**BÁO CÁO FINAL PROJECT**

**Học phần: Thực hành kiến trúc máy tính**

Học kì: 2020-2

Nhóm: 18

*Giảng viên hướng dẫn: Lê Bá Vui*

*Sinh viên 1: Nguyễn Viết Huy – 2018 - Mã đề 07*

*Sinh viên 2: Vũ Công Minh - 20184155 - Mã đề 08*

**Assignment7:**

*- Người thực hiện: Nguyễn Viết Huy*

*- Cách thực hiện:*

+ khởi tạo danh sách opcode kèm với cú pháp tương ứng

Kí hiệu :

1 thanh ghi

2 số nguyên

3 nhãn

4 ‘(‘

5 ’)’

6 imm

x là kí hiệu kết thức xâu

Các bước thực hiện:

1 đọc xâu vào + tiền xử lý

2 thực hiện đọc opcode và tìm cú pháp tương ứng

3 bắt đầu thực hiện kiểm tra cú pháp (readtoken và kiểm tra token có đúng với token tương ứng trong cú pháp không nếu sai báo lỗi và kết thúc)

4 kiểm tra đọc được kí hiệu kết thúc ‘x’ thành công thì đưa ra thông báo nếu ko quay lại bước 3

Các function :

Read\_token : đọc lần lượt các token

Compare\_string : so sánh 2 chuỗi

Make\_regisster : kiểm tra token hiện tai có phải thanh ghi

Make\_number : kiểm tra token có phải số

Make\_index : kiểm tra chỉ số

Make\_label : kiểm tra nhãn

Skip : bỏ qua ‘ ’ và ’,’

Read\_char : đọc từng kí tự

Biến command lưu xâu cần xử lý

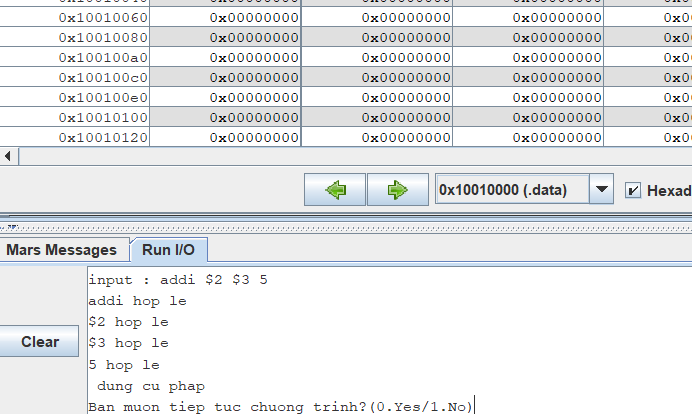
Biến syntax lưu cú pháp của opcode

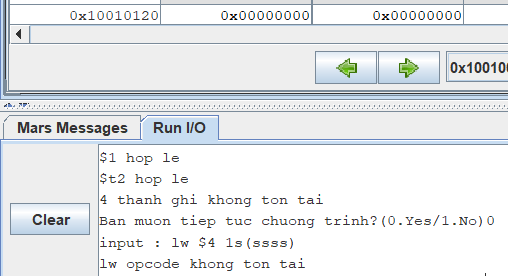
Biến flag xử dụng để lưu kết quả trả về trong các hàm

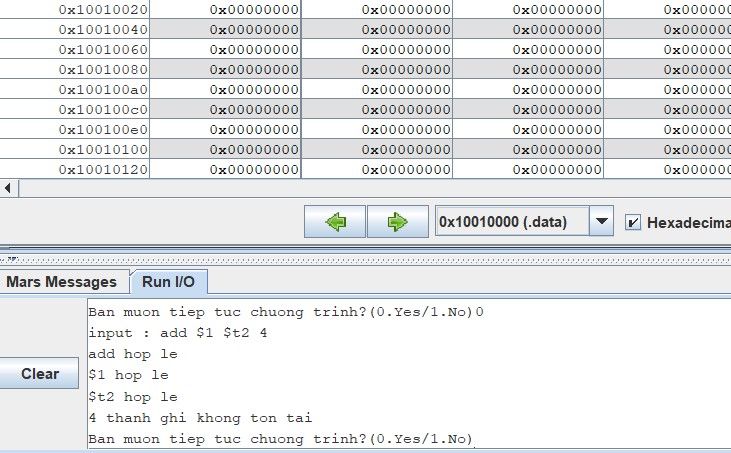
Thanh ghi $s7 lưa địa chỉ command

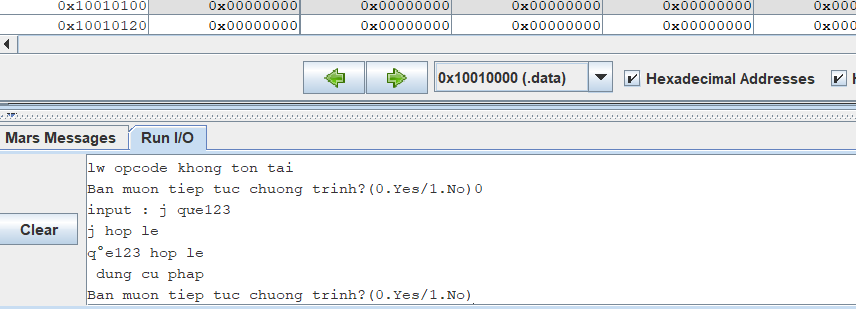
Thanh ghi $s1 lưa địa chỉ syntax

Demo kết quả:



**

****

****

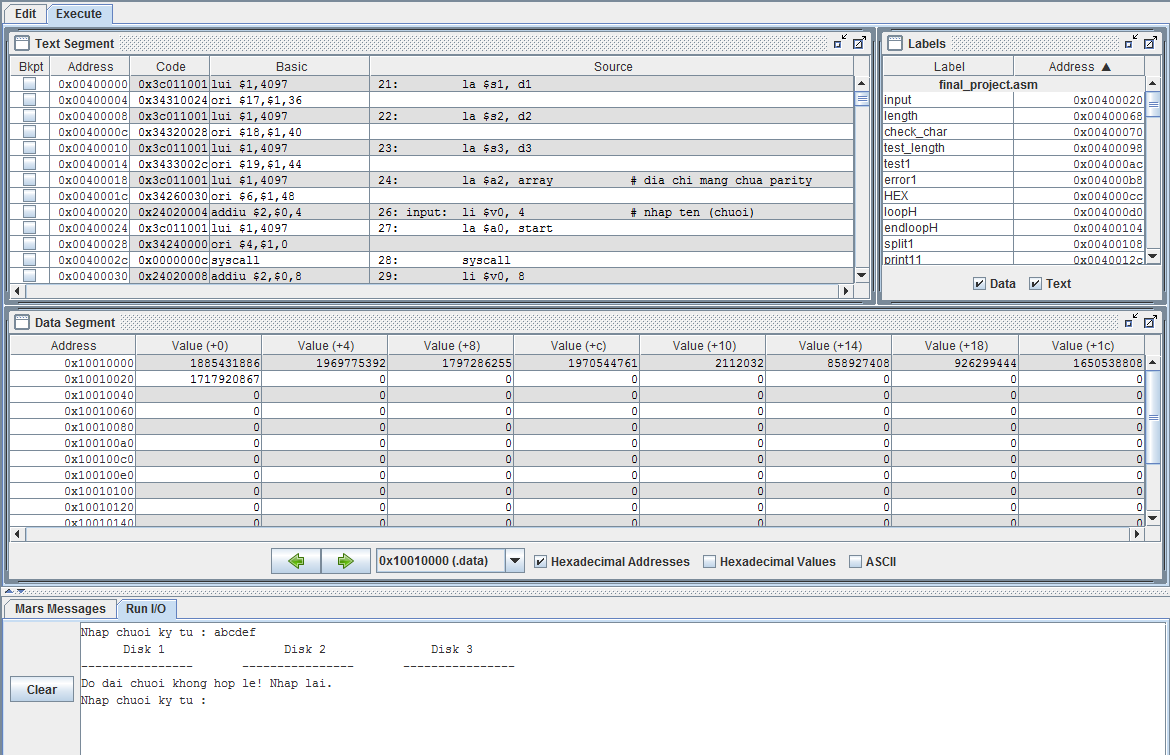
**Assignment8:**

*- Người thực hiện: Vũ Công Minh*

*- Cách thực hiện:*

+ Khởi tạo các nhãn start, error\_length, comma, enter, ms để in ra các thông báo; các nhãn m, m1, m2, m3, m4, m5, m6 để in viền của các đĩa; mảng hex để in mã parity; mảng d1, d2, d3 theo thứ tự chứa dữ liệu đĩa 1,2,3; mảng array lưu mã parity và mảng string lưu xâu được người dùng truyền vào từ bàn phím.

+ Yêu cầu người dùng nhập xâu dữ liệu và lưu vào mảng string, kiểm tra nếu người dùng nhập xâu kí tự không là bội của 8 thì yêu cầu nhập lại. Ở bước này sử dụng một biến đếm để đếm độ dài của xâu, sau đó lấy bit cuối của thanh ghi biến đếm ấy, nếu là “0” hoặc “8” thì mới thỏa mãn điều kiện. Khi không thỏa mãn sẽ in ra màn hình error\_length (như hình)



+ Sau khi nhập input thành công thì chương trình sẽ chia xâu input thành từng block 8 kí tự. Label “split1” đến trước “split2” thực hiện việc chia 4 kí tự đầu của block để lưu vào đĩa 1, 4 kí tự kế tiếp lưu vào block 2, thực hiện phép XOR dữ liệu của 2 đĩa 1 và 2 và lưu vào đĩa 3. Việc hiển thị mã parity của đĩa 3 được thực hiện trong function “HEX”. Thực hiện tương tự với các nhãn “split2” và “split3”.

+ Function “HEX” thực hiện việc lấy parity từ mảng array và in parity ra đĩa tương ứng. Đầu tiên lấy 4 bit gần cuối của 4 byte parity trong mảng, in ra kí tự có mã ASCII tương ứng được lưu trong mảng hex. Tiếp tục lấy 4 bit cuối cùng của 4byte mảng array và in kí tự tương ứng trong mảng hex ra đĩa.

+ Khi in mã parity ra đĩa thì cần gọi 4 lần hàm HEX, 3 lần đầu in thêm dấu “,” (comma), lần cuối cùng không in dấu “,”. Mỗi lần gọi HEX thì cần tăng thanh ghi chứa địa chỉ mảng array lên 1 byte (do hàm HEX lấy và in ra 2 kí tự tương ứng với 8 bit cuối của mảng).

+ Sau khi thực hiện in toàn bộ dữ liệu của đĩa, sẽ gọi đến nhãn “ask” để hỏi người dùng có muốn tiếp tục chương trình không, nếu có sẽ thực nhiện nhãn “clear” để reset trạng thái của mảng lưu và nhãn “goAgain” để bắt đầu lại từ bước nhập input. Nếu chọn không sẽ gọi đến nhãn “exit” và kết thúc chương trình.

*- Ý nghĩa các thanh ghi:*

+ $s0: địa chỉ mảng string (xâu input)

+ $s1, $s2, $s3: tương ứng là địa chỉ mảng disk1,2,3

+ $a2: địa chỉ mảng array (chứa mã parity)

+ $t3, $t5: độ dài xâu input

+ $t0: biến chạy i

+ $t1: địa chỉ string[i]; còn được sử dụng để lấy bit cuối khi xét tính hợp lệ của độ dài xâu

+ $t2: string[i]

+ $t4: biến chạy trong hàm HEX

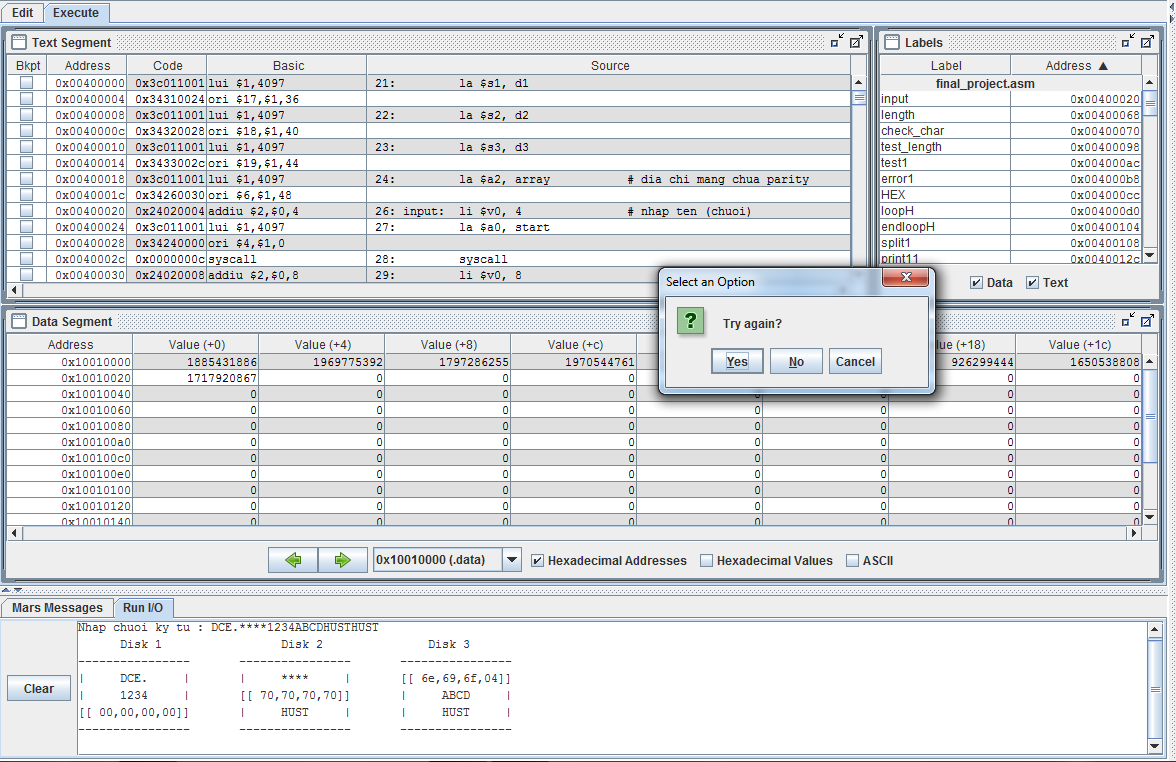
+ $t6: sử dụng khi dịch chuyển các bit của mảng parity để in ra đĩa

+ $t7: địa chỉ mảng hex

+ $t8: địa chỉ mảng array khi gọi hàm HEX

+ $t9: đếm số byte đã in

*- Kết quả thu được:*

**