Влияние машинной архитектуры

- На разрабатываемый язык программирования машинная архитектура влияет двояко:
  - непосредственно через реальный компьютер;
  - через модель выполнения, или виртуальный компьютер, который поддерживает реализуемый язык на реальном компьютере.

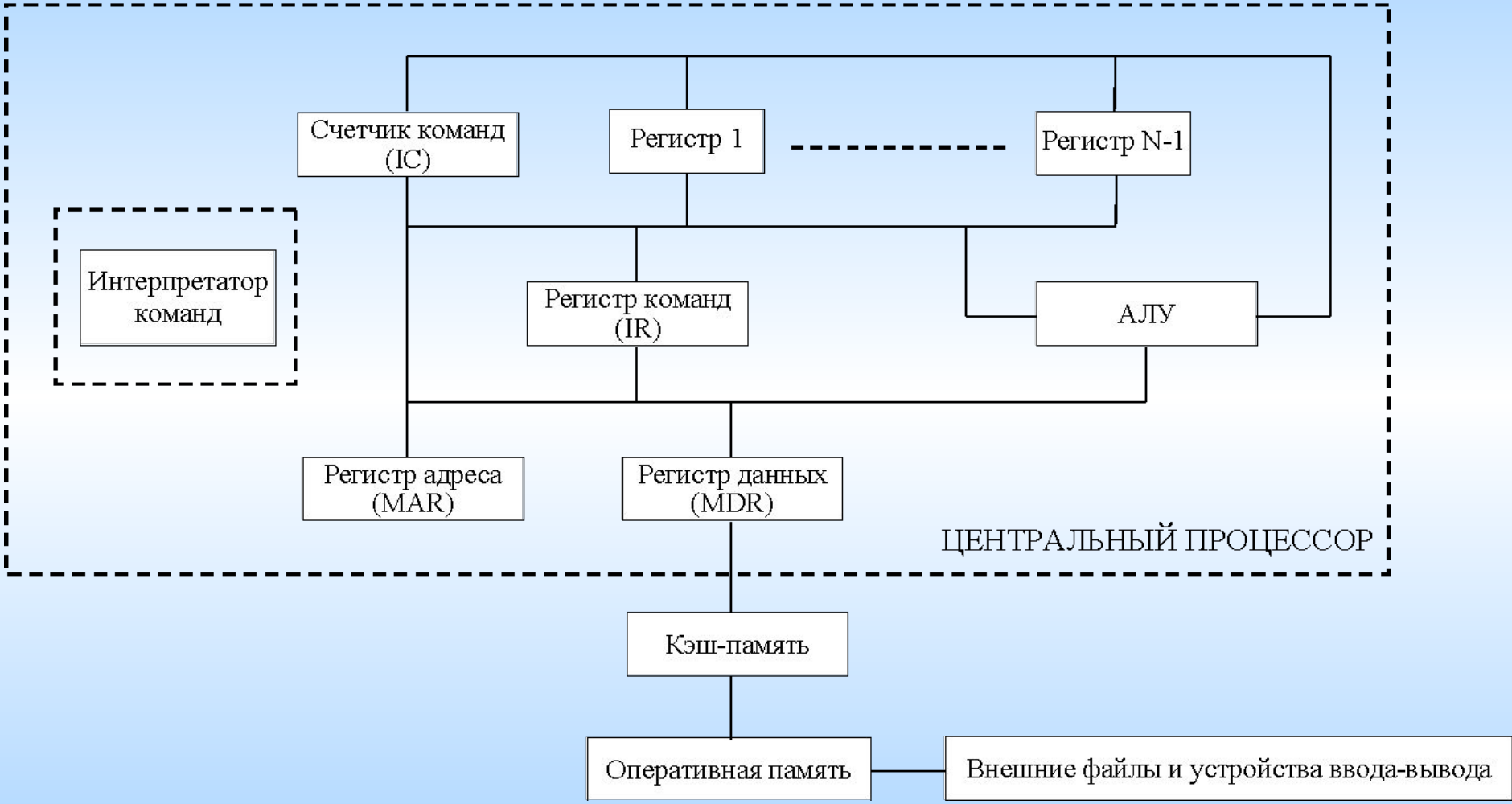
Структура и принципы работы компьютера

- *Компьютер* можно определить как интегрированный набор алгоритмов и структур данных, способный хранить и выполнять программы.
- Компьютер может быть создан как реальное физическое устройство — это *реальный*, или *физический компьютер*, или просто *компьютер*.
- Компьютер можно «построить» с помощью программ, выполняемых на каком-то другом компьютере — это *программно-моделируемым* компьютером.
- Реализация языка программирования осуществляется путем создания *транслятора*, который транслирует программы, написанные на данном языке, в программы на машинном языке, которые уже могут непосредственно выполняться на каком-либо компьютере.

Компоненты компьютера

- *Данные*. Компьютер должен обрабатывать различные типы простейших элементов данных и структур данных.
- *Элементарные операции*. Компьютер должен иметь набор элементарных операций для работы с данными.
- *Управление последовательностью действий*. Компьютер должен обеспечивать управление последовательностью выполнения элементарных операций.
- *Доступ к данным*. Компьютер должен предоставлять механизмы управления данными, которые необходимы для выполнения любой операции.
- *Управление памятью.* Компьютер должен предоставлять механизмы управления распределением памяти для программ и данных.
- *Операционная среда.* Компьютер должен предоставлять механизмы связи с внешней средой, содержащей программы и данные, подлежащие обработке.

Организация традиционного компьютера



Управление последовательностью действий

- **Типичное машинное вычисление**
- Типичный цикл выборки: ([x] означает содержимое регистра x)

```
[IC] → MAR
IC +1 → IC
Чтение содержимого памяти в MDR
MDR → IR
```

- Типичный цикл выполнения: (на примере выбранной из памяти инструкции `OP R, X, DISP`)

```
[IR] декодируется OP R, EA {OP код операции
(например, 8 бит), R регистр (например, 4 бита для
машины с 16 регистрами и словом длиной 32 бита),
EA исполнительный адрес (например, 20 бит)}
[X]+DISP → MAR {EA → MAR}
Чтение содержимого памяти и MDR
[R] → АЛУ; [MDR] → АЛУ
Выполнение операции OP в АЛУ; АЛУ → R
```

- Для, например, процессора с тактовой частотой 500 МГц: каждая инструкция требует в среднем 9-10 циклов (50 MIPS — миллион команд в секунду), с перекрытием циклов выборки и выполнения 60-70 MIPS

Состояния компьютера

- *Статическая организация* компьютера - данные, операции, управляющие структуры и т. п. представляет только часть общей картины.
- Для полного понимания необходимо также ясно представлять себе *динамику работы* компьютера во время выполнения программы.
- Для наблюдения за динамикой работы компьютера удобно использовать понятие *состояние компьютера*. Каждое состояние характеризуется содержимым оперативной памяти, регистров и внешней памяти в определенные моменты времени в процессе выполнения программы.
  - Исходное состояние этих областей памяти определяет, соответственно, *исходное состояние* компьютера.
  - Каждый шаг при выполнении программы преобразует существующее состояние компьютера в некоторое новое состояние посредством изменения содержимого одной или нескольких указанных областей памяти. Это преобразование называется *сменой состояний*.
  - Когда выполнение программы заканчивается, окончательное содержимое этих областей памяти определяет *конечное состояние* компьютера.
- Таким образом, выполнение программы можно рассматривать как процесс последовательной смены состояний компьютера.

Программно-аппаратный компьютер

- машинный язык — это язык низкого уровня с простыми форматами команд и операциями типа «сложить два числа» и «загрузить в регистр содержимое области памяти».
- Всегда можно *сконструировать аппаратное устройство*, машинный язык которого будет в точности тем самым выбранным языком программирования (например, компьютер B5500 фирмы Burroughs с языком ALGOL).
- На практике в качестве машинных языков обычно используются языки низкого уровня
- *Программно-аппаратный компьютер* моделируется *микропрограммой*, выполняемой на специальном *микропрограммируемом компьютере*.
  - Машинный язык этого компьютера состоит из набора *микрокоманд* очень низкого уровня, которые обычно реализуют простые передачи данных между оперативной памятью и быстрыми регистрами, непосредственно между самими регистрами и из одних регистров в другие через такие обрабатывающие устройства, как сумматоры и умножители, а с их помощью программируются другие команды.
  - Получившийся в результате компьютер называется *виртуальным компьютером*, поскольку он смоделирован микропрограммой, без которой он бы просто не существовал.

Трансляторы (1)

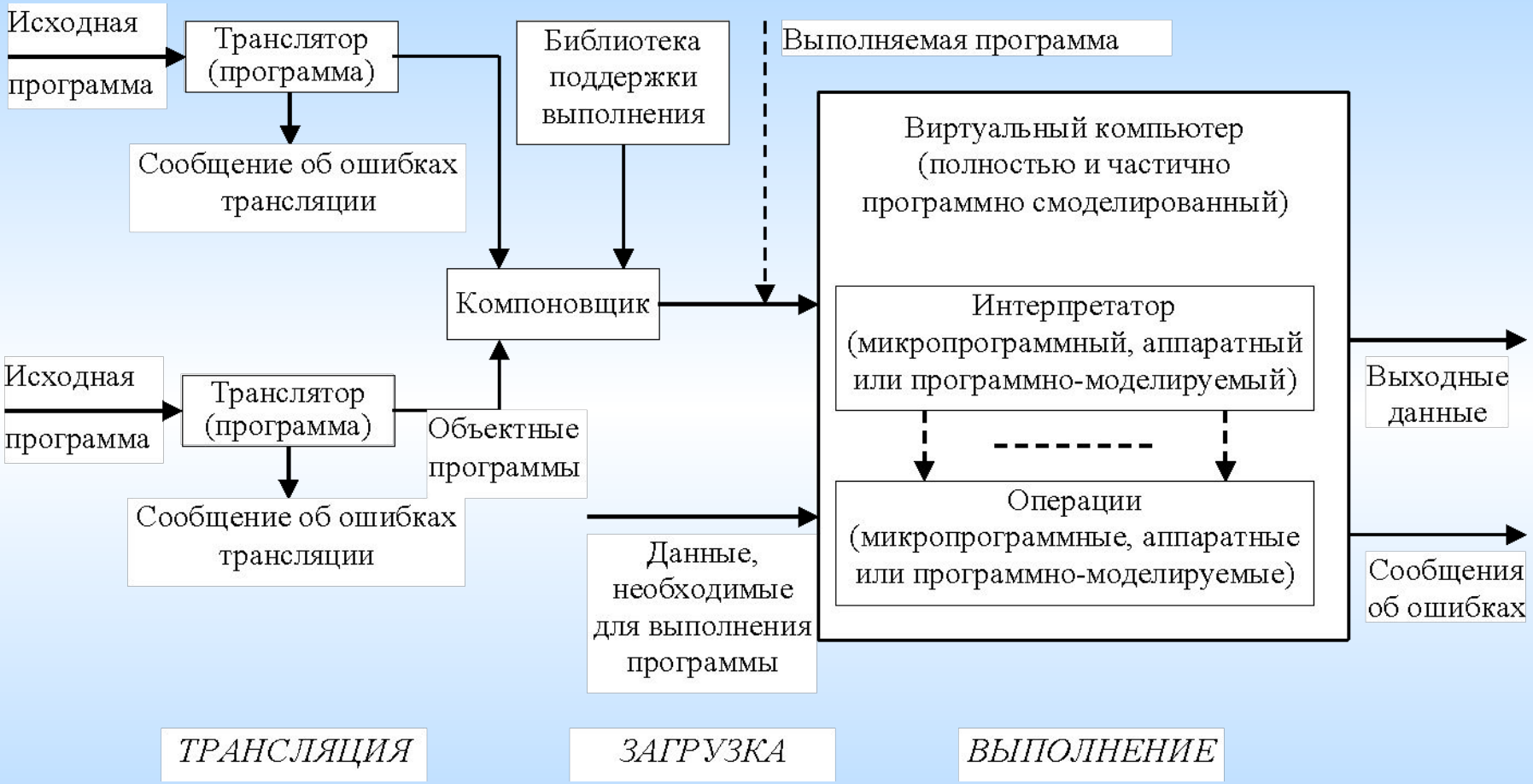
- *Компиляция*. Можно сконструировать транслятор, который будет переводить программы, написанные на языках высокого уровня, в эквивалентные программы на машинном языке используемого компьютера. Существуют термины для обозначения некоторых специальных типов трансляторов:
  - *Ассемблер* — это транслятор, у которого объектный язык представляет собой некую разновидность машинного языка какого-либо реального компьютера, а исходный язык, называемый *языком ассемблера* или просто *ассемблером* — это, как правило, символическое представление объектного машинного кода. В большинстве случаев каждая инструкция на исходном языке переводится в одну команду на объектном языке.
  - *Компилятор* — это транслятор, для которого исходным является язык высокого уровня. Объектный язык близок к машинному языку реального компьютера — это либо язык *ассемблера*, либо какой-либо вариант машинного языка.
  - *Загрузчик* или *редактор связей* — это транслятор, у которого объектный язык состоит из готовых к выполнению машинных команд, а исходный язык почти идентичен объектному. Обычно он *состоит* из программ на машинном языке в *перемещаемой* форме и таблиц данных, указывающих те точки, в которых перемещаемый код должен быть модифицирован, чтобы стать действительно выполняемым. Задачей загрузчика является создание единой *выполняемой программы*, в которой используются согласованные адреса, как показано в следующей таблице.
  - *Препроцессор*, или *макропроцессор* — это транслятор, исходный язык которого является расширенной формой какого-либо языка высокого уровня (например, Java или C++), а объектный язык — стандартной версией этого языка.

Подпрограмма	Адреса после компиляции	Адреса в выполняемой программе
P	0–999	0–999
Q	0–1999	1000–2999
библиотека	0–4999	3000–7999

Трансляторы (2)

- *Программная имитация* (*программная интерпретация*). Вместо того чтобы транслировать программы на языке высокого уровня в эквивалентные программы на машинном языке, можно смоделировать с помощью программ, выполняемых на хост-компьютере, *другой компьютер, для которого машинным языком будет данный язык высокого уровня*. Для этого нужно сконструировать набор программ на машинном языке хост-компьютера, которые моделируют алгоритмы и структуры данных, необходимые для выполнения программ на языке высокого уровня. Другими словами, при помощи программ, выполняемых на хост-компьютере, создается (моделируется) компьютер с машинным языком высокого уровня; этот компьютер мог бы быть создан и аппаратным способом. Подобный метод называется *программной имитацией* (или *программной интерпретацией*) компьютера с машинным языком высокого уровня на хост-компьютере. Для моделируемой машины входными данными являются программы на языке высокого уровня.
  - В таком случае мы говорим, что хост-компьютер создает *виртуальный компьютер*, моделирующий язык высокого уровня. Когда хост-компьютер выполняет программу на языке высокого уровня, сложно сказать, выполняется ли программа непосредственно аппаратной частью компьютера или сначала она преобразуется в программу на машинном языке низкого уровня, а затем уже выполняется.

Трансляторы (3)



Трансляторы (4)

- *Компилируемые языки*. Языки C, C++, FORTRAN, Pascal и Ada принято считать компилируемыми. Это означает, что программы, написанные на этих языках, транслируются в машинный код данного компьютера перед началом выполнения. Программная интерпретация при этом ограничивается только интерпретацией набора *программ поддержки выполнения*, которые моделируют элементарные операции исходного языка, не имеющие близкого аналога в машинном языке. Транслятор компилируемого языка является обычно довольно большой и сложной программой, и при трансляции основное значение имеет создание максимально эффективных, с точки зрения их выполнения, программ.
- *Интерпретируемые языки*. Языки LISP, ML, Perl, Postscript, Prolog и Smalltalk часто реализуются с использованием программного интерпретатора. При такой реализации транслятор выдает не машинный код используемого компьютера, а некую промежуточную форму программы. Эта форма легче для выполнения, чем исходная программа, но все же она отличается от машинного кода. Процесс интерпретации для выполнения полученной программы должен быть реализован программным образом, поскольку аппаратный интерпретатор в данном случае непосредственно применить невозможно. Использование программного интерпретатора обычно приводит к относительно медленному выполнению программы. Трансляторы интерпретируемых языков обычно представляют собой довольно простые программы, основная сложность реализуется в программном обеспечении процесса интерпретации.
- Java – язык всемирной паутины WWW – интерпретируется байт-код в виртуальной машине Java.

Виртуальные компьютеры

- Компьютер можно сконструировать:
  - Путем *аппаратной реализации*, когда структуры данных и алгоритмы непосредственно представлены физическими устройствами.
  - Путем *программно-аппаратной реализации*, когда структуры данных и алгоритмы представлены микропрограммами, которые выполняются на соответствующем микропрограммируемом аппаратном компьютере.
  - Как *виртуальный компьютер*, т.е. путем моделирования структур данных и алгоритмов программами, написанными на каком-либо другом языке программирования.
  - Посредством *комбинации перечисленных методов*, когда некоторые части требуемого компьютера реализованы аппаратным способом, другие — программно-аппаратным, третьи — при помощи программного моделирования, то есть виртуальным способом.
- При реализации языка программирования структуры данных и алгоритмы, задействованные при выполнении программы, определяют некоторую вычислительную машину (компьютер). Поскольку этот компьютер всегда (хотя бы частично) моделируется программным образом (как в случае с программно-аппаратной реализацией компьютера), он называется *виртуальным компьютером, определяемым реализацией языка*. Машинным языком этого виртуального компьютера будет являться выполняемая программа, выдаваемая транслятором этого языка.

Виртуальные компьютеры и реализация ЯП

- Различия в реализациях одного и того же языка объясняются следующими тремя факторами:
  - Разные разработчики могут иметь различные мнения о том, как должен быть устроен виртуальный компьютер, неявно определяемый данным языком программирования.
  - Различные базовые компьютеры, на которых реализуется язык, предоставляют различные возможности.
  - Каждый разработчик выбирает собственные решения для реализации элементов виртуального компьютера, используя возможности, предоставляемые базовым компьютером, и для конструирования транслятора таким образом, чтобы он поддерживал выбранные решения для представления виртуального компьютера.

Иерархия виртуальных компьютеров

Связывание и время связывания (1)

- *Связывание* элемента программы с конкретной характеристикой или свойством — это просто выбор определенного свойства из некоторого ряда допустимых свойств.
- Момент времени, когда при составлении или выполнении программы происходит это связывание, называется *временем связывания* данного свойства с данным элементом.

Связывание и время связывания (2)

- *Во время выполнения программы*. Связывание часто происходит во время выполнения программы. Сюда входят связывание переменных с их значениями, а также (во многих языках) связывание переменных с конкретными областями памяти. Здесь можно выделить две важные подкатегории:
  - *При входе в подпрограмму или блок*. В большинстве языков связывание допустимо только при входе в подпрограмму или блок при *выполнении* программы. Например, в языках C и C++ связывание формальных параметров с фактическими и связывание формальных параметров с определенными областями памяти происходит только во время входа в подпрограмму.
  - *В произвольных точках программы во время ее выполнения*. Некоторые типы связывания могут происходить в произвольных точках программы во время ее выполнения. Наиболее важным примером этого является связывание *переменных* с их значениями путем присваивания. В таких языках как LISP, ML и Smalltalk допустимо также связывание имен с областями памяти в произвольный момент при выполнении программы.

Связывание и время связывания (3)

- *Во время трансляции (компиляции)*. Можно выделить три класса связываний, происходящих во время трансляции:
  - *Связывание по выбору программиста*. При написании программы программист принимает множество решений по вопросам выбора типов и имен переменных, структуры элементов программы и т. п. Эти решения определяют связывания, происходящие при трансляции. Транслятор данного языка использует эти связывания для определения окончательной формы объектной программы.
  - *Связывание по выбору транслятора*. В некоторых случаях связывание определяется транслятором языка без непосредственного участия программиста. Например, относительное расположение объекта данных в области памяти, *отведенной* под некоторую процедуру, обычно задается транслятором без какого-либо вмешательства со стороны программиста. Хранение массивов данных и создание (при необходимости) дескрипторов массивов — это также прерогатива транслятора. Для различных реализаций одного и того же языка это связывание может происходить по-разному.
  - *Связывание по выбору загрузчика*. Обычно программа состоит из нескольких подпрограмм, которые должны быть объединены в одну выполняемую программу. Как правило, транслятор связывает переменные с адресами относительно областей памяти, определенных для каждой подпрограммы. Но эти области должны быть размещены в *физической* памяти с действительными адресами компьютера, на котором программа будет выполняться. Это происходит во *время загрузки*, также называемое *временем редактирования связей*.

Связывание и время связывания (4)

- *Время реализации языка*. Некоторые аспекты определений языка в рамках некоторой его реализации могут быть одними и теми же для всех выполняемых программ, однако они могут отличаться в других его реализациях. Например, часто представление чисел и арифметических операций определяется тем, как эти операции реализованы с помощью аппаратной части базового компьютера. Если программа написана на языке, использующем возможности, определения которых были заданы при его реализации, то эта программа не обязательно будет работать в другой реализации этого же языка; хуже того, программа может работать, но выдавать неверный результат.
- *Время определения языка*. В основном структура языка формируется во время определения языка, на основе спецификации альтернатив, доступных программисту при написании программы. Например, возможные альтернативные формы операторов, типы структур данных, программные структуры и т. п. определяются, как правило, во время определения языка.

Связывание и время связывания (5)

- Пример – оператор присваивания
  `X = X + 10`
- *Набор возможных типов для переменной X*. Обычно для переменной в программе определен какой-либо тип данных: это может быть integer (целочисленный), real (вещественный), Boolean (булевый) или какой-то иной тип. Множество возможных типов для переменной X обычно устанавливается при определении языка (например, могут допускаться только типы integer, real, Boolean, set (множество) и character (символ)). Может быть и так, что язык допускает определение новых типов переменных в каждой программе, как, например, в C, Java и Ada. В таком случае множество возможных типов для переменной X устанавливается во время трансляции программы.
- *Тип переменной X*. Конкретный тип данных, ассоциированный с переменной X, часто устанавливается во время трансляции при помощи явного объявления в программе: например, в языке C для того, чтобы задать X как переменную вещественного типа, используется объявление float X. В таких языках, как, например, Perl или Smalltalk, тип данных переменной X определяется во время выполнения программы, в тот момент, когда переменной присваивается значение определенного типа. В этих языках переменная X в одном месте может содержать целое значение, а в другом месте той же программы — строку символов.

Связывание и время связывания (6)

- Пример – оператор присваивания (продолжение)
  `X = X + 10`
- *Множество возможных значений для переменной X*. Если тип данных для переменной X определен как вещественный (real), то при выполнении программы X может принимать некоторые значения из определенного множества — множества последовательностей битов, представляющих в данном аппаратном компьютере вещественные числа. Множество возможных значений для переменной *X* зависит от того, какие вещественные числа допустимы в данном виртуальном компьютере, определяемом языком программирования. Обычно это множество совпадает с множеством вещественных чисел, представление для которых предусмотрено в базовом аппаратном компьютере.
- *Значение переменной X*. В любой момент выполнения программы переменная X связана с каким-либо значением. Это то значение, которое присваивается данной переменной при выполнении программы. Операция присваивания X = X +10 меняет значение переменной X, заменяя старое значение новым, которое больше исходного на 10.

Связывание и время связывания (7)

- Пример – оператор присваивания (продолжение)
  `X = X + 10`
- *Представление константы 10*. Целое число 10 одновременно представлено как константа в тексте программы (с использованием строки 10) и как некая последовательность битов во время выполнения программы. Выбор десятичного представления этого числа (то есть символа 10 для десяти) обычно происходит во время *определения языка программирования*, а выбор конкретной последовательности битов для представления этого числа при выполнении программы происходит во время реализации языка.
- *Свойства операции «+»*. Выбор символа «+» для обозначения операции сложения происходит во время *определения языка*. Однако в зависимости от контекста этот символ может обозначать *вещественное сложение*, *целочисленное сложение*, *комплексное сложение* и т.п. В компилируемом языке принято определять операцию, представляемую символом «+», во время компиляции. Роль механизма, выполняющего это связывание, обычно играет механизм определения типов переменных: если *X* — целое число, то символ «+» обозначает в данном контексте целочисленное сложение; если *X* — вещественное число, то символ «+» обозначает вещественное сложение, и т.д.
- Подробное определение операции, обозначенной символом «+», может также зависеть от аппаратной части базового компьютера. Если в нашем примере X будет иметь значение 249, то на некоторых компьютерах для X+10 не будет возможности представить результат этого сложения.
- Подводя итог, можно сказать, что для языка, подобного C, символ «+» связывается с набором операций сложения *во время определения языка*. Затем во время реализации языка определяется каждый элемент из этого набора, во время трансляции программы каждое конкретное употребление символа «+» в программе связывается с конкретной операцией сложения, а конкретное значение результата операции сложения определяется уже во время выполнения программы.

Связывание и время связывания (8)

- Важность времени связывания
  - В основе различия между языками лежат различия во временах связывания
- В языках, подобных FORTRAN, большая часть связываний происходит на этапе трансляции. Эта ситуация носит название *раннего связывания*.
- Если же связывания происходят в основном при выполнении программы, как в ML или в HTML, то говорят о *позднем связывании*.
- В определении языка обычно не накладывается строгих определений на время связывания. При определении языка предполагается, что какое-то конкретное связывание будет сделано, например, во время трансляции; но окончательно это определяется только при реализации языка.
- в определении языка указывается самое раннее время, когда в процессе обработки программы может произойти связывание, а при реализации языка связывание фактически может быть отложено до более позднего времени. Тем не менее, обычно в различных реализациях данного языка конкретное связывание происходит в одно и то же время.
- Следует, однако, обратить внимание на то, что часто кажущиеся незначительными изменения в языке могут привести к серьезному изменению времени связывания. Например, введение в FORTRAN 90 рекурсии, изменяет время связывания многих важнейших свойств этого языка.