ЭВМ Билеты Сжато

1. Понятие о структурной организации и архитектуре компьютерных систем

- **Вычислительная машина** - *комплекс технических и программных средств*, предназначенный для автоматизации подготовки и решения задач пользователей.
- **ЭВМ** – *ВМ*, основные функциональные устройства которой выполнены на электронных компонентах.
- **Вычислительная система** - *совокупность* взаимосвязанных и взаимодействующих процессоров или вычислительных машин, периферийного оборудования и программного обеспечения, предназначенная для подготовки и решения задач пользователей.
- **Архитектура ВМ** – логическая *структура* и функциональные *характеристики* ВМ, включая взаимосвязи между ее аппаратными и программными компонентами.

Под структурной организацией ЭВМ (электронной вычислительной машины) понимается некоторая физическая модель, *устанавливающая состав, порядок и принципы взаимодействия основных функциональных частей машины* (без излишних деталей их технической реализации).

2. Принципы концепции машины с хранимой в памяти программой

Основные принципы:

1. **Принцип двоичного кодирования** - вся информация, как данные так и команды, *кодируются двоичными цифрами 0 и 1*. Каждый тип информации представляется двоичной последовательностью и имеет свой формат.
2. **Принцип программного управления** - все *вычисления должны быть представлены в виде программы*, состоящей из последовательности управляющих слоев - *команд*, которые хранятся в последовательных ячейкам памяти ВМ и выполняются в естественной последовательности.
3. **Принцип однородности памяти** - команды и *данные* хранятся в одной и той же памяти и внешне в памяти *неразличимы*. Распознать их можно только по способу использования; то есть одно и то же значение в ячейке памяти может использоваться и как данные, и как команда, и как адрес в зависимости лишь от способа обращения к нему.
4. **Принцип адресности** - основная *память состоит из пронумерованных ячеек*, причем процессору в произвольный момент доступна любая ячейка, для этого

используется ее номер или адрес.

Принцип машины с хранимой в памяти программой обеспечивает эффективное и универсальное исполнение различных задач и является основой для большинства современных компьютерных систем. Этот принцип позволяет создавать гибкие и мощные вычислительные устройства, способные адаптироваться к различным потребностям пользователей.

3. Структура ЭВМ согласно принципам фон Неймана

- Принципы фон Неймана – базовая архитектурная концепция ЭВМ.
- Структура машины: *ЗУ*, *АЛУ*, *УУ*, устройство ввода/вывода.
- Программы и *данные вводятся в память* через *АЛУ*.
- Команда включает операцию, адреса данных и место для результата.
- *АЛУ* выполняет операции над данными.
- Результаты *АЛУ* сохраняются в *ЗУ* или устройство вывода.
- Различие между *ЗУ* и устройством вывода: в *ЗУ* данные обрабатываются, на устройства вывода поступают в удобной для человека форме.

4. Архитектура системы команд (АСК). Классификация АСК по составу и сложности команд

- **Система команд ВМ**: перечень команд ВМ.
- **Архитектура системы команд (АСК)**: средства взаимодействия программиста с аппаратурой.
- **Конечная цель ВМ**: эффективные вычисления за минимальное время.
- **Влияние выбора архитектуры**: адреса команд, их длина, доступ к операндам, общая длина команд.
- **Классификация по составу**: *CISC* (Complex Instruction Set Computer), *RISC* (Reduced Instruction Set Computer), *VLIW* (Very Long Instruction Word).
- **Классификация по месту хранения операндов**: стековая, аккумуляторная, регистровая, с выделенным доступом к памяти.

5. АСК. Стековая архитектура

- **Стек** - логически связанные ячейки памяти, действующие по принципу *LIFO*.
- **Структура стека**: вершина, операции **push** и **pop**, запись только в вершину, чтение только из вершины.
- Информация заносится из памяти или *АЛУ* в вершину стека.

- **Арифметические операции**: данные из двух верхних ячеек стека передаются *АЛУ*, результат автоматически возвращается в вершину.
- **Достоинства АСК на базе стека**: сокращение адресной части команд, компактный код, простое декодирование.
- **Недостатки АСК на базе стека**: отсутствие произвольного доступа к памяти, ограничение производительности.
- **Примеры стековых архитектур**: `JVM`, `Forth-процессоры`.

6. АСК. Регистровая архитектура

- **Регистровая архитектура**: принципы работы с массивом регистров (*РОН*).
- **Три формата команд обработки**: *регистр-регистр*, *регистр-память*, *память-память*.
- Операции загрузки и сохранения данных в регистрах подобны операциям с аккумулятором.
- **Три шины между *АЛУ* и *регистровым файлом***: две для передачи операндов, третья для результата.
- **Достоинства регистровой АСК**: компактный код, высокая скорость вычислений за счет работы с регистрами.
- **Недостаток регистровой АСК**: более длинные команды по сравнению с аккумуляторной архитектурой.
- **Преобладающий вид архитектуры** в современных компьютерах.

7. АСК. Аккумуляторная архитектура

- **Аккумуляторная архитектура (АСК)**: операции выполняются между аккумулятором и другими регистрами.
- **Концепт**: операнды хранятся в аккумуляторе, результат туда же сохраняется.
- Команды для загрузки и сохранения данных в аккумуляторе.
- **Достоинства**: короткие команды, простота декодирования.
- **Недостатки**: многократные обращения к основной памяти.
- Менее распространена, но может использоваться в простых системах с ограниченными ресурсами.

8. АСК. Архитектура с выделенным доступом к памяти

- **Архитектура с выделенным доступом к памяти**: механизмы для выполнения операций над данными и доступа к памяти четко определены.
- **Load/Store architecture**: доступ к памяти только через команды *load* и *store*.
- **Операнды** в командах обработки информации могут быть только в регистрах

процессора.
- Результат операции также сохраняется в регистре, прямого обращения к памяти нет.
- **Достоинство**: простота декодирования и исполнения команд.
- Характерна для RISC-архитектуры.

9. АСК. CISC-архитектура

- **Проблема:** Семантический разрыв при переходе к языкам высокого уровня (ЯВУ).
- **Решение:** CISC-архитектура (Complex Instruction Set Computer) - расширение системы команд сложными операторами, аналогичными ЯВУ.
- **Характеристики CISC-архитектуры:**
  - Небольшое число регистров.
  - Многочисленные машинные команды, включая сложные операторы.
  - Разнообразные способы адресации операндов.
  - Множество форматов команд различной разрядности.
  - Команды, совмещающие обработку и обращение к памяти.
- **Негативные аспекты:** Усложнение управления, негативное влияние на производительность.
- **Привлекательность CISC:** Универсальность, удобство программирования, эффективная обработка различных задач.

10. АСК. RISC-архитектура

- **RISC** (Reduced Instruction Set Computing) - архитектура процессора для повышения производительности через упрощенные и часто используемые команды.
- *Основоположник*: Cray Research.
- *Команды работают* с данными только в *регистрах* процессора.
- Используются *специальные команды* для обращения к памяти.
- **Уменьшено количество форматов команд** и способов указания адресов операндов.
- Это **упростило аппаратные средства ВМ** и улучшило быстродействие.
- Пример RISC-процессора: **ARM** в мобильных устройствах.

11. АСК. VLIW-архитектура

- **VLIW** (Very Long Instruction Word) - разновидность RISC-архитектуры, параллельное выполнение команд без динамического выделения.
- Концепция VLIW-архитектуры **базируется на RISC**, где простые RISC-команды объединяются в одну сверхдлинную команду и выполняются параллельно.

- В плане АСК VLIW мало отличается от RISC, с **добавлением уровня параллелизма** вычислений.
- Архитектуру VLIW логичнее ассоциировать с вычислительными системами, чем с вычислительными машинами.

12. Типы и форматы данных. Числа с фиксированной запятой

- **Числа с фиксированной запятой** (Fixed-Point Numbers) - формат данных для представления чисел с постоянным числом разрядов после запятой.
- Используется во встраиваемых системах для предсказуемости и управления объемом памяти.
- В отличие от чисел с плавающей запятой, **числа с фиксированной запятой имеют постоянное количество десятичных разрядов**.
- **Представление числа в форме с фиксированной запятой**:

$$A = \pm a_{n-1} \dots a_1 a_0 a_{-1} a_{-2} \dots a^{-r}$$

- **Отрицательные числа часто представляются в дополнительном коде**.
- **Положение запятой остается неизменным** для всех чисел, что упрощает аппаратную реализацию вычислительных устройств и ускоряет операции.
- Если число смешанное, обрабатывается как масштабируемое целое.
- Эффективен для аппаратной реализации и ускорения машинных операций в вычислительных системах.

13. Типы и форматы данных. Числа с плавающей запятой

- **Числа с плавающей запятой** (Floating-Point Numbers) - формат данных для представления вещественных чисел с десятичной запятой.
- Обеспечивает **более широкий диапазон значений и большую точность** по сравнению с форматом с фиксированной запятой.
- Формула для числа с плавающей запятой: $(-1)^s \times M \times B^E$, где B - основание системы счисления (чаще всего 2).
- Стандарт IEEE 754 предоставляет два формата: одинарной точности (32 бита) и двойной точности (64 бита), обеспечивая разную точность для различных задач.

14. Типы и форматы данных. BCD-числа

- **BCD** (Binary Coded Decimal) - формат представления десятичных чисел в двоичной системе с использованием 4 бит для каждой цифры.

- Каждая тетрада **может принимать значения** от 0000_2 (0_{10}) до 1001_2 (9_{10}), исключая двоичные комбинации без эквивалента в десятичной системе.
- **Преимущества**:
  - Удобно для индикаторов с одной цифрой, например, в часах.
  - Упрощенный вывод на индикацию и ввод с цифровой клавиатуры.
  - При переводе дробных чисел точность не теряется.
  - Упрощены умножение, деление на 10 и округление.
- **Недостатки**:
  - Требует больше памяти.
  - Усложнены арифметические операции.
- **Операции**:
  - При сложении и вычитании чисел формата 8421-BCD применяются коррекционные значения для обработки переносов и недопустимых комбинаций.

15. Типы и форматы данных. Символьная информация, логические данные, строки

- **Символьная информация**:
  - Каждому символу соответствует определенная двоичная комбинация, образующая таблицу кодировки.
  - Принцип весов: веса кодов цифр возрастают, а веса символов увеличиваются в алфавитном порядке.
  - **Кодовые таблицы**:
    1. **EBCDIC** (Extended Binary Coded Decimal Interchange Code).
    2. **ASCII** (American Standard Code for Information Interchange).
    3. **Unicode (UCS)** - 8-разрядные (256 символов) и 16-разрядные (65536 символов).
- **Логическая информация**:
  - Элемент - булева переменная с двумя значениями: "истина" (1) или "ложь" (0).
  - В виртуальных машинах операции с логическими переменными обычно ведутся наборами длиной в машинное слово.
- **Строки**:
  - Непрерывная последовательность битов, байтов, слов или двойных слов.
  - Длина может варьироваться от 0 до 4 Гбайт.
  - Текстовая строка - байты байтовой строки представляют собой коды символов.
- **Прочие виды информации**:
  - **Статическая** - числовая, символьная и логическая, видеоинформация.
  - **Динамическая** - аудиоинформация, видео- и анимационные фильмы.

16. Типы и форматы данных. Видео- и аудиоинформация

- **Аудиоинформация:**
- Частоты аудиосигналов: 15 Гц - 20 кГц, **аналоговые по природе**.
- В виртуальной машине аудиоинформация **оцифровывается с помощью АЦП** и воспроизводится **обратно с использованием ЦАП**.
- Рекомендуется **16-разрядное представление амплитуды** сигнала и частота выборки порядка **40 кГц**.
- **Видеоинформация:**
  - Используется для передачи **движущихся изображений** (анимация).
  - Существуют **два способа представления** графических изображений: матричный (растровый) и векторный.
  - **Матричный (растровый):**
    - Изображение представляется **матрицей пикселов**.
    - Недостаток: большая емкость памяти, требуемая для хранения изображения.
  - **Векторный:**
    - Изображение задается **графическими примитивами, описываемыми математически формулами**.
    - Достоинства: сокращение объема файла и сохранение качества при масштабировании.
    - Недостаток: искусственность изображений и низкая производительность.

17. Сложные структуры данных. Особенности их обработки

В презентациях про это ничего нет. Угадывай мысли Семененко)

18. Функциональная классификация команд

- Команды пересылки данных - Эти команды обеспечивают передачу информации между процессором и оперативной памятью, внутри процессора и между ячейками памяти.
- Арифметико-логические команды - арифмитические и логические операции с байтами;
- SIMD-команды - SIMD-команды предназначены для выполнения одной и той же операции над несколькими элементами данных одновременно.
- Команды для работы со строками - Эти команды обеспечивают перемещение, сравнение и поиск строк. В большинстве ВМ перечисленные операции просто имитируются за счет других команд.

- **Команды преобразования** - преобразование одних типов данных к другому.
- Команды ввода/вывода - команды для обращения с периферией.
- Команды управления потоком команд - изменяют естественный порядок следования и передают управление в иную точку программы. ;
- Команды управления системой - команды являются привилегированными и могут выполняться только когда центральный процессор ВМ находится в привилегированном состоянии или выполняет программу, находящуюся в привилегированной области памяти.

19. Машинные команды. Арифметико-логические команды и команды сдвига

- **Машинные команды:** Основные операции в машинном языке, включая арифметико-логические и команды сдвига.
- **Арифметико-логические команды:** Обеспечивают арифметическую и логическую обработку информации, сопровождаются признаками, такими как Z, N, V, C.
  - **Операции с целыми числами:** Двухместные (сложение, вычитание, умножение, деление), одноместные (абсолютное значение, изменение знака), операции сравнения.
  - **Операции с числами в форме с плавающей запятой:** Арифметические (сложение, вычитание, умножение, деление), операции сравнения, преобразование формы и формата представления.
- **Команды сдвига:** Реализуют логический, арифметический и циклический сдвиг.

20. Машинные команды. Команды пересылки и команды ввода-вывода

- **Команды ввода/вывода:** *обмен информацией* с ПУ.
  - Команды ввода: Получение данных от ПУ и передача их на шину данных.
  - Команды вывода: Прием данных с шины данных и отправка на ПУ.
- **Команды пересылки данных:**
  - Передача информации между процессором и оперативной памятью, внутри процессора и между ячейками памяти.
  - *Содержат* информацию об адресах источника и получателя, длине данных и способе адресации операндов.

21. Машинные команды. SIMD-команды. Команды работы со строками

- **SIMD-команды:**

    - SIMD-команды реализуют **параллельную обработку** двух **групп чисел** с использованием упакованных форматов, ускоряя вычисления над мультимедийными данными.
    - Первые SIMD-команды **выполняют арифметические операции** над упакованными целыми числами с арифметикой с насыщением.
    - Более поздние SIMD-команды обрабатывают также операнды в упакованных числах с плавающей запятой.
- **Команды для работы со строками:**
    - Обеспечивают перемещение, сравнение и поиск строк.
    - Операции, такие как MOVS, LODS и REP, выполняют копирование, загрузку и управление повторениями соответственно.

22. Машинные команды. Команды передачи управления

- **Команды управления потоком команд изменяют естественный порядок выполнения** и передают управление в другую точку программы. Три разновидности команд включают безусловные и условные переходы (ветвления) и вызовы процедур с возвратами.
- **Команды безусловного перехода обеспечивают переход** по заданному адресу без проверки условий.
- **Условные переходы** происходят при соблюдении определенного условия, иначе выполняется следующая команда программы.
- **Команды вызова процедур** и возврата из процедур реализуют процедурный механизм, позволяя переходить к началу процедуры и возвращаться из нее.
- **Команды управления системой** являются привилегированными и выполняются только в привилегированном состоянии процессора или при выполнении программы в привилегированной области памяти. Эти команды управляют системными ресурсами, такими как чтение и изменение состояния регистров устройства управления.

23. Представление команд в ЭВМ. Форматы команд

- **Типовая команда** содержит **операционную и адресную части**, определяющие операцию и адреса операндов и результата.
- **Формат команды** определяет структуру через количество двоичных разрядов, отведенных под всю команду, и расположение полей.
- Выбор формата команды влияет на **характеристики машины**, включая количество команд, общую длину, типы полей, простоту декодирования, адресуемость и стоимость оборудования.

- **Длина команды** важна для организации памяти, структуры шин, сложности и быстродействия ЦП; часто выбирается кратной байту.
- **Общая длина команды** RK определяется количеством адресов, их разрядностью, разрядностью кода операции и способа адресации.

24. Способы адресации. Непосредственная, прямая и косвенная адресации

- В АСК разные способы адресации операндов в ВМ: **исполнительный адрес операнда** (А ИСП) и **адресный код команды** (А К).
- **Способ адресации** - метод формирования исполнительного адреса по адресному коду команды.
- Выбор способов **адресации влияет на** удобство программирования и эффективность, измеряемую затратами оборудования (С) и затратами времени (Т).

Непосредственная адресация:

- Операнд указывается **непосредственно в команде в виде константы**.
- В адресном поле могут быть только константы, размер ограничен длиной адресного поля
- Эффективен в плане затрат на оборудование и времени выполнения.

Прямая адресация:

- Адресный код напрямую указывает **номер ячейки памяти**.
- Ограниченный размер адресного пространства, адрес не изменяется в процессе вычислений.

Косвенная адресация:

- Адрес операнда **указывается в регистре**, операнд находится в **ячейке с адресом**, хранящимся в регистре.
- Используется двукратное обращение к памяти, задействуется лишняя ячейка памяти для хранения адреса операнда.

25. Способы адресации. Базовая, индексная и относительная адресации

- **Адресное поле команды**, указывающее на регистр процессора, обычно имеет короткое размер, позволяя выбрать один из восьми или шестнадцати регистров общего назначения.

- **Базовая регистровая адресация** вычисляет адрес операнда путем сложения базового адреса и смещения; базовый адрес обычно хранится в регистре.
- Базовую регистровую адресацию **часто применяют для доступа к элементам массива**, где базовый регистр хранит начальный адрес массива, а поле адреса смещение относительно начального адреса.
- **Индексная адресация** использует адрес ячейки памяти в поле адреса и регистр в качестве смещения относительно этого адреса.
- **Автоинкрементная и автодекрементная адресация** автоматически изменяют содержимое индексного регистра до или после обращения к нему.
- **Индексная адресация с масштабированием** и смещением умножает содержимое индексного регистра на масштабный коэффициент и суммирует с адресом в команде.
- **Относительная адресация** получает исполнительный адрес операнда сложением содержимого поля адреса команды с содержимым счетчика команд.
- **Программа перемещаема в памяти** при относительной адресации, обеспечивая неизменное взаимное положение команды и операнда в адресном пространстве.

26. Способы адресации. Адресации с автоувеличением, автоуменьшением, страничная адресация

- **Адресация с автоувеличением** использует регистр, содержащий адрес операнда, который **после выборки увеличивается на размер операнда**.
- **Адресация с автоуменьшением уменьшает содержимое регистра на размер операнда** перед выполнением операции, что позволяет эффективно обрабатывать массивы данных и стек.
- **Страничная адресация** разбивает **адресное пространство на страницы**, с базовым адресом страницы в регистре адреса страницы (РАС) и указанием смещения внутри страницы в адресной части команды.

27. Цикл выполнения команды в процессоре

- **Цикл команды** включает этапы: выборка (извлечение) команды, декодирование, вычисление исполнительных адресов, выборка операндов, исполнение операции, запись результата и формирование адреса следующей команды.
- **На этапе выборки команды** команда извлекается из памяти и размещается в регистре команды (РК).
- **Декодирование команды** включает расшифровку содержимого для определения операции и адресов операндов, а также подготовку электронных схем ВМ к выполнению действий.

- **Вычисление исполнительных адресов** определяет адреса операндов и другие параметры для выполнения команды.
- **Выборка операндов** включает обращение к памяти для получения значений операндов.
- **Исполнение операции** осуществляется с использованием выбранных операндов.
- **Запись результата** в память или регистры процессора является частью этапа исполнения.
- **Формирование адреса следующей команды** осуществляется путем увеличения содержимого счетчика команд на длину текущей команды.

28. Процессор. Одношинная архитектура

- **Процессор** - электронный блок или интегральная схема, осуществляющая обработку данных и управление программным процессом.
- **Счетчик команд** (СК, PC) неотъемлемый элемент устройства управления, реализуется как регистр и обеспечивает управление выполнением программы.
- Согласно фон-неймановскому принципу, **адрес следующей команды** получается увеличением адреса ячейки текущей команды на длину команды.
- **Регистр команды (РК, IR)** хранит команду в течение ее выполнения, разделяется на **регистр кода операции (РКОп) и регистр адреса (РА)**.
- **Регистр адреса (MAR)** используется для обращения к памяти, а регистр данных (MDR) хранит данные, предназначенные для записи в память или чтения из нее.
- **Устройства связаны с одной шиной,** что может ограничивать производительность системы из-за последовательного характера обмена информацией между процессором и устройствами.

29. Многошинная архитектура процессора

- **В многошинной архитектуре** обмен по шинам может быть независимым и параллельным во времени, что позволяет оптимизировать структуры шин для каждой задачи.
- Многошинная архитектура **ускоряет работу** микропроцессорной системы при оптимальном выборе параметров шин, хотя т**ребует дополнительных затрат** на аппаратуру и усложняет структуру процессора.
- Преимущества многошинной архитектуры **проявляются лучше внутри одной микросхемы**, особенно в микроконтроллерах, где требуется максимальное быстродействие при заданной тактовой частоте.

30. Принцип микропрограммного управления

- **Устройство управления (УУ)** ВМ выполняет функции управления вычислительным процессом, обеспечивая автоматическое выполнение команд

программы через последовательность машинных циклов.

- **Микрооперации** - элементарные действия в пределах одного такта сигналов синхронизации, объединяются в микрокоманду (МК).
- **Микропрограмма** - последовательность микрокоманд, определяющая содержание и порядок выполнения цикла команды.
- **Сигналы управления**, генерируемые микропрограммным автоматом (МПА), вызывают выполнение одновременных микроопераций.
- Управляющая часть **УУ включает** регистр команды (РК), микропрограммный автомат (МПА) и узел прерывания программ (УПП).
- **Адресная часть УУ включает** операционный узел устройства управления (ОПУУ), счетчик команд (СК) и регистр адреса памяти (РАП).
- **Принцип микропрограммного управления** предполагает использование микропрограммы для управления выполнением инструкций в процессоре, разбивая инструкции на микроинструкции и управляя их выполнением.

31. Структура устройства управления. Микропрограммный автомат (МПА)

Функционирование вычислительной машины обеспечивают системы управления (СУ), формируемые устройством управления (УУ), в данном случае, конкретно - микропрограммным автоматом (МПА).

УУ для выполнения своих функций должно обладать входами, которые позволяют определить состояние управляемой системы, и выходами, через которые осуществляется управление поведением системы.

32. МПА с жесткой логикой

- Выходные сигналы управления в **микропрограммном автомате (МПА)** формируются с использованием заранее соединенных логических схем.
- Фрагмент схемы управления **предполагает порядок выработки** управляющего сигнала Ck в i-м и S-м тактах выполнения команды, зависящий от значений осведомительных сигналов x1 и x3.
- Особенности данной схемы включают э**кономичность и высокое быстродействие** узлов МПА при реализации простой системы команд.
- С увеличением сложности системы команд схемы МПА становятся более сложными, что снижает их быстродействие.
- **Малая регулярность схемы** МПА и сложности при размещении на кристалле интегральной микросхемы становятся проблемой.

33. МПА с программируемой логикой

- **Для инициирования микрооперации** в микропрограммном автомате (МПА) достаточно установить соответствующий сигнал управления на соответствующей линии.
- **Сигналы управления представлены** управляющими словами, называемыми микрокомандами (МК), в МПА с программируемой логикой.
- **Микрокоманда** соответствует одному такту работы вычислительной машины и активирует необходимые сигналы управления в данном такте.
- Последовательность микрокоманд, описывающая выполнение этапа цикла команды, формирует **микропрограмму (МП)**.
- Микропрограммы, разрабатываемые для каждого этапа машинного цикла, **хранятся в управляющей памяти (УПМ) или памяти микропрограмм**.
- **Процесс формирования сигналов управления** включает извлечение из УПМ микрокоманды, интерпретацию ее содержимого как набора сигналов для выполнения конкретной микрооперации.

34. Запоминающие устройства (ЗУ). Характеристики ЗУ

- Запоминающие устройства (ЗУ) **предназначены для хранения и обеспечения доступа к данным** в компьютерных системах.
- **Емкость** определяет количество байт, а единица пересылки зависит от ширины шины данных.
- **Метод доступа** включает последовательный, произвольный, прямой и ассоциативный доступ.
- **Устойчивость к перезаписи** определяет легкость изменения данных.
- **Энергопотребление** важно для энергоэффективности, особенно в портативных устройствах.
- **Быстродействие** включает время доступа, длительность цикла памяти и скорость передачи данных.
- **Стоимость** определяется отношением общей стоимости к емкости в битах.
- **Физические типы** ЗУ включают полупроводниковую память, магнитные диски и оптические диски.
- **Энергозависимость** различается: магнитная и оптическая память энергонезависимы, полупроводниковая может быть как энергозависимой, так и нет.
- Важно учитывать, приводит ли **считывание** информации **к ее разрушению**.

35. Запоминающие устройства (ЗУ). Классификация ЗУ

- Основная память **включает оперативные** (ОЗУ, RAM) и **постоянные** (ПЗУ, ROM) запоминающие устройства.
- **Стековая память** организована по принципу "последним записан - первым считан", используется эффективно в компилирующих и интерпретирующих программах, а также при вычислении арифметических выражений в польской инверсной записи.
- **Ассоциативная память** (АЗУ) способна сравнивать информацию с заданным образцом и указывать на соответствие или несоответствие.
- **Уровни иерархии взаимосвязаны**, данные на одном уровне могут быть найдены на более низком уровне, соотношение стоимость/бит уменьшается, емкость возрастает, время выборки растет, частота обращения к памяти уменьшается по мере движения вниз по структуре.

36. Запоминающие устройства (ЗУ). Основная память

- Основная память может включать ОЗУ (RAM) и ПЗУ (ROM).
- **Необходимость объединения микросхем** памяти возникает из-за различия разрядности ячеек и шины данных.
- **Увеличение разрядности ЗУ** происходит за счет объединения адресных входов, формируя модуль памяти, который может быть одной микросхемой или несколькими, образуя банк памяти.
- **Блочная схема** распределения адресов разбивает адресное пространство на группы, каждая обслуживаемая отдельным банком, с выбором банка по старшим разрядам адреса.
- **Циклическая схема** чередует адреса между банками, используя младшие разряды адреса для выбора банка и старшие для выбора ячейки внутри банка.

37. Запоминающие устройства (ЗУ). Стековая память

- **Стековая память** - специальный тип запоминающего устройства, организованного по принципу стека с LIFO (Last In, First Out) порядком.
- **Стековая архитектура** представляет собой безадресную память с последовательным доступом.
- В стековом ЗУ ячейки формируют одномерный массив, где слова связаны разрядными цепями и доступны в определенном порядке.
- Слова привязаны к своему относительному **положению относительно других слов**.
- **Слова могут перемещаться, сохраняя упорядоченность**, и считываются в тот момент, когда оказываются в нужной ячейке.

- Стековые ЗУ делятся на два типа: FIFO (First In, First Out) и LIFO (Last In, First Out).

38. Запоминающие устройства (ЗУ). Ассоциативная память

- АЗУ способна **хранить и сравнивать информацию с заданным образцом**, используя ассоциативный признак и признак поиска в виде кодовой комбинации.
- Ассоциативное запоминающее устройство **включает** массив для хранения N m-разрядных слов, регистр ассоциативного признака, схемы совпадения, регистр совпадений, регистр маски и комбинационную схему.
- **Регистр ассоциативного признака** содержит код искомой информации, а его разрядность k меньше длины слова m.
- Схемы совпадения параллельно сравнивают каждый бит всех хранимых слов с соответствующим битом признака поиска.
- **Регистр совпадений** заполняется единицами, если все разряды соответствующей ячейки совпали с признаком поиска.
- **Архитектура** ассоциативного ЗУ **определяется** видом поиска, техникой сравнения признаков, способом считывания и записи информации.

39. Кэш-память. Структурная организация

Когда ЦП пытается прочитать слово из основной памяти, **сначала осуществляется поиск** копии этого слова **в кэше**. Если такая копия существует, **обращение к ОП не производится**, а в ЦП передается слово, извлеченное из кэш-памяти.

Между ОП и процессором размещается небольшая, но **быстродействующая буферная память**, куда в процессе работы копируются те участки ОП, к которым производится обращение со стороны процессора.

Обычно в компьютере имеется **два уровня кэш-памяти**.

- Первичный кэш располагается на микросхеме процессора и называется кэшем первого уровня (**L1**).
- Вторичный кэш имеет больший объем, располагается между первичным кэшем и остальной памятью и называется кэшем второго уровня (**L2**).
- **Попадание в кэш (hit)** – слово, запрашиваемое процессором, есть в кэше.
- **Промах чтения (miss)** - слово, адресуемое операцией считывания, отсутствует в кэше.

40. Способы отображения оперативной памяти на кэш-память

- **Отображение блока** основной памяти на кэш-память включает копирование блока в строку кэш-памяти, и все обращения к блоку в ОП переадресовываются на соответствующую строку кэш-памяти.
- При прямом отображении **адрес** строки i кэш-памяти определяется **выражением** i = j mod m, где m - общее число строк в кэш-памяти, и каждый 128-й блок ОП отображается на строку с номером i.
- **Полностью ассоциативное отображение** позволяет загружать любой блок ОП в любую строку кэш-памяти, выделяя в адресе ОП тег и поле слова.
- **Множественно-ассоциативное отображение** разбивает кэш-память на v подмножеств, каждое из которых содержит k строк, являясь компромиссом между прямым и полностью ассоциативным отображением.

41. Виртуальная память. Варианты ее организации

- **Виртуализация памяти** - механизм для использования программами большего объема памяти, чем физически доступно в оперативной памяти, реализуется на уровне операционной системы и включает различные методы организации.
- Физическая память и виртуальная память **разбиваются на страницы**, где физический адрес определяется парой (Pp, i), а виртуальный адрес - (Pv, i).
- **Для отображения виртуального** адресного пространства на физическую память каждой задачи используются таблицы страниц, где дескрипторы хранят информацию о каждой странице, а преобразователь адресов транслирует номер виртуальной страницы в номер физической страницы.
- Виртуальная память программы **делится на сегменты с независимой** адресацией байтов, и к виртуальному адресу добавляются разряды для определения сегмента, что создает иерархию организации программ: программа-сегмент-страница-байт.
- **Иерархия таблиц**, соответствующая этой иерархии программ, используется для перевода виртуальных адресов в физические.

42. Конвейерная обработка. Основы конвейеризации

Конвейерная обработка — это метод организации процесса вычислений, при котором задача разбивается на несколько последовательных этапов, и каждый этап выполняется независимо от предыдущего. Процессор конвейера разделен на несколько блоков, и каждый из них выполняет свою специфическую функцию. Основные принципы конвейерной обработки включают в себя:

1. **Разделение задачи:**

- Задача разбивается на несколько подзадач, которые могут быть выполнены последовательно.

2. **Последовательность этапов (ступеней):**
   - Задача разбивается на несколько последовательных этапов, называемых ступенями. Каждая ступень выполняет определенный тип операции.
3. **Одновременное выполнение:**
   - Несколько задач выполняются одновременно на различных ступенях конвейера. Каждая новая задача начинает выполнение на следующей доступной ступени, когда предыдущая завершает свое выполнение.
4. **Параллелизм:**
   - Конвейерная обработка предоставляет параллелизм на уровне выполнения инструкций. Разные задачи могут находиться на различных ступенях конвейера одновременно.
5. **Увеличение производительности:**
   - Основная цель конвейерной обработки - увеличение общей производительности путем уменьшения времени выполнения каждой задачи и повышения загрузки процессора.

Конвейерная обработка команд не приводит к ускорению выполнения каждой отдельной команды. Скорость команд остается прежней, но на единицу времени приходится большее их количество.

43. Конфликты в конвейере команд. Риск по данным

- Конфликты в конвейерной обработке команд, такие как конфликт по данным (Data Hazard), могут замедлить выполнение программы из-за ожидания доступа к данным.
- Зависимость по данным включает в себя конфликты по чтению и записи, где вторая инструкция зависит от результатов выполнения первой.
- Конфликт по чтению возникает, когда инструкция пытается считать данные, которые еще не были записаны предыдущей инструкцией.
- Конфликт по записи происходит, когда инструкция пытается записать данные, которые еще не были использованы предыдущей инструкцией.
- Техники управления конфликтами включают переупорядочение инструкций, продвижение данных, промежуточные регистры и инструкции с отсроченным выполнением.

44. Конфликты в конвейере команд. Риск по управлению

- Риск по управлению (Control Hazard) в конвейерной обработке команд возникает из-за задержек, таких как промах при выборке команды из кэша или выполнение команды перехода.
- Организация очереди команд и упреждающая выборка являются решением для управления рисками по управлению в конвейере.
- Риски по управлению включают ожидание ветвления, ожидание прыжка и неопределенность ветвления.
- Использование предсказателей ветвления, техник предсказания прыжков и оптимизация конвейерной архитектуры помогают справиться с рисками по управлению и обеспечить эффективное выполнение программ.

45. Структурные конфликты в конвейере команд

- Структурные конфликты в конвейерной обработке команд возникают, когда двум инструкциям требуется одновременный доступ к аппаратному ресурсу, чаще всего при обращении к памяти.
- Решение проблемы структурных конфликтов включает в себя обеспечение достаточного количества аппаратных ресурсов, таких как раздельные кэши команд и данных.
- В конвейерной обработке скорость выполнения каждой команды остается прежней, но увеличивается количество команд, выполняемых за единицу времени.
- Структурные конфликты могут привести к приостановке работы всего конвейера, если возникают заторы на какой-либо из его ступеней.
- Управление структурными конфликтами включает в себя использование предсказателей структурных конфликтов, увеличение ресурсов, переупорядочение инструкций и применение многозадачности.