**REPORT FINAL PROJECT**

**Assembly Language and Computer Architecture Lab**

**Name: Nguyen Trung Thanh ID: 20163727**

**Name: Ha Minh Duc ID: 20161067**

**Class: ICT - 02 K61**

**Teacher: Mr.Pham Ngoc Hung**

**Problem 9: Write a program to:**

**- Display the DCE image to console.**

**- Display DCE image without colors (border only).**

**- Shuffle the letters (become ECD) and display.**

**- Get color for each letter from keyboard and display with new colors.**

**\* Preparation:**

- The image is divided into 16 strings (Line1 to LineG) and astring contain only null terminator (Endline) to mark the end of the image.

- Each string contain 59 characters:

+ D’s section start from position 1 to 22.

+ C’s section start form position 23 to 42.

+ E’s section start from position 43 to 58.

+ New line character at position 59.

- The above values are stored using .eqv for easy access.

- Other strings for displaying menu and input request.

**\* Sub procedure:**

- shuffle\_print\_char: print character by character until reaching the end address (used only in option 3 – shuffle characters).

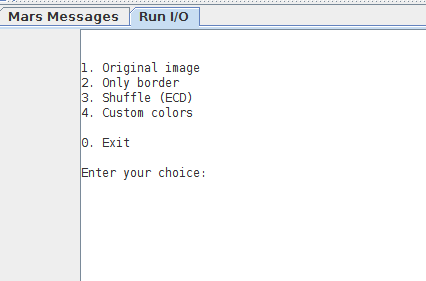
+ Occupy: v0 (syscall), a0 (store current character).

+ Argument: s0 (start address). s1 (end address).

+ Return value: none, output to console.

**\* Main program:**

- Main menu:



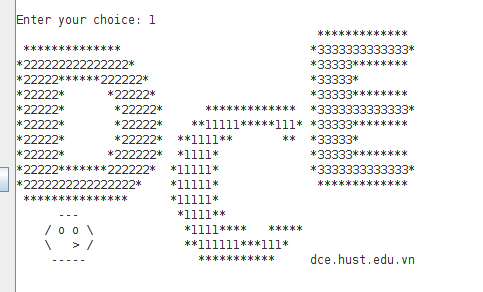
+ Enter value directly to console.

+ Display after finishing any function.

+ Values other than 0 – 4 will only display this menu again.

+ Quit the program by entering 0.

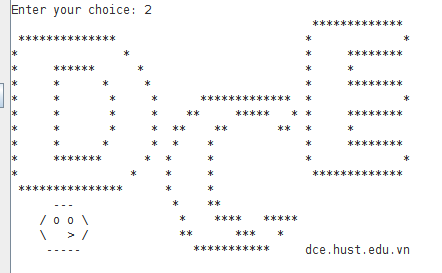
- 1. Display original image:



+ Load the address of the first line into a0, load v0 with value 4 and use syscall to print a string to console.

+ Because each line is initialized without null terminator (.ascii), except Endline after the last line so all lines will be printed out.

- 2. Display image without colors (border only):

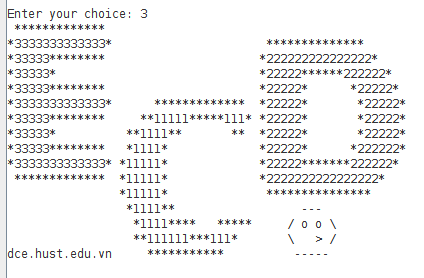


+ Start from the address of the first line, load the character from current pointer position into a0 using lb.

+ If character code > 48 (character 0) and < 57 (character 9), which means the current character is a color value, replace its value (now in a0) by 32 (space character). If not, continue.

+ Use syscall to print the character in a0 to screen, move the pointer up by 1, if it reach the end address, go back to the main menu. If not, continue the next iteration.

- 3. Shuffle text (become ECD):



+ Start from the first line, get the first position of the current line (stored in a1).

+ Print the E’s section of that line by using procedure shuffle\_print\_char:

\* The starting address of this section is first position (a1) + starting I ndex of E’s section in a line. Load this value to s0.

\* The finishing address of this section is first position (a1) + end index of E’s section in a line. Load this value to s1.

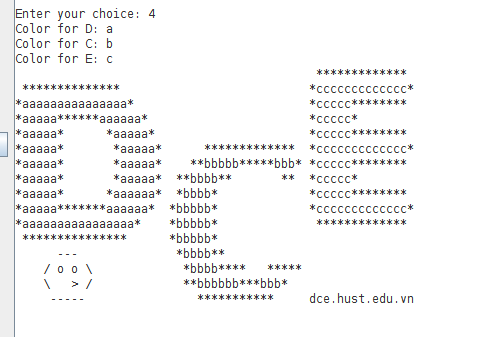
\* Call procedure.

+ The same goes for C and D.

+ Move the pointer to position of \n (a1 + line’s length – 1), print \n to console, add 1 to a1. Now a1 is at the beginning of the new line or at the Endline’s address (in this case, the image is complete, return to main menu).

+ Repeat.

- 4. Get color for each letter from keyboard and display with new colors:



+ Get input value from console using syscall, 3 characters are stored in s0, s1, s2 (run syscall again after each input to clear the leftover new line character from last input).

+ Similar to function 2, but check for update by fixed values (49 - ‘1’, 50 - ‘2’, 51 - ‘3’) and replace with its inputted value.

**\* Register used:**

- v0: system call code, inputted character/integer.

- a0: hold the value which is going to be printed out to console (string, character).

- a1: address of the first line.

- a2: address of Newline (the end of the image).

- a3: hold address of loaded data when a0, a1, a2 are occupied.

- s0, s1: start address and end address (function 3), inputted color of D and C (function 4).

- s2: inputted color of E (function 4).

- s3, s4, s5: temporary values used for comparison.