**Тема 25. Способы адресации в ЭВМ**

**Введение**

Адресный код - это информация об адресе операнда, содержащаяся в команде. Исполнительный адрес - это номер ячейки  памяти,  к  которой производится фактическое обращение. В современных ЭВМ адресный код,  как правило, не совпадает с исполнительным адресом. Разработка кодов операций является важной частью архитектуры команд. Однако значительное число битов программы используется для того, чтобы определить, откуда нужно брать операнды, а не для того, чтобы узнать, какие операции нужно выполнить.

Следует различать понятия адресный код в команде и исполнительный адрес. Адресный код (Ак) – это информация об адресе операнда, содержащаяся в команде. Исполнительный адрес (Аи) – это номер ячейки памяти, к которой производится фактическое обращение. В современных ЭВМ адресный код, как правило, не совпадает с исполнительным адресом.

Для различных применений ЭВМ разработано большое число способов адресации. Способом адресации называется правило определения адреса и операнда на основе информации, указанной в команде.

Выбор способов адресации, формирование исполнительного адреса и преобразование адресов является одним из важнейших вопросов разработки любой ЭВМ. Эффективность способа адресации характеризуется двумя показателями: затратами оборудования и затратами времени на доступ к адресуемой информации.

Затраты оборудования определяются суммой затрат элементов аппаратуры на обработку адресов в процессоре и затратой памяти на хранение адресов, указываемых в команде программы.

Затраты времени принято характеризовать числом обращений к основной ОП, выполняемых с целью выборки или записи операнда.

Рассмотрим основные способы адресации информации, используемые в современных ЭВМ различных классов.

Выбор способов адресации, формирования исполнительного адреса и преобразования адресов является одним из важнейших  вопросов разработки ЭВМ.   В настоящее время используются различные способы адресации, наиболее распространенные из которых рассматриваются ниже.

**Способы адресации в ЭВМ**

***1. Непосредственная адресация***

Самый простой способ определения операнда — содержать в адресной части сам операнд, а не адрес операнда или какую-либо другую информацию, описывающую, где находится операнд. Такой операнд называется **непосредственным операндом,** поскольку он автоматически вызывается из памяти одновременно с командой; следовательно, он сразу непосредственно становится доступным. Один из вариантов команды с непосредственным адресом для загрузки в регистр R1 константы 4 показан на рис. 1



Рис.1. Команда с непосредственным адресом для загрузки константы 4 в регистр 1

При непосредственной адресации не требуется дополнительного обращения к памяти для вызова операнда. Однако у такого способа адресации есть и некоторые недостатки. Во-первых, таким способом можно работать только с константами. Во-вторых, число значений ограничено размером поля. Тем не менее эта технология используется во многих архитектурах для определения целочисленных констант.

Непосредственная адресация сокращает время выполнения команды, так как не требуется обращение к памяти за операндом. Кроме того, экономится память, поскольку отпадает необходимость в ячейке для хранения операнда. В плане эффективности этот способ можно считать «идеальным» , и его можно рекомендовать к использованию во всех ситуациях, когда тому не препятствуют вышеупомянутые ограничения.

***2. Прямая адресация***

Прямая адресация – это основополагающий способ адресации в ЭВМ. В случае прямой адресации исполнительный адрес совпадает с адресной частью команды. Прямой адрес – это номер ячейки памяти, в которой хранится операнд. Прямой адрес представляется в команде m -разрядным полем, где m = log 2 E (E – емкость адресной памяти, исчисляемая в машинных элементах – словах или байтах).

Схема прямой адресации может быть представлена в следующем виде:

КОП Адрес операнда

ателя

ОЗУ

Операнд

Рис.2. Схема прямой адресации

В настоящее время в чистом виде этот способ находит ограниченное применение из-за трудностей: перемещения программ в памяти; обработки массивов данных и организации специальных видов памяти; обработки данных, организованных в списочные структуры и т.п.

При всей простоте использования способ имеет существенный недостаток -ограниченный размер адресного пространства, так как для адресации к памяти большой емкости нужно «длинное» адресное поле. Однако более существенным несовершенством можно считать то, что адрес', команде, не может быть изменен в процессе вычислений (во всяком случае, такое изменение не рекомендуется) . Это ограничивает возможности по произвольному размещению программы в памяти.

***3. Косвенная адресация***

При программировании некоторых задач широко используются указатели, определяющие местоположение операндов в памяти. Указатель – это адрес операнда, хранимый в определенной ячейке памяти. Значение указателя может меняться в процессе выполнения программы по мере обработки одних операндов и перехода к обработке других. Но адрес самого указателя остается постоянным, обеспечивая на каждом этапе вычислений обращение к соответствующему операнду. В таком случае адрес операнда определен в ячейке с заданным адресом.

Адрес, определяющий адрес операнда, называется косвенным адресом. А адресация к операнду через цепочку указателей – косвенной адресацией. Количество указателей в цепочке – кратность косвенной адресации.

На практике кратность не превышает 6 – 8, т.к. при большей кратности программирование значительно затрудняется. Частным случаем косвенной адресации с кратностью 0 может рассматриваться прямая адресация.

Наиболее распространена однократная косвенная адресация, схема которой может быть представлена в виде:

КОП Адрес указателя

ОЗУ

Адрес

операнда

Операнд

Рис.3.Схема косвенной адресации

Постоянство адреса указателя при этой адресации при изменении адреса операнда в процесе выполнения программы обеспечивает возможность переадресации данных и обработку массивов или сложной структуры данных с помощью одной и той же программы или участка программы.

Команда с косвенной адресацией делает дополнительные обращения к памяти за косвенным адресом и выполняется дольше, чем команда с прямой адресацией. Но применение команд с косвенной адресацией выгоднее, т.к. аналогичные действия с использованием только прямой адресации требуют выполнения целой программы.

Признак применения косвенной адресации указывается, как правило, в КОП коман- ды. При большой кратности адресации в некоторых ЭВМ адресные поля команды содержат разряд – указатель косвенной адресации (УА).

Программа, которая сама изменяется подобным образом, называется самоизменяющейся программой.Эта идея была предложена Джоном фон Нейманом и применялась в старых компьютерах, где не было косвенной регистровой адресации. В настоящее время самоизменяющиеся программы считаются неудобными и очень трудными для понимания. Кроме того, их выполнение нельзя разделить между несколькими процессорами. Они даже не могут правильно выполняться на машинах с разделенной кэш-памятью первого уровня, если в кэш-памяти команд нет специальной схемы для обратной записи (поскольку разработчики предполагали, что программы сами себя не изменяют).

В качестве варианта косвенной адресации, правда, достаточно редко используемого, можно упомянуть многоуровневую или каскадную косвенную адресацию, когда к исполнительному адресу цепочка косвенных адресов. В этом случае один из битов в каждом адресе служит признаком косвенной адресации. Состояние бита указывает, является ли содержимое ячейки очередным адресом в цепочке адресов или это уже исполнительный адрес операнда. Особых преимуществ у такого подхода нет, но в некоторых специфических ситуациях он оказывается весьма удобным, например при обработке многомерных массивов. В то же время очевиден и его недостаток — для доступа к операнду требуется три и более обращений к памяти.

***4. Относительная адресация***

Для эффективного использования основной памяти необходимо обеспечить возможность размещения массивов данных и программ в любом месте памяти, начиная от любого адреса. За счет этого появляется возможность динамического распределения памяти, при котором для каждого массива выделяется область памяти, равная не максимальной, а фактической длине массива, что приводит к уменьшению размера области памяти, выделяемой программе. Возможность размещения информации в любой области памяти необходима для обеспечения мультипрограммного режима работы ЭВМ, при котором для программы может быть выделена любая свободная область памяти. Для обеспечения этого используется относительная адресация информации.

Основная идея относительной адресации в том, чтобы в команде указывать не абсолютный адрес операнда, а смещение его относительно начала программного модуля. Все исполнительные адреса (кроме непосредственных) при относительной адресации представляются в виде двух полей. Исполнительный адрес операнда определяется суммой

АИ = [B] + D , где

[B] – содержимое ячейки внутренней памяти с адресом В.

Схему относительной адресации можно представить в виде

КОП

В

D

**.**

**.**

**.**

базовый

адрес

**.**

**.**

**.**

операнд

команда

начало

модуля

программный

модуль

базовые

регистры процессора

Рис.4.Схема относительной адресации

При использовании относительной адресации адреса операндов задаются значениями D = 0, 1,…, 2m-1 относительно произвольных базовых адресов, которые определяются только косвенно – адресами В = 0, 1, …, Q ячеек внутренней памяти, содержащих значения базовых адресов. Базовые адреса (начальные адреса программных модулей или массивов) загружаются в соответствующие ячейки внутренней памяти В при загрузке программы в основную память. При выполнении программы относительные адреса, указанные в командах, обрабатываются процессором путем выборки из внутренней памяти значения базового адреса [B] и сложения его со смещением D. При любом местоположении программы в памяти команды не изменяются, а изменяются лишь значения в ячейках внутренней памяти, отведенных для хранения базовых адресов.

В целях сокращения емкости внутренней памяти процессора, на каждом этапе вычислений используется ограниченное число адресов – например, один базовый адрес для указания начала выполняемой программы (или модуля) и базовые адреса для адресации массивов данных, количество которых определяется структурой данных. Считается, что в ЭВМ общего назначения целесообразно применение четырех-восьми базовых адресов.

При относительной адресации в качестве базового регистра можно использовать и счетчик команд. В этом случае не требуется отдельного регистра для хранения базового адреса и не нужно осуществлять перезагрузку. Недостатком адресации относительно счетчика команд является то, что при использовании одного и того же операнда в разных командах требуется указывать разную величину смещения.

***5. Стековая адресация и польская инверсная запись***

Правила “последний пришел – первый ушел” при обращении к стеку реализуется автоматически, и поэтому при операциях со стеком возможно безадресное задание операнда. Команда в этом случае не содержит адреса ячейки стека, но содержит адрес (или он подразумевается) ячейки памяти или регистра, откуда слово передается в стек или куда загружается из стека.

Механизм стековой адресации можно пояснить следующими рисунками

При выполнении команды передачи в стек слова из регистра или ячейки ОП сначала УК увеличивается на 1, а затем слово помещается в ячейку стека, указываемую УС. При команде загрузки из стека регистра или ячейки памяти сначала слово извлекается из вершины стека, а затем указатель стека уменьшается на 1. При соответствующем расположении операндов в стеке можно вычислять выражения полностью безадресными командами, указывающими только вид операции. Такая команда извлекает из стека в соответствии с КОП один или два операнда, выполняет над ними предписанную операцию и заносит результат в стек.

Вычисления с использованием стековой памяти удобно описывать и программировать с помощью плоской инверсной записи арифметических выражений. Эта запись производится последующему правилу

* читается арифметическое выражение слева направо и последовательно друг за другом выписываются встречающиеся операнды; как только окажется, что все операнды некоторой операции выписаны, запивается знак этой операции и далее продолжается выписывание операндов; если операция имеет операндом результат некоторой предыдущей операции и знак последней выписан, то считается этот операнд выписанным.

Для примера рассмотрим польскую запись выражения: (k+l-m)(p-s)

Она имеет следующий вид kl+m-ps-x

Из примера видно, что польская запись не содержит скобок, но порядок действий определен однозначно.

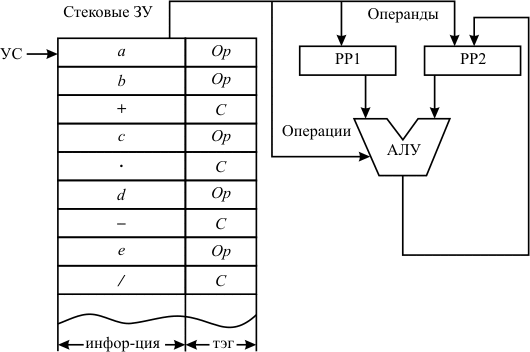


Рис.5.Схема стековой адресации

При использовании стековой памяти последовательность символов в польской записи может рассматриваться как программа вычислений исходного арифметического выражения, если под буквами понимать команды засылки, содержащие только адреса в ОП соответствующих операндов, засылаемых в стек, а под знаками операций подразумевать безадресные команды, содержащие только коды операций

Безадресные команды на основе стековой адресации предельно сокращают форматы команд, экономят память и способствуют повышению производительности ЭВМ.

В современной архитектуре процессора (МП) стек и стековая адресация широко используются при организации переходов к подпрограммам и возврате от них, а также в системах прерывания.

***6. Регистровая адресация***

Для сокращения размера адресного поля в команде и, как следствие, уменьшения числа обращений в ОП в современных ЭВМ стали широко использовать для адресации регистры внутренней памяти процессора. Это: регистровая прямая, регистровая косвенная, индексная адресации, адресация через программный счетчик.

При прямой регистровой адресации в регистре хранится операнд, а в команде указывается короткий номер регистра. При косвенной – в регистре находится адрес операнда. При индексной адресации в отличие от относительной в регистре хранится индекс, а адресное поле команды указывает адрес в оперативной памяти.

**В** архитектурах с загрузкой с запоминанием, например UltraSPARC II, практически все команды используют исключительно этот способ адресации. Он не используется только в том случае, когда операнд перемещается из памяти в регистр (команда **LOAD)** или из регистра в память (команда STORE). Даже в этих командах один из операндов является регистром — туда отправляется слово из памяти **или** оттуда перемещается слово в память.

# *7. Адресация со смещением*

При адресации со смещением исполнительный адрес формируется в результате суммирования содержимого адресного поля команды с содержимым одного или нескольких регистров процессора (рис. 26).

Адресация со смещением предполагает, что адресная часть команды включает в себя как минимум одно поле (Aк). В нем содержится константа, смысл которой в разных вариантах адресации со смещением может меняться. Константа может пред­ставлять собой некий базовый адрес, к которому добавляется хранящееся в регистре смещение. Допустим и прямо противоположный подход: базовый адрес находится в регистре процессора, а в поле Aк указывается смещение относительно этого адре­са. В некоторых процессорах для реализации определенных вариантов адреса­ции со смещением предусмотрены специальные регистры, например базовый или индексный. Использование таких регистров предполагается по умолчанию, поэтому адресная часть команды содержит только поле Ак.

Если же составля­ющая адреса может располагаться в произвольном регистре общего назначения, то для указания конкретного регистра в команду включается дополнительное поле R (при составлении адреса более чем из двух составляющих в команде будет несколь­ко таких полей). Еще одно поле R может появиться в командах, где смещение пе­ред вычислением исполнительного адреса умножается на масштабный коэффи­циент. Такой коэффициент заносится в один из РОН, на который и указывает это дополнительное поле. В наиболее общем случае адресация со смещением подразу­мевает наличие двух адресных полей: Ак и R.

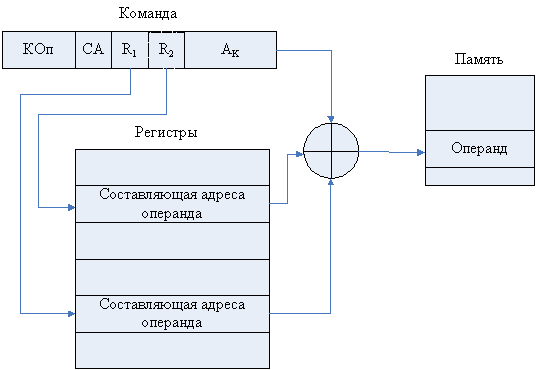


Рис. 26. Адресация со смещением.

 В рамках адресации со смещением имеется еще один вариант, при котором ис­полнительный адрес вычисляется не суммированием, а конкатенацией (присоединением) составляющих адреса. Здесь одна составляющая представляет собой старшую часть исполнительного адреса, а вторая — младшую.

Ниже рассматриваются основные способы адресации со смещением, каждый из которых, впрочем, имеет собственное название.

**Заключение**

На практике для эффективной архитектуры команд вовсе не требуется большого количества различных способов адресации. Поскольку практически весь код, написанный на этом уровне, будет порождаться компиляторами, способов адресации должно быть мало, и они должны быть четкими и ясными. Машина должна предлагать либо все возможные варианты, либо только один вариант. В остальных промежуточных случаях может оказаться так, что компилятор не способен сделать выбор. Поэтому самые простые архитектуры используют очень небольшое число способов адресации, причем на каждый из этих способов накладываются жесткие ограничения. Обычно практически для любых применений достаточно непосредственной, прямой, регистровой и индексной адресации. Каждый регистр (включая указатель локальных переменных, указатель стека и счетчик программ) должен быть пригоден к употреблению всякий раз, когда этот регистр требуется. Более сложные способы адресации могут сократить число команд, но при этом придется ввести последовательности операций, которые трудно будет выполнять параллельно с другими последовательными операциями.

Мы рассмотрели возможные компромиссы между кодами операций и адресами и между различными способами адресации. Когда вы сталкиваетесь с новым компьютером, вы должны изучить все команды и способы адресации не только для того, чтобы знать, какие из них имеются в наличии, но и для того, чтобы понять, почему был сделан именно такой выбор и каковы были бы последствия при другом выборе.

Литература:

1. Бройдо В.Л., Ильина О.П. Вычислительные системы, сети и телекоммуникации. – 4-е изд. – СПб.: Питер, 2011. – 560 с.
2. Ильина, О.П. Архитектура ЭВМ и систем: учебник для вузов / О.П. Ильина, В.Л. Бройдо. -2-е изд. - СПб. : Питер, 2010. - 720 с.
3. Орлов, С.В. Организация ЭВМ и систем : учебник для вузов / С.В. Орлов, Б.А. Цилькер. - 2-е изд. - СПб. : Питер, 2011. - 688 с.
4. Хамахер, З. Вранешич, С. Организация ЭВМ. 5-е изд. /к. Заки. - Спб.: Питер; Киев: Издательская группа BHV, 2003 - 848 с.
5. Архитектура ЭВМ  [Электронный ресурс] : курс лекций КАИ: http://kailib.ru/arhitevm
6. Организация ЭВМ [Электронный ресурс] : курс лекций КАИ: http://kailib.ru/part1