Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Отчёт по лабораторной работе № 3

Дисциплина: Низкоуровневое программирование

Тема: машина Тьюринга

Выполнил студент гр. 3530901/10003 _____Я.А. Иванов
(подпись)

Принял старший преподаватель _____ Д.А. Корнеев
(подпись)

“ _” _____ 2022 г.

Санкт-Петербург
2022

Оглавление

Техническое задание:

Разработать программу для RISC-V, реализующую определение медианы in-place

Метод решения

Для поиска медианы нужно выбрать случайно число и расположить по принципу большие - справа, меньшие – слева. Из этих двух групп мы выбираем ту, в которой находится индекс медианы (половина от длины массива). Далее повторяем прошлые шаги с подмассивом до тех пор, пока наше случайное число не будет иметь индекс медианы. Приведём реализацию на языке C

```c
void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

int main() {
    int arr[7] = {9, 13, 7, 1, 13, 5,13};
    int size = sizeof(arr) / sizeof(int);
    int *interval_right = &arr[size];
    int *left = &arr[0];
    int *right;
    int *i;
    int *j;
    int median = (size - 1) / 2;
    int answer;

    while (true) {
        i = left + 1;
        right = interval_right - 1;
        while (i < interval_right) {
            if (*left <= *i) {
                right--;
            }
            i++;
        }
        swap(left, right);
        j = right + 1;
        i = left;
        while (i < right) {
            if (*i >= *right) {
                while (*j >= *right) {
                    j++;
                }
                swap(left, right);
                j++;
            }
            i++;
        }
        if ((int) (right - &arr[0]) == median) {
          answer = *right;
        } else {
            if (median < (right - &arr[0])) {
                interval_right = right;
            } else {
                left = right + 1;
            }
        }
    }
    return 0;
}
```

Руководство программисту

Начальные данные к программе: адрес нулевого элемента массива (и соответственно сам массив) и его длина. В реализации без подпрограммы адрес и длина хранятся в регистрах a2 и a3 соответственно. В реализации через подпрограмму предполагается, что нулевым аргументов (регистр a0) передается длина массива, а первым аргументом (регистр a1) передается адрес нулевого элемента массива.

Реализация программы 1

```
.text
__start:
.globl __start
    #VARS
    #a2(arr)
    #a3(len)
    #a4(interval_right)
    #a5(left)
    #a6(right)
    #a7(median)

    #t1(i)
    #t2(j)

    la a2, array #a2(arr) адрес начала массива
    lw a3, array_length #a3(len) длина
    slli t1, a3, 2
    add a4, a2, t1 #a4 (interval_right) = arr[size]
    mv a5, a2 # a5(left) = a2(arr)
    addi a7, a3, -1
    li t1, 2 #временная двойка для выполнения деления
    divu a7, a3, t1 # a7(median) = (a3(len) - 1) / 2
endless_loop:
    addi t1, a5, 4 #t1(i) = a5(left) + 1
    addi a6, a4, -4 #a6(right) = a4(interval_right) - 1

find_swap_index_loop:
#while
    bgeu t1, a4, end_find_swap_index_loop #if !(interval_right > i) goto end_find_swap_index_loop
#if
    lw t3, 0(a5) #загрузка *left
    lw t4, 0(t1) #загрузка *i
    blt t4, t3,  endif_find_swap #if !(*left <= *i)  goto endif_find_swap
    addi a6, a6, -4 #right--
endif_find_swap:
    addi t1, t1, 4 #i++
    j find_swap_index_loop
end_find_swap_index_loop:
#swap
    lw t4, 0(a6) #загрузка *right
    sw t3, 0(a6) #right = left(из регистра)
    sw t4, 0(a5) #left = right(из регистра)
#endswap
    mv t1, a5 #t1(i) = left
    addi t2, a6, 4 #t2(j) = right + 1
assig_addres_loop:
#while
    bgeu t1, a6,  end_assig_addres_loop  #if !(right > i) goto end_assig_addres_loop
#if
    lw t3, 0(t1) #загрузка *i
    lw t4, 0(a6) #загрузка *right
    blt t3, t4, end_assig_addres_if  #if !(*right <= *i) goto end_assig_addres_if
assig_addres_loop_j:
#while
    lw t3, 0(t2) #загрузка *j
    blt t3, t4, end_assig_addres_loop_j #if !(*right <= *j ) goto end_assig_addres_loop_j
    addi t2, t2, 4
    j assig_addres_loop_j
end_assig_addres_loop_j:
#swap
    lw t4, 0(t1) #загружаем *i
    sw t3, 0(t1) #i = j(из регистра)
    sw t4, 0(t2) #j = i(из регистра)
#endswap
    addi t2, t2, 4 #j++
end_assig_addres_if:
    addi t1, t1, 4 #i++
    j assig_addres_loop
end_assig_addres_loop:
    sub t3, a6, a2
    li t5, 4 #временная двойка для выполнения деления
    divu t3, t3, t5 # t3 = t3 / 4 для получения индекса
    beq t3, a7, loop_exit #if (right[index] == median) goto loop_exit
    bgeu a7, t3, move_left #if  !(right > median) goto move_left
    mv a4, a6
    j endless_loop
move_left:
    addi a5, a6, 4
    j endless_loop
loop_exit:
    lw a1, 0(a6)
    li a0, 10
    ecall
.rodata
array_length:
    .word 5
.data
array:
    .word -2, -10, 10, 15, 100
```

Работа программы 1

a0	x10	0x0000000a
a1	x11	0x0000000a

Реализация программы 2

```
.text
__start:
.globl __start
    call main
finish:
    li a0, 10
    ecall
```

Настройщик

```
.text
main:
.globl main
    la a0, array_1 #arr
    lw a1, array_length_1 #lenght = 5
    mv s1, ra
    call findMedian
    mv ra, s1
    ret

.rodata
array_length_1:
    .word 5
.data
array_1:
    .word 1,2,3,4,5
```

Основная программа

```
.text
findMedian:
.globl findMedian
    #VARS
    #a2(arr)
    #a3(len)
    #a4(interval_right)
    #a5(left)
    #a6(right)
    #a7(median)

    #t1(i)
    #t2(j)

    mv a2, a0 #загрузка в a2 значение из параметром функции
    mv a3, a1 #загрузка в a3 значение из параметром функции
    slli t1, a3, 2
    add a4, a2, t1 #a4 (interval_right) = arr[size]
    mv a5, a2 # a5(left) = a2(arr)
    addi a7, a3, -1
    li t1, 2 #временная двойка для выполнения деления
    divu a7, a3, t1 # a7(median) = (a3(len) - 1) / 2
endless_loop:
    addi t1, a5, 4 #t1(i) = a5(left) + 1
    addi a6, a4, -4 #a6(right) = a4(interval_right) - 1

find_swap_index_loop:
#while
    bgeu t1, a4, end_find_swap_index_loop #if !(interval_right > i) goto end_find_swap_index_loop
#if
    lw t3, 0(a5) #загрузка *left
    lw t4, 0(t1) #загрузка *i
    blt t4, t3,  endif_find_swap #if !(*left <= *i)  goto endif_find_swap
    addi a6, a6, -4 #right--
endif_find_swap:
    addi t1, t1, 4 #i++
    j find_swap_index_loop
end_find_swap_index_loop:
#swap
    lw t4, 0(a6) #загрузка *right
    sw t3, 0(a6) #right = left(из регистра)
    sw t4, 0(a5) #left = right(из регистра)
#endswap
    mv t1, a5 #t1(i) = left
    addi t2, a6, 4 #t2(j) = right + 1
assig_addres_loop:
#while
    bgeu t1, a6,  end_assig_addres_loop  #if !(right > i) goto end_assig_addres_loop
#if
    lw t3, 0(t1) #загрузка *i
    lw t4, 0(a6) #загрузка *right
    blt t3, t4, end_assig_addres_if  #if !(*right <= *i) goto end_assig_addres_if
assig_addres_loop_j:
#while
    lw t3, 0(t2) #загрузка *j
    blt t3, t4, end_assig_addres_loop_j #if !(*right <= *j ) goto end_assig_addres_loop_j
    addi t2, t2, 4
    j assig_addres_loop_j
end_assig_addres_loop_j:
#swap
    lw t4, 0(t1) #загружаем *i
    sw t3, 0(t1) #i = j(из регистра)
    sw t4, 0(t2) #j = i(из регистра)
#endswap
    addi t2, t2, 4 #j++
end_assig_addres_if:
    addi t1, t1, 4 #i++
    j assig_addres_loop
end_assig_addres_loop:
    sub t3, a6, a2
    li t5, 4 #временная двойка для выполнения деления
    divu t3, t3, t5 # t3 = t3 / 4 для получения индекса
    beq t3, a7, loop_exit #if (right[index] == median) goto loop_exit
    bgeu a7, t3, move_left #if  !(right > median) goto move_left
    mv a4, a6
    j endless_loop
move_left:
    addi a5, a6, 4
    j endless_loop
loop_exit:
    lw a1, 0(a6)
    li a0, 10
    ret
```

Подпрограмма

Работа программы 2

a0	x10	0x0000000a
a1	x11	0x00000003

Выводы:

В ходе выполнения данной лабораторной работы была разработана программа для RISC-V для поиска медианы массива. Была создана самостоятельная программа и версия для подпрограммы.