HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY



BÁO CÁO CUỐI KỲ

**THỰC HÀNH KIẾN TRÚC MÁY TÍNH**

**NHÓM 14**

**Giảng viên hướng dẫn**

Ngô Lam Trung

**Trợ Giảng**

Đỗ Quốc Minh

**Thành viên nhóm**

Phan Khánh Vũ

Hoàng Văn Thắng

*Mục lục*

[1. Assignment 6: Mô phỏng ổ đĩa RAID 5 3](#_Toc199336055)

[1.1. Mô tả bài toán: 3](#_Toc199336056)

[1.2. Thuật toán sử dụng: 4](#_Toc199336057)

[1.2.1. Thuật toán 1 : 4](#_Toc199336058)

[1.2.2. Thuật toán 2 4](#_Toc199336059)

[1.2.3. Thuật toán 3: 4](#_Toc199336060)

[1.2.4. Thuật toán 4: 4](#_Toc199336061)

[1.3. Đoạn mã lệnh và giải thích 5](#_Toc199336062)

[1.3.1. Mã lệnh 1 5](#_Toc199336063)

[1.3.2. Mã lệnh 2 7](#_Toc199336064)

[1.3.3. Mã lệnh 3 8](#_Toc199336065)

[1.3.4. Mã lệnh 4 9](#_Toc199336066)

[1.3.5. Mã lệnh 5 11](#_Toc199336067)

[1.4. Chạy thử mã lệnh và kết quả: 16](#_Toc199336068)

[1.4.1. Kết quả hiển thị ảnh: 16](#_Toc199336069)

[1.4.2. Kết quả hiển thị ảnh đã xóa màu cho các chữ cái DCE trong ảnh: 16](#_Toc199336070)

[1.4.3. Kết quả hiển thị ảnh đã hoán đổi vị trí các chữ cái: 17](#_Toc199336071)

[1.4.4. Kết quả hiển thị ảnh các chữ cái DCE với giá trị màu mới: 17](#_Toc199336072)

[2. Assignment 9: Kiểm thử các thuật toán sắp xếp 18](#_Toc199336073)

[2.1. Mô tả bài toán: 18](#_Toc199336074)

[2.2. Thuật toán sử dụng: 19](#_Toc199336075)

[2.2.1. Thuật toán 1 19](#_Toc199336076)

[2.2.2. Thuật toán 2 19](#_Toc199336077)

[2.2.3. Thuật toán 3 19](#_Toc199336078)

[2.3. Đoạn mã lệnh và giải thích: 20](#_Toc199336079)

[2.3.1. Khởi tạo các mảng ban đầu: 20](#_Toc199336080)

[2.3.2. Khởi tạo ngẫu nhiên các giá trị màu: 21](#_Toc199336081)

[2.3.3. Đầu vào và xử lý đầu vào: 23](#_Toc199336082)

[2.3.4. Xử lý gán màu và hiển thị màu: 25](#_Toc199336083)

[2.3.5. Thắng trò chơi: 29](#_Toc199336084)

[2.4. Kết quả chạy mẫu: 30](#_Toc199336085)

# Assignment 6: Mô phỏng ổ đĩa RAID 5

## Mô tả bài toán:

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 ‘\*’

A close-up of a computer code

AI-generated content may be incorrect.

## Thuật toán sử dụng:

### Thuật toán 1 :

* Hình ảnh sau khi được chuyển thành kí tự ASCII được chuyển thành từng dòng và lưu vào vùng .data với mỗi dòng gồm 60 kí tự bao gồm cả viền, và tổng cộng 16 dòng.
* Để thực hiện in ảnh qua cửa sổ console ta thực hiện duyệt qua mỗi ‘line’ để in ra ảnh dưới dạng ASCII. Sử dụng lệnh hệ thống với a7 = 4 để thực hiện in dữ liệu của các ‘line’.

### Thuật toán 2

* Thực hiện xóa màu cho các chữ cái trong ảnh (với màu ở đây là các chữ số) ta thực hiện duyệt qua toàn bộ phần tử của ảnh dưới dạng ASCII. Nếu kí tự đó nằm trong khoảng từ kí tự ‘0’ đến kí tự ‘9’ thì ta thực hiện thay thế kí tự đó bằng dấu ‘ ‘. Sau đó thực hiện in ra màn hình.

### Thuật toán 3:

* Thực hiện hoán đổi vị trí các chữ cái từ DCE sang ECD ta tách các kí tự thành các phần riêng biệt khác nhau. Sau khi cắt ta được vị trí các chữ cái với D là vị trí bắt đầu x = 0, E có vị trí x = 23, D có vị trí x = 43.
* Sau khi tách ta thực hiện in các kí tự theo thứ tự mong muốn (E – D – C ) bằng cách lấy từ vị trí bắt đầu đến trước vị trí bắt đầu của kí tự tiếp theo của mỗi dòng, và in ra console.

### Thuật toán 4:

* Thực hiện tô màu mới cho các chữ cái, ta xác định từng chữ cái bằng cách chia từng phần theo vị trí bắt đầu và kết thúc của mỗi kí tự. Sau khi cắt ta được vị trí các chữ cái với D là vị trí bắt đầu x = 0, E có vị trí x = 23, D có vị trí x = 43.
* Sau khi cắt ta thực hiện đổi màu. Thực hiện hiển thị yêu cầu nhập màu mới cho chữ cái từ bàn phím, sau đó thay thế các kí tự từ khoảng kí tự ‘0’ đến ‘9’ trong kí tự vừa nhập bằng kí tự vừa nhập và lưu lại vào các line.
* Sau khi đã chọn màu của các kí tự ta thực hiện in các kí tự ra màn hình.

## Đoạn mã lệnh và giải thích

### Mã lệnh 1

|  |
| --- |
| .data  line1: .asciz " \*\*\*\*\*\*\*\*\*\*\*\*\* \n"  line2: .asciz "\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*3333333333333\*\n"  line3: .asciz "\*222222222222222\* \*33333\*\*\*\*\*\*\*\* \n"  line4: .asciz "\*22222\*\*\*\*\*\*222222\* \*33333\* \n"  line5: .asciz "\*22222\* \*22222\* \*33333\*\*\*\*\*\*\*\* \n"  line6: .asciz "\*22222\* \*22222\* \*\*\*\*\*\*\*\*\*\*\*\*\* \*3333333333333\*\n"  line7: .asciz "\*22222\* \*22222\* \*\*11111\*\*\*\*\*111\* \*33333\*\*\*\*\*\*\*\* \n"  line8: .asciz "\*22222\* \*22222\* \*\*1111\*\* \*\* \*33333\* \n"  line9: .asciz "\*22222\* \*222222\* \*1111\* \*33333\*\*\*\*\*\*\*\* \n"  line10: .asciz "\*22222\*\*\*\*\*\*\*222222\* \*11111\* \*3333333333333\*\n"  line11: .asciz "\*2222222222222222\* \*11111\* \*\*\*\*\*\*\*\*\*\*\*\*\* \n"  line12: .asciz "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*11111\* \n"  line13: .asciz " --- \*1111\*\* \n"  line14: .asciz " / o o \\ \*1111\*\*\*\* \*\*\*\*\* \n"  line15: .asciz " \\ > / \*\*111111\*\*\*111\* \n"  line16: .asciz " ----- \*\*\*\*\*\*\*\*\*\*\* dce.hust.edu.vn\n"  menu\_message: .asciz "\n\n ----MENU----\n 1. Show picture.\n 2. Show picture with only border.\n 3. Change the order.\n 4. Enter new color number and update.\n 5. Exit.\n Enter your choice: "  error\_message: .asciz "Input must be a integer from 1 to 5"  input\_d\_color: .asciz "Enter color for D (integer from 0-9):"  input\_c\_color: .asciz "Enter color for C (integer from 0-9):"  input\_e\_color: .asciz "Enter color for E (integer from 0-9):"  .text  menu:  # Hiển thị menu cho người dùng  li a7, 4 # Syscall số 4: in chuỗi  la a0, menu\_message # Load địa chỉ thông điệp menu  ecall # Gọi hệ thống để in menu    # Nhận đầu vào từ người dùng  li a7, 5 # Syscall số 5: đọc số nguyên  ecall # Gọi hệ thống để đọc input    # Kiểm tra lựa chọn của người dùng và thực hiện chức năng tương ứng  add t0, a0, zero # t0 lưu trữ lựa chọn của người dùng  li t1, 1 # Lựa chọn 1: hiển thị hình ảnh có màu  beq t0, t1, printPicture # Nếu chọn 1, nhảy đến printPicture    li t1, 2 # Lựa chọn 2: hiển thị hình ảnh không màu  beq t0, t1, printNotColor # Nếu chọn 2, nhảy đến printNotColor    li t1, 3 # Lựa chọn 3: thay đổi thứ tự hiển thị  beq t0, t1, changeOrder # Nếu chọn 3, nhảy đến changeOrder    li t1, 4 # Lựa chọn 4: thay đổi màu sắc  beq t0, t1, changeColor # Nếu chọn 4, nhảy đến changeColor    li t1, 5 # Lựa chọn 5: thoát chương trình  beq t0, t1, exit # Nếu chọn 5, nhảy đến exit  catch\_error:  # Xử lý lỗi khi người dùng nhập sai lựa chọn  li a7, 4 # Syscall in chuỗi  la a0, error\_message # Load thông điệp lỗi  ecall # In thông điệp lỗi  j menu # Quay lại menu chính |

* Các line từ 1 đến 16 để lưu vào mã ASCII của ảnh menu\_message để lưu menu hiển thị ra màn hình
* Các input\_d,c,e... để lưu chuỗi hiển thị chọn màu mới ra màn hình
* Bắt đầu chương trình thực hiện in menu ra màn hình thông qua lệnh hệ thống a7 = 4, sau đó nhập lựu chọn là số vào màn hình thông qua a7 = 5.
* Thực hiện so sánh số vừa nhập ứng với các lựa chọn của của menu. Nếu đúng với các case của hệ thống thì thực hiện chuyển đến chức năng ứng với số đó, nếu không đúng thực hiện in thông báo lỗi ra màn hình và quay lại menu
* Menu dừng khi nhập số 0 và kết thúc chương trình

### Mã lệnh 2

|  |
| --- |
| printPicture:  # Chức năng 1: In hình ảnh đầy đủ với màu sắc  li t0, 0 # Chỉ số dòng hiện tại (bắt đầu từ 0)  li t1, 16 # Tổng số dòng cần in (16 dòng)  la a0, line1 # Load địa chỉ dòng đầu tiên của hình ảnh  loop\_for\_print\_picture:  # Vòng lặp để in từng dòng của hình ảnh  beq t0, t1, menu # Nếu đã in hết 16 dòng, quay lại menu  li a7, 4 # Syscall in chuỗi  ecall # In dòng hiện tại  addi a0, a0, 60 # Chuyển đến địa chỉ dòng tiếp theo (mỗi dòng 60 ký tự)  addi t0, t0, 1 # Tăng chỉ số dòng lên 1  j loop\_for\_print\_picture # Lặp lại để in dòng tiếp theo |

* Khởi tạo các thanh ghi là đếm và số lượng line tương ứng với line của ảnh
* Sử dụng vòng lặp để in lần lượt dữ liệu mỗi dòng của ảnh ra màn hình, với câu lệnh hệ thống a7 = 4, rồi thực hiện tăng địa chỉ của đến với line tiếp theo bằng a0 = a0 + 60 (với 60 là số kí tự mỗi line).
* Khi đã in hết toàn bộ mã ASCII của ảnh thì thực hiện quay lại menu và tiếp tục chương trình.

### Mã lệnh 3

|  |
| --- |
| printNotColor:  # Chức năng 2: In hình ảnh không màu (thay thế số bằng khoảng trắng)  li t0, 0 # Chỉ số dòng hiện tại  li t1, 16 # Tổng số dòng  li t2, '0' # Ký tự '0' - giới hạn dưới của màu  li t3, '9' # Ký tự '9' - giới hạn trên của màu  li t4, 60 # Số ký tự trong mỗi dòng  la a1, line1 # Địa chỉ bắt đầu của hình ảnh  loop\_for\_print\_not\_color:  # Vòng lặp ngoài: duyệt qua từng dòng  beq t0, t1, menu # Nếu đã duyệt hết 16 dòng, quay về menu  li t5, 0 # Chỉ số cột (bắt đầu từ 0)  inner\_loop\_not\_color:  # Vòng lặp trong: duyệt qua từng ký tự trong dòng  beq t5, t4, cont\_loop\_not\_color # Nếu đã duyệt hết 60 ký tự, chuyển dòng tiếp theo  lb t6, 0(a1) # Load ký tự hiện tại từ bộ nhớ vào t6  # Kiểm tra xem ký tự có phải là màu (số từ 0-9) không  blt t6, t2, printChar # Nếu t6 < '0' => không phải màu, in nguyên bản  bgt t6, t3, printChar # Nếu t6 > '9' => không phải màu, in nguyên bản  li t6, ' ' # Nếu là màu, thay thế bằng khoảng trắng  printChar:  # In ký tự ra màn hình  li a7, 11 # Syscall in ký tự  mv a0, t6 # Chuyển ký tự vào a0  ecall # In ký tự  addi t5, t5, 1 # Tăng chỉ số cột  addi a1, a1, 1 # Chuyển đến địa chỉ ký tự tiếp theo  j inner\_loop\_not\_color # Tiếp tục vòng lặp trong  cont\_loop\_not\_color:  addi t0, t0, 1 # Tăng chỉ số dòng  j loop\_for\_print\_not\_color # Tiếp tục vòng lặp ngoài |

* Khởi tạo các thanh ghi đếm dòng, cột, cũng như giới hạn giá trị của màu (các ký tự từ ‘0’ đến ‘9’)
* Sử dụng hai vòng lặp lồng nhau:
  + Vòng ngoài duyệt qua từng dòng
  + Vòng trong duyệt qua từng ký tự trong dòng
* Nếu ký tự thuộc phạm vi từ ‘0’ đến ‘9’, tức là ký tự màu, thì thay thế bằng dấu cách ‘ ‘ rồi in ra màn hình
* Nếu không phải ký tự màu, in trực tiếp ký tự gốc
* Khi duyệt hết 16 dòng ảnh, quay lại menu chính

### Mã lệnh 4

|  |
| --- |
| changeOrder:  # Chức năng 3: Thay đổi thứ tự hiển thị từ DCE thành ECD  # Cấu trúc ban đầu: D (0-22) C (23-42) E (43-57)  # Cấu trúc mới: E C D  li t0, 0 # Chỉ số dòng hiện tại  li t1, 16 # Tổng số dòng  la a1, line1 # Địa chỉ bắt đầu của hình ảnh  loop\_for\_changeOrder:  beq t0, t1, menu # Nếu đã xử lý hết 16 dòng, quay về menu  # Tạm thời thay thế các khoảng trắng và xuống dòng bằng null để tách chuỗi  sb zero, 22(a1) # Thay khoảng trắng giữa D và C bằng '\0'  sb zero, 42(a1) # Thay khoảng trắng giữa C và E bằng '\0'  sb zero, 58(a1) # Thay '\n' bằng '\0'  # In theo thứ tự mới: E C D  # In phần E (từ vị trí 43)  li a7, 4 # Syscall in chuỗi  addi a0, a1, 43 # E bắt đầu từ địa chỉ gốc + 43  ecall # In phần E  # In khoảng trắng phân cách  li a7, 11 # Syscall in ký tự  li a0, ' ' # Ký tự khoảng trắng  ecall  # In phần C (từ vị trí 23)  li a7, 4 # Syscall in chuỗi  addi a0, a1, 23 # C bắt đầu từ địa chỉ gốc + 23  ecall # In phần C  # In khoảng trắng phân cách  li a7, 11 # Syscall in ký tự  li a0, ' ' # Ký tự khoảng trắng  ecall  # In phần D (từ vị trí 0)  li a7, 4 # Syscall in chuỗi  addi a0, a1, 0 # D bắt đầu từ địa chỉ gốc  ecall # In phần D  # In ký tự xuống dòng  li a7, 11 # Syscall in ký tự  li a0, '\n' # Ký tự xuống dòng  ecall  # Khôi phục lại các ký tự đã thay đổi  li t3, ' ' # Ký tự khoảng trắng  sb t3, 22(a1) # Khôi phục khoảng trắng giữa D và C  sb t3, 42(a1) # Khôi phục khoảng trắng giữa C và E  li t3, '\n' # Ký tự xuống dòng  sb t3, 58(a1) # Khôi phục ký tự xuống dòng  addi t0, t0, 1 # Tăng chỉ số dòng  addi a1, a1, 60 # Chuyển đến địa chỉ dòng tiếp theo  j loop\_for\_changeOrder # Lặp lại cho dòng tiếp theo |

* Khởi tạo thanh ghi đếm dòng và địa chỉ dòng đầu tiên của ảnh
* Trong mỗi vòng lặp dòng:
  + Thực hiện chia mỗi dòng thành ba phần:
    - D nằm trong đoạn a1[0:22]
    - C nằm trong đoạn a1[23:42]
    - E nằm trong đoạn a1[43:58]
  + Thay thế các ký tự space (‘ ‘) và newline (‘\n’) tại các vị trí ngắt bằng \0 để in từng phần tách biệt
  + In lần lượt theo thứ tự mới: E → C → D
  + In thêm khoảng trắng giữa các phần để phân cách
  + Sau khi in, phục hồi lại dữ liệu dòng ban đầu để đảm bảo dữ liệu ảnh không bị thay đổi
* Lặp lại thao tác trên cho 16 dòng, sau đó quay lại menu chính

### Mã lệnh 5

|  |
| --- |
| c changeColor:  # Chức năng 4: Thay đổi màu sắc của các phần D, C, E  li a5, 9 # Giới hạn trên cho đầu vào (0-9)    input\_d:  # Nhập màu mới cho phần D  li a7, 4 # Syscall in chuỗi  la a0, input\_d\_color # Thông điệp yêu cầu nhập màu D  ecall # In thông điệp  li a7, 5 # Syscall đọc số nguyên  ecall # Đọc đầu vào  bltz a0, input\_d # Nếu âm, yêu cầu nhập lại  bgt a0, a5, input\_d # Nếu > 9, yêu cầu nhập lại  addi t0, a0, '0' # Chuyển đổi số thành ký tự (số + '0')    input\_c:  # Nhập màu mới cho phần C  li a7, 4 # Syscall in chuỗi  la a0, input\_c\_color # Thông điệp yêu cầu nhập màu C  ecall # In thông điệp  li a7, 5 # Syscall đọc số nguyên  ecall # Đọc đầu vào  bltz a0, input\_c # Nếu âm, yêu cầu nhập lại  bgt a0, a5, input\_c # Nếu > 9, yêu cầu nhập lại  addi t1, a0, '0' # Chuyển đổi số thành ký tự    input\_e:  # Nhập màu mới cho phần E  li a7, 4 # Syscall in chuỗi  la a0, input\_e\_color # Thông điệp yêu cầu nhập màu E  ecall # In thông điệp  li a7, 5 # Syscall đọc số nguyên  ecall # Đọc đầu vào  bltz a0, input\_e # Nếu âm, yêu cầu nhập lại  bgt a0, a5, input\_e # Nếu > 9, yêu cầu nhập lại  addi t2, a0, '0' # Chuyển đổi số thành ký tự  solve\_change\_color:  # Thực hiện thay đổi màu trong toàn bộ hình ảnh  li t3, 0 # Chỉ số dòng (i)  li t4, 16 # Tổng số dòng  la a1, line1 # Địa chỉ bắt đầu của hình ảnh  li t6, 60 # Số ký tự trong mỗi dòng  li a2, 23 # Vị trí bắt đầu của phần C  li a3, 43 # Vị trí bắt đầu của phần E  li a4, '1' # Ký tự '1' - màu nhỏ nhất cần thay đổi  li a5, '9' # Ký tự '9' - màu lớn nhất cần thay đổi    loop\_for\_change\_color:  # Vòng lặp ngoài: duyệt qua từng dòng  beq t3, t4, printPicture # Sau khi thay đổi xong, hiển thị hình ảnh  li t5, 0 # Chỉ số cột (j)    inner\_loop\_to\_change\_color:  # Vòng lặp trong: duyệt qua từng ký tự trong dòng  beq t5, t6, cont\_loop\_change\_color # Nếu hết cột, chuyển dòng tiếp theo    # Xác định ký tự thuộc phần nào dựa vào vị trí  blt t5, a2, update\_D # Vị trí 0-22: thuộc phần D  blt t5, a3, update\_C # Vị trí 23-42: thuộc phần C  j update\_E # Vị trí 43-59: thuộc phần E  update\_D:  # Cập nhật màu cho phần D  lb s1, 0(a1) # Load ký tự hiện tại  blt s1, a4, cont\_inner\_loop\_change\_color # Nếu < '1', không phải màu  bgt s1, a5, cont\_inner\_loop\_change\_color # Nếu > '9', không phải màu  sb t0, 0(a1) # Thay thế bằng màu mới cho D  j cont\_inner\_loop\_change\_color  update\_C:  # Cập nhật màu cho phần C  lb s1, 0(a1) # Load ký tự hiện tại  blt s1, a4, cont\_inner\_loop\_change\_color # Nếu < '1', không phải màu  bgt s1, a5, cont\_inner\_loop\_change\_color # Nếu > '9', không phải màu  sb t1, 0(a1) # Thay thế bằng màu mới cho C  j cont\_inner\_loop\_change\_color  update\_E:  # Cập nhật màu cho phần E  lb s1, 0(a1) # Load ký tự hiện tại  blt s1, a4, cont\_inner\_loop\_change\_color # Nếu < '1', không phải màu  bgt s1, a5, cont\_inner\_loop\_change\_color # Nếu > '9', không phải màu  sb t2, 0(a1) # Thay thế bằng màu mới cho E  cont\_inner\_loop\_change\_color:  addi t5, t5, 1 # Tăng chỉ số cột  addi a1, a1, 1 # Chuyển đến địa chỉ ký tự tiếp theo  j inner\_loop\_to\_change\_color # Tiếp tục vòng lặp trong  cont\_loop\_change\_color:  addi t3, t3, 1 # Tăng chỉ số dòng  j loop\_for\_change\_color # Tiếp tục vòng lặp ngoài |

* Hiển thị lần lượt các thông báo yêu cầu người dùng nhập vào giá trị màu mới cho từng chữ cái D, C và E (số nguyên từ 0 đến 9)
* Sau khi nhập xong:
  + Với mỗi dòng ảnh:
    - Duyệt qua từng ký tự trong dòng
    - Dựa vào vị trí cột:
      * Cột từ 0–22 thuộc về chữ D → thay ký tự màu thành ký tự mới cho D
      * Cột từ 23–42 thuộc về chữ C → thay ký tự màu thành ký tự mới cho C.
      * Cột từ 43–59 thuộc về chữ E→ thay ký tự màu thành ký tự mới cho E.
  + Việc thay đổi chỉ diễn ra với những ký tự nằm trong khoảng ‘1’ đến ‘9’ (xác định là màu cũ)
* Sau khi hoàn tất việc thay màu, chương trình in lại ảnh mới đã cập nhật màu, rồi quay về menu

## Chạy thử mã lệnh và kết quả:

### Kết quả hiển thị ảnh:

A screenshot of a computer

AI-generated content may be incorrect.

### Kết quả hiển thị ảnh đã xóa màu cho các chữ cái DCE trong ảnh:

A screenshot of a computer

AI-generated content may be incorrect.

### Kết quả hiển thị ảnh đã hoán đổi vị trí các chữ cái:

A screenshot of a computer

AI-generated content may be incorrect.

### Kết quả hiển thị ảnh các chữ cái DCE với giá trị màu mới:

A screen shot of a computer

AI-generated content may be incorrect.

# Assignment 9: Kiểm thử các thuật toán sắp xếp

## Mô tả bài toán:

- 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.

## Thuật toán sử dụng:

### Thuật toán 1

* Sử dụng 2 mảng, mỗi mảng 8 phần tử, một mảng lưu giá trị hexa của màu, một mảng lưu số lần màu đấy còn có thể được xuất hiện. Số thứ tự của màu trong mảng lưu giá trị hexa chính là số thứ tự của số lần màu đến còn có thể được xuất hiện trong mảng còn lại.
* Sử dụng một mảng để lưu màu (các màu ở mỗi lần sinh khác nhau sẽ ở vị trí khác nhau trong mảng – tăng dần theo số lần sinh)
* Sử dụng hàm hệ thống để sinh tự động số thứ tự của màu trong mảng để lấy được màu từ mảng rồi lưu vào mảng lưu màu. Sau đó giảm giá trị của mảng lưu số lần xuất hiện còn lại ở vị trí màu đấy. => Tránh lặp lại màu quá 2 lần.
* Thực hiện vòng lặp cho đến khi mảng lưu màu có đủ 16 màu.

### Thuật toán 2

* Nhấn phím trên keyboard của digital lab sim để chọn ô trên grid (4x4).
* Sử dụng polling để nhận tín hiệu phím được bấm.
* Trong lúc đợi phím được nhả thì kiểm tra lần lượt từng hàng trên bàn phím digital lab sim để xem xem phím ở hàng nào và cột nào được bấm.
* Nếu phím bấm ánh xạ sang bit map có ô đấy đã mở thì báo lỗi và bấm lại.

### Thuật toán 3

* Sau khi lấy được phím bấm thì thực hiện so sánh kiểm tra xem thẻ ở vị trí bấm đấy đã mở hay chưa.
  + Lấy màu được lưu từ mảng lưu màu với giá trị màu được lấy ở vị trí mà nút bấm ánh xạ tới (0 - 15).
  + Lấy màu từ vị trí bấm ở trên bit map để so sánh với màu được lấy từ mảng. (cùng vị trí ấy mà chỗ đấy đã có màu tức là ô đấy đã được mở).
* Nếu tại vị trí bấm đấy mà chưa mở thì thực hiện lưu màu và hiển thị màu trên bit map.
* Lưu chỉ số thẻ và màu thẻ lại nếu đó là lần đầu mở thẻ trong 2 lần của 1 lượt.
* Thực hiện lấy màu của lần bấm thứ hai của lượt chơi đó (cũng kiểm tra có chọn phải ô đã mở chưa)
  + So sánh màu ở 2 lần mở. Nếu giống thì chuyển sang lượt chơi tiếp theo, nếu không giống thì đóng 2 thẻ mở lại.
* Nếu đã mở đủ cả 8 cặp màu thì dừng chương trình.

## Đoạn mã lệnh và giải thích:

### Khởi tạo các mảng ban đầu:

|  |
| --- |
| .eqv IN\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0012  .eqv OUT\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0014  .eqv MONITOR\_SCREEN 0x10040000  .eqv PAIR\_COUNT 8  .eqv TILE\_COUNT 16  .data  info: .asciz "Hãy lựa chọn thẻ\n"  error: .asciz "Thẻ đã được lật! Hãy chọn thẻ khác\n"  finish: .asciz "Bạn đã thắng!!!\n"  Colors: .word 0x00FF0000, 0x0000FF00, 0x000000FF, 0x00FFFF00,  0x0000FFFF, 0x00FF00FF, 0x00FFFFFF, 0x00808080  Mark: .word 2, 2, 2, 2, 2, 2, 2, 2 # Mỗi màu xuất hiện 2 lần  Temp: .space 64 # Lưu màu của 16 thẻ (16\*4 byte)  FirstIdx: .word 0 # Vị trí thẻ đầu tiên  FirstCol: .word 0 # Màu thẻ đầu tiên  # Ánh xạ phím sang chỉ số thẻ  keymap:  .word 0x00000011, 0  .word 0x00000021, 1  .word 0x00000041, 2  .word 0xFFFFFF81, 3  .word 0x00000012, 4  .word 0x00000022, 5  .word 0x00000042, 6  .word 0xFFFFFF82, 7  .word 0x00000014, 8  .word 0x00000024, 9  .word 0x00000044, 10  .word 0xFFFFFF84, 11  .word 0x00000018, 12  .word 0x00000028, 13  .word 0x00000048, 14  .word 0xFFFFFF88, 15  .word 0xFFFFFFFF, -1 # Kết thúc bảng ánh xạ |

* info, error, finish lưu các chuỗi sẽ hiển thị ra màn hình
* Colors là mảng lưu giá trị hexa của các màu
* Mark là mảng lưu số lần xuất hiện của các màu đó
* Temp là mảng lưu vị trí của các màu vào mỗi thẻ
* FirstIdx lưu vị trí của màu đầu tiên được mở trong 1 lượt chơi
* FirstCol lưu giá trị hexa của màu đầu tiên được mở trong 1 lượt chơi
* keymap là mảng 2 chiều lưu giá trị là tọa độ của nút được bấm cùng với vị trí của tọa độ đó trong mảng Temp.
* ***Địa chỉ của BITMAP DISPLAY là ở heap.***

### Khởi tạo ngẫu nhiên các giá trị màu:

|  |
| --- |
| .text  start:  li t0, 0  li t1, TILE\_COUNT  la t3, Temp  # Khởi tạo bảng thẻ với màu ngẫu nhiên  random\_loop:  beq t0, t1, start\_game  li a7, 42 # random  li a1, PAIR\_COUNT # giới hạn từ 0 đến 7  ecall  la t4, Colors  la t5, Mark  slli t6, a0, 2 # offset = a0 \* 4  add t5, t5, t6  lw t2, 0(t5) # số lần còn lại của màu này  beqz t2, random\_loop  add t4, t4, t6  lw s5, 0(t4) # lấy giá trị màu  sw s5, 0(t3) # lưu vào Temp  addi t3, t3, 4  addi t2, t2, -1 # giảm số lần sử dụng màu  sw t2, 0(t5)  addi t0, t0, 1  j random\_loop |

* Khởi tạo giá trị đếm các màu đã được khởi tạo đúng (hợp lý) là 0, đếm tăng dần cho đến khi đủ 16 (TILE\_COUNT) màu đã được khởi tạo.
* Hàm hệ thống khởi tạo số nguyên ngẫu nhiên từ 0 đến bound trừ 1 (PAIR\_COUNT - 1)
* Thực hiện lấy địa chỉ màu và số lần màu đó còn có thể xuất hiện từ mảng Colors (t4 - vị trí 0) và Mark (t5 - vị trí 0) bằng cách lấy địa chỉ tại ví trí 0 cộng với 4 lần giá trị số nguyên ngẫu nhiên vừa khởi tạo (a0).
* Nếu số lần xuất hiện của màu đã là 2 thì lặp lại để lấy giá trị mới còn không thì lưu màu vào mảng Temp ở vị trí mà mỗi lần lặp sẽ tăng lên (t3), giảm số lần màu đó còn có thể xuất hiện rồi lưu vào mảng Mark, và tăng giá trị đếm lên 1.
* Thực hiện lại vòng lặp cho đến khi đủ 16 màu được khởi tạo.

### Đầu vào và xử lý đầu vào:

|  |
| --- |
| start\_game:  li s0, 0 # số cặp đã khớp  li s1, PAIR\_COUNT  li s3, 0 # lượt chọn (0 = lần 1, 1 = lần 2)  game\_loop:  beq s0, s1, game\_done  li a7, 4  la a0, info  ecall  # Đợi người chơi nhấn phím  wait\_key:  li t1, 0x01  wait\_loop:  li t2, IN\_ADDRESS\_HEXA\_KEYBOARD  li t3, OUT\_ADDRESS\_HEXA\_KEYBOARD  sb t1, 0(t2)  lb a0, 0(t3)  beqz a0, next\_row  la t4, keymap  search\_keymap:  lw t5, 0(t4)  li t6, -1  beq t5, t6, wait\_key # không hợp lệ  beq t5, a0, found\_key  addi t4, t4, 8  j search\_keymap  found\_key:  lw s4, 4(t4) # lấy chỉ số thẻ  # Đợi người chơi nhả phím  wait\_key\_release:  li t2, OUT\_ADDRESS\_HEXA\_KEYBOARD  lb t3, 0(t2)  bnez t3, wait\_key\_release  jal retrieve # kiểm tra thẻ đã mở chưa  bnez t2, invalid\_input  beq s3, zero, save\_first  j check\_match  next\_row:  slli t1, t1, 1  li t4, 0x10  blt t1, t4, wait\_loop  j wait\_key |

* Bắt đầu trò chơi, khởi tạo giá trị số cặp đã khớp là 0 (s0) và lưu tổng số cặp là 8 (PAIR\_COUNT), và lưu số lần mở trong 1 lượt chơi là 0 (s3).
* Thực hiện vòng lặp trò chơi, lặp qua từng số cặp đã khớp cho đến khi bằng tổng số cặp màu thì kết thúc trò chơi.
* Thực hiện polling, chờ người chơi bấm phím. Khởi tạo giá trị hàng kiểm tra đầu tiên là 0x01 (t1). Lấy địa chỉ đầu vào (t2) và đầu ra (t3) của bàn phím digital lab sim, truyền vào địa chỉ đầu vào giá trị hàng đang kiểm tra để lấy giá trị hàng và cột kiểm tra ra từ địa chỉ đầu ra (a0) của bàn phím digital lab sim.
  + Nếu gía trị lấy từ địa chỉ đầu ra của bàn phím digital lab sim bằng 0 có nghĩa là người dùng không bấm ở hàng đấy, nhảy đến ***next\_row*** để thực hiện tăng hàng lên gấp đôi rồi quay về ***wait\_loop*** để lấy được phím bấm.
  + Nếu giá trị lấy từ địa chỉ đầu ra của bàn phím digital lab sim khác 0 có nghĩa là người dùng đã bấm nút ở hàng đấy và chương trình bắt đầu thực hiện ánh xạ sang vị trí của bitmap.
    - Thực hiện lặp để so sánh, tìm vị trí phím được lấy ra với các giá trị của mảng 2 chiều keymap, nếu phím bấm trùng hợp với giá trị hexa map vị trí các phần tử của mảng keymap thì sẽ lưu giá trị (s4) là chỉ số của vị trí đó trên bitmap ở địa chỉ &keymap[i + 4].
    - Thực hiện tăng giá trị địa chỉ trỏ tới các phần tử của mảng keymap lên 8 để truy cập đến các giá trị keymap[i] để so sánh trong lúc lặp mà chưa tìm thấy phần tử khớp với phím được bấm.
* Thực hiện kiểm tra xem phím bấm đã được nhả ra hay chưa. Nếu phím chưa được nhả ra thì ta chưa thực hiện gán màu cũng như các hoạt động sau.

### Xử lý gán màu và hiển thị màu:

* Nếu phím đã được nhả thì bắt đầu thực hiện kiểm tra vị trí đấy đã được mở hay chưa rồi gán màu và in màu ra bit map display.

|  |
| --- |
| retrieve:  la t0, Temp  slli t1, s4, 2  add t2, t0, t1  lw a3, 0(t2) # lấy màu  li t0, MONITOR\_SCREEN  add t3, t0, t1  lw t4, 0(t3)  bnez t4, already\_chosen  sw a3, 0(t3)  li t2, 0  li t5, 1  beq s3, t5, delay\_display  ret |

* Lấy địa chỉ của mảng Temp(t0), tính địa chỉ của mảng ở vị trí nút bấm (địa chỉ là 4 bytes nên nhân vị trí nút bấm đã được ánh xạ từ keymap với 4) bằng cách lấy địa chỉ của mảng Temp cộng với giá trị của 4 lần vị trí nút bấm. Ta lấy màu (a3) đã được lưu từ quá trình khởi tạo màu ban đầu ở Temp.
* Lấy địa chỉ của màn hình bit map, tính địa chỉ của vị trí tô màu trong bit map ở vị trí nút bấm (địa chỉ là 4 bytes nên nhân vị trí nút bấm đã được ánh xạ từ keymap với 4) bằng cách lấy địa chỉ của màn hình bit map cộng với giá trị của 4 lần vị trí nút bấm. Lấy màu (t4) từ vị trí đó trên bit map, thực hiện kiểm tra xem ô đấy đã được tô màu chưa.
  + Nếu màu ở vị trí đó trên bit map mà là màu đen (giá trị 0x00000000 trong bảng màu hexa) thì thực hiện gán màu (a3) vào chính vị trí đó trên bit map và gán cờ báo lỗi (t2) bằng 0. Kiểm tra nếu đã bấm 1 lần trong lượt chơi thì nhảy đến ***delay\_display*** để trì hoãn hiển thị.

|  |
| --- |
| delay\_display:  li a7, 32  li a0, 1000 # delay 1000ms  ecall  ret |

* + - Sau khi thực hiện trễ thì nhảy ngược trở về câu lệnh sau khi gọi chương trình con ***retrieve.***
  + Nếu màu ở vị trí đó trên bit map mà không là màu đen thì nhảy đến ***already\_choosen*** rồi hiển thị thông báo lỗi.

|  |
| --- |
| already\_chosen:  li t2, 1  ret |

* + - Sau khi thực hiện gán cờ báo lỗi (t2) bằng 1 thì nhảy ngược trở về câu lệnh sau khi gọi chương trình con ***retrieve.***
* Nếu cờ báo lỗi (t2) mà khác 0 thì nhảy đến chương trình con ***invalid\_input*** để báo lỗi.

|  |
| --- |
| invalid\_input:  li a7, 55  la a0, error  li a1, 0  ecall  j game\_loop |

* Nếu cờ báo lỗi (t2) mà bằng 0 thì thực hiện so sánh số lần màu đó đã được bấm trong một lượt chơi (s3) với 0.
  + Nếu số lần đã bấm trong một lượt chơi bằng 0 thì thực hiện nhảy đến ***save\_first*** để lưu các thông số về màu đó ở lần bấm đầu tiên này.

|  |
| --- |
| save\_first:  la t0, FirstIdx  sw s4, 0(t0)  la t0, FirstCol  sw a3, 0(t0)  li s3, 1  j game\_loop |

* + - Lấy địa chỉ của ***FirstIdx*** để lưu giá trị vị trí đã được ánh xạ vào địa chỉ này rồi lấy địa chỉ của ***FirstCol*** để lưu giá trị màu ở vị trí được ánh xạ vào địa chỉ này và gán số lần đã bấm trong một lượt chơi =1.
  + Nếu số lần đã bấm trong một lượt chơi khác 0 (bằng 1) thì thực hiện nhảy đến ***check\_match*** để so sánh cặp màu trong lượt chơi.

|  |
| --- |
| check\_match:  la t0, FirstCol  lw t1, 0(t0)  xor t2, t1, a3  li s3, 0  beqz t2, matched  j not\_matched |

* + - Lấy màu được lưu từ lần đầu bấm (t1) từ địa chỉ ***FirstCol*** rồi dùng phép toán xor của màu đấy với màu vừa bấm. Nếu hai màu giống nhau thì giá trị của kết quả sẽ là 0 còn ngược lại là khác 0. Sau đó reset lại số lần bấm nút của người dùng. So sánh kết quả với 0.
      * Nếu bằng 0 thì nhảy đến nhãn ***match*** để tăng số cặp giống lên 1 đơn vị rồi quay lại lặp lấy đầu vào.

|  |
| --- |
| matched:  addi s0, s0, 1  j game\_loop |

* + - * Nếu khác 0 thì nhảy đến nhãn ***not\_match*** để xóa màu từ vị trí trước và xóa thẻ vị trí rồi nhảy lặp lại vòng lặp.

|  |
| --- |
| not\_matched:  li t0, MONITOR\_SCREEN  slli t1, s4, 2  add t1, t0, t1  sw zero, 0(t1) # xóa màu hiện tại  la t2, FirstIdx  lw t3, 0(t2)  slli t4, t3, 2  add t4, t0, t4  sw zero, 0(t4) # xóa thẻ trước đó  j game\_loop |

### Thắng trò chơi:

|  |
| --- |
| game\_done:  li a7, 55  la a0, finish  li a1, 1  ecall  li a7, 10  ecall |

* Khi mà số cặp giống nhau bằng số cặp màu thì trò chơi kết thúc, chương trình nhảy đến nhãn ***game\_done*** để in ra màn hình thông báo chiến thắng và kết thúc trò chơi.

## Kết quả chạy mẫu:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.