t5, 94

beq $t2, $t5, False\_code

li $t5, 96

beq $t2, $t5, False\_code

slti $t4, $t2, 123

beq $t4, $zero, False\_code

addi $t1, $t1, 1

Loop\_Label:

add $a2, $s2, $t1

lb $t2, 0($a2)

beq $t2, $zero, True\_Label

li $t5, 10 #t5 = 'New line'

beq $t2, $t5, True\_Label

slti $t4, $t2, 48

bne $t4, $zero, False\_code

li $t5, 58

beq $t2, $t5, False\_code

li $t5, 59

beq $t2, $t5, False\_code

li $t5, 60

beq $t2, $t5, False\_code

li $t5, 61

beq $t2, $t5, False\_code

li $t5, 62

beq $t2, $t5, False\_code

li $t5, 63

beq $t2, $t5, False\_code

li $t5, 64

beq $t2, $t5, False\_code

li $t5, 91

beq $t2, $t5, False\_code

li $t5, 92

beq $t2, $t5, False\_code

li $t5, 93

beq $t2, $t5, False\_code

li $t5, 94

beq $t2, $t5, False\_code

li $t5, 96

beq $t2, $t5, False\_code

slti $t4, $t2, 123

beq $t4, $zero, False\_code

addi $t1, $t1, 1

j Loop\_Label

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

True\_Label:

jal Print\_Right\_Label

j Right\_code

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

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

#Output

False\_opcode:

#Print "Opcode"

li $v0, 4

la $a0, Message2

syscall

nop

#Print Opcode Input

li $v0, 4

add $a0, $zero, $s2

syscall

nop

#Print "Khong hop le!"

li $v0, 4

la $a0, Message4

syscall

nop

jal False\_code

j End\_main

Right\_opcode:

#Print "Opcode"

li $v0, 4

la $a0, Message2

syscall

nop

#Print Opcode Input

li $v0, 4

add $a0, $zero, $s2

syscall

nop

#Print ", hop le!"

li $v0, 4

la $a0, Message3

syscall

nop

jr $ra

Right\_Register:

#Print "\n"

li $v0, 4

la $a0, Message7

syscall

nop

#Print "Thanh ghi"

li $v0, 4

la $a0, Message8

syscall

nop

#Print Register Input

li $v0, 4

add $a0, $zero, $s2

syscall

nop

#Print ", hop le!"

li $v0, 4

la $a0, Message3

syscall

nop

jr $ra

Print\_Right\_Number:

#Print "\n"

li $v0, 4

la $a0, Message7

syscall

nop

#Print "So "

li $v0, 4

la $a0, Message9

syscall

nop

#Print so trong hang doi

li $v0, 4

add $a0, $zero, $s2

syscall

nop

#Print ", hop le!"

li $v0, 4

la $a0, Message3

syscall

nop

jr $ra

Print\_Right\_Label:

#Print "\n"

li $v0, 4

la $a0, Message7

syscall

nop

#Print "So "

li $v0, 4

la $a0, Message10

syscall

nop

#Print label trong hang doi

li $v0, 4

add $a0, $zero, $s2

syscall

nop

#Print ", hop le!"

li $v0, 4

la $a0, Message3

syscall

nop

jr $ra

Right\_code:

#Print "Right code"

li $v0, 4

la $a0, Message5

syscall

nop

j End\_main

False\_code:

#Print "False code"

li $v0, 4

la $a0, Message6

syscall

nop

j End\_main

End\_main:

Run\_Again: li $v0, 50

la $a0, Message11

syscall

nop

beq $a0, $zero, clear

nop

j exit

nop

# clear: dua string ve trang thai ban dau de thuc hien lai qua trinh

clear: add $s3, $zero, $s1

Loop\_Null:

lb $t3, 0($s3)

li $t5, 10

beq $t3, $t5, Loop\_Null\_them

nop

sb $zero, 0($s3)

addi $s3, $s3, 1

j Loop\_Null

Loop\_Null\_them:

sb $zero, 0($s3)

j start

nop

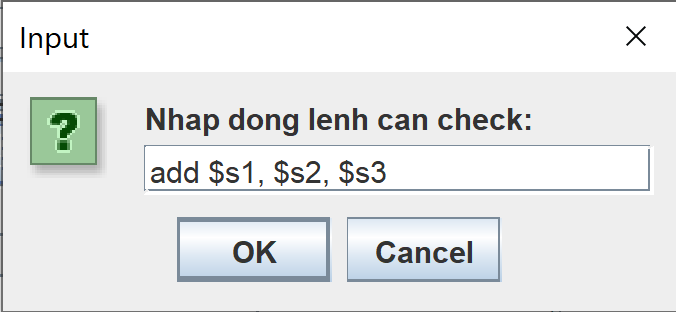
exit: li $v0, 10

syscall

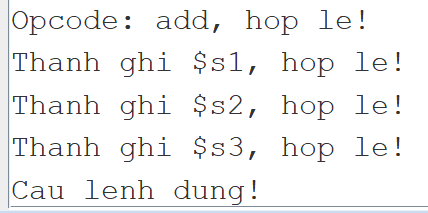
**IV. Kết quả**

**1.Kiểm tra với câu lệnh đúng**

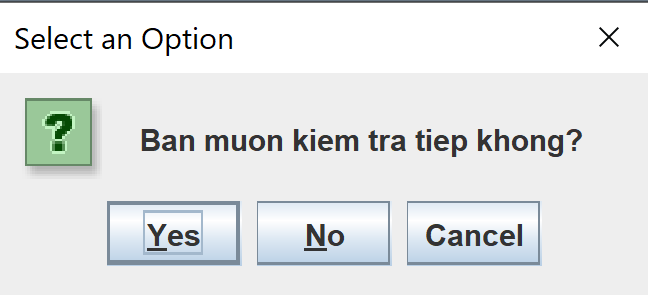
a, Nhập add $s1, $s2, $s3



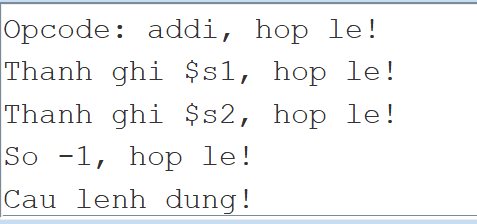
Kết quả thu được:



Và chương trình hỏi có muốn tiếp tục kiểm tra lệnh không

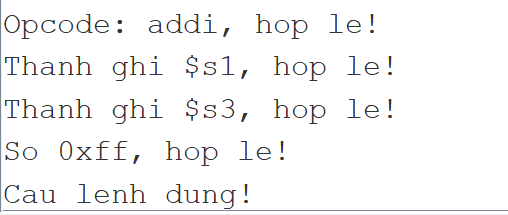


b, Nhập addi $s1, $s2, -1

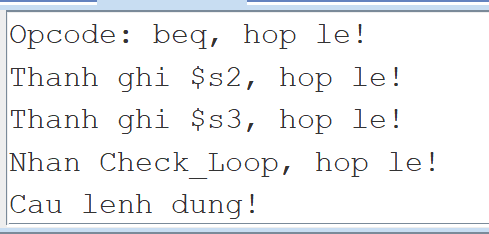


Ta thử tiếp với dạng số hexa:

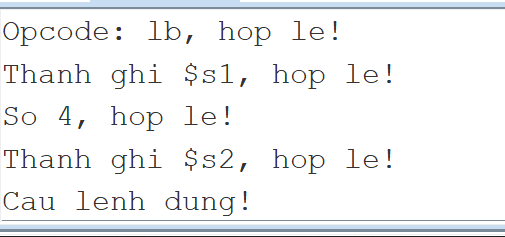
Nhập addi $s1, $s3, 0xff



c, Nhập beq $s2, $s3, Check\_Loop



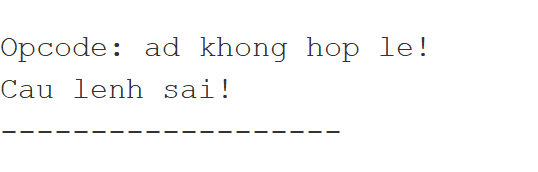
d, Nhập lb $s1, 4($s2)



**2.Nhập lệnh sai**

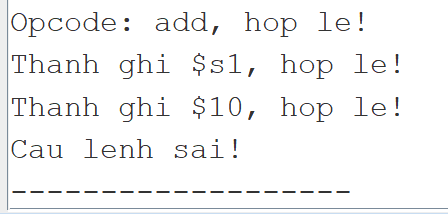
a, Opcode sai

Nhập: ad $s1, $s2, $s3



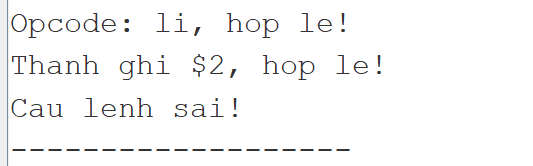
b, Thanh ghi sai

Nhập: add $s1, $10, $ss



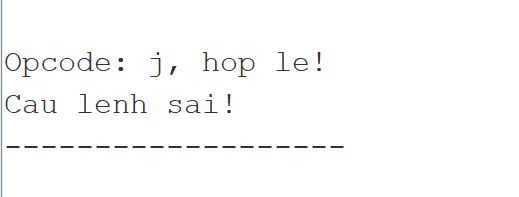
c, Số sai

Nhập: li $2, 1a



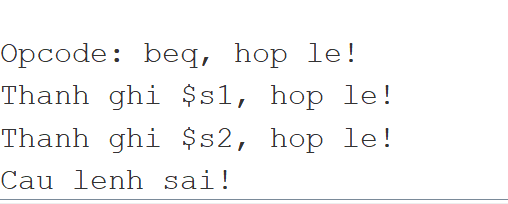
d, Nhãn sai

Nhập: j a@



e, Cấu trúc sai

Nhập: beq $s1, $s2, $s3



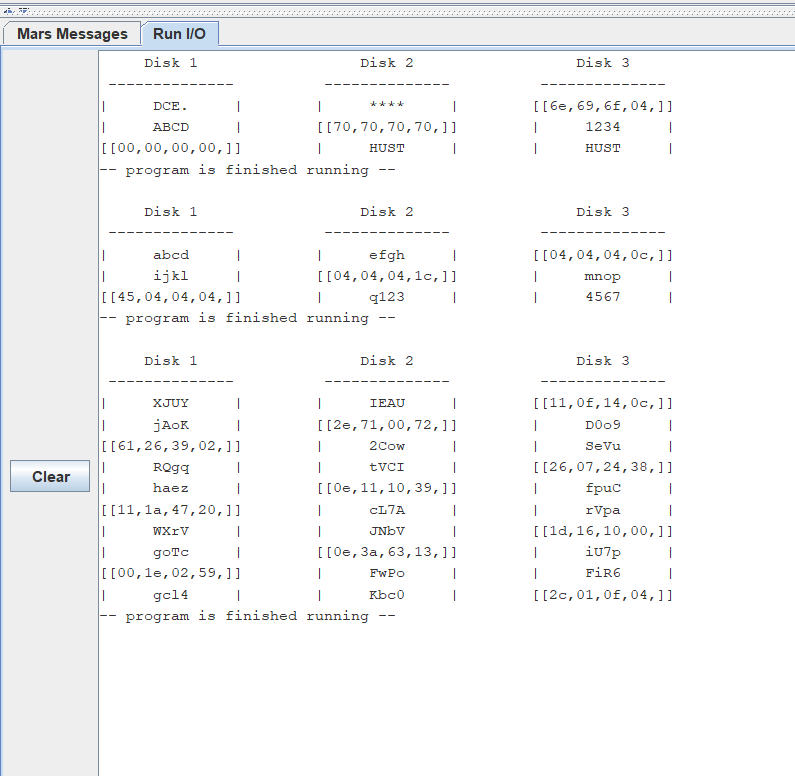
**BÀI 8: MÔ PHỎNG Ổ ĐĨA RAID5**

**Mô tả:**

Hệ thống RAID 5 được thiết kế với khả năng khôi phục dữ liệu.

RAID 5 cần tối thiểu 3 ổ cứng. Các khối parity được lưu trữ đồng đều trên từng ổ cứng.

Ví dụ trong bài tập này, các khối parity được tạo bằng cách xor từng 4 byte dữ liệu liền kề với nhau. Các khối parity sẽ được lưu trữ lần lượt ở ổ Disk 3, rồi Disk 2, Disk 1. Như vậy, khi dữ liệu của 1 trong 3 đĩa bị mất, ta vẫn có thể khôi phục lại dữ diệu từ 2 đĩa còn lại bằng cách thực hiện lại phép xor.



**Phân tích cách làm**

Chương trình cần thực hiện những chức năng sau:

* Yêu cầu người dung nhập vào một xâu ( có kiểm tra độ dài xâu phải là bội của 8 )
* Phân chia xâu ra thành từng khối 4 byte
* Lưu các khối này vào ô nhớ mô phỏng các ổ cứng Disk 1, Disk 2, Disk 3
* Hiển thị ra màn hình dữ liệu đang được lưu trữ trong 3 đĩa

**Nhập và kiểm tra độ dài xâu có là bội của 8:**

Chương trình gọi câu lệnh hệ thống (syscall 54 ) để cho phép nhập vào xâu đầu vào và lưu nó vào msg1. Cần lưu ý lệnh syscall 54 sẽ lưu xâu nhập vào kết thúc bằng ký tự xuống dòng “\n”.

* Sử dụng một vòng lặp để duyệt xâu để lấy độ dài xâu. Mỗi lần duyệt qua một phần tử thì +1 vào thanh ghi lưu độ dài, cho đến khi gặp kí tự xuống dòng “\n”

la $s1, msg1

addi $s7, $zero, 0xA # 0xA = '\n'

li $s2, 0

get\_lenght:

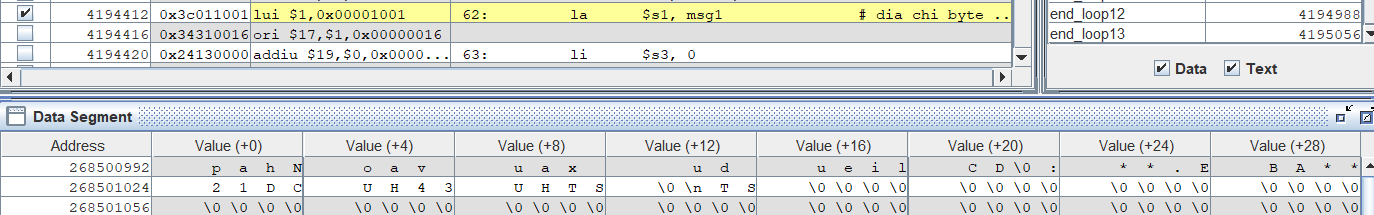
lb $t1, 0($s1)

beq $t1, $s7, check

addi $s2, $s2, 1

addi $s1, $s1, 1

j get\_lenght



* Duyệt xâu để lấy độ dài xâu rồi chia lấy dư cho 8 (thủ tục check), nếu kết quả khác 0 thì sẽ nhảy đến đoạn lệnh báo lỗi.

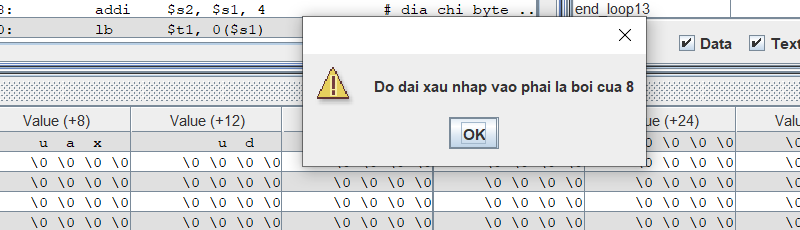
check:

addi $s7, $zero, 8

div $s2, $s7

mfhi $s2 # so du cua phep chia duoc luu o thanh ghi hi

bne $s2, $zero, error



**Duyệt xâu theo từng khối 4 byte:**

* Trong mô hình Raid 5, có dữ liệu sẽ được lưu tuần tự vào 2 đĩa

(VD: ta có xâu “ABCDEFGH” thì “ABCD” sẽ ghi vào Disk 1 còn “EFGH” sẽ lưu vào Disk 2, Disk còn lại lưu giá trị xor của 2 xâu con)

* Ở đây ta sử dụng 2 thanh ghi $s1 và $s2 để duyệt xâu theo từng khối 4 byte. Thanh ghi $s1 trỏ đến phần tử thứ n thì thanh ghi $s2 sẽ trỏ đến phần tử thứ n+4. Khi $s1 tăng thì $s2 cũng tăng. Như vậy có thể đọc và ghi 4 byte song song vào Disk 1 và 2.
* Khi 2 thanh ghi $s1 và $s2 đã duyệt xong một khối 4 byte, cần cộng thêm vào cả hai con trỏ 4 đơn vị để nhảy đến khối tiếp theo, tránh bị lặp dữ liệu. Nếu không, $s1 sẽ đọc vào cả phần dữ liệu mà $s2 đã đọc.
* Duyệt xâu như vậy cho tới khi gặp ký tự xuống dòng.
* Mỗi lần duyệt thì ký tự tại địa chỉ $s1, $s2 sẽ được lấy ra bằng lệnh load byte và ghi vào $t1, $t2
* Mỗi khi lấy ra 1 byte từ $s1 và 1 byte từ $s2, thực hiện xor 2 byte này với nhau để được phần parity, sau đó nhảy đến thủ tục store để lưu trữ vào từng ổ đĩa.

la $s1, msg1 # dia chi byte thu n

li $s3, 0

Load\_4bytes:

addi $s7, $zero, 0xA

addi $s2, $s1, 4 # dia chi byte thu n + 4

lb $t1, 0($s1) # gia tri byte thu n

lb $t2, 0($s2) # gia tri byte thu n + 4

beq $t1, $s7, print

xor $t0, $t1, $t2 # gia tri xor parity

jal store #luu gia tri vao disk

addi $s1, $s1, 1 # advance

addi $s3, $s3, 1

bne $s3, $a3, Load\_4bytes # i > 4 ? tiep tuc lap

add $s3, $zero, $zero # reset i = 0

addi $s1, $s1, 4

addi $t8, $t8, 1 # So block 8 byte da doc

j Load\_4bytes

**Lưu trữ vào từng ổ đĩa:**

Tiếp đến là thủ tục store, nhằm lưu trữ 2 khối 4 byte và 1 khối parity vào các ổ đĩa

* Đầu tiên, cần khai báo địa chỉ cho 3 ổ đĩa ngay từ đầu chương trình khoảng trống giữa các ổ phải đủ lớn để tránh dữ liệu của ổ trước ghi đè lên ổ sau. Ở đây ta khai báo mỗi ổ đĩa có 100 byte để lưu trữ

Graphical user interface, text, application, chat or text message

Description automatically generated

* Ở RAID 5, điều cần lưu ý là các khối parity phải được phân bố đều giữa các đĩa để đảm bảo chức năng khôi phục dữ liệu.

VD: sau khi đọc 8 byte đầu, Disk 1 và 2 sẽ lưu lần lượt 4 byte dữ liệu, phần parity sẽ lưu vào

Disk 3. Ở lần tiếp theo, parity sẽ được lưu vào Disk 2, và lần tiếp theo sẽ phải là Disk 1.

* Vì vậy trong đoạn chương trình, ta sẽ dùng một thanh ghi $t8 để đếm mỗi lần mà $s1, $s2 đã đọc một khối 4byte dữ liệu, sau đó chia lấy dư cho 3.

store:

add $t9, $t8, $zero

addi $s7, $zero, 3

div $t9, $s7 #$t9 chia 3 du 0, 1, 2 ?

mfhi $t9

addi $s7, $zero, 0

beq $t9, $zero, store\_0

addi $s7, $zero, 1

beq $t9, $s7, store\_1

addi $s7, $zero, 2

beq $t9, $s7, store\_2

* Trong trường hợp:
* Kết quả bằng 0, gọi thủ tục store\_0:
* Lưu giá trị đọc được ở $t1 vào Disk 1
* Lưu giá trị đọc được ở $t2 vào Disk 2
* Lưu giá trị parity xor được vào Disk 3

store\_0:

sb $t1, 0($s4)

addi $s4, $s4, 1

sb $t2, 0($s5)

addi $s5, $s5, 1

sb $t0, 0($s6)

addi $s6, $s6, 1 # luu parity vao disk1

jr $ra

* Kết quả bằng 1, gọi thủ tục store\_1:
* Lưu giá trị đọc được ở $t1 vào Disk 1
* Lưu giá trị đọc được ở $t2 vào Disk 3
* Lưu giá trị parity xor được vào Disk 2

store\_1:

sb $t1, 0($s4)

addi $s4, $s4, 1

sb $t0, 0($s5) # luu parity vao disk 2

addi $s5, $s5, 1

sb $t2, 0($s6)

addi $s6, $s6, 1

jr $ra

* Kết quả bằng 2, thực hiện thủ tục store\_2:
* Lưu giá trị đọc được ở $t1 vào Disk 2
* Lưu giá trị đọc được ở $t2 vào Disk 3
* Lưu giá trị parity xor được vào Disk 1

store\_2:

sb $t0, 0($s4) # luu parity vao disk 1

addi $s4, $s4, 1

sb $t1, 0($s5)

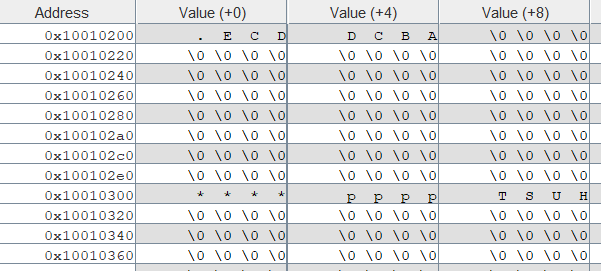
addi $s5, $s5, 1

sb $t2, 0($s6)

addi $s6, $s6, 1

jr $ra

*Kết quả : quan sát vùng data segment*



Table

Description automatically generated

**In dữ liệu từ ổ đĩa ra màn hình:**

* In ra các đầu mục “Disk 1”, “Disk 2”, “Disk 3” và các dòng kẻ, dấu đóng mở ngoặc.

A picture containing text

Description automatically generated

* Sau đó, để in ra được dữ liệu từ ổ đĩa, ta cần load byte từng ký tự một để in ra màn hình. Việc in ra được thực hiện bằng câu lệnh hệ thống syscall 4 và 11
* Tuy nhiên, theo đề bài, định dạng in ra của khối dữ liệu và khối parity là khác nhau:
* Khối dữ liệu in ra sử dụng dấu sổ thẳng: “| DCE. |”
* Khối parity in ra sử dụng 2 dấu ngoặc vuông và biểu diễn bằng hệ 16: “[[ 6e,69,6f,04 ]] ”
* Vì vậy , ta sử dụng lại cách chia lấy dư như ở thủ tục ghi dữ liệu để in ra 2 dạng dữ liệu khác nhau này.
* Sử dụng vòng lặp để có thể in ra hết dữ liệu trong đĩa: sẽ ghi ra đến khi biến đếm đạt đến số lần đọc dữ liệu trong xâu ( được lưu trong thanh ghi $t8 ở đoạn chương trình đọc dữ liệu)

printData:

la $s4, DISK\_1

la $s5, DISK\_2

la $s6, DISK\_3

addi $t7, $zero, 3

add $t6, $zero, $zero

add $t9, $zero, $zero

main\_loop:

addi $a3, $zero, 3

div $t9, $a3 # $s7 = 3

mfhi $t9

beq $t9, $zero, printData\_1

addi $a3, $zero, 1

beq $t9, $a3, printData\_2

addi $a3, $zero, 2

beq $t9, $a3, printData\_3

* Đoạn lệnh in ra từng ổ đĩa được thiết kế như sau:

printData\_1:

addi $t6, $t6, 1

add $s7, $zero, $zero

jal Break\_line

jal Print\_ke\_thang # "| "

jal Print\_5space

#Print disk 1

loop11:

lb $a0, 0($s4)

li $v0, 11

syscall

beq $s7, $t7, end\_loop11

addi $s4, $s4, 1

addi $s7, $s7, 1

j loop11

end\_loop11:

jal Print\_5space

jal Print\_ke\_thang #" |"

add $s7, $zero, $zero

jal Print\_tab

jal Print\_tab

jal Print\_ke\_thang # "| "

jal Print\_5space

#Print disk 2

loop12:

lb $a0, 0($s5)

li $v0, 11

syscall

beq $s7, $t7, end\_loop12

addi $s5, $s5, 1

addi $s7, $s7, 1

j loop12

end\_loop12:

jal Print\_5space

jal Print\_ke\_thang #" |"

add $s7, $zero, $zero

jal Print\_tab

jal Print\_tab

jal Print\_mo\_ngoac\_vuong #"[[ "

jal Print\_mo\_ngoac\_vuong

jal Print\_space

#Print disk 3 (parity)

loop13:

lb $t5, 0($s6)

jal Hexa

beq $s7, $t7, end\_loop13

li $a0, 0x2C

li $v0, 11

syscall

addi $s6, $s6, 1

addi $s7, $s7, 1

j loop13

end\_loop13:

jal Print\_space

jal Print\_dong\_ngoac\_vuong #" ]]"

jal Print\_dong\_ngoac\_vuong

beq $t6, $t8, end

addi $s4, $s4, 1

addi $s5, $s5, 1

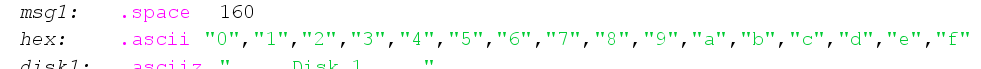
addi $s6, $s6, 1

addi $t9, $t9, 1

j main\_loop

* Thủ tục cho phép in ra dữ liệu theo số hệ 16:

Trước tiên khai báo một mảng các chữ số hệ 16:



Thủ tục xử lý số Hexa:

* Lấy ra chữ số hệ 16 lưu trong ổ đĩa chứa parity và ghi vào $t5. Sau đó cần lấy ra 2 chữ số cuối bằng cách dịch phải 2 bit rồi lưu kết quả vào $t4.
* Như vậy, ta có 2 thanh ghi $t4 và $t5 lưu 2 chữ số hệ 16 cần đọc ra.
* Sau đó đối chiếu vào mảng hex để lấy được chữ số tương ứng dạng char và in ra bằng lệnh syscall 11.

Hexa:

srl $t4, $t5, 4

andi $t4, $t4, 0x0000000f

la $a1, hex

add $a1, $a1, $t4

lb $a0, 0($a1)

li $v0, 11

syscall

andi $t5, $t5, 0x0000000f

la $a1, hex

add $a1, $a1, $t5

lb $a0, 0($a1)

li $v0, 11

syscall

jr $ra

Kết quả:

Graphical user interface, application

Description automatically generated

**Đoạn chương trình hoàn thiện:**

.eqv DISK\_1 0x10010200

.eqv DISK\_2 0x10010300

.eqv DISK\_3 0x10010400

.data

msg0: .asciiz "Nhap vao xau du lieu:"

msg1: .space 160

hex: .ascii "0","1","2","3","4","5","6","7","8","9","a","b","c","d","e","f"

disk1: .asciiz " Disk 1 "

disk2: .asciiz " Disk 2 "

disk3: .asciiz " Disk 3 "

line: .asciiz " -------------- "

tab: .asciiz "\t"

space: .asciiz " "

brkline: .asciiz "\n"

err: .asciiz "Do dai xau nhap vao phai la boi cua 8"

.text

la $s4, DISK\_1

la $s5, DISK\_2

la $s6, DISK\_3

li $a3, 4

#------ Nhap vao du lieu-------#

input:

la $a0, msg0

la $a1, msg1

la $a2, 160

li $v0, 54

syscall

#-------- Kiem tra do dai xau -------#

#Do dai xau phai la boi cua 8

la $s1, msg1

addi $s7, $zero, 0xA # 0xA = '\n'

li $s2, 0

get\_lenght:

lb $t1, 0($s1)

beq $t1, $s7, check

addi $s2, $s2, 1

addi $s1, $s1, 1

j get\_lenght

check:

addi $s7, $zero, 8

div $s2, $s7

mfhi $s2

bne $s2, $zero, error

#------- Load byte ---------#

# Load tung 8 byte mot luot, va luu vao 2 disk,

# dong thoi xor va luu vao disk con lai

# $s1: Dia chi byte thu n

# $s2: Dia chi byte thu n + 4

# $t1, $t2: gia tri byt thu n va n + 4

# $t0: gia tri xor parity

la $s1, msg1 # dia chi byte thu n

li $s3, 0

Load\_4bytes:

addi $s7, $zero, 0xA

addi $s2, $s1, 4 # dia chi byte thu n + 4

lb $t1, 0($s1) # gia tri byte thu n

lb $t2, 0($s2) # gia tri byte thu n + 4

beq $t1, $s7, print

xor $t0, $t1, $t2 # gia tri xor parity

jal store #luu gia tri vao disk

addi $s1, $s1, 1 # advance

addi $s3, $s3, 1

bne $s3, $a3, Load\_4bytes # i > 4 ? tiep tuc lap

add $s3, $zero, $zero # reset i = 0

addi $s1, $s1, 4

addi $t8, $t8, 1 # So block 8 byte da doc

j Load\_4bytes

#------- Luu tru ---------#

# Trong Raid 5, cac block parity phai duoc phan bo deu

# ==> Moi lan luu tru se luu vao mot trong 3 disk luan phien

# Su dung cach chia co du

# VD: lan ghi thu 2 ==> 2 : 3 du 2 ==> ghi vao DISK\_3

store:

add $t9, $t8, $zero

addi $s7, $zero, 3

div $t9, $s7 #$t9 chia 3 du 0, 1, 2 ?

mfhi $t9

addi $s7, $zero, 0

beq $t9, $zero, store\_0

addi $s7, $zero, 1

beq $t9, $s7, store\_1

addi $s7, $zero, 2

beq $t9, $s7, store\_2

#-----Store parity to Disk 1------#

store\_0:

sb $t1, 0($s4)

addi $s4, $s4, 1

sb $t2, 0($s5)

addi $s5, $s5, 1

sb $t0, 0($s6)

addi $s6, $s6, 1 # luu parity vao disk1

jr $ra

#-----Store parity to Disk 2------#

store\_1:

sb $t1, 0($s4)

addi $s4, $s4, 1

sb $t0, 0($s5) # luu parity vao disk 2

addi $s5, $s5, 1

sb $t2, 0($s6)

addi $s6, $s6, 1

jr $ra

#-----Store parity to Disk 3------#

store\_2:

sb $t0, 0($s4)

addi $s4, $s4, 1

sb $t1, 0($s5)

addi $s5, $s5, 1

sb $t2, 0($s6)

addi $s6, $s6, 1

jr $ra

#--------Print--------#

# in tung ky tu ra man hinh

print:

#----------Menu chinh---------#

#disk 1, 2, 3

# disk 1

la $a0, disk1 # "Disk 1"

li $v0, 4

syscall

la $a0, tab # tab

li $v0, 4

syscall

#disk 2

la $a0, tab # tab

li $v0, 4

syscall

la $a0, disk2

li $v0, 4

syscall

la $a0, tab

li $v0, 4

syscall

# disk 3

la $a0, tab

li $v0, 4

syscall

la $a0, disk3

li $v0, 4

syscall

la $a0, tab

li $v0, 4

syscall

# line

la $a0, brkline

li $v0, 4

syscall

add $a3, $zero, $zero

addi $s7, $zero, 3

printLine\_1:

addi $a3, $a3, 1

la $a0, line

li $v0, 4

syscall

la $a0, tab

li $v0, 4

syscall

la $a0, tab

li $v0, 4

syscall

beq $a3, $s7, printData

j printLine\_1

#-------- In ra data-----------#

# Can biet duoc vi tri cua byte parity (disk 1, 2 hay 3)

# lay bien chay chia lay du cho 3

printData:

la $s4, DISK\_1

la $s5, DISK\_2

la $s6, DISK\_3

addi $t7, $zero, 3

add $t6, $zero, $zero

add $t9, $zero, $zero

main\_loop:

addi $a3, $zero, 3

div $t9, $a3 # $s7 = 3

mfhi $t9

beq $t9, $zero, printData\_1

addi $a3, $zero, 1

beq $t9, $a3, printData\_2

addi $a3, $zero, 2

beq $t9, $a3, printData\_3

#-----------PRINT\_I -------------#

# disk 1, 2 chua du lieu

# disk 3 parity

printData\_1:

addi $t6, $t6, 1

add $s7, $zero, $zero

jal Break\_line

jal Print\_ke\_thang # "| "

jal Print\_5space

#Print disk 1

loop11:

lb $a0, 0($s4)

li $v0, 11

syscall

beq $s7, $t7, end\_loop11

addi $s4, $s4, 1

addi $s7, $s7, 1

j loop11

end\_loop11:

jal Print\_5space

jal Print\_ke\_thang #" |"

add $s7, $zero, $zero

jal Print\_tab

jal Print\_tab

jal Print\_ke\_thang # "| "

jal Print\_5space

#Print disk 2

loop12:

lb $a0, 0($s5)

li $v0, 11

syscall

beq $s7, $t7, end\_loop12

addi $s5, $s5, 1

addi $s7, $s7, 1

j loop12

end\_loop12:

jal Print\_5space

jal Print\_ke\_thang #" |"

add $s7, $zero, $zero

jal Print\_tab

jal Print\_tab

jal Print\_mo\_ngoac\_vuong #"[[ "

jal Print\_mo\_ngoac\_vuong

jal Print\_space

#Print disk 3 (parity)

loop13:

lb $t5, 0($s6)

jal Hexa

beq $s7, $t7, end\_loop13

li $a0, 0x2C

li $v0, 11

syscall

addi $s6, $s6, 1

addi $s7, $s7, 1

j loop13

end\_loop13:

jal Print\_space

jal Print\_dong\_ngoac\_vuong #" ]]"

jal Print\_dong\_ngoac\_vuong

beq $t6, $t8, end

addi $s4, $s4, 1

addi $s5, $s5, 1

addi $s6, $s6, 1

addi $t9, $t9, 1

j main\_loop

#-----------PRINT\_II -------------#

# disk 1, 3 chua du lieu

# disk 2 parity

printData\_2:

addi $t6, $t6, 1

add $s7, $zero, $zero

jal Break\_line

jal Print\_ke\_thang # "| "

jal Print\_5space

#Print disk 1

loop21:

lb $a0, 0($s4)

li $v0, 11

syscall

beq $s7, $t7,end\_loop21

addi $s4, $s4, 1

addi $s7, $s7, 1

j loop21

end\_loop21:

jal Print\_5space

jal Print\_ke\_thang

add $s7, $zero, $zero

jal Print\_tab

jal Print\_tab

jal Print\_mo\_ngoac\_vuong #"[[ "

jal Print\_mo\_ngoac\_vuong

jal Print\_space

#Print disk 2(parity)

loop22:

lb $t5, 0($s5)

jal Hexa

beq $s7, $t7, end\_loop22

li $a0, 0x2C

li $v0, 11

syscall

addi $s5, $s5, 1

addi $s7, $s7, 1

j loop22

end\_loop22:

jal Print\_space

jal Print\_dong\_ngoac\_vuong

jal Print\_dong\_ngoac\_vuong #" ]]"

add $s7, $zero, $zero

jal Print\_tab

jal Print\_tab # "| "

jal Print\_ke\_thang

jal Print\_5space

#Print disk 3

loop23:

lb $a0, 0($s6)

li $v0, 11

syscall

beq $s7, $t7, end\_loop23

addi $s6, $s6, 1

addi $s7, $s7, 1

j loop23

end\_loop23:

jal Print\_5space

jal Print\_ke\_thang

beq $t6, $t8, end

addi $s4, $s4, 1

addi $s5, $s5, 1

addi $s6, $s6, 1

addi $t9, $t9, 1

j main\_loop

#-----------PRINT\_III -------------#

# disk 2, 3 chua du lieu

# disk 1 parity

printData\_3:

addi $t6, $t6, 1

add $s7, $zero, $zero

jal Break\_line

jal Print\_mo\_ngoac\_vuong

jal Print\_mo\_ngoac\_vuong

jal Print\_space

#Print disk 1(parity)

loop31:

lb $t5, 0($s4)

jal Hexa

beq $s7, $t7,end\_loop31

li $a0, 0x2C

li $v0, 11

syscall

addi $s4, $s4, 1

addi $s7, $s7, 1

j loop31

end\_loop31:

jal Print\_space

jal Print\_dong\_ngoac\_vuong

jal Print\_dong\_ngoac\_vuong

add $s7, $zero, $zero

jal Print\_tab

jal Print\_tab

jal Print\_ke\_thang

jal Print\_5space

#Print disk 2

loop32:

lb $a0, 0($s5)

li $v0, 11

syscall

beq $s7, $t7, end\_loop32

addi $s5, $s5, 1

addi $s7, $s7, 1

j loop32

end\_loop32:

jal Print\_5space

jal Print\_ke\_thang

add $s7, $zero, $zero

jal Print\_tab

jal Print\_tab

jal Print\_ke\_thang

jal Print\_5space

#Print disk 3

loop33:

lb $a0, 0($s6)

li $v0, 11

syscall

beq $s7, $t7, end\_loop33

addi $s6, $s6, 1

addi $s7, $s7, 1

j loop33

end\_loop33:

jal Print\_5space

jal Print\_ke\_thang

beq $t6, $t8, end\_main\_loop

addi $s4, $s4, 1

addi $s5, $s5, 1

addi $s6, $s6, 1

addi $t9, $t9, 1

j main\_loop

end\_main\_loop:

jal Break\_line

add $a3, $zero, $zero

addi $s7, $zero, 3

printLine\_2:

addi $a3, $a3, 1

la $a0, line

li $v0, 4

syscall

la $a0, tab

li $v0, 4

syscall

la $a0, tab

li $v0, 4

syscall

beq $a3, $s7, end

j printLine\_2

#-----------Xu ly so thap luc phan ------------------#

Hexa:

srl $t4, $t5, 4

andi $t4, $t4, 0x0000000f

la $a1, hex

add $a1, $a1, $t4

lb $a0, 0($a1)

li $v0, 11

syscall

andi $t5, $t5, 0x0000000f

la $a1, hex

add $a1, $a1, $t5

lb $a0, 0($a1)

li $v0, 11

syscall

jr $ra

#----------- In ra mot so ky tu dac biet ------------#

Print\_ke\_thang:

li $a0, 0x7C

li $v0, 11

syscall

jr $ra

Print\_space:

li $a0, 0x20

li $v0, 11

syscall

jr $ra

Print\_mo\_ngoac\_vuong:

li $a0, 0x5B

li $v0, 11

syscall

jr $ra

Print\_dong\_ngoac\_vuong:

li $a0, 0x5D

li $v0, 11

syscall

jr $ra

Print\_tab:

la $a0, tab

li $v0, 4

syscall

jr $ra

Print\_5space:

la $a0, space

li $v0, 4

syscall

jr $ra

Break\_line:

la $a0, brkline

li $v0, 4

syscall

jr $ra

error:

la $a0, err

li $a1, 2

li $v0, 55

syscall

end:

li $v0, 10

syscall