5.3 Пример описания инструкции

Пример описания ниже сделан для вымышленной инструкции FOO. Следующий пример инструкции создан для демонстрации областей таблицы (синтаксис, операнды, действие и т.д.) и используется для описания инструкций представленных в **части 5.4 “Описание инструкций”**.

FOO	Поле заголовка описывает то, что делает инструкция
Синтаксис:	Это поле состоит из необязательной метки, мнемоники инструкции, любых необязательных расширений, которые существуют для инструкции и операндов. Большинство инструкций поддерживают более одного варианта операнда, чтобы поддержать различные способы адресации dsPIC30F/dsPIC33F. В этих случаях, все возможные операнды инструкции перечислены друг под другом (как в случае op2a, op2b и op2c указанных выше). Необязательные операнды включены в фигурные скобки.
Операнды:	Это поле описывает набор значений, которые каждый из операндов может принимать. Операндами могут быть регистры аккумулятора, файловые регистры, литеральные константы (со знаком или без знака), или рабочие регистры.
Действие:	Это поле описывает действие, выполняемое инструкцией.
Влияет на флаги:	Это поле описывает, на какие биты регистра Status воздействует инструкция. Биты состояния перечислены в порядке убывания разрядной позиции.
Код:	Это поле показывает, как закодированы разряды инструкции. Индивидуальные разрядные поля объясняются в поле Описание, и полные подробности кодирования приводятся в Таблице 5.2.
Описание:	Это поле подробно описывает операцию, выполняемую инструкцией. Ключ кодирования битов также обеспечивается.
Слова:	Это поле содержит число слов программы, которые используются при сохранении инструкции в памяти.
Циклы:	Это поле содержит число циклов инструкции, требуемых для её выполнения.
Примеры:	Это поле содержит примеры, которые демонстрируют, как команда работает. Обеспечиваются снимки регистра “Прежде” и “После того, которые позволяют пользователю ясно понимать то, какую операцию инструкция исполняет.

5.4 Описания инструкций

ADD **Прибавить f к WREG**

Синтаксис: {метка:} ADD{.B} f {,WREG}

Операнды: f ∈ [0 ... 8191]

Действие: (f) + (WREG) → место назначение, определённое D

Влияет на флаги: DC, N, OV, Z, C

Код:

1011	0100	0BDf	ffff	ffff	ffff

Описание: Прибавляет содержимое рабочего, по умолчанию, регистра WREG к содержимому файлового регистра, и размещает результат в регистр места назначения. Необязательный операнд WREG определяет регистр места назначения. Если определён WREG, результат сохраняется в WREG. Если WREG не определен, результат сохранен в файловый регистр.

Бит ‘B’ выбирает байтную или словную операцию (‘0’ для слова, ‘1’ для байта).
Бит ‘D’ выбирает место назначения (‘0’ для WREG, ‘1’ для файлового регистра).
Бит ‘f’ выбирает адрес файлового регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение байтной операции перед словной операцией. Вы можете использовать расширение .W для обозначения словной операции, но это не обязательно.
2: WREG установлен на рабочий регистр W0.

Слова: 1

Циклы: 1

Пример 1:
```
ADD.B       RAM100      ; Прибавить WREG к RAM100 (Байтный режим)
```

	Перед инструкцией		После инструкции	
WREG	CC80	WREG	CC80	
RAM100	FFC0	RAM100	FF40	
SR	0000	SR	0005	(OV, C = 1)

Пример 2:
```
ADD         RAM200,WREG ; Прибавить RAM200 к WREG (Словный режим)
```

	Перед инструкцией		После инструкции	
WREG	CC80	WREG	CC80	
RAM200	FFC0	RAM200	FFC0	
SR	0000	SR	0001	(C = 1)

ADD **Прибавить литерал к Wn**

Синтаксис: {метка:} ADD{.B} #lit10, Wn

Операнды: lit10 ∈ [0 ... 255] для байтной операции
lit10 ∈ [0 ... 1023] для словной операции
Wn ∈ [W0 ... W15]

Действие: lit10 + (Wn) → Wn

Влияет на флаги: DC, N, OV, Z, C

Код:

1011	0000	0Bkk	kkkk	kkkk	dddd

Описание: Прибавляет 10-битный литеральный операнд без знака к содержимому рабочего регистра Wn, и возвращает результат в рабочий регистр Wn.

Бит ‘B’ выбирает байтную или словную операцию (‘0’ для слова, ‘1’ для байта).
Биты ‘k’ определяют литеральный операнд.
Биты ‘d’ определяют адрес рабочего регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение байтной операции перед словной операцией. Вы можете использовать расширение .W для обозначения словной операции, но это не обязательно.
2: Для байтных операций, литерал может быть определён как беззнаковое значение [0:255]. Смотреть часть 4.6 “Использование 10-битных литеральных операндов” для информации на использование 10-битных литеральных операндов в байтном режиме.

Слова: 1

Циклы: 1

Пример 1:

```
ADD.B       #0xFF, W7   ; Прибавить -1 к W7 (байтный режим)
```

	Перед инструкцией		После инструкции	
W7	12C0	W7	12BF	
SR	0000	SR	0009	(N, C = 1)

Пример 2:

```
ADD         #0xFF, W1   ; Прибавить 255 к W1 (словный режим)
```

	Перед инструкцией		После инструкции
W1	12C0	W1	13BF
SR	0000	SR	0000

ADD **Прибавить Wb к короткому литералу**

Синтаксис: {метка:} ADD{.B} Wb #lit5, Wd
[Wd]
[Wd++]
[Wd--]
[++Wd]
[--Wd]

Операнды: Wb ∈ [W0 ... W15]
lit5 ∈ [0 ... 31]
Wd ∈ [W0 ... W15]

Действие: (Wb) + lit5 → Wd

Влияет на флаги: DC, N, OV, Z, C

Код:

0100	0www	wBqq	qddd	d11k	kkkk

Описание: Прибавляет содержимое базового регистра Wb к без знаковому короткому 5-битному литеральному операнду, и размещает результат в регистре назначения Wd. Для Wd может быть использован регистр направления адресации.

Биты ‘w’ выбирают адрес базового регистра.
Бит ‘B’ выбирает байтную или словную операцию (‘0’ для слова, ‘1’ для байта).
Биты ‘q’ выбирают режим адресации места назначения.
Биты ‘d’ выбирают регистр назначения.
Биты ‘k’ предоставляют литеральный операнд, пяти-битное целое число.

Примечание: Расширение .B в инструкции обозначает предпочтение байтной операции перед словной операцией. Вы можете использовать расширение .W для обозначения словной операции, но это не обязательно.

Слова: 1

Циклы: 1

Пример 1:

```
ADD.B       W0, #0x1F, W7          ; Прибавить W0 и 31 (Байтный режим)
                                   ; Запомнить результат в W7
```

	Перед инструкцией		После инструкции	
W0	2290	W0	2290	
W7	12C0	W7	12AF	
SR	0000	SR	0008	(N = 1)

Пример 2:

```
ADD         W3, #0x6, [--W4]       ; Прибавить W3 и 6 (Словный режим)
                                   ; Запомнить результат в [--W4]
```

	Перед инструкцией		После инструкции	
W3	6006	W3	6006	
W4	1000	W4	0FFE	
Данные 0FFE	DDEE	Данные 0FFE	600C	
Данные 1000	DDEE	Данные 1000	DDEE	
SR	0000	SR	0000	

ADD **Прибавить Wb к Ws**

Синтаксис:	{метка:}	ADD{.B}	Wb	Ws, [Ws], [Ws++], [Ws--], [++Ws], [--Ws],	Wd [Wd] [Wd++] [Wd--] [++Wd] [--Wd]

Операнды: Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

Действие: (Wb) + (Ws) → Wd

Влияет на флаги: DC, N, OV, Z, C

Код:

0100	0www	wBqq	qddd	dppp	ssss

Описание: Складывает содержимое регистра источника Ws и содержимое базового регистра Wb, и размещает результат в регистр назначения Wd. Регистр направления адресации должен быть использован для Wb. Регистр прямой или косвенной адресации может быть использован для Ws и Wd.

Биты ‘w’ выбирают адрес базового регистра.
Бит ‘B’ выбирает байтную или словную операцию (‘0’ для слова, ‘1’ для байта).
Биты ‘q’ выбирают режим адреса назначения.
Биты ‘d’ выбирают регистр назначения.
Биты ‘p’ выбирают режим адреса источника.
Биты ‘s’ выбирают регистр источник.

Примечание: Расширение .B в инструкции обозначает предпочтение байтной операции перед словной операцией. Вы можете использовать расширение .W для обозначения словной операции, но это не обязательно.

Слова: 1

Циклы: 1

Пример 1:

```
ADD.B       W5, W6, W7   ; Прибавить W5 к W6, запомнить результат в W7
                         ; (Байтный режим)
```

	Перед инструкцией		После инструкции
W5	AB00	W5	AB00
W6	0030	W6	0030
W7	FFFF	W7	FF30
SR	0000	SR	0000

Пример 2:

```
ADD         W5, W6, W7   ; Прибавить W5 к W6, запомнить результат в W7
                         ; (Словный режим)
```

	Перед инструкцией		После инструкции	
W5	AB00	W5	AB00	
W6	0030	W6	0030	
W7	FFFF	W7	FF30	
SR	0000	SR	0008	(N = 1)

ADD **Сложить аккумуляторы**

Синтаксис: {метка:} ADD Acc

Операнды: Acc ∈ [A,B]

Действие: Если (Acc = A):
(ACCA) + (ACCB) → ACCA
Иначе:
(ACCA) + (ACCB) → ACCB

Влияет на флаги: OA, OB, OAB, SA, SB, SAB

Код:

1100	1011	A000	0000	0000	0000

Описание: Складывает содержимое аккумулятора A с содержимым аккумулятора B и помещает результат в выбранный аккумулятор. Эта инструкция выполняет 40-битное дополнение.

Бит ‘A’ определяет аккумулятор, который является назначением.

Слова: 1

Циклы: 1

Пример 1:

```
ADD     A       ; Сложить ACCB к ACCA
```

	Перед инструкцией	После инструкции
ACCA	00 0022 3300	00 1855 7858
ACCB	00 1833 4558	00 1833 4558
SR	0000	0000

Пример 2:

```
ADD     B       ; Сложить ACCA к ACCB
                ; При условии, что разрешен режим супер насыщения
                ; (ACCSAT = 1, SATA = 1, SATB = 1)
```

	Перед инструкцией	После инструкции	
ACCA	00 E111 2222	00 E111 2222	
ACCB	00 7654 3210	01 5765 5432	
SR	0000	4800	(OB, OAB = 1)

ADD **16-битное знаковое число прибавить к аккумулятору**

Синтаксис:	{метка:}	ADD	Ws, [Ws], [Ws++], [Ws--], [--Ws], [++Ws], [Ws+Wb],	{#Slit4,}	Acc

Операнды: Ws ∈ [W0 ... W15]
Wb ∈ [W0 ... W15]
Slit4 ∈ [-8 ... +7]
Acc ∈ [A,B]

Действие: $Shift_{Slit4}$(Расширять(Ws)) + (Acc) → Acc

Влияет на флаги: OA, OB, OAB, SA, SB, SAB

Код:

1100	1001	Awww	wrrr	rggg	ssss

Описание: Складывает 16-битное значение определённое рабочим регистром источником к наиболее значимому слову выбранного аккумулятора. Операнд источник может напрямую определять содержимое рабочего регистра или эффективный адрес. Определённое значение будет добавлено к наиболее значимому слову, знаково расширив и заполнив нулями операнд источник перед операцией. Значение добавленное в аккумулятор может также быть сдвинуто 4-битным знаковым литералом, прежде чем суммирование будет сделано.

Бит ‘A’ определяет аккумулятор места назначения.
Биты ‘w’ определяют смещение регистра Wb.
Биты ‘r’ кодируют дополнительный сдвиг.
Биты ‘g’ выбирают режим адресации источника.
Биты ‘s’ определяют регистр источник Ws.

Примечание: Положительные значения операнда Slit4 означают правый арифметический сдвиг и отрицательные значения операнда Slit4 означают левый арифметический сдвиг. Slit4 не воздействует на содержимое регистра источника.

Слова: 1

Циклы: 1

Пример 1:

```
ADD     W0, #2, A   ; Прибавить W0, сдвинутый вправо на 2, к ACCA
```

	Перед инструкцией	После инструкции
W0	8000	8000
ACCA	00 7000 0000	00 5000 0000
SR	0000	0000

Пример 2:

```
ADD     [W5++], A   ; Добавить эффективное значение W5 к ACCA
                    ; Пост-инкремент W5
```

	Перед инструкцией	После инструкции
W5	2000	2002
ACCA	00 0067 2345	00 5067 2345
Data 2000	5000	5000
SR	0000	4800

ADDC **Прибавить f к WREG с переносом**

Синтаксис: {метка:} ADDC{.B} f {,WREG}

Операнды: f ∈ [0 ... 8191]

Действие: (f) + (WREG) + (C) → место назначение, определённое D

Влияет на флаги: DC, N, OV, Z, C

Код:

1011	0100	1BDf	ffff	ffff	ffff

Описание: Складывает содержимое рабочего регистра по умолчанию WREG, содержимое файлового регистра и бит переноса, и помещает результат в регистр места назначения. Опционально WREG операнд определяется регистром места назначения. Если определён WREG, результат сохраняется в WREG. Если WREG не определён, результат запоминается в файловый регистр.

Бит ‘B’ выбирает байтный или словный операнд (‘0’ для слова, ‘1’ для байта).
Бит ‘D’ выбирает место назначения (‘0’ для WREG, ‘1’ для файлового регистра).
Биты ‘f’ выбирают адрес файлового регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

2: WREG установлен на рабочий регистр W0.

3: Флаг Z является “липким” для ADDC, CPB, SUBB и SUBBR. Эти инструкции могут только очищать Z.

Слова: 1

Циклы: 1

Пример 1:

```
ADDC.B      RAM100      ; Сложить WREG и бит C в RAM100
                        ; (Байтный режим)
```

	Перед инструкцией			После инструкции	
WREG	CC60		WREG	CC60	
RAM100	8006		RAM100	8067	
SR	0001	(C=1)	SR	0000	

Пример 2:

```
ADDC        RAM200, WREG        ; Сложить RAM200 и бит C в WREG
                                ; (Словный режим)
```

	Перед инструкцией			После инструкции	
WREG	5600		WREG	8A01	
RAM200	3400		RAM200	3400	
SR	0001	(C=1)	SR	000C	(N, OV = 1)

ADDC **Прибавить литерал к Wn с переносом**

Синтаксис: {метка:} ADDC{.B} #lit10, Wn

Операнды: lit10 ∈ [0 ... 255] для операции с байтом
lit10 ∈ [0 ... 1023] для операции со словом
Wn ∈ [W0 ... W15]

Действие: lit10 + (Wn) + (C) → Wn

Влияет на флаги: DC, N, OV, Z, C

Код:

1011	0000	1Bkk	kkkk	kkkk	dddd

Описание: Складывает 10-битный без знаковый литеральный операнд, содержимое рабочего регистра Wn и бит переноса, и помещает результат обратно в рабочий регистр Wn.

Бит ‘B’ выбирает операцию с байтом или словом (‘0’ для слова, ‘1’ для байта).
Биты ‘k’ определяют литеральный операнд.
Биты ‘d’ выбирают адрес рабочего регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

2: Для операций с байтом, литерал должен быть определён как беззнаковое значение [0:255]. Смотреть часть 4.6 “Использование 10-битных литеральных операндов" для информации на использование 10-битных литеральных операндов в байтном режиме.

3: Флаг Z является “липким” для ADDC, CPB, SUBB и SUBBR. Эти инструкции могут только очищать Z.

Слова: 1

Циклы: 1

Пример 1:

```
ADDC.B      #0xFF, W7   ; Сложить -1 и бит C в W7 (Байтный режим)
```

	Перед инструкцией			После инструкции	
W7	12C0		W7	12BF	
SR	0000	(C=0)	SR	0009	(N,C = 1)

Пример 2:

```
ADDC        #0xFF, W1   ; Сложить 255 и бит C в W1 (Словный режим)
```

	Перед инструкцией			После инструкции
W1	12C0		W1	13C0
SR	0001	(C=1)	SR	0000

ADDC **Прибавить Wb к короткому литералу с переносом**

Синтаксис:	{метка:}	ADDC{.B}	Wb,	#lit5,	Wd [Wd] [Wd++] [Wd--] [++Wd] [--Wd]
Операнды:	Wb ∈ [W0 ... W15] lit5 ∈ [0 ... 31] Wd ∈ [W0 ... W15]				
Действие:	(Wb) + lit5 + (C) → Wd				
Влияет на флаги:	DC, N, OV, Z, C				

Код:	0100	1www	wBqq	qddd	d11k	kkkk

Описание: Складывает содержимое базового регистра Wb, 5-битного без знакового короткого литерального операнда и бита переноса, и размещает результат в регистре назначения Wd. Прямая регистровая адресация должна использоваться для Wb. Прямая или косвенная регистровая адресация может использоваться для Wd.

Биты ‘w’ выбирают адрес базового регистра.
Бит ‘B’ выбирает операцию с байтом или словом (‘0’ для слова, ‘1’ для байта).
Биты ‘q’ выбирают режим адресации места назначения.
Биты ‘d’ выбирают регистр места назначения.
Биты ‘k’ обеспечивают литеральный операнд, пяти-битное целое число.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.
2: Флаг Z является “липким” для ADDC, CPB, SUBB и SUBBR. Эти инструкции могут только очищать Z.

Слова: 1

Циклы: 1

Пример 1:

```
ADDC.B      W0, #0x1F, [W7]          ; Сложить W0, 31 и бит C (Байтный
                                     ; режим)
                                     ; Запомнить результат в [W7]
```

	Перед инструкцией			После инструкции	
W0	CC80		W0	CC80	
W7	12C0		W7	12C0	
Данные 12C0	B000		Данные 12C0	B09F	
SR	0000	(C=0)	SR	00 08	(N = 1)

Пример 2:

```
ADDC        W3, #0x6, [--W4]         ; Сложить W3, 6 бит C (Словный
                                     ; режим)
                                     ; Запомнить результат в [--W4]
```

	Перед инструкцией			После инструкции
W3	6006		W3	6006
W4	1000		W4	0FFE
Данные 0FFE	DDEE		Данные 0FFE	600D
Данные 1000	DDEE		Данные 1000	DDEE
SR	0001	(C=1)	SR	0000

ADDC **Прибавить Wb к Ws с переносом**

Синтаксис:	{метка:}	ADDC{.B}	Wb,	Ws, [Ws], [Ws++], [Ws--], [++Ws], [--Ws],	Wd [Wd] [Wd++] [Wd--] [++Wd] [--Wd]

Операнды: Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

Действие: (Wb) + (Ws) + (C) → Wd

Влияет на флаги: DC, N, OV, Z, C

Код:

0100	1www	wBqq	qddd	dppp	ssss

Описание: Складывает содержимое регистра источника Ws, содержимое базового регистра Wb и бита переноса, и помещает результат в регистр места назначения Wd. Прямая регистровая адресация может быть использована для Wb. Любая, прямая и косвенная регистровая адресация может быть использована для Ws и Wd.
Биты ‘w’ выбирают адрес базового регистра.
Бит ‘B’ выбирает операцию с байтом или словом (‘0’ для слова, ‘1’ для байта).
Биты ‘q’ выбирают режим адреса места назначения.
Биты ‘d’ выбирают регистр места назначения.
Биты ‘p’ выбирают режим адреса источника.
Биты ‘s’ выбирают регистр источник.
Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.
2: Флаг Z является “липким” для ADDC, CPB, SUBB и SUBBR. Эти инструкции могут только очищать Z.

Слова: 1

Циклы: 1

Пример 1:

```
ADDC.B      W0,[W1++],[W2++]        ; Сложить W0, [W1] и бит C (режим
                                    ; байта)
                                    ; Запомнить результат в [W2]
                                    ; Пост-инкремент W1, W2
```

	Перед инструкцией		После инструкции
W0	CC20	W0	CC20
W1	0800	W1	0801
W2	1000	W2	1001
Данные 0800	AB25	Данные 0800	AB25
Данные 1000	FFFF	Данные 1000	FF46
SR	0001 (C=1)	SR	0000

Пример 2:

```
ADDC        W3,[W2++],[W1++]        ; Сложить W3, [W2] и бит C (режим
                                    ; слова)
                                    ; Запомнить результат в [W1]
                                    ; Пост-инкремент W1, W2
```

	Перед инструкцией		После инструкции
W1	1000	W1	1002
W2	2000	W2	2002
W3	0180	W3	0180
Данные 1000	8000	Данные 1000	2681
Данные 2000	2500	Данные 2000	2500
SR	0001 (C=1)	SR	0000

AND **AND f и WREG**

Синтаксис: {метка:} AND{.B} f {,WREG}

Операнды: f ∈ [0 ... 8191]

Действие: (f).AND.(WREG) → место назначение, определённое D

Влияет на флаги: N, Z

Код:

1011	0110	0BDf	ffff	ffff	ffff

Описание: Вычислить логическую операцию AND содержимого рабочего регистра по умолчанию WREG и содержимого файлового регистра, и поместить результат в регистр места назначения. Необязательный операнд WREG определяет регистр места назначения. Если определён WREG, результат запоминается в WREG. Если WREG не определён, результат запоминается в файловом регистре.

Бит ‘B’ выбирает операцию с байтом или словом (‘0’ для слова, ‘1’ для байта).
Бит ‘D’ выбирает место назначения (‘0’ для WREG, ‘1’ для файлового регистра).
Биты ‘f’ выбирают адрес файлового регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.
2: WREG установлен на рабочий регистр W0.

Слова: 1

Циклы: 1

Пример 1:

```
AND.B       RAM100      ; AND WREG в RAM100 (Режим байта)
```

	Перед инструкцией		После инструкции	
WREG	CC80	WREG	CC80	
RAM100	FFC0	RAM100	FF80	
SR	0000	SR	0008	(N = 1)

Пример 2:

```
AND         RAM200, WREG        ; AND RAM200 в WREG (Режим слова)
```

	Перед инструкцией		После инструкции
WREG	CC80	WREG	0080
RAM200	12C0	RAM200	12C0
SR	0000	SR	0000

AND **AND Литерал и Wd**

Синтаксис: {метка:} AND{.B} #lit10, Wn

Операнды: lit10 ∈ [0 ... 255] для операций с байтом
lit10 ∈ [0 ... 1023] для операций со словом
Wn ∈ [W0 ... W15]

Действие: lit10.AND.(Wn) → Wn

Влияет на флаги: N, Z

Код:

1011	0010	0Bkk	kkkk	kkkk	dddd

Описание: Вычислить логическую операцию AND 10-битного литерального операнда и содержимого рабочего регистра Wn, и разместить результат обратно в рабочий регистр Wn. Прямая регистровая адресация должна быть использована для Wn.

Бит ‘B’ выбирает операцию с байтом или словом (‘0’ для слова, ‘1’ для байта).
Биты ‘k’ определяют литеральный операнд.
Биты ‘d’ выбирают адрес рабочего регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

2: Для байтных операций, литерал может быть определён как беззнаковое значение [0:255]. Смотреть часть 4.6 “Использование 10-битных литеральных операндов” для информации на использование 10-битных литеральных операндов в байтном режиме.

Слова: 1

Циклы: 1

Пример 1:
```
AND.B       #0x83, W7    ; AND 0x83 в W7 (Режим байта)
```

	Перед инструкцией		После инструкции	
W7	12C0	W7	1280	
SR	0000	SR	0008	(N = 1)

Пример 2:
```
AND         #0x333, W1          ; AND 0x333 в W1 (Режим слова)
```

	Перед инструкцией		После инструкции	
W1	12D0	W1	0210	
SR	0000	SR	0000	

AND **AND Wb и короткий литерал**

Синтаксис:	{метка:}	AND{.B}	Wb,	#lit5,	Wd [Wd] [Wd++] [Wd--] [++Wd] [--Wd]
Операнды:	Wb ∈ [W0 ... W15] lit5 ∈ [0 ... 31] Wd ∈ [W0 ... W15]				
Действие:	(Wb).AND.lit5 → Wd				
Влияет на флаги:	N, Z				

Код:	0110	0www	wBqq	qddd	d11k	kkkk

Описание: Вычислить логическую операцию AND содержимого базового регистра Wb и 5-битного литерала, и поместить результат в регистр места назначения Wd. Прямая регистровая адресация может быть использована для Wb. Прямая или косвенная регистровая адресация может быть использована для Wd.

Биты ‘w’ выбирают адрес базового регистра.
Бит ‘B’ выбирает операцию с байтом или словом (‘0’ для слова, ‘1’ для байта).
Биты ‘q’ выбирают режим адреса места назначения.
Биты ‘d’ выбирают регистр места назначения.
Биты ‘k’ обеспечивают литеральный операнд, пяти-битное целое число.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

Слова: 1

Циклы: 1

Пример 1:

```
AND.B       W0,#0x3,[W1++]          ; AND W0 и 0x3 (Режим байта)
                                    ; Запомнить в [W1]
                                    ; Пост-инкремент W1
```

	Перед инструкцией		После инструкции
W0	23A5	W0	23A5
W1	2211	W1	2212
Данные 2210	9999	Данные 2210	0199
SR	0000	SR	0000

Пример 2:

```
AND         W0,#0x1F,W1 ; AND W0 и 0x1F (Режим слова)
                        ; Запомнить в W1
```

	Перед инструкцией		После инструкции
W0	6723	W0	6723
W1	7878	W1	0003
SR	0000	SR	0000

AND **And Wb и Ws**

Синтаксис:	{метка:}	AND{.B}	Wb,	Ws, [Ws], [Ws++], [Ws--], [++Ws], [--Ws],	Wd [Wd] [Wd++] [Wd--] [++Wd] [--Wd]

Операнды: Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

Действие: (Wb).AND.(Ws) → Wd

Влияет на флаги: N, Z

Код:

0110	0www	wBqq	qddd	dppp	ssss

Описание: Вычислить логическую операцию AND содержимого регистра источника Ws и содержимого базового регистра Wb, и поместить результат в регистр места назначения Wd. Прямая регистровая адресация может быть использована для Wb. Любая, прямая или косвенная регистровая адресация может быть использована для Ws и Wd.

Биты ‘w’ выбирают адрес базового регистра.
Бит ‘B’ выбирает операцию с байтом или словом (‘0’ для слова, ‘1’ для байта).
Биты ‘q’ выбирают режим адреса места назначения.
Биты ‘d’ выбирают регистр места назначения.
Биты ‘p’ выбирают режим адреса источника.
Биты ‘s’ выбирают регистр источник.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

Слова: 1

Циклы: 1

Пример 1:

```
AND.B       W0, W1 [W2++]           ; AND W0 и W1, и
                                    ; запомнить в [W2] (Режим байта)
                                    ; Пост-инкремент W2
```

	Перед инструкцией		После инструкции
W0	AA55	W0	AA55
W1	2211	W1	2211
W2	1001	W2	1002
Данные 1000	FFFF	Данные 1000	11FF
SR	0000	SR	0000

Пример 2:

```
AND         W0, [W1++], W2          ; AND W0 и [W1], и
                                    ; запомнить в W2 (Режим слова)
                                    ; Пост инкремент W1
```

	Перед инструкцией		После инструкции
W0	AA55	W0	AA55
W1	1000	W1	1002
W2	55AA	W2	2214
Данные 1000	2634	Данные 1000	2634
SR	0000	SR	0000

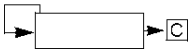
ASR **Арифметический сдвиг вправо f**

Синтаксис: {метка:} ASR{.B} f {,WREG}

Операнды: f ∈ [0 ... 8191]

Действие: Для операции с байтом:

(f<7>) → Dest<7>
(f<7>) → Dest<6>
(f<6:1>) → Dest<5:0>
(f<0>) → C
Для операции со словом:
(f<15>) → Dest<15>
(f<15>) → Dest<14>
(f<14:1>) → Dest<13:0>
(f<0>) → C



Влияет на флаги: N, Z, C

Код:

1101	0101	1BDf	ffff	ffff	ffff

Описание: Сдвинуть содержимого файлового регистра на один бит вправо и поместить результат в регистр места назначения. Наименьший значемый бит файлового регистра сдвигается в разряд C регистра STATUS. После выполнения сдвига, получаем расширенный знаком результат. Опционально операнд WREG определяется регистром места назначения. Если указан WREG, то результат запоминается в WREG. Если WREG не указан, результат запоминается в файловом регистре.

Бит ‘B’ выбирает операцию с байтом или словом (‘0’ для слова, ‘1’ для байта).
Бит ‘D’ выбирает место назначения (‘0’ для WREG, ‘1’ для файлового регистра).
Биты ‘f’ выбирают адрес файлового регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.
2: В качестве WREG используется рабочий регистр W0.

Слова: 1

Циклы: 1

Пример 1:

```
ASR.B       RAM400, WREG          ; ASR RAM400 и запоминаются в WREG
                                  ; (Режим байта)
```

	Перед инструкцией		После инструкции	
WREG	0600	WREG	0611	
RAM400	0823	RAM400	0823	
SR	0000	SR	0001	(C=1)

Пример 2:

```
ASR         RAM200                ; ASR RAM200 (Режим слова)
```

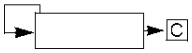
	Перед инструкцией		После инструкции	
RAM200	8009	RAM200	C004	
SR	0000	SR	0009	(N, C = 1)

ASR **Арифметический сдвиг вправо Ws**

Синтаксис: {метка:} ASR{.B} Ws, Wd
[Ws], [Wd]
[Ws++], [Wd++]
[Ws--], [Wd--]
[++Ws], [++Wd]
[--Ws], [--Wd]

Операнды: Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

Действие: Для операции с байтом:
(Ws<7>) → Wd<7>
(Ws<7>) → Wd<6>
(Ws<6:1>) → Wd<5:0>
(Ws<0>) → C
Для операции со словом:
(Ws<15>) → Wd<15>
(Ws<15>) → Wd<14>
(Ws<14:1>) → Wd<13:0>
(Ws<0>) → C



Влияет на флаги: N, Z, C

Код:

1101	0001	1Bqq	qddd	dppp	ssss

Описание: Сдвинуть содержимого регистра источника Ws на один бит вправо и поместить результат в регистр места назначения Wd. Наименьший значимый бит регистра Ws сдвигается в разряд C регистра STATUS. После выполнения сдвига, получаем расширенный знаком результат. Прямая или косвенная регистровая адресация может быть использована для Ws и Wd.
Бит ‘B’ выбирает операцию с байтом или словом (‘0’ для слова, ‘1’ для байта).
Биты ‘q’ выбирают режим адреса места назначения.
Биты ‘d’ выбирают регистр места назначения.
Биты ‘p’ выбирают режим адреса источника.
Биты ‘s’ выбирают регистр источник.
Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

Слова: 1

Циклы: 1

Пример 1:

```
ASR.B       [W0++], [W1++]          ; ASR [W0] и запомнить в [W1]
                                    ; (Режим байта)
                                    ; Пост-инкремент W0 и W1
```

	Перед инструкцией		После инструкции
W0	0600	W0	0601
W1	0801	W1	0802
Данные 600	2366	Данные 600	2366
Данные 800	FFC0	Данные 800	33C0
SR	0000	SR	0000

Пример 2:

```
ASR         W12, W13    ; ASR W12 и запомнить в W13 (Режим слова)
```

	Перед инструкцией		После инструкции	
W12	AB01	W12	AB01	
W13	0322	W13	D580	
SR	0000	SR	0009	(N, C = 1)

ASR **Арифметический сдвиг вправо к короткий литерал**

Синтаксис: {метка:} ASR Wb, #lit4, Wnd

Операнды: Wb ∈ [W0 ... W15]
lit4 ∈ [0...15]
Wnd ∈ [W0 ... W15]

Действие: lit4<3:0> → Shift_Val
Wb<15> → Wnd<15:15-Shift_Val + 1>
Wb<15:Shift_Val> → Wnd<15-Shift_Val:0>

Влияет на флаги: N, Z

Код:

1101	1110	1www	wddd	d100	kkkk

Описание: Арифметически сдвинуть вправо содержимое регистра источника Wb на количество позиций равных 4-битному без знаковому литералу, и запомнить результат в регистре места назначения Wnd. После того как сдвиг будет выполнен, результат будет расширен знаком. Для регистров Wb и Wnd можно использовать прямую адресацию.

Бит ‘B’ выбирает операцию с байтом или словом (‘0’ для слова, ‘1’ для байта).
Биты ‘w’ выбирают адрес базового регистра.
Биты ‘d’ выбирают регистр места назначения.
Биты ‘k’ обеспечивают литеральный операнд.

Примечание: Эта инструкция работает только в режиме слова.

Слова: 1

Циклы: 1

Пример 1:

```
ASR     W0, #0x4, W1        ; ASR W0 на 4 и запомнить в W1
```

	Перед инструкцией	После инструкции
W0	060F	060F
W1	1234	0060
SR	0000	0000

Пример 2:

```
ASR     W0, #0x6, W1        ; ASR W0 на 6 и запомнить в W1
```

	Перед инструкцией	После инструкции	
W0	80FF	80FF	
W1	0060	FE03	
SR	0000	0008	(N = 1)

Пример 3:

```
ASR     W0, #0xF, W1        ; ASR W0 на 15 и запомнить в W1
```

	Перед инструкцией	После инструкции	
W0	70FF	70FF	
W1	CC26	0000	
SR	0000	0002	(Z = 1)

ASR **Арифметический сдвиг вправо к Wns**

Синтаксис: {метка:} ASR Wb, Wns, Wnd

Операнды: Wb ∈ [W0 ... W15]
Wns ∈ [W0 ...W15]
Wnd ∈ [W0 ... W15]

Действие: Wns<3:0> → Shift_Val
Wb<15> → Wnd<15:15-Shift_Val + 1>
Wb<15:Shift_Val> → Wnd<15-Shift_Val:0>

Влияет на флаги: N, Z

Код:

1101	1110	1www	wddd	d000	ssss

Описание: Арифметически сдвинуть вправо содержимое регистра источника Wb на количество позиций равных 4 наименьшим значащим битам регистра Wns (вплоть до 15 позиций) и запомнить результат в регистре места назначения Wnd. После того как сдвиг будет выполнен, результат будет расширен знаком. Для регистров Wb, Wns и Wnd можно использовать прямую адресацию.

Биты ‘w’ выбирают адрес базового регистра.
Биты ‘d’ выбирают регистр места назначения.
Биты ‘s’ выбирают регистр источник.

Примечание 1: Эта инструкция работает только в режиме слова.
2: Если Wns больше чем 15, Wnd = 0x0 если Wb положителен, и Wnd = 0xFFFF если Wb отрицателен.

Слова: 1

Циклы: 1

Пример 1:
```
ASR     W0, W5, W6      ; ASR W0 на W5 и запомнить в W6
```

	Перед инструкцией		После инструкции
W0	80FF	W0	80FF
W5	0004	W5	0004
W6	2633	W6	F80F
SR	0000	SR	0000

Пример 2:
```
ASR     W0, W5, W6      ; ASR W0 на W5 и запомнить в W6
```

	Перед инструкцией		После инструкции
W0	6688	W0	6688
W5	000A	W5	000A
W6	FF00	W6	0019
SR	0000	SR	0000

Пример 3:
```
ASR     W11, W12, W13   ; ASR W11 на W12 и запомнить в W13
```

	Перед инструкцией		После инструкции	
W11	8765	W11	8765	
W12	88E4	W12	88E4	
W13	A5A5	W13	F876	
SR	0000	SR	0008	(N = 1)

BCLR **Очистить бит f**

Синтаксис: {метка:} BCLR{.B} f, #bit4

Операнды: f ∈ [0 ... 8191] для операции с байтом
f ∈ [0 ... 8190] (только для чётного) для операции со словом
bit4 ∈ [0 ... 7] для операции с байтом
bit4 ∈ [0 ... 15] для операции с байтом

Действие: 0 → f<bit4>

Влияет на флаги: Не влияет

Код:

1010	1001	bbbf	ffff	ffff	fffb

Описание: Очистить бит в файловом регистре f определённый через ‘bit4’. Номерация битов начинается с наименьшего значащего бита (bit 0) и продвигается к наиболее значащему биту (бит 7 для операций сбайтом, бит 15 для операций со словом).

Биты ‘b’ выбирают значение bit4 битовой позиции, которая должна быть очищена.
Биты ‘f’ выбирают адрес файлового регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.
2: Когда эта инструкция используется в режиме слова, файловый регистр адресуется с выравниванием по слову.
3: Когда эта инструкция используется в байтном режиме, ‘bit4’ должен быть между 0 и 7.

Слова: 1

Циклы: 1

Пример 1:

```
BCLR.B      0x800, #0x7          ; Очистить бит 7 в 0x800
```

	Перед инструкцией		После инструкции
Данные 0800	66EF	Данные 0800	666F
SR	0000	SR	0000

Пример 2:

```
BCLR        0x400, #0x9          ; Очистить бит 9 в 0x400
```

	Перед инструкцией		После инструкции
Данные 0400	AA55	Данные 0400	A855
SR	0000	SR	0000

BCLR **Очистить бит в Ws**

Синтаксис: {метка:} BCLR{.B} Ws, [Ws], [Ws++], [Ws--], [++Ws], [--Ws], #bit4

Операнды: Ws ∈ [W0 ... W15]
bit4 ∈ [0 ... 7] для операции с байтом
bit4 ∈ [0 ... 15] для операции со словом

Действие: 0 → Ws<bit4>

Влияет на флаги: Не влияет

Код:

1010	0001	bbbb	0B00	0ppp	ssss

Описание: Очистить бит в регистре Ws определённый через ‘bit4’. Нумерация битов начинается с наименьшего значащего бита (bit 0) и продвигается к наиболее значащему биту (бит 7 для операций с байтом, бит 15 для операций со словом). Прямая или косвенная регистровая адресация может быть использована для Ws.

Биты ‘b’ выбирают значение bit4 позиции бита, который должен быть очищен.
Бит ‘B’ выбирает операцию с байтом или словом (‘0’ для слова, ‘1’ для байта).
Биты ‘s’ выбирают регистр источник/назначение.
Биты ‘p’ выбирают режим адреса источника.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.
2: Когда эта инструкция используется в режиме слова, файловый регистр адресуется с выравниванием по слову.
3: Когда эта инструкция используется в байтном режиме, ‘bit4’ должен быть между 0 и 7.

Слова: 1

Циклы: 1

Пример 1:

```
BCLR.B      W2, #0x2            ; Очистить бит 3 в W2
```

	Перед инструкцией		После инструкции
W2	F234	W2	F230
SR	0000	SR	0000

Пример 2:

```
BCLR        [W0++], #0x0        ; Очистить бит 0 в [W0]
                                ; Пост-инкремент W0
```

	Перед инструкцией		После инструкции
W0	2300	W0	2302
Данные 2300	5607	Данные 2300	5606
SR	0000	SR	0000

BRA **Безусловное ветвление**

Синтаксис: {метка:} BRA Expr

Операнды: Expr может быть меткой, абсолютным адресом или выражением.
Expr решается компоновщиком в Slit16, где Slit16 ∈ [-32768 ... +32767].

Действие: (PC + 2) + 2 * Slit16 → PC
NOP → Регистр инструкции

Влияет на флаги: Не влияет

Код:

0011	0111	nnnn	nnnn	nnnn	nnnn

Описание: Вызывает безусловное ветвление программы, относительно следующего PC. Смещение ветвления есть двоичное число дополнения ‘2 * Slit16’, которое поддерживает ветвления до 32K инструкций вперёд или назад. Значение Slit16 решено компоновщиком, чтобы обеспечить метку, абсолютный адрес или выражение. После того, как ветвление принято, новый адрес будет (PC + 2) + 2 * Slit16, поскольку PC увеличился при выборе следующей инструкции.

Биты ‘n’ являются знаковым литералом, который определяет число программных слов смещения от (PC + 2).

Слова: 1

Циклы: 2

Пример 1:

```
002000      HERE:       BRA THERE               ; Перейти на THERE
002002                  . . .
002004                  . . .
002006                  . . .
002008                  . . .
00200A      THERE:      . . .
00200C                  . . .
```

	Перед инструкцией		После инструкции
PC	00 2000	PC	00 200A
SR	0000	SR	0000

Пример 2:

```
002000      HERE:       BRA THERE+0x2           ; Перейти на THERE+0x2
002002                  . . .
002004                  . . .
002006                  . . .
002008                  . . .
00200A      THERE:      . . .
00200C                  . . .
```

	Перед инструкцией		После инструкции
PC	00 2000	PC	00 200C
SR	0000	SR	0000

Пример 3:

```
002000      HERE:       BRA 0x1366              ; Перейти на  0x1366
002002                  . . .
002004                  . . .
```

	Перед инструкцией		После инструкции	
PC	00 2000	PC	00 1366	
SR	0000	SR	0008	(N = 1)

BRA **Вычисляемое ветвление**

Синтаксис: {метка:} BRA Wn

Операнды: Wn ∈ [W0 ... W15]

Действие: (PC + 2) + (2 * Wn) → PC
NOP → Регистр инструкции

Влияет на флаги: Не влияет

Код:

0000	0001	0110	0000	0000	ssss

Описание: Безусловные ветвления программы, относительно следующего PC. Смещение ветвления является расширенное знаком 17-битное значение (2 * Wn), которое поддерживает ветвления до 32K инструкций вперёд или назад. После выполнения этой инструкции, новое значение PC будет (PC + 2) + 2 * Wn, поскольку PC инкрементируется при выборке следующей инструкции.

Биты ‘s’ выбирают регистр источник.

Слова: 1

Циклы: 2

Пример 1:

```
002000      HERE:       BRA W7      ; Ветвление вперёд (2+2*W7)
002002                  . . .
...                     . . .
...                     . . .
002108                  . . .
00210A      TABLE7:     . . .
00210C                  . . .
```

	Перед инструкцией
PC	00 2000
W7	0084
SR	0000

	После инструкции
PC	00 210A
	0084
SR	0000

BRA C **Ветвление, если перенос**

Синтаксис: {метка:} BRA C, Expr

Операнды: Expr может быть меткой, абсолютным адресом или выражением.
Expr решается компоновщиком в Slit16, где Slit16 ∈ [-32768 ... +32767].

Действие: Условие = C
If (Условие)
(PC + 2) + 2 * Slit16 → PC
NOP → Регистр Инструкции

Влияет на флаги: Не влияет

Код:

0011	0001	nnnn	nnnn	nnnn	nnnn

Описание: Если бит флага переноса установлен в ‘1’, то программа ветвится относительного следующего значения PC. Смещение ветвления задаётся двоичным дополненным числом ‘2 * Slit16’, которое поддерживает ветвление до 32K инструкций вперёд и назад. Значение Slit16 решено компоновщиком, чтобы обеспечить метку, абсолютный адрес или выражение. Если ветвление принято, новый адрес будет (PC + 2) + 2 * Slit16, поскольку PC увеличился при выборе следующей инструкции. Тогда инструкция становится двухцикловой, с выполнением NOP во втором цикле.

Биты ‘n’ являются 16-битным знаковым литералом, который определяет смещение от (PC + 2) в словах инструкций.

Слова: 1

Циклы: 1 (2 если ветвление принято)

Пример 1:

```
002000      HERE:       BRA C, CARRY            ; Если C установлен,
002002      NO_C:       . . .                   ; перейти на CARRY
002004                  . . .                   ; Иначе ... продолжить
002006                  GOTO THERE
002008      CARRY:      . . .
00200A                  . . .
00200C      THERE:      . . .
00200E                  . . .
```

	Перед инструкцией		После инструкции	
PC	00 2000		PC 00 2008	
SR	0001	(C = 1)	SR 0001	(C = 1)

Пример 2:

```
002000      HERE:       BRA C, CARRY            ; Если C установлен,
002002      NO_C:       . . .                   ; перейти на CARRY
002004                  . . .                   ; Иначе ... продолжить
002006                  GOTO THERE
002008      CARRY:      . . .
00200A                  . . .
00200C      THERE:      . . .
00200E                  . . .
```

	Перед инструкцией		После инструкции	
PC	00 2000		PC 00 2002	
SR	0000	(C = 0)	SR 0000	(C = 0)

Пример 3:

```
006230      HERE:       BRA C, CARRY            ; Если C установлен,
006232      NO_C:       . . .                   ; перейти на CARRY
006234                  . . .                   ; Иначе ... продолжить
006236                  GOTO THERE
006238      CARRY:      . . .
00623A                  . . .
00623C      THERE:      . . .
00623E                  . . .
```

	Перед инструкцией		После инструкции	
PC	00 6230		PC 00 6238	
SR	0001	(C = 1)	SR 0001	(C = 1)

Пример 4:

```
006230      START:      . . .
006232                  . . .
006234      CARRY:      . . .
006236                  . . .
006238                  . . .
00623A                  . . .
00623C      HERE:       BRA C, CARRY            ; Если С установлен,
00623E                  . . .                   ; перейти на CARRY
                                                ; Иначе ... продолжить
```

	Перед инструкцией		После инструкции	
PC	00 623C		PC 00 6234	
SR	0001	(C = 1)	SR 0001	(C = 1)

BRA GE **Ветвление, если знаковое больше или равно**

Синтаксис: {метка:} BRA GE, Expr

Операнды: Expr может быть меткой, абсолютным адресом или выражением.
Expr решается компоновщиком в Slit16, где Slit16 ∈ [-32768 ... +32767].

Действие: Условие = (N&&OV)||(!N&&!OV)
If (Условие)
(PC + 2) + 2 * Slit16 → PC
NOP → Регистр Инструкции

Влияет на флаги: Не влияет

Код:

0011	1101	nnnn	nnnn	nnnn	nnnn

Описание: Если логическое выражение (N&&OV)||(!N&&!OV) справедливо, то программа ветвится относительного следующего значения PC. Смещение ветвления задаётся двоичным дополненным числом ‘2 * Slit16’, которое поддерживает ветвление до 32K инструкций вперёд и назад. Значение Slit16 решено компоновщиком, чтобы обеспечить метку, абсолютный адрес или выражение.

Если ветвление принято, новый адрес будет (PC + 2) + 2 * Slit16, поскольку PC увеличился при выборе следующей инструкции. Тогда инструкция становится двухцикловой, с выполнением NOP во втором цикле.

Биты ‘n’ являются 16-битным знаковым литералом, который определяет смещение от (PC + 2) в словах инструкций.

Примечание: Ассемблер преобразует указанную метку в смещение, которое нужно использовать.

Слова: 1

Циклы: 1 (2 если ветвление принято)

Пример 1:

```
007600      LOOP:       . . .
007602                  . . .
007604                  . . .
007606                  . . .
007608      HERE:       BRA GE, LOOP            ; Если GE, перейти на
00760A      NO_GE:      . . .                   ; LOOP
                                                ; Иначе ... продолжить
```

	Перед инструкцией		После инструкции
PC	00 7608	PC	00 7600
SR	0000	SR	0000

Пример 2:

```
007600      LOOP:       . . .
007602                  . . .
007604                  . . .
007606                  . . .
007608      HERE:       BRA GE, LOOP            ; Если GE, перейти на
00760A      NO_GE:      . . .                   ; LOOP
                                                ; Иначе ... продолжить
```

	Перед инструкцией		После инструкции	
PC	00 7608		PC	00 760A	
SR	0008	(N = 1)	SR	0008	(N = 1)

BRA GEU **Ветвление, если беззнаковое больше или равно**

Синтаксис: {метка:} BRA GEU, Expr

Операнды: Expr может быть меткой, абсолютным адресом или выражением.
Expr решается компоновщиком в Slit16, где Slit16 ∈ [-32768 ... +32767].

Действие: Условие = C
If (Условие)
(PC + 2) + 2 * Slit16 → PC
NOP → Регистр Инструкции

Влияет на флаги: Не влияет

Код:

0011	0001	nnnn	nnnn	nnnn	nnnn

Описание: Если бит флага переноса установлен в ‘1’, то программа ветвится относительного следующего значения PC. Смещение ветвления задаётся двоичным дополненным числом ‘2 * Slit16’, которое поддерживает ветвление до 32K инструкций вперёд и назад. Значение Slit16 решено компоновщиком, чтобы обеспечить метку, абсолютный адрес или выражение.

Если ветвление принято, новый адрес будет (PC + 2) + 2 * Slit16, поскольку PC увеличился при выборе следующей инструкции. Тогда инструкция становится двухцикловой, с выполнением NOP во втором цикле.

Биты ‘n’ являются 16-битным знаковым литералом, который определяет смещение от (PC + 2) в словах инструкций.

Примечание: Эта инструкция идентична инструкции BRA C, Expr (Ветвление, если перенос) и имеет аналогичный код. Поэтому при деассемблировании может распознаваться, как BRA C, Slit16.

Слова: 1

Циклы: 1 (2 если ветвление принято)

Пример 1:

```
002000      HERE:       BRA GEU, BYPASS         ; Если C установлен,
002002      NO_GEU:     . . .                   ; перейти на BYPASS
002004                  . . .                   ; Иначе ... продолжить
002006                  . . .
002008                  . . .
00200A                  GOTO THERE
00200C      BYPASS:     . . .
00200E
```

	Перед инструкцией		После инструкции	
PC	00 2000		PC 00 200C	
SR	0001	(C = 1)	SR 0001	(C = 1)

BRA GT **Ветвление, если знаковое больше**

Синтаксис: {метка:} BRA GT, Expr

Операнды: Expr может быть меткой, абсолютным адресом или выражением.
Expr решается компоновщиком в Slit16, где Slit16 ∈ [-32768 ... +32767].

Действие: Условие = (!Z&&N&&OV)||(!Z&&!N&&!OV)
If (Условие)
(PC + 2) + 2 * Slit16 → PC
NOP → Регистр Инструкции

Влияет на флаги: Не влияет

Код:

0011	1100	nnnn	nnnn	nnnn	nnnn

Описание: Если логическое выражение (!Z&&N&&OV)||(!Z&&!N&&!OV) справедливо, то программа ветвится относительного следующего значения PC. Смещение ветвления задаётся двоичным дополненным числом ‘2 * Slit16’, которое поддерживает ветвление до 32K инструкций вперёд и назад. Значение Slit16 решено компоновщиком, чтобы обеспечить метку, абсолютный адрес или выражение.

Если ветвление принято, новый адрес будет (PC + 2) + 2 * Slit16, поскольку PC увеличился при выборе следующей инструкции. Тогда инструкция становится двухцикловой, с выполнением NOP во втором цикле.

Биты ‘n’ являются 16-битным знаковым литералом, который определяет смещение от (PC + 2) в словах инструкций.

Слова: 1

Циклы: 1 (2 если ветвление принято)

Пример 1:

```
002000      HERE:       BRA GT, BYPASS          ; Если GT,
002002      NO_GT:      . . .                   ; перейти на BYPASS
002004                  . . .                   ; Иначе ... продолжить
002006                  . . .
002008                  . . .
00200A                  GOTO THERE
00200C      BYPASS:     . . .
00200E
```

	Перед инструкцией		После инструкции	
PC	00 2000		PC	00 200C
SR	0001	(C = 1)	SR	0001 (C = 1)

BRA GTU **Ветвление, если беззнаковое больше**

Синтаксис: {метка:} BRA GTU, Expr

Операнды: Expr может быть меткой, абсолютным адресом или выражением.
Expr решается компоновщиком в Slit16, где Slit16 ∈ [-32768 ... +32767].

Действие: Условие = (C&&!Z)
If (Условие)
(PC + 2) + 2 * Slit16 → PC
NOP → Регистр Инструкции

Влияет на флаги: Не влияет

Код:

0011	1110	nnnn	nnnn	nnnn	nnnn

Описание: Если логическое выражение (C&&!Z) справедливо, то программа ветвится относительного следующего значения PC. Смещение ветвления задаётся двоичным дополненным числом ‘2 * Slit16’, которое поддерживает ветвление до 32K инструкций вперёд и назад. Значение Slit16 решено компоновщиком, чтобы обеспечить метку, абсолютный адрес или выражение.

Если ветвление принято, новый адрес будет (PC + 2) + 2 * Slit16, поскольку PC увеличился при выборе следующей инструкции. Тогда инструкция становится двухцикловой, с выполнением NOP во втором цикле.

Биты ‘n’ являются 16-битным знаковым литералом, который определяет смещение от (PC + 2) в словах инструкций.

Слова: 1

Циклы: 1 (2 если ветвление принято)

Пример 1:

```
002000      HERE:       BRA GTU, BYPASS         ; Если GTU,
002002      NO_GTU:     . . .                   ; перейти на BYPASS
002004                  . . .                   ; Иначе ... продолжить
002006                  . . .
002008                  . . .
00200A                  GOTO THERE
00200C      BYPASS:     . . .
00200E
```

	Перед инструкцией		После инструкции	
PC	00 2000		PC	00 200C
SR	0001	(C = 1)	SR	0001 (C = 1)

BRA LE **Ветвление, если знаковое меньше или равно**

Синтаксис: {метка:} BRA LE, Expr

Операнды: Expr может быть меткой, абсолютным адресом или выражением.
Expr решается компоновщиком в Slit16, где Slit16 ∈ [-32768 ... +32767].

Действие: Условие = Z||(N&&!OV)||(!N&&OV)
If (Условие)
(PC + 2) + 2 * Slit16 → PC
NOP → Регистр Инструкции

Влияет на флаги: Не влияет

Код:

0011	0100	nnnn	nnnn	nnnn	nnnn

Описание: Если логическое выражение (Z||(N&&!OV)||(!N&&OV)) справедливо, то программа ветвится относительного следующего значения PC. Смещение ветвления задаётся двоичным дополненным числом ‘2 * Slit16’, которое поддерживает ветвление до 32K инструкций вперёд и назад. Значение Slit16 решено компоновщиком, чтобы обеспечить метку, абсолютный адрес или выражение.

Если ветвление принято, новый адрес будет (PC + 2) + 2 * Slit16, поскольку PC увеличился при выборе следующей инструкции. Тогда инструкция становится двухцикловой, с выполнением NOP во втором цикле.

Биты ‘n’ являются 16-битным знаковым литералом, который определяет смещение от (PC + 2) в словах инструкций.

Слова: 1

Циклы: 1 (2 если ветвление принято)

Пример 1:

```
002000      HERE:       BRA LE, BYPASS          ; Если LE,
002002      NO_LE:      . . .                   ; перейти на BYPASS
002004                  . . .                   ; Иначе ... продолжить
002006                  . . .
002008                  . . .
00200A                  GOTO THERE
00200C      BYPASS:     . . .
00200E
```

	Перед инструкцией		После инструкции	
PC	00 2000		PC	00 2002
SR	0001	(C = 1)	SR	0001	(C = 1)

BRA LEU **Ветвление, если беззнаковое меньше или равно**

Синтаксис: {метка:} BRA LEU, Expr

Операнды: Expr может быть меткой, абсолютным адресом или выражением.
Expr решается компоновщиком в Slit16, где Slit16 ∈ [-32768 ... +32767].

Действие: Условие = !C||Z
If (Условие)
(PC + 2) + 2 * Slit16 → PC
NOP → Регистр Инструкции

Влияет на флаги: Не влияет

Код:

0011	0110	nnnn	nnnn	nnnn	nnnn

Описание: Если логическое выражение (!C||Z) справедливо, то программа ветвится относительного следующего значения PC. Смещение ветвления задаётся двоичным дополненным числом ‘2 * Slit16’, которое поддерживает ветвление до 32K инструкций вперёд и назад. Значение Slit16 решено компоновщиком, чтобы обеспечить метку, абсолютный адрес или выражение.

Если ветвление принято, новый адрес будет (PC + 2) + 2 * Slit16, поскольку PC увеличился при выборе следующей инструкции. Тогда инструкция становится двухцикловой, с выполнением NOP во втором цикле.

Биты ‘n’ являются 16-битным знаковым литералом, который определяет смещение от (PC + 2) в словах инструкций.

Слова: 1

Циклы: 1 (2 если ветвление принято)

Пример 1:

```
002000      HERE:       BRA LEU, BYPASS         ; Если LEU,
002002      NO_LEU:     . . .                   ; перейти на BYPASS
002004                  . . .                   ; Иначе ... продолжить
002006                  . . .
002008                  . . .
00200A                  GOTO THERE
00200C      BYPASS:     . . .
00200E
```

	Перед инструкцией		После инструкции	
PC	00 2000		PC	00 200C
SR	0001	(C = 1)	SR	0001 (C = 1)

BRA LT **Ветвление, если знаковое меньше**

Синтаксис: {метка:} BRA LT, Expr

Операнды: Expr может быть меткой, абсолютным адресом или выражением.
Expr решается компоновщиком в Slit16, где Slit16 ∈ [-32768 ... +32767].

Действие: Условие = (N&&!OV)||(!N&&OV)
If (Условие)
(PC + 2) + 2 * Slit16 → PC
NOP → Регистр Инструкции

Влияет на флаги: Не влияет

Код:

0011	0101	nnnn	nnnn	nnnn	nnnn

Описание: Если логическое выражение (N&&!OV)||(!N&&OV) справедливо, то программа ветвится относительного следующего значения PC. Смещение ветвления задаётся двоичным дополненным числом ‘2 * Slit16’, которое поддерживает ветвление до 32K инструкций вперёд и назад. Значение Slit16 решено компоновщиком, чтобы обеспечить метку, абсолютный адрес или выражение.

Если ветвление принято, новый адрес будет (PC + 2) + 2 * Slit16, поскольку PC увеличился при выборе следующей инструкции. Тогда инструкция становится двухцикловой, с выполнением NOP во втором цикле.

Биты ‘n’ являются 16-битным знаковым литералом, который определяет смещение от (PC + 2) в словах инструкций.

Слова: 1

Циклы: 1 (2 если ветвление принято)

Пример 1:

```
002000      HERE:       BRA LT, BYPASS          ; Если LT,
002002      NO_LT:      . . .                   ; перейти на BYPASS
002004                  . . .                   ; Иначе ... продолжить
002006                  . . .
002008                  . . .
00200A                  GOTO THERE
00200C      BYPASS:     . . .
00200E
```

	Перед инструкцией		После инструкции	
PC	00 2000		PC 00 2002	
SR	0001	(C = 1)	SR 0001	(C = 1)

BRA LTU **Ветвление, если беззнаковое меньше**

Синтаксис: {метка:} BRA LTU, Expr

Операнды: Expr может быть меткой, абсолютным адресом или выражением.
Expr решается компоновщиком в Slit16, где Slit16 ∈ [-32768 ... +32767].

Действие: Условие = !C
If (Условие)
(PC + 2) + 2 * Slit16 → PC
NOP → Регистр Инструкции

Влияет на флаги: Не влияет

Код:

0011	1001	nnnn	nnnn	nnnn	nnnn

Описание: Если флаг переноса равен 0, то программа ветвится относительного следующего значения PC. Смещение ветвления задаётся двоичным дополненным числом ‘2 * Slit16’, которое поддерживает ветвление до 32K инструкций вперёд и назад. Значение Slit16 решено компоновщиком, чтобы обеспечить метку, абсолютный адрес или выражение.

Если ветвление принято, новый адрес будет (PC + 2) + 2 * Slit16, поскольку PC увеличился при выборе следующей инструкции. Тогда инструкция становится двухцикловой, с выполнением NOP во втором цикле.

Биты ‘n’ являются 16-битным знаковым литералом, который определяет смещение от (PC + 2) в словах инструкций.

Примечание: Эта инструкция идентична инструкции BRA NC, Expr (Ветвление, если не перенос) и имеет аналогичный код. Поэтому при деассемблировании может распознаваться, как BRA NC, Slit16.

Слова: 1

Циклы: 1 (2 если ветвление принято)

Пример 1:

```
002000      HERE:       BRA LTU, BYPASS         ; Если LTU,
002002      NO_LTU:     . . .                   ; перейти на BYPASS
002004                  . . .                   ; Иначе ... продолжить
002006                  . . .
002008                  . . .
00200A                  GOTO THERE
00200C      BYPASS:     . . .
00200E
```

	Перед инструкцией		После инструкции	
PC	00 2000		PC 00 2002	
SR	0001	(C = 1)	SR 0001	(C = 1)

BRA N **Ветвление, если отрицательное**

Синтаксис: {метка:} BRA N, Expr

Операнды: Expr может быть меткой, абсолютным адресом или выражением.
Expr решается компоновщиком в Slit16, где Slit16 ∈ [-32768 ... +32767].

Действие: Условие = N
If (Условие)
(PC + 2) + 2 * Slit16 → PC
NOP → Регистр Инструкции

Влияет на флаги: Не влияет

Код:

0011	0011	nnnn	nnnn	nnnn	nnnn

Описание: Если флаг отрицательного результата установлен, то программа ветвится относительного следующего значения PC. Смещение ветвления задаётся двоичным дополненным числом ‘2 * Slit16’, которое поддерживает ветвление до 32K инструкций вперёд и назад. Значение Slit16 решено компоновщиком, чтобы обеспечить метку, абсолютный адрес или выражение.

Если ветвление принято, новый адрес будет (PC + 2) + 2 * Slit16, поскольку PC увеличился при выборе следующей инструкции. Тогда инструкция становится двухцикловой, с выполнением NOP во втором цикле.

Биты ‘n’ являются 16-битным знаковым литералом, который определяет смещение от (PC + 2) в словах инструкций.

Слова: 1

Циклы: 1 (2 если ветвление принято)

Пример 1:

```
002000      HERE:       BRA N, BYPASS           ; Если N,
002002      NO_N:       . . .                   ; перейти на BYPASS
002004                  . . .                   ; Иначе ... продолжить
002006                  . . .
002008                  . . .
00200A                  GOTO THERE
00200C      BYPASS:     . . .
00200E
```

	Перед инструкцией		После инструкции	
PC	00 2000		PC	00 200C	
SR	0008	(N = 1)	SR	0008	(N = 1)

BRA NC **Ветвление, если нет переноса**

Синтаксис: {метка:} BRA NC, Expr

Операнды: Expr может быть меткой, абсолютным адресом или выражением.
Expr решается компоновщиком в Slit16, где Slit16 ∈ [-32768 ... +32767].

Действие: Условие = !C
If (Условие)
(PC + 2) + 2 * Slit16 → PC
NOP → Регистр Инструкции

Влияет на флаги: Не влияет

Код:

0011	1001	nnnn	nnnn	nnnn	nnnn

Описание: Если флаг переноса сброшен, то программа ветвится относительного следующего значения PC. Смещение ветвления задаётся двоичным дополненным числом ‘2 * Slit16’, которое поддерживает ветвление до 32K инструкций вперёд и назад. Значение Slit16 решено компоновщиком, чтобы обеспечить метку, абсолютный адрес или выражение.

Если ветвление принято, новый адрес будет (PC + 2) + 2 * Slit16, поскольку PC увеличился при выборе следующей инструкции. Тогда инструкция становится двухцикловой, с выполнением NOP во втором цикле.

Биты ‘n’ являются 16-битным знаковым литералом, который определяет смещение от (PC + 2) в словах инструкций.

Слова: 1

Циклы: 1 (2 если ветвление принято)

Пример 1:

```
002000      HERE:       BRA NC, BYPASS          ; Если NC,
002002      NO_NC:      . . .                   ; перейти на BYPASS
002004                  . . .                   ; Иначе ... продолжить
002006                  . . .
002008                  . . .
00200A                  GOTO THERE
00200C      BYPASS:     . . .
00200E
```

	Перед инструкцией		После инструкции	
PC	00 2000		PC	00 2002
SR	0001	(C = 1)	SR	0001 (C = 1)

BRA NN **Ветвление, если неотрицательное**

Синтаксис: {метка:} BRA NN, Expr

Операнды: Expr может быть меткой, абсолютным адресом или выражением.
Expr решается компоновщиком в Slit16, где Slit16 ∈ [-32768 ... +32767].

Действие: Условие = !N
If (Условие)
(PC + 2) + 2 * Slit16 → PC
NOP → Регистр Инструкции

Влияет на флаги: Не влияет

Код:

0011	1011	nnnn	nnnn	nnnn	nnnn

Описание: Если флаг отрицательного результата сброшен, то программа ветвится относительного следующего значения PC. Смещение ветвления задаётся двоичным дополненным числом ‘2 * Slit16’, которое поддерживает ветвление до 32K инструкций вперёд и назад. Значение Slit16 решено компоновщиком, чтобы обеспечить метку, абсолютный адрес или выражение.

Если ветвление принято, новый адрес будет (PC + 2) + 2 * Slit16, поскольку PC увеличился при выборе следующей инструкции. Тогда инструкция становится двухцикловой, с выполнением NOP во втором цикле.

Биты ‘n’ являются 16-битным знаковым литералом, который определяет смещение от (PC + 2) в словах инструкций.

Слова: 1

Циклы: 1 (2 если ветвление принято)

Пример 1:

```
002000      HERE:       BRA NN, BYPASS          ; Если NN,
002002      NO_NN:      . . .                   ; перейти на BYPASS
002004                  . . .                   ; Иначе ... продолжить
002006                  . . .
002008                  . . .
00200A                  GOTO THERE
00200C      BYPASS:     . . .
00200E
```

	Перед инструкцией
PC	00 2000
SR	0000

	После инструкции
PC	00 200C
SR	0000

BRA NOV **Ветвление, если нет переполнения**

Синтаксис: {метка:} BRA NOV, Expr

Операнды: Expr может быть меткой, абсолютным адресом или выражением.
Expr решается компоновщиком в Slit16, где Slit16 ∈ [-32768 ... +32767].

Действие: Условие = !OV
If (Условие)
(PC + 2) + 2 * Slit16 → PC
NOP → Регистр Инструкции

Влияет на флаги: Не влияет

Код:

0011	1000	nnnn	nnnn	nnnn	nnnn

Описание: Если флаг переполнения сброшен, то программа ветвится относительного следующего значения PC. Смещение ветвления задаётся двоичным дополненным числом ‘2 * Slit16’, которое поддерживает ветвление до 32K инструкций вперёд и назад. Значение Slit16 решено компоновщиком, чтобы обеспечить метку, абсолютный адрес или выражение.

Если ветвление принято, новый адрес будет (PC + 2) + 2 * Slit16, поскольку PC увеличился при выборе следующей инструкции. Тогда инструкция становится двухцикловой, с выполнением NOP во втором цикле.

Биты ‘n’ являются 16-битным знаковым литералом, который определяет смещение от (PC + 2) в словах инструкций.

Слова: 1

Циклы: 1 (2 если ветвление принято)

Пример 1:

```
002000      HERE:       BRA NOV, BYPASS         ; Если NOV,
002002      NO_NOV:     . . .                   ; перейти на BYPASS
002004                  . . .                   ; Иначе ... продолжить
002006                  . . .
002008                  . . .
00200A                  GOTO THERE
00200C      BYPASS:     . . .
00200E
```

	Перед инструкцией		После инструкции	
PC	00 2000		PC	00 200C
SR	0008	(N = 1)	SR	0008 (N = 1)

BRA NZ **Ветвление, если не ноль**

Синтаксис: {метка:} BRA NZ, Expr

Операнды: Expr может быть меткой, абсолютным адресом или выражением.
Expr решается компоновщиком в Slit16, где Slit16 ∈ [-32768 ... +32767].

Действие: Условие = !Z
If (Условие)
(PC + 2) + 2 * Slit16 → PC
NOP → Регистр Инструкции

Влияет на флаги: Не влияет

Код:

0011	1010	nnnn	nnnn	nnnn	nnnn

Описание: Если сброшен флаг Z, то программа ветвится относительного следующего значения PC. Смещение ветвления задаётся двоичным дополненным числом ‘2 * Slit16’, которое поддерживает ветвление до 32K инструкций вперёд и назад. Значение Slit16 решено компоновщиком, чтобы обеспечить метку, абсолютный адрес или выражение.

Если ветвление принято, новый адрес будет (PC + 2) + 2 * Slit16, поскольку PC увеличился при выборе следующей инструкции. Тогда инструкция становится двухцикловой, с выполнением NOP во втором цикле.

Биты ‘n’ являются 16-битным знаковым литералом, который определяет смещение от (PC + 2) в словах инструкций.

Слова: 1

Циклы: 1 (2 если ветвление принято)

Пример 1:

```
002000      HERE:       BRA NZ, BYPASS          ; Если NZ,
002002      NO_NZ:      . . .                   ; перейти на BYPASS
002004                  . . .                   ; Иначе ... продолжить
002006                  . . .
002008                  . . .
00200A                  GOTO THERE
00200C      BYPASS:     . . .
00200E
```

	Перед инструкцией		После инструкции	
PC	00 2000		PC	00 2002
SR	0002	(Z = 1)	SR	0002	(Z = 1)

BRA OA **Ветвление, если переполнен аккумулятор A**

Синтаксис: {метка:} BRA OA, Expr

Операнды: Expr может быть меткой, абсолютным адресом или выражением.
Expr решается компоновщиком в Slit16, где Slit16 ∈ [-32768 ... +32767].

Действие: Условие = OA
If (Условие)
(PC + 2) + 2 * Slit16 → PC
NOP → Регистр Инструкции

Влияет на флаги: Не влияет

Код:

0000	1100	nnnn	nnnn	nnnn	nnnn

Описание: Если флаг переполнения аккумулятора A (OA) установлен, то программа ветвится относительного следующего значения PC. Смещение ветвления задаётся двоичным дополненным числом ‘2 * Slit16’, которое поддерживает ветвление до 32K инструкций вперёд и назад. Значение Slit16 решено компоновщиком, чтобы обеспечить метку, абсолютный адрес или выражение.

Если ветвление принято, новый адрес будет (PC + 2) + 2 * Slit16, поскольку PC увеличился при выборе следующей инструкции. Тогда инструкция становится двухцикловой, с выполнением NOP во втором цикле.

Биты ‘n’ являются 16-битным знаковым литералом, который определяет смещение от (PC + 2) в словах инструкций.

Примечание: Ассемблер преобразует указанную метку в смещение, которое нужно использовать.

Слова: 1

Циклы: 1 (2 если ветвление принято)

Пример 1:

```
002000      HERE:       BRA OA, BYPASS          ; Если OA,
002002      NO_OA:      . . .                   ; перейти на BYPASS
002004                  . . .                   ; Иначе ... продолжить
002006                  . . .
002008                  . . .
00200A                  GOTO THERE
00200C      BYPASS:     . . .
00200E
```

Перед инструкцией

PC	00 2000
SR	8800

(OA,OAB = 1)

После инструкции

PC	00 200C
SR	8800

(OA,OAB = 1)

BRA OB **Ветвление, если переполнен аккумулятор B**

Синтаксис: {метка:} BRA OB, Expr

Операнды: Expr может быть меткой, абсолютным адресом или выражением.
Expr решается компоновщиком в Slit16, где Slit16 ∈ [-32768 ... +32767].

Действие: Условие = OB
If (Условие)
(PC + 2) + 2 * Slit16 → PC
NOP → Регистр Инструкции

Влияет на флаги: Не влияет

Код:

0000	1101	nnnn	nnnn	nnnn	nnnn

Описание: Если флаг переполнения аккумулятора B (OB) установлен, то программа ветвится относительного следующего значения PC. Смещение ветвления задаётся двоичным дополненным числом ‘2 * Slit16’, которое поддерживает ветвление до 32K инструкций вперёд и назад. Значение Slit16 решено компоновщиком, чтобы обеспечить метку, абсолютный адрес или выражение.

Если ветвление принято, новый адрес будет (PC + 2) + 2 * Slit16, поскольку PC увеличился при выборе следующей инструкции. Тогда инструкция становится двухцикловой, с выполнением NOP во втором цикле.

Биты ‘n’ являются 16-битным знаковым литералом, который определяет смещение от (PC + 2) в словах инструкций.

Слова: 1

Циклы: 1 (2 если ветвление принято)

Пример 1:

```
002000      HERE:       BRA OB, BYPASS          ; Если OB,
002002      NO_OB:      . . .                   ; перейти на BYPASS
002004                  . . .                   ; Иначе ... продолжить
002006                  . . .
002008                  . . .
00200A                  GOTO THERE
00200C      BYPASS:     . . .
00200E
```

	Перед инструкцией		После инструкции	
PC	00 2000		00 2002	
SR	8800	(OA,OAB = 1)	8800	(OA,OAB = 1)

BRA OV **Ветвление, если не ноль**

Синтаксис: {метка:} BRA OV, Expr

Операнды: Expr может быть меткой, абсолютным адресом или выражением.
Expr решается компоновщиком в Slit16, где Slit16 ∈ [-32768 ... +32767].

Действие: Условие = OV
If (Условие)
(PC + 2) + 2 * Slit16 → PC
NOP → Регистр Инструкции

Влияет на флаги: Не влияет

Код:

0011	0000	nnnn	nnnn	nnnn	nnnn

Описание: Если флаг переполнения установлен, то программа ветвится относительного следующего значения PC. Смещение ветвления задаётся двоичным дополненным числом ‘2 * Slit16’, которое поддерживает ветвление до 32K инструкций вперёд и назад. Значение Slit16 решено компоновщиком, чтобы обеспечить метку, абсолютный адрес или выражение.

Если ветвление принято, новый адрес будет (PC + 2) + 2 * Slit16, поскольку PC увеличился при выборе следующей инструкции. Тогда инструкция становится двухцикловой, с выполнением NOP во втором цикле.

Биты ‘n’ являются 16-битным знаковым литералом, который определяет смещение от (PC + 2) в словах инструкций.

Слова: 1

Циклы: 1 (2 если ветвление принято)

Пример 1:

```
002000      HERE:       BRA OV, BYPASS          ; Если OV,
002002      NO_OV:      . . .                   ; перейти на BYPASS
002004                  . . .                   ; Иначе ... продолжить
002006                  . . .
002008                  . . .
00200A                  GOTO THERE
00200C      BYPASS:     . . .
00200E
```

	Перед инструкцией		После инструкции	
PC	00 2000		PC	00 2002
SR	0002	(Z = 1)	SR	0002	(Z = 1)

BRA SA **Ветвление, если насыщен аккумулятор A**

Синтаксис: {метка:} BRA SA, Expr

Операнды: Expr может быть меткой, абсолютным адресом или выражением.
Expr решается компоновщиком в Slit16, где Slit16 ∈ [-32768 ... +32767].

Действие: Условие = SA
If (Условие)
(PC + 2) + 2 * Slit16 → PC
NOP → Регистр Инструкции

Влияет на флаги: Не влияет

Код:

0000	1110	nnnn	nnnn	nnnn	nnnn

Описание: Если флаг насыщения аккумулятора A (SA) установлен, то программа ветвится относительного следующего значения PC. Смещение ветвления задаётся двоичным дополненным числом ‘2 * Slit16’, которое поддерживает ветвление до 32K инструкций вперёд и назад. Значение Slit16 решено компоновщиком, чтобы обеспечить метку, абсолютный адрес или выражение.

Если ветвление принято, новый адрес будет (PC + 2) + 2 * Slit16, поскольку PC увеличился при выборе следующей инструкции. Тогда инструкция становится двухцикловой, с выполнением NOP во втором цикле.

Биты ‘n’ являются 16-битным знаковым литералом, который определяет смещение от (PC + 2) в словах инструкций.

Слова: 1

Циклы: 1 (2 если ветвление принято)

Пример 1:

```
002000      HERE:       BRA SA, BYPASS          ; Если SA,
002002      NO_SA:      . . .                   ; перейти на BYPASS
002004                  . . .                   ; Иначе ... продолжить
002006                  . . .
002008                  . . .
00200A                  GOTO THERE
00200C      BYPASS:     . . .
00200E
```

	Перед инструкцией		После инструкции	
PC	00 2000		00 200C	
SR	2400	(SA,SAB = 1)	2400	(SA,SAB = 1)

BRA SB **Ветвление, если насыщен аккумулятор B**

Синтаксис: {метка:} BRA SB, Expr

Операнды: Expr может быть меткой, абсолютным адресом или выражением.
Expr решается компоновщиком в Slit16, где Slit16 ∈ [-32768 ... +32767].

Действие: Условие = SB
If (Условие)
(PC + 2) + 2 * Slit16 → PC
NOP → Регистр Инструкции

Влияет на флаги: Не влияет

Код:

0000	1111	nnnn	nnnn	nnnn	nnnn

Описание: Если флаг насыщения аккумулятора B (SB) установлен, то программа ветвится относительного следующего значения PC. Смещение ветвления задаётся двоичным дополненным числом ‘2 * Slit16’, которое поддерживает ветвление до 32K инструкций вперёд и назад. Значение Slit16 решено компоновщиком, чтобы обеспечить метку, абсолютный адрес или выражение.

Если ветвление принято, новый адрес будет (PC + 2) + 2 * Slit16, поскольку PC увеличился при выборе следующей инструкции. Тогда инструкция становится двухцикловой, с выполнением NOP во втором цикле.

Биты ‘n’ являются 16-битным знаковым литералом, который определяет смещение от (PC + 2) в словах инструкций.

Слова: 1

Циклы: 1 (2 если ветвление принято)

Пример 1:

```
002000      HERE:       BRA SB, BYPASS          ; Если SB,
002002      NO_SB:      . . .                   ; перейти на BYPASS
002004                  . . .                   ; Иначе ... продолжить
002006                  . . .
002008                  . . .
00200A                  GOTO THERE
00200C      BYPASS:     . . .
00200E
```

Перед инструкцией

PC	00 2000
SR	0000

После инструкции

PC	00 2002
SR	0000

BRA Z **Ветвление, если ноль**

Синтаксис: {метка:} BRA OV, Expr

Операнды: Expr может быть меткой, абсолютным адресом или выражением.
Expr решается компоновщиком в Slit16, где Slit16 ∈ [-32768 ... +32767].

Действие: Условие = Z
If (Условие)
(PC + 2) + 2 * Slit16 → PC
NOP → Регистр Инструкции

Влияет на флаги: Не влияет

Код:

0011	0010	nnnn	nnnn	nnnn	nnnn

Описание: Если флаг нулевого результат (Z) установлен, то программа ветвится относительного следующего значения PC. Смещение ветвления задаётся двоичным дополненным числом ‘2 * Slit16’, которое поддерживает ветвление до 32K инструкций вперёд и назад. Значение Slit16 решено компоновщиком, чтобы обеспечить метку, абсолютный адрес или выражение.

Если ветвление принято, новый адрес будет (PC + 2) + 2 * Slit16, поскольку PC увеличился при выборе следующей инструкции. Тогда инструкция становится двухцикловой, с выполнением NOP во втором цикле.

Биты ‘n’ являются 16-битным знаковым литералом, который определяет смещение от (PC + 2) в словах инструкций.

Слова: 1

Циклы: 1 (2 если ветвление принято)

Пример 1:

```
002000      HERE:       BRA Z, BYPASS           ; Если Z,
002002      NO_Z:       . . .                   ; перейти на BYPASS
002004                  . . .                   ; Иначе ... продолжить
002006                  . . .
002008                  . . .
00200A                  GOTO THERE
00200C      BYPASS:     . . .
00200E
```

	Перед инструкцией		После инструкции	
PC	00 2000		PC	00 200C
SR	0002	(Z = 1)	SR	0002	(Z = 1)

BSET **Установить бит f**

Синтаксис: {метка:} BSET{.B} f, #bit4

Операнды: f ∈ [0 ... 8191] для операции с байтом
f ∈ [0 ... 8190] (только для чётного) для операции со словом
bit4 ∈ [0 ... 7] для операции с байтом
bit4 ∈ [0 ... 15] для операции с байтом

Действие: 1 → f<bit4>

Влияет на флаги: Не влияет

Код:

1010	1000	bbbf	ffff	ffff	fffb

Описание: Установить бит в файловом регистре f определённый через ‘bit4’. Нумерация битов начинается с наименьшего значащего бита (bit 0) и продвигается к наиболее значащему биту (бит 7 для операций с байтом, бит 15 для операций со словом).

Биты ‘b’ выбирают значение bit4 битовой позиции, которая должна быть очищена.
Биты ‘f’ выбирают адрес файлового регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.
2: Когда эта инструкция используется в режиме слова, файловый регистр адресуется с выравниванием по слову.
3: Когда эта инструкция используется в байтном режиме, ‘bit4’ должен быть между 0 и 7.

Слова: 1

Циклы: 1

Пример 1:

```
BSET.B      0x600, #0x3          ; Установить бит 3 в 0x600
```

	Перед инструкцией		После инструкции
Данные 0600	F234	Данные 0600	FA34
SR	0000	SR	0000

Пример 2:

```
BSET        0x444, #0xF          ; Установить бит 15 в 0x444
```

	Перед инструкцией		После инструкции
Данные 0444	5604	Данные 0444	D604
SR	0000	SR	0000

BSET **Установить бит в Ws**

Синтаксис:	{метка:}	BSET{.B}	Ws, [Ws], [Ws++], [Ws--], [++Ws], [--Ws],	#bit4

Операнды: Ws ∈ [W0 ... W15]
bit4 ∈ [0 ... 7] для операции с байтом
bit4 ∈ [0 ... 15] для операции со словом

Действие: 1 → Ws<bit4>

Влияет на флаги: Не влияет

Код:

1010	0000	bbbb	0B00	0ppp	ssss

Описание: Установить бит в регистре Ws определённый через ‘bit4’. Нумерация битов начинается с наименьшего значащего бита (bit 0) и продвигается к наиболее значащему биту (бит 7 для операций с байтом, бит 15 для операций со словом). Прямая или косвенная регистровая адресация может быть использована для Ws.

Биты ‘b’ выбирают значение bit4 позиции бита, который должен быть очищен.
Бит ‘B’ выбирает операцию с байтом или словом (‘0’ для слова, ‘1’ для байта).
Биты ‘s’ выбирают регистр источник/назначение.
Биты ‘p’ выбирают режим адреса источника.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.
2: Когда эта инструкция используется в режиме слова, файловый регистр адресуется с выравниванием по слову.
3: Когда эта инструкция используется в байтном режиме, ‘bit4’ должен быть между 0 и 7.

Слова: 1

Циклы: 1

Пример 1:

```
BSET.B      W3, #0x7            ; Установить бит 7 в W3
```

	Перед инструкцией		После инструкции
W3	0026	W3	00A6
SR	0000	SR	0000

Пример 2:

```
BSET        [W4++], #0x0        ; Установить бит 0 в [W4]
                                ; Пост-инкремент W4
```

	Перед инструкцией		После инструкции
W4	6700	W4	6702
Данные 6700	1734	Данные 6700	1735
SR	0000	SR	0000

BSW **Записать бит в Ws**

Синтаксис: {метка:} BCLR{.B} Ws, [Ws], [Ws++], [Ws--], [++Ws], [--Ws], Wb

Операнды: Ws ∈ [W0 ... W15]
Wb ∈ [W0 ... W15]

Действие: Для операции “.C”:
C → Ws<(Wb)>
Для операции “.Z” (по умолчанию):
Z → Ws<(Wb)>

Влияет на флаги: Не влияет

Код:

1010	1101	Zwww	w000	0ppp	ssss

Описание: Записывает значение флага C или Z регистра STATUS в бит (Wb) регистра. Биты нумеруются начиная с наименьшего значимого бита (бит 0) и продвигаясь к наиболее значимому биту (бит 15) рабочего регистра. Только четыре наименее значимых бита Wb используются для определения номера бита места назначения. Прямая регистровая адресация может быть использована для Wb, и любая, прямая или косвенная регистровая адресация может быть использована для Ws.

Бит ‘Z’ выбирает C или Z флаг как источник.
Биты ‘w’ выбирают адрес бита выбранного регистра.
Биты ‘p’ выбирают режим адреса источника.
Биты ‘s’ выбирают регистр источник.

Примечание 1: Эта инструкция работает только в режиме слова. Если расширение не определено, то подразумевается операция “.Z”.

Слова: 1

Циклы: 1

Пример 1:

```
BSW.C       W2, W3      ; Присвоить биту W3 в W2 значение бита C.
```

	Перед инструкцией		После инструкции	
W2	F234		7234	
W3	111F		111F	
SR	0002	(Z = 1, C = 0)	0002	(Z = 1, C = 0)

Пример 2:

```
BSW.Z       W2, W3      ; Присвоить биту W3 в W2 значение бита Z.
```

	Перед инструкцией		После инструкции	
W2	E235		E234	
W3	0550		0550	
SR	0002	(Z = 1, C = 0)	0002	(Z = 1, C = 0)

Пример 3:

```
BSW.C       [++W0], W6  ; Присвоить биту W6 в [W0++] значение бита C.
```

	Перед инструкцией		После инструкции	
W0	1000		1002	
W6	34A3		34A3	
Данные 1002	2380		Данные 1002	2388	
SR	0001	(Z = 0, C = 1)	0001	(Z = 0, C = 1)

Пример 4:

```
BSW.C       [W1--], W5    ; Присвоить биту W5 в [W1] дополнение от бита Z
                          ; Пост-декремент W1
```

	Перед инструкцией	
W1	1000	
W5	888B	
Данные 1000	C4DD	
SR	0001	(C = 1)

	После инструкции	
W1	0FFE	
W5	888B	
Данные 1000	CCDD	
SR	0001	(C = 1)

BTG **Инвертировать бит в f**

Синтаксис: {метка:} BTG{.B} f, #bit4

Операнды: f ∈ [0 ... 8191] для операции с байтом
f ∈ [0 ... 8190] (только чётный) для операции со словом
bit4 ∈ [0 ... 7] для операции с байтом
bit4 ∈ [0 ... 15] для операции со словом

Действие: not((f)<bit4>) → (f)<bit4>

Влияет на флаги: Не влияет

Код:

1010	1010	bbbf	ffff	ffff

Описание: Бит ‘bit4’ в файловом регистре ‘f’ будет инвертирован (дополнен). Для операнда bit4, нумерация бит начинается с наименьшего значимого бита (бит 0) и продвигается к наиболее значимому биту (бит 7 для операций с байтом, бит 15 для операций со словом) байта.

Биты ‘b’ выбирают значение bit4, положение инвертируемого бита.
Биты ‘f’ выбирают адрес файлового регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.
2: Когда эта инструкция используется в режиме слова, файловый регистр адресуется с выравниванием по слову.
3: Когда эта инструкция используется в байтном режиме, ‘bit4’ должен быть между 0 и 7.

Слова: 1

Циклы: 1

Пример 1:

```
BTG.B       W0x1001, #0x4          ; Инвертировать бит 4 в 0x1001
```

	Перед инструкцией		После инструкции
Данные 1000	F234	Данные 1000	E234
SR	0000	SR	0000

Пример 2:

```
BTG         0x1660, #0x8           ; Инвертировать бит 8 в RAM660
```

	Перед инструкцией		После инструкции
Данные 1660	5606	Данные 1660	5706
SR	0000	SR	0000

BTG **Инвертировать бит в Ws**

Синтаксис:	{метка:}	BTG{.B}	Ws, [Ws], [Ws++], [Ws--], [++Ws], [--Ws],	#bit4

Операнды: Ws ∈ [W0 ... W15]
bit4 ∈ [0 ... 7] для операции с байтом
bit4 ∈ [0 ... 15] для операции со словом

Действие: not((Ws)<bit4>) → Ws<bit4>

Влияет на флаги: Не влияет

Код:

1010	0010	bbbb	0B00	0ppp	ssss

Описание: Бит ‘bit4’ регистра Ws будет инвертирован (дополнен). Для операнда bit4, нумерация бит начинается с наименьшего значимого бита (бит 0) и продвигается к наиболее значимому биту (бит 7 для операций с байтом, бит 15 для операций со словом) байта. Прямая или косвенная регистровая адресация может быть использована для Ws.

Биты ‘b’ выбирают значение bit4, положения инвертируемого бита.
Бит ‘B’ выбирает операцию с байтом или словом (‘0’ для слова, ‘1’ для байта).
Биты ‘s’ выбирают регистр источник/назначение.
Биты ‘p’ выбирают режим адреса источника.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

2: Когда эта инструкция используется в режиме слова, файловый регистр адресуется с выравниванием по слову.

3: Когда эта инструкция используется в байтном режиме, ‘bit4’ должен быть между 0 и 7.

Слова: 1

Циклы: 1

Пример 1:

```
BTG     W2, #0x0        ; Инвертировать бит 0 в W2
```

	Перед инструкцией		После инструкции
W2	F234	W2	F235
SR	0000	SR	0000

Пример 2:

```
BTG     [W0++], #0x0    ; Инвертировать бит 0 в [W0]
                        ; Пост-инкремент W0
```

	Перед инструкцией		После инструкции
W0	2300	W0	2302
Данные 2300	5606	Данные 2300	5607
SR	0000	SR	0000

BTSC **Тестировать бит f, пропустить если очищен**

Синтаксис: {метка:} BTSC{.B} f, #bit4

Операнды: f ∈ [0 ... 8191] для операции с байтом
f ∈ [0 ... 8190] (только чётный) для операции со словом
bit4 ∈ [0 ... 7] для операции с байтом
bit4 ∈ [0 ... 15] для операции со словом

Действие: Тестировать (f)<bit4>, пропустить если очищен

Влияет на флаги: Не влияет

Код:

1010	1111	bbbf	ffff	ffff	fffb

Описание: Бит ‘bit4’ тестируется в файловом регистре. Ясли тестируемый бит равен ‘0’, следующая инструкция (выбранная в течении выполнения текущей инструкции) будет отвергнута и взамен её будет выполнена инструкция NOP. Если тестируемый бит равен ‘1’ следующая инструкция выполняется в обычном порядке. В любом случае содержимое файлового регистра не изменяется. Для операнда bit4, нумерация битов начинается с наименьшего значащего бита (bit 0) и продолжается к наибольшему значащему биту (бит 7 для операций с байтом, бит 15 для операций со словом).

Биты ‘b’ выбирают значение bit4, положения тестируемого бита.
Биты 'f’ выбирают адрес файлового регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

2: Когда эта инструкция используется в режиме слова, файловый регистр адресуется с выравниванием по слову.

3: Когда эта инструкция используется в байтном режиме, ‘bit4’ должен быть между 0 и 7.

Слова: 1

Циклы: 1 (2 или 3)

Пример 1:

```
002000      HERE:       BTSC.B      0x1201, #2  ; Если бит 2 в 0x1201
002002                  GOTO        BYPASS      ; равен 0,
002004                  . . .                   ; пропустить GOTO
002006                  . . .
002008      BYPASS:     . . .
00200A                  . . .
```

	Перед инструкцией		После инструкции
PC	00 2000	PC	00 2002
Данные 1200	264F	Данные 1200	264F
SR	0000	SR	0000

Пример 2:

```
002000      HERE:       BTSC        0x804, #14  ; Если бит 14 в 0x804
002002                  GOTO        BYPASS      ; равен 0,
002004                  . . .                   ; пропустить GOTO
002006                  . . .
002008      BYPASS:     . . .
00200A                  . . .
```

	Перед инструкцией		После инструкции
PC	00 2000	PC	00 2004
Данные 0804	2647	Данные 0804	2647
SR	0000	SR	0000

BTSC **Тестировать бит Ws, пропустить если очищен**

Синтаксис: {метка:} BTSC Ws, [Ws], [Ws++], [Ws--], [++Ws], [--Ws], #bit4

Операнды: Ws ∈ [W0 ... W15]
bit4 ∈ [0 ... 15]

Действие: Test (Ws)<bit4>, пропустить, если очищен

Влияет на флаги: Не влияет

Код:

1010	0111	bbbb	0000	0ppp	ssss

Описание: Бит ‘bit4’ тестируется в регистре Ws. Ясли тестируемый бит равен ‘0’, следующая инструкция (выбранная в течении выполнения текущей инструкции) будет отвергнута и взамен её будет выполнена инструкция NOP. Если тестируемый бит равен ‘1’ следующая инструкция выполняется в обычном порядке. В любом случае содержимое регистра Ws не изменяется. Для операнда bit4, нумерация битов начинается с наименьшего значащего бита (bit 0) и продолжается к наибольшему значащему биту (бит 7 для операций с байтом, бит 15 для операций со словом).

Биты ‘b’ выбирают значение bit4, положения тестируемого бита.
Биты ‘p’ выбирают режим адреса источника.
Биты ‘s’ выбирают регистр источник.

Примечание: Инструкция работает только в режиме слова.

Слова: 1

Циклы: 1 (2 или 3 если следующая инструкция пропущена)

Пример 1:

```
002000      HERE:       BTSC        W0, #0x0      ; Если бит 0 в W0 равен
002002                  GOTO        BYPASS        ; нулю, пропустить GOTO
002004                  . . .
002006                  . . .
002008      BYPASS:     . . .
00200A                  . . .
```

	Перед инструкцией		После инструкции
PC	00 2000	PC	00 2002
W0	264F	W0	264F
SR	0000	SR	0000

Пример 2:

```
002000      HERE:       BTSC        W6, #0xF      ; Если бит 15 в W6
002002                  GOTO        BYPASS        равен : 0, пропустить
002004                  . . .                     GOTO
002006                  . . .
002008      BYPASS:     . . .
00200A                  . . .
```

	Перед инструкцией		После инструкции
PC	00 2000	PC	00 2004
W6	264F	W6	264F
SR	0000	SR	0000

Пример 3:

```
003400      HERE:       BTSC        [W6++],#0xC  ; Если бит 12 в [W6]
003402                  GOTO        BYPASS       ; равен 0, пропустить
003404                  . . .                    ; GOTO
003406                  . . .                    ; Пост-инкремент W6
003408      BYPASS:     . . .
00340A                  . . .
```

	Перед инструкцией
PC	00 3400
W6	1800
Данные 1800	1000
SR	0000

	После инструкции
PC	00 3402
W6	1802
Данные 1800	1000
SR	0000

BTSS **Тестировать бит f, пропустить если установлен**

Синтаксис: {метка:} BTSS{.B} f, #bit4

Операнды: f ∈ [0 ... 8191] для операции с байтом
f ∈ [0 ... 8190] (только чётный) для операции со словом
bit4 ∈ [0 ... 7] для операции с байтом
bit4 ∈ [0 ... 15] для операции со словом

Действие: Тестировать (f)<bit4>, пропустить если установлен

Влияет на флаги: Не влияет

Код:

1010	1110	bbbf	ffff	ffff	fffb

Описание: Бит ‘bit4’ тестируется в файловом регистре. Ясли тестируемый бит равен ‘1’, следующая инструкция (выбранная в течении выполнения текущей инструкции) будет отвергнута и взамен её будет выполнена инструкция NOP. Если тестируемый бит равен ‘0’ следующая инструкция выполняется в обычном порядке. В любом случае содержимое файлового регистра не изменяется. Для операнда bit4, нумерация битов начинается с наименьшего значащего бита (bit 0) и продолжается к наибольшему значащему биту (бит 7 для операций с байтом, бит 15 для операций со словом).

Биты ‘b’ выбирают значение bit4, положения тестируемого бита.
Биты 'f’ выбирают адрес файлового регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

2: Когда эта инструкция используется в режиме слова, файловый регистр адресуется с выравниванием по слову.

3: Когда эта инструкция используется в байтном режиме, ‘bit4’ должен быть между 0 и 7.

Слова: 1

Циклы: 1 (2 или 3)

Пример 1:

```
007100      HERE:       BTSS.B      0x1401, #1   ; Если бит 1 в 0x1401
007102                  CLR         WREG         ; равен 1, то
007104                  . . .                    ; не очищать WREG
```

	Перед инструкцией		После инструкции
PC	00 7100	PC	00 7104
Данные 1400	0280	Данные 1400	0280
SR	0000	SR	0000

Пример 2:

```
007100      HERE:       BTSS        0x890, #9    ; Если бит 9 в 0x890
007102                  GOTO        BYPASS       ; равен 1,
007104                  . . .                    ; пропустить GOTO
007106      BYPASS:     . . .
```

	Перед инструкцией		После инструкции
PC	00 7100	PC	00 7102
Данные 0890	00FE	Данные 0890	00FE
SR	0000	SR	0000

BTSS **Тестировать бит Ws, пропустить если установлен**

Синтаксис: {метка:} BTSS Ws, [Ws], [Ws++], [Ws--], [++Ws], [--Ws], #bit4

Операнды: Ws ∈ [W0 ... W15]
bit4 ∈ [0 ... 15]

Действие: Test (Ws)<bit4>, пропустить, если установлен

Влияет на флаги: Не влияет

Код:

1010	0110	bbbb	0000	0ppp	ssss

Описание: Бит ‘bit4’ тестируется в регистре Ws. Ясли тестируемый бит равен ‘1’, следующая инструкция (выбранная в течении выполнения текущей инструкции) будет отвергнута и взамен её будет выполнена инструкция NOP. Если тестируемый бит равен ‘0’ следующая инструкция выполняется в обычном порядке. В любом случае содержимое регистра Ws не изменяется. Для операнда bit4, нумерация битов начинается с наименьшего значащего бита (bit 0) и продолжается к наибольшему значащему биту (бит 7 для операций с байтом, бит 15 для операций со словом).

Биты ‘b’ выбирают значение bit4, положения тестируемого бита.
Биты ‘p’ выбирают режим адреса источника.
Биты ‘s’ выбирают регистр источник.

Примечание: Инструкция работает только в режиме слова.

Слова: 1

Циклы: 1 (2 или 3 если следующая инструкция пропущена)

Пример 1:

```
002000      HERE:       BTSS        W0, #0x0    ; Если бит 0 в W0
002002                  GOTO        BYPASS      равен
002004                  . . .                   ; 1, пропустить GOTO
002006                  . . .
002008      BYPASS:     . . .
00200A                  . . .
```

	Перед инструкцией		После инструкции
PC	00 2000	PC	00 2002
W0	264F	W0	264F
SR	0000	SR	0000

Пример 2:

```
002000      HERE:       BTSS        W6, #0xF    ; Если бит 15 в W6
002002                  GOTO        BYPASS      ; равен 1, пропустить
002004                  . . .                   ; GOTO
002006                  . . .
002008      BYPASS:     . . .
00200A                  . . .
```

	Перед инструкцией		После инструкции
PC	00 2000	PC	00 2002
W6	264F	W6	264F
SR	0000	SR	0000

Пример 3:

```
003400      HERE:       BTSS        [W6++],#0xC  ; Если бит 12 в [W6]
003402                  GOTO        BYPASS       ; равен 1, пропустить
003404                  . . .                    ; GOTO
003406                  . . .                    ; Пост-инкремент W6
003408      BYPASS:     . . .
00340A                  . . .
```

	Перед инструкцией
PC	00 3400
W6	1800
Данные 1800	1000
SR	0000

	После инструкции
PC	00 3404
W6	1802
Данные 1800	1000
SR	0000

BTST **Тестировать бит f**

Синтаксис: {метка:} BTST{.B} f, #bit4

Операнды: f ∈ [0 ... 8191] для операции с байтом
f ∈ [0 ... 8190] (только чётный) для операции со словом
bit4 ∈ [0 ... 7] для операции с байтом
bit4 ∈ [0 ... 15] для операции со словом

Действие: $\overline{(f)<bit4>} \rightarrow Z$

Влияет на флаги: Z

Код:

1010	1011	bbbf	ffff	ffff	fffb

Описание: Бит ‘bit4’ тестируется в файловом регистре и дополнение тестируемого бита запоминается в флаг Z регистра STATUS. Содержимое файлового регистра не изменяется. Для операнда bit4, нумерация битов начинается с наименьшего значащего бита (bit 0) и продолжается к наибольшему значащему биту (бит 7 для операций с байтом, бит 15 для операций со словом).

Биты ‘b’ выбирают значение bit4, положения тестируемого бита.
Биты 'f’ выбирают адрес файлового регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.
2: Когда эта инструкция используется в режиме слова, файловый регистр адресуется с выравниванием по слову.
3: Когда эта инструкция используется в байтном режиме, ‘bit4’ должен быть между 0 и 7.

Слова: 1

Циклы: 1

Пример 1:

```
BTST.B      0x1201, #0x3          ; Установить Z = дополнению от
                                  ; бита 3 в 0x1201
```

	Перед инструкцией		После инструкции	
Данные 1200	F7FF	Данные 1200	F7FF	
SR	0000	SR	0002	(Z = 1)

Пример 2:

```
BTST        0x1302, #0x7          ; Set Z = complement of
                                  ; bit 7 in 0x1302
```

	Перед инструкцией		После инструкции	
Данные 1302	F7FF	Данные 1302	F7FF	
SR	0002	(Z = 1) SR	0000	

BTST **Тестировать бит в Ws**

Синтаксис: {метка:} BTST.C / BTST.Z Ws, / [Ws], / [Ws++], / [Ws--], / [++Ws], / [--Ws], #bit4

Операнды: Ws ∈ [W0 ... W15]
bit4 ∈ [0 ... 15]

Действие: Для операции ".C":
(Ws)<bit4> → C
Для операции ".Z" (по умолчанию):
$\overline{(Ws)<bit4>} \rightarrow Z$

Влияет на флаги: Z или C

Код:

1010	0011	bbbb	Z000	0ppp	ssss

Описание: Бит 'bit4' тестируется в регистре Ws. Если определена опция ".Z" инструкции, то дополнение тестируемого бита запоминается в флаг Z регистра STATUS. Если определена опция ".C" инструкции, то значение тестируемого бита запоминается в флаг C регистра STATUS. В любом случае содержимое Ws не изменяется. Для операнда bit4, нумерация битов начинается с наименьшего значащего бита (bit 0) и продолжается к наибольшему значащему биту (бит 7 для операций с байтом, бит 15 для операций со словом). Любая, прямая или косвенная регистровая адресация может быть использована для Ws.

Биты 'b' выбирают значение bit4, положения тестируемого бита.
Бит 'Z' выбирает флаг C или Z как место назначения.
Биты 'p' выбирают режим адресации источника.
Биты 's' выбирают регистр источник.

Примечание: Эта инструкция работает только в режиме слова. Если расширение не указано, то подразумевается операция ".Z".

Слова: 1

Циклы: 1

Пример 1:

```
BTST.C      [W0++], #0x3          ; Установить C = биту 3 в [W0]
                                  ; Пост-инкремент W0
```

	Перед инструкцией		После инструкции	
W0	1200		W0	1202
Данные 1200	FFF7		Данные 1200	FFF7
SR	0001	(C = 1)	SR	0000

Пример 2:

```
BTST.Z      W0, #0x7              ; Установить Z = дополнение от
                                  ; бита 7 в W0
```

	Перед инструкцией		После инструкции	
W0	F234		W0	F234
SR	0000		SR	0002	(Z = 1)

BTST **Тестировать бит в Ws**

Синтаксис: {метка:} BTST.C / BTST.Z Ws, [Ws], [Ws++], [Ws--], [++Ws], [--Ws], Wb

Операнды: Ws ∈ [W0 ... W15]
Wb ∈ [W0 ... W15]

Действие: Для операции ".C":
(Ws)<(Wb)> → C
Для операции ".Z" (по умолчанию):
$\overline{(Ws)<(Wb)>} \rightarrow Z$

Влияет на флаги: Z или C

Код:

1010	0101	Zwww	w000	0ppp	ssss

Описание: Бит (Wb) тестируется в регистре Ws. Если определена опция ".Z" инструкции, то дополнение тестируемого бита запоминается в флаг Z регистра STATUS. Если определена опция ".C" инструкции, то значение тестируемого бита запоминается в флаг C регистра STATUS. В любом случае содержимое Ws не изменяется.
Только четыре младших значащих бита Wb используются для определения номера бита. Нумерация битов начинается с наименьшего значащего бита (bit 0) и продолжается к наибольшему значащему биту (бит 7 для операций с байтом, бит 15 для операций со словом). Прямая или косвенная регистровая адресация может быть использована для Ws.

Бит 'Z' выбирает флаг C или Z как место назначения.
Биты 'w' выбирают адрес регистра в котором выбирается бит.
Биты 'p' выбирают режим адресации источника.
Биты 's' выбирают регистр источник.

Примечание: Эта инструкция работает только в режиме слова. Если расширение не указано, то подразумевается операция ".Z".

Слова: 1

Циклы: 1

Пример 1:

```
BTST.C      W2, W3      ; Установить C = биту W3 регистра W2
```

	Перед инструкцией			После инструкции
W2	F234		W2	F234
W3	2368		W3	2368
SR	0001	(C = 1)	SR	0000

Пример 2:

```
BTST.Z      [W0++], W1  ; Установить Z = дополнению от бита W1 в [W0],
                        ; Пост-инкремент W0
```

	Перед инструкцией			После инструкции
W0	1200		W0	1202
W1	CCC0		W1	CCC0
Данные 1200	6243		Данные 1200	6243
SR	0000	(Z = 1)	SR	0002

BTSTS **Тестировать/установить бит f**

Синтаксис: {метка:} BTSTS{.B} f, #bit4

Операнды: f ∈ [0 ... 8191] для операции с байтом
f ∈ [0 ... 8190] (только чётный) для операции со словом
bit4 ∈ [0 ... 7] для операции с байтом
bit4 ∈ [0 ... 15] для операции со словом

Действие: $\overline{(f)<bit4>} \rightarrow Z$
1 → (f)<bit4>

Влияет на флаги: Z

Код:

1010	1100	bbbf	ffff	ffff	fffb

Описание: Бит ‘bit4’ тестируется в файловом регистре и дополнение тестируемого бита запоминается в флаг Z регистра STATUS. Затем тестируемый бит файлового регистра устанавливается в '1'. Для операнда bit4, нумерация битов начинается с наименьшего значащего бита (bit 0) и продолжается к наибольшему значащему биту (бит 7 для операций с байтом, бит 15 для операций со словом).

Биты ‘b’ выбирают значение bit4, положения тестируемого бита.
Биты 'f’ выбирают адрес файлового регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.
2: Когда эта инструкция используется в режиме слова, файловый регистр адресуется с выравниванием по слову.
3: Когда эта инструкция используется в байтном режиме, ‘bit4’ должен быть между 0 и 7.

Слова: 1

Циклы: 1

Пример 1:

```
BTSTS.B     0x1201,     ; Установить Z = дополнению от бита 3 в 0x1201,
            #0x3        ; затем установить бит 3 в 0x1201 = 1
```

	Перед инструкцией		После инструкции	
Данные 1200	F7FF	Данные 1200	FFFF	
SR	0000	SR	0002	(Z = 1)

Пример 2:

```
BTSTS       0x808, #15  ; Установить Z = дополнению от бита 15 в 0x808,
                        ; затем установить бит 3 в 0x808 = 1
```

	Перед инструкцией		После инструкции	
RAM300	8050	RAM300	8050	
SR	0002	(Z = 1) SR	0000	

BTSTS **Тестировать/установить бит в Ws**

Синтаксис: {метка:} BTSTS.C Ws, #bit4
BTSTS.Z [Ws],
[Ws++],
[Ws--],
[++Ws],
[--Ws],

Операнды: Ws ∈ [W0 ... W15]
bit4 ∈ [0 ... 15]

Действие: Для операции “.C”:
(Ws)<bit4> → C
1 → Ws<bit4>
Для операции “.Z” (по умолчанию):
$\overline{(Ws)<bit4>} \rightarrow Z$
1 → Ws<bit4>

Влияет на флаги: Z или C

Код:

1010	0100	bbbb	Z000	0ppp	ssss

Описание: Бит ‘bit4’ тестируется в регистре Ws. Если определена опция “.Z” инструкции, то дополнение тестируемого бита запоминается в флаг Z регистра STATUS. Если определена опция “.C” инструкции, то значение тестируемого бита запоминается в флаг C регистра STATUS. В обоих случаях тестируемый бит Ws устанавливается в '1'.

Биты ‘b’ выбирают значение bit4, положения тестируемого бита.
Бит ‘Z’ выбирает флаг C или Z как место назначения.
Биты ‘p’ выбирают режим адресации источника.
Биты ‘s’ выбирают регистр источник.

Примечание: Эта инструкция работает только в режиме слова. Если расширение не указано, то подразумевается операция “.Z”.

Слова: 1

Циклы: 1

Пример 1:

```
BTSTS.C     [[W0++], #0x3          ; Установить C = биту 3 в [W0]
                                   ; Установить бит 3 в [W0] = 1
                                   ; Пост-инкремент W0
```

	Перед инструкцией		После инструкции	
W0	1200		W0	1202
Данные 1200	FFF7		Данные 1200	FFFF
SR	0001	(C = 1)	SR	0000

Пример 2:

```
BTSTS.Z     W0, #0x7               ; Установить Z = дополнению от бита 7
                                   ; в W0, и установить бит 7 в W0 = 1
```

	Перед инструкцией		После инструкции	
W0	F234		W0	F2BC
SR	0000		SR	0002	(Z = 1)

CALL **Вызвать подпрограмму**

Синтаксис: {метка:} CALL Expr

Операнды: Expr может быть меткой или выражением (но не литералом).
Expr превращается компоновщиком в lit23, где lit23 ∈ [0 ... 8388606].

Действие:
(PC) + 4 → PC
(PC<15:0>) → (TOS)
(W15) + 2 → W15
(PC<23:16>) → (TOS)
(W15) + 2 → W15
lit23 → PC
NOP → Регистр инструкции

Влияет на флаги: Не влияет

Код:

1-е слово	0000	0010	nnnn	nnnn	nnnn	nnn0
2-е слово	0000	0000	0000	0000	0nnn	nnnn

Описание: Напрямую вызывает подпрограмму в течении целого 4-Mbyte диапазона программной памяти инструкций. Перед тем как сделать CALL, 24-битный адрес возврата (PC + 4) помещается в стек. После того как адрес возврата помещён в стек, 23-битное значение ‘lit23’ загружается в PC.

‘n’ бит формируют целевой адрес.

Примечание: Компоновщик решает определённое выражение на lit23 будет использовано.

Слова: 2

Циклы: 2

Пример 1:

```
026000                  CALL        _FIR        ; Вызвать подпр.  _FIR
026004                  MOV         W0, W1
  .                     . . .
  .                     . . .
  .                     . . .
026844      _FIR:       MOV         #0x400, W2  ; Старт подпр. _FIR
026846                  . . .
```

	Перед инструкцией		После инструкции
PC	02 6000	PC	02 6844
W15	A268	W15	A26C
Данные A268	FFFF	Данные A268	6004
Данные A26A	FFFF	Данные A26A	0002
SR	0000	SR	0000

Пример 2:

```
072000                  CALL        _G66        ; вызвать ПП _G66
072004                  MOV         W0, W1
  .                     . . .
077A28      G_66:       INC         W6, [W7++]  ; старт ПП
077A2A                  . . .
077A2C
```

	Перед инструкцией		После инструкции
PC	07 2000	PC	07 7A28
W15	9004	W15	9008
Данные 9004	FFFF	Данные 9004	2004
Данные 9006	FFFF	Данные 9006	0007
SR	0000	SR	0000

CALL **Косвенный вызов подпрограммы**

Синтаксис: {метка:} CALL Wn

Операнды: Wn ∈ [W0 ... W15]

Действие:
(PC) + 2 → PC
(PC<15:0>) → TOS
(W15) + 2 → W15
(PC<23:16>) → TOS
(W15) + 2 → W15
0 → PC<22:16>
(Wn<15:1>) → PC<15:1>
NOP → Регистр инструкции

Влияет на флаги: Не влияет

Код:

0000	0001	0000	0000	0000	ssss

Описание: Косвенный вызов подпрограммы в течении первых 32K инструкций программной памяти. Перед выполнением CALL, 24-битный адрес возврата (PC + 2) сохраняется в стеке. После сохранения в стеке адреса возврата, Wn<15:1> загружается в PC<15:1> и PC<22:16> очищаются. Поскольку PC<0> всегда ‘0’, Wn<0> игнорируется.

Биты ‘s’ выбирают регистр источник.

Слова: 1

Циклы: 2

Пример 1:

```
001002                  CALL        W0          ; Вызвать косвенно BOOT
001002                  . . .                   ; подпр., используя W0
  .                     . . .
001002      _BOOT:      MOV         #0x400, W2  ; _BOOT старт здесь
001002                  MOV         #0x300, W6
  .                     . . .
```

	Перед инструкцией		После инструкции
PC	00 1002	PC	00 1600
W0	1600	W0	1600
W15	6F00	W15	6F04
Данные 6F00	FFFF	Данные 6F00	1004
Данные 6F02	FFFF	Данные 6F02	0000
SR	0000	SR	0000

Пример 2:

```
004200                  CALL        W7          ; Вызвать косвенно TEST
004202                  . . .                   ; подпр., используя W7
  .                     . . .                   ; _TEST старт здесь
005500      _TEST:      INC         W1, W2      ;
005502                  DEC         W1, W3
  .
```

	Перед инструкцией		После инструкции
PC	00 4200	PC	00 5500
W7	5500	W7	5500
W15	6F00	W15	6F04
Данные 6F00	FFFF	Данные 6F00	4202
Данные 6F02	FFFF	Данные 6F02	0000
SR	0000	SR	0000

CLR **Очистить f или WREG**

Синтаксис: {метка:} CLR{.B} f
WREG

Операнды: f ∈ [0 ... 8191]

Действие: 0 → место назначения, определённое через D

Влияет на флаги: Не влияет

Код:

1110	1111	0BDf	ffff	ffff	ffff

Описание: Очистить содержимое файлового регистра или рабочего регистра по умолчанию WREG. Если определён WREG, WREG будет очищен. В противном случае будет очищен определённый файловый регистр ‘f’.

Бит ‘B’ выбирает операцию с байтом или словом ('0' для слова, '1' для байта.
Бит ‘D’ выбирает место назначения (‘0’ для WREG, ‘1’ для файлового регистра).
Биты 'f' выбирают адрес файлового регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.
2: WREG установлен на рабочий регистр W0.

Слова: 1

Циклы: 1

Пример 1:
```
CLR.B       RAM200      Очистить RAM200 (режим байта)
```

	Перед инструкцией	После инструкции
RAM200	8009	8000
SR	0000	0000

Пример 2:
```
CLR         WREG        ; Очистить WREG (режим слова)
```

	Перед инструкцией	После инструкции
WREG	0600	0000
SR	0000	0000

CLR **Очистить Wd**

Синтаксис: {метка:} CLR{.B} Wd
[Wd]
[Wd++]
[Wd--]
[++Wd]
[--Wd]

Операнды: Wd ∈ [W0 ... W15]

Действие: 0 → Wd

Влияет на флаги: Не влияет

Код:

1110	1011	0Bqq	qddd	d000	0000

Описание: Очистить содержимое регистра Wd. Любая, прямая или косвенная регистровая адресация может быть использована для Wd.

Бит ‘B’ выбирает операцию с байтом или словом ('0' для слова, '1' для байта.
Биты ‘d’ выбирают регистр места назначения.
Биты ‘q’ выбирают режим адресации места назначения.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

Слова: 1

Циклы: 1

Пример 1:
```
CLR.B       W2          ; Очистить W2 (Режим байта)
```

	Перед инструкцией		После инструкции
W2	3333	W2	3300
SR	0000	SR	0000

Пример 2:
```
CLR         [W0++]      ; Очистить [W0]
                        ; Пост-инкремент W0
```

	Перед инструкцией		После инструкции
W0	2300	W0	2302
Данные 2300	5607	Данные 2300	0000
SR	0000	SR	0000

CLR **Очистка аккумулятора, выборка операндов**

Синтаксис:	{метка:}	CLR	Acc	{,[Wx],Wxd} {,[Wx] + = kx,Wxd} {,[Wx] – = kx,Wxd} {,[W9 + W12],Wxd}	{,[Wy],Wyd} {,[Wy] + = ky,Wyd} {,[Wy] – = ky,Wyd} {,[W11 + W12],Wyd}	{,AWB}

Операнды: Acc ∈ [A,B]
Wx ∈ [W8, W9]; kx ∈ [-6, -4, -2, 2, 4, 6]; Wxd ∈ [W4 ... W7]
Wy ∈ [W10, W11]; ky ∈ [-6, -4, -2, 2, 4, 6]; Wyd ∈ [W4 ... W7]
AWB ∈ [W13, [W13] + = 2]

Действие: 0 → Acc(A or B)
([Wx]) → Wxd; (Wx) +/– kx → Wx
([Wy]) → Wyd; (Wy) +/– ky → Wy
(Acc(B or A)) rounded → AWB

Влияет на флаги: OA, OB, SA, SB

Код:

1100	0011	A0xx	yyii	iijj	jjaa

Описание: Очищает все 40 бит определённого аккумулятора, опциональная выборка операндов в подготовке для инструкций типа MAC и опциональное запоминание неопределённого результата аккумулятора. Эта инструкция очищает соответствующие флаги переполнения и насыщения (любые OA, SA или OB, SB).
Операнды Wx, Wxd, Wy и Wyd определяют опциональные операнды выборки, которые поддерживают косвенную и регистра смещения адресацию, как описано в части 4.14.1 “MAC упреждающая выборка”. Операнд AWB определяет опциональное прямое или косвенное регистровое запоминание конвергентно округлённого содержимого “другого” аккумулятора, как описано в секции 4.14.4 “Обратная запись MAC”.

Бит ‘A’ выбирает другой аккумулятор, выбираемый для обратной записи.
Биты ‘x’ выбирают место назначения упреждающей выборки Wxd.
Биты ‘y’ выбирают место назначения упреждающей выборки.
Биты ‘i’ выбирают операцию упреждающей выборки Wx.
Биты ‘j’ выбирают операцию упреждающей выборки Wy.
Биты ‘a’ выбирают место назначения обратной записи аккумулятора.

Слова: 1

Циклы: 1

Пример 1:

```
CLR       A, [W8]+=2, W4, W13      ; Очистить ACCA
                                   ; Загрузить W4 с [W8],пост-инкремент
                                   W8
                                   ; Запомнить ACCB в W13
```

	Перед инструкцией		После инструкции
W4	F001	W4	1221
W8	2000	W8	2002
W13	C623	W13	5420
ACCA	00 0067 2345	ACCA	00 0000 0000
ACCB	00 5420 3BDD	ACCB	00 5420 3BDD
Data 2000	1221	Data 2000	1221
SR	0000	SR	0000

Пример 2:

```
CLR   B,  [W8]+=2,W6,   [W10]+=2,  W7, [W13]+=2   ; Очистить ACCB
                                                  ; Загрузить W6 с [W8]
                                                  ; Загрузить W7 с [W10]
                                                  ; Сохранить ACCA в
                                                  [W13]
                                                  ; Пост- инкремент
                                                  ; W8,W10,W13
```

	Перед инструкцией
W6	F001
W7	C783
W8	2000
W10	3000
W13	4000
ACCA	00 0067 2345
ACCB	00 5420 ABDD
Data 2000	1221
Data 3000	FF80
Data 4000	FFC3
SR	0000

	После инструкции
W6	1221
W7	FF80
W8	2002
W10	3002
W13	4002
ACCA	00 0067 2345
ACCB	00 0000 0000
Data 2000	1221
Data 3000	FF80
Data 4000	0067
SR	0000

CLRWDT **Очистить сторожевой таймер**

Синтаксис: {метка:} CLRWDT

Операнды: Нет

Действие: 0 → WDT счётный регистр
0 → WDT предделитель счётчика A
0 → WDT предделитель счётчика B

Влияет на флаги: Не влияет

Код:

1111	1110	0110	0000	0000	0000

Описание: Очищает содержимое счётного регистра сторожевого таймера и предделителя счётных регистров. Установки предделителя A и предделителя B сторожевого таймера, установленные согласно конфигурации FWDT, не изменяются.

Слова: 1

Циклы: 1

Пример:

```
CLRWDT      ; Очистить сторожевой таймер
```

	Перед инструкцией	После инструкции
SR	0000	0000

COM **Дополнение f**

Синтаксис: {метка:} COM{.B} f {,WREG}

Операнды: f ∈ [0 ... 8191]

Действие: $(\overline{f}) \rightarrow место\,назначения\,,\,определяемое\,D$

Влияет на флаги: N, Z

Код:

1110	1110	1BDf	ffff	ffff	ffff

Описание: Вычисляет дополнение до 1 содержимого файлового регистра и помещает результат в регистр места назначения. Опциональный операнд WREG определяет регистр места назначения. Если определён WREG, результат запоминается в WREG. Если WREG не определён, результат запоминается в файловом регистре.

Бит ‘B’ выбирает операцию с байтом или словом ('0' для слова, '1' для байта.
Бит ‘D’ выбирает место назначения (‘0’ для WREG, ‘1’ для файлового регистра).
Биты ‘f’ выбирают адрес файлового регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.
2: WREG выбирает рабочий регистр W0

Слова: 1

Циклы: 1

Пример 1:

```
COM.b       RAM200      ; COM RAM200 (Режим байта)
```

	Перед инструкцией		После инструкции	
RAM200	80FF	RAM200	8000	
SR	0000	SR	0002	(Z)

Пример 2:

```
COM         RAM400,WREG ; COM RAM400 и запомнить в WREG (режим слова)
```

	Перед инструкцией		После инструкции	
WREG	1211	WREG	F7DC	
RAM400	0823	RAM400	0823	
SR	0000	SR	0008	(N = 1)

COM **Дополнение Ws**

Синтаксис:	{метка:}	COM{.B}	Ws, [Ws], [Ws++], [Ws--], [++Ws], [--Ws],	Wd [Wd] [Wd++] [Wd--] [++Wd] [--Wd]
Операнды:	Ws ∈ [W0 ... W15] Wd ∈ [W0 ... W15]			
Действие:	$(\overline{Ws}) \rightarrow Wd$			
Влияет на флаги:	N, Z			

Код:	1110	1010	1Bqq	qddd	dppp	ssss

Описание: Вычисляет дополнение до 1 содержимого регистра источника Ws и помещает результат в регистр места назначения Wd. Опциональный операнд WREG определяет регистр места назначения. Любая, прямая и косвенная регистровая адресация может быть использована для Ws и Wd.

Бит ‘B’ выбирает операцию с байтом или словом ('0' для слова, '1' для байта).
Биты ‘q’ выбирают режим адресации места назначения.
Биты ‘d’ выбирают регистр места назначения.
Биты ‘p’ выбирают режим адресации источника.
Биты ‘s’ выбирают регистр источник.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.
2: WREG выбирает рабочий регистр W0

Слова: 1

Циклы: 1

Пример 1:
```
COM.B       [W0++],     [W1++]      ; COM [W0] и запомнить в [W1]
                                    ; (Режим байта)
                                    ; Пост-инкремент W0, W1
```

	Перед инструкцией		После инструкции	
W0	2301	W0	2302	
W1	2400	W1	2401	
Данные 2300	5607	Данные 2300	5607	
Данные 2400	ABCD	Данные 2400	ABA9	
SR	0000	SR	0008	(N = 1)

Пример 2:
```
COM         W0, [W1++]  ; COM W0 и запомнить в [W1] (Режим слова)
                        ; Пост-инкремент W1
```

	Перед инструкцией		После инструкции
W0	D004	W0	D004
W1	1000	W1	1002
Данные 1000	ABA9	Данные 1000	2FFB
SR	0000	SR	0000

CP **Сравнить f с WREG, установить флаги состояния**

Синтаксис: {метка:} CP{.B} f

Операнды: f ∈ [0 ...8191]

Действие: (f) – (WREG)

Влияет на флаги: DC, N, OV, Z, C

Код:

1110	0011	0B0f	ffff	ffff	ffff

Описание: Вычислить (f) – (WREG) и обновить регистр STATUS. Эта инструкция эквивалентна инструкции SUBWF, но результат вычитания не запоминается.

Бит ‘B’ выбирает операцию с байтом или словом ('0' для слова, '1' для байта).
Биты 'f’ выбирают адрес файлового регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.
2: WREG выбирает рабочий регистр W0

Слова: 1

Циклы: 1

Пример 1:
```
CP.B        RAM400      ; Сравнить RAM400 с WREG (Режим байта)
```

	Перед инструкцией		После инструкции	
WREG	8823	WREG	8823	
RAM400	0823	RAM400	0823	
SR	0000	SR	0002	(Z = 1)

Пример 2:
```
CP          0x1200      ; Сравнить (0x1200) с WREG (Режим слова)
```

	Перед инструкцией		После инструкции	
WREG	2377	WREG	2377	
Данные 1200	2277	Данные 1200	2277	
SR	0000	SR	0008	(N = 1)

CP **Сравнить Wb с lit5, установить флаги состояния**

Синтаксис: {метка:} CP{.B} Wb, #lit5

Операнды: Wb ∈ [W0 ... W15]
lit5 ∈ [0 ... 31]

Действие: (Wb) – lit5

Влияет на флаги: DC, N, OV, Z, C

Код:

1110	0001	0www	wB00	011k	kkkk

Описание: Вычислить (Wb) – lit5, и обновить регистр STATUS. Эта инструкция эквивалентна инструкции SUB, но результат вычитания не запоминается. Регистровая прямая адресация может быть использована для Wb.

Биты ‘w’ выбирают адрес базового регистра Wb.
Бит ‘B’ выбирает операцию с байтом или словом ('0' для слова, '1' для байта).
Биты ‘k’ обеспечивают литеральный операнд, 5-битное целое число.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

Слова: 1

Циклы: 1

Пример 1:

```
CP.B        W4, #0x12     ; Сравнить W4 с 0x12 (Режим байта)
```

	Перед инструкцией		После инструкции	
W4	7711	W4	7711	
SR	0000	SR	0008	(N = 1)

Пример 2:

```
CP          W4, #0x12     ; Сравнить W4 с 0x12 (Режим слова)
```

	Перед инструкцией		После инструкции
W4	7713	W4	7713
SR	0000	SR	0000

CP **Сравнить Wb с Ws, установить флаги состояния**

Синтаксис: {метка:} CP{.B} Wb, Ws
[Ws]
[Ws++]
[Ws--]
[++Ws]
[--Ws]

Операнды: Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]

Действие: (Wb) – (Ws)

Влияет на флаги: DC, N, OV, Z, C

Код:

1110	0001	0www	wB00	0ppp	ssss

Описание: Вычислить (Wb) – (Ws), и обновить регистр STATUS. Эта инструкция эквивалентна инструкции SUB, но
результат вычитания не запоминается. Прямая регистровая адресация может быть использована для Wb.
Прямая и косвенная регистровая адресация может быть использована для Ws.

Биты ‘w’ выбирают адрес базового регистра Wb.
Бит ‘B’ выбирает операцию с байтом или словом ('0' для слова, '1' для байта).
Биты ‘p’ выбирают режим адресации источника.
Биты ‘s’ выбирают адрес регистра источника.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

Слова: 1

Циклы: 1

Пример 1:
```
CP.B        W0, [W1++]   ; Сравнить [W1] с W0 (Режим байта)
                         ; Пост-инкремент W1
```

	Перед инструкцией		После инструкции	
W0	ABA9	W0	ABA9	
W1	2000	W1	2001	
Данные 2000	D004	Данные 2000	D004	
SR	0000	SR	0008	(N = 1)

Пример 2:
```
CP          W5, W6       ; Сравнить W6 с W5 (Режим слова)
```

	Перед инструкцией		После инструкции	
W5	2334	W5	2334	
W6	8001	W6	8001	
SR	0000	SR	000C	(N, OV = 1)

CP0 **Сравнить f с 0x0, установить флаги состояния**

Синтаксис: {метка:} CP0{.B} f

Операнды: f ∈ [0 ... 8191]

Действие: (f) – 0x0

Влияет на флаги: DC, N, OV, Z, C

Код:

1110	0010	0B0f	ffff	ffff	ffff

Описание: Вычислить (f) – 0x0 и обновить регистр STATUS. Результат вычитания не запоминается.

Бит ‘B’ выбирает операцию с байтом или словом ('0' для слова, '1' для байта).
Биты 'f’ выбирают адрес файлового регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

Слова: 1

Циклы: 1

Пример 1:

```
CP0.B       RAM100      ; Сравнить RAM100 с 0x0 (Режим байта)
```

	Перед инструкцией		После инструкции	
RAM100	44C3	RAM100	44C3	
SR	0000	SR	0008	(N = 1)

Пример 2:

```
CP          0x1FFE      ; Сравнить (0x1FFE) с 0x0 (Режим слова)
```

	Перед инструкцией		После инструкции	
Данные 1FFE	0001	Данные 1FFE	0001	
SR	0000	SR	0000	

CP0 **Сравнить Ws с 0x0, установить флаги состояния**

Синтаксис: {метка:} CP0{.B} Ws
[Ws]
[Ws++]
[Ws--]
[++Ws]
[--Ws]

Операнды: Ws ∈ [W0 ... W15]

Действие: (Ws) – 0x0000

Влияет на флаги: DC, N, OV, Z, C

Код:

1110	0000	0000	0B00	0ppp	ssss

Описание: Вычислить (Ws) – 0x0000 и обновить регистр STATUS. Результат вычитания не запоминается. Прямая и косвенная регистровая адресация может быть использована для Ws.

Бит ‘B’ выбирает операцию с байтом или словом ('0' для слова, '1' для байта).
Биты ‘p’ выбирают режим адресации источника.
Биты ‘s’ выбирают адрес регистра источника.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

Слова: 1

Циклы: 1

Пример 1:

```
CP0.B       [W4--]      ; Сравнить [W4] с 0 (Режим байта)
                        ; Пост-декремент
```

	Перед инструкцией		После инструкции	
W4	1001	W4	1000	
Данные 1000	0034	Данные 1000	0034	
SR	0000	SR	0002	(Z = 1)

Пример 2:

```
CP0         [--W5]      ; Сравнить [--W5] с 0 (Режим слова)
```

	Перед инструкцией		После инструкции	
W5	2400	W5	23FE	
Данные 1FFE	9000	Данные 1FFE	9000	
SR	0000	SR	0008	(N = 1)

CPB **Сравнить f с WREG используя заём, установить флаги состояния**

Синтаксис: {метка:} CPB{.B} f

Операнды: f ∈ [0 ...8191]

Действие: $(f)-(WREG)-\overline{(C)}$

Влияет на флаги: DC, N, OV, Z, C

Код:

1110	0011	1B0f	ffff	ffff	ffff

Описание: Вычислить $(f)-(WREG)-\overline{(C)}$, и обновить регистр STATUS. Эта инструкция эквивалентна инструкции SUBB, но результат вычитания не запоминается.

Бит ‘B’ выбирает операцию с байтом или словом ('0' для слова, '1' для байта).
Биты 'f' выбирают адрес файлового регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

2: WREG установлен на рабочий регистр W0.

3: Флаг Z является “прилипшим” для ADDC, CPB, SUBB and SUBBR. Эти инструкции могут только очищать Z.

Слова: 1

Циклы: 1

Пример 1:

```
CPB.B       RAM400      ; Сравнить RAM400 с WREG, используя C
                        ;(Режим байта)
```

	Перед инструкцией		После инструкции	
WREG	8823	WREG	8823	
RAM400	0823	RAM400	0823	
SR	0000	SR	0008	(N = 1)

Пример 2:

```
CPB         0x1200      ; Сравнить (0x1200) с WREG, используя C
                        ; (Режим слова)
```

	Перед инструкцией		После инструкции	
WREG	2377	WREG	2377	
Данные 1200	2377	Данные 1200	2377	
SR	0001	SR	0001	(C = 1)
	(C = 1)			

CPB **Сравнить Wb с lit5 используя заём, установить флаги состояния**

Синтаксис: {метка:} CPB{.B} Wb, #lit5

Операнды: Wb ∈ [W0 ... W15]
lit5 ∈ [0 ... 31]

Действие: $(Wb)-(lit5)-\overline{(C)}$

Влияет на флаги: DC, N, OV, Z, C

Код:

1110	0001	1www	wB00	011k	kkkk

Описание: Вычислить , $(Wb)-(lit5)-\overline{(C)}$ и обновить регистр STATUS. Эта инструкция эквивалентна инструкции SUBB, но результат вычитания не запоминается. Прямая регистровая адресация может использоваться для Wb.

Биты ‘w’ выбирают адрес базового регистра Wb.
Бит ‘B’ выбирает операцию с байтом или словом ('0' для слова, '1' для байта).
Биты ‘k’ обеспечивают литеральный операнд, пятибитное целое число.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.
2: Флаг Z является “прилипшим” для ADDC, CPB, SUBB and SUBBR. Эти инструкции могут только очищать Z.

Слова: 1

Циклы: 1

Пример 1:
```
CPB.B    W4, #0x12    ; Сравнить W4 с 0x12, используя C (Режим
                      ; байта)
```

	Перед инструкцией		После инструкции	
W4	7711		7711	
SR	0001	(C = 1)	0008	(N = 1)

Пример 2:
```
CPB.B    W4, #0x12    ; Сравнить W4 с 0x12, используя C (Режим
                      ; байта)
```

	Перед инструкцией		После инструкции	
W4	7711		2377	
SR	0000		0001	(C = 1)

Пример 3:
```
CPB      W12, #0x1F   ;Сравнить W12 с 0x1F, используя C (Режим
                      ;слова)
```

	Перед инструкцией		После инструкции	
W12	0020		0020	
SR	0002	(Z = 1)	0003	(Z,C = 1)

Пример 4:
```
CPB      W12, #0x1F   ;Сравнить W12 с 0x1F, используя C (Режим
                      ;слова)
```

	Перед инструкцией		После инструкции	
W12	0020		0020	
SR	0003	(Z,C = 1)	0001	(C = 1)

CPB **Сравнить Ws с Wb используя заём, установить флаги состояния**

Синтаксис: {метка:} CPB{.B} Wb, Ws
[Ws]
[Ws++]
[Ws--]
[++Ws]
[--Ws]

Операнды: Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]

Действие: $(Wb)-(Ws)-\overline{(C)}$

Влияет на флаги: DC, N, OV, Z, C

Код:

1110	0001	1www	wB00	0ppp	ssss

Описание: Вычислить $(Wb)-(Ws)-\overline{(C)}$, и обновить регистр STATUS. Эта инструкция эквивалентна инструкции SUBB, но результат вычитания не запоминается. Прямая регистровая адресация может быть использована для Wb. Прямая или косвенная регистровая адресация может быть использована для Ws.

Биты ‘w’ выбирают адрес базового регистра Wb.
Бит ‘B’ выбирает операцию с байтом или словом ('0' для слова, '1' для байта).
Биты ‘p’ выбирают режим адресации источника.
Биты ‘s’ выбирают адрес регистра источника.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.
2: Флаг Z является “прилипшим” для ADDC, CPB, SUBB and SUBBR. Эти инструкции могут только очищать Z.

Слова: 1

Циклы: 1

Пример 1:
```
CPB.B    W0, [W1++]  ; Сравнить [W1] с W0, используя C
                     ;(Режим байта) ; Пост-инкремент
```

	Перед инструкцией		После инструкции	
W0	ABA9	W0	ABA9	
W1	1000	W1	1001	
Данные 1000	D0A9	Данные 1000	D0A9	
SR	0002 (Z = 1)	SR	0008	(N = 1)

Пример 2:
```
CPB.B    W0, [W1++]  ; Сравнить [W1] с W0, используя C
                     ;(Режим байта) ; Пост-инкремент
```

	Перед инструкцией		После инструкции	
W0	ABA9	W0	ABA9	
W1	1000	W1	1001	
Данные 1000	D0A9	Данные 1000	D0A9	
SR	0001 (C = 1)	SR	0001	(C = 1)

Пример 3:
```
CPB      W4, W5      ; Сравнить W5 с W4 используя C (Режим байта)
```

	Перед инструкцией		После инструкции	
W4	4000	W4	4000	
W5	3000	W5	3000	
SR	0001 (C = 1)	SR	0001	(C = 1)

CPSEQ **Сравнить Wb с Wn, пропустить, если равно (Wb = Wn)**

Синтаксис: {метка:} CPSEQ{.B} Wb, Wn

Операнды: Wb ∈ [W0 ... W15]
Wn ∈ [W0 ... W15]

Действие: (Wb) – (Wn)
Пропустить, если (Wb) = (Wn)

Влияет на флаги: Не влияет

Код:

1110	0111	1www	wB00	0000	ssss

Описание: Сравнить содержимое Wb с содержимым Wn, выполняя вычитание (Wb) – (Wn), но не запоминая результат. Если (Wb) = (Wn), следующая инструкция (выбранная в течении выполнения текущей инструкции) будет отвергнута и взамен её в следующем цикле будет выполнена инструкция NOP. Если (Wb) ≠ (Wn), выполняется следующая инструкция.

Биты ‘w’ выбирают адрес базового регистра Wb.
Бит ‘B’ выбирает операцию с байтом или словом ('0' для слова, '1' для байта).
Биты ‘s’ выбирают адрес регистра источника.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

Слова: 1

Циклы: 1 (2 или 3, если получен пропуск)

Пример 1:

```
002000      HERE:       CPSEQ.B     W0, W1      ; Если W0 = W1 (Режим
002002                  GOTO        BYPASS      ; байта), пропустить
002004                  . . .                   ; GOTO
002006                  . . .
002008      BYPASS:     . . .
00200A                  . . .
```

	Перед инструкцией		После инструкции
PC	00 2000	PC	00 2002
W0	1001	W0	1001
W1	1000	W1	1000
SR	0000	SR	0000

Пример 2:

```
018000      HERE:       CPSEQ       W4, W8      ; Если W4 = W8 (Режим
018002                  CALL        _FIR        ; слова), пропустить
018004                  . . .                   ; вызов подпрограммы
018006                  . . .
```

	Перед инструкцией		После инструкции	
PC	01 8000		PC	01 8006
W4	3344		W4	3344
W8	3344		W8	3344
SR	0002	(Z = 1)	SR	0002 (Z = 1)

CPSGT **Сравнить Wb с Wn, пропустить, если больше (Wb > Wn)**

Синтаксис: {метка:} CPSGT{.B} Wb, Wn

Операнды: Wb ∈ [W0 ... W15]
Wn ∈ [W0 ... W15]

Действие: (Wb) – (Wn)
Пропустить, если (Wb) > (Wn)

Влияет на флаги: Не влияет

Код:

1110	0110	0www	wB00	0000	ssss

Описание: Сравнить содержимое Wb с содержимым Wn, выполняя вычитание (Wb) – (Wn), но не запоминая результат. Если (Wb) > (Wn), следующая инструкция (выбранная в течении выполнения текущей инструкции) будет отвергнута и взамен её в следующем цикле будет выполнена инструкция NOP. В противном случае выполняется следующая инструкция.

Биты ‘w’ выбирают адрес базового регистра Wb.
Бит ‘B’ выбирает операцию с байтом или словом ('0' для слова, '1' для байта).
Биты ‘s’ выбирают адрес регистра источника.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

Слова: 1

Циклы: 1 (2 или 3, если получен пропуск)

Пример 1:

```
002000      HERE:       CPSGT.B     W0, W1      ; Если W0 > W1 (Режим
002002                  GOTO        BYPASS      ; байта), пропустить
002006                  . . .                   ; GOTO
002008                  . . .
00200A      BYPASS:     . . .
00200C                  . . .
```

	Перед инструкцией		После инструкции	
PC	00 2000		PC 00 2006	
W0	00FF		W0 00FF	
W1	26FE		W1 26FE	
SR	0009	(N, C = 1)	SR 0009	(N, C = 1)

Пример 2:

```
018000      HERE:       CPSGT       W4, W5      ; Если W4 > W5 (Режим
018002                  CALL        _FIR        ; слова), пропустить
018006                  . . .                   ; вызов подпрограммы
018008                  . . .
```

	Перед инструкцией		После инструкции	
PC	01 8000		PC 01 8002	
W4	2600		W4 2600	
W5	2600		W5 2600	
SR	0004	(OV = 1)	SR 0004	(OV = 1)

CPSLT **Сравнить Wb с Wn, пропустить, если меньше (Wb < Wn)**

Синтаксис: {метка:} CPSLT{.B} Wb, Wn

Операнды: Wb ∈ [W0 ... W15]
Wn ∈ [W0 ... W15]

Действие: (Wb) – (Wn)
Пропустить, если (Wb) < (Wn)

Влияет на флаги: Не влияет

Код:

1110	0110	1www	wB00	0000	ssss

Описание: Сравнить содержимое Wb с содержимым Wn, выполняя вычитание (Wb) – (Wn), но не запоминая результат. Если (Wb) < (Wn), следующая инструкция (выбранная в течении выполнения текущей инструкции) будет отвергнута и взамен её в следующем цикле будет выполнена инструкция NOP. В противном случае выполняется следующая инструкция.

Биты ‘w’ выбирают адрес базового регистра Wb.
Бит ‘B’ выбирает операцию с байтом или словом ('0' для слова, '1' для байта).
Биты ‘s’ выбирают адрес регистра источника.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

Слова: 1

Циклы: 1 (2 или 3, если получен пропуск)

Пример 1:

```
002000      HERE:       CPSLT.B     W8, W9      ; Если W8 < W9 (Режим
002002                  GOTO        BYPASS      ; байта), пропустить
002006                  . . .                   ; GOTO
002008                  . . .
00200A      BYPASS:     . . .
00200C                  . . .
```

	Перед инструкцией			После инструкции	
PC	00 2000		PC	00 2002	
W8	00FF		W8	00FF	
W9	26FE		W9	26FE	
SR	0008	(N = 1)	SR	0008	(N = 1)

Пример 2:

```
018000      HERE:       CPSLT       W3, W6      ; Если W3 < W6 (Режим
018002                  CALL        _FIR        ; слова), пропустить
018006                  . . .                   ; вызов подпрограммы
018008                  . . .
```

	Перед инструкцией		После инструкции	
PC	01 8000	PC	01 8006	
W3	2600	W3	2600	
W6	3000	W6	3000	
SR	0000		0000	

CPSNE **Сравнить Wb с Wn, пропустить, если не равно (Wb ≠ Wn)**

Синтаксис: {метка:} CPSLT{.B} Wb, Wn

Операнды: Wb ∈ [W0 ... W15]
Wn ∈ [W0 ... W15]

Действие: (Wb) – (Wn)
Пропустить, если (Wb) ≠ (Wn)

Влияет на флаги: Не влияет

Код:

1110	0111	0www	wB00	0000	ssss

Описание: Сравнить содержимое Wb с содержимым Wn, выполняя вычитание (Wb) – (Wn), но не запоминая результат. Если (Wb) ≠ (Wn), следующая инструкция (выбранная в течении выполнения текущей инструкции) будет отвергнута и взамен её в следующем цикле будет выполнена инструкция NOP. В противном случае выполняется следующая инструкция.

Биты ‘w’ выбирают адрес базового регистра Wb.
Бит ‘B’ выбирает операцию с байтом или словом ('0' для слова, '1' для байта).
Биты ‘s’ выбирают адрес регистра источника.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

Слова: 1

Циклы: 1 (2 или 3, если получен пропуск)

Пример 1:

```
002000      HERE:       CPSNE.B     W2, W3      ; Если W2 != W3 (Режим
002002                  GOTO        BYPASS      ; байта), пропустить
002006                  . . .                   ; GOTO
002008                  . . .
00200A      BYPASS:     . . .
00200C                  . . .
```

	Перед инструкцией			После инструкции	
PC	00 2000		PC	00 2006	
W2	00FF		W2	00FF	
W3	26FE		W3	26FE	
SR	0001	(C = 1)	SR	0001	(C = 1)

Пример 2:

```
018000      HERE:       CPSNE       W0, W8      ; Если W0 != W8 (Режим
018002                  CALL        _FIR        ; слова), пропустить
018006                  . . .                   ; вызов подпрограммы
018008                  . . .
```

	Перед инструкцией		После инструкции	
PC	01 8000		PC	01 8002
W0	3000		W0	3000
W8	3000		W8	3000
SR	0000			0000

DAW.B **Десятичная коррекция Wn**

Синтаксис: {метка:} DAW.B Wn

Операнды: Wn ∈ [W0 ... W15]

Действие:
Если (Wn<3:0> > 9) или (DC = 1)
(Wn<3:0>) + 6 → Wn<3:0>
Иначе
(Wn<3:0>) → Wn<3:0>
Если (Wn<7:4> > 9) или (C = 1)
(Wn<7:4>) + 6 → Wn<7:4>
Иначе
(Wn<7:4>) → Wn<7:4>

Влияет на флаги: C

Код:

1111	1101	0100	0000	0000	ssss

Описание: Коррекция наименьшего значимого байта в Wn производящая результат в двоично десятичном коде (BCD). Наиболее значимый байт Wn не изменяется, и флаг переноса используется для индикации любого десятичного одновременно. Прямая регистровая адресация может быть использована для Wn.

Биты ‘s’ выбирают адрес регистра источника.

Примечание 1: Эта инструкция используется для корреции формата данных после тотго как два упакованных байта были сложены.
2: Эта инструкция работает только в режиме байта и расширение .B должно быть включено в код операции.

Слова: 1

Циклы: 1

Пример 1:

```
DAW.B    W0          ; Десятичная коррекция W0
```

	Перед инструкцией			После инструкции	
W0	771A		W0	7720	
SR	0002	(DC = 1)	SR	0002	(DC = 1)

Пример 2:

```
DAW.B    W3          ; Десятичная коррекция W3
```

	Перед инструкцией		После инструкции	
W3	77AA	W3	7710	
SR	0000	SR	0001	(C = 1)

DEC **Декремент f**

Синтаксис: {метка:} DEC{.B} f {,WREG}

Операнды: f ∈ [0 ... 8191]

Действие: (f) – 1 → место назначение, определённое D

Влияет на флаги: DC, N, OV, Z, C

Код:

1110	1101	0BDf	ffff	ffff	ffff

Описание: Вычесть один из содержимого файлового регистра и поместить результат в регистр места назначения. Необязательный операнд WREG определяет регистр места назначения. Если определён WREG, результат запоминается в WREG. Если WREG не определён, результат запоминается в файловом регистре.

Бит ‘B’ выбирает операцию с байтом или словом ('0' для слова, '1' для байта).
Бит ‘D’ выбирает место назначения (‘0’ для WREG, ‘1’ для файлового регистра).
Биты 'f' выбирают адрес файлового регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.
2: В качестве WREG используется рабочий регистр W0.

Слова: 1

Циклы: 1

Пример 1:

```
DEC.B       0x200       ; Декремент (0x200) (Режим байта)
```

	Перед инструкцией		После инструкции	
Данные 200	80FF	Данные 200	80FE	
SR	0000	SR	0009	((N, C = 1)

Пример 2:

```
DEC         RAM400, WREG        ; Декремент RAM400 и запомнить
                                ; в WREG (Режим слова)
```

	Перед инструкцией		После инструкции
WREG	1211	WREG	0822
RAM400	0823	RAM400	0823
SR	0000	SR	0000

DEC **Декремент Ws**

Синтаксис: {метка:} DEC{.B} Ws, [Ws], [Ws++], [Ws--], [++Ws], [--Ws] Wd [Wd] [Wd++] [Wd--] [++Wd] [--Wd]

Операнды: Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

Действие: (Ws) – 1 → Wd

Влияет на флаги: DC, N, OV, Z, C

Код:

1110	1001	0Bqq	qddd	dppp	ssss

Описание: Вычесть один из содержимого регистра источника Ws и поместить результат в регистр назначения Wd. Любая, прямая или косвенная регистровая адресация может быть использована для Ws и Wd.

Бит ‘B’ выбирает операцию с байтом или словом ('0' для слова, '1' для байта).
Биты ‘q’ выбирают режим адресации места назначения.
Биты ‘d’ выбирают регистр места назначения.
Биты ‘p’ выбирают режим адресации источника.
Биты ‘s’ выбирают адрес регистра источника.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

Слова: 1

Циклы: 1

Пример 1:

```
DEC.B       [W7++], [W8++]          ; DEC [W7] и запомнить в [W8]
                                    ; (Режим байта)
                                    ; Пост-инкремент W7, W8
```

	Перед инструкцией		После инструкции
W7	2301	W7	2302
W8	2400	W8	2401
Данные 2300	5607	Данные 2300	5607
Данные 2400	ABCD	Данные 2400	AB55
SR	0000	SR	0000

Пример 2:

```
DEC         W5, [W6++]  ; Декремент W5 и запомнить в [W6] (Режим слова)
                        ; Пост-инкремент W6
```

	Перед инструкцией		После инструкции	
W5	D004	W5	D004	
W6	2000	W6	2002	
Данные 2000	ABA9	Данные 2000	D003	
SR	0000	SR	0009	(N, C = 1)

DEC2 **Декремент f на 2**

Синтаксис: {метка:} DEC2{.B} f {,WREG}

Операнды: f ∈ [0 ... 8191]

Действие: (f) – 2 → место назначения, определённое D

Влияет на флаги: DC, N, OV, Z, C

Код:

1110	1101	1BDf	ffff	ffff	ffff

Описание: Вычесть два из содержимого файлового регистра и поместить результат в регистр назначения. Необязательный операнд WREG определяет регистр места назначения. Если определён WREG, результат запоминается в WREG. Если WREG не определён, результат запоминается в файловом регистре.

Бит ‘B’ выбирает операцию с байтом или словом ('0' для слова, '1' для байта).
Бит ‘D’ выбирает место назначения (‘0’ для WREG, ‘1’ для файлового регистра).
Биты 'f' выбирают адрес файлового регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

Слова: 1

Циклы: 1

Пример 1:

```
DEC2.B      0x200       ; Декремент (0x200) на 2 (Режим байта)
```

	Перед инструкцией		После инструкции	
Данные 200	80FF	Данные 200	80FD	
SR	0000	SR	0009	(N, C = 1)

Пример 2:

```
DEC2        RAM400, WREG        ; Декремент RAM400 на 2 и
                                ; запомнить в WREG (Режим слова)
```

	Перед инструкцией		После инструкции	
WREG	1211	WREG	0821	
RAM 400	0823	RAM 400	0823	
SR	0000	SR	0000	

DEC2 **Декремент Ws на 2**

Синтаксис: {метка:} DEC2{.B} Ws, [Ws], [Ws++], [Ws--], [++Ws], [--Ws] Wd [Wd] [Wd++] [Wd--] [++Wd] [--Wd]

Операнды: Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

Действие: (Ws) – 2 → Wd

Влияет на флаги: DC, N, OV, Z, C

Код:

1110	1001	1Bqq	qddd	dppp	ssss

Описание: Вычесть два из содержимого регистра источника Ws и поместить результат в регистр назначения Wd. Любая, прямая или косвенная регистровая адресация может быть использована для Ws и Wd.

Бит ‘B’ выбирает операцию с байтом или словом ('0' для слова, '1' для байта).
Биты ‘q’ выбирают режим адресации места назначения.
Биты ‘d’ выбирают регистр места назначения.
Биты ‘p’ выбирают режим адресации источника.
Биты ‘s’ выбирают адрес регистра источника.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

Слова: 1

Циклы: 1

Пример 1:

```
DEC2.B      [W7--], [W8--]          ; DEC [W7] на 2, запомнить в [W8]
                                    ; (Режим байта)
                                    ; Пост-декремент W7, W8
```

	Перед инструкцией		После инструкции	
W7	2301	W7	2300	
W8	2400	W8	23FF	
Данные 2300	0107	Данные 2300	0107	
Данные 2400	ABCD	Данные 2400	ABFF	
SR	0000	SR	0008	(N = 1)

Пример 2:

```
DEC2        W5, [W6++]  ; DEC W5 на 2, запомнить в [W6] (Режим слова)
                        ; Пост-инкремент W6
```

	Перед инструкцией		После инструкции	
W5	D004	W5	D004	
W6	1000	W6	1002	
Данные 1000	ABA9	Данные 1000	D002	
SR	0000	SR	0009	(N, C = 1)

DISI **Временно блокировать прерывания**

Синтаксис: {метка:} DISI #lit14

Операнды: lit14 ∈ [0 ... 16383]

Действие: lit14 → DISICNT
1 → DISI
Блокировать прерывания для (lit14 + 1) циклов

Влияет на флаги: Не влияет

Код:

1111	1100	00kk	kkkk	kkkk	kkkk

Описание: Блокировать прерывания с приоритетом от 0 до 6 для (lit14 + 1) циклов инструкции. Прерывания с приоритетом от 0 до 6 будут блокированы начиная с цикла в котором выполнена инструкция DISI, и остаются блокированными следующие (lit 14) циклов. Значение lit14 записано в регистр DISICNT, и флаг DISI (INTCON2<14>) установлен в ‘1’. Это инструкция может быть использована перед выполнением критичного к времени кода, для ограничения эффекта прерываний.

Примечание: Эта инструкция не мешает выполнению прерываний с приоритетом 7 и ловушек. Смотреть “Руководство по семейству dsPIC30F” (DS70046) для деталей.

Слова: 1

Циклы: 1

Пример 1:

```
002000      HERE:  DISI  #100        ;Блокировать прерывания для 101 цикла
002002                               ;Следующие 100 циклов защищены DISI
002004             . . .
```

	Перед инструкцией		После инструкции	
PC	00 2000	PC	00 2002	
DISICNT	0000	W2	0100	
INTCON2	0000	W3	4000	(DISI = 1)
SR	0000	SR	0000	

DIV.S **Знаковое целочисленное деление**

Синтаксис: {метка:} DIV.S{W} Wm, Wn
DIV.SD Wm, Wn

Операнды: Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]Wm ∈ [W0 ... W15] для операции со словом
Wm ∈ [W0, W2, W4 ... W14] для операции с двойным словом
Wn ∈ [W2 ... W15]

Действие: Для операции со словом (по умолчанию):
Wm → W0
Если (Wm<15> = 1):
0xFFFF → W1
Иначе:
0x0 → W1
W1:W0 / Wn → W0
Остаток → W1
Для операции с двойным словом (DIV.SD):
Wm + 1:Wm → W1:W0
W1:W0 / Wn → W0
Остаток → W1

Влияет на флаги: DC, N, OV, Z, C

Код:

1101	1000	0ttt	tvvv	vW00	ssss

Описание: Итеративное, знаковое целочисленное деление, где делимое хранится в Wm (для деления 16-бит на 16-бит) или Wm + 1:Wm (для деления 32-бит на 16-бит) и делитель хранится в Wn. В заданной по умолчанию операции слова, Wm сначала копируется в W0 и расширяется знаком через W1, чтобы выполнить операцию. В двойной операции, Wm + 1:Wm сначала копируется в W1:W0. 16-битное частное операции деления сохраняется в W0, и 16-битный остаток сохраняется W1.
Эта инструкция выполняется 18 раз, используя инструкцию REPEAT (с счётчиком итераций 17) для генерации корректного частного и остатка. Флаг N будет установлен, если остаток отрицательный и очищен иначе. Флаг OV будет установлен, если результат операции деления переполнен и очищен иначе. Флаг Z будет установлен, если остаток равен ‘0’ и очищен иначе. Флаг C используется для обеспечения алгоритма деления и его конечное значение не надо использовать.

Биты ‘t’ выбирают наиболее значимое слово делимого для двойной операции. Эти биты очищены для операции со словом.
Биты ‘v’ выбирают наименее значимое слово делимого.
Бит ‘W’ выбирает размер делимого (‘0’ для 16-бит, ‘1’ для 32-бит).
Биты ‘s’ выбирают регистр делителя.

Примечание 1: Расширение .D в обозначении инструкции двойное слово (32-bit) делимое скорее чем делимое в слово. Вы можете использовать .W расширение для обозначения операции со словом, но это не обязательно.

2: Неожиданный результат будет происходить если частное может не быть представлено в 16 битах. Когда это происходит для двойной операции (DIV.SD), бит состояния OV будет установлен и частное и остаток не следует использовать. Для операции со словом (DIV.S), только один тип переполнения может происходить (0x8000/0xFFFF = + 32768 или 0x00008000), которое позволяет биту состояния OV интерпретировать результат.

3: Деление на ноль может инициировать ловушку арифметической ошибки в течении первого цикла выполнения.

4: Эта инструкция может прерываться в каждой границе инструкции цикла.

Слова: 1

Циклы: 18 (плюс 1 для выполнения REPEAT)

Пример 1:

```
REPEAT      #17         ; Выполнить DIV.S 18 раз
DIV.S       W3, W4      ; Делить W3 на W4
                        ; Запомнить частное в W0, остаток в W1
```

	Перед инструкцией	После инструкции
W0	5555	013B
W1	1234	0003
W3	3000	3000
W4	0027	0027
SR	0000	0000

Пример 2:

```
REPEAT      #17         ; Выполнить DIV.SD 18 раз
DIV.SD      W0, W12     ; Делить W1:W0 на W12
                        ; Запомнить частное в W0, остаток в W1
```

	Перед инструкцией	После инструкции	
W0	2500	FA6B	
W1	FF42	EF00	
w12	2200	2200	
SR	0000	0008	(N = 1)

DIV.U **Беззнаковое целочисленное деление**

Синтаксис: {метка:} DIV.U{W} Wm, Wn
DIV.UD Wm, Wn

Операнды: Wm ∈ [W0 ... W15] для операции со словом
Wm ∈ [W0, W2, W4 ... W14] для операции с двойным словом
Wn ∈ [W2 ... W15]

Действие: Для операции со словом (по умолчанию):
Wm → W0
0x0 → W1
W1:W0/Wn → W0
Остаток → W1
Для операции с двойным словом (DIV.UD):
Wm + 1:Wm → W1:W0
W1:W0/Wns → W0
Остаток → W1

Влияет на флаги: N, OV, Z, C

Код:

1101	1000	1ttt	tvvv	vW00	ssss

Описание: Итеративное, беззнаковое целочисленное деление, где делимое хранится в Wm (для деления 16-бит на 16-бит) или Wm + 1:Wm (для деления 32-бит на 16-бит) и делитель хранится в Wn. В операции со словом, Wm сначала копируется в W0 и расширяется знаком через W1, чтобы выполнить операцию. В операции с двойным словом, Wm + 1:Wm сначала копируется в W1:W0. 16-битное частное операции деления сохраняется в W0, и 16-битный остаток сохраняется W1.
Эта инструкция выполняется 18 раз, используя инструкцию REPEAT (с счётчиком итераций 17) для генерации корректного частного и остатка. Флаг N будет установлен, если остаток отрицательный и очищен иначе. Флаг OV будет установлен, если результат операции деления переполнен и очищен иначе. Флаг Z будет установлен, если остаток равен ‘0’ и очищен иначе. Флаг C используется для обеспечения алгоритма деления и его конечное значение не надо использовать.

Биты ‘t’ выбирают наиболее значимое слово делимого для двойной операции. Эти биты очищены для операции со словом.
Биты ‘v’ выбирают наименее значимое слово делимого.
Бит ‘W’ выбирает размер делимого (‘0’ для 16-бит, ‘1’ для 32-бит).
Биты ‘s’ выбирают регистр делителя.

Примечание 1: Расширение .D в обозначении инструкции двойное слово (32-bit) делимое скорее чем делимое в слово. Вы можете использовать .W расширение для обозначения операции со словом, но это не обязательно.
2: Неожиданный результат будет происходить если частное может не быть представлено в 16 битах. Это может произойти только для операции с двойным словом (DIV.UD). Когда произойдёт переполнение, бит состояния OV будет установлен и частное и остаток не должны использоваться.
3: Деление на ноль может инициировать ловушку арифметической ошибки в течении первого цикла выполнения.
4: Эта инструкция может прерываться в каждой границе инструкции цикла.

Слова: 1

Циклы: 18 (плюс 1 для выполнения REPEAT)

Пример 1:

```
REPEAT      #17         ; Выполнить DIV.U 18 раз
DIV.U       W2, W4      ; Делить W2 на W4
                        ; Запомнить частное в W0, остаток в W1
```

	Перед инструкцией		После инструкции	
W0	5555	W0	0040	
W1	1234	W1	0000	
W2	8000	W2	8000	
W4	0200	W4	0200	
SR	0000	SR	0002	(Z = 1)

Пример 2:

```
REPEAT      #17         ; Выполнить DIV.SD 18 раз
DIV.UD      W10, W12    ; Делить W11:W10 на W12
                        ; Запомнить частное в W0, остаток в W1
```

	Перед инструкцией		После инструкции
W0	5555	W0	01F2
W1	1234	W1	0100
w10	2500	w10	2500
W11	0042	W11	0042
W12	2200	W12	2200
SR	0000	SR	0000

DIV.F **Деление дробных**

Синтаксис: {метка:} DIVF Wm, Wn

Операнды: Wm ∈ [W0 ... W15]
Wn ∈ [W2 ... W15]

Действие: 0x0 → W0
Wm → W1
W1:W0/Wn → W0
Остаток → W1

Влияет на флаги: N, OV, Z, C

Код:

1101	1001	0ttt	t000	0000	ssss

Описание: Итеративное, знаковой дроби 16-бит на 16-бит деление, где делимое хранится в Wm и делитель хранится в Wn. При выполнении операции, W0 сначало очищается и Wm копируется в W1. 16-битное частное операции деления хранится в W0, и 16-битный остаток хранится в W1. Знак остатка будет такой же как знак делимого. Эта инструкция выполняется 18 раз, используя инструкцию REPEAT (с счётчиком итераций 17) для генерации корректного частного и остатка. Флаг N будет установлен, если остаток отрицательный и очищен в противном случае. Флаг OV будет установлен, если результат операции деления переполнен и очищен в противном случае. Флаг Z будет установлен, если остаток равен '0' и очищен иначе. Флаг C используется для обеспечения алгоритма деления и его конечное значение не надо использовать.
Биты 't' выбирают регистр делимого.
Биты 's' выбирают регистр делителя.

Примечание 1: Чтобы дробное деление было эффективным, Wm должен быть меньше чем Wn. Если Wm будет больше или равен Wn, неожиданный результат произойдёт, поскольку дробный результат будет больше или равен 1.0. Когда это произойдёт, бит состояния OV будет установлен и частное и остаток не должны использоваться.
2: Деление на ноль инициирует ловушку арифметической ошибки в течении первого цикла выполнения.
3: Эта инструкция может прерываться в каждой границе инструкции цикла.

Слова: 1

Циклы: 18 (плюс 1 для выполнения REPEAT)

Пример 1:

```
REPEAT      #17         ; Выполнить DIF 18 раз
DIVF        W8, W9      ; Делить W8 на W9
                        ; Запомнить частное в W0, остаток в W1
```

	Перед инструкцией		После инструкции	
W0	8000	W0	2000	
W1	1234	W1	0000	
W8	1000	W8	1000	
W9	4000	W9	4000	
SR	0000	SR	0002	(Z = 1)

Пример 2:

```
REPEAT      #17         ; Выполнить DIF 18 раз
DIVF        W8, W9      ; Делить W8 на W9
                        ; Запомнить частное в W0, остаток в W1
```

	Перед инструкцией		После инструкции	
W0	8000	W0	F000	
W1	1234	W1	0000	
W8	1000	W8	1000	
W9	8000	W9	8000	
SR	0000	SR	0002	(Z = 1)

Пример 2:

```
REPEAT      #17         ; Выполнить DIF 18 раз
DIVF        W0, W1      ; Делить W0 на W1
                        ; Запомнить частное в W0, остаток в W1
```

	Перед инструкцией		После инструкции	
W0	8002	W0	7FFE	
W1	8001	W1	8002	
SR	0000	SR	0008	(N = 1)

DO **Инициализация аппаратного литерального цикла**

Синтаксис: {метка:} DO #lit14, Expr

Операнды: lit14 ∈ [0 ... 16383]
Expr может быть абсолютным адресом, меткой или выражением.
Expr преобразуется компоновщиком в Slit16, где Slit16 ∈ [-32768 ... +32767].

Действие: PUSH DO shadows (DCOUNT, DOEND, DOSTART)
(lit14) → DCOUNT
(PC) + 4 → PC
(PC) → DOSTART
(PC) + (2 * Slit16) → DOEND
Инкремент DL<2:0> (CORCON<10:8>)

Влияет на флаги: DA

Код:

0000	1000	00kk	kkkk	kkkk	kkkk
0000	0000	nnnn	nnnn	nnnn	nnnn

Описание: Инициирует не верхний аппаратный DO цикл, который выполняется (lit14 + 1) раз. DO цикл начинается с адреса следующего за инструкцией DO, и оканчивается по адресу через 2 * Slit16 слов инструкций. 14-битный счётное значение (lit14) поддерживает максимальное количество зацикливаний 16384, и 16-битное значение смещения (Slit16) поддерживает смещения в 32к слов инструкций в обоих направлениях.

Когда эта инструкция выполняется, DCOUNT, DOSTART и DOEND сначала помещаются в их соответствующие теневые регистры, и затем обновляются новыми параметрами DO цикла, определёнными в инструкции. Затем инкрементируется счётчик уровня DO, DL<2:0> (CORCON<8:10>). После завершения выполнения DO цикла, сохранённые значения регистров DCOUNT, DOSTART и DOEND восстанавливаются, и DL<2:0> декрементируется.

Биты ‘k’ определяют счётчик цикла.
Биты ‘n’ являются знаковым литералом, который определяет величину смещения в инструкциях от PC до последней инструкции выполненной в цикле.

Специальные характеристики и ограничения:
Следующие характеристики и ограничения относятся к инструкции DO.
1. Использование нулевого значения счётчика цикла позволяет выполнить тело цикла один раз.
2. Указание размера цикла равным -2, -1 или 0 является не правильным. Может произойти неожиданный результат, если использовать такое смещение.
3. Самые последние две инструкции DO цикла могут не быть:
• инструкцией, которая изменяет программный поток
• инструкцией DO или REPEAT
Неожиданные результаты могут происходить, если любая из этих инструкций будет использована.
4. Если произойдёт аппаратная ловушка во второй или третьей от последней инструкции DO цикла, цикл может не функционировать правильно. Аппаратные ловушки включают исключения с уровнем приоритета от 13 до 15, включительно.

Примечание 1: Инструкция DO является прерываемой и поддерживает 1 аппаратный уровень вложения. Вложенность до 5 дополнительных уровней может быть обеспечена в программе пользователя. Смотреть “Руководство по семейству dsPIC30F” (DS70046) для деталей.

2: Компоновщик преобразует определённое выражение в смещение.

Слова: 2

Циклы: 2

Пример 1:

```
002000 LOOP6:      DO          #5, END6    ;Инициировать DO цикл (6 повторов)
002004             ADD         W1, W2, W3  ;Первая инструкция в цикле
002006             . . .
002008 END6:       . . .
00200A             SUB         W2, W3, W4  ;Последняя инструкция в цикле
00200C             . . .
```

	Перед инструкцией			После инструкции	
PC	00 2000		PC	00 2004	
DCOUNT	0000		DCOUNT	0005	
DOSTART	FF FFFF		DOSTART	00 2004	
DOEND	FF FFFF		DOEND	00 200A	
CORCON	0000		CORCON	0100	(DL = 1)
SR	0001	(C = 1)	SR	0201	(DA, C = 1)

Пример 2:

```
01C000 LOOP12:     DO  #0x160, END12       ;Инициировать DO цикл (353 повтора)
01C004             DEC     W1, W2          ;Первая инструкция в цикле
01C006             . . .
01C008             . . .
01C00A             . . .
01C00C             . . .
01C00E             . . .
01C010             CALL        _FIR88      ; Вызвать подпрограмму FIR88
01C014 END12:      NOP
```

	Перед инструкцией			После инструкции	
PC	00 2000		PC	00 2004	
DCOUNT	0000		DCOUNT	0005	
DOSTART	FF FFFF		DOSTART	00 2004	
DOEND	FF FFFF		DOEND	00 200A	
CORCON	0000		CORCON	0100	(DL = 1)
SR	0001	(C = 1)	SR	0201	(DA, C = 1)

DO **Инициализация аппаратного цикла Wn**

Синтаксис: {метка:} DO Wn, Expr

Операнды: Wn ∈ [W0 ... W15]
Expr может быть абсолютным адресом, меткой или выражением.
Expr преобразуется компоновщиком в Slit16, где Slit16 ∈ [-32768 ... +32767].

Действие: PUSH DO shadows (DCOUNT, DOEND, DOSTART)
(Wn) → DCOUNT
(PC) + 4 → PC
(PC) → DOSTART
(PC) + (2 * Slit16) → DOEND
Инкремент DL<2:0> (CORCON<10:8>)

Влияет на флаги: DA

Код:

0000	1000	1000	0000	0000	ssss
0000	0000	nnnn	nnnn	nnnn	nnnn

Описание: Инициирует не верхний аппаратный DO цикл, который выполняется (Wn + 1) раз. DO цикл начинается с адреса следующего за инструкцией DO, и оканчивается по адресу через 2 * Slit16 слов инструкций. 14-битный счётное значение Wn поддерживает максимальное количество зацикливаний 16384, и 16-битное значение смещения (Slit16) поддерживает смещения в 32к слов инструкций в обоих направлениях.

Когда эта инструкция выполняется, DCOUNT, DOSTART и DOEND сначала помещаются в их соответствующие теневые регистры, и затем обновляются новыми параметрами DO цикла, определёнными в инструкции. Затем инкрементируется счётчик уровня DO, DL<2:0> (CORCON<8:10>). После завершения выполнения DO цикла, сохранённые значения регистров DCOUNT, DOSTART и DOEND восстанавливаются, и DL<2:0> декрементируется.

Биты ‘s’ определяют регистр Wn, который содержит счётчик цикла.
Биты ‘n’ являются знаковым литералом, который определяет число инструкций смещения от (PC + 4), которые быть последними выполняемыми в цикле.

Специальные характеристики и ограничения:
Следующие характеристики и ограничения относятся к инструкции DO.
1. Использование нулевого значения счётчика цикла позволяет выполнить тело цикла один раз.
2. Указание размера цикла равным -2, -1 или 0 является не правильным. Может произойти неожиданный результат, если использовать такое смещение.
3. Самые последние две инструкции DO цикла могут не быть:
• инструкцией, которая изменяет программный поток
• инструкцией DO или REPEAT
Неожиданные результаты могут происходить, если любая из этих инструкций будет использована.
4. Если произойдёт аппаратная ловушка во второй или третьей от последней инструкции DO цикла, цикл может не функционировать правильно. Аппаратные ловушки включают исключения с уровнем приоритета от 13 до 15, включительно.

Примечание 1: Инструкция DO является прерываемой и поддерживает 1 аппаратный уровень вложения. Вложенность до 5 дополнительных уровней может быть обеспечена в программе пользователя. Смотреть “Руководство по семейству dsPIC30F” (DS70046) для деталей.

2: Компоновщик преобразует определённое выражение в смещение.

Слова: 2

Циклы: 2

Пример 1:

```
002000 LOOP6:       DO          W0, END6    ;Инициировать DO цикл (W0 повторов)
002004              ADD         W1, W2, W3  ;Первая инструкция в цикле
002006              . . .
002008              . . .
00200A              . . .
00200C              REPEAT      #6
00200E              SUB         W2, W3, W4  ;Последняя инструкция в цикле
002010 END6:        NOP                     ;(Требуемый NOP наполнитель)
```

	Перед инструкцией		После инструкции	
PC	00 2000	PC	00 2004	
W0	0012	W0	0012	
DCOUNT	0000	DCOUNT	0012	
DOSTART	FF FFFF	DOSTART	00 2004	
DOEND	FF FFFF	DOEND	00 2010	
CORCON	0000	CORCON	0100	(DL = 1)
SR	0000	SR	0080	(DA = 1)

Пример 2:

```
002000 LOOPA:       DO          W7, ENDA    ;Инициировать DO цикл (W7 повторов)
002004              SWAP        W0          ;Первая инструкция в цикле
002006              . . .
002008              . . .
00200A              . . .
002010 ENDA:        MOV         W1, [W2++]  ;Последняя инструкция в цикле
```

	Перед инструкцией		После инструкции	
PC	00 2000	PC	00 2004	
W7	E00F	W7	E00F	
DCOUNT	0000	DCOUNT	200F	
DOSTART	FF FFFF	DOSTART	00 2004	
DOEND	FF FFFF	DOEND	00 2010	
CORCON	0000	CORCON	0100	(DL = 1)
SR	0000	SR	0080	(DA = 1)

(C = 1)

ED **Евклидова дистанция (Не аккумулирует)**

Синтаксис: {метка:} ED Wm * Wm, Acc, [Wx], [Wy], Wxd
[Wx] + = kx, [Wy] + = ky,
[Wx] – = kx, [Wy] – = ky,
[W9 + W12], [W11 + W12],

Операнды: Acc ∈ [A,B]
Wm * Wm ∈ [W4 * W4, W5 * W5, W6 * W6, W7 * W7]
Wx ∈ [W8, W9]; kx ∈ [-6, -4, -2, 2, 4, 6]
Wy ∈ [W10, W11]; ky ∈ [-6, -4, -2, 2, 4, 6]
Wxd ∈ [W4 ... W7]

Действие: (Wm) * (Wm) → Acc(A or B)
([Wx] – [Wy]) → Wxd
(Wx) + kx → Wx
(Wy) + ky → Wy

Влияет на флаги: OA, OB, OAB, SA, SB, SAB

Код:

1111	00mm	A1xx	00ii	iijj	jj11

Описание: Вычисляется квадрат Wm, и также разницу предварительно выбранных значений определённых как [Wx] и [Wy]. Результат Wm * Wm расширяется знаком до 40 бит и добавляется в определённый аккумулятор. Результат [Wx] – [Wy] запоминается в Wxd, который может быть то же самый как Wm.
Операнды Wx, Wxd и Wyd определяют операции предварительной выборки, которые поддерживают косвенную и регистровую со смещением адресацию, как описано в части 4.14.1 “MAC упреждающая выборка”.

Биты ‘m’ выбирают регистр операнда Wm для возведения в квадрат.
Бит ‘A’ выбирает аккумулятор для результата.
Биты ‘x’ выбирают место назначения Wxd разницы предварительной выборки.
Биты ‘i’ выбирают Wx для операции предварительной выборки.
Биты ‘j’ выбирают Wy для операции предварительной выборки.

Примечание: Евклидова дистанция между двумя точками x и y — это наименьшее расстояние между ними. В двух- или трёхмерном случае — это прямая, соединяющая данные точки. Общей формулой для n-мерного случая (n переменных) является:

$$dist = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Слова: 1

Циклы: 1

Пример 1:

```
ED   W4*W4, A, [W8]+=2, [W10]-=2,W4  ; Квадрат W4 в ACCA
                                     ; [W8]-[W10] в W4
                                     ; Пост-инкремент W8
                                     ; Пост-декремент W10
```

	Перед инструкцией		После инструкции
W4	009A	W4	0057
W8	1100	W8	1102
W10	2300	W10	22FE
ACCA	00 3D0A 0000	ACCA	00 0000 5CA4
Данные 1100	007F	Данные 1100	007F
Данные 2300	0028	Данные 2300	0028
SR	0000	SR	0000

Пример 2:

```
ED   W5*W5, B, [W9]+=2, [W11+W12],   W5          ; Квадрат W5 в ACCB
```

```
; [W9]-[W11+W12] в W5
; Пост-инкремент W9
```

	Перед инструкцией
W5	43C2
W9	1200
W11	2500
W12	0008
ACCB	00 28E3 F14C
Данные 1200	6A7C
Данные 2508	2B3D
SR	0000

	После инструкции
W5	3F3F
W9	1202
W11	2500
W12	0008
ACCB	00 11EF 1F04
Данные 1200	6A7C
Данные 2508	2B3D
SR	0000

EDAC **Евклидова дистанция**

Синтаксис: {метка:} EDAC Wm * Wm, Acc, [Wx], [Wy], Wxd
[Wx] + = kx, [Wy] + = ky,
[Wx] – = kx, [Wy] – = ky,
[W9 + W12], [W11 + W12],

Операнды: Acc ∈ [A,B]
Wm * Wm ∈ [W4 * W4, W5 * W5, W6 * W6, W7 * W7]
Wx ∈ [W8, W9]; kx ∈ [-6, -4, -2, 2, 4, 6]
Wy ∈ [W10, W11]; ky ∈ [-6, -4, -2, 2, 4, 6]
Wxd ∈ [W4 ... W7]

Действие: (Acc(A or B)) + (Wm) * (Wm) → Acc(A or B)
([Wx] – [Wy]) → Wxd
(Wx) + kx → Wx
(Wy) + ky → Wy

Влияет на флаги: OA, OB, OAB, SA, SB, SAB

Код:

1111	00mm	A1xx	00ii	iijj	jj10

Описание: Вычисляется квадрат Wm, и опционально вычисляется разница предварительно выбранных значений определённых как [Wx] и [Wy]. Результат Wm * Wm расширяется знаком до 40 бит и запоминается в определённом аккумуляторе. Результат [Wx] – [Wy] запоминается в Wxd, который может быть тот же самый как Wm.
Операнды Wx, Wxd и Wyd определяют операции предварительной выборки, которые поддерживают косвенную и регистровую со смещением адресацию, как описано в части 4.14.1 “MAC упреждающая выборка”.

Биты ‘m’ выбирают регистр операнда Wm для возведения в квадрат.
Бит ‘A’ выбирает аккумулятор для результата.
Биты ‘x’ выбирают место назначения Wxd разницы предварительной выборки.
Биты ‘i’ выбирают Wx для операции предварительной выборки.
Биты ‘j’ выбирают Wy для операции предварительной выборки.

Примечание: Евклидова дистанция между двумя точками x и y — это наименьшее расстояние между ними. В двух- или трёхмерном случае — это прямая, соединяющая данные точки. Общей формулой для n-мерного случая (n переменных) является:

$$dist = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Слова: 1

Циклы: 1

Пример 1:

```
EDAC W4*W4, A, [W8]+=2, [W10]-=2,W4  ; Квадрат W4 добавить в ACCA
                                     ; [W8]-[W10] в W4
                                     ; Пост-инкремент W8
                                     ; Пост-декремент W10
```

	Перед инструкцией
W4	009A
W8	1100
W10	2300
ACCA	00 3D0A 3D0A
Данные 1100	007F
Данные 2300	0028
SR	0000

	После инструкции
W4	0057
W8	1102
W10	22FE
ACCA	00 3D0A 99AE
Данные 1100	007F
Данные 2300	0028
SR	0000

Пример 2:

```
EDAC W5*W5, B, [W9]+=2, [W11+W12],   W5          ; Квадрат W5 добавить
                                                 ; в ACCB
                                                 ; [W9]-[W11+W12] в W5
                                                 ; Пост-инкремент W9
```

	Перед инструкцией
W5	43C2
W9	1200
W11	2500
W12	0008
ACCB	00 28E3 F14C
Данные 1200	6A7C
Данные 2508	2B3D
SR	0000

	После инструкции
W5	3F3F
W9	1202
W11	2500
W12	0008
ACCB	00 3AD3 1050
Данные 1200	6A7C
Данные 2508	2B3D
SR	0000

EXCH **Обмен между Wns и Wnd**

Синтаксис: {метка:} EXCH Wns, Wnd

Операнды: Wns ∈ [W0 ... W15]
Wnd ∈ [W0 ... W15]

Действие: (Wns) ↔ (Wnd)

Влияет на флаги: Не влияет

Код:

1111	1101	0000	0ddd	d000	ssss

Описание: Обмет содержимого слов двух рабочих регистров. Прямая регистровая адресация может быть использована для Wns и Wnd.

Биты ‘d’ выбирают адрес первого регистра.
Биты ‘s’ выбирают адрес второго регистра.

Примечание: Эта инструкция выполняется только в режиме слова.

Слова: 1

Циклы: 1

Пример 1:

```
EXCH        W1, W9      ; Обменять содержимое W1 и W9
```

	Перед инструкцией		После инструкции
W1	55FF	W1	A3A3
W9	A3A3	W9	55FF
SR	0000	SR	0000

Пример 2:

```
EXCH        W4, W5      ; Обменять содержимое W4 и W5
```

	Перед инструкцией		После инструкции
W4	ABCD	W4	4321
W5	4321	W5	ABCD
SR	0000	SR	0000

FBCL **Найти первый изменённый бит слева**

Синтаксис: {метка:} FBCL Ws, [Ws], [Ws++], [Ws--], [++Ws], [--Ws], Wnd

Операнды: Ws ∈ [W0 ... W15]
Wnd ∈ [W0 ... W15]

Действие:
```
Max_Shift = 15
Sign = (Ws) & 0x8000
Temp = (Ws) << 1
Shift = 0
While ( (Shift < Max_Shift) && ( (Temp & 0x8000) == Sign) )
Temp = Temp << 1
Shift = Shift + 1
-Shift → (Wnd)
```

Влияет на флаги: C

Код:

1101	1111	0000	0ddd	dppp	ssss

Описание: Найти первый случай единицы (для позитивного значения) или нуля (для негативного значения), начиная от наиболее значимого бита после знакового бита Ws и работать к наименее значимому биту словного операнда. Результат номера бита расширяется знаком до 16 бит и помещается в Wnd.
Следующий наиболее значимый бит после знакового бита размещённый номер бита 0 и наименее значимый бит размещённый номер бита -14. Это упорядочение бит предусмотрено для немедленного использования Wd с инструкцией SFTAC для масштабирования значений вверх. Если битовое изменение не обнаружено, результат -15 возвращается и устанавливается флаг C. Когда битовое изменение обнаружено, флаг C очищен.

Биты ‘d’ выбирают регистр места назначения.
Биты ‘p’ выбирают режим адреса источника.
Биты ‘s’ выбирают регистр источник.

Примечание: Эта инструкция работает только в режиме слова.

Слова: 1

Циклы: 1

Пример 1:
```
FBCL        W1, W9      ; Найти первый изменённый бит слева в W1
                        ; и записать результат в W9
```

	Перед инструкцией		После инструкции
W1	55FF	W1	55FF
W9	FFFF	W9	0000
SR	0000	SR	0000

Пример 2:
```
FBCL        W1, W9      ; Найти первый изменённый бит слева в W1
                        ; и записать результат в W9
```

	Перед инструкцией		После инструкции	
W1	FFFF	W1	FFFF	
W9	BBBB	W9	FFF1	
SR	0000	SR	0001	(C = 1)

Пример 3:

```
FBCL        [W1++], W9   ; Найти 1-е битовое изменение слева в [W1]
                         ; и запомнить результат в W9
                         ; Посто-инкремент W1
```

	Перед инструкцией		После инструкции
W1	2000	W1	2002
W9	BBBB	W9	FFF9
Данные 2000	FF0A		FF0A
SR	0000	SR	0000

FF1L **Найти первую единицу слева**

Синтаксис: {метка:} FF1L Ws, [Ws], [Ws++], [Ws--], [++Ws], [--Ws], Wnd

Операнды: Ws ∈ [W0 ... W15]
Wnd ∈ [W0 ... W15]

Действие:
```
Max_Shift = 17
Temp = (Ws)
Shift = 1
While ( (Shift < Max_Shift) && !(Temp & 0x8000) )
Temp = Temp << 1
Shift = Shift + 1
If (Shift == Max_Shift)
0 → (Wnd)
Else
Shift → (Wnd)
```

Влияет на флаги: C

Код:

1100	1111	1000	0ddd	dppp	ssss

Описание: Найти первый случай единицы начиная от наиболее значимого бита Ws и работать к наименее значимому биту словного операнда. Результат номера бита расширяется нулём до 16 бит и помещается в Wnd.
Нумерация бит начинается с наболее значимого бита (назначен номер 1) и продвигается к наименее значимому биту (назначен номер 16). Нулевой результат показывает, что '1' не обнаружена и флаг C будет установлен. Если '1' обнаружена, то флаг C будет очищен.

Биты ‘d’ выбирают регистр места назначения.
Биты ‘p’ выбирают режим адреса источника.
Биты ‘s’ выбирают регистр источник.

Примечание: Эта инструкция работает только в режиме слова.

Слова: 1

Циклы: 1

Пример 1:
```
FF1L        W2, W5      ; Найти первую 1 слева в W2
                        ; и записать результат в W5
```

	Перед инструкцией		После инструкции
W2	000A	W2	000A
W5	BBBB	W5	000D
SR	0000	SR	0000

Пример 2:
```
FF1L        [W2++], W5  ; Найти первую 1 слева в [W2]
                        ; и записать результат в W5
                        ; Пост-инкремент W2
```

	Перед инструкцией		После инструкции	
W2	2000	W2	2002	
W5	BBBB	W5	0000	
Данные 2000	0000	Данные 2000	0000	
SR	0000	SR	0001	(C = 1)

FF1R **Найти первую единицу справа**

Синтаксис: {метка:} FF1R Ws, [Ws], [Ws++], [Ws--], [++Ws], [--Ws], Wnd

Операнды: Ws ∈ [W0 ... W15]
Wnd ∈ [W0 ... W15]

Действие:
Max_Shift = 17
Temp = (Ws)
Shift = 1
While ((Shift < Max_Shift) && !(Temp & 0x1))
Temp = Temp >> 1
Shift = Shift + 1
If (Shift == Max_Shift)
0 → (Wnd)
Else
Shift → (Wnd)

Влияет на флаги: C

Код:

1100	1111	0000	0ddd	dppp	ssss

Описание: Найти первый случай единицы начиная от наименее значимого бита Ws и работать к наиболее значимому биту словного операнда. Результат номера бита расширяется нулём до 16 бит и помещается в Wnd.
Нумерация бит начинается с наболее значимого бита (назначен номер 1) и продвигается к наименее значимому биту (назначен номер 16). Нулевой результат показывает, что '1' не обнаружена и флаг C будет установлен. Если '1' обнаружена, то флаг C будет очищен.

Биты ‘d’ выбирают регистр места назначения.
Биты ‘p’ выбирают режим адреса источника.
Биты ‘s’ выбирают регистр источник.

Примечание: Эта инструкция работает только в режиме слова.

Слова: 1

Циклы: 1

Пример 1:

```
FF1R        W1, W9      ; Найти первую 1 справа в W1
                        ; и записать результат в W9
```

	Перед инструкцией		После инструкции
W2	000A	W2	000A
W5	BBBB	W5	0002
SR	0000	SR	0000

Пример 2:

```
FF1R        [W1++], W9  ; Найти первую 1 справа в [W1]
                        ; и записать результат в W9
                        ; Пост-инкремент W2
```

	Перед инструкцией		После инструкции
W2	2000	W2	2002
W5	BBBB	W5	0010
Данные 2000	8000	Данные 2000	8000
SR	0000	SR	0000

GOTO **Безусловный переход**

Синтаксис: {метка:} GOTO Expr

Операнды: Expr может быть меткой или выражением (но не литералом).
Expr решается компоновщиком в lit23, где lit23 ∈ [0 ... 8388606].

Действие:
(PC) + 2 → PC
(PC<15:0>) → TOS
(W15) + 2 → W15
(PC<23:16>) → TOS
(W15) + 2 → W15
0 → PC<22:16>
(Wn<15:1>) → PC<15:1>
NOP → Регистр инструкции

Влияет на флаги: Не влияет

Код:

0000	0100	nnnn	nnnn	nnnn	nnn0
0000	0000	0000	0000	0nnn	nnnn

Описание: Безусловный переход во всём диапазоне программной памяти в пределах 4M слов инструкций. PC загружается 23-битным литералом определённым в инструкции. Поскольку PC должен всегда находится на границе чётного адреса, lit23<0> игнорируется.

Биты ‘n’ формируют целевой адрес.

Примечание: Компоновщик определяет lit23 которое будет использоваться.

Слова: 2

Циклы: 2

Пример 1:

```
026000                  GOTO        _THERE      ; Перейти на _THERE
026004                  MOV         W0, W1
  .                     . . .
  .                     . . .
027844      _THERE:     MOV         #0x400, W2  ; Выполнить код
027846                  . . .                   ; полученный здесь
```

	Перед инструкцией		После инструкции
PC	02 6000	PC	02 7844
SR	0000	SR	0000

Пример 2:

```
000100      _code:      . . .                   ; Начало кода
  .                     . . .
026000                  GOTO        _code+2     ; Перейти на _code+2
026004                  . . .
```

	Перед инструкцией		После инструкции
PC	02 6000	PC	00 0102
SR	0000	SR	0000

GOTO **Безусловный косвенный переход**

Синтаксис: {метка:} GOTO Wn

Операнды: Wn ∈ [W0 ... W15]

Действие: 0 → PC<22:16>
(Wn<15:1>) → PC<15:1>
0 → PC<0>
NOP → Регистр инструкции

Влияет на флаги: Не влияет

Код:

0000	0001	0100	0000	0000	ssss

Описание: Безусловный косвенный переход в пределах первых 32K слов программной памяти. В PC<22:16> загружается ноль и значение определённое в (Wn) загружается в PC<15:1>. Поскольку PC должен всегда находится на границе чётного адреса, Wn<0> игнорируется.

Биты ‘s’ выбирают регистр источник.

Слова: 1

Циклы: 2

Пример 1:

```
006000                  GOTO        W4          ;Безусловный переход по
006004                  MOV         W0, W1      ;16-битн. Значению в W4
  .                     . . .
  .                     . . .
007844      _THERE:     MOV         #0x400, W2  ; Выполнить код
007846                  . . .                   ; полученный здесь
```

	Перед инструкцией
W4	7844
PC	00 6000
SR	0000

	После инструкции
W4	7844
PC	00 7844
SR	0000

INC **Инкремент f**

Синтаксис: {метка:} INC{.B} f {,WREG}

Операнды: f ∈ [0 ... 8191]

Действие: (f) + 1 → место назначение, определённое D

Влияет на флаги: DC, N, OV, Z, C

Код:

1110	1100	0BDf	ffff	ffff	ffff

Описание: Увеличить на один содержимое файлового регистра и поместить результат в регистр места назначения. Необязательный операнд WREG определяет регистр места назначения. Если определён WREG, результат запоминается в WREG. Если WREG не определён, результат запоминается в файловом регистре.

Бит ‘B’ выбирает операцию с байтом или словом ('0' для слова, '1' для байта).
Бит ‘D’ выбирает место назначения (‘0’ для WREG, ‘1’ для файлового регистра).
Биты 'f' выбирают адрес файлового регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.
2: В качестве WREG используется рабочий регистр W0.

Слова: 1

Циклы: 1

Пример 1:

```
INC.B       0x1000      ; Инкремент (0x1000) (Режим байта)
```

	Перед инструкцией		После инструкции	
Данные 1000	8FFF	Данные 1000	8F00	
SR	0000	SR	0101	(DC, C = 1)

Пример 2:

```
INC         0x1000, WREG        ; Инкремент 0x1000 и запомнить
                                ; в WREG (Режим слова)
```

	Перед инструкцией		После инструкции	
WREG	ABCD	WREG	9000	
Данные 1000	8FFF	Данные 1000	8FFF	
SR	0000	SR	0108	(DC, N = 1)

INC **Инкремент Ws**

Синтаксис:	{метка:}	INC{.B}	Ws, [Ws], [Ws++], [Ws--], [++Ws], [--Ws],	Wd [Wd] [Wd++] [Wd--] [++Wd] [--Wd]
Операнды:	Ws ∈ [W0 ... W15] Wd ∈ [W0 ... W15]			
Действие:	(Ws) + 1 → Wd			
Влияет на флаги:	DC, N, OV, Z, C			

Код:	1110	1000	0Bqq	qddd	dppp	ssss

Описание: Добавить один к содержимому регистра источника Ws и поместить результат в регистр назначения Wd. Любая, прямая или косвенная регистровая адресация может быть использована для Ws и Wd.

Бит ‘B’ выбирает операцию с байтом или словом ('0' для слова, '1' для байта).
Биты ‘q’ выбирают режим адресации места назначения.
Биты ‘d’ выбирают регистр места назначения.
Биты ‘p’ выбирают режим адресации источника.
Биты ‘s’ выбирают адрес регистра источника.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

Слова: 1

Циклы: 1

Пример 1:

```
INC.B       W1, [++W2]   ; Пре-инкремент W2
                         ; Инкремент W1 и запомнить в W2
                         ; Режим байта
```

	Перед инструкцией		После инструкции	
W1	FF7F	W1	FF7F	
W2	2000	W2	2001	
Данные 2000	ABCD	Данные 2000	80CD	
SR	0000	SR	010C	(DC, N, OV=1)

Пример 2:

```
INC         W1, W2       ; Инкремент W1 и запомнить в W2.
                         ; Режим слова
```

	Перед инструкцией		После инструкции	
W1	FF7F	W1	FF7F	
W2	2000	W2	FF80	
SR	0000	SR	0108	(DC,N = 1)

INC2 **Инкремент f на 2**

Синтаксис: {метка:} INC2{.B} f {,WREG}

Операнды: f ∈ [0 ... 8191]

Действие: (f) + 2 → место назначения, определённое D

Влияет на флаги: DC, N, OV, Z, C

Код:

1110	1100	1BDf	ffff	ffff	ffff

Описание: Прибавить два к содержимому файлового регистра и поместить результат в регистр назначения. Необязательный операнд WREG определяет регистр места назначения. Если определён WREG, результат запоминается в WREG. Если WREG не определён, результат запоминается в файловом регистре.

Бит ‘B’ выбирает операцию с байтом или словом ('0' для слова, '1' для байта).
Бит ‘D’ выбирает место назначения (‘0’ для WREG, ‘1’ для файлового регистра).
Биты 'f' выбирают адрес файлового регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

Слова: 1

Циклы: 1

Пример 1:

```
INC2.B      0x1000      ; Инкремент 0x1000 на 2 (Режим байта)
```

	Перед инструкцией		После инструкции	
Данные 1000	8FFF	Данные 1000	8F01	
SR	0000	SR	0101	(DC, C = 1)

Пример 2:

```
INC2        0x1000, WREG        ; Инкремент 0x1000 на 2 и
                                ; запомнить в WREG (Режим слова)
```

	Перед инструкцией		После инструкции	
WREG	ABCD	WREG	9001	
Данные 1000	8FFF	Данные 1000	8FFF	
SR	0000	SR	0108	(DC, N = 1)

INC2 **Инкремент Ws на 2**

Синтаксис:	{метка:}	DEC2{.B}	Ws, [Ws], [Ws++], [Ws--], [++Ws], [--Ws],	Wd [Wd] [Wd++] [Wd--] [++Wd] [--Wd]

Операнды: Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

Действие: (Ws) + 2 → Wd

Влияет на флаги: DC, N, OV, Z, C

Код:

1110	1000	1Bqq	qddd	dppp	ssss

Описание: Добавить два к содержимому регистра источника Ws и поместить результат в регистр назначения Wd. Любая, прямая или косвенная регистровая адресация может быть использована для Ws и Wd.

Бит ‘B’ выбирает операцию с байтом или словом ('0' для слова, '1' для байта).
Биты ‘q’ выбирают режим адресации места назначения.
Биты ‘d’ выбирают регистр места назначения.
Биты ‘p’ выбирают режим адресации источника.
Биты ‘s’ выбирают адрес регистра источника.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

Слова: 1

Циклы: 1

Пример 1:

```
INC2.B      [W1, [++W2] ; Пре-инкремент W2
                        ; Инкремент на 2 и запомнить в W1
                        ; (Режим байта)
```

	Перед инструкцией		После инструкции	
W1	FF7F	W1	FF7F	
W2	2000	W2	2001	
Данные 2000	ABCD	Данные 2000	81CD	
SR	0000	SR	010C	(DC,N,OV = 1)

Пример 2:

```
INC2        W1, W2      ; Инкремент W1 на 2 и запомнить в W2
                        ; (режим слова)
```

	Перед инструкцией		После инструкции	
W1	FF7F	W1	FF7F	
W2	2000	W2	FF81	
SR	0000	SR	0108	(DC, N = 1)

IOR **Включающий операцию ИЛИ f и WREG**

Синтаксис: {метка:} IOR{.B} f {,WREG}

Операнды: f ∈ [0 ... 8191]

Действие: (f).ИЛИ.(WREG) → место назначение, определённое D

Влияет на флаги: N, Z

Код:

1011	0111	0BDf	ffff	ffff	ffff

Описание: Вычислить логическую операцию ИЛИ содержимого рабочего регистра по умолчанию WREG и содержимого файлового регистра, и поместить результат в регистр места назначения. Необязательный операнд WREG определяет регистр места назначения. Если определён WREG, результат запоминается в WREG. Если WREG не определён, результат запоминается в файловом регистре.

Бит ‘B’ выбирает операцию с байтом или словом (‘0’ для слова, ‘1’ для байта).
Бит ‘D’ выбирает место назначения (‘0’ для WREG, ‘1’ для файлового регистра).
Биты ‘f’ выбирают адрес файлового регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

2: WREG установлен на рабочий регистр W0.

Слова: 1

Циклы: 1

Пример 1:

```
IOR.B       0x1000      ; IOR WREG в (0x1000) (режим байта)
```

	Перед инструкцией		После инструкции
WREG	1234	WREG	1234
Данные 1000	FF00	Данные 1000	FF34
SR	0000	SR	0000

Пример 2:

```
IOR         0x1000, WREG            ; IOR (0x1000) в WREG
                                    ; (Режим слова)
```

	Перед инструкцией		После инструкции
WREG	1234	WREG	1FBF
Данные 1000	0FAB	Данные 1000	0FAB
SR	0008	(N = 1) SR	0000

IOR **Включающий операцию ИЛИ литерала и Wd**

Синтаксис: {метка:} IOR{.B} #lit10, Wn

Операнды: lit10 ∈ [0 ... 255] для операций с байтом
lit10 ∈ [0 ... 1023] для операций со словом
Wn ∈ [W0 ... W15]

Действие: lit10.ИЛИ.(Wn) → Wn

Влияет на флаги: N, Z

Код:

1011	0011	0Bkk	kkkk	kkkk	dddd

Описание: Вычислить логическую операцию ИЛИ 10-битного литерального операнда и содержимого рабочего регистра Wn, и разместить результат обратно в рабочий регистр Wn. Прямая регистровая адресация должна быть использована для Wn.

Бит ‘B’ выбирает операцию с байтом или словом (‘0’ для слова, ‘1’ для байта).
Биты ‘k’ определяют литеральный операнд.
Биты ‘d’ выбирают адрес рабочего регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

2: Для байтных операций, литерал может быть определён как беззнаковое значение [0:255]. Смотреть часть 4.6 “Использование 10-битных литеральных операндов” для информации на использование 10-битных литеральных операндов в байтном режиме.

Слова: 1

Циклы: 1

Пример 1:

```
IOR.B       #0xAA, W9    ; IOR 0xAA в W9 (Режим байта)
```

	Перед инструкцией		После инструкции	
W9	1234	W9	12BE	
SR	0000	SR	0008	(N = 1)

Пример 2:

```
IOR         #0x2AA, W4   ; IOR 0x2AA в W4 (Режим слова)
```

	Перед инструкцией		После инструкции	
W4	A34D	W4	A3EF	
SR	0000	SR	0008	(N = 1)

IOR **Включающий операцию ИЛИ Wb и короткого литерала**

Синтаксис: {метка:} IOR{.B} Wb, #lit5, Wd
[Wd]
[Wd++]
[Wd--]
[++Wd]
[--Wd]

Операнды: Wb ∈ [W0 ... W15]
lit5 ∈ [0 ... 31]
Wd ∈ [W0 ... W15]

Действие: (Wb).ИЛИ.lit5 → Wd

Влияет на флаги: N, Z

Код:

0111	0www	wBqq	qddd	d11k	kkkk

Описание: Вычислить логическую операцию ИЛИ содержимого базового регистра Wb и 5-битного литерала, и поместить результат в регистр места назначения Wd. Прямая регистровая адресация может быть использована для Wb. Прямая или косвенная регистровая адресация может быть использована для Wd.

Биты ‘w’ выбирают адрес базового регистра.
Бит ‘B’ выбирает операцию с байтом или словом (‘0’ для слова, ‘1’ для байта).
Биты ‘q’ выбирают режим адреса места назначения.
Биты ‘d’ выбирают регистр места назначения.
Биты ‘k’ обеспечивают литеральный операнд, пяти-битное целое число.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

Слова: 1

Циклы: 1

Пример 1:

```
IOR.B       W1, #0x5, [W9++]        ; IOR W1 и 0x5 (Режим байта)
                                    ; Запомнить в [W9]
                                    ; Пост-инкремент W9
```

	Перед инструкцией		После инструкции	
W1	AAAA	W1	AAAA	
W9	2000	W9	2001	
Данные 2000	0000	Данные 2000	00AF	
SR	0000	SR	0008	(N = 1)

Пример 2:

```
IOR         W1, #0x0, W9            ; IOR W1 с 0x0 (Режим слова)
                                    ; Запомнить в W9
```

	Перед инструкцией		После инструкции	
W1	0000	W1	0000	
W9	A34D	W9	0000	
SR	0000	SR	0002	(Z = 1)

IOR **Включающий операцию ИЛИ Wb и Ws**

Синтаксис:	{метка:}	IOR{.B}	Wb,	Ws, [Ws], [Ws++], [Ws--], [++Ws], [--Ws],	Wd [Wd] [Wd++] [Wd--] [++Wd] [--Wd]

Операнды: Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

Действие: (Wb).ИЛИ.(Ws) → Wd

Влияет на флаги: N, Z

Код:

0111	0www	wBqq	qddd	dppp	ssss

Описание: Вычислить логическую операцию ИЛИ содержимого регистра источника Ws и содержимого базового регистра Wb, и поместить результат в регистр места назначения Wd. Прямая регистровая адресация может быть использована для Wb. Любая, прямая или косвенная регистровая адресация может быть использована для Ws и Wd.

Биты ‘w’ выбирают адрес базового регистра.
Бит ‘B’ выбирает операцию с байтом или словом (‘0’ для слова, ‘1’ для байта).
Биты ‘q’ выбирают режим адреса места назначения.
Биты ‘d’ выбирают регистр места назначения.
Биты ‘p’ выбирают режим адреса источника.
Биты ‘s’ выбирают регистр источник.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

Слова: 1

Циклы: 1

Пример 1:

```
IOR.B       W1, [W5++], [W9++]      ; IOR W1 и W5, и
                                    ; запомнить в [W9] (Режим байта)
                                    ; Пост-инкремент W5 и W9
```

	Перед инструкцией		После инструкции	
W1	AAAA	W1	AAAA	
W5	2000	W5	2001	
W9	2400	W9	2401	
Данные 2000	1155	Данные 2000	1155	
Данные 2400	0000	Данные 2400	00FF	
SR	0000	SR	0008	(N = 1)

Пример 2:

```
IOR         IOR W1, W5, W9          ; IOR W1 и W5, и запомнить в W9
                                    ; (Режим слова)
```

	Перед инструкцией		После инструкции	
W1	AAAA	W1	AAAA	
W5	5555	W5	5555	
W9	A34D	W9	FFFF	
SR	0000	SR	0008	(N = 1)

LAC **Загрузить аккумулятор**

Синтаксис: {метка:} LAC Ws, {#Slit4,} Acc
[Ws],
[Ws++],
[Ws--],
[--Ws],
[++Ws],
[Ws+Wb],

Операнды: Ws ∈ [W0 ... W15]
Wb ∈ [W0 ... W15]
Slit4 ∈ [-8 ... +7]
Acc ∈ [A,B]

Действие: ShiftSlit4(Extend(Ws)) → Acc(A or B)

Влияет на флаги: OA, OB, OAB, SA, SB, SAB

Код:

1100	1010	Awww	wrrr	rggg	ssss

Описание: Читать содержимое регистра источника, опционально выполнять знаковый 4-битный сдвиг и запомнить результат в определённом аккумуляторе. Область сдвига есть -8:7, где отрицательный операнд означает левый арифметический сдвиг и позитивный операнд означает правый арифметический сдвиг. Данные загруженные в регистр источник приняты быть 1.15 дробными данными и автоматически расширены знаком (через бит 39) и нулём заполнены (биты [15:0]), до сдвига.

Бит ‘A’ определяет аккумулятор места назначения.
Биты ‘w’ выбирают адрес базового регистра Wb.
Биты ‘r’ кодирующие пре-сдвиг аккумулятора.
Биты ‘g’ выбирают режим адресации источника.
Биты ‘s’ выбирают адрес регистра источника Ws.

Примечание: Если операция перемещения более чем знаковое расширение данных на верхний регистр аккумулятора (AccxU), или вызвала насыщение, соответствующие биты переполнения и насыщения будут установлены.

Слова: 1

Циклы: 1

Пример 1:

```
LAC     [W4++], #-3, B      ; Загрузить ACCB с [W4] << 3
                            ; Содержимое [W4] не изменяется
                            ; Пост-инкремент W4
                            ; Допустить насыщение блокировано
                            ; (SATB = 0)
```

	Перед инструкцией		После инструкции	
W4	2000	W4	2002	
ACCB	00 5125 ABCD	ACCB	FF 9108 0000	
Данные 2000	1221	Данные 2000	1221	
SR	0000	SR	4800	(OB, OAB = 1)

Пример 2:

```
LAC     [--W2], #7, A       ; Пре-декремент W2
                            ; Загрузить ACCA с [W2] >> 7
                            ; Содержимое [W2] не изменено
                            ; Допустить насыщение блокировано
                            ; (SATA = 0)
```

	Перед инструкцией		После инструкции
W2	4002	W2	4000
ACCA	00 5125 ABCD	ACCA	FF FF22 1000
Данные 4000	9108	Данные 4000	9108
Данные 2002	1221	Данные 2002	1221
SR	0000	SR	0000

LNK **Резервировать фрейм стека**

Синтаксис: {метка:} LNK #lit14

Операнды: lit14 ∈ [0 ... 16382]

Действие: (W14) → (TOS)
(W15) + 2 → W15
(W15) → W14
(W15) + lit14 → W15

Влияет на флаги: Не влияет

Код:

1111	1010	00kk	kkkk	kkkk	kkk0

Описание: Эта инструкция резервирует фрейм стека размером lit14 байт для подпрограммы, вызывающей последовательность. Фрейм стека резервируется сохранением содержимого регистра указателя фрейма (W14) в стек, загрузкой обновлённого указателя стека (W15) в указатель фрейма и затем приращением указателя стека на беззнаковый 14-битный литеральный операнд. Эта инструкция поддерживает максимальный фрейм стека размером 16382 байт.

Биты ‘k’ определяют размер фрейма стека.

Примечание: Поскольку указатель стека может находится только на границе слова, lit14 должен быть чётным.

Слова: 1

Циклы: 1

Пример 1:

```
LNK     #0xA0     ;Резервирует фрейм стека размером 160 байт
```

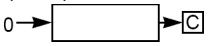
	Перед инструкцией		После инструкции
W14	2000	W14	2002
W15	2000	W15	20A2
Данные 2000	0000	Данные 2000	2000
SR	0000	SR	0000

LSR **Логический сдвиг вправо f**

Синтаксис: {метка:} LSR{.B} f {,WREG}

Операнды: f ∈ [0 ... 8191]

Действие: Для операции с байтом:
0 → Dest<7>
(f<7:1>) → Dest<6:0>
(f<0>) → C
Для операции со словом:
0 → Dest<15>
(f<15:1>) → Dest<14:0>
(f<0>) → C



Влияет на флаги: N, Z, C

Код:

1101	0101	0BDf	ffff	ffff	ffff

Описание: Сдвинуть содержимое файлового регистра на один бит вправо и поместить результат в регистр места назначения. Наименьший значащий бит файлового регистра сдвинут в бит переноса C регистра STATUS. Ноль сдвигается в наиболее значимый бит регистра места назначения. Не обязательный операнд WREG определяет регистр места назначения. Если WREG определён, результат загружается в WREG. Если WREG не определён, результат загружается в файловый регистр.

Бит ‘B’ выбирает операнд размером в байт или слово (‘0’ для слова, ‘1’ для байта).
Бит ‘D’ выбирает место назначения (‘0’ для WREG, ‘1’ для файлового регистра).
Биты ‘f’ выбирают адрес файлового регистра.

Примечание 1: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.
2: WREG установлен в рабочий регистр W0.

Слова: 1

Циклы: 1

Пример 1:

```
LSR.B       0x600       ; Логический сдвиг вправо (0x600) на один бит
                        ; (Режим байта)
```

	Перед инструкцией		После инструкции	
Данные 600	55FF	Данные 600	557F	
SR	0000	SR	0001	(C = 1)

Пример 2:

```
LSR         0x600, WREG ; Логический сдвиг вправо (0x600) на один бит
                        ; Запомнить в WREG
                        ; (Режим слова)
```

	Перед инструкцией		После инструкции	
Данные 600	55FF	Данные 600	55FF	
WREG	0000	WREG	2AFF	
SR	0000	SR	0001	(C = 1)

LSR **Логический сдвиг вправо Ws**

Синтаксис: {метка:} LSR{.B} Ws, Wd
[Ws], [Wd]
[Ws++], [Wd++]
[Ws--], [Wd--]
[++Ws], [++Wd]
[--Ws], [--Wd]

Операнды: Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

Действие:
Для операции с байтом:
0 → Wd<7>
(Ws<7:1>) → Wd<6:0>
(Ws<0>) → C
Для операции со словом:
0 → Wd<15>
(Ws<15:1>) → Wd<14:0>
(Ws<0>) → C

Влияет на флаги: N, Z, C

Код:

1101	0001	0Bqq	qddd	dppp	ssss

Описание: Сдвинуть содержимое регистра источника Ws на один бит вправо и поместить результат в регистр места назначения Wd. Наименьший значимый бит регистра Ws сдвигается в разряд C регистра STATUS. Ноль сдвигается в старший значащий бит Wd. Прямая или косвенная регистровая адресация может быть использована для Ws и Wd.

Бит ‘B’ выбирает операцию с байтом или словом (‘0’ для слова, ‘1’ для байта).
Биты ‘q’ выбирают режим адреса места назначения.
Биты ‘d’ выбирают регистр места назначения.
Биты ‘p’ выбирают режим адреса источника.
Биты ‘s’ выбирают регистр источник.

Примечание: Расширение .B в инструкции обозначает предпочтение операции с байтом перед операцией со словом. Вы можете использовать расширение .W для обозначения операции со словом, но это не обязательно.

Слова: 1

Циклы: 1

Пример 1:

```
LSR.B       W0, W1      ; LSR W0 (Режим байта)
                        ; Сохранить результат в W1
```

	Перед инструкцией		После инструкции	
W0	FF03	W0	FF03	
W1	2378	W1	2301	
SR	0000	SR	0001	(C = 1)

Пример 2:

```
LSR         W0, W1      ; LSR W0 (Режим слова)
                        ; Сохранить результат в W1
```

	Перед инструкцией		После инструкции
W0	8000	W0	8000
W1	2378	W1	4000
SR	0000	SR	0000

LSR **Логический сдвиг вправо к короткий литерал**

Синтаксис: {метка:} LSR Wb, #lit4, Wnd

Операнды: Wb ∈ [W0 ... W15]
lit4 ∈ [0...15]
Wnd ∈ [W0 ... W15]

Действие: lit4<3:0> → Shift_Val
0 → Wnd<15:15-Shift_Val + 1>
Wb<15:Shift_Val> → Wnd<15-Shift_Val:0>

Влияет на флаги: N, Z

Код:

1101	1110	0www	wddd	d100	kkkk

Описание: Логически сдвинуть вправо содержимое регистра источника Wb на количество позиций равных 4-битному без знаковому литералу, и запомнить результат в регистре места назначения Wnd. После того как сдвиг будет выполнен, результат будет расширен знаком. Для Wb и Wnd можно использовать прямую регистровую адресацию.

Бит ‘B’ выбирает операцию с байтом или словом (‘0’ для слова, ‘1’ для байта).
Биты ‘w’ выбирают адрес базового регистра.
Биты ‘d’ выбирают регистр места назначения.
Биты ‘k’ обеспечивают литеральный операнд.

Примечание: Эта инструкция работает только в режиме слова.

Слова: 1

Циклы: 1

Пример 1:

```
LSR          W4, #14, W5  ; LSR W4 на 14
                          ; Запомнить результат в W5
```

	Перед инструкцией		После инструкции
W4	C800	W4	C800
W5	1200	W5	0003
SR	0000	SR	0000

Пример 2:

```
LSR          W4, #1, W5   ; LSR W4 на 1
                          ; Запомнить результат в W5
```

	Перед инструкцией		После инструкции
W4	0505	W4	0505
W5	F000	W5	0282
SR	0000	SR	0000

LSR **Логический сдвиг вправо к Wns**

Синтаксис: {метка:} LSR Wb, Wns, Wnd

Операнды: Wb ∈ [W0 ... W15]
Wns ∈ [W0 ...W15]
Wnd ∈ [W0 ... W15]

Действие: Wns<4:0> → Shift_Val
0 → Wnd<15:15-Shift_Val + 1>
Wb<15:Shift_Val> → Wnd<15 - Shift_Val:0>

Влияет на флаги: N, Z

Код:

1101	1110	0www	wddd	d000	ssss

Описание: Логически сдвинуть вправо содержимое регистра источника Wb на количество позиций равных 5 наименьшим значащим битам регистра Wns (вплоть до 15 позиций) и запомнить результат в регистре места назначения Wnd. Для Wb, Wns и Wnd можно использовать прямую регистровую адресацию.

Биты ‘w’ выбирают адрес базового регистра.
Биты ‘d’ выбирают регистр места назначения.
Биты ‘s’ выбирают регистр источник.

Примечание 1: Эта инструкция работает только в режиме слова.
2: Если Wns больше чем 15, Wnd = 0x0 если Wb положителен, и Wnd = 0xFFFF если Wb отрицателен.

Слова: 1

Циклы: 1

Пример 1:

```
LSR     W0, W1, W2      ; LSR W0 на W1
                        ; Запомнить результат в W2
```

	Перед инструкцией		После инструкции
W0	C00C	W0	C00C
W1	0001	W1	0001
W2	2390	W2	6006
SR	0000	SR	0000

Пример 2:

```
LSR     W5, W4, W3      ; ASR W0 на W5 и запомнить в W6
```

	Перед инструкцией		После инструкции	
W3	DD43	W3	0000	
W4	000C	W4	000C	
W5	0800	W5	0800	
SR	0000	SR	0002	(Z = 1)

MAC **Умножить и суммировать**

Синтаксис: {метка:} MAC Wm*Wn, Acc {,[Wx], Wxd} {,[Wy], Wyd} {,AWB}
{,[Wx] + = kx, Wxd} {,[Wy] + = ky, Wyd}
{,[Wx] – = kx, Wxd} {,[Wy] – = ky, Wyd}
{,[W9 + W12], Wxd} {,[W11 + W12], Wyd}

Операнды: Wm * Wn ∈ [W4 * W5, W4 * W6, W4 * W7, W5 * W6, W5 * W7, W6 * W7]
Acc ∈ [A,B]
Wx ∈ [W8, W9]; kx ∈ [-6, -4, -2, 2, 4, 6]; Wxd ∈ [W4 ... W7]
Wy ∈ [W10, W11]; ky ∈ [-6, -4, -2, 2, 4, 6]; Wyd ∈ [W4 ... W7]
AWB ∈ [W13, [W13] + = 2]

Действие: (Acc(A or B)) + (Wm) * (Wn) → Acc(A or B)
([Wx]) → Wxd; (Wx) + kx → Wx
([Wy]) → Wyd; (Wy) + ky → Wy
(Acc(B or A)) rounded → AWB

Влияет на флаги: OA, OB, OAB, SA, SB, SAB

Код:

1100	0mmm	A0xx	yyii	iijj	jjaa

Описание: Умножить содержимое двух рабочих регистров, опциональная упреждающая выборка операндов в подготовке для другой инструкции типа MAC и опционально записать результатов неопределённого аккумулятора. 32-битный результат знакового умножения расширен знаком до 40 бит и добавлен в определённый аккумулятор.
Операнды Wx, Wxd, Wy и Wyd определяют дополнительные предварительно выбранные операнды, которые поддерживают косвенную и регистровую смещённую адресацию, как описано в части 4.14.1 “Упреждающая выборка MAC”. Операнд AWB определяет опциональную запись “другого” аккумулятора, как описано в части 4.14.4 “Обратная запись MAC”.

Биты ‘m’ выбирают регистры операндов Wm и Wn для умножения.
Бит ‘A’ выбирает аккумулятор для результата.
Биты ‘x’ выбирают место назначения выбранного с упреждением Wxd.
Биты ‘y’ выбирают место назначения выбранного с упреждением Wyd.
Биты ‘i’ выбирают операцию выборки с упреждением Wx.
Биты ‘j’ выбирают операцию выборки с упреждением Wy.
Биты ‘a’ выбирают аккумулятор места назначения обратной записи.

Примечание: Бит IF, CORCON<0>, определён если умножается дробные или целые.

Слова: 1

Циклы: 1

Пример 1:

```
MAC          W4*W5, A, [W8]+=6, W4, [W10]+=2, W5

; Умножить W4*W5 и добавить в ACCA
; Выбрать [W8] в W4, Пост-инкремент W8 на 6
; Выбрать [W10] в W5, Пост-инкремент W10 на 2
; CORCON = 0x00C0 (дробное умножение, нормальное насыщение)
```

	Перед инструкцией		После инструкции
W4	A022	W4	2567
W5	B900	W5	909C
W8	0A00	W8	0A06
W10	1800	W10	1802
ACCA	00 1200 0000	ACCA	00 472D 2400
Данные 0A00	2567	Данные 0A00	2567
Данные 1800	909C	Данные 1800	909C
CORCON	00C0	CORCON	00C0
SR	0000	SR	0000

Пример 2:

```
MAC          W4*W5, A, [W8]-=2, W4, [W10]+=2, W5, W13
; Умножить W4*W5 и добавить в ACCA
; Выбрать [W8] в W4, Пост-инкремент W8 на 2
; Выбрать [W10] в W5, Пост-инкремент W10 на 2
; Обратная запись ACCB в W13
; CORCON = 0x00D0 (дробное умножение, супер насыщение)
```

	Перед инструкцией		После инструкции	
W4	1000	W4	5BBE	
W5	3000	W5	C967	
W8	0A00	W8	09FE	
W10	1800	W10	1802	
W13	2000	W13	0001	
ACCA	23 5000 2000	ACCA	23 5600 2000	
ACCB	00 0000 8F4C	ACCB	00 0000 1F4C	
Данные 0A00	5BBE	Данные 0A00	5BBE	
Данные 1800	C967	Данные 1800	C967	
CORCON	00D0	CORCON	00D0	
SR	0000	SR	8800	(OA, OAB = 1)

MAC **Возвести в квадрат и суммировать**

Синтаксис: {метка:} MAC Wm*Wm, Acc {,[Wx], Wxd} {,[Wy], Wyd}
{,[Wx] + = kx, Wxd} {,[Wy] + = ky, Wyd}
{,[Wx] – = kx, Wxd} {,[Wy] – = ky, Wyd}
{,[W9 + W12], Wxd} {,[W11 + W12], Wyd}

Операнды: Wm * Wm ∈ [W4 * W4, W5 * W5, W6 * W6, W7 * W7]
Acc ∈ [A,B]
Wx ∈ [W8, W9]; kx ∈ [-6, -4, -2, 2, 4, 6]; Wxd ∈ [W4 ... W7]
Wy ∈ [W10, W11]; ky ∈ [-6, -4, -2, 2, 4, 6]; Wyd ∈ [W4 ... W7]

Действие: (Acc(A or B)) + (Wm) * (Wm) → Acc(A or B)
([Wx]) → Wxd; (Wx) + kx → Wx
([Wy]) → Wyd; (Wy) + ky → Wy

Влияет на флаги: OA, OB, OAB, SA, SB, SAB

Код:

1111	00mm	A0xx	yyii	iijj	jj00

Описание: Возвести в квадрат содержимое рабочего регистра, опциональная упреждающая выборка операндов в подготовке для другой инструкции типа MAC и опционально записать результатов неопределённого аккумулятора. 32-битный результат знакового умножения расширен знаком до 40 бит и добавлен в определённый аккумулятор.
Операнды Wx, Wxd, Wy и Wyd определяют дополнительные предварительно выбранные операнды, которые поддерживают косвенную и регистровую смещённую адресацию, как описано в части 4.14.1 “Упреждающая выборка MAC”.

Биты ‘m’ выбирают регистры операндов Wm и Wn для умножения.
Бит ‘A’ выбирает аккумулятор для результата.
Биты ‘x’ выбирают место назначения выбранного с упреждением Wxd.
Биты ‘y’ выбирают место назначения выбранного с упреждением Wyd.
Биты ‘i’ выбирают операцию выборки с упреждением Wx.
Биты ‘j’ выбирают операцию выборки с упреждением Wy.

Примечание: Бит IF, CORCON<0>, определён если умножается дробные или целые.

Слова: 1

Циклы: 1

Пример 1:

```
MAC          W4*W4, B, [W9+W12], W4, [W10]-=2, W5

; Возвести в квадрат W4 и добавить в ACCB
; Выбрать [W9+W12] в W4
; Выбрать [W10] в W5, Пост-декремент W10 на 2
; CORCON = 0x00C0 (дробное умножение, нормальное насыщение)
```

	Перед инструкцией		После инструкции
W4	A022	W4	A230
W5	B200	W5	650B
W9	0C00	W9	0C00
W10	1900	W10	18FE
W12	0020	W12	0020
ACCB	00 2000 0000	ACCB	00 67CD 0908
Данные 0C20	A230	Данные 0C20	A230
Данные 1900	650B	Данные 1900	650B
CORCON	00C0	CORCON	00C0
SR	0000	SR	0000

Пример 2:

```
MAC          W7*W7, A, [W11]-=2, W7

; Возвести в квадрат W7 и добавить в ACCA
; Выбрать [W11] в W7, Пост-декремент W11 на 2
; CORCON = 0x00D0 (дробное умножение, супер насыщение)
```

	Перед инструкцией		После инструкции	
W7	76AE	W7	23FF	
W11	2000	W11	1FFE	
ACCA	FE 9834 4500	ACCA	FF 063E 0188	
Данные 2000	23FF	Данные 2000	23FF	
CORCON	00D0	CORCON	00D0	
SR	0000	SR	8800	(OA, OAB = 1)