НАВЧАЛЬНО-МЕТОДИЧНА ЛІТЕРАТУРА

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Тернопільський національний технічний університет

імені Івана Пулюя

Кафедра автоматизації технологічних

процесів та виробництв

Методичні вказівки

до лабораторної роботи №11

«Ввід/вивід даних через порти мікроконтролера

на програмному симуляторі AVR Simulator IDE»

з курсу

«Мікропроцесорні та програмні засоби автоматизації»

Тернопіль 2020

Методичні вказівки до лабораторної роботи №11 «Ввід/вивід даних через порти мікроконтролера на програмному симуляторі AVR Simulator IDE» з курсу «Мікропроцесорні та програмні засоби автоматизації».

Методичні вказівки розглянуті і схвалені кафедрою «Автоматизація технологічних процесів та виробництв», протокол № 8 від 18.02.2020 р.

Відповідальні за випуск доцент, к.т.н. Медвідь В.Р.,
асистент Пісьціо В.П.

Лабораторна робота №11

Ввід/вивід даних через порти мікроконтролера на програмному симуляторі AVR Simulator IDE

1. Послідовність роботи з програмним симулятором AVR Simulator IDE

Основне вікно програми AVR Simulator IDE має вигляд, показаний на (рис. 1).

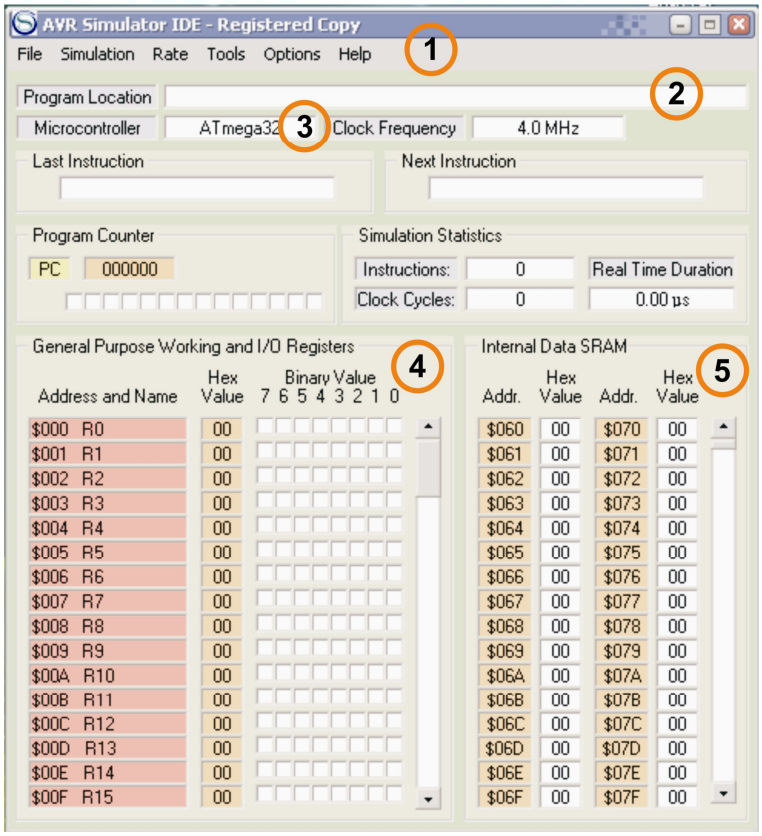

Рис. 1. Основне вікно програми AVR Simulator IDE

У верхній частині знаходяться меню, через які можна отримати доступ до основних і додаткових модулів програми (поз. 1)(рис. 1).

В рядку Program Location вказано шлях до обраної програми і її ім'я (поз. 2).

В рядку Microcontrollers, відображається тип обраного мікроконтролера (поз. 3).

У нижній частині вікна є дві панелі (поз.4 і поз.5), де відображається стан внутрішніх регістрів мікроконтролерів AVR (регістрів загального користування та регістрів вводу/виводу), та SRAM внутрішніх даних відповідно.

Також у основному вікні відображені лічильник програм, мнемоніка останньої виконуваної інструкції, мнемоніка наступної інструкції, що буде виконуватися, цикли та інструкції лічильника і тривалість імітації в режимі реального часу.

2. Послідовність роботи з програмним симулятором наступна:

- запуск програми AVR Simulator IDE;
- вибір типу мікроконтролера, для якого написана програма;
- вибір частоти кварцового генератора (впливає тільки на відображувані програмою дані про час виконання програми або команди, але не на швидкість роботи програми, що налагоджуються в AVR Simulator IDE);
- завантаження програми у вигляді HEX-файлу або запуск вбудованого компілятора мови асемблера і написання в ньому потрібної програми;
- вибір потрібних модулів віртуальних пристроїв;
- вибір швидкості і режиму роботи програми симулятора;
- запуск процесу симуляції роботи програми на обраному МК.

Якщо потрібно скористатися для роботи з симулятором власною програмою або внести зміни у вже розроблену, необхідно створити або завантажити для цього файл асемблера, з якого після компіляції буде створений необхідний для роботи з симулятором hex-файл.

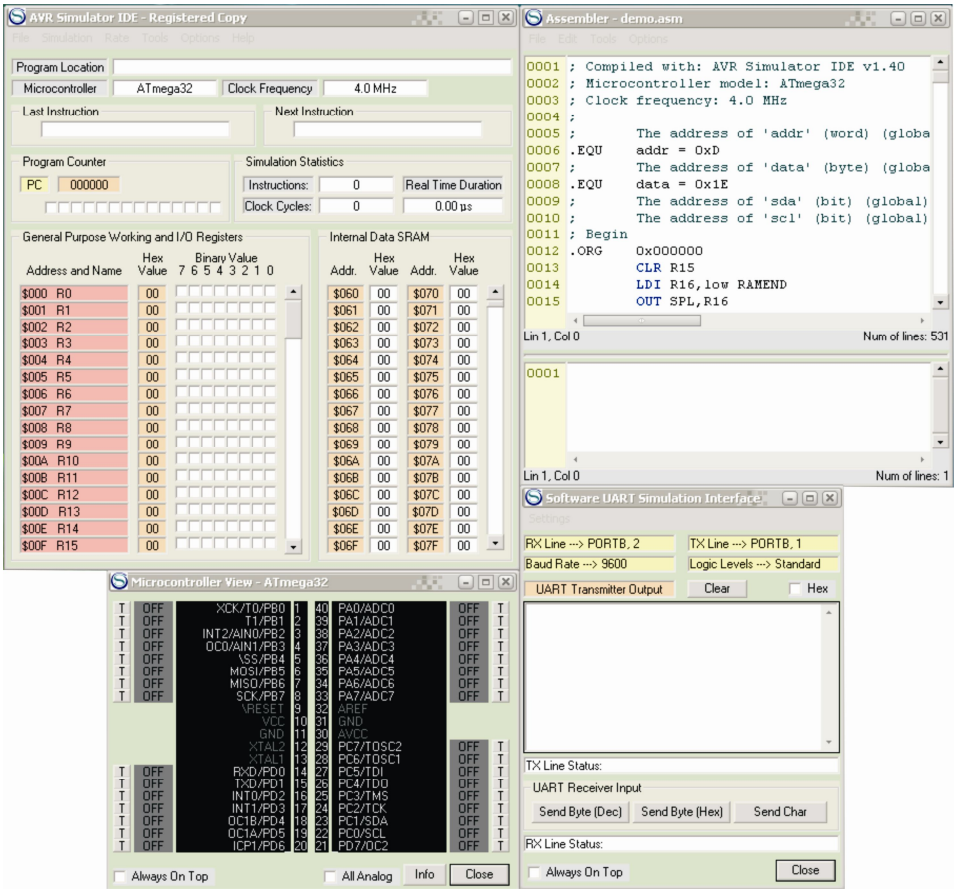

Рис. 2 Вікно симулятора з полем компілятора Assembler, апаратними виводами контролера, полем послідовного інтерфейсу

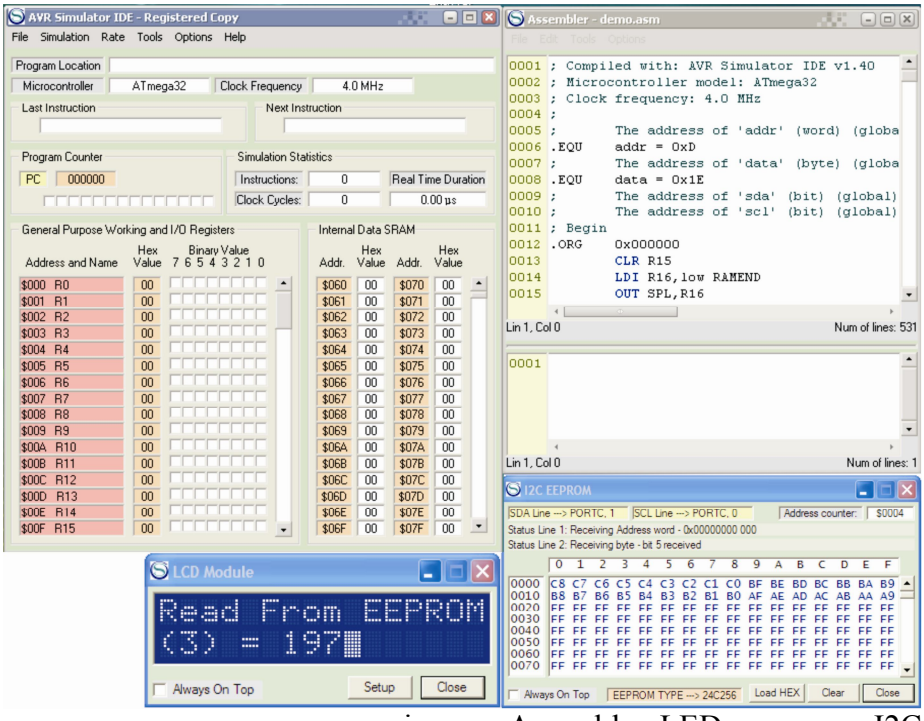

Рис. 3 Вигляд симулятора з полем компілятора Assembler, LED- модулем, I2C EEPROM

Для цього:

1. Натиснути Options | Assembler. Відкриється вікно компілятора Assembler – UNTITLED (рис. 2);

2. У вікні Assembler натиснути опцію File. Розкриється закладка, з якої для створення нового файлу потрібно натиснути New, а для завантаження вже створеного – OPEN.

3. Після вибору і завантаження файлу (з розширенням .asm), його текст з'явиться у вікні Assembler .

4. Для компіляції створеного або завантаженого і потім зміненого файлу, натисніть Tools і у вікні, що розкриється – Assemble. В нижній половині вікна Assembler з'явиться лістинг відкомпільованого файлу і, одночасно, при відсутності помилок, буде створений одноіменний hex-файл.

3. **Завдання на лабораторну роботу:** ввід-вивід даних через порти AVR мікроконтролера.

1. Вивчити програмну модель AVR Simulator IDE.

2. Вивчити команди програмування та обміну даними через порти AVR мікроконтролера.

Завдання 1 та 2

1. Дослідити роботу програм з Прикладу 1 та Прикладу 2 в режимі роботи стимулятора «Normal" та вміст регістрів контролера, які використовуються при виконанні цієї програми, в покроковому режимі роботи.

2. Записати для вибраних команд асемблера коментар щодо їх призначення (див. Приклад 1).

Приклад 1

Виконати програму, що забезпечує ввід даних з чотирьох молодших розрядів порту B, зсув їх на чотири розряди вліво і вивід через чотири старші розряди цього ж порту.

Текст програми має наступний вигляд:

```
  .CSEG
  ldi R16, 0x00
  ldi R17, 0xFF
  out DDRB, R17
  out PORTB, R16          ;порт B на вихід з низьким початковим рівнем
main:
  ldi R20, 0x0F           ;завантаження в регістр числа «00001111»
  out PORTB, R20          ;вивід вмісту регістра в порт B
  swap R20                ;обмін тетрадами регістра
  out PORTB, R20          ;вивід вмісту регістра в порт B
  jmp main                ;організація циклу виводу в порт B
  nop
```

Послідовність роботи з симулятором при виконанні програми

Виконати цю програму на AVR Simulator ID, для чого необхідно:

1. Запустити AVR Simulator IDE;

2. Натиснути Options | Select Microcontroller;

3. Вибрати ATmega32 і натиснути кнопку Select;

4. Натиснути Tools і у вікні, що розкриється, вибрати «Assembler». Відкриється вікно компілятора «Assembler – UNTITLED» (рис. 4, права панель);

5. Набрати текст програми Прикладу 1 у вікні «Assembler»;

6. Натиснути Tools і у вікні, що розкриється – Assemble. В нижній половині вікна Assembler з’явиться лістинг відкомпільованого файлу (рис. 4);

7. Одночасно, при відсутності помилок, буде створений файл, для якого можна вибрати ім’я та шлях для запису. Наприклад, записати його на «Робочий стіл» комп'ютера;

8. Вибрати File | Load Program і завантажити створений файл «….hex»;

9. Натиснути Tools | 8 x LED Board. Відкриється вікно з панеллю, що містить вісім світлодіодів (рис. 4, ліва панель);

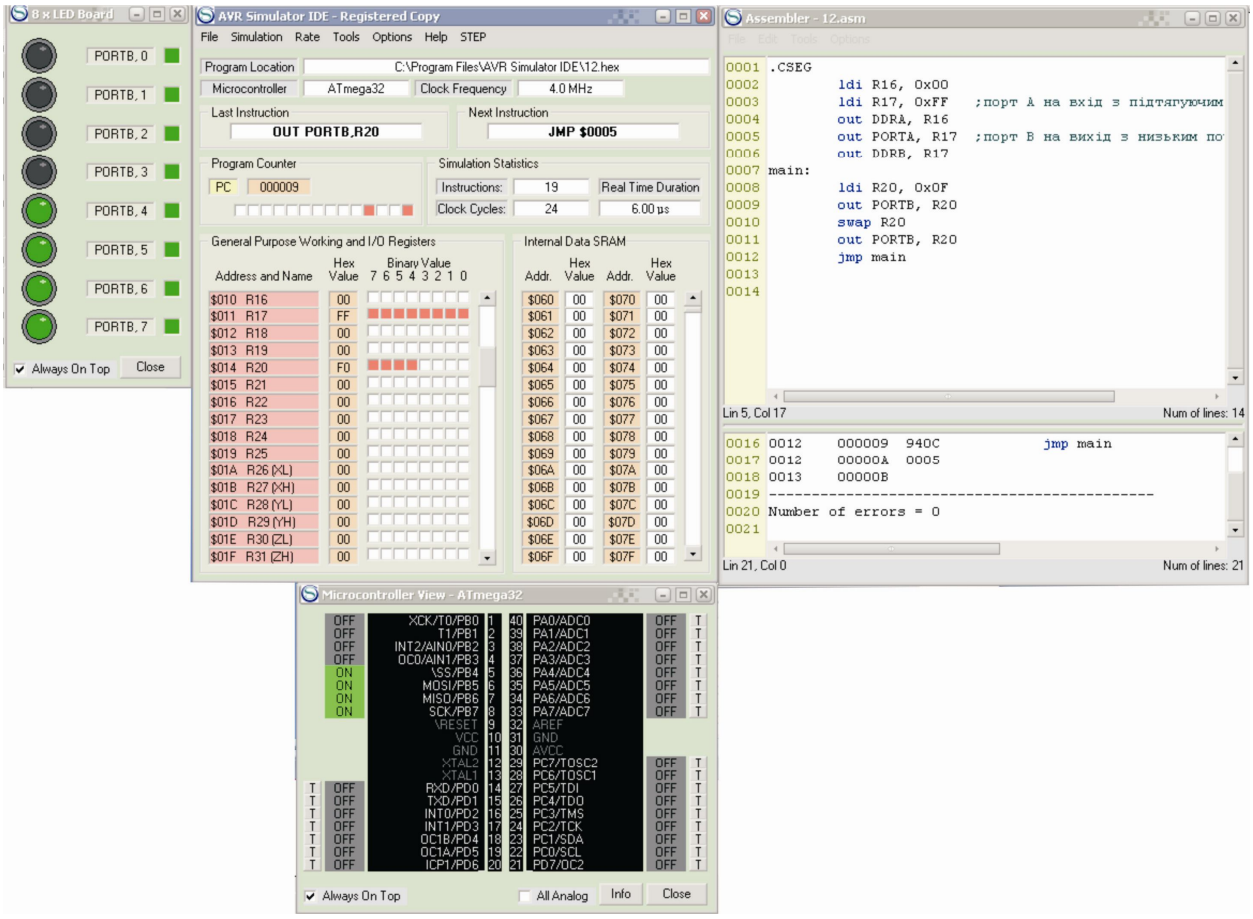

Рис. 4 Вигляд інтерфейсу симулятора з робочою програмою, панеллю «8 x LED Board» та панеллю «Microcontroller Viev»

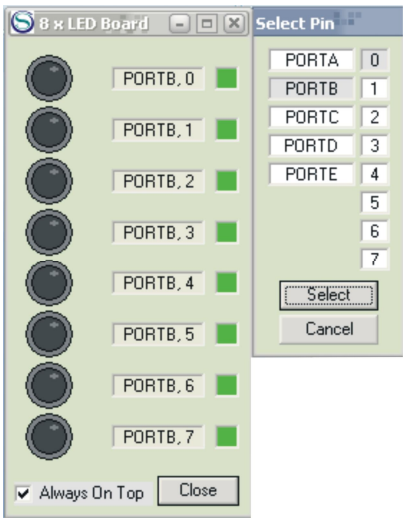

Рис. 5 Налаштування виводів порту B панелі «8 x LED Board»

10. Натиснути Tools | Microcontroller Viev. Відкриється вікно з виводами мікроконтролера (рис. 4, нижня панель);

11. У вікні «Select Pin» панелі «8 x LED Board» (рис. 5) почергово натиснути поле «PORTB» і далі «0», після чого натиснути на поле «Select», яке розташоване внизу вікна. Таким чином, вибрано порт B та його вивід 0. Це повторити для всіх ліній вибраного порту;

12. Вибрати Rate | Normal;

13. Натиснути Simulation | Start (почнеться виконання програми). Якщо при цьому курсором клацнути на одному з виводів мікросхеми (панель «Microcontroller Viev») за номером *n* (це відповідає появі на цьому виводі логічної «1» - світлодіод світиться), то засвітиться світлодіод на виводі мікросхеми з номером *n+4*;

14. Щоб зупинити виконання програми, потрібно натиснути Simulation | Stop.

Для того, щоб мати змогу контролювати вміст регістрів після виконання стимулятором кожної команди, перейти на виконання програми в кроковому режимі роботи.

Для цього:

1. В основному вікні симулятора натиснути Rate | Step By Step, а далі вибрати опцію Simulation і натиснути Start. Симулятор готовий до виконання програми в кроковому режимі;

2. Для виконання наступної команди програми потрібно натиснути на закладку STEP, яка з'явиться справа від закладки HELP вгорі основного вікна симулятора після вибору крокового режиму його роботи.

Вміст регістрів контролера, які використовуються при виконанні команд програми, знайти в області регістрів Adress and Name, яка розташована в лівій нижній частині основного вікна симулятора (виділені рожевим кольором). Всі регістри восьмирозрядні.

В процесі виконання програми по зміні кольору комірок видно, вміст яких регістрів змінюється. Забарвлення комірки відповідного розряду регістру помаранчевим кольором означає наявність "1", білим - "0".

Приклад 2

Виконати програму, що забезпечує ввід даних з чотирьох молодших розрядів порту A, зсув їх на чотири розряди вліво і вивід через чотири старші розряди цього ж порту.

Текст програми має наступний вигляд:

```
  .CSEG
  ldi R16, 0x00
  ldi R17, 0xFF              ;
  out DDRA, R16
  out PORTA, R17             порт A на вхід з підтягуючим резистором
  out DDRB, R17
  out PORTB, R16             ;порт B на вихід з низьким початковим рівнем
main:
  in R20, PINA               ;завантаження в регістр коду з ліній порту A
  swap R20                   ;обмін тетрадами регістра
  out PORTB, R20             ;вивід вмісту регістра в порт B
  jmp main                   ;організація циклу виводу в порт B
  nop
```

Завдання 3

1. Виконати програму (Приклад 2) в режимі «Normal». Перед виконанням програми клацнути курсором на одному чи декількох виводах з чотирьох молодших розрядів порту A мікросхеми (панель «Microcontroller Viev») за номером *n=0...3* (це відповідає появі на цих виводах логічної «1» - вивід забарвиться в зелений колір). При виконанні програми, на

старших чотирьох розрядах порту В з’явиться код, що відповідає вхідному на лініях порту А. Одночасно засвітяться світлодіоди на виводах порту В мікросхеми з номерами *n+4* (рис. 4);

2. Виконати програму в кроковому режимі виконання програми. Вміст тих регістрів, значення яких змінюється в процесі виконання команд програми, записати в шістнадцятковому коді в табл.1.

Таблиця 1

Регістр	PC	R16	R17	R20	R21	DDRB	PORTB	SREG	DDRA	PORT A
Команда 1										
Команда 2										
……..										
Команда n										

3. З програми Прикладу 1 вибрати десять команд і за таблицею команд асемблера для AVR мікроконтролера (табл. 1) записати коментар щодо призначення цих команд (див. Приклад 2, де наведено такий запис для однієї команди).

Приклад 2

Команда	**Виконувана операція (коментар)**
out PORTA, R17	;порт В на вихід з низьким початковим рівнем

і т.д.

Завдання 4

1. Скласти програму, яка забезпечує ввід даних з **чотирьох старших розрядів порту В**, зсув їх на чотири розряди вправо і вивід через **чотири молодші розряди цього ж порту**.
2. Виконати програму в режимі «Normal».
3. В процесі виконання програми клацнути курсором на одному з виводів чотирьох старших розрядів порту В мікросхеми (панель «Microcontroller Viev») за номером *n=4…7*. При цьому повинен засвітитися світлодіод на виводі мікросхеми з номером *n-4* (рис. 4);
4. Виконати програму в кроковому режимі виконання програми.
5. Вміст тих регістрів, значення яких змінюється в процесі виконання команд програми, записати в шістнадцятковому коді в табл. 1.
6. З виконуваної програми вибрати десять команд і за таблицею команд асемблера для AVR-мікроконтролера (Додаток 1) записати коментар щодо призначення цих команд (див. Приклад 2, де наведено такий запис для однієї команди).

Завдання 5

1. Скласти програму, яка забезпечує ввід даних з **чотирьох старших розрядів порту А**, та вивід через **чотири молодші розряди порту В**.
2. Виконати програму в режимі «Normal».
4. В процесі виконання програми клацнути курсором на виводах чотирьох старших розрядів порту А мікросхеми (панель «Microcontroller Viev ») за номером *n=4…7*. (це відповідає появі на цих виводах логічної «1» - вивід забарвиться в зелений колір). При виконанні програми на молодших чотирьох розрядах порту В з’явиться код, що відповідає вхідному на лініях порту А. Одночасно засвітяться світлодіоди на виводах порту В мікросхеми з номерами *n-4* (рис. 4);
3. Виконати програму в кроковому режимі виконання програми.
4. Вміст тих регістрів, значення яких змінюється в процесі виконання команд програми, записати в шістнадцятковому коді в табл. 1.

5. З виконуваної програми вибрати десять команд і за таблицею команд асемблера для AVR- мікроконтролера (Додаток 1) записати коментар щодо призначення цих команд (див. Приклад 2, де наведено такий запис для однієї команди).

4.**Контрольні запитання**

1. Використання AVR-мікроконтролерів.
2. Програмування портів мікроконтролера.
3. Організація циклів в роботі мікроконтролера.
4. Формат та використання регістрів загального призначення.
5. Призначення та позначення основних елементів програмної моделі мікроконтролера.

5. **Література**

1. Програмування мікроконтролерів систем автоматики: конспект лекцій для студентів базового напряму 050201 “Системна інженерія” / Укл.: А.Г. Павельчак, В.В. Самотий, Ю.В. Яцук – Львів: Львівська політехніка. – 2012. – 143 с.

2. Евстифеев А. В. Микроконтроллеры AVR семейств Tiny и Mega фирмы ATMEL, – [5-е изд., стер.] / Евстифеев А. В. – М.: Издательский дом «Додэка-XXI», 2008. 560 с.

Додаток 1
Система команд мікроконтролерів AVR

Система команд AVR мікроконтролерів включає команди арифметичних і логічних операцій, команди передачі даних, команди, що керують послідовністю виконання програми і команди операцій з бітами.

Для зручності написання й аналізу програм всім операціям із системи команд крім двійкового коду зіставлені мнемокоди Ассемблера (символічні позначення операцій), що використовуються при створенні вихідного тексту програми.

Спеціальні програми-транслятори переводять потім символічні позначення в двійкові коди.

Спеціальна директива ассемблера **.device** забезпечує контроль відповідності команд, використовуваних у тексті програми, типу зазначеного процесора.

Під час виконання арифметичних, логічних чи операцій роботи з бітами ALU формує ознаки результату операції, тобто встановлює чи скидає біти в регістрі стану **SREG** (Status Register).

Регістр статусу - SREG - розміщений у просторі I/O за адресою $3F ($5F).

Таблиця 1 – Регістр статусу - SREG

Біти	7	6	5	4	3	2	1	0	
$3F ($5F)	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**	**REG**
Читання/Запис	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початковий стан	0	0	0	0	0	0	0	0	

Bit 7 - I: Global Interrupt Enable - Дозвіл глобального переривання. Біт дозволу глобального переривання для дозволу переривання повинний бути встановлений у стан 1. Керування дозволом конкретного переривання виконується регістрами маски переривання GIMSK і TIMSK. Якщо біт глобального переривання очищений (у стані 0), то жодне з дозволів конкретних переривань, встановлених у регістрах GIMSK і TIMSK, не діє.

Біт I апаратно очищається після переривання і встановлюється для наступного дозволу глобального переривання командою RETI.

Bit 6 - T: Bit Copy Storage - Біт збереження копії. Команди копіювання біта BLD (Bit Load) і BST (Bit STore) використовують біт T, як біт джерело і біт призначення при операціях з бітами. Командою BST біт регістра копіюється до біту T, командою BLD біт T копіюється до регістру.

Bit 5 - H: Half Carry Flag - Прапор напівпереносу. Прапор напівпереносу вказує на напівперенос у ряді арифметичних операцій.

Bit 4 - S: Sign Bit, S = N V - Біт знаку. Біт S завжди знаходиться в стані, обумовленому логічною функцію АБО (OR) між прапором негативного значення N і доповненням до двох прапора переповнення V.

Bit 3 - V: Two's Complement Overflow Flag. Доповнення до двох прапора переповнення. Доповнення до двох прапора V підтримує арифметику доповнення до двох.

Bit 2 - N: Negative Flag – Прапор негативного значення. Прапор негативного значення N вказує на негативний результат ряду арифметичних і логічних операцій.

Bit 1 - Z: Zero Flag – Прапор нульового значення. Прапор нульового значення Z вказує на нульовий результат ряду арифметичних і логічних операцій.

Bit 0 - C: Carry Flag – Прапор переносу. Ознаки результату операції можуть бути використані в програмі для виконання подальших арифметично-логічних операцій чи команд умовних переходів.

Арифметичні і логічні конструкції

Мнемоніка	Операнди	Опис	Операція	Прапори	Цикли
ADD	Rd,Rr	Підсумовування без переносу	Rd = Rd + Rr	Z,C,N,V,H,S	1
ADC	Rd,Rr	Підсумовування з переносом	Rd = Rd + Rr + C	Z,C,N,V,H,S	1
SUB	Rd,Rr	Вирахування без переносу	Rd = Rd - Rr	Z,C,N,V,H,S	1
SUBI	Rd,K8	Вирахування константи	Rd = Rd - K8	Z,C,N,V,H,S	1
SBC	Rd,Rr	Вирахування з переносом	Rd = Rd - Rr - C	Z,C,N,V,H,S	1
SBCI	Rd,K8	Вирахування константи з переносом	Rd = Rd - K8 - C	Z,C,N,V,H,S	1
AND	Rd,Rr	Логічне И	Rd = Rd · Rr	Z,N,V,S	1
ANDI	Rd,K8	Логічне И с константою	Rd = Rd · K8	Z,N,V,S	1
OR	Rd,Rr	Логічне АБО	Rd = Rd V Rr	Z,N,V,S	1
ORI	Rd,K8	Логічне АБО з константою	Rd = Rd V K8	Z,N,V,S	1
EOR	Rd,Rr	Логічне що виключає АБО	Rd = Rd EOR Rr	Z,N,V,S	1
COM	Rd	Побітна Інверсія	Rd = $FF - Rd	Z,C,N,V,S	1
NEG	Rd	Зміна знака (Доп. код)	Rd = $00 - Rd	Z,C,N,V,H,S	1
SBR	Rd,K8	Установити біт (біти) у регістрі	Rd = Rd V K8	Z,C,N,V,S	1
CBR	Rd,K8	Скинути біт (біти) у регістрі	Rd = Rd · ($FF - K8)	Z,C,N,V,S	1
INC	Rd	Інкрементувати значення регістра	Rd = Rd + 1	Z,N,V,S	1
DEC	Rd	Декрементувати значення регістра	Rd = Rd -1	Z,N,V,S	1
TST	Rd	Перевірка на нуль або заперечність	Rd = Rd · Rd	Z,C,N,V,S	1
CLR	Rd	Очистити регістр	Rd = 0	Z,C,N,V,S	1
SER	Rd	Установити регістр	Rd = $FF	None	1
ADIW	Rdl,K6	Скласти константу і слово	Rdh:Rdl=Rdh:Rdl+ K6	Z,C,N,V,S	2
SBIW	Rdl,K6	Вичитати константу зі слова	Rdh:Rdl=Rdh:Rdl - K 6	Z,C,N,V,S	2

Інструкції розгалуження

Мнемоніка	Операнди	Опис	Операція	Прапори	Цикли
RJMP	k	Відносний перехід	PC = PC + k +1	None	2
IJMP	Немає	Непрямий перехід на (Z)	PC = Z	None	2
EIJMP	Немає	Розширений непрямий перехід на (Z)	STACK = PC+1, PC(15:0) = Z, PC(21:16) = EIND	None	2
JMP	k	Перехід	PC = k	None	3
RCALL	k	Відносний виклик підпрограми	STACK=PC+1, PC=PC + k+ 1	None	3/4*
ICALL	Немає	Непрямий виклик (Z)	STACK = PC+1, PC = Z	None	3/4*
EICALL	Немає	Розширений непрямий виклик (Z)	STACK = PC+1, PC(15:0) = Z, PC(21:16) =EIND	None	4*
RET	Немає	Повернення з підпрограми	PC = STACK	None	4/5*
RETI	Немає	Повернення з переривання	PC = STACK	I	4/5*
CPSE	Rd,Rr	Порівняти, пропустити якщо рівні	if (Rd ==Rr) PC = PC 2 or 3	None	1/2/3
CP	Rd,Rr	Порівняти	Rd -Rr	Z,C,N,V,H,S	1
CPC	Rd,Rr	Порівняти з переносом	Rd - Rr - C	Z,C,N,V,H,S	1
CPI	Rd,K8	Порівняти з константою	Rd - K	Z,C,N,V,H,S	1
SBRC	Rr,b	Пропустити якщо біт у регістрі очищений	if(Rr(b)==0) PC = PC + 2 or 3	None	1/2/3

SBRS	Rr,b	Пропустити якщо біт у регістрі встановлений	if(Rr(b)==1) PC = PC + 2 or 3	None	1/2/3
SBIC	P,b	Пропустити якщо біт у порту очищений	if(I/O(P,b)==0) PC=PC + 2 or 3	None	1/2/3
SBIS	P,b	Пропустити якщо біт у порту встановлений	if(I/O(P,b)==1) PC=PC + 2 or 3	None	1/2/3
BRBC	s,k	Перейти якщо прапор у SREG очищений	if(SREG(s)==0) PC=PC+ k + 1	None	1/2
BRBS	s,k	Перейти якщо прапор у SREG установлений	if(SREG(s)==1) PC = PC+k+ 1	None	1/2
BREQ	k	Перейти якщо дорівнює	if(Z==1) PC = PC + k + 1	None	1/2
BRNE	k	Перейти якщо не дорівнює	if(Z==0) PC = PC + k + 1	None	1/2
BRCS	k	Перейти якщо перенос установлений	if(C==1) PC = PC + k + 1	None	1/2
BRCC	k	Перейти якщо перенос очищений	if(C==0) PC = PC + k + 1	None	1/2
BRSH	k	Перейти якщо дорівнює чи більше	if(C==0) PC = PC + k + 1	None	1/2
BRLO	k	Перейти якщо менше	if(C==1) PC = PC + k + 1	None	1/2
BRMI	k	Перейти якщо мінус	if(N==1) PC = PC + k + 1	None	1/2
BRPL	k	Перейти якщо плюс	if(N==0) PC = PC + k + 1	None	1/2
BRGE	k	Перейти якщо більше чи дорівнює (зі знаком)	if(S==0) PC = PC + k + 1	None	1/2
BRLT	k	Перейти якщо менше (зі знаком)	if(S==1) PC = PC + k + 1	None	1/2
BRHS	k	Перейти якщо прапор внутрішнього переносу встановлений	if(H==1) PC = PC + k + 1	None	1/2
BRHC	k	Перейти якщо прапор внутрішнього переносу очищений	if(H==0) PC = PC + k + 1	None	1/2
BRTS	k	Перейти якщо прапор T встановлений	if(T==1) PC = PC + k + 1	None	1/2
BRTC	k	Перейти якщо прапор T очищений	if(T==0) PC = PC + k + 1	None	1/2
BRVS	k	Перейти якщо прапор переповнення встановлений	if(V==1) PC = PC + k + 1	None	1/2
BRVC	k	Перейти якщо прапор переповнення очищений	if(V==0) PC = PC + k + 1	None	1/2
BRIE	k	Перейти якщо переривання дозволені	if(I==1) PC = PC + k + 1	None	1/2
BRID	k	Перейти якщо переривання заборонені	if(I==0) PC = PC + k + 1	None	1/2

Виконувати арифметико-логічні операції й операції читання безпосередньо над змістом комірок пам'яті не можна. Не можна також записати константу чи очистити вміст комірки пам'яті.

Система команд AVR дозволяє лише виконувати операції обміну даними між осередками SRAM і регістрами загального призначення.

Перевагами системи команд можна вважати різноманітні режими адресації комірок пам'яті.

Усі регістри введення/виведення можуть зчитуватися і записуватися через регістри загального призначення за допомогою команд IN, OUT.

Безпосередня установка і скидання окремих розрядів цих регістрів виконується командами SBI і CBI. Команди умовних переходів у якості своїх операндів можуть мати як біти-ознаки результату операції, так і окремі розряди регістрів введення/виведення, що побітно адресуються.

Інструкції передачі даних

Мнемоніка	Операнди	Опис	Операція	Прапори	Цикли
MOV	Rd,Rr	Скопіювати регістр	Rd = Rr	None	1
LDI	Rd,K8	Завантажити константу	Rd = K	None	1
LDS	Rd,k	Пряме завантаження	Rd = (k)	None	2*
LD	Rd,X	Непряме завантаження	Rd = (X)	None	2*
LD	Rd,X+	Непряме завантаження з пост-інкрементом	Rd=(X), X=X+1	None	2*
LD	Rd,-X	Непряме завантаження з пре-декрементом	X=X-1, Rd=(X)	None	2*
LD	Rd,Y	Непряме завантаження	Rd = (Y)	None	2*
LD	Rd,Y+	Непряме завантаження з пост-інкрементом	Rd=(Y),Y=Y+1	None	2*
LD	Rd,-Y	Непряме завантаження з пре-декрементом	Y=Y-1,Rd= (Y)	None	2*
LDD	Rd,Y+q	Непряме завантаження з заміщенням	Rd = (Y+q)	None	2*
LD	Rd,Z	Непряме завантаження	Rd = (Z)	None	2*
LD	Rd,Z+	Непряме завантаження з пост-інкрементом	Rd= (Z), Z=Z+1	None	2*
LD	Rd,-Z	Непряме завантаження з пре-декрементом	Z=Z-1, Rd = (Z)	None	2*
LDD	Rd,Z+q	Непряме завантаження з заміщенням	Rd = (Z+q)	None	2*
STS	k,Rr	Пряме збереження	(k) = Rr	None	2*
ST	X,Rr	Непряме збереження	(X) = Rr	None	2*
ST	X+,Rr	Непряме збереження з пост-інкрементом	(X)=Rr, X=X+1	None	2*
ST	-X,Rr	Непряме збереження з пре-декрементом	X=X-1, (X)=Rr	None	2*
ST	Y,Rr	Непряме збереження	(Y) = Rr	None	2*
ST	Y+,Rr	Непряме збереження з пост-інкрементом	(Y)=Rr, Y=Y+1	None	2
ST	-Y,Rr	Непряме збереження з пре-декрементом	Y=Y-1, (Y)= Rr	None	2
ST	Y+q,Rr	Непряме збереження з заміщенням	(Y+q) = Rr	None	2
ST	Z,Rr	Непряме збереження	(Z) = Rr	None	2
ST	Z+,Rr	Непряме збереження з пост-інкрементом	(Z)= Rr, Z=Z+1	None	2
ST	-Z,Rr	Непряме збереження з пре-декрементом	Z=Z-1, (Z) = Rr	None	2
ST	Z+q,Rr	Непряме збереження з заміщенням	(Z+q) = Rr	None	2
LPM	Нет	Завантаження з програмної пам'яті	R0 = (Z)	None	3
LPM	Rd,Z	Завантаження з програмної пам'яті	Rd = (Z)	None	3
LPM	Rd,Z+	Завантаження з програмної пам'яті з пост-інкрементом	Rd=(Z), Z=Z+1	None	3
SPM	Нет	Збереження в програмній пам'яті	(Z) = R1:R0	None	-
IN	Rd,P	Читання порту	Rd = P	None	1
OUT	P,Rr	Запис у порт	P = Rr	None	1
PUSH	Rr	Занесення регістра в стек	STACK = Rr	None	2
POP	Rd	Витяг регістра зі стека	Rd = STACK	None	2

Інструкції роботи з бітами

Мнемоніка	Операнди	Опис	Операція	Прапори	Цикли
LSL	Rd	Логічний зсув вліво	Rd(n+1)=Rd(n),Rd(0)=0,C=Rd(7)	Z,C,N,V,H,S	1
LSR	Rd	Логічне зрушення вправо	Rd(n)=Rd(n+1), Rd(7)=0, C=Rd(0)	Z,C,N,V,S	1
ROL	Rd	Циклічне зрушення вліво через C	Rd(0)=C, Rd(n+1)=Rd(n), C=Rd(7)	Z,C,N,V,H,S	1
ROR	Rd	Циклічне зрушення вправо через C	Rd(7)=C, Rd(n)=Rd(n+1), C=Rd(0)	Z,C,N,V,S	1
ASR	Rd	Арифметичне зрушення вправо	Rd(n)=Rd(n+1), n=0,...,6	Z,C,N,V,S	1
SWAP	Rd	Перестановка тетрад	Rd(3..0)=Rd(7..4),Rd(7..4)=Rd(3..0)	None	1
BSET	s	Установка прапора	SREG(s) = 1	SREG(s)	1
BCLR	s	Очищення прапора	SREG(s) = 0	SREG(s)	1
SBI	P,b	Установити біт у порту	I/O(P,b) = 1	None	2
CBI	P,b	Очистити біт у порту	I/O(P,b) = 0	None	2
BST	Rr,b	Зберегти біт з регістра в T	T = Rr(b)	T	1
BLD	Rd,b	Завантажити біт з T у регістр	Rd(b) = T	None	1
SEC	Hi	Установити прапор переносу	C =1	C	1
CLC	Hi	Очистити прапор переносу	C = 0	C	1
SEN	Hi	Установити прапор негативного числа	N = 1	N	1
CLN	Hi	Очистити прапор негативного числа	N = 0	N	1
SEZ	Hi	Встановити прапор нуля	Z = 1	Z	1
CLZ	Hi	Очистити прапор нуля	Z = 0	Z	1
SEI	Hi	Встановити прапор переривань	I = 1	I	1
CLI	Hi	Очистити прапор переривань	I = 0	I	1
SES	Hi	Установити прапор числа зі знаком	S = 1	S	1
CLN	Hi	Очистити прапор числа зі знаком	S = 0	S	1
SEV	Hi	Установити прапор переповнення	V = 1	V	1
CLV	Hi	Очистити прапор переповнення	V = 0	V	1
SET	Hi	Установити прапор T	T = 1	T	1
CLT	Hi	Очистити прапор T	T = 0	T	1
SEH	Hi	Установити прапор внутрішнього переносу	H = 1	H	1
CLH	Hi	Очистити прапор внутрішнього переносу	H = 0	H	1
NOP	Hi	Немає операції	Hi	None	1
SLEEP	Hi	Спати (зменшити енергоспоживання)	Дивитися опис інструкції	None	1
WDR	Hi	Скидання сторожового таймера	Дивитися опис інструкції	None	1

Асемблер не розрізняє регістр символів. Операнди можуть бути таких видів:

- Rd: результуючий і вихідний регістр;
- Rr: вихідний регістр;

- b: константа (3 біти), може бути константний вираз;
- s: константа (3 біти), може бути константний вираз;
- P: константа (5-6 біт), може бути константний вираз;
- K6: константа (6 біт), може бути константний вираз;
- K8: константа (8 біт), може бути константний вираз;
- k: константа, може бути константний вираз;
- q: константа (6 біт), може бути константний вираз;
- Rdl: R24, R26, R28, R30 для інструкцій ADIW і SBIW;
- X,Y,Z: регістри непрямої адресації (X=R27:R26, Y=R29:R28, Z=R31:R30).