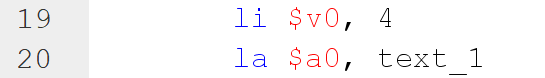
**BẢN BÁO CÁO**

**Phần 2:** Thầy tải file mô phỏng của em để coi dễ hơn ạ: [Lap3 phần 2](https://github.com/kamuisi/KTMT/blob/main/lap3_1.asm)

**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



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



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)



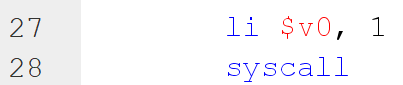
Tiếp đến để lấy giá trị của phần tử mà $t0 đang trỏ đến thì em sử dụng lệnh lw và lưu vào $a0 để tiện cho việc in ra màn I/O



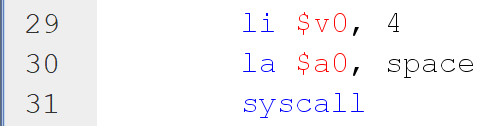
Rồi em tăng giá trị của thanh ghi $t0 lên 4 word để trỏ tới phần tử tiếp theo nằm trong mảng



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



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



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



**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



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



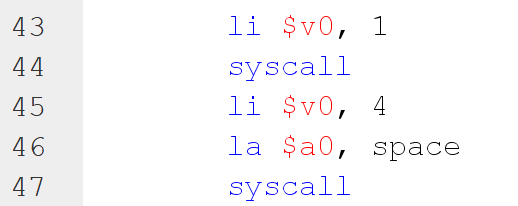
Lấy giá trị của phần tử xuất ra I/O. Do mỗi phần tử array2 chỉ lưu dưới dạng byte nên em dùng lệnh lb thay vì lệnh lw



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



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



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



**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

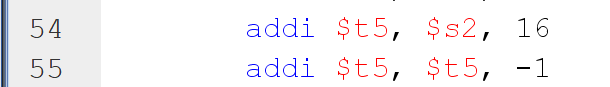
Sau đó em khởi tạo giá trị cho biến i1 ($t0) cho array2



Và khởi tạo biến i2 ($t3) cho array3



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



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



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



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



Lấy giá trị của array[size2 – 1 – i]



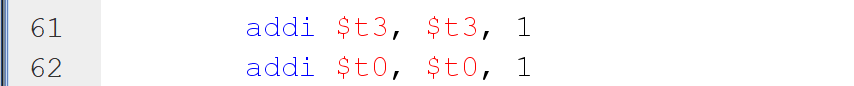
Rồi cộng 2 giá trị và lưu vô $t1



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



Tăng giá trị i1 và i2

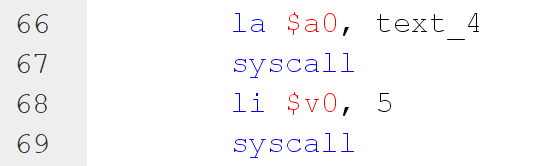


So sánh điều kiện để thoát vòng lặp.



**Đoạn code để xuất ra theo yêu cầu người dùng**

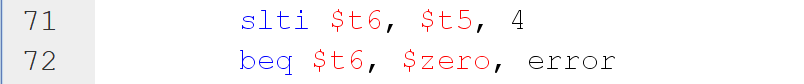
In ra màn hình I/O text\_4 và lấy giá trị của số mảng thứ mấy cần lấy từ input



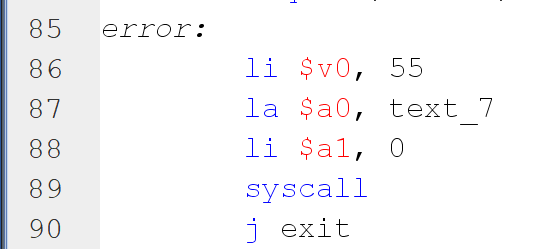
Lưu số đó vào $t5



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



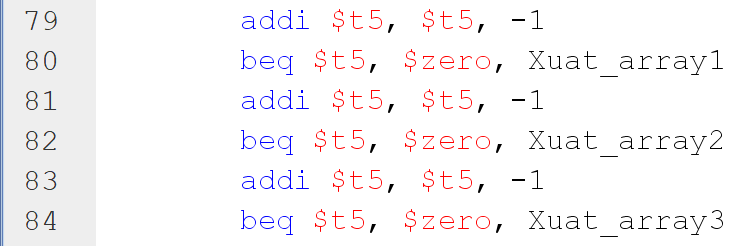
Nếu không thì hiện bảng Invalid Type và kết thúc chương trình



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

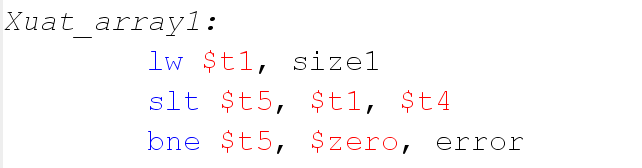


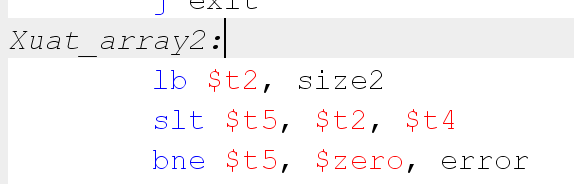
Em tìm array cần lấy giá trị bằng cách lấy giá trị của số mảng thứ mấy trừ cho 1. Nếu bằng 0 thì sẽ nhảy đến label tương ứng, nếu không thì lấy kết quả đó trừ tiếp cho 1 và so sánh

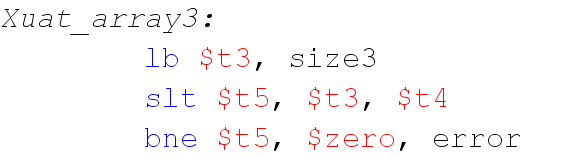


**Đoạn code in ra màn hình**

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

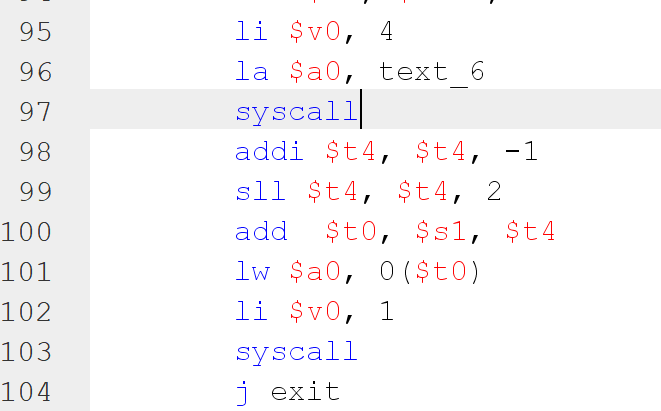




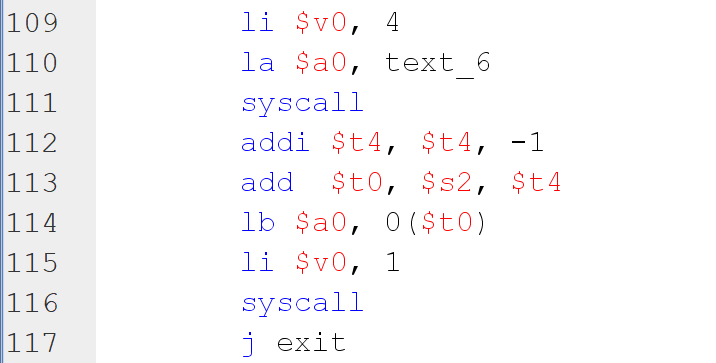


Nếu ok thì em sẽ lấy chỉ số phần tử mảng đó trừ đi 1 (do mảng bắt đầu từ số 0 nên sẽ bị hụt số), lấy giá trị phần tử, xuất ra màn hình I/O text\_6 cùng với giá trị phần tử đó và j exit (riêng array3 không cần vì sau nó là label exit)

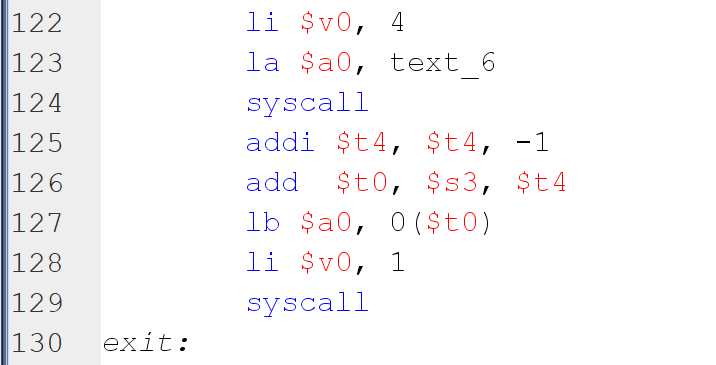
Array1:



Array2:



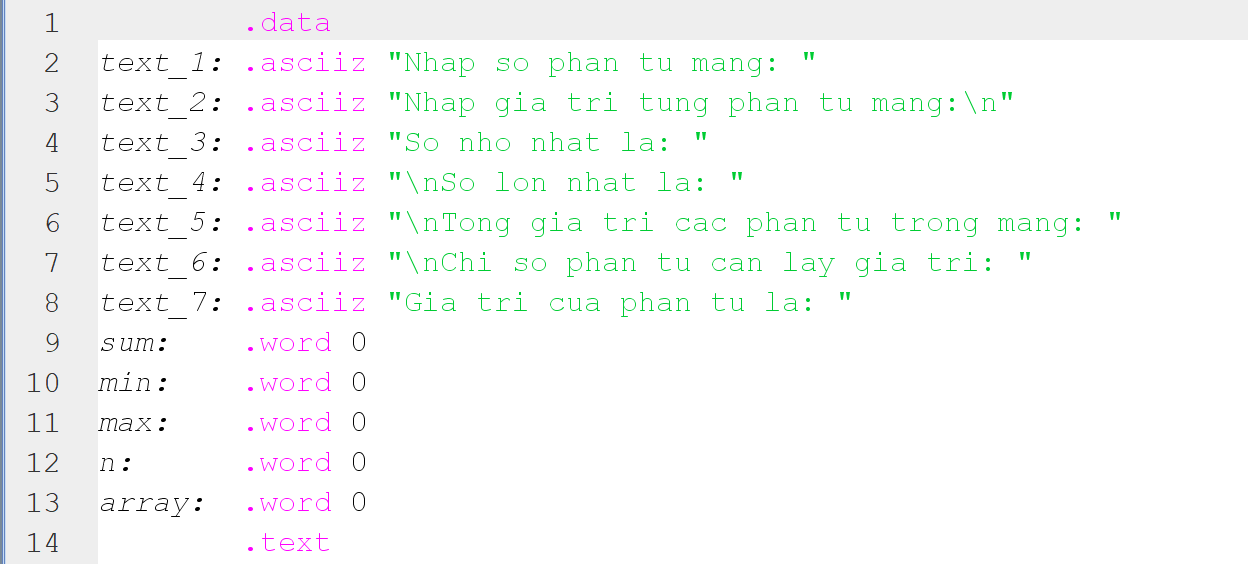
Array3:



**Phần 3:**

1. File mô phỏng: [Lap3 phần 3a](https://github.com/kamuisi/KTMT/blob/main/lab3_3a.asm)

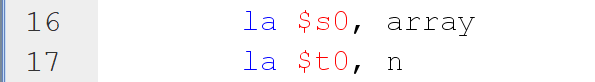
Đầu tiên là em khai báo một vài biến và text để làm theo yêu cầu để bài



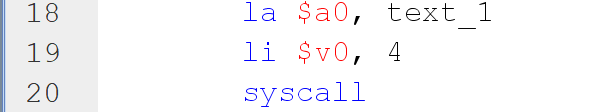
Do mảng array có số phần tử (n) chỉ xác nhận được khi người dùng nhập từ input nên em mới khai báo array ở cuối cùng để nó không chèn lên ô nhớ của những biến khác

**Đoạn code nhập mảng:**

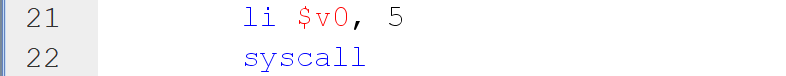
Em sẽ gán địa chỉ của array và n lần lượt cho thanh ghi $s0 và $t0 bằng 2 lệnh sau



In text\_1 ra ngoài màn hình I/O



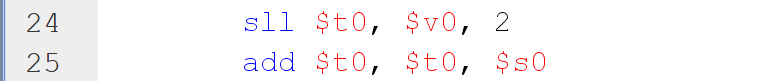
Rồi em lấy giá trị từ input



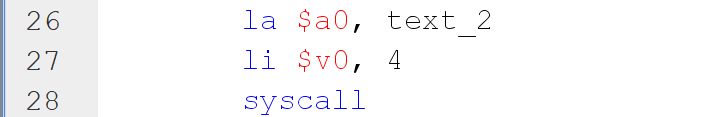
Lưu cho n



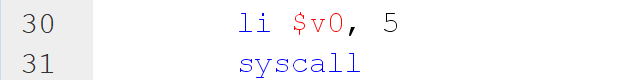
Rồi em lấy số phần tử đó nhân cho 4 và cộng với địa chỉ của array để khi thanh ghi $s0 đi qua phần tử cuối cùng của mảng array thì chương trình thoát vòng lặp



In text\_2 ra ngoài màn hình I/O



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



Lưu giá trị đó xuống mảng array



Và nhảy sang phân tử kế tiếp trong array

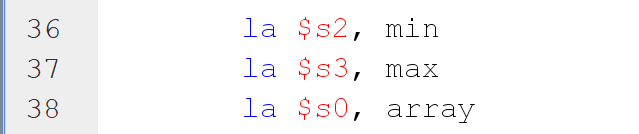


Để tạo vòng lặp thì em xài lệnh bne với điều kiện khi $s0 chạm tới ô nhớ kế tiếp ô nhớ của phần tử cuối trong array thì sẽ thoát vòng lặp

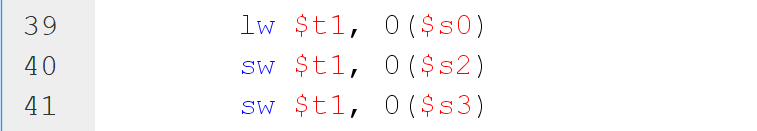


**Đoạn code tìm min max**

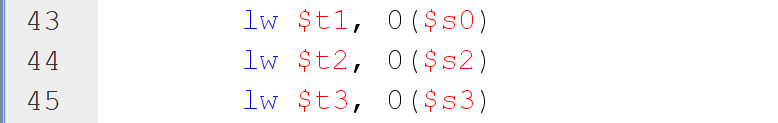
Ban đầu em sẽ gán địa chỉ max, min, array vào lần lượt các thanh ghi $s3, $s2, $s0



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



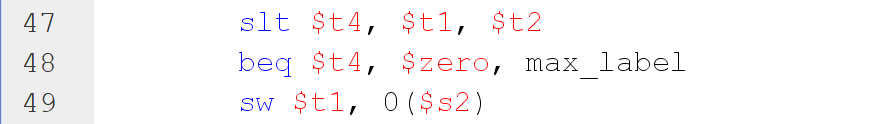
Khi vào vòng lặp thì em sẽ lấy giá trị của phần tử array, max, min bằng đoạn code sau



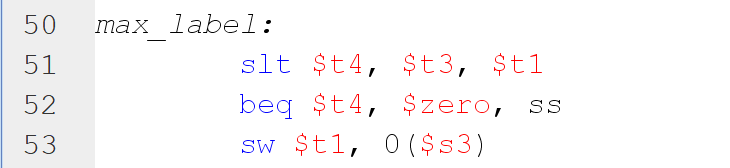
Chuyển $s0 sang ô nhớ kế tiếp trong mảng



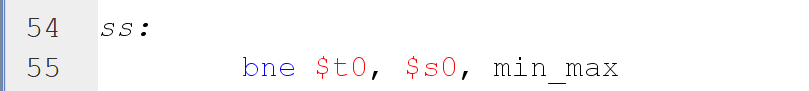
Rồi em sẽ so sánh xem min có bé hơn giá trị phần tử trong mảng hay không. Nếu có thì $t4 sẽ bằng 0 và beq sẽ nhảy tới max\_label. Nếu không thì em sẽ lưu giá trị của phần tử đó xuống min



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

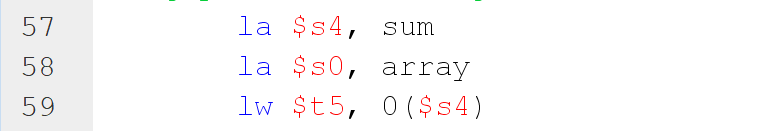


Để tạo vòng lặp thì em sẽ xét xem $s0 đã chạy qua ô nhớ phần tử cuối trong mảng hay không ($t0 == $s0). Nếu không thì sẽ lặp lại để xét min max tiếp. Nếu có thì thoát khỏi vòng lặp



**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)



Sau đó em sẽ lấy giá trị phần tử trong mảng



Rồi đưa $s0 đến ô nhớ kế tiếp



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



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

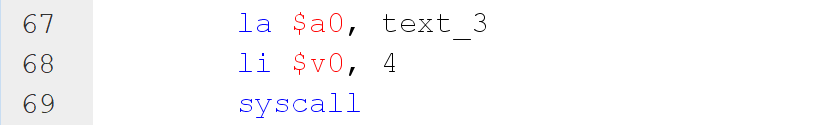


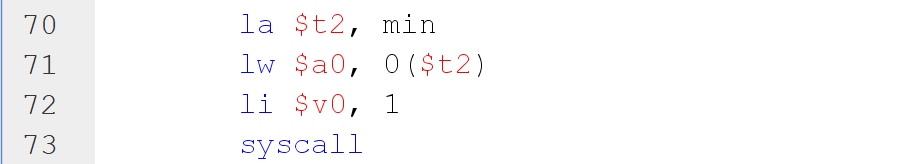
Nếu rồi thì sẽ thoát vòng lặp và lưu $t5 vào sum



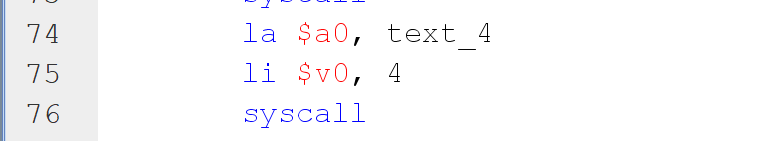
**Đoạn code print**

Em sẽ cho in ra màn hình I/O text\_3 trước

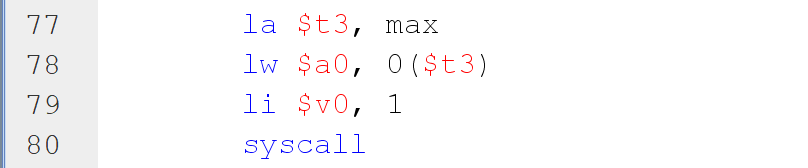
****Rồi lấy giá trị min và in ra màn hình



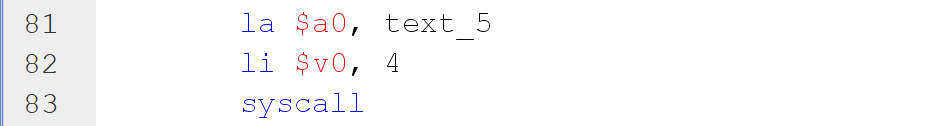
Tiếp đó em in ra màn hình text\_4



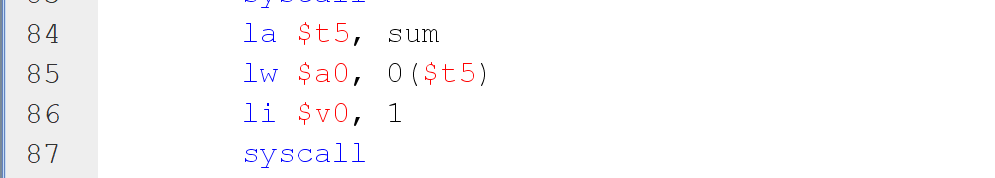
Rồi lấy giá trị max và in ra màn hình



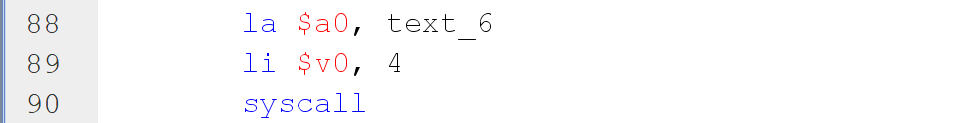
Sau đó em in ra màn hình text\_5



Và lấy giá trị sum rồi in ra màn hình



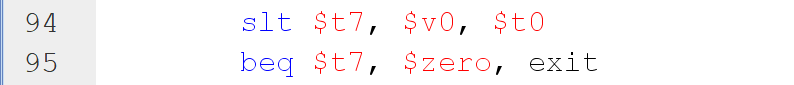
Để hoàn thành yêu câu thứ ba của đề bài thì em sẽ in ra màn hình text\_6



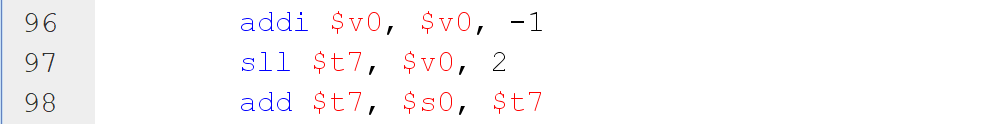
Rồi lấy giá trị từ input



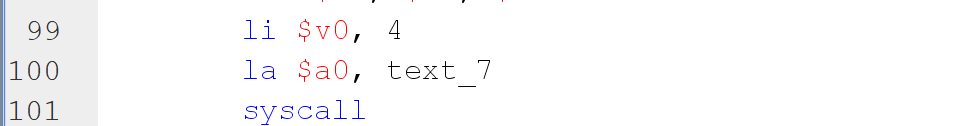
Em sẽ kiểm tra xem chỉ số mà người dùng nhập vào có hợp lệ hay không bằng cách so sánh xem nó có bé hơn n phần tử ($t0) hay không. Nếu không thì beq sẽ nhảy sang label exit



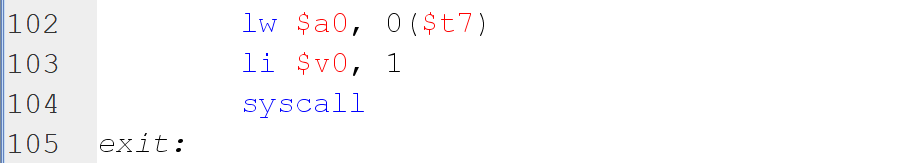
Nếu có thì em sẽ lấy giá trị đó trừ đi 1 do phần tử mảng bắt đầu từ số 0 nên sẽ bị hụt số, dịch trái 2 bit (aka nhân 4) và cộng vô $s0 ( đang trỏ đến phần tử đầu tiên của mảng) rồi lưu vào $t7 để $t7 trỏ đến phần tử đang cần lấy giá trị



Em in text\_7 ra màn hình I/O

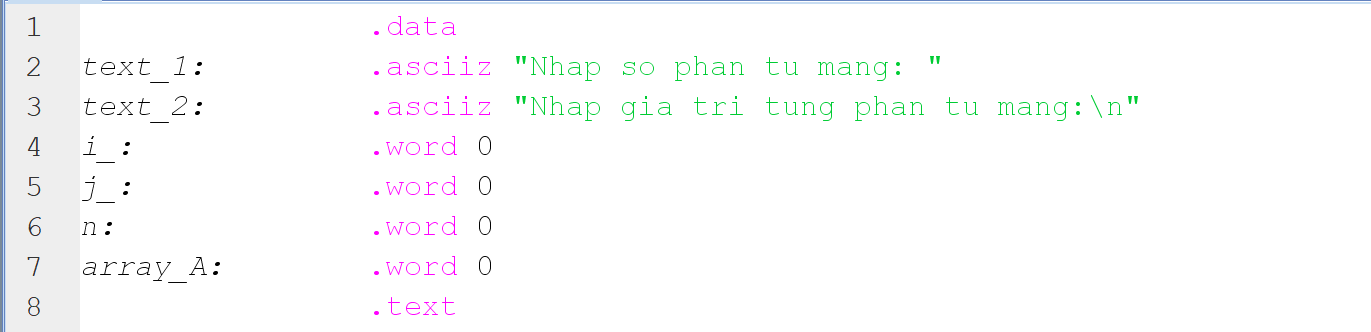


Rồi lấy giá trị phần tử trong mảng mà mình đang cần lấy xong in ra màn hình và kết thúc chương trình

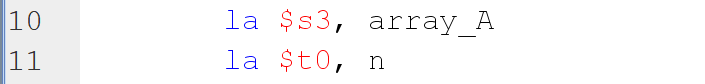


1. File mô phỏng: [Lab3 phần 3b](https://github.com/kamuisi/KTMT/blob/main/lap3_3b.asm)

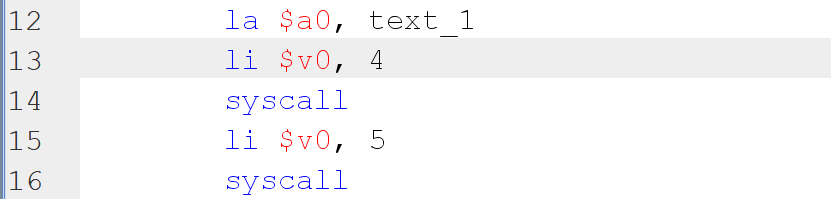
Đầu tiên em khai báo các text và biến. Và do array\_A chưa biết số phần tử nên em sẽ khai báo cuối cùng

****

Tiếp đến em sẽ gán địa chỉ array\_A và n lần lượt vào $s3 và $t0



Rồi em in text\_1 ra màn hình I/O và lấy dữ liệu người dùng nhập vào



Lưu giá trị đó xuống n



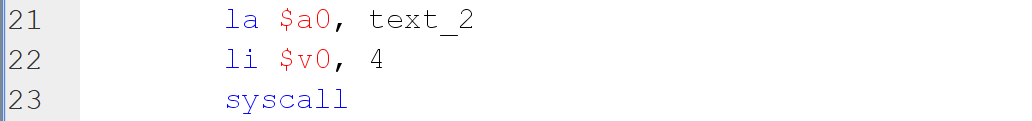
Do đã lưu được giá trị của n nên em sẽ tận dụng $t0 thành thanh ghi điều kiện so sánh để thoát vòng lặp



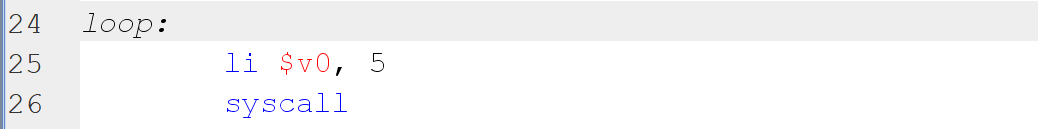
Tạo thanh ghi chạy ô nhớ cho array\_A



In ra màn hình I/O text\_2



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



Lưu giá trị đó xuống array\_A



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

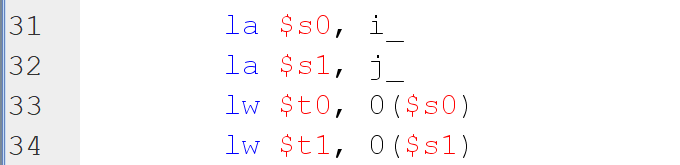


Rồi em so sánh xem $t3 đã đi qua phần tử cuối cùng của array\_A chưa. Nếu chưa thì lặp lại. Nếu rồi thì sẽ thoát vòng lặp



**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



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



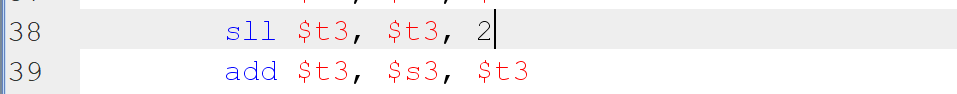
Nếu lớn hơn hoặc bằng thì nhảy đến label else



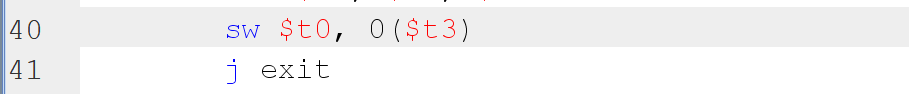
Nếu bé hơn thì sẽ lấy giá trị của i



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



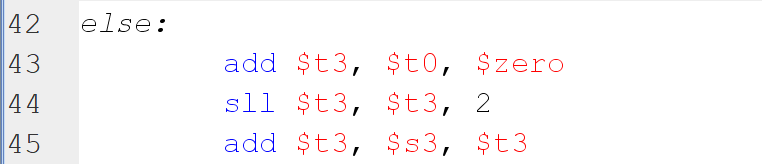
Lưu i vào A[i] và kết thúc chương trình



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



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



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

