ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO CUỐI KÌ

THỰC HÀNH KIẾN TRÚC MÁY TÍNH

MÃ HỌC PHẦN: IT3280

NHÓM 10

Giảng viên hướng dẫn: Lê Bá Vui

Họ và tên	Lớp chuyên ngành	MSSV
Bùi Quang Hưng	Việt Nhật 07 - K67	20225849
Bùi Xuân Nhất	Việt Nhật 07 - K67	20225897

Hà Nội, tháng 6 năm 2024

Nhiệm vụ

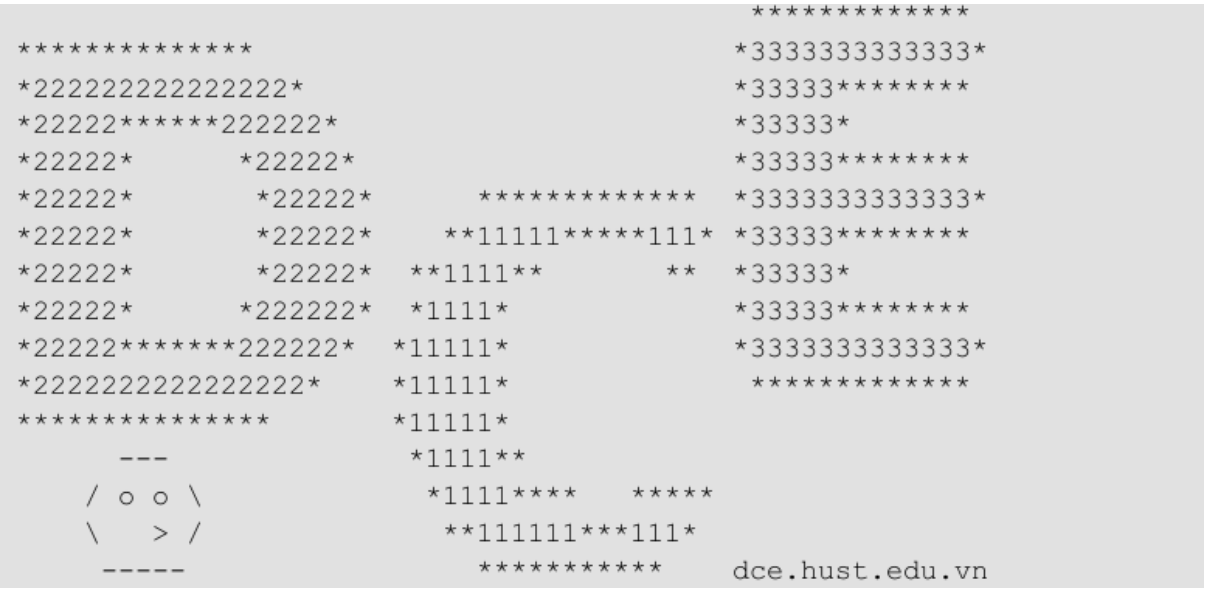
STT	Họ và tên	Vấn đề
1	Bùi Quang Hưng	9
2	Bùi Xuân Nhất	7

Mục Lục

I. Vấn đề 9: Vẽ hình bằng kí tự ASCII

1. Yêu cầu đề bài

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.



- 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.

2. Phân tích cách làm, thuật toán

a. Cách thực hiện và thuật toán

- **Hiển thị hình ảnh trên giao diện console:** Chương trình sẽ duyệt qua tất cả 16 dòng của hình ảnh và in chúng trong giao diện console.
- **Hiển thị hình ảnh chỉ còn lại viền:** Chương trình duyệt qua tất cả 16 dòng của hình ảnh. Trong mỗi dòng, nó duyệt qua từng ký tự, kiểm tra xem chúng có phải là chữ số (màu) hay không và in chúng trong giao diện console. Nếu ký tự là chữ số, nó sẽ được thay thế bằng kí tự khoảng trắng (space) và in ra.
- **Hoán đổi hình ảnh các chữ (từ DCE thành ECD):** Chương trình sẽ duyệt qua tất cả 16 dòng của hình ảnh. Dễ dàng nhận thấy rằng các từ ‘D’, ‘C’, ‘E’ được phân cách bằng khoảng trắng, vì vậy chúng ta có thể tạm thời thay thế khoảng trắng bằng các ký tự null. Trong mỗi dòng, chúng ta sẽ in chuỗi ký tự ‘E’ trước, sau đó là ‘C’ và ‘D’, và

cuối cùng là một ký tự xuống dòng. Sau khi in tất cả các ký tự theo thứ tự chúng ta muốn trong mỗi dòng, chúng ta cần khôi phục khoảng trắng về trạng thái trước đó của chúng.

- **Hiển thị hình ảnh với màu mới:** Đầu tiên, chương trình yêu cầu người dùng nhập màu mới cho ‘D’, ‘C’, ‘E’. Nếu màu không hợp lệ, chương trình yêu cầu người dùng nhập lại cho đến khi nhận được màu hợp lệ. Chương trình lưu trữ cả màu cũ và mới của ‘D’, ‘C’, ‘E’. Chương trình duyệt qua tất cả 16 dòng của hình ảnh. Trong mỗi dòng, chương trình kiểm tra tuần tự 3 từ. Nếu bất kỳ ký tự nào là màu cũ, chương trình cập nhật chúng thành màu mới tương ứng với mỗi từ bằng cách lưu trữ các giá trị mới vào bộ nhớ, để sau đó người dùng có thể thấy hình ảnh được cập nhật. Trong khi cập nhật, chương trình in hình ảnh mới để người dùng có thể thấy sự thay đổi. Sau khi hoàn tất việc cập nhật hình ảnh, các giá trị màu hiện tại của 'D', 'C', và 'E' được cập nhật với các màu mới.

b. Mã nguồn code:

- Phần data khai báo các dòng của hình ảnh và các chuỗi thông báo khác.

```
.data
    line1:  .asciiz "                                        ************ \n"
    line2:  .asciiz "**************                         *3333333333333*\n"
    line3:  .asciiz "*2222222222222*                         *33333******** \n"
    line4:  .asciiz "*22222******222222*                     *33333*        \n"
    line5:  .asciiz "*22222*      *22222*                    *33333******** \n"
    line6:  .asciiz "*22222*       *22222*      ************  *3333333333333*\n"
    line7:  .asciiz "*22222*       *22222*    **11111*****111*  *33333******** \n"
    line8:  .asciiz "*22222*       *22222*  **1111**      **    *33333*        \n"
    line9:  .asciiz "*22222*      *222222*  *1111*             *33333******** \n"
    line10: .asciiz "*22222******222222*  *11111*             *3333333333333*\n"
    line11: .asciiz "*2222222222222222*   *11111*              ************ \n"
    line12: .asciiz "***************      *11111*                           \n"
    line13: .asciiz "      ---             *1111**                         \n"
    line14: .asciiz "    / o o \\            *1111****   *****               \n"
    line15: .asciiz "    \\   > /             **111111***111*               \n"
    line16: .asciiz "     -----                **********     dce.hust.edu.vn\n"
    MENU:   .asciiz "\n ----MENU----\n 1.Show picture.\n 2.Show picture without color.\n 3
.Change the order.\n 4.Change the color.\n 5.Exit.\n Enter your option(From 1 to 5): "
    ERROR:  .asciiz "Your option is invalid. Please choose again.\n"
    ColorD: .asciiz "Choose the color of D(From 0 to 9):"
    ColorC: .asciiz "Chooes the color of C(From 0 to 9):"
    ColorE: .asciiz "Choose the color of E(From 0 to 9):"
```

- Hiển thị menu và yêu cầu người dùng nhập lựa chọn. Nếu lựa chọn không hợp lệ, yêu cầu người dùng nhập lại.

```
.text
main:
    # Display the menu
    li $v0, 4
    la $a0, MENU
    syscall

    # Get the user's option
    li $v0, 5
    syscall

    # Check user input and branch to the corresponding option
    beq $v0, 1, Option1
    beq $v0, 2, Option2
    beq $v0, 3, Option3
    beq $v0, 4, Option4
    beq $v0, 5, Option5

    # If input is invalid, print error message and restart
    li $v0, 4
    la $a0, ERROR
    syscall
    j main
```

- Option1 là hiển thị hình ảnh có màu. Chúng ta duyệt qua tất cả 16 dòng, sử dụng $a0 làm con trỏ đến địa chỉ cơ sở của mỗi dòng và in chúng ra. Sau một lần lặp, chúng ta tăng $a0 lên 60 vì mỗi dòng có 60 ký tự.

```
#Function1
Option1:
    li $t0, 0   # Initialize loop counter i=0
    li $t1, 16  # Set loop limit to 16
    la $a0, line1   #Load address of line1 to $a0
Loop_1:
    beq $t0, $t1, main  # If all rows are printed, return to main menu
    li $v0, 4       #Print line1
    syscall

    addi $a0,$a0, 60    #Move to the next line
    addi $t0, $t0, 1    # Increment loop counter
    j Loop_1        # Jump back to the start of the loop
```

- Option 2 là hiển thị hình ảnh chỉ với đường viền. Chương trình sử dụng các vòng lặp lồng nhau để đi qua mọi ký tự của mọi dòng. Nếu ký tự không phải là chữ số, chỉ cần in ra. Nếu là chữ số, thay thế kí tự bằng khoảng trắng (space) và in ra.

```
#Function2
Option2:
    li $t0, 0   # Initialize row counter i=0
    li $t1, 16  # Set row limit to 16
    la $t2,line1    # Load address of line1 into $t2
Loop_row:
    beq $t0, $t1, main  # If all rows are visited, return to main menu
    li $t3, 0       # Initialize column counter j=0
    li $t4, 60      # Set column limit to 60
Loop_col:
    beq $t3, $t4, Next_row  # If all columns are visited, go to next row
    lb $t5, 0($t2)      # Load current byte from memory into $t5
    blt $t5, 48, Print_char # If character is not a digit, print it
    bgt $t5, 57, Print_char
    li $t5, 32      # Replace digit with a space
Print_char:
    li $v0, 11
    move $a0, $t5
    syscall

    addi $t2, $t2, 1    # Move to the next character in the current row
    addi $t3, $t3, 1    # Increment the column counter
    j Loop_col      # Jump back to column loop
Next_row:
    addi $t0, $t0, 1    # Increment the row counter
    j Loop_row      # Jump back to row loop
```

- Option3 là in hình ảnh theo thứ tự ‘E’, ‘C’, ‘D’. Chương trình sử dụng 1 vòng lặp để duyệt qua từng dòng. Ở mỗi dòng, chúng ta tạm thời cập nhật ký tự thứ 21, 42, 58 thành các ký tự kết thúc bằng ký tự null trong bộ nhớ vì các vị trí này là ranh giới giữa các từ. Sau đó, chúng ta có thể dễ dàng in các từ ‘E’, ‘C’, ‘D’ với sự trợ giúp của các ký tự kết thúc bằng ký tự null. Chúng ta cũng in khoảng cách giữa các từ và xuống dòng ở cuối để yêu cầu đề bài được trình bày rõ ràng. Cuối cùng, chúng ta cần khôi phục trạng thái của hình ảnh trong bộ nhớ.

```
#Function3
Option3:
    li $t0, 0   # Initialize row counter i = 0
    li $t1, 16  # Set row limit to 16
    la $t2,line1    # Load address of line1 into $t2
Loop_row3:
    beq $t0, $t1, main  # If all columns are visited, go to next row
    sb $0, 21($t2)      # Set the 22nd character in the line to null (end of D)
    sb $0, 42($t2)      # Set the 43rd character in the line to null (end of C)
    sb $zero, 58($t2)   # Set the 59th character in the line to null (end of E)

    li $v0, 4
    addi $a0, $t2, 43   # Set $a0 to the start of E
    syscall         # Print the string starting at the 43rd character (E)



   li $v0, 11
   li $a0, 32      # Set $a0 to space character
   syscall         # Print a space

   li $v0, 4
   addi $a0, $t2, 22   # Set $a0 to the start of C
   syscall         # Print the string starting at the 22nd character (C)

   li $v0, 11
   li $a0, 32      # Set $a0 to space character
   syscall         # Print a space

   li $v0, 4
   add $a0, $t2, $0    # Set $a0 to the start of D (beginning of the line)
   syscall         # Print the string starting at the beginning of the line (D)

   li $v0, 11
   li $a0, 10      # Set $a0 to newline character
   syscall         # Print a newline

   # Restore original characters
   li $t3, 32      # Set $t3 to space character
   sb $t3, 21($t2)     # Restore the 22nd character to space
   sb $t3, 42($t2)     # Restore the 43rd character to space
   li $t3, 10      # Set $t3 to newline character
   sb $t3, 58($t2)     # Restore the 59th character to newline

   addi $t0, $t0, 1    # Increment the row counter
   addi $t2, $t2, 60   # Move to the next row
   j Loop_row3     # Jump back to row loop
```

- Option4 là cập nhật hình ảnh bằng màu mới do người dùng nhập. Chương trình yêu cầu người dùng nhập 3 màu mới, lưu trữ chúng trong $s4, $s5, $s6 và yêu cầu nhập lại nếu bất kỳ màu nào không hợp lệ (màu hợp lệ là chữ số từ 0 đến 9).

```
#Function4
Option4:

D_color:
    li $v0, 4
    la $a0, ColorD
    syscall         # Print the prompt to choose color for D
    li $v0, 5       # Read the user input (color for D)
    syscall
    blt $v0, 0, D_color # If input is less than 0, ask again
    bgt $v0, 9, D_color # If input is less than 0, ask again
    addi $s4, $v0, 48   # Convert the number to its ASCII character equivalent
C_color:
    li $v0, 4
    la $a0, ColorC
    syscall         # Print the prompt to choose color for C
    li $v0, 5       # Read the user input (color for C)
    syscall
    blt $v0, 0, C_color     # If input is less than 0, ask again
    bgt $v0, 9, C_color     # If input is greater than 9, ask again


    addi $s5, $v0, 48       # Convert the number to its ASCII character equivalent
E_color:
    li $v0, 4
    la $a0, ColorE
    syscall         # Print the prompt to choose color for E
    li $v0, 5       # Read the user input (color for E)
    syscall
    blt $v0, 0, E_color     # If input is less than 0, ask again
    bgt $v0, 9, E_color     # If input is greater than 9, ask again
    addi $s6, $v0, 48       # Convert the number to its ASCII character equivalent
```

- Chương trình sử dụng các vòng lặp lồng nhau để duyệt qua mọi ký tự của mọi dòng. Trong mỗi dòng, 21 ký tự đầu tiên thuộc về ‘D’, 21 ký tự tiếp theo thuộc về ‘C’ và các ký tự còn lại thuộc về ‘E’. Chương trình xử lý các ký tự của mỗi từ với 3 nhánh ‘check_D’, ‘check_C’ và ‘check_E’. Trong mỗi nhánh, nếu ký tự là chữ số, chương trình cập nhật nó bằng màu tương ứng mới trong bộ nhớ với 1 trong 3 nhánh ‘Change_color_D’, ‘Change_color_C’, ‘Change_color_E’. Chương trình cũng in tất cả các ký tự ra để người dùng thấy được hình ảnh được cập nhật. Sau khi lặp qua tất cả các dòng, chúng tôi cập nhật các giá trị màu hiện tại ($s0, $s1, $s2) bằng các giá trị mới ($s4, $s5, $s6).

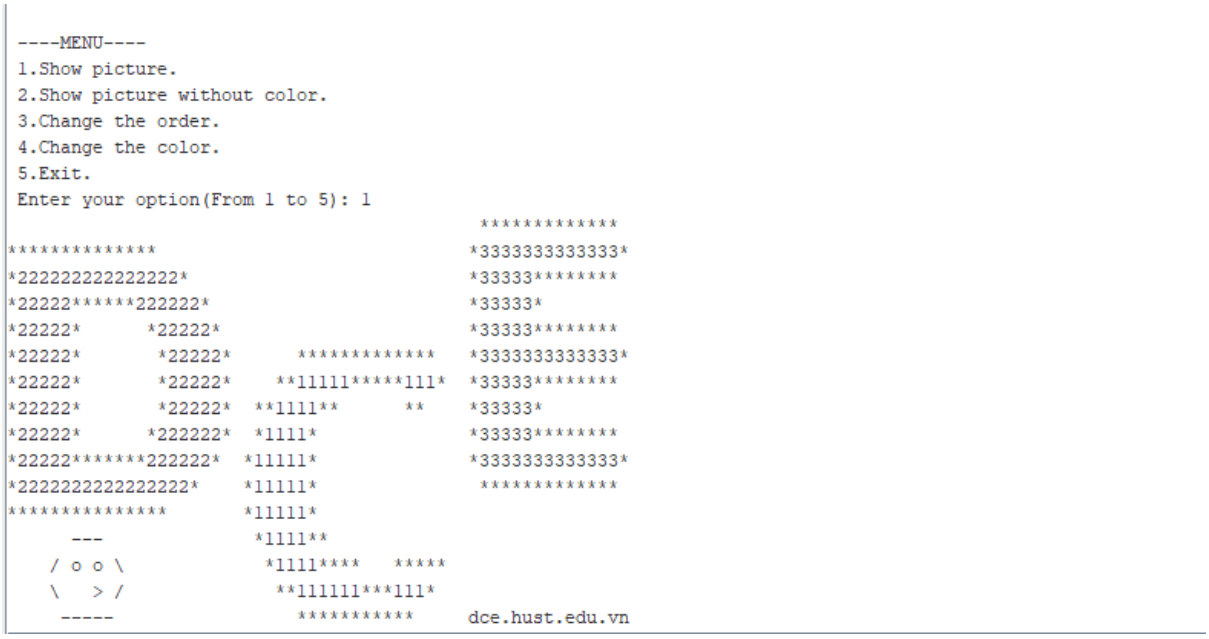
```
Func4:
    li $t0, 0   # Initialize row counter i = 0
    li $t1, 16  # Set row limit to 16
    la $t2,line1    # Load address of line1 into $t2
Loop_row4:
    beq $t0, $t1, main # If all columns are visited, go to next row
    li $t3, 0      # Initialize column counter j = 0
    li $t4, 60     # Set the maximum column count to 60
Loop_col4:
    beq $t3, $t4, Next_row4 # If all columns are visited, go to next row
    lb $t6, 0($t2)      # Load the current character into $t6
    blt $t3, 21, Check_D    # If column is less than 22, check if it's part of D
    blt $t3, 42, Check_C    # Otherwise, check if it's part of E
    j Check_E       # Otherwise, check if it's part of E
Check_D:
    beq $t6,'2', Change_color_D # If character is '2' (part of D), change color
    j Print             # Otherwise, print the character
Check_C:
    beq $t6, '1', Change_color_C    # If character is '1' (part of C), change color
    j Print             # Otherwise, print the character
Check_E:
    beq $t6, '3', Change_color_E    # If character is '3' (part of E), change color
    j Print             # Otherwise, print the character
Change_color_D:
    move $t6, $s4   # Change character to the chosen color for D
    j Print
Change_color_C:
    move $t6, $s5   # Change character to the chosen color for C
    j Print
Change_color_E:
    move $t6, $s6   # Change character to the chosen color for E
    j Print
Print:
    li $v0, 11
    move $a0, $t6   # Move character to $a0
    syscall

    addi $t2, $t2, 1    # Move to the next character
    addi $t3, $t3, 1    # Increment column counter
    j Loop_col4     # Jump back to column loop
Next_row4:
   addi $t0, $t0, 1    # Increment row counter
   j Loop_row4     # Jump back to row loop
```

- Option5 là kết thúc chương trình

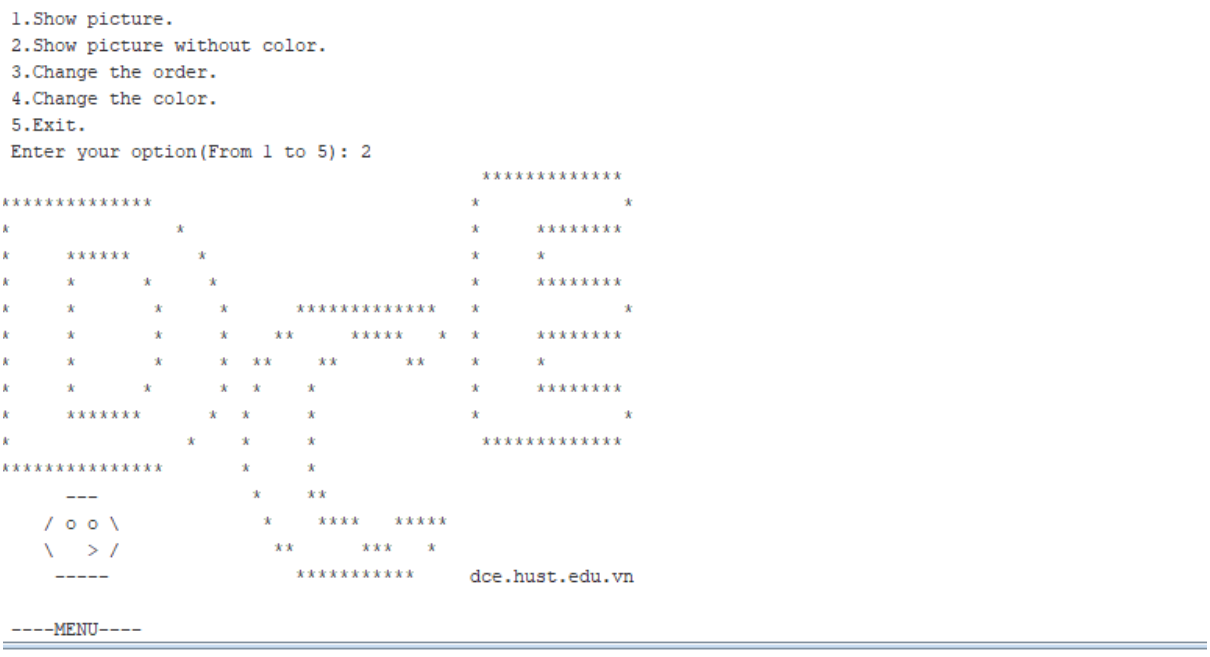
```
#End program
Option5:
    li $v0, 10
    syscall
```

3. Kết quả

- Hiển thị hình ảnh trên giao diện console



- Hiển thị hình ảnh chỉ còn lại viền



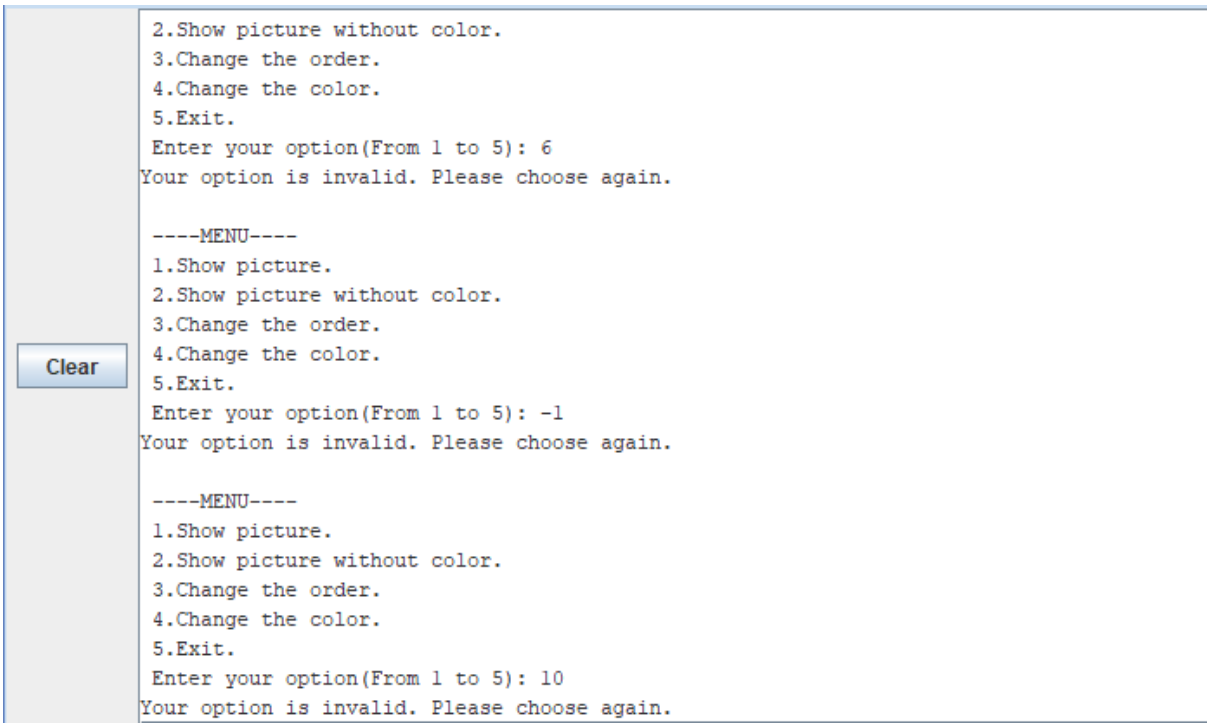
- Hoán đổi hình ảnh các chữ (từ DCE thành ECD)

```
 ----MENU----
 1.Show picture.
 2.Show picture without color.
 3.Change the order.
 4.Change the color.
 5.Exit.
 Enter your option(From 1 to 5): 3
 *************
*3333333333333*                      **************
*33333********                       *222222222222222*
*33333*                              *22222******222222*
*33333********                       *22222*      *22222*
*3333333333333*      *************   *22222*       *22222*
*33333********     **llllll*****lll*  *22222*       *22222*
*33333*           **llll**      **    *22222*       *22222*
*33333********    *llll*              *22222*      *222222*
*3333333333333* *llllll*              *22222*******222222*
 *************  *llllll*              *2222222222222222*
                *llllll*              ***************
                 *llll**                    ---
                  *llll****   *****       / o o \
                   **llllll***lll*        \   > /
dce.hust.edu.vn      ***********           -----
```

- Hiển thị hình ảnh với màu mới nhập vào từ người dùng

```
 3.Change the order.
 4.Change the color.
 5.Exit.
 Enter your option(From 1 to 5): 4
Choose the color of D(From 0 to 9):6
Chooes the color of C(From 0 to 9):7
Choose the color of E(From 0 to 9):8
                                        *************
**************                         *8888888888888*
*6666666666666666*                      *88888********
*66666******666666*                     *88888*
*66666*      *66666*                    *88888********
*66666*       *66666*      *************  *8888888888888*
*66666*       *66666*    **77777*****777*  *88888********
*66666*       *66666*  **7777**      **    *88888*
*66666*      *666666*  *7777*             *88888********
*66666******666666*  *77777*             *8888888888888*
*6666666666666666*    *77777*              *************
***************       *77777*
      ---              *7777**
    / o o \             *7777****   *****
    \   > /              **777777***777*
     -----                  **********     dce.hust.edu.vn
```

- **Trường hợp input đầu vào không hợp lệ**



- **Kết thúc chương trình**

```
----MENU----
1.Show picture.
2.Show picture without color.
3.Change the order.
4.Change the color.
5.Exit.
Enter your option(From 1 to 5): 5

-- program is finished running --
```

II. Vấn đề 7: Chương trình kiểm tra cú pháp lệnh MIPS :

1. Yêu cầu đề bài :

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.*
- Kiểm 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.

2. Phân tích cách làm, thuật toán:

a. Cách thực hiện và Thuật toán :

- **Bước 1**: Gọi ra menu thực hiện lệnh (bắt đầu đọc vào input) hoặc thoát.
- **Bước 2**: Kiểm tra câu lệnh :
    - Kiểm tra Opcode (duyệt các Opcode ở trong thư viện nếu không có Opcode nào giống với Input thì in ra thông báo lỗi và kết thúc).
    - Kiểm tra đến các toán hạng (lấy được dạng các toán hạng trong thư viện và kiểm tra ứng với dạng toán hạng tương ứng).
    - Khi kiểm tra hết toán hạng thì kiểm tra xem người dùng có nhập thừ ký tự nào không.
- **Bước 3**: In ra thông báo ứng với kết quả kiểm tra được và đặt lại các giá trị về mặc định và gọi ra menu.

b. Mã nguồn:

```
.data
# ----------------------------------------library----------------------------------------
--------------------
# opcode (7) - operation (3)
# Trong so luong operation: 1 - thanh ghi; 2 - hang so nguyen; 3 - dinh danh (ident); 4 -
imm($rs); 0 - khong co
    list:   .asciiz "add****111;sub****111;addi***112;addu***111;addiu**112;subu***111;mfc
0***110;mult***110;multu**110;div****110;mfhi***100;mflo***100;and****111;or*****111;andi*
**112;ori****112;sll****112;srl****112;lw*****140;sw*****140;lbu****140;sb*****140;lui****
120;beq****113;bne****113;slt****111;slti***112;sltiu**112;j******300;jal****300;jr*****10
0;nop****000"
    register: .asciiz "$zero $at   $v0   $v1   $a0   $a1   $a2   $a3   $t0   $t1   $t2   $
t3   $t4   $t5   $t6   $t7   $s0   $s1   $s2   $s3   $s4   $s5   $s6   $s7   $t8   $t9   $
k0   $k1   $gp   $sp   $fp   $ra   $0    $1    $2    $3    $4    $5    $6    $7    $8    $
9    $10   $11   $12   $13   $14   $15   $16   $17   $18   $19   $20   $21   $22   $21   $
22   $23   $24   $25   $26   $27   $28   $29   $30   $31   "
    number:     .asciiz "0123456789-"
    character: .asciiz "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ_"
#-----------------------------------------------Message-------------------------------------
    msg_menu:       .asciiz "\n1. Kiem tra cu phap lenh\n2. Thoat \nChon: "
    msg_menu_error: .asciiz "\nNhap sai, vui long nhap lai!\n"
    msg_input:      .asciiz "\nNhap vao lenh Mips: "

    msg_opcode:     .asciiz "Opcode: "
    msg_operand:    .asciiz "Toan hang: "
    msg_valid:  .asciiz " - hop le.\n"
    msg_invalid:    .asciiz "\nLenh hop ngu khong hop le, sai khuon dang lenh !\n"
    msg_valid_mips: .asciiz "\nLenh hop ngu chinh xac !\n"
    command:    .space 100  # Luu cau lenh
    opcode:     .space 30   # Luu ma lenh, vi du: add, and,...
    ident:      .space 30   # nhan | hoac number
    token:      .space 30   # cac thanh ghi, vi du: $zero, $at,...
```

- Khai báo các thư viện chứa các cấu trúc lệnh MIPS và chứa các thư viện character, number, register (mỗi cấu trúc lệnh chiếm 11 ô trong list, mỗi thanh nhớ chiếm 6 ô trong register).

```
.text
main:
# ----- MENU -----
call_menu:
    li $v0, 4
    la $a0, msg_menu
    syscall

    # Read number input menu
    li $v0, 5
    syscall

    beq $v0, 2, end_main        # 2: ket thuc
    beq $v0, 1, end_call_menu       # 1: thuc hien kiem tra

    li $v0, 4
    la $a0, msg_menu_error      # Nhap sai
    syscall

   j call_menu
end_call_menu:
```

- Gọi ra menu chương trình để cho người dùng lựa chọn 1 (Kiểm tra mã lệnh) hoặc 2 (Thoát chương trình) và yêu cầu người dùng nhập lại khi nhập khác với 2 giá trị trên.

```
input:
    li $v0, 4
    la $a0, msg_input
    syscall

    li $v0, 8
    la $a0, command
    li $a1, 100
    syscall
```

- Thực hiện quá trình đọc input.

```
check:
    #CHECK OPCODE
    jal checkOpcode
    nop

    # CHECK OPERAND 1
    li  $s3, 7          # Vi tri operand trong Library
    jal check_operand
    nop

    # CHECK OPERAND 2
    li  $s3, 8          # Vi tri operand trong Library
    add $t0, $s5, $s3
    lb  $t0, 0($t0)
    beq $t0, 48, check_none     # Kiem tra neu operand = 0 -> ket thuc; ky tu 0 trong ASCI
I

    la   $a0, command
    add  $t0, $a0, $s7      # tro toi vi tri tiep tuc cua command
    lb   $t1, 0($t0)
    bne  $t1, 44, not_found     # Dau ','
    add  $s7, $s7, 1

    jal check_operand
    nop

    #CHECK OPERAND 3
    li  $s3, 9          # Vi tri operand trong Library
    add $t0, $s5, $s3
    lb  $t0, 0($t0)
    beq $t0, 48, check_none     # Kiem tra neu operand = 0 -> ket thuc; ky tu 0 trong ASCI
I

    la   $a0, command
    add  $t0, $a0, $s7      # tro toi vi tri tiep tuc cua command
    lb   $t1, 0($t0)
    bne  $t1, 44, not_found     # Dau ','
```

```
    add  $s7, $s7, 1

    jal check_operand
    nop

    # KIEM TRA KY TU THUA
    j check_none

    j reset

end_main:
    li $v0, 10
    syscall
```

- Phần chương trình chính để bắt đầu kiểm tra cấu trúc lệnh lần lượt thực hiện các chương trình con để kiểm tra từng thành phần của cấu trúc lệnh MIPS. Mỗi khi kiểm tra xong 1 toán hạng thì kiểm tra xem có dấu phẩy không rồi mới kiểm tra tiếp.
- Sau khi kiểm tra hết thì đặt lại các giá trị đã sử dụng và kết thúc chương trình.

```
checkOpcode:
    la  $a0, command                # Dia chi cua command
    la  $a1, opcode                 # Dia chi cua opcode
    li  $t0, 0

remove_space_command:                   # Xoa cac dau cach phia truoc lenh
    add $t1, $a0, $t0
    lb  $t2, 0($t1)
    bne $t2, 32, end_remove_space_command       # Neu khong phai ' ' -> Ket thuc
    addi $t0, $t0, 1
    j remove_space_command
end_remove_space_command:

    li  $t9, 0                  # index for opcode
    li  $s6, 0                  # so luong cac ki tu cua opcode = 0
read_opcode:
    add $t1, $a0, $t0               # Dich bit cua command
    add $t2, $a1, $t9               # Dich bit cua opcode
    lb  $t3, 0($t1)

    beq $t3, 32, read_opcode_done           # Neu co dau cach ' ' ket thuc read opcode
    beq $t3, 10, read_opcode_done           # Neu dau '\n' ket thuc read opcode
    beq $t3, 0,  read_opcode_done           # Ket thuc chuoi

    sb  $t3, 0($t2)
    addi $t9, $t9, 1
    addi $t0, $t0, 1
    j read_opcode
read_opcode_done:

    addi $s6, $t9, 0                # $s6: So luong ki tu cua opcode
    add $s7, $s7, $t0               # luu index cua command
    la $a2, list
    li $t0, -11

checkOpcode_inlib:
    addi $t0, $t0, 11               # Buoc nhay bang 10 de nhay den tung Instruction
```

```
    li $t1, 0                   # i = 0
    li $t2, 0                   # j = 0
    add $t1, $t1, $t0               # Cong buoc nhay

    compare_opcode:
        add $t3, $a2, $t1           # t3 tro thanh vi tri tro den dau cua tung Instruction
        lb  $t4, 0($t3)
        beq $t4, 0, not_found
        beq $t4, 42, check_len_opcode       # Neu gap ky tu `*` => Kiem tra do dai
        add $t5, $a1, $t2           # Load opcode
        lb  $t6, 0($t5)
        bne $t4, $t6, checkOpcode_inlib # So sanh 2 ki tu, neu khong bang nhau thi tinh de
n Instruction tiep theo.
        addi $t1, $t1, 1            # i = i + 1
        addi $t2, $t2, 1            # j = j + 1
        j compare_opcode
    check_len_opcode:
        bne $t2, $s6, checkOpcode_inlib
end_checkOpcode_inlib:

    add $s5, $t0, $a2               # Luu lai vi tri Instruction trong Library.

    # ----- In thong tin ra man hinh -----
    li $v0, 4
    la $a0, msg_opcode
    syscall

li $v0 , 4
la $a0 , opcode
syscall

    li $v0, 4
    la $a0, msg_valid
    syscall

    jr $ra
```

- Phần chương trình để kiểm tra Opcode của lệnh.
- Đọc dữ liệu vào chuỗi Opcode:
    - Thực hiện xóa khoảng trắng trước input trước khi đọc dữ liệu vào xâu Opcode.
    - Thực hiện sao chép Opcode từ input vào trong xâu nếu gặp khoảng trắng hoặc dấu xuống dòng, kí tự kết thúc xâu thì dừng.
    - Thực hiện tìm cấu trúc lệnh ở trong list. Duyệt từng cấu trúc lệnh trong list và thực hiện so sánh với Opcode vừa thu được ở bước trên. Nếu có thì nhảy đến in ra màn hình thông báo về Opcode và trở về hàm chính để tiếp tục kiểm tra toán hạng, nếu không thì nhảy đến nhãn not_found và thực hiện kết thúc quá trình kiểm tra.

```
check_operand:
    # Luu $ra de tro ve check_operand
    addi $sp, $sp, -4
    sw   $ra, 0($sp)

    add $t9, $s5, $s3           # Tro toi operand trong Library
    lb  $t9, 0($t9)
    addi $t9, $t9, -48          # Char -> Number

    la  $a0, command
    add $t0, $a0, $s7

      li $t1, 0                   # i = 0
      space_remove:               # Xoa cac khoang trang thua
          add $t2, $t0, $t1
          lb  $t2, 0($t2)         # Lay ky tu tiep theo
          bne $t2, 32, end_space_remove   # Ky tu ' '
          addi $t1, $t1, 1        # i = i + 1
          j space_remove
      end_space_remove:

      add $s7, $s7, $t1           # Cap nhat lai index command

      li $s2, 0               # Tat kich hoat check number_register
      li $t8, 0               # Khong co
      beq $t8, $t9, check_none
      li $t8, 1               # Thanh ghi
      beq $t8, $t9, go_register
      li $t8, 2               # So hang nguyen
      beq $t8, $t9, go_number
      li $t8, 3               # Ident
      beq $t8, $t9, go_ident
      li $t8, 4               # Check number & register
      beq $t8, $t9, go_number_register

  end_check_operand:
      # Tra lai $ra de tro ve check_operand
      lw   $ra, 0($sp)
      addi $sp, $sp, 4
      jr   $ra
```

- Hàm dùng để kiểm tra toán hạng với các bước xử lý tương tự với Opcode là xóa khoảng trắng và kiểm tra dạng toán hạng dựa vào các cấu trúc đã khai báo sẵn trong list.

```
go_register:                # Check register
    jal check_register
    nop
j end_check_operand

go_number:              # Check number
    la $a2, number
    jal check_ident
    nop
j end_check_operand

go_ident:               # Check Ident
    la $a2, character
    jal check_ident
    nop
j end_check_operand

go_number_register:         # Check number-register
    jal check_number_register
    nop
j end_check_operand
```

- Thực hiện gọi đến các hàm để thực hiện kiểm tra các toán hạng.

```
check_none:
    la $a0, command
    add $t0, $a0, $s7

    lb $t1, 0($t0)

    beq $t1, 10, none_ok    # Ky tu '\n'
    beq $t1, 0, none_ok # Ket thuc chuoi

    j not_found

none_ok:
    li $v0, 4
    la $a0, msg_valid_mips
    syscall
    j call_menu
```

- Kiểm tra với dạng khuyết toán hạng và kiểm tra các ký tự thừa.

```
check_register:
    la $a0, command
    la $a1, token
    la $a2, register
    add $t0, $a0, $s7               # Tro den vi tri cac instruction

    li $t1, 0                   # i = 0
    li $t9, 0                   # index cua token

read_token_register:
    add $t2, $t0, $t1               # command
    add $t3, $a1, $t1               # token
    lb $t4, 0($t2)

    beq $t4, 41, end_read_token         # Gap ky tu ')'
    beq $t4, 44, end_read_token         # Gap ky tu ' , '
    beq $t4, 10, end_read_token         # Gap ky tu '\n'
    beq $t4, 0, end_read_token          # Ket thuc

    addi $t1, $t1, 1
    beq $t4, 32, read_token_register         # Neu gap dau ' ' thi tiep tuc

    sb $t4, 0($t3)
    addi $t9, $t9, 1
    j read_token_register

end_read_token:
    add $s7, $s7, $t1               # Cap nhat lai gia tri index
```

```
    li $t0, -6
compare_token_register:
    addi $t0, $t0, 6                # Buoc nhay bang 6 de nhay den tung Register

    li $t1, 0                   # i = 0
    li $t2, 0                   # j = 0

    add $t1, $t1, $t0               # Cong buoc nhay

    compare_reg:
        add $t3, $a2, $t1           # t3 tro thanh vi tri tro den dau cua tung Register
        lb  $t4, 0($t3)
        beq $t4, 0, not_found
        beq $t4, 32, check_len_reg      # Neu gap ky tu ` ` => Kiem tra do dai

        add $t5, $a1, $t2           # Load token
        lb  $t6, 0($t5)

        bne $t4, $t6, compare_token_register    # So sanh 2 ki tu, neu khong bang nhau thi
 tinh den Register tiep theo.
        addi $t1, $t1, 1            # i = i + 1
        addi $t2, $t2, 1            # j = j + 1
        j compare_reg

    check_len_reg:
        bne $t2, $t9, compare_token_register    # Neu do dai khong bang nhau di den regist
er tiep theo

end_compare_token_register:

    # >>>>>>>>> In thong tin ra man hinh <<<<<<<<<<
    beq $s2, 1, on_token_number_register
    li $v0, 4
    la $a0, msg_operand
    syscall

li $v0 , 4
la $a0 , token
syscall

    li $v0, 4
    la $a0, msg_valid
    syscall
    jr $ra

on_token_number_register:

li $v0 , 4
la $a0 , token
syscall
   li $v0, 11
   li $a0, 41
   syscall
   li $v0, 4
   la $a0, msg_valid
   syscall
   jr $ra
```

- Hàm thực hiện kiểm tra các toán hạng là thanh ghi với các thao tác như sao chép chuỗi token từ input và thực hiện so sánh với các thanh ghi có sẵn trong vùng "register". Nếu không có, thực hiện nhảy đến nhãn not_found và kết thúc, ngược lại nếu như tìm thấy thì in ra màn hình thông tin và tiếp tục kiểm tra chương trình.
- Ở đây, ta có thanh ghi $s2 để kiểm tra rằng trước thanh ghi có đọc vào số nào không để kiểm tra các lệnh sb, lb,…

```
check_ident:
    la $a0, command
    la $a1, ident

    add $t0, $a0, $s7           # Tro den vi tri cac instruction

    li $t1, 0               # i = 0
    li $t9, 0               # index cua ident

read_ident:
    add $t2, $t0, $t1           # command
    add $t3, $a1, $t1           # ident
    lb $t4, 0($t2)

    beq $t4, 40, end_read_ident     # Gap ky tu '('
    beq $t4, 44, end_read_ident     # Gap ky tu ' , '
    beq $t4, 10, end_read_ident     # Gap ky tu '\n'
    beq $t4, 0, end_read_ident      # Ket thuc

    addi $t1, $t1, 1
    beq $t4, 32, read_ident         # Neu gap dau ' ' thi tiep tuc

    sb $t4, 0($t3)
    addi $t9, $t9, 1
    j read_ident

end_read_ident:
    add $s7, $s7, $t1           # Cap nhat lai gia tri index
    beq $t9, 0, not_found           # Khong co label

    li $t2, 0               # index cho Ident
compare_ident:
    beq  $t2, $t9, end_compare_ident    # ket thuc chuoi
    li   $t1, 0             # index cho character

    add  $t3, $a1, $t2
    lb   $t3, 0($t3)            # Tung char trong Ident

    loop_Group:             # Kiem tra tung ky tu Ident co trong Group hay khong
        add $t4, $a2, $t1
        lb $t4, 0($t4)
        beq $t4, 0, not_found       # Khong co -> Khong tim thay
        beq $t4, $t3, end_loop_Group
```

```
        addi $t1, $t1, 1
        j loop_Group

    end_loop_Group:

    addi $t2, $t2, 1

    j compare_ident

end_compare_ident:

    beq $s2, 1, on_number_register

    # ----- In thong tin ra man hinh -----
    li $v0, 4
    la $a0, msg_operand
    syscall

    la $a3, ident
    li $t0, 0
    print_ident:
        beq $t0, $t9, end_print_ident
        add $t1, $a3, $t0
        lb  $t2, 0($t1)
        li $v0, 11
        add $a0, $t2, $zero
        syscall
        addi $t0, $t0, 1
        j print_ident
    end_print_ident:

    li $v0, 4
    la $a0, msg_valid
    syscall
    jr $ra

on_number_register:
    li $v0, 4
    la $a0, msg_operand
    syscall

    la $a3, ident
    li $t0, 0
    print_on_ident:
        beq $t0, $t9, end_print_on_ident
        add $t1, $a3, $t0
        lb  $t2, 0($t1)
        li $v0, 11
        add $a0, $t2, $zero
        syscall


       addi $t0, $t0, 1
       j print_on_ident
   end_print_on_ident:

   li $v0, 11
   li $a0, 40
   syscall
   jr $ra
```

- Hàm thực hiện các thao tác kiểm tra với các toán hạng là số, hoặc là nhãn. Với các thao tác cơ bản như hàm kiểm tra thanh ghi ở trên.

```
check_number_register:
    # Luu $ra de tro ve
    addi $sp, $sp, -4
    sw   $ra, 0($sp)

    li $s2, 1               # Bat kich hoat number_register

    # Check number
    la $a2, number
    jal check_ident
    nop

    la $a0, command
    add $t0, $a0, $s7           # Tro den vi tri cac instruction
    lb $t0, 0($t0)
    bne $t0, 40, not_found          # Neu ki tu khong phai la dau '('
    addi $s7, $s7, 1

    # Check register
    jal check_register
    nop
    la $a0, command
    add $t0, $a0, $s7           # Tro den vi tri cac instruction
    lb $t0, 0($t0)
    bne $t0, 41, not_found          # Neu ki tu khong phai la dau ')'
    addi $s7, $s7, 1

    # Tra lai $ra de tro ve
    lw   $ra, 0($sp)
    addi $sp, $sp, 4
    jr $ra
```

- Hàm thực hiện kiểm tra các lệnh đặc biệt như lb, sb,...

```
not_found:
    li $v0, 4
    la $a0, msg_invalid
    syscall
    j reset
```

- Hàm để kết thúc khi không tìm thấy cấu trúc lệnh trong thư viện.

```
    li $v0, 0
    li $v1, 0
  jal clean_opcode                  # jump to clean_block
  jal clean_ident
  jal clean_token
  li $a0, 0
  li $a1, 0
    li $a2, 0
    li $a3, 0
    li $t0, 0
    li $t1, 0
    li $t2, 0
    li $t3, 0
    li $t4, 0
    li $t5, 0
    li $t6, 0
    li $t7, 0
    li $t8, 0
    li $t9, 0
    li $s0, 0
    li $s1, 0
    li $s2, 0
    li $s3, 0
    li $s4, 0
    li $s5, 0
    li $s6, 0
        li $s7, 0
    li $k0, 0
    li $k1, 0

    j call_menu
```

- Hàm để đặt lại các giá trị trước khi gọi ra menu lựa chọn.

3. Kết quả:

- Dưới đây là kết quả khi kiểm tra 2 cấu trúc lệnh **add** và lệnh **sb**:

```
Nhap vao lenh Mips: add $s1,$s1,1
Opcode: add - hop le.
Toan hang: $s1 - hop le.
Toan hang: $s1 - hop le.

Lenh hop ngu khong hop le, sai khuon dang lenh !

1. Kiem tra cu phap lenh
2. Thoat
Chon:


Nhap vao lenh Mips: sb $s1,0($s3)
Opcode: sb - hop le.
Toan hang: $s1 - hop le.
Toan hang: 0($s3) - hop le.

Lenh hop ngu chinh xac !

1. Kiem tra cu phap lenh
2. Thoat
Chon:
```