Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

**”Memory and data application design”**

**Вариант №7**

|  |  |  |  |
| --- | --- | --- | --- |
| Преподаватель |  | подпись, дата | Яблонский А.П. |
| Студент КИ19-09Б | 031940750  номер зач. книжки | подпись, дата | Кудрявцев Н.М. |

Красноярск 2022 г.

**ВВЕДЕНИЕ**

**Цель работы:**

Операции записи и чтения в память данных, операции сложения и вычитания над регистрами, операции сравнение регистров и констант, организация циклов.

**Задание:**

1. Расположить исходные массивы A и B в памяти данных последовательно, с начального адреса – $0060.
2. Заполнить исходные массивы произвольными значениями.
3. Выполнить требуемую операцию над массивами А и В, и результат записать в память в массив С, расположенный по указанному адресу.
4. В массиве С найти минимальный/максимальный элемент и записать его про требуемому адресу.
5. Отсортировать массив С по возрастанию/убыванию.

Таблица 1 ─ Вариант задания.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Размер массивов A и B** | **Операция** | **Размер массива C** | **Адрес массива C** | **Поиск**  **Элемента** | **Адрес**  **Элемента** | **Сортировка** |
| 10 | Пересечение | 10 | 0x7B | Min | 0x8C | Убывание |

**Команды:**

Справка по Ассемблеру для Atmel AVR: <https://dfe.karelia.ru/koi/posob/avrlab/avrasm-rus.htm>

Ldi r31, 22 – Загрузить константу (22 записывается в регистр r31);

Mov r19, r0 – Значение регистра r0 копируется в регистр r19;

Cp r28, r27 – Сравнить значения регистров r28 и r27;

Brsh LARGER – Перейти к метке “LARGER” если равно или больше;

Rjmp END1 – Относительный переход к метке “END1”;

Nop – Нет операции;

SER r20 – Устанавливает все биты регистра r20 в “1”;

.def c1 = r16 – Назначить регистру r16 символическое имя c1;

.set AB\_size = 10 – Присваивает имени некоторое значение;

.dseg – Определяет начало сегмента данных;

.org AB\_addr – Переходим на ячейку памяти по адресу, равному заданной величине;

arr\_A: .byte AB\_size – Резервирует байты в ОЗУ;

.cseg – Определяет начало программного сегмента;

st x+, r18 – Косвенное сохранение с пост-инкрементом;

brne filling – Перейти если не равно;

breq cross\_A\_store – Перейти если равно;

ld r18, z+ – Косвенная загрузка с пост-инкрементом;

ld r18, -z – Косвенная загрузка с пре-декрементом;

sts k, r19 – Прямое сохранение значения из r19 в отведённую ячейку памяти.

**Ход работы:**

Ссылка на задание: [тут](https://e.sfu-kras.ru/pluginfile.php/2646245/mod_resource/content/1/Lab%202%20Memory%20and%20data%20application%20design.pdf)

Ссылки на лекцию: [тут](https://e.sfu-kras.ru/pluginfile.php/2646439/mod_resource/content/2/Lecture%203%20%D0%A1%D0%BD%D0%9A%20%D0%B8%20%D0%A1%D0%B2%D0%9A%20AVR%20%D0%B0%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80%20%D1%87%D0%B0%D1%81%D1%82%D1%8C%201.pdf) и [тут](https://e.sfu-kras.ru/pluginfile.php/2646405/mod_resource/content/3/%D0%9B%D0%B5%D0%BA%D1%86%D0%B8%D1%8F-3%20%D0%90%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80%20%D0%A7%D0%B0%D1%81%D1%82%D1%8C%202.pdf)

**Код программы**

// ВАРИАНТ №7

//

// Размер массивов A и B: 10,

// Операция: Пересечение,

// Размер массива C: 10

// Адрес массива C: 0x7B

// Поиск: MIN Элемента

// Адрес Элемента: 0x8C

// Сортировка: Убывание

// 1) Расположить исходные массивы последовательно, с начального адреса – $0060;

// 2) Заполнить исходные массивы произвольными значениями.

// 3) Выполнить операцию ПЕРЕСЕЧЕНИЯ над массивами A и B и результат записать в память в массив C, расположенный по адресу 0x7B.

// 4) В массиве C найти МИНИМАЛЬНЫЙ элемент и записать его по адресу 0x8C.

// 5) Отсортировать массив C по УБЫВАНИЮ.

// Инициализируем несколько "переменных"

.def counter = r16 ; Счетчик

.def alt\_counter = r17 ; Второй счетчик

.def a\_current = r18 ; Элемент массива А

.def b\_current = r19 ; Элемент массива B

.def c\_current = r20 ; Элемент массива C

.def bool = r21 ; Логическая переменная

.def mx = r22 ; Логическая переменная

.def counterWorkSize = r23 ; Счетчик заполненной зоны в массиве

.set sizeAB = 10 ; Размер массивов A и B

.set sizeC = 10 ; Размер массивов C

.set false = 0 ; Логический ноль

.set true = 1 ; Логическая единица

// Сегмент данных, работа с памятью

.dseg ; Указатель на сегмент данных

.org 0x60 ; Начальный адрес массивов

arr\_a: .byte sizeAB ; Выделяем sizeAB байт на массив А

arr\_b: .byte sizeAB ; Выделяем sizeAB байт на массив В

.org 0x7B

arr\_c: .byte sizeC ; Выделяем sizeС байт на массив С

.org 0x8C

min\_el: .byte 1 ; Выделим 1 байт под максимальный элемент

// Сегмент кода, выполнение задания

.cseg ; Входим в сегмент кода

// Формируем регистровые пары

ldi xl, low(arr\_c) ; X

ldi xh, high(arr\_c)

ldi yl, low(arr\_b) ; Y

ldi yh, high(arr\_b)

ldi zl, low(arr\_a) ; Z

ldi zh, high(arr\_a)

// Заполнение массивов A и B

// A = [ 9,10,11,12,13,14,15,16,17,18]

// B = [15,14,13,12,11,10, 9, 8, 7, 6]

// Общие элементы: 9, 10, 11, 12, 13, 14, 15

ldi a\_current, 9 ; Переменная для заполнения массива a

ldi counter, sizeAB ; Устанавливаем размер счетчика

MASA: ; Заполняем массив a

st z+, a\_current ; Сохранение a\_current в массив В и инкремент указателя

inc a\_current

dec counter

brne MASA ; Условия цикл заполнения массива a

ldi b\_current, 15 ; Переменная для заполнения массива B

ldi counter, sizeAB ; Устанавливаем размер счетчика

MASB: ; Заполняем массив B

st y+, b\_current ; Сохранение b\_current в массив В и инкремент указателя

dec b\_current

dec counter

brne MASB ; Условия цикл заполнения массива В

// ПЕРЕСЕЧЕНИЕ.

// Массив C должен содержать следующие числа:

// C = [0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F, 0x00, 0x00, 0x00].

// Просматриваем массив A на элементы пересечения с элементами массива B

// Повторно формируем регистровые пары X и Y, т.к. работали с их адресами

ldi zl, low(arr\_a) ; Z

ldi zh, high(arr\_a)

ldi alt\_counter, sizeAB

ldi counterWorkSize, 0

FILL1AB:

ld a\_current, z+ ; a\_current=z, z=z+1

ldi bool, false ; bool = true

ldi yl, low(arr\_b) ; Y

ldi yh, high(arr\_b)

ldi counter, sizeAB

FILL2AB:

ld b\_current, y+ ; b\_current=y, y=y+1

cp b\_current, a\_current ; Сравнить b\_current с a\_current

brne RAZN1 ; Перейти, если b\_current не равно a\_current

ldi bool, true ; Иначе bool == true

inc counterWorkSize

RAZN1:

dec counter

brne FILL2AB ; Условия цикл нахождения элемента a\_current в В

cpi bool, true ; Сравнить bool с false (true)

brne RAZN2 ; Перейти, если bool не равно false

st x+, a\_current ; x = a\_current, x = x + 1

RAZN2:

dec alt\_counter ; Декремент счетчика №2

brne FILL1AB

// Поиск МИНИМАЛЬНОГО элемента.

// По адресу 0x8C должен записаться 0x00 (пустой элемент), т.к. размер массива C (10)

// больше найденного вектора пересечения (7),

// и минимальным элементом обозначится пустой элемент (0x00).

// Формируем регистровую пару Z

ldi xl, low(arr\_c) ; X

ldi xh, high(arr\_c)

ld mx,x ; Устанавливаем значение первого элемента

mov counter, counterWorkSize

Find\_min:

ld c\_current, x+ ; сi=x, x=x+1

cp mx, c\_current ; Сравнить c\_current с mx

BRLO nebol ; Перейти если c\_current меньше mx

mov mx, c\_current ; Иначе mx = c\_current

nebol:

dec counter

brne Find\_min

ldi yl, low(min\_el) ; Y

ldi yh, high(min\_el)

st y, mx

// Сортировка по УБЫВАНИЮ

// После сортировки получится следующий массив:

// C = [0x0F, 0x0E, 0x0D, 0x0C, 0x0B, 0x0A, 0x09, 0x00, 0x00, 0x00]

// 3 нуля в конце - пустые элементы из-за размера массива C = 10 элементов, где 7 не нулевых из них составляют вектор пересечения массивов A и B

SORT:

ldi xl, low(arr\_c) ; X

ldi xh, high(arr\_c)

ldi bool, false ; bool = false

mov counter, counterWorkSize

SORT2:

ld c\_current, x+ ; сi=x, x=x+1

ld mx, x ; mx = x

cp mx, c\_current ; Сравнить mx с c\_current

BRLO SWIP ; Перейти если меньше (больше brsh)

ldi bool, true ; bool = true

mov a\_current, c\_current ; a\_current = c\_current//

st -x, mx ; x=x-1, x = mx

ld r13, x+ ; r13 не нужен, быстрый способ увеличить x

st x, a\_current ; x = a\_current

SWIP:

dec counter ; Декремент счетчика

brne SORT2

cpi bool, true ; Сравнить bool с true

breq SORT ; Перейти если bool равно true

nop

// Бесконечный цикл

inf:

rjmp inf

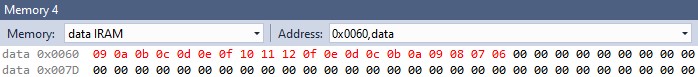


Рисунок 1 – Массивы А и В (по адресу 0x60).

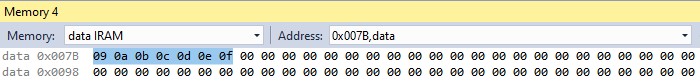


Рисунок 2 – Массив C (по адресу 0x7B).

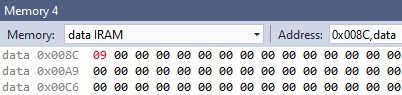


Рисунок 3 – Минимальный элемент по адресу 0x8C (0x00 или же “0”).

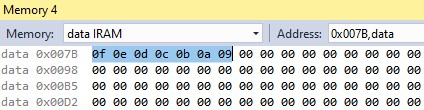


Рисунок 4 – Массив С, отсортированный по убыванию.