НИУ ВШЭ

Архитектура вычислительных систем

Индивидуальное домашнее задание №1

Вариант 11

Студент: Лебедев Андрей Андреевич БПИ234

Москва
2024

Введение

В этом проекте была разработана программа на ассемблере RISC-V, которая обрабатывает массив целых чисел, введенный пользователем. Программа принимает количество элементов, проверяет корректность введенных данных, считывает массив, а затем обрабатывает его для создания нового массива по заданным критериям.

Структура файлов

Проект состоит из следующих файлов:

1. `main.s` – основной файл программы, который обрабатывает ввод пользователя и вызывает функцию для обработки массива.
2. `array_processing.s` – модуль, содержащий функцию обработки массива.
3. `io_macros.s` – библиотека макросов для упрощения работы с вводом/выводом.
4. `test.s` – файл для тестирования функциональности программы.

Основная программа (`main.s`)

Основная программа выполняет функции управления пользователем и обработки ввода:

```
.include "io_macros.s"      # Include the macros library
.data
    promptE:    .asciz "\nN is out of range\n"
    promptN:    .asciz "Enter the number of elements N (1-10): "
    promptA:    .asciz "Enter the element of array A: "
    msgB:       .asciz "Elements of array B: "
    newline:     .asciz "\n"
    .align 2
    arrayA:     .space 40        # Array A (10 elements of 4 bytes)
    .align 2
    arrayB:     .space 40        # Array B (10 elements of 4 bytes)

.text
.global main

main:
    # s0 - N
    # s1 - arrayA
    # s2 - arrayB
    # s3 - length of array B
    la s1, arrayA
    la s2, arrayB

    # Input the number of elements N
    print_string promptN
    read_int s0

    # Check for valid N
    li t1, 1
    li t2, 10
    blt s0, t1, invalidN
    bgt s0, t2, invalidN
```

```
33
34      # Input array A
35      read_array(s0, s1)  # Input array A
36
37      # Parameters for process_array function
38      mv a0, s0
39      mv a1, s1
40      mv a2, s2
41      li a3, 0
42
43      # Create array B
44      jal process_array
45      mv s3, a3
46      mv s2, a2
47      print_new_array
48      print_array(s3, s2)
49
50  exit_program:
51      li a7, 10           # System call to exit the program
52      ecall               # Call the operating system
53
54  invalidN:
55      print_string promptE      # Output error message
56      j main                    # Repeat input for N
```

Основные функции `main.s`

- **Обработка ввода:** Программа запрашивает у пользователя количество элементов (N) и проверяет, что оно находится в пределах 1-10.
- **Ввод массива:** Считывает элементы массива A от пользователя.
- **Обработка массива:** Вызывает функцию process_array для обработки введенного массива и создания нового.
- **Вывод:** Программа выводит новый массив на экран.

Обработка массива (`array_processing.s`)

Функция process_array выполняет основную логику обработки массива:

```
1   .text
2   .global process_array
3   process_array:
4       # Input parameters:
5       # a0 - number of elements,
6       # a1 - address of the start of array A,
7       # a2 - address of the start of array B
8       # Output data:
9       # a2 - address of the start of array B
10      # a3 - length of array B
11
12      li t1, 0                # Index for array A
13      li t3, 0                # Flag for first positive element found
14      li t6, 4                # Required offset
15      mul t6, t6, a0
16      add a2, a2, t6
```

```
17
18 reverse_loop:
19     bge t1, a0, reverse_done    # If index >= N, exit the loop
20
21     lw t4, 0(a1)             # Load element from array A
22     addi a1, a1, 4           # Address of the next element
23
24     bgtz t4, skip_positive    # If element is positive, skip it
25     j add_element
26
27 add_element:
28     sw t4, 0(a2)             # Store element in array B
29     addi a2, a2, -4          # Address of the next element
30     addi t1, t1, 1
31     addi a3, a3, 1
32     j reverse_loop
33
34 skip_positive:
35     bgtz t3, add_element
36     addi t3, t3, 1
37     addi t1, t1, 1
38     j reverse_loop
39
40 reverse_done:
41     addi a2, a2, 4
42     ret
```

Основные функции `process_array`

- **Итерация по массиву A:** Программа проходит по всем элементам массива A и определяет, какие элементы должны быть добавлены в массив B.
- **Условие добавления:** Все элементы массива A, кроме первого положительного, добавляются в список B в обратном порядке.

Макросы ввода/вывода (`io_macros.s`)

Файл `io_macros.s` содержит макросы для упрощения работы с вводом и выводом:

```
1  .data
2      prompt:      .asciz "Enter array element: "
3      promptOLD:   .asciz "Old array: "
4      promptO:     .asciz "New array: "
5      space:       .asciz " "
6      new_line:    .asciz "\n"
7
8  .macro read_int %reg
9      li a7, 5                 # System call for reading an integer
10     ecall
11     mv %reg, a0              # Save the result in the register
12 .end_macro
13
14 .macro print_string %str
15     li a7, 4                 # System call for printing a string
16     la a0, %str              # Load the address of the string
17     ecall
18 .end_macro
```

```
19
20 .macro read_array %n, %array
21     mv t5, %array           # Index for the array
22     li t1, 0
23 input_loop:
24     bge t1, %n, input_done  # If index >= N, exit the loop
25     print_string prompt
26     read_int t2             # Read the element
27     sw t2, 0(t5)            # Store in the array
28     addi t5, t5, 4
29     addi t1, t1, 1          # Move to the next element
30     j input_loop
31 input_done:
32     li t1, -4
33     mul t1, t1, %n
34     add t5, t5, t1
35 .end_macro
36
37 .macro print_array %n, %array
38     mv t5, %array           # Address of the start
39     li t1, 0                # Index of the element
40
41 output_loop:
42     lw a0, 0(t5)            # Load element from the array
43     li a7, 1                # System call for printing an integer
44     ecall
45     print_string space       # Print a space
46     addi t5, t5, 4          # Move to the next element
47     addi t1, t1, 1
48     blt t1, %n, output_loop # Condition to continue the loop
49     li t1, -4
50     mul t1, t1, %n
51     add t5, t5, t1
52     print_string new_line
53     print_string new_line
54 .end_macro
55
56 .macro print_new_array
57     print_string prompt0
58 .end_macro
```

Тестирование (test.s)

Файл test.s используется для тестирования функциональности программы:

```
1  .include "io_macros.s"      # Include the macros library
2
3  .data
4      test:       .asciz "_____TEST_____\n\n"
5      test1:      .word 3, -1, 4, 5, -2, 9  # Test array A1
6      expected1:  .asciz "Expected B: 9 -2 5 4 -1 \n"
7
8      test2:      .word -3, -1, -2, 1, 5, 11, 4  # Test array A2
9      expected2:  .asciz "Expected B: 4 11 5 -2 -1 -3 \n"
10
11     test3:      .word 1, 2, 3, 4, 5     # Test array A3
12     expected3:  .asciz "Expected B: 5 4 3 2 \n"
13
```

```
    .align 2
    testB1:     .space 40
    .align 2
    testB2:     .space 40
    .align 2
    testB3:     .space 40

    # Parameters for tests
    testN1:     .word 6             # N for test1
    testN2:     .word 7             # N for test2
    testN3:     .word 5             # N for test3

.text
.globl main

main:
    print_string test
    # Test 1
    lw a0, testN1               # N
    li a3, 0
    la a1, test1                # Address of array A1
    la a2, testB1               # Address of array B
    jal process_array           # Call array processing
    mv s2, a2
    mv s3, a3
    print_string expected1      # Expected result
    print_new_array
    print_array s3, s2          # Output array B


    # Test 2
    lw a0, testN2
    li a3, 0                    # N
    la a1, test2                # Address of array A2
    la a2, testB2               # Address of array B
    jal process_array           # Call array processing
    mv s2, a2
    mv s3, a3
    print_string expected2      # Expected result
    print_new_array
    print_array s3, s2          # Output array B

    # Test 3
    lw a0, testN3               # N
    li a3, 0
    la a1, test3                # Address of array A3
    la a2, testB3               # Address of array B
    jal process_array           # Call array processing
    mv s2, a2
    mv s3, a3
    print_string expected3      # Expected result
    print_new_array
    print_array s3, s2          # Output array B

    # End of program
    li a7, 10                   # System call to exit
    ecall
```

Результаты тестирования

```
_____TEST_____

Expected B: 9 -2 5 4 -1
New array: 9 -2 5 4 -1

Expected B: 4 11 5 -2 -1 -3
New array: 4 11 5 -2 -1 -3

Expected B: 5 4 3 2
New array: 5 4 3 2


-- program is finished running (0) --
```

Работа основной программы (`main.s`)

```
Enter the number of elements N (1-10): 8
Enter array element: -1
Enter array element: -2
Enter array element: -3
Enter array element: 16
Enter array element: -4
Enter array element: -5
Enter array element: 6
Enter array element: 7
New array: 7 6 -5 -4 -3 -2 -1


-- program is finished running (0) --


Reset: reset completed.

Enter the number of elements N (1-10): 5
Enter array element: -1
Enter array element: -2
Enter array element: -3
Enter array element: -4
Enter array element: -5
New array: -5 -4 -3 -2 -1


-- program is finished running (0) --


Enter the number of elements N (1-10): 19

N is out of range
Enter the number of elements N (1-10): 2
Enter array element: 1
Enter array element: 2
New array: 2


-- program is finished running (0) --
```

Заключение

Разработанная программа на ассемблере RISC-V демонстрирует основные принципы работы с массивами, обработки пользовательского ввода и использования модульного программирования. Применение макросов для упрощения операций ввода/вывода повышает читаемость и удобство кода. В будущем можно добавить более сложные проверки на ошибки и улучшить функциональность программы.

Исходный код можно найти по ссылке или qr-коду