ИДЗ-1 Кокорев Артём
Вариант 38

Файлы расположены в публичном репозитории https://github.com/w1sq/ABC

Задание:

Разработать программу, которая вводит одномерный массив A, состоящий из N элементов (значение N вводится при выполненпии программы),после чего формирует из элементов массива A новый массив B по правилам, указанным в варианте, и выводит его. Память под массивы может выделяться статически, на стеке, автоматичеси по выбору разработчика с учетом требований к оценке работы.
При решении задачи необходимо использовать подпрограммы для реализации ввода, вывода и формирования нового массива массива. Допустимы (при необходимости) дополнительные подпрограммы.
Максимальное количество элементов в массиве не должно превышать 10 (ограничение обуславливается вводом данных с клавиатуры). При этом необходимо обрабатывать некорректные значения как для нижней, так и для верхней границ массивов в зависимости от условия задачи.

Сформировать массив B из элементов массива A, которые одновременно имеют четные и отрицательные значения.

Код на 9 баллов:

```
.data
# Выделение памяти для массивов
array_A: .space 40 # 10 элементов * 4 байта
array_B: .space 40 # 10 элементов * 4 байта
array_C: .space 40 # 10 элементов * 4 байта (для правильного массива)

# Сообщения для пользовательского интерфейса
prompt_mode: .string "Выберите режим (1 - ручной ввод, 2 - автотест): "
prompt_size: .string "Введите размер массива (макс. 10): "
prompt_element: .string "Введите элемент массива: "
result_msg: .string "Результирующий массив B: "
generated_msg: .string "Сгенерированный массив A: "
empty_msg: .string "Массив B пуст"
space: .string " "
newline: .string "\n"
error_msg: .string "Ошибка: размер должен быть от 1 до 10\n"
prompt_test_count: .string "Введите количество тестовых массивов: "
prompt_test_size: .string "Введите размер тестового массива #"
colon: .string ": "
correct_msg: .string "Тест пройден успешно\n"
incorrect_msg: .string "Тест не пройден\n"

.text
.globl main

# Макрос для ввода целого числа
.macro read_int(%reg)
li a7, 5
ecall
mv %reg, a0
.end_macro
```

```
# Макрос для вывода целого числа
.macro print_int(%reg)
mv a0, %reg
li a7, 1
ecall
.end_macro

# Макрос для вывода строки
.macro print_string(%label)
la a0, %label
li a7, 4
ecall
.end_macro

# Макрос для генерации случайного числа
.macro random_int(%reg)
li a7, 41
ecall
mv %reg, a0
.end_macro

# Макрос для генерации тестового массива
.macro generate_test_array(%array_addr, %size)
mv t0, %array_addr
li t1, 0
generate_loop:
random_int(t2)
li t3, 301 # Диапазон от 0 до 300
remu t2, t2, t3 # Используем remu для беззнакового деления
addi t2, t2, -150 # Сдвигаем диапазон на -150
sw t2, 0(t0)
addi t0, t0, 4
addi t1, t1, 1
blt t1, %size, generate_loop
.end_macro

# Макрос для вывода массива
.macro print_array(%array_addr, %size)
mv t0, %array_addr
li t1, 0
print_loop:
lw t2, 0(t0)
print_int(t2)
print_string(space)
addi t0, t0, 4
addi t1, t1, 1
blt t1, %size, print_loop
print_string(newline)
.end_macro



# Макрос для генерации правильного массива
.macro generate_correct_array(%src_addr, %src_size, %dst_addr, %dst_size)
mv t0, %src_addr
mv t1, %dst_addr
li t2, 0 # счетчик для исходного массива
mv %dst_size, zero # обнуляем размер результирующего массива
generate_correct_loop:
beq t2, %src_size, generate_correct_done
lw t3, 0(t0)
# Проверка на четность и отрицательность
```

```
li t4, 2
rem t5, t3, t4
bgtz t3, generate_correct_next
bnez t5, generate_correct_next
# Добавление элемента в правильный массив
sw t3, 0(t1)
addi t1, t1, 4
addi %dst_size, %dst_size, 1
generate_correct_next:
addi t0, t0, 4
addi t2, t2, 1
j generate_correct_loop
generate_correct_done:
.end_macro

# Макрос для сравнения массивов
.macro compare_arrays(%arr1, %size1, %arr2, %size2)
# Проверка на пустой массив B
beqz %size1, compare_empty
bne %size1, %size2, compare_fail
mv t0, %arr1
mv t1, %arr2
li t2, 0
compare_loop:
beq t2, %size1, compare_success
lw t3, 0(t0)
lw t4, 0(t1)
bne t3, t4, compare_fail
addi t0, t0, 4
addi t1, t1, 4
addi t2, t2, 1
j compare_loop
compare_empty:
beqz %size2, compare_success
j compare_fail
compare_success:
print_string(correct_msg)
j compare_done
compare_fail:
print_string(incorrect_msg)
compare_done:
.end_macro

main:
# Выбор режима работы
print_string(prompt_mode)
read_int(t0)

li t1, 1
beq t0, t1, manual_input
li t1, 2
beq t0, t1, auto_test
# Если введено не 1 и не 2, повторяем запрос
j main

manual_input:
# Ввод размера массива с проверкой
li s0, 0 # Инициализируем s0 нулем
input_size_loop:
print_string(prompt_size)
read_int(s0)
li t0, 1
blt s0, t0, input_error # Если меньше 1, выводим ошибку
```

```
li t0, 10
bgt s0, t0, input_error # Если больше 10, выводим ошибку
j input_elements

input_error:
print_string(error_msg)
j input_size_loop

input_elements:
la s1, array_A # s1 = адрес массива A

# Ввод элементов массива A
mv t1, zero # t1 = счетчик
input_loop:
beq t1, s0, input_done
print_string(prompt_element)
read_int(t2)
sw t2, 0(s1)
addi s1, s1, 4
addi t1, t1, 1
j input_loop

input_done:
la s1, array_A # Восстановление адреса начала массива A
j process_array_manual

auto_test:
# Ввод количества тестовых массивов
print_string(prompt_test_count)
read_int(s4) # s4 = количество тестовых массивов

li s5, 1 # s5 = счетчик текущего тестового массива
auto_test_loop:
bgt s5, s4, exit # Если все тесты выполнены, завершаем программу

# Ввод размера текущего тестового массива
print_string(prompt_test_size)
print_int(s5)
print_string(colon)
read_int(s0)
# Проверка размера массива
li t0, 1
blt s0, t0, auto_test_error
li t0, 10
bgt s0, t0, auto_test_error
j generate_test

auto_test_error:
print_string(error_msg)
j auto_test_loop

generate_test:
# Генерация тестового массива
la s1, array_A # s1 = адрес массива A
generate_test_array(s1, s0)

# Вывод сгенерированного массива
print_string(generated_msg)
print_array(s1, s0)

# Генерация правильного массива
```

```
la s6, array_C # s6 = адрес правильного массива C
mv s7, zero # s7 = размер правильного массива C
generate_correct_array(s1, s0, s6, s7)

# Обработка массива
j process_array



process_array_manual:
la s2, array_B # s2 = адрес массива B
mv s3, zero # s3 = размер массива B (сохраняемый регистр)

mv t1, zero # t1 = счетчик для A
process_loop_manual:
beq t1, s0, process_done_manual
lw t3, 0(s1) # Загрузка элемента из A
# Проверка на четность и отрицательность
li t4, 2
rem t5, t3, t4 # t5 = остаток от деления на 2
bgtz t3, next_element_manual # Пропуск, если положительное
bnez t5, next_element_manual # Пропуск, если нечетное
# Добавление элемента в B
sw t3, 0(s2)
addi s2, s2, 4
addi s3, s3, 1 # Увеличиваем счетчик элементов B

next_element_manual:
addi s1, s1, 4
addi t1, t1, 1
j process_loop_manual

process_done_manual:
# Вывод результата
print_string(result_msg)
la s2, array_B # Восстановление адреса начала массива B
beqz s3, print_empty_manual # Если B пуст, выводим сообщение
print_array(s2, s3) # Используем s3 вместо t2
j exit

print_empty_manual:
print_string(empty_msg)
print_string(newline)
j exit

process_array:
la s2, array_B # s2 = адрес массива B
mv s3, zero # s3 = размер массива B (сохраняемый регистр)

mv t1, zero # t1 = счетчик для A
process_loop:
beq t1, s0, process_done
lw t3, 0(s1) # Загрузка элемента из A
# Проверка на четность и отрицательность
li t4, 2
rem t5, t3, t4 # t5 = остаток от деления на 2
bgtz t3, next_element # Пропуск, если положительное
bnez t5, next_element # Пропуск, если нечетное
# Добавление элемента в B
sw t3, 0(s2)
addi s2, s2, 4
addi s3, s3, 1 # Увеличиваем счетчик элементов B
```

```
next_element:
addi s1, s1, 4
addi t1, t1, 1
j process_loop

process_done:
# Вывод результата
print_string(result_msg)
la s2, array_B # Восстановление адреса начала массива B
beqz s3, print_empty # Если B пуст, выводим сообщение
print_array(s2, s3) # Используем s3 вместо t2
j compare_result

print_empty:
print_string(empty_msg)
print_string(newline)
compare_result:
# Сравнение с правильным массивом
compare_arrays(s2, s3, s6, s7)
# Переход к следующему тестовому массиву
addi s5, s5, 1
j auto_test_loop

exit:
# Завершение программы
li a7, 10
ecall
```

Описание программы:

Программа поддерживает два режима работы, контролируемые вводом с клавиатуры
При вводе 1 – происходит ручной ввод одного массива с запрашиваемым размером
При вводе 2 – происходит автоматическое тестирование. Запрашивается количество тестов(количество проверяется лишь на положительность) и затем размер для каждого из n массивов, с последующей генерацией и проверкой массивов.
Честно признаюсь, что не до конца понял формат тестирования, т.к. в задании лишь указано *генерации тестовых массивов,* без указания формата работы этого макроса. Поэтому вместо ручного прописывания массивов и их корректных обработок, я генерирую массив из случайных чисел(приводя их в диапазон от -150 до 150), паралельно заполняя корректный массив.
После их генерации происходит отработка самой программы и сверка массива-результата макроса и основной программы с последующим выводом вердикта.

Результаты запуска:

```
Выберите режим (1 - ручной ввод, 2 - автотест): 1
Введите размер массива (макс. 10): 3
Введите элемент массива: 60
Введите элемент массива: 0
Введите элемент массива: -46
Результирующий массив B: 0 -46

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

```
Выберите режим (1 - ручной ввод, 2 - автотест): 2
Введите количество тестовых массивов: 2
Введите размер тестового массива #1: 3
Сгенерированный массив A: -57 -16 36
Результирующий массив B: -16
Тест пройден успешно
Введите размер тестового массива #2: 9
Сгенерированный массив A: 7 68 -146 65 -41 -59 -81 -30 79
Результирующий массив B: -146 -30
Тест пройден успешно

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