BẢN BÁO CÁO

Phần 2: Thầy tải file mô phỏng của em để coi dễ hơn ạ: Lap3 phần 2

Đoạn code array1

Để có thể truy xuất phần tử trong mảng array1 thì em gán địa chỉ cho $s1 bằng lệnh **la $s1, array1**.

Đoạn code này để em in ra màn hình đoạn text_1

```
19          li $v0, 4
20          la $a0, text_1
```

Để sau mỗi vòng lặp em có thể in ra giá trị kế tiếp của mảng thì em tạo ra một thanh ghi lưu địa chỉ của $s1 đang trỏ đến tức là của array1

```
22          add $t0, $s1, $zero
```

Rồi sau đó em lấy địa chỉ kế tiếp địa chỉ cuối cùng của mảng để làm điều kiện cho việc $t0 đã đi qua hết phần tử array1 hay chưa (số 40 là do em lười lấy giá trị của size1 (10) để dịch trái 2 bit nên em cộng thẳng luôn)

```
23          addi $t4, $s1, 40
```

Tiếp đến để lấy giá trị của phần tử mà $t0 đang trỏ đến thì em sử dụng lệnh lw và lưu vào $a0 để tiện cho việc in ra màn I/O

```
25          lw $a0, 0($t0)
```

Rồi em tăng giá trị của thanh ghi $t0 lên 4 word để trỏ tới phần tử tiếp theo nằm trong mảng

```
26          addi $t0, $t0, 4
```

Rồi em in ra màn hình giá trị vừa lấy được trong mảng

```
27          li $v0, 1
28          syscall
```

Đoạn lệnh này để chương trình in ra khoảng trắng giữa các số

```
29          li $v0, 4
30          la $a0, space
31          syscall
```

Và cuối cùng là điều kiện thoát khỏi vòng lặp. Nếu $t0 chạy tới địa chỉ kế tiếp địa chỉ phần tử cuối cùng trong mảng ($t0 == $t4) thì lệnh bne sẽ không nhảy tới label array1_loop nữa

```
32          bne $t0, $t4, array1_loop
```

Đoạn code cho phần array2 thì cơ bản cũng giống ý tưởng với cách em làm ở array1 nên em sẽ đi nhanh xíu:

Lấy địa chỉ array2 bằng lệnh **la $s2, array2**

Tạo thanh ghi có thể trỏ đến phần tử sau trong mảng

```
38          add $t0, $s2, $zero
```

Tạo thanh ghi là điều kiện để thoát khỏi loop. Ở đây em cộng cho 16 là do array2 có 16 byte, mà mỗi ô nhớ chứa 4 byte nên array2 sẽ cần 4 ô nhớ để chứa dữ liệu suy ra cần 4x4 = 16 word để đi hết tới ô nhớ kế tiếp phần tử cuối array2

```
39          addi $t4, $s2, 16
```

Lấy giá trị của phần tử xuất ra I/O. Do mỗi phần tử array2 chỉ lưu dưới dạng byte nên em dùng lệnh lb thay vì lệnh lw

```
41          lb $a0, 0($t0)
```

Tăng giá trị $t0 lên 1 byte để sang phần tử tiếp theo

```
42          addi $t0, $t0, 1
```

Xuất ra màn hình I/O số và khoảng cách đằng sau

```
43          li $v0, 1
44          syscall
45          li $v0, 4
46          la $a0, space
47          syscall
```

Và cuối cùng là lệnh bne để tạo vòng lặp khi điều kiện $t0 == $t4 chưa thỏa

```
48          bne $t0, $t4, array2_loop
```

Đoạn code ở array3:

Lấy địa chỉ ở array3 bằng lệnh **la $s3, array3**

Do array3 và array2 có 2 địa chỉ khác nhau nên biến i sẽ có 2 biến khác nhau tạm gọi là i1 và i2

Sau đó em khởi tạo giá trị cho biến i1 ($t0) cho array2

```
51          add $t0, $s2, $zero
```

Và khởi tạo biến i2 ($t3) cho array3

```
52          add $t3, $s3, $zero
```

Rồi tiếp đó là cụm giá trị size2 – 1 – i thì em sẽ lấy giá trị của size2 gán vào $t5 rồi cho $t5 = $t5 – 1

```
54          addi $t5, $s2, 16
55          addi $t5, $t5, -1
```

Và cho $t5 = $t5 – 1 mỗi khi lặp lại vòng lặp

```
61          addi $t5, $t5, -1
```

Tạo điều kiện thoát vòng lặp. Do array3 có space = 8 tức là có 8 byte dữ liệu nên sẽ được chương trình cấp cho 2 ô nhớ, mỗi ô 4 byte từ đó suy ra khi i chạy đến ô nhớ kế tiếp ô nhớ phần tử cuối cùng của mảng array3 thì sẽ cần 2x4 = 8 word

```
53          addi $t4, $s3, 8
```

Lấy giá trị của array[i]. Do là byte nên em sử dụng lb thay cho lw

```
57          lb $t1, 0($t0)
```

Lấy giá trị của array[size2 – 1 – i]

```
58          lb $t2, 0($t5)
```

Rồi cộng 2 giá trị và lưu vô $t1

```
59          add $t1, $t2, $t1
```

Và cập nhật giá trị cho array3[i]. Do là byte nên em sử dụng sb thay cho sw

```
60          sb $t1, 0($t3)
```

Tăng giá trị i1 và i2

```
61          addi $t3, $t3, 1
62          addi $t0, $t0, 1
```

So sánh điều kiện để thoát vòng lặp.

```
64          bne $t3, $t4 array3_loop
```

Đoạn code để xuất ra theo yêu cầu người dùng

In ra màn hình I/O text_4 và lấy giá trị của số mảng thứ mấy cần lấy từ input

```
66          la $a0, text_4
67          syscall
68          li $v0, 5
69          syscall
```

Lưu số đó vào $t5

```
70          add $t5, $v0, $zero
```

So sánh xem số thứ tự đó có nhỏ hơn 4 không

```
71          slti $t6, $t5, 4
72          beq $t6, $zero, error
```

Nếu không thì hiện bảng Invalid Type và kết thúc chương trình

```
85  error:
86          li $v0, 55
87          la $a0, text_7
88          li $a1, 0
89          syscall
90          j exit
```

Nếu thỏa mãn thì em sẽ in ra màn hình I/O text_5 và lấy chỉ số phần tử cần lấy giá trị tử input và lưu vào $t4

```
78          add $t4, $v0, $zero
```

Em tìm array cần lấy giá trị bằng cách lấy giá trị của số mảng thứ mấy trừ cho 1. Nếu bằng 0 thì sẽ nhảy đến label tương ứng, nếu không thì lấy kết quả đó trừ tiếp cho 1 và so sánh

```
79          addi $t5, $t5, -1
80          beq $t5, $zero, Xuat_array1
81          addi $t5, $t5, -1
82          beq $t5, $zero, Xuat_array2
83          addi $t5, $t5, -1
84          beq $t5, $zero, Xuat_array3
```

Đoạn code in ra màn hình

Để biết được chỉ số phần tử cần xuất ra có hợp lệ hay không thì em đã so sánh xem nó có nhỏ hơn size của từng array tương ứng hay không. Nếu không thì hiện Invalid Type và kết thúc chương trình. Array1 có dữ liệu là word nên em xài lw còn array2 và array3 có dữ liệu là byte nên em xài lb

```
Xuat_array1:
        lw $t1, size1
        slt $t5, $t1, $t4
        bne $t5, $zero, error
        j exit
Xuat_array2:
        lb $t2, size2
        slt $t5, $t2, $t4
        bne $t5, $zero, error
Xuat_array3:
        lb $t3, size3
        slt $t5, $t3, $t4
        bne $t5, $zero, error
```

Nếu ok thì em sẽ lấy chỉ số phần tử mảng đó trừ đi 1 (do mảng bắt đầu từ số 0 nên sẽ bị hụt số), lấy giá trị phần tử, xuất ra màn hình I/O text_6 cùng với giá trị phần tử đó và j exit (riêng array3 không cần vì sau nó là label exit)

Array1:

```
 95          li $v0, 4
 96          la $a0, text_6
 97          syscall
 98          addi $t4, $t4, -1
 99          sll $t4, $t4, 2
100          add  $t0, $s1, $t4
101          lw $a0, 0($t0)
102          li $v0, 1
103          syscall
104          j exit
```

Array2:

```
109          li $v0, 4
110          la $a0, text_6
111          syscall
112          addi $t4, $t4, -1
113          add  $t0, $s2, $t4
114          lb $a0, 0($t0)
115          li $v0, 1
116          syscall
117          j exit
```

Array3:

```
122             li $v0, 4
123             la $a0, text_6
124             syscall
125             addi $t4, $t4, -1
126             add  $t0, $s3, $t4
127             lb $a0, 0($t0)
128             li $v0, 1
129             syscall
130     exit:
```

Phần 3:

a) File mô phỏng: Lap3 phần 3a

Đầu tiên là em khai báo một vài biến và text để làm theo yêu cầu để bài

```
1           .data
2   text_1: .asciiz "Nhap so phan tu mang: "
3   text_2: .asciiz "Nhap gia tri tung phan tu mang:\n"
4   text_3: .asciiz "So nho nhat la: "
5   text_4: .asciiz "\nSo lon nhat la: "
6   text_5: .asciiz "\nTong gia tri cac phan tu trong mang: "
7   text_6: .asciiz "\nChi so phan tu can lay gia tri: "
8   text_7: .asciiz "Gia tri cua phan tu la: "
9   sum:    .word 0
10  min:    .word 0
11  max:    .word 0
12  n:      .word 0
13  array:  .word 0
14          .text
```

Do mảng array có số phần tử (n) chỉ xác nhận được khi người dùng nhập từ input nên em mới khai báo array ở cuối cùng để nó không chèn lên ô nhớ của những biến khác

Đoạn code nhập mảng:

Em sẽ gán địa chỉ của array và n lần lượt cho thanh ghi $s0 và $t0 bằng 2 lệnh sau

```
16          la $s0, array
17          la $t0, n
```

In text_1 ra ngoài màn hình I/O

```
18          la $a0, text_1
19          li $v0, 4
20          syscall
```

Rồi em lấy giá trị từ input

```
21          li $v0, 5
22          syscall
```

Lưu cho n

```
23          sw $v0, 0($t0)
```

Rồi em lấy số phần tử đó nhân cho 4 và cộng với địa chỉ của array để khi thanh ghi $s0 đi qua phần tử cuối cùng của mảng array thì chương trình thoát vòng lặp

```
24          sll $t0, $v0, 2
25          add $t0, $t0, $s0
```

In text_2 ra ngoài màn hình I/O

```
26          la $a0, text_2
27          li $v0, 4
28          syscall
```

Em lấy dữ liệu từ input khi đang còn trong vòng lặp bằng đoạn code sau

```
30          li $v0, 5
31          syscall
```

Lưu giá trị đó xuống mảng array

```
32          sw $v0, 0($s0)
```

Và nhảy sang phân tử kế tiếp trong array

```
33          addi $s0, $s0, 4
```

Để tạo vòng lặp thì em xài lệnh bne với điều kiện khi $s0 chạm tới ô nhớ kế tiếp ô nhớ của phần tử cuối trong array thì sẽ thoát vòng lặp

```
34          bne $s0, $t0, loop
```

Đoạn code tìm min max

Ban đầu em sẽ gán địa chỉ max, min, array vào lần lượt các thanh ghi $s3, $s2, $s0

```
36          la $s2, min
37          la $s3, max
38          la $s0, array
```

Rồi lấy giá trị của phần tử đầu mảng array gán cho max và min

```
39          lw $t1, 0($s0)
40          sw $t1, 0($s2)
41          sw $t1, 0($s3)
```

Khi vào vòng lặp thì em sẽ lấy giá trị của phần tử array, max, min bằng đoạn code sau

```
43          lw $t1, 0($s0)
44          lw $t2, 0($s2)
45          lw $t3, 0($s3)
```

Chuyển $s0 sang ô nhớ kế tiếp trong mảng

```
46          addi $s0, $s0, 4
```

Rồi em sẽ so sánh xem min có bé hơn giá trị phần tử trong mảng hay không. Nếu có thì $t4 sẽ bằng 0 và beq sẽ nhảy tới max_label. Nếu không thì em sẽ lưu giá trị của phần tử đó xuống min

```
47          slt $t4, $t1, $t2
48          beq $t4, $zero, max_label
49          sw $t1, 0($s2)
```

Để lấy được giá trị max thì em sẽ so sánh xem max hiện tại có bé hơn giá trị của phần tử array đã lấy hay không. Nếu có thì em sẽ lưu nó xuống max. Còn không thì lệnh beq sẽ nhảy đến label ss bỏ qua lệnh lưu

```
50  max_label:
51          slt $t4, $t3, $t1
52          beq $t4, $zero, ss
53          sw $t1, 0($s3)
```

Để tạo vòng lặp thì em sẽ xét xem $s0 đã chạy qua ô nhớ phần tử cuối trong mảng hay không ($t0 == $s0). Nếu không thì sẽ lặp lại để xét min max tiếp. Nếu có thì thoát khỏi vòng lặp

```
54  ss:
55          bne $t0, $s0, min_max
```

Đoạn code tìm tổng các phần tử trong mảng:

Đầu tiên em sẽ gán địa chỉ của sum và array lần lượt vào $s4 và $s0. Rồi sau đó em sẽ lấy giá trị của sum (thật ra bước này em không cần cũng được bởi vì lúc đầu sum == 0)

```
57          la $s4, sum
58          la $s0, array
59          lw $t5, 0($s4)
```

Sau đó em sẽ lấy giá trị phần tử trong mảng

```
61          lw $t6, 0($s0)
```

Rồi đưa $s0 đến ô nhớ kế tiếp

```
62          addi $s0, $s0, 4
```

Cộng giá trị phần tử vào $t5 (giá trị sum)

```
63          add $t5, $t5, $t6
```

Rồi em so sánh xem $s0 đã đi hết array hay chưa. Nếu chưa thì sẽ lặp lại để tính sum

```
64          bne $s0, $t0, sum_loop
```

Nếu rồi thì sẽ thoát vòng lặp và lưu $t5 vào sum

```
65          sw $t5, 0($s4)
```

Đoạn code print

Em sẽ cho in ra màn hình I/O text_3 trước

```
67          la $a0, text_3
68          li $v0, 4
69          syscall
```

Rồi lấy giá trị min và in ra màn hình

```
70          la $t2, min
71          lw $a0, 0($t2)
72          li $v0, 1
73          syscall
```

Tiếp đó em in ra màn hình text_4

```
74          la $a0, text_4
75          li $v0, 4
76          syscall
```

Rồi lấy giá trị max và in ra màn hình

```
77          la $t3, max
78          lw $a0, 0($t3)
79          li $v0, 1
80          syscall
```

Sau đó em in ra màn hình text_5

```
81          la $a0, text_5
82          li $v0, 4
83          syscall
```

Và lấy giá trị sum rồi in ra màn hình

```
84          la $t5, sum
85          lw $a0, 0($t5)
86          li $v0, 1
87          syscall
```

Để hoàn thành yêu câu thứ ba của đề bài thì em sẽ in ra màn hình text_6

```
88          la $a0, text_6
89          li $v0, 4
90          syscall
```

Rồi lấy giá trị từ input

```
91          li $v0, 5
92          syscall
```

Em sẽ kiểm tra xem chỉ số mà người dùng nhập vào có hợp lệ hay không bằng cách so sánh xem nó có bé hơn n phần tử ($t0) hay không. Nếu không thì beq sẽ nhảy sang label exit

```
94          slt $t7, $v0, $t0
95          beq $t7, $zero, exit
```

Nếu có thì em sẽ lấy giá trị đó trừ đi 1 do phần tử mảng bắt đầu từ số 0 nên sẽ bị hụt số, dịch trái 2 bit (aka nhân 4) và cộng vô $s0 (đang trỏ đến phần tử đầu tiên của mảng) rồi lưu vào $t7 để $t7 trỏ đến phần tử đang cần lấy giá trị

```
96          addi $v0, $v0, -1
97          sll $t7, $v0, 2
98          add $t7, $s0, $t7
```

Em in text_7 ra màn hình I/O

```
 99          li $v0, 4
100          la $a0, text_7
101          syscall
```

Rồi lấy giá trị phần tử trong mảng mà mình đang cần lấy xong in ra màn hình và kết thúc chương trình

```
102          lw $a0, 0($t7)
103          li $v0, 1
104          syscall
105  exit:
```

b) File mô phỏng: Lab3 phần 3b

Đầu tiên em khai báo các text và biến. Và do array_A chưa biết số phần tử nên em sẽ khai báo cuối cùng

```
1                .data
2    text_1:     .asciiz "Nhap so phan tu mang: "
3    text_2:     .asciiz "Nhap gia tri tung phan tu mang:\n"
4    i_:         .word 0
5    j_:         .word 0
6    n:          .word 0
7    array_A:    .word 0
8                .text
```

Tiếp đến em sẽ gán địa chỉ array_A và n lần lượt vào $s3 và $t0

```
10           la $s3, array_A
11           la $t0, n
```

Rồi em in text_1 ra màn hình I/O và lấy dữ liệu người dùng nhập vào

```
12           la $a0, text_1
13           li $v0, 4
14           syscall
15           li $v0, 5
16           syscall
```

Lưu giá trị đó xuống n

```
17              sw $v0, 0($t0)
```

Do đã lưu được giá trị của n nên em sẽ tận dụng $t0 thành thanh ghi điều kiện so sánh để thoát vòng lặp

```
18          sll $t0, $v0, 2
19          add $t0, $t0, $s3
```

Tạo thanh ghi chạy ô nhớ cho array_A

```
20              add $t3, $s3, $zero
```

In ra màn hình I/O text_2

```
21          la $a0, text_2
22          li $v0, 4
23          syscall
```

Em sẽ lưu giá trị nhập vào của người dùng bằng đoạn code sau

```
24  loop:
25          li $v0, 5
26          syscall
```

Lưu giá trị đó xuống array_A

```
27              sw $v0, 0($t3)
```

Rồi em cho $t3 sang ô nhớ kế tiếp trong mảng A

```
28            addi $t3, $t3, 4
```

Rồi em so sánh xem $t3 đã đi qua phần tử cuối cùng của array_A chưa. Nếu chưa thì lặp lại. Nếu rồi thì sẽ thoát vòng lặp

```
29          bne $t3, $t0, loop
```

Đoạn code chuyển từ C sang

Ban đầu em sẽ gán địa chỉ của biến i và j vào lần lượt $s0 và $s1, sau đó em sẽ lấy giá trị của hai biến i và j lưu vào lần lượt $t0 và $t1

```
31          la $s0, i_
32          la $s1, j_
33          lw $t0, 0($s0)
34          lw $t1, 0($s1)
```

Để mô phỏng if (i<j) A[i] == i ; thì em sẽ lấy $t0 (i) so sánh với $t1 (j) xem có bé hơn không

```
35          slt $t4, $t0, $t1
```

Nếu lớn hơn hoặc bằng thì nhảy đến label else

```
36          beq $t4, $zero, else
```

Nếu bé hơn thì sẽ lấy giá trị của i

```
37          add $t3, $t0, $zero
```

Dịch trái 2 bit và cộng với $s3 để $s3 trỏ đến phần tử thứ i trong mảng A

```
38          sll $t3, $t3, 2
39          add $t3, $s3, $t3
```

Lưu i vào A[i] và kết thúc chương trình

```
40          sw $t0, 0($t3)
41          j exit
```

Để chuyển lệnh else A[i] == j; thì lệnh else đã được mô phỏng bằng lệnh

```
36          beq $t4, $zero, else
```

Vậy là còn mỗi A[i] == j thì em sẽ lấy giá trị của i. Sau đó dịch trái 2 bit và cộng với $s3 trỏ đến phần tử thứ i trong mảng A

```
42  else:
43          add $t3, $t0, $zero
44          sll $t3, $t3, 2
45          add $t3, $s3, $t3
```

Và lưu j vào A[i] rồi kết thúc chương trình

```
46          sw $t1, 0($t3)
47  exit:
```