|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |

Институт искусственного интеллекта

Кафедра проблем управления

**ЛАБОРАТОРНАЯ РАБОТА №1**

по дисциплине **Микропроцессорная техника в мехатронике и**

**робототехнике**

**Тема лабораторной работы: «**Реализация многоразрядного умножения на 8-миразрядном микроконтроллере семейства AVR.»

|  |  |
| --- | --- |
| **Студенты группы:** КРБО-03-23 | Зенина А. А. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
|  | Грачев А. А. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
|  | Гришаев А. К. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| **Преподаватель:** | ассистент Бессонова А. В. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

|  |  |
| --- | --- |
| Работа представлена к защите: | «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2025 г. |

Москва 2025

# **1. Цель работы**

Реализация многоразрядного умножения, сложения и вычитания на 8-ми разрядном микроконтроллере семейства AVR.

# **2. Задача работы**

1. Выбрать два 16-битных числа (>256), записать представление в 10-ричной, двоичной и 16-ричной системах счисления.

2. Реализовать на языке ассемблер на МК AVR алгоритмы умножения, сложения и вычитания двух 16-битных чисел.

3. Сравнить результаты, полученные с помощью калькулятора и с помощью алгоритмов.

# **3. Теоретические сведения**

Для перемножения двух 8-ми разрядных чисел на 8-ми разрядном микроконтроллере AVR используется ассемблерная инструкция MUL. Данная инструкция принимает два аргумента (два регистра общего назначения (РОН)), содержащих значения перемножаемых величин. Результат умножения помещается в РОН R0 и R1, причём в регистр R0 помещаются младшие 8 бит результата, в R1 – старшие 8 бит результата.

Для реализации многоразрядного умножения применяется специальный алгоритм, оперирующий с частями исходных множителей. Пусть имеются два 16-битных числа, которые необходимо перемножить X и Y. Представим данные числа в виде суммы младшего и старшего байт исходного числа:

X = 28 XH + XL

Y = 28 YH + YL

где XH и XL – старший и младший байты исходного числа X, YH и YL – старший и младший байты исходного числа Y.

Тогда операцию умножения X и Y можно представить в следующем виде:

X \* Y = 216 XH YH + 28(XH YL + YH XL) + XLYL

Таким образом, операция умножения двух 16-битных чисел свелась к 4-м умножениям и трём сложениям между 8-ми битными числами. Для хранения результата такого умножения необходимо выделить 4 байта.

Для операций сложения и вычитания 16-битных чисел используются инструкции ADD (сложение) и SUB (вычитание) с учетом переноса (ADC и SBC для старших байтов).

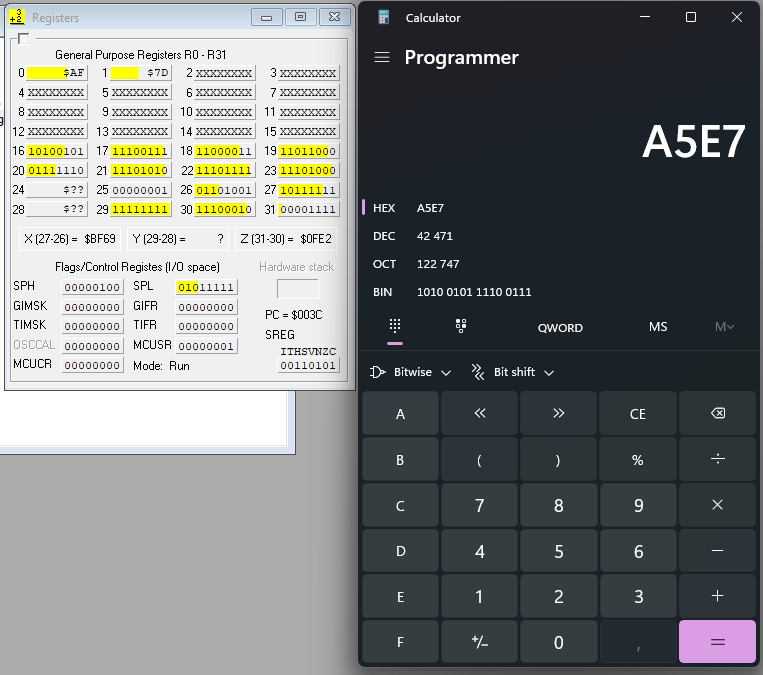
# **4. Расчетная часть**

4.1. Исходные данные

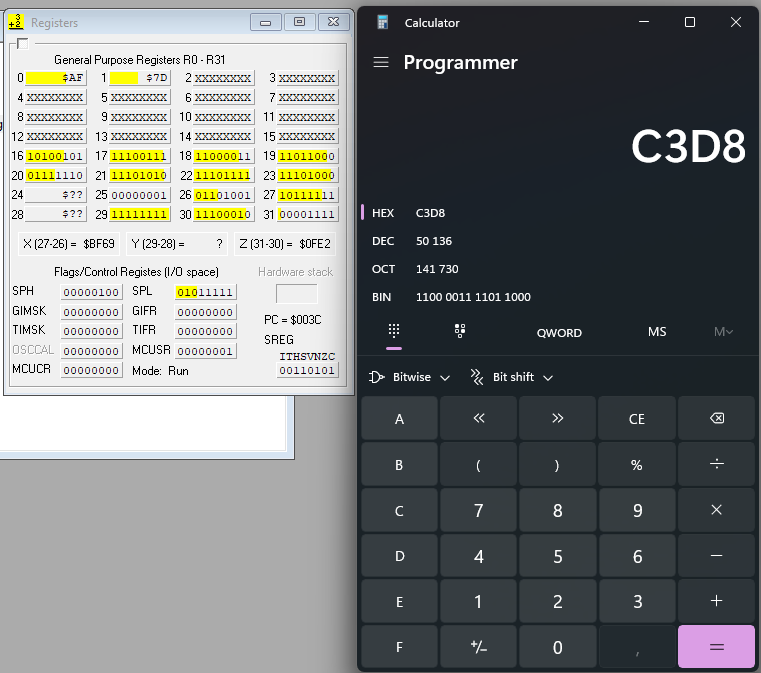
Были выбраны два 16-битных числа:

- Первое число: 0xA5E7 (42471 в десятичной системе)

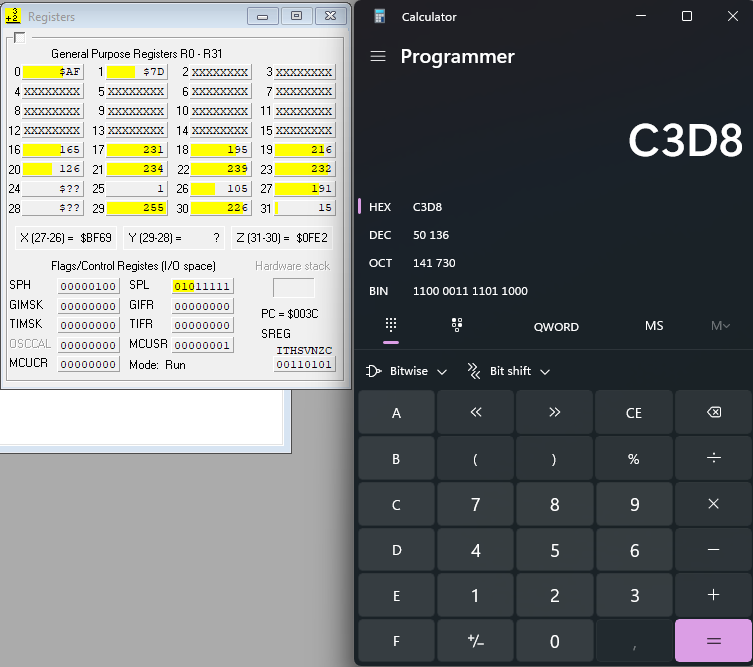
- Второе число: 0xC3D8 (50136 в десятичной системе)



*Рисунок 1 – Первое число в двоичной системе*



*Рисунок 2 – Второе число в двоичной системе*



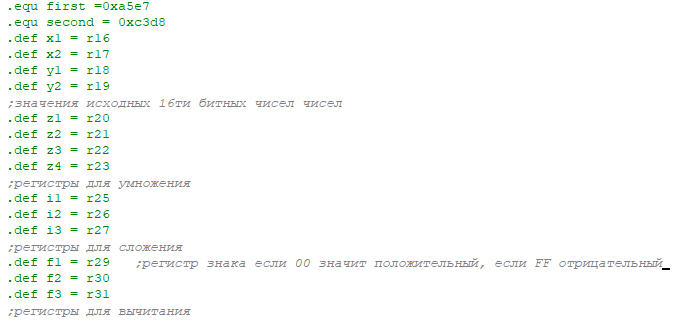
*Рисунок 3 – Числа в десятичной системе*

4.2. Реализация алгоритмов на ассемблере

Код программы на ассемблере для микроконтроллера AVR представлен в приложении А.

4.3. Назначение регистров

Назначение регистров в программе указано на рисунке 1.



*Рисунок 4 – Назначение регистров в VMLAB*

Регистры, содержащие исходные числа:

- R16 (x1): Старший байт первого числа (0xA5)

- R17 (x2): Младший байт первого числа (0xE7)

- R18 (y1): Старший байт второго числа (0xC3)

- R19 (y2): Младший байт второго числа (0xD8)

Регистры, содержащие результат:

- R20 (z1): Старший байт результата (для 32-битного результата)

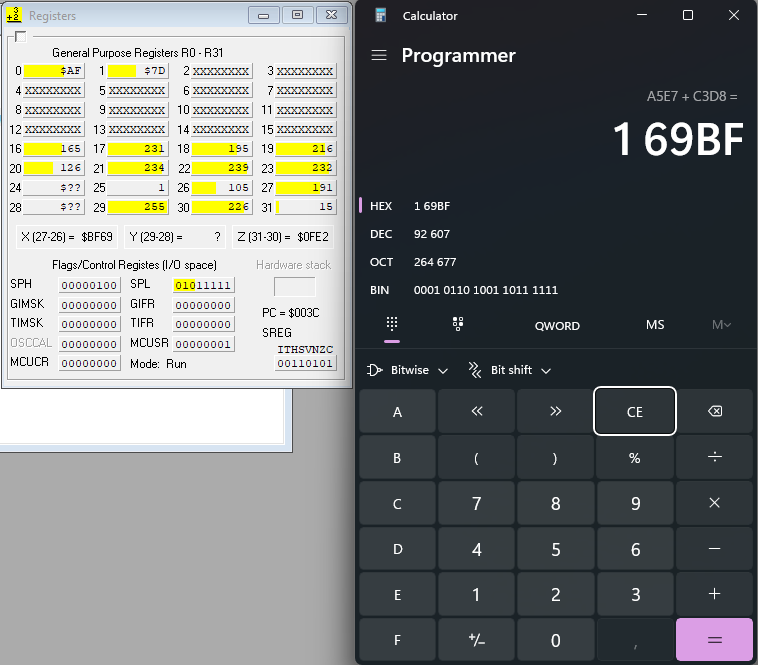
- R21 (z2): Байт 3 результата

- R22 (z3): Байт 2 результата

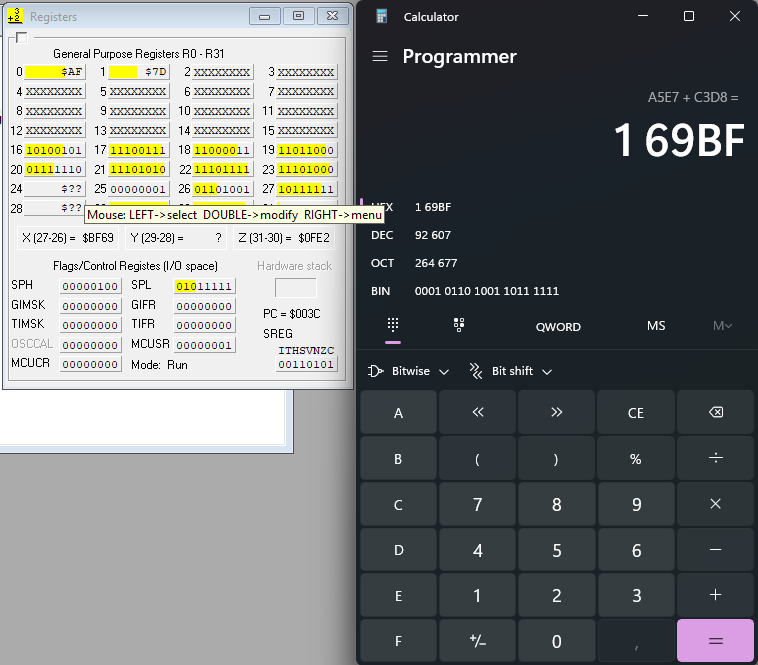
- R23 (z4): Младший байт результата

4.4. Результаты вычислений

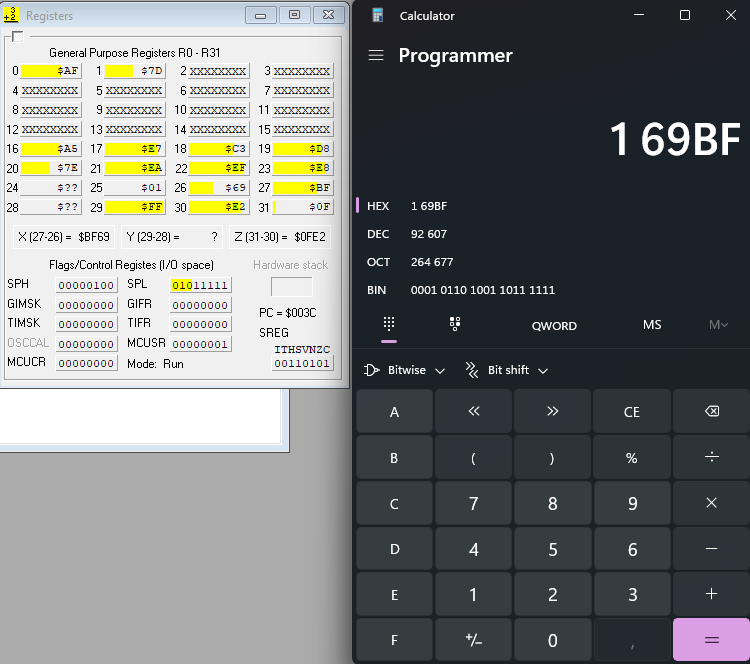
Результат сложения:



*Рисунок 5 – Сложение в десятичной системе*

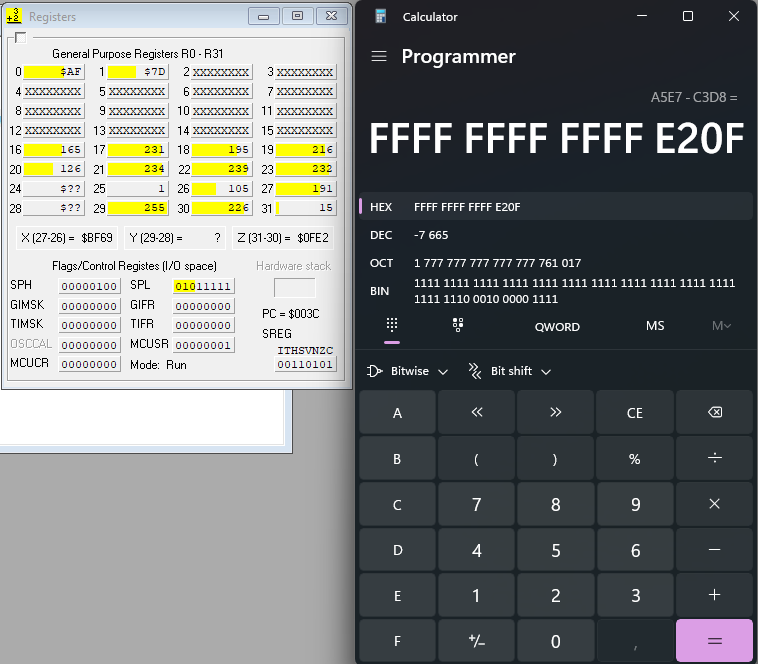


*Рисунок 6 – Сложение в двоичной системе*

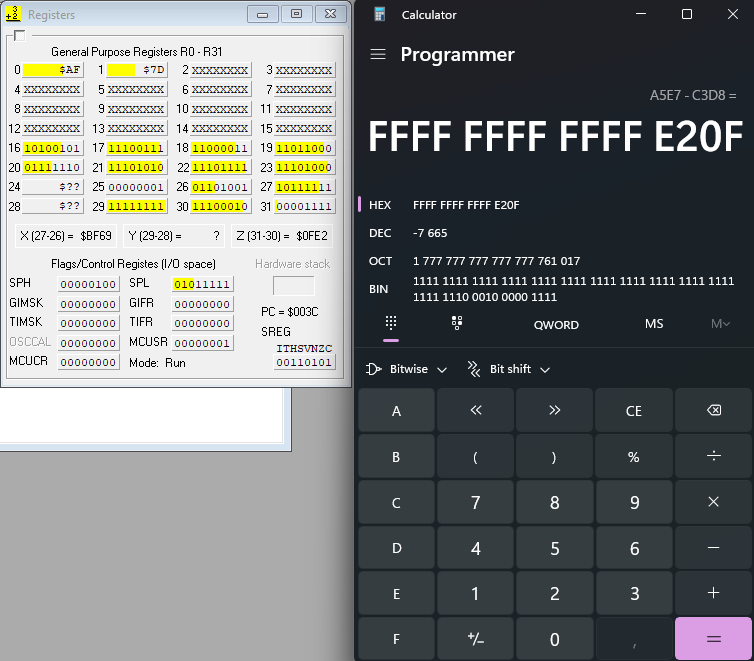


*Рисунок 7 – Сложение в шестнадцатеричной системе*

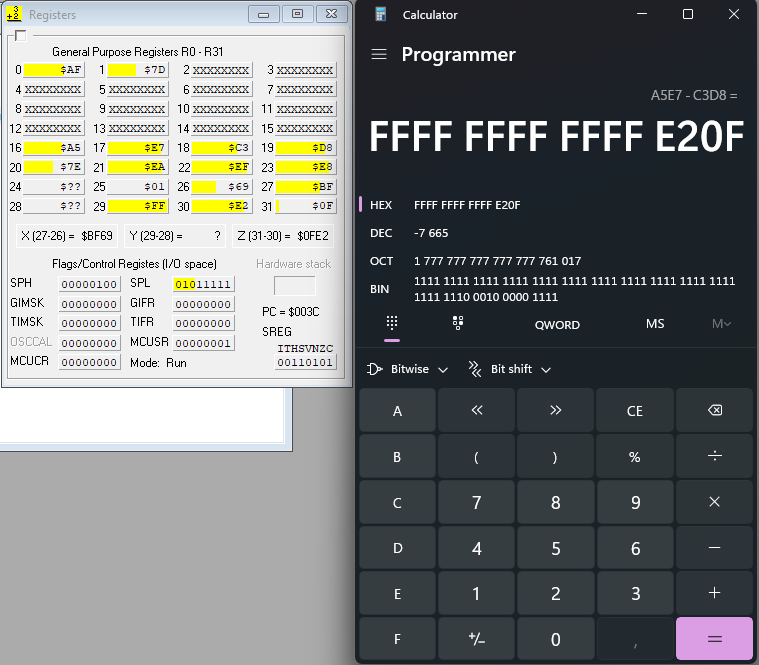
Результат вычитания:



*Рисунок 8 – Вычитание в десятичной системе*

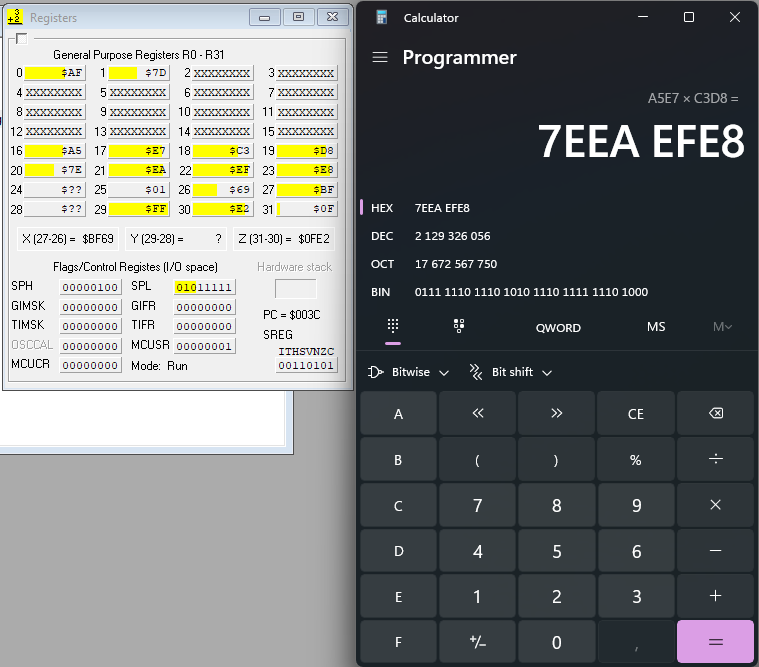


*Рисунок 9 – Вычитание в двоичной системе*

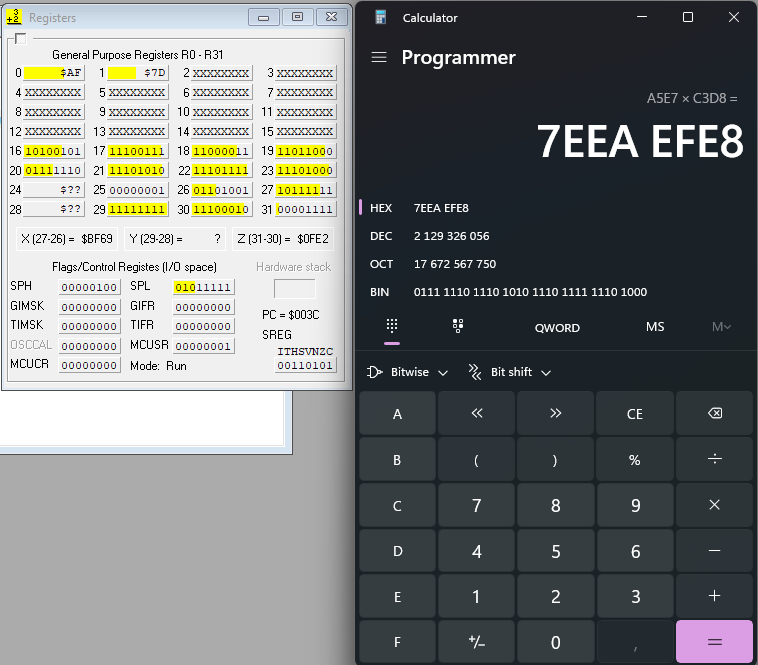


*Рисунок 10 – Вычитание в шестнадцатеричной системе*

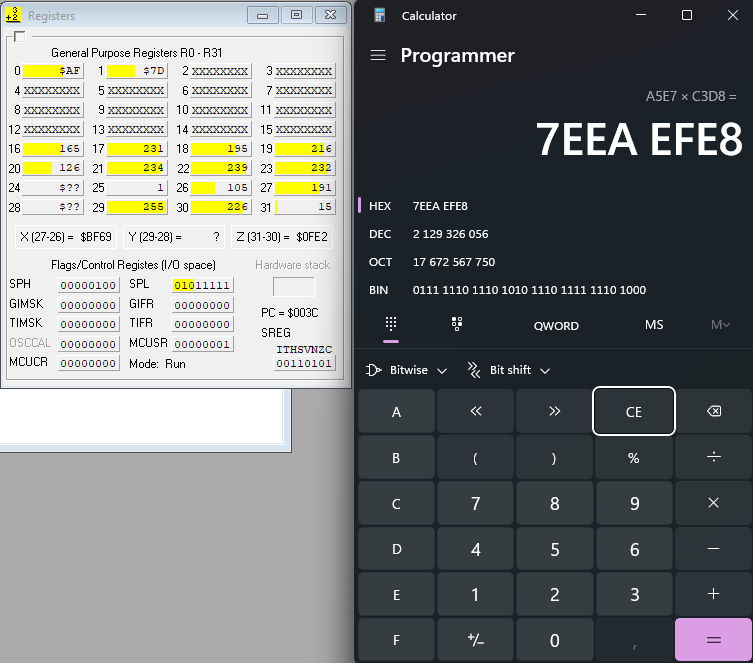
Результат умножения:



*Рисунок 11 – Умножение в шестнадцатеричной системе*



*Рисунок 12 – Умножение в двоичной системе*



*Рисунок 13 – Умножение в десятичной системе*

4.5. Сравнение с калькулятором

Результаты, полученные с помощью реализованных алгоритмов, полностью совпадают с результатами, полученными с помощью калькулятора, что подтверждает корректность работы программы.

# **5. Выводы по работе**

В ходе лабораторной работы были успешно реализованы алгоритмы умножения, сложения и вычитания 16-битных чисел на 8-ми разрядном микроконтроллере AVR.

# **6. Список используемой литературы**

1. Документация по микроконтроллерам AVR - Atmel Corporation

**ПРИЛОЖЕНИЕ А**

.include "C:\VMLAB\include\m8def.inc"  
; Определение чисел  
.equ first = 0xA5E7  
.equ second = 0xC3D8  
; Определение регистров  
.def x1 = r16 ; Старший байт первого числа  
.def x2 = r17 ; Младший байт первого числа  
.def y1 = r18 ; Старший байт второго числа   
.def y2 = r19 ; Младший байт второго числа  
.def z1 = r20 ; Байт 1 результата (младший)  
.def z2 = r21 ; Байт 2 результата  
.def z3 = r22 ; Байт 3 результата  
.def z4 = r23 ; Байт 4 результата (старший)  
; Программа начинается здесь  
start:  
 ; Инициализация чисел  
 ldi x1, high(first)  
 ldi x2, low(first)  
 ldi y1, high(second)   
 ldi y2, low(second)  
 ; Обнуление регистров результата  
 ldi z1, 0  
 ldi z2, 0  
 ldi z3, 0  
 ldi z4, 0

; Блок умножения  
mul\_for\_numbers:  
 mul x2, y2 ; X\_L \* Y\_L  
 add z4, r0 ; Младший байт результата  
 add z3, r1 ; Старший байт с переносом  
 mul x1, y2 ; X\_H \* Y\_L  
 add z3, r0 ; Добавляем младший байт  
 adc z2, r1 ; Добавляем старший байт с переносом  
 mul x2, y1 ; X\_L \* Y\_H   
 add z3, r0 ; Добавляем младший байт  
 adc z2, r1 ; Добавляем старший байт с переносом  
 adc z1, z1 ; Учитываем перенос в старший байт  
 mul x1, y1 ; X\_H \* Y\_H  
 add z2, r0 ; Добавляем младший байт  
 adc z1, r1 ; Добавляем старший байт с переносом  
; Блок сложения  
summ:  
 add z4, x2 ; Складываем младшие байты  
 add z4, y2  
 adc z3, x1 ; Складываем старшие байты с переносом  
 add z3, y1  
 adc z2, z2 ; Учитываем перенос  
; Блок вычитания   
sub\_numbers:  
 mov z3, x2 ; Копируем младший байт первого числа  
 mov z2, x1 ; Копируем старший байт первого числа  
 sub z3, y2 ; Вычитаем младший байт второго числа  
 sbc z2, y1 ; Вычитаем старший байт с учетом заема  
 ldi z1, 0 ; Обнуляем для расширения знака  
 sbc z1, z1 ; Распространяем заем