СИСТЕМА КОМАНД

Система команд микроконтроллера **КР1878ВЕ1** включает 52 команды, объединенные в 6 групп:

- **двухоперандные команды** – производят арифметические и логические операции над двумя операндами, адреса которых образуются из индексов, содержащихся в самой команде, и значений регистров адресов сегментов. Один из операндов не изменяет своего содержимого и обозначается далее **src** (источник). Результат операции помещается во второй операнд, обозначаемый **dst** (приемник);
- **литерные команды** – производят арифметические и логические операции с операндом **dst** и литерой **const**, указанной в самой команде. Результат операции помещается в операнд, обозначаемый **dst** (приемник);
- **однооперандные команды** – производят арифметические или логические операции над операндом, адрес которого образуются из индекса, содержащегося в самой команде, и значения регистра адреса сегмента. Результат операции помещается в тот же операнд, обозначаемый **dst** (приемник);
- **команды работы со служебными регистрами и регистром состояния процессора** – предназначены для безусловной установки необходимых значений регистров адресов сегментов для тех случаев, когда необходима адресация к новым переменным или регистрам ввода/вывода, не охватываемым текущими сегментами. Также этой командой загружаются регистры косвенной адресации и устанавливаются режимы их работы;
- **команды передачи управления** – производят передачу управления программе, находящейся по адресу, указанному в младших десяти разрядах команды перехода, либо безусловно, либо по одному из условий перехода;
- **специальные команды** – относятся к классу команд, не вписывающихся в предыдущие разделы, и предназначены в основном для управления определенными состояниями процессора.

Полностью система команд представлена в табл. 8. При описании системы команд в таблице приняты следующие сокращения:

s (**src**) – операнд источника;
d (**dst**) – операнд приемника;
c (**const**) – константа;
p (**place**) – местоположение тетрады константы;
a (**address**) – адрес команды;
n (**number**) – номер служебного регистра;
b (**bit**) – устанавлимое значение разряда в RS;
***** – разряд PC меняется;
- – разряд PC не изменяется.

ДВУХОПЕРАНДНЫЕ КОМАНДЫ

Двухоперандные команды производят арифметические и логические операции над двумя операндами, адреса которых образуются из индексов, содержащихся в самой команде, и значений регистров адресов сегментов. Один из операндов не изменяет своего содержимого и обозначается далее **src** (источник). Результат операции помещается во второй операнд, обозначаемый **dst** (приемник).

Двухоперандные команды имеют единый формат. При выполнении любой команды производится чтение обоих операндов из памяти (как оперативной памяти данных, так и регистров периферийных устройств) и дальнейшая запись результата в память по адресу операнда **dst**. При выполнении команды пересылки **MOV** происходит чтение одного операнда **src**.

Таблица 8

Система команд ТЕСЕЙ

Наименование команды	Мне-мони-ка	Код команды	Действие команды	Состояние S Z C oF dC
Двухоперандные команды				
Пересылка	MOV	0000 01ss sssd dddd	src → dst	* * - 0 -
Сравнение	CMP	0000 10ss sssd dddd	dst - src S,Z,C → S	* * * * *
Сложение	ADD	0001 00ss sssd dddd	dst + src → dst	* * * * *
Вычитание	SUB	0000 11ss sssd dddd	dst - src → dst	* * * * *
Логическое И	AND	0001 01ss sssd dddd	dst. AND.src → dst	* * 0 0 0
Логическое ИЛИ	OR	0001 10ss sssd dddd	dst .OR. src → dst	* * 0 0 0
Исключающее ИЛИ	XOR	0001 11ss sssd dddd	dst .XOR.src → dst	* * 0 0 0
Литерные команды				
Пересылка литеры	MOVL	010c cccc cccd dddd	const → dst	* * - 0 -
Сравнение с литерой	CMPL	011c cccc cccd dddd	dst - const S,Z,C → RS	* * * * *
Сложение с литерой	ADDL	0011 00cc cccd dddd	dst + sconst → dst	* * * * *
Вычитание литеры	SUBL	0010 11cc cccd dddd	dst - sconst → dst	* * * * *
Сброс разрядов	BIC	0010 10pc cccd dddd	NOT(const).AND.dst → dst	* * 0 0 0
Установка разрядов	BIS	0011 10pc cccd dddd	dst. OR. tconst → dst	* * 0 0 0
Инверсия разрядов	BTG	0011 11pc cccd dddd	dst.XOR. tconst → dst	* * 0 0 0
Проверка разрядов	BTT	0011 01pc cccd dddd	dst.AND. tconst, S,Z → RS	* * 0 0 0
Одноопераднные команды				
Обмен тетрад	SWAP	0000 0000 001d dddd	dst(n) → dst(n+4) n<4 dst(n) → dst(n-4)	* * 0 0 0
Смена знака	NEG	0000 0000 010d dddd	-dst → dst	* * * * *
Инверсия всех разрядов	NOT	0000 0000 011d dddd	NOT(dst) → dst	* * - 0 -
Логический сдвиг влево	SHL	0000 0000 100d dddd	dst(n) → dst(n+1), 0 → dst(0), dst(7) → C	* * * * 0
Логический сдвиг вправо	SHR	0000 0000 101d dddd	dst(n+1) → dst(n), 0 → dst(7), dst(0) → C	0 * * 0 0
Арифметический сдвиг вправо	SHRA	0000 0000 110d dddd	dst(n+1) → dst(n), dst(7) → dst(7), dst(0) → C	* * * 0 0
Циклический сдвиг влево	RLC	0000 0000 111d dddd	dst(n) → dst(n+1), C → dst(0), dst(7) → C	* * * * 0
Циклический сдвиг вправо	RRC	0000 0001 000d dddd	dst(n+1) → dst(n), C → dst(7), dst(0) → C	* * * 0 0
Сложение с переносом	ADC	0000 0001 001d dddd	dst + C → dst	* * * * *
Вычитание переноса	SBC	0000 0001 010d dddd	dst - C → dst	* * * * *

Продолжение таблицы 8

Система команд ТЕСЕЙ

Наименование команды	Мне-мони-ка	Код команды	Действие команды	Состояние S Z C oF dC
Команды работы со служебными регистрами и регистром состояния				
Загрузка служебных регистров	LDR	0010 0ccc cccc cnnn	const → reg	- - - - -
Запись в служебные регистры	MTPR	0000 0010 nnns ssss	src → reg	- - - - -
Чтение служебных регистров	MFPR	0000 0011 nnnd dddd	reg → dst	- - - - -
Запись в стек данных	PUSH	0000 0000 0001 0nnn	reg → data stack, DSP = DSP + 1	- - - - -
Чтение из стека данных	POP	0000 0000 0001 1nnn	data stack → reg, DSP = DSP - 1	- - - - -
Установка разрядов RS	SST	0000 0001 1000 bbbb	if mask(n) =1 then RS(n) = 1	* * * - -
Сброс разрядов RS	CST	0000 0001 1100 bbbb	if mask(n) = 1 then RS(n) = 0	* * * - -
Проверка переполнения	TOF	0000 0000 0000 0100	OF → Z	- * - - -
Проверка тетрадного переноса	TDC	0000 0000 0000 0101	DC → Z	- * - - -
Команды передачи управления				
Безусловный переход	JMP	1000 00aa aaaa aaaa	address → PC	- - - - -
Переход к подпрограмме	JSR	1001 00aa aaaa aaaa	PC → istack, address → PC, ISP = ISP+ 1	- - - - -
Переход по Z=0 (не равно)	JNZ (JNE)	1011 00aa aaaa aaaa	address → PC if Z = 0	- - - - -
Переход по Z=1 (равно)	JZ (JEQ)	1010 00aa aaaa aaaa	address → PC if Z = 1	- - - - -
Переход по S=0 (плюс)	JNS	1100 00aa aaaa aaaa	address → PC if S = 0	- - - - -
Переход по S=1 (минус)	JS	1101 00aa aaaa aaaa	address → PC if S = 1	- - - - -
Переход по C=0	JNC	1110 00aa aaaa aaaa	address → PC if C = 0	- - - - -
Переход по C=1	JC	1111 00aa aaaa aaaa	address → PC if C = 1	- - - - -
Косвенный переход	IJMP	0000 0000 0000 0011	IR1 → PC	- - - - -
Косвенный переход к подпрограммме	IJSR	0000 0000 0000 0111	PC → istack, IR1 → PC, ISP = ISP + 1	- - - - -
Возврат из подпрограммы	RTS	0000 0000 0000 1100	istack → PC, ISP = ISP - 1	- - - - -
Возврат из подпро-граммы с битом C	RTSC	0000 0000 0000 111c	istack → PC c → RS(0), ISP = ISP - 1	- - * - -
Возврат из прерывания	RTI	0000 0000 0000 1101	istack → PC , data stack → RS	* * * * *
Специальные команды				
Нет операции	NOP	0000 0000 0000 0000		- - - - -
Ожидание	WAIT	0000 0000 0000 0001	RS(3) = 1 (INT Enable)	- - - - -
Останов	STOP	0000 0000 0000 1000	RS(3) = 1 (INT Enable)	- - - - -
Сброс	RESET	0000 0000 0000 0010	DSP = 0, ISP = 0	- - - - -
Прогон стека команд	SKSP	0000 0000 0000 0110	ISP = ISP- 1	- - - - -

Формат двухоперандных команд:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Код команды						№ SR src		Индекс src			№ SR dst		Индекс dst		

№ SR	Назначение
00	Сегмент **A**
01	Сегмент **B**
10	Сегмент **C**
11	Сегмент **D**

Соответствие номеров служебных регистров сегментам адресации

Команда пересылки

Мнемоника: **MOV dst,src** (MOVe)

Код команды: **000001 | src | dst**

Действие команды: **src ® dst**

Описание команды: Содержимое операнда **src** пересылается в операнд **dst**.

Влияние на разряды регистра состояния: Устанавливается разряд **Z** при нулевом результате, разряд **S** при отрицательном результате, сбрасывается разряд **OF**.

Пример использования команды: **MOV %a0, %b3**

Значения операндов:

- начальное **a0 = 23_{16} b3 = 86_{16} RS = 10** (регистр состояния)
- конечное **a0 = 86_{16} b3 = 86_{16} RS = 4 (S = 1)**

Команда сложения

Мнемоника: **ADD dst,src** (ADD)

Код команды: **000100 | src | dst**

Действие команды: **dst + src ® dst**

Описание команды: Содержимое операнда **src** складывается с содержимым операнда **dst**, результат сложения пересылается в операнд **dst**.

Влияние на разряды регистра состояния: Устанавливается разряд **Z** при нулевом результате, разряд **S** при отрицательном результате, при арифметическом переполнении в результате выполнения команды устанавливается разряд **OF**, при переносе устанавливается разряд **C,** при тетрадном переносе устанавливается разряд **DC**.

Пример использования команды: **ADD %a0,%b3**

Начальные значения операндов: **a0 = $F9_{16}$ b3 = 98_{16} RS = 0**

Значения операндов:

- начальное **a0 = $F9_{16}$ b3 = 98_{16} RS = 0**
- конечное **a0 = 91_{16} b3 = 98_{16} RS = 25_{16}**

Команда вычитания

Мнемоника: **SUB dst,src** (SUBtract)

Код команды:

000011	src	dst

Действие команды: **dst - src ® dst**

Описание команды: Содержимое операнда **src** вычитается из содержимого операнда **dst**, результат вычитания пересылается в операнд **dst**.

Влияние на разряды регистра состояния: Устанавливается разряд **Z** при нулевом результате, разряд **S** при отрицательном результате, при арифметическом переполнении в результате выполнения команды устанавливается разряд **OF**, при переносе устанавливается разряд **C,** при тетрадном переносе устанавливается разряд **DC**.

Пример использования команды: **SUB %a0,%b3**

Значения операндов:

- начальное **a0 = 12_{16} b3 = 54_{16} RS = 7**
- конечное **a0 = BE_{16} b3 = 54_{16} RS = 35_{16}**

Команда сравнения

Мнемоника: **CMP dst,src** (CoMPare)

Код команды:

000010	src	dst

Действие команды: **dst - src S, Z, C ® RS**

Описание команды: Содержимое операнда **src** вычитается из содержимого операнда **dst**, по результатам вычитания формируются признаки в регистре состояния.

Влияние на разряды регистра состояния: Устанавливается разряд **Z** при нулевом результате, разряд **S** при отрицательном результате, при арифметическом переполнении в результате выполнения команды устанавливается разряд **OF**, при переносе устанавливается разряд **C,** при тетрадном переносе устанавливается разряд **DC**.

Пример использования команды: **CMP %a0,%b3**

Значения операндов:

- начальное **a0 = 33_{16} b3 = 33_{16} RS = 7**
- конечное **a0 = 33_{16} b3 = 33_{16} RS = 2 (Z = 1)**

Команда логического И

Мнемоника: **AND dst,src** (AND)

Код команды:

000101	src	dst

Действие команды: **dst.AND.src ® dst**

Описание команды: Над содержимым операнда **src** и содержимым операнда **dst** проводится логическая операция **И (AND)**, результат пересылается в операнд **dst**.

Влияние на разряды регистра состояния: Устанавливается разряд **Z** при нулевом результате, разряд **S** при отрицательном результате, разряды **C, OF** и **DC** сбрасываются.

Пример использования команды: **AND %a0,%b3**

Значения операндов:

- начальное **a0 = 12_{16} b3 = $F0_{16}$ RS = 17_{16}**
- конечное **a0 = 10_{16} b3 = $F0_{16}$ RS = 0**

Команда логического ИЛИ

Мнемоника: **OR dst,src** (OR)

Код команды:

000110	src	dst

Действие команды: **dst.OR.src ® dst**

Описание команды: Над содержимым операнда **src** и содержимым операнда **dst** проводится логическая операция **ИЛИ (OR)**, результат пересылается в операнд **dst**.

Влияние на разряды регистра состояния: Устанавливается разряд **Z** при нулевом результате, разряд **S** при отрицательном результате, разряды **C, OF** и **DC** сбрасываются.

Пример использования команды: **OR %a0,%b3**

Значения операндов:

- начальное **a0 = 12_{16} b3 = $F0_{16}$ RS = 37_{16}**
- конечное **a0 = $F2_{16}$ b3 = $F0_{16}$ RS = 4 (S = 1)**

Команда исключающего ИЛИ

Мнемоника: **XOR dst,src** (eXclusive OR)

Код команды:

000111	src	dst

Действие команды: **dst.XOR.src ® dst**

Описание команды: Над содержимым операнда **src** и содержимым операнда **dst** проводится логическая операция исключающее **ИЛИ (XOR)** , результат пересылается в операнд **dst**.

Влияние на разряды регистра состояния: Устанавливается разряд **Z** при нулевом результате, разряд **S** при отрицательном результате, разряды **C, OF** и **DC** сбрасываются.

Пример использования команды: **XOR %a0,%b3**

Значения операндов:

- начальное **a0 = 12_{16} b3 = $F0_{16}$ RS = 37_{16}**
- конечное **a0 = $E2_{16}$ b3 = $F0_{16}$ RS = 4 (S = 1)**

ЛИТЕРНЫЕ КОМАНДЫ

Литерные команды производят арифметические и логические операции с операндом **dst**, и литерой **const**, указанной в самой команде. Результат операции помещается в операнд, обозначаемый **dst** (приемник). Две команды, **MOVL** и **CMPL**, оперируют с полной 8-разрядной литерой **const**, команды **ADDL** и **SUBL** с 5-разрядной литерой **sconst**, старшие три разряда

заполняются нулями. Оставшиеся четыре команды в качестве операнда используют 4-разрядную (тетрадную) литеру **tconst**, местоположение которой в образуемой для литерного операнда **src** 8-разрядной литере определяется разрядом местоположения тетрадной литеры - **p**. При **0**-м разряде местоположения **p** тетрадная литера размещается в младшей тетраде полной литеры, а остальные разряды заполняются нулями. Установленный в единицу разряд местоположения **p** размещает тетрадную литеру в старшей тетраде, младшая тетрада полной литеры заполняется нулями.

*Формат литерных команд **MOVL** и **CMPL**:*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Код команды			Литера **const**								№ SR dst		Индекс dst		

Литера **const** задается в самой команде в диапазоне чисел от **-127** до **+127**.

*Формат литерных команд **ADDL** и **SUBL**:*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Код команды						Короткая литера **sconst**					№ SR dst		Индекс dst		

Литера **sconst** задается в самой команде в диапазоне чисел от **0** до **31**.

*Формат литерных команд **BIC**, **BIS**, **BTG** и **BTT**:*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Код команды						p	Литера **tconst**				№ SR dst		Индекс dst		

p – местоположение тетрадной литеры **tconst**.

Разряды	7	6	5	4	3	2	1	0
Полная литера	const							
p = 0	0	0	0	0	tconst			
p = 1	tconst				0	0	0	0

Команда пересылки литеры

Мнемоника: **MOVL dst,const** (MOVe Literal)

Код команды:

010	const	dst

Действие команды: **const ® dst**

Описание команды: Содержимое литеры **const** пересылается в операнд **dst**.

Влияние на разряды регистра состояния: Устанавливается разряд **Z** при нулевом результате, разряд **S** при отрицательном результате, сбрасывается разряд **OF**.

Пример использования команды: **MOVL %a0,0A5h**

Значения операнда **dst**:

- начальное **a0 = 23_{16} RS = 10_{16}**
- конечное **a0 = $A5_{16}$ RS = 4 (S = 1)**

Альтернативная ассемблерная команда: **CLR %a0 = MOVL %a0,0**

Команда сравнения с литерой

Мнемоника: **CMPL dst, const** (CoMPare with Literal)

Код команды:

011	const	dst

Действие команды: **dst - const** → **S**, **Z**, **C** → **RS**

Описание команды: Содержимое литеры **const** вычитается из содержимого операнда **dst**, по результатам вычитания формируются признаки в регистре состояния.

Влияние на разряды регистра состояния: Устанавливается разряд **Z** при нулевом результате, разряд **S** при отрицательном результате, при арифметическом переполнении в результате выполнения команды устанавливается разряд **OF**, при переносе устанавливается разряд **C,** при тетрадном переносе устанавливается разряд **DC**.

Пример использования команды: **CMPL %a0,23h**

Значения операнда **dst**:

- начальное **a0 = 23_{16} RS = 10_{16}**
- конечное **a0 = 23_{16} RS = 2 (Z = 1)**

Команда сложения с литерой

Мнемоника: **ADDL dst,sconst** (ADD Literal)

Код команды:

001100	sconst	dst

Действие команды: **dst + sconst → dst**

Описание команды: Содержимое короткой литеры **sconst** дополняется до байта нулями и складывается с содержимым операнда **dst**, результат сложения пересылается в операнд **dst**.

Влияние на разряды регистра состояния: Устанавливается разряд **Z** при нулевом результате, разряд **S** при отрицательном результате, при арифметическом переполнении в результате выполнения команды устанавливается разряд **OF**, при переносе устанавливается разряд **C,** при тетрадном переносе устанавливается разряд **DC**.

Пример использования команды: **ADDL %a0, 01Dh**

Значения операнда **dst**:

- начальное **a0 = $E3_{16}$ RS = 10_{16}**
- конечное **a0 = 0 RS = 23 (DC = Z = C = 1)**

Альтернативная ассемблерная команда: **INC %a0 = ADDL %a0,1**

Команда вычитания литеры

Мнемоника: **SUBL dst,sconst** (SUBtract Literal)

Код команды:

001011	sconst	dst

Действие команды: **dst - sconst → dst**

Описание команды: Содержимое операнда **src** вычитается из содержимого операнда **dst**, результат вычитания пересылается в операнд **dst**.

Влияние на разряды регистра состояния: Устанавливается разряд **Z** при нулевом результате, разряд **S** при отрицательном результате, при арифметическом переполнении в результате выполнения команды устанавливается разряд **OF**, при переносе устанавливается разряд **C,** при тетрадном переносе устанавливается разряд **DC**.

Пример использования команды: **SUBL %a0, 01Dh**

Значения операнда **dst**:

- начальное **a0 = $E3_{16}$ RS = 10_{16}**
- конечное **a0 = $C6_{16}$ RS = 21 (DC = S = 1)**

Альтернативная ассемблерная команда: **DEC %a0 = SUBL %a0,1**

Команда логического И с литерой (сброс разрядов)

Мнемоника: **BIC dst,tconst** (BIt Clear)

BICH - p = 1 BICL - p = 0

Код команды:

001010	p	tconst	dst

Действие команды: **dst.AND.NOT(tconst) ® dst**

Описание команды: В операнде **dst** сбрасываются разряды, установленные в литере **const**, составленной из тетрадной литеры **tconst** и разрядов нулей (проводится операция логического **И** над инверсией литеры **const** и операндом **dst**).

Влияние на разряды регистра состояния: Устанавливается разряд **Z** при нулевом результате, разряд **S** при отрицательном результате, разряды **C, OF** и **DC** сбрасываются.

Пример использования команды: **BICH %a0, 5h**

Значения операнда dst :

- начальное **a0 = FF_{16} RS = 37_{16}**
- конечное **a0 = AF_{16} RS = 4 (S = 1)**

Команда логического ИЛИ с литерой (установка разрядов)

Мнемоника: **BIS dst,tconst** (BIt Set)

BISH - p = 1 BISL - p = 0

Код команды:

001110	p	tconst	dst

Действие команды: **dst.OR.tconst ® dst**

Описание команды: В операнде **dst** устанавливаются разряды, установленные в литере **const**, составленной из тетрадной литеры **tconst** и разрядов нулей (проводится операция логического **ИЛИ** над литерой **const** и операндом **dst**).

Влияние на разряды регистра состояния: Устанавливается разряд **Z** при нулевом результате, разряд **S** при отрицательном результате, разряды **C, OF** и **DC** сбрасываются.

Пример использования команды: **BISL %a0, 8h**

Значения операнда **dst** :

- начальное **a0 = 22_{16} RS = 37_{16}**
- конечное **a0 = $2A_{16}$ RS = 0**

Команда исключающего ИЛИ с литерой (инверсии разрядов)

Мнемоника: **BTG dst,tconst** (Bit ToGgle)

BTGH - p = 1 BTGL - p = 0

Код команды:

001111	p	tconst	dst

Действие команды: **dst .XOR. tconst ® dst**

Описание команды: В операнде **dst** инвертируются разряды, установленные в литере **const**, составленной из тетрадной литеры **tconst** и разрядов нулей (проводится операция логического исключающего **ИЛИ** над литерой **const** и операндом **dst**).

Влияние на разряды регистра состояния: Устанавливается разряд **Z** при нулевом результате, разряд **S** при отрицательном результате, разряды **C, OF** и **DC** сбрасываются.

Пример использования команды: **BTGH %a0, 5h**

Значения операнда **dst**:

- начальное **a0 = 5_{16} RS = 37_{16}**
- конечное **a0 = 0 RS = 2 (Z = 1)**

Команда проверки разрядов

Мнемоника: **BTT dst,tconst** (BiT Test)

BTTH - p = 1 BTTL - p = 0

Код команды:

001101	p	tconst	dst

Действие команды: **dst .AND. tconst S,Z ® RS**

Описание команды: По наличию в операнде **dst** разрядов, установленных в литере **const**, составленной из тетрадной литеры **tconst** и разрядов нулей, формируются признаки в регистре состояния (проводится операция логического **И** над литерой **const** и операндом **dst**).

Влияние на разряды регистра состояния: Устанавливается разряд **Z** при нулевом результате, разряд **S** при отрицательном результате, разряды **C, OF** и **DC** сбрасываются.

Пример использования команды: **BTTL%a0, 3h**

Значения операнда **dst**:

- начальное **a0 = 1 RS = 37_{16}**
- конечное **a0 = 1 RS = 0**

ОДНООПЕРАНДНЫЕ КОМАНДЫ

Одноперандные команды производят арифметические или логические операции над операндом, адрес которого образуются из индекса, содержащегося в самой команде, и значения регистра адреса сегмента. Результат операции помещается в тот же операнд, обозначаемый **dst** (приемник).

Одноперандные команды имеют единый формат. При выполнении любой команды производится чтение операнда **dst** из памяти (как оперативной памяти данных, так и регистров периферийных устройств) и дальнейшая запись результата в память по адресу операнда **dst**.

Формат одноопе рандных команд:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Код команды											№ SR dst		Индекс dst		

Команда обмена тетрад

Мнемоника: **SWAP dst** (SWAP)

Код команды: **00000000001 | dst**

Действие команды: **dst(n)** ® **dst(n+4)** при n<4, **dst(n)** ® **dst (n - 4)** при n³4,

где n - номер разряда в **dst**, n = 0 ¸ 7

Описание команды: Тетрады в операнде **dst** меняются местами.

Влияние на разряды регистра состояния: Устанавливается разряд **Z** при нулевом результате, разряд **S** при отрицательном результате, разряды **C, OF** и **DC** сбрасываются.

Пример использования команды: **SWAP %a0**

Значения операнда **dst**:

- начальное **a0 = 19_{16} RS = 37_{16}**
- конечное **a0 = 91_{16} RS = 4 (S = 1)**

Команда смены знака

Мнемоника: **NEG dst** (NEGative)

Код команды: **00000000010 | dst**

Действие команды: **-dst** ® **dst**

Описание команды: Меняется знак операнда **dst**.

Влияние на разряды регистра состояния: Устанавливается разряд **Z** при нулевом результате, разряд **S** при отрицательном результате, при арифметическом переполнении в результате выполнения команды устанавливается разряд **OF**, при переносе устанавливается разряд **C**.

Пример использования команды: **NEG %a0**

Значения операнда **dst**:

- начальное **a0 = 3 RS = 37_{16}**
- конечное **a0 = FD_{16} RS = 25 (DC = S = C = 1)**

Команда инверсии всех разрядов

Мнемоника: **NOT dst** (NOT)

Код команды: **00000000011 | dst**

Действие команды: **NOT(dst) dst**

Описание команды: Разряды операнда **dst** инвертируются.

Влияние на разряды регистра состояния: Устанавливается разряд **Z** при нулевом результате,

разряд **S** при отрицательном результате, разряд **OF** сбрасывается.

Пример использования команды: **NOT %a0**

Значения операнда **dst**:

- начальное **a0 = F3$_{16}$ RS = 37$_{16}$**
- конечное **a0 = C$_{16}$ RS = 0**

Команда логического сдвига влево

Мнемоника: **SHL dst** (SHift Left)

Код команды: **00000000100 | dst**

Действие команды: **dst (n)** ® **dst (n+1)** при n < 7, 0 ® **dst (0)**, **dst (7)** ® **C**,

где n - номер разряда в **dst**, n = 0¸ 7

Описание команды: Содержимое операнда **dst** сдвигается влево на один разряд, младший разряд заполняется нулем, старший разряд заносится в **C**-разряд регистра состояния.

Влияние на разряды регистра состояния: Устанавливается разряд **Z** при нулевом результате, разряд **S** при отрицательном результате, разряд **C** принимает значение старшего разряда исходного операнда**,** разряд **DC** сбрасывается.

Пример использования команды: **SHL %a0**

Значения операнда **dst**:

- начальное **a0 = 5A$_{16}$ RS = 37$_{16}$**
- конечное **a0 = B4$_{16}$ RS = 14$_{16}$ (OF = 1, S = 1)**

Команда логического сдвига вправо

Мнемоника: **SHR dst** (SHift Right)

Код команды: **00000000101 | dst**

Действие команды: **dst(n+1)** ® **dst(n)** при n > 0, 0 ® **dst(7)**, **dst(0)** ® C,

где n - номер разряда в dst, n = 0¸ 7

Описание команды: Содержимое операнда **dst** сдвигается вправо на один разряд, старший разряд заполняется нулем, младший разряд заносится в **C**-разряд регистра состояния.

Влияние на разряды регистра состояния: Устанавливается разряд **Z** при нулевом результате, разряд **S** обнуляется, разряд **C** принимает значение младшего разряда исходного операнда**,** разряды **DC** и **OF** сбрасываются.

Пример использования команды: **SHR %a0**

Значения операнда **dst**:

- начальное **a0 = 5B$_{16}$ RS = 37$_{16}$**
- конечное **a0 = 2D$_{16}$ RS = 1 (C = 1)**

Команда арифметического сдвига вправо

Мнемоника: **SHRA dst** (SHift Right Arithmetical)

Код команды: **00000000110 | dst**

Действие команды: **dst(n+1)** ® **dst(n)** при n > 0, **dst(7)** ® **dst(7)**, **dst(0)** ® **C**,

где n - номер разряда в dst, n = 0¸ 7

Описание команды: Содержимое операнда **dst** сдвигается вправо на один разряд, старший разряд оставляет свое предыдущее значение, младший разряд заносится в **C**-разряд регистра состояния.

Влияние на разряды регистра состояния: Устанавливается разряд **Z** при нулевом результате, разряд **S** при отрицательном результате, разряд **C** принимает значение младшего разряда исходного операнда**,** разряды **DC** и **OF** сбрасываются.

Пример использования команды: **SHRA %a0**

Значения операнда **dst**:

- начальное **a0 = BC_{16} RS = 37_{16}**
- конечное **a0 = CE_{16} RS = 4 (S = 1)**

Команда циклического сдвига влево

Мнемоника: **RLC dst** (Roll Left Cyclic)

Код команды: **00000000111 | dst**

Действие команды: **dst(n)** ® **dst(n+1)** при n < 7, **C** ® **dst(0)**, **dst(7)** ® **C**,

где n - номер разряда в dst, n = 0¸ 7

Описание команды: Содержимое операнда **dst** сдвигается влево на один разряд, младший разряд заполняется значением разряда **C** регистра состояния, старший разряд заносится в **C**-разряд регистра состояния.

Влияние на разряды регистра состояния: Устанавливается разряд **Z** при нулевом результате, разряд **S** при отрицательном результате, разряд **C** принимает значение старшего разряда исходного операнда**,** разряд **DC** сбрасывается.

Пример использования команды: **RLC %a0**

Значения операнда **dst**:

- начальное **a0 = $5A_{16}$ RS = 37_{16} (C = 1)**
- конечное **a0 = $B5_{16}$ RS = 14_{16} (OF = 1, S = 1)**

Команда циклического сдвига вправо

Мнемоника: **RRC dst** (Roll Right Cyclic)

Код команды: **00000001000 | dst**

Действие команды: **dst(n+1)** ® **dst(n)** при n < 7, **C** ® **dst(7)**, **dst(0)** ® **C**,

где n - номер разряда в dst, n = 0¸ 7

Описание команды: Содержимое операнда **dst** сдвигается вправо на один разряд, старший

разряд заполняется значением разряда **C** регистра состояния, младший разряд заносится в **C**-разряд регистра состояния.

Влияние на разряды регистра состояния: Устанавливается разряд **Z** при нулевом результате, разряд **S** при отрицательном результате, разряд **C** принимает значение старшего разряда исходного операнда**,** разряд **DC** сбрасывается.

Пример использования команды: **RLC %a0**

Значения операнда **dst**:

- начальное **a0 = $5A_{16}$ RS = 37_{16} (C = 1)**
- конечное **a0 = $B5_{16}$ RS = 14_{16} (OF = 1, S = 1)**

Команда сложения с переносом

Мнемоника: **ADC dst** (ADD Carry)

Код команды: **00000001001 | dst**

Действие команды: **dst + C ® dst**

Описание команды: Содержимое **C**-разряда регистра состояния складывается с содержимым операнда **dst**, результат сложения пересылается в операнд **dst**.

Влияние на разряды регистра состояния: Устанавливается разряд **Z** при нулевом результате, разряд **S** при отрицательном результате, при арифметическом переполнении в результате выполнения команды устанавливается разряд **OF**, при переносе устанавливается разряд **C,** при тетрадном переносе устанавливается разряд **DC**.

Пример использования команды: **ADC %a0**

Значения операнда **dst**:

- начальное **a0 = $5A_{16}$ RS = 37_{16}**
- конечное **a0 = $5B_{16}$ RS = 0**

Команда вычитания переноса

Мнемоника: **SBC dst** (SuBtract Carry)

Код команды: **00000001010 | dst**

Действие команды: **dst - C ® dst**

Описание команды: Содержимое **C**-разряда регистра состояния вычитается из содержимого операнда **dst**, результат вычитания пересылается в операнд **dst**.

Влияние на разряды регистра состояния: Устанавливается разряд **Z** при нулевом результате, разряд **S** при отрицательном результате, при арифметическом переполнении в результате выполнения команды устанавливается разряд **OF**, при переносе устанавливается разряд **C,** при тетрадном переносе устанавливается разряд **DC**.

Пример использования команды: **SBC %a0**

Значения операнда **dst**:

- начальное **a0 = $5A_{16}$ RS = 37_{16}**
- конечное **a0 = $5B_{16}$ RS = 0**

КОМАНДЫ РАБОТЫ СО СЛУЖЕБНЫМИ РЕГИСТРАМИ ПРОЦЕССОРА И РЕГИСТРАМИ СОСТОЯНИЯ

Команда загрузки служебных регистров процессора **LDR** предназначена для безусловной установки необходимых значений регистров адресов сегментов, для тех случаев, когда требуется адресация к новым переменным или регистрам ввода вывода, не охватываемым текущими сегментами. Этой командой загружаются также регистры косвенной адресации и устанавливаются режимы их работы. Служебные регистры могут сохраняться в памяти данных и восстанавливаться из нее при помощи команд **MFPR** и **MTPR**. Регистры адресов сегментов и регистры косвенной адресации доступны по номерам служебных регистров, указываемым в командах **LDR**, **MFPR** и **MTPR**. Для сохранения служебных регистров процессора при прерываниях и переходах к подпрограммам предназначен стек, размером 16x8. При прерываниях в стеке данных происходит автоматическое сохранение регистра состояния процессора. При выполнении команды возврата из прерывания сохраненное значение автоматически переписывается в регистр состояния.

Для записи и чтения из стека значений служебных регистров имеются команды **PUSH** – записи в стек и **POP** – чтения из стека. Указатель глубины заполнения увеличивает значение **DSP**, а команда – уменьшает. Ситуации, когда при значении **DSP** = 15 подается команда **PUSH**, или **DSP** = 0 подается команда **POP**, приводят к прерыванию по ошибке стеков. Команды работы с регистром состояний процессора **SST** и **CST** дают возможность установить или сбросить необходимые разряды в регистре состояний, указанные маской в теле этих команд.

Каждая из команд этого раздела имеет свой формат.

Команда загрузки служебных регистров

Формат команды:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Код команды					Константа const								№ регистра		

Мнемоника: **LDR reg,const** (LoaD service Register)

Код команды:

00100	**const**	**reg**

Действие команды: **const ® reg**

Описание команды: В служебный регистр **reg** записывается значение **const**, указанное в команде.

Влияние на разряды регистра состояния: Регистр состояния не изменяется.

Пример использования команды: **LDR #2, 8h**

(**Примечание**: в ассемблере для **SR0 ¸ SR3** указывается адрес начала сегмента в памяти, а не значение константы).

Значения операнда **reg**:

- начальное **SR2 = 3**
- конечное **SR2 = 1**

Команда чтения служебных регистров

Формат команды:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Код команды								№ регистра			№ SR dst		Индекс dst		

Мнемоника: **MFPR dst,reg** (Move From Processor service Register)

Код команды: **00000011 | reg | dst**

Действие команды: **reg ® dst**

Описание команды: Служебный регистр **reg** пересылается в операнд **dst**.

Влияние на разряды регистра состояния: Регистр состояния не изменяется.

Пример использования команды: **MFPR %a0,#2**

Значения операнда **reg**:

- начальное **a0 =** $\mathbf{5A_{16}}$ **SR2 = 7**
- конечное **a0 = 7 SR2 = 7**

Команда записи в служебные регистры

Формат команды:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Код команды								№ регистра			№ SR dst		Индекс dst		

Мнемоника: **MTPR reg,src** (Move To Processor service Register)

Код команды: **00000010 | reg | src**

Действие команды: **src ® reg**

Описание команды: Содержимое операнда **src** пересылается в служебный регистр **reg**.

Влияние на разряды регистра состояния: Регистр состояния не изменяется.

Пример использования команды: **MTPR #2, %a0**

Значения операнда **reg**:

- начальное **a0 =** $\mathbf{5A_{16}}$ **SR2 = 7**
- конечное **a0 =** $\mathbf{5A_{16}}$ **SR2 =** $\mathbf{5A_{16}}$

Команда записи в стек служебных регистров

Формат команды:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Код команды												№ регистра			

Мнемоника: **PUSH reg** (PUSH service register)

Код команды: **0000000000010 | reg**

Действие команды: **reg ® data stack**, **DSP + 1 ® DSP**

Описание команды: Значение указанного служебного регистра процессора заносится в стек

данных, указатель стека данных увеличивает свое значение на единицу.

Влияние на разряды регистра состояния: Регистр состояния не изменяется.

Команда восстановления из стека служебных регистров процессора

Формат команды:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Код команды												№ регистра			

Мнемоника: **POP reg,src** (POP service register)

Код команды: **0000000000011 reg**

Действие команды: **data stack ® reg, DSP - 1 ® DSP**

Описание команды: Значение указанного служебного регистра процессора восстанавливается из стека данных, указатель стека данных уменьшает свое значение на единицу.

Влияние на разряды регистра состояния: Регистр состояния не изменяется.

Команда установки разрядов регистра состояния процессора

Формат команды:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Код команды												Разряды SR			

Мнемоника: **SST mask** (Set Status bits)

Код команды: **000000011000 mask**

Действие команды: **if mask(n) = 1 then RS(n) = 1** для n = 0 ¸ 3

Описание команды: В регистре состояний процессора устанавливаются те разряды, которые установлены в константе **mask**. Команда устанавливает только младшие четыре разряда.

Влияние на разряды регистра состояния: Устанавливаются разряды **IE**, **S**, **Z** и **C** в зависимости от значения константы **mask**.

Пример использования команды: **SST 5**

Значения регистра состояния **RS**:

- начальное **RS = 0**
- конечное **RS = 5 (S = 1, C = 1)**

Команда сброса разрядов регистра состояния процессора

Формат команды:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Код команды												Разряды SR			

Мнемоника: **CST mask** (Clear Status bits)

Код команды: **000000011100 mask**

Действие команды: **if mask(n) = 1 then RS(n) = 0** для n = 0 ¸ 3

Описание команды: В регистре состояний процессора сбрасываются те разряды, которые установлены в константе **mask**. Команда сбрасывает только младшие четыре разряда

Влияние на разряды регистра состояния: Сбрасываются разряды **IE**, **S**, **Z** и **C** в зависимости от значения константы **mask**.

Пример использования команды: **CST 9**

Значения регистра состояния **RS**:

- начальное **RS = $1F_{16}$**
- конечное **RS = 16_{16} (OF = 1, S = 1, Z = 1)**

Команда проверки арифметического переполнения

Формат команды:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Код команды															

Мнемоника: **TOF** (Test OverFlow bit)

Код команды: **0000000000000100**

Действие команды: **OF Z**

Описание команды: Этой командой значение **OF**-разряда регистра состояния переписывается в **Z**-разряд.

Команда проверки тетрадного переноса

Формат команды:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Код команды															

Мнемоника: **TDC** (Test Digit Carry bit)

Код команды: **0000000000000101**

Действие команды: **DC ® Z**

Описание команды: Этой командой значение **DC**-разряда регистра состояния переписывается в **Z**-разряд.

КОМАНДЫ ПЕРЕДАЧИ УПРАВЛЕНИЯ

Команды передачи управления производят передачу управления программе, находящейся по адресу, указанному в младших десяти разрядах команды перехода, либо безусловно, либо по одному из условий перехода. При переходе к подпрограмме в стек команд записывается адрес возврата. При возврате из подпрограммы или прерывания адрес, восстановленный из стека адресов возврата (стека команд), записывается в счетчик команд. Указатель глубины заполнения стека команд **ISP** (Instruction Stack Pointer) может меняться от 0 до 7. Команда перехода к подпрограмме или прерывание увеличивает значение **ISP**, а команды возврата из подпрограммы или прерывания – уменьшают. Ситуации, когда при значении **ISP** = 7 подается

команда перехода к подпрограмме или происходит прерывание, или при **ISP** = 0 подается команды **RTS** или **RTI**, приводят к прерыванию по ошибке стеков.

Две команды, **IJMP** и **IJSR**, передают управление программам, адрес которых записан в косвенном регистре **IR1**, вне зависимости от режима его работы.

Формат команд перехода (**JMP, JSR, JNZ, JNE, JZ, JEQ, JNS, JS, JNC, JC)**:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Код команды				Резерв		Адрес перехода address									

Формат команд возврата и косвенных переходов (**IJMP, IJSR, RTS, RTSC, RTI)**:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Код команды															

Команда перехода к подпрограмме

Мнемоника: **JSR address** (Jump to SubRoutine)

Код команды: **1001** | **address**

Действие команды: **PC** ® **istack** (стек команд), **address PC**, **ISP + 1 ISP**

Описание команды: Адрес следующей команды записывается в стек команд, а в счетчик команд записывается новый адрес, указанный в команде. Указатель стека команд сдвигается на одну позицию в сторону увеличения адреса.

Команда безусловного перехода

Мнемоника: **JMP address** (JuMP)

Код команды: **1000** | **address**

Действие команды: **address** ® **PC** (счетчик команд)

Описание команды: В счетчик команд **PC** записывается новый адрес, указанный в команде.

Команда перехода по установленному S-разряду

Мнемоника: **JS address** (Jump if Sign bit is set)

Код команды: **1101** | **address**

Действие команды: **address** ® **PC if S = 1**

Описание команды: Если разряд **S** в регистре состояния установлен, в счетчик команд записывается новый адрес, указанный в команде, если сброшен – будет выполняться следующая за переходом команда. То есть переход осуществляется при отрицательном результате.

Пример использования команды:

```
ADDL    %a0,#2
JS      minus
```

Переход произойдет при отрицательном результате.

Команда перехода по сброшенному S-разряду

Мнемоника: **JNS address** (Jump if No Sign bit)

Код команды:

1100	address

Действие команды: **address ® PC if S = 0**

Описание команды: Если разряд **S** в регистре состояния не установлен, в счетчик команд записывается новый адрес, указанный в команде, если установлен – будет выполняться следующая за переходом команда. То есть переход осуществляется при положительном или нулевом результате.

Пример использования команды:

```
ADDL    %a0,#2
JS      plus
```

Переход произойдет при положительном результате**.**

Команда перехода по установленному Z-разряду

Мнемоника: **JZ address** (Jump if Zero bit is set)

JEQ address (Jump if EQual)

Код команды:

1010	address

Действие команды: **address ® PC if Z = 1**

Описание команды: Если разряд **Z** в регистре состояния установлен, в счетчик команд записывается новый адрес, указанный в команде, если сброшен – будет выполняться следующая за переходом команда. При использовании этой команды после команды сравнения переход произойдет при равенстве сравниваемых операндов. При использовании после команды проверки разрядов переход произойдет в случае отсутствия требуемых разрядов.

Пример использования команды:

```
CMP     %a0,%b4
JEQ     equal
```

Переход произойдет, если операнд **%a0** равен операнду **%b4.**

Команда перехода по сброшенному Z-разряду

Мнемоника: **JNZ address** (Jump if No Zero bit)

JNE address (Jump if No Equal)

Код команды:

1011	address

Действие команды: address ® PC if Z = 0

Описание команды: Если разряд **Z** в регистре состояния не установлен, в счетчик команд записывается новый адрес, указанный в команде, если установлен – будет выполняться следующая за переходом команда. При использовании этой команды после команды сравнения переход произойдет при неравенстве сравниваемых операндов. При использовании после команды проверки разрядов переход произойдет в случае присутствия требуемых разрядов.

Пример использования команды:

```
BTTL    %a0,#2
JNZ     bitset
```

Переход произойдет, если в операнде **%a0** установлен разряд **#2.**

Команда перехода по установленному С-разряду

Мнемоника: **JC address** (Jump if Carry bit is set)

Код команды:

1111	address

Действие команды: **address ® PC if C = 1**

Описание команды: Если разряд **C** в регистре состояния установлен, в счетчик команд записывается новый адрес, указанный в команде, если сброшен – будет выполняться следующая за переходом команда.

Пример использования команды:

```
SHL   %a0
JC    cset
```

Переход произойдет, если операнд **%a0** имел отрицательное значение до выполнения команды сдвига влево.

Команда перехода по сброшенному С-разряду

Мнемоника: **JNC address** (Jump if No Carry bit)

Код команды:

1110	address

Действие команды: **address ® PC if C = 0**

Описание команды: Если разряд **C** в регистре состояния не установлен, в счетчик команд записывается новый адрес, указанный в команде, если установлен – будет выполняться следующая за переходом команда.

Пример использования команды:

```
SHL   %a0
JNC   cclear
```

Переход произойдет, если операнд **%a0** имел положительное значение до выполнения команды сдвига влево**.**

Команда безусловного косвенного перехода

Мнемоника: **IJMP** (Indirect JuMP)

Код команды:

0000000000000011

Действие команды: **IR1 ® PC**

Описание команды: В счетчик команд **PC** записывается новый адрес, который содержится в младших 10 разрядах регистра косвенной адресации **IR1**. Команда может использоваться для выполнения вычисляемого перехода. Вычисленное значение переписывается в **IR1**, а затем выполняется косвенный переход.

Команда косвенного перехода к подпрограмме

Мнемоника: **IJSR** (Indirect Jump to SubRoutine)

Код команды: **0000000000000111**

Действие команды: **PC** ® **istack** (стек команд), **IR1** ® **PC**, **ISP + 1 ISP**

Описание команды: Адрес следующей команды записывается в стек команд, а в счетчик команд записывается новый адрес, который содержится в младших 10 разрядах регистра косвенной адресации **IR1**. Указатель стека команд сдвигается на одну позицию в сторону увеличения адреса. Команда может использоваться для выполнения вычисляемого перехода к подпрограмме. Вычисленное значение переписывается в **IR1**, а затем выполняется косвенный переход к подпрограмме.

Команда возврата из подпрограммы

Мнемоника: **RTS** (ReTurn from Subroutine)

Код команды: **0000000000001100**

Действие команды: **istack** ® **PC**, **ISP - 1** ® **ISP**

Описание команды: В счетчик команд переписывается адрес возврата из стека команд. Указатель стека команд сдвигается на одну позицию в сторону уменьшения адреса.

Команда возврата из подпрограммы с изменением С-разряда

Мнемоника: RTSC c (ReTurn from Subroutine with C)

Код команды: **000000000000111c**

Действие команды: **istack PC, c ® RS(0), ISP - 1 ® ISP**

Описание команды: В счетчик команд переписывается адрес возврата из стека команд, значение бита "**c**" из команды переписывается в **C**-разряд регистра состояния. Указатель стека команд сдвигается на одну позицию в сторону уменьшения адреса. В **C**-разряде удобно передавать информацию о результате выполнения завершенной подпрограммы. Например при успешном завершении подпрограммы **C** = 0, при ошибке в выполнении подпрограммы **C** = 1.

Команда возврата из прерывания

Мнемоника: **RTI** (ReTurn from Interrupt)

Код команды: **0000000000001101**

Действие команды: **istack PC**, **data stack** ® **RS**, **ISP- 1** ® **ISP**,
DSP - 1 ® **DSP**

Описание команды: В счетчик команд переписывается адрес возврата из стека команд, а в регистр состояния – содержимое верхней ячейки стека данных. Указатели стека команд и данных сдвигаются на одну позицию в сторону уменьшения адреса.

СПЕЦИАЛЬНЫЕ КОМАНДЫ

Специальные команды относятся к классу команд, не вписывающихся в предыдущие разделы, и предназначены в основном для управления определенными состояниями процессора.

Формат специальных команд:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Код команды															

Команда отсутствия операции

Мнемоника: **NOP** (No OPeration)

Код команды: **0000000000000000**

Описание команды: Пустая команда, просто пропуск по времени исполнительных фаз.

Команда ожидания

Мнемоника: **WAIT** (WAIT)

Код команды: **0000000000000001**

Действие команды: **RS(3) = 1**

Описание команды: По команде ожидания прекращается выборка последующих команд, микроконтроллер переходит в режим ожидания прерываний. Генератор тактовой частоты продолжает работать. В случае прихода сигнала прерывания процессор приступает к выполнению процедуры прерывания.

Команда останова

Мнемоника: **STOP** (STOP)

Код команды: **0000000000001000**

Описание команды: По команде ожидания прекращается выборка последующих команд, микроконтроллер переходит в режим ожидания прерываний. Генератор тактовой частоты останавливается. Устанавливается разряд разрешения прерывания в регистре состояния. В случае прихода сигнала прерывания процессор приступает к выполнению процедуры прерывания только после того, как счетчик времени установки тактового генератора (счетчик задержки начального пуска) завершит отсчет полного интервала.

Команда сброса

Мнемоника: **RESET** (RESET)

Код команды: **0000000000000010**

Действие команды: **ISP = 0**, **DSP = 0**

Описание команды: По этой команде происходит установка указателей стеков команд и данных в начальное положение и сбрасывается сигнал прерывания по переполнению сте-

ков команд или данных. Рекомендуется использовать эту команду в программе обработки прерывания по ошибке работы со стеками. Следует отметить, что данную команду нельзя ставить самой первой в программе обработки прерывания, т.к. при этом не гарантируется сброс указателей стеков.

Команда прогона стека команд

Мнемоника: **SKSP** (SKip Stack Pointer)

Код команды: **0000000000000110**

Действие команды: **ISP - 1 ISP**

Описание команды: По этой команде указатель стека команд сдвигается на одну позицию в сторону уменьшения адреса. Можно использовать эту команду для изменения уровня вложенности подпрограмм.

Пример использования команды:

Выполнялся следующий отрезок программы.

```
            JSR subprog1      ; Вызов 1-й подпрограммы
retpoint1:  ...               ; Точка возврата 1
            ...
subprog1:   ...               ; Начало 1-й подпрограммы
            JSR subprog2      ; Вызов 2-й подпрограммы
retpoint1:  ...               ; Точка возврата 2
subprog2:   ...               ; Начало 1-й подпрограммы
            SKSP                    ; Прогон стека команд
            RTS               ; Возврат в точку возврата 1
```

Если бы не подавалась команда **SKSP**, по команде возврата из подпрограммы произошла бы передача управления на точку возврата 2. А так передача управления произошла на точку возврата, минуя возврат в первую подпрограмму.