Машинный язык

Эволюция языков программирования

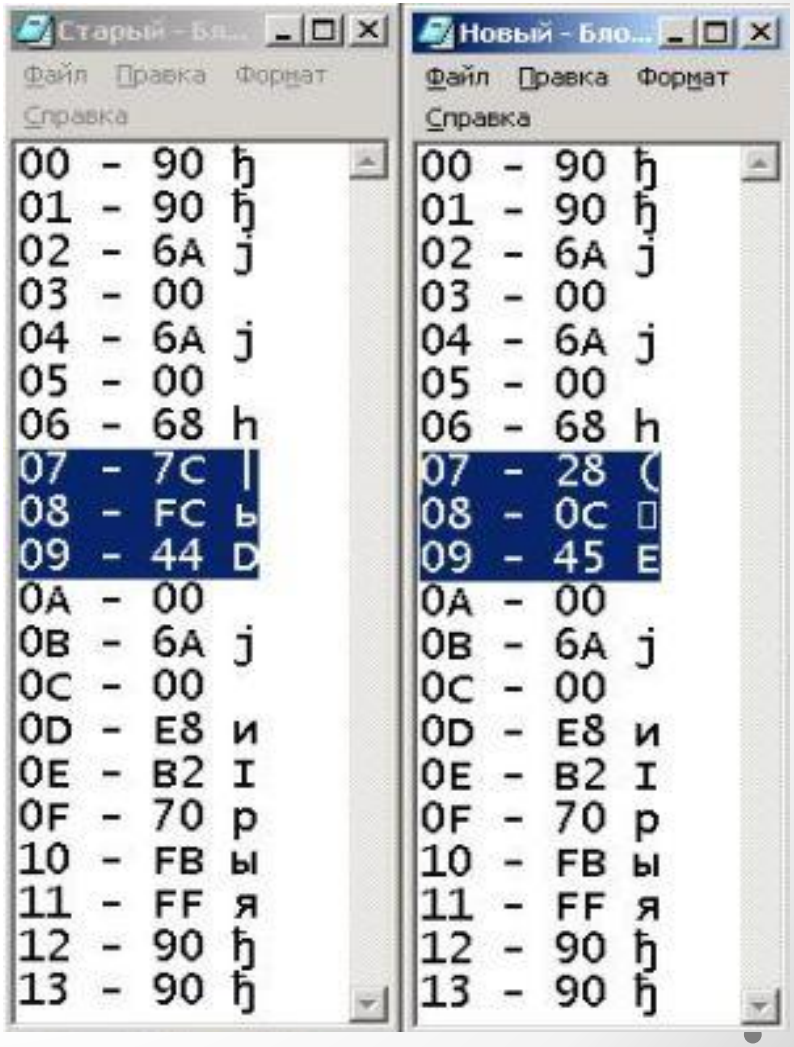
Первоначально процесс программирования предусматривал запись программистом всех алгоритмов непосредственно на машинном языке. Такой подход усугублял и без того трудную задачу разработки алгоритмов.

Программисты были вынуждены запоминать множество кодов в своей памяти. Программирование на машинном коде - сложный и трудоемкий процесс, зачастую приводивший к ошибкам, которые необходимо было обнаружить и исправить до того, как работу можно было считать законченной.

Первым шагом на пути к облегчению задачи программирования был отказ от использования цифр для записи команд и операндов непосредственно в той форме, в которой они используются в машине. Программисты стали использовать мнемонические обозначения для написания программы, и лишь когда программа была полностью составлена – они начинали переводить их в машинный код

Языки программирования низкого уровня

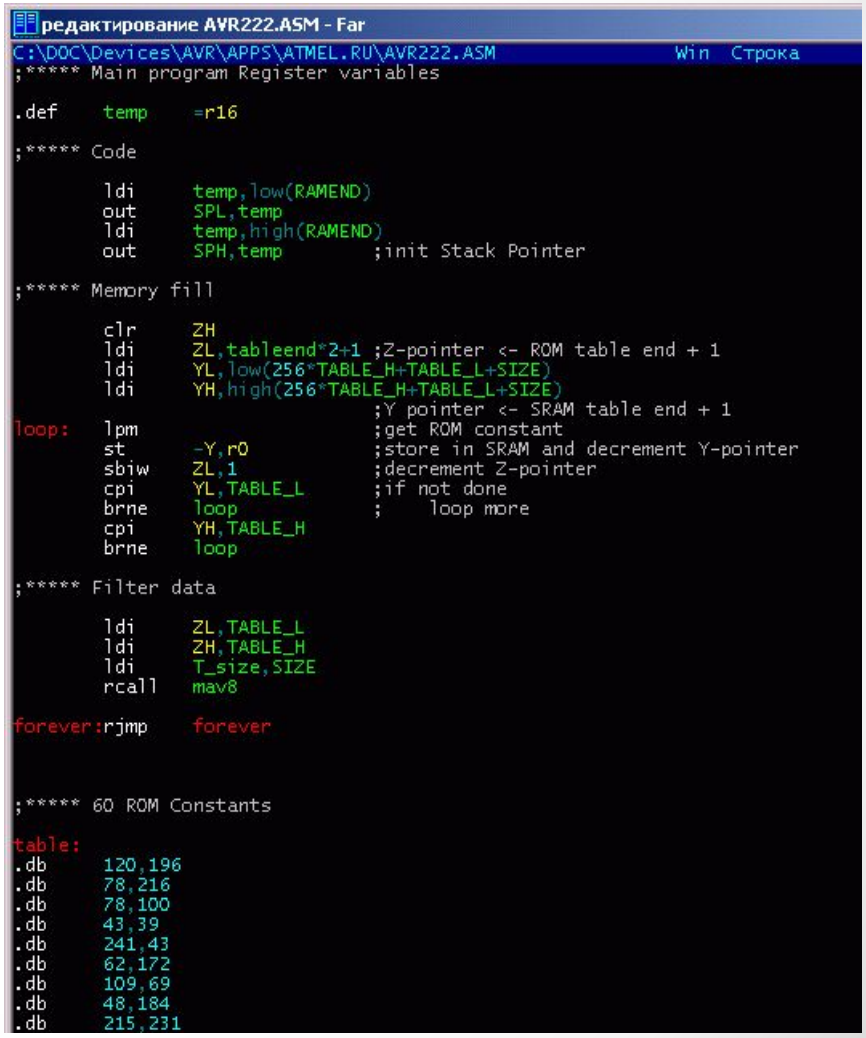
Низкоуровневый язык программирования — язык программирования, близкий к программированию непосредственно в машинных кодах используемого реального или виртуального процессора. Для обозначения машинных команд обычно применяется мнемоническое обозначение. Это позволяет запоминать команды не в виде последовательности двоичных нулей и единиц, а в виде осмысленных сокращений слов человеческого языка (обычно английских).

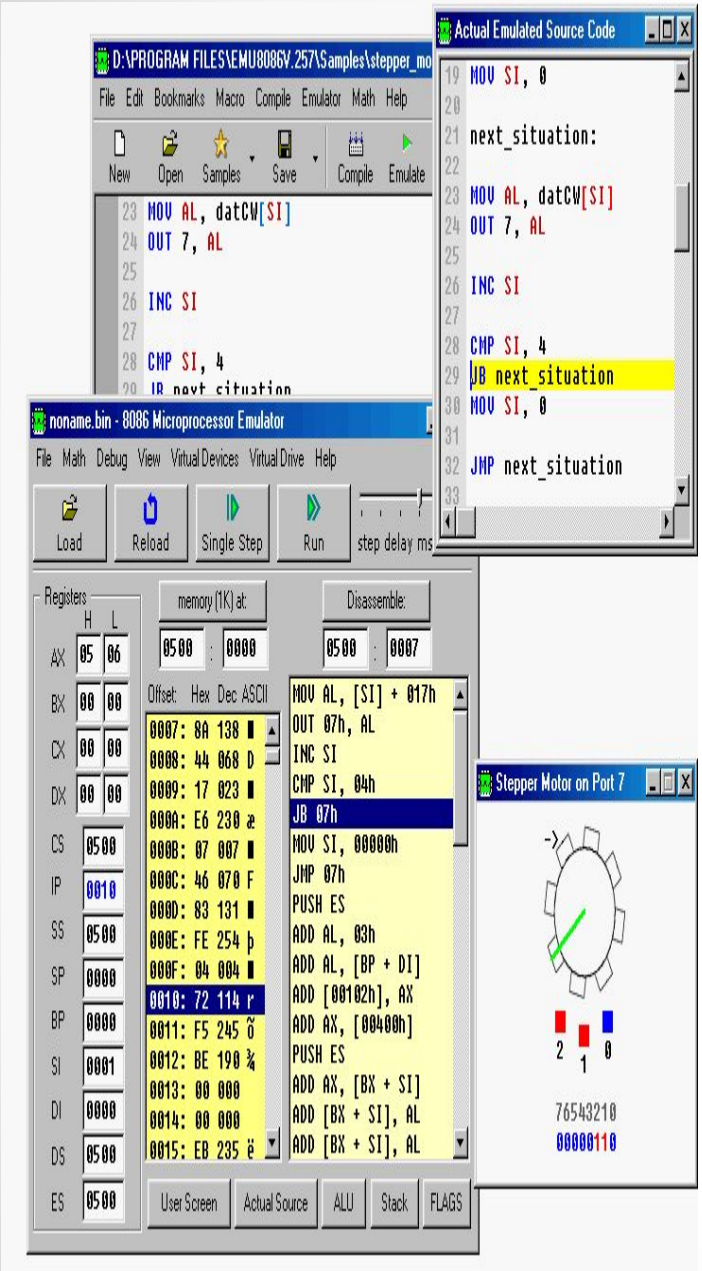


Вначале программисты использовали мнемонические обозначения при разработке программ на бумаге, а затем переводили их на машинный язык. Однако вскоре стало понятно, что такой перевод может выполнить и сама машина. В результате были разработаны программы, предназначенные для перевода записанных в мнемоническом виде программ на машинный язык. Так появились не совсем еще совершенные языки программирования второго поколения.

Языки программирования второго поколения.

Название **ассемблер** (assembler — сборщик) программы получили потому, что их назначение заключалось в сборке машинных команд из кодов команд и операндов, полученных в результате перевода мнемонических обозначений и идентификаторов. Мнемонические системы записи программ стали, в свою очередь, рассматриваться как особые языки программирования, именуемые **языками ассемблера.**





Хотя языки второго поколения имели много преимуществ по сравнению с машинными языками, они все же не могли обеспечить завершенную среду программирования. Программу на языке ассемблера достаточно сложно выполнить на другой машине, поскольку для этого ее нужно переписать с учетом новой конфигурации регистров и набора команд.

В свое время разработка языков ассемблера считалась гигантским шагом вперед в поисках более совершенных технологий программирования. Многие считали, что они представляют собой совершенно новое поколение языков программирования. Со временем языки ассемблера стали называть языками программирования второго поколения, а к первому поколению были отнесены сами машинные языки.

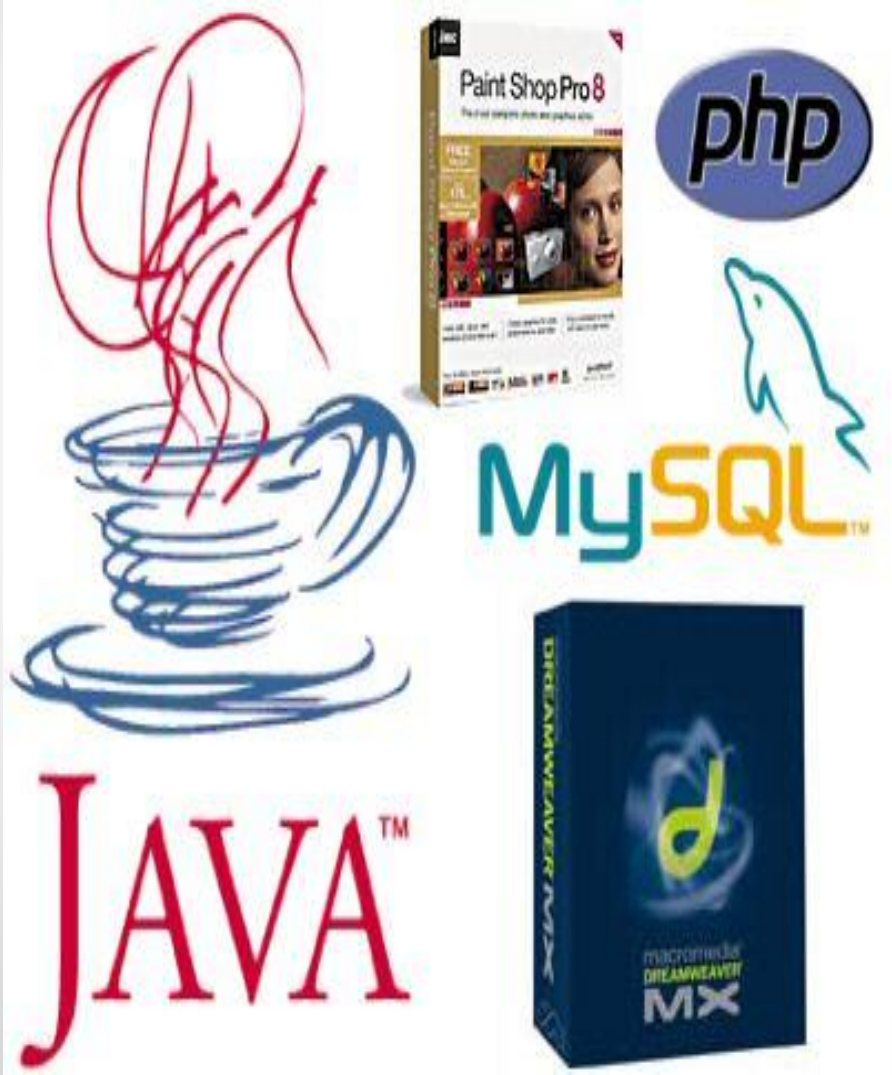
Столкнувшись с проблемами машинной зависимости программисты стали думать над ее решением. Они разработали компиляторы, которые теоретически могли работать на любом компьютере. Так появились языки программирования третьего поколения с их машинной независимостью.

В действительности не все так просто. При разработке самого компилятора приходится учитывать определенные ограничения, накладываемые той машиной, для которой он предназначен. В результате эти ограничения отражаются на языке программирования, который подлежит переводу на машинный язык.

Классификация языков программирования

- **Императивные** - языки, представляющие собой последовательность команд, в основном оперируют ветвлениями и операциями присвоения;
- **Функциональные** - языки, ориентированные на операции с функциями. Переменные и ветвления в них либо вообще отсутствуют, либо практически не используются;
- **Логические** - языки, представляющие собой совокупность предикатов и отношений типа p(x, Y) Программы на языках логического программирования выражены как формулы математической логики, а компилятор пытается получить следствия из них;
- **Объектно-ориентированные** - языки, оперирующие объектами, представляющими "вещь в себе" и обменивающимися "сообщениями";

Машинная независимость



С появлением языков программирования третьего поколения цель обеспечения машинной независимости программ была в основном достигнута. Поскольку операторы в языках третьего поколения не привязаны к особенностям какой-то конкретной машины, они легко могут быть скомпилированы на любом компьютере. Теоретически программа, написанная на языке третьего поколения, может быть выполнена на любой машине за счет использования соответствующего компилятора.

Проблема переноса программ с одной машины на другую заключается в отсутствии общей точки зрения на то, что именно считать стандартом данного языка программирования. В связи с этим Американский национальный институт стандартов (ANSI) и Международная организация по стандартизации (ISO) приняли и опубликовали стандарты для многих популярных языков программирования. В других случаях применяются неформальные стандарты, которые являются следствием популярности того или иного диалекта языка, а также желания многих разработчиков компиляторов создавать продукты, совместимые с другими, подобными им.

Парадигмы программирования

Классификация языков программирования по поколениям требует распределения их по линейной шкале в соответствии с той степенью свободы от компьютерной тарабарщины, которую данный язык предоставляет программисту. Это позволяет ему мыслить понятиями, связанными непосредственно с решаемой задачей. В действительности развитие языков программирования происходило несколько иначе. Оно протекало по разным направлениям, связанным с альтернативными подходами к процессу программирования (называемыми парадигмами программирования).

Программирование в будущем

Кто изобретет язык программирования будущего?

Одна из поразительных тенденций последнего десятилетия - это появление множества языков с отрытыми исходниками, таких как Perl, Python и Ruby. Дизайн языков захвачен хакерами. Результаты пока неоднозначны, но уже способны воодушевить. В языке Perl, например, попадаются сногсшибательные идеи. Правда, попадаются и ужасные, но так всегда и бывает с амбициозными проектами. Неизвестно, что может вырасти из Perl за сто лет, если он продолжит мутировать такими темпами.