ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KỸ THUẬT MÁY TÍNH

TỔ CHỨC VÀ CẤU TRÚC MÁY TÍNH II

(IT012)

Sinh viên: Trần Nguyễn Thái Bình

MSSV: 23520161

Giảng viên hướng dẫn: Nguyễn Thành Nhân

IT012.O21.1 – Lab 05

MỤC LỤC

1. Thao tác với mảng

- Thao tác với mảng Mảng với n phần tử là một chuỗi n phần tử liên tiếp nhau trong bộ nhớ. Thao tác với mảng trong MIPS là thao tác trực tiếp với byte/word trong bộ nhớ.
  - Để cấp phát chuỗi word hoặc byte trong bộ nhớ, có giá trị khởi tao sử dụng ".word" hoặc ".byte" trong ".data"
  - Để cấp phát chuỗi byte không có giá trị khởi tạo trước, sử dụng ".space" trong ".data"
- Cho ba mảng với cấp phát dữ liệu trong bộ nhớ như sau:

```
1. .data
2. array1: .word 5, 6, 7, 8, 1, 2, 3, 9, 10, 4
3. size1: .word 10
4.
5. array2: .byte 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
6. size2: .word 16
7.
8. array3: .space 8
9. size3: .word 8
```

- Mảng array1 có 10 word, kích thước được lưu trong size1; Mảng array2 có 16 byte, kích thước được lưu trong size2; Mảng array3 có 8 byte, kích thước được lưu trong size3.

Viết code trong phần ".text" thực hiện riêng từng phần việc:

- In ra cửa sổ I/O của MARS tất cả các phần tử của mảng array1 và array2
- Gán các giá trị cho mảng array3 sao cho

  array3[i] = array2[i] + array2[size2 - 1 - i]

- Người sử dụng nhập vào mảng thứ mấy và chỉ số phần tử cần lấy trong mảng đó, chương trình xuất ra phần tử tương ứng.

- Chương trình ASM:

```
1. .data
2. array1:                      .word   5, 6, 7, 8, 1, 2, 3, 9, 10, 4
3. size1:                       .word   10
```

```
array2:                         .byte   1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
size2:                          .word   16
array3:                         .space  8
size3:                          .word   8
str_print_arr_1:                .asciiz "xuat mang 1:"
str_print_arr_2:                .asciiz "xuat mang 2:"
str_print_arr_3:                .asciiz "xuat mang 3:"
str_print_prompt_input:         .asciiz "Nhap thu tu mang can truy xuat: "
str_print_prompt_input_index:   .asciiz "Nhap index cua mang: "
tab_char:                       .asciiz "\t"
end_line_char:                  .asciiz "\n"

.text

main:

print_arr_1:
    la      $a0,                str_print_arr_1
    li      $v0,                4
    syscall
    jal     print_end_line
    la      $s1,                array1
    lw      $s2,                size1

    li      $t0,                0

loop_print_arr_1:
    bge     $t0,                $s2,                            continue_1
    sll     $t1,                $t0,                            2
    add     $t1,                $t1,                            $s1
    lw      $a0,                0($t1)
    li      $v0,                1
    syscall
    addi    $t0,                $t0,                            1
    jal     print_tab
    j       loop_print_arr_1

continue_1:
    jal     print_end_line

print_arr_2:
    la      $a0,                str_print_arr_2
    li      $v0,                4
```

```
47.     syscall
48.     jal     print_end_line
49.     la      $s1,                array2
50.     lw      $s2,                size2
51.
52.     li      $t0,                0
53.
54. loop_print_arr_2:
55.     bge     $t0,                $s2,                        continue_2
56.     add     $t1,                $t0,                        $s1
57.     lb      $a0,                0($t1)
58.     li      $v0,                1
59.     syscall
60.     addi    $t0,                $t0,                        1
61.     jal     print_tab
62.     j       loop_print_arr_2
63.
64. continue_2:
65.     jal     print_end_line
66.
67. prepare_arr_3:
68.     la      $s1,                array2
69.     lw      $s2,                size2
70.     lw      $s3,                size2
71.     add     $s3,                $s3,                        $s1
72.     subi    $s3,                $s3,                        1
73.     la      $s4,                array3
74.     lw      $s5,                size3
75.     move    $s6,                $s4
76.     li      $t0,                0
77.     move    $t1,                $s3
78.
79. loop_process_arr_3:
80.     bge     $t0,                $s5,                        continue_3
81.     add     $t2,                $s1,                        $t0
82.     lb      $t3,                0($t2)
83.     lb      $t4,                0($t1)
84.     add     $t5,                $t3,                        $t4
85.     sb      $t5,                0($s6)
86.     addi    $t0,                $t0,                        1
87.     subi    $t1,                $t1,                        1
88.     addi    $s6,                $s6,                        1
89.     j       loop_process_arr_3
90.
```

```
continue_3:
    jal     print_end_line

prepare_print_arr_3:
    la      $a0,                str_print_arr_3
    li      $v0,                4
    syscall
    jal     print_end_line
    la      $s1,                array3
    lw      $s2,                size3

    li      $t0,                0

loop_print_arr_3:
    bge     $t0,                $s2,                            continue_4
    add     $t1,                $t0,                            $s1
    lb      $a0,                0($t1)
    li      $v0,                1
    syscall
    addi    $t0,                $t0,                            1
    jal     print_tab
    j       loop_print_arr_3

continue_4:
    jal     print_end_line

prompt_user_input_which_arr:
    la      $a0,                str_print_prompt_input
    li      $v0,                4
    syscall
    li      $v0,                5
    syscall
    move    $s0,                $v0

prompt_user_input_which_index:
    la      $a0,                str_print_prompt_input_index
    li      $v0,                4
    syscall
    li      $v0,                5
    syscall
    move    $s1,                $v0

compare_input:
    li      $t0,                1
```

```
135.     beq     $s0,                $t0,                            get_ele_of_arr_1
136.     li      $t0,                2
137.     beq     $s0,                $t0,                            get_ele_of_arr_2
138.     li      $t0,                3
139.     beq     $s0,                $t0,                            get_ele_of_arr_3
140.
141. get_ele_of_arr_1:
142.     la      $s3,                array1
143.     sll     $s1,                $s1,                            2
144.     add     $s4,                $s3,                            $s1
145.     lw      $s5,                0($s4)
146.     j       print_got_ele
147.
148. get_ele_of_arr_2:
149.     la      $s3,                array2
150.     add     $s4,                $s3,                            $s1
151.     lb      $s5,                0($s4)
152.     j       print_got_ele
153.
154. get_ele_of_arr_3:
155.     la      $s3,                array3
156.     add     $s4,                $s3,                            $s1
157.     lb      $s5,                0($s4)
158.     j       print_got_ele
159.
160. print_got_ele:
161.
162.     move    $a0,                $s5
163.     li      $v0,                1
164.     syscall
165.
166. end:
167.     j       exit
168.
169. print_tab:
170.     la      $a0,                tab_char
171.     li      $v0,                4
172.     syscall
173.     jr      $ra
174.
175. print_end_line:
176.     la      $a0,                end_line_char
177.     li      $v0,                4
178.     syscall
```

```
179.     jr      $ra
180.
181. exit:
```

- Chú thích code:

Dòng	Chú thích	Ghi chú
1 → 7	Khai báo và khởi tạo mảng **array1** có kiểu dữ liệu là word; **array2** có kiểu byte và **array3** kiểu space. Kích thước 3 mảng khác nhau.	
8 → 14	Khởi tạo các string cần thiết để in ra.	
21 → 23	In ra thông báo "Xuat mang 1".	
24	In xuống dòng.	
25	Load địa chỉ của **array1** vào **$s1**.	**$s1** lưu địa chỉ **array1**
26	Gán kích thước **array1** vào **$s2**.	**$s2** lưu số phần tử **array1**
28	Gán **$t0** = 0, để hỗ trợ xử lý duyệt mảng.	
31 → 39	So sánh **$t0** và **$s2**, thực hiện việc duyệt qua tất cả các phần tử của mảng và in ra các phần tử.	
41 → 42	In xuống dòng.	
44 → 47	In ra thông báo "Xuat mang 2".	
48	In xuống dòng.	
49	Load địa chỉ của **array2** vào **$s1**.	**$s1** lưu địa chỉ **array2**
50	Gán kích thước **array2** vào **$s2**.	**$s2** lưu số phần tử **array2**
54 → 62	So sánh **$t0** và **$s2**, thực hiện việc duyệt qua tất cả các phần tử của mảng và in ra các phần tử.	
64 → 65	In xuống dòng.	
68	Load địa chỉ **array2** vào **$s1**.	
69 → 70	Gán **size2** vào thanh ghi **$s2** và **$s3**.	
71 → 72	Cộng **$s3** với **$s1**, sau đó trừ đi 1.	Lấy địa chỉ của **array2** cộng với kích thước **array2** rồi gán vào **$s3**, rồi trừ đi 1

73	Load địa chỉ **array3** vào **$s4**.	
74	Gán kích thước **array3** vào **$s5**.	
75	Sao chép địa chỉ của **array3** từ **$s4** sang $s6.	
76	Gán **$t0** = 0.	
77	Lệnh này sao chép giá trị (địa chỉ đã được điều chỉnh của phần tử cuối cùng trong mảng **array2**) từ thanh ghi **$s3** sang thanh ghi **$t1.**	
79 → 80	So sánh **$t0** và **$s5** để lặp.	
81	Lệnh này tính địa chỉ của phần tử hiện tại trong **array2** bằng cách cộng địa chỉ cơ sở của **array2** (**$s1**) với chỉ số hiện tại (**$t0**), và lưu kết quả vào **$t2.**	
82 → 85	Cộng **$t3** và **$t4** rồi gán vào **$t5**, sau đó store vào **$s6**.	Cộng 2 phần tử đầu và cuối của **array2** rồi store vào **array3**
86 → 88	Tăng và giảm chỉ số ở 2 đầu mảng của **array2**.	
89	Jump lại ra **loop_process_arr_3** thực hiện vòng lặp.	
91 → 92	Xuống dòng.	
95 → 97	In ra “Xuat mang 3: ”.	
98 → 100	In xuống dòng. Load địa chỉ **array3** vào **$s1**. Load size3 (kích thước mảng 3) vào **$s2**.	
102	Gán 0 vào **$t0**.	
104 → 113	So sánh **$t0** và **$s2** rồi vào vòng lặp. Thực hiện cộng **$t0** với **$s1** rồi gán vào **$t1**. Load phần tử tại địa chỉ **$t1** vào $a0 rồi in ra. Sau đó **$t1** tăng thêm 1. Jump về **loop_print_arr_3** để tiếp tục việc lặp.	Duyệt mảng và thực hiện việc in ra tất cả phần tử của mảng.
114 → 115	In xuống dòng.	
117 → 120	In ra “Nhap thu tu mang can truy xuat: ”.	
121 → 123	Sao chép chuỗi được nhập vào **$s0**.	

126 → 128	In ra "Nhap index của mảng: ".	
129 → 131	Sao chép chuỗi được nhập vào **$s1**.	
133 → 139	Thực hiện so sánh chuỗi được nhập. Nếu = 1 thì jump đến **get_ele_of_arr_1**, nếu = 2 thì jump đến **get_ele_of_arr_2**, nếu = 3 thì jump đến **get_ele_of_arr_3**.	
141 → 146	Thực hiện việc in ra phần tử tại index được nhập vào đối với trường hợp mảng 1.	
148 →152	Thực hiện việc in ra phần tử tại index được nhập vào đối với trường hợp mảng 2.	
154 →158	Thực hiện việc in ra phần tử tại index được nhập vào đối với trường hợp mảng 3.	
160 → 164	Khối lệnh thực hiện việc in phần tử tại index của 1 mảng xác định.	
169 → 173	Khối lệnh thực hiện in ra kí hiệu '**tab**'.	
175 → 179	Khối lệnh thực hiện in ra kí hiệu xuống dòng.	

- Chương trình chạy mẫu:

```
xuat mang 1:
5       6       7       8       1       2       3       9       10      4
xuat mang 2:
1       2       3       4       5       6       7       8       9       10      11
        12      13      14      15      16


xuat mang 3:
17      17      17      17      17      17      17      17
Nhap thu tu mang can truy xuat: 1
Nhap index cua mang: 3
8
```

```
xuat mang 1:
5       6       7       8       1       2       3       9       10      4
xuat mang 2:
1       2       3       4       5       6       7       8       9       10      11
        12      13      14      15      16
```

```
xuat mang 3:
17      17      17      17      17      17      17      17
Nhap thu tu mang can truy xuat: 2
Nhap index cua mang: 1
2
```

2. **Thao tác với con trỏ**

   (giảm tải)

3. **Bài tập *(chỉ sử dụng con trỏ)***

   3.1. Nhập một mảng các số nguyên n phần tử *(nhập vào số phần tử và giá trị của từng phần tử)*, xuất ra cửa sổ I/O của MARS theo từng yêu cầu sau:

   - Xuất ra giá trị lớn nhất và nhỏ nhất của mảng
   - Tổng tất cả các phần tử của mảng
   - Người sử dụng nhập vào chỉ số của một phần tử nào đó và giá trị của phần tử
   - đó được in ra cửa sổ

```
.data

str_prompt_in_num_of_eles:  .asciiz "Nhap so phan tu cua mang: "
str_prompt_in_ele:          .asciiz "Nhap phan tu: "
str_promp_in_index:         .asciiz "Nhap chi so phan tu: "
str_min:                    .asciiz "Phan tu nho nhat: "
str_max:                    .asciiz "Phan tu lon nhat: "
str_sum:                    .asciiz "Tong cua mang: "
str_index_value:            .asciiz "Phan tu tai index vua nhap: "
end_line_char:              .asciiz "\n"
array:                      .space  100

.text

main:

input_array:
    li      $v0,                4
    la      $a0,                str_prompt_in_num_of_eles
    syscall
    li      $v0,                5
    syscall
    move    $s0,                $v0
```

```
24.     la      $s1,                array
25.
26. read_array:
27.     li      $t0,                0
28.
29. loop_read_array:
30.     bge     $t0,                $s0,                        prepare_min
31.     la      $a0,                str_prompt_in_ele
32.     li      $v0,                4
33.     syscall
34.     li      $v0,                5
35.     syscall
36.     add     $t1,                $s1,                        $t0
37.     sb      $v0,                0($t1)
38.     addi    $t0,                $t0,                        1
39.     j       loop_read_array
40.
41. prepare_min:
42.     li      $t0,                1
43.     lb      $s2,                0($s1)
44.
45. loop_get_min:
46.     bge     $t0,                $s0,                        print_min
47.     add     $t1,                $s1,                        $t0
48.     addi    $t0,                $t0,                        1
49.     lb      $t2,                0($t1)
50.     bge     $t2,                $s2,                        loop_get_min
51.     move    $s2,                $t2
52.     j       loop_get_min
53.
54. print_min:
55.     li      $v0,                4
56.     la      $a0,                str_min
57.     syscall
58.     move    $a0,                $s2
59.     li      $v0,                1
60.     syscall
61.
62.     jal     print_end_line
63.
64. prepare_max:
65.     li      $t0,                1
66.     lb      $s3,                0($s1)
67.
```

```
loop_get_max:
    bge     $t0,            $s0,            print_max
    add     $t1,            $s1,            $t0
    addi    $t0,            $t0,            1
    lb      $t2,            0($t1)
    ble     $t2,            $s3,            loop_get_max
    move    $s3,            $t2
    j       loop_get_max

print_max:
    li      $v0,            4
    la      $a0,            str_max
    syscall
    move    $a0,            $s3
    li      $v0,            1
    syscall

    jal     print_end_line

prepare_sum:
    li      $t0,            0
    li      $s4,            0

loop_get_sum:
    bge     $t0,            $s0,            print_sum
    add     $t1,            $s1,            $t0
    lb      $t2,            0($t1)
    add     $s4,            $s4,            $t2
    addi    $t0,            $t0,            1
    j       loop_get_sum

print_sum:
    li      $v0,            4
    la      $a0,            str_sum
    syscall
    move    $a0,            $s4
    li      $v0,            1
    syscall

    jal     print_end_line

input_index:
    la      $a0,            str_promp_in_index
    li      $v0,            4
```

```
112.     syscall
113.     li      $v0,                5
114.     syscall
115.     move    $t0,                $v0
116.
117. print_index_value:
118.     la      $a0,                str_index_value
119.     li      $v0,                4
120.     syscall
121.     add     $t1,                $t0,                        $s1
122.     lb      $a0,                0($t1)
123.     li      $v0,                1
124.     syscall
125.
126. end:
127.     j       exit
128.
129. print_end_line:
130.     la      $a0,                end_line_char
131.     li      $v0,                4
132.     syscall
133.     jr      $ra
134.
135. exit:
```

- Chú thích code:

Dòng	Chú thích	Ghi chú
3 → 10	Các string cần thiết để xuất ra màn hình.	
11	Khởi tạo **array** với kiểu **space** có kích thước 100 bytes.	
18 → 20	Xuất thông báo yêu cầu nhập số phần tử của **array**.	
21 → 23	Nhập số phần tử của **array** và lưu vào **$s0**.	**$s0** lưu số phần tử.
24	Load địa chỉ của **array** vào **$s1**.	**$s1** lưu địa chỉ array.
27	Gán **$t0** = 0 nhằm hỗ trợ việc lặp xử lý mảng.	
30	So sánh nếu **$t0** ≥ **$s0** thì jump label **prepare_min**.	
31 → 35	Xuất nhập phần tử lần lượt.	

36 → 38	Lưu giá trị vừa nhập vào index tương ứng lần lượt vào **array**.	Nhập giá trị cho các phần tử của **array**.
39	Jump ngược lên dòng 30 để lặp.	
42 → 43	Gán **$t0** = 1 nhằm hỗ trợ việc lặp xử lý mảng. Gán **$s2** là phần tử đầu tiên của mảng.	**$s2** là giá trị của phần tử nhỏ nhất.
46 → 52	Việc lặp để kiểm tra xem có phần tử nào nhỏ hơn giá trị hiện tại của **$s2** không. Nếu có thì gán giá trị đó vào **$s2**.	
55 → 60	Xuất ra giá trị nhỏ nhất của **array**.	Xuất **$s2**.
65 → 66	Gán **$t0** = 1 nhằm hỗ trợ việc lặp xử lý mảng. Gán **$s3** là phần tử đầu tiên của mảng.	**$s3** là giá trị của phần tử lớn nhất của mảng.
69 → 75	Việc lặp để kiểm tra xem có phần tử nào lớn hơn giá trị hiện tại của **$s3** không. Nếu có thì gán giá trị đó vào **$s3**.	
77 → 83	Xuất ra giá trị lớn nhất của **array**.	Xuất **$s3**.
88 → 89	Gán **$t0** = 0 nhằm hỗ trợ việc lặp xử lý mảng. Gán **$s4** = 0.	**$s4** là tổng giá trị các phần tử của **array**.
92 → 97	Lặp cộng dần từng phần tử của **array** vào **$s4**.	
100 → 105	Xuất ra tổng các phần tử của **array**.	Xuất **$s4**.
110 → 115	Xuất và yêu cầu nhập index muốn truy xuất từ **array**. Lưu giá trị index vừa nhập vào **$t0**.	**$t0** là giá trị index muốn truy xuất.
118 → 124	Xuất ra giá trị của phần tử index thứ **$t0**.	Xuất giá trị phần tử tại index **$t0**.

- Chương trình chạy mẫu:

```
Nhap so phan tu cua mang: 5
Nhap phan tu: 7
Nhap phan tu: 2
Nhap phan tu: 5
Nhap phan tu: 4
Nhap phan tu: 9
Phan tu nho nhat: 2
Phan tu lon nhat: 9
```

```
Tong cua mang: 27
Nhap chi so phan tu: 2
Phan tu tai index vua nhap: 5
```

3.2. Nhập một mảng các số nguyên n phần tử *(nhập vào số phần tử và giá trị của từng phần tử)*. Mảng này gọi là A.

Chuyển dòng lệnh C dưới đây sang mã assembly của MIPS. Với các biến nguyên i, j được gán lần lượt vào thanh ghi **$s0**, **$s1**; và địa chỉ nền của mảng số nguyên A được lưu trong thanh ghi **$s3**.

```
1. if (i<j) A[i]= i;
2. else A[i] = j;
```

- Bài code:

```
1. .data
2.
3. str_prompt_in_1:    .asciiz "Nhap so phan tu cua mang: "
4. str_prompt_in_2:    .asciiz "Nhap phan tu: "
5. str_prompt_in_i:    .asciiz "Nhap i: "
6. str_prompt_in_j:    .asciiz "Nhap j: "
7. tab_char:           .asciiz "\t"
8. array:              .space  100
9.
10. .text
11.
12. main:
13.
14. read_number_of_eles_of_arr:
15.     li      $v0,        4
16.     la      $a0,        str_prompt_in_1
17.     syscall
18.     li      $v0,        5
19.     syscall
20.     move    $t0,        $v0
21.     la      $a3,        array
22.     li      $t1,        0
23.
24. read_array:
25.     bge     $t1,        $t0,                input_i_j
26.     li      $v0,        4
27.     la      $a0,        str_prompt_in_2
```

```
    syscall
    li      $v0,        5
    syscall
    add     $t5,        $a3,                $t1
    sb      $v0,        0($t5)
    addi    $t1,        $t1,                1
    j       read_array

input_i_j:
    la      $s3,        array
    li      $v0,        4
    la      $a0,        str_prompt_in_i
    syscall
    li      $v0,        5
    syscall
    move    $s0,        $v0
    li      $v0,        4
    la      $a0,        str_prompt_in_j
    syscall
    li      $v0,        5
    syscall
    move    $s1,        $v0

compare_i_j:
    add     $t3,        $s3,                $s0
    blt     $s0,        $s1,                asign_i
    j       asign_j

asign_i:
    sb      $s0,        0($t3)
    j       continue_2

asign_j:
    sb      $s1,        0($t3)

continue_2:
    la      $a3,        array
    li      $t1,        0

print:
    bge     $t1,        $t0,                end
    add     $t5,        $a3,                $t1
    lb      $a0,        0($t5)
    li      $v0,        1
```

```
72.     syscall
73.     jal     print_tab
74.     addi    $t1,        $t1,            1
75.     j       print
76.
77. end:
78.     j       exit
79.
80. print_tab:
81.     la      $a0,        tab_char
82.     li      $v0,        4
83.     syscall
84.     jr      $ra
85.
86. exit:
```

- Chú thích code:

Dòng	Chú thích	Ghi chú
3 → 7	Các string cần thiết để xuất ra màn hình.	
8	Khởi tạo **array** với kiểu **space** có kích thước 100 bytes.	
15 → 17	Xuất thông báo yêu cầu nhập số phần tử của **array**.	
18 → 20	Nhập số phần tử của **array** và lưu vào **$t0**.	**$t0** lưu số phần tử.
21	Load địa chỉ của **array** vào **$a3**.	**$a3** lưu địa chỉ array.
22	Gán **$t1** = 0 nhằm hỗ trợ việc lặp xử lý mảng.	
25 → 34	Xuất và yêu cầu nhập giá trị cho từng phần tử vào **array**.	
37 → 43	Nhập i và lưu vào **$s0**.	**$s0** là i.
44 → 49	Nhập j và lưu vào **$s1**.	**$s1** là j.
52	Lưu địa chỉ của array index thứ i vào **$t3**	**$t3** là vị tí cần thay thế
53	So sánh i và j. Nếu i < j thì **array[i]** = i tại label **asign_i**, ngược lại **array[i]** = j tại label **asign_j**.	Gán **array[i]** theo yêu cầu của đề bài.
64 → 65	Lưu địa chỉ của **array** vào **$a3**. Gán **$t1** = 0 để hỗ trợ lặp xuất **array**.	

68 → 75	Lặp xuất **array**.	

- Chương trình chạy mẫu:

  - TH i < j:

```
Nhap so phan tu cua mang: 5
Nhap phan tu: 3
Nhap phan tu: 4
Nhap phan tu: 6
Nhap phan tu: 1
Nhap phan tu: 3
Nhap i: 1
Nhap j: 2
3       1       6       1       3
```

  - TH i > j:

```
Nhap so phan tu cua mang: 5
Nhap phan tu: 3
Nhap phan tu: 4
Nhap phan tu: 6
Nhap phan tu: 1
Nhap phan tu: 3
Nhap i: 4
Nhap j: 2
3       4       6       1       2
```