ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
Государственное образовательное учреждение высшего профессионального образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Ассистент		А.Н.Долидзе
должность, уч.степень,звание	подпись, дата	инициалы,фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2

«Программная реализация алгоритма выполнения целочисленной операции для архитектуры набора команд ARM»

по курсу: Организация ЭВМ и систем.

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР.	4143		Е.Д.Тегай
		подпись, дата	инициалы,фамилия

Санкт-Петербург
2024

1. Задание

Вариант №4

Промежуточные расчёты для выявления номера индивидуального варианта продемонстрированы на рисунке 1. На рисунке 2 изображена формулировка задания.

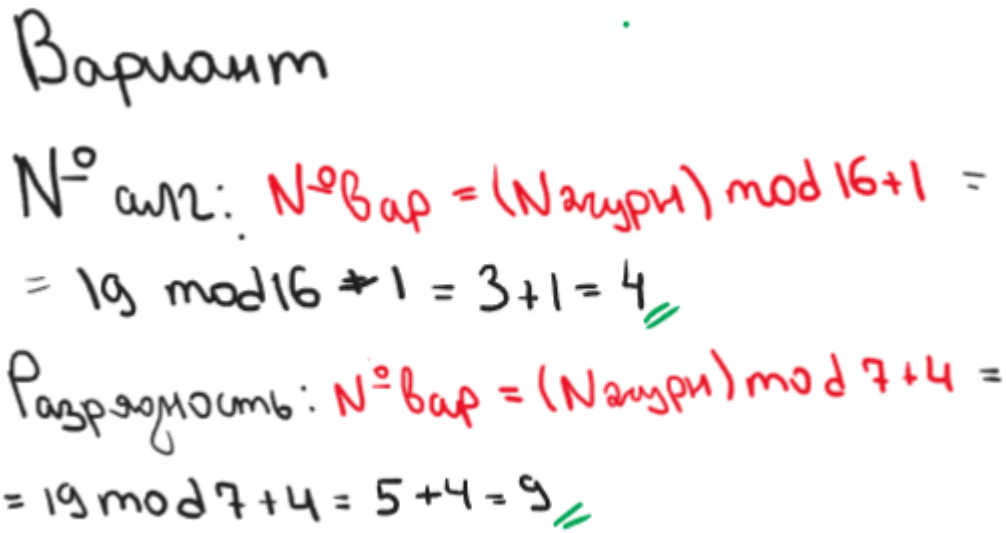

Рисунок 1 – Промежуточные расчёты

4	Умножение целых чисел со знаком в	С коррекцией результата

5	дополнительном коде со сдвигом суммы частичных произведений вправо, неподвижным множимым и анализом множителя, начиная с младших разрядов.	С предварительным изменением знака
6		С преобразованием множителя

Рисунок 2 – Формулировка задания

Текстовое описание алгоритма

- A – множимое,
- B – множитель,
- Rez –частичные произведения
- Razryad – разряд.
- Otr_A – число, обратное числу A.
- >> – сдвиг вправо на один разряд.

2. Текстовое описание алгоритма

Данный алгоритм работает с целыми числами. Пусть A – множимое, B –множитель. Для начала обнуляем СЧП(суммы частичных произведений). Затем анализируем знаки A и B. Это определяется так: если самый старший разряд числа – 1, то это отрицательное число, иначе – положительное. Это понадобится для дальнейших действий согласно алгоритму.

Рассмотрим самый младший разряд множителя. Если он равен 0, то в результат ничего не записывается, а затем происходит сдвиг вправо с сохранением знака числа на 1 разряд. Если же он равен 1, то к результату прибавляется содержимое множимого, а затем так же происходит сдвиг вправо на разряд. После этого происходит цикличное прохождение по тому же принципу по всем разрядам множителя, кроме знакового (самого старшего).

Окончанием работы алгоритма является вывод результата.

3. Блок-схема:

Искомая блок-схема изображена на рисунке 3.

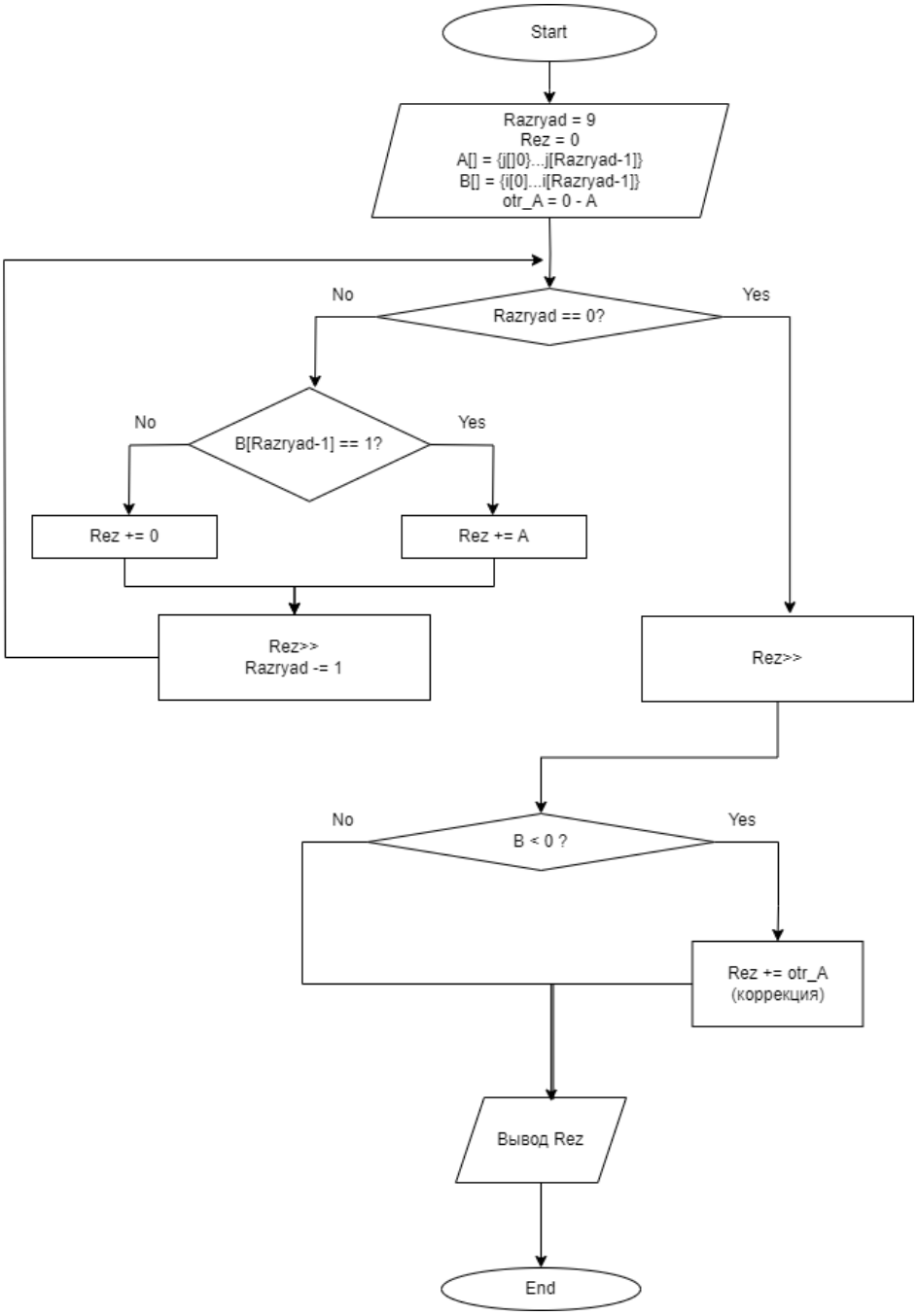

Рисунок 3 – Блок-схема

4. Код программы на языке ассемблера ARM

```
mov r0, #0xFFFFFFEE @Множимое A
mov r1, #0x00000010@Множитель B
mov r0,r0,lsl #8 @ Сдвиг влево, чтобы потом было куда сдвигать вправо
```

```
    без потери битов
mov r2, #0 @Сумма частичных произведений
mov r3, #0 @Разряд числа B
mov r4, #0 @Результат
mov r6, #0 @Значение для последующего вычитания для отр. A
sub r5, r6,r0 @ Отр. A

and r9, r1, #0x80000000 @ Берём самый старший бит
mov r9, r9, lsr #31 @ Сдвигаем его в начало (для определения
    отрицательности B)

@Пока разряд не станет равным 9
if_razryad_eq_9:
cmp r3, #8 @ Проверка регистра на равность значению 9
bne razryad_ne_9 @ Если разряд != 9,  то переход к метке  razryad_ne_9
b razryad_0 @ Иначе переход к метке razryad_0

@ Если разряд != 9
razryad_ne_9:
ldr r6, =1 @ Загрузка в регистр 1
mov r6, r6, LSL r3 @ Сдвигаем единицу на i-ый бит (номер разряда)
and r6,r1,r6 @ Берём у числа B i-ый бит
cmp r6, #0 @ Если значение в регистре не поменялось, то бит = 1, иначе -
    0
bne zifra_1 @ Если 1 - переход к метке zifra_1
b zifra_0 @ Иначе - переход к метке zifra_0

@ Если значение - 0
```

```
zifra_0:
mov r4,r2 @ Копирование значения в другой регистр с целью сохранить
    промежуточный результат
mov r2, r2, ASR #1 @ Сдвиг вправо на 1 разряд
add r3, r3, #1 @ Увеличиваем разряд
b if_razryad_eq_9 @ Переход к метке if_razryad_eq_9

@ Если значение - 1
zifra_1:
add r2, r2, r0 @ К сумме частичных произведений прибавляем A
mov r4, r2 @ Копирование значения в другой регистр с целью сохранить
    промежуточный результат
mov r2, r2, ASR #1 @ Сдвиг вправо на 1 разряд
add r3, r3, #1 @ Увеличиваем разряд
b if_razryad_eq_9 @ Переход к метке if_razryad_eq_9

@ Если разряд == 0
razryad_0:
mov r4, r2 @ Копирование значения в другой регистр с целью сохранить
    промежуточный результат
cmp r9, #1 @ Сравниваем с 1
beq less_then_0 @ Если оно == 1, значит число - отрицательное, иначе -
    положительное
b more_then_0 @ Переход к метке  more_then_0 (полож.число)

@ Когда число отрицательное
less_then_0:
add r4, r4, r5 @ Прибавляем отр. A к результату
```

@Конец программы

more_then_0:

5. Карта распределения памяти под программу и данные

Искомая карта распределения памяти продемонстрирована в таблице 1.

Следует отметить, что в таблице 1 отражена работа кода при входных данных как в примере 4, показанном на рисунках 4 – 5.

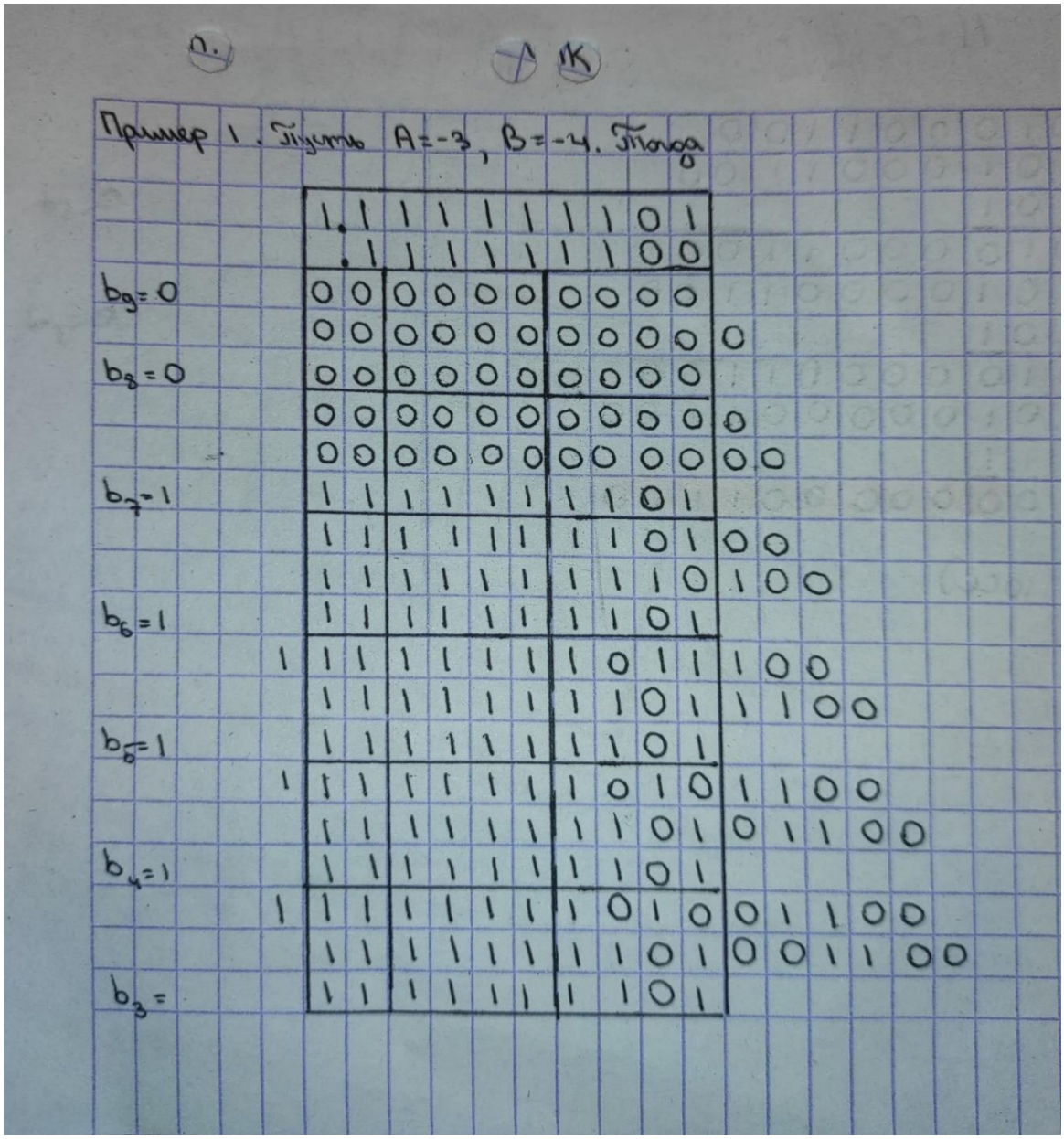

Рисунок 4 – Пример 1

Рисунок 5 – Пример 1

Таблица 1

Команда	Состояние памяти и регистров	
	Было	Стало
mov r0, #0xFFFFFFFD	R0 = 00000000 R15(pc) = 00000000	R0 = fffffffd R15(pc) = 00001004
mov r1, #0xFFFFFFFC	R1 = 00000000 R15(pc) = 00001004	R1 = fffffffc R15(pc) = 00001008
mov r0,r0,lsl #8	R0 = fffffffd R15(pc) = 00001008	R0 = fffffd00 R15(pc) = 0000100c
mov r2, #0	R15(pc) = 0000100c	R15(pc) = 00001010
mov r3, #0	R3 = 00000000 R15(pc) = 00001010	R3 = 00000000 R15(pc) = 00001014
mov r4, #0	R4 = 00000000 R15(pc) = 00001014	R4 = 00000000 R15(pc) = 00001018
mov r6, #0	R6 = 00000000 R15(pc) = 00001018	R6= 00000000 R15(pc) = 0000101c

sub r5, r6,r0	R5 = 00000000 R15(pc) = 0000101c	R5 = 0000300 R15(pc) = 00001020
and r9, r1, #0x80000000	R9 = 00000000 R15(pc) = 00001020	R9 = 80000000 R15(pc) = 00001024
mov r9, r9, lsr #31	R9 = 80000000 R15(pc) = 00001024	R9 = 00000001 R15(pc) = 00001028
cmp r3, #8	R15(pc) = 00001028 N = 0	R15(pc) = 0000102c N = 1
bne razryad_ne_9	R15(pc) = 0000102c	R15(pc) = 00001034
ldr r6, =1	R6= 00000000 R15(pc) = 00001034	R6= 00000001 R15(pc) = 00001038
mov r6, r6, LSL r3	R15(pc) = 00001038	R15(pc) = 0000103c
and r6,r1,r6	R6= 00000001 R15(pc) = 0000103c	R6= 00000000 R15(pc) = 00001040
cmp r6, #0	R15(pc) = 00001040 N = 1 Z = 0 C = 0	R15(pc) = 00001044 N = 0 Z = 1 C = 1
bne zifra_1	R15(pc) = 00001044	R15(pc) = 00001048
b zifra_0	R15(pc) = 00001048	R15(pc) = 0000104c
mov r4,r2	R15(pc) = 0000104c	R15(pc) = 00001050
mov r2, r2, ASR #1	R15(pc) = 00001050	R15(pc) = 00001054
add r3, r3, #1	R3 = 00000000 R15(pc) = 00001054	R3 = 00000001 R15(pc) = 00001058
b if_razryad_eq_9	R15(pc) = 00001058	R15(pc) = 00001028
cmp r3, #8	R15(pc) = 00001028 Z = 1	R15(pc) = 0000102c Z = 0

	C = 1	C = 0
bne razryad_ne_9	R15(pc) = 0000102c	R15(pc) = 00001034
ldr r6, =1	R6 = 00000000 R15(pc) = 00001034	R6 = 00000001 R15(pc) = 00001038
mov r6, r6, LSL r3	R6 = 00000001 R15(pc) = 00001038	R6 = 00000002 R15(pc) = 0000103c
and r6,r1,r6	R6 = 00000002 R15(pc) = 0000103c	R6 = 00000000 R15(pc) = 00001040
cmp r6, #0	R15(pc) = 00001040 Z = 0 C = 0	R15(pc) = 00001044 Z = 1 C = 1
bne zifra_1	R15(pc) = 00001044	R15(pc) = 00001048
b zifra_0	R15(pc) = 00001048	R15(pc) = 0000104c
mov r4,r2	R15(pc) = 0000104c	R15(pc) = 00001050
mov r2, r2, ASR #1	R15(pc) = 00001050	R15(pc) = 00001054
add r3, r3, #1	R3 = 00000001 R15(pc) = 00001054	R3 = 00000002 R15(pc) = 00001058
b if_razryad_eq_9	R15(pc) = 00001058	R15(pc) = 00001028
cmp r3, #8	R15(pc) = 00001028 N = 0 Z = 1 C = 1	R15(pc) = 0000102c N = 1 Z = 0 C = 0
bne razryad_ne_9	R15(pc) = 0000102c	R15(pc) = 00001034
ldr r6, =1	R6 = 00000000 R15(pc) = 00001034	R6 = 00000001 R15(pc) = 00001038
mov r6, r6, LSL r3	R6 = 00000001 R15(pc) = 00001038	R6 = 00000004 R15(pc) = 0000103c

and r6,r1,r6	R15(pc) = 0000103c	R15(pc) = 00001040
cmp r6, #0	R15(pc) = 00001040 N = 1 C = 0	R15(pc) = 00001044 N = 0 C = 1
bne zifra_1	R15(pc) = 00001044	R15(pc) = 0000105c
add r2, r2, r0	R2 = 00000000 R15(pc) = 0000105c	R2 = ffffd000 R15(pc) = 00001060
mov r4, r2	R4 = 00000000 R15(pc) = 00001060	R4 = ffffd000 R15(pc) = 00001064
mov r2, r2, ASR #1	R2 = ffffd000 R15(pc) = 00001064	R2 = ffffe800 R15(pc) = 00001068
add r3, r3, #1	R3 = 00000002 R15(pc) = 00001068	R3 = 00000003 R15(pc) = 0000106c
b if_razryad_eq_9	R15(pc) = 0000106c	R15(pc) = 00001028
cmp r3, #8	R15(pc) = 00001028 N = 0 C = 1	R15(pc) = 0000102c N = 1 C = 0
bne razryad_ne_9	R15(pc) = 0000102c	R15(pc) = 00001034
ldr r6, =1	R6 = 00000004 R15(pc) = 00001034	R6 = 00000001 R15(pc) = 00001038
mov r6, r6, LSL r3	R6 = 00000001 R15(pc) = 00001038	R6 = 00000008 R15(pc) = 0000103c
and r6,r1,r6	R15(pc) = 0000103c	R15(pc) = 00001040
cmp r6, #0	R15(pc) = 00001040 N = 1 C = 0	R15(pc) = 00001044 N = 0 C = 1
bne zifra_1	R15(pc) = 00001044	R15(pc) = 0000105c

add r2, r2, r0	R2 = ffffe800 R15(pc) = 0000105c	R2 = ffffb800 R15(pc) = 00001060
mov r4, r2	R4 = ffffd000 R15(pc) = 00001060	R4 = ffffb800 R15(pc) = 00001064
mov r2, r2, ASR #1	R2 = ffffb800 R15(pc) = 00001064	R2 = ffffdc00 R15(pc) = 00001068
add r3, r3, #1	R3 = 00000003 R15(pc) = 00001068	R3 = 00000004 R15(pc) = 0000106c
b if_razryad_eq_9	R15(pc) = 0000106c	R15(pc) = 00001028
cmp r3, #8	R15(pc) = 00001028 N = 0 C = 1	R15(pc) = 0000102c N = 1 C = 0
bne razryad_ne_9	R15(pc) = 0000102c	R15(pc) = 00001034
ldr r6, =1	R6 = 00000008 R15(pc) = 00001034	R6 = 00000001 R15(pc) = 00001038
mov r6, r6, LSL r3	R6 = 00000001 R15(pc) = 00001038	R6 = 00000010 R15(pc) = 0000103c
and r6,r1,r6	R15(pc) = 0000103c	R15(pc) = 00001040
cmp r6, #0	R15(pc) = 00001040 N = 1 C = 0	R15(pc) = 00001044 N = 0 C = 1
bne zifra_1	R15(pc) = 00001044	R15(pc) = 0000105c
add r2, r2, r0	R2 = ffffdc00 R15(pc) = 0000105c	R2 = ffffac00 R15(pc) = 00001060
mov r4, r2	R4 = ffffb800 R15(pc) = 00001060	R4 = ffffac00 R15(pc) = 00001064
mov r2, r2, ASR #1	R2 = ffffac00	R2 = ffffd600

	R15(pc) = 00001064	R15(pc) = 00001068
add r3, r3, #1	R3 = 00000004 R15(pc) = 00001068 N = 0 C = 1	R3 = 00000005 R15(pc) = 0000106c N = 1 C = 0
b if_razryad_eq_9	R15(pc) = 0000106c	R15(pc) = 00001028
cmp r3, #8	R15(pc) = 00001020 N = 1 C = 0	R15(pc) = 00001024 N = 0 C = 1
bne razryad_ne_9	R15(pc) = 00001028	R15(pc) = 0000102c
ldr r6, =1	R6 = 00000010 R15(pc) = 0000102c	R6 = 00000001 R15(pc) = 00001034
mov r6, r6, LSL r3	R6 = 00000001 R15(pc) = 00001034	R6 = 00000020 R15(pc) = 00001038
and r6,r1,r6	R15(pc) = 00001038	R15(pc) = 0000103c
cmp r6, #0	R15(pc) = 0000103c N = 0 C = 1	R15(pc) = 00001040 N = 1 C = 0
bne zifra_1	R15(pc) = 00001040	R15(pc) = 00001044
add r2, r2, r0	R2 = ffffd600 R15(pc) = 00001044	R2 = ffffa600 R15(pc) = 0000105c
mov r4, r2	R4 = ffffac00 R15(pc) = 0000105c	R4 = ffffa600 R15(pc) = 00001060
mov r2, r2, ASR #1	R2 = ffffa600 R15(pc) = 00001060	R2 = ffffd300 R15(pc) = 00001064
add r3, r3, #1	R3 = 00000005 R15(pc) = 00001064	R3 = 00000006 R15(pc) = 00001068

b if_razryad_eq_9	R15(pc) = 0000106c	R15(pc) = 00001028
cmp r3, #8	R15(pc) = 00001028 N = 1 C = 0	R15(pc) = 0000102c N = 0 C = 1
bne razryad_ne_9	R15(pc) = 0000102c	R15(pc) = 00001034
ldr r6, =1	R6 = 00000020 R15(pc) = 00001034	R6 = 00000001 R15(pc) = 00001038
mov r6, r6, LSL r3	R6 = 00000001 R15(pc) = 00001038	R6 = 00000040 R15(pc) = 0000103c
and r6,r1,r6	R15(pc) = 0000103c	R15(pc) = 00001040
cmp r6, #0	R15(pc) = 00001044	R15(pc) = 00001048
bne zifra_1	R15(pc) = 00001048	R15(pc) = 0000105c
add r2, r2, r0	R2 = ffffd300 R15(pc) = 0000105c	R2 = ffffa300 R15(pc) = 00001060
mov r4, r2	R4 = ffffa600 R15(pc) = 00001060	R4 = ffffa300 R15(pc) = 00001064
mov r2, r2, ASR #1	R2 = ffffa300 R15(pc) = 00001064	R2 = ffffd180 R15(pc) = 00001068
add r3, r3, #1	R3 = 00000006 R15(pc) = 00001068	R3 = 00000007 R15(pc) = 0000106c
b if_razryad_eq_9	R15(pc) = 0000106c	R15(pc) = 00001028
cmp r3, #8	R15(pc) = 00001028 N = 0 C = 1	R15(pc) = 0000102c N = 1 C = 0
bne razryad_ne_9	R15(pc) = 0000102c	R15(pc) = 00001034
ldr r6, =1	R6 = 00000040 R15(pc) = 00001034	R6 = 00000001 R15(pc) = 00001038

mov r6, r6, LSL r3	R6 = 00000001 R15(pc) = 00001038	R6 = 00000080 R15(pc) = 0000103c
and r6,r1,r6	R15(pc) = 0000103c	R15(pc) = 00001040
cmp r6, #0	R15(pc) = 00001040 N = 1 C = 0	R15(pc) = 00001044 N = 0 C = 1
bne zifra_1	R15(pc) = 00001044	R15(pc) = 0000105c
add r2, r2, r0	R2 = ffffd180 R15(pc) = 0000105c	R2 = ffffa180 R15(pc) = 00001060
mov r4, r2	R4 = ffffa300 R15(pc) = 00001060	R4 = ffffa180 R15(pc) = 00001064
mov r2, r2, ASR #1	R2 = ffffa180 R15(pc) = 00001064	R2 = ffffd0c0 R15(pc) = 00001068
add r3, r3, #1	R3 = 00000007 R15(pc) = 00001068	R3 = 00000008 R15(pc) = 0000106c
b if_razryad_eq_9	R15(pc) = 0000106c	R15(pc) = 00001028
cmp r3, #8	R15(pc) = 00001028 N = 0 C = 1	R15(pc) = 0000102c N = 1 C = 0
bne razryad_ne_9	R15(pc) = 0000102c	R15(pc) = 00001030
mov r4, r2	R4 = ffffa0c0 R15(pc) = 00001030	R4 = ffffd060 R15(pc) = 00001070
cmp r9, #1	R15(pc) = 00001070 N = 1 Z = 0 C = 0	R15(pc) = 00001074 N = 0 Z = 1 C = 1
beq less_then_0	R15(pc) = 00001074	R15(pc) = 00001078

add r4, r4, r5	R4 = ffffd060 R15(pc) = 00001078	R4 = 00000060 R15(pc) = 00001080
more_then_0:	R15(pc) = 00001080	R15(pc) = 00001088

6. Результаты работы программы

Результаты работы программы с различными примерами показаны на рисунках 6 - 17.

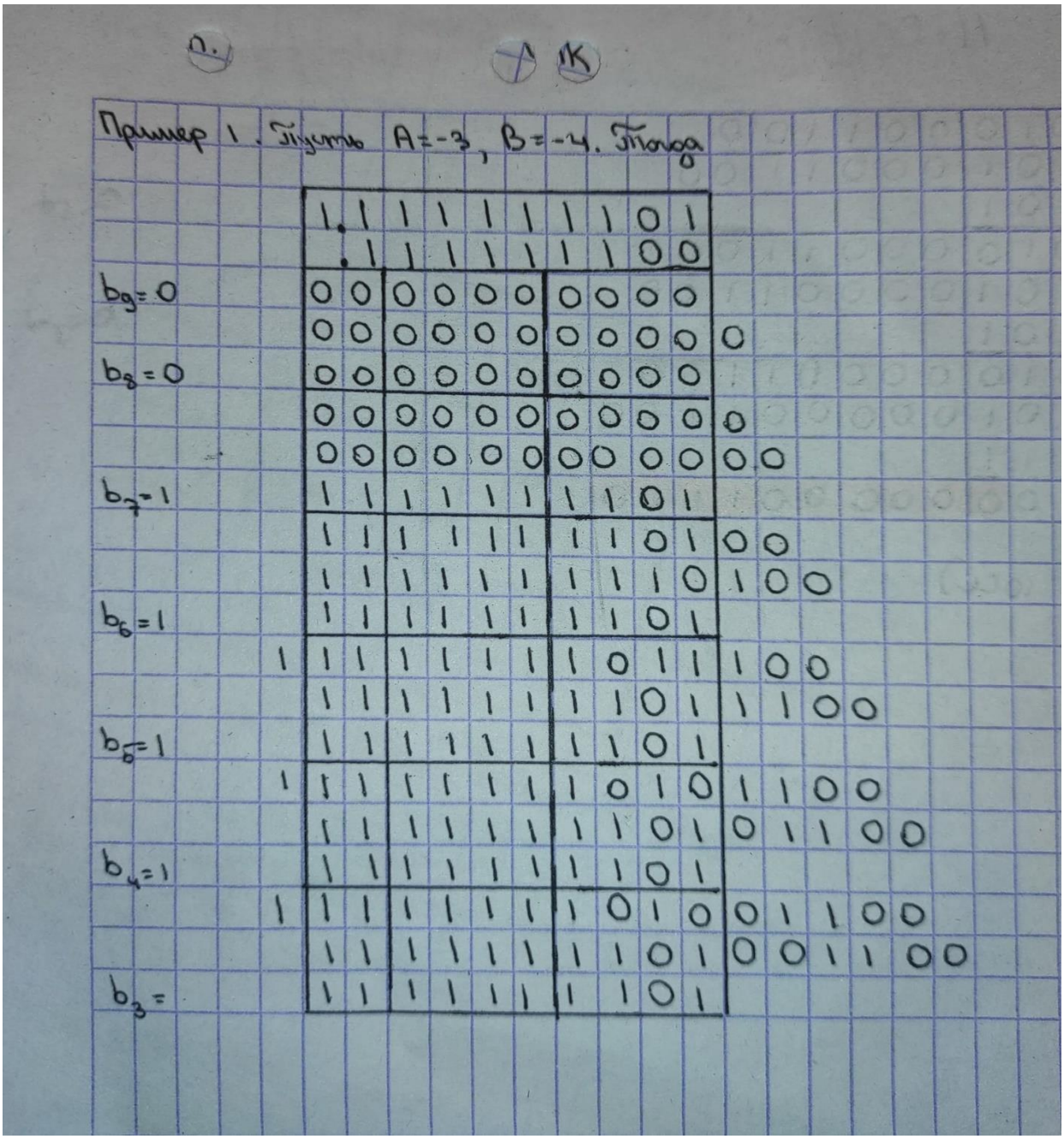

Рисунок 6 – Пример 1

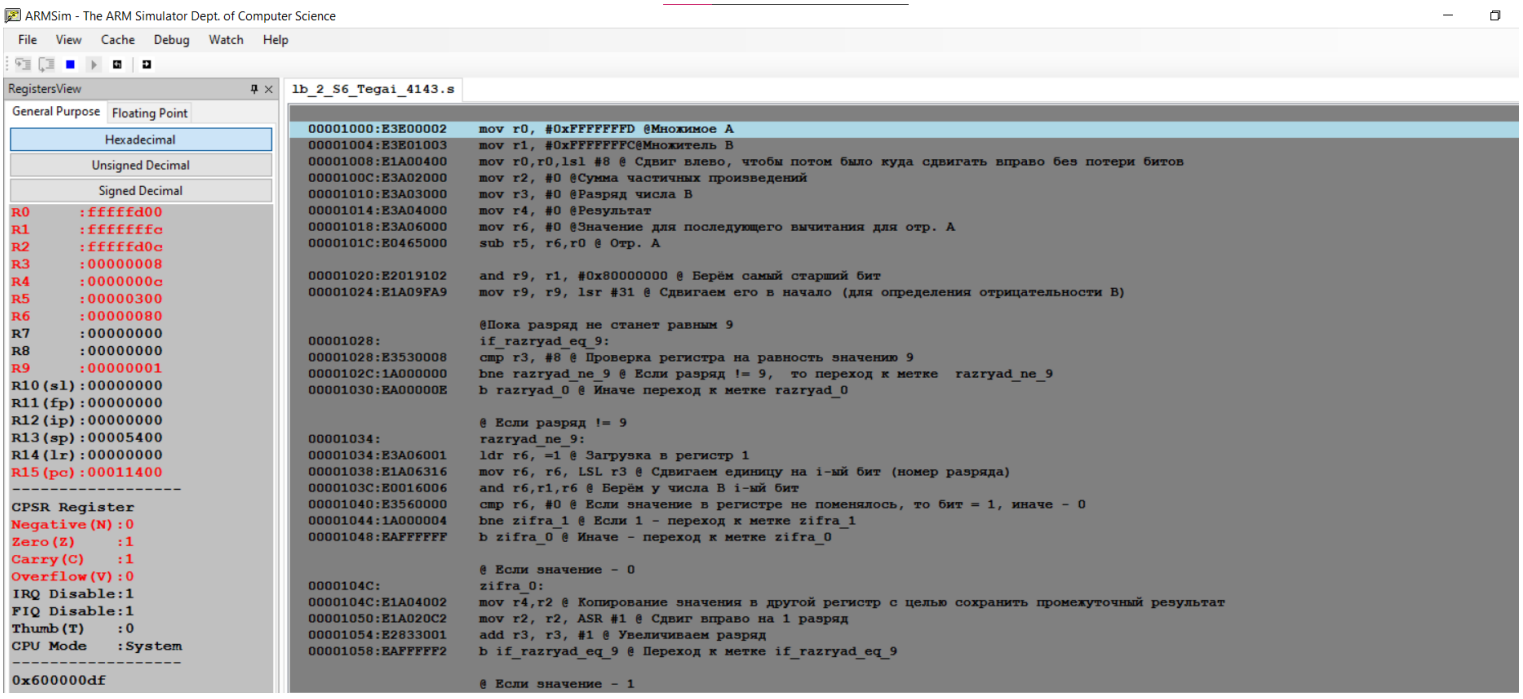

Рисунок 7 – Пример 1

Рисунок 8 – Результат работы программы для примера 1

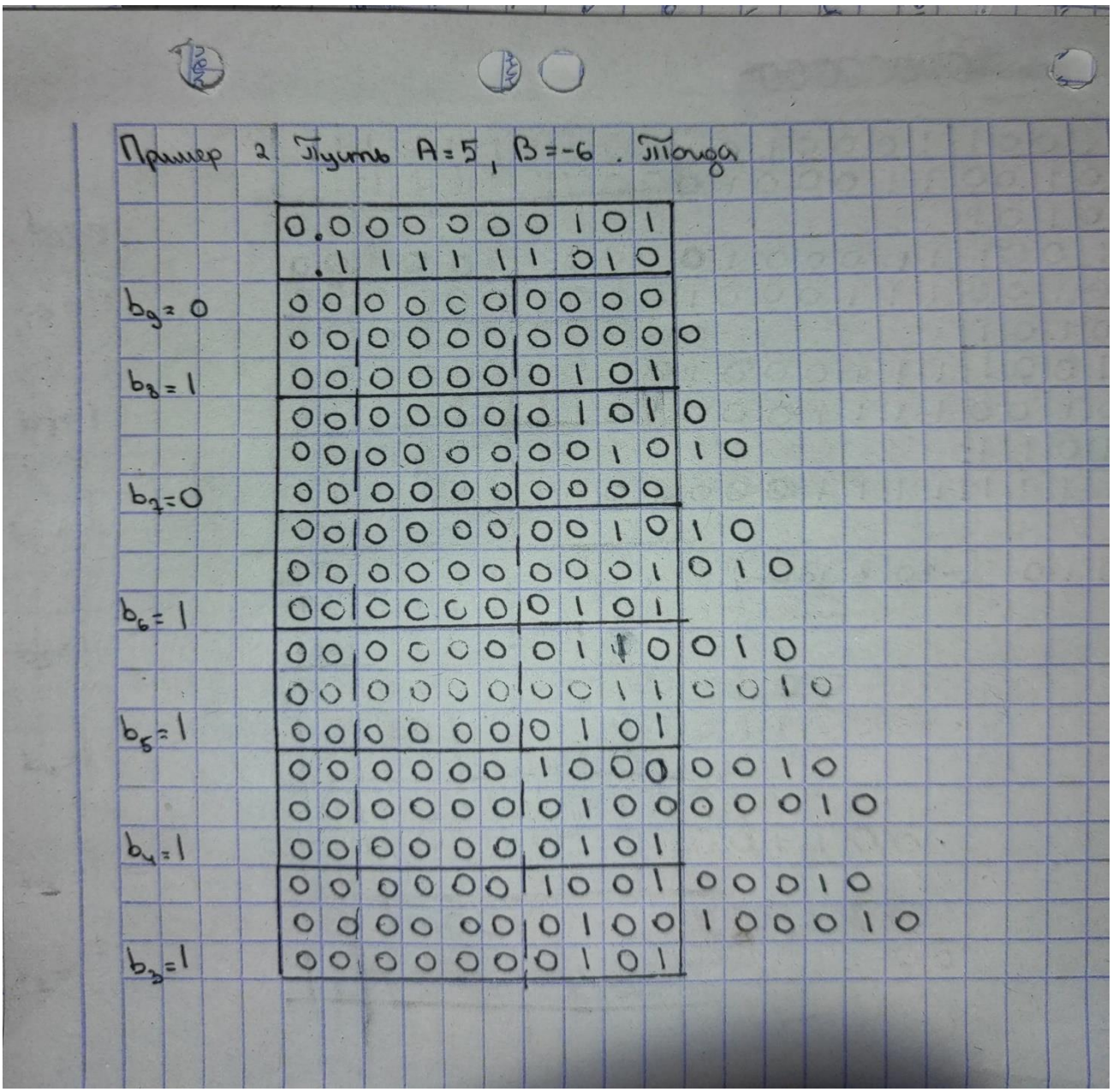

Рисунок 9 – Пример 2

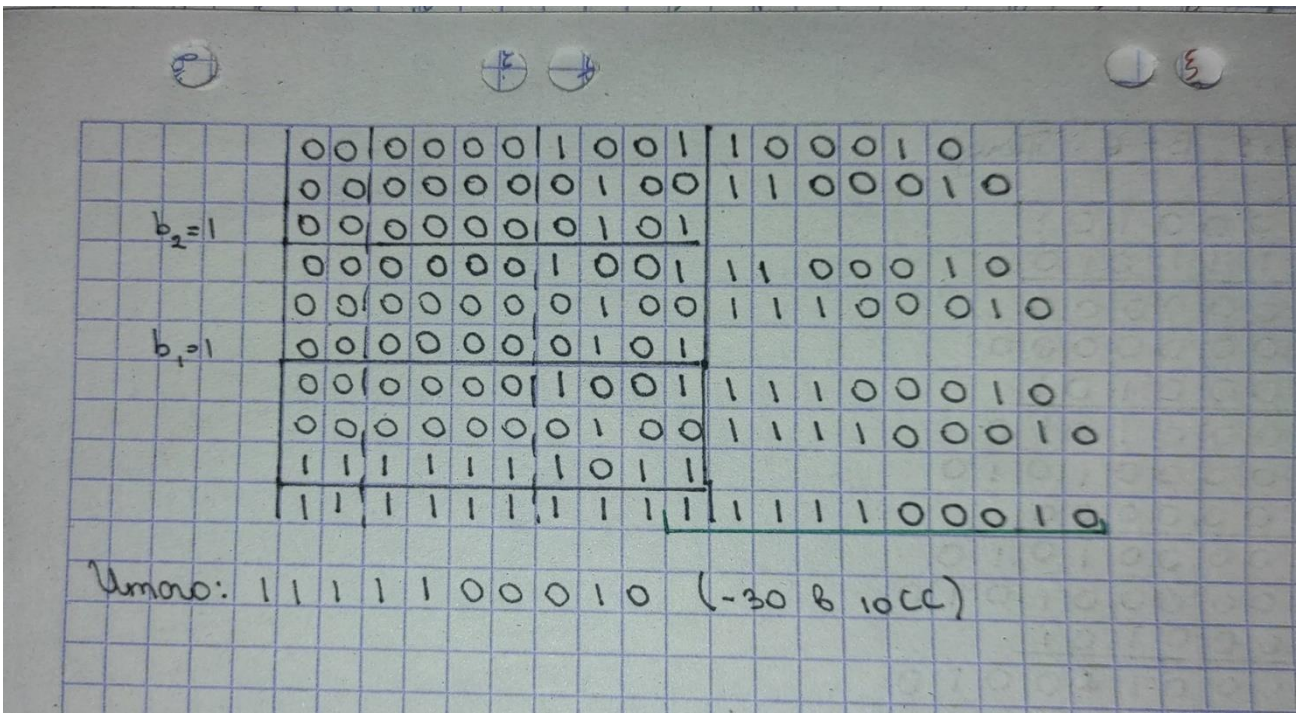

Рисунок 10 – Пример 2

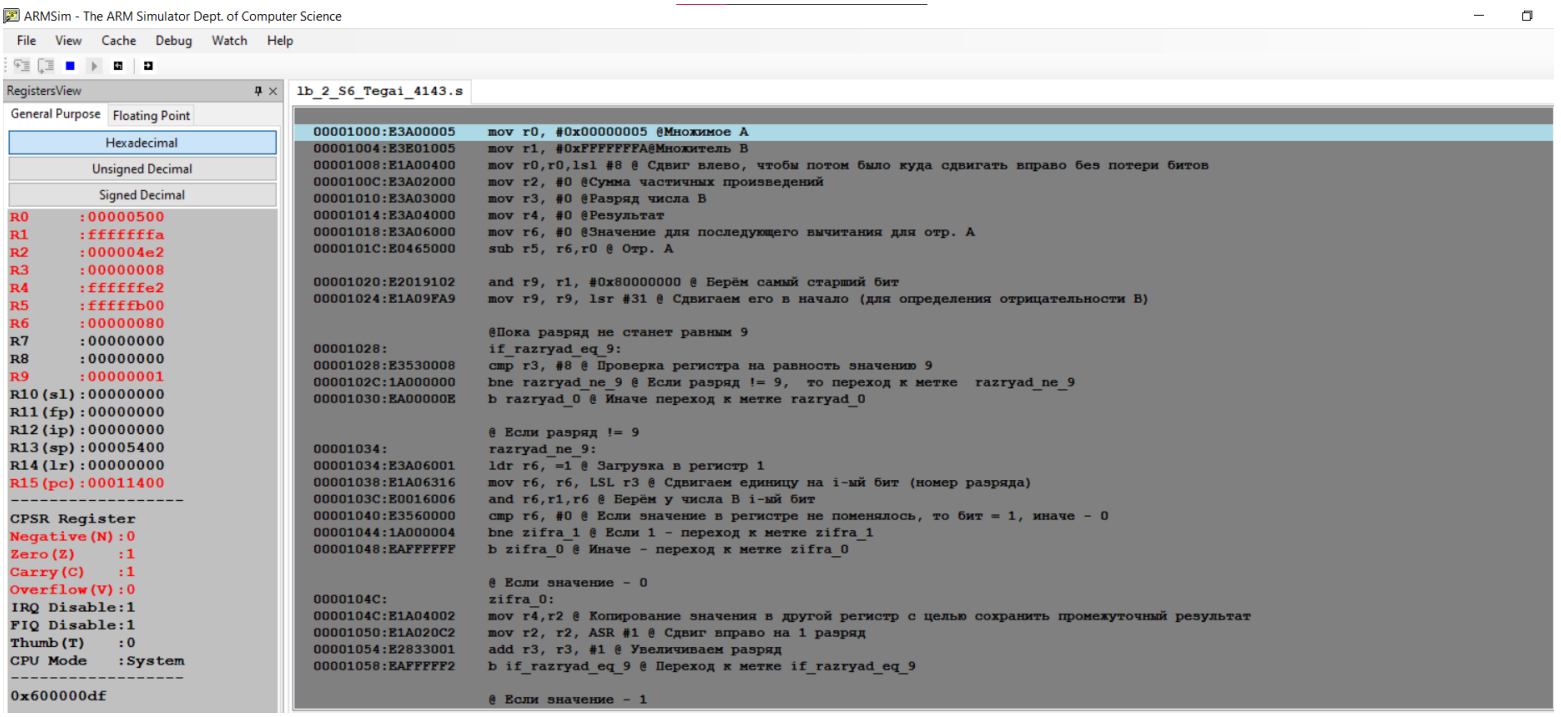

Рисунок 11 – Результат работы программы для примера 2

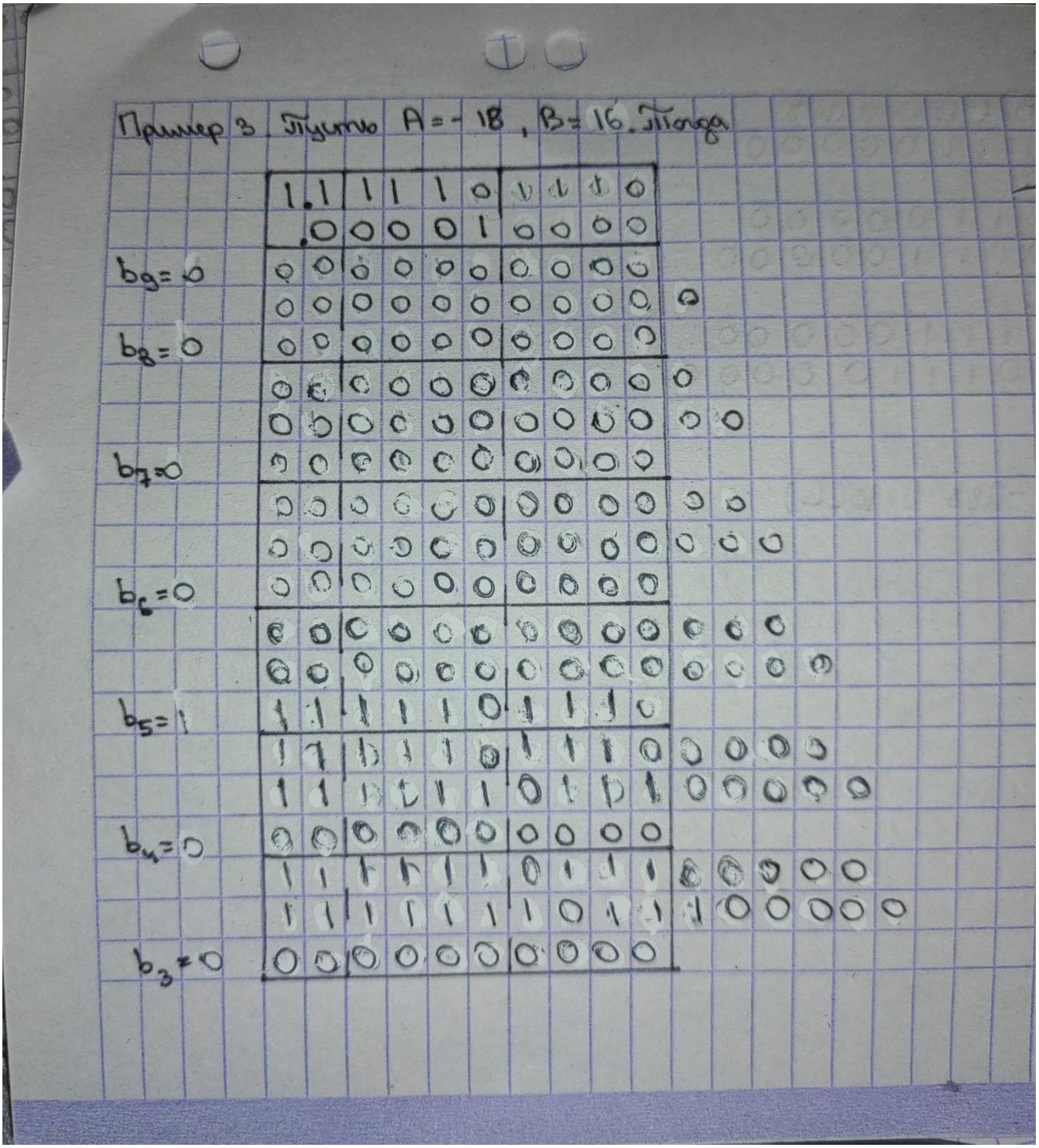

Рисунок 12 – Пример 3

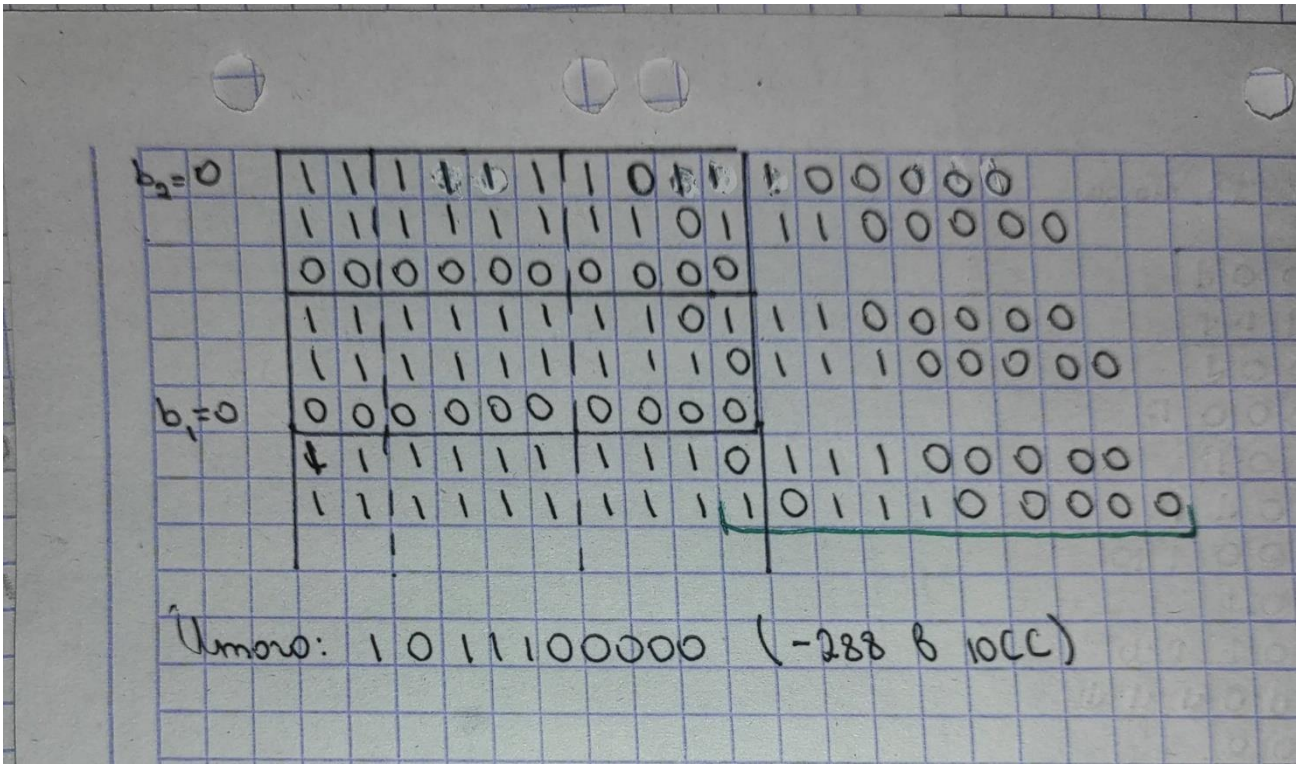

Рисунок 13 – Пример 3

File View Cache Debug Watch Help

RegistersView

General Purpose | Floating Point

Hexadecimal | Unsigned Decimal | Signed Decimal

```
R0       :ffffee00
R1       :00000010
R2       :ffffffe0
R3       :00000008
R4       :ffffffe0
R5       :00001200
R6       :00000000
R7       :00000000
R8       :00000000
R9       :00000000
R10(sl):00000000
R11(fp):00000000
R12(ip):00000000
R13(sp):00005400
R14(lr):00000000
R15(pc):00011400
------------------
CPSR Register
Negative(N):1
Zero(Z)    :0
Carry(C)   :0
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T)   :0
CPU Mode   :System
------------------
0x800000df
```

lb_2_S6_Tegai_4143.s

```
00001000:E3E00011    mov r0, #0xFFFFFFEE @Множимое A
00001004:E3A01010    mov r1, #0x00000010@Множитель B
00001008:E1A00400    mov r0,r0,lsl #8 @ Сдвиг влево, чтобы потом было куда сдвигать вправо без потери битов
0000100C:E3A02000    mov r2, #0 @Сумма частичных произведений
00001010:E3A03000    mov r3, #0 @Разряд числа B
00001014:E3A04000    mov r4, #0 @Результат
00001018:E3A06000    mov r6, #0 @Значение для последующего вычитания для отр. A
0000101C:E0465000    sub r5, r6,r0 @ Отр. A

00001020:E2019102    and r9, r1, #0x80000000 @ Берём самый старший бит
00001024:E1A09FA9    mov r9, r9, lsr #31 @ Сдвигаем его в начало (для определения отрицательности B)

                     @Пока разряд не станет равным 9
00001028:            if_razryad_eq_9:
00001028:E3530008    cmp r3, #8 @ Проверка регистра на равность значению 9
0000102C:1A000000    bne razryad_ne_9 @ Если разряд != 9,  то переход к метке  razryad_ne_9
00001030:EA00000E    b razryad_0 @ Иначе переход к метке razryad_0

                     @ Если разряд != 9
00001034:            razryad_ne_9:
00001034:E3A06001    ldr r6, =1 @ Загрузка в регистр 1
00001038:E1A06316    mov r6, r6, LSL r3 @ Сдвигаем единицу на i-ый бит (номер разряда)
0000103C:E0016006    and r6,r1,r6 @ Берём у числа B i-ый бит
00001040:E3560000    cmp r6, #0 @ Если значение в регистре не поменялось, то бит = 1, иначе - 0
00001044:1A000004    bne zifra_1 @ Если 1 - переход к метке zifra_1
00001048:EAFFFFFF    b zifra_0 @ Иначе - переход к метке zifra_0

                     @ Если значение - 0
0000104C:            zifra_0:
0000104C:E1A04002    mov r4,r2 @ Копирование значения в другой регистр с целью сохранить промежуточный результат
00001050:E1A020C2    mov r2, r2, ASR #1 @ Сдвиг вправо на 1 разряд
00001054:E2833001    add r3, r3, #1 @ Увеличиваем разряд
00001058:EAFFFFF2    b if_razryad_eq_9 @ Переход к метке if_razryad_eq_9

                     @ Если значение - 1
```

Рисунок 14 – Результат работы программы для примера 3

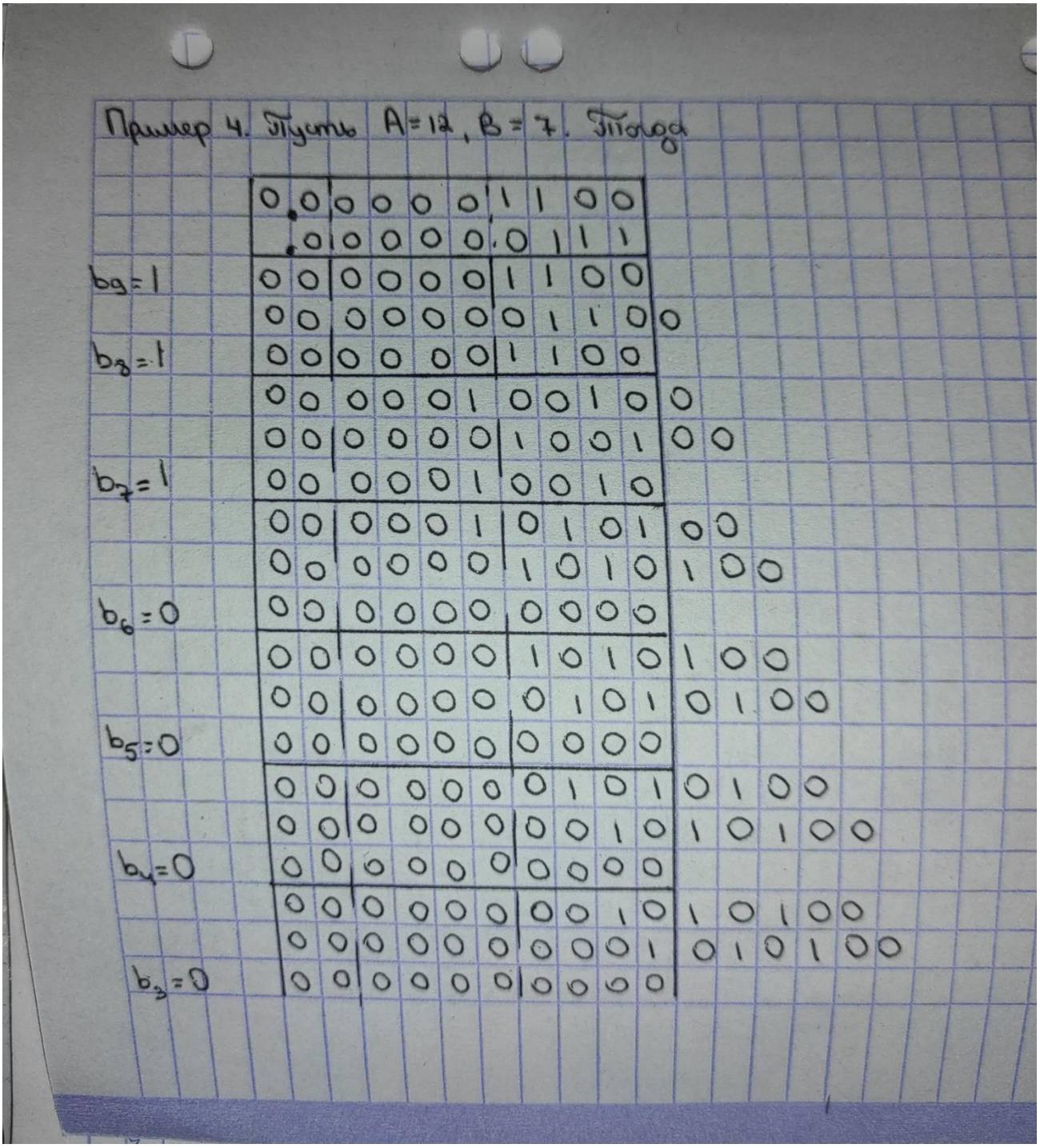

Рисунок 15 – Пример 4

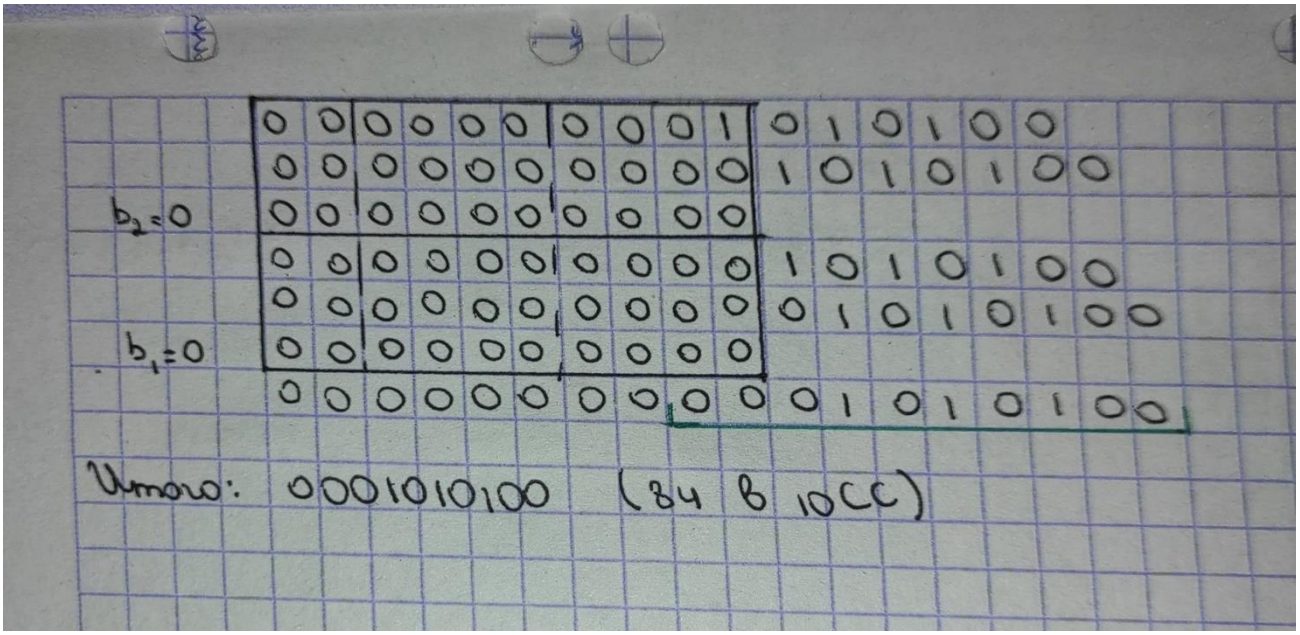

Рисунок 16 – Пример 4

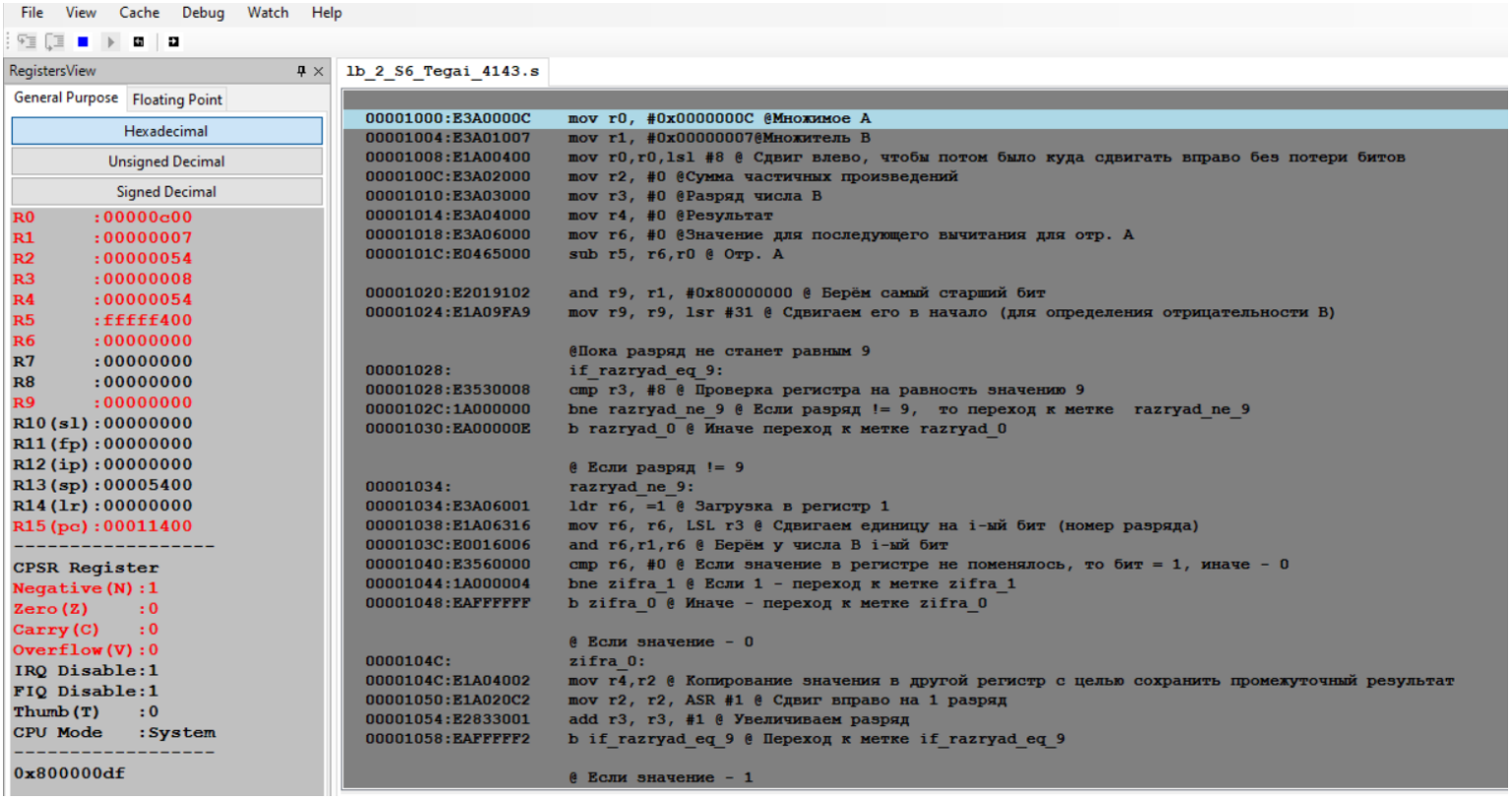

Рисунок 17 – Результат работы программы для примера 4