**Министерство науки и высшего образования Российской̆ Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования  
«Московский̆ государственный̆ технический̆ университет   
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)**

|  |  |
| --- | --- |
| ФАКУЛЬТЕТ | «ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ» (ИУ) |
| КАФЕДРА | «ПРОЕКТИРОВАНИЕ И ТЕХНОЛОГИЯ ПРОИЗВОДСТВА ЭА» (ИУ4) |

**Домашнее задание**

по дисциплине:

**ПРОЕКТИРОВАНИЕ МИКРОПРОЦЕССОРНЫХ СИСТЕМ**

**Вариант 1**

**Группа:** ИУ4-72Б

**Студент:** Дычаковская А.А.

**Преподаватель:** Леонидов В.В.

ИУ4-72Б

Дычаковская А.А.

Леонидов В.В.

Москва, 2023 г.

**Преподаватель:**

ИУ4-72Б

Дычаковская А.А.

Леонидов В.В.

Москва, 2023 г.

Москва, 2023 г.

1. Задание

Задача №1

Разработать на языке Assembler для ядра Cortex-M3 подпрограмму расчёта контрольной суммы массива 8-разрядных беззнаковых чисел по алгоритму CRC8. Указатель на строку хранится в регистре R0, количество элементов массива – в R1, результат записать в регистр R2.

В качестве микроконтроллера (МК) использовать STM32F103RBT6. Все используемые в подпрограмме регистры общего назначения необходимо сохранить в стек при входе в подпрограмму и выгрузить из стека обратно при выходе из неё.

2. Алгоритмы работы

Алгоритм работы программы расчета контрольной суммы по алгоритму CRC8.

Инициализация стека. В регистр r0 указываем адрес ячейки памяти, начиная с которой будем хранить элементы массива. В регистре r1 задаем количество элементов массива. В регистр r3 записываем непосредственно элемент массива, после чего записываем этот элемент по адресу, содержащийся в регистре r0, и инкрементируем адрес на 1, чтобы записать следующий элемент. После записи массива возвращаем значение r0 в исходное. Переходим в подпрограмму *crc8*.

*crc8:*

В стек записываем значения регистров r1- r4. Обнуляем регистр r2, в котором будет содержаться результат работы программы. К значению регистра r1 добавляем значение, содержащееся в r0, таким образом в r0 будет храниться адрес начала области памяти, в которую будет записан массив, а в r3 – конец этой области памяти. Переход в подпрограмму *array\_proc*.

*array\_proc:*

Подпрограмма выполняется, пока не будут обработаны все элементы массива. Загружаем значение, содержащееся по адресу r0 в r3 (в данном случае – адрес памяти, по которому находится первый элемент массива) и инкрементируем значение адреса. Выполняется операция XOR между r2 и r3 (согласно методу работы алгоритма CRC8). Обнуляем счетчик r4 обработанных битов регистра. Переход в подпрограмму *element\_proc.*

*element\_proc:*

Подпрограмма выполняется, пока не будут обработаны все 8 битов. Сначала происходит логический сдвиг значения r2 влево, согласно методу работы алгоритма CRC8, далее производится проверка, что число в регистре r2 по объему не превысило байт. В случае, если размер превышает байт, то старший бит числа обнулится и произойдет инверсия трех младших битов числа (согласно алгоритму CRC8). Если изначально число не занимало целый байт, то идет переход в подпрограмму *element\_proc.*

*end\_proc:*

Увеличиваем счетчик битов. Если не достигли 8, то переходим в подпрограмму *element\_proc* и продолжаем обработку битов. Если достигли значения 8, то проверяем, все ли элементы массива были обработаны (путем сравнения текущего адреса, в котором лежит элемент массива, с адресом конечного элемента массива). В случае, когда еще остались элементы массива – переходим в подпрограмму *array\_proc* и продолжаем обрабатывать элементы. Если текущий адрес элемента массива совпал с адресом последнего элемента массива, то считаем, что работа алгоритма CRC8 завершена. Следовательно, достаем элементы из стека и переходим в под программу *loop.*

3. Листинг кода

Листинг кода для задачи 1:

**.section** **.text**.Reset\_Handler

.weak Reset\_Handler

**.type** Reset\_Handler, **%function**

**Reset\_Handler:**

ldr r0, =\_estack

mov sp, r0 /\* set stack pointer \*/

mov r0, #0x20000000 @указатель на начало строки

mov r1, #0x3 @количество элементов массива

mov r3, #0x10 @запись элементов массива в r3

strb r3, [r0], #1

mov r3, #0x20

strb r3, [r0], #1

mov r3, #0x30

strb r3, [r0], #1

bl crc8

**loop:**

b loop

**crc8:**

push {r0, r1, r3, r4}

mov r2, #0x0 @в r2 будет результат CRC8, 0х0 - init (нач. значение)

add r1, r0

**array\_proc:**

ldrb r3, [r0], #1 @значение по адресу r0 загрузить в r3 и прибавить 1 к адресу

eor r2, r3

mov r4, #0 @счетчик количества обработанных битов (до 8)

**element\_proc:**

lsl r2, #1

tst r2, #0x100 @логическое "и" - проверка, что число не вылезло за

@байт (т.е. не 1 0000 0000)

beq end\_proc @если не вылезло, переходим в подпрограмму

and r2, #0xFF @старшая 1 сбросится, нули так и останутся

eor r2, #0x07 @инверсия трех младших битов (полином crc8 7=0111)

**end\_proc:**

add r4, #1 @увеличиваем счетчик

cmp r4, #8 @если НЕ равны, то bne и обраб. следующий бит

bne element\_proc

cmp r1, r0 @счетчик элем. массива

bne array\_proc @если не обработали все, продолжаем обрабатывать

pop {r0, r1, r3, r4}

bx lr

.size Reset\_Handler, .-Reset\_Handler

**4. Проверка работоспособности**

4. Проверка работоспособности задачи 1

Начиная с адреса памяти 0x20000000 запишем 3 элемента массива (в шестнадцатеричной системе счисления): 10, 20, 30.

mov r3, #0x10

strb r3, [r0], #1

mov r3, #0x20

strb r3, [r0], #1

mov r3, #0x30

strb r3, [r0], #1

Проверим результат, используя онлайн-калькулятор (https://crccalc.com/) для расчета контрольной суммы по алгоритму CRC.

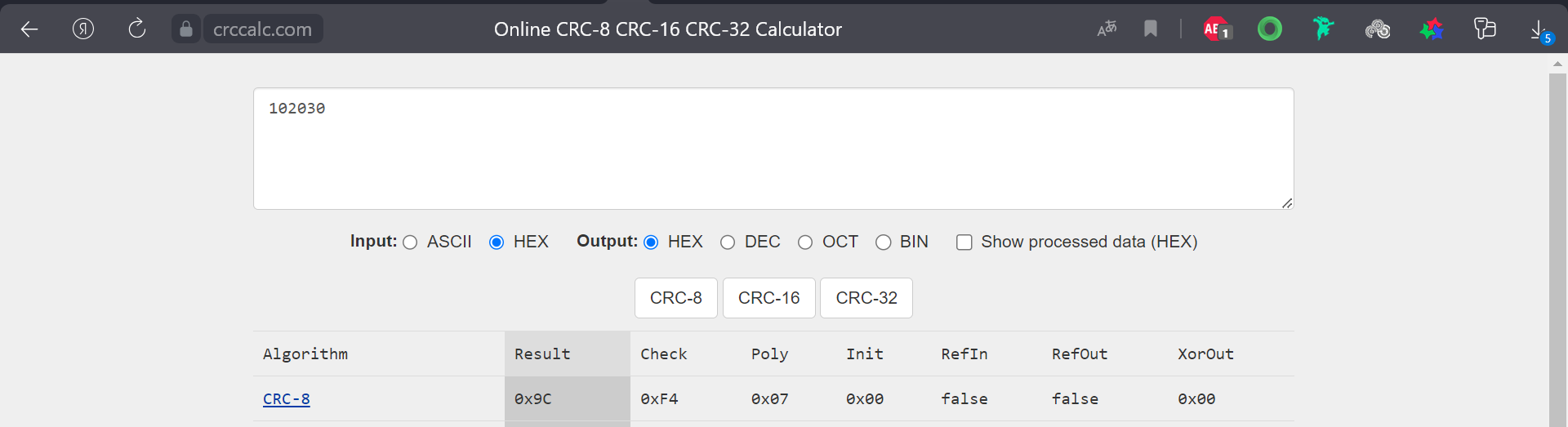


Рисунок 1 – Проверка результата в онлайн-калькуляторе CRC

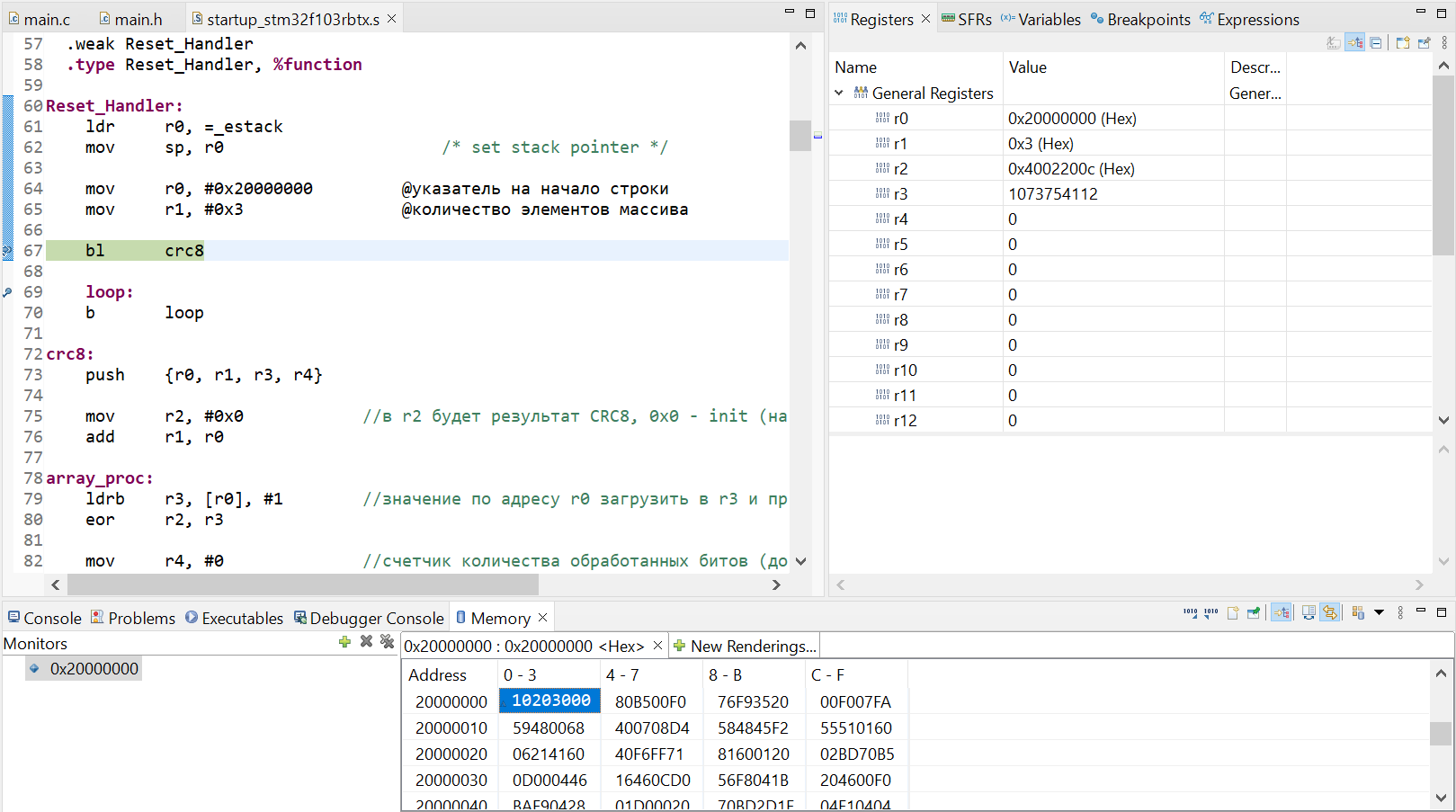
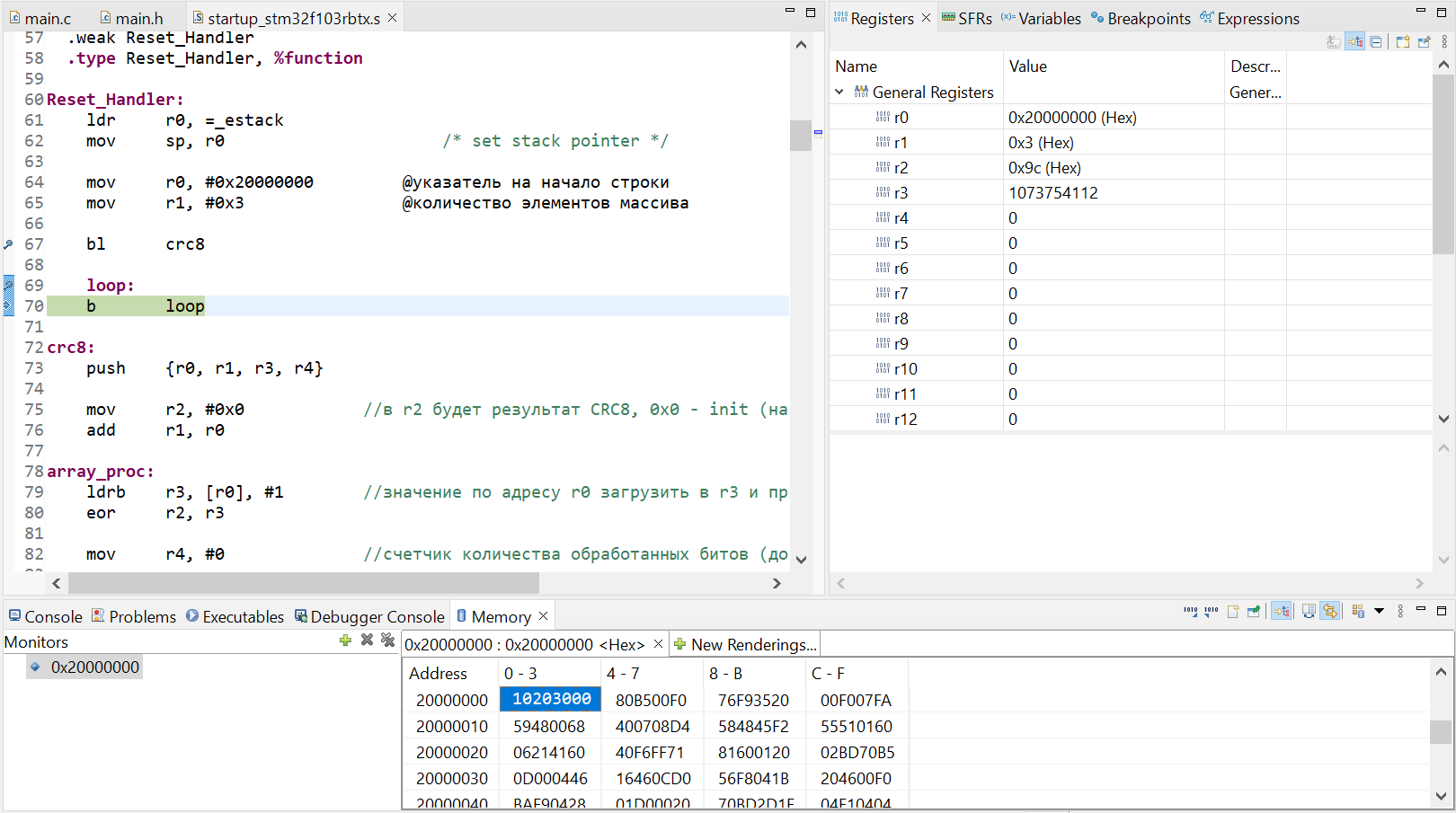


Рисунок 2 – Значение регистра r2 в момент входа в подпрограмму

 Рисунок 3 – Значение регистра r2 после отработки подпрограммы

Заключение

В ходе данной работы был получен опыт написания кода на языке Assembler, а также разработана программа для расчета контрольной суммы по алгоритму CRC8.