

Тема 8

Конвейерная организация и принципы конвейерной обработки

Основные вопросы:

- 8.1. Простейший конвейер команд и оценки его эффективности
- 8.2. Уровни конвейеризации
- 8.3. Понятие конфликтов в конвейере и пути их устранения: структурные конфликты, конфликты по данным и по управлению
- 8.4. Реализация точного прерывания в конвейере
- 8.5. Длинные конвейеры

Разработчики архитектуры компьютеров издавна прибегали к методам проектирования, известным под общим названием "*совмещение операций*", при котором аппаратура компьютера в любой момент времени выполняет одновременно более одной базовой операции.

Этот общий метод включает два понятия: *параллелизм и конвейеризацию*. Эти термины отражают два совершенно различных подхода. При *параллелизме* совмещение операций достигается путем воспроизведения в нескольких копиях аппаратной структуры. Высокая производительность достигается за счет одновременной работы всех элементов структур, осуществляющих решение различных частей задачи.

Конвейеризация (или конвейерная обработка) в общем случае основана на разделении подлежащей исполнению функции на более мелкие части, называемые **ступенями**, и выделении для каждой из них отдельного блока аппаратуры. Так обработку любой машинной команды можно разделить на несколько этапов (несколько ступеней), организовав передачу данных от одного этапа к следующему. При этом конвейерную обработку можно использовать для совмещения этапов выполнения разных команд. Производительность при этом возрастает благодаря тому, что одновременно на различных ступенях конвейера выполняются несколько команд. Конвейерная обработка такого рода широко применяется во всех современных быстродействующих процессорах.

8.1. Простейший конвейер команд и оценки его эффективности

Для иллюстрации основных принципов построения процессоров будем использовать простейшую архитектуру, содержащую 32 целочисленных регистра общего назначения (R_0, \dots, R_{31}), 32 регистра плавающей точки (F_0, \dots, F_{31}) и счетчик команд РС. Будем считать, что набор команд нашего процессора включает типичные арифметические и логические операции, операции с плавающей точкой, операции пересылки данных, операции управления потоком команд и системные операции. В арифметических командах используется трехадресный формат, типичный для **RISC-процессоров**, а для обращения к памяти используются операции загрузки и записи содержимого регистров в память.

Выполнение типичной команды можно разделить на следующие этапы:

- выборка команды – **IF** (по адресу, заданному счетчиком команд, из памяти извлекается команда);
- декодирование команды / выборка операндов из регистров – **ID**;

- выполнение операции / вычисление эффективного адреса памяти – **EX**;
- обращение к памяти – **MEM**;
- запоминание результата – **WB**.

Чтобы конвейеризовать выполнение команд можно просто разбить выполнение каждой команды на указанные выше этапы, отведя для выполнения каждого этапа один такт синхронизации, и начинать в каждом такте выполнение новой команды. Для хранения промежуточных результатов каждого этапа необходимо использовать *регистровую станцию* (память). Промежуточные регистровые станции обеспечивают передачу данных и управляющих сигналов с одной ступени конвейера на следующую. Хотя общее время выполнения одной команды в таком конвейере будет составлять пять тактов, в каждом такте аппаратура будет выполнять в совмещенном режиме пять различных команд.

Работу конвейера можно условно представить в виде сдвинутых во времени схем процессора (рис. 8.1). Этот рисунок хорошо отражает совмещение во времени выполнения различных этапов команд. Однако чаще для представления работы конвейера используются временные диаграммы (рис. 8.2), на которых обычно изображаются выполняемые команды, номера тактов и этапы выполнения команд.

Конвейеризация увеличивает пропускную способность процессора (количество команд, завершающихся в единицу времени), но она ***не сокращает время выполнения отдельной команды***. В действительности, даже несколько увеличивается время выполнения каждой команды из-за накладных расходов, связанных с управлением регистровыми станциями. Однако увеличение пропускной способности означает, что программа будет выполняться быстрее по сравнению с простой не конвейерной схемой.

Тот факт, что время выполнения каждой команды в конвейере не уменьшается, накладывает некоторые ограничения на практическую длину конвейера. Кроме ограничений, связанных с задержкой конвейера, имеются также ограничения, возникающие в результате несбалансированности задержки на каждой его ступени и из-за накладных расходов на конвейеризацию. Частота синхронизации не может быть выше, а, следовательно, такт синхронизации не может быть меньше, чем время, необходимое для работы наиболее медленной ступени конвейера.

Накладные расходы на организацию конвейера возникают из-за:

- задержки сигналов в конвейерных регистрах (защелках) и
- из-за перекосов сигналов синхронизации.

Конвейерные регистры к длительности такта добавляют время установки и задержку распространения сигналов. В предельном случае длительность такта можно уменьшить до суммы накладных расходов и перекоса сигналов синхронизации, однако при этом в такте не останется времени для выполнения полезной работы по преобразованию информации.

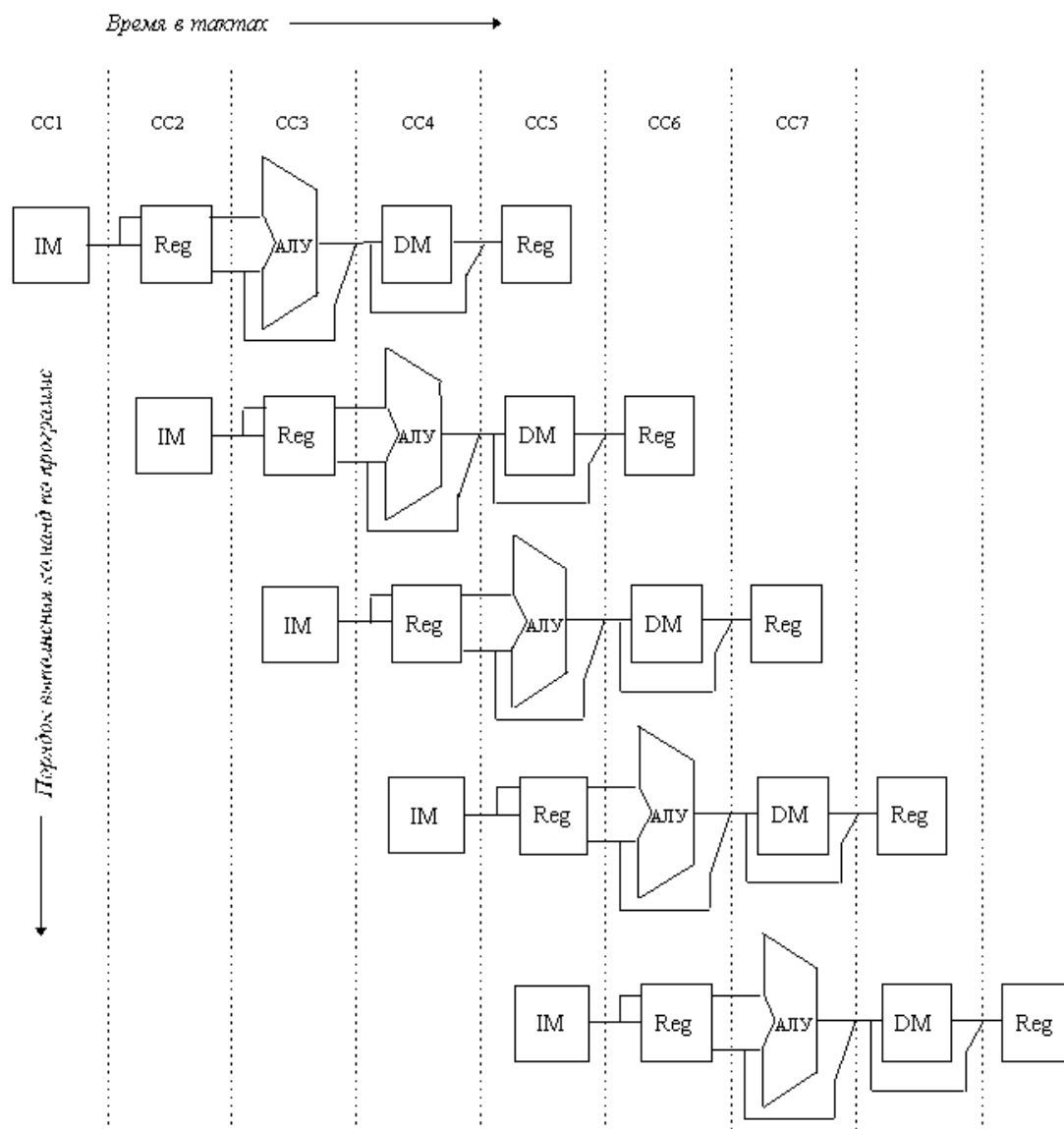


Рис. 8.1. Схемы процессора, реализующие конвейер команд

Номер команды	Номер такта								
	1	2	3	4	5	6	7	8	9
Команда i	IF	ID	EX	MEM	WB				
Команда i+1		IF	ID	EX	MEM	WB			
Команда i+2			IF	ID	EX	MEM	WB		
Команда i+3				IF	ID	EX	MEM	WB	
Команда i+4					IF	ID	EX	MEM	WB

Рис. 8.2. Представление о работе конвейера в форме временной диаграммы

В качестве примера рассмотрим не конвейерную машину с пятью этапами выполнения операций, которые, например, имеют длительность 50, 50, 60, 50 и 50 нс соответственно (рис. 8.3). Пусть накладные расходы на организацию конвейерной обработки составляют 5 нс. Тогда среднее время выполнения команды в не конвейерной машине будет равно 260 нс. Если же используется конвейерная организация, длительность такта будет равна длительности самого медленного этапа обработки плюс накладные расходы, т.е. 65 нс. Это время соответствует среднему

времени выполнения команды в конвейере. Таким образом, ускорение, полученное в результате конвейеризации, будет равно отношению $260/65=4$:

Среднее время выполнения команды в не конвейерном режиме = 260

Среднее время выполнения команды в конвейерном режиме = 6

Ускорение от конвейеризации команды =
$$\frac{\text{Среднее время вып. команды в неконвейерном режиме}}{\text{Среднее время вып. команды в конвейерном режиме}}$$

Ускорение от конвейеризации = $260/65=4$.

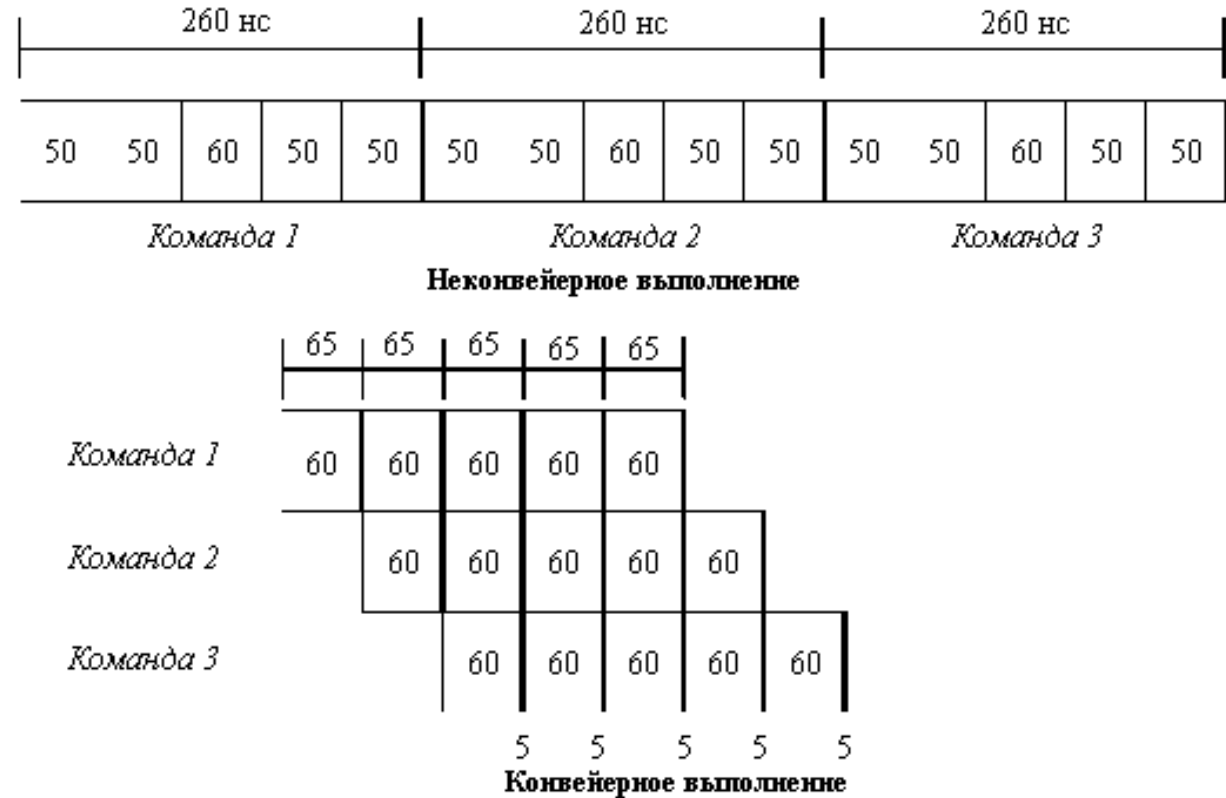


Рис. 8.3. Эффект от конвейеризации при выполнении 3-х команд – четырехкратное ускорение

Общая формула ускорения для арифметического конвейера может быть получена на примере сложения двух векторов длины n с плавающей запятой в конвейерном режиме и имеет вид:

$$\xi = \frac{n * k}{(n + k)}$$

где k – число ступеней конвейера, а n – длина одной последовательности исходных данных.

Важной характеристикой конвейерной обработки является среднее количество тактов (CPI) для выполнения команды в конвейере:

CPI конвейера = CPI идеального конвейера +

Приостановки из-за структурных конфликтов +
Приостановки из-за конфликтов типа RAW +

Приостановки из-за конфликтов типа WAR +
Приостановки из-за конфликтов типа WAW +
Приостановки из-за конфликтов по управлению

CPI идеального конвейера есть не что иное, как максимальная пропускная способность, достижимая при реализации.

Уменьшая каждое из слагаемых в правой части выражения, минимизируем общий CPI конвейера и таким образом увеличиваем пропускную способность команд. Это выражение позволяет также охарактеризовать различные методы сокращения CPI, по тому компоненту общего CPI, который соответствующий метод уменьшает (см. ниже).

8.2. Уровни конвейеризации

- Макроконвейер – конвейеризация на уровне процессоров
- Конвейер команд – конвейеризация команд процессора
- Конвейер арифметический - конвейеризация на уровне выполнения команд процессора

8.3. Понятие конфликтов в конвейере и пути их устранения: структурные конфликты, конфликты по данным и по управлению

Конвейеризация эффективна только тогда, когда загрузка конвейера близка к полной, а скорость подачи новых команд и операндов соответствует максимальной производительности конвейера. Если произойдет задержка, то параллельно будет выполняться меньше операций, и суммарная производительность снизится. Такие задержки могут возникать в результате возникновения конфликтных ситуаций.

При реализации конвейерной обработки возникают ситуации, которые препятствуют выполнению очередной команды из потока команд в предназначенном для нее такте. Такие ситуации называются **конфликтами**. Конфликты снижают реальную производительность конвейера, которая могла бы быть достигнута в идеальном случае.

Рассмотрим различные типы конфликтов, возникающие при выполнении команд в конвейере, и способы их разрешения.

Существуют три класса конфликтов:

1. **Структурные конфликты**, которые возникают из-за конфликтов по ресурсам, когда аппаратные средства не могут поддерживать все возможные комбинации команд в режиме одновременного выполнения с совмещением.
2. **Конфликты по данным**, возникающие в случае, когда выполнение одной команды зависит от результата выполнения предыдущей команды.
3. **Конфликты по управлению**, которые возникают при конвейеризации команд переходов и других команд, которые изменяют значение счетчика команд.

Конфликты в конвейере приводят к необходимости приостановки выполнения команд (pipeline stall). Обычно в простейших конвейерах, если приостанавливается какая-либо команда, то все следующие за ней команды также приостанавливаются. Команды, *предшествующие* приостановленной, могут *продолжать выполняться*, но во время приостановки *не выбирается ни одна новая команда*.

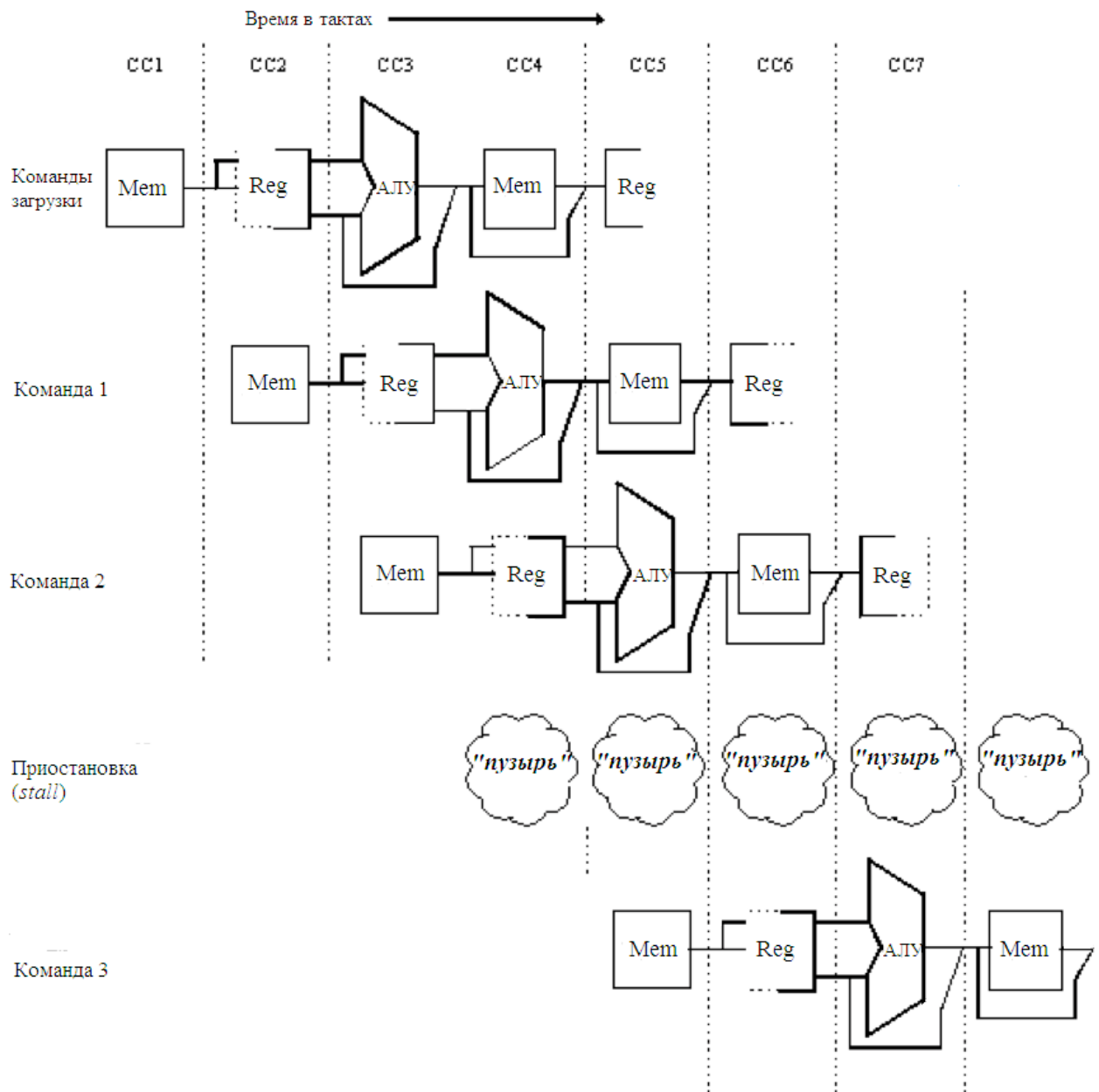


Рис. 8.4. Пример структурного конфликта при реализации памяти с одним портом

Совмещенный режим выполнения команд в общем случае требует конвейеризации функциональных устройств и дублирования ресурсов для разрешения всех возможных комбинаций команд в конвейере. Если какая-нибудь комбинация команд не может быть принята из-за конфликта по ресурсам, то говорят, что в машине имеется структурный конфликт.

Возникает вопрос: почему разработчики допускают **наличие структурных конфликтов**? Для этого имеются две причины: снижение стоимости и уменьшение задержки устройства. Конвейеризация всех функциональных устройств может оказаться слишком дорогой. Машины, допускающие два обращения к памяти в одном такте, должны иметь удвоенную пропускную способность памяти, например, путем организации **раздельных кэшей для команд и данных**. Аналогично, полностью конвейерное устройство деления с плавающей точкой требует огромного количества вентилях. Если структурные конфликты не будут возникать слишком часто, то может быть и не стоит платить за то, чтобы их обойти.

Как правило, можно разработать не конвейерное, или не полностью конвейерное устройство, имеющее меньшую общую задержку, чем полностью конвейерное.

Команда	Номер такта									
	1	2	3	4	5	6	7	8	9	10
Команда загрузки	IF	ID	EX	MEM	WB					
Команда 1		IF	ID	EX	MEM	WB				
Команда 2			IF	ID	EX	MEM	WB			
Команда 3				stall	IF	ID	EX	MEM	WB	
Команда 4						IF	ID	EX	WB	
Команда 5							IF	ID	EX	WB
Команда 6								IF	ID	EX

Рис. 8.5. Диаграмма работы конвейера при структурном конфликте

Одним из факторов, который оказывает существенное влияние на производительность конвейерных систем, являются *межкомандные логические зависимости* - **конфликты по данным**. Такие зависимости в большой степени ограничивают потенциальный параллелизм смежных операций, обеспечиваемый соответствующими аппаратными средствами обработки. Степень влияния этих зависимостей определяется как архитектурой процессора (в основном, структурой управления конвейером команд и параметрами функциональных устройств), так и характеристиками программ.

Конфликты по данным возникают в том случае, когда применение конвейерной обработки может изменить порядок обращений за операндами так, что этот порядок будет отличаться от порядка, который наблюдается при последовательном выполнении команд на не конвейерной машине. Рассмотрим конвейерное выполнение последовательности команд на рис. 8.6 и рис. 8.8.

ADD	R1, R2, R3	IF	ID	EX	MEM	WB				
SUB	R4, R4, R5		IF	ID	EX	MEM	WB			
AND	R6, R1, R7			IF	ID	EX	MEM	WB		
OR	R8, R1, R9				IF	ID	EX	MEM	WB	
OR	R10, R1, R11					IF	ID	EX	MEM	WB

Рис. 8.6. Последовательность команд в конвейере и ускоренная пересылка данных

В этом примере все команды, следующие за командой ADD, используют результат ее выполнения. Команда ADD записывает результат в регистр R1, а команда SUB читает это значение. Если не предпринять мер для предотвращения этого конфликта, то команда SUB прочтает неправильное значение и попытается его использовать.

ADD	R1, R2, R3	IF	ID	EX	MEM	WB				
			R			W				
SUB	R4, R4, R5		IF	ID	EX	MEM	WB			
				R			W			
AND	R6, R1, R7			IF	ID	EX	MEM	WB		
					R			W		
OR	R8, R1, R9				IF	ID	EX	MEM	WB	
						R			W	
OR	R10, R1, R11					IF	ID	EX	MEM	WB
							R			W

Рис. 8.8. Совмещение чтения и записи регистров в одном такте

Проблема, поставленная в этом примере, может быть разрешена с помощью достаточно простой аппаратной техники, которая называется *пересылкой* или *продвижением данных* (data forwarding), обходом (data bypassing), иногда закороткой (short-circuiting).

Эта аппаратура работает следующим образом:

- результат операции АЛУ с его выходного регистра всегда снова подается назад на входы АЛУ.
- если аппаратура обнаруживает, что предыдущая операция АЛУ записывает результат в регистр, соответствующий источнику операнда для следующей операции АЛУ, то логические схемы управления выбирают в качестве входа для АЛУ результат, поступающий по цепи "обхода", а не значение, прочитанное из регистрового файла (см. рис. 8.8).

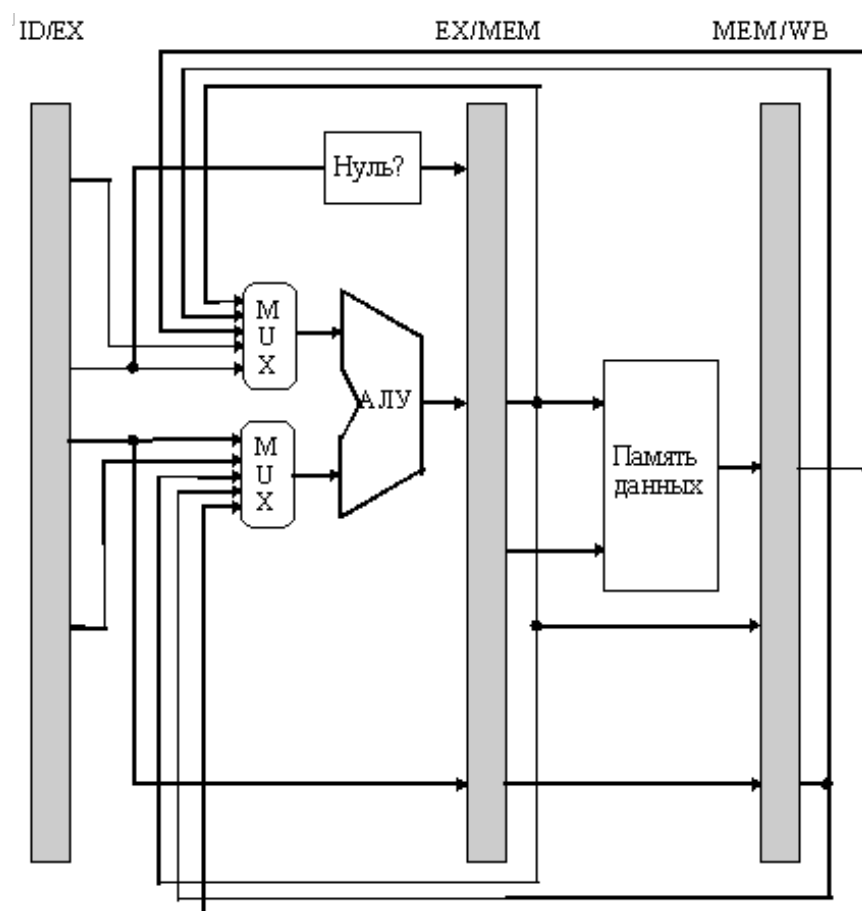


Рис. 8.8. АЛУ с цепями обхода и ускоренной пересылки

Эта техника "обходов" может быть обобщена для того, чтобы включить передачу результата прямо в то функциональное устройство (ФУ), которое в нем нуждается: результат с выхода одного устройства "пересылается" на вход другого, а не с выхода некоторого устройства только на его вход.

Конфликты по данным принято классифицировать на следующие три группы в зависимости от порядка операций чтения и записи:

- **RAW** (read after write) – чтение после записи;
- **WAR** (write after read) – запись после чтения;
- **WAW** (write after write) – запись после записи.

Конфликты по управлению могут вызывать даже большие потери производительности конвейера, чем конфликты по данным. Когда выполняется команда условного перехода, она может либо изменить, либо не изменить значение счетчика команд.

Если команда условного перехода заменяет счетчик команд значением адреса, вычисленного в команде, то **переход** называется **выполняемым**; в противном случае, он называется **невыполняемым**.

Простейший метод работы с условными переходами заключается в приостановке конвейера, как только обнаружена команда условного перехода до тех пор, пока она не достигнет ступени конвейера, которая вычисляет новое значение счетчика команд (см. рис. 8.9).

Команды перехода	IF	ID	EX	ME M	WB					
Следующая команда		IF	stall	stall	IF	ID	EX	ME M	WB	
Следующая команда +1			stall	stall	stall	IF	ID	EX	ME M	WB
Следующая команда +2				stall	stall	stall	IF	ID	EX	ME M
Следующая команда +3					stall	stall	stall	IF	ID	EX
Следующая команда +4						stall	stall	stall	IF	ID
Следующая команда +5							stall	stall	stall	IF

Рис. 8.9. Приостановка конвейера при выполнении команды условного перехода

Если конвейер будет приостановлен на три такта на каждой команде условного перехода, то это может существенно отразиться на производительности машины. Например, при частоте команд условного перехода в программах, равной 30% и идеальном CPI (**среднее число тактов на выдачу команды**), равным 1, машина с приостановками условных переходов достигает примерно только половины ускорения, получаемого за счет конвейерной организации. Таким образом, снижение потерь от условных переходов становится критическим вопросом.

Имеется несколько методов сокращения приостановок конвейера, возникающих из-за задержек выполнения условных переходов:

- метод выжидания;
- метод возврата;
- прогнозирование перехода как выполняемого;
- задержанные переходы.

8.4. Реализация точного прерывания в конвейере

Обработка прерываний в конвейерной машине достаточно сложна из-за того, что совмещенное выполнение команд затрудняет определение возможности безопасного изменения состояния машины произвольной командой. В конвейерной машине команда выполняется по этапам, и ее завершение осуществляется через несколько тактов после выдачи для выполнения. Кроме того, в процессе выполнения отдельных ступеней команда может изменить состояние машины. Тем временем возникшее прерывание может вынудить машину прервать выполнение еще не завершенных команд.

Когда происходит прерывание, для корректного сохранения состояния машины необходимо выполнить следующие шаги:

1. В последовательность команд, поступающих на обработку в конвейер, принудительно вставить команду перехода на прерывание.
2. Пока выполняется команда перехода на прерывание, погасить все требования записи, выставленные командой, вызвавшей прерывание, а также всеми следующими за ней в конвейере командами. Эти действия позволяют предотвратить все изменения состояния машины командами, которые не завершились к моменту начала обработки прерывания.
3. После передачи управления подпрограмме обработки прерываний операционной системы, она немедленно должна сохранить значение адреса команды (PC), вызвавшей прерывание. Это значение будет использоваться позже для организации возврата из прерывания.

Если конвейер может быть остановлен так, что команды, непосредственно предшествовавшие вызвавшей прерывание команде, завершаются, а следовавшие за ней могут быть заново запущены для выполнения, то говорят, что **конвейер обеспечивает точное прерывание**.

Поддержка точных прерываний во многих системах является обязательным требованием, а в некоторых системах была бы весьма желательной, поскольку она упрощает интерфейс операционной системы. Как минимум в машинах со страничной организацией памяти средства обработки прерываний должны обеспечивать точное прерывание **либо целиком с помощью аппаратуры**, либо с **некоторой поддержкой со стороны программных средств**.

Необходимость реализации в машине точных прерываний иногда оспаривается из-за некоторых проблем, которые осложняют повторный запуск команд. Повторный запуск сложен из-за того, что команды могут изменить состояние машины еще до того, как они гарантировано завершают свое выполнение (иногда гарантированное завершение команды называется фиксацией команды или фиксацией результатов выполнения команды). Поскольку команды в конвейере могут быть взаимозависимыми, блокировка изменения состояния машины может оказаться непрактичной, если конвейер продолжает работать. Таким образом, по мере увеличения степени конвейеризации машины возникает необходимость отката любого изменения состояния, выполненного до фиксации команды. К счастью, в простых конвейерах, подобных рассмотренному, эти проблемы не возникают. На рис. 8.10 показаны ступени рассмотренного конвейера и причины прерываний, которые могут возникнуть на соответствующих ступенях при выполнении команд.

Ступень конвейера	Причина прерывания
IF	Ошибка при обращении к странице памяти при выборке команды; не выровненное обращение к памяти; нарушение защиты памяти
ID	Неопределенный или запрещенный код операции
EX	Арифметическое прерывание
MEM	Ошибка при обращении к странице памяти при выборке данных; не выровненное обращение к памяти; нарушение защиты памяти
WB	Отсутствует

Рис. 8.10. Причины прерываний в простейшем конвейере

8.5. Длинные конвейеры

В рассмотренном нами конвейере стадия выполнения команды (EX) составляла всего один такт, что вполне приемлемо для целочисленных операций. Однако для большинства операций плавающей точки требуется большее число тактов. Это привело бы к существенному увеличению такта синхронизации конвейера, либо к сверхмерному увеличению количества оборудования (объема логических схем) для реализации устройств плавающей точки. Проще всего представить, что команды плавающей точки используют тот же самый конвейер, что и целочисленные команды, но с двумя важными изменениями:

- во-первых, такт EX может повторяться многократно столько раз, сколько необходимо для выполнения операции;
- во-вторых, в процессоре может быть несколько функциональных устройств, реализующих операции плавающей точки. При этом могут возникать приостановки конвейера, если выданная для выполнения команда, либо вызывает структурный конфликт по ФУ, которое она использует, либо существует конфликт по данным.

Конвейер с такими дополнениями называется *длинным конвейером*.

Рассмотрим понятие длинного конвейера на примере.

Допустим, что в нашей реализации процессора имеются четыре отдельных функциональных устройства (см. рис. 8.11):

1. Основное целочисленное устройство.
2. Устройство умножения целочисленных операндов и операндов с плавающей точкой.
3. Устройство сложения с плавающей точкой.
4. Устройство деления целочисленных операндов и операндов с плавающей точкой.

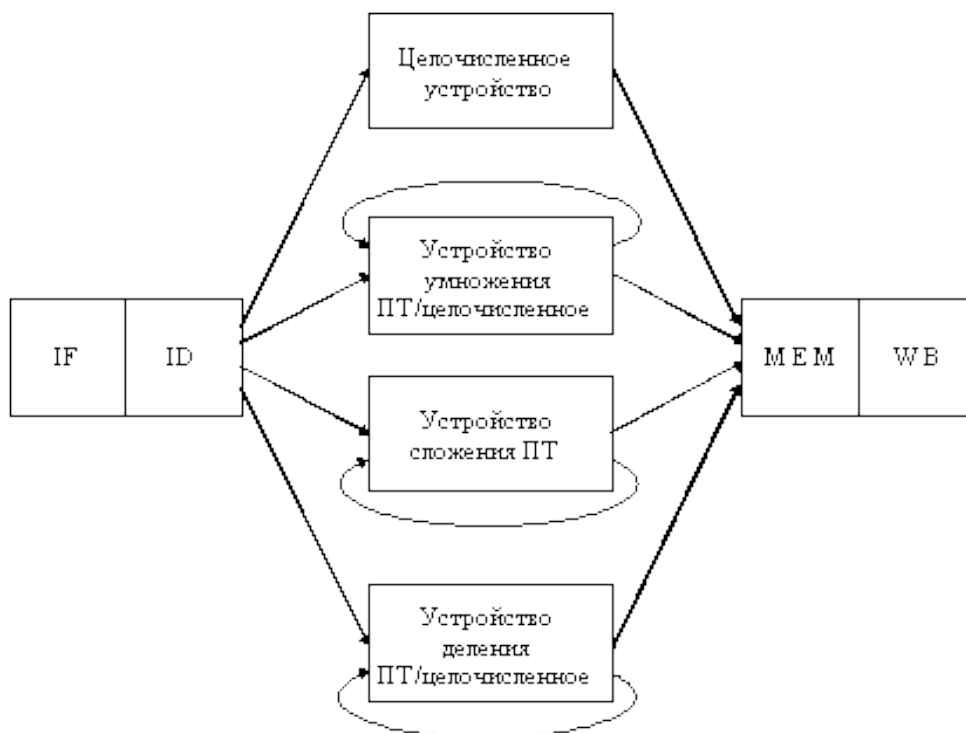


Рис. 8.11. Конвейер с дополнительными функциональными устройствами

Целочисленное устройство обрабатывает все команды загрузки и записи в память при работе с двумя наборами регистров (целочисленных и с плавающей точкой), все целочисленные операции (за исключением команд умножения и деления) и все команды переходов.

Если предположить, что стадии выполнения других ФУ неконвейерные, то рис. 8.11 показывает структуру такого конвейера. Поскольку стадия EX является неконвейерной, никакая команда, использующая ФУ, не может быть выдана для выполнения до тех пор, пока предыдущая команда не покинет ступень EX. Более того, если команда не может поступить на ступень EX, весь конвейер за этой командой будет приостановлен.

В действительности промежуточные результаты, возможно, не используются циклически ступенью EX, как это показано на рис. 8.11, и ступень EX имеет задержки длительностью более одного такта. Можно обобщить структуру конвейера плавающей точки, допустив конвейеризацию некоторых ступеней и параллельное выполнение нескольких операций. Чтобы описать работу такого конвейера, необходимо определить задержки функциональных устройств, а также скорость инициаций или скорость повторения операций (скорость, с которой новые операции данного типа могут поступать в ФУ).

Например, предположим, что имеют место следующие задержки функциональных устройств и скорости повторения операций:

Функциональное устройство	Задержка	Скорость повторения
Целочисленное АЛУ	1	1
Сложение с ПТ	4	2
Умножение с ПТ (и целочисленное)	6	3
Деление с ПТ (и целочисленное)	15	15

На рис. 8.12. представлена структура подобного конвейера. Ее реализация требует введения конвейерной регистровой станции EX1/EX2 и модификации связей между регистрами ID/EX и EX/MEM.

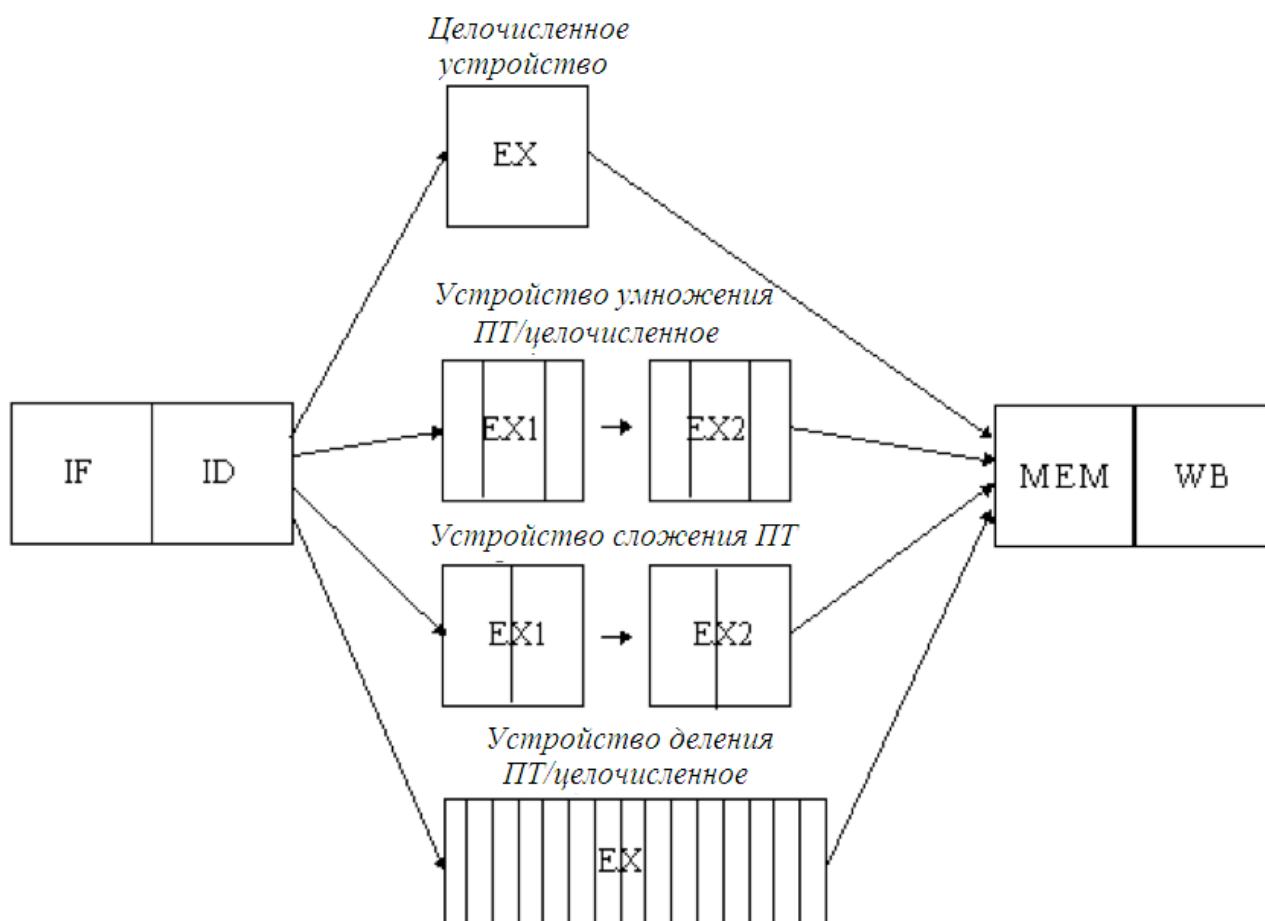


Рис. 8.12. Конвейер с многоступенчатыми функциональными устройствами

В заключение отметим несколько различных аспектов обработки конфликтов и организации ускоренных пересылок в длинных конвейерах:

1. Поскольку устройства не являются полностью конвейерными, в данной схеме возможны структурные конфликты. Эти ситуации необходимо обнаруживать и приостанавливать выдачу команд.
2. Поскольку устройства имеют разные времена выполнения, количество записей в регистровый файл в каждом такте может быть больше 1.
3. Возможны конфликты типа **WAW**, поскольку команды больше не поступают на ступень **WB** в порядке их выдачи для выполнения. Заметим, что конфликты типа **WAR** невозможны, поскольку чтение регистров всегда осуществляется на ступени **ID**.
4. Команды могут завершаться не в том порядке, в котором они были выданы для выполнения, что вызывает проблемы с реализацией прерываний.

Многотактные операции плавающей точки создают также новые проблемы для механизма прерывания.

P.S. См. презентации по теме из папки «Дополнительные материалы»