**Computer Architecture**

**Final exam**

**Full name:** Ngo Vinh Khanh

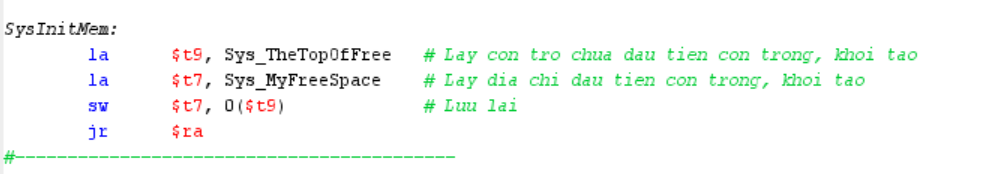
**Student ID:** 20215211

**Full name:** Nguyen Dinh Trung

**Student ID:** 20215249

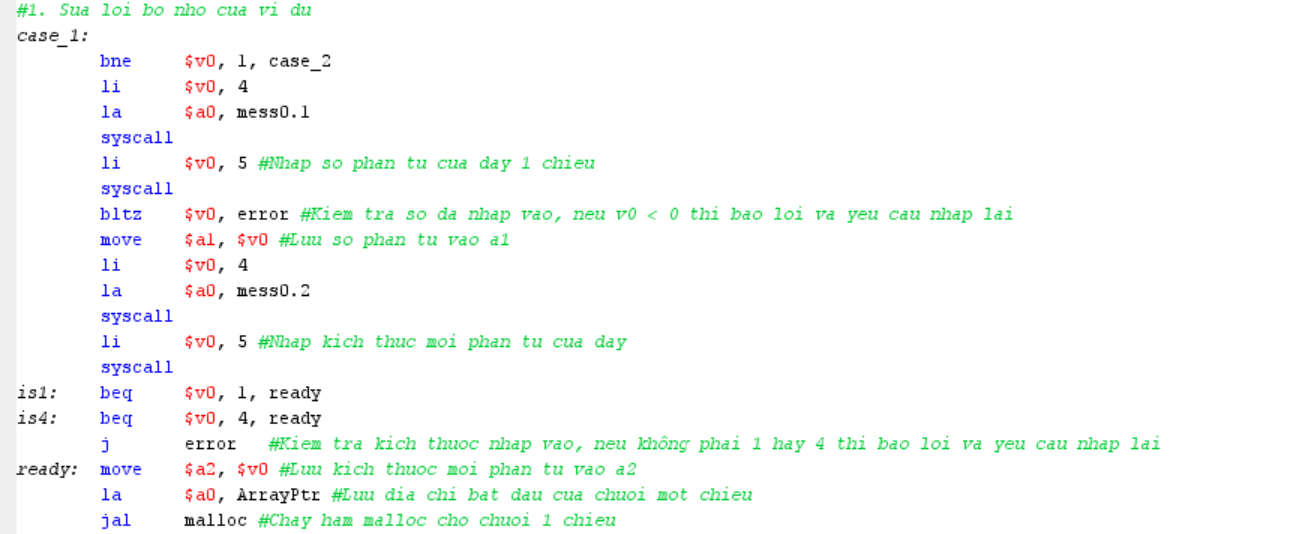
Assignment 6:

Phân tích code và kết quả:

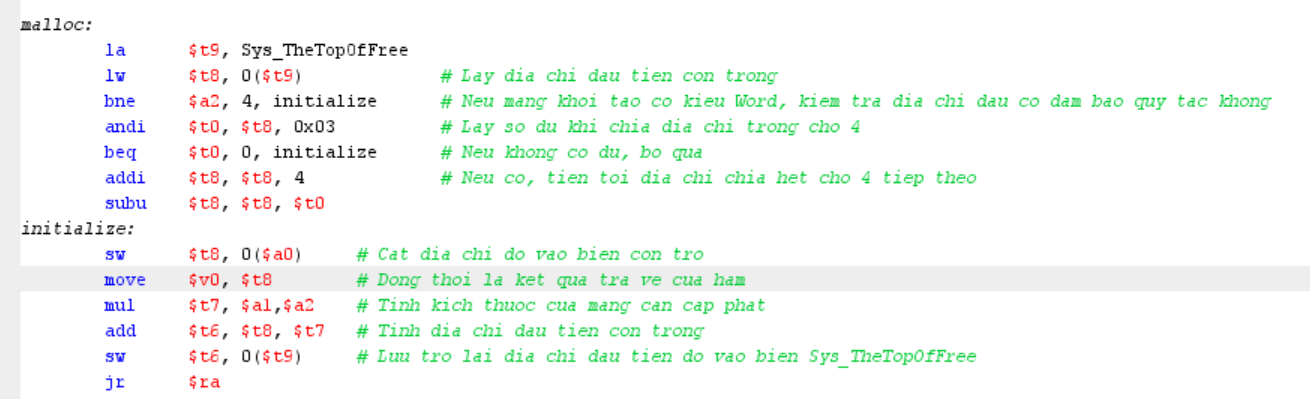


Hàm SysInitMem, bắt đầu gắn địa chỉ còn trống đầu tiên vào t7, sau đó lưu địa chỉ đó vào bộ nhớ của t9.

1. Sửa lỗi không chia hết cho 4 của kiểu word trong ví dụ:

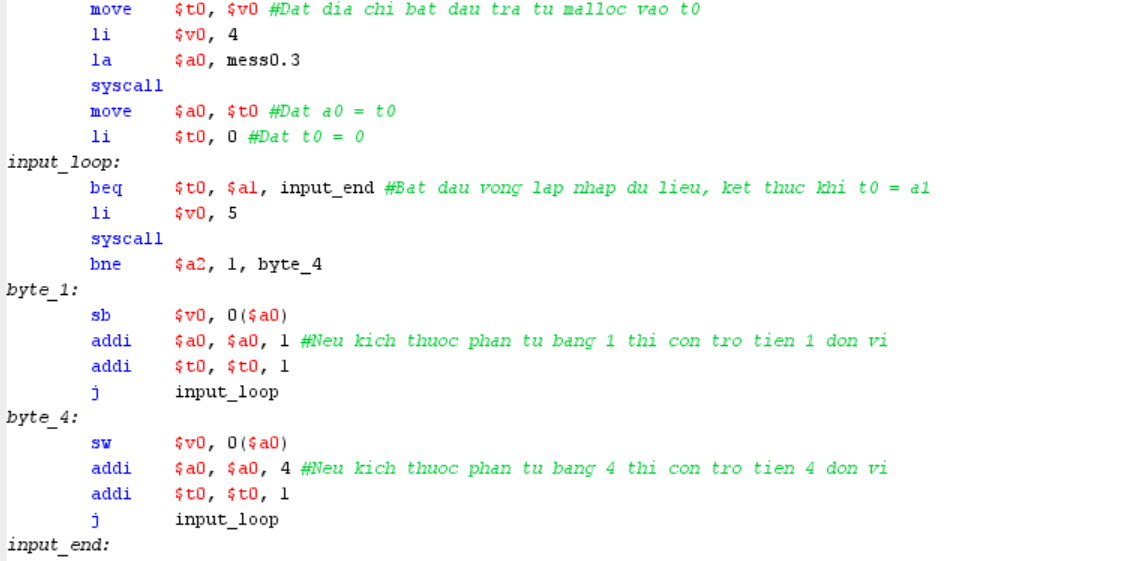


Kiểm tra số phần tử và kích thước phần tử của dãy, nếu không thỏa mãn số phần tử lớn không hay kích thước phần tử bằng 1 hoặc 4 thì báo lỗi, còn không thì di chuyển đến hàm malloc.



Hàm malloc, trước hết gán địa chỉ bắt đầu của bộ nhớ còn trống vào biến con trỏ t8. Nếu mảng khởi tạo là kiểu word (kích thước phần tử bằng 4 byte) thì kiểm tra xem con trỏ có thỏa mãn quy tắc. Đặt t0 là số dư của con trỏ cho 4. Nếu không dư thì bỏ qua còn nếu có thì tiến tới cho đến khi địa chỉ chia hết cho 4.

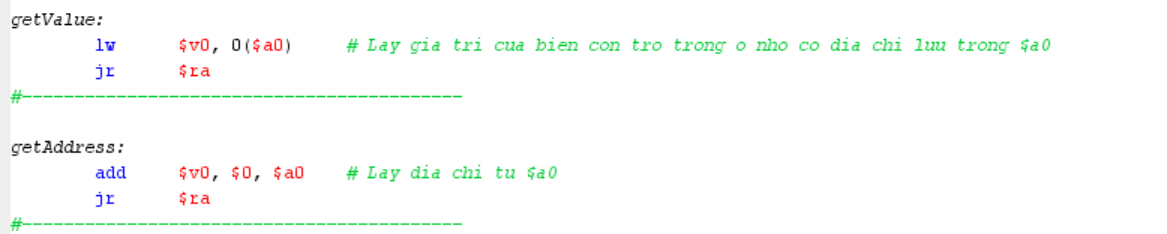
Lưu địa chỉ đã qua kiểm tra vào biến con trỏ. Tính kích thước của mảng cần cập nhật rồi khởi tạo lại địa chỉ bắt đầu của bộ nhớ còn trống rồi quay về.



Nhập dữ liệu, với kích thước 1 byte, con trỏ tiến 1 đơn vị mỗi vòng lặp, với kích thước 4 byte, con trỏ tiến 4 đơn vị mỗi vòng lặp.

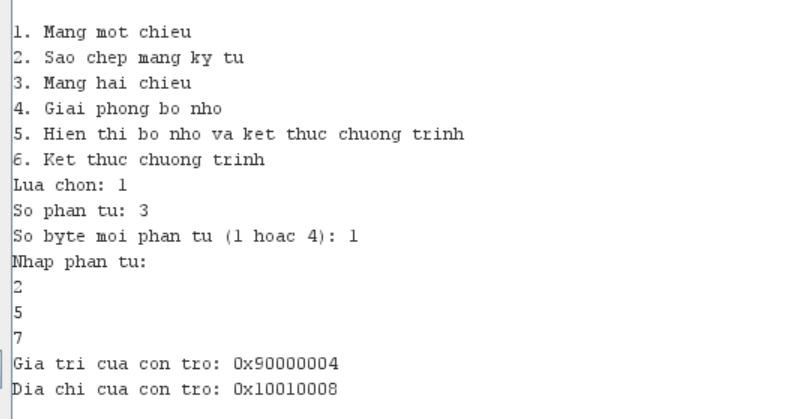
2. + 3. Viết hàm lấy giá trị của biến con trỏ + Viết hàm lấy địa chỉ biến con trỏ.



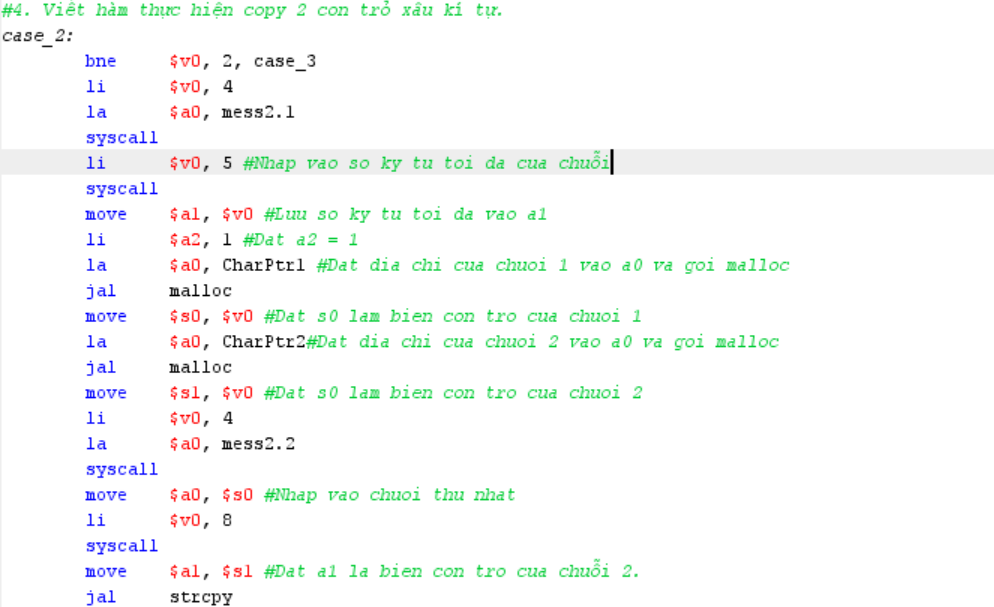


Hai hàm getValue và getAddress lần lượt lấy giá trị lưu trong a0 và địa chỉ của a0.

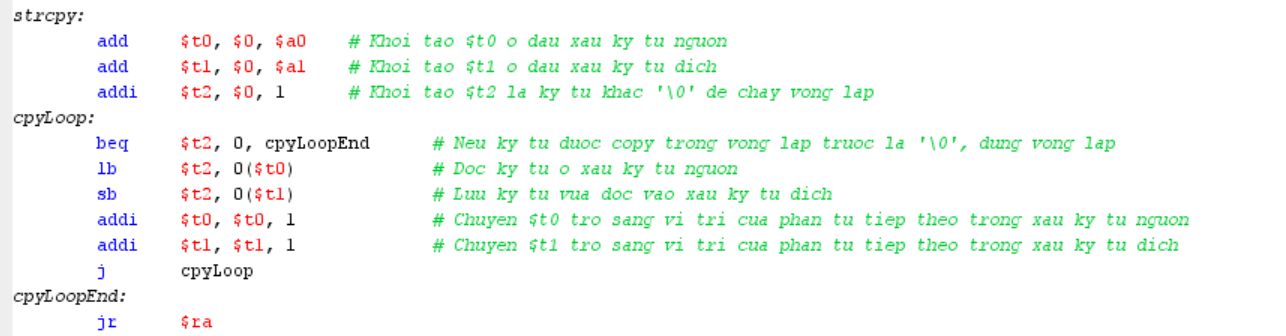
Kết quả:



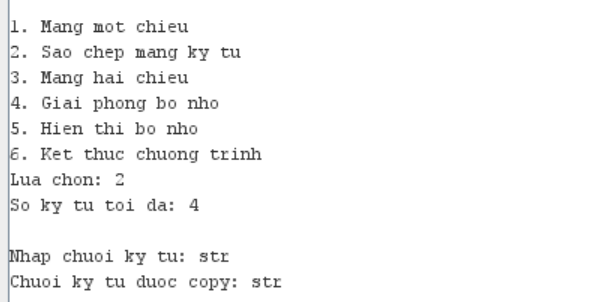
1. Viết hàm thực hiện copy 2 con trỏ xâu kí tự.



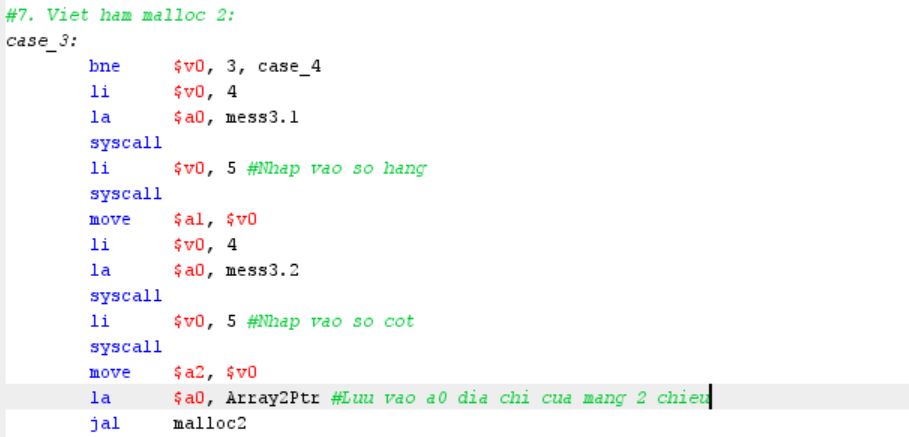
Trước hết, nhập vào số ký tự tối đa của các chuỗi, lưu vào trong a1. Đặt a2 = 1. Cấp phát bộ nhớ cho 2 chuỗi bằng hàm malloc sau đó đặt lại a0 là biến con trỏ chuỗi 1 rồi nhập vào chuỗi 1 và đặt lại a1 là biến con trỏ chuỗi 2.



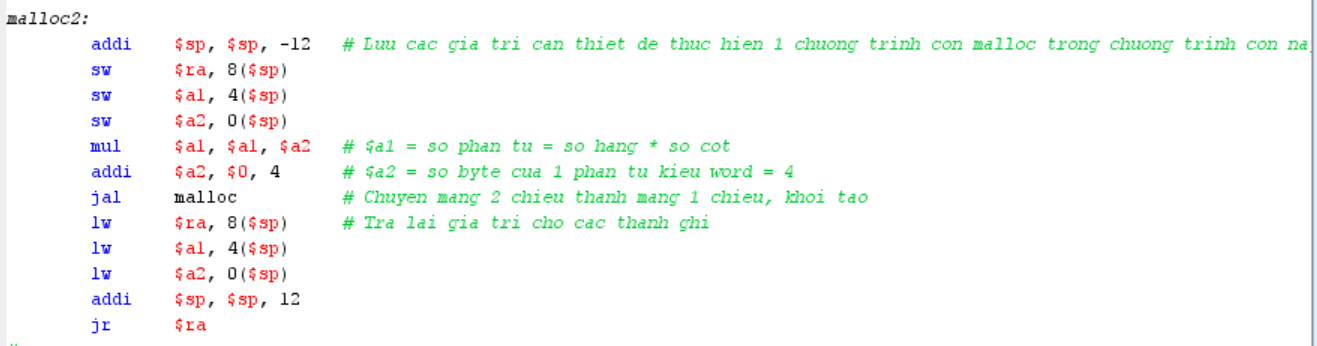
Hàm strcpy chạy vòng lặp để sao chép chuỗi 1 sang chuỗi 2 thông qua vòng lặp.



1. Malloc 2:

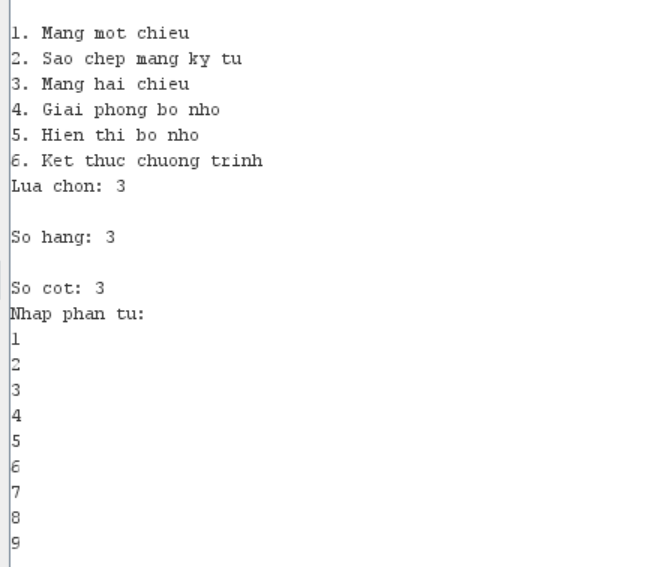


Nhập vào số hàng và số cột của mảng 2 chiều, được lưu ở a1 và a2.



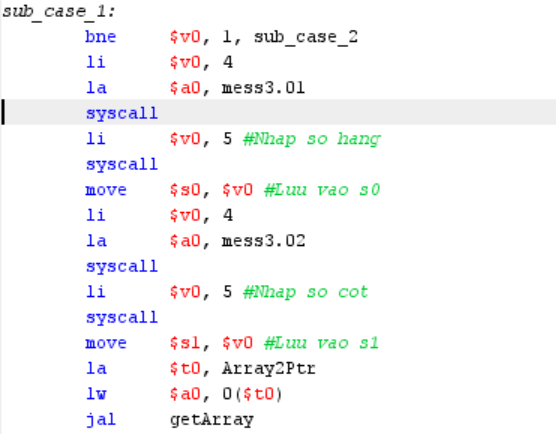
Hàm malloc2, trước hết lưu lại giá trị của ra, số hàng, số cột để chạy được hàm con malloc. Số phần tử trong mang sẽ là hàng x cột, lưu tại a1. Coi kích thước của mỗi phần tử là 4 (kiểu word). Sử dụng hàm malloc với các thông số như trên, biến mảng 2 thành 1 chiều rồi trả lại các giá trị của thanh ghi.

Kết quả:

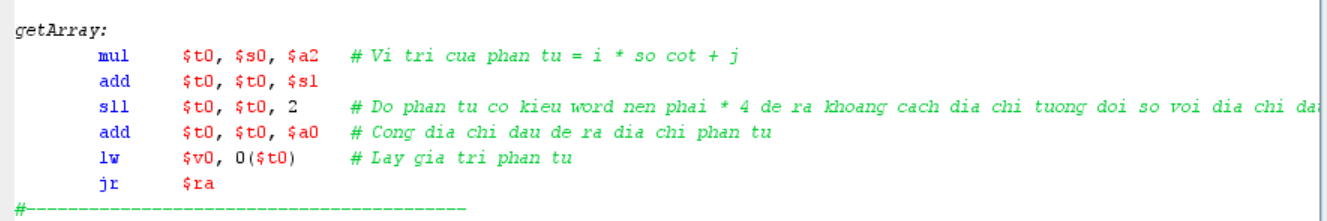


8.

getArray:

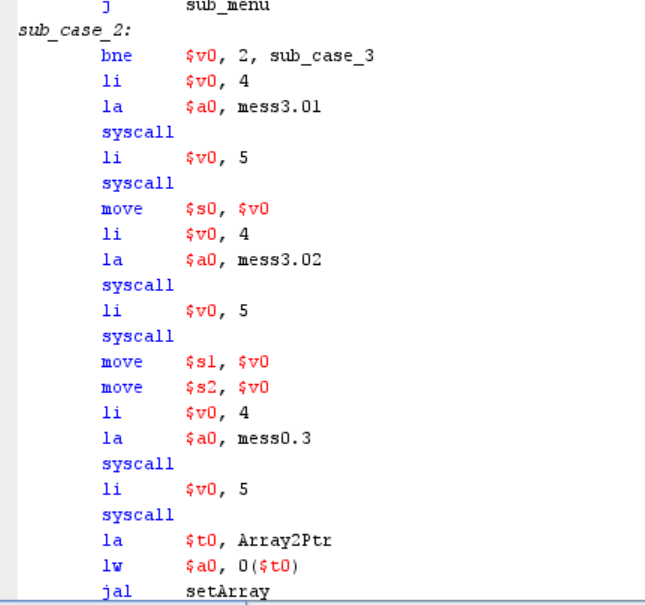


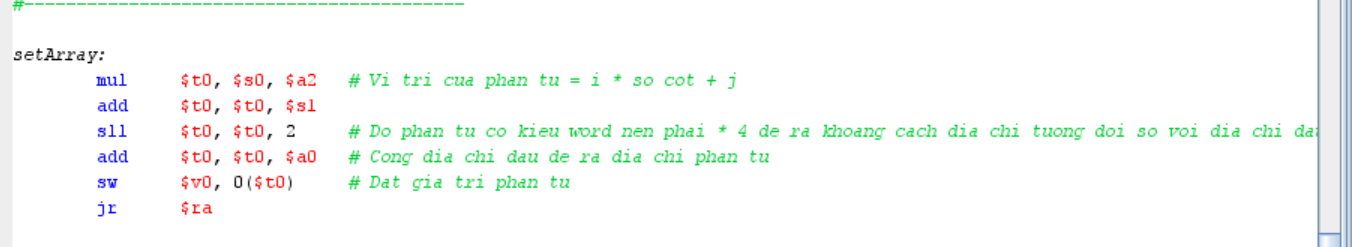
Nhập vào số hàng và số cột.



Hàm getArray, phần tử thứ A[i][j] của mảng 2 chiều là phần tử thứ A[i\*số cột + j] của mảng 1 chiều. Do mỗi phần tử cách nhau 4 byte nên phải nhân 4 để tính khoảng cách từ vị trí đầu.

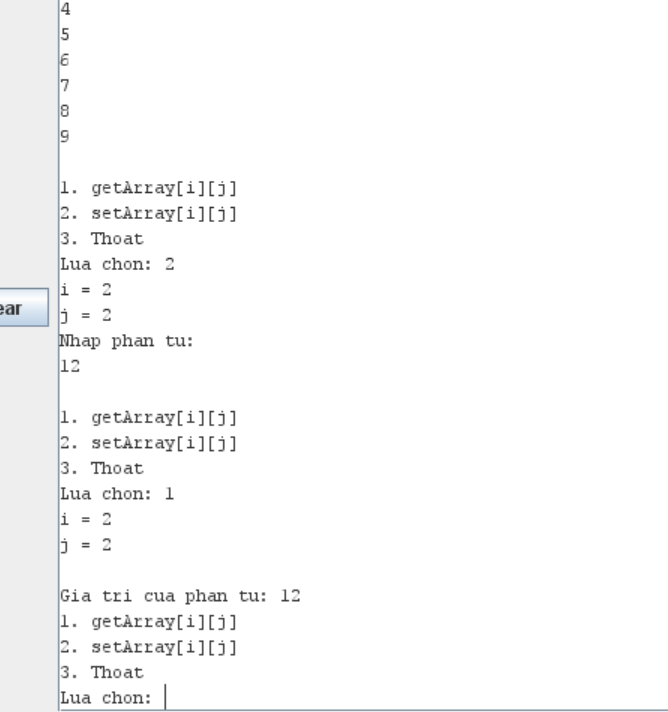
setArray:



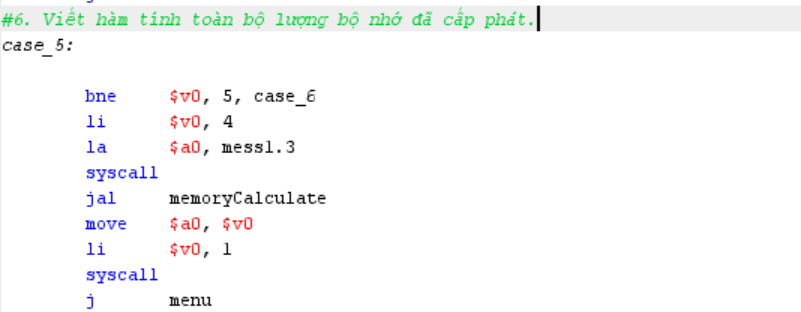


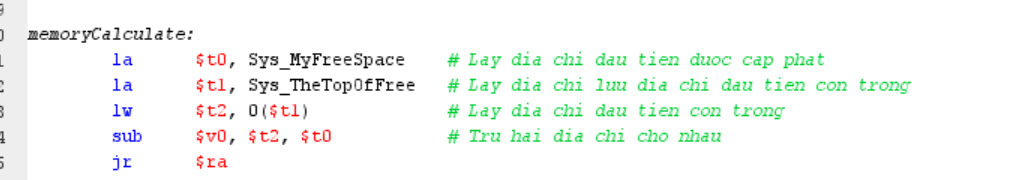
Dùng cách tương tự với getArray để tìm được địa chỉ muốn tìm, sau đó thay vì in ra, ta lưu giá trị mới nhập vào.

Kết quả:



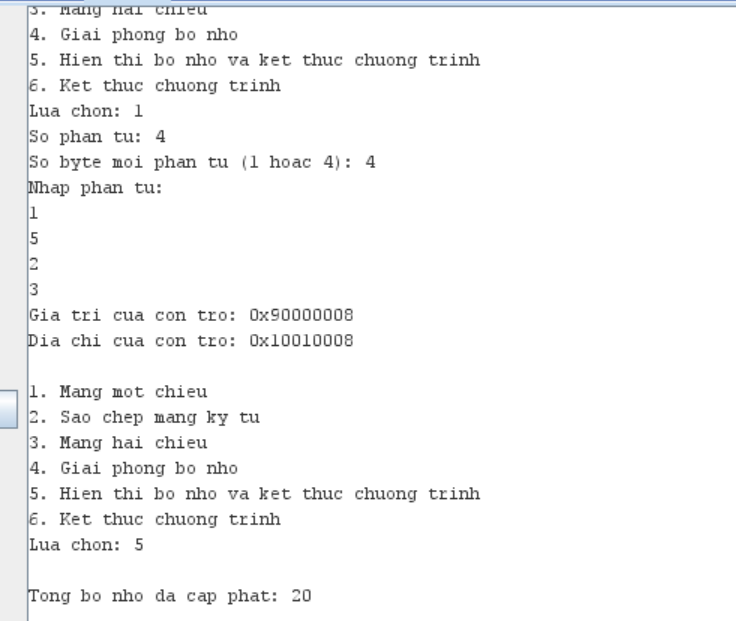
1. Tính lượng bộ nhớ đã cấp phát



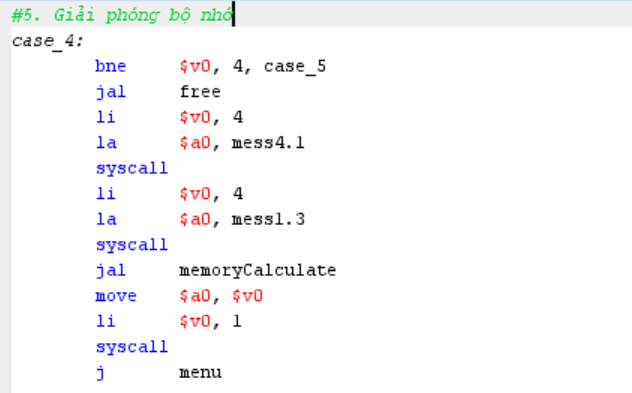


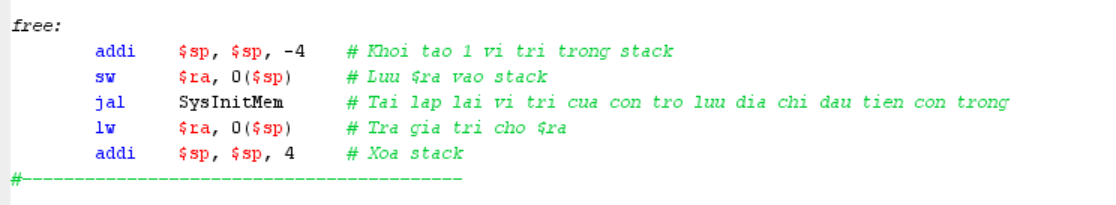
Ta dùng địa chỉ của vị trí trống đầu mới nhất trừ đi vị trí trống đầu ban đầu.

Kết quả:



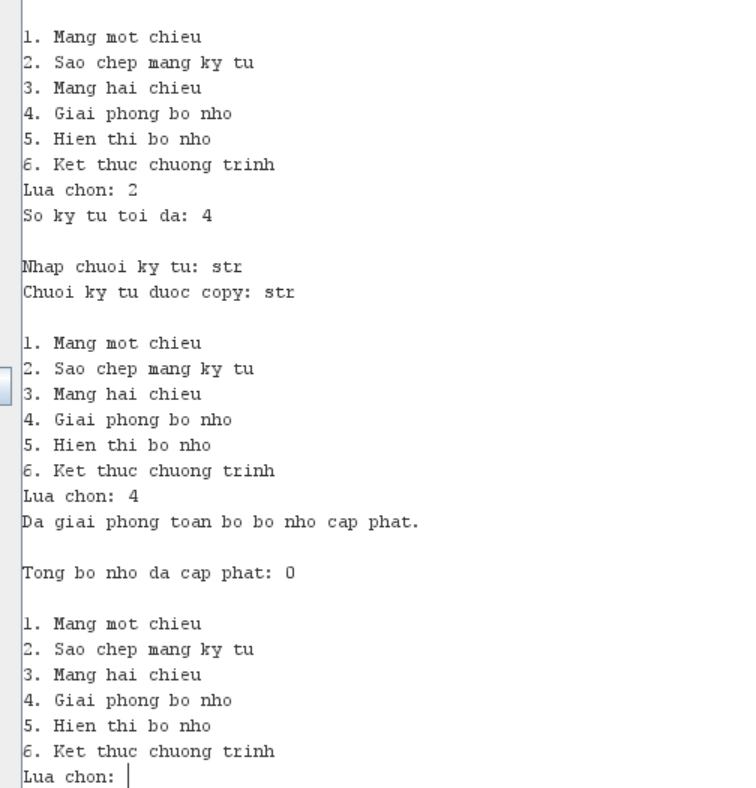
1. Giải phóng bộ nhớ:





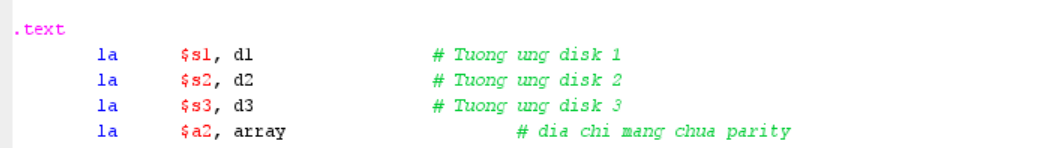
Chạy hàm SysInitMem bên trong để tái lập vị trí của con trỏ lưu địa chỉ đầu tiên còn trống.

Kết quả:

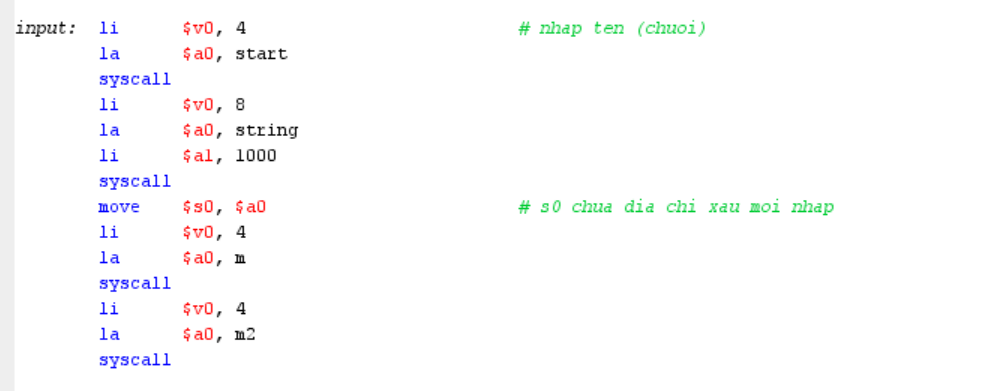


Assignment 8:

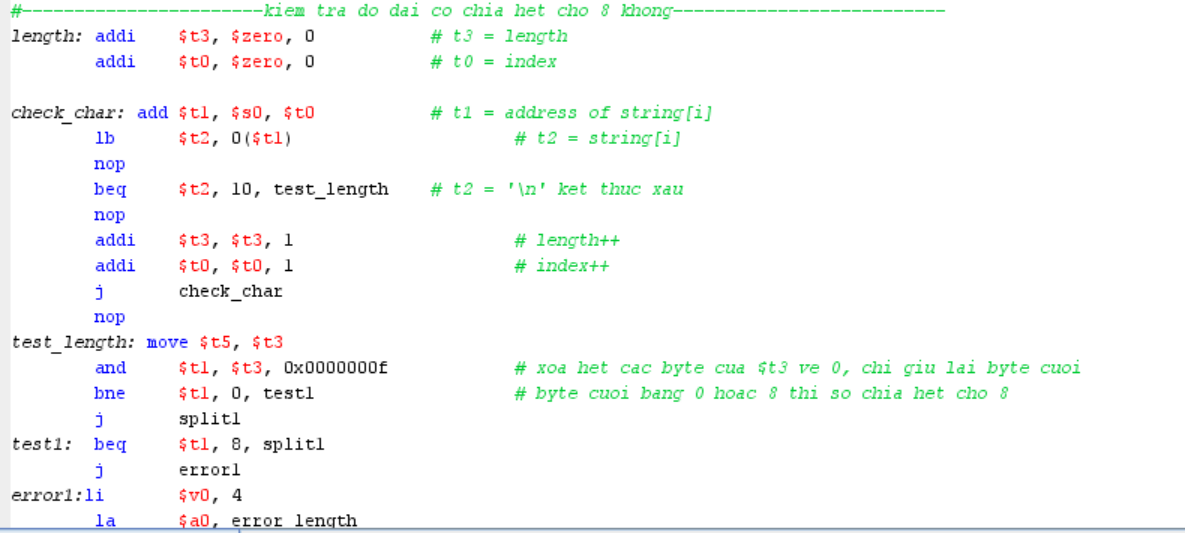
Phân tích code và kết quả:



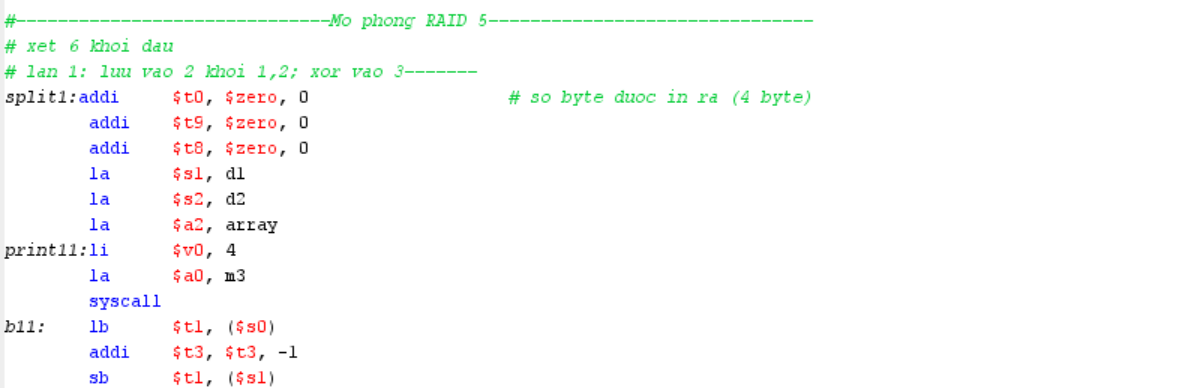
Khởi tạo đĩa 1, đĩa 2, đĩa 3 và mảng chứa parity vào s1, s2, s3 và a2



Nhập chuỗi vào. Lưu địa chỉ của chuỗi vào s0

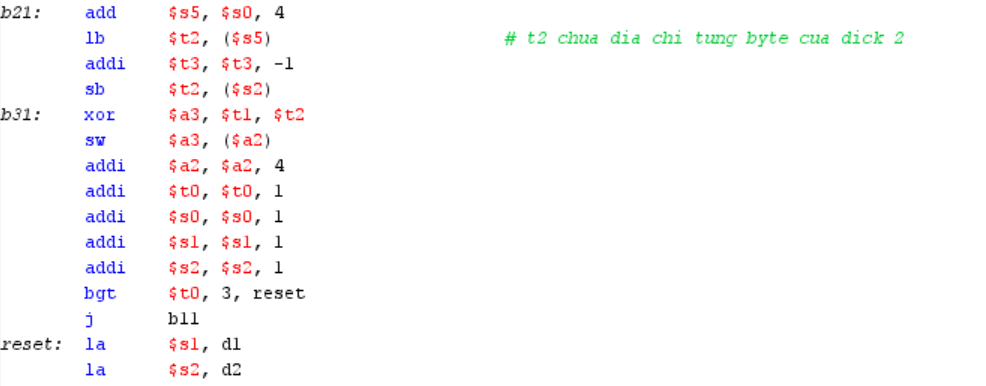


Kiểm tra độ dài chuỗi mới nhập. Dùng check\_char để tính độ dài của chuỗi qua vòng lặp rồi sau đó xóa hết các byte của t3 trừ byte cuối để kiếm tra xem t3 có bằng 0 hoặc 8 hay không. Nếu không thì báo lỗi và nhập lại.



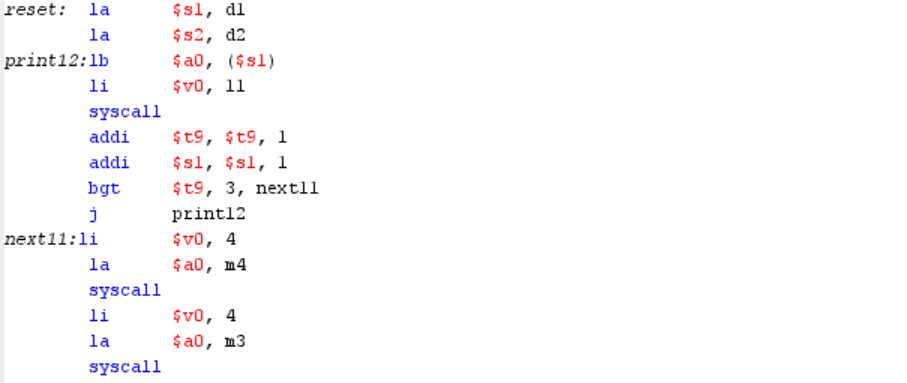
Khởi tạo t0, t9, t8, đĩa 1, 2 và mảng chứa parity.

Bắt đầu vòng lặp, lấy giá trị đầu tiên từ chuỗi rồi lưu vào đĩa 1.

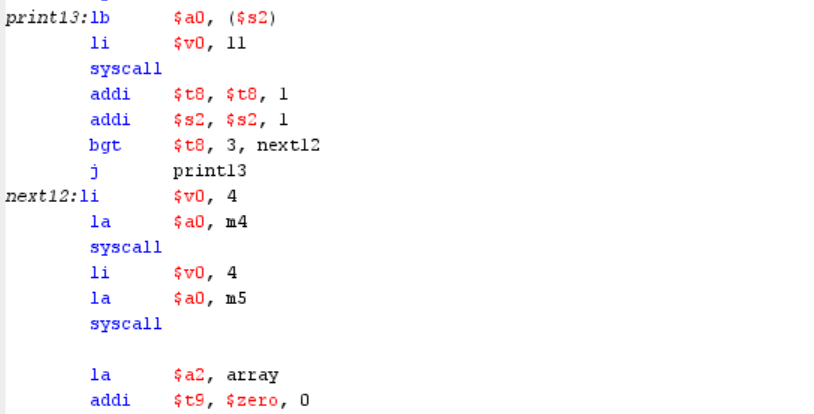


Tiến con trỏ của chuỗi lên 4 đơn vị (do mỗi đĩa chứa 4 byte) và bắt đầu lưu giá trị tại vị chí đó vào chuỗi.

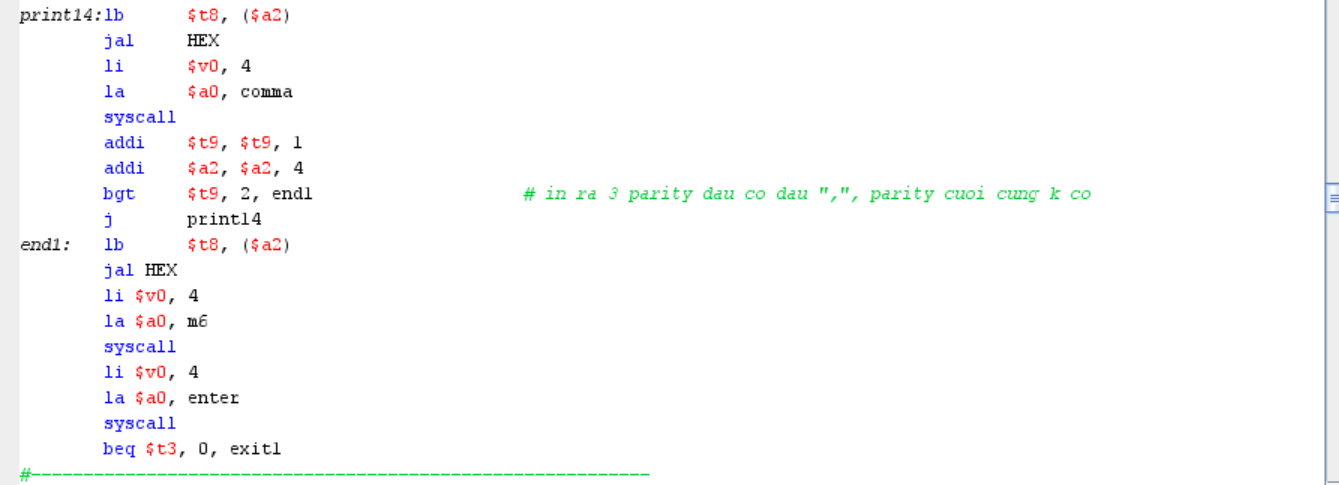
Tính giá trị của xor a3, t1, t2 sau đó lưu vào trong mảng chứa parity đồng thời tiến index lên 1 đơn vị. Lặp lại cho đến khi index = 3.

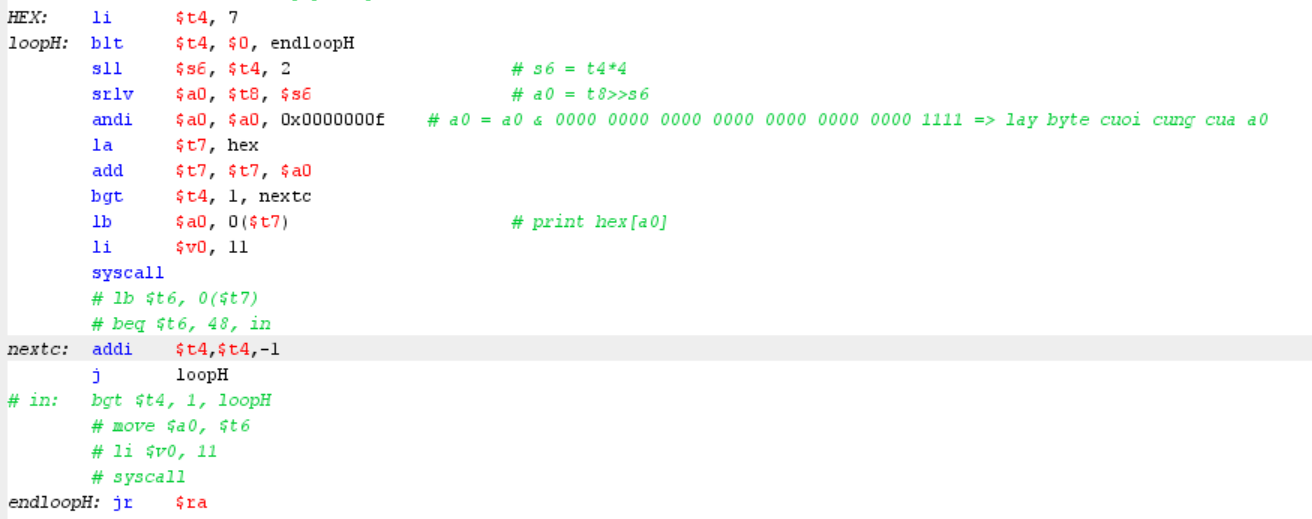


Đặt lại s1 và s2 về đầu địa chỉ của đĩa. In ra màn hình các giá trị của đĩa 1.

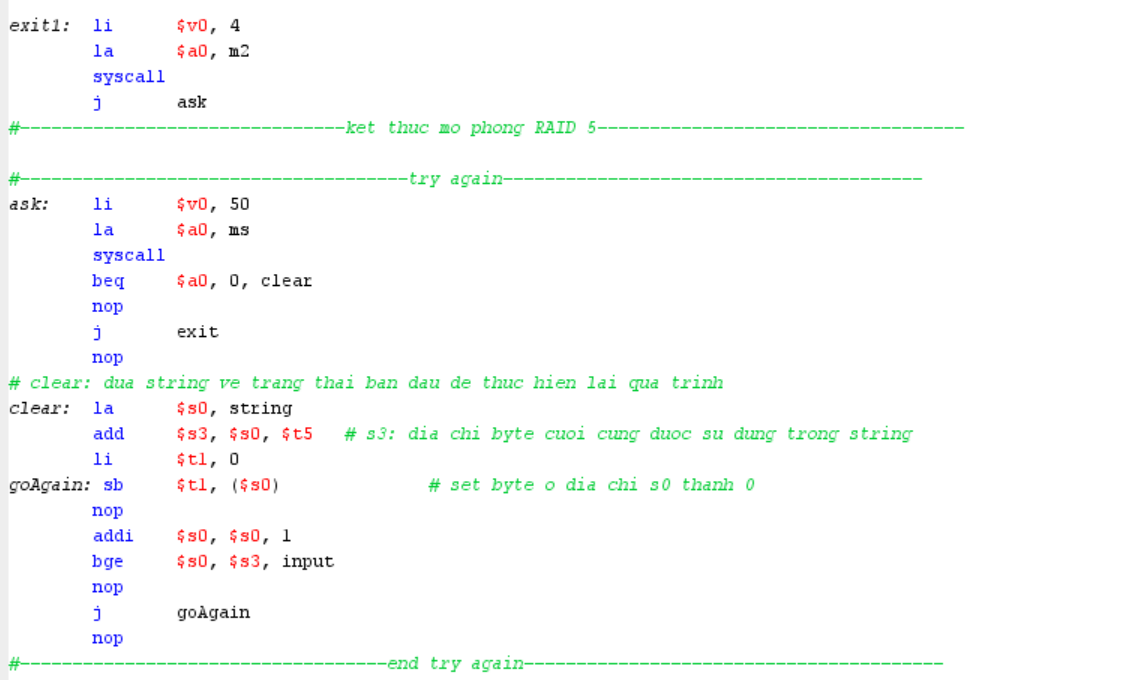


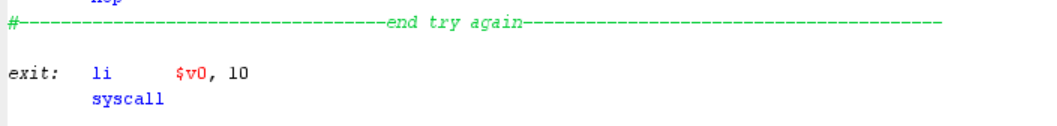
Tương tự, in ra màn hình các giá trị của s2.





Sau đó, ta gọi hàm HEX để in ra kết quả của mảng chứa parity dưới dạng hex. Kiểm tra xem nếu toàn bộ chuỗi đã được in hết ra chưa. Rồi thì hỏi lại người dùng có muốn tiếp tục, nếu có thì đưa làm mới chuỗi nhập vào, không thì kết thúc.





Nếu chuỗi chưa kết thúc ta tiến hành tìm và in tiếp các khối sau. Ta làm tương tự như khối một nhưng ở khối 2 thì sẽ là khởi tạo đĩa 1, 3 còn khối 3 sẽ là đĩa 2, 3. Sau khi kết thúc khối thứ 3 mà chưa duyệt hết chuỗi thì ta quay lại chuỗi một.

Kết quả:

