1. Объясните смысл и напишите формулы для следующих метрик производительности процессора

- IPC (Instructions Per Cycle) – среднее количество выполняемых процессором инструкций за 1 такт

  $IPC = \frac{N_{inst}}{N_{cycle}}$, где N_{inst} – количество инструкций в программе, N_{cycle} – количество затраченных тактов
- CPI (Cycles Per Instructions) – среднее количество процессорных тактов для выполнения 1 инструкции

  $CPI = \frac{1}{IPC}$
- Performance - производительность процессора

  $Perf = \frac{1}{Time} = \frac{1}{N \cdot CPI \cdot T_{cycle}} = \frac{1}{N} \cdot IPC \cdot f$, где N - количество инструкций в программе, T_{cycle} - время 1 процессорного такта, f - тактовая частота процессора
- Dynamic Power - динамически потребляемая процессором мощность; затраты на перезарядку паразитных емкостей при переключении транзисторов $DynPower = \frac{1}{2} C \cdot V^2 \cdot f \cdot a$, где C - эффективная емкость, V - приложенное напряжение, a - частота переключения транзисторов

2. Что такое суперскалярный процессор?

Суперскалярный процессор - процессор, который поддерживает ILP (Instruction Level Parallelism), параллелизм на уровне инструкций, то есть способен исполнять больше одной инструкции за такт.

Примеры:

- дополнительные исполнительные устройства (ALU - Arithmetic and Logical Unit, FPU - Floating-Point Unit)
- дополнительные конвейеры (возможно, урезанные)
- наборы векторных инструкций (SIMD - Single Instruction Multiple Data) - MMX, SSE, AVX
- VLIW (Very Large Instruction Word) архитектура, исполняющая независимые команды пачками (bundles)

3. Какие типы зависимостей по данным существуют? Приведите примеры аппаратных оптимизаций, которые позволяют сократить связанные задержки или разрешить каждый тип зависимостей

Выделяют 3 типа зависимостей по данным:

- WAW (Write After Write) - ложная зависимость

- WAR (Write After Read) - ложная зависимость
- RAW (Read After Write) - истинная зависимость

В случае ложных зависимостей последующей команде реально не нужен результат, выработанный предыдущей. Но они мешают переупорядочиванию, ведь здесь важен порядок следования store-ов в память для сохранения и дальнейшего чтения правильного значения. К тому же, здесь возникает вопрос, на какой регистр стоит положить результат, чтобы не было пересечений.

Ложные зависимости устраняются с помощью

- переименования архитектурных регистров в физические (Register Renaming) с помощью RAT (Register Aliases Table)
- разбиения операций store на STA (STore Address calculation) и STD (STore Data calculation)

Истинные зависимости более проблематичные, ведь load инструкция может использовать значение, которое вырабатывает инструкция store, а store инструкции не могут быть переупорядочены.

Они устраняются с помощью

- использования Load и Store Buffers для непересекающихся адресов load-а и предшествующих store-ов
- Store Forwarding оптимизации, в процессе которой значения передаются из операций store в операции load напрямую через байпасс

К тому же, компилятор может помочь аппаратуре, сгенерировав более оптимальный код с меньшим количеством зависимостей с помощью грамотно выполненных оптимизаций распределения регистров и планирования инструкций.

4. С какой целью инструкцию Store разделяют на микро-операции STA (Store address calculation) и STD (Store data calculation)?

Для уменьшения количества ложных зависимостей в случае отсутствия пересечения адресов. Для этого необходимо заранее знать адрес store-а, который и вычисляется в данной оптимизации с помощью STA микрооперации. Если последующие операции не используют этот адрес, они могут быть переупорядочены.

5. Объясните назначение и функции следующих аппаратных структур:

- ROB - ReOrder Buffer. Буфер для in-order хранения инструкций и исполнения их out-of-order в случае отсутствия зависимостей для поддержания спекулятивного локального состояния. Для поддержания реального архитектурного состояния инструкции покидают ROB в in-order порядке и изменяют состояние памяти на стадии retire

- Scheduler Queue (Issue Queue, Reservation Station) - очередь для еще не выполненных инструкций. Все части out-of-order исполнения производятся над инструкциями из этой очереди, а на ROB остается только in-order запись результатов в память
- RAT - Register Aliases Table, таблица соответствия физических и архитектурных регистров для выполнения оптимизации Register Renaming. Каждая запись в ROB-е обновляет ее при обновлении архитектурного состояния
- PRF - Physical Register File. Регистровый файл, хранящий все физические регистры процессора. Используется при Register Renaming для создания соотвествия между физическими и архитектурными регистрами, чтобы на стадии retire корректно обновить состояние памяти
- Load Buffer - очередь для хранения операций load и необходимой для них информации. Нужна для исполнения специфичных для load-ов оптимизаций, например, поиска ложных зависимостей
- Store Buffer - аналогичная очередь для хранения операций store. Нужна для исполнения специфичных для store-ов оптимизаций. Для аналога спекулятивного исполнения для store-ов сначала запись идет не в память, а именно в этот буфер, откуда еще можно откатиться.

6. Пусть каждая 5ая инструкция в процессоре это Branch. Предсказатель переходов имеет точность 90%. Оцените, ROB какого максимального размера имеет смысл для такого процессора.

При ложном предсказании адреса происходит очистка конвейера и ROB, чего следует избегать. Обозначим $P_{branch} = 0.2$, $P_{predict} = 0.9$, N - размер ROB

Всего branch-ей в ROB $N_{branch} = N \cdot P_{branch}$

Вероятность правильного предсказания их всех: $P_N = (P_{predict})^{N_{branch}}$

Откуда получаем $N = \frac{log(P)}{P_{branch} \cdot log(P_{predict})}$

Взяв $P_N = 0.9$, получим $N = 5$. Размер слишком мал для реального использования, стоит сделать лучший branch predictor.

7. Что такое Memory Disambiguation?

Memory Disambiguation - это набор вышеперечисленных техник для уменьшения количества зависимостей по данным и повышения ILP для большей возможности внеочередного исполнения команд. К нему относятся Register Renaming, STA/STD разбиение, Store Forwarding, Load and Store Buffer, Load Speculation, и другие оптимизации.

8. В чем заключается проблема со спекулятивным исполнением Store инструкций?

Store инструкции необходимо исполнять в in-order порядке, поскольку они влияют на состояние памяти, и внеочередное их исполнение может привести к последующему чтению невер-

ных результатов, а адекватных способов отменить store-ы в случае неверного спекулятивного исполнения нет.

9. Что такое Store forwarding и Load speculation в OOO процессоре?

Store Forwarding - передача данных из store в load напрямую через байпасс, если адреса совпадают.

Load Speculation - исполнение load-а спекулятивно, не дожидаясь предшествующих инструкций. Включает в себя следующие техники:

- Dependence Prediction - предсказывает завимости между load и store инструкциями
- Address Prediction - предсказывает адрес load-а
- Value Prediction - предсказывает значение, которое загрузит load
- Memory Renaming - предсказывает зависимости между load и store инструкциями и передает результат store-а в load

10. Что такое Simultaneous Multithreading?

Simultaneous MultiThreading (SMT) - техника, позволяющая использование нескольких логических потоков на одном физическом ядре отдельно друг от друга. Происходит разделение физического процессора на несколько логических, у каждого из которых имеется свое архитектурное состояние, а каждая стадия конвейера может быть занята только 1 логическим процессором.