Requirements

-There can be more lines.

-You have to add just one space between numbers.

-You can’t add any unnecessary new lines.

-Every line has to has 6 number, not below or not more.

Registers Meaning

$a1 - str2 array addres for reading

$t0 - counter for digit of number

$t1 - save the read character for controlling that it is space or new line

$t2 - save new line character

$t3 - save space character

$t4 - saves the hexadecimal of first digit of number

$t5 - saves the hexadecimal of second digit of number

$t6 - str3 array address for saving

$s0 - size

$s1 - i

$s2 - i_1

$s3 - seq_number

$s4 - seq_counter

$s5 - seq_len

$s6 - str3

$s7 - temp_srt3

$t7 - temp2_srt3

File Read Function

```c
while(1){
    for (i = 0; fscanf(fp,"%d",&arr[i]) == 1 && i < 5  ; i++);

    if (i!=0)
    {
        seq_len = 0;
        recursive_max_squence_finder(temp_arr,print_arr,arr,size,0,0,0,0,&seq_len);
    }
    else
        break;
    for ( i = 0; i < seq_len; i++)
    {
        printf("%d  ",print_arr[i]);
    }
}
```

Time Complexity: O(n2) Space Complexity: O(1)

Firstly we will read file but we have to read one line every time. Because every line has different test sequence so that there are two loops for that .First loop is reading one line numbers to the array after that call the recursive max sequence function.

Recursive Max Squence Finder Function

```c
void recursive_max_squence_finder(
    int *temp_arr,
    int *print_arr,
    int *arr,
    int size,
    int i,
    int i_1,
    int seq_number,
    int seq_counter,
    int *seq_len
    ){



        for (;i<size;i++) {
            if(seq_number <= *(arr+i))
            {
                seq_counter++;
                *(temp_arr+i_1) = *(arr+i);
                recursive_max_squence_finder(temp_arr,print_arr,arr,size,i+1,i_1+1,*(arr+i),seq_counter,seq_len);
                seq_counter--;
            }
        }

    if(seq_counter >= *seq_len)
    {
        for(int j=0;j<seq_counter;j++ )
        {
            *(print_arr+j) = *(temp_arr+j);
        }

        //printf("Sequence-leng:%d \n\n",seq_counter);
        *seq_len = seq_counter;
    }
}
```

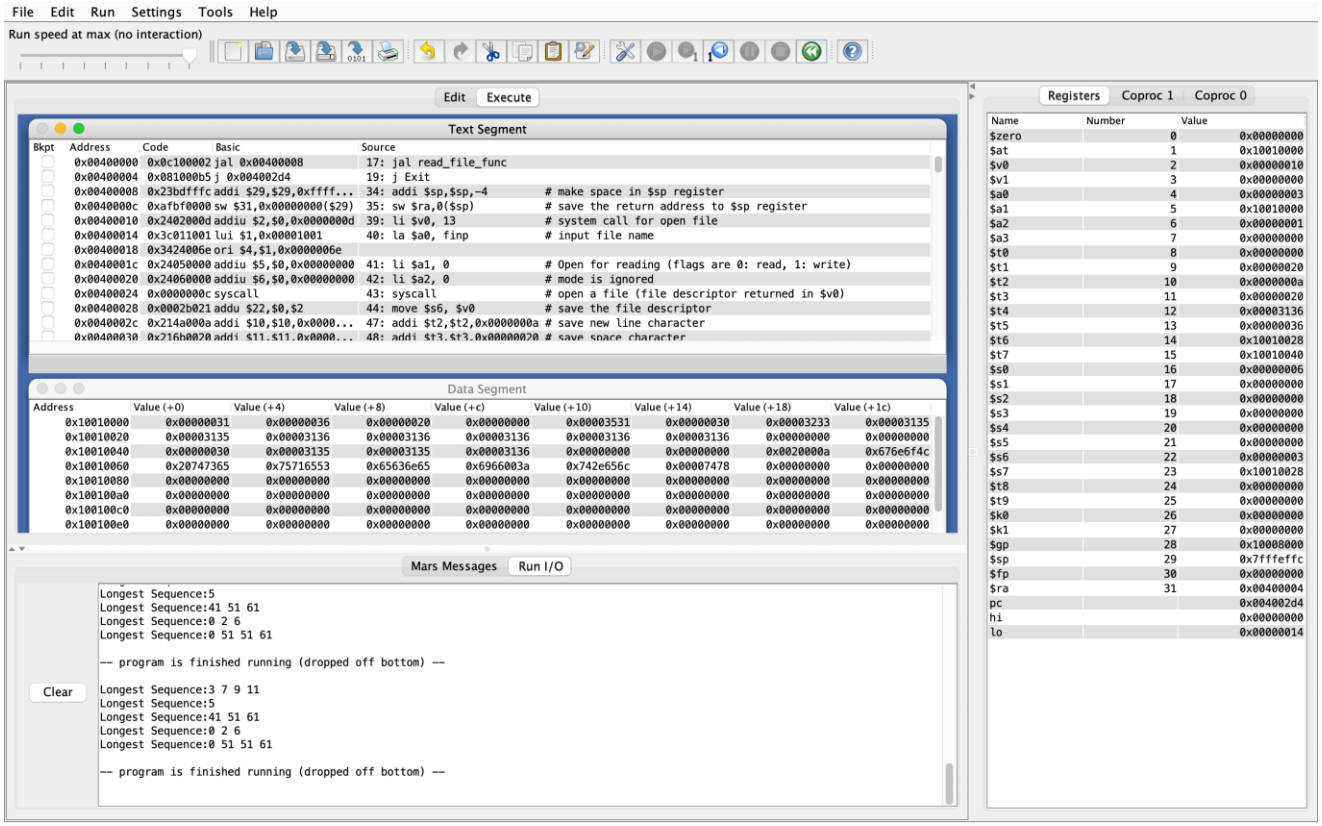
Time Complexity: O(n2) Space Complexity: O(n)

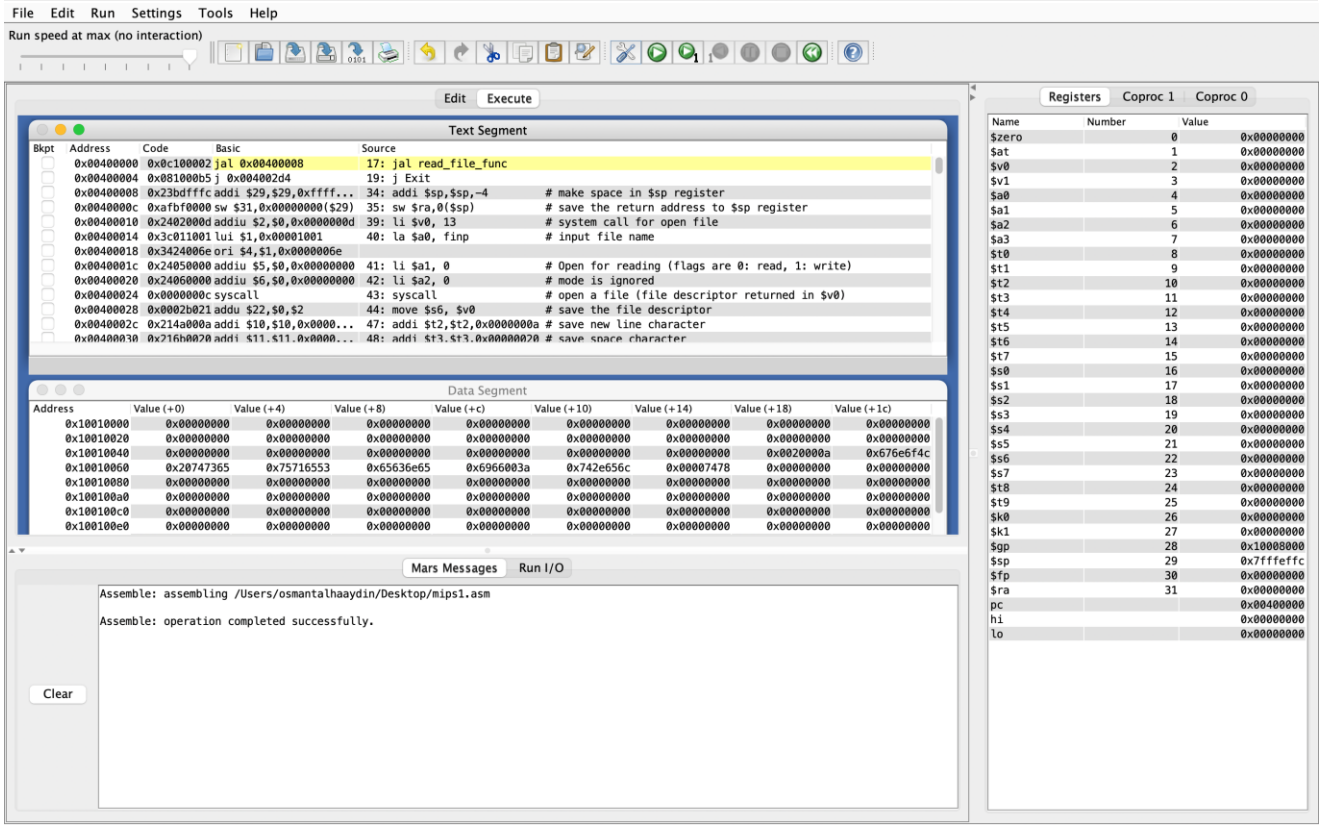
The first loop traverses the array elements one by one. And the number of sequence we keep while traveling and the number we visit are compared each time. If equal, this number is

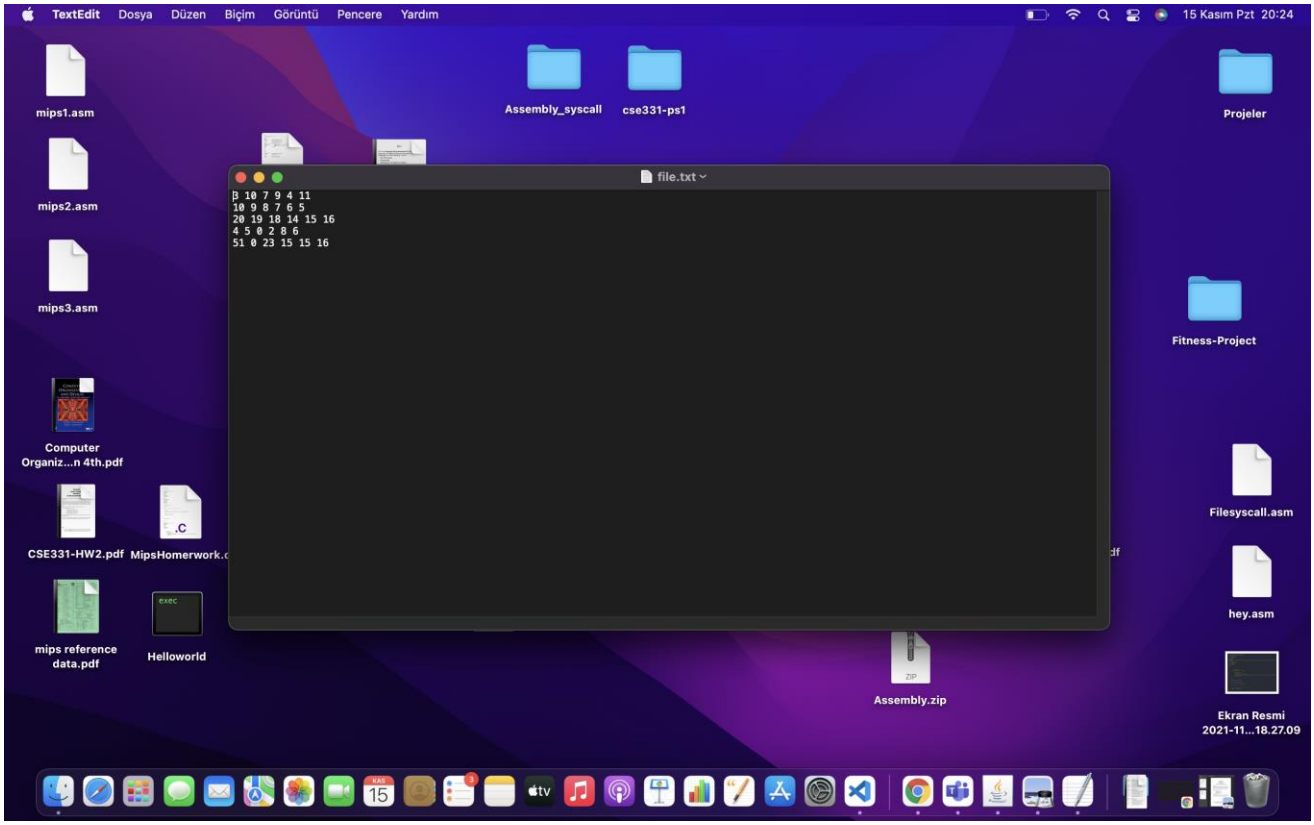
saved in the temp_arr array. Indexes are incremented by one. The new sequence number becomes our current number in the array(arr) and this function is called again.

If the length of the array we already have when the first loop is over is lower than what is calculated now, the longer array is recorded in the print array.

Test Photos


File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x0c100002 jal 0x00400008 17: jal read_file_func
0x00400004 0x081000b5 j 0x004002d4 19: j Exit
0x00400008 0x23bdfffc addi $29,$29,0xffff... 34: addi $sp,$sp,-4 # make space in $sp register
0x0040000c 0xafbf0000 sw $31,0x00000000($29) 35: sw $ra,0($sp) # save the return address to $sp register
0x00400010 0x2402000d addiu $2,$0,0x0000000d 39: li $v0, 13 # system call for open file
0x00400014 0x3c011001 lui $1,0x00001001 40: la $a0, finp # input file name
0x00400018 0x3424006e ori $4,$1,0x0000006e
0x0040001c 0x24050000 addiu $5,$0,0x00000000 41: li $a1, 0 # Open for reading (flags are 0: read, 1: write)
0x00400020 0x24060000 addiu $6,$0,0x00000000 42: li $a2, 0 # mode is ignored
0x00400024 0x0000000c syscall 43: syscall # open a file (file descriptor returned in $v0)
0x00400028 0x0002b021 addu $22,$0,$2 44: move $s6, $v0 # save the file descriptor
0x0040002c 0x214a000a addi $10,$10,0x0000... 47: addi $t2,$t2,0x0000000a # save new line character
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 0x00000031 0x00000036 0x00000020 0x00000000 0x00003531 0x00000030 0x00003233 0x00003135
0x10010020 0x00003135 0x00003136 0x00003136 0x00003136 0x00003136 0x00000000 0x00000000 0x00000000
0x10010040 0x00000030 0x00003135 0x00003135 0x00003136 0x00000000 0x00000000 0x0020000a 0x676e6f4c
0x10010060 0x20747365 0x75716553 0x65636e65 0x6966003a 0x742e656c 0x00007478 0x00000000 0x00000000
0x10010080 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100a0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100c0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100e0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
Mars Messages Run I/O
Clear
Longest Sequence:5
Longest Sequence:41 51 61
Longest Sequence:0 2 6
Longest Sequence:0 51 51 61
-- program is finished running (dropped off bottom) --
Longest Sequence:3 7 9 11
Longest Sequence:5
Longest Sequence:41 51 61
Longest Sequence:0 2 6
Longest Sequence:0 51 51 61
-- program is finished running (dropped off bottom) --
Registers Coproc 1 Coproc 0
Name Number Value
$zero 0 0x00000000
$at 1 0x10010000
$v0 2 0x00000010
$v1 3 0x00000000
$a0 4 0x00000003
$a1 5 0x10010000
$a2 6 0x00000001
$a3 7 0x00000000
$t0 8 0x00000000
$t1 9 0x00000020
$t2 10 0x0000000a
$t3 11 0x00000020
$t4 12 0x00003136
$t5 13 0x00000036
$t6 14 0x10010028
$t7 15 0x10010040
$s0 16 0x00000006
$s1 17 0x00000000
$s2 18 0x00000000
$s3 19 0x00000000
$s4 20 0x00000000
$s5 21 0x00000000
$s6 22 0x00000003
$s7 23 0x10010028
$t8 24 0x00000000
$t9 25 0x00000000
$k0 26 0x00000000
$k1 27 0x00000000
$gp 28 0x10008000
$sp 29 0x7fffeffc
$fp 30 0x00000000
$ra 31 0x00400004
pc 0x004002d4
hi 0x00000000
lo 0x00000014


File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x0c100002 jal 0x00400008 17: jal read_file_func
0x00400004 0x081000b5 j 0x004002d4 19: j Exit
0x00400008 0x23bdfffc addi $29,$29,0xffff... 34: addi $sp,$sp,-4 # make space in $sp register
0x0040000c 0xafbf0000 sw $31,0x00000000($29) 35: sw $ra,0($sp) # save the return address to $sp register
0x00400010 0x2402000d addiu $2,$0,0x0000000d 39: li $v0, 13 # system call for open file
0x00400014 0x3c011001 lui $1,0x00001001 40: la $a0, finp # input file name
0x00400018 0x3424006e ori $4,$1,0x0000006e
0x0040001c 0x24050000 addiu $5,$0,0x00000000 41: li $a1, 0 # Open for reading (flags are 0: read, 1: write)
0x00400020 0x24060000 addiu $6,$0,0x00000000 42: li $a2, 0 # mode is ignored
0x00400024 0x0000000c syscall 43: syscall # open a file (file descriptor returned in $v0)
0x00400028 0x0002b021 addu $22,$0,$2 44: move $s6, $v0 # save the file descriptor
0x0040002c 0x214a000a addi $10,$10,0x0000... 47: addi $t2,$t2,0x0000000a # save new line character
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010020 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010040 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x0020000a 0x676e6f4c
0x10010060 0x20747365 0x75716553 0x65636e65 0x6966003a 0x742e656c 0x00007478 0x00000000 0x00000000
0x10010080 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100a0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100c0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100e0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
Mars Messages Run I/O
Clear
Assemble: assembling /Users/osmantalhaaydin/Desktop/mips1.asm
Assemble: operation completed successfully.
Registers Coproc 1 Coproc 0
Name Number Value
$zero 0 0x00000000
$at 1 0x00000000
$v0 2 0x00000000
$v1 3 0x00000000
$a0 4 0x00000000
$a1 5 0x00000000
$a2 6 0x00000000
$a3 7 0x00000000
$t0 8 0x00000000
$t1 9 0x00000000
$t2 10 0x00000000
$t3 11 0x00000000
$t4 12 0x00000000
$t5 13 0x00000000
$t6 14 0x00000000
$t7 15 0x00000000
$s0 16 0x00000000
$s1 17 0x00000000
$s2 18 0x00000000
$s3 19 0x00000000
$s4 20 0x00000000
$s5 21 0x00000000
$s6 22 0x00000000
$s7 23 0x00000000
$t8 24 0x00000000
$t9 25 0x00000000
$k0 26 0x00000000
$k1 27 0x00000000
$gp 28 0x10008000
$sp 29 0x7fffeffc
$fp 30 0x00000000
$ra 31 0x00000000
pc 0x00400000
hi 0x00000000
lo 0x00000000



My C Code For This Program

```
#include <stdio.h>

#include <stdio.h>

void recursive_max_squence_finder(

int *temp_arr,

int *print_arr,

int *arr,

int size,

int i,

int i_1,

int seq_number,

int seq_counter,

int *seq_len);


int main()

{
```

```
int arr[6];
int print_arr[6];
int temp_arr[6];
int i;
int temp;
int seq_len =0;
int size=6;

FILE *fp;
fp = fopen("file.txt","r");

if(fp != NULL)
{
while(1){
for (i = 0; fscanf(fp,"%d",&arr[i]) == 1 && i < 5 ; i++);
if (i!=0)
{
seq_len = 0;
recursive_max_squence_finder(temp_arr,print_arr,arr,size,0,0,0,0,&seq_len);
}
else
break;
for ( i = 0; i < seq_len; i++)
{
printf("%d ",print_arr[i]);
}
}
}
else
printf("Açılmadı.");
return 0;
}
```

```
void recursive_max_squence_finder(
int *temp_arr,
int *print_arr,
int *arr,
int size,
int i,
int i_1,
int seq_number,
int seq_counter,
int *seq_len
){
for (;i<size;i++) {
if(seq_number <= *(arr+i))
{
seq_counter++;
*(temp_arr+i_1) = *(arr+i);
recursive_max_squence_finder(temp_arr,print_arr,arr,size,i+1,i_1+1,*(arr+i),seq_counter,seq_len);
seq_counter--;
}
}

if(seq_counter >= *seq_len)
{
for(int j=0;j<seq_counter;j++ )
{
*(print_arr+j) = *(temp_arr+j);
}
*seq_len = seq_counter;
}
}
```