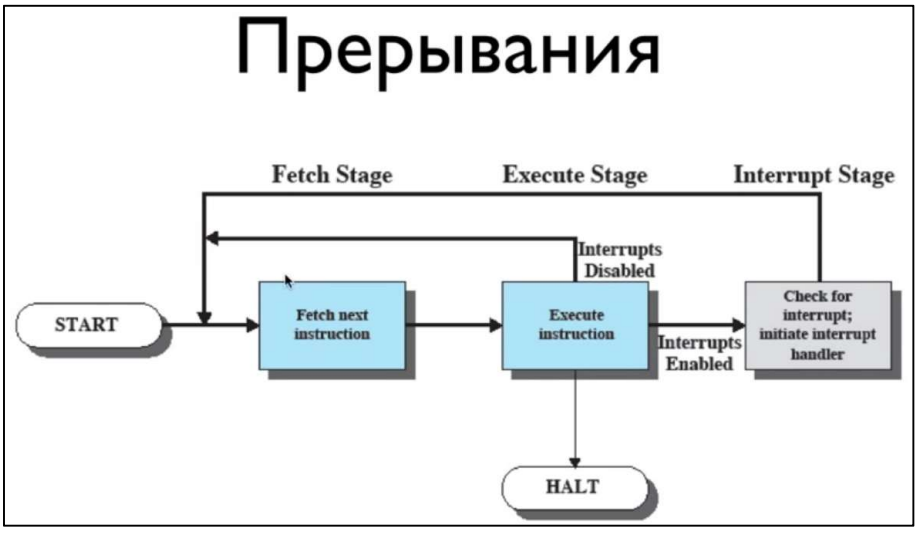
Прерывания

Инструкции могут быть важные, а могут быть не очень важные. Когда исполняется одна инструкция или какая-то команда, или программа – зачастую бывает необходимо исполнить какую-то другую инструкцию или другую программу. Для этого существуют прерывания. **Прерывания** – это прерывание нормального порядка исполнения инструкции. Если нам нужно исполнить программу, которая состоит из ста инструкций, нормальный порядок – это когда все эти 100 инструкций исполняются одна за другой: 1 – 2 – 3 и так до 100. Прерывание – это когда в какой-то момент этот порядок останавливается и происходит что-то другое. Это необходимо **для улучшения использования центрального процессора**. Дело в том, что **процессор** очень **быстрый**, это очень **мощный** мозг, но другие части компьютера намного медленнее. **Самые**, наверное, **медленные** части компьютера – **это устройства ввода-вывода**, особенно те устройства, которые механические, например, старые магнитные жесткие диски или какой-нибудь принтер. Они очень медленные по сравнению с процессором, поэтому если процессор будет ждать от них ответа, то он будет «бездельничать» и, чтобы этого бездействия не проходило, чтобы мы не тратили это время, чтобы этих издержек не было, мы можем прервать его текущую операцию (допустим он ждет какого-то ответа от принтера) и занять его чем-то другим. В любом случае нужно помнить, что **паузы и простои – это очень плохо**. Если процессор отдыхает, значит мы теряем производительность.

Также прерывания нужны для **реализации многозадачности**. Несмотря на то, что во многих компьютерах один процессор, на нем одновременно могут работать несколько программ: это операционная система, это множество программ, которые находятся в операционной системе, это множество наших программ (у нас играет музыка, открыт браузер и так далее). Это возможно если процессор будет постоянно и очень быстро переключаться с программы на программу, и это будет настолько быстро, что для нас и для многих программ будет казаться, что все они исполняются одновременно. На самом деле он просто очень быстро переключается между ними, и мы не можем заметить этого приключения.



Теперь дополним нашу схему прерываниями. До этого у нас был простой цикл, и он будет продолжать существовать до тех пор, пока не происходит прерывание. В любой момент, после исполнения очередной инструкции, мы проверяем – нет ли прерывания, нет ли чего-то, что необходимо исполнить прямо сейчас. Если нет, то мы продолжаем цикл, продолжаем нашу нормальную обработку программы. Однако если в какой-то момент прерывание существует, то мы останавливаемся, выходим из этого цикла и обрабатываем то, что нам нужно обработать согласно

этому прерыванию (это может быть какая другая инструкция или целая отдельная программа). После этого мы возвращаемся и продолжаем цикл.

Пока всё еще довольно просто. Мы делаем все по порядку, но в какой-то момент нас могут отвлечь. Можно найти аналогию этому в нашей работе: мы сидим и пишем какой-нибудь доклад, и мы пишем страницу за страницей, пока не закончим, но если нас отвлекут на какую-то более важную работу, то мы поставим «паузу» в этом докладе и пойдем делать эту работу, потом вернемся и закончим свой доклад. Нас могут отвлечь еще раз, естественно мы потеряем время, но зато мы сделаем две вещи. И если мы будем делать это очень быстро, то наш начальник подумает, что мы многозадачные, что мы на самом деле делали одновременно две вещи.

Другой пример с ожиданием процессора, когда процессор ждет более медленное устройство, это, как если бы мы набрали этот свой доклад, который нам нужно было набрать, потом пошли его печатать, а медленный офисный принтер печатает 100 страниц в течение 10 минут и мы стоим около принтера, ждем и ничего не делаем. Умное прерывание могло бы нас отвлечь, и мы пошли бы и сделали что-нибудь полезное, потом вернулись, забрали бы свой распечатанный доклад и не потеряли бы эти 10 минут.

Множественные исключения

Что, если прерывание произошло в тот момент, когда обрабатывается другое прерывание? Тут есть несколько путей, но два самых простых варианта: первое – это **запретить прерывания при обработке прерываний**. В той же аналогии с людьми, если я что-то делаю, а меня отвлекли, чтобы сделать что-то другое, то в этот момент меня уже нельзя отвлекать. Если ко мне кто-то подойдет и скажет: «Слушай, а давай сделаем что-то еще», – я скажу: «Нет, вот когда я вернусь к своей основной работе, то там меня можно отвлекать, а сейчас меня уже отвлекли, я не могу углубляться». Другой, намного более гибкий путь – это **использовать приоритеты**. Если компьютер или, в этой аналогии, работник может оценить важность каждой операции, тогда мы можем решить, стоит ли отвлекаться или не стоит отвлекаться. Если меня отвлекают на какую-то чушь, то у этой чуши приоритет ниже, поэтому я не буду отвлекаться сейчас и закончу данную важную задачу. Если же меня отвлекают на критическую задачу, допустим пожар и мне нужно убегать, то у этой задачи приоритет выше, и если компьютер знает о каждой операции, о каждом приоритете, то он может принимать такое решение и исключения будут обрабатываться более гибко. Всё это делается только для того, чтобы компьютер работал более эффективно.

Мультипрограммирование

Есть такое понятие как мультипрограммирование. Процессор должен исполнять несколько программ и порядок исполнения зависит от приоритета и от взаимодействия с устройствами ввода-вывода. Устройства ввода-вывода медленные, они очень сильно влияют на порядок. Но основная идея всей этой архитектуры в том, чтобы не ждать, не простаивать, не стоять на паузе. Из-за того, что у нас может быть это мультипрограммирование, когда мы работаем с несколькими программами одновременно, пытаемся обработать нужды нескольких программ, когда обработка одного прерывания завершена управление не обязательно возвращается к той программе, которая исполнялась до этого. Опять же в аналогии, если мы писали доклад и нас отвлекли, возможно такое, что, когда мы эту задачу, на которую нас отвлекли, закончим, мы вернемся не к докладу, а к чему-то еще, потому что пока мы там работали, кто-то пришел и оставил у нас на столе записку, что есть еще более важные задачи.

Это именно то, что может происходить с компьютерами. Процессор не будет «злиться», он просто будет всё исполнять. Это с одной стороны здорово, потому что мы можем рассчитать всё это

и можем рассчитывать на процессор. С другой стороны, это не очень хорошо, потому что человек, в отличии от процессора, может принять какое-то решение, а какое-то глупое решение отвергнуть, а процессор не отвергнет глупое решение и, если мы написали такую архитектуру, где процессор используется очень плохо, то он и будет использоваться очень плохо, и полная ответственность за производительность и за весь результат этой операции будет лежать только на инженерах и собственно людях.