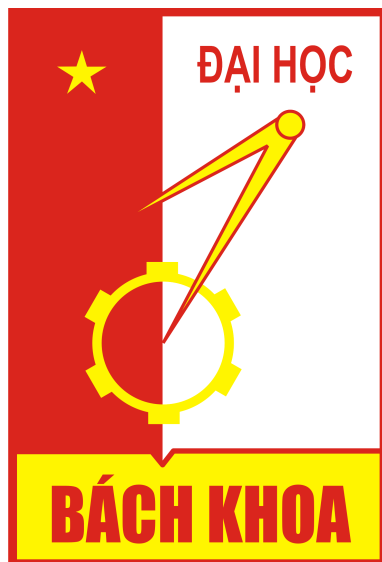


ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



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

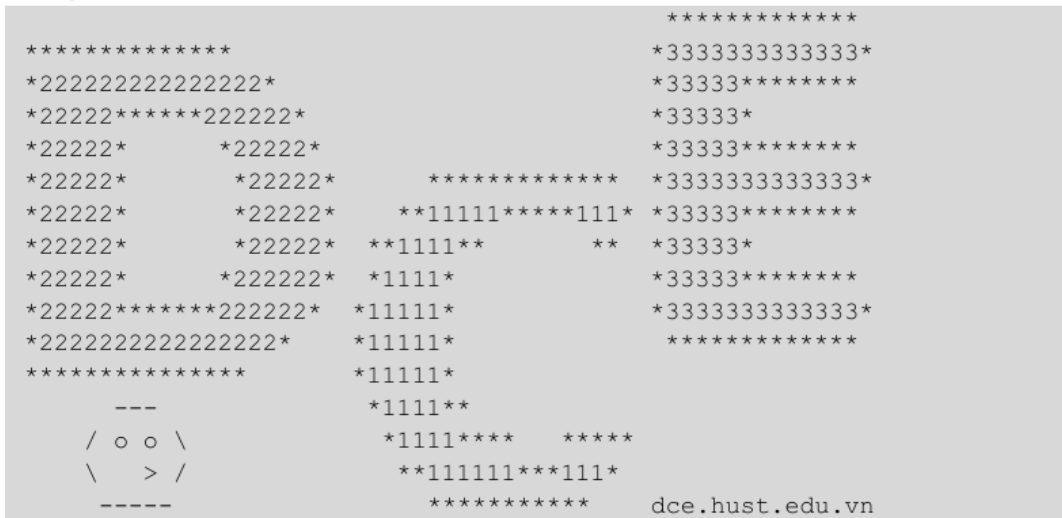
1	Drawing shape using ASCII characters	2
1.1	Đề bài	2
1.2	Thuật toán và hàm	2
1.2.1	Khai báo dữ liệu	2
1.2.2	Phần mã chính:	3
1.2.3	Menu 1: In Đầy Đủ	4
1.2.4	Menu 2: In Không Màu	5
1.2.5	Menu 3: In ECD	6
1.2.6	Menu 4: Nhập màu và thay đổi màu của DCE	7
1.3	Kết quả	10
1.3.1	Khi chạy menu 1	10
1.3.2	Khi chạy menu 2	10
1.3.3	Khi chạy menu 3	11
1.3.4	Khi chạy menu 4	11
2	Testing sorting algorithms	16
2.1	Đề bài	16
2.2	Thuật toán và hàm	16
2.2.1	Khai báo dữ liệu	16
2.2.2	Đọc tên file đầu vào và đầu ra:	17
2.2.3	Lựa chọn thuật toán sắp xếp:	17
2.2.4	Kiểm tra lựa chọn thuật toán hợp lệ:	18
2.2.5	Mở và đọc file:	18
2.2.6	Đọc dữ liệu từ file và chuyển đổi:	18
2.2.7	Sắp xếp mảng theo thuật toán đã chọn:	19
2.2.8	Đóng file và thoát chương trình:	19
2.3	Kết quả	19
2.3.1	Khi chạy Bubble Sort	19
2.3.2	Khi chạy Insertion Sort	19
2.3.3	Khi chạy Selection Sort	20

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.



- 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 "*****
      *333333333333*\n"
4 line3: .asciz "*22222222222222*      *33333*****
      \n"
5 line4: .asciz "*22222*****22222*      *33333*
      \n"
6 line5: .asciz "*22222*      *22222*      *33333*****
      \n"
7 line6: .asciz "*22222*      *22222*      *****
      *333333333333*\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*****22222*      *11111*
      *333333333333*\n"
12 line11: .asciz "*2222222222222222*      *11111*      *****
      \n"
13 line12: .asciz "*****      *11111*
```

```

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 Message0: .string "-----Menu-----\n"
19 Message1: .string "1. Print with color\n"
20 Message2: .string "2. Print without color\n"
21 Message3: .string "3. Change order\n"
22 Message4: .string "4. Change color\n"
23 Message5: .string "5. Exit\n"
24 Message6: .string "Enter choice: "
25 Message4_1: .string "Enter color for D(0->9): "
26 Message4_2: .string "Enter color for C(0->9): "
27 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:

```

1      li t0, 50          # t0: curr D color (ASCII '2' = 50)
2      li t1, 49          # t1: curr C color (ASCII '1' = 49)
3      li t2, 51          # t2: curr E color (ASCII '3' = 51)
4  menu:
5      la a0, Message0
6      li a7, 4
7      ecall
8      li a7, 5
9      ecall
10     addi t3, a0, 0      # L u l a c h n n g í d n g v o t3
11     li t4, 1
12     li t5, 2
13     li t6, 3
14     li s0, 4
15     li s1, 5
16     beq t3, t4, menu1
17     beq t3, t5, menu2
18     beq t3, t6, menu3
19     beq t3, s0, menu4
20     beq t3, s1, end_main
21     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: Đặt 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 Đây Đủ

```

1 menu1:
2     # X? l í in d ng c m u
3     li t0, 0
4     li t1, 16
5     la t2, line1
6 loop1:
7     beq t0, t1, menu
8     mv a0, t2
9     li a7, 4
10    ecall
11    addi t2, t2, 60
12    addi t0, t0, 1
13    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

```
1 menu2:
2     li t0, 0 #i=0
3     li t1, 16 #max =16
4
5     la t2,line1 #t2: pointer to character, starting at first character of
        line1
6 outer_loop2:
7     beq t0, t1, menu #i=16 -> main
8     li t3, 0 #j=0
9     li t4, 60 #max=60
10 inner_loop2:
11     beq t3, t4, continue_outer_loop2 #j=60 -> continue_outer_loop2
12     lb t5, 0(t2) #t5: cur char
13     li t6, 48
14     li s0, 57
15     blt t5, t6, print_char2
16     bgt t5, s0, print_char2
17     li t5, ' ' #if char is digit, replace it with blank space
18 print_char2:
19     li a7, 11
20     mv a0, t5
21     ecall
22 continue_inner_loop2:
23     addi t2, t2, 1 #move to next char
24     addi t3, t3, 1 #j=j+1
25     j inner_loop2
26 continue_outer_loop2:
27     addi t0, t0, 1 #i=i+1
28     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

```

1 menu3:
2     # Thay ??i th? t? in
3     li t0, 0
4     li t1, 16
5     la t2, line1
6 loop3:
7     beq t0, t1, menu
8     sb zero, 22(t2)
9     sb zero, 42(t2)
10    sb zero, 58(t2)
11
12    li a7, 4
13    addi a0, t2, 43
14    ecall
15
16    li a7, 11
17    li a0, 32          # In kho?ng tr?ng
18    ecall
19
20    li a7, 4
21    addi a0, t2, 23
22    ecall
23
24    li a7, 11
25    li a0, 32
26    ecall
27
28    li a7, 4
29    mv a0, t2
30    ecall
31
32    li a7, 11
33    li a0, 10          # In xu?ng d ng
34    ecall
35
36    # Ph?c h?i
37    li t3, 32
38    sb t3, 22(t2)
39    sb t3, 42(t2)
40    li t3, 10
41    sb t3, 58(t2)
42
43    addi t0, t0, 1
44    addi t2, t2, 60
45    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 trở 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

```

1  menu4:
2  # Kh?i t?o m u ban ??u
3  li s3, 50          # t0: curr D color (ASCII '2' = 50)
4  li s4, 49          # t1: curr C color (ASCII '1' = 49)
5  li s5, 51          # t2: curr E color (ASCII '3' = 51)
6
7  # X? l í thay ??i m u
8  enter_D:
9  li a7, 4           # Print "Enter color for D"
10 la a0, Message4_1
11 ecall
12
13 li a7, 5           # Read user input
14 ecall
15 mv s8, a0          # L?u gi tr? nh?p v o t6
16 li s0, 48          # ASCII '0' = 48
17 li s1, 57          # ASCII '9' = 57
18 blt s8, s0, enter_D # N?u t6 < '0', nh?p l?i
19 bgt s8, s1, enter_D # N?u t6 > '9', nh?p l?i
20 enter_C:

```



```

21     li a7, 4                # Print "Enter color for C"
22     la a0, Message4_2
23     ecall
24
25     li a7, 5                # Read user input
26     ecall
27     mv s9, a0
28     blt s9, s0, enter_C    # L?u gi tr? nh?p v o t6
29     bgt s9, s1, enter_C    # N?u t6 < '0', nh?p l?i
30     # N?u t6 > '9', nh?p l?i
31 enter_E:
32     li a7, 4                # Print "Enter color for E"
33     la a0, Message4_3
34     ecall
35
36     li a7, 5                # Read user input
37     ecall
38     mv s10, a0
39     blt s10, s0, enter_E    # L?u gi tr? nh?p v o t6
40     # N?u t6 < '0', nh?p l?i
41     bgt s10, s1, enter_E    # N?u t6 > '9', nh?p l?i
42 init_menu4:
43     li t0, 0                # i = 0
44     li t1, 16               # max = 16
45
46     la s2, line1            # s2: con tr? ??n k í t? ??u ti n c?a line1
47 outer_loop4:
48     beq t0, t1, update_color # N?u i == 16, tho t v ng l?p
49     li t3, 0                # j = 0
50     li t4, 60               # max = 60 (s? k í t? tr n m?i d ng)
51 inner_loop4:
52     beq t3, t4, continue_outer_loop4 # N?u j == 60, tho t v ng l?p trong
53     lb t5, 0(s2)            # t5: k í t? hi?n t?i
54
55     li s6, 22
56     li s7, 42
57     blt t3, s6, check_D    # N?u j < 22, thu?c D
58     blt t3, s7, check_C    # N?u j < 42, thu?c C
59     j check_E              # Ng??c l?i, thu?c E
60 check_D:
61     beq t5, s3, update_D    # N?u k í t? tr ng m u D hi?n t?i, c?p nh?t
62     j print_char4
63
64 check_C:
65     beq t5, s4, update_C    # N?u k í t? tr ng m u C hi?n t?i, c?p nh?t
66     j print_char4
67
68 check_E:
69     beq t5, s5, update_E    # N?u k í t? tr ng m u E hi?n t?i, c?p nh?t
70     j print_char4
71
72 update_D:
73     sb s8, 0(s2)            # L?u m u m?i c?a D v o b? nh?
74     mv t5, s8              # C?p nh?t gi tr? k í t? t5
75     j print_char4
76
77 update_C:
78     sb s9, 0(s2)            # L?u m u m?i c?a C v o b? nh?
79     mv t5, s9              # C?p nh?t gi tr? k í t? t5
80     j print_char4
81
82 update_E:
83     sb s10, 0(s2)           # L?u m u m?i c?a E v o b? nh?
84     mv t5, s10             # C?p nh?t gi tr? k í t? t5
85     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

```

-----Menu-----
1. Print with color
2. Print without color
3. Change order
4. Change color
5. Exit
Enter choice: 1

*****
*****
*2222222222222222*
*22222*****22222*
*22222*          *22222*
*22222*          *22222*          *****
*22222*          *22222*          *11111*****111*
*22222*          *22222*          *11111*          *
*22222*          *222222*          *11111*
*22222*****22222*          *11111*
*2222222222222222*          *11111*          *****
*****
*11111*
*11111*
*11111*****
*1111111111111111*
*****
dce.hust.edu.cn

```

1.3.2 Khi chạy menu 2

[illegible]

1.3.3 Khi chạy menu 3

```

-----Menu-----
1. Print with color
2. Print without color
3. Change order
4. Change color
5. Exit
Enter choice: 3

*****
*33333333333333*                *****
*33333*****          *22222222222222*
*33333*              *22222*****22222*
*33333*****          *22222*        *22222*
*33333333333333*      *****       *22222*        *22222*
*33333*****          *11111*****111*   *22222*        *22222*
*33333*            **1111**           **   *22222*        *22222*
*33333*****          *1111*             *22222*        *22222*
*33333333333333*    *11111*           *22222*****22222*
*****             *11111*           *22222222222222*
*11111*            *****
*1111*
*1111*****     *****      / o o \
*11111111***111*      \   > /
*****
dce.hust.edu.vn      *****

```

1.3.4 Khi chay menu 4

```

-----Menu-----
1. Print with color
2. Print without color
3. Change order
4. Change color
5. Exit
Enter choice: 4
Enter color for D(0->9): 54
Enter color for C(0->9): 53
Enter color for E(0->9): 57

                                     *****
*****
*666666666666666666*
*66666*****666666*
*66666*          *66666*
*66666*          *66666*          *****
*66666*          *66666*          **55555*****555*
*66666*          *66666*          **5555*          **
*66666*          *6666666*          *5555*
*66666*****666666*          *55555*
*666666666666666666*          *55555*
*****                      *****
*****                      *5555*
---
/  o o  \
\   >   /
-----
                                     *****
                                     dce.hust.edu.cn

```

→ Chương trình chạy đúng

```

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

```

```

4 line3: .asciz  "*22222222222222*"                      *33333*****
   \n"
5 line4: .asciz  "*22222*****22222*"                      *33333*
   \n"
6 line5: .asciz  "*22222*          *22222*"                *33333*****
   \n"
7 line6: .asciz  "*22222*          *22222*          *****"
   *33333333333333*\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*****22222*"          *11111*"
   *33333333333333*\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

```

```

56     la a0, Message4
57     li a7, 4
58     ecall
59
60     la a0, Message5
61     li a7, 4
62     ecall
63
64     la a0, Message6
65     li a7, 4
66     ecall
67
68     li a7, 5
69     ecall
70     addi t3, a0, 0      # L?u l?a ch?n v o t3
71
72     # ?i?u h??ng menu
73     li t4, 1
74     li t5, 2
75     li t6, 3
76     li s0, 4
77     li s1, 5
78     beq t3, t4, menu1
79     beq t3, t5, menu2
80     beq t3, t6, menu3
81     beq t3, s0, menu4
82     beq t3, s1, end_main
83     j menu
84
85 menu1:
86     # X? l í in d ng c m u
87     li t0, 0
88     li t1, 16
89     la t2, line1
90 loop1:
91     beq t0, t1, menu
92     mv a0, t2
93     li a7, 4
94     ecall
95     addi t2, t2, 60
96     addi t0, t0, 1
97     j loop1
98
99 menu2:
100    li t0, 0 #i=0
101    li t1, 16 #max =16
102
103    la t2, line1 #t2: pointer to character, starting at first character of
        line1
104 outer_loop2:
105     beq t0, t1, menu #i=16 -> main
106     li t3, 0 #j=0
107     li t4, 60 #max=60
108 inner_loop2:
109     beq t3, t4, continue_outer_loop2 #j=60 -> continue_outer_loop2
110     lb t5, 0(t2) #t5: cur char
111     li t6, 48
112     li s0, 57
113     blt t5, t6, print_char2
114     bgt t5, s0, print_char2
115     li t5, ' ' #if char is digit, replace it with blank space
116 print_char2:
117     li a7, 11
118     mv a0, t5
119     ecall
120 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

```



```

252     j print_char4
253
254 update_E:
255     sb s10, 0(s2)           # L?u m u m?i c?a E v o b? nh?
256     mv t5, s10             # C?p nh?t gi tr? k í t? t5
257     j print_char4
258
259 print_char4:
260     li a7, 11               # ecall ?? in k í t?
261     mv a0, t5
262     ecall
263
264 continue_inner_loop4:
265     addi s2, s2, 1          # Di chuy?n ??n k í t? ti?p theo
266     addi t3, t3, 1          # j = j + 1
267     j inner_loop4
268
269 continue_outer_loop4:
270     addi t0, t0, 1          # i = i + 1
271     j outer_loop4
272
273 update_color:
274     mv t0, s8               # C?p nh?t m u D hi?n t?i
275     mv t1, s9               # C?p nh?t m u C hi?n t?i
276     mv t2, s10              # C?p nh?t m u E hi?n t?i
277     j menu
278
279
280
281 end_main:
282     li a7, 10
283     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

```

1     .data
2     numbers:      .space   80000
3     input_buffer_size: .word   80000
4     count:        .word    0
5
6     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 $prompt_{in}$, $prompt_{out}$, $prompt_{algo}$, $error_{msg}$, $invalid_{algo}$ cho các thông báo yêu cầu người dùng nhập vào.
- $algo_{choice}$ là chuỗi chọn thuật toán sắp xếp.

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 khi 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

```

```

13     beq a0, t0, bubble_sort_array
14     li t0, 2
15     beq a0, t0, insertion_sort_array
16     li t0, 3
17     beq a0, t0, selection_sort_array
18     li t0, 4
19     beq a0, t0, quick_sort_array
20     li t0, 5
21     beq a0, t0, exit
22     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 *algochoice*.

2.2.4 Kiểm tra lựa chọn thuật toán hợp lệ:

```

1     read_loop:
2     # Doc mot ky tu tu file
3     li a7, 63          # Syscall ReadFile
4     lw a0, fd          # Mo ta file
5     la a1, file_read_buffer # Dia chi buffer
6     li a2, 1          # Doc mot ky tu
7     ecall
8
9     # Kiem tra neu cuoi file
10    beqz a0, read_done
11
12    # Tai ky tu
13    lb t2, 0(a1)
14
15    # Kiem tra dau tru
16    li t3, 45          # ASCII cho '-'
17    bne t2, t3, not_char_minus
18    beqz t1, set_negative # Chi dat am neu o dau so
19    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:

```

1     open_input_file:
2     li a7, 1024
3     la a0, input_filename
4     li a1, 0
5     ecall
6     bltz a0, file_error_open
7     la t1, fd
8     sw a0, 0(t1)
9     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:

```

1     read_numbers:
2     addi sp, sp, -16
3     sw ra, 12(sp)
4     sw s0, 8(sp)
5     sw s1, 4(sp)

```

```

6      sw s2, 0(sp)
7
8      # Reset count
9      la t1, count          # Tai dia chi cua count
10     sw zero, 0(t1)        # Luu gia tri zero vao count
11
12     # Khoi tao bien
13     li t0, 0              # So hien tai dang duoc xay dung
14     li t1, 0              # Co cho biet dang trong so
15     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:

```

1      # Choose sorting algorithm based on user input
2      la t0, algo_choice
3      lw t1, 0(t0)
4      li t2, 1
5      beq t1, t2, do_bubble_sort
6
7      do_insertion_sort:
8      # Call insertion sort
9      la a0, array
10     la t3, array_length
11     lw a1, 0(t3)
12     jal insertion_sort

```

- Nếu người dùng chọn Bubble Sort (tương ứng với giá trị 1), hàm `bubble_sortsOcgi.Nungidngchnthuttonkhe(n`

2.2.8 Đóng file và thoát chương trình:

```

1      close_file:
2      # Dong file
3      li a7, 57             # Syscall Close file
4      lw a0, fd
5      ecall
6
7      lw ra, 12(sp)
8      lw s0, 8(sp)
9      lw s1, 4(sp)
10     lw s2, 0(sp)
11     addi sp, sp, 16
12     ret

```

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

2.3 Kết quả

2.3.1 Khi chạy Bubble Sort

```

s KUNIU
Enter input file name: D:\HUST\Computer Architecture Lab\sort.txt
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): 1
1 2 2 3 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) --

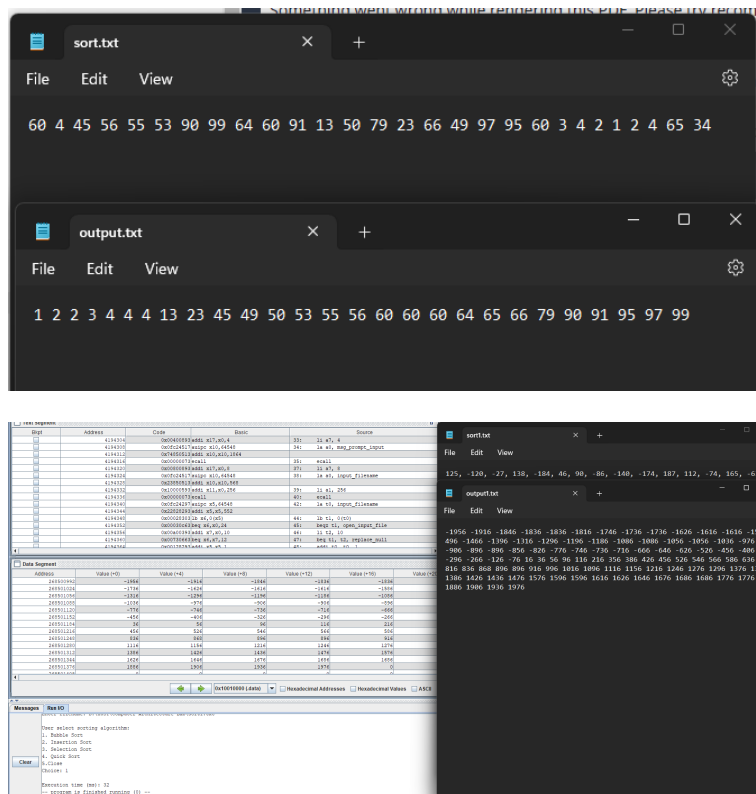
```

2.3.2 Khi chạy Insertion 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 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) --

```



Hình 1: Output khi Chạy 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

```
1 .data
2     numbers: .space 80000
3     input_buffer_size: .word 80000
4     count: .word 0
5     neg_bitmask: .space 2500
6     input_filename: .space 256
7     file_read_buffer: .space 1024
8     msg_prompt_input: .string "Enter filename: "
9     error_msg: .string "\nError opening file\n"
10    menu: .string "\nUser select sorting algorithm:\n1. Bubble
        Sort\n2. Insertion Sort\n3. Selection Sort\n4. Quick
        Sort\n5. Close\nChoice: "
11    fd: .word 0
12    newline: .string "\n"
13    space: .string " "
14    start_time: .word 0
15    end_time: .word 0
16    msg_execution_time: .string "\nExecution time (ms): "
17    output_filename: .string "D:/HUST/Computer Architecture
        Lab/output1.txt"
18    out_fd: .word 0
19    buffer_number: .space 12
20    msg_file_error_open: .string "\nError writing to output file\n"
21    char_minus: .string "-"
22    tmp_sort_buffer: .space 80000
23 .text
```

```

24 .globl main
25 main:
26     # print msg_prompt_input
27     li a7, 4
28     la a0, msg_prompt_input
29     ecall
30     # read input_filename
31     li a7, 8
32     la a0, input_filename
33     li a1, 256
34     ecall
35     # delete newline from input_filename
36     la t0, input_filename
37 remove_newline_from_filename:
38     lb t1, 0(t0)
39     beqz t1, open_input_file
40     li t2, 10
41     beq t1, t2, replace_null
42     addi t0, t0, 1
43     j remove_newline_from_filename
44 replace_null:
45     sb zero, 0(t0)
46 open_input_file:
47     li a7, 1024
48     la a0, input_filename
49     li a1, 0
50     ecall
51     bltz a0, file_error_open
52     la t1, fd
53     sw a0, 0(t1)
54     jal read_numbers
55 menu_loop:
56     li a7, 4
57     la a0, menu
58     ecall
59     li a7, 5
60     ecall
61     li t0, 1
62     beq a0, t0, bubble_sort_array
63     li t0, 2
64     beq a0, t0, insertion_sort_array
65     li t0, 3
66     beq a0, t0, selection_sort_array
67     li t0, 4
68     beq a0, t0, quick_sort_array
69     li t0, 5
70     beq a0, t0, exit
71     j exit
72 file_error_open:
73     li a7, 4
74     la a0, error_msg
75     ecall
76     j exit
77 read_numbers:
78     addi sp, sp, -16
79     sw ra, 12(sp)
80     sw s0, 8(sp)
81     sw s1, 4(sp)
82     sw s2, 0(sp)
83     # Reset count
84     la t1, count
85     sw zero, 0(t1)
86     li t0, 0
87     li t1, 0
88     li t6, 0
89 read_loop:

```

So hien tai dang duoc xay dung

Co cho biet dang trong so

Co danh dau (0 = duong, 1 = am)

```

90     # Doc mot ky tu tu file
91     li a7, 63          # Syscall ReadFile
92     lw a0, fd          # Mo ta file
93     la a1, file_read_buffer    # Dia chi buffer
94     li a2, 1          # Doc mot ky tu
95     ecall
96     # Kiem tra neu cuoi file
97     beqz a0, read_done
98     # Tai ky tu
99     lb t2, 0(a1)
100    # Kiem tra dau tru
101    li t3, 45          # ASCII cho '-'
102    bne t2, t3, not_char_minus
103    beqz t1, set_negative    # Chi dat am neu o dau so
104    j read_loop
105    set_negative:
106    li t6, 1          # Dat co danh dau la am
107    li t1, 1          # Dat co trong so
108    j read_loop
109    not_char_minus:
110    # Kiem tra neu la dau cach hoac newline
111    li t3, 32          # Space
112    beq t2, t3, save_number
113    li t3, 10          # Newline
114    beq t2, t3, save_number
115    # Chuyen doi ASCII thanh so va cong vao so hien tai
116    addi t2, t2, -48    # Chuyen doi ASCII thanh so
117    li t3, 10
118    mul t0, t0, t3     # So hien tai * 10
119    add t0, t0, t2     # Cong vao chu so moi
120    li t1, 1          # Dat co trong so
121    j read_loop
122    save_number:
123    beqz t1, read_loop    # Neu khong trong so, tiep tuc
124    # Ap dung dau neu la am
125    beqz t6, save_positive
126    neg t0, t0          # Doi dau so neu co am duoc dat
127    save_positive:
128    # Luu so vao mang
129    la t3, count        # Tai dia chi cua count
130    lw t3, 0(t3)        # Tai gia tri count
131    slli t4, t3, 2      # t4 = count * 4
132    la t5, numbers
133    add t5, t5, t4
134    sw t0, 0(t5)        # Luu so
135    # Tang count
136    addi t3, t3, 1
137    la t4, count        # Tai dia chi cua count
138    sw t3, 0(t4)        # Luu count moi
139    # Reset cho so tiep theo
140    li t0, 0            # Reset so hien tai
141    li t1, 0            # Reset co trong so
142    li t6, 0            # Reset co am
143    j read_loop
144    read_done:
145    # Neu chung ta dang trong so khi file ket thuc, luu no
146    beqz t1, close_file
147    # Ap dung dau neu la am
148    beqz t6, save_last_positive
149    neg t0, t0          # Doi dau so neu co am duoc dat
150    save_last_positive:
151    la t3, count        # Tai dia chi cua count
152    lw t3, 0(t3)        # Tai gia tri count
153    slli t4, t3, 2
154    la t5, numbers
155    add t5, t5, t4

```

```

156     sw t0, 0(t5)
157     addi t3, t3, 1
158     la t4, count           # Tai dia chi cua count
159     sw t3, 0(t4)          # Luu count moi
160 close_file:
161     # Dong file
162     li a7, 57              # Syscall Close file
163     lw a0, fd
164     ecall
165     lw ra, 12(sp)
166     lw s0, 8(sp)
167     lw s1, 4(sp)
168     lw s2, 0(sp)
169     addi sp, sp, 16
170     ret
171 flag_negative_numbers:
172     # a0 = dia chi mang
173     # a1 = size
174     addi sp, sp, -16
175     sw ra, 12(sp)
176     sw s0, 8(sp)
177     sw s1, 4(sp)
178     sw s2, 0(sp)
179     mv s0, a0              # s0 = dia chi mang
180     mv s1, a1              # s1 = size
181     li s2, 0               # s2 = bo dem
182 flag_loop:
183     bge s2, s1, flag_done
184     # Tai so hien tai
185     slli t0, s2, 2         # t0 = bo dem * 4
186     add t0, s0, t0
187     lw t1, 0(t0)          # Tai so
188     # Bo qua neu duong
189     bgez t1, skip_flag
190     # Tinh toan byte va vi tri bit trong bitmask
191     mv t0, s2              # Sao chep chi so
192     srai t1, t0, 3         # Byte offset = chi so / 8
193     andi t2, t0, 0x7       # Vi tri bit = chi so % 8
194     li t3, 1
195     sll t3, t3, t2         # Dich chuyen 1 den vi tri bit chinh xac
196     # Dat bit trong bitmask
197     la t4, neg_bitmask
198     add t4, t4, t1         # Them byte offset
199     lb t5, 0(t4)           # Tai byte hien tai
200     or t5, t5, t3          # Dat bit
201     sb t5, 0(t4)          # Luu byte da cap nhat
202 skip_flag:
203     addi s2, s2, 1
204     j flag_loop
205 flag_done:
206     lw ra, 12(sp)
207     lw s0, 8(sp)
208     lw s1, 4(sp)
209     lw s2, 0(sp)
210     addi sp, sp, 16
211     ret
212 quick_sort_array:
213     # Lay thoi gian bat dau
214     jal get_time
215     sw a0, start_time, t0
216     # Khoi tao quicksort
217     la a0, numbers
218     li a1, 0
219     lw a2, count
220     addi a2, a2, -1
221     jal quick_sort_logic

```



```

222     # Danh dau cac so am
223     la a0, numbers
224     lw a1, count
225     jal flag_negative_numbers
226     # Lay thoi gian ket thuc va tinh thoi gian thuc thi
227     jal get_time
228     sw a0, end_time, t0
229     jal print_time
230     # Ghi ket qua vao file
231     jal write_results
232     j menu_loop
233 quick_sort_logic:
234     # a0 = dia chi mang
235     # a1 = chi so ben trai
236     # a2 = chi so ben phai
237     addi sp, sp, -24
238     sw ra, 20(sp)
239     sw s0, 16(sp)
240     sw s1, 12(sp)
241     sw s2, 8(sp)
242     sw s3, 4(sp)
243     sw s4, 0(sp)
244     # Luu cac tham so
245     mv s0, a0           # s0 = dia chi mang
246     mv s1, a1           # s1 = ben trai
247     mv s2, a2           # s2 = ben pha
248     # Dieu kien dung: neu ben trai >= ben phai, thoat
249     bge s1, s2, quick_sort_end
250     # Goi ham partition_elements
251     mv a0, s0           # dia chi mang
252     mv a1, s1           # chi so ben trai
253     mv a2, s2           # chi so ben phai
254     jal partition_elements
255     mv s3, a0           # s3 = chi so pivot
256     # De quy sap xep phan ben trai
257     mv a0, s0           # dia chi mang
258     mv a1, s1           # chi so ben trai
259     addi a2, s3, -1     # pivot - 1
260     jal quick_sort_logic
261     # De quy sap xep phan ben phai
262     mv a0, s0           # dia chi mang
263     addi a1, s3, 1      # pivot + 1
264     mv a2, s2           # chi so ben phai
265     jal quick_sort_logic
266 quick_sort_end:
267     lw ra, 20(sp)
268     lw s0, 16(sp)
269     lw s1, 12(sp)
270     lw s2, 8(sp)
271     lw s3, 4(sp)
272     lw s4, 0(sp)
273     addi sp, sp, 24
274     ret
275 partition_elements:
276     addi sp, sp, -24
277     sw ra, 20(sp)
278     sw s0, 16(sp)
279     sw s1, 12(sp)
280     sw s2, 8(sp)
281     sw s3, 4(sp)
282     sw s4, 0(sp)
283     mv s0, a0           # s0 = dia chi mang
284     mv s1, a1           # s1 = ben trai
285     mv s2, a2           # s2 = ben phai
286     slli t0, s2, 2      # t0 = ben phai * 4
287     add t0, s0, t0

```

```

288     lw s3, 0(t0)          # s3 = gia tri pivot
289     addi s4, s1, -1       # i = ben trai - 1
290     mv t1, s1             # j = ben trai
291 partition_loop_elements:
292     bge t1, s2, partition_elements_done
293     # Tai phan tu hien tai
294     slli t0, t1, 2
295     add t0, s0, t0
296     lw t2, 0(t0)          # t2 = arr[j]
297     # So sanh voi pivot
298     bgt t2, s3, skip_swap
299     # Tang i
300     addi s4, s4, 1
301     # Doi phan tu neu i != j
302     slli t0, s4, 2
303     add t0, s0, t0        # Dia chi cua arr[i]
304     slli t3, t1, 2
305     add t3, s0, t3        # Dia chi cua arr[j]
306     lw t4, 0(t0)          # t4 = arr[i]
307     lw t5, 0(t3)          # t5 = arr[j]
308     sw t5, 0(t0)          # arr[i] = arr[j]
309     sw t4, 0(t3)          # arr[j] = arr[i]
310 skip_swap:
311     addi t1, t1, 1        # j++
312     j partition_loop_elements
313 partition_elements_done:
314     # Dat pivot vao vi tri cuoi cung
315     addi s4, s4, 1        # i++
316     # Doi pivot voi phan tu tai i
317     slli t0, s4, 2
318     add t0, s0, t0        # Dia chi cua arr[i]
319     slli t1, s2, 2
320     add t1, s0, t1        # Dia chi cua arr[right]
321     lw t2, 0(t0)          # t2 = arr[i]
322     lw t3, 0(t1)          # t3 = arr[right]
323     sw t3, 0(t0)          # arr[i] = arr[right]
324     sw t2, 0(t1)          # arr[right] = arr[i]
325     # Tra ve chi so pivot
326     mv a0, s4
327     lw ra, 20(sp)
328     lw s0, 16(sp)
329     lw s1, 12(sp)
330     lw s2, 8(sp)
331     lw s3, 4(sp)
332     lw s4, 0(sp)
333     addi sp, sp, 24
334     ret
335 bubble_sort_array:
336     # Lay thoi gian bat dau
337     jal get_time
338     sw a0, start_time, t0
339     # Thuc hien bubble sort
340     la a0, numbers        # Tai dia chi mang
341     lw a1, count          # Tai kich thuoc mang
342     jal bubble_sort_core
343     # Danh dau cac so am
344     la a0, numbers
345     lw a1, count
346     jal flag_negative_numbers
347     # Lay thoi gian ket thuc va tinh thoi gian thuc thi
348     jal get_time
349     sw a0, end_time, t0
350     jal print_time
351     # Ghi ket qua vao file
352     jal write_results
353     j exit

```

```

354 insertion_sort_array:
355     # Lay thoi gian bat dau
356     jal get_time
357     sw a0, start_time, t0
358     # Thuc hien insertion sort
359     la a0, numbers      # Tai dia chi mang
360     lw a1, count        # Tai kich thuoc mang
361     jal insertion_sort_array_impl
362     # Danh dau cac so am
363     la a0, numbers
364     lw a1, count
365     jal flag_negative_numbers
366     # Lay thoi gian ket thuc va tinh thoi gian thuc thi
367     jal get_time
368     sw a0, end_time, t0
369     jal print_time
370     # Ghi ket qua vao file
371     jal write_results
372     j exit
373 selection_sort_array:
374     # Lay thoi gian bat dau
375     jal get_time
376     sw a0, start_time, t0
377     # Thuc hien selection sort
378     la a0, numbers      # Tai dia chi mang
379     lw a1, count        # Tai kich thuoc mang
380     jal selection_sort_array_impl
381     # Danh dau cac so am
382     la a0, numbers
383     lw a1, count
384     jal flag_negative_numbers
385     # Lay thoi gian ket thuc va tinh thoi gian thuc thi
386     jal get_time
387     sw a0, end_time, t0
388     jal print_time
389     # Ghi ket qua vao file
390     jal write_results
391     j exit
392 bubble_sort_core:
393     # a0 = dia chi mang
394     # a1 = kich thuoc
395     addi sp, sp, -16
396     sw ra, 12(sp)
397     sw s0, 8(sp)
398     sw s1, 4(sp)
399     sw s2, 0(sp)
400     mv s0, a0            # s0 = dia chi mang
401     mv s1, a1            # s1 = kich thuoc
402     li s2, 0             # s2 = i
403 outer_loop_bubble_sort:
404     bge s2, s1, bubble_done
405     li t0, 0             # j = 0
406 inner_loop_bubble_sort:
407     sub t1, s1, s2
408     addi t1, t1, -1
409     bge t0, t1, inner_done_bubble_sort
410     slli t2, t0, 2
411     add t2, s0, t2
412     lw t3, 0(t2)         # arr[j]
413     lw t4, 4(t2)         # arr[j+1]
414     ble t3, t4, no_swap_bubble_sort
415     sw t4, 0(t2)
416     sw t3, 4(t2)
417 no_swap_bubble_sort:
418     addi t0, t0, 1
419     j inner_loop_bubble_sort

```

```

420 inner_done_bubble_sort:
421     addi s2, s2, 1
422     j outer_loop_bubble_sort
423 bubble_done:
424     lw ra, 12(sp)
425     lw s0, 8(sp)
426     lw s1, 4(sp)
427     lw s2, 0(sp)
428     addi sp, sp, 16
429     ret
430 insertion_sort_array_impl:
431     addi sp, sp, -16
432     sw ra, 12(sp)
433     sw s0, 8(sp)
434     sw s1, 4(sp)
435     sw s2, 0(sp)
436     mv s0, a0           # s0 = dia chi mang
437     mv s1, a1           # s1 = kich thuoc
438     li s2, 1            # s2 = i = 1
439 outer_loop_insertion:
440     bge s2, s1, insertion_done
441
442     # Lay phan tu hien tai
443     slli t0, s2, 2      # t0 = i * 4
444     add t0, s0, t0
445     lw t1, 0(t0)        # key = arr[i]
446     addi t2, s2, -1     # j = i-1
447
448 inner_loop_insertion:
449     bltz t2, inner_done_insertion    # neu j < 0, thoat
450     # So sanh cac phan tu
451     slli t3, t2, 2
452     add t3, s0, t3
453     lw t4, 0(t3)        # arr[j]
454     ble t4, t1, inner_done_insertion
455     # Di chuyen phan tu
456     sw t4, 4(t3)        # arr[j+1] = arr[j]
457     addi t2, t2, -1     # j--
458     j inner_loop_insertion
459 inner_done_insertion:
460     # Dat key vao vi tri chinh xac
461     addi t2, t2, 1
462     slli t3, t2, 2
463     add t3, s0, t3
464     sw t1, 0(t3)
465     addi s2, s2, 1     # i++
466     j outer_loop_insertion
467 insertion_done:
468     lw ra, 12(sp)
469     lw s0, 8(sp)
470     lw s1, 4(sp)
471     lw s2, 0(sp)
472     addi sp, sp, 16
473     ret
474 selection_sort_array_impl:
475     # a0 = dia chi mang
476     # a1 = kich thuoc
477     addi sp, sp, -16
478     sw ra, 12(sp)
479     sw s0, 8(sp)
480     sw s1, 4(sp)
481     sw s2, 0(sp)
482     mv s0, a0           # s0 = dia chi mang
483     mv s1, a1           # s1 = kich thuoc
484     li s2, 0            # s2 = i
485 outer_loop_selection:

```

```

486     addi t0, s1, -1
487     bge s2, t0, selection_done
488     mv t1, s2          # min_idx = i
489     addi t2, s2, 1     # j = i + 1
490 inner_loop_selection:
491     bge t2, s1, inner_done_selection
492     # So sanh cac phan tu
493     slli t3, t2, 2
494     add t3, s0, t3
495     lw t4, 0(t3)       # arr[j]
496     slli t5, t1, 2
497     add t5, s0, t5
498     lw t6, 0(t5)       # arr[min_idx]
499     bge t4, t6, no_update_min
500     mv t1, t2          # Cap nhat min_idx
501 no_update_min:
502     addi t2, t2, 1
503     j inner_loop_selection
504 inner_done_selection:
505     # Hoan doi cac phan tu neu can
506     beq t1, s2, no_swap_selection
507     slli t2, s2, 2
508     add t2, s0, t2
509     lw t3, 0(t2)       # temp = arr[i]
510     slli t4, t1, 2
511     add t4, s0, t4
512     lw t5, 0(t4)       # arr[min_idx]
513     sw t5, 0(t2)       # arr[i] = arr[min_idx]
514     sw t3, 0(t4)       # arr[min_idx] = temp
515 no_swap_selection:
516     addi s2, s2, 1
517     j outer_loop_selection
518 selection_done:
519     lw ra, 12(sp)
520     lw s0, 8(sp)
521     lw s1, 4(sp)
522     lw s2, 0(sp)
523     addi sp, sp, 16
524     ret
525 parse_loop:
526     bge s2, s0, read_loop # Neu da parse het buffer, doc tiep
527     # Doc ky tu
528     add t0, s1, s2
529     lb t1, 0(t0)
530     # Kiem tra xem co phai space khong
531     li t2, 32          # ASCII cho space
532     beq t1, t2, next_char # Chuyen doi ASCII sang so
533     addi t1, t1, -48    # Chuyen ASCII thanh so
534     # Luu so vao mang numbers
535     lw t3, count
536     slli t4, t3, 2      # t4 = count * 4 (de tinh offset)
537     la t5, numbers
538     add t5, t5, t4
539     sw t1, 0(t5)       # Luu so vao mang
540     # Tang count
541     addi t3, t3, 1
542     sw t3, count, t6
543 next_char:
544     addi s2, s2, 1
545     j parse_loop
546 get_time:
547     li a7, 30          # Syscall GetTime
548     ecall
549     ret
550 print_time:
551     # Tinh toan va in thoi gian thuc thi

```

```

552     la t0, start_time
553     lw t1, 0(t0)
554     la t0, end_time
555     lw t2, 0(t0)
556     sub t3, t2, t1      # Thoi gian thuc thi
557     li a7, 4
558     la a0, msg_execution_time
559     ecall
560     li a7, 1
561     mv a0, t3
562     ecall
563     ret
564 number_to_string:
565     # a0 = dia chi buffer
566     # a1 = so can chuyen doi
567     addi sp, sp, -24
568     sw ra, 20(sp)
569     sw s0, 16(sp)
570     sw s1, 12(sp)
571     sw s2, 8(sp)
572     sw s3, 4(sp)
573     sw s4, 0(sp)
574     mv s0, a0           # Luu dia chi buffer
575     mv s1, a1           # Luu so can chuyen doi
576     li s2, 0           # Khoi tao bo dem do dai
577     li s3, 0           # Co danh dau so am
578     # Xu ly truong hop so 0
579     bnez s1, check_sign
580     li t0, 48           # ASCII '0'
581     sb t0, 0(s0)
582     li a0, 1           # Do dai la 1
583     j num_to_str_done
584 check_sign:
585     # Kiem tra so am
586     bgez s1, convert_digits
587     li s3, 1           # Dat co danh dau am
588     neg s1, s1         # Chuyen so thanh duong
589 convert_digits:
590     # Chuyen doi cac chu so theo thu tu nguoc lai
591     mv t0, s0
592 digit_loop:
593     beqz s1, finalize_string
594     li t1, 10
595     rem t2, s1, t1      # Lay chu so cuoi cung
596     div s1, s1, t1
597     addi t2, t2, 48     # Chuyen thanh ASCII
598     sb t2, 0(t0)
599     addi t0, t0, 1
600     addi s2, s2, 1
601     j digit_loop
602 finalize_string:
603     # Them dau tru neu la so am
604     beqz s3, reverse_string
605     li t1, 45          # ASCII '-'
606     sb t1, 0(t0)
607     addi t0, t0, 1
608     addi s2, s2, 1
609 reverse_string:
610     mv a0, s0
611     addi a1, t0, -1    # Cuoi chuoai
612     jal str_reverse
613     mv a0, s2         # Tra ve do dai
614 num_to_str_done:
615     lw ra, 20(sp)
616     lw s0, 16(sp)
617     lw s1, 12(sp)

```

```

618     lw s2, 8(sp)
619     lw s3, 4(sp)
620     lw s4, 0(sp)
621     addi sp, sp, 24
622     ret
623     # Ham tro giup dao nguoc chuoi tai cho
624     str_reverse:
625         # a0 = dia chi bat dau
626         # a1 = dia chi ket thuc
627         bge a0, a1, str_rev_done
628
629         # Hoan doi ky tu
630         lb t0, 0(a0)
631         lb t1, 0(a1)
632         sb t1, 0(a0)
633         sb t0, 0(a1)
634
635         # Di chuyen con tro
636         addi a0, a0, 1
637         addi a1, a1, -1
638         j str_reverse
639     str_rev_done:
640         ret
641     write_results:
642         addi sp, sp, -16
643         sw ra, 12(sp)
644         sw s0, 8(sp)
645         sw s1, 4(sp)
646         sw s2, 0(sp)
647         # Mo file results.txt de ghi
648         li a7, 1024          # Syscall Open file
649         la a0, output_filename # input_filename
650         li a1, 1             # Chi ghi
651         li a2, 0x1ff         # QUYEN truy cap file (777 trong octal)
652         ecall
653         # Kiem tra neu mo file thanh cong
654         bltz a0, msg_file_error_openor
655         sw a0, out_fd, t0    # Luu mo ta file
656         # Ghi so vao file
657         li s0, 0             # Khoi tao bo dem
658         lw s1, count         # Tai tong so luong
659         la s2, numbers       # Tai dia chi mang
660     write_loop:
661         bge s0, s1, write_done
662         # Tai so hien tai
663         slli t0, s0, 2
664         add t1, s2, t0
665         lw t2, 0(t1)         # Tai so
666         # Chuyen so thanh chuoi
667         la a0, buffer_number
668         mv a1, t2
669         jal number_to_string
670         mv t3, a0            # t3 = do dai cua chuoi
671         # Ghi so vao file
672         li a7, 64            # Syscall WriteFile
673         lw a0, out_fd
674         la a1, buffer_number
675         mv a2, t3            # Do dai chinh xac
676         ecall
677         # Ghi dau cach sau so (tru so cuoi cung)
678         addi t0, s1, -1
679         bge s0, t0, skip_space
680         li a7, 64
681         lw a0, out_fd
682         la a1, space
683         li a2, 1

```

```

684     ecall
685 skip_space:
686     addi s0, s0, 1
687     j write_loop
688 write_done:
689     # Ghi newline o cuoi
690     li a7, 64
691     lw a0, out_fd
692     la a1, newline
693     li a2, 1
694     ecall
695     # Dong file output
696     li a7, 57          # Syscall Close file
697     lw a0, out_fd
698     ecall
699     lw ra, 12(sp)
700     lw s0, 8(sp)
701     lw s1, 4(sp)
702     lw s2, 0(sp)
703     addi sp, sp, 16
704     ret
705 msg_file_error_openor:
706     # In thong bao loi
707     li a7, 4
708     la a0, msg_file_error_open
709     ecall
710     lw ra, 12(sp)
711     lw s0, 8(sp)
712     lw s1, 4(sp)
713     lw s2, 0(sp)
714     addi sp, sp, 16
715     ret
716 write_positive_numbers:
717     # Tai so
718     slli t0, s0, 2
719     add t1, s2, t0
720     lw t2, 0(t1)
721     # Lay gia tri tuyet doi thu cong
722     bgez t2, skip_abs
723     neg t2, t2
724 skip_abs:
725
726     la a0, buffer_number
727     mv a1, t2
728     jal number_to_string # Ghi so
729     li a7, 64
730     lw a0, out_fd
731     la a1, buffer_number
732     mv a2, a0          # Do dai tra ve boi number_to_string
733     ecall
734     # Ghi dau cach (tru so cuoi cung)
735     addi t0, s1, -1
736     bge s0, t0, skip_space
737     li a7, 64
738     lw a0, out_fd
739     la a1, space
740     li a2, 1
741     ecall
742 check_negative:
743     bgez s1, positive_conversion    # Neu so >= 0, bo qua xu ly so am
744     # Xu ly so am
745     li t0, 45          # ASCII '-'
746     sb t0, 0(s0)        # Luu ky tu tru
747     addi s0, s0, 1      # Di chuyen con tro buffer
748     addi s2, s2, 1      # Tang do dai
749     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    # Cuoai 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

```