**Задание для группы 136 на 06.02.2022**

**Задание пересылать мне на электронную почту: param\_e@mail.ru**

**Задание** также можно найти на google-диске по ссылке:

[**https://drive.google.com/drive/folders/1OdfZ8PIZeNUsoYyyX5TUltvCSpFfm64w?hl=ru**](https://drive.google.com/drive/folders/1OdfZ8PIZeNUsoYyyX5TUltvCSpFfm64w?hl=ru)

**Конспект лекции высылаете до 20-00 часов 05.02.2022 (конспект в тетради; высылаете фотографии конспекта со своей подписью на каждом листе).**

**Без подписи не принимаю конспект.**

Ответы на вопросы в конце лекции, если они есть (ответы в тетради после конспекта).

Если конспект выслан позже указанного времени, то баллы снижаются за каждый час просрочки на 0,5.

**Лекция. Классы процессоров и арифметико-логическое устройство.**

В зависимости от набора и порядка выполнения команд процессоры подразделяются на четыре класса, отражающих также последовательность развития ЭВМ. Ранее других появились процессоры **CISC.** Затем, с целью повышения быстродействия процессоров были разработаны процессоры **RISC,** которые характеризуются сокращенным набором быстро выполняемых команд. Ряд редко встречающихся команд процессора CISC выполняется последовательностями команд процессора RISC. Позже появилась концепция процессоров MISC, использующая минимальный набор длинных команд. Вслед за ними возникли процессоры VLIW, работающие со сверхдлинными командами.

**CISC** (complex instruction set computer) - традиционная архитектура, в которой ЦП использует микропрограммы для выполнения исчерпывающего набора команд. Они могут иметь различную длину, методы адресации и требуют сложных электронных цепей для декодирования и исполнения. В течение долгих лет производители компьютеров разрабатывали и воплощали в изделиях все более сложные и полные системы команд.

Основоположником CISC-архитектуры считается фирма IBM с архитектурой IBM/360. Здесь выполнение любой сколь угодно сложной команды из системы команд процессора реализовывается аппаратно, внутри самого процессора. В данной архитектуре стремятся иметь отдельную машинную команду для каждого возможного (типового) действия по обработке данных.

Совершенствование процессоров шло по пути создания ВМ, способных выполнять как можно больше разных команд. Это упрощало работу программистов, которые писали программы на языке ассемблера (то есть практически на уровне машинных команд). Использование сложных команд позволяло сократить размер и время разработки программы.

В итоге сложились следующие черты организации CISC-процессоров:

* большое количество различных машинных команд (сотни), каждая из которых выполняется за несколько тактов центрального процессора;
* устройство управления с программируемой логикой;
* небольшое количество регистров общего назначения;
* различные форматы команд с разной длиной;
* преобладание двухадресной адресации;
* развитый механизм адресации операндов, включающий различные методы косвенной адресации.

CISC-подход, однако, привел к тому, что некоторые команды стало невозможно выполнять чисто аппаратными средствами (при разумной сложности таких средств). В результате в процессорах появились блоки, «на лету» заменяющие наиболее сложные команды последовательностями из более простых команд. Из-за высокой сложности команд и их обилия устройство управления ВМ приходилось строить только на основе программируемой логики, то есть с применением «медленной» управляющей памяти. Последнее обстоятельство существенно ограничивало возможности наращивания тактовой частоты процессора.

Анализ работы процессоров показал, что в течение примерно 80 % времени выполняется лишь 20 % большого набора команд. Поэтому была поставлена задача оптимизации выполнения небольшого по числу, но часто используемых команд.

В то же время целый ряд несомненных достоинств CISC-архитектуры сохраняют ее актуальность (прежде всего, в глазах разработчиков программных приложений). Именно поэтому ведущие фирмы-производители ВМ (Intel, AMD, IBM и др.) в своих последних разработках, по-прежнему, не отказываются от CISC-подхода.

В 1974 г. John Cocke-Джон Кок (IBM Research) решил испробовать подход, который мог бы существенно уменьшить количество машинных команд в ЦП. В середине 70-х это привело многих производителей компьютеров к пересмотру своих позиций и к разработке ЦП с весьма ограниченным набором команд.

**RISC** (Redused Instuction Set Computer) - процессор, функционирующий с сокращенным набором команд, т.е. наиболее употребляемыми простейшими командами. Так, в процессоре CISC для выполнения одной команды необходимо в большинстве случаев 10 и более тактов. Что же касается процессоров RISC, то они близки к тому, чтобы выполнять по одной команде в каждом такте. Следует также иметь в виду, что благодаря своей простоте процессоры RISC не патентуются. Это также способствует их быстрой разработке и широкому производству. Между тем, в сокращенный набор RISC вошли только наиболее часто используемые команды.

Первый процессор RISC был создан корпорацией IBM в 1979 г. и имел шифр IBM 801. В настоящее время процессоры RISC получили широкое распространение. Современные процессоры RISC характеризуются следующим:

• упрощенный набор команд, имеющих одинаковую длину;

• большинство команд выполняются за один такт процессора;

• отсутствуют макрокоманды, усложняющие структуру процессора и уменьшающие скорость его работы;

• взаимодействие с оперативной памятью ограничивается операциями пересылки данных;

• резко уменьшено число способов адресации памяти (не используется косвенная адресация);

• используется *конвейер* команд, позволяющий обрабатывать несколько из них одновременно;

• применяется высокоскоростная память

• простая структура устройства управления.

Новый подход к архитектуре процессора значительно сократил площадь, требуемую для него на кристалле интегральной схемы. Это позволило резко увеличить число регистров. В современном процессоре RISC уже используется более 100 регистров. В результате процессор на 20÷30 % реже обращается к оперативной памяти, что также повысило скорость обработки данных. Упростилась топология процессора, выполняемого в виде одной интегральной схемы, сократились сроки ее разработки, она стала дешевле.

Начиная с процессора Pentium корпорация Intel начала внедрять элементы RISC-технологий в свои изделия.

Недостатки RISC прямо связаны с некоторыми преимуществами этой архитектуры. Принципиальный недостаток - сокращенное число команд: на выполнение ряда функций приходится тратить несколько команд вместо одной в CISC. Это удлиняет код программы, увеличивает загрузку памяти и трафик команд между памятью и ЦП. Исследования показали, что RISC-программа в среднем на 30% длиннее CISC-программы, реализующей те же функции.  
Хотя большое число регистров дает существенные преимущества, само по себе оно усложняет схему декодирования номера регистра, тем самым увеличивается время доступа к регистрам.  
Устройство управления с аппаратной логикой, реализованное в большинстве RISC-систем, менее гибко, более склонно к ошибкам, затрудняет поиск и исправление ошибок, уступает при выполнении сложных команд.

**Процессор VLIW** (Very Long Instruction Word - очень длинная машинная команда) - процессор, работающий с системой команд сверхбольшой разрядности.

Идея технологии VLIW заключается в том, что создается специальный компилятор планирования, который перед выполнением прикладной программы проводит ее анализ, и по множеству ветвей последовательности операций определяет группу команд, которые могут выполняться параллельно. Каждая такая группа образует одну сверхдлинную команду. Это позволяет решать две важные задачи. Во-первых, в течение одного такта выполнять группу коротких («обычных») команд. И, во-вторых, упростить структуру процессора. Этим технология VLIW отличается от суперскалярности. В последнем случае отбор групп одновременно выполняемых команд происходит непосредственно в ходе выполнения прикладной программы (а не заранее). Из-за чего усложняется структура процессора и замедляется скорость его работы.

Технология VLIW появилась в результате работ, проведенных корпорациями HP и Intel.

**Процессор MISC** - MISC processor, работающий с минимальным набором длинных команд.

Увеличение разрядности процессоров привело к идее укладки нескольких команд в одно слово (связку, bound) размером 128 бит. Оперируя с одним словом, процессор получил возможность обрабатывать сразу несколько команд. Это позволило использовать возросшую производительность компьютера и его возможность обрабатывать одновременно несколько потоков данных.

Процессор MISC, как и процессор RISC, характеризуется небольшим набором чаще всего встречающихся команд. Вместе с этим принцип команд VLIW обеспечивает выполнение группы команд за один цикл работы процессора. Порядок выполнения команд распределяется таким образом, чтобы в максимальной степени загрузить маршруты, по которым проходят потоки данных. Таким образом, архитектура MISC объединила вместе суперскалярную (много­поточную) и VLIW концепции. Компоненты процессора просты и работают с высокими скоростями.

**Арифметико-логическое устройство (АЛУ)**

Arithmetic and Logical Unit (ALU) - компонента процессора, выполняющая арифметические и логические операции над данными.

АЛУ реализует важную часть процесса обработки данных. Она заключается в выполнении набора простых операций. Арифметической операцией называют процедуру обработки данных, аргументы и результат которой являются числами (сложение, вычитание, умножение, деление). Логической операцией именуют процедуру, осуществляющую построение сложного высказывания (операции И, ИЛИ, НЕ, ...). АЛУ состоит из регистров, сумматора с соответствующими логическими схемами и блока управления выполняемым процессом. Устройство работает в соответствии с сообщаемыми ему именами (кодами) операций, которые при пересылке данных нужно выполнить над переменными, помещенными в регистр.

АЛУ классифицируются следующим образом:

*1. По способу действий над операндами:*

• АЛУ последовательного действия;

• параллельного действия.

В последовательных АЛУ действия над операндами производятся последовательно разряд за разрядом начиная с младшего. В параллельных АЛУ все разряды операндов обрабатываются одновременно.

*2. По виду обрабатываемых чисел* АЛУ могут производить операции над двоичными числами с фиксированной или плавающей запятой и над двоично-десятичными числами. В последнем случае каждая десятичная цифра записывается четырьмя разрядами двоичного кода:

2003 - 0010 0000 0000 0011

АЛУ при действии над двоично-десятичными числами должны содержать схему десятичной коррекции. Схема десятичной коррекции преобразует полученный результат таким образом, чтобы каждый двоично-десятичный разряд не содержал цифру больше 9.

При записи числа с фиксированной запятой запятая фиксируется после младшего разряда, если число целое, и перед старшим, если число меньше 1.

При записи чисел с плавающей запятой выделяется целая часть, которая называется мантиссой, и показатель степени, который характеризует положение запятой.

*3. По организации действий над операндами:*

• блочные;

• многофункциональные АЛУ.

В блочных АЛУ отдельные блоки предназначены для действий над двоично-десятичными числами, отдельно для действий над числами с фиксированной запятой, отдельно с плавающей запятой.

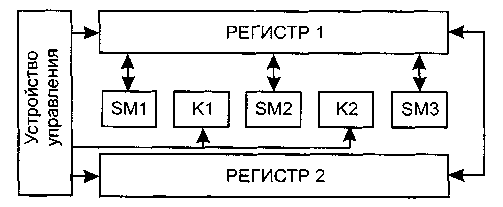
В многофункциональных АЛУ одни и те же блоки обрабатывают числа с фиксированной запятой, плавающей запятой и двоично-десятичные числа (рис. 1).

Рис. 1. Многофункциональное АЛУ.

Клапаны К1 и К2 объединяют сумматоры 1, 2 и 3 для действий над числами с фиксированной запятой.

Для действий над числами с плавающей запятой клапан К2 объединяет сумматоры 2 и 3 для обработки мантисс, а клапан К1 отсоединяет первый сумматор от второго. Сумматор 1 обрабатывает порядки.

*4. По структуре:*

• АЛУ с непосредственными связями;

• многосвязные.

В многосвязных АЛУ входы и выходы регистров приемников и источников информации подсоединяются к одной шине. Распределение входных и выходных сигналов происходит под действием управляющих сигналов.

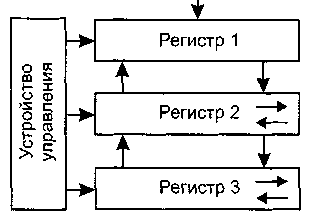
В АЛУ с непосредственной связью вход регистра приемника связан с выходом регистра источника операндов и регистра, в котором происходит обработка (рис. 2).

Рис. 2. АЛУ с непосредственной связью.

В этой схеме суммирование происходит так: операнды подаются в регистр 1. Регистр 2 является накапливающим сумматором или автоматом с памятью. Он суммирует слагаемые, поступающие в разные моменты времени, и передает результат в регистр 3.

Умножение в этом АЛУ происходит так: множимое помещают в регистр 4, множитель - в регистр 1. Регистры 2 и 3 являются, кроме того, сдвигающими регистрами. В зависимости от содержимого разряда множителя, множимое сдвигается на один разряд, если множитель содержит 1, и на два, если множитель содержит 0. Эти частные произведения суммируются в регистре 2.