Лекция10. Режимы функционирования микропроцессорной системы: выполнение основной программы, вызов подпрограмм

Микропроцессорная система может находиться в одном из следующих режимов работы:

- выполнение основной программы;
- вызов подпрограмм;
- обработка прерываний и исключений;
- прямой доступ к памяти.

Выполнение основной программы. В этом режиме микропроцессор последовательно загружает из памяти команды и выполняет их. Для загрузки следующей команды микропроцессор хранит указатель (адрес) на нее в специальном регистре — *программном счетчике* (Program Counter — PC, или Instruction Pointer — IP). Содержимое этого регистра увеличивается на единицу каждый раз после выборки из памяти очередного слова* основной программы. При выполнении команд ветвления (безусловный и условный переходы, циклы) в счетчик загружается новое значение из кода адресации операндов, в результате процессор переходит к выполнению другой части программы. При использовании механизмов относительной адресации регистр PC является базовым регистром, к которому прибавляется адресная часть команды.

Очередная команда, загруженная из памяти команд, поступает в регистр команд в устройстве управления. Затем она дешифрируется, т. е. расшифровывается КОП, и вычисляются адреса операндов согласно КАД. Далее устройство управления в соответствии с операндом команды генерирует последовательность управляющих сигналов, которая в конечном счете приводит к выполнению поступившей команды и обеспечивает получение нужного результата.

Выборка каждой команды требует определенного количества тактов сигнала частоты. Если в системе команд процессора предусмотрены команды различной разрядности, то для выборки всей команды

*Разрядность слова зависит от разрядности шины данных общения с памятью. В самом простом случае слово равно одному байту.

необходимо знать ее общую длину. Поэтому в такой системе устройство управления считывает из памяти сначала первое слово команды, в котором содержится длина команды (непосредственно или косвенно в КОП), а затем остальную команду и помещает в регистр устройства управления. Применение системы команд с фиксированной разрядностью упрощает разработку процессора, поскольку отпадает необходимость в реализации механизмов определения длины команды.

Машинным (*процессорным*) *тактом* в микропроцессорных системах называют длительность тактовых сигналов T, которая задается тактовой частотой F микропроцессора. При выполнении операций, не требующих обращений к системной шине, эта частота определяет производительность микропроцессора.

Вызов подпрограмм. Для эффективного использования памяти дублирующиеся участки программы выделяют в отдельные подпрограммы. Для обращения к таким подпрограммам в систему команд процессора вводятся специальные команды вызова подпрограмм и возвращений из них (обычно обозначаемые CALL и RET соответственно). Команда CALL в коде адреса содержит указатель на первый байт подпрограммы (или смещение, если используется относительная адресация), который загружается в регистр PC, обеспечивая на следующем такте выборку первой команды из подпрограммы. Предварительно текущее значение программного счетчика сохраняется для последующего возврата в основную программу. При поступлении команды RET процессор восстанавливает предыдущее значение регистра PC и продолжает выполнение основной программы с команды, следующей за командой CALL.

Основная проблема при реализации механизма вызова подпрограмм — необходимость обеспечить возможность последовательного вызова вложенных подпрограмм (рис. 2.5.1). При вложенном вызове подпрограмм процессор должен запоминать адрес возврата каждой подпрограммы, при этом количество вложений не должно быть ограничено.

Для реализации такой возможности существует несколько подходов, в частности, использование стека. *Стек* (от англ. *stack* — стопка) — это структура данных, работающая по принципу «последним пришел — первым вышел» (Last In, First Out — LIFO).

Стековая память может различаться по способу реализации: регистровый стек или стек на основе ОЗУ.

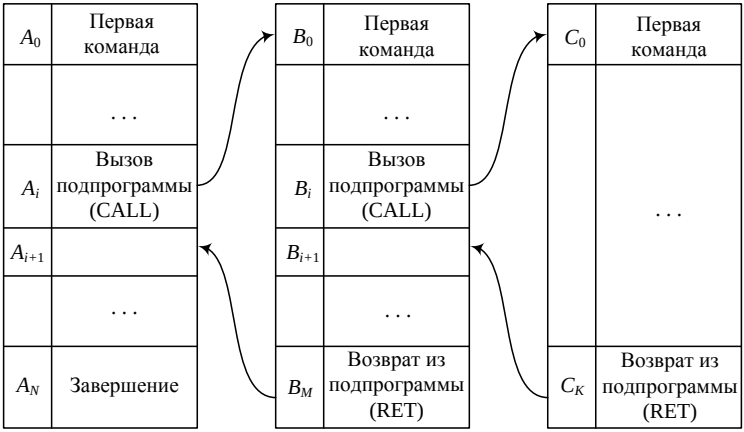

Рис. 2.5.1. Последовательный вызов вложенных подпрограмм

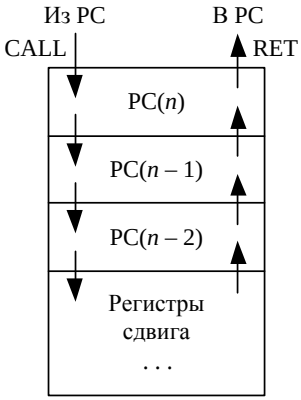

Рис. 2.5.2. Регистровый стек

Регистровый стек состоит из набора регистров (рис. 2.5.2), данные по которым могут перемещаться как в одну, так и в другую сторону. Команда CALL вызывает запись содержимого регистра PC в вершину стека, одновременно сдвигая остальные данные вниз. Команда RET, напротив, считывает содержимое самого верхнего регистра в PC, сдвигая остальные данные вверх. Количество регистров в стеке (называемое *глубиной стека*) определяет максимальное количество вложенных подпрограмм. Если все регистры заняты и при этом снова происходит вызов подпрограммы, то последняя точка возврата теряется, и процессор не может вернуться к основной программе. Обычно в микропроцессорных системах с такой организацией стека при исчерпании всех ячеек генерируется внутреннее исключение, обработчик которого может выгрузить дан-

ные из стека в ОЗУ. Такой подход снимает ограничение на количество вложенных подпрограмм, но требует от программиста написания дополнительных секций программы, отвечающих за обработку исключения. Если число вложений подпрограмм ограничено на уровне проектирования программы и не превышает глубину стека, то никаких дополнительных мер предпринимать не нужно.

Стек на основе ОЗУ (рис. 2.5.3) — более совершенное решение стековой памяти. В этом случае в ОЗУ выделяется отдельный участок для работы в качестве стека. Адресация к ячейкам стека производится с помощью специального регистра — указателя стека (Stack Pointer — SP), который входит в состав УУ либо является одним из регистров общего назначения. Регистр SP содержит адрес последней (самой верхней) заполненной ячейки стека, в которой хранится значение PC, записанное командой CALL. При переходе на новую подпрограмму значение SP увеличивается на единицу, указывая на новую еще не заполненную ячейку памяти. После этого в нее записывается текущее значение PC. Таким образом происходит последовательное заполнение стека «снизу-вверх», при этом регистр SP всегда указывает на вершину стека. При выходе из подпрограммы с помощью оператора RET в регистр PC записывается значение, хранящееся в вершине стека, а регистр SP декрементируется.

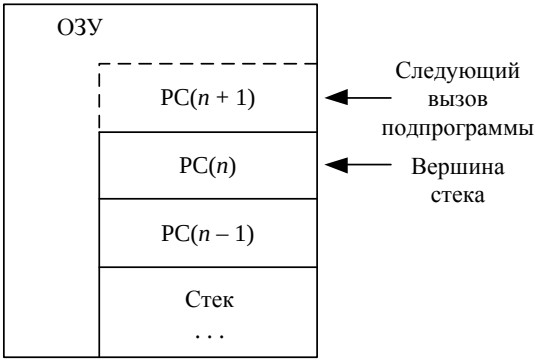

Рис. 2.5.3. Стек на основе ОЗУ

Поскольку ОЗУ, как правило, имеет значительный объем, то для стека можно выделить достаточно большое количество ячеек, обес-

печивая необходимый уровень сложения подпрограмм. Вместе с тем этот способ реализации стековой памяти имеет недостаток, связанный с необходимостью доступа к памяти всякий раз при входе в подпрограмму или выходе из нее. Это замедляет работу программы, особенно при большом количестве маленьких подпрограмм.

Для снижения накладных расходов на работу со стеком при вызове подпрограмм в некоторых системах получил распространение *метод регистрового окна*, разработанный в университете Беркли (США) и примененный в архитектуре системы команд SPARC, а позднее в процессоре Intel i960.

Предположим, что блок РОН состоит из K регистров, расположенных по кругу (рис. 2.5.4). Программе доступна лишь часть регистров L, составляющих текущее регистровое окно. Окно делится на три части: L_1 — входные регистры, L_2 — локальные регистры, L_3 — выходные регистры. При переходе в подпрограмму, регистровое окно сдвигается таким образом, что выходные регистры предыдущей активной программы становятся входными регистрами текущей. Как следствие, сохраняется в неизменном виде текущее состояние программы, а частичное перекрытие окон дает возможность передачи входных параметров запускаемой подпрограмме.

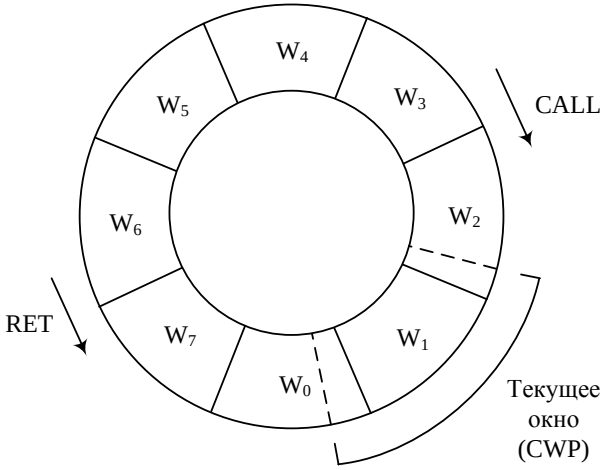

Рис. 2.5.4. Блок РОН с восемью регистровыми окнами

Указатель на начало текущего окна хранится в специальном регистре CWP (Current Window Pointer). В случае исчерпания лимита окон генерируется исключение и необходимо выгрузить содержимое первого регистрового окна в ОЗУ (так же, как это делается в регистровом стеке).

Обработка прерываний и исключений. При работе микропроцессорной системы появляются события (например ввод данных от ВУ, или сработавший системный таймер), которые требуют немедленной реакции на них, для чего необходимо прервать текущую выполняемую программу и запустить подпрограмму обработки возникшего события. Такие события называются *прерываниями* (interrupt), или исключениями.

Механизм обработки прерываний похож на механизм перехода в подпрограмму. При появлении сигнала прерывания процессор должен сохранить текущее состояние (*контекст*):

- регистр PC для обеспечения возможности возврата к выполнению основной программы;
- регистры состояния процессора (флаги АЛУ и т. п.);
- регистры общего назначения;
- специальные регистры.

После этого процессор передает управление обработчику прерывания, который производит необходимые действия для обеспечения реакции системы на прерывание и возвращает управление основной программе. Затем процессор возвращает контекст основной программы и продолжает выполнение с точки останова.

Прямой доступ к памяти. Для выполнения рутинных операций пересылки массивов данных между ОЗУ и каким-либо внешним устройством применяется специальный режим прямого доступа к памяти (ПДП). Для этого в современные процессоры встраивают отдельный контроллер ПДП, связанный общей шиной с ОЗУ и периферийными устройствами.

Когда процессору необходимо переслать некоторый объем данных, он программирует контроллер ПДП на выполнение этой задачи. Процесс программирования состоит в передаче контроллеру трех значений: адреса—источника данных, адреса—приемника данных и количества передаваемых слов. Затем контроллер сам начинает передачу данных, а процессор в это время может выполнять основную программу.

Контроллер ПДП может иметь внешние сигналы запросов от периферийных устройств. При появлении такого сигнала на входе контроллер начинает копировать данные по заложенной процессором заранее программе. После завершения копирования контроллер ПДП генерирует сигнал прерывания, сообщая процессору, что данные скопированы и с ними можно работать.

Применение режима ПДП позволяет избавить процессор от выполнения задач пересылки данных, которые могут быть весьма длительными, особенно при общении с медленными периферийными устройствами.

Учитывая материал предыдущих лекций, рассмотрим типовую структуру современной микропроцессорной системы, построенной по магистрально-модульному принципу (рис. 2.5.5). Все отдельные устройства (модули), входящие в состав такой системы, обмениваются данными по общей системной магистрали.

Основным модулем такой системы является микропроцессор, содержащий устройство управления, одно или несколько операционных устройств и блок РОН. Модуль ОЗУ необходим для хранения выполняемой программы (или ее части) и обрабатываемых данных. ПЗУ предназначено для хранения выполняемой программы и констант. В процессорах общего назначения в ПЗУ хранится *загрузчик* — небольшая программа, которая загружает из внешнего запоминающего устройства в ОЗУ основную программу. В процессорах цифровой обработки сигналов в ПЗУ могут храниться таблицы часто используемых арифметических функций, например синусов.

Внешние устройства подключаются к общей шине посредством *периферийных адаптеров* (ПА), реализующих протоколы параллельного или последовательного обмена. В однокристальных микропроцессорных системах внешними устройствами могут быть различные таймеры, встроенные блоки АЦП и ЦАП, контроллеры внешней памяти, интерфейсы USB и др. Многие внешние устройства могут генерировать сигналы прерываний и запросы ПДП. Для реализации всех возможностей таких устройств к общей шине подключаются также контроллер прерываний и контроллер ПДП.

Структура системы с общей шиной упрощает разработку и программирование процессора. Доступ к памяти и внешним устройствам осуществляется с помощью одних и тех же команд пересылки. С другой стороны, такая реализация является достаточно медленной, по-

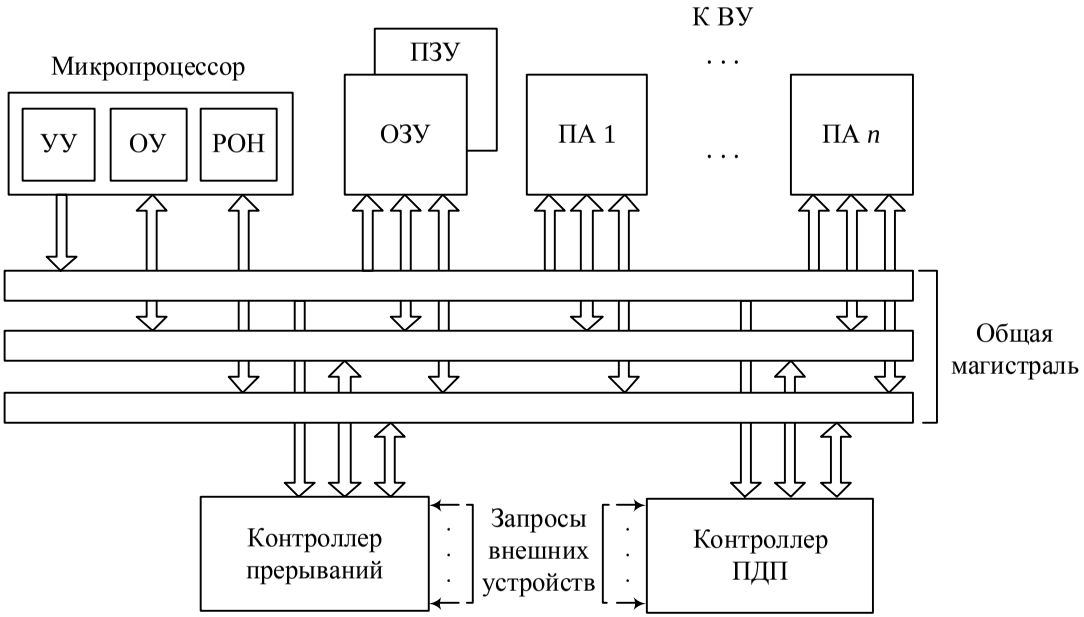

Рис. 2.5.5. Типовая структура микропроцессорной системы

скольку обмен по магистрали происходит в несколько тактов, в течение которых процессор вынужден простаивать в ожидании ответа устройства.

Контрольные вопросы

1. Перечислите основные режимы функционирования микропроцессорной системы.
2. Опишите режим выполнения основной программы.
3. Какие шаги предпринимает процессор для перехода к выполнению подпрограммы?
4. Что такое стек? Какие существуют варианты его реализации?
5. Что такое прерывание?
6. В чем суть режима прямого доступа к памяти?

Литература

1. Микропроцессоры. В 3-х кн. Кн. 1. Архитектура и проектирование микроЭВМ: учебник для втузов / ***Под ред. Л.Н. Преснухина***. — М.: Высшая школа, 1986. — 495 с.

2. Микропроцессорные системы: учеб. пособие для вузов / ***Е.К. Александров, Р.И. Грушвицкий, М.С. Куприянов и др.***; ***под ред. Д.В. Пузанкова***. — СПб.: Политехника, 2002. — 935 с.