TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

———————— * ————————

BÁO CÁO

Thực hành kiến trúc máy tính

Giảng viên hướng dẫn: ThS. Lê Bá Vui
Phạm Quang Hà – 20194546
Nguyễn Tiến Nhật Minh - 20194621

Hà Nội, tháng 7 năm 2022

Contents

BÀI 6

Sinh viên thực hiện: Phạm Quang Hà – 20194546

I. Đề bài

Chương trình cho bên dưới là hàm malloc(), kèm theo đó là ví dụ minh họa, được viết bằng hợp ngữ MIPS, để cấp phát bộ nhớ cho một biến con trỏ nào đó. Hãy đọc chương trình và hiểu rõ nguyên tắc cấp phát bộ nhớ động.Trên cơ sở đó, hãy hoàn thiện chương trình như sau: (Lưu ý, ngoài viết các hàm đó, cần viết thêm một số ví dụ minh họa để thấy việc sử dụng hàm đó như thế nào)

1) Việc cấp phát bộ nhớ kiểu word/mảng kiểu word có 1 lỗi, đó là chưa bảo đảm qui tắc địa chỉ của kiểu word phải chia hết cho 4. Hãy khắc phục lỗi này.
2) Viết hàm lấy giá trị của biến con trỏ.
3) Viết hàm lấy địa chỉ biến con trỏ.
4) Viết hàm thực hiện copy 2 con trỏ xâu kí tự.
5) Viết hàm giải phóng bộ nhớ đã cấp phát cho các biến con trỏ
6) Viết hàm tính toàn bộ lượng bộ nhớ đã cấp phát.
7) Hãy viết hàm malloc2 để cấp phát cho mảng 2 chiều kiểu .word với tham số vào gồm:
a. Địa chỉ đầu của mảng
b. Số dòng
c. Số cột
8) Tiếp theo câu 7, hãy viết 2 hàm getArray[i][j] và setArray[i][j] để lấy/thiết lập giá trị cho phần tử ở dòng i cột j của mảng.

II. Phân tích xây dựng thuật toán

- Cấp phát bộ nhớ cho các con trỏ kiểu char, byte, word. Chương trình ban đầu ở đề bài có 1 lỗi đó là qui tắc địa chỉ kiểu word phải chia hết cho 4. Vấn đề đã được xử lý bằng cách tính số lượng thanh ghi cần cấp phát. Mỗi thanh ghi có thể chứa 4 byte dữ liệu và với kích thước các mảng được cho trước ta hoàn toàn có thể tính được số thanh ghi cần cấp phát qua công thức:

+ Nếu độ rộng của mảng chia hết cho 4:
  Số thanh ghi = Độ rộng mảng / 4
+ Nếu độ rộng của mảng chia 4 có dư:
  Số thanh ghi = [Độ rộng mảng / 4] + 1

- Thực hiện gán giá trị cho các con trỏ để kiểm tra các hàm phía sau.

Hàm lấy giá trị của CharPtr, BytePtr, WordPtr: Đầu tiên ta lấy địa chỉ của con trỏ ($a0), truy cập đến địa chỉ vùng nhớ của con trỏ bằng địa chỉ vừa lấy được ($t1), sau đó tuỳ theo kiểu dữ liệu mà chúng ta thực hiện phương thức khác nhau để lấy dữ liệu ($t0):

- Ở con trỏ kiểu Char và kiểu Byte: gọi đến hàm value sử dụng lb để lấy giá trị con trỏ. Và sử dụng hàm in kiểu string bằng cách gọi syscall $v0 = 4

• Ở con trỏ kiểu Word: gọi đến hàm value_word sử dụng lw để lấy giá trị con trỏ. Và sử dụng hàm in kiểu hexa bằng cách gọi syscall $v0 = 34

- Hàm lấy địa chỉ các con trỏ CharPtr, BytePtr, WordPtr: Sử dụng giá trị địa chỉ vùng nhớ con trỏ được lưu ở $t1 của hàm lấy giá trị, thực hiện in địa chỉ của con trỏ qua lệnh gọi syscall $v0 = 34 in địa chỉ kiểu hexa

- Hàm thực hiện copy 2 con trỏ xâu kí tự (CharPtr): Lấy kích thước mảng cần copy, ở đây là con trỏ CharPtr có lưu số thanh ghi cần thiết ở thanh ghi $s5, sau đấy gọi đến hàm CopyChar. Hàm CopyChar sẽ khởi tạo vùng nhớ cho con trỏ Copy bằng cách lấy địa chỉ vùng nhớ trống ở Sys_TheTopOfFree và kích thước vùng nhớ bằng kích thước vùng nhớ con trỏ cần Copy.

Hàm Copy sẽ thực hiện việc copy dữ liệu 2 con trỏ. Sử dụng lw để lấy dữ liệu thanh ghi của con trỏ cần copy và sw để lưu dữ liệu đó vào con trỏ mới .Sử dụng thanh ghi $t5 để làm biến đếm số thanh ghi cần copy. Khi số thanh ghi cần copy = 0 thì sẽ thoát hàm để quay về main

- Hàm tính toán lượng bộ nhớ đã cấp phát cho các biến động: Lấy ra giá trị biến chứa địa chỉ đầu tiên của vùng nhớ còn trống ở Sys_TheTopOfFree trừ đi giá trị vùng không gian tự do dùng để cấp phát bộ nhớ cho các biến con trỏ ở Sys_MyFreeSpace. Sử dụng lệnh in kiểu integer để ra lượng bộ nhớ đã cấp phát (byte).
- Hàm malloc2() cấp phát mảng 2 chiều kiểu word với tham số vào: Kiểm tra các giá trị cột và dòng nhập vào phải lớn hơn 0 sau đó tính kích thước mảng = số dòng * số cột. Nạp kích thước mảng cần cấp phát ($a1) và kích thước từng phần tử mảng ($a2) vào hàm malloc2.
Hàm malloc2 thực hiện việc lấy ra địa chỉ đầu tiên của vùng nhớ còn trống, tính địa chỉ vùng nhớ còn trống mới đưa lại vào Sys_TheTopOfFree
- Hàm getArray[i][j] và setArray[i][j] để lấy/thiết lập giá trị cho phần tử ở dòng i, cột j của mảng: Nhập giá trị dòng (i) và cột (j) muốn truy cập đến (2 giá trị này phải nhỏ hơn giá trị khởi tạo hàng và cột ở phía trên), chương trình sẽ yêu cầu nhập đến khi đúng giá trị dòng cột tương ứng
Với hàm getArray dữ liệu đầu vào sẽ là 1 word (4 byte) chương trình sẽ tự động ngắt khi người dung nhập đủ 4 kí tự vào store vào địa chỉ Arrayp[i][j]
Hàm setArray sẽ truy cập đến địa chỉ của Array[i][j] và dung lệnh lw để lấy giá trị thanh ghi ra ngoài

**Ý nghĩa các thanh ghi được sử dụng:*
- $a0: Lưu địa chỉ các con trỏ
- $s2: Số cột của mảng 2 chiều (j)
- $s3: Số dòng của mảng 2 chiều (i)
- $s4: Số thanh ghi cần cấp phát cho con trỏ Copy
- $s5: Lưu số thanh ghi cần để cấp phát con trỏ kiểu Char

- $s6: Lưu số thanh ghi cần để cấp phát con trỏ kiểu Byte
- $s7: Lưu số thanh ghi cần để cấp phát con trỏ kiểu Word

III. Mã nguồn

```
#================================ Computer Architecture
============================
#     @Author   :      Pham Quang Ha
#     @StudentID: 20194546
#     @Language :      Assembly, MIPS
#=============================================================================
==========
.data
CharPtr:          .word  0 # Bien con tro, tro toi kieu asciiz
BytePtr:          .word  0 # Bien con tro, tro toi kieu Byte
WordPtr:    .word  0 # Bien con tro, tro toi mang kieu Word
CharPtrCopy:      .word  0 # Bien con tro, tro toi mang Copy kieu Word
WordPtr2:   .word  0 # Bien con tro, tro toi mang 2 chieu kieu Word
Enter:            .word  1 # Buffer store word
ValueResult:    .asciiz "\nGia tri con tro la: "
AddressResult: .asciiz "\nDia chi con tro la: "
MemoryResult:  .asciiz "\nLuong bo nho da cap phat la: "
RowMessage:     .asciiz "Moi ban nhap so dong"
ColMessage:     .asciiz "Moi ban nhap so cot"
Notification:  .asciiz "Phuong thuc GetArray va SetArray !!!"
Message:        .asciiz "Nhap 0 de thiet lap gia tri\nNhap 1 de lay gia
tri\nNhap 2 de thoat ham"
EnterMessage:  .asciiz "\nMoi ban nhap 1 word (toi da 4 byte): "
.kdata
# Bien chua dia chi dau tien cua vung nho con trong
Sys_TheTopOfFree: .word  1
# Vung khong gian tu do, dung de cap bo nho cho cac bien con tro
Sys_MyFreeSpace:
.text
      #Khoi taovung nho cap phat dong
      jal  SysInitMem
      #-----------------------
      #  Cap phat cho bien con tro, gom 3 phan tu, moi phan tu 1 byte
      #  $s5: Luu so thanh ghi can de cap phat CharPtr
      #-----------------------
      la   $a0, CharPtr
      addi $a1, $zero, 3
      addi $a2, $zero, 1
      jal  malloc
      add  $s5, $0, $t1
      #-----------------------
      #  Cap phat cho bien con tro, gom 6 phan tu, moi phan tu 1 byte
      #  $s6: Luu so thanh ghi can de cap phat BytePtr
      #-----------------------
      la   $a0, BytePtr
      addi $a1, $zero, 6
      addi $a2, $zero, 1
      jal  malloc
      add  $s6, $0, $t1
      #-----------------------
```

```
#  Cap phat cho bien con tro, gom 5 phan tu, moi phan tu 4 byte
#  $s7: Luu so thanh ghi can de cap phat WordPtr
#-----------------------
la   $a0, WordPtr
addi $a1, $zero, 5
addi $a2, $zero, 4
jal  malloc
add  $s7, $0, $t1
#-----------------------
#  Gan gia tri cho con tro
#-----------------------
la   $a0, CharPtr
lw   $t1, 0($a0)
addi $t0, $0, 'a'
sb   $t0, 0($t1)

la   $a0, BytePtr
lw   $t1, 0($a0)
addi $t0, $0, -34
sb   $t0, 0($t1)

la   $a0, WordPtr
lw   $t1, 0($a0)
addi $t0, $0, 0xab34
sw   $t0, 0($t1)

#-----------------------
#  Lay gia tri va dia chi cua Word/Byte
#  $a0: Dia chi con tro
#  $t1: Dia chi vung nho duoc cap phat ( dia chi con tro )
#  $t0: Gia tri bien con tro
#-----------------------
la   $a0, CharPtr
lw   $t1, 0($a0)
lb   $t0, 0($t1)
jal  getValue
nop
jal  getAddress

la   $a0, BytePtr
lw   $t1, 0($a0)
lb   $t0, 0($t1)
jal  getValue
nop
jal  getAddress

la   $a0, WordPtr
lw   $t1, 0($a0)       # Lay ra dia chi cua con tro
lw   $t0, 0($t1)       # Lay ra gia tri word tai dia chi con tro
jal  getValue_word
nop
jal  getAddress

#-----------------------
```

```
      # Copy con tro xau ki tu (CharPtr)
      #-----------------------
      la   $a0, CharPtrCopy
      la   $a1, CharPtr
      add  $s4, $0, $s5      # Lay kich thuoc mang copy
      jal  CopyChar

      # Kiem tra ket qua Copy
      la   $a0, CharPtrCopy
      lw   $t1, 0($a0)
      lb   $t0, 0($t1)
      jal  getValue
      nop
      jal  getAddress
      nop


      #-----------------------
      #  Tinh gia tri bo nho da cap phat
      #-----------------------
      jal  memoryAllocated
      nop

      #-----------------------
      #  Cap phat mang 2 chieu kieu .word voi tham so dau vao
      #  $s3 = i: so dong
      #  $s2 = j: so cot
      #-----------------------
SetRow:    li   $v0, 51
      la   $a0, RowMessage
      syscall
      slt  $s3, $a0, $0      # Kiem tra gia tri nhap vao phai lon hon 0
      bne  $s3, $0, SetRow
      add  $s3, $0, $a0

SetCol:    li   $v0, 51
      la   $a0, ColMessage
      syscall
      slt  $s2, $a0, $0      # Kiem tra gia tri nhap vao phai lon hon 0
      bne  $s2, $0, SetCol
      add  $s2, $0, $a0

      la   $a0, WordPtr2           # Load dia chi cua mang 2 chieu
      mul  $a1, $s3, $s2           # Kich thuoc mang 2 chieu
      addi $a2, $zero, 4           # Kich thuoc phan tu mang
      jal  malloc2
      nop

      #-----------------------
      #  SetArray[i][j]  && GetArray[i][j]
      #  $t0 = i: so dong < $s3
      #  $t1 = j: so cot  < $s2
      #-----------------------
Notifi:    li   $v0, 55
```

```
      la   $a0, Notification # In thong bao chuyen sang ham Get va Set
      li   $a1, 1
      syscall
EnterRow:  li   $v0, 51
      la   $a0, RowMessage
      syscall
      sle  $t0, $a0, $s3          # Kiem tra so dong nhap vao nho hon so
dong cap phat
      beq  $t0, $0, EnterRow
      beq  $a0, $0, EnterRow # So dong nhap vao lá»›n hon 0
      add  $t0, $0, $a0

EnterCol:  li   $v0, 51
      la   $a0, ColMessage
      syscall
      sle  $t1, $a0, $s2          # Kiem tra so cot nhap vao nho hon so
cot cap nhat
      beq  $t1, $0, EnterRow
      beq  $a0, $0, EnterRow # So dong nhap vao lá»›n hÆ¡n 0
      add  $t1, $0, $a0
      addi $t2, $0, 4        # Kich thuoc moi phan tu 4 byte

      li   $v0, 51           # Lua chon Get hoac Set hoac thoat chuong
trinh
      la   $a0, Message
      syscall
      add  $a1, $0, $a0
      la   $a0,  WordPtr2
      beq  $a1, $0, SetArray
      nop
      beq  $a1, 1,  GetArray
      nop


      #-----------------------
      #  Tinh gia tri bo nho da cap phat
      #-----------------------
      jal  memoryAllocated
      nop
lock:      j lock
      nop
# ================================== END MAIN
==================================
#----------------------------------------
#  Ham khoi tao cho viec cap phat dong
#  @param    khong co
#  @detail   Danh dau vi tri bat dau cua vung nho co the cap phat duoc
#----------------------------------------
SysInitMem:
      la   $t9, Sys_TheTopOfFree  # Lay con tro chua dau tien con trong,
khoi tao
      la   $t7, Sys_MyFreeSpace   # Lay dia chi dau tien con trong, khoi
tao
      sw   $t7, 0($t9)         # Luu lai
```

```
      jr   $ra


#-----------------------------------------
#  Ham cap phat bo nho dong cho cac bien con tro
#  @param  [in/out]   $a0   Chua dia chi cua bien con tro can cap phat
#                           Khi ham ket thuc, dia chi vung nho duoc cap
phat se luu tru vao bien con tro
#  @param  [in]       $a1   So phan tu can cap phat
#  @param  [in]       $a2   Kich thuoc 1 phan tu, tinh theo byte
#  @return            $v0   Dia chi vung nho duoc cap phat
#-----------------------------------------
malloc:    la   $t9, Sys_TheTopOfFree     # Lay con tro chua dau tien con
trong, khoi tao
           lw   $t8, 0($t9)       # Lay dia chi dau tien con trong
           sw   $t8, 0($a0)             # Cat dia chi do vao bien con tro
           addi $v0, $t8, 0             # Dong thoi la ket qua tra ve cua
ham
           mul  $t7, $a1, $a2           # Tinh kich thuoc cua mang can cap
phat = so phan tu * kich thuoc phan tu

           addi $t0, $0, 4        # Luu kich thuoc kieu word de tinh so
luong thanh ghi can cap
           div  $t7, $t0          # So luong thanh ghi can cap phat
           mflo $t1               # $lo luu thuong
           mfhi $t2               # $hi luu so du
           beq  $t2, $0, allocation
           addi $t1, $t1, 1       # Cap them 1 thanh ghi de luu
# Cap phat bo nho chia het 4 cho cac bien con tro
allocation:
           mul  $t7, $t1, $t0           # Kich thuoc mang can cap phat
           add  $t6, $t8, $t7           # Tinh dia chi dau tien con trong
           sw   $t6, 0($t9)             # Luu tro lai dia chi dau tien do
vao bien Sys_TheTopOfFree
           jr   $ra

#-----------------------------------------
#  Ham lay gia tri cua bien con tro (*CharPtr, *BytePtr, *WordPtr)
#  getValue:          dung de tra gia tri cua *CharPtr, *BytePtr
#  getValue_word:     dung de tra gia tri cua *WordPtr
#-----------------------------------------
getValue:
      li   $v0, 4      # In message
        la   $a0, ValueResult
           syscall
           li   $v0, 1
           add  $a0, $0, $t0
        syscall
        jr   $ra

getValue_word:
      li   $v0, 4      # In message
        la   $a0, ValueResult
        syscall
```

```
        li   $v0, 34
        add  $a0, $0, $t0
        syscall
        jr   $ra

#-----------------------------------------
#  Ham lay dia chi cua bien con tro (&CharPtr, &BytePtr, &WordPtr)
#  @param [in]   $t1: Dia chi vung nho duoc cap phat ( dia chi con tro )
#-----------------------------------------
getAddress: li   $v0, 4      # In message
            la   $a0, AddressResult
            syscall
            li   $v0, 34           # In dia chi con tro
            add  $a0, $0, $t1
            syscall
            jr   $ra


#-----------------------------------------
#   Ham copy con tro xau ki tu
#   @param  [in/out]   $a0   Chua dia chi cua bien con tro can cap phat
#                            Khi ham ket thuc, dia chi vung nho duoc cap
phat se luu tru vao bien con tro
#           $a1   Chua dia chi cua bien con tro muon sao chep
#   @param  [in]        $s4   So thanh ghi can cap phat
#   return  $v0   Dia chi vung nho duoc cap phat
#-----------------------------------------
CopyChar:  la   $t9, Sys_TheTopOfFree       # Lay con tro chua dau tien
con trong, khoi tao
            lw   $t8, 0($t9)       # Lay dia chi dau tien con trong
            sw   $t8, 0($a0)             # Cat dia chi do vao bien con tro
            add  $v0, $0, $t8
            mul  $t7, $s4, 4       # Kich thuoc bo nho can cap phat
            add  $t6, $t8, $t7           # Tinh dia chi dau tien con trong
            sw   $t6, 0($t9)       # Luu tro lai dia chi con trong do vao
Sys_TheTopOfFree
            add  $t5, $0, $s4      # Luu so thanh ghi can load de copy

Copy: lw   $t1, 0($a1)       # Load dia chi cua con tro can Copy
      lw   $t4, 0($t1)       # Load du lieu cua con tro can Copy
      lw   $t0, 0($a0)       # Load dia chi con tro Copy
      sw   $t4, 0($t0)       # Store du lieu vao con tro Copy
      addi $a1, $a1, 4       # Tang dia chi con tro can Copy
      addi $a0, $a0, 4       # Tang dia chi con tro Copy
      addi $t5, $t5, -1      # Giam so luong thanh ghi can Copy
      bne  $t5, $0, Copy           # Neu so thanh ghi can load khac 0 thi
tiep tuc Copy
      nop
      jr   $ra
                        # Copy done
#-----------------------------------------
#   Ham lay luong bo nho da cap phat
#   Method : Lay Dia chi o Sys_TheTopOfFree - Sys_MyFreeSpace
#-----------------------------------------
```

```
memoryAllocated:
      li   $v0, 4       # In message
            la   $a0, MemoryResult
            syscall

            li   $v0, 1       # Tinh dung luong bo nho
      la   $a1, Sys_TheTopOfFree
      lw   $a1, 0($a1)
      la   $a2, Sys_MyFreeSpace
      sub  $a0, $a1, $a2
      syscall
      jr   $ra


#-----------------------------------------
#  Ham cap phat bo nho dong cho mang 2 chieu kieu word
#  @param  [in/out]   $a0   Chua dia chi cua bien con tro can cap phat
#                           Khi ham ket thuc, dia chi vung nho duoc cap
phat se luu tru vao bien con tro
#  @param  [in]       $a1   So phan tu can cap phat
#  @param  [in]       $a2   Kich thuoc 1 phan tu, tinh theo byte
#-----------------------------------------
malloc2:   la   $t9, Sys_TheTopOfFree       # Lay con tro chua dau tien
con trong, khoi tao
            lw   $t8, 0($t9)       # Lay dia chi dau tien con trong
            sw   $t8, 0($a0)             # Cat dia chi do vao bien con tro
            mul  $t7, $a1, $a2           # Kich thuoc mang can cap phat
      add  $t8, $t8, $t7           # Tinh dia chi dau tien con trong
      sw   $t8, 0($t9)       # Luu tro lai dia chi dau tien do vao bien
Sys_TheTopOfFree
      jr   $ra


#-----------------------------------------
#  Ham thiet lap gia tri cho phan tu o dong i cot j cua mang 2 chieu kieu
word
#  @param  [in]        $a0   Dia chi mang 2 chieu
#  @param  [in]       $s2   So cot cua mang 2 chieu
#  @param  [in]       $t0   Vi tri hang cua mang 2 chieu
#  @param  [in]       $t1   Vi tri cot cua mang 2 chieu
#  @param  [in]       $t2   Kich thuoc 1 phan tu word
#  Khoang cua phan tu = (hang - 1)*Socot + cot
#-----------------------------------------
SetArray:  lw   $t3, 0($a0)
      addi $t0, $t0, -1      # Hang - 1
      mul  $t0, $t0, $s2           # (Hang-1)*SoCot
      addi $t1, $t1, -1      # cot-1
      add  $t0, $t0, $t1           # (Hang-1)*SoCot + cot-1
      mul  $t0, $t0, $t2           # Khoang cach tu con tro dau tien den
phan tu can thay doi
                        # 4 * ((Hang-1)*SoCot + cot-1)
      add  $t3, $t3, $t0           # Dia chi cua phan tu
      li   $v0, 4      # In message bat dau nhap
            la   $a0, EnterMessage
```

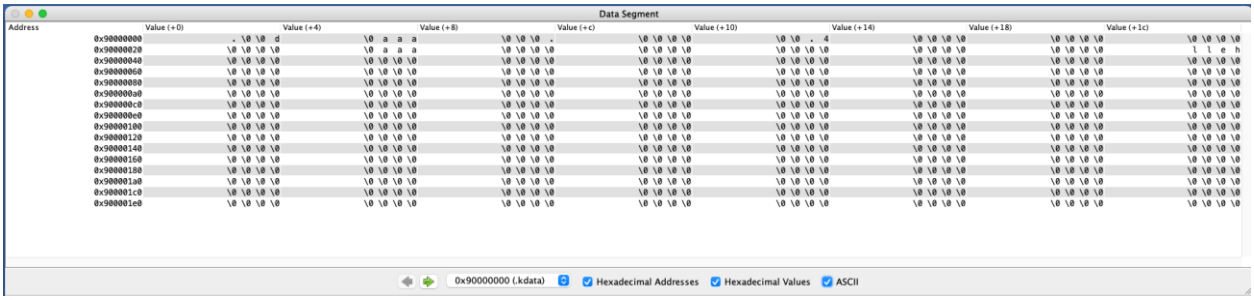
```
            syscall

      add  $s0, $0, $0       # i = 0
      la   $s1, Enter
ReadChar:
      li   $v0, 12           # Read char
      syscall
CheckChar:
      beq  $v0, 10, Return   # Kiem tra ky tu enter
      add  $t4, $s1, $s0           # $t4 = dia chi cua string[i] nhap vao
      sb   $v0, 0($t4)       # dua ki tu vao string nhap
      add  $t5, $t3, $s0           # $t5 = dia chi cua WordPtr2[i][j][k] k
< 4
      sb   $v0, 0($t5)
      addi $s0, $s0, 1       # i = i + 1
      slti $t5, $s0, 4       # if i < 4
      beq  $t5, $0, Return
      nop
      j    ReadChar
      nop

Return:    j    EnterRow

#-----------------------------------------
#  Ham lay gia tri cho phan tu o dong i cot j cua mang 2 chieu kieu word
#  @param  [in]        $a0   Dia chi mang 2 chieu
#                      $s2   So cot cua mang 2 chieu
#                $t0   Vi tri hang cua mang 2 chieu
#           $t1   Vi tri cot cua mang 2 chieu
#           $t2   Kich thuoc 1 phan tu word
#  Khoang cua phan tu = (hang - 1)*Socot + cotword-1
#-----------------------------------------
GetArray:  lw   $t3, 0($a0)             # Lay dia chi phan tu dau tien cua
mang
      addi $t0, $t0, -1      # Hang - 1
      mul  $t0, $t0, $s2           # (Hang-1)*SoCot
      addi $t1, $t1, -1      # Cot-1
      add  $t0, $t0, $t1           # (Hang-1)*SoCot + cot-1
      mul  $t0, $t0, $t2           # Khoang cach tu con tro dau tien den
phan tu can thay doi
                        # 4 * ((Hang-1)*SoCot + cot-1 )
      add  $t3, $t3, $t0           # Dia chi cua phan tu

      li   $v0, 4      # In message
            la   $a0, ValueResult
            syscall

      li   $v0, 34           # In gia tri ra
      lw   $a0, 0($t3)       # Lay ra gia tri cua phan tu tai hang i, cot
j
      syscall

      j EnterRow
```

```
#============================================== END FILE
=========================================
```

IV. Kiểm thử chương trình

```
Gia tri con tro la: 97
Dia chi con tro la: 0x90000004
Gia tri con tro la: -34
Dia chi con tro la: 0x90000008
Gia tri con tro la: 0x0000ab34
Dia chi con tro la: 0x90000010
Gia tri con tro la: 97
Dia chi con tro la: 0x90000024
Luong bo nho da cap phat la: 36
```


```
Moi ban nhap 1 word (toi da 4 byte): hell
```



```
Gia tri con tro la: 97
Dia chi con tro la: 0x90000004
Gia tri con tro la: -34
Dia chi con tro la: 0x90000008
Gia tri con tro la: 0x0000ab34
Dia chi con tro la: 0x90000010
Gia tri con tro la: 97
Dia chi con tro la: 0x90000024
Luong bo nho da cap phat la: 36
Moi ban nhap 1 word (toi da 4 byte): hell
Gia tri con tro la: 0x6c6c6568
Luong bo nho da cap phat la: 96
```

```
Value (+1c)
                    0x00000000
                    0x6c6c6568
```

⇨ Như vậy, chương trình hoạt động đúng chức năng

BÀI 10

Sinh viên thực hiện: Nguyễn Tiến Nhật Minh – 20194621

I. Đề bài

Sử dụng 2 ngoại vi là bàn phím keypad và led 7 thanh để xây dựng một máy tính bỏ túi đơn giản. Hỗ trợ các phép toán +, -, *, /,% với các toán hạng là số nguyên. 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 hiển thị 2 số cuối cùng. Ví dụ khi nhấn phím 1 → hiển 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 Calculator 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

II. Phân tích xây dựng thuật toán

-Để xây dựng một chương trình máy tính bỏ túi đơn giản, ta chia ra thành 5 trạng thái như sau:

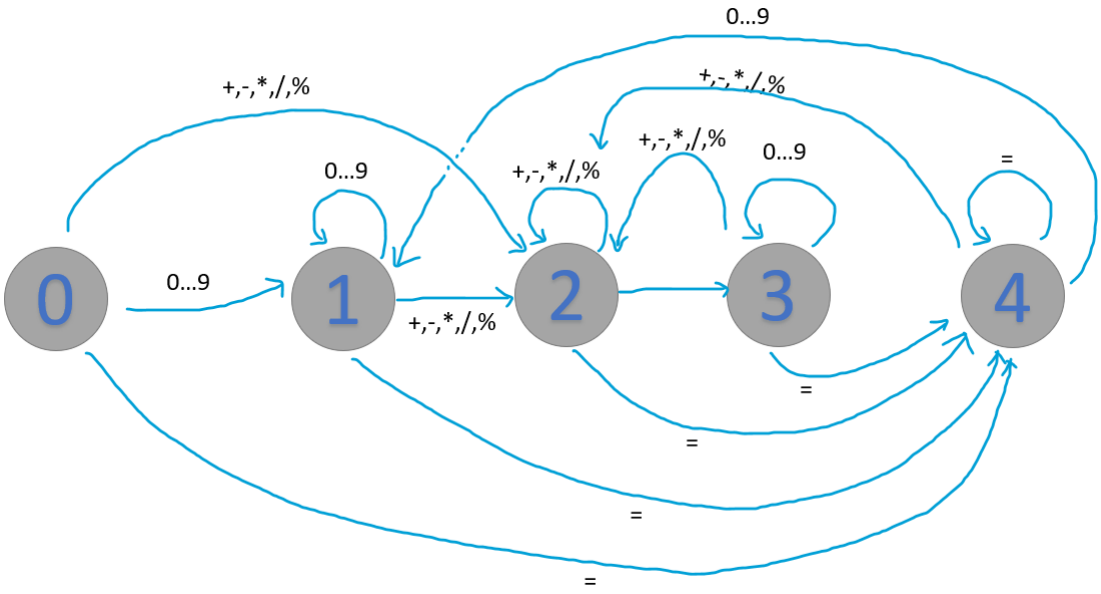
0)Trạng thái khởi tạo: các giá trị được reset và bằng 0

1)Trạng thái nhập số thứ nhất: nhập số thứ nhất và trên hiển thị led

2)Trạng thái chờ số thứ hai: lưu lại số thứ nhất và chờ đợi sự kiện nhập số thứ hai

3)Trạng thái nhập số thứ hai: nhập số thứ hai và hiển thị trên led

4)Trạng thái tính toán và hiện thị kết quả: lưu số vừa nhập, đồng thời tính toán và hiện thị kết quả



Sơ đồ trạng thái

*Chuẩn bị:

+)Các địa chỉ cổng thao tác tới hai thiết bị ngoại vi: **màn hình led** và **key pad**

```
.eqv SEVENSEG_LEFT 0xFFFF0010 # Dia chi cua den led 7 doan trai.
.eqv SEVENSEG_RIGHT 0xFFFF0011 # Dia chi cua den led 7 doan phai


.eqv IN_ADDRESS_HEXA_KEYBOARD 0xFFFF0012
.eqv OUT_ADDRESS_HEXA_KEYBOARD 0xFFFF0014
```

+)Các thanh ghi:

- **$s0 :** Lưu giá trị toán hạng thứ nhất
- **$s1 :** Lưu giá trị toán hạng thứ hai
- **$s2 :** Thông tin trạng thái hoạt động (0-4: 5 trạng thái mô tả ở trên)
- **$s3 :** 0 – Chế độ tính toán bình thường / 1 – Chế độ tính toán liên tiếp
- **$s4 :** Trạng thái ngoại lệ
  # 0 - No exception, 1 - overflow, 2 - divide/mod for 0
- **$s5 :** Thông tin về toán tử đang được thực hiện

  # 1-plus, 2-minus, 3-mult, 4-div, 5-mod

+) Các hàm tiện ích

-Hàm hiển thị led 7 thanh
<-Input: Một số nguyên từ 0 – 9 hoặc dấu trừ
->Output: Hiển thị trên bộ led 7 thanh
-Hàm hiển thị 2 số cuối của một số nguyên
<-Đầu vào: Một số nguyên
->Đầu ra: Hiển thị 2 số cuối trên bộ led 7 thanh
++) Trường hợp số dương: Chia số đó cho 100, lấy phần dư chính là hai chữ số cuối. Sau đó lại đem chia hai số cuối cho 10, thương và phần dư lần lượt là chữ số hàng chục và hàng đơn vị. Sau đó gọi tới **Hàm hiển thị led 7 thanh** để hiển thị từng số một.
++) Trường hợp số âm: Lấy nghịch đảo dấu xong làm tương tự như số dương. Trong trường hợp chữ số hàng chục bằng 0 thay bằng dấu trừ.
-Hàm tính toán
<-Đầu vào: Hai số nguyên và toán tử
->Đầu ra: Kết quả
+) Thuật toán
-Chương trình chính gồm một hàm loop vô tận, mục tiêu là liên tục lắng nghe các tín hiệu *interrupt* đến từ thiết bị ngoại vi

```
loop: nop
sleep:  addi $v0,$zero,32
        li $a0,300 # sleep 300 ms
        syscall
        nop # WARNING: nop is mandatory here.
        b loop # Loop
end_main:
```

-Mỗi khi thực hiện ấn một phím trên keypad, CPU sẽ thực hiện các chỉ dẫn được lưu trong vùng không gian **.ktext 0x80000180**
-Đầu tiên, CPU sẽ kiểm tra phím nào được ấn thông qua việc kiểm tra từng hàng của ma trận phím và lấy ra giá trị bấm được qua địa chỉ **0xFFFF0014** và lưu vào thanh ghi **$t0**
-Sau đó, kiểm tra giá trị của **$t0** vừa nhập, có thể là một số, một toán tử, hoặc dấu bằng.
-Tùy vào giá trị của **$t0** và trạng thái hiện tại được lưu trong **$s2**, chương trình sẽ chuyển tới trạng thái kế tiếp và thực hiện các tác vụ của mình (như thể hiện ở sơ đồ trạng thái phía trên).
****Lưu ý:** Ở trên có đề cập tới việc thực hiện liên tiếp các phép tính, vì vậy cần chú ý lưu thông tin của toán tử ở pha trước để tránh việc bị ghi đè ở toán tử hiện tại. Khi đó $s3 sẽ được kích hoạt và có giá trị bằng 1. Khi dấu "=" được đọc từ keypad, có nghĩa đây không phải là chế độ tính toán liên tiếp nữa, và $s3 trở về trạng thái bình thường (bằng 0).

III. Mã nguồn

```
.eqv SEVENSEG_LEFT 0xFFFF0010 # Dia chi cua den led 7 doan trai.
.eqv SEVENSEG_RIGHT 0xFFFF0011 # Dia chi cua den led 7 doan phai
.eqv IN_ADDRESS_HEXA_KEYBOARD 0xFFFF0012
.eqv OUT_ADDRESS_HEXA_KEYBOARD 0xFFFF0014
```

```
.data
Message: .asciiz "\nJump to state "
Message2: .asciiz "\nKet qua tinh toan =  "
Message3: .asciiz "\nKey scan code "
Overflow_Error: .asciiz "Vuot qua gioi han nhap cho phep : Stack OverFlow"
Logic_Error: .asciiz "Khong the chia mot so cho 0"
.text
main:
 # Enable interrupts you expect
 #------------------------------------------------------
 # Enable the interrupt of Keyboard matrix 4x4 of Digital Lab Sim
      li $k0, IN_ADDRESS_HEXA_KEYBOARD
      li $k1, OUT_ADDRESS_HEXA_KEYBOARD
      li $t9, 0x80 # bit 7 = 1 to enable
      sb $t9, 0($k0)

      li $s0, 0 #Luu gia tri cua toan hang thu nhat
      li $s1, 0 #Luu gia tri cua toan hang thu hai
      li $s2, 0 #Trang thai hoat dong
      # 0- Trang thai khoi tao
      # 1- Trang thai nhap so thu nhat
      # 2- Trang thai cho so thu hai
      # 3- Trang thai nhap so thu hai
      # 4- Trang thai tinh toan va hien thi ket qua
      li $s3, 0 # 0- Normal Calculation Mode, 1 - Continuous Calculation
Mode
      li $s4, 0 # Exception Code
      # 0 - No exception, 1 - overflow, 2 - divide/mod for 0
      li $s5, 0
      #s5 is the operator state
      # 1-plus, 2-minus, 3-mult, 4-div, 5-mod the rest for null


loop: nop
sleep:     addi $v0,$zero,32
      li $a0,300 # sleep 300 ms
      syscall
      nop # WARNING: nop is mandatory here.
      b loop # Loop
end_main:
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# GENERAL INTERRUPT SERVED ROUTINE for all interrupts
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.ktext 0x80000180
 #---------------------------------------------------------
 # SAVE the current REG FILE to stack
 #---------------------------------------------------------
IntSR1:  addi $sp,$sp,4 # Save $ra because we may change it later
       sw $ra,0($sp)
       addi $sp,$sp,4 # Save $at because we may change it later
       sw $at,0($sp)
       addi $sp,$sp,4 # Save $t1 because we may change it later
       sw $k0,0($sp)
 #---------------------------------------------------------
```

```
 # Processing
 #-----------------------------------------------------------
prn_msg:addi $v0, $zero, 4
       la $a0 , Message3
      syscall
polling1 :
      li $t9 , 0x81 # check row 4 with key 0, 1, 2, 3
      sb $t9 , 0( $k0 ) # must reassign expected row
      lb $a0 , 0( $k1 ) # read scan code of key button
      beq $a0 , 0x0 , polling2
      jal pooling_keypad
      jal Sleep
      j prn_cod # continue pooling
polling2:
      li $t9  , 0x82 # check row 4 with key 4, 5, 6, 7
      sb $t9 , 0( $k0 ) # must reassign expected row
      lb $a0 , 0( $k1 ) # read scan code of key button
      beq $a0 , 0x0 , polling3
      jal pooling_keypad
      jal Sleep
      j prn_cod # continue pooling
polling3:
      li $t9 , 0x84 # check row 4 with key 8, 9, A, B
      sb $t9 , 0( $k0 ) # must reassign expected row
      lb $a0 , 0( $k1 ) # read scan code of key button
      beq $a0 , 0x0 , polling4
      jal pooling_keypad
      jal Sleep
      j prn_cod # continue pooling
polling4:
      li $t9 , 0x88 # check row 4 with key C, D, E, F
      sb $t9 , 0( $k0 ) # must reassign expected row
      lb $a0 , 0( $k1 ) # read scan code of key button
      jal pooling_keypad
      jal Sleep
prn_cod:
      li $v0 , 11
      li $a0 , '\n'
      syscall
Input_Check:
      blt $t0, 48, Operator_Processor #if < 0
      nop
      bgt $t0, 57, Operator_Processor #if > 9
      j Operand_Processor
      nop
Operand_Processor:
Check_state:
is_State0: bne $s2, 0 , is_State1
         jal Change_state1
         j keep_on
is_State1: bne $s2, 1 , is_State2
          j keep_on
is_State2: bne $s2, 2 , is_State3
          jal Change_state3
```

```
          j keep_on
is_State3: bne $s2, 3 , is_State4
          j keep_on
is_State4: jal Change_state1
          j keep_on
keep_on:
      addi $t0, $t0, -48# change input key
      mul $t3, $t3, 10
      add $t3, $t3, $t0
      addi $t0, $t0, 48

      addi $a2, $t3, 0
      blt $t3, 0, Change_error1 #overflow
      jal  display_last_two_number
      j End_Interupt

Operator_Processor:
If_plus:
      bne $t0, 'a', If_minus
      li $s5, 1
      li $t0, '+'
      j After_Operator
If_minus:
      bne $t0, 'b', If_mult
      li $s5, 2
      li $t0, '-'
      j After_Operator
If_mult:
      bne $t0, 'c', If_div
      li $s5, 3
      li $t0, '*'
      j After_Operator
If_div:
      bne $t0, 'd', If_mod
      li $s5, 4
      li $t0, '/'
      j After_Operator
If_mod:
      bne $t0, 'e', If_equal
      li $s5, 5
      li $t0, '%'
      j After_Operator
If_equal:
      bne $t0, 'f', loop
      li $t0, '='
      j After_Equal
After_Operator:
Is_state1:  bne $s2, 1, Is_state2
            addi $s0, $t3, 0     #save the first operand in $s0
            li $t3, 0            #reset t3
            jal Change_state2
            j End_Interupt
Is_state2:  bne $s2, 2, Is_state3
            j End_Interupt
```

```
Is_state3:     bne $s2, 3, Is_state4
            addi $s1, $t3, 0     #save the second operand in $s1
            li $t3,0              #reset $t3
            li $s3, 1 # Continuous Calculation Mode : Activate
            jal Change_state4
            j go_on
Is_state4:      jal Change_state2
            j End_Interupt

After_Equal:
is_state1: bne $s2, 1, is_state2
      addi $s0, $t3, 0     #save the first operand in $s1
      li $t3, 0  #reset $t3
      jal Change_state4
      j go_on
is_state2:bne $s2, 2, is_state3
      jal Change_state4
      j go_on
is_state3:bne $s2, 3, is_state4
      addi $s1, $t3, 0     #save the second operand in $s1
      li $t3, 0  #reset $t3
      li $s3, 0 # Continuous Calculation Mode : Terminate
      jal Change_state4
      j go_on
is_state4:
      jal Change_state4
      j go_on
go_on:
      addi $a1, $s0, 0
      addi $a2, $s1, 0
      beq $s3, 1, continuous_calculate
normal_calculate:
      jal Calculator
      j finish_calculate
continuous_calculate:
      addi $t9, $t8, 0   #swap to previous oparetor
      addi $t8, $s5, 0
      addi $s5, $t9, 0
      jal Calculator
      addi $t9, $t8, 0   #swap again
      addi $t8, $s5, 0
      addi $s5, $t9, 0
      jal Change_state2
finish_calculate:
      addi $s0, $v0, 0 #Save the result to the first operand
      addi $a2, $s0, 0
      jal  display_last_two_number
      j End_Interupt

End_Interupt:
      addi $t8, $s5, 0 #luu trang thai cuar toan tu truoc do trong truong
hop tinh toan lien tiep
      nop
next_pc:mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc
```

```
      addi $at, $at, 4 # $at = $at + 4 (next instruction)
      mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at
 #-----------------------------------------------------
 # RESTORE the REG FILE from STACK
 #-----------------------------------------------------
 restore:
      lw $k0, 0($sp) # Restore the registers from stack
      addi $sp,$sp,-4
      lw $at, 0($sp) # Restore the registers from stack
      addi $sp,$sp,-4
      lw $ra, 0($sp) # Restore the registers from stack
      addi $sp,$sp,-4
return: eret # Return from exception
#----------------------------------------
#Read corresponding data from the keypad
pooling_keypad:
Case_0:    bne $a0, 0x11, Case_1
      li $t0, '0'
      j print_key
Case_1: bne $a0, 0x21, Case_2
      li $t0, '1'
      j print_key
Case_2: bne $a0, 0x41, Case_3
      li $t0, '2'
      j print_key
Case_3: bne $a0, 0xffffff81, Case_4
      li $t0, '3'
      j print_key
Case_4: bne $a0, 0x12, Case_5
      li $t0, '4'
      j print_key
Case_5: bne $a0, 0x22, Case_6
      li $t0, '5'
      j print_key
Case_6: bne $a0, 0x42, Case_7
      li $t0, '6'
      j print_key
Case_7: bne $a0, 0xffffff82, Case_8
      li $t0, '7'
      j print_key
Case_8: bne $a0, 0x14, Case_9
      li $t0, '8'
      j print_key
Case_9: bne $a0, 0x24, Case_a
      li $t0, '9'
      j print_key
Case_a: bne $a0, 0x44, Case_b
      li $t0, 'a'
      j print_key
Case_b: bne $a0, 0xffffff84, Case_c
      li $t0, 'b'
      j print_key
Case_c: bne $a0, 0x18, Case_d
      li $t0, 'c'
```

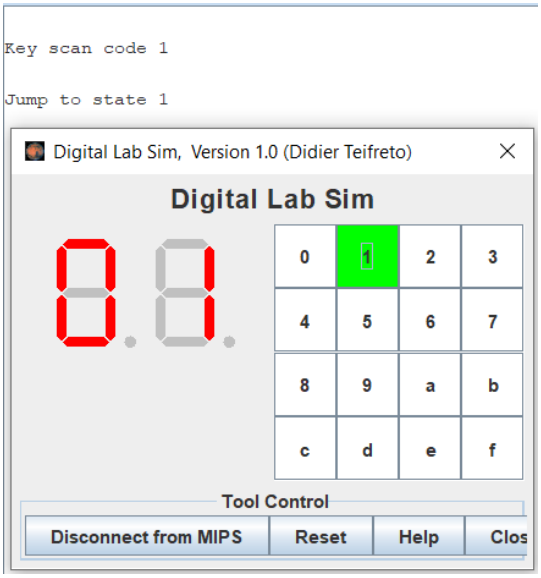
```
      j print_key
Case_d: bne $a0, 0x28, Case_e
      li $t0, 'd'
      j print_key
Case_e: bne $a0, 0x48, Case_f
      li $t0, 'e'
      j print_key
Case_f:
      li $t0, 'f'
      j print_key
print_key:
      addi $at, $v0, 0
      li $v0, 11
      addi $a0 , $t0, 0
      syscall
      addi $v0, $at, 0
      jr $ra
#-----------------------------------------
#-----------------------------------------
#Put thread into a sleep
Sleep:
      addi $at, $v0, 0
      li $a0 , 100 # sleep 100ms
      li $v0 , 32
      syscall
      addi $v0, $at, 0
      jr $ra

#-----------------------------------------
# Ham hien thi so tren bo led 7 thanh
#  @param  [in]   $a1:  Chua mot so
display_number:
case_0: bne $a1, 0, case_1
      li $a0, 0x3F
      j end_switch
case_1: bne $a1, 1, case_2
      li $a0, 0x6
      j end_switch
case_2: bne $a1, 2, case_3
      li $a0, 0x5B
      j end_switch
case_3: bne $a1, 3, case_4
      li $a0, 0x4F
      j end_switch
case_4: bne $a1, 4, case_5
      li $a0, 0x66
      j end_switch
case_5: bne $a1, 5, case_6
      li $a0, 0x6D
      j end_switch
case_6: bne $a1, 6, case_7
      li $a0, 0x7D
      j end_switch
case_7: bne $a1, 7, case_8
```

```
      li $a0, 0x7
      j end_switch
case_8: bne $a1, 8, case_9
      li $a0, 0x7F
      j end_switch
case_9: bne $a1, 9, case_Minus
      li $a0, 0x6F
      j end_switch
case_Minus: li $a0, 0x40
          j end_switch
end_switch:
      jr $ra
#----------------------------------------
# Ham hien thi 2 so cuoi tren bo led 7 thanh
#  @param  [in]   $a2:  Chua mot so
      #trich xuat 2 chu so cuoi
display_last_two_number:
      addi $a3, $a2, 0   #swap
      bge $a2, 0, display
      sub $a2, $zero , $a2
display:li $t4, 100
      div $a2, $t4
      mfhi $t5
      li $t4, 10
      div $t5, $t4
      mfhi $t5   #chu so hang don vi
      mflo $t4   #chu so hang chuc

      addi $a2, $a3, 0  #swap

      bge $a2, 0, display2 #display negative number
      bne $t4, 0, display2
      nop
      li $t4, 10
display2:addi $sp,$sp,-4 #making room return address
      sw $ra,0($sp) #save return address

      addi $a1, $t4, 0
      jal display_number
      li $t6, SEVENSEG_RIGHT # assign port's address
      sb $a0, 0($t6) # assign new value

      addi $a1, $t5, 0
      jal display_number
      li $t7, SEVENSEG_LEFT # assign port's address
      sb $a0, 0($t7) # assign new value
      #----------------------------------------

      lw $ra,0($sp) #restore return address (5)
      addi $sp,$sp,4
      jr $ra

#----------------------------------------
#----------------------------------------
```

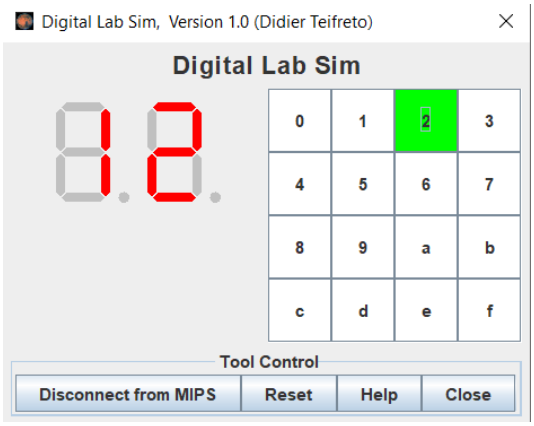
```
# Ham tinh toan
#  @param  [in]  : $a1: Toan hang thu nhat
#  @param  [in]  : $a2: Toan hang thu hai
#  @return_value  : $v0: Ket qua tinh toan
Calculator:      li $v0, 0
case_plus:  bne $s5,1,case_minus
            add $v0, $a1, $a2
blt $a1, 0, print_result
blt $v0, 0, Change_error1
            j print_result
case_minus: bne $s5,2,case_mult
            sub $v0, $a1, $a2
bgt $a1, 0, print_result
bgt $v0, 0, Change_error1
            j print_result
case_mult:  bne $s5,3,case_div
            mul $v0, $a1, $a2
#blt $a1, 0, print_result
#blt $v0, 0, Change_error1
            j print_result
case_div:   bne $s5,4,case_mod
            beq $a2, 0, Change_error2
            div $a1, $a2
            mflo $v0
            j print_result
case_mod:   bne $s5,5,case_null
            beq $a2, 0, Change_error2
            div $a1, $a2
            mfhi $v0
            j print_result
case_null:     jr $ra
print_result:
      addi $v1, $v0, 0
      li $v0, 4
      la $a0, Message2
      syscall

      li $v0, 1
      addi $a0, $v1, 0
      syscall
      addi $v0, $v1, 0

      jr $ra
#-----------------------------------------
#-----------------------------------------
#Changing state
Change_state0: li $s2, 0
            #reset all value
            li $s0, 0
            li $s1, 0
            li $s3, 0
            li $s4, 0
            li $s5, 0
            li $t0, 0
```

```
            li $t3, 0

            j Print_State
Change_state1: li $s2, 1
            j Print_State
Change_state2: li $s2, 2
            j Print_State
Change_state3: li $s2, 3
            j Print_State
Change_state4: li $s2, 4
            j Print_State
Print_State:
      addi $v1, $v0, 0
      li $v0, 4
      la $a0, Message
      syscall

      li $v0, 1
      addi $a0, $s2, 0
      syscall
      addi $v0, $v1, 0

      jr $ra
#-----------------------------------------
#-----------------------------------------
#Execption Handler
Change_error1: li $s4, 1
            j Print_Error
Change_error2:   li $s4, 2
            j Print_Error
Print_Error:
case_err1: bne $s4, 1, case_err2
      addi $v1, $v0, 0
      li $v0, 55
      la $a0, Overflow_Error
      li $a1, 0
      syscall
      addi $v0, $v1, 0
      j Reset_new
case_err2:       bne $s4, 2, Reset_new
      addi $v1, $v0, 0
      li $v0, 55
      la $a0, Logic_Error
      li $a1, 0
      syscall
      addi $v0, $v1, 0

Reset_new:
      jal Change_state0
      addi $a2, $t3, 0
      jal  display_last_two_number
      j End_Interupt
#-----------------------------------------
```

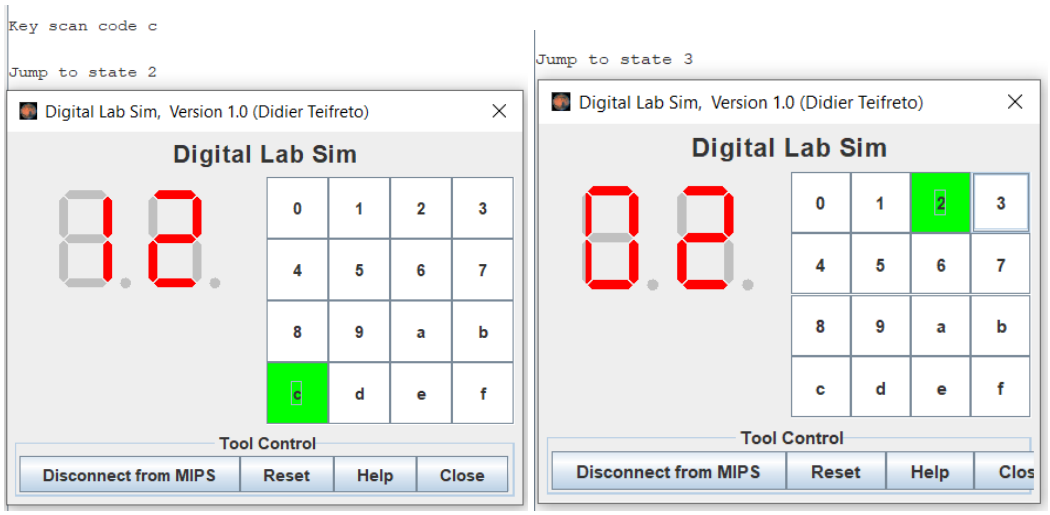
IV. Kiểm thử chương trình



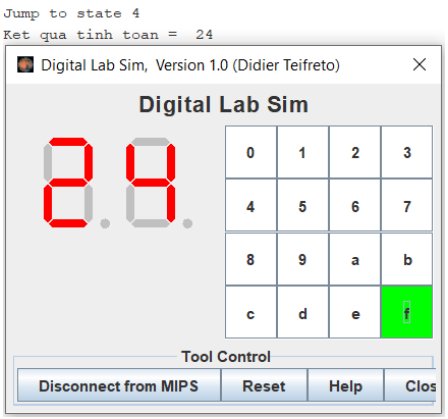
Ấn phím 1, nhảy tới trạng thái 1



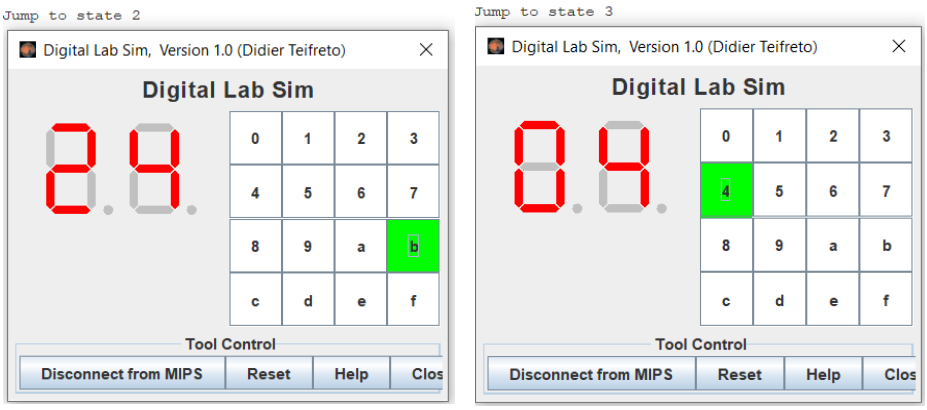
Ấn phím 2, màn hình hiển thị 12



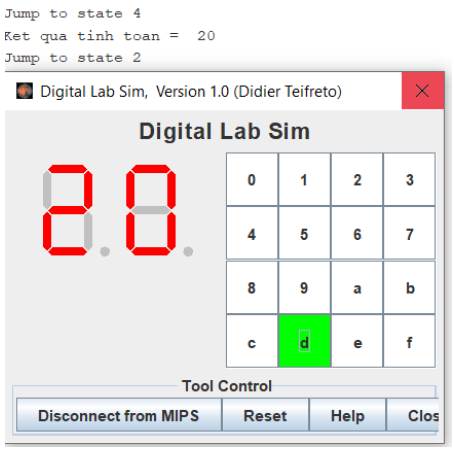
Ấn phím c, thực hiện phép tính nhân, chuyển sang trạng thái 2, xong phím hai, chuyển sang trạng thái tiếp theo



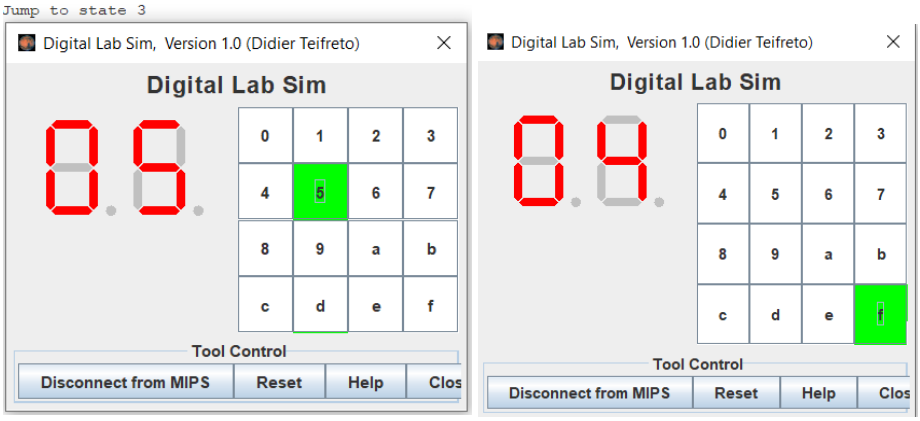
Nhấn phím bằng để hiển thị kết quả 12 x 2 = 24



Tiếp tục nhấn phím b chuẩn bị cho phép trừ, muốn trừ 4 thì nhấn phím 4

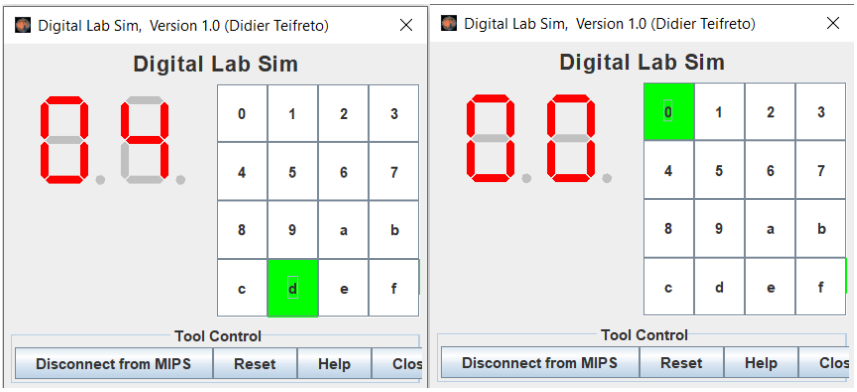


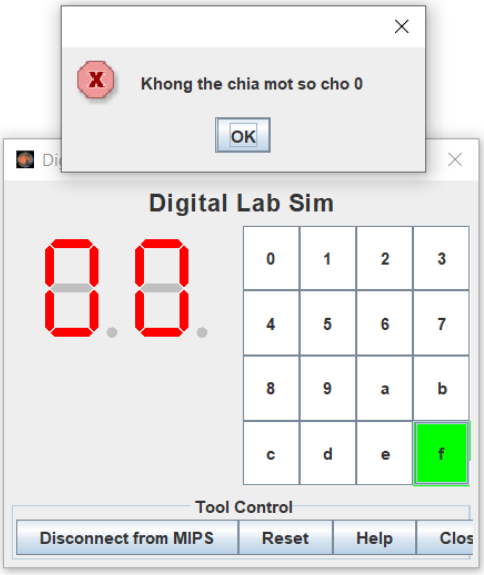
Không ấn '=' mà ấn phím chia, thực hiện phép tính 20-4 trước đó và nhảy tới trạng thái 2



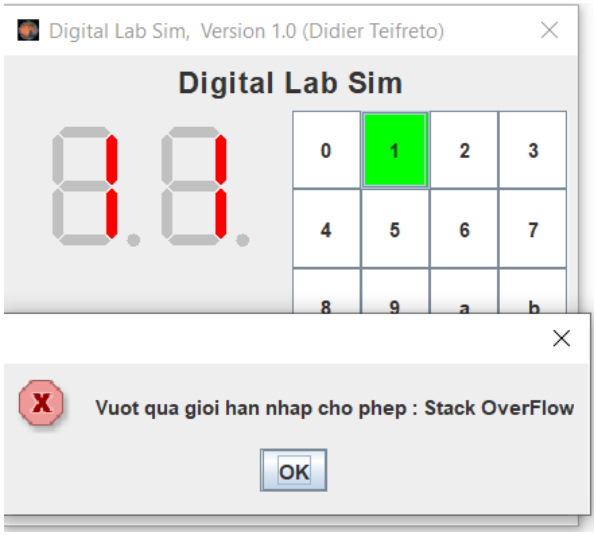
Chia cho 5, kết quả được 4

-Kiểm tra ngoại lệ:





Chia một số cho 0, hiển thị thông báo lỗi



Nhập 11111111111 thì thông báo tràn số