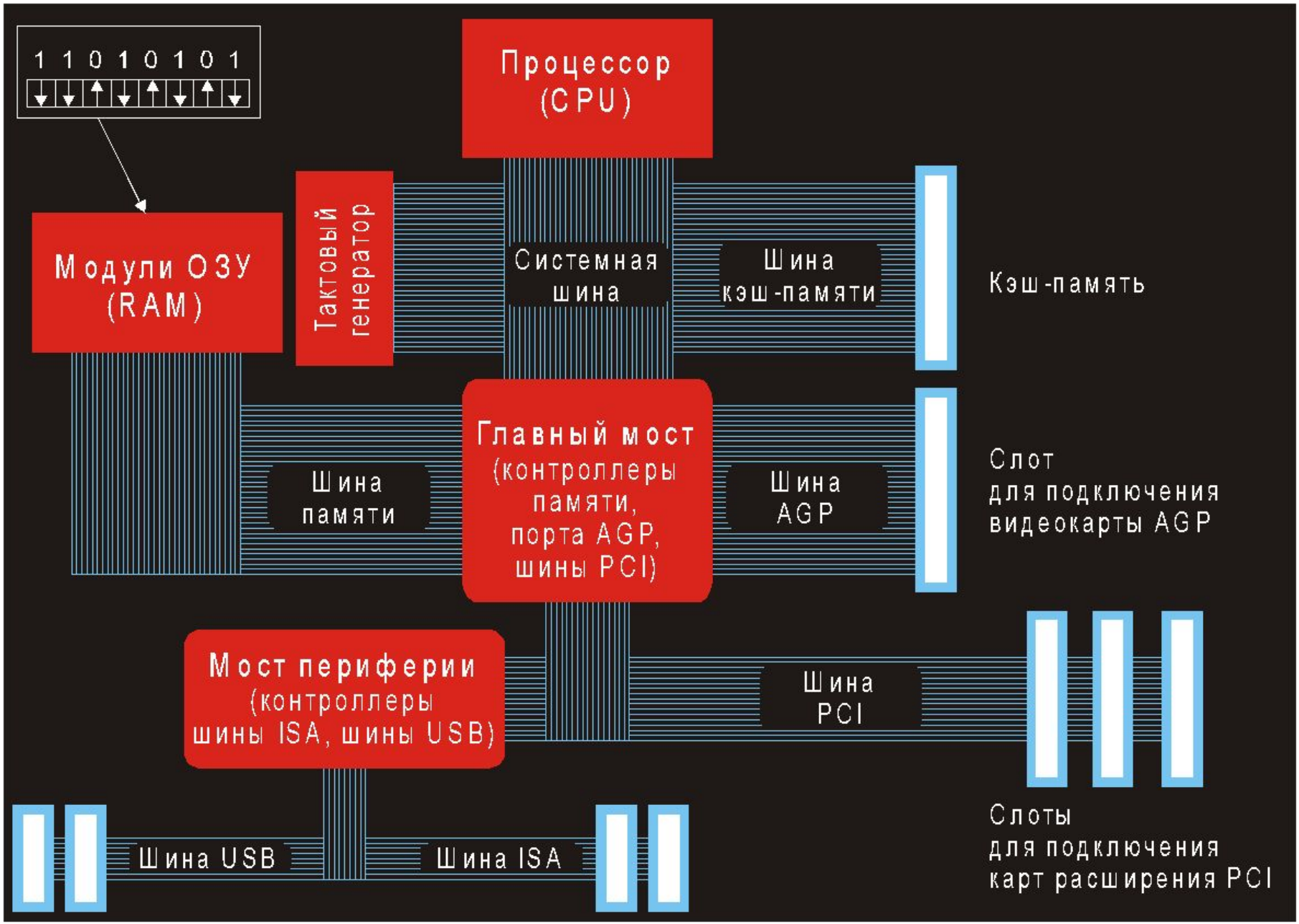
Программирование – это процесс создания программ.
Программа – набор инструкций, посланный вычислительной машине (компьютеру).

Носителем информации является *сообщение*.
Для кодировки *сообщений* применяется двоичный набор, состоящий из двух знаков 0 и 1 (*binary digit*, сокращенно *bit*).

Данные – это сообщения, закодированные в форму, пригодную для хранения и обработки их компьютером на основе двоичного набора знаков.

Порядок выполнения операций над данными строится на основе некоторого *алгоритма*.

Программу можно рассматривать как *алгоритм*, записанный на понятном для компьютера языке, и *данные*, которые компьютер будет обрабатывать в соответствии с этим алгоритмом.


1 1 0 1 0 1 0 1
Процессор (CPU)
Модули ОЗУ (RAM)
Тактовый генератор
Системная шина
Шина кэш-памяти
Кэш-память
Шина памяти
Главный мост (контроллеры памяти, порта AGP, шины PCI)
Шина AGP
Слот для подключения видеокарты AGP
Мост периферии (контроллеры шины ISA, шины USB)
Шина PCI
Шина USB
Шина ISA
Слоты для подключения карт расширения PCI

Байт – группа из восьми бит, обрабатываемая как единое целое.
(*бит* – единица измерения информации, *байт* – единица измерения объема данных).

Объем ОЗУ измеряется в *байтах*. Емкость ячейки памяти – 1 байт.
Производные: Кбайт (1024 (2^{10}) байт), Мбайт (1 048 576 (2^{20}) байт) , Гбайт (1 073 741 824 (2^{30}) байт).

Шина – набор линий-проводников на материнской плате, по которым обмениваются информацией компоненты и устройства компьютера.
Системная шина – проводники, которыми процессор соединен с *Chipset* – набором микросхем, установленных на материнской плате для обеспечения обмена данными между разными устройствами компьютера и процессором.

Архитектура любой из шин (системной, памяти, ввода-вывода и др.) включает:
- линии для обмена данными (*шина данных*),
- линии для адресации данных (*шина адресов*),
- линии для управления данными (*шины управления*).

Пропускная способность шины (Мбайт/с) определяется ее *разрядностью*, умноженной на *тактовую частоту*.

Современные компьютеры преимущественно базируются на *архитектуре фон Неймана* – совместное хранение данных и программ. *Алгоритм* любой программы реализуется в виде команд, выполняемых процессором *шаг за шагом* (программа = данные + команды).

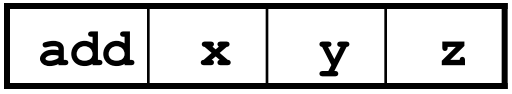
Команда состоит из кода выполняемой опера-ции (*оператор*) и адресной части (*операнды*).

Закодированное представление команды процессора - **машинный код**.

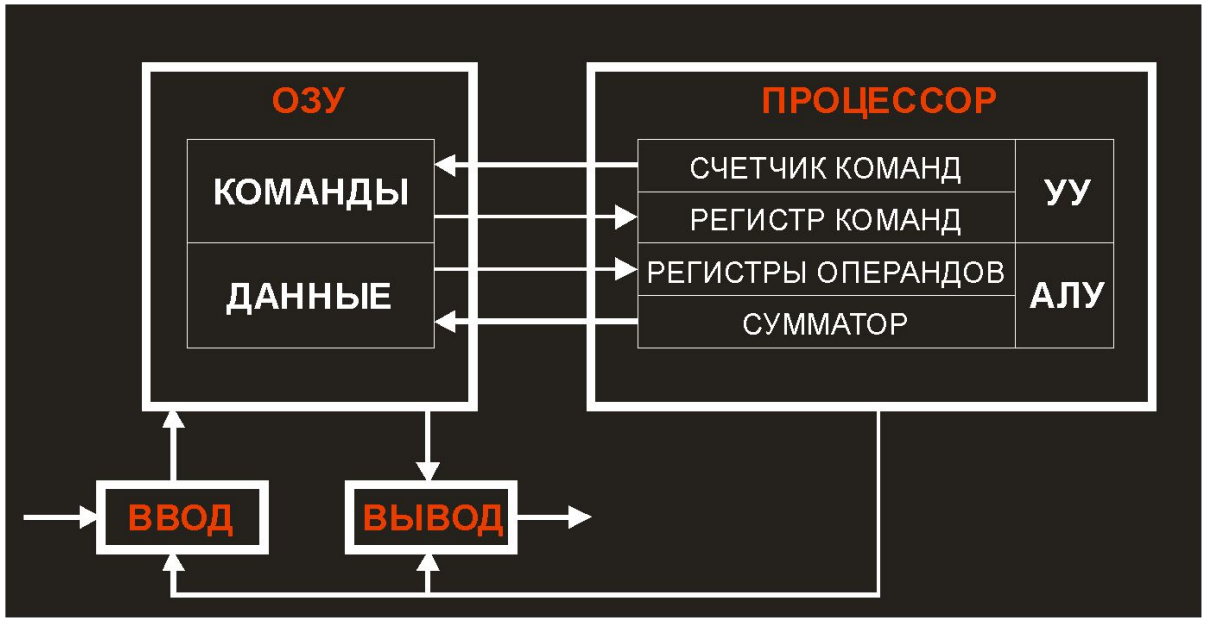
Регистры – дополнительные поименованные ячейки памяти процессора.

одноадресная команда (содержимое ячейки x ОЗУ сложить с содержимым сумматора (регистр АЛУ процессора), а результат оставить в сумматоре);

двухадресная команда (сложить содержимое ячеек x и y, а результат поместить в ячейку y);



трехадресная команда (содержимое ячейки x сложить с содержимым ячейки y, сумму поместить в ячейку z);



1. из ячейки памяти, адрес которой хранится в счетчике команд, выбирается очередная команда; содержимое счетчика команд при этом увеличивается на длину команды;
2. выбранная команда передается в УУ на регистр команд;
3. устройство управления расшифровывает адресное поле команды;
4. по сигналам УУ операнды считываются из памяти и записываются в АЛУ на специальные регистры операндов;
5. УУ расшифровывает код операции и выдает в АЛУ сигнал выполнить соответствующую операцию над данными;
6. результат операции либо остается в процессоре, либо отправляется в память, если в команде был указан адрес результата.

Операционная система – комплекс программ, управляющих базовыми и периферийными устройствами компьютера и обеспечивающих правильную загрузку других программ (MS-DOS, Windows, UNIX, MacOS и др.).

Для обслуживания периферийных устройств в состав операционной системы входят специальные программы – *драйверы* устройств.

Современные *системы программирования* (например, Borland Delphi, Microsoft Visual Basic, Borland C++) кроме трансляторов включают в себя интегрированную среду разработки: средства создания и редактирования текстов программ; обширные библиотеки стандартных программ и функций; отладочные программы и утилиты; встроенный ассемблер; справочную службу и т.д.

Системное программирование – процесс разработки операционных систем, утилит и трансляторов.

Прикладное программирование – процесс разработки прикладных программ.

Двоичная система счисления – *позиционная* (основание - 2).

Перевод *целого числа* из двоичной системы в десятичную:

$$(1101)_2 = 1*2^0 + 0*2^1 + 1*2^2 + 1*2^3 = (13)_{10}$$

Перевод *целого числа* из десятичной системы в двоичную осуществляется последовательным делением на 2. В качестве остатка от деления получается очередная цифра двоичного числа, начиная с младшей.

13/2 = 6 (остаток 1 - младшая цифра), 6/2 = 3 (0),
3/2 = 1 (1), 1/2 = 0 (1). Результат - $(1101)_2$

Перевод *дроби* из двоичной системы в десятичную:

$$(0.1011)_2 = 1*2^{-1} + 0*2^{-2} + 1*2^{-3} + 1^*2^{-4} =$$
$$1/2 + 1/8 + 1/16 = 0.5 + 0.125 + 0.0625 = (0.6875)_{10}$$

Перевод *дроби* из десятичной системы в двоичную осуществляется умножением на 2. Целая часть полученного числа – очередная цифра двоичного, начиная с первой цифры после запятой:

0.6875*2 = 1.375 (первая цифра - 1), 0.375*2 = 0.75 (0), 0.75*2 =
1.5 (1), 0.5*2 = 1.0 (1). Результат - $(0.1011)_2$

Восьмеричная система счисления – *позиционная* (основание – 8, используются цифры 0 1 2 3 4 5 6 7).

Шестнадцатеричная система счисления – *позиционная* (основание – 16, используются цифры 0 1 2 3 4 5 6 7 8 9 и первые буквы латинского алфавита A B C D E F).

Перевод восьмеричных и шестнадцатеричных чисел в двоичную систему (и обратно) осуществляется заменой каждой цифры эквивалентной ей двоичной триадой (тройкой цифр) или тетрадой (четверкой цифр).

$$(537.1)_8 = \begin{matrix} 101 & 011 & 111. & 001 \\ 5 & 3 & 7. & 1 \end{matrix} = (101\ 011\ 111.001)_2$$

$$(1A3.F)_{16} = \begin{matrix} 0001 & 1010 & 0011. & 1111 \\ 1 & A & 3. & F \end{matrix} = (1\ 1010\ 0011.1111)_2$$

10	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111	10000
8	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17	20
16	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10

Представление (вид) числа в памяти определяется тем **типом**, к которому он принадлежит. В частности, тип числа задает *количество двоичных разрядов*, отводимых под хранение числа.

Для хранения числа в памяти отводится *целое* число ячеек емкостью 1 байт – 8 бит (разрядов).

Целые беззнаковые типы (представление числа 5)

BYTE (1 байт, 0...255)

```
0000 0101
```

WORD (2 байта, 0...65535)

```
0000 0000 0000 0101
```

Для хранения чисел со знаком старший разряд отводится под знак (0 – если число положительное, 1 – если отрицательное).

Целые знаковые типы (представление числа 5)

SHORTINT (1 байт, -128...127)

```
0 000 0101
```

INTEGER (2 байта, -32768...32767)

```
0 000 0000 0000 0101
```

Отрицательные числа представлены в виде **дополнительного кода**.

Для получения дополнительного кода необходимо сначала поменять все разряды числа на обратные, а затем к полученному результату прибавить 1.

Целые знаковые типы (представление числа -5)

SHORTINT (1 байт, -128...127)

```
1 111 1011
```

INTEGER (2 байта, -32768...32767)

```
1 111 1111 1111 1011
```

Все математические операции с числами основаны на *сложении*.

Сложение осуществляется *поразрядно* с соблюдением дополнительного правила: "Если в результате сложения двух соответствующих разрядов чисел получилось число большее или равное основанию системы счисления (2), то следует из полученного числа отнять основание системы счисления и записать в строке итога полученный результат. Кроме того, необходимо запомнить единицу с тем, чтобы добавить ее при сложении следующего разряда".

```
0 1 0 0  1 0 0 1     (73)10
  + 0 0 1 0  0 1 0 1     (37)10
_____________________________
0 1 1 0  1 1 1 0     (110)10
```

Вычитание заменяется сложением чисел, одно из который берется с обратным знаком (используется дополнительный код).

```
  0  0 0 0   0 1 1 1    (7)10
+ 1  1 1 1   1 0 1 1    (-5)10
-------------------------------
  0  0 0 0   0 0 1 0    (2)10
```

Умножение заменяется сложением чисел, сдвинутых на разное число двоичных разрядов подобно тому, как это делается при умножении чисел «в столбик». **Деление**, соответственно, выполняется через вычитание (путем многократного прибавления к делимому дополнительного кода делителя). **Возведение в степень** выполняется через умножение и т.д.

Современные компьютеры помимо центрального процессора оснащаются **математическим сопроцессором** – специальным устройством, выполняющим математические операции с плавающей точкой, что позволяет разгрузить центральный процессор. Его использование позволяет ~ на 80 % сократить время выполнения таких операций как умножение и возведение в степень.

Работа компьютера (в том числе выполнение арифметических операций) основана на *логических действиях*.

Возможны только четыре комбинации соответствующих разрядов при сложении двух чисел, представленных в двоичном виде.

Разряд I слагаемого	Разряд II слагаемого
0	0
1	0
0	1
1	1

Правила по установке соответствующего разряда итогового числа построено по законам логики:

"Если складываются разряды с равным состоянием (ноль с нулем или единица с единицей), то итоговый разряд устанавливается равным нулю. В противном случае, он устанавливается равным единице. Если складываются два разряда, равные единице, то вырабатывается сигнал переноса единицы в следующий разряд".

Логическая конструкция "*Если* условие обращается в истину, *то* выполнить некую последовательность действий" называется **импликацией**.

В практическом программировании применяют более сложную конструкцию: "*Если* условие истинно, *то* выполнить последовательность действий №1, *иначе* (то есть если условие ложно) выполнить последовательность действий №2". Реализуется с помощью логического оператора: **if – then – else**.