ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KỸ THUẬT MÁY TÍNH

NGUYỄN PHƯỚC HƯNG – 21520252

Tổ chức và Cấu trúc Máy tính II/ IT012.M11.1

TP. HỒ CHÍ MINH, 2022

MỤC LỤC

DANH MỤC HÌNH

DANH MỤC BẢNG

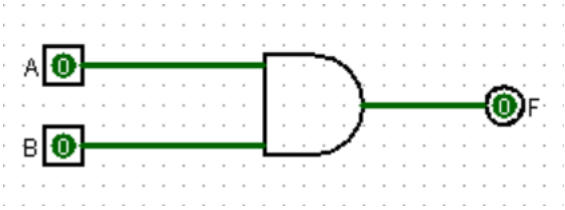
Chương 1. **BÁO CÁO LAB01**

1.1. Báo cáo phần thực hành

1.1.1. Mô phỏng chức năng các cổng luận lý

Cổng AND

Hình 1.1.1.1 AND

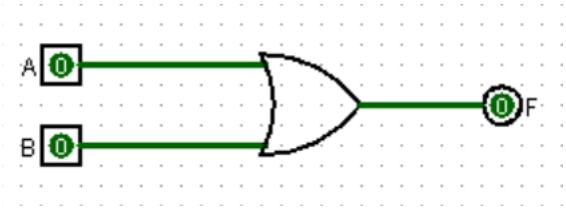


Bảng 1.1.1.1 AND

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

Cổng OR

Hình 1.1.1.2 OR

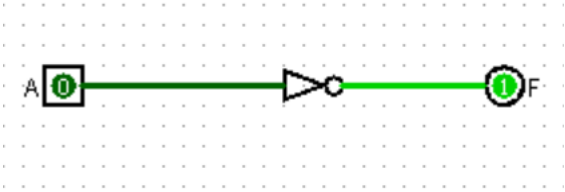


Bảng 1.1.1.2 OR

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

Cổng NOT

Hình 1.1.1.3 NOT

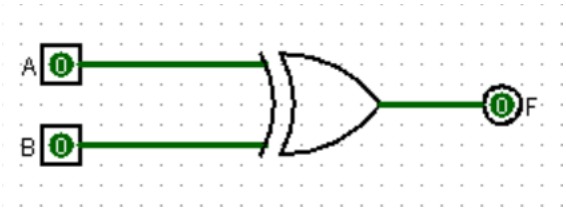


Bảng 1.1.1.3 NOT

A	F
0	1
1	0

Cổng XOR

Hình 1.1.1.4 XOR

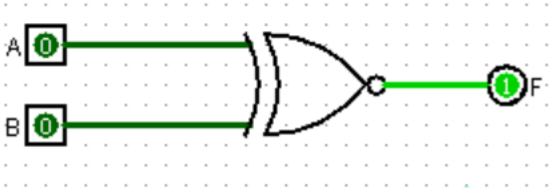


Bảng 1.1.1.4 XOR

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

Cổng XNOR

Hình 1.1.1.5 XNOR

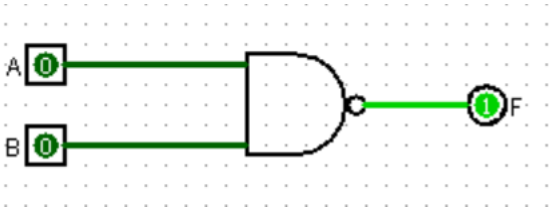


Bảng 1.1.1.5 XNOR

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

Cổng NAND

Hình 1.1.1.6 NAND

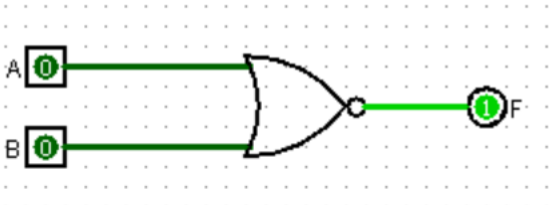


Bảng 1.1.1.6 NAND

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

Cổng NOR

Hình 1.1.1.7 NOR



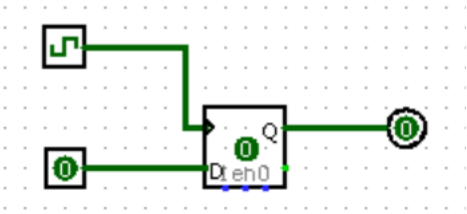
Bảng 1.1.1.7 NOR

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

1.1.2. Mô phỏng chức năng các thiết bị lưu trữ

- D flipflop

Hình 1.1.2.1 D flipflop

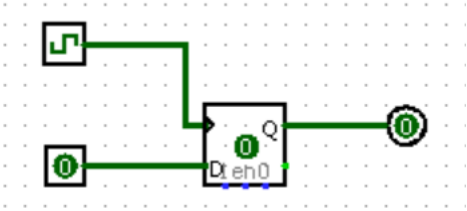


Bảng 1.1.2.1 D flipflop

CLK	D	Q	Q^+
-	0	0	0
-	0	1	1
-	1	0	0
-	1	1	1
↑	0	0	0
↑	0	1	0
↑	1	0	1
↑	1	1	1

- D latch

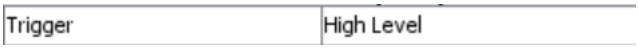
Hình 1.1.2.2 D latch



Bảng 1.1.2.2 D latch

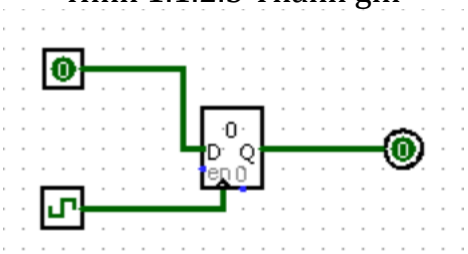
E	D	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Khi sữ dụng D latch trong logisim, ta phải chỉnh attribute trigger sang High level để mô tả đúng cách hoạt động của D latch



- Thanh ghi

Hình 1.1.2.3 Thanh ghi



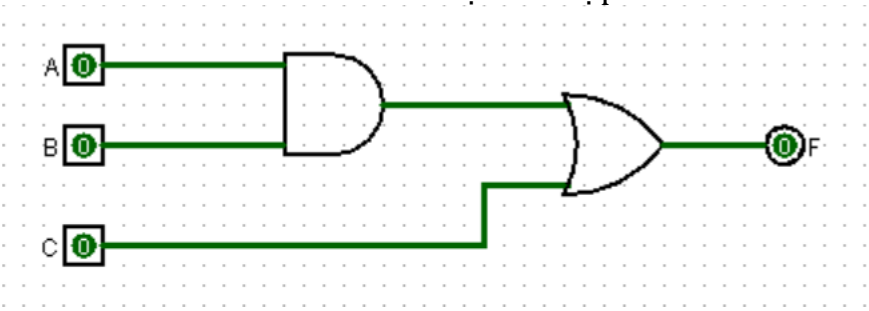
Bảng 1.1.2.3 Bảng giá trị của thanh ghi

CLK	D	Q	Q^+
-	0	0	0
-	0	1	1
-	1	0	0
-	1	1	1
↑	0	0	0
↑	0	1	0
↑	1	0	1
↑	1	1	1

1.2. Báo cáo phần bài tập

1.2.1. Mô phỏng mạch tổ hợp

Hình 1.2.1.1 Mạch tổ hợp

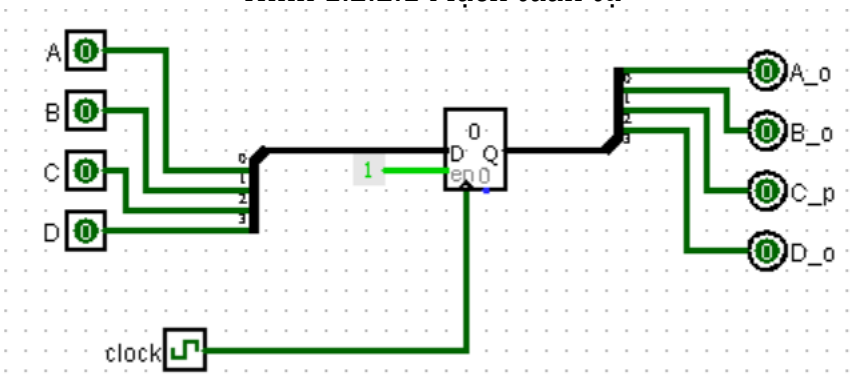


Bảng 1.2.1.1 Bảng chân trị

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

1.2.2. Mô phỏng mạch tuần tự

Hình 1.2.2.1 Mạch tuần tự



Bảng 1.2.2.1 Bảng giá trị

CLK	A_o*	B_o*	C_o*	D_o*
0	A_o	B_o	C_o	D_o
1	A	B	C	D

1.3. Báo cáo phần bài tập bổ sung

1.3.1. Trình bày ngắn gọn chức năng và nguyên lý hoạt động D-Flipflop, Thanh ghi.

Trả lời:

- D-Flipflop là thiết bị điện tử có khả năng lưu trữ một bit nhị phân và có thể thay đổi được nhờ CLK(clock)
- Trong khi ngõ vào quyết định ngõ ra sẽ là gì thì, CLK(clock) sẽ quyết định khi nào có sự thay đổi đó, cụ thể khi xung CLK tác động theo cạnh lên thì ngõ ra sẽ bằng ngõ vào và khi xung tác động theo cạnh xuống thì ngõ ra không thay đổi.

1.3.2. Phân biệt sự khác nhau giữa mạch tổ hợp và mạch tuần tự.

Trả lời:

Mạch tổ hợp là mạch mà đầu ra chỉ phụ thuộc vào đầu vào hiện tại

Mạch tuần tự là mạch mà đầu ra phụ thuộc vào đầu vào hiện tại và đầu vào quá khứ

1.3.3. Clock(xung nhịp) CPU là gì, các trạng thái của clock.

Trả lời:

Clock(xung nhịp) CPU là một mạch tạo xung nhằm tạo mối quan hệ thời gian để cho phép các tác vụ có thể thực hiện một cách tuần tự, trước sau hoặc đồng thời cùng lúc, nó như một "đồng hồ" cho phép máy tính có thể tham chiếu thời gian để hoạt động chính xác

Clock có hai trạng thái là cạnh lên và cạnh xuống.

1.3.4. Mô phỏng mạch bằng logisim

a) $(ABC + AC + AB + BC)C$

Biến đổi

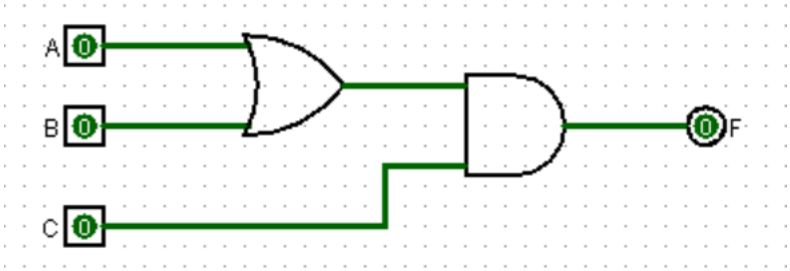
$(ABC + AC + AB + BC)C$

$= ABC + AC + ABC + BC$

$= AC(B + 1) + BC(A + 1)$

$= AC + BC = C(A + B)$

Hình 1.3.4.1 Mạch tổ hợp



Bảng 1.3.4.1 Bảng giá trị mạch tổ hợp

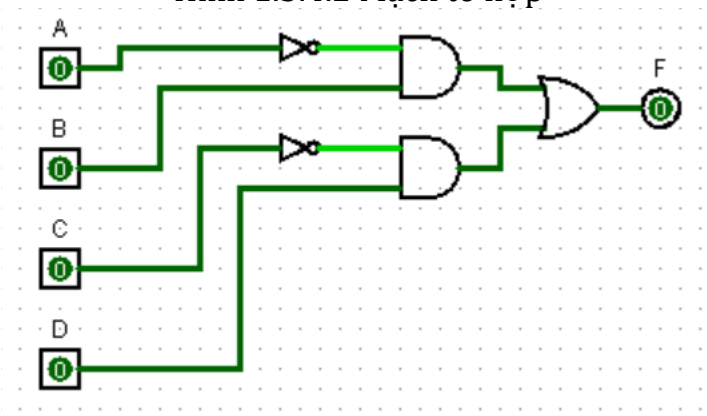
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

b) $(AD + ABC + ABD + ACD)\,\bar{A} + \bar{A}B + \bar{C}D$

Biến đổi

$(AD + ABC + ABD + ACD)\,\bar{A} + \bar{A}B + \bar{C}D = \bar{A}B + \bar{C}D$

Hình 1.3.4.2 Mạch tổ hợp



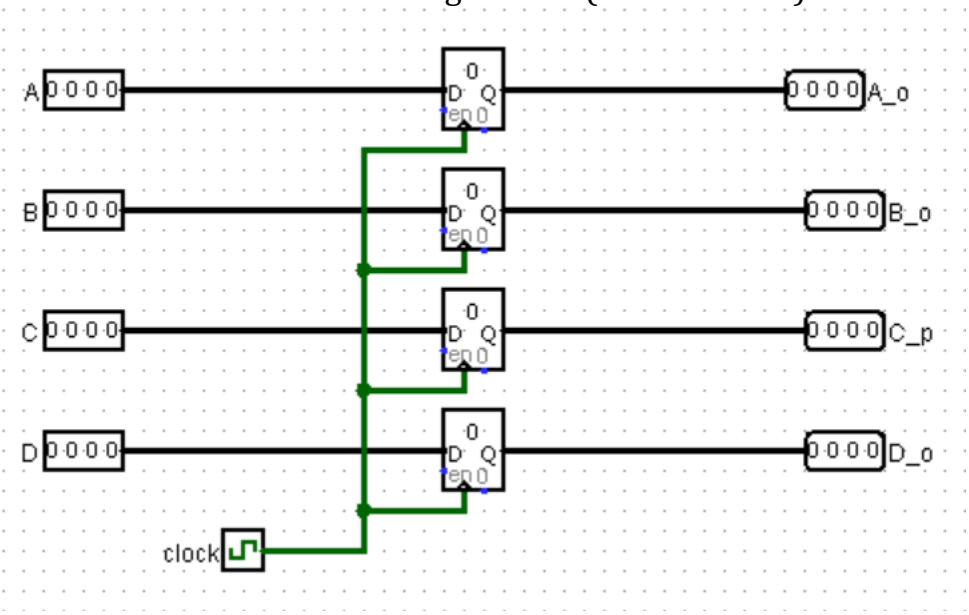
Bảng 1.3.4.2 Bảng giá trị mạch tổ hợp

A	B	C	D	F
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0

1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

1) Thiết kế lại mạch tuần tự ở 1.2.1 với 16bit dữ liệu

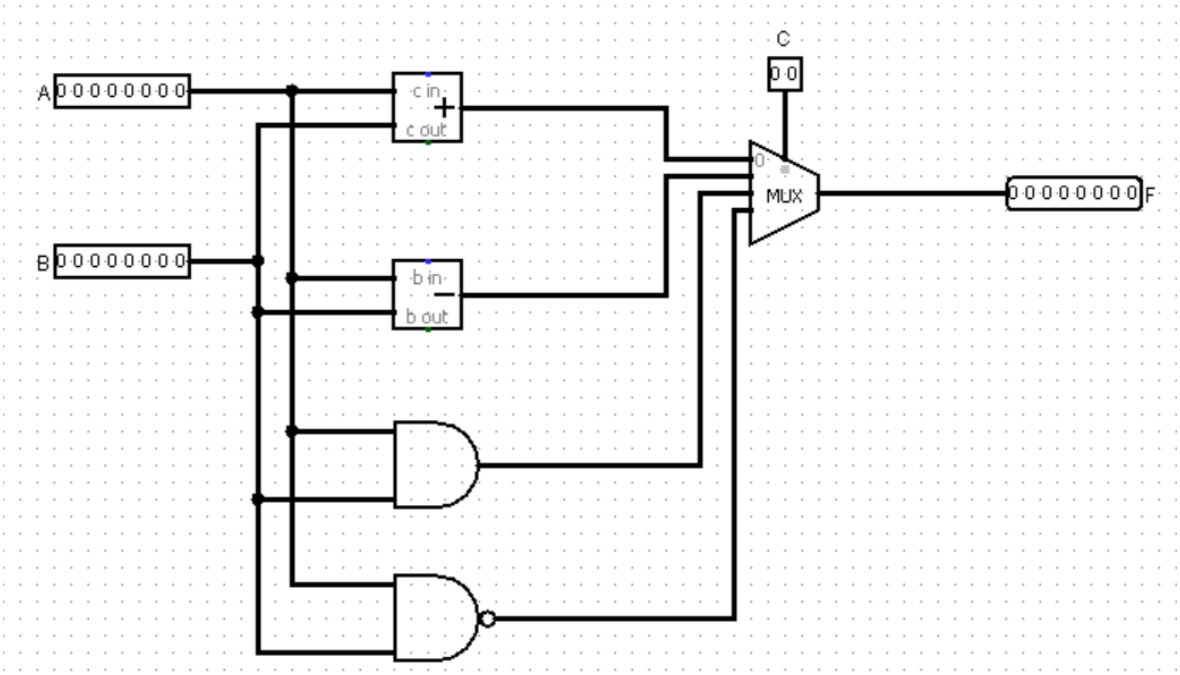
Hình 1.3.4.3 Thanh ghi 16 bit(4 thanh 4 bit)



Chương 2. BÁO CÁO LAB02

2.1. Báo cáo phần thực hành

2.1.1. Mô phỏng ALU

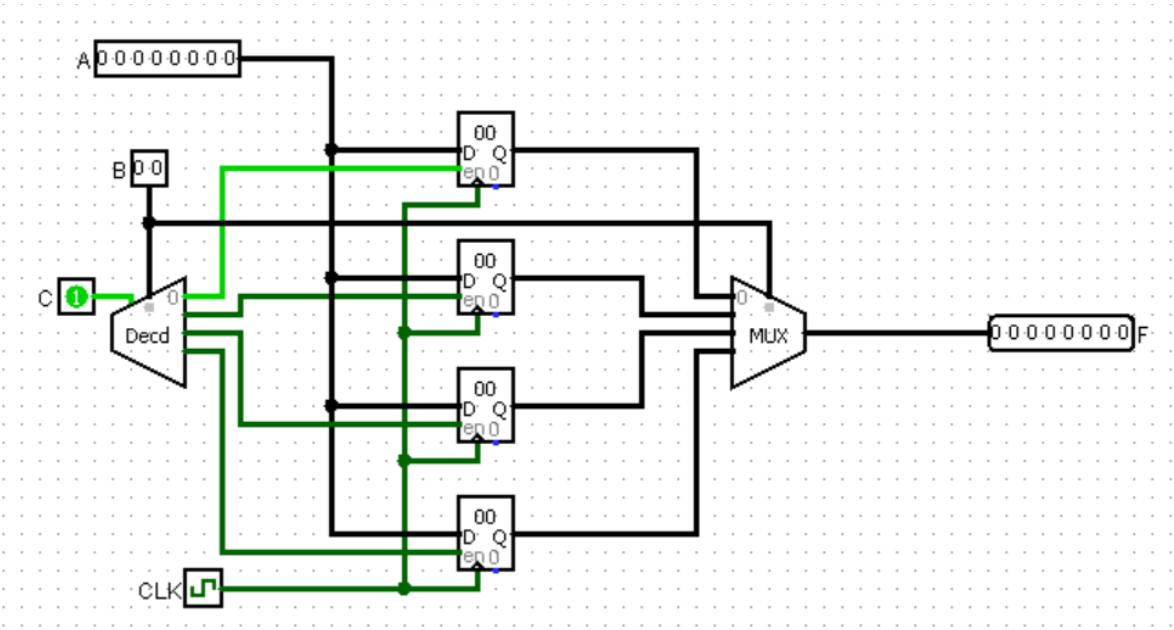


Hình 2.1.1.1 Cấu tạo ALU

Bảng 2.1.1.1 Cấu tạo ALU

C	F
00	A+B
01	A-B
10	A AND B
11	NOT (A AND B)

2.1.2. Mô phỏng Register Files gồm 4 thanh ghi 8 bit



Hình 2.1.2.1 Cấu tạo Register Files

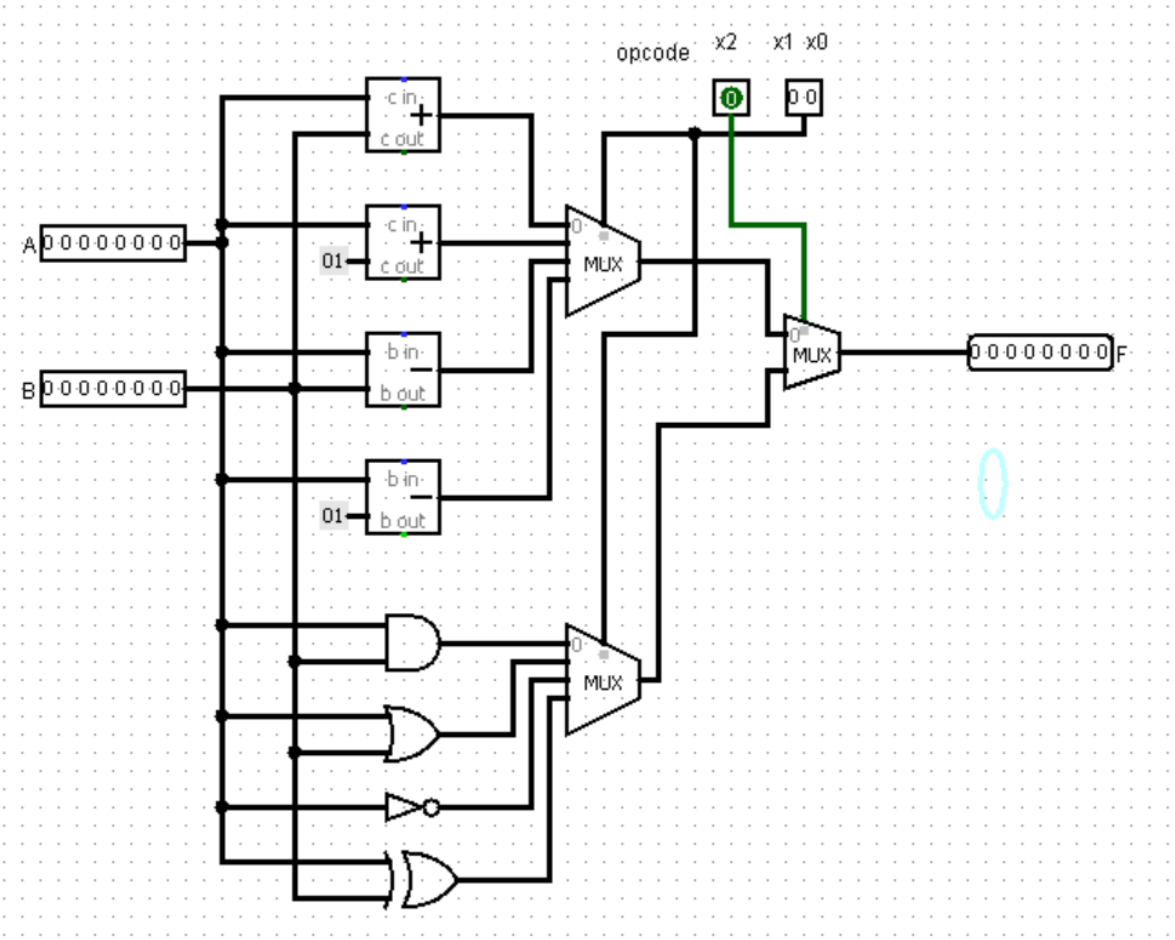
C	CLK	F*
0	0	F*
0	1	F*
1	0	F*
1	1	A

Bảng 2.1.2.1 Bảng chân trị Register Files

2.2. Báo cáo phần bài tập

2.2.1. Cải tiến ALU

Cải tiến ALU với các phép toán: A + B, A + 1, A – B, A – 1, A AND B, A OR B, NOT A, A XOR B



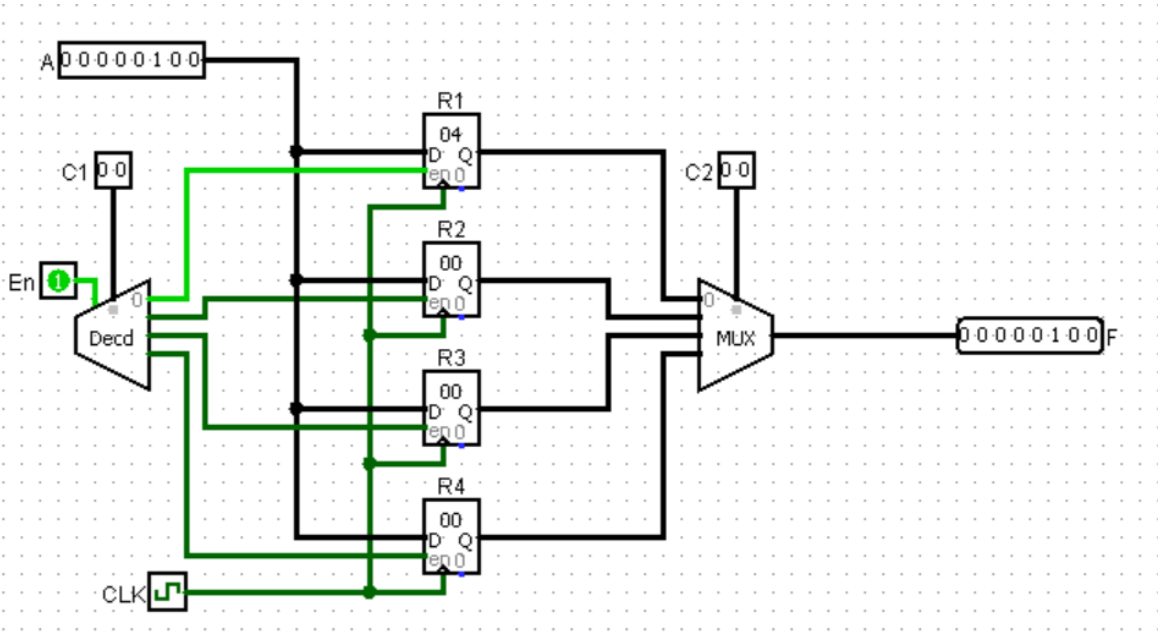
Hình 2.2.1.1 Cấu tạo ALU cải tiến

Opcode			F
0	0	0	A+B
0	0	1	A+1
0	1	0	A-B
0	1	1	A-1
1	0	0	A&B
1	0	1	A|B
1	1	0	A'
1	1	1	A ⊕ B

Bảng 2.2.1.1 Bảng chân trị ALU cải tiến

2.2.2. Thiết kế lại Register Files

Thiết kế và mô phỏng lại Register Files với địa chỉ xuất riêng với địa chỉ ghi



Hình 2.2.2.1 Cấu tạo Register Files(2)

2.3. Báo cáo phần bài tập Bổ sung

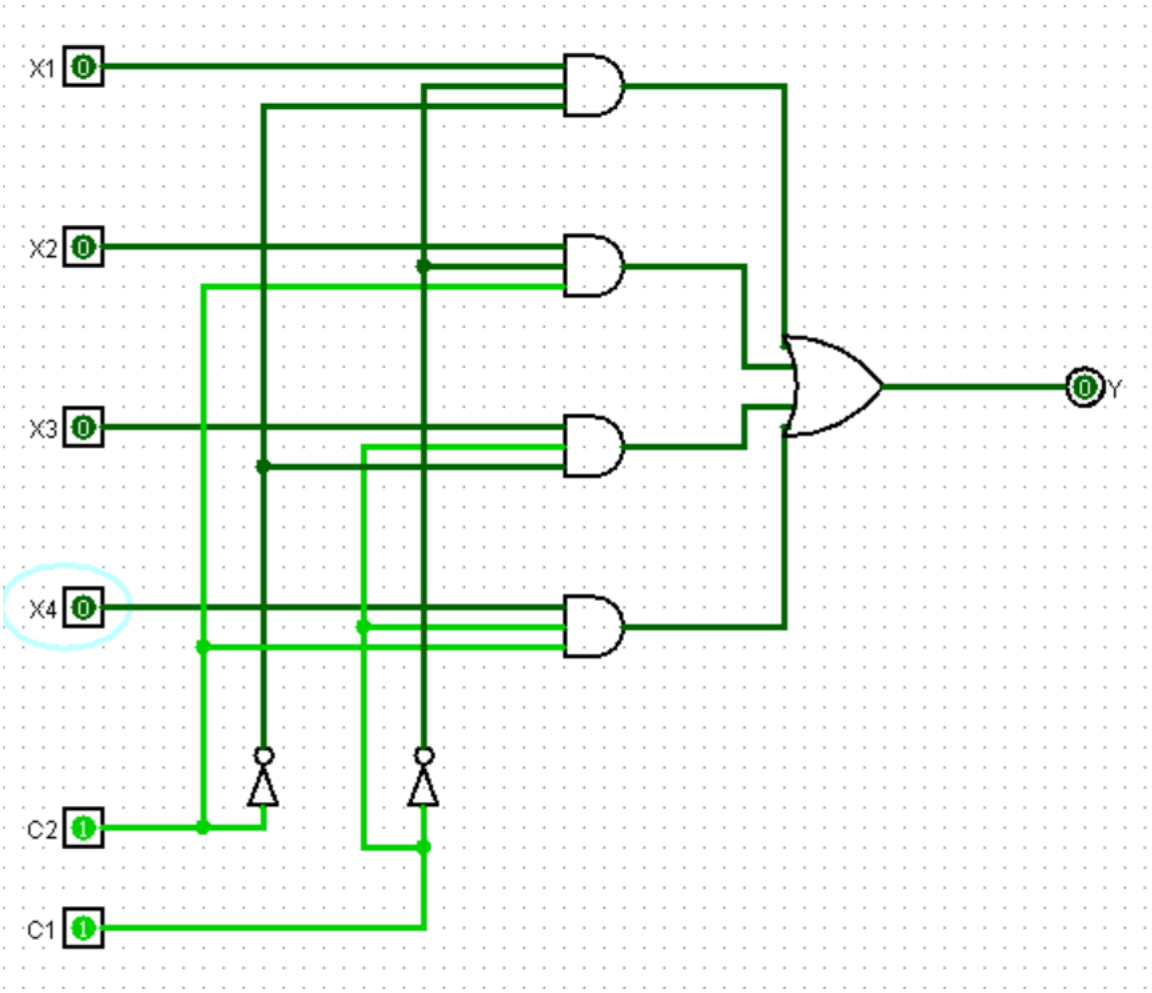
2.3.1. Phân biết Mux và Decode? Thiết kế mux 4to1 và decode 2to4 bằng các cổng luận lý

Trả lời:

Mux là mạch có chức năng chọn lần lượt 1 trong N kênh vào để đưa đến ngõ ra duy nhất.

Decode là mạch biến đổi tín hiệu đầu vào nhị phân “n” thành mã tương đương sử dụng 2^n đầu ra.

Thiết kế Mux4to1:

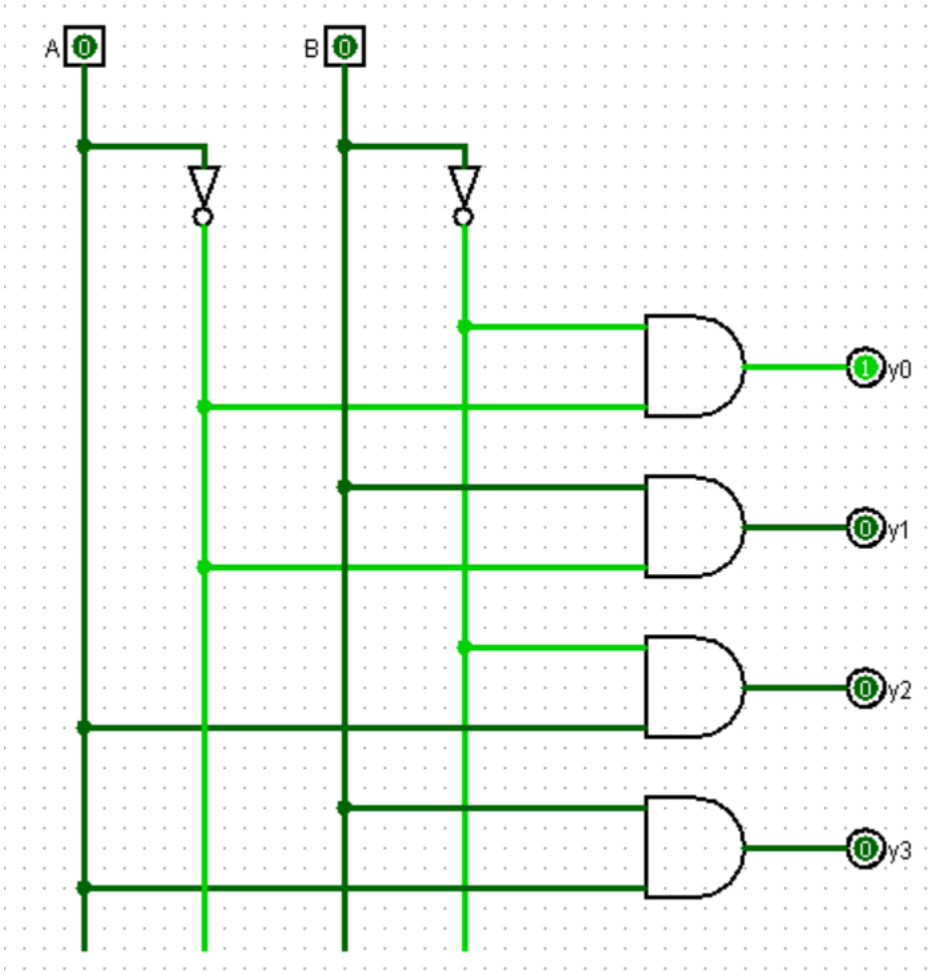


Hình 2.3.1.1 Cấu tạo Mux4to1

C1	C2	F
0	0	X1
0	1	X2
1	0	X3
1	1	X4

Bảng 2.3.1.1 Bảng chân trị Mux4to1

Thiết kế decode2to4:



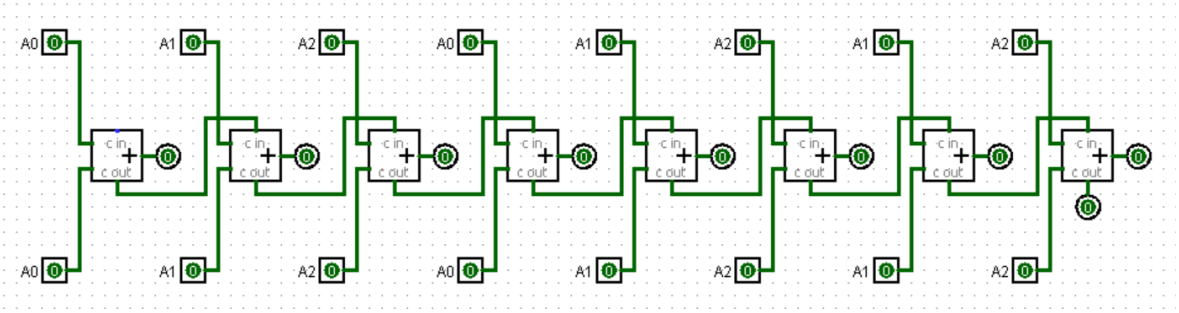
Hình 2.3.1.2 Cấu tạo Decode2to4

B	A	x0	x1	x2	x3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Bảng 2.3.1.2 Bảng chân trị Decode2to4

2.3.2. Thiết kế lại bộ cộng có chức năng cộng 2 số 8 bit.

Trả lời:

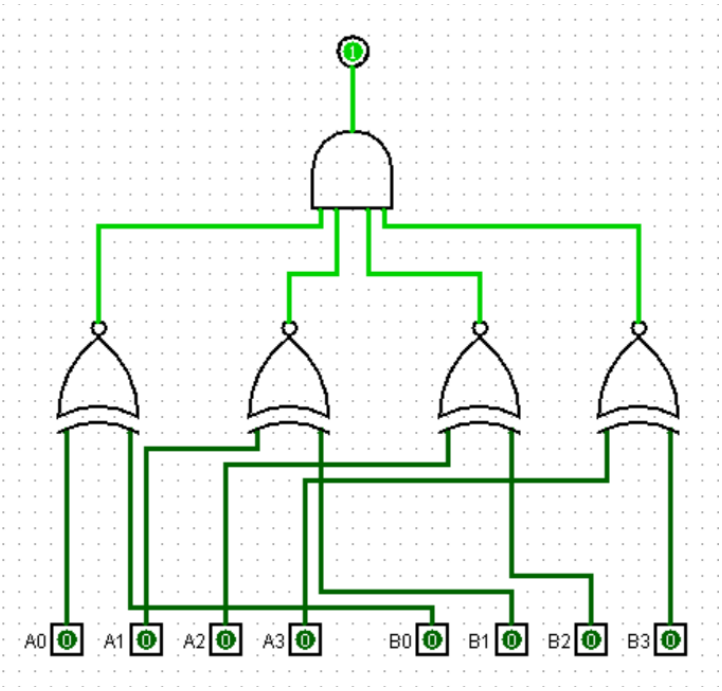


Hình 2.3.2.1 Bộ cộng 8bit

2.3.3. Sinh viên thiết kế mạch có chức năng so sánh hai input 4 bit có bằng nhau hay không

Sinh viên thiết kế mạch có chức năng so sánh hai input 4 bit có bằng nhau hay không? Trường hợp bằng nhau, output bằng 1 ngược lại output bằng 0

Trả lời:



Hình 2.3.3.1 Bộ so sánh 4bit

Chương 3. BÁO CÁO LAB03

3.1. Báo cáo phần thực hành

3.1.1. Mô phỏng và cho biết chức năng của một số lệnh MIPS

Mô phỏng	Chức Năng
add $s1,$s2,$s3	Cộng giá trị hai thanh ghi s2, s3 và lưu kết quả vào s1
addi $s1,$s2,100	Cộng giá trị thanh ghi s2 với một số nguyên và lưu kết quả vào s1
addu $s1,$s2,$s3	Cộng giá trị hai thanh ghi s2, s3 và lưu kết quả vào s1, giá trị của s2,s3 được xem như số nguyên không dấu
addiu $s1,$s2,100	cộng giá trị thanh ghi s2, với một số nguyên và lưu kết quả vào s1, giá trị của s2, số nguyên được xem như số nguyên không dấu
sub $s1,$s2,$s3	trừ giá trị thanh ghi s2 cho s3 và lưu kết quả vào s1
subu $s1,$s2,$s3	trừ giá trị thanh ghi s2 cho s3 và lưu kết quả vào s1, giá trị s2,s3 được xem như số nguyên không dấu
and $s1,$s2,$s3	Lưu kết quá s2 and s3 vào s1
andi $s1,$s2,100	Lưu kết quá s2 and 100 vào s1
or $s1,$s2,$s3	Lưu kết quá s2 or s3 vào s1
nor $s1,$s2,$s3	Lưu kết quá s2 nor s3 vào s1
lw $s1,10($s2)	lưu dữ liệu trong địa chỉ thanh ghi s2 cộng với số nguyên 10 vào thanh ghi s1
sw $s1,10($s2)	lưu dữ liệu trong địa chỉ thanh ghi s1 vào địa chỉ thanh ghi $s2 cộng với số nguyên 10
slt $s1,$s2,$s3	Gán 1 váo s1 nếu s2<s3 và ngược lại gán 0
slti $s1,$s2,100	Gán 1 váo s1 nếu s2<100 và ngược lại gán 0
sltu $s1,$s2,$s3	Gán 1 váo s1 nếu s2<s3 và ngược lại gán 0, so sánh trên là so sánh không dấu
sltiu $s1,$s2,100	Gán 1 váo s1 nếu s2<100 và ngược lại gán 0, so sánh trên là so sánh không dấu
syscall	Lệnh syscall làm treo sự thực thi của chương trình và chuyển quyền điều khiển cho HĐH (được giả lập bởi MARS). Sau đó, HĐH sẽ xem giá trị thanh ghi $v0 để xác định xem chương trình muốn nó làm việc gì.

Bảng 3.1.1.1 Chức năng một số câu lệnh MIPS

3.1.2. Mô phỏng các chương trình và cho biết ý nghĩa

Code	Giải thích

.data
varl: .word 23
.text
__start:
lw $t0,varl
li $t1,5
sw $t1,varl
	#khai báo vùng nhớ data
	#khai báo biến kiểu word: var1=23
	#khai báo vùng nhớ text
	
	#t0 lưu giá trị var1
	#t1=5
	#var1 lưu giá trị t1

Bảng 3.1.2.1 Ví dụ 1

Code	Giải thích

.data
array1: .space 12
.text
__start:
la $t0,array1
li $t1,5
sw $t1,($t0)
li $t1,13
sw $t1,4($t0)
li $t1,-7
sw $t1,8($t0)
	#khai báo vùng nhớ data
	#cấp 12-byte bộ nhớ, chưa được khởi tạo
	#khai báo vùng nhớ text
	
	#t0 = địa chỉ array1
	#t1=5
	#array1[0]=t1
	#t1=13
	#array1[1]=t1
	#t1=-7
	#array1[2]=t1

Bảng 3.1.2.2 Ví dụ 2

Code	Giải thích
li $v0,5 syscall	#truyền tham số 5 vào thanh ghi v0, do v0 là thanh ghi đặc biệt nên có thể hiểu là chọn chức năng mong muốn sẽ được thực hiện khi chạy lệnh syscall (ở đây là đọc số nguyên) # #thực hiện chức năng

Bảng 3.1.2.3 Ví dụ 3

Code	Giải thích
.data string1: .asciiz "Print this.\n" .text main: li $v0,4 la $a0,string1 syscall	#khai báo vùng nhớ data #khai báo mảng string1 = "Print this.\n" #khai báo vùng nhớ text #khai báo label main #truyền tham số 4 vào v0, ở đây là chức năng in chuỗi ký tự mà địa chỉ được lưu trong a0 #a0 = địa chỉ string1 #thực hiện chức năng

Bảng 3.1.2.4 Ví dụ 4

3.2. Báo cáo phần bài tập

3.2.1. Nhập vào một chuỗi, xuất ra cửa sổ I/O theo từng yêu cầu

a) Khai báo và xuất ra cửa sổ I/O 2 chuỗi có giá trị như sau:

- Chuỗi 1: Chao ban! Ban la sinh vien nam thu may?
- Chuỗi 2: Hihi, minh la sinh vien nam thu 1 ^-^

b) Xuất ra lại đúng chuỗi đã nhập

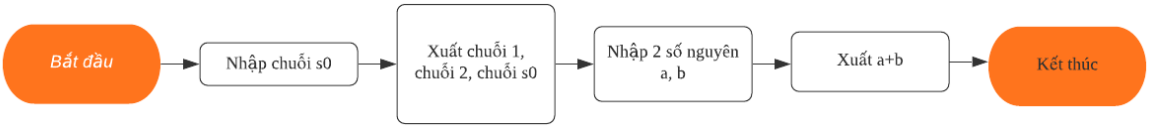
Ví dụ:

Nhap: Truong Dai hoc Cong nghe Thong tin

Xuất: Truong Dai hoc Cong nghe Thong tin

c) Nhập vào 2 số nguyên sau đó xuất tổng của 2 số nguyên này

Hình 3.2.1.1 Lưu đồ thuật toán 3.1



Bảng 3.2.1.1 Chương trình 3.1

Code	Giải thích
.data	#khai báo vùng nhớ data
nhap: .asciiz "Nhap: "	#nhap = "Nhap: "
nhapa: .asciiz "a = "	#nhapa = "a = "
nhapb: .asciiz "b = "	#nhapb = "b = "
tong: .asciiz "Tong a + b = "	#tong = "Tong a + b = "
xuat: .asciiz "Xuat: "	#xuat = "Xuat: "
constr1: .asciiz "Chao ban! Ban la sinh vien nam thu may?"	#constr1 = "Chao ban! Ban la sinh vien nam thu may?"
constr2: .asciiz "Hihi, minh la sinh vien nam thu 1 ^-^"	#constr2 = "Hihi, minh la sinh vien nam thu 1 ^-^"
endl: .asciiz "\n"	#endl = "\n"
buffer : .space 100	#cấp phát 100-byte bộ nhớ
.text	#khai báo vùng nhớ text
li $v0,4	#xuất "Nhap: "
la $a0,nhap	
syscall	
li $v0,8	#nhập vào 1 chuỗi
la $a0,buffer	
li $a1,100	
syscall	
move $s0,$a0	#s0 = chuỗi vừa nhập
li $v0,4	#xuất "Chao ban! Ban la sinh vien nam thu may?"
la $a0,constr1	
syscall	
la $a0,endl	#xuất dấu xuống dòng
syscall	
la $a0,constr2	#xuất "Hihi, minh la sinh vien nam thu 1 ^-^"
syscall	
la $a0,endl	#xuất dấu xuống dòng
syscall	
la $a0,xuat	# xuất "Xuat: "
syscall	

```
move $a0,$s0                    #a0=s0
syscall                         #xuất s0

li $v0,4
la $a0,nhap                     #xuất "Nhap: "
syscall
la $a0,endl                     #xuất dấu xuống dòng
syscall
la $a0,nhapa                    #xuất "a = "
syscall
li $v0,5                        #nhập a
syscall
move $s1,$v0                    #s1=a

li $v0,4
la $a0,nhapb                    #xuất "b = "
syscall
li $v0,5                        #nhập b
syscall
move $s2,$v0                    #s2=b

li $v0,4
la $a0,tong                     #xuất "Tong a + b = "
syscall

li $v0,1
add $a0,$s1,$s2                 #a0=s1+s2
syscall                         #xuất a0
```

3.3. Báo cáo phần bài tập bổ sung

3.3.1. Assembly là gì? Trình bày các quá trình một chương trình viết bằng ngôn ngữ C/C++ được thực hiện trên máy tính?

Trả lời:

Assembly là một loại ngôn ngữ lập trình cấp thấp cho bộ vi xử lý và các thiết bị có thể lập trình khác. Ngôn ngữ này có thể được tạo bằng cách biên dịch mã nguồn từ một ngôn ngữ lập trình cấp cao, chẳng hạn như C, C++.

Sau khi viết một chương trình bằng C/C++, để có thể chạy được chương trình này, chúng ta cần một hành động gọi là compile hay còn gọi là thông dịch, nhằm biên dịch mã nguồn của C thành dạng mã mà máy tính có thể hiểu và thực thi nó trong chương trình.

3.3.2. Trình bày các kiểu dữ liệu trong MIPS32 và kích thước của từng kiểu dữ liệu.

Trả lời:

Các kiểu dữ liệu:

- word: 32bits
- byte: 8bits
- ascii: 4bits
- asciiz: 4bits
- space: 4bits

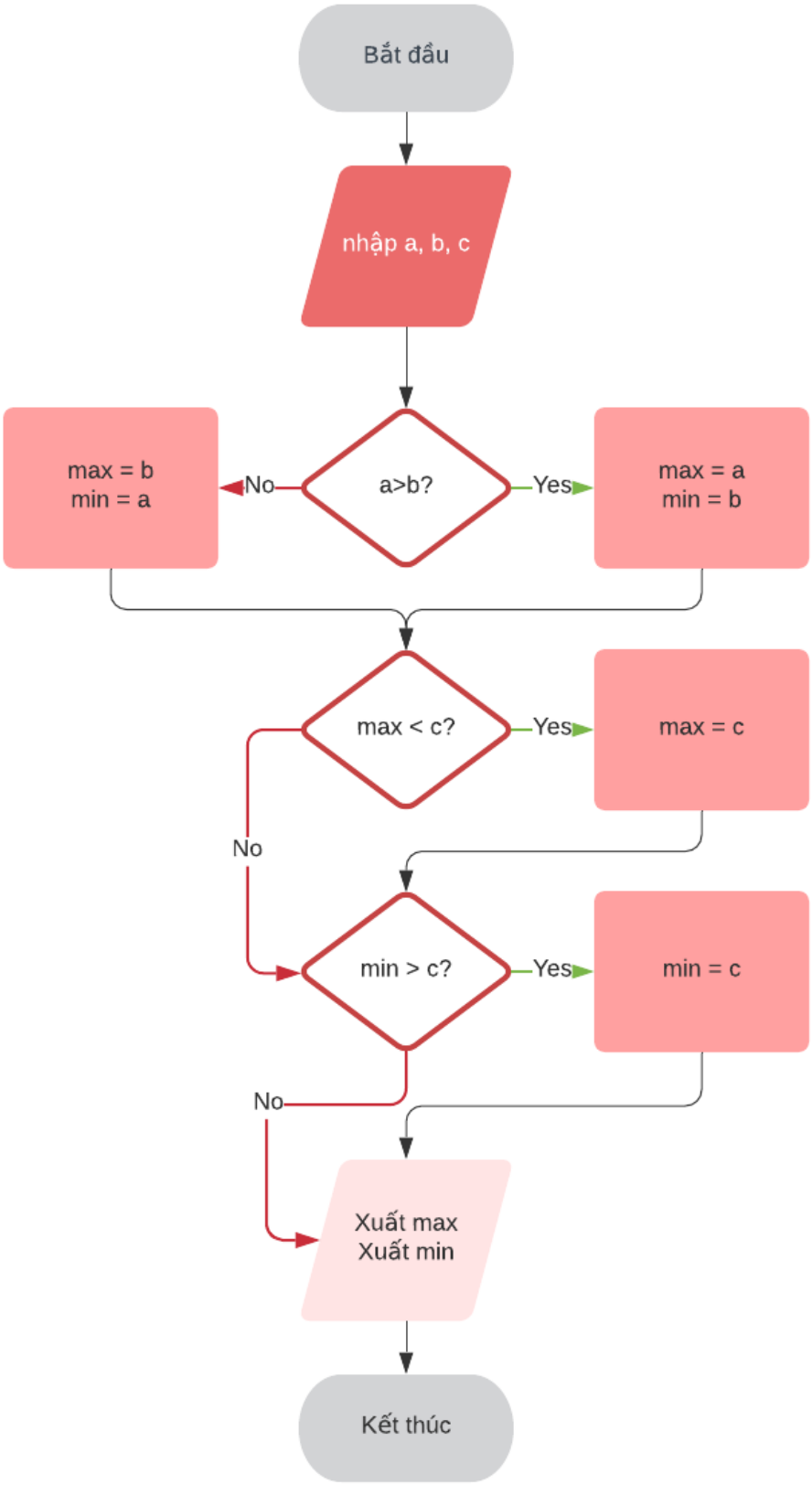
3.3.3. Trình bày cấu trúc bộ nhớ của một chương trình C++(layout memory).

Trả lời:

Memory layout của một chương trình C/C++ gồm 5 phần chính: Text Segment, Initialized Data Segment, Uninitialized Data Segment, Heap và Stack

3.3.4. Viết chương trình hợp ngữ nhập vào ba số a, b, c. Kiểm tra và in ra số lớn nhất, số bé nhất(không dùng vòng lặp)

Trả lời:



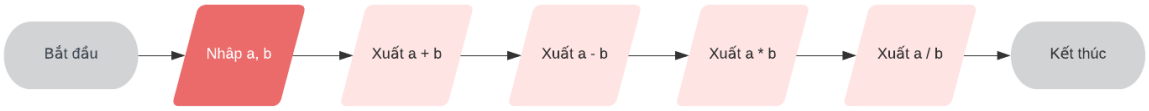
Hình 3.3.4.1 Lưu đồ thuật toán tìm max a,b,c

Bảng 3.3.4.1 Chương trình tìm max a,b,c

Code	Giải thích
.data	#Khai báo vùng nhớ data
input: .asciiz "Input : "	
a: .asciiz "a = "	
b: .asciiz "b = "	
c: .asciiz "c = "	
output: .asciiz "Output : "	
max: .asciiz "Max = "	
min: .asciiz "Min = "	
endl: .asciiz "\n"	
.text	#Khai báo vùng nhớ text
li $v0,4	
la $a0,input	
syscall	
la $a0,a	
syscall	
li $v0,5	
syscall	
move $s0,$v0	#nhập a = s0
li $v0,4	
la $a0,b	
syscall	
li $v0,5	
syscall	
move $s1,$v0	#nhập b = s1
li $v0,4	
la $a0,c	
syscall	
li $v0,5	
syscall	
move $s2,$v0	#nhập c = s2
blt $s0,$s1,else1	#nếu a<b

```
move $s3,$s0                    #max = b, min = a
move $s4,$s1                    #ngược lại
j endif1                        #max = a, min = b
else1:
move $s3,$s1 #s3 = max
move $s4,$s0 #s4 = min
endif1:

blt $s3,$s2,do2                 #nếu max < c
j endif2                        #max = c
do2:
move $s3,$s2
endif2:

bgt $s4,$s2,do3                 #nếu min>c
j endif3                        #min = c
do3:
move $s4,$s2
endif3:

li $v0,4                        #xuất kết quả max, min
la $a0,output
syscall
la $a0,max
syscall

li $v0,1
move $a0,$s3
syscall

li $v0,4
la $a0,endl
syscall
la $a0,min
syscall
li $v0,1
move $a0,$s4
syscall
```

3.3.5. Viết chương trình hợp ngữ nhập vào số nguyên a, b. In ra kết quả của phép cộng, trừ nhân, chia



Hình 3.3.5.1 Lưu đồ thuật toán tính tổng hiệu tích thương

Bảng 3.3.5.1 Chương trình tính tổng hiệu tích thương

Code	Giải thích
.data input: .asciiz "Input : " a: .asciiz "a = " b: .asciiz "b = " cong: .asciiz "a + b = " tru: .asciiz "a - b = " nhan: .asciiz "a * b = " chia: .asciiz "a / b = " output: .asciiz "Output : " endl: .asciiz "\n"	#khai báo vùng nhớ data
.text li $v0,4 la $a0,input syscall la $a0,a syscall li $v0,5 syscall	#khai báo vùng nhớ text
move $s0,$v0	#nhập a, s0 = a
li $v0,4 la $a0,b syscall	

```
li $v0,5
syscall
move $s1,$v0                #nhập b, s1 = b

li $v0,4
la $a0,output
syscall
la $a0,cong
syscall

add $s2,$s0,$s1             #s2 = s0 + s1

li $v0,1                    #xuất s2
move $a0,$s2
syscall

li $v0,4
la $a0,endl
syscall
la $a0,tru
syscall

sub $s2,$s0,$s1             #s2 = s0 - s1

li $v0,1
move $a0,$s2
syscall                     #xuất s2

li $v0,4
la $a0,endl
syscall
la $a0,nhan
syscall

mul $s2,$s0,$s1             #s2 = s0 * s1

li $v0,1
move $a0,$s2
syscall                     #xuất s2
```

li $v0,4 la $a0,endl syscall la $a0,chia syscall	
div $s2,$s0,$s1	#s2 = s0 / s1
li $v0,1 move $a0,$s2 syscall	#xuất s2

Chương 4. BÁO CÁO LAB04

4.1. Báo cáo phần thực hành

4.1.1. Chuyển đoạn code theo sau sang MIPS và sử dụng MARS để kiểm tra lại kết quả:

if (i == j)

f = g + h;

else

f = g – h;

(Với giá trị của i, j, f, g, h lần lượt chứa trong các thanh ghi $s0, $s1, $s2, $t0, $t1)

Thực hành:

Bảng 4.1.1.1 Chuyển IF-ELSE sang MIPS

Code	Giải thích

.text
beq $s0,$s1,do
sub $s2,$t0,$t1
j endif
do:
add $s2,$t0,$t1
endif:
	#khai báo vùng nhớ text
	#if (i==j) thực hiện label do
	#ngược lại, j = g - h
	#kết thúc if
	#label do
	#j = g + h
	#kết thúc if

int Sum = 0

for (int i = 1; i <=N; ++i){

Sum = Sum + i;

}

(Với giá trị của i, N, Sum lần lượt chứa trong các thanh ghi $s0, $s1, $s2)

Thực hành:

Bảng 4.1.1.2 Chuyển vòng lặp for sang MIPS

Code	Giải thích

.text
li $s2,0
li $s0,1
loop:
bgt $s0,$s1,endloop
add $s2,$s2,$s0
addi $s0,$s0,1
j loop
endloop:
	#khai báo vùng nhớ text
	#sum = 0
	#i = 1
	
	#nếu I > N, nhảy xuống label endloop
	#sum = sum + i
	#i = i +1
	#nhảy lên label loop, mô phỏng vòng for
	#label endloop, kết thúc vòng for

4.2. Báo cáo phần bài tập

4.2.1. Nhập vào một ký tự, xuất ra cửa sổ I/O của MARS theo từng yêu cầu sau

✓ Ký tự liền trước và liền sau của ký tự nhập vào

Ví dụ:

Nhap ky tu (chỉ một ký tự): b

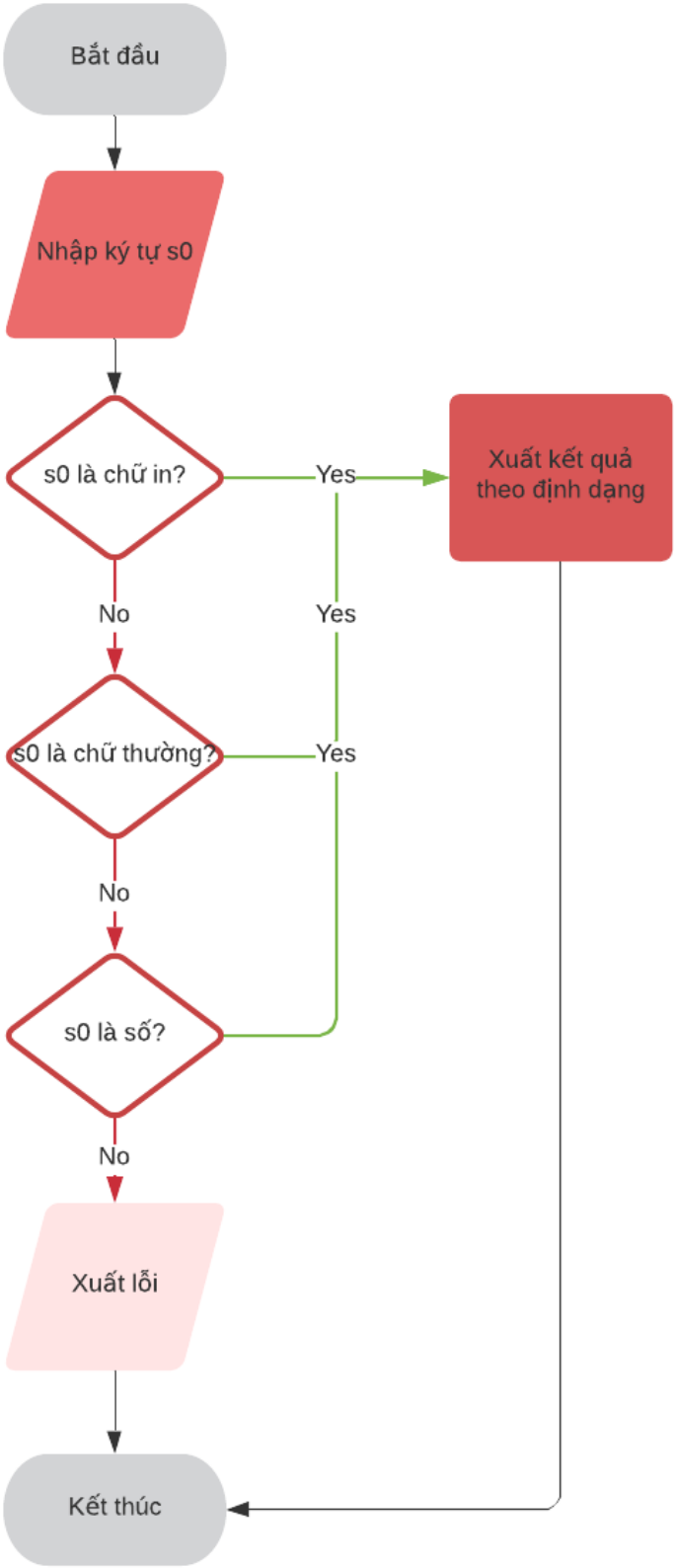
Ky tu truoc: a

Ky tu sau: c

✓ Ký tự nhập vào chỉ được phép là ba loại: số, chữ thường và chữ hoa. Nếu ký tự nhập vào rơi vào một trong ba loại, xuất ra cửa sổ đó là loại nào; nếu ký tự nhập không rơi vào một trong ba loại trên, xuất ra thông báo “invalid type”

Trả lời:

Hình 4.2.1.1 Lưu đồ thuật toán chương trình
kiểm tra ký tự



Bảng 4.2.1.1 Chương trình kiểm tra và xuất ký tự

Code	Giải thích
.data nofront: .asciiz "Khong ton tai ky tu truoc\n" noback: .asciiz "Khong ton tai ky tu sau\n" nhap: .asciiz "Nhap ky tu: " truoc: .asciiz "Ky tu truoc: " sau: .asciiz "Ky tu sau: " loi: .asciiz "invalid type" endl: .asciiz "\n" in: .asciiz "Ky tu la chu in" thuong: .asciiz "Ky tu la chu thuong" so: .asciiz "Ky tu la so"	#khai báo vùng nhớ data
.text li $v0,4 la $a0,nhap syscall li $v0,12 syscall move $s0,$v0 li $v0,4 la $a0,endl syscall jal check_upper jal check_lower jal check_number jal invalid j end_proc check_upper: blt $s0,'A',end_func bgt $s0,'Z',end_func	#khai báo vùng nhớ text #xuất "Nhap ky tu: " #nhập ký tự #gọi hàm kiểm tra ký tự in #gọi hàm kiểm tra ký tự thường #gọi hàm kiểm tra số #nếu ký tự không thuộc một trong 3 trường hợp trên hàm invalid sẽ xuất thông báo lỗi #kết thúc chương trình #phần khai báo chương trình con #hàm kiểm tra ký tự in

```
li $t2,'A'
jal front                  #hàm kiểm tra có tồn tại ký tự trước hay không

li $t2,'Z'
jal back                   #hàm kiểm tra có tồn tại ký tự sau hay không


li $v0,4
la $a0,in                  #xuất ký tự là chữ in
syscall
j end_proc                 #kết thúc chương trình

check_lower:               #hàm kiểm tra ký tự thường
blt $s0,'a',end_func
bgt $s0,'z',end_func

li $t2,'a'
jal front                  #hàm kiểm tra có tồn tại ký tự trước hay không

li $t2,'z'
jal back                   #hàm kiểm tra có tồn tại ký tự sau hay không


li $v0,4
la $a0,thuong              #xuất ký tự là chữ thường
syscall
j end_proc                 #kết thúc chương trình

check_number:              #hàm kiểm tra ký tự thường
blt $s0,'0',end_func
bgt $s0,'9',end_func

li $t2,'0'
jal front                  #hàm kiểm tra có tồn tại ký tự trước hay không

li $t2,'9'
jal back                   #hàm kiểm tra có tồn tại ký tự sau hay không
```

```
li $v0,4
la $a0,so                    #xuất ký tự là số
syscall
j end_proc                   #kết thúc chương trình

invalid:                     #hàm xuất lỗi khi ký tự không hợp lệ

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

front:                       #hàm kiểm tra ký tự trước và xuất
addi $a0,$s0,-1
blt $a0,$t2,do

li $v0,4
la $a0,truoc
syscall

li $v0,11
addi $a0,$s0,-1
syscall

li $v0,4
la $a0,endl
syscall

jr $ra
do:
li $v0,4
la $a0,nofront
syscall
jr $ra

back:                        #hàm kiểm tra ký tự sau và xuất
addi $a0,$s0,1
bgt $a0,$t2,do2

li $v0,4
```

```
la $a0,sau
syscall

li $v0,11
addi $a0,$s0,1
syscall

li $v0,4
la $a0,endl
syscall

jr $ra
do2:
li $v0,4
la $a0,noback
syscall
jr $ra

end_func:
jr $ra

end_proc:
```

4.3. Báo cáo phần bài tập bổ sung

4.3.1. Con trỏ là gì? Chức năng của con trỏ? Mảng là gì? Chức năng của mảng.

Trả lời:

Con trỏ là một vùng nhớ đặc biệt lưu địa chỉ của một vùng nhớ khác, chức năng của con trỏ như tên gọi là để "trỏ" vào một vùng nhớ khác hoặc bản thân nó.

Mảng là một danh sách các phần tử có cùng kiểu dữ liệu, chức năng của mảng là để lưu trữ, truy xuất các giá trị mà lập trình viên cần sử dụng.

4.3.2. Thủ tục là gì? Trình bày luồng hoạt động của một thủ tục trong MIPS

Trả lời:

Thủ tục là một đoạn code có cấu trúc gần giống với một chương trình "con" được người lập trình viết bên cạnh chương trình chính để thực hiện một(hoặc nhiều) công việc khi được gọi bên trong chương trình chính.

Trong MIPS chương trình hoạt động từng dòng từ trên xuống dưới, để thủ tục có thể được thực hiện ở bất kì đâu trong chương trình chính, ta dùng 2 câu lệnh jal và jr. Khi thực hiện câu lệnh jal <thủ tục> ta sẽ nhảy xuống nơi khai báo thủ tục để thực hiện những câu lệnh và lưu vị trí sau câu lệnh jal vào thanh ghi $ra. Ở cuối thủ tục, ta dùng lệnh jr $ra để nhảy vào vị trí mà thanh ghi $ra đang lưu hay nói cách khác là tiếp tục chương trình từ vị trí sau câu lệnh jal <thủ tục> mà ta đã gọi.

4.3.3. Viết chương trình hợp ngữ nhập vào số nguyên a, b. In ra kết quả của phép cộng, trừ nhân, chia. Theo cấu trúc như bên dưới

Input : a = {nhập a}

b = {nhập b}

Output a + b = {Kết quả phép cộng}

a – b = {Kết quả phép trừ}

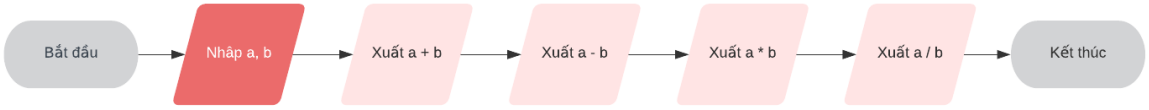
.

.

Program is finished …

Lưu ý: kết quả phải đúng với những phép tính có giá trị lớn

*Vd: 12345678*10000 = 123456780000*



Hình 4.3.3.1 Lưu đồ chương trình tính tổng hiệu tích thương

Bảng 4.3.3.1 Chương trình tính tổng hiệu tích thương 2 số lớn

Code	Giải thích
.data input: .asciiz "Input : " a: .asciiz "a = " b: .asciiz "b = " cong: .asciiz "a + b = " tru: .asciiz "a - b = " nhan: .asciiz "a * b = " chia: .asciiz "a / b = " output: .asciiz "Output : " endl: .asciiz "\n"	#khai báo vùng nhớ data
.text li $v0,4 la $a0,input syscall	#khai báo vùng nhớ text #xuất "Input : "
la $a0,a syscall	#xuất "a = "
li $v0,7 syscall mov.d $f2,$f0	#nhập a
li $v0,4 la $a0,b syscall	#xuất "b = "
li $v0,7 syscall mov.d $f4,$f0	#nhập b

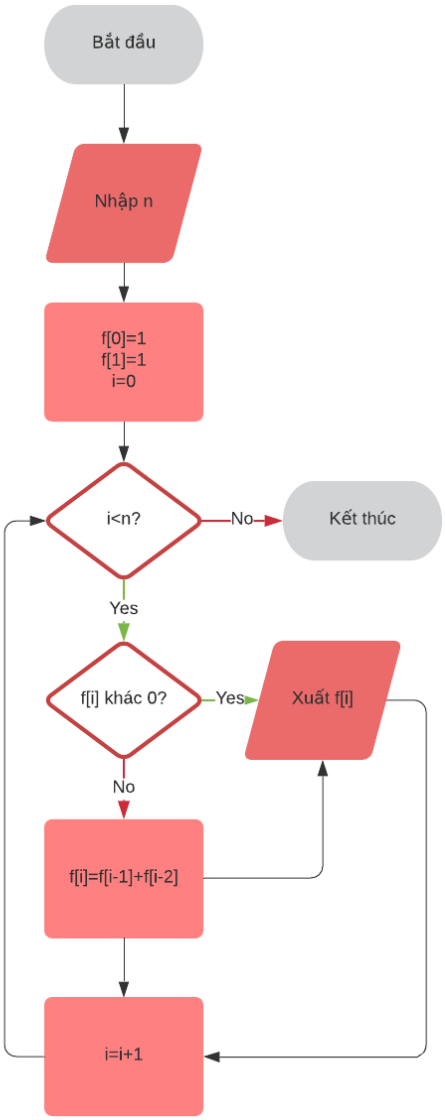
```
li $v0,4                    #xuất kết quả phép cộng
la $a0,output
syscall
la $a0,cong
syscall

li $v0,3
add.d $f12,$f2,$f4
syscall

li $v0,4                    #xuất kết quả phép trừ
la $a0,endl
syscall
la $a0,tru
syscall

sub.d $f12,$f2,$f4
li $v0,3
syscall

li $v0,4                    #xuất kết quả phép nhân
la $a0,endl
syscall
la $a0,nhan
syscall

mul.d $f12,$f2,$f4
li $v0,3
syscall

li $v0,4                    #xuất kết quả phép chia
la $a0,endl
syscall
la $a0,chia
syscall

div.d $f12,$f2,$f4
li $v0,3
```

syscall	

4.3.4. Viết chương trình in ra N(N>2) số fibonaci đầu tiên

Mẫu chương trình:

```
 Input : N = {nhập N}

Output:  {N số fibonaci đầu tiên}
#Vd: N = 5
#      1 1 2 3 5

Program is finished ...
```



Hình 4.3.4.1 Lưu đồ thuật toán xuất n số fibonaci đầu tiên

Bảng 4.3.4.1 Chương trình xuất n số fibonaci đầu tiên

Code	Giải thích
.data	#khai báo vùng nhớ data
input: .asciiz "Input : N = "	
space: .asciiz " "	
output: .asciiz "Output: "	
f: .word 0	#khai báo mảng f
.text	#khai báo vùng nhớ text
li $v0,4	
la $a0,input	
syscall	
li $v0,5	#nhập n
syscall	
move $t0,$v0	
li $t4,1	
sw $t4,f+0	#f[0]=1
sw $t4,f+4	#f[1]=1
li $t1,1	
la $s0,f	#lưu địa chỉ mảng f vào s0
loop:	
bgt $t1,$t0,endloop	#vòng lặp chạy n lần
lw $t5,($s0)	#lấy giá trị f[i]
bne $t5,0,continue	#nếu f[i]!=0 thực hiện, thực hiện label continue
lw $t6,-4($s0)	#lấy giá trị f[i-1]
lw $t7,-8($s0)	#lấy giá trị f[i-2]
add $t5,$t6,$t7	#lưu f[i-1]+f[i-2] vào $t5
sw $t5,($s0)	#lưu $t5 vào f[i]
continue:	#label continue
li $v0,1	#xuất f[i]
lw $a0,($s0)	
Syscall	

li $v0,4 la $a0,space syscall	#xuất dấu cách

addi $t1,$t1,1
addi $s0,$s0,4
 	
j loop
endloop:
	#tăng biến đếm của vòng lặp
	#tăng địa chỉ f của s0 để thao tác trên f kế tiếp

Chương 5. BÁO CÁO LAB05

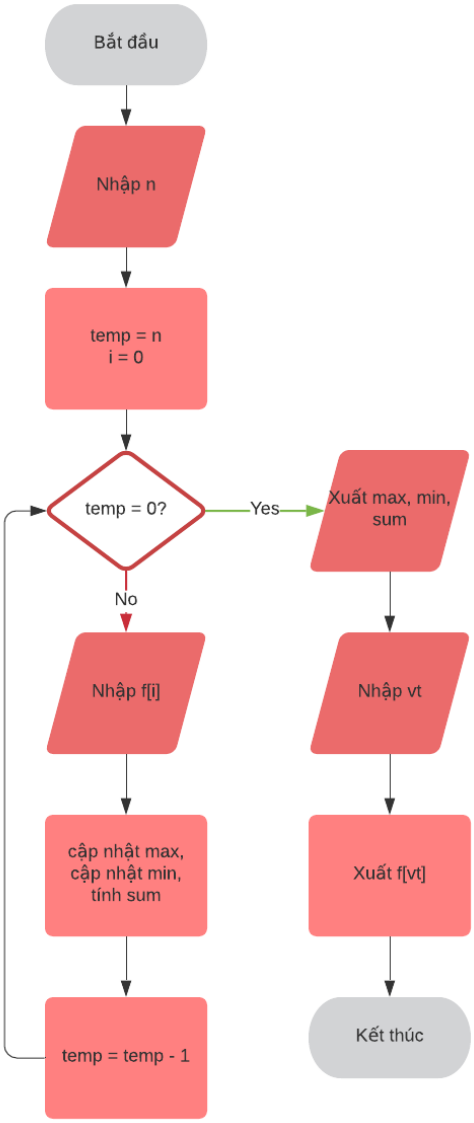
5.1. Báo cáo phần bài tập

5.1.1. Nhập một mảng các số nguyên n phần tử xuất ra cửa sổ I/O của MARS theo từng yêu cầu sau

✓ Xuất ra giá trị lớn nhất và nhỏ nhất của mảng

✓ Tổng tất cả các phần tử của mảng

✓ Người sử dụng nhập vào chỉ số của một phần tử nào đó và giá trị của phần tử đó được in ra cửa sổ



Hình 5.1.1.1 Lưu đồ chương trình thực hành LAB05

Bảng 5.1.1.1 Chương trình thực hành LAB05

Code	Giải thích
.data	
xuongdong : .asciiz " \n"	
max: .asciiz "Max = "	
min: .asciiz "Min = "	
sum: .asciiz "Sum = "	
vt: .asciiz "F["	
vt2: .asciiz "] = "	
mangso: .word 100	
.text	
li $v0,5	
syscall	
move $s0,$v0	#s0 lưu số n
move $s1,$s0	#s1 lưu số n
la $t0,mangso	#t0 lưu địa chỉ mangso
addi $s3,$s3,9999999	#dat min(s3) = so rat lon
do1:	#vòng lặp để nhập n giá trị vào mảng
beq $s1,0,ndo1	
li $v0,5	#nhập phần từ thứ i của mảng
syscall	
sw $v0,($t0)	#lưu i vào mảng
addi $t0,$t0,4	
addi $s1,$s1,-1	
add $s4,$s4,$v0	#s4 lưu tổng giá trị phần tử của mảng
blt $v0,$s2,skip1	#cập nhật max
move $s2,$v0	#s2 lưu max
skip1:	
bgt $v0,$s3,skip2	#cập nhật min
move $s3,$v0	#s3 lưu min
skip2:	
j do1	

```
ndo1:
li $v0,4                    #xuất max
la $a0,max
syscall

li $v0,1
move $a0,$s2
syscall

li $v0,4
la $a0,xuongdong
syscall

li $v0,4                    #xuất min
la $a0,min
syscall

li $v0,1
move $a0,$s3
syscall

li $v0,4
la $a0,xuongdong
syscall

li $v0,4                    #xuất tổng các phần tử
la $a0,sum
syscall

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

li $v0,4
la $a0,xuongdong
syscall

li $v0,5
syscall
```

move $s5,$v0	#s5 lưu giá trị của phần tử cần xuất
li $v0,4 la $a0,vt syscall	#xuất thông báo xuất
li $v0,1 move $a0,$s5 syscall	
li $v0,4 la $a0,vt2 syscall	

mul $s5,$s5,4
la $t0,mangso
add $t0,$t0,$s5
	#s5 = s5 * 4
	#t0 lưu địa chỉ nền của mảng
	#t0 = t0 + s5

li $v0,1 lw $a0,($t0) syscall	#xuất giá trị tại địa chỉ t0

5.1.2. Chuyển đổi code

Chuyển dòng lệnh C dưới đây sang mã assembly của MIPS. Với các biến nguyên i, j được gán lần lượt vào thanh ghi $s0, $s1; và địa chỉ nền của mảng số nguyên A được lưu trong thanh ghi $s3

if (i<j) A[i]= i;

else A[i] = j;

Trả lời

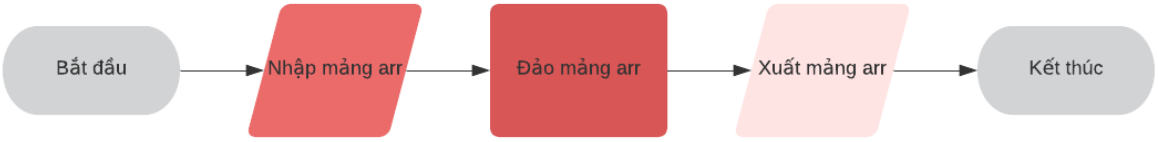
Bảng 5.1.2.1 Chuyển đổi code

Code	Giải thích

blt $s0,$s1,do
mul $s4,$s0,4
add $s3,$s3,$s4
sw $s1,($s3)
j endif
do:
mul $s4,$s0,4
add $s3,$s3,$s4
sw $s0,($s3)
endif:
	#nếu s0<s1, nhảy xuống label do
	#tính địa chỉ A[i]
	
	#A[i] = j
	
	#label do
	#tính địa chỉ A[i]
	
	#A[i] = i

5.2. Báo cáo phần bài tập bổ sung

5.2.1. Viết chương trình hợp ngữ nhập vào N và mang gồm N phần tử. In ra mảng đảo ngược của mảng vừa nhập



Hình 5.2.1.1 Lưu đồ thuật toán chương trình đảo xâu

Bảng 5.2.1.1 Chương trình đảo xâu

Code	Giải thích
.data input: .asciiz "Input : N = " input2: .asciiz "Arr = " space: .asciiz " " arr: .word 0 .text li $v0,4 la $a0,input syscall li $v0,5	#khai báo vùng nhớ data #khai báo vùng nhớ text #nhập n

```
syscall
move $s5,$v0                    #s5 = n
move $s6,$v0                    #s6 = n

li $v0,4                        #xuất thông báo nhập mảng
la $a0,input2
syscall

la $s0,arr

                                #vòng lặp nhập từng phần tử trong
loop:                           mảng
beq $s6,0,endloop

li $v0,5
syscall

sw $v0,($s0)
addi $s0,$s0,4
addi $s6,$s6,-1
j loop
endloop:
addi $s1,$s0,-4
la $s0,arr
loop2:                          #vòng lặp để đảo mảng
bge $s0,$s1,endloop2
lw $t0,($s0)
lw $t1,($s1)
sw $t0,($s1)
sw $t1,($s0)
addi $s0,$s0,4
addi $s1,$s1,-4
j loop2
endloop2:
move $s6,$s5
la $s0,arr

                                #vòng lặp để xuất các phần tử trong
loop3:                          mảng
beq $s6,0,endloop3
li $v0,1
lw $a0,($s0)
```

syscall
li $v0,4
la $a0,space
syscall
addi $s0,$s0,4
addi $s6,$s6,-1
j loop3
endloop3:	

TÀI LIỆU THAM KHẢO

1. https://www.youtube.com/watch?v=RxjfB2fp2lk&t=1875s&ab_channel=ThienBuiVan
2. https://www.youtube.com/user/amellperalta
3. http://www.cit.ctu.edu.vn/~dtnghi/cod/ch3.pdf
4. https://vietcodes.github.io/algo/mips
5. https://www.dsi.unive.it/~gasparetto/materials/MIPS_Instruction_Set.pdf
6. https://buivanluongueh.files.wordpress.com/2011/09/vanluong-blogspot-com_mips.pdf
7. https://www.alpharithms.com/mips-store-word-sw-vs-load-word-lw-475521/
8. https://drive.google.com/file/d/1D_HJ2EMZdjikkuQwB2brtm2qwHEHyC0U/edit
9. Stack overflow community