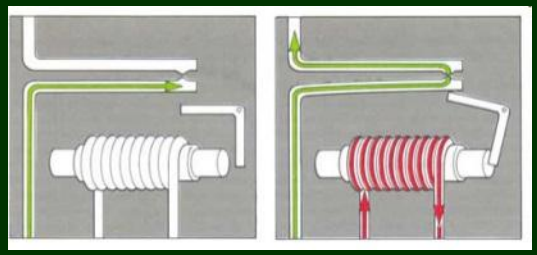
Лекция 2

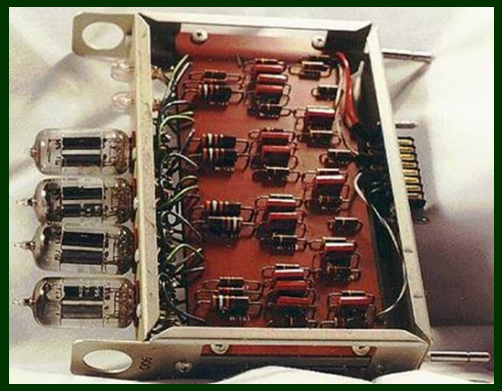
Структура персонального компьютера. Процессор

Эволюция технологий изготовления процессора

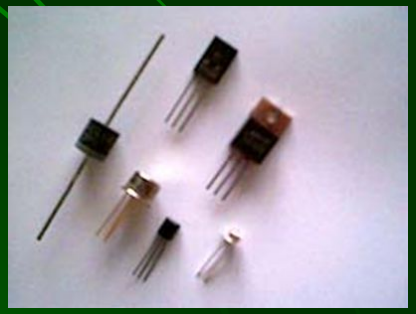
- Электромеханическое реле

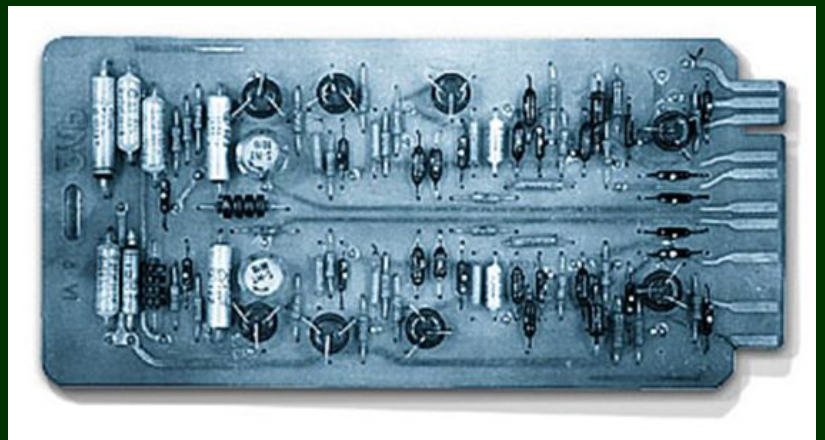


- Вакуумные лампы и ячейки на лампах

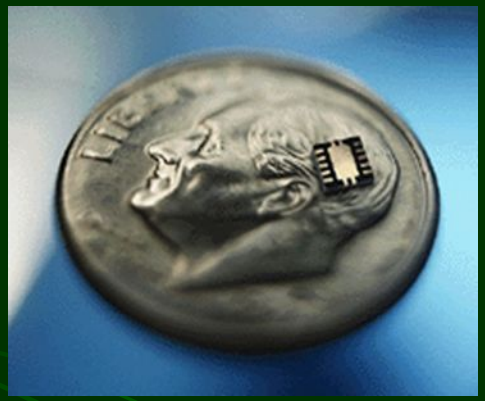


- Транзисторы





Эволюция технологий изготовления процессора: микросхемы



Микропроцессор Intel 4004

- 1971 год
- первый в мире коммерчески доступный однокристальный микропроцессор
- стоимость 200 долларов
- на одном кристалле все функции процессора большой ЭВМ
- 60 000 (в среднем, максимально до 93 000) инструкций в секунду
- Количество транзисторов: 2250

В настоящее время

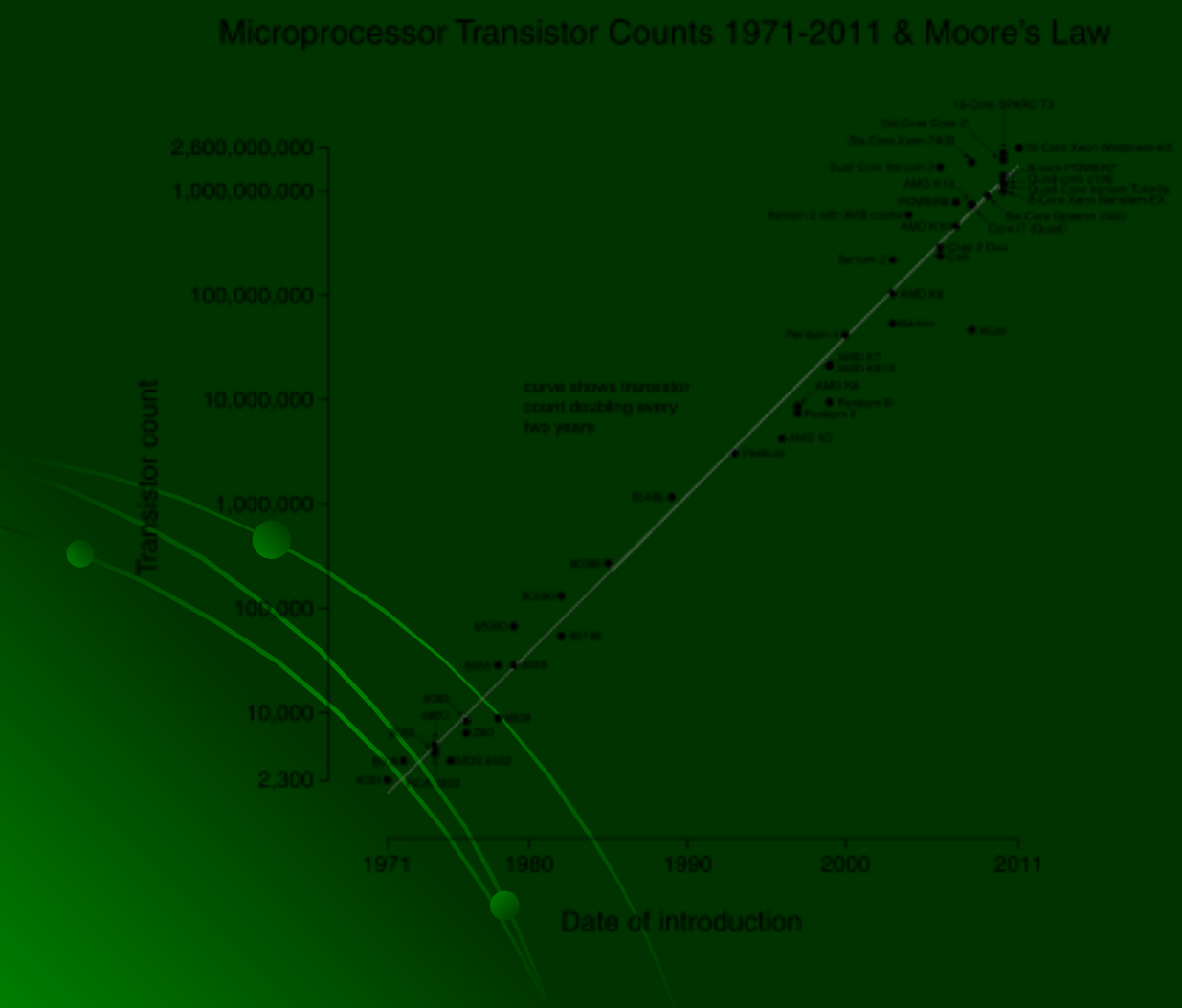
- **Intel Core i3 2010**

- 2011 год
- 995 000 000 транзисторов
- ~145 000 000 000 операций с плавающей точкой в секунду
- 2013 год: Ivy Bridge 1,4 млрд. транзисторов на площади кристалла 160 мм².

- Processor Number i7-5960X
- Intel® Smart Cache 20 MB
- Instruction Set 64-bit
- Lithography 22 nm
- # of Cores 8
- Processor Base Frequency 3 GHz
- Max Turbo Frequency 3.5 GHz
- TDP 140 W (Thermal Design Power)
- Max Memory Size (dependent on memory type) 64 GB
- Memory Types DDR4-1333/1600/2133
- Max # of Memory Channels 4
- Max Memory Bandwidth 68 GB/s
- Processor Graphics ‡ None
- 2,2 млрд транзисторов
- ~ 1000 долларов

Закон Мура

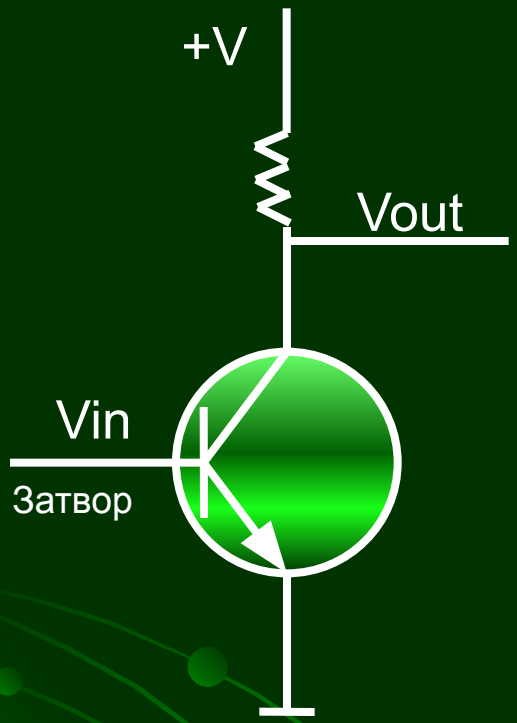
- Удвоение числа транзисторов каждые 2 года



Журнал «В мире науки» (1983, № 08)

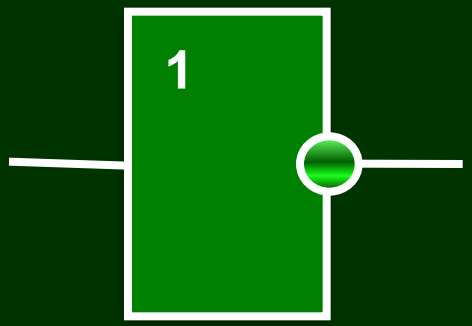
Если бы авиапромышленность в последние 25 лет развивалась столь же стремительно, как промышленность средств вычислительной техники, то сейчас самолёт Boeing 767 стоил бы 500 долл. и совершал облёт земного шара за 20 минут, затрачивая при этом пять галлонов (~18,9 л) топлива.

Схема работы транзистора



- Напряжение на базе ниже критического – транзистор действует как большое сопротивление; выходное напряжение высоко
- Напряжение на базе выше критического - транзистор открывается; выходное напряжение падает

Таблица истинности	
Vin	Vout
0	1
1	0



Инвертор

Построение логических элементов на транзисторах

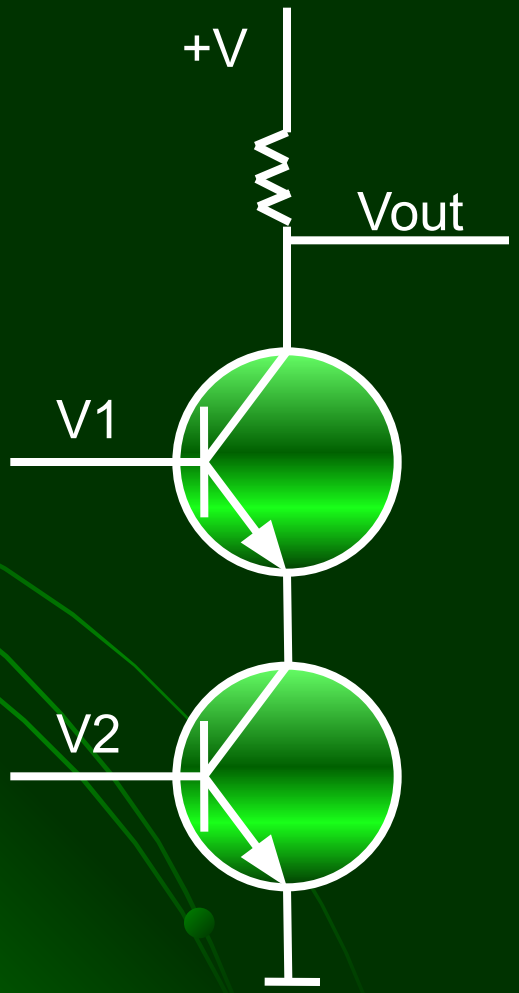
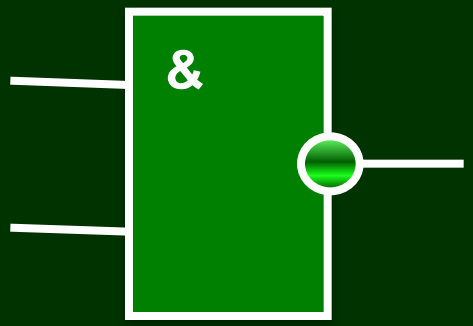
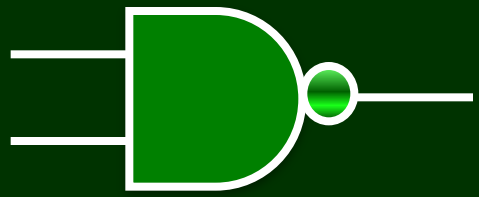


Таблица истинности		
V1	V2	Vout
0	0	1
0	1	1
1	0	1
1	1	0





И-НЕ

Построение элемента И

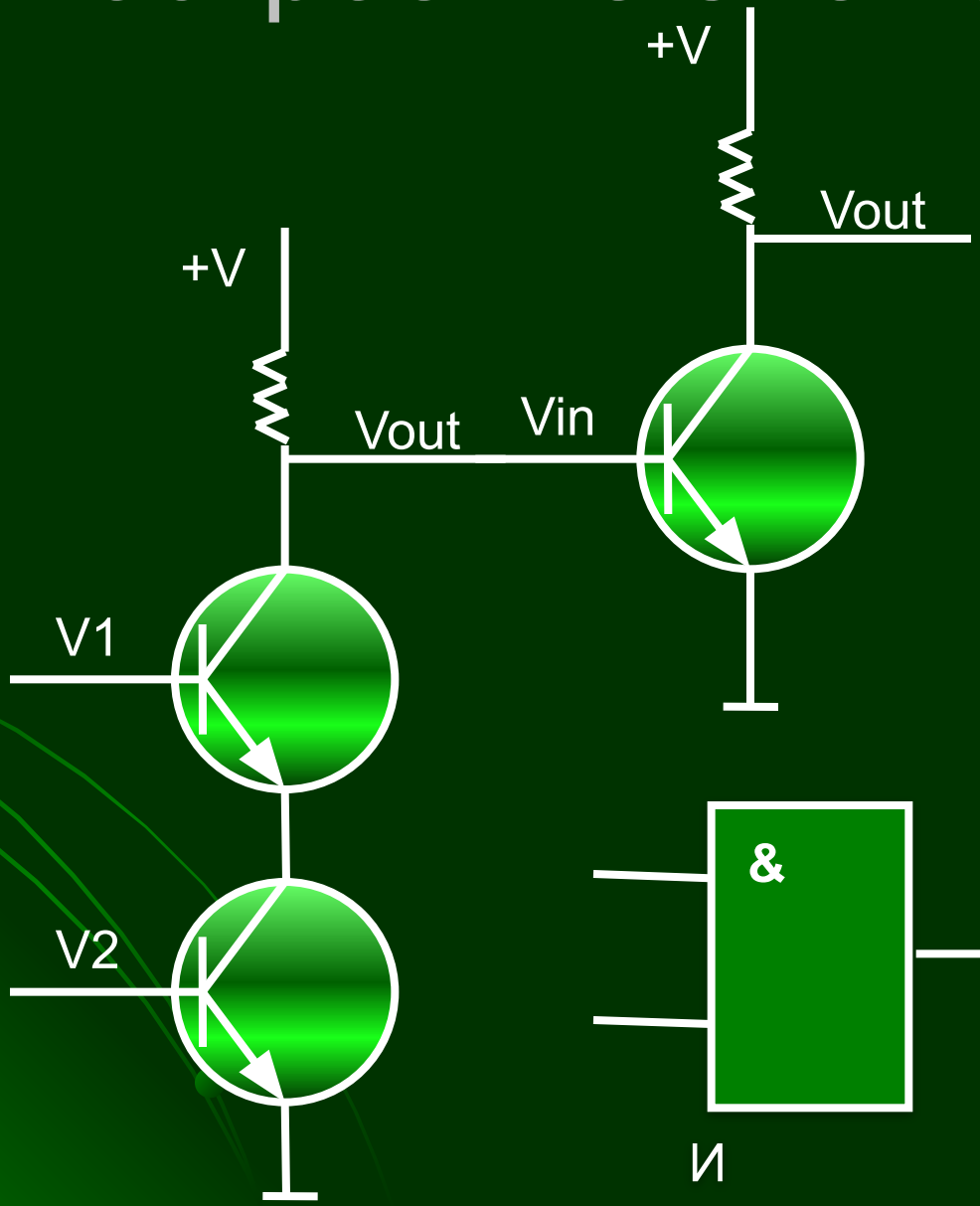
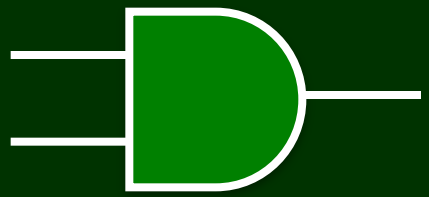


Таблица истинности		
V1	V2	Vout
0	0	0
0	1	0
1	0	0
1	1	1

И



Построение логических элементов на транзисторах

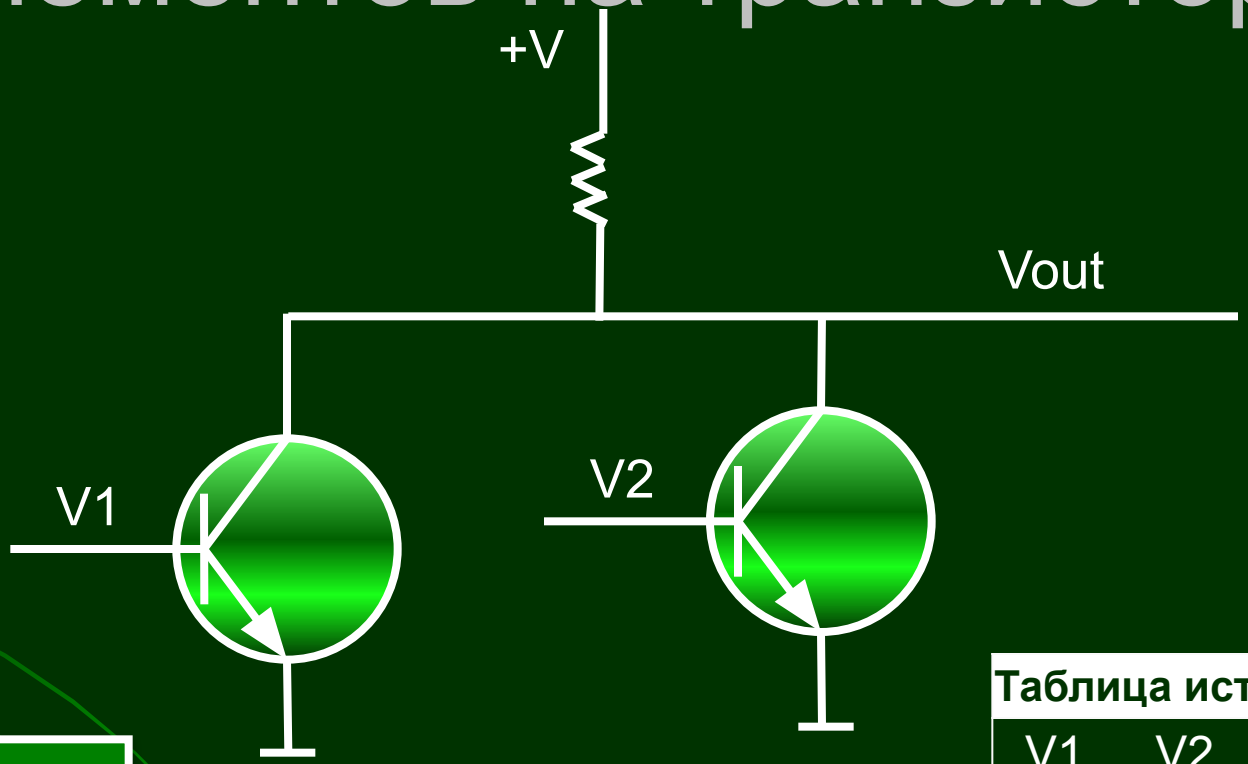
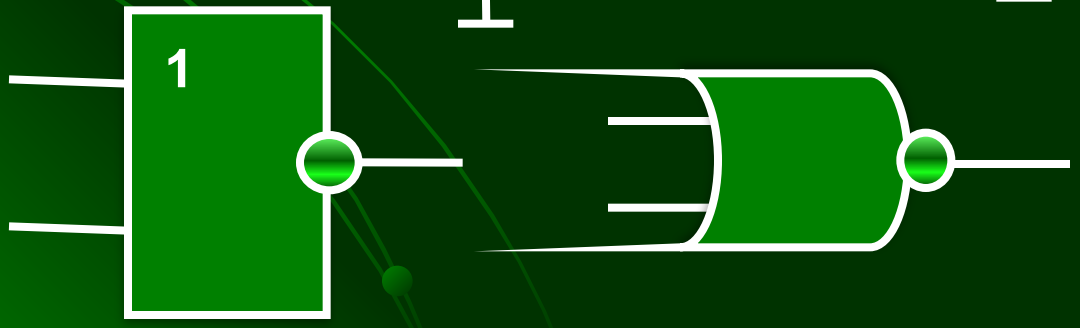


Таблица истинности		
V1	V2	Vout
0	0	1
0	1	0
1	0	0
1	1	0



ИЛИ-НЕ

Построение элемента ИЛИ

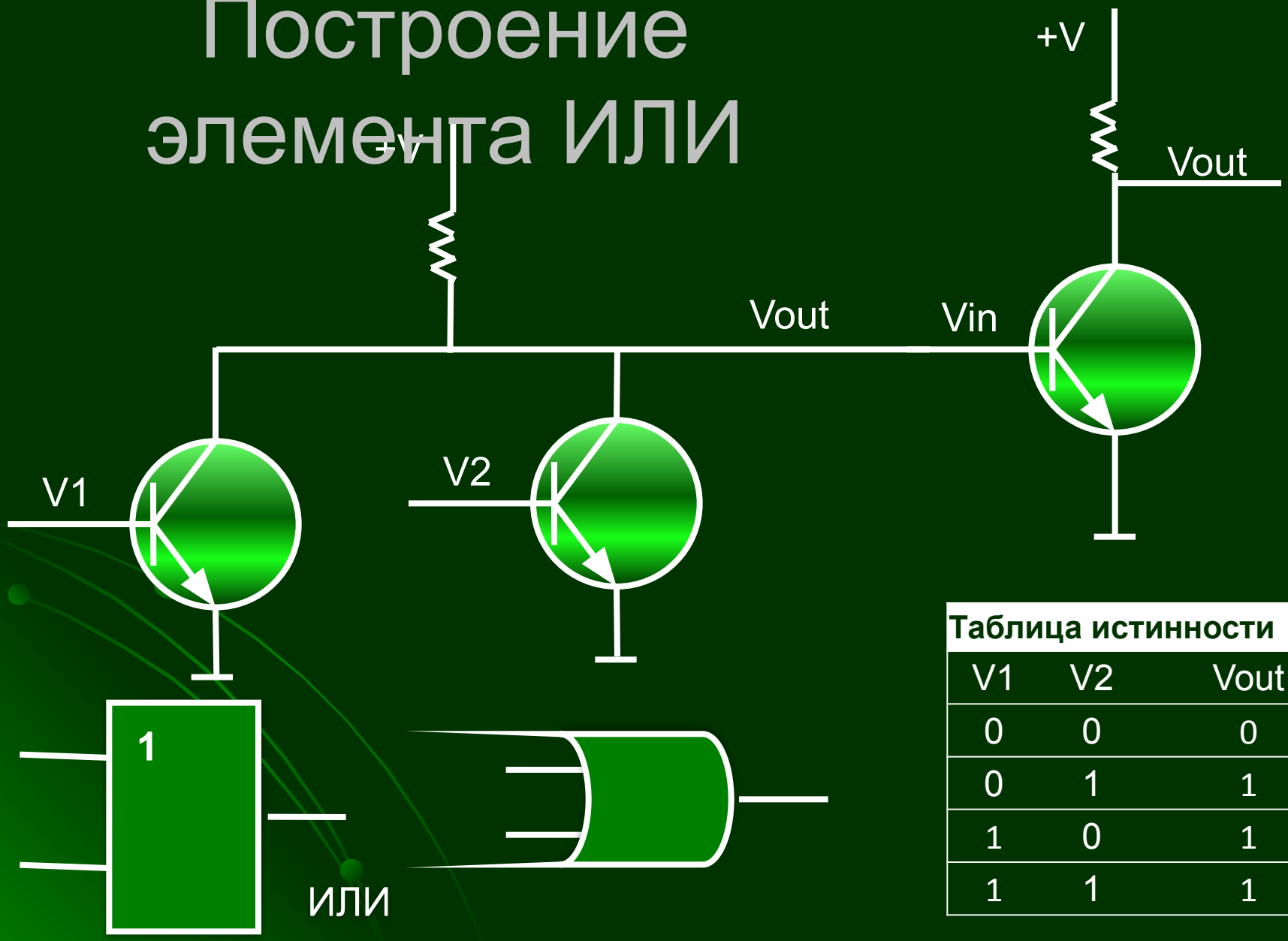


Таблица истинности		
V1	V2	Vout
0	0	0
0	1	1
1	0	1
1	1	1

Логические функции

Таблица истинности																
V1	V2	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

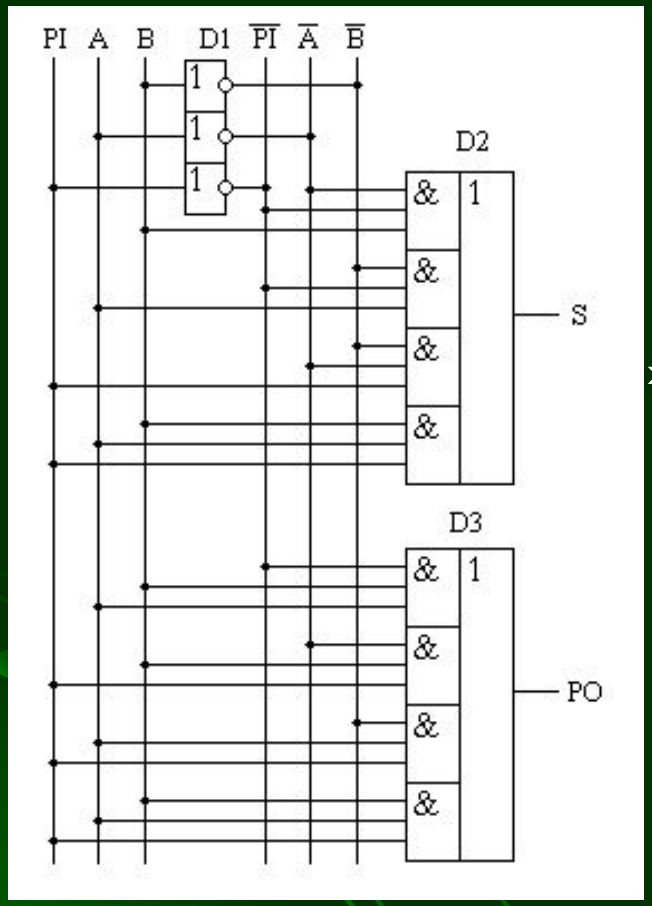
- Любую логическую функцию можно построить на базисных логических элементах
- Все операции являются результатом работы логических функций

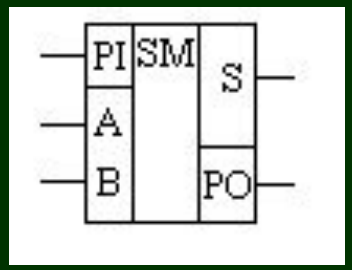
Пример: Таблица истинности сумматора

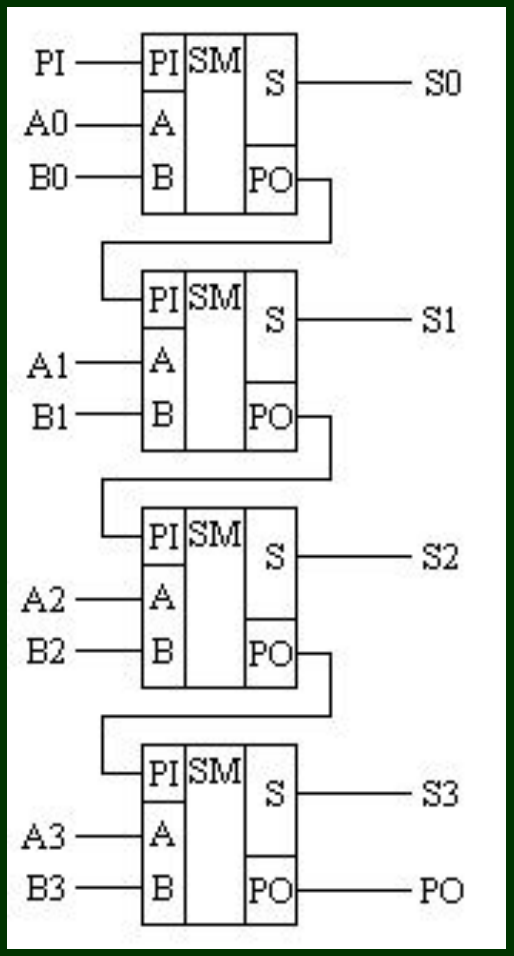
Таблица истинности			
V1	V2	Сумма	Перенос
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Таблица истинности				
V1	V2	Перенос1	Сумма	Перенос2
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

Пример: Схема сумматора



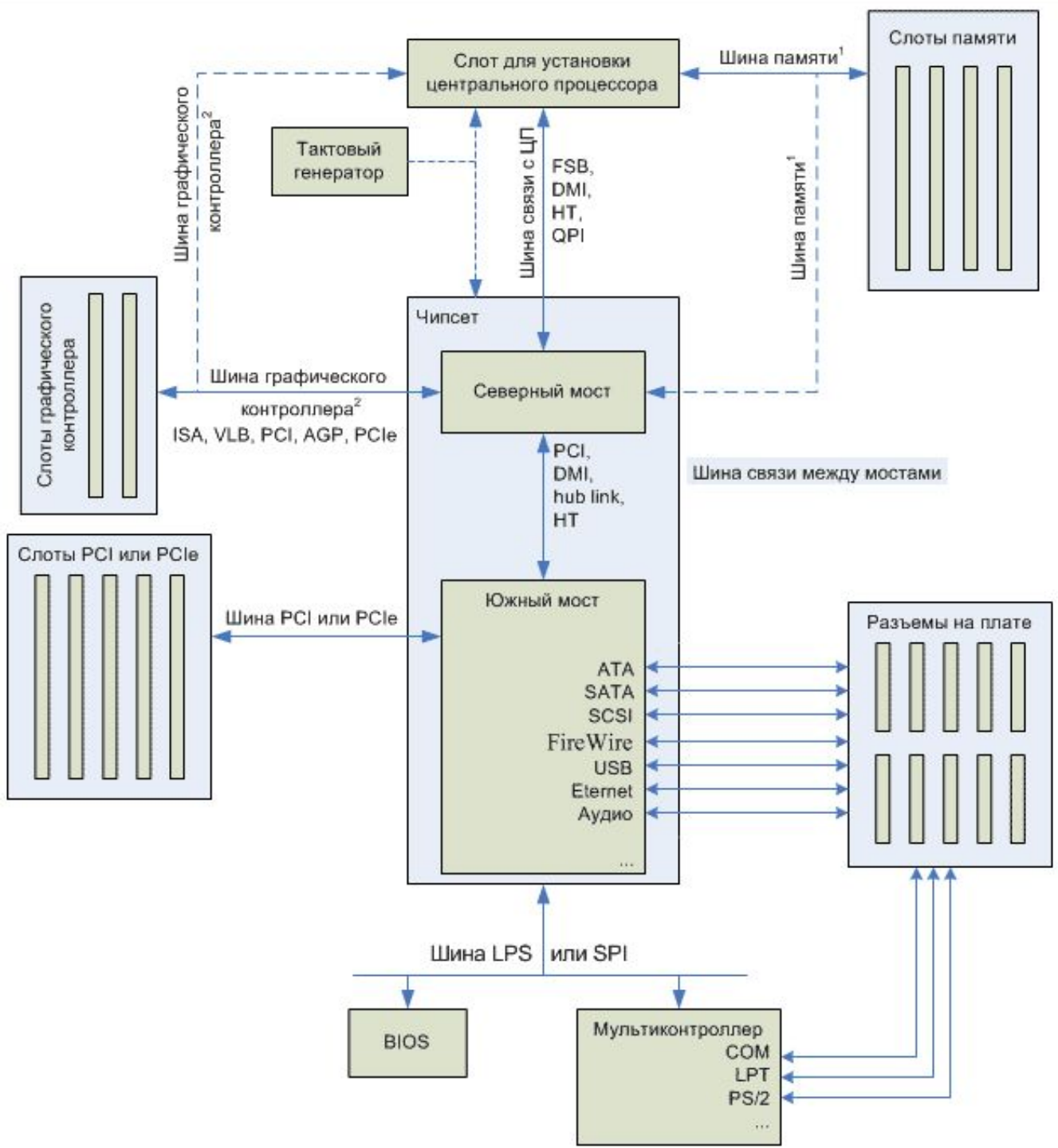




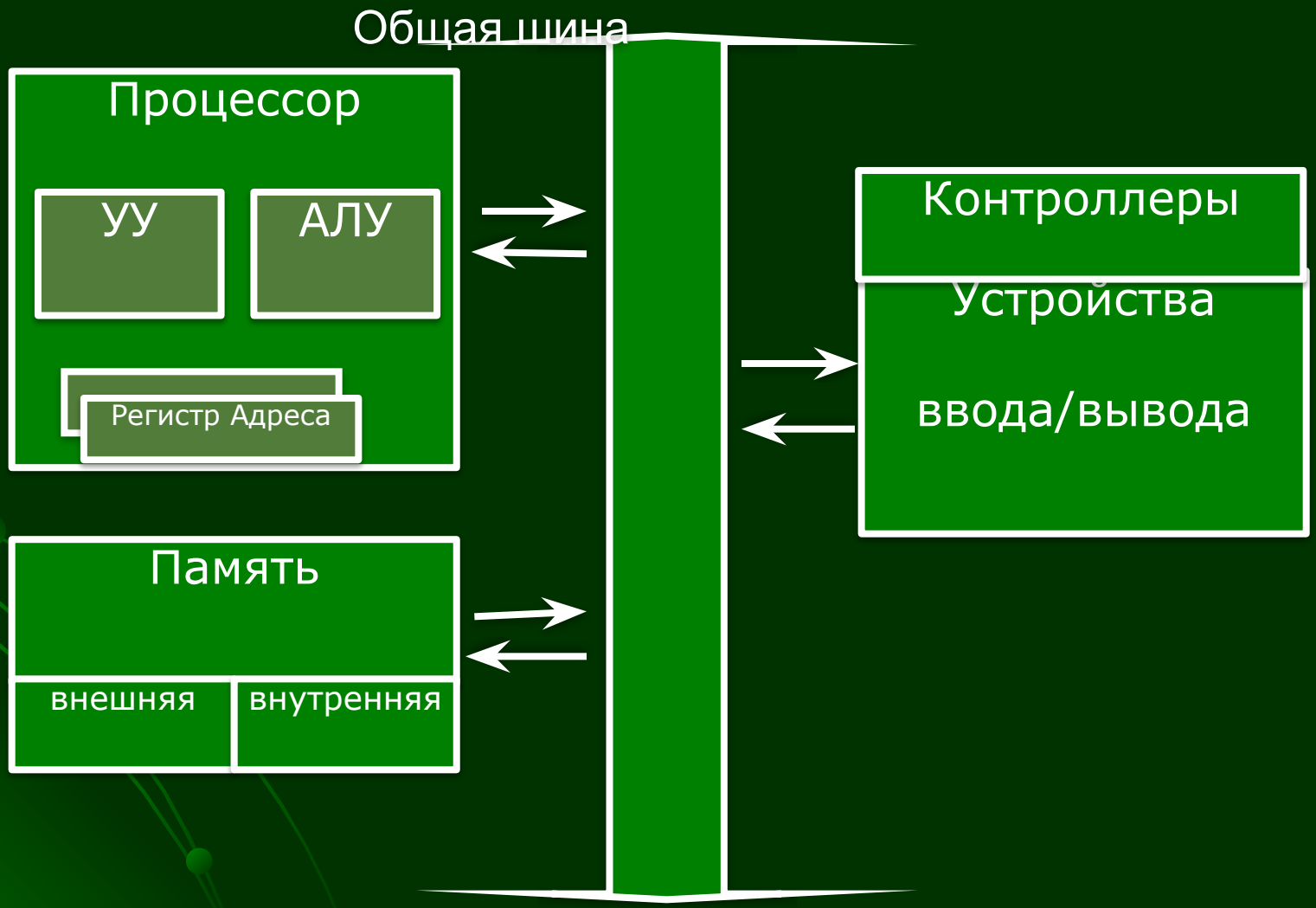
Принципиальная схема, реализующая таблицу истинности полного двоичного одноразрядного сумматора

Принципиальная схема многоразрядного двоичного сумматора

Схема материнской платы ПК



Предельно-упрощенная схема ПК



Алгоритм работы процессора: обращение в память за командой

Формат команды процессора

N (1,2,…) байт (в зависимости от архитектуры)

Числовая комбинация, определяющая действия процессора

Данные для команды или указание, откуда взять данные для команды

Алгоритм работы процессора: обращение в память за данными

Алгоритм работы процессора: обработка данных в АЛУ

Алгоритм работы процессора: отправка данных в память

Алгоритм работы процессора: определение адреса команды

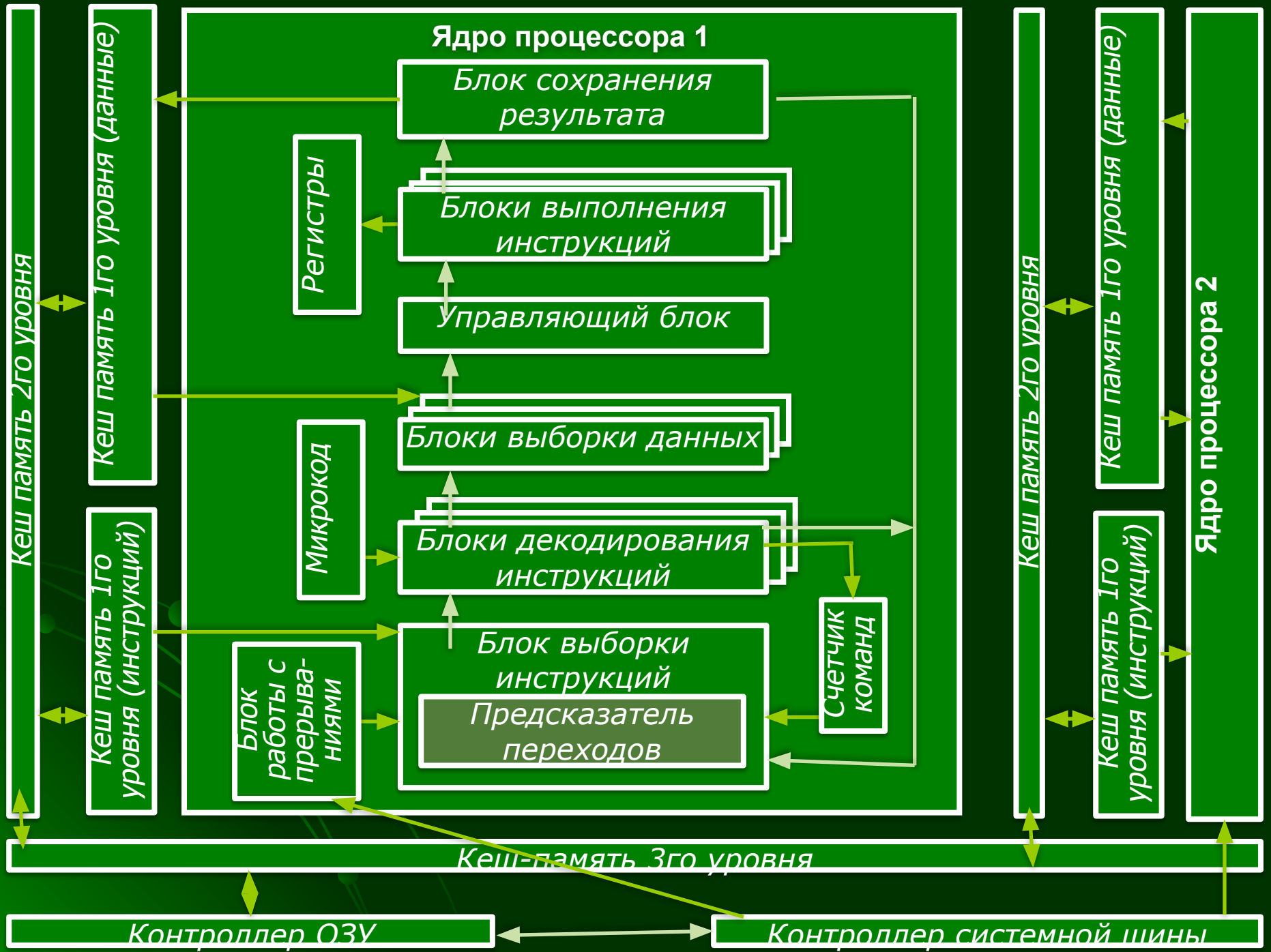
В счетчике команд определяется адрес следующей команды

Алгоритм работы процессора

- Выбор команды
- Дешифрация
- Запрос операндов
- Выполнение команды с получением результата и/или формированием признаков
- Запись результата
- Увеличение (изменение) счетчика команд

Упрощенная структурная схема процессора

(следующий слайд)


Ядро процессора 1
Блок сохранения результата
Регистры
Блоки выполнения инструкций
Управляющий блок
Блоки выборки данных
Микрокод
Блоки декодирования инструкций
Блок выборки инструкций
Предсказатель переходов
Счетчик команд
Блок работы с прерыва-ниями
Кеш память 2го уровня
Кеш память 1го уровня (данные)
Кеш память 1го уровня (инструкций)
Кеш память 2го уровня
Кеш память 1го уровня (данные)
Кеш память 1го уровня (инструкций)
Ядро процессора 2
Кеш-память 3го уровня
Контроллер ОЗУ
Контроллер системной шины

Способы увеличения производительности процессора

Конвейеризация

Суперскалярность

Параллельная обработка данных

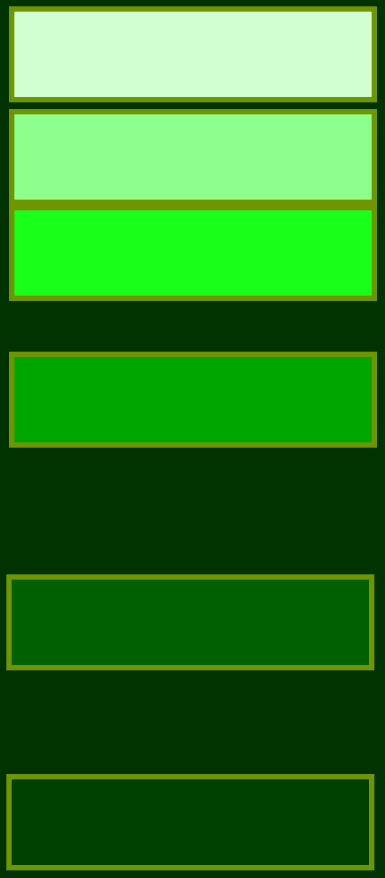
Технология Hyper-Threading

Технология Turbo Boost

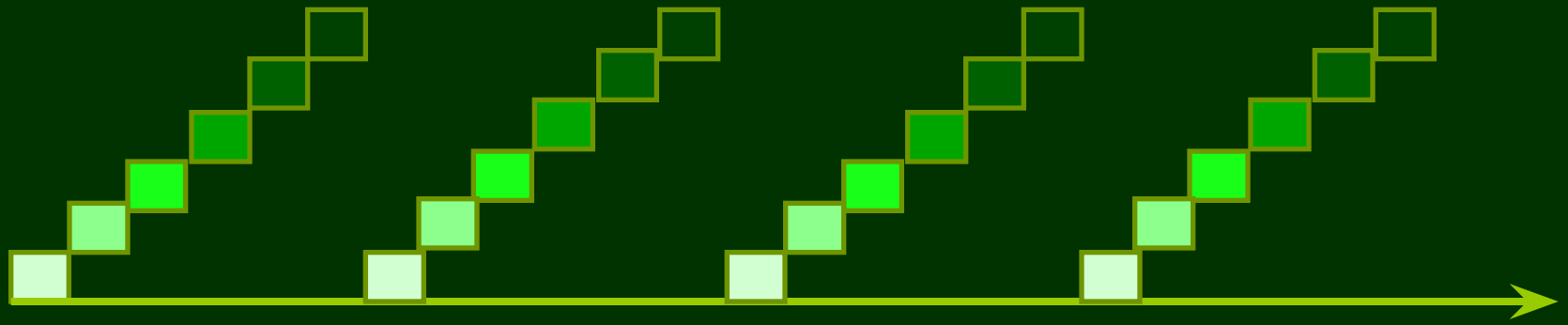
Эффективность выполнения команд

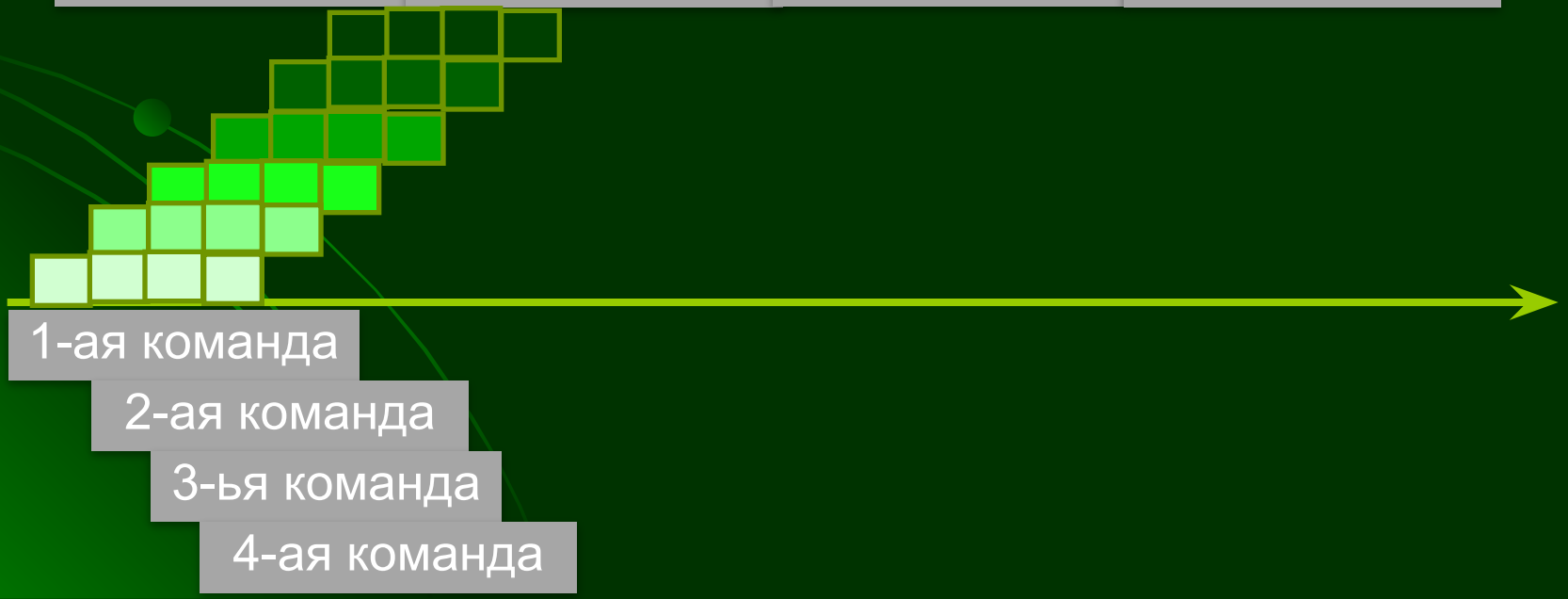
Конвейер

- Выбор команды
- Дешифрация
- Запрос операндов
- Выполнение команды с получением результата и/или формированием признаков
- Запись результата
- Увеличение (изменение) счетчика команд

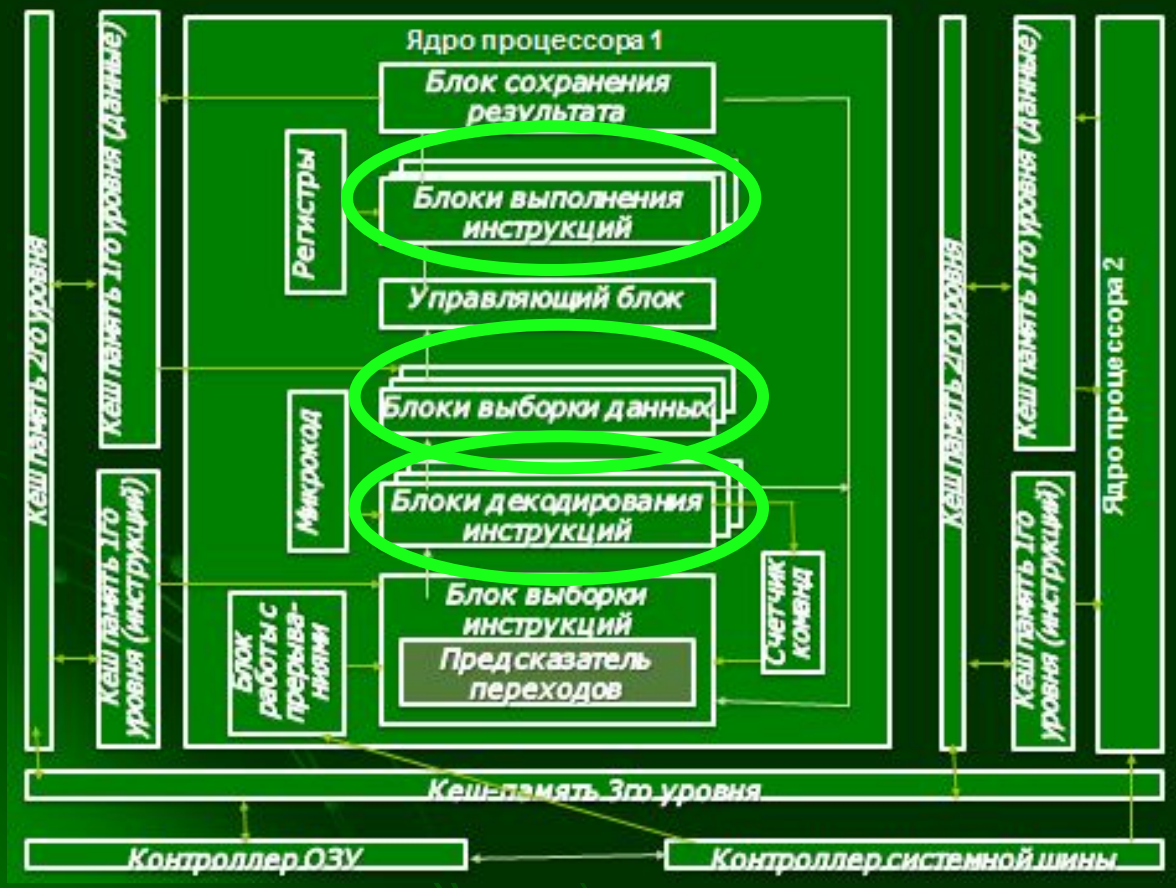


Конвейеризация



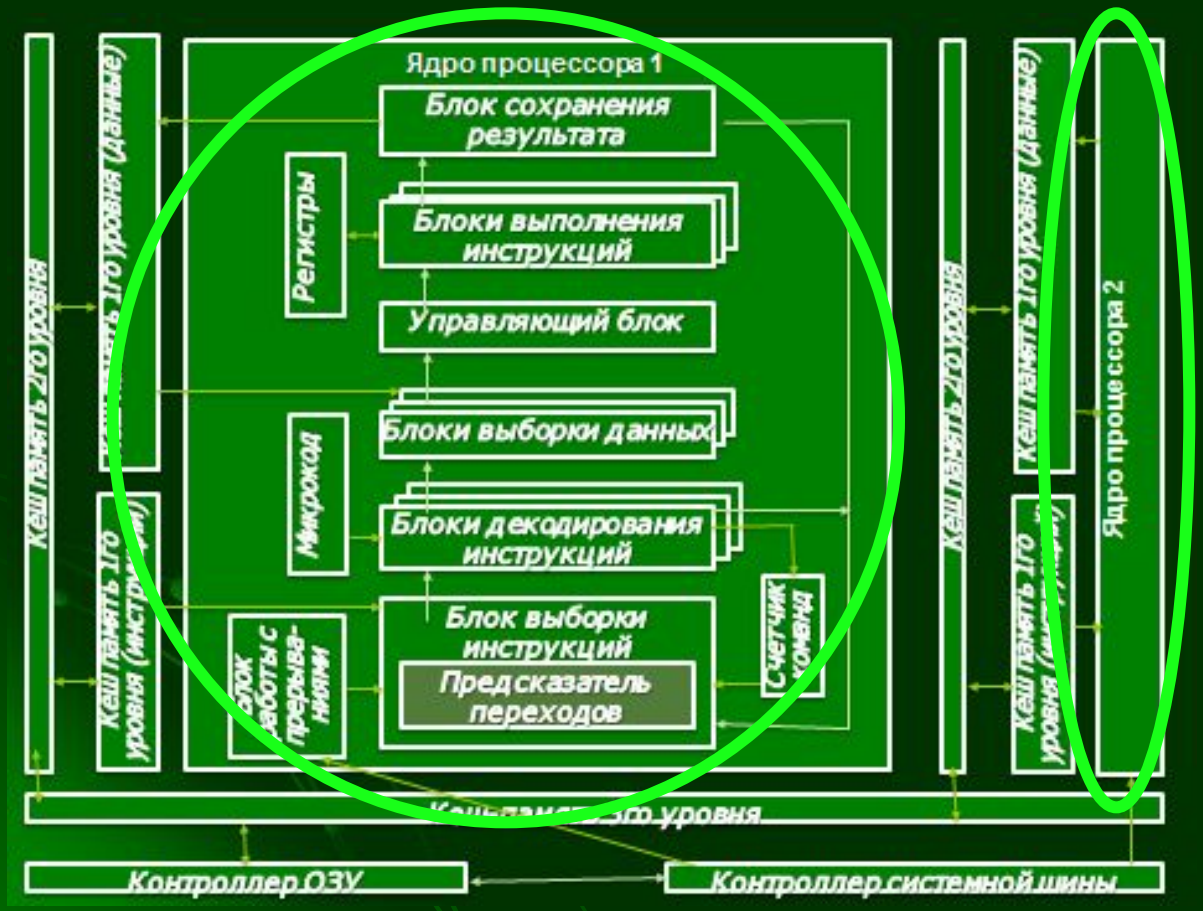


Суперскалярность



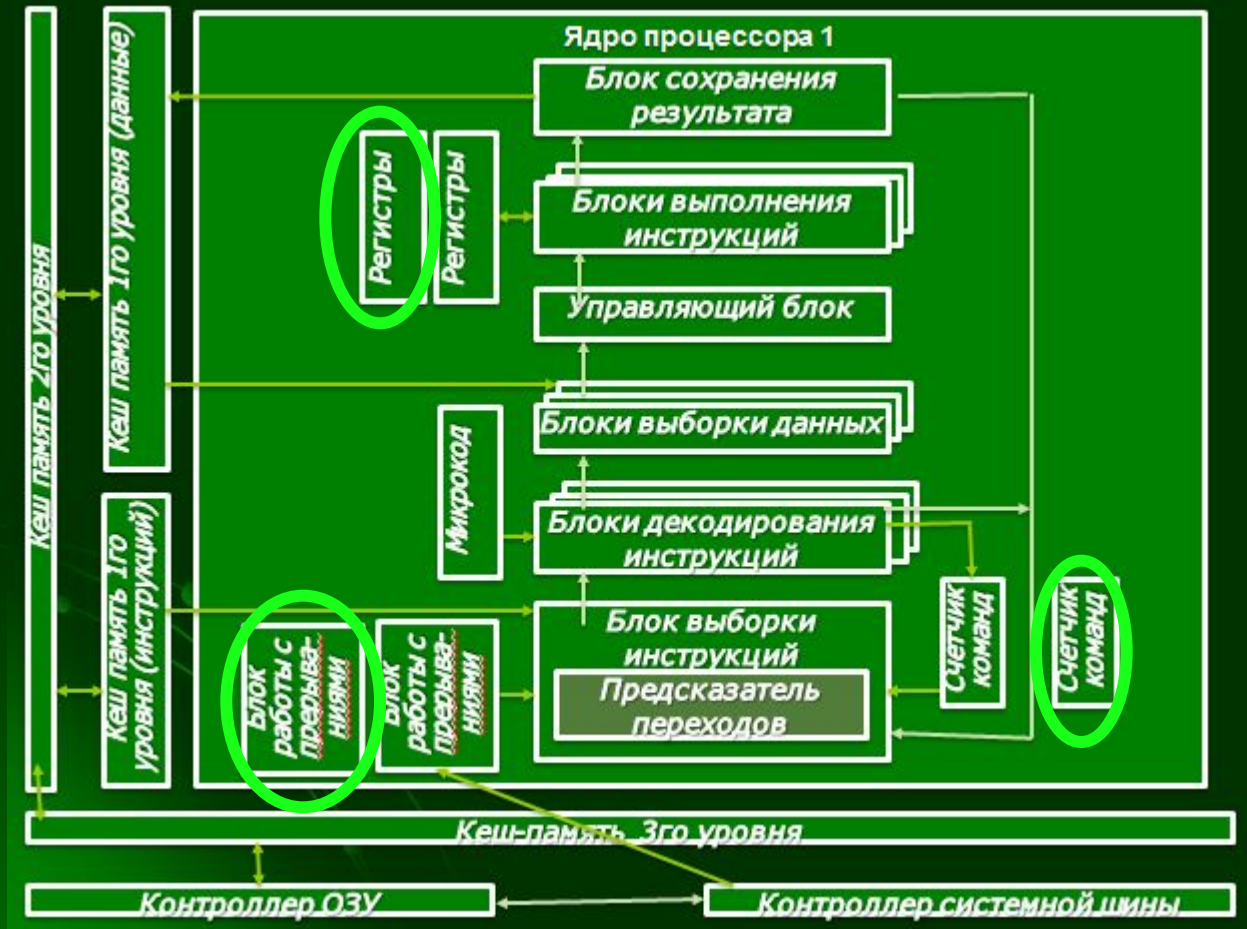
- Наиболее нагруженные блоки присутствуют в нескольких экземплярах
- Параллельное выполнение возможно при независимости инструкций

Параллельная обработка данных



- Не все программы могут работать на нескольких ядрах
- Одна программа – одно ядро: а если программе надо более одного ядра?
- Сложный механизм доступа к ОП и проч. Ресурсам.
- Увеличение энергопотребления
- Стоимость

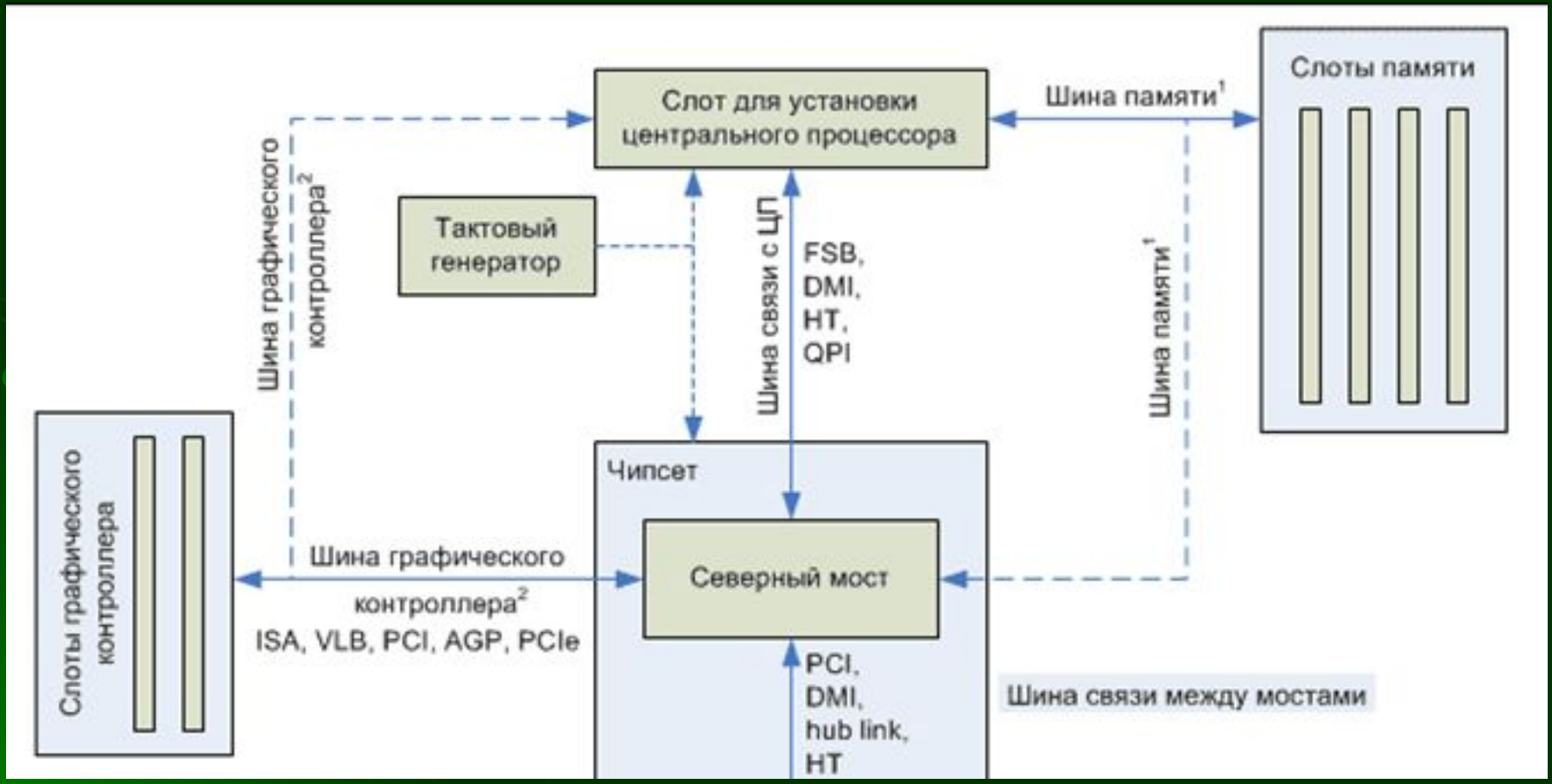
Технология Hyper-Threading



- Одно ядро выполняет две задачи одновременно: два потока (два виртуальных ядра)
- Каждое ядро имеет свой набор регистров, свой счетчик команд, свой блок работы с прерываниями для каждого потока
- Остальные элементы ядра общие

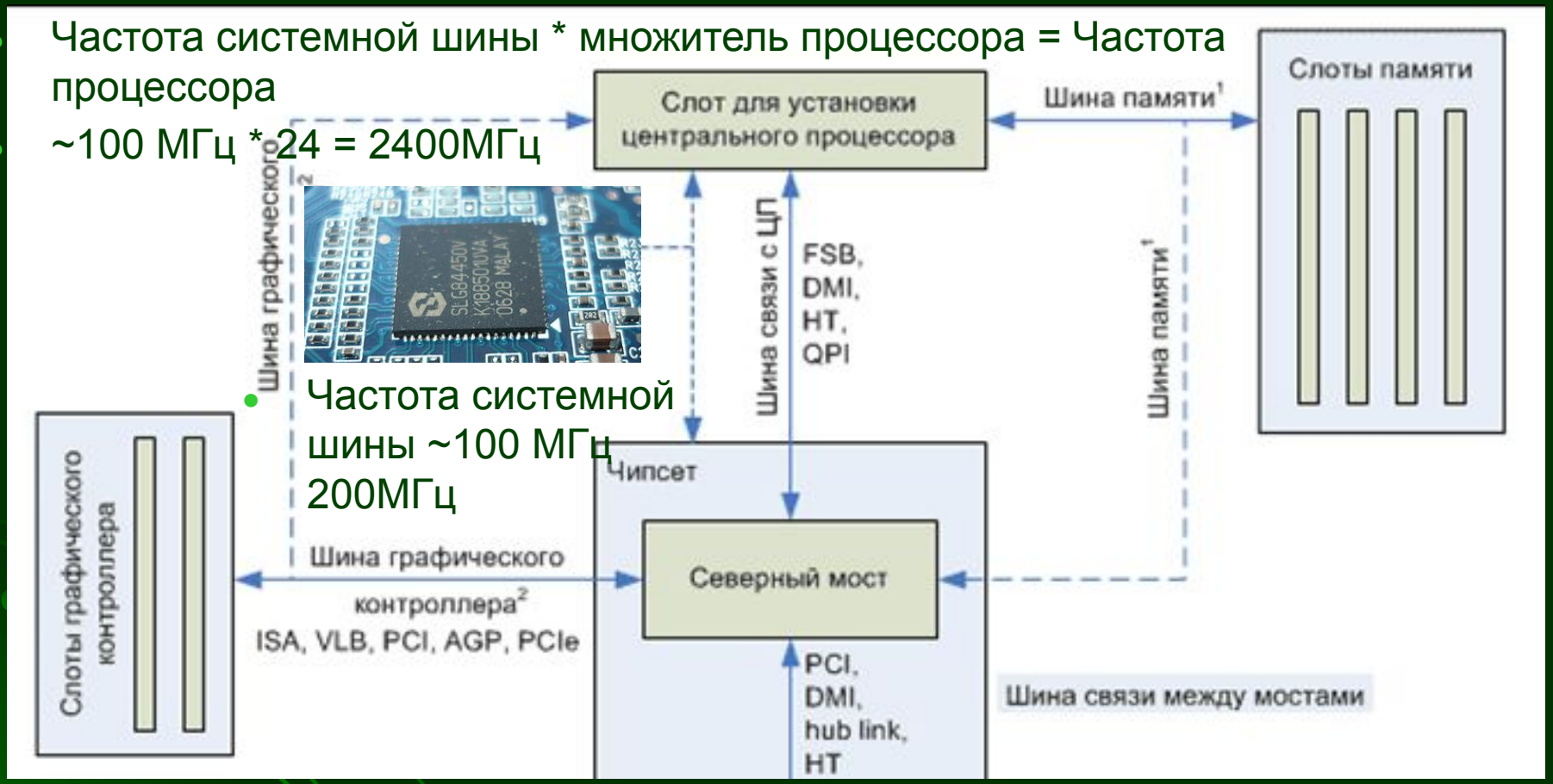
Технология Turbo Boost

- автоматический разгон ядер процессора до частоты выше базовой при контроле параметров: если мощность, потребляемый ток и температура не превышают максимальных значений



Технология Turbo Boost

- Частота системной шины * множитель процессора = Частота процессора
- ~100 МГц * 24 = 2400МГц



- Частота системной шины ~100 МГц 200МГц

- Динамическое повышение частоты
- Процессор контролирует все параметры своей работы: напряжение, силу тока, температуру и т.д.,
- Процессор может отключить неиспользуемые ядра

Эффективность выполнения команд : направления развития архитектур

- RISC (Reduced Instruction Set Computer): Небольшое количество простых команд, выполняемых за небольшое время
- MISC (Minimal Instruction Set Computer): Развитие RISC; 20-30 простых инструкций
- CISC (Complex Instruction Set Computing): Много сложных команд, способных выполнять различные действия; Много шагов на одну команду
- VLIW (Very long instruction word): длина инструкций может достигать 256 бит

RISC (Reduced Instruction Set Computer)

- фиксированная длина инструкций;
- небольшой набор стандартизированных инструкций;
- большое количество регистров общего назначения;
- отсутствие микрокода;
- меньшее энергопотребление, по сравнению с CISC-процессорами аналогичной производительности;
- более простое внутреннее устройство;
- меньшее количество транзисторов, по сравнению с CISC-процессорами аналогичной производительности;
- отсутствие сложных специализированных блоков в ядре процессора
- Проще распараллеливать вычисления.

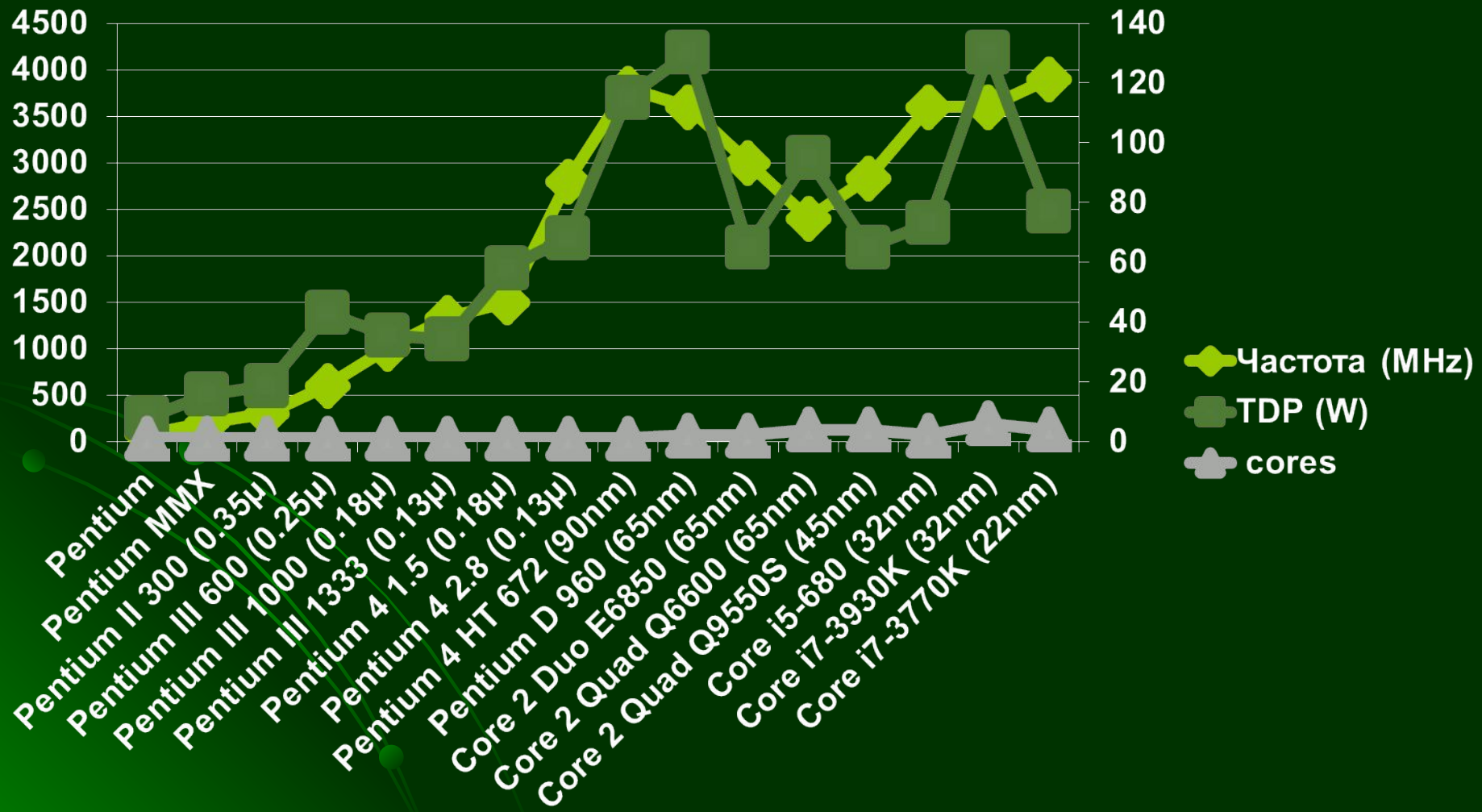
CISC (Complex Instruction Set Computing)

- Исторически первые
- Характеризовались
  - сложными и многоплановыми инструкциями;
  - большим набором различных инструкций;
  - нефиксированной длиной инструкций;
  - многообразием режимов адресации.
- Появление языков высокого уровня
- Начиная с Intel486DX CISC-процессоры стали производить с использованием RISC-ядра (микрокод)

Энергопотребление процессора

Модель	Частота (MHz)	TDP (W)	cores
Pentium	75	8	1
Pentium MMX	200	15,7	1
Pentium II 300 (0.35µ)	300	18,6	1
Pentium III 600 (0.25µ)	600	43	1
Pentium III 1000 (0.18µ)	1000	35,5	1
Pentium III 1333 (0.13µ)	1330	34	1
Pentium 4 1.5 (0.18µ)	1500	58	1
Pentium 4 2.8 (0.13µ)	2800	68	1
Pentium 4 HT 672 (90nm)	3800	115	1
Pentium D 960 (65nm)	3600	130	2
Core 2 Duo E6850 (65nm)	3000	65	2
Core 2 Quad Q6600 (65nm)	2400	95	4
Core 2 Quad Q9550S (45nm)	2830	65	4
Core i5-680 (32nm)	3600	73	2
Core i7-3930K (32nm)	3600	130	6
Core i7-3770K (22nm)	3900	77	4

Энергопотребление процессора



Способы снижения энергопотребления процессора

- Портативные устройства
- Снижение частоты – потеря производительности…

Технология EIST (Enhanced Intel SpeedStep Technology)

- позволяет динамически изменять энергопотребление процессора, за счет изменения тактовой частоты процессора и напряжения: если процессор используется не полностью, тактовую частоту можно снизить

Технология Cool’n’Quiet (AMD)

Подитог: Характеристики процессора

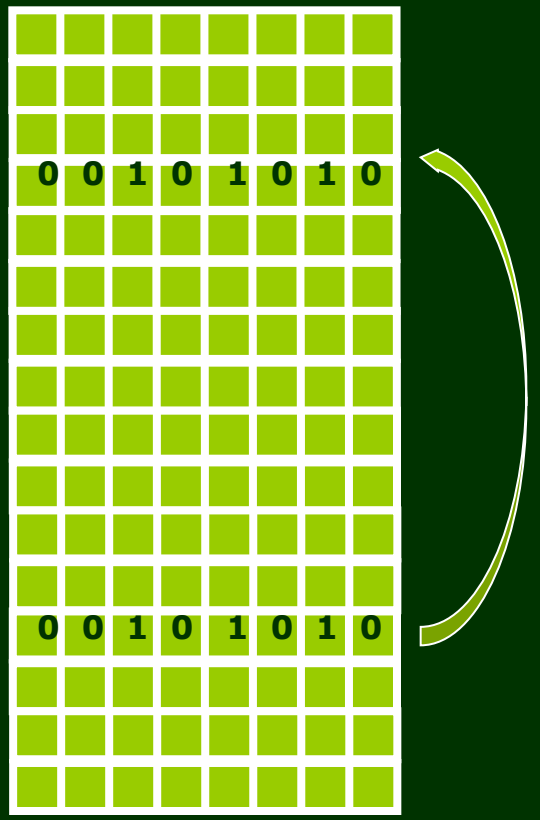
- Количество ядер
- Частота процессора как количество элементарных операций, которые процессор может выполнить в секунду (ГГц)
- Техпроцесс
- Энергопотребление
- Системная шина (FSB)
- Разрядность
- Кеш-память

Команды процессора

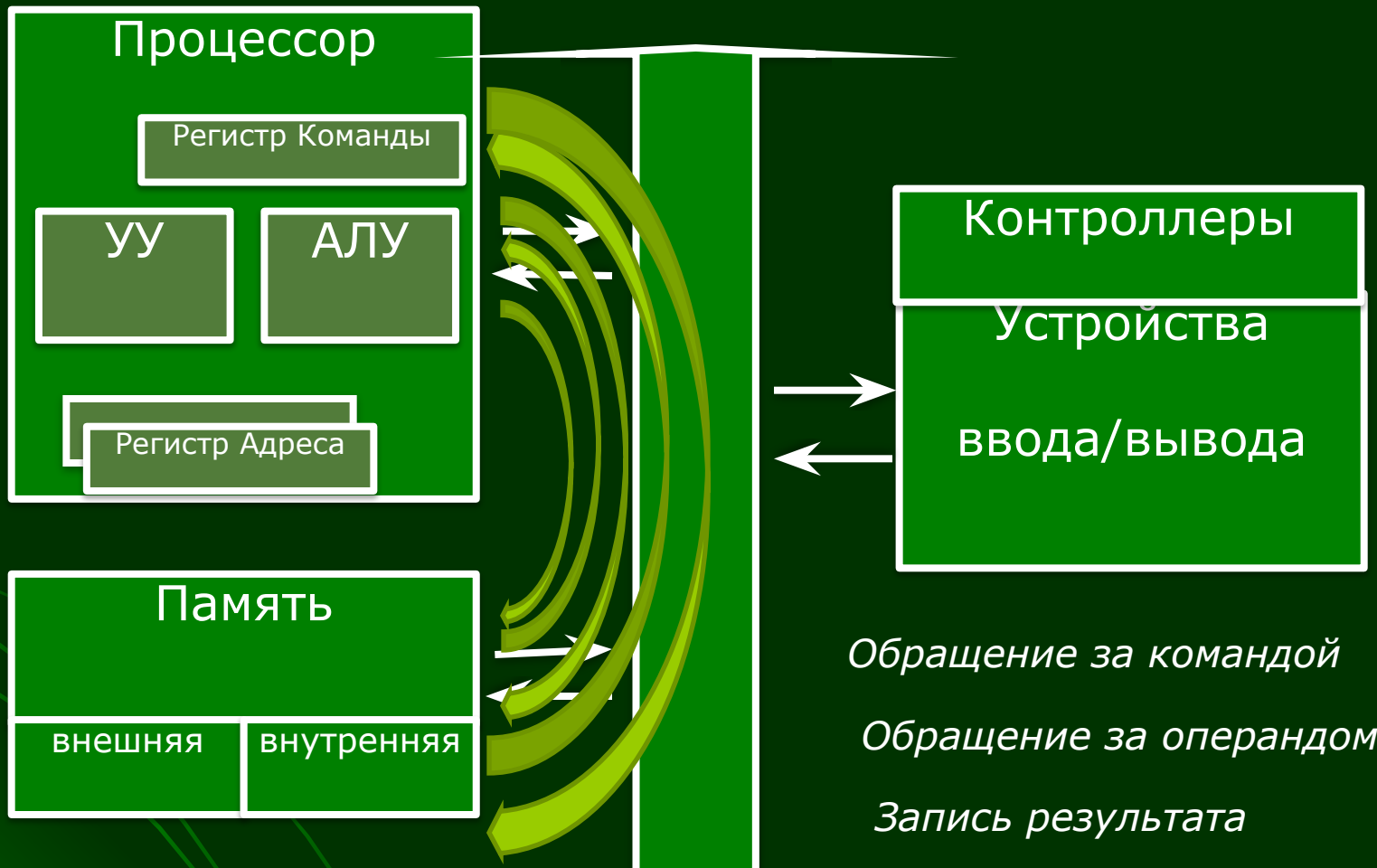
- Любые операторные языки
  - СИ
  - Паскаль
  - Бейсик
- - опираются на систему команд процессора

- A=B

пересылка данных



Выполнение команды пересылки



Вывод: команды должны быть как можно более короткие

Время!

Как ускорить выполнение команды?

Уменьшить длину команды

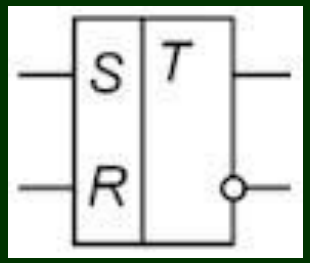
Сложение	Слагаемое 1	Слагаемое 2	Сумма

Сложение	Слагаемое 1 Сумма	Слагаемое 2

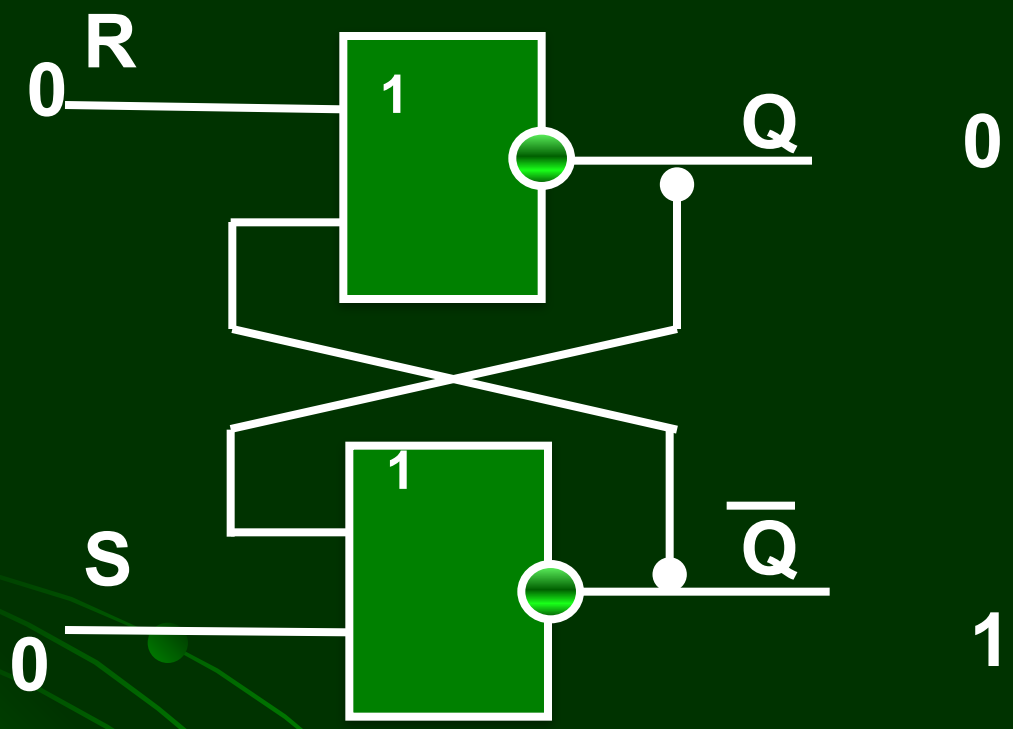
Организовывать вычисления с минимальным количеством обращений к памяти **РЕГИСТРЫ**

Регистры

- Один из операторов обязательно регистр
- Регистр - последовательное или параллельное логическое устройство, используемое для хранения n-разрядных двоичных чисел и выполнения преобразований над ними.
- Регистр - упорядоченная последовательность триггеров
- Триггер – устройство для хранения бита информации



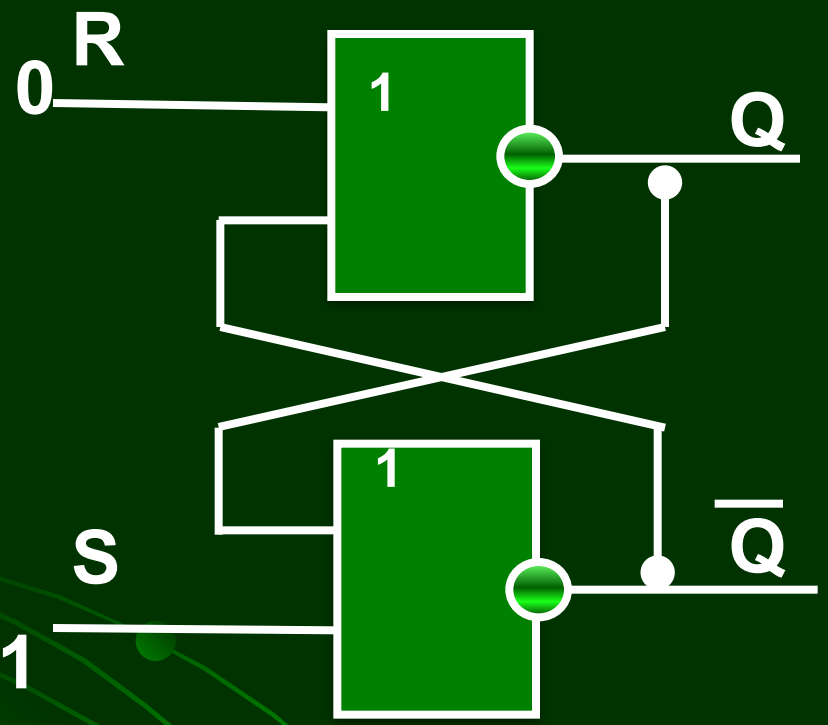
Устройство триггера



R=0
S=0
Не изменяет состояние триггера

Таблица истинности ИЛИ-НЕ		
V1	V2	Vout
0	0	1
0	1	0
1	0	0
1	1	0

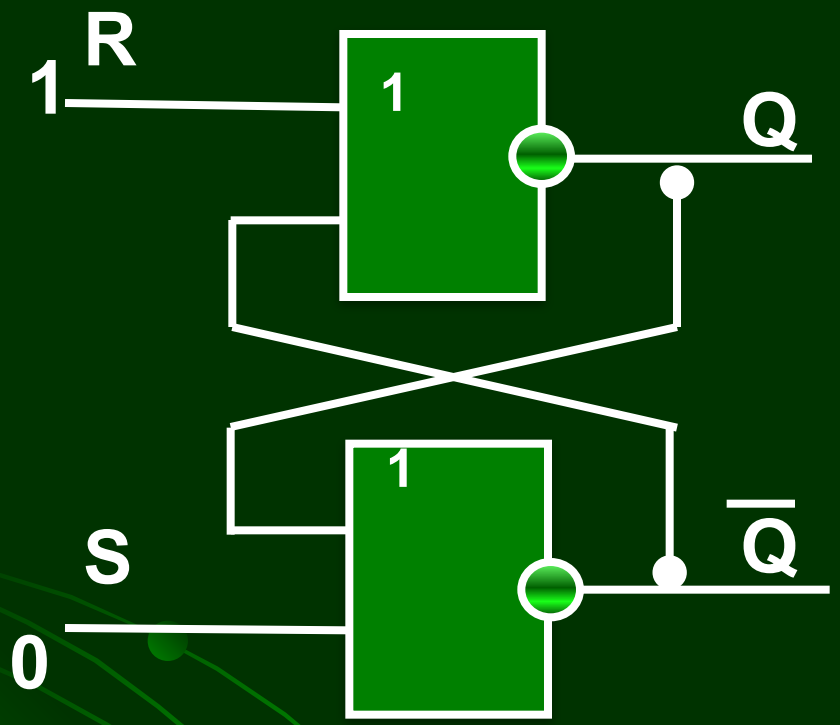
Устройство триггера



R=0
S=1
Устанавливает триггер в единицу

Таблица истинности ИЛИ-НЕ		
V1	V2	Vout
0	0	1
0	1	0
1	0	0
1	1	0

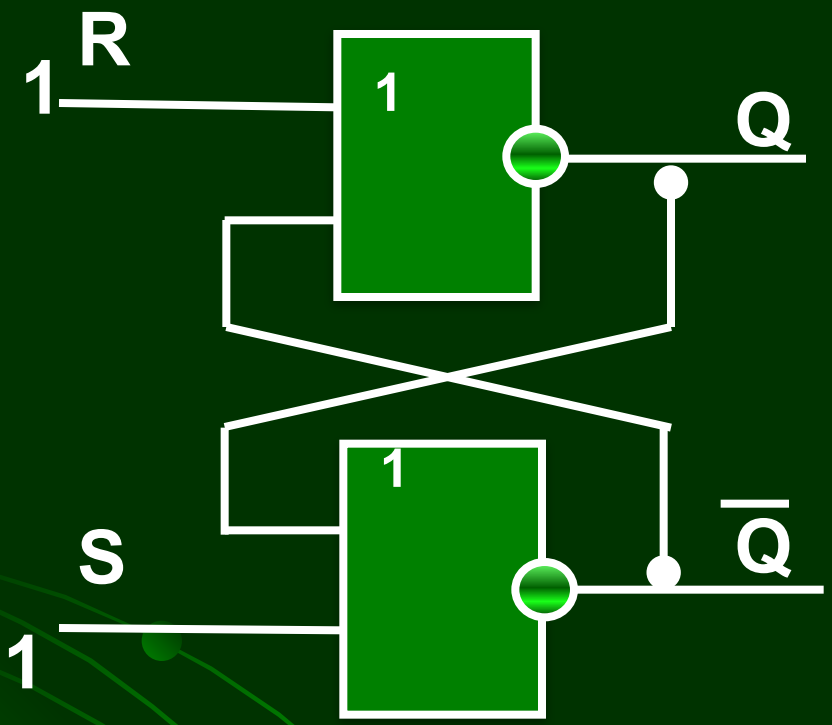
Устройство триггера



R=1
S=0
Устанавливает триггер в ноль

Таблица истинности ИЛИ-НЕ		
V1	V2	Vout
0	0	1
0	1	0
1	0	0
1	1	0

Устройство триггера



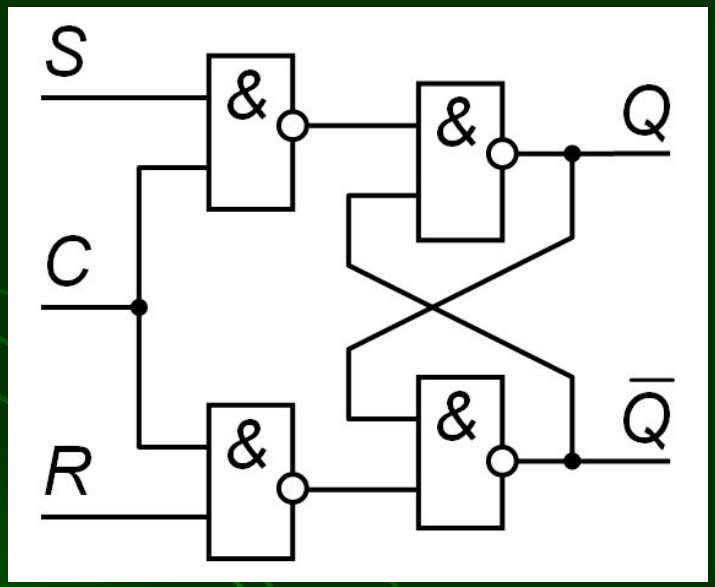
R=1
S=1
Запрещенная комбинация

Таблица истинности ИЛИ-НЕ

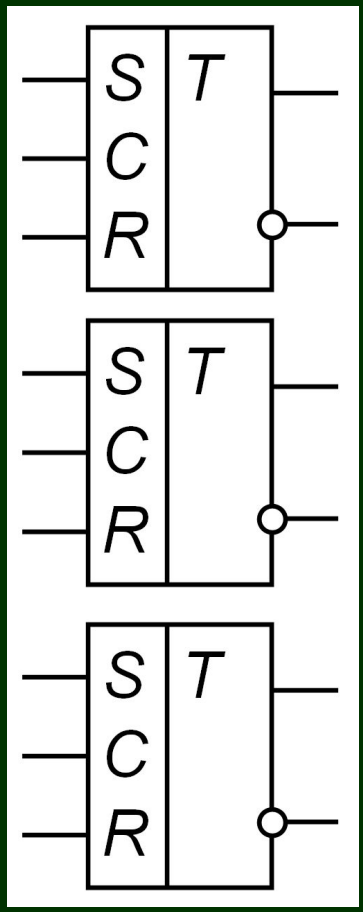
V1	V2	Vout
0	0	1
0	1	0
1	0	0
1	1	0

Регистр – совокупность триггеров

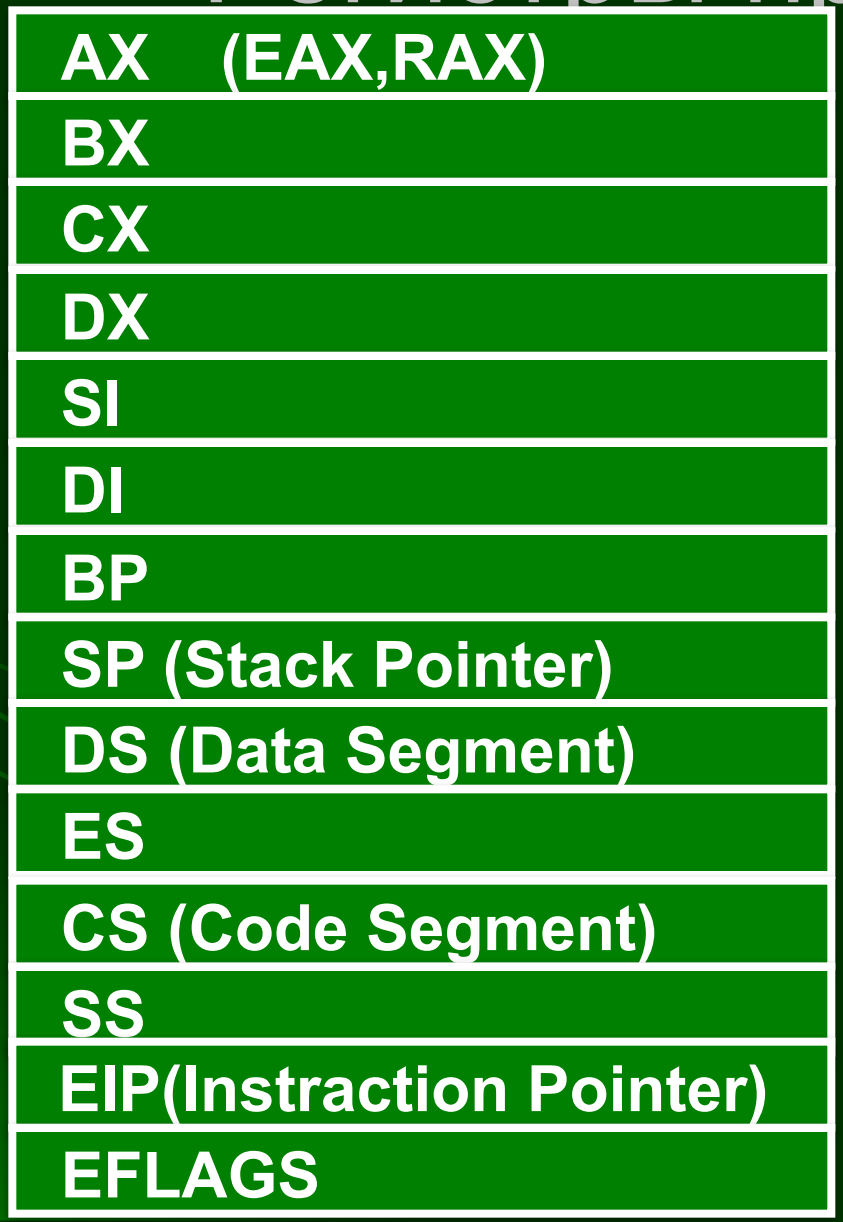
- Схема синхронного RS-триггера на элементах 2И-НЕ



- Условное графическое обозначение синхронного RS-триггера



Регистры процессора



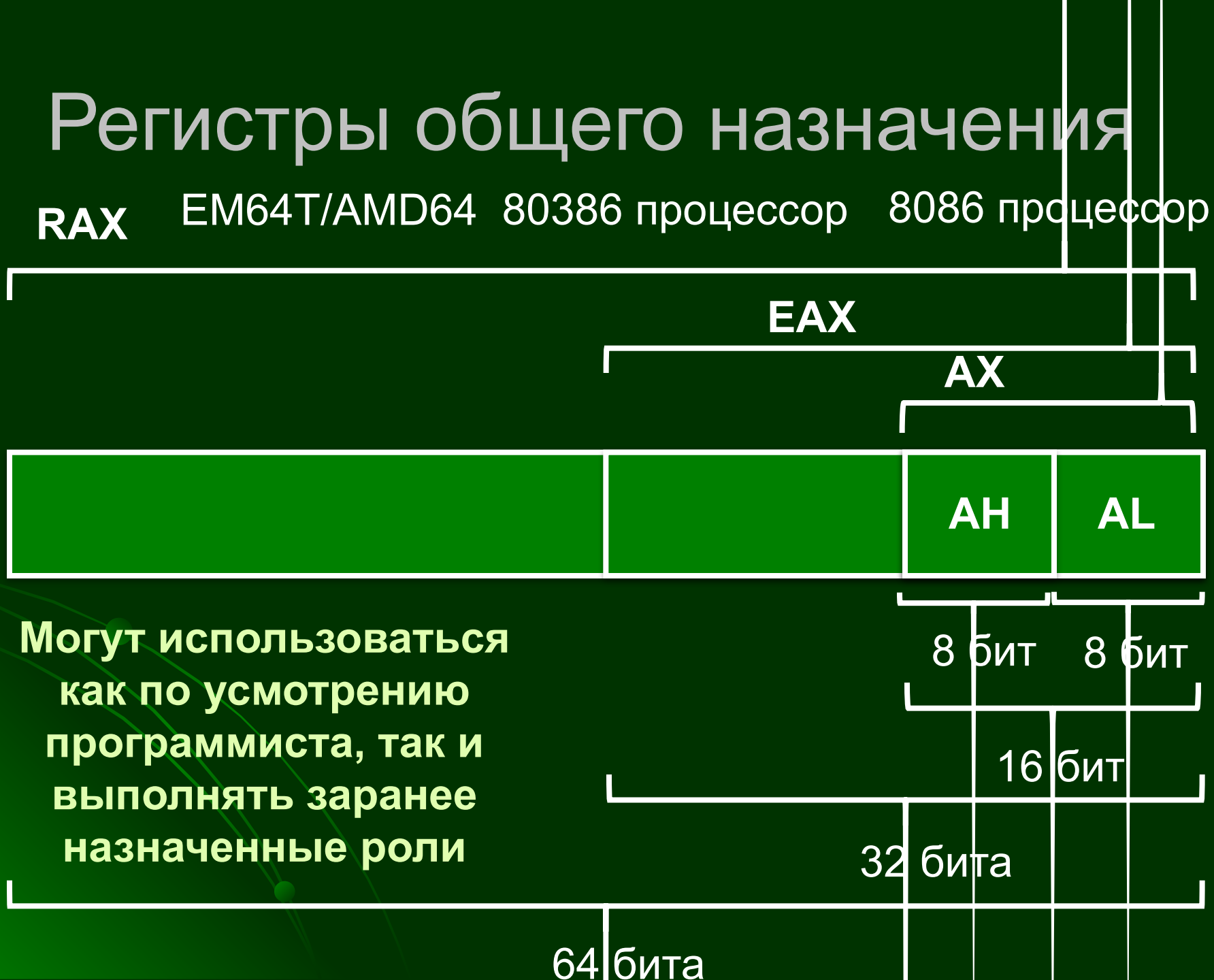
Регистры общего назначения

Сегментные регистры – обращение к памяти

Счетчик команд

Регистр признаков

Регистры общего назначения



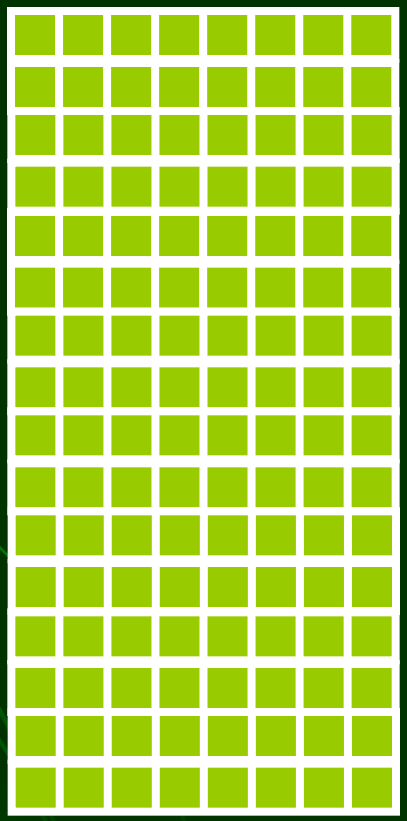
- **Могут использоваться как по усмотрению программиста, так и выполнять заранее назначенные роли**

Регистры общего назначения

- **AX — аккумулятор; для хранения операндов в командах умножения и деления, ввода-вывода, в некоторых командах обработки строк и других операциях;**
- **BX — регистр базы; для хранения адреса или части адреса операнда, находящегося в памяти;**
- **CX — счётчик. Содержит количество повторений строковых операций, циклов и сдвигов;**
- **DX — регистр данных. Используется для косвенной адресации портов ввода-вывода, а также как «расширитель» аккумулятора в операциях удвоенной разрядности;**
- **SI — регистр адреса источника. Используется в строковых операциях, а также в качестве индексного регистра при обращении к операндам в памяти;**
- **DI — регистр адреса приёмника. Используется в строковых операциях, а также в качестве индексного регистра при обращении к операндам в памяти;**
- **BP — указатель кадра стека. Используется для адресации операндов, расположенных в стеке;**
- **SP — указатель стека. Используется при выполнении операций со стеком, но не для явной адресации операндов в стеке.**

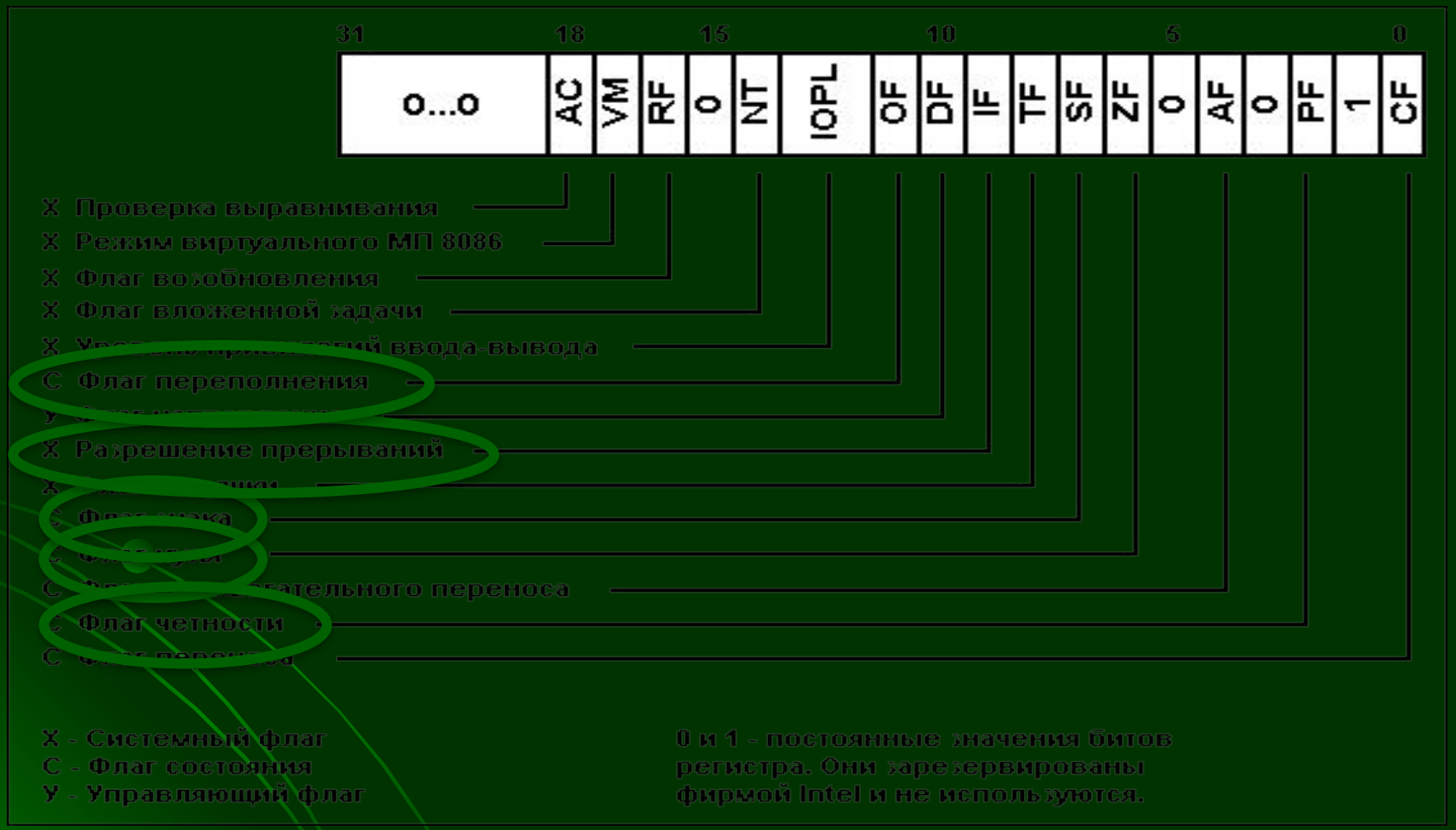
Сегментные регистры

- Сегмент – выделенная область пространства памяти



- **CS** сегмента кода - в каком месте памяти находится программа
- **DS** сегмента данных - локализует используемые программой данные.
- **ES** дополняет сегмент данных.
- **SS** сегмента стека - стек компьютера.

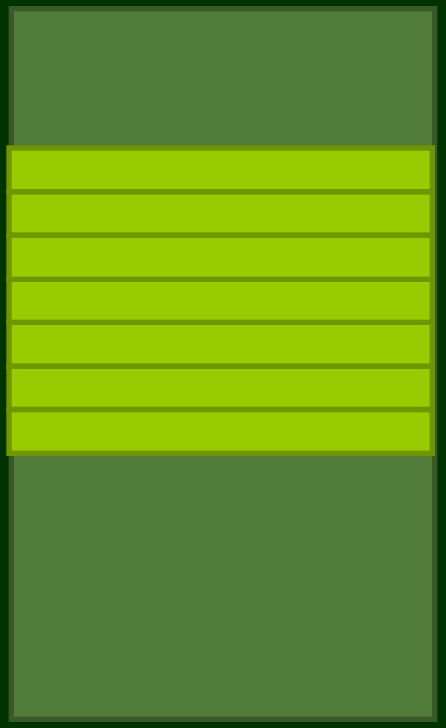
Регистр признаков



- Содержит слово состояния процессора

Стек

- Специальная область памяти
- Структура данных с методом доступа к элементам **LIFO** (Last In — First Out, «последним пришёл — первым вышел»)

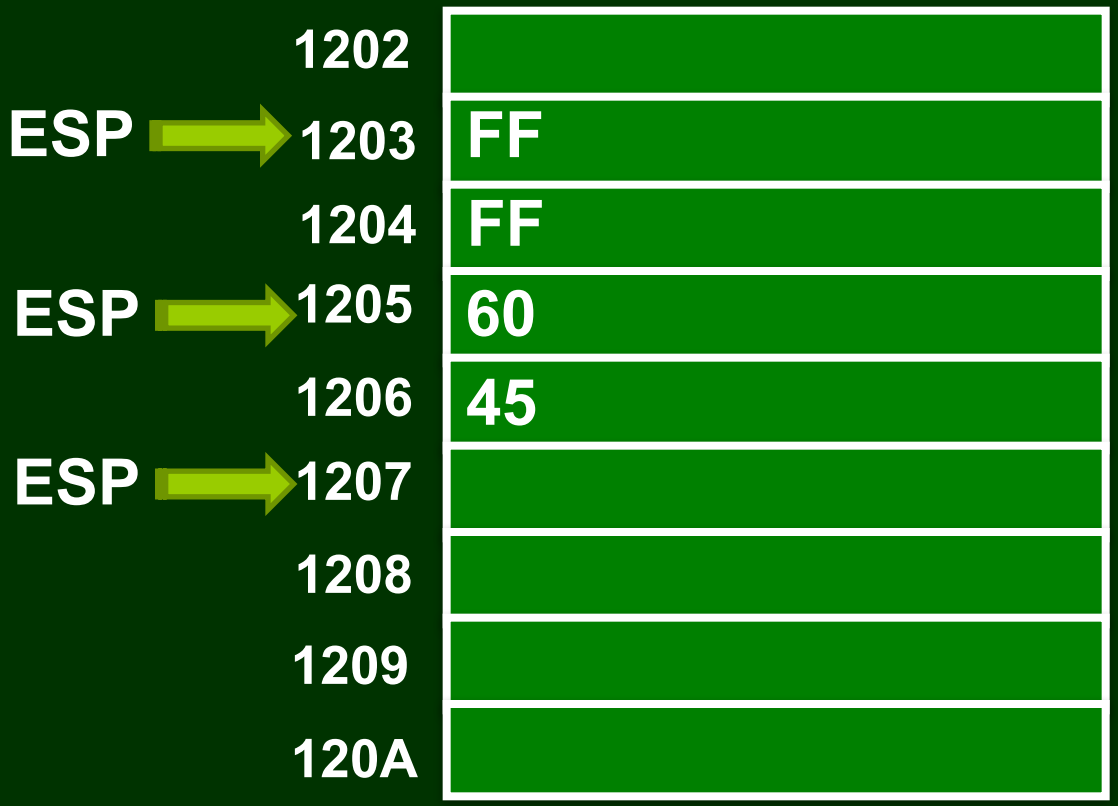


Размещении и извлечении значений в стеке

```
mov ax, 4560h
push ax

mov cx, FFFFh
push cx

pop edx
```



edx: 4560FFFFh

Система команд

- Команды пересылки
- Команды обработки данных:
  - Арифметические
  - Логические команды
  - Команды сдвига
  - Команды ветвления или управления
- Команды обращения к процедурам
- Системные команды

Команды пересылки

- A=B
- Mov

- Между регистрами
- Между памятью и регистрами

```
mov ax,1234h
```

AX = 1234h, AH = 12h, AL = 34h

Арифметические команды

- Сложение ADD
- Вычитание SUB
- Умножение MUL
- Деление DIV
- Увеличение INC
- Уменьшение DEC
- Смена знака NEG

```
mov al,10
add al,15
---> al = 25
```

- i=i+1
- A++

```
mov cl,4Fh
inc cl
---> cl = 50h
```

Логические команды

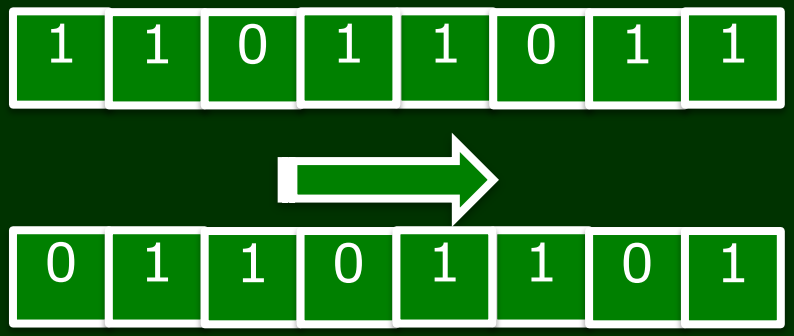
- Выполнение операций Булевой алгебры И (AND), ИЛИ (OR), НЕ (NOT), Исключающее ИЛИ (XOR)
- Команды применяются к байту; вычисления производятся с каждым битом
- Используются для установки, сброса и проверки требуемых бит.

```
and eax , 0fffffffdh
```

```
or eax , 10b
```

Команды сдвига

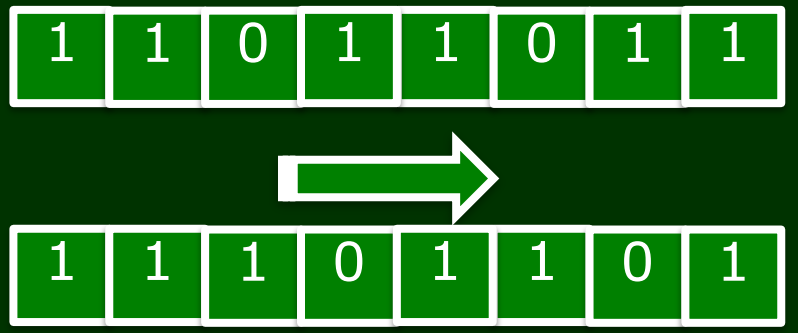
- Логический сдвиг



shr al,1

- Циклический сдвиг

rol al,1



Команды ветвления управления

- Безусловная передача управления
- Go to Label

jmp

- Команды условного перехода
- If A>B then …

CMP
<команда условного перехода>

Команды условного перехода

Мнемокод команды условного перехода	Критерий условного перехода	Значения флагов для перехода
JE	операнд_1=операнд_2	ZF=1
JNE	операнд_1<>операнд_2	ZF=0
JL/JNGE	операнд_1<операнд_2	SF<>OF
JLE/JNG	операнд_1<=операнд_2	SF<>OF или ZF=1
JG/JNLE	операнд_1>операнд_2	SF=OF или ZF=0
JGE/JNL	операнд_1=>операнд_2	SF=OF
JB/JNAE	операнд_1<операнд_2	CF=1
JBE/JNA	операнд_1<=операнд_2	CF=1 или ZF=1
JA/JNBE	операнд_1>операнд_2	CF=0 и ZF=0
JAE/JNB	операнд_1=>операнд_2	CF=0

Команды ветвления управления

- Команды циклов
- For x=5 to 17
- …
- Next x

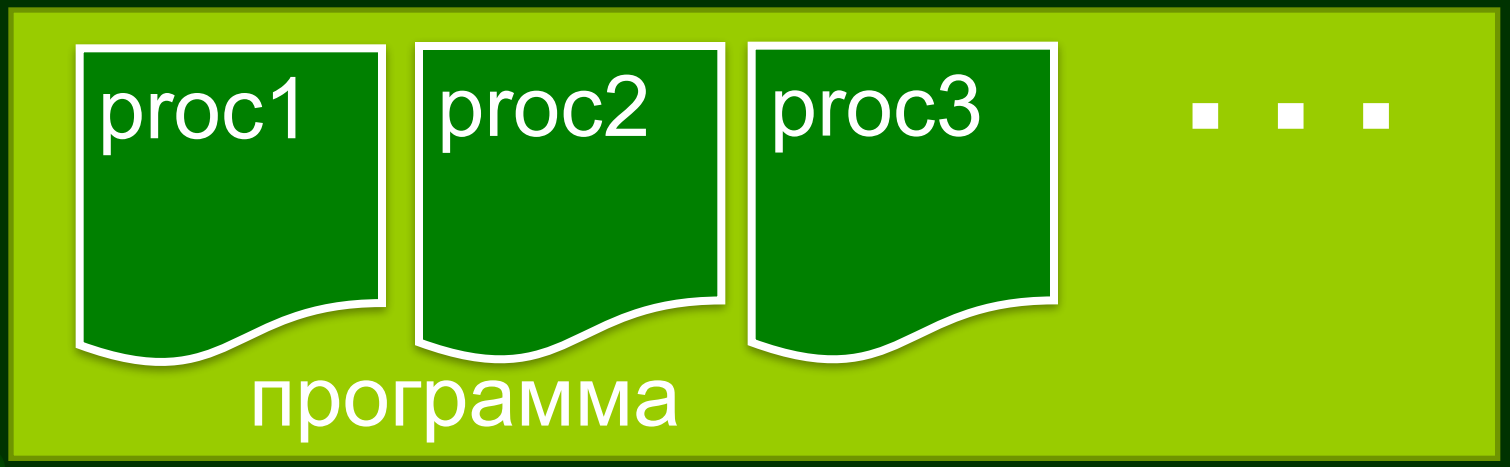
```
mov cx,5
метка:

LOOP <метка>
```

перевод на указанную метку до тех пор пока регистр **CX** не станет равный нулю

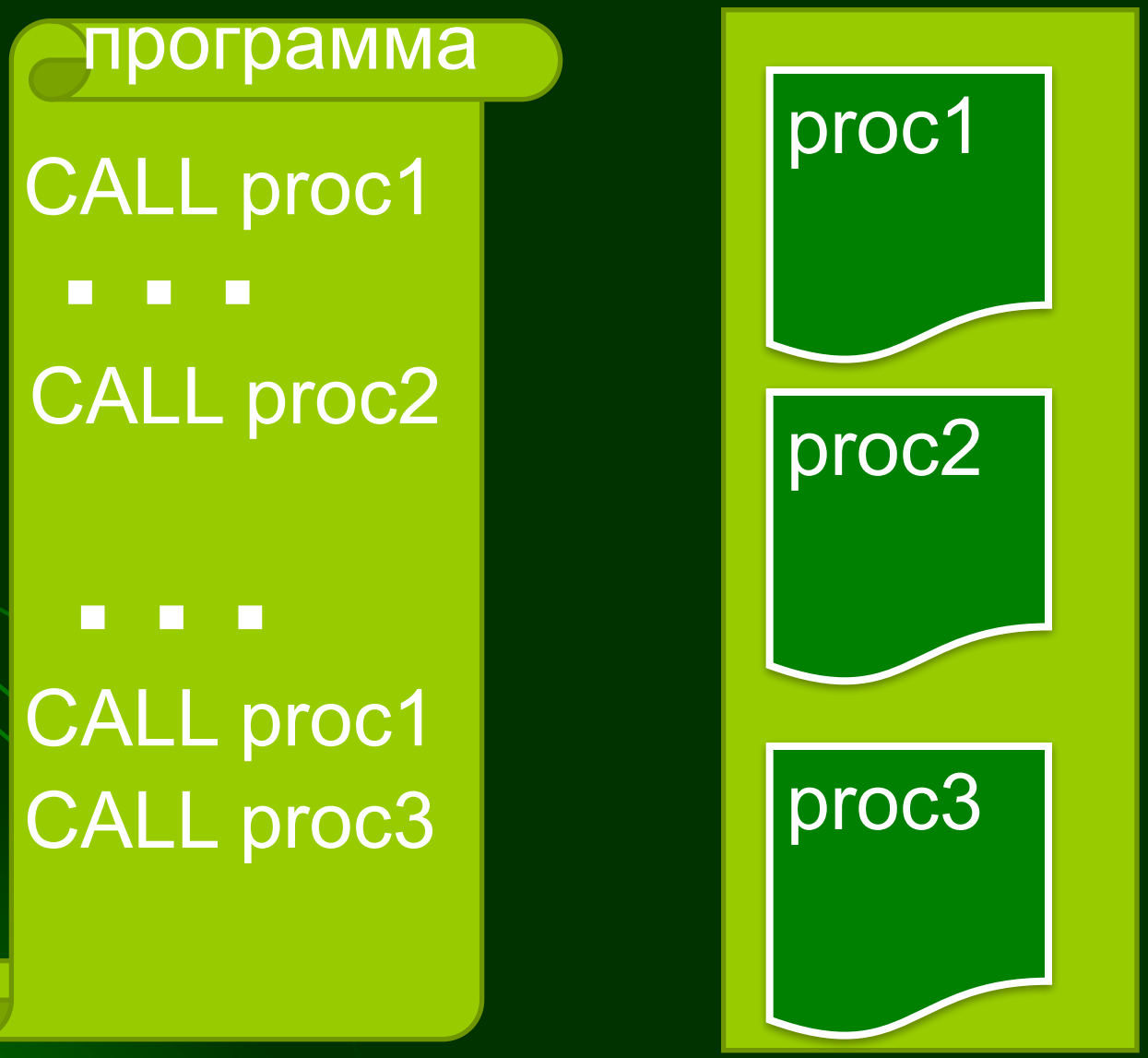
Процедуры

- Программа разбивается на части



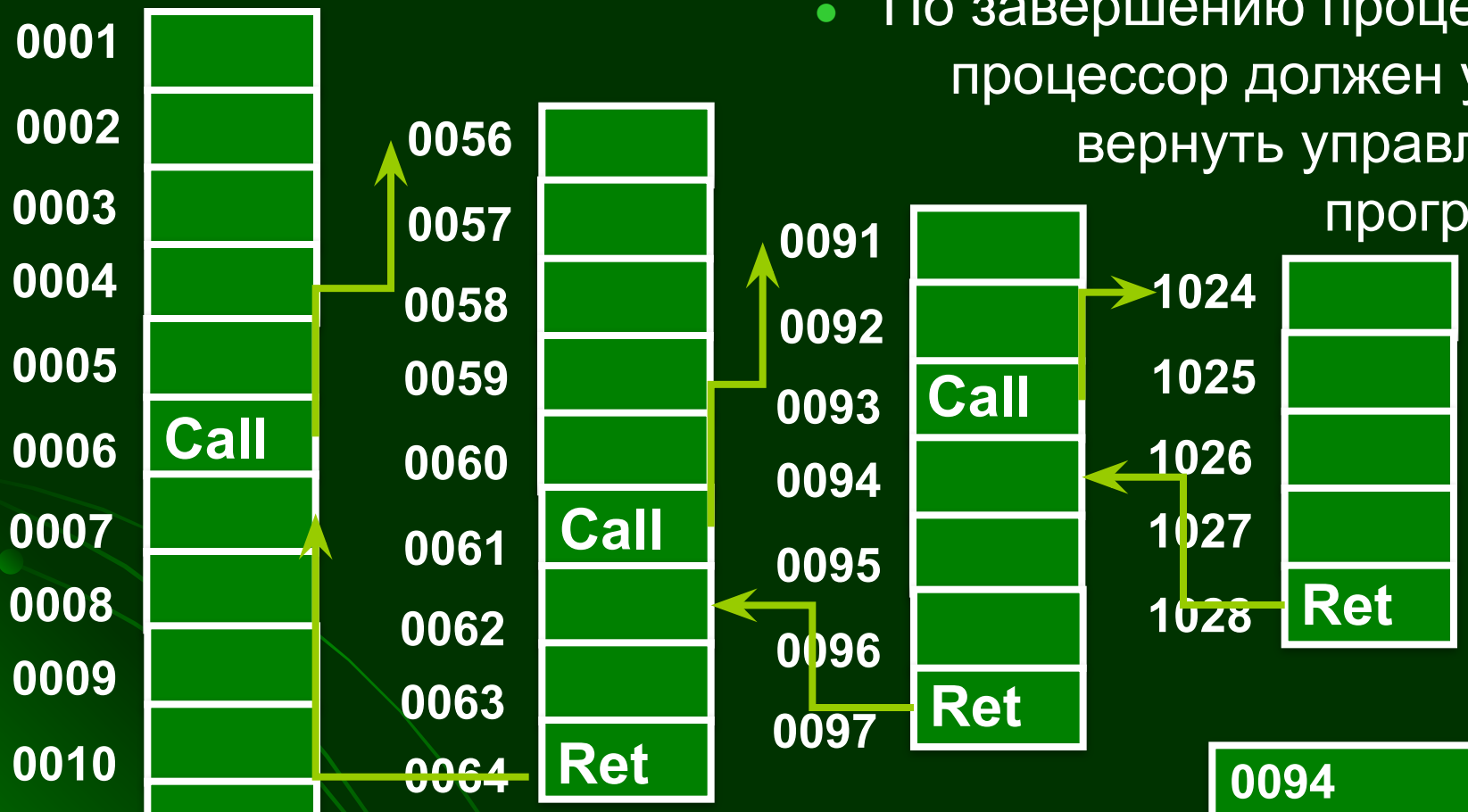
- CALL передача управления процедуре
- В конце процедуры команда RET возвращает управление программе

Использование процедур

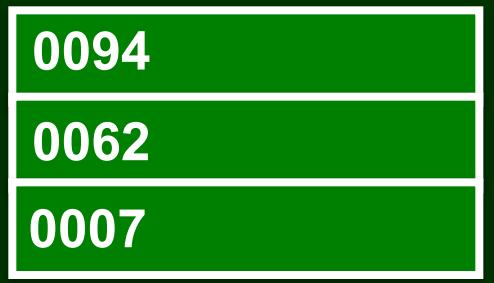


Обращения к процедурам

- По завершению процедуры процессор должен уметь вернуть управление программе



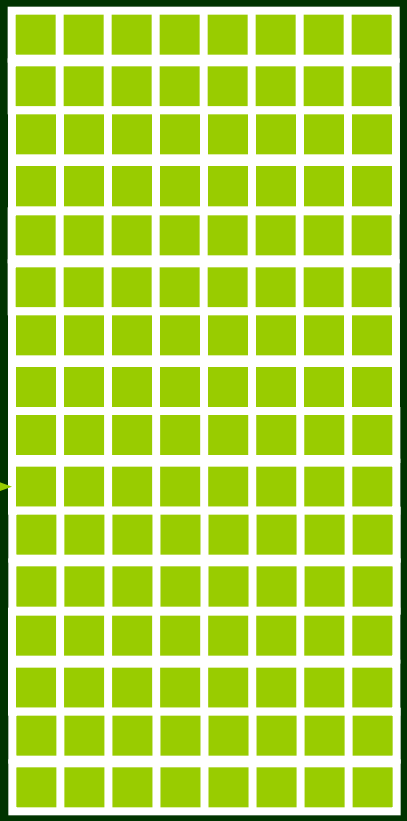
- Адрес возврата записывается в стек



Адресация

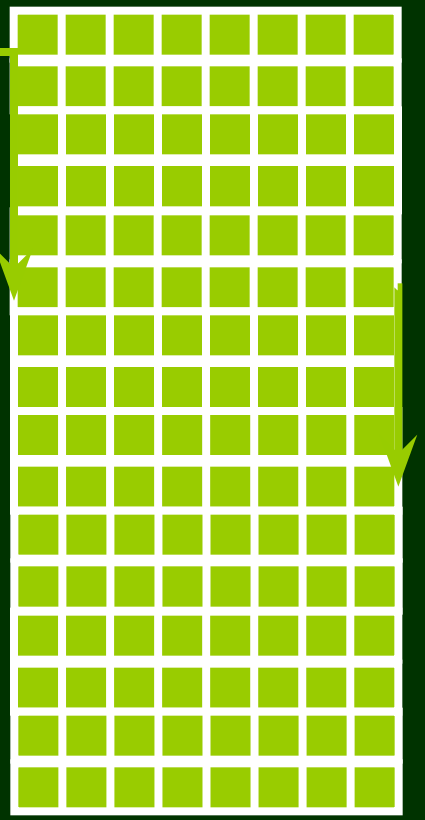
- Прямая
- Косвенная (адрес адреса операнда)

Прямая адресация



```
mov ax,1234h
add bx,ax
```

Косвенная адресация



Операнд указывает на адрес требуемых данных

```
mov     ax,[cx]
add ax,[bx+2]
```