WIKIPEDIA
The Free Encyclopedia

Исполнение (вычисления)

Исполнение в компьютерной и программной инженерии — это процесс, посредством которого компьютер или виртуальная машина интерпретирует и выполняет инструкции компьютерной программы . Каждая инструкция программы представляет собой описание конкретного действия, которое необходимо выполнить для решения конкретной задачи. Исполнение включает в себя многократное выполнение цикла « выборка–декодирование–выполнение » для каждой инструкции, выполняемой устройством управления . По мере того, как исполняющая машина следует инструкциям, возникают определённые эффекты в соответствии с семантикой этих инструкций.

Программы для компьютера могут выполняться в пакетном режиме без участия человека, или пользователь может вводить команды в интерактивном сеансе интерпретатора . В этом случае « команды » — это просто инструкции программы, выполнение которых связано в цепочку.

Термин **«запустить»** используется практически как синоним. Близкие значения слов «запустить» и «выполнить» относятся к определённому действию пользователя, запускающему (или *вызывающему*) *программу* , например: «Пожалуйста, запустите приложение».

Процесс

Перед выполнением программа должна быть написана. Обычно это делается в исходном коде , который затем компилируется во время компиляции (и статически линкуется во время компоновки) для создания исполняемого файла. Затем этот исполняемый файл вызывается, чаще всего операционной системой, которая загружает программу в память (во время загрузки), возможно, выполняет динамическую компоновку , а затем начинает выполнение, передавая управление точке входа программы; все эти шаги зависят от двоичного интерфейса приложения операционной системы. В этот момент начинается выполнение, и программа переходит в режим выполнения . Затем программа выполняется до завершения, либо в результате нормального завершения , либо в результате сбоя .

Исполняемый файл

Исполняемый код , *исполняемый файл* или *исполняемая программа* , иногда называемый просто *исполняемым файлом* или *двоичным файлом* , представляет собой список инструкций и данных, заставляющих компьютер «выполнять указанные задачи в соответствии с закодированными инструкциями » [1] , в отличие от файла данных , который должен быть интерпретирован (анализирован) программой, чтобы иметь смысл.

Точное толкование зависит от сферы применения. Под «инструкциями» традиционно понимаются инструкции машинного кода для физического процессора . [2] В некоторых контекстах файл, содержащий инструкции скрипта (например, байт-код), также может

считаться исполняемым.

Контекст исполнения

Контекст, в котором происходит выполнение, имеет решающее значение. Очень немногие программы выполняются на «голой» машине . Программы обычно содержат явные и неявные предположения о ресурсах, доступных во время выполнения. Большинство программ выполняются в многозадачной операционной системе и библиотеках времени выполнения, специфичных для исходного языка, которые предоставляют критически важные сервисы, не предоставляемые непосредственно самим компьютером. Эта поддерживающая среда, например, обычно отделяет программу от прямого управления периферийными устройствами компьютера, предоставляя вместо этого более общие, абстрактные сервисы.

Переключение контекста

Чтобы программы и обработчики прерываний работали бесперебойно, используя одну и ту же аппаратную память и доступ к системе ввода-вывода, в многозадачной операционной системе , работающей на цифровой системе с одним ЦП/МК, требуются программные и аппаратные средства для отслеживания данных исполняемого процесса (адресов страниц памяти, регистров и т. д.), а также для их сохранения и восстановления до состояния, в котором они были до приостановки. Это достигается переключением контекста. [3] : 3.3 [4] Выполняемым программам часто назначаются идентификаторы контекста процесса (PCID).

В операционных системах на базе Linux набор данных, хранящихся в регистрах , обычно сохраняется в дескрипторе процесса в памяти для реализации переключения контекста. [3] Также используются идентификаторы PCID.

Время выполнения

Время выполнения (runtime) , **время выполнения (runtime**) или **время исполнения (execution time)** — это заключительная фаза жизненного цикла компьютерной программы , в которой код выполняется на центральном процессоре (ЦП) компьютера в виде машинного кода . Другими словами, «время выполнения» — это фаза выполнения программы.

Ошибка времени выполнения обнаруживается после или во время выполнения программы (в рабочем состоянии), тогда как ошибка времени компиляции обнаруживается компилятором до её выполнения. Проверка типов , выделение регистров , генерация кода и оптимизация кода обычно выполняются во время компиляции, но могут выполняться и во время выполнения в зависимости от конкретного языка программирования и компилятора. Существует множество других ошибок времени выполнения, которые обрабатываются по-разному в разных языках программирования , например, ошибки деления на ноль , ошибки домена, ошибки выхода за пределы индекса массива , ошибки арифметического переполнения , несколько типов ошибок переполнения и переполнения , а также многие

другие ошибки времени выполнения, обычно рассматриваемые как программные ошибки, которые могут быть обнаружены и обработаны или не обнаружены каким-либо конкретным языком программирования.

Подробности реализации

При запуске программы загрузчик сначала выполняет необходимую настройку памяти и связывает программу со всеми необходимыми динамически подключаемыми библиотеками , после чего начинается выполнение программы с точки входа . В некоторых случаях язык или реализация программирования поручает эти задачи исполняющей среде языка, хотя это нетипично для распространённых языков программирования в распространённых потребительских операционных системах.

Отладка некоторых программ может быть выполнена (или выполняется более эффективно и точно) только во время выполнения. Примерами служат логические ошибки и проверка границ массива . По этой причине некоторые ошибки программирования не обнаруживаются до тех пор, пока программа не будет протестирована в рабочей среде с реальными данными, несмотря на сложную проверку во время компиляции и предрелизное тестирование. В этом случае конечный пользователь может столкнуться с сообщением об ошибке выполнения.

Ошибки приложения (исключения)

Обработка исключений — одна из функций языка, предназначенная для обработки ошибок времени выполнения, которая обеспечивает структурированный способ выявления как совершенно неожиданных ситуаций, так и предсказуемых ошибок или необычных результатов без необходимости встроенной проверки ошибок, характерной для языков без неё. Более поздние достижения в области механизмов выполнения позволяют автоматизировать обработку исключений , предоставляя отладочную информацию о «первопричине» каждого интересующего исключения и реализуя её независимо от исходного кода путём подключения специального программного продукта к механизму выполнения.

Система выполнения

Система *выполнения* , также называемая *средой выполнения* , в первую очередь реализует части модели выполнения . Это не следует путать с фазой жизненного цикла выполнения программы, в течение которой система выполнения находится в эксплуатации. При рассмотрении системы *выполнения* как отдельного элемента прикладного программного обеспечения (IDE), используемого для программирования , части программного обеспечения, которая предоставляет программисту более удобную среду для запуска программ во время их производства (*тестирования* и т.п.), в то время как вторая (RTE) будет самим примером модели выполнения, применяемой к разработанной программе, которая затем сама запускается в вышеупомянутой *системе выполнения* .

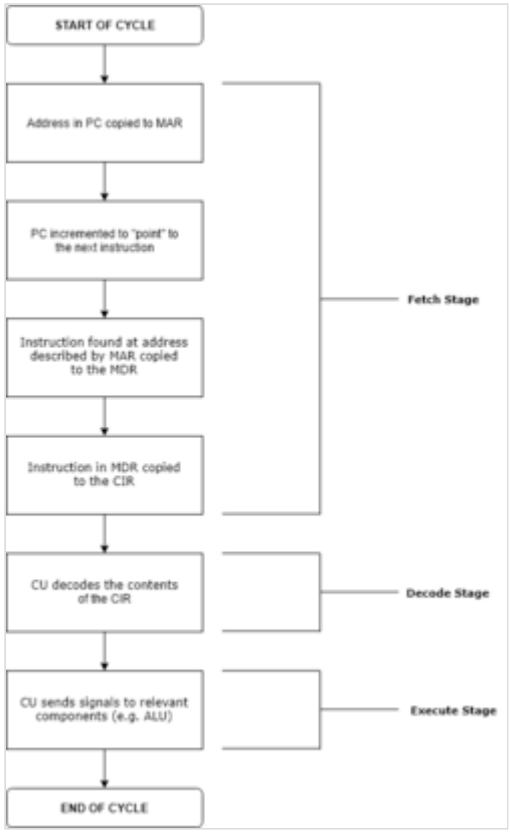
Большинство языков программирования имеют некую форму системы выполнения, которая предоставляет среду, в которой работают программы. Эта среда может решать ряд вопросов, включая управление памятью приложения , способ доступа программы к

переменным , механизмы передачи параметров между процедурами , взаимодействие с операционной системой и т. д. Компилятор делает предположения в зависимости от конкретной системы выполнения для генерации корректного кода. Обычно система выполнения отвечает за настройку и управление стеком и кучей и может включать такие функции, как сборка мусора , потоки или другие динамические функции, встроенные в язык. [5]

Цикл инструкций

Цикл *инструкций* (также известный как *цикл выборки-декодирования-выполнения* или просто *цикл выборки-выполнения*) — это цикл, который центральный процессор (ЦП) выполняет с момента загрузки до выключения компьютера для обработки инструкций. Он состоит из трёх основных этапов: этапа выборки, этапа декодирования и этапа выполнения.

В более простых процессорах цикл инструкций выполняется последовательно, при этом каждая инструкция обрабатывается до начала выполнения следующей. В большинстве современных процессоров циклы инструкций выполняются одновременно , а часто и параллельно , через конвейер инструкций : обработка следующей инструкции начинается до завершения предыдущей, что возможно благодаря разделению цикла на отдельные этапы. [6]



Это простая диаграмма, иллюстрирующая отдельные этапы цикла выборка-декодирование-выполнение.

Устный переводчик

Система, выполняющая программу, называется интерпретатором программы. Грубо говоря, интерпретатор непосредственно выполняет программу. Это отличается от переводчика языка , который преобразует программу с одного языка на другой перед её выполнением.

Виртуальная машина

Виртуальная *машина* (*ВМ*) — это виртуализация / эмуляция компьютерной системы . Виртуальные машины основаны на компьютерных архитектурах и обеспечивают функциональность физического компьютера. Их реализация может включать специализированное оборудование, программное обеспечение или их комбинацию.

Виртуальные машины различаются и организованы по своим функциям, как показано здесь:

- **Системные виртуальные машины** (также называемые виртуальными машинами полной виртуализации) заменяют реальную машину. Они предоставляют функциональность, необходимую для работы целых операционных систем . Гипервизор

использует встроенные функции для совместного использования и управления оборудованием, позволяя создавать несколько изолированных друг от друга сред на одной физической машине. Современные гипервизоры используют аппаратную виртуализацию , специфичную для виртуализации, в основном на базе процессоров хоста.

- **Виртуальные машины процессов** предназначены для выполнения компьютерных программ в среде, независимой от платформы.

Некоторые эмуляторы виртуальных машин, такие как QEMU и эмуляторы игровых консолей , также предназначены для эмуляции (или «виртуальной имитации») различных системных архитектур, что позволяет запускать приложения и операционные системы, написанные для другого процессора или архитектуры. Виртуализация на уровне ОС позволяет разделять ресурсы компьютера через ядро . Эти термины не являются взаимозаменяемыми.

Смотрите также

- Исполняемый файл
- Система времени выполнения
- Фаза выполнения программы
- Счетчик программ

Ссылки

1. "executable" (http://www.merriam-webster.com/dictionary/executable) . *Электронный словарь Merriam-Webster* . Merriam-Webster . Дата обращения : 19 июля 2008 г.
2. "Машинные инструкции" (https://www.geeksforgeeks.org/machine-instructions/) . *GeeksforGeeks* . 2015-11-03 . Дата обращения : 2019-09-18 . (https://www.geeksforgeeks.org/machine-instructions/)
3. Бовет, Дэниел П. (2005).*Понимание ядра Linux*. Марко Чезати (3-е изд.). Севастополь, Калифорния: O'Рейли.ISBN 0-596-00565-2. OCLC 64549743 (https://search.worldcat.org/oclc/64549743) .
4. "Разница между подкачкой и переключением контекста" (https://www.geeksforgeeks.org/difference-between-swapping-and-context-switching/) . *GeeksforGeeks* . 2021-06-10 . Дата обращения : 2022-08-10 . (https://www.geeksforgeeks.org/difference-between-swapping-and-context-switching/)
5. Ахо, Альфред В.; Лам , Моника Син-Линг ; Сети, Рави ; Ульман, Джеффри Дэвид (2007). *Составители: Принципы, методы и инструменты* (https://archive.org/details/compilers00alfr_0/page/427) (2 - е изд.). Бостон, Массачусетс, США: Pearson Education . С. 427. (https://archive.org/details/compilers00alfr_0/page/427) ISBN (https://archive.org/details/compilers00alfr_0/page/427) (https://archive.org/details/compilers00alfr_0/page/427) 978-0-321-48681-3.
6. Кристал Чен, Грег Новик и Кирк Шимано (2000). «Конвейеризация» (https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/pipelining/index.html). Дата обращения : 26 июня 2019 г. (https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/pipelining/index.html)