УТВЕРЖДАЮ

РАЯЖ.00150-01 93 01-1-ЛУ

DSP-КЛАСТЕР DELCORE-30M. АРХИТЕКТУРА
DSP-ЯДРО ELCORE-30M. СИСТЕМА ИНСТРУКЦИЙ

РАЯЖ.00150-01 93 01-1

CD-R

Листов 125

2010

Подп. и дата	
Инв.№ дубл.	
Взам.инв.№	
Подп. и дата	
Инв. № подл.	

АННОТАЦИЯ

Настоящий документ содержит описание системы инструкций ядра процессора сигнальной обработки (DSP-ядра) Elcore-30M, входящего в состав DSP-кластера DElcore-30M.

СОДЕРЖАНИЕ

1. Введение. Структура DSP-кластера DElcore-30M

DSP-ядро Elcore-30M является составной частью DSP-кластера DElcore-30M. DSP-ядро Elcore-30M представляет собой ядро сопроцессора-акселератора сигнальной обработки. Оно имеет гарвардскую архитектуру с внутренним параллелизмом по потокам обрабатываемых данных и предназначено для обработки информации в форматах с фиксированной и с плавающей точкой. Система инструкций, реализующих параллельно несколько вычислительных операций и пересылок, семифазный программный конвейер и гибкие адресные режимы позволяют реализовать алгоритмы сигнальной обработки с высокой производительностью.

1.1 Основные технические характеристики DSP-кластера DElcore-30M

1.1.1 DSP-кластер DElcore-30M содержит два DSP-ядра Elcore-30M (DSP0 – DSP1), работающих на общем поле памяти данных, а также набор общих для всего кластера регистров управления/состояния и буфер обмена XBUF. Каждое DSP-ядро функционирует под управлением RISC-ядра (CPU) и расширяет его возможности по обработке сигналов.

- 2 вычислительных DSP-ядра Elcore-30M;
- объем общей памяти данных 256 Кбайт (128 Кбайт на ядро);
- объем памяти программ 32 Кбайт на ядро;
- максимальная пропускная способность коммутатора ядер с памятью – 512 бит за такт;
- максимальная скорость обмена внешних устройств с памятью кластера – 64 бит за такт;
- суммарная пиковая производительность кластера:

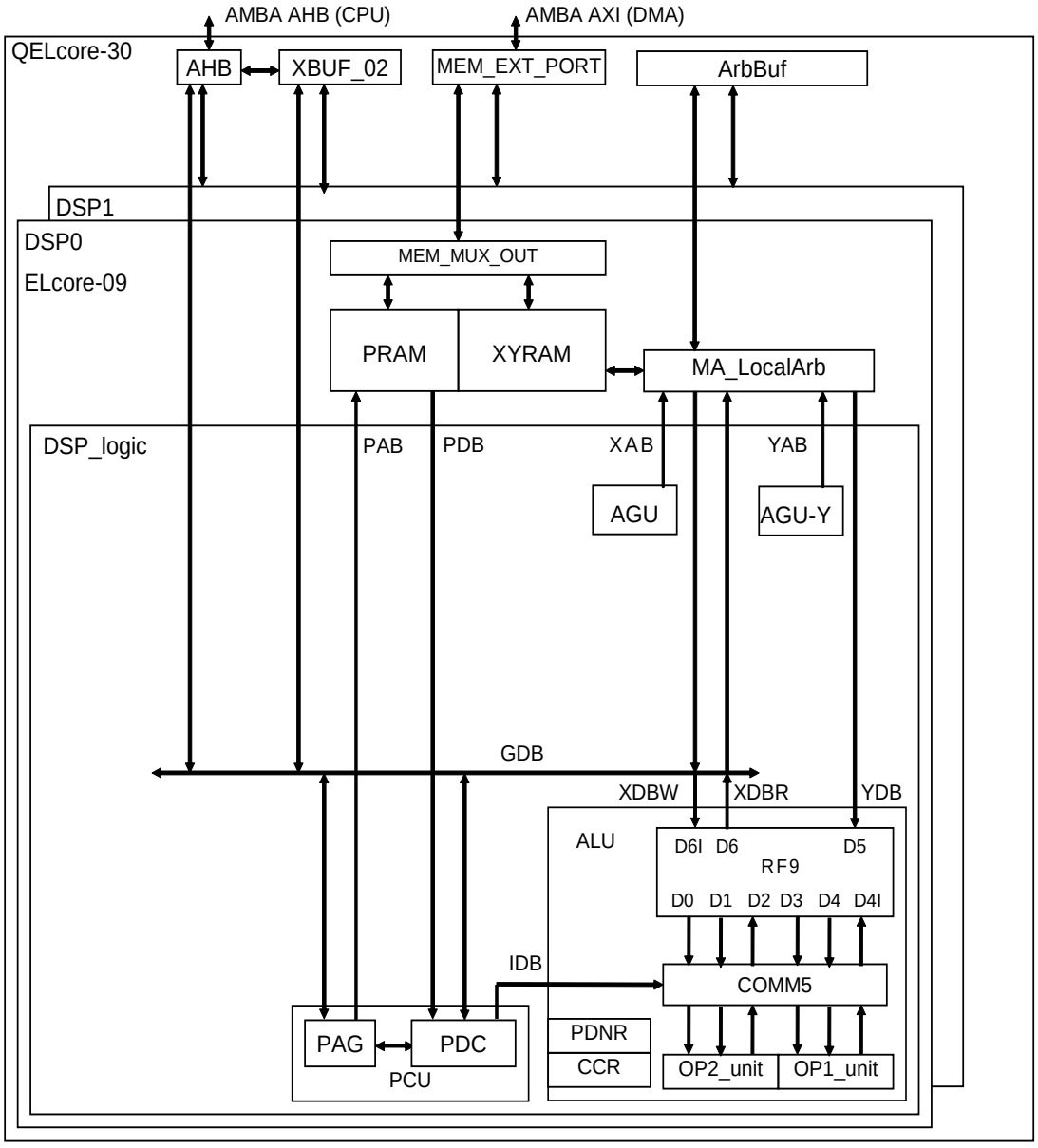
√ 16 операции с плавающей точкой (IEEE 754) за такт;

√ 16 32-битных операций с фиксированной точкой за такт;

√ 48 16-битных операций с фиксированной точкой за такт.

1.2 Структурная схема DSP-кластера

1.2.1 Структурная схема двухъядерного DSP-кластера DElcore-30M приведена на рисунке 1.1.



Примечание. На схеме приняты следующие обозначения:

DSP0 – DSP1 – DSP-ядра Elcore-30M;
PRAM – память программ;
XYRAM – память данных;
AHB – контроллер шины AMBA AHB (slave);
XBUF_04 – буфер обмена (регистровый файл 32 слова по 64 разряда, 5 портов);
ArbBuf, MA_LocalArb – распределенный арбитр;
AGU, AGU-Y – адресные генераторы памяти данных;
PAG – адресный генератор памяти программ;
PDC_17 – программный декодер;
RF9 – регистровый файл 16 слов по 128 разрядов, 9 портов;
COMM5 – коммутатор входных данных операционных устройств;
OP1_unit, OP2_unit – операционные (вычислительные) устройства;
DSP_logic – вычислительное ядро;
CCR_REG, PDN – регистры признаков результата операции и параметра денормализации;

Рисунок 1.1 ─ Структурная схема 2-ядерного DSP-кластера DElcore-30M

1.3 Организация работы DSP-кластера

1.3.1 Кластер DSP представляет собой двухпроцессорную MIMD систему. Каждое DSP ядро обладает собственной программной памятью, и может работать независимо от остальных ядер (DSP и CPU). На верхнем уровне DSP-кластера имеется набор общих для всего кластера регистров управления и состояния.

Для синхронизации работы DSP ядер в кластере предусмотрено два механизма: механизм прерываний и механизм обменов через XBUF в синхронном режиме.

Каждое DSP ядро может сформировать прерывание для любого другого ядра в кластере. Ядро, получившее прерывание, переходит в состояние RUN, если было остановлено, и начинает исполнение подпрограммы, адрес которой храниться в специальном регистре этого ядра.

Для оперативных обменов данными между CPU, DSP0 – DSP1 в составе DElcore-30M имеется буфер обмена XBUF, состоящий из 32-х 64-разрядных регистров X0-X31, доступных по записи и чтению для всех процессорных ядер.

Буфер обмена XBUF представляет собой многопортовую память и допускает одновременное чтение одной и той же ячейки со стороны нескольких абонентов - CPU, DSP0 – DSP1. При одновременном запросе на запись в одну и ту же ячейку приоритет отдается CPU, затем - DSP0, затем - DSP1.

Обменный буфер может работать в обычном режиме, когда при обмене данными через него не происходит никаких блокировок и в синхронном режиме. В синхронном режиме для конкретного регистра XBUF обязательно должны чередоваться операции чтения записи, если какое либо ядро пытается осуществить запись после записи или чтение после чтения – оно блокируется. Обмен через XBUF в синхронном режиме является дополнительным программным способом синхронизации ядер DSP.

Программная память и память данных кластера DSP физически организована как двухпортовая. По одному порту производятся внешние обращения от RISC ядра и контроллеров DMA, по другому порту производятся обращения от ядер DSP. Такая организация позволяет производить бесконфликтный фоновый обмен данными между памятью кластера DSP и внешними устройствами.

1.4 Карта памяти DSP-кластера

1.4.1 Карта памяти кластера DSP приведена на рисунке 1.2.

Каждое из DSP-ядер имеет свою программную память (PRAM) объемом 32 Кбайт и общую для всех память данных XYRAM объемом 256 Кбайт.

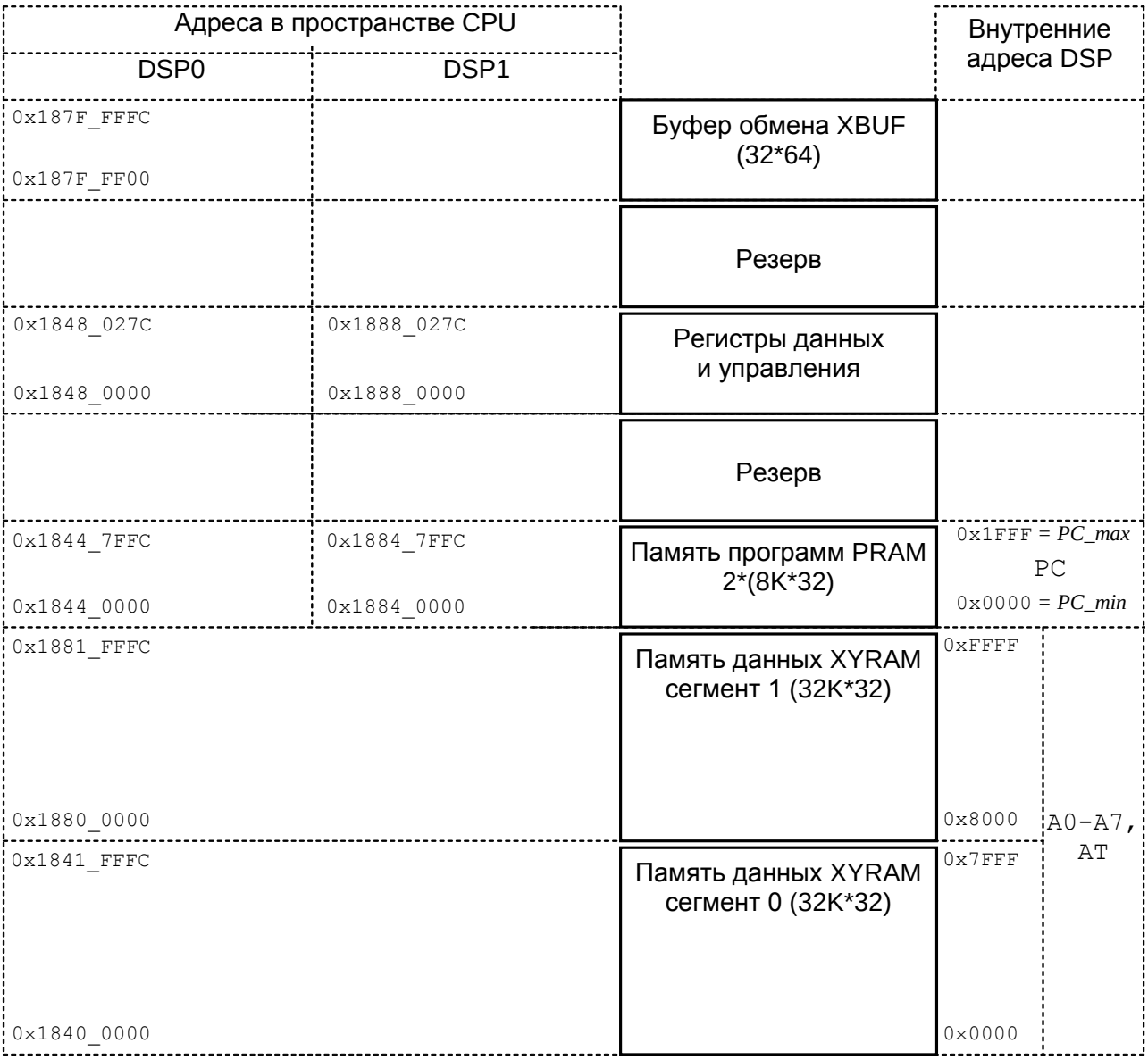


Рисунок 1.2 ─ Карта памяти DSP-кластера при BASEx(0)=0

Объем PRAM (DSP0) – 8K 32-разрядных слов (32 Кбайт).

Объем PRAM (DSP1) – 8K 32-разрядных слов (32 Кбайт).

Объем XYRAM – 64K 32-разрядных слов (256 Кбайт), (память XYRAM состоит из двух сегментов по 32K 32-разрядных слов, относящихся к соответствующему DSP-ядру).

1.5 Регистры управления и состояния DSP-кластера DElcore-30M

На верхнем уровне DSP-кластера имеются 4 регистра управления и состояния, доступ к которым осуществляется только со стороны CPU. Назначение и адреса этих регистров приведены в таблице 1.1.

Таблица 1.1. ─ Назначение и адреса регистров управления и состояния DSP-кластера

Условное обозначение	Разряд-ность	Тип	Назначение регистра	Адрес регистра
			Регистры управления и состояния	
MASKR_DSP	32	R/W	Регистр маски прерываний	0x1848_1000
QSTR_DSP	32	R	Регистр запросов прерываний	0x1848_1004
CSR_DSP	32	R/W	Регистр управления и состояния	0x1848_1008
TOTAL_RUN_CNTR	32	R/W	Счетчик тактов в состоянии RUN	0x1848_100C
TOTAL_CLK_CNTR	32	R/W	Счетчик тактов	0x1848_1010

1.5.1 Регистр маски прерываний (MASKR_DSP)

Регистр маски прерываний MASKR_DSP содержит 32 разряда, каждый из которых разрешает («1») либо запрещает («0») запрос на прерывание в CPU от соответствующего разряда регистра запросов прерываний QSTR_DSP. Регистр доступен по чтению и записи. Начальное состояние регистра MASKR_DSP=0x0.

1.5.2 Регистр запросов прерываний (QSTR_DSP)

Регистр запросов прерываний QSTR_DSP доступен только по чтению и содержит флаги запросов прерываний от DSP-ядер. Назначение разрядов регистра QSTR_DSP приведено в таблице 1.2. Для сброса флагов прерывания необходимо обнулить соответствующие разряды регистров DCSR, входящих в состав DSP0-DSP1.

Таблица 1.2 ─ Назначение разрядов регистра QSTR_DSP

Номер разряда	Наименование разряда	Назначение
0	PI0	Программное прерывание DSP0
1	SE0	Прерывание по ошибке стека DSP0
2	BREAK0	Прерывание по останову BREAK DSP0
3	STP0	Прерывание по останову STOP DSP0
4-7	-	Резерв
8	PI1	Программное прерывание DSP1
9	SE1	Прерывание по ошибке стека DSP1
10	BREAK1	Прерывание по останову BREAK DSP1
11	STP1	Прерывание по останову STOP DSP1
12-27	-	Резерв
28	WAIT	Прерывание по состоянию ожидания DSP0 – DSP1
29-31	-	Резерв

Начальное состояние регистра QSTR_DSP=0x0.

1.5.3 Регистр управления и состояния (CSR_DSP)

Регистр управления и состояния CSR_DSP доступен по чтению и записи и содержит биты управления кластером DSP-ядер. Назначение разрядов регистра CSR_DSP приведено в таблице 1.3.

Таблица 1.3 ─ Назначение разрядов регистра CSR_DSP

Номер разряда	Наименование разряда	Назначение
0	SYNSTART	Одновременный старт DSP0 – DSP1
1	SYNWORK	Работа XBUF в синхронном режиме
2-3	PMCONFIG	Конфигурация программной памяти
4-15	-	Резерв
16	HEN	Включение режима определения высокой плотности потоков
17	DEN	Разрешение установки явного приоритета (статический режим)
18	LEN	Бит разрешения ограничителя
21:20	DPTR	Номер ядра, обладающего наивысшим приоритетом
29:24	Limit	Максимальное значение счетчика обращений
25-31	-	Резерв

Начальное состояние регистра CSR_DSP=0x0.

Запись «1» в разряд SYNSTART приводит к одновременному запуску четырёх DSP–ядер. При этом в регистрах DCSR каждого из DSP–ядер бит RUN устанавливается в «1», состояние других разрядов не изменяется. Запись «1» в разряд SYNWORK устанавливает буфер обмена XBUF в синхронный режим.

Арбитраж.

Для управления арбитражем обращений от различных DSP ядер в регистр CSR_DSP введены дополнительные разряды HEN, DEN, LEN, DPTR, Limit.

Более подробно данные биты описаны в п.2.3.17.2.

1.5.4 Счетчик тактов (TOTAL_RUN_CNTR)

32-разрядный счетчик тактов (TOTAL_ RUN _CNTR) выполняет подсчет числа тактов, в течение которых хотя бы одно из DSP-ядер находилось в состоянии RUN. Любая запись в данный счетчик приводит к его обнулению.

Начальное состояние счетчика тактов также равно нулю: TOTAL_ RUN_CNTR = 0x0.

1.5.5 Счетчик тактов (TOTAL_CLK_CNTR)

32-разрядный счетчик тактов (TOTAL_CLK_CNTR) выполняет подсчет числа тактов. Любая запись в данный счетчик приводит к его обнулению.

Начальное состояние счетчика тактов также равно нулю: TOTAL_CLK_CNTR = 0x0.

1.5.6 Буфер обмена XBUF и регистр флагов обмена EFR

Для оперативных обменов данными между CPU, DSP0 – DSP1 в составе DSP-кластера имеется буфер обмена XBUF, состоящий из 32 64-разрядных регистров X0-X31, доступных по записи и чтению для всех процессорных ядер. Регистры буфера обмена XBUF доступны для CPU и DSP-ядер. Адреса регистров XBUF приведены в таблице 1.6. Начальное состояние регистров XBUF не определено.

Таблица 1.6 ─ Адреса регистров XBUF

Условное обозначение	Разрядность, тип	Назначение регистра	Адрес регистра
		Регистры буфера обмена XBUF	
X0[31:0]	32 R/W	Регистр обмена X0	0x187F_FF00
X0[63:32]	32 R/W	Регистр обмена X0	0x187F_FF04
X1[31:0]	32 R/W	Регистр обмена X1	0x187F_FF08
X1[63:32]	32 R/W	Регистр обмена X1	0x187F_FF0C
X2[31:0]	32 R/W	Регистр обмена X2	0x187F_FF10
X2[63:32]	32 R/W	Регистр обмена X2	0x187F_FF14
X3[31:0]	32 R/W	Регистр обмена X3	0x187F_FF18
X3[63:32]	32 R/W	Регистр обмена X3	0x187F_FF1C
X4[31:0]	32 R/W	Регистр обмена X4	0x187F_FF20
X4[63:32]	32 R/W	Регистр обмена X4	0x187F_FF24
X5[31:0]	32 R/W	Регистр обмена X5	0x187F_FF28
X5[63:32]	32 R/W	Регистр обмена X5	0x187F_FF2C
X6[31:0]	32 R/W	Регистр обмена X6	0x187F_FF30
X6[63:32]	32 R/W	Регистр обмена X6	0x187F_FF34
X7[31:0]	32 R/W	Регистр обмена X7	0x187F_FF38
X7[63:32]	32 R/W	Регистр обмена X7	0x187F_FF3C
X8[31:0]	32 R/W	Регистр обмена X8	0x187F_FF40
X8[63:32]	32 R/W	Регистр обмена X8	0x187F_FF44
X9[31:0]	32 R/W	Регистр обмена X9	0x187F_FF48
X9[63:32]	32 R/W	Регистр обмена X9	0x187F_FF4C
X10[31:0]	32 R/W	Регистр обмена X10	0x187F_FF50
X10[63:32]	32 R/W	Регистр обмена X10	0x187F_FF54
X11[31:0]	32 R/W	Регистр обмена X11	0x187F_FF58
X11[63:32]	32 R/W	Регистр обмена X11	0x187F_FF5C
X12[31:0]	32 R/W	Регистр обмена X12	0x187F_FF60
X12[63:32]	32 R/W	Регистр обмена X12	0x187F_FF64
X13[31:0]	32 R/W	Регистр обмена X13	0x187F_FF68
X13[63:32]	32 R/W	Регистр обмена X13	0x187F_FF6C
X14[31:0]	32 R/W	Регистр обмена X14	0x187F_FF70
X14[63:32]	32 R/W	Регистр обмена X14	0x187F_FF74
X15[31:0]	32 R/W	Регистр обмена X15	0x187F_FF78
X15[63:32]	32 R/W	Регистр обмена X15	0x187F_FF7C
X16[31:0]	32 R/W	Регистр обмена X16	0x187F_FF80
X16[63:32]	32 R/W	Регистр обмена X16	0x187F_FF84
X17[31:0]	32 R/W	Регистр обмена X17	0x187F_FF88
X17[63:32]	32 R/W	Регистр обмена X17	0x187F_FF8C
X18[31:0]	32 R/W	Регистр обмена X18	0x187F_FF90
X18[63:32]	32 R/W	Регистр обмена X18	0x187F_FF94
X19[31:0]	32 R/W	Регистр обмена X19	0x187F_FF98
X19[63:32]	32 R/W	Регистр обмена X19	0x187F_FF9C

Окончание таблицы 1.6

Условное обозначение	Разрядность, тип	Назначение регистра	Адрес регистра
X20[31:0]	32 R/W	Регистр обмена X20	0x187F_FFA0
X20[63:32]	32 R/W	Регистр обмена X20	0x187F_FFA4
X21[31:0]	32 R/W	Регистр обмена X21	0x187F_FFA8
X21[63:32]	32 R/W	Регистр обмена X21	0x187F_FFAC
X22[31:0]	32 R/W	Регистр обмена X22	0x187F_FFB0
X22[63:32]	32 R/W	Регистр обмена X22	0x187F_FFB4

X23[31:0]	32 R/W	Регистр обмена X23	0x187F_FFB8
X23[63:32]	32 R/W	Регистр обмена X23	0x187F_FFBC
X24[31:0]	32 R/W	Регистр обмена X24	0x187F_FFC0
X24[63:32]	32 R/W	Регистр обмена X24	0x187F_FFC4
X25[31:0]	32 R/W	Регистр обмена X25	0x187F_FFC8
X25[63:32]	32 R/W	Регистр обмена X25	0x187F_FFCC
X26[31:0]	32 R/W	Регистр обмена X26	0x187F_FFD0
X26[63:32]	32 R/W	Регистр обмена X26	0x187F_FFD4
X27[31:0]	32 R/W	Регистр обмена X27	0x187F_FFD8
X27[63:32]	32 R/W	Регистр обмена X27	0x187F_FFDC
X28[31:0]	32 R/W	Регистр обмена X28	0x187F_FFE0
X28[63:32]	32 R/W	Регистр обмена X28	0x187F_FFE4
X29[31:0]	32 R/W	Регистр обмена X29	0x187F_FFE8
X29[63:32]	32 R/W	Регистр обмена X29	0x187F_FFEC
X30[31:0]	32 R/W	Регистр обмена X30	0x187F_FFF0
X30[63:32]	32 R/W	Регистр обмена X30	0x187F_FFF4
X31[31:0]	32 R/W	Регистр обмена X31	0x187F_FFF8
X31[63:32]	32 R/W	Регистр обмена X31	0x187F_FFFC

Буфер обмена XBUF представляет собой многопортовую память и допускает одновременное чтение одной и той же ячейки со стороны нескольких абонентов - CPU, DSP0 – DSP1. При одновременном запросе на запись в одну и ту же ячейку приоритет отдается CPU, затем - DSP0, затем - DSP1.

1.5.7 Регистр флагов обмена (EFR)

Регистр флагов обмена (EFR) является общим для всего кластера DSP и предназначен для отображения флагов обменов через буфер XBUF. Регистр EFR содержит 32 бита, доступных только по чтению каждому из DSP-ядер и CPU, начальное состояние EFR=0x0.

Каждый разряд этого регистра формируется аппаратно и отображает тип последней транзакции, выполненной с соответствующей ячейкой XBUF (0 – чтение из XBUF, 1 – запись). Заметим, что при 8/16/32-разрядных обращениях со стороны CPU изменение состояния EFR происходит только при обращении к младшему байту 64-разрядной ячейки XBUF.

1.5.8 Режимы обменов с XBUF

Имеются два режима обменов с XBUF – обычный и синхронный.

В обычном режиме (устанавливается битом 1 регистра CSR_DSP SYNWORK=0) любой из абонентов - CPU, DSP0 – DSP1 - в любое время может обращаться к любой ячейке XBUF, и это обращение немедленно исполняется (с учетом приоритета по записи).

В синхронном режиме (устанавливается битом 1 регистра CSR_DSP SYNWORK=1):

- CPU обращается к XBUF так же, как и в обычном режиме;

- обращения со стороны DSP0 – DSP1 могут выполняться с задержкой в зависимости от состояния регистра EFR и типа обращения. Если тип обращения не совпадает с типом последней транзакции, выполненной с данной ячейкой XBUF (то есть если за записью следует чтение, а за чтением - запись) то исполнение такого обращения происходит без задержки. Если же за записью вновь следует запрос на запись в ту же ячейку (либо за чтением – вновь запрос на чтение), то такое обращение выполняется с задержкой. Выдавшее запрос DSP переводится в со-

стояние ожидания, продолжающееся до тех пор, пока соответствующий бит EFR не сменит свое значение на противоположное.

В регистре DCSR каждого DSP-ядра имеется бит WT=DCSR[4], указывающий на то, что DSP находится в состоянии ожидания при обращении к XBUF.

1.6 Перечень адресуемых регистров DSP-ядер Elcore-30M

1.6.1 Перечень адресуемых регистров DSP-ядер в составе DSP-кластера DElcore-30M с указанием начального состояния приведен в таблице 1.7.

Таблица 1.7. ─ Перечень адресуемых регистров DSP-ядер (i=0,1 – номер DSP-ядра)
BASE(0)=0x1848_0000; BASE(1)=0x1888_0000)

Наименование	Бит	Нач. сост.	Тип	Назначение регистра	Адрес регистра
				PCU	
DCSR	16	0	R/W	Регистр режима работы	BASE(i)+0x0100
SR	16	0	R/W	Регистр состояния	BASE(i)+0x0104
IDR	16	0xn108	R	Регистр-идентификатор	BASE(i)+0x0108
EFR	32	0	R	Регистр флагов обмена	BASE(i)+0x010C
DSTART	32	0	W	Регистр запуска DMA со стороны DSP и запросов на прерывания других DSP	BASE(i)+0x010C
IRQR	32	0	R/W	Регистр запросов на прерывание DSP	BASE(i)+0x0110
IMASKR	32	0	R/W	Регистр маски запросов на прерывание DSP	BASE(i)+0x0114
TMR	32	0	R/W	Регистр таймера DSP	BASE(i)+0x0118
ARBR	16	0x0F01	R/W	Регистр управления арбитром памяти DSP	BASE(i)+0x011C
PC	16	0	R/W	Программный счетчик	BASE(i)+0x0120
SS	16	0	R/W	Стек программного счетчика	BASE(i)+0x0124
LA	16	0xFFFF	R/W	Регистр адреса цикла	BASE(i)+0x0128
CSL	16	0	R/W	Стек адреса цикла	BASE(i)+0x012C
LC	16	0	R/W	Счетчик циклов	BASE(i)+0x0130
CSH	16	0	R/W	Стек счетчика циклов	BASE(i)+0x0134
SP	16	0	R/W	Регистр указателя стека	BASE(i)+0x0138
SAR	16	0xFFFF	R/W	Регистр адреса останова	BASE(i)+0x013C
CNTR	16	0	R/W	Счетчик исполненных команд	BASE(i)+0x0140
SAR1	16	0xFFFF	R/W	Регистр адреса останова	BASE(i)+0x0144
SAR2	16	0xFFFF	R/W	Регистр адреса останова	BASE(i)+0x0148
SAR3	16	0xFFFF	R/W	Регистр адреса останова	BASE(i)+0x014C
SAR4	16	0xFFFF	R/W	Регистр адреса останова	BASE(i)+0x0150
SAR5	16	0xFFFF	R/W	Регистр адреса останова	BASE(i)+0x0154
SAR6	16	0xFFFF	R/W	Регистр адреса останова	BASE(i)+0x0158
SAR7	16	0xFFFF	R/W	Регистр адреса останова	BASE(i)+0x015C
				Регистры состояния ALU	
CCR	16	0	R/W	Регистр кодов условий	BASE(i)+0x0160
PDNR	16	0	R/W	Регистр параметра денормализации	BASE(i)+0x0164
SFR	32	0	R/W	Регистр специальных функций	BASE(i)+0x0168
QMASKR0	32	0	R/W	Регистр маски запросов на прерывание со стороны CPU (QSTR0)	BASE(i)+0x0170
QMASKR1	32	0	R/W	Регистр маски запросов на прерывание со стороны CPU (QSTR1)	BASE(i)+0x0174
QMASKR2	32	0	R/W	Регистр маски запросов на прерывание со стороны CPU (QSTR2)	BASE(i)+0x0178
				AGU, AGU-Y	

Наименование	Бит	Нач. сост.	Тип	Назначение регистра	Адрес регистра
A0	32	см.п.2.6	R/W	Регистр адреса A0	BASE(i)+0x0080
A1	32	см.п.2.6	R/W	Регистр адреса A1	BASE(i)+0x0084
A2	32	см.п.2.6	R/W	Регистр адреса A2	BASE(i)+0x0088
A3	32	см.п.2.6	R/W	Регистр адреса A3	BASE(i)+0x008C
A4	32	см.п.2.6	R/W	Регистр адреса A4	BASE(i)+0x0090
A5	32	см.п.2.6	R/W	Регистр адреса A5	BASE(i)+0x0094
A6	32	см.п.2.6	R/W	Регистр адреса A6	BASE(i)+0x0098
A7	32	см.п.2.6	R/W	Регистр адреса A7	BASE(i)+0x009C
I0	16	0	R/W	Регистр индекса I0	BASE(i)+0x00A0
I1	16	0	R/W	Регистр индекса I1	BASE(i)+0x00A4
I2	16	0	R/W	Регистр индекса I2	BASE(i)+0x00A8
I3	16	0	R/W	Регистр индекса I3	BASE(i)+0x00AC
I4	16	0	R/W	Регистр индекса I4	BASE(i)+0x00B0
I5	16	0	R/W	Регистр индекса I5	BASE(i)+0x00B4
I6	16	0	R/W	Регистр индекса I6	BASE(i)+0x00B8
I7	16	0	R/W	Регистр индекса I7	BASE(i)+0x00BC
M0	16	0xFFFF	R/W	Регистр модификатора M0	BASE(i)+0x00C0
M1	16	0xFFFF	R/W	Регистр модификатора M1	BASE(i)+0x00C4
M2	16	0xFFFF	R/W	Регистр модификатора M2	BASE(i)+0x00C8
M3	16	0xFFFF	R/W	Регистр модификатора M3	BASE(i)+0x00CC
M4	16	0xFFFF	R/W	Регистр модификатора M4	BASE(i)+0x00D0
M5	16	0xFFFF	R/W	Регистр модификатора M5	BASE(i)+0x00D4
M6	16	0xFFFF	R/W	Регистр модификатора M6	BASE(i)+0x00D8
M7	16	0xFFFF	R/W	Регистр модификатора M7	BASE(i)+0x00DC
AT	32	см.п.2.6	R/W	Регистр адреса AT	BASE(i)+0x00E0
IT	16	0	R/W	Регистр индекса IT	BASE(i)+0x00E4
MT	16	0xFFFF	R/W	Регистр модификатора MT	BASE(i)+0x00E8
DT	16	0	R/W	Регистр модификатора DT	BASE(i)+0x00EC
IVAR	16	0x1F00	R/W	Регистр адреса вектора прерывания	BASE(i)+0x00FC
				Регистры данных RF	
R0.L	32	X	R/W	Регистр данных	BASE(i)+0x0000
R2.L	32	X	R/W	Регистр данных	BASE(i)+0x0004
R4.L	32	X	R/W	Регистр данных	BASE(i)+0x0008
R6.L	32	X	R/W	Регистр данных	BASE(i)+0x000C
R8.L	32	X	R/W	Регистр данных	BASE(i)+0x0010
R10.L	32	X	R/W	Регистр данных	BASE(i)+0x0014
R12.L	32	X	R/W	Регистр данных	BASE(i)+0x0018
R14.L	32	X	R/W	Регистр данных	BASE(i)+0x001C
R16.L	32	X	R/W	Регистр данных	BASE(i)+0x0020
R18.L	32	X	R/W	Регистр данных	BASE(i)+0x0024
R20.L	32	X	R/W	Регистр данных	BASE(i)+0x0028
R22.L	32	X	R/W	Регистр данных	BASE(i)+0x002C
R24.L	32	X	R/W	Регистр данных	BASE(i)+0x0030
R26.L	32	X	R/W	Регистр данных	BASE(i)+0x0034
R28.L	32	X	R/W	Регистр данных	BASE(i)+0x0038
R30.L	32	X	R/W	Регистр данных	BASE(i)+0x003C
R1.L	32	X	R/W	Регистр данных	BASE(i)+0x0040
R3.L	32	X	R/W	Регистр данных	BASE(i)+0x0044
R5.L	32	X	R/W	Регистр данных	BASE(i)+0x0048
R7.L	32	X	R/W	Регистр данных	BASE(i)+0x004C
R9.L	32	X	R/W	Регистр данных	BASE(i)+0x0050
R11.L	32	X	R/W	Регистр данных	BASE(i)+0x0054
R13.L	32	X	R/W	Регистр данных	BASE(i)+0x0058
R15.L	32	X	R/W	Регистр данных	BASE(i)+0x005C

Наименование	Бит	Нач. сост.	Тип	Назначение регистра	Адрес регистра
R17.L	32	X	R/W	Регистр данных	BASE(i)+0x0060
R19.L	32	X	R/W	Регистр данных	BASE(i)+0x0064
R21.L	32	X	R/W	Регистр данных	BASE(i)+0x0068
R23.L	32	X	R/W	Регистр данных	BASE(i)+0x006C
R25.L	32	X	R/W	Регистр данных	BASE(i)+0x0070
R27.L	32	X	R/W	Регистр данных	BASE(i)+0x0074
R29.L	32	X	R/W	Регистр данных	BASE(i)+0x0078
R31.L	32	X	R/W	Регистр данных	BASE(i)+0x007C
R1.D[31:0]	32	X	R/W	Регистр данных	BASE(i)+0x0180
R1.D[63:32]	32	X	R/W	Регистр данных	BASE(i)+0x0184
R3.D[31:0]	32	X	R/W	Регистр данных	BASE(i)+0x0188
R3.D[63:32]	32	X	R/W	Регистр данных	BASE(i)+0x018C
R5.D[31:0]	32	X	R/W	Регистр данных	BASE(i)+0x0190
R5.D[63:32]	32	X	R/W	Регистр данных	BASE(i)+0x0194
R7.D[31:0]	32	X	R/W	Регистр данных	BASE(i)+0x0198
R7.D[63:32]	32	X	R/W	Регистр данных	BASE(i)+0x019C
R9.D[31:0]	32	X	R/W	Регистр данных	BASE(i)+0x01A0
R9.D[63:32]	32	X	R/W	Регистр данных	BASE(i)+0x01A4
R11.D[31:0]	32	X	R/W	Регистр данных	BASE(i)+0x01A8
R11.D[63:32]	32	X	R/W	Регистр данных	BASE(i)+0x01AC
R13.D[31:0]	32	X	R/W	Регистр данных	BASE(i)+0x01B0
R13.D[63:32]	32	X	R/W	Регистр данных	BASE(i)+0x01B4
R15.D[31:0]	32	X	R/W	Регистр данных	BASE(i)+0x01B8
R15.D[63:32]	32	X	R/W	Регистр данных	BASE(i)+0x01BC
R17.D[31:0]	32	X	R/W	Регистр данных	BASE(i)+0x01C0
R17.D[63:32]	32	X	R/W	Регистр данных	BASE(i)+0x01C4
R19.D[31:0]	32	X	R/W	Регистр данных	BASE(i)+0x01C8
R19.D[63:32]	32	X	R/W	Регистр данных	BASE(i)+0x01CC
R21.D[31:0]	32	X	R/W	Регистр данных	BASE(i)+0x01D0
R21.D[63:32]	32	X	R/W	Регистр данных	BASE(i)+0x01D4
R23.D[31:0]	32	X	R/W	Регистр данных	BASE(i)+0x01D8
R23.D[63:32]	32	X	R/W	Регистр данных	BASE(i)+0x01DC
R25.D[31:0]	32	X	R/W	Регистр данных	BASE(i)+0x01E0
R25.D[63:32]	32	X	R/W	Регистр данных	BASE(i)+0x01E4
R27.D[31:0]	32	X	R/W	Регистр данных	BASE(i)+0x01E8
R27.D[63:32]	32	X	R/W	Регистр данных	BASE(i)+0x01EC
R29.D[31:0]	32	X	R/W	Регистр данных	BASE(i)+0x01F0
R29.D[63:32]	32	X	R/W	Регистр данных	BASE(i)+0x01F4
R31.D[31:0]	32	X	R/W	Регистр данных	BASE(i)+0x01F8
R31.D[63:32]	32	X	R/W	Регистр данных	BASE(i)+0x01FC
				Регистры-аккумуляторы	
AC0	32	0	R/W	Регистр-аккумулятор AC0	BASE(i)+0x0200
AC1	32	0	R/W	Регистр-аккумулятор AC1	BASE(i)+0x0204
AC2	32	0	R/W	Регистр-аккумулятор AC2	BASE(i)+0x0208
AC3	32	0	R/W	Регистр-аккумулятор AC3	BASE(i)+0x020C
AC4	32	0	R/W	Регистр-аккумулятор AC4	BASE(i)+0x0210
AC5	32	0	R/W	Регистр-аккумулятор AC5	BASE(i)+0x0214
AC6	32	0	R/W	Регистр-аккумулятор AC6	BASE(i)+0x0218
AC7	32	0	R/W	Регистр-аккумулятор AC7	BASE(i)+0x021C
AC8	32	0	R/W	Регистр-аккумулятор AC8	BASE(i)+0x0220
AC9	32	0	R/W	Регистр-аккумулятор AC9	BASE(i)+0x0224
AC10	32	0	R/W	Регистр-аккумулятор AC10	BASE(i)+0x0228
AC11	32	0	R/W	Регистр-аккумулятор AC11	BASE(i)+0x022C
AC12	32	0	R/W	Регистр-аккумулятор AC12	BASE(i)+0x0230

Наименование	Бит	Нач. сост.	Тип	Назначение регистра	Адрес регистра
AC13	32	0	R/W	Регистр-аккумулятор AC13	BASE(i)+0x0234
AC14	32	0	R/W	Регистр-аккумулятор AC14	BASE(i)+0x0238
AC15	32	0	R/W	Регистр-аккумулятор AC15	BASE(i)+0x023C
				Отладочные регистры	
dbDCSR	16	0	R/W	Регистр управления в режиме отладки	BASE(i)+0x0500
SMASKR	32	0	R/W	Регистр маски условий останова	BASE(i)+0x0514
Cnt_RUN	32	0	R	Счетчик тактов	BASE(i)+0x0518
dbPCa	16	0	R	Программный счетчик, стадия a	BASE(i)+0x0524
dbPCf	16	0	R	Программный счетчик, стадия f	BASE(i)+0x0528
dbPCd	16	0	R	Программный счетчик, стадия d	BASE(i)+0x052C
dbPCe	16	0	R	Программный счетчик, стадия e	BASE(i)+0x0520
dbPCe1	16	0	R	Программный счетчик, стадия e1	BASE(i)+0x0530
dbPCe2	16	0	R	Программный счетчик, стадия e2	BASE(i)+0x0534
dbPCe3	16	0	R	Программный счетчик, стадия e3	BASE(i)+0x0538
dbSAR	16	0xFFFF	R/W	Регистр адреса останова 0 в режиме отладки	BASE(i)+0x053C
dbCNTR	16	0	R/W	Счетчик исполненных команд в режиме отладки	BASE(i)+0x0540
dbSAR1	16	0xFFFF	R/W	Регистр адреса останова 1 в режиме отладки	BASE(i)+0x0544
dbSAR2	16	0xFFFF	R/W	Регистр адреса останова 2 в режиме отладки	BASE(i)+0x0548
dbSAR3	16	0xFFFF	R/W	Регистр адреса останова 3 в режиме отладки	BASE(i)+0x054C
dbSAR4	16	0xFFFF	R/W	Регистр адреса останова 4 в режиме отладки	BASE(i)+0x0550
dbSAR5	16	0xFFFF	R/W	Регистр адреса останова 5 в режиме отладки	BASE(i)+0x0554
dbSAR6	16	0xFFFF	R/W	Регистр адреса останова 6 в режиме отладки	BASE(i)+0x0558
dbSAR7	16	0xFFFF	R/W	Регистр адреса останова 7 в режиме отладки	BASE(i)+0x055C

1.7 Доступ DSP кластера к ресурсам процессора

1.7.1 Каждое DSP ядро может обращаться к ресурсам процессора (внешняя и внутренняя памяти, регистры, периферия).

В целях совместимости адресация внутренней памяти DSP кластера не изменена.

Адресное пространство DSP находится в диапазоне адресов 0x00000000 – 0x000FFFFF при пословной адресации, которая применяется в ядрах DSP, что соответствует диапазону 0x00000000 – 0x003FFFFC при побайтовой адресации, используемой в адресном пространстве всей системы на кристалле.

Таким образом, обращаясь к адресам адресного пространства DSP (0x00000000 – 0x000FFFFF - пословная) ядро выполняет обращение к внутренней памяти кластера. В этом случае обращения в зависимости от адреса и номера DSP ядра могут направляться либо в

ближний сегмент памяти данного ядра (быстрые обращения), либо в дальний сегмент памяти другого ядра (обращения через коммутатор кластера).

При обращениях к старшим адресам адресного пространства, лежащим вне адресного пространства DSP (0x000FFFFF - 0xFFFFFFFF - пословная), обращение от DSP ядра перенаправляется на глобальный коммутатор AXI и может быть направлено к любому адресуемому регистру или ячейке памяти, за исключением диапазона 0x00000000 – 0x003FFFFC (адреса полностью соответствуют карте памяти RISC ядра). Важной особенностью внешних обращений DSP, о которой необходимо помнить программисту, является тот факт, что при переходе из адресного пространства DSP с пословной адресацией в глобальное пространство с побайтовой адресацией выполняется аппаратный сдвиг значения адресного указателя на 2 бита влево. Так, например обращение DSP ядра по значению A0 = 0x2ff00001 приведет к обращению по физическому адресу 0xbfc00004.

(DSP адресует память 32-х разрядными словами, поэтому реальный физический адрес внешнего обращения получается сдвигом влево на два разряда текущего значения адресного указателя).

Весь DSP кластер является одним мастером для шины AXI (все ядра кластера выполняют внешние обращения через один общий порт), таким образом, между обращениями от разных DSP ядер могут иметь место конфликты, даже если эти обращения выполняются к различным ресурсам процессора.

DSP ядро поддерживает 32,64,128 разрядные пересылки, в то время, как доступ ко многим ресурсам процессора возможен только 64/32 или даже только 32-х разрядными обращениями.

В связи с этим введён механизм разбиения обращения от DSP ядра на 32-х или 64-х разрядные обращения. Для управления режимом разбиения в регистре SR введены биты SplitMode = SR[15:14]

Таблица 1.8. Режим разбиения в зависимости от значения бит SR[15:14] = SplitMode[1:0]

SplitMode[1:0]	Разрядность обращения от DSP	Обращения к ресурсам процессора
00/11 нет разбиения	32	одно 32-х разрядное
00/11 нет разбиения	64	одно 64-х разрядное
00/11 нет разбиения	128	одно 64-х разрядное. биты [127:96] как для данных на запись, так и читаемых данных игнорируются
01 разбиение на 32-х разрядные обращения	32	одно 32-х разрядное
01 разбиение на 32-х раз-	64	два 32-х разрядных

рядные обращения		
01 разбиение на 32-х разрядные обращения	128	четыре 32-х разрядных
10 разбиение на 64-х разрядные обращения	32	одно 32-х разрядное
10 разбиение на 64-х разрядные обращения	64	одно 64-х разрядное
10 разбиение на 64-х разрядные обращения	128	два 64-х разрядных

Запись во внешнюю память является буферизованной, таким образом операции записи не приводят к останову конвейера DSP ядра за исключением следующих случаев:

Идут непрерывные 128 разрядные записи и включено разбиение обращений (SplitMode = 01 или SplitMode = 10), либо идут непрерывные 128 или 64 разрядные записи и SplitMode = 01, в этом случае пропускной способности внешнего порта не хватает, буфер обращений переполняется и до готовности принять новое обращение ядро блокируется. Такая же ситуация может возникнуть при конфликтах между ядрами при одновременном обращении к внешнему адресному пространству.

Любое чтение по адресам из внешнего для DSP адресного пространства приводит к останову конвейера вплоть до момента получения прочитанных данных.

Поскольку каждое чтение приводит к останову, имеет смысл группировать чтения в два 128 разрядных обращения. Так, например, чтение группы регистров, выполненное по следующей программе:

```
Move (a0)+i0, r2.l
Move (a0)+i0, r4.l
Move (a0)+i0, r6.l
Move (a0)+i0, r8.l
Move (a0)+i0, r10.l
Move (a0)+i0, r12.l
Move (a0)+i0, r14.l
Move (a0)+i0, r16.l
```

в среднем занимает в 5.5-6 раз больше тактов, чем чтение пакета из 8 слов, выполненное командой

```
Move (a0), r2.q (at),r0.q
```

2. Программная модель DSP-ядра Elcore-30M

Программная модель DSP–ядра включает в себя память (программ и данных) и программно-доступные регистры. Регистры обменного буфера XBUF и регистр флагов обмена EFR являются общими для всего DSP–кластера, остальные регистры принадлежат конкретному DSP–ядру и входят в состав одного из его исполнительных устройств. К исполнительным устройствам DSP–ядра относятся:

- вычислительная секция ALU;
- адресные генераторы для XY-памяти данных (AGU и AGU-Y);
- устройство программного управления PCU.

По своему назначению все регистры делятся на регистры данных, объединенные в регистровый файл (RF), и регистры управления (все остальные). Регистры управления разделены на четыре подмножества:

- регистры адресных генераторов AGU, AGU-Y;
- регистры обменного буфера XBUF;
- регистры устройства управления PCU;
- регистры-аккумуляторы (в составе ALU);

Программно-доступные регистры DSP–ядра (включая стеки и регистровый файл) приведены на рисунке 2.1.

2.1 Регистры ALU

2.1.1 Каждая вычислительная секция ALU содержит регистровый файл RF – реконфигурируемый массив (16x128 или 32x64 или 32x32 или 32x16) регистров данных, регистр параметра денормализации PDNR, регистр кодов условий (регистр признаков) CCR, реконфигурируемый массив (8x64 или 16x32) регистров-аккумуляторов.

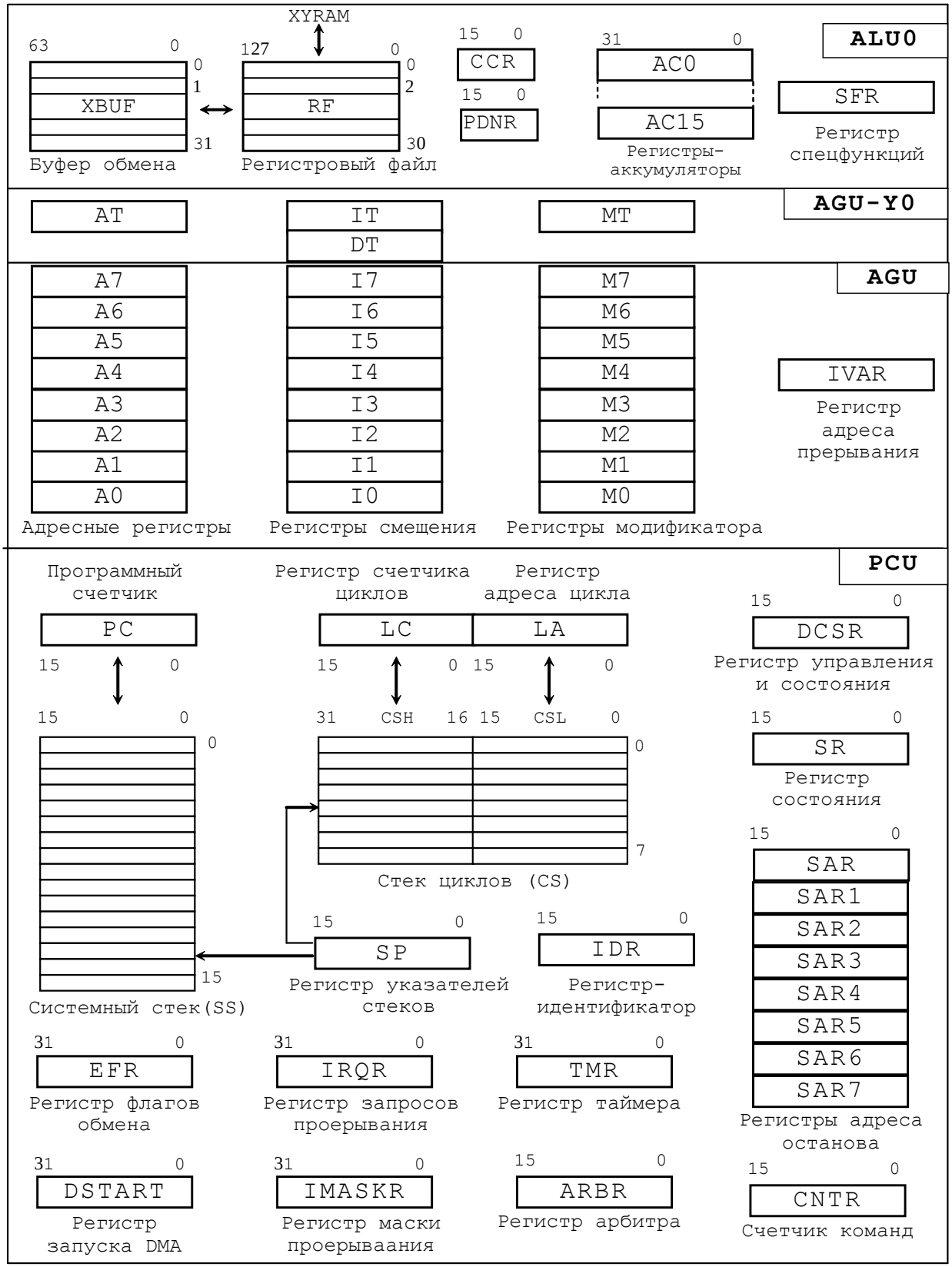


Рис.2.1 ─ Программно-доступные регистры DSP–ядра Elcore-30M

2.1.2 Регистровый файл

Исходные данные и результаты всех операций ALU хранятся в регистровом файле (RF), который представляет собой реконфигурируемый массив регистров данных (16 регистров по 128 разрядов; или 32 регистра по 64 разряда; или 32 регистра по 32 разряда; или 32 регистра по 16 разрядов). Структура регистрового файла приведена на рисунке 2.2.

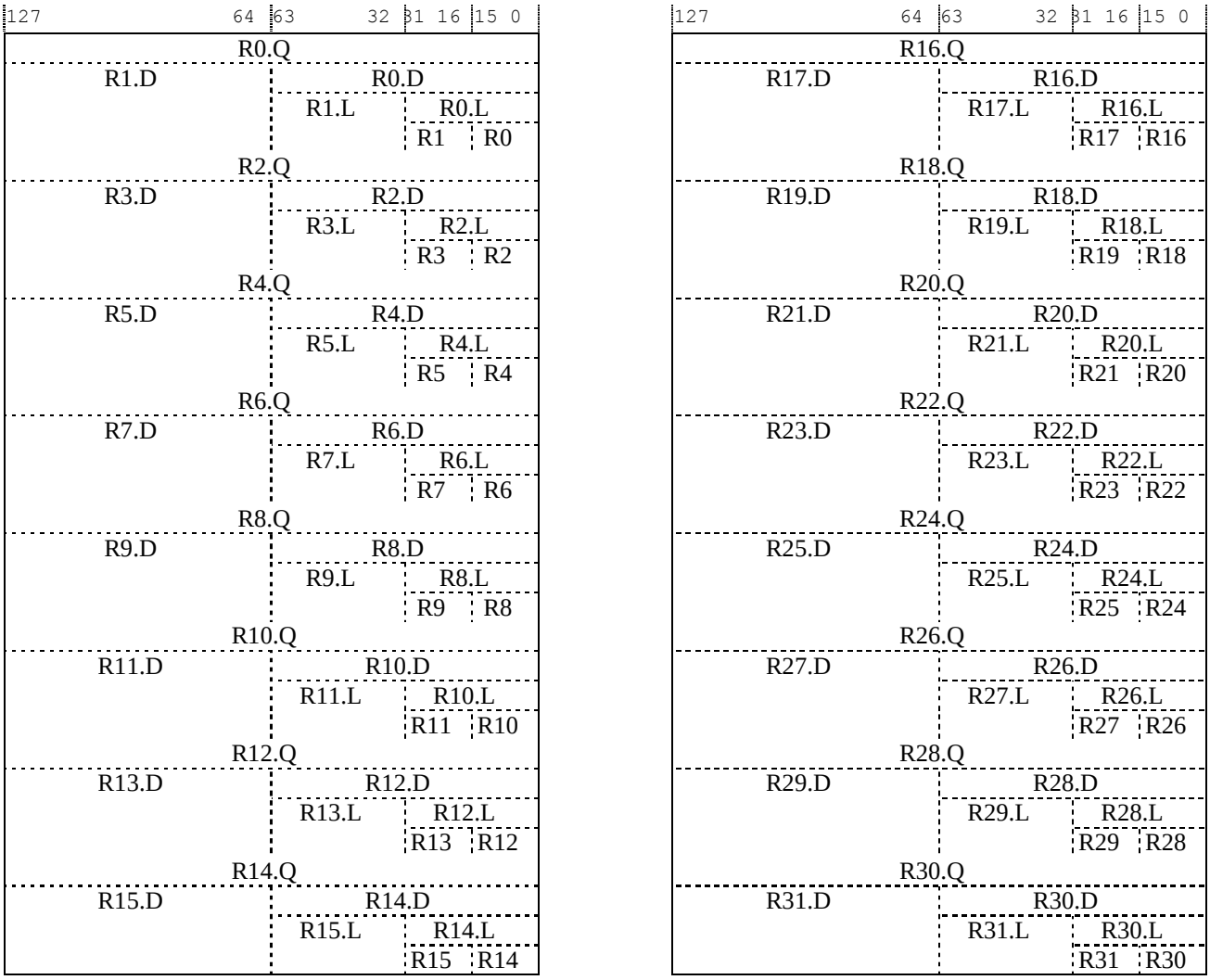


Рисунок 2.2 ─ Структура регистрового файла

Для определения форматов регистров используются следующие мнемоники:

R – 16-разрядные регистры;

R.L – 32-разрядные регистры;

R.D – 64-разрядные регистры;

R.Q ─ 128-разрядные регистры.

16/32/64-разрядные регистры данных могут иметь номера с R0 по R31, а 128-разрядные регистры – только четные номера с R0 по R30. Четный и нечетный (с номером, большим на единицу) регистры одинаковой разрядности объединяются попарно и образуют 16 регистров большей разрядности с четными номерами, например, два 16-разрядных регистра R0 и R1 образуют 32-разрядный регистр R0.L.

2.1.3 Регистры-аккумуляторы

Регистры-аккумуляторы являются специализированными 32/64-разрядными регистрами данных, предназначенными для накопления результата в операциях умножения с накоплением.

DSP-ядро Elcore-30M содержит шестнадцать 32-разрядных регистров-аккумуляторов AC0-AC15, которые могут попарно объединяться в восемь 64-разрядных, либо четыре 128-разрядных регистров. Два 32-разрядных регистра, четный и нечетный (с номером, большим на единицу), объединяются в один 64-разрядный регистр для получения 64-разрядного результата.

Структура регистрового файла регистров-аккумуляторов приводится на рисунке 2.3.

ACn.L – 32-разрядные регистры; n=0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15;

ACn.D – 64-разрядные регистры; n=0,2,4,6,8,10,12,14;

ACn.Q – 128-разрядные регистры; n=0,4,8,12.

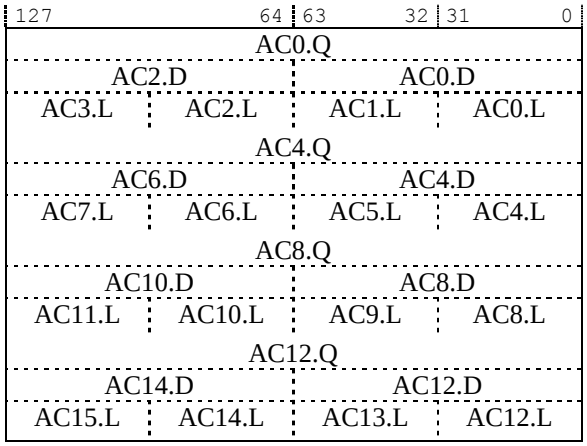


Рисунок 2.3 ─ Структура регистрового файла регистров-аккумуляторов Elcore-30M

Регистры-аккумуляторы доступны по записи и по чтению как со стороны CPU, так и со стороны DSP-ядра.

Начальное состояние регистров-аккумуляторов равно нулю.

2.1.4 Регистр PDNR

Регистр PDNR - регистр управления, предназначенный для измерения параметра денормализации (PDN) и управления режимом блочной экспоненты и режимом масштабирования (Scaling).

Назначение разрядов регистра PDNR приведено в таблице 2.1.

Начальное состояние регистра PDNR = 0x0000.

Таблица 2.1

Разряды регистра	Идентификатор	Назначение
0 – 4	Cpdn	Текущий код PDN
5	F	(X/L) – формат анализируемой информации (0 – Long, 1 – X16)
7	Epdn	Программный признак разрешения детектирования и изменения PDN (0 – нет разрешения, 1 – разрешение)
8,9	SC	Величина масштабирования результата (00 – нет сдвига, 01 - сдвиг на 1 разряд, 10 - сдвиг на 2 разряда)
15	Esc	Признак разрешения масштабирования результата (0 – нет разрешения, 1 – разрешение)
6,10-14	-	Не используются

2.1.5 Регистр CCR

Регистр CCR - регистр управления, предназначенный для хранения признаков результатов вычислительных операций. Регистр CCR содержит два поля признаков: основное {Ev,U,N,Z,V,C} (разряды [5:0]) и дополнительное {Evm,Um,Nm,Zm,Vm,Cm} (разряды [15:10]). Поле признаков в младшем байте регистра CCR является основным, т.к. на его основе формируются условия исполнения команд.

Поля признаков формируются по следующим правилам:

- при исполнении одной операции типа OP1 (AU/LU/FASU) ее признаки помещаются только в основное поле;
- при исполнении одной операции типа OP2 (MS/SH/FMU) ее признаки помещаются в оба поля;
- при одновременном выполнении двух вычислительных операций признаки, формируемые операцией типа OP1 поступают в основное поле, признаки операции типа OP2 - в дополнительное поле;
- в тех случаях, когда операция типа OP1 заполняет только часть признаков в основном поле, оставшиеся формируются операцией OP2.

Регистр CCR содержит также специальные признаки E, t и два управляющих разряда RND и S.

Назначение разрядов регистра CCR приведено в таблице 2.2.

Начальное состояние регистра CCR = 0x0000.

Таблица 2.2

Разряды регистра	Идентификатор	Назначение
0	C	Признак переноса, сформированного в результате выполнения операции (0 – нет переноса, 1 – есть перенос)
1	V	Признак переполнения результата (0 – нет переполнения, 1 – есть переполнение)
2	Z	Признак нулевого результата (0 – результат не нулевой, 1 – результат нулевой)
3	N	Знак результата (0 – знак положительный, 1 – знак отрицательный)
4	U	Признак ненормализованного результата (0 – нормализованный результат, 1 – ненормализованный результат)

Окончание таблицы 2.2

Разряды регистра	Идентификатор	Назначение

5	Ev	Запомненный ранее возникший признак переполнения результата (0 – не было переполнения, 1 – было переполнение)
6	E	Экспоненциальный признак (формируется командой CMPE)
7	t	Признак истинности условия после исполнения условной команды (t=0 – безусловная команда либо условие ложно; t=1 – условие истинно
8	S	Бит включения режима насыщения результата (0 – отключение режима насыщения, 1 – включение режима насыщения)
9	RND	Бит управления режимом округления результата (0 – CR (Convergent Rounding), 1 – TCR (Two's-Complement Rounding))
10	Cm	Признак переноса сформированного в результате выполнения операции OP2 (0 – нет переноса, 1 – есть перенос)
11	Vm	Признак переполнения результата операции OP2 (0 – нет переполнения, 1 – есть переполнение)
12	Zm	Наличие нулевого результата операции OP2 (0 – результат не нулевой, 1 – результат нулевой
13	Nm	Значение знака результата операции OP2 (0 – знак положительный, 1 – знак отрицательный)
14	Um	Признак ненормализованного результата операции OP2 (0 – нормализованный результат, 1 – ненормализованный результат
15	Evm	Запомненный ранее возникший признак переполнения результата операции OP2 (0 – не было переполнения, 1 – было переполнение)

2.2 Регистры адресных генераторов AGU, AGU-Y и виды адресной арифметики

2.2.1 Регистры AGU

Генератор адреса AGU содержит восемь наборов по три регистра (Рисунок 2.4): регистр адреса An, регистр смещения In, регистр модификатора Mn (n=0-7). Эти регистры могут использоваться для хранения адресных указателей или других данных. При косвенной адресации операндов в памяти автоматически включается механизм обновления адресных указателей. Адресные регистры могут быть запрограммированы для линейной адресации, модульной адресации или реверсивной адресации.

Адресные регистры	Регистры смещения	Регистры модификатора
A7	I7	M7
A6	I6	M6
A5	I5	M5
A4	I4	M4
A3	I3	M3
A2	I2	M2
A1	I1	M1
A0	I0	M0

Рисунок 2.4 ─ Программная модель AGU

Эти регистры могут также использоваться для хранения произвольных данных.

2.2.2 Регистры AGU-Y

Генератор адреса AGU-Y содержит набор из четырех регистров (Рисунок 2.5): регистра адреса AT, регистров смещения IT и DT, регистра модификатора MT.

Рисунок 2.5 ─ Программная модель AGU-Y

2.2.3 Назначение регистров адресных генераторов

2.2.3.1 Особенностью DSP-ядра Elcore-30M по сравнению с предшествующими модификациями DSP-ядер ELcore-xx платформы «Мультикор» является то, что в Elcore-30M расширен до 32 разрядов формат адресных регистров A0 – A7, AT. Это вызвано расширением адресного пространства DSP-кластера и выходом его за пределы доступности 16-разрядных адресных регистров, существовавших в предшествующих модификациях DSP ELcore-xx. При этом регистры смещения I0–I7,IT,DT и регистры модификаторов M0–M7, MT в Elcore-30M попрежнему остаются 16-разрядными.

32-разрядные адресные регистры A0-A7, AT содержат адреса памяти данных. Содержимое адресного регистра может непосредственно указывать на данные в памяти либо используется для формирования указателя со смещением. Адресный регистр обновляется после формирования адресного указателя (пост-модификация).

16-разрядные регистры смещений I0-I7, IT, DT содержат значения смещений, используемых для инкрементации или декрементации адресных регистров при выполнении обновления адреса.

16-разрядные регистры модификаторов M0-M7, MT определяют тип адресной арифметики, применяемой при модификации адреса.

Адресные АЛУ поддерживают три типа арифметики: *линейную, модульную и арифметику с обратным переносом.* Для модульной арифметики содержимое регистров модификаторов определяет также модуль.

Значения модификатора Mn и соответствующие им типы адресной арифметики указаны в таблице 2.3.

Таблица 2.3. ─ Типы адресной арифметики

Модификатор Mn	Адресная арифметика
0x0000	Арифметика с обратным переносом
0x0001	Модуль 2
0x0002	Модуль 3
…	…
0x7FFE	Модуль 32767 ($2^{15} – 1$)
0x7FFF	Модуль 32768 (2^{15})
0x8001	Модуль 2 с кратным обращением
0x8003	Модуль 4 с кратным обращением
0x8007	Модуль 8 с кратным обращением
…	…
0x9FFF	Модуль 2^{13} с кратным обращением
0xBFFF	Модуль 2^{14} с кратным обращением
0xFFFF	Линейная арифметика (Модуль 2^{16})
Остальные комбинации – резерв	

2.2.3.2 Линейная адресная арифметика (Mn = 0xFFFF)

Модификация адреса выполняется с использованием линейной адресной арифметики. 16-разрядное смещение, In, +1 или -1 используется для модификации адреса. Диапазон значений In рассматривается как знаковый и находится в пределах от –32768 до +32767.

Важной особенностью линейной адресной арифметики Elcore-30M является то, что операции инкремента и декремента выполняются в 16-разрядном формате и диапазон изменения адресов находится в пределах от 0 до 0xFFFF. Таким образом, путем операций инкремента и декремента возможен переход из адресного пространства одного сегмента в адресное пространство другого сегмента в пределах всего DSP-кластера.

2.2.3.3 Адресная арифметика с обратным переносом (Mn = 0x0000)

Этот вариант адресной арифметики выбирается посредством установки регистра модификатора в 0. Модификация адреса в этом случае выполняется аппаратно с распространением переноса в обратном направлении – от старших разрядов к младшим.

Операция модификации адреса с обратным переносом эквивалентна последовательному выполнению следующих процедур:

- изменению на обратный порядка следования разрядов в регистрах адреса и смещения (при этом старший бит становится младшим и т.д.);
- иодификации адреса посредством нормальной операции сложения;
- возвращению первоначального порядка следования разрядов адреса.
- в случае, когда величина смещения составляет $2^{(k-1)}$ (целая степень двойки), такая модификация адреса эквивалентна:
- обращению порядка следования k младших разрядов An;

- увеличению на 1;
- возвращению исходного порядка следования k младших разрядов An.

Рассматриваемый режим адресной арифметики удобен при реализации алгоритма быстрого преобразования Фурье (БПФ).

2.2.3.4 Модульная адресная арифметика (Mn = Modulus – 1)

Модификация адреса выполняется по модулю M, где M - целое число в пределах от 2 до 32768. Арифметика по модулю M вынуждает значение адреса оставаться в пределах диапазона значений, отличающихся друг от друга не более чем на M-1.

Величина M-1 хранится в регистре модификатора адреса. Нижняя граница диапазона (базовый адрес) должна иметь нули в младших k разрядах, где $2^k >= M$. Верхняя граница диапазона определяется как сумма нижней границы и модуля минус единица (базовый адрес + M - 1). Нижняя и верхняя границы диапазона определяются значением An.

При этом необязательно устанавливать An равным базовому адресу. Достаточно того, чтобы величина An находилась в пределах требуемого диапазона.

Если при вычислении адреса в этом режиме используется смещение In, его величина не должна превышать M.

Рассматриваемый тип адресной арифметики удобен при организации циклических буферов для реализации на их основе структур данных типа очередей (FIFO), линий задержки и т.п.

2.2.3.5 Кратная модификация адреса по модулю

Этот тип адресной арифметики выбирается посредством установки в «1» 15-го разряда регистра модификатора Mn, как это показано в таблице 2.3.

Модификация адреса выполняется по модулю M, где M - степень двойки в пределах от 2^1 до 2^{14}. Арифметика по модулю M вынуждает значение адреса оставаться в пределах диапазона значений, отличающихся друг от друга не более чем на M-1.

Величина M-1 хранится в младших 15-ти разрядах регистра модификатора адреса Mn. Нижняя граница диапазона (базовый адрес) должна иметь нули в младших k разрядах, где $2^k >=$ M. Верхняя граница диапазона определяется как сумма нижней границы и модуля минус единица (базовый адрес + M - 1).

Нижняя и верхняя границы диапазона определяются значением An. При этом необязательно устанавливать An равным базовому адресу. Достаточно того, чтобы величина An находилась в пределах требуемого диапазона.

2.2.4 Особенности X- и Y- указателей

Виды адресации памяти данных XRAM сведены в в таблице 2.4. Режим адресации определяется полем “mode” командного слова инструкции.

Таблица 2.4. ─ Виды X-адресации памяти данных (указатели A0-A7)

Код режима		

адресации (mode)	Обозначение	Пояснение
000	-	Отмена пересылки
001	(An)	Косвенная
010	(An)+	Пост - автоинкремент
011	(An)-	Пост - автодекремент
100	(An)+In	Пост - автоувеличение
101	(An)-In	Пост - автоуменьшение
110	(An+In)	Индексирование (An не меняется)
111	(An+dspl)	С непосредственным смещением (A не меняется)
Примечание. По установленному признаку “u” в командном слове вычисляется исполнительный адрес без выполнения самой пересылки		

Виды Y-адресации сведены в в таблице 2.5. Режим адресации определяется полем “AT” инструкции и управляющим параметром YM (11-й разряд регистра SR).

Таблица 2.5. ─ Виды Y-адресации памяти данных (указатель AT)

Код режима адресации (поле “AT”)	YM	Обозначение	Пояснение
00	X	-	Отмена пересылки
01	X	(AT)	Косвенная
10	X	(AT)+IT	Пост - автоувеличение
11	0	(AT+IT)	Индексирование (An не меняется)
11	1	(AT)+DT	Пост - автоувеличение

2.2.5 Начальное состояние регистров адреса A0-A7, AT

Начальное состояние регистров A0-A7, AT DSP-ядер приведено в таблице 2.6.

Таблица 2.6. ─ Начальное состояние регистров A0-A7, AT

Условное обозначение	Разрядность	Наименование	Начальное состояние	
			DSP0	DSP1
A0-A7	32 R/W	Адресный регистр AGU	0x00000	0x08000
AT	32 R/W	Адресный регистр AGU-Y	0x04000	0x0C000

Таким образом, при начальной установке регистры A0-A7 указывают на начало, а регистры AT – на середину ближней (локальной) памяти соответствующего DSP.

2.2.6 Регистр адреса вектора прерывания IVAR

В Elcore-30M реализован механизм прерываний, рассмотренный подробнее в п.2.3.12. При отработке прерывания автоматически выполняется команда JSR IVAR, по которой происходит переход на подпрограмму обработки прерываний, находящуюся по адресу, содержащемуся в регистре адреса вектора прерывания IVAR (16 бит, запись/чтение).

Начальное состояние регистра IVAR=0x1F00.

2.3 Регистры устройства управления PCU

Устройство программного управления PCU включает в себя набор управляющих регистров и стеков:

- регистр управления и состояния DCSR;
- программный счетчик PC;
- регистр состояния SR;
- регистр-идентификатор IDR;
- регистр флагов обмена EFR;
- регистр запуска DMA DSTART;
- регистр запросов на прерывание IRQR;
- регистры масок запросов на прерывания IMASKR, QMASKR0, QMASKR1, QMASKR2;
- регистр управления арбитром памяти ARBR;
- регистр таймера TMR;
- регистр адреса окончания цикла LA;
- регистр счетчика циклов LC;
- системный стек SS;
- стеки циклов CSL, CSH;
- регистр указателей стека SP;
- регистры адреса останова SAR, SAR1 – SAR7;
- счетчик команд CNTR;
- регистр спецфункций SFR.

2.3.1 Регистр управления и состояния DCSR

Регистр управления и состояния (DCSR) содержит разряды управления, определяющие состояние и режим работы DSP-ядра, а также прерывания, формируемые DSP-ядром для обработки в RISC-ядре.

Назначение разрядов регистра DCSR указано в таблице 2.7.

Начальное состояние DCSR = 0x0000.

Таблица 2.7. ─ Назначение разрядов регистра DCSR

Разряды регистра	Идентификатор	Назначение
0	PI	Программное прерывание PI.
1	SE	Прерывание по ошибке стека SE
2	BRK	Прерывание по останову BREAK
3	STP	Прерывание по останову STOP
4	WT	Состояние ожидания обмена с XBUF
5−13	−	Не используется
14	RUN	Состояние исполнения программы

15	–	Не используется

2.3.2 Программный счетчик PC

Регистр программного счетчика PC предназначен для хранения 16-разрядного адреса инструкции в программной памяти. Инкрементированное значение PC заносится в системный стек при инициализации нового программного цикла DO, DOFOR и при входе в подпрограмму.

Начальное состояние PC = 0x0000.

2.3.3 Регистр состояния SR

Регистр состояния SR содержит параметры управления и состояния DSP-ядра. Разряды [7:0] регистра SR доступны только по чтению, остальные - по записи/чтению.

Назначение разрядов регистра SR указано в таблице 2.8.

Таблица 2.8. ─ Назначение разрядов регистра SR

Разряды регистра	Идентификатор	Назначение
0	C	Перенос
1	V	Признак переполнения
2	Z	Признак нулевого результата
3	N	Признак отрицательного результата
4	U	Признак ненормализованного результата
5	Ev	Флаг переполнения
6	E	Экспоненциальный признак
7	t	Признак истинности последнего условия
8	nBLKmod	Управление отключением блокировки конвейера на такт в случае предмодификации адреса при обращении к памяти (индексация)
9	DD	Управление режимом записи результата в инструкциях ADDSUB, ADDSUBL, ADDSUBX, FAS, CVFE (Double Destination)
10	BD	Управление блокировкой конвейера (Blocking Disabled)
11	YM	Управление режимом адресации памяти YRAM
12-15	–	Не используются

Начальное состояние регистра SR = 0x0000.

Разряды [7:0] регистра SR содержат интегральные признаки предыдущей арифметической операции.

Бит nBLKmod предназначен для управления отключением блокировки конвейера на такт в случае предмодификации адреса при обращении к памяти (режим индексации):

1) nBLKmod = 0. Каждое обращение к памяти DSP с предмоджификацией адреса приводит к блокировке ядра на 1 такт. Обращение отрабатывается корректно, ограничений для модификации адреса нет.

2) nBLKmod = 1. Обращения к памяти DSP с предмоджификацией адреса не приводят к блокировке ядра, однако существует ограничение на модификацию адреса: исходный адрес и адрес получаенный после модификации обязательно должны находится в одной странице памяти. Для обращений без модификации и с постмодификацией адреса ограничений нет.

Бит DD (Double Destination) = SR[9] предназначен для выбора режимов исполнения вычислительных команд, формирующих двойной результат: ADDSUB, ADDSUBL, ADDSUBX,

FAS, CVFE. При DD=0 (по умолчанию) указанные команды выполняются в варианте с двумя результатами и двумя адресами записи, при DD=1 один результат удвоенного формата записывается по одному адресу D.L(D.D). (Более подробную информацию можно получить из описания указанных инструкций).

Бит BD (Blocking Disabled) = SR[10] предназначен для управления автоматической блокировкой программного конвейера: при BD = 0 блокировка включена, при BD = 1 отключена.

Пояснение: автоматическая блокировка (включена по умолчанию при BD=0) вызывает торможение программного конвейера в тех случаях, когда последующая инструкция использует еще не сформированный результат предыдущей инструкции. Отключение автоматической блокировки (BD=1) может производиться с целью ускорения работы программы при условии хорошего понимания работы программного конвейера Elcore-30M.

Отключение автоматической блокировки не оказывает влияния на остановы вычислительного ядра, вызванные конфликтами при обращении к памяти.

Назначение бита YM = SR[11] описано в таблице 2.5.

2.3.4 Регистр-идентификатор IDR

Состояние регистров-идентификаторов DSP-ядер Elcore-30M в составе DSP-кластера: IDR=0xn108, где n=0,1 – номер DSP-ядра.

2.3.5 Регистр адреса окончания цикла LA

Регистр адреса окончания цикла LA содержит адрес последней инструкции в программном цикле DO, DOFOR. Этот регистр заносится в стек SS по команде DO, DOFOR и извлекается обратно по окончании вложенного цикла либо по команде ENDDO.

Начальное состояние LA = 0x0000.

2.3.6 Регистр счетчика циклов LC

Формат регистра LC приведен в таблице 2.9.

Таблица 2.9. ─ Назначение разрядов регистра LC

Разряды регистра	Идентификатор	Назначение
0 - 13	Nc	Текущее значение 14-разрядного счетчика программных циклов Nc – разряды 0-13 регистра LC
14	LF	Флаг цикла DO – разряд 14 регистра LC
15	FV	Флаг цикла DOFOR – разряд 15 регистра LC

Значение счетчика программных циклов Nc определяет количество повторений программного цикла DO, в пределах от 1 до $(2^{14} – 1)$. Этот регистр заносится в верхнюю (старшую) половину стека циклов CSL по команде DO (образуется вложенный программный цикл) и извлекается обратно по окончании вложенного цикла либо по команде ENDDO.

Начальное состояние LC = 0x0000.

2.3.7 Стеки SS, CSL, CSH

Устройство программного управления содержит системный стек SS и стеки циклов CSL, CSH. Системный стек SS имеет объем 15 16-разрядных слов и используется для автоматического сохранения содержимого регистра программного счетчика PC при входе в подпрограмму или в цикл DO, DOFOR. Стеки циклов имеют объем по 7×16 бит и предназначены для хранения соответственно длины цикла и адреса последней инструкции цикла (LC и LA). Стеки участвуют в обменах как 16-разрядные регистры управления – SS, CSL и CSH.

2.3.8 Регистр указателей стека SP

Регистр указателей стека SP содержит указатели на последнее записанное в стеки SS, CSH слово. Назначение разрядов регистра SP указано в таблице 2.10.

Таблица 2.10. ─ Назначение разрядов регистра SP

Разряды регистра	Идентификатор	Назначение
0 - 3	SP	указатель системного стека
4	SSE	флаг ошибки системного стека
5	UFS	флаг переполнения системного стека
6, 7	–	не используются
8-10	CP[2:0]	указатель стека циклов
11	CSE	флаг ошибки стека циклов
12	UFC	флаг переполнения стека циклов
13-15	-	не используются

Младший байт регистра SP содержит указатель и флаги системного стека; старший байт - указатель и флаги стека циклов.

Начальное состояние SP = 0x0000.

2.3.9 Регистры адреса останова SAR, SAR1-SAR7

Регистры адреса останова SAR, SAR1–SAR7 являются специализированными 16-разрядными регистрами, использующимися при отладке DSP-ядра. Регистры SAR, SAR1–SAR7 определяют точки останова (Breakpoint) - адрес инструкции, непосредственно перед исполнением которой должен произойти останов DSP-ядра. Перед исполнением инструкции с указанным адресом DSP-ядро переходит в состояние останова (RUN=0) и флаг прерывания BRK устанавливается в «1».

Начальное состояние SAR, SAR1–SAR7 = 0xFFFF.

2.3.10 Счетчик команд CNTR

Счетчик команд CNTR - специализированный 16-разрядный регистр, предназначенный для отладки DSP-ядра. Регистр CNTR задает пошаговый режим исполнения программ в соответствии с таблицей 2.11.

Начальное состояние CNTR = 0x0000.

Таблица 2.11. ─ Назначение разрядов регистра CNTR

Счетчик CNTR	Режим исполнения программ
0x0000	Нормальный режим исполнения программ. Число исполняемых команд не ограничено.
N > 0	Пошаговый режим исполнения программ. После исполнения N инструкций DSP-ядро переходит в состояние останова (RUN=0) и флаг прерывания BRK устанавливается в “1”.

2.3.11 Регистры управления прерываниями и DMA-обменами

В Elcore-30M имеется механизм прерываний, с помощью которого, в частности, осуществляется запуск DSP со стороны DMA. Кроме того, прерывания в DSP Elcore-30M могут поступать также со стороны CPU, другого DSP-ядра, таймеров.

Для управления DMA-обменами и прерываниями имеется следующий набор регистров:

- регистр запросов на прерывание DSP со стороны DMA, CPU, других DSP-ядер, таймеров – IRQR;

- регистр маски запросов на прерывание DSP – IMASKR;

- псевдорегистр (только запись) запуска со стороны DSP каналов DMA и других DSP-ядер – DSTART.

2.3.12 Механизм отработки прерываний

Отработка запросов на прерывание (в том числе на запуск DSP со стороны DMA) отрабатывается одинаковым образом:

1) аппаратно взводится в состояние «1» соответствующий бит регистра IRQR;

2) аппаратно переводится в состояние «1» бит RUN регистра DCSR (если он еще не находится в этом состоянии);

3) автоматически выполняется команда JSR IVAR, по которой происходит переход на подпрограмму обработки прерываний, находящуюся по адресу, содержащемуся в регистре адреса вектора прерывания IVAR. Подпрограмма обработки прерываний должна оканчиваться командой возврата из подпрограммы обработки прерывания RTI.

Поступающие прерывания не имеют иерархии приоритетов и обрабатываются последовательно. Если во время обработки прерывания приходит новый запрос, то обработка его начнется только после завершения текущей подпрограммы обработки прерывания.

2.3.13 Регистр запросов на прерывание DSP (IRQR)

Регистр IRQR содержит флаги запросов («1» - наличие запроса, «0» - отсутствие запроса) на прерывание DSP со стороны DMA, CPU, другого DSP-ядра, таймера. Назначение разрядов регистра IRQR приведено в таблице 2.12.

Регистр IRQR доступен по записи и чтению со стороны CPU и DSP.

Таким образом, состояние разрядов регистра IRQR может изменяться как аппаратно – при приходе соогветствующего сигнала запроса на прерывание, так и программно – при записи со стороны CPU или DSP.

Таблица 2.12. Назначение разрядов регистра IRQR

Номер разряда	Наименование разряда	Назначение

0	DRQ0	Запрос на прерывание DSP со стороны канала DMA MemCh0
1	DRQ1	Запрос на прерывание DSP со стороны канала DMA MemCh1
2	DRQ2	Запрос на прерывание DSP со стороны канала DMA MemCh2
3	DRQ3	Запрос на прерывание DSP со стороны канала DMA MemCh3
4-23	-	Резерв
24	IRQ0	Запрос на прерывание DSP со стороны DSP0
25	IRQ1	Запрос на прерывание DSP со стороны DSP1
26-27	-	Резерв
28	INT_TMR	Запрос на прерывание DSP со стороны таймера TMR
29	FPE	Исключение при исполении операции в формате плавающей точки (V=1)
30	QT0	Запрос на прерывание DSP со стороны CPU (QSTR0)
31	QT1	Запрос на прерывание DSP со стороны CPU (QSTR1, QSTR2)

Начальное состояние регистра IRQR =0x0.

2.3.14 Регистры масок запросов на прерывание DSP (IMASKR, QMASKR0, QMASKR1, QMASKR2)

Регистр IMASKR содержит 32 разряда, каждый из которых разрешает («1») либо запрещает («0») запрос на прерывание DSP от соответствующего разряда регистра IRQR. Регистр доступен по чтению и записи со стороны CPU или DSP.

Регистр маски запросов на прерывание QMASKR0 содержит 32 разряда, каждый из которых разрешает («1») либо запрещает («0») прерывание DSP от соответствующего разряда регистра запросов прерываний со стороны CPU (регистр QSTR0).

Регистр маски запросов на прерывание QMASKR1 содержит 32 разряда, каждый из которых разрешает («1») либо запрещает («0») прерывание DSP от соответствующего разряда регистра запросов прерываний со стороны CPU (регистр QSTR1).

Регистр маски запросов на прерывание QMASKR2 содержит 32 разряда, каждый из которых разрешает («1») либо запрещает («0») прерывание DSP от соответствующего разряда регистра запросов прерываний со стороны CPU (регистр QSTR2).

Начальное состояние регистров IMASKR, QMASKR0, QMASKR1, QMASKR2 - нулевое.

2.3.15 Регистр запуска DMA со стороны DSP (DSTART)

Регистр DSTART доступен по только записи и предназначен для запуска соответствующего канала DMA со стороны DSP. Назначение разрядов регистра DSTART приведено в табл. 2.13.

Таблица 2.13. ─ Назначение разрядов регистра DSTART

Номер разряда	Наименование разряда	Назначение
0	DE0	Запрос со стороны DSP на запуск канала DMA MemCh0
1	DE1	Запрос со стороны DSP на запуск канала DMA MemCh1
2	DE2	Запрос со стороны DSP на запуск канала DMA MemCh2
3	DE3	Запрос со стороны DSP на запуск канала DMA MemCh3
4-23	-	Резерв
24	DSP0	Запрос на прерывание DSP0
25	DSP1	Запрос на прерывание DSP1

26-31	-	Резерв

2.3.16 Регистр таймера (TMR)

Регистр таймера TMR (32 разряда, запись/чтение) предназначен для формирования периодических запросов на прерывание DSP. Период запросов определяется значением, содержащимся в регистре TMR по формуле:

$T_{INT} = (TMR +1)* T_{CLK,}$

где T_{CLK} - период тактовой частоты DSP.

При TMR = 0 запросы на прерывание DSP не формируются.

Регистр TMR доступен по записи и чтению. Начальное состояние регистра TMR = 0x0.

2.3.17 Регистр управления локальным арбитром (ARBR)

2.3.17.1 Принципы арбитража и режимы работы

Вся память DSP кластера разбита на 2 сегмента, каждый из которых соответствует определенному DSP ядру и состоит из 4 страниц каждый. Таким образом, для каждого ядра существует сегмент “своей” или ближней памяти. В архитектуре глобального коммутатора предусмотрены два локальных арбитра, каждый из них осуществляет арбитраж обращений к определенному сегменту памяти. Каждый из локальных арбитров настраивается и работает независимо от другого арбитра. Таким образом, одно ядро может иметь высший приоритет для обращений к одному сегменту памяти и низший для обращений к другому.

Каждая страница памяти состоит из четырёх физических блоков по 4К 32 разрядных слов каждый. Для организации чтения 128 разрядных слов, а так же для повышения производитетльности при 32 разрядных обменах с памятью применена технология расслоения памяти. Т.е. любые четыре последовательно идущих адреса одной страницы располагаются в 4-х разных физических блоках.

В случае если оба ядра обращаются к одной странице памяти, отрабатывается обращение от ядра, имеющего на данный момент высший приоритет (другое ядро останавливаются до момента получения высшего приоритета). Если обращения идут к разным страницам (даже внутри одного сегмента), конфликтов не возникает. Конфликтов так же не возникает при обращении одного ядра по X и Y указателям к одной странице памяти, при условии, что обращения идут к разным физическим блокам (условие бесконфликтоно обращения одного DSP кодной странице памяти: для 32-х и 64-х разрядных обращений XAB % 4 != YAB % 4).

Обращения к своей памяти не приводят к останову конвейера, если отсутствуют конфликты с другими ядрами, либо для данного ядра явно установлен высший приоритет для обращений к своей памяти (заданы значения бит DEN=1 и DPTR = 0 в регистре ARBR данного ядра).

Остальная память является для текущего ядра дальней. Чтение из дальней памяти неизбежно приводит к останову конвейера на четыре дополнительных такта. Одиночная запись в дальнюю память буферизуется и не приводит к блокировкам. Поддерживается пакетная запись в дальнюю память, которая так же проходит без дополнительных блокировок конвейера. Поддержка пакетных обращений имеет место при работе в режиме захвата, либо при явном зада-

нии высшего приоритета для данного ядра. При работе в режиме ограничения, максимальная длина пакета определяется значением ограничителя.

Локальный арбитр может работать в режиме *захвата* (режим по умолчанию). В этом режиме, ядро, получившее разрешение для обращений к определенному сегменту памяти, получает высший приоритет, и сохраняет его до тех пор, пока есть обращения к данному сегменту памяти. Как только обращения от текущего ядра прекращаются, право на захват циклически передается следующему ядру.

Так же предусмотрен режим *ограничения*. В этом режиме включаются счетчики обращений для каждого ядра. Если значение счетчика обращений от ядра, обладающего высшим приоритетом, превышает заданный лимит, то высший приоритет автоматически передается следующему ядру, осуществляющему обращение к памяти. Если обращений со стороны других ядер нет – счетчик сбрасывается, и передачи приоритета не происходит.

В *статическом* режиме приоритет ядер задается явно.

Регистры управления локальными арбитрами располагаются в каждом из DSP ядер и задают режим работы соответствующего локального арбитра.

2.3.17.2 Назначение разрядов регистра ARBR

Назначение разрядов регистра ARBR приведено в таблице 2.14.

Таблица 2.14. ─ Назначение разрядов регистра ARBR

Номер разряда	Наименование разряда	Назначение
0	HEN	Включение режима определения высокой плотности потоков
1	DEN	Разрешение установки явного приоритета (статический режим)
2	LEN	Бит разрешения ограничителя
3, 6, 7	-	Резерв
4-5	DPTR	Номер ядра, обладающего наивысшим приоритетом
8-13	Limit	Максимальное значение счетчика обращений
14-15	-	резерв

HEN – Включение режима определения высокой плотности потоков. Используется в режиме захвата (LEN = 0). Если HEN = 1, то включаются счетчики, определяющие плотность обращений ядер к данному сегменту. Если плотность обращений хотя бы от одного ядра больше 75% – то при значениях HEN = 1 и LEN = 0 передача приоритета происходит каждый такт.

DEN – разрешение установки явного приоритета (статический режим). Если данный бит установлен в 1, то при возникновении конфликта приоритет отдается обращению от ядра, номер которого определяется битами DPTR.

DPTR – определяет номер ядра, обладающего наивысшим приоритетом при обращении к сегменту памяти данного DSP. DPTR = 0 задает высший приоритет для данного ядра, 1 – высший приоритет для соседа с меньшим номером, далее циклически в сторону уменьшения номера ядра.

LEN – бит разрешения ограничителя. Если данный бит установлен в 1, арбитр работает в режиме ограничения, если бит установлен в 0 арбитр работает в режиме захвата.

Limit – задает максимальное значение счетчика обращений, в режиме ограничения. В этом режиме предусмотрена автоматическая смена приоритета.

2.3.17.3 Механизм передачи приоритета

Передача приоритета осуществляется циклически, между ядрами, осуществляющими обращение к памяти. Механизм передачи приоритета срабатывает в следующих случаях:

- ядро, обладавшее высшим приоритетом, не обращается к текущему сегменту памяти,

- в режиме захвата при LEN = 0 и HEN = 1 плотность обращений хотя бы от одного ядра больше 75%.

- в режиме ограничения LEN = 1, если значение счетчика обращений от ядра с высшим приоритетом достигло значения Limit.

В статическом режиме передачи приоритета не осуществляется.

Начальное состояние регистра ARBR = 0x0F01.

2.3.18 Регистр спецфункций (SFR)

Регистр спецфункций SFR (32 разряда, запись/чтение) предназначен для реализации специальных вычислительных функций. Назначение разрядов регистра SFR определяется реализуемой функцией.

3. ФОРМАТЫ И ТИПЫ ДАННЫХ

3.1 Общие положения

3.1.1 DSP–ядро Elcore-30M обладает широкими возможностями по работе с различными типами числовых данных: числа с *фиксированной* точкой могут быть представлены с точностью 8/16/32/64 бит; числа с *плавающей* точкой могут быть представлены с точностью 24E8 (соответствует IEEE-754) или 32E16 (числа с плавающей точкой повышенной точности).

В дальнейшем будут использоваться следующие определения форматов и типов данных.

Формат данных – характеристика размера и структуры представления числа в памяти (регистрах) данных DSP–ядра Elcore-30M.

Тип данных – правило, устанавливающее соответствие между математическим значением числа и его представлением в памяти.

Иными словами, формат – это способ размещения, а тип – это способ интерпретации данных.

DSP-ядро Elcore-30M поддерживает обработку различных типов чисел с фиксированной точкой - для целочисленных/дробных, знаковых/беззнаковых, действительных/комплексных чисел.

3.1.2 Форматы с фиксированной точкой:

Для чисел с фиксированной точкой используются следующие форматы данных:

- короткий (16-разрядный) формат (*short*);
- длинный (32-разрядный) формат (*long*);
- длинный двойной (64-разрядный) формат (*__int64*);
- комплексный 32-разрядный формат (*X32*) – (32+32);
- комплексный 16-разрядный формат (*X16*) – (16+16);
- комплексный 8-разрядный формат (*X8*) – (8+8);
- байтовый (8-разрядный) формат (*char*).

3.1.3 Форматы с плавающей точкой

3.1.3.1 Типы данных с плавающей точкой предназначены для высокоточной обработки данных, изменяющихся в больших динамических диапазонах. DSP–ядро Elcore-30M поддерживает два формата для чисел с плавающей точкой:

- 32-разрядный формат (*24E8*), соответствующий типу float (спецификации IEEE754);
- расширенный 48-разрядный формат (*32E16*) для представления чисел с плавающей точкой повышенной точности (тип double).

Ниже приводится определение указанных форматов и соответствующих им типов данных, поддерживаемых DSP–ядром Elcore-30M.

3.2 Форматы данных для чисел с фиксированной точкой

3.2.1 16-разрядный формат

Короткий (шестнадцатиразрядный) формат размещается в шестнадцатиразрядном регистре RF, либо занимает половину слова памяти данных.

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

Формат предназначен для хранения следующих типов данных:

- шестнадцатиразрядное целое со знаком (short) в дополнительном коде;
- шестнадцатиразрядное целое без знака (unsigned short);
- шестнадцатиразрядное дробное (fractional short) в дополнительном коде.

3.2.1.1 Тип short

Тип short определяет 16-разрядные целые числа со знаком в дополнительном коде. Значение числа определяется формулой

$s = -s[15]*2^{15} + s[14]*2^{14} + s[13]*2^{13} +\ldots+ s[1]*2^{1} + s[0]*2^{0}$

Некоторые возможные значения для чисел данного типа приведены в таблице 3.1.

Таблица 3.1. ─ 16-разрядные целые числа со знаком

Шестнадца-теричный код	Значение (десятичное)
0x8000	-32768 (минимальное значение)
0xFFFF	-1
0x0000	0
0x0001	1
0x7FFF	32767 (максимальное значение)

Данный тип является основным для 16-разрядного формата, то есть, если не оговаривается иное, операции в формате short используют числа данного типа.

3.2.1.2 Тип unsigned short

Тип unsigned short определяет 16-разрядные целые числа без знака. Значение числа определяется формулой

$s = s[15]*2^{15} + s[14]*2^{14} + s[13]*2^{13} +\ldots+ s[1]*2^{1} + s[0]*2^{0}$

Некоторые возможные значения для чисел данного типа приведены в таблице 3.2.

Таблица 3.2. ─ 16-разрядные целые числа без знака

Шестнадцатиричный код	Значение (десятичное)
0xFFFF	65535 (максимальное значение)
0x8000	32768
0x7FFF 0x0001	32767
0x0000	1
	0 (минимальное значение)

Данный тип используется как входной в операции MPUU. Кроме того, 16-разрядные целые числа без знака могут использоваться для подготовки адресных указателей.

3.2.1.3 Тип fractional short

Тип fractional short определяет 16-разрядные дробные числа со знаком в дополнительном коде. Значение числа определяется формулой

s = -s[15]*20 + s[14]*2-1 + s[13]*2-2 +…+ s[1]*2-14 + s[0]*2-15

Некоторые возможные значения для чисел данного типа указаны в таблице 3.3.

Таблица 3.3. ─ 16-разрядные дробные числа

Шестнадцатиричный код	Значение (десятичное)
0x8000	-1 (минимальное значение)
0xFFFF	-2^{-15}
0x0000	0
0x0001	2^{-15}
0x7FFF	$2^{-1} + 2^{-2} + \ldots + 2^{-14} + 2^{-15} = 1 - 2^{-15}$ (максимальное значение)

Данный тип используется как входной в операциях MPF, MPF2, MPF2S.

3.2.2 32-разрядный формат

Длинный (32-разрядный) формат размещается в 32-разрядном регистре RF, либо в одном из регистров-аккумуляторов ACi, либо занимает одно слово памяти данных.

31	30	29	28	27	26	25	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
S																															

Формат предназначен для хранения следующих типов данных:

- 32-разрядное целое со знаком (long) в дополнительном коде;

- 32-разрядное целое без знака (unsigned long);

- 32-разрядное дробное (fractional long) в дополнительном коде.

3.2.2.1 Тип long

Тип long определяет 32-разрядные целые числа со знаком в дополнительном коде. Значение числа определяется формулой

$S = -S[31]*2^{31} + S[30]*2^{30} + S[29]*2^{29} +\ldots+ S[1]*2^{1} + S[0]*2^{0}$

Некоторые возможные значения для чисел данного типа приведены в таблице 3.4.

Таблица 3.4. 32-разрядные целые числа со знаком

Шестнадцате-ричный код	Значение (десятичное)
0x80000000	-2147483648 (минимальное значение)
0xFFFFFFFF	-1
0x00000000	0
0x00000001	1
0x7FFFFFFF	2147483647 (максимальное значение)

Данный тип является основным для 32-разрядного формата, то есть, если не оговаривается иное, операции в формате long используют числа данного типа.

3.2.2.2 Тип unsigned long

Тип unsigned long определяет 32-разрядные целые числа без знака. Значение числа определяется формулой

$S = S[31]*2^{31} + S[30]*2^{30} + S[29]*2^{29} +\ldots+ S[1]*2^{1} + S[0]*2^{0}$

Некоторые возможные значения для чисел данного типа приведены в таблице 2.5.

Таблица 3.5. ─ 32-разрядные целые числа без знака

Шестнадцате-ричный код	Значение (десятичное)
0xFFFFFFFF	4294967295(максимальное значение)
0x80000000	2147483648
0x7FFFFFFF	2147483647
0x00000001	1
0x00000000	0 (минимальное значение)

Данный тип является выходным для операции MPUU.

3.2.2.3 Тип fractional long

Тип fractional long определяет 32-разрядные дробные числа со знаком в дополнительном коде. Значение числа определяется формулой

$S = -S[31]*2^{0} + S[30]*2^{-1} + S[29]*2^{-2} +\ldots+ S[1]*2^{-30} + S[0]*2^{-31}$

Некоторые возможные значения для чисел данного типа приведены в таблице 3.6.

Таблица 3.6. ─ 32-разрядные дробные числа

Шестнадцате-ричный код	Значение (десятичное)
0x80000000	-1 (минимальное значение)
0xFFFFFFFF	-2^{-31}
0x00000000	0
0x00000001	2^{-31}
0x7FFFFFFF	$2^{-1} + 2^{-2} + \ldots + 2^{-30} + 2^{-31} = 1 - 2^{-31}$ (максимальное значение)

Данный тип является выходным в операции MPF.

3.2.3 64-разрядный формат

Длинный двойной 64-разрядный размещается в 64-разрядных регистрах RF, либо в 64-разрядных регистрах-аккумуляторах, либо занимает два слова памяти данных.

31	30	29	28	27	26	25	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
S																															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
S																															

Формат предназначен для хранения следующего типа данных:

- 64-разрядное целое со знаком (__int64) в дополнительном коде.

Значение числа этого типа определяется формулой:

$$S = -S[63]*2^{63} + S[62]*2^{62} + S[61]*2^{61} + \ldots + S[1]*2^{1} + S[0]*2^{0}$$

Некоторые возможные значения для чисел данного типа приведены в таблице 3.7.

Таблица 3.7. ─ 64-разрядные целые числа со знаком

Шестнадцате-ричный код	Значение (десятичное)
0x80000000	-2147483648 (минимальное значение)
0xFFFFFFFF	-1
0x00000000	0
0x00000001	1
0x7FFFFFFF	2147483647 (максимальное значение)

Данный тип является выходным, в частности, в операциях умножения с накоплением MAC, MACL и умножения MPYL.

Примечание - Для выполнения арифметических действий над данными, представленными в разрядной сетке свыше 32 разрядов, к примеру, над 48- или 64–разрядными данными, предусмотрены также операции сложения/вычитания с учетом запоминаемого бита переноса – ADC, ADCL, SBC, SBCL.

3.2.4 Комплексный 16-разрядный формат

Комплексный 16-разрядный формат используется для представления пары 16-разрядных дробных или целых чисел вида (Re,Im). Размещается в 32-разрядном регистре RF, либо занимает одно слово памяти данных.

31	30	29	28	27	26	25	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
Re																Im															

Формат предназначен для хранения следующих типов данных:

- пара 16-разрядных целых чисел (Re,Im) со знаком в дополнительном коде (X16);

- пара 16-разрядных дробных чисел (Re,Im) со знаком в дополнительном коде (fractional X16).

Пара чисел (Re,Im) представляет собой две компоненты – действительную и мнимую – комплексного числа. При этом действительная часть размещается в старшем полуслове, а мнимая часть – в младшем.

3.2.4.1 Тип X16 определяет пару 16-разрядных целых чисел (Re,Im) со знаком в дополнительном коде.

Значение компонент (Re,Im) определяется формулами

$Re = -S[31]*2^{15} + S[30]*2^{14} + S[29]*2^{13} +\ldots+ S[17]*2^{1} + S[16]*2^{0}$,

$Im = -S[15]*2^{15} + S[14]*2^{14} + S[13]*2^{13} +\ldots+ S[1]*2^{1} + S[0]*2^{0}$.

Некоторые возможные значения для чисел данного типа приведены в таблице 3.8.

Таблица 3.8. ─ Комплексные 16-разрядные целые числа

Шестнадцате-ричный код	Значение (десятичное)
0x0000FFFF 0x80000001 0x7FFF0000 0x7FFE8001	(0,-1) (-32768,1) (32767,0) (32766,-32767)

3.2.4.2 Тип fractional X16 определяет пару 16-разрядных дробных чисел (Re,Im) со знаком в дополнительном коде.

Значение компонент (Re,Im) определяется формулами

$Re = -S[31]*2^{0} + S[30]*2^{-1} + S[29]*2^{-2} +\ldots+ S[17]*2^{-14} + S[16]*2^{-15}$,

$Im = -S[15]*2^{0} + S[14]*2^{-1} + S[13]*2^{-2} +\ldots+ S[1]*2^{-14} + S[0]*2^{-15}$.

Некоторые возможные значения для чисел данного типа указаны в таблице 3.9.

Таблица 3.9. ─ Комплексные 16-разрядные дробные числа

Шестнадцате-ричный код	Значение
0x0000FFFF 0x80000001 0x7FFF0000 0x7FFE8001	$(0,-2^{-15})$ $(-1,2^{-15})$ $(1-2^{-15},0)$ $(1-2^{-14},-1+2^{-15})$

Данный тип является выходным в операциях MPX, MMACX.

3.2.5 Комплексный 32-разрядный формат

Комплексный 32-разрядный формат используется для представления пары 32-разрядных дробных или целых чисел вида (Re,Im). Размещается в 64-разрядном регистре RF, либо занимает два слова памяти данных.

31	30	29	28	27	26	25	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
Im																															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Re																															

Формат предназначен для хранения следующих типов данных:

- пара 32-разрядных целых чисел (Re,Im) со знаком в дополнительном коде (X32);

Пара чисел (Re,Im) представляет собой две компоненты – действительную и мнимую – комплексного числа. При этом действительная часть размещается в старшем полуслове, а мнимая часть – в младшем.

3.2.5.1 Тип X32 определяет пару 32-разрядных целых чисел (Re,Im) со знаком в дополнительном коде.

Значение компонент (Re,Im) определяется формулами

$Re = -S[63]*2^{31} + S[62]*2^{30} + S[61]*2^{29} + \ldots + S[33]*2^{1} + S[32]*2^{0}$,

$Im = -S[31]*2^{31} + S[30]*2^{30} + S[29]*2^{29} + \ldots + S[1]*2^{1} + S[0]*2^{0}$.

Некоторые возможные значения для чисел данного типа приведены в таблице 3.10.

Таблица 3.10. ─ Комплексные 32-разрядные целые числа

Шестнадцатеричный код	Значение (десятичное)
0x00000000_FFFFFFFF	(0,-1)
0x80000000_00000001	(-2147483648,1)
0x7FFFFFFF_00000000	(2147483647,0)
0x7FFFFFFE_80000001	(2147483646,- 2147483647)

Данный тип используется в операциях ASLXL, ASRXL.

3.2.6 Комплексный восьмиразрядный формат

Комплексный 8-разрядный (байтовый) формат используется для представления двух пар 8-разрядных дробных или целых чисел вида (Re1,Im1) и (Re0,Im0). Формат X8 размещается в 32-разрядном регистре RF, либо занимает одно слово памяти данных, и упакован.

31	30	29	28	27	26	25	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
Re1								Re0								Im1								Im0							

Формат предназначен для хранения следующих типов данных:

- две пары восьмиразрядных целых чисел (Re1,Im1) и (Re0,Im0) со знаком в дополнительном коде (X8);

- две пары восьмиразрядных дробных чисел (Re1,Im1) и (Re0,Im0) со знаком в дополнительном коде (fractional X8).

3.2.6.1 Тип X8 определяет две пары восьмиразрядных целых чисел (Re1,Im1) и (Re0,Im0) со знаком в дополнительном коде.

Каждая пара составляет комплексное число. Значения компонент чисел определяются формулами

$Re1 = -S[31]*2^7 + S[30]*2^6 + S[29]*2^5 +\ldots+ S[25]*2^1 + S[24]*2^0$,

$Im1 = -S[15]*2^7 + S[14]*2^6 + S[13]*2^5 +\ldots+ S[9]*2^1 + S[8]*2^0$,

$Re0 = -S[23]*2^7 + S[22]*2^6 + S[21]*2^5 +\ldots+ S[17]*2^1 + S[16]*2^0$,

$Im0 = -S[7]*2^7 + S[6]*2^6 + S[5]*2^5 +\ldots+ S[1]*2^1 + S[0]*2^0$.

Некоторые возможные значения для чисел данного типа приведены в таблице 3.11.

Таблица 3.11. ─ Пара восьмиразрядных комплексных целых чисел

Шестнадцатеричный код	Значение	
	(Re1,Im1)	(Re0,Im0)
0x0001FFFE	(0,-1)	(1,-2)
0x80000201	(-128,2) (127,-128)	(0,1)
0x7FFF8000	(127,-128)	(-1,0)
0x7FFE8001		(-2,1)

Данный тип используется как входной в операции MMACX.

3.2.6.2 Тип fractional X8 определяет две пары восьмиразрядных дробных чисел (Re,Im) со знаком в дополнительном коде

Каждая пара составляет комплексное число. Значения компонент определяются формулами

$Re1 = -S[31]*2^0 + S[30]*2^{-1} + S[29]*2^{-2} +\ldots+ S[25]*2^{-6} + S[24]*2^{-7}$,

$Im1 = -S[15]*2^0 + S[14]*2^{-1} + S[13]*2^{-2} +\ldots+ S[9]*2^{-6} + S[8]*2^{-7}$,

$Re0 = -S[23]*2^0 + S[22]*2^{-1} + S[21]*2^{-2} +\ldots+ S[17]*2^{-6} + S[16]*2^{-7}$,

$Im0 = -S[7]*2^0 + S[6]*2^{-1} + S[5]*2^{-2} +\ldots+ S[1]*2^{-6} + S[0]*2^{-7}$.

Некоторые возможные значения для чисел данного типа указаны в таблице 3.12.

Таблица 3.12. ─ Пара восьмиразрядных комплексных дробных чисел

Шестнадцатеричный код	Значение	
	(Re1,Im1)	(Re0,Im0)
0x0001FFFE	$(0,-2^{-7})$	$(2^{-7},-2^{-6})$
0x80000201	$(-1,2^{-6})$	$(0,2^{-7})$
0x7FFF8000	$(1-2^{-7},-1)$	$1-2^{-6},0)$
0x7FFE8001	$(1-2^{-7},-1)$	$(1-2^{-6},2^{-7})$

Данный тип используется как входной в операциях MPX, MMACX.

3.2.7 8-разрядный формат

8-разрядный (байтовый) формат используется для представления 8-разрядных целых чисел со знаком (char) или без знака (unsigned char). Четыре операнда байтового формата размещаются в 32-разрядном регистре RF, либо в одном слове памяти данных.

31	30	29	28	27	26	25	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
t_3								t_1								t_2								t_0							

Формат предназначен для хранения следующих типов данных:

- 8-разрядное целое со знаком (char) в дополнительном коде;
- 8-разрядное целое без знака (unsigned char);

3.2.7.1 Тип char

Тип char определяет 8-разрядные целые числа со знаком в дополнительном коде. Значение числа определяется формулой

$s = -s[7]*2^7 + s[6]*2^6 + s[5]*2^5 + \ldots + s[1]*2^1 + s[0]*2^0$

Некоторые возможные значения для чисел данного типа приведены в таблице 3.1.

Таблица 3.13. 8-разрядные целые числа со знаком

Шестнадца-теричный код	Значение (десятичное)
0x80	-128 (минимальное значение)
0xFF	-1
0x00	0
0x01	1
0x7F	127 (максимальное значение)

3.2.7.2 Тип unsigned char

Тип unsigned short определяет 16-разрядные целые числа без знака. Значение числа определяется формулой

$s = s[7]*2^7 + s[6]*2^6 + s[5]*2^5 + \ldots + s[1]*2^1 + s[0]*2^0$

Некоторые возможные значения для чисел данного типа приведены в таблице 3.2.

Таблица 3.14. ─ 8-разрядные целые числа без знака

Шестнадцати-ричный код	Значение (десятичное)
0xFF	255 (максимальное значение)
0x80	-256
0x7F	255
0x01	1
0x00	0 (минимальное значение)

3.2.8 Граничные значения для типов данных с фиксированной точкой

3.2.8.1 Во всех знаковых типах с фиксированной точкой данные представлены *в дополнительном коде*. В таблице 3.15 приводятся граничные значения для указанных типов чисел. В случае комплексных чисел приводимые граничные значения относятся к каждой из компонент.

Таблица 3.15. ─ Граничные значения различных типов чисел

Граничные значения		Форматы			
		8 разрядов	16 разрядов	32 разряда	64 разряда
Наименьшее значение	Шестнадцатеричное представление	0x80	0x8000	0x80000000	0x8000000000000000
	дробное	-1.0	-1.0	-1.0	-1.0
	целое	-128	-32768	-2147483648	-9223372036854775808
Наибольшее значение	Шестнадцатеричное представление	0x7F	0x7FFF	0x7FFFFFFF	0x7FFFFFFFFFFFFFFF
	дробное	$1 - 2^{-7}$	$1 - 2^{-15}$	$1 - 2^{-31}$	$1 - 2^{-63}$
	целое	127	32767	2147483647	9223372036854775807

Граничные значения используются при выполнении арифметических операций в режиме насыщения (Saturation). Включение данного режима означает присвоение результату операции граничного значения в случае выхода результата за пределы разрешенного диапазона.

3.2.9 Особенности выполнения операций умножения для целых и дробных чисел

Представление результата операции над целыми (integer) числами или дробями (fractional) – одинаково для операций сложения или вычитания, но различается при умножении. Различие в формировании умножения двух чисел в целом и дробном формате проиллюстрировано на рисунке 3.1.

Ключевое различие между целочисленным и дробным умножением находится в представлении (2N – 1) - разрядного результата. *В дробном умножении* (2N – 1) значащие биты результата должны быть выровнены по левой границе, а младший бит должен быть заполнен нулем, чтобы обеспечить дробное представление. *В целочисленном умножении* (2N – 1) значащие биты результата должны быть выровнены по правому краю и дублируется знаковый разряд.

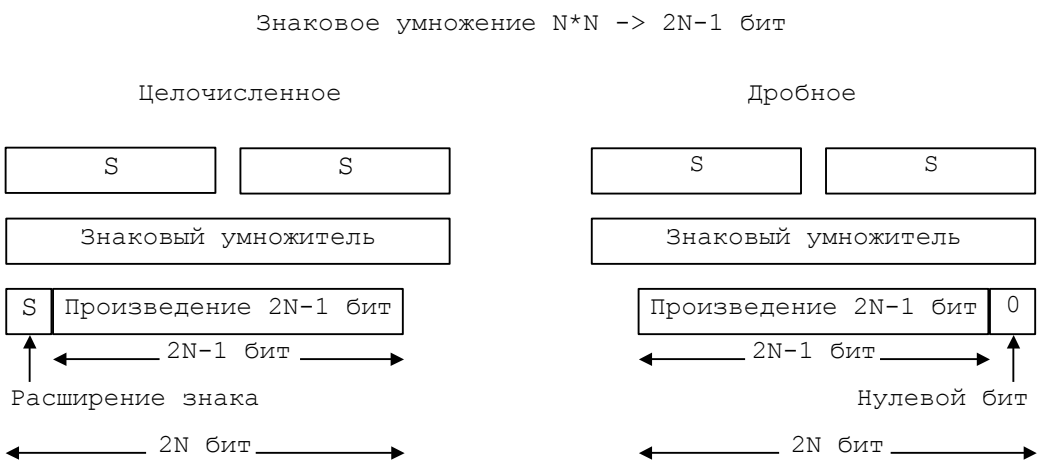


Рисунок 3.1 ─ Различие целочисленных / дробных умножений

3.3 Форматы данных для чисел с плавающей точкой

3.3.1 Формат 24E8 (тип float) IEEE-754

3.3.1.1 32-разрядный формат (*24E8*) предназначен для хранения чисел с плавающей точкой типа float, соответствующих спецификации IEEE-754 с некоторыми ограничениями, указанными ниже. Формат размещается в 32-разрядном регистре или занимает одно слово памяти данных.

31	30	29	28	27	26	25	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
s	e								f																						

Формат имеет структуру, состоящую из трех полей:

`s = S[31]` – знак числа;

`e[7:0] = S[30:23]` – экспонента (порядок) числа со смещением +127;

`f[22:0] = S[22:0]` – дробная часть мантиссы числа.

Значение числа X типа float вычисляется по формуле

$X = (-1)^s * 2^E * F$,

где **E** – экспонента без смещения

$E = e-127 = e[7]*2^7+e[6]*2^6+e[5]*2^5+e[4]*2^4+e[3]*2^3+e[2]*2^2+e[1]*2^1+e[0]*2^0–127$

F – полная мантисса:

$F = 1 + f[22]*2^{-1} + f[21]*2^{-2} + \ldots + f[1]*2^{-22} + f[0]*2^{-23}$;

$F \geq 1.0$, т.е. мантисса может быть только нормализованной и только положительной. Бит, соответствующий 1.0, “умалчивается”.

Некоторые возможные значения для чисел данного типа приведены в таблице 3.16.

Таблица 3.16 ─ Числа в формате с плавающей точкой (тип float) IEEE-754

s	e	f	Значение числа
0/1	$1 \leq e \leq 254$	x	Числа с плавающей точкой
0	255	0	$+\infty$
1	255	0	$-\infty$
X	255	xx..x, f≠0	NaN: не - число общего вида, только входное

s	e	f	Значение числа
0	255	11..1	QNaN: выходное не – число
			Примеры чисел:
1	254	11..1	Минимальное число $-2^{127}(1.0+1/2^1+..+1/2^{23})$
1	127	10..0	-1.5
1	127	00..0	-1.0
1	1	00..0	Максимальное отрицательное число -2^{-126}
0	0	0	Нуль, входной или выходной
0/1	0	0	Нуль, только входной
0	1	00..0	Минимальное положительное число 2^{-126}
0	127	00..0	1.0
0	127	10..0	1.5
0	254	11..1	Максимальное число $2^{127}(1.0+1/2^1+..+1/2^{23})$

3.3.1.2 Случаи формирования результата QNaN:

- входной операнд NaN;
- входной операнд QNaN;
- $(\pm\infty)*0$;
- $(+\infty)+(-\infty)$, $(-\infty)+(+\infty)$, $(+\infty)-(+\infty)$, $(-\infty)-(-\infty)$.

3.3.1.3 Генерация запросов на прерывание при выполнении команд в формате с плавающей точкой

К исключениям при выполнении команд в формате с плавающей точкой относятся следующие случаи:

- формирование результата $+\infty$;
- формирование результата $-\infty$;
- формирование результата QNaN.

В этих случаях формируется признак результата V=1 и запрос на прерывание (29-й разряд регистра IRQR). При открытой маске (29-й разряд регистра MASKR равен «1») это вызывает переход на обработку прерывания (п.2.3.12).

3.3.1.4 Способы округления

Во всех специально не оговоренных случаях используется один вариант округления: к ближайшему числу (при равноудаленности – к четному, с нулевым младшим битом мантиссы) (см.таблицу 3.17).

Таблица 3.17. ─ Способы округления

f до округления	f после округления	Пояснение
100x 00	100x	округление не требуется
100x 01	100x	округление к меньшему
1000 10	1000	равноудаленность, округление к четному
1001 10	1010	равноудаленность, округление к четному
1000 11	1001	округление к большему
1001 11	1010	округление к большему

3.3.1.5 Ограничения

Тип float, реализованный в DSP Elcore-30M, соответствует спецификации IEEE-754 со следующими ограничениями:

− денормализованные числа не обрабатываются, на выходе не бывает денормализованного результата, он заменяется нулем;

− не используется в качестве результата знаковый ноль;

− не используется подраздел из спецификации IEEE-754 работы с не числами, а лишь фиксируется не число в виде результата и оно всегда имеет вид QNaN и не имеет знака;

− используется одна мода округления – к ближайшему четному – а не четыре, предложенных в стандарте;

− результат команды FINR (обратной величины квадратного корня) из нуля со знаком минус не равен минус бесконечности, как должно быть по спецификации IEEE-754, а равен QNaN;

− не реализованы предусмотренные стандартом исключения “UNDERFLOW”, “INEXACT”.

3.3.2 Формат 32E16 (тип double)

3.3.2.1 Расширенный 48-разрядный формат (*32E16*), или E-формат, предназначен для представления чисел с плавающей точкой повышенной точности (тип double).

Формат имеет структуру, состоящую из двух полей:

E[15:0] – экспонента (порядок) числа – 16-разрядное целое со знаком в дополнительном коде;

F[31:0] – мантисса числа – 32-разрядное дробное число в дополнительном коде.

Мантисса F размещается в 32-разрядном регистре RF или одной ячейке памяти; экспонента - в 16-разрядном регистре RF или младшем полуслове ячейки памяти.

31	30	29	28	27	26	25	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
F																															

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

Значение числа X типа double, за исключением нескольких специальных случаев, рассмотренных ниже, вычисляется по формуле

$$X = 2E * F, \qquad (2.21)$$

где $E = - E[15]*2^{15} + E[14]*2^{14} + E[13]*2^{13} + \ldots + E[1]*2^{1} + E[0]*2^{0}$;

$F = - F[31]*2^{0} + F[30]*2^{-1} + F[29]*2^{-2} + \ldots + F[1]*2^{-30} + F[0]*2^{-31}$;

мантисса может быть только нормализованной.

В таблице 3.18 указаны специальные случаи для чисел типа double.

Таблица 3.18. ─ Расширенный 48-разрядный формат с плавающей точкой

E[15:0]	F[31:0]	Назначение
0x8000	Любое число	число 0
32767=0x7FFF	0x00000000	$+\infty$
32767=0x7FFF	0x80000000	$-\infty$
32767=0x7FFF	0x7FFFFFFF	QNaN

32767=0x7FFF	Любое число, не равное 0x7FFFFFFF	NaN
0x8001 ≤ E ≤ 0x7FFE	0x40000000 ≤ F ≤ 0x7FFFFFFF 0x80000000 ≤ F ≤ 0xFFFFFFFF	число>0 число<0

Некоторые примеры записи чисел данного типа приведены в таблице 3.19.

Таблица 3.19. ─ Примеры чисел в 48-разрядном формате с плавающей точкой

E	F	Значение
0x8000	0x00000000	0
0x0000	0x40000000	+0.5
0xFFFF	0x80000000	-0.5
0x0001	0x40000000	+1.
0x0000	0x80000000	-1.
0x0004	0x54000000	+10.5
0x0004	0xAC000000	-10.5
0x0007	0x7F800000	+127.5
0x0008	0x40000000	+128.
0x0008	0x40400000	+128.5
0x0007	0x80800000	-127.5
0x0007	0x80000000	-128.
0x0008	0xBFC00000	-128.5
0x0002	0x6487ED50	π
0x0002	0x56FC2A2C	e

Для преобразования чисел различных типов предусмотрены операции CVEF, CVFE, CVFI, CVIF.

3.4 Запись различных типов констант в операндах и памяти

3.4.1 Средства программирования и отладки DSP-ядра Elcore-30M обеспечивают поддержку записи различных типов констант в память и использования их в качестве непосредственных операндов. В таблице 3.20 приведены синтаксические правила и примеры задания различных типов констант в ассемблере DSP-ядра Elcore-30M.

Таблица 3.20

Тип данных	Непосредственный операнд	Константа в памяти
Целый 16-разрядный (short)	#s *Пример - -32767*	.dw #s *Пример - .dw -32767*
Целый 32-разрядный (long)	#S *Пример - -32767*32767*	.dl #S *Пример - .dl 0x80000000*
Целый комплексный (X16) (16+16)	#[Re,Im] *Пример - [-32767,0x8000]*	.dl #[Re,Im] *Пример - .dl [-32767,0x8000]*
Целый комплексный байтный (X8) (8+8+8+8)	#[[@Re1,Re0],[@Im1,Im0]] *Пример - [[@-1,0],[@17,0x80]]*	.dl #[[@Re1,Re0],[@Im1,Im0]] *Пример -.dl[[@1,0],[@17,0x8]]*
Дробный 16-разрядный (fractional short)	#s *Пример - -0.875*	.fr #s *Пример - .fr* 0.99999
Дробный 32-разрядный (fractional long)	#S *Пример - 0.875*	.frl #S *Пример - .frl -0.999999999*
Дробный комплексный (fractional X16)	#[Re,Im] *Пример - [0.875,-0.375]*	.frl #[Re,Im] *Пример - .frl [-0.375,0.875]*
Дробный комплексный байтный (fractional X8) (8+8+8+8)	#[[@Re1,Re0],[@Im1,Im0]] *Пример - [[@0.5,-0.5],[@0.25,-0.5]]*	.frl #[[@Re1,Re0],[@Im1,Im0]] *Пример - .frl [[@-0.5,0.2],[@0.17,0.8]]*
Плавающая точка 24E8 (float)	#S *Пример - # -2.75*	.real #S *Пример - .real -3.7e6*
Плавающая точка 32E16 (double)	-	*.double* #S *Пример - .double -31.25e-1*

4. РЕЖИМЫ ВЫЧИСЛЕНИЙ И ПРИЗНАКИ РЕЗУЛЬТАТА ОПЕРАЦИИ

4.1 Режим насыщения (Saturation)

Режим насыщения (Saturation) включается 8-м битом регистра CCR. Включение данного режима подразумевает присвоение результату операции граничного значения в случае выхода результата за пределы разрешенного диапазона. В таблицах 4.1, 4.2 приведён перечень операций, в которых может быть использован режим насыщения.

Таблица 4.1. ─ Операции с режимом насыщения (базовая система инструкций).

Long	Short	Complex
Блок MS		
-	MPF	-
-	MPF2	-
-	MPF2S	-
-	-	MPX
ASLL	ASL	ASLX
Блок AU		
ABSL	ABS	-
ADCL	ADC	-
ADC16L	AD1	-
ADDL	ADD	ADDX
ADDLR	ASH	-
ADDLRTR	-	-
ADDSUBL	ADDSUB	ADDSUBX
FTRL	-	-
NEGL	NEG	-
RNDL	-	-
SUBL	SUB	SUBX
SBCL	SBC	-
SUBLR	SAH	-
SUBLRTR	-	-

Таблица 4.2. ─ Операции с режимом насыщения (расширение системы инструкций).

Long	Short	Char	X16	X32	__Int64
Операция OP2e					
				ASLXL	ASLD
					ASLDi
Операция OP1e					
ALL2	A4	AB16	ASX2		ADCD
ALL4	A8	SB16	ASXS2		ADDD
AL2	A8s	MFB16	ASXS		ADDLD
AL4	S4	UMFB16	BF4		SBCD
ALLFT41	S8		BIF4		SUBD
ALLFT22	S8s		AXJ4		

Окончание таблицы 4.2

Long	Short	Char	X16	X32	__Int64
SLL2	MS2		SXJ4		

SLL4	MS4		MFX		
ALL41	MS8		MFX2		
	A81		MFXC		
	A42		MFXC2		
	A24				
	SGA8				
	SGA4				
	MFA21				
	MFA22				
	MFA24				
	MFA41				
	MFA42				
	MFA81				

4.2 Режим округления (Rounding)

Округление (Rounding) может выполняться как самостоятельная операция (RNDL), либо в составе более сложных операций для преобразования 32-разрядных чисел в 16-разрядные.

Для базовой системы инструкций округление может выполняться одним из двух способов: округление к ближайшему (Convergent Rounding) при RND=CCR[9]=0 и округление дополнительного кода (Two's-Complement Rounding) при RND=CCR[9]=1. Для расширения системы инструкций (команды ALLFT41, ALLFT22) выполняется либо округление к ближайшему при RND=0, либо отсечение при RND=1.

Перечень операций, в которых используется округление, приведен в таблице 4.3.

Таблица 4.3. ─ Операции с режимом округления.

Long	*Short*	*Complex*
Базовая система инструкций (OP1)		
RNDL		
ADDLR		
SUBLR		
ADDLRTR		
SUBLRTR		
FTRL		
Расширение системы инструкций (OP1e)		
ALLFT41		
ALLFT22		

4.2.1 Режим округления к ближайшему четному (Convergent Rounding)

Округление к ближайшему (также называется "к самому близкому четному числу") – способ округления по умолчанию.

Традиционный метод округления округляет вверх при большем значении числа, чем половина, и округляет вниз для любого значения меньше, чем половина. Вопрос возникает только относительно того, как эта половина должна быть округлена. Если это всегда будет округляться одним способом, то результаты в конечном счете будут смещены в том же направлении.

Округление к ближайшему четному решает эту проблему так:

— округление осуществляется в меньшую сторону, если число четное (младший бит = 0).

— округление выполняется в большую сторону, если число нечетно (МЛАДШИЙ БИТ = 1).

В в качестве примера - алгоритм округления 16-разрядного результата описывается следующим логическим выражением:

r=(~R[15]| (~R[16]&R[15]&(~(|R[14:0])))?0’b:1’b;

где: r - единица округления;

R – округляемые данные.

4.2.2 Режим округления дополнительного кода (Two’s-Complement Rounding)

Все значения, большие или равные половине, округляются вверх, а все меньшие, чем половина, округлены в меньшую сторону.

В результате алгоритм округления 16-разрядного результата описывается следующим логическим выражением:

r=(~R[15])? 0’b: 1’b;

где: r - единица округления;

R – округляемые данные.

4.3 Режим масштабирования (Scaling)

4.3.1 Масштабирование позволяет избежать переполнения при выполнении арифметических операций путем сдвига вправо полученного результата.

Этот режим может быть полезен, в частности, при реализации алгоритма БПФ с прореживанием по частоте (Decimation-In-Frequency), когда при выполнении операций сложения/вычитания над комплексными числами необходимо избежать переполнения на выходе сумматора.

Масштабирование выполняется путем арифметического сдвига результата операции вправо на 0/1/2/3 разряда (на 3 разряда – только для команды **A81**), при этом величина сдвига определяется полем SC (разряды 9, 8) регистра PDNR.

Включение режима масштабирования может быть выполнено путем установки в «1» бита 15 (ESC) регистра PDNR.

Другой способ включения этого режима применяется только для базовой системы инструкций и состоит в установке в «1» поля M непосредственно в командном слове (форматы 8a-8d). Синтаксически это выражается в добавлении к мнемоническому имени команды суффикса “s”, например, ADDLs, SUBXs и т.п. Подробно синтаксис и кодирование инструкций рассматриваются в разделе 9.

Перечень операций, в которых может быть использован режим масштабирования, приведен в таблицах 4.4 и 4.5.

Таблица 4.4. — Операции с режимом масштабирования (базовая система инструкций).

Long	Short	X16
Блок AU		
ABSL	ABS	-
NEGL	NEG	-
ADDL	ADD	ADDX
SUBL	SUB	SUBX
ADCL	ADC	-
ADC16L	AD1	-
SBCL	SBC	-
ADDSUBL	ADDSUB	ADDSUBX
RNDL	-	-
ADDLR	ASH	-
SUBLR	SAH	-
ADDLRTR	-	-
SUBLRTR	-	-
FTRL	-	-

Таблица 4.5. ─ Операции с режимом масштабирования (расширение системы инструкций).

Long	Short	X16	__Int64
ALL2	A4	ASX2	ADCD
ALL4	A8	ASXS2	ADDD
AL2	A8s	ASXS	ADDLD
AL4	S4	BF4	SBCD
SLL2	S8	BIF4	SUBD
SLL4	S8s	AXJ4	-
ALL41	MS2	SXJ4	-
-	MS4	-	-
-	MS8	-	-
-	A81	-	-
-	A42	-	-
-	A24	-	-
-	SGA8	-	-
-	SGA4	-	-

4.4 Режим измерения блочной экспоненты

4.4.1 Режим измерения блочной экспоненты позволяет автоматически, при помощи встроенных аппаратных средств, на фоне выполнения программы измерить параметр денормализации массива данных, пересылаемых из регистрового файла в память. Параметром денормализации масиива считается наименьший из параметров денормализации входящих в него элементов. Параметром денормализации элемента является количество разрядов слева до старшей значащей цифры без учета разряда знака.

Этот режим может быть полезен при реализации различных преобразований массивов данных в целочисленных форматах, когда, с одной стороны, необходимо избежать переполнения, с другой стороны, обеспечить максимальную точность вычислений.

Включение режима измерения блочной экспоненты выполняется путем установки в «1» бита 7 (Epdn) регистра PDNR.

Текущий код PDN фиксируется в поле Cpdn (разряды [4:0]) регистра PDNR.

Данные, входящие в анализируемый массив, могут иметь либо тип long, при этом этом 5-й разряд регистра PDNR должен быть установлен в «0» (F=PDNR[5]=0); либо комплексеый тип X16, при этом этом 5-й разряд регистра PDNR должен быть установлен в «1» (F=PDNR[5]=1).

4.5 Признаки результата

4.5.1 В таблице 4.6 приводятся стандартные правила формирования признаков результата вычислительной операции: U(unnormalized), N(negative), Z(zero), V(overflow), C(carry). Для отдельных операций некоторые признаки могут формироваться по иным специально оговоренным правилам. В дальнейшем при описании правил формирования признаков используются следующие обозначения: msb – номер старшего (знакового) разряда результата D, т.е msb=15 для 16-разрядных чисел, msb=31 для 32-разрядных чисел и msb=63 для 64-разрядных.

Кроме указанных выше основных признаков, при выполнении операций могут формироваться и некоторые дополнительные признаки, определение которых дано ранее при описании регистра CCR.

Таблица 4.6. ─ Стандартные правила формирования признаков результата операции

<table>
<tr><td rowspan="2">Признак</td><td colspan="2">Стандартные правила формирования признаков</td></tr>
<tr><td>Все вычислительные операции (кроме сдвига)</td><td>Операции сдвига:
ASL, ASLL, ASLX, ASR, ASRL, ASRX, ASRLE, LSL, LSLL, LSLX, LSR, LSRL, LSRX, ROL, ROLL, ROR, RORL</td></tr>
<tr><td>U</td><td colspan="2">Для знаковых типов: U = 0, если D[msb]≠D[msb-1]; U = 1, если D[msb]=D[msb-1].
Для беззнаковых типов: U = 0, если D[msb]≠0; U = 1, если D[msb]=0.</td></tr>
<tr><td>N</td><td colspan="2">N=D[msb];</td></tr>
<tr><td>Z</td><td colspan="2">Z = 1, если D=0;
Z = 0, если D≠0;</td></tr>
<tr><td>V</td><td>Целочисленные операции:
V = 1, если [msb+1]≠D[msb];
V = 0, если D[msb+1]=D[msb];
Операции в формате с плавающей точкой:
V = 1, если результат равен ±∞, QNaN;
V = 0, иначе.</td><td>Операции ASL, ASLL, ASLX, ASLD, ASLXL:
V = 0, если хотя бы один разряд, выдвигаемый за пределы разрядной сетки или на место знака, не равен знаку;
V = 1, иначе.</td></tr>
<tr><td>C</td><td>C = Cout[msb]</td><td>C принимает значения последнего из битов, выдвинутых за разрядную сетку результата D[msb:0] вправо или влево, в зависимости от направления сдвига. При нулевом сдвиге C = 0.</td></tr>
<tr><td colspan="3">Примечание. Целочисленные арифметические устройства выполнены как полные (msb+2)-разрядные сумматоры/вычитатели с дополнительным старшим разрядом под номером msb+1, используемым только для формирования признаков. На выход поступают (msb+1) младших разряда результата D[msb:0]. Каждый из (msb+2) каскадов сумматора формирует как соответствующий бит результата D[i], так и перенос в следующий разряд Cout[i], i=0,1,…, msb+1.</td></tr>
</table>

5. БАЗОВАЯ СИСТЕМА ИНСТРУКЦИЙ

5.1 Общая характеристика

5.1.1 Система инструкций DSP-ядра Elcore-30M включает в себя базовую часть (или базовую систему инструкций), описание которой приводится в настоящем разделе, и расширение, описание которого дается в следующем разделе. Базовая система инструкций оперирует с данными, формат которых не превышает 32 разряда, расширение системы инструкций позволяет оперировать с 64/128 разрядными данными.

Система инструкций DSP-ядра Elcore-30M ориентирована на высокопроизводительную параллельную обработку данных. В рамках одной инструкции может выполняться несколько команд (операций). В настоящем документе приняты следующие терминологические определения:

Инструкция – набор команд (операций), выполняющихся одновременно.

Команда (операция) – часть инструкции, определяющая действие того или иного исполнительного устройства DSP-ядра.

Инструкции размещаются в программной памяти DSP-ядра последовательно в порядке нарастания адреса. Каждая инструкция занимает одно (короткие форматы) или два (длинные форматы) 32-разрядных слова. Адрес чтения программной памяти формируется с помощью регистра программного счетчика PC, автоматически инкрементирующегося (на 1 или 2) при последовательном ходе программы.

Команды (операции) DSP-ядра по своему действию делятся на три больших группы:

- вычислительные команды;
- команды пересылок;
- команды программного управления.

5.2 Вычислительные команды

5.2.1 Каждая из команд данного типа производит некоторое действие над данными, хранящимися в регистровом файле (RF) DSP-ядра, и полученные результаты помещаются также в регистры RF. Кроме того, формируется набор признаков результата, который помещается в регистр CCR.

Вычислительные команды, в свою очередь, делятся по характеру исполняемой операции и по форматам обрабатываемых данных на более мелкие группы команд, приведенные ниже в таблицах.

При определении мнемоники команд приняты следующие соглашения:

- команды, работающие в длинном формате, имеют на конце суффикс “L”;
- команды, работающие в комплексных форматах, имеют на конце суффикс “X”;
- команды, работающие в формате с плавающей точкой, имеют префикс “F”.

Имеются некоторые исключения из приведенных выше правил, в частности, для команд, работающих одновременно с различными форматами данных. Детальное описание команд дается в разделе 5.

5.2.2 Команды сложения/вычитания в форматах с фиксированной точкой

В таблице 5.1 представлен перечень команд сложения/вычитания в форматах с фиксированной точкой.

Таблица 5.1. — Команды сложения/вычитания в форматах с фиксированной точкой

Мнемоника	Содержание команды (формат данных)
Сложение	
ADC	Сложение с переносом (short)
ADCL	Сложение с переносом (long)
ADC16L	Сложение смешанное
ADD	Сложение (short)
ADDL	Сложение (long)
ADDLR	Сложение (long) с округлением
ADDLRTR	Сложение (long) с округлением и преобра-зованием формата (в short)
ADDX	Сложение комплексное (X16)
AD1	Сложение и инкремент (short)
Вычитание	
SBC	Вычитание с переносом (short)
SBCL	Вычитание с переносом (long)
SUB	Вычитание (short)
SUBL	Вычитание (long)
SUBLR	Вычитание (long) с округлением
SUBLRTR	Вычитание (long) с округлением и преобразованием формата (в short)
SUBX	Вычитание комплексное (X16)
Сложение-вычитание	
ADDSUB	Сложение-вычитание (short)
ADDSUBL	Сложение-вычитание (long)
ADDSUBX	Сложение-вычитание (X16)
ASH	Сложение и вычитание двух пар чисел (short)
SAH	Сложение и вычитание двух пар чисел (short)
	Инкремент/декремент
INC	Инкремент (short)
INCL	Инкремент (long)
DEC	Декремент (short)
DECL	Декремент (long)

5.2.3 Команды умножения/накопления в форматах с фиксированной точкой

В таблице 5.2 представлен перечень команд умножения/накопления в форматах с фиксированной точкой.

Таблица 5.2. — Команды умножения/накопления в форматах с фиксированной точкой

Мнемоника	Содержание команд (форматы данных)
Умножение	
MPF	Умножение дробное со знаком (short)
MPF2	Парное умножение дробное со знаком (short)

MPF2S	Парное умножение дробное со знаком (short), с перестановкой сомножителей
MPSS	Умножение целое со знаком (short)
MPUU	Умножение целое без знака (short)
MPX	Умножение дробное комплексное (X8), второй операнд - комплексно-сопряженный
MPYL	Умножение целое со знаком (long)
Умножение с накоплением (MAC)	
MAC	Умножение целое со знаком (short) и накопление (в формате __Int64)
MACL	Умножение целое со знаком (long) и накопление (в формате __Int64)
MMACX	Умножение дробное комплексно-сопряженное (X8) и целочисленное (X16)
MAC2	Парное умножение (short) и накопление 2-х результатов (в формате long)
SAC2	Парное накопление (в формате long) со знаком

5.2.4 Команды сдвига в форматах с фиксированной точкой

В таблице 5.3 представлен перечень команд сдвига в форматах с фиксированной точкой.

Таблица 5.3. Команды сдвига

Мнемоника	Содержание команды (формат данных)
Арифметический сдвиг	
ASL	Арифметический сдвиг влево (short)
ASLL	Арифметический сдвиг влево (long)
ASLX	Арифметический сдвиг влево (X16)
ASR	Арифметический сдвиг вправо (short)
ASRL	Арифметический сдвиг вправо (long)
ASRX	Сдвиг арифметический вправо (X16)
Логический сдвиг	
LSL	Логический сдвиг влево (short)
LSLL	Логический сдвиг влево (long)
LSLX	Логический сдвиг влево (X16)
LSR	Логический сдвиг вправо (short)
LSRL	Логический сдвиг вправо (long)
LSRX	Логический сдвиг вправо (X16)
Циклический сдвиг на один разряд	
ROL	Циклический сдвиг на один разряд влево (short)
ROLL	Циклический сдвиг на один разряд влево (long)
ROR	Циклический сдвиг на один разряд вправо (short)
RORL	Циклический сдвиг на один разряд вправо (long)

5.2.5 Другие арифметические команды в форматах с фиксированной точкой

В таблице 5.4 представлен перечень команд.

Таблица 5.4. Другие арифметические команды в форматах с фиксированной точкой

Мнемоника	Содержание команд (форматы данных)
Абсолютное значение	
ABS	Абсолютное значение (short)
ABSL	Абсолютное значение (long)
Обнуление регистра	
CLR	Обнуление (очистка) регистра (short)
CLRL	Обнуление (очистка) регистра (long)

Изменение знака	
NEG	Изменение знака (short)
NEGL	Изменение знака (long)
Транзит	
TR	Транзит (short)
TRL	Транзит (long)
Сравнение	
CMP	Сравнение (short)
CMPL	Сравнение (long)
CMPM	Сравнение модулей (short)
CMPML	Сравнение модулей (long)
CS2	Парная операция выбора большего из двух чисел (short) с фиксацией бита выбора
Максимум/минимум	
MAX	Выбор большего числа (short)
MAXL	Выбор большего числа (long)
MAXM	Выбор числа с большим модулем (short)
MAXML	Выбор числа с большим модулем (long)
MIN	Выбор меньшего числа (short)
MINL	Выбор меньшего числа (long)
MINM	Выбор числа с меньшим модулем (short)
MINML	Выбор числа с меньшим модулем (long)
Определение признаков операнда	
TST	Определение признаков операнда (short)
TSTL	Определение признаков операнда (long)
TSTX	Определение признаков операнда (X16)

5.2.6 Округление, преобразования форматов, упаковка/распаковка

В таблице 5.5 представлен перечень команд округления, преобразования форматов, упаковки/распаковки.

Таблица 5.5 — Команды округления, преобразования форматов, упаковки/распаковки

Мнемоника	Содержание команды (формат данных)
Округление	
RNDL	Округление
Преобразование формата	
FTR	Преобразование формата
FTRFL	Преобразование формата
FTRL	Преобразование формата
Упаковка/распаковка	
PACK	Упаковка (short)
PACKL	Упаковка (long)
DISPFX	Распаковка (дробная) X8 в X16
DISPX	Распаковка (целочисленная) X8 в X16

5.2.7 Логические команды, операции с битами и битовыми полями

В таблице 5.6 представлен перечень команд.

Таблица 5.6 — Логические команды, операции с битами и битовыми полями

Мнемоника	Содержание команды (формат данных)
Логические команды	
AND	Логическое И (short)
ANDC	Логическое И с инверсией (short)
ANDCL	Логическое И с инверсией (long)
ANDI	Инверсия логического И (short)
ANDL	Логическое И (long)
EOR	Логическое исключающее ИЛИ (short)
EORL	Логическое исключающее ИЛИ (long)
NOT	Логическое отрицание (short)
NOTL	Логическое отрицание (long)
OR	Логическое ИЛИ (short)
ORC	Логическое ИЛИ с инверсией (short)
ORCL	Логическое ИЛИ с инверсией (long)
ORI	Инверсия логического ИЛИ (short)
ORL	Логическое ИЛИ (long)
Определение параметра денормализации	
PDN	Определение параметра денормализации (short)
PDNL	Определение параметра денормализации (long)
PDNX	Определение параметра денормализации (X16)
Операции с битами и битовыми полями	
BTST	Проверка разряда (short)
BTSTL	Проверка разряда (long)
MSKG	Формирование маски (short)
MSKGL	Формирование маски (long)
INSL	Побитное мультиплексирование (long)
SWL	Перестановка (long)
Сложение бит	
SMB	Сложение бит (short)
SMBL	Сложение бит (long)

5.2.8 Команды обработки данных в формате с плавающей точкой IEEE-754

В таблице 5.7 представлен перечень команд в формате IEEE.

Таблица 5.7 — Команды обработки данных в формате с плавающей точкой IEEE-754

Мнемоника	Содержание команды (формат данных)
FADD	Сложение (24E8)
FSUB	Вычитание (24E8)
FAS	Сложение-вычитание (24E8)
FINT	Округление к ближайшему целому (24E8)
FLOOR	Округление к ближайшему целому (24E8)
FMPY	Умножение (24E8)
FTST	Определение признаков операнда (24E8)

Мнемоника	Содержание команды (формат данных)
FIN	1-я итерация обратной величины
FINR	1-я итерация обратной величины квадратного корня
CVFI	Преобразование формата: формат 24E8 в 32-разрядное целое в дополнительном коде
CVIF	Преобразование формата: 32-разрядное целое в дополнительном коде со знаком в формат 24E8

5.2.9 Команды обработки данных в формате с плавающей точкой 32E16

В таблице 5.8 представлен перечень команд для обработки данных в формате с расширенной плавающей точкой 32E16.

Таблица 5.8 ─ Команды обработки данных в формате с плавающей точкой 32E16

Мнемоника	Содержание команды (формат данных)
CMPE	Сравнение экспонент
ASRLE	Условный арифметический сдвиг вправо (long)
PDNE	Измерение параметра денормализации 16-разрядной мантиссы
PDNLE	Измерение параметра денормализации 32-разрядной мантиссы
CVEF	Преобразование формата: формат 32E16 в 24E8
CVFE	Преобразование формата: формат 24E8 в 32E16

5.3 Команды пересылок

5.3.1 При помощи команд пересылок производится обмен данными между регистрами, регистрами и памятью данных, либо загрузка непосредственных данных в регистры.

Для всех видов команд пересылок используется одна и та же мнемоническая запись – MOVE, однако форматы и коды инструкций зависят от типа пересылки и ее параметров. Подробно форматы и коды инструкций рассматриваются в следующих разделах.

5.4 Команды программного управления

5.4.1 Команды программного управления производят изменения в последовательности исполнения инструкций DSP-ядра. С их помощью организуются программные переходы, циклы, вход в подпрограмму и выход из нее, останов DSP-ядра (см. таблицу 5.9).

Команды программных переходов B, BD, BS, J, JD, JS являются условными, остальные команды - безусловные.

Таблица 5.9

Мнемоника	Содержание команды (формат данных)
DO	Оператор цикла
DOFOR	Оператор бесконечного цикла

ENDDO	Окончание цикла
B	Ветвление программы
BD	Ветвление программы (отложенное)
BS	Вызов подпрограммы
J	Программный переход
JD	Программный переход (отложенный)
JS	Вызов подпрограммы
RTS	Возврат из подпрограммы
RTI	Возврат из прерывания
NOP	Пустая операция
STOP	Останов

5.5 Ограничения при исполнении инструкций

5.5.1 Ограничение на адреса результатов одновременно исполняемых операций.

Одновременно исполняемые вычислительные операции и пересылки не должны иметь одинаковые адреса операндов-приемников (регистров данных). Ассемблер DSP-ядра дает в этом случае предупреждение.

5.5.2 Ограничения при исполнении инструкций программного управления

Заданное количество повторений цикла DO (регистр LC) должно находиться в пределах от одного (от двух - для циклов, состоящих из одной инструкции) до 16383.

Количество вложенных циклов DO, DOFOR не должно превышать семи (ограничение связано с глубиной стека циклов). Допускаются вложения только одноименных циклов – циклы DO вкладываются в циклы DO, циклы DOFOR - в циклы DOFOR.

Количество вложенных друг в друга циклов (DO, DOFOR) и подпрограмм (BScc, JScc) не должно превышать пятнадцать (ограничение связано с глубиной системного стека).

Цикл DO, DOFOR не может оканчиваться на команду программного управления - DO, DOFOR, B, J, BD, JD, BS, JS, RTS, ENDDO.

Цикл DO, DOFOR не может оканчиваться на ту же инструкцию, что и вложенный в него цикл. Если вложенный цикл состоит из одной инструкции, между его окончанием и последней инструкцией внешнего цикла должна быть еще хотя бы одна инструкция.

Адрес последней команды исполняемого цикла DO, DOFOR не может использоваться как адрес перехода для команд B, J, BD, JD, BS, JS. (Пояснение: переход на метку конца цикла возможен в тех случаях, когда данный цикл не запущен).

Непосредственно после команды отложенного перехода BD, JD не может следовать команда программного управления - DO, DOFOR, B, J, BD, JD, BS, JS, RTS, ENDDO.

Запись в регистры LA, LC, SP, запись/чтение из стеков SS, CSH, CSL во время исполнения цикла DO, DOFOR может привести к неправильной работе цикла; запись в регистр SP, запись/чтение из стека SS во время исполнения подпрограммы может привести к неправильной работе подпрограммы

5.5.3 Ограничения при исполнении пересылок

К запрещенным комбинациям команд относятся следующие:

- регистры CCR, PDNR недоступны для пересылок непосредственных данных (форматы 3, 7);
- некоторые вычислительные команды не могут сочетаться с пересылкой, источником либо получателем в которой является какой-либо регистр RF; указанные ограничения приведены в описаниях соответствующих команд.

6. РАСШИРЕНИЕ СИСТЕМЫ ИНСТРУКЦИЙ

Расширение системы инструкций, в отличие от базовой системы инструкций, позволяет оперировать с 64/128 разрядными данными. Краткое описание команд, составляющие расширение системы инструкций, приводится в настоящем разделе. Команды сгруппированы по функциональному признаку.

6.1 Принятые обозначения

6.1.1 Обозначения, принятые при описании расширения системы инструкций, приведены в таблице 6.1.

Таблица 6.1. ─ Форматы данных в регистровом файле

Обозначение регистра	Разрядность регистра	Номера	Варианты форматов данных
T /S / D	16	31:0	R = r[15:0]
T.L / S.L / D.L	32	31:0	L = (l_1 , l_0), l_1 = L[31:16], l_0 = L[15:0] L = (complex) LX = l_1 + j l_0
T.D / S.D / D.D	64	31:0	D = (d_3 , d_2 , d_1 , d_0), D_3 = D[63:48], d_2 = D[47:32], d_1 = D[31:16], d_0 = D[15:0] D = (LX_1 , LX_0), LX_1 = d_3 + j d_2, LX_0 = d_1 + j d_0 D = (complex) DX = L_1 + j L_0, L_1 = (d_3 , d_2), L_0 = (d_1 , d_0)
T.Q / S.Q / D.Q	128	30:0 16 четных	Q = (q_7, q_6, q_5, q_4, q_3, q_2, q_1, q_0), q_7 = D[127:112], q_6 = D[111:96], q_5 = D[95:80], q_4 = D[79:64], q_3 = D[63:48], q_2 = D[47:32], q_1 = D[31:16], q_0 = D[15:0] Q = (DX_1, DX_0) = (LX_3, LX_2, LX_1, LX_0), LX_3 = q_7 + j q_6, LX_2 = q_5 + j q_4, LX_1 = q_3 + j q_2, LX_0 = q_1 + j q_0 Q = (L_3 , L_2 , L_1 , L_0), L_3 = (d_7 , d_6), L_2 = (d_5 , d_4), L_1 = (d_3 , d_2), L_0 = (d_1 , d_0)

6.2 Умножитель, плавающая точка

Мнемоника	Содержание команды (формат данных)
FM2	Два умножения, (float)
FMS2	Два умножения с перестановкой полуслов в S, (float)
FM4	Четыре умножения, (float)
FM4C	Четыре умножения на общую константу, (float)
FIN4	Четыре нулевых приближения к обратной величине
FINR4	Четыре нулевых приближения к обратной величине от квадратного корня

6.3 АЛУ, плавающая точка

Мнемоника	Содержание команды (формат данных)
FASX	Сложение и вычитание комплексных операндов, (float)
FASXS	Сложение и вычитание комплексных операндов с перестановкой в T, (float)
FAX	Сложение комплексных операндов, (float)
FSA	Вычитание и сложение, (float)
FSX	Вычитание комплексных операндов, (float)
FA4	Четыре сложения, (float)
FS4	Четыре вычитания, (float)

6.4 Умножитель, фиксированная точка

6.4.1 Дополнительный режим работы: сатурация

Сатурация: присваивание результату предельного (максимального или минимального) значения при переполнении. Режим инициируется посредством записи признака в служебный регистр.

Мнемоника	Содержание команды (формат данных)
M2	Два умножения, целые, 16 •16 → 32
M4	Четыре умножения, целые, 16 •16 → 32
MF4	Четыре умножения, дробные, 16 •16 → 32 округление → 16, сатурация
MF8	Восемь умножений, дробные, 16 •16 → 32 округление → 16, сатурация
MFA21	Сумма двух произведений, дробные умножения, (16 • 16 + 32) округление → 16
MFA22	Две суммы двух произведений, дробные умножения, (16 • 16 + 32) округление → 16
MFA24	Четыре суммы двух произведений, дробные умножения, (16 • 16 + 32) округление → 16
MFA41	Сумма четырех произведений, дробные умножения, (16 • 16 + 32) округление → 16
MFA42	Две суммы четырех произведений, дробные умножения, (16 • 16 + 32) округление → 16
MFA81	Сумма восьми произведений, дробные умножения, (16 • 16 + 32) округление → 16
MFX	Умножение комплексное, дробное, (16+j16) • (16+j16) → (32+j32) округление → (16+j16)
MFX2	Два умножения комплексные, дробные, (16+j16) • (16+j16) → (32+j32) округление → (16+j16)
MFXC	Умножение комплексное с сопряжением SX, дробное, (16+j16) • (16+j16) → (32+j32) округление → (16+j16)
MFXC2	Два умножения комплексные с сопряжением SX, дробные, (16+j16) • (16+j16) → (32+j32) округление → (16+j16)
ML2	Два умножения, целые, 32 • 32 → 64
UML	Умножение, целое, без знака, 32 • 32 → 64
UML2	Два умножения, целые, S без знака, 32 • 32 → 64

6.5 Умножитель/ аккумулятор, управление аккумуляторами

Мнемоника	Содержание команды (формат данных)
CLRAC	Групповая очистка 32-разрядных аккумуляторов ACn по маске в 16-разрядном регистре T («1» в n-м бите вызывает сброс) Номер аккумулятора (n) задается установкой в «1» n-го разряда 16-разрядного регистра T.
LDAC	Выгрузка содержимого 32-разрядного аккумулятора ACn в 32-разрядный регистр регистрового файла D.L.
LDACD	Выгрузка содержимого 64-разрядного аккумулятора ACn.D в 64-разрядный регистр регистрового файла D.D.Номер аккумулятора (n) задается установкой в «1» n-го разряда 16-разрядного регистра T.
STAC	Загрузка 32-разрядного слова из 32-разрядного регистра регистрового файла D.L в 32-разрядный аккумулятор ACn. Номер аккумулятора (n) задается установкой в «1» n-го разряда 16-разрядного регистра T.
STACD	Загрузка 64-разрядного слова из 64-разрядного регистра регистрового файла S.D в 64-разрядный аккумулятор ACn.D Номер аккумулятора (n) задается установкой в «1» n-го разряда 16-разрядного регистра T.

6.6 Умножитель/ аккумулятор, фиксированная точка

Мнемоника	Содержание команды (формат данных)
MAC11 (MAC)	Умножение с аккумуляцией, целое, 16 • 16 + 64
MAC12	Два умножения с аккумуляцией, целые, 16 • 16 + 64
MAC14	Четыре умножения с аккумуляцией, целые, 16 • 16 + 64
MAC18	Восемь умножений с аккумуляцией, целые, 16 • 16 + 64
MAC21	Сумма двух произведений с аккумуляцией, целые, 16 • 16 + 64
MAC22	Две суммы двух произведений с аккумуляцией, целые, 16 • 16 + 64
MAC24	Четыре суммы двух произведений с аккумуляцией, целые, 16 • 16 + 64
MAC41	Сумма четырех произведений с аккумуляцией, целые, 16 • 16 + 64
MAC42	Две суммы четырех произведений с аккумуляцией, целые, 16 • 16 + 64
MAC81	Сумма восьми произведений с аккумуляцией, целые, 16 • 16 + 64
MACX	Умножение с аккумуляцией, комплексное, целое, (16+j16) • (16+j16) + (64+j64)
MACX2	Два умножения – аккумуляция комплексных операндов, целые, (16+j16) • (16+j16) → (32+j32)+ (64+j64)
MACXC2	Два умножения – аккумуляция комплексных операндов, сопряжение SX, целые, (16+j16) • (16+j16) → (32+j32)+ (64+j64)
MACXC	Умножение с аккумуляцией, комплексное, целое, сопряжение SX, (16+j16) • (16+j16) + (64+j64)

6.7 Умножитель/ аккумулятор, дополнительные инструкции

Мнемоника	Содержание команды (формат данных)
ACSG1	Накопление со знаком, целое, 16 + 64
ACSG2	Два накопления со знаками, целые, 16 + 64
ACSG4	Четыре накопления со знаками, целые, 16 + 64
ACSG8	Восемь накоплений со знаками, целые, 16 + 64
MACL2	Два умножения с суммированием и аккумуляцией, целые, 32 • 32 → 64 + 64
MACXB4	Четыре комплексных умножения с аккумуляцией, сопряжение SBX, целые, (8+j8) • (8-j8) → (16+j16) + (64+j64) Комплексные 8-разрядные операнды: TBX $_i$ = T [31:24 + 32•i] + j T [15:8 + 32•i], i = 3,2,1,0 (SBX – аналогично)
UACB44	Четыре накопления 4-х смежных байт в четырех 32-разрядных аккумуляторах, целые, беззнаковые, 8 + 32 → 32
ACB44	Четыре накопления 4-х смежных байт в четырех 32-разрядных аккумуляторах, целые, знаковые, 8 + 32 → 32

6.8 Сдвигатель, 64-разрядные операнды, фиксированная точка

Мнемоника	Содержание команды (формат данных)
ASRD	Арифметический сдвиг 64-разрядного операнда S вправо на T разрядов: D = S>>T Параметр сдвига задается 16-разрядным числом в регистре T
ASRDi	Арифметический сдвиг 64-разрядного операнда S вправо Параметр сдвига задается непосредственно 5-разрядным числом
ASLD	Арифметический сдвиг 64-разрядного операнда S влево на T разрядов: D = S<<T. Параметр сдвига задается 16-разрядным числом в регистре T
ASLDi	Арифметический сдвиг 64-разрядного операнда S влево. Параметр сдвига задается либо непосредственно 5-разрядным числом
LSRD	Логический сдвиг 64-разрядного операнда S вправо на T разрядов: D = S>>T Параметр сдвига задается 16-разрядным числом в регистре T
LSRDi	Логический сдвиг 64-разрядного операнда S вправо Параметр сдвига задается либо непосредственно 5-разрядным числом
LSLD	Логический сдвиг 64-разрядного операнда S влево на T разрядов: D = S<<T Параметр сдвига задается 16-разрядным числом в регистре T
LSLDi	Логический сдвиг 64-разрядного операнда S влево Параметр сдвига задается либо непосредственно 5-разрядным числом
RORD	Круговой сдвиг 64-разрядного операнда S вправо на 1 разряд
ROLD	Круговой сдвиг 64-разрядного операнда S влево на 1 разряд
ASRXL	Арифметический сдвиг компонентов комплексного операнда S вправо: D1 + jD0 = (S1>>t1) + j(S0>>t0). Параметры сдвига задаются парой 16-разрядных чисел в регистре T = (t1 , t0)
ASLXL	Арифметический сдвиг компонентов комплексного операнда S влево: D1 + jD0 = (S1<<t1) + j(S0<<t0). Параметры сдвига задаются парой 16-разрядных чисел в регистре T = (t1 , t0)
LSRXL	Логический сдвиг компонентов комплексного операнда S вправо: D1 + jD0 = (S1>>t1) + j(S0>>t0). Параметры сдвига задаются парой 16-разрядных чисел в регистре T = (t1 , t0)
LSLXL	Логический сдвиг компонентов комплексного операнда S влево: D1 + jD0 = (S1<<t1) + j(S0<<t0). Параметры сдвига задаются парой 16-разрядных чисел в регистре T = (t1 , t0)

SMBD	Подсчет количества единичных разрядов в 64-разрядном операнде S
ASRDE	Приведение двух 64-разрядных мантисс S и D к общей экспоненте посредством арифметического сдвига вправо одной из мантисс. При E=0 сдвигается мантисса S, при E=1 – мантисса D. Параметр сдвига задается в 16-разрядном регистре T. При реализации E-формата используются бит E регистра CCR – указатель денормализуемой мантиссы

6.9 АЛУ, 16/32-разрядные операнды, фиксированная точка

6.9.1 Дополнительные режимы работы: масштабирование, сатурация.

Масштабирование: деление результата на 1/2/4/8 (8 - для **A81**) посредством сдвига вправо на 0/1/2/3 (3 - для **A81**) разрядов. Параметр сдвига содержится в служебном регистре. Режим инициируется модификатором команды.

Сатурация: присваивание результату предельного (максимального или минимального) значения при переполнении. Режим инициируется посредством записи признака в служебный регистр.

Мнемоника	Содержание команды (формат данных)
A8	Восемь сложений, целые, 16 – разрядные, масштабирование, сатурация
S8	Восемь вычитаний, целые, 16 – разрядные, масштабирование, сатурация
MS8	Модули от восьми вычитаний, целые, 16 – разрядные, масштабирование, сатурация
A4	Четыре сложения, целые, 16 – разрядные, масштабирование, сатурация
S4	Четыре вычитания, целые, 16 – разрядные, масштабирование, сатурация
MS4	Модули от четырех вычитаний, целые, 16 – разрядные, масштабирование, сатурация
MS2	Модули от двух вычитаний, целые, 16 – разрядные, масштабирование, сатурация
ALL4	Четыре сложения 32-разрядные, целые, масштабирование, сатурация
ALL2	Два сложения 32-разрядные, целые, масштабирование, сатурация
AL4	Четыре сложения 16-разрядных t_i и 32-разрядных S_i, целые, 16+32 → 32, масштабирование, сатурация
AL2	Два сложения 16-разрядных t_i и 32-разрядных S_i, целые, 16+32 → 32, масштабирование, сатурация
A81	Сложение по 8, целые, 16-разрядное, масштабирование, масштабирование, сатурация
A42	Два сложения по 4, целые, 16-разрядные, масштабирование, масштабирование, сатурация
A24	Четыре сложения по 2, целые, 16-разрядные, масштабирование, сатурация
ALL41	Сложение по 4, целое 32-разрядное, масштабирование, сатурация
ALLFT41	Сложение по 4, целое 32-разрядное, преобразование формата дробное с округлением 32 → 16, сатурация
ALLFT22	Два сложения по 2, целые 32-разрядные, преобразование формата дробное с округлением 32 → 16, сатурация
ASX2	Два сложения и вычитания комплексные, целые, 16 – разрядные, масштабирование, сатурация
ASXS2	Два сложения и вычитания комплексные, перестановка в TX, целые, 16 –

	разрядные, масштабирование, сатурация
ASXS	Сложение и вычитание комплексные, перестановка в ТХ, целые, 16 – разрядные, масштабирование, сатурация
RA8	Восемь скользящих сумм, целые, 16 – разрядные
RA4	Четыре скользящие суммы, целые, 16– разрядные
SGA8	Восемь знаковых сумм, целые, 16 – разрядные, масштабирование, сатурация
SGA4	Четыре знаковых суммы, целые, 16 – разрядные, масштабирование, сатурация
A8s	Восемь сложений, целые, 16 – разрядные, обязательное масштабирование, сатурация
S8s.	Восемь вычитаний, целые, 16 – разрядные, обязательное масштабирование, сатурация
SLL2	Два вычитания 32-разрядные, целые, масштабирование, сатурация
SLL4	Четыре вычитания 32-разрядные, целые, масштабирование, сатурация
BF4 ,	Базовая операция FFT-4, формат целый (16+j16), масштабирование, сатурация
BIF4	Базовая операция IFFT-4, формат целый (16+j16), масштабирование, сатурация
AXJ4	Четыре комплексных сложения с предварительным умножением операндов ТХ на j , целые, 16 – разрядные, масштабирование, сатурация
SXJ4	Четыре комплексных сложения с предварительным умножением операндов ТХ на j , целые, 16 – разрядные, масштабирование, сатурация
CMPN4	Четыре сравнения $s_3 – t_3$, $s_2 – t_2$, $s_1 – t_1$, $s_0 – t_0$, целые 16-разрядные,выработка признаков отрицательных результатов N3:0, упаковка признаков D = (D>>4) | (N3:0)<<60
CMPZ4	Четыре сравнения $s_3 – t_3$, $s_2 – t_2$, $s_1 – t_1$, $s_0 – t_0$, целые 16-разрядные, выработка признаков равенства результатов Z3:0, упаковка признаков D = (D>>4) | (Z3:0)<<60
CMPN8	Восемь сравнений: $s_i – t_i$, i = 7:0, целые 16-разрядные, выработка признаков отрицательных результатов N7:0; упаковка признаков D = (D>>8) | (N7:0)<<120
CMPZ8	Восемь сравнений: $s_i – t_i$, i = 7:0, целые 16-разрядные, выработка признаков равенства результатов Z7:0; упаковка признаков D = (D>>8) | (N7:0)<<120
CMPNL2	Два сравнения $S_1 – T_1$, $S_0 – T_0$, целые 32-разрядные, выработка признаков отрицательных результатов N1:0, упаковка признаков D = (D>>2) | (N3:0)<<62
CMPZL2	Два сравнения $S_1 – T_1$, $S_0 – T_0$, целые 32-разрядные, выработка признаков равенства результатов Z1:0, упаковка признаков D = (D>>2) | (N3:0)<<62
CMPNL4	Четыре сравнения: $S_i – T_i$, i = 3:0, целые 32-разрядные, выработка признаков отрицательных результатов N3:0; упаковка признаков D = (D>>4) | (N7:0)<<124
CMPZL4	Четыре сравнения: $S_i – T_i$, i = 3:0, целые 32-разрядные, выработка признаков равенства результатов Z3:0; упаковка признаков D = (D>>4) | (N7:0)<<124
CMPNB16	Шестнадцать сравнений sb [i] – tb [i], целые 8-разрядные, выработка признаков отрицательного результата N15:0, упаковка признаков: D = (D>>16) | (N15:0)<<112
CMPZB16	Шестнадцать сравнений sb [i] – t b[i], целые 16-разрядные, выработка признаков равенства Z15:0, упаковка признаков: D = (D>>16) | (Z15:0)<<112
MAX4	Поиск максимума и его номера, целые 16-разрядные значения и номер

MAXL2	Поиск максимума и его номера, целые 32-разрядные значения и 16-разрядный номер
MAX8	Поиск максимума и его номера, целые 16-разрядные значения и 16-разрядный номер.
MAXL4	Поиск максимума и его номера, целые 32-разрядные значения и 16-разрядный номер
MIN4	Поиск минимума и его номера, целые 16-разрядные значения и номер
MINL2	Поиск минимума и его номера, целые 32-разрядные значения и 16-разрядный номер
MIN8	Поиск минимума и его номера, целые 16-разрядные значения и 16-разрядный номер
MINL4	Поиск минимума и его номера, целые 32-разрядные значения и 16-разрядный номер

6.10 АЛУ, 64-разрядные операнды, фиксированная точка

Мнемоника	Содержание команды (формат данных)
ANDD	Поразрядные логические «И» над 64-разрядными операндами: Dn = (Tn)&(Sn), n=0:63
ANDCD	Поразрядные логические «И» над 64-разрядными операндами с инверсией T: Dn = (~Tn)&(Sn), n=0:63
ANDID	Поразрядные логические «И» над 64-разрядными операндами с инверсией результатов: Dn = ~((Tn)&(Sn)), n=0:63
ORD	Поразрядные логические «ИЛИ» над 64-разрядными операндами: Dn = (Tn)|(Sn), n=0:63
ORCD	Поразрядные логические «ИЛИ» над 64-разрядными операндами с инверсией T: Dn = (~Tn)|(Sn), n=0:63
ORID	Поразрядные логические «ИЛИ» над 64-разрядными операндами с инверсией результатов: Dn = ~((Tn)|(Sn)), n=0:63
EORD	Поразрядные логические «ИСКЛЮЧАЮЩИЕ ИЛИ» над 64-разрядными операндами: Dn = (Tn)^(Sn), n=0:63
NOTD	Поразрядные логические «НЕ» над 64-разрядным операндом: Dn =~(Sn), n=0:63
INSD	Поразрядное объединение двух 64-разрядных операндов T и D по маске S: Dn =((Dn)&(~(Sn))) | ((Dn)&(Sn)), n=0:63
BTSTD	Запись n-го разряда 64-разрядного операнда S в разряд признака C: C=Sn Номер разряда n задается 16-разрядным числом в регистре T
BTSTDi	Запись n-го разряда 64-разрядного операнда S в разряд признака C: C=Sn Номер разряда n задается непосредственно 5-разрядным числом
CLRD	Сброс в ноль разрядов 64-разрядного регистра
ADDD	Целое сложение двух 64-разрядных операндов: D = S + T
SUBD	Целое вычитание двух 64-разрядных операндов: D = S - T
ADCD	Целое сложение двух 64-разрядных операндов с добавлением признака переноса: D = S + T + C
SBCD	Целое вычитание двух 64-разрядных операндов с вычитанием признака переноса: D = S − T − (~C)
ADDLD	Целое знаковое сложение 32-разрядного операнда T и 64-разрядного опе-

	ранда S с образованием 64-разрядного результата: D = S + T
TRD	Пересылка 64-разрядного операнда: D = S
PDND	Определение параметра денормализации 64-разрядного операнда S
PDNXL	Определение параметра денормализации комплексного операнда с 32-разрядными компонентами (S1 + jS0)
NORVD	Нормализация 64-разрядного операнда после возможного переполнения Если V = 0, то D = S Если V = 1, то: 1) операнд нормализуется сдвигом вправо на 1 бит с учетом произошедшего переполнения: D[63:0] = (~S[63]), (S[63:1]); 2) выполняется стандартное округление выдинутого бита S[0] (к четному результату)
PDNDE	Определение параметра денормализации 64-разрядной мантиссы Е-формата Если S ≠ 0, то параметр денормализации записывается в старшее полуслово результата d1; Если S = 0, то d1=0 и d0=0
NEGDE	Изменение знака мантиссы S При возникновении переполнения: записывается результат D=0,5 (дробный формат) и устанавливается признак переполнения V = 1

6.11 Специализированный блок пересылок

Мнемоника	Содержание команды (формат данных)
TRS0	Пересылка 128-разрядного операнда
TRS1	Сдвиг двух 64-разрядных операндов на одно16-разрядное слово При T=S: круговой сдвиг 64-разрядного операнда на одно 16-разрядное слово
TRS2	Сдвиг двух 64-разрядных операндов на n = 2 16-разрядных слова При T=S: круговой сдвиг 64-разрядного операнда на n = 2 16-разрядных слова
TRS3	Сдвиг двух 64-разрядных операндов на n = 3 16-разрядных слова При T=S: круговой сдвиг 64-разрядного операнда на n = 3 16-разрядных слова
TRS4	Преобразование восьми 8-разрядных операндов в 16—разрядные, дробный формат
TRS5	Преобразование восьми 8-разрядных операндов в 16—разрядные, целый формат, расширение знака
TRS6	Преобразование восьми 8-разрядных операндов в 16—разрядные, целый формат, без знака (старшие 8 бит – нулевые)
TRS7	Преобразование восьми 16-разрядных операндов в 8—разрядные, дробный формат, округление, сатурация
TRS8	Преобразование восьми 16-разрядных операндов в 8—разрядные, целый формат, без знака, сатурация
TRS9	Перестановка четырех 16-разрядных операндов инверсная
TRS10	Перестановка четырех 16-разрядных операндов двоично – инверсная
TRS11	Перестановка четырех 16-разрядных операндов матричная
TRS12	Вложение двух векторов из четырех 16-разрядных операндов каждый

TRS13	Раскладка восьми 16-разрядных операндов на два массива из четырех четных и четырех нечетных элементов
TRS14	Два преобразования трех 16-разрядных операндов в составной 16-разрядный код (5,6,5 разрядов). Входные операнды – целые со знаком, выходные – целые без знака. Используется сатурация сверху (31 для t2,t0,s2,s0 и 63 для t1,s1), отрицательные числа обнуляются.
TRS15	Преобразование четырех 32-разрядных операндов в 16-разрядные, дробный формат, округление, сатурация.
TRS16	Преобразование четырех 32-разрядных операндов в 16-разрядные, целый формат, сатурация.
TRS17	Преобразование четырех 32-разрядных операндов в 16-разрядные, целый формат, без знака, сатурация.
TRS18	Поразрядное вложение двух 16-разрядных операндов (T, S) с образованием 32-разрядного операнда D. Разряды S становятся четными разрядами D, T – нечетными.
TRS19	Преобразование восьми 16-разрядных операндов в 8—разрядные, целый формат, сатурация

6.12 Операции над байтами (сложения-вычитания, преобразования)

Мнемоника	Содержание команды (формат данных)
DRGB	Упаковка/распаковка байтных RGB-компонентов изображения для четырех пикселей (транспонирование матрицы 4x4 байт).
TRSB	Конкатенация двух 128-разрядных операндов: совместный сдвиг 128-разрядных операндов T.Q (старшие 128 разр.) и S.Q (младшие 128 разр.) вправо на nsb Байт (на 8nsb бит) с сохранением 128 младших разр. в D.Q. Параметр сдвига nsb поступает из управляющего регистра PDNR[13:10].
AB16	Шестнадцать сумм, целые, 8-разрядные, беззнаковые, сатурация сверху
SB16	Шестнадцать разностей, целые, 8-разрядные, беззнаковые, сатурация снизу
MSB16	Шестнадцать модулей разностей, целые, 8-разрядные, беззнаковые

6.13 Операции над байтами (умножения)

Мнемоника	Содержание команды (формат данных)
MFB16	Шестнадцать умножений, дробные, разрядность 8 •8 → 16 округление → 8, сатурация
UMFB16	Шестнадцать умножений, дробные, беззнаковые, разрядность 8 •8 → 16 округление → 8, сатурация

7. РАБОТА ПРОГРАММНОГО КОНВЕЙЕРА И ВРЕМЯ ИСПОЛНЕНИЯ КОМАНД

Программный конвейер DSP-ядра Elcore-30M содержит 7 фаз. Конвейеризация приводит к тому, что в один и тот же момент времени происходит обработка нескольких инструкций, находящихся на разных стадиях. При отсутствии торможения конвейера скорость выполнения инструкций в конвейерном режиме составляет одну инструкцию в течение одного командного цикла. Исполнение вычислительных команд выполняется на двух последних фазах (на 6-й и 7-й) конвейера.

7.1 Содержание фаз конвейера

7.1.1 При исполнении различных типов операций фазы конвейера DSP-ядра Elcore-30M имеют следующее содержание:

а) при выполнении вычислительной операции:

1 фаза (A): Формирование адреса памяти программ.
2 фаза (F): Выборка инструкции из программной памяти.
3 фаза (D): Декодирование инструкции.
4 фаза (E): Формирование блокировок конвейера.
5 фаза (E1): Чтение данных из RF.
6 фаза (E2): Исполнение инструкции.
7 фаза (E3): Исполнение инструкции, запись данных в RF.

б) при чтении из памяти данных:

1 фаза (A): Формирование адреса памяти программ.
2 фаза (F): Выборка инструкции из программной памяти.
3 фаза (D): Декодирование инструкции.
4 фаза (E): Формирование адреса памяти данных.
5 фаза (E1): Выдача адреса на память данных.
6 фаза (E2): Чтение из памяти данных в буферный регистр.
7 фаза (E3): Запись данных в RF.

в) при записи в память данных:

1 фаза (A): Формирование адреса памяти программ.
2 фаза (F): Выборка инструкции из программной памяти.
3 фаза (D): Декодирование инструкции.
4 фаза (E): Формирование адреса памяти данных.
5 фаза (E1): Выдача адреса на память данных и запись в память данных.

Примечание. При записи/чтении памяти данных арбитром могут вводиться дополнительные такты ожидания.

г) при записи в регистр RF:

1 фаза (A): Формирование адреса памяти программ.
2 фаза (F): Выборка инструкции из программной памяти.

3 фаза (D): Декодирование инструкции.

4 фаза (E): Формирование блокировок конвейера.

5 фаза (E1): Чтение данных из RF или регистра управления.

6 фаза (E2): Запись в RF.

д) при записи в регистр управления:

1 фаза (A): Формирование адреса памяти программ.

2 фаза (F): Выборка инструкции из программной памяти.

3 фаза (D): Декодирование инструкции.

4 фаза (E): Чтение данных из RF.

5 фаза (E1): Запись в регистр управления.

е) при исполнении команд программных переходов B(J), BD(JD) #A:

1 фаза (A): Формирование адреса памяти программ.

2 фаза (F): Выборка инструкции из программной памяти.

3 фаза (D): Декодирование инструкции.

4 фаза (E): Выборка адреса перехода #A

5 фаза (E1): Выдача адреса #A на PRAM

1) Исполнение вычислительных команд

1 фаза (A)	2 фаза (F)	3 фаза (D)	4 фаза (E)	5 фаза (E1)	6 фаза (E2)	7 фаза (E3)
Выдача адреса на PRAM	Чтение инструкции из PRAM	Декодирование инструкции	Формирование блокировок	Выборка данных из RF	Исполнение инструкции (1 фаза)	Исполнение инструкции (2 фаза)

2) Исполнение команд MOVE XRAM, YRAM -> RF

1 фаза (A)	2 фаза (F)	3 фаза (D)	4 фаза (E)	5 фаза (E1)	6 фаза (E2)	7 фаза (E3)
Выдача адреса на PRAM	Чтение инструкции из PRAM	Декодирование инструкции	Модификация адреса XRAM	Выдача адреса на XRAM	Чтение данных из XRAM	Запись данных в RF

3) Исполнение команд MOVE RF -> XRAM

1 фаза (A)	2 фаза (F)	3 фаза (D)	4 фаза (E)	5 фаза (E1)	6 фаза (E2)	7 фаза (E3)
Выдача адреса на PRAM	Чтение инструкции из PRAM	Декодирование инструкции	Модификация адреса XRAM	Запись данных в XRAM	-	-

4) Исполнение команд MOVE RF, RC, #16/32 -> RF

1 фаза (A)	2 фаза (F)	3 фаза (D)	4 фаза (E)	5 фаза (E1)	6 фаза (E2)	7 фаза (E3)
Выдача адреса на PRAM	Чтение инструкции из PRAM	Декодирование инструкции	Формирование блокировок	Выборка данных из RC	Запись данных в RF	-

5) Исполнение команд MOVE RF, #16/32 -> RC(кр.CCR,PDNR,AC)

1 фаза (A)	2 фаза (F)	3 фаза (D)	4 фаза (E)	5 фаза (E1)	6 фаза (E2)	7 фаза (E3)
Выдача адреса на PRAM	Чтение инструкции из PRAM	Декодирование инструкции	Выборка данных из RF	Запись данных в RC	-	-

6 Исполнение команд программных переходов B(J) #A

1 фаза (A)	2 фаза (F)	3 фаза (D)	4 фаза (E)	5 фаза (E1)	6 фаза (E2)	7 фаза (E3)
Выдача адреса а PRAM	Чтение инструкции из PRAM	Декодирование инструкции	Выборка адреса перехода#A	Выдача адреса #A на PRAM	Чтение инструкции по адресу#A	Декодир инструкции по адресу#A.

7.2 Работа программного конвейера при последовательной выборке команд

7.2.1 Работа программного конвейера при последовательной выборке команд из программной памяти иллюстрируется временной диаграммой (рис.7.1), **n** - номер инструкции.

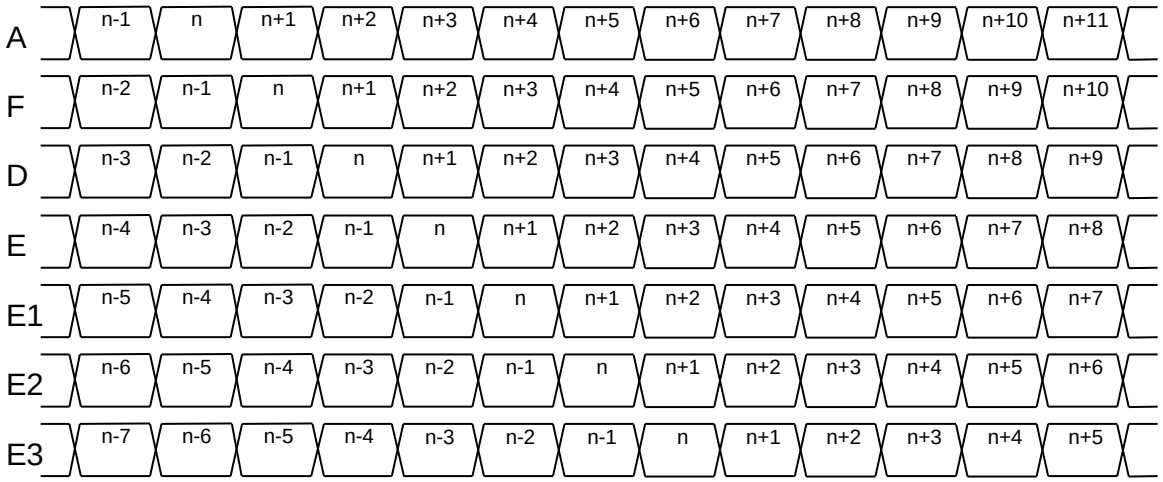


Рис.7.1 ─ Работа программного конвейера при последовательной выборке команд

7.3 Работа программного конвейера при программных переходах

7.3.1 Работа программного конвейера при программных переходах (команды B, BS, J, JS, RTS) иллюстрируется временной диаграммой (рисунок 7.2). Инструкция, следующая за инструкцией программного перехода, не исполняется (заменяется на NOP).

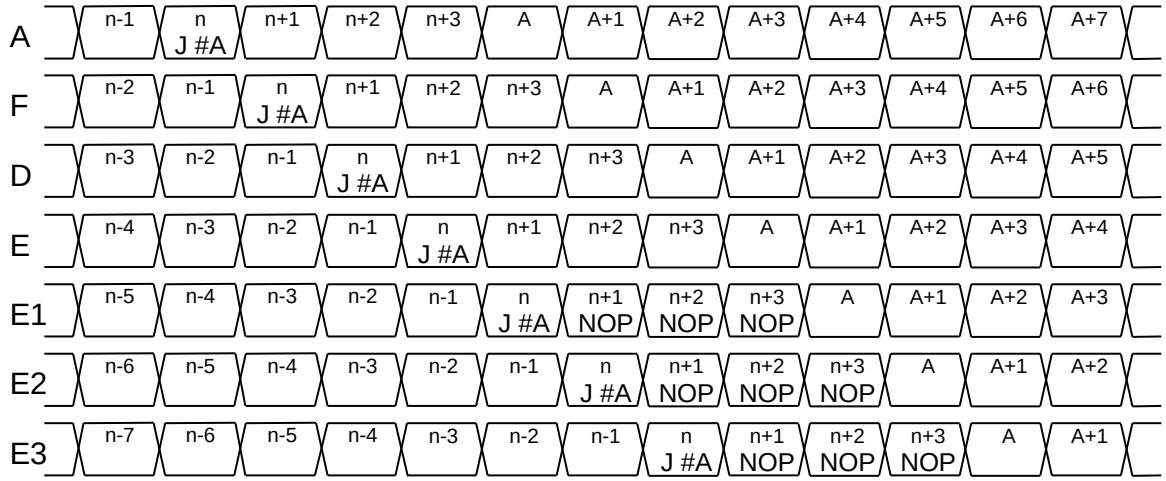


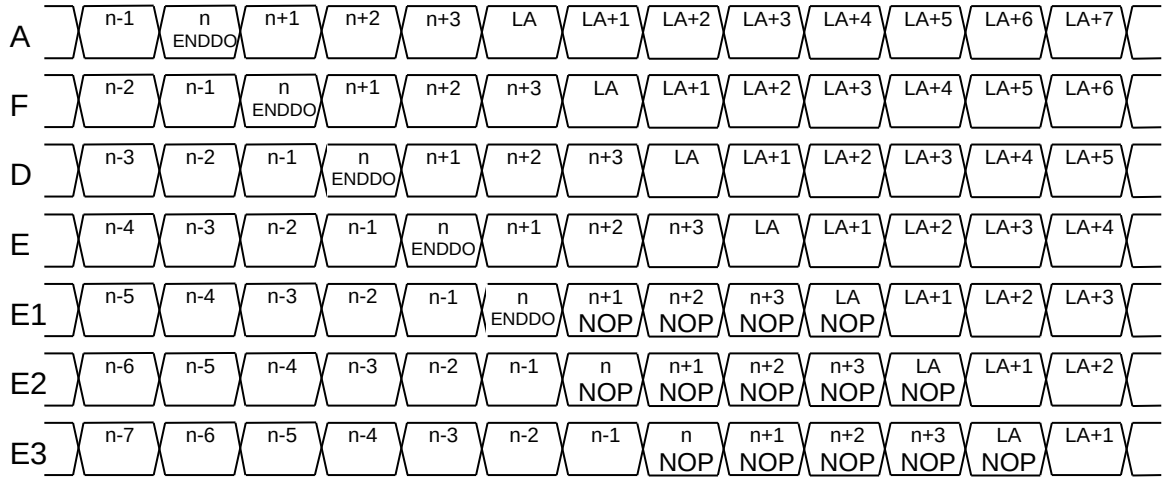
Рис. 7.2 - Работа программного конвейера при программных переходах (команды J (B) #A, JS (BS) #A, RTS)

Примечание. A – адрес перехода

При отложенных программных переходах (команды BD, JD) инструкция, следующая за инструкцией программного перехода, исполняется (рисунок 7.3).

Рис.7.3 ─ Работа программного конвейера при отложенных программных переходах (команды JD (BD) #A). Примечание - A – адрес перехода

Действие команды ENDDO иллюстрируется временной диаграммой (рис.7.4)



Примечание. LA – адрес последней инструкции цикла.

Рис. 7.4 ─ Работа программного конвейера при выполнении команды ENDDO

7.4 Работа программного конвейера при исполнении команды STOP

7.4.1 Работа программного конвейера при исполнении команды STOP иллюстрируется временной диаграммой (рис.7.5).

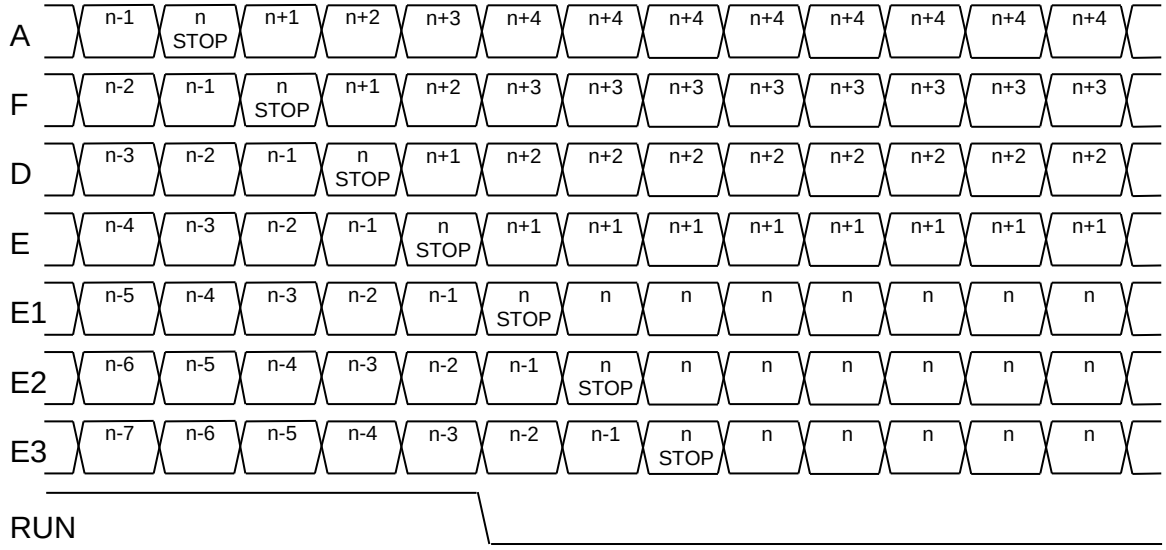


Рис.7.5. Работа программного конвейера при исполнении команды STOP

Команда STOP исполняется на стадии e1. В результате исполнения команды STOP бит “RUN” сбрасывается в “0”, программный конвейер останавливается. Инструкции, находящиеся на стадии декодирования и исполнения, заменяются на NOP. Флаг STP (четвёртый разряд DCSR) по команде STOP устанавливается в “1” и стоит до тех пор, пока не будет сброшен по команде CPU.

7.5 Время исполнения вычислительных операций

7.5.1 В таблице 7.1 указано время исполнения инструкций DSP-ядра Elcore-30M, измеряемое в командных циклах.

Таблица 7.1

Вычислительные операции	Время исполнения, тактов
Все логические операции: AND, ANDL, ANDD, ANDC, ANDCL, ANDCD, ANDI, ANDID, OR, ORL, ORD, ORC, ORCL, ORCD, ORI, ORID, EOR, EORL, EORD, NOT,NOTL,NOTD, INSD, INSL Все операции транзита: TR,TRL,TRD,TRS0-TRS19 Все операции загрузки аккумулятора: CLRAC, LDAC, LDACD, STAC, STACD Другие операции: ROL,ROLL,ROLD,ROR,RORL,RORD,CLR,CLRL,CLRD,TST,TSTL,TSTX,FTST	1
Все остальные вычислительные операции	2

7.6 Торможение конвейера вследствие зависимостей по данным

7.6.1 Помимо программных переходов, другим источником временных потерь, возникающих при конвейеризации вычислительных операций, является наличие в программном коде рекурсивных зависимостей по данным между вычислительными инструкциями. Под данными в общем случае подразумеваются как числовые операнды, так и их признаки, используемые в качестве кодов условий (condition codes) при исполнении условных инструкций.

Рекурсивная зависимость по данным между вычислительными инструкциями возникает в тех случаях, когда текущая исполняемая инструкция использует в качестве входного операнда результат предыдущей инструкции. В таких случаях текущая инструкция называется зависимой, а инструкция, результат которой используется, - разрешающей.

Пример программного кода на ассемблере DSP-ядра с зависимостью по данным между вычислительными инструкциями приводится ниже (индекс в скобках указывает номер инструкции):

```
(n-1)           ADDL R1.L,R2.L,R3.L
(n)             MOVE R3.L,(A0)+
(n+1)           ASL R6,R7,R5
(n+2)           SUB R5,R8,R8
```

В приведенном примере инструкция пересылки (n) использует в качестве входного операнда значение, хранящееся в регистре R3.L, являющееся результатом инструкции (n-1), а инструкция (n+2) - значение R5, являющееся результатом вычислительной инструкции (n+1).

Для получения правильного результата при работе такой программы в конвейер необходимо ввести дополнительные такты торможения (NOP) – два такта перед исполнением инструкция (n), так как операция пересылки исполняется на фазе e1, и один такт перед исполнением инструкция (n+2). Временная диаграмма работы конвейера при исполнении приведенного выше программного кода представлена на рис.7.6.

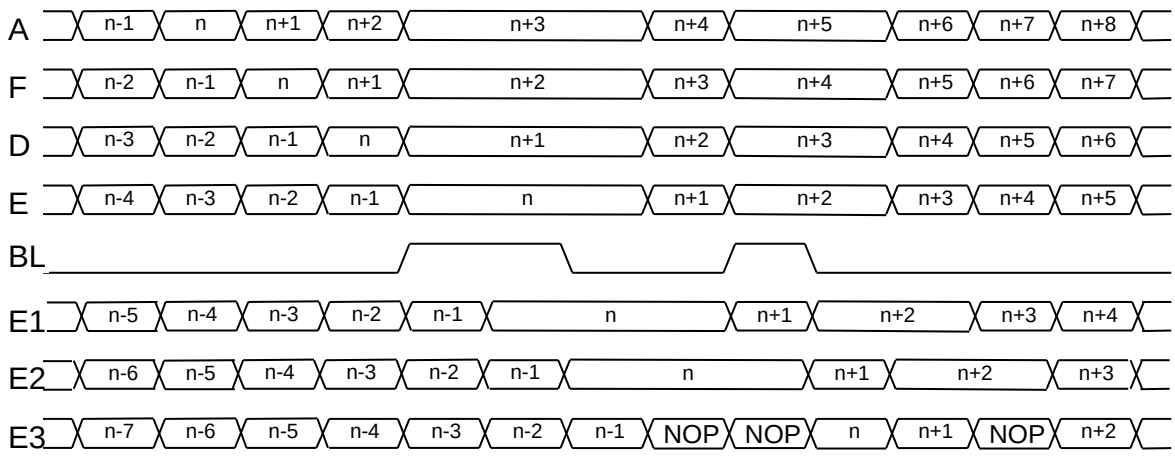


Рис.7.6. Пример торможения программного конвейера Elcore-30M при выполнении фрагмента программы с наличием рекурсивных зависимостей по данным. (BL – сигнал блокировки конвейера)

Таким образом, повышение производительности, достигаемое при увеличении числа фаз программного конвейера, носит в определенной степени условный характер - в тех программах, в которых следующие друг за другом инструкции используют результаты предыдущих, в конвейер приходится вводить такты торможения, что приводит к снижению эффективности.

7.7 Ограничения конвейера

7.7.1 При наличии в исполяемом коде зависимостей по данным, как правило, срабатывает автоматическая аппаратная блокировка программного конвейера на нужное число тактов (автоматическая блокировка программного конвейера управляется битом BD = SR[10] см. п.3.6.2.2). Имеются следующие исключения, когда аппаратная блокировка конвейера отсутствует, и необходимо это учитывать принаписании программного кода:

1) Непосредственно после команды, в результате которой изменяется адресный регистр, нельзя выполнять программный переход по этому регистру.

Пример:

Неправильно:

```
MOVE R0,A7
J A7
```

Правильно:

```
MOVE R0,A7
NOP
J A7
```

2) Непосредственно после команды, в результате которой изменяется YM (11-й разряд регистра SR), нельзя выполнять обращение к памяти по указателю AT с модификацией адреса.

Пример:

Неправильно:

```
MOVE 0x800, SR
MOVE (AT)+DT, R0
```

Правильно:

```
MOVE 0x800, SR
NOP
MOVE (AT)+DT, R0
```

8. ФОРМАТЫ ИНСТРУКЦИЙ

В рамках одной инструкции может быть выполнено до двух вычислительных операций и до двух операций пересылок.Количество одновременно исполняемых операций и их тип определяют *формат инструкции* – т.е. размер и структуру ее кода. Участки кода инструкции, определяющие те или иные параметры входящих в нее операций, называются *полями* инструкции. В настоящем разделе рассматриваются форматы и назначение полей инструкций DSP-ядра Elcore-30M. В таблице 5.1 приведен перечень форматов инструкций. Форматы 1-8 соответствуют базовой системе инструкций, формат 9 – расширению системы инструкций. В форматах 1-7 единственная вычислительная операция OP может иметь тип OP1 или OP2.

Таблица 8.1

Формат	Условие	Операция 1	Операция 2	Пересылка 1	Пересылка 2	Длина кода, 32-разрядных слов
1	[cc]	OP #5/S1,S2,D	-	-	-	1
2	[cc]	OP #16/32,S2,D	-	-	-	2
2d	-	-	DO #16, #16	-	-	2
2t	[cc]	-	-	R/R.L/RC←→R/R.L/RC	-	1
3	-	OP #16,d		-	-	1
3m	[cc]	-	B/J #16	-	-	1
3mb	[cc]	-	B/J Ai	-	-	1
4	-	OP #5/S,D	-	XRAM←→R.L	-	1
4t	-	-	-	XRAM←→BUF	-	1
5	-	OP #5/S,D	-	R/R.L←→R/R.L	-	1
6	-	OP #5/S,D	-	R←→RC	-	1
6t	[cc]	-	-	XRAM←→ R.L	-	1
7	[cc]	OP S,D	-	#16/32→RC/R/R.L	-	2
7t	[cc]	-	-	XRAM(Ai+#16)←→R.L	-	2
8a	-	OP2 #5/S1,S2,D	OP1[s] S1,S2,D	XRAM←→R.L	YRAM→R0	2
8b	-	OP2 #5/S1,S2,D	OP1[s] S1,S2,D	R/R.L←→R/R.L	YRAM→R0	2
8c	[cc]	OP2 #5/S1,S2,D	OP1[s] S1,S2,D	R.L←→R.L	-	2
8d	-	OP2 #5/S1,S2,D	OP1[s] S1,S2,D	R←→RC	-	2
9a	-	OP2e T,S,D	OP1e T,S,D	XRAM←→R.L	YRAM→R0	2
9b	-	OP2e T,S,D	OP1e T,S,D	R/R.L←→R/R.L	YRAM→R0	2
9d	-	OP2e T,S,D	OP1e T,S,D	R←→RC	-	2

Примечание – `S,S1,S2,D` - 16- или 32-разрядный регистр данных (в форматах 1-8)
`s, d, R` - 16-разрядный регистр данных
`R.L` - 32-разрядный регистр данных
`#x` - x-разрядные непосредственные данные
`RC` - управляющий регистр
`cc` - код условия
`Ai` - адресный регистр, i=0,1,…,7
`Ai` - адресный регистр, i=0,1,…,7
`T,S,D` - 16/32/64/128-разрядный регистр данных (в форматах 9)

8.1 Описание форматов инструкций

8.1.1 Форматы инструкций отличаются друг от друга по следующим признакам:

- по возможности использования условия при исполнении инструкции все инструкции DSP Elcore-30M делятся на условно исполняемые (при истинности специфицированного условия) и безусловные. Заметим, что специфицированное условие имеет действие на все операции, входящие в инструкцию;
- по количеству одновременно выполняемых вычислительных операций. Одновременно в рамках одной инструкции может быть выполнено не более двух вычислительных операций. Некоторые вычислительные операции (например, ADDSUB) могут включать в себя несколько арифметических действий. Тем не менее при рассмотрении системы инструкций каждая из них считается одной операцией;
- по типу и количеству операндов, используемых вычислительной операцией. В качестве операндов-источников могут использоваться регистры данных (S, S1, S2) либо непосредственные данные #5, #16 или #32. Операндами-получателями (приемниками) результата являются регистры данных D (для некоторых команд получателями результата являются также регистры CCR, PDNR, регистры-аккумуляторы ACi);
- формат инструкции зависит от числа адресуемых операндов в вычислительных операциях. В форматах 1, 2, 8 это число – не более трех (S1, S2, D). Такие операции называются 3-адресными. В форматах 4 – 7 разрешены двухадресные вычислительные операции, в которых число адресуемых операндов не более двух (S, D). Поле каждого адресуемого операнда состоит из 5 бит, при помощи которых адресуется один из регистров RF. Для операций сдвига в поле операнда S1 (S) может помещаться непосредственное значение сдвига #5;
- по количеству и типу одновременно выполняемых операций пересылок. Максимальное количество одновременно выполняемых операций пересылок – две (в форматах 8a, 8b). Источником в операции пересылки могут быть регистры данных (R - 16-разрядные или R.L - 32-разрядные), регистры управления RC, память XRAM или YRAM, е данные #5, #16 или #32. Получателем могут являться регистры данных, регистры управления RC, память XRAM или YRAM;
- некоторые операции программного управления и пересылок требуют для своей кодировки специальных форматов – 2d, 3m, 3mb, 2t, 6t, 7t.

Формат 1

```
OP[.cc] #5/S1,S2,D
```

Содержит одну вычислительную операцию (трёх, двух, или одноадресную) с условным исполнением, длина кода - одно слово (32 бита).

В рамках данного формата может быть выполнена любая вычислительная операция. Операнды-источники (S1, S2) и приемник (D) – регистры регистрового файла (16 или 32-разрядные). Вместо первого источника может использоваться непосредственный пятиразрядный операнд #5 (параметр сдвига, номер бита).

Формат 2

OP[.cc] #16/#32,S2, D

Содержит одну трёхадресную вычислительную операцию с условным исполнением, длина кода - два слова (64 бита).

Первый операнд-источник – непосредственное 16- или 32-разрядное значение, содержащееся во втором слове инструкции. Второй операнд-источник (S2) и приемник (D) – регистры регистрового файла (16- или 32-разрядные).

Формат 2d

DO #16, #16

Формат 2d применяется для кодирования команды DO с непосредственно заданным числом повторения циклов, содержащимся во втором слове инструкции. Длина кода – 64 разряда.

Формат 2t

MOVE.cc	R, R
	R.L, R.L
	R, RC
	RC, R

Формат 2t является производным от формата 6 и применяется для кодирования условной пересылки между регистром данных и регистром управления.

Длина кода - 32 разряда.

Формат 3

OP #16, d

Содержит одну двухадресную вычислительную операцию с безусловным исполнением. Длина кода - одно слово.

Обрабатываются только 16-разрядные операнды.

1-й операнд (источник) #16 – непосредственное значение.

2-й операнд – источник и приемник (d) – 16–разрядный регистр регистрового файла.

Формат 3m

B	[.cc] #16
BD	
BS	
J	
JD	
JS	

Формат 3m применяется для кодирования команд программных переходов B, BD, BS, J, JD, JS с непосредственно заданным адресом перехода (или с переходом по метке). Длина кода – 32 разряда.

Формат 3mb

B BD BS J JD JS	[.cc] Ai

Формат 3mb применяется для кодирования команд программных переходов B, BD, BS, J, JD, JS с переходом по адресу, содержащемуся в адресном регистре Ai. Длина кода – 32 разряда.

Формат 4

OP #5/S,D [M]	<XRAM>, R.L R.L, <XRAM>

В этом формате двуадресная (или одноадресная) вычислительная операция сочетается с параллельной пересылкой между ячейкой памяти данных <XRAM> и 32-разрядным регистром R.L регистрового файла. Ячейка памяти данных <XRAM> адресуется при помощи одного из адресных регистров в соответствии с режимами адресации, поддерживаемыми адресным генератором AGU.

Длина кода - 32 разряда.

Формат 5

OP #5/S,D [M]	R, R R.L, R.L

В формате 5 двухадресная (или одноадресная) вычислительная операция сочетается с параллельной пересылкой между двумя регистрами данных (оба регистра, источник и приемник, имеют одинаковую разрядность).

Длина кода - 32 разряда.

Формат 6

OP #5/S,D [M]	R, RC RC, R

В формате 6 двухадресная (или одноадресная) вычислительная операция сочетается с параллельной пересылкой между регистром данных и регистром управления.

Длина кода - 32 разряда.

Формат 6t

MOVE.cc	<XRAM>, R.L	
	R.L, <XRAM>	

Формат 6t является производным от формата 4 и применяется для кодирования условных пересылок между ячейкой памяти данных <XRAM> и 32-разрядным регистром регистрового файла.

Длина кода - 32 разряда.

Формат7

OP.cc S,D[M]	#16,R	
	#16,RC	
	#32,R.L	

В формате 7 двухадресная (или одноадресная) вычислительная операция сочетается с параллельной пересылкой непосредственных данных, содержащихся во втором слове инструкции, в регистр данных или управления.

Длина кода - 64 разряда.

Формат 7t

MOVE.cc	<XRAM(Ai+#16)>, R.L	
	R.L, <XRAM(Ai+#16)>	

Формат 7t является производным от формата 6t и применяется для кодирования условных пересылок между ячейкой памяти данных <XRAM>, адресуемой в режиме непосредственного смещения адреса (величина смещения #16 содержится во втором слове инструкции) и 32-разрядным регистром регистрового файла.

Длина кода - 64 разряда.

Формат 8a

OP2 #5/S3,S4,D2	OP1[s] S1,S2,D [M]	<XRAM>, R.L	<YRAM>, R0
		R.L, <XRAM>	

В формате 8a выполняются две вычислительных операции (трёх, двух или одноадресных) в сочетании с двумя параллельными пересылками, одна из которых - между ячейкой памяти данных <XRAM> и 32-разрядным регистром R.L регистрового файла, другая - из ячейки памяти данных <YRAM> в 32-разрядный регистр R0 регистрового файла. Память данных

<XRAM>, <YRAM> адресуется в соответствии с режимами адресации, поддерживаемыми соответствующими адресными генераторами AGU, AGU-Y.

Длина кода - 64 разряда.

Формат 8b

OP2 #5/S3,S4,D2	OP1[s] S1,S2,D [M]	R, R R.L, R.L	<YRAM>, R0

В формате 8b выполняются две вычислительных операции (трёх, двух или одноадресных) в сочетании с двумя параллельными пересылками, одна из которых - между двумя регистрами данных (16- или 32-разрядными), другая - из ячейки памяти данных <YRAM> в 32-разрядный регистр R0 регистрового файла. Память данных <YRAM> адресуется в соответствии с режимами адресации, поддерживаемыми AGU-Y.

Длина кода - 64 разряда.

Формат 8c

OP2[.cc] #5/S3,S4,D2	OP1[s] S1,S2,D [M]	R.L, R.L

В формате 8c по специфицированному условию выполняются две вычислительных операции (трёх, двух или одноадресных) в сочетании с одной пересылкой между двумя 32-разрядными регистрами R.L регистрового файла.

Длина кода - 64 разряда.

Формат8d

OP2 #5/S3,S4,D2	OP1[s] S1,S2,D [M]	R, RC RC, R

В формате 8d выполняются две вычислительных операции (трёх, двух или одноадресных) в сочетании с одной пересылкой между регистром данных и регистром управления.

Длина кода - 64 разряда.

Формат 9a

OP2e T,S,D	OP1e T,S,D	<XRAM>, R.Q[D] R. Q[D], <XRAM>	<YRAM>, R0.Q[D]

В формате 9a выполняются две вычислительных операции в сочетании с двумя параллельными пересылками, одна из которых - между ячейкой памяти данных <XRAM> и 64-разрядным регистром Rn.D, либо 128-разрядным регистром Rn.Q регистрового файла, другая - из ячейки памяти данных <YRAM> в 64-разрядный R0.D, либо 128-разрядный регистр R0.Q регистрового файла. Память данных <XRAM>, <YRAM> адресуется в соответствии с режимами адресации, поддерживаемыми соответствующими адресными генераторами AGU, AGU-Y.

При написании инструкций в форматах 9a, 9b разрядность обменов мнемонически задается после номера регистра RF и точки ключом «D» для 64-разрядных обменов или «Q» для 128-разрядных обменов.

Длина кода - 64 разряда.

Формат 9b

OP2e T,S,D	OP1e T,S,D	R, R R.L, R.L	<YRAM>, R0

В формате 9b выполняются две вычислительных операции в сочетании с двумя параллельными пересылками, одна из которых - между двумя 64-разрядными (либо 128-разрядными) регистрами регистрового файла Rn.D(Q), другая - из ячейки памяти данных <YRAM> в 64(128)-разрядный R0.D (Q) регистрового файла.

Длина кода - 64 разряда.

Формат 9d

OP2e T,S,D	OP1e T,S,D	R, RC RC, R

В формате 9d выполняются две вычислительных операции в сочетании с параллельной пересылкой между регистром управления и регистром данных. При написании инструкции формата 9d разрядность пересылки задается в поле «Пересылка 1» после номера регистра RF и точки ключом «D» для 64-разрядных обменов или «L» для 32-разрядных обменов. Для 16-разрядных пересылок ключ не указывается.

Длина кода - 64 разряда.

8.2 Два типа операций

8.2.1 Параллельное выполнение двух вычислительных операций в DSP возможно только при условии, что они выполняются на разных операционных устройствах (ОУ). Например, не могут одновременно исполняться операции AND и OR, так как обе они должны исполняться при помощи одного и того же логического устройства LU.

Вычислительные команды в зависимости от исполняющего их операционного устройства делятся на две большие группы (два типа) - OP1 и OP2 лля базовой системы инструкций; OP1e и OP2e - лля расширения системы инструкций. К первому типу (OP1 или OP1e) относятся операции, исполняемые при помощи арифметического или логического устройства (AU, LU), ко второму типу (OP2 или OP2e) - операции, исполняемые при помощи умножителя-сдвигателя MS.

Только принадлежность к различным типам дает возможность двум операциям исполняться одновременно, то есть одновременное исполнение двух вычислительных команд возможно только в следующих сочетаниях:

<Операция OP2> < Операция OP1> <Пересылка 1> <Пересылка 2>,

Либо:

<Операция OP2e> < Операция OP1e> <Пересылка 1> <Пересылка 2>.

В таблице 8.2 в алфавитном порядке дан перечень операций базовой системы инструкций с указанием их типа и форматов инструкций, в которых данные операции могут применяться, а также кода операции (КОП). В таблице 8.3 дан аналогичный перечень вычислительных операций расширения системы инструкций.

Таблица 8.2 - Перечень операций базовой системы инструкций

Мнемоника	Тип	Содержание команды (формат данных)	Форматы	КОП	#
ABS	OP1	Абсолютное значение (short)	1,4,5,6,7,8	001 0000	x
ABSL	OP1	Абсолютное значение (long)	1,4,5,6,7,8	011 0000	x
ADC	OP1	Сложение с переносом (short)	1,2,3,4,5,6,7,8	000 0101	x
ADC16L	OP1	Сложение смешанное	1,2,4,5,6,7,8	000 1000	x
ADCL	OP1	Сложение с переносом (long)	1,2,4,5,6,7,8	010 0101	x
AD1	OP1	Сложение и инкремент (short)	1,2,3,4,5,6,7,8	010 1111	x
ADD	OP1	Сложение (short)	1,2,3,4,5,6,7,8	000 0100	x
ADDL	OP1	Сложение (long)	1,2,4,5,6,7,8	010 0100	x
ADDL	OP2	Сложение (long)	1,2,4,5,6,8	111 1110	1
ADDLR	OP1	Сложение (long) с округлением	1,2,4,5,6,7,8	010 1001	x
ADDLRTR	OP1	Сложение (long) с округлением и преобразованием формата (в short)	1,4,5,6,7,8	010 1010	x
ADDSUB	OP1	Сложение-вычитание (short)	1, 4,5,6,8 *)	000 0111	x
ADDSUBL	OP1	Сложение-вычитание (long)	1, 4,5,6,8 *)	001 1011	x
ADDSUBX	OP1	Сложение-вычитание (X16)	1, 4,5,6,8 *)	010 0000	x
ADDX	OP1	Сложение комплексное (X16)	1,2,4,5,6,7,8	010 0111	x
AND	OP1	Логическое И (short)	1,2,3,4,5,6,7,8	100 0001	x

Продолжение таблицы 8.2

Мнемоника	Тип	Содержание команды (формат данных)	Форматы	КОП	
ANDC	OP1	Логическое И с инверсией (short)	1,2,3,4,5,6,7,8	100 0010	x
ANDCL	OP1	Логическое И с инверсией (long)	1,2,4,5,6,7,8	101 0010	x
ANDI	OP1	Инверсия логического И (short)	1,2,3,4,5,6,7,8	100 0011	x
ANDL	OP1	Логическое И (long)	1,2,4,5,6,7,8	101 0001	x
ASH	OP1	Сложение и вычитание двух пар чисел (short)	1,4,5,6,7,8	011 1110	x
ASL	OP2	Арифметический сдвиг влево (short)	1,4,5,6,7,8	110 0100	x
ASLL	OP2	Арифметический сдвиг влево (long)	1,4,5,6,7,8	110 1100	x
ASLX	OP2	Арифметический сдвиг влево (X16)	1,4,5,6,7,8	110 0101	x
ASR	OP2	Арифметический сдвиг вправо (short)	1,4,5,6,7,8	111 0100	x
ASRL	OP2	Арифметический сдвиг вправо (long)	1,4,5,6,7,8	111 1100	x
ASRLE	OP2	Условный арифметический сдвиг вправо (long)	1,4,5,6,8 **)	110 1101	1
ASRX	OP2	Сдвиг арифметический вправо (X16)	1,4,5,6,7,8	111 0101	x
B	OP1	Ветвление программы	3m,3mb	001 1100	x
BD	OP1	Ветвление программы (отложенное)	3m,3mb	001 1110	x
BS	OP1	Вызов подпрограммы	3m,3mb	010 1100	x
BTST	OP2	Проверка разряда (short)	4,5,6,7,8	111 0000	x
BTSTL	OP2	Проверка разряда (long)	4,5,6,7,8	111 1010	x
CLR	OP1	Обнуление (очистка) регистра (short)	1,4,5,6,7,8	000 0001	x

CLRL	OP1	Обнуление (очистка) регистра (long)	1,4,5,6,7,8	010 0001	x
CMP	OP1	Сравнение (short)	1,2,3,4,5,6,7,8	001 0101	x
CMPE	OP1	Сравнение экспонент	1,2,4,5,6,7,8	010 1110	x
CMPL	OP1	Сравнение (long)	1,2,4,5,6,8	011 0101	x
CMPM	OP1	Сравнение модулей (short)	1,2,3,4,5,6,7,8	001 0110	x
CMPML	OP1	Сравнение модулей (long)	1,2,4,5,6,7,8	011 0110	x
CS2	OP2	Парная операция выбора большего из двух чисел (short) с фиксацией бита выбора	1,8	110 0110	x
CVEF	OP1	Преобразование формата: 32E16 в 24E8	1,8	001 1100	x
CVFE	OP1	Преобразование формата: 24E8 в 32E16	1,8*)	001 1100	x
CVFI	OP1	Преобразование формата: 24E8 в 32-разр. целое в дополнительном коде	1,4,5,6,7,8	000 1110	x
CVIF	OP1	Преобразование формата: 32-разрядное целое в дополнительном коде со знаком в формат 24E8	1,4,5,6,7,8	000 1111	x
DEC	OP1	Декремент (short)	1,4,5,6,7,8	001 0010	x
DECL	OP1	Декремент (long)	1,4,5,6,7,8	011 0010	x
DISPFX	OP1	Распаковка (дробная) X8 в X16	1,4,5,6,7,8	100 1110	x
DISPX	OP1	Распаковка (целочисленная) X8 в X16	1,4,5,6,7,8	100 1101	x
DO	OP1	Оператор цикла	2d,3	000 1100 000 1101 000 1110 000 1111	x
DOFOR	OP1	Оператор бесконечного цикла	3	000 1010 000 1011	x
ENDDO	OP1	Окончание цикла	3	001 1011	x
EOR	OP1	Логическое исключающее ИЛИ (short)	1,2,3,4,5,6,7,8	100 1000	x
EORL	OP1	Логическое исключающее ИЛИ (long)	1,2,4,5,6,7,8	101 1000	x
FADD	OP1	Сложение (24E8)	1,2,4,5,6,7,8	000 1010	x
FAS	OP1	Сложение-вычитание (24E8)	1,8 *)	000 1011	x

Продолжение таблицы 8.2

Мнемоника	Тип	Содержание команды (формат данных)	Форматы	КОП	#
FIN	OP2	1-я итерация обратной величины	1,4,5,6	001 1110	x
FINR	OP2	1-я итерация обратной величины квадратного корня	1,4,5,6	001 1111	x
FINT	OP1	Округление к ближайшему целому (24E8)	1,4,5,6,7,8	000 1101	x
FLOOR	OP1	Округление к ближайшему целому (24E8)	1,4,5,6,7,8	000 1100	x
FMPY	OP2	Умножение (24E8)	1,2,4,5,6,7,8	110 1111	x
FSUB	OP1	Вычитание (24E8)	1,2,4,5,6,7,8	010 1101	x
FTR	OP1	Преобразование формата	1,4,5,6,7,8	000 0110	x
FTRFL	OP1	Преобразование формата	1,4,5,6,7,8	000 1001	x
FTRL	OP1	Преобразование формата	1,4,5,6,7,8	010 0110	x
FTST	OP1	Определение признаков операнда (24E8)	1,4,5,6,7,8	010 1100	x
INC	OP1	Инкремент (short)	1,4,5,6,7,8	000 0010	x
INCL	OP1	Инкремент (long)	1,4,5,6,7,8	010 0010	x
INSL	OP1	Побитное мультиплексирование (long)	1,8 ***)	101 0100	x
J	OP1	Программный переход	3m,3mb	001 1101	x
JD	OP1	Программный переход (отложенный)	3m,3mb	001 1110	x
JS	OP1	Вызов подпрограммы	3m,3mb	010 1101	x
LSL	OP2	Логический сдвиг влево (short)	1,4,5,6,7,8	110 0001	x
LSLL	OP2	Логический сдвиг влево (long)	1,4,5,6,7,8	110 1000	x
LSLX	OP2	Логический сдвиг влево (X16)	1,4,5,6,7,8	110 0010	x
LSR	OP2	Логический сдвиг вправо (short)	1,4,5,6,7,8	111 0001	x
LSRL	OP2	Логический сдвиг вправо (long)	1,4,5,6,7,8	111 1000	x

LSRX	OP2	Логический сдвиг вправо (X16)	1,4,5,6,7,8	111 0010	x
MAC	OP2	Умножение целое со знаком (short) и накопление (__Int64)	1,8		1
MACL	OP2	Умножение целое со знаком (long) и накопление (__Int64)	1,8	111 1011	1
MMACX	OP2	Умножение дробное комплексно-сопряженное (X8) и целочисленное (X16)	1,4,5,6,8	111 1001	1
MAC2	OP2	Парное умножение (short) и накопление 2-х результатов (long)	1,8	110 1110	1
MAX	OP1	Выбор большего числа (short)	1,2,3,4,5,6,7,8	001 0111	x
MAXL	OP1	Выбор большего числа (long)	1,2,4,5,6,7,8	011 0111	x
MAXM	OP1	Выбор числа с большим модулем (short)	1,2,3,4,5,6,7,8	001 1001	x
MAXML	OP1	Выбор числа с большим модулем (long)	1,2,4,5,6,7,8	011 1001	x
MIN	OP1	Выбор меньшего числа (short)	1,2,3,4,5,6,7,8	001 1000	x
MINL	OP1	Выбор меньшего числа (long)	1,2,4,5,6,7,8	011 1000	x
MINM	OP1	Выбор числа с меньшим модулем (short)	1,2,3,4,5,6,7,8	001 1010	x
MINML	OP1	Выбор числа с меньшим модулем (long)	1,2,4,5,6,7,8	011 1010	x
MOVE	OP3	Пересылка данных	3,2t,6t,7t	110 0111	x
MOVE	OP3	Пересылка данных	3	110 1101	0
MOVE	OP3	Пересылка данных	3	110 1111	0
MPF	OP2	Умножение дробное со знаком (short)	1,2,3,4,5,6,7,8	111 1101	0
MPF2	OP2	Парное умножение дробное со знаком (short)	1,4,5,6,8	111 1101	1
MPF2S	OP2	Парное умножение дробное со знаком (short), с перестановкой сомножителей	1,4,5,6,8	110 0011	1

Продолжение таблицы 8.2

Мнемоника	Тип	Содержание команды (формат данных)	Форматы	КОП	#
MPSS	OP2	Умножение целое со знаком (short)	1,2,3,4,5,6,7,8	111 1110	0
MPUU	OP2	Умножение целое без знака (short)	1,2,3,4,5,6,7,8	111 1011	0
MPX	OP2	Умножение дробное комплексное (X8), второй операнд - комплексно-сопряженный	1,2,4,5,6,7,8	111 0110	0
MPYL	OP2	Умножение целое со знаком (long)	1,8	111 0011	1
MSKG	OP2	Формирование маски (short)	1,4,5,6,7,8	110 1010	x
MSKGL	OP2	Формирование маски (long)	1,4,5,6,7,8	110 1011	x
NEG	OP1	Изменение знака (short)	1,4,5,6,7,8	001 0001	x
NEGL	OP1	Изменение знака (long)	1,4,5,6,7,8	011 0001	x
NOP	OP1 OP2	Пустая операция	3,8	100 0000 110 0000	x
NOT	OP1	Логическое отрицание (short)	1,4,5,6,7,8	100 1001	x
NOTL	OP1	Логическое отрицание (long)	1,4,5,6,7,8	101 1001	x
OR	OP1	Логическое ИЛИ (short)	1,4,5,6,7,8	100 0101	x
ORC	OP1	Логическое ИЛИ с инверсией (short)	1,2,3,4,5,6,7,8	100 0110	x
ORCL	OP1	Логическое ИЛИ с инверсией (long)	1,2, 4,5,6,7,8	101 0101	x
ORI	OP1	Инверсия логического ИЛИ (short)	1,2,3,4,5,6,7,8	100 0111	x
ORL	OP1	Логическое ИЛИ (long)	1,2,4,5,6,7,8	101 0101	x
PACK	OP1	Упаковка (short)	1,4,5,6,7,8	100 1100	x
PACKL	OP1	Упаковка (long)	1,4,5,6,7,8	101 1100	x
PDN	OP1	Определение параметра денормализации (short)	1,4,5,6,7,8	100 1111	x
PDNE	OP1	Измерение параметра денормализации 16-разрядной мантиссы	1,4,5,6,7,8	100 0100	x

PDNL	OP1	Определение параметра денормализации (long)	1,4,5,6,7,8	101 1111	x
PDNLE	OP1	Измерение параметра денормализации 32-разрядной мантиссы	1,4,5,6,7,8	101 1110	x
PDNX	OP1	Определение параметра денормализации (X16)	1,4,5,6,7,8	101 1101	x
RNDL	OP1	Округление	1,4,5,6,7,8	010 1000	x
ROL	OP2	Циклический сдвиг на один разряд влево (short)	1,4,5,6,7,8	110 0011	0
ROLL	OP2	Циклический сдвиг на один разряд влево (long)	1,4,5,6,7,8	110 0111	0
ROR	OP2	Циклический сдвиг на один разряд вправо (short)	1,4,5,6,7,8	111 0011	0
RORL	OP2	Циклический сдвиг на один разряд вправо (long)	1,4,5,6,7,8	111 1001	0
RTS(RTI)	OP1	Возврат из подпрограммы(прерывания)	3m	010 0000	x
SAC2	OP2	Парное накопление (в формате long) со знаком	1,8	111 0110	1
SAH	OP1	Сложение и вычитание двух пар чисел (short)	1,4,5,6,7,8	011 1111	x
SBC	OP1	Вычитание с переносом (short)	1,2,3,4,5,6,7,8	001 0100	x
SBCL	OP1	Вычитание с переносом (long)	1,2,4,5,6,7,8	011 0100	x
SMB	OP2	Сложение бит (short)	4,5,6,7,8	111 0111	0
SMBL	OP2	Сложение бит (long)	4,5,6,8	111 0111	1

Окончание таблицы 8.2

Мнемоника	Тип	Содержание команды (формат данных)	Форматы	КОП	#
STOP	OP1	Конец программы	3m	011 1110	x
SUB	OP1	Вычитание (short)	1,2,3,4,5,6,7,8	001 0011	x
SUBL	OP1	Вычитание (long)	1,2,4,5,6,7,8	011 0011	x
SUBL	OP2	Вычитание (long)	1,2,4,5,6,8	111 1111	x
SUBLR	OP1	Вычитание (long) с округлением	1,2,4,5,6,7,8	011 1100	x
SUBLRTR	OP1	Вычитание (long) с округлением и преобразованием формата (в short)	1,2,4,5,6,7,8	011 1101	x
SUBX	OP1	Вычитание комплексное (X16)	1,2,4,5,6,7,8	011 1011	x
SWL	OP1	Перестановка (long)	1,2,4,5,6,7,8	101 1011	x
TR	OP1	Транзит (short)	4,5,6,7,8	100 1010	x
TR	OP2	Транзит (short)	4,5,6,7,8	110 0110	0
TRL	OP1	Транзит (long)	4,5,6,7,8	101 1010	x
TRL	OP2	Транзит (long)	4,5,6,7,8	110 1110	0
TST	OP1	Определение признаков операнда(short)	1,4,5,6,7,8	000 0011	x
TSTL	OP1	Определение признаков операнда (long)	1,4,5,6,7,8	010 0011	x
TSTX	OP1	Определение признаков операнда (X16)	1,4,5,6,7,8	010 1011	x

*) При DD=0 данная операция несовместима с пересылкой <YRAM←→R.L>

**) Данная операция не может сочетаться с пересылкой, источником в которой является какой-либо регистр данных (RF).

***) Данная операция не может сочетаться с пересылкой, источником в которой является какой-либо регистр данных (RF).

Таблица 8.3 Перечень операций расширения системы инструкций

Мнемоника	Тип	Содержание команды (формат данных)	Форматы инструкций	КОП

FM2	OP2e	Два умножения, (float)	9a,9b,9d	0000010
FMS2	OP2e	Два умножения с перестановкой полуслов в S, (float)	9a,9b,9d	0000011
FM4	OP2e	Четыре умножения, (float)	9a,9b,9d	1101010
FM4C	OP2e	Четыре умножения на общую константу, (float)	9a,9b,9d	1101011
FIN4	OP2e	Четыре нулевых приближения к обратной величине	9a,9b,9d	1111110
FINR4	OP2e	Четыре нулевых приближения к обратной величине от квадратного корня	9a,9b,9d	1111111
FASX	OP1e	Сложение и вычитание комплексных операндов, (float)	9a,9b,9d	0001000
FASXS	OP1e	Сложение и вычитание комплексных операндов с перестановкой в T, (float)	9a,9b,9d	0001001
FSA	OP1e	Вычитание и сложение, (float)	9a,9b,9d	0001111
FAX	OP1e	Сложение комплексных операндов, (float)	9a,9b,9d	0001010
FSX	OP1e	Вычитание комплексных операндов, (float)	9a,9b,9d	0001011
FA4	OP1e	Четыре сложения комплексных операндов, (float)	9a,9b,9d	1001010
FS4	OP1e	Четыре вычитания комплексных операндов, (float)	9a,9b,9d	1001011
ML2	OP2e	Два умножения, целые, 32 • 32 → 64	9a,9b,9d	0100111
UML	OP2e	Умножение, целое, без знака, 32 • 32→ 64	9a,9b,9d	0101000

Продолжение таблицы 8.3

Мнемоника	Тип	Содержание команды (формат данных)	Форматы инструкций	КОП
UML2	OP2e	Два умножения, целые, без знака, 32 • 32 → 64	9a,9b,9d	0101001
MF4	OP2e	Четыре умножения, дробные, 16 •16 → 32 округление → 16, сатурация	9a,9b,9d	0100101
MF8	OP2e	Восемь умножений, дробные, 16 •16 → 32 округление → 16, сатурация	9a,9b,9d	0100110
MFX	OP2e	Умножение комплексное, дробное, (16+j16) • (16+j16) → (32+j32) округление → (16+j16)	9a,9b,9d	0100001
MFXC	OP2e	Умножение комплексное с сопряжением SX, дробное, (16+j16) • (16+j16) → (32+j32) округление → (16+j16)	9a,9b,9d	0100010
MFX2	OP2e	Два умножения комплексные, дробные, (16+j16) • (16+j16) → (32+j32) округление → (16+j16)	9a,9b,9d	0100011
MFXC2	OP2e	Два умножения комплексные с сопряжением SX, дробные, (16+j16) • (16+j16) → (32+j32) округление → (16+j16)	9a,9b,9d	0100100
MFA81	OP2e	Сумма восьми произведений, дробные умножения, (16 • 16+32) округление → 16	9a,9b,9d	0101010
MFA41	OP2e	Сумма четырех произведений, дробные умножения, (16 • 16+32) округление → 16	9a,9b,9d	0101011
MFA42	OP2e	Две суммы четырех произведений, дробные умножения, (16 • 16 + 32) округление → 16	9a,9b,9d	0101100
MFA21	OP2e	Сумма двух произведений, дробные умножения, (16 • 16 + 32) округление →16	9a,9b,9d	0101101
MFA22	OP2e	Две суммы двух произведений, дробные умножения, (16 • 16 + 32) округление→16	9a,9b,9d	0101110
MFA24	OP2e	Четыре суммы двух произведений, дробные умножения, (16 • 16 + 32) округление → 16	9a,9b,9d	0101111

M4	OP2e	Четыре умножения, целые, 16 •16 → 32	9a,9b,9d	0110000
M2	OP2e	Два умножения, целые, 16 •16 → 32	9a,9b,9d	0110001
UMFB16	OP2e	Шестнадцать умножений, дробные, беззнаковые, разрядность 8 •8 → 16 округление → 8	9a,9b,9d	0111001
MFB16	OP2e	Шестнадцать умножений, дробные, разрядность 8 •8 → 16 округление → 8, сатурация	9a,9b,9d	0100000
CLRAC	OP2e	Групповая очистка 32-разрядных аккумуляторов ACn по маске	9a,9b,9d	1001001
LDAC	OP2e	Выгрузка содержимого 32-разрядного аккумулятора ACn в 32-разрядный регистр регистрового файла.	9a,9b,9d	1001010
LDACD	OP2e	Выгрузка содержимого 64-разрядного аккумулятора ACn.D в 64-разрядный регистр регистрового файла.	9a,9b,9d	1001011
STAC	OP2e	Загрузка 32-разрядного слова из регистра регистрового файла в 32-разрядный аккумулятор ACn.	9a,9b,9d	1001100
STACD	OP2e	Загрузка 64-разрядного слова из регистра регистрового файла в 64-разрядный аккумулятор ACn.D	9a,9b,9d	1001101

Продолжение таблицы 8.3

Мнемоника	Тип	Содержание команды (формат данных)	Форматы инструкций	КОП
MAC81	OP2e	Сумма восьми произведений с аккумуляцией, целые, 16 • 16 + 64	9a,9b,9d	1000110
MAC42	OP2e	Две суммы четырех произведений с аккумуляцией, целые, 16 • 16+64	9a,9b,9d	1010000
MAC41	OP2e	Сумма четырех произведений с аккумуляцией, целые, 16 • 16 + 64	9a,9b,9d	1000101
MAC24	OP2e	Четыре суммы двух произведений с аккумуляцией, целые, 16 • 16+64	9a,9b,9d	1010001
MAC22	OP2e	Две суммы двух произведений с аккумуляцией, целые, 16 • 16 + 64	9a,9b,9d	1010010
MAC21	OP2e	Сумма двух произведений с аккумуляцией, целые, 16 • 16 + 64	9a,9b,9d	1010011
MAC18	OP2e	Восемь умножений с аккумуляцией, целые, 16•16+64	9a,9b,9d	1010100
MAC14	OP2e	Четыре умножения с аккумуляцией, целые, 16•16+64	9a,9b,9d	1010101
MAC12	OP2e	Два умножения с аккумуляцией, целые, 16 • 16 + 64	9a,9b,9d	1010110
MAC11 (MAC)	OP2e	Умножение с аккумуляцией, целое, 16 • 16 + 64	9a,9b,9d	1010111
MACX	OP2e	Умножение с аккумуляцией, комплексное, целое, (16+j16) • (16+j16) + (64+j64)	9a,9b,9d	1000001
MACX2	OP2e	Два умножения – аккумуляция комплексных операндов, целые, (16+j16) • (16+j16) → (32+j32)+ (64+j64)	9a,9b,9d	1000010
MACXC	OP2e	Умножение с аккумуляцией, комплексное, целое, сопряжение SX, (16+j16) • (16+j16) + (64+j64)	9a,9b,9d	1000011
MACXC2	OP2e	Два умножения – аккумуляция комплексных операндов, сопряжение SX, целые, (16+j16) • (16+j16) → (32+j32)+ (64+j64)	9a,9b,9d	1000100
MACXB4	OP2e	Четыре комплексных умножения с аккумуляцией, сопряжение SBX, целые, (8+j8) • (8-j8) → (16+j16) + (64+j64)	9a,9b,9d	1000111
MACL2	OP2e	Два умножения с суммированием и аккумуляцией, целые, 32 • 32 → 64 + 64	9a,9b,9d	1001000
ACSG8	OP2e	Восемь накоплений со знаком, целые, 16 + 64	9a,9b,9d	1011000
ACSG4	OP2e	Четыре накопления со знаком, целые, 16 + 64	9a,9b,9d	1011001
ACSG2	OP2e	Два накопления со знаком, целые, 16 + 64	9a,9b,9d	1011010
ACSG1	OP2e	Накопление со знаком, целое, 16 + 64	9a,9b,9d	1011011
ACSG1	OP2e	Накопление со знаком, целое, 16 + 64	9a,9b,9d	1011011
ACB44	OP2e	Четыре накопления 4-х смежных байт в четырех 32-разрядных аккумуляторах, целые, знаковые, 8 +32	9a,9b,9d	1101110
UACB44	OP2e	Четыре накопления 4-х смежных байт в четырех 32-разрядных аккумуляторах, целые, беззнаковые, 8 +32	9a,9b,9d	1101111
AR4	OP2e	Вставка: 16-разрядный операнд вставляется в упорядоченный массив из 4-х отсчетов.	9a,9b,9d	0110100

AR4в	OP2e	Инициализирующая инструкция для AR4	9a,9b,9d	0111100
ARL2	OP2e	Вставка: 32-разрядный операнд вставляется в упорядоченный массив из 2-х отсчетов.	9a,9b,9d	0110101

Продолжение таблицы 8.3

Мнемоника	Тип	Содержание команды (формат данных)	Форматы инструкций	КОП
ARL2в	OP2e	Инициализирующая инструкция для ARL2	9a,9b,9d	0111101
AE4	OP2e	Извлечение (медианная фильтрация): 16-разрядный операнд извлекается из упорядоченного массива из четырех отсчетов.	9a,9b,9d	0110110
AE4в	OP2e	Инициализирующая инструкция для AE4	9a,9b,9d	0111110
AI4	OP2e	Вставка (медианная фильтрация): 16-разрядный операнд t вставляется в упорядоченный массив из четырех отсчетов (s_3,s_2,s_1,s_0) при условии: $t \leq s$.	9a,9b,9d	0110111
AI4в	OP2e	Инициализирующая инструкция для AI4	9a,9b,9d	0111111
COR	OP2e	Корреляция: четыре комплексных отсчета формата (16+j16) коррелируются с комплексным сопряжением от эталона в знаковом формате (1+j1) по набору из 4-х задержек. Формат накопления корреляции (32+j32).	9a,9b,9d	0111000
CORD	OP2e	Корреляция дифференциальная: восемь 16-разрядных отсчетов коррелируются с 16-разрядным эталоном по набору из 8-ми задержек. Формат накопления корреляции 32-разрядный.	9a,9b,9d	0110010
CORDв	OP2e	Инициализация для CORD	9a,9b,9d	0111010
DCR	OP2e	Декодер Витерби / Турбо – базовая операция	9a,9b,9d	0110011
DCRв	OP2e	Инициализирующая инструкция для DCR	9a,9b,9d	0111011
ASRD	OP2e	Арифметический сдвиг 64-разрядного операнда вправо Параметр сдвига 16-разрядным числом в регистре.	9a,9b,9d	1111100
ASRDI	OP2e	Арифметический сдвиг 64-разрядного операнда вправо Параметр сдвига задается непосредственно 5-разрядным числом	9a,9b,9d	1110100
ASLD	OP2e	Арифметический сдвиг 64-разрядного операнда влево Параметр сдвига задается 16-разрядным числом в регистре	9a,9b,9d	1101100
ASLDI	OP2e	Арифметический сдвиг 64-разрядного операнда влево Параметр сдвига задается непосредственно 5-разрядным числом	9a,9b,9d	1100100
LSRD	OP2e	Логический сдвиг 64-разрядного операнда вправо Параметр сдвига задается 16-разрядным числом в регистре	9a,9b,9d	1111000
LSRDI	OP2e	Логический сдвиг 64-разрядного операнда вправо Параметр сдвига задается непосредственно 5-разрядным числом	9a,9b,9d	1110001
LSLD	OP2e	Логический сдвиг 64-разрядного операнда влево Параметр сдвига задается 16-разрядным числом в	9a,9b,9d	1101000

		регистре T		

Продолжение таблицы 8.3

Мнемоника	Тип	Содержание команды (формат данных)	Форматы инструкций	КОП
LSLDI	OP2e	Логический сдвиг 64-разрядного операнда влево Параметр сдвига задается непосредственно 5-разрядным числом	9a,9b,9d	1100001
RORD	OP2e	Круговой сдвиг 64-разрядного операнда вправо на 1 разряд:	9a,9b,9d	1111001
ROLD	OP2e	Круговой сдвиг 64-разрядного операнда влево на 1 разряд:	9a,9b,9d	1101001
ASRXL	OP2e	Арифметический сдвиг компонентов комплексного операнда вправо	9a,9b,9d	1110101
ASLXL	OP2e	Арифметический сдвиг компонентов комплексного операнда влево	9a,9b,9d	1100101
LSRXL	OP2e	Логический сдвиг компонентов комплексного операнда вправо	9a,9b,9d	1110010
LSLXL	OP2e	Логический сдвиг компонентов комплексного операнда влево	9a,9b,9d	1100010
BTSTD	OP2e	Запись n-го разряда 64-разрядного операнда в разряд признака C. Номер разряда задается 16-разрядным числом в регистре	9a,9b,9d	1111010
BTSTDI	OP2e	Запись n-го разряда 64-разрядного операнда в разряд признака C. Номер разряда задается непосредственно 5-разрядным числом	9a,9b,9d	1110000
SMBD	OP2e	Подсчет количества единичных разрядов в 64-разрядном операнде	9a,9b,9d	1110111
ASRDE	OP2e	Приведение двух 64-разрядных мантисс к общей экспоненте посредством арифметического сдвига вправо одной из мантисс.	9a,9b,9d	1101101
NORVD	OP1e	Нормализация 64-разрядного операнда	9a,9b,9d	0111000
PDNDE	OP1e	Определение параметра денормализации 64-разрядной мантиссы E-формата	9a,9b,9d	1011110
NEGDE	OP1e	Изменение знака мантиссы E-формата	9a,9b,9d	0110001
A4	OP1e	Четыре сложения, целые, 16 – разрядные	9a,9b,9d	0100100
A8	OP1e	Восемь сложений, целые, 16 – разрядные	9a,9b,9d	0100101
S4	OP1e	Четыре вычитания, целые, 16 – разрядные	9a,9b,9d	0100110
S8	OP1e	Восемь вычитаний, целые, 16 – разрядные	9a,9b,9d	0100111
RA4	OP1e	Четыре скользящие суммы, целые, 16 – разрядные	9a ,9d	0101000
RA8	OP1e	Восемь скользящих сумм, целые, 16 – разрядные	9a ,9d	0101001
SGA4	OP1e	Четыре знаковых суммы, целые, 16 – разрядные	9a ,9d	0101010
SGA8	OP1e	Восемь знаковых сумм, целые, 16 – разрядные	9a ,9d	0101011
A8S	OP1e	Восемь сложений, целые, 16 – разрядные с масштабированием	9a,9b,9d	0101100
MS2	OP1e	Модули от двух вычитаний, целые, 16 – разрядные	9a,9b,9d	0101101
MS4	OP1e	Модули от четырех вычитаний, целые, 16 – разрядные	9a,9b,9d	0101110

Продолжение таблицы 8.3

Мнемоника	Тип	Содержание команды (формат данных)	Форматы ин-	КОП

			струкций	
MS8	OP1e	Модули от восьми вычитаний, целые, 16 – разрядные	9a,9b,9d	0101111
S8s	OP1e	Восемь вычитаний, целые, 16 – разрядные с масштабированием	9a,9b,9d	0110000
ALL4	OP1e	Четыре сложения 32-разрядные, целые	9a,9b,9d	0110100
ALL2	OP1e	Два сложения 32-разрядные, целые	9a,9b,9d	0110101
SLL4	OP1e	Четыре вычитания 32-разрядные, целые	9a,9b,9d	1001001
SLL2	OP1e	Два вычитания 32-разрядные, целые	9a,9b,9d	1001000
AL4	OP1e	Четыре сложения 16-разрядных и 32-разрядных операндов , целые, 16+32 → 32	9a,9b,9d	0110110
AL2	OP1e	Два сложения 16-разрядных и 32-разрядных операндов, целые, 16+32 → 32	9a,9b,9d	0110111
A81	OP1e	Сложение по 8, целые, 16-разрядное	9a,9b,9d	0111001
A42	OP1e	Два сложения по 4, целые, 16-разрядные	9a,9b,9d	0111010
A24	OP1e	Четыре сложения по 2, целые, 16-разрядные	9a,9b,9d	0111011
ALL41	OP1e	Сложение по 4, целое 32-разрядное	9a,9b,9d	0100000
ALLFT41	OP1e	Сложение по 4, целое 32-разрядное, преобразование формата дробное с округлением 32 → 16	9a,9b,9d	0110010
ALLFT22	OP1e	Два сложения по 2, целые 32-разрядные, преобразование формата дробное с округлением 32 → 16	9a,9b,9d	0110011
ASX2	OP1e	Два сложения и вычитания комплексные, целые, 16 – разрядные	9a,9b,9d	0100001
ASXS	OP1e	Сложение и вычитание комплексные, перестановка в ТХ, целые, 16 – разрядные	9a,9b,9d	0100010
ASXS2	OP1e	Два сложения и вычитания комплексные, перестановка в ТХ, целые, 16 – разрядные	9a,9b,9d	0100011
BF4	OP1e	Базовая операция FFT-4, формат целый (16+j16)	9a,9b,9d	0111100
BIF4	OP1e	Базовая операция IFFT-4, формат целый (16+j16)	9a,9b,9d	0111101
AXJ4	OP1e	Четыре комплексных сложения с предварительным умножением операндов ТХ на j , целые, 16 – разрядные	9a,9b,9d	0111110
SXJ4	OP1e	Четыре комплексных сложения с предварительным умножением операндов ТХ на j , целые, 16 – разрядные	9a,9b,9d	0111111
CMPN4	OP1e	Четыре сравнения целые 16-разрядные, выработка признаков отрицательных результатов	9a,9d	1100000
CMPN8	OP1e	Восемь сравнений целые 16-разрядные, выработка признаков отрицательных результатов	9a,9d	1100001
CMPZ4	OP1e	Четыре сравнения целые 16-разрядные, выработка признаков равенства	9a,9d	1100010
CMPZ8	OP1e	Восемь сравнений целые 16-разрядные, выработка признаков равенства	9a,9d	1100011

Продолжение таблицы 8.3

Мнемоника	Тип	Содержание команды (формат данных)	Форматы инструкций	КОП
CMPNL2	OP1e	Два сравнения целые 32-разрядные, выработка признаков отрицательных результатов	9a,9d	1100100
CMPNL4	OP1e	Четыре сравнения целые 32-разрядные, выработка признаков отрицательных результатов	9a,9d	1100101
CMPZL2	OP1e	Два сравнения целые 32-разрядные, выработка признаков равенства	9a,9d	1100110
CMPZL4	OP1e	Четыре сравнения целые 32-разрядные, выработка признаков равенства	9a,9d	1100111
MAX4	OP1e	Поиск максимума и его номера, целые 16-разрядные значения	9a,9d	1101000
MAX8	OP1e	Поиск максимума и его номера, целые 16-разрядные значения	9a,9d	1101001
MIN4	OP1e	Поиск минимума и его номера, целые 16-разрядные значения	9a,9d	1101010
MIN8	OP1e	Поиск минимума и его номера, целые 16-разрядные значения	9a,9d	1101011
MAXL2	OP1e	Поиск максимума и его номера, целые 32-разрядные значения	9a,9d	1101100
MAXL4	OP1e	Поиск максимума и его номера, целые 32-разрядные значения	9a,9d	1101101
MINL2	OP1e	Поиск минимума и его номера, целые 32-разрядные значения	9a,9d	1101110
MINL4	OP1e	Поиск минимума и его номера, целые 32-разрядные значения	9a,9d	1101111
ANDD	OP1e	Поразрядные логические «И» над 64-разрядными операндами	9a,9b,9d	1010001
ANDCD	OP1e	Поразрядные логические «И» над 64-разрядными операндами с инверсией одного из них	9a,9b,9d	1010010
ANDID	OP1e	Поразрядные логические «И» над 64-разрядными операндами с инверсией результата	9a,9b,9d	1010011
ORD	OP1e	Поразрядные логические «ИЛИ» над 64-разрядными операндами	9a,9b,9d	1010101
ORCD	OP1e	Поразрядные логические «ИЛИ» над 64-разрядными операндами с инверсией одного из них	9a,9b,9d	1010110
ORID	OP1e	Поразрядные логические «ИЛИ» над 64-разрядными операндами с инверсией результата	9a,9b,9d	1010111
EORD	OP1e	Поразрядные логические «ИСКЛЮЧАЮЩИЕ ИЛИ» над 64-разрядными операндами	9a,9b,9d	1011000
NOTD	OP1e	Поразрядные логические «НЕ» над 64-разрядным операндом	9a,9b,9d	1011001
INSD	OP1e	Поразрядное объединение двух 64-разрядных операндов по маске	9a,9b,9d	1010100
CLRD	OP1e	Сброс в ноль разрядов 64-разрядного операнда	9a,9b,9d	0000001

Продолжение таблицы 8.3

Мнемоника	Тип	Содержание команды (формат данных)	Форматы инструкций	КОП

ADDD	OP1e	Целое сложение двух 64-разрядных операндов	9a,9b,9d	0000010
SUBD	OP1e	Целое вычитание двух 64-разрядных операндов	9a,9b,9d	0000011
ADCD	OP1e	Целое сложение двух 64-разрядных операндов с добавлением переноса	9a,9b,9d	0000100
SBCD	OP1e	Целое вычитание двух 64-разрядных операндов с вычитанием переноса	9a,9b,9d	0000101
ADDLD	OP1e	Целое знаковое сложение 32-разрядного и 64-разрядного операнда с образованием 64-разрядного результата	9a,9b,9d	0000110
TRD	OP1e OP2e	Пересылка 64-разрядного операнда	9a,9b,9d	1011010
PDND	OP1e	Определение параметра денормализации 64-разрядного операнда	9a,9b,9d	1011111
PDNXL	OP1e	Определение параметра денормализации комплексного операнда с 32-разрядными компонентами	9a,9b,9d	1011101
TRS0	OP1e OP2e	Пересылка 128-разрядного операнда.	9a,9b,9d	0010000
TRS1	OP1e OP2e	Сдвиг двух 64-разрядных операндов на одно 16-разрядное слово	9a,9b,9d	0010001
TRS2	OP1e OP2e	Сдвиг двух 64-разрядных операндов на два 16-разрядных слова	9a,9b,9d	0010010
TRS3	OP1e OP2e	Сдвиг двух 64-разрядных операндов на три 16-разрядных слова	9a,9b,9d	0010011
TRS4	OP1e OP2e	Преобразование восьми 8-разрядных операндов в 16—разрядные, дробный формат	9a,9b,9d	0010100
TRS5	OP1e OP2e	Преобразование восьми 8-разрядных операндов в 16—разрядные, целый формат, расширение знака	9a,9b,9d	0010101
TRS6	OP1e OP2e	Преобразование восьми 8-разрядных операндов в 16—разрядные, целый формат, без знака	9a,9b,9d	0010110
TRS7	OP1e OP2e	Преобразование восьми 16-разрядных операндов в 8—разрядные, дробный формат, округление, сатурация	9a,9b,9d	0010111
TRS8	OP1e OP2e	Преобразование восьми 16-разрядных операндов в 8—разрядные, целый формат, без знака, сатурация	9a,9b,9d	0011000
TRS9	OP1e OP2e	Перестановка четырех 16-разрядных операндов инверсная	9a,9b,9d	0011001
TRS10	OP1e OP2e	Перестановка четырех 16-разрядных операндов двоично – инверсная	9a,9b,9d	0011010
TRS11	OP1e OP2e	Вложение двух векторов из двух 32-разрядных операндов	9a,9b,9d	0011011
TRS12	OP1e OP2e	Вложение двух векторов из четырех 16-разрядных операндов	9a,9b,9d	0011100
TRS13	OP1e OP2e	Группировка двух векторов из четырех 16-разрядных операндов каждый по четным и нечетным элементам	9a,9b,9d	0011101

Окончание таблицы 8.3

Мнемоника	Тип	Содержание команды (формат данных)	Форматы инструкций	КОП
TRS14	OP1e OP2e	Два преобразования трех 16-разрядных операндов в составной 16-разрядный код (5,6,5 разрядов).	9a,9b,9d	0011110
TRS15	OP1e	Преобразование четырех 32-разрядных операндов в	9a,9b,9d	0011111

	OP2e	16-разрядные, дробный формат		
TRS16	OP1e OP2e	Преобразование четырех 32-разрядных операндов в 16-разрядные, целый формат	9a,9b,9d	0000111
TRS17	OP1e OP2e	Преобразование четырех 32-разрядных операндов в 16-разрядные, целый формат	9a,9b,9d	0001100
TRS18	OP1e OP2e	Поразрядное вложение двух 16-разрядных операндов с образованием 32-разрядного операнда	9a,9b,9d	0001101
TRS19	OP1e OP2e	Преобразование восьми 16-разрядных операндов в 8—разрядные, целый формат	9a,9b,9d	0001110
DRGB	OP1e OP2e	Упаковка/распаковка байтных RGB-компонентов изображения для четырех пикселей	9a,9b,9d	1011011 0001010
TRSB	OP1e OP2e	Сдвиг двух 128-разрядных операндов на один байт	9a,9b,9d	1010000 0001001
AB16	OP1e OP2e	Шестнадцать сумм, целые, 8-разрядные, беззнаковые	9a,9b,9d	1000101 0000001
SB16	OP1e OP2e	Шестнадцать разностей, целые, 8-разрядные, беззнаковые	9a,9b,9d	1000111 0000110
MSB16	OP1e OP2e	Шестнадцать модулей разностей, целые, 8-разрядные, беззнаковые	9a,9b,9d	1001111 0001000
UPK1	OP2e	Распаковка четырех байт целого беззнакового формата, преобразование к формату плавающей точки с масштабным коэффициентом	9a,9b,9d	1100011
UPK2	OP2e	Распаковка трех целых беззнаковых разрядности 5-6-5, преобразование к формату плавающей точки с масштабным коэффициентом	9a,9b,9d	1100111
UPK3	OP2e	Распаковка четырех беззнаковых целых разрядности 5-5-5-1, преобразование к формату плавающей точки с масштабным коэффициентом	9a,9b,9d	1110011
UPK4	OP2e	Распаковка четырех беззнаковых целых разрядности 4-4-4-4, преобразование к формату плавающей точки с масштабным коэффициентом	9a,9b,9d	1110110
PK1	OP2e	Преобразование четырех чисел формата плавающей точки в целые беззнаковые байты, упаковка байт в 32-разрядное слово	9a,9b,9d	1101110
PK2	OP2e	Преобразование трех чисел формата плавающей точки в целые беззнаковые формата 5-6-5, упаковка результатов в 16-разрядное слово	9a,9b,9d	1101111

8.3 Коды операций базовой системы инструкций

8.3.1 Таблица кодов операций базовой системы инструкций

В таблице 8.4 приведены коды всех операций (КОП) базовой системы инструкций, включая операции управления и пересылок.

Таблица 8.4 ─ Коды операций базовой системы инструкций

AU				LU + BFU		MS	
КОП	Операция	КОП	Операция	КОП	Операция	КОП	Операция
0000000	NOP	0100000	RTS/ RTI/ ADDSUBX	1000000	NOP	1100000	NOP
0000001	*CLR*	0100001	*CLRL*	1000001	*AND*	1100001	*LSL*
0000010	*INC*	0100010	*INCL*	1000010	*ANDC*	1100010	*LSLX*
0000011	*TST*	0100011	*TSTL*	1000011	*ANDI*	1100011	*ROL /MPF2S*
0000100	*ADD*	0100100	*ADDL*	1000100	*PDNE*	1100100	*ASL*
0000101	*ADC*	0100101	*ADCL*	1000101	*OR*	1100101	*ASLX*
0000110	*FTR*	0100110	*FTRL*	1000110	*ORC*	1100110	*TR /CS2*
0000111	*ADDSUB*	0100111	*ADDX*	1000111	*ORI*	1100111	*MOVE.cc*
0001000	*ADC16L*	0101000	*RNDL*	1001000	*EOR*	1101000	*LSLL*
0001001	*FTRFL*	0101001	*ADDLR*	1001001	*NOT*	1101001	*ROLL /MAC*
0001010	*DOFOR /FADD*	0101010	*ADDLRTR*	1001010	*TR*	1101010	*MSKG*
0001011	*DOFORR /FAS*	0101011	*TSTX*	1001011		1101011	*MSKGL*
0001100	*DO /FLOOR*	0101100	*BS /FTST*	1001100	*PACK*	1101100	*ASLL*
0001101	*DOR /FINT*	0101101	*JS /FSUB*	1001101	*DISPX*	1101101	*MIA /ASRLE*
0001110	*DO_R /CVFI*	0101110	*CMPE*	1001110	*DISPFX*	1101110	*TRL /MAC2*
0001111	*DOR_R /CVIF*	0101111	*AD1*	1001111	*PDN*	1101111	*MIP /FMPY*
0010000	*ABS*	0110000	*ABSL*	1010000		1110000	BTST
0010001	*NEG*	0110001	*NEGL*	1010001	*ANDL*	1110001	*LSR*
0010010	*DEC*	0110010	*DECL*	1010010	*ANDCL*	1110010	*LSRX*
0010011	*SUB*	0110011	*SUBL*	1010011		1110011	*ROR /MPYL*
0010100	*SBC*	0110100	*SBCL*	1010100	*INSL*	1110100	*ASR*
0010101	*CMP*	0110101	*CMPL*	1010101	*ORL*	1110101	*ASRX*
0010110	*CMPM*	0110110	*CMPML*	1010110	*ORCL*	1110110	*MPX /SAC2*
0010111	*MAX*	0110111	*MAXL*	1010111		1110111	*SMB /SMBL*
0011000	*MIN*	0111000	*MINL*	1011000	*EORL*	1111000	*LSRL*
0011001	*MAXM*	0111001	*MAXML*	1011001	*NOTL*	1111001	*RORL /MMACX*
0011010	*MINM*	0111010	*MINML*	1011010	*TRL*	1111010	*BTSTL*
0011011	ENDDO /ADDSUBL	0111011	*SUBX*	1011011	*SWL*	1111011	*MPUU /MACL*
0011100	*B /CVEF*	0111100	*SUBLR*	1011100	*PACKL*	1111100	*ASRL*
0011101	*J /CVFE*	0111101	*SUBLRTR*	1011101	*PDNX*	1111101	*MPF /MPF2*
0011110	*BD /FIN*	0111110	*STOP /ASH*	1011110	*PDNLE*	1111110	*MPSS /ADDL*
0011111	*JD /FINR*	0111111	*SAH*	1011111	*PDNL*	1111111	*SUBL*

8.3.2 Операции с несколькими вариантами кодировки

Операции MOVE, DO, DOFOR могут иметь различные варианты кодировки в зависимости от параметров данных операций.

8.3.2.1 Варианты кодировки операции MOVE. Операция пересылки может иметь следующие варианты кодировки:

1) операции MOVE *не имеет кода операции* в тех случаях, когда наличие и тип пересылки определяется кодом формата;

2) условные пересылки имеют код MOVE.cc (1100111);

3) пересылки непосредственных данных в регистры адресных генераторов AGU, AGU-Y имеют код MIA (1101101);

4) пересылки непосредственных данных в регистры программного управления PCU имеют код MIP (1101111).

8.3.2.2 Варианты кодировки операции DO. Операция цикла DO может иметь два варианта кодировки:

1) операция DO с абсолютной адресацией последней команды цикла и непосредственно заданным числом повторений имеет код DO (0001100);

2) операция DO с адресацией последней команды цикла относительно программного счетчика PC и непосредственно заданным числом повторений имеет код DOR (0001101);

3) операция DO с абсолютной адресацией последней команды цикла и косвенно заданным числом повторений имеет код DO_R (0001110);

4) операция DO с адресацией последней команды цикла относительно программного счетчика PC и косвенно заданным числом повторений имеет код DOR_R (0001111).

8.3.2.3 Варианты кодировки операции DOFOR. Операция бесконечного цикла DOFOR может иметь два варианта кодировки:

1) операция DOFOR с абсолютной адресацией последней команды цикла имеет код DOFOR (0001010);

2) операция DOFOR с адресацией последней команды цикла относительно программного счетчика PC имеет код DOFORR (0001011).

8.3.3 Расширение кода операции за счет формата инструкции и бита #

В некоторых случаях двум различным операциям соответствует одинаковый код. Выбор исполняемой операции в этих случаях производится с учетом либо кода формата, либо бита # (бит непосредственного операнда #5 в командах сдвига).

В тех случаях, когда код вычислительной операции совпадает с кодом операции управления или пересылки, выбор исполняемой операции определяется форматом инструкции согласно таблице 8.5.

Для кодирования команд, реализуемых на базе устройства умножения-сдвига (MS) и относящихся к группе операций OP2, производится расширение кода операции за счет бита # (бит непосредственного операнда #5 в командах сдвига) согласно таблице 8.6.

8.3.4 Кодирование операции RTI

Команда возврата из подпрограммы обработки прерывания RTI кодируется так же, как команда RTS (формат 3m), с тем отличием, что 27-й бит слова инструкции устанавливается при команде RTS в «1», а при команде RTI в «0» (формат 3mb).

Таблица 8.5

Расширение кода операции за счет формата инструкции

КОП	Формат	
	2d, 3	другие
0001010	DOFOR	FADD
0001011	DOFORR	FAS
0001100	DO	FLOOR
0001101	DOR	FINT
0001110	DO_R	CVFI
0001111	DOR_R	CVIF
0011011	ENDDO	ADDSUBL
0011100	B	CVEF
0011101	J	CVFE
0011110	BD	FIN
0011111	JD	FINR
0100000	RTS	ADDSUBX
0101100	BS	FTST
0101101	JS	FSUB
0111110	STOP	ASH
0111111	-	SAH
1101101	MOVE(MIA)	ASRLE
1101111	MOVE(MIP)	FMPY

Таблица 8.6

Расширение кода операции за счет бита #

КОП	Операция	
	# = 0	# = 1
1100011	ROL	MPF2S
1100110	TR	CS2
1101001	ROLL	MAC
1101110	TRL	MAC2
1110011	ROR	MPYL
1110110	MPX	SAC2
1110111	SMB	SMBL
1111001	RORL	MMACX
1111011	MPUU	MACL
1111101	MPF	MPF2
1111110	MPSS	ADDL(OP2)
1111111	-	SUBL(OP2)

8.4 Коды операций расширения системы инструкций

8.4.1 Таблицы кодов операций расширения системы инструкций

Коды операций (КОП) расширения системы инструкций DSP Elcore-30M приведены в таблицах 8.7 (тип операций OP1e) и 8.8 (тип операций OP1e).

Таблица 8.7 ─ Коды операций типа OP1e в форматах 9a, 9b, 9d DSP-ядра Elcore-30M

FAS + AU + TU		SUM		LU		SU	
OP1	Операция	OP1	Операция	OP1	Операция	OP1	Операция
0000000	NOP	0100000	ALL41	1000000	NOP	1100000	CMPN4
0000001	CLRD	0100001	ASX2	1000001	CMPNB16	1100001	CMPN8
0000010	ADDD	0100010	ASXS	1000010		1100010	CMPZ4
0000011	SUBD	0100011	ASXS2	1000011	CMPZB16	1100011	CMPZ8
0000100	ADCD	0100100	A4	1000100		1100100	CMPNL2
0000101	SBCD	0100101	A8	1000101	AB16	1100101	CMPNL4
0000110	ADDLD	0100110	S4	1000110		1100110	CMPZL2
0000111	TRS16	0100111	S8	1000111	SB16	1100111	CMPZL4
0001000	FASX	0101000	RA4	1001000	SLL2	1101000	MAX4
0001001	FASXS	0101001	RA8	1001001	SLL4	1101001	MAX8
0001010	FAX	0101010	SGA4	1001010	FA4	1101010	MIN4
0001011	FSX	0101011	SGA8	1001011	FS4	1101011	MIN8
0001100	TRS17	0101100	A8s	1001100		1101100	MAXL2
0001101	TRS18	0101101	MS2	1001101		1101101	MAXL4
0001110	TRS19	0101110	MS4	1001110		1101110	MINL2
0001111	FSA	0101111	MS8	1001111	MSB16	1101111	MINL4
0010000	TRS0	0110000	S8s	1010000	TRSB	1110000	
0010001	TRS1	0110001	NEGDE	1010001	ANDD	1110001	
0010010	TRS2	0110010	ALLFT41	1010010	ANDCD	1110010	
0010011	TRS3	0110011	ALLFT22	1010011	ANDID	1110011	
0010100	TRS4	0110100	ALL4	1010100	INSD	1110100	
0010101	TRS5	0110101	ALL2	1010101	ORD	1110101	
0010110	TRS6	0110110	AL4	1010110	ORCD	1110110	
0010111	TRS7	0110111	AL2	1010111	ORID	1110111	
0011000	TRS8	0111000	NORVD	1011000	EORD	1111000	
0011001	TRS9	0111001	A81	1011001	NOTD	1111001	
0011010	TRS10	0111010	A42	1011010	TRD	1111010	
0011011	TRS11	0111011	A24	1011011	DRGB	1111011	
0011100	TRS12	0111100	BF4	1011100		1111100	
0011101	TRS13	0111101	BIF4	1011101	PDNXL	1111101	
0011110	TRS14	0111110	AXJ4	1011110	PDNDE	1111110	
0011111	TRS15	0111111	SXJ4	1011111	PDND	1111111	

Таблица 8.8. ─ Коды операций типа OP2e в форматах 9a, 9b, 9d DSP-ядра Elcore-30M

FMU + COR + TU		MU + SFU		MAC		MS	
{M,OP2}	Операция	{M,OP2}	Операция	{M,OP2}	Операция	{M,OP2}	Операция
0000000	NOP	0100000	MFB16	1000000	NOP	1100000	NOP
0000001	AB16	0100001	MFX	1000001	MACX	1100001	LSLDi
0000010	FM2	0100010	MFXC	1000010	MACX2	1100010	LSLXL
0000011	FMS2	0100011	MFX2	1000011	MACXC	1100011	UPK1
0000100		0100100	MFXC2	1000100	MACXC2	1100100	ASLDi
0000101		0100101	MF4	1000101	MAC41	1100101	ASLXL
0000110	SB16	0100110	MF8	1000110	MAC81	1100110	TRD
0000111	TRS16	0100111	ML2	1000111	MACXB4	1100111	UPK2
0001000	MSB16	0101000	UML	1001000	MACL2	1101000	LSLD
0001001	TRSB	0101001	UML2	1001001	CLRAC	1101001	ROLD
0001010	DRGB	0101010	MFA81	1001010	LDAC	1101010	FM4
0001011		0101011	MFA41	1001011	LDACD	1101011	FM4C
0001100	TRS17	0101100	MFA42	1001100	STAC	1101100	ASLD
0001101	TRS18	0101101	MFA21	1001101	STACD	1101101	ASRDE
0001110	TRS19	0101110	MFA22	1001110	ACB44	1101110	PK1
0001111		0101111	MFA24	1001111	UACB44	1101111	PK2
0010000	TRS0	0110000	M4	1010000	MAC42	1110000	BTSTDi
0010001	TRS1	0110001	M2	1010001	MAC24	1110001	LSRDi
0010010	TRS2	0110010	CORD	1010010	MAC22	1110010	LSRXL
0010011	TRS3	0110011	DCR	1010011	MAC21	1110011	UPK3
0010100	TRS4	0110100	AR4	1010100	MAC18	1110100	ASRDi
0010101	TRS5	0110101	ARL2	1010101	MAC14	1110101	ASRXL
0010110	TRS6	0110110	AE4	1010110	MAC12	1110110	UPK4
0010111	TRS7	0110111	AI4	1010111	MAC11	1110111	SMBD
0011000	TRS8	0111000	COR	1011000	ACSG8	1111000	LSRD
0011001	TRS9	0111001	UMFB16	1011001	ACSG4	1111001	RORD
0011010	TRS10	0111010	CORDb	1011010	ACSG2	1111010	BTSTD
0011011	TRS11	0111011	DCRb	1011011	ACSG1	1111011	
0011100	TRS12	0111100	AR4b	1011100		1111100	ASRD
0011101	TRS13	0111101	ARL2b	1011101		1111101	
0011110	TRS14	0111110	AE4b	1011110		1111110	FIN4
0011111	TRS15	0111111	AI4b	1011111		1111111	FINR4

8.5 Таблица полей форматов инструкций

8.5.1 Участки кода инструкции, определяющие те или иные параметры входящих в нее операций, называются полями инструкции. В таблице 8.9 определяется структура форматов инструкций и назначение входящих в них полей.

Таблица 8.9 - Таблица полей форматов инструкций

Разряды слова инструкции																															Формат
31	30	29	28	27	26	25	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	
S1 / s1 / #5					D / d					S2 / s2					#	0	cc				0	0	0	1	OP							1
xxxxx					D / d					S2 / s2					0	0	cc				0	0	1	0	OP							2
#32 / #I16																															
xxxxx					#16																0	0	1	0	OP							2d
#16																															
d					#16																0	0	1	1	OP							3
cc				1	#16																0	0	1	1	OP							3 m
cc				0	xxxxx									A							0	0	1	1	OP							3mb
S / s / #5					D / d					R				A			mode			u	0	1	de	#	OP							4
xxxxx					cc				0	R				A			mode			u	0	1	de	0	OP							6t
xxxxx					cc				1	R				A			010			0	0	1	de	0	OP							7t
#16																															
S / s / #5					D / d					R					sc	RC					1	0	de	#	OP							6
sx	L2	x	sr	L	cc				0	s/d/s/S					sc	RC/RC/d/D					1	0	de	0	OP							2t
S / s / #5					D / d					RS /Rs					L	RD /Rd					1	1	0	#	OP							5
S / s					D / d					RD /Rd / RC					L/ sc	sr	cc				1	1	1	0	OP							7
#32 / #16																															
S1/s1					D/d					S2/s2					M	R				u	0	0	0	0	OP1							8a
S3/s3/ #5					D2/d2					S4/s4					AT		mode			A			de	OP3		#	OP2				OP3=00
S1/s1					D/d					S2/s2					M	RS/Rs					0	0	0	0	OP1							8b
S3/s3/ #5					D2/d2					S4/s4					AT		RD/Rd					L	x	OP3		#	OP2				OP3=01
S1/s1					D/d					S2/s2					M	RS				x	0	0	0	0	OP1							8c
S3/s3/ #5					D2/d2					S4/s4					cc		RD				x	cc		OP3		#	OP2				OP3=10
S1/s1					D/d					S2/s2					M	Rs/Rd					0	0	0	0	OP1							8d
S3/s3/ #5					D2/d2					S4/s4					sx	L	RC					sc	de	OP3		#	OP2				OP3=11
T (OP1e)					D (OP1e)					S (OP1e)					M	R				u	1	1	1	1	OP1							9a
T (OP2e) / #5					D (OP2e)					S (OP2e)					AT		mode			A			de	OP3			OP2				OP3=00
T (OPe)					D (OP1e)					S (OP1e)					M	RS					1	1	1	1	OP1							9b
T (OP2e) / #5					D (OP2e)					S (OP2e)					AT		RD					L	u	OP3			OP2				OP3=01
T (OP1e)					D (OP1e)					S (OP1e)					M	Rs/Rd					1	1	1	1	OP1							9d
T (OP2e) / #5					D (OP2e)					S (OP2e)					sx	L	RC					sc	de	OP3			OP2				OP3=11

Примечание. Обозначения:

`S,D,RS,RD,R` – 32-разрядные регистры RF

`s,d,Rs,Rd` – 16-разрядные регистры RF

`#n` – `n` -разрядные непосредственные данные

`RC` – управляющие регистры (включая AGU, PCU, ACn, XBUF)

`cc` – код условия

`A` – номер генератора адреса

`mode` – код режима адресации AGU

`u` – код блокировки обмена (0 – обмен, 1 – блокировка)
либо разрядности пересылки в форматах 9a, 9b

`x` – произвольное значение разряда

8.6 Описание полей форматов инструкций

В настоящем разделе содержится детальное описание кодировки полей инструкции.

Здесь и в дальнейшем, при описании форматов инструкций, приняты следующие *соглашения по обозначениям* (соответствующие правилам языка Verilog):

1) разряды формата данных (операнда) указываются в квадратных скобках, через двоеточие. *Пример* - S[4:0];

2) слева указывается старший разряд (в примере - 4-й), справа - младший (0-й).

8.6.1 Основное и дополнительное поле кода формата инструкции

Разряды [10:7] первого слова каждой инструкции составляют *основное поле кода формата,* в котором содержится код формата данной инструкции (см. таблицу 8.10). Производные форматы 2d, 2t, 3m, 3mb, 4t, 6t, 7t определяются не только кодом формата, но и кодом инструкции.

Таблица 8.10

Код формата (основное поле)	Формат
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 x x	4
1 0 x x	5
1 1 0 x	6
1 1 1 0	7
0 0 0 0	8
1 1 1 1	9

Среди форматов параллельных инструкций 8a, 8b, 8c, 8d, 9a, 9b, 9d выбор осуществляется при помощи *дополнительного поля кода формата* (OP3), размещенного в разрядах [7:6] второго слова инструкции (см. таблицу 8.11).

Таблица 8.11

Код формата (основное поле)	Код формата (дополнительное поле)	Формат
0 0 0 0	0 0	8a
0 0 0 0	0 1	8b
0 0 0 0	1 0	8c
0 0 0 0	1 1	8d
1 1 1 1	0 0	9a
1 1 1 1	0 1	9b
1 1 1 1	1 1	9d

8.6.2 Основное и дополнительное поле кода операции

Разряды [6:0] первого слова каждой инструкции составляют основное поле кода операции OP, в котором содержится код исполняемой операции.

В параллельных инструкциях формата 8 данное поле отводится для кодирования операции OP1, а в формате 9 - для кодирования операции OP1e.

Для кода операции OP2 в инструкциях формата 8 отводится *дополнительное поле кода операции* (разряды [4:0] второго слова инструкции). Поле OP2 содержит только пять младших разрядов кода операции, так как оба старших разряда кода операции для всех операций этого типа равны единице.

В параллельных инструкциях формата 9 поле OP2 соостоит из разрядов [5:0] второго слова инструкции. Данное поле отводится для кодирования операции OP1e, а для кода операции OP2e, отводится конкатенация поля M и.

8.6.3 Поле кода условия (cc)

Коды условия (cc) для условно исполняемых инструкций приведены в таблице 8.12.

Поле кода условия размещается:

- для форматов 1, 2, 7 - в разрядах [14:11] первого слова инструкции;
- для форматов 3m, 3mb - в разрядах [31:28] первого слова инструкции;
- для формата 8c - в разрядах {[16:15],[9:8]} второго слова инструкции.

Таблица 8.12

Коды условия для условно исполняемых инструкций

Номер кода	Код условия	Условие	Мнемоника
0	0000	C = 0	cc (Carry Clear) / hs (Higher or Same)
1	0001	C = 1	cs (Carry Set) / lo (LOwer)
2	0010	Z = 0	ne (Not Equal to zero)
3	0011	Z = 1	eq (EQual to zero)
4	0100	N = 0	pl (Plus)
5	0101	N = 1	mi (MInus)
6	0110	N^V = 0	ge (GrEater than or equal)
7	0111	N^V = 1	lt (Less Than)
8	1000	Z|(N^V) = 0	gt (Greater Than)
9	1001	Z|(N^V) = 1	le (Less than or Equal)
10	1010	U = 0	nr (NoRmalized)
11	1011	U&(~V) = 1	un (UnNormalized)
12	1100	V = 1	vs (oVerflow Set)
13	1101	V=0	vc (oVerflow clear)
14	1110	t=1	t (признак истинности условия после исполнения условной команды)
15	1111	–	al (Always)

8.6.4 Поля адресов регистров данных (T/S/S1/S2/S3/S4/D/D1/D2/s/s1/s2/s3/s4/d/d1/d2 R/RS/Rs/RD/Rd)

Исходные операнды и результаты вычислительных операций размещаются в регистрах регистрового файла (RF), который представляет собой набор реконфигурируемых 16/32/64/128-разрядных регистров R0–R31.

В таблице полей форматов поля адресов операндов обозначаются следующим образом:

S, S1, S2, S3, S4, D, D1, D2 – 32-разрядные регистры RF;

s, s1, s2, s3, s4, d, d1, d2 – 16-разрядные регистры RF.

T,S,D – 16/32/64/128-разрядные регистры RF (для форматов 9a, 9b, 9d).

Кодирование адресов регистров данных в 5-битных полях производится путем указания номера регистра от 0 до 31.

Адрес регистра данных, являющегося источником либо получателем в операциях пересылок, определяется пятиразрядным (при обменах с памятью - четырёхразрядным) кодом (R/RS/Rs/RD/Rd). При обменах с памятью используются только регистры с чётными номерами – в поле R в форматах 4, 6t,7t, 8a, 9a указываются четыре старших разряда этого номера.

8.6.5 Кодирование адресов регистров адресных генераторов AGU, AGU-Y (поле RC)

Поле RC в форматах 2t, 6, 7, 8d, 9d при обменах с регистрами адресных генераторов AGU, AGU-Y или регистром IVAR кодируется согласно таблице 8.13.

Таблица 8.13. - Кодирование адресов регистров адресных генераторов AGU, AGU-Y

Условное обозначение	Разрядность	Тип	Назначение регистра	Код адреса регистра
A0	32	R/W	Регистр адреса A0	00000
A1	32	R/W	Регистр адреса A1	00001
A2	32	R/W	Регистр адреса A2	00010
A3	32	R/W	Регистр адреса A3	00011
A4	32	R/W	Регистр адреса A4	00100
A5	32	R/W	Регистр адреса A5	00101
A6	32	R/W	Регистр адреса A6	00110
A7	32	R/W	Регистр адреса A7	00111
I0	16	R/W	Регистр смещения I0	01000
I1	16	R/W	Регистр смещения I1	01001
I2	16	R/W	Регистр смещения I2	01010
I3	16	R/W	Регистр смещения I3	01011
I4	16	R/W	Регистр смещения I4	01100
I5	16	R/W	Регистр смещения I5	01101
I6	16	R/W	Регистр смещения I6	01110
I7	16	R/W	Регистр смещения I7	01111
M0	16	R/W	Регистр модификатора M0	10000
M1	16	R/W	Регистр модификатора M1	10001
M2	16	R/W	Регистр модификатора M2	10010

Окончание таблицы 8.13

Условное обозначение	Разрядность	Тип	Назначение регистра	Код адреса регистра
M3	16	R/W	Регистр модификатора M3	10011
M4	16	R/W	Регистр модификатора M4	10100
M5	16	R/W	Регистр модификатора M5	10101
M6	16	R/W	Регистр модификатора M6	10110
M7	16	R/W	Регистр модификатора M7	10111
AT	32	R/W	Регистр адреса AT	11000
IT	16	R/W	Регистр смещения IT	11001

MT	16	R/W	Регистр модификатора MT	11010
DT	16	R/W	Регистр смещения DT	11011
-	-	-	Резерв	11100
-	-	-	Резерв	11101
-	-	-	Резерв	11110
IVAR	16	R/W	Регистр адреса вектора прерывания	11111

8.6.6 Кодирование адресов регистров управления PCU

Поле RC в форматах 2t, 6, 7, 8d, 9d при обменах с регистрами управления PCU, включая регистры состояния ALU (CCR, PDNR) и регистр специальных функций SFR, кодируется согласно таблице 8.14.

Таблица 8.14 ─ Кодирование адресов регистров управления PCU

Условное обозначение	Разряд-ность	Тип	Назначение регистра	Код адреса регистра
Регистры управления				
DCSR	16	R/W	Регистр режима работы	00000
SR	16	R/W	Регистр состояния	00001
IDR	16	R	Регистр-идентификатор	00010
EFR	32	R	Регистр флагов обмена	00011
DSTART	32	W	Регистр запуска DMA со стороны DSP и запросов на прерывания других DSP	00011
IRQR	32	R/W	Регистр запросов на прерывание DSP	00100
IMASKR	32	R/W	Регистр маски запросов на прерывания DSP	00101
TMR	32	R/W	Регистр таймера DSP	00110
ARBR	16	R/W	Регистр управления арбитром памяти DSP	00111
PC	16	R/W	Программный счетчик	01000
SS	16	R/W	Стек программного счетчика	01001
LA	16	R/W	Регистр адреса цикла	01010
CSL	16	R/W	Стек адреса цикла	01011
LC	16	R/W	Счетчик циклов	01100
CSH	16	R/W	Стек счетчика циклов	01101
SP	16	R/W	Регистр указателя стека	01110
SAR	16	R/W	Регистр адреса останова	01111
CNTR	16	R/W	Счетчик исполненных команд	10000
SAR1	16	R/W	Регистр адреса останова	10001
SAR2	16	R/W	Регистр адреса останова	10010
SAR3	16	R/W	Регистр адреса останова	10011

Окончание таблицы 8.14

Условное обозначение	Разряд-ность	Тип	Назначение регистра	Код адреса регистра
SAR4	16	R/W	Регистр адреса останова	10100
SAR5	16	R/W	Регистр адреса останова	10101
SAR6	16	R/W	Регистр адреса останова	10110
SAR7	16	R/W	Регистр адреса останова	10111
Регистры состояния ALU				
PDNR	16	R/W	Регистр параметра денормализации	11000
CCR	16	R/W	Регистр кодов условий	11001
SFR	32	R/W	Регистр специальных функций	11010

8.6.7 Кодирование адресов регистров-аккумуляторов

Поле RC в форматах 2t, 8d, 9d при обменах с регистрами-аккумуляторами ACn.[L/D/Q] кодируется путем указания номера регистра-аккумулятора n от 0 до 15.

8.6.8 Кодирование адресов регистров обменного буфера XBUF

Кодирование адресов регистров обменного буфера XBUF в 5-битном поле “RC” в форматах 2t, 8d, 9d производится путем указания номера регистра XBUF от 0 до 31.

8.6.9 Поля выбора типа регистра (sr, sc, sx)

Выбор типа регистра, являющегося источником либо получателем в операциях пересылок в форматах 2t, 6, 7, 8d, 9d, определяется полями выбора типа регистра (sr, sc, sx) согласно таблице 8.15.

Таблица 8.15.

sr	sc	sx	Выбор подмножества регистров RC
Формат 2t			
1	x	x	Регистры данных (RF)
0	0	0	Регистры адресных генераторов AGU, AGU-Y
0	0	1	Регистры обменного буфера XBUF
0	1	0	Регистры управления PCU
0	1	1	Регистры-аккумуляторы ACi.[L/D/Q]
Формат 6			
x	0	x	Регистры адресных генераторов AGU, AGU-Y
x	1	x	Регистры управления PCU
Формат 7			
1	x	x	Регистры данных (RF)
0	0	x	Регистры адресных генераторов AGU, AGU-Y
0	1	x	Регистры управления PCU
Форматы 8d, 9d			
x	0	0	Регистры адресных генераторов AGU, AGU-Y
x	0	1	Регистры обменного буфера XBUF
x	1	0	Регистры управления PCU
x	1	1	Регистры-аккумуляторы ACi.[L/D/Q]

8.6.10 Поля формата пересылки (L, L2)

Поля формата пересылки (L, L2) определяют формат данных в операциях пересылок между регистрами данных и управления (форматы 5, 2t, 8b, 8d, 9b, 9d) либо непосредственных данных в регистр RF (формат 7) согласно таблице 8.16.

Таблица 8.16

L2	L	Разрядность пересылаемых данных
Формат 2t, пересылка между двумя регистрами данных		
0	0	16-разрядная пересылка
0	1	32-разрядная пересылка
1	0	64-разрядная пересылка
1	1	128-разрядная пересылка
Форматы 5, 8b, пересылка между двумя регистрами данных		

x	0	16-разрядная пересылка
x	1	32-разрядная пересылка
Формат 9b, пересылка между двумя регистрами данных		
x	0	64-разрядная пересылка
x	1	128-разрядная пересылка
Форматы 2t, 8d, 9d, пересылка между адресным регистром An, AT и регистром данных		
x	0	16-разрядная пересылка
x	1	32-разрядная пересылка
Формат 2t, пересылка между регистром-аккумулятором ACi.[L/D/Q], и регистром данных		
0	0	-
0	1	32-разрядная пересылка (ACi.L)
1	0	64-разрядная пересылка (ACi.D)
1	1	128-разрядная пересылка (ACi.Q)
Формат 8d, пересылка между регистром–аккумулятором ACi.[L/D], и регистром данных		
x	0	64-разрядная пересылка (ACi.D)
x	1	32-разрядная пересылка (ACi.L)
Формат 9d, пересылка между регистром-аккумулятором ACi.[D/Q], и регистром данных		
	0	64-разрядная пересылка (ACi.D)
	1	128-разрядная пересылка (ACi.Q)

8.6.11 Поле направления пересылки (de)

Поле de определяет направление перемещения данных в операциях пересылок между регистрами RF и памятью, либо между регистрами RF и регистрами управления. При de=1 производится запись в регистр RF, при de=0 – чтение из регистра RF.

8.6.12 Поле режима адресации AGU (mode) и поле кода блокировки обмена (u)

Поле mode определяет режим адресации памяти в операциях пересылок между регистрами RF и памятью данных под управлением адресного генератора AGU. Режимы адресации памяти приведены в таблице 8.17.

Код блокировки обмена u позволяет управлять выполнением пересылки независимо от режима адресации памяти. При u=0 пересылка выполняется, при u=1 модификация адреса An происходит в соответствии с режимом адресации без выполнения самой пересылки.

Таблица 8.17

mode	u	Мнемоника	Режим адресации, наличие обмена	Автокод (ea)
000	X	-	Отсутствие пересылки	nm
001	X	(An)	Косвенная без модификации	k
100	0	(An)+	Пост - автоинкремент	i
100	1	(An)+	Пост – автоинкремент, отмена пересылки	ui
101	0	(An)-	Пост - автодекремент	d
101	1	(An)-	Пост – автодекремент, отмена пересылки	ud

mode	u	Мнемоника	Режим адресации, наличие обмена	Автокод (ea)
110	0	(An)+In	Пост - автоувеличение	ai
110	1	(An)+In	Пост – автоувеличение, отмена пересылки	uai
111	0	(An)-In	Пост - автоуменьшение	ad
111	1	(An)-In	Пост – автоуменьшение, отмена пересылки	uad
011	X	(An+In)	Индексирование (An не меняется)	In
010	X	(An+dspl)	Адресация с непосредственным смещением (An не меняется)	Ds

8.6.13 Назначение поля “u” в форматах 9a, 9b

В форматах 9a, 9b могут выполняться 64-разрядные либо 128-разрядные обмены с памятью данных. Для кодировки разрядности обменов с памятью используется поле “u” (u=0 - 64-разрядные; u=1 - 128-разрядные обмены).

Таким образом, возможность блокировки обмена, для которой используется это поле в формате 8a, в формате 9a отсутствует.

8.6.14 Поле регистра адреса AGU (A)

Поле A определяет адресный регистр AGU, используемый для адресации памяти в операциях пересылок между регистрами RF и памятью данных. Кроме того, в формате 3mb адресный регистр определяет адрес программного перехода (см. таблицу 8.18).

Таблица 8.18

Поле A	Адресный регистр
000	A0
001	A1
010	A2
011	A3
100	A4
101	A5
110	A6
111	A7

8.6.15 Поле режима адресации AGU-Y (AT)

Поле AT совместно с управляющим параметром YM (11-й разряд регистра SR) определяет режим адресации Y-памяти в операциях пересылок между Y-памятью и регистрами RF, выполняемыми под управлением адресного генератора AGU-Y. Режимы адресации приведены в таблице 8.19.

Таблица 8.19

Поле AT	YM	Мнемоника	Режим адресации	Автокод (ea)
00	X	-	Отмена пересылки	uT
01	X	(AT)	Косвенная	AT
10	X	(AT)+IT	Пост - автоувеличение	ATp
11	0	(AT+IT)	Индексирование (AT не меняется)	ATIT
11	1	(AT)+DT	Пост - автоувеличение	ATp

8.6.16 Поля непосредственных операндов (#5, #16, #32) и поле выбора непосредственного пятиразрядного операнда (#)

8.6.16.1 В вычислительных операциях, также как в операциях пересылок и управления, могут использоваться непосредственные операнды. Короткие пятиразрядные непосредственные операнды #5 могут использоваться в операциях ASL, ASLL, ASLX, ASR, ASRL, ASRX, BTST, BTSTL, LSL, LSLL, LSLX, LSR, LSRL, LSRX, MSK, MSKL вместо операнда S1 (S). Выбор непосредственного операнда #5 для этих операций контролируется полем # (при #=0 в операции участвует регистровый операнд, при #=1 - непосредственный пятиразрядный операнд). Поле #, кроме того, используется для расширения поля кода операции (см. 8.3.3).

Для непосредственных операндов #16, #32 используется либо специальное поле (в форматах 2d, 3, 3m разряды [26:11] используются для 16-разрядного непосредственного операнда #16) либо дополнительное (второе) слово инструкции.

8.6.17 Поле выбора режима Scaling (M)

Поле M в инструкциях форматов 8a, 8b, 8c, 8d позволяет установить (M=1) или отключить (M=0) режим Scaling в арифметических операциях, использующих этот режим.

Конкатенация поля M с полем OP2 - { M, OP2} в инструкциях форматов 9a, 9b, 9d определяет код операции OP2e согласно таблице 8.8.

9. СИНТАКСИС ИНСТРУКЦИЙ

9.1 Общие положения

9.1.1 Каждая инструкция может содержать до двух вычислительных операций и до двух операций пересылок.

Синтаксически инструкция записывается в одну строку, в которой поля вычислительных операций и пересылок отделены друг от друга некоторым количеством пробелов или табуляций. Каждая новая инструкция начинается с новой строки.

Операции, составляющие инструкцию базовой системы инструкций, записываются в следующем порядке:

<Операция `OP2`> < Операция `OP1`> <Пересылка `1`> <Пересылка `2`>

Операции, составляющие инструкцию расширения системы инструкций, записываются в следующем порядке:

<Операция `OP2e`> < Операция `OP1e`> <Пересылка `1`> <Пересылка `2`>

Система инструкций DSP-ядра Elcore-30M является *ортогональной* в том смысле, что каждая операция, входящая в состав инструкции, кодируется независимо от других в соответствующих полях инструкции. Тем самым, имея описание способов кодировки для каждой из операций, можно однозначно определить код всей инструкции при любом сочетании составляющих ее операций.

В настоящем разделе приводится описание операций DSP-ядра Elcore-30M. В описание каждой операции (команды) включено:

- мнемоническое имя;
- содержание исполняемой операции;
- словесное описание операции с указанием специальных случаев и вырабатываемых признаков;
- тип операции и разрешенные форматы;
- синтаксис написания команды на ассемблере и кодировка соответствующих полей инструкции;
- автокод (структура инструкции) – мнемоническая запись, позволяющая однозначно установить формат и код инструкции (формируется только для базовой системы инструкций).

9.2 Соглашения по обозначениям

9.2.1 Для описания содержания исполняемой операции применяется условный язык, базирующийся, в основном, на средствах языка C. Используются также математические символы | | и √ для обозначения абсолютного значения и квадратного корня числа.

Оператор присваивания выражается стрелкой → . Присваивание выполняется слева направо. Например, содержание операции ADC (сложение с переносом) выражается формулой `s1 + s2 + C → d`, которая означает, что вычисляется сумма операндов-источников s1,s2 и входного переноса C и результат помещается в операнд-приемник d.

Кроме того, для описания манипуляций с отдельными битами и битовыми полями применяются конструкции языка `Verilog`, в частности:

- рассматриваемые разряды операнда указываются в квадратных скобках, через двоеточие, причем слева указывается старший разряд. *Пример* - D[30:23];

- фигурные скобки { } означают конкатенацию битовых полей, старшие разряды стоят слева.

При описании синтаксиса инструкций используются следующие условные обозначения:

- < > - в фигурные скобки заключаются операции (вычислительные или пересылки, для которых указывается лишь тип, без раскрытия операндов и параметров;
- [ ] - в квадратные скобки заключаются части инструкции, которые могут быть опущены. Это касается, например, команды MOVE, выполняющейся параллельно с вычислительной операцией – [M]; модификаторов кода условия [.cc]; включения режима Scaling [s] в арифметических командах 8-го формата.

`XRAM, YRAM` – ссылки на X- или Y- память данных;
`S, S1, S2, D` – 16- или 32-разрядный регистр данных (в форматах 1-8);
`T, S, D` – 16/ 32/64/128-разрядный регистр данных (в форматах 9);
`s, d, R` – 16-разрядный регистр данных;
`R.L` – 32-разрядный регистр данных;
`#x` – x-разрядные непосредственные данные;
`RC` – управляющий регистр;
`cc` – код условия;
`An` – адресный регистр, n=0,1,…,7.

В таблицах признаков результата способ их формирования отмечается следующим образом:

- “√” - стандартный способ формирования;
- “*” - нестандартный способ формирования, оговаривается особо в самой таблице или примечании;
- “–” - признак не формируется.

Изм	Номера листов (страниц)				Всего листов (страниц) в документе	N документа	Подпись	Дата
	изме-нённых	заменённых	новых	аннулиро-ванных				