1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт компьютерных наук и кибербезопасности
5. **Кафедра «Высшая школа кибербезопасности»**

**ЛАБОРАТОРНАЯ РАБОТА № 1**

1. «**Основы программирования МК ATmega32**»
2. по дисциплине «Аппаратные средства вычислительной техники»
3. Выполнил
4. студент гр. 5131001/30003 Шевчук Н.Е.

<*подпись*>

Преподаватель Макаров А.С.

<*подпись*>

1. Санкт-Петербург
2. 2025

**Оглавление**

[1 Цель работы 3](#_Toc191506864)

[2 Ход работы 3](#_Toc191506865)

[2.1 Схема установки 3](#_Toc191506866)

[2.2 Блок-схема алгоритма работы программы 3](#_Toc191506867)

[2.3 Зависимость количества тактов, за которое выполняется заменённый блок кода, от констант x и y 3](#_Toc191506868)

[2.4 Анализ .lss файла 4](#_Toc191506869)

[2.5 Анализ .hex файла 5](#_Toc191506870)

[2.6 Порядок выполнения ассемблерной команды внутри процессора 6](#_Toc191506871)

[2.7 Машинные слова 6](#_Toc191506872)

[2.7.1 1334 6](#_Toc191506873)

[2.7.2 FD31 6](#_Toc191506874)

[3 Ответы на контрольные вопросы 6](#_Toc191506875)

[4 Вывод 9](#_Toc191506876)

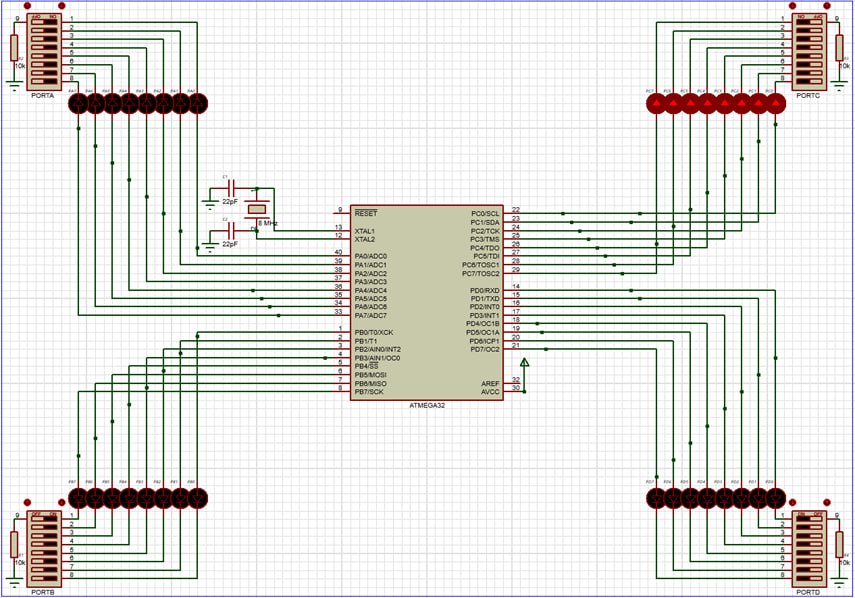
[Приложение 11](#_Toc191506877)

# Цель работы

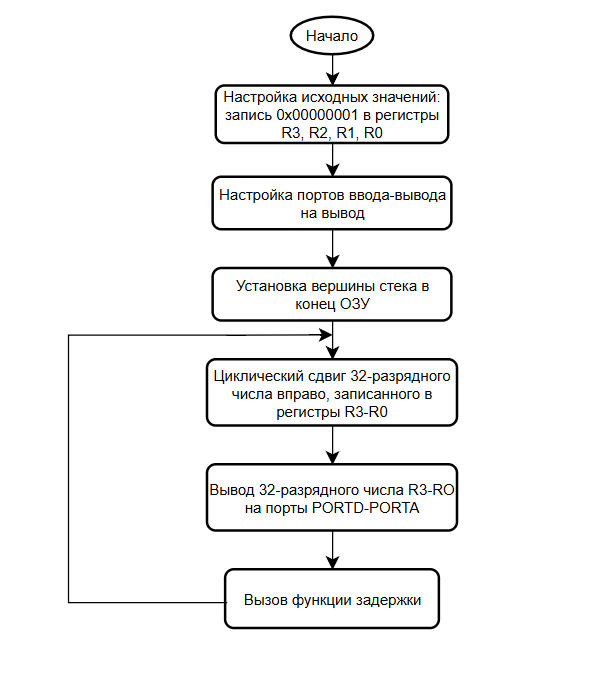
Изучить основы программирования микроконтроллеров семейства Atmel ATmega в среде разработки Atmel Studio. Познакомиться с системой команд микроконтроллера ATmega32.

# Ход работы

## Схема установки



## Блок-схема алгоритма работы программы



## Зависимость количества тактов, за которое выполняется заменённый блок кода, от констант x и y

|  |  |
| --- | --- |
| Команда | Число тактов |
| LDI | 1 - загрузка |
| DEC | 1 - декремент |
| BRNE | 1 – сравнение  1/0 - переход |
| NOP | 1 – простой |
| INC | 1 - инкремент |

Число тактов до первого выхода из внутреннего цикла:

Число тактов до последующих выходов из внутреннего цикла:

Общее число тактов:

Для значений x = 153, y = 200 (принимаем y = 55, т.к. значение R30 инкрементируется):

Чтобы блок кода delay выполнялся ровно 0,01 секунды, необходимо, чтобы:

= 8000000\*0,01 = 80000

Наиболее близкая снизу к числу 81279 сумма равна 81276 и получается при значениях x = 102, y = 63. Но так как y инкрементируется во внешнем цикле, необходимо взять значение y = 255 – 63 = 192.

Соответственно, необходимо добавить 81279-81276 = 3 такта, т.е. три команды NOP. Тогда измененный блок кода будет выглядеть так:

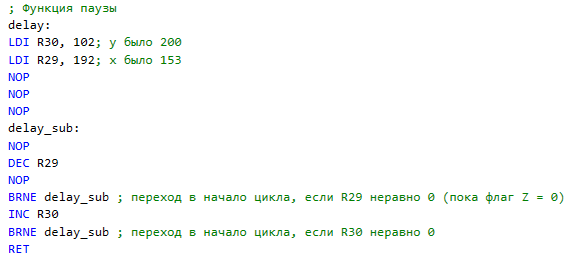


Рисунок 1. Измененный блок кода, выполняющийся ровно 0,01 c

## Анализ .lss файла

Адреса меток:

delay: 0x000002

delay\_sub: 0x000007

reset: 0x00000e

loop: 0x00001d

Используемые команды:

|  |  |  |
| --- | --- | --- |
| Команда | Операнды | Размерность операндов, бит |
| JMP | k | k = 22 |
| LDI | Rd, k | Rd =4, k = 8 |
| NOP | - | - |
| DEC | Rd | Rd = 5 |
| BRNE | k | k = 6 |
| INC | Rd | Rd = 5 |
| RET | - | - |
| MOV | Rd, Rr | Rd = 5, Rr =5 |
| CLR | Rd | Rd = 10 |
| SER | Rd | Rd = 4 |
| OUT | P, Rr | P = 6, Rr = 5 |
| BST | Rr, b | Rr = 5, b = 3 |
| LSR | Rd | Rd = 5 |
| ROR | Rd | Rd = 5 |
| BLD | Rd, b | Rd = 5, b = 3 |
| CALL | k | k = 12 |
| RJMP | k | k = 12 |

## Анализ .hex файла

:020000020000FC

:100000000C940E00E8ECD9E90000000000000000AC

:10001000DA950000E1F7E395D1F7089541E0042E69

:100020004427142E242E342E4FEF4ABB47BB44BB2B

:1000300041BB48E04EBF4FE54DBF00FA36942794D0

:100040001794079437F80BBA18BA25BA32BA0E9437

:040050000200F3CFE8

:00000001FF

: – RECORD MARK – начало записи

00 – RECLEN, задаёт количество байт полезной информации

00 – LOAD OFFSET, смещение, определяющие куда будет записан первый байт данных

00 – RECTYP, задаёт тип записи

00 – DATA, данные для записи, каждая пара – байт данных

00 – CHKSUM, контрольная сумма

00 – USBA, расширенный сегментный адрес

Количество записей: 8

Количество машинных слов: 42

## Порядок выполнения ассемблерной команды внутри процессора

Команда CLR R7

1. Передача памяти программ (ПЗУП) адреса текущей команды
2. Извлечение команды из ПЗУП и запись в регистр команды
3. Декодирование команды – определение типа операции (XOR) и формата (работа с регистрами)
4. Извлечение адресов операндов – оба операнда R7
5. Извлечение операндов из регистров общего назначения (РОН)
6. Передача операндов в арифметико-логическое устройство (АЛУ)
7. Выполнение операции R7 ⊕ R7, результат всегда 0
8. Обновление флагов состояния (Z=1, остальные зависят от архитектуры)
9. Запись результата (0) обратно в регистр R7
10. Увеличение значения счётчика команд

## Машинные слова

### 1334

В архитектуре AVR команда CPSE Rd, Rr (сравнение, пропуск следующей команды если равно) имеет следующий формат: 0001 00rd dddd rrrr. Rd (регистр назначения) занимает 5 бит (биты d). Rr (регистр-операнд) занимает 5 бит (биты r).

Значит, искомая ассемблерная команда: CPSE R19, R20

### FD31

В архитектуре AVR инструкция SBRC Rr, b (пропуск следующей команды, если бит очищен) имеет следующий формат: 1111 110r rrrr 0bbb. Rr (регистр) занимает 5 бит (биты d), b (номер бита, который проверяется) занимает 3 бита.

Значит, искомая ассемблерная команда: SBRC R19, 1

При добавлении полученных команд в код и открытии .lss файла получаем искомые машинные слова:



Рисунок 2. Проверка через .lss файл

# Ответы на контрольные вопросы

1. *Перечислите основные узлы микроконтроллера ATmega32 и укажите их назначение.*

* Постоянное запоминающее устройство программ
* Хранит программу, загруженную в микроконтроллер.
* Является энергонезависимой, т.е. сохраняет данные при выключении питания.
* В ATmega32 – 32 КБ Flash-памяти.
* Основная память (данных)
* Используется для временного хранения данных и переменных во время работы программы.
* В ATmega32 – 2 КБ SRAM (оперативной памяти).
* При выключении питания данные стираются.
* Порты вывода
* Позволяют микроконтроллеру взаимодействовать с внешними устройствами.
* Могут работать как входы (считывание данных) и выходы (управление внешними устройствами).
* В ATmega32 – 4 порта по 8 бит (всего 32 линии ввода-вывода).
* Арифметико-логическое устройство
* Выполняет арифметические (сложение, вычитание и др.) и логические (AND, OR, XOR) операции.
* В ATmega32 – 8-разрядное АЛУ.
* Поддерживает работу с регистрами.
* Устройство управления
* Контролирует выполнение программы и управляет всеми узлами микроконтроллера.
* Отвечает за декодирование команд, управление регистрами, взаимодействие с памятью и периферией.
* Периферийные устройства ввода-вывода
* Таймеры/счётчики – измеряют время, управляют событиями.
* USART, SPI, I²C – интерфейсы для связи с другими устройствами.
* АЦП (Аналого-цифровой преобразователь) – преобразует аналоговые сигналы в цифровые.
* Сторожевой таймер – перезапускает микроконтроллер при сбоях.
* Регистры
* Используются для хранения временных данных, адресов, флагов.
* В ATmega32 – 32 регистра общего назначения (R0–R31), а также специальные регистры состояния и управления.

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

Микроконтроллер ATmega32 является маломощным 8-разрядным КМОП микроконтроллером на основе усовершенствованной архитектуры RISC, выполняющий большинство инструкций за один такт. Он имеет гарвардскую архитектуру, которая представляет собой гибрид RISC и CISC архитектур. Особенности RISC в ATmega32 включают небольшой набор инструкций, простой и единообразный формат инструкций, а также ограниченное количество режимов адресации.

Преимущества RISC-особенностей в ATmega32:

* + Простая и быстрая декодировка инструкций, уменьшающая задержки при их выполнении.
  + Высокая скорость работы за счёт однотактного выполнения большинства команд.

Недостатки:

* Ограниченные режимы адресации могут усложнять работу с памятью и требовать дополнительных инструкций.
* Малый набор команд может привести к увеличению длины программы, что влияет на использование памяти.

1. *От чего зависит время выполнения команд CPSE и BRHS? Приведите примеры кода (до 3-10 команд каждый), приводящие к различной продолжительности выполнения указанных команд.*
   1. Команда CPSE **-** сравнивает два регистра и пропускает следующую команду, если они равны.

* Выполняется за 1 такт, если регистры не равны.
* Выполняется за 2 такта, если регистры равны (так как следующая команда пропускается, но занимает 1 такт).
* Выполняется за 3 такта, если следующая команда — двухсловная (например, RJMP или CALL)

Примеры:

LDI R18, 0x05

LDI R19, 0x04

CPSE R18, R19 ; R18 != R19, следующая команда выполнится

NOP ; Не пропускается, общее время 1 такт

LDI R16, 0x05

LDI R17, 0x05

CPSE R16, R17 ; Пропускает следующую команду, так как R16 == R17

NOP ; Будет пропущено, общее время 2 такта

LDI R20, 0x05

LDI R21, 0x05

CPSE R20, R21 ; Пропустит двухбайтовую команду, так как R20 == R21

RJMP loop ; Эта команда пропустится, время выполнения 3 такта

loop:

* 1. Команда BRHS - выполняет переход, если установлен флаг H (отслеживает перенос из 3-го в 4-й бит при арифметических операциях) в регистре статуса SREG.
* Выполняется за 1 такт, если условие ложно (переход не выполняется).
* Выполняется за 2 такта, если переход происходит (так как нужно изменить адрес исполнения).

Примеры:

LDI R18, 0x01

LDI R19, 0x02

ADD R18, R19 ; 0x01 + 0x02 = 0x03, флаг H не устанавливается

BRHS loop ; Условие не выполняется, команда занимает 1 такт

NOP ; Будет выполнено

loop:

LDI R16, 0x08

LDI R17, 0x08

ADD R16, R17 ; Операция 0x08 + 0x08 вызывает установку флага H

BRHS loop ; Переход выполняется, команда занимает 2 такта

NOP ; Будет пропущено

loop:

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

LSR R16 ; Логический сдвиг вправо, младший бит попадает в флаг C

BRCC skip ; Если C = 0, пропустить установку старшего бита

ORI R16, 0x80 ; Установить старший бит, если младший бит был 1

* 1. Сложение:

LDI R19, LOW(500) ; Загружаем младший байт константы

LDI R20, HIGH(500) ; Загружаем средний байт константы

LDI R21, BYTE3(500) ; Загружаем старший байт константы

ADD R16, R19 ; Складываем младшие байты

ADC R17, R20 ; Складываем средние байты с учетом переноса

ADC R18, R21 ; Складываем старшие байты с учетом переноса

1. *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, которая определяет количество битов, доступных для кодирования различных частей инструкции.

# Вывод

В ходе лабораторной работы были изучены основы программирования микроконтроллеров семейства Atmel ATmega в среде разработки Atmel Studio, а также проведено ознакомление с системой команд микроконтроллера ATmega32. Была проанализирована структура .hex и .lss файлов, рассмотрен алгоритм выполнения ассемблерных команд и проведен расчет времени выполнения фрагмента кода, соответствующего функции задержки, в зависимости от констант.

# Приложение

Листинг ассемблерного кода с комментариями:

.def TMP = R20 ; переименование регистра

.org $000 ; начальный адрес в памяти

JMP reset ; Указатель на начало программы

; Функция паузы

delay:

LDI R30, 200; y

LDI R29, 153; x

NOP

NOP

NOP

delay\_sub:

NOP

DEC R29

NOP

BRNE delay\_sub ; переход в начало цикла, если R29 неравно 0 (пока флаг Z = 0)

INC R30

BRNE delay\_sub ; переход в начало цикла, если R30 неравно 0

RET

; Начальная настройка

reset:

; настройка исходных значений

LDI TMP, 0x01

MOV R0, TMP ; R0 = 0x01

CLR TMP ; обнуляем все биты TMP

MOV R1, TMP ; R1 = 0x00

MOV R2, TMP ; R2 = 0x00

MOV R3, TMP ; R3 = 0x00

; настройка портов ввода-вывода

SER TMP ; 0xFF (установка в 1 всех бит TMP)

OUT DDRA, TMP ; Вывод (режим порта А)

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 ; Установка старшего байта указателя стека

LDI TMP, LOW(RAMEND) ; Младшие разряды адреса

OUT SPL, TMP ;Установка младшего байта указателя стека

; Основной цикл

loop:

; Циклический сдвиг 32-разрядного числа R0-R3

BST R0, 0 ; сохранение младшего бита во флаге Т

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 ;