НАБОР КОМАНД 15

15.1 СПИСОК КОМАНД

В данной главе содержится вся справочная информация по набору команд для процессоров семейства ADSP-2100. Отдельные команды организованы в группы команд. Ниже приводится список всех команд.

Команды АЛУ
Сложение/Сложение с переносом
Вычитание X-Y/Вычитание X-Y с заемом
Вычитание Y-X/Вычитание Y-X с заемом
И, ИЛИ, исключающее ИЛИ
Проверка бита, установка бита, сброс бита, переключение бита
Пропустить/Очистить
Отрицание
НЕ
Абсолютное значение
Инкремент
Декремент
Деление
Генерирование статуса АЛУ

Команды умножителя-накопителя
Умножение
Умножение с накоплением
Умножение с вычитанием
Очистить
Передача MR
Условное насыщение MR

Команды устройства сдвига
Арифметический сдвиг
Логический сдвиг
Нормализация
Нахождение порядка
Нахождение блочного порядка
Непосредственный арифметический сдвиг
Непосредственный логический сдвиг

Команды пересылки данных
Пересылка данных в/из регистра
Непосредственная загрузка регистра
Считывание из памяти данных (прямая адресация)
Считывание из памяти данных (косвенная адресация)
Считывание из памяти программы (косвенная адресация)
Запись в память данных (прямая адресация)
Запись в память данных (косвенная адресация)
Запись в память программы (косвенная адресация)
Чтение/запись области ввода/вывода

Команды последовательности выполнения программы
JUMP (Переход)
CALL (Вызов)
Переход или Вызов по состоянию вывода FI "вход флага"
Модифицировать состояния на выводе "выход флага"
Возвращение из подпрограммы
Возвращение из программы обслуживания прерывания
Do Until (Выполнять ... пока)
IDLE

Другие команды
Управление стеком
Управление режимом

15 НАБОР КОМАНД

Модификации регистра адреса

NOP (нет операций)

Разрешение и блокирование прерываний

Многофункциональные команды

Выполнение операций в АЛУ/Умножителе/Устройстве сдвига с одновременным считыванием из памяти
Выполнение операций в АЛУ/ Умножителе/Устройстве сдвига с одновременной пересылкой данных в/из регистра данных
Выполнение операций в АЛУ/Умножителе/Устройстве сдвига с одновременной записью в память
Считывание из памяти данных и памяти программы
Выполнение операций в АЛУ/ Умножителе с одновременным считыванием данных из памяти данных и памяти программы

15.2 ОБЗОР

В данной главе дается обзор и детальное описание набора команд, использующихся в цифровых сигнальных микропроцессорах семейства ADSP-2100.

Всю информацию о программных средствах отладки процессоров семейства ADSP-2100 можно найти в следующих англоязычных изданиях: *"ADSP-2100 Family Assembler Tools&Simulator Manual", "ADSP-2100 Family C Tools Manual", "ADSP-2100 Family C Runtime Library Manual"*. В перечисленных руководствах пользователя содержатся подробные указания по использованию программных средств отладки. Описание и примеры программ с исходным кодом приводятся в двухтомном справочнике *"Digital Signal Processing Applications Using the ADSP-2100 Family"*.

Набор команд приспособлен для алгоритмов, использующихся в цифровой обработке данных. Так, например, поддерживаются операции умножения с накоплением за один цикл. Набор команд полностью обеспечивает управление тремя вычислительными устройствами процессора: АЛУ, умножителем-накопителем и устройством сдвига. Арифметические команды могут непосредственно обрабатывать 16-разрядные операнды с одинарной точностью; предусмотрены возможности операций с многократной точностью.

Синтаксис исходного кода процессоров семейства ADSP-2100 является эффективным и удобным для чтения синтаксисом высокого уровня. В отличие от многих языков Ассемблера, в наборе команд процессоров семейства ADSP-2100

для арифметических операций и операций пересылки данных используется алгебраическое представление, обеспечивающее удобство чтения исходного кода. Это удобство не оказывает отрицательного влияния на работу процессора; каждое предложение программы транслируется в 24-разрядную команду, которая выполняется в течение одного цикла. В наборе команд нет ни одной команды, выполнение которой занимает несколько циклов. (Дополнительные циклы могут потребоваться при обращении к памяти, если ее быстродействие недостаточно, или при возникновении ситуации соперничества при получении доступа к внешней памяти; в остальных случаях все команды выполняются за один цикл).

Кроме команд JUMP и CALL в наборе команд имеются команды управления, поддерживающие выполнение большинства вычислений по условию и команда организации цикла DO UNTIL. Также предусмотрены команды возвращения в главную программу из подпрограммы (RTS) и после обслуживания прерывания (RTI).

Во всех процессорах семейства имеется команда IDLE, по которой процессор бездействует до очередного прерывания. По команде IDLE процессор входит в режим пониженной потребляемой мощности, ожидая прерывания.

Для выборки данных из памяти предусмотрены два режима адресации. При прямой адресации используются непосредственные значения адреса; при косвенной адресации используются индексные регистры (I) двух генераторов адреса данных.

24-разрядное командное слово позволяет достичь высокой степени параллелизма при выполнении операций. Каждая из нижеследующих комбинаций операций может быть выполнена за один цикл:

- любая операция АЛУ, умножителя или устройства сдвига (по условию или без условия);
- любая пересылка данных между регистрами;
- любая операция считывания из памяти или записи в память;
- вычисление с одновременной пересылкой данных между регистрами;
- вычисление с одновременным считыванием или записью в память;
- вычисление с одновременным считыванием из памяти данных и памяти программы.

Набор команд обеспечивает максимальную гибкость программы, предусматривая возможность пересылок данных из одного регистра в другой и из большинства регистров в память, и наоборот. Кроме того, большинство операций АЛУ, умножителя и устройства сдвига могут выполняться в комбинации с пересылкой данных между регистрами или с пересылкой данных из регистра во внутреннюю или внешнюю память и, наоборот, из внутренней или внешней памяти в какой-либо регистр.

15 НАБОР КОМАНД

15.3 ТИПЫ КОМАНД И УСЛОВНЫЕ ОБОЗНАЧЕНИЯ

Команды процессоров семейства ADSP-2100 сгруппированы в следующие категории:

- вычислительные команды АЛУ, умножителя и устройства сдвига;
- команды пересылки данных;
- команды управления последовательностью выполнения программы;
- многофункциональные команды;
- остальные.

Так как многофункциональные команды являют собой самый наглядный пример возможностей, обеспечивающихся архитектурой процессоров этого семейства, следующий раздел посвящен описанию этой группы команд.

На протяжении данной главы вы найдете таблицы, обобщающие сказанное и синтаксис команд различных групп. В этих таблицах и в ходе описания команд используются следующие условные обозначения:

Квадратные скобки [ ]	Все, что дается в квадратных скобках, является дополнительной частью оператора команды
Параллельные линии | |	В такие вертикальные линии заключается список операндов. Должен быть выбран один из перечисленных в списке операндов. Если вертикальные линии находятся внутри квадратных скобок, то операнд является дополнительным для данной команды.
ЗАГЛАВНЫЕ БУКВЫ	Заглавными буквами обозначаются названия команд (например, ADD), названия регистров или операндов. Эти литеральные константы должны набираться в программах точно так, как они показаны.
Операнды	Операнды некоторых команд показаны прописными буквами. Эти операнды могут принимать различные значения в кодах ассемблера. Например, операнд yop может быть одним из регистров AY0, AY1 или AF.
<exp>	Обозначает порядок (величину сдвига) в команде непосредственного сдвига; должен выражаться 8-разрядной знаковой целочисленной константой.
<data>	Обозначает непосредственное значение данных. Это также может быть символ (метка адреса или имя переменной/буфера), отмеченный оператором "%" или "^".

<addr>	Обозначает непосредственное значение адреса, который кодируется в команде. В качестве <addr> может использоваться либо непосредственное значение (константа) либо метка программы.
<reg>	Относится к любому доступному регистру, см. табл.15.7
<dreg>	Используется по отношению к любому регистру данных, см. табл. 15.7

Непосредственные значения <exp>, <data>, <addr> могут быть выражены константами в десятичном, шестнадцатиричном, восьмиричном или двоичном формате. По умолчанию принимается десятичный формат.

15.4 МНОГОФУНКЦИОНАЛЬНЫЕ КОМАНДЫ

В многофункциональных командах используются преимущества присущей архитектуре процессоров ADSP-2100 высокой степени параллелизма, за счет чего становится возможным одновременное выполнение за один цикл пересылок данных, операций считывания/записи в память и вычислений.

15.4.1 Выполнение операций АЛУ/Умножителя с одновременным считыванием из памяти данных и памяти программы

Суммирование результатов является, пожалуй, самой распространенной в алгоритмах цифровой обработки сигналов операцией. Выполняется она следующим образом:

- осуществляется выбор двух операндов (например, коэффициент и данные);
- производится перемножение операндов и суммирование результата умножения с полученными до этого результатами.

Процессоры семейства ADSP-2100 могут выполнять за один цикл операции выборки данных и умножения с накоплением. Как правило, для выражения цикла умножения с накоплениями в исходном коде ADSP-2100 требуется всего две строчки программы. Так как внутренняя память процессоров семейства ADSP-2100 обладает достаточным быстродействием, чтобы обеспечить выбор операнда и следующей команды в течение одного цикла, при выполнении таких циклов программы поддерживается высокая производительность (цикл программы выполняется за один командный цикл). Примером такой команды может служить:

```
MR=MR+MX0*MY0(SS), MX0=DM(I0,M0), MY0= PM(I4,M5);
```

15 НАБОР КОМАНД

В первом операторе данной команды (до первой запятой) указывается, что в MR, регистр результата умножителя-накопителя, записывается сумма его предыдущего значения и произведения (текущих) значений входных регистров умножителя X и Y (MX0 и MY0), причем оба слагаемых рассматриваются в знаковом формате (SS).

Второй и третий операторы данной многофункциональной команды осуществляют выбор двух новых операндов. Один операнд выбирается из памяти данных (DM) по адресу, указанному в индексном регистре I0, содержимое которого затем модифицируется на значение, содержащееся в регистре M0. Второй операнд выбирается из памяти программы (PM) по адресу, указанному в регистре I4 (содержимое которого, в данном случае, модифицируется величиной, содержащейся в регистре M5). Обратите внимание, что для косвенной адресации памяти используется синтаксис, сходный с тем, который используется при индексной адресации массивов, а значения индексов берутся из регистров генераторов адреса данных. Любой индексный (I) регистр одного генератора адреса данных может использоваться в паре с любым регистром модификации (M) того же генератора адреса данных.

Как уже обсуждалось в главе 2, "Вычислительные устройства", считывание данных из регистров производится в начале цикла, а запись в регистры - в конце цикла. Операнды, находящиеся в регистрах MX0 и MY0 в начале командного цикла, перемножаются, а затем произведение добавляется к содержащемуся в регистре результата умножителя, MR, значению. После окончания операции умножения старые операнды перезаписываются новыми, которые были выбраны в конце того же командного цикла и становятся доступными для вычислений с ними в следующем цикле. Конечно, одновременно с вычислением можно загрузить любой регистр данных, а не только регистры умножителя/накопителя, как в приведенном выше примере.

Вычислительной частью данной многофункциональной команды может быть любая безусловно выполняемая команда АЛУ, за исключением деления, или любая команда умножителя, кроме насыщения. Имеются и некоторые другие исключения: следующий операнд X должен загружаться в регистр MX0 из памяти данных, а новый операнд Y должен загружаться в регистр MY0 из памяти программ (для команд данного уровня несущественно из внутренней или из внешней памяти). Результат вычисления должен записываться в регистр результата (MR или AR), ни в коем случае не в регистр обратной связи (MF или AF).

15.4.2 Одновременное считывание из памяти данных и памяти программ

Команды одновременного считывания из памяти данных и памяти программ представляют собой разновидность описанного выше вида многофункциональных команд, их особенностью является отсутствие вычислений. Эти команды задают только одновременную выборку двух операндов, как можно видеть из примера ниже:

```
A X 0 = D M ( I 2 , M 0 ) ,    A Y 0 = P M ( I 4 , M 6 ) ;
```

В данном примере в качестве регистров назначения используются входные регистры АЛУ. Как и в предыдущем типе многофункциональных команд, операнды X должны извлекаться из памяти данных, а операнды Y - из памяти программ (и в том и в другом случае из внешней или из внутренней памяти процессоров с внутренней памятью на кристалле).

15.4.3 Выполнение вычислений с одновременным считыванием из памяти

Если, в отличие от описанных выше многофункциональных команд, вместо двойного считывания из памяти, команда задает выполнение только одной операции считывания из одного типа памяти, то одновременно с выборкой данных из памяти можно произвести различного рода вычисления. К числу таких разрешенных вычислений относятся все операции АЛУ, за исключением деления, все операции умножителя-накопителя и все операции устройства сдвига, за исключением непосредственного сдвига (SHIFT IMMEDIATE). Все вычисления должны быть безусловными. Примером многофункциональной команды такого типа может служить:

```
A R = A X 0 + A Y 0 ,    A X 0 = D M ( I 0 , M 3 ) ;
```

В данном случае в АЛУ выполняется операция сложения, в то время как из памяти данных выбирается один операнд. Этот тип многофункциональных команд имеет те же ограничения, что и предыдущий. Значение, хранящееся в регистре AX0 и используемое как исходное в ходе вычислений, представляет собой содержимое этого регистра в начале цикла. В конце цикла в результате операции считывания данных из памяти в регистр AX0 загружается новое значение. По той же самой причине регистром назначения (AR в рассматриваемом примере) не может быть регистр, указанный в качестве регистра назначения при считывании данных из памяти.

15 НАБОР КОМАНД

15.4.4 Выполнение вычислений с одновременной записью данных в память

Команда вычисления с одновременной записью в память имеет ту же структуру, что и команда вычисления с одновременным считыванием из памяти: однако, порядок операторов в строке данной команды изменен на обратный. Сначала выполняется запись данных в память, а затем вычисление:

```
D M ( I 0 , M 0 ) = A R ,    A R = A X 0 + A Y 0 ;
```

В приведенной команде исходным значением для записи в память (в данном примере это значение, содержащееся в регистре AR) является значение, хранящееся в этом регистре в начале цикла. В результате произведенного вычисления в этот же регистр загружается новое значение, которое является содержимым регистра AR в конце цикла. При постановке операторов данной команды в обратном порядке, что является неразрешенным действием, ассемблер генерирует предупреждение, подразумевающее, что в память записывается результат вычисления, а не предыдущее значение регистра, которое должно было записываться в память. Использование при этом одного и того же регистра не является обязательным, хотя именно таким образом чаще всего организуется конвейер операндов для вычисления.

Для вычислительных операций по данной команде действительны те же ограничения, что и для команд выше. Разрешенными являются все операции АЛУ, кроме деления, все операции умножителя-накопителя и все операции устройства сдвига, кроме непосредственного сдвига. Вычисления должны быть безусловными.

15.4.5 Выполнение вычислений с одновременной пересылкой данных между регистрами

Многофункциональные команды этого последнего типа задают выполнение пересылки данных из одного регистра данных в другой с одновременным выполнением какой-либо вычислительной операции. Большинство ограничений, существенных для предыдущих типов многофункциональных команд, также относятся и к таким командам.

```
A R = A X 0 + A Y 0 ,    A X 0 = M R 2 ;
```

В вышеприведенном примере операция сложения в АЛУ производится одновременно с загрузкой в регистр AX0 нового значения, взятого из регистра MR2. Как и в предыдущих примерах, для вычисления используется значение, содержащееся в регистре AX0 в начале цикла. Данные могут пересылаться из или

во все регистры ввода или вывода АЛУ, умножителя-накопителя и устройства сдвига, за исключением регистров обратной связи (AF и MF) и регистра SB.

В рассматриваемом примере новое значение загружается в регистр AX0 из регистра данных в конце цикла. Разрешенными являются все операции АЛУ, кроме деления, все операции умножителя-накопителя и все операции устройства сдвига, кроме непосредственного сдвига. Вычисления должны быть безусловными.

Полный список регистров данных приводится в таблице 15.7. Полный список доступных операндов *yop* и *xop* для всех команд приводится вместе с описанием каждой конкретной команды. Разрешенные комбинации операторов для многофункциональных команд приведены в табл.15.1. Операции, указанные в одном ряду, могут выполняться одновременно в многофункциональной команде.

Таблица 15.1

Комбинации операторов, разрешенные для многофункциональных команд

Безусловные вычисления	*Пересылка данных* (DM = Генератор адреса данных 1)	*Пересылка данных* (PM = Генератор адреса данных 2)
Нет или любая операция АЛУ (кроме деления) или умножителя-накопителя	Считывание из памяти данных	Считывание из памяти программ
Любая операция АЛУ, кроме деления Любая операция умножителя-накопителя	Считывание из памяти данных ———	——— Считывание из памяти программ
Любой сдвиг, кроме непосредственного	Запись в память данных ——— Пересылка данных между регистрами	——— Запись в память программ

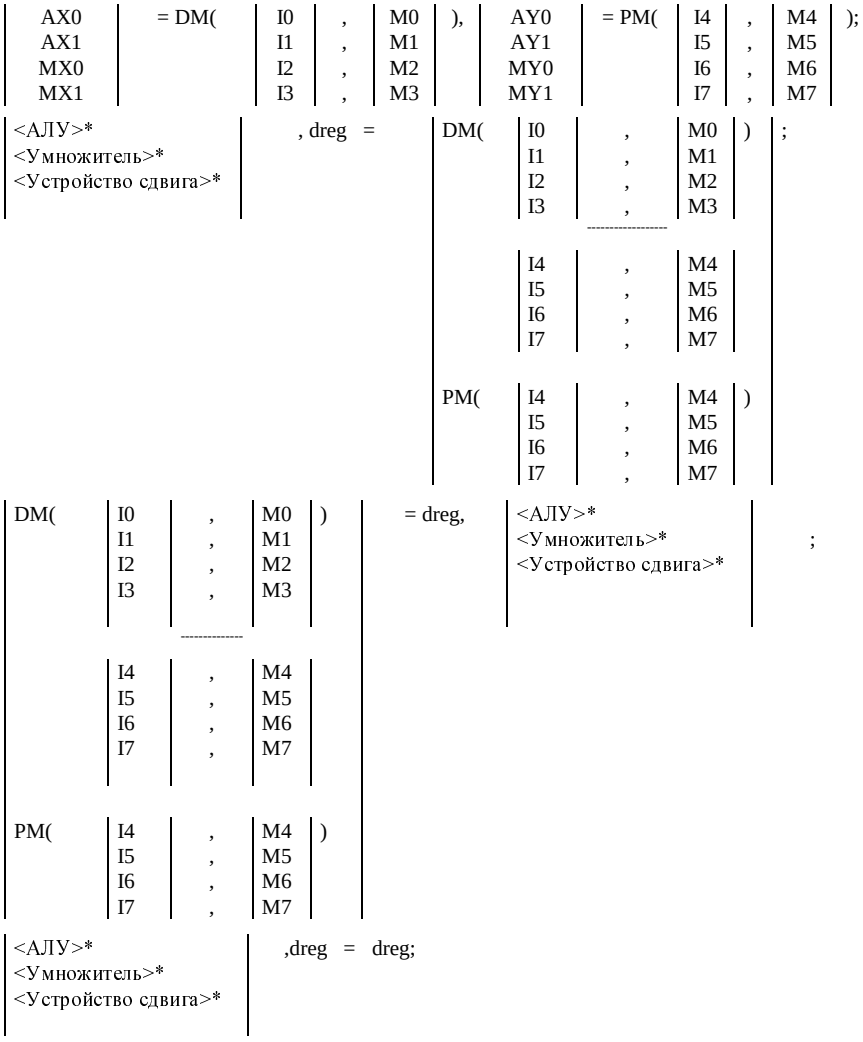
Таблица 15.2

Многофункциональные команды

<АЛУ>*+ <Умножитель>*+	,	AX0 AX1 MX0 MX1	= DM(	I0 I1 I2 I3	,	M0 M1 M2 M3	),	AY0 AY1 MY0 MY1	= PM(	I4 I5 I6 I7	,	M4 M5 M6 M7	);

15 НАБОР КОМАНД

табл. 15.2 (продолжение)



<ALU> Любая команда АЛУ (*Кроме DIVS, DIVQ)*
<MAC> Любая команда умножителя-накопителя
<SHIFT> Любая команда устройства сдвига *(кроме непосредственного сдвига)*
* Не может быть командой по условию
+ Должны использоваться регистры AR и MR - а не регистры обратной связи AF и MF (см. раздел 15.4.1 "Выполнение операций АЛУ/умножителя с одновременным считыванием данных из памяти данных и памяти программы").

15.5 КОМАНДЫ АЛУ, УМНОЖИТЕЛЯ-НАКОПИТЕЛЯ И УСТРОЙСТВА СДВИГА

Эта группа команд задает выполнение вычислений. Все эти команды, за исключением команд деления АЛУ и непосредственного сдвига, могут выполняться по условию.

15.5.1 Группа команд АЛУ

Ниже приводится пример команды АЛУ, а именно команда Сложения/ Сложения с переносом:

```
I F  A C  A R = A X 0 + A Y 0 + C ;
```

Выражение условия (которое является необязательным для этой команды), IF AC, проверяет наличие бита переноса АЛУ (AC). Приведенная выше команда выполняется, если выставляется флаг переноса по результатам предыдущей команды, в противоположном случае выполняется NOP и выполнение программы продолжается со следующей команды. Алгебраическое выражение AR=AX0+AY0+C означает, что в регистр результата АЛУ (AR) записывается сумма значений регистров ввода АЛУ X и Y плюс значение бита переноса.

Список всех команд АЛУ дается в таблице 15.3. В этом списке *условие* используется для обозначения всех возможных условий, которые могут быть проверены, а *xop* и *yop* - для обозначения регистров, которые могут быть определены в качестве регистров ввода АЛУ. Оператор условия является не обязательным и поэтому заключен в квадратные скобки. Полный список всех разрешенных *xop* и *yop* сопровождает описание каждой команды. Полный список всех условий приводится в табл. 15.9 данного издания.

Таблица 15.3

Команды АЛУ

[IF условие]	AR AF	= xop	+ yop + C + yop + C + постоянная + постоянная + C	;
[IF условие]	AR AF	= xop	- yop - yop + C - 1 + C - 1 - постоянная - постоянная + C - 1	;

табл. 15.3 (продолжение)

[IF условие]	AR AF	=	yop	- xop - xop + C - 1 - xop + C - 1 - xop + постоянная - xop + постоянная + C -1 ;
[IF условие]	AR AF	= xop	AND OR XOR	yop постоянная ;
[IF условие]	AR AF	=	TSTBIT n OF xop SETBIT n OF xop CLRBIT n OF xop TGLBIT n OF xop	;
[IF условие]	AR AF	= PASS	yop yop постоянная	;
[IF условие]	AR AF	= –	xop yop	;
[IF условие]	AR AF	= NOT	xop yop	;
[IF условие]	AR AF	= ABS	xop	;
[IF условие]	AR AF	= yop	+ 1	;
[IF условие]	AR AF	= yop	- 1	;

DIVS yop, xop;
DIVQ xop;
NONE=<ALU>;

15.5.2 Группа команд умножителя-накопителя

Ниже приводится пример одной из команд умножителя-накопителя:

```
I F  N O T  M V  M R = M R + M X 0 * M Y 0 ( U U ) ;
```

Условие, IF NOT MV, проверяет состояние бита переполнения умножителя. Если это условие не истинно, выполняется NOP. Выражение MR=MR +MX0*MY0 обозначает операцию умножения с накоплением: в регистр результата умножителя-накопителя записывается сумма его текущего значения и

произведения операндов X и Y из выбранных регистров ввода. Модификатор в скобках (UU) обозначает, что операнды принимаются за беззнаковые величины. Только один такой модификатор может выбираться из набора доступных модификаторов. (SS), напротив, означает, что оба операнда являются знаковыми величинами, в то время как (US) и (SU) означают, что либо первый, либо второй операнд является знаковым; (RND) означает округление (подразумевается знакового) результата.

В табл. 15.4. приводится общий список всех команд умножителя/накопителя. В этом списке *условие* используется для обозначения всех возможных условий, которые могут быть проверены, а *xop* и *yop* - для обозначения регистров, которые могут быть определены в качестве регистров ввода умножителя/накопителя. Полный список всех разрешенных *xop* и *yop* сопровождает описание каждой команды.

Таблица 15.4

Команды умножителя-накопителя

[IF условие]	MR MF	= xop *	yop xop	(	SS SU US UU RND	) ;		
[IF условие]	MR MF	= MR + xop *	yop xop	(	SS SU US UU RND	) ;		
[IF условие]	MR MF	= MR - xop *	yop xop	(	SS SU US UU RND	) ;		
[IF условие]	MR MF	= 0 ;						
[IF условие]	MR MF	= MR [(RND)];						

IF MV SAT MR;

15 НАБОР КОМАНД

15.5.3. Группа команд устройства сдвига

Ниже дается пример команды устройства сдвига, а именно - команды нормализации:

```
IF NOT CE SR=SR OR NORM SI (HI);
```

Выражение условия, IF NOT CE, производит проверку условия заполненности счетчика ("счетчик не пуст"). Если это условие не является истинным, то выполняется NOP. Регистром назначения для всех операций устройства сдвига является регистр результата устройства сдвига SR. (Регистрами назначения для команд нахождения порядка являются SE или SB, как будет показано ниже). Размер и направление сдвига для всех операций устройства сдвига, за исключением непосредственного сдвига, определяется знаковой величиной, содержащейся в регистре SE. Положительные значения, содержащиеся в этом регистре, вызывают сдвиг влево; отрицательные - вправо.

Модификатор "SR OR" (который является необязательным для рассматриваемой команды) задает операцию логического ИЛИ над полученным результатом и текущим содержимым регистра SR, что позволяет получить 32-разрядное значение из двух его 16-разрядных кусочков. "NORM" является оператором, а модификатор определяет, производится ли сдвиг относительно старших (HI) или младших (LO) 16 бит регистра SR. В случае отсутствия "SR OR" полученный результат сразу передается в регистр SR.

Общий список всех команд устройства сдвига приводится в табл. 15.5. В этой таблице *условие* используется для обозначения всех возможных условий, истинность которых можно проверить.

Таблица 15.5

Команды устройства сдвига

[IF условие]	SR	=	[SR OR] ASHIFT xop	(	HI LO	);
[IF условие]	SR	=	[SR OR] LSHIFT xop	(	HI LO	);
[IF условие]	SR	=	[SR OR] NORM xop	(	HI LO	);
[IF условие]	SR	=	EXP xop	(	HI LO HIX	);
[IF условие]	SR	=	EXPADJ xop ;			
SR	=	[SR OR] ASHIFT xop BY <exp>		(	HI LO	);
SR	=	[SR OR] LSHIFT xop BY <exp>		(	HI LO	);

15.6 ПЕРЕСЫЛКА ДАННЫХ: СЧИТЫВАНИЕ И ЗАПИСЬ

Команды пересылки данных (см. табл. 15.6) осуществляют пересылку данных из и в регистры данных и из и во внешнюю память. Регистры делятся на две группы. Регистры первой группы обозначаются как *reg* и включают в себя почти все регистры. Регистры второй группы (регистры данных) обозначаются как *dreg* и являются разновидностью первых. Недоступными остаются только счетчик программ (PC) и регистры обратной связи АЛУ и умножителя-накопителя (AF и MF).

Табл. 15.7 показывает какие регистры принадлежат каким группам. Многие регистры управления отображены в карте памяти (в процессорах с внутренней памятью на кристалле); запись и считывание из этих регистров задается по адресам в памяти, а не по названию регистра.

Таблица 15.6

Команды пересылки данных

```
reg      =      reg ;
reg      =      DM(<адрес>);

dreg     =      DM ( | I0 | , | M0 | ) ;
                     | I1 | , | M1 |
                     | I2 | , | M2 |
                     | I3 | , | M3 |
                     ----------------
                     | I4 | , | M4 |
                     | I5 | , | M5 |
                     | I6 | , | M6 |
                     | I7 | , | M7 |

DM ( | I0 | , | M0 | )  =  | dreg      | ;
     | I1 | , | M1 |       | <данные>  |
     | I2 | , | M2 |       |           |
     | I3 | , | M3 |       |           |
     ----------------
     | I4 | , | M4 |
     | I5 | , | M5 |
     | I6 | , | M6 |
     | I7 | , | M7 |

DM (<адрес>)  =  reg;
reg      =       <данные>;

dreg     =      PM ( | I4 | , | M4 | ) ;
                     | I5 | , | M5 |
                     | I6 | , | M6 |
                     | I7 | , | M7 |
```

табл. 15.6 (продолжение)

PM (	I4	,	M4	)	dreg;
	I5	,	M5		
	I6	,	M6		
	I7	,	M7		

Таблица 15.7

Регистры процессора: reg и dreg

*Регистры**: reg***

	*Регистры данных: **dreg***
SB	AX0, AX1, AY0, AY1, AR
PX	MX0, MX1, MY0, MY1, MR0, MR1, MR2
I0 -I7, M0 -M7, L0 - L7	SI, SE, SR0, SR1
CNTR	
ASTAT, MSTAT, SSTAT	
IMASK, ICNTL, IFC	
TX0, TX1, RX0, RX1	

15.7 КОМАНДЫ УПРАВЛЕНИЯ ПОСЛЕДОВАТЕЛЬНОСТЬЮ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Управление последовательностью выполнения программы осуществляется в процессорах семейства ADSP-2100 простыми, но эффективными средствами. Ниже приводится пример одной из таких команд:

```
IF EQ JUMP my_label;
```

Команда JUMP, естественно, знакома многим по работе с другими процессорами. *My_label* используется для обозначения любого идентификатора, который вы хотите использовать в качестве метки того оператора, к которому осуществляется переход. Вместо метки можно также использовать индексный регистр Генератора адреса данных 2. По умолчанию областью действия метки является тот модуль (блок) исходного кода, в котором эта метка объявлена. Директива ассемблера .ENTRY делает метку видимой в качестве точки входа подпрограмм, находящихся за пределами этого модуля программы. И наоборот, с помощью директивы .EXTERNAL можно использовать метку, находящуюся в другом модуле программы.

Если в качестве условия используется условие заполненности счетчика (CE, NOT CE), то для инициализации значения счетчика следует выполнить операцию присвоения значения регистру CNTR. В командах JUMP и CALL разрешается использовать дополнительные операторы условия "FLAG_IN" и "NOT

FLAG_IN" для организации условных переходов в соответствии с состоянием вывода флага FI, но только, если в качестве источника адреса при этом используется прямая адресация, а не Генератор адреса данных 2

Команды RTS (возвращение из подпрограммы) и RTI (возвращение в главную программу после обслуживания прерывания) могут обеспечивать, соответственно, возврат по условию из подпрограммы, вызванной командой CALL, или от векторов прерывания.

С помощью команды IDLE можно задать ожидание прерываний. Получив команду IDLE, процессор ожидает прерывания, находясь в это время в состоянии пониженной потребляемой мощности. После обслуживания прерывания управление программой вновь переходит к команде, следующей за командой IDLE. Выполнение команды IDLE требует меньших затрат мощности, чем организация циклов с помощью команды JUMP.

Общий список всех команд управления последовательностью выполнения программы приведен в табл. 15.8. Коды *условий* объясняются в табл. 15.9.

Таблица 15.8

Команды управления последовательностью выполнения программы

[IF условие]		JUMP	(I4) (I5) (I6) (I7) <адрес>	;
IF	FLAG_IN NOT FLAG_IN		JUMP	<адрес>;
[IF условие]		CALL	(I4) (I5) (I6) (I7) <адрес>	;
IF	FLAG_IN NOT FLAG_IN		CALL	<адрес>;

[IF условие] RTS;
[IF условие] RTI;
DO <адрес> [UNTIL условие прекращения цикла];
IDLE [(n)];

15 НАБОР КОМАНД

Таблица 15.9

Коды условий

Синтаксис	*Условие состояния*	*Истинно, если*
EQ	Равно нулю	AZ=1
NE	Не равно нулю	AZ=0
LT	Меньше нуля	AN.XOR.AV=1
GE	Больше или равно нулю	AN.XOR.AV=0
LE	Меньше или равно нулю	(AN.XOR.AV).OR.AZ=1
GT	Больше нуля	(AN.XOR.AV).OR.AZ=0
AC	Перенос в АЛУ	AC=1
NOT AC	Нет переноса в АЛУ	AC=0
AV	Переполнение в АЛУ	AV=1
NOT AV	Нет переполнения в АЛУ	AV=0
MV	Переполнение в умножителе-накопителе	MV=1
NOT MV	Нет переполнения в умножителе-накопител	MV=0
NEG	Операнд X последней команды ABS был отрицательным	AS=1
POS	Операнд X последней команды ABS был положительным	AS=0
NOT CE	Счетчик не пуст	
FLAG_IN*	Значение на выводе FI	Последнее значение на выводе FI = 1
NOT FLAG_IN*	Значение на выводе FI	Последнее значение на выводе FI =0

*Только в командах JUMP и CALL

15.8 ДРУГИЕ КОМАНДЫ

Имеется несколько таких команд, которые нельзя отнести ни к одной из выделенных выше групп. NOP - это команда отсутствия операций. Команды PUSH/POP позволяют явным образом непосредственно управлять содержимым стеков состояния, счетчика, счетчика программ (PC) и циклов. Некоторые из этих стеков автоматически заполняются и опустошаются во время обслуживания прерываний.

Команда управления режимом разрешает и блокирует режимы некоторых операций процессора. Эта команда управляет режимами, общими для всех процессоров семейства ADSP-2100 (бит-реверсной адресацией в Генераторе адреса данных 1, фиксацией переполнения АЛУ, насыщением регистра результата АЛУ,

выбором набора основных или теневых регистров, режимом GO для непрекращающейся работы процессора во время предоставления шины, режимом сдвига в умножителе для выполнения целочисленных или дробных вычислений, активизацией таймера).

За операторами ENA или DIS может следовать через запятую сколько угодно идентификаторов; ENA и DIS могут повторяться в командной строке. Все семь режимов могут разрешаться, блокироваться или изменяться при помощи всего одной команды.

Команда MODIFY позволяет модифицировать указатель адреса в заданном регистре I на значение, содержащееся в заданном регистре M, не обращаясь при этом к памяти. Как и во всех других случаях, регистры I и M должны быть регистрами одного и того же генератора адреса данных; любой регистр I0-I3 может использоваться в комбинации с любым регистром M0-M3, а любой регистр I4-I7 - с любым регистром M4-M7. В случае использования циклической буферизации применяется логика адресации по модулю (см. главу 4 "Передача данных").

Выводы FO (выход флага), FL0, FL1 и FL2 могут устанавливаться, сбрасываться или работать как тумблеры. Эта команда обеспечивает возможность управления мультипроцессорной структурой.

Таблица 15.10

Разные команды

```
NOP ;

[ | PUSH | STS ] [, POP CNTR] [, POP PC] [, POP LOOP] ;
  | POP  |     ]
  |      |

| ENA | | BIT_REV    | [ , ]  ;
| DIS | | AV_LATCH   |
          | AR_SAT     |
          | SEG_REG    |
          | G_MODE     |
          | M_MODE     |
          | TIMER      |

MODIFY ( | I0 | , | M0 | ) ;
         | I1 | , | M1 |
         | I2 | , | M2 |
         | I3 | , | M3 |
          ----------------------------
         | I4 | , | M4 |
         | I5 | , | M5 |
         | I6 | , | M6 |
         | I7 | , | M7 |
```

табл. 15.10 (продолжение)

[IF условие]	SET RESET TOGGLE		FLAG_OUT FL0 FL1 FL2	[,] ;

ENA DIS	INTS;

15.9 ВРЕМЯ ЦИКЛА

Все команды выполняются за один цикл, но в некоторых случаях, которые описываются ниже, выполнение команды может потребовать дополнительных циклов.

15.9.1 Многократное обращение к внешней памяти

Шины адреса и данных процессоров семейства ADSP-2100 мультиплексированы вне кристалла. Поэтому эти процессоры могут при выполнении команды только один раз в течение цикла обращаться к внешним устройствам. Если для одной и той же команды требуется два обращения к внешней памяти, например, выборка команды и выборка данных или одновременная выборка данных из памяти данных и из памяти программы, то возникает дополнительный цикл. В этом случае, сначала выполняется обращение к памяти программы, а затем обращение к памяти данных. Если же требуются три обращения к внешней памяти (выборка команды и две выборки данных из памяти программы и памяти данных), то возникает два дополнительных цикла.

Для выполнения многофункциональной команды из внешней памяти требуется выбрать саму команду и два слова данных. Пока из внешней памяти производится только одна выборка, для выполнения такой команды не требуется никаких дополнительных циклов. (Две остальные выборки должны быть из внутренней памяти данных или программы).

15.9.2 Состояния ожидания

Все процессоры, кроме ADSP-2100, поддерживают программируемые состояния ожидания для внешних ЗУ. К времени, требуемому процессору для доступа к внешней памяти, может добавляться до семи состояний ожидания. Появившиеся благодаря состояниям ожидания дополнительные циклы, добавляются к

циклам, возникающим в результате нескольких обращений к внешней памяти (как описывалось выше). Программирование состояний ожидания обсуждалось в главе 10, "Интерфейс памяти".

Состояния ожидания и многократное обращение к внешней памяти являются теми двумя случаями, когда дополнительный цикл генерируется во время выполнения команды. В случае автобуферизации последовательного порта и прямого доступа к памяти (ADSP-2181) дополнительные циклы вставляются *между* командами.

15.9.3 Автобуферизация последовательного порта и прямой доступ к данным

Если для передачи слов данных последовательного порта из или в память данных используется автобуферизация или прямой доступ (ADSP-2181), то каждая такая передача "крадет" одно обращение к памяти. Занятие цикла под обращение к памяти происходит только между завершенными командами. Если для выполнения какой-либо команды требуются дополнительные циклы (по одной из указанных выше причин), то процессор ожидает завершения выполнения команды перед тем, как "украсть" цикл под обращение к памяти.

15.10 СИНТАКСИС КОМАНД

В следующих разделах рассказывается о синтаксисе команд и условных обозначениях, принятых при описании каждой команды.

15.10.1 Пунктуация и многофункциональные команды

В конце всех команд ставится точка с запятой. Запятая отделяет операторы многофункциональной команды, но не означает окончания такой команды. Например, приведенные ниже в Примере A операторы содержат одну многофункциональную команду (которая может выполняться за один цикл). В Примере B показаны две отдельные команды, для выполнения которых требуются два командных цикла.

Пример A: Одна многофункциональная команда

```
A X 0 = D M ( I 0 , M 0 ) ,    запятая использована в многофункциональной
AY0=PM(I4,M4);                 команде
```

Пример B: Две отдельных команды

```
A X 0 = D M ( I 0 , M 0 ) ;    точка с запятой прерывает команду
A Y 0 = P M ( I 4 , M 4 ) ;
```

15 НАБОР КОМАНД

15.10.2 Обозначения, принятые в синтаксисе процессоров семейства ADSP-2100

Ниже приводится пример команды АЛУ (Сложение/Сложение с переносом):

[IF условие]	AR AF	= хор +	уор C уор + C	;

Разрешенные *условия, хор* и *уор* имеются в списке. Оператор условия IF заключен в квадратные скобки, указывающие на его необязательность для данной команды.

Регистром назначения для операции сложения должен быть назначен либо регистр AR либо регистр AF. Эти регистры даны в параллельных вертикальных линиях, которые указывают на то, что выбран должен быть какой-либо один из них. Аналогичным образом, на месте *уор* может стоять операнд Y, бит переноса или их сумма. Следует выбрать что-то одно из трех.

15.10.3 Обозначения, принятые для регистров состояния

При рассмотрении влияния каждой команды на регистры состояния процессора используются следующие обозначения:

* Звездочка указывает на тот бит в регистре состояния, который изменяется при выполнении команды

– Тире указывает, что команда никак не влияет на данный бит регистра

0 или 1 Указывает на безусловный сброс или установку бита

В качестве примера ниже приводится слово состояния в регистре ASTAT.

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	*	-	-	-	0	-	-

В данном случае бит MV обновляется, а бит AV сбрасывается в 0.

Синтаксис:	[IF условие]	AR AF	= хор	+ уор + C + уор + C +постоянная +постоянная+C	;

Разрешенные хор *Разрешенные уор* *Разрешенные условия*

AX0	MR2	AY0	EQ	LE	AC
AX1	MR1	AY1	NE	NEG	NOT AC
AR	MR0	AF	GT	POS	MV
	SR1		GE	AV	NOT MV
	SR0		LT	NOT AV	NOT CE

Разрешенные постоянные (ADSp-217x, ADSP-218x, ADSP-21msp58/59)

0, 1,2,4,8,16,32,64,128,256,512,1024,2048,4096,8192,16384,32767
-2, -3, -5, -9, -17, -33, -65, -129, -257, -513, -1025, -2049, -4097, -8193, -16385, -32768

Пример:
```
IF EQ AR=AX0 + AY0 + C;
AR=AR+512;
```

Описание: Проверка необязательного условия и, если оно истинно, выполнение сложения заданного типа. Если условие не выполняется, то нет операций. При отсутствии условия в команде операция сложения выполняется без условия. При операции сложения первый операнд добавляется ко второму и биту переноса АЛУ (AC) (если в команде имеется " + C") с использованием двоичного сложения. Результат записывается в регистр назначения. Операнды хранятся в регистрах данных, которые определены в команде.

Генерируемое состояние:

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	-	-	-	*	*	*	*

AZ Установлен, если результат равен 0. В других случаях, сброшен.
AN Установлен, если результат - отрицательное число. Иначе сброшен.
AV Установлен, если происходит переполнение арифметического устройства.

Иначе сброшен.
AC Установлен, если генерируется перенос. Иначе сброшен.

15 Команды АЛУ
Сложение/Сложение с переносом

Формат команды:
Выполняемая по условию операция АЛУ или умножителя-накопителя. Тип 9.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	Z	AMF					Yop		Xop			0	0	0	0	COND			

AMF определяет операцию АЛУ или умножителя-накопителя, в данном случае:
AMF = 10010 для операции типа: xop + yop + C
AMF = 10011 для операции типа: xop + yop

Обратите внимание, что операция типа: xop + C, - является разновидностью операции: xop + yop + C, когда yop = 0.

Z:	Регистр назначения	Yop:	Операнд Y
Xop:	Операнд X	COND	Условие

(xop + постоянная) Выполняемая по условию операция АЛУ или умножителя-накопителя. Тип 9. (*Только ADSP-217x, ADSP-218x, ADSP-21msp58/59)*

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	Z	AMF					YY		Xop			CC		BO		COND			

AMF определяет операцию АЛУ или умножителя-накопителя, в данном случае:
AMF = 10010 для операции типа: xop + постоянная + C
AMF = 10011 для операции типа: xop + постоянная

Z:	Регистр назначения	COND:	Условие
Xop:	Операнд X		

BO, CC, YY определяют постоянную. См. Приложение A, "Коды команд"

Команды АЛУ 15
вычитание X - Y/вычитание X - Y с заемом

Синтаксис:	[IF условие]	AR AF	= хор	- уор -уор + C - 1 + C - 1 -постоянная -постоянная+C-1	;

Разрешенные хор		*Разрешенные уор*	*Разрешенные условия*		
AX0	MR2	AY0	EQ	LE	AC
AX1	MR1	AY1	NE	NEG	NOT AC
AR	MR0	AF	GT	POS	MV
	SR1		GE	AV	NOT MV
	SR0		LT	NOT AV	NOT CE

Разрешенные постоянные (ADSP-217x, ADSP-218x, ADSP-21msp58/59)
0, 1,2,4,8,16,32,64,128,256,512,1024,2048,4096,8192,16384,32767
-2, -3, -5, -9, -17, -33, -65, -129, -257, -513, -1025, -2049, -4097, -8193, -16385, -32768

Пример: IF GE AR = AX0 -AY0;

Описание: Проверка необязательно присутствующего условия и, если оно истинно, выполнение вычитания заданного типа. Если условие не выполняется, нет операций. При отсутствии условия в команде производится безусловная операция вычитания. При операции вычитания второй операнд отнимается от первого операнда и к их разности может добавляться бит переноса АЛУ (AC) минус единица (H#0001), а затем результат записывается в регистр назначения. Количественное выражение C - 1 эффективно используется в качестве заема при вычитаниях с повышенной точностью. Операнды содержатся в регистрах данных, которые определены в команде.

Генерируемое состояние:

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	-	-	-	*	*	*	*

AZ Установлен, если результат равен 0. В других случаях сброшен.
AN Установлен, если результат - отрицательное число. Иначе сброшен.
AV Установлен, если происходит переполнение арифметического устройства. Иначе сброшен.
AC Установлен, если генерируется перенос. Иначе сброшен.

15 Команды АЛУ
Вычитание X - Y/вычитание X - Y с заемом

Формат команды:
Выполняемая по условию операция АЛУ или умножителя-накопителя. Тип 9.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	Z	AMF					Yop		Xop			0	0	0	0	COND			

AMF определяет операцию АЛУ или умножителя-накопителя, в данном случае:

AMF = 10110 для операции типа: xop - yop + C - 1.

AMF = 10111 для операции типа: xop - yop.

Обратите внимание, что операция типа: xop + C - 1, - является разновидностью операции: xop - yop + C - 1, когда yop = 0.

Z:	Регистр назначения	Yop:	Операнд Y
Xop:	Операнд X	COND	Условие

(xop - постоянная) Выполняемая по условию операция АЛУ или умножителя-накопителя. Тип 9. *(Только ADSP-217x, ADSP-218x, ADSP-21msp58/59)*

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	Z	AMF					YY			Xop		CC		BO			COND		

AMF определяет операцию АЛУ или умножителя-накопителя, в данном случае:

AMF = 10110 для операции типа: xop - постоянная + C - 1.
AMF = 10111 для операции типа: xop - постоянная.

Z:	Регистр назначения	COND:	Условие
Xop:	Операнд X		

BO, CC, YY определяют постоянную. См. Приложение A, "Коды команд"

Команды АЛУ 15
Вычитание Y - X/вычитание Y - X с заемом

Синтаксис: [IF условие] | AR / AF | = | yop - | xop / xop + C - 1 | ;

- xop + C - 1
-xop+постоянная
-xop+постоянная+C-1

Разрешенные xop *Разрешенные yop* *Разрешенные условия*

AX0	MR2	AY0	EQ	LE	AC
AX1	MR1	AY1	NE	NEG	NOT AC
AR	MR0	AF	GT	POS	MV
	SR1		GE	AV	NOT MV
	SR0		LT	NOT AV	NOT CE

Разрешенные постоянные (ADSp-217x, ADSP-218x, ADSP-21msp58/59)

0, 1,2,4,8,16,32,64,128,256,512,1024,2048,4096,8192,16384,32767
-2, -3, -5, -9, -17, -33, -65, -129, -257, -513, -1025, -2049, -4097, -8193, -16385, -32768

Пример: IF GT AR=AY0 - AX0 + C - 1;

Описание: Проверка необязательного условия и, если оно истинно, выполнение вычитания заданного типа. Если условие не выполняется, то нет операций. При отсутствии условия в команде операция вычитания выполняется без условия. При операции вычитания второй операнд вычитается из первого операнда, к разности может добавляться бит переноса АЛУ (AC) минус 1(H#0001), и полученный результат записывается в регистр назначения. Количественное значение (C - 1) эффективно используется как заем в операциях вычитания с повышенной точностью. Операнды хранятся в регистрах данных, которые определены в команде.

Генерируемое состояние:

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	-	-	-	*	*	*	*

AZ Установлен, если результат равен 0. В других случаях, сброшен.

AN Установлен, если результат - отрицательное число. Иначе сброшен.

AV Установлен, если происходит переполнение арифметического устройства. Иначе сброшен.

AC Установлен, если генерируется перенос. Иначе сброшен.

15 Команды АЛУ
вычитание Y - X/вычитание Y - X с заемом

Формат команды:
Выполняемая по условию операция АЛУ или умножителя-накопителя. Тип 9.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	Z	AMF					Yop		Xop			0	0	0	0	COND			

AMF определяет операцию АЛУ или умножителя-накопителя, в данном случае:

AMF = 11010 для операции типа: yop - xop + C - 1.
AMF = 11001 для операции типа: yop - xop.

Обратите внимание, что операция типа: - xop + C - 1, - является разновидностью операции: yop - xop + C - 1, когда yop = 0.

Z:	Регистр назначения	Yop:	Операнд Y
Xop:	Операнд X	COND:	Условие

(-xop + постоянная) Выполняемая по условию операция АЛУ или умножителя-накопителя. Тип 9. *(Только ADSP-217x, ADSP-218x, ADSP-21msp58/59)*

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	Z	AMF					YY		Xop			CC		BO		COND			

AMF определяет операцию АЛУ или умножителя-накопителя, в данном случае:
AMF = 11010 для операции типа: постоянная - xop + C - 1.
AMF = 11001 для операции типа: постоянная - xop.

Z:	Регистр назначения	COND:	Условие
Xop:	Операнд X		

BO, CC, YY определяют постоянную. См. Приложение A, "Коды команд".

Синтаксис:	[IF условие]	AR AF	= хор	AND OR XOR	уор постоянная	;

Разрешенные хор		*Разрешенные уор*	*Разрешенные условия*		
AX0	MR2	AY0	EQ	LE	AC
AX1	MR1	AY1	NE	NEG	NOT AC
AR	MR0	AF	GT	POS	MV
	SR1		GE	AV	NOT MV
	SR0		LT	NOT AV	NOT CE

Разрешенные постоянные (ADSP217x, ADSP-218x, ADSP-21msp58/59)

0, 1,2,4,8,16,32,64,128,256,512,1024,2048,4096,8192,16384,32767
-2, -3, -5, -9, -17, -33, -65, -129, -257, -513, -1025, -2049, -4097, -8193, -16385, -32768

Пример: AR=AX0 XOR AY0;
IF FLAG_IN AR=MR0 AND 8192;

Описание: Проверка необязательного условия и, если оно истинно, побитовое выполнение заданной логической операции (логическое И, ИЛИ, исключающее ИЛИ). Если условие не выполняется, то нет операций. При отсутствии условия в команде логическая операция выполняется без условия. Операнды хранятся в регистрах данных, которые определены в команде.

Генерируемое состояние:

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	-	-	-	0	0	*	*

AZ Установлен, если результат равен 0. В других случаях, сброшен.

AN Установлен, если результат - отрицательное число. Иначе сброшен.

AV Всегда сброшен.

AC Всегда сброшен.

15 Команды АЛУ
Команда PASS/Сброс

Формат команды:

Выполняемая по условию операция АЛУ или умножителя-накопителя. Тип 9.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	Z	AMF					Yop		Xop			0	0	0	0	COND			

AMF определяет операцию АЛУ или умножителя-накопителя, в данном случае:

AMF = 11100 для операции логического И.
AMF = 11101 для операции логического ИЛИ.
AMF = 11110 для операции исключающего ИЛИ.

Z: Регистр назначения
Y: Операнд Y
Xop: Операнд X
COND: Условие

(xop И/ИЛИ/Исключающее ИЛИ постоянная) Выполняемая по условию операция АЛУ или умножителя-накопителя. Тип 9. *(Только ADSP-217x, ADSP-218x, ADSP-21msp58/59)*

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	Z	AMF					YY		Xop			CC		BO		COND			

AMF определяет операцию АЛУ или умножителя-накопителя, в данном случае:

AMF = 11100 для операции логического И.

AMF = 11101 для операции логического ИЛИ

AMF = 11110 для операции исключающего ИЛИ

Z: Регистр назначения
COND: Условие
Xop: Операнд X

BO, CC, YY определяют постоянную. См. Приложение A, "Коды команд"

Синтаксис:

[IF условие]				
	AR	=	TSTBIT n OF xop	;
	AF		SETBIT n OF xop	
			CLRBIT n OF xop	
			TGLBIT n OF xop	

Разрешенные xop		*Разрешенные условия*		
AX0	MR2	EQ	LE	AC
AX1	MR1	NE	NEG	NOT AC
AR	MR0	GT	POS	MV
	SR1	GE	AV	NOT MV
	SR0	LT	NOT AV	NOT CE

Разрешенные значения n (0=самый младший бит)
0, 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15

Пример: AF=TSTBIT 5 OF AR;
AR=TGLBIT 13 OF AX0;

Описание: Проверка необязательного условия и, если оно истинно, выполнение указанной операции с битом. Если условие не выполняется, то нет операций. При отсутствии условия в команде операция выполняется без условия. Эти операнды не могут использоваться в многофункциональных командах.

Указанные операции определяются следующим образом:
TSTBIT является операцией логического И выбранного бита с 1.
SETBIT устанавливает выбранный бит в 1.
CLRBIT сбрасывает выбранный бит в 0.
TGLBIT изменяет состояние выбранного бита на противоположное.

Выполнение этих команд влияет на состояние битов в ренистре состояний арифметических устройств ASTAT. Например, для проверки бита и задания соответствующего условного ветвления может быть использована следующая команда:

```
AF=TSTBIT 5 OF AR;
IF NE JUMP set; /*переход к "set", если бит 5 регистра AR установлен*/
```

Генерируемое состояние:

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	-	-	-	0	0	*	*

AZ Установлен, если результат равен 0. В других случаях, сброшен.
AN Установлен, если результат - отрицательное число. Иначе сброшен.
AV Всегда сброшен.
AC Всегда сброшен.

15 Команды АЛУ
Команда PASS/Сброс

Формат команды:

(хор И/ИЛИ/Исключающее ИЛИ постоянная)

Выполняемая по условию операция АЛУ или умножителя-накопителя. Тип 9.
(Только ADSp-217x, ADSP-218x, ADSP-21msp58/59)

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	Z	AMF					YY		Xop			CC		BO		COND			

AMF определяет операцию АЛУ или умножителя-накопителя, в данном случае:
AMF = 11100 для операции логического И.
AMF = 11101 для операции логического ИЛИ.
AMF = 11110 для операции исключающего ИЛИ

Z: Регистр назначения COND: Условие
Xop: Операнд X

BO, CC, YY определяют постоянную. См. Приложение A, "Коды команд".

Синтаксис: [IF условие] | AR / AF | = PASS | хор / уор / постоянная | ;

Разрешенные хор		*Разрешенные уор*	*Разрешенные условия*		
AX0	MR2	AY0	EQ	LE	AC
AX1	MR1	AY1	NE	NEG	NOT AC
AR	MR0	AF	GT	POS	MV
	SR1		GE	AV	NOT MV
	SR0		LT	NOT AV	NOT CE

Разрешенные постоянные (все процессоры семейства ADSP-2100)
-1,0,1
Разрешенные постоянные (ADSP217x, ADSP-218x, ADSP-21msp58/59)
2,3,4,5,6,7,8,9,15,16,17,31,32,33,63,64,65,127,128,129,255,256,257,511,512,513,1023,1024,1025,2047,2048,2049,4095,4096,4097,8191,8192,8193,16383,16384,16385,32766,32767
-2, -3, -4, -5, -6, -7, -8, -9, -10, -16, -17, -18, -32, -33, -34, -64, -65, -66, -128, -129, -130, -256, -257, -258, -512, -513, -514, -1024, -1025, -1026, -2048, -2049, -2050, -4096, -4097, -4098, -8192, -8193, -8194, -16384, -16385, -32767, -32768

Пример ы: IF GE AR= PASS AY0;
AR = PASS 0;
AR = PASS 8191;

Описание: Проверка необязательно присутствующего условия и, если оно истинно, значение операнда в блоке АЛУ не изменяется и передается таким в регистр назначения. Если условие не выполняется, то нет операций. При отсутствии условия в команде операция передачи операнда неизменным выполняется без условия. Операнд хранится в регистре данных, который определен в команде.

Команда PASS 0 является одним из способов очистки содержимого регистра AR и может комбинироваться в многофункциональной команде с операциями записи в память и считывания из памяти.

По команде PASS выполняется передача значения в регистр AR или AF, при этом меняются флаги состояний регистра ASTAT (только для хор, уор, -1, 0, 1). Эта команда отличается от команды пересылки данных в или из регистра, поскольку последняя не влияет на флаги состояния. Операция *PASS постоянна*я (в которой используются любые постоянные, а не только -1, 0 и 1) приводит к неопределенному состоянию флагов состояний в регистре состояний арифметических устройств ASTAT. Эта операция может выполняться только процессорами ADSP-217x, ADSP-218x, ADSP-21msp58/59 и не может использоваться в составе многофункциональной команды.

Генерируемое сосояние:

15 Команды АЛУ
Команда PASS/Сброс

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	-	-	-	0	0	*	*

AZ Установлен, если результат равен 0. В других случаях, сброшен.
AN Установлен, если результат - отрицательное число. Иначе сброшен.
AV Всегда сброшен.
AC Всегда сброшен.

Примечание: после выполнения команды *PASS постоянная* (с любыми постоянными, кроме -1, 0 и 1) флаги состояний регистра ASTAT не определены.

Формат команды:

Выполняемая по условию операция АЛУ или умножителя-накопителя. Тип 9.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	Z	AMF					Yop		Xop			0	0	0	0	COND			

AMF определяет операцию АЛУ или умножителя-накопителя, в данном случае:
AMF = 10000 для операции типа: PASS yop.
AMF = 10011 для операции типа: PASS xop.
AMF = 10001 для операции типа: PASS 1.
AMF = 11000 для операции типа: PASS -1.

Обратите внимание, что операция типа: PASS xop, - является разновидностью операции: xop + yop, когда yop = 0; операция: PASS 1 является разновидностью операции типа: yop + 1, когда yop = 0; операция типа: PASS -1, - является разновидностью операции yop - 1, когда yop = 0.

Z: Регистр назначения Yop: Операнд Y
Xop: Операнд X COND: Условие

Выполняемая по условию операция АЛУ или умножителя-накопителя. Тип 9.
(PASS постоянная; постоянная ≠-1,0,1)
(Только ADSP-217x, ADSP-218x, ADSP-21msp58/59)

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	Z	AMF					YY		Xop			CC		BO		COND			

AMF определяет операцию АЛУ или умножителя-накопителя, в данном случае:
AMF = 10000 для PASS yop (особый случай yop, когда yop=постоянная)
AMF = 10001 для PASS yop+1 (особый случай yop+1, когда yop=постоянная)
AMF = 11000 для PASS yop-1 (особый случай yop-1, когда yop=постоянная)
Z: Регистр назначения COND: Условие
Xop: Операнд X
BO, CC, YY определяют постоянную. См. Приложение A, "Коды команд".

Синтаксис: [IF условие] | AR / AF | = – | xop / yop | ;

Разрешенные xop		*Разрешенные yop*	*Разрешенные условия*		
AX0	MR2	AY0	EQ	LE	AC
AX1	MR1	AY1	NE	NEG	NOT AC
AR	MR0	AF	GT	POS	MV
	SR1		GE	AV	NOT MV
	SR0		LT	NOT AV	NOT CE

Пример: IF LT AR= - AY0;

Описание: Проверка необязательно присутствующего условия и, если оно истинно, отрицание исходного операнда и запись результата в регистр назначения. Если условие не выполняется, то нет операций. При отсутствии условия в команде выполняется безусловное отрицание. Исходный операнд хранится в регистре данных, который определен в команде.

Генерируемое состояние:

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	-	-	-	*	*	*	*

AZ Установлен, если результат равен 0. В других случаях, сброшен.
AN Установлен, если результат - отрицательное число. Иначе сброшен.
AV Установлен, если операнд равен H#8000. Иначе сброшен.
AC Установлен, если операнд равен нулю. Иначе сброшен.

Формат команды:
Выполняемая по условию операция АЛУ или умножителя-накопителя. Тип 9.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	Z	AMF					Yop		Xop			0	0	0	0	COND			

AMF определяет операцию АЛУ или умножителя-накопителя, в данном случае:
AMF = 10101 для операции типа: - yop.
AMF = 11001 для операции типа: - xop.
Обратите внимание, что операция типа: - xop является разновидностью операции: yop - xop, когда yop = 0.

Z:	Регистр назначения	Y:	Операнд Y
Xop:	Операнд X	COND:	Условие

15 Команды АЛУ
Команда NOT

Синтаксис: [IF условие] | AR / AF | = NOT | xop / yop | ;

Разрешенные xop	*Разрешенные yop*	*Разрешенные условия*		
AX0	MR2	AY0	EQ	LE	AC
AX1	MR1	AY1	NE	NEG	NOT AC
AR	MR0	AF	GT	POS	MV
	SR1	0	GE	AV	NOT MV
	SR0		LT	NOT AV	NOT CE

Пример: IF NE AF = NOT AX0;

Описание: Проверка необязательно присутствующего условия и, если оно истинно, выполнение поразрядного инвертирования исходного операнда и запись результата в регистр назначения. Если условие не выполняется, то нет операций. При отсутствии условия в команде выполняется безусловная операция инвертирования. Исходный операнд хранится в регистре данных, который определен в команде.

Генерируемое состояние:

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	-	-	-	0	0	*	*

AZ Установлен, если результат равен 0. В других случаях сброшен.
AN Установлен, если результат - отрицательное число. Иначе сброшен.
AV Всегда сброшен.
AC Всегда сброшен.

Формат команды:
Выполняемая по условию операция АЛУ или умножителя-накопителя. Тип 9.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	Z	AMF					Yop		Xop			0	0	0	0	COND			

AMF определяет операцию АЛУ или умножителя-накопителя, в данном случае:
AMF = 10100 для операции типа: NOT yop.
AMF = 11011 для операции типа: NOT xop.

Z: Регистр назначения Yop: Операнд Y
Xop: Операнд X COND: Условие

Команды АЛУ 15
Нахождение абсолютного значения

Синтаксис: [IF условие] | AR / AF | = ABS хор ;

Разрешенные хор		*Разрешенные условия*		
AX0	MR2	EQ	LE	AC
AX1	MR1	NE	NEG	NOT AC
AR	MR0	GT	POS	MV
	SR1	GE	AV	NOT MV
	SR0	LT	NOT AV	NOT CE

Пример: IF NEG AF = ABS AX0;

Описание: Проверка необязательно присутствующего условия и, если оно истинно, абсолютное значение исходного операнда записывается в регистр назначения. Если условие не выполняется, то нет операций. При отсутствии условия в команде операция нахождения абсолютного значения числа выполняется без условия. Исходный операнд хранится в регистре данных, который определен в команде.

Генерируемое состояние:

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	-	-	*	0	*	*	*

AZ Установлен, если результат равен 0. В других случаях, сброшен.
AN Установлен, если хор равен H#8000. Иначе сброшен.
AV Установлен, если хор равен H#8000. Иначе сброшен.
AC Всегда сброшен.
AS Установлен, если исходный операнд - отрицательное число. Иначе сброшен.

Формат команды:
Выполняемая по условию операция АЛУ или умножителя-накопителя. Тип 9.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	Z	AMF					0	0	Xop			0	0	0	0	COND			

AMF определяет операцию АЛУ или умножителя-накопителя, в данном случае:
AMF = 11111 для операции типа: ABS хор

Z: Регистр назначения COND: Условие
Хор: Операнд X

15 Команды АЛУ
Инкремент

Синтаксис: [IF условие] | AR / AF | = yop + 1;

Разрешенные yop *Разрешенные условия*

AY0	EQ	LE	AC
AY1	NE	NEG	NOT AC
AF	GT	POS	MV
	GE	AV	NOT MV
	LT	NOT AV	NOT CE

Пример: IF GT AF = AF + 1;

Описание: Проверка необязательно присутствующего условия и, если оно истинно, инкрементирование исходного операнда на H#0001 и запись результата в регистр назначения. Если условие не выполняется, то нет операций. При отсутствии условия в команде выполняется безусловная операция инкрементирования. Исходный операнд хранится в регистре данных, который определен в команде.

Генерируемое состояние:

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	-	-	-	*	*	*	*

AZ Установлен, если результат равен 0. В других случаях, сброшен.
AN Установлен, если результат - отрицательное число. Иначе сброшен.
AV Установлен, если генерируется переполнение. Иначе сброшен.
AC Установлен, если генерируется перенос. Иначе сброшен.

Формат команды:
Выполняемая по условию операция АЛУ или умножителя-накопителя. Тип 9.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	Z	AMF					Yop		Xop			0	0	0	0	COND			

AMF определяет операцию АЛУ или умножителя-накопителя, в данном случае:
AMF = 10001 для операции типа: yop + 1
Обратите внимание, что операция инкрементирования не производится над xop.

Z: Регистр назначения Y: Операнд Y
Xop: Операнд X COND: Условие

Синтаксис: [IF условие] | AR / AF | = yop - 1;

Разрешенные yop *Разрешенные условия*

AY0	EQ	LE	AC
AY1	NE	NEG	NOT AC
AF	GT	POS	MV
	GE	AV	NOT MV
	LT	NOT AV	NOT CE

Пример: IF EQ AR = AY1 - 1;

Описание: Проверка необязательно присутствующего условия и, если оно истинно, декрементирование исходного операнда на H#0001 и запись результата в регистр назначения. Если условие не выполняется, то нет операций. При отсутствии условия в команде выполняется безусловная операция декрементирования. Исходный операнд хранится в регистре данных, который определен в команде.

Генерируемое состояние:

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	-	-	-	*	*	*	*

AZ Установлен, если результат равен 0. В других случаях, сброшен.
AN Установлен, если результат - отрицательное число. Иначе сброшен.
AV Установлен, если генерируется переполнение. Иначе сброшен.
AC Установлен, если генерируется перенос. Иначе сброшен.

Формат команды:
Выполняемая по условию операция АЛУ или умножителя-накопителя. Тип 9.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	Z	AMF					Yop		Xop			0	0	0	0	COND			

AMF определяет операцию АЛУ или умножителя-накопителя, в данном случае:
AMF = 11000 для операции типа: yop - 1
Обратите внимание, что операция декрементирования не производится над xop.

Z:	Регистр назначения	Y:	Операнд Y
Xop:	Операнд X	COND:	Условие

15 Команды АЛУ
Деление

Синтаксис: DIVS yop, xop;
DIVQ xop;

Разрешенные xop		*Разрешенные yop*
AX0	MR2	AY1
AX1	MR1	AF
AR	MR0	
	SR1	
	SR0	

Описание: Эти команды выполняют вычисления типа: *yop/xop*. Имеется два примитива деления, DIVS и DIVQ. Операция деления с одинарной точностью, когда 32-разрядный числитель делится на 16-разрядный знаменатель и в результате получается 16-разрядное частное , выполняется за 16 циклов. Возможно деление с более высокой точностью.

Деление может быть либо знаковым либо беззнаковым, но числитель и знаменатель обязательно должны быть в одном и том же формате: знаковом или беззнаковом. При операции деления старшая половина бит числителя помещается в любой из разрешенных регистров *yop* (AY1 или AF), младшая половина бит числителя - в регистр AY0, а знаменатель - в любой разрешенный регистр *xop*. Затем выполняется операция деления с примитивам DIVS и DIVQ. При повторном выполнении операции DIVQ используется невосстанавливаемый условный алгоритм деления сложением-вычитанием. По окончанию операции деления частное будет находиться в регистре AY0.

Чтобы выполнить деление знаковых чисел, следует сначала вычислить знак частного, для чего один раз выполняется команда DIVS. Затем команда DIVQ выполняется столько раз, сколько бит остается в частном (т.е. для знакового деления с одинарной точностью DIVS выполняется один раз, а DIVQ - 15 раз).

Чтобы разделить беззнаковое число на беззнаковое, следует сначала поместить старшую половину бит числителя в регистр AF, а затем установить бит AQ равным нулю, непосредственно сбросив этот бит в регистре состояний арифметических устройств ASTAT, что служит указанием на положительный знак частного. Затем команда DIVQ выполняется столько раз, сколько бит имеется в частном (т.е. для беззнакового деления с одинарной точностью DIVQ выполняется 16 раз).

Бит частного, который генерируется каждый раз, когда выполняется команда DIVS или DIVQ, является битом AQ, который записывается в регистр ASTAT в конце каждого цикла. Последний остаток, полученный в результате

выполнения данного алгоритма (который остается в регистре AF), не является достоверным значением и должен при необходимости подвергаться исправлениям. См. исключения при делении в приложении к данному изданию.

Генерируемое состояние:

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	-	*	-	-	-	-	-

AQ Загружается значением равным биту AQ, который вычисляется в каждом цикле в результате выполнения команды DIVS или DIVQ.

Формат команды:

Команда DIVQ. Тип 23.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	1	0	0	0	1	0	Xop			0	0	0	0	0	0	0	0

Команда DIVS. Тип 24.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	0	0	0	0	Yop		Xop			0	0	0	0	0	0	0	0

Xop: Операнд X Y: Операнд Y

15 Команды умножителя-накопителя
Умножение с вычитанием

Синтаксис: NONE = <ALU>;

Примеры: NONE = AX0-AY0;
NONE = PASS SR0;

Описание: Выполнение указанной в команде операции АЛУ, генерирование флагов состояния АЛУ, а затем сброс полученного результата. Эта команда позволяет проводить тестирование содержащихся в регистрах АЛУ значений, не изменяя содержимого регистров AR или AF.

Обратите внимание, что операции АЛУ с добавлением постоянной (процессоры ADSP-217x, ADSP-218x, ADSp-21msp58/59) также разрешены.

Сложение *(xop + постоянная)*
Вычитание X-Y (*xop - постоянная)*
Вычитание Y-X *(-xop + постоянная)*
И, ИЛИ, исключающее ИЛИ *(xop•постоянная)*
PASS (*PASS постоянная, постоянная ≠0, -1,1)*
TSTBIT, SETBIT, CRLBIT, TGLBIT.

Генерируемое состояние:

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	-	-	-	*	*	*	*

AZ Установлен, если результат равен 0. В других случаях, сброшен.
AN Установлен, если результат - отрицательное число. Иначе сброшен.
AV Установлен, если генерируется переполнение. Иначе сброшен.
AC Установлен, если генерируется перенос. Иначе сброшен.

Формат команды:
Операция АЛУ или умножителя-накопителя с одновременной пересылкой данных в регистр.
Тип 8.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	0	AMF					Yop		Xop			1	0	1	0	1	0	1	0

только коды АЛУ

AMF определяет операцию АЛУ или умножителя-накопителя, в данном случае разрешены только операции АЛУ

Xop: Операнд X Yop: Операнд Y

Синтаксис: [IF условие] | MR / MF | = xop * | yop / xop | | (SS) / (SU) / (US) / (UU) / (RND) | ;

Разрешенные xop *Разрешенные yop* *Разрешенные условия (См. табл. 15.9)*

MX0	AR	MY0	EQ	LE	AC
MX1	SR1	MY1	NE	NEG	NOT AC
MR2	SR0	MF	GT	POS	MV
MR1			GE	AV	NOT MV
MR0			LT	NOT AV	NOT CE

Пример:
```
IF EQ MR = MX0 * MF (UU);
MF=SR0*SR0 (SS);
```

Описание: Проверка необязательно присутствующего условия и, если оно истинно, перемножение операндов и запись результата в регистр назначения. Если условие не выполняется, то нет операций. При отсутствии условия в команде операция умножения выполняется без условия. Операнды хранятся в регистрах данных, которые определены в команде. Когда MF является операндом назначения, в этом регистре хранятся только биты 31-16 произведения.

Операция типа *xop***xop* допустима только на процессорах ADSP-217x, ADSP-218x и ADSP-21msp58/59. Оба операнда *xop* должны быть одним регистром. Эта команда позволяет выполнять инструкции X^2 и $\sum X^2$ за один цикл.

Поле выбора формата данных, которое следует за двумя операндами, определяет, является ли соответствующий операнд знаковым (S) или беззнаковым (U). Сначала определяется *xop*, затем - *yop*. Формат данных не задается по умолчанию; один из форматов должен быть определен. Если за операндами следует RND (округление), умножитель-накопитель перемножает два операнда, округляет результат до 24 самых старших бит (или округляет биты 31-16 до 16 бит, если в результате умножения не происходит переполнения) и записывает результат в регистр назначения. Два операнда, *xop* и *yop*, рассматриваются как числа в формате с дополнительным кодом. Операция округления производится без смещения, за исключением процессоров ADSP-217x, ADSP-218x, ADSP-21msp58/59, в которых имеется режим умножения со смещением. См. описание округления без смещения в разделе "Умножитель-накопитель", подраздел "Режимы округления".

15 Команды умножителя-накопителя
Умножение с вычитанием

Генерируемое состояние:

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	*	-	-	-	-	-	-

MV Установлен в случае переполнения умножителя-накопителя (когда не все старшие 9 бит регистра MR равны нулю или единице). В других случаях сброшен.

Формат команды:

*(xop*yop)* Выполняемая по условию операция АЛУ или умножителя-накопителя. Тип 9.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	Z	AMF					Yop		Xop			0	0	0	0	COND			

AMF определяет операцию АЛУ или умножителя-накопителя, в данном случае:

AMF	*ФУНКЦИЯ*	*Формат данных*	*Операнд X*	*Операнд Y*
00100	xop*yop	(SS)	Знаковый	Знаковый
00101	xop*yop	(SU)	Знаковый	Беззнаковый
00110	xop*yop	(US)	Беззнаковый	Знаковый
00111	xop*yop	(UU)	Беззнаковый	Беззнаковый
00001	xop*yop	(RND)	Знаковый	Знаковый

Z:	Регистр назначения	Y:	Операнд Y
Xop:	Операнд X	COND:	Условие

*(xop*xop)* Выполняемая по условию операция АЛУ или умножителя-накопителя. Тип 9. (ADSP-217x, ADSP-218x, ADSP-21msp58/59)

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	Z	AMF					0	0	Xop			0	0	0	1	COND			

AMF определяет операцию АЛУ или умножителя-накопителя, в данном случае:

AMF	*ФУНКЦИЯ*	*Формат данных*	*Операнд X*
00100	xop*xop	(SS)	Знаковый
00111	xop*xop	(UU)	Беззнаковый
00001	xop*xop	(RND)	Знаковый

Z:	Регистр назначения		
Xop:	Операнд X	COND:	Условие

Умножение с вычитанием

Синтаксис:	[IF условие]	MR MF	= MR + xop *	yop xop	(SS) (SU) (US) (UU) (RND)	;

Разрешенные xop *Разрешенные yop* *Разрешенные условия (См. табл. 15.9)*

MX0	AR	MY0	EQ	LE	AC
MX1	SR1	MY1	NE	NEG	NOT AC
MR2	SR0	MF	GT	POS	MV
MR1			GE	AV	NOT MV
MR0			LT	NOT AV	NOT CE

Пример: IF GE MR = MR + MX0 * MY1 (SS); *xop*xop*
MR=MR+MX0*MX0 (SS); *xop*xop*

Описание: Проверка необязательно присутствующего условия и, если оно истинно, перемножение операндов, добавление произведения к значению, содержащемуся в регистре MR, и запись результата в регистр назначения. Если условие не выполняется, то нет операций. При отсутствии условия в команде операция умножения с накоплением выполняется без условия. Операнды хранятся в регистрах данных, которые определены в команде. Когда MF является операндом назначения, в этом регистре хранятся только биты 31-16 40-разрядного результата произведения.

Операция типа *xop*xop* допустима только на процессорах ADSP-217x, ADSP-218x и ADSP-21msp58/59. Оба операнда *xop* должны быть одним регистром. Эта команда позволяет выполнять инструкции X^2 и $\sum X^2$ за один цикл.

Поле выбора формата данных справа от двух операндов определяет, является ли соответствующий операнд знаковым (S) или беззнаковым (U). Сначала определяется формат *xop*, затем - формат *yop*. Формат данных не задается по умолчанию, а должен быть определен.

Если за операндами следует RND (округление), умножитель-накопитель перемножает два операнда, округляет результат до 24 самых старших бит (или округляет биты 31-16 до ближайших 16 бит, если в результате умножения с накоплением не происходит переполнения) и записывает результат в регистр назначения. Два операнда, *xop* и *yop*, рассматриваются как числа в формате с дополнительным кодом. Операция округления производится без смещения, за исключением процессроов ADSP-217x, ADSP-218x, ADSP-21msp58/59, в которых имеется режим округления со смещением.. См. описание округления без смещения в разделе "Умножитель-накопитель", подраздел "Режимы округления".

15 Команды умножителя-накопителя
Умножение с вычитанием

Генерируемое состояние:

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	*	-	-	-	-	-	-

MV Установлен в случае переполнения умножителя-накопителя (когда не все старшие 9 бит регистра MR равны нулю или единице). В других случаях, сброшен.

Формат команды:

*(xop*yop)* Выполняемая по условию операция АЛУ или умножителя-накопителя. Тип 9.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	Z	AMF					Yop		Xop			0	0	0	0	COND			

AMF определяет операцию АЛУ или умножителя-накопителя, в данном случае:

AMF	*ФУНКЦИЯ*	*Формат данных*	*Операнд X*	*Операнд Y*
01000	MR + xop*yop	(SS)	Знаковый	Знаковый
01001	MR + xop*yop	(SU)	Знаковый	Беззнаковый
01010	MR + xop*yop	(US)	Беззнаковый	Знаковый
01011	MR + xop*yop	(UU)	Беззнаковый	Беззнаковый
00010	MR + xop*yop	(RND)	Знаковый	Знаковый

Z: Регистр назначения Y: Операнд Y
Xop: Операнд X COND: Условие

*(xop*xop)* Выполняемая по условию операция АЛУ или умножителя-накопителя. Тип 9. (ADSP-217x, ADSP-218x, ADSP-21msp58/59)

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	Z	AMF					0	0	Xop			0	0	0	1	COND			

AMF определяет операцию АЛУ или умножителя-накопителя, в данном случае:

AMF	*ФУНКЦИЯ*	*Формат данных*	*Операнд X*
01000	MR+xop*xop	(SS)	Знаковый
01011	MR+xop*xop	(UU)	Беззнаковый
00010	MR+xop*xop	(RND)	Знаковый

Z: Регистр назначения

Умножение с вычитанием

Xop: Операнд X COND: Условие

Синтаксис:	[IF условие]	MR MF	= MR - xop *	yop xop	(SS) (SU) (US) (UU) (RND)	;

Разрешенные xop *Разрешенные yop* *Разрешенные условия (См. табл. 15.9)*

MX0	AR	MY0	EQ	LE	AC
MX1	SR1	MY1	NE	NEG	NOT AC
MR2	SR0	MF	GT	POS	MV
MR1			GE	AV	NOT MV
MR0			LT	NOT AV	NOT CE

Пример: IF LT MR = MR - MX1 * MY0 (SU); *xop*yop*
MR=MR-MX0*MX0 (SS); *xop*xop*

Описание: Проверка необязательного условия и, если оно истинно, перемножение операндов, вычитание полученного произведения из значения, содержащегося в регистре MR, и запись результата в регистр назначения. Если условие не выполняется, то нет операций. При отсутствии условия в команде, выполняется операция умножения с вычитанием. Операнды хранятся в регистрах данных, которые определены в команде. Когда MF является операндом назначения, в этом регистре хранятся только биты 31-16 40-разрядного результата произведения.

Операция типа *xop*xop* допустима только на процессорах ADSP-217x, ADSP-218x и ADSP-21msp58/59. Оба операнда *xop* должны быть одним регистром. Эта команда позволяет выполнять инструкции X^2 и $\sum X^2$ за один цикл.

Поле выбора формата данных, символ справа от двух операндов определяет, является ли соответствующий операнд знаковым (S) или беззнаковым (U). Сначала определяется формат *xop*, затем - формат *yop*. Формат данных не задается по умолчанию, а должен быть определен.

Если за операндами следует RND (округление), умножитель-накопитель перемножает два операнда, вычитает произведение из содержащегося в регистре MR значения и округляет результат до 24 самых старших бит (или округляет биты 31-16 до 16 бит, если в результате умножения с вычитанием не происходит переполнения) и записывает результат в регистр назначения. Два операнда, *xop* и *yop*, рассматриваются как числа в формате с дополнительным кодом. Операция округления производится без смещения, за исключением процессоров ADSP-217x, ADSP-218x, ADSP-21msp58/59, в которых имеется режим округления со смещением. См. описание округления без смещения в разделе "Умножитель-накопитель", подраздел "Режимы округления".

15 Команды умножителя-накопителя
Умножение с вычитанием

Генерируемое состояние:

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	*	-	-	-	-	-	-

MV Установлен в случае переполнения умножителя-накопителя (когда не все старшие 9 бит регистра MR равны нулю или единице). В других случаях, сброшен.

Формат команды:

*(xop*yop)* Выполняемая по условию операция АЛУ или умножителя-накопителя. Тип 9.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	Z	AMF					Yop		Xop			0	0	0	0	COND			

AMF определяет операцию АЛУ или умножителя-накопителя, в данном случае:

AMF	*ФУНКЦИЯ*	*Формат данных*	*Операнд X*	*Операнд Y*
01100	MR - xop*yop	(SS)	Знаковый	Знаковый
01101	MR - xop*yop	(SU)	Знаковый	Беззнаковый
01110	MR - xop*yop	(US)	Беззнаковый	Знаковый
01111	MR - xop*yop	(UU)	Беззнаковый	Беззнаковый
00011	MR - xop*yop	(RND)	Знаковый	Знаковый

Z: Регистр назначения
Y: Операнд Y
Xop: Операнд X
COND: Условие

*(xop*xop)* Выполняемая по условию операция АЛУ или умножителя-накопителя. Тип 9. (ADSP-217x, ADSP-218x, ADSP-21msp58/59)

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	Z	AMF					0	0	Xop			0	0	0	1	COND			

AMF определяет операцию АЛУ или умножителя-накопителя, в данном случае:

AMF	*ФУНКЦИЯ*	*Формат данных*	*Операнд X*
01100	MR-xop*xop	(SS)	Знаковый
01111	MR-xop*xop	(UU)	Беззнаковый
00011	MR-xop*xop	(RND)	Знаковый

Z: Регистр назначения
Xop: Операнд X
COND: Условие

Команды умножителя-накопителя 15
Сброс (Clear)

Синтаксис: [IF условие] | MR / MF | = 0 ;

Разрешенные условия (См. табл. 15.9)

EQ	NE	GT	GE	LT
LE	NEG	POS	AV	NOT AV
AC	NOT AC	MV	NOT MV	NOT CE

Пример: IF GT MR = 0;

Описание: Проверка необязательно присутствующего условия и, если оно истинно, запись в заданный регистр значения 0. Если условие не выполняется, то нет операций. При отсутствии условия содержимое регистра очищается без условия. Все 40 бит регистра MR или все 16 бит регистра MF становятся равными нулю.

Генерируемое состояние:

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	0	-	-	-	-	-	-

MV Всегда сброшен.

Формат команды:

Выполняемая по условию операция АЛУ или умножителя-накопителя. Тип 9.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	Z	AMF					1	1	0	0	0	0	0	0	0	COND			

AMF определяет операцию АЛУ или умножителя-накопителя, в данном случае:

AMF = 00100 для операции очистки содержимого регистра.

Обратите внимание, что данная команда представляет собой особый случай команды xop * yop, когда yop = 0.

Z: Регистр назначения COND: Условие

15 Команды умножителя-накопителя
Пересылка данных регистра MR

Синтаксис: [IF условие] | MR / MF | =MR [(RND)] ;

Разрешенные условия (См. табл. 15.9)

EQ	NE	GT	GE	LT
LE	NEG	POS	AV	NOT AV
AC	NOT AC	MV	NOT MV	NOT CE

Пример: IF EQ MF= MR (RND);

Описание: Проверка необязательно присутствующего условия и, если оно истинно, выполняется пересылка данных, содержащихся в регистре MR, согласно приводимому ниже описанию. Если условие не выполняется, то нет операций. При отсутствии условия в команде выполняется безусловная пересылка данных. В действительности, данная команда представляет собой особый случай операции умножения с накоплением, когда один из множителей, а именно *yop = 0,* и к содержимому регистра MR добавляется произведение равное нулю. Содержимое регистра MR может округляться по границе 15-ого и 16-ого битов произведения, для чего в команде следует поставить RND. Если в качестве регистра назначения определен регистр MF, то в этот регистр записываются биты 31-16 полученного произведения. Когда в качестве регистра назначения используется регистр MR, то весь 40-разрядный результат целиком содержится в этом регистре.

Генерируемое состояние:

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	*	-	-	-	-	-	-

MV Устанавливается в случае переполнения умножителя-накопителя (когда не все 9 старших бит регистра MR равны нулю или единице). В других случаях, сброшен.

Формат команды:
Выполняемая по условию операция АЛУ или умножителя-накопителя. Тип 9.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	Z	AMF					1	1	0	0	0	0	0	0	0	COND			

AMF определяет операцию АЛУ или умножителя-накопителя, в данном случае:

AMF = 01000 для операции пересылки данных.

Обратите внимание, что данная команда представляет собой особый случай команды типа: MR + xop*yop, когда yop = 0.

Z: Регистр назначения COND: Условие

Команды умножителя-накопителя 15
Насыщение регистра MR по условию

Синтаксис: IF MV SAT MR;

Описание: Проверка бита переполнения умножителя-накопителя (MV) в регистре состояния арифметических устройств, и в случае, когда этот бит установлен, насыщение 32 младших бит 40-разрядного регистра MR; если бит MV не установлен, то нет операций.

Операция насыщения содержимого регистра MR выполняется по этой команде в течение только одного цикла. Насыщение умножителя-накопителя не является постоянным режимом, который то разрешается, то блокируется. Команда насыщения предназначена для использования по завершению серии операций умножения с накоплением, так чтобы промежуточные переполнения не приводили к насыщению сумматора.

Результат операции насыщения зависит от состояния бита переполнения MV и от знака регистра MR (самого старшего бита регистра MR2). В таблице ниже приводятся все возможные результаты, получаемые после выполнения команды насыщения.

MV	*Самый старший бит MR2*	*Содержимое регистра MR после насыщения*
0	0	Не изменяется
0	1	Не изменяется
1	0	00000000 0111111111111111 1111111111111111
1	1	11111111 1000000000000000 0000000000000000

Генерируемое состояние: Не влияет на биты состояния

Формат команды:

Операция насыщения регистра MR. Тип 25.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15 Команды устройства сдвига
Арифметический сдвиг

Синтаксис: [IF условие] SR = [SR OR] ASHIFT хор | (HI) (LO) | ;

Разрешенные хор *Разрешенные условия*

SR	AR	EQ	LE	AC
SR1	MR2	NE	NEG	NOT AC
SR0	MR1	GT	POS	MV
	MR0	GE	AV	NOT MV
		LT	NOT AV	NOT CE

Пример: IF LT SR= SR OR ASHIFT SI(LO);

Описание: Проверка необязательно присутствующего условия и, если оно истинно, выполняется назначенный арифметический сдвиг. Если условие не выполняется, то нет операций. При отсутствии условия операция сдвига выполняется без условия. В данной операции осуществляется арифметический сдвиг битов операнда на число позиций и по направлению согласно коду сдвига в регистре SE. Положительные коды сдвига обозначают сдвиг влево (вверх), отрицательные коды - сдвиг вправо (вниз).

Сдвиг может производиться относительно старшей половины выходной группы разрядов (HI) или относительно младшей половины выходной группы разрядов (LO). Над результатом, полученным после операции сдвига, путем выбора опции SR OR может производиться операция логического ИЛИ с текущим содержимым регистра SR.

По команде ASHIFT с положительным кодом (т.е. когда в регистре SE содержится положительное значение) биты операнда сдвигаются влево; по той же команде с отрицательным кодом (т.е. когда в SE содержится отрицательное значение) биты операнда сдвигаются вправо. Число позиций сдвига определяется значением в коде сдвига. Группа из 32 разрядов дополняется по знаку влево (самый старший бит входного значения дублируется влево), а правые биты группы заполняются нулями. Биты, сдвигаемые влево от старшего бита (SR_{31}) в 32-разрядном поле назначения, опускаются. Биты, сдвигаемые вправо от самого младшего бита (SR_0) в 32-разрядном поле назначения, также опускаются.

Для сдвига числа с двойной точностью для обеих половинок такого числа используется тот же код сдвига. В первом цикле, старшие биты числа сдвигаются по команде ASHIFT относительно старших бит выходной группы разрядов (опция HI); в следующем цикле по команде LSHIFT сдвигаются младшие биты числа и при этом используются опции LO и OR. Это предотвращает дополнение по знаку младшего слова числа.

Генерируемое состояние: Не влияет на биты состояния.

Формат команды:
Выполняемая по условию операция сдвига. Тип 16.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	0	0	SF				Xop			0	0	0	0	COND			

SF	*Функция сдвига*
0100	ASHIFT (HI)
0101	ASHIFT (HI, OR)
0110	ASHIFT (LO)
0111	ASHIFT (LO, OR)

Xop: Операнд устройства сдвига COND: Условие

15 Команды устройства сдвига
Логический сдвиг

Синтаксис: [IF условие] SR = [SR OR] LSHIFT хор | (HI) (LO) | ;

Разрешенные хор *Разрешенные условия*

SR	AR	EQ	LE	AC
SR1	MR2	NE	NEG	NOT AC
SR0	MR1	GT	POS	MV
	MR0	GE	AV	NOT MV
		LT	NOT AV	NOT CE

Пример: IF GE SR= LSHIFT SI(HI);

Описание: Проверка необязательно присутствующего условия и, если оно истинно, выполняется назначенный логический сдвиг. Если условие не выполняется, то нет операций. При отсутствии условия в команде операция сдвига выполняется без условия. В данной операции осуществляется логический сдвиг битов операнда на число позиций и по направлению согласно коду сдвига, который определяется регистром SE. Положительные коды сдвига обозначают сдвиг влево (вверх), отрицательные коды - сдвиг вправо (вниз).

Сдвиг может производиться относительно старшей половины выходной группы разрядов (HI) или относительно младшей половины выходной группы разрядов (LO). Над результатом, полученным после операции сдвига, путем выбора опции SR OR может производиться операция логического ИЛИ с текущим содержимым регистра SR.

По команде LSHIFT с положительным кодом биты операнда сдвигаются влево на число позиций, определяемое значением в регистре SE. 32-разрядная выходная группа справа заполняется нулями. Биты, сдвигаемые влево от старшего бита (SR_{31}) в 32-разрядном поле назначения, опускаются.

По команде LSHIFT с отрицательным кодом биты операнда сдвигаются вправо на число позиций, определяемое значением в регистре SE. 32-разрядная выходная группа слева заполняется нулями. Биты, сдвигаемые вправо от самого младшего бита (SR_0) в 32-разрядном поле назначения, также опускаются.

Для сдвига числа с двойной точностью для обеих половинок такого числа используется тот же код сдвига. В первом цикле, старшие биты числа сдвигаются относительно старших бит выходной группы разрядов (HI); в следующем цикле сдвигаются младшие биты числа и при этом используются опции LO и OR.

Генерируемое состояние: Не влияет на биты состояния.

Команды устройства сдвига 15
Логический сдвиг

Формат команды:
Выполняемая по условию операция сдвига. Тип 16.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	0	0	SF				Xop			0	0	0	0	COND			

SF	*Функция сдвига*
0000	LSHIFT (HI)
0001	LSHIFT (HI, OR)
0010	LSHIFT (LO)
0011	LSHIFT (LO, OR)

Xop: Операнд устройства сдвига COND: Условие

15 Команды устройства сдвига
Нормализация

Синтаксис: [IF условие] SR = [SR OR] NORM хор | (HI) (LO) | ;

Разрешенные хор *Разрешенные условия*

SR	AR	EQ	LE	AC
SR1	MR2	NE	NEG	NOT AC
SR0	MR1	GT	POS	MV
	MR0	GE	AV	NOT MV
		LT	NOT AV	NOT CE

Пример: SR= NORM SI(HI);

Описание: Проверка необязательно присутствующего условия и, если оно истинно, выполняется операция нормализации. Если условие не выполняется, то нет операций. При отсутствии условия в команде операция нормализации выполняется без условия. В данной операции осуществляется арифметический сдвиг битов входного операнда таким образом, чтобы в нем остался только один знаковый бит. Число позиций сдвига берется из регистра SE. В регистр SE используя команду нахождения порядка может загружаться соответствующий код сдвига, который позволяет удалить излишние знаковые биты; загруженный код сдвига будет иметь отрицательный знак и равняться числу знаковых бит минус единица.

Сдвиг может производиться относительно старшей половины выходной группы разрядов (HI) или относительно младшей половины выходной группы разрядов (LO). Над результатом, полученным после операции сдвига, путем выбора опции SR OR может производиться операция логического ИЛИ с текущим содержимым регистра SR. Когда выбирается сдвиг относительно младшей половины разрядов (LO), 32-разрядное выходное поле заполняется слева нулями. Биты, сдвигаемые левее старшего бита (SR_{31}) 32-разрядного поля опускаются.

32-разрядное выходное поле справа заполняется нулями. Если порядок результата переполненного АЛУ был определен с помощью модификатора HIX, то 32-х разрядное выходное поле при выполнении команды NORM(HI) дополняется слева битом переноса АЛУ (AC) из регистра состояний арифметических устройств (ASTAT). При этом (т.к. в результате нахождения порядка результата АЛУ в случае его переполнения SE=1) происходит сдвиг вправо.

Для нормализации числа с двойной точностью для обеих половинок такого числа используется тот же код сдвига. В первом цикле, старшие биты числа сдвигаются относительно старших бит выходной группы разрядов (HI); в следующем цикле сдвигаются младшие биты числа и при этом используются опции LO и OR.

Генерируемое состояние: Не влияет на биты состояния.

Формат команды:
Выполняемая по условию операция сдвига. Тип 16.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	0	0	SF				Xop			0	0	0	0	COND			

SF	*Функция сдвига*
1000	NORM (HI)
1001	NORM (HI, OR)
1010	NORM (LO)
1011	NORM (LO, OR)

Xop: Операнд устройства сдвига COND: Условие

15 Команды устройства сдвига
Нахождение порядка

Синтаксис: [IF условие] SE = EXP хор | (HI) (LO) (HIX) | ;

Разрешенные хор *Разрешенные условия*

SR	AR	EQ	LE	AC
SR1	MR2	NE	NEG	NOT AC
SR0	MR1	GT	POS	MV
	MR0	GE	AV	NOT MV
		LT	NOT AV	NOT CE

Пример: IF GT SE = EXP MR1(HI);

Описание: Проверка необязательно присутствующего условия и, если оно истинно, выполняется операция нахождения порядка. Если условие не выполняется, то нет операций. При отсутствии условия в команде порядок находится без условия.

Операция EXP находит порядок входного операнда, что необходимо для последующей операции нормализации (NORM). По команде EXP исходный операнд загружается в определитель порядка, который генерирует код сдвига из числа ведущих знаковых бит этого операнда. По завершению выполнения команды EXP полученный код сдвига записывается в регистр SE и используется в последующих операциях как порядок входного значения. Полученный код сдвига зависит от режима, использованного при нахождении порядка (HI, LO или HIX).

В режиме HI входное значение интерпретируется как знаковое число с одинарной точностью или как старшие биты знакового числа с двойной точностью. Определитель порядка считает число ведущих знаковых бит в исходном операнде и записывает полученный код сдвига в регистр SE. Код сдвига в данном случае будет равняться числу излишних знаковых бит в данном операнде, взятому с отрицательным знаком.

В режиме HIX входное значение интерпретируется как результат сложения или вычитания, который может быть переполнен. Операции сдвига и нормализации в режиме HIX используются при работе с результатами, полученными в различных операциях АЛУ. В режиме HIX проверяется бит переполнения АЛУ (AV) в регистре состояний арифметических устройств (ASTAT); если этот бит установлен, то порядок входного значения равен +1 (указывая на то, что перед операцией EXP произошло переполнение АЛУ), и в регистр SE записывается значение +1. Если бит AV не установлен, то работа в режиме HIX не отличается от работы в режиме HI.

В режиме LO входное значение интерпретируется как младшая половина бит числа с двойной точностью. При выполнении операции EXP над числом с двойной точностью сначала обрабатываются старшие биты этого числа в режиме HI или HIX, после чего операция нахождения порядка производится над младшими битами этого числа в режиме LO. Если в старших битах числа содержался беззнаковый бит, то правильный код сдвига был сгенерирован в режиме HI или HIX и записан в регистр SE. Однако, если все старшие биты числа с двойной точностью были знаковыми, то в регистр SE записывается результат, полученный при подсчете числа ведущих знаковых бит этого числа в режиме LO.

Генерируемое состояние:

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	*	-	-	-	-	-	-	-

SS Равно самому старшему биту входного значения для операции нахождения порядка в режиме HI или HIX, когда AV = 0. Равно инвертированному самому старшему биту в режиме HIX, когда AV = 1. Не затрагивается в режиме LO.

Формат команды:

Выполняемая по условию операция сдвига. Тип 16.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	0	0	SF				Xop			0	0	0	0	COND			

SF	*Функция сдвига*
1100	EXP (HI)
1101	EXP (HIX)
1110	EXP (LO)

Xop: Операнд устройства сдвига COND: Условие

15 Команды устройства сдвига
Нахождение блочного порядка

Синтаксис: [IF условие] SB = EXPADJ хор ;

Разрешенные хор *Разрешенные условия*

SR	AR	EQ	LE	AC
SR1	MR2	NE	NEG	NOT AC
SR0	MR1	GT	POS	MV
	MR0	GE	AV	NOT MV
		LT	NOT AV	NOT CE

Пример: IF GT SB= EXPADJ SI;

Описание: Проверка необязательно присутствующего условия и, если оно истинно, выполняется назначенная операция нахождения порядка. Если условие не выполняется, то нет операций. При отсутствии условия в команде операция нахождения порядка выполняется без условия. Когда операция нахождения блочного порядка выполняется над рядом чисел, то в результате находится порядок наибольшего по модулю числа. Затем полученный порядок может связываться со всеми числами в представлении с блочной плавающей точкой.

Для нахождения блочного порядка входной операнд, который должен быть знаковым числом в дополнительном коде, загружается в определитель порядка. Определитель порядка работает в данном случае в режиме HI (см. описание команды EXP выше).

В начале блока чисел регистр SB должен быть установлен в исходное состояние и содержать минимальное возможное значение -16. Каждый раз после выполнения команды EXPADJ найденный в ней порядок операнда сравнивается с текущим содержимым регистра SB. Если новое значение порядка превышает текущее значение в регистре SB, содержимое этого регистра обновляется. Таким образом, в конце в регистре SB окажется максимальное найденное значение порядка. EXPADJ является только операцией проверки; в ходе ее выполнения не производится никаких сдвигов, так как истинное значение порядка остается неизвестным до окончания проверки всех чисел в блоке Однако, эти числа могут сдвигаться позднее после нахождения истинного значения порядка.

Дополненные (переполненные) числа и младшие биты чисел с двойной точностью не могут быть обработаны при помощи команды нахождения блочного порядка.

Генерируемое состояние: Не влияет на биты состояния.

Формат команды:
Выполняемая по условию операция сдвига. Тип 16.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	0	0	SF				Xop			0	0	0	0	COND			

SF = 1111

Xop: Операнд устройства сдвига COND: Условие

15 Команды устройства сдвига
Непосредственный арифметический сдвиг

Синтаксис: SR = [SR OR] ASHIFT xop BY <exp> | (HI) (LO) | ;

Разрешенные хор *<exp>*

SR	MR0	Любая постоянная от - 128 до 127*
SR1	MR1	
SR0	MR2	
AR		

Пример: SR= SR OR ASHIFT SR0 BY 3 (LO);

Описание: Арифметический сдвиг битов операнда на число позиций и по направлению, которые определяются постоянной в поле порядка (<exp>). Положительные постоянные вызывают сдвиг влево (вверх), отрицательные - вправо (вниз). Перед положительной постоянной **не должно** стоять знака "+".

Сдвиг может производиться относительно старшей половины выходной группы разрядов (HI) или относительно младшей половины выходной группы разрядов (LO). Над результатом, полученным после операции сдвига, путем выбора опции SR OR, может производиться операция логического ИЛИ с текущим содержимым регистра SR.

По команде ASHIFT с положительным кодом биты операнда сдвигаются влево; по той же команде с отрицательным кодом биты операнда сдвигаются вправо. 32-разрядная выходная группа дополняется влево по знаку (самый старший бит входного значения дублируется влево), а вправо - нулями. Биты, сдвигаемые влево от старшего бита (SR_{31}) в 32-разрядном поле назначения, опускаются. Биты, сдвигаемые вправо от самого младшего бита (SR_0) в 32-разрядном поле назначения, также опускаются.

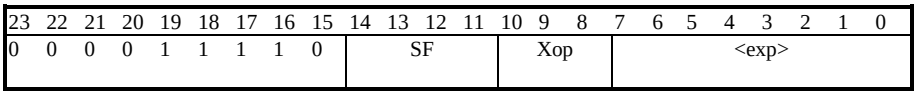
Для сдвига числа с двойной точностью для обеих половинок такого числа используется один и тот же код сдвига. В первом цикле старшие биты числа сдвигаются по команде ASHIFT относительно старшей половины выходной группы разрядов (HI); в следующем цикле по команде LSHIFT сдвигаются младшие биты числа и при этом используются опции LO и OR. Это предотвращает дополнение по знаку младшего слова числа.

* См. табл. 2.4 в главе 2

Команды устройства сдвига 15
Непосредственный арифметический сдвиг

Генерируемое состояние: Не влияет на биты состояния.

Операция непосредственного сдвига. Тип 15.



SF	*Функция сдвига*
0100	ASHIFT (HI)
0101	ASHIFT (HI, OR)
0110	ASHIFT (LO)
0111	ASHIFT (LO, OR)

Xop: Операнд устройства сдвига <exp>: знаковое значение сдвига разрядностью 8 бит

15 Команды устройства сдвига
Непосредственный логический сдвиг

Синтаксис: SR = [SR OR] LSHIFT xop BY <exp> | (HI) (LO) | ;

Разрешенные хор		*<exp>*
SR	MR0	Любая постоянная от - 128 до 127*
SR1	MR1	
SR0	MR2	
AR		

Пример: SR= LSHIFT SR1 BY -6 (HI);

Описание: Логический сдвиг битов операнда на число позиций и по направлению, которые определяются постоянной в поле порядка (<exp>). Положительные постоянные вызывают сдвиг влево (вверх), отрицательные - вправо (вниз). Перед положительной постоянной **не должно** стоять знака "+".

Сдвиг может производиться относительно старшей половины выходной группы разрядов (HI) или относительно младшей половины выходной группы разрядов (LO). Над результатом, полученным после операции сдвига, путем выбора опции SR OR, может производиться операция логического ИЛИ с текущим содержимым регистра SR.

По команде LSHIFT с положительным кодом биты операнда сдвигаются влево. 32-разрядное выходное поле слева и справа заполняется нулями. Биты, сдвигаемые влево от старшего бита (SR_{31}) в 32-разрядном поле назначения, опускаются. По команде LSHIFT с отрицательным кодом биты операнда сдвигаются вправо. 32-разрядное выходное поле слева и справа заполняется нулями. Биты, сдвигаемые вправо от самого младшего бита (SR_0) в 32-разрядном поле назначения, также опускаются.

Для сдвига числа с двойной точностью для обеих половинок такого числа используется один и тот же код сдвига. В первом цикле, старшие биты числа сдвигаются относительно старших бит выходной группы разрядов (HI); в следующем цикле по команде LSHIFT сдвигаются младшие биты числа и при этом используются опции LO и OR.

---

* См. табл. 2.4 в главе 2

Генерируемое состояние: Не влияет на биты состояния.

Формат команды:
Операция непосредственного сдвига. Тип 15.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	0	SF				Xop			<exp>							

SF	*Функция сдвига*
0000	LSHIFT (HI)
0001	LSHIFT (HI, OR)
0010	LSHIFT (LO)
0011	LSHIFT (LO, OR)

Xop: Операнд устройства сдвига <exp>: Знаковое значение сдвига разрядностью 8 бит

15 Команды пересылки данных

Пересылка данных между регистрами

Синтаксис: reg = reg ;

Разрешенные регистры

AX0	MX0	SI	SB	CNTR
AX1	MX1	SE	PX	OWRCNTR *(только для записи)*
AY0	MY0	SR1	ASTAT	RX0
AY1	MY1	SR0	MSTAT	RX1
AR	MR2	I0-I7	SSTAT *(только для чтения)*	TX0
	MR1	M0-M7	IMASK	TX1
	MR0	L0-L7	ICNTL	IFC *(только для записи)*

Пример: I7= AR;

Описание: Пересылка содержимого исходного регистра по указанному назначению. Содержимое исходного регистра всегда выравнивается в регистре назначения вправо после пересылки. При пересылке значения из регистра с меньшей разрядностью в регистр с большей разрядностью (например, из 8-разрядного регистра в 16-разрядный) значение в регистре назначения либо дополняется по знаку влево, если в исходном регистре было знаковое число, либо, если в исходном регистре было беззнаковое число, пространство слева от него заполняется нулями. Такими исходными регистрами, из которых может пересылаться беззнаковое число, при помещении которого в регистр назначения происходит заполнение содержимого последнего нулями, являются: регистры I0-I7, L0-L7, CNTR, PX, ASTAT, MSTAT, SSTAT, IMASK, ICNTL. При пересылке данных из всех остальных регистров происходит дополнение по знаку влево значения в регистре назначения.

При пересылке данных из регистра с большей разрядностью в регистр с меньшей разрядностью (например, из 16-разрядного регистра в 14-разрядный) значение в регистре назначения выравнивается вправо (бит 0 отражается в бит 0), а самые старшие биты теряются. Обратите внимание, что при загрузке данных в регистр MR1 он всегда дополняется по знаку в регистре MR2.

Генерируемое состояние: Не влияет на биты состояния.

Формат команды:
Операция внутренней пересылки данных. Тип 17.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	1	0	0	0	0	DST RGP		SRC RGP		DEST REG				SOURCE REG			

Группа исходных регистров (SRC RGP) и исходный регистр (SOURCE REG) выбирают исходный регистр по Таблице выбора регистров (см. Приложение A).

Группа регистров назначения (DST RGP) и регистр назначения (DEST REG) выбирают регистр назначения по Таблице выбора регистров (см. Приложение A).

15 Команды пересылки данных
Непосредственная загрузка регистра

Синтаксис: reg = <data>;

dreg = <data>;

data: <constant>
'%'<symbol>
'^'<symbol>

Разрешенные регистры

dreg (Тип команды 6) (загрузка 16 бит)			*regs (тип команды 7) (максимальная загрузка 14 бит)*	
AX0	MX0	SI	SB	CNTR
AX1	MX1	SE	PX	OWRCNTR *(только для записи)*
AY0	MY0	SR1	ASTAT	RX0
AY1	MY1	SR0	MSTAT	RX1
AR	MR2		IMASK	TX0
	MR1		ICNTL	TX1
	MR0		I0-I7	IFC *(только для записи)*
			M0-M7	
			L0-L7	

Пример: I0 = ^*data_buffer;*
L0 = %*data_buffer;*

Описание: Перемещение определенного значения данных по назначенному адресу. Данные могут представлять собой постоянную или любой символ, сопровождаемый операторами "длины" (%) и "указателя на" (^). Значение данных, содержащееся в командном слове имеет разрядность 16 бит , если оно загружается в регистр данных, и до 14 бит, если оно загружается в регистр другого типа. Значение всегда выравнивается по правому краю в регистре назначения после загрузки (бит 0 отображается в бит 0). При пересылке значения, длина которого меньше чем длина регистра назначения, это пересылаемое значение дополняется по знаку влево, чтобы заполнить все разряды в регистре назначения.

Обратите внимание, что всегда при загрузке данных в регистр MR1, он дополняется по знаку в регистр MR2.

Этой командой в регистры RX и TX может загружаться максимум 14 бит (несмотря на то, что сами эти регистры имеют разрядность 16 бит). Чтобы загрузить в эти регистры 16-разрядное значение, следует использовать команду пересылки данных между регистрами или команду пересылки данных из памяти в регистр с использованием прямой адресации.

Команды пересылки данных 15
Непосредственная загрузка регистра

Генерируемое состояние: Не влияет на биты состояния.

Формат команды:
Команда непосредственной загрузки данных в регистр. Тип 6:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	ДАННЫЕ																DREG			

Группа разрядов ДАННЫЕ содержит непосредственное значение, которое должно загружаться в регистр данных по адресу назначения. Эти данные выравниваются по правому краю таким образом, что значение, загружаемое в N-разрядный регистр назначения содержится в младших N битах поля ДАННЫЕ.

Группа разрядов DREG выбирает назначенный регистр данных для непосредственной загрузки. Согласно Таблице выбора регистров DREG (см. Приложение А) выбирается один из 16 регистров данных.

Команда непосредственной загрузки в регистр, не являющийся регистром данных. Тип 7:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	RGP		ДАННЫЕ														REG			

Группа разрядов ДАННЫЕ содержит непосредственное значение, которое будет загружаться в регистр по назначенному адресу. Данные выравниваются по правому краю, таким образом, значение, загружаемое в N-разрядный регистр назначения, содержится в N младших битах поля ДАННЫЕ.

Группы разрядов RGP (группа регистров) и REG (регистр) выбирают регистр назначения согласно Таблице выбора регистров (см. Приложение А).

15 Команды пересылки данных
Считывание из памяти данных (Прямая адресация)

Синтаксис: reg = DM (<addr>);

Разрешенные регистры

AX0	MX0	SI	SB	CNTR
AX1	MX1	SE	PX	OWRCNTR *(только для записи)*
AY0	MY0	SR1	ASTAT	RX0
AY1	MY1	SR0	MSTAT	RX1
AR	MR2	I0-I7		TX0
	MR1	M0-M7	IMASK	TX1
	MR0	L0-L7	ICNTL	IFC *(только для записи)*

Пример: SI = DM(ad_port0);

Описание: По команде считывания содержимое ячейки памяти перемещается в регистр назначения. Режим адресации - прямая (непосредственная) адресация (назначается непосредственным значением адреса или по метке). Адрес памяти данных содержится непосредственно в командном слове в виде полной группы из четырех разрядов. Содержимое исходной ячейки памяти всегда выравнивается по правому краю в регистре назначения после считывания (бит 0 отображается в бит 0).

Обратите внимание, что всегда при загрузке данных в регистр MR1, он дополняется по знаку в регистр MR2.

Генерируемое состояние: Не влияет на биты состояния.

Формат команды:
Команда считывания из памяти данных. Тип 3:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	RGP		АДРЕС														REG			

Группа разрядов АДРЕС содержит непосредственный адрес ячейки в памяти данных, из которой осуществляется считывание.

Группы разрядов RGP (группа регистров) и REG (регистр) выбирают регистр назначения согласно Таблице выбора регистров (см. Приложение А).

Команды пересылки данных 15
Считывание из памяти данных (косвенная адресация)

Синтаксис: dreg = DM(I0 , M0) ;
I1 M1
I2 M2
I3 M3

I4 M4
I5 M5
I6 M6
I7 M7

Разрешенные dreg

AX0	MX0	SI
AX1	MX1	SE
AY0	MY0	SR1
AY1	MY1	SR0
AR	MR2	
	MR1	
	MR0	

Пример: AY0 = DM(I3,M1);

Описание: По команде считывания из памяти данных с использованием косвенной адресации содержимое ячейки памяти данных перемещается в регистр назначения. Используется косвенный режим адресации с пост-модификацией адреса. **При косвенной адресации линейного (т.е. не циклического) буфера регистр L, соответствующий используемому регистру I, должен быть установлен равным нулю.** Содержимое исходной ячейки всегда выравнивается по правому краю в регистре назначения (бит 0 отображается в бит 0).

Генерируемое состояние: Не влияет на биты состояния.

Формат команды:

Команда АЛУ/умножителя-накопителя с одновременным считыванием из памяти данных. Тип 4:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	G	0	0	AMF					0	0	0	0	0	DREG				I		M	

AMF определяет операцию АЛУ или умножителя-накопителя, которая выполняется параллельно со считыванием из памяти данных. В данном случае, AMF = 00000, указывая на отсутствие операций АЛУ или умножителя-накопителя.

Группа разрядов DREG выбирает назначенный регистр данных. Согласно Таблице выбора регистров DREG (см. Приложение A) выбирается один из 16 регистров данных.

G определяет, регистры I и M какого генератора адреса данных (1 или 2) используются для данной операции. Оба регистра должны выбираться из одного и того же генератора адреса данных, черта в таблице выше отделяет регистры разных генераторов адреса данных друг от друга. I определяет указатель косвенного адреса (регистр I), группа разрядов M определяет регистр модификации (регистр M).

15 Команды пересылки данных
Считывание из памяти программы (косвенная адресация)

Синтаксис: dreg = PM(| I4 | I5 | I6 | I7 | , | M4 | M5 | M6 | M7 |) ;

Разрешенные dreg

AX0	MX0	SI
AX1	MX1	SE
AY0	MY0	SR1
AY1	MY1	SR0
AR	MR2	
	MR1	
	MR0	

Пример: MX1 = PM(I6,M5);

Описание: По команде считывания из памяти программы содержимое ячейки памяти программы перемещается в регистр назначения. Используется косвенный режим адресации с пост-модификацией адреса. **При косвенной адресации линейного (т.е. не циклического) буфера регистр L, соответствующий используемому регистру I, должен быть установлен равным нулю.** В регистр назначения загружаются 16 самых старших бит шины данных памяти программы ($ДПП_{23-8}$), таким образом что бит $ДПП_8$ становится битом 0 регистра назначения (правое выравнивание). Если разрядность регистра назначения меньше 16 бит, при загрузке опускаются самые старшие биты. Биты $ДПП_{7-0}$ всегда загружаются в регистр PX и могут игнорироваться или считываться в последующем цикле.

Генерируемое состояние: Не влияет на биты состояния.

Формат команды:
Команда АЛУ/умножителя-накопителя с одновременным считыванием из памяти программы.
Тип 5:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	0	AMF					0	0	0	0	0	DREG				I		M	

AMF определяет операцию АЛУ или умножителя-накопителя, которая выполняется параллельно со считыванием из памяти программы. В данном случае, AMF = 00000, указывая на отсутствие операций АЛУ или умножителя-накопителя.

Группа разрядов DREG выбирает назначенный регистр данных. Согласно Таблице выбора регистров (см. Приложение A) выбирается один из 16 регистров данных.

I определяет указатель косвенного адреса (регистр I), группа разрядов M определяет регистр модификации (регистр M).

Команды пересылки данных 15
Запись в память данных (прямая адресация)

Синтаксис: DM (<addr>) = reg;

Разрешенные регистры

AX0	MX0	SI	SB	CNTR
AX1	MX1	SE	PX	RX0
AY0	MY0	SR1	ASTAT	RX1
AY1	MY1	SR0	MSTAT	TX0
AR	MR2	I0-I7	SSTAT *(только для считывания)*	TX1
	MR1	M0-M7	IMASK	
	MR0	L0-L7	ICNTL	

Пример: DM(*cntl_port0)* = AR;

Описание: Перемещение содержимого исходного регистра в определенную в командном слове ячейку памяти данных. Используется непосредственный режим адресации (назначается непосредственное значение адреса или метка). Адрес ячейки в памяти данных содержится в самом командном слове в виде группы разрядов из 14 бит. При записи в память содержимого регистра, разрядность которого менее 16 бит, записываемое значение либо дополняется по знаку влево, если исходное значение имеет знаковый формат, либо пространство влево от него заполняется нулями, если это значение беззнаковое. Следующие регистры содержат беззнаковые значения, при записи которых в память пространство слева должно заполняться нулями: I0-I7, L0-L7, CNTR, PX, ASTAT, MSTAT, SSTAT, IMASK, CNTL. При записи в память содержимого всех других регистров происходит дополнение записываемого значения по знаку влево.

Содержимое исходного регистра всегда выравнивается по правому краю в назначенных ячейках памяти данных после записи (бит 0 отображается в бит 0). Обратите внимание, что всегда при загрузке данных в регистр MR1, он дополняется по знаку в регистр MR2.

Генерируемое состояние: Не влияет на биты состояния.

Формат команды:
Команда считывания из памяти данных. Тип 3:

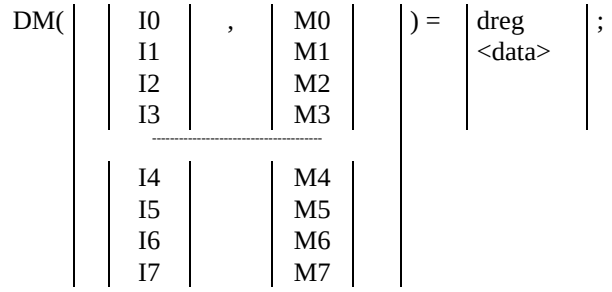
23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	RGP		АДРЕС														REG			

Группа разрядов АДРЕС содержит непосредственный адрес ячейки в памяти данных.

Группы разрядов RGP (группа регистров) и REG (регистр) выбирают исходный регистр согласно Таблице выбора регистров (см. Приложение А).

15 Команды пересылки данных
Запись в память данных (косвенная адресация)

Синтаксис:



data: <constant>
'%'<symbol>
'^'<symbol>

Разрешенные dreg

AX0	MX0	SI
AX1	MX1	SE
AY0	MY0	SR1
AY1	MY1	SR0
AR	MR2	
	MR1	
	MR0	

Пример: DM(I2,M0) = MR1;

Описание: По команде записи в память данных с использованием косвенной адресации содержимое исходного регистра пересылается в определенную в командном слове ячейку памяти данных. Непосредственные данные могут представлять собой постоянную или любой символ, сопровождаемый операторами "длины" (%) и "указатель на" (^).

Используется косвенный режим адресации с пост-модификацией адреса. **При косвенной адресации линейного (т.е. не циклического) буфера регистр L, соответствующий используемому регистру I, должен быть установлен равным нулю.** При записи в память содержимого регистра с разрядностью менее 16 бит записываемое значение дополняется по знаку до 16 бит. Содержимое исходного регистра всегда выравнивается по правому краю в ячейке назначения после записи (бит 0 отображается в бит 0).

Генерируемое состояние: Не влияет на биты состояния.

Команды пересылки данных 15
Запись в память данных (косвенная адресация)

Формат команды:
Команда АЛУ/умножителя-накопителя с одновременной записью в память данных. Тип 4:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	G	1	0	AMF					0	0	0	0	0	DREG				I		M	

Команда записи в память, непосредственные данные. Тип 2:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	G	ДАННЫЕ																I		M	

AMF определяет операцию АЛУ или умножителя-накопителя, которая выполняется параллельно с записью в память данных. В данном случае, AMF = 00000, указывая на отсутствие операций АЛУ или умножителя-накопителя.

Данные представляют собой действительное 16-разрядное значение.

Группа разрядов DREG выбирает исходный регистр данных. Согласно таблице выбора регистров (см. Приложение А) выбирается один из 16 регистров данных.

G определяет, регистры I и M какого генератора адреса данных (1 или 2) используются для данной операции. Оба регистра должны выбираться из одного и того же генератора адреса данных, черта в таблице выше отделяет регистры разных генераторов адреса данных друг от друга. I определяет указатель косвенного адреса (регистр I), группа разрядов M определяет регистр модификации (регистр M).

15 Команды пересылки данных
Запись в память программы (косвенная адресация)

Синтаксис: PM(I4 | I5 | I6 | I7 , M4 | M5 | M6 | M7) = dreg;

Разрешенные dreg

AX0	MX0	SI
AX1	MX1	SE
AY0	MY0	SR1
AY1	MY1	SR0
AR	MR2	
	MR1	
	MR0	

Пример: PM(I6,M5) = AR;

Описание: По команде записи в память программы с использованием косвенной адресации содержимое исходного регистра пересылается в определенную в командном слове ячейку памяти программы. Используется режим косвенной адресации с пост-модификацией адреса. **При косвенной адресации линейного (т.е. не циклического) буфера регистр L, соответствующий используемому регистру I, должен быть установлен равным нулю.** 16 самых старших бит шины данных памяти программы ($ДПП_{23-8}$) загружаются из исходного регистра так, что бит $ДПП_8$ загружается битом 0 исходного регистра (правое выравнивание). 8 самых младших бит шины данных памяти программы ($ДПП_{7-0}$) загружаются из регистра PX. При записи значения из регистра, длина которого менее 16 бит, записываемое значение дополняется по знаку до 16 бит.

Генерируемое состояние: Не влияет на биты состояния.

Формат команды:

Команда АЛУ/умножителя-накопителя с одновременной записью в память программы. Тип 5 (см. Приложение А):

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	1	0	AMF					0	0	0	0	0	DREG					I		M

AMF определяет операцию АЛУ или умножителя-накопителя, которая выполняется параллельно с записью в память программы. В данном случае, AMF = 00000, указывая на отсутствие операций АЛУ или умножителя-накопителя.

Группа разрядов DREG выбирает исходный регистр данных. Согласно таблице выбора регистров (см. Приложение А) выбирается один из 16 регистров данных.

I определяет указатель косвенного адреса (регистр I), группа разрядов M определяет регистр модификации (регистр M).

Команды пересылки данных 15
Считывание/запись в область ввода/вывода
(Только ADSP-218x)

Синтаксис: IO(<адрес> = dreg; *Запись в область ввода/вывода*
dreg = IO(<адрес>); *Считывание из области ввода/вывода*

Разрешенные dreg

AX0	MX0	SI
AX1	MX1	SE
AY0	MY0	SR1
AY1	MY1	SR0
AR	MR2	
	MR1	
	MR0	

Пример: IO(23) = AX0;
MY1 = IO(2047);

Описание: Команды считывания и записи даных в область ввода/вывода используются для получения доступа к области памяти ввода/вывода процессора ADSP-218x. При выполнении этих команд производятся пересылки данных между регистрами данных процессора и областью памяти ввода/вывода.
Генерируемое состояние: Не влияет на биты состояния.
Формат команды:
Команда считывания/записи в область ввода/вывода. Тип 29:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	D	АДРЕС												DREG		

АДРЕС содержит 11-разрядный адрес исходной ячейки или ячейки назначения памяти ввода/вывода.

Группа разрядов DREG выбирает регистр данных. Согласно таблице выбора регистров (см. Приложение A) выбирается один из 16 регистров данных.

D указывает на направление передачи (0 = считывание, 1 = запись)

15 Команды управления последовательностью выполнения программы

Переход (JUMP)

Синтаксис: [IF условие] JUMP | (I4) | (I5) | (I6) | (I7) | <addr> | ;

Разрешенные условия

EQ	NE	GT	GE	LT
LE	NEG	POS	AV	NOT AV
AC	NOT AC	MV	NOT AV	NOT CE

Пример: IF NOT CE JUMP *top_loop;* {CNTR декрементируется}

Описание: Проверка необязательно присутствующего условия и, если оно истинно, выполнение заданного перехода. Если условие не истинно, никаких операций не выполняется. При отсутствии условия в тексте команды выполняется безусловный переход. По команде JUMP выполнение программы продолжается по указанному в команде адресу. Могут использоваться прямой или косвенный с использованием регистра I режимы адресации.

При прямой (непосредственной) адресации (когда используется непосредственное значение адреса или метка) адрес программы содержится непосредственно в командном слове в виде полной группы разрядов из 14 бит. При переходе с косвенной аресацией адрес берется из выбранного регистра I, содержимое которого в данном случае не пост-модифицируется.

Если JUMP является последней командой в цикле DO UNTIL, следует убедиться в правильном обращении с содержимым стеков. Если в качестве условия используется IF NOT CE (счетчик не пуст), то при выполнении команды JUMP декрементируется значение счетчика (регистр CNTR).

Генерируемое состояние: Не влияет на биты состояния.

Формат команды:

Команда условного перехода с использованием прямой адресации. Тип 10:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	0	АДРЕС														COND			

Команда условного перехода с использованием косвенной адресации. Тип 19:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	I		0	0	COND			

Группа разрядов I определяет регистр I (указатель косвенного адреса).

АДРЕС: непосредственный адрес перехода COND: условие

Команды управления последовательностью выполнения программы 15
Вызов (CALL)

Синтаксис: [IF условие] CALL | (I4) (I5) (I6) (I7) <addr> | ;

Разрешенные условия

EQ	NE	GT	GE	LT
LE	NEG	POS	AV	NOT AV
AC	NOT AC	MV	NOT AV	NOT CE

Пример: IF AV CALL *scale_down;*

Описание: Проверка необязательно присутствующего условия и, если оно истинно, выполнение заданного вызова. Если условие не истинно, никаких операций не выполняется. При отсутствии условия в тексте команды выполняется безусловный вызов. Команда CALL предназначена для вызова подпрограмм. При выполнении этой команды адрес для возвращения в главную программу помещается в стек счетчика программ (PC), и выполняется программа по указанному в команде адресу. Для команды CALL может использоваться как прямая, так и косвенная адресация.

При прямой (непосредственной) адресации (когда используется непосредственное значение адреса или метка) адрес программы содержится непосредственно в командном слове в виде полной группы разрядов из 14 бит. При переходах с косвенной аресацией адрес берется из выбранного регистра I, содержимое которого, в данном случае, не пост-модифицируется.

Если CALL является последней командой в цикле DO UNTIL, следует убедиться в правильном обращении с содержимым стеков.

Генерируемое состояние: Не влияет на биты состояния.

Формат команды:

Команда условного перехода с использованием прямой адресации. Тип 10:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	1	АДРЕС														COND			

Команда условного перехода с использованием косвенной адресации. Тип 19:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	I		0	1	COND			

Группа разрядов I определяет регистр I (указатель косвенного адреса).

АДРЕС: непосредственный адрес перехода COND: условие

15 Команды управления последовательностью выполнения программы

Переход или вызов в соответствии с состоянием вывода "Flag In"

Синтаксис: IF | FLAG_IN / NOT FLAG_IN | | JUMP / CALL | | <addr> | ;

Пример: IF FLAG_IN JUMP *service_proc_three;*

Описание: Проверка состояния вывода FI процессора и , если на этом выводе единица, выполнение заданного перехода или вызова. Если FI равно нулю, никаких операций не выполняется. При отсутствии условия проверки состояния флага мы имеем дело с обыкновенной командой перехода или вызова.

По команде JUMP выполнение программы продолжается по указанному в команде адресу. Команда перехода в соответствии с состоянием вывода FI может использоваться только в режиме прямой адресации.

Команда CALL предназначена для вызова подпрограмм. По этой команде адрес возвращения в главную программу помещается в стек счетчика программ (PC), и выполнение программы продолжается по указанному в команде адресу. Команда вызова в соответствии с состоянием вывода FI может использовать только прямую адресацию.

Если JUMP или CALL является последней командой в цикле DO UNTIL, следует убедиться в правильном обращении с содержимым стеков.

В случае прямой адресации (когда используется непосредственное значение адреса или метка) адрес программы непосредственно содержится в командном слове в виде группы из 14 разрядов.

Генерируемое состояние: Не влияет на биты состояния.

Формат команды:
Команда условного перехода или вызова в соответствии с состоянием вывода FI с использованием прямой адресации. Тип 27:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	1	Адрес													Адрес	FIC	S

12 самых младших бит — 2 самых старших бита

S: определяет JUMP(0) или CALL(1) FIC: фиксированное состояние вывода FI

Команды управления последовательностью выполнения программы 15

Управление состоянием вывода "Flag Out"

Синтаксис:	[IF условие]	SET		FLAG_OUT	[,...];
		RESET		FL0	
		TOGGLE		FL1	
				FL2	

Пример: IF MV SET FLAG_OUT, RESET FL1;

Описание: Оценка необязательно присутствующего условия и, если оно истинно, на заданном(ых) выводе(ах) флага устанавливается единица, они перезапускаются (устанавливается ноль) или работают как тумблеры. Если условие не истинно, по этой команде не выполняется никаких операций и выполнение программы продолжается со следующей команды. При отсутствии условия в тексте команды данная операция осуществляется без условия. Модификация состояний нескольких флагов может осуществляться за счет включения в одну команду нескольких операторов, отделяемых друг от друга запятыми. Эта команда не влияет непосредственно на последовательность выполнения программы - но предназначена для подачи сигналов внешним устройствам.

Обратите внимание, что в синтаксисе данной команды вывод FO описывается через "FLAG_OUT".

В следующей таблице показано, какие выводы флагов имеют различные процессоры семейства ADSP-2100.

Процессор	*Вывод(ы) флагов*
ADSP-2101	FO
ADSP-2105	FO
ADSP-2115	FO
ADSP-2111	FO, FL0, FL1, FL2
ADSP-217x	FO, FL0, FL1, FL2
ADSP-218x	FO, FL0, FL1, FL2
ADSP-21msp58/59	FO, FL0, FL1, FL2

Генерируемое состояние: Не влияет на биты состояния.

Формат команды:

Команда управления выводами флагов. Тип 28:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	0	0	0	0	FO		FO		FO		FO		COND			
												/		/		/		/					
												FL2		FL1		FL0		FLAG_OUT					

FO: операция, выполняемая над выходом флага COND: условие

15 Команды управления последовательностью выполнения программы

Возвращение из подпрограммы (RTS)

Синтаксис: [IF условие] RTS ;

Разрешенные условия

EQ	NE	GT	GE	LT
LE	NEG	POS	AV	NOT AV
AC	NOT AC	MV	NOT AV	NOT CE

Пример: IF LE RTS;

Описание: Проверка необязательно присутствующего условия и, если оно истинно, выполнение заданного возвращения. Если условие не истинно, никаких операций не выполняется. При отсутствии условия в тексте команды выполняется безусловный возврат в главную программу. По команде RTS происходит возвращение в главную программу из подпрограммы. Из стека счетчика программ (PC) выталкивается самый верхний адрес, который и используется как адрес, по которому осуществляется возврат. Содержимое из других стеков не извлекается.

Если RTS является последней командой в цикле DO UNTIL, следует убедиться в правильном обращении с содержимым стеков циклов.

Генерируемое состояние: Не влияет на биты состояния.

Формат команды:
Команда возврата по условию. Тип 20:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	COND			

COND: условие

Команды управления последовательностью выполнения программы 15

Возвращение из подпрограммы обслуживания прерывания (RTI)

Синтаксис: [IF условие] RTI ;

Разрешенные условия

EQ	NE	GT	GE	LT
LE	NEG	POS	AV	NOT AV
AC	NOT AC	MV	NOT AV	NOT CE

Пример: IF MV RTI;

Описание: Проверка необязательно присутствующего условия и, если оно истинно, выполнение заданного возвращения. Если условие не истинно, никаких операций не выполняется. При отсутствии условия в тексте команды выполняется безусловное возвращение в главную программу. По команде RTI происходит возвращение в главную программу из подпрограммы обслуживания прерывания. Самый верхний адрес в стеке счетчика программ (PC) выталкивается и используется в качестве адреса возвращения в главную программу. При этом значение, находящееся на вершине стека состояний, также извлекается и затем загружается в регистры состояния арифметических устройств (ASTAT), состояния режима (MSTAT) и маскирования прерываний (IMASK).

Если RTI является последней командой в цикле DO UNTIL, следует убедиться в правильном обращении с содержимым стеков циклов.

Генерируемое состояние: Не влияет на биты состояния.

Формат команды:
Команда возвращения по условию. Тип 20:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	COND			

COND: Условие

15 Команды управления последовательностью выполнения программы

Команда организации цикла DO UNTIL

Синтаксис: DO <addr> [UNTIL условие окончания];

Разрешенные условия окончания

EQ	NE	GT	GE	LT	FOREVER
LE	NEG	POS	AV	NOT AV	
AC	NOT AC	MV	NOT MV	NOT CE	

Пример: DO *loop_label* UNTIL CE; {CNTR декрементируется с каждым прохождением через цикл}

Описание: Команда DO UNTIL организует циклы с нулевыми издержками. Цикл программы начинается с команды, непосредственно следующей за командой DO, заканчивается по указанному в команде адресу и повторяется до тех пор, пока не удовлетворяется заданное условие окончания цикла (если такое условие задается) или повторяется бесконечно (если такое условие не указано). Условие прекращения цикла проверяется во время выполнения последней команды цикла, когда по завершению выполнения предыдущей команды уже был сгенерирован бит состояния. Адрес последней команды цикла (<addr>) содержится непосредственно в командном слове.

Если в качестве условия окончания цикла используется CE (счетчик пуст), то содержимое счетчика процессора декрементируется при каждом прохождении цикла.

Во время выполнения команды DO адрес последней команды помещается в стек цикла вместе с условием прекращения цикла, а текущее значение программного счетчика плюс единица помещается в стек программного счетчика (PC).

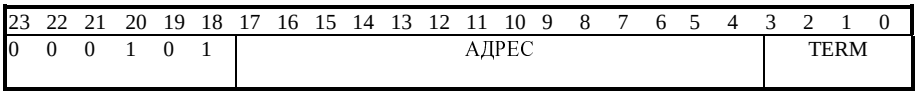
При вложении циклов DO UNTIL в стеки циклов и программного счетчика помещаются таким же образом другие значения для вложенных циклов до тех пор, пока это позволяет размер стека циклов (4 уровня вложенности циклов) и стека счетчика программ (16 уровней для подпрограмм плюс прерываний плюс циклов). Когда один или оба указанных стека заполняются до конца, при попытке выполнить еще одну команду DO будет сгенерирован бит переполнения соответствующего стека и не будет выполняться никаких операций.

Генерируемое состояние:
ASTAT: Не влияет на биты состояния.

SSTAT:	7	6	5	4	3	2	1	0
	LSO	LSE	SSO	SSE	CSO	CSE	PSO	PSE
	*	0	-	-	-	-	*	0

LSO Бит переполнения стека циклов: устанавливается при переполнении стека циклов; в других случаях никак не изменяется.
LSE Бит, указывающий, что стек циклов пуст: всегда сброшен (указывая на то, что стек циклов не пуст).
PSO Бит переполнения стека программного счетчика (PC): устанавливается при переполнении стека счетчика программ; в других случаях не изменяется.
PSE Бит, указывающий, что стек программного счетчика пуст: всегда сброшен (указывая на то, что стек PC не пуст).

Формат команды:
Команда DO UNTIL. Тип 11:



Группа разрядов АДРЕС определяет адрес последней команды в цикле. Синтаксис данной команды позволяет записывать в это поле метку программы или непосредственно значение адреса.

Группа разрядов TERM определяет условие окончания цикла, как показано ниже:

TERM	*Синтаксис*	*Проверяемое условие*
0000	NE	Не равно нулю
0001	EQ	Равно нулю
0010	LE	Меньше или равно нулю
0011	GT	Больше нуля
0100	GE	Больше или равно нулю
0101	LT	Меньше нуля
0110	NOT AV	Нет переполнения в АЛУ
0111	AV	Переполнение в АЛУ
1000	NOT AC	Нет переноса в АЛУ
1001	AC	Перенос в АЛУ
1010	POS	Операнд X последней команды ABS был положительным
1011	NEG	Операнд X последней команды ABS был отрицательным
1100	NOT MV	Нет переполнения в умножителе-накопителе
1101	MV	Переполнение в умножителе-накопителе
1110	CE	Счетчик пуст
1111	FOREVER	Всегда

15 Команды управления последовательностью выполнения программы
Ожидание (IDLE)

Синтаксис: IDLE;
IDLE(n); *Команда IDLE с замедлением частоты работы процессора*

Описание: По команде IDLE процессор ожидает прерывания в течение неопределенного периода времени находясь в состоянии пониженной мощности. По приходу прерывания оно обслуживается, и программа продолжает выполняться, начиная с команды, следующей за командой IDLE. Как правило, за командой IDLE следует команда перехода JUMP обратно к команде IDLE, за счет чего осуществляется организация цикла, в котором процессор находится в состоянии ожидания при пониженной потребляемой мощности. (Обратите внимание на ограничения, возникающие при использовании команд IDLE и JUMP в качестве последней команды в цикле DO UNTIL; об этих ограничениях рассказывалось в главе 3).

IDLE(n) представляет собой особую разновидность команды IDLE, которая замедляет внутреннюю тактовую частоту процессора для еще большего уменьшения потребления энергии. Уменьшенная тактовая частота является программируемой долей от нормальной тактовой частоты и задается по выбору делителем n, который указывается в тексте команды: n = 16, 32, 64 или 128. Во время выполнения команды IDLE n процессор остается в рабочем состоянии, уменьшается его быстродействие. Все время, пока процессор находится в таком состоянии, частота всех других его внутренних тактовых сигналов (например, SCLKIN, CLKOUT и тактовые синхроимпульсы таймера) уменьшается в той же пропорции.

При использовании команды IDLE(n) внутренняя тактовая частота процессора замедляется и, следовательно, время отклика процессора на приходящие прерывания, составляющее в нормальном состоянии IDLE 1 цикл, также увеличивается на величину делителя тактовой частоты n. При приеме разрешенного прерывания процессор семейства ADSP-2100 остается в состоянии IDLE в течение максимум n циклов CLKIN (где n = 16, 32, 64, 128) прежде, чем возобновит свою нормальную работу.

При использовании команды IDLE(n) в системах с внешним генератором тактовых синхроимпульсов частота последовательных тактовых синхроимпульсов может превышать уменьшенную внутреннюю тактовую частоту процессора. В этих условиях прерывания не должны генерироваться с большей скоростью, чем они могут быть обслужены, из-за того, что процессору требуется дополнительное время для выхода из состояния IDLE (максимум n циклов CLKIN).

Команды управления последовательнстью выполнения программы 15
Ожидание (IDLE)

Во время выполнения команды IDLE по-прежнему остаются возможными операции автобуферизации данных, которые не влияют на состояние IDLE.

Генерируемое состояние: Не влияет на биты состояния.

Формат команды:
Команда IDLE. Тип 31:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Медленная команда IDLE. Тип 31:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	ДЕЛИТЕЛЬ			

ДЕЛИТЕЛЬ: делитель тактовой частоты n.

15 Прочие команды
Управление стеками

Синтаксис: [PUSH | POP | STS] [,POP CNTR] [,POP PC] [,POP LOOP];

Пример: POP CNTR, POP PC, POP LOOP;

Описание: По командам управления стеками содержимое назначенных стеков выталкивается или в эти стеки, наоборот, помещаются значения. Вся команда целиком выполняется в течение одного цикла независимо от числа стеков, над которыми производятся вышеназванные действия.

По команде PUSH STS (поместить значение в стек состояний) осуществлятся приращение указателя стека состояний на единицу так, чтобы он указывал на новую доступную ячейку в этом стеке, а затем в стек состояний процессора помещается значение регистра состояний арифметических устройств (ASTAT), регистра состояния режима (MSTAT) и регистра маскирования прерываний (IMASK). Обратите внимание, что операция PUSH STS автоматически выполняется каждый раз при входе в подпрограмму обслуживания прерывания.

По любой команде POP из назначенного в ней стека выталкивается его верхнее значение, а указатель этого стека декрементируется, чтобы указывать на следующую ниже ячейку в стеке. По команде POP STS помещенная в стек информация о состоянии арифметических устройств, режима и о маскировании прерываний извлекается в регистры ASTAT, MSTAT и IMASK соответственно. Данная операция также осуществляется автоматически при возвращении из подпрограммы обслуживания прерывания в главную программу (по команде RTI).

По команде POP CNTR содержимое стека счетчика извлекается в вычитающий счетчик. При извлечении данных из стека циклов или стека программного счетчика (PC) (по командам POP LOOP и POP PC, соответственно) извлекаемая из них информация теряется. При возвращении в главную программу после обслуживания прерывания (команда RTI) или из подпрограммы (команда RTS) происходит автоматическое извлечение значения из стека программного счетчика.

(Примечание: для считывания (и извлечения) или записи (и помещения в стек) значения на вершине стека программного счетчика предусмотрена особая команда. Эта команда использует псевдорегистр TOPPCSTACK, о котором рассказывается в обзорном приложении к данному изданию).

Генерируемое состояние:

SSTAT:	7	6	5	4	3	2	1	0
	LSO	LSE	SSO	SSE	CSO	CSE	PSO	PSE
	-	*	*	*	-	*	-	*

PSE Стек счетчика программ пуст: этот бит устанавливается, если в результате извлечения данных из стека счетчика программ тот полностью опустошается; в других случаях сброшен.

CSE Стек счетчика пуст: этот бит устанавливается, если в результате извлечения данных из стека счетчика тот полностью опустошается; в других случаях сброшен.

SSE Стек состояний пуст: этот бит всегда сброшен для команды PUSH STS (указывая на то, что стек состояний не пуст). Для команды POP STS этот бит устанавливается, если в результате извлечения данных из стека состояний тот полностью опустошается; в других случаях этот бит сброшен.

SSO Переполнение стека состояний: для команды PUSH STS этот бит установлен, если происходит переполнение стека состояний; в других случаях этот бит не затрагивается.

LSE Стек циклов пуст: этот бит установлен, если в результате извлечения данных из стека циклов тот полностью опустошается; в других случаях этот бит сброшен.

Обратите внимание, что как только происходит переполнение какого-либо стека, немедленно генерируется соответствующий бит переполнения стека в регистре SSTAT, этот бит остается установленным, указывая на то, что произошла потеря информации. Будучи установленным бит переполнения стека может быть сброшен только при перезапуске процессора.

Формат команды:

Команда управления стеками. Тип 26:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	Pp	Lp	Cp	Spp	

Pp: Управление стеком программного счетчика (PC)

Lp: Управление стеком циклов

Cp Управление стеком счетчика

Spp: Управление стеком состояний

15 Прочие команды
Управление стеками

TOPPCSTACK

Особая версия команды пересылки данных между регистрами, Тип 17, используется для считывания (или извлечения данных) или записи (или помещения данных) верхнего значения стека счетчика команд. При выполнении обычной команды POP PC извлекаемое из стека значение не сохраняется, поэтому для того, чтобы сохранить это значение, следует использовать следующую команду:

```
reg=TOPPCSTACK;  {извлечение данных из стека счетчика команд в регистр}
                 {"toppcstack" может быть набрано малыми буквами}
```

Кроме того, выполнение этой команды требует задержки в один цикл. Поэтому необходимо поместить после TOPPCSTACK команду NOP для правильного выполнения операции извлечения данных из стека и их записи в регистр.

```
reg=TOPPCSTACK;
NOP;    {позволяет правильно выполнить операцию над стеком}
```

Нет никакой стандартной команды для помещения данных в стек счетчика команд. Чтобы записать опередеденное значение в стек счетчика команд, следует использовать следующую специальную команду:

```
TOPPCSTACK=reg;  {содержимое регистра помещается в стек счетчика команд}
```

Данные помещаются в стек в течение того же цикла.
Обратите внимание, что "TOPPCSTACK" не может использоваться в качестве регистра в команде любого другого типа!

Примеры:

```
AX0=TOPPCSTACK;     {содержимое стека счетчика команд помещается в ре-}
NOP;                {гистр AX0}

TOPPCSTACK=I7;      {содержимое регистра I7 помещается в стек счетчика }
                    {команд}
```

В специальной команде TOPPCSTACK могут использоваться только следующие регистры:

Регистры АЛУ, умножителя-накопителя и устройства сдвига			*Регистры генераторов адреса данных*					
AX0	AR	SI	I0	I4	M0	M4	L0	L4
AX1	MR0	SE	I1	I5	M1	M5	L1	L5
MX0	MR1	SR0	I2	I6	M2	M6	L2	L6
MX1	MR2	SR1	I3	I7	M3	M7	L3	L7
AY0								
AY1								
MY0								
MY1								

Имеется несколько ограничений на использование особой команды TOPPCSTACK, эти ограничения описывались в главе 3, "Управление программой".

Формат команды:

TOPPCSTACK=reg

Команда внутренней пересылки данных. Тип 17:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	1	0	0	0	0	1	1	SRC RGP		1	1	1	1	SOURCE REG			

Группа исходных регистров (SRC RGP) и исходный регистр (SOURCE REG) выбирают исходный регистр по Таблице выбора регистров (см. Приложение А).

reg=TOPPCSTACK

Команда внутренней пересылки данных. Тип 17:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	1	0	0	0	0	DST RGP		1	1	DEST REG				1	1	1	1

Группа регистров назначения (DST RGP) и регистр назначения (DEST REG) выбирают регистр назначения по Таблице выбора регистров (см. Приложение А).

15 Прочие команды
Управление режимом

Синтаксис:	ENA DIS		BIT_REV AV_LATCH AR_SAT SEC_REG G_MODE M_MODE TIMER	[, ...];

Пример: DIS AR_SAT, ENA M_MODE;

Описание: Разрешение (ENA) или блокирование (DIS) назначенного режима процессора. Соответствующий бит состояния режима устанавливается в регистре состояния режима MSTAT при разрешении режима и сбрасывается при его блокировании. При перезапуске все биты регистра MSTAT устанавливаются равными нулю, что означает, что блокированы все режимы. По этой команде в течение одного цикла может изменяться любое количество режимов. Несколько операторов ENA или DIS должны отделяться друг от друга запятыми.

Биты регистра MSTAT:

0	SEC_REG	Банк данных теневого регистра
1	BIT_REV	Режим бит-реверсной адресации для генератора адреса данных №1
2	AV_LATCH	Режим фиксации состояния переполнения АЛУ
3	AR_SAT	Режим насыщения регистра AR АЛУ
4	M_MODE	Режим помещения результата в умножителе-накопителе
5	TIMER	Таймер активизирован
6	G_MODE	Режим GO разрешен

Бит выбора банка регистра данных (SEC_REG) определяет активный на текущий момент набор регистров данных (0=основной, 1=теневой).

Когда бит режима бит-реверсной адресации (BIT_REV) равен единице, биты адресов, сгенерированные генератором адреса данных №1, ставятся на выходе этого генератора в обратном порядке.

Когда бит режима фиксации переполнения АЛУ (AV_LATCH) равен единице, бит AV в регистре состояний арифметических устройств остается установленным при единственном переполнении АЛУ. В этом режиме в случае переполнения АЛУ бит AV будет установлен и останется в таком состоянии, даже если при последующих операциях АЛУ переполнения больше не происходит. Бит AV может быть сброшен только путем записи его нулем непосредственно по шине ДПД.

Прочие команды 15
Управление режимом

Когда бит насыщения регистра AR (AR_SAT) равен единице, в случае переполнения АЛУ происходит насыщение регистра AR, как уже рассказывалось в разделе, посвященном АЛУ.

Режим размещения результата в умножителе-накопителе (M_MODE) определяет, будет ли сделан левый сдвиг между произведением, полученным умножителем, и регистром MR.

После установки бита активизации таймера (TIMER) начинает работать логика декрементирования таймера. При сбросе этого бита останавливается работа таймера.

Режим GO позволяет любому процессору семейства ADSP-2100 продолжать выполнение команд, взятых из внутренней памяти (если это возможно), во время операции предоставления шины. Режим GO позволяет процессору работать во всех случаях, кроме тех, когда требуется обращение к внешней памяти - в последнем случае процессор приостанавливает работу, ожидая освобождения шины.

Формат команды:
Команда управления режимом. Тип 18:

23	22	21	20	19	18	17	16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
0	0	0	0	1	1	0	0	TI	MM	AS	OL	BR	SR	GM	0 0

TI: Активизация таймера

MM: Размещение результата в умножителе

AS: Управление режимом насыщения AR

BR: Управление режимом бит-реверсной адресации

OL: Управление режимом фиксации переполнения АЛУ

GM: Режим GO

SR: Режим банка теневых регистров

15 Прочие команды
Модификация адреса

Синтаксис: MODIFY (| I0 | , | M0 |);
I1		M1
I2		M2
I3		M3
I4		M4
I5		M5
I6		M6
I7		M7

Пример: MODIFY (I1,M1);

Описание: Добавление значения из выбранного регистра M (Mn) к содержимому выбранного регистра I (Im) с последующей обработкой модифицированного адреса (логика адресации по модулю) с учетом длины буфера, заданной в соответствующем выбранному регистру I регистре L (Lm), после чего запись вычисленного в результате указателя адреса в выбранный регистр I. Модификация содержимого регистра I осуществляется как при индексировании адреса памяти, но при этом не происходит никаких пересылок данных. **При адресации линейных (т.е. нециклических) буферов регистр L, соответствующий используемому регистру I, должен быть установлен равным нулю.**

Выбор регистров I и M ограничивается требованием их принадлежности к одному генератору адреса данных: выбор регистров I0-I3 Генератора адреса данных №1 ограничивает выбор регистра M регистрами M0-M3. Аналогичным образом, если выбирается любой из регистров I4-I7, выбор регистра M ограничивается регистрами M4-M7.

Генерируемое состояние: Не затрагивает биты состояния.

Формат команды:
Команда модификации адреса. Тип 21:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	G	I		M	

G определяет выбор генератора адреса данных. Заданные регистры I и M должны быть из одного и того же генератора адреса данных, на что указывает отделение регистров разных генераторов адреса данных чертой. Группа разрядов I определяет регистр I (что зависит от того, какой генератор адреса данных выбран битом G). M определяет используемый регистр M (также зависит от выбранного битом G генератора адреса данных).

Прочие команды 15
Команда NOP (нет операций)

Синтаксис: NOP;

Описание: В течение одного цикла не производится никаких операций. Выполнение программы продолжается со следующей за командой NOP команды.

Генерируемое состояние: Не влияет на биты состояния.

Формат команды:
Команда "нет операций". Тип 30 (см. Приложение А):

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15 Прочие команды

Разрешение/блокирование прерываний

(Только ADSp-217x, ADSP-218x, ADSp-21msp58/59)

Синтаксис: ENA INTS;
DIS INTS;

Описание: после перезапуска процессора все прерывания по умолчанию разрешены. При выполнении команды DIS INTS все прерывания (включая прерывание при входе в режим пониженной мощности) маскируются, а содержимое регистра IMASK при этом не меняется.

По команде ENA INTS все немаскированные прерывания обслуживаются заново.

Примечание: блокирование прерываний не влияет на автобуферизацию последовательных портов и передачи данных через порты прямого обращения к памяти процессоров ADSP-218x (BDMA, IDMA). Независимо от того, разрешены или запрещены прерывания, указанные операции выполняются в нормальном режиме.

Генерируемое состояние: Не изменяется

Формат команды:

Команда блокирования прерываний. Тип 26:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Команда разрешения прерываний. Тип 26:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Вычисление с одновременным считыванием из памяти

Синтаксис:

<ALU> <MAC> <SHIFT>	, dreg =	DM(		I0 I1 I2 I3	,	M0 M1 M2 M3		)	;
				I4 I5 I6 I7		M4 M5 M6 M7			
		PM(		I4 I5 I6 I7	,	M4 M5 M6 M7		)	

Разрешенные dreg

AX0	MX0	SI
AX1	MX1	SE
AY0	MY0	SR0
AY1	MY1	SR1
AR	MR0	
	MR1	
	MR2	

Описание: Выполнение обозначенной арифметической операции и пересылка данных. При операции считывания данные пересылаются в регистр назначения. Когда арифметическая операция выполняется в комбинации со считыванием из памяти, используется режим косвенной адресации с пост-модификацией адреса. **При адресации линейного (т.е. нециклического) буфера регистр L, соответствующий используемому регистру I, должен быть установлен равным нулю.** Пересылаемые из памяти данные всегда выравниваются по правому краю регистра назначения.

Вычислительная операция должна быть безусловной. Разрешены все операции АЛУ (ALU), умножителя-накопителя (MAC) и устройства сдвига (SHIFT), за исключением непосредственного сдвига и команд АЛУ DIVS и DIVQ.

Основное правило, управляющее всеми многофункциональными командами, гласит, что считывание из регистров (и памяти) осуществляется в начале цикла процессора, а запись - в конце цикла. Нормальный порядок расположения операторов слева направо (т.е. сначала вычисление, затем считывание из памяти) соответствует такой последовательности выполнения

команды. В действительности, вполне возможно поменять порядок операторов в данной команде. При этом ассемблер выдаст предупреждение, но на уровне кода операции результат будет одним и тем же. Если вы отключаете проверку семантики в ассемблере (используя для этого переключатель -s), ассемблер не выдает такого предупреждения.

Благодаря тому, что процессор сначала считывает, а затем только записывает данные, один и тот же регистр может использоваться как исходный в одном операторе и как регистр назначения в другом операторе. В таких случаях из регистра берется одно значение в начале цикла, а в конце цикла в него записывается новое значение. Например, команда:

(1) AR = AX0 + AY0, AX0 = DM(I0,M0);

является разрешенной версией рассматриваемой многофункциональной команды, которую ассемблер принимает без замечаний. При перестановке операторов в обратном порядке:

(2) AX0 = DM(I0,M0), AR = AX0 + AY0;

ассемблер выдает предупреждение, но транслирует и выполняет вторую команду в точности так же, как и предыдущую. Обратите внимание, что при чтении примера (2) слева направо возникает предположение, что значение сначала загружается из памяти данных в регистр AX0, а затем используется в вычислении, и все это в течение одного цикла. В действительности это невозможно. Логика расположения операторов слева направо в примере (1) более точно соответствует порядку выполнения команды. Независимо от того, в каком порядке расположены операторы слева направо, выполнение команды определяется правилом процессора: сначала считывание, затем запись.

Однако, при использовании одного и того же регистра в качестве регистра назначения в обоих операторах, чего не следует делать, получается неопределенный результат. Если проверка семантики не отключена, ассемблер выдает предупреждение. Тем не менее, независимо от того, выдано или нет предупреждение, данная операция не будет поддержана процессором.

Приводимая ниже команда является недопустимой и не поддерживается, хотя при проверке семантики выдается только предупреждение:

(3) AR = AX0 + AY0, AR = DM(I0,M0); *недопустимо!*

Генерируемое состояние: все биты состояний меняются также как при соответствующей однофункциональной арифметической операции.

Операция АЛУ (<ALU>):

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	-	-	*	*	*	*	*

Z Установлен, если результат равен нулю. В других случаях сброшен.
AN Установлен, если результат - отрицательное число. Иначе сброшен.
AV Установлен, если генерируется переполнение. Иначе сброшен.
AC Устанавливается, если генерируется перенос. Иначе сброшен.
AS Затрагивается только при выполнении операции на нахождение абсолютного значения (ABS). Устанавливается, если исходный операнд - отрицательное число.

Операция умножителя-накопителя (<MAC>):

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	*	-	-	-	-	-	-

MV Устанавливается, если накопленный результат переполняет 32 младших бита в регистре MR. Иначе сброшен.

Операция устройства сдвига (<SHIFT>):

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	*	-	-	-	-	-	-	-

SS Затрагивается только при выполнении операции EXP; устанавливается, если исходный операнд - отрицательное число и сброшен, если это положительное число.

Формат команды:
Операция АЛУ/умножителя-накопителя с одновременным считыванием из памяти данных. Тип 4:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	G	0	Z	AMF					Yop		Xop			DREG				I		M	

Операция АЛУ/умножителя-накопителя с одновременным считыванием из памяти программы.
Тип 5:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	Z	AMF					Yop		Xop			DREG				I		M	

15 Многофункциональные команды
Вычисление с одновременным считыванием из памяти

Операция устройства сдвига с одновременным считыванием из памяти данных. Тип 12:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	1	G	0	SF				Xop			DREG				I		M	

Операция устройства сдвига с одновременным считыванием из памяти программы. Тип 13:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	0	1	0	SF				Xop			DREG				I		M	

Z:	Регистр результата	DREG:	Регистр назначения
SF:	Операция устройства сдвига	AMF:	Операция АЛУ/умножителя-накопителя
Yop:	Операнд Y		
G:	Генератор адреса данных	Xop:	Операнд X
M:	Регистр модификации	I:	Регистр косвенного адреса

Многофункциональные команды 15
Вычисление с одновременной пересылкой данных между регистрами

Синтаксис:	<ALU> <MAC> <SHIFT>	, dreg = dreg;

Разрешенные dreg

AX0	MX0	SI
AX1	MX1	SE
AY0	MY0	SR0
AY1	MY1	SR1
AR	MR0	
	MR1	
	MR2	

Описание: Выполнение заданной арифметической операции и пересылка данных. Пересылаемое значение всегда выравнивается по правому краю в регистре назначения после считывания.

Вычислительная операция должна быть безусловной. Разрешены все операции АЛУ (ALU), умножителя-накопителя (MAC) и устройства сдвига (SHIFT), кроме непосредственного сдвига и команд АЛУ DIVS и DIVQ.

Основное правило, управляющее всеми многофункциональными командами, гласит, что считывание из регистров (и памяти) осуществляется в начале цикла процессора, а запись - в конце цикла. Нормальный порядок расположения операторов слева направо (т.е. сначала вычисление, затем считывание из памяти) поддерживает такую последовательность выполнения команды. В действительности, вполне возможно поменять порядок операторов в данной команде. При этом ассемблер выдаст предупреждение, но на уровне кода операции результат будет одним и тем же. Если вы отключаете проверку семантики в ассемблере (используя для этого переключатель -s), ассемблер не выдает такого предупреждения.

Благодаря тому, что процессор сначала считывает, а затем только записывает данные, один и тот же регистр может использоваться как исходный в одном операторе и как регистр назначения в другом операторе. В таких случаях из регистра берется одно значение в начале цикла, а конце цикла в него записывается новое значение.

Например, команда:

(1) AR = AX0 + AY0, AX0 = MR1;

является разрешенной версией рассматриваемой многофункциональной команды, которую ассемблер принимает без замечаний. При перестановке операторов в обратном порядке:

15 Многофункциональные команды
Вычисление с одновременной пересылкой данных между регистрами

(2) AX0 = MR1, AR = AX0 + AY0;

ассемблер выдает предупреждение, но транслирует и выполняет вторую команду в точности так же, как и предыдущую. Обратите внимание, что при чтении примера (2) слева направо возникает предположение, что значение сначала загружается из регистра MR1 в регистр AX0, а затем содержимое последнего используется в вычислении, и все это в течение одного цикла. В действительности, это невозможно. Логика расположения операторов слева направо в примере (1) более точно соответствует порядку выполнения команды. Независимо от того, в каком порядке расположены операторы слева направо, выполнение команды определяется правилом процессора: сначала считывание, затем запись.

При использовании одного и того же регистра в качестве регистра назначения в обоих операторах, что недопустимо, получается неопределенный результат. Если проверка семантики не отключена, ассемблер выдает предупреждение. Тем не менее, независимо от того, выдано или нет предупреждение, данная операция не будет поддержана процессором. Приводимая ниже команда является недопустимой и не поддерживается, хотя при проверке семантики выдается только предупреждение:

(3) AR = AX0 + AY0, AR = MR1; *недопустимо!*

Генерируемое состояние: все биты состояний меняются также как при соответствующей однофункциональной арифметической операции.

Операция АЛУ (<ALU>):

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	-	-	*	*	*	*	*

AZ Установлен, если результат равен нулю. В других случаях сброшен.
AN Установлен, если результат - отрицательное число. Иначе сброшен.
AV Установлен, если генерируется переполнение. Иначе сброшен.
AC Устанавливается, если генерируется перенос. Иначе сброшен.
AS Затрагивается только при выполнении операции на нахождение абсолютного значения (ABS). Устанавливается, если исходный операнд - отрицательное число.

Операция умножителя-накопителя (<MAC>):

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	*	-	-	-	-	-	-

MV Устанавливается, если накопленный результат переполняет 32 младших бита в регистре MR. Иначе сброшен.

Операция устройства сдвига (<SHIFT>):

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	*	-	-	-	-	-	-	-

SS Затрагивается только при выполнении операции EXP; устанавливается, если исходный операнд - отрицательное число и сброшен, если это положительное число.

Формат команды:
Операция АЛУ/умножителя с одновременной пересылкой данных. Тип 8:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	Z	AMF					Yop		Xop			Dreg dest				Dreg source			

Операция устройства сдвига с одновременной пересылкой данных. Тип 14:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	0	0	0	SF				Xop			Dreg dest				Dreg source			

Z: Регистр результата

SF: Операция устройства сдвига

Yop: Операнд Y

Dreg: Регистр данных

AMF: Операция АЛУ/умножителя-накопителя

Xop: Операнд X

15 Многофункциональные команды
Вычисление с одновременной записью в память

Синтаксис:

DM(		I0	,	M0	)	= dreg,	<ALU>	;
		I1		M1			<MAC>	
		I2		M2			<SHIFT>	
		I3		M3				
		I4		M4				
		I5		M5				
		I6		M6				
		I7		M7				
PM(		I4	,	M4	)			
		I5		M5				
		I6		M6				
		I7		M7				

Разрешенные dreg

AX0	MX0	SI
AX1	MX1	SE
AY0	MY0	SR0
AY1	MY1	SR1
AR	MR0	
	MR1	
	MR2	

Описание: Выполнение обозначенной арифметической операции и пересылка данных. При операции записи содержимое исходного регистра пересылается по указанному адресу в памяти. Когда арифметическая операция выполняется в комбинации с записью в память, используется режим косвенной адресации с пост-модификацией адреса. **При адресации линейного (т.е. нециклического) буфера регистр L, соответствующий используемому регистру I, должен быть установлен равным нулю.** Пересылаемые данные всегда выравниваются по правому краю по месту назначения.

Вычислительная операция должна быть безусловной. Разрешены все операции АЛУ (ALU), умножителя-накопителя (MAC) и устройства сдвига (SHIFT), за исключением непосредственного сдвига и команд АЛУ DIVS и DIVQ.

Основное правило, управляющее всеми многофункциональными командами, гласит, что считывание из регистров (и памяти) осуществляется в начале цикла процессора, а запись - в конце цикла. Нормальный порядок расположения операторов слева направо (т.е. сначала запись в память, а затем вычисление) поддерживает такую последовательность выполнения команды. В действительности,

вполне возможно поменять порядок операторов в данной команде. При этом ассемблер выдаст предупреждение, но на уровне кода операции результат будет одним и тем же. Если вы отключите проверку семантики в ассемблере (используя для этого переключатель -s), ассемблер не выдаст такого предупреждения.

Благодаря тому, что процессор сначала считывает, а затем только записывает данные, один и тот же регистр может использоваться как исходный в одном операторе и как регистр назначения в другом операторе. В таких случаях из регистра берется одно значение в начале цикла, а в конце цикла в него записывается новое значение. Например, команда:

(1) DM(I0,M0) = AR, AR = AX0 + AY0;

является разрешенной версией рассматриваемой многофункциональной команды, которую ассемблер принимает без замечаний. При перестановке операторов в обратном порядке:

(2) AR = AX0 + AY0, DM(I0,M0)=AR;

ассемблер выдает предупреждение, но транслирует и выполняет вторую команду так же, как и предыдущую. Обратите внимание, что при чтении примера (2) слева направо возникает предположение, что результат вычисления сначала загружается в регистр AR, а затем записывается в память, и все это в течение одного цикла. В действительности, это невозможно. Логика расположения операторов слева направо в примере (1) более точно соответствует порядку выполнения команды. Независимо от того, в каком порядке расположены операторы слева направо, выполнение команды определяется правилом процессора: сначала считывание, затем запись.

Генерируемое состояние: все биты состояний меняются также как при соответствующей однофункциональной арифметической операции.

Операция АЛУ (<ALU>):

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	-	-	*	*	*	*	*

AZ Установлен, если результат равен нулю. В других случаях сброшен.
AN Установлен, если результат - отрицательное число. Иначе сброшен.
AV Установлен, если генерируется переполнение. Иначе сброшен.
AC Устанавливается, если генерируется перенос. Иначе сброшен.
AS Затрагивается только при выполнении операции на нахождение абсолютного значения (ABS). Устанавливается, если исходный операнд - отрицательное число.

15 Многофункциональные команды
Вычисление с одновременной записью в память

Операция умножителя-накопителя (<MAC>):

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	*	-	-	-	-	-	-

MV Устанавливается, если накопленный результат переполняет 32 младших бита в регистре MR. Иначе сброшен.

Операция устройства сдвига (<SHIFT>):

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	*	-	-	-	-	-	-	-

SS Затрагивается только при выполнении операции EXP; устанавливается, если исходный операнд - отрицательное число и сброшен, если это положительное число.

Формат команды:

Операция АЛУ/умножителя-накопителя с одновременной записью в память данных. Тип 4:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	G	1	Z	AMF					Yop		Xop			DREG				I		M	

Операция АЛУ/умножителя-накопителя с одновременной записью в память программы. Тип 5:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	1	Z	AMF					Yop		Xop			DREG				I		M	

Операция устройства сдвига с одновременной записью в память данных. Тип 12:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	1	G	1	SF				Xop			DREG				I		M	

Операция устройства сдвига с одновременной записью в память программы. Тип 13:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	0	1	1	SF				Xop			DREG				I		M	

Z:	Регистр результата	DREG:	Регистр назначения
SF:	Операция устройства сдвига	AMF:	Операция АЛУ/умножителя-накопителя
Yop:	Операнд Y		
G:	Генератор адреса данных	Xop:	Операнд X
M:	Регистр модификации	I:	Регистр косвенного адреса

Многофункциональные команды 15
Одновременное считывание из памяти данных и памяти программы

Синтаксис:

AX0 AX1 MX0 MX1	= DM(	I0 I1 I2 I3	,	M0 M1 M2 M3	),	AY0 AY1 MY0 MY1	= PM(	I4 I5 I6 I7	,	M4 M5 M6 M7	);

Описание: Выполнение нескольких операций считывания из памяти, одно - из памяти данных, другое - из памяти программы. При каждой операции считывания содержимое ячейки памяти пересылается в регистр назначения. Регистрами назначения являются регистры X АЛУ и умножителя-накопителя для выборки из памяти данных и регистры Y - для выборки из памяти программы. Для данной двойной выборки из памяти используется режим косвенной адресации с пост-модификацией адреса. **При косвенной адресации линейного (т.е. нециклического) буфера регистр L, соответствующий используемому регистру I, должен быть установлен равным нулю.** Пересылаемые данные всегда выравниваются по правому краю в регистре назначения.

Об условиях, при которых для считывания из памяти требуются дополнительные циклы, уже рассказывалось в самом начале данной главы.

Генерируемое состояние: не влияет на биты состояния.

Формат команды:
Команда АЛУ/умножителя-накопителя с одновременным считыванием из памяти данных и памяти программы. Тип 1:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	PD		DD		AMF					0	0	0	0	0	PM I		PM M		DM I		DM M	

AMF определяет операцию АЛУ или умножителя-накопителя. В данном случае, AMF = 00000, указывая на отсутствие операций АЛУ или умножителя-накопителя.

PD: Регистр назначения для памяти программы
DD: Регистр назначения для памяти данных
AMF: Операция АЛУ/умножителя-накопителя
I: Регистр косвенного адреса
M: Регистр модификации

15 Многофункциональные команды
Выполнение операций АЛУ/умножителя-накопителя с одновременным считыванием из памяти данных и памяти программы

Синтаксис:

<ALU>	,	AX0	=DM(	I0	,	M0	),	AY0	= PM(	I4	,	M4	);
<MAC>		AX1		I1		M1		AY1		I5		M5	
		MX0		I2		M2		MY0		I6		M6	
		MX1		I3		M3		MY1		I7		M7	

Описание: Данная команда объединяет операцию АЛУ или умножителя-накопителя с двойным считыванием из памяти данных и памяти программы. При операциях считывания содержимое указанной в команде ячейки памяти пересылается в регистр назначения. Регистрами назначения для считывания из памяти данных являются регистры X АЛУ и умножителя-накопителя, регистрами назначения для считывания из памяти программы - регистры Y. Используется косвенная адресация с постмодификацией адреса. **Для косвенной адресации линейного (т.е. нециклического) буфера регистр L, соответствующий используемому регистру I, должен быть установлен равным нулю.** Пересылаемые данные всегда выравниваются по правому краю в регистре назначения после считывания.

Вычислительная операция должна быть безусловной. Разрешены все операции АЛУ и умножителя-накопителя, за исключением команд АЛУ DIVS и DIVQ. Результаты вычисления должны записываться в регистр R вычислительного устройства: результаты АЛУ - в регистр AR, результаты умножителя-накопителя - в регистр MR.

Основной принцип, управляющий всеми многофункциональными командами, гласит: сначала считывание из регистров (и из памяти) в начале цикла, затем запись в них в конце цикла. Обычный порядок расположения операторов слева направо (сначала вычисления, затем считывания из памяти) поддерживает данную последовательность выполнения команды. В действительности, можно поменять операторы местами. При этом ассемблер выдаст предупреждение, однако результаты на уровне кода операции будут теми же. Если вы отключите проверку семантики в ассемблере (используя переключатель -s), предупреждение не будет выдано.

Один и тот же регистр может использоваться как исходный для арифметической операции и как регистр назначения для считывания из памяти. В начале цикла из этого регистра берется его текущее значение, а в конце цикла в него записывается новое значение. Например, команда:

(1) MR = MR + MX0*MY0(UU), MX0 = DM(I0,M0), MY0 = PM(I4,M4);

представляет собой разрешенную версию рассматриваемой многофункциональной команды и никак не комментируется ассемблером. При изменении порядка операторов в данной команде:

(2) MX0 = DM(I0,M0), MY0 = PM(I4,M4), MR = MR +MX0*MY0(UU);

ассемблер выдает предупреждение, но транслирует и выполняет команду совершенно также, как и предыдущую (1). Обратите внимание, что при чтении примера (2) слева направо может возникнуть предположение, что значения из памяти сначала загружаются MX0 и MY0, а затем используются в вычислении, и все это в течение одного цикла. Однако, это невозможно. Логика расположения операторов слева направо в примере (1) более точно отражает последовательность выполнения этой многофункциональной команды. Независимо от расположения операторов слева направо, порядок действий процессора определяется его основным правилом: сначала считывание, затем запись.

Генерируемое состояние: все биты состояний меняются так же, как при соответствующей однофункциональной арифметической операции.

Операция АЛУ (<ALU>):

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	-	-	*	*	*	*	*

AZ Установлен, если результат равен нулю. В других случаях сброшен.
AN Установлен, если результат - отрицательное число. Иначе сброшен.
AV Установлен, если генерируется переполнение. Иначе сброшен.
AC Устанавливается, если генерируется перенос. Иначе сброшен.
AS Затрагивается только при выполнении операции на нахождение абсолютного значения (ABS). Устанавливается, если исходный операнд - отрицательное число.

Операция умножителя-накопителя (<MAC>):

ASTAT:	7	6	5	4	3	2	1	0
	SS	MV	AQ	AS	AC	AV	AN	AZ
	-	*	-	-	-	-	-	-

MV Устанавливается, если накопленный результат переполняет 32 младших бита в регистре MR. Иначе сброшен.

Формат команды:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	PD		DD		AMF					Yop		Xop			PM I		PM M		DM I		DM M	

Многофункциональные команды 15
Выполнение операций АЛУ/умножителя-накопителя с одновременным считыванием из памяти данных и памяти программы

PD:	Регистр назначения для памяти программы	DD:	Регистр назначения для памяти данных
AMF:	Операция АЛУ/умножителя-накопителя	M:	Регистр модификации
Yop:	Операнд Y	Xop:	Операнд X
I:	Регистр косвенного адреса		