1 Формулировка задания

1. В базовом примере (см. ниже) заменить фрагмент кода, выделенный зелёным, на фрагмент из своего варианта.

2. Создать новый проект в Atmel Studio на языке ассемблера, разместить исходный код из п.1 в файле проекта. Сформировать «.lss»-файл и «.hex»-файл. Записать «.hex»-файл в контроллер с помощью программы AVRFlash.

3. Изучить архитектуру и систему команд микроконтроллера ATmega32. Разобраться в алгоритме работы текущей программы. Убедиться в правильности работы программы.

4. Определить зависимость количества тактов, за которое выполняется заменённый блок кода, от констант x и y.

5. Вычислить значения констант x и y при которых блок кода «delay» будет выполняться ровно 0,01 секунды (при тактовой частоте 8 МГц). При невозможности обеспечения точной величины задержки необходимо дополнить блок кода соответствующим количеством команд NOP (на месте закомментированной команды NOP в базовом примере).

6. Изучить сформированный «.lss»-файл, выписать адреса всех меток программы, перечислить используемые форматы команд в части состава и размерности операндов.

7. Изучить структуру сформированного «.hex»-файла, определить количество записей в файле и количество машинных слов программы.

8. Взять команду ассемблера в соответствии с вариантом и описать порядок её выполнения внутри центрального процессора: определить этапы командного цикла, задействованные узлы МК, состав пересылаемых данных и управляющих сигналов.

2 Схема лабораторной установки

Схема лабораторной установки показана на Рисунок 1.

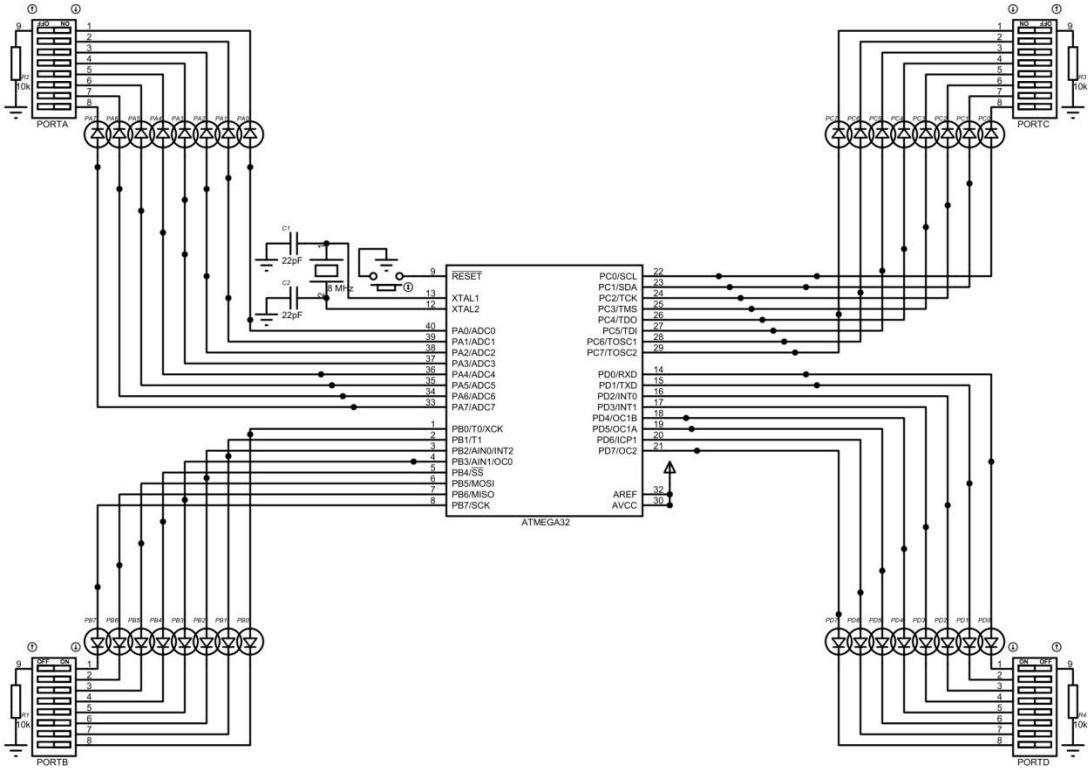

Рисунок 1 – Схема лабораторной установки

3 Блок-схема алгоритма работы программы

Блок-схема алгоритма работы программы показана на Рисунок 2.

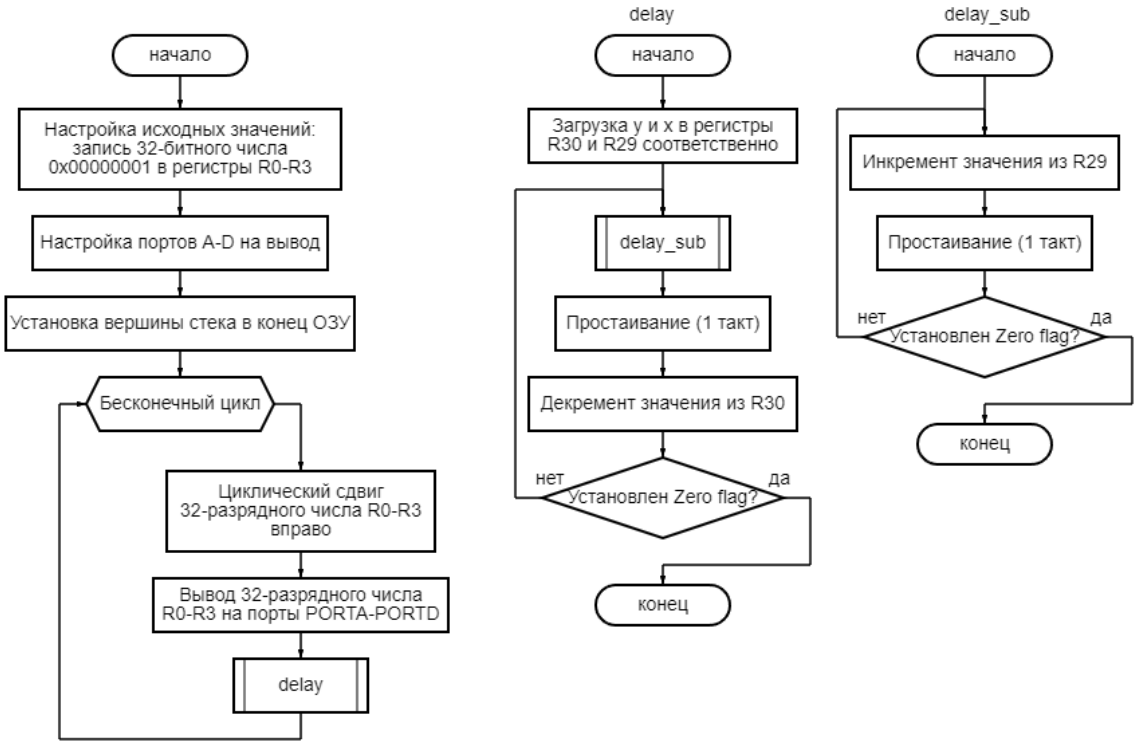

Рисунок 2 – Блок-схема алгоритма работы программы

4 Ход работы

4.1 Определение зависимости количества тактов от констант x и y

$$N_{\text{вн}_1} = (1_{INC} + 1_{NOP} + 2_{BRNE}) * (2^8 - x - 1) + (1_{INC} + 1_{NOP} + 1_{BRNE})$$
$$= 1023 - 4x$$

$$N_{\text{вн}_x} = (1_{INC} + 1_{NOP} + 2_{BRNE}) * (2^8 - 1) + (1_{INC} + 1_{NOP} + 1_{BRNE}) = 1023$$

$$N_{\text{общ}} = 2_{LDI} + \left(N_{\text{вн}_1} + 1_{NOP} + 1_{DEC} + 2_{BRNE}\right) + \left(N_{\text{вн}_x} + 1_{NOP} + 1_{DEC} + 2_{BRNE}\right)$$
$$* (y - 2) + \left(N_{\text{вн}_x} + 1_{NOP} + 1_{DEC} + 1_{BRNE}\right) + 4_{RET} = -4x + 1027y + 5$$
$$= -4 * 153 + 1027 * 200 + 5 = 204\,794$$

Время выполнения: $t = \frac{N}{T} = \frac{204794}{8000000} = 0{,}02559$ с $\approx 25{,}6$ мс

Чтобы время задержки составило 10 мс:

$$-4x + 1027y + 5 = 80\,000$$

должно выполняться для подобранных x и y.

Если x = 0, y = 78, то:

$$1027 * 78 + 5 = 80\,111$$

Превышает значение 80 000.

Если x = 28, y = 78, то:

$$-4 * 28 + 1027 * 78 + 5 = 79\,999$$

Если в конец функции паузы при этих значениях добавить команду NOP, то время задержки составит 10 мс (N = 80 000).

4.2 Анализ сформированного «.lss»-файла

Адреса меток:

- delay: 0x000002,
- delay_sub: 0x000004,
- reset: 0x00000c,
- loop: 0x00001b.

Используемые форматы команд:

#	Опер.	Описание	Операция	КОП	Флаги	Такт
JMP	k	Прямой безусловный переход	PC ← k	1001.010k.kkkk.110k.kkkk.kkkk.kkkk.kkkk	–	3
LDI	Rd, K (d∈[16;31])	Загрузка константы	Rd ← K	1110.KKKK.dddd.KKKK	–	1
INC	Rd	Инкремент	Rd ← Rd + 1	1001.010d.dddd.0011	Z,N,V,S	1
NOP		Пустая команда		0000.0000.0000.0000	–	1
BRNE	K (k∈[-64;63])	Переход если неравно (Z=0)	if (Z = 0) then PC ← PC + k + 1	1111.01kk.kkkk.k001	–	1/2
DEC	Rd	Декремент	Rd ← Rd - 1	1001.010d.dddd.1010	Z,N,V,S	1
RET		Возврат из подпрограммы	PC ← Stack	1001.0101.0000.1000	–	4
MOV	Rd, Rr	Копирование регистров	Rd ← Rr	0010.11rd.dddd.rrrr	–	1
CLR	Rd	Очистка всех битов регистра	Rd ← Rd ⊕ Rd	0010.01dd.dddd.dddd	Z,N,V,S	1

SER	Rd (d∈[16;31])	Установка всех битов регистра	Rd ← $FF	1110.1111.dddd.1111	–	1
OUT	P, Rr	Запись значения регистра в порт	P ← Rr	1011.1AAr.rrrr.AAAA	–	1
BST	Rr, b	Сохранить бит регистра Rr во флаг T регистра SREG	T ← Rr(b)	1111.101d.dddd.0bbb	T	1
LSR	Rd	Логический сдвиг вправо	Rd(n) ← Rd(n+1), Rd(7) ← 0, C←Rd(0)	1001.010d.dddd.0110	Z,C,N,V,S	1
ROR	Rd	Циклический сдвиг вправо (через флаг переноса)	Rd(7)← C, Rd(n)← Rd(n+1), C←Rd(0)	1001.010d.dddd.0111	Z,C,N,V,S	1
BLD	Rd, b	Считать флаг T регистра SREG в бит регистра Rd	Rd(b) ← T	1111.100d.dddd.0bbb	–	1

CAL L	k	Прямой вызов подпрограмы	Stack ← PC, PC ← k	1001.010k.kkkk.111k. kkkk.kkkk.kkkk.kkkk	–	4

4.3 Анализ сформированного «.hex»-файла

Структура сформированного файла:

```
:020000020000FC
:100000000C940C00EEE4DCE1D3950000E9F700006D
:10001000EA95D1F70000089541E0042E4427142EFC
:10002000242E342E4FEF4ABB47BB44BB41BB48E0B4
:100030004EBF4FE54DBF00FA369427941794079 4AE
:1000400037F80BBA18BA25BA32BA0E940200F3CFB9
:00000001FF
```

: – поле маркера начала записи, RECORD MARK

_ – количество байт данных в записи, RECLEN

_ – смещение, определяющее адрес загрузки данных, LOAD OFFSET

_ – тип записи, RECTYP

_ – данные для загрузки в память, DATA

_ – байт контрольной суммы, CHKSUM

_ – расширенный сегментный адрес, USBA

Количество записей в файле – 7.

4.4 Порядок выполнения команды ассемблера

Этапы выполнения команды ADD R12, R8:

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: 0000.11rd.dddd.rrrr

4. Извлечение адресов операндов (R12, R8)

5. Извлечение операндов из команды (непосредственная адресация)

6. Передача операндов в АЛУ

7. Выполнение операции суммирования

8. Передача результата операции суммирования в регистр R12

9. Увеличение значения счётчика команд

5 Ответы на контрольные вопросы

1. Укажите, в чём проявляются признаки RISC-архитектуры в микроконтроллере ATmega32. В чём преимущества и недостатки приведённых особенностей?

Признаки RISC-архитектуры в микроконтроллере ATmega32 проявляются в небольшом наборе инструкций, их простом и единообразном формате и ограниченном количестве режимов адресации. Преимуществом этих особенностей являются снижение сложности декодирования инструкций и увеличение плотности кода. Недостатками являются ограниченность возможностей микроконтроллера и увеличение объёма кода, необходимого для реализации сложных алгоритмов (из-за ограниченного количества режимов адресации), а также менее эффективное использование памяти (набор инструкций небольшой, поэтому для выполнения определенных задач может потребоваться больше инструкций).

2. От чего зависит время выполнения команд SBIS и BRLT? Приведите примеры кода (до 3-10 команд каждый), приводящие к различной продолжительности выполнения указанных команд

Время выполнения команды SBIS зависит от количества инструкций, пропущенных после её выполнения. Например:

```
waitset: sbis $10,0 ; пропускает следующую инструкцию, если бит в
порту D равен 1 (2 такта)
rjmp waitset ; бит в порту D был равен 0, инструкция sbis заняла 1
такт
nop ; продолжение
```

Время выполнения команды BRLT зависит от значения битов N и V регистра SREG (результат предыдущего сравнения операндов с учётом их знака). Например:

```
cp r16,r1 ;
brlt less ; совершает прыжок на метку less и занимает 2 такта в
случае, если r16 < r1
 ... ; не совершает прыжок, если r16 >= r1, и занимает 1 такт
 less: nop ;
```

3. Укажите команды и их аргументы в виде имён регистров общего назначения, портов ввода-вывода и констант в десятичной или шестнадцатеричной системе счисления для следующих машинных слов: 1001 0110 0111 1111, 1111 0011 1110 1101, 1010 1010 0011 1010 и 1001 1010 0011 0110?

1001 0110 0111 1111 – ADIW R30, 0x1F

1111 0011 1110 1101 – BRHS -2

1010 1010 0011 1010 – STD Y+50, R3

1001 1010 0011 0110 – SBI ADCSRA, 6

4. Приведите пример выполнения циклического сдвига вправо 8-разрядного числа (с переходом младшего разряда в старший) без использования флага T. Приведите пример из трёх машинных команд, обеспечивающих сложение 24-разрядного числа с 24-разрядной константой.

Делаем сдвиг вправо при помощи команды ROR, при этом в конце на место старшего бита запишем значение флага C.

Для сложения нужно воспользоваться командой ADC, которая выполняет сложение с переносом.

```
ADD R0, R3
ADC R1, R4
ADC R2, R5
```

5. Обоснуйте, чем вызваны ограничения допустимых значений номеров регистров и диапазонов констант в некоторых командах микроконтроллера ATmega32?

Ограничения допустимых значений номеров регистров и диапазонов констант в некоторых командах микроконтроллера ATmega32 обусловлены ограниченным количеством битов в кодировке команды. В ATmega32 используется архитектура набора инструкций (ISA), которая определяет количество битов, доступных для кодирования различных частей инструкции, таких как КоП, номера регистров и непосредственные значения.

6 Выводы

В ходе лабораторной работы были рассмотрены основные характеристики микроконтроллера ATmega32, изучены его регистры и ассемблерные команды. Также была изучена структура .hex и .lss файлов. Был рассмотрен алгоритм выполнения команды ADD, а также был проведен расчет времени выполнения фрагмента кода, соответствующего варианту.

ПРИЛОЖЕНИЕ А

Листинг программы на языке ассемблера

```
.def TMP = R20

.org $000
   JMP reset ; Указатель на начало программы

; Функция паузы
delay:
   LDI R30, 78; y
   LDI R29, 28; x
delay_sub:
   INC R29 ; нач
   NOP
   BRNE delay_sub ; вн цикл кон
   NOP
   DEC R30
   BRNE delay_sub ; внешн цикл кон
   NOP
   RET

; Начальная настройка
reset:
; настройка исходных значений
   LDI  TMP, 0x01;
   MOV  R0, TMP
   CLR  TMP;
   MOV  R1, TMP
   MOV  R2, TMP
   MOV  R3, TMP
; настройка портов ввода-вывода
   SER  TMP ; 0xFF
   OUT  DDRA, TMP ; Вывод
   OUT  DDRB, TMP ; Вывод
   OUT  DDRC, TMP ; Вывод
```

```
    OUT  DDRD, TMP ; Вывод
; Установка вершины стека в конец ОЗУ
    LDI  TMP, HIGH(RAMEND) ; Старшие разряды адреса
    OUT  SPH, TMP
    LDI  TMP, LOW(RAMEND) ; Младшие разряды адреса
    OUT  SPL, TMP

; Основной цикл
loop:
; Циклический сдвиг 32-разрядного числа R0-R3
    BST R0, 0 ; сохранение младшего бита во флаге T
    LSR R3 ; логический сдвиг вправо
    ROR R2 ; циклический сдвиг вправо
    ROR R1 ; циклический сдвиг вправо
    ROR R0 ; циклический сдвиг вправо
    BLD R3, 7 ; заполнение 7 бита значением из флага T
; Вывод 32-разрядного числа R0-R3 на порты PORTA-PORTD
    OUT PORTA, R0
    OUT PORTB, R1
    OUT PORTC, R2
    OUT PORTD, R3
; Пауза
    CALL delay;
; Возврат в начало основного цикла
    RJMP loop ;
```