Спецификация архитектурных отличий 2-ядерного DSP-кластера DELcore-30M в составе процессора 1892ВМ10Я

СОДЕРЖАНИЕ

1. СВОДНАЯ ТАБЛИЦА ОСНОВНЫХ АРХИТЕКТУРНЫХ ХАРАКТЕРИСТИК DSP-ЯДЕР ПРОЦЕССОРОВ «МУЛЬТИКОР» 4

2. АРХИТЕКТУРА 2-ЯДЕРНОГО DSP-КЛАСТЕРА DELCORE-30M 5

2.1 Карта памяти DSP-кластера 5
2.1.1 Дисциплина отработки одновременных обращений к общему полю памяти данных со стороны DSP-ядер (арбитраж). *6*
2.2 Регистры управления и состояния DELcore-30M 6
2.2.1 Регистр маски прерываний (MASKR_DSP) *7*
2.2.2 Регистр запросов прерываний (QSTR_DSP) *7*
2.2.3 Регистр управления и состояния (CSR_DSP) *7*
2.3 Буфер обмена XBUF 8
2.3.1 Регистр флагов обмена (EFR) *8*
2.3.2 Режимы обменов с XBUF *8*
2.4 Синхронизация кластера DELcore-30M 8

3. АРХИТЕКТУРА DSP-ЯДРА ELCORE-30M 9

3.1 Регистровый файл 9
3.2 Регистры-аккумуляторы 10
3.3 Адресные регистры A0-A7, AT 10
3.4 Регистр адреса вектора прерывания IVAR 11
3.5 Отладочные регистры 11
3.5.1 Регистр dbDCSR *12*
3.5.2 Регистры dbSAR, dbSAR1-dbSAR7 ..... *12*
3.5.3 Регистр dbCNTR *13*
3.5.4 Регистр Cnt_RUN *13*
3.6 Назначение разрядов регистров DCSR, SR 13
3.6.1 Назначение разрядов регистра DCSR *13*
3.6.2 Дополнительные биты управления в регистре SR *13*
3.7 Регистры управления прерываниями и DMA-обменами 14
3.7.1 Механизм отработки прерываний *14*
3.7.2 Регистр запросов на прерывание DSP (IRQR) *14*
3.7.3 Регистр маски запросов на прерывание DSP (IMASKR) *15*
3.7.4 Регистр запуска DMA со стороны DSP (DSTART) *15*
3.8 Регистр таймера (TMR) 15
3.9 Регистр управления локальным арбитром (ARBR) 16
3.9.1 Принцип арбитража и режимы работы *16*
3.9.2 Назначение разрядов регистра ARBR *17*
3.9.3 Механизм передачи приоритета *17*
3.10 Регистр спецфункций (SFR) 17

4. ПРОГРАММНЫЙ КОНВЕЙЕР DSP-ЯДРА ELCORE-30M 17

4.1 Время исполнения вычислительных операций 20
4.2 Ограничения конвейера 20

5. СИСТЕМА ИНСТРУКЦИЙ 21

5.1 Форматы данных 22
5.2 Форматы пересылок 22
5.3 Модификация адреса при 64- и 128-разрядных обменах 22
5.4 Команда возврата из прерывания RTI 23
5.5 Дополнительный набор вычислительных команд 23
5.6 Изменение форматов результатов некоторых вычислительных команд 23
5.7 Рекомендации по переносу DSP-программ с процессоров 1892ВМ3Т,1892ВМ2Я, 1892ВМ4Я, 1892ВМ5Я на процессор 1892ВМ10Я 23
5.8 Форматы инструкций и коды операций 24
5.8.1 Форматы 9a, 9b и 9d *24*
5.8.2 Назначение поля “u” в форматах 9a, 9b *24*
5.8.3 Назначение полей “sc”, “sx” в форматах 2t, 8d, 9d *25*
5.8.4 Кодирование адресов регистров *25*

5.8.5 Назначение полей “L”, “L2” в форматах 2t, 8d, 9b, 9d 27
5.8.6 Коды операций в форматах 9a, 9b и 9d 27
5.8.7 Кодирование операции RTI 27

6. ПЕРЕЧЕНЬ АДРЕСУЕМЫХ РЕГИСТРОВ DSP-КЛАСТЕРА 31

1. Сводная таблица основных архитектурных характеристик DSP-ядер процессоров «Мультикор»

В таблице 1 приведены основные архитектурные характеристики DSP-ядер в составе процессоров «Мультикор»

Таблица 1. Основные архитектурные характеристики DSP-ядер в составе процессоров «Мультикор»

N	Процессор	Число DSP	Тип DSP	Число SIMD секций в DSP	Число фаз кон-вейера	Значение IDR	Память программ (PRAM)	Память данных (XRAM,YRAM)	Система инструкций	Состав программно-доступных регистров
1	**1892ВМ3Т (МС-12)**	1	**ELcore-14**	1	3	0x0003	PRAM 4Kx32	XRAM 24Kx32 YRAM 12Kx32	DSP-ядро **ELcore-x4** СИСТЕМА ИНСТРУКЦИЙ. РАЯЖ. 431280.003Д2	РАЯЖ. 431280.003Д2
2	**1892ВМ2Я (МС-24)**	1	**ELcore-24**	2	3	0x0013	PRAM 4Kx32	XRAM 32Kx32 YRAM 8Kx32	DSP-ядро **ELcore-x4** СИСТЕМА ИНСТРУКЦИЙ. РАЯЖ. 431280.003Д2	РАЯЖ. 431280.003Д2
3	**1892ВМ4Я** **1892ВМ5Я** **(МС-0226 МС-0226G)**	2 **(DELcore-26)**	**ELcore-26**	2	4	0x0115	**DSP0** PRAM 4Kx32 **DSP1** PRAM 4Kx32	**DSP0** XRAM 8Kx32 YRAM 8Kx32 **DSP1** XRAM 8Kx32 YRAM 8Kx32	Та же, что в **ELcore-x4**	Тот же, что в **ELcore-x4**, плюс SAR1-SAR7, DMAR, EFR плюс XBUF 32x32
4	**МС-0428** **(MCom01) (MForce)**	4 **(QELcore-28)**	**ELcore-28**	1	7	0xn309*	**DSP0** PRAM 8Kx32 **DSP1** PRAM 8Kx32 **DSP2** PRAM 8Kx32 **DSP3** PRAM 8Kx32	**DSP0** XYRAM 32Kx32 **DSP1** XYRAM 32Kx32 **DSP2** XYRAM 32Kx32 **DSP3** XYRAM 32Kx32	Та же, что в **ELcore-x4** с дополнениями и уточнениями, приведенными в документе whatsnew_MForce_v92.doc	Тот же, что в **ELcore-x4** с дополнениями и уточнениями, приведенными в документе whatsnew_MForce_v92.doc
5	**1892ВМ10Я (NVCom-02T)**	2 **(DELcore-30M)**	**ELcore-30M**	1	7	0xn108*	**DSP0** PRAM 8Kx32 **DSP1** PRAM 8Kx32	**DSP0** XYRAM 32Kx32 **DSP1** XYRAM 32Kx32	Та же, что в **ELcore-x4** с дополнениями и уточнениями, приведенными в настоящем документе	Тот же, что в **ELcore-x4** с дополнениями и уточнениями, приведенными в настоящем документе

* n – номер DSP-ядра

2. Архитектура 2-ядерного DSP-кластера DELcore-30M

1) В состав процессора входит 2-ядерный DSP-кластер DELcore-30M - мультипроцессор, состоящий из 2-х DSP-ядер ELcore-30M - DSP0 и DSP1, работающих на общем поле памяти данных, имеющих набор общих регистров управления/состояния, а также буфера обмена XBUF.

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

2.1 Карта памяти DSP-кластера

Карта памяти DSP-кластера в составе процессора 1892ВМ10Я приведена на рисунке 1. Каждое из DSP-ядер имеет свою программную память (PRAM) объемом 32 Кбайт и общую для всех память данных XYRAM объемом 256 Кбайт.

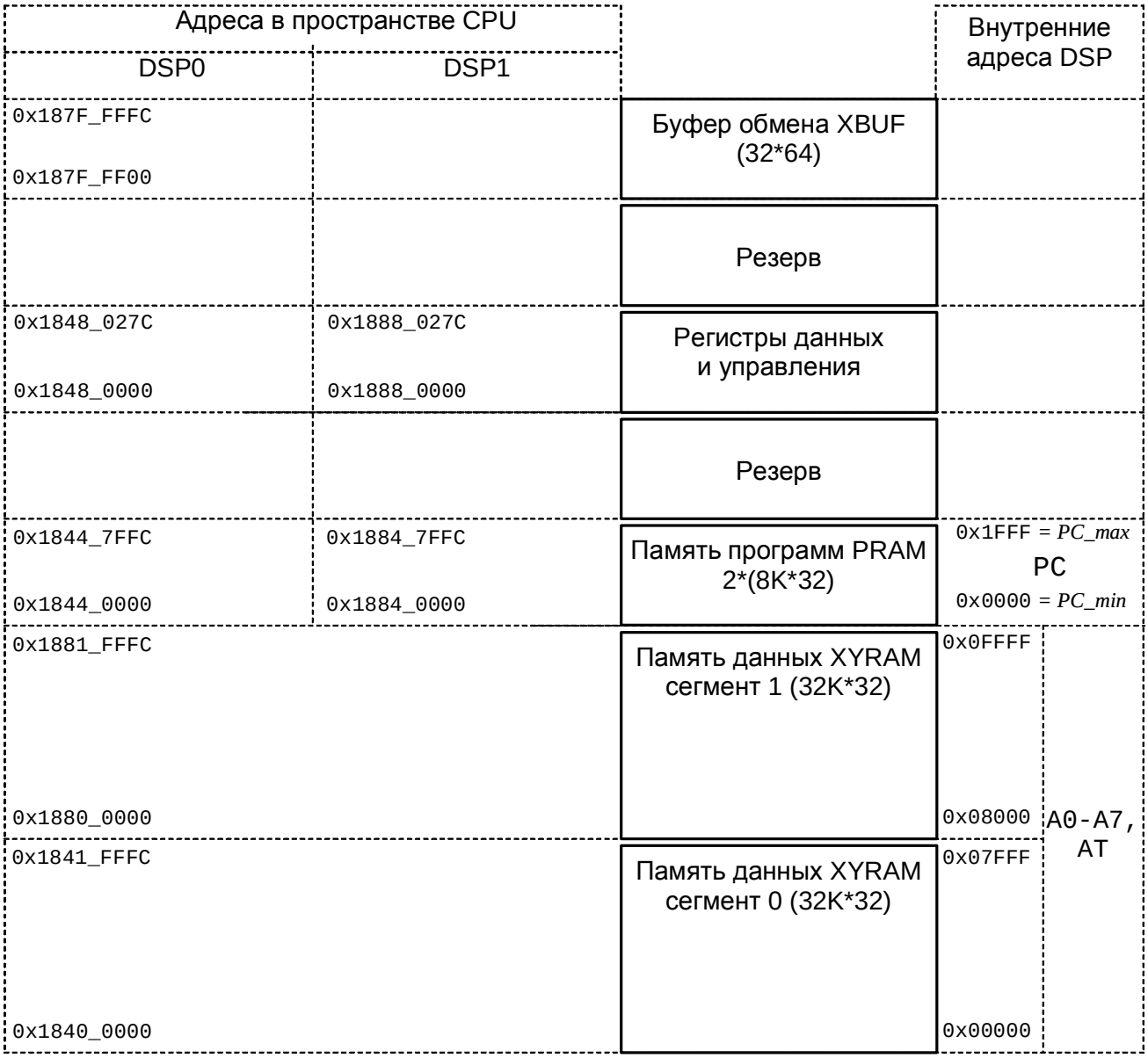


Рис. 1 Карта памяти DSP0-DSP1 в составе процессора 1892ВМ10Я

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).

Для обеспечения возможности одновременного доступа к памяти программ и данных DSP как со стороны CPU (DMA), так и со стороны DSP блоки памяти XYRAM и PRAM аппаратно реализованы как 2-портовые. С внешней стороны возможны как 32-разрядные (CPU), так и 64-разрядные обращения (DMA). Со стороны DSP0–DSP1 возможны 32/64/128-разрядные обращения (чтение и запись) к памяти данных XYRAM. Программная память PRAM со стороны DSP доступна только для чтения 32/64-разрядных слов иструкций.

Два входящих в состав процессора 1892ВМ10Я DSP-ядра работают на общем поле памяти данных XYRAM. Для каждого DSP-ядра сегмент памяти с соответствующим номером является «ближней» памятью, доступ к которой осуществляется с наименьшей задержкой. Доступ к остальной («дальней») памяти производится с дополнительной задержкой, необходимой для чтения данных из дальней памяти.

Указатели A0-A7 адресного генератора AGU и указатель AT адресного генератора AGU-Y полностью равноправны, т.е. по указателям A0-A7, AT каждому из DSP-ядер доступна вся память данных XYRAM.

Начальное состояние регистров A0-A7, AT каждого из DSP-ядер приведено в таблице 2.

Таблица 2. Начальное состояние регистров A0-A7, AT

Условное обознач.	Разрядность	Наименование	Начальное состояние	
			DSP0	DSP1
A0-A7	32 R/W	Адресный регистр AGU	0x00000	0x8000
AT	32 R/W	Адресный регистр AGU-Y	0x04000	0xC000

2.1.1 Дисциплина отработки одновременных обращений к общему полю памяти данных со стороны DSP-ядер (арбитраж).

Так как память данных XYRAM является общим ресурсом для обоих DSP-ядер, при одновременном обращении к ней со стороны нескольких DSP-ядер возможны коллизии.

Для уменьшения числа таких коллизий память данных XYRAM разделена на 2 сегмента, каждый из которых содержит 4 страницы объемом 8К 32-разрядных слов. Аппаратно каждая страница реализована в виде четырех блоков памяти по 2К*32 бит каждый.

Таким образом, обращения от различных DSP-ядер к различным страницам памяти могут происходить одновременно и не приводить к коллизиям (конфликтам) и задержкам. Кроме того, возможны два одновременных обращения по X и Y указателям от одного DSP-ядра к одной странице памити, при условии, что обращения идут к разным блокам памяти.

Коллизии возникают лишь при одновременном обращении нескольких DSP-ядер к одной и той же странице, либо при одновременном обращении X-указателя (A0-A7) и Y-указателя (AT) одного из DSP-ядер к одному физическому блоку памяти.

Для разрешения возникающих конфликтов вводится дополнительное устройство – арбитр памяти. Процедура арбитража позволяет корректно отработать все обращения, однако, в случае конфликтов между устройствами, приводит к некоторому замедлению работы программы из-за введения дополнительных тактов ожидания обмена.

Подробнее дисциплина отработки одновременных обращений к одной и той же странице памяти данных со стороны нескольких DSP-ядер (арбитраж) рассматривается в п.3.9.

2.2 Регистры управления и состояния DELcore-30M

На верхнем уровне кластера DSP имеются 3 регистра управления и состояния, доступ к которым осуществляется только со стороны CPU. Назначение и адреса этих регистров указаны в таблице 3.

Таблица 3 Назначение и адреса регистров управления и состояния кластера DSP

Имя	Разрядность	Тип обращений	Назначение	Адрес
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

2.2.1 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.

2.2.2 Регистр запросов прерываний (QSTR_DSP)

Регистр запросов прерываний QSTR_DSP доступен только по чтению и содержит флаги запросов прерываний от 2-х DSP-ядер. Назначение разрядов регистра QSTR_DSP приведено в таблице 4. Для сброса флагов прерывания необходимо обнулить соответствующие разряды регистров DCSR, входящих в состав DSP0-DSP1.

Таблица 4 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.

2.2.3 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 5.

Таблица 5 Назначение разрядов регистра CSR_DSP

Номер разряда	Наименование разряда	Назначение
0	SYNSTART	Одновременный старт DSP0 – DSP1
1	SYNWORK	Работа XBUF в синхронном режиме
2-31	-	Резерв

Начальное состояние регистра CSR_DSP=0x0.

Запись «1» в разряд SYNSTART приводит к одновременному запуску 2-х DSP–ядер. При этом в регистрах DCSR каждого из DSP–ядер бит RUN устанавливается в «1», состояние других разрядов не изменяется.

2.3 Буфер обмена XBUF

Для оперативных обменов данными между CPU, DSP0 – DSP1 в составе процессоров 1892ВМ10Я 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.

Особенностью работы XBUF в составе процессора 1892ВМ10Я является то, что обмены со стороны DSP0 – DSP1 – 64-разрядные, а со стороны CPU – 32-разрядные. Размещение 64-разрядных регистров X0-X31 в адресном пространстве CPU приведено в таблице 14.

2.3.1 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 32-разрядных обращениях со стороны CPU изменение состояния EFR происходит только при обращении к младшей половине 64-разрядной ячейки XBUF**.

2.3.2 Режимы обменов с XBUF

Имеются два режима обменов с XBUF – обычный и синхронный (семафорный).

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

В регистре DCSR имеется бит WT=DCSR[4], указывающий на то, что DSP находится в состоянии ожидания при обращении к XBUF. Одновременная установка битов WT в состояние «1» во всех четырех DSP-ядрах (то есть зависание программы) вызывает прерывание WAIT в CPU (разряд 28 регистра QSTR_DSP).

2.4 Синхронизация кластера DELcore-30M

Имеются три частотных домена синхронизации кластера DELcore-30M:

1) домен синхронизации CPU, к котророму относятся регистры верхнего уровня: MASKR_DSP, QSTR_DSP, CSR_DSP;

2) домен синхронизации DSP0, к котророму относятся регистры и память DSP0, а также буфер обмена XBUF;

3) домен синхронизации DSP1, к котророму относятся регистры и память DSP1.

С целью уменьшения энергопотребления на уровне CPU при помощи соответствующего регистра управления может быть включена/отключена частота синхронизации DSP0 и DSP1.

При отключенной частоте регистры соответствующего DSP-ядра становятся недоступны. При этом должен соблюдаться следующий порядок: включение частоты начинается с DSP0, а отключение - с DSP1. Частота CPU не отключается и соответственно регистры верхнего уровня доступны всегда.

3. Архитектура DSP-ядра ELcore-30M

По сравнению с базовым для платформы «Мультикор» вариантом DSP-ядра Elcore-14 в DSP-ядро ELcore-30M внесены следующие основные архитектурные изменнения:

- увеличен объём и изменен формат регистрового файла RF;

- увеличено количество и изменен формат регистров-аккумуляторов;

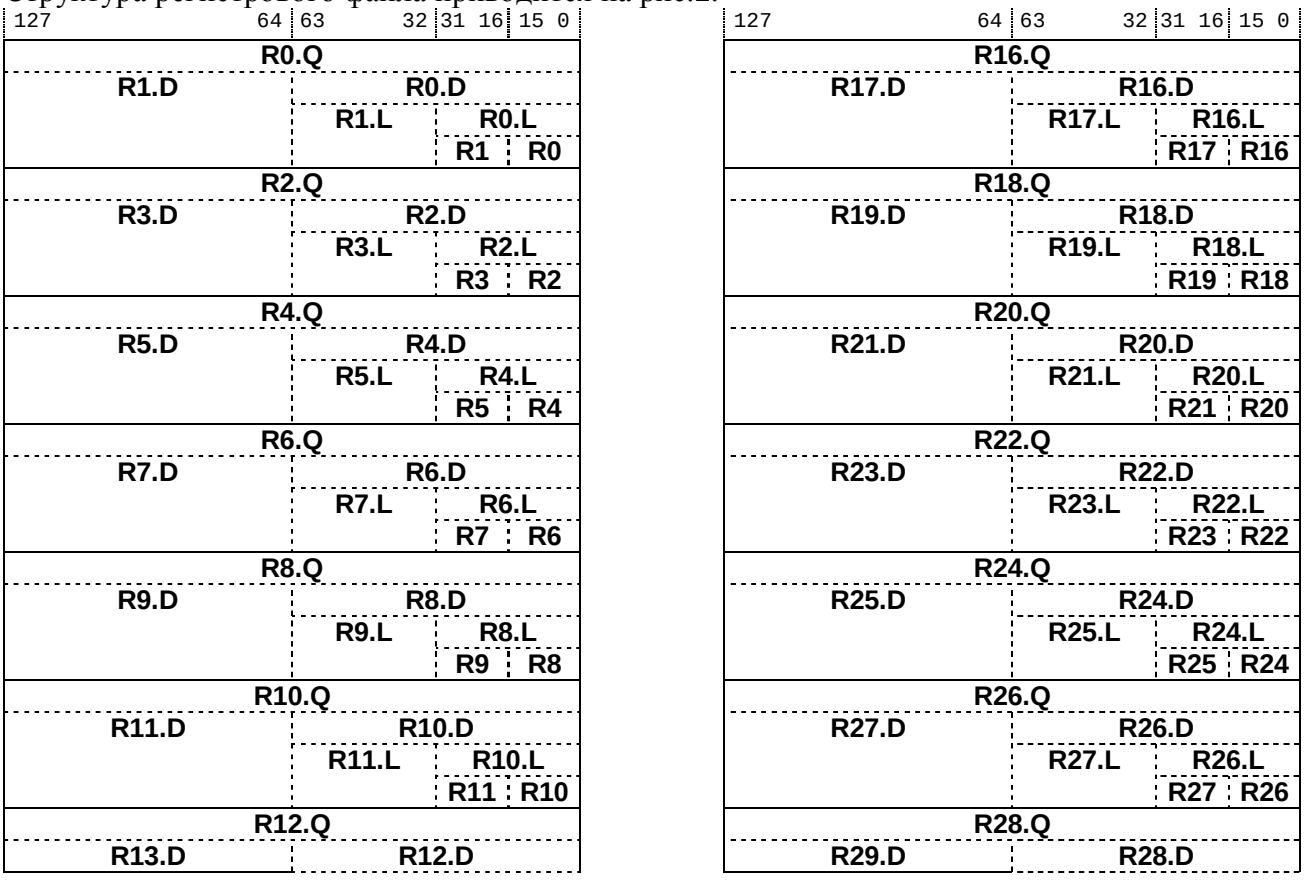
- изменен формат адресных регистров A0 – A7, AT;

- изменен состав и назначение управляющих и отладочных регистров.

3.1 Регистровый файл

В ELcore-30M регистровый файл (RF) используется для хранения и оперативной обработки операндов, имеющих формат 16, 32, 64 или 128 бит.

Структура регистрового файла приводится на рис.2.



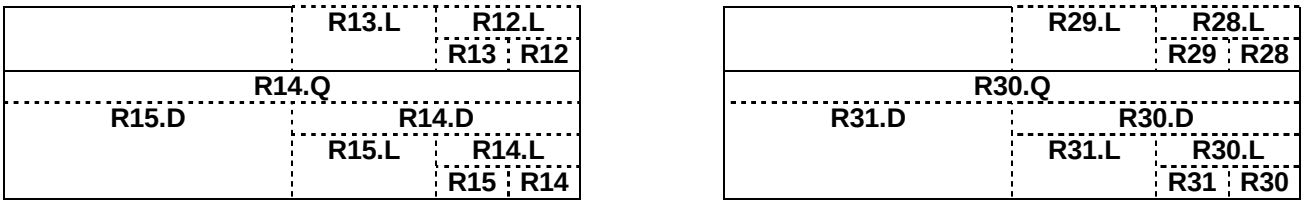


Рис. 2 Структура регистрового файла ELcore-30M.

Для определения форматов регистров вводятся следующие мнемоники:
R – 16-разрядные регистры;
R.L – 32-разрядные регистры;
R.D – 64-разрядные регистры;
R.Q – 128-разрядные регистры.

Адреса регистров RF в адресном пространстве CPU приведены в таблице 14.

3.2 Регистры-аккумуляторы

Регистры-аккумуляторы предназначены для хранения данных, получаемых в результате выполнения операций умножения с накоплением (см. Приложение 1). Начальное состояние регистров-аккумуляторов равно нулю.

Каждое 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.

Структура регистрового файла регистров-аккумуляторов приводится на рис.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.

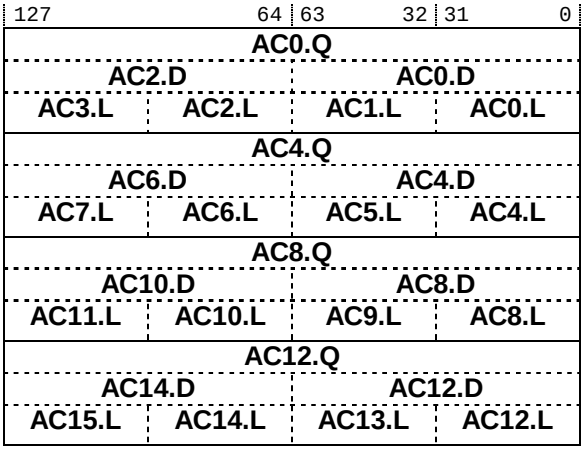


Рис. 3 Структура регистрового файла регистров-аккумуляторов ELcore-30M.

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

Адреса регистров-аккумуляторов в адресном пространстве CPU приведены в таблице 14.

3.3 Адресные регистры A0-A7, AT

В ELcore-30M расширен формат адресных регистров A0 – A7, AT до 32 разрядов. Это вызвано расширением адресного пространства DSP 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 являются 16-разрядными.

Важной особенностью адресных регистров является то, что операции инкремента и декремента выполняются в 16-разрядном формате. Таким образом, путем опреаций инкремента и декремента невозможен переход из адресного пространства одного сегмента в адресное пространство другого сегмента. Для перехода в аджресное пространство другого сегмента памяти необходима явная запись 32-разрядного адреса в нужный адресный регистр.

Адреса регистров адресных генераторов A0–A7,AT,I0–I7,IT,DT,M0–M7,MT в пространстве CPU указаны в таблице 14.

Начальное состояние регистров A0-A7, AT приведено в таблице 2.

3.4 Регистр адреса вектора прерывания IVAR

В ELcore-30M реализован механизм прерываний, рассмотренный подробнее в п.3.7. 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).

Адреса регистров IVAR для DSP0-DSP1 в пространстве CPU указаны в таблице 14.

Начальное состояние регистра IVAR=0x1F00.

3.5 Отладочные регистры

По сравнению с базовым для платформы «Мультикор» вариантом DSP-ядра Elcore-14 в ELcore-30M вводятся дополнительные специализированные отладочные регистры и изменяется назначение связанных с отладкой бит в регистре управления DCSR. Состав и адреса специализированных отладочных регистров приведены в таблице 6. Указанные регистры предназначены только для поддержки режима отладки. Их мнемонические имена не поддерживаются ассемблером DSP-ядра ELcore-30M. С введением данных регистров существующие регистры DCSR, SAR, CNTR, SAR1-SAR7 освобождаются от отладочных функций и могут использоваться только самой прикладной программой.

Регистры стадий программного счетчика dbPCx доступны только по чтению.

Таблица 6. Специализированные отладочные регистры ELcore-30M

Условное обознач.	Разрядность	Наименование	Адрес регистра (DSP0)	Адрес регистра (DSP1)
dbDCSR	16 R/W	Регистр управления в режиме отладки	0x1848_0500	0x1888_0500
Cnt_RUN	32 R	Счетчик тактов	0x1848_0518	0x1888_0518
dbPCe	16 R	Программный счетчик, стадия e	0x1848_0520	0x1888_0520
dbPCa	16 R	Программный счетчик, стадия a	0x1848_0524	0x1888_0524
dbPCf	16 R	Программный счетчик, стадия f	0x1848_0528	0x1888_0528
dbPCd	16 R	Программный счетчик, стадия d	0x1848_052C	0x1888_052C
dbPCe1	16 R	Программный счетчик, стадия e1	0x1848_0530	0x1888_0530
dbPCe2	16 R	Программный счетчик, стадия e2	0x1848_0534	0x1888_0534
dbPCe3	16 R	Программный счетчик, стадия e3	0x1848_0538	0x1888_0538
dbSAR	16	Регистр адреса останова 0	0x1848_053C	0x1888_053C

	R/W	в режиме отладки		
dbCNTR	16 R/W	Счетчик исполненных команд в режиме отладки	0x1848_0540	0x1888_0540
dbSAR1	16 R/W	Регистр адреса останова 1 в режиме отладки	0x1848_0544	0x1888_0544
dbSAR2	16 R/W	Регистр адреса останова 2 в режиме отладки	0x1848_0548	0x1888_0548
dbSAR3	16 R/W	Регистр адреса останова 3 в режиме отладки	0x1848_054C	0x1888_054C
dbSAR4	16 R/W	Регистр адреса останова 4 в режиме отладки	0x1848_0550	0x1888_0550
dbSAR5	16 R/W	Регистр адреса останова 5 в режиме отладки	0x1848_0554	0x1888_0554
dbSAR6	16 R/W	Регистр адреса останова 6 в режиме отладки	0x1848_0558	0x1888_0558
dbSAR7	16 R/W	Регистр адреса останова 7 в режиме отладки	0x1848_055C	0x1888_055C

3.5.1 Регистр dbDCSR

Формат отладочного регистра dbDCSR указан ниже.

dbDCSR:

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

dbRUN – состояние исполнения программы в режиме отладки;
dbBRK – флаг останов исполнения программы в режиме отладки.

Начальное состояние dbDCSR = 0x0.

Назначение бита dbRUN регистра dbDCSR в режиме отладки аналогично назначению бита DBG регистра DCSR в предыдущих модификациях DSP-ядер Elcore-xx.

Наличие этого бита позволяет производить автономную отладку DSP-ядра при остановленном контроллере (в том числе CPU). Установка бита dbRUN в «1» переводит DSP-ядро в состояние исполнения программы в режиме отладки, установка в «0» - в состояние останова. Бит dbRUN автоматически сбрасывается по останову dbBRK.

Флаг dbBRK (флаг останова исполнения программы в режиме отладки) устанавливается в «1» в случае останова DSP по одной из следующих причин:

1) по достижении адреса останова, содержащегося в одном из отладочных регистров dbSAR, dbSAR1-dbSAR7;

2) по завершении требуемого числа шагов, содержащегося в отладочном регистре dbCNTR.

Примечание. В случае останова по достижении адреса, содержащегося в одном из штатных регистров SAR, SAR1-SAR7 либо по завершении требуемого числа шагов, содержащегося в штатном регистре CNTR, флаг dbBRK в «1» не устанавливается.

3.5.2 Регистры dbSAR, dbSAR1-dbSAR7

Назначение регистров dbSAR, dbSAR1-dbSAR7 в режиме отладки аналогично назначению штатных регистров SAR, SAR1-SAR7 в режиме штатного исполнения программы.

Регистры dbSAR, dbSAR1-dbSAR7 определяют точки останова в режиме отладки. Перед исполнением инструкции с указанным адресом DSP-ядро переходит в состояние останова (dbRUN=0) и флаг dbBRK устанавливается в «1».

Начальное состояние dbSAR, dbSAR1-dbSAR7 равно 0xFFFF.

3.5.3 Регистр dbCNTR

Регистр dbCNTR задает пошаговый режим исполнения программ в режиме отладки аналогично тому, как регистр CNTR делает это в режиме штатного исполнения.

Начальное состояние dbCNTR = 0x0.

3.5.4 Регистр Cnt_RUN

Регистр Cnt_RUN представляет собой счетчик тактов, затраченных на исполнение программы начиная с момента последнего запуска DSP. Доступен только по чтению.

Начальное состояние Cnt_RUN = 0x0.

3.6 Назначение разрядов регистров DCSR, SR

3.6.1 Назначение разрядов регистра DCSR

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

Формат регистра DCSR ELcore-30M указан ниже.

DCSR:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	RUN	-	-	-	-	-	-	-	-	-	WAIT	STP	BRK	SE	PI

RUN - состояние исполнения программы; WAIT – состояние ожидания при обращениях к XBUF;
STP – прерывание по останову STOP;
BRK –прерывание по останову BREAK;
SE – прерывание по ошибке стека SE;
PI – программное прерывание PI.

Начальное состояние DCSR = 0x0000.

Назначение разрядов RUN, WAIT, STP, SE, PI регистра DCSR Elcore-28 сохраняется таким же, как в предыдущих модификациях DSP-ядер Elcore-xx.

Бит RUN, как и в предыдущих модификациях DSP-ядер, автоматически сбрасывается по останову STOP или BREAK.

Флаг прерывания BRK устанавливается в «1» в случае останова DSP по одной из следующих причин:

1) по достижении адреса останова, содержащегося в одном из регистров SAR, SAR1-SAR7;

2) по завершении требуемого числа шагов, содержащегося в регистре CNTR.

Примечание. В случае останова по достижении адреса, содержащегося в одном из отладочных регистров dbSAR, dbSAR1-dbSAR7 либо по завершении требуемого числа шагов, содержащегося в регистре dbCNTR, флаг BRK в «1» не устанавливается.

3.6.2 Дополнительные биты управления в регистре SR

В регистре SR водятся дополнительные биты управления режимами работы DSP:

DD (Double Destination) = SR[9] и

BD (Blocking Disabled) = SR[10].

3.6.2.1 Бит DD

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 прежнем варианте (с двумя результатами и двумя адресами записи), при DD=1 один результат удвоенного формата записывается по одному адресу D.L(D.D).

Более подробно назначение этого бита рассматривается в п.5.6.

3.6.2.2 Бит BD

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.

3.7 Регистры управления прерываниями и DMA-обменами

В DSP ELcore-30M вводится 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:

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

- вводится регистр маски запросов на прерывание DSP – **IMASKR**;

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

Адреса указанных регистров приведены в таблице 14.

3.7.1 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.

3.7.2 Регистр запросов на прерывание DSP (IRQR)

Регистр IRQR содержит флаги запросов («1» - наличие запроса, «0» - отсутствие запроса) на прерывание DSP со стороны DMA, CPU, других DSP-ядер, таймера. Назначение разрядов регистра IRQR приведено в таблице 7.

Регистр IRQR доступен по записи и чтению со стороны CPU, 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 7. 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	IRQ4	Запрос на прерывание DSP со стороны таймера TMR
29	IRQ5	Запрос на прерывание DSP со стороны CPU
30	IRQ6	Запрос на прерывание DSP со стороны CPU
31	IRQ7	Запрос на прерывание DSP со стороны CPU

Начальное состояние регистра IRQR =0x0.

3.7.3 Регистр маски запросов на прерывание DSP (IMASKR)

Регистр DMASKR содержит 32 разряда, каждый из которых разрешает («1») либо запрещает («0») запрос на прерывание DSP от соответствующего разряда регистра запросов прерываний IRQR. Регистр доступен по чтению и записи со стороны CPU или DSP. Начальное состояние регистра IMASKR=0x0.

3.7.4 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. 8.

Таблица 8. 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

3.8 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.

3.9 Регистр управления локальным арбитром (ARBR)

3.9.1 Принцип 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 2 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 4-х физических блоков по 2К 32 разрядных слов каждый. Для организации чтения 128 разрядных слов, а так же для повышения производитетльности при 32-х разрядных обменах с памятью применена технология расслоения памяти. Т.е. любые 4 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 задании 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.

3.9.2 Назначение разрядов регистра ARBR

Регистр управления локальным арбитром.

	Limit	резерв	DPTR	резерв	LEN	DEN	HEN
Биты	13:8		5:4		2	1	0
Reset	0x0F		0		0	0	1

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.

3.9.3 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.

3.10 Регистр спецфункций (SFR)

Регистр спецфункций SFR (32 разряда, запись/чтение) предназначен для реализации специальных вычислительных функций, таких, как декодер Витерби, медианная фильтрация, сортировка и др. Назначение разрядов регистра SFR определяется реализуемой функцией.

4. Программный конвейер DSP-ядра ELcore-30M

Программный конвейер DSP-ядра Elcore-28 содержит 7 фаз.

Отличие его от программного конвейера DSP-ядра Elcore-18 состоит в том, что исполнение вычислительных команд выполняется не за три, а за два такта (на 6-й и 7-й фазе конвейера).

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	Декодиро-вание инструкции	Модифика-ция 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	Декодиро-вание инструкции	Формиро-вание 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	Декодиро-вание инструкции	Выборка данных из RF	Запись данных в RC	-	-

5) 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 на PRAM	Чтение инструкции из PRAM	Декодиро-вание инструкции	Выборка адреса перехода#A	Выдача адреса #A на PRAM	Чтение инструкции по адресу#A	Декодир инструкции по адресу#A.

Таким, образом, при исполнении различных операций фазы конвейера DSP-ядра ELcore-28 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

Временные диаграммы работы программного конвейера при исполнении команд программных переходов B(J), BD(JD) #A приведены на рис.3,4.

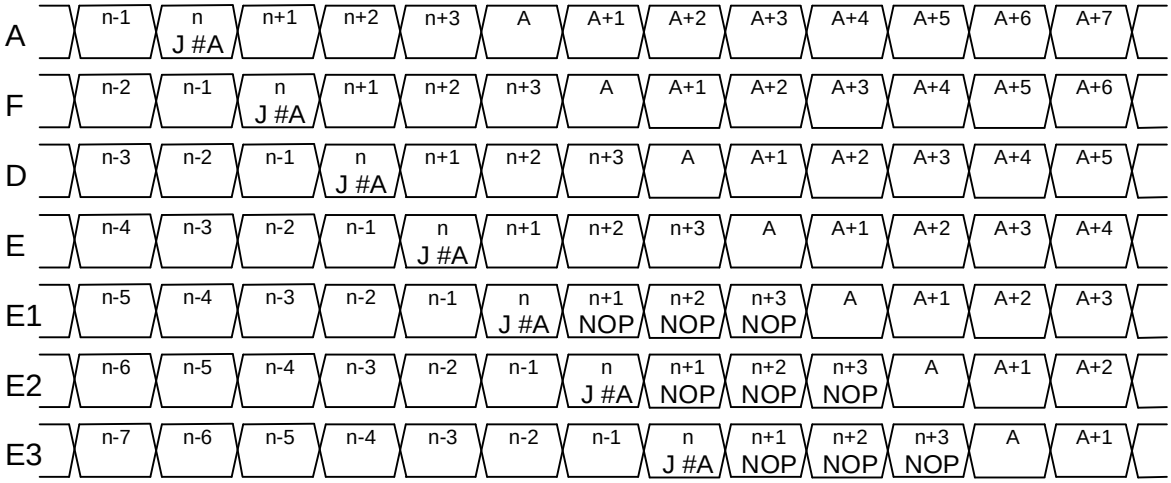


Рис.3. Временная диаграмма работы 7-фазного программного конвейера DSP-ядра при выполнении инструкции программного перехода B(J), #A.

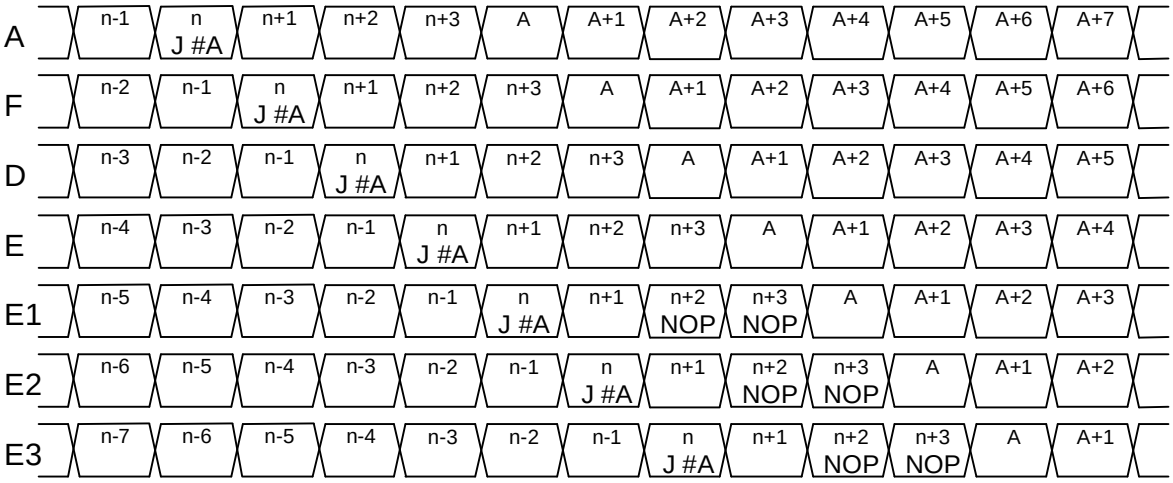


Рис.4. Временная диаграмма работы 7-фазного программного конвейера DSP-ядра при выполнении инструкции программного перехода BD(JD) #A.

4.1 Время исполнения вычислительных операций

В таблице указано время исполнения инструкций DSP-ядра, измеряемое в командных циклах.

Вычислительные операции	Время исполнения, тактов
Все логические операции: AND, ANDL, ANDD, ANDC, ANDCL, ANDCD, ANDI, ANDIL, ANDID, OR, ORL, ORD, ORC, ORCL, ORCD, ORI, ORIL, 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

4.2 Ограничения конвейера

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 16-разрядный регистр данных, нельзя начинать выполнение программного цикла DO по этому регистру, необходимо выдержать 2 такта пропуска.

Пример:

Неправильно:

```
MOVE R0,R3
DO R3, label
```

Правильно:

```
MOVE R0,R3
NOP
NOP
DO R3, label
```

В том случае, когда регистр данных изменяется в результате 2-тактной вычислительной операции, необходимо выдержать 3 такта пропуска.

Пример:

Неправильно:

```
ADD R0,R3
DO R3, label
```

Правильно:

```
ADD R0,R3
NOP
NOP
NOP
DO R3, label
```

3) 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
```

5. Система инструкций

В систему инструкций вносятся следующие изменения по сравнению с системой инструкций, определенной для предшествующих модификаций DSP-ядер ELcore-xx,:

1) Введены новые 64- и 128- разрядные форматы данных;

2) Введены дополнительный набор операций над новыми форматами и типами данных, в том числе 64- и 128- разрядные пересылки данных;

3) Изменены форматы результатов для команд ADDSUB, ADDSUBL, ADDSUBX, FAS, CVFE, MPYL, MAC, MAC2, MACL, SAC2, CS2;

4) Введены новые форматы инструкций и кодировки для новых команд.

5.1 Форматы данных

Введены 64- и 128- разрядные форматы данных в памяти и регистровом файле, предназначенные для хранения нескольких операндов меньшей разрядности (см. п.3.1 и Приложение 1).

5.2 Форматы пересылок

В дополнение к 16-ти и 32-разрядным пересылкам, определенным в предшествующих модификациях DSP-ядер Elcore-xx, вводятся новые форматы пересылок для выпонения обменов 64- и 128- разрядными данными. Вносимые изменения сводятся к следующим пунктам:

1) Пересылки непосредственных данных (16- и 32- разрядных) остаются без изменения.

2) Вводятся 64- и 128- разрядные обмены с памятью данных XRAM, YRAM. Эти обмены осуществляются инструкциями форматов 9a, 9b. Разрядность обменов определяется битом “u”(u=0 - 64-разрядные; u=1 - 128-разрядные обмены).

3) Вводятся 64- и 128- разрядные обмены между регистрами данных (RF). Такие обмены выполняются инструкциями форматов 2t, 9b. Разрядность обменов определяется битами L2, L (см. п.5.5.4).

4) Обмены с регистрами управления выполняются в форматах 2t, 8d, 9d. Для выполнения обменов с регистрами управления RC все они условно разделены на 4 подмножества:

- регистры адресных генераторов AGU, AGU-Y;
- регистры обменного буфера XBUF;
- регистры управления PCU;
- регистры-аккумуляторы;

Выбор подмножества регистров управления определяется в форматах 2t, 8d, 9d битами “sc”, “sx” (см. п.5.5.3).

Разрядность обменов с регистрами управления определяется прежде всего разрядностью самих этих регистров. В некоторых случаях с одними и теми же регистрами управления возможны обмены с различной разрядностью:

а) это имеет место для адресных регистров An, AT, с которыми возможны как 16-, так и 32-разрядные обмены.

б) это имеет место для регистров-аккумуляторов ACi.[L/D/Q], с которыми возможны как 32-, так и 64- и 128-разрядные обмены.

В тех случаях, когда с регистрами управления возможны обмены с различной разрядностью, разрядность определяется форматом полем L2, L согласно п.5.5.4.

5.3 Модификация адреса при 64- и 128-разрядных обменах

При выпонения обменов 64- и 128- разрядными данными с памятью XYRAM при выборе вида адресации (An) + (пост–инкремент на 1) или (An) - (пост–декремент на 1) модификация адресов памяти данных в регистрах A0-A7 адресного генератора AGU происходит с учетом формата пересылаемых данных по следующим правилам:

Вид адресации	Модификация адреса		
	32-р обмен	64-р обмен	128-р обмен
(An) +	An = An + 1	An = An + 2	An = An + 4
(An) -	An = An – 1	An = An – 2	An = An – 4

5.4 Команда возврата из прерывания RTI

В дополнение к определенным в предшествующих модификациях DSP-ядер ELcore-xx, вводится новая команда упраления RTI - команда возврата из подпрограммы обработки прерывания.

5.5 Дополнительный набор вычислительных команд

В дополнение к определенной в предшествующих модификациях DSP-ядер ELcore-x4 системе инструкций, вводятся новые вычислительные команды, перечень которых приведен в Приложении 1.

5.6 Изменение форматов результатов некоторых вычислительных команд

Вследствие введения 64- и 128- разрядных форматов данных изменяются форматы результатов для следующих вычислительных команд:
OP1: ADDSUB, ADDSUBL, ADDSUBX, FAS, CVFE,
OP2: MPYL, MAC, MAC2, MACL, SAC2, CS2.
Особенностью указанных команд в предшествующих модификациях DSP-ядер ELcore-xx было наличие двух результатов (16- либо 32-разрядных), один из которых записывался по адресу результата (d/D), другой - по адресу исходного операнда (s/S/s2/S2). Недостаток состоял в том, что исходный операнд при этом затирался.
В ELcore-30M для каждой из этих команд оба результата объединяются в один результат вдвое большего формата (32 или 64 разряда), который записывается по адресу результата D.L(D.D).
Однако для обеспечения преемственности системы инструкций для команд ADDSUB, ADDSUBL, ADDSUBX, FAS, CVFE сохранена возможность исполнения в прежнем варианте. Выбор того или иного варианта исполнения производится при помощи флага DD (Double Destination), хранящегося в 9-м разряде регистра SR: DD=SR[9]. При DD=0 (по умолчанию) указанные команды выполняются в прежнем варианте (с двумя результатами и двумя адресами записи), при DD=1 один результат удвоенного формата записывается по одному адресу D.L(D.D). Для команд MPYL, MAC, MAC2, MACL, SAC2, CS2 предусмотрен только один, одноадресный вариант.

5.7 Рекомендации по переносу DSP-программ с процессоров 1892ВМ3Т,1892ВМ2Я, 1892ВМ4Я, 1892ВМ5Я на процессор 1892ВМ10Я

1) Управляющие биты регистра SR должны оставаться в исходном состоянии: DD=SR[9]=0, BD=SR[10]=0.
2) При наличии в программе какой-либо из следующих команд - MPYL, MAC, MAC2, MACL, SAC2, CS2 – необходимо скорректировать код с учётом п.5.7 в части адреса результата.
Пример:
Исходный код:
MPYL R0.L,R2.L,R8.L
MOVE R2.L,R4.L
Скорректированный код:
MPYL R0.L,R2.L,R8.D
MOVE R9.L,R4.L
4) Необходимо учесть ограничения конвейера согласно п.4.

5.8 Форматы инструкций и коды операций

5.8.1 Форматы 9a, 9b и 9d

В дополнение к определенным в предшествующих модификациях DSP-ядер ELcore-xx вводятся новые форматы инструкций - 9a, 9b и 9d.

Разряды [10:7] первого слова инструкции этих форматов (основное поле кода формата) = 1111. Для формата 9a поле OP3=00, для формата 9b поле OP3=01, для формата 9d поле OP3=11.

Формат	Условие	Операция 1	Операция 2	Пересылка 1	Пересылка 2	Длина инстр.
9a	-	OP2 T,S,D	OP1 T,S,D	XRAM ß à R.D(Q)	YRAM à R0	64 бит
9b	-	OP2 T,S,D	OP1 T,S,D	R.D(Q) ß à R.D(Q)	YRAM à R0	64 бит
9d	-	OP2 T,S,D	OP1 T,S,D	RC ß à R.(L,D,Q)	-	64 бит

Длина кода инструкции форматов 9a, 9b и 9d - 64 разряда.

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.

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.

При написании инструкции формата 9a разрядность обменов с памятью задается в поле «Пересылка 1» после номера регистра RF и точки ключом «D» для 64-разрядных обменов или «Q» для 128-разрядных обменов.

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.

В таблице 16 приведена структура всех форматов инструкций DSP Elcore-28. Вновь введенные форматы выделены жирным шрифтом в нижних строках таблицы.

Приведенная структура форматов 9a, 9b, 9d в основном повторяет структуру форматов соответственно 8a, 8b, 8d с тем основным отличием, что код операции OP2 для новых форматов формируется как конкатенация полей M (разряд 16) и OP2 (разряды 37:32).

5.8.2 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.

5.8.3 Назначение полей “sc”, “sx” в форматах 2t, 8d, 9d

Поля “sc”, “sx” в форматах 2t, 8d, 9d определяют выбор подмножества регистров управления RC, участвующих в пересылке, согласно следующему правилу:

sc	sx	Выбор подмножества регистров RC
0	0	Регистры адресных генераторов AGU, AGU-Y
0	1	Регистры обменного буфера XBUF
1	0	Регистры управления PCU
1	1	Регистры-аккумуляторы ACi.[L/D/Q]

Примечание. В формате 2t при sr=1 пересылка выполняентся между двумя регистрами RF, и и состояние полей “sc”, “sx” в этом случае не играет роли.

5.8.4 Кодирование адресов регистров

5.8.4.1 Кодирование адресов регистров данных (RF)

Кодирование адресов регистров данных в 5-битных полях T, S, D, R/Rs/Rd производится путем указания номера регистра регистрового файла от 0 до 31. При кодировании операций обменов с памятью в форматах 8a, 9a 4-битным полем R кодируются только регистры с четными номерами, младший бит не указывается.

5.8.4.2 Кодирование адресов регистров адресных генераторов AGU, AGU-Y

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

Таблица 9. Кодирование адресов регистров адресных генераторов AGU, AGU-Y

Условное обозначение	Разряд-ность	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
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	16	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

5.8.4.3 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.

5.8.4.4 Кодирование адресов регистров управления PCU

Поле “RC” в форматах 2t, 8d, 9d при обменах с регистрами управления PCU, включая регистры состояния ALU (CCR, PDNR) и регистр специальных функций SFR, кодируется согласно таблице 10.

Таблица 10. Кодирование адресов регистров управления PCU

Условное обозначение	Разряд-ность	Тип	Назначение регистра	Код адреса регистра
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
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

5.8.4.5 Кодирование адресов регистров-аккумуляторов

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

5.8.5 Назначение полей “L”, “L2” в форматах 2t, 8d, 9b, 9d

Поля “L”, “L2” в форматах 2t, 8d, 9b, 9d определяют разрядность пресылаемых данных согласно следующим правилам:

1) Формат 2t, пересылка между двумя регистрами данных

L2	L	Разрядность пересылаемых данных
0	0	16-разрядная пересылка
0	1	32-разрядная пересылка
1	0	64-разрядная пересылка
1	1	128-разрядная пересылка

2) Формат 9b, поле “L” определяет разрядность пересылки между двумя регистрами данных

L	Разрядность пересылаемых данных
0	64-разрядная пересылка
1	128-разрядная пересылка

3) Формат 2t, 8d, 9d, пересылка между адресным регистром An, AT и регистром данных

L	Разрядность пересылаемых данных
0	16-разрядная пересылка
1	32-разрядная пересылка

4) Формат 2t, пересылка между регистром-аккумулятором ACi.[L/D/Q], и регистром данных

L2	L	Разрядность пересылаемых данных
0	0	-
0	1	32-разрядная пересылка (ACi.L)
1	0	64-разрядная пересылка (ACi.D)
1	1	128-разрядная пересылка (ACi.Q)

5) Формат 8d, пересылка между регистром–аккумулятором ACi.[L/D], и регистром данных

L	Разрядность пересылаемых данных
0	64-разрядная пересылка (ACi.D)
1	32-разрядная пересылка (ACi.L)

6) Формат 9d, пересылка между регистром-аккумулятором ACi.[D/Q], и регистром данных

L	Разрядность пересылаемых данных
0	64-разрядная пересылка (ACi.D)
1	128-разрядная пересылка (ACi.Q)

5.8.6 Коды операций в форматах 9a, 9b и 9d

Коды операций типа OP1 в форматах 9a, 9b и 9d приведены в таблице 12. Коды операций типа OP2 в форматах 9a, 9b и 9d приведены в таблице 13. *Курсивом выделены команды, не реализованные в первой версии процессора МС-0428.*

5.8.7 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 11. Форматы инструкций Elcore-28

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 (OP1)					**D (OP1)**					**S (OP1)**					**M**	**R**			**L**		**1**	**1**	**1**	**1**	**OP1**								**9a**
T (OP2) / #5					**D (OP2)**					**S (OP2)**					**AT**	**mode**			**A**				**de**	**OP3**		**OP2**					**OP3=00**
T (OP1)					**D (OP1)**					**S (OP1)**					**M**	**RS**					**1**	**1**	**1**	**1**	**OP1**								**9b**
T (OP2) / #5					**D (OP2)**					**S (OP2)**					**AT**	**RD**						**L**	**u**	**OP3**		**OP2**					**OP3=01**
T (OP1)					**D (OP1)**					**S (OP1)**					**M**	**Rs/Rd**					**1**	**1**	**1**	**1**	**OP1**								**9d**
T (OP2) / #5					**D (OP2)**					**S (OP2)**					**sx**	**L**	**RC**					**sc**	**de**	**OP3**		**OP2**					**OP3=11**

Таблица 12. Коды операций типа OP1 в форматах 9a, 9b, 9d DSP-ядра Elcore-28

FAS + AU + TU		SUM		LU		SU	
OP1	Операция	**OP1**	Операция	**OP1**	Операция	**OP1**	Операция
0000000	**NOP**	0100000		1000000	**NOP**	1100000	**CMPN4**
0000001	*CLRD*	0100001	**ASX2**	1000001		1100001	**CMPN8**
0000010	*ADDD*	0100010	**ASXS**	1000010		1100010	**CMPZ4**
0000011	*SUBD*	0100011	**ASXS2**	1000011		1100011	**CMPZ8**
0000100	*ADCD*	0100100	**A4**	1000100		1100100	**CMPNL2**
0000101	*SBCD*	0100101	**A8**	1000101		1100101	**CMPNL4**
0000110	*ADDLD*	0100110	**S4**	1000110		1100110	**CMPZL2**
0000111	**TRS16**	0100111	**S8**	1000111		1100111	**CMPZL4**
0001000	**FASX**	0101000	**RA4**	1001000		1101000	**MAX4**
0001001	**FASXS**	0101001	**RA8**	1001001		1101001	**MAX8**
0001010	**FAX**	0101010	**SGA4**	1001010		1101010	**MIN4**
0001011	**FSX**	0101011	**SGA8**	1001011		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		1101111	**MINL4**
0010000	**TRS0**	0110000	**S8s**	1010000		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		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 13. Коды операций типа OP2 в форматах 9a, 9b, 9d DSP-ядра Elcore-28

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

6. Перечень адресуемых регистров DSP-кластера

Перечень адресуемых регистров DSP-кластера DELCore-30M приведен в таблице 14.

Таблица 14. Перечень адресуемых регистров DSP-кластера (i=0,1 – номер DSP; BASE(0)=0x1848_0000; BASE(1)=0x1888_0000)

Условное обозначение	Разряд-ность	Тип	Назначение регистра	Адрес регистра
			Общие 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
			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
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

Продолжение таблицы 14.

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

Продолжение таблицы 14.

Условное обозначение	Разряд-ность	Тип	Назначение регистра	Адрес регистра
			AGU, AGU-Y	
A0	32	R/W	Регистр адреса A0	BASE(i)+0x0080
A1	32	R/W	Регистр адреса A1	BASE(i)+0x0084
A2	32	R/W	Регистр адреса A2	BASE(i)+0x0088
A3	32	R/W	Регистр адреса A3	BASE(i)+0x008C
A4	32	R/W	Регистр адреса A4	BASE(i)+0x0090
A5	32	R/W	Регистр адреса A5	BASE(i)+0x0094
A6	32	R/W	Регистр адреса A6	BASE(i)+0x0098
A7	32	R/W	Регистр адреса A7	BASE(i)+0x009C
I0	16	R/W	Регистр индекса I0	BASE(i)+0x00A0
I1	16	R/W	Регистр индекса I1	BASE(i)+0x00A4
I2	16	R/W	Регистр индекса I2	BASE(i)+0x00A8
I3	16	R/W	Регистр индекса I3	BASE(i)+0x00AC
I4	16	R/W	Регистр индекса I4	BASE(i)+0x00B0
I5	16	R/W	Регистр индекса I5	BASE(i)+0x00B4
I6	16	R/W	Регистр индекса I6	BASE(i)+0x00B8
I7	16	R/W	Регистр индекса I7	BASE(i)+0x00BC
M0	16	R/W	Регистр модификатора M0	BASE(i)+0x00C0
M1	16	R/W	Регистр модификатора M1	BASE(i)+0x00C4
M2	16	R/W	Регистр модификатора M2	BASE(i)+0x00C8
M3	16	R/W	Регистр модификатора M3	BASE(i)+0x00CC
M4	16	R/W	Регистр модификатора M4	BASE(i)+0x00D0
M5	16	R/W	Регистр модификатора M5	BASE(i)+0x00D4
M6	16	R/W	Регистр модификатора M6	BASE(i)+0x00D8
M7	16	R/W	Регистр модификатора M7	BASE(i)+0x00DC
AT	32	R/W	Регистр адреса AT	BASE(i)+0x00E0
IT	16	R/W	Регистр индекса IT	BASE(i)+0x00E4
MT	16	R/W	Регистр модификатора MT	BASE(i)+0x00E8
DT	16	R/W	Регистр модификатора DT	BASE(i)+0x00EC
IVAR	16	R/W	Регистр адреса вектора прерывания	BASE(i)+0x00FC

Продолжение таблицы 14.

Условное обозначение	Разряд-ность	Тип	Назначение регистра	Адрес регистра
			Регистры данных RF	
R0.L	32		Регистр данных	BASE(i)+0x0000
R2.L	32		Регистр данных	BASE(i)+0x0004
R4.L	32		Регистр данных	BASE(i)+0x0008
R6.L	32		Регистр данных	BASE(i)+0x000C
R8.L	32		Регистр данных	BASE(i)+0x0010
R10.L	32		Регистр данных	BASE(i)+0x0014
R12.L	32		Регистр данных	BASE(i)+0x0018
R14.L	32		Регистр данных	BASE(i)+0x001C
R16.L	32		Регистр данных	BASE(i)+0x0020
R18.L	32		Регистр данных	BASE(i)+0x0024
R20.L	32		Регистр данных	BASE(i)+0x0028
R22.L	32		Регистр данных	BASE(i)+0x002C

R24.L	32		Регистр данных	BASE(i)+0x0030
R26.L	32		Регистр данных	BASE(i)+0x0034
R28.L	32		Регистр данных	BASE(i)+0x0038
R30.L	32		Регистр данных	BASE(i)+0x003C
R1.L	32		Регистр данных	BASE(i)+0x0040
R3.L	32		Регистр данных	BASE(i)+0x0044
R5.L	32		Регистр данных	BASE(i)+0x0048
R7.L	32		Регистр данных	BASE(i)+0x004C
R9.L	32		Регистр данных	BASE(i)+0x0050
R11.L	32		Регистр данных	BASE(i)+0x0054
R13.L	32		Регистр данных	BASE(i)+0x0058
R15.L	32		Регистр данных	BASE(i)+0x005C
R17.L	32		Регистр данных	BASE(i)+0x0060
R19.L	32		Регистр данных	BASE(i)+0x0064
R21.L	32		Регистр данных	BASE(i)+0x0068
R23.L	32		Регистр данных	BASE(i)+0x006C
R25.L	32		Регистр данных	BASE(i)+0x0070
R27.L	32		Регистр данных	BASE(i)+0x0074
R29.L	32		Регистр данных	BASE(i)+0x0078
R31.L	32		Регистр данных	BASE(i)+0x007C
R1.D[31:0]	32		Регистр данных	BASE(i)+0x0180
R1.D[63:32]	32		Регистр данных	BASE(i)+0x0184
R3.D[31:0]	32		Регистр данных	BASE(i)+0x0188
R3.D[63:32]	32		Регистр данных	BASE(i)+0x018C
R5.D[31:0]	32		Регистр данных	BASE(i)+0x0190
R5.D[63:32]	32		Регистр данных	BASE(i)+0x0194
R7.D[31:0]	32		Регистр данных	BASE(i)+0x0198
R7.D[63:32]	32		Регистр данных	BASE(i)+0x019C
R9.D[31:0]	32		Регистр данных	BASE(i)+0x01A0
R9.D[63:32]	32		Регистр данных	BASE(i)+0x01A4
R11.D[31:0]	32		Регистр данных	BASE(i)+0x01A8
R11.D[63:32]	32		Регистр данных	BASE(i)+0x01AC
R13.D[31:0]	32		Регистр данных	BASE(i)+0x01B0
R13.D[63:32]	32		Регистр данных	BASE(i)+0x01B4
R15.D[31:0]	32		Регистр данных	BASE(i)+0x01B8
R15.D[63:32]	32		Регистр данных	BASE(i)+0x01BC
R17.D[31:0]	32		Регистр данных	BASE(i)+0x01C0
R17.D[63:32]	32		Регистр данных	BASE(i)+0x01C4
R19.D[31:0]	32		Регистр данных	BASE(i)+0x01C8
R19.D[63:32]	32		Регистр данных	BASE(i)+0x01CC
R21.D[31:0]	32		Регистр данных	BASE(i)+0x01D0
R21.D[63:32]	32		Регистр данных	BASE(i)+0x01D4
R23.D[31:0]	32		Регистр данных	BASE(i)+0x01D8
R23.D[63:32]	32		Регистр данных	BASE(i)+0x01DC
R25.D[31:0]	32		Регистр данных	BASE(i)+0x01E0
R25.D[63:32]	32		Регистр данных	BASE(i)+0x01E4
R27.D[31:0]	32		Регистр данных	BASE(i)+0x01E8
R27.D[63:32]	32		Регистр данных	BASE(i)+0x01EC
R29.D[31:0]	32		Регистр данных	BASE(i)+0x01F0
R29.D[63:32]	32		Регистр данных	BASE(i)+0x01F4
R31.D[31:0]	32		Регистр данных	BASE(i)+0x01F8
R31.D[63:32]	32		Регистр данных	BASE(i)+0x01FC

Продолжение таблицы 14.

Условное обозначение	Разряд-ность	Тип	Назначение регистра	Адрес регистра
			Регистры-аккумуляторы	
AC0	32	R/W	Регистр-аккумулятор AC0	BASE(i)+0x0200
AC1	32	R/W	Регистр-аккумулятор AC1	BASE(i)+0x0204
AC2	32	R/W	Регистр-аккумулятор AC2	BASE(i)+0x0208
AC3	32	R/W	Регистр-аккумулятор AC3	BASE(i)+0x020C
AC4	32	R/W	Регистр-аккумулятор AC4	BASE(i)+0x0210
AC5	32	R/W	Регистр-аккумулятор AC5	BASE(i)+0x0214
AC6	32	R/W	Регистр-аккумулятор AC6	BASE(i)+0x0218
AC7	32	R/W	Регистр-аккумулятор AC7	BASE(i)+0x021C
AC8	32	R/W	Регистр-аккумулятор AC8	BASE(i)+0x0220
AC9	32	R/W	Регистр-аккумулятор AC9	BASE(i)+0x0224
AC10	32	R/W	Регистр-аккумулятор AC10	BASE(i)+0x0228
AC11	32	R/W	Регистр-аккумулятор AC11	BASE(i)+0x022C
AC12	32	R/W	Регистр-аккумулятор AC12	BASE(i)+0x0230
AC13	32	R/W	Регистр-аккумулятор AC13	BASE(i)+0x0234
AC14	32	R/W	Регистр-аккумулятор AC14	BASE(i)+0x0238
AC15	32	R/W	Регистр-аккумулятор AC15	BASE(i)+0x023C

Продолжение таблицы 14.

Условное обозначение	Разряд-ность	Тип	Назначение регистра	Адрес регистра
			Отладочные регистры	
dbDCSR	16	R/W	Регистр управления в режиме отладки	BASE(i)+0x0500
Cnt_RUN	32	R	Счетчик тактов	BASE(i)+0x0518
dbPCa	16	R	Программный счетчик, стадия a	BASE(i)+0x0524
dbPCf	16	R	Программный счетчик, стадия f	BASE(i)+0x0528
dbPCd	16	R	Программный счетчик, стадия d	BASE(i)+0x052C
dbPCe	16	R	Программный счетчик, стадия e	BASE(i)+0x0520
dbPCe1	16	R	Программный счетчик, стадия e1	BASE(i)+0x0530
dbPCe2	16	R	Программный счетчик, стадия e2	BASE(i)+0x0534
dbPCe3	16	R	Программный счетчик, стадия e3	BASE(i)+0x0538
dbSAR	16	R/W	Регистр адреса останова 0 в режиме отладки	BASE(i)+0x053C
dbCNTR	16	R/W	Счетчик исполненных команд в режиме отладки	BASE(i)+0x0540
dbSAR1	16	R/W	Регистр адреса останова 1 в режиме отладки	BASE(i)+0x0544
dbSAR2	16	R/W	Регистр адреса останова 2 в режиме отладки	BASE(i)+0x0548
dbSAR3	16	R/W	Регистр адреса останова 3 в режиме отладки	BASE(i)+0x054C
dbSAR4	16	R/W	Регистр адреса останова 4 в режиме отладки	BASE(i)+0x0550
dbSAR5	16	R/W	Регистр адреса останова 5 в режиме отладки	BASE(i)+0x0554
dbSAR6	16	R/W	Регистр адреса останова 6 в режиме отладки	BASE(i)+0x0558
dbSAR7	16	R/W	Регистр адреса останова 7 в режиме отладки	BASE(i)+0x055C

Приложение П 1

Расширение системы команд DSP-ядер процессора 1892ВМ10Я

(x)* - число тактов исполнения

Обозначения и пояснения

Форматы данных в регистровом файле

№	Обозначение регистра	Разрядность регистра	Номера	Варианты форматов данных
1	**T** или **S** или **D**	16	31:0	R = r[15:0]
2	**T.L** или **S.L** или **D.L**	32	31:0	L = (l_1 , l_0), l_1 = L[31:16 bits], l_0 = L[15:0 bits] L = (complex) LX = l_1 + j l_0
3	**T.D** или **S.D** или **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)
4	**T.Q** или **S.Q** или **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)

Примечание. Приведенные в настоящем приложении команды не формируют признаков результата в регистре CCR, если обратное не указано в описании команды.

	Умножитель, плавающая точка
1	**FM2 T.D,S.D,D.D (2)*** $D = (T_1 \bullet S_1 , T_0 \bullet S_0)$ Два умножения, (float32)
2	**FMS2 T.D,S.D,D.D (2)** $D=(T_1 \bullet S_0 , T_0 \bullet S_1)$ Два умножения с перестановкой полуслов в S, (float32)

	АЛУ, плавающая точка
1	**FASX T.D,S.D,D.Q (2)** $D = (SX - TX , SX + TX) = (S_1 - T_1 , S_0 - T_0 , S_1 + T_1 , S_0 + T_0)$ Сложение и вычитание комплексных операндов, (float32)
2	**FASXS T.D,S.D,D.Q (2)** $D = (SX + j\,TX , SX - j\,TX) = (S_1 - T_0 , S_0 + T_1 , S_1 + T_0 , S_0 - T_1)$ Сложение и вычитание комплексных операндов с перестановкой в T, (float32)
3	**FSA T.D,S.D,D.D (2)** $D = (T_1 - T_0 , S_1 + S_0)$ Вычитание и сложение, (float32)
4	**FAX T.D,S.D,D.D (2)** $D = (S_1 + T_1 , S_0 + T_0)$ Сложение комплексных операндов, (float32)
5	**FSX T.D,S.D,D.D (2)** $D = (S_1 - T_1 , S_0 - T_0)$ Вычитание комплексных операндов, (float32)

	Умножитель, фиксированная точка Дополнительный режим работы: сатурация. **Сатурация**: присваивание результату предельного (максимального или минимального) значения при переполнении. Режим инициируется посредством записи признака в служебный регистр.
1	**ML2 T.D,S.D,D.Q (2)** $D = (T_1 \cdot S_1, T_0 \cdot S_0)$ Два умножения, целые, 32 • 32 → 64
2	**UML2 T.D,S.D,D.Q (2)** $D = (T_1 \cdot S_1, T_0 \cdot S_0)$ Два умножения, целые, S без знака, 32 • 32 → 64
4	**UML T.L,S.L,D.D (2)** D = T • S Умножение, целое, без знака, 32 • 32 → 64
5	**MF8 T.Q,S.Q,D.Q (2)** $D = (t_7 \cdot s_7, t_6 \cdot s_6, t_5 \cdot s_5, t_4 \cdot s_4, t_3 \cdot s_3, t_2 \cdot s_2, t_1 \cdot s_1, t_0 \cdot s_0)$ Восемь умножений, дробные, 16 •16 → 32 округление → 16, сатурация
6	**MF4 T.D,S.D,D.D (2)** $D = (t_3 \cdot s_3, t_2 \cdot s_2, t_1 \cdot s_1, t_0 \cdot s_0)$ Четыре умножения, дробные, 16 •16 → 32 округление → 16, сатурация
7	**MFX2 T.D,S.D,D.D (2)** $D = (TX_1 \cdot SX_1, TX_0 \cdot SX_0)$ Два умножения комплексные, дробные, (16+j16) • (16+j16) → (32+j32) округление → (16+j16)
8	**MFXC2 T.D,S.D,D.D (2)** $D = (TX_1 \cdot (conj)SX_1, TX_0 \cdot (conj)SX_0)$ Два умножения комплексные с сопряжением SX, дробные, (16+j16) • (16+j16) → (32+j32) округление → (16+j16)
9	**MFX T.L,S.L,D.L (2)** D = DX = TX • SX Умножение комплексное, дробное, (16+j16) • (16+j16) → (32+j32) округление → (16+j16)
10	**MFXC T.L ,S.L,D.L (2)** D = DX = TX • (conj)SX Умножение комплексное с сопряжением SX, дробное, (16+j16) • (16+j16) → (32+j32) округление → (16+j16)
11	**MFA81 T.Q,S.Q,D (2)** $D = (t_7 \cdot s_7 + t_6 \cdot s_6 + t_5 \cdot s_5 + t_4 \cdot s_4 + t_3 \cdot s_3 + t_2 \cdot s_2 + t_1 \cdot s_1 + t_0 \cdot s_0)$ Сумма восьми произведений, дробные умножения, (16 • 16 + 32) округление → 16

12	**MFA42 T.Q,S.Q,D.L (2)** D = (($t_7 \bullet s_7 + t_6 \bullet s_6 + t_5 \bullet s_5 + t_4 \bullet s_4$, $t_3 \bullet s_3 + t_2 \bullet s_2 + t_1 \bullet s_1 + t_0 \bullet s_0$) Две суммы четырех произведений, дробные умножения, (16 • 16 + 32) округление → 16
13	**MFA41 T.D,S.D,D (2)** D = ($t_3 \bullet s_3 + t_2 \bullet s_2 + t_1 \bullet s_1 + t_0 \bullet s_0$) Сумма четырех произведений, дробные умножения, (16 • 16 + 32) округление → 16
14	**MFA24 T.Q,S.Q,D.D (2)** D = ($t_7 \bullet s_7 + t_6 \bullet s_6$, $t_5 \bullet s_5 + t_4 \bullet s_4$, $t_3 \bullet s_3 + t_2 \bullet s_2$, $t_1 \bullet s_1 + t_0 \bullet s_0$) Четыре суммы двух произведений, дробные умножения, (16 • 16 + 32) округление → 16
15	**MFA22 T.D,S.D,D.L (2)** D = ($t_3 \bullet s_3 + t_2 \bullet s_2$, $t_1 \bullet s_1 + t_0 \bullet s_0$) Две суммы двух произведений, дробные умножения, (16 • 16 + 32) округление → 16
16	**MFA21 T.L,S.L,D (2)** D = ($t_1 \bullet s_1 + t_0 \bullet s_0$) Сумма двух произведений, дробные умножения, (16 • 16 + 32) округление → 16
17	**M4 T.D,S.D,D.Q (2)** D = ($t_3 \bullet s_3$, $t_2 \bullet s_2$, $t_1 \bullet s_1$, $t_0 \bullet s_0$) Четыре умножения, целые, 16 •16 → 32
18	**M2 T.L,S.L,D.D (2)** D = ($t_1 \bullet s_1$, $t_0 \bullet s_0$) Два умножения, целые, 16 •16 → 32

Умножитель + Аккумулятор, управление аккумуляторами	
1	**CLRAC T (2)** Групповая очистка 32-разрядных аккумуляторов ACn по маске в 16-разрядном регистре T («1» в n-м бите вызывает сброс)
2	**LDAC T,D.L (2)** Выгрузка содержимого 32-разрядного аккумулятора ACn в 32-разрядный регистр регистрового файла D.L. Номер аккумулятора (n) задается установкой в «1» n-го разряда 16-разрядного регистра T.
3	**LDACD T,D.D (2)** 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.
4	**STAC T,S.L (2)** 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.
5	**STACD T,S.D (2)** 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.

	Умножитель + Аккумулятор, фиксированная точка
1	**MAC81 T.Q,S.Q,ACn.D (2)** $ACn.D + = (t_7 \cdot s_7 + t_6 \cdot s_6 + t_5 \cdot s_5 + t_4 \cdot s_4 + t_3 \cdot s_3 + t_2 \cdot s_2 + t_1 \cdot s_1 + t_0 \cdot s_0)$, n=0,1,2,3,4,5,6,7 Сумма восьми произведений с аккумуляцией, целые, 16 • 16 + 64
2	**MAC42 T.Q,S.Q,ACn.D (2)** $AC(n+1) + = (t_7 \cdot s_7 + t_6 \cdot s_6 + t_5 \cdot s_5 + t_4 \cdot s_4)$ $ACn + = (t_3 \cdot s_3 + t_2 \cdot s_2 + t_1 \cdot s_1 + t_0 \cdot s_0)$, n=0,2,4,6 Две суммы четырех произведений с аккумуляцией, целые, 16 • 16 + 64
3	**MAC41 T.D,S.D,ACn.D (2)** $ACn + = (t_3 \cdot s_3 + t_2 \cdot s_2 + t_1 \cdot s_1 + t_0 \cdot s_0)$, n=0,1,2,3,4,5,6,7 Сумма четырех произведений с аккумуляцией, целые, 16 • 16 + 64
4	**MAC24 T.Q,S.Q,ACn.D (2)** $AC(n+3) + = (t_7 \cdot s_7 + t_6 \cdot s_6)$ $AC(n+2) + = (t_5 \cdot s_5 + t_4 \cdot s_4)$ $AC(n+1) + = (t_3 \cdot s_3 + t_2 \cdot s_2)$ $ACn + = (t_1 \cdot s_1 + t_0 \cdot s_0)$, n=0,4 Четыре суммы двух произведений с аккумуляцией, целые, 16 • 16 + 64

5	**MAC22 T.D,S.D,ACn.D (2)** $AC(n+1) + = (t_3 \bullet s_3 + t_2 \bullet s_2)$ $ACn + = (t_1 \bullet s_1 + t_0 \bullet s_0)$, n=0,2,4,6 Две суммы двух произведений с аккумуляцией, целые, 16 • 16 + 64
6	**MAC21 T.L,S.L,ACn.D (2)** $ACn + = (t_1 \bullet s_1 + t_0 \bullet s_0)$, n=0,1,2,3,4,5,6,7 Сумма двух произведений с аккумуляцией, целые, 16 • 16 + 64
7	**MAC18 T.Q,S.Q (2)** $AC(n+7) + = (t_7 \bullet s_7)$ $AC(n+6) + = (t_6 \bullet s_6)$ $AC(n+5) + = (t_5 \bullet s_5)$ $AC(n+4) + = (t_4 \bullet s_4)$ $AC(n+3) + = (t_3 \bullet s_3)$ $AC(n+2) + = (t_2 \bullet s_2)$ $AC(n+1) + = (t_1 \bullet s_1)$ $ACn + = (t_0 \bullet s_0)$, n=0 Восемь умножений с аккумуляцией, целые, 16 • 16 + 64
8	**MAC14 T.D,S.D,ACn.D (2)** $AC(n+3) + = (t_3 \bullet s_3)$ $AC(n+2) + = (t_2 \bullet s_2)$ $AC(n+1) + = (t_1 \bullet s_1)$ $ACn + = (t_0 \bullet s_0)$, n=0,4 Четыре умножения с аккумуляцией, целые, 16 • 16 + 64
9	**MAC12 T.L,S.L,ACn.D (2)** $AC(n+1) + = (t_1 \bullet s_1)$ $ACn + = (t_0 \bullet s_0)$, n=0,2,4,6 Два умножения с аккумуляцией, целые, 16 • 16 + 64
10	**MAC11** (MAC) **T,S,ACn.D (2)** $ACn + = (t \bullet s)$, n=0,1,2,3,4,5,6,7 Умножение с аккумуляцией, целое, 16 • 16 + 64
11	**MACX2 T.D,S.D,ACn.D (2)** $(AC\ n+1 + j\ ACn) + = (TX_1 \bullet SX_1 + TX_0 \bullet SX_0)$, $AC\ n+1 + = (t_3 \bullet s_3 – t_2 \bullet s_2 + t_1 \bullet s_1 – t_0 \bullet s_0)$, $ACn + = (t_3 \bullet s_2 + t_2 \bullet s_3 + t_1 \bullet s_0 + t_0 \bullet s_1)$, n=0,2,4,6 Два умножения – аккумуляция комплексных операндов, целые, (16+j16) • (16+j16) → (32+j32)+ (64+j64)
12	**MACXC2 T.D,S.D,ACn.D (2)**

	(AC n+1 + j ACn) + = (TX_1 • (conj) SX_1 + TX_0 • (conj) SX_0), AC n+1 + = (t_3 • s_3+ t_2 • s_2 + t_1 • s_1 + t_0 • s_0), ACn + = (- t_3 • s_2 + t_2 • s_3 – t_1 • s_0 + t_0 • s_1) , n=0,2,4,6 Два умножения – аккумуляция комплексных операндов, сопряжение SX, целые, (16+j16) • (16+j16) → (32+j32)+ (64+j64)
13	**MACX T.L,S.L,ACn.D (2)** (AC(n+1) + j AC(n)) + = (TX • SX), AC(n+1) + = (t_1 • s_1 – t_0 • s_0), AC(n) + = (t_1 • s_0 + t_0 • s_1) , n=0,2,4,6 Умножение с аккумуляцией, комплексное, целое, (16+j16) • (16+j16) + (64+j64)
14	**MACXC T.L,S.L,ACn.D (2)** (AC(n+1) + j AC(n)) + = (TX •(conj) SX), AC(n+1) + = (t_1 • s_1 + t_0 • s_0), AC(n) + = (- t_1 • s_0 + t_0 • s_1) , n=0,2,4,6 Умножение с аккумуляцией, комплексное, целое, сопряжение SX, (16+j16) • (16+j16) + (64+j64)

	Умножитель + Аккумулятор + дополнительные инструкции, фиксированная точка
1	**MACXB4 T.Q,S.Q,ACn.D (2)** (AC (n+1) + j AC(n)) + = (TBX_3 • (conj) SBX_3 + TBX_2 • (conj) SX_2 + TX_1 • (conj) SX_1 + TX_0 • (conj) SX_0) , n=0,2,4,6 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)
2	**MACL2 T.D,S.D,ACn.D (2)** AC(n) + = T_1 • S_1 + T_0 • S_0 , n=0,1,2,3,4,5,6,7 Два умножения с суммированием и аккумуляцией, целые, 32 • 32 → 64 + 64
3	**ACSG8 T,S.Q (2)** AC(i) + = $(-1)^{T[i]}$ • s_i , i = 0:7, T[i] – биты t, n=0 Восемь накоплений со знаками, целые, 16 + 64
4	**ACSG4 T,S.D,ACn.D (2)** AC(n+i) + = $(-1)^{T[i]}$ • s_i , i = 0:3 , T[i] – биты t, n=0,4 Четыре накопления со знаками, целые, 16 + 64
5	**ACSG2 T,S.L,ACn.D (2)** AC(n+i) + = $(-1)^{T[i]}$ • s_i , i = 0:1 , T[i] – биты t, n=0,2,4,6 Два накопления со знаками, целые, 16 + 64
6	**ACSG1 T,S,ACn.D (2)** AC(n) + = $(-1)^{T[0]}$ • s , T[0] – бит t, n=0,1,2,3,4,5,6,7 Накопление со знаком, целое, 16 + 64

	Сдвигатель, 64-разрядные операнды, фиксированная точка
	а) сдвиги
1	**ASRDi #5, S.D,D.D** или **ASRD T, S.D,D.D (2)** Арифметический сдвиг 64-разрядного операнда S вправо на T разрядов: D = S>>T Параметр сдвига задается либо непосредственно 5-разрядным числом, либо 16-разрядным числом в регистре T
2	**ASLDi #5, S.D,D.D** или **ASLD T, S.D,D.D (2)** Арифметический сдвиг 64-разрядного операнда S влево на T разрядов: D = S<<T Параметр сдвига задается либо непосредственно 5-разрядным числом, либо 16-разрядным числом в регистре T
3	**LSRDi #5 S.D,D.D** или **LSRD T, S.D,D.D (2)** Логический сдвиг 64-разрядного операнда S вправо на T разрядов: D = S>>T Параметр сдвига задается либо непосредственно 5-разрядным числом, либо 16-разрядным числом в регистре T
4	**LSLDi #5 S.D,D.D** или **LSLD T, S.D,D.D (2)** Логический сдвиг 64-разрядного операнда S влево на T разрядов: D = S<<T Параметр сдвига задается либо непосредственно 5-разрядным числом, либо 16-разрядным числом в регистре T
5	**RORD S.D,D.D (2)** Круговой сдвиг 64-разрядного операнда S вправо на 1 разряд: (D , C) = C → S → C
6	**ROLD S.D,D.D (2)** Круговой сдвиг 64-разрядного операнда S влево на 1 разряд: (D , C) = C ← S ← C
7	**ASRXL T, S.D,D.D (2)** Арифметический сдвиг компонентов комплексного операнда S вправо: D1 + jD0 = (S1>>T) + j(S0>>T) Параметр сдвига задаётся 16-разрядным числом в регистре T
8	**ASLXL T, S.D,D.D (2)** Арифметический сдвиг компонентов комплексного операнда S влево: D1 + jD0 = (S1<<T) + j(S0<<T) Параметр сдвига задаётся 16-разрядным числом в регистре T
9	**LSRXL T.L, S.D,D.D (2)** Логический сдвиг компонентов комплексного операнда S вправо: D1 + jD0 = (S1>>t1) + j(S0>>t0) Параметры сдвига задаются парой 16-разрядных чисел в регистре T.L = (t1 , t0)
10	**LSLXL T.L, S.D,D.D (2)** Логический сдвиг компонентов комплексного операнда S влево: D1 + jD0 = (S1<<t1) + j(S0<<t0) Параметры сдвига задаются парой 16-разрядных чисел в регистре T.L = (t1 , t0)

	б) вспомогательные операции
2	**SMBD S.D,D (2)** Подсчет количества единичных разрядов в 64-разрядном операнде S
	в) поддержка E-формата плавающей точки с 64-разрядной мантиссой При реализации E-формата используются аппаратный служебный бит **E** – указатель денормализуемой мантиссы
1	**ASRDE T,S.D,D.D (2)** 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.

	АЛУ, фиксированная точка
	Дополнительные режимы работы: масштабирование, сатурация. **Масштабирование**: деление результата на 1/2/4/8 (8 - для **A81**, **A42**) посредством сдвига вправо на 0/1/2/3 (3 - для **A81**, **A42**) 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.
1	**A8 T.Q,S.Q,D.Q (2)** $D = (s_7 + t_7 , s_6 + t_6 , s_5 + t_5 , s_4 + t_4 , s_3 + t_3 , s_2 + t_2 , s_1 + t_1 , s_0 + t_0)$ Восемь сложений, целые, 16 – разрядные, масштабирование, сатурация
2	**S8 T.Q,S.Q,D.Q (2)** $D = (s_7 - t_7 , s_6 - t_6 , s_5 - t_5 , s_4 - t_4 , s_3 - t_3 , s_2 - t_2 , s_1 - t_1 , s_0 - t_0)$ Восемь вычитаний, целые, 16 – разрядные, масштабирование, сатурация
3	**MS8 T.Q,S.Q,D.Q (2)** $D = (\lvert s_7 - t_7 \rvert , \lvert s_6 - t_6 \rvert , \lvert s_5 - t_5 \rvert , \lvert s_4 - t_4 \rvert , \lvert s_3 - t_3 \rvert , \lvert s_2 - t_2 \rvert , \lvert s_1 - t_1 \rvert , \lvert s_0 - t_0 \rvert)$ Модули от восьми вычитаний, целые, 16 – разрядные, масштабирование, сатурация
4	**A4 T.D,S.D,D.D (2)** $D = (s_3 + t_3 , s_2 + t_2 , s_1 + t_1 , s_0 + t_0)$ Четыре сложения, целые, 16 – разрядные, масштабирование, сатурация
5	**S4 T.D,S.D,D.D (2)** $D = (s_3 - t_3 , s_2 - t_2 , s_1 - t_1 , s_0 - t_0)$

	Четыре вычитания, целые, 16 – разрядные, масштабирование, сатурация
6	**MS4 T.D,S.D,D.D (2)** $D = (\lvert s_3 - t_3 \rvert , \lvert s_2 - t_2 \rvert , \lvert s_1 - t_1 \rvert , \lvert s_0 - t_0 \rvert)$ Модули от четырех вычитаний, целые, 16 – разрядные, масштабирование, сатурация
7	**MS2 T.L,S.L,D.L (2)** $D = (\lvert s_1 - t_1 \rvert , \lvert s_0 - t_0 \rvert)$ Модули от двух вычитаний, целые, 16 – разрядные, масштабирование, сатурация
8	**ALL4 T.Q,S.Q,D.Q (2)** $D = (T_3 + S_3 , T_2 + S_2 , T_1 + S_1 , T_0 + S_0)$ Четыре сложения 32-разрядные, целые, масштабирование, сатурация
9	**ALL2 T.D,S.D,D.D (2)** $D = (T_1 + S_1 , T_0 + S_0)$ Два сложения 32-разрядные, целые, масштабирование, сатурация
10	**AL4 T.D,S.Q,D.Q (2)** $D = (t_3 + S_3 , t_2 + S_2 , t_1 + S_1 , t_0 + S_0)$ Четыре сложения 16-разрядных t_i и 32-разрядных S_i , целые, 16+32 → 32, масштабирование, сатурация
11	**AL2 T.L,S.D,D.D (2)** $D = (t_1 + S_1 , t_0 + S_0)$ Два сложения 16-разрядных t_i и 32-разрядных S_i , целые, 16+32 → 32, масштабирование, сатурация
12	**A81 S.Q,D (2)** $D = (s_7 + s_6 + s_5 + s_4 + s_3 + s_2 + s_1 + s_0)$ Сложение по 8, целые, 16-разрядное, масштабирование, масштабирование, сатурация
13	**A42 S.Q,D.L (2)** $D = (s_7 + s_6 + s_5 + s_4 , s_3 + s_2 + s_1 + s_0)$ Два сложения по 4, целые, 16-разрядные, масштабирование, масштабирование, сатурация
14	**A24 S.Q,D.D (2)** $D = (s_7 + s_6 , s_5 + s_4 , s_3 + s_2 , s_1 + s_0)$ Четыре сложения по 2, целые, 16-разрядные, масштабирование, сатурация
15	**ALLFT41 S.Q,D (2)** $D = (S_3 + S_2 + S_1 + S_0)$ Сложение по 4, целое 32-разрядное, преобразование формата дробное с округлением 32 → 16, сатурация
16	**ALLFT22 S.Q,D.L (2)** $D = (d_1 , d_0) , d_1 = (S_3 + S_2) , \ d_0 = (S_1 + S_0)$ Два сложения по 2, целые 32-разрядные, преобразование формата дробное с округлением 32 → 16, сатурация

17	**ASX2 T.D,S.D,D.Q (2)** $T = (TX1 , TX0) , S = (SX1 , SX0) , D = (SX_1 – TX_1 , SX_1 + TX_1 , SX_0 – TX_0 , SX_0 + TX_0)$ или: $T = (t_3 , t_2 , t_1 , t_0) , S = (s_3 , s_2 , s_1 , s_0) , D = (d_7 , d_6 , d_5 , d_4 , d_3 , d_2 , d_1 , d_0) ,$ $d_7 = s_3 - t_3 , d_6 = s_2 - t_2 , d_5 = s_3 + t_3 , d_4 = s_2 + t_2 , d_3 = s_1 - t_1 , d_2 = s_0 - t_0 , d_1 = s_1 + t_1 , d_0 = s_0 + t_0$ Два сложения и вычитания комплексные, целые, 16 – разрядные, масштабирование, сатурация
18	**ASXS2 T.D,S.D,D.Q (2)** $D = (SX_1 + j TX_1 , SX_0 + j TX_0 , SX_1 – j TX_1 , SX_0 – j TX_0)$ Два сложения и вычитания комплексные, перестановка в TX, целые, 16 – разрядные, масштабирование, сатурация
19	**ASXS T.L,S.L,D.D (2)** $D = (SX +j TX , SX – j TX)$ Сложение и вычитание комплексные, перестановка в TX, целые, 16 – разрядные, масштабирование, сатурация
20	**RA8 T.Q,S.Q,D.Q (2)** $D = (d_7 , d_6 , d_5 , d_4 , d_3 , d_2 , d_1 , d_0), d_0 += s_0 – t_0 , d_1 = d_0 + s_1 – t_1 , d_2 = d_1 + s_2 – t_2 , d_3 = d_2 + s_3 – t_3 ,$ $d_4 = d_3 + s_4 – t_4 , d_5 = d_4 + s_5 – t_5 , d_6 = d_5 + s_6 – t_6 , d_7 = d_6 + s_7 – t_7$ Восемь скользящих сумм, целые, 16 – разрядные
21	**RA4 T.D,S.D,D.D (2)** $D = (d_3 , d_2 , d_1 , d_0), d_0 += s_0 – t_0 , d_1 = d_0 + s_1 – t_1 , d_2 = d_1 + s_2 – t_2 , d_3 = d_2 + s_3 – t_3$ Четыре скользящие суммы, целые, 16 – разрядные
22	**SGA8 T.Q,S.Q,D.Q (2)** $D = (d_i + (-1)^{T[i]} • s_i , , , , , , , ,),$ i = 7,6,5,4,3,2,1,0 – номер разряда операнда T; Восемь знаковых сумм, целые, 16 – разрядные, масштабирование, сатурация
23	**SGA4 T.D,S.D,D.D (2)** $D = (d_i + (-1)^{T[i]} • s_i , , ,),$ i = 3,2,1,0 – номер разряда операнда T; Четыре знаковых суммы, целые, 16 – разрядные, масштабирование, сатурация
24	**A8s T.Q,S.Q,D.Q (2)** $D = (s_7 + t_7 , s_6 + t_6 , s_5 + t_5 , s_4 + t_4 , s_3 + t_3 , s_2 + t_2 , s_1 + t_1 , s_0 + t_0)$ Восемь сложений, целые, 16 – разрядные, обязательное масштабирование, сатурация
25	**S8s T.Q,S.Q,D.Q (2)** $D = (s_7 – t_7 , s_6 – t_6 , s_5 – t_5 , s_4 – t_4 , s_3 – t_3 , s_2 – t_2 , s_1 – t_1 , s_0 – t_0)$ Восемь вычитаний, целые, 16 – разрядные, обязательное масштабирование, сатурация
26	**BF4 T.D,S.D,D.Q (2)** $T = (TX1 , TX0) , S = (SX1 , SX0) , D = (SX_1 + j TX_1 , SX_0 - TX_0 , SX_1 - j TX_1 , SX_0 + TX_0)$ или: $T = (t_3 , t_2 , t_1 , t_0) , S = (s_3 , s_2 , s_1 , s_0) , D = (d_7 , d_6 , d_5 , d_4 , d_3 , d_2 , d_1 , d_0) ,$ $d_7 = s_3 - t_2 , d_6 = s_2 + t_3 , d_5 = s_1 - t_1 , d_4 = s_0 - t_0 , d_3 = s_3 + t_2 , d_2 = s_2 - t_3 , d_1 = s_1 + t_1 , d_0 = s_0 + t_0$ Базовая операция FFT-4, формат целый (16+j16), масштабирование, сатурация

27	**BIF4 T.D,S.D,D.Q (2)** T = (TX1 , TX0) , S = (SX1 , SX0) , D = (SX_1 - j TX_1 , SX_0 - TX_0 , SX_1 + j TX_1 , SX_0 + TX_0) или: T = (t_3 , t_2 , t_1 , t_0) , S = (s_3 , s_2 , s_1 , s_0) , D = (d_7 , d_6 , d_5 , d_4 , d_3 , d_2 , d_1 , d_0) , $d_7 = s_3 + t_2$, $d_6 = s_2 - t_3$, $d_5 = s_1 - t_1$, $d_4 = s_0 - t_0$, $d_3 = s_3 - t_2$, $d_2 = s_2 + t_3$, $d_1 = s_1 + t_1$, $d_0 = s_0 + t_0$ Базовая операция IFFT-4, формат целый (16+j16), масштабирование, сатурация
28	**AXJ4 T.Q,S.Q,D.Q (2)** D = (SX_3 + jTX_3 , SX_2 + jTX_2 , SX_1 + jTX_1 , SX_0 + jTX_0) , или D = (s_7 - t_6 , s_6 + t_7 , s_5 - t_4 , s_4 + t_5 , s_3 - t_2 , s_2 + t_3 , s_1 - t_0 , s_0 + t_1) Четыре комплексных сложения с предварительным умножением операндов TX на j , целые, 16 – разрядные, масштабирование, сатурация
29	**SXJ4 T.Q,S.Q,D.Q (2)** D = (SX_3 - jTX_3 , SX_2 - jTX_2 , SX_1 - jTX_1 , SX_0 - jTX_0) , или D = (s_7 + t_6 , s_6 - t_7 , s_5 + t_4 , s_4 - t_5 , s_3 + t_2 , s_2 - t_3 , s_1 + t_0 , s_0 - t_1) Четыре комплексных сложения с предварительным умножением операндов TX на j , целые, 16 – разрядные, масштабирование, сатурация
	АЛУ , Сравнения, фиксированная точка
1	**CMPN4 T.D,S.D,D.D (1)** 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
2	**CMPZ4 T.D,S.D,D.D (1)** 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
3	**CMPN8 T.Q,S.Q,D.Q (1)** 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
4	**CMPZ8 T.Q,S.Q,D.Q (1)** 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
5	**CMPNL2 T.D,S.D,D.D (1)** 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
6	**CMPZL2 T.D,S.D,D.D (1)** 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

7	**CMPNL4 T.Q,S.Q,D.Q (1)** 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
8	**CMPZL4 T.Q,S.Q,D.Q (1)** 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
9	**MAX4 T,S.D,D.D (2)** D = (d_m , d_2 , d_1 , d_n), $d_m = max(d_3 , s_3 , s_2 , s_1 , s_0)$, $d_n = t + n_s$ или d_0. Поиск максимума и его номера, целые 16-разрядные значения и номер. Если $s_m = max (s_3 , s_2 , s_1 , s_0) \geq d_3$, то: $d_m = s_m$ и d_n = t + 3:0 (n_s , локальный номер s_m) Иначе: d_3 и d_0 не изменяются.
10	**MAXL2 T,S.D,D.D (2)** D = (D_m , d_1 , d_n), $D_m = max(D_1 , S_1 , S_0)$, $d_n = t + n_s$ или d_0. Поиск максимума и его номера, целые 32-разрядные значения и 16-разрядный номер. Если $S_m = max (S_1 , S_0) \geq D_1$, то: $D_m = S_m$ и d_n = t + 1:0 (n_s , локальный номер S_m) Иначе: D_1 и d_0 не изменяются.
11	**MAX8 T,S.Q,D.D (2)** D = (d_m , d_2 , d_1 , d_n), $d_m = max(d_3 , s_i , , , , , , ,)$, i = 7:0, $d_n = t + n_s$ или d_0. Поиск максимума и его номера, целые 16-разрядные значения и 16-разрядный номер. Если $s_m = max (s_i , , , , , , ,) \geq d_3$, то: $d_m = s_m$ и d_n = t + 7:0 (n_s , локальный номер s_m) Иначе: d_3 и d_0 не изменяются.
12	**MAXL4 T,S.Q,D.D (2)** D = (D_m , d_1 , d_n), $D_m = max(d_3 , S_i , , ,)$, i = 3:0, $d_n = t + n_s$ или d_0. Поиск максимума и его номера, целые 32-разрядные значения и 16-разрядный номер. Если $S_m = max (S_i , , , , , , ,) \geq D_1$, то: $D_m = S_m$ и d_n = t + 3:0 (n_s , локальный номер S_m) Иначе: D_1 и d_0 не изменяются.
13	**MIN4 T,S.D,D.D (2)** D = (d_m , d_2 , d_1 , d_n), $d_m = min(d_3 , s_3 , s_2 , s_1 , s_0)$, $d_n = t + n_s$ или d_0. Поиск минимума и его номера, целые 16-разрядные значения и номер. Если $s_m = min (s_3 , s_2 , s_1 , s_0) \leq d_3$, то: $d_m = s_m$ и d_n = t + 3:0 (n_s , локальный номер s_m) Иначе: d_3 и d_0 не изменяются.
14	**MINL2 T,S.D,D.D (2)** D = (D_m , d_1 , d_n), $D_m = min(D_1 , S_1 , S_0)$, $d_n = t + n_s$ или d_0. Поиск минимума и его номера, целые 32-разрядные значения и 16-разрядный номер. Если $S_m = min (S_1 , S_0) \leq D_1$, то: $D_m = S_m$ и d_n = t + 1:0 (n_s , локальный номер S_m)

	Иначе: D_1 и d_0 не изменяются.
15	**MIN8 T,S.Q,D.D (2)** $D = (d_m, d_2, d_1, d_n)$, $d_m = \min(d_3, s_i, , , , , , ,)$, i = 7:0, $d_n = t + n_s$ или d_0. Поиск минимума и его номера, целые 16-разрядные значения и 16-разрядный номер. Если $s_m = \min(s_i, , , , , , ,) \le d_3$, то: $d_m = s_m$ и $d_n = t + 7:0$ (n_s, локальный номер s_m) Иначе: d_3 и d_0 не изменяются.
16	**MINL4 T,S.Q,D.D (2)** $D = (D_m, d_1, d_n)$, $D_m = \min(d_3, S_i, , ,)$, i = 3:0, $d_n = t + n_s$ или d_0. Поиск минимума и его номера, целые 32-разрядные значения и 16-разрядный номер. Если $S_m = \min(S_i, , ,) \le D_1$, то: $D_m = S_m$ и $d_n = t + 3:0$ (n_s, локальный номер S_m) Иначе: D_1 и d_0 не изменяются.

	АЛУ, 64-разрядные операнды, фиксированная точка
	а) логические операции
1	**ANDD T.D,S.D,D.D (1)** Поразрядные логические «И» над 64-разрядными операндами: Dn = (Tn)&(Sn), n=0:63
2	**ANDCD T.D,S.D,D.D (1)** Поразрядные логические «И» над 64-разрядными операндами с инверсией T: Dn = (~Tn)&(Sn), n=0:63
3	**ANDID T.D,S.D,D.D (1)** Поразрядные логические «И» над 64-разрядными операндами с инверсией результатов: Dn = ~((Tn)&(Sn)), n=0:63
4	**ORD T.D,S.D,D.D (1)** Поразрядные логические «ИЛИ» над 64-разрядными операндами: Dn = (Tn)|(Sn), n=0:63
5	**ORCD T.D,S.D,D.D (1)** Поразрядные логические «ИЛИ» над 64-разрядными операндами с инверсией T: Dn = (~Tn)|(Sn), n=0:63
6	**ORID T.D,S.D,D.D (1)** Поразрядные логические «ИЛИ» над 64-разрядными операндами с инверсией результатов: Dn = ~((Tn)|(Sn)), n=0:63
7	**EORD T.D,S.D,D.D (1)** Поразрядные логические «ИСКЛЮЧАЮЩИЕ ИЛИ» над 64-разрядными операндами: Dn = (Tn)^(Sn), n=0:63
8	**NOTD S.D,D.D (1)** Поразрядные логические «НЕ» над 64-разрядным операндом: Dn =~(Sn), n=0:63

9	**INSD T.D,S.D,D.D (1)** Поразрядное объединение двух 64-разрядных операндов T и D по маске S: Dn =((Dn)&(~(Sn))) | ((Dn)&(Sn)), n=0:63
10	**BTSTDi #5,S.D** или **BTSTD T,S.D (1)** Запись n-го разряда 64-разрядного операнда S в разряд признака C: C=Sn Номер разряда n задается либо непосредственно 5-разрядным числом, либо 16-разрядным числом в регистре T
	б) арифметические операции
1	**CLRD D.D** Сброс в ноль разрядов 64-разрядного регистра
2	**ADDD T.D,S.D,D.D (2)** Целое сложение двух 64-разрядных операндов: D = S + T
3	**SUBD T.D,S.D,D.D (2)** Целое вычитание двух 64-разрядных операндов: D = S - T
4	**ADCD T.D,S.D,D.D (2)** Целое сложение двух 64-разрядных операндов с добавлением признака переноса: D = S + T + C
5	**SBCD T.D,S.D,D.D (2)** Целое вычитание двух 64-разрядных операндов с вычитанием признака переноса: D = S − T − (~C)
6	**ADDLD T.L,S.D,D.D (2)** Целое знаковое сложение 32-разрядного операнда T и 64-разрядного операнда S с образованием 64-разрядного результата: D = S + T
	в) вспомогательные операции
1	**TRD S.D,D.D (2)** Пересылка 64-разрядного операнда: D = S
2	**PDND S.D,D (2)** Определение параметра денормализации 64-разрядного операнда S
3	**PDNXL S.D,D (2)** Определение параметра денормализации комплексного операнда с 32-разрядными компонентами (S1 + jS0)
	г) поддержка E-формата плавающей точки с 64-разрядной мантиссой При реализации E-формата используется аппаратный служебный бит **E** – указатель денормализуемой мантиссы
1	**NORVD S.D,D.D (2)** 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)
2	**PDNDE S.D,D.L (2)** Определение параметра денормализации 64-разрядной мантиссы E-формата Если S ≠ 0, то параметр денормализации записывается в старшее полуслово результата d1; Если S = 0, то d1=0 и d0=0
3	**NEGDE S.D,D.D (2)** Изменение знака мантиссы S При возникновении переполнения: записывается результат D=0,5 (дробный формат) и устанавливается признак переполнения V = 1

	Специализированный блок пересылок	
n	**Пересылки: TRS n, S,D**	**Назначение**
0	**TRS0 S.Q,D.Q (1)** D = S	Пересылка 128-разрядного операнда.
1:3	**TRSn T.D,S.D,D.D (2)** D = (S>>16•n) | (T<<(64 -16•n)) , n = 1,2,3	Сдвиг двух 64-разрядных операндов на n = 1,2,3 16-разрядных слова При T=S: круговой сдвиг 64-разрядного операнда на n = 1,2,3 16-разрядных слова
4	**TRS4 S.D,D.Q (1)** D = (d_i , , , , , , , ,), i = 7,6,5,4,3,2,1,0; d_i = S[7:0+8•i] <<8	Преобразование восьми 8-разрядных операндов в 16—разрядные, дробный формат
5	**TRS5 S.D,D.Q (1)** D = (d_i , , , , , , , ,), i = 7,6,5,4,3,2,1,0; D_i = (S[7+8•i] x 8 bits , S[7:0+8•i]	Преобразование восьми 8-разрядных операндов в 16—разрядные, целый формат, расширение знака
6	**TRS6 S.D,D.Q (1)** D = (d_i , , , , , , , ,), i = 7,6,5,4,3,2,1,0; D_i = (0 x 8 bits , S[7:0+8•i]	Преобразование восьми 8-разрядных операндов в 16—разрядные, целый формат, без знака (старшие 8 бит – нулевые)
7	**TRS7 S.Q,D.D (1)** D[7:0+8•i] = s_i , i=7,6,5,4,3,2,1,0	Преобразование восьми 16-разрядных операндов в 8—разрядные, дробный формат, округление, сатурация

8	**TRS8 S.Q,D.D (2)** D[7:0+8•i] = s_i , i=7,6,5,4,3,2,1,0	Преобразование восьми 16-разрядных операндов в 8—разрядные, целый формат, без знака, сатурация
9	**TRS9 S.D,D.D (1)** D = (s_0 , s_1 , s_2 , s_3)	Перестановка четырех 16-разрядных операндов инверсная
10	**TRS10 S.D,D.D (1)** D = (s_3 , s_1 , s_2 , s_0)	Перестановка четырех 16-разрядных операндов двоично – инверсная
11	**TRS11 T.D,S.D,D.Q (1)** D = (t_3 , t_2 , s_3 , s_2 , t_1 , t_0 , s_1 , s_0)	Перестановка четырех 16-разрядных операндов матричная
12	**TRS12 T.D,S.D,D.Q (1)** D = (t_3 , s_3 , t_2 , s_2 , t_1 , s_1 , t_0 , s_0)	Вложение двух векторов из четырех 16-разрядных операндов каждый
13	**TRS13 T.D,S.D,D.Q (1)** D = (t_3 , t_1 , s_3 , s_1 , t_2 , t_0 , s_2 , s_0)	Раскладка восьми 16-разрядных операндов на два массива из четырех четных и четырех нечетных элементов
14	**TRS14 T.D,S.D,D.L (2)** D = (d_1 , d_0), (t_2 , t_1 , t_0) → d_1 , (s_2 , s_1 , s_0) → d_0	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.
15	**TRS15 S.Q,D.D (2)** D = (d_3 , d_2 , d_1 , d_0) , d_i = round(S_i) , i = 3,2,1,0 S = (S_3 , S_2 , S_1 , S_0)	Преобразование четырех 32-разрядных операндов в 16-разрядные, дробный формат, округление, сатурация.
16	**TRS16 S.Q,D.D (2)** D = (d_3 , d_2 , d_1 , d_0) , d_i = S_i , i = 3,2,1,0 S = (S_3 , S_2 , S_1 , S_0)	Преобразование четырех 32-разрядных операндов в 16-разрядные, целый формат, сатурация.
17	**TRS17 S.Q,D.D (2)** D = (d_3 , d_2 , d_1 , d_0) , d_i = S_i , i = 3,2,1,0 S = (S_3 , S_2 , S_1 , S_0)	Преобразование четырех 32-разрядных операндов в 16-разрядные, целый формат, без знака, сатурация.
18	**TRS18 T,S,D.L (1)** D = (bd_i , , ,) , i = 31:0, bd_{2i+1} = bt_i, bd_{2i} = bs_i , i = 15:0; bt_i , bs_i, bd_i – i-е разряды операндов T, S и D, соответственно (i = 0 – младшие разряды)	Поразрядное вложение двух 16-разрядных операндов (T, S) с образованием 32-разрядного операнда D. Разряды S становятся четными разрядами D, T – нечетными.
19	**TRS19 S.Q,D.D (2)** D[7:0+8•i] = s_i , i=7,6,5,4,3,2,1,0	Преобразование восьми 16-разрядных операндов в 8—разрядные, целый формат, сатурация