Mục lục

Hướng dẫn thực hiện

Chia nhóm 2 sinh viên thực hiện 2 đề tài ngẫu nhiên gồm 1 đề tài từ 1 đến 8 và 1 đề tài từ 9 đến 18.

Cách thức đánh giá

- Trình bày vấn đáp: chạy mô phỏng RARS, trình bày mã nguồn, trả lời câu hỏi.
- Viết báo cáo in quyển trình bày các vấn đề: phân tích cách làm, thuật toán, mã nguồn và kết quả chạy mô phỏng.
- Nộp chương trình hợp ngữ, bản mềm. Qui định cách đặt tên file hợp ngữ như sau:

gxx_syy.asm

+ xx: số thứ tự của nhóm (2 chữ số)
+ yy: số thứ tự của bài (2 chữ số)

Ví dụ, nhóm 21, làm bài 8 và bài 15, thì cần đặt tên file mã nguồn là g21_s08.asm và g21_s15.asm, file báo cáo đặt tên là g21.pdf

Lưu ý khi thực hiện:

- Chương trình nên có giao diện menu chọn (nếu cần), có thể lặp lại nhiều lần mà không cần chạy lại chương trình.

- Các thao tác nhập dữ liệu nên có kiểm tra, xác thực tính hợp lệ của dữ liệu.
- Mã nguồn nên tổ chức thành chương trình con (nếu cần).
- Viết mã nguồn sáng sủa, dễ đọc, dễ hiểu (có căn lề, chú thích).

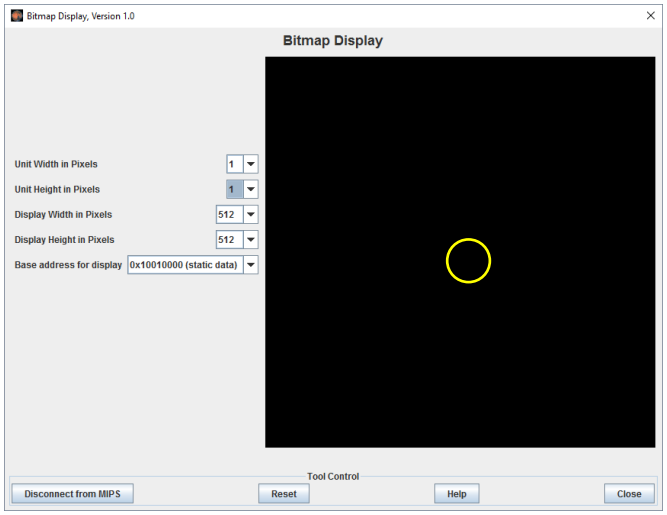
1. Vẽ hình trên màn hình Bitmap

Viết chương trình vẽ một quả bóng hình tròn di chuyển trên màn hình mô phỏng Bitmap Display của RARS. Nếu đối tượng đập vào cạnh của màn hình thì sẽ di chuyển theo chiều ngược lại.

Yêu cầu:

- Thiết lập màn hình ở kích thước 512x512. Kích thước pixel 1x1.
- Chiều di chuyển phụ thuộc vào phím người dùng bấm, gồm có (di chuyển lên (W), di chuyển xuống (S), sang trái (A), sang phải (D), tăng tốc độ (Z), giảm tốc độ (X) trong bộ giả lập Keyboard and Display MMIO Simulator).
- Vị trí bóng ban đầu ở giữa màn hình.

Gợi ý: Để làm một đối tượng di chuyển thì chúng ta sẽ xóa đối tượng ở vị trí cũ và vẽ đối tượng ở vị trí mới. Để xóa đối tượng chúng ta chỉ cần vẽ đối tượng đó với màu là màu nền.

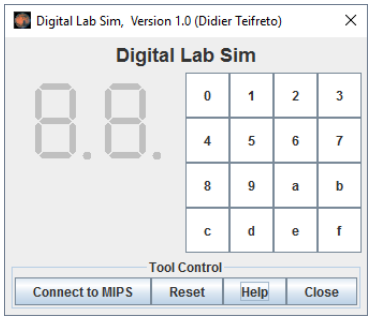


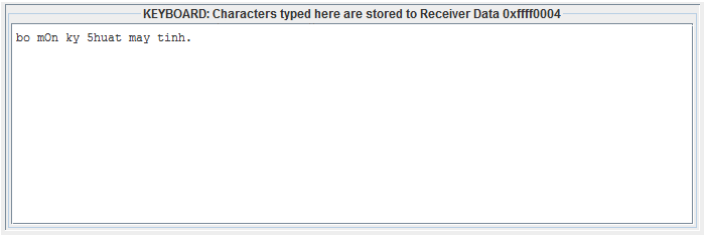
`

2. Kiểm tra tốc độ và độ chính xác khi gõ văn bản

Thực hiện chương trình đo tốc độ gõ bàn phím và hiển thị kết quả bằng 2 đèn led 7 đoạn. Nguyên tắc:

- Cho một đoạn văn bản mẫu, cố định sẵn trong mã nguồn. Ví dụ *"bo mon ky thuat may tinh"*
- Sử dụng bộ định thời Timer (trong bộ giả lập Digital Lab Sim) để tạo ra khoảng thời gian để đo. Đây là thời gian giữa 2 lần ngắt, chu kì ngắt.
- Người dùng nhập các kí tự từ bàn phím. Ví dụ nhập "*bo m**O**n ky **5**huat may tinh*". Chương trình cần phải đếm số kí tự đúng (trong ví dụ trên thì người dùng gõ sai chữ **O** và **5**) mà người dùng đã gõ và hiển thị lên các đèn led.
- Chương trình đồng thời cần tính được tốc độ gõ: thời gian hoàn thành và số từ trên một đơn vị thời gian.





3. Biểu thức trung tố hậu tố

Viết chương trình tính giá trị biểu thức bất kỳ bằng phương pháp duyệt biểu thức hậu tố.

Các yêu cầu cụ thể:

1. Nhập vào biểu thức trung tố, ví dụ: 9 + 2 + 8 * 6
2. In ra biểu thức ở dạng hậu tố, ví dụ: 9 2 + 8 6 * +
3. Tính ra giá trị của biểu thức vừa nhập

Các hằng số là số nguyên, trong phạm vi từ 0 → 99.

Toán tử bao gồm các phép toán cộng, trừ, nhân, chia lấy thương (/), chia lấy dư (%), đóng mở ngoặc.

4. Hàm cấp phát bộ nhớ malloc()

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.

Toàn bộ chương trình

```
.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 kieu Word

.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 tao vung nho cap phat dong
     jal   SysInitMem

     #-----------------------
     #  Cap phat cho bien con tro, gom 3 phan tu,moi phan tu 1 byte
     #-----------------------
     la   a0, CharPtr
     addi a1, zero, 3
     addi a2, zero, 1
     jal malloc
     #-----------------------
     #  Cap phat cho bien con tro, gom 6 phan tu, moi phan tu 1 byte
     #-----------------------
     la   a0, BytePtr
     addi a1, zero, 6
     addi a2, zero, 1
     jal  malloc
     #-----------------------
     #  Cap phat cho bien con tro, gom 5 phan tu, moi phan tu 4 byte
     #-----------------------
     la   a0, WordPtr
     addi a1, zero, 5
     addi a2, zero, 4
     jal  malloc


lock: j lock
      nop

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

5. Chương trình kiểm tra cú pháp lệnh hợp ngữ

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

- Nhập vào từ bàn phím một dòng lệnh hợp ngữ. Ví dụ *beq s1, 31, t4*
- *K*iểm tra xem mã opcode có đúng hay không? Trong ví dụ trên, opcode là beq là hợp lệ thì hiện thị thông báo *"opcode: beq, hợp lệ"*
- Kiểm tra xem tên các toán hạng phía sau có hợp lệ hay không? Trong ví dụ trên, *toán hạng s1 là hợp lệ, 31 là không hợp lệ, t4 thì khỏi phải kiểm tra nữa vì toán hạng trước đã bị sai rồi.*

Gợi ý: nên xây dựng một cấu trúc chứa khuôn dạng của từng lệnh với tên lệnh, kiểu của toán hạng 1, toán hạng 2, toán hạng 3.

6. Mô phỏng ổ đĩa RAID 5

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 '*'

```
Nhap chuoi ki tu : DCE.****ABCD1234HUSTHUST
     Disk 1                Disk 2                Disk 3
 --------------        --------------        --------------
|     DCE.     |      |     ****     |      [[ 6e,69,6f,04]]
|     ABCD     |      [[ 70,70,70,70]]      |     1234     |
[[ 00,00,00,00]]      |     HUST     |      |     HUST     |
 --------------        --------------        --------------
```

7. Vẽ hình bằng kí tự ASCII

Cho hình ảnh đã được chuyển thành các kí tự ASCII như hình vẽ. Đây là hình của chữ DCE có viền * và màu là các con số.

```
                                          *************
 **************                          *3333333333333*
 *222222222222222*                        *33333********
 *22222******222222*                      *33333*
 *22222*      *22222*                     *33333********
 *22222*       *22222*      *************  *3333333333333*
 *22222*       *22222*    **11111*****111* *33333********
 *22222*       *22222*  **1111**       **  *33333*
 *22222*      *222222*  *1111*             *33333********
 *22222*******222222*  *11111*             *3333333333333*
 *2222222222222222*    *11111*              *************
 ***************       *11111*
       ---              *1111**
     / o o \             *1111****   *****
     \   > /              **111111***111*
      -----                 ***********    dce.hust.edu.vn
```

- Hãy hiển thị hình ảnh trên lên giao diện console (hoặc giao diện Display trong công cụ giả lập Keyboard and Display MMIO Simulator)
- Hãy sửa ảnh để các chữ cái DCE chỉ còn lại viền, không còn màu số ở giữa, và hiển thị
- Hãy sửa ảnh để hoán đổi vị trí của các chữ, thành ECD, và hiển thị. Để đơn giản, các hoạ tiết đính kèm cũng được phép di chuyển theo.
- Hãy nhập từ bàn phím kí tự màu cho chữ D, C, E, rồi hiển thị ảnh trên với màu mới.

Chú ý: ngoài vùng nhớ lớn chứa ảnh được chứa sẵn trong code, không được tạo thêm vùng nhớ mới để chứa ảnh hiệu chỉnh.

8. Máy tính bỏ túi

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)

9. Kiểm thử các thuật toán sắp xếp

+ Tìm hiểu hàm hệ thống để đọc, ghi file văn bản.
+ Cho trước file văn bản chứa các số nguyên ngẫu nhiên, phân cách bởi dấu cách. Số lượng phần tử có thể lớn, tối đa 10000 phần tử.
+ Tạo giao diện cho phép người dùng nhập tên file để mở, các số trong file được đọc và lưu vào bộ nhớ.
+ Người dùng chọn thuật toán sắp xếp cần thực hiện (Nổi bọt, Chèn, Lựa chọn). Được điểm cộng nếu thực hiện thêm các thuật toán khác.
+ Chương trình chạy thuật toán và in ra thời gian thực hiện.
+ Chương trình ghi kết quả sắp xếp vào file kết quả.

10. Khóa điện tử

Sử dụng ngoại vi là bàn phím ma trận và LED 7 đoạn để lập trình ứng dụng khóa điện tử với các yêu cầu như sau:
+ Mật khẩu để mở khóa là một dãy gồm n chữ số (các chữ số từ 0 đến 9, có ít nhất 4 chữ số) được thiết lập sẵn trong bộ nhớ.
+ Để mở cửa, người dùng nhập mật khẩu bằng cách nhấn các nút số trên bàn phím ma trận, kết thúc bằng nút f. Nếu mật khẩu đúng, khóa mở, LED hiển thị ON. Nếu mật khẩu sai, LED hiển thị OF(F).
+ Nếu mật khẩu mở sai quá 3 lần nhập, khóa sẽ bị treo trong vòng 1 phút. Trong thời gian này, các phím bấm không có tác dụng.
+ Người dùng có thể thay đổi mật khẩu bằng cách nhấn nút a, sau đó nhấn các nút số ứng với mật khẩu hiện tại, rồi nhấn nút f. Nếu đúng mật khẩu cũ, hệ thống sẽ yêu cầu nhập mật khẩu mới. Người dùng nhập mật khẩu mới bằng cách nhấn các nút số, rồi nhấn nút f để kết thúc. Mật khẩu mới được lưu đè vào mật khẩu cũ trong bộ nhớ.

11. Vẽ đồ thị hàm số

Vẽ đồ thị biểu diễn hàm số trong cửa sổ Bitmap Display với các yêu cầu cụ thể sau:
+ Nhập các hệ số của phương trình $y = ax^2 + bx + c$ từ bàn phím
+ Nhập mã màu đồ thị
+ Vẽ đồ thị với trục tọa độ Oxy nằm giữa màn hình Bitmap Display (thiết lập màn hình có kích thước 512x512)
+ Sau khi vẽ xong một đồ thị, chương trình cho phép người dùng chọn vẽ tiếp đồ thị khác (chồng lên các đồ thị đã vẽ) hoặc kết thúc chương trình.

12. Đọc file ảnh BMP hiển thị ra màn hình Bitmap Display

+ Tìm hiểu về các hàm dịch vụ để mở và đọc dữ liệu từ file.
+ Đọc dữ liệu ảnh thì file BMP với độ phân giải lớn nhất là 512x512.
+ Hiển thị ảnh ra màn hình Bitmap Display.
+ Các ảnh có thể có độ phân giải khác nhau. Nếu độ phân giải của ảnh nhỏ hơn 512x512 thì hiển thị ảnh vào giữa màn hình.

13. Game luyện trí nhớ

+ Tham khảo: https://www.memozor.com/simon-games/simon-game
+ Tìm hiểu về hàm hệ thống sinh số ngẫu nhiên.
+ Trên màn hình Bitmap Display hiển thị 4 hình vuông có màu khác nhau
+ Lượt chơi đầu tiên bắt đầu với một hình vuông ngẫu nhiên được chọn, hiển thị với màu sáng hơn. Sau khoảng 1s hình được chọn trở về màu bình thường.
+ Người chơi chọn hình bằng cách nhấn vào nút tương ứng trong KeyMatrix (chỉ sử dụng các nút từ 1 đến 4)
+ Nếu người dùng nhập đúng số theo thứ tự thì chuyển sang lượt chơi tiếp theo với số ô được chọn ngẫu nhiên tăng thêm 1. Nếu nhập sai thì kết thúc trò chơi.

14. Game lật thẻ

+ Tham khảo: https://www.memozor.com/memory-games/for-toddlers-babies/dick-bruna
+ Tìm hiểu về hàm hệ thống tạo số ngẫu nhiên
+ Trên màn hình Bitmap Display hiển thị 4x4 thẻ hình vuông với 8 cặp màu ở vị trí ngẫu nhiên, ở trạng thái úp.
+ Người dùng mở thẻ bằng cách nhấn các nút trên KeyMatrix. Màn hình sẽ mở thẻ tương ứng.
Nếu cặp thẻ vừa được mở giống màu nhau thì sẽ ở trạng thái mở, nếu cặp thẻ khác màu thì chuyển lại về trạng thái úp.
+ Nếu tất cả các thẻ được mở thì trò chơi kết thúc.

15. Game nhớ số

+ Tham khảo: https://www.memozor.com/other-memory-games/numbers-memory-games/grid-of-numbers-to-remember-1-1000
+ Tìm hiểu về hàm hệ thống tạo số ngẫu nhiên
+ Hiển thị 4 số ngẫu nhiên trong cửa sổ DISPLAY (của thiết bị Bitmap and Display MMIO). Các số này có ít nhất 3 chữ số.
+ Sau một khoảng thời gian đếm ngược thì xóa các số này
+ Người dùng nhập các số vào cửa sổ Keyboard, phân cách bởi dấu cách, kết thúc bởi Enter.
+ Chương trình sẽ xác định các số người dùng nhập vào có đúng không. Nếu đúng thì chuyển sang lượt tiếp theo. Nếu sai thì kết thúc trò chơi. Các số không cần đúng thứ tự.

16. Game Tic Tac Toe

+ Tham khảo Tic-Tac-Toe - Play retro Tic-Tac-Toe online for free
+ Lập trình game cho 2 người chơi với kích thước 4x4
+ Thiết lập kích thước Bitmap Display là 512x512
+ Vẽ lưới 4x4
+ Người chơi sử dụng bàn phím 4x4 để chọn vị trí
+ Vẽ X hoặc O vào vị trí người chơi đã chọn
+ Xác định trạng thái kết thúc ván chơi

17. Phát nhạc theo kịch bản – Đàn điện tử

+ Tìm hiểu về hàm hệ thống phát nhạc
+ Bản nhạc là chuỗi ký tự chứa các bộ 4 tham số gồm cao độ, trường độ, loại nhạc cụ và âm lượng. Ví dụ "60, 1200,1, 120, 73, 220,1, 125, ..."
+ Chuẩn bị trước 4 bản nhạc
+ Khi chương trình chạy, người dùng chọn bản nhạc để phát bằng cách nhấn vào các nút tương ứng từ 1 đến 4, nhấn nút 0 để dừng phát.

18. Đồng hồ điện tử

+ Hiển thị thời gian hiện tại lên đèn LED 7 đoạn
+ Bấm nút trên bàn phím KeyMatrix để chuyển chế độ hiển thị
    - Số 1: Hiển thị giờ
    - Số 2: Hiển thị phút
    - Số 3: Hiển thị giây
    - Số 4: Hiển thị ngày
    - Số 5: Hiển thị tháng
    - Số 6: Hiển thị 2 số cuối của năm
+ Khi mỗi giây trôi qua cần cập nhật thời gian và hiển thị
+ Khi chẵn một phút thì phát âm thanh báo hiệu.
+ Gợi ý: Tìm hiểu về hàm hệ thống lấy giờ hiện tại, phát âm thanh.