|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 2 |

**Название:**

Обработка внешних прерываний микроконтроллерах AVR.

**Дисциплина:** Микропроцессорные системы.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-62Б |  |  | С.В. Астахов, Д.И. Вариханов |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  |  |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2022

**Вариант 1.**

**Цели работы:**

- изучение системы прерываний микроконтроллеров AVR;

- освоение системы команд микроконтроллеров AVR;

- ознакомление с работой стека при вызове подпрограмм и обработчиков прерываний;

- программирование внешних прерываний.

**Ход работы.**

**Задание 1.**

Убедившись в правильности работы программы восстановить параметры подпрограмм задержки и заново откомпилировать программу.

Загрузить программу в память микроконтроллера и проверить её работу на плате.

Схема контроллера и схема алгоритма приведены на рисунке 1.

|  |  |
| --- | --- |
| 88 |  |

а) б)

Рисунок 1 - контроллер (а) и схема алгоритма работы (б)

Исходный код программы:

*.include "m8515def.inc" ;файл определений для ATmega8515*

*.def temp = r16 ;временный регистр*

*.equ led = 0 ;0-й бит порта PB*

*.equ sw0 = 0 ;0-й бит порта PA*

*.equ sw1 = 1 ;1-й бит порта PA*

*.org $000*

*rjmp INIT ;обработка сброса*

*;\*\*\*Инициализация МК\*\*\**

*INIT: ldi temp,$5F ;установка*

*out SPL,temp ; указателя стека*

*ldi temp,$02 ; на последнюю*

*out SPH,temp ; ячейку ОЗУ*

*ser temp ;инициализация выводов*

*out DDRB,temp ; порта PB на вывод*

*out PORTB,temp ;погасить LED*

*clr temp ;инициализация*

*out DDRA,temp ; порта PA на ввод*

*ldi temp,0b00000011 ;включение ‘подтягивающих’*

*out PORTA,temp ; резисторов порта PA*

*test\_sw0: sbic PINA,sw0 ;проверка состояния*

*rjmp test\_sw1 ; кнопки sw0*

*cbi PORTB, led*

*rcall delay1*

*sbi PORTB,led*

*wait\_0: sbis PINA,sw0*

*rjmp wait\_0*

*test\_sw1: sbic PINA, sw1 ;проверка состояния*

*rjmp test\_sw0 ; кнопки sw1*

*cbi PORTB,led*

*rcall delay2*

*sbi PORTB,led*

*wait\_1: sbis PINA,sw1*

*rjmp wait\_1*

*rjmp test\_sw0*

*delay1: ; подпрограмма 1 с*

*ldi r17, 55*

*d1: ldi r18,95*

*d2: ldi r19, 255*

*d3: dec r19*

*brne d3*

*dec r18*

*brne d2*

*dec r17*

*brne d1 ; подпрограмма 1 с*

*ret*

*delay2: ; подпрограмма 2 с*

*rcall delay1*

*rcall delay1*

*ret*

Расчёт задержки delay1:

T1 = 255\*(1+2) = 765 циклов

T2 = 95\*(765+1+2) = 72960 циклов

T3 = 55\*(72960 +1+2) = 4012965 циклов

T = 4012965 : 4 Мгц = 4012965 \* 0,25 мкс = 1003241,25 мкс = = ~1 с.

На рисунках 2 и 3 показано время до входа в цикл задержки и после исполнения цикла задержки.

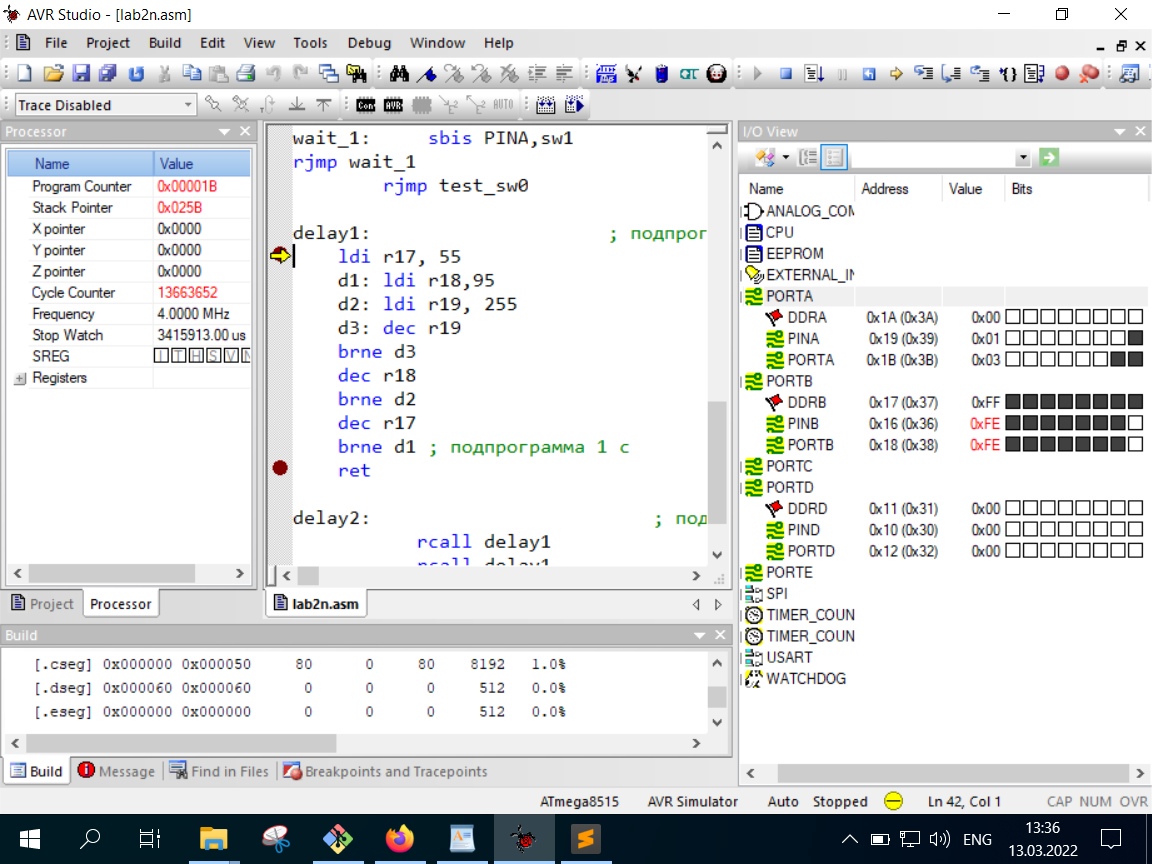


Рисунок 2 – время до цикла задержки

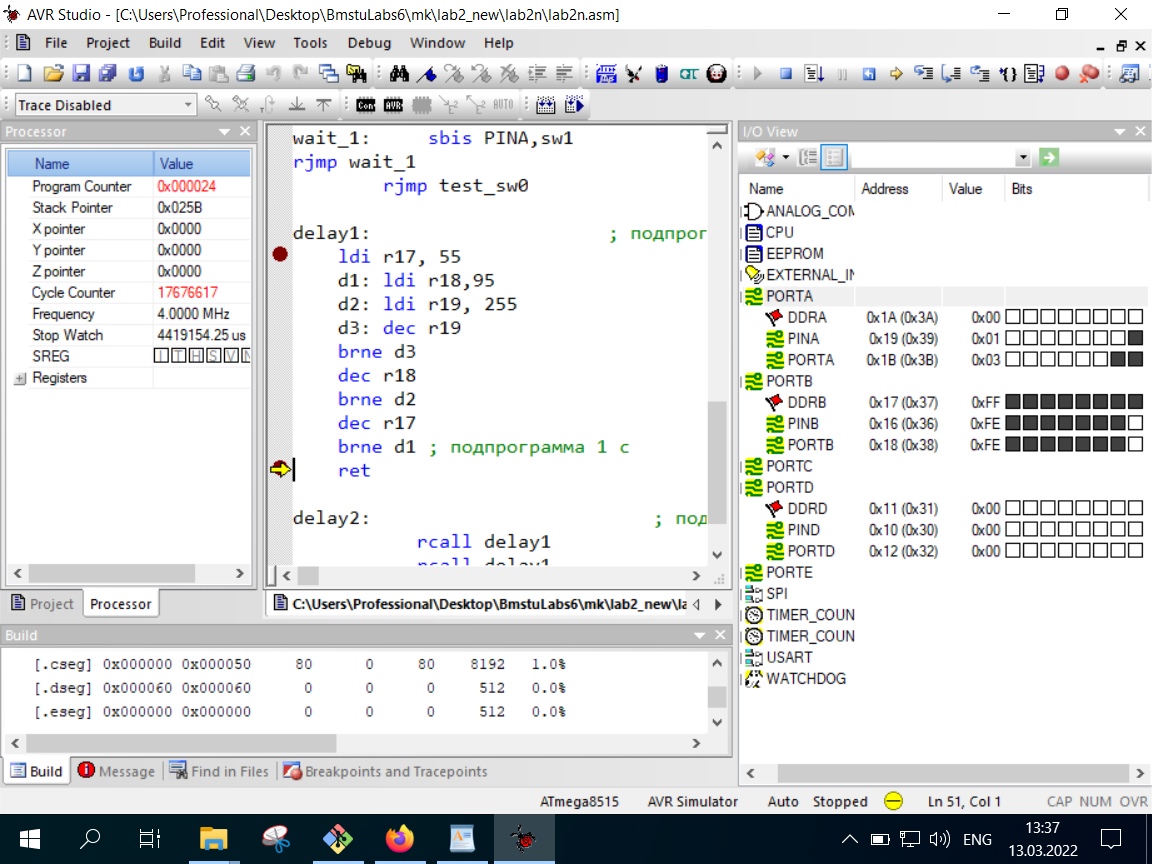


Рисунок 3 – время после цикла задержки

Практическое время задержки delay1 T = 4419154.25 - 3415913.00 мкс = 1003241.25 мкс = ~1 c. Время задержки было рассчитано верно.

Работа стека в ходе выполнения программы показана на рисунках 4-6.

Stack Pointer: 0x025F – адрес вершины стека.

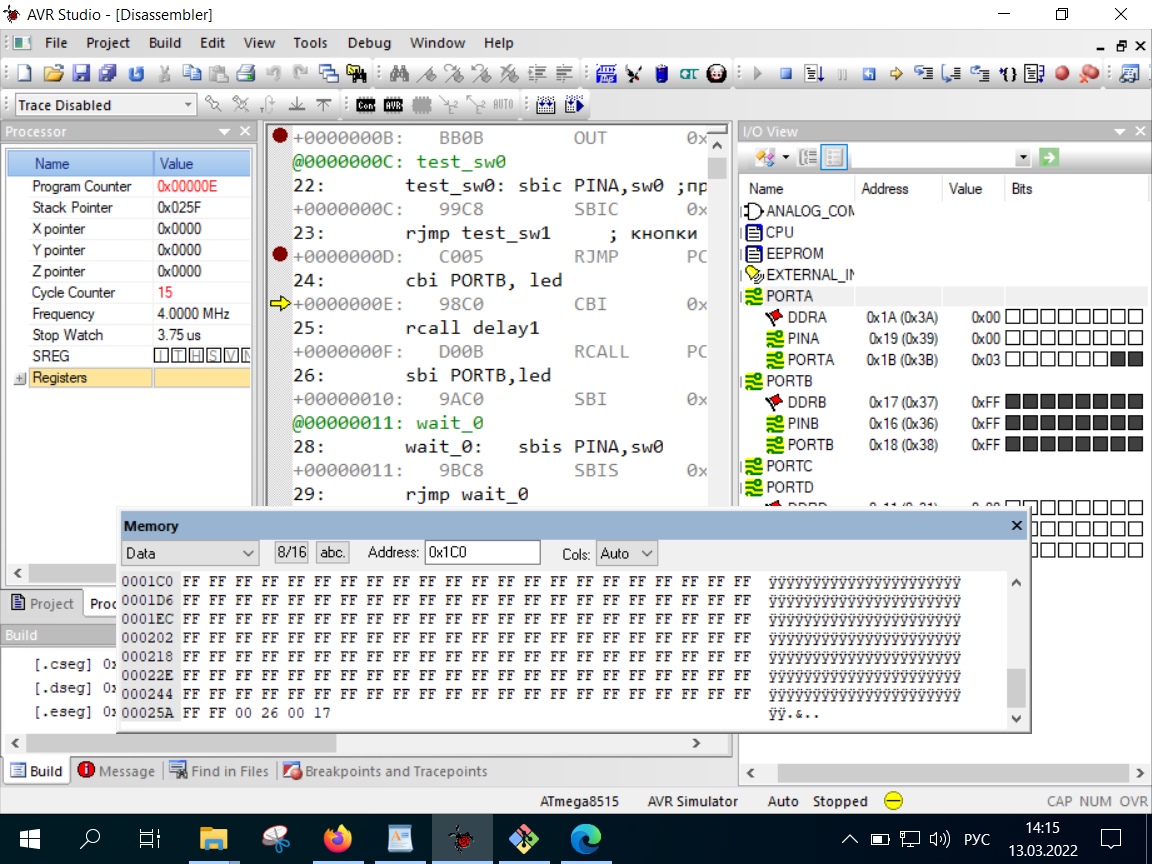


Рисунок 4 – состояние стека до передачи управления

Stack Pointer: 0x025D – в стек записался адрес возврата (адрес команды «sbi PORTB, led», идущей после вызова подрограммы «rcall delay1»), стек вырос в область младших адресов.

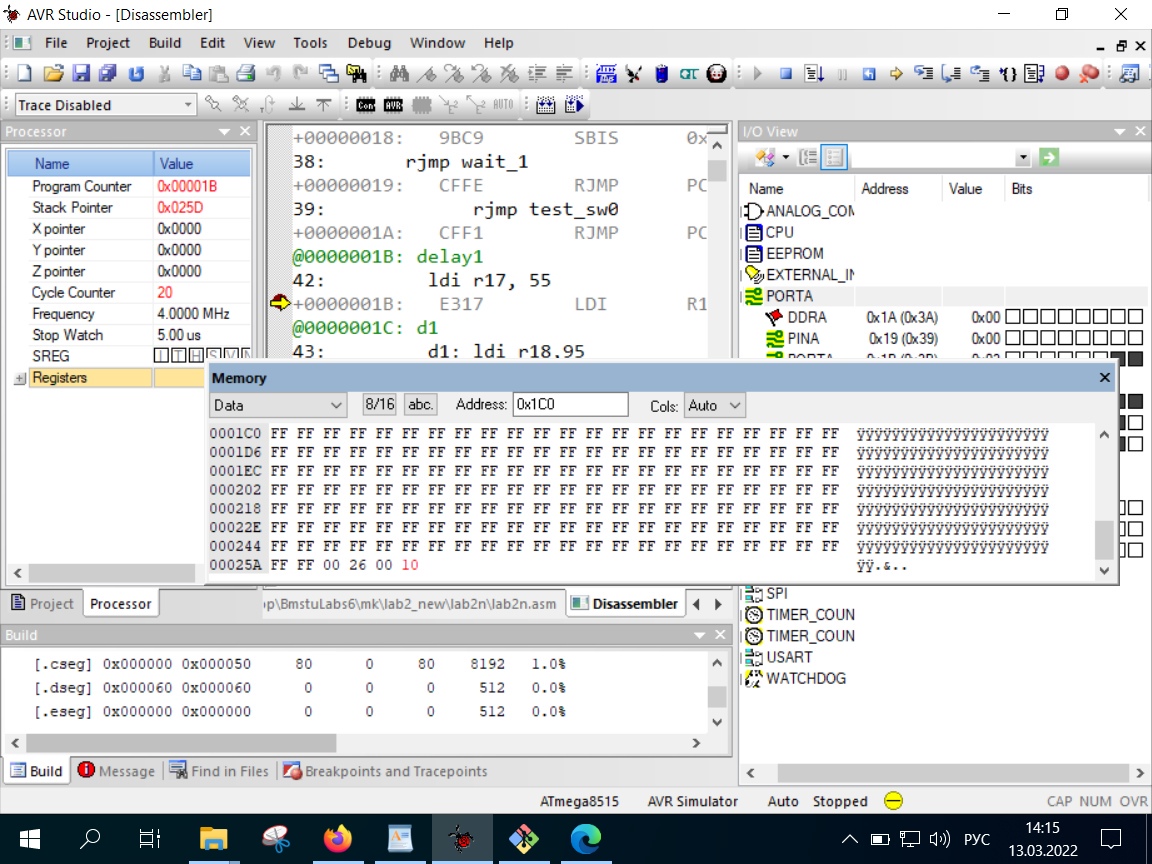


Рисунок 5 – стек после передачи управления

Stack Pointer: 0x025F – произошел возврат управления, вершина стека сместилась к старшим адресам, адрес возврата в стеке может быть перезаписан.

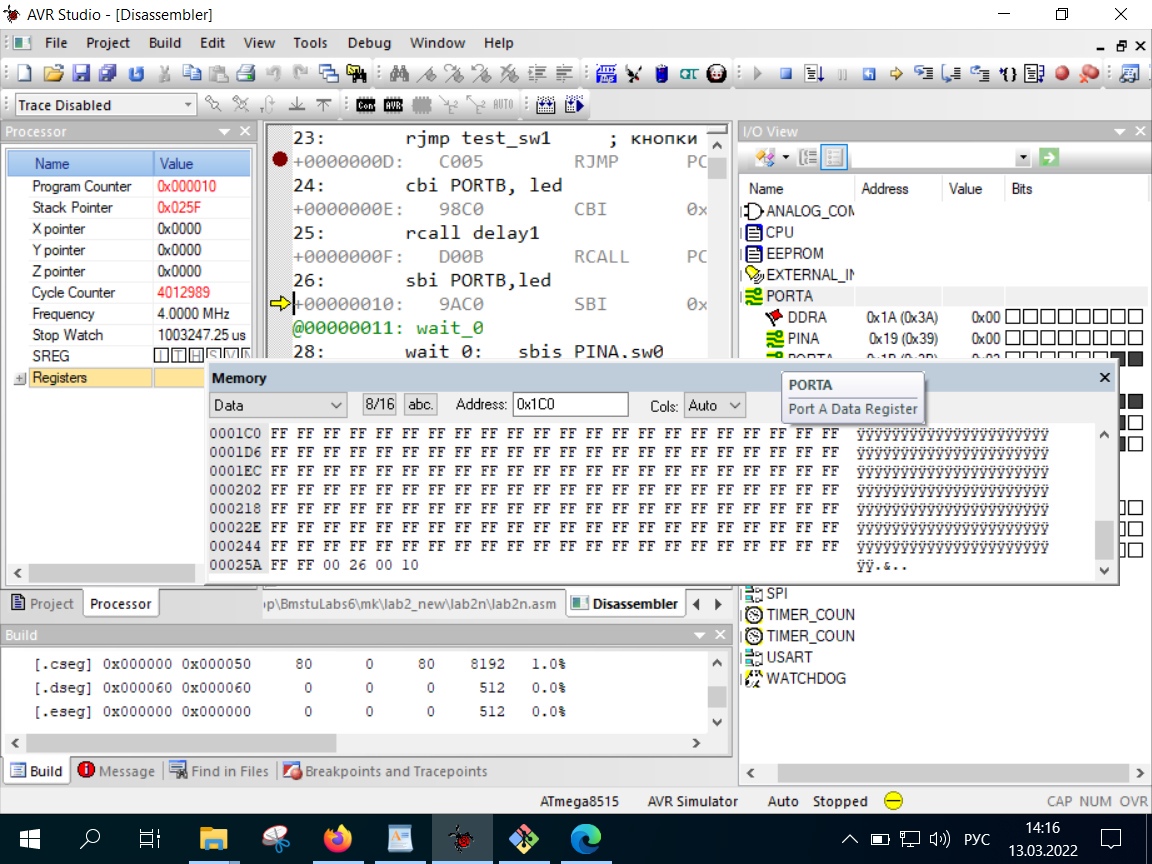


Рисунок 6 – стек после возврата управления

Как видно, при передаче управления адрес возврата записывает в стек, а указатель стека перемещается в область младших адресов. При возврате из подпрограммы указатель стека смещается в область старших адресов, позволяя перезаписывать ненужные теперь значения.

**Задание 2.**

Внести изменения и дополнения, касающиеся обработки прерываний, в исходный текст программы. На этапе инициализации указываются область стека для сохранения адресов возврата, при необходимости адреса векторов прерываний и сами векторы, маска прерываний, общее разрешение прерываний. Завершить инициализацию переводом процессора в фоновый режим ожидания.

Схема контроллера и схемы алгоритмов приведены на рисунках 7 и 8.

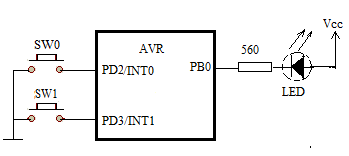


Рисунок 7 - контроллер с двумя прерываниями

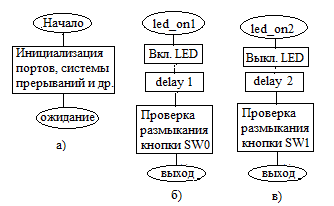


Рисунок 8 - схемы алгоритмов основной программы (а) и прерываний (б, в)

Исходный код программы:

*;Соединения на плате STK500: SW0-PD2, SW1-PD3, LED0-PB0*

*;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*.include "m8515def.inc" ;файл определений для ATmega8515*

*.def temp = r16 ;временный регистр*

*.equ led = 0 ;0-о бит порта PB*

*.equ sw0 = 2 ;2-й бит порта PD*

*.equ sw1 = 3 ;3-й бит порта PD*

*.org $000*

*;\*\*\*Таблица векторов прерываний, начиная с адреса $000\*\*\**

*rjmp INIT ;обработка сброса*

*rjmp led\_on1 ;на обработку запроса INT0*

*rjmp led\_on2 ;на обработку запроса INT1*

*;\*\*\*Инициализация SP, портов, регистра маски\*\*\**

*INIT: ldi temp,$5F ;установка*

*out SPL,temp ; указателя стека*

*ldi temp,$02 ; на последнюю*

*out SPH,temp ; ячейку ОЗУ*

*ser temp ;инициализация выводов*

*out DDRB,temp ; порта PB на вывод*

*out PORTB,temp ;погасить СД*

*clr temp ;инициализация*

*out DDRD,temp ; порта PD на ввод*

*ldi temp,0b00001100 ;включение ‘подтягивающих’*

*out PORTD,temp ; резисторов порта PD*

*ldi temp,((1<<INT0)|(1<<INT1));разрешение прерываний*

*out GICR,temp ; в 6,7 битах регистра маски GICR*

*ldi temp,0 ;обработка прерываний*

*out MCUCR,temp ; по низкому уровню*

*sei ;глобальное разрешение прерываний*

*loop: nop ;режим ожиданий*

*rjmp loop*

*led\_on1:*

*cbi PORTB,led*

*rcall delay1*

*sbi PORTB,led*

*wait\_0: sbis pind,sw0*

*rjmp wait\_0*

*reti*

*led\_on2:*

*cbi PORTB,led*

*rcall delay2*

*sbi PORTB,led*

*wait\_1: sbis pind,sw1*

*rjmp wait\_1*

*reti*

*delay1:*

*;для подпрограммы задержки 1 c*

*ldi r17, 55*

*d1: ldi r18,95*

*d2: ldi r19, 1;255*

*d3: dec r19*

*brne d3*

*dec r18*

*brne d2*

*dec r17*

*brne d1 ; подпрограмма 1 с*

*ret*

*delay2: ;подпрограмма задержки 2 c*

*rcall delay1*

*rcall delay1*

*ret*

Циклы задержки аналогичны циклам задержки из задания 1. Работа стека показана на рисунка 9-14.

Stack Pointer: 0x025F – адрес вершины стека.

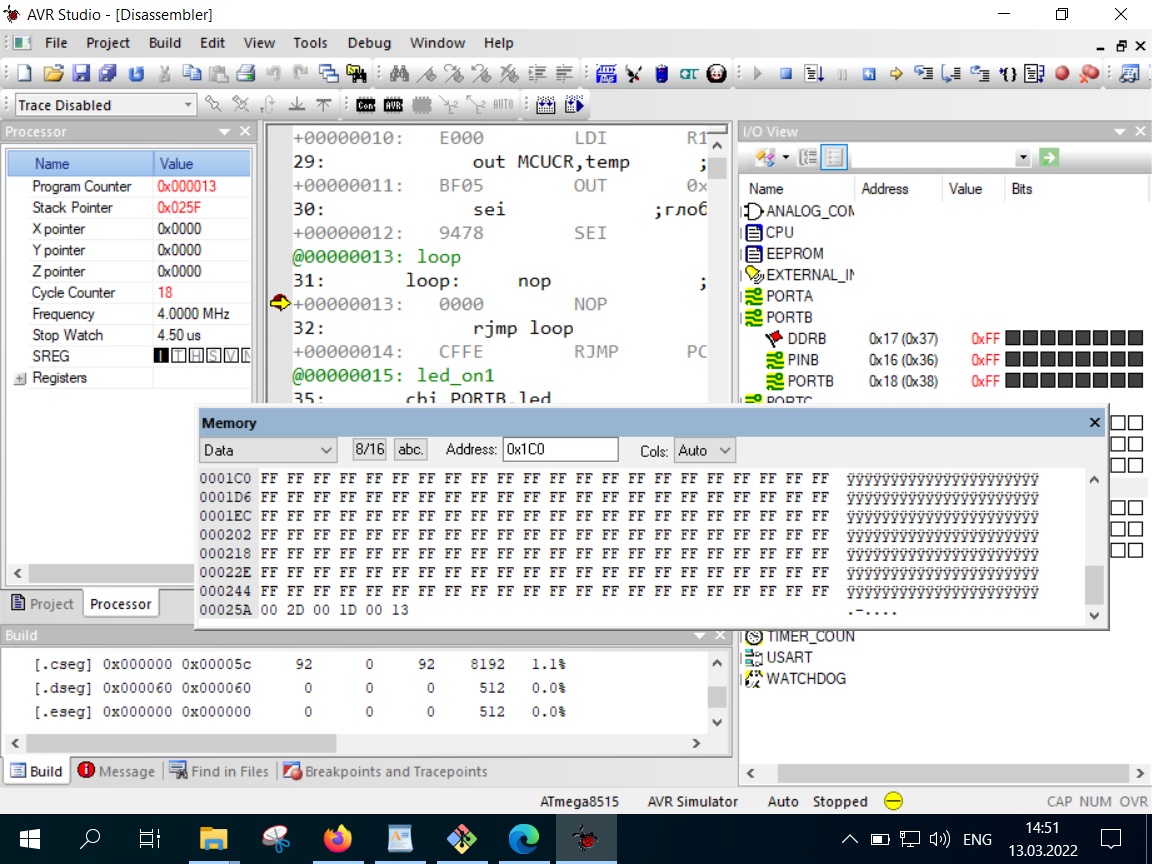


Рисунок 9 – содержимое стека до вызова прерывания

Stack Pointer: 0x025D – при вызове прерывания в стек записывается адрес возврата (в данном случае, адрес команды «rjmp loop», которая должна была быть исполнена следующей, если бы не произошло прерывания). Стек растет в область младших адресов.

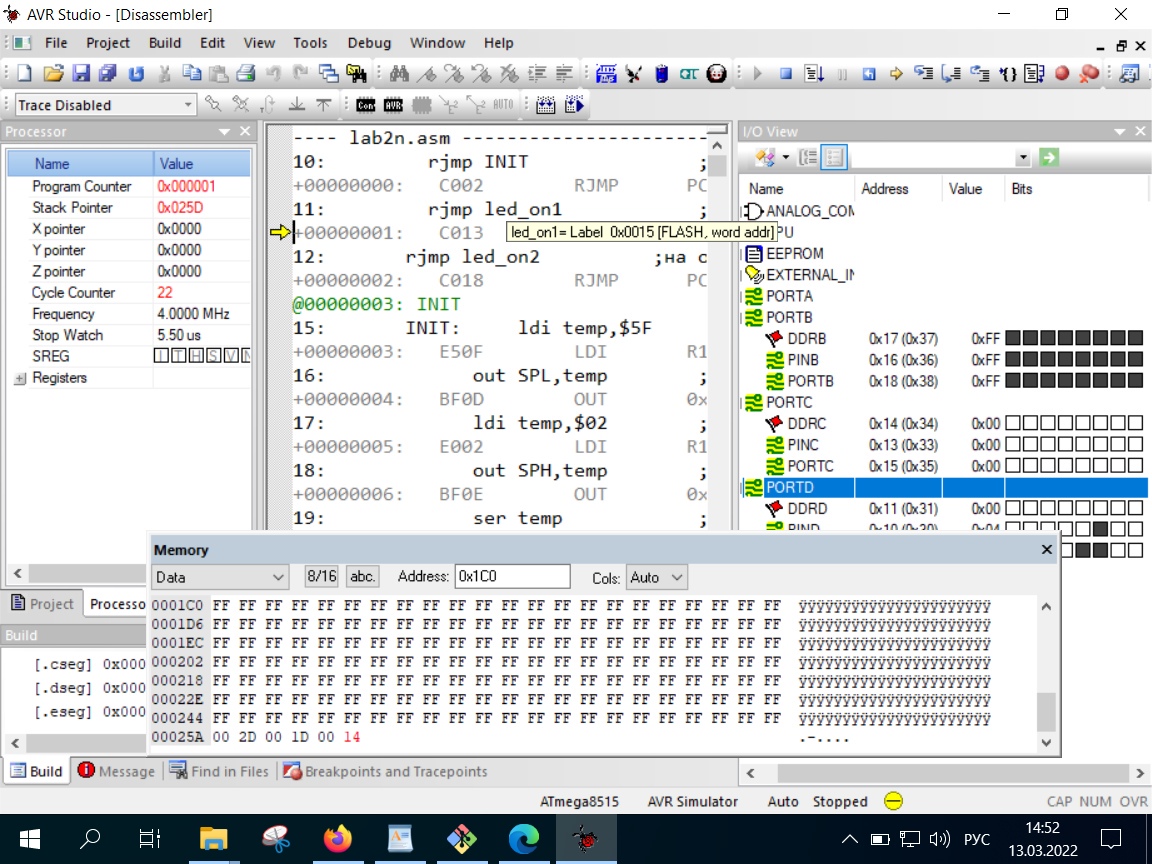


Рисунок 10 – содержимое стека после вызова прерывания

Stack Pointer: 0x025D – состояние стека аналогично предыдущему пункту, скриншот приведен для демонстрации адресов команд в обработчике прерываний.

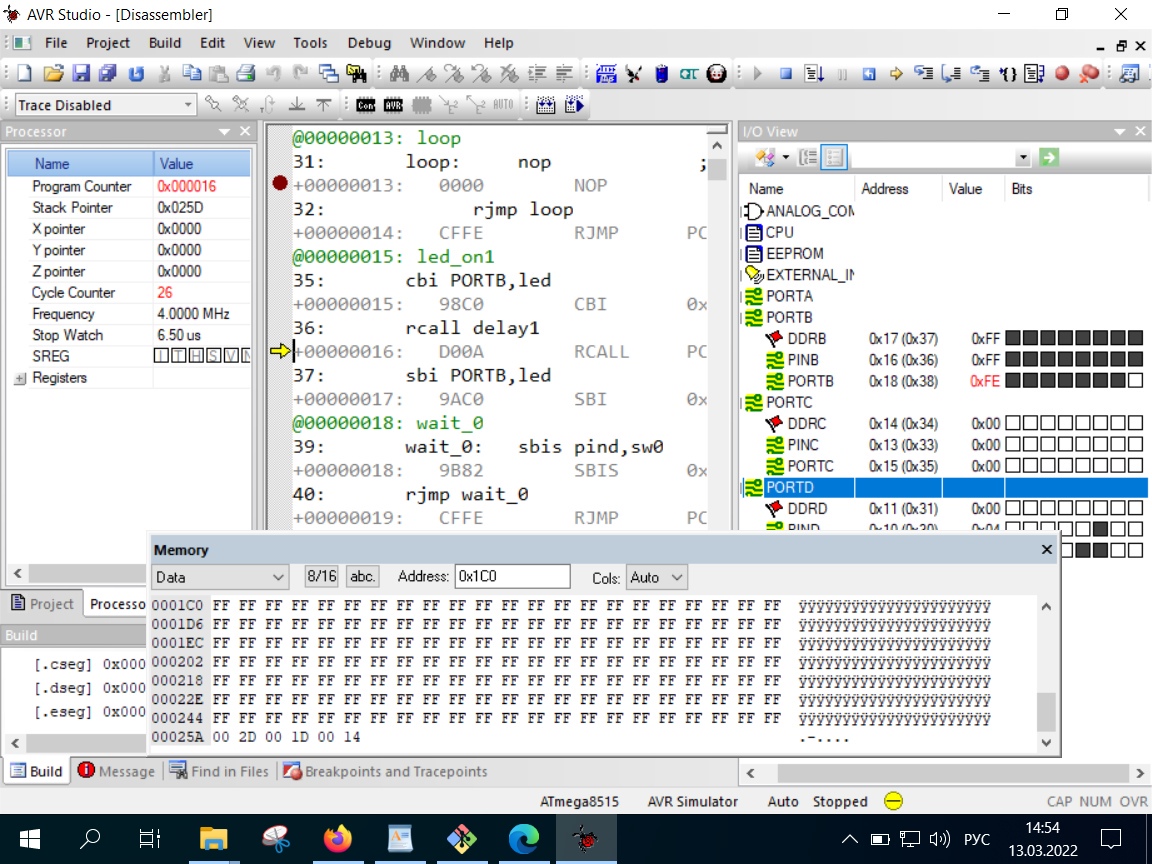


Рисунок 11 – содержимое стека после вызова прерывания, до вызова подпрограммы

Stack Pointer: 0x025B – В стек записался адрес возврата (адрес команды «rcall delay1» + 1) , стек вырос в область младших адресов.

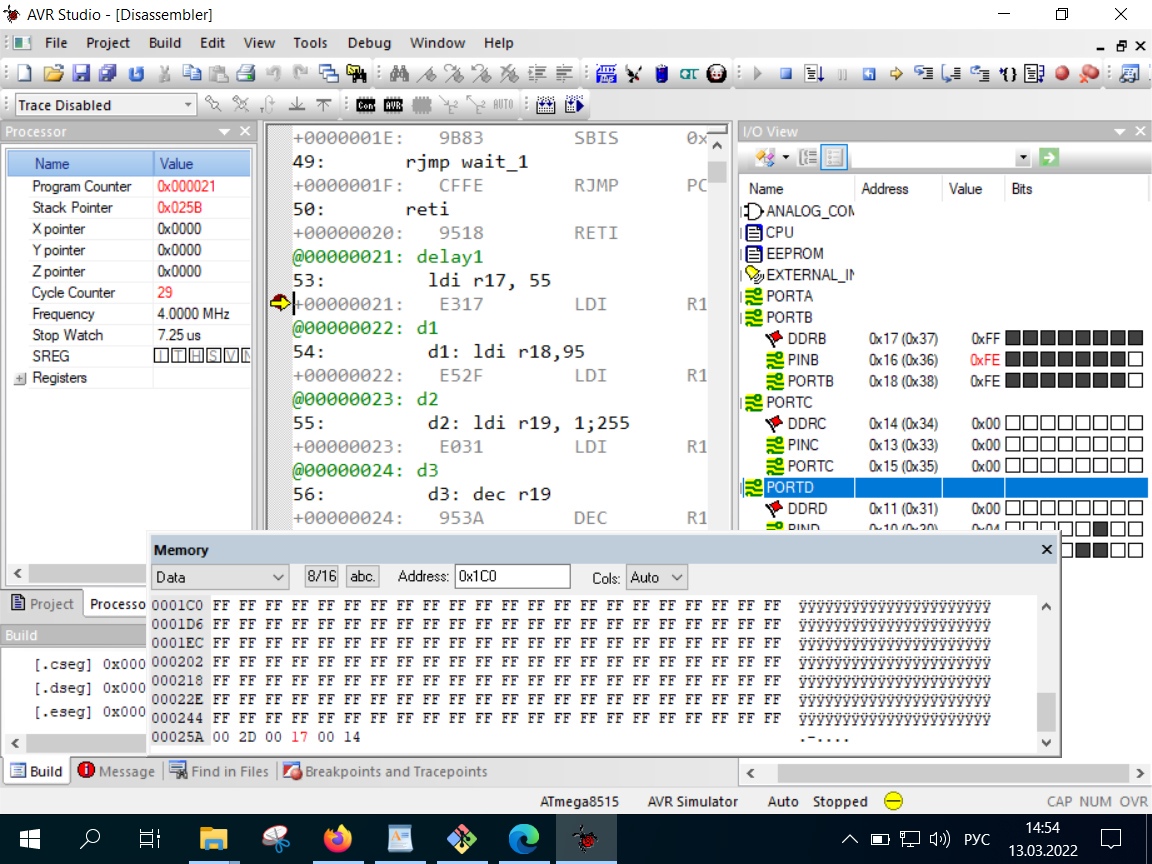


Рисунок 12 – содержимое стека после вызова прерывания и подпрограммы

Stack Pointer: 0x025D – после возврата управления вершина стека сместилась к старшим адресам.

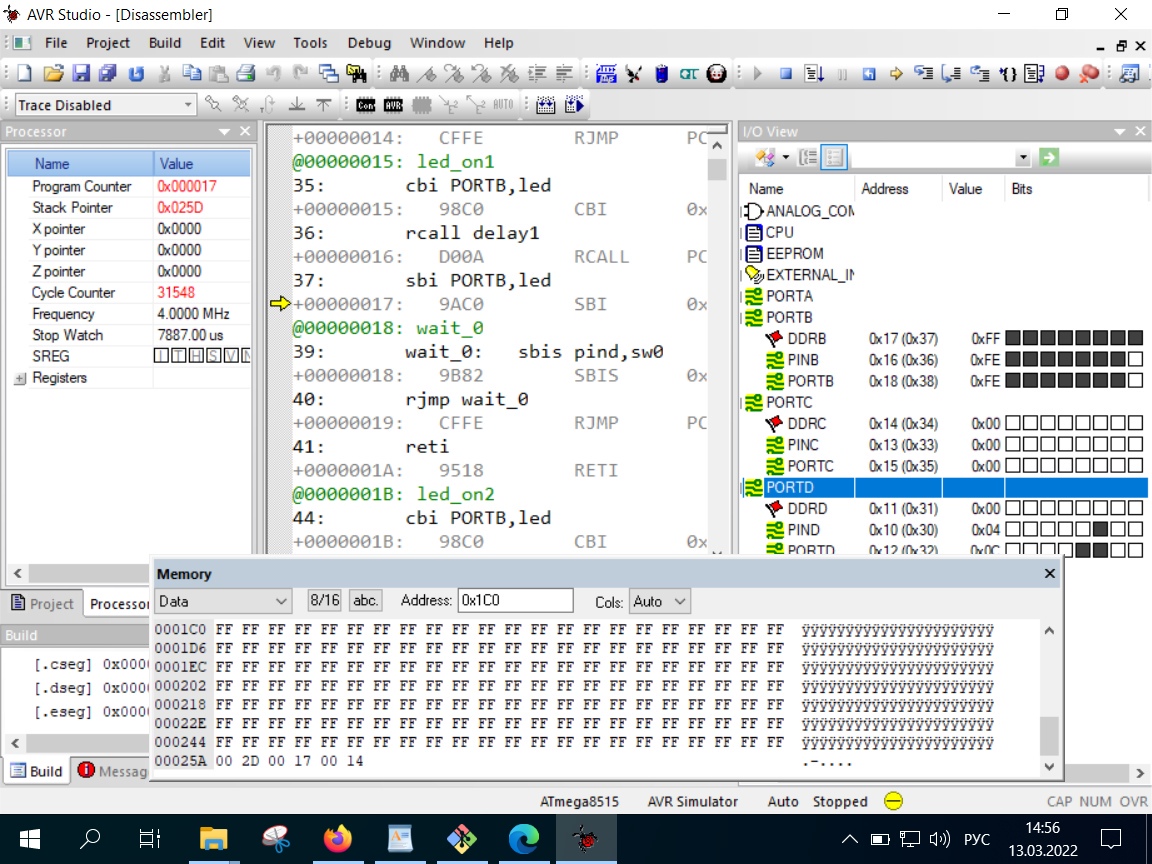


Рисунок 13 – содержимое стека после возврата из подпрограммы

Stack Pointer: 0x025F - после еще одного возврата управления вершина стека снова сместилась к старшим адресам.

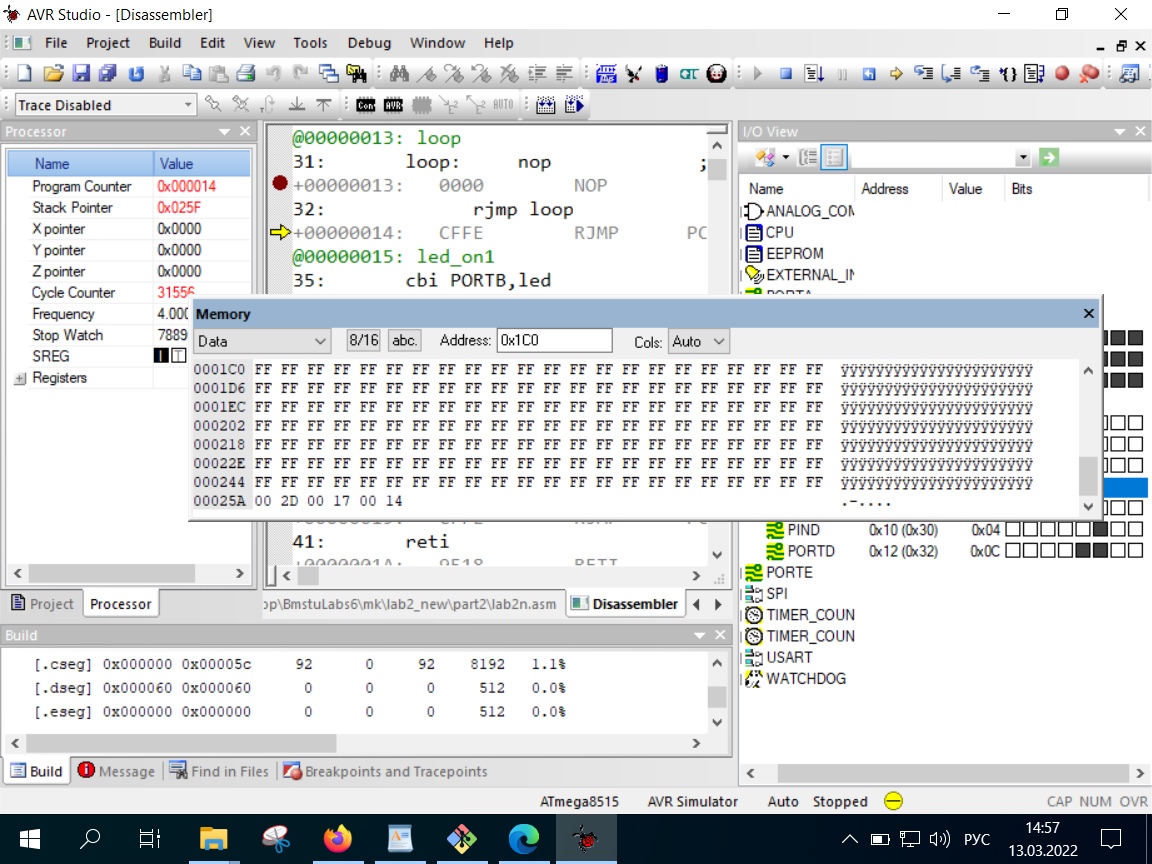


Рисунок 14 – содержимое стека после возврата из подпрограммы и обработчика прерываний

Стек работает аналогично заданию 1, но теперь передачи управления две – при вызове прерывания и при обычном вызове подпрограммы.

**Задание 3.**

Схема алгоритма представлена на рисунке 15.

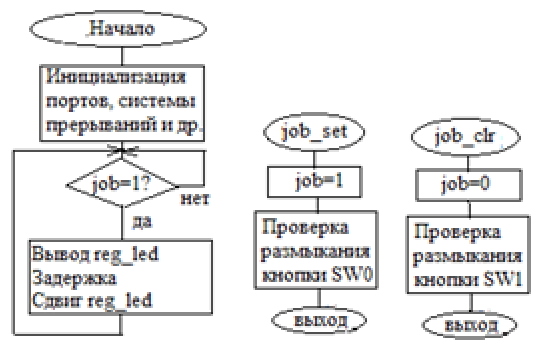


Рисунок 15 – схема алгоритма

Подготовить программу, соответствующую заданному алгоритму работы. При инициализации помимо общих директив установить исходный управляющий код в регистре индикации, нулевой разряд которого инициирует зажигание светодиода, настроить на вывод порт микроконтроллера и указатель стека.

В цикле алгоритма на каждой итерации выполнять вывод в порт микроконтроллера управляющего слова, временную задержку, затем циклический сдвиг влево управляющего слова.

Исходный код программы:

*.include "m8515def.inc"*

*.def job = r21*

*.def temp = r16*

*.def reg\_led = r20*

*.equ sw3 = 3*

*.equ sw0 = 0*

*.org $000*

*rjmp INIT*

*.org $002*

*rjmp job\_set*

*.org $00D*

*rjmp job\_clr*

*INIT: ldi temp,$5F*

*out SPL,temp*

*ldi temp,$02*

*out SPH,temp*

*ser temp*

*out DDRB,temp*

*out PORTB,temp*

*clr temp*

*out DDRD,temp*

*ldi temp,0b00001000*

*out PORTD,temp*

*ldi temp,0b00000001*

*out PORTE,temp*

*ldi temp,((1<<INT2)|(1<<INT1))*

*out GICR,temp ;*

*ldi temp,0b00000011*

*out MCUCR,temp*

*ldi temp,0b00000001*

*out EMCUCR,temp*

*ldi reg\_led, 0xFE*

*ldi temp, 0xFF*

*ldi job, 0x00*

*sec*

*sei*

*loop: sbrs job,0*

*rjmp loop*

*out PORTB,reg\_led*

*rcall delay*

*rol reg\_led*

*rjmp LOOP*

*end: rjmp loop*

*job\_set: ldi job,1*

*wait\_0: sbis pind,sw3*

*rjmp wait\_0*

*reti*

*job\_clr: clr job*

*wait\_1: sbis pine,sw0*

*rjmp wait\_1*

*reti*

*delay:*

*ldi r17, 215*

*d1: ldi r18,255*

*d2: ldi r19, 4*

*d3: dec r19*

*brne d3*

*dec r18*

*brne d2*

*dec r17*

*brne d1*

*ret*

**Задание 4.**

Объединить запросы прерываний от двух кнопок. При нажатии кнопки SW0 включается светодиод на 1 с, кнопки SW1 - на 2 с. Схема проекта представлена на рисунке 16.

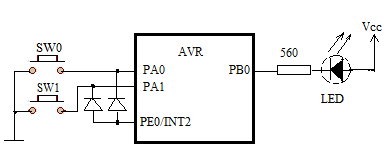


Рисунок 16 – схема проекта

Схема алгоритма представлена на рисунке 17.

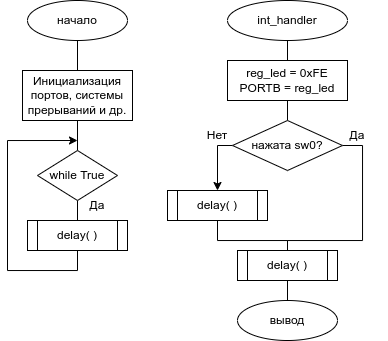


Рисунок 17 – схема алгоритма

Составить программу согласно описанному алгоритму работы. Отладить работу программы в пошаговом режиме в среде AVR Studio.

Исходный код программы:

*.include "m8515def.inc"*

*.def job = r21*

*.def temp = r16*

*.def reg\_led = r20*

*.equ sw0 = 0*

*.equ sw1 = 1*

*.org $000*

*rjmp INIT*

*.org $00D*

*rjmp int\_handler*

*INIT: ldi temp,$5F*

*out SPL,temp*

*ldi temp,$02*

*out SPH,temp*

*ser temp*

*out DDRB,temp*

*out PORTB,temp*

*clr temp*

*out DDRA,temp*

*ldi temp,0b00000011 ;подтягивающие*

*out PORTA,temp*

*ldi temp,0b00000001*

*out PORTE,temp*

*ldi temp,(1<<INT2)*

*out GICR,temp ;*

*;ldi temp,0b00000011 ;LH*

*;out MCUCR,temp*

*ldi temp,0b00000000*

*out EMCUCR,temp*

*ldi reg\_led, 0b11111111*

*ldi temp, 0xFF*

*ldi job, 0x00*

*sec*

*sei*

*loop:*

*rcall delay*

*rjmp loop*

*int\_handler:*

*ldi reg\_led, 0xFE*

*out portb, reg\_led*

*sbic pina, sw0*

*rcall delay*

*rcall delay*

*ldi reg\_led, 0xFF*

*out portb, reg\_led*

*reti*

*delay:*

*ldi r17, 235*

*d1: ldi r18,255*

*d2: ldi r19, 30*

*d3: dec r19*

*brne d3*

*dec r18*

*brne d2*

*dec r17*

*brne d1*

*ret*

Собрать схему для моделирования в среде ISIS Proteus (рисунок 18).

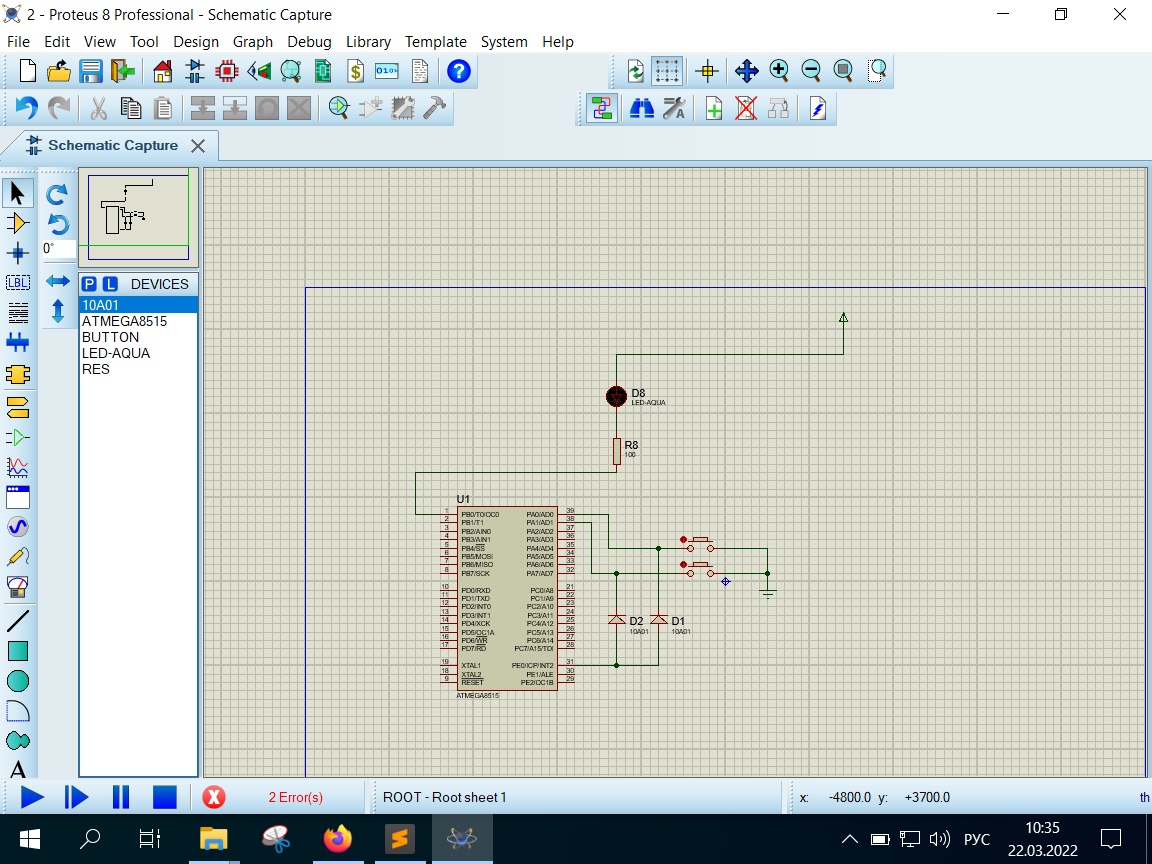


Рисунок 18 – схема в Proteus

Проверка работы программы посредством симуляции в Proteus показала, что программа работает корректно.

**Задание 5.**

Собрать схему на плате STK500. Загрузить программу в память микроконтроллера и проверить работу программы на макете.

**Вывод:** в ходе выполнения лабораторной работы была изучена система прерываний в микроконтроллерах AVR, работа стека при вызове подпрограмм и прерываний, получены навыки программирования внешних прерываний. Также, был получен опыт объединения сигналов прерываний с последующим распознаванием источника прерываний.