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

BÁO CÁO
FINAL PROJECT

Học phần: Thực hành kiến trúc máy tính

Giảng viên hướng dẫn: Ths. Lê Bá Vui

Mã lớp: 139363

Nhóm sinh viên thực hiện:

Hoàng Đức Gia Hưng	20215062
Mai Minh Khôi	20210492

Hà Nội, tháng 7 năm 2023

Phân chia công việc trong nhóm

Họ và tên	MSSV	Email	Công việc thực hiện
Hoàng Đức Gia Hưng	20215062	Hung.HDG215062@sis.hust.edu.vn	Bài 9: Vẽ hình bằng ký tự ASCII
Mai Minh Khôi	20210492	Khoi.MM210492@sis.hust.edu.vn	Bài 2: Vẽ hình trên màn hình Bitmap

Nội Dung

I. Vẽ hình trên màn hình Bitmap

1. Đề bài

Viết chương trình vẽ một quả bóng hình tròn di chuyển trên màn hình mô phỏng Bitmap của Mars. Nếu đối tượng đập vào cạnh của màn hình thì sẽ di chuyển theo chiều ngược lại.
Yêu cầu:

- Thiết lập màn hình ở kích thước 512x512. Kích thước pixel 1x1.
- Chiều di chuyển phụ thuộc vào phím người dùng bấm, gồm có (di chuyển lên (W), di chuyển xuống (S), sang trái (A), sang phải (D), tăng tốc độ (Z), giảm tốc độ (X) trong bộ giả lập Keyboard and Display MMIO Simulator).
- Vị trí bóng ban đầu ở giữa màn hình.

Gợi ý: Để làm một đối tượng di chuyển thì chúng ta sẽ xóa đối tượng ở vị trí cũ và vẽ đối tượng ở vị trí mới. Để xóa đối tượng chúng ta chỉ cần vẽ đối tượng đó với màu là màu nền.

2. Phân tích cách thực hiện.

a. Ý tưởng thực hiện:

- Ở đây chúng ta sẽ sử dụng Keyboard and Display và Bitmap.

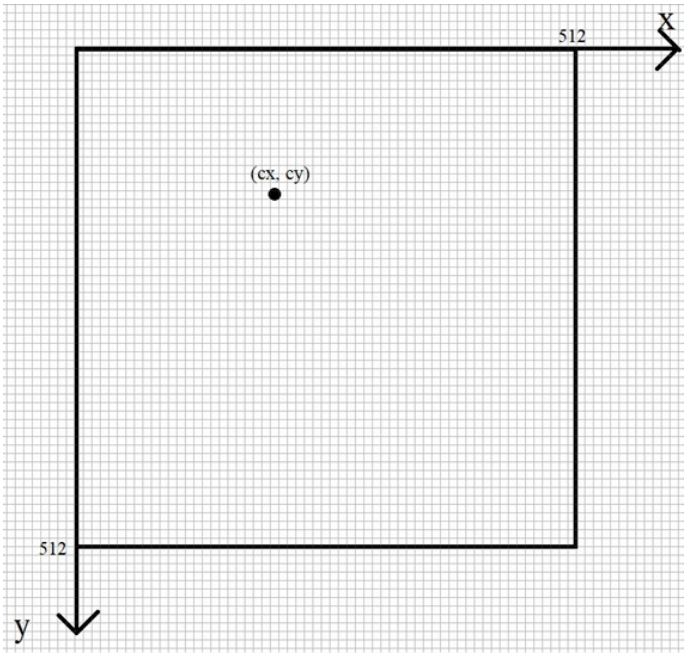
    - Keyboard and Display: Nhận các nút điều hướng.

    - Bitmap: Vẽ hình quả bóng, là nơi hiển thị quả bóng sẽ di chuyển.

- Các thuật toán sẽ sử dụng: Thuật toán Bresenham để vẽ đường tròn cho quả bóng.

    Thuật toán Bresenham: Vẽ hình tròn bán kính trên màn hình Bitmap

        +) Ta vẽ hình tròn bán kính 20.

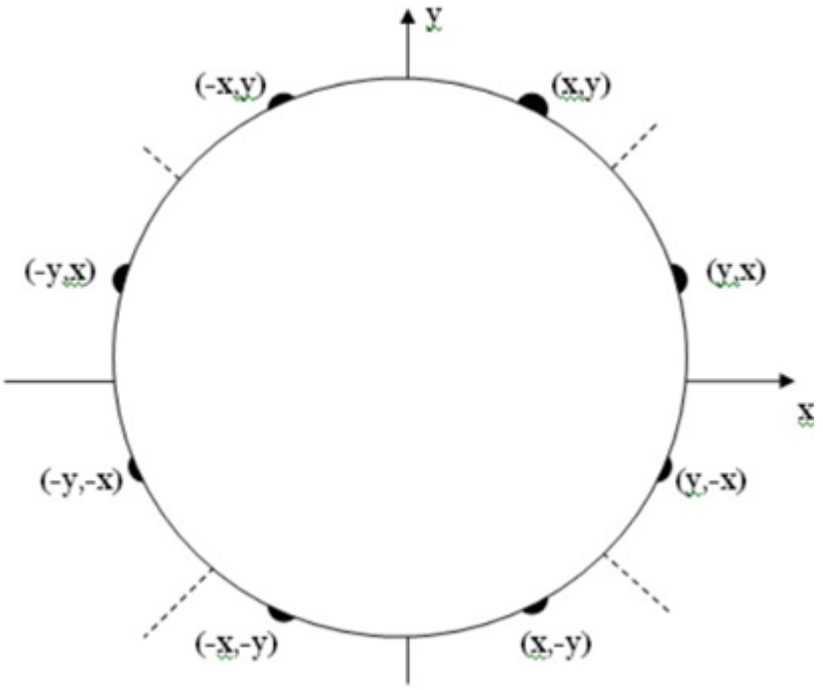


Toạ độ hoá màn hình bitmap

+) Ta đưa hệ tọa độ Oxy vào màn hình Bitmap, gọi tọa độ tâm mỗi hình tròn cần vẽ là (cx, cy).

+) Tọa độ mỗi điểm cần tô màu là (cx+x, cy+y).

+) Do tính đối xứng của đường tròn nên ta chỉ cần vẽ 1/8 đường tròn, sau đó lấy đối xứng qua 2 trục tọa độ và 2 đường phân giác.



+) Với mỗi giá trị x, y không âm được thỏa mãn, ta sẽ vẽ được 8 điểm:

1. (cx + x, cy + y)
2. (cx – x, cy + y)
3. (cx – x, cy - y)
4. (cx + x, cy - y)
5. (cx + y, cy + x)
6. (cx – y, cy + x)
7. (cx – y, cy - x)
8. (cx + y, cy - x)

+) Để có được các cặp giá trị (x, y), ta áp dụng thuật toán Bresenham:

Bước 1: Khởi tạo $(x_1, y_1) = (20, 0)$, $p_1 = 3 – 2R = -37$

Bước i+1:

· Nếu $p_i < 0$ thì $(x_{i+1}, y_{i+1}) = (x_i, y_{i+1})$; $p_{i+1} = p_i + 4y_i + 6$

· Nếu $p_i > 0$ thì

$(x_{i+1}, y_{i+1}) = (x_i - 1, y_i + 1)$;

$$p_{i+1} = p_i + 4(y_i - x_i) + 10$$

3. Mã nguồn

- Cài đặt thông số mặc định cho bộ KeyBoard and Display và Bitmap

```
# Bo khoi dong mac dinh
.eqv KEY_CODE 0xFFFF0004        # ASCII code to show, 1 byte
.eqv KEY_READY 0xFFFF0000       # =1 if has a new keycode ?
                                # Auto clear after lw
.eqv DISPLAY_CODE 0xFFFF000C    # ASCII code to show, 1 byte
.eqv DISPLAY_READY 0xFFFF0008   # =1 if the display has already to do
                                # Auto clear after sw
```

- Tọa độ tâm của hình tròn nằm chính giữa màn hình, vẽ hình tròn với bán kính 20

```
#circle:
# Ta set up hình tròn ở vị trí chính giữa màn hình
addi    $a0, $0, 256        #x = 256
addi    $a1, $0, 256        #y = 256
addi    $a2, $0, 20         #r = 20
addi    $s0, $0, 0x00FF0000     # $s0 là màu của hình tròn, ta dùng màu đỏ
jal     DrawCircle
nop
```

- Bắt đầu chương trình

```
li $a3,2
moving:
        # nhan ki tu tu ban phim va di chuyen
# so sanh trong ma ASCII
beq $t0,97,left         # 97 = 'a'
beq $t0,100,right       # 100 = 'd'
beq $t0,115,down        # 115 = 's'
beq $t0,119,up          # 119 = 'w'
beq $t0,122,increase    # 122 = 'x'
beq $t0,120,decrease    # 120 = 'z'
j Input
```

Ta lần lượt kiểm tra phím được nhập vào là phím nào (so sánh với mã ascii) và thực hiện chức năng tương ứng với từng phím.

- Di chuyển sang trái (Phím a)

```
left:                               # di chuyen sang trai
    addi $s0,$0,0x00000000          # Chuyển hình tròn thành màu đen
    jal DrawCircle                  # xóa hình tròn vị trí cũ, bằng cách tô đen
    sub $a0,$a0,$a3                 # giảm hoành độ để di chuyển sang bên trái
    add $a1,$a1, $0                 # tung độ tâm đường tròn giữ nguyên
    addi $s0,$0,0x00FF0000          # đặt lại màu đỏ cho hình tròn
    jal DrawCircle                  # vẽ hình tròn ở vị trí mới
    jal Pause
    bltu $a0,20,ToTheRight # nếu khoảng cách từ tâm đến trục < r = 20 thì đổi hướng
    j Input
```

Ta thay đổi màu của hình tròn thành màu đen (trùng với màu của màn hình bitmap) để xóa hình tròn tại vị trí hiện tại. Tiếp đó ta thay đổi vị trí của tâm hình tròn (giảm hoành độ). Chuyển màu của hình tròn sang màu ban đầu và gọi chương trình con vẽ hình tròn. Chương trình con PAUSE để cho hình tròn không bị di chuyển quá nhanh. Nếu mà khoảng cách từ tâm đến trục tung nhỏ hơn bán kính thì phải đổi hướng di chuyển.

```
ToTheRight:
    li $t3 100
    sw $t3,0($k0)    # thay đổi giá trị ở địa chỉ KEY_CODE để nhảy đến right
    j Input
```

Tương tự với sang phải, lên trên, xuống dưới.

- Sang phải

```
right:                              # di chuyển sang phải
    addi $s0,$0,0x00000000          # Chuyển hình tròn thành màu đen
    jal DrawCircle                  # xoá hình tròn cũ
    add $a0,$a0,$a3                 # tăng hoành độ để di chuyển sang bên trái
    add $a1,$a1, $0                 # tung độ giữ nguyên
    addi $s0,$0,0x00FF0000
    jal DrawCircle                  # vẽ hình tròn mới
    jal Pause
    bgtu $a0,492,ToTheLeft # nếu khoảng cách từ tâm đến trục > 512 - r = 492 thì đổi hướng
    j Input
```

```
ToTheLeft:
    li $t3 97
    sw $t3,0($k0)     # thay đổi giá trị ở địa chỉ KEY_CODE để nhảy đến left
    j Input
```

- Lên trên

```
up:                                  # di chuyển lên trên
    addi $s0,$0,0x00000000
    jal DrawCircle                   # xoá hình tròn cũ
    sub $a1,$a1,$a3                  # tung độ giảm 1
    add $a0,$a0,$0                   # hoành độ giữ nguyên
    addi $s0,$0,0x00FF0000
    jal DrawCircle                   # vẽ hình tròn mới
    jal Pause
    bltu $a1,20,TurnDown    # đập thành thì nảy xuống dưới
    j Input

TurnDown:
    li $t3 115
    sw $t3,0($k0)     # thay đổi giá trị ở địa chỉ KEY_CODE để nhảy đến down
    j Input
```

- Xuống dưới

```
down:                                # di chuyển xuống dưới
      addi $s0,$0,0x00000000
      jal DrawCircle                 # xoá hình tròn cũ
      add $a1,$a1,$a3                # tung độ tăng them a3
      add $a0,$a0,$0                 # hoành độ giữ nguyên
      addi $s0,$0,0x00FF0000
      jal DrawCircle                 # vẽ hình tròn mới
      jal Pause
      bgtu $a1,492,TurnUp     # đập thành thì nảy lên trên
      j Input

TurnUp:
    li $t3 119
    sw $t3,0($k0)    # thay đổi giá trị ở địa chỉ KEY_CODE để nhảy đến up
    j Input
```

- Khi ấn ‘z’ hoặc ‘x’ thì sẽ tăng tốc hoặc giảm tốc. Ban đầu mỗi lần di chuyển thì tung độ hoặc hoành độ của tâm hình tròn thay đổi 2 đơn vị do đó khi ấn z ta thay đổi hoành độ hoặc tung độ 3 đơn vị ⇒ hình tròn sẽ di chuyển nhanh hơn. Để giữ nguyên hướng di chuyển trước khi ấn phím ‘z’ hoặc ‘x’ thì ta dùng thanh ghi $t7 để lưu giữ lại.

```
increase:
      li $a3,3
      add $t0,$0,$t7
      j moving
```

- Để giữ nguyên hướng di chuyển trước khi ấn phím ‘z’ hoặc ‘x’ thì ta dùng thanh ghi $t7 để lưu giữ lại.

```
Input:
    ReadKey: lw $t0, 0($k0) # $t0 = [$k0] = KEY_CODE
    # Ta lưu lại hướng di chuyển trước đó, để khi tăng tốc quả bóng giữ nguyên hướng di chuyển
    beq $t0,97,store
    beq $t0,100,store
    beq $t0,115, store
    beq $t0,119, store
    j moving
store:
        add $t7, $0, $t0
        j moving
```

- Chương trình con Drawcircle
  Vùng lệnh DrawCirle để vẽ hình tròn, vùng lệnh này được viết dựa trên thuật toán Bresenham. Đầu tiên sẽ giảm giá trị thanh ghi $sp để tạo ô nhớ lưu trữ trong ngăn xếp, để lưu giá trị của các thanh ghi $ra, nhằm mục đích có thể quay lại địa chỉ của hàm sau khi thực hiện hàm con.

    - Khởi tạo giá trị ban đầu cho tọa độ (x, y) = (20, 0). Giá tri của $p_1 = 3 – 2r$
    - Bắt đầu vòng lặp DrawCircleLoop. Điều kiện dừng vòng lặp sẽ là khi giá trị của y lớn hơn giá trị của x thì vẽ xong vòng trong và thoát khỏi vòng lặp. Đầu tiên gọi hàm plot8points để vẽ 8 điểm đối xứng của đường tròn.
    - Hàm plot8points: Tạo ngăn xếp lưu trữ địa chỉ thanh ghi $ra để có thể quay về chương trình trước. Trong hàm plot8points sẽ gọi hàm plot4points để vẽ 4 điểm đối xứng qua trục tung và trục hoành. Sau đó hàm này thực hiện tiếp việc hoán đổi giá trị của biến x và y để vẽ thêm 4 điểm đối xứng qua phân giác của đường tròn.
    - Hàm plot4points: Hàm nhận đầu vào là tọa độ x và y của một điểm nằm trên đường tròn, sau đó sử dụng hàm SetPixel để tô màu cho 4 điểm trên đường tròn. Các điểm đối xứng của (x, y) là (x, -y), (-x, y),(-x, -y).
    - Hàm SetPixel: Hàm này có công việc là gán giá trị màu ở thanh ghi $s0 cho địa chỉ bộ nhớ tương ứng với pixel có tọa độ ($a0, $a2) trên màn hình. Đầu tiên là lưu địa chỉ trở về trong thanh thi $ra vào ngăn xếp. Sau đấy tính toán địa chỉ bộ nhớ của pixel cần tô màu bằng cách sử dụng công thức:

· Địa chỉ bắt đầu của bộ nhớ màn hình được lưu trong thanh ghi $s1

· Nhân giá trị x ($a0) với 4 (vì mỗi pixel chiếm 4 byte) để tính toán offset ngang.

· Nhân giá trị y ($a2) với số byte trên mỗi hàng để tính toán offset dọc.

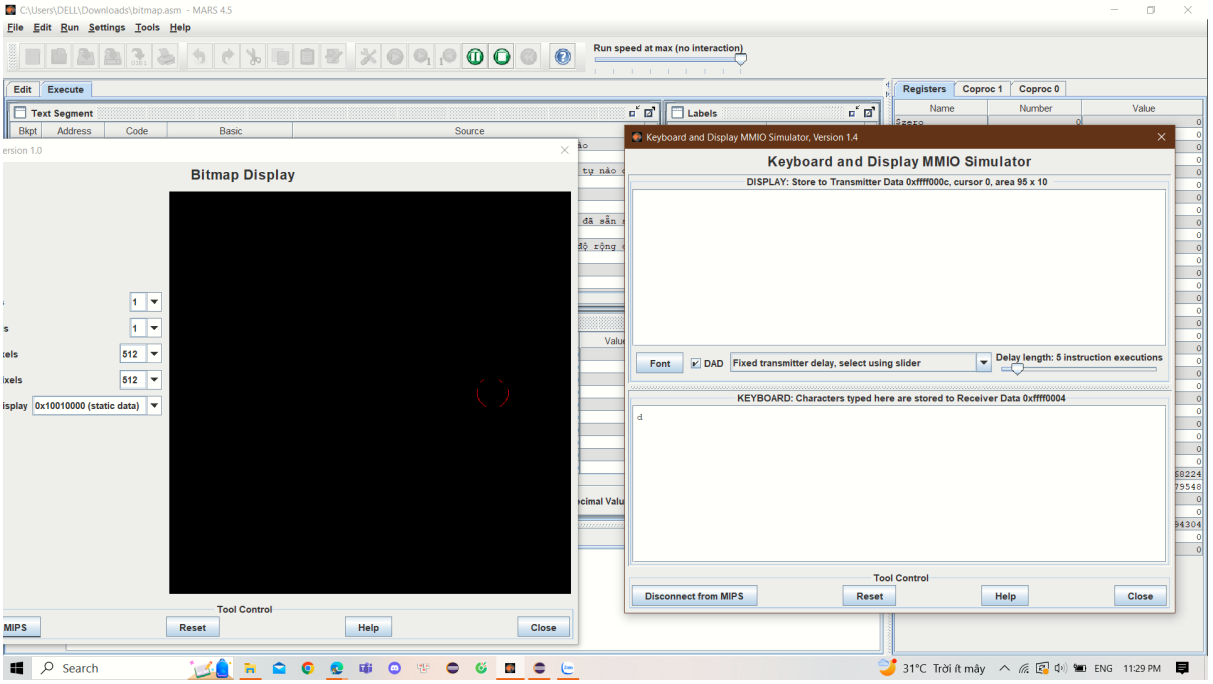
· Cộng offset ngang và offset dọc để tìm địa chỉ cuối cùng của pixel.

- Lưu giá trị màu ($s0) vào địa chỉ bộ nhớ tương ứng.
- Khôi phục các giá trị thanh ghi ban đầu và địa chỉ trở về.
- Quay lại vòng lặp. Tiếp tục tính toán các điểm (x, y) tiếp theo, rồi vẽ các điểm mới. Tiếp theo tăng y = y + 1. Nếu p < 0 thì sử dụng công thức

p = p + 4y + 6. Ngược lại p = p + 4(y - x) + 10 và x = x – 1.

- Sau khi vòng lặp kết thúc sẽ nhảy đến hàm exitDrawCircle, hàm này sẽ lấy ra và trả về ngăn xếp ban đầu.

Kết quả



II. Vẽ hình bằng kí tự ASCII

1. Đề bài

9. Vẽ hình bằng kí tự ASCII

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.

```
                                          *************
**************                           *3333333333333*
*222222222222222*                        *33333********
*22222******222222*                      *33333*
*22222*      *22222*                     *33333********
*22222*       *22222*      *************  *3333333333333*
*22222*       *22222*    **11111*****111* *33333********
*22222*       *22222*  **1111**       **  *33333*
*22222*      *222222*  *1111*             *33333********
*22222*******222222*  *11111*             *3333333333333*
*2222222222222222*    *11111*              *************
***************       *11111*
      ---              *1111**
    / o o \             *1111****   *****
    \   > /             **111111***111*
     -----                ***********    dce.hust.edu.vn
```

- 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. Cách thực hiện :

Tách hình ảnh thành từng string theo từng dòng.

a) Hiển thị hình ảnh

- Sử dụng vòng lặp để in ra từng string từ 1 -> 16 tương ứng với từng dòng

b) Hiển thị hình ảnh chỉ có viền, không có màu

- Duyệt từng ký tự theo từng dòng

- Nếu gặp chữ số (xét theo bảng ascii với 0 tương ứng với 48 và 9 tương ứng với 57) thì thay ký tự đó bằng space để xóa chữ số tương ứng với màu

- Nếu gặp ký tự khác chữ số thì bỏ qua việc thay đổi và in ra như bình thường

c) Đảo vị trí D với E và hiển thị ra màn hình

- Chia mỗi string thành 4 phần:

· Chữ D: Từ cột 0 đến cột 21

· Chữ C: Từ cột 22 đến cột 41

· Chữ E: Từ cột 42 đến cột 57

· Còn lại: Từ cột 57 đến hết

- In từng kí tự theo từng dòng lần lượt các kí tự từ vị trí 42 -> 57, sau đó từ cột 22 -> 41, sau đó từ cột 0 -> 21 và từ cột 57 -> 62

d) Đổi màu

- Lưu các màu hiện tại của D, C, E lần lượt vào các thanh ghi $t5, $t6, $t7

- Nhập màu muốn thay đổi lần lượt cho D, C, E và lưu vào các thanh ghi $s3, $s4, $s5. Nếu số nhập không phải màu từ 0 -> 9 thì nhập lại

- Duyệt từng kí tự theo từng dòng lần lượt theo từng chữ cái

· Chữ D (0->21): so sánh kí tự hiện tại với thanh ghi $t5 lưu màu lúc đầu của D , nếu bằng nhau thì lưu giá trị màu mới ($s3) vào kí tự hiện tại, nếu không thì duyệt kí tự tiếp theo. Nếu kí tự hiện tại ở vị

trí 23 thì duyệt sang chữ C

· Tương tự với chữ C (22 -> 41), E (42 -> 57)

- Sau khi duyệt hết 1 dòng thì in ra và duyệt dòng tiếp theo cho đến khi hết 16 dòng

3. Mã nguồn :

```
.data

        String1:  .asciiz  "                                  *************    \n"
        String2:  .asciiz  "**************                    *3333333333333*
\n"
        String3:  .asciiz  "*222222222222222*
*33333********    \n"
        String4:  .asciiz  "*22222******222222*                  *33333*
\n"
        String5:  .asciiz  "*22222*      *22222*                 *33333********
\n"
        String6:  .asciiz  "*22222*      *22222*      *************
*3333333333333*   \n"
        String7:  .asciiz  "*22222*      *22222*    **11111*****111*
*33333********    \n"
        String8:  .asciiz  "*22222*      *22222*  **1111**       **  *33333*
\n"
        String9:  .asciiz  "*22222*      *222222*  *1111*
*33333********    \n"
        String10: .asciiz  "*22222*******222222*  *11111*
*3333333333333*   \n"
```

```
        String11: .asciiz  "*2222222222222222*    *11111*
*************    \n"
        String12: .asciiz  "***************       *11111*                    \n"
        String13: .asciiz  "      ---            *1111**                    \n"
        String14: .asciiz  "    / o o \\          *1111****   *****           \n"
        String15: .asciiz  "    \\  >  /          **111111***111*             \n"
        String16: .asciiz  "     -----            ***********    dce.hust.edu.vn
\n"


        Message0:       .asciiz "------------IN CHU-----------\n"
        P1:             .asciiz"1. In ra chu\n"
        P2:             .asciiz"2. In ra chu rong\n"
        P3:             .asciiz"3. Thay doi vi tri\n"
        P4:             .asciiz"4. Doi mau cho chu\n"
        Exit:           .asciiz"5. Exit\n"
        Input:          .asciiz"Nhap gia tri: "
        ChuD:               .asciiz"Nhap màu cho chu D(0->9): "
        ChuC:               .asciiz"Nhap màu cho chu C(0->9): "
        ChuE:               .asciiz"Nhap màu cho chu E(0->9): "
.text

        li $t5, 50 #t5 mau chu hien tai cua chu D
        li $t6, 49 #t6 mau chu hien tai cua chu C
        li $t7, 51 #t7 mau chu hien tai cua chu E
```

```
main:
        la $a0, Message0        # nhap menu
        li $v0, 4
        syscall

        la $a0, P1
        li $v0, 4
        syscall

        la $a0, P2
        li $v0, 4
        syscall

        la $a0, P3
        li $v0, 4
        syscall

        la $a0, P4
        li $v0, 4
        syscall

        la $a0, Exit
        li $v0, 4
        syscall
```

```
la $a0, Input
li $v0, 4
syscall

li $v0, 5
syscall

Case1:
        addi $v1, $0, 1
        bne $v0, $v1, Case2
        j Menu1
Case2:
        addi $v1, $0, 2
        bne $v0, $v1, Case3
        j Menu2
Case3:
        addi $v1, $0, 3
        bne $v0 $v1 Case4
        j Menu3
Case4:
        addi $v1, $0, 4
        bne $v0, $v1, Case5
        j Menu4
Case5:
        addi $v1 $0 5
```

```
        bne $v0 $v1 default
        j Exitall
    default:
        j main

Menu1:
    addi $t0, $0, 0     #bien dem =0
    addi $t1, $0, 16 # Khởi tạo biến t1 = số hàng = 16.

    la $a0, String1 # Lấy địa chỉ của chuỗi String1 và lưu vào thanh ghi $a0
Loop:
    beq $t1, $t0, main # Kiểm tra nếu t1 bằng t0, nhảy tới nhãn main.
    li $v0, 4
    syscall

    addi $a0, $a0, 62 # Tăng địa chỉ chuỗi a0 lên 62 (di chuyển sang hàng tiếp theo).
    addi $t0, $t0, 1 #  Tăng biến đếm t0 lên 1
    j Loop # Quay lại vòng lặp

Menu2:      addi $s0, $0, 0    #bien dem tung hang =0
    addi $s1, $0, 16
    la $s2, String1     # $s2 la dia chi cua string1

Loop2:      beq $s1, $s0, main
```

```
        addi $t0, $0, 0     # $t0 la bien dem tung kí tu cua 1 hang =0
        addi $t1, $0, 62 # $t1 max 1 hang là 62 ki tu

print_1_line:
        beq $t1, $t0, End #t1 = t0 tuc la da het 1 hang thi chuyen den End
        lb $t2, 0($s2) #load tung ki tu vao $t2
        bgt    $t2, 47, Checklaso #neu >= 0 (trong bang ascii) thi nhay den
Checklaso
        j print_1_char
Checklaso:
        bgt    $t2, 57, print_1_char #neu lon hon 9 thi giu nguyen nhay den
print_1_char de in ra
        addi $t2, $0, 0x20 # thay doi $t2 thanh dau cach
        j print_1_char

print_1_char:
        li $v0, 11 # in tung ki tu
        addi $a0, $t2, 0
        syscall

        addi $s2, $s2, 1 #sang chu tiep theo
        addi $t0, $t0, 1# bien dem chu
        j print_1_line

End:  addi $s0, $s0, 1 # tang bien dem hang lên 1
        j Loop2
```

```
Menu3:      addi $s0, $0, 0    #bien dem tung hàng =0
        addi $s1, $0, 16 # Khởi tạo biến t1 = số hàng = 16.
        la $s2, String1 #$s2 luu dia chi cua string1

Loop3:      beq $s1, $s0, main
        #tao thanh 3 string nho
        sb $0, 21($s2) # Gán giá trị 0 vào vị trí 21 của chuỗi String1.
        sb $0, 41($s2) # Gán giá trị 0 vào vị trí 41 của chuỗi String1.
        sb $0, 57($s2) # Gán giá trị 0 vào vị trí 57 của chuỗi String1.
        #doi vi tri
        li $v0, 4
        la $a0, 42($s2) # Lấy địa chỉ của kí tự tại vị trí 42 của chuỗi String1 và lưu vào thanh ghi $a0 để in kí tự E.
        syscall

        li $v0, 4
        la $a0, 22($s2) # Lấy địa chỉ của kí tự tại vị trí 22 của chuỗi String1 và lưu vào thanh ghi $a0 để in kí tự C.
        syscall

        li $v0, 4
        la $a0, 0($s2) # Lấy địa chỉ của kí tự tại vị trí 0 của chuỗi String1 và lưu vào thanh ghi $a0 để in kí tự D.
        syscall
```

li $v0, 4

la $a0, 58($s2) # Lấy địa chỉ của kí tự tại vị trí 58 của chuỗi String1 và lưu vào thanh ghi $a0 để in kí tự.

syscall

ghep lai thanh string ban dau

addi $t1, $0, 0x20 # 0x20: Gán giá trị 0x20 vào thanh ghi t1 (dấu cách

sb $t1, 21($s2) # Gán giá trị trong thanh ghi t1 vào vị trí 21 của chuỗi String1

sb $t1, 41($s2) # Gán giá trị trong thanh ghi t1 vào vị trí 41 của chuỗi String1.

sb $t1, 57($s2) # Gán giá trị trong thanh ghi t1 vào vị trí 57 của chuỗi String1.

addi $s0, $s0, 1 # Tăng biến đếm hàng s0 lên 1.

addi $s2, $s2, 62 # Tăng địa chỉ của chuỗi String1 lên 62 (di chuyển sang chuỗi kế tiếp).

j Loop3 # Nhảy tới nhãn Loop3 để tiếp tục vòng lặp.

Menu4:

NhapmauD:	li $v0, 4

la $a0, ChuD

syscall

li $v0, 5		# lay mau cua ki tu D

syscall

blt $v0,0, NhapmauD # Kiểm tra nếu giá trị trong $v0 nhỏ hơn 0, quay lại nhãn NhapmauD

bgt $v0,9, NhapmauD # Kiểm tra nếu giá trị trong $v0 lớn hơn 9, quay lại nhãn NhapmauD.

addi $s3, $v0, 48# Thêm 48 vào giá trị trong $v0 và lưu kết quả vào thanh ghi $s3 (chuyển giá trị thành mã ASCII tương ứng).

```
NhapmauC:       li $v0, 4
                la $a0, ChuC
                syscall

                li $v0, 5               # lay mau cua ki tu C
                syscall
```

blt $v0, 0, NhapmauC # Kiểm tra nếu giá trị trong $v0 nhỏ hơn 0, quay lại nhãn NhapmauC.

bgt $v0, 9, NhapmauC # Kiểm tra nếu giá trị trong $v0 lớn hơn 9, quay lại nhãn NhapmauC.

addi $s4, $v0, 48# Thêm 48 vào giá trị trong $v0 và lưu kết quả vào thanh ghi $s4 (chuyển giá trị thành mã ASCII tương ứng).

```
NhapmauE:       li $v0, 4
                la $a0, ChuE
                syscall
```

li $v0, 5 # lay mau cua ki tu E

syscall

blt $v0, 0, NhapmauE # Kiểm tra nếu giá trị trong $v0 nhỏ hơn 0, quay lại nhãn NhapmauE.

bgt $v0, 9, NhapmauE # Kiểm tra nếu giá trị trong $v0 lớn hơn 9, quay lại nhãn NhapmauE.

addi $s5, $v0, 48# Thêm 48 vào giá trị trong $v0 và lưu kết quả vào thanh ghi $s5 (chuyển giá trị thành mã ASCII tương ứng).

addi $s0, $0, 0 # bien dem tung hàng =0

addi $s1, $0, 16 # Khởi tạo biến s1 bằng 16.

la $s2,String1 # Lấy địa chỉ của chuỗi String1 và lưu vào thanh ghi $s2.

li $a1, 48 # Đặt giá trị 48 (mã ASCII của số 0) vào thanh ghi $a1.

li $a2, 57 # Đặt giá trị 57 (mã ASCII của số 9) vào thanh ghi $a2.

Loop4: beq $s1, $s0, update_color # Kiểm tra nếu s1 bằng s0, nhảy tới nhãn update_color.

addi $t0, $0, 0 # Khởi tạo biến đếm kí tự trong một hàng t0 bằng 0.

addi $t1, $0, 62 # Đặt giá trị 62 vào thanh ghi $t1 (số kí tự tối đa trong một hàng).

store_1_line_change_color:

beq $t1, $t0, End_change_color # Kiểm tra nếu t1 bằng t0, nhảy tới nhãn End_change_color.

lb $t2, 0($s2) # Đọc giá trị từ vị trí hiện tại trong chuỗi String1 và lưu vào thanh ghi $t2.

CheckD: bgt $t0, 21, CheckC # Kiểm tra nếu t0 lớn hơn 21, nhảy tới nhãn CheckC.

beq $t2, $t5, fixD # Kiểm tra nếu giá trị trong $t2 bằng $t5, nhảy tới nhãn fixD.

j Tmpdoimau # Nhảy tới nhãn Tmpdoimau để thực hiện các lệnh tiếp theo.

CheckC: bgt $t0, 41, CheckE # Kiểm tra nếu t0 lớn hơn 41, nhảy tới nhãn CheckE.

beq $t2, $t6, fixC # Kiểm tra nếu giá trị trong $t2 bằng $t6, nhảy tới nhãn fixC.

j Tmpdoimau # Nhảy tới nhãn Tmpdoimau để thực hiện các lệnh tiếp theo.

CheckE: beq $t2, $t7, fixE # Kiểm tra nếu giá trị trong $t2 bằng $t7, nhảy tới nhãn fixE

j Tmpdoimau # Nhảy tới nhãn Tmpdoimau để thực hiện các lệnh tiếp theo.

fixD: sb $s3, 0($s2) # Lưu giá trị từ thanh ghi $s3 (mã ASCII của màu cho kí tự D) vào vị trí hiện tại trong chuỗi String1.

j Tmpdoimau

fixC: sb $s4, 0($s2) # Lưu giá trị từ thanh ghi $s4 (mã ASCII của màu cho kí tự C) vào vị trí hiện tại trong chuỗi String1.

j Tmpdoimau

fixE: sb $s5, 0($s2) # Lưu giá trị từ thanh ghi $s5 (mã ASCII của màu cho kí tự E) vào vị trí hiện tại trong chuỗi String1.

j Tmpdoimau

```
Tmpdoimau:
        addi $s2, $s2, 1 #sang chu tiep theo
        addi $t0, $t0, 1# bien dem chu
        j store_1_line_change_color
End_change_color:
        li $v0, 4
        addi $a0, $s2, -62 # Đặt địa chỉ bắt đầu của chuỗi String1 vào $a0 để in chuỗi đã thay đổi màu.
        syscall
        addi $s0, $s0, 1 # Tăng biến đếm hàng s0 lên 1.
        j Loop4 # Nhảy tới nhãn Loop4 để tiếp tục vòng lặp.
update_color:
        move $t5, $s3 # Di chuyển giá trị trong $s3 vào $t5.
        move $t6, $s4 # Di chuyển giá trị trong $s4 vào $t6
        move $t7, $s5 # i chuyển giá trị trong $s5 vào $t7
        j main          # Nhảy tới nhãn main để tiếp tục thực hiện các lệnh trong chương trình chính.
Exitall:
```

4. Kết quả:


----------IN CHU----------
1. In ra chu
2. In ra chu rong
3. Thay doi vi tri
4. Doi mau cho chu
5. Exit
Nhap gia tri: 1
dce.hust.edu.vn
----------IN CHU----------
1. In ra chu
2. In ra chu rong
3. Thay doi vi tri
4. Doi mau cho chu
5. Exit
Nhap gia tri: 2
dce.hust.edu.vn

```
----------IN CHU----------
1. In ra chu
2. In ra chu rong
3. Thay doi vi tri
4. Doi mau cho chu
5. Exit
Nhap gia tri: 3
 ************
*3333333333333*                  **************
*33333********                    *222222222222222*
*33333*                           *22222******222222*
*33333********                    *22222*      *22222*
*3333333333333*    *************  *22222*       *22222*
*33333********    **111111*****111**22222*       *22222*
*33333*          **1111**       ** *22222*       *22222*
*33333********  *1111*             *22222*      *222222*
*3333333333333**11111*             *22222******222222*
 ************ *11111*             *2222222222222222*
              *11111*             ****************
               *1111**                  ---
                *1111****   *****     / o o \
                 **111111***111*      \   > /
dce.hust.edu.vn     ***********        -----
------------IN CHU----------
1. In ra chu
2. In ra chu rong
3. Thay doi vi tri
4. Doi mau cho chu
5. Exit
Nhap gia tri: 4
Nhap màu cho chu D(0->9): 7
Nhap màu cho chu C(0->9): 8
Nhap màu cho chu E(0->9): 9
                                          ************
**************                           *9999999999999*
*777777777777777*                         *99999********
*77777******777777*                       *99999*
*77777*      *77777*                      *99999********
*77777*       *77777*     *************   *9999999999999*
*77777*       *77777*    **88888*****888* *99999********
*77777*       *77777*  **8888**       **  *99999*
*77777*      *777777*  *8888*             *99999********
*77777*******777777*  *88888*             *9999999999999*
*7777777777777777*    *88888*              *************
***************       *88888*
       ---             *8888**
     / o o \            *8888****   *****
     \   > /             **888888***888*
      -----                ***********    dce.hust.edu.vn
```

```
------------IN CHU-----------
1. In ra chu
2. In ra chu rong
3. Thay doi vi tri
4. Doi mau cho chu
5. Exit
Nhap gia tri: 5

-- program is finished running (dropped off bottom) --
```