Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«Севастопольский государственный университет»

ПРОГРАММИРОВАНИЕ НА АССЕМБЛЕРЕ ДЛЯ AVR-МИКРОКОНТРОЛЛЕРОВ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению лабораторных работ

для студентов, обучающихся по направлению
«Информационные системы и технологии»

Севастополь
2019

ЛАБОРАТОРНАЯ РАБОТА №1

Изучение интегрированной среды разработки программного обеспечения и исследование функционирования микроконтроллеров AVR

1 Цель работы

Ознакомиться с назначением и органами управления среды разработки, исследовать процессы содержимого регистров и портов микроконтроллера в процессе отладки программы. Приобрести практические навыки программирования и отладки программ на языке Ассемблера и Си.

2 Краткие теоретические сведения

2.1 Архитектура микроконтроллеров AVR семейства MEGA

Микроконтроллеры семейства Mega представляют собой однокристальную ЭВМ, состоящую из процессорного ядра, памяти, периферийных устройств и портов ввода/вывода. Эти микроконтроллеры предназначены для использования в мобильных телефонах, в контроллерах различного периферийного оборудования (такого как принтеры, сканеры, современные дисковые накопители, приводы CD-ROM/DVD-ROM и т. п.), в сложной офисной технике и т. д.

Процессорное ядро AVR-микроконтроллеров (рис. 2.1) выполнено по Гарвардскийой архитектуре на основе сокращенного набора команд (RISC команд). Эта архитектура характеризуется раздельной памятью программ и данных, каждая из которых имеет собственные шины доступа. Такая организация позволяет одновременно работать как с памятью программ, так и с памятью данных.

Арифметико-логическое устройство (АЛУ), выполняющее все вычисления, подключено непосредственно к 32 рабочим регистрам, объединенным в регистровый файл. Благодаря этому АЛУ может выполнять одну операцию (чтение содержимого регистров, выполнение операции и запись результата обратно в регистровый файл) за такт. Кроме того, практически каждая из команд (за исключением команд, у которых одним из операндов является 16-битный адрес) занимает одну ячейку памяти программ.

Разделение информационных шин позволяет использовать для каждого типа памяти шины различной разрядности, причем способы адресации и доступа к каждому типу памяти также различаются. В сочетании с двухуровневым конвейером команд (одна команда выполняется, другая параллельно выбирается и декодируется) такая архитектура позволяет достичь производительности в 1 MIPS на 1 МГц тактовой частоты.

AVR-микроконтроллеры изготавливаются по малопотребляющей КМОП-технологии и имеют полностью статическую архитектуру, минимальная тактовая частота равна нулю.

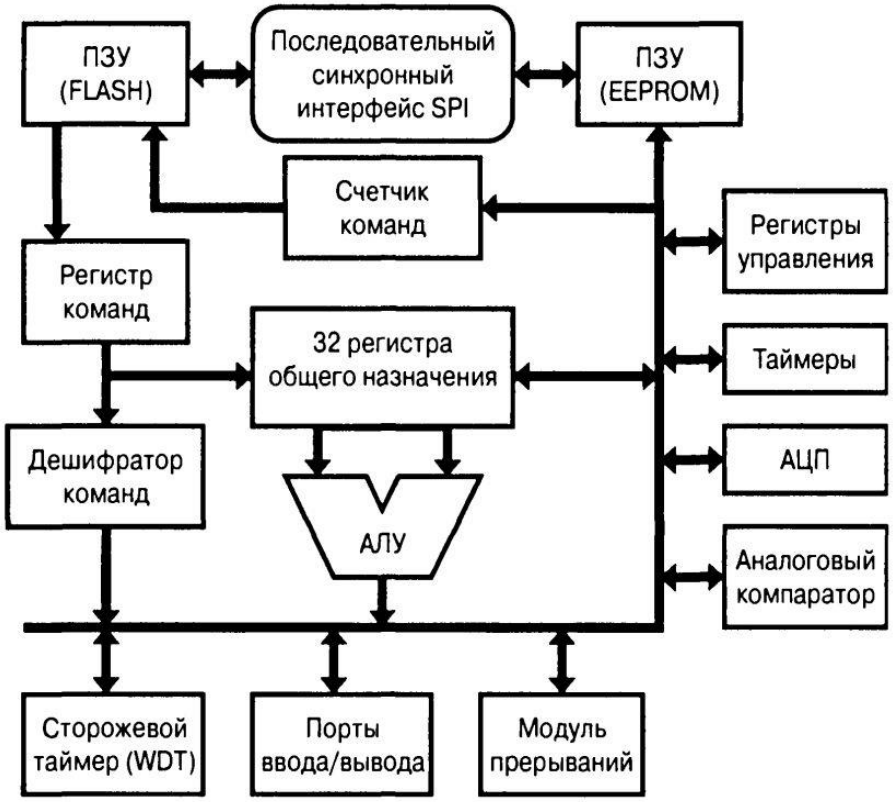

Рисунок 2.1− Архитектура процессорного ядра AVR-микроконтроллера

Основные параметры и характеристики микроконтроллеров AVR семейства Mega:

• FLASH-память программ объемом от 8 до 256 Кбайт (число циклов стирания/записи не менее 10 000);

• оперативная память (статическое ОЗУ) объемом от 512 байт до 8 Кбайт;

• память данных на основе ЭСППЗУ (EEPROM) объемом от 256 байт до 4 Кбайт (число циклов стирания/записи не менее 100 000);

• возможность защиты от чтения и модификации памяти программ и данных;

• возможность программирования непосредственно в системе через последовательные интерфейсы SPI и JTAG;

• возможность внутрисхемной отладки в соответствии со стандартом IEEE 1149.1 (JTAG), а также наличие собственного однопроводного интерфейса внутрисхемной отладки (debugWire1);

• разнообразные способы синхронизации: встроенный RC-генератор с внутренней или внешней времязадающей RC-цепочкой, встроенный генератор с внешним кварцевым или пьезокерамическим резонатором, внешний сигнал синхронизации;

• наличие нескольких режимов пониженного энергопотребления;

• наличие сторожевого таймера;

• наличие детектора пониженного напряжения питания;

• возможность программного снижения частоты тактового генератора.

• векторная система прерываний, поддержка очереди прерываний;

• большое число источников прерываний (до 45 внутренних и до 32 внешних);

• наличие аппаратного умножителя.

Структурная схема микроконтроллера ATmega8535 приведена в приложении А. Микроконтроллеры AVR семейства Mega имеют от 23 до 86 линий ввода/вывода, которые объединяются в 8-разрядные порты ввода/вывода, причем отдельные линии могут быть запрограммированы как входные или как выходные независимо друг от друга. Каждая линия имеет входной буфер с триггером Шмита, а индивидуально отключаемый внутренний (подтягивающий) резистор сопротивлением 20...50 кОм.

Кроме того, имеется большой набор периферийных устройств:

• один или два 8-битных таймера/счетчика, во всех моделях с двумя 8-битными таймерами/счетчиками один из них может работать в качестве часов реального времени (в асинхронном режиме);

• от одного до четырех 16-битных таймеров/счетчиков;

• одно- и двухканальные генераторы 8-битного ШИМ-сигнала (один из режимов работы 8-битных таймеров/счетчиков);

• двух- и трехканальные генераторы ШИМ-сигнала регулируемой разрядности (один из режимов работы 16-битных таймеров/счетчиков), разрешение формируемого сигнала может составлять от 1 до 16 бит;

• аналоговый компаратор;

• многоканальный 10-битный АЦП последовательного приближения, имеющий как несимметричные, так и дифференциальные входы;

• последовательный синхронный интерфейс SPI;

• последовательный двухпроводный интерфейс TWI (аналог интерфейса I^2C);

• от одного до четырех полнодуплексных универсальных синхронных/асинхронных приемо-передатчиков (USART), которые в ряде моделей могут использоваться в качестве ведущего устройства шины SPI;

• универсальный последовательный интерфейс USI, который может использоваться в качестве интерфейса SPI или I^2C, а также полудуплексного UART или 4/12-битного счетчика.

В качестве примера на рис. 2 приведена структурная схема микроконтроллера ATmega8535, который имеет четыре 8-битных порта ввода/вывода (порты A...D), два 8-битных (T0, T2) и один 16-битный (T1) таймер/счетчик, 4 канала ШИМ, по одному интерфейсному модулю USART, SPI и TWI.

Обобщенная карта памяти микроконтроллеров AVR семейства Mega приведена на рис.2.2. Поскольку микроконтроллеры AVR имеют 16-битную систему команд, объем памяти программ на рисунке указан не в байтах, а в 16-битных словах. Символ «$» перед числом означает, что это число записано в шестнадцатеричной системе счисления. Для микроконтроллера ATmega8535 объем памяти программ составляет 8 Кбайт (4К*16), ее верхняя граница F_END = $FFF, объем

ОЗУ данных – 512 байт, верхняя граница S_END = $25F, объем ПЗУ данных – также 512 байт, верхняя граница E_END = $1FF.

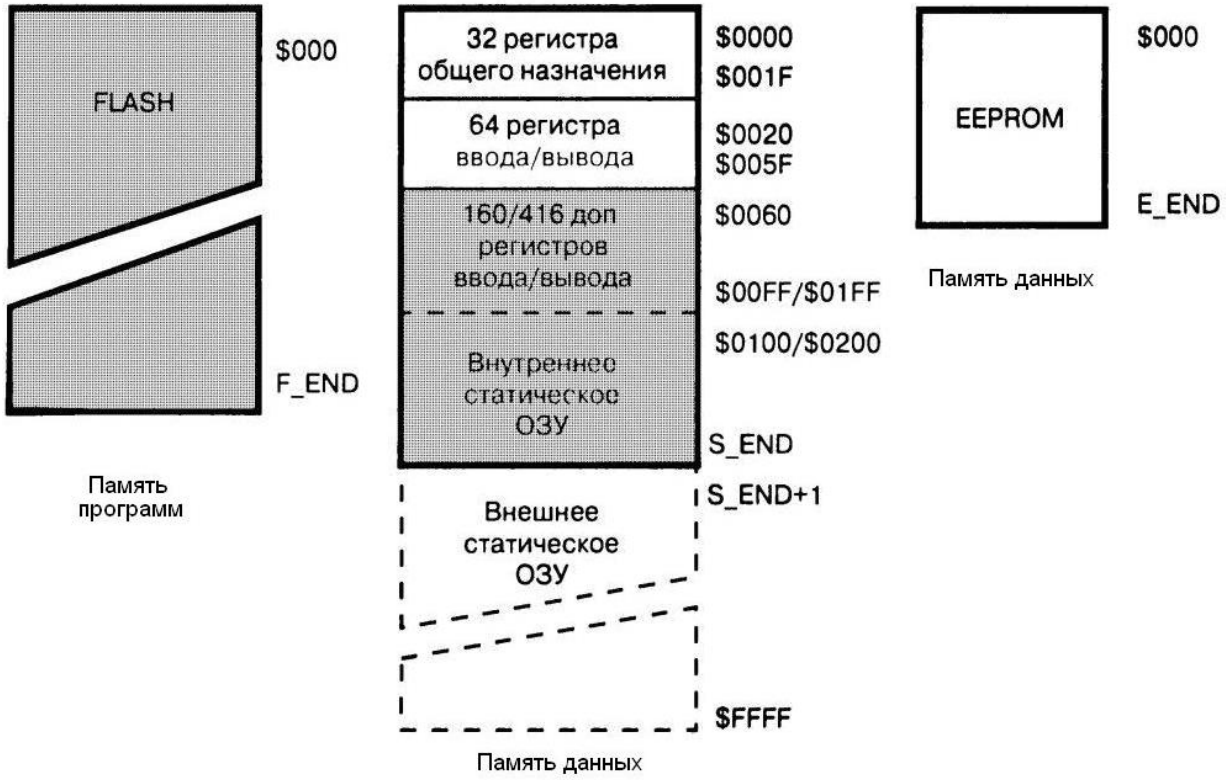

Рисунок 2.2 − Карта памяти AVR-микроконтроллеров семейства Mega

Память программ (объемом от 8 до 256 Кбайт) предназначена для хранения команд, управляющих работой микроконтроллера, а также часто используется для хранения таблиц констант, не меняющихся во время работы программы. Для пересылки байта из памяти программ в память данных существует специальная команда — LPM. При использовании команды LPM адрес, по которому производится чтение, определяется содержимым индексного регистра Z. При этом старшие 15 битов содержимого регистра будут определять адрес слова (0 ... 32 К), младший бит будет определять, какой из байтов будет прочитан: 0 — младший байт, 1 — старший байт.

В подавляющем большинстве моделей микроконтроллеров семейства Mega память программ логически разделена на две неравные части: область прикладной программы и область загрузчика. В последней может располагаться специальная программа (загрузчик), позволяющая микроконтроллеру самостоятельно управлять загрузкой и выгрузкой прикладных программ. Если же возможность самопрограммирования микроконтроллера не используется, прикладная программа может располагаться и в области загрузчика.

По адресу $0000 памяти программ находится вектор сброса. После инициализации (сброса) микроконтроллера выполнение программы начинается с этого адреса (по этому адресу должна размещаться команда перехода к инициализационной части программы). Начиная с адреса $001 (модели с памятью программ 8

Кбайт и меньше) или $0002 (остальные модели) памяти программ располагается таблица векторов прерываний. Размер этой области зависит от модели микроконтроллера.

При возникновении прерывания после сохранения в стеке текущего значения счетчика команд происходит выполнение команды, расположенной по адресу соответствующего вектора. Поэтому по данным адресам располагаются команды перехода к подпрограммам обработки прерываний. В моделях с памятью программ небольшого объема (8 Кбайт и менее) в таблице векторов прерываний используются команды относительного перехода (RJMP), а в остальных моделях — команды абсолютного перехода (JMP). Если в программе прерывания не используются, то основная программа может начинаться непосредственно с адреса $0001.

Память данных микроконтроллеров семейства Mega разделена на три части: регистровая память, оперативная память (статическое ОЗУ) и энергонезависимое ЭСППЗУ (EEPROM).

Регистровая память включает 32 регистра общего назначения (РОН), и 64 служебных регистра ввода/вывода (РВВ). В сложных моделях с развитой периферией имеется также область дополнительных (extended) регистров ввода/вывода (ДРВВ). Под РВВ в памяти микроконтроллера отводится 64 байта, а под ДРВВ — 160 или 416 байт (в зависимости от модели).

В отличие от процессоров с аккумулятором, в процессорном ядре AVR все 32 РОН непосредственно доступны АЛУ. Благодаря этому любой РОН может использоваться практически во всех командах и как операнд-источник, и как операнд-приемник. Последние 6 регистров общего назначения (R26...R31) могут также объединяться в три 16-битных регистра X, Y и Z, используемых в качестве указателей при косвенной адресации памяти данных.

В областях регистров ввода/вывода располагаются различные служебные регистры (регистр управления микроконтроллера, регистр состояния и т. п.), а также регистры управления периферийными устройствами, входящими в состав микроконтроллера. Общее их количество зависит от конкретной модели микроконтроллера.

К регистрам ввода/вывода, расположенным в основном пространстве ввода/вывода, можно напрямую обратиться с помощью команд IN и OUT, выполняющих пересылку данных между одним из 32 РОН и пространством ввода/вывода. В системе команд имеется также четыре команды побитового доступа, использующие в качестве операндов регистры ввода/вывода: команды установки/сброса отдельного бита (SBI и CBI) и команды проверки состояния отдельного бита (SBIS и SBIC). Но эти команды могут обращаться только к 1-й половине основного пространства ввода/вывода (адреса $00...$1F).

Среди РВВ есть один регистр, используемый наиболее часто в процессе выполнения программ. Это регистр состояния SREG. Он располагается по адресу $3F ($5F) и содержит набор флагов, показывающих текущее состояние микроконтроллера. Большинство флагов автоматически устанавливаются в 1 или сбрасываются в 0 при наступлении определенных событий (в соответствии с результатом выполнения команд). Все биты этого регистра доступны как для чтения, так и

для записи; после сброса микроконтроллера все биты регистра сбрасываются в 0. Формат этого регистра следующий:

Регистр состояния – SREG

Бит	7	6	5	4	3	2	1	0
SREG $3F($5F)	I	T	H	S	V	N	Z	C
Чтение/Запись	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Исходное значение	0	0	0	0	0	0	0	0

• Бит 7 – **I: Общее разрешение прерываний**. Для разрешения прерываний этот флаг должен быть установлен в «1». Разрешение/запрещение отдельных прерываний производится установкой или сбросом соответствующих разрядов регистров масок прерываний (регистра управления прерываниями). Если флаг сброшен, то прерывания запрещены независимо от состояния разрядов этих регистров. Флаг сбрасывается аппаратно после входа в прерывание и восстанавливается командой RETI для разрешения обработки следующих прерываний.

• Бит 6 – **T: Хранение копируемого бита.** Этот разряд регистра используется в качестве источника или приемника команд копирования битов BLD (Bit LoaD) и BST (Bit Store). Заданный разряд любого РОН может быть скопирован в этот разряд командой BST или установлен в соответствии с содержимым данного разряда командой BLD.

• Бит 5 – **H: Флаг половинного переноса.** Устанавливается в «1», если произошел перенос из младшей половины байта в старшую (из 3-го разряда в 4-й) или заем из старшей половины байта при выполнении некоторых арифметических операций.

• Бит 4 – **S: Флаг знака.** Этот флаг равен результату операции «Исключающее ИЛИ» между флагами N (отрицательный результат) и V (переполнение числа в дополнительном коде). Соответственно этот флаг устанавливается в «1», если результат выполнения арифметической операции меньше нуля.

• Бит 3 – **V: Флаг переполнения дополнительного кода.** Устанавливается в «1», при переполнении разрядной сетки знакового результата. Используется при работе со знаковыми числами (представленными в дополнительном коде).

• Бит 2 – **N: Флаг отрицательного значения.** Устанавливается в «1», если старший (7-й) разряд результата операции равен «1». В противном случае флаг равен «0».

• Бит 1 – **Z: Флаг нулевого значения.** Этот флаг устанавливается в «1», если результат выполнения операции равен нулю.

• Бит 0 – **C: Флаг переноса.** Устанавливается в «1», если в результате выполнения операции произошел выход за границы байта.

Адреса всех регистров ввода-вывода микроконтроллера ATmega8535 можно найти в справочном листке на микроконтроллер.

Для хранения переменных помимо регистров общего назначения также используется статическое ОЗУ объемом от 512 байт до 8 Кбайт. Ряд микроконтрол-

леров семейства, кроме того, имеют возможность подключения внешнего статического ОЗУ объемом до 64 Кбайт.

В адресном пространстве ОЗУ также расположены все регистры микроконтроллеров, под них отведены младшие 96 (256) адресов. Каждый регистр имеет свой собственный адрес в пространстве памяти данных. Поэтому к регистрам можно обращаться двумя способами — как к регистрам и как к памяти, несмотря на то что физически эти регистры не являются ячейками ОЗУ. Так, регистрам общего назначения R0 – R31 соответствуют адреса ОЗУ $000 – $01F, регистрам ввода/вывода $00 – $3F соответствуют адреса ОЗУ $020 – $05F (номер регистра плюс $20).

2.2 Общие сведения о платформах разработки

Для разработки программного обеспечения для микроконтроллеров типа AVR используется несколько сред (платформ) разработки. Основными из них являются WinAVR, AVR Studio и некоторые другие.

WinAVR представляет собой среду разработки для операционных систем семейства Windows, созданной с целью написания и отладки программ для микроконтроллеров серии AVR и AVR32 от компании Atmel.

WinAVR состоит из GNU GCC компилятора, поддерживающего языки C, C+ и Objective-C, обеспечивая полный цикл разработки для AVR/AVR32 и дополнительных инструментов, каждый из которых выполняет определенную задачу, помогая написанию программного обеспечения для микроконтроллеров. Наиболее важные из них:

- Programmers Notepad – текстовый редактор для создания программ;
- AVR-LibC – библиотека микроконтроллеров AVR;
- AVRDUDE – утилита, предназначенная для программирования микросхем и использующая SPI-интерфейс;
- GNU Debugger (GDB) – отладчик с командной строкой;
- Insight – отладчик с графическим интерфейсом;
- Simulavr – симулятор микроконтроллеров AVR с поддержкой отладчика GDB;
- SRecord – набор утилит для работы с загрузочными файлам EPROM разных форматов.

Существует несколько версий интегрированной среды разработки IDE (Integrated Development Environment) микропроцессорных устройств AVR Studio, функциональные возможности которых постоянно расширяются. Одна из самых простых и удобных в работе является версия AVR Studio 4, предназначенная для написания и отладки прикладных программ для AVR микропроцессоров в среде Windows. AVR Studio 4 содержит ассемблер и симулятор.

В интегрированную среду разработки AVR Studio 5 внесены ряд добавок, расширяющих функциональные возможности среды и облегчающих процесс разработки программного обеспечения. В частности, она содержит встроенные примеры программ со всеми настройками для AVR32 UC3 и AVR Xmega, встроенный

набор библиотек AVR Software Framework, написанных единообразно для разных семейств и архитектур, который обеспечивает возможность быстрого перевода проектов как между семействами, так и между архитектурами и др.

Интегрированная платформа разработки Atmel Studio 6 предназначена для проектирования и отладки приложений как на базе микроконтроллеров AVR Atmel, так и новых типов контроллеров ARM Cortex-M. Atmel Studio 6 представляет собой единую, простую в использовании среду разработки, компоновки и отладки приложений, написанных на языке C/C++ либо ассемблере. Эта платформа, как и ее предшественники являются бесплатными.

В нее интегрирована обширная библиотека бесплатного исходного кода Atmel Software Framework (ASF), содержащая свыше 1600 примеров ARM- и AVR-проектов. Эта библиотека дополняет преимущества платформы Atmel Studio 6, предоставляя доступ к готовому коду в той же среде, что сводит к минимуму необходимость низкоуровневого программирования в проектах.

В версии платформы, Atmel Studio 6.2, были добавлены современные функции отладки, в частности отслеживание данных и прерываний (Data and Interrupt Trace). С появлением решений Atmel Gallery и Atmel Spaces проектирование встраиваемых микроконтроллеров на платформе Atmel Studio 6 стало еще проще, а время разработки и затраты снизились. Atmel Gallery представляет собой сетевой магазин приложений, где можно найти инструменты разработки и встраиваемое программное обеспечение. Atmel Spaces — это облачное пространство для совместной разработки, позволяющее хранить проекты по созданию программного и аппаратного обеспечения для микроконтроллеров Atmel.

Основное достоинство новой версии бесплатной популярной среды разработки Atmel Studio 6 от Atmel - она теперь основывается на движке Visual Studio 2010, и имеет абсолютно такой же интерфейс, как и популярная среда разработки программ для PC - Microsoft Visual Studio 2010. Это упрощает освоение среды программирования AVR для тех, кто уже знаком с традиционным интерфейсом Visual Studio.

Существенным недостатком 5-й и последующих версий Atmel Studio является их громоздкость (от 600 до 900 Мбайт), что затрудняет использовать их на лабораторных компьютерах с ограниченным объемом оперативной памяти.

3 Описание лабораторной установки

Лабораторная установка для создания и отладки приложений и исследования процессов функционирования микроконтроллеров типа AVR Atmel представляет собой персональный компьютер, на котором установлена интегрированная среда разработки AVR Studio 4. При запуске AVR Studio 4 открывается окно, показанное на рис.3.1.

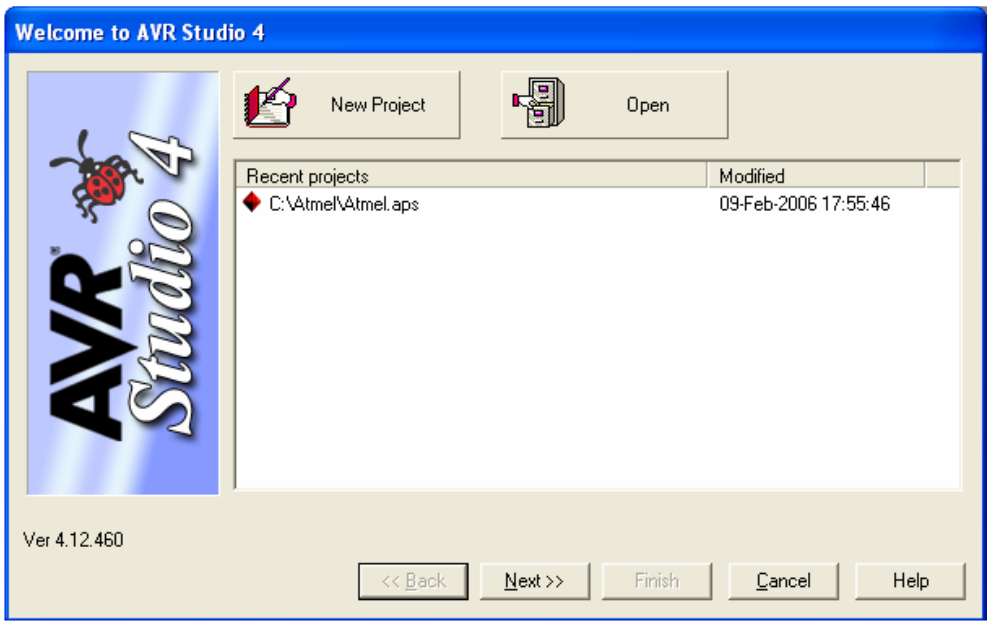

Рисунок 3.1 – Вид окна AVR Studio после старта системы

Система предлагает создать новый проект (New Project) или открыть уже существующий (Open) проект приложения. Если приложение уже существует, то открывается файл этой программы, а если она создается вновь, то выбирается New Project. В результате открывается окно (рис.3.2), в котором нужно выбрать тип проекта.

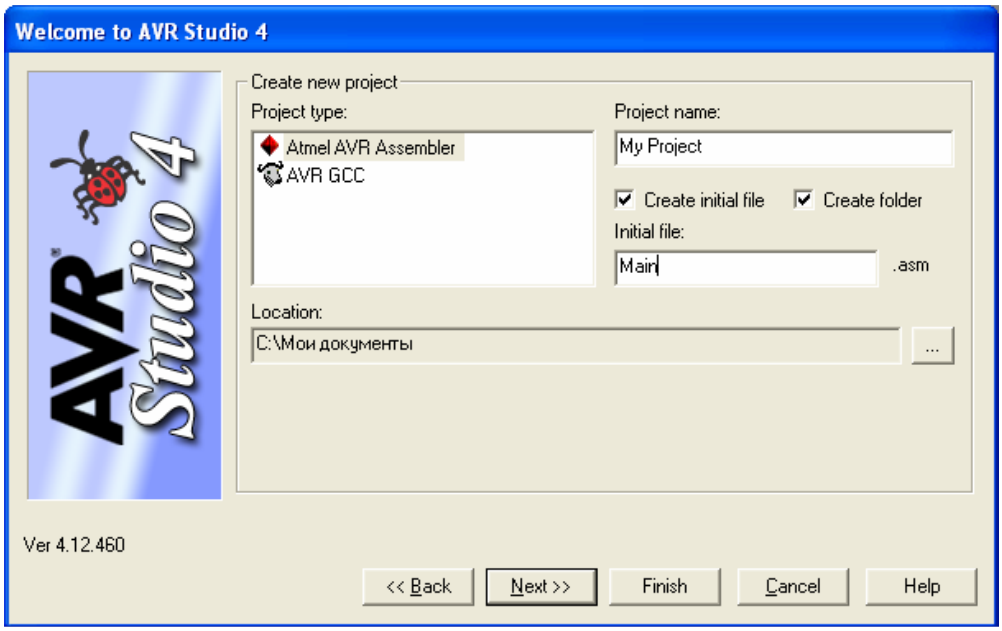

Рисунок 3.2 – Вид окна при создании проекта

В данной работе программирование будет выполняться на языке Ассемблер. Для этого в окне тип проекта (Project Type) выбирается строка Atmel AVR Assembler, задается произвольное имя проекта (Project Name) и заглавного файла (Initial File). Затем нажимается клавиша Next. Открывается окно (рис.3.3), в кото-

ром выбирается платформа отладки (Debug Platform) и отлаживаемое устройство Device (тип микроконтроллера).

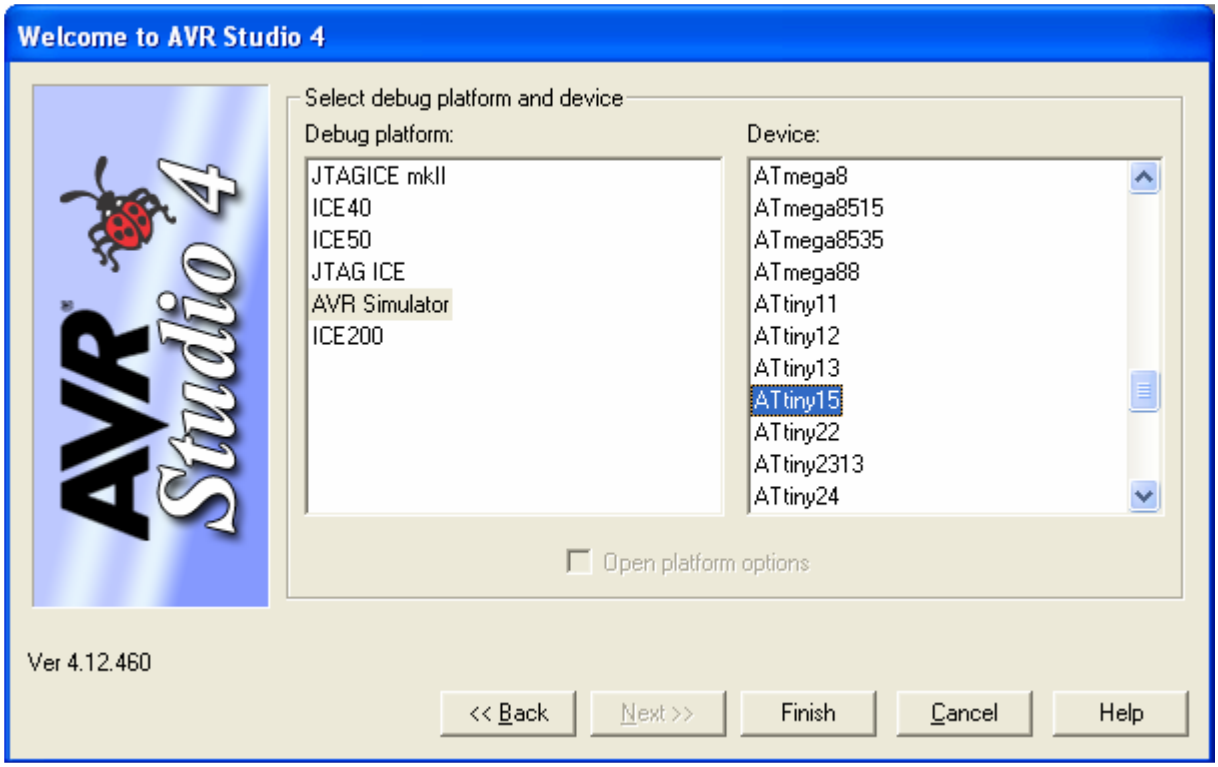

Рисунок 3.3 – Окно выбора платформы отладки и типа микроконтроллера

В окне платформы отладки следует выбрать симулятор, а в окне устройство – подходящий микроконтроллер (в данной работе ATtiny15). Для завершения процедуры создания проекта нужно нажать клавишу Finish. В результате открывается главное окно интегрированной среды разработки (рисунок 3.4). Кроме стандартных кнопок управления главного меню, имеются специальные кнопки, применяемые преимущественно при компиляции и отладки программы. Назначение этих кнопок выводится при наведении на них курсора мышки.

Окно разделено на 4 части. В верхней части расположены строка меню и «плавающие» панели с кнопками. Чуть ниже слева располагается вкладки **Диспетчер проекта** (Project), **Просмотр ввода/вывода** (I/O View), **Информация** (Info), справа – **Текст программы.**

Снизу расположены следующие вкладки: **Конструкция** (Build), **Сообщения** (Message), **Поиск в файлах** (Find in Files), **Контрольные точки** (Breakpoints and Tracepoints).

Текст разрабатываемой программы размещается в окне **Текст программы.** При написании программы инструкции выделяются синим цветом, комментарии – зеленым, остальное – черным. При написании ПО рекомендуется периодически сохраняться.

После завершения написания текста программы осуществляется его **компиляция,** т.е. перевод программы, написанной на языке программирования, в исполняемый модуль, содержащий машинные команды конкретного процессора. При-

менительно к микроконтроллерам процесс компиляции называется ассемблированием.

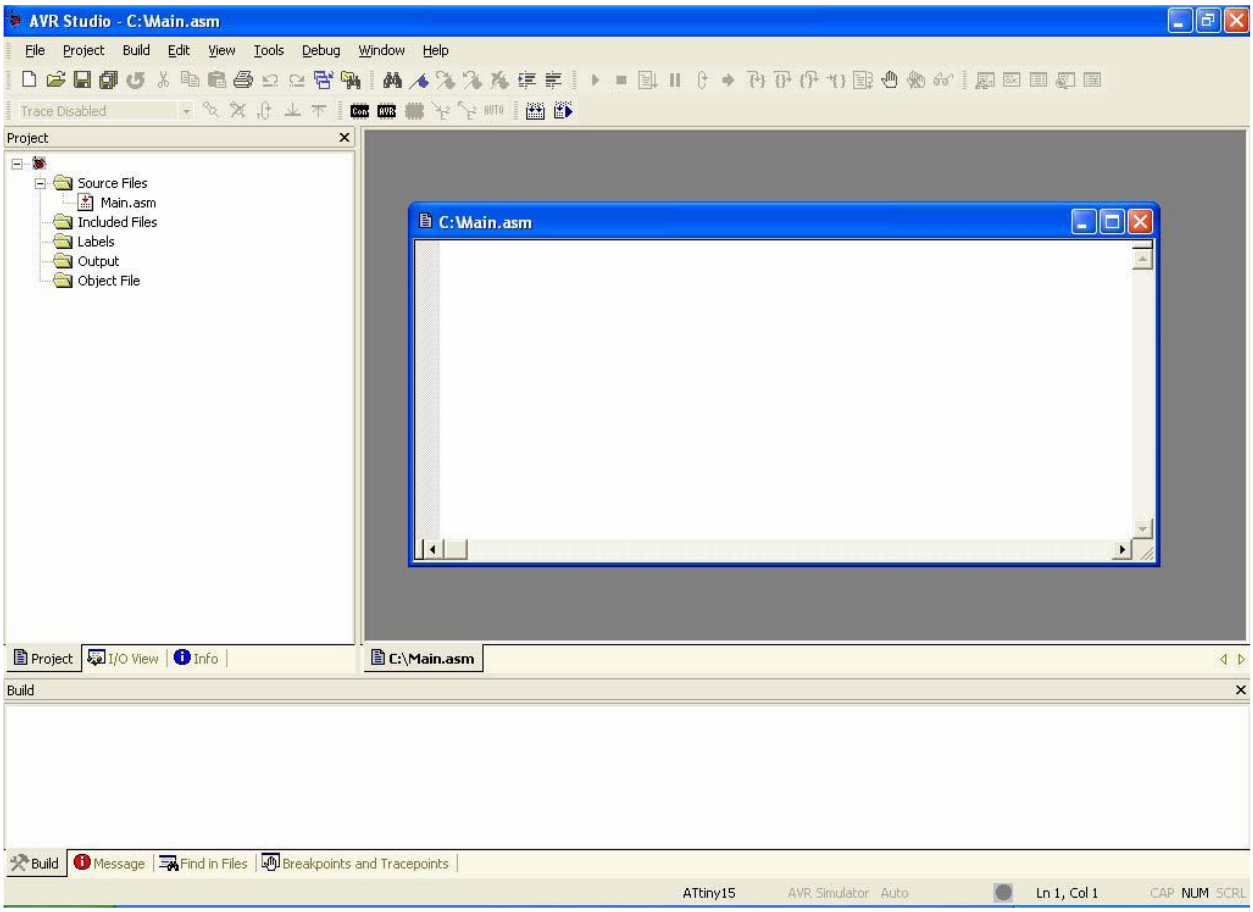

Рисунок 3.4 – Вид главного окна среды разработки

Ассемблирование — трансляция с языка ассемблера в команды машинного языка. Данные кнопки на верхней панели запускают процесс ассемблирования. Кнопка слева ассемблирует проект, справа – ассемблирует и запускает на выполнение.

Если при написании текста программы были допущены синтаксические ошибки, компиляция прерывается и во вкладке **Конструкция** выдается сообщения о допущенных ошибках (рисунок 3.5).

При безошибочной компиляции на вкладке **Конструкция** выводится отчет о прохождении процесса ассемблирования и таблица использованных ресурсов.

После удачного ассемблирования можно переходить к фазе симуляции. **Симуляция** – моделирование процесса выполнения программы микроконтроллером на персональном компьютере. Она выполняется в режиме отладки (Debugging). Для управления режимом отладки предназначены функциональные кнопки, назначение которых приведено в таблице 3.1.

Инструкция, которая будет выполняться следующей, обозначается желтой стрелочкой. Для удобства отладки применяется контрольные точки. **Контрольная точка** – инструкция в программе дойдя до которой выполнение программы приостановится. Установленная контрольная точка отмечена красным кружком.

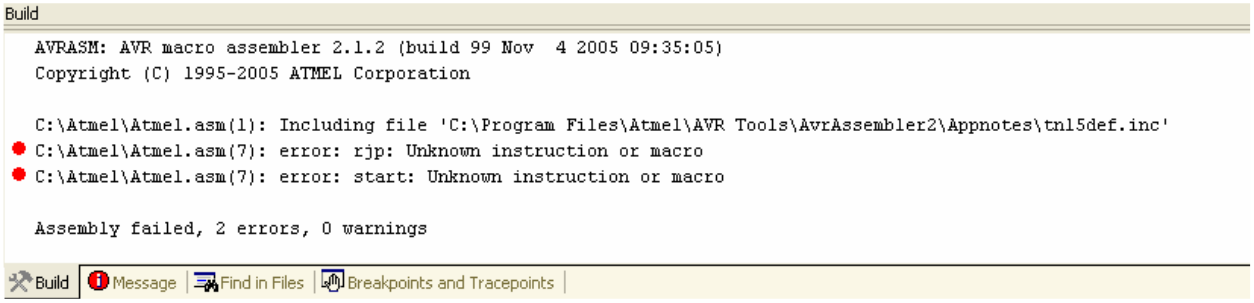

Рисунок 3.5 – Вид сообщения об ошибках

Таблица 3.1 Кнопки управления процессом моделирования

	Запустить отладку (симуляцию).
	Остановить отладку.
	Запустить программу на выполнение.
	Пауза в выполнении программы.
	Показать выполняемую инструкцию.
	Перезапустить программу.
	Шаг вперед с заходом в подпрограммы.
	Шаг вперед без захода в подпрограммы.
	Перейти к последней инструкции программы (подпрограммы).
	Выполнить программу до места, указанного курсором.
	Автоматическое пошаговое выполнение программы.
	Установить/снять контрольную точку.
	Удалить все контрольные точки.

В пакете AVR Studio предусмотрено множество средств, облегчающих программирование, в частности:

Смотровое окно (рис. 3.6) – предназначено для просмотра и редактирования значений всех предопределенных символов.

Окно памяти (рис.3.7) – показывает распределение и содержание памяти в микроконтроллере (памяти программ, регистров общего пользования, регистров ввода/вывода, EEPROM и др.).

Окно регистров (рис.3.7) – служит для просмотра и редактирования значений регистров.

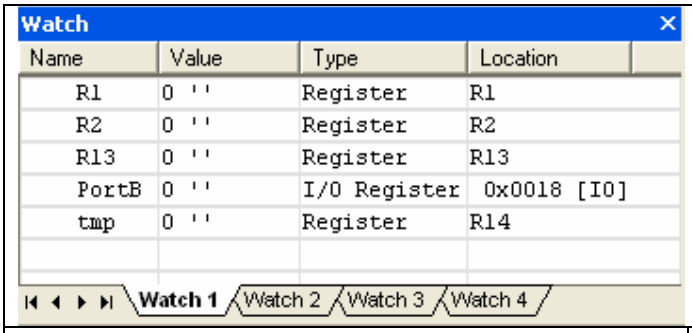

Рисунок 3.6 – Вид смотрового окна AVR Studio 4

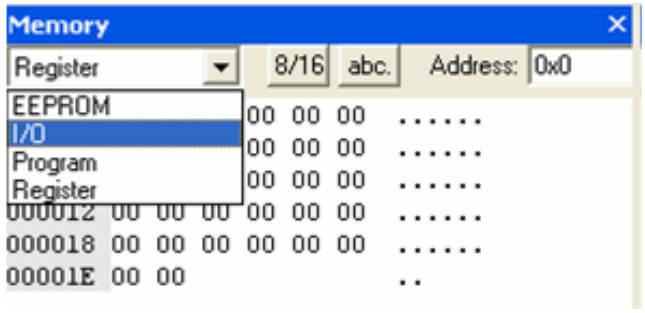

Рисунок 3.7 - Окно памяти

Информация о регистрах ввода/вывода, процессоре и регистрах общего пользования расположена и распределена по группам на вкладке I/O View (рис.3.9).

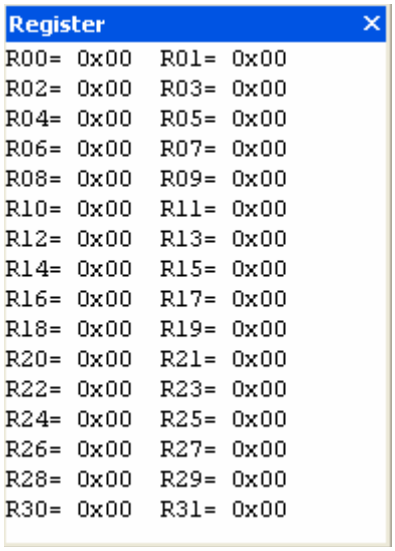

Рисунок 3.8 - Окно регистров

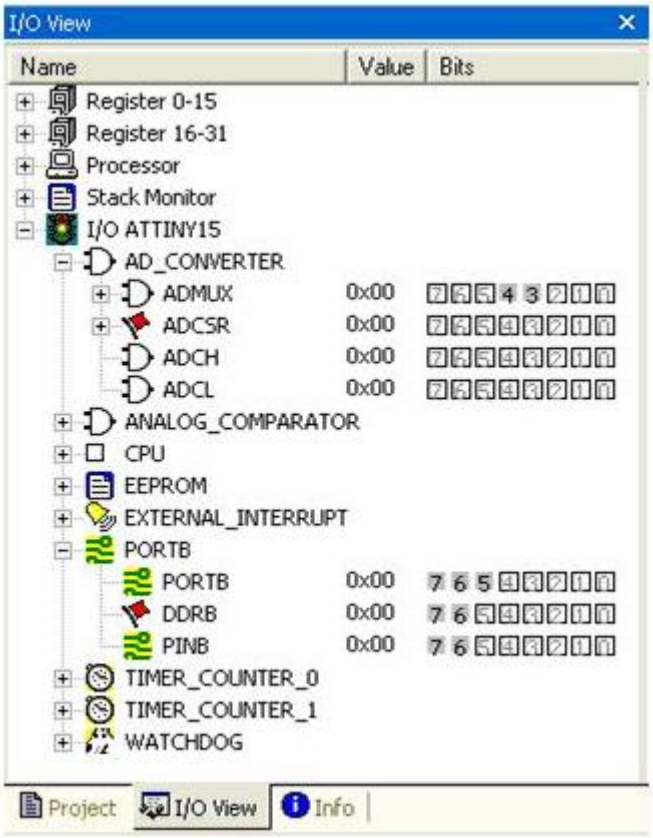

Рисунок 3.9 – Вид окна ввода/вывода

После того как проект прошел отладку данную программу можно «зашивать» в микроконтроллер.

4. Программа и методика исследований

В процессе выполнения лабораторной работы при программировании в среде AVR Studio необходимо выполнить следующие действия.

4.1 Изучить структуру и назначение функциональных блоков микроконтроллера.
4.2 Ознакомиться с особенностями системы команд микроконтроллеров типа AVR.
4.3 Ознакомиться со средой программирования и отладки программ типа AVR Studio 4 (или 5).
4.4 Подготовить в редакторе AVR Studio ознакомительную программу на ассемблере, приведенную в приложении А п.1.
4.5 Записать в комментариях значение каждой команды.
4.6 Выполнить ассемблирование программы.
4.7 Запустить отладчик программы и исправить, при их наличии, синтаксические ошибки.
4.8 Исследовать изменение содержимых рабочих регистров, указателя стека, флагов и ячеек памяти при пошаговом выполнении программы.
4.9 Повторить пп.4.4-4.8 для задания 2, приложения А согласно варианту.
4.10 Оформить отчет по лабораторной работе.

ПРИМЕЧАНИЕ: Первые три пункта программы выполняются в процессе домашней подготовки.

5. Содержание отчета

5.1 Цель и программа работы.
5.2 Структурная схема микроконтроллера.
5.3 Текст программы с комментариями на ассемблере.
5.4 Выводы по результатам исследований.

6. Контрольные вопросы

6.1 В чем состоит отличие архитектуры микроконтроллеров AVR от архитектуры универсальных микропроцессоров?
6.2 Поясните назначение функциональных узлов микроконтроллера ATmega16 и расскажите о работе устройства.
6.3 Расскажите об особенностях настройки портов ввода/вывода.
6.4 Какие функции выполняют регистры X, Y, Z?
6.5 С какой целью в схему микроконтроллера включен сторожевой таймер?
6.6 Для чего в микроконтроллер встроен аналоговый компаратор?
6.7 Расскажите о структуре flash –памяти программ и о адресном пространстве статической памяти микроконтроллера.
6.8 Раскройте особенности программирования RISC процессоров.

6.9 Какие ассемблерные директивы наиболее широко применяются в ассемблере МК AVR и каково их назначение?
6.10 Какие битовые операции и с какой целью они используются в программах для микроконтроллеров?
6.11 Как осуществляется процедура исследования функционирования микроконтроллера?

7 Список рекомендованной литературы

7.1 Баранов В.Н. Применение микроконтроллеров AVR: схемы, алгоритмы, программы. – М.: Издательский дом «Додека-XXI», 2004. – 288 с.
7.2 Мортон Дж. Микроконтроллеры AVR. Вводный курс. / Пер. с англ.- М.: Издательский дом «Додека-XXI», 2006. – 272 с.
7.3 Ревич Ю.В. Практическое программирование микроконтроллеров Atmel AVR на языке ассемблера. – СПб., БХВ-Петербург, 2008. – 384 с.
7.4 Хартов В. Я. Микроконтроллеры AVR. Практикум для начинающих: учеб. пособие / В. Я. Хартов. – 2-е изд., испр. и доп. – М. : Изд-во МГТУ им. Н. Э. Баумана, 2012. – 280 с.
7.5 Программирование с нуля в AVRStudio 5. http://chipmk.ru/index.php/component/k2/item/118-programmirovanie-c-nulya-v-avrstudio-5-ch-1

Приложение А. Варианты задания

1) Ознакомительная программа

```
1.  * Lab_1mega16.asm
2.  *
3.  */
4. .include "m16def.inc"
5. .def temp=r16
6. .def count=r17
7. rjmp init
8. ;
9. init:
10. ldi temp,$80
11. out SPL,temp
12. ldi count,5
13. m1:
14. inc r18
15. inc r19
16. add r18,r19
17. mov r20,r18
18. push r20
19. pop r21
20. subi r20,2
21. dec count
22. brne m1
23. ret
```

2) Составить программу на ассемблере по следующему словесному описанию и исследовать функционирование (содержимое регистров) при пошаговом выполнении программы:

– Установить указатель стека на ячейку с адресом N+80.
– Организовать цикл. Установить значение счетчика циклов равное N+5.
– Инкрементировать регистр 18, затем 19, сложить их содержимое и перенести сумму в регистр 20.
– Сохранить значение регистра 20 в стеке, затем перенести это число из вершины стека в регистр 21.
– Вычесть из регистра 20 число 2, уменьшить счетчик циклов на единицу.

Продолжать выполнять действия в цикле, пока счетчик не станет равным нулю.

N - последняя цифра в номере зачетной книжки.

Приложение Б. Структурная схема микроконтроллера ATmega8535