Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт инженерной физики и радиоэлектроники

«Радиоэлектронная техника информационных систем»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1**

Изучение языка ассемблера, ядра Cortex-M3 и среды разработки Keil MDK

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Преподаватель | |  |  | \_\_\_\_\_\_\_\_\_\_\_\_ | Т. Н. Батурин |
|  | |  |  | подпись, дата |  |
| Студент | РФ19-32Б | | | \_\_\_\_\_\_\_\_\_\_\_\_ | И. А. Негодеева |
|  |  | | | подпись, дата |  |

Красноярск 2022

# 1 Задание лабораторной работы (вариант 5)

Написать программу расчета среднего арифметического значения отрицательных элементов в одномерном массиве. Определить минимальный элемент в одномерном массиве.

# 2 Выполнение лабораторной работы

1. Блок схема выполнения лабораторной работы представлена на рис. 1.



Рисунок – Блок схема программы

2. Проинициализировать массив, состоящий из однобайтных чисел и двух переменных, в которых будут храниться результаты вычисления (рис. 2). Объявление массива констант происходит в сегменте данных во FLASH памяти, так как не требуется изменять значения массива. Объявление переменных, в которых будут сохраняться результаты выполнения задания, объявляется в сегменте данных в ОЗУ (RAM), так как это значение необходимо менять входе выполнения программы.

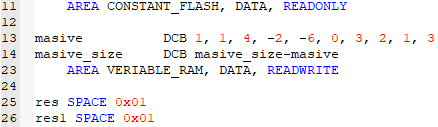


Рисунок – Инициализация массива и переменных

3. Обнулить переменные i, n, nm, sn.

4. Проверить условие i < massive\_size. Организация условия включает в себя использование меток, инструкции CMP и инструкции условного перехода.

5. Если условие «истина», то переходим к проверке элемента массива (mas[i] < 0). Для обращения к массиву необходимо с помощью инструкции LDRB узнать адрес первого элемента массива. Затем считать значение из памяти в регистр и снова организовать условие. Если «ложь» устанавливаем переход на конец программы.

6. Далее просуммировать отрицательные элементы массива (sn = sn + mas[i]). Рассчитать количество отрицательных элементов массива (n).

7. Проверить условие mas[i] < mas[nm]. Если условие «истина», то присваиваем элементу nm значение i (nm = i). Если «ложь» устанавливаем переход на конец программы.

8. Конечным кодом программы является организация цикла для обработки всех элементов массива и при выходе из цикла запись результатов в переменных res и res1 (рис. 3).



Рисунок – Результат выполнения лабораторной работы

**ПРИЛОЖЕНИЕ А**

**Код программы**

GET main.s

;�"иректива PRESERVE8 Указывает, что текущий файл требует или сохраняет восьмибайтовое выранивание стека

PRESERVE8

;�"иректива THUMB инструктирует ассемблер интерпретировать последующие инструкции,

;как инструкции Thumb используя синтаксис UAL.

THUMB

AREA CONSTANT\_FLASH, DATA, READONLY

masive DCB 1, 1, 4, -2, -6, 0, 3, 2, 1, 3

masive\_size DCB masive\_size-masive

;masive1 DCB 1, 1, 4, -3, 2, 0, 3, 2, 1, 3, 2

;masive\_size1 DCB masive\_size1-masive1

;masive2 DCB 1, 1, 4, -3, 2, 0, 3, 2, 1, 3, 3, 5

;masive\_size2 DCB masive\_size2-masive2

;masive3 DCB 1, 1, 4, -3, 2, 0, 3, 2, 1, 3, 5, 6, 1, 1, 4, -3, 2, 0, 3, 2, 1, 3, 5, 6, 1

;masive\_size3 DCB masive\_size3-masive3

; Объявляем сегмент переменных

AREA VERIABLE\_RAM, DATA, READWRITE

res SPACE 0x01

res1 SPACE 0x01

;Объявляем стартовый сегмент кода

AREA **RESET**, CODE, READONLY

; Таблица векторов прерываний

DCD STACK\_TOP ; Указатель на вершину стека

DCD startup ; �'ектор сброса

;Точка входа

ENTRY

;Startup код

startup PROC ; Начало startup кода

; �нициализация МК

; �'ызов подпрограмм

**PUSH** {R0-R12} ; Сохранение регистров в памяти

; Подготовка регистров

LDR R0, =masive ; Указатель на адрес массива

LDR R1, =masive\_size ; Указатель на адрес размера массива

LDRB R1, [R1] ; �' R1 значение размера массива (число)

BL var\_5 ; �'ызов ПП, вызывающей остальные ПП

**POP** {R0-R12} ; �'озврат регистров

B .

ENDP ; Конец программы

var\_5 PROC ; ПП, вызывающая остальные ПП

**PUSH** {LR} ; Сохранение адреса возврата

; �'ызов подпрограмм

**PUSH** {R0-R12} ; Сохранение регистров в памяти

; Переход в подпрограмму avg\_calc для расчета средн. арифм.

BL avg\_calc ; Переход к ПП для расчета средн. арифм.

; Обработка результата

LDR R7, =res ; �-апись адреса переменной "res" в R7

STR R0, [R7] ; Сохранить содержимое регистра R0 в памяти по адресу в регистре R7

; �'осстановление контекста

**POP** {R0-R12} ; �'озврат регистров

; �'ызов подпрограмм

**PUSH** {R0-R12} ; Сохранение регистров в памяти

; Переход в подпрограмму find\_min для поиска минимального значения

BL find\_min ; Переход к ПП для поиска минимального значения

; Обработка результатов

LDR R6, =res1 ; �-апись адреса переменной "res1" в R6

STR R0, [R6] ; Сохранить содержимое регистра R0 в памяти по адресу в регистре R6

; �'осстановление контекста

**POP** {R0-R12} ; �'озврат регистров

**POP** {LR} ; �'озврат адреса для выхода из ПП

BX LR

avg\_calc PROC ; ПП для расчета средн. арифм.

MOV R2, #NULL ; R2 = "i" = 0 - элемент массива

MOV R3, #NULL ; R3 = 0 - обнуление регистра R3

MOV R4, #NULL ; R4 = "sn" - 0 - сумма отрицательных элементов

MOV R5, #NULL ; R5 = "n" = 0 - кол-во отрицательных элементов

CVTR ; Метка CVTR

CMP R2, R1 ; Проверка "i" с "masive\_size"

BLT CAJ ; Если i < masive\_size, переход в "CAJ"

B SRZN ; �'езусловный переход в "SRZN" - выполняется только, когда не произошел переход в "CAJ"

CAJ ; Метка CAJ

; �-апись в регистр "R3" значения по адресу сохран�'нном в "R2"

LDRB R3, [R0, R2] ; R3 = mas[i]

SXTB R3, R3 ; Расширение знака

CMP R3, #NULL ; Сравнить mas[i] с 0

BLT JNT ; Если mas[i] < 0, то переходим в JNT

B OUTXX ; �'езусловный переход в OUTXX

JNT ; Метка JNT

SXTB R4, R4 ; Расширение знака

ADD R4, R3 ; sn = sn + mas[i]

ADD R5, #ONE ; n = n + 1

OUTXX ; Метка OUTXX

ADD R2, #ONE ; i = i + 1

B CVTR ; Переход в начало подпрограммы

; Среднее арифметическое

CMP R5, #NULL ; Сравнение n с нулем

BNE SRZN ; Если не равно нулю, то переход в SRZN

B OUTEX ; �наче переход в OUTEX

SRZN ; Метка SRZN

SDIV R4, R4, R5 ; Среднее арифм.

; �-аписать значение результата в регистр "R0"

OUTEX ; Метка OUTEX

SXTB R0, R0 ; Расширение знака

MOV R0, R4 ; �-апись R4 в R0

BX LR ; �'озвращение по адресу, сохраненном в LR

ENDP ; Конец подпрограммы

;-------------------------------------------------------------------------------

find\_min PROC ; ПП для поиска минимального значения

MOV R2, #NULL ; R2 = "i" = 0 - элемент массива

MOV R3, #NULL ; R3 = 0 - обнуление регистра R3

MOV R4, #NULL ; R4 = "nm" = 0 - наименьший элемент

KYTR ; Метка KYTR

CMP R2, R1 ; Проверка "i" c "masive\_size"

BLT CAT ; Если i < MAX\_size, переход в "CAT"

B OUTEXX ; �'езусловный переход в "OUTEXX" - выполняется только, когда не произошел переход в "CAT"

CAT ; Метка CAT

; �-апись в регистр "R3" значения по адресу сохран�'нном в "R2"

LDRB R3, [R0, R2] ; R3 = mas[i]

SXTB R3, R3 ; Расширение знака

CMP R3, #NULL ; Сравнить mas[i] с 0

BLT LKJ ; Если mas[i] < 0, то переходим в LKJ

B OUTXXX ; �'езусловный переход в OUTXXX

LKJ ; Метка LKJ

CMP R3, R4 ; Сравнение mas[i] с nm

BLE SRAW ; Если mas[i] < nm, то переход в SRAW

B OUTXXX ; �наче переход в OUTXXX

SRAW ; Метка SRAW

MOV R4, R3 ; �-апись R4 = R3, присвоили R4 Титул элемента с минимальным значением

OUTXXX ; Метка OUTXX

ADD R2, #ONE ; i = i + 1

B KYTR ; Переход в начало подпрограммы

OUTEXX ; Метка OUTEXX

SXTB R0, R0

MOV R0, R4 ; �-апись R4 в R0

BX LR ; �'озвращение по адресу, сохраненном в LR

ENDP ; Конец ПП

;-------------------------------------------------------------------------------

END ; Конец файла