CSE 331/503 Computer Organization Homework 2

Hasan Mutlu – 1801042673

Statement: I couldn’t do file read/write so I have created arrays in the .data section and printed the outputs in stdout. Instead of taking one input at a time, my program tests 6 arrays at once and their length is 10 by default. Their longest increasing subsequences are printed in order.

Also, I haven’t taken Data Structures and Alghoritms class yet so I don’t know about time and space complexity of my program.

Pseudocode: Pseudocode is included as a C code in the pseudocode.c file with extensive commenting as well. The comments in the assembly code are heavily in correlation with the C code.

```c
#define N 10    //Size of array
int array[] = {3,10,7,9,4,11,8,6,12,13};
int weight[N];
int longest[N];
int lenght = 1;
int endIndex = 0;
weight[0] = 1;

for (int i = 1; i < N; i++){
    weight[i] = 1;
    for (int j = i - 1; j >= 0; j--){
        if (array[i] > array[j] && weight[j] >= weight[i]){
            weight[i] = weight[j] + 1;
        }
    }
    if (weight[i] > lenght){
        lenght = weight[i];
        endIndex = i;}
}

int k = 0;
for(int i = endIndex; i >= 0 ; i--){
    if(weight[i] == lenght - k){
        longest[k] = array[i];
        k++;
    }
}

k--;
for(int i = k; i >= 0; i--){
    printf("%d ", longest[i]);
}
printf("- Size: %d", lenght);
```

Explanation of Functions:

main: It is the starting point of the program. length, endIndex, address of first array and weight[] are initialized in registers. It includes an inner loop, which covers all arrays by incrementing array address in $s6 by adding 4*size, which means address of the next array. It calls for1, assignLongest and print functions. When calling for1, PC address of the remaining of the main is recorded in $s5 as for1 function also calls another function by jal. After looping for numOfArrays times, program ends by jumping to empty finish label.

for1: It is the outer for loop in the pseudocode. Two iterators for array[] and weight[] are created in $t0 and $t1 in initialization part. Also, i is created as 4 in $s2, and arrays are iterated by adding 4*i to their starting addresses. It then covers entire array in the inner loop, starting from second element and calls the for2 function. The size of the subsequence and the index of greatest element of subsequence are also found in it.

for2: It is the nested for loop in the pseudocode. Initialization part of it is done before calling it, j is created and similarly to i, is added to array addresses for iteration. Inside this loop, the previous elements are covered. If there is a previous element found, which is smaller than current element and has a greater than or equal to weight, current element's weight is assigned to be 1 more than that.

assignLongest: It is the second standalone for loop in pseudocode. Addresses of array[], weight[] and longest[] are loaded in $t0, $t1 and $t4 in initialization part. i is initialized as endIndex*4 and used for iteration. k is initialized as it is, didn't used for iteration as longest[] will be always increasing. In inner loop, weight[] array is covered from weight[endIndex] element to first element. For every first encounter of (length – k) weight value in weight[] array, the corresponding element from main array is assigned into longest[], in other words, the longest subsequence is assigned into longest[], in a decreasing way.

print: It is the last standalone for loop in pseudocode. In initialization part, k is decremented to being size of longest[], i is initialized as it is for loop control and address of longest[] is decremented by 4 to show the last element in it. In its inner loop this function prints the longest increasing subsequence to the stdout. After the inner loop, " Size: " string, size of the subsequence and a newline character is printed.

Explanation of Weight Array:

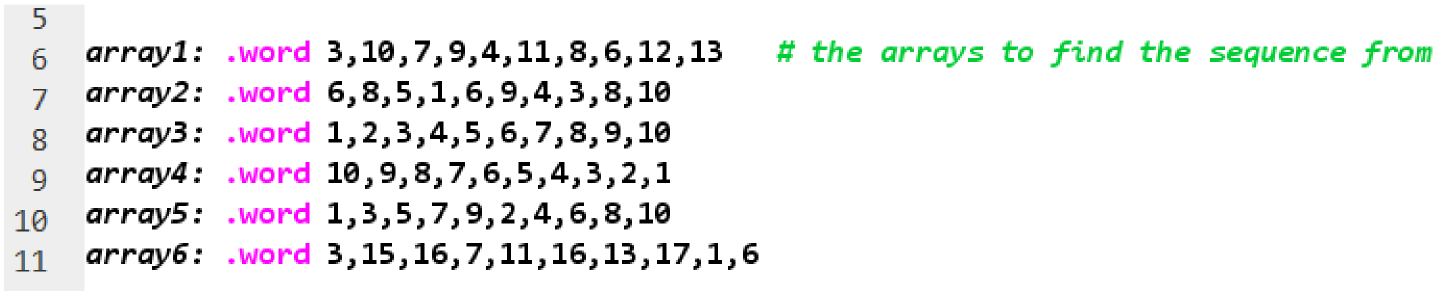
index	0	1	2	3	4	5	6	7	8	9
array[]	**3**	**10**	**7**	**9**	**4**	**11**	**8**	**6**	**12**	**13**
weight[]	**1**	**2**	**2**	**3**	**2**	**4**	**3**	**3**	**5**	**6**

longest = {array[9], array[8], array[5], array[3], array[2], array[0]} = {13, 12, 11, 9, 7, 3}

Then this longest array is printed in reverse order.

Test Cases:

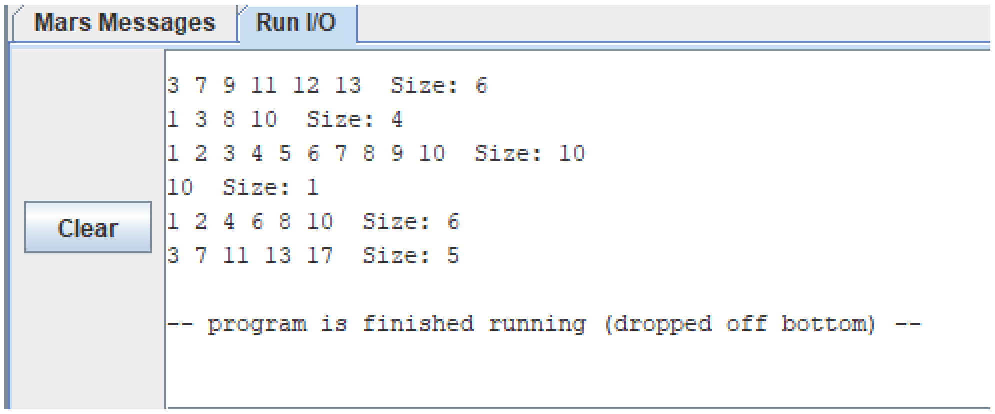
Input arrays:



```

array1: .word 3,10,7,9,4,11,8,6,12,13    # the arrays to find the sequence from
array2: .word 6,8,5,1,6,9,4,3,8,10
array3: .word 1,2,3,4,5,6,7,8,9,10
array4: .word 10,9,8,7,6,5,4,3,2,1
array5: .word 1,3,5,7,9,2,4,6,8,10
array6: .word 3,15,16,7,11,16,13,17,1,6
```

Output:



```
3 7 9 11 12 13  Size: 6
1 3 8 10  Size: 4
1 2 3 4 5 6 7 8 9 10  Size: 10
10  Size: 1
1 2 4 6 8 10  Size: 6
3 7 11 13 17  Size: 5

-- program is finished running (dropped off bottom) --
```