ОП.09. Основы алгоритмизации и программирования

Раздел 1. Общая концепция языков программирования

Тема 1.2.
Эволюция языков программирования

Писать

Язык программирования *– это система команд, понятных ЭВМ.*

Языки программирования делятся на:

- ***машинноориентированные языки*** (языки низкого уровня): Автокоды, Ассемблеры – позволяют управлять вычислительным процессом напрямую, при помощи машинных команд,
- ***языки программирования высокого уровня.***

Писат
ь **Языки высокого уровня делятся на:**

- **процедурные (алгоритмические)** (Basic, Pascal, C и др.), которые предназначены для однозначного описания алгоритмов;

- **логические** (Prolog, Lisp и др.), которые ориентированы не на разработку алгоритма решения задачи, а на систематическое и формализованное описание задачи с тем, чтобы решение следовало из составленного описания;

- **объектно-ориентированные** (Object Pascal, C++, Java и др.), в основе которых лежит **понятие объекта, сочетающего в себе данные и действия над нами.**

- Среди программистов, пишущих программы для персональных компьютеров, наибольшей популярностью пользуются языки **Си, Си++, Паскаль и Бейсик.**

Язык Си

- Язык Си был изобретен в 1972 году **Денисом Ричи и Кеном Томпсоном** для использования в написании весьма ныне популярной операционной системы Unix. Си соединяет свойства языка высокого уровня с возможностью эффективного использования ресурсов компьютера, которое обычно обеспечивается только при программировании на языке Ассемблера. Си не очень прост в обучении и требует тщательности в программировании, но позволяет писать сложные и весьма высокоэффективные программы. Бьярном Страустрапом был разработан язык Си++ - расширение языка Си, реализующее популярные в последнее время концепции объектно-ориентированного программирования и облегчающее создание сложных программ.

Язык Паскаль

- Язык **Паскаль** был разработан в 1970 году **Никлаусом Виртом** как язык для обучения студентов программированию. Паскаль позволяет писать программы, легко читаемые даже новичком, и содержит в себе все элементы, необходимые для соблюдения хорошего строгого стиля программирования (называемого структурным программированием), упрощающего разработку сложных программ. Системы программирования на Паскале для IBM PC также реализуют расширенные варианты этого языка. Из этих реализаций наиболее популярны - Turbo Pascal, Borland Pascal и Borland Pascal for Windows

Язык Бейсик

Язык Бейсик был создан в 1964 году Томасом Куртом и Джоном Кемени как язык для начинающих, облегчающий написание простых программ. Существует много различных версий Бейсика Это язык очень широко распространен на микрокомпьютерах. На IBM PC широко используются Quick Basic и Visual Basic фирмы Microsoft и Turbo Basic фирмы Borland.

Основная идея авторов языка Бейсик - снабдить простым языком программирования непрофиссиональных программистов - оказалась очень привлекательной для большого числа разработчиков различных фирм.

В феврале 1975 года на рынке программмных средств появилась первая версия языка Бейсик для персональных компьютеров, авторами которой были Билл Гейтс и Пол Аллен, сотрудники корпорации Microsoft - ведущего лидера поставщиков программного обеспечения.

Именно этот год стал, по существу, годом, когда Basic вышел в свет. Сочетая в себе простоту, гибкость и универсальность, этот язык стал прообразом многих

Другие языки

- На IBM, кроме выше упомянутых языков, используется и много других языков программирования. Для построения экспертных систем употребляются языки **Лисп (Джона Маккарти)** и **Пролог (Алан Кулмероэ** и **Филипп Руссел)** , для создания информационных систем используют язык **Clipper**. Имеются и реализации языков, которые использовались ранее на больших компьютерах, например **Фортрана (Джон Бэкус**) и **Кобола (Грейс Хоппер**).

Писать

Поколения языков программирования

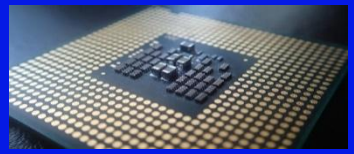
Поколения	Языки программирования	Характеристика
Первое	Машинные	Ориентированы на использование в конкретной ЭВМ, сложны в освоении, требуют хорошего знания архитектуры ЭВМ
Второе	Ассемблеры, макроассемблеры	Более удобны для использования, но по-прежнему машинно-зависимы
Третье	Языки высокого уровня	Мобильные, человеко-ориентированные, проще в освоении
Четвёртое	Непроцедурные, объектно-ориентированные, языки запросов, параллельные	Ориентированы на непрофессионального пользователя и на ЭВМ с параллельной архитектурой
Пятое	Языки искусственного интеллекта, экспертных систем и баз знаний, естественные языки	Ориентированы на повышение интеллектуального уровня ЭВМ и интерфейса с языками

Писать

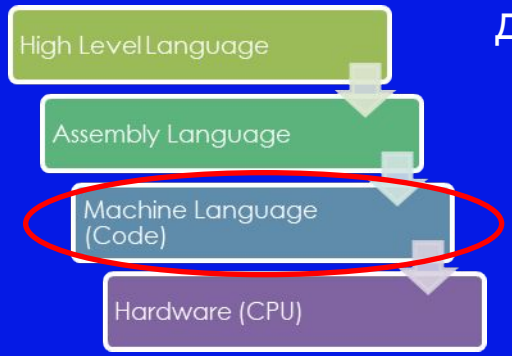
Разница между языками высокого и низкого уровней…

Машинный код ***(Язык очень низкого уровня)***

Процессор может понимать только один тип кода: **Машинный код**



Сделан из кодированных команд и данных.



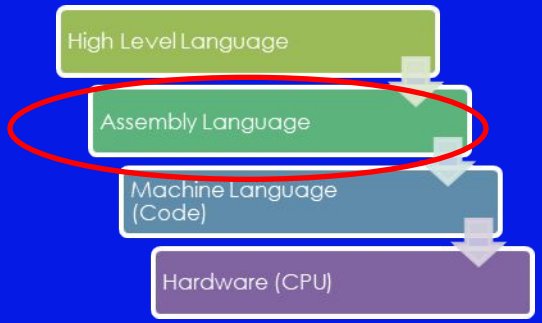
Например, 01001100 (двоичный)

или

(шестнадцатеричный)

написанная программа на «Машинном коде» сложная для понимания человеком, но имеет самую высокую скорость выполнения.

Писат
ь



Machine Code	Mnemonic Code
B80200	MOV AX, 0002
BB0400	MOV BX, 0004
01DB	ADD AX, BX
CD20	INT 20

Язык Ассемблер *(Язык низкого уровня)*

Чтобы преодолеть проблему был создан язык Ассемблер.

Этот язык имеет небольшой набор команд, который представляет определенную часть машинного кода.

Это помогло программистам, т.к. они **не должны были помнить двоичный код**, вместо этого они **учили команды**.

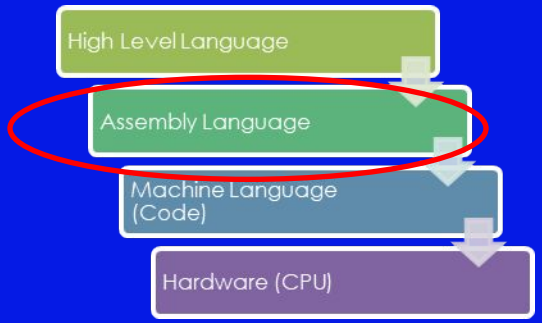
Например: Вместо запоминания, что значит 1011 или 1001, они уже запоминали команды такие как ADD и SUB.

Эти команды известны как **Мнемоники** (простые средства памяти).

Для различных процессоров используются различные версии языка Ассемблер, следовательно каждый процессор может обработать только свою версию Ассемблера.

Писат
ь

Язык Ассемблер *(Язык низкого уровня)*

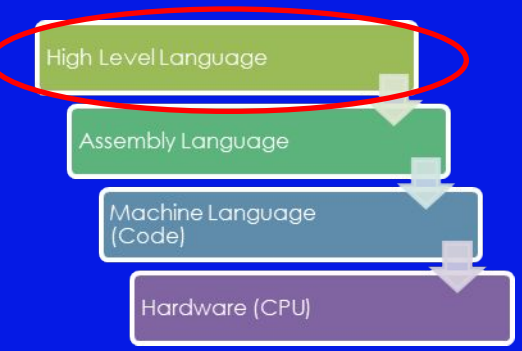


Machine Code	Mnemonic Code
B80200	MOV AX, 0002
BB0400	MOV BX, 0004
01DB	ADD AX, BX
CD20	INT 20

Язык Ассемблер также сложен для изучения и использования и поэтому **были разработаны Языки** программирования **Высокого Уровня**.

*(Благодаря меньшему количеству кода, **язык ассемблер** мог быть обработан процессором быстрее, чем **языки высокого уровня**)*

Писат
ь



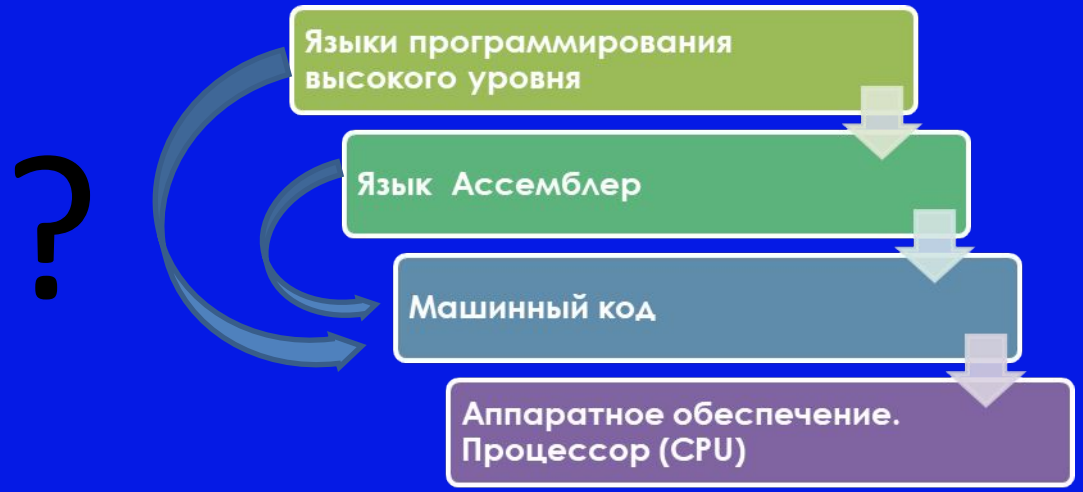
Языки программирования высокого уровня *(Pascal, Basic, Python, Java, C++ и другие)*

Код высокого уровня более **лёгкий для написания** и следовательно, **для понимания человеком**.

Писать

Трансляторы

Порядок перевода *‘НАПИСАННОГО КОДА НА ЧЕЛОВЕЧЕСКОМ ЯЗЫКЕ’* в Машинный код?



Писать

Трансляторы

Трансляторы это программы, которые **конвертируют** команды языка программирования **высокого уровня**:

write, IF, For и т.д.

…в инструкции **машинного кода**:

1011, 11001, 11000011110 и т.д.

…таким образом процессор может обработать данные!

Два типа пути трансляции:

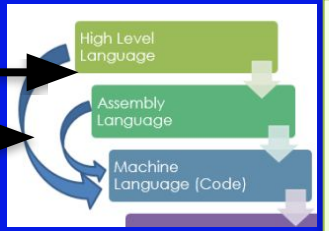
1. Берет целый код и конвертирует его в машинный перед его исполнением (известный как **компиляция**).

2. Берет одну инструкцию кода за один раз, переводит и выполняет ее перед переводом следующей инструкции (известно как **интепретация**).

Писать

Три типа трансляторов:

Translators



Компилятор

Переводит целый код в один файл (часто в .exe файл).

Файл может быть запущен на любом компьютере без транслятора.

Может занять длительное время компиляция исходного кода, т.к. транслятор будет часто конвертировать инструкции в различные наборы машинного кода и проверять будет ли понимать их процессор.

Использует больше памяти, чем интерпретатор, но быстрее запускает программу

Интерпретатор

Конвертирование исходного кода в машинный код строка за строкой «line by line».

Использует меньше памяти, чем компилятор.

Следовательно программа работает очень медленно.

Главная причина почему используется интерпретатор это тестирование на этапе разработки.

Программисты могут быстро выявить ошибки и исправить их.

Транслятор должен присутствовать для запуска программы.

Ассемблер

Это тип транслятора, используемый для языка Ассемблер (не является языком высокого уровня).

Он конвертирует мнемонические инструкции языка ассемблер в машинный код.

Писать

Стандартизация языков программирования.

Язык программирования может быть представлен в виде набора спецификаций, определяющих его **синтаксис и семантику**.

Для многих широко распространённых языков программирования созданы международные стандарты.

Писать

Специальные организации проводят регулярное обновление и публикацию спецификаций и формальных определений соответствующего языка.

В рамках таких комитетов продолжается разработка и модернизация языков программирования и решаются вопросы о расширении или поддержке уже существующих и новых языковых конструкций.

Писать

Типы данных

Современные цифровые компьютеры обычно являются двоичными и данные хранят в двоичном (бинарном) коде.

Эти данные как правило отражают информацию из реального мира (имена, банковские счета, измерения и др.), представляющую высокоуровневые концепции.

Писат
ь

Особая система, по которой данные организуются в программе, — это система типов языка программирования; разработка и изучение систем типов известна под названием теория типов.

Писать

Языки могут быть классифицированы как системы со **статической типизацией** и языки с **динамической типизацией**.

Статически-типизированные языки могут быть в дальнейшем подразделены на языки с обязательной декларацией, где каждая переменная и объявление функции имеет обязательное объявление типа, и языки с выводимыми типами.

Иногда динамически-типизированные языки называются латентно-типизированными.

Писать

Структуры данных

Системы типов в языках высокого уровня позволяют определять сложные, составные типы, так называемые структуры данных.

Основные структуры данных (списки, очереди, хеш-таблицы, двоичные деревья и пары) часто представлены особыми синтаксическими конструкциями в языках высокого уровня. Такие данные структурируются автоматически.

Писать

Семантика языков программирования

Наиболее широко распространены разновидности следующих трёх:

- операционного,
- денотационного (математического)
- и деривационного (аксиоматического).

Писать

- В рамках **операционного подхода** обычно исполнение конструкций языка программирования интерпретируется с помощью некоторой воображаемой (абстрактной) ЭВМ.
- **Деривационная семантика** описывает последствия выполнения конструкций языка с помощью языка логики и задания пред- и постусловий.
- **Денотационная семантика** оперирует понятиями, типичными для математики — множества, соответствия, а также суждения, утверждения и др.

Писать

Парадигма программирования

Язык программирования строится в соответствии с той или иной базовой моделью вычислений и парадигмой программирования.

Писать

Модели вычислений

- императивная модель вычислений (задаётся фон-неймановской архитектурой ЭВМ);
- функциональное программирование (Лисп, Haskell, ML и др.);
- логическое программирование (Пролог) и язык Рефал;
- проблемно-ориентированные программирование;
- декларативные модели языков;
- визуальные языки программирования.