МИКРОСХЕМЫ ИНТЕГРАЛЬНЫЕ

1867ВЦ4Т

Руководство пользователя №2

ОПИСАНИЕ ИНСТРУКЦИЙ

2006

Содержание

Введение

Набор инструкций процессора цифровой обработки сигналов 1867ВЦ4Т

Настоящий документ является приложением к КФДЛ.431299.010ТО микросхемы 1867ВЦ4Т и описывает систему команд этого процессора. Благодаря глубокой конвейеризации и разветвленной системе команд процессор 1867ВЦ4Т одинаково хорошо поддерживает численно емкие операции обработки сигналов и приложения общего назначения, такие как многопроцессорная обработка и высокоскоростное управление.

Необходимо отметить, что под термином "мнемоника" в настоящем документе подразумевается символьное обозначение инструкции, поступающей на вход транслятора (ассемблера). Для многих мнемоник существует несколько вариантов синтаксиса, отличающихся наличием и количеством различных операндов, различными вариантами их расположения и/или сдвига; вариантами размещения и/или сдвига полученного результата и т.д. Поэтому после ассемблирования одной и той же мнемоники с разными синтаксисами программа-транслятор выдает разные двоичные коды операций процессора (opcode), которые представляют собой, по сути, самостоятельные инструкции. Несмотря на возможность создания программного кода с использованием высокоуровневых языков, для оптимизации объема задействованной пограммной памяти и обеспечения вычислений в реальном масштабе времени, разработчики широко применяют ассемблер, как минимум, для написания наиболее часто повторяемых и/или наиболее критичных по времени сегментов программы. Зачастую, один и тот же результат может быть получен с использованием разных инструкций и их сочетаний. Настоящий документ содержит всю необходимую информацию для создания оптимального исходного ассемблерного кода. В нем подробно изложены все особенности выполнения каждой инструкции, включая количество циклов, требуемых для ее выполнения в конвейере с учетом различных комбинаций хранения программного кода и операндов.

Настоящий документ состоит из четырех основных разделов, структурированных следующим образом:

– Раздел 1 содержит описание символов и аббревиатур набора инструкций и пример описания инструкции.

– Раздел 2 включает краткое описание инструкций, классифицированное по их функциональному назначению.

– Раздел 3 описывает классы инструкций и количество циклов их выполнения для различных комбинаций хранения программы кода и операндов.

– Раздел 4 посвящен детальному описанию каждой инструкции.

В качестве дополнения в конце документа приведены коды условий их возможные комбинации, применяемые в условных инструкциях, а также формат регистров состояния процессора.

1 Символы и аббревиатуры набора инструкций, пример описания инструкции

1.1 Набор инструкций: символы и аббревиатуры

В таблицах с 1 по 4 приводятся символы и аббревиатуры, используемые в разделе краткое описание инструкций (раздел 2) и в индивидуальном описании инструкций (раздел 4).

Таблица 1 – Символы и аббревиатуры набора инструкций

Символ	Описание
A	Аккумулятор A
ALU	Арифметико-логическое устройство
AR	Вспомогательный регистр общего назначения (РОН)
ARx	Вспомогательный регистр с определенным номером (0 ≤x≤7)
ARP	Поле в ST0, указывающее на вспомогательный регистранице это 3х разрядное поле, которое указывает на выбранный (текущий) вспомогательный регистр (AR)
ASM	5-разрядное поле в ST1, определяющее режим сдвига в аккумуляторе (-16 ≤ASM ≤15)
B	Аккумулятор B
BRAF	Флаг активизации повтора блока в ST1
BRC	Счетчик повтора блока
BITC	4-разрядное значение, которое определяет, какой бит (номер бита) в указанной ячейке памяти данных будет проверен инструкцией тестирования битов (0 ≤ BITC ≤ 15)
C16	Бит в ST1, определяющий режим арифметических операций: две 16-разрядых операции/операция с двойной точностью
C	Бит переноса в ST0
CC	2-разрядный код условия (0 ≤ CC ≤ 3)
CMPT	Бит режима совместимости в ST1
CPL	Бит режима компиляции в ST1
cond	Операнд, представляющий собой условие, тестируемое в условных инструкциях
[D]	Опция задержки (выполнения инструкции)
DAB	Адресная шина D
DAR	Регистр адреса DAB
dmad	16-разрядный непосредственный адрес памяти данных (0 ≤ dmad ≤ 65 535)
Dmem	Операнд памяти данных
DP	9-разрядное поле в ST0, указывающее на страницу памяти данных (0 ≤ DP ≤ 511)
dst	Аккумулятор, в который будет записан результат операции – приемник (A или B)
dst_	Аккумулятор, противоположный (альтернативный) приемному: если dst = A, то dst_ = B; если dst = B, то dst_ = A
EAB	Адресная шина E
EAR	Регистр адреса EAB
FRCT	Бит в ST1 – указатель дробного режима
hi(A)	Старшая часть (старшее слово) аккумулятора A (биты 31-16)
HM	Бит HOLD-режима {фиксации или захвата} (в ST1)
IFR	Регистр флага прерывания
INTM	Бит в ST1, указывающий на режим прерывания
K	Короткое непосредственное значение длиной менее 9 разрядов
k3	3-разрядное непосредственное значение (0 ≤ k3 ≤ 7)
k5	5-разрядное непосредственное значение (-16 ≤ k5 ≤ 15)
k9	3-разрядное непосредственное значение (0 ≤ k9 ≤ 511)
lk	16-разрядное длинное непосредственное значение
Lmem	32-разрядный единый операнд памяти данных, использующий длинно-словную адресацию
mmr, MMR	Регистр, отображаемый в память
MMRx, MMRy	Регистр, картированный в памяти, AR0-AR7 или SP
n	Количество слов после инструкции XC; n = 1 или 2

Окончание таблицы 1

Символ	Описание
N	Номер статусного регистра, модифицируемого инструкциями RSBX, SSBX: N = 0 – регистр состояния ST0; N = 1 – регистр состояния ST1
OVA	Флаг переполнения для аккумулятора A (в ST0)
OVB	Флаг переполнения для аккумулятора B (в ST0)
OVdst	Флаг переполнения приемного аккумулятора (A или B)
OVdst_	Флаг переполнения аккумулятора, противоположного приемному (A или B)
OVsrc	Флаг переполнения аккумулятора–источника (A или B)
OVM	Бит режима обработки переполнения (в ST1)
PA	16-разрядный непосредственный адрес порта ($0 \le PA \le 65\,535$)
PAR	Регистр программного адреса
PC	Счетчик команд
pmad	16-разрядный непосредственный адрес памяти программ ($0 \le pmad \le 65\,535$)
Pmem	Операнд в программной памяти
PMST	Регистр состояния режимов процессора
prog	Операнд в программной памяти
[R]	Опция округления
RC	Счетчик повторов
REA	Регистр конечного адреса повторяемого блока
rnd	Округление
RSA	Регистр стартового адреса повторяемого блока
RTN	Регистр быстрого возврата, используемый в инструкции RETF[D]
SBIT	4-разрядное значение, указывающее номер бита в регистре состояния, который будет модифицирован инструкциями RSBX, SSBX ($0 \le SBIT \le 15$)
SHFT	4-разрядное значение сдвига ($0 \le SHFT \le 15$)
SHIFT	5-разрядное значение сдвига ($-16 \le SHIFT \le 15$)
Sind	Одинарный операнд памяти данных, использующий косвенную адресацию
Smem	16-разрядный одинарный операнд памяти данных
SP	Указатель стека
src	Аккумулятор–источник (A или B)
ST0, ST1	Регистры состояния 0 и 1
SXM	Бит режима знакового расширения (ST1)
T	Регистр временного хранения
TC	Тестовый флаг в ST0
TOS	Вершина стека
TRN	Регистр перемещений
TS	Значение сдвига, определяемое битами 5-0 T–регистра ($-16 \le TS \le 15$)
uns	Беззнаковый
XF	Бит состояния внешнего флага в ST1
Xmem	16-разрядный операнд памяти данных, используемый в инструкциях с двойными операндами и некоторых инструкциях с одинарными операндами
Ymem	16-разрядный операнд памяти данных, используемый в инструкциях с двойными операндами
– – SP	Значение указателя стека уменьшается на 1 (декремент на 1)
+ + SP	Значение указателя стека увеличивается на 1 (инкремент на 1)
+ + PC	Значение счетчика команд увеличивается на 1 (инкремент на 1)

Таблица 2 – Символы и аббревиатуры кода операций

Символ	Описание
A	Разряд адреса памяти данных
ARX	3-разрядная величина, определяющая вспомогательный регистр
BITC	4-разрядный код бита
CC	2-разрядный код условия
CCCC CCCC	8-разрядный код условия
COND	4- разрядный код условия
D	Разряд выбора аккумулятора–приемника D = 0 – аккумулятор A D = 1 – аккумулятор B
I	Разряд режима адресации I = 0 – режим прямой адресации I = 1 – режим косвенной адресации
K	Короткий непосредственный операнд меньше 9 разрядов
MMRX	4-разрядное значение, указывающее на один из девяти картированных в памяти регистров ($0 \leq MMRX \leq 8$)
MMRY	4-разрядное значение, указывающее на один из девяти картированных в памяти регистров ($0 \leq MMRY \leq 8$)
N	Одиночный бит
NN	2-разрядное значение, определяющее тип прерывания
R	Разряд опции округления (rnd) R = 0 – инструкция выполняется без округления R = 1 – результат округляется
S	Разряд выбора аккумулятора–источника S = 0 – аккумулятор A S = 1 – аккумулятор B
SBIT	4-разрядный номер бита статусного регистра
SHFT	4-разрядное значение сдвига ($0 \leq SHFT \leq 15$)
SHIFT	5-разрядное значение сдвига ($-16 \leq SHIFT \leq 15$)
X	Разряд памяти данных
Y	Разряд памяти данных
Z	Разрзяд активизации задержанных инструкций Z = 0 – инструкция выполняется без задержки Z = 1 – инструкция выполняется с задержкой

Таблица 3 – Синтаксис набора инструкций (условные символы и форматы)

Символ	Описание
Жирные символы	Обязательная и неизменяемая часть синтаксиса инструкций выделяется жирным шрифтом. *Пример:* В синтаксисе ADD ***Xmem, Ymem, dst*** можно использовать различные значения для ***Xmem*** и ***Ymem***, но слово ADD является неизменяемым и вводится, как приведено
Наклонные символы	Переменные в синтаксисе инструкций выделяются наклонным шрифтом. *Пример:* В синтаксисе ADD ***Xmem, Ymem, dst*** можно использовать различные значения для ***Xmem*** и ***Ymem***
[x]	Операнд в квадратных скобках является опционным (необязательным). *Пример:* В синтаксисе ADD ***Smem [, SHIFT], src [, dst]*** необходимо обязательно указать значения ***Smem*** и ***src***, а ***SHIFT*** и ***dst*** указываются в случае необходимости. Если не указывать ***SHIFT*** и ***dst*** в мнемоническом синтаксисе, компилятор сформирует двоичный код инструкции таким образом, что сложение будет выполнено без сдвига, а результат запишется в то место, откуда брались исходные данные (по умолчанию ***dst*** = ***src***)
#	Префикс константы, используемой для непосредственной адресации. Применяется с короткими и длинными непосредственными операндами в инструкциях, допускающих различные режимы адресации, чтобы избежать неопределенности и указать на непосредственный операнд. *Например*: RPT #15 – короткая непосредственная адресация; следующая инструкция будет выполнена 16 раз. RPT 15 – прямая адресация; количество повторов следующей инструкции определяется значением ячейки памяти с адресом 15 на соответствующей странице (в зависимости от DP). Допускается применение в инструкциях, использующих только непосредственные операнды. *Например*: RPTZ A,#15 эквивалентно RPTZ A, 15
(abc)	Содержимое регистра или ячейки abc. *Пример*: (src) означает *содержимое аккумулятора–источника*
x → y	Значение x присваивается регистру или ячейке y. *Пример*: (Smem) → dst означает, что содержимое указанной ячейки памяти загружается в аккумулятор–приемник
r(n-m)	Разряды от n до m регистра или ячейки r. *Пример*: src(15-0) разряды с 15-го по 0-вой аккумулятора–источника
<< nn	Сдвиг на nn разрядов влево (отрицательный или положительный)
||	Параллельная инструкция
\\	Вращение влево
//	Вращение вправо
~x	Логическая инверсия от x
| x |	Абсолютное значение x
AAh	Показывает что значение AA представлено в шестнадцатиричном виде

Таблица 4 – Операторы набора инструкций

Символы	Операторы	Просмотр (анализ)
+ – ~	Унарные плюс, минус, логическая инверсия	Справа налево
* / %	Умножение, деление, модуль	Слева направо
+ –	Сложение, вычитание	Слева направо
<< >>	Сдвиг влево, сдвиг вправо	Слева направо
< < <	Логический сдвиг влево	Слева направо
< ≤	Меньше чем, меньше или равно	Слева направо
> ≥	Больше чем, больше или равно	Слева направо
≠ !=	Не равно	Слева направо
&	Поразрядное “И”	Слева направо
^	Поразрядное “Исключающее ИЛИ”	Слева направо
|	Поразрядное “ИЛИ”	Слева направо
Примечание – +, –, * имеют приоритет над бинарными формами.		

1.2 Пример описания инструкции

Это пример типичного описания инструкции, позволяющий ознакомиться с форматом описания инструкций и объясняюший, что описывает каждый заголовок. В данном разделе при описании каждой инструкции представлена следующая информация:

- Ассемблерный синтаксис.
- Операнды.
- Опкод (код операции – двоичное представление инструкции).
- Выполнение.
- Статусные биты.
- Количество слов.
- Описание.
- Количество циклов.
- Классы.
- Примеры.

Описание каждой инструкции начинается с ассемблерного синтаксиса. Метки могут размещаться перед инструкцией в той же строке или в первом столбце предыдущей строки. Синтаксическое выражение может заканчиваться дополнительным полем комментария. Следующие поля должны быть разделены пробелами:

- Метка.
- Инструкция и операнды.
- Комментарий.

Синтаксис

1: ПРИМЕР *Smem, src*
2: ПРИМЕР *Smem,* TS, *src*
3: ПРИМЕР *Smem,* 16, *src* [, *dst]*
4: ПРИМЕР *Smem* [, *SHIFT], src [, dst]*

Описание каждой инструкции начинается с ассемблерного синтаксиса. Смотри таблицы 1–3, описывающие символы синтаксиса инструкций.

Операнды

Smem:	Одинарный операнд памяти данных
Xmem, Ymem:	Двойные операнды памяти данных
src, dst:	A (аккумулятор A) B (аккумулятор B)

$-16 \le SHIFT \le 15$

Операнды могут быть константами или выражениями, вычисляемыми при ассемблировании, для ссылки на память, порты ввода/вывода, адреса регистров, указателей и различные другие константы. В этой секции приводится также допустимый диапазон значений операндов.

Опкод

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

Опкод раскладывается на различные битовые поля для составления каждой инструкции. Смотри таблицы 1–2, описывающие символы, применяемые в кодах инструкций.

Выполнение

1: (Smem) + (src) → src
2: (Smem) << (TS) + (src) → src
3: (Smem) << 16 + (src) → dst
4: (Smem) [<< SHIFT] + (src) → dst

Секция Выполнение описывает процедуры, имеющие место при выполнении инструкции. Для инструкций, имеющих несколько вариантов синтаксиса, нумерация вариантов опкода и выполнения соответствует нумерации синтаксиса. Смотри таблицу 1, описывающую символы, применяемые в секции Выполнение.

Статусные биты

Выполнение инструкции может зависеть от состояния полей или отдельных разрядов статусных регистров; и, в свою очередь, вызывать в ходе выполнения изменение этих полей или разрядов. Оба эти варианта описываются в данной секции

Описание

Эта секция описывает выполнение инструкций, и его влияние на состояние процессора и содержимое памяти. Обсуждаются все ограничения, накладываемые на операнды со стороны процессора или транслятора. Описание дублирует и дополняет символическую информацию, приведенную в секции Выполнение.

Слов

В этом поле указывается количество слов памяти, требуемых для хранения инструкции и расширяющего ее слова. Для инструкций, использующих одноадресный режим, приводится количество слов для каждой модификации, за исключением модификаций с длинным смещением, которые требуют одного дополнительного слова.

Циклов

В этом поле указывается количество циклов, требуемых процессору 1867ВЦ4Т, для выполнения одной инструкции с данными, расположенными в DARAM, и программным адресом из внутрикристального ПЗУ. Дополнительная детализация количества требуемых циклов при других конфигурациях памяти и в режиме повтора приводится в разделе 3 "Классы инструкций и количество циклов их выполнения".

Классы — В этом поле указывается класс инструкции для каждого синтаксиса. Смотри раздел 3 "Классы инструкций и количество циклов их выполнения" для описания каждого класса.

Пример — Код примера включен в каждую инструкцию. Приводится исходное содержимое памяти и/или регистров (до выполнения) и их именение в результате выполнения инструкции.

2 Краткое описание набора инструкций

Набор инструкций процессора 1867ВЦ4Т можно разделить на четыре основных типа операций:

- Арифметические операции
- Логические операции
- Операции управления выполнением программы
- Операции загрузки и сохранения

В данном разделе каждый тип операций в свою очередь разделен на более мелкие группы, в соответствии с выполняемыми функциями. Для каждого листинга инструкций приводится наилучшее количество слов, циклов и класс инструкции, а также соответствующий номер страницы в разделе 4 подробного описания набора инструкций. Включена информация по повтору однократных инструкций и список инструкций, не работающих в режиме повтора (неповторяемых инструкций).

2.1 Арифметические операции

В этой секции собраны инструкции арифметических операций. В таблицах с 5 по 10 приводятся инструкции, разделенные на следующие функциональные группы:

- Инструкции сложения (таблица 5).
- Инструкции вычитания (таблица 6 на странице 13).
- Инструкции умножения (таблица 7 на странице 14).
- Инструкции умножения с накоплением и умножения с вычитанием (таблица 8 на странице 14).
- Инструкции умножения с накоплением и умножения с вычитанием (продолжение таблицы 8 на странице 15).
- Двойные (с 32-разрядным операндом) инструкции (таблица 9 на странице 16).
- Инструкции для специфических применений (таблица 10 на странице 17).

Таблица 5 – Инструкции сложения

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
ADD *Smem, src*	src = src + Smem	1	1	3A, 3B	83
ADD *Smem,* TS, *src*	src = src + Smem << TS	1	1	3A, 3B	83
ADD *Smem*, 16, *src*[*, dst*]	dst = src + Smem << 16	1	1	3A, 3B	83
ADD *Smem*[*, SHIFT*], *src*[*, dst*]	dst = src + Smem << SHIFT	2	2	4A, 4B	83
ADD *Xmem, SHFT, src*	src = src + Xmem << SHFT	1	1	3A	83
ADD *Xmem, Ymem, dst*	dst = Xmem << 16 + Ymem << 16	1	1	7	83
ADD *#lk*[*, SHFT*], *src*[*, dst*]	dst = src + #lk << SHFT	2	2	2	83
ADD *#lk,* 16, *src[, dst*]	dst = src + #lk << 16	2	2	2	83
ADD *src*[*, SHIFT*][*, dst*]	dst = dst + src << SHIFT	1	1	1	83
ADD *src,* ASM[, *dst*]	dst = dst + src << ASM	1	1	1	83
ADDC *Smem, src*	src = src + Smem + C	1	1	3A, 3B	87
ADDM *#lk, Smem*	Smem = Smem + #lk	2	2	18A, 18B	88
ADDS *Smem, src*	src = src + uns(Smem)	1	1	3A, 3B	89

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM. При использовании косвенной адресации с длинным смещением или абсолютной адресации с *Smem* добавляется еще 1 слово и один цикл.

Таблица 6 – Инструкции вычитания

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
SUB *Smem, src*	src = src — Smem	1	1	3A, 3B	247
SUB *Smem,* TS, *src*	src = src — Smem << TS	1	1	3A, 3B	247
SUB Smem, 16, *src*[, *dst*]	dst = src — Smem << 16	1	1	3A, 3B	247
SUB *Smem*[*, SHIFT*], *src*[*, dst*]	dst = src — Smem << SHIFT	2	2	4A, 4B	247
SUB *Xmem, SHFT, src*	src = src — Xmem << SHFT	1	1	3A	247
SUB *Xmem, Ymem, dst*	dst = Xmem << 16 — Ymem << 16	1	1	7	247
SUB *#lk*[*, SHFT*], *src*[*, dst*]	dst = src — #lk << SHFT	2	2	2	247
SUB *#lk,* 16, *src*[*, dst*]	dst = src — #lk << 16	2	2	2	247
SUB *src*[*, SHIFT*][*, dst*]	dst = dst — src << SHIFT	1	1	1	247
SUB *src,* ASM[, *dst*]	dst = dst — src << ASM	1	1	1	247
SUBB *Smem, src*	src = src — Smem — Č	1	1	3A, 3B	251

SUBC *Smem, src*	Если (src — Smem << 15) ≥ 0
	  src = (src — Smem << 15) << 1 + 1
	Иначе
	  src = src << 1
		1	1	3A, 3B	252

SUBS *Smem, src*	src = src — uns(Smem)	1	1	3A, 3B	254

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM. При использовании косвенной адресации с длинным смещением или абсолютной адресации с *Smem* добавляется еще 1 слово и один цикл.

Таблица 7 – Инструкции умножения

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
MPY *Smem, dst*	dst = T * Smem	1	1	3A, 3B	169
MPYR *Smem, dst*	dst = rnd(T * Smem)	1	1	3A, 3B	169
MPY *Xmem, Ymem, dst*	dst = Xmem * Ymem, T = Xmem	1	1	7	169
MPY *Smem, #lk, dst*	dst = Smem * #lk, T = Smem	2	2	6A, 6B	169
MPY *#lk, dst*	dst = T * #lk	2	2	2	169
MPYA *dst*	dst = T * A(32-16)	1	1	1	171
MPYA *Smem*	B = Smem * A(32-16), T = Smem	1	1	3A, 3B	171
MPYU *Smem, dst*	dst = uns(T) * uns(Smem)	1	1	3A, 3B	173
SQUR *Smem, dst*	dst = Smem * Smem, T = Smem	1	1	3A, 3B	223
SQUR A*, dst*	dst = A(32-16) * A(32-16)	1	1	1	223

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM. При использовании косвенной адресации с длинным смещением или абсолютной адресации с *Smem* добавляется еще 1 слово и один цикл.

Таблица 8 – Инструкции умножения с накоплением и умножения с вычитанием

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
MAC *Smem, src*	src = src + T * Smem	1	1	3A, 3B	151
MAC *Xmem, Ymem, src*[, *dst*]	dst = src + Xmem * Ymem, T = Xmem	1	1	7	151
MAC *#lk,* src[*, dst*]	dst = src + T * #lk	2	2	2	151

MAC *Smem, #lk, src*[, *dst*]	dst = src + Smem * #lk,
	T = Smem
		2	2	6A, 6B	151

MACR *Smem, src*	src = rnd(src +T * Smem)	1	1	3A, 3B	151
MACR *Xmem, Ymem, src*[*, dst*]	dst = rnd(src + Xmem * Ymem), T = Xmem	1	1	7	151
MACA *Smem*[, B]	B = B + Smem * A(32-16), T = Smem	1	1	3A, 3B	154

Продолжение таблицы 8

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
MACA T, *src*[, *dst*]	dst = src + T * A(32-16)	1	1	1	154
MACAR *Smem*[, B]	B = rnd(B + Smem * A(32-16)), T = Smem	1	1	3A, 3B	154
MACAR T, *src*[, *dst*]	dst = rnd(src + T * A(32-16))	1	1	1	154
MACD *Smem, pmad, src*	src = src + Smem * pmad, T = Smem, (Smem + 1) = Smem	2	3	23A, 23B	156
MACP *Smem, pmad, src*	src = src + Smem * pmad, T = Smem	2	3	22A, 22B	158
MACSU *Xmem, Ymem, src*	src = src + uns(Xmem) * Ymem, T = Xmem	1	1	7	160
MAS *Smem, src*	src = src — T * Smem	1	1	3A, 3B	163
MASR *Smem, src*	src = rnd(src — T * Smem)	1	1	3A, 3B	163
MAS *Xmem, Ymem, src*[, *dst*]	dst = src — Xmem * Ymem, T = Xmem	1	1	7	163
MASR *Xmem, Ymem, src*[, *dst*]	dst = rnd(src — Xmem * Ymem), T = Xmem	1	1	7	163
MASA *Smem*[, B]	B = B — Smem * A(32-16), T = Smem	1	1	3A, 3B	165
MASA T, *src*[, *dst*]	dst = src — T * A(32-16)	1	1	1	165
MASAR T, *src*[, *dst*]	dst = rnd(src — T * A(32-16))	1	1	1	165
SQURA *Smem, src*	src = src + Smem * Smem, T = Smem	1	1	3A, 3B	225
SQURS *Smem, src*	src = src — Smem * Smem, T = Smem	1	1	3A, 3B	226

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM. При использовании косвенной адресации с длинным смещением или абсолютной адресации с *Smem* добавляется еще 1 слово и один цикл.

Таблица 9 – Двойные (с 32-разрядным операндом) инструкции

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
DADD *Lmem, src*[*, dst*]	Если C16 = 0 dst = Lmem + src Если C16 = 1 dst(39-16) = Lmem(31-16) + src(31-16) dst(15-0) = Lmem(15-0) + src(15-0)	1	1	9A, 9B	115
DADST *Lmem, dst*	Если C16 = 0 dst = Lmem + (T << 16 + T) Если C16 = 1 dst(39-16) = Lmem(31-16) + T dst(15-0) = Lmem(15-0) — T	1	1	9A, 9B	115
DRSUB *Lmem, src*	Если C16 = 0 src = Lmem — src Если C16 = 1 src(39-16) = Lmem(31-16) — src(31-16) src(15-0) = Lmem(15-0) — src(15-0)	1	1	9A, 9B	121
DSADT *Lmem, dst*	Если C16 = 0 dst = Lmem — (T << 16 + T) Если C16 = 1 dst(39-16) = Lmem(31-16) — T dst(15-0) = Lmem(15-0) + T	1	1	9A, 9B	123
DSUB *Lmem, src*	Если C16 = 0 src = src — Lmem Если C16 = 1 src(39-16) = src(31-16) — Lmem(31-16) src(15-0) = src(15-0) — Lmem(15-0)	1	1	9A, 9B	126
DSUBT *Lmem, dst*	Если C16 = 0 dst = Lmem — (T << 16 + T) Если C16 = 1 dst(39-16) = Lmem(31-16) — T dst(15-0) = Lmem(15-0) — T	1	1	9A, 9B	128

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM. При использовании косвенной адресации с длинным смещением или абсолютной адресации с *Lmem* добавляется еще 1 слово и один цикл.

Таблица 10 – Инструкции для специфических применений

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
ABDST *Xmem, Ymem*	B = B + |A(32-16)| A = (Xmem — Ymem) << 16	1	1	7	81
ABS *src*[*, dst*]	dst = |src|	1	1	1	82
CMPL *src*[*, dst*]	dst = ~src	1	1	1	110
DELAY *Smem*	(Smem + 1) = Smem	1	1	24A, 24B	119
EXP *src*	T = кол-во знаковых разрядов (src) — 8	1	1	1	130
FIRS *Xmem, Ymem, pmad*	B = B + A * pmad A = (Xmem + Ymem) << 16	2	3	8	131
LMS *Xmem, Ymem*	B = B + Xmem * Ymem A = A + Xmem << 16 + 2^{15}	1	1	7	149
MAX *dst*	dst = max(A, B)	1	1	1	167
MIN *dst*	dst = min(A, B)	1	1	1	168
NEG *src*[*, dst*]	dst = –src	1	1	1	185
NORM *src*[*, dst*]	dst = src << TS dst = norm(src, TS)	1	1	1	187
POLY *Smem*	B = Smem << 16 A = rnd(A(32-16) * T + B)	1	1	3A, 3B	191
RND *src*[*, dst*]	dst = src + 2^{15}	1	1	1	205
SAT *src*	насыщение(src)	1	1	1	216
SQDST *Xmem, Ymem*	B = B + A(32-16) * A(32-16) A = (Xmem — Ymem) << 16	1	1	7	222

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM. При использовании косвенной адресации с длинным смещением или абсолютной адресации с *Smem* добавляется еще 1 слово и один цикл.

2.2 Логические операции

В этой секции собраны инструкции логических операций. В таблицах с 11 по 15 приводятся инструкции, разделенные на следующие функциональные группы:

- Инструкции "И" (таблица 11).
- Инструкции "ИЛИ" (таблица 12 на странице 18).
- Инструкции "Исключающее ИЛИ" (таблица 13 на странице 19).
- Инструкции сдвига (таблица 14 на странице 19).
- Инструкции тестирования (таблица 15 на странице 19).

Таблица 11 – Инструкции "И"

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
AND *Smem, src*	src = src & Smem	1	1	3A, 3B	90
AND *#lk*[*, SHFT*], *src*[*, dst*]	dst = src & #lk << SHFT	2	2	2	90
AND *#lk*, 16, *src[, dst*]	dst = src & #lk << 16	2	2	2	90
AND *src*[*, SHIFT*][*, dst*]	dst = dst & src << SHIFT	1	1	1	90
ANDM *#lk, Smem*	Smem = Smem & #lk	2	2	18A, 18B	92

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM. При использовании косвенной адресации с длинным смещением или абсолютной адресации с *Smem* добавляется еще 1 слово и один цикл.

Таблица 12 – Инструкции "ИЛИ"

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
OR *Smem, src*	src = src | Smem	1	1	3A, 3B	188
OR *#lk*[*, SHFT*], *src*[*, dst*]	dst = src | #lk << SHFT	2	2	2	188
OR *#lk*, 16, *src[, dst*]	dst = src | #lk << 16	2	2	2	188
OR *src*[*, SHIFT*][*, dst*]	dst = dst | src << SHIFT	1	1	1	188
ORM *#lk, Smem*	Smem = Smem | #lk	2	2	18A, 18B	190

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM. При использовании косвенной адресации с длинным смещением или абсолютной адресации с *Smem* добавляется еще 1 слово и один цикл.

Таблица 13 – Инструкции "Исключающее ИЛИ"

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
XOR *Smem, src*	src = src ^ Smem	1	1	3A, 3B	260
XOR *#lk*[*, SHFT*], *src*[*, dst*]	dst = src ^ #lk << SHFT	2	2	2	260
XOR *#lk*, 16, *src*[*, dst*]	dst = src ^ #lk << 16	2	2	2	260
XOR *src*[*, SHIFT*][*, dst*]	dst = dst ^ src << SHIFT	1	1	1	260
XORM *#lk, Smem*	Smem = Smem ^ #lk	2	2	18A, 18B	262

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM. При использовании косвенной адресации с длинным смещением или абсолютной адресации с *Smem* добавляется еще 1 слово и один цикл.

Таблица 14 – Инструкции сдвига

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
ROL *src*	Вращение влево с включением переноса	1	1	1	206
ROLTC *src*	Вращение влево с включением TC-бита	1	1	1	207
ROR *src*	Вращение вправо с включением переноса	1	1	1	208
SFTA *src, SHIFT*[*, dst*]	dst = src << SHIFT {арифметический сдвиг}	1	1	1	217
SFTC *src*[*, SHIFT*][*, dst*]	если src(31) = src(30), тогда src = src << 1	1	1	1	219
SFTL *src, SHIFT*[*, dst*]	dst = src << SHIFT {логический сдвиг}	1	1	1	220

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM.

Таблица 15 – Инструкции тестирования

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
BIT *Xmem, BITC*	TC = Xmem(15 — BITC)	1	1	3A	100
BITF *Smem, #lk*	TC = (Smem && #lk)	2	2	6A, 6B	101
BITT *Smem*	TC = Smem(15 — T(3-0))	1	1	3A, 3B	102
CMPM *Smem, #lk*	TC = (Smem == #lk)	2	2	6A, 6B	111
CMPR *CC, ARx*	Сравнение ARx с AR0	1	1	1	112

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM. При использовании косвенной адресации с длинным смещением или абсолютной адресации с *Smem* добавляется еще 1 слово и один цикл.

2.3 Операции управления выполнением программы

В этой секции собраны инструкции управления выполнением программы. В таблицах с 16 по 22 приводятся инструкции, разделенные на следующие функциональные группы:

– Инструкции переходов (таблица 16).
– Инструкции вызовов (таблица 17 на странице 20).
– Инструкции прерываний (таблица 18 на странице 21).
– Инструкции возвратов (таблица 19 на странице 21).
– Инструкции повторов (таблица 20 на странице 21).
– Инструкции работы со стеком (таблица 21 на странице 22).
– Смешанные инструкции управления выполнением программы (таблица 22 на странице 22).

Таблица 16 – Инструкции переходов

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
B[D] *pmad*	PC = pmad(15-0)	2	4/[2[2)]]	29A	93
BACC[D] *src*	PC = src(15-0)	1	6/[4[2)]]	30A	94
BANZ[D] *pmad, Sind*	если (Sind ≠ 0) тогда PC = pmad(15-0)	2	4[3)]/2[4)]/ [2[2)]]	29A	95
BC[D] *pmad, cond*[*, cond*[*, cond*]]	если (условие(я)) тогда PC = pmad(15-0)	2	5[3)]/3[4)]/ [3[2)]]	31A	97

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM.
[3)] Условие выполнено.
[4)] Условие невыполнено.
[2)] Задержанная инструкция.

Таблица 17 – Инструкции вызовов

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
CALA[D] *src*	– – SP, PC + 1[3[2)]] =TOS, PC = src(15-0)	1	6/[4[2)]]	30B	103
CALL[D] *pmad*	– – SP, PC + 2[4[2)]] =TOS PC = pmad(15-0)	2	4/[2[3)]]	29B	105
CC[D] *pmad, cond*[*, cond*[*, cond*]]	если (условие(я)) тогда – – SP, PC + 2[4[2)]] = TOS, PC = pmad(15-0)	2	5[4)]/3[3)]/ [3[2)]]	31B	107

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM.
[2)] Задержанная инструкция.
[3)] Условие невыполнено.
[4)] Условие выполнено.

Таблица 18 – Инструкции прерываний

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
INTR *K*	– – SP, + +PC =TOS, PC = IPTR(15-7) + K << 2, INTM = 1	1	3	35	135
TRAP *K*	– – SP, + +PC =TOS, PC = IPTR(15-7) + K << 2	1	3	35	255

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM.

Таблица 19 – Инструкции возвратов

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
RC[D] *cond*[*, cond*[*, cond*]]	если (условие(я)) тогда PC = TOS, + + SP	1	5[2)]/3[3)]/ [3[4)]]	32	198
RET[D]	PC = TOS, + + SP	1	5/[3[4)]]	32	202
RETE[D]	PC = TOS, + + SP, INTM = 0	1	5/[3[4)]]	32	203
RETF[D]	PC = RTT, + + SP, INTM = 0	1	3/[1[4)]]	33	204

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM.
[2)] Условие выполнено.
[3)] Условие невыполнено.
[4)] Задержанная инструкция.

Таблица 20 – Инструкции повторов

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
RPT *Smem*	Повтор инструкции, RC = Smem	1	3	5A, 5B	209
RPT *#K*	Повтор инструкции, RC = #K	1	1	1	209
RPT *#lk*	Повтор инструкции, RC = #lk	2	2	2	209
RPTB[D] *pmad*	Повтор блока, RSA = PC + 2[4[2)]], REA = pmad, BRAF =1	2	4/[2[2)]]	29A	211
RPTZ *dst, #lk*	Повтор инструкции, RC = #lk, dst = 0	2	2	2	213

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM. При использовании косвенной адресации с длинным смещением или абсолютной адресации с *Smem* добавляется еще 1 слово и один цикл.
[2)] Задержанная инструкция.

Таблица 21 – Инструкции работы со стеком

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
FRAME *K*	SP = SP + K	1	1	1	132
POPD *Smem*	Smem = TOS, + + SP	1	1	17A, 17B	192
POPM *MMR*	MMR = TOS, + + SP	1	1	17A	193
PSHD *Smem*	– – SP, Smem = TOS	1	1	16A, 16B	196
PSHM *MMR*	– – SP, MMR = TOS	1	1	16A	197

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM. При использовании косвенной адресации с длинным смещением или абсолютной адресации с *Smem* добавляется еще 1 слово и один цикл.

Таблица 22 – Смешанные инструкции управления выполнением программы

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
IDLE *K*	idle(k)	1	4	36	133
MAR *Smem*	Если CMPT = 0, тогда модифицировать ARx Если CMPT = 0 и ARx ≠ AR0, тогда модифицировать ARx, ARP = x Если CMPT = 0 и ARx = AR0, тогда модифицировать AR(ARP)	1	1	1, 2	161
NOP	нет операции	1	1	1	186
RESET	программный сброс	1	3	35	201
RSBX *N, SBIT*	STN(SBIT) = 0	1	1	1	214
SSBX *N, SBIT*	STN(SBIT) = 1	1	1	1	228
XC *n, cond*[*, cond*[*, cond*]]	если (условие(я)) тогда выполнить следующие n инструкций; n = 1 или 2	1	1	1	257

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM. При использовании косвенной адресации с длинным смещением или абсолютной адресации с *Smem* добавляется еще 1 слово и один цикл.

2.4 Операции загрузки и сохранения

В этой секции собраны инструкции загрузки и сохранения. В таблицах с 23 по 30 приводятся инструкции, разделенные на следующие функциональные группы:

- Инструкции загрузки (таблица 23).
- Инструкции сохранения (таблица 24 на странице 24).
- Инструкции условного сохранения (таблица 25 на странице 25).
- Параллельные инструкции загрузки и сохранения (таблица 26 на странице 25).
- Параллельные инструкции загрузки и умножения (таблица 27 на странице 25).
- Параллельные инструкции сохранения и сложения/вычитания (таблица 28 на странице 26).
- Параллельные инструкции сохранения и умножения (таблица 29 на странице 26).
- Смешанные инструкции загрузки и сохранения (таблица 30 на странице 27).

Таблица 23 – Инструкции загрузки

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
DLD *Lmem, dst*	dst = Lmem	1	1	9A, 9B	120
LD *Smem, dst*	dst = Smem	1	1	3A, 3B	136
LD *Smem,* TS, *dst*	dst = Smem << TS	1	1	3A, 3B	136
LD *Smem*, 16*, dst*	dst = Smem << 16	1	1	3A, 3B	136
LD *Smem*[*, SHIFT*]*, dst*	dst = Smem << SHIFT	2	2	4A, 4B	136
LD *Xmem, SHFT, dst*	dst = Xmem << SHFT	1	1	3A	136
LD *#K, dst*	dst = #K	1	1	1	136
LD *#lk*[*, SHFT*]*, dst*	dst = #lk << SHFT	2	2	2	136
LD *#lk,* 16, *dst*	dst = #lk << 16	2	2	2	136
LD *src,* ASM[, *dst*]	dst = src << ASM	1	1	1	136
LD *src*[*, SHIFT*]*, dst*	dst = src << SHIFT	1	1	1	136
LD *Smem,* T	T = Smem	1	1	3A, 3B	140
LD *Smem,* DP	DP = Smem(8-0)	1	3	5A, 5B	140
LD *#k9,* DP	DP = #k9	1	1	1	140
LD *#k5,* ASM	ASM = #k5	1	1	1	140
LD *#k3,* ARP	ARP = #k3	1	1	1	140
LD *Smem,* ASM	ASM = Smem(4-0)	1	1	3A, 3B	140

Продолжение таблицы 23

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
LDM *MMR, dst*	dst = MMR	1	1	3A	142
LDR *Smem,* dst	dst = rnd(Smem)	1	1	3A, 3B	147
LDU *Smem,* dst	dst = uns(Smem)	1	1	3A, 3B	148
LTD *Smem*	T = Smem, (Smem +1) = Smem	1	1	24A, 24B	150

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM. При использовании косвенной адресации с длинным смещением или абсолютной адресации с *Lmem* или *Smem* добавляется еще 1 слово и один цикл.

Таблица 24 – Инструкции сохранения

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
DST *src, Lmem*	Lmem = src	1	2	13A, 13B	125
ST T, *Smem*	Smem = T	1	1	10A, 10B	229
ST TRN, *Smem*	Smem = TRN	1	1	10A, 10B	229
ST *#lk, Smem*	Smem = #lk	2	2	12A, 12B	229
STH *src, Smem*	Smem = src << — 16	1	1	10A, 10B	231
STH *src*, ASM, *Smem*	Smem = src << (ASM — 16)	1	1	10A, 10B	231
STH *src, SHFT, Xmem*	Xmem = src << (SHFT — 16)	1	1	10A	231
STH *src*[*, SHIFT*], *Smem*	Smem = src << (SHIFT — 16)	2	2	11A, 11B	231
STL *src, Smem*	Smem = src	1	1	10A, 10B	233
STL *src,* ASM, *Smem*	Smem = src << ASM	1	1	10A, 10B	233
STL *src, SHFT, Xmem*	Xmem = src << SHFT	1	1	10A, 10B	233
STL *src*[*, SHIFT*], *Smem*	Smem = src << SHIFT	2	2	11A, 11B	233
STLM *src, MMR*	MMR = src	1	1	10A	235
STM *#lk, MMR*	MMR = #lk	2	2	12A	236

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM. При использовании косвенной адресации с длинным смещением или абсолютной адресации с *Lmem* или *Smem* добавляется еще 1 слово и один цикл.

Таблица 25 – Инструкции условного сохранения

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
CMPS *src, Smem*	Если = src(31-16) > src(15-0) тогда Smem = src(31-16) Если = src(31-16) ≤ src(15-0) тогда Smem = src(15-0)	1	1	10A, 10B	113
SACCD *src, Xmem, cond*	Если (условие) Xmem = src << (ASM — 16)	1	1	15	215
SRCCD *Xmem, cond*	Если (условие) Xmem = BRC	1	1	15	227
STRCD *Xmem, cond*	Если (условие) Xmem = T	1	1	15	246

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM. При использовании косвенной адресации с длинным смещением или абсолютной адресации с *Smem* добавляется еще 1 слово и один цикл.

Таблица 26 – Параллельные инструкции загрузки и сохранения

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
ST *src, Ymem* || LD *Xmem, dst*	Ymem = src << (ASM — 16) || dst = Xmem << 16	1	1	14	238
ST *src, Ymem* || LD *Xmem,* T	Ymem = src << (ASM — 16) || T = Xmem	1	1	14	238

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM.

Таблица 27 – Параллельные инструкции загрузки и умножения

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
LD *Xmem, dst* || MAC *Ymem, dst_*	dst = Xmem << 16 || dst_ = dst_ + T * Ymem	1	1	7	143
LD *Xmem, dst* || MACR *Ymem, dst_*	dst = Xmem << 16 || dst_ = rnd(dst_ + T * Ymem)	1	1	7	143
LD *Xmem, dst* || MAS *Ymem, dst_*	dst = Xmem << 16 || dst_ = dst_ — T * Ymem	1	1	7	145
LD *Xmem, dst* || MASR *Ymem, dst_*	dst = Xmem << 16 || dst_ = rnd(dst_ — T * Ymem)	1	1	7	145

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM.

Таблица 28 – Параллельные инструкции сохранения и сложения/вычитания

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
ST *src, Ymem* || ADD *Xmem, dst*	Ymem = src << (ASM — 16) || dst = dst_ + Xmem << 16	1	1	14	237
ST *src, Ymem* || SUB *Xmem, dst*	Ymem = src << (ASM — 16) || T = (Xmem << 16) — dst_	1	1	14	245

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM.

Таблица 29 – Параллельные инструкции сохранения и умножения

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
ST *src, Ymem* || MAC *Xmem, dst*	Ymem = src << (ASM — 16) || dst = dst + T * Xmem	1	1	14	240
ST *src, Ymem* || MACR *Xmem, dst*	Ymem = src << (ASM — 16) || dst = rnd(dst + T * Xmem)	1	1	14	240
ST *src, Ymem* || MAS *Xmem, dst*	Ymem = src << (ASM — 16) || dst = dst — T * Xmem	1	1	14	242
ST *src, Ymem* || MASR *Xmem, dst*	Ymem = src << (ASM — 16) || dst = rnd(dst — T * Xmem)	1	1	14	242
ST *src, Ymem* || MPY *Xmem, dst*	Ymem = src << (ASM — 16) || dst = T * Xmem	1	1	14	244

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM.

Таблица 30 – Смешанные инструкции загрузки и сохранения

Синтаксис	Выражение	Сл.[1)]	Цкл.[1)]	Класс	Стр.
MVDD *Xmem, Ymem*	Ymem = Xmem	1	1	14	174
MVDK *Smem, dmad*	dmad = Smem	2	2	19A, 19B	175
MVDM *dmad, MMR*	MMR = dmad	2	2	19A	177
MVDP *Smem, pmad*	pmad = Smem	2	4	20A, 20B	178
MVKD *dmad, Smem*	Smem = dmad	2	2	19A, 19B	179
MVMD *MMR, dmad*	dmad = MMR	2	2	19A	181
MVMM *MMRx, MMRy*	MMRy = MMRx	1	1	1	182
MVPD *pmad, Smem*	Smem = pmad	2	3	21A, 21B	183
PORTR *PA, Smem*	Smem = PA	2	2	27A, 27B	194
PORTW *Smem, PA*	PA = Smem	2	2	28A, 28B	195
READA *Smem*	Smem = A	1	5	25A, 25B	200
WRITA *Smem*	A = Smem	1	5	26A, 26B	256

[1)] Значения в столбцах Сл. (количество слов) и Цкл. (количество циклов) приведены для случая, когда используется DARAM. При использовании косвенной адресации с длинным смещением или абсолютной адресации с *Smem* добавляется еще 1 слово и один цикл.

2.5 Повтор однократной инструкции

В процессоре 1867ВЦ4Т предусмотрены инструкции повтора. Инструкция, следующая за инструкцией повтора с операндом N, будет выполнена N + 1 раз. Это значение сохраняется в 16-разрядном регистре повтора (RC). Регистр RC программно не доступен. Он загружается только инструкциями повтора. Однократная инструкция может быть выполнена максимально 65536 раз. При повторе инструкции происходит автоматическая инкрементация абсолютного адреса программы или данных.

При активизации повтора инструкции все прерывания, включая NMI# (но не RS#!) запрещаются до окончания цикла повтора. При этом процессор реагирует на сигнал HOLD# во время цикла повтора – реакция зависит от состояния HM бита в регистре ST1.

Функция повтора может применяться к различным инструкциям, таким как умножение с накоплением и блочное перемещение данных, чтобы увеличить скорость выполнения этих инструкций. Эти инструкции, приведенные в таблице 31, являются многоцикловыми при обычном однократном выполнении, но в режиме повтора становятся эффективно одноцикловыми после выполнения первой итерации.

Таблица 31 – Многоцикловые инструкции, становящиеся одноцикловыми в режиме повтора

Инструкция	Описание	#Цкл.[1)]
FIRS	Симметричный КИХ-фильтр	3
MACD	Умножить и поместить результат в аккумулятор с задержкой	3
MACP	Умножить и поместить результат в аккумулятор	3
MVDK	Перемещение внутри памяти данных	2
MVDM	Перемещение из памяти данных в MMR	2
MVDP	Перемещение из памяти данных в программную память	4
MVKD	Перемещение внутри памяти данных	2
MVMD	Перемещение из MMR в память данных	2
MVPD	Перемещение из программной памяти в память данных	3
READA	Чтение из программной памяти в память данных	5
WRITA	Запись из памяти данных в программную память	5

[1)] Количество циклов в режиме однократного выполнения.

Инструкции с единственным операндом памяти данных не могут выполняться в режиме повтора, если в них используется модификатор длинного смещения или абсолютная адресация (например: *ARn(lk), *+ARn(lk), *+ARn(lk)% и *(lk)). Инструкции, приведенные в таблице 32 , не могут выполняться в режиме повтора с использованием RPT или RPTZ.

Таблица 32 – Неповторяемые инструкции

Инструкция	Описание
ADDM	Прибавить длинную константу к памяти данных
ANDM	Операция "И" памяти данных с длинной константой
B[D]	Безусловный переход
BACC[D]	Переход по адресу в аккумуляторе
BANZ[D]	Переход, если содержимое вспомогательного регистра не равно
BC[D]	Условный переход
CALA[D]	Вызов с адресом из аккумулятора
CALL[D]	Безусловный вызов
CC[D]	Условный вызов
CMPR	Сравнение вспомогательных регистров
DST	Сохранение длинного (32 разряда) слова
IDLE	Инструкция IDLE
INTR	Инструкция прерывания
LD ARP	Загрузка указателя вспомогательных регистров (ARP)
LD DP	Загрузка указателя страницы памяти данных (DP)
MVMM	Перемещение из MMR в MMR
ORM	Операция "ИЛИ" памяти данных с длинной константой
RC[D]	Условный возврат
RESET	Программный сброс
RET[D]	Безусловный возврат
RETE[D]	Возврат из прерывания
RETF[D]	Быстрый возврат из прерывания
RND	Округление аккумулятора
RPT	Повтор следующей инструкции
RPTB[D]	Повтор блока
RPTZ	Повтор следующей инструкции с очисткой аккумулятора
RSBX	Сброс разряда статусного регистра
SSBX	Установка разряда статусного регистра
TRAP	Программное прерывание
XC	Условное выполнение
XORM	Операция "Исключающее ИЛИ" памяти данных с длинной константой

3 Классы инструкций и количество циклов их выполнения

В этом разделе инструкции процессора 1867ВЦ4Т классифицированы по нескольким категориям или классам в зависимости от количества циклов, требуемых для их выполнения. Поскольку одна и та же инструкция может иметь множественный синтаксис и разные варианты выполнения, она может быть одновременно представлена в разных классах.

В таблицах, приведенных в этом разделе, указывается количество циклов, необходимых процессору 1867ВЦ4Т для выполнения инструкции, в режиме однократного выполнения и в режиме повтора, для различных вариантов конфигурации памяти. Рассматривается также однократный доступ к памяти данных с использованием длинной константы. Заголовки столбцов в таблицах указывают на источник программного кода. Программный код может выполняться из:

ROM – внутреннего ПЗУ программ;

DARAM – внутреннего ОЗУ двойного доступа;

External – внешней программной памяти.

Если тот или иной класс инструкций предусматривает использование операнда(ов) памяти данных, в строках таблиц указывается расположение операнда(ов). Операнды могут распологаться в:

DARAM – внутреннем ОЗУ двойного доступа;

ROM – внутреннем ПЗУ программ;

External – внешней памяти;

MMR – картированном в память регистре.

Под количеством циклов, требуемых для выполнения инструкции, понимается количество машинных циклов процессора (период сигнала CLKOUT). Дополнительные состояния ожидания при доступе к памяти программ/данных и к пространству ввода/вывода определяются следующим образом:

d – состояние ожидания памяти данных — количество дополнительных циклов, необходимых процессору, для доступа к внешней памяти данных;

io – состояние ожидания пространства ввода/вывода — количество дополнительных циклов, необходимых процессору, для доступа к внешним устройствам ввода/вывода;

n – повторение — количество повторов выполнения инструкции;

nd – состояние ожидания памяти данных, повторенное n раз;

np – состояние ожидания памяти программ, повторенное n раз;

npd – состояние ожидания памяти программ, повторенное n раз;

p – состояние ожидания памяти программ — количество дополнительных циклов, необходимых процессору, для доступа к внешней памяти программ;

pd – состояние ожидания памяти программ — количество дополнительных циклов, необходимых процессору, для доступа к операнду из внешней памяти программ.

Эти переменные могут также использовать надписи src, dst и code, чтобы указать на источник, приемник и код соответственно.

При любом чтении из внешней памяти требуется, по меньшей мере, один цикл инструкции; при любой записи во внешнюю память требуется, по меньшей мере, два цикла инструкций для завершения операции. Доступ к внешней памяти может удлиняться, если добавляются дополнительные циклы состояний ожидания за счет использования программно-управляемого генератора состояний ожидания или внешнего входа READY. Однако, внутри процессора всякая запись во внешнюю память занимает только один цикл, до тех пор, пока нет других доступов во внешнюю память в то же самое время. Это возможно, потому что конвейер инструкций требует только одного цикла для запроса доступа на запись во внешнюю память, а блок интерфейса внешней шины завершает доступы на запись независимо.

Циклы инструкций базируются на следующих допущениях:

– По меньшей мере, пять инструкций, следующих за текущей инструкцией, выбираются из той же самой секции памяти (внутренней или внешней), что и текущая инструкция, кроме инструкций, вызывающих нарушение последовательного увеличения программного счетчика, таких как переходы или вызовы.

– При выполнении однократной инструкции нет конфликтов шины или конвейера между текущей инструкцией и любыми другими инструкциями в конвейере. Рассматривается только исключение, когда есть конфликт между выборкой инструкции и доступом записи/чтения к памяти (если это имеет место).

– Для режима повтора однократной инструкции рассматриваются все конфликты, вызванные конвейерным выполнением этой инструкции.

Класс 1 1 слово, 1 цикл. Нет операнда или короткий непосредственный или регистровый операнд и отсутствие операндов памяти.

Мнемоники

ABS	MACA[R]	NORM	SFTA
ADD	MAR	OR	SFTC
AND	MASA[R]	RND	SFTL
CMPL	MAX	ROL	SQUR
CMPR	MIN	ROLTC	SSBX
EXP	MPYA	ROR	SUB
FRAME	MVMM	RPT	XC
LD	NEG	RSBX	XOR
LD T/DP/ASM/ARP	NOP	SAT	

Циклов

Циклов при однократном выполнении

Программа		
ROM	DARAM	External
1	1	1 + p

Циклов при выполнении в режиме повтора

Программа		
ROM	DARAM	External
n	n	n + p

Класс 2 2 слова, 2 цикла. Длинный непосредственный операнд и отсутствие операндов памяти.

Мнемоники

ADD	MAC	OR	SUB
AND	MAR	RPT	XOR
LD	MPY	RPTZ	

Циклов

Циклов при однократном выполнении

Программа		
ROM	DARAM	External
2	2	2 + 2p

Циклов при выполнении в режиме повтора

Программа		
ROM	DARAM	External
n + 1	n + 1	n + 1 + 2p

Класс 3A 1 слово, 1 цикл. Единственный операнд чтения памяти данных (Smem или Xmem) или операнд чтения MMR.

Мнемоники

ADD	LDM	MPYA	SUBB
ADDC	LDR	MPYU	SUBC
ADDS	LDU	OR	SUBS
AND	MAC[R]	POLY	XOR
BIT	MACA[R]	SQUR	
BITT	MAS[R]	SQURA	
LD	MASA	SQURS	
LD T/DP/ASM/ARP	MPY[R]	SUB	

Циклов

Циклов при однократном выполнении

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	1	1, 2[1)]	1 + p
External	1 + d	1 + d	2 + d + p
MMR[2)]	1	1	1 + p

[1)] Операнд и программный код в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Циклов при выполнении в режиме повтора

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	n	n, n + 1[1)]	n + p
External	n + nd	n + nd	n + 1 + nd + p
MMR[2)]	n	n	n + p

[1)] Операнд и программный код в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 3B 2 слова, 2 цикла. Единственный операнд чтения памяти данных (Smem), использующий косвенную адресацию с длинным смещением.

Мнемоники

ADD	LDU	OR	SUBS
ADDC	MAC[R]	POLY	XOR
ADDS	MACA[R]	SQUR	
AND	MAS[R]	SQURA	
BITT	MASA	SQURS	
LD	MPY[R]	SUB	
LD T/DP/ASM/ARP	MPYA	SUBB	
LDR	MPYU	SUBC	

Циклов

Циклов для однократного выполнения с длинным смещением

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	2	2, 3[1)]	2 + 2p
External	2 + d	2 + d	3 + d + 2p
MMR[2)]	2	2	2 + 2p

[1)] Операнд и программный код в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 4A 2 слова, 2 цикла. Единственный операнд чтения памяти данных (Smem).

Мнемоники ADD LD SUB

Циклов

Циклов при однократном выполнении

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	2	2, 3[1)]	2 + 2p
External	2 + d	2 + d	3 + d + 2p
MMR[2)]	2	2	2 + 2p

[1)] Операнд и программный код в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Циклов при выполнении в режиме повтора

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	n + 1	n + 1, n + 2[1)]	n + 1 + 2p
External	n + 1 + nd	n + 1 + nd	n + 2 + nd + 2p
MMR[2)]	n + 1	n + 1	n + 1 + 2p

[1)] Операнд и программный код в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 4B 3 слова, 3 цикла. Единственный операнд чтения памяти данных (Smem), использующий косвенную адресацию с длинным смещением.

Мнемоники ADD LD SUB

Циклов

Циклов для однократного выполнения с длинным смещением

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	3	3, 4[1)]	3 + 3p
External	3 + d	3 + d	4 + d + 3p
MMR[2)]	3	3	3 + 3p

[1)] Операнд и программный код в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 5A 1 слово, 3 цикла. Единственный операнд (Smem) чтения памяти данных (с DP в качестве приемника в инструкции загрузки).

Мнемоники LD RPT

Циклов

Циклов при однократном выполнении

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	3	3	3 + p
External	3 + d	3 + d	3 + d + p
MMR[1)]	3	3	3 + p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 5B 2 слова, 4 цикла. Единственный операнд чтения памяти данных (Smem), использующий косвенную адресацию с длинным смещением (с DP в качестве приемника в инструкции загрузки).

Мнемоники LD RPT

Циклов

Циклов для однократного выполнения с длинным смещением

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	4	4	4 + 2p
External	4 + d	4 + d	4 + d + 2p
MMR[1)]	4	4	4 + 2p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 6A 2 слова, 2 цикла. Единственный операнд чтения памяти данных (Smem) и единственный длинный непосредственный операнд.

Мнемоники BITF CMPM MAC MPY

Циклов

Циклов при однократном выполнении

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	2	2, 3[1)]	2 + 2p
External	2 + d	2 + d	3 + d + 2p
MMR[2)]	2	2	2 + 2p

[1)] Операнд и программный код в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Циклов при выполнении в режиме повтора

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	n + 1	n + 1, n + 2[1)]	n + 1 + 2p
External	n + 1 + nd	n + 1 + nd	n + 2 + nd + 2p
MMR[2)]	n + 1	n + 1	n + 1 + 2p

[1)] Операнд и программный код в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 6B 3 слова, 3 цикла. Единственный операнд чтения памяти данных (Smem), использующий косвенную адресацию с длинным смещением и единственный длинный непосредственный операнд.

Мнемоники BITF CMPM MAC MPY

Циклов

Циклов для однократного выполнения с длинным смещением

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	3	3, 4[1)]	3 + 3p
External	3 + d	3 + d	4 + d + 3p
MMR[2)]	3	3	3 + 3p

[1)] Операнд и программный код в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 7 1 слово, 1 цикл. Двойной операнд чтения памяти данных (Xmem и Ymem).

Мнемоники				
	ABDST	LD||MAS[R]	MACSU	SQDST
	ADD	LMS	MAS[R]	SUB
	LD||MAC[R]	MAC[R]	MPY	

Циклов

Циклов при однократном выполнении

Операнд		Программа		
Xmem	Ymem	ROM	DARAM	External
DARAM	DARAM	1	1, 2[1)]	1 + p
	External	1 + d	1 + d, 2[2)]	2 + d + p
External	DARAM	1 + d	1 + d	2 + d + p
	External	2 + 2d	2 + 2d	3 + 2d + p
MMR[3)]	DARAM	1	1	1 + p
	External	1 + d	1 + d	2 + d + p

[1)] Операнд и программный код в одном блоке памяти.
[2)] Операнд и программный код в одном блоке памяти.
[3)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Циклов при выполнении в режиме повтора

Операнд		Программа		
Xmem	Ymem	ROM	DARAM	External
DARAM	DARAM	n	n, n + 1[1)]	n + p
	External	n + nd	n + nd, 1 + n[2)]	n + 1 + nd + p
External	DARAM	n + nd	n + nd	n + 1 + nd + p
	External	2n + 2nd	2n + 2nd	2n + 1 + 2nd + p
MMR[3)]	DARAM	n	n	n + p
	External	n + nd	n + nd	n + 1 + nd + p

[1)] Операнд и программный код в одном блоке памяти.
[2)] Операнд и программный код в одном блоке памяти.
[3)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 8 2 слова, 3 цикла. Двойной операнд чтения памяти данных (Xmem и Ymem) и один операнд в программной памяти (pmad).

Мнемоники FIRS

Циклов

Циклов при однократном выполнении

Операнд			Программа		
Pmad	Xmem	Ymem	ROM	DARAM	External
DARAM	DARAM	DARAM	3, 4[1)]	3, 4[1)]	3 + 2p, 4 + 2p[1)]
		External	3 + d, 4 + d[1)]	3 + d, 4 + d[1)]	3 + d + 2p, 4 + d + 2p[1)]
	External	DARAM	3 + d	3 + d	3 + d + 2p
		External	4 + 2d	4 + 2d	4 + 2d + 2p
External	DARAM	DARAM	3 + pd	3 + pd	3 + pd + 2p
		External	4 + pd + d	4 + pd + d	4 + pd + d + 2p
	External	DARAM	4 + pd + d	4 + pd + d	4 + pd + d + 2p
		External	5 + pd + 2d	5 + pd + 2d	5 + pd + 2d + 2p

[1)] Xmem и pmad в одном блоке памяти.

Циклов при выполнении в режиме повтора

Операнд			Программа		
Pmad	Xmem	Ymem	ROM	DARAM	External
DARAM	DARAM	DARAM	n + 2, 2n + 2[1)]	n + 2, 2n + 2[1)]	n + 2 + 2p, 2n + 2 + 2p[1)]
		External	n + 2 + nd, 2n + 2 + nd [1)]	n + 2 + nd, 2n + 2 + nd [1)]	n + 2 + nd + 2p, 2n + 2 + nd + 2p[1)]
	External	DARAM	n + 2 + nd	n + 2 + nd	n + 2 + nd + 2p
		External	2n + 2 + 2nd	2n + 2 + 2nd	2n + 2 + 2nd + 2p
External	DARAM	DARAM	n + 2 + npd	n + 2 + npd	n + 2 + npd + 2p
		External	2n + 2 + npd + nd	2n + 2 + npd + nd	2n + 2 + npd + + nd + 2p
	External	DARAM	2n + 2 + npd + nd	2n + 2 + npd + nd	2n + 2 + npd + + nd + 2p
		External	3n + 2 + npd + + 2nd	3n + 2 + npd + + 2nd	3n + 2 + npd + + 2nd + 2p

[1)] Xmem и pmad в одном блоке памяти.

Класс 9A 1 слово, 1 цикл. Единственный длиннословный операнд чтения памяти данных (Lmem).

Мнемоники

DADD	DLD	DSADT	DSUBT
DADST	DRSUB	DSUB	

Циклов

Циклов при однократном выполнении

Операнд	Программа		
Lmem	ROM	DARAM	External
DARAM	1	1, 2[1)]	1 + p
External	2 + 2d	2 + 2d	3 + 2d + p

[1)] Операнд и программный код в одном блоке памяти.

Циклов при выполнении в режиме повтора

Операнд	Программа		
Lmem	ROM	DARAM	External
DARAM	n	n, n + 1[1)]	n + p
External	2n + 2nd	2n + 2nd	1 + 2n + 2nd + p

[1)] Операнд и программный код в одном блоке памяти.

Класс 9B 2 слова, 2 цикла. Единственный длиннословный операнд чтения памяти данных (Lmem), использующий косвенную адресацию с длинным смещением.

Мнемоники

DADD	DLD	DSADT	DSUBT
DADST	DRSUB	DSUB	

Циклов

Циклов для однократного выполнения с длинным смещением

Операнд	Программа		
Lmem	ROM	DARAM	External
DARAM	2	2, 3[1)]	2 + 2p
External	3 + 2d	3 + 2d	4 + 2d + 2p

[1)] Операнд и программный код в одном блоке памяти.

Класс 10A 1 слово, 1 цикл. Единственный операнд записи памяти данных (Smem или Xmem) или единственный операнд записи в MMR.

Мнемоники CMPS STH STLM
ST STL

Циклов

Циклов при однократном выполнении

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	1	1	1 + p
External	1	1	4 + d + p
MMR[1)]	1	1	1 + p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Циклов при выполнении в режиме повтора

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	n	n	n + p
External	2n – 1 + (n – 1)d	2n – 1 + (n – 1)d	2n + 2 + nd + p
MMR[1)]	n	n	n + p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 10B 2 слова, 2 цикла. Единственный операнд записи памяти данных (Smem или Xmem), использующий косвенную адресацию с длинным смещением.

Мнемоники CMPS ST STH STL

Циклов

Циклов для однократного выполнения с длинным смещением

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	2	2	2 + 2p
External	2	2	5 + d + 2p
MMR[1)]	2	2	2 + 2p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 11A 2 слова, 2 цикла. Единственный операнд записи памяти данных (Smem).

Мнемоники STH STL

Циклов

Циклов при однократном выполнении

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	2	2	2 + 2p
External	2	2	5 + d + 2p
MMR[1)]	2	2	2 + 2p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Циклов при выполнении в режиме повтора

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	n + 1	n + 1	n + 1 + 2p
External	2n + (n – 1)d	2n + (n – 1)d	2n + 3 + nd + 2p
MMR[1)]	n + 1	n + 1	n + 1 + 2p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 11B 3 слова, 3 цикла. Единственный операнд записи памяти данных (Smem), использующий косвенную адресацию с длинным смещением.

Мнемоники STH STL

Циклов

Циклов для однократного выполнения с длинным смещением

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	3	3	3 + 3p
External	3	3	6 + d + 3p
MMR[1)]	3	3	3 + 3p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 12A 2 слова, 2 цикла. Единственный операнд записи памяти данных (Smem) или единственный операнд записи в MMR.

Мнемоники ST STM

Циклов

Циклов при однократном выполнении

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	2	2	2 + 2p
External	2	2	5 + d + 2p
MMR[1)]	2	2	2 + 2p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Циклов при выполнении в режиме повтора

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	2n	2n	2n + 2p
External	2n + (n – 1)d	2n + (n – 1)d	2n + 3 + nd + p
MMR[1)]	2n	2n	2n + 2p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 12B 3 слова, 3 цикла. Единственный операнд записи памяти данных (Smem), использующий косвенную адресацию с длинным смещением.

Мнемоники ST

Циклов

Циклов для однократного выполнения с длинным смещением

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	3	3	3 + 3p
External	3	3	6 + d + 3p
MMR[1)]	3	3	3 + 3p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 13A 1 слово, 2 цикла. Единственный длиннословный операнд записи памяти данных (Lmem).

Мнемоники DST

Циклов

Циклов при однократном выполнении

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	2	2	2 + p
External	3 + d	3 + d	8 + 2d + p
MMR[1)]	2	2	2 + p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Циклов при выполнении в режиме повтора

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	2n	2n	2n + p
External	4n – 1+ (2n – 1)d	4n – 1+ (2n – 1)d	4n + 4 + 2nd + p
MMR[1)]	2n	2n	2n + p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 13B 2 слова, 3 цикла. Единственный длиннословный операнд записи памяти данных (Lmem), использующий косвенную адресацию с длинным смещением.

Мнемоники DST

Циклов

Циклов для однократного выполнения с длинным смещением

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	3	3	3 + 2p
External	4 + d	4 + d	9 + 2d + 2p
MMR[1)]	3	3	3 + 2p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 14 1 слово, 1 цикл. Двойной операнд чтения и записи памяти данных (Xmem и Ymem).

Мнемоники MVDD ST||LD ST||MAS[R] ST||SUB
ST||ADD ST||MAC[R] ST||MPY

Циклов

Циклов при однократном выполнении

Операнд		Программа		
Xmem	Ymem	ROM	DARAM	External
DARAM	DARAM	1	1, 2[1)]	1 + p
	External	1	1, 2[1)]	4 + d + p
External	DARAM	1 + d	1 + d	2 + d + p
	External	1 + d	1 + d	5 + 2d + p
MMR[2)]	DARAM	1	1, 2[1)]	1 + p
	External	1	1	4 + d + p

[1)] Операнд и программный код в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Циклов при выполнении в режиме повтора

Операнд		Программа		
Xmem	Ymem	ROM	DARAM	External
DARAM	DARAM	n	n, n + 1[1)]	n + p
	External	2n – 1 + (n – 1)d	2n – 1 + (n – 1)d, 2n + (n – 1)d[1)]	2n + 2 + nd + p
External	DARAM	n + nd	n + nd	n + 1 + nd + p
	External	4n – 3 + (2n – 1)d	4n – 3 + (2n – 1)d	4n + 1 + 2nd + p
MMR[2)]	DARAM	n	n, 2n[1)]	n + p
	External	2n – 1 + (n – 1)d	2n – 1 + (n – 1)d	2n + 2 + nd + p

[1)] Операнд и программный код в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 15 1 слово, 1 цикл. Единственный операнд записи памяти данных (Xmem).

Мнемоники SACCD SRCCD STRCD

Циклов

Циклов при однократном выполнении

Операнд	Программа		
Xmem	ROM	DARAM	External
DARAM	1	1	1 + p
External	1	1	4 + d + p
MMR[1)]	1	1	1 + p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Циклов при выполнении в режиме повтора

Операнд	Программа		
Xmem	ROM	DARAM	External
DARAM	n	n	n + p
External	2n – 1+ (n – 1)d	2n – 1+ (n – 1)d	2n + 2 + nd + p
MMR[1)]	n	n	n + p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 16A 1 слово, 1 цикл. Единственный операнд чтения памяти данных (Smem) или чтения MMR и операнд записи в стековую память (Stack).

Мнемоники PSHD PSHM

Циклов

Циклов при однократном выполнении

Операнд		Программа		
Smem	Stack	ROM	DARAM	External
DARAM	DARAM	1	1, 2[1)]	1 + p
	External	1	1, 2[1)]	4 + d + p
External	DARAM	1 + d	1 + d	2 + d + p
	External	1 + d	1 + d	5 + 2d + p
MMR[2)]	DARAM	1	1, 2[1)]	1 + p
	External	1	1	4 + d + p

[1)] Операнд и программный код в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Циклов при выполнении в режиме повтора

Операнд		Программа		
Smem	Stack	ROM	DARAM	External
DARAM	DARAM	n	n, n + 1[1)]	n + p
	External	2n – 1 + (n – 1)d	2n – 1 + (n – 1)d, 2n + (n – 1)d[1)]	2n + 2 + nd + p
External	DARAM	n + nd	n + nd	n + 1 + nd + p
	External	4n – 3 + (2n – 1)d	4n – 3 + (2n – 1)d	4n + 1 + 2nd + p
MMR[2)]	DARAM	n	n, 2n[1)]	n + p
	External	2n – 1 + (n – 1)d	2n – 1 + (n – 1)d	2n + 2 + nd + p

[1)] Операнд и программный код в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 16B 2 слова, 2 цикла. Единственный операнд чтения памяти данных (Smem), использующий косвенную адресацию с длинным смещением, и операнд записи в стековую память (Stack).

Мнемоники PSHD

Циклов

Циклов для однократного выполнения с длинным смещением

Операнд		Программа		
Smem	Stack	ROM	DARAM	External
DARAM	DARAM	2	2, 3[1)]	2 + 2p
	External	2	2, 3[1)]	5 + d + 2p
External	DARAM	2 + d	2 + d	3 + d + 2p
	External	2 + d	2 + d	6 + 2d + 2p
MMR[2)]	DARAM	2	2, 3[1)]	2 + 2p
	External	2	1	5 + d + 2p

[1)] Операнд и программный код в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 17A 1 слово, 1 цикл. Единственный операнд записи в память данных (Smem) или записи в MMR и операнд чтения стековой памяти (Stack).

Мнемоники POPD POPM

Циклов

Циклов при однократном выполнении

Операнд		Программа		
Smem	Stack	ROM	DARAM	External
DARAM	DARAM	1	1, 2[1)]	1 + p
	External	1 + d	1 + d	2 + d + p
	MMR[2)]	1	1, 2[1)]	1 + p
External	DARAM	1	1, 2[1)]	4 + d + p
	External	1 + d	1 + d	5 + 2d + p
	MMR[2)]	1	1	4 + d + p

[1)] Операнд и программный код в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Циклов при выполнении в режиме повтора

Операнд		Программа		
Smem	Stack	ROM	DARAM	External
DARAM	DARAM	n	n, n + 1[1)]	n + p
	External	n + nd	n + nd	n + 1 + nd + p
	MMR[2)]	n	n, 2n[1)]	n + p
External	DARAM	2n – 1 + (n – 1)d	2n – 1 + (n – 1)d, 2n + (n – 1)d[1)]	2n + 2 + nd + p
	External	4n – 3 + (2n – 1)d	4n – 3 + (2n – 1)d	4n + 1 + 2nd + p
	MMR[2)]	2n – 1 + (n – 1)d	2n – 1 + (n – 1)d	2n + 2 + nd + p

[1)] Операнд и программный код в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 17B 2 слова, 2 цикла. Единственный операнд записи в память данных (Smem), использующий косвенную адресацию с длинным смещением, и операнд чтения стековой памяти (Stack).

Мнемоники POPD

Циклов

Циклов для однократного выполнения с длинным смещением

Операнд		Программа		
Smem	Stack	ROM	DARAM	External
DARAM	DARAM	2	2, 3[1)]	2 + 2p
	External	2 + d	2 + d	3 + d + 2p
	MMR[2)]	2	2, 3[1)]	2 + 2p
External	DARAM	2	2, 3[1)]	5 + d + 2p
	External	2 + d	2 + d	6 + 2d + 2p
	MMR[2)]	2	2	5 + d + 2p

[1)] Операнд и программный код в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 18A 2 слова, 2 цикла. Единственный операнд (Smem) чтения и записи памяти данных.

Мнемоники ADDM ANDM ORM XORM

Циклов

Циклов при однократном выполнении

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	2	2, 3[1)]	2 + 2p
External	2 + d	2 + d	6 + 2d + 2p
MMR[2)]	2	2	2 + 2p

[1)] Операнд и программный код в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 18B 3 слова, 3 цикла. Единственный операнд (Smem) чтения и записи памяти данных, использующий косвенную адресацию с длинным смещением.

Мнемоники ADDM ANDM ORM XORM

Циклов

Циклов для однократного выполнения с длинным смещением

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	3	3, 4[1)]	3 + 3p
External	3 + d	3 + d	7 + 2d + 3p
MMR[2)]	3	3	3 + 3p

[1)] Операнд и программный код в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 19A 2 слова, 2 цикла. Единственный операнд чтения памяти данных (Smem) или чтения MMR и единственный операнд записи в память данных (dmad); или единственный операнд чтения памяти данных (dmad) и единственный операнд записи в память данных (Smem) или записи в MMR.

Мнемоники MVDK MVDM MVKD MVMD

Циклов

Циклов при однократном выполнении

Операнд		Программа		
Smem	dmad	ROM	DARAM	External
DARAM	DARAM	2	2, 3[1)]	2 + 2p
	External	2	2, 3[1)]	5 + d + 2p
	MMR[2)]	2	2	2 + 2p
External	DARAM	2 + d	2 + d	3 + d + 2p
	External	2 + d	2 + d	6 + 2d + p
	MMR[2)]	2 + d	2 + d	3 + d + 2p
MMR[2)]	DARAM	2	2, 3[1)]	2 + 2p
	External	2	2	5 + d + 2p
	MMR[2)]	2	2	2 + 2p

[1)] Операнд и программный код в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Циклов при выполнении в режиме повтора

Операнд		Программа		
Smem	dmad	ROM	DARAM	External
DARAM	DARAM	n + 1	n + 1, n + 2[1)]	n + 1 + 2p
	External	2n + (n – 1)d	2n + (n – 1)d, 2n + 1 + (n – 1)d[1)]	2n + 3 + nd + 2p
	MMR[2)]	n + 1	n + 1	n + 1 + 2p
External	DARAM	n + 1 + nd	n + 1 + nd	n + 1 + nd + 2p
	External	4n – 2 + (2n – 1)d	4n – 2 + (2n – 1)d	4n + 2 + 2nd + 2p
	MMR[2)]	n + 1 + nd	n + 1 + nd	n + 1 + nd + 2p
MMR[2)]	DARAM	n + 1	n + 1	n + 1 + 2p
	External	2n + (n – 1)d	2n + (n – 1)d	2n + 3 + nd + 2p
	MMR[2)]	n + 1	n + 1	n + 1 + 2p

[1)] Операнд и программный код в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 19B 2 слова, 2 цикла. Единственный операнд чтения памяти данных (Smem), использующий косвенную адресацию с длинным смещением, и единственный операнд записи в память данных (dmad); или единственный операнд чтения памяти данных (dmad) и единственный операнд записи в память данных (Smem), использующий косвенную адресацию с длинным смещением.

Мнемоники MVDK MVKD

Циклов

Циклов для однократного выполнения с длинным смещением

Операнд		Программа		
Smem	dmad	ROM	DARAM	External
DARAM	DARAM	3	3, 4[1)]	3 + 3p
	External	3	3, 4[1)]	6 + d + 3p
	MMR[2)]	3	3	3 + 3p
External	DARAM	3 + d	3 + d	4 + d + 3p
	External	3 + d	3 + d	7 + 2d + 2p
	MMR[2)]	3 + d	3 + d	4 + d + 3p
MMR[2)]	DARAM	3	3	3 + 3p
	External	3	3	6 + d + 3p
	MMR[2)]	3	3	3 + 3p

[1)] Операнд и программный код в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 20A 2 слова, 4 цикла. Единственный операнд чтения памяти данных (Smem) и единственный операнд записи в память программ (pmad).

Мнемоники MVDP

Циклов

Циклов при однократном выполнении

Операнд		Программа		
Smem	pmad	ROM	DARAM	External
DARAM	DARAM	4	4	4 + 2p
	External	4	4	6 + pd + 2p
External	DARAM	4 + d	4 + d	4 + d + 2p
	External	4 + d + pd	4 + d + pd	6 + d + pd + 2p
MMR[1)]	DARAM	4	4	4 + 2p
	External	4	4	6 + pd + 2p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Циклов при выполнении в режиме повтора

Операнд		Программа		
Smem	pmad	ROM	DARAM	External
DARAM	DARAM	n + 3	n + 3	n + 3 + 2p
	External	2n + 2 + (n – 1)pd	2n + 2 + (n – 1)pd	2n + 4 + npd + 2p
External	DARAM	n + 3 + npd	n + 3 + npd	n + 3 + npd + 2p
	External	4n + nd + npd	4n + nd + npd	4n + 2 + nd + npd + 2p
MMR[1)]	DARAM	n + 3	n + 3	n + 3 + 2p
	External	2n + 2 + (n – 1)pd	2n + 2 + (n – 1)pd	2n + 4 + npd + 2p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 20B 3 слова, 5 циклов. Единственный операнд чтения памяти данных (Smem), использующий косвенную адресацию с длинным смещением, и единственный операнд записи в память программ (pmad).

Мнемоники MVDP

Циклов

Циклов для однократного выполнения с длинным смещением

Операнд		Программа		
Smem	pmad	ROM	DARAM	External
DARAM	DARAM	5	5	5 + 3p
	External	5	5	7 + 2pd + 3p
External	DARAM	5 + d	5 + d	5 + d + 3p
	External	5 + d + 2pd	5 + d + 2pd	7 + d + 2pd + 3p
MMR[1)]	DARAM	5	5	5 + 3p
	External	5	5	7 + 3pd + 3p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 21A 2 слова, 3 цикла. Единственный операнд чтения памяти программ (pmad) и единственный операнд записи в память данных (Smem).

Мнемоники MVPD

Циклов

Циклов при однократном выполнении

Операнд		Программа		
pmad	Smem	ROM	DARAM	External
DARAM	DARAM	3	3	3 + 2p
	External	3	3	6 + d + 2p
	MMR[1)]	3	3	3 + 2p
ROM	DARAM	3	3	3 + 2p
	External	3	3	6 + d + 2p
	MMR[1)]	3	3	3 + 2p
External	DARAM	3 + pd	3 + pd	3 + pd + 2p
	External	3 + pd	3 + pd	6 + d + pd + 2p
	MMR[1)]	3 + pd	3 + pd	3 + pd + 2p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Циклов при выполнении в режиме повтора

Операнд		Программа		
pmad	Smem	ROM	DARAM	External
DARAM	DARAM	n + 2	n + 2	n + 2 + 2p
	External	2n + 1 + (n – 1)d	2n + (n – 1)d	2n + 4 + nd + 2p
	MMR[1)]	n + 2	n + 2	n + 2 + 2p
ROM	DARAM	n + 2	n + 2	n + 2 + 2p
	External	2n + 1 + (n – 1)d	2n + (n – 1)d	2n + 4 + nd + 2p
	MMR[1)]	n + 2	n + 2	n + 2 + 2p
External	DARAM	n + 2 + npd	n + 2 + npd	n + 2 + npd + 2p
	External	4n – 1 + (n – 1)d + + npd	4n – 1 + (n – 1)d + + npd	4n + 2 + nd + + npd + 2p
	MMR[1)]	n + 2 + npd	n + 2 + npd	n + 2 + npd + 2p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 21B 3 слова, 4 цикла. Единственный операнд чтения памяти программ (pmad) и единственный операнд записи в память данных (Smem), использующий косвенную адресацию с длинным смещением.

Мнемоники MVPD

Циклов

Циклов для однократного выполнения с длинным смещением

Операнд		Программа		
pmad	Smem	ROM	DARAM	External
DARAM	DARAM	4	4	4 + 3p
	External	4	4	7 + d + 3p
	MMR[1)]	4	4	4 + 3p
ROM	DARAM	4	4	4 + 3p
	External	4	4	7 + d + 3p
	MMR[1)]	4	4	4 + 3p
External	DARAM	4 + 2pd	4 + 2pd	4 + 2pd + 3p
	External	4 + 2pd	4 + 2pd	7 + d + 2pd + 3p
	MMR[1)]	4 + 2pd	4 + 2pd	4 + 2pd + 3p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 22A 2 слова, 3 цикла. Единственный операнд чтения памяти данных (Smem) и единственный операнд чтения памяти программ (pmad).

Мнемоники MACP

Циклов

Циклов при однократном выполнении

Операнд		Программа		
pmad	Smem	ROM	DARAM	External
DARAM	DARAM	3	3, 4[1)]	3 + 2p
	External	3 + d	3 + d	4 + d + 2p
	MMR[2)]	3	3	3 + 2p
ROM	DARAM	3	3, 4[1)]	3 + 2p
	External	3 + d	3 + d	4 + d + 2p
	MMR[2)]	3	3	3 + 2p
External	DARAM	3 + pd	3 + pd, 4 + pd[1)]	3 + pd + 2p
	External	4 + d + pd	4 + d + pd	4 + d + pd + 2p
	MMR[2)]	3 + pd	3 + pd	3 + pd + 2p

[1)] Операнд и программный код в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Циклов при выполнении в режиме повтора

Операнд		Программа		
pmad	Smem	ROM	DARAM	External
DARAM	DARAM	n + 2	n + 2, n + 3[1)]	n + 2 + 2p
	External	n + 2 + nd	n + 2 + nd	n + 2 + nd + 2p
	MMR[2)]	n + 2	n + 2	n + 2 + 2p
ROM	DARAM	n + 2	n + 2, n + 3[1)]	n + 2 + 2p
	External	n + 2 + nd	n + 2 + nd	n + 2 + nd + 2p
	MMR[2)]	n + 2	n + 2	n + 2 + 2p
External	DARAM	n + 2 + npd	n + 2 + npd n + 3 + npd[1)]	n + 2 + npd + 2p
	External	2n + 2 + nd + npd	2n + 2 + nd + npd	2n + 2 + nd + + npd + 2p
	MMR[2)]	n + 2 + npd	n + 2 + npd	n + 2 + npd + 2p

[1)] Операнд и программный код в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 22B 3 слова, 4 цикла. Единственный операнд чтения памяти данных (Smem), использующий косвенную адресацию с длинным смещением, и единственный операнд чтения памяти программ (pmad).

Мнемоники MACP

Циклов

Циклов для однократного выполнения с длинным смещением

Операнд		Программа		
pmad	Smem	ROM	DARAM	External
DARAM	DARAM	4	4, 5[1)]	4 + 3p
	External	4 + d	4 + d	5 + d + 3p
	MMR[2)]	4	4	4 + 3p
ROM	DARAM	4	4, 5[1)]	4 + 3p
	External	4 + d	4 + d	5 + d + 3p
	MMR[2)]	4	4	4 + 3p
External	DARAM	4 + 2pd	4 + 2pd, 5+ 2pd[1)]	4 + 2pd + 3p
	External	5 + d + 2pd	5 + d + 2pd	5 + d + 2pd + 3p
	MMR[2)]	4 + 2pd	4 + 2pd	4 + 2pd + 3p

[1)] Операнд и программный код в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 23A 2 слова, 3 цикла. Единственный операнд чтения памяти данных (Smem), единственный операнд записи в память данных (Smem) и единственный операнд чтения памяти программ (pmad).

Мнемоники MACD

Циклов

Циклов при однократном выполнении

Операнд		Программа		
pmad	Smem	ROM	DARAM	External
DARAM	DARAM	3, 4[1)]	3, 4[1)]	3 + 2p, 4 + 2p[1)]
	External	3 + d	3 + d	6 + 2d + 2p
	MMR[2)]	3	3	3 + 2p
ROM	DARAM	3	3	3 + 2p
	External	3 + d	3 + d	6 + 2d + 2p
	MMR[2)]	3	3	3 + 2p
External	DARAM	3 + pd	3 + pd	3 + pd + 2p
	External	4 + d + pd	4 + d + pd	7 + d + pd + 2p
	MMR[2)]	3 + pd	3 + pd	4 + pd + 2p

[1)] Два операнда в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Циклов при выполнении в режиме повтора

Операнд		Программа		
pmad	Smem	ROM	DARAM	External
DARAM	DARAM	n + 2, 2n + 2[1)]	n + 2, 2n + 2[1)]	n + 2 + 2p 2n + 2 + 2p[1)]
	External	4n + 1 + 2nd	4n + 1 + 2nd	4n + 2 + 2nd + 2p
	MMR[2)]	n + 2	n + 2	n + 2 + 2p
ROM	DARAM	n + 2	n + 2	n + 2 + 2p
	External	4n + 1 + 2nd	4n + 1 + 2nd	4n + 2 + 2nd + 2p
	MMR[2)]	n + 2	n + 2	n + 2 + 2p
External	DARAM	n + 2 + npd	n + 2 + npd n + 3 + npd[3)]	n + 2 + npd + 2p
	External	5n – 1 + nd + npd	5n – 1 + nd + npd	5n + 2 + nd + + npd + 2p
	MMR[2)]	n + 2 + npd	n + 2 + npd	4n + 3 + npd + 2p

[1)] Два операнда в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.
[3)] Операнд и программный код в одном блоке памяти.

Класс 23B 3 слова, 4 цикла. Единственный операнд чтения памяти данных (Smem), использующий косвенную адресацию с длинным смещением, единственный операнд записи в память данных (Smem), использующий косвенную адресацию с длинным смещением, и единственный операнд чтения памяти программ (pmad).

Мнемоники MACD

Циклов

Циклов для однократного выполнения с длинным смещением

Операнд		Программа		
pmad	Smem	ROM	DARAM	External
DARAM	DARAM	4, 5[1)]	4, 5[1)]	4 + 3p, 5 + 3p[1)]
	External	4 + d	4 + d	7 + 2d + 3p
	MMR[2)]	4	4	4 + 3p
ROM	DARAM	4	4	4 + 3p
	External	4 + d	4 + d	7 + 2d + 3p
	MMR[2)]	4	4	4 + 3p
External	DARAM	4 + 2pd	4 + 2pd	4 + pd + 3p
	External	5 + d + 2pd	5 + d + 2pd	8 + d + 2pd + 3p
	MMR[2)]	4 + 2pd	4 + 2pd	5 + 2pd + 3p

[1)] Два операнда в одном блоке памяти.
[2)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 24A 1 слово, 1 цикл. Единственный операнд чтения памяти данных (Smem) и единственный операнд записи в память данных (Smem).

Мнемоники DELAY LTD

Циклов

Циклов при однократном выполнении

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	1	1, 2[1)]	1 + p
External	1 + d	1 + d	5 + p + 2d

[1)] Операнд и программный код в одном блоке памяти.

Циклов при выполнении в режиме повтора

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	n	n, n + 1[1)]	n + p
External	4n – 3 + (2n – 1)d	4n – 3 + (2n – 1)d	4n + 1 + p + nd

[1)] Операнд и программный код в одном блоке памяти.

Класс 24B 2 слова, 2 цикла. Единственный операнд чтения памяти данных (Smem), использующий косвенную адресацию с длинным смещением, и единственный операнд записи в память данных (Smem), использующий косвенную адресацию с длинным смещением.

Мнемоники DELAY LTD

Циклов

Циклов для однократного выполнения с длинным смещением

Операнд	Программа		
Smem	ROM	DARAM	External
DARAM	2	2, 3[1)]	2 + 2p
External	2 + d	2 + d	6 + 2p + 2d

[1)] Операнд и программный код в одном блоке памяти.

Класс 25A 1 слово, 5 циклов. Единственный адрес памяти программ (pmad) и единственный операнд записи в память данных (Smem).

Мнемоники READA

Циклов

Циклов при однократном выполнении

Операнд		Программа		
pmad	Smem	ROM	DARAM	External
DARAM	DARAM	5	5	5 + p
	External	5	5	8 + d + p
	MMR[1)]	5	5	5 + p
ROM	DARAM	5	5	5 + p
	External	5	5	8 + d + p
	MMR[1)]	5	5	5 + p
External	DARAM	5 + pd	5 + pd	5 + pd + p
	External	5 + pd	5 + pd	8 + pd + d + p
	MMR[1)]	5 + pd	5 + pd	5 + pd + p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Циклов при выполнении в режиме повтора

Операнд		Программа		
pmad	Smem	ROM	DARAM	External
DARAM	DARAM	n + 4	n + 4	n + 4 + p
	External	2n + 3 + (n – 1)d	2n + 3 + (n – 1)d	2n + 6 + nd + np
	MMR[1)]	n + 4	n + 4	n + 4 + p
ROM	DARAM	n + 4	n + 4	n + 4 + p
	External	2n + 3 + (n – 1)d	2n + 3 + (n – 1)d	2n + 6 + nd + np
	MMR[1)]	n + 4	n + 4	n + 4 + p
External	DARAM	n + 4 + npd	n + 4 + npd	n + 4 + npd + p
	External	4n + 1 + (n – 1)d + + npd	4n + 1 + (n – 1)d + + npd	4n + 4 + nd + + npd + p
	MMR[1)]	n + 4 + npd	n + 4 + npd	n + 4 + npd + p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 25B 2 слова, 6 циклов. Единственный адрес памяти программ (pmad) и единственный операнд записи в память данных (Smem), использующий косвенную адресацию с длинным смещением.

Мнемоники READA

Циклов

Циклов для однократного выполнения с длинным смещением

Операнд		Программа		
pmad	Smem	ROM	DARAM	External
DARAM	DARAM	6	6	6 + 2p
	External	6	6	9 + d + 2p
	MMR[1)]	6	61	6 + 2p
ROM	DARAM	6	6	6 + 2p
	External	6	6	9 + d + 2p
	MMR[1)]	6	6	6 + 2p
External	DARAM	6 + 2pd	6 + 2pd	6 + 2pd + 2p
	External	6 + 2pd	6 + 2pd	9 + 2pd + d + 2p
	MMR[1)]	6 + 2pd	6 + 2pd	6 + 2pd + 2p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 26A 1 слово, 5 циклов. Единственный операнд чтения памяти данных (Smem) и единственный адрес записи в память программ (pmad).

Мнемоники WRITA

Циклов

Циклов при однократном выполнении

Операнд		Программа		
Smem	pmad	ROM	DARAM	External
DARAM	DARAM	5	5	5 + p
	External	5	5	5 + pd + p
External	DARAM	5 + pd	5 + pd	5 + pd + p
	External	5 + d	5 + d	7 + d + pd + p
MMR[1)]	DARAM	5	5	5 + p
	External	5	5	5 + pd + p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Циклов при выполнении в режиме повтора

Операнд		Программа		
Smem	pmad	ROM	DARAM	External
DARAM	DARAM	n + 4	n + 4	n + 4 + p
	External	2n + 3 + (n – 1)d	2n + 3 + (n – 1)d	2n + 3 + npd + p
External	DARAM	n + 4 + npd	n + 4 + npd	n + 4 + npd + p
	External	4n + 1 + nd + + (n – 1)pd	4n + 1 + nd + + (n – 1)pd	4n + 3 + nd + + npd + p
MMR[1)]	DARAM	n + 4	n + 4	n + 4 + p
	External	2n + 3 + (n – 1)d	2n + 3 + (n – 1)d	2n + 3 + npd + p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 26B 2 слова, 6 циклов. Единственный операнд чтения памяти данных (Smem), использующий косвенную адресацию с длинным смещением, и единственный адрес записи в память программ (pmad).

Мнемоники WRITA

Циклов

Циклов для однократного выполнения с длинным смещением

Операнд		Программа		
Smem	pmad	ROM	DARAM	External
DARAM	DARAM	6	6	6 + 2p
	External	6	6	6 + 2pd + 2p
External	DARAM	6 + 2pd	6 + 2pd	6 + 2pd + 2p
	External	6 + d	6 + d	8 + d + 2pd +2p
MMR[1)]	DARAM	6	6	6 + 2p
	External	6	6	6 + 2pd + 2p

[1)] При доступе к периферийным картированным в памяти регистрам добавляется один цикл.

Класс 27A 2 слова, 2 цикла. Единственный операнд чтения порта ввода/вывода и единственный операнд записи в память данных (Smem).

Мнемоники PORTR

Циклов

Циклов при однократном выполнении

Операнд		Программа		
Port	Smem	ROM	DARAM	External
External	DARAM	3 + io	3 + io	6 + 2p + io
	External	3 + io	3 + io	9 + 2p + d + io

Циклов при выполнении в режиме повтора

Операнд		Программа		
Port	Smem	ROM	DARAM	External
External	DARAM	2n +1 + nio	2n +1 + nio	2n +4 + 2p + nio
	External	5n – 2 + nio + + (n – 1)d	5n – 2 + nio + + (n – 1)d	5n + 4 + 2p + + nio + nd

Класс 27B 3 слова, 3 цикла. Единственный операнд чтения порта ввода/вывода и единственный операнд записи в память данных (Smem), использующий косвенную адресацию с длинным смещением.

Мнемоники PORTR

Циклов

Циклов для однократного выполнения с длинным смещением

Операнд		Программа		
Port	Smem	ROM	DARAM	External
External	DARAM	4 + io	4 + io	7 + 3p + io
	External	4 + io	4 + io	10 + 3p + d + io

Класс 28A 2 слова, 2 цикла. Единственный операнд чтения памяти данных (Smem) и единственный операнд записи в порта ввода/вывода.

Мнемоники PORTW

Циклов

Циклов при однократном выполнении

Операнд		Программа		
Port	Smem	ROM	DARAM	External
External	DARAM	2	2, 3[1)]	6 + 2p + io
	External	2 + d	2 + d	7 + 2p + d + io

[1)] Операнд и программный код в одном блоке памяти.

Циклов при выполнении в режиме повтора

Операнд		Программа		
Port	Smem	ROM	DARAM	External
External	DARAM	2n + (n – 1)io	2n + (n – 1)io 2n +1 + (n – 1)io[1)]	2n +4 + 2p + nio
	External	5n – 3 + nd + + (n – 1)io	5n – 3 + nd + + (n – 1)io	5n + 2 + 2p + + nd + nio

[1)] Операнд и программный код в одном блоке памяти.

Класс 28B 3 слова, 3 цикла. Единственный операнд чтения памяти данных (Smem), использующий косвенную адресацию с длинным смещением, и единственный операнд записи в порта ввода/вывода.

Мнемоники PORTW

Циклов

Циклов для однократного выполнения с длинным смещением

Операнд		Программа		
Port	Smem	ROM	DARAM	External
External	DARAM	3	3, 4[1)]	7 + 3p + io
	External	3 + d	3 + d	8 + 3p + d + io

[1)] Операнд и программный код в одном блоке памяти.

Класс 29A 2 слова, 4 цикла, 2 цикла (для задержанных), 2 цикла (условие невыполнено). Единственный операнд памяти программ (pmad).

Мнемоники B[D] BANZ[D] RPTB[D]

Циклов

Циклов при однократном выполнении

Программа		
ROM	DARAM	External
4	4	4 + 4p

Циклов при однократном задержанном выполнении

Программа		
ROM	DARAM	External
2	2	2 + 2p

Класс 29B 2 слова, 4 цикла, 2 цикла (для задержанных). Единственный операнд памяти программ (pmad).

Мнемоники CALL[D]

Циклов

Циклов при однократном выполнении

Операнд	Программа		
Stack	ROM	DARAM	External
DARAM	4	4	4 + 4p
External	4	4	7 + 4p + d

Циклов при однократном задержанном выполнении

Операнд	Программа		
Stack	ROM	DARAM	External
DARAM	2	2	2 + 2p
External	2	2	5 + 2p + d

Класс 30A 1 слово, 6 циклов, 4 цикла (для задержанных). Единственный регистровый операнд.

Мнемоники BACC[D]

Циклов

Циклов при однократном выполнении

Программа		
ROM	DARAM	External
6	6	6 + 3p

Циклов при однократном задержанном выполнении

Программа		
ROM	DARAM	External
4	4	4 + p

Класс 30B 1 слово, 6 циклов, 4 цикла (для задержанных). Единственный регистровый операнд.

Мнемоники CALA[D]

Циклов

Циклов при однократном выполнении

Операнд	Программа		
Stack	ROM	DARAM	External
DARAM	6	6	6 + 3p
External	6	4	7 + 3p + d

Циклов при однократном задержанном выполнении

Операнд	Программа		
Stack	ROM	DARAM	External
DARAM	4	4	4 + p
External	4	4	5 + p + d

Класс 31A 2 слова, 5 циклов, 3 цикла (для задержанных). Единственный операнд программной памяти (pmad) и короткие непосредственные операнды.

Мнемоники BC[D]

Циклов

Циклов при однократном выполнении

	Программа		
Условие	ROM	DARAM	External
Выполнено	5	5	5 + 4p
Невыполнено	3	3	3 + 2p

Циклов при однократном задержанном выполнении

	Программа		
Условие	ROM	DARAM	External
Выполнено	3	3	3 + 2p
Невыполнено	3	3	3 + 2p

Класс 31B 2 слова, 5 циклов, 3 цикла (для задержанных), 3 цикла (условие невыполнено). Единственный операнд программной памяти (pmad) и короткие непосредственные операнды.

Мнемоники CC[D]

Циклов

Циклов при однократном выполнении, если условие выполнено

Операнд	Программа		
Stack	ROM	DARAM	External
DARAM	5	5	5 + 4p
External	5	5	8 + 4p + d

Циклов при однократном выполнении, если условие выполнено

Операнд	Программа		
Stack	ROM	DARAM	External
DARAM	3	3	3 + 2p
External	3	3	6 + 2p + d

Циклов при однократном задержанном выполнении

Операнд	Программа		
Stack	ROM	DARAM	External
DARAM	3	3	3 + 2p
External	3	3	6 + 2p + d

Класс 32 1 слово, 5 циклов, 3 цикла (для задержанных), 3 цикла (условие невыполнено). Нет операндов или короткие непосредственные операнды.

Мнемоники RC[D] RET[D] RETE[D]

Циклов

Циклов при однократном выполнении

Операнд	Программа		
Stack	ROM	DARAM	External
DARAM	5	5, 6[1)]	5 + 3p
External	5 + d	5 + d	6 + d + 3p

[1)] Операнд и программный код в одном блоке памяти.

Циклов при однократном задержанном выполнении

Операнд	Программа		
Stack	ROM	DARAM	External
DARAM	3	3, 4[1)]	3 + p
External	3 + d	3 + d	4 + d + p

[1)] Операнд и программный код в одном блоке памяти.

Класс 33 1 слово, 3 цикла, 1 цикл (для задержанных). Нет операндов.

Мнемоники RETF[D]

Циклов

Циклов при однократном выполнении

Программа		
ROM	DARAM	External
3	3	3 + p

Циклов при однократном задержанном выполнении

Программа		
ROM	DARAM	External
1	1	1 + p

Класс 34 ЗАРЕЗЕРВИРОВАН ДЛЯ ДАЛЬНЕЙШИХ МОДИФИКАЦИЙ ПРОЦЕССОРА

Класс 35 1 слово, 3 цикла. Нет операндов или короткий непосредственный операнд.

Мнемоники INTR RESET TRAP

Циклов

Циклов при однократном выполнении

Программа		
ROM	DARAM	External
3	3	3 + p

Класс 36 1 слово, 4 цикла (минимум). Короткий непосредственный операнд.

Мнемоники IDLE

Циклов Количество циклов выполнения этой инструкции зависит продолжительности нахождения процессора в режиме ожидания (idle).

4 Подробное описание набора инструкций

Символы и аббревиатуры, использованные при описании набора инструкций, приведены в подразделе 1.1 "Символы и аббревиатуры набора инструкций"; элементы инструкции описаны в подразделе 1.2 "Пример описания инструкции". Раздел 2 содержит краткое сводное описание набора инструкций, сгруппированное по их функциональному назначению.

ABDST Абсолютная расстояние

Синтаксис: ABDST *Xmem, Ymem*

Операнды: Xmem, Ymem: Двойные операнды памяти данных

Опкод:

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

Выполнение: (B) + |A(32-16)| → B
((Xmem) – (Ymem)) << 16 → A

Биты состояния: зависит от OVM, FRCT и SXM
оказывает влияние на C, OVA и OVB

Описание: Эта инструкция вычисляет абсолютное значение расстояния между двумя векторами *Xmem* и *Ymem*. Абсолютное значение аккумулятора A(32-16) прибавляется к аккумулятору B. Содержимое Ymem вычитается из содержимого Xmem и результат, сдвинутый влево на 16 разрядов, сохраняется в аккумуляторе A. Если активен режим дробных вычислений (бит FRCT = 1), то абсолютное значение умножается на 2.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 7 (см. страницу 41)

Пример:

```
ABDST *AR3+, *AR4+
```

	Before Instruction	After Instruction
A	FF ABCD 0000	FF FFAB 0000
B	00 0000 0000	00 0000 5433
AR3	0100	0101
AR4	0200	0201
FRCT	0	0
Data Memory		
0100h	0055	0055
0200h	00AA	00AA

ABS Абсолютное значение аккумулятора

Синтаксис: ABS *src*[, *dst*]

Операнды: src, dst: A (аккумулятор A)
B (аккумулятор B)

Опкод:

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

Выполнение: |(src)| → dst (или src, если dst не определен)

Биты состояния: OVM влияет на эту инструкцию следующим образом:
Если OVM = 1, абсолютное значение 80 000 000h
будет равно 00 7FFF FFFFh
Если OVM = 0, абсолютное значение 80 000 000h
будет равно 80 0000 0000h
Оказывает влияние на C и OVdst (или OVsrc, если dst = src)

Описание: Эта инструкция вычисляет абсолютное значение src и загружает полученное значение в dst. Если dst не определен, полученное абсолютное значение загружается в src.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 1 (см. страницу 32)

Пример 1:

```
ABS A, B
```

	Before Instruction		After Instruction	
A	FF FFFF FFCB	-53	FF FFFF FFCB	-53
B	FF FFFF FC18	-1000	00 0000 0035	+53

Пример 2:

```
ABS A
```

	Before Instruction	After Instruction
A	03 1234 5678	00 7FFF FFFF
OVM	1	1

Пример 3:

```
ABS A
```

	Before Instruction	After Instruction
A	03 1234 5678	03 1234 5678
OVM	0	0

ADD Сложение с аккумулятором

Синтаксис:

1: ADD *Smem, src*
2: ADD *Smem,* TS, *src*
3: ADD *Smem,* 16, *src* [, *dst*]
4: ADD *Smem* [, *SHIFT*], *src* [, *dst*]
5: ADD *Xmem, SHFT, src*
6: ADD *Xmem, Ymem, dst*
7: ADD *#lk* [, *SHFT*], *src* [, *dst*]
8: ADD *#lk,* 16, *src*[, *dst*]
9: ADD *src* [, *SHIFT*], [, *dst*]
10: ADD *src,* ASM [, *dst*]

Операнды:

Smem:	Одиночный операнд памяти данных
Xmem, Ymem	Двойные операнды памяти данных
src, dst:	A (аккумулятор A)
	B (аккумулятор B)

$-32\,768 \le lk \le 32\,767$
$-16 \le SHIFT \le 15$
$0 \le SHFT \le 15$

Опкоды:

1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	S	I	A	A	A	A	A	A	A

2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	S	I	A	A	A	A	A	A	A

3:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	1	S	D	I	A	A	A	A	A	A	A

4:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	1	1	1	I	A	A	A	A	A	A	A
0	0	0	0	1	1	S	D	0	0	0	S	H	I	F	T

5:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	0	0	0	S	X	X	X	X	S	H	F	T

6:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	0	0	0	D	X	X	X	X	Y	Y	Y	Y

7:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	0	0	0	0	S	H	F	T
16-битная константа															

8:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	0	1	1	0	0	0	0	0
16-битная константа															

ADD Сложение с аккумулятором

9:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	S	D	0	0	0	S	H	I	F	T

10:

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

Выполнение:

1: (Smem) + (src) → src
2: (Smem) << (TS) + (src) → src
3: (Smem) << 16 + (src) → dst
4: (Smem) [<< SHIFT] + (src) → dst
5: (Xmem) << SHFT + (src) → src
6: ((Xmem) + (Ymem)) << 16 → dst
7: lk << SHFT + (src) → dst
8: lk << 16 + (src) → dst
9: (src или [dst]) + (src) << SHIFT → dst
10: (src или [dst]) + (src) << ASM → dst

Биты состояния:

Зависит от SXM и OVM
Оказывает действие на C и OVdst (или OVsrc, если dst = src)
Для синтаксиса 3, если в результате сложения происходит генерация переноса, то бит переноса C устанавливается в 1; в других случаях бит C не затрагивается.

Описание:

Эта инструкция прибавляет 16-разрядное значение к содержимому выбранного аккумулятора или к 16-разрядному операнду *Xmem* при использовании адресации с двойным операндом памяти данных. В качестве 16-разрядного значения могут выступать:

- Содержимое одиночного операнда памяти данных (*Smem*).
- Содержимое двойного операнда памяти данных (*Ymem*).
- 16-разрядный непосредственный операнд (*#lk*).
- Сдвинутое значение *src*.

Если *dst* определен, инструкция сохраняет результат в *dst*. Если *dst* не определен, то результат будет сохранен в *src*. К большинству вторых операндов может быть применен сдвиг.
При сдвиге влево:

- Младшие биты очищаются.
- Старшие биты:
  - Расширяются значением знакового бита, при SXM = 1.
  - Очищаются, при SXM = 0.

При сдвиге вправо, старшие биты:

- Расширяются значением знакового бита, при SXM = 1.
- Очищаются, при SXM = 0.

ADD Сложение с аккумулятором

Примечание – Следующие синтаксисы инструкций при определенных условиях могут быть транслированы как другие синтаксисы:

– Синтаксис 4: Если *dst* = *src* и *SHIFT* = 0, опкод инструкции транслируется как синтаксис 1.

– Синтаксис 4: Если *dst* = *src* и *SHIFT* ≤ 15 и режимом косвенной адресации *Smem* включен в *Xmem*, опкод инструкции транслируется как синтаксис 5.

– Синтаксис 5: Если *SHIFT* = 0, опкод инструкции транслируется как синтаксис 1.

Слов: Синтаксисы 1, 2, 3, 5, 6, 9 и 10: 1 слово
Синтаксисы 4, 7, и 8: 2 слова
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: Синтаксисы 1, 2, 3, 5, 6, 9 и 10: 1 цикл
Синтаксисы 4, 7, и 8: 2 цикла
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Синтаксисы 1, 2, 3 и 5: Класс 3A (см. страницу 34)
Синтаксисы 1, 2 и 3: Класс 3B (см. страницу 35)
Синтаксис 4: Класс 4A (см. страницу 36)
Синтаксис 4: Класс 4B (см. страницу 37)
Синтаксис 6: Класс 7 (см. страницу 41)
Синтаксисы 7 и 8: Класс 2 (см. страницу 33)
Синтаксисы 9 и 10: Класс 1 (см. страницу 32)

Пример 1:

```
ADD *AR3+, 14, A
```

	Before Instruction		After Instruction
A	00 0000 1200	A	00 0540 1200
C	1	C	0
AR3	0100	AR3	0101
SXM	1	SXM	1
Data Memory			
0100h	1500	0100h	1500

Пример 2:

```
ADD A, -8, B
```

	Before Instruction		After Instruction
A	00 0000 1200	A	00 0000 1200
B	00 0000 1800	B	00 0000 1812
C	1	C	0

ADD Сложение с аккумулятором

Пример 3:

```
ADD #4568, 8, A, B
```

Пример 4:

```
ADD *AR2+, *AR2-, A ;после осуществления доступа
                    ;к операнду, AR2 увеличивается на 1
```

Пример 4 показывает один и тот же вспомогательный регистр (AR2) с различными режимами адресации, определенными для обоих операндов. Для адресации используется режим, определяемый полем Xmod (*AR2+).

ADDC Сложение с аккумулятором с добавлением переноса

Синтаксис: ADDC *Smem, src*

Операнды:
Smem: Одиночный операнд памяти данных
src: A (аккумулятор A)
B (аккумулятор B)

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	S	I	A	A	A	A	A	A	A

Выполнение: (Smem) + (src) + (C) → src

Биты состояния: Зависит от OVM, C
Оказывает влияние на C и OVsrc

Описание: Эта инструкция складывает 16-разрядный одиночный операнд памяти данных *Smem* и значение бита переноса C с *src*. Результат сохраняется в *src*. Независимо от значения бита SXM режим расширения знака не активизируется.

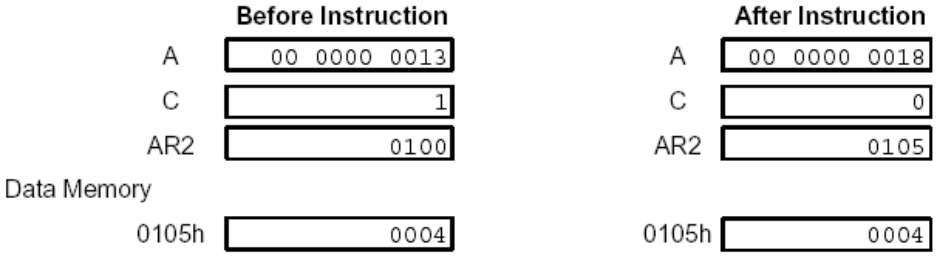
Слов: 1 слово
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 1 цикл
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 3A (см. страницу 34)
Класс 3B (см. страницу 35)

Пример:

```
ADDC *+AR2(5), A
```



ADDM Добавление длинного непосредственного значения в память

Синтаксис: ADDM *#lk, Smem*

Операнды: Smem: Одиночный операнд памяти данных
$-32\,768 \le lk \le 32\,767$

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	0	1	1	I	A	A	A	A	A	A	A
16-битная константа															

Выполнение: #lk + (Smem) → (Smem)

Биты состояния: Зависит от OVM и SXM
Оказывает влияние на C и OVA

Описание: Эта инструкция добавляет 16-разрядный одиночный операнд памяти данных *Smem* к 16-разрядному непосредственному значению *lk* и сохраняет результат в *Smem*.

> Примечание – Эта инструкция не может быть повторяемой.

Слов: 2 слова

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 2 цикла

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 18A (см. страницу 54)
Класс 18B (см. страницу 54)

Пример 1:

```
ADDM 0123Bh, *AR4+
```

	Before Instruction	After Instruction
AR4	0100	0101
Data Memory		
0100h	0004	123F

Пример 2:

```
ADDM 0FFF8h, *AR4+
```

	Before Instruction	After Instruction
OVM	1	1
SXM	1	1
AR4	0100	0101
Data Memory		
0100h	8007	8000

ADDS Добавление в аккумулятор с подавлением знакового расширения

Синтаксис: ADDS *Smem, src*

Операнды:
Smem: Одиночный операнд памяти данных
src: A (аккумулятор A)
B (аккумулятор B)

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	S	I	A	A	A	A	A	A	A

Выполнение: uns(Smem) + (src) → src

Биты состояния: Зависит от OVM
Оказывает влияние на C и OVsrc

Описание: Эта инструкция добавляет 16-разрядный одиночный операнд памяти данных *Smem* к *src* и сохраняет результат в *src*. Независимо от значения бита SXM режим расширения знака не активизируется.

Слов: 1 слово
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 1 цикл
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 3A (см. страницу 34)
Класс 3B (см. страницу 35)

Пример:

```
ADDS *AR2-, B
```

	Before Instruction	After Instruction
B	00 0000 0003	00 0000 F009
C	x	0
AR2	0100	00FF
Data Memory		
0104h	F006	F006

AND Логическое "И" с аккумулятором

Синтаксис:

1: AND *Smem, src*
2: AND *#lk*[, *SHFT*], *src*[, *dst*]
3: AND *#lk*, 16, *src*[, *dst*]
4: AND *src*[, *SHIFT*], [, *dst*]

Операнды:

Smem:	Одиночный операнд памяти данных
src, dst:	A (аккумулятор A)
	B (аккумулятор B)

$0 \le lk \le 65\,535$
$-16 \le SHIFT \le 15$
$0 \le SHFT \le 15$

Опкоды:

1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	0	0	S	I	A	A	A	A	A	A	A

2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	0	0	1	1	S	H	F	T
16-битная константа															

3:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	0	1	1	0	0	0	1	1
16-битная константа															

4:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	1	0	0	S	H	I	F	T

Выполнение:

1: (Smem) AND (src) → src
2: lk << SHFT AND (src) → dst
3: lk << 16 AND (src) → dst
4: (dst) AND (src) << SHIFT → dst

Биты состояния: нет

Описание:

Эта инструкция осуществляет выполнение логической операции "И" (логического умножения) между *src* и:

- 16-разрядным операндом *Smem*.
- 16-разрядным непосредственным операндом *lk*.
- Аккумулятором-источником или приемником (*src* или *dst*)

Если в инструкции задан сдвиг, то левый сдвиг операнда осуществляется перед выполнением операции "И". При сдвиге влево младшие биты очищаются, старшие биты не расширяются на знак. При сдвиге вправо старшие биты не расширяются на знак.

AND Логическое "И" с аккумулятором

Слов: Синтаксисы 1 и 4: 1 слово
Синтаксисы 2 и 3: 2 слова
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: Синтаксисы 1 и 4: 1 цикл
Синтаксисы 2 и 3: 2 цикла
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Синтаксис 1: Класс 3A (см. страницу 34)
Синтаксис 1: Класс 3B (см. страницу 35)
Синтаксисы 2 и 3: Класс 2 (см. страницу 33)
Синтаксис 4: Класс 1 (см. страницу 32)

Пример 1:

```
AND *AR3+, A
```

	Before Instruction		After Instruction
A	00 00FF 1200	A	00 0000 1000
AR3	0100	AR3	0101
Data Memory			
0100h	1500	0100h	1500

Пример 2:

```
AND A, 3, B
```

	Before Instruction		After Instruction
A	00 0000 1200	A	00 0000 1200
B	00 0000 1800	B	00 0000 1000

ANDM Логическое "И" ячейки памяти с длинным непосредственным значением

Синтаксис: ANDM *#lk, Smem*

Операнды: Smem: Одиночный операнд памяти данных
$0 \le lk \le 65\,535$

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	0	0	0	I	A	A	A	A	A	A	A
16-битная константа															

Выполнение: lk AND (Smem) → Smem

Биты состояния: нет

Описание: Эта инструкция производит логическую операцию "И" над одиночным операндом памяти данных *Smem* и 16-разрядной длинной константой *lk*. Результат сохраняется в ячейке памяти данных, определяемой *Smem*.

Примечание – Эта инструкция неповторяемая.

Слов: 2 слова
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 2 цикла
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 18A (см. страницу 54)
Класс 18B (см. страницу 54)

Пример 1:

```
ANDM #00FFh, *AR4+
```

	Before Instruction	After Instruction
AR4	0100	0101
Data Memory		
0100h	0444	0044

Пример 2:

```
ANDM #0101h, 4; DP = 0
```

	Before Instruction	After Instruction
Data Memory		
0004h	00 0000 0100	00 0000 0100

B[D] Безусловный переход

Синтаксис: B[D] *pmad*

Операнды: $0 \leq pmad \leq 65\,535$

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	Z	0	0	1	1	1	0	0	1	1
16-битная константа															

Выполнение: pmad → PC

Биты состояния: нет

Описание: Эта инструкция передает управление программой определенному адресу памяти программ (*pmad*), который может быть как цифровым, так и символическим. Если переход выполняется с задержкой (при указании суффикса D), выбираются и исполняются две однословных инструкции или одна двухсловная инструкция, следующие(ая) за инструкцией перехода.

Примечание – Эта инструкция неповторяемая.

Слов: 2 слова

Циклов: 4 цикла
2 цикла (для задержанных)

Классы: Класс 29A (см. страницу 74)

Пример 1:

```
B 2000h
```

	Before Instruction		After Instruction
PC	1F45	PC	2000

Пример 2:

```
BD 1000h
ANDM 4444h, *AR1+
```

	Before Instruction		After Instruction
PC	1F45	PC	1000

После того, как операнд логически умножается на 4444h, программа продолжает выполнение с адреса 1000h.

BACC[D] Переход по содержимому аккумулятора

Синтаксис: BACC[D] *src*

Операнды: src: A (аккумулятор A)
B (аккумулятор B)

Опкод:

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

Выполнение: (src(15-0)) → PC

Биты состояния: нет

Описание: Эта инструкция передает управление программой по 16-разрядному адресу в младшей части *src* (биты 15-0). Если переход выполняется с задержкой (при указании суффикса D), выбираются и исполняются две однословных инструкции или одна двухсловная инструкция, следующие(ая) за инструкцией перехода.

> Примечание – Эта инструкция неповторяемая.

Слов: 1 слово

Циклов: 6 циклов
4 цикла (для задержанных)

Классы: Класс 30A (см. страницу 75)

Пример 1:

```
BACC A
```

	Before Instruction		After Instruction
A	00 0000 3000	A	00 0000 3000
PC	1F45	PC	3000

Пример 2:

```
BACCD B
ANDM 4444h, *AR1+
```

	Before Instruction		After Instruction
B	00 0000 2000	B	00 0000 2000
PC	1F45	PC	2000

После того, как операнд логически умножается на 4444h, программа продолжает выполнение с адреса 2000h.

BANZ[D] Условный переход, если вспомогательный регистр не равен нулю

Синтаксис: BANZ[D] *pmad, Sind*

Операнды: Sind: Одиночный операнд с косвенной адресацией
$0 \leq pmad \leq 65\,535$

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	1	Z	0	I	A	A	A	A	A	A	A
16-битная константа															

Выполнение: Если ((ARx) ≠ 0),
то
pmad → PC
иначе
(PC) + 2 → PC

Биты состояния: нет

Описание: Эта инструкция выполняет переход к указанному адресу памяти программ (pmad), если значение вспомогательного регистра ARx не ноль. В противном случае, значение счетчика комманд увеличивается на 2. Если переход выполняется с задержкой (при указании суффикса D), выбираются и исполняются две однословных инструкции или одна двухсловная инструкция, следующие(ая) за инструкцией перехода.

Примечание – Эта инструкция неповторяемая.

Слов: 2 слова

Циклов: 4 цикла (условие выполнено)
2 цикла (условие не выполнено)
2 цикла (для задержанных)

Классы: Класс 29A (см. страницу 74)

Пример 1:

```
BANZ 2000h, *AR3-
```

	Before Instruction		After Instruction
PC	1000	PC	2000
AR3	0005	AR3	0004

Пример 2:

```
BANZ 2000h, *AR3-
```

	Before Instruction		After Instruction
PC	1000	PC	1002
AR3	0000	AR3	FFFF

BANZ[D] Условный переход, если вспомогательный регистр не равен нулю

Пример 3:

```
BANZ 2000h, *AR3(-1)
```

	Before Instruction		After Instruction
PC	1000	PC	1003
AR3	0001	AR3	0001

Пример 4:

```
BANZD 2000h, *AR3-
ANDM 4444h, *AR5+
```

	Before Instruction		After Instruction
PC	1000	PC	2000
AR3	0004	AR3	0003

После того, как содержимое памяти данных логически умножается на 4444h, программа продолжает выполнение с адреса 2000h.

BC[D] Условный переход

Синтаксис: BC[D] *pmad*, *cond*[, *cond*[, *cond*]]

Операнды: $0 \le pmad \le 65\,535$

Следующая таблица содержит список условий (cond) для этой инструкции

Условие	Описание	Код условия	Условие	Описание	Код условия
BIO	BIO# low	0000 0011	NBIO	BIO# high	0000 0010
C	C = 1	0000 1100	NC	C = 0	0000 1000
TC	TC = 1	0011 0000	NTC	TC = 0	0010 0000
AEQ	(A) = 0	0100 0101	BEQ	(B) = 0	0100 1101
ANEQ	(A) ≠ 0	0100 0100	BNEQ	(B) ≠ 0	0100 1100
AGT	(A) > 0	0100 0110	BGT	(B) > 0	0100 1110
AGEQ	(A) ≥ 0	0100 0010	BGEQ	(B) ≥ 0	0100 1010
ALT	(A) < 0	0100 0011	BLT	(B) < 0	0100 1011
ALEQ	(A) ≤ 0	0100 0111	BLEQ	(B) ≤ 0	0100 1111
AOV	A переполнен	0111 0000	BOV	B переполнен	0111 1000
ANOV	A не переп-н	0110 0000	BNOV	B не переп-н	0110 1000
UNC	безусловный	0000 0000			

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	Z	0	C	C	C	C	C	C	C	C
16-битная константа															

Выполнение: Если (условие(я)),
то
pmad → PC
иначе
(PC) + 2 → PC

Биты состояния: Влияет на OVA или OVB, если выбраны OV или NOV

Описание: Эта инструкция выполняет переход к адресу памяти программ (*pmad*), если выполнены все указанные условия. Независимо от выполнения условия(й) из памяти программ выбираются (но не исполняются) две однословные или одна двухсловная инструкция, следующие(ая) за инструкцией перехода. Если указанные условия выполнены, то эти два слова изымаются из конвейера и выполнение продолжается с *pmad*. Если условие(я) невыполнено(ы), PC увеличивается на 2 и исполняются два (уже находящихся в конвейере) слова, располагавшиеся после инструкции перехода.
Если инструкция выполняется с задержкой (при указании суффикса D), то сначала из памяти программ выбираются и исполняются две однословные инструкции или одна двухсловная. Эти инструкции не оказывают влияния на проверяемые условия. Если указанные условия выполняются, то выполнение программы продолжается с *pmad*. Если условия не выполняются, то PC увеличивается на 2, и выполняются два слова после инструкции условного перехода.

BC[D] Условный переход

Эта инструкция позволяет осуществлять проверку нескольких условий перед передачей управления в другую секцию программы. Инструкция может проверять условия индивидуально или в комбинации с другими условиями. Можно компоновать условия только из одной группы следующим образом:

Группа 1 – Допускается выбрать до двух условий. Каждое условие должно принадлежать разным категориям (A или B, см. таблицу ниже). Не допускается использование в инструкции двух условий из одной категории. Например, одновременно можно тестировать EQ и OV, однако, нельзя выполнить одновременную проверку GT и NEQ. Для обоих условий аккумулятор должен быть одним и тем же. Не допускается проверка двух условий для разных аккумуляторов в одной инструкции. Например, разрешена одновременная проверка AGT и AOV, однако, не допускается использование в одной инструкции условий AGT и BOV.

Группа 2 – Допускается выбрать до трех условий. Каждое из трех условий должно быть из различных категорий (A, B или C). Не допускается использование двух или более условий из одной категории, т. е. допускается одновременная проверка TC, C и BIO#, однако, не допускается одновременная проверка NTC, C и NC.

Сочетаемость условий для этой инструкции

Группа 1		Группа 2		
Категория A	Категория B	Категория A	Категория B	Категория C
EQ	OV	TC	C	BIO
NEQ	NOV	NTC	NC	NBIO
LT				
LEQ				
GT				
GEQ				

Примечание – Эта инструкция неповторяемая.

Слов: 2 слова

Циклов: 5 циклов (условие выполнено)
3 цикла (условие невыполнено)
3 цикла (для задержанных)

Классы: Класс 31A (см. страницу 76)

BC[D] Условный переход

Пример 1:

```
BC 2000h, AGT
```

	Before Instruction		After Instruction
A	00 0000 0053	A	00 0000 0053
PC	1000	PC	2000

Пример 2:

```
BC 2000h, AGT
```

	Before Instruction		After Instruction
A	FF FFFF FFFF	A	FF FFFF FFFF
PC	1000	PC	1002

Пример 3:

```
BCD 1000h, BOV
ANDM 4444h, *AR1+
```

	Before Instruction		After Instruction
PC	3000	PC	1000
OVB	1	OVB	1

После логического умножения содержимого памяти на 4444h, переход выполняется, если условие выполнено. В противном случае, выполнение программы продолжается с инструкции, расположенной за инструкцией перехода.

Пример 4:

```
BC 1000h, TC, NC, BIO
```

	Before Instruction		After Instruction
PC	3000	PC	3002
C	1	C	1

BIT Тестирование бита

Синтаксис: BIT *Xmem*, BITC

Операнды: Xmem: Двойной операнд памяти данных
$0 \le BITC \le 15$

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	0	1	1	0	X	X	X	X	B	I	T	C

Выполнение: (Xmem(15 – BITC)) → TC

Биты состояния: Оказывает действие на состояние TC

Описание: Эта инструкция копирует указанный бит двойного операнда памяти данных *Xmem* в бит TC регистра состояния ST0. В таблице ниже перечислены коды разрядов, соответствующие каждому биту в памяти данных.
Коды битов соответствуют BITC и адреса битов соответствуют (15 – BITC).

Коды битов для инструкции BIT

Адрес бита		Код бита	Адрес бита		Код бита
(LSB)	0	1111		8	0111
	1	1110		9	0110
	2	1101		10	0101
	3	1100		11	0100
	4	1011		12	0011
	5	1010		13	0010
	6	1001		14	0001
	7	1000	(MSB)	15	0000

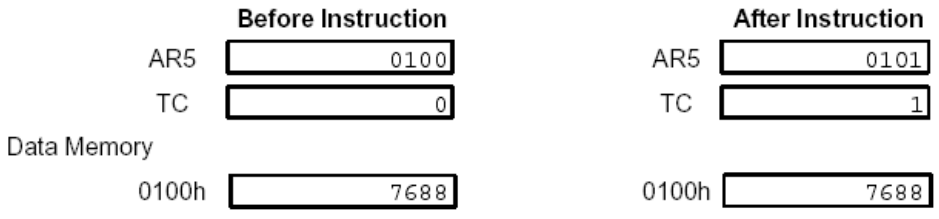
Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 3A (см. страницу 34)

Пример 1:

```
BIT *AR5+, 15-12; test bit 12
```



BITF Тестирование бита, указанного непосредственным значением

Синтаксис: BITF *Smem, #lk*

Операнды: Smem: Одиночный операнд памяти данных
$0 \le lk \le 65\,535$

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	0	1	I	A	A	A	A	A	A	A
16-битная константа															

Выполнение: Если ((Smem) AND lk) = 0
то
0 → TC
иначе
1 → TC

Биты состояния: Оказывает действие на состояние TC

Описание: Эта инструкция тестирует указанный(е) бит(ы) содержимого памяти данных *Smem*. Если указанный(ые) бит(ы) – 0, бит TC регистра состояния ST0 очищается в 0, в противном случае TC устанавливается в 1. Константа *lk* – маска для тестируемого бита или битов.

Слов: 2 слова
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 2 цикла
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 6A (см. страницу 39)
Класс 6B (см. страницу 40)

Пример 1:

```
BITF 5, 00FFh
```

Before Instruction		After Instruction	
TC	x	TC	0
DP	004	DP	004
Data Memory			
0205h	5400	0205h	5400

Пример 2:

```
BITF 5, 0800h
```

Before Instruction		After Instruction	
TC	x	TC	1
DP	004	DP	004
Data Memory			
0205h	0F7F	0205h	0F7F

BITT Тестирование бита, определенного T

Синтаксис: BITT *Smem*

Операнды: Smem: Одиночный операнд памяти данных

Опкод:

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

Выполнение: (Smem(15 – T(3-0))) → TC

Биты состояния: Оказывает действие на состояние TC

Описание: Эта инструкция копирует указанный бит значения памяти данных *Smem* в бит TC регистра состояния ST0. Четыре младших бита регистра T содержат код бита, который определяет – какой бит будет скопирован.
Адрес бита соответствует (15 – T(3-0)). Код бита соответствует содержимому T (3-0).

Коды битов для инструкции BITT

Адрес бита		Код бита	Адрес бита		Код бита
(LSB)	0	1111		8	0111
	1	1110		9	0110
	2	1101		10	0101
	3	1100		11	0100
	4	1011		12	0011
	5	1010		13	0010
	6	1001		14	0001
	7	1000	(MSB)	15	0000

Слов: 1 слово
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 1 цикл
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 3A (см. страницу 34)
Класс 3B (см. страницу 35)

Пример 1:

```
BITT *AR7+0
```

	Before Instruction	After Instruction
T	C	C
TC	0	1
AR0	0008	0008
AR7	0100	0108
Data Memory		
0100h	0008	0008

CALA[D] Вызов подпрограммы по адресу, содержащемуся в аккумуляторе

Синтаксис: CALA[D] *src*

Операнды: src: A (аккумулятор A)
B (аккумулятор B)

Опкод:

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

Выполнение: Без задержки:
(SP) – 1 → SP
(PC) + 1 → TOS
(src(15-0)) → PC

С задержкой:
(SP) – 1 → SP
(PC) + 3 → TOS
(src(15-0)) → PC

Биты состояния: нет

Описание: Эта инструкция передает управление 16-разрядному адресу в младшей части *src* (биты 15-0). При задержанном вызове (когда указан суффикс D), из памяти программ выбираются и исполняются две однословные инструкции или одна двухсловная, расположенные после инструкции вызова.

> Примечание – Эта инструкция неповторяемая.

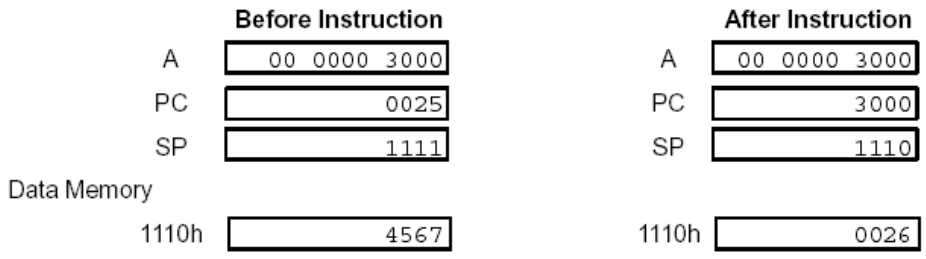
Слов: 1 слово

Циклов: 6 циклов
4 цикла (для задержанных)

Классы: Класс 30B (см. страницу 75)

Пример 1:

```
CALA A
```



CALA[D] Вызов подпрограммы по адресу, содержащемуся в аккумуляторе

Пример 2:

```
CALAD B
ANDM 4444h, *AR1+
```

	Before Instruction		After Instruction
B	00 0000 2000	B	00 0000 2000
PC	0025	PC	2000
SP	1111	SP	1110
Data Memory			
1110h	4567	1110h	0028

После того, как содержимое памяти данных логически умножается на 4444h, программа продолжает выполнение с адреса 2000h.

CALL[D] Безусловный вызов

Синтаксис: CALL[D] *pmad*

Операнды: $0 \leq pmad \leq 65\,535$

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	Z	0	0	1	1	1	0	1	0	0
16-битная константа															

Выполнение: Без задержки:
(SP) – 1 → SP
(PC) + 2 → TOS
pmad → PC

С задержкой:
(SP) – 1 → SP
(PC) + 4 → TOS
pmad → PC

Биты состояния: нет

Описание: Эта инструкция передает управление программой адресу памяти программ (*pmad*). Перед загрузкой *pmad* в PC, адрес возврата помещается в TOS. При задержанном вызове (суффикс D), выбираются и исполняются две однословные инструкции или одна двухсловная, расположенные после инструкции вызова.

> Примечание – Эта инструкция неповторяемая.

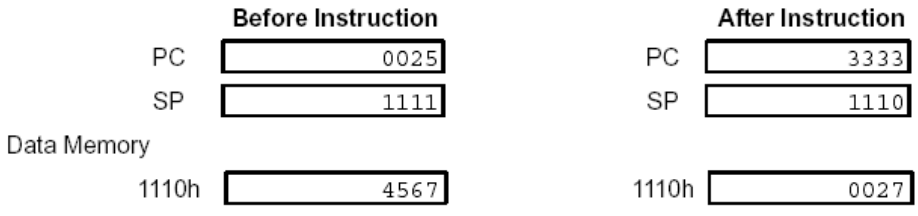
Слов: 2 слова

Циклов: 4 цикла
2 цикла (для задержанных)

Классы: Класс 29B (см. страницу 74)

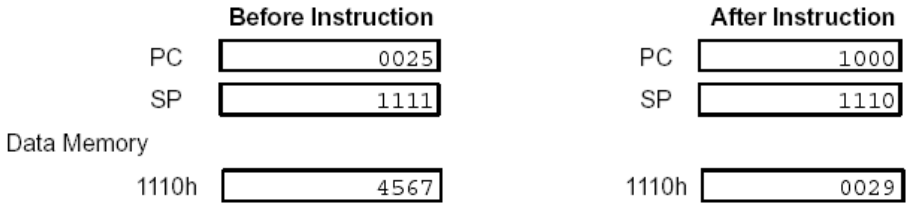
Пример 1:

```
CALL 3333h
```



Пример 2:

```
CALLD 1000h
ANDM #4444h, *AR1+
```



После того, как содержимое памяти данных логически умножается на 4444h, программа продолжает выполнение с адреса 1000h.

CC[D] Условный вызов

Синтаксис: CC[D] *pmad*, *cond*[, *cond*[, *cond*]]

Операнды: $0 \leq pmad \leq 65\,535$

Следующая таблица содержит список условий (cond) для этой инструкции

Условие	Описание	Код условия	Условие	Описание	Код условия
BIO	BIO# low	0000 0011	NBIO	BIO# high	0000 0010
C	C = 1	0000 1100	NC	C = 0	0000 1000
TC	TC = 1	0011 0000	NTC	TC = 0	0010 0000
AEQ	(A) = 0	0100 0101	BEQ	(B) = 0	0100 1101
ANEQ	(A) ≠ 0	0100 0100	BNEQ	(B) ≠ 0	0100 1100
AGT	(A) > 0	0100 0110	BGT	(B) > 0	0100 1110
AGEQ	(A) ≥ 0	0100 0010	BGEQ	(B) ≥ 0	0100 1010
ALT	(A) < 0	0100 0011	BLT	(B) < 0	0100 1011
ALEQ	(A) ≤ 0	0100 0111	BLEQ	(B) ≤ 0	0100 1111
AOV	A переполнен	0111 0000	BOV	B переполнен	0111 1000
ANOV	A не переп-н	0110 0000	BNOV	B не переп-н	0110 1000
UNC	безусловный	0000 0000			

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	Z	1	C	C	C	C	C	C	C	C
16-битная константа															

Выполнение: Без задержки

Если (условие(я)),
то
(SP) – 1 → SP
(PC) + 2 → TOS
pmad → PC
иначе
(PC) + 2 → PC

С задержкой
Если (cond(s))
то
(SP) – 1 → SP
(PC) + 4 → TOS
pmad → PC
иначе
(PC) + 2 → PC

Биты состояния: Затрагивает OVA или OVB, если выбраны OV или NOV

CC[D] Условный вызов

Описание: Эта инструкция передает управление адресу памяти программ (*pmad*), если выполнены все указанные условия. Независимо от выполнения условия(й) из памяти программ выбираются (но не исполняются) две однословные или одна двухсловная инструкция, следующие(ая) за инструкцией вызова. Если указанные условия выполнены, то эти два слова изымаются из конвейера и выполнение продолжается с *pmad*. Если условие(я) невыполнено(ы), PC увеличивается на 2 и исполняются два (уже находящихся в конвейере) слова, распологавшиеся после инструкции вызова.

Если инструкция выполняется с задержкой (при указании суффикса D), то сначала из памяти программ выбираются и исполняются две однословные инструкции или одна двухсловная. Эти инструкции не оказывают влияния на проверяемые условия. Если указанные условия выполняются, то выполнение программы продолжается с *pmad*. Если условия не выполняются, то PC увеличивается на 2, и выполняются два слова следующие за задержанной инструкцией условного вызова.

Эта инструкция позволяет осуществлять проверку нескольких условий перед передачей управления в другую секцию программы. Инструкция может проверять условия индивидуально или в комбинации с другими условиями. Можно компоновать условия только из одной группы следующим образом:

Группа 1: Допускается выбрать до двух условий. Каждое условие должно принадлежать разным категориям (A или B, см. таблицу ниже). Не допускается использование в инструкции двух условий из одной категории. Например, одновременно можно тестировать EQ и OV, однако, нельзя выполнить одновременную проверку GT и NEQ. Для обоих условий аккумулятор должен быть одним и тем же. Не допускается проверка двух условий для разных аккумуляторов в одной инструкции. Например, разрешена одновременная проверка AGT и AOV, однако, не допускается использование в одной инструкции условий AGT и BOV.

Группа 2: Допускается выбрать до трех условий. Каждое из трех условий должно быть из различных категорий (A, B или C). Не допускается использование двух или более условий из одной категории, т. е. допускается одновременная проверка TC, C и BIO, однако, не допускается одновременная проверка NTC, C и NC.

CC[D] Условный вызов

Сочетаемость условий для этой инструкции

Группа 1		Группа 2		
Категория A	Категория B	Категория A	Категория B	Категория C
EQ	OV	TC	C	BIO
NEQ	NOV	NTC	NC	NBIO
LT				
LEQ				
GT				
GEQ				

Примечание – Эта инструкция неповторяемая.

Слов: 2 слова

Циклов: 5 циклов (условие выполнено)
3 цикла (условие не выполнено)
3 цикла (для задержанных)

Классы: Класс 31B (см. страницу 77)

Пример 1:

```
CC 2222h, AGT
```

	Before Instruction		After Instruction
A	00 0000 3000	A	00 0000 3000
PC	0025	PC	2222
SP	1111	SP	1110
Data Memory			
1110h	4567	1110h	0027

Пример 2:

```
CCD 1000h, BOV
ANDM 4444h, *AR1+
```

	Before Instruction		After Instruction
PC	0025	PC	1000
OVB	1	OVB	0
SP	1111	SP	1110
Data Memory			
1110h	4567	1110h	0029

После того, как содержимое памяти данных логически умножается на 4444h, программа продолжает выполнение с адреса 1000h.

CMPL Двоичное дополнение аккумулятора

Синтаксис: CMPL *src*, [, *dst*]

Операнды: src, dst: A (аккумулятор A)
B (аккумулятор B)

Опкод:

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

Выполнение: $(\overline{src}) \rightarrow dst$

Биты состояния: нет

Описание: Эта инструкция вычисляет дополнение до единицы содержимого *src* (логическая инверсия). Результат сохраняется в *dst*, если он определен, иначе в *src*.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 1 (см. страницу 32)

Пример:

```
CMPL A, B
```

	Before Instruction		After Instruction
A	FC DFFA AEAA	A	FC DFFA AEAA
B	00 0000 7899	B	03 2005 5155

CMPM Сравнение памяти с длинным непосредственным значением

Синтаксис: CMPM *Smem, #lk*

Операнды: Smem: Одиночный операнд памяти данных
-32 768 ≤ lk ≤ 32 767

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	0	0	I	A	A	A	A	A	A	A
16-битная константа															

Выполнение: Если (Smem) = lk
то
1 → TC
иначе
0 → TC

Биты состояния: оказывает действие на состояние TC

Описание: Эта инструкция сравнивает 16-разрядный одиночный операнд памяти данных *Smem* с 16-разрядной константой *lk*. Если они равны, то TC устанавливается в 1, иначе – очищается в 0.

Слов: 2 слова
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 2 цикла
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 6A (см. страницу 39)
Класс 6B (см. страницу 40)

Пример

```
CMPM *AR4+, 0404h
```

	Before Instruction		After Instruction
TC	1	TC	0
AR4	0100	AR4	0101
Data Memory			
0100h	4444	0100h	4444

CMPR Сравнение вспомогательного регистра с AR0

Синтаксис: CMPR *CC, ARx*

Операнды: $0 \leq CC \leq 3$
ARx: AR0-AR7

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	C	C	1	0	1	0	1	A	R	X

Выполнение: Если (условие)
то
$1 \rightarrow TC$
иначе
$0 \rightarrow TC$

Биты состояния: оказывает действие на состояние TC

Описание: Эта инструкция сравнивает содержимое указанного вспомогательного регистра (*ARx*) с содержимым AR0 и устанавливает TC в соответствие с результатом сравнения. Сравнение определяется значением *CC* (код условия) (см. таблицу ниже). Если условие выполнено, то TC устанавливается в 1. Если условие не выполняется, то TC очищается в 0. Вычисление всех условий осуществляется как беззнакковая операция.

Условие	Код условия (CC)	Описание
EQ	00	Проверка (ARx) = (AR0)
LT	01	Проверка (ARx) < (AR0)
GT	10	Проверка (ARx) > (AR0)
NEQ	11	Проверка (ARx) ≠ (AR0)

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 1 (см. страницу 32)

Пример

```
CMPR 2, AR4
```

	Before Instruction		After Instruction
TC	1	TC	0
AR0	FFFF	AR0	FFFF
AR4	7FFF	AR4	7FFF

CMPS Сравнение, выборка и сохранение максимума

Синтаксис: CMPS *src, Smem*

Операнды:
src: A (аккумулятор A)
B (аккумулятор B)
Smem: Одиночный операнд памяти данных

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	1	1	S	I	A	A	A	A	A	A	A

Выполнение:
Если ((src(31-16)) > (src(15-0)))
то
(src(31-16)) → Smem
(TRN) << 1 → TRN
0 → TRN(0)
0 → TC
иначе
(src(15-0)) → Smem
(TRN) << 1 → TRN
1 → TRN(0)
1 → TC

Биты состояния: оказывает действие на состояние TC

Описание: Эта инструкция сравнивает значения двух величин, расположенных в младшей и старшей частях *src* и сохраняет максимальное значение в указанную ячейку памяти данных *Smem*. Если старшая часть *src* (биты 31-16) больше, ноль сдвигается в младший значащий бит (LSB) регистра перемещений (TRN) и бит TC очищается в 0. Если же из двух чисел больше значение в младшей части *src* (биты 15-0), то 1 сдвигается в LSB регистра перемещений и TC устанавливается в 1.
Эта инструкция имеет особенность при конвейерном выполнении. Сравнение выполняется в фазе чтения, поэтому значение *src* должено быть определено за один цикл до выполняемой операции. Иными словами, тестирование аккумулятора *src* с помощью инструкции CMPS будет некорректным, если проверять результат выполнения предыдущей инструкции, изменившей содержимое этого аккумулятора.
TRN и бит TC обновляются во время фазы выполнения.

Слов: 1 слово

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 1 цикл

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

CMPS Сравнение, выборка и сохранение максимума

Классы: Класс 10A (см. страницу 44)
Класс 10B (см. страницу 44)

Пример:

```
CMPS A, *AR4+
```

	Before Instruction	After Instruction
A	00 2345 7899	00 2345 7899
TC	0	1
AR4	0100	0101
TRN	4444	8889
Data Memory		
0100h	0000	7899

DADD Двойное 16-разрядное/с двойной точностью сложение с аккумулятором

Синтаксис: DADD *Lmem, src*[, *dst*]

Операнды:
Lmem: Длинный операнд памяти данных
src, dst: A (аккумулятор A)
B (аккумулятор B)

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	0	S	D	I	A	A	A	A	A	A	A

Выполнение:
Если C16 = 0
то
(Lmem) + (src) → dst
иначе
(Lmem(31-16)) + (src(31-16)) → dst(39-16)
(Lmem(15-0) + (src(15-0)) → dst(15-0)

Биты состояния: зависит от SXM и OVM (только, если C16 = 0)
оказывает влияние на C и OVdst (или OVsrc, если dst не определен)

Описание: Эта инструкция добавляет содержимое *src* к 32-разрядному длинному операнду памяти данных *Lmem*. Если *dst* определен, то результат сохраняется в *dst*. Если *dst* не определен, то результат сохраняется в *src*. Значение бита C16 определяет режим работы инструкции:

– Если C16 = 0, инструкция выполняется в режиме вычисления с двойной точностью. 40-разрядное значение *src* добавляется к *Lmem*. Биты насыщения и переполнения устанавливаются в соответствии с результатом операции.

– Если C16 = 1, инструкция выполняется в двойном 16-разрядном режиме. Старшая часть *src* (биты 31-16) складывается с 16 старшими битами *Lmem*, а младшие 16 бит *src*(15-0) с 16 младшими битами *Lmem*(15-0). Биты переполнения и насыщения не затрагиваются при таком режиме работы инструкции. В этом режиме результат не насыщается, независимо от состояния бита OVM.

Слов: 1 слово

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem добавляется еще 1 слово.

Циклов: 1 цикл

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem добавляется еще 1 цикл.

DADD Двойное 16-разрядное/с двойной точностью сложение с аккумулятором

Классы: Класс 9A (см. страницу 43)
Класс 9B (см. страницу 43)

Пример 1:

```
DADD *AR3+, A, B
```

	Before Instruction	After Instruction
A	00 5678 8933	00 5678 8933
B	00 0000 0000	00 6BAC BD89
C16	0	0
AR3	0100	AR3[1)] 0102
Data Memory		
0100h	1534	1534
0101h	3456	3456

[1)] Поскольку эта инструкция – длннооперандная, AR3 увеличивается на 2 после выполнения

Пример 2:

```
DADD *AR3-, A, B
```

	Before Instruction	After Instruction
A	00 5678 3933	00 5678 3933
B	00 0000 0000	00 6BAC 6D89
C16	1	1
AR3	0100	AR3[1)] 00FE
Data Memory		
0100h	1534	1534
0101h	3456	3456

[1)] Поскольку эта инструкция – длннооперандная, AR3 уменьшается на 2 после выполнения

Пример 3:

```
DADD *AR3-, A, B
```

	Before Instruction	After Instruction
A	00 5678 3933	00 5678 3933
B	00 0000 0000	00 8ACE 4E67
C16	0	0
AR3	0101	AR3[1)] 00FF
Data Memory		
0100h	1534	1534
0101h	3456	3456

[1)] Поскольку эта инструкция – длннооперандная, AR3 уменьшается на 2 после выполнения

DADST Сложение с T с двойной точностью/двойное 16-разр. сложение/вычитание с T

Синтаксис: DADST *Lmem, dst*

Операнды:
Lmem: Длинный операнд памяти данных
dst: A (аккумулятор A)
B (аккумулятор B)

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	1	0	1	D	I	A	A	A	A	A	A	A

Выполнение:
Если C16 = 1
то
(Lmem(31-16)) + (T) → dst(39-16)
(Lmem(15-0) – (T) → dst(15-0)
иначе
(Lmem) + ((T) + (T) << 16) → dst

Биты состояния: зависит от SXM и OVM (только, если C16 = 0)
оказывает влияние на C и OVdst

Описание: Эта инструкция добавляет содержимое T к 32-разрядному длинному операнду памяти данных *Lmem*. Значение бита C16 определяет режим работы инструкции:

– Если C16 = 0, инструкция выполняется в режиме вычисления с двойной точностью. *Lmem* добавляется к 32-разрядному значению, состоящему из содержимого T, конкатенированного с содержимым T, сдвинутым на 16 разрядов влево (T << 16 + T). Результат сохраняется в *dst*.

– Если C16 = 1, инструкция выполняется в двойном 16-разрядном режиме. 16 старших бит *Lmem* складываются с содержимым T и результат сохраняется в 24 старших разрядах *dst* (39-16). В то же время содержимое T вычитается из 16 младших бит *Lmem*. Результат сохраняется в 16 младших битах *dst*. В этом режиме результат не насыщается, независимо от состояния бита OVM.

Примечание – Эта инструкция имеет смысл только, когда бит C16 установлен в 1 (двойной 16-разрядный режим).

Слов: 1 слово
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem добавляется еще 1 слово.

Циклов: 1 цикл
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem добавляется еще 1 цикл

DADST Сложение с T с двойной точностью/двойное 16-разр. сложение/вычитание с T

Классы: Класс 9A (см. страницу 43)
Класс 9B (см. страницу 43)

Пример 1:

```
DADST *AR3-, A
```

	Before Instruction		After Instruction
A	00 0000 0000	A	00 3879 1111
T	2345	T	2345
C16	1	C16	1
AR3	0100	AR3[1)]	00FE
Data Memory			
0100h	1534	0100h	1534
0101h	3456	0101h	3456

[1)] Поскольку эта инструкция – длнноперандная, AR3 уменьшается на 2 после выполнения

Пример 2:

```
DADST *AR3+, A
```

	Before Instruction		After Instruction
A	00 0000 0000	A	00 3879 579B
T	2345	T	2345
C16	0	C16	0
AR3	0100	AR3[1)]	0102
Data Memory			
0100h	1534	0100h	1534
0101h	3456	0101h	3456

[1)] Поскольку эта инструкция – длнноперандная, AR3 увеличивается на 2 после выполнения

DELAY Задержка памяти

Синтаксис: DELAY *Smem*

Операнды: Smem: Одиночный операнд памяти данных

Опкод:

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

Выполнение: (Smem) → Smem + 1

Биты состояния: нет

Описание: Эта инструкция копирует содержимое одиночного операнда памяти данных *Smem* в следующий адрес, стоящий выше. После операции копирования само содержимое Smem остается прежним. Эта инструкция используется для реализации Z-задержки в приложениях цифровой обработки сигналов. Функция задержки включена в инструкции загрузки T и вставки задержки (LTD) и в инструкции умножения памяти программ и накопления с задержкой (MACD).

Слов: 1 слово

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 1 цикл

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 24A (см. страницу 67)
Класс 24B (см. страницу 67)

Пример:

```
DELAY *AR3
```

	Before Instruction	After Instruction
AR3	0100	0100
Data Memory		
0100h	6CAC	6CAC
0101h	0000	6CAC

DLD Двойная 16-разрядная/с двойной точностью загрузка длинного слова в аккумулятор

Синтаксис: DLD *Lmem, dst*

Операнды:
Lmem: Длинный операнд памяти данных
dst: A (аккумулятор A)
B (аккумулятор B)

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	1	1	D	I	A	A	A	A	A	A	A

Выполнение:
Если C16 = 0
то
(Lmem) → dst
иначе
(Lmem(31-16)) → dst(39-16)
(Lmem(15-0)) → dst(15-0)

Биты состояния: зависит от SXM

Описание: Эта инструкция загружает *dst* 32-разрядным длинным операндом *Lmem*. Значение бита C16 определяет режим работы инструкции:

– Если C16 = 0, инструкция выполняется в режиме с двойной точностью. *Lmem* загружается в *dst*.

– Если C16 = 1, инструкция выполняется в двойном 16-разрядном режиме. 16 старших бит *Lmem* загружаются в 24 разряда *dst*(39-16). Одновременно 16 младших бит *Lmem* загружаются в 16 младших бит *dst*.

Слов: 1 слово
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem добавляется еще 1 слово.

Циклов: 1 цикл
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem добавляется еще 1 цикл.

Классы: Класс 9A (см. страницу 43)
Класс 9B (см. страницу 43)

Пример:

```
DLD *AR3+, B
```

	Before Instruction		After Instruction
B	00 0000 0000	B	00 6CAC BD90
AR3	0100	AR3[1)]	0102
Data Memory			
0100h	6CAC	0100h	6CAC
0101h	BD90	0101h	BD90

[1)] Поскольку эта инструкция – длнноoперандная, AR3 увеличивается на 2 после выполнения

DRSUB Двойное 16-разрядное/с двойной точностью вычитание из длинного слова

Синтаксис: DRSUB *Lmem, dst*

Операнды:
Lmem: Длинный операнд памяти данных
src: A (аккумулятор A)
B (аккумулятор B)

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	1	0	0	S	I	A	A	A	A	A	A	A

Выполнение:
Если C16 = 0
то
(Lmem) – (src) → src
иначе
(Lmem(31-16)) – (src(31-16)) → src(39-16)
(Lmem(15-0)) – (src(15-0)) → src(15-0)

Биты состояния: зависит от SXM и OVM (только, если C16 = 0)
Оказывает действие на C и OVsrc

Описание: Эта инструкция вычитает содержимое *src* из 32-разрядного длинного операнда памяти данных *Lmem*; результат сохраняется в *src*. Значение бита C16 определяет режим работы инструкции:

– Если C16 = 0, инструкция выполняется в режиме вычисления с двойной точностью. Содержимое *src*(32 бита) вычитается из *Lmem*. Результат сохраняется в *src*.

– Если C16 = 1, инструкция выполняется в двойном 16-разрядном режиме. Старшая часть *src*(31-16) вычитается из 16 старших разрядов *Lmem* и результат сохраняется в старшую часть *src*(39-16). В то же самое время, младшие 16 бит *src* вычитаются из 16 младших бит *Lmem*. Результат сохраняется в младшей части *src*(15-0). В этом режиме результат не насыщается, независимо от состояния бита OVM.

Слов: 1 слово

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem добавляется еще 1 слово.

Циклов: 1 цикл

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem добавляется еще 1 цикл.

Классы:
Класс 9A (см. страницу 43)
Класс 9B (см. страницу 43)

DRSUB Двойное 16-разрядное/с двойной точностью вычитание из длинного слова

Пример 1:

```
DRSUB *AR3+, A
```

	Before Instruction		After Instruction
A	00 5678 8933	A	FF BEBB AB23
C	x	C	0
C16	0	C16	0
AR3	0100	AR3 [1)]	0102
Data Memory			
0100h	1534	0100h	1534
0101h	3456	0101h	3456

[1)] Поскольку эта инструкция – длнноoперандная, AR3 увеличивается на 2 после выполнения

Пример 2:

```
DRSUB *AR3-, A
```

	Before Instruction		After Instruction
A	00 5678 3933	A	FF BEBC FB23
C	1	C	0
C16	1	C16	1
AR3	0100	AR3 [1)]	00FE
Data Memory			
0100h	1534	0100h	1534
0101h	3456	0101h	3456

[1)] Поскольку эта инструкция – длнноoперандная, AR3 уменьшается на 2 после выполнения

DSADT Загрузка длинного слова со сложением T/двойная 16-разрядная загрузка со сложением/вычитанием T

Синтаксис: DSADT *Lmem, dst*

Операнды:
Lmem: Длинный операнд памяти данных
dst: A (аккумулятор A)
B (аккумулятор B)

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	1	1	1	D	I	A	A	A	A	A	A	A

Выполнение:
Если C16 = 1
то
(Lmem(31-16)) – (T) → dst(39-16)
(Lmem(15-0)) + (T) → dst(15-0)
иначе
(Lmem) – ((T) + (T << 16)) → dst

Биты состояния: зависит от SXM и OVM (только, если C16 = 0)
Оказывает действие на C и OVdst

Описание: Эта инструкция вычитает/складывает содержимое T с 32-разрядным длинным операндом памяти данных Lmem и сохраняет результат в dst. Значение бита C16 определяет режим работы инструкции:

– Если C16 = 0, инструкция выполняется в режиме вычисления с двойной точностью. 32-разрядное значение, состоящее из содержимого T, конкатенированного с содержимым T, сдвинутым на 16 разрядов влево (T << 16 + T), вычитается из *Lmem*. Результат сохраняется в *dst*.

– Если C16 = 1, инструкция выполняется в двойном 16-разрядном режиме. Содержимое T вычитается из 16 старших разрядов *Lmem* и результат сохраняется в старшей части *dst* (биты 39-16). В то же самое время, содержимое T складывается с 16 младшими битами *Lmem* и результат сохраняется в младшей части *dst* (биты 15-0). В этом режиме результат не насыщается, независимо от состояния бита OVM.

Примечание – Эта инструкция имеет смысл только, когда бит C16 установлен на 1 (двойной 16-разрядный режим).

Слов: 1 слово

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem добавляется еще 1 слово.

Циклов: 1 цикл

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem добавляется еще 1 цикл.

DSADT Загрузка длинного слова со сложением T/двойная 16-разрядная загрузка со сложением/вычитанием T

Классы: Класс 9A (см. страницу 43)
Класс 9B (см. страницу 43)

Пример 1:

```
DSADT *AR3+, A
```

	Before Instruction		After Instruction
A	00 0000 0000	A	FF F1EF 1111
T	2345	T	2345
C	0	C	0
C16	0	C16	0
AR3	0100	AR3[1)]	0102
Data Memory			
0100h	1534	0100h	1534
0101h	3456	0101h	3456

[1)] Поскольку эта инструкция – длннооперандная, AR3 увеличивается на 2 после выполнения

Пример 2:

```
DSADT *AR3-, A
```

	Before Instruction		After Instruction
A	00 0000 0000	A	FF F1EF 579B
T	2345	T	2345
C	0	C	1
C16	1	C16	1
AR3	0100	AR3[1)]	00FE
Data Memory			
0100h	1534	0100h	1534
0101h	3456	0101h	3456

[1)] Поскольку эта инструкция – длннооперандная, AR3 уменьшается на 2 после выполнения

DST Сохранение аккумулятора в длинном слове

Синтаксис: DST *src*, *Lmem*

Операнды: src: A (аккумулятор A)
B (аккумулятор B)
Lmem: Длинный операнд памяти данных

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	S	I	A	A	A	A	A	A	A

Выполнение: (src(31-0)) → Lmem

Биты состояния: нет

Описание: Эта инструкция сохраняет содержимое *src* в 32-разрядном длинном операнде памяти данных *Lmem*.

Слов: 1 слово
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 1 цикл
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 13A (см. страницу 47)
Класс 13B (см. страницу 47)

Пример 1:

```
DST B, *AR3+
```

	Before Instruction		After Instruction
B	00 6CAC BD90	B	00 6CAC BD90
AR3	0100	AR3[1)]	0102
Data Memory			
0100h	0000	0100h	6CAC
0101h	0000	0101h	BD90

[1)] Поскольку эта инструкция – длнноoперандная, AR3 увеличивается на 2 после выполнения

Пример 2:

```
DST B, *AR3-
```

	Before Instruction		After Instruction
B	00 6CAC BD90	B	00 6CAC BD90
AR3	0101	AR3[1)]	00FF
Data Memory			
0100h	0000	0100h	BD90
0101h	0000	0101h	6CAC

[1)] Поскольку эта инструкция – длнноoперандная, AR3 уменьшается на 2 после выполнения

DSUB Двойное 16-разрядное/с двойной точностью вычитание из аккумулятора

Синтаксис: DSUB *Lmem, src*

Операнды:
Lmem: Длинный операнд памяти данных
src: A (аккумулятор A)
B (аккумулятор B)

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	1	0	S	I	A	A	A	A	A	A	A

Выполнение:
Если C16 = 0
то
(src) – (Lmem) → src
иначе
(src(31-16)) – (Lmem(31-16)) → src(39-16)
(src(15-0)) – (Lmem(15-0)) → src(15-0)

Биты состояния: зависит от SXM и OVM (только, если C16 = 0)
Оказывает действие на C и OVsrc

Описание: Эта инструкция вычитает значение 32-разрядного операнда памяти данных *Lmem* из содержимого *src* и сохраняет результат в *src*. Значение бита C16 определяет режим работы инструкции:

– Если C16 = 0, инструкция выполняется в режиме вычисления с двойной точностью. *Lmem* вычитается из содержимого *src*.

– Если C16 = 1, инструкция выполняется в двойном 16-разрядном режиме. 16 старших бит *Lmem* вычитаются из старшей части *src* (биты 31-16) и результат сохраняется в старшей части *src*(39-16). В то же самое время, 16 младших бит *Lmem* вычитаются из младшей части *src* (биты 15-0) и результат сохраняется в младшей части *src*(15-0).

Слов: 1 слово

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem добавляется еще 1 слово.

Циклов: 1 цикл

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem добавляется еще 1 цикл.

Классы: Класс 9A (см. страницу 43)
Класс 9B (см. страницу 43)

DSUB Двойное 16-разрядное/с двойной точностью вычитание из аккумулятора

Пример 1:

```
DSUB *AR3+, A
```

	Before Instruction		After Instruction
A	00 5678 8933	A	00 4144 54DD
C16	0	C16	0
AR3	0100	AR3[1)]	0102
Data Memory			
0100h	1534	0100h	1534
0101h	3456	0101h	3456

[1)] Поскольку эта инструкция – длнноoперандная, AR3 увеличивается на 2 после выполнения

Пример 2:

```
DSUB *AR3-, A
```

	Before Instruction		After Instruction
A	00 5678 3933	A	00 4144 04DD
C	1	C	1
C16	1	C16	1
AR3	0100	AR3[1)]	00FE
Data Memory			
0100h	1534	0100h	1534
0101h	3456	0101h	3456

[1)] Поскольку эта инструкция – длнноoперандная, AR3 уменьшается на 2 после выполнения

DSUBT Загрузка длинным словом с вычитанием T/двойная 16-разрядная загрузка с вычитанием T

Синтаксис: DSUBT *Lmem, dst*

Операнды:
Lmem: Длинный операнд памяти данных
dst: A (аккумулятор A)
B (аккумулятор B)

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	1	1	0	D	I	A	A	A	A	A	A	A

Выполнение:
Если C16 = 1
то
(Lmem(31-16)) – (T) → dst(39-16)
(Lmem(15-0)) – (T) → dst(15-0)
иначе
(Lmem) – ((T) + (T << 16) → dst

Биты состояния: зависит от SXM и OVM (только, если C16 = 0)
Оказывает действие на C и OVdst

Описание: Эта инструкция вычитает содержимое T из 32-разрядного длинного операнда памяти данных *Lmem* и сохраняет результат в *dst*. Значение бита C16 определяет режим работы инструкции:

– Если C16 = 0, инструкция выполняется в режиме вычисления с двойной точностью. 32-разрядное значение, состоящее из содержимого T, конкатенированного с содержимым T, сдвинутым на 16 разрядов влево (T << 16 + T), вычитается из *Lmem*. Результат сохраняется в *dst*.

– Если C16 = 1, инструкция выполняется в двойном 16-разрядном режиме. Содержимое T вычитается из 16 старших бит *Lmem* и результат сохраняется в старшую часть dst (биты 39-16). В то же самое время содержимое T вычитается из 16 младших битов *Lmem* и результат заносится в младшую часть dst(15-0). В этом режиме результат не насыщается, независимо от состояния бита OVM.

Примечание – Эта инструкция имеет смысл только, когда бит C16 установлен на 1 (двойной 16-разрядный режим).

Слов: 1 слово

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem добавляется еще 1 слово.

Циклов: 1 цикл

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem добавляется еще 1 цикл.

DSUBT Загрузка длинным словом с вычитанием T/двойная 16-разрядная загрузка с вычитанием T

Классы: Класс 9A (см. страницу 43)
Класс 9B (см. страницу 43)

Пример 1:

```
DSUBT *AR3+, A
```

	Before Instruction		After Instruction
A	00 0000 0000	A	FF F1EF 1111
T	2345	T	2345
C16	0	C16	0
AR3	0100	AR3[1)]	0102
Data Memory			
0100h	1534	0100h	1534
0101h	3456	0101h	3456

[1)] Поскольку эта инструкция – длнноoперандная, AR3 увеличивается на 2 после выполнения

Пример 2:

```
DSUBT *AR3-, A
```

	Before Instruction		After Instruction
A	00 0000 0000	A	FF F1EF 1111
T	2345	T	2345
C16	1	C16	1
AR3	0100	AR3[1)]	00FE
Data Memory			
0100h	1534	0100h	1534
0101h	3456	0101h	3456

[1)] Поскольку эта инструкция – длнноoперандная, AR3 уменьшается на 2 после выполнения

EXP Экспонента аккумулятора

Синтаксис: EXP *src*

Операнды: src: A (аккумулятор A)
B (аккумулятор B)

Опкод:

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

Выполнение:
Если (src) = 0
то
0 → T
иначе
(число главных бит src) – 8 → T

Биты состояния: нет

Описание: Эта инструкция вычисляет значение экспоненты, которое является знаковым двоичным дополнением в интервале от – 8 до 31 и сохраняет результат в T. Экспонента вычисляется путем подсчета количества главных (или "смысловых", не содержащих знакового расширения) разрядов в *src* и вычитанием 8 из этого числа. Число главных разрядов эквивалентно количеству левых сдвигов, необходимых для изъятия знаковых разрядов в 40-разрядном *src* за исключением бита знака. Само содержимое *src* не изменяется после выполнения этой инструкции.
В результате вычитания 8 из количества главных разрядов получается отрицательная степень содержимого аккумулятора, имеющего значащие разряды в поле разрядов безопасности (восемь старших разрядов аккумулятора, используемых для обнаружения и исправления ошибок). См также описание инструкции NORM (нормализация, страница 187).

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 1 (см. страницу 32)

Пример 1:

```
EXP A
```

	Before Instruction		After Instruction	
A	FF FFFF FFCB	-53	FF FFFF FFCB	-53
T	0000		0019	25

Пример 2:

```
EXP B
```

	Before Instruction	After Instruction	
B	07 8543 2105	07 8543 2105	
T	FFFC	FFFC	-4[1)]

[1)] Число в аккумуляторе B имеет значащие разряды в поле разрядов безопасности, поэтому результатом операции EXP является отрицательная степень.

FIRS Симметричный КИХ фильтр

Синтаксис: FIRS *Xmem, Ymem, pmad*

Операнды: Xmem, Ymem: Двойные операнды памяти данных
$0 \leq pmad \leq 65535$

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	0	0	0	0	X	X	X	X	Y	Y	Y	Y
16-битная константа															

Выполнение:
pmad → B
Пока (RC) ≠ 0
(B) + (A(32-16)) x (Pmem по адресу из PAR) → B
((Xmem) + (Ymem)) << 16 → A
(PAR) + 1 → PAR
(RC) – 1 → RC

Биты состояния: зависит от SXM, FRCT и OVM
Оказывает влияние на C, OVA и OVB

Описание: Эта инструкция реализует фильтр с конечной импульсной характеристикой (КИХ). Аккумулятор A(биты 32-16) умножается на содержимое ячейки Pmem, адресуемой *pmad* (в регистре программного адреса PAR) и результат складывается с содержимым аккумулятора B. Одновременно, содержимое *Xmem* складывает с содержимым *Ymem*; полученный результат сдвигается влево на 16 разрядов и загружается в аккумулятор A. В следующей итерации *pmad* увеличивается на 1. В режиме повтора, после загрузки инструкции в конвейер (первое выполнение), она становится эффективно одноцикловой.

Слов: 2 слова

Циклов: 3 цикла

Классы: Класс 8 (см. страницу 42)

Пример:

```
FIRS *AR3+, *AR4+, COEFFS
```

	Before Instruction	After Instruction
A	00 0077 0000	00 00FF 0000
B	00 0000 0000	00 0008 762C
FRCT	0	0
AR3	0100	0101
AR4	0200	0201
Data Memory		
0100h	0055	0055
0200h	00AA	00AA
Program Memory		
COEFFS	1234	1234

FRAME Смещение указателя стека непосредственным операндом

Синтаксис: FRAME *K*

Операнды: $-128 \le K \le 127$

Опкод:

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

Выполнение: (SP) + K → SP

Биты состояния: нет

Описание: Эта инструкция добавляет к SP смещение в формате короткого непосредственного операнда K. Для этой инструкции не требуется никакого времени ожидания для генерации адреса в режиме компиляции (CPL = 1) или для манипуляций над стеком осуществляемых, последующей инструкцией.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 1 (см. страницу 32)

Пример:

```
FRAME 10h
```

	Before Instruction		After Instruction
SP	1000	SP	1010

IDLE Переход в режим ожидания прерывания

Синтаксис: IDLE *K*

Операнды: $1 \leq K \leq 3$

Опкод:

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

Если K:	NN равно
1	00
2	10
3	01

Выполнение: (PC) + 1 → PC

Биты состояния: Оказывает действие на INTM

Описание: Эта инструкция вмешивается в работу программы и переводит ее в режим ожидания немаскированного прерывания или сброса. PC при этом увеличивается на 1. До прихода прерывания устройство остается в режиме ожидания (режим пониженной мощности).

Даже, если INTM = 1, устройство выходит из состояния ожидания после немаскированного прерывания. Если INTM = 1, программа продолжит выполнение с инструкции, следующей за инструкцией ожидания. Если INTM = 0, программа выполняет переход к подпрограмме, обслуживающей соответствующее прерывание. Прерывание разрешается регистром маски прерывания (IMR), независимо от значения бита INTM. Следующие опции, зависящие от значения K, определяют тип прерывания, которое может вывести устройство из состояния ожидания:

K = 1 – Внешние устройства, такие как таймер и последовательные порты, остаются активными. Процессор выводится из режима ожидания прерываниями от периферии, а так же сбросом и внешними прерываниями.

K = 2 – Внешние устройства, такие как таймер и последовательные порты, неактивны. Процессор из режима ожидания выводится сбросом или внешним прерыванием. Поскольку в отличие от нормальной работы устройства, в режиме ожидания прерывания не защелкиваются, они должны быть короче количества циклов для необходимых для подтверждения.

K = 3 – Внешние устройства, такие как таймер и последовательные порты, неактивны; система ФАПЧ остановлена. Процессор из режима ожидания выводится сбросом или внешним прерыванием. Поскольку в отличие от нормальной работы устройства, в режиме ожидания прерывания не защелкиваются, они должны быть короче количества циклов для необходимых для подтверждения.

IDLE Переход в режим ожидания прерывания

Примечание – Эта инструкция неповторяемая.

Слов: 1 слово

Циклов: Число циклов, необходимых для выполнения этой инструкции, зависит от периода ожидания. Поскольку при K = 3 все остановлено, количество циклов в этом случае не может быть определено. Минимальное количество циклов 4.

Классы: Класс 36 (см. страницу 79)

Пример 1:
```
IDLE 1
```
Процессор простаивает (находится в состоянии ожидания) до сброса или немаскированного прерывания.

Пример 2:
```
IDLE 2
```
Процессор простаивает (находится в состоянии ожидания) до сброса или немаскированного внешнего прерывания.

Пример 3:
```
IDLE 3
```
Процессор простаивает (находится в состоянии ожидания) до сброса или немаскированного внешнего прерывания.

INTR Программное прерывание

Синтаксис: INTR *K*

Операнды: $0 \le K \le 31$

Опкод:

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

Выполнение:
(SP) – 1 → SP
(PC) + 1 → TOS
вектор прерывания, определяемый K → PC
1 → INTM

Биты состояния: Оказывает действие на INTM и IFR

Описание: Эта инструкция передает управление программой вектору прерывания, определяемому K. Она позволяет использовать программное обеспечение для выполнения прерывания. Список прерываний и соответствующие им значения K приведены в КФДЛ.431299.010 процессора 1867ВЦ4Т. В ходе выполнения инструкции содержимое PC увеличивается на 1 и вталкивается в вершину стека (TOS). После этого, вектор прерывания, определяемый K, загружается в PC и начинает выполняться подпрограмма обработки прерываний (ISR). Соответствующий бит в регистре флага прерываний (IFR) очищается и прерывания глобально запрещаются (INTM = 1). Регистр маски прерывания (IMR) не влияет на инструкцию INTR. INTR выполняется независимо от значения INTM.

Примечание – Эта инструкция неповторяемая.

Слов: 1 слово

Циклов: 3 цикла

Классы: Класс 35 (см. страницу 79)

Пример:

```
INTR 3
```

	Before Instruction	After Instruction
PC	0025	FF8C
INTM	0	1
IPTR	01FF	01FF
SP	1000	0FFF
Data Memory		
0FFFh	9653	0026

LD Загрузка аккумулятора со сдвигом

Синтаксис:

1: LD *Smem, dst*
2: LD *Smem*, TS, *dst*
3: LD *Smem*, 16, *dst*
4: LD *Smem*[, *SHIFT*], *dst*
5: LD *Xmem, SHFT, dst*
6: LD *#K, dst*
7: LD *#lk*[, *SHFT*], *dst*
8: LD *#lk*, 16, *dst*
9: LD *src*, ASM[, *dst*]
10: LD *src*[, *SHIFT*], *dst*

Загрузка других регистров – см. *Загрузка T/DP/ASM/ARP*, стр. 140.

Операнды:

Smem:	Одиночный операнд памяти данных
Xmem	Двойной операнд памяти данных
src, dst:	A (аккумулятор A)
	B (аккумулятор B)

$0 \le K \le 255$
$-32\,768 \le lk \le 32\,767$
$-16 \le SHIFT \le 15$
$0 \le SHFT \le 15$

Опкоды:

1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	0	D	I	A	A	A	A	A	A	A

2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	1	0	D	I	A	A	A	A	A	A	A

3:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	1	0	D	I	A	A	A	A	A	A	A

4:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	1	1	1	I	A	A	A	A	A	A	A
0	0	0	0	1	1	0	D	0	1	0	S	H	I	F	T

5:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	0	1	0	D	X	X	X	X	S	H	F	T

6:

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

7:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	D	0	0	1	0	S	H	F	T
16-битная константа															

8:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	D	0	1	1	0	0	0	1	0
16-битная константа															

LD Загрузка аккумулятора со сдвигом

9:

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

10:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	S	D	0	1	0	S	H	I	F	T

Выполнение:

1: (Smem) → dst
2: (Smem) << TS → dst
3: (Smem) << 16 → dst
4: (Smem) << SHIFT → dst
5: (Xmem) << SHFT → dst
6: K → dst
7: lk << SHFT → dst
8: lk << 16 → dst
9: (src) << ASM → dst
10: (src) << SHIFT → dst

Биты состояния:

Зависит от SXM при всех загрузках аккумулятора
Зависит от OVM в случае загрузок с SHIFT или со сдвигом ASM
Оказывают действие на OVdst (или OVsrc, если dst = src) при загрузках с SHIFT или со сдвигом ASM

Описание:

Эта инструкция загружает в аккумулятор (в *dst* или *src*, если *dst* не определен) значение памяти данных или непосредственное значение, обеспечивая при этом различные варианты сдвига загружаемого значения. В дополнение, инструкция поддерживает перемещения из аккумулятора в аккумулятор со сдвигом.

Примечание – При определенных ситуациях некоторые синтаксисы могут быть транслированы как другие:

– Синтаксис 4: если *SHIFT* = 0, опкод инструкции транслируется как синтаксис 1.

– Синтаксис 4: если $0 < SHIFT \leq 15$ и Smem режимом косвенной адресации включен в Xmem, опкод инструкции транслируется как синтаксис 5.

– Синтаксис 5: если *SHFT* = 0, опкод инструкции транслируется как синтаксис 1.

– Синтаксис 7: если *SHFT* = 0 и $0 \leq lk \leq 255$, опкод инструкции транслируется как синтаксис 6.

LD Загрузка аккумулятора со сдвигом

Слов: Синтаксисы 1, 2, 3, 5, 6, 9 и 10: 1 слово
Синтаксисы 4, 7 и 8: 2 слова
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: Синтаксисы 1, 2, 3, 5, 6, 9 и 10: 1 цикл
Синтаксисы 4, 7 и 8: 2 цикла
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Синтаксисы 1, 2, 3 и 5: Класс 3A (см. страницу 34)
Синтаксисы 1, 2 и 3: Класс 3B (см. страницу 35)
Синтаксис 4: Класс 4A (см. страницу 36)
Синтаксис 4: Класс 4B (см. страницу 37)
Синтаксис 6, 9 и 10: Класс 1 (см. страницу 32)
Синтаксисы 7 и 8: Класс 2 (см. страницу 33)

Пример 1:

```
LD *AR1, A
```

	Before Instruction	After Instruction
A	00 0000 0000	00 0000 FEDC
SXM	0	0
AR1	0200	0200
Data Memory		
0200h	FEDC	FEDC

Пример 2:

```
LD *AR1, A
```

	Before Instruction	After Instruction
A	00 0000 0000	FF FFFF FEDC
SXM	1	1
AR1	0200	0200
Data Memory		
0200h	FEDC	FEDC

Пример 3:

```
LD *AR1, TS, B
```

	Before Instruction	After Instruction
B	00 0000 0000	FF FFFE DC00
SXM	1	1
AR1	0200	0200
T	8	8
Data Memory		
0200h	FEDC	FEDC

LD Загрузка аккумулятора со сдвигом

Пример 4:

```
LD *AR3+, 16, A
```

	Before Instruction	After Instruction
A	00 0000 0000	FF FEDC 0000
SXM	1	1
AR3	0300	AR1 0301
Data Memory		
0300h	FEDC	0300h FEDC

Пример 5:

```
LD #248, B
```

	Before Instruction	After Instruction
B	00 0000 0000	00 0000 00F8
SXM	1	1

Пример 6:

```
LD A, 8, B
```

	Before Instruction	After Instruction
A	00 7FFD 0040	00 7FF0 0040
B	00 0000 FFFF	7F FD00 4000
OVB	0	1
SXM	1	1
Data Memory		
0200h	FEDC	0200h FEDC

LD Загрузка T/DP/ASM/ARP

Синтаксис:
1: LD *Smem*, T
2: LD *Smem*, DP
3: LD *#k9*, DP
4: LD *#k5*, ASM
5: LD *#k3*, ARP
6: LD *Smem*, ASM

Загрузка аккумулятора со сдвигом – см. страницу 136.

Операнды:
Smem: Одиночный операнд памяти данных
$0 \le k9 \le 511$
$-16 \le k5 \le 15$
$0 \le k3 \le 7$

Опкоды:

1:

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

2:

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

3:

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

4:

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

5:

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

6:

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

Выполнение:
1: (Smem) → T
2: (Smem(8-0)) → DP
3: k9 → DP
4: k5 → ASM
5: k3 → ARP
6: (Smem(4-0)) → ASM

Биты состояния: нет

Описание: Эта инструкция загружает определенное значение в T или в поля DP, ASM или ARP регистров ST1 или ST0. Загружаемое значение может быть одиночным операндом памяти данных *Smem* или константой.

LD Загрузка T/DP/ASM/ARP

Слов: 1 слово

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: Синтаксисы 1, 3, 4, 5 и 6: 1 цикл
Синтаксис 2: 3 цикла

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Синтаксисы 1 и 6: Класс 3A (см. страницу 34)
Синтаксисы 1 и 6: Класс 3B (см. страницу 35)
Синтаксис 2: Класс 5A (см. страницу 38)
Синтаксис 2: Класс 5B (см. страницу 38)
Синтаксис 3, 4 и 5: Класс 1 (см. страницу 32)

Пример 1:

```
LD *AR3+, T
```

	Before Instruction	After Instruction
T	0000	FEDC
AR3	0300	0301
Data Memory		
0300h	FEDC	FEDC

Пример 2:

```
LD *AR4, DP
```

	Before Instruction	After Instruction
AR4	0200	0200
DP	1FF	0DC
Data Memory		
0200h	FEDC	FEDC

Пример 3:

```
LD #23, DP
```

	Before Instruction	After Instruction
DP	1FF	017

Пример 4:

```
LD 15, ASM
```

	Before Instruction	After Instruction
ASM	00	0F

Пример 5:

```
LD 3, ARP
```

	Before Instruction	After Instruction
ARP	0	3

Пример 6:

```
LD 0, ASM
```

	Before Instruction	After Instruction
ASM	00	1C
DP	004	004
Data Memory		
0200h	FEDC	FEDC

LDM Загрузка аккумулятора из картированного в памяти регистра

Синтаксис: LDM *MMR, dst*

Операнды:
MMR: Картированный в памяти регистр
src: A (аккумулятор A)
B (аккумулятор B)

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	D	I	A	A	A	A	A	A	A

Выполнение: (MMR) → dst(15-0)
00 0000h → dst(39-16)

Биты состояния: нет

Описание: Эта инструкция загружает *dst* содержимым картированного в памяти регистра *MMR*. И при прямой и при косвенной адресации, девять старших разрядов действующего адреса принудительно очищаются (заполняются 0) для выбора нулевой страницы памяти данных, независимо от текущего содержимого (DP) или содержимого старших девяти бит ARx. Независимо от значения бита SXM режим расширения знака не активизируется.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 3A (см. страницу 34)

Пример 1:

```
LDM AR4, A
```

	Before Instruction		After Instruction
A	00 0000 1111	A	00 0000 FFFF
AR4	FFFF	AR4	FFFF

Пример 2:

```
LDM 060h, B
```

	Before Instruction		After Instruction
B	00 0000 0000	B	00 0000 1234
Data Memory			
0060h	1234	0060h	1234

LD||MAC[R] Загрузка аккумулятора параллельно с умножением-накоплением с/без округления

Синтаксис: LD *Xmem, dst*
|| MAC[R] *Ymem*[, *dst_*]

Операнды:
dst: A (аккумулятор A)
B (аккумулятор B)
dst_: Если *dst* = A, то *dst_* = B; если *dst* = B, то *dst_* = A
Xmem, Ymem: Двойные операнды памяти данных

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	1	0	R	D	X	X	X	X	Y	Y	Y	Y

Выполнение:
(Xmem) << 16 → dst(31-16)
Если указано округление (R – rounding)
Round (((Ymem) × (T)) + (dst_) → dst_
Иначе
((Ymem) × (T)) + (dst_) → dst_

Биты состояния:
Зависит от SXM, FRCT и OVM
Оказывает влияние на OVdst_

Описание: Эта инструкция загружает в старшую часть *dst* (биты 31-16) 16-разрядный двойной операнд памяти данных *Xmem*, сдвинутый влево на 16 разрядов. Параллельно, эта инструкция умножает двойной операнд памяти данных *Ymem* на содержимое *T*, прибавляет к полученному значению содержимое *dst_* и результат заносит в *dst_*. Если в инструкции указан суффикс R, результат умножения и сложения округляется путем добавления к нему числа 2^{15} и очистки младших бит (15-0) в 0. Результат сохраняется в *dst_*.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 7 (см. страницу 41)

Пример 1:

```
LD *AR4+, A
||MAC *AR5+, B
```

	Before Instruction	After Instruction
A	00 0000 1000	00 1234 0000
B	00 0000 1111	00 010C 9511
T	0400	0400
FRCT	0	0
AR4	0100	0101
AR5	0200	0201
Data Memory		
0100h	1234	1234
0200h	4321	4321

LD||MAC[R] Загрузка аккумулятора параллельно с умножением-накоплением с/без округления

Пример 2:

```
LD *AR4+, A
||MACR *AR5+, B
```

	Before Instruction		After Instruction
A	00 0000 1000	A	00 1234 0000
B	00 0000 1111	B	00 010D 0000
T	0400	T	0400
FRCT	0	FRCT	0
AR4	0100	AR4	0101
AR5	0200	AR5	0201
Data Memory			
0100h	1234	0100h	1234
0200h	4321	0200h	4321

LD||MAS[R] Загрузка аккумулятора параллельно с умножением-вычитанием с/без округления

Синтаксис: LD *Xmem, dst*
|| MAS[R] *Ymem* [, *dst_*]

Операнды:
dst: A (аккумулятор A)
B (аккумулятор B)
dst_: Если *dst* = A, то *dst_* = B; если *dst* = B, то *dst_* = A
Xmem, Ymem: двойные операнды памяти данных

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	1	1	R	D	X	X	X	X	Y	Y	Y	Y

Выполнение:
(Xmem) << 16 → dst(31-16)
Если указано округление (R – rounding)
Round ((dst_) – ((T) × (Ymem))) → dst_
Иначе
(dst_) – ((T) × (Ymem)) → dst_

Биты состояния:
Зависит от SXM, FRCT и OVM
Оказывает влияние на OVdst_

Описание:
Эта инструкция загружает в старшую часть *dst* (биты 31-16) 16-разрядный двойной операнд памяти данных *Xmem*, сдвинутый влево на 16 разрядов. Параллельно, эта инструкция умножает двойной операнд памяти данных *Ymem* на содержимое *T* и вычитает полученный результат из содержимого *dst_*; конечный результат заносится в *dst_*.
Если в инструкции указан суффикс R, результат умножения и вычитания округляется путем добавления к нему числа 2^{15} и очистки младших бит (15-0) в 0. Результат сохраняется в *dst_*.

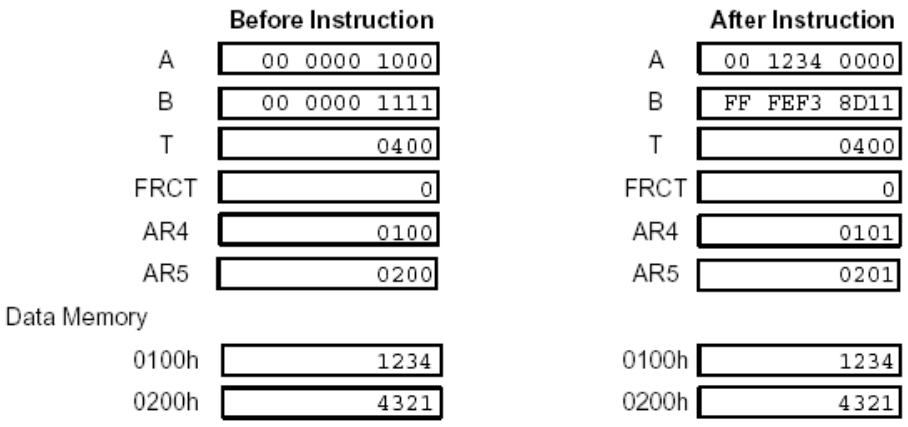
Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 7 (см. страницу 41)

Пример 1:

```
LD *AR4+, A
||MAS *AR5+, B
```



LD||MAS[R] Загрузка аккумулятора параллельно с умножением-вычитанием с/без округления

Пример 2:

```
LD *AR4+, A
||MASR *AR5+, B
```

	Before Instruction	After Instruction
A	00 0000 1000	00 1234 0000
B	00 0000 1111	FF FEF4 0000
T	0400	0400
FRCT	0	0
AR4	0100	0101
AR5	0200	0201
Data Memory		
0100h	1234	1234
0200h	4321	4321

LDR Загрузка значения памяти в старшую часть аккумулятора с округлением

Синтаксис: LDR *Smem, dst*

Операнды:
Smem: Одиночный операнд памяти данных
dst: A (аккумулятор A)
B (аккумулятор B)

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	1	1	D	I	A	A	A	A	A	A	A

Выполнение: (Smem) << 16 + 1 << 15 → dst(31-16)

Биты состояния: Зависит от SXM

Описание: Эта инструкция загружает значение памяти данных Smem, сдвинутое на 16 разрядов в старшую часть *dst* (биты 31-16). Содержимое Smem округляется путем добавления к нему числа 2^{15} и очистки 15 младших битов (14-0) аккумулятора в 0. 15-й бит аккумулятора устанавливается в 1.

Слов: 1 слово

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 1 цикл

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы:
Класс 3A (см. страницу 34)
Класс 3B (см. страницу 35)

Пример:

```
LDR *AR1, A
```

	Before Instruction	After Instruction
A	00 0000 0000	00 FEDC 8000
SXM	0	0
AR1	0200	0200
Data Memory		
0200h	FEDC	FEDC

LDU Загрузка в аккумулятор беззнакового значения памяти

Синтаксис: LDU *Smem, dst*

Операнды:
Smem: Одиночный операнд памяти данных
dst: A (аккумулятор A)
B (аккумулятор B)

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	1	D	I	A	A	A	A	A	A	A

Выполнение: (Smem) → dst(15-0)
00 0000h → dst(39-16)

Биты состояния: нет

Описание: Эта инструкция загружает значение памяти данных Smem в младшую часть *dst* (биты 15-0). Биты безопасности и старшие биты *dst* (биты 39-16) очищаются в 0. В этом случае данные интерпретируются как 16-разрядное число без знака. Независимо от состояния бита SXM инструкция подавляет знаковое расширение.

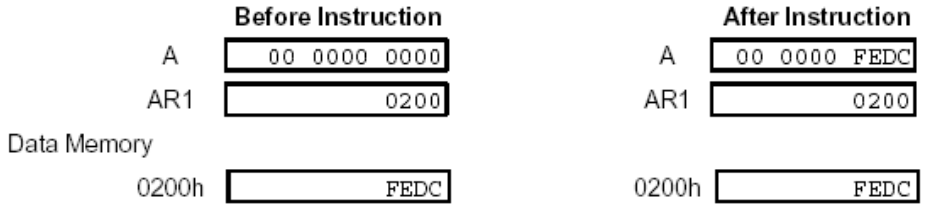
Слов: 1 слово
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 1 цикл
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 3A (см. страницу 34)
Класс 3B (см. страницу 35)

Пример:

```
LDU *AR1, A
```



LMS Минимальная среднеквадратичная ошибка

Синтаксис: LMS *Xmem, Ymem*

Операнды: Xmem, Ymem: Двойные операнды памяти данных

Опкод:

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

Выполнение: (A) + (Xmem) << 16 + 2^{15} → A
(B) + (Xmem) × (Ymem) → B

Биты состояния: Зависит от SXM, FRCT и OVM
Оказывает действие на биты C, OVA и OVB.

Описание: Эта инструкция выполняет LMS-алгоритм – вычисление минимальной среднеквадратичной ошибки. Двойной операнд памяти данных (*Xmem*) сдвигается влево на 16 разрядов и складывается с содержимым аккумулятора A. Результат этого сложения округляется (путем прибавления числа 2^{15} к старшей части аккумулятора (биты 31-16)), и полученное значение сохраняется в аккумуляторе A. Параллельно *Xmem* умножается на *Ymem* и результат складывается с содержимым аккумулятора B. *Xmem* не переписывает содержимое T; поэтому в T всегда содержится значение ошибки, используемое для обновления коэффициентов.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 7 (см. страницу 41)

Пример:

```
LMS *AR3+, *AR4+
```

	Before Instruction	After Instruction
A	00 7777 8888	00 77CD 0888
B	00 0000 0100	00 0000 3972
FRCT	0	0
AR3	0100	0101
AR4	0200	0201
Data Memory		
0100h	0055	0055
0200h	00AA	00AA

LTD Загрузка T и вставка задержки

Синтаксис: LTD *Smem*

Операнды: Smem: Одиночный операнд памяти данных

Опкод:

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

Выполнение: (Smem) → T
(Smem) → Smem + 1

Биты состояния: нет

Описание: Эта инструкция копирует содержимое ячейки *Smem* в T и в ячейку с адресом, следующим за адресом *Smem*. После выполнения копирования, копируемое значение остается прежним. Эта инструкция полезна при реализации Z-задержки в приложениях цифровой обработки сигналов. Эта функкция также включена в инструкцию DELAY (задержка памяти страница 119).

Слов: 1 слово

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 1 цикл

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 24A (см. страницу 67)
Класс 24B (см. страницу 67)

Пример:

```
LTD *AR3
```

	Before Instruction	After Instruction
T	0000	6CAC
AR3	0100	0100
Data Memory		
0100h	6CAC	6CAC
0101h	xxxx	6CAC

MAC[R] Умножение и сложение с/без округления

Синтаксис:
1: MAC[R] *Smem, src*
2: MAC[R] *Xmem, Ymem, src*[, *dst*]
3: MAC *#k, src*[, *dst*]
4: MAC *Smem, #lk, src*[, *dst*]

Операнды:

Smem:	Одиночный операнд памяти данных
Xmem, Ymem:	Двойные операнды памяти данных
src, dst:	A (аккумулятор A)
	B (аккумулятор B)

$-32\,768 \le lk \le 32\,767$

Опкоды:

1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	0	R	S	I	A	A	A	A	A	A	A

2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	0	R	S	D	X	X	X	X	Y	Y	Y	Y

3:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	0	1	1	0	0	1	1	1
16-битная константа															

4:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	1	S	D	I	A	A	A	A	A	A	A
16-битная константа															

Выполнение:
1: (Smem) × (T) + (src) → src
2: (Xmem) × (Ymem) + (src) → dst
(Xmem) → T
3: (T) × lk + (src) → dst
4: (Smem) × lk + (src) → dst
(Smem) → T

Биты состояния:
Зависит от FRCT и OVM
Оказывает действие на OVdst (или OVsrc, если dst не определен)

Описание:
Эта инструкция производит умножение и сложение с/без округления. Результат сохраняется в dst или src, в зависимости от того – что указано в инструкции. В синтаксисах 2 и 4 значение памяти данных после выполнения инструкции сохраняется в T. T обновляется в ходе фазы чтения.

Если в инструкции указан суффикс R, то результат умножения и сложения округляется, путем прибавления числа 2^{15} и очистки младших битов (15-0) в 0.

MAC[R] Умножение и сложение с/без округления

Слов: Синтаксисы 1 и 2: 1 слово
Синтаксисы 3 и 4: 2 слова
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: Синтаксисы 1 и 2: 1 цикл
Синтаксисы 3 и 4: 2 цикла
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Синтаксис 1: Класс 3A (см. страницу 34)
Синтаксис 1: Класс 3B (см. страницу 35)
Синтаксис 2: Класс 7 (см. страницу 41)
Синтаксис 3: Класс 2 (см. страницу 33)
Синтаксис 4: Класс 6A (см. страницу 39)
Синтаксис 4: Класс 6B (см. страницу 40)

Пример 1:

```
MAC *AR5+, A
```

	Before Instruction		After Instruction
A	00 0000 1000	A	00 0048 E000
T	0400	T	0400
FRCT	0	FRCT	0
AR5	0100	AR5	0101
Data Memory			
0100h	1234	0100h	1234

Пример 2:

```
MAC #345h, A, B
```

	Before Instruction		After Instruction
A	00 0000 1000	A	00 0000 1000
B	00 0000 0000	B	00 001A 3800
T	0400	T	0400
FRCT	1	FRCT	1

Пример 3:

```
MAC *AR5+, #1234h, A
```

	Before Instruction		After Instruction
A	00 0000 1000	A	00 0626 1060
T	0000	T	5678
FRCT	0	FRCT	0
AR5	0100	AR5	0101
Data Memory			
0100h	5678	0100h	5678

MAC[R] Умножение и сложение с/без округления

Пример 4:

```
MAC *AR5+, *AR6+,A, B
```

	Before Instruction	After Instruction
A	00 0000 1000	00 0000 1000
B	00 0000 0004	00 0C4C 10C0
T	0008	5678
FRCT	1	1
AR5	0100	0101
AR6	0200	0201
Data Memory		
0100h	5678	5678
0200h	1234	1234

Пример 5:

```
MACR *AR5+, A
```

	Before Instruction	After Instruction
A	00 0000 1000	00 0049 0000
T	0400	0400
FRCT	0	0
AR5	0100	0101
Data Memory		
0100h	1234	1234

Пример 6:

```
MACR *AR5+, *AR6+,A, B
```

	Before Instruction	After Instruction
A	00 0000 1000	00 0000 1000
B	00 0000 0004	00 0C4C 0000
T	0008	5678
FRCT	1	1
AR5	0100	0101
AR6	0200	0201
Data Memory		
0100h	5678	5678
0200h	1234	1234

MACA[R] Умножение на аккумулятор A и накопление с/без округления

Синтаксис:
1: MACA[R] *Smem*, [, B]
2: MACA[R] T, *src* [, *dst*]

Операнды:
Smem: Одиночный операнд памяти данных
src, dst: A (аккумулятор A)
B (аккумулятор B)

Опкоды:

1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	R	1	I	A	A	A	A	A	A	A

2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	S	D	1	0	0	0	1	0	0	R

Выполнение:
1: (Smem) × (A(32-16)) + (B) → B
(Smem) → T
2: (T) × (A(32-16)) + (src) → dst

Биты состояния:
Зависит от FRCT и OVM
Оказывает действие на OVdst (или OVsrc, если dst не определен) и от OVB в синтаксисе 1.

Описание:
Эта инструкция умножает старшую часть аккумулятора A (биты 32-16) на одиночный операнд памяти данных *Smem* или на содержимое T, и сохраняет полученный результат в аккумулятор B (синтаксис 1) или в *dst* (или в *src*, если *dst* не определен). При умножении A(32-16) используется как 17-разрядный операнд.

Если в инструкции указан суффикс R, то результат умножения и сложения округляется, путем прибавления числа 2^{15} и очистки 16 младших разрядов приемника (15-0) в 0.

Слов:
1 слово
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов:
1 цикл
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы:
Синтаксисы 1 и 2: Класс 3A (см. страницу 34)
Синтаксисы 1 и 2: Класс 3B (см. страницу 35)
Синтаксисы 3 и 4: Класс 1 (см. страницу 32)

MACA[R] Умножение на аккумулятор A и накопление с/без округления

Пример 1:

```
MACA *AR5+
```

	Before Instruction	After Instruction
A	00 1234 0000	00 1234 0000
B	00 0000 0000	00 0626 0060
T	0400	5678
FRCT	0	0
AR5	0100	0101
Data Memory		
0100h	5678	5678

Пример 2:

```
MACA T, B, B
```

	Before Instruction	After Instruction
A	00 1234 0000	00 1234 0000
B	00 0002 0000	00 009D 4BA0
T	0444	0444
FRCT	1	1

Пример 3:

```
MACAR *AR5+, B
```

	Before Instruction	After Instruction
A	00 1234 0000	00 1234 0000
B	00 0000 0000	00 0626 0000
T	0400	5678
FRCT	0	0
AR5	0100	0101
Data Memory		
0100h	5678	5678

Пример 4:

```
MACAR T, B, B
```

	Before Instruction	After Instruction
A	00 1234 0000	00 1234 0000
B	00 0002 0000	00 009D 0000
T	0444	0444
FRCT	1	1

MACD Умножение с памятью программ и накопление с задержкой

Синтаксис: MACD *Smem, pmad, src*

Операнды:

Smem:	Одиночный операнд памяти данных
src, dst:	A (аккумулятор A)
	B (аккумулятор B)

$0 \leq pmad \leq 65\,535$

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	0	1	S	I	A	A	A	A	A	A	A
16-битная константа															

Выполнение:

pmad → PAR
Если (RC) ≠ 0
То
(Smem) × (Pmem по адресу из PAR) + (src) → src
(Smem) → T
(Smem) → Smem + 1
(PAR) + 1 → PAR
Иначе
(Smem) × (Pmem по адресу из PAR) + (src) → src
(Smem) → T
(Smem) → Smem + 1

Биты состояния: Зависит от FRCT и OVM
Оказывает действие на OVsrc

Описание: Эта инструкция умножает одиночный операнд памяти данных *Smem* на значение памяти программ *pmad*, складывает произведение с *src* и полученный результат сохраняет в *src*. Содержимое ячейки *Smem* копируется в T и в ячейку с адресом, следующим за адресом *Smem*. В режиме повтора, адрес памяти программ (в регистре программного адреса PAR) увеличивается на 1 в каждой следующей итерации, а сама инструкция, после загрузки в конвейер (первое выполнение), становится эффективно одноцикловой. Эта инструкция содержит функцию DELAY (задержка памяти страница 119).

Слов: 2 слова
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 3 цикла
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 23A (см. страницу 64)
Класс 23B (см. страницу 66)

MACD Умножение с памятью программ и накопление с задержкой

Пример: `MACD *AR3-, COEFFS, A`

	Before Instruction		After Instruction
A	00 0077 0000	A	00 007D 0B44
T	0008	T	0055
FRCT	0	FRCT	0
AR3	0100	AR3	00FF
Program Memory			
COEFFS	1234	COEFFS	1234
Data Memory			
0100h	0055	0100h	0055
0101h	0066	0101h	0055

MACP Умножение с памятью программ и накопление

Синтаксис: MACP *Smem, pmad, src*

Операнды:
Smem: Одиночный операнд памяти данных
src: A (аккумулятор A)
B (аккумулятор B)
$0 \leq pmad \leq 65\,535$

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	0	0	S	I	A	A	A	A	A	A	A
16-битная константа															

Выполнение:
pmad → PAR
Если (RC) ≠ 0
То
(Smem) × (Pmem по адресу из PAR) + (src) → src
(Smem) → T
(PAR) + 1 → PAR
Иначе
(Smem) × (Pmem по адресу из PAR) + (src) → src
(Smem) → T

Биты состояния: Зависит от FRCT и OVM
Оказывает действие на OVsrc

Описание: Эта инструкция умножает одиночный операнд памяти данных *Smem* на значение памяти программ *pmad*, складывает произведение с *src* и полученный результат сохраняет в *src*. Содержимое ячейки *Smem* копируется в T. В режиме повтора, адрес памяти программ (в регистре программного адреса PAR) увеличивается на 1 в каждой следующей итерации, а сама инструкция, после загрузки в конвейер (первое выполнение), становится эффективно одноцикловой.

Слов: 2 слова
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 3 цикла

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 22A (см. страницу 62)
Класс 22B (см. страницу 63)

MACP Умножение с памятью программ и накопление

Пример:

```
MACP *AR3-, COEFFS, A
```

	Before Instruction		After Instruction
A	00 0077 0000	A	00 007D 0B44
T	0008	T	0055
FRCT	0	FRCT	0
AR3	0100	AR3	00FF
Program Memory			
COEFFS	1234	COEFFS	1234
Data Memory			
0100h	0055	0100h	0055
0101h	0066	0101h	0066

MACSU Знаковое и беззнаковое умножение и накопление

Синтаксис: MACSU *Xmem, Ymem, src*

Операнды:

Xmem, Ymem:	Двойные операнды памяти данных
src:	A (аккумулятор A)
	B (аккумулятор B)

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	0	1	1	S	X	X	X	X	Y	Y	Y	Y

Выполнение: беззнаковый(Xmem) × знаковый(Ymem) + (src) → src
(Xmem) → T

Биты состояния: Зависит от FRCT и OVM
Оказывает действие на OVsrc

Описание: Эта инструкция умножает беззнаковое значение памяти данных *Xmem* на значение памяти данных *Ymem* со знаком, складывает произведение с *src* и полученный результат сохраняет в *src*. 16-разрядное беззнаковое значение *Xmem* сохраняется в T. Обновление T беззнаковым значением *Xmem* происходит в фазе чтения.

Данные, адресуемые *Xmem*, подаются по шине D. Данные, адресуемые *Ymem*, подаются по шине C.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 7 (см. страницу 41)

Пример:

```
MACSU *AR4+, *AR5+, A
```

	Before Instruction	After Instruction
A	00 0000 1000	00 09A0 AA84
T	0008	8765
FRCT	0	0
AR4	0100	0101
AR5	0200	0201
Data Memory		
0100h	8765	8765
0200h	1234	1234

MAR Модификация вспомогательного регистра

Синтаксис: MAR *Smem*

Операнды: Smem: Одиночный операнд памяти данных

Опкод:

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

Выполнение: При косвенной адресации вспомогательный регистр модифицируется следующим образом:

В режиме совместимости (CMPT = 1):

Если (ARx = AR0)

AR(ARP) модифицируется

ARP не изменяется

Иначе

ARx модифицируется

x → ARP

При выключенном режиме совместимости (CMPT = 0):

ARx модифицируется

ARP не изменяется

Биты состояния: Зависит от CMPT

Оказывает действие на ARP (если CMPT = 1)

Описание: Эта инструкция модифицирует содержимое выбранного вспомогательного регистра (ARx) в соответствии с *Smem*. В режиме совместимости (CMPT = 1) модифицируется содержимое ARx, а так же значение указателя вспомогательного регистра (ARP).

Если CMPT = 0, вспомогательный регистр модифицируется, однако ARP остается прежним.

Слов: 1 слово

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 1 цикл

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 1 (см. страницу 32)

Класс 2 (см. страницу 33)

MAR Модификация вспомогательного регистра

Пример 1: `MAR *AR3+`

	Before Instruction	After Instruction
CMPT	0	0
ARP	0	0
AR3	0100	0101

Пример 2: `MAR *AR0-`

	Before Instruction	After Instruction
CMPT	1	1
ARP	4	4
AR4	0100	00FF

Пример 3: `MAR *AR3`

	Before Instruction	After Instruction
CMPT	1	1
ARP	0	3
AR0	0008	0008
AR3	0100	0100

Пример 4: `MAR *+AR3`

	Before Instruction	After Instruction
CMPT	1	1
ARP	0	3
AR3	0100	0101

Пример 5: `MAR *AR3-`

	Before Instruction	After Instruction
CMPT	1	1
ARP	0	3
AR3	0100	00FF

MAS[R] Умножение и вычитание с/без округления

Синтаксис: 1: MAS[R] *Smem, src*
2: MAS[R] *Xmem, Ymem, src*[*, dst*]

Операнды:

Xmem, Ymem:	Двойные операнды памяти данных
Smem:	Одиночный операнд памяти данных
src, dst:	A (аккумулятор A)
	B (аккумулятор B)

Опкоды:

1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	R	S	I	A	A	A	A	A	A	A

2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	R	S	D	X	X	X	X	Y	Y	Y	Y

Выполнение: 1: (src) – (Smem) × (T) → src
2: (src) – (Xmem) × (Ymem) → dst
(Xmem) → T

Биты состояния: Зависит от FRCT и OVM
Оказывает действие на OVdst (или OVsrc, если dst = src)

Описание: Эта инструкция умножает операнд на содержимое T или перемножает два операнда, вычитает произведение из *src* и сохраняет результат в *src* или *dst*. *Xmem* загружается в T в фазе чтения.
Если в инструкции указан суффикс R, то результат умножения и вычитания округляется, путем прибавления числа 2^{15} и очистки младших бит (15-0) в 0.
Данные, адресуемые *Xmem*, подаются по шине D. Данные, адресуемые *Ymem*, подаются по шине C.

Слов: 1 слово
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 1 цикл
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Синтаксис 1: Класс 3A (см. страницу 34)
Синтаксис 1: Класс 3B (см. страницу 35)
Синтаксис 2: Класс 7 (см. страницу 41)

MAS[R] Умножение и вычитание с/без округления

Пример 1:

```
MAS *AR5+, A
```

	Before Instruction	After Instruction
A	00 0000 1000	FF FFB7 4000
T	0400	0400
FRCT	0	0
AR5	0100	0101
Data Memory		
0100h	1234	1234

Пример 2:

```
MAS *AR5+, *AR6+, A, B
```

	Before Instruction	After Instruction
A	00 0000 1000	00 0000 1000
B	00 0000 0004	FF F9DA 0FA0
T	0008	5678
FRCT	1	1
AR5	0100	0101
AR6	0200	0201
Data Memory		
0100h	5678	5678
0200h	1234	1234

Пример 3:

```
MASR *AR5+, A
```

	Before Instruction	After Instruction
A	00 0000 1000	FF FFB7 0000
T	0400	0400
FRCT	0	0
AR5	0100	0101
Data Memory		
0100h	1234	1234

Пример 4:

```
MASR *AR5+, *AR6+, A, B
```

	Before Instruction	After Instruction
A	00 0000 1000	00 0000 1000
B	00 0000 0004	FF F9DA 0000
T	0008	5678
FRCT	1	1
AR5	0100	0101
AR6	0200	0201
Data Memory		
0100h	5678	5678
0200h	1234	1234

MASA[R] Умножение на аккумулятор A и вычитание с/без округления

Синтаксис: 1: MASA *Smem*[, B]
2: MASA[R] T, *src*[, *dst*]

Операнды:

Smem:	Одиночный операнд памяти данных
src, dst:	A (аккумулятор A)
	B (аккумулятор B)

Опкоды:

1:

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

2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	S	D	1	0	0	0	1	0	1	R

Выполнение: 1: (B) – (Smem) × (A(32-16)) → B
(Smem) → T
2: (src) – (T) × (A(32-16)) → dst

Биты состояния: Зависит от FRCT и OVM
Оказывает действие на OVdst (или OVsrc, если dst не определен) и OVB в синтаксисе 1.

Описание: Эта инструкция умножает старшую часть аккумулятора A (биты 32-16) на одиночный операнд памяти данных *Smem* или на содержимое T, произведение вычитается из B (синтаксис 1) или *src*. Результат вычислений сохраняется в аккумуляторе B (синтаксис 1) или в *dst* или в *src*, если *dst* не определен. *Smem* загружается в T в фазе чтения.
Если в инструкции указан суффикс R, то результат умножения и вычитания округляется, путем прибавления числа 2^{15} и очистки младших бит (15-0) в 0.

Слов: 1 слово
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Циклов: 1 цикл
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Синтаксис 1: Класс 3A (см. страницу 34)
Синтаксис 1: Класс 3B (см. страницу 35)
Синтаксис 2: Класс 1 (см. страницу 32)

MASA[R] Умножение на аккумулятор A и вычитание с/без округления

Пример 1:

```
MASA *AR5+
```

	Before Instruction	After Instruction
A	00 1234 0000	00 1234 0000
B	00 0002 0000	FF F9DB FFA0
T	0400	5678
FRCT	0	0
AR5	0100	0101
Data Memory		
0100h	5678	5678

Пример 2:

```
MASA T, B
```

	Before Instruction	After Instruction
A	00 1234 0000	00 1234 0000
B	00 0002 0000	FF FF66 B460
T	0444	0444
FRCT	1	1

Пример 3:

```
MASAR T, B
```

	Before Instruction	After Instruction
A	00 1234 0000	00 1234 0000
B	00 0002 0000	FF FF67 0000
T	0444	0444
FRCT	1	1

MAX Выбор максимального аккумулятора

Синтаксис: MAX *dst*

Операнды: dst: A (аккумулятор A)
B (аккумулятор B)

Опкод:

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

Выполнение:
Если (A > B)
То
(A) → dst
0 → C
Иначе
(B) → dst
1 → C

Биты состояния: Оказывает действие на C

Описание: Эта инструкция сравнивает содержимое аккумуляторов и максимальное значение сохраняет в *dst*. Если максимальное значение находится в аккумуляторе A бит переноса C очищается в 0; в другом случает бит C устанавливается в 1.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 1 (см. страницу 32)

Пример 1:

```
MAX A
```

	Before Instruction		After Instruction	
A	FFF6	-10	FFF6	-10
B	FFCB	-53	FFCB	-53
C	1		0	

Пример 2:

```
MAX A
```

	Before Instruction	After Instruction
A	00 0000 0055	00 0000 1234
B	00 0000 1234	00 0000 1234
C	0	1

MIN Выбор минимального аккумулятора

Синтаксис: MIN *dst*

Операнды: dst: A (аккумулятор A)
B (аккумулятор B)

Опкод:

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

Выполнение:
Если (A < B)
То
(A) → dst
0 → C
Иначе
(B) → dst
1 → C

Биты состояния: Оказывает действие на C

Описание: Эта инструкция сравнивает содержимое аккумуляторов и минимальное значение сохраняет в *dst*. Если минимальное значение находится в аккумуляторе A бит переноса C очищается в 0; в другом случает бит C устанавливается в 1.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 1 (см. страницу 32)

Пример 1:

```
MIN A
```

	Before Instruction		After Instruction	
A	FFCB	-53	FFCB	-53
B	FFF6	-10	FFF6	-10
C	1		0	

Пример 2:

```
MIN A
```

	Before Instruction	After Instruction
A	00 0000 1234	00 0000 1234
B	00 0000 1234	00 0000 1234
C	0	1

MPY[R] Умножение с/без округления

Синтаксис:
1: MPY[R] *Smem, dst*
2: MPY *Xmem, Ymem, dst*
3: MPY *Smem, #lk, dst*
4: MPY *#lk, dst*

Операнды:
Smem: Одиночный операнд памяти данных
Xmem, Ymem: Двойные операнды памяти данных
dst: A (аккумулятор A)
B (аккумулятор B)
$-32\,768 \le lk \le 32\,767$

Опкоды:

1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	0	R	D	I	A	A	A	A	A	A	A

2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	0	1	0	D	X	X	X	X	Y	Y	Y	Y

3:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	1	D	I	A	A	A	A	A	A	A
16-битная константа															

4:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	D	0	1	1	0	0	1	1	0
16-битная константа															

Выполнение:
1: (T) × (Smem) → dst
2: (Xmem) × (Ymem) → dst
(Xmem) → T
3: (Smem) × lk → dst
(Smem) → T
4: (T) × lk → dst

Биты состояния:
Зависит от FRCT и OVM
Оказывает действие на OVdst

Описание:
Эта инструкция умножает содержимое T или значение памяти данных на значение памяти данных или непосредственное значение и сохраняет результат в *dst*. Загрузка *Smem* и *Xmem* в T происходит в фазе чтения.
Если в инструкции указан суффикс R, то результат умножения округляется, путем прибавления числа 2^{15} и очистки младших бит (15-0) в 0.

Слов:
Синтаксисы 1 и 2: 1 слово
Синтаксисы 3 и 4: 2 слова
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

MPY[R] Умножение с/без округления

Циклов: Синтаксисы 1 и 2: 1 цикл
Синтаксисы 3 и 4: 2 цикла
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Синтаксис 1: Класс 3A (см. страницу 34)
Синтаксис 1: Класс 3B (см. страницу 35)
Синтаксис 2: Класс 7 (см. страницу 41)
Синтаксис 3: Класс 6A (см. страницу 39)
Синтаксис 3: Класс 6B (см. страницу 40)
Синтаксис 4: Класс 2 (см. страницу 33)

Пример 1:

```
MPY 13, A
```

	Before Instruction	After Instruction
A	00 0000 0036	00 0000 0054
T	0006	0006
FRCT	1	1
DP	008	008
Data Memory		
040Dh	0007	0007

Пример 2:

```
MPY *AR2-, *AR4+0%, B;
```

	Before Instruction	After Instruction
B	FF FFFF FFE0	00 0000 0020
FRCT	0	0
AR0	0001	0001
AR2	01FF	01FE
AR4	0300	0301
Data Memory		
01FFh	0010	0010
0300h	0002	0002

Пример 3:

```
MPY #0FFFEh, A
```

	Before Instruction	After Instruction
A	000 0000 1234	FF FFFF C000
T	2000	2000
FRCT	0	0

Пример 4:

```
MPYR 0, B
```

	Before Instruction	After Instruction
B	FF FE00 0001	00 0626 0000
T	1234	1234
FRCT	0	0
DP	004	004
Data Memory		
0200h	5678	5678

MPYA Умножение на аккумулятор A

Синтаксис:
1: MPYA *Smem*
2: MPYA *dst*

Операнды:

Smem:	Одиночный операнд памяти данных
dst:	A (аккумулятор A)
	B (аккумулятор B)

Опкоды:

1:

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

2:

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

Выполнение:
1: (Smem) × (A(32-16)) → B
(Smem) → T
2: (T) × (A(32-16)) → dst

Биты состояния:
Зависит от FRCT и OVM
Оказывает действие на OVdst (OVB в синтаксисе 1)

Описание:
Эта инструкция умножает старшую часть аккумулятора A (биты 32-16) на одиночный операнд памяти данных *Smem* или на содержимое T. Результат вычислений сохраняется в *dst* или в аккумуляторе B. Загрузка *Smem* в T происходит в фазе чтения.

Слов:
1 слово
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов:
1 цикл
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы:
Синтаксис 1: Класс 3A (см. страницу 34)
Синтаксис 1: Класс 3B (см. страницу 35)
Синтаксис 2: Класс 1 (см. страницу 32)

MPYA Умножение на аккумулятор A

Пример 1:

```
MPYA *AR2
```

	Before Instruction	After Instruction
A	FF 8765 1111	FF 8765 1111
B	00 0000 0320	FF D743 6558
T	1234	5678
FRCT	0	0
AR2	0200	0200
Data Memory		
0200h	5678	5678

Пример 2:

```
MPYA B
```

	Before Instruction	After Instruction
A	FF 8765 1111	FF 8765 1111
B	00 0000 0320	FF DF4D B2A3
T	4567	4567
FRCT	0	0

MPYU Беззнаковое умножение

Синтаксис: MPYU *Smem, dst*

Операнды:

Smem:		Одиночный операнд памяти данных
dst:		A (аккумулятор A)
		B (аккумулятор B)

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	1	0	D	I	A	A	A	A	A	A	A

Выполнение: беззнаковое (T) × беззнаковое (Smem) → dst

Биты состояния: Зависит от FRCT и OVM
Оказывает действие на OVdst

Описание: Эта инструкция умножает беззнаковое содержимое T на беззнаковый одиночный операнд памяти данных *Smem* и сохраняет результат в *dst*. Умножитель при этом работает в режиме знакового умножения 17×17, с нулевым старшим битом в обоих операндах. Эта инструкция особенно полезна для вычислений с высокой точностью, таких как умножение двух 32-разрядных чисел с получением 64-разрядного результата.

Слов: 1 слово
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 1 цикл
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 3A (см. страницу 34)
Класс 3B (см. страницу 35)

Пример:

```
MPYU *AR0-, A
```

	Before Instruction	After Instruction
A	FF 8000 0000	00 3F80 0000
T	4000	4000
FRCT	0	0
AR0	1000	0FFF
Data Memory		
1000h	FE00	FE00

MVDD Перемещение из памяти данных в память данных с использованием X,Y адресации

Синтаксис: MVDD *Xmem, Ymem*

Операнды: Xmem, Ymem: Двойные операнды памяти данных

Опкод:

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

Выполнение: (Xmem) → Ymem

Биты состояния: нет

Описание: Эта инструкция копирует содержимое ячейки памяти данных, адресуемой Xmem в ячеку памяти данных, адресуемую Ymem.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 14 (см. страницу 48)

Пример:

```
MVDD *AR3+, *AR5+
```

	Before Instruction		After Instruction
AR3	8000	AR3	8001
AR5	0200	AR5	0201
Data Memory			
0200h	ABCD	0200h	1234
8000h	1234	8000h	1234

MVDK Перемещение из памяти данных в память данных по непосредственному адресу

Синтаксис: MVDK *Smem, dmad*

Операнды: Smem: Одиночный операнд памяти данных
$0 \leq dmad \leq 65\,535$

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	0	0	1	I	A	A	A	A	A	A	A
16-битная константа															

Выполнение:
(dmad) → EAR
Если (RC) ≠ 0
То
(Smem) → Dmem по адресу из EAR
(EAR) + 1 → EAR
Иначе
(Smem) → Dmem по адресу из EAR

Биты состояния: нет

Описание: Эта инструкция копирует одиночный операнд памяти данных *Smem* в ячейку памяти данных, адресуемую 16-разрядным непосредственным значением *dmad* (адрес назначения располагается в регистре EAR шины EAB).
MVDK (в варианте с косвенной адресацией) можно использовать в режиме повтора для перемещения и пересортировки массивов в памяти данных. Выборка исходных данных поддерживается широкими возможности косвенной адресации процессора, а адрес назначения (в EAR) автоматически увеличивается на 1 в каждой следующей итерации. Количество слов, которые будут перемещены, на единицу больше значения, содержащегося в счетчике повторов перед началом выполнения инструкции. После загрузки в конвейер (первое выполнение), инструкция становится эффективно одноцикловой.

Слов: 2 слова
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 2 цикла
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 19A (см. страницу 55)
Класс 19B (см. страницу 57)

MVDK Перемещение из памяти данных в память данных по непосредственному адресу

Пример 1:

```
MVDK 10, 8000h
```

	Before Instruction	After Instruction
DP	004	004
Data Memory		
020Ah	1234	1234
8000h	ABCD	1234

Пример 2:

```
MVDK *AR3-, 1000h
```

	Before Instruction	After Instruction
AR3	01FF	01FE
Data Memory		
1000h	ABCD	1234
01FFh	1234	1234

MVDM Перемещение из памяти данных в MMR

Синтаксис: MVDM *dmad, MMR*

Операнды: MMR: Картированный в памяти регистр
$0 \leq dmad \leq 65\,535$

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	0	1	0	I	A	A	A	A	A	A	A
16-битная константа															

Выполнение:
dmad → DAR
Если (RC) ≠ 0
То
(Dmem по адресу из DAR) → MMR
(DAR) + 1 → DAR
Иначе
(Dmem по адресу из DAR) → MMR

Биты состояния: нет

Описание: Эта инструкция копирует данные из ячейки памяти данных *dmad* (адрес источника располагается в регистре DAR шины DAB) в картированный в памяти регистр (MMR). Содержимое памяти данных адресуется 16-разрядным непосредственным значением *dmad*.
В режиме повтора, с учетом того, что адрес источника (в DAR) автоматически увеличивается на 1 в каждой следующей итерации, эта инструкция (в варианте с косвенной адресацией) может быть использована, например, для восстановления контекста при возврате из вызова или прерывания. Количество слов, которые будут перемещены, на единицу больше значения, содержащегося в счетчике повторов перед началом выполнения инструкции. После загрузки в конвейер (первое выполнение), инструкция становится эффективно одноцикловой.

Слов: 2 слова

Циклов: 2 цикла

Классы: Класс 19A (см. страницу 55)

Пример:

```
MVDM 300h, BK
```

	Before Instruction		After Instruction
BK	ABCD	BK	1234
Data Memory			
0300h	1234	0300h	1234

MVDP Перемещение из памяти данных в память программ

Синтаксис: MVDP *Smem, pmad*

Операнды: Smem: Одиночный операнд памяти данных
$0 \leq pmad \leq 65\,535$

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	0	1	I	A	A	A	A	A	A	A
16-битная константа															

Выполнение:
pmad → PAR
Если (RC) ≠ 0
То
(Smem) → Pmem по адресу из PAR
(PAR) + 1 → PAR
Иначе
(Smem) → Pmem по адресу из PAR

Биты состояния: нет

Описание: Эта инструкция копирует 16-разрядный одиночный операнд памяти данных *Smem* в ячейку памяти программ, адресуемую 16-разрядным непосредственным значением *pmad*. Эту инструкцию можно использовать в режиме повтора для перемещения массивов из памяти данных (с использованием косвенной адресации) в непрерывную область памяти программ, адресуемую 16-разрядным непосредственным значением. Источник и приемник не должны находиться полностью на кристалле или полностью вне его. Если используется повтор, то после загрузки в конвейер (первое выполнение), инструкция становится эффективно одноцикловой. Кроме того, при повторе этой инструкции прерывания запрещены.

Слов: 2 слова
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 4 цикла
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 20A (см. страницу 58)
Класс 20B (см. страницу 59)

Пример:

```
MVDP 0, 0FE00h
```

	Before Instruction	After Instruction
DP	004	004
Data Memory		
0200h	0123	0123
Program Memory		
FE00h	FFFF	0123

MVKD Перемещение из непосредственного адреса памяти данных в память данных

Синтаксис: MVKD *dmad, Smem*

Операнды: Smem: Одиночный операнд памяти данных
$0 \leq dmad \leq 65\,535$

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	0	0	0	I	A	A	A	A	A	A	A
16-битная константа															

Выполнение:
(dmad) → DAR
Если (RC) ≠ 0
То
(Dmem по адресу из DAR) → Smem
(DAR) + 1 → DAR
Иначе
(Dmem по адресу из DAR) → Smem

Биты состояния: нет

Описание: Эта инструкция перемещает информацию внутри памяти данных. Копируемая ячейка (источник) адресуется 16-разрядным непосредственным операндом *dmad*; приемником служит *Smem*.
MVKD (в варианте с косвенной адресацией) можно использовать в режиме повтора для перемещения и пересортировки массивов в памяти данных. В этом случае выборка исходных данных осуществляется последовательно за счет автоматического увеличения на 1 адреса источника (в DAR) в каждой следующей итерации, а формирование адреса приемника поддерживается широкими возможностями косвенной адресации процессора. Количество слов, которые будут перемещены, на единицу больше значения, содержащегося в счетчике повторов перед началом выполнения инструкции. После загрузки в конвейер (первое выполнение), инструкция становится эффективно одноцикловой.

Слов: 2 слова
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 2 цикла
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 19A (см. страницу 55)
Класс 19B (см. страницу 57)

MVKD Перемещение из непосредственного адреса памяти данных в память данных

Пример 1:

```
MVKD 300h, 0
```

	Before Instruction	After Instruction
DP	004	004
Data Memory		
0200h	ABCD	1234
0300h	1234	1234

Пример 2:

```
MVKD 1000h, *+AR5
```

	Before Instruction	After Instruction
AR5	01FF	0200
Data Memory		
1000h	1234	1234
0200h	ABCD	1234

MVMD Перемещение изMMR в память данных

Синтаксис: MVMD *MMR, dmad*

Операнды: MMR: Картированный в памяти регистр
$0 \leq dmad \leq 65\,535$

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	0	1	1	I	A	A	A	A	A	A	A
16-битная константа															

Выполнение:
dmad → EAR
Если (RC) ≠ 0
То
(MMR) → Dmem по адресу из EAR
(EAR) + 1 → EAR
Иначе
(MMR) → Dmem по адресу из EAR

Биты состояния: нет

Описание: Эта инструкция перемещает данные из картированного в памяти регистра *MMR* в память данных. Данные копируются в ячейку, адресуемую 16-разрядным непосредственным значением *dmad*. В режиме повтора, после загрузки в конвейер (первое выполнение), инструкция становится эффективно одноцикловой.

Слов: 2 слова

Циклов: 2 цикла

Классы: Класс 19A (см. страницу 55)

Пример:

```
MVMD AR7, 8000h
```

	Before Instruction		After Instruction
AR7	1234	AR7	1234
Data Memory			
8000h	ABCD	8000h	1234

MVMM Перемещение данных из MMR в MMR

Синтаксис: MVMM *MMRx, MMRy*

Операнды: MMRx: AR0 – AR7, SP
MMRy: AR0 – AR7, SP

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	0	1	1	1	M	M	R	X	M	M	R	Y

Регистр	MMRX/MMRY	Регистр	MMRX/MMRY
AR0	0000	AR5	0101
AR1	0001	AR6	0110
AR2	0010	AR7	0111
AR3	0011	SP	1000
AR4	0100		

Выполнение: (MMRx) → MMRy

Биты состояния: нет

Описание: Эта инструкция перемещает содержимое картированного в памяти регистра *MMRx* в картированный в памяти регистр *MMRy*. Разрешены к использованию только девять операндов: AR0 – AR7 и SP. Операция чтения из *MMRx* выполняется в фазе декодирования. Запись в *MMRy* осуществляется в фазе доступа.

Примечание – Эта инструкция неповторяемая.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 1 (см. страницу 32)

Пример:

```
MVMM SP, AR1
```

	Before Instruction	After Instruction
AR1	3EFF	0200
SP	0200	0200

MVPD Перемещение из памяти программ в память данных

Синтаксис: MVPD *pmad, Smem*

Операнды: Smem: Одиночный операнд памяти данных
$0 \leq pmad \leq 65\,535$

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	0	0	I	A	A	A	A	A	A	A
16-битная константа															

Выполнение:
pmad → PAR
Если (RC) ≠ 0
То
(Pmem по адресу из PAR) → Smem
(PAR) + 1 → PAR
Иначе
(Pmem по адресу из PAR) → Smem

Биты состояния: нет

Описание: Эта инструкция копирует слово из памяти программ, адресуемое 16-разрядным непосредственным операндом *pmad* в ячейку памяти данных, адресуемую *Smem*. Эту инструкцию можно использовать в режиме повтора для перемещения массивов, адресуемых 16-разрядным непосредственным значением, из памяти программ, в непрерывную область памяти данных, адресуемую *Smem*. Источник и приемник не должны находиться полностью на кристалле или полностью вне его. Если используется повтор, то после загрузки в конвейер (первое выполнение), инструкция становится эффективно одноцикловой. Кроме того, при повторе этой инструкции прерывания запрещены.

Слов: 2 слова
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 3 цикла
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 21A (см. страницу 60)
Класс 21B (см. страницу 61)

MVPD Перемещение из памяти программ в память данных

Пример 1:

```
MVPD 0FE00h, 5
```

	Before Instruction	After Instruction
DP	006	006
Program Memory		
FE00h	8A55	8A55
Data Memory		
0305h	FFFF	8A55

Пример 2:

```
MVPD 2000h, *AR7-0
```

	Before Instruction	After Instruction
AR0	0002	0002
AR7	0FFE	0FFC
Program Memory		
2000h	1234	1234
Data Memory		
0FFEh	ABCD	1234

NEG Инверсия аккумулятора

Синтаксис: NEG *src*[, *dst*]

Операнды: src, dst: A (аккумулятор A)
B (аккумулятор B)

Опкод:

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

Выполнение: (src) × –1 → dst

Биты состояния: зависит от OVM
Оказывает действие на C и OVdst (или OVsrc, если dst = src)

Описание: Эта инструкция вычисляет двоичное дополнение содержимого *src* (A или B) и сохраняет результат в *dst* или *src*, если *dst* не определен. Инструкция очищает бит переноса C в 0 для всех ненулевых значений аккумулятора. Если значение аккумулятора равно нулю, то бит переноса устанавливается в 1.
Если значение аккумулятора равно FF 8000 0000h, то операция NEG вызовет переполнение, поскольку двоичное дополнение к значению FF 8000 0000h выходит за границы младших 32 разрядов аккумулятора. Если OVM = 1, то *dst* принимает значение 00 7FFF FFFFh. Если OVM = 0, то *dst* принимает значение 00 8000 0000h. В любом случае, бит OV для *dst* устанавливается для индикации переполнения.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 1 (см. страницу 32)

Пример 1:

```
NEG A, B
```

	Before Instruction	After Instruction
A	FF FFFF F228	FF FFFF F228
B	00 0000 1234	00 0000 0DD8
OVA	0	0

Пример 2:

```
NEG B, A
```

	Before Instruction	After Instruction
A	00 0000 1234	FF 8000 0000
B	00 8000 0000	00 8000 0000
OVB	0	0

Пример 3:

```
NEG A
```

	Before Instruction	After Instruction
A	80 0000 0000	80 0000 0000
OVA	0	1
OVM	0	0

Пример 4:

```
NEG A
```

	Before Instruction	After Instruction
A	80 0000 0000	00 7FFF FFFF
OVA	0	1
OVM	1	1

NOP Нет операции

Синтаксис: NOP

Операнды: нет

Опкод:

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

Выполнение: нет

Биты состояния: нет

Описание: Никакие операции не выполняются, кроме увеличения PC (инкремент). Инструкция полезна при формировании конвейера или для организации задержек выполнения.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 1 (см. страницу 32)

Пример:

```
NOP
```

Нет выполняемой операции

NORM Нормализация

Синтаксис: NORM *src*[, *dst*]

Операнды: src, dst: A (аккумулятор A)
B (аккумулятор B)

Опкод:

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

Выполнение: (src) << TS → dst

Биты состояния: зависит от OVM и SXM
Оказывает действие OVdst (или OVsrc, если dst = src)

Описание: Эта инструкция нормализует число со знаком, содержащееся в *src*, и сохраняет полученное значение в *dst* или в *src*, если *dst* не определен. Нормализация числа с фиксированной запятой заключается в отделении мантиссы и экспоненты путем нахождении количества разрядов, содержащих раширение знака.
Эта инструкция позволяет производить одноцикловую нормализацию аккумулятора путем его сдвига на определенное число разрядов влево или вправо, чтобы оставить только один знаковый бит. Величина этого сдвига может быть получена выполнением инструкции вычисления экспоненты числа (EXP – см. страницу 130). Значение сдвига хранится в T(5-0) в виде числа в двоичном дополнительном коде. Допустимые значения сдвига лежат в интервале от –16 до 31. При выполнении нормализации сдвигателю необходима величина сдвига (в T) в фазе чтения конвейера; сама нормализация выполняется в фазе исполнения.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 1 (см. страницу 32)

Пример 1:

```
NORM A
```

	Before Instruction		After Instruction
A	FF FFFF F001	A	FF 8008 0000
T	0013	T	0013

Пример 2:

```
NORM B, A
```

	Before Instruction		After Instruction
A	FF FFFF F001	A	00 4214 1414
B	21 0A0A 0A0A	B	21 0A0A 0A0A
T	0FF9	T	0FF9

OR Логическое "ИЛИ" с аккумулятором

Синтаксис:

1: OR *Smem, src*
2: OR, *#lk*[, *SHFT*], *src*[, *dst*]
3: OR *#lk,* 16, *src*[, *dst*]
4: OR *src*[, *SHIFT*], [, *dst*]

Операнды:

Smem: Одиночный операнд памяти данных
src, dst: A (аккумулятор A)
B (аккумулятор B)

$0 \leq SHFT \leq 15$
$-16 \leq SHIFT \leq 15$
$0 \leq lk \leq 65\,535$

Опкоды:

1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	0	1	S	I	A	A	A	A	A	A	A

2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	0	1	0	0	S	H	F	T
16-битная константа															

3:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	0	1	1	0	0	1	0	0
16-битная константа															

4:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	1	0	1	S	H	I	F	T

Выполнение:

1: (Smem) OR (src(15-0)) → src
src(39-16) не изменяется
2: lk << SHFT OR (src) → dst
3: lk << 16 OR (src) → dst
4: (src or [dst]) OR (src) << SHIFT → dst

Биты состояния: нет

Описание: Эта инструкция выполняет операцию ИЛИ между *src* и: одиночным операндом памяти данных *Smem*; 16-разрядным непосредственным значением *lk*, сдвинутым влево; *dst*; или с самим собой. Результат сохраняется в *dst* или *src*, если *dst* не определен. Варианты сдвига приведены в синтаксисе инструкции. При положительном сдвиге (влево) младшие биты очищаются, старшие разряды не расширяются на знак. При отрицательном сдвиге (вправо) старшие биты не расширяются на знак.

OR Логическое "ИЛИ" с аккумулятором

Слов: Синтаксисы 1 и 4: 1 слово
Синтаксисы 2 и 3: 2 слова
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: Синтаксисы 1 и 4: 1 цикл
Синтаксисы 2 и 3: 2 цикла
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Синтаксис 1: Класс 3A (см. страницу 34)
Синтаксис 1: Класс 3B (см. страницу 35)
Синтаксисы 2 и 3: Класс 2 (см. страницу 33)
Синтаксис 4: Класс 1 (см. страницу 32)

Пример 1:

```
OR *AR3+, A
```

	Before Instruction		After Instruction
A	00 00FF 1200	A	00 00FF 1700
AR3	0100	AR3	0101
Data Memory			
0100h	1500	0100h	1500

Пример 2:

```
OR A, +3, B
```

	Before Instruction		After Instruction
A	00 0000 1200	A	00 0000 1200
B	00 0000 1800	B	00 0000 9800

ORM Логическое "ИЛИ" ячейки памяти с длинным непосредственным значением

Синтаксис: ORM *#lk, Smem*

Операнды: Smem: Одиночный операнд памяти даннх
$0 \le lk \le 65\,535$

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	0	0	1	I	A	A	A	A	A	A	A
16-битная константа															

Выполнение: lk OR (Smem) → Smem

Биты состояния: нет

Описание: Эта инструкция производит логическую операцию "ИЛИ" над одиночным операндом памяти данных *Smem* и 16-разрядной константой *lk* и сохраняет результат в *Smem*. Эта инструкция выполняет операцию "из памяти – в память".

Примечание – Эта инструкция неповторяемая.

Слов: 2 слова

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 2 цикла

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 18A (см. страницу 54)
Класс 18B (см. страницу 54)

Пример:

```
ORM 0404h, *AR4+
```

	Before Instruction		After Instruction
AR4	0100	AR4	0101
Data Memory			
0100h	4444	0100h	4444

POLY Полиномиальный анализ

Синтаксис: POLY *Smem*

Операнды: Smem: Одиночный операнд памяти данных

Опкод:

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

Выполнение: Round (A(32-16) × (T) + (B)) → A
(Smem) << 16 → B

Биты состояния: зависит от FRCT, OVM и SXM
Оказывает действие OVA

Описание: Эта инструкция сдвигает влево на 16 разрядов содержимое одиночного операнда памяти данных *Smem* и сохраняет результат в аккумулятор B. Параллельно, инструкция умножает старшую часть аккумулятора A (биты 32-16) на содержимое T, складывает полученное значение с содержимым аккумулятора B, округляет результат последней операции и сохраняет получившееся значение в аккумулятор A. Эта инструкция полезна для вычисления полиномов, когда один член полинома вычисляется за один цикл.

Слов: 1 слово

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 1 цикл

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 3A (см. страницу 34)
Класс 3B (см. страницу 35)

Пример:

```
POLY *AR3+%
```

	Before Instruction	After Instruction
A	00 1234 0000	00 0627 0000
B	00 0001 0000	00 2000 0000
T	5678	5678
AR3	0200	0201
Data Memory		
0200h	2000	2000

POPD Перемещение вершины стека в память данных

Синтаксис: POPD *Smem*

Операнды: Smem: Одиночный операнд памяти данных

Опкод:

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

Выполнение: (TOS) → Smem
(SP) + 1 → SP

Биты состояния: нет

Описание: Эта инструкция перемещает содержимое ячейки памяти данных, адресуемой SP, в ячейку памяти, опередяемую *Smem*. SP увеличивается на 1.

Слов: 1 слово

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 1 цикл

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 17A (см. страницу 52)
Класс 17B (см. страницу 53)

Пример:

```
POPD 10
```

	Before Instruction	After Instruction
DP	008	008
SP	0300	0301
Data Memory		
0300h	0092	0092
040Ah	0055	0092

POPM Перемещение вершины стека в MMR

Синтаксис: POPM *MMR*

Операнды: MMR: Картированный в памяти регистр

Опкод:

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

Выполнение: (TOS) → MMR
(SP) + 1 → SP

Биты состояния: нет

Описание: Эта инструкция перемещает содержимое ячейки памяти данных, адресуемой SP, в указанный картированный в памяти регистр *MMR*. SP увеличивается на 1.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 17A (см. страницу 52)

Пример:

```
POPM AR5
```

	Before Instruction		After Instruction
AR5	0055	AR5	0060
SP	03F0	SP	03F1
Data Memory			
03F0h	0060	03F0h	0060

PORTR Чтение данных из порта

Синтаксис: PORTR *PA, Smem*

Операнды: Smem: Одиночный операнд памяти данных
$0 \le PA \le 65\,535$

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	1	0	0	I	A	A	A	A	A	A	A
Адрес порта															

Выполнение: (PA) → Smem

Биты состояния: нет

Описание: Эта инструкция считывает 16-разрядное значение из внешнего порта ввода/вывода *PA* (16-разрядный непосредственный адрес) в указанную ячейку памяти данных *Smem*. Сигнал IS# устанавливается в низкий уровень для индикации обращения к пространству ввода/вывода; IOSTRB# и READY синхронизируют доступ подобно тому, как это происходит при чтении внешней памяти.

Слов: 1 слово

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

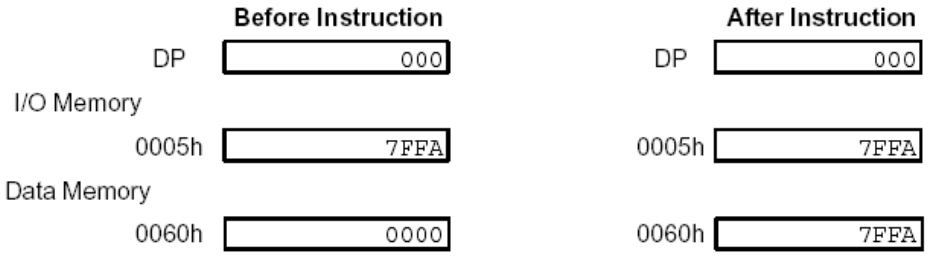
Циклов: 1 цикл

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 27A (см. страницу 72)
Класс 27B (см. страницу 72)

Пример:

```
PORTR 05, INDAT ; INDAT .equ 60h
```



PORTW Запись данных в порт

Синтаксис: PORTW *Smem, PA*

Операнды: Smem: Одиночный операнд памяти данных
$0 \le PA \le 65\,535$

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	1	0	1	I	A	A	A	A	A	A	A
Адрес порта															

Выполнение: (Smem) → PA

Биты состояния: нет

Описание: Эта инструкция записывает 16-разрядное значение из указанной ячейки памяти данных *Smem* во внешний порт ввода/вывода *PA*. Сигнал IS# устанавливается в низкий уровень для индикации обращения к пространству ввода/вывода; IOSTRB# и READY синхронизируют доступ подобно тому, как это происходит при записи во внешнюю память.

Слов: 2 слова

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

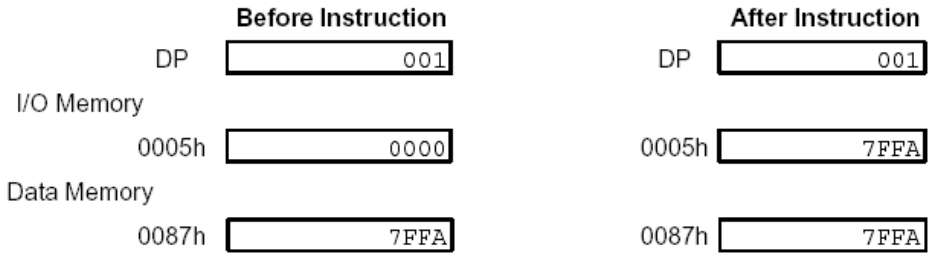
Циклов: 2 цикла

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 28A (см. страницу 73)
Класс 28B (см. страницу 73)

Пример:

```
PORTW OUTDAT, 5h ; OUTDAT .equ 07h
```



PSHD Перемещение из памяти данных в стек

Синтаксис: PSHD *Smem*

Операнды: Smem: Одиночный операнд памяти данных

Опкод:

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

Выполнение: (SP) – 1 → SP
(Smem) → TOS

Биты состояния: нет

Описание: После уменьшения SP на 1, эта инструкция сохраняет содержимое ячейки памяти *Smem* в ячейку памяти данных, адресуемую SP. SP считывается в течение фазы декодирования; сохранение происходит в фазе доступа.

Слов: 1 слово
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 1 цикл
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 16A (см. страницу 50)
Класс 16B (см. страницу 51)

Пример:

```
PSHD *AR3+
```

	Before Instruction		After Instruction
AR3	0200	AR3	0201
SP	8000	SP	7FFF
Data Memory			
0200h	07FF	0200h	07FF
7FFFh	0092	7FFFh	07FF

PSHM Перемещение из MMR данных в стек

Синтаксис: PSHM *MMR*

Операнды: MMR: Картированный в памяти регистр

Опкод:

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

Выполнение: (SP) – 1 → SP
(MMR) → TOS

Биты состояния: нет

Описание: После уменьшения SP на 1, эта инструкция сохраняет содержимое картированного в памяти регистра *MMR* в ячейку памяти данных, адресуемую SP.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 16A (см. страницу 50)

Пример:

```
PSHM BRC
```

	Before Instruction	After Instruction
BRC	1234	1234
SP	2000	1FFF
Data Memory		
1FFFh	07FF	1234

RC[D] Условный возврат

Синтаксис: RC[D] *cond*[, *cond*[, *cond*]]

Операнды: Следующая таблица содержит список условий (cond) для этой инструкции

Условие	Описание	Код условия	Условие	Описание	Код условия
BIO	BIO# low	0000 0011	NBIO	BIO# high	0000 0010
C	C = 1	0000 1100	NC	C = 0	0000 1000
TC	TC = 1	0011 0000	NTC	TC = 0	0010 0000
AEQ	(A) = 0	0100 0101	BEQ	(B) = 0	0100 1101
ANEQ	(A) ≠ 0	0100 0100	BNEQ	(B) ≠ 0	0100 1100
AGT	(A) > 0	0100 0110	BGT	(B) > 0	0100 1110
AGEQ	(A) ≥ 0	0100 0010	BGEQ	(B) ≥ 0	0100 1010
ALT	(A) < 0	0100 0011	BLT	(B) < 0	0100 1011
ALEQ	(A) ≤ 0	0100 0111	BLEQ	(B) ≤ 0	0100 1111
AOV	A переполнен	0111 0000	BOV	B переполнен	0111 1000
ANOV	A не переп-н	0110 0000	BNOV	B не переп-н	0110 1000
UNC	безусловный	0000 0000			

Опкод:

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

Выполнение: Если (условие(я))
то
(TOS) → PC
(SP) + 1 → SP
иначе
(PC) + 1 → PC

Биты состояния: нет

Описание: Если выполнены все указанные условия, инструкция заменяет содержимое PC значением вершины стека и увеличивает SP на 1. Если условие(я) невыполнено(ы), то происходит только увеличение счетчика команд (PC) на 1.
Если инструкция выполняется с задержкой (при указании суффикса D), то сначала из памяти программ выбираются и исполняются две однословные инструкции или одна двухсловная. Эти инструкции не оказывают влияния на проверяемые условия.
Эта инструкция позволяет осуществлять проверку нескольких условий перед передачей управления в другую секцию программы. Инструкция может проверять условия индивидуально или в комбинации с другими условиями. Можно компоновать условия только из одной группы следующим образом:

Группа 1: Допускается выбрать до двух условий. Каждое условие должно принадлежать разным категориям (A или B, см. таблицу ниже). Не допускается использование в инструкции двух условий из одной категории. Например, одновременно можно тестировать EQ и OV, однако, нельзя выполнить одновременную

проверку GT и NEQ. Для обоих условий аккумулятор должен быть одним и тем же. Не допускается проверка двух условий для разных аккумуляторов в одной инструкции. Например, разрешена одновременная проверка AGT и AOV, однако, не допускается использование в одной инструкции условий AGT и BOV.

Группа 2: Допускается выбрать до трех условий. Каждое из трех условий должно быть из различных категорий (A, B или C). Не допускается использование двух или более условий из одной категории, т. е. допускается одновременная проверка TC, C и BIO#, однако, не допускается одновременная проверка NTC, C и NC.

Сочетаемость условий для этой инструкции

Группа 1		Группа 2		
Категория A	Категория B	Категория A	Категория B	Категория C
EQ	OV	TC	C	BIO
NEQ	NOV	NTC	NC	NBIO
LT				
LEQ				
GT				
GEQ				

Примечание – Эта инструкция неповторяемая.

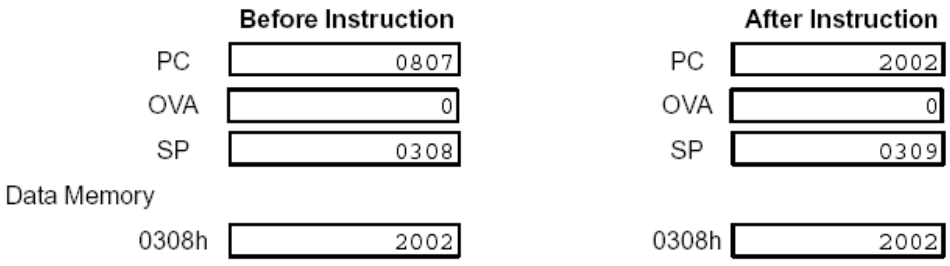
Слов: 1 слово

Циклов: 5 циклов (условие выполнено)
3 цикла (условие невыполнено)
3 цикла (для задержанных)

Классы: Класс 32 (см. страницу 78)

Пример:

```
RC AGEQ, ANOV   ; возврат выполняется, если
                ; содержимое аккумулятора A
                ; положительно и бит OVA равен нулю
```



READA Чтение памяти программ, адресуемой аккумулятором A и сохранение в памяти данных

Синтаксис: READA *Smem*

Операнды: Smem: Одиночный операнд памяти данных

Опкод:

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

Выполнение:
A → PAR
Если ((RC) ≠ 0)
(Pmem (по адресу из PAR)) → Smem
(PAR) + 1 → PAR
(RC) – 1 → RC
Иначе
(Pmem(по адресу из PAR)) → Smem

Биты состояния: нет

Описание: Эта инструкция перемещает слово из ячейки памяти программ, определяемой младшей частью аккумулятора A(биты 15-0), в ячейку памяти данных, определяемую *Smem*.
Эту инструкцию можно использовать в режиме повтора для перемещения из памяти программ массивов со стартовым адресом из аккумулятора A, в непрерывную область памяти данных, адресуемую (в формате косвенной адресации) *Smem*. Источник и приемник не должны находиться полностью на кристалле или полностью вне его. Если используется повтор, то после загрузки в конвейер (первое выполнение), инструкция становится эффективно одноцикловой.

Слов: 1 слово

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

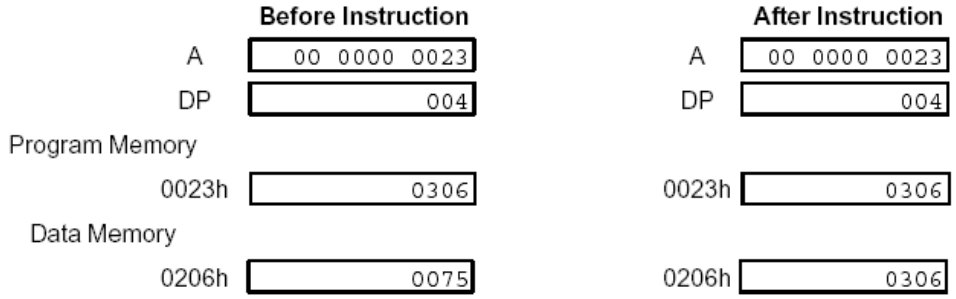
Циклов: 5 циклов

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 25A (см. страницу 68)
Класс 25B (см. страницу 69)

Пример:

```
READA 6
```



RESET Программный сброс

Синтаксис: RESET

Операнды: нет

Опкод:

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

Выполнение: Поля PMST, ST0 и ST1 загружаются следующими значениями:

(IPTR) << 7 → PC	0 → OVA	0 → OVB
1 → C	1 → TC	0 → ARP
0 → DP	1 → SXM	0 → ASM
0 → BRAF	0 → HM	1 → XF
0 → C16	0 → FRCT	0 → CMPT
0 → CPL	1 → INTM	0 → IFR
0 → OVM		

Биты состояния: все затрагиваемые биты перечислены в пункте "Выполнение"

Описание: Эта инструкция выполняет немаскируемый программный сброс, который может быть использован для приведения процессора в известное состояние. Когда выполняется инструкция сброса, происходят операции, перечисленные в пункте "Выполнение". При программном сбросе вход MP/MC# не опрашивается (в отличие от аппаратного сброса). Инициализация IPTR и периферийных регистров также отличается от инициализации с использованием RS#. Эта инструкция не зависит от INTM; но устанавливает INTM в 1, запрещая прерывания.

Примечание – Эта инструкция не повторяемая.

Слов: 1 слово

Циклов: 3 цикла

Классы: Класс 35 (см. страницу 79)

Пример:

```
RESET
```

	Before Instruction	After Instruction
PC	0025	0080
INTM	0	1
IPTR	1	1

RET[D] Возврат

Синтаксис: RET[D]

Операнды: нет

Опкод:

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

Выполнение: (TOS) → PC
(SP) + 1 → SP

Биты состояния: нет

Описание: Эта инструкция заменяет содержимое PC 16-разрядным значением из вершины стека (TOS). SP увеличивается на 1. Если возврат выполняется с задержкой (при указании в синтаксисе суффикса D), то выбираются и исполняются одна двухсловная инструкция или две однословные, расположенные после инструкции возврата.
По сути, это частный случай инструкции RC[D] (страница 198). Т. е. RET[D] ≡ RC[D] UNC (см. Опкоды и таблицу условий).

Примечание – Эта инструкция неповторяемая.

Слов: 1 слово

Циклов: 5 циклов
3 цикла (для задержанных)

Классы: Класс 32 (см. страницу 78)

Пример:

```
RET
```

	Before Instruction	After Instruction
PC	2112	1000
SP	0300	0301
Data Memory		
0300h	1000	1000

RETE[D] Возврат из прерывания с разрешением прерываний

Синтаксис: RETE[D]

Операнды: нет

Опкод:

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

Выполнение: (TOS) → PC
(SP) + 1 → SP
0 → INTM

Биты состояния: оказывает действие на INTM

Описание: Эта инструкция заменяет содержимое PC 16-разрядным значением из вершины стека (TOS). Выполнение программы продолжается с этого адреса. SP увеличивается на 1. Эта инструкция автоматически очищает бит запрета прерываний INTM в ST1. (Очистка этого бита разрешает прерывания). Если возврат выполняется с задержкой (при указании в синтаксисе суффикса D), то выбираются и исполняются одна двухсловная инструкция или две однословные, расположенные после инструкции возврата.

Примечание – Эта инструкция неповторяемая.

Слов: 1 слово

Циклов: 5 циклов
3 цикла (для задержанных)

Классы: Класс 32 (см. страницу 78)

Пример:

```
RETE
```

	Before Instruction		After Instruction
PC	01C3	PC	0110
SP	2001	SP	2002
ST1	xCxx	ST1	x4xx
Data Memory			
2001h	0110	2001h	0110

RETF[D] Быстрый возврат из прерывания с разрешением прерываний

Синтаксис: RETF[D]

Операнды: нет

Опкод:

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

Выполнение:
(RTN) → PC
(SP) + 1 → SP
0 → INTM

Биты состояния: оказывает действие на INTM

Описание: Эта инструкция заменяет содержимое PC 16-разрядным значением из RTN. RTN хранит адрес, к которому должна вернуться программа после обработки прерывания. При таком возврате, вместо считывания в PC из стека, PC загружается из RTN. SP увеличивается на 1. Эта инструкция автоматически очищает бит запрета прерываний INTM в ST1. (Очистка этого бита разрешает прерывания). Если возврат выполняется с задержкой (при указании в синтаксисе суффикса D), то выбираются и исполняются одна двухсловная инструкция или две однословные, расположенные после инструкции возврата.

Примечание – Эту инструкцию можно использовать только в том случае, если во время работы подпрограммы обработки текущего прерывания не выполнялись никакие вызовы и не запускались подпрограммы обработки других прерываний.
Эта инструкция неповторяемая.

Слов: 1 слово

Циклов: 3 цикла
1 цикл (для задержанных)

Классы: Класс 33 (см. страницу 79)

Пример:

```
RETF
```

	Before Instruction	After Instruction
PC	01C3	0110
SP	2001	2002
ST1	xCxx	x4xx
Data Memory		
2001h	0110	0110

RND Округлене аккумулятора

Синтаксис: RND *src*[, *dst*]

Операнды: src, dst: A (аккумулятор A)
B (аккумулятор B)

Опкод:

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

Выполнение: (src) + 8000h → dst

Биты состояния: зависит от OVM

Описание: Эта инструкция округляет содержимое *src* (аккумулятора A или B) путем прибавления числа 2^{15}. Полученное значение сохраняется в *dst* или в *src*, если *dst* не определен.

Примечание – Эта инструкция неповторяемая.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 1 (см. страницу 32)

Пример 1:

```
RND A, B
```

	Before Instruction		After Instruction
A	FF FFFF FFFF	A	FF FFFF FFFF
B	00 0000 0001	B	00 0000 7FFF
OVM	0	OVM	0

Пример 2:

```
RND A
```

	Before Instruction		After Instruction
A	00 7FFF FFFF	A	00 7FFF FFFF
OVM	1	OVM	1

ROL Вращение аккумулятора влево

Синтаксис: ROL *src*

Операнды: src: A (аккумулятор A)
B (аккумулятор B)

Опкод:

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

Выполнение: (C) → src(0)
(src(30-0)) → src(31-1)
(src(31)) → C
0 → src(39-32)

Биты состояния: зависит от C
Оказывает влияние на C

Описание: Эта инструкция сдвигает каждый бит *src* влево на 1 бит. Значение бита переноса C перед выполнением инструкции сдвигается в младший разряд *src*. Затем старший разряд *src* сдвигается в C. Биты безопасности *src* очищаются.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 1 (см. страницу 32)

Пример:

```
ROL A
```

	Before Instruction		After Instruction
A	5F B000 1234	A	00 6000 2468
C	0	C	1

ROLTC Вращение аккумулятора влево с использованием TC

Синтаксис: ROLTC *src*

Операнды: src: A (аккумулятор A)
B (аккумулятор B)

Опкод:

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

Выполнение: (TC) → src(0)
(src(30-0)) → src(31-1)
(src(31)) → C
0 → src(39-32)

Биты состояния: зависит от C
Оказывает влияние на TC

Описание: Эта инструкция сдвигает каждый бит *src* влево на 1 бит. Значение бита TC перед выполнением инструкции сдвигается в младший разряд *src*. Затем старший разряд сдвигается в C. Биты безопасности *src* очищаются.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 1 (см. страницу 32)

Пример:

```
ROLTC A
```

	Before Instruction		After Instruction
A	81 C000 5555	A	00 8000 AAAB
C	x	C	1
TC	1	TC	1

ROR Вращение аккумулятора вправо

Синтаксис: ROR *src*

Операнды: src: A (аккумулятор A)
B (аккумулятор B)

Опкод:

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

Выполнение: (C) → src(31)
(src(31-1)) → src(30-0)
(src(0)) → C
0 → src(39-32)

Биты состояния: зависит от C
Оказывает влияние на C

Описание: Эта инструкция сдвигает каждый бит *src* вправо на 1 бит. Значение бита переноса C перед выполнением инструкции сдвигается в старший разряд *src*. Затем младший разряд сдвигается в C. Биты безопасности *src* очищаются.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 1 (см. страницу 32)

Пример:

```
ROR A
```

	Before Instruction		After Instruction
A	7F B000 1235	A	00 5800 091A
C	0	C	1

RPT Повтор следующей инструкции

Синтаксис:
1: RPT *Smem*
2: RPT *#K*
3: RPT *#lk*

Операнды:
Smem: Одиночный операнд памяти данных
$0 \le K \le 255$
$0 \le lk \le 65\,535$

Опкоды:

1:

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

2:

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

3:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	1	1	1	0	0	0	0
16-битная константа															

Выполнение:
1: (Smem) → RC
2: K → RC
3: lk → RC

Биты состояния: нет

Описание: После выполнения инструкции счетчик повторов (RC) загружается числом необходимых итераций. Количество итераций (n) может задаваться в виде 16-разрядного одиночного операнда памяти данных *Smem*; 8- или 16-разрядной константы K или *lk* соответственно. Инструкция, расположенная после RPT, будет выполнена (n + 1) раз. Пока RC уменьшается, доступ к нему запрещен.

Примечание – Эта инструкция неповторяемая.

Слов:
Синтаксисы 1 и 2: 1 слово
Синтаксис 3: 2 слова
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов:
Синтакис 1: 3 цикла
Синтаксис 2: 1 цикл
Синтаксис 3: 2 цикла
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

RPT Повтор следующей инструкции

Классы: Синтаксис 1: Класс 5A (см. страницу 38)
Синтаксис 1: Класс 5B (см. страницу 38)
Синтаксис 2: Класс 1 (см. страницу 32)
Синтаксис 3: Класс 2 (см. страницу 33)

Пример 1:

```
RPT DAT127 ; DAT127 .EQU 0FFF
```

	Before Instruction		After Instruction
RC	0	RC	000C
DP	031	DP	031
Data Memory			
0FFFh	000C	0FFFh	000C

Пример 2:

```
RPT #2 ; Повтор следующей инструкции 3 раза
```

	Before Instruction		After Instruction
RC	0	RC	0002

Пример 3:

```
RPT #1111h ; Повтор следующей инструкции 4370 раз
```

	Before Instruction		After Instruction
RC	0	RC	1111

RPTB[D] Повтор блока

Синтаксис: RPTB[D] *pmad*

Операнды: $0 \le pmad \le 65\,535$

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	Z	0	0	1	1	1	0	0	1	0
16-битная константа															

Выполнение:
1 → BRAF
Если (задержка), то
(PC) + 4 → RSA
Иначе
(PC) + 2 → RSA
pmad → REA

Биты состояния: Оказывает влияние на BRAF

Описание: Эта инструкция повторяет блок инструкций определенное число раз, задаваемое, содержимым картированного в памяти регистра-счетчика повтора блока (BRC). Этот регистр должен быть загружен до начала работы инструкции. После выполнения инструкции регистр стартового адреса повтора блока (RSA) будет загружен значением PC + 2 (или PC + 4, если инструкция выполняется с задержкой); регистр конечного адреса повтора блока (REA) загружается адресом памяти программ (*pmad*).
Эта инструкция является прерываемой. Циклы повтора однократной инструкции могут быть включены в программный код повторяемых блоков. Вложения внутри повторяемого блока инструкций (в виде подпрограмм обработки прерываний или повторов однократной инструкции) допустимы при соблюдении следующих условий:

– BRC, RSA и REA должны быть надлежащим образом сохранены и восстановлены.

– Флаг активности повтора блока (BRAF) должен быть правильно установлен.

Если повтор блока выполняется с задержкой (при указании в синтаксисе суффикса D), то выбираются и исполняются одна двухсловная инструкция или две однословные, расположенные после этой инструкции.

Примечание – Повтор блока может быть отменен очисткой бита BRAF.
Эта инструкция неповторяемая.

Слов: 2 слова

Циклов: 4 цикла
2 цикла (для задержанных)

RPTB[D] Повтор блока

Классы: Класс 29A (см. страницу 74)

Пример 1:

```
ST #99, BRC
RPTB end_block - 1
       ; end_block = Bottom of Block
```

	Before Instruction		After Instruction
PC	1000	PC	1002
BRC	1234	BRC	0063
RSA	5678	RSA	1002
REA	9ABC	REA	end_block - 1

Пример 2:

```
ST #99, BRC ;Выполнение блока 100 раз
RPTBD end_block - 1
MVDM POINTER, AR1
       ; инициализация указателя
       ; end_block ; Bottom of Block
```

	Before Instruction		After Instruction
PC	1000	PC	1004
BRC	1234	BRC	0063
RSA	5678	RSA	1004
REA	9ABC	REA	end_block - 1

RPTZ Повтор следующей инструкции с очисткой аккумулятора

Синтаксис: RPTZ *dst*, *#lk*

Операнды: dst: A (аккумулятор A)
B (аккумулятор B)
$0 \le lk \le 65535$

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	D	0	1	1	1	0	0	0	1
16-битная константа															

Выполнение: 0 → dst
lk → RC

Биты состояния: нет

Описание: Эта инструкция очищает *dst* и повторяет следующую инструкцию (n + 1) раз, где n – содержимое счетчика повтора (RC). Значение RC задается 16-разрядной константой *lk*.

Слов: 2 слова

Циклов: 2 цикла

Классы: Класс 2 (см. страницу 33)

Пример:

```
RPTZ A, 1023  ; Повторение следующей инструкции 1024 раз
STL A, *AR2+
```

	Before Instruction		After Instruction
A	0F FE00 8000	A	00 0000 0000
RC	0000	RC	03FF

RSBX Сброс бита регистра состояния

Синтаксис: RSBX *N, SBIT*

Операнды: $0 \leq SBIT \leq 15$
N = 0 или 1

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	N	0	1	0	1	1	S	B	I	T

Выполнение: 0 → STN(SBIT)

Биты состояния: нет

Описание: Эта инструкция очищает в 0 указанный бит в регистре состояния 0 или 1. *N* определяет регистр состояния, а *SBIT* определяет бит, который будет обнулен. Символьные имена разрядов также распознаются транслятором и могут использоваться в качестве операндов вместо *N* и *SBIT* (см. пример 1).

Примечание – Эта инструкция неповторяемая.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 1 (см. страницу 32)

Пример 1:

```
RSBX SXM ; SXM означает что : n=1   и   SBIT=8
```

	Before Instruction		After Instruction
ST1	35CD	ST1	34CD

Пример 2:

```
RSBX 1,8
```

	Before Instruction		After Instruction
ST1	35CD	ST1	34CD

SACCD Условное сохранение аккумулятора

Синтаксис: SACCD *src, Xmem, cond*

Операнды:

src:	A (аккумулятор A)	
	B (аккумулятор B)	
Xmem:	Двойной операнд памяти данных	

Следующая таблица содержит список условий (cond) для этой инструкции

Условие	Описание	Код условия	Условие	Описание	Код условия
AEQ	(A) = 0	0101	BEQ	(B) = 0	1101
ANEQ	(A) ≠ 0	0100	BNEQ	(B) ≠ 0	1100
AGT	(A) > 0	0110	BGT	(B) > 0	1110
AGEQ	(A) ≥ 0	0010	BGEQ	(B) ≥ 0	1010
ALT	(A) < 0	0011	BLT	(B) < 0	1011
ALEQ	(A) ≤ 0	0111	BLEQ	(B) ≤ 0	1111

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	1	1	1	S	X	X	X	X	C	O	N	D

Выполнение:

Если (условие)
то
(src) << (ASM – 16) → Xmem
иначе
(Xmem) → Xmem

Биты состояния: зависит от ASM и SXM

Описание: Если условие выполнено, то инструкция сохраняет *src*, сдвинутый влево на (ASM – 16). Значение величины сдвига располагается в ячейке памяти, определяемой *Xmem*. Если условие невыполнено, то инструкция считывает содержимое *Xmem* и записывает его обратно в ту же ячейку, таким образом, операнд *Xmem* остается прежним. Независимо от указанного условия операнд *Xmem* всегда считывается и обновляется.

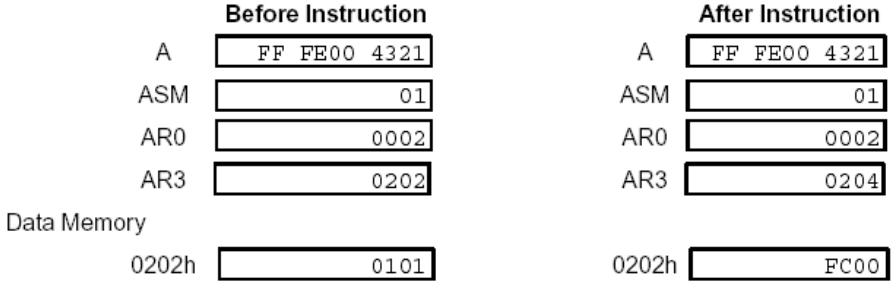
Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 15 (см. страницу 49)

Пример:

```
SACCD A, *AR3+0%, ALT
```



SAT Насыщение аккумулятора

Синтаксис: SAT *src*

Операнды: src: A (аккумулятор A)
B (аккумулятор B)

Опкод:

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

Выполнение: Насыщенный (src) → src

Биты состояния: Оказывает влияние на OVsrc

Описание: Независимо от значения OVM, инструкция позволяет насыщать содержимое 32 разрядов аккумулятора.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 1 (см. страницу 32)

Пример 1:

```
SAT B
```

	Before Instruction	After Instruction
B	71 2345 6789	00 7FFF FFFF
OVB	x	1

Пример 2:

```
SAT A
```

	Before Instruction	After Instruction
A	F8 1234 5678	FF 8000 0000
OVA	x	1

Пример 3:

```
SAT B
```

	Before Instruction	After Instruction
B	00 0012 3456	00 0012 3456
OVB	x	0

SFTA Арифметический сдвиг аккумулятора

Синтаксис: SFTA *src, SHIFT*[, *dst*]

Операнды: src, dst: A (аккумулятор A)
B (аккумулятор B)
$-16 \le SHIFT \le 15$

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	S	D	0	1	1	S	H	I	F	T

Выполнение:
Если SHIFT < 0
То
(src((–SHIFT) – 1)) → C
(src(39-0)) << SHIFT → dst
Если SXM = 1
То
(src(39)) → dst(39 – (39 + (SHIFT + 1)))
[или src(39 – (39 + (SHIFT + 1))),
если *dst* не определен]
Иначе
0 → dst(39 – (39+(SHIFT+1)))
[или src(39 – (39+(SHIFT+1))),
если *dst* не определен]
Иначе
(src(39 – SHIFT)) → C
(src) << SHIFT → dst
0 → dst((SHIFT – 1) – 0)
[или src((SHIFT – 1) – 0),
если *dst* не определен]

Биты состояния: Зависит от SXM и OVM
Оказывает действие на C и OVdst (или OVsrc, если dst = src)

Описание: Эта инструкция арифметически сдвигает *src* и сохраняет результат в *dst* или в *src*, если *dst* не определен. Выполнение инструкции зависит от значения SHIFT:

1 Если значение SHIFT меньше нуля, то инструкция выполняется следующим образом:

– *src*((–SHIFT) – 1) копируется в бит переноса C.
– Если SXM равно 1, инструкция выполняет арифметический сдвиг вправо и старшие разряды *src* сдвигаются в *dst*(39 – (39 + (SHIFT + 1))).
– Если SXM равно нулю, то ноль записывается в *dst*(39 – (39 + (SHIFT + 1))).

2 Если значение SHIFT больше нуля, то инструкция выполняется следующим образом:

– *src*(39 – SHIFT) копируется в бит переноса C.
– Инструкция производит арифметический сдвиг влево.
– 0 записывается в *dst*((SHIFT – 1) – 0).

SFTA Арифметический сдвиг аккумулятора

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 1 (см. страницу 32)

Пример 1:

```
SFTA A, -5, B
```

	Before Instruction	After Instruction
A	FF 8765 0055	FF 8765 0055
B	00 4321 1234	FF FC3B 2802
C	x	1
SXM	1	1

Пример 2:

```
SFTA B, +5
```

	Before Instruction	After Instruction
B	80 AA00 1234	15 4002 4680
C	0	1
OVM	0	0
SXM	0	0

SFTC Условный сдвиг аккумулятора

Синтаксис: SFTC *src*

Операнды: src: A (аккумулятор A)
B (аккумулятор B)

Опкод:

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

Выполнение:
Если (src) = 0
То
1 → TC
Иначе
Если (src(31)) XOR (src(30)) = 0
То (два знаковых бита)
0 → TC
(src) << 1 → src
Иначе (только один знаковый бит)
1 → TC

Биты состояния: Оказывает влияние на TC

Описание: Если *src* имеет два знаковых разряда, то инструкция производит сдвиг 32-разрядного *src* влево на 1 бит. При наличие двух знаковых разрядов бит TC очищается в ноль, в ином случае устанавливается в 1.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 1 (см. страницу 32)

Пример:

```
SFTC A
```

	Before Instruction	After Instruction
A	FF FFFF F001	FF FFFF E002
TC	x	0

SFTL Логический сдвиг аккумулятора

Синтаксис: SFTL *src*, *SHIFT*[, *dst*]

Операнды: src, dst: A (аккумулятор A)
B (аккумулятор B)
-16 ≤ SHIFT ≤ 15

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	1	1	1	S	H	I	F	T

Выполнение:
Если SHIFT < 0
То
src((–SHIFT) – 1) → C
src(31-0) << SHIFT → dst
0 → dst(39 – (31 + (SHIFT + 1)))
Если SHIFT = 0
То
0 → C
Иначе
src(31 – (SHIFT – 1)) → C
src((31 – SHIFT) – 0) << SHIFT → dst
0 → dst((SHIFT – 1) – 0)
[или src((SHIFT – 1) – 0), если dst не определен]
0 → dst(39-32)
[или src(39-32), если dst не определен]

Биты состояния: Оказывает влияние на C

Описание: Эта инструкция осуществляет логический сдвиг *src* и сохраняет полученный результат в *dst* или в *src*, если *dst* не определен. Биты безопасности *dst* или *src*, если *dst* не определен очищаются. Выполнение инструкции зависит от значение SHIFT следующим образом:

1 Если значение SHIFT меньше нуля, то инструкция выполняется следующим образом:
- *src*((–SHIFT) – 1) копируется в бит переноса C.
- Инструкция производит логический сдвиг вправо.
- Ноль записывается в *dst*(39 – (31 + (SHIFT + 1))).

2 Если значение SHIFT больше нуля, то инструкция выполняется следующим образом:
- *src*(31 – (SHIFT – 1)) копируется в бит переноса C.
- Инструкция производит логический сдвиг влево.
- 0 записывается в *dst*((SHIFT – 1) – 0).

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 1 (см. страницу 32)

FTL Логический сдвиг аккумулятора

Пример 1: `SFTL A, -5, B`

	Before Instruction	After Instruction
A	FF 8765 0055	FF 8765 0055
B	FF 8000 0000	00 043B 2802
C	0	1

Пример 2: `SFTL B, +5`

	Before Instruction	After Instruction
B	80 AA00 1234	00 4002 4680
C	0	1

SQDST Квадрат расстояния

Синтаксис: SQDST *Xmem, Ymem*

Операнды: Xmem, Ymem: Двойные операнды памяти данных

Опкод:

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

Выполнение: (A(32-16)) × (A(32-16)) + (B) → B
((Xmem) – (Ymem)) << 16 → A

Биты состояния: Зависит от OVM, FRCT и SXM
Оказывает влияние на C, OVA и OVB

Описание: Используемая в режиме повтора, эта инструкция вычисляет квадрат расстояния между двумя векторами. Старшая часть аккумулятора A (биты 32-16) возводится в квадрат, произведение прибавляется к аккумулятору B и полученная сумма сохраняется в аккумуляторе B. *Ymem* вычитается из *Xmem*, разница сдвигается влево на 16 разрядов и полученный результат сохраняется в аккумуляторе A.
При выполнении этой инструкции, возведение квадрат (A(32-16)) производится до сохранения в A результата вычитания.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 7 (см. страницу 41)

Пример:

```
SQDST *AR3+, AR4+
```

	Before Instruction	After Instruction
A	FF ABCD 0000	FF FFAB 0000
B	00 0000 0000	00 1BB1 8229
FRCT	0	0
AR3	0100	0101
AR4	0200	0201
Data Memory		
0100h	0055	0055
0200h	00AA	00AA

SQUR Возведение в квадрат

Синтаксис:
1: SQUR *Smem, dst*
2: SQUR *A, dst*

Операнды:
Smem: Одиночный операнд памяти данных
dst: A (аккумулятор A)
B (аккумулятор B)

Опкоды:

1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	1	1	D	I	A	A	A	A	A	A	A

2:

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

Выполнение:
1: (Smem) → T
(Smem) × (Smem) → dst
2: (A(32-16)) × (A(32-16)) → dst

Биты состояния:
Зависит от OVM и FRCT
Оказывает влияние на OVsrc

Описание:
Эта инструкция вычисляет квадрат одиночного операнда памяти данных *Smem* или старшей части аккумулятора A (биты 32-16) и сохраняет результат в *dst*. В синтаксисе 1 *Smem* сохраняется в T; в синтаксисе 2 T не используется и не изменяется.

Слов:
1 слово
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов:
1 цикл
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы:
Синтаксис 1: Класс 3A (см. страницу 34)
Синтаксис 1: Класс 3B (см. страницу 35)
Синтаксис 2: Класс 1 (см. страницу 32)

SQUR Возведение в квадрат

Пример 1:

```
SQUR 30, B
```

	Before Instruction	After Instruction
B	00 0000 01F4	00 0000 00E1
T	0003	000F
FRCT	0	0
DP	006	006
Data Memory		
031Eh	000F	000F

Пример 2:

```
SQUR A, B
```

	Before Instruction	After Instruction
A	00 000F 0000	00 000F 0000
B	00 0101 0101	00 0000 01C2
FRCT	1	1

SQURA Возведение в квадрат с накоплением

Синтаксис: SQURA *Smem, src*

Операнды:
Smem: Одиночный операнд памяти данных
dst: A (аккумулятор A)
B (аккумулятор B)

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	0	0	S	I	A	A	A	A	A	A	A

Выполнение:
(Smem) → T
(Smem) × (Smem) + (src) → src

Биты состояния:
Зависит от OVM и FRCT
Оказывает влияние на OVsrc

Описание: Эта инструкция сохраняет значение операнда памяти данных *Smem* в T, затем вычисляет квадрат *Smem*, прибавляет произведение к *src* и сохраняет полученный результат в *src*.

Слов:
1 слово
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов:
1 цикл
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы:
Класс 3A (см. страницу 34)
Класс 3B (см. страницу 35)

Пример 1:

```
SQURA 30, B
```

	Before Instruction	After Instruction
B	00 0320 0000	00 0320 00E1
T	0003	000F
FRCT	0	0
DP	006	006
Data Memory		
031Eh	000F	000F

Пример 2:

```
SQURA *AR3+, A
```

	Before Instruction	After Instruction
A	00 0000 01F4	00 0000 02D5
T	0003	000F
FRCT	0	0
AR3	031E	031F
Data Memory		
031Eh	000F	000F

SQURS Возведение в квадрат с вычитанием

Синтаксис: SQURS *Smem, src*

Операнды:
Smem: Одиночный операнд памяти данных
dst: A (аккумулятор A)
B (аккумулятор B)

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	0	1	S	I	A	A	A	A	A	A	A

Выполнение:
(Smem) → T
(src) – (Smem) × (Smem) → src

Биты состояния:
Зависит от OVM и FRCT
Оказывает влияние на OVsrc

Описание: Эта инструкция сохраняет значение операнда памяти данных *Smem* в T, затем вычисляет квадрат *Smem*, вычитает произведение из *src* и сохраняет полученное значение в *src*.

Слов:
1 слово
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов:
1 цикл
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы:
Класс 3A (см. страницу 34)
Класс 3B (см. страницу 35)

Пример 1:

```
SQURS 9, A
```

	Before Instruction	After Instruction
A	00 014B 5DB0	00 0000 0320
T	8765	1234
FRCT	0	0
DP	006	006
Data Memory		
0309h	1234	1234

Прмер 2:

```
SQURS *AR3, B
```

	Before Instruction	After Instruction
B	00 014B 5DB0	00 0000 0320
T	8765	1234
FRCT	0	0
AR3	0309	0309
Data Memory		
0309h	1234	1234

SRCCD Условное сохранение счетчика повтора блоков

Синтаксис: SRCCD *Xmem, cond*

Операнды: Xmem: Двойной операнд памяти данных

Следующая таблица содержит список условий (cond) для этой инструкции

Условие	Описание	Код условия	Условие	Описание	Код условия
AEQ	(A) = 0	0101	BEQ	(B) = 0	1101
ANEQ	(A) ≠ 0	0100	BNEQ	(B) ≠ 0	1100
AGT	(A) > 0	0110	BGT	(B) > 0	1110
AGEQ	(A) ≥ 0	0010	BGEQ	(B) ≥ 0	1010
ALT	(A) < 0	0011	BLT	(B) < 0	1011
ALEQ	(A) ≤ 0	0111	BLEQ	(B) ≤ 0	1111

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	1	1	0	1	X	X	X	X	C	O	N	D

Выполнение:
Если (условие)
то
(BRC) → Xmem
иначе
(Xmem) → Xmem

Биты состояния: нет

Описание: Если условие выполнено, то инструкция сохраняет содержимое счетчика повтора блоков (BRC) в *Xmem*. Если условие невыполнено, то инструкция считывает содержимое *Xmem* и записывает его обратно в ту же ячейку, таким образом, операнд *Xmem* остается прежним. Независимо от указанного условия операнд *Xmem* всегда считывается и обновляется.

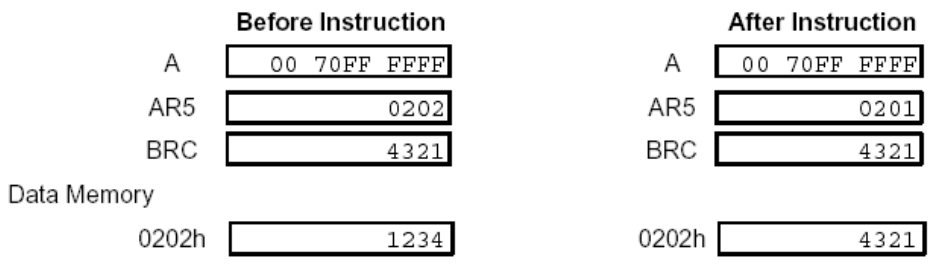
Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 15 (см. страницу 49)

Пример:

```
SRCCD *AR5-, AGT
```



SSBX Установка бита регистра состояния

Синтаксис: SSBX *N*, *SBIT*

Операнды: $0 \leq SBIT \leq 15$
N = 0 или 1

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	N	1	1	0	1	1	S	B	I	T

Выполнение: 1 → STN(SBIT)

Биты состояния: нет

Описание: Эта инструкция устанавливает в 1 указанный бит в регистре состояния 0 или 1. *N* определяет регистр состояния, а *SBIT* определяет бит, который будет установлен. Символьные имена разрядов также распознаются транслятором и могут использоваться в качестве операндов вместо *N* и *SBIT* (см. пример 1).

Примечание – Эта инструкция неповторяемая.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 1 (см. страницу 32)

Пример 1:

```
SSBX SXM  ; SXM означает что N=1, SBIT=8
```

	Before Instruction		After Instruction
ST1	34CD	ST1	35CD

Пример 2:

```
SSBX 1,8
```

	Before Instruction		After Instruction
ST1	34CD	ST1	35CD

ST Сохранение в память T, TRN или непосредственного значения

Синтаксис:
1: ST T, *Smem*
2: ST TRN, *Smem*
3: ST *#lk, Smem*

Операнды:
Smem: Одиночный операнд памяти данных
$-32\ 768 \leq lk \leq 32\ 767$

Опкоды:

1:

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

2:

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

3:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	1	1	0	I	A	A	A	A	A	A	A
16-битная константа															

Выполнение:
1: (T) → Smem
2: (TRN) → Smem
3: lk → Smem

Биты состояния: нет

Описание: Эта инструкция сохраняет содержимое T, регистра перемещений (TRN) или 16-разрядную константу *lk* в ячейку памяти данных *Smem*.

Слов:
Синтаксисы 1 и 2: 1 слово
Синтаксис 3: 2 слова
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов:
Синтаксисы 1 и 2: 1 цикл
Синтаксис 3: 2 цикла
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы:
Синтаксисы 1 и 2: Класс 10A (см. страницу 44)
Синтаксисы 1 и 2: Класс 10B (см. страницу 44)
Синтаксис 3: Класс 12A (см. страницу 46)
Синтаксис 3: Класс 12B (см. страницу 46)

ST Сохранение в память T, TRN или непосредственного значения

Пример 1:

```
ST FFFFh, 0
```

	Before Instruction	After Instruction
DP	004	004
Data Memory		
0200h	0101	FFFF

Пример 2:

```
ST TRN, 5
```

	Before Instruction	After Instruction
DP	004	004
TRN	1234	1234
Data Memory		
0205h	0030	1234

Пример 3:

```
ST T, *AR7-
```

	Before Instruction	After Instruction
T	4210	4210
AR7	0321	0320
Data Memory		
0321h	1200	4210

STH Сохранение старшей части аккумулятора в память

Синтаксис:
1: STH *src, Smem*
2: STH *src,* ASM, *Smem*
3: STH *src, SHFT, Xmem*
4: STH *src*[*, SHIFT*], *Smem*

Операнды:
src: A (аккумулятор A)
B (аккумулятор B)
Smem: Одиночный операнд памяти данных
Xmem: Двойной операнд памяти данных
$0 \le SHFT \le 15$
$-16 \le SHIFT \le 15$

Опкоды

1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	S	I	A	A	A	A	A	A	A

2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	1	1	S	I	A	A	A	A	A	A	A

3:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	1	0	1	S	X	X	X	X	S	H	F	T

4:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	1	1	1	I	A	A	A	A	A	A	A
0	0	0	0	1	1	0	S	0	1	1	S	H	I	F	T

Выполнение:
1: (src) << (–16) → Smem
2: (src) << (ASM – 16) → Smem
3: (src) << (SHFT – 16) → Xmem
4: (src) << (SHIFT – 16) → Smem

Биты состояния: Зависит от SXM

Описание: Эта инструкция сохраняет старшую часть (старшее слово) *src* (биты 31-16) в ячейку памяти данных *Smem*. *Src* сдвигается влево (величина сдвига определяется ASM, SHFT или SHIFT) и биты 31-16 сдвигаемого значения сохраняются в памяти данных (*Smem* или *Xmem*). Если SXM = 0, то бит 39 *src* копируется в старший бит ячейки памяти данных. Если SXM = 1, то знаково-расширенное значение с битом 39 *src* сохраняется в старших разрядах ячейки памяти данных, будучи предварительно сдвинуто вправо на превышение границы разрядов безопасности. *Src* остается неизменным.

STH Сохранение старшей части аккумулятора в память

Примечание – Следующие синтаксисы инструкций при определенных условиях могут быть транслированы как другие синтаксисы: – Синтаксис 3: Если *SHFT* = 0, то опкод инструкции транслируется как синтаксис 1. – Синтаксис 4: Если *SHIFT* = 0, то опкод инструкции транслируется как синтаксис 1. – Синтаксис 4: Если 0 < *SHIFT* ≤ 15 и косвенный модификатор эквивалентен одному из *Xmem* режимов, опкод инструкции транслируется как синтаксис 3.

Слов: Синтаксисы 1, 2 и 3: 1 слово
Синтаксис 4: 2 слова
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: Синтаксисы 1, 2 и 3: 1 цикл
Синтаксис 4: 2 цикла
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Синтаксисы 1, 2 и 3: Класс 10A (см. страницу 44)
Синтаксисы 1 и 2: Класс 10B (см. страницу 44)
Синтаксис 4: Класс 11A (см. страницу 45)
Синтаксис 4: Класс 11B (см. страницу 45)

Пример 1:

```
STH A, 10
```

	Before Instruction	After Instruction
A	FF 8765 4321	FF 8765 4321
DP	004	004
Data Memory		
020Ah	1234	8765

Пример 2:

```
STH B, -8, *AR7-
```

	Before Instruction	After Instruction
B	FF 8421 1234	FF 8421 1234
AR7	0321	0320
Data Memory		
0321h	ABCD	FF84

Пример 3:

```
STH A, -4, 10
```

	Before Instruction	After Instruction
A	FF 8421 1234	FF 8421 1234
SXM	1	1
DP	004	004
Data Memory		
020Ah	7FFF	F842

STL Сохранение младшей части аккумулятора в память

Синтаксис:
1: STL *src*, *Smem*
2: STL *src*, ASM, *Smem*
3: STL *src*, *SHFT*, *Xmem*
4: STL *src*[, *SHIFT*], *Smem*

Операнды:
src: A (аккумулятор A)
B (аккумулятор B)
Smem: Одиночный операнд памяти данных
Xmem: Двойной операнд памяти данных
$0 \le SHFT \le 15$
$-16 \le SHIFT \le 15$

Опкоды

1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	S	I	A	A	A	A	A	A	A

2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	1	0	S	I	A	A	A	A	A	A	A

3:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	1	0	0	S	X	X	X	X	S	H	F	T

4:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	1	1	1	I	A	A	A	A	A	A	A
0	0	0	0	1	1	0	S	1	0	0	S	H	I	F	T

Выполнение:
1: (src) → Smem
2: (src) << ASM → Smem
3: (src) << SHFT → Xmem
4: (src) << SHIFT → Smem

Биты состояния: зависит от SXM

Описание: Эта инструкция сохраняет младшую часть (младшее слово) *src* (биты 15-0) в ячейку памяти данных *Smem*. *Src* сдвигается влево (величина сдвига определяется ASM, SHFT или SHIFT) и биты 15-0 сдвигаемого значения сохраняются в памяти данных (*Smem* или *Xmem*). Когда значение сдвига положительное, в младшие разряды сдвигаются нули.

STL Сохранение младшей части аккумулятора в память

> Примечание – Следующие синтаксисы инструкций при определенных условиях могут быть транслированы как другие синтаксисы:
>
> – Синтаксис 3: Если *SHFT* = 0, то опкод инструкции транслируется как синтаксис 1.
>
> – Синтаксис 4: Если *SHIFT* = 0, то опкод инструкции транслируется как синтаксис 1.
>
> – Синтаксис 4: Если 0 < *SHIFT* ≤ 15 и косвенный модификатор эквивалентен одному из *Xmem* режимов, опкод инструкции транслируется как синтаксис 3.

Слов: Синтаксисы 1, 2 и 3: 1 слово
Синтаксис 4: 2 слова
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: Синтаксисы 1, 2 и 3: 1 цикл
Синтаксис 4: 2 цикла
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Синтаксисы 1, 2 и 3: Класс 10A (см. страницу 44)
Синтаксисы 1 и 2: Класс 10B (см. страницу 44)
Синтаксис 4: Класс 11A (см. страницу 45)
Синтаксис 4: Класс 11B (см. страницу 45)

Пример 1:

```
STL A, 11
```

	Before Instruction	After Instruction
A	FF 8765 4321	FF 8765 4321
DP	004	004
Data Memory		
020Bh	1234	4321

Пример 2:

```
STL B, -8, *AR7-
```

	Before Instruction	After Instruction
B	FF 8421 1234	FF 8421 1234
SXM	0	0
AR7	0321	0320
Data Memory		
0321h	0099	2112

Пример 3:

```
STL A, 7, 11
```

	Before Instruction	After Instruction
A	FF 8421 1234	FF 8421 1234
DP	004	004
Data Memory		
020Bh	0101	1A00

STLM Сохранение младшей части аккумулятора в MMR

Синтаксис: STLM *src, MMR*

Операнды:
src: A (аккумулятор A)
B (аккумулятор B)
MMR: Картированный в памяти регистр

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	S	I	A	A	A	A	A	A	A

Выполнение: (src(15-0)) → MMR

Биты состояния: нет

Описание: Эта инструкция сохраняет младшую часть *src* (биты 15-0) в указанный картированный в памяти регистр, *MMR*. И при прямой и при косвенной адресации, девять старших разрядов действующего адреса принудительно очищаются (заполняются 0) для выбора нулевой страницы памяти данных, независимо от текущего содержимого (DP) или содержимого старших девяти бит ARx. Инструкция позволяет сохранять *src* в ячейки нулевой страницы памяти без изменения поля DP в регистре состояния ST0.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 10A (см. страницу 44)

Пример 1:

```
STLM A, BRC
```

	Before Instruction		After Instruction
A	FF 8765 4321	A	FF 8765 4321
BRC(1Ah)	1234	BRC	4321

Пример 2:

```
STLM B, *AR1-
```

	Before Instruction		After Instruction
B	FF 8421 1234	B	FF 8421 1234
AR1	3F17	AR1	0016
AR7(17h)	0099	AR7	1234

STM Сохранение непостредственного значения в MMR

Синтаксис: STM *#lk, MMR*

Операнды: MMR: регистр, отображаемый в память
–32 768 ≤ lk ≤ 32 767

Опкод:

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

16-битная константа

Выполнение: lk → MMR

Биты состояния: нет

Описание: Эта инструкция сохраняет 16-разрядную константу *lk* в картированный в памяти регистр, *MMR* или в ячейки нулевой страницы памяти без изменения поля DP в регистре состояния ST0. И при прямой и при косвенной адресации, девять старших разрядов действующего адреса принудительно очищаются (заполняются 0) для выбора нулевой страницы памяти данных, независимо от текущего содержимого (DP) или содержимого старших девяти бит ARx.

Слов: 2 слова

Циклов: 2 цикла

Классы: Класс 12A (см. страницу 46)

Пример 1:

```
STM 0FFFFh, IMR
```

	Before Instruction	After Instruction
IMR	FF01	FFFF

Пример 2:

```
STM 8765h, *AR7+
```

	Before Instruction	After Instruction
AR0	0000	8765
AR7	8010	0011

ST||ADD Сохранение аккумулятора с параллельным сложением

Синтаксис:
ST *src, Ymem*
|| ADD *Xmem, dst*

Операнды:
src, dst: A (аккумулятор A)
B (аккумулятор B)
Xmem, Ymem: Двойные операнды памяти данных
dst_: Если *dst* = A, то *dst_* = B; если *dst* = B, то *dst_* = A

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	S	D	X	X	X	X	Y	Y	Y	Y

Выполнение:
(src) << (ASM – 16) → Ymem
(dst_) + (Xmem) << 16 → dst

Биты состояния:
Зависит от OVM, SXM и ASM
Оказывает действие на C и OVdst

Описание:
Эта инструкция сохраняет *src*, сдвинутый на величину (ASM – 16) в ячейку памяти данных *Ymem*. Параллеллельно содержимое dst_ прибавляется к операнду памяти данных *Xmem*, сдвинутому на 16 разрядов влево и результат сохраняется в *dst*. Если в качестве *src* и *dst*, указан один и тот же аккумулятор, то в *Ymem* сохраняется содержимое *src* до выполнения инструкции.

Слов:
1 слово

Циклов:
1 цикл

Классы:
Класс 14 (см. страницу 48)

Пример:

```
ST A, *AR3
||ADD *AR5+0%, B
```

	Before Instruction	After Instruction
A	FF 8421 1000	FF 8021 1000
B	00 0000 1111	FF 0422 1000
OVM	0	0
SXM	1	1
ASM	1	1
AR0	0002	0002
AR3	0200	0200
AR5	0300	0302
Data Memory		
0200h	0101	0842
0300h	8001	8001

ST||LD Сохранение аккумулятора с параллельной загрузкой

Синтаксис:
1: ST *src, Ymem*
|| LD *Xmem, dst*
2: ST *src, Ymem*
|| LD *Xmem*, T

Операнды:
src, dst: A (аккумулятор A)
B (аккумулятор B)
Xmem, Ymem: Двойные операнды памяти данных

Опкоды

1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	1	0	S	D	X	X	X	X	Y	Y	Y	Y

2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	0	1	S	0	X	X	X	X	Y	Y	Y	Y

Выполнение:
1. (src) << (ASM – 16) → Ymem
(Xmem) << 16 → dst
2. (src) << (ASM – 16) → Ymem
(Xmem) → T

Биты состояния:
зависит от OVM и ASM
Оказывает действие на C

Описание:
Эта инструкция сохраняет *src*, сдвинутый на величину (ASM – 16) в ячейку памяти данных *Ymem*. Паралеллельно эта инструкция загружает 16-разрядный двойной операнд памяти данных *Xmem* в *dst* или в T. Если в качестве *src* и *dst*, указан один и тот же аккумулятор, то в *Ymem* сохраняется содержимое *src* до выполнения инструкции.

Слов:
1 слово

Циклов:
1 цикл

Классы:
Класс 14 (см. страницу 48)

ST||LD Сохранение аккумулятора с параллельной загрузкой

Пример 1:

```
ST B, *AR2-
||LD *AR4+, A
```

	Before Instruction	After Instruction
A	00 0000 001C	FF 8001 0000
B	FF 8421 1234	FF 8421 1234
SXM	1	1
ASM	1C	1C
AR2	01FF	01FE
AR4	0200	0201
Data Memory		
01FFh	xxxx	F842
0200h	8001	8001

Пример 2:

```
ST A, *AR3
||LD *AR4, T
```

	Before Instruction	After Instruction
A	FF 8421 1234	FF 8421 1234
T	3456	80FF
ASM	1	1
AR3	0200	0200
AR4	0100	0100
Data Memory		
0200h	0001	0842
0100h	80FF	80FF

Пример 3:

```
ST A, *AR2+
||LD *AR2-, A
```

В примере 3 LD считывает исходный операнд из ячейки памяти, указываемой AR2 прежде, чем инструкция ST произведет запись в эту ячейку. ST в свою очередь считывает операнд из аккумулятора A перед тем, как LD загрузит значение в аккумулятор A.

ST||MAC[R] Сохранение аккумулятора параллельно с умножением-накоплением с/без округления

Синтаксис: ST *src, Ymem*
|| MAC[R] *Xmem, dst*

Операнды: src, dst: A (аккумулятор A)
B (аккумулятор B)
Xmem, Ymem: Двойные операнды памяти данных

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	0	R	S	D	X	X	X	X	Y	Y	Y	Y

Выполнение: (src << (ASM – 16)) → Ymem
Если (округление)
То
Round ((Xmem) × (T) + (dst)) → dst
Иначе
(Xmem) × (T) + (dst) → dst

Биты состояния: зависит от OVM, ASM, SXM и FRCT
Оказывает действие на C и OVdst

Описание: Эта инструкция сохраняет *src*, сдвинутый на величину (ASM – 16) в ячейку памяти данных *Ymem*. Параллеллельно содержимое T умножается на операнд памяти данных *Xmem*, это произведение складывается с содержимым *dst* (с округлением или без округления), и полученный результат сохраняется в *dst*. Если в качестве *src* и *dst*, указан один и тот же аккумулятор, то в *Ymem* сохраняется содержимое *src* до выполнения инструкции.
Если в инструкции указан суффикс R, то результат умножения и сложения округляется, путем прибавления числа 2^{15} и очистки младших битов (15-0) в 0.

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 14 (см. страницу 48)

ST||MAC[R] Сохранение аккумулятора параллельно с умножением-накоплением с/без округления

Пример 1:

```
ST A, *AR4-
||MAC *AR5, B
```

	Before Instruction	After Instruction
A	00 0011 1111	00 0011 1111
B	00 0000 1111	00 010C 9511
T	0400	0400
ASM	5	5
FRCT	0	0
AR4	0100	00FF
AR5	0200	0200
Data Memory		
100h	1234	0222
200h	4321	4321

Пример 2:

```
ST A, *AR4+
||MACR *AR5+, B
```

	Before Instruction	After Instruction
A	00 0011 1111	00 0011 1111
B	00 0000 1111	00 010D 0000
T	0400	0400
ASM	1C	1C
FRCT	0	0
AR4	0100	0101
AR5	0200	0201
Data Memory		
100h	1234	0001
200h	4321	4321

ST||MAS[R] Сохранение аккумулятора параллельно с умножением-вычитанием с/без округления

Синтаксис:
ST *src, Ymem*
|| MAS[R] *Xmem, dst*

Операнды:
src, dst: A (аккумулятор A)
B (аккумулятор B)
Xmem, Ymem: Двойные операнды памяти данных

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	1	R	S	D	X	X	X	X	Y	Y	Y	Y

Выполнение:
(src << (ASM – 16)) → Ymem
Если (округление)
То
Round ((dst) – (Xmem) × (T)) → dst
Иначе
(dst) – (Xmem) × (T) → dst

Биты состояния:
зависит от OVM, ASM, SXM и FRCT
Оказывает действие на C и OVdst

Описание:
Эта инструкция сохраняет *src*, сдвинутый на величину (ASM – 16) в ячейку памяти данных *Ymem*. Параллеллельно содержимое T умножается на операнд памяти данных *Xmem*, это произведение вычитается из содержимого *dst* (с округлением или без округления), и полученный результат сохраняется в *dst*. Если в качестве *src* и *dst*, указан один и тот же аккумулятор, то в *Ymem* сохраняется содержимое *src* до выполнения инструкции.

Если в инструкции указан суффикс R, то результат умножения и сложения округляется, путем прибавления числа 2^{15} и очистки младших битов (15-0) в 0.

Слов:
1 слово

Циклов:
1 цикл

Классы:
Класс 14 (см. страницу 48)

ST||MAS[R] Сохранение аккумулятора параллельно с умножением-вычитанием с/без округления

Пример 1:

```
ST A, *AR4+
||MAS *AR5, B
```

	Before Instruction	After Instruction
A	00 0011 1111	00 0011 1111
B	00 0000 1111	FF FEF3 8D11
T	0400	0400
ASM	5	5
FRCT	0	0
AR4	0100	0101
AR5	0200	0200
Data Memory		
0100h	1234	0222
0200h	4321	4321

Пример 2:

```
ST A, *AR4+
||MASR *AR5+, B
```

	Before Instruction	After Instruction
A	00 0011 1111	00 0011 1111
B	00 0000 1111	FF FEF4 0000
T	0400	0400
ASM	1	1
FRCT	0	0
AR4	0100	0101
AR5	0200	0201
Data Memory		
0100h	1234	0022
0200h	4321	4321

ST||MPY Сохранение аккумулятора с параллельным умножением

Синтаксис:
ST *src, Ymem*
|| MPY *Xmem, dst*

Операнды:
src, dst: A (аккумулятор A)
B (аккумулятор B)
Xmem, Ymem: Двойные операнды памяти данных

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	1	1	S	D	X	X	X	X	Y	Y	Y	Y

Выполнение:
(src << (ASM – 16)) → Ymem
(T) × (Xmem) → dst

Биты состояния:
зависит от OVM, ASM, SXM и FRCT
Оказывает действие на C и OVdst

Описание:
Эта инструкция сохраняет *src*, сдвинутый на величину (ASM – 16) в ячейку памяти данных *Ymem*. Параллельно содержимое T умножается на операнд памяти данных *Xmem* и произведение сохраняется в *dst*. Если в качестве *src* и *dst*, указан один и тот же аккумулятор, то в *Ymem* сохраняется содержимое *src* до выполнения инструкции.

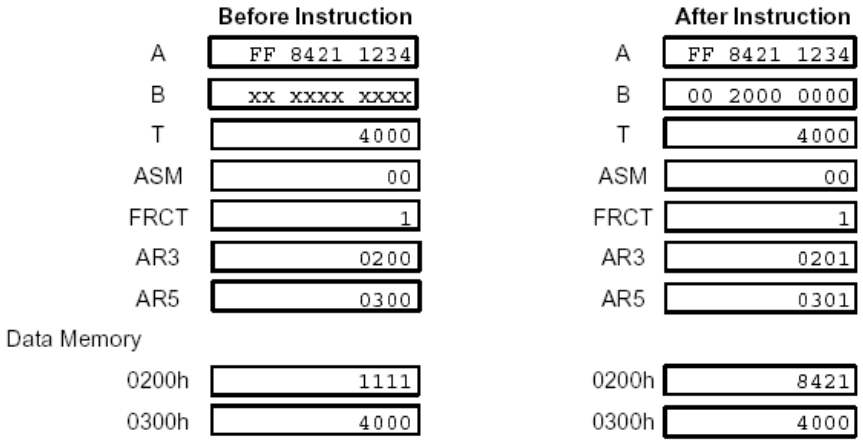
Слов:
1 слово

Циклов:
1 цикл

Классы:
Класс 14 (см. страницу 48)

Пример:

```
ST A, *AR3+
||MPY *AR5+, B
```



ST||SUB Сохранение аккумулятора с параллельным вычитанием

Синтаксис:
ST *src, Ymem*
|| SUB *Xmem, dst*

Операнды:

src, dst:	A (аккумулятор A) B (аккумулятор B)
Xmem, Ymem:	Двойные операнды памяти данных
dst_:	Если *dst* = A, то *dst_* = B; если *dst* = B, то *dst_* = A

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	1	S	D	X	X	X	X	Y	Y	Y	Y

Выполнение:
(src << (ASM – 16)) → Ymem
(Xmem) << 16 – (dst_) → dst

Биты состояния:
зависит от OVM, ASM, SXM
Оказывает действие на C и OVdst

Описание:
Эта инструкция сохраняет *src*, сдвинутый на величину (ASM – 16) в ячейку памяти данных *Ymem*. Паралеллельно содержимое dst_ вычитается из 16-разрядного двойного операнда памяти *Xmem*, сдвинутого на 16 разрядов влево и результат сохраняется в *dst*. Если в качестве *src* и *dst*, указан один и тот же аккумулятор, то в *Ymem* сохраняется содержимое *src* до выполнения инструкции.

Слов:
1 слово

Циклов:
1 цикл

Классы:
Класс 14 (см. страницу 48)

Пример:

```
ST A, *AR3-
||SUB *AR5+0%, B
```

	Before Instruction	After Instruction
A	FF 8421 0000	FF 8421 0000
B	00 1000 0001	FF FBE0 0000
ASM	01	01
SXM	1	1
AR0	0002	0002
AR3	01FF	01FE
AR5	0300	0302
Data Memory		
01FFh	1111	0842
0300h	8001	8001

STRCD Условное сохранение T

Синтаксис: STRCD *Xmem, cond*

Операнды: Xmem: Двойной операнд памяти данных

Следующая таблица содержит список условий (cond) для этой инструкции

Условие	Описание	Код условия	Условие	Описание	Код условия
AEQ	(A) = 0	0101	BEQ	(B) = 0	1101
ANEQ	(A) ≠ 0	0100	BNEQ	(B) ≠ 0	1100
AGT	(A) > 0	0110	BGT	(B) > 0	1110
AGEQ	(A) ≥ 0	0010	BGEQ	(B) ≥ 0	1010
ALT	(A) < 0	0011	BLT	(B) < 0	1011
ALEQ	(A) ≤ 0	0111	BLEQ	(B) ≤ 0	1111

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	1	1	0	0	X	X	X	X	C	O	N	D

Выполнение:
Если (условие)
то
(T) → Xmem
иначе
(Xmem) → Xmem

Биты состояния: нет

Описание: Если условие выполнено, то инструкция сохраняет содержимое T в ячейку памяти данных *Xmem*. Если условие невыполнено, то инструкция считывает содержимое *Xmem* и записывает его обратно в ту же ячейку, таким образом, операнд *Xmem* остается прежним. Независимо от указанного условия операнд *Xmem* всегда считывается и обновляется.

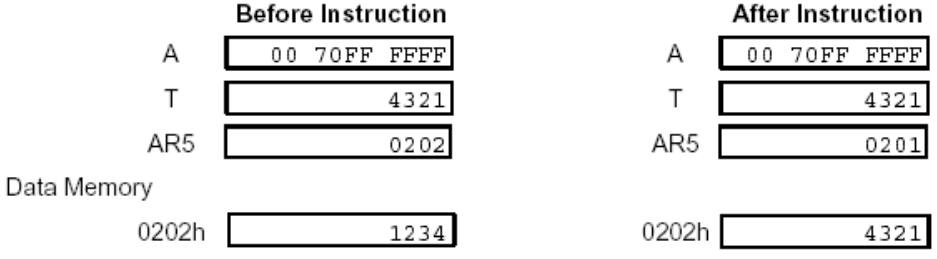
Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 15 (см. страницу 49)

Пример:

```
STRCD *AR5-, AGT
```



SUB Вычитание из аккумулятора

Синтаксис:

1: SUB *Smem, src*
2: SUB *Smem,* TS, *src*
3: SUB *Smem,* 16, *src*[, *dst*]
4: SUB *Smem*[, *SHIFT*], *src*[, *dst*]
5: SUB *Xmem, SHFT, src*
6: SUB *Xmem, Ymem, dst*
7: SUB *#lk*[, *SHFT*], *src*[, *dst*]
8: SUB *#lk,* 16, *src*[, *dst*]
9: SUB *src*[, *SHIFT*], [, *dst*]
10: SUB *src,* ASM [, *dst*]

Операнды:

Smem:	Одиночный операнд памяти данных
Xmem, Ymem	Двойные операнды памяти данных
src, dst:	A (аккумулятор A)
	B (аккумулятор B)

$-32\,768 \le lk \le 32\,767$
$0 \le SHFT \le 15$
$-16 \le SHIFT \le 15$

Опкод:

1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	S	I	A	A	A	A	A	A	A

2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	S	I	A	A	A	A	A	A	A

3:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	S	D	I	A	A	A	A	A	A	A

4:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	1	1	1	I	A	A	A	A	A	A	A
0	0	0	0	1	1	S	D	0	0	1	S	H	I	F	T

5:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	0	0	1	S	X	X	X	X	S	H	F	T

6:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	0	0	1	D	X	X	X	X	Y	Y	Y	Y

7:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	0	0	0	1	S	H	F	T
16-битная константа															

8:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	0	1	1	0	0	0	0	1
16-битная константа															

SUB Вычитание из аккумулятора

9:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	S	D	0	0	1	S	H	I	F	T

10:

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

Выполнение:

1: (src) – (Smem) → src
2: (src) – (Smem) << (TS) → src
3: (src) – (Smem) << 16 → dst
4: (src) – (Smem) << SHIFT → dst
5: (src) – (Xmem) << SHFT → src
6: (Xmem) << 16 – Ymem << 16 → dst
7: (src) – lk << SHFT → dst
8: (src) – lk << 16 → dst
9: (dst) – (src) << SHIFT → dst
10: (dst) – (src) << ASM → dst

Биты состояния:

Зависят от SXM и OVM
Оказывают действие на C и OVdst (или OVsrc, если dst = src)

Для синтаксиса 3, если в результате действия происходит генерация переноса, то бит переноса C устанавливается в 1; в других случаях бит C не затрагивается.

Описание:

Эта инструкция вычитает 16-разрядное значение из содержимого выбранного аккумулятора или 16-разрядного операнда *Xmem* при использовании адресации двойным операндом памяти данных. В качестве 16-разрядного вычитаемого значения могут выступать:

- Содержимое одиночного операнда памяти данных(*Smem*).
- Содержимое двойного операнда памяти данных (*Ymem*).
- 16-разрядный непосредственный операнд (*#lk*).
- Сдвинутое значение *src*.

Если *dst* определен, инструкция сохраняет результат в *dst*. Если *dst* не определен, то результат будет сохранен в *src*. К большинству вторых операндов может быть применен сдвиг.
При сдвиге влево:

- Младшие биты очищаются.
- Старшие биты:
  - Расширяются значением знакового бита, при SXM = 1.
  - Очищаются, при SXM = 0.

При сдвиге вправо, старшие биты:

- Расширяются значением знакового бита, при SXM = 1.
- Очищаются, при SXM = 0.

SUB Вычитание из аккумулятора

Примечание – Следующие синтаксисы инструкций при определенных условиях могут быть транслированы как другие синтаксисы:

– Синтаксис 4: Если *dst* = *src* и *SHIFT* = 0, опкод инструкции транслируется как синтаксис 1.

– Синтаксис 4: Если *dst* = *src* и *SHIFT* ≤ 15 и режимом косвенной адресации *Smem* включен в *Xmem*, опкод инструкции транслируется как синтаксис 1.

Слов: Синтаксисы 1, 2, 3, 5, 6, 9 и 10: 1 слово
Синтаксисы 4, 7 и 8: 2 слова
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: Синтаксисы 1, 2, 3, 5, 6, 9 и 10: 1 цикл
Синтаксисы 4, 7 и 8: 2 цикла
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Синтаксисы 1, 2, 3 и 5: Класс 3A (см. страницу 34)
Синтаксисы 1, 2 и 3: Класс 3B (см. страницу 35)
Синтаксис 4: Класс 4A (см. страницу 36)
Синтаксис 4: Класс 4B (см. страницу 37)
Синтаксис 6: Класс 7 (см. страницу 41)
Синтаксисы 7 и 8: Класс 2 (см. страницу 33)
Синтаксисы 9 и 10: Класс 1 (см. страницу 32)

SUB Вычитание из аккумулятора

Пример 1:

```
SUB *AR1+, 14, A
```

	Before Instruction	After Instruction
A	00 0000 1200	FF FAC0 1200
C	x	0
SXM	1	1
AR1	0100	0101
Data Memory		
0100h	1500	1500

Пример 2:

```
SUB A, -8, B
```

	Before Instruction	After Instruction
A	00 0000 1200	00 0000 1200
B	00 0000 1800	00 0000 17EE
C	x	1
SXM	1	1

Пример 3:

```
SUB #12345, 8, A, B
```

	Before Instruction	After Instruction
A	00 0000 1200	00 0000 1200
B	00 0000 1800	FF FFCF D900
C	x	0
SXM	1	1

SUBB Вычитание из аккумулятора с заемом

Синтаксис: SUBB *Smem, src*

Операнды:
Smem: Одиночный операнд памяти данных
src: A (аккумулятор A)
B (аккумулятор B)

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	D	I	A	A	A	A	A	A	A

Выполнение: (src) – (Smem) – (логически инверсный C) → src

Биты состояния: Зависит от OVM, C
Оказывает влияние на C и OVsrc

Описание: Эта инструкция вычитает содержимое 16-разрядного одиночного операнда памяти данных *Smem* и логически инверсный бит переноса C из *src* без знакового расширения.

Слов: 1 слово

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 1 цикл

При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 3A (см. страницу 34)
Класс 3B (см. страницу 35)

Пример 1:

```
SUBB 5, A
```

	Before Instruction	After Instruction
A	00 0000 0006	FF FFFF FFFF
C	0	0
DP	008	008
Data Memory		
0405h	0006	0006

Пример 2:

```
SUBB *AR1+, B
```

	Before Instruction	After Instruction
B	FF 8000 0006	FF 8000 0000
C	1	1
OVM	1	1
AR1	0405	0406
Data Memory		
0405h	0006	0006

SUBC Условное вычитание

Синтаксис: SUBC *Smem, src*

Операнды:
src: A (аккумулятор A)
B (аккумулятор B)
Smem: Одиночный операнд памяти данных

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	1	1	S	I	A	A	A	A	A	A	A

Выполнение:
(src) – ((Smem) << 15) → выход АЛУ
Если выход АЛУ ≥ 0
То
((выход АЛУ) << 1) + 1 → src
Иначе (src) << 1 → src

Биты состояния:
Зависит от SXM
Оказывает влияние на C и OVsrc

Описание:
Эта инструкция вычитает 16-разрядный одиночный операнд памяти данных *Smem*, сдвинутый влево на 15 разрядов, из содержимого *src*. Если полученный результат больше нуля, то он сдвигается влево на 1 разряд, к полученному значению прибавляется 1 и результат сложения сохраняется в *src*. В противном случае содержимое *src* сдвигается влево на 1 разряд и сохраняется в *src*.
Предполагается, что делимое и делитель в этой инструкции положительны. Бит SXM влияет на выполнение инструкции следующим образом:
– Если SXM = 1, то в старшем бите делителя должен быть 0.
– Если SXM = 0, то любой 16-разрядный делитель обеспечивает ожидаемый результат.
Делимое, которое находится в *src*, первоначально должно быть положительным (бит 31 равен 0) и оставаться положительным после сдвига аккумулятора, который происходит в первой части выполнения инструкции.
Эта инструкция оказывает действие на OVA или OVB (в зависимости от *src*), но не зависит от OVM; поэтому при выполнении инструкции, не происходит положительного или отрицательного насыщения *src*.

Слов:
1 слово
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов:
1 цикл
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы:
Класс 3A (см. страницу 34)
Класс 3B (см. страницу 35)

SUBC Условное вычитание

Пример 1:

```
SUBC 2, A
```

	Before Instruction		After Instruction
A	00 0000 0004	A	00 0000 0008
C	x	C	0
DP	006	DP	006
Data Memory			
0302h	0001	0302h	0001

Пример 2:

```
RPT #15
SUBC *AR1, B
```

	Before Instruction		After Instruction
B	00 0000 0041	B	00 0002 0009
C	x	C	1
AR1	1000	AR1	1000
Data Memory			
1000h	0007	1000h	0007

SUBS Вычитание из аккумулятора в беззнаковом режиме

Синтаксис: SUBS *Smem, src*

Операнды:
Smem: Одиночный операнд памяти данных
src: A (аккумулятор A)
B (аккумулятор B)

Опкод:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	S	I	A	A	A	A	A	A	A

Выполнение: (src) – беззнаковый(Smem) → src

Биты состояния: Зависит от OVM
Оказывает влияние на C и OVsrc

Описание: Эта инструкция вычитает содержимое 16-разрядного одиночного операнда памяти данных *Smem* из содержимого *src*. *Smem* считается 16-разрядным беззнаковым числом независимо от значения *SXM*. Результат сохраняется в *src*.

Слов: 1 слово
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 1 цикл
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 3A (см. страницу 34)
Класс 3B (см. страницу 35)

Пример:

```
SUBS *AR2-, B
```

	Before Instruction	After Instruction
B	00 0000 0002	FF FFFF 0FFC
C	x	0
AR2	0100	00FF
Data Memory		
0100h	F006	F006

TRAP Программное прерывание

Синтаксис: TRAP *K*

Операнды: $0 \le K \le 31$

Опкод:

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

Выполнение:
(SP) – 1 → SP
(PC) + 1 → TOS
Вектор прерывания, определяемый K → PC

Биты состояния: нет

Описание: Эта инструкция передает управление программой вектору прерывания, определяемому *K*. Она позволяет использовать программное обеспечение для выполнения прерывания. Список прерываний и соответствующие им значения K приведены в руководство процессора 1867ВЦ4Т КФДЛ.431299.010. Инструкция помещает PC + 1 в ячейку памяти данных, адресуемую SP. Это дает возможность инструкции возврата отыскать в ячейке памяти данных, адресуемой SP указатель на инструкцию, следующую за прерыванием. Эта инструкция немаскируемая; не зависит от INTM и не оказывает влияния на INTM.

Примечание – Эта инструкция неповторяемая.

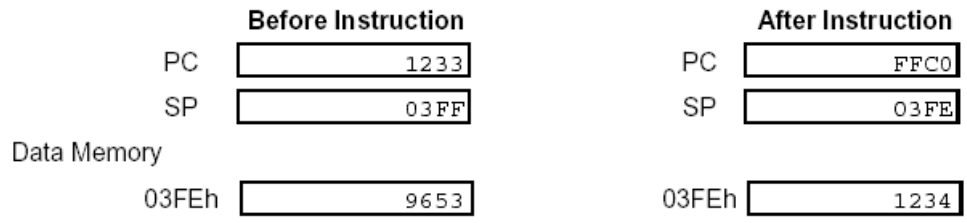
Слов: 1 слово

Циклов: 3 цикла

Классы: Класс 35 (см. страницу 79)

Пример:

```
TRAP 10h
```



WRITA Запись данных в память программ, адресуемую аккумулятором A

Синтаксис: WRITA *Smem*

Операнды: Smem: Одиночный операнд памяти данных

Опкод:

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

Выполнение:
A → PAR
Если ((RC) ≠ 0)
(Smem) → (Pmem по адресу из PAR)
(PAR) + 1 → PAR
(RC) – 1 → RC
Иначе
(Smem) → (Pmem по адресу из PAR)

Биты состояния: нет

Описание: Эта инструкция перемещает слово из ячейки памяти данных, определяемой *Smem* в ячейку памяти программ. Адрес приемника определяется содержимым младшей части аккумулятора A (биты 15-0).
Эту инструкцию можно использовать в режиме повтора для перемещения массивов из памяти данных, адресуемой (в формате косвенной адресации) *Smem*, в непрерывную область памяти программ, адресуемую PAR, с автоматическим увеличением PAR на 1. Начальное значение определяется 16-разрядной младшей частью аккумулятора A. Источник и приемник не должны находиться полностью на кристалле или полностью вне его. Если используется повтор, то после загрузки в конвейер (первое выполнение), инструкция становится эффективно одноцикловой.
Содержимое аккумулятора A не изменяется после выполнения инструкции.

Слов: 1 слово
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 5 циклов
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Класс 26A (см. страницу 70)
Класс 26B (см. страницу 71)

Пример:

```
WRITA 5
```

	Before Instruction		After Instruction
A	00 0000 0257	A	00 0000 0257
DP	032	DP	032
Program Memory			
0257h	0306	0257h	4339
Data Memory			
1005h	4339	1005h	4339

C Условное выполнение

Синтаксис: XC *n*, *cond*[, *cond*[, *cond*]]

Операнды: n = 1 или 2
Следующая таблица содержит список условий (cond) для этой инструкции

Условие	Описание	Код условия	Условие	Описание	Код условия
BIO	BIO# low	0000 0011	NBIO	BIO# high	0000 0010
C	C = 1	0000 1100	NC	C = 0	0000 1000
TC	TC = 1	0011 0000	NTC	TC = 0	0010 0000
AEQ	(A) = 0	0100 0101	BEQ	(B) = 0	0100 1101
ANEQ	(A) ≠ 0	0100 0100	BNEQ	(B) ≠ 0	0100 1100
AGT	(A) > 0	0100 0110	BGT	(B) > 0	0100 1110
AGEQ	(A) ≥ 0	0100 0010	BGEQ	(B) ≥ 0	0100 1010
ALT	(A) < 0	0100 0011	BLT	(B) < 0	0100 1011
ALEQ	(A) ≤ 0	0100 0111	BLEQ	(B) ≤ 0	0100 1111
AOV	A переполнен	0111 0000	BOV	B переполнен	0111 1000
ANOV	A не переп-н	0110 0000	BNOV	B не переп-н	0110 1000
UNC	безусловный	0000 0000			

Опкод:

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

Синтаксис n	Опкод N
1	0
2	1

Выполнение: Если (условие)
то
выполняются следующие n инструкций
иначе
выполняется инструкция NOP
вместо следующих n инструкций

Биты состояния: нет

Описание: Выполнение этой инструкции зависит от значения n и указанных условий:

– Если n = 1 и условие(я) выполнено(ы), то исполняется однословная инструкция, следующая за инструкцией XC.

– Если n = 2 и условие(я) выполнено(ы), то исполняется одна двухсловная инструкция или две однословных, следующих за инструкцией XC.

– Если условие(я) невыполнено(ы), то в зависимости от n исполняются одна или две инструкции NOP.

Эта инструкция позволяет осуществлять перед выполнением проверку нескольких условий, как индивидуально, так и в комбинации с другими условиями. Можно компоновать условия только из одной группы следующим образом:

Группа 1: Допускается выбрать до двух условий. Каждое условие должно принадлежать разным категориям (A или B, см. таблицу ниже). Не допускается использование в инструкции двух условий из одной категории. Например, одновременно можно тестировать EQ и OV, однако, нельзя выполнить одновременную проверку GT и NEQ. Для обоих условий аккумулятор должен быть одним и тем же. Не допускается проверка двух условий для разных аккумуляторов в одной инструкции. Например, разрешена одновременная проверка AGT и AOV, однако, не допускается использование в одной инструкции условий AGT и BOV.

Группа 2: Допускается выбрать до трех условий. Каждое из трех условий должно быть из различных категорий (A, B или C). Не допускается использование двух или более условий из одной категории, т. е. допускается одновременная проверка TC, C и BIO, однако, не допускается одновременная проверка NTC, C и NC.

Сочетаемость условий для этой инструкции

Группа 1		Группа 2		
Категория A	Категория B	Категория A	Категория B	Категория C
EQ	OV	TC	C	BIO
NEQ	NOV	NTC	NC	NBIO
LT				
LEQ				
GT				
GEQ				

При выполнении этой инструкции и двух, следующих за ней слов, прерывания подавляются.

Примечание – Условия проверяются за два полных цикла до выполнения инструкции. Поэтому, если две однословные или одна двухсловная инструкция модифицируют условия, это не оказывает влияния на выполнение инструкции. Однако, если условия изменяются в течение двух тактов, инструкции прерываний, использующие эту инструкцию могут привести к нежелательным результатам.

Эта инструкция неповторяемая.

XC Условное выполнение

Слов: 1 слово

Циклов: 1 цикл

Классы: Класс 1 (см. страницу 32)

Пример:

```
XC 1, ALEQ
MAR *AR1+
ADD A, DAT100
```

Если содержимое аккумулятора A меньше или равно нулю, то вспомогательный регистр AR1 изменяется перед выполнением инструкции ADD.

XOR Исключающее ИЛИ с аккумулятором

Синтаксис:
1: XOR *Smem, src*
2: XOR *#lk*[, *SHFT*], *src*[, *dst*]
3: XOR *#lk*, 16, *src*[, *dst*]
4: XOR *src*[, *SHIFT*], [, *dst*]

Операнды:
src, dst: A (аккумулятор A)
B (аккумулятор B)
Smem: Одиночный операнд памяти данных
$0 \leq lk \leq 65\,535$
$-16 \leq SHIFT \leq 15$
$0 \leq SHFT \leq 15$

Опкоды:

1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	1	0	S	I	A	A	A	A	A	A	A

2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	0	1	0	1	S	H	F	T
16-битная константа															

3:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	0	1	1	0	0	1	0	1
16-битная константа															

4:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	1	1	0	S	H	I	F	T

Выполнение:
1: (Smem) XOR (src) → src
2: lk << SHFT XOR (src) → dst
3: lk << 16 XOR (src) → dst
4: (src) << SHIFT XOR (dst)→ dst

Биты состояния: нет

Описание: Эта инструкция выполняет операцию "Исключающее ИЛИ" между 16-разрядным одиночным операндом памяти данных *Smem* (сдвинутым как указано в синтаксисе инструкции) и содержимым выбранного аккумулятора и сохраняет результат в *dst* или в *src*, как указано. При сдвиге влево младшие биты очищаются, старшие – не расширяются на знак. При сдвиге вправо знак не расширяется.

XOR Исключающее ИЛИ с аккумулятором

Слов: Синтаксисы 1 и 4: 1 слово
Синтаксисы 2 и 3: 2 слова
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: Синтаксисы 1 и 4: 1 цикл
Синтаксисы 2 и 3: 2 цикла
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Синтаксис 1: Класс 3A (см. страницу 34)
Синтаксис 1: Класс 3B (см. страницу 35)
Синтаксисы 2 и 3: Класс 2 (см. страницу 33)
Синтаксис 4: Класс 1 (см. страницу 32)

Пример 1:

```
XOR *AR3+, A
```

	Before Instruction		After Instruction
A	00 00FF 1200	A	00 00FF 0700
AR3	0100	AR3	0101
Data Memory			
0100h	1500	0100h	1500

Пример 2:

```
XOR A, +3, B
```

	Before Instruction		After Instruction
A	00 0000 1200	A	00 0000 1200
B	00 0000 1800	B	00 0000 8800

XORM "Исключающее ИЛИ" ячейки памяти с длинным непосредственным значением

Синтаксис: XORM #*lk, Smem*

Операнды: Smem: Одиночный операнд памяти данных
$0 \le lk \le 65\,535$

Опкод:

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

Выполнение: lk XOR (Smem) → Smem

Биты состояния: нет

Описание: Эта инструкция выполняет логическую операцию "Исключающее ИЛИ" между содержимым ячейки памяти данных *Smem* и 16-разрядной константой *lk*. Результат записывается в *Smem*.

> Примечание – Эта инструкция неповторяемая.

Слов: 2 слова
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 слово.

Циклов: 2 цикла
При использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem добавляется еще 1 цикл.

Классы: Синтаксис 18A: Класс 3A (см. страницу 34)
Синтаксис 18B: Класс 3B (см. страницу 34)

Пример:

```
XORM 0404h, *AR4-
```

	Before Instruction	After Instruction
AR4	0100	00FF
Data Memory		
0100h	4444	4040

Приложение А

(обязательное)

Коды условий

Это приложение содержит список условий, применяемых в условных инструкциях (таблица А.1) и сочетаемости условий (таблица А.2). Условные инструкции могут осуществлять проверку условия, как индивидуально, так и в комбинации с другими условиями. Можно компоновать условия только из одной группы следующим образом:

Группа 1: Допускается выбрать до двух условий. Каждое условие должно принадлежать разным категориям (A или B, см. таблицу А.2). Не допускается использование в инструкции двух условий из одной категории. Например, одновременно можно тестировать EQ и OV, однако, нельзя выполнить одновременную проверку GT и NEQ. Для обоих условий аккумулятор должен быть одним и тем же. Не допускается проверка двух условий для разных аккумуляторов в одной инструкции. Например, разрешена одновременная проверка AGT и AOV, однако, не допускается использование в одной инструкции условий AGT и BOV.

Группа 2: Допускается выбрать до трех условий. Каждое из трех условий должно быть из различных категорий (A, B или C). Не допускается использование двух или более условий из одной категории, т. е. допускается одновременная проверка TC, C и BIO, однако, не допускается одновременная проверка NTC, C и NC.

Таблица А.1 – Условия для условных инструкций

Операнд	Условие	Описание
AEQ	A = 0	Аккумулятор A равен 0
BEQ	B = 0	Аккумулятор B равен 0
ANEQ	A ≠ 0	Аккумулятор A не равен 0
BNEQ	B ≠ 0	Аккумулятор B не равен 0
ALT	A < 0	Аккумулятор A меньше 0
BLT	B < 0	Аккумулятор B меньше 0
ALEQ	A ≤ 0	Аккумулятор A меньше или равен 0
BLEQ	B ≤ 0	Аккумулятор B меньше или равен 0
AGT	A > 0	Аккумулятор A больше 0
BGT	B > 0	Аккумулятор B больше 0
AGEQ	A ≥ 0	Аккумулятор A больше или равен 0
BGEQ	B ≥ 0	Аккумулятор B больше или равен 0
AOV[1)]	AOV = 1	Переполнение в аккумуляторе A
BOV[1)]	BOV = 1	Переполнение в аккумуляторе B
ANOV[1)]	AOV = 0	Нет переполнения в аккумуляторе A
BNOV[1)]	BOV = 0	Нет переполнения в аккумуляторе B
C[1)]	C = 1	Перенос АЛУ установлен в 1
NC[1)]	C = 0	Перенос АЛУ сброшен в 0
TC[1)]	TC = 1	Тестовый флаг установлен в 1
NTC[1)]	TC = 0	Тестовый флаг сброшен в 0
BIO[1)]	BIO низкий	Сигнал BIO низкий
NBIO[1)]	BIO высокий	Сигнал BIO высокий
UNC[1)]	нет	Безусловная операция

[1)] Не могут быть использованы в инструкциях условного сохранения

Таблица А.2 – Сочетаемость условий

Группа 1		Группа 2		
Категория A	Категория B	Категория A	Категория B	Категория C
EQ	OV	TC	C	BIO
NEQ	NOV	NTC	NC	NBIO
LT				
LEQ				
GT				
GEQ				

Приложение Б

(обязательное)

Регистры состояния и управления процессором

Ниже приведено описание битов и полей регистров состояния и управления процессора 1867ВЦ4Т. Процессор имеет три регистра состояния и управления:

- Регистр состояния 0 (ST0).
- Регистр состояния 1 (ST1).
- Регистр состояния режима процессора (PMST).

В таблице Б.1 приведены названия разрядов и полей, используемые в описании формата регистров.

Таблица Б.1

Поле/разряд	Описание
ARP	Указатель вспомогательного регистра
ASM	Режим сдвига аккумулятора
AVIS	Режим видимости адреса
BRAF	Флаг активности повтора блока
C	Перенос
CLKOFF	CLOCKOUT выключен
CMPT	Режим совместимости
CPL	Режим копиляции
C16	Двойной 16-разрядный/двойной точности арифметический режим
DP	Указатель страниц памяти данных
FRCT	Дробный режим
HM	Режим HOLD (удержания)
INTM	Режим прерывания
IPTR	Указатель страницы векторов прерываний
MP/MC#	Микропроцессор/микрокомпьютер
OVA	Флаг переполнения A
OVB	Флаг переполнения B
OVLY	Оверлей ОЗУ
OVM	Режим переполнения
SXM	Режим расширения знака
TC	Флаг тестирования/управления
XF	Состояние внешнего флага

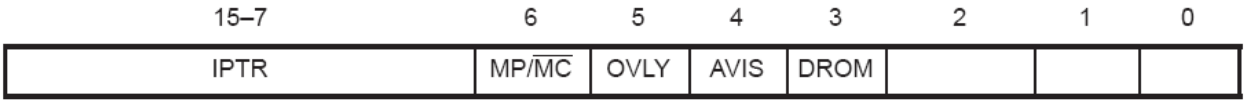


Рисунок Б.1 – Регистр состояния режима процессора (PMST)

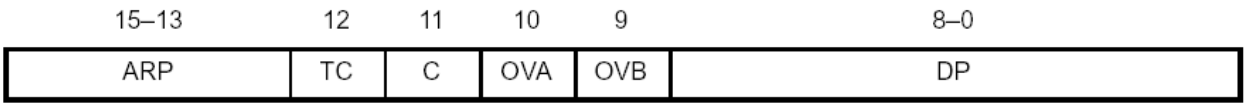


Рисунок Б.2 – Регистр состояния 0 (ST0)

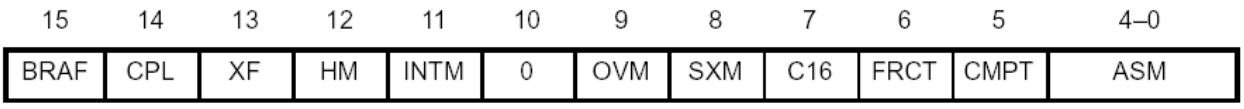


Рисунок Б.3 – Регистр состояния 1 (ST1)

Лист регистрации изменений								
Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ докум.	Подп.	Дата
	изме-ненных	заме-ненных	новых	анну-лиро-ванных				