

єЛАБОРАТОРНА РОБОТА № 1

СЕРЕДОВИЩЕ РОЗРОБКИ ПРОГРАМ ДЛЯ МК AVR

Мета: навчитися використовувати для написання програм інтегроване середовище розробки AVR Studio та пакету ISIS Proteus.

1 Теоретичні відомості

1.1 Загальні відомості при роботі з AVR Studio

AVR Studio 4 – професійне інтегроване середовище розробки (Integrated Development Environment - IDE), призначене для написання і налагодження прикладних програм для мікроконтролерів AVR в середовищі Windows NT/XP та містить асемблер і симулятор.

Програмування в середовищі AVR Studio зазвичай виконується в такій послідовності:

- створення проекту;
- написання програми;
- компіляція;
- симуляція.

Створення проекту

При запуску AVR Studio пропонується створити новий проект (New Project) або відкрити вже існуючий (Open).

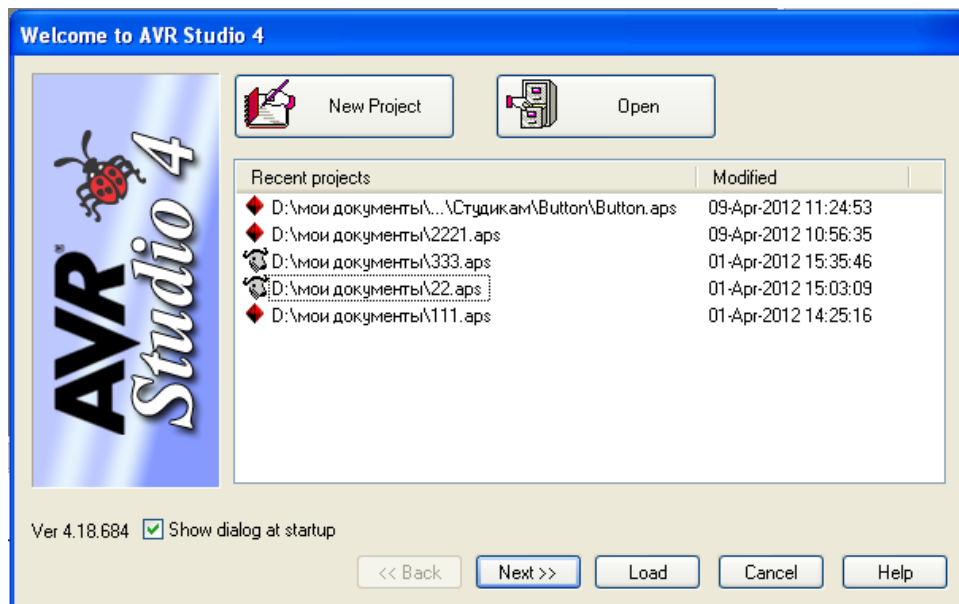


Рисунок 1.1

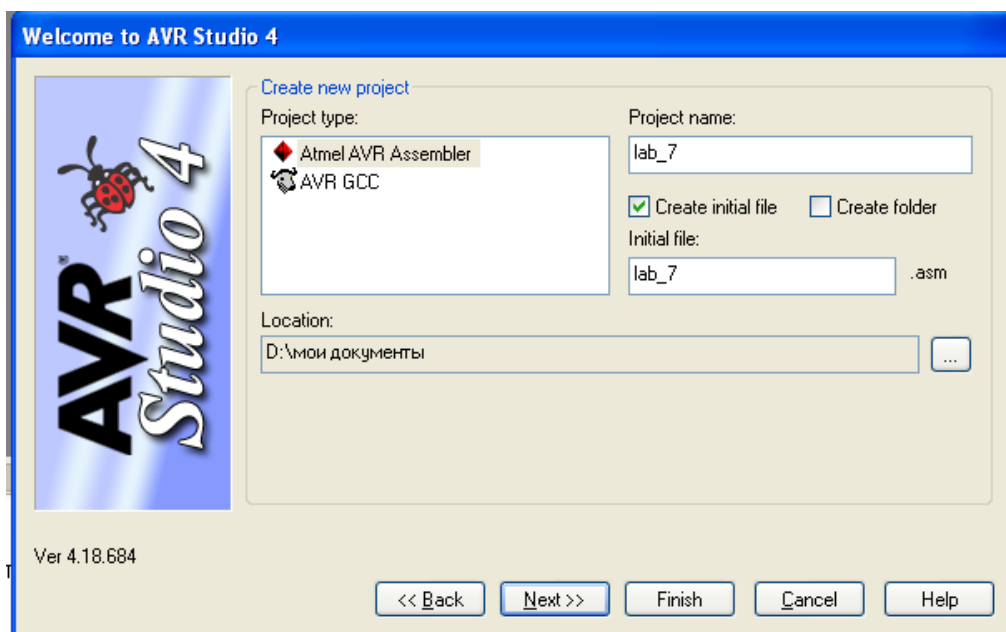


Рисунок 1.2 – Створення нового проекту

У вікні тип проекту (Project Type) вибираємо асемблер (Atmel AVR Assembler), заповнюємо поля ім'я проекту (Project Name) і заголовний файл (Initial File). Натискаємо далі (Next) ...

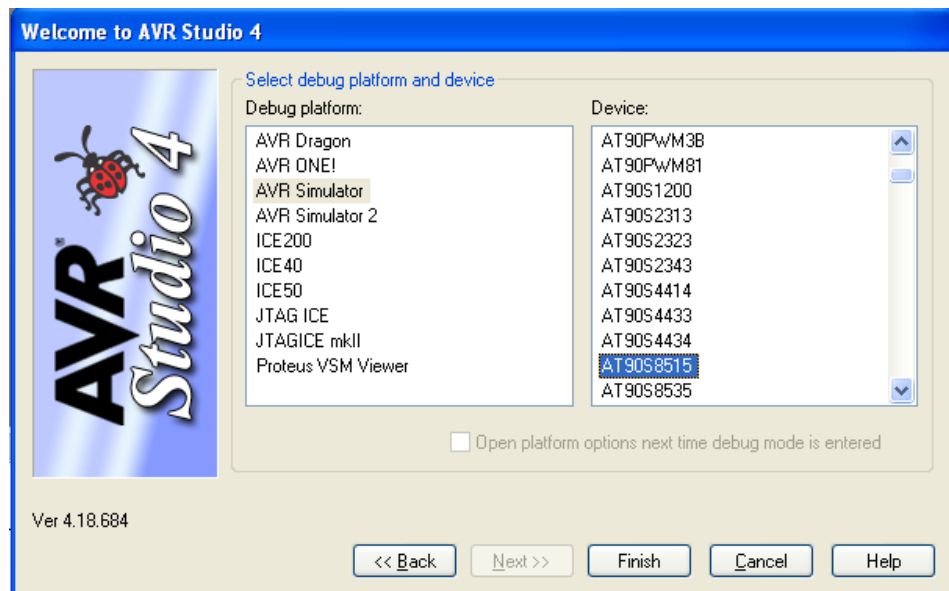


Рисунок 1.3 – Вибір симулятора та МК

У вікні платформа відладки (Debug Platform) вибираємо симулятор, а у вікні пристрій (Device) – відповідний мікроконтролер (в даному варіанті AT90S8515). Натискаємо завершити (Finish) – на даному етапі проект створений. Після переходимо в головне вікно програми.

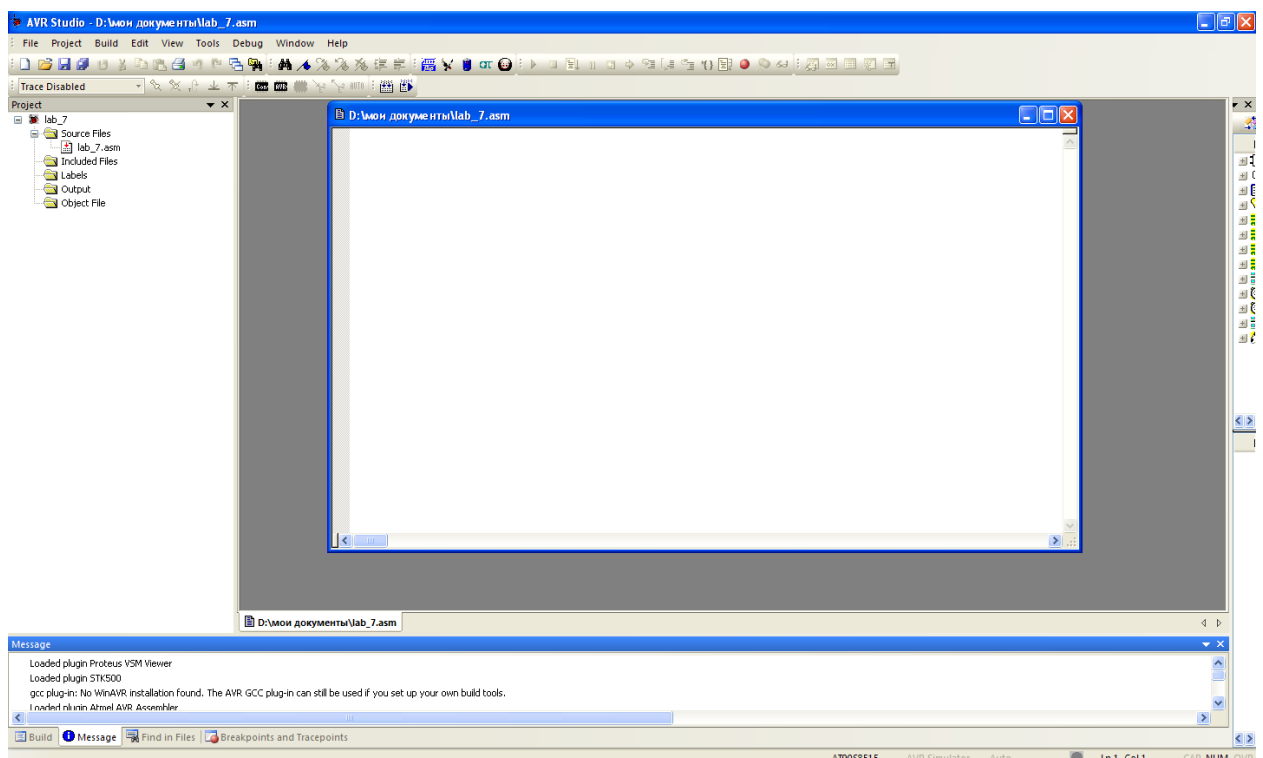


Рисунок 1.4 – Загальний вигляд вікна програми

Вікно розділене на 4 частини. У верхній частині знаходиться рядок меню і «плаваючі» панелі з кнопками. Трохи нижче ліворуч знаходяться вкладки **Диспетчер проекту (Project)**, **Перегляд вводу/виводу (I/O View)**, **Інформація (Info)**, праворуч – **Текст програми**. Знизу знаходяться наступні вкладки: **Конструкція (Build)**, **Повідомлення (Message)**, **Пошук в файлах (Find in Files)**, **Контрольні точки (Breakpoints and Tracerpoints)**.

У вікні **Текст програми** користувач створює програму.

Для першого знайомства можна взяти програму з Додатка 1.

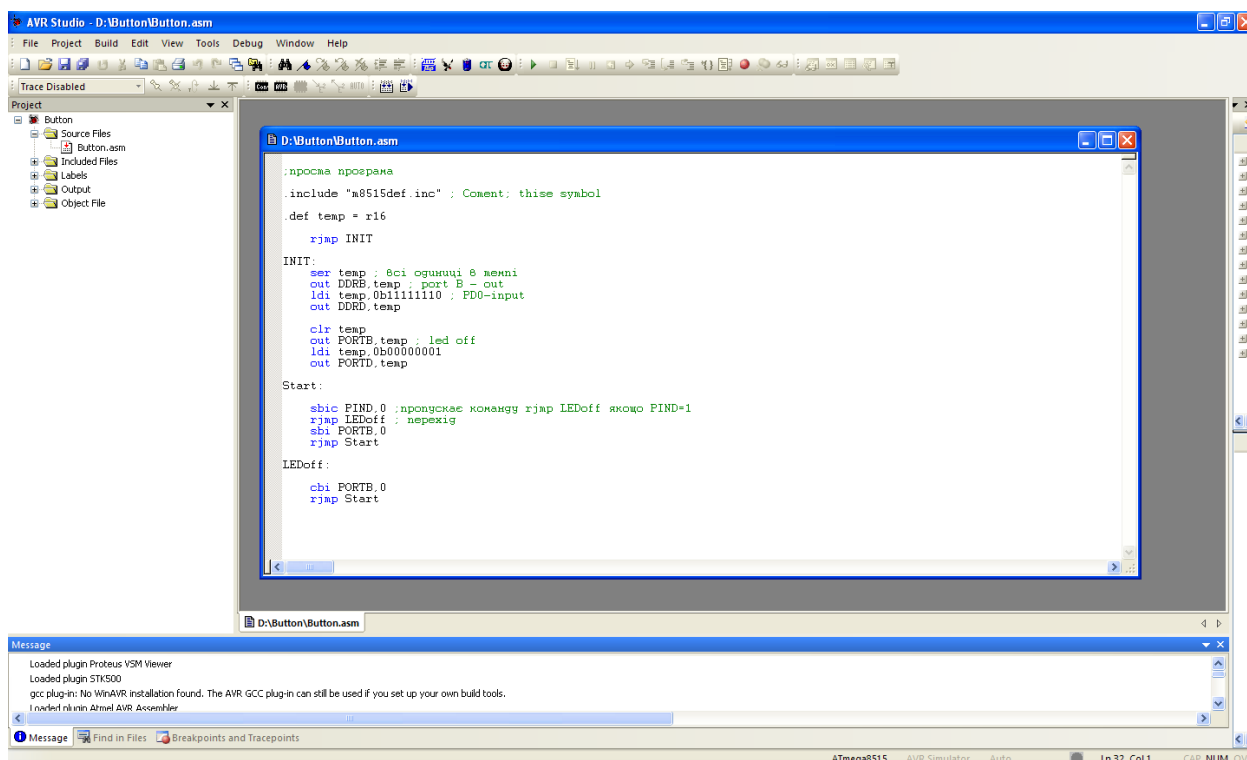


Рисунок 1.5 – Написання тексту програми

При написанні програми, текстовий редактор середовища забезпечує під світку синтаксису – інструкції виділяються синім кольором, коментарі - зеленим, інше - чорним.

При написанні ПО слід не забувати періодично зберігати внесені в програму зміни.

Компіляція – процес перекладу тексту програми, написаної мовою програмування, в виконуваний модуль, що містить машинні команди конкретного процесора. Процес компіляції з мови асемблера називається асемблюванням (утворення *.hex файлу).

Асемблювання – трансляція з мови асемблера в команди машинної мови.



Дані кнопки на верхній панелі запускають процес асемблювання. Кнопка зліва асемблює проект, справа - асемблює і запускає на виконання.

Якщо при написанні тексту програми були допущені синтаксичні помилки, компіляція переривається і на вкладці **Конструкція** виводяться повідомлення про допущені помилки.

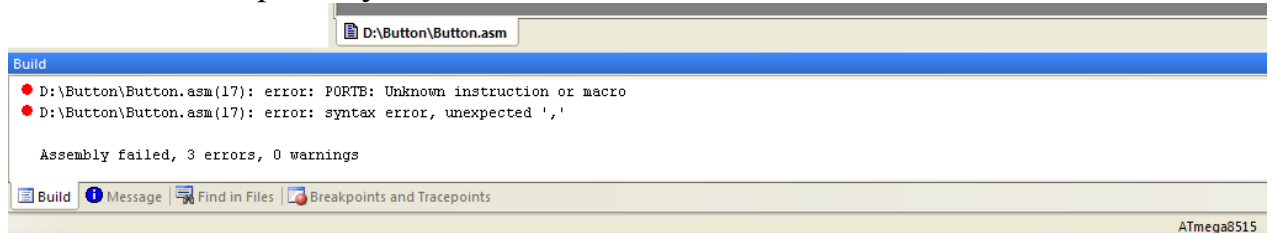


Рисунок 1.6 – Вкладка Конструкція при синтаксичних помилках

При вдалій компіляції на вкладці **Конструкція** показується звіт про проходження процесу асемблювання і таблиця використаних ресурсів.

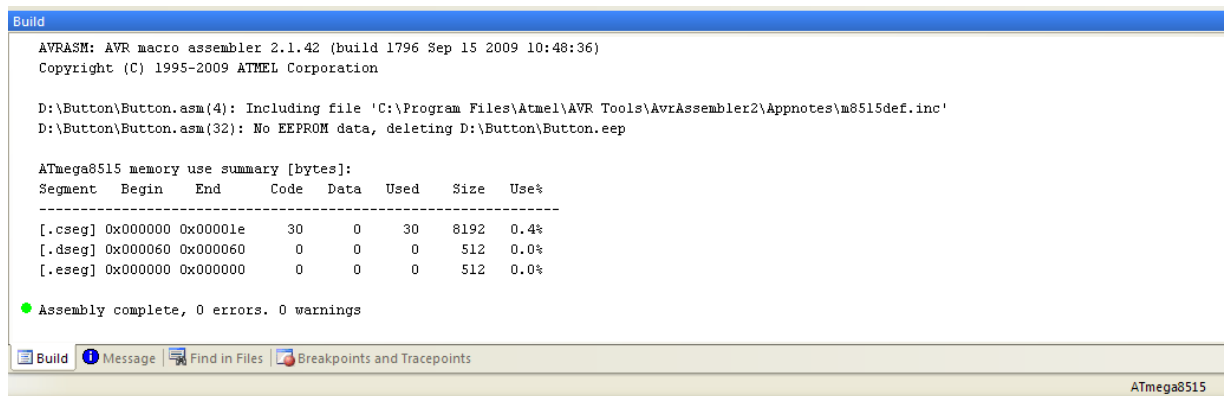


Рисунок 1.7 – Вкладка Конструкція при вдалій компіляції

Після вдалого асемблювання можна переходити до фази симуляції.

Контрольна точка – інструкція в програмі, дійшовши до якої виконання програми призупиниться. Встановлена контрольна точка відзначається у редакторі червоним кружком.

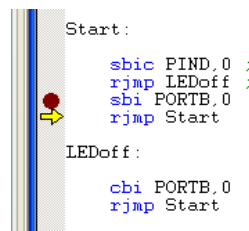















Рисунок 1.8 – Контрольна точка

Симуляція – моделювання процесу виконання програми мікроконтролером на персональному комп'ютері. Являється одним із режимів налагодження (Debugging).

Налагодження – етап комп'ютерного розв'язання задачі, при якому відбувається усунення явних помилок у програмі. Часто проводиться з використанням спеціальних програмних засобів - відладчиків.

Для управління режимом налагодження призначені наступні кнопки.

	Запустити відладку (симуляцію).
	Зупинити відладку .
	Запустити програму на виконання.
	Пауза у виконанні програми.
	Показати виконувану інструкцію.
	Перезапустити програму.
	Крок вперед із заходом в підпрограми.
	Крок вперед без заходу в підпрограми.
	Перейти до останньої інструкції програми (підпрограми).
	Виконати програму до місця вказаного курсором.
	Автоматичне покрокове виконання програми.
	Встановити / зняти контрольну точку.
	Видалити всі контрольні точки.

Інструкція, яка буде виконуватися наступною, позначається жовтою стрілкою.

Інформація про регістри введення/виведення, процесор і регістри загального призначення розташована і розподілена по групах в вкладці.

Після успішної компіляції проект можна завантажувати у контролер, в тому числі у пакеті Proteus.

Перегляд вводу/виводу.

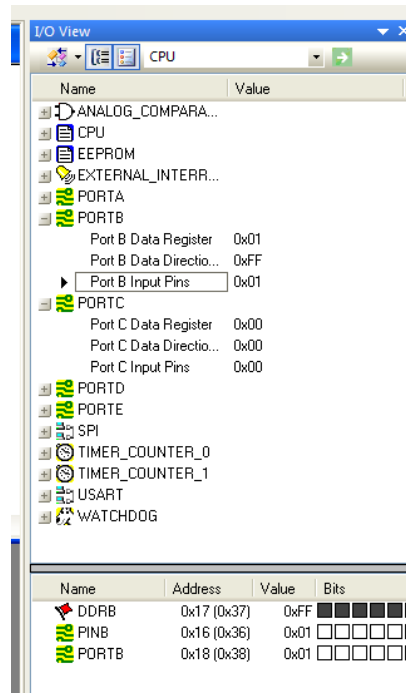


Рисунок 1.9 – Вкладка вводу/виводу

1.2 Загальні відомості при роботі з Proteus

Proteus – це комерційний пакет програм класу САПР, що об'єднує у собі дві основні програми: ISIS – засіб розробки та налагодження в режимі реального часу електронних схем та ARES – засіб розробки друкованих плат.

Розробником пакету Proteus є фірма Labcenter Electronics Великобританія.

[http://radio-hobby.org/uploads/journal/RYB/2013/RYB_2013_24.pdf]

Proteus підтримує симуляцію МК: PIC, 8051, AVR, HC11, ARM7/LPC2000 та інших розповсюджених процесорів. Працює з більшістю компіляторів та асемблерів.

Proteus дозволяє достовірно моделювати та налагоджувати складні пристрої, в яких може міститися декілька МК.

Proteus містить велику бібліотеку електронних компонентів. Компоненти, що відсутні, можна створити. Якщо компонент не програмуємий, то необхідно на сайті виробника скачати його SPICE модель та додати в прийнятний корпус.

Proteus має можливість підключитися до реального USB та COM порту комп'ютера. [http://eldigi.ru/articles/proteus]

Інтерфейс програми ISIS з поясненнями представлений на рис.1.10

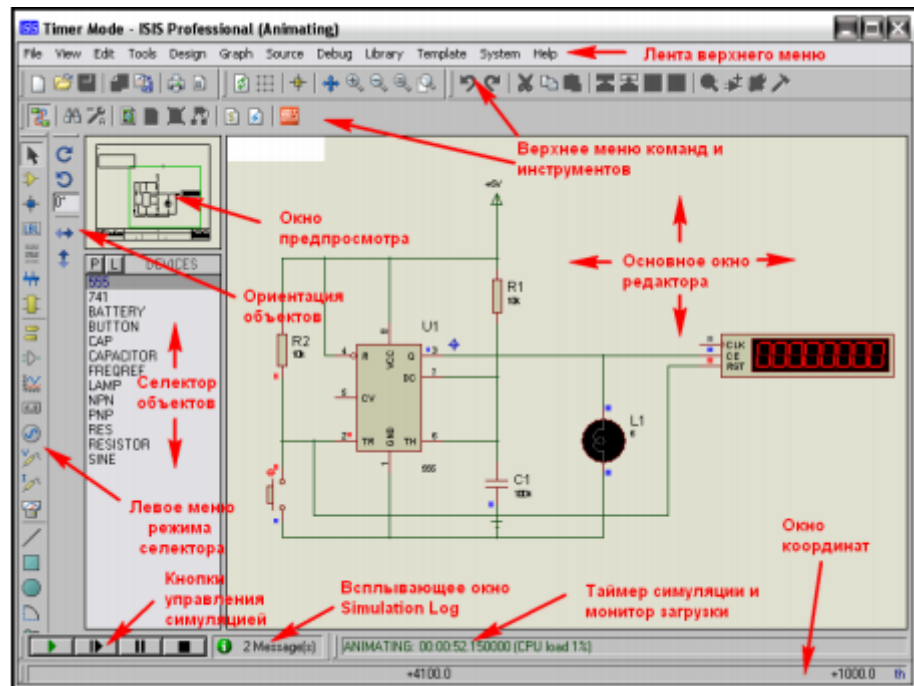


Рисунок 1.10 – Інтерфейс ISIS Proteus

2 Порядок виконання роботи

2.1 Згідно Таблиці 1.1 та Додатку 1 набрати тексти програм у AVR Studio. Програма повинна виконувати задану операцію згідно завдання над вводом числами. Сформувані необхідні значення операндів на входах портів А та В. Та зчитати виведенні на індикатори значення з портів С та D.

2.2 У пакеті ISIS Proteus згідно Додатку 2 створити схему та вибрати відповідних контролер (AT90S8515) і зашити у нього утворений hex файл і переконатись і працездатності програми.

2.3 Набрану програму скопіювати і налагодити, провівши покрокове виконання програми та відстежити, як змінюється вміст відповідних регістрів.

2.4 Виправити помилки, якщо вони є.

Таблиця 1.1 – Вхідні дані

PortA	PortB	Операція 1	Операція 2
1	2	+	mul
2	4	-	mul
3	6	+	xor
4	8	-	or
5	10	+	or
6	12	-	xor
7	14	+	and
8	16	-	and
9	18	mul	+
10	20	xor	-
11	22	or	+
12	24	or	-
13	26	xor	+
14	28	and	-
15	30	and	+

3 Зміст звіту

- 3.1 Назва та мета роботи.
- 3.2 Схема в ISIS Proteus.
- 3.3 Текст програми.
- 3.4 Результат виконання програми.
- 3.5 Висновки по роботі.

4 Контрольні запитання

- 4.1 Що таке AVR Studio?
- 4.2 Яка послідовність дій по створенню проекту та відпрацюванню помилок у середовищі AVR Studio?
- 4.3 Що таке компіляція?
- 4.4 Що таке асемблювання?
- 4.5 Що таке симуляція?
- 4.6 Що таке налагодження?
- 4.7 Що являє собою Proteus?
- 4.8 Які дві основні програми об'єднує Proteus?
- 4.9 Які основні можливості пакету програм Proteus?

Додаток 1

```
C:\Projects\Thermo\Thermo.asm

; register symbol names
.equ   SREG   = $3F
.equ   PORTA  = $1B
.equ   DDRA   = $1A
.equ   PINA   = $19
.equ   PORTB  = $18
.equ   DDRB   = $17
.equ   PINB   = $16
.equ   PORTC  = $15
.equ   DDRC   = $14
.equ   PINC   = $13
.equ   PORTD  = $12
.equ   DDRD   = $11
.equ   PIND   = $10

; Initialisation

    ser R16: R16 <- #0FF
    out PORTA, R16; R16 <- #0FF
    out PORTB, R16; R16 <- #0FF
    out DDRC, R16  ; R16 <- #0FF
    out DDRD, R16  ; R16 <- #0FF

; Main part
M0: in  R0, PINA    ; R0 <- PINA
    in  R1, PINB    ; R0 <- PINB
;zavdanya 3
eor R0, R1

    out PORTC, R0   ; PORTC <- PINA XOR
    out PORTD, R1   ; PORTD <- PINB

rjmp M0            ; PC <- M0

;swap R0
```

Додаток 2

