1. ФОРМУЛИРОВКА ЗАДАНИЯ

1. Написать программу на языке ассемблера в соответствии с вариантом.
2. Скомпилировать программу в среде Atmel Studio, прошить программу в ПЗУ МК и проверить её работоспособность.
3. Для «.lss»-файла выписать адреса всех меток и перечислить используемые форматы команд в части состава и размера операндов.
4. Для «.hex»-файла разобрать структуру в части определения количества записей и машинных команд.
5. Взять команду ассемблера в соответствии с вариантом и представить порядок её выполнения ЦП МК в части определения этапов выполнения, задействованных узлов, пересылаемых данных и управляющих сигналов.

2. СХЕМА ЛАБОРАТОРНОЙ УСТАНОВКИ

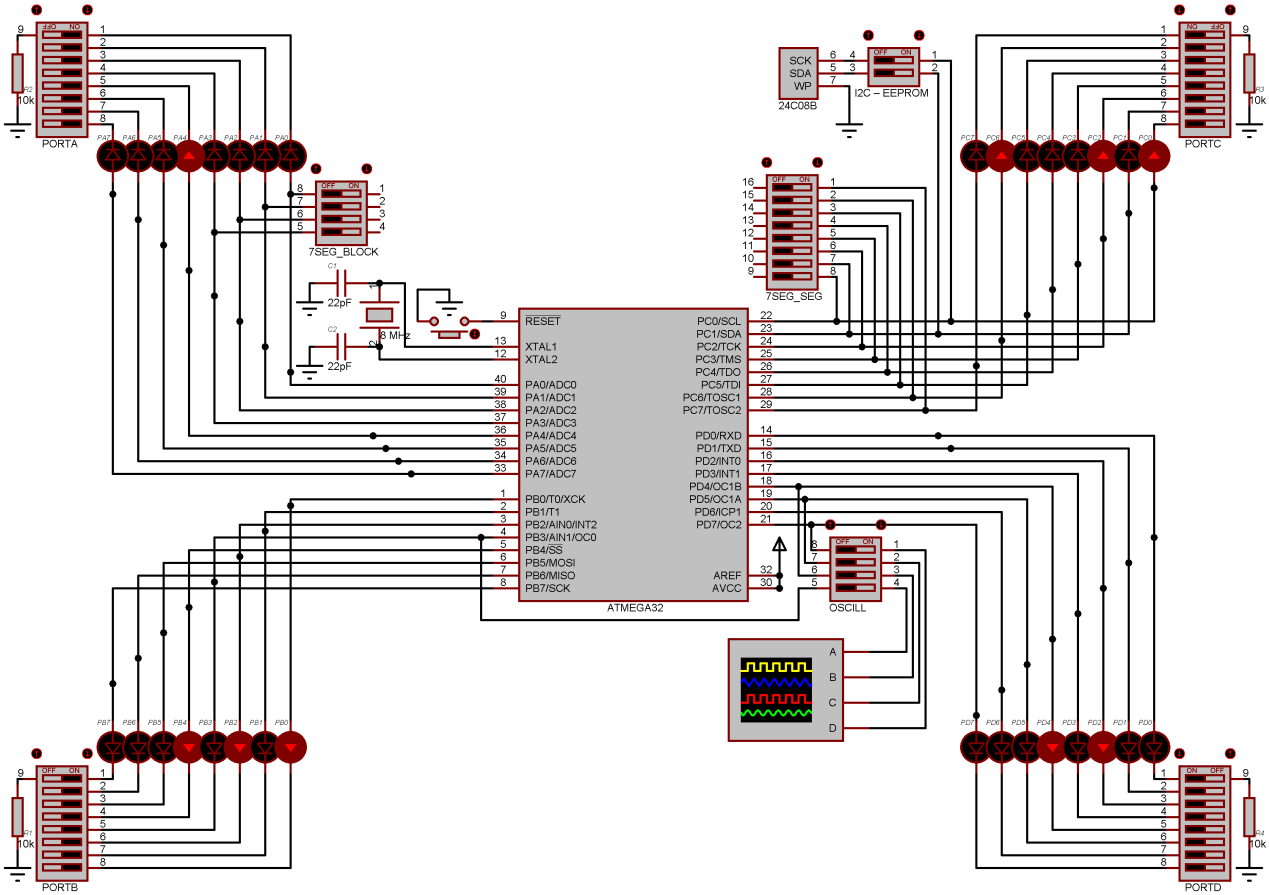

Рисунок 1 — Схема лабораторной установки.

3. БЛОК-СХЕМА АЛГОРИТМА РАБОТЫ ПРОГРАММЫ

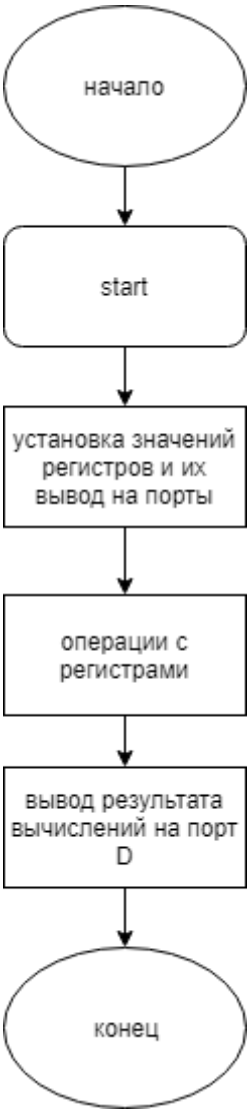

Рисунок 2 — Блок-схема алгоритма работы программы.

4. АЛГОРИТМ ВЫПОЛНЕНИЯ ЗАДЕЙСТВОВАННЫХ КОМАНД (КОНСТРУКЦИЙ) АССЕМБЛЕРА

Команда	Тип	Описание	Операнды (ограничения)	КОП
ser	Битовая	Установка всех битов регистра	Rd d∈[16;31]	1110.1111.dddd.1111
clr	Битовая	Очистка всех битов регистра	Rd	0010.01dd.dddd.dddd
out	Пересылка	Запись значения регистра в порт	P, Rr	1011.1AAr.rrrr.AAAA
ldi	Пересылка	Загрузка константы	Rd, K d∈[16;31]	1110.KKKK.dddd.KKKK
sub	Арифметическая	Вычитание	Rd, Rr	0001.10rd.dddd.rrrr
add	Арифметическая	Сложение	Rd, Rr	0000.11rd.dddd.rrrr
mul	Арифметическая	Умножение беззнаковых чисел	Rd, Rr	1001.11rd.dddd.rrrr
rjmp	Безусловный переход	Относительный безусловный переход	k k∈[-2048;2047]	1100.kkkk.kkkk.kkkk

5. РЕЗУЛЬТАТЫ РАБОТЫ

1. Анализ .lss-файла.

```
000000 e100     ldi R16, 16 ; a1
000001 e115     ldi R17, 21 ; a2
000002 e425     ldi R18, 69 ; a3
000003 ef6f     ser R22 ; r22=1111 1111
; Переносим содержимое r22 в регистр ввода\вывода портов A-D, устанавливаем на
вывод.
000004 bb6a     out DDRA, R22
000005 bb67     out DDRB, R22
000006 bb64     out DDRC, R22
000007 bb61     out DDRD, R22
; Выводим значение регистров R16-R18 (a1-a3) на порты A-C.
000008 bb0b     out PORTA,R16
000009 bb18     out PORTB,R17
00000a bb25     out PORTC,R18
00000b ef3f     ldi R19, 255 ;  R3 = 255 = 0xFF
00000c 1b30     sub R19, R16 ; R3 = R3 - R0 = 0xFF - a1
00000d 0f31     add R19, R17 ; R3 = R3 + R1 = (0xFF - a1) + a2
00000e 2400     clr R0
00000f 0e03     add R0, R19
000010 9e02     mul R0, R18 ; R3 = R3 * R2 = (0xFF - a1 + a2) * a3
000011 ba02     out PORTD, R0 ;выводим рез-ат вычислений на порт D
```

2. Анализ .hex-файла.

```
020000020000FC
1000000000E115E125E46FEF6ABB67BB64BB61BB30
100010000BBB18BB25BB3FEF301B310F0024030E79
06002000029E02BAEDCFC2
00000001FF
```

количество байт в записи, RECLEN
смещение, определяющее адрес загрузки данных, LOAD OFFSET
тип записи, RECTYP ('00' Data Record - запись, содержащая данные, '01' End of File Record - запись, сигнализирующая о конце файла)
данные для загрузки в память, DATA
байт контрольной суммы, CHKSUM

3. Анализ команды ассемблера SBI DDRC, 3.

Устанавливает заданный бит в регистр I/O.

Каждый бит регистра DDRC отвечает за режим работы соответствующей ножки (PC0-PC7). Записав 0 в соответствующий бит, мы настраиваем соответствующую ножку в режим входа. То есть можно считывать с неё входящие с внешних устройств данные. Записав в соответствующий (3-й) бит 1, мы устанавливаем ножку в режим вывода. Кол-во тактов: 2.

Команда SBI DDRC, 3 в .lss файле:
000000 9aa3 *sbi ddrc, 3*

В .hex файле:
:020000020000FC
:04000000A39AFECFF2
:00000001FF

16-разрядный код операции:

9	A	A	3
1001	1010	1010	0011

6. ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие типы данных может содержать адресная часть команды?

Данная часть команды может содержать различные типы данных в зависимости от назначения команды и используемой адресации. Так, при непосредственной адресации в адресном поле команды располагается сам операнд, при прямой адресации в адресной части команды лежит адрес операнда в памяти, при косвенной же адресации адресная часть указывает адрес ячейки памяти, в который находится указатель на операнд.

2. Какое назначение имеют регистры числа и регистры адреса в ЦП ЭВМ?

Регистр числа – регистр временного хранения, используется в процессе вычисления. Также ЭВМ может использовать данные регистры при выполнении каких-либо логических операций. Поскольку эти типы регистров часто используются при работе ЭВМ, пересылка из памяти и в память происходит быстрее. Регистры адреса – содержат адрес данных в памяти. Адреса могут представлять собой часть команд или данные.

3. Перечислите основные функции, выполняемые устройством управления?

Устройство управления ядра AVR можно представить в виде совокупности нескольких блоков: блок генерации адресов инструкций, блок выборки инструкции, блок декодирования инструкций, блок обработки исключений, блок переходов, блок отладки.

4. Поясните, что делают команды со следующими машинными кодами: 1111.0110.0001.0001, 1011.1101.0101.0101 и 0000.0011.0101.0001?

* BRNE (1111.01kk.kkkk.k001) - условный переход. Где kk.kkkk.k – биты переменной, принимающей значения k∈[-64;63]. Выполняет переход если неравно (Z=0). То есть, если (Z = 0) тогда запись в регистр: PC ← PC + k + 1. Выполнение занимает 1 или 2 такта в зависимости от значения регистра.

* OUT (1011.1AAr.rrrr.AAAA) - операция пересылки. Запись значения регистра в порт. То есть запись регистра Rr в порт P: P ← Rr. Время выполнения занимает 1 такт, флаги не используются.

* MOVW (0000.0001.dddd.rrrr) - операция пересылки, работает с двумя регистрами Rd, Rr, где d, r – чётные. Выполняет копирование регистровой пары, то есть Rr+1:Rr в регистровую пару Rd+1:Rd. Время выполнения занимает 1 такт.

5. Чем вызваны ограничения допустимых значений номеров регистров и диапазонов констант в некоторых командах микроконтроллера ATmega32 (например, SUBI, RJMP, BRCC)?

Так как КОП для машинной команды ограничен по размеру (4 байта) и как раз в нем хранятся значения операндов, то при отсутствии ограничений на

значения регистров или констант, для их записи просто не хватило бы разрядов. Например в команде RJMP (КОП 1100.kkkk.kkkk.kkkk) под ее единственный операнд выделено только 3 байта (в четвертом байте хранятся сведения о типе операции), за счет этого и накладываются ограничения.

7. ВЫВОДЫ ПО ЛАБОРАТОРНОЙ РАБОТЕ

В ходе проделанной работы была написана требуемая программа на языке ассемблера в соответствии с вариантом, проверена ее работоспособность на отладочной плате.

Также проведено исследование lss-файла, исследована его структура, найдены адреса меток.

Исследована структура hex-файла, определено количество записей и машинных команд, выделены цветом блоки записей, проведено ознакомление с форматом Intel HEX.

ПРИЛОЖЕНИЕ 1

КОММЕНТИРОВАННЫЙ ЛИСТИНГ ПРОГРАММЫ ДЛЯ МК НА ЯЗЫКЕ АССЕМБЛЕРА

```
start:
   ldi R16, 16 ; a1
   ldi R17, 21 ; a2
   ldi R18, 69 ; a3

   ser R22 ; r22=1111 1111
   ; Переносим содержимое r22 в регистр ввода\вывода портов A-D, устанавливаем на
вывод.
   out DDRA, R22
   out DDRB, R22
   out DDRC, R22
   out DDRD, R22
   ; Выводим значение регистров R16-R18 (a1-a3) на порты A-C.
   out PORTA,R16
   out PORTB,R17
   out PORTC,R18

   ldi R19, 255 ;  R3 = 255 = 0xFF
   sub R19, R16 ; R3 = R3 - R0 = 0xFF - a1
   add R19, R17 ; R3 = R3 + R1 = (0xFF - a1) + a2
   clr R0
   add R0, R19
   mul R0, R18 ; R3 = R3 * R2 = (0xFF - a1 + a2) * a3
   out PORTD, R0 ;выводим рез-ат вычислений на порт D

   rjmp start
```