Система команд і програмна модель AVR

1 Програмна модель мікроконтролерів AVR

Пам'ять програм має обсяг 1К х 16 і займає адресний простір $000-$3FF. Пам'ять даних складається з 32 регістрів загального призначення ($00-$1F), 64 регістрів введення/виведення ($20-$5F) і оперативної пам'яті 128 х 8 ($60-$DF). При адресації пам'яті даних використовуються п'ять режимів адресації: безпосередня адресація, непряма зі зсувом, непряма, непряма з предекрементом і непряма з постдекрементом. Регістри з R26 по R31 реєстрового файлу працюють як X, Y і Z регістри показники непрямої адресації.

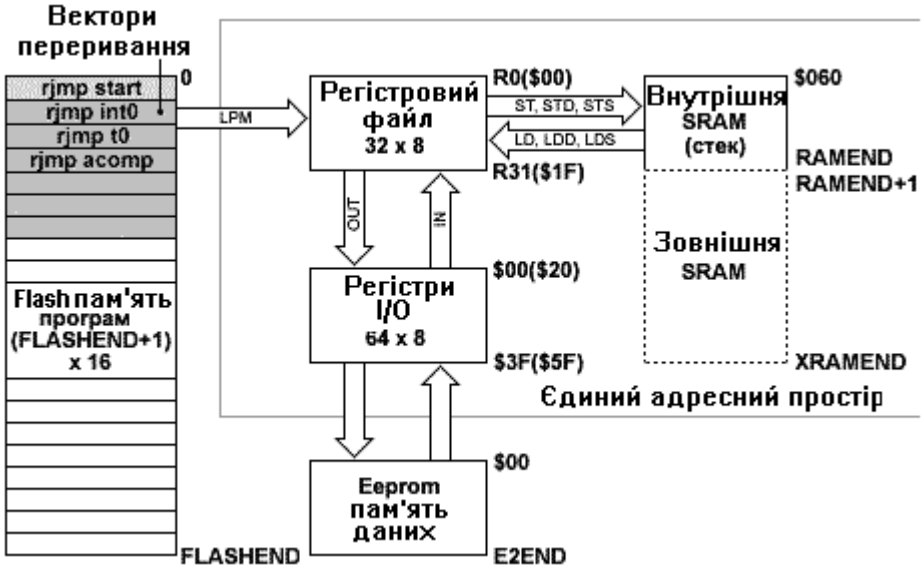

Рисунок 1 – Програмна модель AVR-мікроконтролерів

Непрямій адресації зі зсувом доступні 63 адреси щодо базових адрес, які знаходяться в регістрах X, Y, Z. При використанні непрямої адресації з автоматичним предекрементом і постдекрементом автоматично декрементуються і інкрементуються адреси, що записані в регістри X, Y і Z. Усіма цими режимами перекривається весь адресний простір даних, включаючи 32 регістра загального призначення і 64 регістра I/O.

На рисунку 2 зображена програмна модель AVR-мікроконтролерів, що являє собою діаграму програмно доступних ресурсів AVR. Центральним блоком на цій діаграмі є регістровий файл на 32 оперативних регістра (R0-R31), що безпосередньо доступні ALU. Старші регістри об'єднані парами й

утворять три 16-розрядних регістри, призначених для непрямої адресації комірок пам'яті .

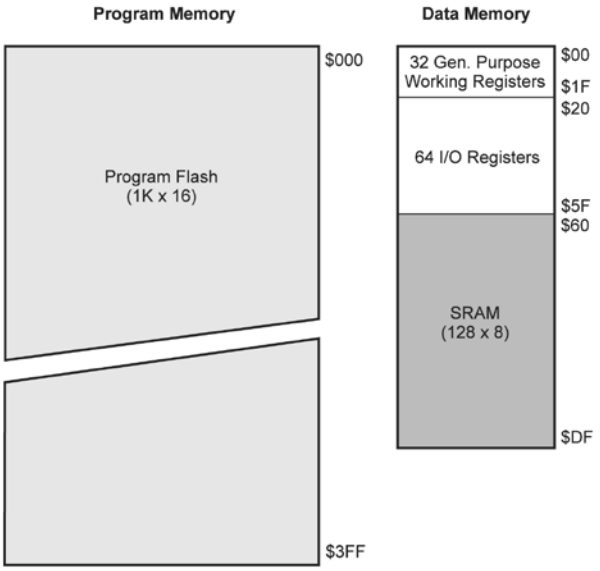

Рисунок 2 – Карта пам'яті

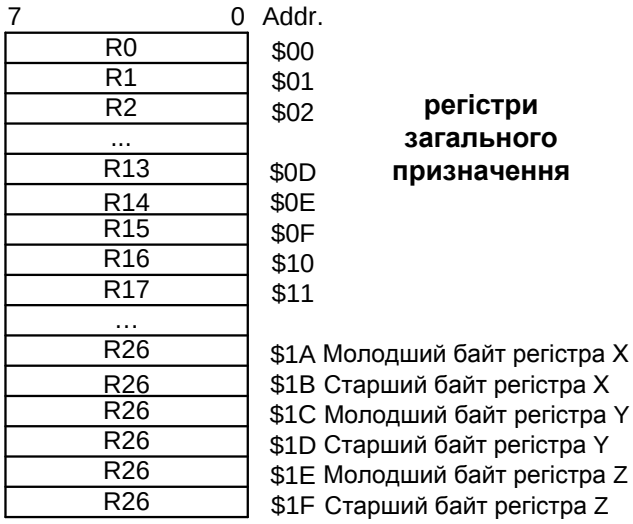

Рисунок 3– Регістри загального призначення мікроконтролерів AVR.

Оперативний запам’ятовуючий пристрій SRAM

SRAM даних AT90S2313 має обсяг 128x8 байт і займає адресний простір від $60 до $DF (рис.6). Шість регістрів з R26 по R31, крім звичайної для інших регістрів функцій, виконують функцію 16-розрядних регістрів покажчиків адреси при непрямій адресації SRAM. Ці три регістри непрямої адресації визначаються як регістри X,Y і Z.

Постійний запам'ятовуючий пристрій FlashROM призначений для

збереження кодів команд програми і констант. Комірка пам'яті містить 16 розрядів. У ній можуть зберігатися код команди формату "слово", половина коду команди формату "два слова" або коди двох констант.

При читанні кодів команд адреса в FlashROM надходить з лічильника команд. При читанні констант адреса надходить з пари Z регістрів загального призначення.

Запис кодів у FlashROM виконується в процесі програмування побайтно. У пам'яті програм можна розміщувати також будь-які дані, що в процесі роботи програми залишаються незмінними. Для зчитування цих даних з програмної пам'яті використовується команда LPM.

```
        Ldi ZH, high(2*char)
        Ldi ZL, low(2*char)


load:   Lpm
        adiwZL,1
        rjmp load

char:
    .db"abcd"
```

Вказівник стека

Стек, головним чином, використовується:

- для тимчасового зберігання даних;
- для зберігання локальних змінних;
- для зберігання адреси виходу з підпрограми або процедури обробки переривання.

Регістр покажчика стека завжди вказує на його вершину. Стек виконаний таким чином, що переміщається від своєї вершини вниз, до елементів пам'яті з меншою адресою. З цієї причини команда PUSH (записати в стек) зменшує покажчик стека.

Покажчик стека вказує на стекову область в пам'яті даних (SRAM). У стеку, окрім іншого, зберігаються;

- адреса виходу з підпрограми;
- адреса виходу з процедури обробки переривання.

Тому у будь-якій програмі адреса початку стека потрібно задати перед тим, як буде викликана будь-яка підпрограма, і перед тим, як будуть дозволені переривання. Покажчик стека має бути встановлений на адресу не нижче 0x60.

Покажчик стека в усіх мікросхемах AVR виконаний у вигляді двох 8-розрядних регістрів введення-виведення. У деяких моделях, у тому числі і в ATiny2313, об'єм пам'яті даних настільки малий, що для покажчика стека використовується тільки молодший з регістрів покажчика стека (SPL –$3D). Регістр SPH у них відсутній. Для ініціалізаціїї вершини стеку використовують комнду

```
Ldi  r16, RAMEND
out SPL ,r16
```

Постійний запам'ятовуючий пристрій EEPROM

Постійний запам'ятовуючий пристрій EEPROM призначений для збереження даних, записаних при програмуванні мікроконтролера й одержуваних у процесі виконання програми.

EEPROM має відособлений адресний простір. При звертанні до EEPROM адреса записується в регістр адреси EEAR ($1E). Байт, призначений для запису, заноситься в регістр даних EEDR ($1D). Байт, одержуваний при читанні, надходить у цей самий регістр. Для керування процедурами запису і читання використовується регістр керування EECR ($1C).

7	6	5	4	3	2	1	0	
–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	EECR
R	R	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	X	X	0	0	X	0	

Рисунок 4 – Регістр керування EEPROM

Біти 5,4 - EEPM1 і EEPM0: Розряди вибору режиму EEPROM.
Біти установки режиму програмування EEPROM визначають, яким способом виконуватиметься команда програмування, якщо скинутий прапор EEPE. Після системного скидання розряди EEPMn будуть встановлені в 0b00 у тому випадку, якщо в цей час не відбувається процес програмування EEPROM. Будь-які значення EEPMn ігноруються, коли установльон прапор EEPE.

Таблиця 1 – Біти вибору режиму EEPROM

EEPM1	EEPM0	Час програмування	Операція
0	0	3,4 мс	Стирання і запис за одну операцію (атомарна дія)
0	1	1,8 мс	Тільки стирання
1	0	1,8 мс	Тільки запис
1	1	-	Зарезервовано

Біт 3 - EERIE: Дозвіл переривання від EEPROM. Якщо цей розряд встановлений в одиницю, переривання дозволені (якщо прапор I регістра SREG також установленbq в одиницю).

Біт 2 - EEMPE: Управління дозволом програмування EEPROM. Значення біта EEMPE визначає функціонування прапора EEPE. Якщо біт EEMPE встановлений (рівний 1), установка біта EEPE в одиницю викликає програмування EEPROM за вибраною адресою. Впродовж чотирьох машинних циклів відразу після установки EEMPE треба робити установку EEPE. Інакше біт EEMPE буде апаратно скинутий, а програмування виявиться неможливим.

Біт 1 - EEPE: Дозвіл програмування EEPROM. Цей біт управляє процесом програмування EEPROM. Установка біта EEPE в одиницю викликає один з варіантів програмування EEPROM у відповідності зі значеннями бітів EEPMn. Перед тим, як записувати в EEPE одиницю, необхідно раніше встановити в одиницю біт EEMPE. Інакше процес програмування EEPROM не почнеться.

Після закінчення процесу програмування біт EEPE автоматично скидається в нуль. Відразу після установки EEPE в одиницю робота CPU призупиняється на два машинних цикли.

Біт 0 - EERE: Дозвіл читання EEPROM

Для запису байта в EEPROM необхідно:

- записати адресе в регістр адреси;
- записати байт у регістр даних;
- установити в одиничний стан розряд EEMPE регістра EECR,
- при EEMPE = 1 встановити в одиничний стан розряд EEPE регістра EECR.

Процедура запису виконується в залежності від величини напруги живлення за 2,5-4 мс. При завершенні запису розряд EEPE регістра EECR

апаратно скидається в нульовий стан.

Розряд EEMPE зберігає одиничний стан протягом 4-х тактів після установки і апаратно скидається в нульовий стан.

Для читання байта з EEPROM необхідно:

- записати адреса в регістр адреси;
- установити в одиничний стан розряд EERE регістра EECR. Прочитаний байт надходить у регістр даних. Розряд EERE регістра EECR апаратно скидається в нульовий стан.

2 Система команд

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 <типAVR>

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 2 – 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.

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.

Арифметичні і логічні інструкції

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

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)