**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘIVIỆN CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**- - - - - - - - - - - - - - - -

**BÁO CÁO BÀI TẬP**

**MÔN KIẾN TRÚC MÁY TÍNH**

**Họ và tên: Đào Xuân Minh.MSSV: 20205100.**

**Bài tập số 7.**

1. **Công việc:**

**7. Chương trình kiểm tra cú pháp lệnh MIPS**

Trình biên dịch của bộ xử lý MIPS 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 MIPS 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.*- Cho biết lệnh hợp ngữ đó cần bao nhiêu chu kì thì mới thực hiện xong.  
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, số chu kì thực hiện.

**II. Nội dung bài làm**

1. **Cách thực hiện chương trình:**

• Yêu cầu người dùng nhập vào một câu lệnh mips  
• Một câu lệnh hợp ngữ gồm 4 phần: mã opcode, toán hạng 1, toán hạng 2, toán hạng 3. Quy ước các dạng của toán hạng: 0 – null, 1 – register, 2 – constant, 3 – label.  
• Tách một thành phần của chuỗi tương ứng với thành phần của câu lệnh rồi so sánh với kiểu dữ liệu quy ước. Nếu phù hợp thì tiếp tục so sánh các phần phía sau. Nếu không phù hợp thì báo không hợp lệ  
+ Đầu chương trình, khởi tạo thư viện opcode chuẩn, opcode sau khi cắt được thì sẽ so sánh với thư viện đã tạo  
+ Nếu không có opcode phù hợp -> báo opcode không hợp lệ và exit. Nếu có opcode hợp lệ -> tìm khuôn dạng các toán hạng phù hợp với opcode. Đầu chương trình khi tạo chuỗi opcode chuẩn thì sẽ tạo một chuỗi khuôn dạng tương ứng với từng lệnh bằng cách so vị trí lệnh và khuôn lệnh. Nếu opcode phù hợp thì sẽ lấy được khuôn dạng lệnh và tiến hành kiểm tra lần lượt các toán hạng. Nếu có 1 toán hạng không hợp lệ -> báo ra màn hình run I/O và exit  
+ Tương tự opcode chuẩn thì cũng có một chuỗi register chuẩn được tạo ở đầu chương trình. Nếu toán hạng có mã là 1 -> nó là thanh ghi và sẽ so sánh với chuỗi các regisster chuẩn.  
+ Trường hợp toán hạng là constant: một constant hợp lệ có thể có kí tự đầu là dấu trừ ‘- ’ hoặc dấu cộng ‘+’, các ký tự phía sau bắt buộc phải từ 0 đến 9.

+ Trường hợp toán hạng là label: label hợp lệ có thể có kí tự đầu là dấu gach dưới ‘\_’ hoặc chữ thường, chữ hoa, nếu có các ký tự tiếp theo, thì các ký tự này bắt buộc phải là chữ cái, chữ số hoặc dấu gạch  
dưới.  
+ Trường hợp toán hạng null: kiểm tra xem chuỗi vừa cắt được có kí tự nào hay không, nếu có -> không hợp lệ, nếu không có kí tự nào -> hợp lệ và thông báo câu lệnh hợp lệ -> Thoát chương trình.

• Sau khi kiểm tra xong 3 toán hạng, chương trình sẽ kiểm tra xem còn kí tự nào khác hay không. Nếu còn -> câu lệnh không hợp lệ. Và báo kết quả của câu lệnh vừa nhập -> Hỏi người dùng có muốn kiểm tra tiếp một câu lệnh khác hay không?  
• Nếu người dùng chọn không -> Kết thúc chương trình.

1. **Mô tả các hàm:**
   1. **cutComponent:** Là hàm được chuyền đầu vào là các tham số như địa chỉ cơ sở của tmp và 1 biến để lưu vị trí muốn cắt tiếp theo từ xâu đầu vào.

Khi gặp phải các dấu khoảng trắng hay dấu ‘,’ ở đầu thì sẽ bỏ qua tiếp tục duyệt. Tiếp tục cắt từng kí tự ra để có thể cắt được hết các thành phần ra.

Đầu ra của hàm sẽ trả về xâu cắt được lưu ở địa chỉ của nhãn tmp và trả về số kí tự cắt được. Sau đó ta sẽ cập nhật lại biến lưu vị trí đã được cắt đến ở hiện tại bằng cách cộng nó với số kí tự cắt được hiện tại.

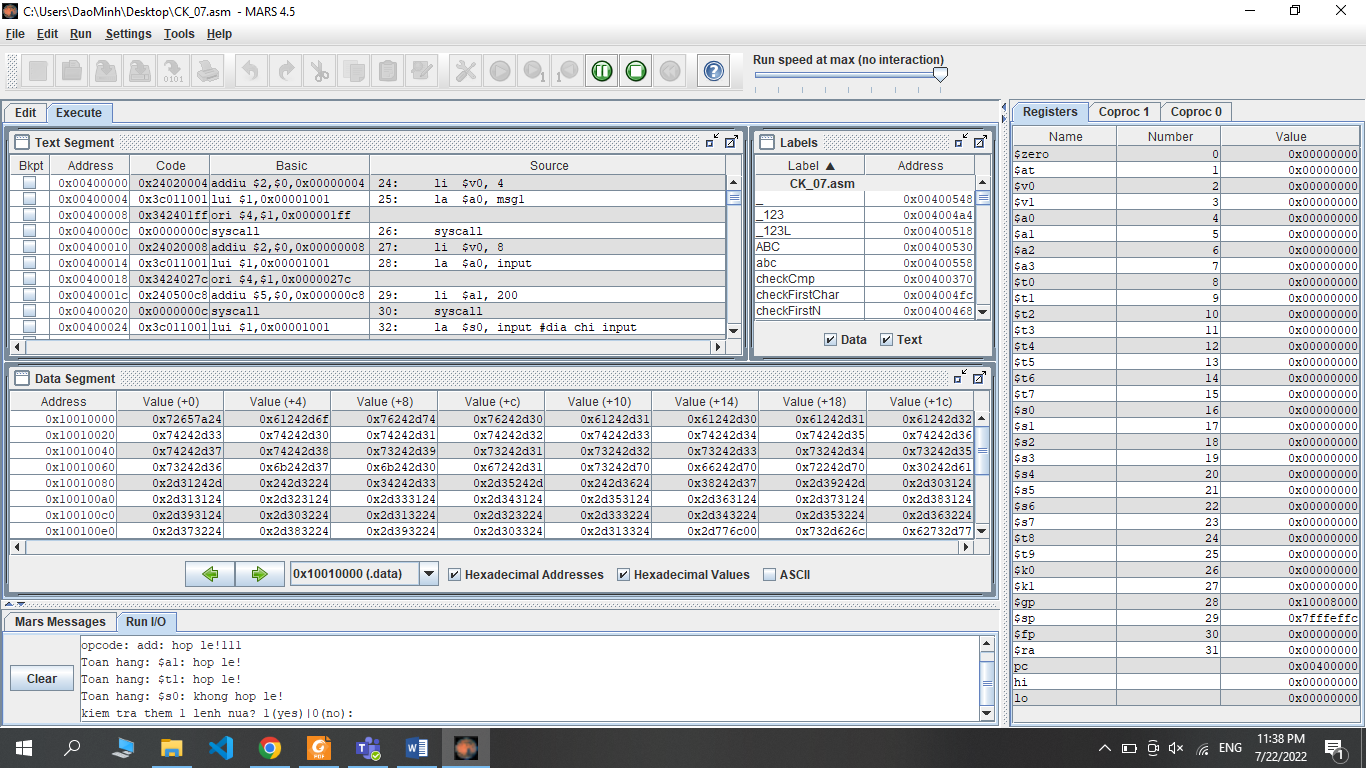
* 1. **compareOpcode:** Là hàm dùng để check mã opcode vừa cắt được ra với thư viện khởi tạo sẵn. Đầu vào của hàm sẽ là địa chỉ cơ sở của tmp và địa chỉ cơ sở của thư viện opcode, cuối cùng là số kí tự hiện có trong tmp.

Sau khi tìm được opcode tương ứng trong thư viện thì hàm sẽ trả $v0 về 1 tương ứng với có opcode, trả về 0 nếu ko có opcode trong thư viện, và trả về $v1 là vị trí của opcode đó trong thư viện. Vì thư viện opcode và thư viện khuôn dạng có cùng định dạng nên ta có thể dùng luôn hàm này để tiến hành check đúng sai của các thanh ghi mà không cần phải tạo lại hàm mới để check.

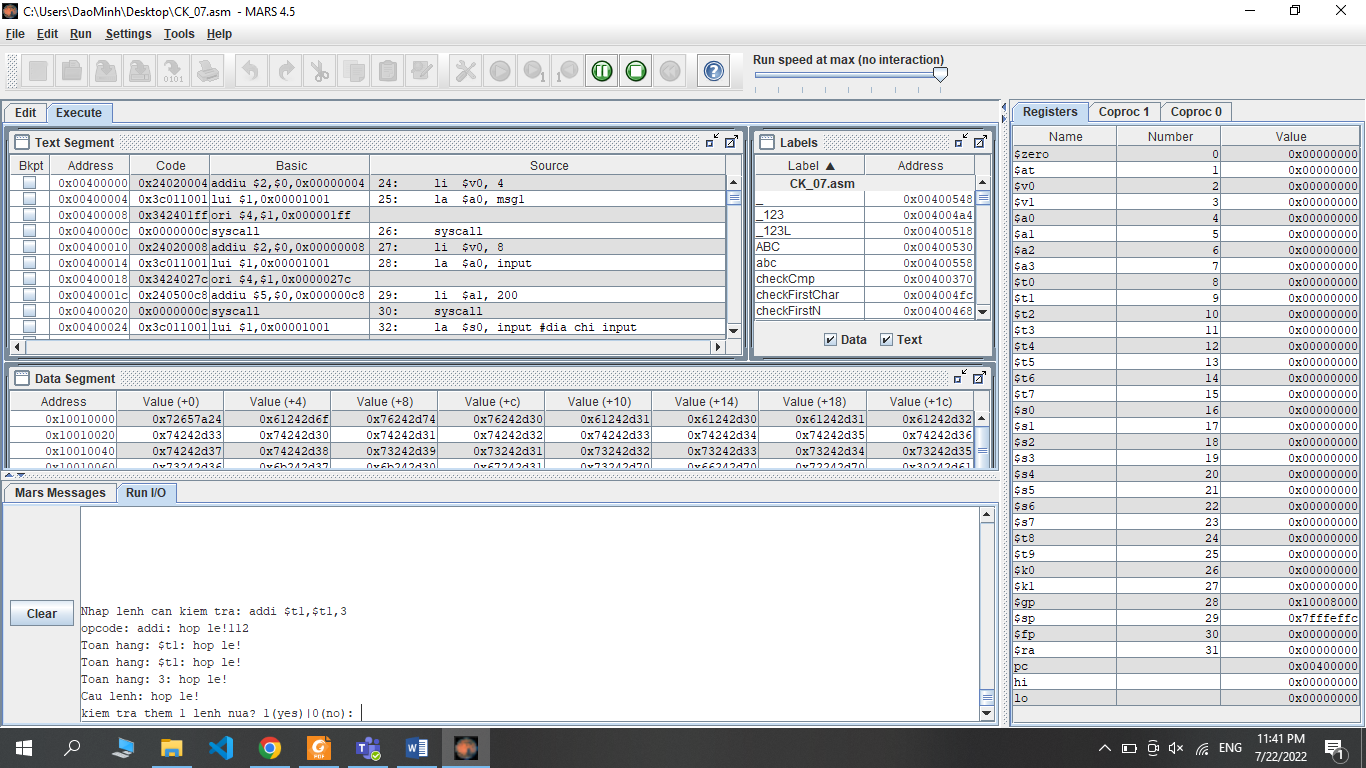
* 1. **getOperand:** Là hàm load ra khuôn dạng của opcode tương ứng với vị trí trả về của op code. Đầu vào của hàm là địa chỉ cơ sở của thư viện khuôn dạng và vị trí tìm thấy trong thư viện của opcode. Đầu ra sẽ trả về khuôn dạng mẫu được đọc ra ở địa chỉ của nhãn tmp2.
  2. **checkNumber:** Là hàm dùng để check các ký tự trong toán hạng có phải là chữ số hay không. Với đầu vào là toán hạng đã cắt được và số kí tự cắt được của toán hạng đó. Còn đầu ra của hàm trả về $v0=1 nếu đầu vào của toán hạng đều là các chứ số âm hay dương, =0 nếu đầu vào không hợp lệ**.**
  3. **checkLabel:** Đầu vào là thành phần cắt ra được từ toán hạng nếu như lable đúng định dạng hàm sẽ trả về giá trị $v0=1 còn nếu không hợp lệ thì sẽ trả $v0=0.

1. **Kết quả mô phỏng**

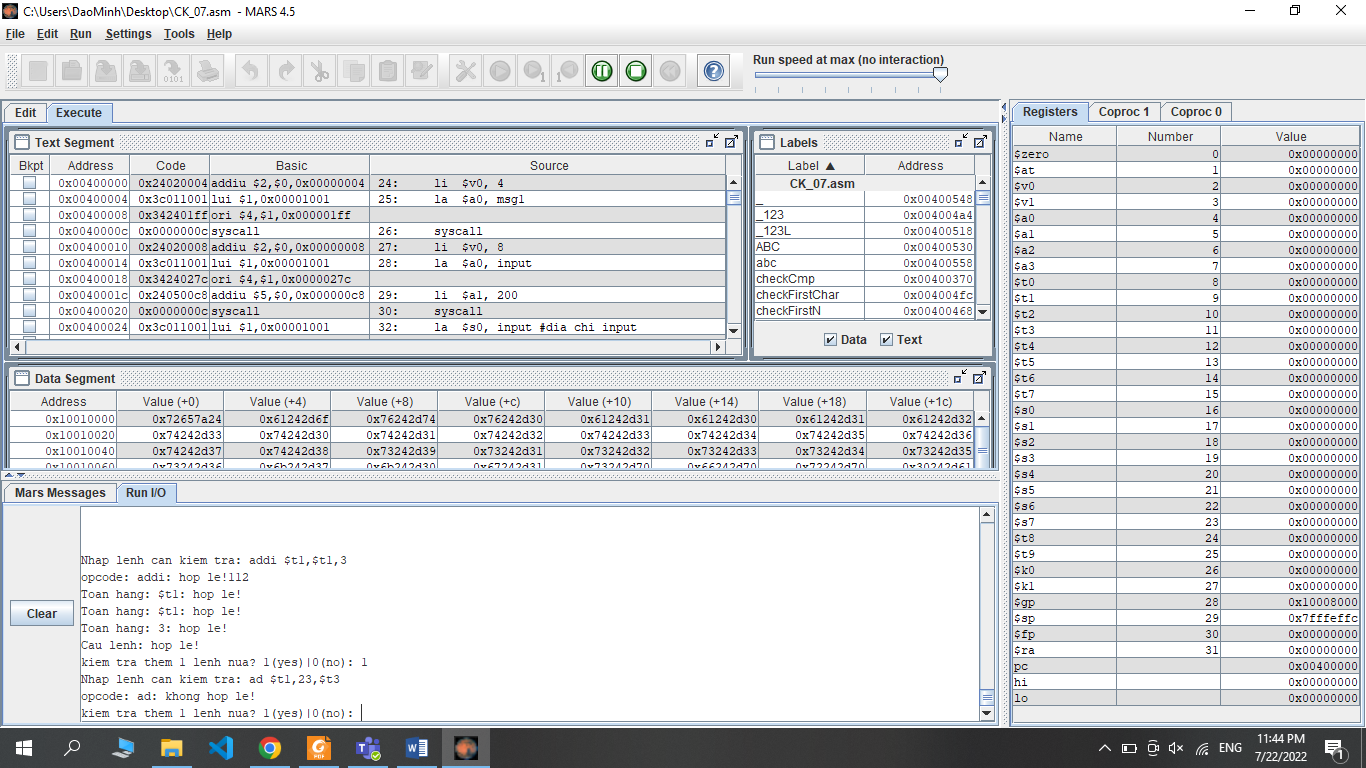
* Chạy code với lệnh add $a1,$t1,$s0



* Với lệnh addi $t1,$t1,3



* Chạy code với lệnh ad $t1,23,$t3



**IV. Giả mã**

Khai báo các thư viện mẫu đề làm

Gồm các thu viện của

Register Thư viện thanh ghi mẫu

Opcode Thư viện opcode mẫu

Operand Thư viện khuôn mẫu

Main {

Char input[200];

Printf (“Nhap lenh: ”)

Scanf (“%s”,input)

Char tmp[20];

Char tmp2[20];

Int viTriCat=0;

cutComponent(); Cat opcode

checkOpcode(); Check opcode

Nếu có opcode thì:

getOperand()

Nếu không thì endmain;

cutComponent(); Cắt toán hạng đầu so sánh với khuôn dạng đầu

Dựa vào khuôn mẫu để gọi hàm check tương ứng

1🡪 checkOpcode (gọi hàm này để check thanh ghi)

2🡪 checkNumber

3🡪 checkLabel

In thông báo ra màn hình

}

Void cutComponent(char input){

For (int i=viTriCat; i<=strlen(input);i++)

If (input[i] == ‘ ’|| ‘,’ || ‘\n’||’\t’ )

Break;

Tmp[0:] = input[i];

End;

Return strlen(tmp);

Return i+viTriCat

}

Vì khuôn dạng của thư viện các thanh ghi giống với thư viện của opcode nên ta sử dụng luôn hàm check opcode de xem thanh ghi có trong thư viện thanh ghi hay không.

Void checkOpcode (char tmp, char opcode)

{

For (I=0;i<strlen(opcode);i++)

For (j=0;j<strlen(tmp);j++)

If (tmp[j] == opcode[j] )

Dem++;

If (dem==strlen(tmp) && opcode[i][j+1] == ‘-’)

Return v0=1;

Return v1=j;

}

Void getOperand (char operand[], v1)

{

Strcpy(tmp2,Operand[v1]);

}

Void checkNumber (char tmp)

{

For (int i=0; i<strlen(tmp); i++)

If (tmp[0]==’+’ || ‘-’)

Continue;

If (tmp[i]>=0 && tmp[i]<=9)

Dem++;

Else

V0=0;

Return v0;

If (dem == strlen(tmp))

V0=1;

Return v0;

Endfor;

}

Void checkLabel(char tmp)

{

If (stmp == null)

End function;

Dem=0;

If (tmp[0] == [‘0’🡪’9’])

Return v0=0;

For (int i=1;i<strlen(tmmp); i++)

If (stmp[i] == ‘\_’ || ’a’🡪’z’ || A🡪Z)

Dem++;

If (dem==strlen(tmp))

V0 = 1;

Return v0;

Continue;

}