.data

arr: .word 12 121 1322 522 6

length: .word 5

space: .asciiz " "

.text

la $a0, arr

lw $a1, length

jal insertionsort

la $a0, arr

lw $a1, length

jal printarray

li $v0, 10

syscall

insertionsort:

addi $sp, $sp, -12

sw $s0, 8($sp)

sw $s1, 4($sp)

sw $s2, 0($sp)

# salvo s0, s1, s2

li $s0, 1 # i in $s0

forloop:

beq $s0, $a1, escifor

sll $t0, $s0, 2 # ASM index in t0

add $t0, $t0, $a0 #address in t0

lw $s1, 0($t0) #arr[i] in key

addi $s2, $s0, -1 # j in s2

whileloop:

blt $s2, $zero, esciwhile

sll $t0, $s2, 2 # j\*4

add $t0, $t0, $a0

lw $t0, 0($t0) #arr[j]

ble $t0, $s1, esciwhile

addi $t1, $s2, 1

sll $t1, $t1, 2

add $t1, $t1, $a0

sw $t0, 0($t1) #t0 indirizzo

addi $s2, $s2, -1

j whileloop

esciwhile:

addi $t0, $s2, 1

sll $t0, $t0, 2

add $t0, $t0, $a0

sw $s1, 0($t0)

addi $s0, $s0, 1

j forloop

escifor:

lw $s2, 0($sp)

lw $s1, 4($sp)

lw $s0, 8($sp)

addi $sp, $sp, 12

jr $ra

printarray:

addi $sp, $sp, -4

sw $s0, 0($sp)

li $s0, 0

move $t1, $a0

forprint:

beq $s0, $a1, esciprint

sll $t0, $s0, 2

add $t0, $t0, $t1

lw $a0, 0($t0)

li $v0, 1

syscall

li $v0, 4

la $a0, space

syscall

addi $s0, $s0, 1

j forprint

esciprint:

lw $s0, 0($sp)

addi $sp, $sp, 4

jr $ra

.data

ciao: .word 12 11 13 5 6

dimensione: .word 5

n: .word 5

acapo: .asciiz "\n"

space: .asciiz " "

.text

#carico i dati

la $a1, ciao

lw $a2, dimensione

lw $a3, n

#chiamata a funzione

jal InsertionSort

#chimata a funzione print

jal printArray

#uscita dal programma

li $v0, 10

syscall

#InsertionSort

InsertionSort:

li $t0,1

#inizio ciclo for

ciclo:

bgt $t0,$a3,exitfor

#tutta sta roba per k=a[i]

sll $t2,$t0,2 #shift di 2 quindi moltiplico per 4 i e mi da l'indirizzo successivo

add $t2,$t2,$a1 #indirizzo di a[i] //sommo all'indirizzo base di $a1

lw $t3,0($t2) #$t3 = key

sub $t4,$t0,1

#inizio ciclo while

while:

#condizione ciclo

ble $t4,0,continua

#se $t4 non e' maggiore o uguale a 0

j exitwhile

continua: #$t4 = j

#tutto sto popo per arr[j] > key

sll $t5,$t4,2

add $t5,$t5,$a1

lw $t6, 0($t5)

bgt $t6,$t3,continua2

j exitwhile

#fine arr[j] > key

continua2:

#inizio istruzione arr[j + 1] = arr[j];

sll $t7,$t6,2

add $t7,$t7,$a1

lw $t8, 0($t7) #$t8 = arr[j + 1]

move $t8,$t6

#fine istruzione arr[j + 1] = arr[j];

sub $t4,$t4,1 #j = j - 1;

#fine ciclo while

j while

exitwhile:

move $t8,$t3 #arr[j + 1] = key;

#fine ciclo for

add $t0,$t0,1

j ciclo

exitfor:

jr $ra

printArray:

#inizio ciclo for

li $t0, 0

printfor:

bgt $t0,$a3,exitprint

sll $t2,$t0,2 #shift di 2 quindi moltiplico per 4 i e mi da l'indirizzo successivo

add $t2,$t2,$a1 #indirizzo di a[i]

lw $t3, 0($t2)

#inizio prima stampa a video

move $a0, $t3

li $v0, 1

syscall

#stampa spazio tra una variabile e un'altra

la $a0, space

li $v0, 4

syscall

#seconda stampa a video

la $a0, acapo

li $v0, 4

syscall

#fine ciclo for

add $t0,$t0,1

j printfor

exitprint:

jr $ra

#include <stdio.h>

// C program for insertion sort #include <math.h>  
#include <stdio.h>

/\* Function to sort an array using insertion sort\*/

void insertionSort(int arr[], int n) {

int i, key, j;  
for (i = 1; i < n; i++) {

key = arr[i]; j = i - 1;

/\* Move elements of arr[0..i-1], that are greater than key, to one position ahead of their current position \*/

while (j >= 0 && arr[j] > key) {

arr[j + 1] = arr[j];

j = j - 1; }

arr[j + 1] = key;

} }

// A utility function to print  
// an array of size n  
void printArray(int arr[], int n) {

int i;  
for (i = 0; i < n; i++)

printf("%d ", arr[i]); printf("\n");

}

// Driver code int main()  
{

int arr[] = {12, 11, 13, 5, 6}; int n = 5;  
insertionSort(arr, n); printArray(arr, n);

return 0; }