Процессор TMS 320C6000

Это руководство описывает архитектуру, конвейер, набор команд и прерывания для семейства цифровых сигнальных процессоров TMS 320C6000.

Введение

Платформа TMS 320C6000 является основой для целого семейства цифровых сигнальных процессоров. Это семейство составляют процессоры TMS 320C62x, TMS 320C64x, TMS 320C67x. Все три этих процессора используют VelociTI архитектуру, высокопроизводительную VLIW (очень длинное командное слово) архитектуру, что позволяет эффективно использовать их в многоканальных и многофункциональных приложениях.

Семейство процессоров TMS 320 включает в себя процессоры, выполняющие операции над числами как с плавающей, так и с фиксированной точкой, многопроцессорные процессоры. Первый процессор этого семейства - TMS 32010 был предназначен для проведения операций над числами с фиксированной точкой, он был представлен в 1982 году.

Рассматриваемые далее процессоры серии TMS 320C6x имеют эффективный C - компилятор и возможность выполнения до 6000 операций в секунду, что позволяет использовать их для реализации множества технических задач, таких как:
- модемные пулы
- локальные беспроводные станции
- модемы
- многоканальные телефонные системы

Эти процессоры могут использоваться для решения таких новейших задач, как:
- охранные системы с распознаванием отпечатков пальцев/рук
- круиз-контроль с системами GPS-навигации
- 3-D графика
- распознавание речи
- атмосферное моделирование

Особенности и возможности процессоров TMS 320C62x/64x/67x

Процессоры этой серии могут выполнять до восьми 32 – разрядных команд за один цикл. Ядро процессоров C62x/C67x включает в себя 32 32-х разрядных регистра общего назначения и восемь функциональных модулей. Ядро процессора C64x включает в себя 64 32-х разрядных регистров общего назначения и восемь функциональных модулей.

Функциональные модули содержат:

- два мультипликатора (множителя)
- шесть арифметико-логических устройств (АЛУ)

Процессоры имеют полный набор усовершенствованных инструментов, включающих эффективный C-компилятор, специально оптимизированный для наиболее легкого программирования и управления процессором, Windows – отладчик, позволяющий

контролировать выполнение программы. Также возможно использование платы эмулятора, совместимой с интерфейсом TI XDS510.

Особенности процессоров:

- Новое VLIW ядро с восемью функциональными модулями
- Выполнение до восьми команд за один цикл
- Уменьшение размера команды, уменьшение времени выборки и уменьшение потребляемой мощности
- Поддержка 8/16/32 – разрядного формата данных, что позволяет наиболее эффективно распределять память для различных приложений
- 40- разрядная арифметика для повышения точности вычислений
- Насыщенность и нормализация, обеспечивающие поддержку ключевых арифметических действий

Процессоры C67x имеют ряд характерных отличий:

- Аппаратная поддержка с одинарной (32 разряда) и двойной точностью (64 разряда) для чисел с плавающей запятой
- 32*32 –разрядный мультипликатор с 32х или 64х – разрядным результатом

Особенности процессоров C64x:

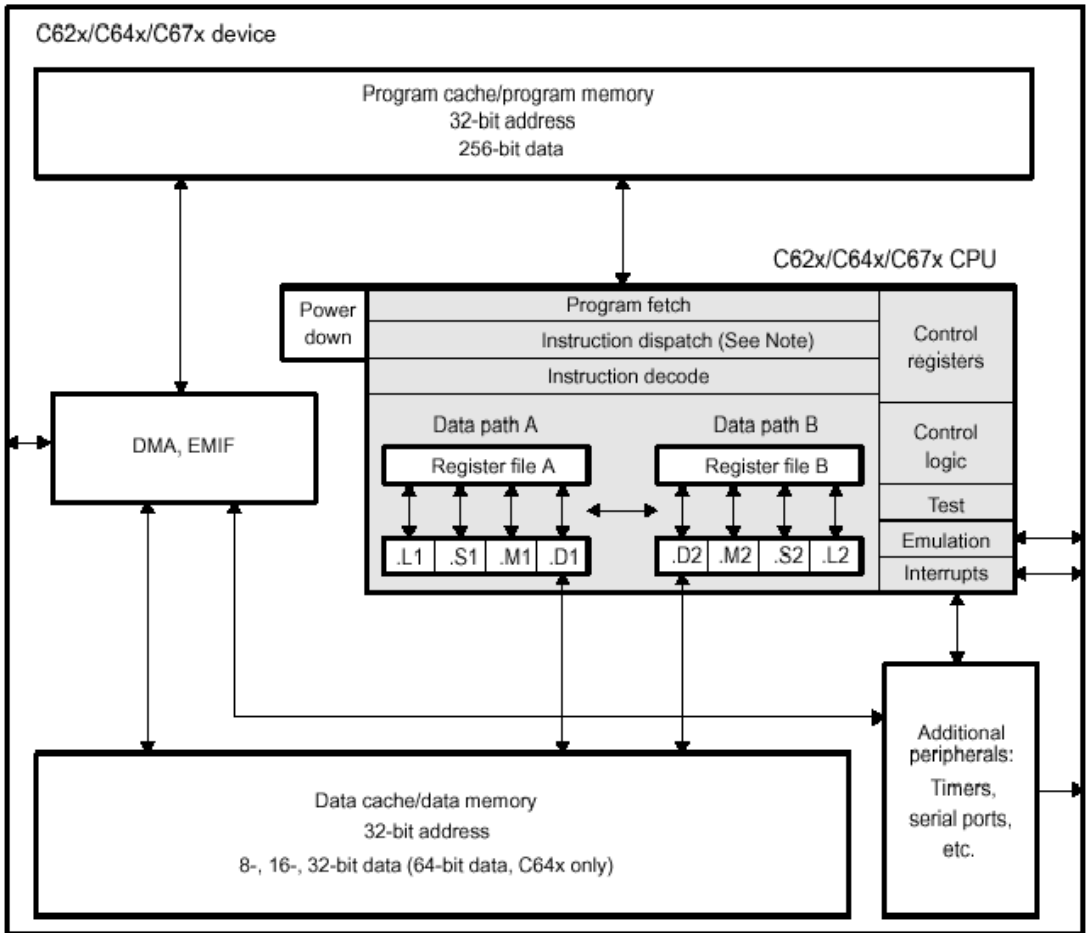
- Каждый мультипликатор способен выполнять два 16*16 - разрядных или четыре 8*8 – разрядных умножения за один цикл
- Учетверенный 8 – разрядный или двойной 16 – разрядный набор команд
- Возможность добавления к адресу специального кода при ошибочном коде
- Поддержка 32 – разрядных и 64 – разрядных слов в памяти

Архитектура процессоров TMS 320C62x/C64x/C67x

Структурная схема процессора представлена на рисунке 1.
Процессоры этой серии имеют программную память, которая в некоторых случаях также может быть использована как кэш – память. Процессоры могут иметь различную по размеру память данных.

Рисунок 1

Figure 1–1. TMS320C62x/C64x/C67x Block Diagram



Центральный процессор (CPU)

Центральный процессор, обозначенный на рисунке серым цветом одинаков для всех процессоров этой серии. Он включает в себя:

- Модуль получения программы
- Модуль распаковки команды (при использовании упаковки)
- Модуль раскодировки команды
- Две шины данных, (data paths)каждая с четырьмя функциональными модулями
- 32 32х – разрядных регистра, 64 32х – разрядных регистра для процессоров С64
- Элементы управления регистрами и логикой

Модули получения программы, распаковки и раскодировки команды позволяют передавать до восьми 32х – разрядных команд в функциональные модули за один цикл. Команды попадают в одну из двух шин данных (А и В), а затем – в функциональные модули (2 по 4) и 16 32х – разрядных регистра общего назначения (32 32х – разрядных регистра для процессоров С64). Элементы управления регистрами и логикой позволяют контролировать и управлять различными операциями, выполняемыми процессором.

Внутренняя память (internal memory)

Данные процессоры имеют 32х битную адресную память. Организация памяти позволяет разместить память программ и память данных отдельно. При использовании внешней памяти это пространство унифицируется с большинством устройств как общая память.

Процессоры C62x/C67x имеют два 32-разрядных внутренних порта доступа к памяти данных. Процессоры C64x имеют два 64-разрядных порта доступа к памяти данных. Все процессоры этой серии имеют один порт доступа к внутренней программной памяти с приемником команд длиной 256 бит.

Возможности памяти и периферийных устройств

- Кэш программ
- 32x – битный внешний интерфейс памяти, поддерживающий SDRAM, SBSRAM, SRAM
- Контроллер DMA (прямого доступа к памяти) позволяет перемещать данные без вмешательства центрального процессора. Контроллер DMA имеет четыре программируемых канала и пятый дополнительный канал.
- Контроллер EDMA выполняет функции, аналогичные контроллеру DMA. Он имеет 16 программируемых каналов.
- Параллельный порт HPI осуществляет непосредственный доступ процессора к памяти. Процессор может принимать информацию как от внутренней, так и от внешней памяти. Кроме того, процессор имеет прямой доступ к памяти периферийных устройств.
- Заменой HPI может служить шина расширения. Расширение обеспечивают главный порт (host port) и порт ввода/вывода, которые могут сосуществовать в системе. Главный порт может работать в асинхронном slave –режиме, подобно работе HPI, или в синхронном master/slave – режиме. Это позволяет работать с различными протоколами передачи данных.
- Многоканальный последовательный буферный порт (McBSP – Multichannel Buffered Serial Port) основан на стандартном последовательном порту. В дополнение, он может буферизовать последовательные участки памяти с помощью DMA\EDMA контроллера.
- Процессоры серии C6000 имеют два 32 –битных таймера, выполняющих следующие функции: счет времени, генерация импульсов и прерываний, посылка синхронизирующих импульсов в DMA\EDMA контроллеры.

Шины данных и управление процессора (CPU Data paths and control)

Далее рассматриваются шины данных процессора и его регистровые файлы.
Компоненты этих элементов показаны на трех рисунках, следующих ниже.
Эти компоненты состоят из:

- Два регистровых файла общего назначения (A и B)
- Восемь функциональных модулей (.L1, .L2, .S1, .S2, .M1, .M2, .D1, .D2)
- Два пути загрузки данных из памяти (LD1 и LD2)
- Два пути сохранения данных в памяти (ST1 и ST2)
- Две адресных шины (DA1 и DA2)
- Две регистровых шины данных (1X и 2X)

Регистровые файлы общего назначения

В процессорах серии C6000 имеется два регистровых файла общего назначения, обозначаемых A и B. У процессоров C62x/C67x каждый из этих регистровых файлов

состоит из 16 32 – разрядных регистров (регистры A0-A15 для файла A и регистры B0-B15 для файла B). Регистровые файлы общего назначения могут быть использованы для хранения данных, адресных указателей, условных регистров. Регистровые файлы процессора C64x состоят из 32 32-разрядных регистров (регистры A0-A31 для файла A и B0-B31 для файла B).

Регистровые файлы общего назначения процессоров C62x/C67x поддерживают данные в формате от 16 бит до 40-битного для чисел с фиксированной точкой и 64-битного для чисел с плавающей точкой. Числа длиной более 32 бит, такие как 40-битные и 64-битные, содержатся в регистровых парах. Процессор C64x поддерживает 64-битные формат для чисел с фиксированной запятой (C64x непосредственно не поддерживает числа с плавающей запятой).

Всего существуют 16 возможных регистровых пар для 40-битных и 64-битных чисел у процессоров C62x/C67x и 32 возможные регистровые пары для процессора C64, эти пары показаны на рисунке 5.

Рисунок 5. Регистровые пары

Register Files		Applicable
A	B	Devices
A1:A0	B1:B0	C62x/C64x/C67x
A3:A2	B3:B2	
A5:A4	B5:B4	
A7:A6	B7:B6	
A9:A8	B9:B8	
A11:A10	B11:B10	
A13:A12	B13:B12	
A15:A14	B15:B14	
A17:A16	**B17:B16**	**C64x only**
A19:A18	**B19:B18**	
A21:A20	**B21:B20**	
A23:A22	**B23:B22**	
A25:A24	**B25:B24**	
A27:A26	**B27:B26**	
A29:A28	**B29:B28**	
A31:A30	**B31:B30**	

Функциональные модули процессоров

Восемь функциональных модулей процессоров C6000 можно разделить на четыре группы, так как для каждого модуля одной шины данных (A) существует идентичный в другой шине данных (B). Функциональные модули и выполняемые ими операции рассмотрены в таблице 1. Операции, выполняемые только процессором C64x, показаны жирным шрифтом.

Таблица 1. Функциональные модули

Функциональный модуль	Операции над числами с фиксированной точкой	Операции над числами с плавающей точкой
.L модуль (.L1, .L2)	32/40 –битные арифметические операции и операции сравнения 32 – битные логические операции	Арифметические операции Операции преобразования DP-SP, INT-DP, INT-SP

	Сдвиг байт **Упаковка/распаковка данных** **Генерация 5-битных констант** **Двойные 16-битные арифметические операции** **Четверные 8-битные арифметические опреации** **Двойные 16-битные min\max операции** **Четверные 8-битные min\max операции**	
.S модуль(.S1, .S2)	32-битные арифметические операции 32/40- битные операции сдвига 32-битные логические операции Операции перехода Генерация констант Регистровые перемещения (только .S2) **Сдвиг байт** **Упаковка/распаковка данных** **Двойные 16-битные операции сравнения** **Четверные 8-битные операции сравнения** **Двойные 16-битные операции сдвига** **Двойные 16-битные арифметические операции насыщения** **Четверные 8-битные арифметические операции насыщения**	Сравнение Операции возведения в квадрат и извлечения корня Операции с абсолютными значениями Операции преобразования SP-DP
.M модуль (.M1, .M2)	16*16 операции умножения **16*32 операции умножения** **Четверные 8*8 операции умножения** **Двойные 16*16 операции умножения** **Двойные 16*16 операции умножения с добавлением/вычитанием** **Четверные 8*8 операции умножения с добавлением** **Расширение бит** **Различные операции сдвига**	Операции умножения
.D модуль (.D1, .D2)	32-битное сложение, вычитание Загрузка и хранение с5-битными константами Загрузка и хранение с 15-битными константами **Загрузка и хранение двойных слов с 5-битными константами** **Загрузка и хранение неориентированных и двойных слов** **Генерация 5-битных констант**	Загрузка двойных слов с 5-битными константами

	32-битные логические операции	

Операции над числами с фиксированной точкой возможны на всех трех процессорах. Операции над числами с плавающей точкой и 32*32 – битное умножение чисел с фиксированной точкой возможны только у процессора C67x. Операции, выполняемые только процессором C64x, показаны жирным шрифтом.
Каждый функциональный модуль имеет свой собственный 32-битный порт записи в регистровом файле общего назначения. Все модули, обозначение которых заканчивается на 1(например, .L1), относятся к регистровому файлу A, все модули, обозначение которых заканчивается на 2 относятся к регистровому файлу B. Каждый функциональный модуль имеет два 32-битных порта чтения для текущих операндов src1 и src2. Для модулей (.L1, .L2, .S1, .S2) существуют добавочные 8-битные расширенные порты записи и чтения для 40 – битных длинных слов. Так как каждый модуль имеет собственный 32-разрядный порт записи, при выполнении 32-разрядных операций могут быть задействованы параллельно все восемь портов в каждом цикле.

Шины данных регистровых файлов

Каждый функциональный модуль может записывать данные в регистры и читать данные из регистра только посредством своей шины данных. Модули L1, S1, D1 и M1 записывают данные в регистровый файл A, а модули L2, S2, D2 и M2 записывают данные в регистровый файл B.

Шины данных памяти, загрузки и хранения **(Memory, Load and Store Paths)**

Процессор C62x имеет две 32-разрядные шины для загрузки данных из памяти в регистровый файл: LD1 для регистрового файла A и LD2 для регистрового файла B. Процессор C67x имеет еще по две шины на каждый регистр, для того, чтобы загружать две 32-битные величины в регистровый файл A и также две 32-битные величины в регистровый файл B. Здесь также имеются еще две шины ST1 и ST2 для сохранения регистровых значений в памяти из каждого регистрового файла.
Процессор C64x поддерживает чтение и сохранение двойных слов, он имеет четыре 32-разрядные шины для загрузки значений из памяти в регистровый файл. Он также имеет по две 32-разрядные шины для сохранения значений на каждый регистр.

Регистровый файл управления процессоров TMS320C6000 **(TMS320C6000 Control Register File)**

Один из модулей (.S2) может читать и записывать данные в регистровый файл контроля.
Таблица 2 показывает управляющие, регистры, содержащиеся в регистровом файле управления и имеет краткое описание каждого из них.

Таблица 2. Регистры управления процессоров 62x/67x и 64x.

Аббревиатура	Название регистра	Описание
AMR	Регистр формата адреса	Указывает на линейный или циклический формат адреса для каждого из 8ми регистров, также содержит размеры для циклической адресации

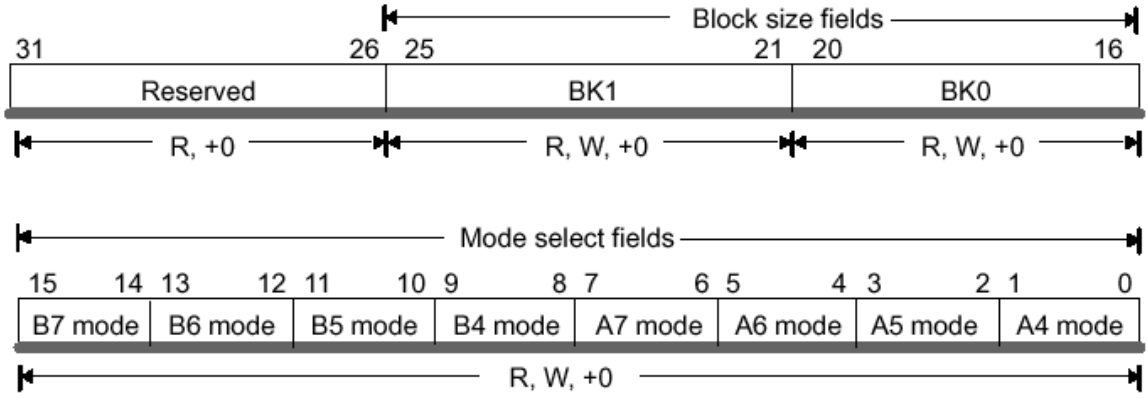
CSR	Регистр статуса управления	Содержит бит глобальных прерываний, бит управления КЭШем, и другие разнообразные биты управления и статуса
IFR	Регистр признака прерывания	Показывает статус прерываний
ISR	Регистр установки прерываний	Позволяет установить прерывание вручную
ICR	Регистр удаления прерываний	Позволяет вручную очистить прерывания
IER	Регистр активирования прерываний	Позволяет активировать/деактивировать некоторые прерывания
ISTP	Указатель прерываний сервисной таблицы	Указывает на начало сервисной таблицы прерываний
IRP	Указатель возвращения прерываний	Содержит адрес для использования возвращенных замаскированных прерываний
NRP	Регистр возвращения немаскированных прерываний	Содержит адрес для использования возвращенных незамаскированных прерываний
PCE1	Программный счетчик	

Регистр формата адреса (Addressing mode Register) (AMR)

Для каждого из восьми регистров (А4-А7, В4-Б7), которые могут быть представлены с помощью линейной или циклической адресации, этот регистр указывает формат адресации. 2х-битное поле для каждого регистра указывает на формат адресации: линейная (используется по умолчанию) или циклическая. При циклической адресации это поле также содержит размер блока для использования циклического буфера. Формат выбираемых полей и размеров блока показаны на рисунке 6.
Различные варианты форматов для регистра показаны в таблице 3.
В исходном состоянии регистр сбрасывается на 0.
Поля размера блока, ВК0 и ВК1, содержат 5-битные значения, используемые для расчета размеров блока, используемого в циклической адресации.

Размер блока=$2^{(N+1)}$
Где N – это 5-битное значение в ВК0 или ВК1.
Вычисления размеров блоков показаны в таблице 4.

Рисунок 6. Регистр формата адреса



Legend: R Readable by the **MVC** instruction
W Writeable by the MVC instruction
+0 Value is zero after reset

Таблица 3. Различные варианты форматов в регистре

Формат		Описание
0	0	Линейная адресация (используется по умолчанию)
0	1	Циклическая адресация с использованием поля BK0
1	0	Циклическая адресация с использованием поля BK1
1	1	Зарезервированная комбинация

Таблица 4. Вычисления размеров блока

N	Block Size	N	Block Size
00000	2	10000	131 072
00001	4	10001	262 144
00010	8	10010	524 288
00011	16	10011	1 048 576
00100	32	10100	2 097 152
00101	64	10101	4 194 304
00110	128	10110	8 388 608
00111	256	10111	16 777 216
01000	512	11000	33 554 432
01001	1 024	11001	67 108 864
01010	2 048	11010	134 217 728
01011	4 096	11011	268 435 456
01100	8 192	11100	536 870 912
01101	16 384	11101	1 073 741 824
01110	32 768	11110	2 147 483 648
01111	65 536	11111	4 294 967 296

Расширения регистрового файла управления у процессора C67x
Control Register File Extensions

Процессор C67x имеет три дополнительных регистра для поддержания операций над числами с плавающей точкой.

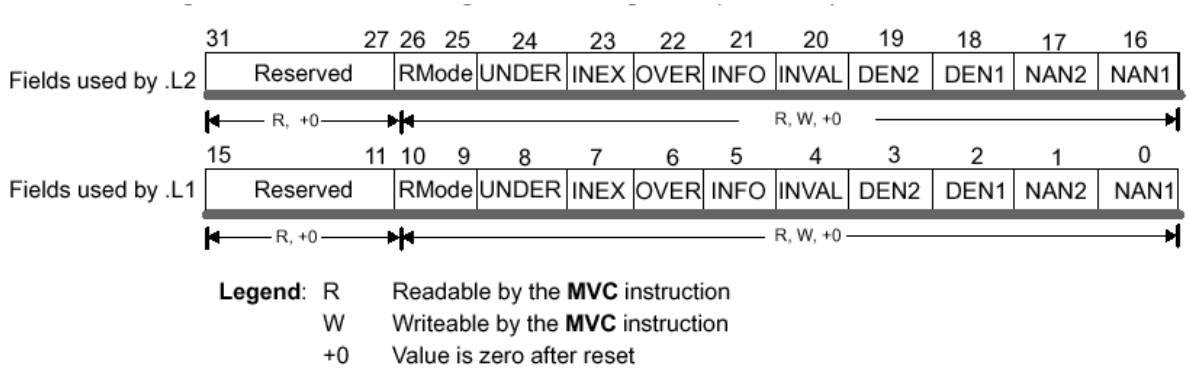
Таблица 5. Дополнительные регистры

Регистр		Описание
Аббревиатура	Название	
FADCR	Регистр сложения	Используется для .L модуля
FAUCR	Дополнительный регистр	Используется для .S модуля
FMCR	Регистр умножения	Используется для .M модуля

Регистр сложения
Floating-Point Adder Configuration Register (FADCR)

Этот регистр содержит поля, которые отвечают за операции опустошения/переполнения, округления, ненормализованные числа и недопустимые результаты в операциях над числами с плавающей запятой, используемых в функциональном модуле .L(.L1 и .L2). Схема регистра показана на рисунке 7.

Рисунок 7. Схема регистра FADCR



Функции полей этого регистра показаны в следующей таблице:

Таблица 6. Функции полей регистра FADCR

Позиция	Длина	Имя поля	Функция
31-27	5		Зарезервирован
26-25	2	Rmode .L2	Метка: 00 – Округление до ближайшего числа с плавающей точкой 01 – Округление до нуля (truncate) 10 - Округление до бесконечности (round up) 11 – Округление до отрицательной бесконечности (round down)
24	1	Under .L2	Устанавливается 1 при опустошении
23	1	Inex .L2	

22	1	Over .L2	Устанавливается 1 при переполнении
21	1	Info .L2	Устанавливается 1 при бесконечном результате
20	1	Inval .L2	
19	1	Den2 .L2	Src2 является денормализованным числом
18	1	Den1 .L2	Src1 является денормализованным числом
17	1	Nan2 .L2	Src2 является NaN
16	1	Nan1 .L2	Src1 является NaN
15-11	5		Функции данных полей аналогичны предыдущим, но относятся к функциональному модулю .L1
10-9	2	Rmode .L1	
8	1	Under .L1	
7	1	Inex .L1	
6	1	Over .L1	
5	1	Info .L1	
4	1	Inval .L1	
3	1	Den2 .L1	
2	1	Den1 .L1	
1	1	Nan2 .L1	
0	1	Nan1 .L1	

Дополнительный регистр

Floating-point Auxiliary Configuration Register

Этот регистр содержит поля, отвечающие за операции опустошения/переполнения, округления, ненормализованные числа и недопустимые результаты в операциях над числами с плавающей запятой, используемых в функциональном модуле .S(.S1 и .S2). Основные функции и устройство аналогичны регистру FADCR.

Регистр умножения. Floating-point Multiplier Configuration Register

Этот регистр содержит поля, отвечающие за операции опустошения/переполнения, округления, ненормализованные числа и недопустимые результаты в операциях над числами с плавающей запятой, используемых в функциональном модуле .M(.M1 и .M2). Основные функции и устройство аналогичны регистрам, рассмотренным ранее (FADCR и FAUCR).

Набор команд для проведения операций над числами с фиксированной точкой

Все три описываемых процессора содержат наборы команд для операций над числами с фиксированной точкой. Все команды, применимые у процессора C62x также применимы к процессорам C64x и C67x. Однако, так как процессор C67x предназначен в основном для операций с плавающей точкой, некоторые команды являются уникальными и подходят только для него. У процессора C64 также добавляются некоторые уникальные операции, свойственные только ему. Этот раздел описывает операции, выполняемые процессорами C62x, C64x и C67x. Описываются параллельные, условные операции, средства принуждения и форматы адресации.

Таблица 7. Команды и их выполнение

Обозначение	Значение
abs(x)	Абсолютное значение x
and	Поразрядное и
-a	Вычитание
+a	Сложение
$b_{y..z}$	Выборка битов y и z из строки b
cond	
creg	3-битное поле для условного регистра
cstn	n-битное константное поле
int	32-битное integer значение
lmb0(x)	Левый поиск 0 в x
lmb1(x)	Левый поиск 1 в x
long	40-битное integer значение
lsbn or LSBn	N последний значащий бит
msbn or MSBn	N наиболее значащий бит
nop	Нет операции
norm(x)	Левый поиск неизбыточного бита в x
not	Побитовое логическое дополнение
or	Побитовое или
op	
R	Любой регистр общего назначения
scstn	n-битное константное поле
sint	Значащее 32-битное integer значение
slong	Значащее 40-битное integer значение
-s	2s-комплементарное вычитание с приведением результата к нужному виду при переполнении
+s	2s- комплементарное сложение с приведением результата к нужному виду при переполнении
ucstn	n-битное беззнаковое константное поле
uint	Незначащее 32-битное integer значение
ulong	Незначащее 40-битное integer значение
X clear b,e	Очищение поля X, обозначенного с помощью начального (b) и конечного (e) бит
X ext l,r	Извлечение и знаковое заполнение поля в x, обозначенного через l (левое значение) и r(правое значение)
X extu l,r	Извлечение беззнакового поля в x, обозначенного через l и r
X set b,e	Установить все 1 в поле x, начинающимся битом b и заканчивающимся битом e
Xor	Побитовое или
→	Присваивание
+	Сложение
*	Умножение
-	Вычитание
<<	Левый сдвиг

>>s	Правый сдвиг с расширением знака
>>z	Правый сдвиг с нулевым заполнением

Карта кодов операций процессоров C62x/C64x/C67x

Таблица 8 показывает описание команд, их синтаксис и значение. «Карта» кодов операций в функциональных модулях показана на рисунке 8.

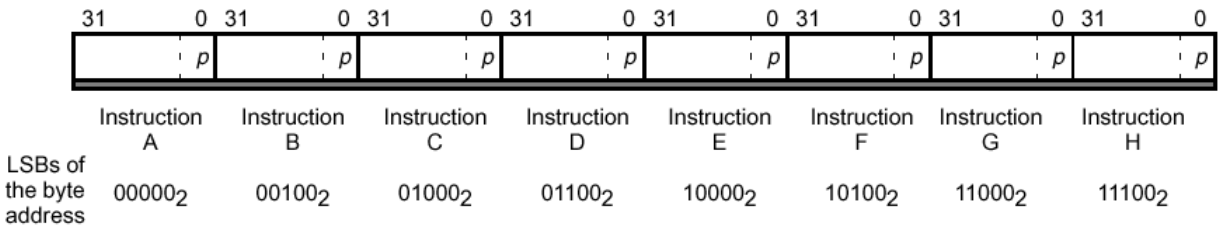
Таблица 8. Коды операций

Обозначение	Значение
baseR	Базовый адрес регистра
Creg	3-битное поле условного регистра
Cst	Константа
csta	Константа a
Cstb	Константа b
Dst	Нахождение
H	MVK или MVKH бит
ld/st	Загрузка/хранение операционного поля
Mode	Формат адреса
offsetR	Регистр смещения
op	Поле, содержащее код операции
p	Параллельное выполнение
r	LDDW бит
rsv	резерв
s	Выбор стороны A или B
src2	Источник 2
src1	Источник 1
ucstn	n-битное беззнаковое поле
y	Выбор .D1 или .D2
z	Тест сравнения с 0

Параллельные операции

Команды выбираются из памяти блоком из 8 команд (fetch packet). Основной формат такого блока показан на рисунке 9. Каждый блок состоит из 256 бит (восьми слов).

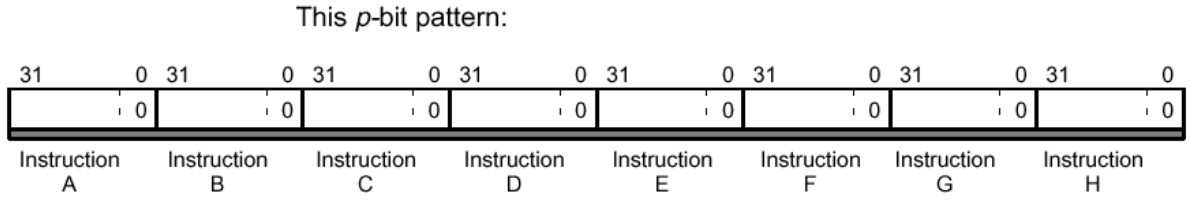
Рисунок 9. Основной формат блока команд



Выполнение каждой команды контролируется специальным битом в каждой команде, p-битом. P – бит устанавливается, когда команда выполняется параллельно с другой командой. P – биты читаются слева направо (от меньшего к большему адресу).

Когда p – бит команды i равен 1, то команда i+1 выполняется параллельно с командой i (в одном цикле). Если p – бит равен 0, то команда i+1 выполняется в следующем цикле после выполнения команды i.
На рисунках 10, 11 и 12 показаны различные варианты выполнения команд из блока команд.

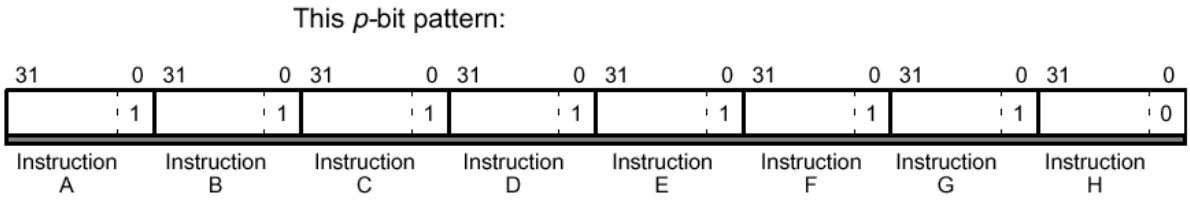
Рисунок 10. Полностью последовательное выполнение команд



Cycle/Execute Packet	Instructions
1	A
2	B
3	C
4	D
5	E
6	F
7	G
8	H

Здесь все команды выполняются последовательно, так как значение всех p – битов равны 0.

Рисунок 11. Полностью параллельное выполнение команд

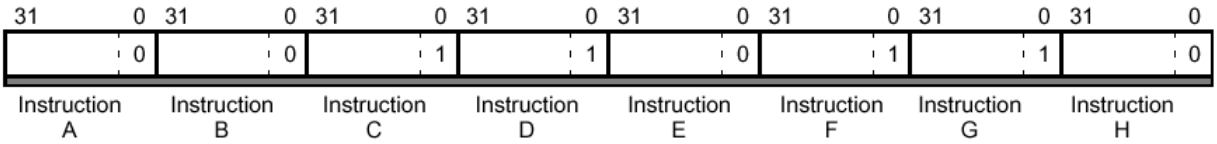


Cycle/Execute Packet	Instructions							
1	A	B	C	D	E	F	G	H

Здесь все команды выполняются параллельно (в течении одного цикла), так как значения всех p - битов равны 1.

Рисунок 12. Смешанное выполнение команд

This *p*-bit pattern:



results in this execution sequence:

Cycle/Execute Packet	Instructions		
1	A		
2	B		
3	C	D	E
4	F	G	H

Здесь команды выполняются как последовательно, так и параллельно.

Набор команд для проведения операций над числами с плавающей точкой

Процессор TMS 320C67x является цифровым сигнальным процессором для выполнения операций над числами как с фиксированной, так и с плавающей точкой. Поэтому у него, кроме описанных выше операций добавляются операции для работы над числами с плавающей точкой. У этого процессора также добавлены 32-битное integer умножение, загрузка двойных слов и операции над числами с плавающей точкой, включающие в себя сложение, вычитание и умножение.

К командам, описанным в таблице 7 здесь добавляются команды, описанные в таблице 9.

Таблица 9. Команды и их выполнение для процессора C67x

Обозначение	Значение
dp	Значение регистра числа представленного с двойной точностью (64 – битное представление)
dp(x)	Конвертация x к формату dp
int	32 – битное integer значение
int(x)	Конвертация x к формату int
rcp(x)	Обоюдное приближение x
sdint	Значащее 64 – битное integer значение (два регистра)
sp	Представление числа с одинарной точностью
sp(x)	Конвертация x к формату sp
sqrcp(x)	Квадратный корень из rcp(x)

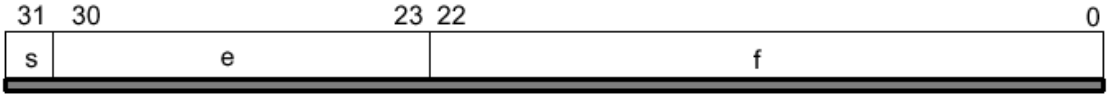
Числа с плавающей точкой могут быть представлены в двух форматах: с одинарной и двойной точностью (single-precision и double-precision).

Числа представленные с одинарной точностью являются 32-разрядными и хранятся в одном регистре. Числа представленные с двойной точностью являются 64-разрядными и хранятся в регистровых парах. Регистровые пары представляют собой соединенные четный и нечетный регистры одного регистрового файла. Последние значащие 32 разряда числа загружаются в четный регистр. При записи таких регистровых пар вначале указывается нечетный, а потом четный регистры. Например: A1:A0, B3:B2.

На следующих рисунках показаны поля регистров для различных типов записи чисел.

Для этих чисел возможны нормализованное и ненормализованное представления.

Рисунок 13. Представление чисел с одинарной точностью



Legend: s sign bit (0 positive, 1 negative)
e 8-bit exponent (0 < e < 255)
f 23-bit fraction
$0 < f < 1*2^{-1} + 1*2^{-2} + ... + 1*2^{-23}$ or
$0 < f < ((2^{23})-1)/(2^{23})$

Здесь для записи такого числа используется один регистр.
Следующие формулы показывают перевод s, e, f полей в формат числа с одинарной точностью.

$-1^s*2^{(e-127)}*1.f$ *при* $0<e<255$ - нормализованное представление
$-1^s*2^{-126}*0.f$ *при* $e=0, f\neq0$ - ненормализованное представление

Рисунок 14. Представление чисел с двойной точностью

Legend: s sign bit (0 positive, 1 negative)
e 11-bit exponent (0 < e < 2047)
f 52-bit fraction
$0 < f < 1*2^{-1} + 1*2^{-2} + ... + 1*2^{-52}$ or
$0 < f < ((2^{52})-1)/(2^{52})$

Здесь для записи числа используется регистровая пара, состоящая из четного(even) и нечетного (odd) регистров.
Следующие формулы показывают перевод s, e, f полей в формат числа с двойной точностью.

$-1^s*2^{(e-1023)}*1.f$ *при* $0<e<2047$ - нормализованное представление
$-1^s*2^{-1022}*0,f$ *при* $e=0, f\neq0$ - ненормализованное представление

Конвейер команд процессоров C62x/C64x

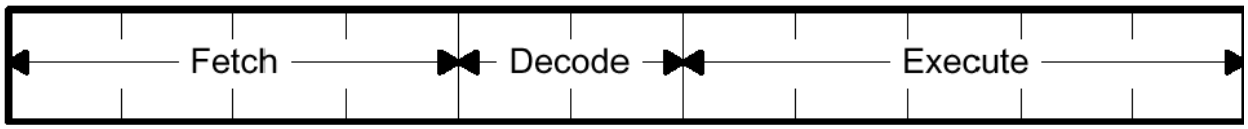
Конвейер команд организован для более простого выполнения команд процессором. Конвейер этих процессоров может реализовывать выполнение до восьми операций в течении одного цикла. Загрузка и сохранение адресов с помощью конвейера команд происходит быстрее и предотвращая возможные конфликты памяти.

Фазы работы конвейера команд представляются тремя ступенями:

- Выборка (Fetch)
- Декодирование (Decode)
- Выполнение (Execute)

Все команды процессоров C62x/C64x проходят через эти три ступени. Ступень выборки представляется четырьмя фазами, ступень декодирования содержит две фазы, а ступень выполнения команды содержит различное количество фаз, в зависимости от выполняемой команды. Ступени конвейера команд показаны на рисунке 15.

Рисунок 15. Ступени конвейера команд



Фазы процесса выборки:

- PG генерация адреса программы
- PS посылка адреса программы
- PW ожидание доступа
- PR передача выборки программы

Выборка состоит из восьми команд. На рисунке 16(a) показаны фазы выборки команд, очередь идет слева направо. Рисунок 16 (b) показывает функциональную диаграмму передачи команд через фазы выборки. Во время PG фазы адрес программы генерируется в центральном процессоре. Во время PS фазы адрес программы посылается в память. Во время PW фазы происходит чтение. И, наконец, во время PR фазы программа отсылается в центральный процессор. На рисунке 17 показана выборка команд, проходящая через фазы этой ступени.

Рисунок 16. Фазы выборки конвейера команд

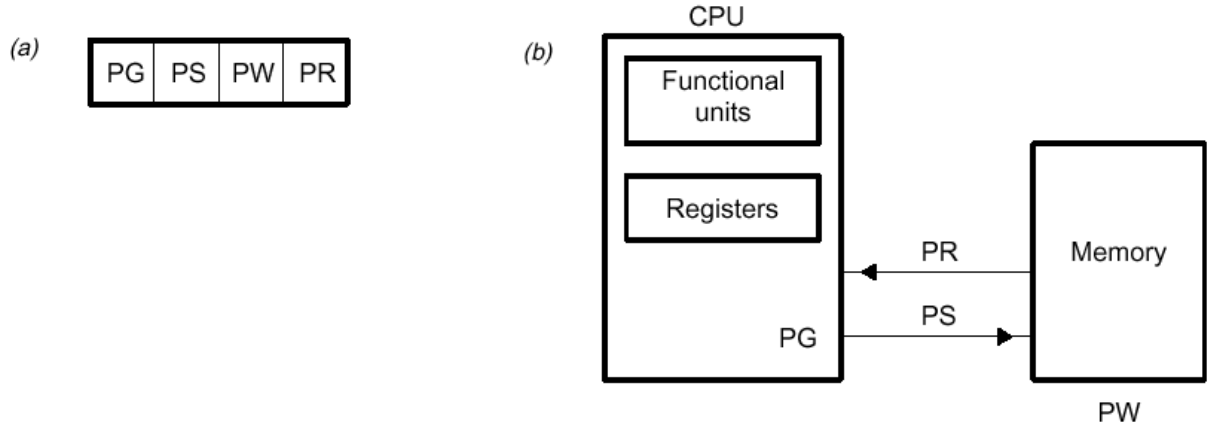
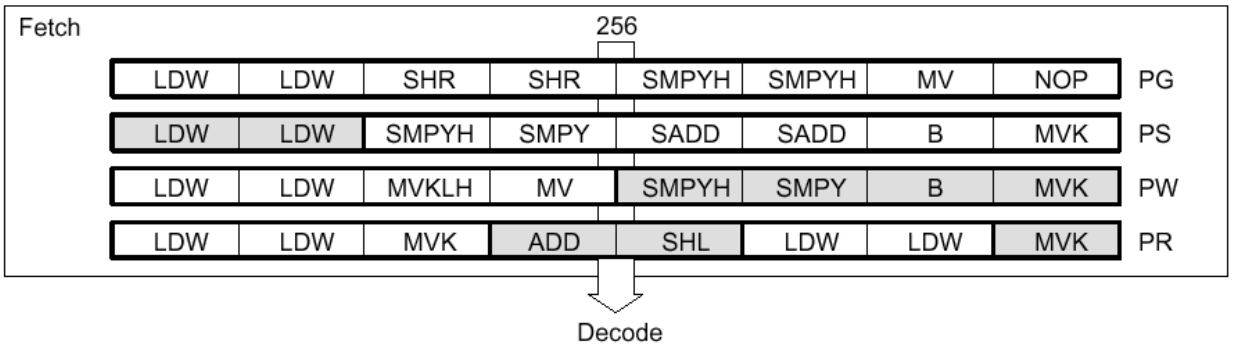


Рисунок 17. Прохождение выборки команд



Фазы процесса декодирования:

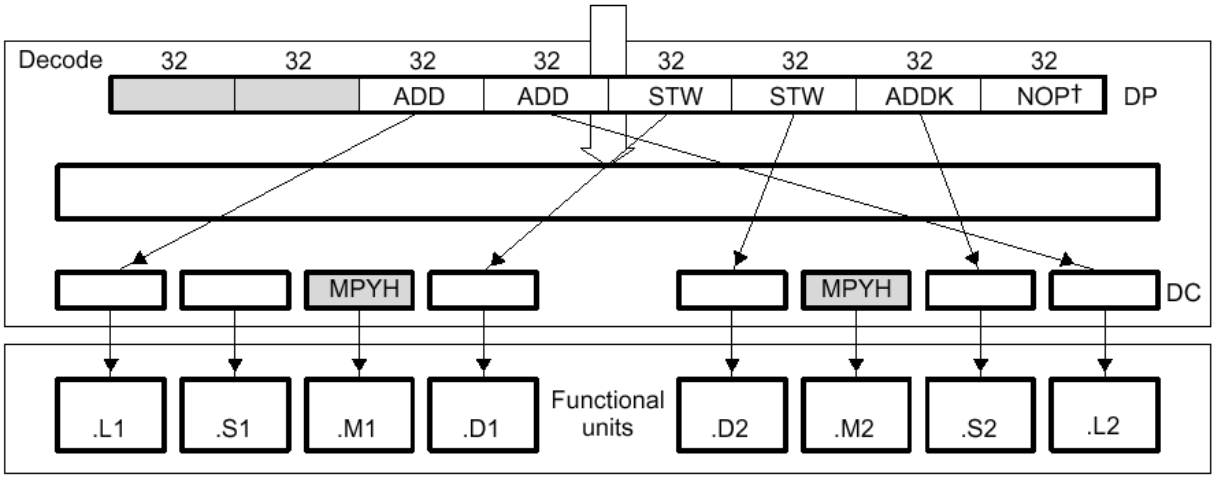
- DP организатор команд
- DC декодирование команды

В первой фазе (DP) выборка команд разделяется на выполняемые пакеты. Выполняемые пакеты состоят из одной команды или из двух – восьми команд, выполняющихся параллельно. Во время DC фазы регистры – источники и регистры назначения декодируются для выполнения операции в функциональном модуле. На рисунке 18 представлены фазы этой ступени (декодирования), очередь слева направо.

Рисунок 18. Фазы ступени декодирования

На рисунке 19 показана выборка команд, состоящая из двух выполняемых пакетов, которая проходит через фазы ступени декодирования.

Рисунок 19. Выборка команд, проходящая через ступень декодирования

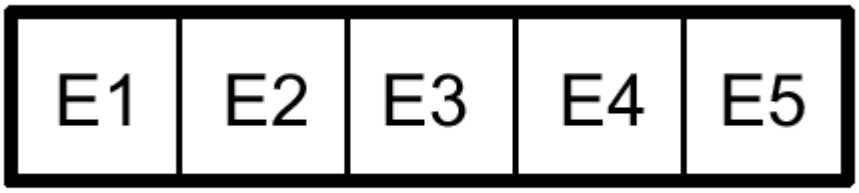


Ступень выполнения

Ступень выполнения состоит из пяти фаз (E1-E5). Различные типы команд требуют различное количество этих фаз для своего выполнения.

Последовательность расположения фаз показана на рисунке 20, очередь слева направо.

Рисунок 20. Фазы ступени выполнения



На рисунках 21 и 22 показаны функциональные диаграммы выполнения для процессоров C62x и C64x соответственно.

Рисунок 21. Ступень выполнения для процессора C62x

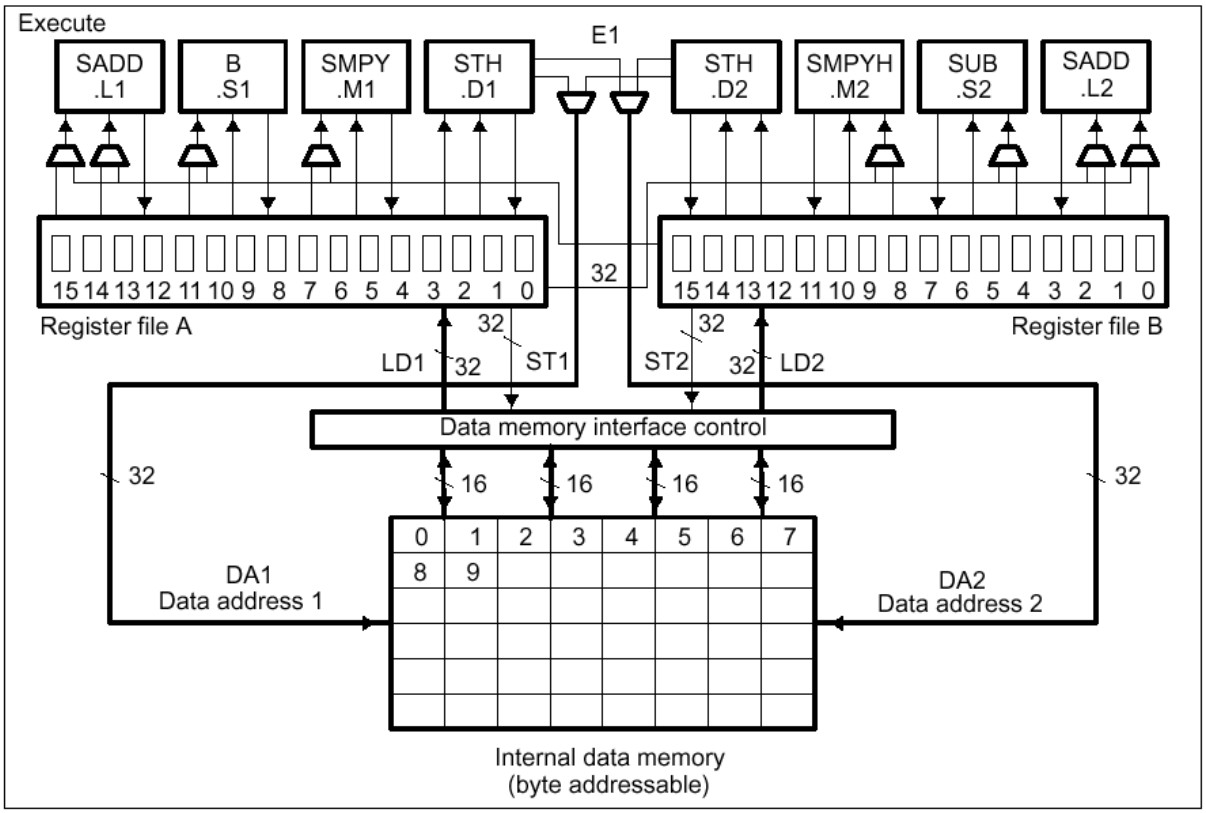
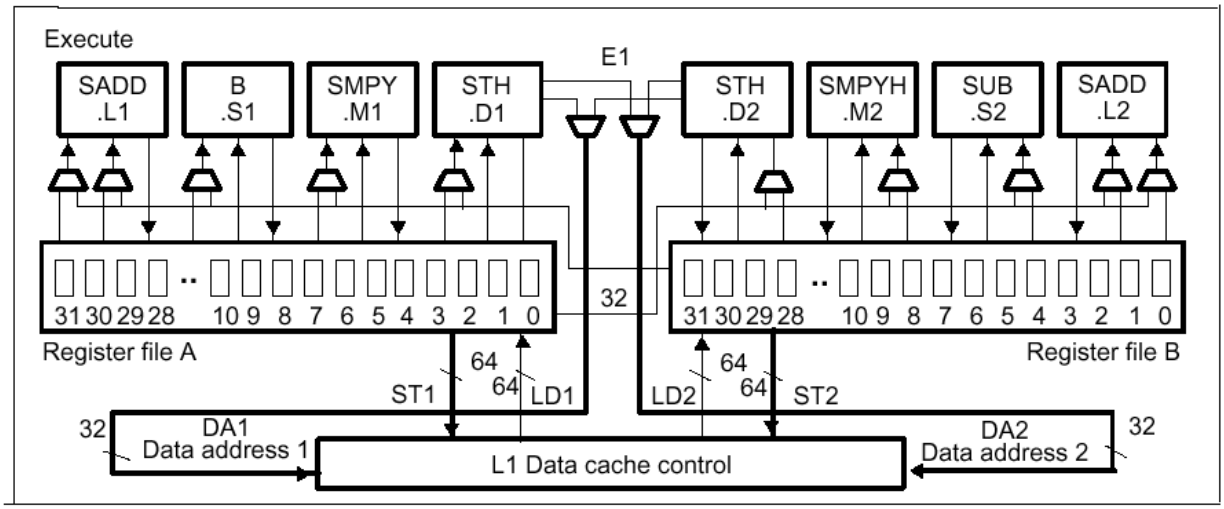


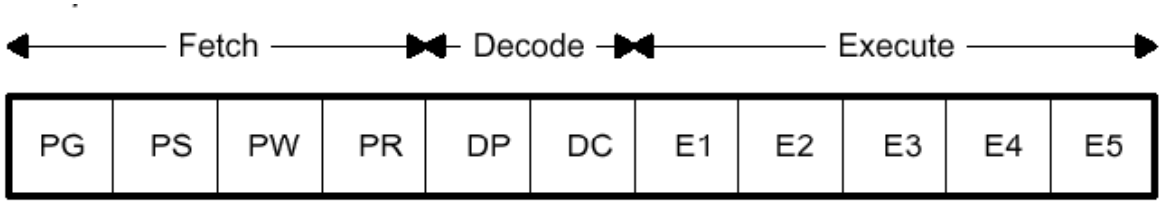
Рисунок 22. Ступень выполнения для процессора C64x



Обобщение операций конвейера команд

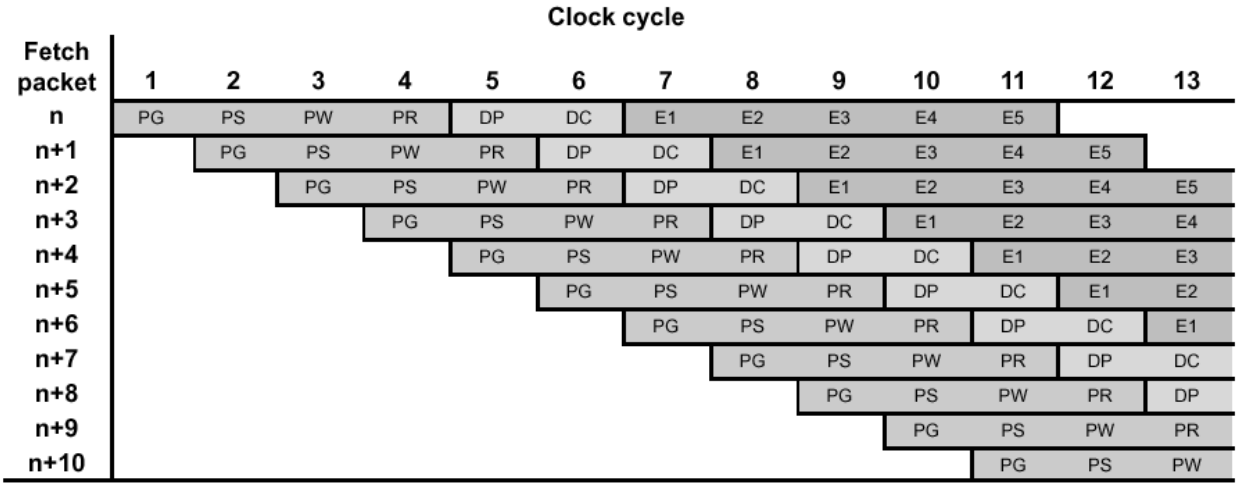
На рисунке 23 показаны все фазы каждой ступени конвейера команд в порядке очереди, слева направо.

Рисунок 23. Все фазы конвейера команд



На следующем рисунке показан пример прохождения выборки последовательных пакетов, которые содержат восемь параллельных операций. В случае, когда конвейер команд полон, все команды выполняются параллельно и переносятся в следующий пакет выполнения.

Рисунок 24. Пример прохождения выборки через конвейер команд



В следующей таблице показаны операции, выполняющиеся в конвейере команд процессоров C62x/C64x при выполнении операций над числами с фиксированной точкой.

Все операции конвейера команд основаны на цикле работы центрального процессора. Цикл работы CPU –это период времени, когда текущий выполняемый пакет находится в текущей фазе конвейера команд.

Таблица 10. Операции, выполняющиеся в конвейере команд

Ступень	Фаза	Символ	Выполнение фазы	Тип выполнения команды
Выборка	Генерация адреса программы	PG	Вычисление адреса пакета выборки	
	Посылка адреса программы	PS	Посылка адреса пакета выборки в память	
	Ожидание программы	PW	Предоставление доступа к программной памяти	
	Посылка данных программы	PR	Пакет выборки попадает в CPU	
Декодирование	Организация	DP	Следующий выполняемый пакет из пакета выборки вычислен и отправлен в соответствующий функциональный модуль для декодирования	
	Декодирование	DC	Команды декодируются в функциональном модуле	
Выполнение	Выполнение 1	E1	Для загрузки и сохранения команд генерируется адрес и проводится его модификация для записи в регистровый файл. Для одноцикловых команд результат записывается в регистровый файл.	Один цикл
	Выполнение 2	E2	Для загрузки команды адрес посылается в память. Для сохранения команды в память посылаются данные и адрес. Для единичных 16*16 - умножение команд результат записывается в регистровый файл.	Умножение
	Выполнение 3	E3	Предоставляется доступ к памяти.	Сохранение
	Выполнение 4 Выполнение	E4 E5	Результат записывается в регистровый файл. Данные для загрузки команды записываются в регистр.	Умножение Расширение Загрузка

	5			

Операции конвейера команд могут быть разделены на семь основных типов.
Шесть из них показаны в таблице 11.

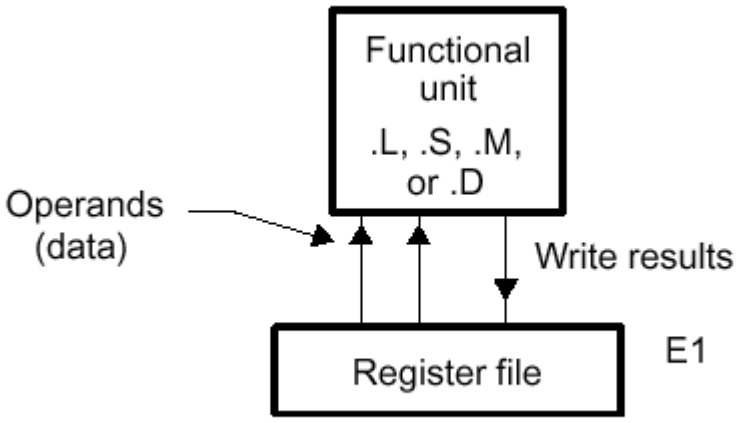
Таблица 11. Описание типов операций конвейера команд

	Тип операции					
Фазы выполнения		Single Cycle	16*16 Single Multiply	Store	C64x Multiply Extension	Load
	E1	Вычисление результата и запись в регистр	Чтение операндов и начало вычислений	Вычисление адреса	Чтение операндов и начало вычислений	Вычисление адреса
	E2		Вычисление результата и запись вы регистр	Посылка адреса и данных в память		Посылка адреса в память
	E3			Доступ к памяти		Доступ к памяти
	E4				Запись результата в регистр	Посылка данных обратно в ЦП
	E5					Запись данных в регистр
Слоты задержки		0	1	0	3	4

Операции одного цикла (Single-Cycle Instructions)

Эти операции выполняются в течении одной фазы E1 конвейера команд. Такое выполнение показано на рисунке 25. Здесь читаются операнды, операция представлена, результат записывается в регистр, всё в течении фазы E1. Эти операции не имеют слота задержки.

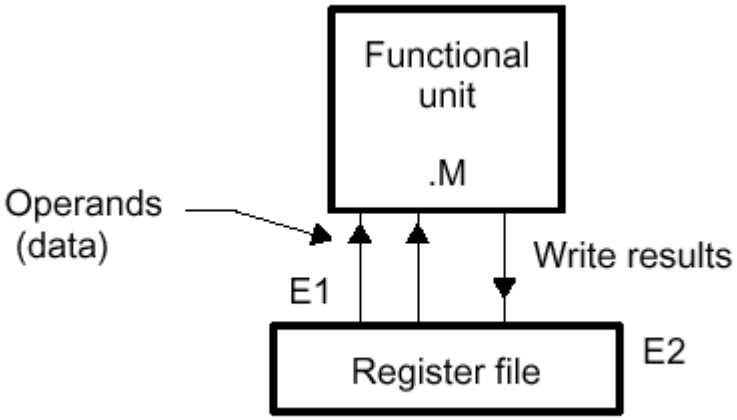
Рисунок 25. Single-Cycle Выполнение



Операции двух циклов (Two-Cycle Instructions)

Эти операции умножения для выполнения используют две фазы – E1 и E2. В фазе E1 читаются операнды и умножение начинается. В фазе E2 умножение заканчивается и результат записывается в регистр назначения. Операции умножения имеют один слот задержки. Выполнение такой операции умножения показано на рисунке 26. Схема подходит и для других операций, выполняемых процессором C64x.

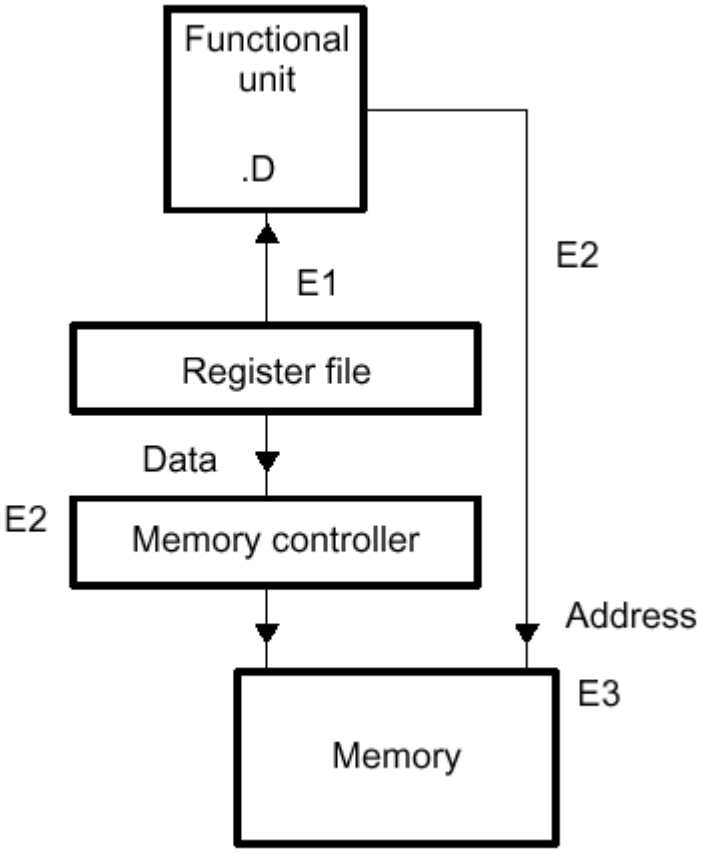
Рисунок 26. 16*16 умножение



Операции хранения (Store Instructions)

Эти операции требуют для своего выполнения фазы от E1 до E3. Рисунок 27 иллюстрирует выполнение операции хранения. В фазе E1 вычисляется адрес хранения данных. В фазе E2 адрес хранения данных и сами данные посылаются в память. В фазе E3 реализуется запись данных в память. Модификация адреса происходит в течении фазы E1. Эти операции не имеют слота задержки.

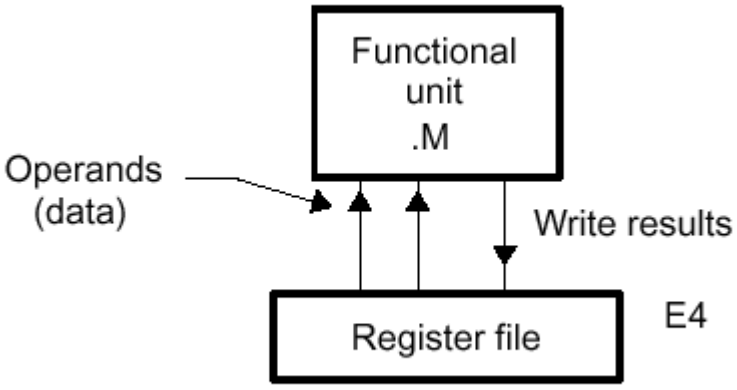
Рисунок 27. Выполнение операции хранения



Расширенные операции умножения (Extended Multiply Instructions)

Эти операции для своего выполнения требуют использования фаз от E1 до E4. В фазе E1 читаются операнды и умножение начинается. В фазе E4 умножение заканчивается и результат записывается в регистр назначения. Такие операции имеют три слота задержки.

Рисунок 28. Выполнение расширенной операции умножения

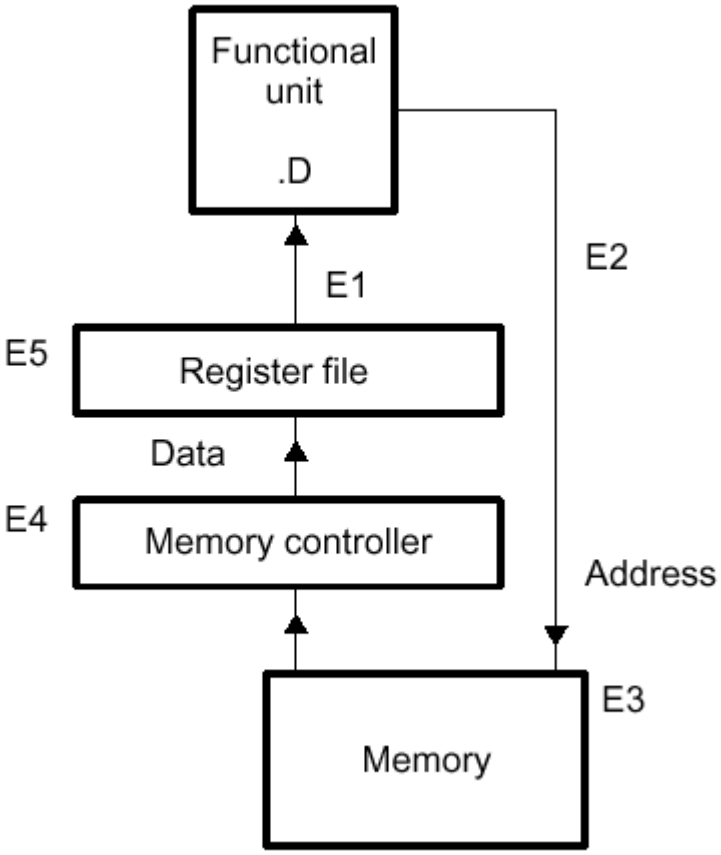


Операции загрузки (Load Instructions)

Такие операции требуют для своего выполнения все пять фаз ступени выполнения конвейера команд. На рисунке 29 показаны операции, производимые конвейером команд для загрузки. В фазе E1 указатель адреса данных модифицируется в своем

регистре. В фазе E2 адрес данных передаётся в память данных. В фазе E3 читается содержимое этого адреса. В фазе E4 данные посылаются центральному процессору. И, наконец, в фазе E5 данные загружаются в регистр. Так как данные не записываются в регистр до E5, эти операции имеют 4 слота задержки.

Рисунок 29. Выполнение операции загрузки



Прерывания (Interrupts)

Далее описываются прерывания процессора, включающие сброс и немаскированные (NMI) прерывания. У процессоров TMS 320C6000 существует три различных типа прерываний. Прерывание сброса имеет самый высокий приоритет. Второе по приоритетам прерывание – это немаскированное прерывание. Самый низкий приоритет имеют прерывания INT4-INT15. Reset, NMI и некоторые из INT4-INT15 устанавливаются контактами процессора. Некоторые из прерываний INT4-INT15 могут быть использованы внешними периферийными устройствами или с помощью программного обеспечения.

Приоритеты прерываний показаны в таблице 12.

Таблица 12. Приоритеты прерываний

Priority	Interrupt Name
Highest	Reset
	NMI
	INT4
	INT5
	INT6
	INT7
	INT8
	INT9
	INT10
	INT11
	INT12
	INT13
	INT14
Lowest	INT15

Reset Имеет самый высокий уровень приоритета. Предназначен для остановки работы процессора и возвращения его в начальный режим.

NMI Имеет второй по уровню приоритет и служит в основном для предупреждения процессора о серьёзных проблемах работы, связанных, например, с неминуемым отключением питания.

INT4-INT15 Являются маскированными прерываниями и имеют самый низкий приоритет. Эти прерывания могут быть связаны с внешними устройствами, on-chip периферией, контролироваться с помощью программного обеспечения или могут быть не задействованы.