Лекція 5

Характеристика команд AVR-мікроконтролерів

1 Загальні відомості

Мнемоніка команди та мнемокод

Для написання програм мовою «Асемблер» використовують мнемоніки команд та мнемокоди. Мнемоніки введені для полегшення написання програм, і складають основу мови *Асемблера.*

Мнемоніка команди - це представлення коду операції у вигляді сполучення латинських літер, що мають визначений зміст (використовуються англійські слова або скорочення, наприклад, MOV, PUSH, POP, JMP, CLR, NOP і т. інД

Мнемокод включає в себе мнемоніку команди та символічний опис операндів - даних, які беруть участь в операції.

Код операції команди

Код операції команди (КОП) - це комбінація бітів (не більше восьми для 8-розрядних МК), що знаходяться на початку машинного коду команди і визначають тип операції, що підлягає виконанню у конкретний момент часу. КОП читається з пам’яті і розміщується в програмно недоступний регістр команд мікроконтролера (РК). Потім він декодується і визначається тип команди, яка повинна бути виконана. Крім КОП в машинний код команди можуть входити адреси та операнди.

Машинний код команди

Машинний код команди представляє собою двійковий код команди, який складається з одного або декількох байтів в залежності від типу команд конкретного МК.

Операнди

Операндами у мікропроцесорній техніці називають дані, які приймають участь у виконанні тієї чи іншої команди (операції). В залежності від способу адресації операндів, дані можуть знаходитися у регістрах та пам'яті. Існують одноопераднідні або двоопераднідні команди.

2 Формати команд

Більшість мікроконтролерів сімейства AVR мають 14 форматів (типів) базового набору команд, наведених на рисунку 1.22. На рисунку використано наступні скорочення: КОП – код операції; K – константа даних; k – адресна константа; b – номер біта в регістрі введення/виведення або регістрового файлу (РЗП) – 3 біти; s – номер біта в регістрі стану – 3 біти; A – адреса регістра в основному просторі введення/виведення; q – задає зміщення для непрямої адресації – 6 біт; d(r) – регістр-приймач (джерело) з області регістрового файлу.

До першого типу належать команди, код операції яких займає всю довжину команди – 16 біт. До цієї групи належить, наприклад, команда LPM – завантаження регістра загального призначення R0 з пам'яті програм за адресою, яка міститься у регістровій парі Z = R31:R30. Такий формат також мають команди IJMP, ICALL, RET, RETI, NOP, SLEEP та WDT.

До другого типу належать команди, які окрім коду операції містять п'ятирозрядну адресу одного з регістрів загального призначення. Наприклад, це команди DEC Rd; LD Rd, -X; ST Y, Rr і т. ін.

Третій тип мають команди, в яких адресуються два операнди – регістри загального призначення: Rd – приймач, Rr – джерело. Це, наприклад, команди ADD Rd, Rr; CPSE Rd, Rr; AND Rd, Rr і т. ін.

До четвертого типу належать двоопераднідні команди, в яких один операнд: К6 = 6 біт (К = 0...63) входить в саму команду – безпосередня адресація, а другим є пара регістрів: R24, R25; R26, R27; R28, R29; R30, R31, які кодуються двома бітами команди – dd: 00 – R24, R25; 01 – R26, R27; 10 – R28, R29; 11 – R30, R31.

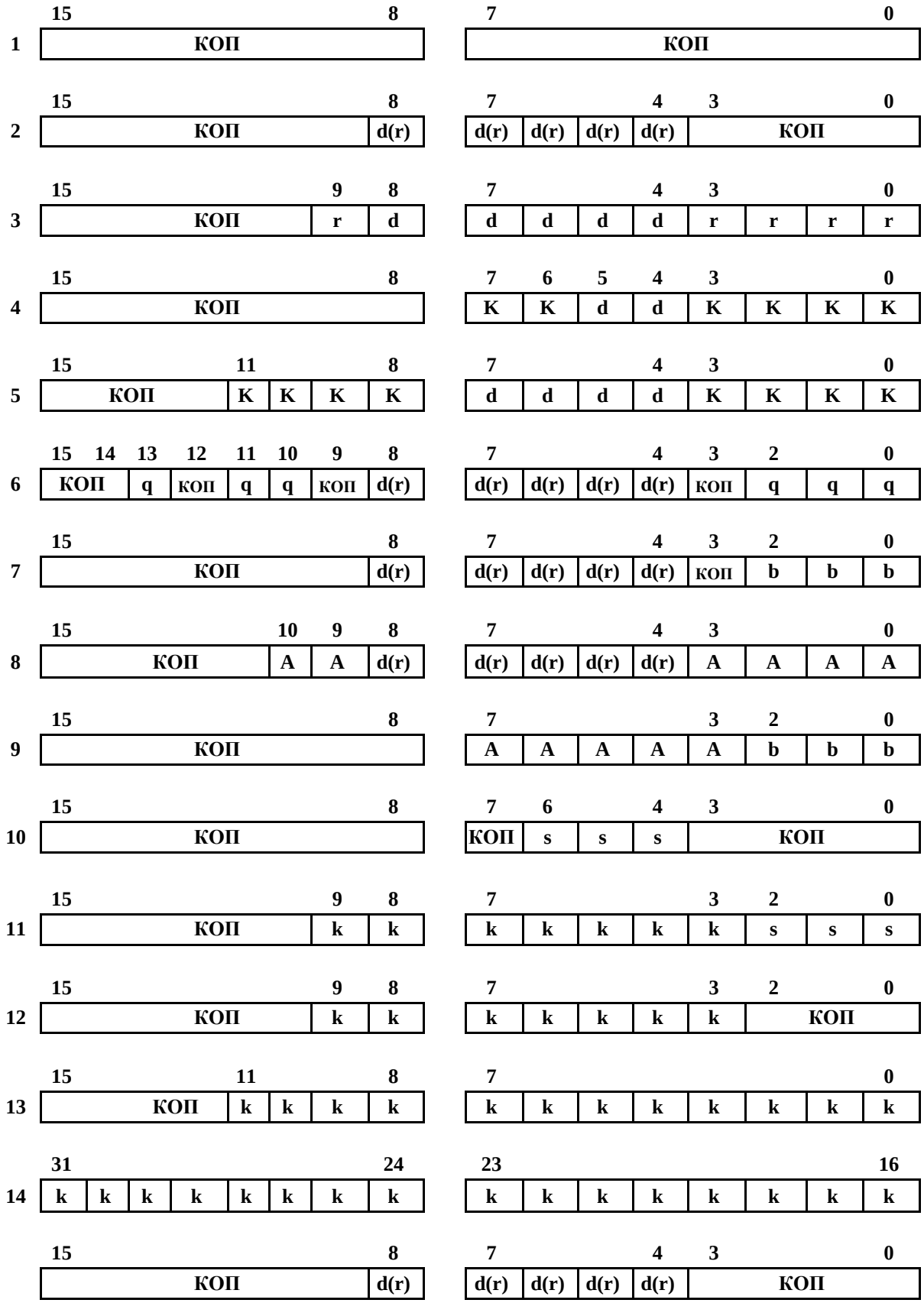

Рисунок 1.22 – Формати базового набору команд

Наприклад, команди ADIW Rdl, K6; SBIW Rdl, K6, де Rdl = R24/R26/R28/R30.

П'ятий тип мають двоопераднi команди, наприклад, SBCI Rd*, K; ORI Rd*, K, в яких Rd* – один із шістнадцяти РЗП (R16...R31), а другий К – восьмибітна константа (безпосередній операнд): $0 \le K \le 255$.

До шостого типу належать команди, в яких один з двох операндів може бути РЗП (Rd – приймач, Rr – джерело), а другий міститься у СПД і адресується за допомогою непрямої відносної адресації, коли до вмісту індексного регістра Y або Z додається зміщення q ($0 \le q \le 63$). Це, наприклад, команди LDD Rd, z+q; STD Y+q, Rr і т. ін.

Сьомий тип мають команди, в яких адресується один із восьми бітів РЗП. Наприклад, команди BLD Rd, b; BST Rr, b; SBRC Rr, b і т. ін., де Rd – приймач, Rr – джерело (один з 32-х РЗП), а b = 0...7 – номер біта РЗП.

До восьмого типу належать команди, в яких одним операндом є РЗП (Rd – приймач/Rr – джерело), а другий міститься в одному з 64-х РВВ основного простору. Наприклад, команди IN Rd, P; OUT P, Rr, де P – адреса РВВ (P = 0…63).

Дев'ятий тип мають команди, в яких адресуються окремі біти молодшої половини РВВ (P* = 0...31). Наприклад, команди SBI P*, b, де b = 0...7 – номер біта РВВ.

До десятого типу належать команди, в яких адресується один із 8-ми бітів регістра прапорців SREG. Наприклад, команди BSET s; BCLR s, де s = 0...7 – номер біта регістра стану.

Одинадцятий тип мають команди умовного переходу в залежності від значення вказаних бітів регістра стану. Наприклад, команди BRBS s, k; BRBC s, k, де s = 0...7 – номер біта регістра стану SREG, а k = 7 біт ($-64 \le k \le 63$) при переході додається до адреси наступної команди.

Дванадцятий тип мають команди умовного відносного переходу в залежності від значень окремих прапорців регістра стану (прапорці адресуються неявно). Наприклад, команди BREQ k; BRCC k і т. ін., де k = 7 біт ($-64 \le k \le 63$) при переході додається до адреси наступної команди.

До тринадцятого типу належать команди відносного безумовного переходу: RJMP k та відносного безумовного виклику підпрограм: RCALL k, де k = 12 біт. Діапазон переходів та виклику підпрограм: -2048…+ 2047.

Чотирнадцятий тип мають команди: LDS Rd, k – пряме завантаження та STS k, Rr – пряме збереження, в яких одним операндом є РЗП (Rd – приймач/Rr – джерело), а другий міститься у СПД (РЗП, РВВ та СОЗП). Оскільки k = 16 біт, то кількість комірок СПД = 2^{16} = 65536.

3 Формати даних

При роботі з мікроконтролером необхідно знати не тільки формати (типи) команд (рисунок 1.22), які керують роботою мікроконтролера, але й формати (типи) даних (операндів), що беруть участь у виконанні тієї або іншої команди (операції) (рисунок 1.23).

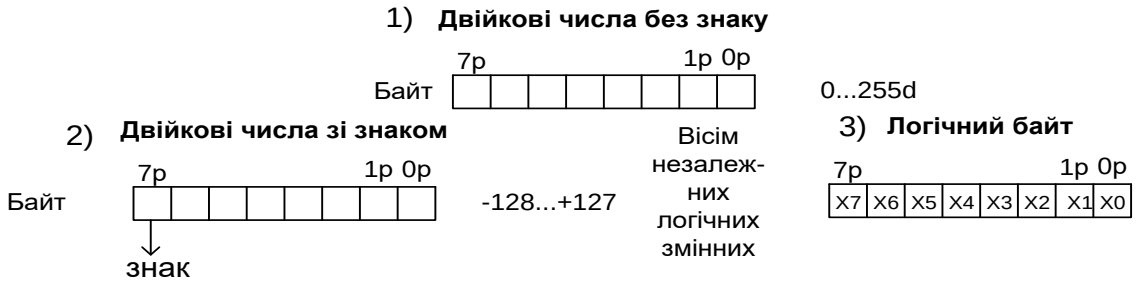

Рисунок 1.23 – Типи (формати) даних мікроконтролера

4 Довжина команд у байтах та їх розміщення у пам’яті програм

Відповідно до рисунку 1.22 більшість команд мікроконтролерів сімейства AVR мають довжину – одне слово (2 байти), за виключенням команди чотирнадцятого типу, яка має довжину 2 слова (4 байти).

Програма, яка керує роботою МК, послідовно, команда за командою, розміщується в сусідніх комірках пам’яті в порядку зростання їх адрес. Адреса команди, яка повинна виконуватись, знаходиться у програмному лічильнику PC. Пристрій керування мікроконтролера на основі прочитаного коду операції, що міститься в першому байті команди, визначає, скільки ще байтів міститься в команді, та керує їх читанням з пам’яті шляхом збільшення на одиницю адреси, яка

видається при кожному зверненні до пам’яті. Після читання з пам’яті чергової команди МК формує адресу КОП наступної команди.

5 Вплив команд на прапорці

Як відзначалося раніше, одним з основних регістрів МК-ра є регістр прапорців (ознак), який було описано у підрозділі 1.1.3. Прапорці регістра ознак встановлюються під час виконання ряду команд МК-ра. Під час опису системи команд, яка, як правило, оформлюється у вигляді таблиці, в одній з колонок показується вплив окремих команд на ті або інші прапорці (таблиця 1.3).

Як правило, команди пересилання не змінюють прапорці, окрім команд, які пересилають дані у регістр прапорців. Частіше прапорці змінюють арифметичні, логічні команди і команди порівняння.

Окремі прапорці (ознаки) аналізуються під час виконання команд умовних переходів, виклику і повернення з підпрограм, що дозволяє передавати керування в програмі в залежності від поточного значення прапорців.

6 Час виконання команд

В AVR-мікроконтролерах окремі команди виконуються за 1, 2, 3 або 4 такти. Тривалість одного такту дорівнює одному періоду тактової частоти f_{BQ}.

7 Базовий набір команд мікроконтролера

Загальні відомості

AVR-мікроконтролери мають RISC-архітектуру, основною перевагою якої є збільшення швидкодії за рахунок скорочення кількості операцій обміну з пам’яттю програм. Практично всі команди займають одну комірку пам’яті. Виняток становлять команди, в яких одним з операндів є 16-розрядна адреса СПД (РЗП, РВВ та СОЗП).

Підвищення швидкодії досягнуто не за рахунок скорочення кількості команд процесора, а за рахунок збільшення розрядності комірки пам’яті програм до 16. Більшість команд виконується за один машинний цикл (1 такт).

Всі команди AVR-мікроконтролерів можна розділити на декілька груп:

– логічні операції;

– арифметичні операцій та команди зсуву;

– операції з бітами;

– команди пересилання даних;

– команди передачі керування;

– команди керування системою.

Нижче скорочено описано ці команди.

У таблиці 1.3 наведено базовий набір команд, якими можна користуватись в типовому AVR-мікроконтролері, а також основні відомості про команди, такі як мнемонічне позначення команди, її опис, тип, кількість тактів, необхідних для її виконання, а також прапорці регістра SREG, на які впливає ця команда.

В таблиці 1.3 використано наступні позначення:

- Rd – регістр-приймач результату, $0 \le d \le 31$;
- Rd* – регістр-приймач результату, $16 \le d \le 31$;
- Rdl = R24/R26/R28/R30 для команд ADIW і SBIW;
- Rr – регістр-джерело;
- P – адреса регістра введення/виведення;
- P* – адреса регістра введення/виведення, який адресується побітово (адреси $00…$1F);
- K – символьна або числова константа (8 біт);
- K6 – символьна або числова константа (6 біт);
- k – адресна константа ;
- b – номер біта в регістрі (3 біти);
- q – константа (6 біт), може бути константний вираз;
- s – номер біта в регістрі стану (3 біти);
- X, Y, Z – індексні регістри непрямої адресації (X = R27:R26, Y = R29:R28, Z = R31:R30).

Таблиця 1.3 – Базовий набір команд AVR-мікроконтролера

№	Мнемонічне позначення	Операнди	Опис	Операція	Прапорці	Тип	Кільк. тактів
АРИФМЕТИЧНІ, ЛОГІЧНІ КОМАНДИ ТА КОМАНДИ ЗСУВУ							
1	ADD	Rd, Rr	Додавання без перенесення	Rd ← Rd + Rr	Z,C,N, V,H,S	3	1
2	ADC	Rd, Rr	Додавання з перенесенням	Rd ← Rd + Rr + C	Z,C,N, V,H,S	3	1
3	ADIW	Rdl, K6	Додавання двох байт із константою	Rdh:Rdl ← Rdh:Rdl+K6	Z,C,N,V,S	4	2
4	SUB	Rd, Rr	Віднімання без перенесення	Rd ← Rd – Rr	Z,C,N, V,H,S	3	1
5	SUBI	Rd*, K	Віднімання константи	Rd* ← Rd* – K	Z,C,N, V,H,S	5	1
6	SBC	Rd, Rr	Віднімання з перенесенням	Rd ← Rd – Rr – C	Z,C,N, V,H,S	3	1
7	SBCI	Rd*, K	Віднімання константи з перенесенням	Rd* ← Rd* – K – C	Z,C,N, V,H,S	5	1
8	SBIW	Rdl, K6	Віднімання з двох байт константи	Rdh:Rdl ← Rdh:Rdl – K6	Z,C,N,V,S	4	2
9	AND	Rd, Rr	Логічне І	Rd ← Rd • Rr	Z,N,V,S	3	1
10	ANDI	Rd*, K	Логічне І з константою	Rd* ← Rd* • K	Z,N,V,S	5	1
11	OR	Rd, Rr	Логічне АБО	Rd ← Rd v Rr	Z,N,V,S	3	1
12	ORI	Rd*, K	Логічне АБО з константою	Rd* ← Rd* v K	Z,N,V,S	5	1
13	EOR	Rd, Rr	Логічне виключне АБО	Rd ← Rd ⊕ Rr	Z,N,V,S	3	1
14	COM	Rd	Побітова інверсія	Rd ← $FF – Rd	Z,C,N,V,S	2	1
15	NEG	Rd	Зміна знака (формування додаткового коду)	Rd ← $00 – Rd	Z,C,N, V,H,S	2	1
16	INC	Rd	Інкремент значення регістра	Rd ← Rd + 1	Z,N,V,S	2	1
17	DEC	Rd	Декремент значення регістра	Rd ← Rd – 1	Z,N,V,S	2	1
18	ASR	Rd	Арифметичний зсув вправо	Rd(n) ← Rd(n+1), n = 0…6, C←Rd(0), Rd(7)←Rd(7)	Z,C,N,V	2	1
19	LSL	Rd	Логічний/арифметичний зсув вліво	Rd(n+1) ← Rd(n), Rd(0) ← 0, C←Rd(7)	Z,C,N,V	2	1
20	LSR	Rd	Логічний зсув вправо	Rd(n) ← Rd(n+1), Rd(7) ← 0, C←Rd(0)	Z,C,N,V,S	2	1
21	ROL	Rd	Циклічний зсув вліво через С	Rd(0)←C,Rd(n+1)← Rd(n), C←Rd(7)	Z,C,N, V,H,S	2	1
22	ROR	Rd	Циклічний зсув вправо через С	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V,S	2	1
23	TST	Rd	Перевірка на нуль чи від’ємне значення	Rd ← Rd • Rd	Z,N,V,S	2	1
24	CLR	Rd	Очищення регістра	Rd ← Rd ⊕ Rd	Z,N,V,S	2	1
25	SWAP	Rd	Обмін тетрадами	Rd(3…0)←Rd(7…4), Rd(7…4)←Rd(3…0)	-	2	1
26	SER	Rd	Встановлення регістра	Rd ← $FF	-	2	1
КОМАНДИ ПЕРЕДАЧІ КЕРУВАННЯ							
1	RJMP	k	Відносний безумовний перехід	PC ← PC + k + 1	-	13	2
2	IJMP		Непрямий безумовний перехід	PC ← Z	-	1	2
3	RCALL	k	Відносний безумовний виклик підпрограми	STACK ← PC+1, PC ← PC + k + 1, SP ← SP-2	-	13	3
4	ICALL		Непрямий безумовний виклик підпрограми	STACK ← PC+1, PC ← Z, SP ← SP-2	-	1	3

Продовження таблиці 1.3

№	Мнемо-нічне позначення	Операнди	Опис	Операція	Прапорці	Тип	Кільк. тактів
5	RET		Повернення з підпрограми	PC ← STACK, SP ← SP+2	-	1	4
6	RETI		Повернення з підпрограми обробки переривання	PC ← STACK, SP ← SP+2, I ← 1	I	1	4
7	CPSE	Rd, Rr	Порівняти, пропустити, якщо рівні	if (Rd = Rr), то PC ← PC + 2/3	-	3	1/2/3
8	CP	Rd, Rr	Порівняти	Rd - Rr	Z,N,V, C,H,S	3	1
9	CPC	Rd, Rr	Порівняти з перенесенням	Rd - Rr - C	Z,N,V, C,H,S	3	1
10	CPI	Rd*, K	Порівняти з константою	Rd* - K	Z,N,V, C,H,S	5	1
11	SBRC	Rr, b	Пропустити, якщо біт у регістрі скинутий	if (Rr(b) = 0), то PC ← PC + 2/3	-	7	1/2/3
12	SBRS	Rr, b	Пропустити, якщо біт у регістрі встановлений	if (Rr(b) = 1), то PC ← PC + 2/3	-	7	1/2/3
13	SBIC	P*, b	Пропустити, якщо біт у порту скинутий	if (P*(b) = 0), то PC ← PC + 2/3	-	9	1/2/3
14	SBIS	P*, b	Пропустити, якщо біт у порту встановлений	if (P*(b) = 1), то PC ← PC + 2/3	-	9	1/2/3
15	BRBS	s, k	Перейти, якщо прапорець у SREG встановлений	if (SREG(s) = 1) then C←PC+k+1	-	11	1/2
16	BRBC	s, k	Перейти, якщо прапорець у SREG скинутий	if(SREG(s) = 0) then PC←PC+k+1	-	11	1/2
17	BREQ	k	Перейти, якщо дорівнює	if (Z = 1) then PC ← PC + k + 1	-	12	1/2
18	BRCS	k	Перейти, якщо прапорець перенесення встановлений	if (C = 1) then PC ← PC + k + 1	-	12	1/2
19	BRNE	k	Перейти, якщо не дорівнює	if (Z = 0) then PC ← PC + k + 1	-	12	1/2
20	BRCC	k	Перейти, якщо прапорець перенесення скинутий	if (C = 0) then PC ← PC + k + 1	-	12	1/2
21	BRSH	k	Перейти, якщо дорівнює або більше	if (C = 0) then PC ← PC + k + 1	-	12	1/2
22	BRLO	k	Перейти, якщо менше	if (C = 1) then PC ← PC + k + 1	-	12	1/2
23	BRMI	k	Перейти, якщо мінус	if (N = 1) then PC ← PC + k + 1	-	12	1/2
24	BRPL	k	Перейти, якщо плюс	if (N = 0) then PC ← PC + k + 1	-	12	1/2
25	BRGE	k	Перейти, якщо більше або дорівнює (зі знаком)	if (N ⊕ V = 0) then PC ← PC + k+1	-	12	1/2
26	BRLT	k	Перейти, якщо менше (зі знаком)	if (N ⊕ V = 1) then PC← PC+ k+1	-	12	1/2
27	BRHS	k	Перейти, якщо прапорець половинного перенесення встановлений	if (H = 1) then PC ← PC + k + 1	-	12	1/2
28	BRHC	k	Перейти, якщо прапорець половинного перенесення скинутий	if (H = 0) then PC ← PC + k + 1	-	12	1/2
29	BRTS	k	Перейти, якщо прапорець T встановлений	if (T = 1) then PC ← PC + k + 1	-	12	1/2
30	BRTC	k	Перейти, якщо прапорець T скинутий	if (T = 0) then PC ← PC + k + 1	-	12	1/2
31	BRVS	k	Перейти, якщо прапорець переповнення встановлений	if (V = 1) then PC ← PC + k + 1	-	12	1/2
32	BRVC	k	Перейти, якщо прапорець переповнення скинутий	if (V = 0) then PC ← PC + k + 1	-	12	1/2
33	BRIE	k	Перейти, якщо переривання дозволені	if (I = 1) then PC ← PC + k + 1	-	12	1/2
34	BRID	k	Перейти, якщо переривання заборонені	if (I = 0) then PC ← PC + k + 1	-	12	1/2

Продовження таблиці 1.3

№	Мнемо-нічне позначення	Операнди	Опис	Операція	Прапорці	Тип	Кіл. тактів
КОМАНДИ ПЕРЕСИЛАННЯ ДАНИХ							
1	MOV	Rd, Rr	Копіювання регістра	Rd ← Rr	-	3	1
2	LDI	Rd*, K	Завантаження константи	Rd* ← K	-	5	1
3	LD	Rd, X	Непряме завантаження	Rd ← (X)	-	2	2
4	LD	Rd, X+	Непряме завантаження з постінкрементом	Rd ← (X), X ← X + 1	-	2	2
5	LD	Rd, -X	Непряме завантаження з переддекрементом	X ← X - 1, Rd ← (X)	-	2	2
6	LD	Rd, Y	Непряме завантаження	Rd ← (Y)	-	2	2
7	LD	Rd, Y+	Непряме завантаження з постінкрементом	Rd ← (Y), Y ← Y + 1	-	2	2
8	LD	Rd, -Y	Непряме завантаження з переддекрементом	Y ← Y - 1, Rd ← (Y)	-	2	2
9	LDD	Rd, Y+q	Непряме завантаження зі зміщенням	Rd ← (Y + q)	-	6	2
10	LD	Rd, Z	Непряме завантаження	Rd ← (Z)	-	2	2
11	LD	Rd, Z+	Непряме завантаження з постінкрементом	Rd ← (Z), Z ← Z+1	-	2	2
12	LD	Rd, -Z	Непряме завантаження з переддекрементом	Z ← Z - 1, Rd ← (Z)	-	2	2
13	LDD	Rd, Z+q	Непряме завантаження зі зміщенням	Rd ← (Z + q)	-	6	2
14	LDS	Rd, k	Пряме завантаження	Rd ← (k)	-	14	2
15	ST	X, Rr	Непряме збереження	(X) ← Rr	-	2	2
16	ST	X+, Rr	Непряме збереження з постінкрементом	(X) ← Rr, X ← X + 1	-	2	2
17	ST	- X, Rr	Непряме збереження з переддекрементом	X ← X - 1, (X) ← Rr	-	2	2
18	ST	Y, Rr	Непряме збереження	(Y) ← Rr	-	2	2
19	ST	Y+, Rr	Непряме збереження з постінкрементом	(Y) ← Rr, Y ← Y + 1	-	2	2
20	ST	- Y, Rr	Непряме збереження з переддекрементом	Y ← Y - 1, (Y) ← Rr	-	2	2
21	STD	Y+q, Rr	Непряме збереження зі зміщенням	(Y + q) ← Rr	-	6	2
22	ST	Z, Rr	Непряме збереження	(Z) ← Rr	-	2	2
23	ST	Z+, Rr	Непряме збереження з постінкрементом	(Z) ← Rr, Z ← Z + 1	-	2	2
24	ST	-Z, Rr	Непряме збереження з переддекрементом	Z ← Z - 1, (Z) ← Rr	-	2	2
25	STD	Z+q, Rr	Непряме збереження зі зміщенням	(Z + q) ← Rr	-	6	2
26	STS	k, Rr	Пряме збереження	(k) ← Rr	-	14	2
27	LPM*		Завантаження байта з програмної пам’яті	R0 ← (Z)	-	1	3
28	IN	Rd, P	Читання порту	Rd ← P	-	8	1
29	OUT	P, Rr	Запис у порт	P ← Rr	-	8	1
30	PUSH	Rr	Занесення регістра в стек	STACK ← Rr; SP← SP - 1	-	2	2
31	POP	Rd	Витягнення регістра зі стека	SP← SP + 1, Rd ← STACK	-	2	2
КОМАНДИ РОБОТИ З БІТАМИ							
1	SBR	Rd*, K	Встановити біт (біти) у регістрі	Rd* ← Rd* v K	Z,N,V,S	5	1
2	CBR	Rd* , K	Скинути біт (біти) у регістрі	Rd* ← Rd* • ($FF - K)	Z,N,V,S	5	1
3	SBI	P*, b	Встановити біт у РВВ	I/O(P*, b) ← 1	-	9	2
4	CBI	P*, b	Скинути біт у РВВ	I/O(P*, b) ← 0	-	9	2
5	BSET	S	Встановити вказаний розряд регістра SREG	SREG(s) ← 1	SREG(s)	10	1
6	BCLR	S	Скинути заданий розряд регістра SREG	SREG(s) ← 0	SREG(s)	10	1
7	BLD	Rd, b	Завантажити біт з T у регістр	Rd(b) ← T	-	7	1
8	BST	Rr, b	Зберегти біт з регістра в T	T ← Rr(b)	T	7	1
9	SEC		Встановити прапорець перенесення	C ← 1	C	1	1
10	CLC		Скинути прапорець перенесення	C ← 0	C	1	1
11	SEN		Встановити прапорець від’ємного числа	N ← 1	N	1	1
12	CLN		Скинути прапорець від’ємного числа	N ← 0	N	1	1

Закінчення таблиці 1.3

№	Мнемонічне позначення	Операнди	Опис	Операція	Прапорці	Тип	Кільк. тактів
13	SEZ		Встановити прапорець нуля	Z←1	Z	1	1
14	CLZ		Скинути прапорець нуля	Z←0	Z	1	1
15	SEI		Встановити прапорець переривань	I←1	I	1	1
16	CLI		Скинути прапорець переривань	I←0	I	1	1
17	SES		Встановити прапорець знака	S←1	S	1	1
18	CLS		Скинути прапорець знака	S←0	S	1	1
19	SEV		Встановити прапорець переповнення	V←1	V	1	1
20	CLV		Скинути прапорець переповнення	V←0	V	1	1
21	SET		Встановити прапорець T	T←1	T	1	1
22	CLT		Скинути прапорець T	T←0	T	1	1
23	SEH		Встановити прапорець половинного перенесення	H←1	H	1	1
24	CLH		Очистити прапорець половинного перенесення	H←0	H	1	1
КОМАНДИ КЕРУВАННЯ МІКРОКОНТРОЛЕРОМ							
1	NOP		Пуста операція	—	—	1	1
2	SLEEP		Переведення у режим сну	—	—	1	3
3	WDR		Скидання вартового таймера	—	—	1	1

Нижче наведено короткий опис окремих команд базового набору.

Арифметичні операції і команди зсуву

До даної групи належать команди, що виконують такі операції, як додавання, віднімання, зсув – вправо/вліво та інкремент/декремент. Усі операції виконуються тільки над регістрами загального призначення. Операнди можуть бути знаковими/беззнаковими числами, а також подані у додатковому коді.

Усі команди цієї групи виконуються за один машинний цикл, за винятком команд, що оперують двобайтовими значеннями, що виконуються за два цикли.

Логічні операції

Ці команди дозволяють виконувати стандартні логічні операції над байтами: «логічне множення» – І, «логічне додавання» – АБО, операцію «виключне АБО», а також обчислення зворотного і додаткового кодів числа. До цієї групи можна віднести також команди очищення/встановлення регістрів і команду перестановки

тетрад. Всі логічні операції виконуються за один машинний цикл над регістрами загального призначення, а результат зберігається в одному з РЗП.

Команди операцій з бітами

До даної групи належать команди, що виконують встановлення або скидання заданого розряду РЗП або РВВ. Є також команди для зміни розрядів регістра стану SREG, тому що перевірка стану розрядів саме цього регістра виконується найчастіше. Умовно до цієї групи можна віднести також дві команди передачі керування типу «перевірка/пропуск», що пропускають наступну команду в залежності від стану розряду РЗП або РВВ.

Усі задіяні розряди РВВ мають свої символічні імена. Визначення цих імен описано у тому ж файлі, що й визначення символічних імен для адрес регістрів [1; 2; 10]. Таким чином, після включення у програму зазначеного файлу в командах замість числових значень номерів розрядів можна буде вказувати їхні символічні імена.

Всі команди цієї групи виконуються за один машинний цикл, за винятком випадків, коли в результаті перевірки відбувається пропуск наступної команди. У цьому разі команда виконується за два або три машинних цикли залежно від довжини команди, що пропускається.

Команди пересилання даних

Команди цієї групи призначено для пересилання вмісту комірок, що лежать в просторі адрес статичної пам’яті даних – РЗП, РВВ та СОЗП. Поділ простору адрес на три частини обумовлює різноманітність команд даної групи. Пересилання даних може виконуватись в наступних напрямках:

- РЗП <=> РЗП;
- РЗП <=> РВВ;
- РЗП <=> статична пам’ять даних.

До даної групи також віднесяться команди PUSH і POP, які дозволяють зберігати у стеку і відновлювати зі стека вміст РЗП.

На виконання команд пересилання потрібно від одного до трьох машинних циклів в залежності від типу команди.

Команди передачі керування

У цю групу входять команди переходу, виклику підпрограм, повернення з них і команди типу «перевірка/пропуск», що пропускають наступну за ними команду при виконанні деякої умови. Також до цієї групи належать команди порівняння, що формують прапорці регістра SREG, які призначено переважно для роботи разом з командами умовного переходу. У командах цього типу виконується перевірка умови, результат якої впливає на виконання наступної команди. Якщо умова істинна, наступна команда ігнорується. Наприклад, команда SBRS Rd, b перевіряє розряд b регістра Rd і ігнорує (пропускає) наступну команду, якщо цей розряд дорівнює одиниці. Перехід до наступної інструкції виконується збільшенням лічильника команд на одиницю, а пропуск команди викликає завантаження нового значення в лічильник команд. Отже, коли умова, що перевіряється, істинна, у конвеєрі виникає затримка. Тривалість затримки залежить від довжини команди, що пропускається, і становить від одного до двох машинних циклів.

Команди передачі керування порушують нормальне (лінійне) виконання основної програми. Щоразу, коли виконується команда з цієї групи (окрім команд порівняння), нормальне функціонування конвеєра порушується. Перед завантаженням у конвеєр нової адреси він зупиняється й очищується послідовність виконуваних команд. Реініціалізація конвеєра призводить до того, що такі команди виконуються протягом декількох машинних циклів [1; 10].

В системі команд мікроконтролерів сімейства є команди як безумовного, так і умовного переходів. Команди непрямого – JMP і відносного – RJMP безумовного переходу є найпростішими в цій групі. Їх функція полягає тільки у записі нової адреси в лічильник команд.

При виконанні команди RJMP змінюється вміст лічильника команд шляхом додавання до нього або віднімання з нього деякого значення, що є операндом команди. Ця команда має обмеження за областю дії.

Через те, що операнд є 12-розрядним числом зі знаком, максимальна величина переходу відносно адреси наступної команди становить від -2048 до +2047 слів (приблизно ±4 Кбайт).

В програмах в якості операндів цієї команди замість констант використовуються мітки. Компілятор (асемблер) віднімає від адреси мітки адресу наступної команди і підставляє отримане значення у відповідне місце машинного коду команди. Нижче наведено приклад, який ілюструє сказане:

```
        cpi r16, $42; порівння регістра r16 з числом $42;
        brne error; перехід на мітку error, якщо r16 ≠ $42;
        rjmp ok ; безумовний перехід на мітку ok, якщо r16 = $42;
error:
        . . .
ok:     nop; місце переходу за командою RJMP.
```

Оскільки команда відносного переходу змінює вміст лічильника команд, вона виконується за 2 машинних цикли.

В результаті виконання команди непрямого переходу IJMP програма продовжує виконуватися з адреси, що міститься в індексному регістрі Z. Таким чином, дія команди зводиться до завантаження вмісту індексного регістра в лічильник команд.

На відміну від команди відносного переходу ця команда має менше обмежень за областю дії. Насправді, оскільки індексний регістр Z – 16-розрядний, максимально можлива величина переходу становить: 64 Кслів (128 Кбайт). Як і команда відносного переходу, команда непрямого переходу виконується за 2 машинних цикли.

Команди умовного переходу

В цих командах виконується перевірка умови, результат якої впливає на стан лічильника команд. Якщо умова істинна, відбувається перехід за вказаною адресою, якщо ж умова хибна, виконується наступна команда.

Команди умовного переходу мають обмеження за областю дії. Нове значення лічильника команд обчислюється додаванням до нього або відніманням від нього деякого зміщення. Оскільки під значення зміщення (число зі знаком) в слові

команди виділяється лише 7 біт, максимальна величина переходу відносно адреси наступної команди становить від -64 до +63 слів.

Оскільки перехід за вказаною адресою здійснюється завантаженням нового значення в лічильник команд, то у випадку істинності умови, що перевіряється, у конвеєрі виникає затримка тривалістю в один машинний цикл.

Всі команди умовного переходу можна розділити на дві підгрупи. Перша підгрупа – команди умовного переходу загального призначення. У цю підгрупу входять дві команди BRBS s, k і BRBC s, k, в яких явно вказується номер прапорця регістра SREG, який перевіряється. Відповідно, перехід здійснюється при SREG.s = 0 (BRBC) або SREG.s = 1 (BRBS). Іншу підгрупу складають 18 команд, кожна з яких виконує перехід за якою-небудь конкретною умовою: «дорівнює», «більше або дорівнює», «було перенесення» і т. ін. Одні з цих команд використовуються після порівняння беззнакових чисел, інші – після порівняння чисел зі знаком. Можливі умови, що перевіряються, а також відповідні їм команди умовного переходу наведено в таблиці 1.4.

Команди, які наведено в таблиці 1.4, є тільки еквівалентними мнемонічними позначеннями команд BRBS s, k та BRBC s, k з визначеними значеннями операнда – s. Наприклад, команда BREQ k має, такий же код операції, що і команда BRBS 1, k, а команда BRGE k – такий же, що і BRBC 4, k. За останньою командою відбувається перехід за значенням k, якщо прапорець S = 1. Значення S = (N+V), де N – значення прапорця від'ємного значення, а V – прапорця переповнення додаткового коду.

Команди виклику підпрограм

Для виклику підпрограм є дві команди: команда відносного виклику – RCALL і команда непрямого виклику – ICALL.

Враховуючі деякі відмінності, які описано нижче, команда RCALL працює так само, як і команда відносного безумовного переходу RJMP.

Команда RCALL зберігає у стеку значення лічильника команд (адресу наступної команди). Потім вміст лічильника команд збільшується або зменшується на деяке значення, що є операндом команди. Оскільки останній є 12-розрядним

числом зі знаком, максимальна величина переходу відносно адреси наступної команди становить від -2048 до +2047 слів (приблизно ±4 Кбайт).

Таблиця 1.4 – Зведена таблиця команд умовного переходу

Перевірка	**Логічна умова**	**Команда**	**Зворотна перевірка**	**Логічна умова**	**Команда**	**Тип даних**
Rd>Rr	(N⊕V) = 0	BRLT*	Rd≤Rr	(N⊕V)=1	BRGE*	Зі знаком
Rd≥Rr	(N⊕V) = 0	BRGE	Rd<Rr	(N⊕V)=1	BRLT	Зі знаком
Rd=Rr	Z = 1	BREQ	Rd≠Rr	Z=0	BRNE	Зі знаком
Rd≤Rr	(N⊕V) = 1	BRGE*	Rd>Rr	(N⊕V)=0	BRLT*	Зі знаком
Rd<Rr	(N⊕V) = 1	BRLT	Rd≥Rr	(N⊕V)=0	BRGE	Зі знаком
Rd>Rr	C = 0	BRLO*	Rd≤Rr	C=1	BRSH*	Без знака
Rd≥Rr	C = 0	BRSH/BRCC	Rd<Rr	C=1	BRLO/BRCS	Без знака
Rd=Rr	Z = 1	BREQ	Rd≠Rr	Z=0	BRNE	Без знака
Rd≤Rr	C = 1	BRSH*	Rd>Rr	C=0	BRLO*	Без знака
Rd<Rr	C = 1	BRLO/BRCS	Rd≥Rr	C=0	BRSH/BRCC	Без знака
«Перенесення»	C = 1	BRCS	«Немає перенесення»	C=0	BRCC	—
«Менше за нуль»	N = 1	BRMI	«Більше за нуль»	N=0	BRPL	—
«Переповнення»	V = 1	BRVS	«Немає переповнення»	V=0	BRVC	—
«Нуль»	Z = 1	BREQ	«Не нуль»	Z=0	BRNE	—
«Половинне перенесення»	H = 1	BRHS	«Немає половинного перенесення»	H=0	BRHC	—

* Оскільки у мікроконтролері відсутні команди переходів за умовами: більше та менше, або дорівнює, то для переходу за цими умовами операнди попередньої команди порівняння повинні бути записані у зворотному порядку, тобто замість CP Rd, Rr –> CP Rr, Rd.

У програмах в якості операндів команди RCALL, як і у випадку команди RJMP, використовуються мітки. Асемблер сам обчислює величину зміщення відносно адреси наступної команди шляхом віднімання від адреси мітки адреси наступної команди і підставляє це значення в машинний код команди.

Команда відносного виклику підпрограм виконується за 3 машинних цикли, два з яких витрачаються на збереження у стеку двох байт лічильника команд (адреси наступної команди).

Враховуючі деякі відмінності, які описано нижче, команда ICALL працює так само, як і команда непрямого безумовного переходу IJMP.

Команда ICALL зберігає у стеку значення лічильника команд (адресу наступної команди). Потім у лічильник команд завантажується вміст індексного

регістра. Оскільки індексний регістр – 16-розрядний, максимально можлива величина переходу становить 64 Кслів (128 Кбайт). Як і команда RCALL, команда непрямого виклику підпрограм виконується за 3 машинних цикли.

Підпрограма повинна закінчуватися командою повернення RET, як показано в наступному прикладі:

```
rcall   sp_test    ;  виклик підпрограми  sp_test
...                ;  текст основної програми
sp_test            ;  мітка підпрограми
push    r2         ;  збереження r2 у стеку
...                ;  виконання підпрограми
pop     r2         ;  відновити r2 зі стека
ret                ;  повернення з підпрограми
```

У наведеному вище прикладі команда ret замінює адресу, що міститься в лічильнику команд, адресою команди, наступної за командою rcall.

Команди повернення з підпрограм

В кінці кожної підпрограми обов'язково повинна міститися команда повернення з неї. В системі команд AVR-мікроконтролерів таких команд є дві. Для повернення з підпрограми, що викликається командами RCALL і ICALL, використовується команда RET. Для повернення з підпрограми обробки переривання використовується команда RETI.

Обидві команди відновлюють зі стека вміст лічильника команд, який зберігається там перед переходом до підпрограми. Команда повернення з підпрограми RETI додатково встановлює в одиницю прапорець глобального дозволу переривань I регістра SREG, що скидається апаратно при виникненні переривання.

На виконання кожної з команд повернення з підпрограми потрібно 4 машинних цикли.

Команди керування мікроконтролером

У цю групу входять 3 команди:

– NOP – пуста операція;

– SLEEP – переведення мікроконтролера в режим зниженого енергоспоживання;

– WDR – скидання вартового таймера.

Команди NOP і WDR виконуються за один машинний цикл, а команда SLEEP – за три машинних цикли.

Вище скорочено було розглянуто 118 базових команд. Реальна кількість команд більша, тому що за деякими номерами наведено опис команд, які виконуються за схожими алгоритмами, але відрізняються способами адресації операндів. Детальний опис базових команд наведено у [1; 2; 10].

8 Нові команди AVR-мікроконтролерів

Архітектура AVR-мікроконтролерів постійно оновлялася. Це оновлення стосується як структури мікроконтролера, так і його системи команд. Кожне сімейство, послідовно успадковує набір команд попереднього сімейства. Але є деякі виключення.

Нижче розглядаються нові команди, які на даний час з'явилися у нових AVR-мікроконтролерах: Mega та XMega (таблиця 1.5) [1; 3].

JMP

Код операції:

15														1	0
1	0	0	1	0	1	0	k	k	k	k	k	1	1	0	k
31														17	16
k	k	k	k	k	k	k	k	k	k	k	k	k	k	k	k

Алгоритм виконання:

$$PC \leftarrow k.$$

Опис: для мікроконтролерів з 22-розрядним програмним лічильником (PC), максимальна ємність пам'яті програм дорівнює 2^{22} = 4 Мслів (8 Мбайт), наприклад у контролері AT90SC6464C-USB, а для контролерів з 16-розрядним PC, відповідно 2^{16} = 64 Кслів (128 Кбайт).

Довжина команди: 2 слова (4 байти).

Таблиця 1.5 – Нові команди мікроконтролерів AVR

Команда	**Операція**	**Опис**	**Дія**	**Прапорці**	**Такти**
Classic Core AVR (до 128К програмного простору)					
JMP	k	Безумовний прямий перехід у пам’яті програм ємністю 64 К/4 Мслів	PC ← k (16/22 біти)	–	3/4
CALL	k	Безумовний прямий виклик підпрограми із пам’яті програм ємністю 64 К/4 Мслів	STACK ← PC + 2 SP ← SP - 2/SP - 3 PC ← k (16/22 біти)	–	4/5
Enhanced Core AVR (до 8К програмного простору)					
MUL	Rd, Rr	Множення беззнакових чисел	R1:R0 ← RdxRr	Z, C	2
MULS	Rd, Rr	Множення знакових чисел	R1:R0 ← RdxRr	Z, C	2
MULSU	Rd, Rr	Множення знакового числа на беззнакове	R1:R0 ← RdxRr	Z, C	2
FMUL	Rd, Rr	Множення дробових беззнакових чисел	R1:R0←RdxRr	Z, C	2
FMULS	Rd, Rr	Множення дробових знакових чисел	R1:R0 ← RdxRr	Z, C	2
FMULSU	Rd, Rr	Множення дробового знакового числа на беззнакове	R1:R0 ← RdxRr	Z, C	2
MOVW	Rd, Rr	Копіювання слова	Rd+1:Rd ← Rr+1:Rr	–	1
LPM Rd, Z	Rd	Завантаження регістра Rd із пам’яті програм з адресою у індексному регістрі Z	Rd ←(Z)	–	3
LPM Rd, Z+	Rd	Завантаження регістра Rd із пам’яті програм з адресою у індексному регістрі Z та наступним інкрементом вмісту Z	Rd ← (Z) Z ← Z + 1	–	3

Продовження таблиці 1.5

Команда	Операція	Опис	Дія	Прапорці	Такти
SPM		Використовується для самопрограмування мікроконтролера, який має відповідний блок	Дивись опис команди, наведений нижче	–	Виконується із пам'яті завантажувача
Enhanced Core AVR (до 128 Кслів програмного простору)					
BREAK		Зупинка процесора після виконання команди	PC ← PC + 1 зупинка виконання програми	–	1
Enhanced Core AVR (до 4 Мслів програмного простору)					
EIJMP		Безумовний непрямий перехід у пам'яті програм ємністю 4 Мслів	PC← (EIND:Z)	–	2
EICALL		Безумовний непрямий виклик підпрограми із пам'яті програм ємністю 4 Мслів	STACK← PC + 1 SP ← SP-3 PC ← (EIND:Z)	–	4
ELPM		Завантаження регістра R0 із розширеної пам'яті програм, з адресою у регістрах RAMPZ та Z	R0 ← (RAMPZ:Z)	–	3
ELPM Rd, Z	Rd	Завантаження регістра Rd із розширеної пам'яті програм, з адресою у регістрах RAMPZ та Z	Rd ← (RAMPZ:Z)	–	3
ELPM Rd, Z+	Rd	Завантаження регістра Rd із розширеної пам'яті програм, з адресою у регістрах RAMPZ та Z, та наступним інкрементом вмісту RAMPZ:Z	Rd ← (RAMPZ:Z) RAMPZ:Z ← RAMPZ:Z+1	–	3

Закінчення таблиці. 1.5

Команда	Операція	Опис	Дія	Прапорці	Такти
XMEGA CoreAVR					
DES	K	Шифрування даних	якщо H = 0: R15:R0←ENCRYPT (R15:R0, K) якщо H = 1: R15:R0←DECRYPT (R15:R0, K)	–	1/2
LAC Z, Rd	Rd	Запис (Z) у Rd та очищення бітів (Z) за маскою, яку вказано у Rd	Temp ← Rd; Rd ← (Z); (Z) ← ($FF – Temp)*(Z); PC ← PC + 1	–	1
LAS Z, Rd	Rd	Запис (Z) у Rd та встановлення бітів (Z) за маскою, яку вказано в Rd	Temp ← Rd; Rd ← (Z); (Z) ← Temp v (Z); PC ← PC + 1	–	1
LAT Z, Rd	Rd	Запис (Z) у Rd та підсумовування за модулем 2 бітів (Z) з маскою, яку вказано в Rd	Temp ← Rd; Rd ← (Z); (Z) ← Temp xor (Z); PC ← PC + 1	–	1
XCH Z, Rd	Rd	Обмін даними між Rd та (Z)	Temp ← Rd; Rd ← (Z); (Z) ← Temp; PC ←PC+1	–	1

CALL

Код операції:

15														1	0
1	0	0	1	0	1	0	k	k	k	k	k	1	1	1	k
31															16
k	k	k	k	k	k	k	k	k	k	k	k	k	k	k	k

Алгоритм виконання:

STACK ← PC + 2
SP ← SP - 2
PC ← k, якщо k = 16 біт

} Безумовний прямий виклик підпрограми із пам'яті програм ємністю 2^{16} = 64К слів (128 Кбайт)

STACK ← PC + 2
SP ← SP - 3
PC ← k, якщо k = 22 біт

} Безумовний прямий виклик підпрограми із пам'яті програм ємністю 2^{22} = 4М слів (8 Мбайт)

Довжина команди: 2 слова (4 байти).

MUL

Код операції:

15							8	7							0
1	0	0	1	1	1	r	d	d	d	d	d	r	r	r	r

Алгоритм виконання:

R1:R0 ← RdxRr

PC ← PC + 1.

Операнди: $0 \le d \le 31$; $0 \le r \le 31$ – беззнакові величини.

Довжина команди: 1 слово (2 байти).

MULS

Код операції:

15							8	7							0
0	0	0	0	0	0	1	0	d	d	d	d	r	r	r	r

Алгоритм виконання:

R1:R0 ← RdxRr

PC ← PC + 1.

Операнди: $16 \le d \le 31$; $16 \le r \le 31$ – знакові величини.

Довжина команди: 1 слово (2 байти).

MULSU

Код операції:

15							8	7							0
1	0	0	1	0	0	1	1	0	d	d	d	0	r	r	r

Алгоритм виконання:

R1:R0 ← RdxRr

PC ← PC + 1.

Операнди: 16 ≤ d ≤ 23; 16 ≤ r ≤ 23 (у Rr – беззнакове число, Rd – знакове число)

Довжина команди: 1 слово (2 байти).

FMUL

Код операції:

15							8	7							0
0	0	0	0	0	0	1	1	0	d	d	d	0	r	r	r

Алгоритм виконання:

R1:R0 ← RdxRr

PC ← PC + 1.

Опис: числа подаються у вигляді N.Q, де N – значення числа до коми у двійковому вигляді, Q – значення числа після коми у двійковому вигляді. Таким чином операнди подаються в вигляді (N1.Q1) і (N2.Q2). Беззнакові 8-розрядні дробові числа використовують формат, в якому числа можуть лежати у діапазоні: [0, 2]. Біти 6…0 являють собою дробову частину, а 7-й біт – цілу частину (0 або 1), тобто 1.7 формат. Результат множення (формат результату): 2.14 зсувається вліво на один розряд для приведення до формату: 1.15 та записується у пару регістрів R1:R0.

Операнди: 16 ≤ d ≤ 23; 16 ≤ r ≤ 23 – беззнакові дробові величини.

Довжина команди: 1 слово (2 байти).

FMULS

Код операції:

15							8	7							0
0	0	0	0	0	0	1	1	1	d	d	d	0	r	r	r

Алгоритм виконання:

R1:R0 ← RdxRr

PC ← PC + 1.

Опис: числа подаються в вигляді N.Q, де N – значення числа до коми у двійковому вигляді, Q – значення числа після коми у двійковому вигляді. Таким чином операнди подаються в вигляді (N1.Q1) і (N2.Q2). Дробові числа зі знаком подібно до цілих чисел зі знаком використовують формат двійкового доповнення, що дозволяє подавати перші у діапазоні: [-1, 1]. Якщо, наприклад, ціле число без знака має двійковий код: 10110010, то його десятковий еквівалент дорівнює: 128 + 32 + 16 + 2 = 178. Відповідно, десятковий еквівалент цього числа як знакового цілого дорівнює: 178 - 256 = -78. Якщо наведене вище двійкове число подати у форматі беззнакового дробового, то отримаємо:

1 + 0,25 + 0,125 + 0,015625 = 1,390625.

Аналогічно наведеному вище правилу обчислення цілих чисел зі знаком можна отримати десятковий еквівалент двійкового числа 10110010, якщо вважати що воно відповідає дробовому числу зі знаком: 1,390625 - 2 = -0,609375.

Результат множення (формат результату): 2.14 зсувається вліво на один розряд для приведення до формату: 1.15 та записується у пару регістрів R1:R0.

Операнди: $16 \le d \le 23$; $16 \le r \le 23$ – дробові величини зі знаком.

Довжина команди: 1 слово (2 байти).

FMULSU

Код операції:

15							8	7							0
0	0	0	0	0	0	1	1	1	d	d	d	1	r	r	r

Алгоритм виконання:

R1:R0 ← RdxRr

PC ← PC + 1.

Опис: числа подаються в вигляді N.Q, де N – значення числа до коми у двійковому вигляді, Q – значення числа після коми у двійковому вигляді. Таким чином операнди подаються в вигляді: N1.Q1 і N2.Q2 (дивись опис команди FMULS).

Операнди: $16 \le d \le 23$; $16 \le r \le 23$ (обидві величини дробові, одна з яких, у Rr – беззнакова, друга величина, у Rd – знакова).

Довжина команди: 1 слово (2 байти).

MOVW

Код операції:

15 8 7 0

0	0	0	0	0	0	0	1	d	d	d	d	r	r	r	r

Алгоритм виконання:

Rd+1:Rd ← Rr+1:Rr

PC ← PC + 1.

Опис: команда копіює вміст однієї пари регістрів загального призначення в іншу пару регістрів загального призначення. Пара регістрів джерела (Rr+1:Rr) залишається незмінною. Пара регістрів приймача (Rd+1:Rd) завантажується копією регістрів Rr+1:Rr. Номери регістрів Rr, Rd повинні бути парними.

Операнди: $0 \le d \le 30$; $0 \le r \le 30$.

Довжина команди: 1 слово (2 байти).

LPM Rd, Z

Код операції:

15 8 7 0

1	0	0	1	0	0	0	d	d	d	d	d	0	1	0	0

Алгоритм виконання:

Rd ← ПП (Z)

PC ← PC + 1.

Операнди: 0 ≤ d ≤ 31, ПП (Z) – комірка пам’яті програм, адреса якої міститься в індексному регістрі Z.

Довжина команди: 1 слово (2 байти).

LPM Rd, Z+

Код операції:

15							8	7							0
1	0	0	1	0	0	0	d	d	d	d	d	0	1	0	1

Алгоритм виконання:

Rd ← ПП (Z), Z ← Z + 1

PC ← PC + 1.

Операнди: 0 ≤ d ≤ 31, ПП (Z) – комірка пам’яті програм, адреса якої міститься в індексному регістрі Z.

Довжина команди: 1 слово (2 байти).

SPM

Код операції:

15							8	7							0
1	0	0	1	0	1	0	1	1	1	1	0	1	0	0	0

Опис: ця команда використовується для самопрограмування мікроконтролера, який має відповідний блок. Команда зберігається, і відповідно виконується, у спеціальній області пам’яті, яка називається завантажувачем. Існує п’ять режимів використання команди, які програмуються за допомогою додаткових регістрів керування:

(RAMPZ : Z) ← $FFFF, очищення сторінки пам’яті програм;

(RAMPZ : Z)← R1:R0, запис слова у тимчасовий буфер пам’яті програм;

(RAMPZ : Z) ← R1:R0, запис сторінки у тимчасовий буфер пам’яті програм;

(RAMPZ : Z)← TEMP, запис тимчасового буфера у пам’ять програм;

BLBITS ← R1:R0, встановлення бітів блокування завантаження програм.

Більш детальний опис цієї команди наведено у [1; 2].

BREAK

Код операції:

15 8 7 0

1	0	0	1	0	1	0	1	1	0	0	1	1	0	0	0

Алгоритм виконання:

PC ← PC+1.

Опис: ця інструкція використовується налагоджувачем, який вбудовано у мікроконтролер, та зазвичай в робочих програмах не використовується. Після виконання команди BREAK процесор зупиняється, що дає можливість налагоджувачеві працювати із внутрішніми ресурсами.

Операнди: немає.

Довжина команди: 1 слово (2 байти).

EIJMP

Код операції:

15 8 7 0

1	0	0	1	0	1	0	0	0	0	0	1	1	0	0	1

Алгоритм виконання:

PC (15:0) ← Z (15:0)

PC (21:16) ← EIND (6 біт),

де EIND – ім’я додаткового регістра керування.

Довжина команди: 1 слово (2 байти).

EICALL

Код операції:

15 8 7 0

1	0	0	1	0	1	0	1	0	0	0	1	1	0	0	1

Алгоритм виконання:

STACK ← PC + 1

SP ← SP - 3

PC (15:0) ← Z(15:0)

PC (21:16) ← EIND (6 біт),

де EIND – ім’я додаткового регістра керування.

Довжина команди: 1 слово (2 байти).

ELPM

Код операції:

15 8 7 0

1	0	0	1	0	1	0	1	1	1	0	1	1	0	0	0

Алгоритм виконання:

R0 ← ПП (RAMPZ:Z),

де RAMPZ – ім’я додаткового регістра керування;

PC ← PC + 1.

Операнди: R0 – регістр загального призначення; (RAMPZ:Z) – комірка пам’яті програм, адреса якої міститься у двох регістрах: RAMPZ (6 біт) та Z – 16 біт.

Довжина команди: 1 слово (2 байти).

ELPM Rd, Z

Код операції:

15 8 7 0

1	0	0	1	0	0	0	d	d	d	d	d	0	1	1	0

Алгоритм виконання:

Rd ← ПП (RAMPZ:Z),

де RAMPZ – ім’я додаткового регістра керування;

PC ← PC + 1.

Операнди: Rd ($0 \le d \le 31$) – регістр загального призначення;

ПП (RAMPZ:Z) – комірка пам’яті програм, адреса якої міститься у двох регістрах: RAMPZ (6 біт) та Z – 16 біт.

Довжина команди: 1 слово (2 байти).

ELPM Rd, Z+

Код операції:

15 8 7 0

1	0	0	1	0	0	0	d	d	d	d	d	0	1	1	1

Алгоритм виконання:

Rd ← ПП (RAMPZ:Z), RAMPZ:Z ← RAMPZ:Z+1,

де RAMPZ – ім’я додаткового регістра керування;

PC ← PC + 1.

Операнди: Rd ($0 \le d \le 31$) – регістр загального призначення;

ПП (RAMPZ:Z) – комірка пам’яті програм, адреса якої міститься у двох регістрах: RAMPZ (6 біт) та Z – 16 біт.

Довжина команди: 1 слово (2 байти).

DES

Алгоритм виконання:

R15:R0 ← ENCRYPT(R15:R0, K), якщо прапорець H = 0,

R15:R0 ← DECRYPT(R15:R0, K), якщо прапорець H = 1.

Опис: команда DES використовується в сучасних мікроконтролерах сімейства XMega для шифрування інформації. Більш детальну інформацію про команду та метод кодування, наведено у додатковій літературі [1; 19].

LAC

Код операції:

15 8 7 0

1	0	0	1	0	0	1	d	d	d	d	d	0	1	1	0

Алгоритм виконання:

Temp ← Rd;

Rd ← (Z);

(Z) ← ($FF – Temp) * (Z);

PC ← PC + 1.

Опис: команда LAC записує Rd у тимчасовий регістр Temp, пересилає (Z) в Rd, інвертоване значення Temp множить на (Z) та результат записує в (Z).

Таким чином до виконання команди в Rd знаходиться маска, за якою необхідно скинути у нуль відповідні біти (Z), а після виконання команди результат записується у (Z) з обнуленими бітами на тих місцях, де у масці знаходилась одиниця. При виконанні команди початкове значення (Z) записується у Rd.

Операнди: $0 \leq d \leq 31$.

Довжина команди: 1 слово (2 байти).

LAS

Код операції:

15 8 7 0

1	0	0	1	0	0	1	d	d	d	d	d	0	1	0	1

Алгоритм виконання:

Temp ← Rd;

Rd ← (Z);

(Z) ← Temp v (Z);

PC ← PC + 1.

Опис: команда LAS записує Rd у тимчасовий регістр Temp, пересилає (Z) в Rd, логічно підсумовує Temp і (Z) та результат записує в (Z). Таким чином до виконання команди в Rd знаходиться маска, за якою необхідно встановити в одиницю відповідні біти (Z), а після виконання команди результат записується у (Z) з встановленими бітами на тих місцях, де у масці знаходилась одиниця. При виконання команди початкове значення (Z) записується у Rd.

Операнди: $0 \leq d \leq 31$.

Довжина команди: 1 слово (2 байти).

LAT

Код операції:

15 8 7 0

1	0	0	1	0	0	1	d	d	d	d	d	0	1	1	1

Алгоритм виконання:

Temp ← Rd;

Rd ← (Z);

(Z) ← Temp xor (Z);

PC ← PC + 1.

Опис: команда LAT записує Rd у тимчасовий регістр Temp, пересилає (Z) в Rd, виконує логічне «Виключне АБО» (XOR) між Temp і (Z) та результат записує в (Z). Таким чином до виконання команди в Rd знаходиться маска, за якою необхідно проінвертувати відповідні біти (Z), а після виконання команди результат записується у (Z) з проінвертованими бітами на тих місцях, де у масці знаходилась одиниця. При виконання команди початкове значення (Z) записується у Rd.

Операнди: $0 \leq d \leq 31$.

Довжина команди: 1 слово (2 байти).

XCH

Код операції:

15 8 7 0

1	0	0	1	0	0	0	d	d	d	d	d	0	1	0	0

Алгоритм виконання:

Temp ← Rd;

Rd ← (Z);

(Z) ← Temp;

PC ←PC+1.

Опис: команда XCH міняє місцями значення, що знаходиться за адресою, вказаною в Z, зі значенням Rd.

Операнди: $0 \leq d \leq 31$.

Довжина команди: 1 слово (2 байти).