BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC BÁCH KHOA HÀ NỘI

Trường Công nghệ thông tin và truyền thông

----- o0o -----

BÁO CÁO BÀI TẬP CUỐI KỲ 20212

Môn Học: Thực hành Kiến trúc Máy Tính

Mã học phần: IT3280

Sinh viên thực hiện:

Nguyễn Thị Hạnh MSSV: 20194552

Hoàng Đức Thiện MSSV: 20194681

Giảng viên hướng dẫn: Thầy Lê Bá Vui

MỤC LỤC

LỜI CẢM ƠN 3
Bài 8: Mô phỏng ổ đĩa RAID 5 4
1. Đề bài 4
2. Phân tích cách thực hiện 4
2.1. Phân tích đề bài 4
2.2. Cách thực hiện 4
3. Mã nguồn 5
4. Chú thích 18
4.1. Ý nghĩa thanh ghi 18
4.2. Ý nghĩa các hàm 18
5. Kết quả 19
Bài 10: Máy tính bỏ túi 20
1. Đề bài: 20
2. Phân tích cách thực hiện: 20
2.1. Phân tích đề bài: 20
2.2. Ý tưởng thực hiện: 20
2.3. Xử lí các ngoại lệ: 21
2.4. Các bước thực hiện: 21
3. Ý nghĩa của các thanh ghi được sử dụng: 21
4. Mã nguồn 22
5. Kết quả 38

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin được bày tỏ lòng biết ơn chân thành và sâu sắc nhất tới thầy giáo Lê Bá Vui – người luôn tận tình, nhiệt huyết giúp đỡ, hướng dẫn chúng em trong thời gian học môn Thực hành kiến trúc máy tính, đã truyền đạt cho chúng em những kiến thức bổ ích giúp chúng em có đủ kiến thức để hoàn thành tốt môn học cũng như bài tập lần này.

Bài 8: Mô phỏng ổ đĩa RAID 5

Sinh viên thực hiện: Hoàng Đức Thiện - 20194681

1. Đề bài

Hệ thống ổ đĩa RAID5 cần tối thiểu 3 ổ đĩa cứng, trong đó phần dữ liệu parity sẽ được chứa lần lượt lên 3 ổ đĩa như trong hình bên. Hãy viết chương trình mô phỏng hoạt động của RAID 5 với 3 ổ đĩa, với giả định rằng, mỗi block dữ liệu có 4 kí tự. Giao diện như trong minh họa dưới. Giới hạn chuỗi kí tự nhập vào có độ dài là bội của 8. Trong ví dụ sau, chuỗi kí tự nhập vào từ bàn phím (DCE.****ABCD1234HUSTHUST) sẽ được chia thành các block 4 byte. Block 4 byte đầu tiên "DCE." sẽ được lưu trên Disk 1, Block 4 byte tiếp theo "****" sẽ lưu trên Disk 2, dữ liệu trên Disk 3 sẽ là 4 byte parity được tính từ 2 block đầu tiên với mã ASCII là 6e='D' xor '*'; 69='C' xor '*'; 6f='E' xor '*'; 04='.' xor '*'

2. Phân tích cách thực hiện

2.1. Phân tích đề bài

- Nhập vào chuỗi ký tự có độ dài là bội của 8 rồi tiến hành lưu dữ liệu vào 3 disk theo hệ thống RAID5.
- Block 4 bytes đầu sẽ lưu vào disk 1
- Block 4 bytes tiếp theo sẽ lưu vào disk 2
- Dữ liệu lưu vào disk 3 là 4 bytes parity được tính từ 2 block đầu tiên theo toán tử XOR.
- 8 bytes tiếp theo lưu lần lượt vào disk 1 và disk 3, disk 2 lại được tính theo toán tử XOR từ 2 block trước. Cuối cùng là 8 bytes tiếp sẽ lưu lần lượt vào disk 2 và disk 3, disk1 lại được tính theo toán tử XOR từ 2 block trước

2.2. Cách thực hiện

- Nhập chuỗi ký tự từ bàn phím
- Lưu chuỗi vừa nhập vào vùng nhớ rồi sau đó
- Kiểm tra xem độ dài kí tự nhập vào có phải là bội của 8 không, nếu không thì nhập lại
- Lấy lần lượt từng kí tự ở vị trí 0 và 4 của chuỗi xor với nhau và lưu vào mảng kết quả, đồng thời độ dài của chuỗi giảm.
- Chuyển từ hệ nhị phân sang hệ 16.
- Tạo vòng lặp 8 lần rồi sau đó dịch trái

3. Mã nguồn

```
.data
start: .asciiz "Nhap chuoi ky tu : "
hex: .byte
'0','1','2','3','4','5','6','7','8','9','a','b','c','d','e','f'
#dung chuyen doi ma ascii sang hexa
d1: .space 4 #tuong trung cho disk1,disk2,disk3 co 4 byte
d2: .space 4
d3: .space 4
array: .space 32 #l?u tru c□c k□ tu dc XOR
string: .space 5000 #chuoi ki input
enter: .asciiz "\n"
error_length: .asciiz "Chuoi ky tu co do dai khong la boi cua 8,
moi nhap lai\n"
disk: .asciiz  "      Disk 1               Disk 2
Disk 3\n"
ms1: .asciiz   "-----------------      ----------------       -----
-----------\n"
ms2: .asciiz   "|     "
ms3: .asciiz   "      |      "
ms4: .asciiz  "[[ "
ms5: .asciiz "]]       "
comma: .asciiz ","
message: .asciiz "Try again?"

.text
main:
     la $s1, d1                    # s1 = address of disk 1
     la $s2, d2                    # s2 = address of disk 2
     la $s3, d3                    # s3 = address of disk 3
     la $a2, array                 # address of parity'array

     j input
     nop

input:    li $v0, 4                # print " Nhap chuoi ky tu"
     la $a0, start
     syscall

     li $v0, 8                # Get string
```

```
     la $a0, string
     li $a1, 1000
     syscall

     move $s0, $a0                  # s0 chua dia chi xau moi nhap



#check ?? d�i k? t?
length:
     addi $t3, $zero, 0             # t3 = length
     addi $t0, $zero, 0             # t0 = index

check_char:
# H�m kiem tra k? tu:  k? tu ket th?c: "\n"
     add $t1, $s0, $t0              # t1 = address of string[i]
     lb $t2, 0($t1)                      # t2 = string[i]
     beq $t2, 10, test_length            # khi stirng[i]  = '\n' ket
thuc kiem tra k? tu ket th?c
     nop

     addi $t3, $t3, 1               # length++
     addi $t0, $t0, 1               # index++
     j check_char
     nop

test_length:
     move $t5, $t3                  # t5 chua dia chi length
     beq $t0,0,error                # if has only "\n" -> error

     and $t1, $t3, 0x0000000f       # xoa het cac byte cua $t3 ve 0,
chi giu lai byte cuoi
     bne $t1, 0, test1              # byte cuoi bang 0 hoac 8 thi so
chia het cho 8
     li $v0, 4                 # print " Disk1 Disk2 Disk3"
     la $a0, disk
     syscall
     li $v0, 4                 # print " ------ "
     la $a0, ms1
     syscall
     j block1
```

```
     nop

test1:    beq $t1, 8, block1            # neu byte cuoi  != 8 va != 
0 => error
     j error
     nop

error:    li $v0, 4                # Ham in loi thong bao
     la $a0, error_length
     syscall
     j input                       # bat nguoi dung nhap lai input
     nop

#end_check

HEX:
#Lay Parity
# Co 1 dau vao la t8 chua parity string roi chuyen tu ascii sang 
hexa
     li $t4, 7               #t4 = 7

loopH:
     blt $t4, $0, endloopH              # t4 < 0  -> endloop
     sll $s6, $t4, 2                    # s6 = t4*4
     srlv $a0, $t8, $s6            # a0 = t8>>s6
     andi $a0, $a0, 0x0000000f          # a0 = a0 & 0000 0000 0000 
0000 0000 0000 0000 1111 => lay byte cuoi cung cua a0
     la $t7, hex                   # t7 = adrress of hex
     add $t7, $t7, $a0             # t7 = t7 + a0
     bgt $t4, 1, nextc             # if t4 > 1 , jump to nextC
     lb $a0, 0($t7)                     # print hex[a0]
     li $v0, 11
     syscall


nextc:    addi $t4,$t4,-1                  # t4 --
     j loopH
     nop

endloopH:
     jr $ra
```

```
     nop

#RAID5
#----------------lan 1: luu  2 khoi 4-byte vao  disk 1,2; xor vao
disk 3---------------------
RAID5:
# RAID 5 gom 3 phan,
#block 1 : byte parity luu vao disk 3
#block 2 : byte parity luu vao disk 2
#block 3 : byte parity luu vao disk 1
block1:
#Funtion block1:Lan thu nhat xet 2 khoi 4 byte  luu vao Disk 1 ,
Disk 2 ;
#-------------- Byte parity luu vao Disk 3;

     addi $t0, $zero, 0            # so byte duoc in ra (4 byte)
     addi $t9, $zero, 0
     addi $t8, $zero, 0
     la $s1, d1                    # s1 = adress of d1
     la $s2, d2                    # s2 = address of d2
     la $a2, array                 #

print11:
     li $v0, 4                # print message2 : "|     "
     la $a0, ms2
     syscall

#    vi du DCE.****
b11:
# luu DCE. vao disk 1
     lb $t1, ($s0)                 # t1 = first value of input
string
     addi $t3, $t3, -1             # t3 = length -1,giam do dai sau
can xet
     sb $t1, ($s1)                 # store t1 to disk 1
b12:
# luu **** vao disk 2
     add $s5, $s0, 4                    # s5 = s0 +4
     lb $t2, ($s5)                 # t2 = inputstring[5]
     addi $t3, $t3, -1             # t3 = t3  - 1  , giam do dai
xau can xet
```

```
     sb $t2, ($s2)                 # store t2 vao disk 2
b13:
# luu ket qua xor vao disk 3
     xor $a3, $t1, $t2             # a3 = t1 xor t2
     sw $a3, ($a2)                 # luu a3 vao dia chi chuoi a2
     addi $a2, $a2, 4              #   parity string
     addi $t0, $t0, 1              # xet char tiep theo
     addi $s0, $s0, 1              # loai bo ki tu vua xet , Vi
du : "D"
     addi $s1, $s1, 1              # tang dia chi disk 1 len 1
     addi $s2, $s2, 1              # tang dia chi disk 2 len 1
     bgt $t0, 3, reset             #  da xet duoc 4 byte , reset
disk
     j b11
     nop
reset:
     la $s1, d1                    # reset con tro ve  disk 1 VD :
"D" trong "DCE."
     la $s2, d2                    # reset con tro ve disk 2

print12:            #in Disk 1
     lb $a0, ($s1)            #print each char  in Disk 1
     li $v0, 11
     syscall
     addi $t9, $t9, 1
     addi $s1, $s1, 1
     bgt $t9, 3, next11       # sau khi in du 4 lan => in het Disk
1
     j print12
     nop

next11:                  #Ham chuan bi bat dau de print Disk 2
"|         |"
     li $v0, 4
     la $a0, ms3
     syscall
     li $v0, 4
     la $a0, ms2
     syscall

print13:            # Ham print disk 2
```

```
     lb $a0, ($s2)
     li $v0, 11
     syscall
     addi $t8, $t8, 1
     addi $s2, $s2, 1
     bgt $t8, 3, next12            # in dc 4 byte => xong Disk 2
     j print13
     nop

next12:                  # ham chuan bi in Disk 3
     li $v0, 4
     la $a0, ms3
     syscall
     li $v0, 4
     la $a0, ms4
     syscall
     la $a2, array            # a2 = address of parity string[i]
     addi $t9, $zero, 0       # t9 = i

print14:                 # Ham chuyen doi  parity string -> ma
ASCII va in ra man hinh
     lb $t8, ($a2)            # t8 = adress of parity string[i]
     jal HEX
     nop
     li $v0, 4
     la $a0, comma            # print  " , "
     syscall

     addi $t9, $t9, 1         # parity string's index  + 1
     addi $a2, $a2, 4         # bo qua parity string da xet'
     bgt $t9, 2, end1         # in ra 3 parity dau co dau ",",
parity cuoi cung k co
     j print14
end1:                    # in ra parity cuoi cung va hoan thanh
Disk 3
     lb $t8, ($a2)
     jal HEX
     nop
     li $v0, 4
     la $a0, ms5
     syscall
```

```
     li $v0, 4           # xuong dong , bat dau khoi block moi
     la $a0, enter
     syscall
     beq $t3, 0, exit1        # neu length string con lai can xet =
0 , exit
     j block2            # neu con lai ki tu can xet => block2
     nop

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

block2:
#Ham block 2 :
# xet 2 khoi 4  byte tiep theo  vao Disk 1 va Disk 3;  byte parity
vao Disk 2

     la $a2, array
     la $s1, d1                  # s1 = address of Disk 1
     la $s3, d3                  # s3 =            Disk 3
     addi $s0, $s0, 4
     addi $t0, $zero, 0

print21:
# print "|     "
     li $v0, 4
     la $a0, ms2
     syscall

b21:
# xet tung byte trong 4 byte dau vao Disk 1
     lb $t1, ($s0)               # t1 = address of Disk 1
     addi $t3, $t3, -1           # length con' phai kiem tra   -1
     sb $t1, ($s1)
b23:
# xet 4 byte ke tiep vao Disk 3
     add $s5, $s0, 4
     lb $t2, ($s5)
     addi $t3, $t3, -1
     sb $t2, ($s3)
```

```
b22:
#Tinh 4 byte parity vao Disk 2
     xor $a3, $t1, $t2
     sw $a3, ($a2)
     addi $a2, $a2, 4
     addi $t0, $t0, 1
     addi $s0, $s0, 1
     addi $s1, $s1, 1
     addi $s3, $s3, 1
     bgt $t0, 3, reset2
     j b21
     nop
reset2:
     la $s1, d1               # reset de chuan bi print ra Disk 1
     la $s3, d3               # reset de chuan bi print ra Disk 3
     addi $t9, $zero, 0       # index

print22:
# print Disk 1
     lb $a0, ($s1)
     li $v0, 11
     syscall
     addi $t9, $t9, 1
     addi $s1, $s1, 1
     bgt $t9, 3, next21
     j print22
     nop

next21:        # print khoang cach
     li $v0, 4
     la $a0, ms3
     syscall
     la $a2, array
     addi $t9, $zero, 0
     li $v0, 4
     la $a0, ms4
     syscall

print23:  # print Disk 2 chua byte parity
     lb $t8, ($a2)
     jal HEX                  # chuyen doi ve ASCII
```

```
     nop
     li $v0, 4
     la $a0, comma            #print ","
     syscall
     addi $t9, $t9, 1
     addi $a2, $a2, 4
     bgt $t9, 2, next22
     j print23
     nop

next22:
#print Disk 2 theo ACSII
     lb $t8, ($a2)
     jal HEX
     nop

     li $v0, 4
     la $a0, ms5
     syscall

     li $v0, 4
     la $a0, ms2
     syscall
     addi $t8, $zero, 0

print24:
# print Disk 3
     lb $a0, ($s3)
     li $v0, 11
     syscall
     addi $t8, $t8, 1
     addi $s3, $s3, 1
     bgt $t8, 3, end2
     j print24
     nop

end2:
# Neu string can xet da het thi nhay den nhan exit1
# chua het thi tiep tuc block3
     li $v0, 4
     la $a0, ms3
```

```
     syscall
     li $v0, 4
     la $a0, enter
     syscall
     beq $t3, 0, exit1
#-----------------------------------------------------------
block3:
# Byte parity duoc luu o Disk1
# 2 block 4 byte dc luu vao Disk 2 , Disk 3
     la $a2, array
     la $s2, d2
     la $s3, d3
     addi $s0, $s0, 4              # xet den vi tri 4 byte hien tai
     addi $t0, $zero, 0            # index
print31:
# chuan bi print parity print: "[[ "
     li $v0, 4
     la $a0, ms4
     syscall
b32:
#  byte stored in Disk 2
#Vi du DCE.****ABCD1234HUSTHUST
     lb $t1, ($s0)             # in first loop, t1 = first H
     addi $t3, $t3, -1
     sb $t1, ($s2)
b33:
     # store in Disk 3 first
     add $s5, $s0, 4               #
     lb $t2, ($s5)             # in first loop , t2 = the second "H"

     addi $t3, $t3, -1         # stored in disk 3
     sb $t2, ($s3)             # stored t2 in disk 3

b31:
# ham xor tinh parity
     xor $a3, $t1, $t2         # a3 = parity number
     sw $a3, ($a2)             # stored in parity string
     addi $a2, $a2, 4          # parity string's index + 4
     addi $t0, $t0, 1          # index so char dang xet
     addi $s0, $s0, 1          # loai bo ki tu da xet , VD: "H",
string dang xet la "USTHUST"
```

```
     addi $s2, $s2, 1          #    disk2 +1
     addi $s3, $s3, 1          #    disk 3 +1
     bgt $t0, 3, reset3        # net xet duoc 4 lan , thoat khoi
vong lap
     j b32                     # neu chua xet du 4 byte , tiep tuc
xet
     nop
reset3:
# to first of disk2 , disk 3
     la $s2, d2
     la $s3, d3
     la $a2, array
     addi $t9, $zero, 0        #index

print32:
# Ham' print parity byte duoi dang ASCII
     lb $t8, ($a2)             # luu chuoi can chuyen duoi ASCII
     jal HEX                   # dung ham HEX de chuyen duoi ve
ASCII
     nop
     li $v0, 4           # print
     la $a0, comma
     syscall

     addi $t9, $t9, 1
     addi $a2, $a2, 4          # loai bo parity string da duoc xet
     bgt $t9, 2, next31        # neu in du 3 lan dau phay -> next31
     j print32
     nop

next31:
# print 1 byte parity con' lai
     lb $t8, ($a2)
     jal HEX
     nop

     li $v0, 4
     la $a0, ms5
     syscall
     li $v0, 4
     la $a0, ms2
```

```
     syscall
     addi $t9, $zero, 0

print33:
#print disk 2, print 4 byte from Disk 2
     lb $a0, ($s2)
     li $v0, 11
     syscall
     addi $t9, $t9, 1
     addi $s2, $s2, 1
     bgt $t9, 3, next32
     j print33
     nop

next32:
# print ki tu ngan cach
     addi $t9, $zero, 0
     addi $t8, $zero, 0
     li $v0, 4
     la $a0, ms3
     syscall
     li $v0, 4
     la $a0, ms2
     syscall
print34:
#  print  4 byte from Disk 3
     lb $a0, ($s3)
     li $v0, 11
     syscall
     addi $t8, $t8, 1
     addi $s3, $s3, 1
     bgt $t8, 3, end3
     j print34
     nop

end3:
#in ra cac ki tu ket thuc khi liet ke Disk 3
     li $v0, 4
     la $a0, ms3              # ki tu : "      |"
     syscall
```

```
     li $v0, 4
     la $a0, enter            # ki tu xuong dong
     syscall
     beq $t3, 0, exit1        # neu ko con ki tu can xet -> exit
                         #neu con ki tu can xet -> tro ve block1

#---------------------------------end 6 block 4-byte dau------------
---------------------------
# chuyen sang 6 block 4-byte tiep theo

nextloop: addi $s0, $s0, 4         #bo qua 4 ki tu da xet roi
     j block1
     nop

exit1:    # in ra dong ------ va ket thuc mo phong RAID
     li $v0, 4
     la $a0, ms1
     syscall
     j ask
     nop

#End RAID5

#Again
ask: li $v0, 50               #ask if try again
     la $a0, message
     syscall
     beq $a0, 0, clear        # a0 :     0 = yes;  1 = NO ;  2 =
cancel
     nop
     j exit
     nop

# H□m clear: dua string ve trang thai ban dau
clear:
     la $s0, string
     add $s3, $s0, $t5   # s3: dia chi byte cuoi cung duoc su dung
trong string
     li $t1, 0      # set t1 = 0

goAgain:       # Dua string ve trang thai rong~ de bat dau lai .
```

```
     sb $t1, ($s0)       # set byte o dia chi s0 thanh 0
     nop
     addi $s0, $s0, 1
     bge $s0, $s3, input
     nop
     j goAgain
     nop
#End Do again

exit:     li $v0, 10
     syscall
```

4. Chú thích

4.1. Ý nghĩa thanh ghi

$s0: chuỗi nhập vào
$t3: độ dài chuỗi
$t4: giá trị 4 bit cuối
$s1: giá trị của disk1
$s2:giá trị của disk2
$a2: giá trị của mảng kết quả

4.2. Ý nghĩa các hàm

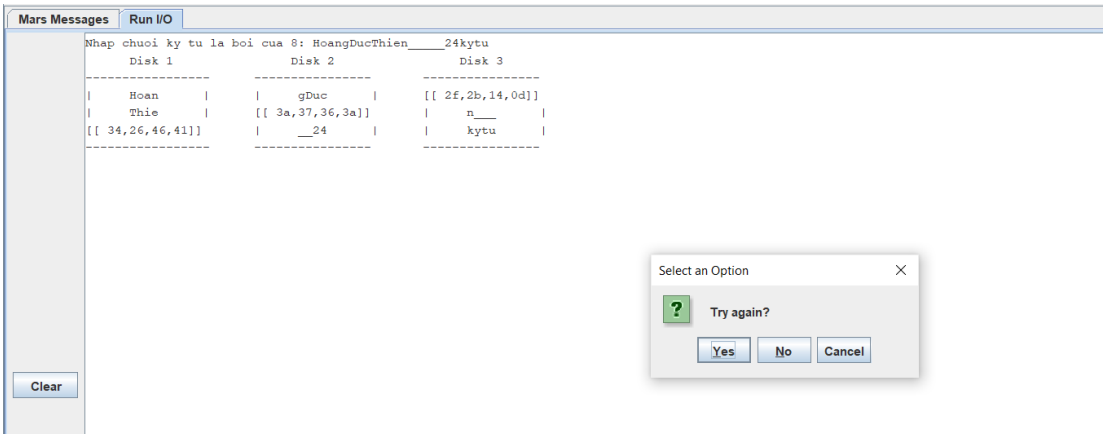
- input: hàm nhập chuỗi đầu vào
- length: khởi tạo các giá trị test_
- length: kiểm tra độ dài chuỗi
- hexa_convert: hàm chuyển từ hệ 2 sang 16 3
- convert_end: in ra kết quả sau khi chuyển sang hệ 16
- procedureX: hàm xử lý dữ liệu ở dòng thứ X tương ứng có
- b1X: hàm lấy kí tự ở vị trí 0
- b2X: hàm lấy kí tự ở vị trí 4
- b3X: hàm xor 2 giá trị vừa lấy
- reset: reset 2 ổ đĩa Với (X =1, 2, 3) Các hàm print22, print23, print24, print12, print13. print14: in giá trị các ổ đĩa trong từng procedure

– Ask: hàm hỏi có nhập lại không go Again: reset chuỗi nhập vào

5. Kết quả

```
Nhap chuoi ky tu la boi cua 8: HoangDucThien_Nhhap sai
      Disk 1                  Disk 2                  Disk 3
-----------------       -----------------       -----------------
Do dai chuoi khong hop le! Nhap lai.
Nhap chuoi ky tu la boi cua 8:
```

Nếu nhập chuỗi không là bội cùa 8 (chuỗi sai), yêu cầu nhập lại



Nếu chuỗi ký tự là bội 8 (nhập đúng), thực hiện thành công chương trình, hiện pop-up hỏi có thực hiện lại chương trình không

Bài 10: Máy tính bỏ túi

Sinh viên thực hiện: Nguyễn Thị Hạnh – 20194552

1. Đề bài:

Sử dụng 2 ngoại vi là bàn phím và led 7 thanh để xây dựng một máy tính bỏ túi đơn giản. Hỗ trợ các phép toán +, -, *, /. Do trên bàn phím không có các phím trên nên sẽ dùng các phím:

+ Bấm phím a để nhập phép tính +
+ Bấm phím b để nhập phép tính –
+ Bấm phím c để nhập phép tính *
+ Bấm phím d để nhập phép tính /
+ Bấm phím e để nhập phép tính %
+ Bấm phím f để nhập phép =

Yêu cầu cụ thể như sau:

+ Khi nhấn các phím số, hiển thị lên LED, do chỉ có 2 LED nên chỉ hiện thị 2 số cuối cùng. Ví dụ khi nhấn phím 1 → hiện thị 01. Khi nhấn thêm phím 2 → hiển thị 12. Khi nhấn thêm phím 3 → hiển thị 23.
+ Sau khi nhập số, sẽ nhập phép tính + - * / %
+ Sau khi nhấn phím f (dấu =), tính toán và hiển thị kết quả lên LED.
+ Có thể thực hiện các phép tính liên tiếp (tham khảo ứng dụng Calcutor trên hệ điều hành Windows)

Chú ý: Do bài toán sẽ có rất nhiều trường hợp xảy ra, yêu cầu cơ bản là thực hiện được phép tính và hiển thị lên LED. Các yêu cầu về bắt lỗi, các trường hợp tràn số,... là tùy chọn.

2. Phân tích cách thực hiện:

2.1. Phân tích đề bài:

– Input: Sử dụng công cụ Digital Lab Sim để nhập số và phép tính

– Output: Hiển thị kết quả ra màn hình và hiển thị lên LED. (Trường hợp lỗi hoặc nhập sai sẽ thông báo lỗi)

2.2. Ý tưởng thực hiện:

– Sử dụng kĩ thuật ngắt để lấy các các giá trị từ công cụ Digital Lab Sim

- Lần lượt nhập số thứ nhất, số thứ hai, toán tử và cuối cùng ấn f (=) để hiển thị kết quả ra màn hình.

2.3. Xử lí các ngoại lệ:

- Khi nhập toán tử, nếu nhập ký tự khác "a, b, c, d" thì khi ấn f (=) sẽ có thông báo lỗi. (Hoặc báo lỗi khi ấn f mà chưa nhập toán tử)
- Khi phép tính thực hiện là phép chia, sẽ có thông báo lỗi nếu số chia bằng 0.
- Nếu kết quả là một số âm, đưa ra thông báo lỗi không hiển thị được trên LED.

2.4. Các bước thực hiện:

- Bước 1: Khởi tạo các biến mặc định ứng với các số từ 0 đến 9 trên LED
- Bước 2: Đọc từng kí tự từ LED để lấy số thứ nhất, số thứ hai.
  + Sử dụng các vòng lặp vô hạn cho đến khi đạt được điều kiện ngắt.
  + Sử dụng biến đếm để kiểm tra việc nhập số, khi biến đếm tăng đến giá trị đặt trước (mặc định 4 chữ số) thì dừng vòng lặp và lưu giá trị số thứ nhất vào thanh ghi. Sau đó reset biến đếm và nhập tiếp số thứ hai.
  + Làm tương tự để có được số thứ hai.
- Bước 3: Đọc từ LED để lấy ra toán tử và dấu "=".
  + Dùng vòng lặp vô hạn và điều kiện ngắt để lấy ra tiếp toán tử và dấu "=".
  + Khi nhập toán tử, nếu nhập đúng a, b, c, d thì chương trình sẽ nhận được các toán tử tương ứng + - * / % đồng thời đưa ra thông báo đã nhập toán tử nào.
  + Nếu nhập ký tự khác thì khi nhập dấu "=" chương trình sẽ đưa ra thông báo lỗi.
- Bước 4: Kết thúc việc nhập. Hàm show sẽ kiểm tra toán tử đã nhập và thực hiện phép toán. Sau đó in kết quả ra console.
  + Nếu kết quả là 1 số âm, thông báo không hiển thị được lên LED.
  + Nếu là phép chia cho 0, thông báo lỗi không thực hiện được.
- Bước 5: Nạp giá trị cho LED và kết thúc chương trình.
  + Nếu kết quả là 1 số lớn hơn 2 chữ số, dùng phép chia cho 10 lấy phần dư để lấy ra giá trị hiển thị.

3. Ý nghĩa của các thanh ghi được sử dụng:

- $t6: Lưu giá trị của đèn LED trái
- $t7: Lưu giá trị của đèn LED phải

- $s1: Giá trị lấy ra
- $s2: Toán tử lấy ra
- $s3: Số thứ nhất
- $s4: Số thứ hai
- $s5: Dấu =
- $s6: Biến đếm

4. Mã nguồn

```
.data
  welc:           .asciiz "\n Welcome to Calculator MARS\n"
  p_int:          .asciiz "\n Enter the first 4 digit number.\n
The first number is: "
  p_int1:         .asciiz "\n Enter the second 4 digit number.\n
The second number is: "
  p_toantu:       .asciiz "\n Enter operator:\n Enter a to get the
addition.\n Enter b to get the subtraction. \n Enter b to get the
multiplication \n Enter b to get the division. \n Enter b to get
the modulo.\n "
  ketqua:         .asciiz "\n The result is:  "
  space:     .asciiz " "
  err1:           .asciiz  "\n  An  error  occurred.  Please  re-
enter.\n"
  err2:      .asciiz "\n You entered division by 0.\n Please re-
enter.\n"
  erram:          .asciiz "\n The result is a negative number,
which cannot be displayed on the LED.\n"
  sb_cong:   .asciiz "\n You have entered addition. \n"
  sb_tru:         .asciiz "\n You have entered subtraction. \n"
  sb_nhan:        .asciiz "\n You have entered multiplication. \n"
  sb_chia:        .asciiz "\n You have entered division. \n"
  sb_lay_du: .asciiz "\n You have entered modulo. \n"
  sb_daubang:     .asciiz "\n Enter f to display the results. \n"
  rep:            .asciiz "\n Enter 0 to exit the program. \n Enter
a non-zero number to continue. \n "
  goodbye:        .asciiz "\n Goodbye\n"
```

```
.eqv ZERO                   63        # Gia tri byte hien thi so 0
tren den LED
.eqv ONE                    6              # Gia tri byte hien thi
so 1 tren den LED
.eqv TWO                    91             # Gia tri byte hien thi
so 2 tren den LED
.eqv THREE                  79             # Gia tri byte hien thi
so 3 tren den LED
.eqv FOUR                   102       # Gia tri byte hien thi so 4
tren den LED
.eqv FIVE                   109       # Gia tri byte hien thi so 5
tren den LED
.eqv SIX                    125       # Gia tri byte hien thi so 6
tren den LED
.eqv SEVEN                  7              # Gia tri byte hien thi
so 7 tren den LED
.eqv EIGHT                  127       # Gia tri byte hien thi so 8
tren den LED
.eqv NINE                   111       # Gia tri byte hien thi so 9
tren den LED
.eqv IN_ADDRESS_HEXA_KEYBOARD   0xFFFF0012      # chua byte dieu
khien dong cua ban phim
.eqv OUT_ADDRESS_HEXA_KEYBOARD  0xFFFF0014      # chua byte tra ve
vi tri cua phim duoc bam
.eqv LEFT_LED            0xFFFF0010        # chua byte dieu khien
den led ben phai
.eqv RIGHT_LED           0xFFFF0011        # chua byte dieu khien
den led ben trai
.text
main:
start:
            la $a0, welc
            li $v0, 4
            syscall
            la $a0, p_int
            li $v0, 4
            syscall
# Kich hoat interupt ----------------------------------------
            li $t1, IN_ADDRESS_HEXA_KEYBOARD
            li $t3, 0x80    # Kich hoat interupt tu ban phim hexa
        sb $t3, 0($t1)
```

```
# Khai bao bien ------------------------------------------------
            li $t6, 0                # $t6: Bien gia tri so cua
den LED trai
            li $t7, 0                # $t7: Bien gia tri so cua
den LED phai
            li $s1, 0                # $s1: Gia tri lay ra
            li $s2, 0                # $s2: toan tu lay ra
            li $s5, 0                # $s5:  dau =
            li $s3, 0                # so thu nhat
            li $s4, 0                # so thu 2
            li $s6, 0                # bien dem s6
# Vong lap cho tin hieu interupt so thu nhat
Loop:       beq $s6, 4, nhapso2
            nop
            beq $s6, 4, nhapso2
            nop
            beq $s6, 4, nhapso2
            nop
            beq $s6, 4, nhapso2
            j Loop              #Wait for interrupt
            beq $s6, 4, nhapso2
            nop
            beq $s6, 4, nhapso2
            j Loop
#-----------------------------------------------------------------
nhapso2:    add $s3, $s1, $0 # luu gia tri so thu nhat vao s3
            add $s1, $0, $0 # reset s1
            addi $s6, $0, 0 # reset bien dem s6
            move $a0, $s3
            li $v0, 1
            syscall
            li $t6, 0      # reset den led
            li $t7, 0
            la $a0, p_int1
            li $v0, 4
            syscall
# Vong lap cho interupt so thu 2 ----------------------------
Loop1:      beq $s6, 4, nhaptoantu
            nop
            beq $s6, 4, nhaptoantu
            nop
```

```
             beq $s6, 4, nhaptoantu
             nop
             beq $s6, 4, nhaptoantu
             b Loop1      #Wait for interrupt
             beq $s6, 4, nhaptoantu
             nop
             beq $s6, 4, nhaptoantu
             b Loop1
nhaptoantu:  add $s4, $s1, $0 # luu gia tri so thu hai vao s4
             add $s1, $0, $0  # reset s1
             addi $s6, $0, 0 # reset bien dem s6
             la $a0, space
             li $v0, 4
             syscall
             move $a0, $s4  # in so thu 2
             li $v0, 1
             syscall
             li $t6, 0  # reset den led
             li $t7, 0
             la $a0, p_toantu
             li $v0, 4
             syscall
Loop2:       beq $s6, 1, nhaptoantu2
             nop
             beq $s6, 1, nhaptoantu2
             nop
             beq $s6, 1, nhaptoantu2
             nop
             beq $s6, 1, nhaptoantu2
             b Loop2    #Wait for interrupt
             beq $s6, 1, nhaptoantu2
             nop
             beq $s6, 1, nhaptoantu2
             b Loop2
nhaptoantu2: add $a3, $0, $0
             add $s1, $0, $0 # reset s1
             addi $s6, $0, 0 # reset bien dem s6
cong1:       bne $s2, 1, tru1
             la $a0, sb_cong
             li $v0, 4
             syscall
```

```
            j baonhapdaubang
tru1:       bne $s2, 2, nhan1
            la $a0, sb_tru
            li $v0, 4
            syscall
            j baonhapdaubang
nhan1:      bne $s2, 3, chia1
            la $a0, sb_nhan
            li $v0, 4
            syscall
            j baonhapdaubang
chia1:      bne $s2, 4, baonhapdaubang
            la $a0, sb_chia
            li $v0, 4
            syscall
            j baonhapdaubang
laydu1:     bne $s2, 5, baonhapdaubang
            la $a0, sb_lay_du
            li $v0, 4
            syscall
            j baonhapdaubang
baonhapdaubang:
            la $a0, sb_daubang
            li $v0, 4
            syscall

Loop3:      beq $s5, 6, show
            nop
            beq $s5, 6, show
            nop
            beq $s5, 6, show
            nop
            beq $s5, 6, show
            j Loop3    #Wait for interrupt
            beq $s5, 6, show
            nop
            beq $s5, 6, show
            j Loop3
show:
case_cong:  bne $s2, 1, case_tru  # neu la phep cong
            addu $s7, $s3, $s4 # thuc hien phep cong
```

```
            la $a0, ketqua
            li $v0, 4
            syscall
            move $a0, $s7 # in ket qua ra console
            li $v0, 1
            syscall
            j showketqua
case_tru:   bne $s2, 2, case_nhan
            la $a0, ketqua
            li $v0, 4
            syscall
            sub $s7, $s3, $s4
            move $a0, $s7
            li $v0, 1
            syscall
            j showketqua
case_nhan:  bne $s2, 3, case_chia
            la $a0, ketqua
            li $v0, 4
            syscall
            mul $s7, $s3, $s4
            move $a0, $s7
            li $v0, 1
            syscall
            j showketqua
case_chia:  bne $s2, 4, case_lay_du
            beq $s4, 0, chia_cho_0
            la $a0, ketqua
            li $v0, 4
            syscall
            div $s7, $s3, $s4
            move $a0, $s7
            li $v0, 1
            syscall
            j showketqua
case_lay_du: bne $s2, 5, pheptinhdf
            beq $s4, 0, chia_cho_0
            la $a0, ketqua
            li $v0, 4
            syscall
            div $s3, $s4
```

```
            mfhi $a0
            li $v0, 1
            syscall
            j showketqua

chia_cho_0: la $a0, err2   # loi chia cho 0
            li $v0, 4
            syscall
            j showketqua

pheptinhdf: la $a0, err1   # bao loi
            li $v0, 4
            syscall
            li $v0, 51
            la $a0, rep
            syscall
            beq $a0, $0, END
            nop
            j start
            nop
            la $v0, 10
            syscall

showketqua: li $t9, 0
            li $t8, 0
            slt $k1, $s7, $0 # kiem tra ket  qua la so am hay
khong
            bne $k1, $0, loisoam
            div $t8, $s7, 10
            mfhi $t9
            beq $t8, 0, napgiatricholed
            div $t8, $t8, 10
            mfhi $t8
napgiatricholed:
            li $t2, LEFT_LED   # hien thi den LED trai
            add $s0, $zero, $t9 # truyen bien left
            jal hienthi
            nop
            li $t2, RIGHT_LED  # hien thi den LED phai
            add $s0, $zero, $t8 # truyen bien right
            jal hienthi
```

```
            nop
            j endmain
endmain:
            li $v0, 51
            la $a0, rep
            syscall
            beq $a0, $0, END
            nop
            j start
            nop
END:
            la $a0, goodbye
            li $v0, 4
            syscall
            la $v0, 10
            syscall
loisoam:                        # bao loi ket qua am
             la $a0, erram
             li $v0, 4
             syscall
             li $v0, 51
             la $a0, rep
             syscall
             beq $a0, $0, END
             nop
             j start
             nop
             la $v0, 10
             syscall
hienthi:
led_0:      bne $s0, 0, led_1  # case $s0 = 0
            li $t4, ZERO
            j napgiatri
led_1:      bne $s0, 1, led_2  # case $s0 = 1
            li $t4, ONE
            j napgiatri
led_2:      bne $s0, 2, led_3  # case $s0 = 2
            li $t4, TWO
            j napgiatri
led_3:      bne $s0, 3, led_4  # case $s0 = 3
            li $t4, THREE
```

```
            j napgiatri
led_4:      bne $s0, 4, led_5  # case $s0 = 4
            li $t4, FOUR
            j napgiatri
led_5:      bne $s0, 5, led_6  # case $s0 = 5
            li $t4, FIVE
            j napgiatri
led_6:      bne $s0, 6, led_7  # case $s0 = 6
            li $t4, SIX
            j napgiatri
led_7:      bne $s0, 7, led_8  # case $s0 = 7
            li $t4, SEVEN
            j napgiatri
led_8:      bne $s0, 8, led_9  # case $s0 = 8
            li $t4, EIGHT
            j napgiatri
led_9:      bne $s0, 9, led_df # case $s0 = 9
            li $t4, NINE
            j napgiatri
led_df:     jr $ra
napgiatri:  sb $t4, 0($t2)
            jr $ra

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Xu ly khi xay ra interupt
# Hien thi so vua bam len den led 7 doan
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

.ktext 0x80000180
  #-----------------------------------------------------------
  # SAVE the current REG FILE to stack
  #-----------------------------------------------------------
IntSR:      addi $sp, $sp, 4    # Save $ra because we may change
it later
            sw $ra, 0($sp)
            addi $sp, $sp, 4    # Save $ra because we may change
it later
            sw  $at, 0($sp)
            addi $sp, $sp, 4    # Save $ra because we may change
it later
            sw $v0, 0($sp)
```

```
            addi $sp, $sp, 4    # Save $a0, because we may change
it later
            sw $a0, 0($sp)
            addi $sp, $sp, 4    # Save $t1, because we may change
it later
            sw $t1, 0($sp)
            addi $sp, $sp, 4    # Save $t3, because we may change
it later
            sw $t3, 0($sp)
            addi $sp, $sp, 4
            sw $s6, 0($sp)
  # ---------------------------------------------------------------
  # Processing
  # ---------------------------------------------------------------
            addi $s6, $s6, 1
            jal getInt1
            nop
            jal getInt2
            nop
            jal getInt3
            nop
            jal getInt4
            nop
next_pc:    mfc0 $at, $14    # $at <= Copro0.$14 = Coproc0.epc
            addi $at, $at, 4    # $at = $at + 4
            mtc0 $at, $14    #Coproc0.$14 = Coproc0.epc <= $at
  #---------------------------------------------------------------
  #---------------------------------------------------------------
  # RESTORE the REG FILE from STACK
  #---------------------------------------------------------------
restore:
            lw $t3, 0($sp)
            addi $sp, $sp, -4
            lw $t1, 0($sp)
            addi $sp, $sp, -4
            lw $a0, 0($sp)
            addi $sp, $sp, -4
            lw $v0, 0($sp)
            addi $sp, $sp, -4
            lw $ra, 0($sp)
            addi $sp, $sp, -4
```

```
             lw $t4, 0($sp)
             addi $sp, $sp, -4
back_main:   eret
  #-----------------------------------------------------------
  # Thu tuc quet cac phim o hang 1 va xu ly
  # Tham so truyen vao:
  # Tra ve:
  #-----------------------------------------------------------
getInt1:     addi $sp, $sp, 4
             sw $ra, 0($sp)
             li $t1, IN_ADDRESS_HEXA_KEYBOARD
             li $t3, 0x81     # Kich hoat interrupt, cho phep bam
phim o hang 1
             sb $t3, 0($t1)
             li $t1, OUT_ADDRESS_HEXA_KEYBOARD
             lb $t3, 0($t1)   # Nhan byte the hien vi tri cua phim
duoc bam trong hang 1
case_0:      li $t5, 0x11
             bne $t3, $t5, case_1  # case 0x11
             addi $t7, $t6, 0    # left=right
             addi $t6, $zero, 0  # left = 0
             mul $s1, $s1, 10
             add $s1, $s1, $t6   # factor=factor*10+left
             j show1
case_1:      li $t5, 0x21
             bne $t3, $t5, case_2  # case 0x21
             addi $t7, $t6, 0    # left=right
             addi $t6, $zero, 1  # left = 1
             mul $s1, $s1, 10
             add $s1, $s1, $t6 # factor=factor*10+left
             j show1
case_2:      li $t5, 0x41
             bne $t3, $t5, case_3  # case 0x41
             addi $t7, $t6, 0    # left=right
             addi $t6, $zero, 2  # left = 2
             mul $s1, $s1, 10
             add $s1, $s1, $t6 # factor=factor*10+left
             j show1
case_3:      li $t5, 0xffffff81
             bne $t3, $t5, case_default1  # case 0xffffff81
             addi $t7, $t6, 0    # left=right
```

```
            addi $t6, $zero, 3  # left = 3
            mul $s1, $s1, 10
            add $s1, $s1, $t6 # factor=factor*10+left
            j show1
show1:      li $t2, LEFT_LED   # hien thi den LED trai
            add $s0, $zero, $t6 # truyen bien left
            jal displayLED
            nop
            li $t2, RIGHT_LED  # hien thi den LED phai
            add $s0, $zero, $t7 # truyen bien right
            jal displayLED
            nop
case_default1:  j getInt1rt
getInt1rt:  lw $ra, 0($sp)
            addi $sp, $sp, -4
            jr $ra
#################### GETCODE 2 ####################
getInt2:    addi $sp, $sp, 4
            sw $ra, 0($sp)
            li $t1, IN_ADDRESS_HEXA_KEYBOARD
            li $t3, 0x82     # Kich hoat interrupt, cho phep bam
phim o hang 1
            sb $t3, 0($t1)
            li $t1, OUT_ADDRESS_HEXA_KEYBOARD
            lb $t3, 0($t1)   # Nhan byte the hien vi tri cua phim
duoc bam trong hang 1
case_4:     li $t5, 0x12
            bne $t3, $t5, case_5     # case 0x11
            addi $t7, $t6, 0         # left=right
            addi $t6, $zero, 4       # left = 4
            mul $s1, $s1, 10
            add $s1, $s1, $t6        # factor=factor*10+left
            j show2
case_5:     li $t5, 0x22
            bne $t3, $t5, case_6     # case 0x21
            addi $t7, $t6, 0         # left=right
            addi $t6, $zero, 5       # left = 5
            mul $s1, $s1, 10
            add $s1, $s1, $t6   # factor=factor*10+left
            j show2
case_6:     li $t5, 0x42
```

```
             bne $t3, $t5, case_7     # case 0x41
             addi $t7, $t6, 0         # left=right
             addi $t6, $zero, 6       # left = 6
             mul $s1, $s1, 10
             add $s1, $s1, $t6        # factor=factor*10+left
             j show2
case_7:      li $t5, 0xffffff82
             bne $t3, $t5, case_default2  # case 0xffffff81
             addi $t7, $t6, 0         # left=right
             addi $t6, $zero, 7       # left = 7
             mul $s1, $s1, 10
             add $s1, $s1, $t6        # factor=factor*10+left
             j show2
show2:       li $t2, LEFT_LED         # hien thi den LED trai
             add $s0, $zero, $t6      # truyen bien left
             jal displayLED
             nop
             li $t2, RIGHT_LED        # hien thi den LED phai
             add $s0, $zero, $t7      # truyen bien right
             jal displayLED
             nop
case_default2:  j getInt2rt
getInt2rt:   lw $ra, 0($sp)
             addi $sp, $sp, -4
             jr $ra
###################### GETCODE 3 ######################
 getInt3:    addi $sp, $sp, 4
             sw $ra, 0($sp)
             li $t1, IN_ADDRESS_HEXA_KEYBOARD
             li $t3, 0x84     # Kich hoat interrupt, cho phep bam
phim o hang 3
             sb $t3, 0($t1)
             li $t1, OUT_ADDRESS_HEXA_KEYBOARD
             lb $t3, 0($t1)   # Nhan byte the hien vi tri cua phim
duoc bam trong hang 3
case_8:      li $t5, 0x00000014
             bne $t3, $t5, case_9     # case 0x14
             addi $t7, $t6, 0         # left=right
             addi $t6, $zero, 8       # left = 8
             mul $s1, $s1, 10
             add $s1, $s1, $t6        # factor=factor*10+left
```

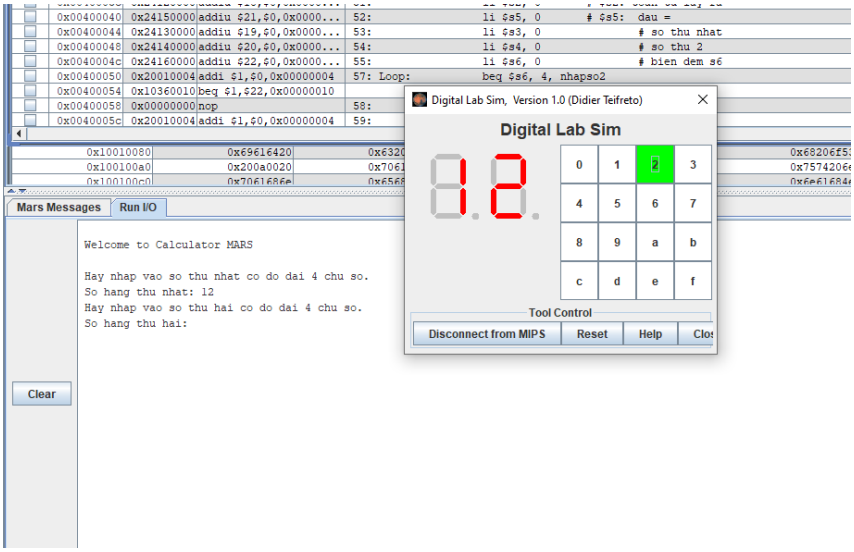
```
            j show3
case_9:     li $t5, 0x00000024
            bne $t3, $t5, case_a     # case 0x24
            addi $t7, $t6, 0         # left=right
            addi $t6, $zero, 9       # left = 9
            mul $s1, $s1, 10
            add $s1, $s1, $t6        # factor=factor*10+left
            j show3
case_a:     li $t5, 0x44
            bne $t3, $t5, case_b     # case 0x44
            addi $s2, $0, 1
            j case_default3
case_b:     li $t5, 0xffffff84
            bne $t3, $t5, case_default3
            addi $s2, $0, 2
        j  case_default3
show3:      li $t2, LEFT_LED         # hien thi den LED trai
            add $s0, $zero, $t6      # truyen bien left
            jal displayLED
            nop
            li $t2, RIGHT_LED        # hien thi den LED phai
            add $s0, $zero, $t7      # truyen bien right
            jal displayLED
            nop
case_default3:  j getInt3rt
getInt3rt:  lw $ra, 0($sp)
            addi $sp, $sp, -4
            jr $ra
#################### GETCODE 4 ####################
 getInt4:   addi $sp, $sp, 4
            sw $ra, 0($sp)
            li $t1, IN_ADDRESS_HEXA_KEYBOARD
            li $t3, 0x88     # Kich hoat interrupt, cho phep bam
phim o hang 4
            sb $t3, 0($t1)
            li $t1, OUT_ADDRESS_HEXA_KEYBOARD
            lb $t3, 0($t1)   # Nhan byte the hien vi tri cua phim
duoc bam trong hang 4
case_c:     li $t5, 0x18
            bne $t3, $t5, case_d     # case 0x44
            addi $s2, $0, 3
```

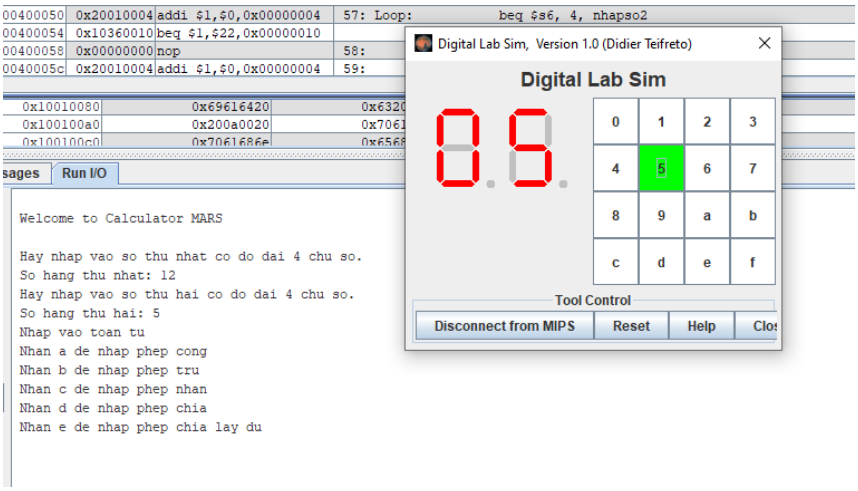
```
             j case_default3
case_d:      li $t5, 0x28
             bne $t3, $t5, case_e
             addi $s2, $0, 4
             j  case_default4
case_e:      li $t5, 0x48
             bne $t3, $t5, case_f
             addi $s2, $0, 5
             j case_default4
case_f:      li $t5, 0xffffff88
             bne $t3, $t5, case_default4
             addi $s5, $0, 6
             j case_default4
case_default4:  j getInt4rt
getInt4rt:      lw $ra, 0($sp)
                addi $sp, $sp, -4
                jr $ra
  #-------------------------------------------------------------
-----------
  # Thu tuc hien thi den LED
  # Tham so truyen vao: $t2 (dia chi cua LEFT_LED hoac RIGHT_LED),
$s0 : bien kieu int
  # Den LED $t2 se hien thi so $s0
  #-------------------------------------------------------------
-----------
displayLED:  addi $sp, $sp, 4
             sw $ra, 0($sp)      # save $ra
display:
so_0:        bne $s0, 0, so_1    # case $s0 = 0
             li $t4, ZERO
             j assign
so_1:        bne $s0, 1, so_2    # case $s0 = 1
             li $t4, ONE
             j assign
so_2:        bne $s0, 2, so_3    # case $s0 = 2
             li $t4, TWO
             j assign
so_3:        bne $s0, 3, so_4    # case $s0 = 3
             li $t4, THREE
             j assign
so_4:        bne $s0, 4, so_5    # case $s0 = 4
```

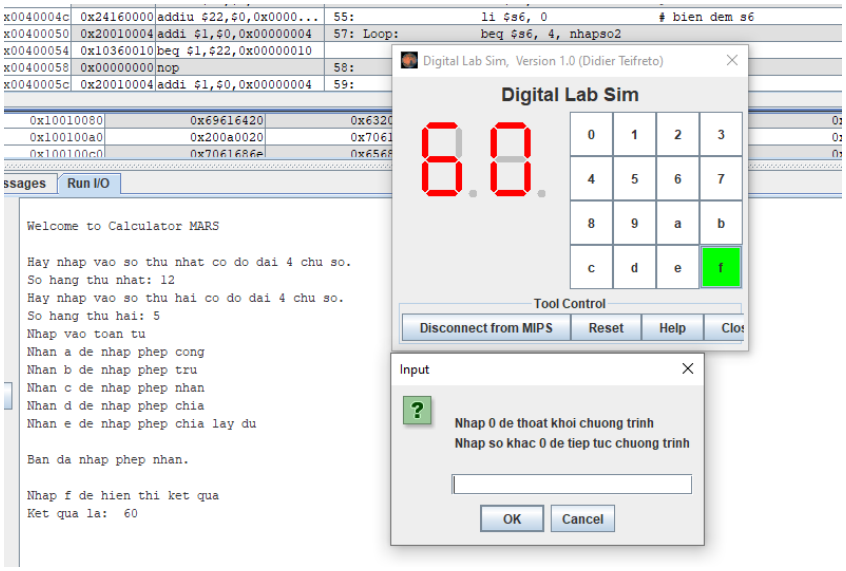
```
            li $t4, FOUR
            j assign
so_5:       bne $s0, 5, so_6    # case $s0 = 5
            li $t4, FIVE
            j assign
so_6:       bne $s0, 6, so_7    # case $s0 = 6
            li $t4, SIX
            j assign
so_7:       bne $s0, 7, so_8    # case $s0 = 7
            li $t4, SEVEN
            j assign
so_8:       bne $s0, 8, so_9    # case $s0 = 8
            li $t4, EIGHT
            j assign
so_9:       bne $s0, 9, hienthi_df # case $s0 = 9
            li $t4, NINE
            j assign
hienthi_df: j displayLEDrt
assign:     sb $t4, 0($t2)
displayLEDrt:     lw $ra, 0($sp)
            addi $sp, $sp, -4
            jr $ra
```

5. Kết quả

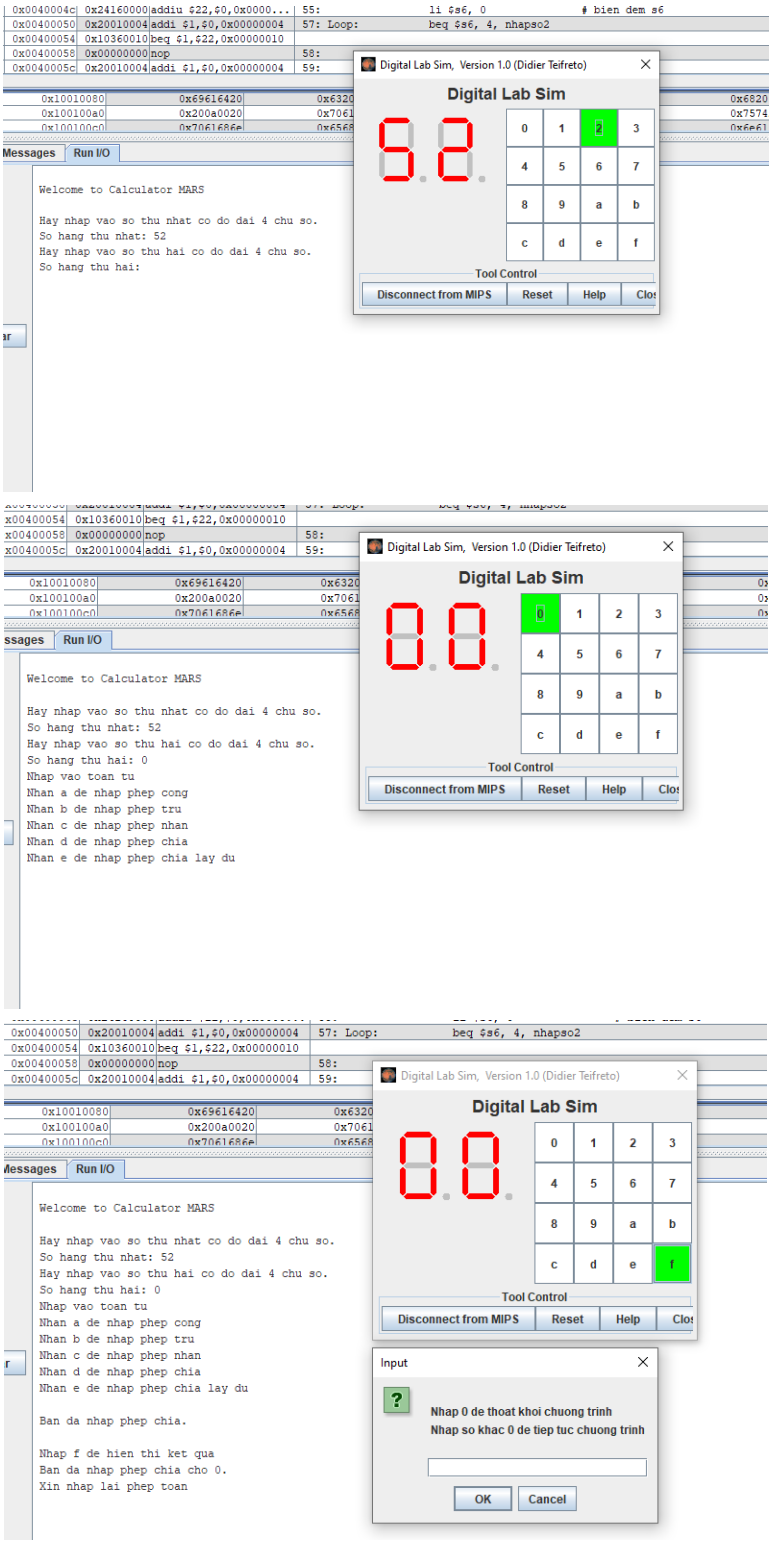
– Phép toán đúng và chương trình chạy bình thường



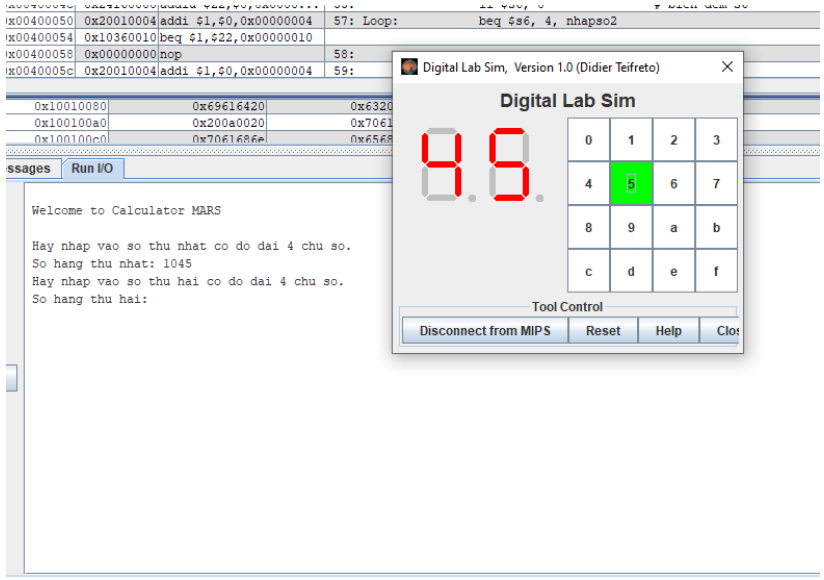


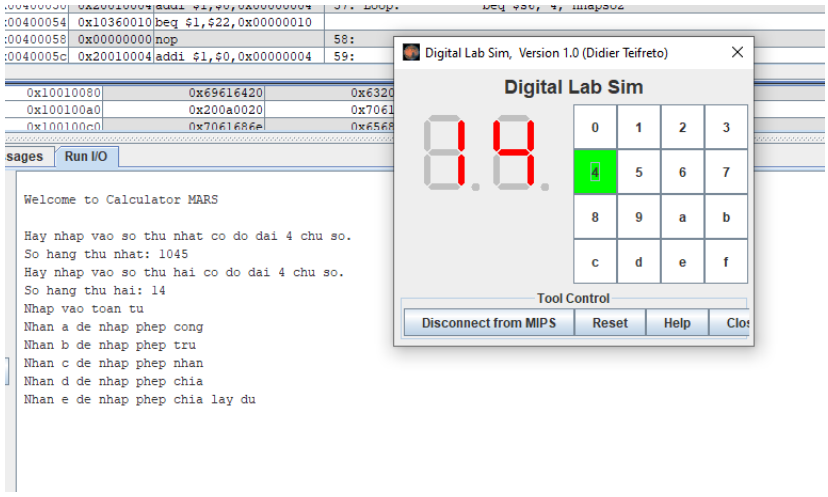


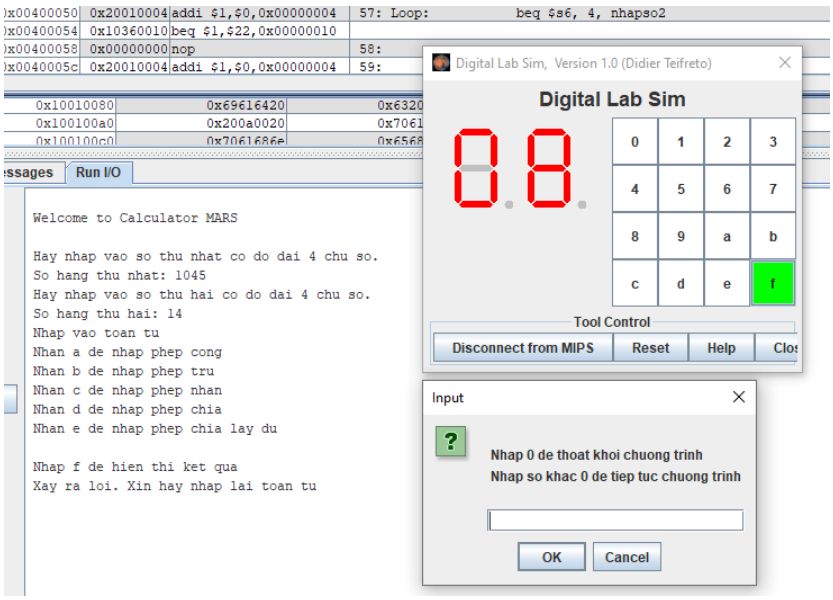
– Lỗi phép chia với số 0



– Lỗi phép toán







– Lỗi kết quả số âm