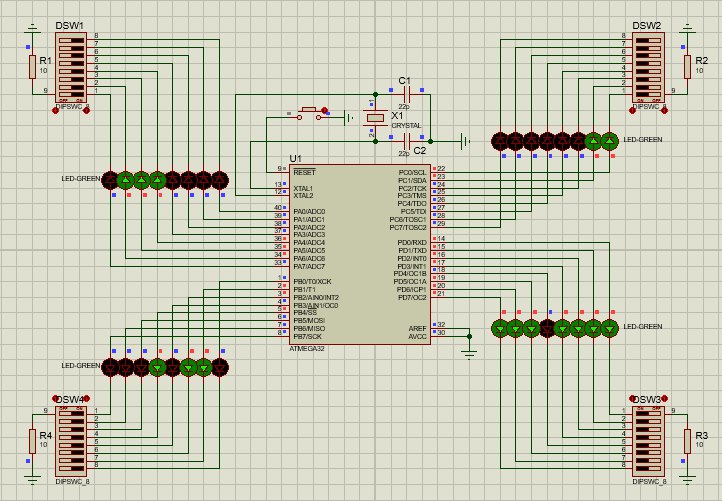
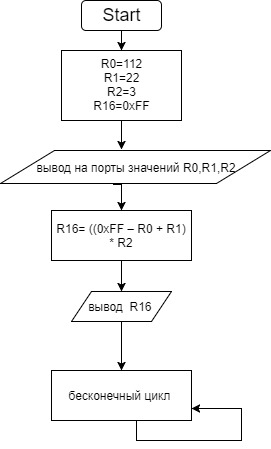
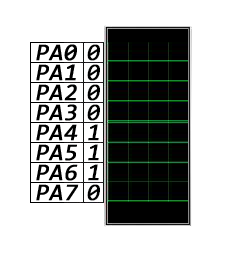
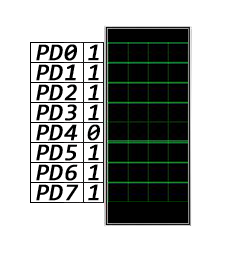
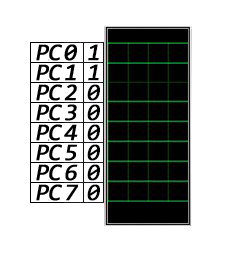
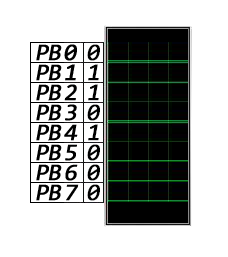
# Схема тестовой установки



# Блок-схема алгоритма работы программы



# Временные диаграммы логических сигналов на портах МК (фрагмент)

Порт А 112 порт В 22 порт С 3 порт D 239 (495)

# Алгоритм выполнения задействованных команд (конструкций) ассемблера

В коде были задействованы следующие команды:

SER Rd – команда устанавливаем все биты регистра Rd единицами

OUT – Команда сохраняет данные регистра Rr в регистровом файле пространства I/O, подготавливает названный порт к выводу данных (DDRA), после просто выводит данные через указанный порт (PORTA, R21)

LDI Rd, x – устанавливает значение регистр Rd константой x

MOV Rd, Rr –Копирование регистров Rd ← Rr

SUB Rd, Rr –Вычитание Rd ← Rd - Rr

ADD Rd, Rr –Сложение Rd ← Rd + Rr

MUL Rd, Rr –Умножение беззнаковых чисел, R1:R0 ← Rd x Rr (результат кладется в пару R0:R1)

RJMP k , k∈[-2048;2047] – Относительный безусловный переход P C ← PC + k + 1

# Результаты работы

Написали код для вывода элементов и подсчета выражения----

LDI R16, 112 ;a1 инициализация костант

MOV R0, R16 перемещение констант,т.к. невозможно сразу класть их значения в R0-2

LDI R16, 22 ;a2

MOV R1, R16

LDI R16, 3 ;a3

MOV R2, R16

LDI R16, 0xFF

SER R20 ; устанавливает в R20 все единицы: R20 = 0xFF /

OUT DDRA, R20 ; сконфигурировали все порты как порты вывода

OUT DDRB, R20

OUT DDRC, R20

OUT DDRD, R20

OUT PORTA, R0 ;вывод а0, а1 и а2

OUT PORTB, R1

OUT PORTC, R2

SUB R16,R0 ; R16 = 0хFF - a0

ADD R16,R1 ; R16 = R16 + a1

MUL R16,R2 ; R16 = R16 \* a2 (результат кладется в пару R0:R1)

OUT PORTD, R0 ;вывод результата R0 = (0хFF - a0 + a1) \* a2

ddd:

RJMP ddd

Анализ содержимого hex файла:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| длина записи | адрес команды | тип записи | данные и команды | | | | | | | | контрольная сумма |
| :02 | 0000 | 02 | 0000 |  | | | | | | | FC |
| 1 | 2 | 3 | 4 |  | | | | | | | 5 |
| :10 | 0000 | 00 | 00E7 | 002E | 06E1 | 102E | 03E0 | 202E | 0FEF | 4FEF | 49 |
| 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| :10 | 0010 | 00 | 4ABB | 47BB | 44BB | 41BB | 0BBA | 18BA | 25BA | 0019 | 4F |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 |
| :08 | 0020 | 00 | 010D | 029D | 02BA | FFCF |  | | | | A1 |
| 30 | 31 | 32 | 33 | 34 | 35 | 36 |  |  |  |  | 37 |
| :00 | 0000 | 01 |  | | | | | | | | FF |
| 38 | 39 | 40 |  |  | | | | | | | 41 |

1я и последняя строки отвечают за начало и конец файла и не содержат информации и данных. байты в каждой отдельной команде записывается в обратном порядке (сначала второй, потом первый)

1-длина первой записи - 2 байта.

2-7-адреса первых команд

3-тип, соответствующий адресу расширенного сегмента

5,17,29,37,41-контрольная сумма, рассчитывается так чтобы сумма всех байтов в записи была равна 0.

6,18-длина записи 16 битов (максимально возможная)

8,20-тип-запись содержащая данные

9-1110 0111 0000 0000 - загрузка константы 112 в регистр R16 (код команды 1110.KKKK.dddd.KKKK) (т.к. команда работает с регистрами начиная с 16)

10-10 1110 0000 0000 - Копирование регистров Rd←Rr, R0←R16(код команды 0010.11rd.dddd.rrrr)

11,13,15 - аналогично 9.

12,14 - аналогично 10.

16 -1110 1111 0100 1111 - SER ,Установка всех битов регистра R20 = 0xFF (код команды 1110.1111.dddd.1111) (т.к. команда работает с регистрами начиная с 16, и 100(2)=4(10), то 16+4=20й регистр

19-адрес, начиная с которого далее записываются команды - 16. т.к. предыдущие команды заняли первые 15

21-1011 101**1 0100** 1010 Запись значения регистра R**20** в порт 1(А) (1011.1AAr.rrrr.AAAA)

22,23,24 - аналогичный вывод значения R20 на порты 2,3,4 (B,C,D)

25 - 1011 101**0 0000** 1011 вывод значения R**0** на A (1011.1AAr.rrrr.AAAA)

26,27 -аналогичный вывод значений R1, R2 на порты 2,3(B,C)

28 - 0001 100**1 0000** 0000- вычитание Rd ← Rd - Rr , в R**16**  кладется разница R**16**  -R0 (0001.10rd.dddd.rrrr)

30-длина записи, сод. 8 битов

31-адрес, начиная с которого далее записываются команды - 32. т.к. предыдущие команды заняли первые 31

33 - 0000 11**0**1 0000 **0001** Сложение Rd ← Rd + Rr регостров R16+**R1** (0000.11rd.dddd.rrrr)

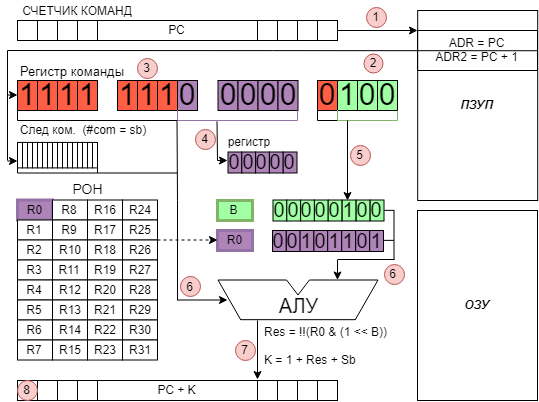
34-1001 11**0**1 0000**0010** Умножение беззнаковых чисел R2 x R16 по схеме R1:R0 ← Rd x Rr , результат кладется в пару R0:R1(1001.11rd.dddd.rrrr)

35-1011 1010 0000 0010 вывод значения R**0** на D (1011.1AAr.rrrr.AAAA)

36-1100 1111 1111 1111 - Относительный безусловный переход PC ← PC + k + 1 (1100.kkkk.kkkk.kkkk)

38-41 пустая запись, означающая конец файла

Порядок выполнения команды SBRS R0, 4 ЦП МК в части определения этапов выполнения, задействованных узлов, пересылаемых данных и управляющих сигналов:



1) Передача памяти программ (ПЗУП) адреса текущей команды

2) Извлечение команды из ПЗУП и запись в регистр команды

3) Декодирование команды – определение типа операции и формата

4) Извлечение адресов операндов (номеров регистров общего назначения (РОН))

5) Извлечение операндов из команды

6) Передача операндов в АЛУ

7) Расчет шага счетчика команд

8) Увеличение значения счётчика команд

000000 e700 LDI R16, 112 //x1

000001 2e00 MOV R0, R16

000002 e106 LDI R16, 22 //x2

000003 2e10 MOV R1, R16

000004 e003 LDI R16, 3 //x3

000005 2e20 MOV R2, R16

000006 ef0f LDI R16, 0xFF

000007 ef4f SER R20

000008 bb4a OUT DDRA, R20 ;сконфигурировали все порты как порты вывода

000009 bb47 OUT DDRB, R20

00000a bb44 OUT DDRC, R20

00000b bb41 OUT DDRD, R20

00000c ba0b OUT PORTA, R0

00000d ba18 OUT PORTB, R1

00000e ba25 OUT PORTC, R2

00000f 1900 SUB R16,R0

000010 0d01 ADD R16,R1

000011 9d02 MUL R16,R2

000012 ba02 OUT PORTD, R0 ;вывод результата R0 = (0хFF - a0 + a1) \* a2

ddd:

000013 cfff RJMP ddd

ddd: //адрес метки ddd 000013