Санкт-Петербургский политехнический университет Петра Великого
Кафедра компьютерных систем и программных технологий

Отчёт по лабораторной работе

Дисциплина: Низкоуровневое программирование

Тема: RISC-V

Выполнил студент гр. 3530901/10005 _____
Калашников О. Ю.

(подпись)

Преподаватель _____
Коренев Д.А.

(подпись)

“___”_____ 2022 г.

Санкт-Петербург
2022

Оглавление

ТЗ

Написать программу для машины RISC-V, которая реализует нахождение максимального элемента в массиве чисел. Необходимо также выделить основной функционал в подпрограмму и написать для неё тестовую программу.

Метод решения

Для нахождения максимального элемента массива необходимо циклически пройти каждый элемент, сравнивая его с максимальным значением, найденным на данный момент (для первого элемента текущее максимальное значение принимается за 0).

Например, рассмотрим массив [3, 5, 9, 4, 1]:
1. 3 > 0 => 3 — текущий максимальный элемент;
2. 5 > 3 => 5 — текущий максимальный элемент;
3. 9 > 5 => 9 — текущий максимальный элемент;
4. 4 < 9 => 9 — остаётся текущим максимальным элементом;
5. 1 < 9 => 9 — остаётся текущим максимальным элементом;
Ответ: 9.

Реализация программы

program.s

```
.text
__start:
.globl __start

    la a0, array_length
    lw a0, 0(a0)            # загружаем длину массива
    la a1, array            # загружаем адрес массива
    la t0, answer           # загружаем адрес ячейки памяти для ответа

loop:
    beqz a0, exit           # если длина массива 0, то останов
    addi a0, a0, -1         # отнимаем 1 от длинны массива
    lw t2, 0(a1)            # загружаем в t2 текущий элемент
    bltu t3, t2, writeMax   # if t2 > t3 goto writeMax

    addi a1, a1, 4          # записываем в a1 следующее значение массива
    jal zero, loop          # начинаем новую итерацию массива

writeMax:
    lw t3, 0(a1)            # записываем в t3 текущее максимальное значение массива
    addi a1, a1, 4          # записываем в a1 следующее значение массива
    jal zero, loop          # начинаем новую итерацию массива

exit:
    sw t3, 0(t0)
    li a0, 10               # записываем 10 в a0
    ecall                   # останов

.rodata
array_length:
    .word 8

array:
    .word 6, 12, 18, 24, 48, 69, 70, 69

.data
answer:
    .word 0                 # 0x00010078
```

Результат работы программы

Возьмём массив [6, 12, 18, 24, 48, 69, 70, 69]. Ожидаемый результат — 70. Ответ находится в ячейке памяти с адресом 0x00010078:

Registers Memory Cache

Address	+3	+2	+1	+0
0x00010088	0	0	0	0
0x00010084	0	0	0	0
0x00010080	0	0	0	0
0x0001007c	0	0	0	0
0x00010078	0	0	0	70
0x00010074	0	0	0	69
0x00010070	0	0	0	70
0x0001006c	0	0	0	69
0x00010068	0	0	0	48
0x00010064	0	0	0	24

Результат совпал с ожидаемым.

Реализация подпрограммы и тестовой программы

Стартовый файл программы

```
1 # start.s
2 .text
3 __start:
4 .globl __start
5     call main
6     mv a1, a0
7     li a0, 17
8     ecall
```

Тестовая программа

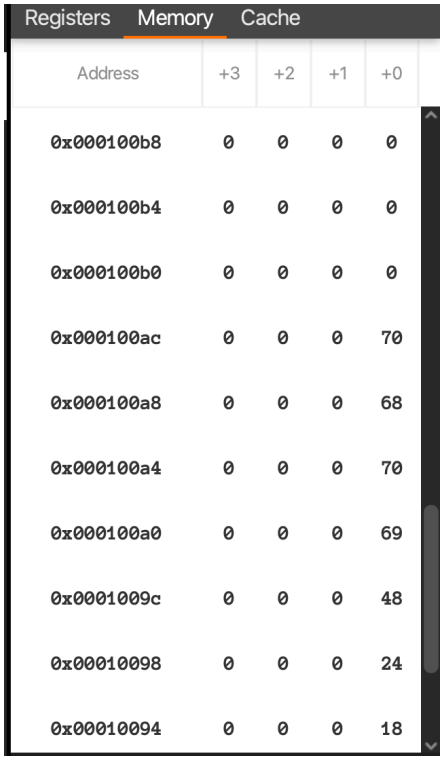
```
# main.s
.text
main:
.globl main

    addi sp, sp, -16        # выделяем память в стеке
    sw ra, 12(sp)           # сохраняем адрес возврата

    la a0, array_length
    lw a0, 0(a0)            # загружаем длину массива
    la a1, array            # загружаем адрес массива
    call find_max           # вызов подпрограммы
    la t0, answer           # загружаем адрес ячейки памяти для ответа
    sw a0, 0(t0)            # сохраняем ответ

    lw ra, 12(sp)           # восстанавливаем адрес возврата
    addi sp, sp, 16         # освобождаем выделенную память в стеке
    li a0, 0                # записываем 0 в a0
    ret                     # возврат

.rodata
array_length:
    .word 8

array:
    .word 6, 12, 18, 24, 48, 69, 70, 68

.data
answer:
    .word 0                 # 0x000100ac
```

Подпрограмма

```
# find_max.s
.text
find_max:
.globl find_max

loop:
    beqz a0, exit           # если длина массива 0, то останов
    addi a0, a0, -1         # отнимаем 1 от длинны массива
    lw t2, 0(a1)            # загружаем в t2 текущий элемент
    bltu t3, t2, writeMax   # if t2 > t3 goto writeMax

    addi a1, a1, 4          # записываем в a1 следующее значение массива
    jal zero, loop          # начинаем новую итерацию массива

writeMax:
  lw t3, 0(a1)              # записываем в t3 текущее максимальное значение массива
  addi a1, a1, 4            # записываем в a1 следующее значение массива
  jal zero, loop            # начинаем новую итерацию массива

exit:
    mv a0, t3               # записываем ответ в a1
    ret                     # возврат
```

Результат работы программы

Возьмём массив [6, 12, 18, 24, 48, 69, 70, 69]. Ожидаемый результат — 70. Ответ находится в ячейке памяти с адресом 0x000100ac:



Address	+3	+2	+1	+0
0x000100b8	0	0	0	0
0x000100b4	0	0	0	0
0x000100b0	0	0	0	0
0x000100ac	0	0	0	70
0x000100a8	0	0	0	68
0x000100a4	0	0	0	70
0x000100a0	0	0	0	69
0x0001009c	0	0	0	48
0x00010098	0	0	0	24
0x00010094	0	0	0	18

Результат совпал с ожидаемым.

Вывод

В ходе выполнения лабораторной работы мной была разработана программы для машины RISC-V, в том числе и для реализации с подпрограммой, которые реализуют нахождение максимального элемента в заданном массиве чисел. Результаты работы программы для обоих загрузчиков совпали с ожидаемыми.