1. Что такое компьютерная система? Отличие информационной и управляющей системы? Почему большинство современных компьютерных систем считаются системами с преобладающей программной составляющей? Примеры. Понятие информационного процессора.

什么是计算机系统？ 信息系统和控制系统之间的区别？ 为什么大多数现代计算机系统被认为是软件主导的系统？ 例子。 信息处理器的概念。

·计算机系统是一种能够按照事先存储的程序（按照人们的要求）自动、高速地对数据进行输入、处理、输出和存储的机器系统。

计算机系统主要由硬件系统和软件系统组成

硬件系统包括中央处理器、内存、输入/输出设备和存储设备等

软件系统由程序和数据组成的指令集合，用于指导计算机执行特定任务。软件部分可以分为系统软件和应用软件两类。系统软件负责管理计算机的硬件资源；应用软件是直接面向用户的软件，能够完成特定的任务或提供特定的服务

·信息系统是用来收集、存储、处理、传输和管理数据和信息的系统。（性能是重要指标，可以预测用户行为，同时执行多个任务）

控制系统是用来监测、调节和控制物理或工程系统的系统，以实现所需的输出或行为。（自动化操作，实时监控和调节）

·绝大多数现代 系统具有主要的软件组件，因为几乎所有主要功能都是在软件级别实现的

·信息处理器是指专门用于处理和操作数据、信息或信号的设备或组件。旨在执行各种信息处理任务，包括数据存储、检索、转换、传输、分析等。

---------------------------------------------------------------------------------------------------------------------------------

1. Системная инженерия. Понятие системы. Варианты рассмотрения систем. Модульность. Жизненный цикл. Операционное окружение и обеспечивающие системы. Заинтересованные стороны (stakeholders). Проблема передачи информации при разработке компьютерных систем.

系统工程。 系统的概念。 考虑系统的选项。 模块化。 生命周期。 运行环境和支撑系统。 利益相关方（利益相关者）。 计算机系统开发中的信息传输问题。

·系统工程是为了最好地实现系统的目的，它侧重于全面、同步地了解用户的需求； 探索可能性和记录要求； 在考虑整个问题（从系统概念探索到系统退役）时综合、验证、接受和开发解决方案，使系统的整体与局部之间的关系协调和相互配合，实现总体的最优运行。

·系统是为了实现一个或多个既定目的而组织的相互作用元素的组合。系统是由相互作用相互依赖的若干组成部分结合而成的，具有特定功能的有机整体，而且这个有机整体又是它从属的更大系统的组成部分

·系统身份：系统的身份由其功能位置决定，操作环境：系统部署的环境，系统开发所针对的问题或机会存在于该环境中。操作环境是定义所需系统功能、期望的利益相关者结果和收益以及约束的重要因素。

·模块化： 模块化是系统工程中常用的方法，它将系统分解为较小的、可管理的模块或组件，以简化系统设计、实现和维护。

·Conceptual stage概念阶段

Development stage发展阶段

Production stage生产阶段

Disposal stage处置阶段

Support stage支持阶段

Decommissioning phase退役阶段

·操作环境：系统部署的环境，系统开发所针对的问题或机会存在于该环境中。操作环境是定义所需系统功能、期望的利益相关者结果和收益以及约束的重要因素。

支持系统：是在其生命周期阶段补充感兴趣的系统的系统，但不一定直接有助于其在运行期间的功能。

·利益相关者： 利益相关者是对系统设计、开发和运行结果有利害关系的个人或组织，包括客户、用户、开发人员、管理人员等。

·信息传输问题涉及到在系统内部或系统与外部环境之间传输数据、信息或信号时可能出现的一系列挑战和考虑因素。这些问题可能涉及到数据传输的速度、可靠性、安全性、实时性等方面。

---------------------------------------------------------------------------------------------------------------------------------

1. Цели и задачи архитектурного проектирования компьютерных систем, его эффект. Понятие архитектуры. Различные трактовки (Гради Буч, ISO 42010 и д.р.) и их практическая значимость.

计算机系统架构设计的目标和目标及其效果。 建筑学的概念。 各种解释（Гради Буч、ISO 42010 等）及其实际意义。

·计算机系统架构设计的目标和任务是确保系统能够满足其预期的功能、性能、可靠性、可扩展性和安全性要求。通过良好的架构设计，可以实现功能性，性能，可靠性，可扩展性，安全性

·格拉迪·布查将架构描述为系统中元素的分布、它们之间的关系以及系统与环境之间的交互。这种理解强调了对系统整体结构和组成部分之间关系的把握。

布查的解释帮助开发人员更好地理解系统的结构和设计，从而更好地规划和管理系统的开发过程。

·ISO 42010标准将架构定义为系统的基本组织结构、在该结构下的组件、它们之间的关系，以及与环境的接口。ISO 42010着重于对系统架构文档的标准化和形式化。

ISO 42010标准的应用使得系统架构文档更具可读性和可理解性，促进了团队之间的协作和交流。

---------------------------------------------------------------------------------------------------------------------------------

1. Реле как базис компьютерной системы. Область применения и принципы построения систем на базе реле. Примеры релейных схем и принцип их функционирования. Формирование логического базиса.

继电器作为计算机系统的基础。 构建基于中继的系统的范围和原则。 继电器电路示例及其工作原理。 形成逻辑基础。

·继电器是一种电子元件，可用作计算机系统的基础。它们通过控制电流的流动来实现逻辑和计算操作。

继电器广泛用于自动控制系统、电力系统、电信系统以及过去的早期计算机系统中。在计算机系统中，继电器被用于构建逻辑电路和执行基本计算操作。

·基于继电器构建系统时，需要考虑以下原则：逻辑设计，组合逻辑，时序逻辑，信号传输

·逻辑门：使用继电器可以构建基本的逻辑门，如与门、或门和非门。例如，与门由两个继电器组成，当且仅当两个输入同时为高电平时，输出为高电平。

加法器：通过连接多个逻辑门，可以构建加法器电路，实现二进制数的加法运算。

存储器：使用触发器和时钟信号，可以构建存储器单元，实现数据的存储和读取功能。

·通过适当设计和连接继电器，可以构建出完整的逻辑基础，实现逻辑运算、计算和存储功能。这为构建基本的计算机系统提供了基础。

与门：由多个继电器串联连接，当且仅当所有输入信号同时为高电平时，输出信号才为高电平。

或门：由多个继电器并联连接，当任何一个输入信号为高电平时，输出信号就为高电平。

非门：由一个继电器控制的开关来实现，当输入信号为高电平时，开关闭合，输出低电平；当输入信号为低电平时，开关断开，输出高电平。

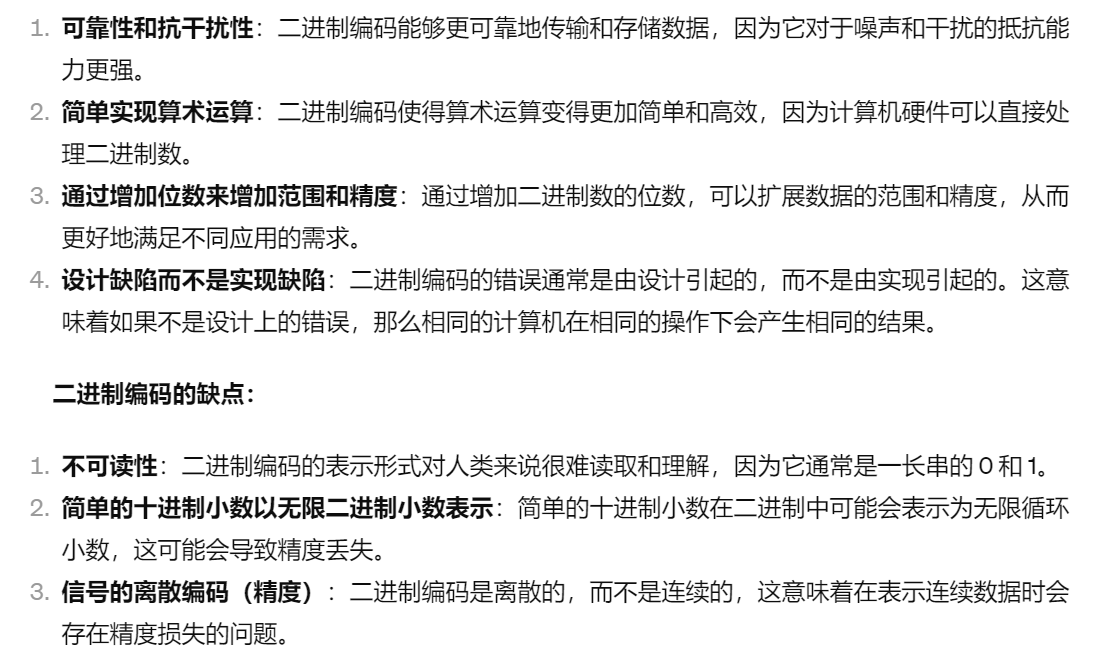
---------------------------------------------------------------------------------------------------------------------------------

1. Принципы кодирования информации в компьютерных системах. Двоичный код, достоинства и недостатки. Машинное слово и адресация. Код грея, BCD, Base64.

计算机系统中的信息编码原理。 二进制代码，优点和缺点。 机器字和寻址。 格雷码、BCD、Base64。

·计算机系统中的信息编码原理是指将各种形式的数据（如文本、图像、音频等）转换为计算机可以处理的数字形式

·二进制代码是计算机系统中最基本的编码方式，它使用 0 和 1 来表示数字、字符和指令。



·机器字： 在计算机科学中，机器字是指计算机中一次能够处理的二进制数据单元大小。

·地址寻址： 地址寻址是指计算机系统中访问内存中特定位置的过程。每个内存单元都有一个唯一的地址标识符，通过这个地址标识符，计算机可以定位并访问内存中的数据。地址寻址是计算机中重要的操作之一，它为程序提供了对数据和指令的访问方式。

·格雷码（Gray Code）是一种特殊的二进制编码方式，其中相邻的两个数只有一个位的变化。0000、0001、0011

·BCD（Binary Coded Decimal）是一种将十进制数表示为二进制数的编码方式。每个十进制数字用四位二进制数表示。529 的BCD编码为 0101 0010 1001

·Base64 是一种将二进制数据编码为可打印字符的编码方式，Base64 编码可以将二进制数据转换为由 A-Z、a-z、0-9 和一些特殊字符组成的字符串，以便在文本协议中传输和存储

---------------------------------------------------------------------------------------------------------------------------------

1. Что такое комбинационная схема? Состояние и параллелизм в комбинационных схемах и схемах с регистрами, переходные процессы. Чтение и запись значений в регистры. Состояние x и z. Параллелизм уровня битов. Особенности поведения комбинационных схем по сравнению с программами.

什么是组合电路？ 组合电路和寄存器电路中的状态和并行性、瞬态过程。 向寄存器读取和写入值。 状态 x 和 z。 位级并行。 组合电路与程序相比的行为特征。

组合电路是由一组逻辑元件组成的电路，这些逻辑元件一起实现给定的真值表。

组合电路是一种数字电路，它们的输出仅取决于当前输入的状态，而不受过去输入状态的影响。

状态和并行性：

组合电路没有内部状态或存储元件，因此它们不会保持任何状态。

组合电路中的操作是并行执行的，每个输入变化都立即影响输出，不存在时序延迟。

过渡过程：

过渡过程指的是电路在输入发生变化时，输出从一个状态过渡到另一个状态的过程。

对于组合电路，过渡过程是瞬时的，没有延迟，输出立即响应输入的变化。

读写寄存器：

寄存器电路包含存储元件，可以存储数据并在需要时读取。

写操作将数据写入寄存器，而读操作从寄存器中读取数据。

状态 x 和 z：

在数字电路中，x 表示未知状态或高阻态，z 表示高阻态或未连接状态。

x 和 z 状态可能会导致电路输出产生不确定的结果或异常行为。

位级并行性：

位级并行性是指电路在每个时钟周期内同时处理多个数据位。

组合电路和寄存器电路都可以实现位级并行性，但是它们的实现方式有所不同。

与程序的比较：

组合电路与程序不同，程序是顺序执行的，而组合电路是并行执行的。

程序可以实现复杂的逻辑和控制流，而组合电路主要用于执行固定的逻辑功能。

---------------------------------------------------------------------------------------------------------------------------------

1. Аппаратное обеспечение. Виды монтажа и особенности производства. Навесной и поверхностный монтаж. Интегральные схемы. Проблемы, специфичные для аппаратного обеспечения: производство, эксплуатация, устаревание.

硬件。 安装类型和生产特点。 壁挂式和表面安装。 集成电路。 硬件特定问题：制造、操作、过时。

·硬件设备是计算机系统的物理组成部分，包括各种电子元件和机械部件。它们负责处理、存储和传输数据，支持计算机系统的功能。

·壁挂式安装： 一种安装电子电路的方法，其中位于绝缘底盘上的无线电元件通过电线或直接通过引线相互连接。优点：易于生产、易于生产准备、灵活性。缺点：成本高，密度受限，可能降低可靠性

PCB安装：是由电介质制成的板，在其表面（SMT）和/或体积内（PTH）形成电子电路的导电电路。 PCB 的主要功能是提供机械支撑和电气连接，使得电子组件能够安全地安装在其表面上，并且相互之间能够正确连接以形成完整的电路。

表面贴装技术（Surface-mount technology，SMT）是一种电子组装技术，用于在印刷电路板（PCB）的表面上安装电子元件。与传统的插件安装技术相比，SMT 可以实现更高的组件密度、更快的生产速度和更好的电气性能。

·集成电路（Integrated Circuit）也称为芯片，是一种在单个半导体晶片上集成了大量电子元件（如晶体管、电阻、电容等）的电路。它将传统电路中的多个功能组件集成到一个微小的芯片上，实现了高度集成化和微型化。

·生产问题：集成电路的生产过程需要高度精密的制造设备和工艺，这意味着制造成本较高。此外，生产过程中的一些变量，如工艺参数的波动、设备故障等，可能会导致制造中的不良率增加，从而影响生产效率和产品质量。

·使用问题：过热、电气故障、外部干扰等。正确的安装、散热和维护对于确保集成电路的稳定运行至关重要。此外，某些环境条件，如高温、高湿度等，可能会对集成电路的性能和寿命产生负面影响。

·过时问题：随着技术的发展和市场需求的变化，集成电路的技术和规格可能会迅速过时。为了应对这一问题，需要定期进行技术更新和产品升级，这会增加额外的成本和管理复杂度。

---------------------------------------------------------------------------------------------------------------------------------

1. 2-этапное производство, варианты обеспечения. Понятия Hardware и Software, их свойства. Сравнение с понятиями программного и аппаратного обеспечения. Особенности. Причины разделения. Принципы совместного проектирование (HW/SW CoDesign).

2 阶段生产、供应选项。 硬件和软件的概念及其属性。 软件和硬件概念的比较。 特点。 分离的原因。 协同设计原则（HW/SW CoDesign）。

·2-阶段生产 是制造过程分为两个主要阶段的生产方法。第一阶段是制造和组装组件，第二阶段是组装和制造最终产品。这种方法允许制造商在两个阶段进行测试和调整，以确保最终产品的质量和性能。

·硬件是指电子设备中的物理部分，如电路板、处理器、存储器、传感器等。硬件通常由金属、塑料和其他材料制成，直接执行计算和控制任务。

软件是指电子设备中的程序和代码，用于控制硬件执行特定的任务和功能。软件通常由计算机语言编写，然后在硬件上执行，实现各种功能，如操作系统、应用程序、驱动程序等。

·硬件是物理实体，可以直接观察和测量，而软件是代码和程序，需要计算机来执行和运行。

硬件通常比软件更昂贵和更难更改，因为它们涉及物理部件的制造和组装。

软件通常比硬件更易于更新和调整，因为它们可以通过编程和代码修改来改变其功能和行为。

·硬件/软件协同设计（HW/SW CoDesign）是一种设计方法，旨在优化硬件和软件之间的协同作用，实现最佳的系统性能和功能。该方法涉及硬件和软件工程师之间的紧密合作，以确保两者之间的良好匹配和协调。它通常包括以下几个步骤：

硬件和软件需求分析

硬件和软件设计和开发

硬件和软件集成和测试

系统优化和调整

---------------------------------------------------------------------------------------------------------------------------------

1. Понятие модели вычислений (MoC). Сопоставление понятия парадигмы программирования и MoC. Примеры MoC (последовательные, параллельные, функциональные) и их роль в разработке компьютерных систем. Использование MoC в разных вычислительных платформах.

计算模型（MoC）的概念。 编程范式和MoC概念的比较。 MoC（串行、并行、功能）示例及其在计算机系统开发中的作用。 在不同的计算平台上使用MoC。

Model of Computation

·Moc用于描述计算机系统中数据和控制流的行为方式。它们提供了一种形式化的框架，以帮助理解和分析计算机系统的行为，从而指导系统设计和优化。

计算机系统的过程可以用模型/规范/程序来描述。

MoC提供了一种描述过程模型的语言。

MoC定义了计算机器的能力。

MoC描述了决定模型和计算机行为的执行规则（不一定是确定性的）。

MoC 相对于真实计算机来说是简约的。

·与编程范式相比，计算模型更侧重于描述计算机系统的行为方式和结构，而编程范式更关注于编写代码和解决问题的方法。

·顺序模型Sequential models：顺序模型描述了计算机系统中的数据和控制流按照顺序执行的方式。

Finite state machines

Pushdown automata

Random access machines

Turing machines

von Neumann Machine

函数模型Functional models：函数式模型描述了计算机系统中的计算过程可以看作是函数之间的组合和应用

Lambda calculus

General recursive functions

Combinatory logic

Abstract rewriting systems.

并发模型Concurrent models：并行模型描述了计算机系统中的数据和控制流可以同时执行的方式

Cellular automaton

Kahn process networks

Petri nets

Synchronous Data Flow

Interaction nets

Actor model.

·Computer Science. Формальные модели.

计算机科学。 正式模型。

Дизайн языков программирования.

编程语言的设计。

Управление сложностью.

管理复杂性。

Ограничение творческого порыва.

创作冲动的限制。

Модель-ориентированная инженерия (xtUML, Capella, Switch-технология и т.п.):

基于模型的工程（xtUML、Capella、Switch技术等）：

·для ответственных применений;

对于关键应用；

·для переносимости;

便于携带；

·для экспертизы. 用于检查。

---------------------------------------------------------------------------------------------------------------------------------

1. Универсальный процессор и его свойства. Машина Тьюринга и полнота по Тьюрингу. Виды процессоров (СБИС, FPGA, CGRA, GPU, DSP, CPU) и их сопоставление с точки зрения универсальности и эффективности.

通用处理器及其属性。 图灵机和图灵完备性。 处理器类型（VLSI、FPGA、CGRA、GPU、DSP、CPU）及其在多功能性和效率方面的比较。

·通用处理器是一种能够执行各种类型指令并处理不同任务的处理器

属性：

1从生产中分离出“编程”阶段：通用处理器的设计和生产是分离的过程。在生产阶段，制造商生产处理器的物理硬件。而在编程阶段，用户可以通过加载不同的软件或编写程序来配置处理器的行为和功能。

2图灵完备性：通用处理器具有图灵完备性，这意味着它们可以模拟任何图灵机可以完成的计算任务。

3没有“严格”的程序大小限制：用户可以编写和加载各种大小的程序，以适应不同的应用需求。通常，处理器的存储容量限制了程序的大小，但在现代处理器中，这种限制通常较大，可以容纳大型程序。

4可修改的软件：这意味着用户可以通过更新软件或加载新的程序来改变处理器的功能和行为，从而满足不断变化的需求和应用场景。

·图灵机是一种抽象的数学模型，用于描述计算和计算机。图灵机由一个无限长的纸带和一个可以在纸带上移动、读写符号的读写头组成。

1无限长的纸带：纸带被分成了一个个的单元格，每个单元格可以写入一个符号，包括空白符号。

2读写头：读写头可以在纸带上移动，并且可以读取当前所在单元格的符号，并根据事先定义的规则进行操作，比如写入新的符号、向左或向右移动等。

3程序

图灵完备性是指一个系统或模型具有模拟图灵机的能力，也就是说，它能够模拟图灵机能够执行的任何计算任务。具有图灵完备性的系统被认为是通用的计算模型，因为它们可以模拟任何其他计算设备的行为。

·СБИС（片上系统）

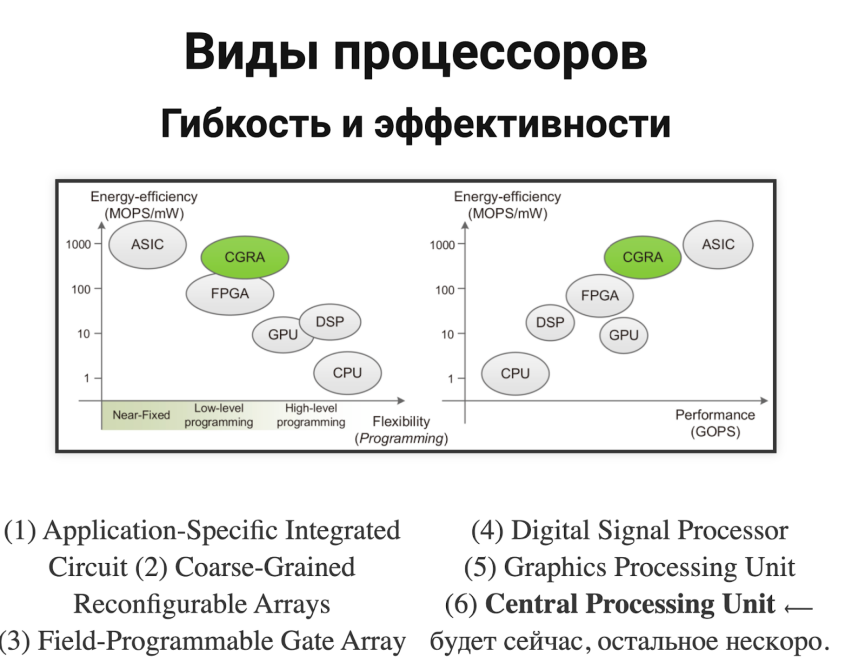
FPGA（可编程逻辑阵列）

CGRA（图形可重构架构）

GPU（图形处理器）

DSP（数字信号处理器）

CPU（中央处理器）



---------------------------------------------------------------------------------------------------------------------------------

1. Архитектура фон Неймана. Принципы. Свойства. Особенности и ограничения. Применение на практике. Машинное слово. Понятие системы команд и её роль в построении процессоров. Control Unit и DataPath.

冯·诺依曼架构。 原则。 特性。 特点和限制。 实际应用。 机字。 指令系统的概念及其在处理器构造中的作用。 控制单元和数据路径。

·冯·诺依曼体系结构是一种计算机系统设计的基本原则

存储介质从无限长的纸带改为了随机存取存储器

指令和数据相结合

计得在实施和生产中尽可能简单

·特点：采用二进制编码

软件控制，命令按照顺序进行，一个接着一个。

计算机存储器用于统一存储数据和程序。

地址寻址：存储器中的每个单元都有唯一的地址，用于定位数据和指令。

条件转移：程序可以根据条件执行不同的指令序列，实现条件转移。

·优点：简单、灵活、通用、易于实现和理解。

缺点：由于采用顺序执行，效率较低；单一总线结构可能导致瓶颈；存储器访问速度限制了整体性能。

·应用实践：冯·诺依曼体系结构被广泛应用于现代计算机，包括个人电脑、服务器、移动设备等。

·机器字：机器字是计算机能够一次处理的二进制位的长度，通常与处理器的数据总线宽度相关。机器字是处理器操作的最小数据单元。 机器字大小决定了处理器在一次操作中可以处理的最大信息量。

·指令系统是一种抽象处理器模型，形成软件和处理器之间的接口

指令系统定义了计算机可以执行的所有指令集合，包括算术运算、逻辑运算、数据传输等。

·Control Unit和DataPath：

Control Unit负责解码和执行指令，控制数据流和控制信号的生成；is a component of a computer's central processing unit (CPU) that directs the operation of the processor. A CU typically uses a binary decoder to convert coded instructions into timing and control signals that direct the operation of the other units.

DataPath是指执行指令时数据传输的路径，包括运算器和寄存器等组件。这两者共同组成了CPU的核心部分。is the ALU, the set of registers, and the CPU's internal bus(es) that allow data to flow between them.

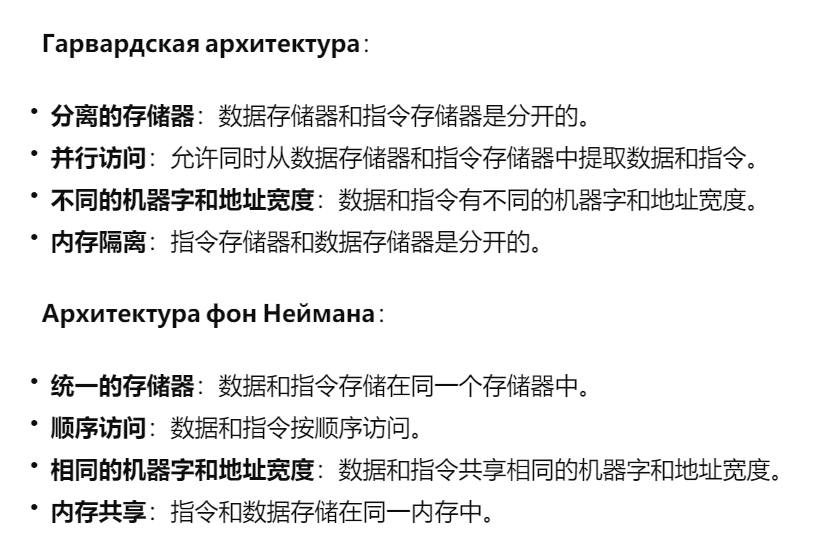
---------------------------------------------------------------------------------------------------------------------------------

1. Гарвардская архитектура и её отличия от архитектуры фон Неймана. Принципы. Свойства. Особенности и ограничения. Применение на практике. Подходы к обходу ограничений архитектуры. Машинное слово. Control Unit и DataPath.

哈佛架构及其与冯诺依曼架构的差异。 原则。 特性。 特点和限制。 实际应用。 绕过架构限制的方法。 机字。 控制单元和数据路径。

·哈佛架构数据储存期和指令储存器是独立分开的，允许同时从数据存储器和指令存储器中提取数据和指令。指令和数据通道在物理上是分开的

冯诺依曼是数据和指令共享同一储存器，指令和数据按顺序提取，可能导致性能瓶颈。



·优点：

同时内存访问： 允许同时从内存中获取数据和指令，提高了性能。

数据和程序的机器字宽和地址不同： 这使得每种类型的存储器都可以被优化，以满足不同的需求。

优化手头的任务： 可以针对特定任务进行优化，更有效地利用资源。

数据和程序存储器总是混合的： 这种混合存储方式提供了更灵活和高效的内存使用方式。

·缺点：

实施的复杂性和成本： 由于需要单独存储数据和指令，因此实施哈佛架构可能更加复杂和昂贵。

指令和数据的隔离： 数据和指令分开存储，可能导致处理数据和指令的复杂性，需要额外的机制来确保它们的有效交互。

·冯诺依曼的限制；指令访问限制（处理器必须在执行指令时等待内存访问完成），数据访问限制，非同时访问限制（无法同时访问指令和数据）

·哈佛架构通常用于对性能要求较高的嵌入式系统和数字信号处理

冯诺依曼更常见于通用计算机系统，例如个人电脑和服务器。

·使用高速缓存来减少内存访问的时间。

使用流水线技术来提高指令执行的效率。

使用多核处理器来提高并行处理能力。

·机器字：机器字是计算机能够一次处理的二进制位的长度，通常与处理器的数据总线宽度相关。机器字是处理器操作的最小数据单元。 机器字大小决定了处理器在一次操作中可以处理的最大信息量。

·Control Unit和DataPath：

Control Unit负责解码和执行指令，控制数据流和控制信号的生成；is a component of a computer's central processing unit (CPU) that directs the operation of the processor. A CU typically uses a binary decoder to convert coded instructions into timing and control signals that direct the operation of the other units.

DataPath是指执行指令时数据传输的路径，包括运算器和寄存器等组件。这两者共同组成了CPU的核心部分。is the ALU, the set of registers, and the CPU's internal bus(es) that allow data to flow between them.

---------------------------------------------------------------------------------------------------------------------------------

1. Механизм микроопераций, микропрограммирование и его роль в развитии компьютерных систем. Особенности и ограничения. Применение на практике. Пример оптимизации через микрокод. NISC архитектура.

微操作、微编程的机制及其在计算机系统开发中的作用。 特点和限制。 实际应用。 通过微代码优化的示例。 NISC 架构。

·微操作引擎是一种通过将每条指令分解为称为微操作（更小的操作）来管理处理器中的操作的方法。

·微编程是一种使用微程序来控制微操作执行的方式，微程序是执行每条指令的微操作序列。

·在计算机系统开发中的作用：

微编程允许您创建更灵活和模块化的处理器，因为更改固件允许您在不更改硬件的情况下更改处理器的行为。这使得开发新处理器和升级现有处理器变得更加容易。

·优点：易于实施（CISC）

对指令系统进行“编程”的可能性。

程序员可以直接访问微代码。

任务的ISA生成（减少体积，提高效率）微程序控制可以用于生成指令集体系结构（ISA）

·缺点

微码存储在处理器内部。

CISC需要专业知识。

架构的多样性=工具问题。

各种命令（格式、大小、持续时间、访问）。 复杂化：

处理器优化；

工具。

微代码介绍了编程的所有问题（复杂性、调试、技术）。

·实际应用：微码优化：通过微码优化，可以对处理器进行性能调整和改进，以适应不同的应用场景。

应对新需求：微程序设计使得处理器可以快速适应新的需求和技术变化。

·微代码优化是通过更改或添加微代码指令来优化处理器性能的过程。 微代码是一组用于控制处理器内部操作的指令。

微代码优化的示例可能包括添加新的微代码指令以提高性能或纠正处理器中的错误。 例如，如果发现特定指令序列执行效率低下，开发人员可以创建新的微代码指令来更有效地执行该序列。

通过微代码优化的另一个例子是修复错误，而无需发布新的处理器。 如果在特定指令中发现错误，开发人员可以创建新的微代码来修复该错误并将其作为微代码更新进行分发。

因此，通过微代码的优化允许开发人员提高性能并修复处理器中的错误，而无需发布新的处理器物理版本。

·NISC优点：

设备简化

充分利用处理器

没有ISA，没有设计问题

·缺点：

无法实现二进制的兼容性

机器代码密度低

·应用：

适用于加速器和高级综合（HLS）应用：NISC架构通常用于加速器和高级综合（HLS）应用，能够有效地处理复杂的数据并行任务。

并行NISC处理器：NITTA是一种并行NISC处理器，它将NISC架构与并行处理相结合，能够在处理复杂的并行任务时提供更高的性能和效率。

---------------------------------------------------------------------------------------------------------------------------------

1. Что такое CISC? Роль в развитии компьютерных систем. Применение на практике. Достоинства и недостатки. Отличия от архитектуры фон Неймана. Особенности программирования.

什么是 CISC？ 在计算机系统开发中的作用。 实际应用。 的优点和缺点。 与冯诺依曼架构的差异。 编程的特点。

·CISC : is a computer architecture in which single instructions can execute several low-level operations (a load from memory, an arithmetic operation, and a memory store) or are capable of multi-step operations or addressing modes within single instructions.

是一种计算机体系结构，其中单个指令可以执行多个低级操作（从内存加载、算术运算和内存存储），或者能够在单个指令内执行多步操作或寻址模式。

·出现的原因：

1早期的计算机编程主要使用低级语言，如汇编语言，这些语言对硬件的细节要求较高，因此设计一种指令集能够直接映射到这些语言，可以更方便地编写程序。

2早期的编译器技术相对较为简单，难以实现将高级语言直接转换为简单的指令。因此，设计一种能够容纳更多复杂功能的指令集有助于降低编译器的复杂度。

·优点：CISC架构能够适应不同的硬件设计，并提供统一的编程接口

CISC架构通常提供了更多的指令和操作符，使得编程更加灵活和方便

CISC架构通过在单个指令中执行多个操作，可以实现更高的性能

较少的指令，从而减小了程序的体积

CISC架构通常能够更好地利用硬件资源，从而降低了系统的开销

·缺点：复杂的命令系统（使用、分析）。

复杂的控制单元和处理器设备。

难以生成高效的机器代码。

·CISC 架构在早期计算机中被广泛使用，因为它允许用单条指令执行复杂的操作，从而简化了编程并提高了某些类型任务的性能（当时内存非常昂贵）。

与冯诺依曼架构的区别：

冯·诺依曼架构（或冯·诺依曼原理）定义了计算机设计的基本原理，包括将程序和数据存储在同一内存中。 CISC 和 RISC 是指定义指令集及其执行方式的处理器架构类型。

编程特点：

CISC 处理器的编程通常更容易，因为丰富的指令集允许使用单个指令执行复杂的操作。 然而，由于解码指令的复杂性，优化 CISC 处理器的代码可能需要额外的工作。

---------------------------------------------------------------------------------------------------------------------------------

1. Что такое RISC? Роль в развитии компьютерных систем. Применение на практике. Достоинства и недостатки. Отличия от архитектуры фон Неймана. Особенности программирования.

什么是RISC？ 在计算机系统开发中的作用。 实际应用。 的优点和缺点。 与冯诺依曼架构的差异。 编程的特点。

·精简指令集计算机（RISC）是一种计算机架构，其核心设计理念是通过减少指令集的复杂性和增加硬件实现的简单性来提高计算机的性能。

·特点：简化指令集：RISC架构的指令集通常被设计为简单、固定长度，并且具有规整的格式。

鼓励流水线技术：RISC架构倾向于采用流水线技术，即将指令的执行过程划分为多个阶段，并在不同的阶段同时执行多条指令，从而提高了指令执行的并行度和效率。

注重硬件实现：RISC架构的设计目标之一是尽可能将复杂逻辑移至硬件层面实现，而不是依赖于微程序控制或微指令。

优化寄存器使用：RISC架构通常提供大量的通用寄存器，并且鼓励频繁使用寄存器进行数据操作，而不是通过内存访问来传递数据。

·优点：

高性能：由于指令集简化、硬件优化等原因，RISC架构通常能够提供较高的性能表现。

高效能：RISC架构鼓励使用寄存器进行数据操作，减少了内存访问次数，提高了程序的执行效率。

易于实现：RISC指令集的简化和硬件优化使得RISC架构的处理器设计更加容易实现，同时也降低了制造成本。

易于编译：RISC指令集的简单性使得编译器的设计和优化变得更加容易，有助于生成高效的机器代码。

适用于并行处理：RISC架构通常采用流水线技术和乱序执行等方法，使得处理器更容易实现并行处理，提高了系统的整体性能。

缺点：

可能需要更多指令：由于RISC指令集较为简单，实现某些复杂功能可能需要较多的指令，导致程序的体积增大。

不适合某些特定应用：对于某些特定的应用场景，如科学计算、图形处理等，可能需要更复杂的指令集来实现更高的性能。

硬件开销较大：尽管RISC架构鼓励硬件优化，但一些复杂的功能可能仍需要较多的硬件资源来实现。

·与冯诺依曼架构的区别：

冯·诺依曼架构（或冯·诺依曼原理）定义了计算机设计的基本原理，包括将程序和数据存储在同一内存中。 CISC 和 RISC 是指定义指令集及其执行方式的处理器架构类型。

·在RISC架构下编程的特点是，通常需要更多的寄存器来存储临时数据，以及更多的指令来完成相同的任务。但是，由于指令集的简单性，编写和优化程序可能会更加容易和高效。

---------------------------------------------------------------------------------------------------------------------------------

1. Конвейеризированное исполнение команд. Стадии конвейера. Виды конфликтов (по данным, по управлению), их примеры и влияние на производительность. Достоинства и недостатки.

流水线命令执行。 输送机阶段。 冲突类型（根据数据、管理）、示例以及对生产力的影响。 的优点和缺点。

·流水线通过将指令处理过程划分为互不冲突的阶段，允许指令并行执行。 指令分阶段一个接一个地执行，而不等待前一个指令完成。

1.Instruction Fetch

指令获取。 从程序计数器地址读取指令。

2.Instruction Decode

指令译码。 解码指令并读取寄存器。

3.Instruction Execute

指令执行。 执行更改数据的操作：

Register-Register Operation (Single-cycle latency)

寄存器-寄存器操作（单周期延迟）：加法、减法、比较和逻辑运算。

Memory Reference (Two-cycle latency)

内存参考（两个周期延迟）：为内存访问准备地址。

Multi-cycle Instructions (Many cycle latency)

多周期指令（多周期延迟）：整数乘法、除法和所有浮点运算。

4.Memory Access.

内存访问。 操作数从内存中读取和写入。

5.Write Back

回信。 将结果写入寄存器。

·如何建立输送机

1 确定处理指令所需的不同阶段。典型的阶段包括取指、解码、执行、访存和写回。

2 确保处理器的内部结构与指令执行的各个阶段相对应。有输入输出端。各段通过寄存器链接，所有部件都应该受到单一时钟信号的控制，以保持同步和协调。

3每个时钟周期都应该有一个指令被加载到处理器中，并且每个阶段都应该在时钟周期内完成其操作。

4 由于并行执行指令可能导致数据相关性和控制相关性的冲突，需要解决冲突

·数据冲突

由于管道执行的命令的依赖性而导致的冲突。

冲突始终与写入寄存器有关。

RAW -- Read after Write

(Data-dependency)

i1. R3 <- R1 - R2

i2. R5 <- R3 + R4

WAR -- Write after Read

(Anti-dependency)

i1. R3 <- R1 + R2

i2. R1 <- R4 + R5

WAW -- Write after Write

(Output dependency)

i1. R3 <- R1 + R2

i2. R3 <- R4 + R5

解决方法：

气泡插入

Out-of-order

operand forwarding

Register renaming

·管理冲突

由于条件和/或无条件跳转操作而导致的冲突。

不必要的命令可能会加载到管道中。

转换错误需要重置管道

解决方法：branch prediction

输送机

优点：提高生产力和资源利用率

缺点降低单个命令的执行速度；

并非所有操作都在一个机器周期内执行；

解决冲突的需要；

不可预测的执行时间；

与冯·诺依曼架构（单一内存）矛盾；

“间接渠道”的漏洞。

---------------------------------------------------------------------------------------------------------------------------------

1. Виды конфликтов при работе конвейера и механизмы их разрешения, сокращения их числа. Пузырёк, разворачивание циклов, предсказания переходов (статические и динамические).

输送机运行期间的冲突类型以及解决冲突和减少冲突数量的机制。 气泡、循环展开、转换预测（静态和动态）。

冲突类型：

·Structural Dependency结构冲突

不允许同时执行多个访问操作

解决方案：使用哈佛架构

使用双端口内存

·Data Dependency (Hazard) 数据冲突

RAW，WAR，WAW

out-of-order

重命名寄存器

操作数在处理器级之间转发，绕过寄存器文件。

·Control Dependency / Branch Hazards控制冲突

条件或者无条件跳转导致的冲突

分支预测

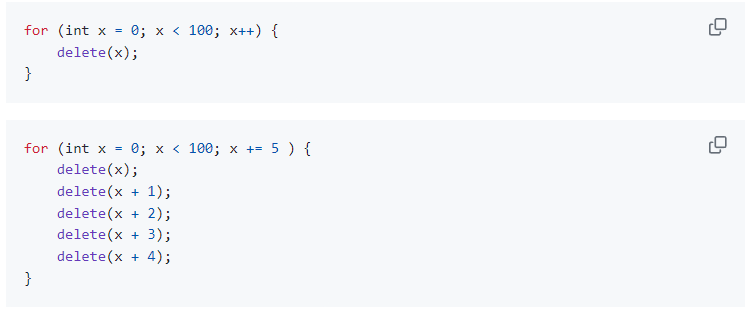
·Bubble Conflict Resolution

0空操作

占用传送带，不执行任何操作

Пузырёк (Pipeline Bubble)：在流水线处理器中，如果某个阶段无法立即执行下一条指令，会导致流水线停顿，这种停顿称为"пузырёк"，意思是流水线中的一个空闲周期。通常由于数据冲突或控制冲突而引起。为了解决这种问题，可能需要采取流水线暂停（stalling）或插入气泡（bubble）的方法。

Разворачивание циклов (Loop Unrolling)：是一种循环转换技术，有助于优化程序的执行时间。 我们基本上删除或减少迭代。 循环展开通过消除循环控制指令和循环测试指令来提高程序的速度。



Предсказания переходов (Branch Prediction)：这是一种处理器技术，用于预测分支指令的执行路径，以提高流水线的性能。分支指令可能导致流水线停顿，因为处理器无法立即知道分支的执行路径。

静态分支预测（Static Branch Prediction）通过基于代码结构和历史信息对分支进行预测。

动态分支预测（Dynamic Branch Prediction）则根据运行时的分支行为和历史信息进行预测，通常通过分支目标缓冲器（Branch Target Buffer）或者分支历史表（Branch History Table）来实现。

预测正确时，流水线可以继续执行下一条指令，否则可能需要插入气泡来等待分支结果。

---------------------------------------------------------------------------------------------------------------------------------

1. Что такое SOP (Stack-Oriented Processors, стековый процессор)? Роль в развитии компьютерных систем. Применение на практике. Достоинства и недостатки. Отличия от архитектуры фон Неймана. Особенности программирования.

什么是SOP（Stack-Oriented Processors，堆栈处理器）？ 在计算机系统开发中的作用。 实际应用。 的优点和缺点。 与冯诺依曼架构的差异。 编程的特点。

·SOP（面向堆栈的处理器）是一种使用堆栈来存储和管理数据和指令的处理器。 与数据存储在寄存器中的基于寄存器的处理器不同，基于堆栈的处理器将数据存储在堆栈上，堆栈按后进先出 (LIFO) 方式运行。

·在计算机系统开发中的作用：

- SOP 在某些专业领域具有其优势，例如嵌入式系统，其简单性和低功耗在这些领域具有优势。

·实际应用：

- SOP 用于嵌入式系统、微控制器和一些专用计算设备。

优点：

可以立即编写高级代码。 开箱即用（纯汇编）：过程调用、自动内存、递归（迭代）、逆波兰表示法的表达式和简单的编译器。

简单的命令系统带来高性能（速度很快）

对于缓存来说很方便，因为我们使用的是缓存的顶部

使用线程非常方便 - 我们只需保存堆栈

缺点：

如果我们需要的不是从栈顶获取数据+栈太大，效率就会下降。 由于这些问题，我们必须大量重新排列堆栈。

动态结构不能放入栈中

指令级并行是不可能的

结构上与经典不同

·与冯诺依曼架构的区别：

- 在冯诺依曼架构中，数据存储在存储单元中，指令和数据由各种设备处理。 在堆栈架构中，数据和指令存储在单个堆栈上

- 寄存器被堆栈取代，堆栈的顶部是一个或多个寄存器。 因此，命令中没有操作数，因为它始终只是从堆栈中弹出，因此指令系统非常不同。

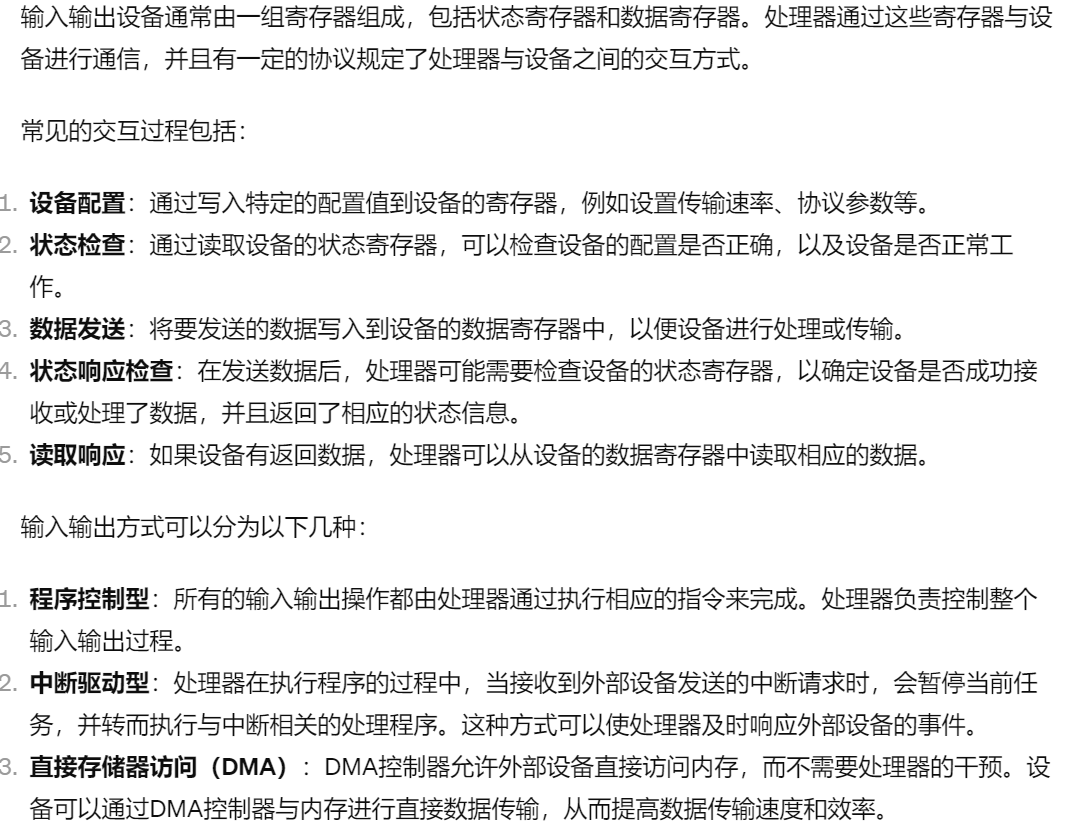
·编程特点：

- 堆栈处理器的编程需要使用特殊的堆栈指令，这在编写代码时可能需要额外的小心和注意。

---------------------------------------------------------------------------------------------------------------------------------

1. Поддержка операций ввода-вывода в фон Неймановских процессорах. Поддержка на уровне системы команд (порт, отображение в память). Программно-управляемый ввод-вывод. Ввод-вывод через механизм прерываний. Механизм прямого доступа к памяти.

支持冯·诺依曼处理器中的 I/O 操作。 命令系统级支持（端口、内存映射）。 软件控制的 I/O。 通过中断机制进行 I/O。 直接内存访问机制。



输入输出设备通常由一组寄存器组成，包括了状态寄存器和数据寄存器，处理器通过这些寄存器与设备进行通信，并且有一定的协议规定了寄存器与设备的交互方式

·通过内存输入/输出。 内存映射 I/O (MMIO)

外部设备寄存器映射到内存地址空间。

I/O是通过内存访问指令实现的。

无需更改处理器微架构即可实现

优点：处理器简单，统一的访问机制。通用命令系统（执行命令无优先级）

缺点：一根总线用于 I/O 和内存访问。内存异构，系统配置复杂。如果 I/O 慢于内存访问，性能可能会降低。

·通过端口输入/输出。 端口映射 I/O (PMIO)

I/O是通过专门的指令实现的。

I/O 设备的寻址独立于内存寻址（隔离 I/O）。

优点：io和内存分开，整体简单。最小化控制逻辑，更简单的I/O控制器，优化操作

缺点：命令系统和处理器的复杂化（输入/输出控制器、新指令）

I/O 操作不如其他内存访问操作

不必要的复制（寄存器→内存→寄存器）

- 软件控制的 I/O - 所有操作均由处理器执行，包括持续监控外部设备传输数据的准备情况。 需要大量时间与外部设备交互。 循环（轮询）中的处理器轮询外部设备的状态，并在必要时开始与其交互。

- 通过中断进行 I/O。 减轻处理器不断监视外部设备的任务，并允许根据外部事件来实现这一任务，例如，设备准备好传输数据或发生指定事件（定时器触发、按键按下等） 。 直接 I/O 仍然由处理器实现。 处理器收到中断后，立即中断当前控制线程，并沿着中断向量转移控制权，保存被中断线程的状态，完成后返回该线程，恢复原来的状态。

- 通道 I/O 和直接内存访问 (DMA)。 如果需要传输大量数据，则使用处理器来实现此目的是不合适的，因为它对这些任务来说具有绝对的冗余性。 在这种情况下，可以使用I/O处理器和直接内存访问控制器，它们允许处理器声明数据传输的需要（从哪里、哪里、多少），然后可以由专门的设备来实现，该设备将实际上提供数据传输并通过中断系统通知处理器结果。

---------------------------------------------------------------------------------------------------------------------------------

1. Параллелизм уровня задач. Кооперативная многозадачность. Принцип работы и подходы к реализации. Примеры использования. Достоинства и ограничения. Зелёные процессы. Проблема синхронизации процессов по управлению и по данным. Сравнение с альтернативами.

任务级并行性。 协作多任务处理。 操作原则和实施方法。 使用示例。 优点和局限性。 绿色流程。 同步控制和数据过程的问题。 与替代方案的比较。

并行性：当您需要同时处理多项任务时（输入输出、控制系统）。当你需要提高资源利用率时（我们不会闲着，而是做一些有用的事情）。当您需要提高计算机性能时（单位时间做更多的事情）。

并行类型：

位级并行性(Bit-level Parallelism)：位级并行性指的是计算机处理多个数据位的能力。

指令级并行性(Instruction-level parallelism)：指令级并行性是指处理器同时执行多条指令的能力。

任务/线程级并行性(Task/Thread-level parallelism)：任务/线程级并行性是指在计算机系统中同时执行多个程序或线程的能力。

协作多任务处理是一种多任务处理的方法，其中任务在需要时主动释放处理器控制权，以便其他任务获得执行的机会（其中当“当前任务”明确宣布准备放弃 CPU 时间时，执行“下一个任务”）。

在这种方式下，任务之间相互合作，以确保每个任务都能在适当的时候释放控制权。

1. 当前活动的程序正在运行，而且正在使用的处理器的全部时间片用来执行其任务，在这种情况下其他程序无法获得处理器时间
2. 这意味着后台的程序或者进程正处于冻结，不在活跃状态，不会占用处理器资源
3. 应用程序可以根据需要获取尽可能多的CPU资源
4. 处理器的时间片是按照一定的调度算法分配给不同的应用程序或进程。当一个程序完成其时间片时，控制权会被转移到下一个程序上，以便其他程序也能有机会执行。

计算机制

1 停止执行任务的机制：自愿将控制权转移给“调度员”。

2 保存任务状态的机制：寄存器、堆栈、协处理器状态、内存、I/O、缓存、分支预测器状态等。

3 调度机制 - 将处理器时间分配给下一个任务。

4 恢复停止进程的机制：恢复状态和转移控制。

– 1 任务隔离机制：独立执行、安全。

– 2 任务之间交互的机制：数据和信号的传输、共享资源。

优点：协作式多任务处理简单直观，易于理解和实现。它可以有效地利用处理器资源，并且不需要复杂的调度算法。

限制：如果某个任务出现死循环或长时间运行，会导致整个系统变得不稳定。此外，如果一个任务崩溃或挂起，可能会影响其他任务的执行。

绿色进程：绿色进程是指在协作式多任务处理中，任务需要在执行期间主动释放处理器资源，以避免长时间占用处理器导致其他任务无法执行的情况。绿色进程通常会定期检查自己的运行时间，并在需要时主动放弃处理器控制权。目的是为了保证系统的稳定性和响应速度，特别是在多任务环境下

同步问题：

当多个进程/线程访问共享资源时，同时访问这些资源就会出现问题。

同步允许您协调对资源的访问以避免冲突。

备择方案：

顺序执行：一次执行一个任务，无需并行。

异步执行：一种并行性，其中任务继续执行而不会阻塞其他任务。

---------------------------------------------------------------------------------------------------------------------------------

1. Параллелизм уровня задач. Вытесняющая многозадачность. Механизмы переключения задач. Примеры использования. Достоинства и ограничения. Проблема синхронизации процессов по управлению и по данным. Сравнение с альтернативами.

任务级并行性。 抢占式多任务处理。 任务切换机制。 使用示例。 优点和局限性。 同步控制和数据过程的问题。 与替代方案的比较。

抢占式多任务处理 - 一种多任务处理，其中操作系统本身在 I/O 操作完成、计算机硬件中的事件、计时器和时间片到期或收到某些信号从一个程序发送到另一个程序。

1. 抢占式多任务处理中，进程切换可以在任何时候发生，而不受当前进程执行状态的影响。’
2. 抢占式多任务处理中，调度程序负责根据任务的优先级、时间片等因素来确定任务的执行顺序和时间分配，以最大限度地提高系统的性能和响应速度。
3. 抢占式多任务处理使得操作系统能够在任何时候中断当前执行的任务，并立即转移到对用户操作的响应上

·机制：

1. 操作系统可以通过中断机制随时中断当前正在执行的任
2. 在中断进程之前，操作系统会保存当前任务的状态信息
3. 操作系统通过调度机制决定将处理器分配给哪个任务
4. 一旦调度器决定了下一个要执行的任务，操作系统会通过切换机制将处理器的控制权转移到该任务上
5. 抢占式多任务处理中，任务之间可以通过各种机制进行交互。实现协作和数据共享

·优点：1 资源隔离和保护：抢占式多任务处理确保不同进程或任务之间的资源相互隔离和保护，防止它们相互干扰或滥用系统资源。

2 快速响应：由于操作系统可以随时中断当前正在执行的任务，并立即转移到其他任务上，因此抢占式多任务处理可以实现快速响应

3 交互式系统：抢占式多任务处理能够在用户进行操作时迅速地响应，提供几乎即时的用户体验，适用于交互式系统

·缺点：

1 由于操作系统可以在任何时候中断当前任务，因此任务之间的切换时机难以预测，可能导致不可预测的系统行为和性能波动。

2 抢占式多任务处理可能导致竞争条件和同步问题的出现，例如数据竞争、死锁等，需要额外的同步机制来解决。

3 在处理器级别，线程切换可能会带来较大的开销，包括上下文切换所需的时间和资源消耗，导致性能下降。

同步问题：

当多个进程/线程访问共享资源时，同时访问这些资源就会出现问题。

同步允许您协调对资源的访问以避免冲突。

备择方案：

与协作多任务处理相比，其中任务本身决定何时释放处理器。

抢占式多任务处理更具可预测性，并提供更稳定的系统控制。

---------------------------------------------------------------------------------------------------------------------------------

1. Система прерываний. Виды прерываний. Механизм обработки прерываний по шагам. Задачи, решаемые механизмом прерываний. Сторожевой таймер.

中断系统。 中断的类型。 中断处理机制逐级进行。 通过中断机制解决的问题。 看门狗定时器。

冯·诺依曼架构基于指令顺序执行的原理和条件分支的可能性。 他们使得这种类型的处理器的编程成为一个相对简单和灵活的过程。

缺点是无法从外部更改控制流；只要处理器认为有必要，就会不间断地“消化”命令序列

这种行为并不总是方便，因为：

● 不允许您有效地处理外部事件和输入/输出设备（我们被迫通过轮询来执行这些操作，而不是通过新信息的出现）；

● 除了使用错误代码或标志之外，不允许处理异常情况（这对软件开发纪律提出了很高的要求）；

● 不允许多个命令线程独立执行（抢占式多任务）。

这个问题可以通过一种称为“中断”的机制来解决，该机制允许您向处理器发出信号，表明当前控制线程应该：

·立即被打断；

·其状态为已保存；

·通过指定的中断向量转移；

·中断处理程序完成后，控制权必须返回到原始控制线程并恢复状态。

中断的类型：

1 异步 与程序执行无关的中断，通常由外部事件引起。可以在与处理器相关的操作的任何时间发生。例如，定时器、网卡、按键。

2 同步 程序执行期间产生的中断，通常由特定的条件或指令触发。示例：除以零或堆栈溢出、访问无效内存地址、无效操作码。

3 软件 由软件指令生成的中断，通常用于向操作系统或其他软件组件发送信号或请求

从处理器的角度来看，中断处理是一个相当复杂的过程：

● 决定是否处理中断。 中断分为：

○ 可屏蔽中断，开发人员可以使用这些中断，其配置允许忽略它们。 例如：在代码关键部分的执行中禁用中断。

○ 不可忽略的不可屏蔽中断。 例如：访问主存储器或处理看门狗定时器（WatchDogtimer）时出错。

● 中断优先级。 如果在中断处理过程中有另一个中断到来，则分为：

○ 相关中断（可以稍后处理）；

○ 绝对（必须中断当前中断并立即继续处理新的中断）。

通过中断机制解决的问题：

·多任务处理（不同的任务之间进行快速切换）

·异常处理

·实时响应，中断当前正在执行的任务，并立即转移到处理中断的代码

看门狗定时器（Watchdog Timer）是一种硬件计时器，通常用于监视和保护计算机系统或嵌入式系统的正常运行。它的作用类似于一个“看门狗”，定期检查系统是否处于正常运行状态，如果发现系统出现故障或停滞，它会自动触发系统复位或其他预定义的故障处理机制。

--------------------------------------------------------------------------------------------------------------------------------

1. Параллелизм уровня задач. Проблемы совмещения, изоляции и взаимодействия задач между собой. Методы разрешения данных проблем (с точки зрения опыта программиста и пользователя): распределение по адресному пространству, банки памяти, сегментная организация памяти и виртуальная память.

任务级并行性。 任务相互组合、隔离和交互的问题。 解决这些问题的方法（从程序员和用户体验的角度）：地址空间、存储体、分段内存组织和虚拟内存的分配。

命令流的隔离是必要的，以便命令流仅对其自己的数据进行操作，并且不能直接影响其他命令/数据流。 实际系统中的这种隔离是多级的，并且通过硬件（由于处理器架构考虑了访问权限）和软件级别（通过操作系统）实现

隔离是指通过机制和政策，将系统的各个组件相互分离和保护的过程。这包括进程、线程、用户、应用程序、资源和数据的隔离，以防止未经授权的访问或干扰，并确保系统的安全性和稳定性。

1 Memory Bank：内存银行是计算机系统中的一种硬件结构，用于将物理内存划分为多个独立的区域，每个区域称为一个内存银行。

2 Segmentation：分段是一种管理虚拟内存的方法，它将进程的地址空间划分为逻辑上相关联的块，称为段。

3 Virtual Memory：虚拟内存是一种计算机系统的概念，通过它，操作系统为每个进程创建了一个巨大的地址空间，即使物理内存不足。

·并发：Concurrency

指在计算机系统中同时或并行执行多个任务或进程的能力。

问题：不同的任务可能会竞争共享资源，例如共享内存或 I/O 设备。

解决方法：使用互斥体、信号量等同步机制来保证共享资源访问的互斥性。

·隔离：Isolation

确保系统中不同任务或进程之间的分离和保护

问题：任务可以相互影响，这可能会导致不可预测的结果。

解决方法：同上

·相互作用：Interaction

不同任务或进程之间数据交换和互动的方式

问题：任务可能需要通信或同步其执行。

解决方法：使用消息队列、共享内存、异步事件等进程间通信（IPC）机制来同步任务。

·地址空间分配：

程序员的经验：程序员可以有效地利用不同的地址空间区域来执行不同的任务，从而减少冲突并提高性能。

用户的经验：用户可以观察到应用程序之间的独立运行，确保系统平稳运行。

·内存银行：

程序员的经验：程序员可以将数据有效地分配到不同的内存银行中，以改善资源管理并避免冲突。

用户的经验：用户可以在同时运行多个任务时，感受到系统的稳定性，因为内存银行有效地利用了资源。

·内存分段：

程序员的经验：程序员可以使用分段来隔离数据和任务，确保安全性并有效地进行交互。

用户的经验：用户可以观察到系统的稳定性和安全性，即使一个任务发生错误也不会影响其他任务的运行。

·虚拟内存：

程序员的经验：程序员可以开发具有大量地址空间的应用程序，而无需担心物理内存的限制，这得益于虚拟内存的使用。

用户的经验：用户可以同时运行多个应用程序，而不必担心内存不足的问题，因为虚拟内存有效地管理了资源。

--------------------------------------------------------------------------------------------------------------------------------

1. Сегментная и виртуальная память. Решаемые задачи и принципы работы. Проблема фрагментации. Достоинства и недостатки.

段内存和虚拟内存。 需要解决的问题和运行原则。 碎片问题。 的优点和缺点。

段内存寻址是x86架构中计算机内存逻辑寻址的一种方案。 特定存储单元的线性地址分为两部分：段和偏移量。 段是一定大小的地址空间中有条件分配的区域，偏移量是内存单元相对于段开头的地址。 段基址是一个线性地址（相对于整个内存量的地址），它指向地址空间中段的开头。 结果是一个段（逻辑）地址，对应于段基址+偏移量的线性地址，并由处理器设置到地址总线。

优点：提高了内存管理的灵活性，每个段都可以独立管理，易于调试和优化程序。

缺点：可能会导致内存碎片化，降低内存利用率。

目的：

● 命令流内相对于段的独立寻址。

● 管理对段中数据的访问权限（rwx，特权模式）。

● 程序模块之间独立寻址和隔离。

● 为不同的命令线程提供相同的内存段进行交互。

虚拟内存在某种意义上是段内存的发展，同时也是段内存的简化。 与分段内存的主要区别在于，从应用软件的角度来看，它是透明的，而且虚拟内存地址空间（地址总线为 32 位 - 4 GB）不必与物理内存地址空间（可能更少，512MB），同时，使用物理内存并不一定需要使用物理内存（工作在缓存原理上）。

优点：扩展了可用内存的容量，允许在物理内存不足时仍然运行大型程序。

缺点：增加了访问磁盘的开销，可能导致性能下降。虚拟内存管理的复杂性也会增加系统的开销。

·内部碎片：

内部碎片是指分配给进程的内存块中有一部分空间没有被使用，但由于内存分配是以固定大小的块为单位进行的，这些未使用的空间无法被其他进程利用。

内部碎片通常发生在采用固定大小的分配单位时，例如固定大小的分页系统或固定大小的内存块分配。

·外部碎片：

外部碎片是指内存中有很多小的、不连续的空闲块，但它们的总大小却足够无法满足大块内存的请求。这些空闲块之间的碎片化使得无法满足大型内存分配请求。

外部碎片通常发生在采用动态内存分配策略时，如首次适应、最佳适应或最差适应分配算法。由于动态分配可能在内存中留下不连续的空闲块，因此会导致外部碎片问题。

--------------------------------------------------------------------------------------------------------------------------------

1. Иерархия памяти (явная и скрытая). Виды памяти. Особенности использования на практике. Устройство памяти с произвольным доступом. Устройство и принцип работы ROM, SRAM, DRAM ячеек.

内存的层次结构（显式和隐藏）。 记忆的类型。 实际使用的特点。 随机存取存储设备。 ROM、SRAM、DRAM 单元的设计和工作原理。

计算机系统中的内存解决了两个主要问题：

● 存储系统运行所需的初始数据（软件、设置和初始值）；

● 存储和更新工作数据，可以减少计算机系统功能的处理（视频流、计算数据、文本文档）。

计算机内存有大量不同类型，以下规则成立：随着内存速度的提高，其成本（生产和放置在正确的位置）也会增加。

内存层次结构指的是计算机系统中不同类型内存的层次化组织，旨在实现数据访问的最佳性能和效率。

显式内存层次是指系统中明确定义的内存层次结构，例如缓存、RAM和外部存储器（如硬盘）。

隐式内存层次包含系统中被硬件和软件利用以优化数据访问的隐含级别，例如虚拟内存和操作系统内核级缓存。

内存类型：

记忆的类型呈现在记忆金字塔中。 但是，内存也按访问类型划分：

● 通过随机访问，内存可以在任何地址提供数据，并且相对于上次请求具有相同的延迟。

● 顺序访问，包括数据流意义上的顺序访问（网络、输入等）、磁带、硬盘等。

● 混合选项也是可能的：磁带库，其中对磁带的访问是随机实现的，而对磁带内数据的访问是顺序的。

实际使用特点：

从实际的角度来看，绝大多数情况下的顺序访问并不意味着不能进行随机访问，而是持续时间非常高。 例如，硬盘驱动器对数据的访问取决于磁头在磁盘上方的物理定位，这意味着顺序读取数据可以非常快，而移动和搜索由于等待数据而需要很长时间。磁头移动，磁盘旋转。

P.S：如果我们谈论内存金字塔中呈现的内存类型，那么实际使用的特征就是成本/体积/速度的比率

随机存取存储器设备：

对内存数据的随机访问是通过数据存储的特殊组织来实现的。一般来说，这是一个具有存储单元阵列。 结构的规律性简化了其扩展并增加了数据存储密度。 大型数据仓库的结构更加复杂，从而减少了数据行的数量。

·ROM（只读存储器）：

设计：ROM用于存储固定数据，如计算机的基本输入/输出系统（BIOS）或不经常更改的固件。ROM的内容在制造时被编程，一旦编程完成，数据就无法被修改。

工作原理：ROM存储器单元使用非易失性存储器技术，例如闪存或EPROM（可编程只读存储器）。数据被写入ROM单元时，改变内部电子设备的状态，以表示二进制值0或1。这些状态在供电断开后仍然保持不变，因此ROM可以保留存储的数据。

·SRAM（静态随机存取存储器）：

设计：SRAM用于构建处理器缓存等需要快速访问的应用，因为它具有较快的访问速度和较低的功耗。

工作原理：SRAM存储器单元由多个互相连接的存储单元组成，每个单元由一个双稳态触发器（flip-flop）构成，可以存储一个比特的数据。SRAM中的数据通过在两个稳态之间切换来存储和检索。由于采用了这种触发器结构，SRAM可以在不需要刷新的情况下持续保持数据，因此速度更快。

·DRAM（动态随机存取存储器）：

设计：DRAM通常用作主内存，具有相对较大的容量和相对较低的成本，但速度比SRAM慢。

工作原理：DRAM存储器单元由一个电容器和一个访问晶体管组成，电容器存储每个位的数据，而访问晶体管用于控制读取和写入操作。DRAM中的数据通过在电容器中存储电荷来表示。由于电荷会逐渐泄漏，所以DRAM需要定期刷新以保持数据的完整性。这种刷新操作会导致DRAM的访问速度相对较慢，但它能够实现更大的存储容量和更低的成本。

--------------------------------------------------------------------------------------------------------------------------------

1. Механизм кеширования в компьютерных системах, принцип локальности. Функционирование кеш памяти процессора (чтение, запись). Виды кеш промахов. Механизм вытеснения (LRU, PLRU).

计算机系统中的缓存机制，局部性原理。 处理器高速缓存的功能（读、写）。 缓存未命中的类型。 位移机制（LRU、PLRU）。

缓存是一种用于提高计算机系统性能的重要机制，它利用局部性原理存储最近或频繁使用的数据，并通过提供快速访问来减少对主存储器的访问。

缓存由一组记录（缓存块、缓存行）组成，每个记录与慢速存储器中的一个元素相关联。每个记录都有一个标识符（标签），用于确定匹配项。内存访问是透明的，意味着系统会尽可能地隐藏访问慢速存储器的延迟。无论缓存的存在，内存都可以被修改，例如通过DMA（直接内存访问）或其他核心。

从缓存中读取数据时：

·未找到标签：这种情况称为缓存未命中。当请求的数据不在缓存中时，需要从主存中获取数据。这个过程的持续时间是不可预测的，因为主存的访问速度较慢。

·缓存已经繁忙：这是一种典型情况，其中缓存中的所有条目都已被占用，没有空闲的位置存储新数据。在这种情况下，需要进行决策以选择哪些数据应该被替换或驱逐出缓存。这涉及到缓存的替换策略和缓存层次结构的考虑。

·标签已经找到：这是一种缓存命中的情况，表示请求的数据已经在缓存中。在这种情况下，数据可以直接从缓存中传输到处理器，而不需要访问主存。

将数据写入缓存时：

·立即写入（直写）：即时更新主存，确保数据的一致性。有时可能比没有缓存的情况下要慢，但是操作是可预测的。

·延迟写入（回写）：在数据被替换或需要时定期或按需更新主存。通过分组更新和隐藏中间状态来提高效率。

·修改标志：用于标识缓存中的数据是否已被修改。这可能需要两次内存访问：一次写入被替换的数据，一次读取需要的数据。

·混合选项：例如，带有缓冲区的即时写入，结合了立即写入和延迟写入的特点，提供了一种平衡的解决方案。

缓存未命中不同的类型：

·指令读取未命中：这是最严重的类型，因为处理器必须等待读取指令。这会导致最大的延迟，处理器被迫空闲。

·数据读取未命中：相对于指令读取未命中，延迟较低，因为处理器可以执行其他不相关的指令，如超标量和重新排序。

·数据写入未命中：这种类型的未命中延迟最小，因为写入可以被排队，并且不会导致处理器空闲。

Least recently used (LRU);根据最近使用的历史记录来确定哪些数据应该保留在缓存中。当缓存已满并需要替换数据时，LRU算法会淘汰最长时间没有被访问的数据。

Pseudo-LRU.使用更少的硬件资源来实现LRU的效果。PLRU算法通过维护一个二叉树结构来决定替换哪些数据。在每次替换操作中，PLRU算法会根据当前数据的使用情况调整二叉树的状态，并根据状态来确定应该被替换的数据。

--------------------------------------------------------------------------------------------------------------------------------

1. Устройство кеш-памяти процессора. Ассоциативность кеш-памяти (полностью ассоциативная, прямое отображение, множественно-ассоциативный кеш). Принципы работы. Детальное описание принципов работы кеш-памяти с разными вариантами ассоциативности.

处理器高速缓存设备。 缓存关联性（全关联、直接映射、多关联缓存）。 工作原则。 详细描述具有不同关联性选项的高速缓存的操作原理。

缓存存储器通常由一组缓存行组成，每个缓存行包含一个存储单元（块或线）。缓存存储器的每个存储单元都与主存储器中的某个存储单元相关联，并包含两部分：数据和标记（tag）。标记用于识别关联的主存储器块，数据则是主存储器中对应块的副本。

缓存关联性指的是缓存中存储的数据与主存中的数据之间的关系。在缓存中，每个数据块都有一个标签（Tag），它是一个唯一的标识符，用于标识该数据块在主存中的位置。当处理器需要访问某个内存位置时，它首先检查缓存中是否存在该数据块，这需要比较缓存中数据块的标签与待访问数据的标签是否匹配。如果匹配成功，就会发生缓存命中；否则，就会发生缓存未命中。

直接映射缓存（Direct Mapped Cache）：每个内存块只能映射到缓存中的一个特定位置，这种关联性称为一对一关联性。

直接映射缓存的工作原理如下：

·内存地址被划分为标签、索引和偏移三部分。索引用于确定数据存储在哪个缓存行中。

·当处理器需要访问某个内存位置时，使用索引来确定对应的缓存行。

·如果该缓存行的标签与待访问数据的标签匹配，则发生缓存命中；否则，发生缓存未命中。

全关联缓存（Fully Associative Cache）：内存中的任何一个数据块都可以存储到缓存的任何位置，缓存中的每个位置都可以存储来自内存的任何数据块，这种关联性称为全关联性。（更高的效率，很多逻辑）

全关联缓存的工作原理如下：

·当处理器需要访问某个内存位置时，首先检查所有缓存行的标签，以确定是否存在该数据块。

·如果存在，则发生缓存命中，处理器直接从缓存中读取数据。

·如果不存在，则发生缓存未命中，需要从主存中加载数据到缓存中的某个空闲缓存行，并更新相应的标签。

组相联缓存（Set Associative Cache）：缓存被划分为多个组，每个组包含多个缓存行。每个内存块可以映射到一个组中的任何一个位置，但是每个组内部的映射是直接映射的，这种关联性介于直接映射和全关联之间。（基于抢占算法选择bank，更强的关联性，更低的速度，更高的效率）

多关联缓存的工作原理如下：

·内存地址被划分为标签、组索引和偏移三部分。组索引用于确定数据存储在哪个组中。

·当处理器需要访问某个内存位置时，首先确定对应的组。

·在该组内部，使用直接映射的方式来确定具体的缓存行。

·如果待访问数据的标签与缓存行的标签匹配，则发生缓存命中；否则，发生缓存未命中。

--------------------------------------------------------------------------------------------------------------------------------

1. Иерархия кеш-памяти процессора. Причины множества уровней кеша. Разделённый/унифицированный, включающий/исключающий, частный/общий. Типовые уровни кеша в современных процессорах.

处理器缓存层次结构。 多个缓存级别的原因。 分割/统一、包容/排他、私有/通用。 现代处理器中的典型缓存级别。

·L0 是最专门化的缓存，通常大小为 256 字节，不一定在所有处理器中都存在。通常具有狭窄的专门化用途，例如用于缓存堆栈、整数变量、FPU（浮点数单元）中的浮点数等。该级别的缓存通常能够在一个时钟周期内访问数据，因此直接集成到处理器中。

·L1 缓存（一级缓存）是最快速的缓存。它实际上是处理器的一部分，位于同一芯片上，并且根据哈佛架构分为指令缓存和数据缓存。大多数处理器都依赖于 L1 缓存的存在。L1 缓存工作在处理器的时钟频率下，通常可以在每个时钟周期内进行访问，有时可以执行多个读/写操作。

·L2 缓存通常也位于同一芯片上，与 L1 缓存相邻。在早期处理器中，L2 缓存是作为单独的芯片组件存在于主板上的。L2 缓存的容量通常介于 128KB 到 1-12MB 之间。在现代多核处理器中，L2 缓存作为共享缓存，分配给每个核心，每个核心的 L2 缓存大小相等。

·L3 缓存的容量可以达到 24MB 或更多，速度比前面的缓存略慢，但仍然比主内存快得多。在多处理器系统中，L3 缓存通常是共享的，用于同步不同核心的数据，特别适用于支持并行编程的多核系统。

·L4 缓存是一种相对较为罕见的缓存，通常只在高性能服务器和大型机中使用。它通常由单独的芯片实现。专家指出，L4 缓存可能会采用 eDRAM 技术实现，以提高与主存储器的数据传输带宽，而不是为了快速访问数据。

多个缓存级别的原因

·层次化设计：缓存层次结构可以有效地利用存储器层次结构的局部性原理。较小但更快速的缓存（如 L1 缓存）可以存储最常用的数据，而较大但速度较慢的缓存（如 L3 缓存）可以存储更多的数据

·减少主存访问：较快的缓存层可以在处理器需要数据时快速提供，减少了对主存储器的访问次数。

·减少总线和内存带宽负载： 多级缓存可以减少处理器对总线和内存带宽的占用

·分隔式（Split）/统一式（Unified）：分隔式缓存将指令和数据存储在不同的缓存中，而统一式缓存则将指令和数据混合存储在同一个缓存中。

·包含式（Inclusive）/排斥式（Exclusive）：包含式缓存包含了较低级别缓存中的所有数据，而排斥式缓存则不包含较低级别缓存中的任何数据，这两种类型的缓存管理方式会影响缓存的一致性和性能。

·私有式（Private）/共享式（Shared）：私有式缓存专门为单个处理器核心或线程设计，而共享式缓存则可被多个处理器核心或线程共享。

--------------------------------------------------------------------------------------------------------------------------------

1. Когерентность кеш-памяти. Возможные состояния кеш линий. Механизмы обмена информацией между кешами: справочник, отслеживание и перехват. CAP теорема.

缓存一致性。 缓存行的可能状态。 缓存之间交换信息的机制：目录、窥探和拦截。 CAP定理。

缓存一致性是指多个处理器或处理器核心之间共享的缓存数据保持一致的状态。

缓存行的可能状态：

Modified（修改）： 表示缓存中的数值是脏的，即当前缓存中的数值与主存中的不同。

Exclusive（独占）： 表示缓存中的数值与主存中的相同，即该数值是干净的。

Shared（共享）： 表示缓存中保存着最新的数据副本，并且这份数据是所有缓存和主存共享的。

Owned（拥有）： 表示当前缓存持有该块，并且现在是该块的所有者，拥有对该特定块的所有权利。

Invalid（无效）： 表示当前缓存块本身无效，需要从其他缓存或主存中获取。

·目录（directory）： 是一种管理分布式系统中缓存数据一致性的方法。它包含了系统中所有内存块的状态信息，以及哪个缓存包含了这些内存块的副本。物理上，目录可以分布在不同的节点或处理器上。

·跟踪（snooping）： 每个缓存都维护有关内存块的状态信息。当其他处理器或设备访问内存时，所有缓存都会“窥探”这些访问，以确保数据的一致性。例如，如果一个处理器写入了某个内存块，其他缓存可能会通过“窥探”该写入操作来更新自己的副本。

·拦截（snarfing）： 当一个缓存写入某个内存块时，其他缓存的控制器会“拦截”这个写入操作，以便及时更新自己的副本。这确保了系统中所有缓存之间的数据一致性，即使数据被修改，所有缓存也会得到更新。

CAP定理指出，在分布式系统中，无法同时满足一致性（Consistency）、可用性（Availability）和分区容忍性（Partition tolerance）：

·一致性（Consistency）： 指系统中的所有节点在同一时间具有相同的数据视图，即任何时刻对系统的读操作都会返回最新的写操作结果。

·可用性（Availability）： 指系统中的所有节点都能够在有限的时间内响应客户端的请求，即系统能够保证服务的可用性和及时性。

·分区容忍性（Partition tolerance）： 指系统能够在面对网络分区的情况下继续运行，即使系统的一部分节点之间的通信被切断，系统仍然可以继续工作。

--------------------------------------------------------------------------------------------------------------------------------

1. Закон Мура. Закон Деннарда. Закон Амдала. Power-wall. Memory-wall. Их роль в развитии компьютерных систем. Источники роста производительности процессоров тогда и сегодня.

摩尔定律。 登纳德定律。 阿姆达尔定律。 电源墙。 记忆墙。 他们在计算机系统开发中的作用。 当时和今天处理器性能增长的来源。

摩尔定律： 它预测了集成电路中可容纳的晶体管数量将每隔约两年翻一番，从而带来处理器性能的指数级增长。这个观点在计算机领域长期被认为是一种发展趋势，但在最近几年受到了一些挑战。

丹纳德定律： 存储器（如RAM）的容量和速度每隔大约十年增加十倍，而价格保持不变。这个定律在存储技术领域被广泛引用，它指导了存储器技术的发展。

阿姆达尔定律： 它描述了并行计算系统中加速比的上限。该定律认为，一个程序中可以并行执行的部分所占比例越大，加速比就越接近理想值。但是，存在串行部分的存在会限制加速比的提高。

功耗墙： 由于处理器频率的提高和晶体管密度的增加，处理器的功耗也随之增加。功耗墙指的是在继续提高处理器性能的同时，处理器的功耗和散热问题成为了限制性因素，导致了进一步提高处理器性能的困难。

内存墙： 内存墙是指处理器和内存之间速度差异的问题。处理器的速度远远快于内存的速度，导致了内存访问成为了性能瓶颈。为了解决这一问题，需要采取各种优化策略，如缓存、预取等。

值得强调的是摩尔定律对世界的三种影响：

● 开发商之间的竞争，即为创造更快、更强大的处理器而争夺主导地位；

● 算力架构发展：技术算法持续定期更新（每2年一次）；

● 市场预测。

自 2010 年以来，遵循摩尔定律已不再对开发者有利。 为了遵守它，你需要花费大量的资源：材料、设备、增加人员等。 截至2019年，摩尔定律已不再有效发挥作用；硅晶体管的时代大概将在2030年之前结束。

• 当前技术的局限性（复杂性和生产成本）。

• 缩小晶体管的可能性的物理限制（原子尺寸和与缩小晶体管相关的效应）。

但登纳德缩放定律在 2006 年左右不再起作用。芯片中晶体管的数量不断增加，但这一事实并没有带来器件性能的显着提高。

随着晶体管尺寸的减小和频率的增加，电流开始使微电路升温更多，这可能会损坏它。

从2006年开始，量产芯片的频率就定在4-5GHz。

--------------------------------------------------------------------------------------------------------------------------------

1. Проблема обеспечения реального времени в современных компьютерных системах. Влияние параллелизма уровня инструкций, языков программирования высокого уровня, многозадачности и организации памяти.

现代计算机系统中提供实时性的问题。 指令级并行性、高级编程语言、多任务处理和内存组织的影响。

实时供应的问题是确保任务在给定时间内完成。 这对于延迟或错过任务可能会产生严重后果的系统尤其重要，例如控制系统、安全系统或信号处理系统。

指令级并行允许同时执行多条指令。 这可以提高系统性能，但也会导致实时任务的延迟。 这是因为指令可能以随机顺序执行，并且系统可能不确信任务会在指定时间内完成。

高级编程语言可以使代码更加复杂且更难以预测。 这可能会导致系统性能降低并增加实时任务延迟的可能性。

多任务允许系统同时执行多个任务。 这可以提高系统性能，但也会导致实时任务的延迟。 这是因为系统必须在任务之间分配资源，并且实时任务可以排在其他任务后面。

内存组织会影响系统性能和实时任务中延迟的可能性。 例如，如果系统使用虚拟内存，则实时任务可以被另一个任务从 RAM 中取代。 这可能会导致实时任务执行的延迟。 您还可以通过更新它（DRAM）和DMA来给出多级缓存和内存锁定的示例

为了提供实时性，现代计算机系统使用各种方法。 这些包括：

1. 实时操作系统 (RTOS)：专门设计的操作系统，可保证在严格定义的时间段内执行任务和处理事件。 即使在高负载条件下，RTOS 也能提供可预测性和性能。

2、实时调度：采用各种调度算法实时对任务的执行进行优先级排序和控制。 其中包括固定间隔调度、速率单调调度、最早截止日期优先等。

3.实时硬件支持：一些计算机系统具有特殊的硬件支持，例如实时协处理器或硬件定时器，它们提供定时间隔的精确测量和控制。

4. 隔离和优先级：为了确保实时性，可以隔离关键任务并使其优先于系统中的其他任务。 采用优先级、中断、阻塞等机制，保证重要任务在规定时间内完成。

5.性能和延迟优化：在需要实时性的计算机系统中，进行特殊的优化和调整以减少延迟并提高性能。 这可能包括优化代码、尽量减少高阻塞外部设备的使用等。

6.验证和测试（定时输入输出自动机）：实时需要对系统进行仔细的验证和测试。 这包括使用边界检查、时序分析、仿真和其他技术来确保系统满足实时要求。

--------------------------------------------------------------------------------------------------------------------------------

1. Уровневая организация компьютерных систем. Элементы уровня организации. Уровневый архитектурный стиль. Примеры. Явление разделения на уровни (disaggregation) и их смешения. Документирование инструментальных цепочек.

计算机系统的层次组织。 组织层面的要素。 层次建筑风格。 例子。 分层（分解）及其混合的现象。 工具链的文档。

计算机系统的分层组织指的是将系统划分为多个层次，每个层次负责特定的功能或任务，层与层之间通过接口进行通信和交互

计算机系统的层次组织是组织计算系统的自然方式

层次组织的要素包括编程语言中的基本结构，如条件语句（if）、循环语句（cycle），以及由平台本身提供的基本元素，如数据类型（int、long）、实例化（标准库）。此外，还包括在这些基本元素之上构建的设计模式和最佳实践，这些模式和实践有助于高效地设计程序并解决问题。

层次式架构风格是一种设计风格，其中所有可用的模块被划分为独立的集合，其中较高级别的集合可以访问较低级别的集合。

例如：一家商业公司拥有业务逻辑部门和基础设施部门。基础设施部门的人员实现了位于较低级别的模块，而业务逻辑部门的员工在此基础上实现了业务逻辑，这位于较高级别。

通常为了提高性能和安全性，我们不得不混合使用不同级别的组件。由此破坏了抽象化，增加了系统和开发流程的复杂性。

工具链文档化是指对系统中使用的工具链进行文档化，包括工具的功能、用法、参数设置等信息，以便开发人员和维护人员能够更好地理解和使用这些工具。

优点：如果系统中有很多层次和相互关联，那么我们可以轻松地在一张纸上展示它们，并找到这些链条的起点和终点。

--------------------------------------------------------------------------------------------------------------------------------

1. Особенности реализации структурного программирования в фон Неймановских процессорах. Работа с памятью, регистрами. Реализация процедур. Реентерабельность.

冯·诺依曼处理器中结构化编程的实现特点。 使用内存和寄存器。 程序的执行。 重新进入。

在冯·诺伊曼结构的处理器中实现结构化编程有以下特点：

·模块化设计： 结构化编程将程序分解为多个模块或函数，每个模块执行特定的任务。在冯·诺伊曼结构中，这些模块通常对应于程序中的不同功能或操作，它们被设计为相对独立并且可以被重复使用。

·逻辑清晰： 结构化编程强调逻辑上的清晰性和可读性，使得程序的结构更易于理解和维护。在冯·诺伊曼结构中，这意味着使用清晰的控制结构（如if语句、循环和函数调用），以及良好的命名和注释来说明代码的用途。

·控制结构的使用： 结构化编程倾向于使用顺序、选择和循环等基本控制结构来组织程序的逻辑流程。在冯·诺伊曼结构中，处理器能够执行这些控制结构所需的指令，从而实现程序的逻辑流程。

·局部性原理： 结构化编程通常遵循局部性原理，即相关的数据和操作应该在空间和时间上尽可能靠近。在冯·诺伊曼结构中，这意味着程序应该尽可能地利用处理器的缓存和寄存器来提高访问速度和效率。

·可读性和可维护性： 结构化编程的一个重要目标是编写易于阅读、理解和维护的代码。在冯·诺伊曼结构中，这可以通过遵循良好的编程实践、模块化设计和适当的注释来实现。

·处理内存和寄存器的工作： 冯·诺伊曼结构的处理器通过指令集中的指令来执行操作。结构化编程通常涉及对内存中存储的数据进行操作，以及使用寄存器来保存临时数据或执行指令。处理器在执行程序时，会根据指令从内存中读取数据或指令，并将结果写回内存或寄存器。

·程序过程的实现： 在结构化编程中，程序通常被组织为模块化的过程或函数。这些过程或函数执行特定的任务，并可以接收输入参数并返回结果。在冯·诺伊曼结构的处理器中，这些过程或函数的实现通常通过一系列的指令来完成，这些指令依赖于处理器的指令集架构。

·可重入性： 可重入性是指一个过程或函数可以被多个同时运行的线程或进程调用，而不会产生不正确的结果或导致程序崩溃。在冯·诺伊曼结构的处理器中，实现可重入性通常涉及对共享资源的正确管理，例如对内存和寄存器的访问，以及对临界区的同步访问。

--------------------------------------------------------------------------------------------------------------------------------

1. Уровни параллелизма. Параллелизм уровня бит. Низкоуровневый параллелизм. Параллелизм уровня инструкций. Параллелизм уровня задач. Примеры и особенности.

并行级别。 位级并行。 低级并行性。 指令级并行性。 任务级并行性。 示例和特征。

并行级别： 指的是并行计算中的不同层次或类型，用于同时执行多个任务或操作的能力。

位级并行： 涉及对数据的单个位或比特进行并行处理的能力。例如，同时对一个字中的每个位进行操作，以加速数据处理。这种并行性在硬件电路中很常见，可以通过并行处理单个字中的多个位来提高计算速度。

低级并行性： 指在单个处理器或核心内部同时执行指令或操作的能力。这种并行性通常通过流水线处理器架构来实现，允许在单个时钟周期内执行多个指令的不同阶段。例如，处理器可以同时从内存中读取指令、解码指令、执行指令和写回结果，从而提高了处理器的吞吐量。

指令级并行性： 涉及同时执行多个程序指令的能力。这种并行性通过超标量、超流水线和乱序执行等技术来实现，在单个时钟周期内执行多个指令。例如，处理器可以同时执行多个指令，如算术运算、逻辑运算和内存访问，以提高处理器的效率。

任务级并行性： 涉及同时执行多个任务或代码流的能力。这种并行性通常在多核处理器或分布式系统中实现，允许不同的处理单元同时执行不同的任务或操作，以提高整体性能。例如，一个应用程序可以被分解成多个并行执行的任务，每个任务在不同的处理器核心上执行，从而加速应用程序的执行。

--------------------------------------------------------------------------------------------------------------------------------

1. Параллелизм уровня инструкций. Суперскалярные процессора. Особенности и принципы работы. Сравнение с VLIW и практика использования. Достоинства и недостатки. Барьеры памяти.

指令级并行性。 超标量处理器。 特点及工作原理。 与VLIW的比较及使用实践。 的优点和缺点。 记忆障碍。

指令级并行性（ILP）是一种计算机处理器的并行计算模式，它允许多条指令在同一时间段内同时执行。

超标量处理器是一种实现指令级并行性的处理器- 具有多个并行运行的 ALU 的处理器（我们同时处理多个标量）

·特点：

能够同时执行多条指令，提高了处理器的性能和效率。

包含多个功能单元，允许并行执行不同类型的指令。

采用动态调度和乱序执行技术，以最大程度地利用处理器资源。

·工作原理：

处理器通过识别和提取指令级别的并行性，将可并行执行的指令同时发送到不同的执行单元。

指令执行过程中可能存在依赖关系，因此处理器需要执行乱序调度和数据相关性检测以确保正确执行指令。

与VLIW的比较：

超标量处理器在运行时动态调度和执行指令，而VLIW处理器在编译时静态生成长指令字，因此超标量处理器更灵活。

VLIW需要在编译时进行指令调度和资源分配，这可能导致编译器复杂度增加，并且不够灵活适应不同的应用场景。

超标量处理器通常具有更好的性能和更广泛的应用，因为它们能够在运行时根据实际情况调整指令执行顺序和资源分配。

·优点：

高性能： 超标量处理器能够同时执行多条指令，充分利用处理器资源，从而提高了处理器的性能和吞吐量。

灵活性： 它能够根据当前执行的指令和资源情况，动态调度和执行指令，适应不同的应用场景，提高了处理器的灵活性和适应性。

增强的指令级并行性（ILP）： 超标量处理器能够充分利用指令级并行性，同时执行多条指令，减少了指令之间的等待时间，提高了指令执行效率。

较低的时钟周期： 由于能够并行执行多条指令，超标量处理器通常具有较低的时钟周期，从而提高了处理器的运行速度。

·缺点：

复杂性和成本： 超标量处理器需要复杂的硬件支持，包括乱序执行、动态调度、数据相关性检测等功能单元，因此处理器的设计、制造和维护成本较高。

资源竞争和数据相关性： 同时执行多条指令可能导致资源竞争和数据相关性问题，需要额外的硬件支持来解决这些问题，增加了处理器的复杂度和功耗。

能效比降低： 由于需要较多的硬件资源支持同时执行多条指令，超标量处理器的能效比（性能与功耗的比值）可能相对较低，特别是在负载较轻的情况下。

指令调度开销： 动态调度和乱序执行可能会引入额外的指令调度开销，导致处理器性能的波动和不稳定性。

**内存屏障**是一种指令，命令编译器和处理器在内存访问之前和之后确立严格的顺序关系（在内存屏障之前的所有内存访问都将在内存屏障之后的第一个内存访问之前完成）。

解决的问题：处理器可能会改变指令执行的顺序，导致意外行为。内存屏障解决了这些问题。

--------------------------------------------------------------------------------------------------------------------------------

1. Параллелизм уровня инструкций. VLIW процессора. Особенности и принципы работы. Сравнение с суперскалярными и практика использования. Достоинства и недостатки. Барьеры памяти.

指令级并行性。 处理器的 VLIW。 特点及工作原理。 与超标量的比较和使用实践。 的优点和缺点。 记忆障碍。

指令级并行性（ILP）是一种计算机处理器的并行计算模式，它允许多条指令在同一时间段内同时执行。

VLIW（Very Long Instruction Word）是一种处理器，通过创建包含多个操作的大型指令来实现并行性。这些指令在进入处理器时被硬件分解为多个ALU。

特点：

长字指令: VLIW处理器使用长指令字，每个指令字包含多个操作。

硬件并行性: 每个指令字中的操作可以并行执行，这允许处理器同时执行多个操作。

编译器依赖: VLIW处理器的性能高度依赖于编译器的能力，编译器负责生成有效的指令序列以利用硬件并行性。

工作原理：

指令级并行性: VLIW处理器在单个指令中执行多个操作，这些操作可以并行执行。

编译器优化: 编译器负责识别和组织适当的操作序列，并将它们打包到长指令字中，以便最大限度地利用处理器的并行性。

硬件执行: 处理器在执行指令时，根据指令字中的操作将其发送到相应的功能单元，从而同时执行多个操作。

无需依赖动态调度: 与超标量处理器不同，VLIW处理器在硬件级别上执行并行性，因此无需动态调度单元来确定指令的执行顺序。

超标量处理器更适用于通用系统，因为对于开发人员来说，并行处理的过程是透明的，并且可以根据情况进行调整。

VLIW 则更适用于特殊应用，常用于嵌入式系统中。这是因为可以编写高效的专用编译器。

优点：

并行性明显：VLIW 指令中包含多个操作，因此可以在同一时钟周期内执行多个操作，提高了处理器的效率和性能。

简化硬件设计：由于 VLIW 指令在编译时已经确定了执行顺序，因此处理器硬件设计相对简单，可以更容易地实现高性能。

适用于特定应用：VLIW 处理器适用于一些特定的嵌入式系统和DSP应用，可以通过专门的编译器优化指令流程，提高性能。

缺点：

高要求的编译器：为了充分利用 VLIW 处理器的并行性能，需要专门优化的编译器来生成有效的指令流。这增加了软件开发的复杂性和成本。

静态调度限制：VLIW 指令的并行性在编译时已经确定，因此无法根据运行时的情况进行动态调度和优化，导致可能存在指令浪费的情况。

难以适应通用应用：由于 VLIW 处理器对编译器的要求较高，难以处理复杂的通用应用程序，因此在通用计算领域的应用相对有限。

**内存屏障**是一种指令，命令编译器和处理器在内存访问之前和之后确立严格的顺序关系（在内存屏障之前的所有内存访问都将在内存屏障之后的第一个内存访问之前完成）。

解决的问题：处理器可能会改变指令执行的顺序，导致意外行为。内存屏障解决了这些问题。

--------------------------------------------------------------------------------------------------------------------------------

1. Программируемые логические контроллеры (ПЛК). Область применения. Особенности аппаратуры и программирования по сравнению с компьютерами общего назначения.

可编程逻辑控制器（PLC）。 应用领域。 与通用计算机相比的硬件和编程特点。

程序可编程逻辑控制器（PLC）是一种专用于工业自动化领域的计算设备（最初是为了简化工厂流水线而创建的）。它们通常用于监控和控制生产过程中的机器和设备，例如在制造、工厂和生产线等环境中。PLC的主要功能是根据预先定义的逻辑和条件执行特定的操作，例如打开或关闭阀门、启动或停止电机等。

特点：

硬件特性：PLC通常采用工业级组件和模块化设计，具有较强的抗干扰能力和耐用性，适应恶劣的工业环境。与通用计算机相比，PLC的输入输出（I/O）接口更多样化，可以连接各种传感器和执行器。模块化设计，允许根据具体的应用需求进行灵活拓展和配置

编程特点：

实时性要求：PLC需要及时响应现场信号并执行控制操作，因此具有较高的实时性要求。它们通常采用实时操作系统和硬件时钟来确保任务的及时执行。

编程语言：PLC的编程通常使用专门的图形化编程语言，如梯形图（Ladder Diagram）和功能块图（Function Block Diagram）。这些语言更直观易懂，适合工程师和技术人员进行逻辑设计和编程。

可靠性和安全性：由于PLC用于控制生产过程中的设备和机器，因此其稳定性、可靠性和安全性至关重要。PLC通常采用冗余设计和安全控制功能，以防止故障和意外事故的发生。

总的来说，PLC在工业自动化领域具有广泛的应用，并且与通用计算机相比，具有更强的适应性和可靠性，能够满足工业生产环境的特殊需求。

--------------------------------------------------------------------------------------------------------------------------------

1. Программируемые логические интегральные схемы (ПЛИС). Область применения. Особенности. Устройство. Задачи высокоуровневого синтеза, достоинства и недостатки.

可编程逻辑集成电路（FPGA）。 应用领域。 特点。 设备。 高级综合任务、优点和缺点。

可编程逻辑集成电路（PLD）是一种可编程的集成电路，可在电路设计中用于实现特定的逻辑功能。它们通常被用于数字电路设计，以替代传统的硬连线逻辑电路，提供更灵活的设计和开发过程。

PLD的应用领域包括但不限于：

数字逻辑电路设计

控制系统

信号处理

通信系统

图像处理

计算机体系结构

PLD的主要特点包括：

可编程性：PLD的逻辑功能可以通过编程来实现，而不需要物理硬连线。

灵活性：由于可编程性，PLD可以用于多种不同的应用，且能够根据需求进行灵活的修改和重新编程。

集成度高：PLD集成了大量的逻辑门和存储器单元，可以实现复杂的逻辑功能，并且通常具有较小的体积和封装。

快速原型开发：PLD可用于快速原型开发，可以在短时间内实现和测试新的电路设计。

PLD的主要组成部分包括可编程逻辑阵列（PLA）和可编程互连网络（Interconnect Network）输入/输出接口（I/O Interface）配置存储器（Configuration Memory）时钟管理单元（Clock Management Unit）。通过编程软件（如HDL，硬件描述语言）或专用设计工具进行设计和编程。

高级综合（HLS）是一种将高级编程语言（如C/C++）转换为硬件描述语言（如Verilog或VHDL）的过程，使得开发人员可以使用高级编程语言来描述硬件功能，从而提高设计效率和降低开发成本。HLS的优点包括：提高开发效率、减少设计复杂度、缩短上市时间、降低成本。然而，HLS也存在一些缺点，如：对设计人员的要求较高、生成的硬件性能可能不如手动优化、某些设计场景下生成的硬件可能不够优化。

--------------------------------------------------------------------------------------------------------------------------------

1. Классификация Флинна. Выделяемые классы и примеры машин этих классов. SIMT архитектура.

弗林分类。 杰出的类别和这些类别的机器示例。 SIMT架构。

并行处理器的分类基于指令线程数和数据流的数目

仅适用于冯·诺依曼处理器，它甚至在任何其他奇异处理器出现之前就被创建了

优点：简单易懂

缺点：不适用于非冯诺依曼

重新加载MIMD类

弗林分类是针对计算机系统的分类方法，根据它们能够同时处理的指令和数据流的数量将其分为几个类别。主要的弗林分类包括：

SISD（单指令流，单数据流）：这是经典的单流程计算机，一次只处理一条指令和一组数据。例如传统的单处理器计算机。

SIMD（单指令流，多数据流）：在这些系统中，一条指令同时作用于多组数据。例如具有向量计算的多处理器系统，如图形处理单元（GPU）。

MISD（多指令流，单数据流）：在这个类别中，多条指令同时处理相同的数据。MISD架构很少见，主要用于冗余控制系统。例如某些航空控制系统。

MIMD（多指令流，多数据流）：这些计算机同时处理多组数据，使用多个指令流。例如计算节点集群或多核处理器，其中每个核心可以独立执行自己的程序。

SIMT（单指令，多线程）架构是NVIDIA GPU引入的一种SIMD变体，其中多个执行线程同时处理相同的指令，但使用不同的数据集。这使得能够有效利用GPU资源进行并行数据处理。

--------------------------------------------------------------------------------------------------------------------------------

1. Классификация Дункана. Цели и задачи, классификация первого уровня. Синхронные архитектуры (векторные, SIMD, ассоциативные массивы, систолические). Принципы их работы и примеры.

邓肯的分类。 目的和目标，一级分类。 同步架构（矢量、SIMD、关联数组、脉动）。 他们的工作原理和例子。

邓肯的分类是一种用于描述并行计算机架构的系统分类方法。它根据系统的结构和功能将并行计算机架构分为几个层次。

第一级的分类包括以下几种同步架构：

向量架构：向量架构是一种并行计算机架构，它通过使用向量寄存器和向量指令来执行相同的操作，以同时处理大量数据。这种架构适用于科学和工程领域中需要大规模数据并行处理的应用，如数值计算和模拟。

SIMD架构：SIMD（Single Instruction, Multiple Data）架构是一种并行计算机架构，它通过在多个处理单元上同时执行相同的指令来实现数据并行处理。每个处理单元都执行相同的操作，但操作的数据可以不同。SIMD架构通常用于图形处理、多媒体处理和向量化计算等领域。

关联数组架构：关联数组架构是一种并行计算机架构，它通过将计算任务分配到多个处理单元，并使用关联内存来存储和访问数据，以实现并行处理。这种架构适用于需要高度并行化和大规模数据共享的应用，如数据库查询和数据挖掘。

Systolic架构：Systolic架构是一种并行计算机架构，它通过使用多个处理单元和数据流水线来执行计算任务，并将计算结果传递给下一个处理单元，以实现高效的并行计算。Systolic架构通常用于数字信号处理、神经网络和矩阵计算等领域。

这些同步架构在并行计算中起着重要作用，每种架构都有其独特的特点和适用场景。选择合适的架构取决于应用的需求、数据的特性和系统的性能要求。

--------------------------------------------------------------------------------------------------------------------------------

1. Классификация Дункана. Цели и задачи, классификация первого уровня. MIMD архитектуры (распределённая и разделяемая память). Примеры и принципы их работы.

邓肯的分类。 目的和目标，一级分类。 MIMD 架构（分布式和共享内存）。 他们的工作示例和原则。

邓肯的分类是一种用于描述并行计算机架构的系统分类方法。它根据系统的结构和功能将并行计算机架构分为几个层次。

在第一级的分类中，我们有MIMD（Multiple Instruction, Multiple Data）架构，它涵盖了两种主要类型的架构：分布式内存和共享内存。

分布式内存架构：在分布式内存架构中，多个处理单元通过网络连接，每个处理单元都有自己的本地内存，并且处理单元之间的通信通过消息传递来实现。每个处理单元可以独立执行不同的指令，并且可以访问分布式系统中的不同内存位置。这种架构适用于需要大规模数据并行处理和高度灵活性的应用，例如科学模拟、气候建模和大数据分析。

共享内存架构：在共享内存架构中，多个处理单元共享同一块内存，并且可以通过读写内存来进行通信。每个处理单元可以同时访问共享内存中的数据，并且可以通过同步机制来保证数据的一致性和完整性。这种架构适用于需要更紧密的协作和数据共享的应用，例如多线程程序、数据库系统和并行计算任务。

这些MIMD架构在并行计算中具有重要作用，它们提供了不同的方式来处理并行任务，并且可以根据应用的需求选择最合适的架构类型。通过合理选择架构，可以充分利用系统资源，提高计算效率和性能。

--------------------------------------------------------------------------------------------------------------------------------

1. Классификация Дункана. Цели и задачи, классификация первого уровня. MIMD парадигмы (MIMD/SIMD, потоки данных, редукционные, wavefront). Примеры и принципы их работы.

邓肯的分类。 目的和目标，一级分类。 MIMD 范例（MIMD/SIMD、数据流、缩减、波前）。 他们的工作示例和原则。

邓肯分类法用于描述并行计算机系统的架构和工作方式，其中第一级分类包括了MIMD（Multiple Instruction, Multiple Data）范式，它又可以进一步分为几种子范式：

MIMD/SIMD：这种范式结合了MIMD和SIMD（Single Instruction, Multiple Data）的特性。在这种范式下，系统中有多个处理单元（MIMD），每个处理单元可以执行不同的指令，但同时也可以进行向量化操作（SIMD），即一次执行多个数据元素上的相同操作。这种范式常见于现代并行处理器中，例如GPU（图形处理器）和某些多核处理器。通过MIMD/SIMD范式，系统可以实现更高效的并行计算，利用了MIMD的灵活性和SIMD的并行性。

数据流：数据流范式中，计算是由数据流的传输和处理来驱动的，而不是像传统的命令式计算那样按照严格的指令序列执行。每个操作都会等待其输入数据可用，并且一旦数据就绪就立即执行操作。数据流范式适用于流式数据处理、图像处理和信号处理等应用领域，其中数据的产生和处理是持续不断的。

Reduction：这种范式涉及到对数据进行归约操作，即将多个数据元素合并成一个单一的结果。这种操作通常涉及到对数组、列表或其他数据结构中的元素进行聚合操作，例如求和、求平均值等。在并行计算中，归约操作通常用于并行计算任务的结果汇总阶段，以便生成最终的结果。

Wavefront：Wavefront范式是一种并行计算模型，其中数据流沿着处理单元形成波前传播。每个处理单元在收到数据后执行计算，并将结果传递给下一个处理单元。这种范式适用于那些具有逐步计算过程的问题，例如图像处理中的滤波操作或地图处理中的地形平滑。

这些MIMD范式提供了不同的方式来进行并行计算，适用于不同类型的应用和算法。选择合适的范式可以帮助优化并行计算任务的执行效率，并充分利用系统资源。

PPA：

1. Spraying атака. Суть. Виды. Методы противодействия.

喷射攻击。 底线。 种类。 对策。

1. Shell-скрипты. Недостатки. Почему до сих пор используются. Возможные альтернативы.

外壳脚本。 缺陷。 为什么它们仍然被使用？ 可能的替代方案。

1. Принципы работы и уязвимости GSM-сетей, методы защиты.

GSM网络的工作原理、漏洞、保护方法。

1. Уязвимость Heartbleed. Какие ещё вызовы и угрозы безопасности существуют в современной веб-разработке, как защитить свои веб-приложения от возможных атак.

心血漏洞。 现代 Web 开发中还存在哪些其他安全挑战和威胁，以及如何保护您的 Web 应用程序免受可能的攻击。

Heartbleed是一种安全漏洞，影响了使用OpenSSL加密库的许多Web服务器。这个漏洞允许攻击者从受影响的服务器内存中读取敏感信息，例如加密密钥、用户身份验证凭据等，而不需要任何有效的凭据。

原理：攻击者可以发送一个特制的心跳请求，其中包含一个不正确的数据长度值，导致服务器返回的响应中包含了额外的内存内容。攻击者可以通过多次发送这样的恶意请求，并收集服务器返回的内存信息，从而逐步泄露服务器内存中的敏感信息，如加密密钥、用户凭据等。

跨站脚本攻击（XSS）是一种常见的网络安全漏洞，攻击者利用这种漏洞在受害者的浏览器中执行恶意脚本。这些脚本可以窃取用户的会话令牌、cookie、密码或其他敏感信息，或者以受害者身份执行未经授权的操作。

保护措施：

HTTPS 加密：使用 HTTPS 协议加密传输数据，以防止恶意脚本在传输过程中被窃取或篡改。

定期漏洞扫描和安全审计：定期对 Web 应用程序进行漏洞扫描和安全审计，及时发现和修补潜在的漏洞。

输入验证和过滤：对用户输入数据进行验证和过滤，移除或转义特殊字符和标记，以防止恶意脚本的注入。

1. Как работает Garbage Collector. Проблемы. Инкрементальный Garbage Collector.

垃圾收集器的工作原理。 问题。 增量垃圾收集器。

垃圾收集器（Garbage Collector）是一种自动内存管理机制，用于在程序执行过程中自动识别和回收不再被程序使用的内存，以防止内存泄漏和提高内存利用率。

工作原理：垃圾收集器通过识别和标记不再被程序使用的内存对象，并将其释放以供重新使用。通常，它会跟踪程序中所有的引用，从根对象开始，逐步遍历整个对象图，标记所有可以访问到的对象，然后清除未标记的对象，释放它们所占用的内存。

问题：

·性能开销：垃圾收集器的运行会消耗系统资源，可能会影响程序的性能。

·内存碎片：由于垃圾收集器通常使用分代收集策略，释放的内存可能会产生碎片，降低内存的连续性。

·停顿时间：垃圾收集过程中，程序可能会暂停执行，造成停顿时间，影响用户体验

增量式垃圾收集器：为了解决停顿时间的问题，增量式垃圾收集器将垃圾收集过程分解成多个阶段，在程序执行过程中逐步完成垃圾收集工作，从而减少单次垃圾收集造成的停顿时间。

增量式垃圾收集器可以在程序执行的空闲时间进行垃圾收集，或者通过将垃圾收集工作分解成多个阶段，在程序执行过程中交替执行垃圾收集和程序代码，从而减少停顿时间的影响。

1. Что такое FSD (полностью самостоятельное вождение). Технические средства. Существующие проблемы.

什么是FSD（全自动驾驶）。 技术手段。 现有问题。

1. Блокчейн. Устройство и уязвимости.

区块链。 设备和漏洞。

区块链是一种去中心化的分布式数据库技术，用于记录交易和数据的不断增长的链式数据结构。

·区块（Block）：每个区块包含一定数量的交易数据

·链式结构（Chain）：区块以链式结构相互链接，从而形成一个连续的、不可篡改的链条

·节点（Node）：节点是参与区块链网络的计算机设备

·共识机制（Consensus Mechanism）：共识机制是区块链网络中确保所有节点达成一致的算法或规则

·交易（Transaction）：交易是在区块链上进行的数据传输操作

漏洞：1. 51%攻击：当一个实体控制了超过区块链网络总算力的51%时，可以控制网络的交易记录，造成双重花费等问题。

1. 隐私泄露：区块链交易的公开性可能导致个人隐私泄露和信息泄露风险。
2. GPS и ГЛОНАСС: Принцип работы, уязвимости, средства защиты.

GPS 和 GLONASS：工作原理、漏洞、保护手段。

GPS和GLONASS都是全球定位卫星系统。

GPS由美国国防部维护，而GLONASS由俄罗斯维护。

工作原理：

·GPS由美国国防部维护，它由一组24颗卫星组成，分布在地球中等轨道上。每颗GPS卫星都持续地向地面发送时间和位置信息的信号。接收器通过接收来自至少四颗卫星的信号，并测量信号的传播时间来计算自身的位置、速度和时间。

·GLONASS由俄罗斯维护，也由一组24颗卫星组成，分布在地球的三个轨道平面上。每颗GLONASS卫星持续地向地面发送时间和位置信息的信号，接收器通过接收来自至少四颗卫星的信号来确定位置、速度和时间。

漏洞：GPS系统容易受到以下攻击：

信号干扰和伪造：攻击者可以通过发送干扰信号或伪造卫星信号来干扰GPS接收器，导致误导用户或使其无法正常工作。

空间信号阻塞：高楼、山脉、天气等因素可能会阻塞GPS信号的传播，影响定位的准确性和可用性。

保护手段：为了保护GPS系统，通常采取以下措施：

加密和数字签名：使用加密技术和数字签名来验证信号的真实性，防止信号的干扰和伪造。

备用系统：使用多个卫星系统的组合，如GPS和GLONASS，以增强定位的可靠性。

1. Каковы основные принципы и методики, которые позволяют обнаруживать и анализировать ошибки в компиляторах? Приведите 2 способа и их краткое описание.

允许我们检测和分析编译器中的错误的基本原理和技术是什么？ 给出2种方法及其简要说明。

1. Почему программисту важно знать о текущих абстракциях и думать на уровень ниже, чем он работает. Примеры, процессор Skylake и параллельное программирование.

为什么对于程序员来说了解当前的抽象并在低于他正在工作的水平上思考很重要？ 示例：Skylake 处理器和并行编程。

1. Вирусы-вымогатели. Суть, механизмы обеспечения незаметности и шифрования. Защита.

勒索软件病毒。 本质是确保不可见性和加密的机制。 保护。

1. Технические средства и принципы фильтрации интернет-трафика. Способы обхода.

过滤互联网流量的技术手段和原理。 解决方法。

1. Суть обработки естественного языка(NLP). Основные этапы. Проблемы. Полисемия.

自然语言处理（NLP）的本质。 主要阶段。 问题。 一词多义。