ГУАП

КАФЕДРА № 44

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

канд. техн. наук		Т.Н.Соловьёва
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

АРХИТЕКТУРА ЯДРА И СИСТЕМА КОМАНД МИКРОКОНТРОЛЛЕРОВ MCS-51

по курсу: МИКРОКОНТРОЛЛЕРНЫЕ СИСТЕМЫ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4143		Д.В.Пономарев
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

Цель работы

Изучение архитектуры ядра и системы команд микроконтроллеров семейства MCS-51; приобретение навыков программирования микроконтроллеров.

Задание по работе

Необходимо разработать три программы на языке ассемблера MCS-51:

1) программу для вычисления заданного арифметического выражения (для всех операциях полагайте, что операнды и результат – целые однобайтные числа; результат вычислений разместите в ячейке внутренней памяти данных 30h);

2) программу для записи заданного массива чисел во внешнюю память данных;

3) программу на ассемблере битового процессора для вычисления заданного логического выражения (результат выполнения разместите в любой ячейке памяти данных битового процессора).

Работу программ необходимо проверить с помощью симулятора.

Задание индивидуального варианта продемонстрировано на рисунке 1, где в первой ячейке показан номер индивидуального варианта, а в ячейках 2 – 4 показаны соответственно задания для задач 1 – 3.

4	$\left(XZ-\frac{X}{Y}\right)(2Z+Y)$	30h…20h…30h	$(x\vee\bar{y})(r\oplus d)$

Рисунок 1 – Индивидуальное задание

Разработка программы 1

Задумка алгоритма работы программы довольно проста. Для начала необходимо обозначить и идентифицировать некоторыми числовыми значениями переменные X, Y и _Z. Обозначение проводилось с помощью команды ***equ***, где в результате для каждой переменной (не только для перечисленных трёх, но и для промежуточных и результирующих, о которых далее) было объявлено своё место в памяти. Выбор места определялся

«случайным» в некотором смысле образом, не влияющим на суть логики программы.

Так как выражение состоит из нескольких математических операций, следует завести промежуточные переменные, которые будут хранить в себе значение какой-либо операции для дальнейшего использования. Также для хранения результата была добавлена переменная rez.

Затем нужно задать начальные числовые значения для исходных переменных X, Y и _Z. Выбор значения осуществлялся случайным образом.

Промежуточные переменные были даны каждому арифметическому действию из двух переменных.

Математические операции проводились с помощью команд: ***div, add, subb*** и ***mul***. Также следует отметить, что для корректной работы некоторых команд использовались аккумуляторы A и B.

Текст программы

```
;*****************************************************************
; Filename: ex3.asm *
; Date: 2024/02/08 *
; File Version: 0 *
; Author:Ponomarev D. V.  *
; Company: SUAI  *
; Description: lb_1 *
; *
;*****************************************************************
*
X equ 18h
Y equ 19h
_Z equ 1ah
buf1 equ 1ch ; для (X*Z)
buf2 equ 1dh ; для (X/Y)
buf3 equ 1eh ; для ((X*Z - X/Y))
```

```
buf4 equ 1fh ; для (2*Z)
buf5 equ 20h ; для (2*Z + Y)
rez equ 30h ; для результата
;*****************************************************************
*
; Reset Vector
;*****************************************************************
*
RES_VECT CODE 0x0000 ; вектор сброса процессора
SJMP START ; переход к началу программы
;*****************************************************************
*
; MAIN PROGRAM
;*****************************************************************
*
MAIN_PROG CODE 0x0100
START:
mov _Z,#2h
mov X,#3h
mov Y,#1h
mov a, X ; загружаем X в A
mov b, _Z ; загружаем Z в B
mul ab ; умножаем X на Z
mov buf1, a ; сохраняем результат (X*Z) в buf1
mov a, X ; загружаем X в A
mov b, Y ; загружаем Y в B
div ab ; делим X на Y
mov buf2, a ; сохраняем результат (X/Y) в buf2
mov a, buf1 ; загружаем (X*Z) из buf1 в A
subb a, buf2 ; вычитаем (X/Y) из (X*Z)
```

```
mov buf3, a ; сохраняем результат ((X*Z - X/Y)) в buf3
mov a, _Z ; загружаем Z в A
add a, a ; умножаем Z на 2
mov buf4, a ; сохраняем результат (2*Z) в buf4
mov a, buf4 ; загружаем (2*Z) из buf4 в A
mov b, Y ; загружаем Y в B
add a, b ; добавляем Y к (2*Z)
mov buf5, a ; сохраняем результат (2*Z + Y) в buf5
mov a, buf3 ; загружаем ((X*Z - X/Y)) из buf3 в A
mov b, buf5 ; загружаем (2*Z + Y) из buf5 в B
mul ab ; умножаем ((X*Z - X/Y)) на (2*Z + Y)
mov rez, a ; сохраняем результат в rez
SJMP $ ; бесконечный цикл
END
```

Результат работы программы

Результаты работы программы продемонстрированы на рисунках 2 - 5.

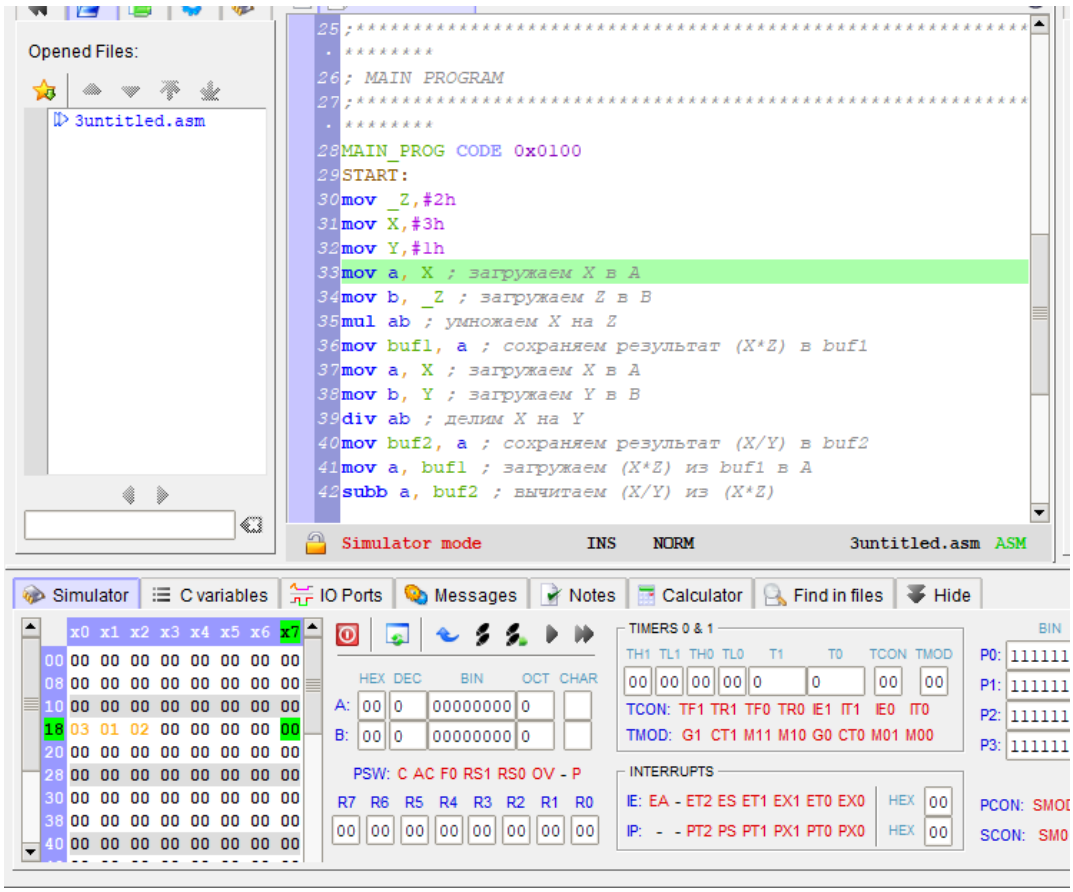

Рисунок 2 – Вывод числовых значений X,Y,_Z

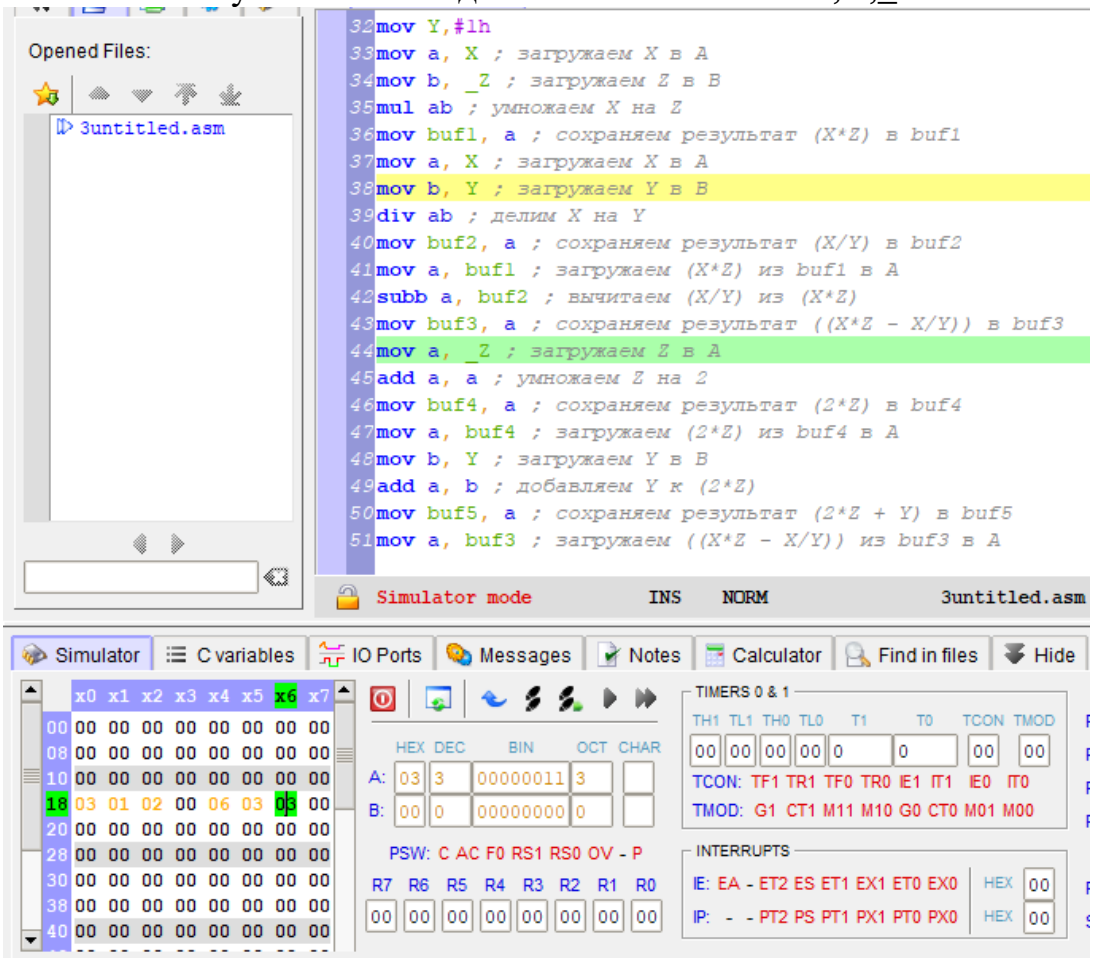

Рисунок 3 – Результат первой скобки

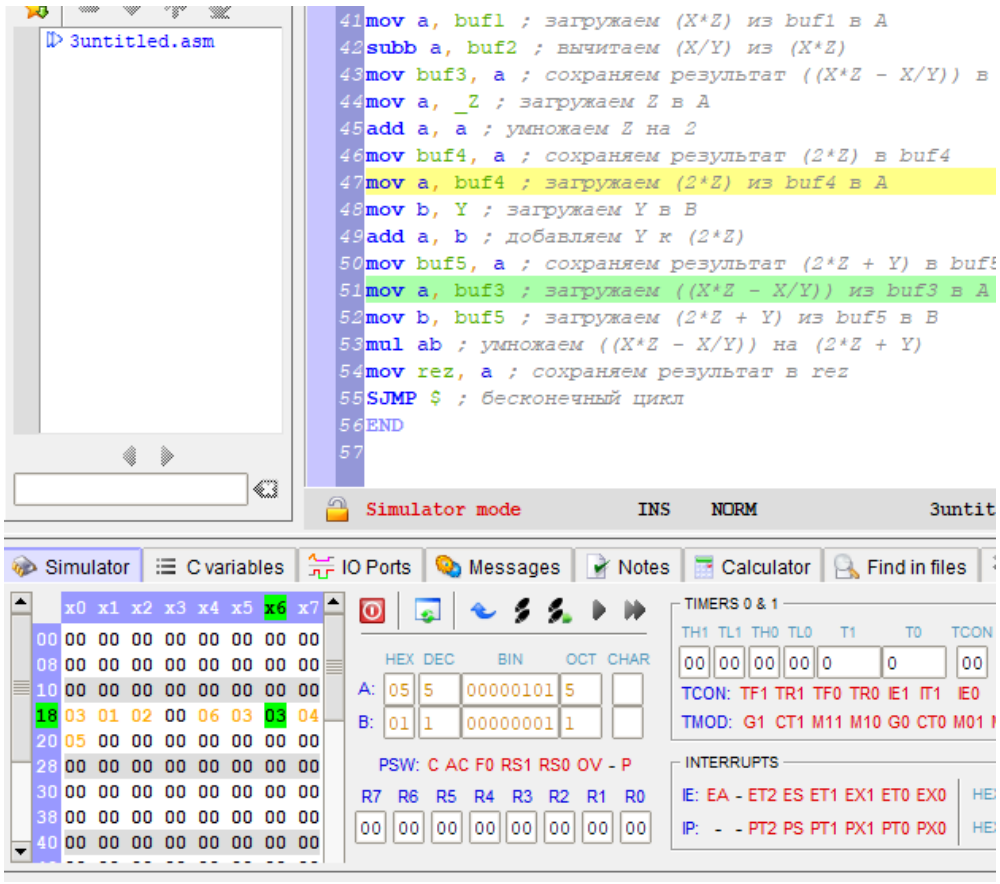

Рисунок 4 – Вывод результата 2 скобки

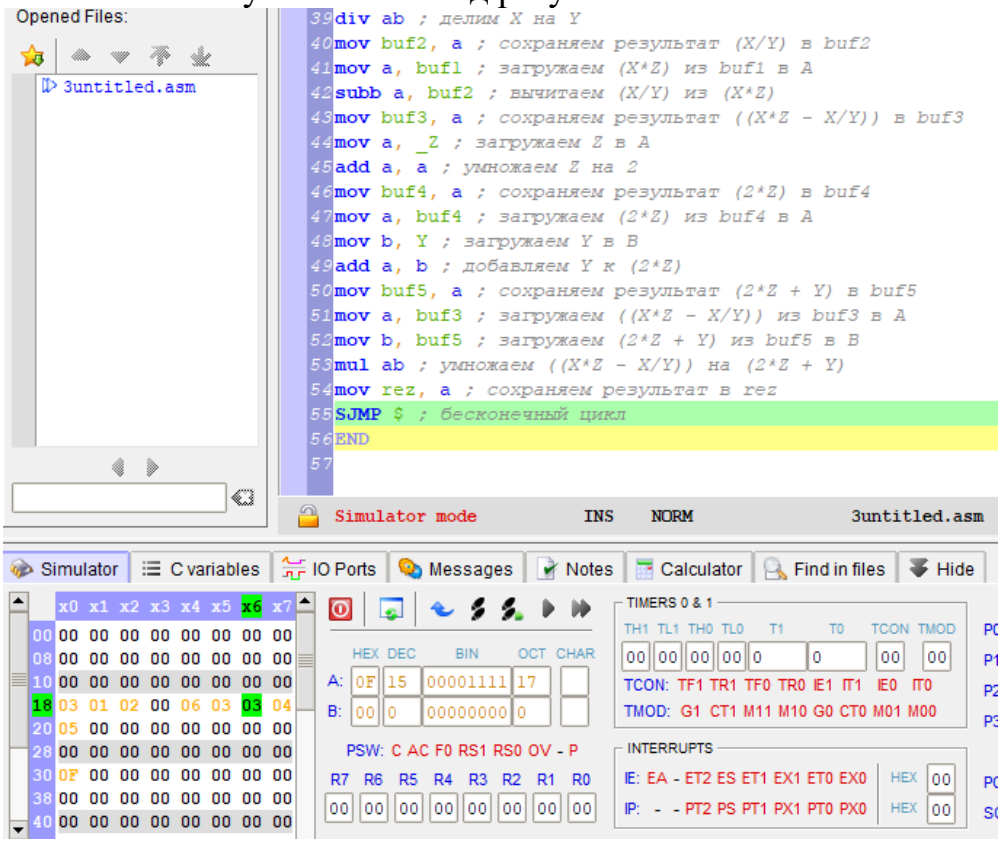

Рисунок 5 – Вывод итогового результата

Разработка программы 2

В коде на языке ассемблера для микроконтроллера сначала устанавливаются начальные значения (R0) и значения (A) равного 0x30 (48 в десятичной системе).

Затем программа поочередно уменьшает значение переменной и записывает его в память, уменьшая адрес, пока значение A не достигнет 0x20 (32 в десятичной системе).

Затем программа увеличивает значение переменной и записывает его в память, увеличивая адрес, пока значение A не достигнет 0x30 (48 в десятичной системе). После этого программа зацикливается в бесконечном цикле.

Текст программы

```
;*******************************************************************
; Filename: ex1.asm *
; Date: 2024/02/08 *
; File Version: 1 *
; Author: Ponomarev D. V. *
; Company: SUAI *
; Description:lb_1 *
;*******************************************************************
; Reset Vector
;*******************************************************************
*
org 0h ; процедура сброса процессора
ajmp start ; переход к началу программы
;*******************************************************************
*
; MAIN PROGRAM
;*******************************************************************
*
```

```
org 100h
start:
mov R0, #030h ; начальный адрес -> R0
mov A, #030h ; начальное значение -> A
decrement:
movx @R0, A
dec R0
dec A
cjne A, #020h, decrement ; если A равно 20h, пропускаем уменьшение

increment:
movx @R0, A
inc R0
inc A
cjne A, #030h, increment ; если A равно 20h, пропускаем уменьшение

sjmp $ ; бесконечный цикл
END
```

Результат работы программы

Результаты работы программы продемонстрированы на рисунках 6 - 8

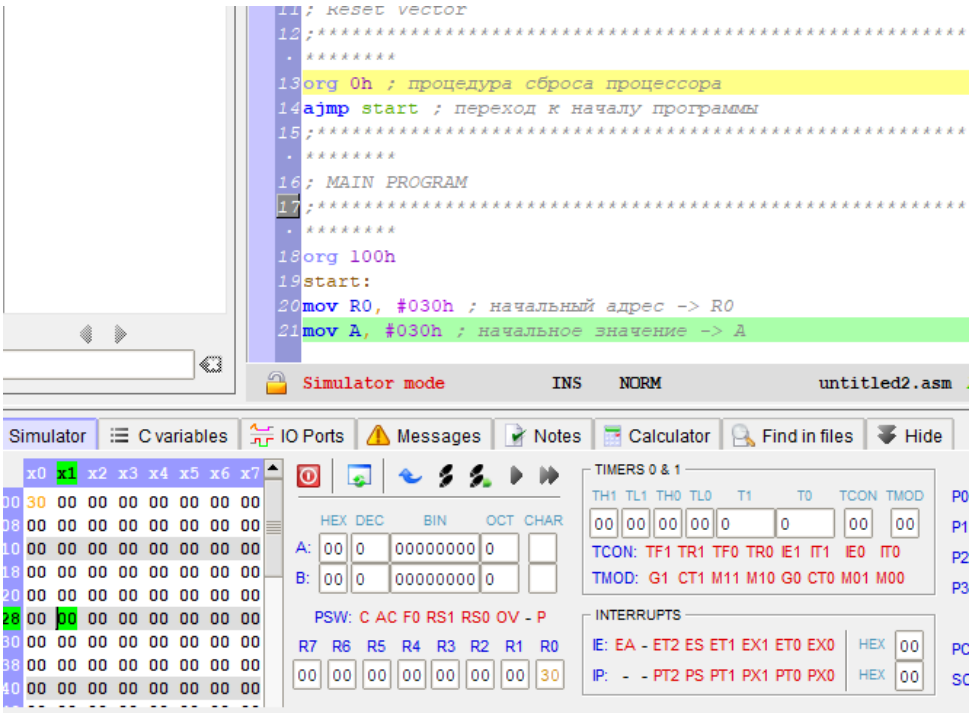

Рисунок 6 – Начало

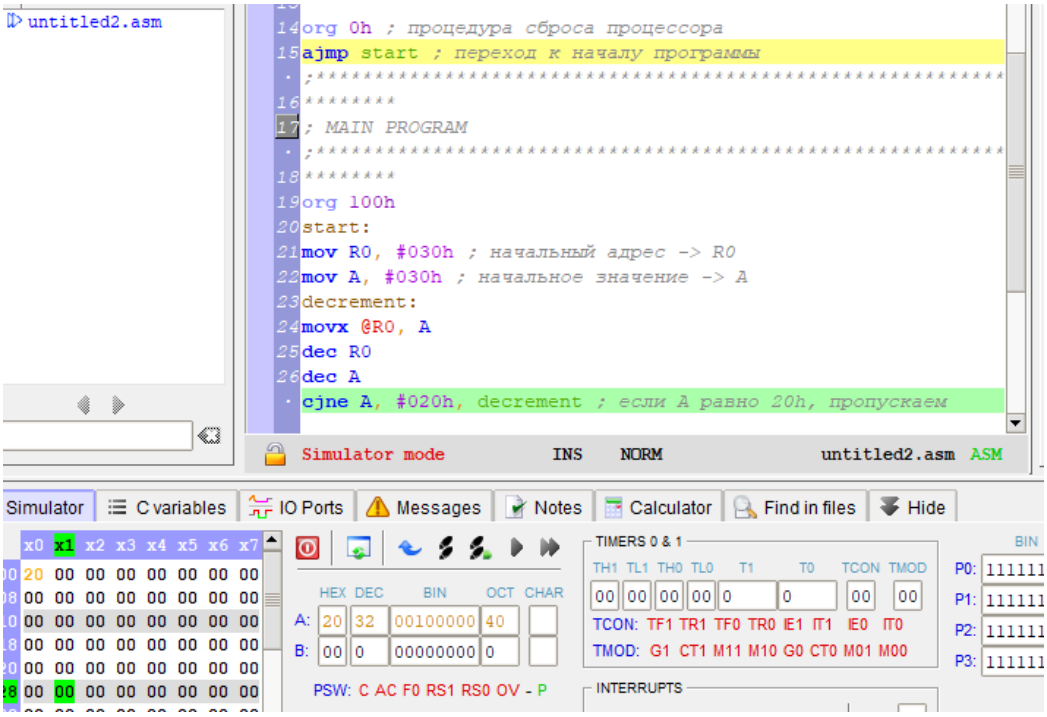

Рисунок 7 – Достижение 20h

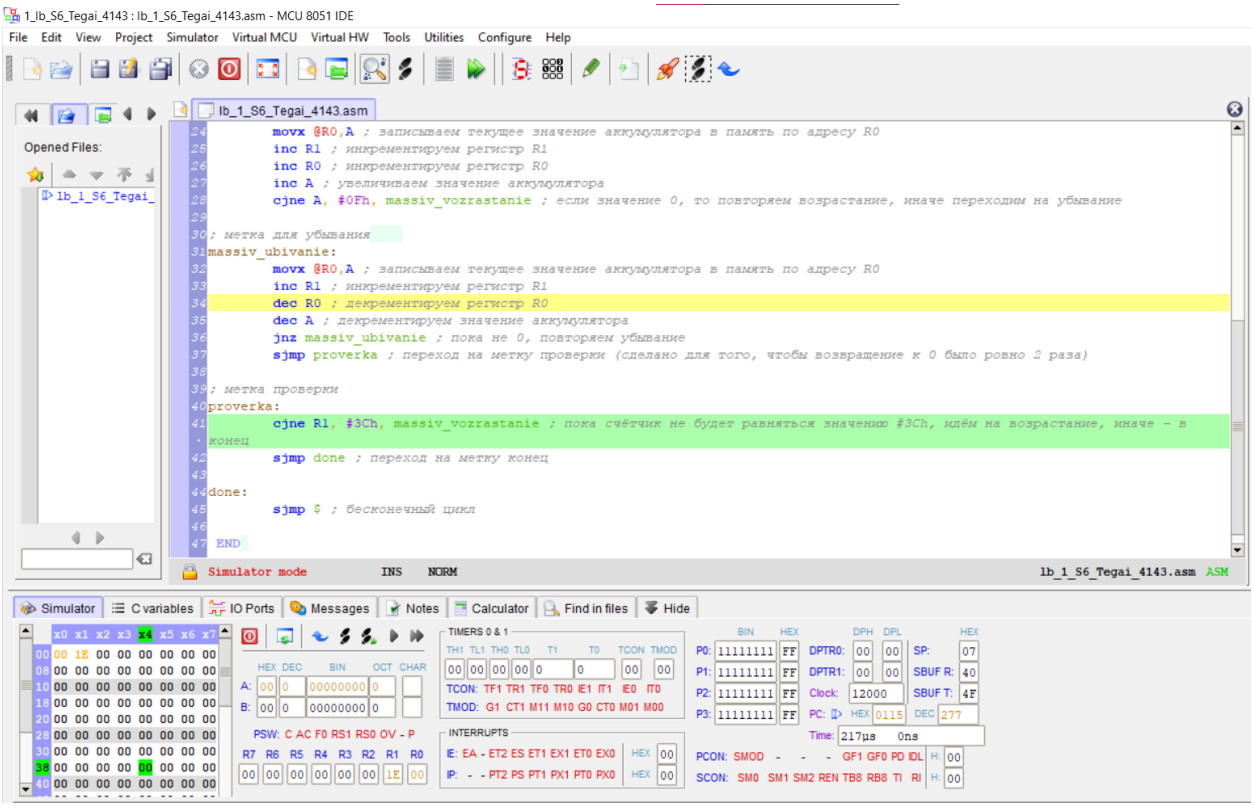

Рисунок 8 – Окончание работы

Разработка программы 3

Разработка алгоритма программы для последнего задания по сложности ничем не отличается от разработки первого. Так что для начала нужно раскрыть сложение по модулю 2. Это показано на рисунке 9.

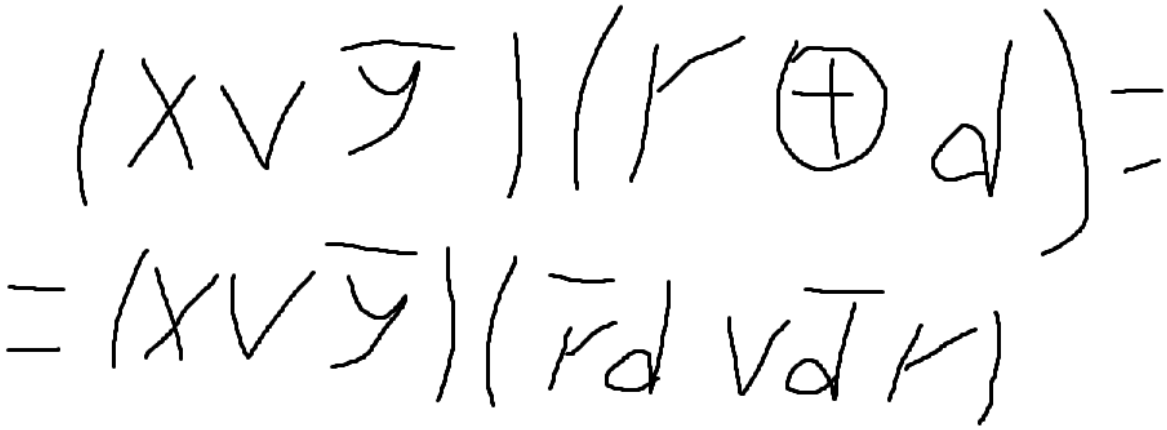

Рисунок 9 – Раскрытие сложения по модулю 2

Именно с этим выражением и будет связана логика работы кода.

Для начала определяется место для хранения переменных x,y,r,d. Также определяются места хранения для промежуточных переменных buf,buf1 и результирующей переменной rez.

Результатом работы является полученное значение на флаге C. Если он горит зелёным – результатом выражения является 1, иначе – 0.

Текст программы

```
;*******************************************************************
; Filename: ex2.asm *
; Date: 2023/02/09 *
; File Version: 1 *
; Author: Ponomarev D. V. *
; Company: SUAI *
; Description: lb_1 *
;*******************************************************************
; Variables *
;*******************************************************************
x equ 18h
y equ 19h
r equ 1ah
d equ 1bh
buf equ 1ch
buf1 equ 1dh
rez equ 20h;
;*******************************************************************
; Reset Vector
;*******************************************************************
org 0h ; processor reset vector
ajmp start ; go to beginning of program
;*******************************************************************
; MAIN PROGRAM
```

```
;*****************************************************************
;(x v !y)((!rd)+(r!d))
start:
clr c
orl c, /y ;добовляем не y
orl c, x ; добовляем x
mov buf, c ;записываем результат

mov c, d ;присваиваем с значение д
anl c, /r ;умножаем на не р
mov buf1, c ;записываем результат
mov c, r ; присваиваем с значение р
anl c, /d ;умножаем на не д
orl c, buf1 ;прибавляем полученный результат bud1
anl c,buf ; перемножаем buf и buf1
mov rez, c ;записываем результат в рез
sjmp $ ; loop forever
END
```

Результат работы программы

Результат работы программы продемонстрирован на рисунке 18.

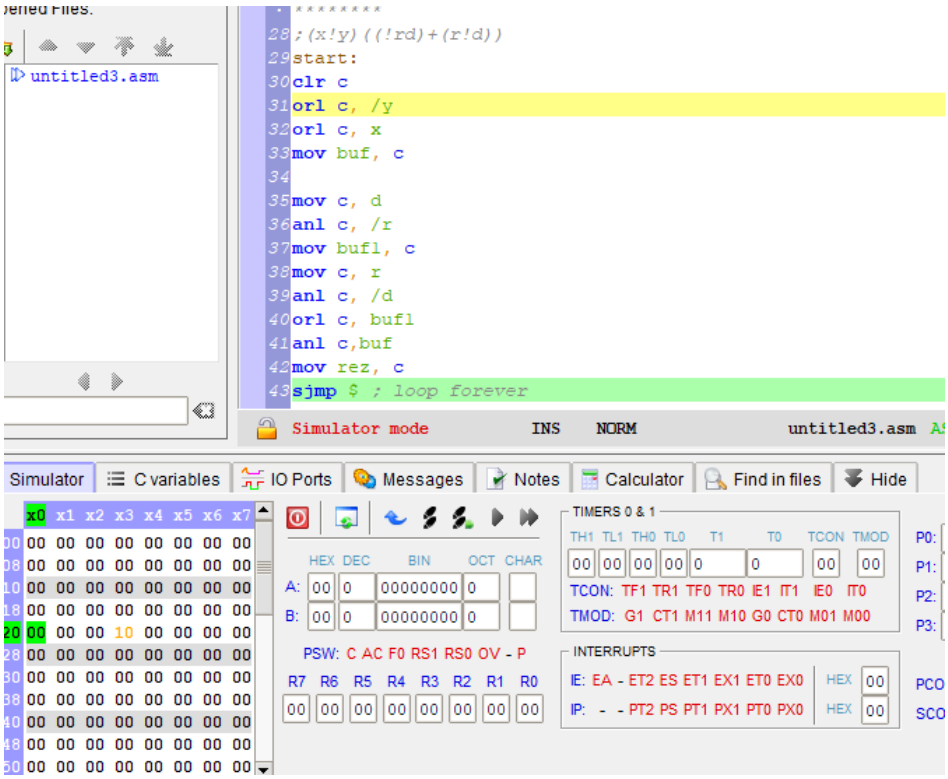

Рисунок 10 - Результат работы программы

Вывод

В результате выполнения работы созданы три программы на языке ассемблера MCS-51: программа для вычисления заданного арифметического выражения, программа для записи заданного массива чисел во внешнюю память данных, а также программа на ассемблере битового процессора для вычисления заданного логического выражения. Проверка работоспособности программ произведена в среде MCU 8051 IDE. Изучена архитектура и система команд микроконтроллера семейства MCS-51; приобретены навыки программирования микроконтроллера