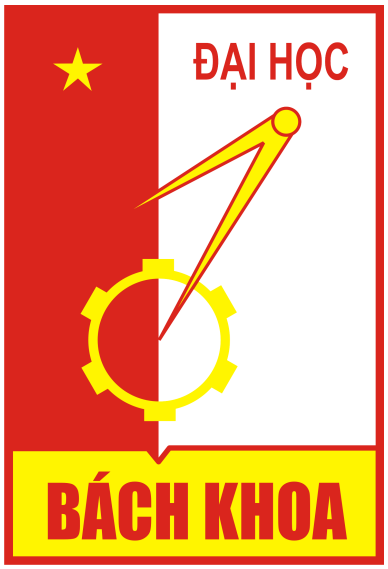
ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



BÁO CÁO BÀI TẬP CUỐI KÌ

THỰC HÀNH KIẾN TRÚC MÁY TÍNH IT3280E

Học kì 20241 - Năm học: 2024 - 2025

Giảng viên hướng dẫn: **TS. Lê Bá Vui**

Sinh viên: **Nguyễn Trọng Minh Phương - 20225992**

Mã lớp: **152003**

Hà Nội, 2024

Mục lục

1 Drawing shape using ASCII characters

1.1 Đề bài

7. Drawing shape using ASCII characters

Given a picture translated to ASCII characters as follows, this is the shapes of DCE with border * and colors are digits.

```
                                          *************
**************                           *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
```

- Show this picture in the console window.
- Change the picture so that DCE has only a border without color inside.
- Change the order of DCE to ECD.
- Enter the new color number from the keyboard, update the picture with new colors.

Note: Except the memory used to store the picture in source code, do not use any extra memory space.

1.2 Thuật toán và hàm

1.2.1 Khai báo dữ liệu

```
1  .data
2  line1:  .asciz  "                                          *************
       \n"
3  line2:  .asciz  "**************
       *3333333333333*\n"
4  line3:  .asciz  "*222222222222222*                        *33333********
       \n"
5  line4:  .asciz  "*22222******222222*                      *33333*
       \n"
6  line5:  .asciz  "*22222*       *22222*                    *33333********
       \n"
7  line6:  .asciz  "*22222*         *22222*      *************
       *3333333333333*\n"
8  line7:  .asciz  "*22222*         *22222*    **11111*****111*  *33333********
       \n"
9  line8:  .asciz  "*22222*         *22222*   **1111**       **   *33333*
       \n"
10 line9:  .asciz  "*22222*        *222222*   *1111*              *33333********
       \n"
11 line10: .asciz  "*22222*******222222*    *11111*
       *3333333333333*\n"
12 line11: .asciz  "*2222222222222222*      *11111*               *************
       \n"
13 line12: .asciz  "***************         *11111*
       \n"
```

```
line13: .asciz  "      ---               *1111**
    \n"
line14: .asciz  "    / o o \\              *1111****   *****
                    \n"
line15:  .asciz  "    \\   > /               **111111***111*
                     \n"
line16: .asciz  "     -----                  ***********
    dce.hust.edu.vn\n"
Message0: .string "------------Menu----------\n"
Message1: .string "1. Print with color\n"
Message2: .string "2. Print without color\n"
Message3: .string "3. Change order\n"
Message4: .string "4. Change color\n"
Message5: .string "5. Exit\n"
Message6: .string "Enter choice: "
Message4_1: .string "Enter color for D(0->9): "
Message4_2: .string "Enter color for C(0->9): "
Message4_3: .string "Enter color for E(0->9): "
```

Các dữ liệu bao gồm:

- line1, line2, ..., line16: Những chuỗi ký tự (string) này chứa các dòng văn bản được in ra màn hình trong phần menu hoặc hiển thị đồ họa.
- Message0, Message1, ..., Message6: Các chuỗi văn bản dùng để hiển thị menu và các hướng dẫn cho người dùng.
- Các chuỗi văn bản này sẽ được sử dụng trong mã trong các lệnh la và ecall để in chúng ra màn hình.

1.2.2 Phần mã chính:

```
    li t0, 50           # t0: curr D color (ASCII '2' = 50)
    li t1, 49           # t1: curr C color (ASCII '1' = 49)
    li t2, 51           # t2: curr E color (ASCII '3' = 51)
menu:
    la a0, Message0
    li a7, 4
    ecall
    li a7, 5
    ecall
    addi t3, a0, 0      # L u  l a  c h n  n g  Í  d ng  v  o  t3
    li t4, 1
    li t5, 2
    li t6, 3
    li s0, 4
    li s1, 5
    beq t3, t4, menu1
    beq t3, t5, menu2
    beq t3, t6, menu3
    beq t3, s0, menu4
    beq t3, s1, end_main
    j menu
```

- li là lệnh "load immediate", dùng để gán giá trị vào thanh ghi.
- Đoạn mã này khởi tạo màu sắc của các ký tự D, C, và E với các giá trị ASCII tương ứng với các số '2', '1', và '3'
- Xử lý các phím số từ 0 đến 9.
- Cập nhật toán hạng và chế độ hiện tại sang Case_1 (chế độ xử lý toán hạng)
- la a0, Message0: Đưa địa chỉ của chuỗi Message0 vào thanh ghi a0.

- li a7, 4: Đặt lệnh hệ thống (system call) là 4 (in chuỗi ra màn hình).
- ecall: Thực thi lệnh hệ thống, in chuỗi từ a0 ra màn hình.
- li a7, 5: Đặt lệnh hệ thống là 5 (đọc dữ liệu từ bàn phím).
- ecall: Đọc giá trị nhập vào từ người dùng và lưu vào thanh ghi a0.
- addi t3, a0, 0: Sao chép giá trị từ a0 vào t3 để lưu trữ lựa chọn của người dùng.
- Đoạn mã này kiểm tra giá trị của lựa chọn người dùng (t3), và dựa trên giá trị đó, chương trình sẽ nhảy đến phần tương ứng (menu1, menu2, menu3, menu4) hoặc kết thúc chương trình (endmain).

1.2.3 Menu 1: In Đầy Đủ

```
menu1:
    # X? l í in d ng c   m u
    li t0, 0
    li t1, 16
    la t2, line1
loop1:
    beq t0, t1, menu
    mv a0, t2
    li a7, 4
    ecall
    addi t2, t2, 60
    addi t0, t0, 1
    j loop1
```

1. **Khởi tạo giá trị:**
   - `t0` (biến đếm số dòng đã in) được khởi tạo bằng 0 và `t1` (số dòng tối đa) được khởi tạo bằng 16.
   - Biến `t2` sẽ trỏ tới địa chỉ bắt đầu của dòng đầu tiên trong bảng ký tự `line1`.
2. **Duyệt qua các dòng:**
   - Chương trình sẽ duyệt qua từng dòng từ `t0` đến `t1` (tối đa 16 dòng).
   - Ở mỗi vòng lặp, `t2` sẽ trỏ đến vị trí bắt đầu của dòng hiện tại trong `line1`.
3. **In dòng ký tự:**
   - Để in dòng ký tự ra màn hình, chương trình sử dụng gọi hệ thống (system call) với mã `a7 = 4` (in chuỗi) và truyền địa chỉ dòng ký tự (trong `t2`) cho tham số `a0`.
   - Hệ thống sẽ in ra toàn bộ nội dung dòng ký tự tại địa chỉ mà `t2` trỏ đến.
4. **Tiến đến dòng tiếp theo:**
   - Sau khi in xong một dòng, chương trình sẽ chuyển con trỏ `t2` đến dòng tiếp theo bằng cách tăng `t2` thêm 60 (vì mỗi dòng có 60 ký tự).
   - Đồng thời, giá trị của `t0` (biến đếm số dòng đã in) sẽ tăng lên 1.
5. **Quay lại vòng lặp:**
   - Quá trình này sẽ lặp lại cho đến khi `t0` đạt giá trị `t1` (16 dòng).
6. **Quay lại menu chính:**
   - Khi tất cả các dòng đã được in, chương trình sẽ quay lại menu chính.

1.2.4 Menu 2: In Không Màu

```
menu2:
    li t0, 0 #i=0
    li t1, 16 #max  =16

    la t2,line1 #t2: pointer to character, starting at first character of
        line1
outer_loop2:
    beq t0, t1, menu #i=16 -> main
    li t3, 0 #j=0
    li t4, 60 #max=60
inner_loop2:
    beq t3, t4, continue_outer_loop2 #j=60 -> continue_outer_loop2
    lb t5, 0(t2) #t5: cur char
    li t6, 48
    li s0, 57
    blt t5, t6, print_char2
    bgt t5, s0, print_char2
    li t5, ' ' #if char is digit, replace it with blank space
print_char2:
    li a7, 11
    mv a0, t5
    ecall
continue_inner_loop2:
    addi t2, t2, 1 #move to next char
    addi t3, t3, 1 #j=j+1
    j inner_loop2
continue_outer_loop2:
    addi t0, t0, 1  #i=i+1
    j outer_loop2
```

1. **Khởi tạo giá trị:**
   - `t0` (biến đếm số dòng đã xử lý) được khởi tạo bằng 0 và `t1` (số dòng tối đa) được khởi tạo bằng 16.
   - Biến `t2` sẽ trỏ tới địa chỉ bắt đầu của dòng đầu tiên trong bảng ký tự (`line1`).
2. **Duyệt qua các dòng:**
   - Chương trình sẽ duyệt qua từng dòng từ `t0` đến `t1` (tối đa 16 dòng).
   - Ở mỗi dòng, chương trình sẽ duyệt qua từng ký tự và kiểm tra xem ký tự đó có phải là một chữ số hay không.
3. **Thay thế chữ số bằng khoảng trắng:**
   - Nếu ký tự hiện tại là một chữ số (giá trị ASCII từ '0' đến '9'), chương trình sẽ thay thế ký tự đó bằng ký tự khoảng trắng (ASCII 32).
   - Nếu không phải là chữ số, chương trình sẽ giữ nguyên ký tự đó.
4. **In dòng:**
   - Sau khi thay thế các ký tự cần thiết, chương trình sẽ in dòng ký tự ra màn hình.
   - Hệ thống gọi (system calls) sẽ được sử dụng để in các ký tự từ địa chỉ mà `t2` đang trỏ tới.
   - Mỗi ký tự được in ra lần lượt, bao gồm việc in ra ký tự thay thế và ký tự tiếp theo cho đến khi hết các ký tự trong dòng.
5. **Tiến đến dòng tiếp theo:**

- Sau khi in hết các ký tự của một dòng, chương trình sẽ tiến đến dòng tiếp theo bằng cách tăng giá trị của `t0` lên 1 và dịch chuyển con trỏ `t2` tới dòng kế tiếp (mỗi dòng có 60 ký tự).

6. **Quay lại vòng lặp:**
   - Quy trình này sẽ được lặp lại cho đến khi `t0` đạt giá trị `t1` (16 dòng).
7. **Quay lại menu chính:**
   - Khi tất cả các dòng đã được xử lý, chương trình sẽ quay lại menu chính.

1.2.5 Menu 3: In ECD

```
menu3:
    # Thay ??i th? t? in
    li t0, 0
    li t1, 16
    la t2, line1
loop3:
    beq t0, t1, menu
    sb zero, 22(t2)
    sb zero, 42(t2)
    sb zero, 58(t2)

    li a7, 4
    addi a0, t2, 43
    ecall

    li a7, 11
    li a0, 32           # In kho?ng tr?ng
    ecall

    li a7, 4
    addi a0, t2, 23
    ecall

    li a7, 11
    li a0, 32
    ecall

    li a7, 4
    mv a0, t2
    ecall

    li a7, 11
    li a0, 10           # In xu?ng d ng
    ecall

    # Ph?c h?i
    li t3, 32
    sb t3, 22(t2)
    sb t3, 42(t2)
    li t3, 10
    sb t3, 58(t2)

    addi t0, t0, 1
    addi t2, t2, 60
    j loop3
```

1. **Khởi tạo giá trị:**
   - Đầu tiên, giá trị t0 (biến đếm số dòng đã xử lý) được khởi tạo bằng 0 và t1 (số dòng tối đa) được khởi tạo bằng 16.

- Biến t2 sẽ trỏ tới địa chỉ bắt đầu của dòng đầu tiên trong bảng ký tự (`line1`).

2. **Duyệt qua các dòng:**
   - Chương trình sẽ duyệt qua từng dòng từ t0 đến t1 (tối đa 16 dòng).
   - Ở mỗi dòng, các ký tự tại các vị trí 22, 42 và 58 sẽ được thay bằng ký tự trắng (ASCII 32).
3. **In dòng:**
   - Sau khi thay đổi ký tự tại các vị trí trên dòng, chương trình sẽ in ra màn hình bằng cách sử dụng các lệnh in (system calls).
   - Các hệ thống gọi (system calls) được thực hiện để in ký tự trên dòng, bao gồm in ra các ký tự từ địa chỉ `t2 + 43`, `t2 + 23`, và sau đó in ký tự `t2`.
   - Dòng được in với khoảng cách (space) ở giữa và kết thúc bằng ký tự xuống dòng (line feed).
4. **Phục hồi các giá trị thay đổi:**
   - Sau khi in dòng, các ký tự tại vị trí 22, 42 và 58 sẽ được phục hồi về trạng thái ban đầu.
   - Dòng được phục hồi bằng cách thay thế giá trị tại các vị trí này bằng ký tự ASCII phù hợp.
5. **Tiến đến dòng tiếp theo:**
   - Sau khi hoàn thành việc xử lý một dòng, chương trình sẽ tiếp tục xử lý dòng tiếp theo bằng cách tăng giá trị t0 lên 1 và di chuyển con trỏ t2 tới dòng tiếp theo (mỗi dòng có 60 ký tự).
6. **Quay lại vòng lặp:**
   - Quy trình này sẽ được lặp lại cho đến khi t0 đạt giá trị t1 (16 dòng).
7. **Quay lại menu chính:**
   - Khi tất cả các dòng đã được xử lý, chương trình sẽ quay lại menu chính.

1.2.6 Menu 4: Nhập màu và thay đổi màu của DCE

```
    menu4:
  # Kh?i t?o  m  u  ban  ??u
    li s3, 50           # t0: curr D color (ASCII '2' = 50)
    li s4, 49           # t1: curr C color (ASCII '1' = 49)
    li s5, 51           # t2: curr E color (ASCII '3' = 51)

    # X? l í thay ??i  m  u
enter_D:
    li a7, 4                # Print "Enter color for D"
    la a0, Message4_1
    ecall

    li a7, 5                # Read user input
    ecall
    mv s8, a0               # L?u gi  tr? nh?p v o t6
    li s0, 48               # ASCII '0' = 48
    li s1, 57               # ASCII '9' = 57
    blt s8, s0, enter_D     # N?u t6 < '0', nh?p l?i
    bgt s8, s1, enter_D     # N?u t6 > '9', nh?p l?i
enter_C:
```

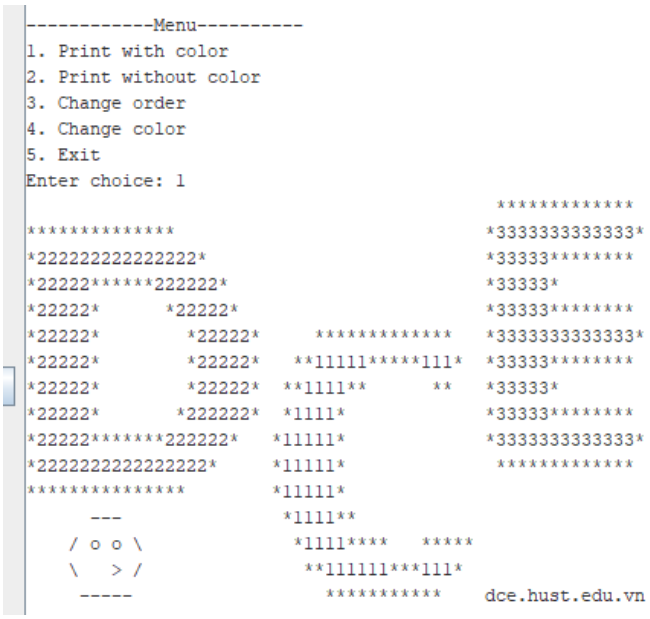
```
    li a7, 4                # Print "Enter color for C"
    la a0, Message4_2
    ecall

    li a7, 5                # Read user input
    ecall
    mv s9, a0               # L?u gi  tr? nh?p v o t6
    blt s9, s0, enter_C     # N?u t6 < '0', nh?p l?i
    bgt s9, s1, enter_C     # N?u t6 > '9', nh?p l?i
enter_E:
    li a7, 4                # Print "Enter color for E"
    la a0, Message4_3
    ecall

    li a7, 5                # Read user input
    ecall
    mv s10, a0               # L?u gi  tr? nh?p v o t6
    blt s10, s0, enter_E     # N?u t6 < '0', nh?p l?i
    bgt s10, s1, enter_E     # N?u t6 > '9', nh?p l?i
init_menu4:
    li t0, 0                # i = 0
    li t1, 16               # max = 16

    la s2, line1            # s2: con tr? ??n k í t? ??u ti n c?a line1
outer_loop4:
    beq t0, t1, update_color # N?u i == 16, tho t v ng l?p
    li t3, 0                # j = 0
    li t4, 60               # max = 60 (s? k í t? tr n m?i d ng)

inner_loop4:
    beq t3, t4, continue_outer_loop4 # N?u j == 60, tho t v ng l?p trong
    lb t5, 0(s2)           # t5: k í t? hi?n t?i

    li s6, 22
    li s7, 42
    blt t3, s6, check_D     # N?u j < 22, thu?c D
    blt t3, s7, check_C     # N?u j < 42, thu?c C
    j check_E               # Ng??c l?i, thu?c E

check_D:
    beq t5, s3, update_D    # N?u k í t? tr ng m u D hi?n t?i, c?p nh?t
    j print_char4

check_C:
    beq t5, s4, update_C    # N?u k í t? tr ng m u C hi?n t?i, c?p nh?t
    j print_char4

check_E:
    beq t5, s5, update_E    # N?u k í t? tr ng m u E hi?n t?i, c?p nh?t
    j print_char4

update_D:
    sb s8, 0(s2)           # L?u m u m?i c?a D v o b? nh?
    mv t5, s8              # C?p nh?t gi  tr? k í t? t5
    j print_char4

update_C:
    sb s9, 0(s2)           # L?u m u m?i c?a C v o b? nh?
    mv t5, s9              # C?p nh?t gi  tr? k í t? t5
    j print_char4

update_E:
    sb s10, 0(s2)           # L?u m u m?i c?a E v o b? nh?
    mv t5, s10             # C?p nh?t gi  tr? k í t? t5
    j print_char4
```

```
86
87  print_char4:
88      li a7, 11               # ecall ?? in k í t?
89      mv a0, t5
90      ecall
91
92  continue_inner_loop4:
93      addi s2, s2, 1          # Di chuy?n ??n k í t? ti?p theo
94      addi t3, t3, 1          # j = j + 1
95      j inner_loop4
96
97  continue_outer_loop4:
98      addi t0, t0, 1          # i = i + 1
99      j outer_loop4
100
101 update_color:
102     mv t0, s8               # C?p nh?t m u D hi?n t?i
103     mv t1, s9               # C?p nh?t m u C hi?n t?i
104     mv t2, s10               # C?p nh?t m u E hi?n t?i
105     j menu
```

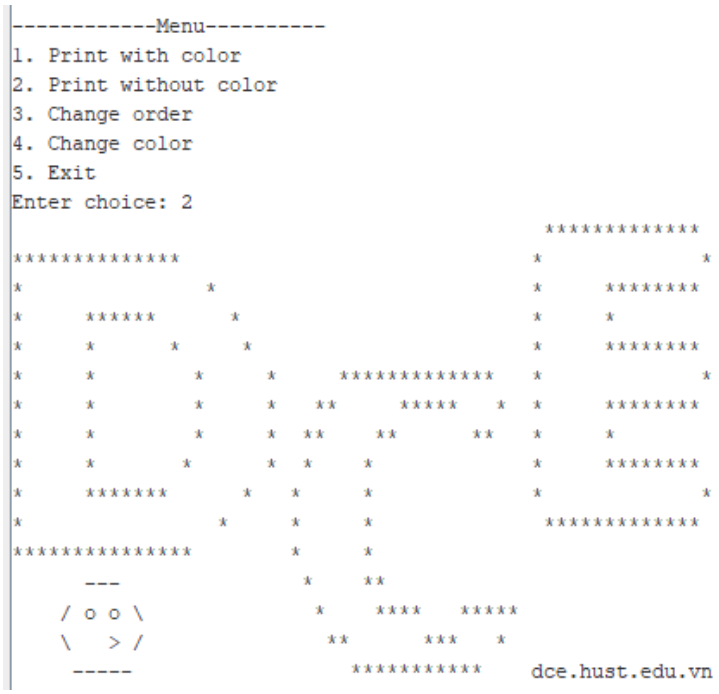
1. **Khởi tạo màu ban đầu:**
   - Mỗi thành phần D, C, E được khởi tạo với một màu mặc định dưới dạng ký tự ASCII:
     - D = '2' (mã ASCII = 50)
     - C = '1' (mã ASCII = 49)
     - E = '3' (mã ASCII = 51)
2. **Nhập màu mới cho D, C, E:**
   - Chương trình yêu cầu người dùng nhập một màu mới cho từng thành phần (D, C, E). Màu nhập vào phải là một số trong phạm vi từ '0' đến '9'.
   - Nếu giá trị người dùng nhập không hợp lệ, chương trình sẽ yêu cầu nhập lại cho đến khi nhận được giá trị hợp lệ.
3. **Duyệt qua bảng ký tự:**
   - Chương trình sẽ lặp qua từng dòng (16 dòng) và mỗi dòng có tối đa 60 ký tự.
   - Mỗi ký tự sẽ được kiểm tra xem có thuộc về thành phần nào (D, C, E) hay không dựa trên vị trí của nó trong dòng:
     - Các ký tự từ vị trí 0 đến 21 thuộc về D.
     - Các ký tự từ vị trí 22 đến 41 thuộc về C.
     - Các ký tự từ vị trí 42 đến 59 thuộc về E.
4. **Cập nhật màu cho D, C, E:**
   - Nếu ký tự trong bảng trùng với màu của D, C, hoặc E, chương trình sẽ thay thế ký tự đó với màu mới mà người dùng đã nhập.
   - Sau khi cập nhật màu, ký tự sẽ được in lại trên màn hình.
5. **Lặp lại quá trình:**
   - Sau khi tất cả các ký tự trong bảng được duyệt và cập nhật, chương trình sẽ quay lại menu chính để người dùng có thể thực hiện các thao tác tiếp theo.

1.3 Kết quả

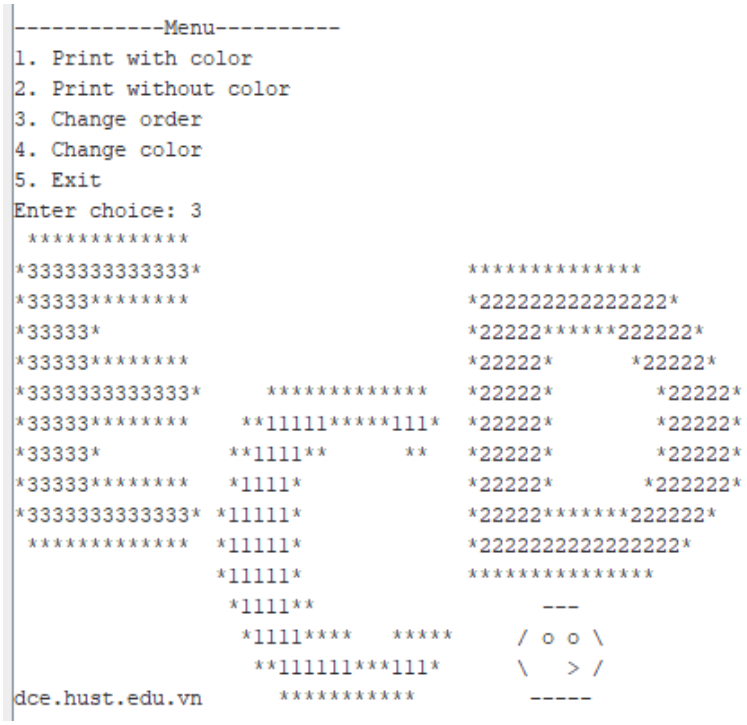
1.3.1 Khi chạy menu 1



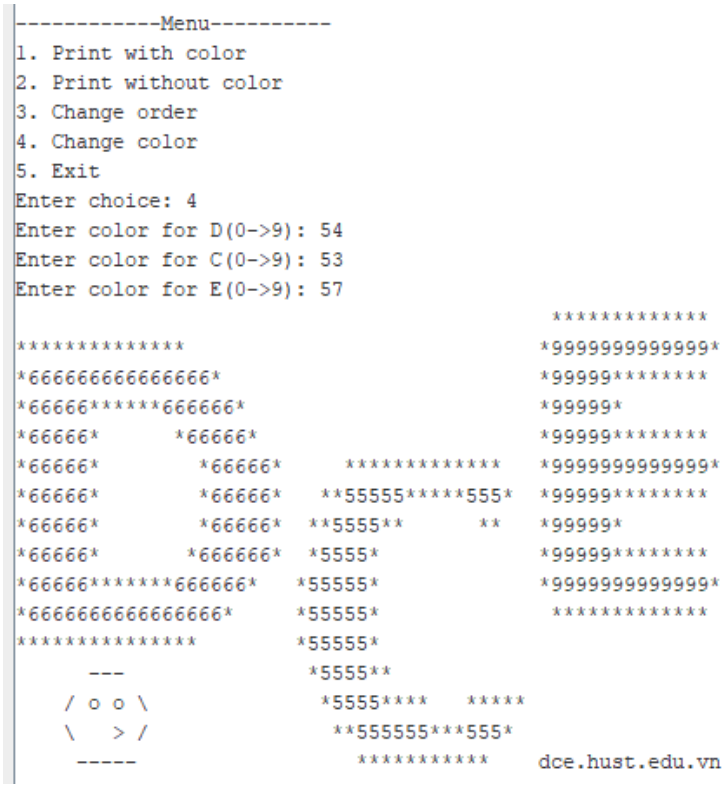
1.3.2 Khi chạy menu 2



1.3.3 Khi chạy menu 3



1.3.4 Khi chạy menu 4



→ Chương trình chạy đúng

```
1 .data
2 line1:  .asciz  "                                                  **********
    \n"
3 line2:  .asciz  "**************
    *3333333333333*\n"
```

```
4  line3:  .asciz  "*222222222222222*                          *33333********
       \n"
5  line4:  .asciz  "*22222******222222*                        *33333*
       \n"
6  line5:  .asciz  "*22222*      *22222*                       *33333********
       \n"
7  line6:  .asciz  "*22222*        *22222*     *************
       *3333333333333*\n"
8  line7:  .asciz  "*22222*        *22222*   **11111*****111*  *33333********
       \n"
9  line8:  .asciz  "*22222*        *22222*  **1111**      **   *33333*
       \n"
10 line9:  .asciz  "*22222*       *222222*  *1111*            *33333********
       \n"
11 line10: .asciz  "*22222******222222*   *11111*
       *3333333333333*\n"
12 line11: .asciz  "*2222222222222222*    *11111*             *************
       \n"
13 line12: .asciz  "***************       *11111*
       \n"
14 line13: .asciz  "      ---               *1111**
       \n"
15 line14: .asciz  "    / o o \\              *1111****   *****
                     \n"
16 line15: .asciz  "    \\   > /               **111111***111*
                      \n"
17 line16: .asciz  "     -----                  **********
       dce.hust.edu.vn\n"
18
19 Message0: .string "------------Menu----------\n"
20 Message1: .string "1. Print with color\n"
21 Message2: .string "2. Print without color\n"
22 Message3: .string "3. Change order\n"
23 Message4: .string "4. Change color\n"
24 Message5: .string "5. Exit\n"
25 Message6: .string "Enter choice: "
26 Message4_1: .string "Enter color for D(0->9): "
27 Message4_2: .string "Enter color for C(0->9): "
28 Message4_3: .string "Enter color for E(0->9): "
29
30 .text
31 .globl main
32 main:
33     # Kh?i t?o m u ban ??u
34     li t0, 50           # t0: curr D color (ASCII '2' = 50)
35     li t1, 49           # t1: curr C color (ASCII '1' = 49)
36     li t2, 51           # t2: curr E color (ASCII '3' = 51)
37
38 menu:
39     # Hi?n th? menu
40     la a0, Message0
41     li a7, 4
42     ecall
43
44     la a0, Message1
45     li a7, 4
46     ecall
47
48     la a0, Message2
49     li a7, 4
50     ecall
51
52     la a0, Message3
53     li a7, 4
54     ecall
55
```

```
    la a0, Message4
    li a7, 4
    ecall

    la a0, Message5
    li a7, 4
    ecall

    la a0, Message6
    li a7, 4
    ecall

    li a7, 5
    ecall
    addi t3, a0, 0      # L?u l?a ch?n v o t3

    # ?i?u h??ng menu
    li t4, 1
    li t5, 2
    li t6, 3
    li s0, 4
    li s1, 5
    beq t3, t4, menu1
    beq t3, t5, menu2
    beq t3, t6, menu3
    beq t3, s0, menu4
    beq t3, s1, end_main
    j menu

menu1:
    # X? l í in d ng c   m u
    li t0, 0
    li t1, 16
    la t2, line1
loop1:
    beq t0, t1, menu
    mv a0, t2
    li a7, 4
    ecall
    addi t2, t2, 60
    addi t0, t0, 1
    j loop1

menu2:
    li t0, 0 #i=0
    li t1, 16 #max  =16

    la t2,line1 #t2: pointer to character, starting at first character of
        line1
outer_loop2:
    beq t0, t1, menu #i=16 -> main
    li t3, 0 #j=0
    li t4, 60 #max=60
inner_loop2:
    beq t3, t4, continue_outer_loop2 #j=60 -> continue_outer_loop2
    lb t5, 0(t2) #t5: cur char
    li t6, 48
    li s0, 57
    blt t5, t6, print_char2
    bgt t5, s0, print_char2
    li t5, ' ' #if char is digit, replace it with blank space
print_char2:
    li a7, 11
    mv a0, t5
    ecall
continue_inner_loop2:
```

```
121     addi t2, t2, 1 #move to next char
122     addi t3, t3, 1 #j=j+1
123     j inner_loop2
124 continue_outer_loop2:
125     addi t0, t0, 1  #i=i+1
126     j outer_loop2
127 
128 menu3:
129     # Thay ??i th? t? in
130     li t0, 0
131     li t1, 16
132     la t2, line1
133 loop3:
134     beq t0, t1, menu
135     sb zero, 22(t2)
136     sb zero, 42(t2)
137     sb zero, 58(t2)
138 
139     li a7, 4
140     addi a0, t2, 43
141     ecall
142 
143     li a7, 11
144     li a0, 32          # In kho?ng tr?ng
145     ecall
146 
147     li a7, 4
148     addi a0, t2, 23
149     ecall
150 
151     li a7, 11
152     li a0, 32
153     ecall
154 
155     li a7, 4
156     mv a0, t2
157     ecall
158 
159     li a7, 11
160     li a0, 10          # In xu?ng d ng
161     ecall
162 
163     # Ph?c h?i
164     li t3, 32
165     sb t3, 22(t2)
166     sb t3, 42(t2)
167     li t3, 10
168     sb t3, 58(t2)
169 
170     addi t0, t0, 1
171     addi t2, t2, 60
172     j loop3
173 menu4:
174   # Kh?i t?o m u ban ??u
175     li s3, 50          # t0: curr D color (ASCII '2' = 50)
176     li s4, 49          # t1: curr C color (ASCII '1' = 49)
177     li s5, 51          # t2: curr E color (ASCII '3' = 51)
178 
179     # X? l í thay ??i m u
180 enter_D:
181     li a7, 4                # Print "Enter color for D"
182     la a0, Message4_1
183     ecall
184 
185     li a7, 5                # Read user input
186     ecall
```

```
187     mv s8, a0               # L?u gi  tr? nh?p v o t6
188     li s0, 48               # ASCII '0' = 48
189     li s1, 57               # ASCII '9' = 57
190     blt s8, s0, enter_D     # N?u t6 < '0', nh?p l?i
191     bgt s8, s1, enter_D     # N?u t6 > '9', nh?p l?i
192 enter_C:
193     li a7, 4                # Print "Enter color for C"
194     la a0, Message4_2
195     ecall
196
197     li a7, 5                # Read user input
198     ecall
199     mv s9, a0               # L?u gi  tr? nh?p v o t6
200     blt s9, s0, enter_C     # N?u t6 < '0', nh?p l?i
201     bgt s9, s1, enter_C     # N?u t6 > '9', nh?p l?i
202 enter_E:
203     li a7, 4                # Print "Enter color for E"
204     la a0, Message4_3
205     ecall
206
207     li a7, 5                # Read user input
208     ecall
209     mv s10, a0               # L?u gi  tr? nh?p v o t6
210     blt s10, s0, enter_E     # N?u t6 < '0', nh?p l?i
211     bgt s10, s1, enter_E     # N?u t6 > '9', nh?p l?i
212 init_menu4:
213     li t0, 0                # i = 0
214     li t1, 16               # max = 16
215
216     la s2, line1            # s2: con tr? ??n k í t? ??u ti n c?a line1
217 outer_loop4:
218     beq t0, t1, update_color # N?u i == 16, tho t v ng l?p
219     li t3, 0                # j = 0
220     li t4, 60               # max = 60 (s? k í t? tr n m?i d ng)
221
222 inner_loop4:
223     beq t3, t4, continue_outer_loop4 # N?u j == 60, tho t v ng l?p trong
224     lb t5, 0(s2)           # t5: k í t? hi?n t?i
225
226     li s6, 22
227     li s7, 42
228     blt t3, s6, check_D    # N?u j < 22, thu?c D
229     blt t3, s7, check_C    # N?u j < 42, thu?c C
230     j check_E              # Ng??c l?i, thu?c E
231
232 check_D:
233     beq t5, s3, update_D   # N?u k í t? tr ng m u D hi?n t?i, c?p nh?t
234     j print_char4
235
236 check_C:
237     beq t5, s4, update_C   # N?u k í t? tr ng m u C hi?n t?i, c?p nh?t
238     j print_char4
239
240 check_E:
241     beq t5, s5, update_E   # N?u k í t? tr ng m u E hi?n t?i, c?p nh?t
242     j print_char4
243
244 update_D:
245     sb s8, 0(s2)           # L?u m u m?i c?a D v o b? nh?
246     mv t5, s8              # C?p nh?t gi  tr? k í t? t5
247     j print_char4
248
249 update_C:
250     sb s9, 0(s2)           # L?u m u m?i c?a C v o b? nh?
251     mv t5, s9              # C?p nh?t gi  tr? k í t? t5
```

```
    j print_char4

update_E:
    sb s10, 0(s2)            # L?u m u m?i c?a E v o b? nh?
    mv t5, s10               # C?p nh?t gi  tr? k í t? t5
    j print_char4

print_char4:
    li a7, 11               # ecall ?? in k í t?
    mv a0, t5
    ecall

continue_inner_loop4:
    addi s2, s2, 1          # Di chuy?n ??n k í t? ti?p theo
    addi t3, t3, 1          # j = j + 1
    j inner_loop4

continue_outer_loop4:
    addi t0, t0, 1          # i = i + 1
    j outer_loop4

update_color:
    mv t0, s8               # C?p nh?t m u D hi?n t?i
    mv t1, s9               # C?p nh?t m u C hi?n t?i
    mv t2, s10               # C?p nh?t m u E hi?n t?i
    j menu



end_main:
    li a7, 10
    ecall
```

2 Testing sorting algorithms

2.1 Đề bài

9. Testing sorting algorithms

+ Research about the system calls to open and read data from a text file.
+ Given text files contain random numbers separated by spaces. The maximum number of elements is 10000.
+ The program allows the user to enter a file name to open, the numbers in the file are read and saved to memory.
+ The user selects the sorting algorithm to be performed (Bubble, Insertion, Selection). Bonus points are rewarded if other algorithms are performed.
+ The program runs the algorithm and prints the execution time (Using TimerTool).
+ The program writes the sorting results to the result file.

2.2 Thuật toán và hàm

2.2.1 Khai báo dữ liệu

```
    .data
    numbers:     .space  80000
    input_buffer_size:    .word   80000
    count:       .word   0

    neg_bitmask:    .space  2500
```

```
7
8       input_filename:    .space  256
9       file_read_buffer:     .space  1024
10      msg_prompt_input:      .string "Enter filename: "
11      error_msg:  .string "\nError opening file\n"
12      menu:       .string "\nUser select sorting algorithm:\n1. Bubble
            Sort\n2. Insertion Sort\n3. Selection Sort\n4. Quick
            Sort\n5.Close\nChoice: "
13
14      fd:         .word   0
15      newline:    .string "\n"
16      space:      .string " "
17      start_time: .word   0
18      end_time:   .word   0
19
20      msg_execution_time:    .string "\nExecution time (ms): "
21
22      output_filename:     .string "D:/HUST/Computer Architecture
            Lab/output1.txt"
23      out_fd:     .word   0
24      buffer_number:      .space  12
25      msg_file_error_open:  .string "\nError writing to output file\n"
26      char_minus:       .string "-"
27      # Them buffer tam cho sap xep tron
28      tmp_sort_buffer:    .space  80000
```

Các dữ liệu bao gồm:

- Các chuỗi như prompt$_in, prompt_out, prompt_algo, error_msg, vinvalid_algochaccthngboyucungidngnhpv$

- algo$_choicelutrlachnthuttoncangidng$.

2.2.2 Đọc tên file đầu vào và đầu ra:

```
1   # In ra msg_prompt_input
2       li a7, 4
3       la a0, msg_prompt_input
4       ecall
5       # Doc input_filename
6       li a7, 8
7       la a0, input_filename
8       li a1, 256
9       ecall
10      # Loai bo newline khoi input_filename
11      la t0, input_filename
```

- li a7, 4 thực hiện syscall để in thông báo yêu cầu người dùng nhập tên file đầu vào.

- li a7, 8 thực hiện syscall để đọc tên file vào thanh ghi a0 và lưu vào bộ nhớ input$_file$.

2.2.3 Lựa chọn thuật toán sắp xếp:

```
1   menu_loop:
2       # Hien thi menu
3       li a7, 4
4       la a0, menu
5       ecall
6
7       # Doc lua chon
8       li a7, 5
9       ecall
10
11      # Nhanh den lua chon
12      li t0, 1
```

```
    beq a0, t0, bubble_sort_array
    li t0, 2
    beq a0, t0, insertion_sort_array
    li t0, 3
    beq a0, t0, selection_sort_array
    li t0, 4
    beq a0, t0, quick_sort_array
    li t0,5
    beq a0, t0, exit
    j exit
```

- li a7, 5 thực hiện syscall để đọc lựa chọn thuật toán từ người dùng (dạng số nguyên) và lưu vào bộ nhớ $algo_choice$.

2.2.4 Kiểm tra lựa chọn thuật toán hợp lệ:

```
    read_loop:
    # Doc mot ky tu tu file
    li a7, 63           # Syscall ReadFile
    lw a0, fd           # Mo ta file
    la a1, file_read_buffer     # Dia chi buffer
    li a2, 1            # Doc mot ky tu
    ecall

    # Kiem tra neu cuoi file
    beqz a0, read_done

    # Tai ky tu
    lb t2, 0(a1)

    # Kiem tra dau tru
    li t3, 45           # ASCII cho '-'
    bne t2, t3, not_char_minus
    beqz t1, set_negative   # Chi dat am neu o dau so
    j read_loop
```

- Nếu người dùng chọn thuật toán không hợp lệ (không phải 1 hoặc 2), chương trình sẽ thông báo lỗi và thoát.

2.2.5 Mở và đọc file:

```
open_input_file:
    li a7, 1024
    la a0, input_filename
    li a1, 0
    ecall
    bltz a0, file_error_open
    la t1, fd
    sw a0, 0(t1)
    jal read_numbers
```

- li a7, 1024 thực hiện syscall mở file, với mã lệnh 0 cho chế độ đọc (a1 = 0) và 1 cho chế độ ghi (a1 = 1).

2.2.6 Đọc dữ liệu từ file và chuyển đổi:

```
 read_numbers:
    addi sp, sp, -16
    sw ra, 12(sp)
    sw s0, 8(sp)
    sw s1, 4(sp)
```

```
    sw s2, 0(sp)

    # Reset count
    la t1, count        # Tai dia chi cua count
    sw zero, 0(t1)      # Luu gia tri zero vao count

    # Khoi tao bien
    li t0, 0            # So hien tai dang duoc xay dung
    li t1, 0            # Co cho biet dang trong so
    li t6, 0            # Co danh dau (0 = duong, 1 = am)
```

- Sau khi đọc dữ liệu từ file vào bộ đệm (buffer), chương trình chuyển đổi dữ liệu từ định dạng ASCII sang số nguyên và lưu vào mảng array.

2.2.7 Sắp xếp mảng theo thuật toán đã chọn:

```
    # Choose sorting algorithm based on user input
la t0, algo_choice
lw t1, 0(t0)
li t2, 1
beq t1, t2, do_bubble_sort

do_insertion_sort:
    # Call insertion sort
    la a0, array
    la t3, array_length
    lw a1, 0(t3)
    jal insertion_sort
```

- Nếu người dùng chọn Bubble Sort (tương ứng với giá trị 1), hàm bubble$_s ortsØcgi.Nungidngchnthuttonkhc(n$

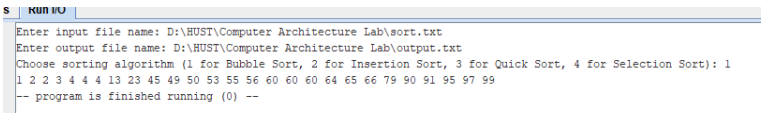
2.2.8 Đóng file và thoát chương trình:

```
   close_file:
    # Dong file
    li a7, 57           # Syscall Close file
    lw a0, fd
    ecall

    lw ra, 12(sp)
    lw s0, 8(sp)
    lw s1, 4(sp)
    lw s2, 0(sp)
    addi sp, sp, 16
    ret
```

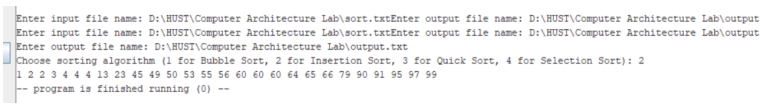
- li a7, 57 thực hiện syscall để đóng các file.

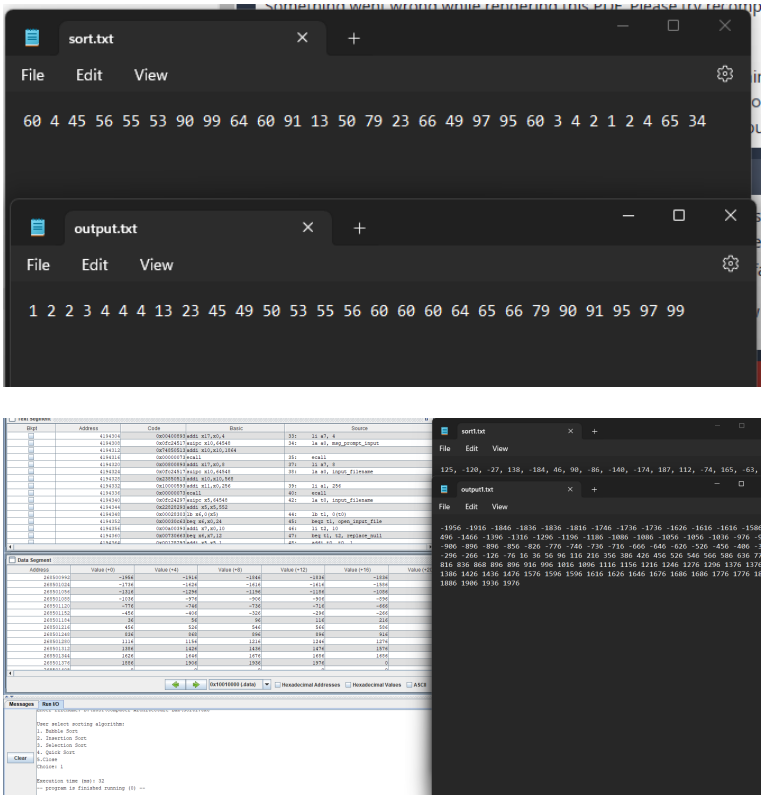
2.3 Kết quả

2.3.1 Khi chạy Bubble Sort



2.3.2 Khi chạy Insertion Sort





Hình 1: Output khi Chay so Am

2.3.3 Khi chạy Selection Sort

```
Enter input file name: D:\HUST\Computer Architecture Lab\sort.txtEnter output file name: D:\HUST\Computer Architecture Lab\output
Enter input file name: D:\HUST\Computer Architecture Lab\sort.txtEnter output file name: D:\HUST\Computer Architecture Lab\output
Enter output file name: D:\HUST\Computer Architecture Lab\output.txt
Choose sorting algorithm (1 for Bubble Sort, 2 for Insertion Sort, 3 for Quick Sort, 4 for Selection Sort): 2
1 2 2 3 4 4 4 13 23 45 49 50 53 55 56 60 60 60 64 65 66 79 90 91 95 97 99
-- program is finished running (0) --
```

→ Chương trình chạy đúng

```
  .data
    numbers:     .space  80000
    input_buffer_size:    .word    80000
    count:       .word   0
    neg_bitmask:    .space  2500
    input_filename:    .space  256
    file_read_buffer:     .space  1024
    msg_prompt_input:      .string "Enter filename: "
    error_msg:   .string "\nError opening file\n"
    menu:        .string "\nUser select sorting algorithm:\n1. Bubble
        Sort\n2. Insertion Sort\n3. Selection Sort\n4. Quick
        Sort\n5.Close\nChoice: "
    fd:          .word   0
    newline:     .string "\n"
    space:       .string " "
    start_time: .word   0
    end_time:    .word   0
    msg_execution_time:    .string "\nExecution time (ms): "
    output_filename:     .string "D:/HUST/Computer Architecture
        Lab/output1.txt"
    out_fd:      .word   0
    buffer_number:      .space  12
    msg_file_error_open:   .string "\nError writing to output file\n"
    char_minus:       .string "-"
    tmp_sort_buffer:    .space  80000
.text
```

```
.globl main
main:
    # print msg_prompt_input
    li a7, 4
    la a0, msg_prompt_input
    ecall
    # read input_filename
    li a7, 8
    la a0, input_filename
    li a1, 256
    ecall
    # delete newline from input_filename
    la t0, input_filename
remove_newline_from_filename:
    lb t1, 0(t0)
    beqz t1, open_input_file
    li t2, 10
    beq t1, t2, replace_null
    addi t0, t0, 1
    j remove_newline_from_filename
replace_null:
    sb zero, 0(t0)
open_input_file:
    li a7, 1024
    la a0, input_filename
    li a1, 0
    ecall
    bltz a0, file_error_open
    la t1, fd
    sw a0, 0(t1)
    jal read_numbers
menu_loop:
    li a7, 4
    la a0, menu
    ecall
    li a7, 5
    ecall
    li t0, 1
    beq a0, t0, bubble_sort_array
    li t0, 2
    beq a0, t0, insertion_sort_array
    li t0, 3
    beq a0, t0, selection_sort_array
    li t0, 4
    beq a0, t0, quick_sort_array
    li t0,5
    beq a0, t0, exit
    j exit
file_error_open:
    li a7, 4
    la a0, error_msg
    ecall
    j exit
read_numbers:
    addi sp, sp, -16
    sw ra, 12(sp)
    sw s0, 8(sp)
    sw s1, 4(sp)
    sw s2, 0(sp)
    # Reset count
    la t1, count
    sw zero, 0(t1)
    li t0, 0            # So hien tai dang duoc xay dung
    li t1, 0            # Co cho biet dang trong so
    li t6, 0            # Co danh dau (0 = duong, 1 = am)
read_loop:
```

```
    # Doc mot ky tu tu file
    li a7, 63           # Syscall ReadFile
    lw a0, fd           # Mo ta file
    la a1, file_read_buffer     # Dia chi buffer
    li a2, 1            # Doc mot ky tu
    ecall
    # Kiem tra neu cuoi file
    beqz a0, read_done
    # Tai ky tu
   lb t2, 0(a1)
    # Kiem tra dau tru
    li t3, 45           # ASCII cho '-'
    bne t2, t3, not_char_minus
    beqz t1, set_negative   # Chi dat am neu o dau so
    j read_loop
set_negative:
    li t6, 1            # Dat co danh dau la am
    li t1, 1            # Dat co trong so
    j read_loop
not_char_minus:
    # Kiem tra neu la dau cach hoac newline
    li t3, 32           # Space
    beq t2, t3, save_number
    li t3, 10           # Newline
    beq t2, t3, save_number
    # Chuyen doi ASCII thanh so va cong vao so hien tai
    addi t2, t2, -48    # Chuyen doi ASCII thanh so
    li t3, 10
    mul t0, t0, t3      # So hien tai * 10
    add t0, t0, t2      # Cong vao chu so moi
    li t1, 1            # Dat co trong so
    j read_loop
save_number:
    beqz t1, read_loop  # Neu khong trong so, tiep tuc
    # Ap dung dau neu la am
    beqz t6, save_positive
    neg t0, t0          # Doi dau so neu co am duoc dat
save_positive:
    # Luu so vao mang
    la t3, count        # Tai dia chi cua count
    lw t3, 0(t3)        # Tai gia tri count
    slli t4, t3, 2      # t4 = count * 4
    la t5, numbers
    add t5, t5, t4
    sw t0, 0(t5)        # Luu so
    # Tang count
    addi t3, t3, 1
    la t4, count        # Tai dia chi cua count
    sw t3, 0(t4)        # Luu count moi
    # Reset cho so tiep theo
    li t0, 0            # Reset so hien tai
    li t1, 0            # Reset co trong so
    li t6, 0            # Reset co am
    j read_loop
read_done:
    # Neu chung ta dang trong so khi file ket thuc, luu no
    beqz t1, close_file
    # Ap dung dau neu la am
    beqz t6, save_last_positive
    neg t0, t0          # Doi dau so neu co am duoc dat
save_last_positive:
    la t3, count        # Tai dia chi cua count
    lw t3, 0(t3)        # Tai gia tri count
    slli t4, t3, 2
    la t5, numbers
    add t5, t5, t4
```

```
    sw t0, 0(t5)
    addi t3, t3, 1
    la t4, count        # Tai dia chi cua count
    sw t3, 0(t4)        # Luu count moi
close_file:
    # Dong file
    li a7, 57           # Syscall Close file
    lw a0, fd
    ecall
    lw ra, 12(sp)
    lw s0, 8(sp)
    lw s1, 4(sp)
    lw s2, 0(sp)
    addi sp, sp, 16
    ret
flag_negative_numbers:
    # a0 = dia chi mang
    # a1 = size
    addi sp, sp, -16
    sw ra, 12(sp)
    sw s0, 8(sp)
    sw s1, 4(sp)
    sw s2, 0(sp)
    mv s0, a0           # s0 = dia chi mang
    mv s1, a1           # s1 = size
    li s2, 0            # s2 = bo dem
flag_loop:
    bge s2, s1, flag_done
    # Tai so hien tai
    slli t0, s2, 2      # t0 = bo dem * 4
    add t0, s0, t0
    lw t1, 0(t0)       # Tai so
    # Bo qua neu duong
    bgez t1, skip_flag
    # Tinh toan byte va vi tri bit trong bitmask
    mv t0, s2           # Sao chep chi so
    srai t1, t0, 3      # Byte offset = chi so / 8
    andi t2, t0, 0x7    # Vi tri bit = chi so % 8
    li t3, 1
    sll t3, t3, t2      # Dich chuyen 1 den vi tri bit chinh xac
    # Dat bit trong bitmask
    la t4, neg_bitmask
    add t4, t4, t1      # Them byte offset
    lb t5, 0(t4)        # Tai byte hien tai
    or t5, t5, t3       # Dat bit
    sb t5, 0(t4)        # Luu byte da cap nhat
skip_flag:
    addi s2, s2, 1
    j flag_loop
flag_done:
    lw ra, 12(sp)
    lw s0, 8(sp)
    lw s1, 4(sp)
    lw s2, 0(sp)
    addi sp, sp, 16
    ret
quick_sort_array:
    # Lay thoi gian bat dau
    jal get_time
    sw a0, start_time, t0
    # Khoi tao quicksort
    la a0, numbers
    li a1, 0
    lw a2, count
    addi a2, a2, -1
    jal quick_sort_logic
```

```
    # Danh dau cac so am
    la a0, numbers
    lw a1, count
    jal flag_negative_numbers
    # Lay thoi gian ket thuc va tinh thoi gian thuc thi
    jal get_time
    sw a0, end_time, t0
    jal print_time
    # Ghi ket qua vao file
    jal write_results
    j menu_loop
quick_sort_logic:
    # a0 = dia chi mang
    # a1 = chi so ben trai
    # a2 = chi so ben phai
    addi sp, sp, -24
    sw ra, 20(sp)
    sw s0, 16(sp)
    sw s1, 12(sp)
    sw s2, 8(sp)
    sw s3, 4(sp)
    sw s4, 0(sp)
    # Luu cac tham so
    mv s0, a0          # s0 = dia chi mang
    mv s1, a1          # s1 = ben trai
    mv s2, a2          # s2 = ben pha
   # Dieu kien dung: neu ben trai >= ben phai, thoat
    bge s1, s2, quick_sort_end
    # Goi ham partition_elements
    mv a0, s0          # dia chi mang
    mv a1, s1          # chi so ben trai
    mv a2, s2          # chi so ben phai
    jal partition_elements
    mv s3, a0          # s3 = chi so pivot
    # De quy sap xep phan ben trai
    mv a0, s0          # dia chi mang
    mv a1, s1          # chi so ben trai
    addi a2, s3, -1    # pivot - 1
    jal quick_sort_logic
    # De quy sap xep phan ben phai
    mv a0, s0          # dia chi mang
    addi a1, s3, 1     # pivot + 1
    mv a2, s2          # chi so ben phai
    jal quick_sort_logic
quick_sort_end:
    lw ra, 20(sp)
    lw s0, 16(sp)
    lw s1, 12(sp)
    lw s2, 8(sp)
    lw s3, 4(sp)
    lw s4, 0(sp)
    addi sp, sp, 24
    ret
partition_elements:
    addi sp, sp, -24
    sw ra, 20(sp)
    sw s0, 16(sp)
    sw s1, 12(sp)
    sw s2, 8(sp)
    sw s3, 4(sp)
    sw s4, 0(sp)
    mv s0, a0          # s0 = dia chi mang
    mv s1, a1          # s1 = ben trai
    mv s2, a2          # s2 = ben phai
    slli t0, s2, 2     # t0 = ben phai * 4
    add t0, s0, t0
```

```
    lw s3, 0(t0)       # s3 = gia tri pivot
    addi s4, s1, -1    # i = ben trai - 1
    mv t1, s1          # j = ben trai
partition_loop_elements:
    bge t1, s2, partition_elements_done
    # Tai phan tu hien tai
    slli t0, t1, 2
    add t0, s0, t0
    lw t2, 0(t0)       # t2 = arr[j]
    # So sanh voi pivot
    bgt t2, s3, skip_swap
    # Tang i
    addi s4, s4, 1
    # Doi phan tu neu i != j
    slli t0, s4, 2
    add t0, s0, t0     # Dia chi cua arr[i]
    slli t3, t1, 2
    add t3, s0, t3     # Dia chi cua arr[j]
    lw t4, 0(t0)       # t4 = arr[i]
    lw t5, 0(t3)       # t5 = arr[j]
    sw t5, 0(t0)       # arr[i] = arr[j]
    sw t4, 0(t3)       # arr[j] = arr[i]
skip_swap:
    addi t1, t1, 1     # j++
    j partition_loop_elements
partition_elements_done:
    # Dat pivot vao vi tri cuoi cung
    addi s4, s4, 1     # i++
    # Doi pivot voi phan tu tai i
    slli t0, s4, 2
    add t0, s0, t0     # Dia chi cua arr[i]
    slli t1, s2, 2
    add t1, s0, t1     # Dia chi cua arr[right]
    lw t2, 0(t0)       # t2 = arr[i]
    lw t3, 0(t1)       # t3 = arr[right]
    sw t3, 0(t0)       # arr[i] = arr[right]
    sw t2, 0(t1)       # arr[right] = arr[i]
    # Tra ve chi so pivot
    mv a0, s4
    lw ra, 20(sp)
    lw s0, 16(sp)
    lw s1, 12(sp)
    lw s2, 8(sp)
    lw s3, 4(sp)
    lw s4, 0(sp)
    addi sp, sp, 24
    ret
bubble_sort_array:
    # Lay thoi gian bat dau
    jal get_time
    sw a0, start_time, t0
    # Thuc hien bubble sort
    la a0, numbers     # Tai dia chi mang
    lw a1, count       # Tai kich thuoc mang
    jal bubble_sort_core
        # Danh dau cac so am
    la a0, numbers
    lw a1, count
    jal flag_negative_numbers
    # Lay thoi gian ket thuc va tinh thoi gian thuc thi
    jal get_time
    sw a0, end_time, t0
    jal print_time
    # Ghi ket qua vao file
    jal write_results
    j exit
```

```
insertion_sort_array:
    # Lay thoi gian bat dau
    jal get_time
    sw a0, start_time, t0
    # Thuc hien insertion sort
    la a0, numbers      # Tai dia chi mang
    lw a1, count        # Tai kich thuoc mang
    jal insertion_sort_array_impl
        # Danh dau cac so am
    la a0, numbers
    lw a1, count
    jal flag_negative_numbers
    # Lay thoi gian ket thuc va tinh thoi gian thuc thi
    jal get_time
    sw a0, end_time, t0
    jal print_time
    # Ghi ket qua vao file
    jal write_results
    j exit
selection_sort_array:
    # Lay thoi gian bat dau
    jal get_time
    sw a0, start_time, t0
    # Thuc hien selection sort
    la a0, numbers      # Tai dia chi mang
    lw a1, count        # Tai kich thuoc mang
    jal selection_sort_array_impl
    # Danh dau cac so am
    la a0, numbers
    lw a1, count
    jal flag_negative_numbers
    # Lay thoi gian ket thuc va tinh thoi gian thuc thi
    jal get_time
    sw a0, end_time, t0
    jal print_time
    # Ghi ket qua vao file
    jal write_results
    j exit
bubble_sort_core:
    # a0 = dia chi mang
    # a1 = kich thuoc
    addi sp, sp, -16
    sw ra, 12(sp)
    sw s0, 8(sp)
    sw s1, 4(sp)
    sw s2, 0(sp)
    mv s0, a0           # s0 = dia chi mang
    mv s1, a1           # s1 = kich thuoc
    li s2, 0            # s2 = i
outer_loop_bubble_sort:
    bge s2, s1, bubble_done
    li t0, 0            # j = 0
inner_loop_bubble_sort:
    sub t1, s1, s2
    addi t1, t1, -1
    bge t0, t1, inner_done_bubble_sort
    slli t2, t0, 2
    add t2, s0, t2
    lw t3, 0(t2)        # arr[j]
    lw t4, 4(t2)        # arr[j+1]
    ble t3, t4, no_swap_bubble_sort
    sw t4, 0(t2)
    sw t3, 4(t2)
no_swap_bubble_sort:
    addi t0, t0, 1
    j inner_loop_bubble_sort
```

```
inner_done_bubble_sort:
    addi s2, s2, 1
    j outer_loop_bubble_sort
bubble_done:
    lw ra, 12(sp)
    lw s0, 8(sp)
    lw s1, 4(sp)
    lw s2, 0(sp)
    addi sp, sp, 16
    ret
insertion_sort_array_impl:
    addi sp, sp, -16
    sw ra, 12(sp)
    sw s0, 8(sp)
    sw s1, 4(sp)
    sw s2, 0(sp)
    mv s0, a0           # s0 = dia chi mang
    mv s1, a1           # s1 = kich thuoc
    li s2, 1            # s2 = i = 1
outer_loop_insertion:
    bge s2, s1, insertion_done

    # Lay phan tu hien tai
    slli t0, s2, 2      # t0 = i * 4
    add t0, s0, t0
    lw t1, 0(t0)        # key = arr[i]
    addi t2, s2, -1     # j = i-1

inner_loop_insertion:
    bltz t2, inner_done_insertion    # neu j < 0, thoat
    # So sanh cac phan tu
    slli t3, t2, 2
    add t3, s0, t3
    lw t4, 0(t3)        # arr[j]
    ble t4, t1, inner_done_insertion
    # Di chuyen phan tu
    sw t4, 4(t3)        # arr[j+1] = arr[j]
    addi t2, t2, -1     # j--
    j inner_loop_insertion
inner_done_insertion:
    # Dat key vao vi tri chinh xac
    addi t2, t2, 1
    slli t3, t2, 2
    add t3, s0, t3
    sw t1, 0(t3)
    addi s2, s2, 1      # i++
    j outer_loop_insertion
insertion_done:
    lw ra, 12(sp)
    lw s0, 8(sp)
    lw s1, 4(sp)
    lw s2, 0(sp)
    addi sp, sp, 16
    ret
selection_sort_array_impl:
    # a0 = dia chi mang
    # a1 = kich thuoc
    addi sp, sp, -16
    sw ra, 12(sp)
    sw s0, 8(sp)
    sw s1, 4(sp)
    sw s2, 0(sp)
    mv s0, a0           # s0 = dia chi mang
    mv s1, a1           # s1 = kich thuoc
    li s2, 0            # s2 = i
outer_loop_selection:
```

```
    addi t0, s1, -1
    bge s2, t0, selection_done
    mv t1, s2           # min_idx = i
    addi t2, s2, 1      # j = i + 1
inner_loop_selection:
    bge t2, s1, inner_done_selection
    # So sanh cac phan tu
    slli t3, t2, 2
    add t3, s0, t3
    lw t4, 0(t3)        # arr[j]
    slli t5, t1, 2
    add t5, s0, t5
    lw t6, 0(t5)        # arr[min_idx]
    bge t4, t6, no_update_min
    mv t1, t2           # Cap nhat min_idx
no_update_min:
    addi t2, t2, 1
    j inner_loop_selection
inner_done_selection:
    # Hoan doi cac phan tu neu can
    beq t1, s2, no_swap_selection
    slli t2, s2, 2
    add t2, s0, t2
    lw t3, 0(t2)        # temp = arr[i]
    slli t4, t1, 2
    add t4, s0, t4
    lw t5, 0(t4)        # arr[min_idx]
    sw t5, 0(t2)        # arr[i] = arr[min_idx]
    sw t3, 0(t4)        # arr[min_idx] = temp
no_swap_selection:
    addi s2, s2, 1
    j outer_loop_selection
selection_done:
    lw ra, 12(sp)
    lw s0, 8(sp)
    lw s1, 4(sp)
    lw s2, 0(sp)
    addi sp, sp, 16
    ret
parse_loop:
    bge s2, s0, read_loop  # Neu da parse het buffer, doc tiep
    # Doc ky tu
    add t0, s1, s2
    lb t1, 0(t0)
    # Kiem tra xem co phai space khong
    li t2, 32           # ASCII cho space
    beq t1, t2, next_char    # Chuyen doi ASCII sang so
    addi t1, t1, -48  # Chuyen ASCII thanh so
    # Luu so vao mang numbers
    lw t3, count
    slli t4, t3, 2    # t4 = count * 4 (de tinh offset)
    la t5, numbers
    add t5, t5, t4
    sw t1, 0(t5)        # Luu so vao mang
    # Tang count
    addi t3, t3, 1
    sw t3, count, t6
next_char:
    addi s2, s2, 1
    j parse_loop
get_time:
    li a7, 30           # Syscall GetTime
    ecall
    ret
print_time:
    # Tinh toan va in thoi gian thuc thi
```

```
    la t0, start_time
    lw t1, 0(t0)
    la t0, end_time
    lw t2, 0(t0)
    sub t3, t2, t1     # Thoi gian thuc thi
    li a7, 4
    la a0, msg_execution_time
    ecall
    li a7, 1
    mv a0, t3
    ecall
    ret
number_to_string:
    # a0 = dia chi buffer
    # a1 = so can chuyen doi
    addi sp, sp, -24
    sw ra, 20(sp)
    sw s0, 16(sp)
    sw s1, 12(sp)
    sw s2, 8(sp)
    sw s3, 4(sp)
    sw s4, 0(sp)
    mv s0, a0          # Luu dia chi buffer
    mv s1, a1          # Luu so can chuyen doi
    li s2, 0           # Khoi tao bo dem do dai
    li s3, 0           # Co danh dau so am
    # Xu ly truong hop so 0
    bnez s1, check_sign
    li t0, 48          # ASCII '0'
    sb t0, 0(s0)
    li a0, 1           # Do dai la 1
    j num_to_str_done
check_sign:
    # Kiem tra so am
    bgez s1, convert_digits
    li s3, 1                    # Dat co danh dau am
    neg s1, s1                  # Chuyen so thanh duong
convert_digits:
    # Chuyen doi cac chu so theo thu tu nguoc lai
    mv t0, s0
digit_loop:
    beqz s1, finalize_string
    li t1, 10
    rem t2, s1, t1     # Lay chu so cuoi cung
    div s1, s1, t1
    addi t2, t2, 48    # Chuyen thanh ASCII
    sb t2, 0(t0)
    addi t0, t0, 1
    addi s2, s2, 1
    j digit_loop
finalize_string:
    # Them dau tru neu la so am
    beqz s3, reverse_string
    li t1, 45          # ASCII '-'
    sb t1, 0(t0)
    addi t0, t0, 1
    addi s2, s2, 1
reverse_string:
    mv a0, s0
    addi a1, t0, -1    # Cuoi chuoi
    jal str_reverse
    mv a0, s2          # Tra ve do dai
num_to_str_done:
    lw ra, 20(sp)
    lw s0, 16(sp)
    lw s1, 12(sp)
```

```
    lw s2, 8(sp)
    lw s3, 4(sp)
    lw s4, 0(sp)
    addi sp, sp, 24
    ret
# Ham tro giup dao nguoc chuoi tai cho
str_reverse:
    # a0 = dia chi bat dau
    # a1 = dia chi ket thuc
    bge a0, a1, str_rev_done

    # Hoan doi ky tu
    lb t0, 0(a0)
    lb t1, 0(a1)
    sb t1, 0(a0)
    sb t0, 0(a1)

    # Di chuyen con tro
    addi a0, a0, 1
    addi a1, a1, -1
    j str_reverse
str_rev_done:
    ret
write_results:
    addi sp, sp, -16
    sw ra, 12(sp)
    sw s0, 8(sp)
    sw s1, 4(sp)
    sw s2, 0(sp)
    # Mo file results.txt de ghi
    li a7, 1024       # Syscall Open file
    la a0, output_filename    # input_filename
    li a1, 1          # Chi ghi
    li a2, 0x1ff      # Quyen truy cap file (777 trong octal)
    ecall
    # Kiem tra neu mo file thanh cong
    bltz a0, msg_file_error_openor
    sw a0, out_fd, t0 # Luu mo ta file
    # Ghi so vao file
    li s0, 0          # Khoi tao bo dem
    lw s1, count      # Tai tong so luong
    la s2, numbers    # Tai dia chi mang
write_loop:
    bge s0, s1, write_done
    # Tai so hien tai
    slli t0, s0, 2
    add t1, s2, t0
    lw t2, 0(t1)       # Tai so
    # Chuyen so thanh chuoi
    la a0, buffer_number
    mv a1, t2
    jal number_to_string
    mv t3, a0          # t3 = do dai cua chuoi
    # Ghi so vao file
    li a7, 64          # Syscall WriteFile
    lw a0, out_fd
    la a1, buffer_number
    mv a2, t3          # Do dai chinh xac
    ecall
    # Ghi dau cach sau so (tru so cuoi cung)
    addi t0, s1, -1
    bge s0, t0, skip_space
    li a7, 64
    lw a0, out_fd
    la a1, space
    li a2, 1
```

```
    ecall
skip_space:
    addi s0, s0, 1
    j write_loop
write_done:
    # Ghi newline o cuoi
    li a7, 64
    lw a0, out_fd
    la a1, newline
    li a2, 1
    ecall
    # Dong file output
    li a7, 57          # Syscall Close file
    lw a0, out_fd
    ecall
    lw ra, 12(sp)
    lw s0, 8(sp)
    lw s1, 4(sp)
    lw s2, 0(sp)
    addi sp, sp, 16
    ret
msg_file_error_openor:
    # In thong bao loi
    li a7, 4
    la a0, msg_file_error_open
    ecall
    lw ra, 12(sp)
    lw s0, 8(sp)
    lw s1, 4(sp)
    lw s2, 0(sp)
    addi sp, sp, 16
    ret
write_positive_numbers:
    # Tai so
    slli t0, s0, 2
    add t1, s2, t0
    lw t2, 0(t1)
    # Lay gia tri tuyet doi thu cong
    bgez t2, skip_abs
    neg t2, t2
skip_abs:

    la a0, buffer_number
    mv a1, t2
    jal number_to_string # Ghi so
    li a7, 64
    lw a0, out_fd
    la a1, buffer_number
    mv a2, a0          # Do dai tra ve boi number_to_string
    ecall
    # Ghi dau cach (tru so cuoi cung)
    addi t0, s1, -1
    bge s0, t0, skip_space
    li a7, 64
    lw a0, out_fd
    la a1, space
    li a2, 1
    ecall
check_negative:
    bgez s1, positive_conversion   # Neu so >= 0, bo qua xu ly so am
    # Xu ly so am
    li t0, 45          # ASCII '-'
    sb t0, 0(s0)       # Luu ky tu tru
    addi s0, s0, 1     # Di chuyen con tro buffer
    addi s2, s2, 1     # Tang do dai
    neg s1, s1         # Chuyen so thanh duong
```

```
750 positive_conversion:
751     mv t0, s0          # Vi tri hien tai trong buffer
752     mv t1, s1          # Ban sao lam viec cua so
753 reverse_digits:
754     mv a0, s0          # Bat dau cua chuoi
755     add a1, s0, s2
756     addi a1, a1, -1    # Cuoi chuoi
757     # Them ky tu ket thuc
758     add t0, s0, s2
759     sb zero, 0(t0)
760     jal str_reverse
761     mv a0, s2          # Tra ve do dai
762 exit:
763     li a7, 10          # Syscall Exit
764     ecall
```