Министерство образования и науки Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности
Высшая школа технологий искусственного интеллекта
Направление: 02.03.01 «Математика и компьютерные науки»

Вопросы к зачёту

Санкт-Петербург
2025

Вопрос 1. В каком году была построена первая вычислительная машина?

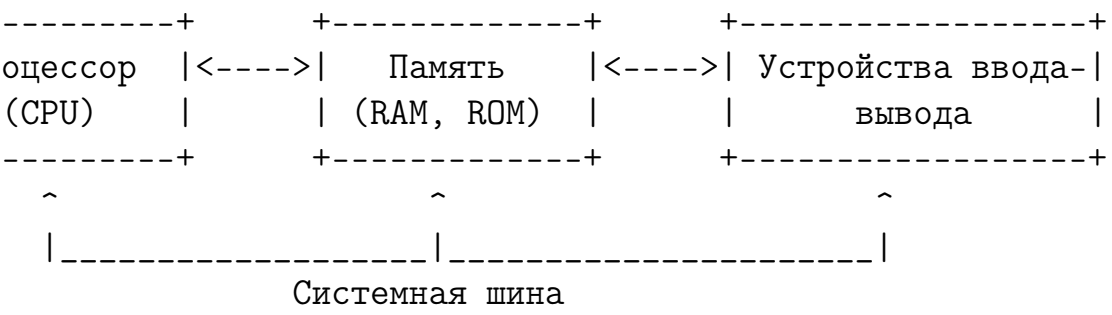
Первая электронная вычислительная машина ENIAC была построена в **1946 году** в США (Пенсильванский университет). Она содержала около 18 000 электронных ламп, весила 30 тонн и выполняла около 5000 операций сложения в секунду.

Вопрос 2. В каком году и кем была построена первая отечественная вычислительная машина?

Первая отечественная ЭВМ — МЭСМ (Малая Электронная Счётная Машина) — была создана в **1950 году** под руководством академика **Сергея Алексеевича Лебедева** в Киеве. В 1952 году под его же руководством была создана БЭСМ — на тот момент самая быстрая ЭВМ в Европе.

Вопрос 3. Нарисуйте простейшую структурную схему вычислительной машины.

Классическая архитектура фон Неймана:



Основные принципы: программа и данные хранятся в общей памяти, команды выполняются последовательно, всё представлено в двоичном коде.

Вопрос 4. Машина с радиальной архитектурой — это...

Архитектура, в которой каждое периферийное устройство подключается к процессору по **отдельной, выделенной линии связи**. Преимущество: высокая пропускная способность, нет конфликтов за общую шину. Недостаток: сложность, дороговизна, плохая масштабируемость.

Вопрос 5. Магистраль — это...

Магистраль (системная шина) — совокупность линий, обеспечивающих передачу информации между компонентами ЭВМ. Состоит из трёх частей:

- **Шина данных** — передача данных (двунаправленная)
- **Шина адреса** — передача адресов (однонаправленная от CPU)
- **Шина управления** — сигналы чтения, записи, прерываний

Вопрос 6. Северный и южный мосты. Назначение.

Северный мост (MCH — Memory Controller Hub):

- Связывает процессор с быстрыми устройствами
- Управляет оперативной памятью
- Обеспечивает связь с видеокартой (AGP, PCI-E x16)

Южный мост (ICH — I/O Controller Hub):

- Связывает северный мост с медленными устройствами
- Управляет USB, SATA, PCI, аудио, сетью
- Содержит контроллер прерываний, BIOS

Вопрос 7. Отличительные особенности Hub-архитектуры.

Hub-архитектура заменяет общую шину на **выделенные каналы (point-to-point)** между компонентами. Каждая подсистема имеет собственный канал к хабу. Преимущества: устранение конфликтов за шину, увеличение пропускной способности, возможность параллельной работы устройств.

Вопрос 8. Пропускная способность шины — это...

Максимальный объём данных, передаваемый по шине за единицу времени. Формула:

$$\text{ПС} = \frac{\text{Разрядность (бит)}}{8} \times \text{Частота (Гц)} \times \text{Передач за такт}$$

Пример: 64 бита, 100 МГц, DDR (2 передачи): $\frac{64}{8} \times 100 \times 10^6 \times 2 = 1600$ МБ/с.

Вопрос 9. Контроль по чётности, привести пример.

Метод обнаружения ошибок: к данным добавляется бит так, чтобы общее число единиц было чётным (even parity) или нечётным (odd parity).

Пример (even parity):

- Данные: `1010110` (4 единицы — чётное)
- Бит чётности: `0`
- Передаётся: `10101100`

Ограничение: обнаруживает только нечётное число ошибок, не исправляет.

Вопрос 10. Шинный цикл состоит из... (Нарисовать)

Шинный цикл — последовательность действий для одной операции обмена:

```
   T1         T2         T3         T4         Tw
   |          |          |          |          |
Выставле-  Подго-    Передача  Заверше-   Такты
ние адреса товка     данных    ние цикла  ожидания
```

- **T1** — процессор выставляет адрес
- **T2** — устройство готовится к обмену
- **T3** — чтение или запись данных
- **T4** — завершение, снятие сигналов
- **Tw** — такты ожидания для медленных устройств

Вопрос 11. Прерывание — это...

Прерывание — механизм, позволяющий приостановить выполнение текущей программы для обработки некоторого события (внешнего или внутреннего) с последующим возвратом к прерванной программе. Обеспечивает реакцию системы на асинхронные события.

Вопрос 12. Стек — это...

Стек — область памяти, организованная по принципу LIFO (Last In — First Out): последний записанный элемент извлекается первым. Используется для хранения адресов возврата, параметров функций, локальных переменных.

Вопрос 13. Принцип работы стека.

В архитектуре x86 стек растёт в сторону **младших адресов**:

- При записи (PUSH): сначала SP уменьшается, затем данные записываются по адресу [SP]
- При чтении (POP): данные читаются из [SP], затем SP увеличивается

Регистр SP (Stack Pointer) всегда указывает на последний записанный элемент.

Вопрос 14. Пример записи и чтения из стека.

```
MOV AX, 1234h       ; AX = 1234h
MOV BX, 5678h       ; BX = 5678h

PUSH AX             ; SP = SP - 2, [SP] = 1234h
PUSH BX             ; SP = SP - 2, [SP] = 5678h

POP CX              ; CX = [SP] = 5678h, SP = SP + 2
POP DX              ; DX = [SP] = 1234h, SP = SP + 2

; Результат: CX = 5678h, DX = 1234h (обратный порядок!)
```

Вопрос 15. Команды, используемые при обращении к стеку.

`PUSH src`	Запись в стек: SP = SP - 2 (или 4), [SP] = src
`POP dst`	Чтение из стека: dst = [SP], SP = SP + 2 (или 4)
`PUSHA/PUSHAD`	Сохранить все регистры общего назначения
`POPA/POPAD`	Восстановить все регистры общего назначения
`PUSHF/PUSHFD`	Сохранить регистр флагов
`POPF/POPFD`	Восстановить регистр флагов

Вопрос 16. Команды, используемые при вызове и возврате из подпрограммы.

`CALL addr`	Вызов подпрограммы: сохраняет адрес следующей команды в стек (PUSH IP), затем переходит по адресу addr
`RET`	Возврат: извлекает адрес из стека (POP IP) и передаёт управление
`RET n`	Возврат с очисткой стека: после возврата SP увеличивается на n байт

Вопрос 17. Типы прерываний (перечислите и опишите назначение).

1. **Аппаратные (внешние)** — генерируются периферийными устройствами (клавиатура, таймер, диск). Асинхронны по отношению к программе. Сигнализируют о событиях, требующих внимания.

2. **Программные** — вызываются командой `INT n`. Синхронны. Используются для системных вызовов ОС и сервисов BIOS.

3. **Исключения (exceptions)** — возникают при ошибках выполнения команд: деление на ноль, обращение к несуществующей странице (page fault), нарушение защиты.

Вопрос 18. Какие прерывания могут быть маскируемыми, а какие нет.

Маскируемые:

- Аппаратные прерывания, приходящие по линии INTR
- Можно запретить сбросом флага IF

Немаскируемые (NMI):

- Приходят по отдельной линии NMI
- Критические события: ошибка памяти, сбой питания
- Исключения процессора
- Программные прерывания INT n

Вопрос 19. Как происходит маскирование прерываний.

Два способа:

1. **Глобальное** — через флаг IF в регистре флагов:
   - `CLI` — сброс IF (IF=0), запрет всех маскируемых прерываний
   - `STI` — установка IF (IF=1), разрешение прерываний
2. **Индивидуальное** — через контроллер прерываний:
   - Запись маски в регистр IMR контроллера 8259A или APIC
   - Позволяет запретить отдельные линии IRQ

Вопрос 20. По каким ситуациям происходит вызов программных прерываний.

- **Системные вызовы ОС** — INT 21h (DOS), INT 80h (Linux)
- **Сервисы BIOS** — INT 10h (видео), INT 13h (диск), INT 16h (клавиатура)
- **Отладка** — INT 3 (точка останова), INT 1 (пошаговый режим)
- **Обработка ошибок** — INT 0 (деление на ноль)

Вопрос 21. Перечислите последовательность событий при входе процессора в прерывание.

1. Завершение выполнения текущей команды
2. Сохранение регистра флагов в стек (PUSH FLAGS)
3. Сброс флагов IF и TF (запрет прерываний и трассировки)
4. Сохранение CS в стек (PUSH CS)
5. Сохранение IP в стек (PUSH IP)
6. Чтение адреса обработчика из таблицы векторов
7. Загрузка нового CS:IP и передача управления обработчику

Вопрос 22. Обработчик прерывания — это...

Обработчик прерывания (ISR — Interrupt Service Routine) — подпрограмма, выполняющая действия по обслуживанию прерывания. Должен сохранить используемые регистры, выполнить обработку события, послать подтверждение контроллеру (EOI) и завершиться командой IRET.

Вопрос 23. Где располагается обработчик прерывания.

Обработчик располагается в **оперативной памяти**. Его адрес хранится в таблице векторов прерываний. Обработчики могут быть частью BIOS, операционной системы или пользовательской программы.

Вопрос 24. Вектор прерывания — это...

Вектор прерывания — адрес обработчика прерывания. В реальном режиме x86 это 4 байта: 2 байта смещения (offset) + 2 байта сегмента (segment). Номер прерывания (0-255) служит индексом в таблице векторов.

Вопрос 25. Таблица векторов прерываний — это...

Таблица векторов прерываний (IVT) — массив адресов обработчиков для всех 256 возможных прерываний. Размер: $256 \times 4 = 1024$ байта. По номеру прерывания n адрес обработчика находится по смещению n $\times$ 4.

Вопрос 26. Таблица векторов прерываний находится...

- **Реальный режим x86:** по адресу `0000:0000` (первый килобайт памяти)
- **Защищённый режим:** расположение определяется регистром IDTR (Interrupt Descriptor Table Register), таблица называется IDT

Вопрос 27. Обслуживание прерываний бывает (перечислите, поясните).

1. **Последовательное (невложенное):**
   - Во время обработки прерывания другие прерывания запрещены (IF=0)
   - Новые запросы ждут завершения текущего обработчика
   - Простота, но возможны задержки для критичных событий

2. **Вложенное (nested):**
    - Более приоритетное прерывание может прервать обработчик
    - Обработчик должен разрешить прерывания (STI) и сохранить контекст
    - Быстрая реакция на критические события

Вопрос 28. Какой бывает приоритизация прерываний (перечислите, поясните).

1. **Фиксированная:** приоритеты жёстко заданы (чем меньше номер IRQ, тем выше приоритет). Просто, но негибко.
2. **Циклическая (ротируемая):** после обслуживания устройства его приоритет становится наименьшим. Равномерное распределение времени.
3. **Программируемая:** приоритеты задаются программно через регистры контроллера. Максимальная гибкость.

Вопрос 29. Контроллер прерываний. Сколько линий в многопроцессорных системах?

8259A PIC: 8 линий на контроллер, каскад из двух = 15 линий (одна для каскадирования).

APIC (Advanced PIC): I/O APIC имеет **24 линии прерываний**. Каждый процессор имеет Local APIC. Поддерживает межпроцессорные прерывания (IPI) и MSI.

Вопрос 30. Как центральный процессор узнаёт об аппаратном прерывании?

Контроллер прерываний выставляет сигнал на линии **INTR** (для маскируемых) или **NMI** (для немаскируемых). Процессор проверяет эти линии между выполнением команд. Если IF=1 и есть сигнал на INTR, процессор начинает цикл подтверждения.

Вопрос 31. Как устройства ввода-вывода узнают о подтверждении запроса на прерывание.

Процессор генерирует два импульса сигнала **INTA (Interrupt Acknowledge)**:

1. Первый импульс — уведомляет контроллер о начале обработки
2. Второй импульс — запрашивает номер вектора прерывания

Контроллер выставляет номер вектора на шину данных.

Вопрос 32. Системные ресурсы — это...

Аппаратные ресурсы, выделяемые устройствам для взаимодействия с системой:

- **Адреса портов ввода-вывода** (I/O ports)
- **Номера прерываний** (IRQ)
- **Каналы DMA** (прямой доступ к памяти)
- **Области памяти** (memory ranges)

Вопрос 33. Адреса портов ввода-вывода делятся на 2 группы. По какому принципу?

1. **Фиксированные** — для стандартных устройств, адреса определены стандартом:
   - Клавиатура — 60h
   - Таймер — 40h-43h
   - COM1 — 3F8h
2. **Назначаемые (конфигурируемые)** — для устройств PCI, USB, PnP. Адреса назначаются BIOS или ОС при инициализации.

Вопрос 34. Plug and Play — это... (в чём его необходимость)?

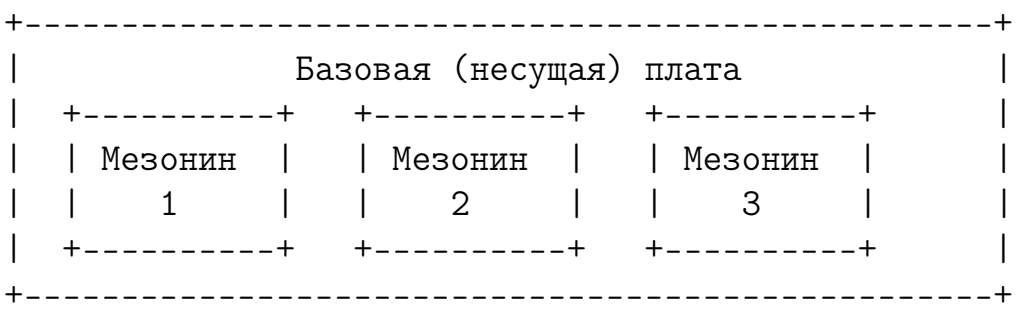
Plug and Play (PnP) — технология автоматического обнаружения и конфигурирования устройств без ручного задания ресурсов.

Необходимость:

- Устранение конфликтов ресурсов (два устройства с одним IRQ)
- Упрощение установки оборудования для пользователя
- Автоматическое определение оптимальных настроек
- Динамическое перераспределение ресурсов

Вопрос 35. Мезонинная архитектура — это... (приведите пример, рисунок)

Архитектура с использованием **дочерних плат (мезонинов)**, устанавливаемых на базовую (несущую) плату:



Примеры стандартов: PMC, XMC, FMC, AMC.

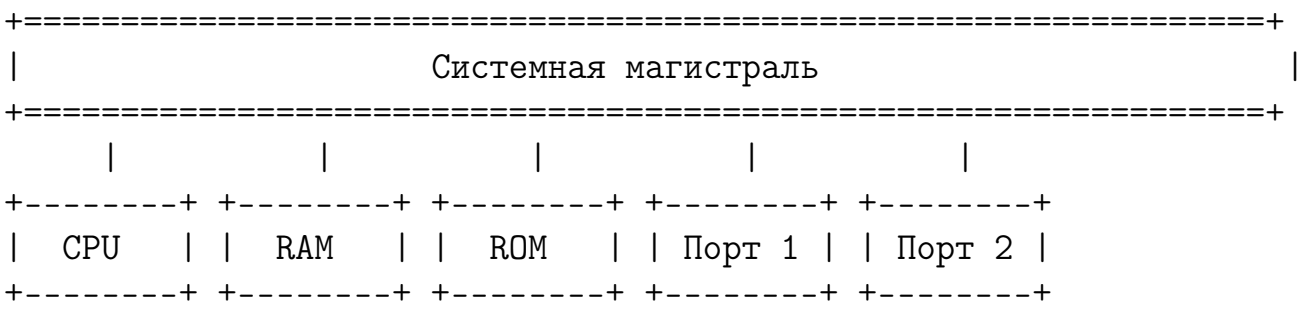
Вопрос 36. Шина PCI и аппаратные прерывания. Какие линии используются?

PCI использует **4 линии прерываний**: INTA#, INTB#, INTC#, INTD#.

Особенности:

- Линии разделяемые (shared) — несколько устройств на одной линии
- Уровневая сигнализация (level-triggered)
- Маршрутизируются через чипсет на IRQ
- Современные системы используют MSI (Message Signaled Interrupts)

Вопрос 37. Приведите пример структурной схемы ЭВМ с магистральным принципом построения.



Все устройства подключены к общей шине и обмениваются данными через неё. Арбитраж определяет, кто в данный момент использует шину.

Вопрос 38. При помощи каких команд происходит обращение к подпрограмме и возврат из неё.

- `CALL addr` — вызов: сохраняет IP (и CS для дальнего вызова) в стек, переходит к addr
- `RET` — возврат: извлекает IP из стека, передаёт управление
- `RET n` — возврат с очисткой n байт параметров из стека

Вопрос 39. В зависимости от источника возникновения сигнала прерывания делятся на...?

1. **Внешние (аппаратные)** — от периферийных устройств через линии INTR/NMI
2. **Внутренние (исключения)** — генерируются самим процессором при ошибках
3. **Программные** — вызываются командой INT n

Вопрос 40. Программируемый контроллер прерываний APIC. Чем отличается от каскада контроллеров прерываний?

Каскад 8259A	**APIC**
15 линий прерываний	24 линии (I/O APIC)
Только однопроцессорные системы	Поддержка многопроцессорности (SMP)
Фиксированная маршрутизация	Гибкая маршрутизация на любой CPU
Нет межпроцессорных прерываний	Поддержка IPI
Edge/level triggered	+ MSI (Message Signaled Interrupts)

Вопрос 41. Чем определяется (ограничивается) тактовая частота процессора?

- **Задержки переключения транзисторов** — физический предел скорости
- **Тепловыделение** — при высокой частоте выделяется больше тепла
- **Паразитные ёмкости и индуктивности** — искажают сигналы
- **Стабильность питания** — высокая частота требует качественного питания
- **Распространение тактового сигнала** — сложность синхронизации

Вопрос 42. В чём причина зависимости между частотой и потребляемой мощностью?

Динамическая мощность КМОП-схем:

$$P = C \cdot V^2 \cdot f$$

где C — паразитная ёмкость, V — напряжение, f — частота.

Причины пропорциональности:

- Мощность линейно зависит от частоты переключений
- При повышении частоты часто требуется повышение напряжения
- Токи утечки растут с уменьшением размеров транзисторов

Вопрос 43. Какие существуют способы оценки производительности цифрового компьютера?

- **Тактовая частота** — простой, но неточный показатель
- **MIPS** — миллионы команд в секунду
- **FLOPS** — операции с плавающей точкой в секунду (для научных вычислений)
- **Бенчмарки** — SPEC CPU, Geekbench, Cinebench
- **Время выполнения реальных задач** — наиболее объективный метод
- **CPI** — среднее число тактов на команду

Вопрос 44. Назовите три основных класса цифровых компьютеров и их особенности.

1. **Персональные компьютеры:**
   - Универсальные, для индивидуального использования
   - Баланс цены и производительности
   - Интерактивная работа пользователя
2. **Серверы:**
   - Высокая производительность и надёжность
   - Обслуживание множества клиентов
   - Отказоустойчивость, работа 24/7
3. **Встраиваемые системы:**
   - Специализированные под конкретную задачу
   - Ограниченные ресурсы, низкое энергопотребление
   - Работа в реальном времени

Вопрос 45. Назовите и охарактеризуйте основные части структуры простейшего цифрового компьютера.

1. **Процессор (CPU)** — выполняет команды, арифметические и логические операции, управляет работой системы
2. **Память** — хранит программы и данные (ОЗУ — оперативная, ПЗУ — постоянная)
3. **Устройства ввода-вывода** — обеспечивают взаимодействие с внешним миром (клавиатура, дисплей, диски)
4. **Системная шина** — обеспечивает связь между компонентами

Вопрос 46. В чём смысл понятия машинное слово?

Машинное слово — единица данных, которую процессор обрабатывает за одну операцию. Размер определяется архитектурой:

- 8-битные процессоры: слово = 8 бит
- 16-битные (8086): слово = 16 бит
- 32-битные (i386): слово = 32 бита
- 64-битные (x86-64): слово = 64 бита

Вопрос 47. В чём смысл понятия минимальная адресуемая единица (MAE)?

MAE — наименьший объём памяти, имеющий собственный уникальный адрес.

Возможные размеры:

- **1 байт (8 бит)** — наиболее распространённый (x86, ARM)
- **1 слово** — в некоторых DSP-процессорах
- **1 бит** — в специализированных системах

Вопрос 48. Что называют физическим адресом?

Физический адрес — реальный адрес ячейки в физической памяти, который выставляется на адресную шину. В системах с виртуальной памятью получается преобразованием виртуального адреса блоком MMU (Memory Management Unit).

Вопрос 49. Какие группы регистров выделяют в модели программиста?

1. **Регистры общего назначения (РОН)** — хранение данных и адресов
   x86: EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP

2. **Сегментные регистры** — адресация сегментов памяти
   x86: CS, DS, SS, ES, FS, GS

3. **Регистр состояния (флагов)** — результаты операций, управление
   x86: FLAGS/EFLAGS/RFLAGS

4. **Указатель команд** — адрес следующей команды
   x86: IP/EIP/RIP

Вопрос 50. В чём состоит назначение регистра состояния в цифровом процессоре.

Регистр состояния (FLAGS) хранит:

- **Флаги результата:** CF (перенос), ZF (ноль), SF (знак), OF (переполнение), PF (чётность)
- **Управляющие флаги:** IF (разрешение прерываний), DF (направление), TF (трассировка)

Используется для условных переходов и управления режимами процессора.

Вопрос 51. Назначение и функция регистра счётчика команд (указателя команд).

Указатель команд (IP/EIP/RIP) содержит адрес следующей команды.

Функции:

- Автоматически увеличивается после выборки каждой команды
- Изменяется командами переходов (JMP, CALL, RET)
- Изменяется при входе в прерывание и возврате (IRET)

Вопрос 52. Какие регистры x86 могут использоваться в качестве операндов?

- **32-битные РОН:** EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP
- **16-битные части:** AX, BX, CX, DX, SI, DI, BP, SP
- **8-битные части:** AH, AL, BH, BL, CH, CL, DH, DL
- **Сегментные:** CS, DS, ES, SS, FS, GS (с ограничениями)

Вопрос 53. Является ли счётчик команд в x86 программно доступным?

Нет, счётчик команд (IP/EIP/RIP) **программно недоступен** — его нельзя использовать как операнд-источник или операнд-приёмник в командах. Изменяется только косвенно: командами переходов, вызовов, возвратов и прерываний.

Вопрос 54. В чём состоит функция АЛУ?

АЛУ (Арифметико-Логическое Устройство) выполняет:

- Арифметические операции: сложение, вычитание, умножение, деление
- Логические операции: AND, OR, XOR, NOT
- Сдвиги и вращения битов
- Сравнение операндов
- Формирование флагов результата

Вопрос 55. Структурные части типовой процессорной команды и их назначение.

1. **Код операции (opcode)** — определяет, какое действие выполнить
2. **Операнд-приёмник (destination)** — куда записать результат
3. **Операнд-источник (source)** — откуда взять данные
4. **Модификаторы адресации (ModR/M, SIB)** — как найти операнды
5. **Непосредственное значение** — константа в команде
6. **Смещение** — для адресации памяти

Вопрос 56. Объясните, каким может быть поведение флагов при выполнении команд.

1. **Устанавливаются** — флаг становится 1 по условию
2. **Сбрасываются** — флаг становится 0 по условию
3. **Модифицируются** — изменяются в соответствии с результатом
4. **Не изменяются** — команда не влияет на флаг
5. **Не определены** — значение непредсказуемо после команды

Пример: ADD модифицирует CF, ZF, SF, OF; MOV не изменяет флаги; MUL оставляет ZF, SF неопределёнными.

Вопрос 57. Изменяются ли значения флагов при выполнении команд пересылки данных?

Нет, команды пересылки данных (MOV, XCHG, PUSH, POP, LEA) **не изменяют флаги**. Это важное свойство, позволяющее сохранять результат предыдущей операции для последующего условного перехода.

Вопрос 58. Приведите разновидности команд с условным исполнением в x86.

1. **Условные переходы (Jcc):** JZ, JNZ, JC, JNC, JG, JL, JA, JB, JO, JS и др.
2. **Условная пересылка (CMOVcc):** CMOVZ, CMOVNZ, CMOVG, CMOVL — пересылка выполняется только при выполнении условия
3. **Условная установка байта (SETcc):** SETZ, SETNZ — устанавливает байт в 1 или 0 в зависимости от условия
4. **Условные циклы:** LOOPE, LOOPNE — цикл с дополнительным условием

Вопрос 59. Какие команды помогают в организации вычислений с повышенной разрядностью?

`ADC dst, src`	Сложение с переносом: dst = dst + src + CF
`SBB dst, src`	Вычитание с заёмом: dst = dst - src - CF
`MUL src`	Умножение с двойным результатом: EDX:EAX = EAX × src
`IMUL src`	Знаковое умножение с двойным результатом
`DIV src`	Деление двойного делимого: EAX = EDX:EAX / src

Вопрос 60. Что происходит с операндом-источником при выполнении команд пересылки?

Операнд-источник **остаётся неизменным**. Данные **копируются**, а не перемещаются. После команды MOV EAX, EBX значение в EBX сохраняется, в EAX записывается копия.

Вопрос 61. Почему для машинного представления чисел используется двоичная система?

- **Простота реализации** — два устойчивых состояния (ток есть/нет, напряжение высокое/низкое)
- **Помехоустойчивость** — большой запас между уровнями 0 и 1
- **Надёжность** — меньше вероятность ошибки распознавания
- **Математическая основа** — булева алгебра для логических операций

Вопрос 62. Всегда ли используется двоичная система кодирования?

Нет, существуют альтернативные системы:

- **BCD** — двоично-десятичный код (4 бита на цифру)
- **Код Грея** — соседние значения отличаются одним битом
- **ASCII, Unicode** — кодирование символов
- **Коды с исправлением ошибок** — код Хэмминга, ECC

Вопрос 63. Для чего используются восьмеричная и шестнадцатеричная системы?

Для **компактной записи двоичных чисел**:

- 1 восьмеричная цифра = 3 бита
- 1 шестнадцатеричная цифра = 4 бита (полубайт)

Пример: $11010110_2 = D6_{16} = 326_8$

Вопрос 64. Как перевести число из одной позиционной системы счисления в другую?

В десятичную:

$$N_{10} = \sum_i a_i \cdot p^i$$

где a_i — цифры, p — основание исходной системы.

Из десятичной:

- Целая часть: последовательное деление на основание, остатки — цифры (снизу вверх)
- Дробная часть: последовательное умножение на основание, целые части — цифры

Вопрос 65. Каков диапазон представимых значений в n-битовой сетке для беззнаковых чисел?

$$0 \leq x \leq 2^n - 1$$

Примеры:

- 8 бит: 0 ... 255
- 16 бит: 0 ... 65535
- 32 бита: 0 ... 4 294 967 295

Вопрос 66. Каков диапазон представимых значений в n-битовой сетке для знаковых чисел (дополнительный код)?

$$-2^{n-1} \leq x \leq 2^{n-1} - 1$$

Примеры:

- 8 бит: -128 ... +127
- 16 бит: -32768 ... +32767
- 32 бита: -2 147 483 648 ... +2 147 483 647

Вопрос 67. Как обнаруживается переполнение разрядной сетки при сложении/вычитании?

- **Для беззнаковых:** флаг **CF (Carry Flag)** — перенос из старшего бита
- **Для знаковых:** флаг **OF (Overflow Flag)** — устанавливается, когда знак результата некорректен (оба операнда одного знака, результат — противоположного)

Вопрос 68. Что делать, если диапазон целочисленных операндов слишком мал?

- Использовать типы **большей разрядности** (16 → 32 → 64 бита)
- Использовать **библиотеки длинной арифметики**
- Применять **команды ADC/SBB** для вычислений с повышенной разрядностью
- Использовать **числа с плавающей точкой** (с потерей точности)

Вопрос 69. Как поменять знак двоичного числа в дополнительном коде?

Алгоритм: инвертировать все биты и прибавить 1.
Команда x86: `NEG dst` — выполняет dst = 0 - dst
Пример:

```
+5  = 00000101
инв = 11111010
+1  = 11111011 = -5
```

Вопрос 70. Для чего используется команда AND?

`AND dst, src` — побитовое логическое И.
Применения:

- **Маскирование (обнуление) битов:** `AND AL, 0Fh` — обнуляет старшие 4 бита
- **Проверка битов:** `AND AL, 01h` — ZF=1 если младший бит = 0
- **Выделение битового поля**

Вопрос 71. Для чего используется команда OR?

`OR dst, src` — побитовое логическое ИЛИ.

Применения:

- **Установка битов в 1:** `OR AL, 80h` — устанавливает старший бит
- **Объединение битовых полей**
- **Проверка на ноль:** `OR EAX, EAX` — ZF=1 если EAX=0

Вопрос 72. Для чего используется команда XOR?

`XOR dst, src` — побитовое исключающее ИЛИ.

Применения:

- **Обнуление регистра:** `XOR EAX, EAX` — самый быстрый способ
- **Инверсия битов:** `XOR AL, 0FFh`
- **Переключение бита:** `XOR AL, 01h`
- **Простое шифрование** — XOR с ключом

Вопрос 73. Чем определяются точность и диапазон чисел с плавающей точкой?

Формат числа: $(-1)^S \times M \times 2^E$

- **Точность** определяется количеством бит **мантиссы** (M)
- **Диапазон** определяется количеством бит **порядка** (экспоненты E)

Формат	Порядок	Мантисса	Всего
float	8 бит	23 бита	32 бита
double	11 бит	52 бита	64 бита

Вопрос 74. Как преодолевают неоднозначность представления числа в формате ПТ?

Одно число можно представить по-разному: $1.5 \times 2^3 = 3.0 \times 2^2 = 0.75 \times 2^4$

Решение — **нормализация**: приведение к единственной стандартной форме.

Вопрос 75. Что такое нормализация числа ПТ?

Нормализация — приведение мантиссы к виду $1.xxx..._2$, где старший бит всегда равен 1. Это обеспечивает единственность представления и максимальную точность.

Вопрос 76. Какое преимущество имеет нормализованное представление?

- Единственное представление каждого числа
- **Максимальная точность** — все биты мантиссы значащие
- Возможность использования скрытого бита

Вопрос 77. За счёт чего можно не хранить старший бит мантиссы?

В нормализованном представлении старший бит мантиссы **всегда равен 1**. Поэтому его можно не хранить (**скрытый бит**), получая дополнительный бит точности бесплатно.

Вопрос 78. Что такое «псевдонуль»?

Псевдонуль (денормализованное число) — число с нулевым порядком и ненулевой мантиссой. Используется для представления очень малых чисел, близких к нулю, расширяя диапазон в сторону малых значений.

Вопрос 79. Сколько разновидностей сдвига реализовано в x86 и каковы их свойства?

Команда	**Название**	**Свойства**
SHL/SAL	Сдвиг влево	Младшие биты заполняются 0. Умножение на 2^n
SHR	Логический сдвиг вправо	Старшие биты заполняются 0. Для беззнаковых — деление на 2^n
SAR	Арифметический сдвиг вправо	Знаковый бит сохраняется. Для знаковых — деление на 2^n
ROL	Циклический влево	Выдвигаемый бит возвращается справа
ROR	Циклический вправо	Выдвигаемый бит возвращается слева
RCL	Циклический влево через CF	В цикл включён флаг CF
RCR	Циклический вправо через CF	В цикл включён флаг CF

Вопрос 80. Какие команды позволяют проверить операнд на условия =0, >0 и т.п.?

- `CMP dst, src` — сравнение: вычисляет dst - src, устанавливает флаги, результат не сохраняет
- `TEST dst, src` — проверка: вычисляет dst AND src, устанавливает флаги, результат не сохраняет

После этих команд используются условные переходы (Jcc).

Вопрос 81. Условия команд ветвления разбиты на три подгруппы. Укажите их.

1. **Для беззнаковых (Above/Below):**
   - JA/JNBE — выше (CF=0 и ZF=0)
   - JAE/JNB — выше или равно (CF=0)
   - JB/JNAE — ниже (CF=1)
   - JBE/JNA — ниже или равно (CF=1 или ZF=1)
2. **Для знаковых (Greater/Less):**
   - JG/JNLE — больше (ZF=0 и SF=OF)
   - JGE/JNL — больше или равно (SF=OF)
   - JL/JNGE — меньше (SF≠OF)
   - JLE/JNG — меньше или равно (ZF=1 или SF≠OF)
3. **По отдельным флагам:** JZ/JE, JNZ/JNE, JC, JNC, JS, JNS, JO, JNO, JP, JNP

Вопрос 82. Есть две команды «переход, если больше нуля». Как выбрать?

- `JA` (Jump if Above) — для **беззнаковых** чисел
- `JG` (Jump if Greater) — для **знаковых** чисел

Пример: FFh как беззнаковое = 255 > 0 (JA сработает), как знаковое = -1 < 0 (JG не сработает).

Вопрос 83. Как организовать цикл, если специальная команда отсутствует?

```
        MOV CX, count       ; Счётчик итераций
loop_start:
        ; ... тело цикла ...
        DEC CX              ; CX = CX - 1
        JNZ loop_start      ; Если CX != 0, повторить
```

Вопрос 84. Есть ли специальная команда организации цикла в x86?

Да:

- `LOOP label` — уменьшает CX/ECX на 1, переход если CX≠0
- `LOOPE/LOOPZ` — переход если CX≠0 И ZF=1
- `LOOPNE/LOOPNZ` — переход если CX≠0 И ZF=0

Примечание: на современных процессорах DEC+JNZ часто быстрее, чем LOOP.

Вопрос 85. Какие регистры x86 можно использовать в косвенно-регистровой адресации?

- **32-битный режим:** EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP
- **16-битный режим:** только BX, BP, SI, DI

Вопрос 86. Чем отличается CALL от JMP?

	JMP	**CALL**
Сохранение адреса возврата	Нет	Да (в стек)
Возможность возврата	Нет	Да (командой RET)
Назначение	Безусловный переход	Вызов подпрограммы

Вопрос 87. Каковы основные действия команды вызова подпрограммы?

1. `ESP = ESP - 4` (в 32-битном режиме)
2. `[ESP] = EIP` (сохранение адреса следующей команды)
3. `EIP = адрес_подпрограммы` (переход)

Вопрос 88. Где может сохраняться адрес возврата?

- **В стеке** — x86, ARM (при вызовах с сохранением LR)
- **В регистре связи (Link Register)** — многие RISC-процессоры (ARM, PowerPC)
- **В первой ячейке подпрограммы** — старые архитектуры

Вопрос 89. Что такое «контекст программы»?

Контекст программы — полное состояние процессора, необходимое для продолжения выполнения:

- Регистры общего назначения
- Регистр флагов
- Указатель команд (IP)
- Указатель стека (SP)
- Сегментные регистры

Вопрос 90. Зачем требуется сохранение или восстановление контекста?

- При **вызове подпрограмм** — чтобы не испортить данные вызывающей функции
- При **обработке прерываний** — чтобы вернуться к прерванной программе
- При **переключении задач** — для многозадачности в ОС
- При **обработке исключений**

Вопрос 91. Какие команды помогают сохранять-восстанавливать контекст в x86?

`PUSH reg / POP reg`	Сохранение/восстановление одного регистра
`PUSHA / POPA`	Все 16-битные РОН (AX,CX,DX,BX,SP,BP,SI,DI)
`PUSHAD / POPAD`	Все 32-битные РОН
`PUSHF / POPF`	Регистр флагов (16 бит)
`PUSHFD / POPFD`	Регистр флагов (32 бита)

Вопрос 92. Что такое «стековый кадр»?

Стековый кадр (stack frame) — область стека, выделяемая для одного вызова функции:

```
[EBP+12]  Параметр 2
[EBP+8]   Параметр 1
[EBP+4]   Адрес возврата
[EBP]     Сохранённый EBP  <-- EBP указывает сюда
[EBP-4]   Локальная 1
[EBP-8]   Локальная 2      <-- ESP указывает сюда
```

Вопрос 93. Как организуется доступ к элементам стекового кадра?

Используется **базовая адресация** через регистр EBP:

- `[EBP+8]`, `[EBP+12]`, ... — параметры функции
- `[EBP-4]`, `[EBP-8]`, ... — локальные переменные
- `[EBP]` — сохранённый EBP вызывающей функции
- `[EBP+4]` — адрес возврата

Вопрос 94. Что такое «ввод-вывод, отображённый на память» (MMIO)?

Memory-Mapped I/O — способ организации, при котором регистры устройств располагаются в адресном пространстве памяти. Доступ через обычные команды MOV. Можно использовать все режимы адресации.

Вопрос 95. Что такое «изолированный ввод-вывод»?

Isolated I/O (Port-Mapped I/O) — отдельное адресное пространство для портов ввода-вывода. Доступ через специальные команды IN и OUT. В x86: 64K портов (0000h-FFFFh).

Вопрос 96. Преимущества и недостатки изолированного ввода-вывода.

Преимущества	**Недостатки**
Не занимает адресное пространство памяти	Требуются специальные команды (IN/OUT)
Явное разделение памяти и устройств	Ограниченные возможности адресации
Легче контролировать доступ к портам	Меньше гибкости

Вопрос 97. Как организован ввод-вывод в архитектуре x86?

Используется **изолированный ввод-вывод**:

- `IN AL, port` — ввод байта из порта
- `IN AL, DX` — ввод, номер порта в DX
- `OUT port, AL` — вывод байта в порт
- `OUT DX, AL` — вывод, номер порта в DX

Также поддерживается MMIO для современных устройств (видеопамять, PCI).

Вопрос 98. Каковы недостатки синхронизации с использованием поллинга?

Поллинг — циклическая проверка флага готовности устройства.
Недостатки:

- Непроизводительная трата процессорного времени
- Задержка реакции (зависит от частоты опроса)
- Невозможность эффективной работы с несколькими устройствами
- Повышенное энергопотребление

Вопрос 99. Преимущества использования механизма прерывания.

- Процессор свободен до наступления события
- Быстрая реакция на события
- Возможность приоритизации событий

- Эффективная работа с множеством устройств
- Экономия энергии (процессор может «спать»)

Вопрос 100. Различия между входом в подпрограмму и входом в обработчик прерывания.

CALL (подпрограмма)	**Прерывание**
Сохраняется только IP (и CS)	Сохраняются FLAGS, CS, IP
IF не изменяется	IF сбрасывается (прерывания запрещаются)
TF не изменяется	TF сбрасывается
Вызов синхронный	Вызов асинхронный
Возврат: RET	Возврат: IRET

Вопрос 101. Когда может начаться вход в прерывание по отношению к текущей команде?

Вход в прерывание начинается **после завершения текущей команды**. Процессор не прерывает команду посередине. Исключение: некоторые длинные строковые команды (REP MOVS) могут быть прерваны.

Вопрос 102. Какова судьба запроса, пришедшего при запрещённых прерываниях?

Запрос **остаётся активным** (удерживается контроллером прерываний) и будет обработан сразу после разрешения прерываний (STI), если к тому моменту не будет снят устройством.

Вопрос 103. Прервётся ли обработчик при запросе от более приоритетного источника?

Зависит от режима:

- **При запрещённых прерываниях (IF=0):** нет, запрос будет ждать
- **При разрешённых вложенных прерываниях:** да, если новый запрос более приоритетный

Вопрос 104. Как процессор узнаёт, где находится обработчик?

По номеру прерывания процессор обращается к **таблице векторов прерываний**:

- Реальный режим: адрес = номер × 4, читает 4 байта (segment:offset) из IVT
- Защищённый режим: читает дескриптор из IDT по регистру IDTR

Вопрос 105. При каких условиях допустимы вложенные прерывания?

- Флаг IF установлен (командой STI) внутри обработчика
- Новое прерывание имеет более высокий приоритет
- Обработчик корректно сохраняет весь используемый контекст
- Достаточно места в стеке

Вопрос 106. К каким последствиям может привести необдуманное разрешение вложенных прерываний?

- **Переполнение стека** — каждое прерывание занимает место
- **Взаимные блокировки (deadlock)** — при неправильной синхронизации
- **Гонки данных** — при работе с разделяемыми ресурсами
- **Инверсия приоритетов**
- **Непредсказуемые задержки**

Вопрос 107. Что такое программные прерывания?

Программные прерывания — прерывания, вызываемые командой `INT n`. Синхронны (выполняются в определённой точке программы). Используются для:

- Системных вызовов ОС
- Вызова сервисов BIOS
- Отладки (INT 3)

Вопрос 108. Какую функцию выполняет бит IF регистра состояния?

IF (Interrupt Flag) — флаг разрешения прерываний:

- IF = 1 — маскируемые прерывания разрешены
- IF = 0 — маскируемые прерывания запрещены

Управляется: CLI (сброс), STI (установка), автоматически при входе в прерывание (сброс).

Вопрос 109. Какую функцию выполняет бит TF регистра состояния?

TF (Trap Flag) — флаг трассировки:

- TF = 1 — после каждой команды генерируется прерывание INT 1
- TF = 0 — нормальное выполнение

Используется отладчиками для пошагового выполнения программы.

Вопрос 110. Требование к команде для организации точки останова.

Команда точки останова должна:

1. Занимать **1 байт** — чтобы можно было заменить любую команду
2. Вызывать прерывание — для передачи управления отладчику

В x86: `INT 3` имеет однобайтовый код `0xCC`.

Вопрос 111. Сколько векторов прерываний имеется в x86?

256 векторов (номера 0-255):

- 0-31 — зарезервированы для исключений процессора
- 32-255 — доступны для аппаратных и программных прерываний

Вопрос 112. Как разрешить/запретить аппаратные прерывания в x86?

- `CLI` — запретить (IF = 0)
- `STI` — разрешить (IF = 1)
- Программирование маски в контроллере прерываний (IMR)

Вопрос 113. Какие прерывания можно запретить, какие нельзя?

- **Можно запретить:** маскируемые аппаратные (INTR)
- **Нельзя запретить:** NMI, исключения процессора, программные прерывания INT n

Вопрос 114. Почему при исключениях сохраняется адрес вызвавшей команды?

Для возможности **повторного выполнения команды** после устранения причины исключения.

Пример: при Page Fault:

1. Сохраняется адрес команды, вызвавшей исключение
2. ОС загружает нужную страницу в память
3. Команда выполняется повторно

Вопрос 115. Можно ли использовать JMP для точки останова?

Нет, потому что:

- JMP занимает минимум 2 байта (короткий) или 5 байт (дальний)
- JMP не сохраняет адрес возврата
- JMP не вызывает прерывание для передачи управления отладчику

Вопрос 116. Что такое «сигнал» и «информационный параметр сигнала»?

Сигнал — физическая величина (или процесс), несущая информацию. Примеры: напряжение, ток, электромагнитная волна.

Информационный параметр — характеристика сигнала, кодирующая данные:

- Уровень напряжения (в цифровой логике)
- Частота (в FM-модуляции)
- Фаза (в PSK-модуляции)

Вопрос 117. Почему используют двухуровневое кодирование?

- **Высокая помехоустойчивость** — два уровня легко различить
- **Простота реализации** — два устойчивых состояния транзистора
- **Надёжность** — большой запас между уровнями
- **Совместимость с булевой алгеброй**

Вопрос 118. Как изображаются два значения двоичного разряда?

- **Логический 0:** низкое напряжение (около 0 В, до V_{OL})
- **Логическая 1:** высокое напряжение (от V_{OH} до V_{CC})

Между уровнями — запрещённая зона для помехоустойчивости.

Вопрос 119. Простейшая электрическая цепь для формирования двухуровневого сигнала.

Инвертор (элемент НЕ) на транзисторе с резистором подтяжки:

- Вход = 0: транзистор закрыт → выход = 1 (через резистор к V_{CC})
- Вход = 1: транзистор открыт → выход = 0 (через транзистор к земле)

Вопрос 120. Почему используют двунаправленные линии ввода-вывода?

- **Экономия выводов** корпуса микроконтроллера
- **Гибкость конфигурации** — один вывод может быть и входом, и выходом
- **Поддержка двунаправленных протоколов** (I^2C)
- **Адаптация под задачу** без изменения аппаратуры

Вопрос 121. Типовая регистровая модель подсистемы параллельного ввода-вывода.

Регистр	**Название**	**Назначение**
DDR	Data Direction Register	Задаёт направление: 0=вход, 1=выход
PORT	Port Register	Значения для вывода (или подтяжка для входа)
PIN	Pin Register	Чтение текущего состояния выводов

Вопрос 122. Во сколько раз возрастёт производительность вычисления $(a+b)(b-c)(d+b)$ на Pentium по сравнению с i486?

Анализ:

- Операции $(a+b)$, $(b-c)$, $(d+b)$ — **независимы**
- Два умножения требуют результатов сложений

Intel 80486: один конвейер — все операции последовательно.
Intel Pentium: два конвейера (U и V) — суперскалярная архитектура:

- $(a+b)$ и $(b-c)$ — параллельно
- $(d+b)$ — следующий такт
- Умножения — с учётом зависимостей

Ускорение: примерно в 1.5-2 раза, но не более чем в 2 раза из-за:

- Ограничения двумя конвейерами
- Зависимостей по данным между операциями
- Не все пары команд могут выполняться параллельно