Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт инженерной физики и радиоэлектроники

Кафедра «Радиоэлектронная техника информационных систем»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**

Дисциплина «Цифровые устройства и микропроцессоры»

Лабораторная работа №2

Изучение функций и процедур

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Руководитель |  |  |  |  | Батурин Т. Н. |
|  |  |  | подпись, дата |  | инициалы, фамилия |
|  |  |  |  |  |  |
| Студент | РФ19-32Б 051940816 |  |  |  | Сивов Н. О. |
|  | номер группы, зачетной книжки |  | подпись, дата |  | инициалы, фамилия |

Красноярск 2022

1. **Цель работы**

Выполнить задание лабораторной №1 на языке Си используя функции и указатели. Переписать программу, написанную в лабораторной работе №1 с помощью процедур языка ассемблера. Вызвать процедуры, написанные на ассемблере в коде на языке Си. Сравнить результат выполнения функций на языке Си и на языке ассемблера.

1. **Ход работы**

В одномерном массиве поменять местами максимальный и минимальный элементы.

Для выполнения работы в программной среде Keil MDK были созданы три файла: startup.s, main.c (отвечает за исполнение кода на языке СИ), main\_1.s (отвечает за исполнение кода на языке ассемблера). Листинг файлов приведен в приложениях А-В.

На языке СИ код реализуется с помощью объявления переменных, указателей и прототипов функций, затем описания этих функций и входных аргументов. Функции реализованы с помощью циклов for, while и условных операторов if.

На языке СИ код реализуется с помощью объявления переменных, указателей и прототипов функций, затем описания этих функций и входных аргументов. Функции реализованы с помощью циклов for, while и условных операторов if. После выполнения задания производится переход в файл main\_1.s с экспортированием функций и входных аргументов (адрес изначального массива и его размер).

На языке ассемблера код реализуется также, как в первой лабораторной работе, за исключением того, что в ассемблер экспортируются функции с входными аргументами.

1. **Вывод**

Результаты сравнения времени вычисления представлены в таблице 1.

Таблица 1 – Сравнение результатов вычисления по времени

|  |  |  |
| --- | --- | --- |
| Уровень оптимизации (си) | Время вычисления, мкс | |
| Си | Ассемблер |
| 0 | 30,2 | 27,72 |
| 1 | 30,33 | 27,59 |
| 2 | 14,08 | 25,5 |
| 3 | 13,92 | 25,5 |

**ПРИЛОЖЕНИЕ А**

Листинг 1. Файл startup.s

Stack\_Size EQU 0x400

AREA STACK, NOINIT, READWRITE, ALIGN=3

Stack\_Mem SPACE Stack\_Size

\_\_initial\_sp

; <h> Heap Configuration

; <o> Heap Size (in Bytes) <0x0-0xFFFFFFFF:8>

; </h>

Heap\_Size EQU 0x200

AREA HEAP, NOINIT, READWRITE, ALIGN=3

\_\_heap\_base

Heap\_Mem SPACE Heap\_Size

\_\_heap\_limit

PRESERVE8

THUMB

; Vector Table Mapped to Address 0 at Reset

AREA RESET, DATA, READONLY

DCD \_\_initial\_sp ; Top of Stack

DCD Reset\_Handler ; Reset Handler

AREA |.text|, CODE, READONLY

; Reset handler

Reset\_Handler PROC

IMPORT \_\_main

LDR R0, =\_\_main

BX R0

; B main\_asm

ENDP

IMPORT \_\_use\_two\_region\_memory

EXPORT \_\_user\_initial\_stackheap

\_\_user\_initial\_stackheap

LDR R0, = Heap\_Mem

LDR R1, =(Stack\_Mem + Stack\_Size)

LDR R2, = (Heap\_Mem + Heap\_Size)

LDR R3, = Stack\_Mem

BX LR

ALIGN

END

**ПРИЛОЖЕНИЕ Б**

Листинг 2. Файл main.c

#include <stdio.h>

#include <stdint.h>

/\*

ЗАДАНИЕ:

В одномерном массиве поменять местами максимальный и минимальный элементы.

\*/

// Инициализируем первичный массив с числами

int32\_t array\_original [] = {45, 63, 12, 54, 9, 36, 19, 32}; //2D, 3F, 0C, 36, 09, 24, 13, 20

// Инициализируем указатель на новый массив. Начальное значение NULL

int32\_t\* array\_result = NULL;

// Объявляем прототип функции, которая будет выполнять задание

int32\_t\* array\_processing (int32\_t\* array\_pointer, size\_t array\_size);

// Объявляем прототипы функций, которые будут вычислять минимум и максимум

int32\_t array\_min (int32\_t\* array\_pointer, size\_t array\_size);

int32\_t array\_max (int32\_t\* array\_pointer, size\_t array\_size);

//Экспортируем функцию из main.s написанную на ассемблере.

extern void array\_processing\_asm (int32\_t\* array\_pointer, size\_t array\_size, int32\_t\* new\_array\_pointer);

extern int32\_t array\_minmax\_asm (int32\_t\* array\_pointer, size\_t array\_size);

extern int32\_t array\_change\_asm (int32\_t\* array\_pointer, size\_t array\_size);

extern int32\_t array\_copy\_asm (int32\_t\* array\_pointer, size\_t array\_size);

// Объявляем пустой массив для передачи его адреса в функцию array\_processing\_asm

int32\_t new\_array [sizeof(array\_original)/sizeof(int32\_t)]= {0};

int main (void)

{

array\_result = array\_processing(array\_original, sizeof(array\_original)/sizeof(int32\_t));

array\_processing\_asm(array\_original, sizeof(array\_original)/sizeof(int32\_t), new\_array);

while (1)

{

}

return 0;

}

int32\_t\* array\_processing (int32\_t\* array\_pointer, size\_t array\_size)

{

int32\_t \*new\_array = NULL;

int32\_t min\_index = 0;

int32\_t max\_index = 0;

new\_array = (int32\_t\*) (malloc(array\_size));

min\_index = array\_min(array\_pointer, array\_size);

max\_index = array\_max(array\_pointer, array\_size);

for (uint32\_t i = 0; i < array\_size; i++)

{

\*(new\_array + i) = \*(array\_pointer + i);

}

\*(new\_array + max\_index) = \*(array\_pointer + min\_index);

\*(new\_array + min\_index) = \*(array\_pointer + max\_index);

return new\_array;

}

int32\_t array\_min (int32\_t\* array\_pointer, size\_t array\_size)

{

int32\_t min\_index = 0;

int32\_t min = \*array\_pointer;

int32\_t i = 0;

while (i < array\_size)

{

if (\*(array\_pointer + i) < min) {

min = \*(array\_pointer+i);

min\_index = i;

}

i++;

}

return min\_index;

}

int32\_t array\_max (int32\_t\* array\_pointer, size\_t array\_size)

{

int32\_t max\_index = 0;

int32\_t max = \*array\_pointer;

int32\_t i = 0;

while (i < array\_size)

{

if (\*(array\_pointer + i) > max) {

max = \*(array\_pointer+i);

max\_index = i;

}

i++;

}

return max\_index;

}

**ПРИЛОЖЕНИЕ В**

Листинг 3. Файл main\_1.s

AREA |.text|, CODE, READONLY

array\_processing\_asm PROC

EXPORT array\_processing\_asm

PUSH {R0-R11, LR}

MOV R4, #4

MUL R1, R4

BL array\_copy\_asm

BL array\_minmax\_asm

BL array\_change\_asm

POP {R0-R11, PC}

ENDP

; функция копирования массива в оперативную память

array\_copy\_asm PROC

EXPORT array\_copy\_asm

PUSH {R4-R11, LR}

MOV R4, #0

exactly\_COPY

LDRB R5, [R0, R4]

STRB R5, [R2, R4]

ADD R4, #4

CMP R4, R1

BLT exactly\_COPY

POP {R4-R11, PC}

ENDP

; функция поиска минимума и максимума

array\_minmax\_asm PROC

EXPORT array\_minmax\_asm

PUSH {R4-R11, LR}

MOV R4, #0

LDRB R5, [R2, #0]

LDRB R6, [R2, #0]

CHECK

ADD R4, #4

CMP R4, R1

BLT CHECK\_MAX

MOV R0, R8

MOV R1, R9

POP {R4-R11, PC}

CHECK\_MAX

LDRB R7, [R2, R4]

CMP R7, R6

BGT CALC\_MAX

B CHECK\_MIN

CALC\_MAX

MOV R6, R7

MOV R8, R4

B CHECK

CHECK\_MIN

CMP R7, R5

BLT CALC\_MIN

B CHECK

CALC\_MIN

MOV R5, R7

MOV R9, R4

B CHECK

ENDP

; функция перемены местами

array\_change\_asm PROC

EXPORT array\_change\_asm

PUSH {R4-R11, LR}

LDRB R4, [R2, R0]

LDRB R5, [R2, R1]

STRB R4, [R2, R1]

STRB R5, [R2, R0]

POP {R4-R11, PC}

ENDP

END