**Язык программирования** – формальная знаковая система, предназначенная для записи компьютерных программ.

Язык программирования определяет набор лексических, синтаксических и семантических правил, задающих вид программы и действия, которые будут выполняться под её управлением.

Язык программирования представляется в виде набора спецификаций, определяющих его синтаксис и семантику

**Классы языков программировани**

(я вообще в этом не бумбум)

(https://spravochnick.ru/programmirovanie/yazyki\_programmirovaniya/klassifikaciya\_yazykov\_programmirovaniya/)

(я так полагаю у нас всех примерно такой)  
**Процедурные языки**

Процедурные языки являются языками высокого уровня, в которых используется метод разбиения программ на отдельные связанные между собой модули – подпрограммы (процедуры и функции). Компоненты языка состоят из последовательности операторов, которые используют библиотечные процедуры и функции

**Транслятор**

Трансля́тор — программа или техническое средство, выполняющее трансляцию программы. Трансля́ция програ́ммы — преобразование программы, представленной на одном из языков рограммирования, в программу, написанную на другом языке.

**компилятор**

компьютерная программа, которая переводит компьютерный код, написанный на одном языке программирования (исходный язык), на другой язык (целевой язык).

**Интерпретация**

построчный анализ, обработка и выполнение исходного кода программы или запроса, в отличие от компиляции, где весь текст программы, перед запуском анализируется и транслируется в машинный или байт-код без её выполнения.

**компилируемый или интерпретируемый**

Компилятор анализирует программу целиком, превращает её в машинный код и сохраняет для последующего выполнения.

Интерпретатор разбирает и выполняет программу построчно в режиме реального времени.

**Лексический анализатор**

Лексический анализатор (или сканер ) — это часть компилятора, которая читает исходную программу и выделяет в ее тексте лексемы входного языка. На вход лексиче- ского анализатора поступает текст исходной программы.

**Синтаксический анализатор**

Синтаксический анализатор, — часть программы, преобразующей входные данные (как правило, текст) в некий структурированный формат, нужный для задач последующего их (данных) анализа и использования.

**Семантический анализатор**

Семантический анализатор использует синтаксическое дерево и информацию из таблицы идентификаторов для проверки входной программы на семантическую согласованность с определением языка программирования.

(проверка смысловой правильности конструкций языка программирования.)

**Препроцессор**

Препроцессор — это специальная программа, являющаяся частью компилятора. Она предназначена для предварительной обработки текста программы. Препроцессор позволяет включать в текст программы файлы и вводить макроопределения. Работа препроцессора осуществляется с помощью специальных директив.

**форма Бэкуса-Наура (БНФ)**

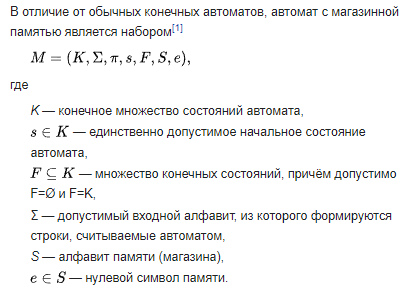
[формальная система](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0" \o "Формальная система) описания [синтаксиса](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81" \o "Синтаксис), в которой одни [синтаксические категории](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BA%D0%B0%D1%82%D0%B5%D0%B3%D0%BE%D1%80%D0%B8%D1%8F&action=edit&redlink=1" \o "Синтаксическая категория (страница отсутствует)) последовательно определяются через другие категории. БНФ используется для описания [контекстно-свободных](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BD%D0%BE-%D1%81%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0" \o "Контекстно-свободная грамматика) [формальных грамматик](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0" \o "Формальная грамматика)

**Сеппаратор**

разделитель операторов [предложений языка программирования] символ, используемый в языке программирования для отделения одного оператора от другого.

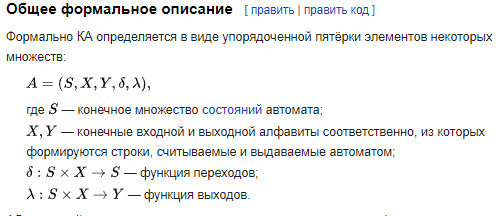
# **Автомат с магазинной памятью**

автомат с магазинной памятью — это [конечный автомат](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D0%B5%D1%87%D0%BD%D1%8B%D0%B9_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82" \o "Конечный автомат), который использует [стек](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA" \o "Стек) для хранения состояний.



# **Конечный автомат**

[математическая абстракция](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%B0%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%86%D0%B8%D1%8F" \o "Математическая абстракция), [модель](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C" \o "Модель) [дискретного устройства](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%BE%D0%B5_%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE" \o "Дискретное устройство), имеющего один вход, один выход и в каждый момент времени находящегося в одном [состоянии](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D0%B5" \o "Состояние) из множества возможных.



**Литерал**

Литерал, или безымянная константа — запись в исходном коде компьютерной программы, представляющая собой фиксированное значение. Литералами также называют представление значения некоторого типа данных

**Rvalue**

rvalue определено путём исключения, говоря, что любое выражение является либо lvalue, либо rvalue. Таким образом из определения lvalue следует, что rvalue — это выражение, которое не представляет собой объект, который занимает идентифицируемое место в памяти

**Пользовательский тип данных**

Пользовательские типы данных — это типы данных, которые могут быть созданы пользователем на основе того, что доступно в языке. Пользователи хотели иметь возможность составлять данные так, как они сами того хотят. Хотели, хотят, и наверняка будут хотеть. Всё больше, всё разнообразней и сильнее.

**Зарезервированные идентификаторы**

Имя которое уже используется, которое присваивается переменной, константе, функции

**Ключевые слова**

Зарезерви́рованное сло́во (или ключево́е сло́во) — в языках программирования слово, имеющее специальное значение

**Объектами-инициализаторами**

Инициализаторы объектов позволяют присваивать значения всем доступным полям и свойствам объекта во время создания без вызова конструктора, за которым следуют строки операторов присваивания. Синтаксис инициализатора объекта позволяет задавать аргументы конструктора или опускать их (и синтаксис в скобках). В приведенном ниже примере демонстрируется использование инициализатора объекта с именованным типом Cat и вызов конструктора без параметров. Обратите внимание на использование в классе Cat автоматически внедренных свойств.

**Обратная польская запись**

форма записи математических и логических выражений, в которой операнды расположены перед знаками операций. Также именуется как обратная бесскобочная запись, постфиксная нотация, бесскобочная символика Лукасевича, польская инверсная запись, ПОЛИЗ

**Прямая польская запись**

По́льская нота́ция (за́пись), также известна как пре́фиксная нота́ция (запись), это форма записи логических, арифметических и алгебраических выражений. Характерная черта такой записи — оператор располагается слева от операндов.

**Stdcall**

stdcall или winapi — соглашение о вызовах, применяемое в ОС Windows для вызова функций WinAPI. Аргументы функций передаются через стек, справа налево. Очистку стека производит вызываемая подпрограмма.

**Иерархия хомского**

классификация [формальных языков](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA" \o "Формальный язык) и [формальных грамматик](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0" \o "Формальная грамматика), согласно которой они делятся на 4 типа по их условной сложности

Иерархия Хомского — это общепринятое деление формальных языков на 4 уровня. Чем выше номер типа, тем более скудный язык и тем легче его разбирать программно:

* Тип 0: неограниченные
  + правила замены символов ничем не ограничены, что делает машинный анализ таких текстов невозможным
* Тип 1: контекстно-зависимые
  + правила замены символов зависят от контекста
  + в этом примере на C++ без контекста неясно, что это — сравнение двух переменных или специализация шаблона: vec < a > b.
  + в этом примере неясно, объявлена ли функция “getSize”: getSize(sprite)
* Тип 2: контекстно-свободные
  + правила описывают замену одного нетерминала на цепочку нетерминалов и терминалов (возможно, пустую), т.е. способ замены каждого нетерминала на другие символы не зависит от контекста
  + формально: в левой части правила может быть только нетерминал, A → β, где “A” — нетерминал, “β” — цепочка нетерминалов и терминалов
  + например, в языках программирования символ “присваивание” раскрывается однозначно независимо от того, что окружает присваивание
* Тип 3: регулярные
  + праворегулярные грамматики могут содержать три вида правил: B → α, B → αC, B → ε, где “ε” — пустое множество, “B” и “С” — нетерминалы, и “α” — терминал
  + леворегулярные грамматики могут содержать три вида правил: A → α, A → Bα, A → ε, где “ε” — пустое множество, “A” и “B” — нетерминалы, и “α” — терминальный символ

https://ps-group.github.io/compilers/grammars

<https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%8F_%D0%A5%D0%BE%D0%BC%D1%81%D0%BA%D0%BE%D0%B3%D0%BE>

## [Контекстно-свободные грамматики](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%8F_%D0%A5%D0%BE%D0%BC%D1%81%D0%BA%D0%BE%D0%B3%D0%BE)

[Контекстно-свободная грамматика, не являющаяся регулярной, не имеет ни эквивалентного регулярного выражения, ни эквивалентного детерминированного конечного автомата. При попытке сформировать ДКА количество его состояний будет расти бесконечно, и так же бесконечно будет возрастать глубина регулярного выражения. В контекстно-свободной грамматике правила для нетерминалов определяются без окружающего контекста, т.е. A → β, где “A” — нетерминал, “β” — цепочка нетерминалов и терминалов. Примеры таких грамматик:](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%8F_%D0%A5%D0%BE%D0%BC%D1%81%D0%BA%D0%BE%D0%B3%D0%BE)

* [арифметические выражения с разным приоритетом операторов](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%8F_%D0%A5%D0%BE%D0%BC%D1%81%D0%BA%D0%BE%D0%B3%D0%BE)
* [выражения со скобками, в которых глубина вложенности скобок может быть любой](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%8F_%D0%A5%D0%BE%D0%BC%D1%81%D0%BA%D0%BE%D0%B3%D0%BE)

## [Регулярные грамматики](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%8F_%D0%A5%D0%BE%D0%BC%D1%81%D0%BA%D0%BE%D0%B3%D0%BE)

[Любая регулярная грамматика имеет эквивалентное регулярное выражение, а также эквивалентный детерминированный конечный автомат (ДКА). Это всё три формы одной и той же сущности. Разобрать строку по правилам регулярной грамматики можно за один проход из начала в конец строки без дополнительной памяти с помощью ДКА.](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%8F_%D0%A5%D0%BE%D0%BC%D1%81%D0%BA%D0%BE%D0%B3%D0%BE)

**[Грамматикой в нормальной форме Грейбах](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%8F_%D0%A5%D0%BE%D0%BC%D1%81%D0%BA%D0%BE%D0%B3%D0%BE)** [называется](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%8F_%D0%A5%D0%BE%D0%BC%D1%81%D0%BA%D0%BE%D0%B3%D0%BE)[контекстно-свободная грамматика](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9A%D0%BE%D0%BD%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BD%D0%BE-%D1%81%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D1%8B%D0%B5_%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B8,_%D0%B2%D1%8B%D0%B2%D0%BE%D0%B4,_%D0%BB%D0%B5%D0%B2%D0%BE-_%D0%B8_%D0%BF%D1%80%D0%B0%D0%B2%D0%BE%D1%81%D1%82%D0%BE%D1%80%D0%BE%D0%BD%D0%BD%D0%B8%D0%B9_%D0%B2%D1%8B%D0%B2%D0%BE%D0%B4,_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D1%80%D0%B0%D0%B7%D0%B1%D0%BE%D1%80%D0%B0" \o "Контекстно-свободные грамматики, вывод, лево- и правосторонний вывод, дерево разбора)[, в которой могут содержаться только правила одного из следующих типов](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%8F_%D0%A5%D0%BE%D0%BC%D1%81%D0%BA%D0%BE%D0%B3%D0%BE)

[A→aγ](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%8F_%D0%A5%D0%BE%D0%BC%D1%81%D0%BA%D0%BE%D0%B3%D0%BE)

[S→ε](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%8F_%D0%A5%D0%BE%D0%BC%D1%81%D0%BA%D0%BE%D0%B3%D0%BE)

[где a — терминал, A — нетерминал (возможно, стартовый), S — стартовый нетерминал (причём он не должен встречаться в правых частях правил), ε — пустая строка, γ — строка из не более, чем двух нетерминалов.](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%8F_%D0%A5%D0%BE%D0%BC%D1%81%D0%BA%D0%BE%D0%B3%D0%BE)

**[Грамматикой в ослабленной нормальной форме](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%8F_%D0%A5%D0%BE%D0%BC%D1%81%D0%BA%D0%BE%D0%B3%D0%BE)** [Грейбах (англ. Greibach weak normal form) называется](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%8F_%D0%A5%D0%BE%D0%BC%D1%81%D0%BA%D0%BE%D0%B3%D0%BE)[контекстно-свободная грамматика](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9A%D0%BE%D0%BD%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BD%D0%BE-%D1%81%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D1%8B%D0%B5_%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B8,_%D0%B2%D1%8B%D0%B2%D0%BE%D0%B4,_%D0%BB%D0%B5%D0%B2%D0%BE-_%D0%B8_%D0%BF%D1%80%D0%B0%D0%B2%D0%BE%D1%81%D1%82%D0%BE%D1%80%D0%BE%D0%BD%D0%BD%D0%B8%D0%B9_%D0%B2%D1%8B%D0%B2%D0%BE%D0%B4,_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D1%80%D0%B0%D0%B7%D0%B1%D0%BE%D1%80%D0%B0" \o "Контекстно-свободные грамматики, вывод, лево- и правосторонний вывод, дерево разбора)[, в которой могут содержаться только правила одного из следующих типов:](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%8F_%D0%A5%D0%BE%D0%BC%D1%81%D0%BA%D0%BE%D0%B3%D0%BE)

[A→aγA→aγ](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%8F_%D0%A5%D0%BE%D0%BC%D1%81%D0%BA%D0%BE%D0%B3%D0%BE)

[S→εS→ε](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%8F_%D0%A5%D0%BE%D0%BC%D1%81%D0%BA%D0%BE%D0%B3%D0%BE)

[где aa — терминал, AA — нетерминал (возможно, стартовый), SS — стартовый нетерминал (причём он не должен встречаться в правых частях правил), εε — пустая строка, γγ — строка из произвольного числа терминалов и нетерминалов.](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%8F_%D0%A5%D0%BE%D0%BC%D1%81%D0%BA%D0%BE%D0%B3%D0%BE)

**Формальный язык** – множество символьных

цепочек

**Формальная грамматика** – набор правил, с

помощью которых порождаются цепочки

формального языка.

Формальные грамматики можно преобразовать

в конечные распознаватели и обрабатывающие

автоматы, которые распознают/транслируют

соответствующие множества цепочек

**регулярная и контекстно-свободная грамматика**

Для задания КС-грамматики необходимо:

• конечное множество терминалов – символов, не

требующих дополнительных определений

• конечное множество нетерминалов – символов,

которые определяются через терминалы и другие

нетерминалы

• конечное множество правил –

определений вида <А> , где

левая часть <А> – нетерминал

правая часть – конечная, возможно пустая,

цепочка терминалов и нетерминалов

• начальный нетерминал

**Лево- и правосторонние выводы**

 **Лево-/правосторонний вывод** –

последовательность подстановок, в которой на

каждом шаге заменяется самый левый/правый

нетерминал.

 Каждому дереву вывода соответствует

единственный левосторонний и единственный

правосторонний выводы.

**Для задания КС-грамматики** нужно определить

конечное множество терминалов, конечное

множество нетерминалов, конечное множество

правил и начальный нетерминал.

**Assembler**

**Push**

Команда PUSH уменьшает значение регистра стека на размер операнда (2 или 4) и копирует содержимое операнда в память по адресу SS:SP. Начиная с 80186 процессора появилась возможность помещать в стек непосредственные значения. При этом, если в стек заносится байт, то он расширяется до слова с сохранением знака

**Pop**

Команда POP копирует содержимое ячейки памяти по адресу SS:SP в операнд и увеличивает значение регистра стека на размер операнда (2 или 4). Если операнд использует регистр SP для косвенной адресации, команда POP вычисляет адрес уже после того, как она увеличивает SP.

**Mov**

Команда mov (MOVe operand) Назначение: пересылка данных между регистрами или регистрами и памятью.

**Cmp**

Инструкция CMP (от англ. «COMPARE») сравнивает два операнда. Фактически, она выполняет операцию вычитания между двумя операндами для проверки того, равны ли эти операнды или нет. Используется вместе с инструкцией условного прыжка.

**Jmp**

Команда JMP - это команда безусловного перехода в Ассемблере. Выполняет, соответственно, безусловный переход в указанное место. МЕТКОЙ может быть один из следующих: Идентификатор метки в исходном коде

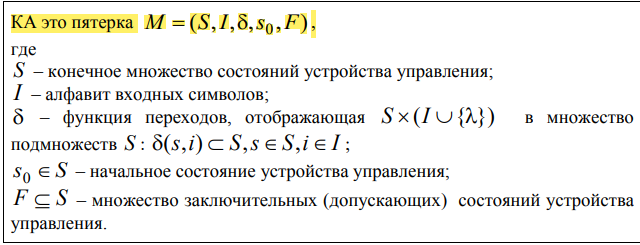
**Call**

Как уже было сказано, команда CALL передаёт управление по адресу, который передаётся в команду в качестве параметра. При этом процессор начинает выполнять команду, расположенную по этому адресу. Если операндом команды CALL является регистр или переменная, то её значение рассматривается как абсолютное смещение.

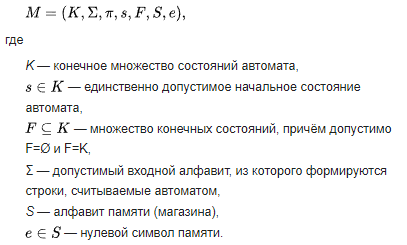
**Jg**

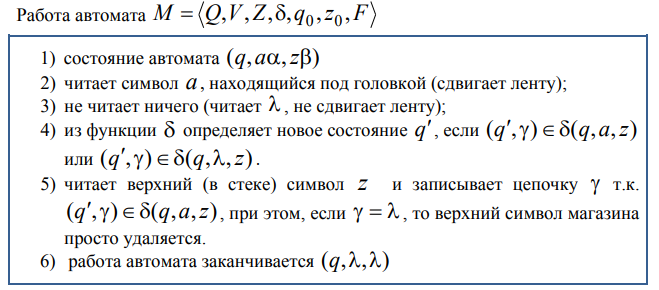
В конце, как всегда, расскажу, почему эта команда ассемблера называется JG. Буква J - это первая буква слова слова JUMP (прыжок, переход). А буква G - это первая буква слова GREATER (больше). Таким образом набор слов, от которых взяты первые буквы имени команды JG, можно перевести как “переход, если больше”.

Конечный автомат



Магазинный автомат





G = T N P S

P - правила граматики

S - начальный символ

T -терминальынй

N - нетерминальынй

G - Граматика грейбах