

Стиль программирования



Утверждения по поводу стиля

1. Для любого языка существует один распространённый **стиль программирования**, или число таких стилей невелико.
2. Не существует и никогда не будет существовать стиля программирования, который обладал бы очевидными преимуществами по сравнению с любым из стандартных стилей.
3. В любом нетривиальном проекте желательно применять стандартный стиль программирования.
4. Каждый член команды должен писать в

```
// Пример простой программы на языке Си
// Стиль – для новичков
#include <stdio.h>
#include <math.h>

int main(int argc, _TCHAR* argv[])
{
    float x = 0;
    float y = 0;

    printf("Введите X: ");
    scanf("%f", &x);

    if (x < 0)
    {
        y = -1;
    }
    else if (x == 0)
    {
        y = 0;
    }
    else
    {
        y = sqrt(x);
    }
    printf("Результат: y = %.2f", y);
    getchar();

    return 0;
}
```

Tab
или
4 пробела

```
// Пример простой программы на языке Си
// Стиль - для новичков
#include <stdio.h>
#include <math.h>

int main(int argc, _TCHAR* argv[])
{
    float x = 0;
    float y = 0;

    printf("Введите X: ");
    scanf("%f", &x);

    if (x < 0)
    {
        y = -1;
    }
    else if (x == 0)
    {
        y = 0;
    }
    else
    {
        y = sqrt(x);
    }

    printf("Результат: y = %.2f", y);
    getchar();

    return 0;
}
```

```
// Пример простой программы на языке Си
// Стиль - для новичков
#include <stdio.h>
#include <math.h>

int main(int argc, _TCHAR* argv[])
{
    float x = 0;
    float y = 0;

    printf("Введите X: ");
    scanf("%f", &x);

    if (x < 0)
    {
        y = -1;
    }
    else if (x == 0)
    {
        y = 0;
    }
    else
    {
        y = sqrt(x);
    }

    printf("Результат: y = %.2f", y);
    getchar();

    return 0;
}
```

```
// Пример простой программы на языке Си
// Стиль - для новичков
#include <stdio.h>
#include <math.h>

int main(int argc, _TCHAR* argv[])
{
    float x = 0;
    float y = 0;

    printf("Введите X: ");
    scanf("%f", &x);

    if (x < 0)
    {
        y = -1;
    }
    else if (x == 0)
    {
        y = 0;
    }
    else
    {
        y = sqrt(x);
    }

    printf("Результат: y = %.2f", y);
    getchar();

    return 0;
}
```

Вертикальный
«пробел», используется
для деления кода на
логические блоки

```
// Пример простой программы на языке Си
// Стиль – эконом, логический блок одной строкой
#include <stdio.h>
#include <math.h>

int main()
{
    float x = 0, y = 0;

    // %f - float, %c - char, * == ignore
    printf("Введите X: "); scanf("%f%c", &x);

    y = (x < 0) ? -1 : sqrt(x);

    printf("Результат: y = %.2f", y); getchar();

    return 0;
}
```

Пробелы

```
float x = 0;
```

```
scanf ("%f", &x);
```

```
if (x < 0)  
{
```

```
    y = -1;
```

```
}  
else if (0 == x)  
{  
    y = 0;  
}
```

```
printf("Результат: при x = %d, y = %.2f", x, y);
```

Пробелы в
условиях

Пробелы между
if, for, while и
скобкой:

Пробелы в
операторах
присваивания и
математически
выражениях:
Пробелы между
параметрами
функций


```
//Пример “профи”-стиля
#include <stdio.h>
#include <math.h>

int main()
{
    float x = 0, y = 0;

    printf("Введите X: ");
    scanf("%f", &x);

    if      (x < 0)  y = -1;
    else if (x == 0) y = 0;
    else          y = sqrt(x);

    printf("Результат: y = %.2f", y);
    getchar();

    return 0;
}
```

//Пример отсутствия стиля программирования

```
#include <stdio.h>
#include <math.h>

int main(int argc, _TCHAR* argv[]){
    float x=0;float y=0;
    { printf("Введите X: "); }

    scanf("%f",&x);
    if(x<0)
    {y=-1;}
    else if(x==0) y=0;
    else
    {
        y=sqrt(x);}

    printf("Результат: y = %.2f",y);
    getchar();
    return 0;
}
```

Аппаратное обеспечение (aka Hardware, железо)

Представляет собой фундамент (платформу) на котором строится **программное обеспечение**



Любое электронное или механическое устройство (**device**) как в сборке так и по частям можно называть аппаратным обеспечением

Modern hardware types of computer

- Калькулятор
- **Персональный компьютер** (ПК, Personal Computer, PC)
 - **Настольный компьютер** (Desktop PC)
 - **Ноутбук** (Notebook, Лэптоп, Mobile PC)
 - **Нетбук** (Netbook)
 - **Игровая приставка** (Игровая консоль)
 - **Карманный компьютер** (КПК, PDA)
 - **Коммуникатор** (*Communicator, PDA Phone*)
 - **Смартфон** (Smartphone)
 - **Планшетный ПК** (Tablet PC)
- **Рабочая станция** (workstation)
- **Сервер** (server)
- **Суперкомпьютер**

Архитектура фон Неймана

1. Принцип **двоичности** - для представления **данных** и **команд** используется **двоичная система счисления**.
2. Принцип **последовательного программного управления** - **программа** состоит из набора **команд**, которые **выполняются процессором** последовательно, одна после завершения другой.
3. Принцип **однородности памяти** - Как **программы (команды)**, так и **данные** хранятся в одной и той же **памяти**. Над **командами** можно **выполнять** такие же **действия**, как и над **данными**.
4. Принцип **адресуемости памяти** - структурно **основная память** состоит из пронумерованных **ячеек**; **процессору** в произвольный момент времени **доступна** любая **ячейка**.
5. Принцип **условного перехода** - возможно присутствие в **программе команд**, которые изменяют последовательность **выполнения программы**

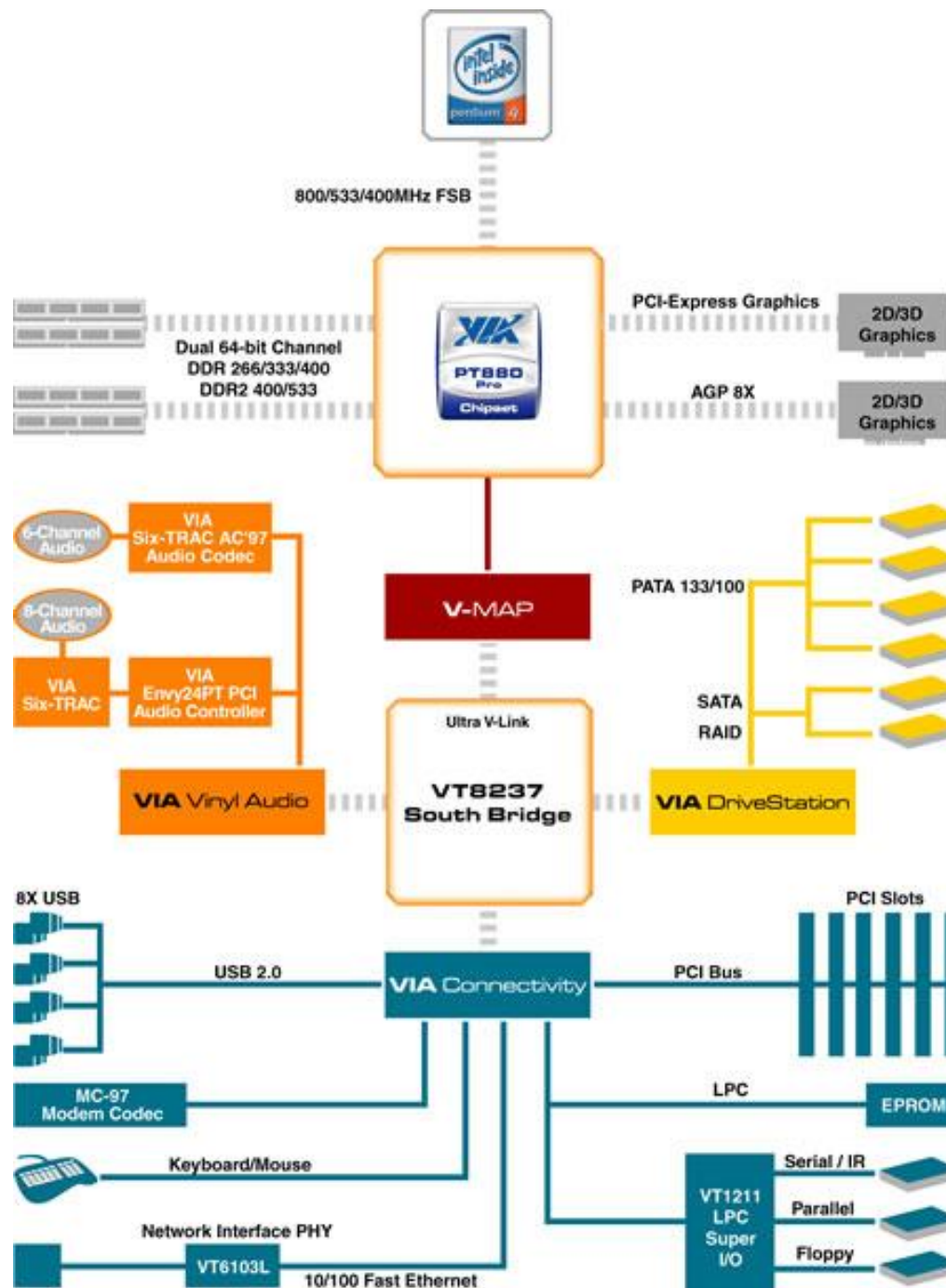
Главные компоненты компьютера

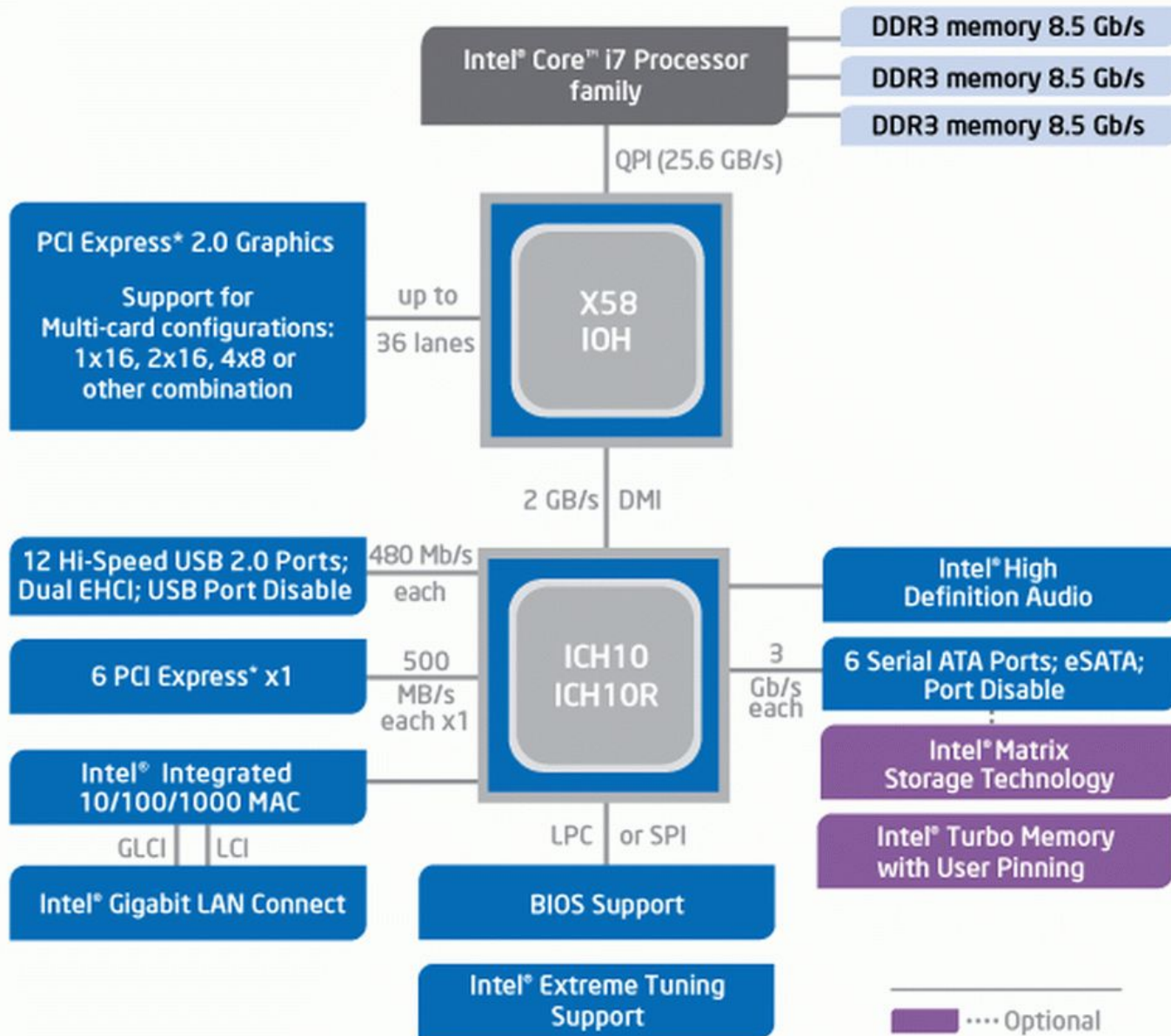
Процессор (Central Processing Unit) – выполняет команды, производит вычисления

Оперативная память (Random Access Memory) предназначена для хранения программных команд и результатов вычислений.

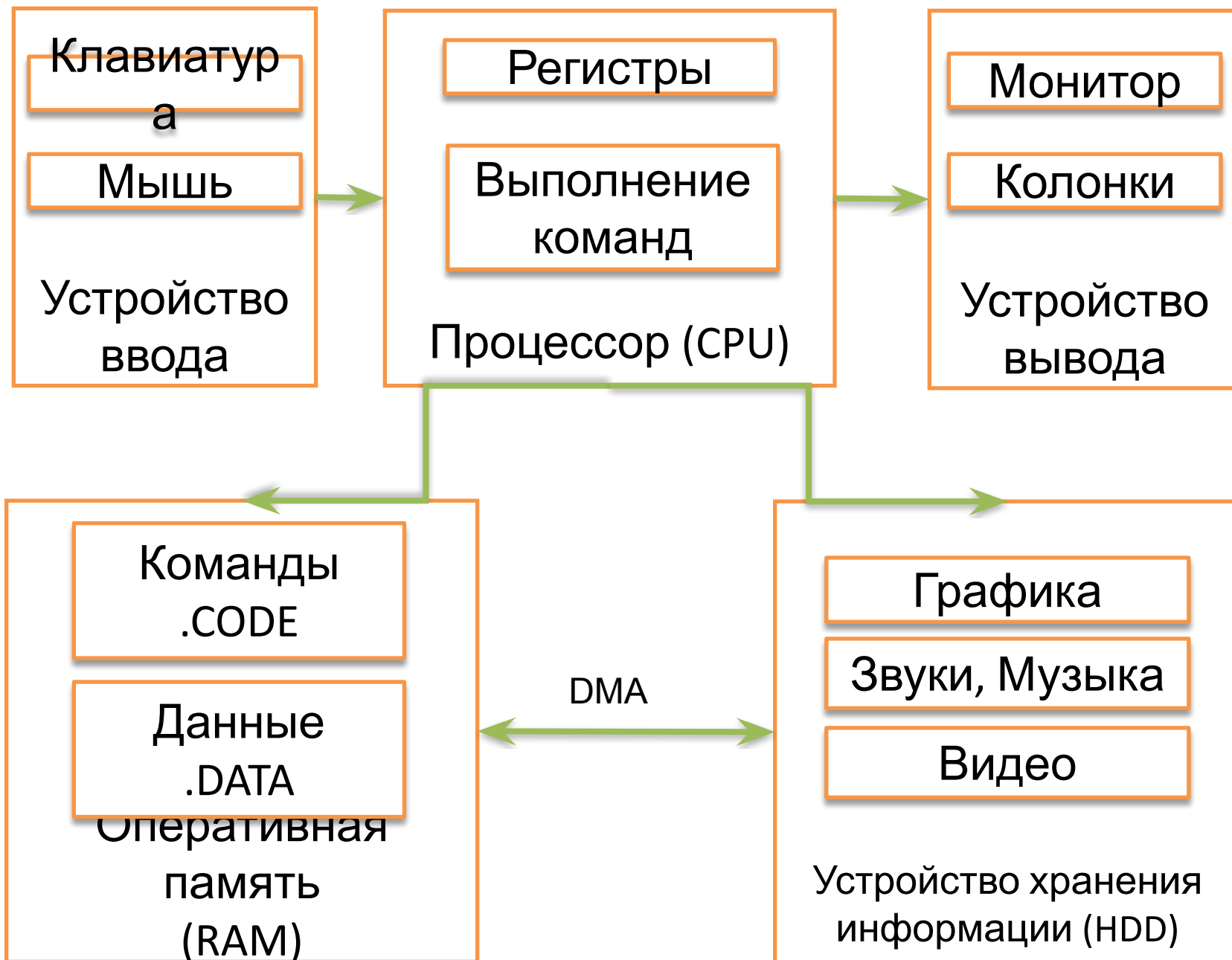
Устройством ввода (Input Device) называется любое устройство, позволяющее человеку передать информацию в компьютер.

Устройством вывода (Output Device) называется любое устройство, с помощью которого компьютер выдаёт информацию человеку.



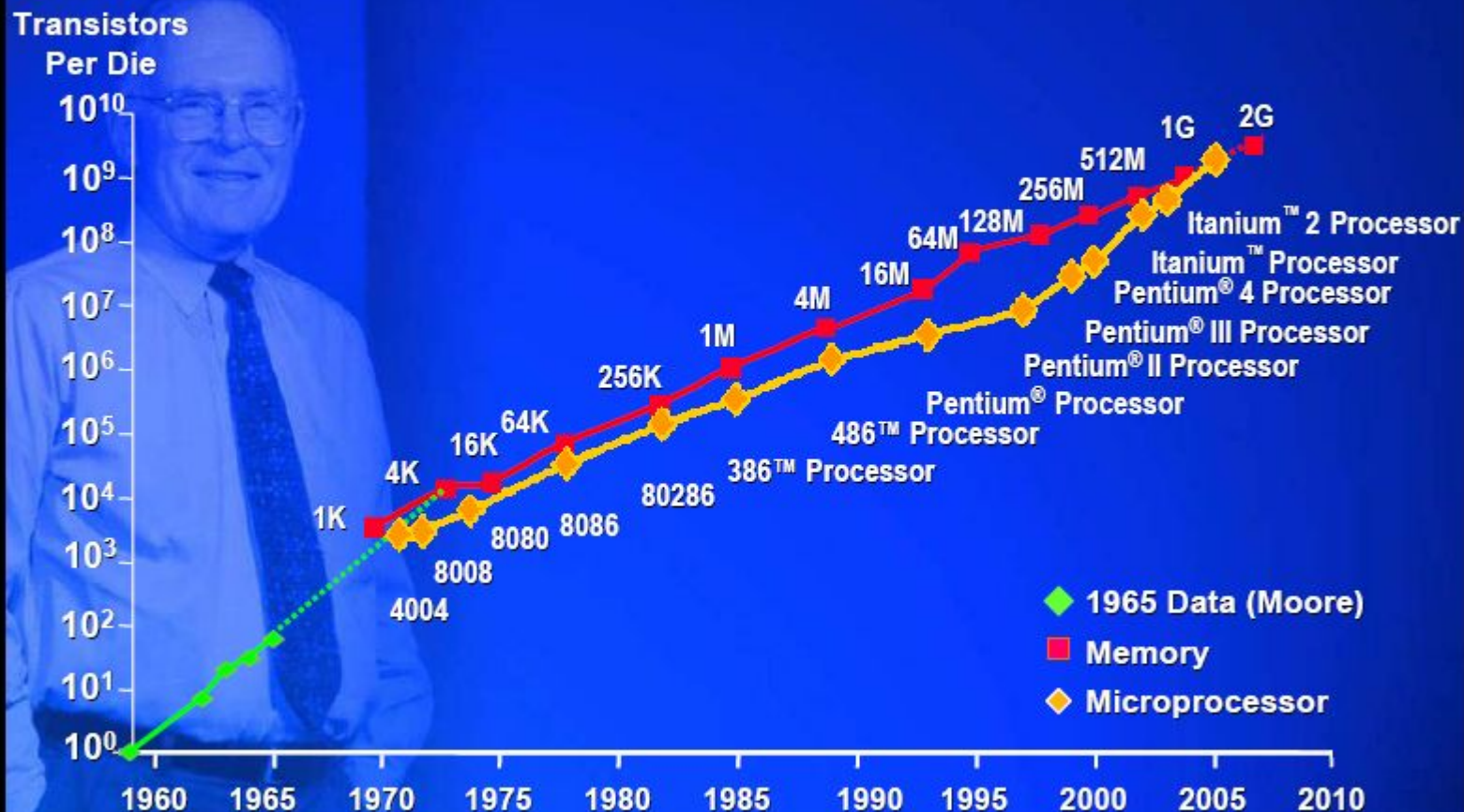


Упрощённая **Архитектура** компьютера

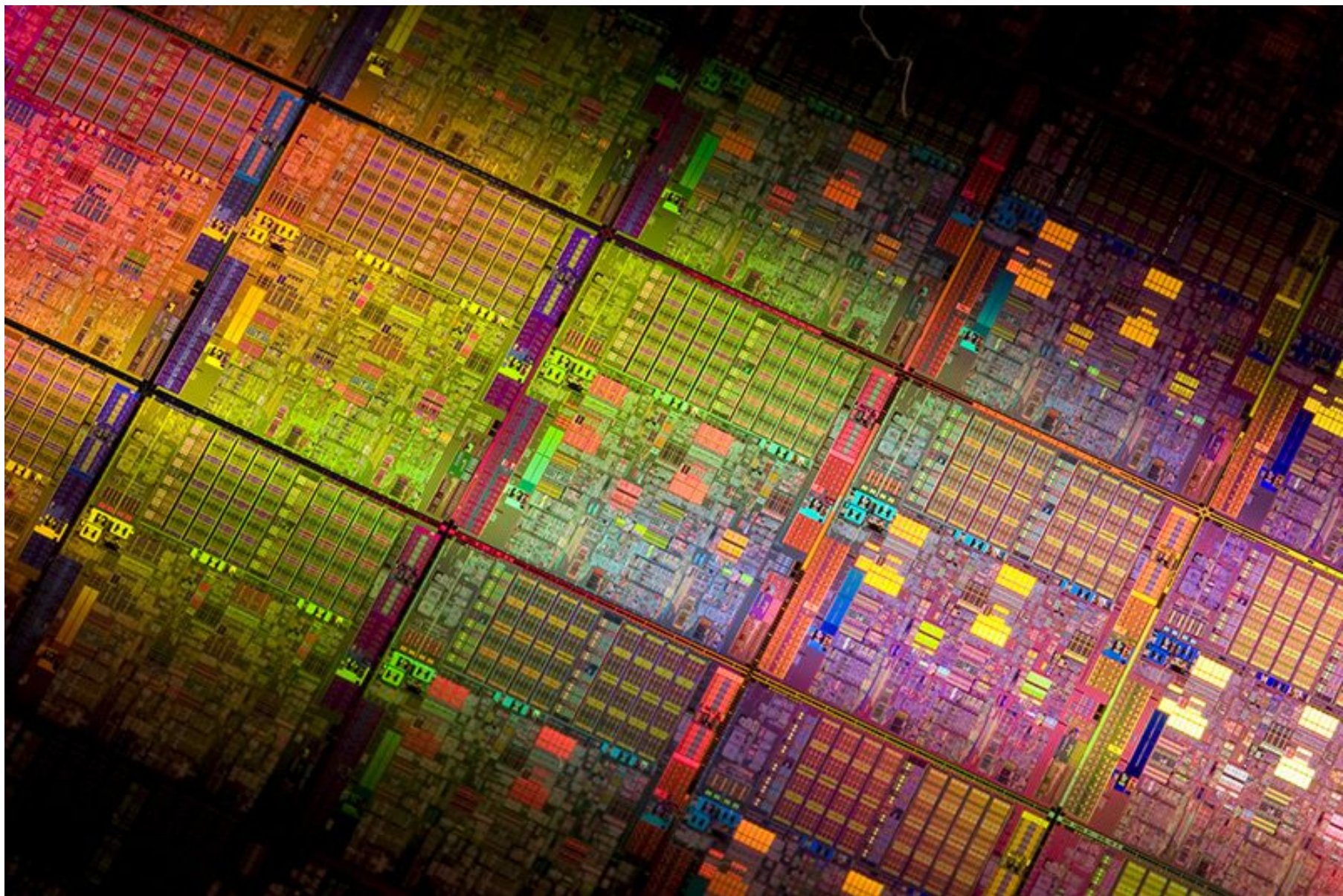


**Закон Мура - число транзисторов на кристалле
будет удваиваться каждые 24 месяца**

Moore's Law - 2005



Source: Intel



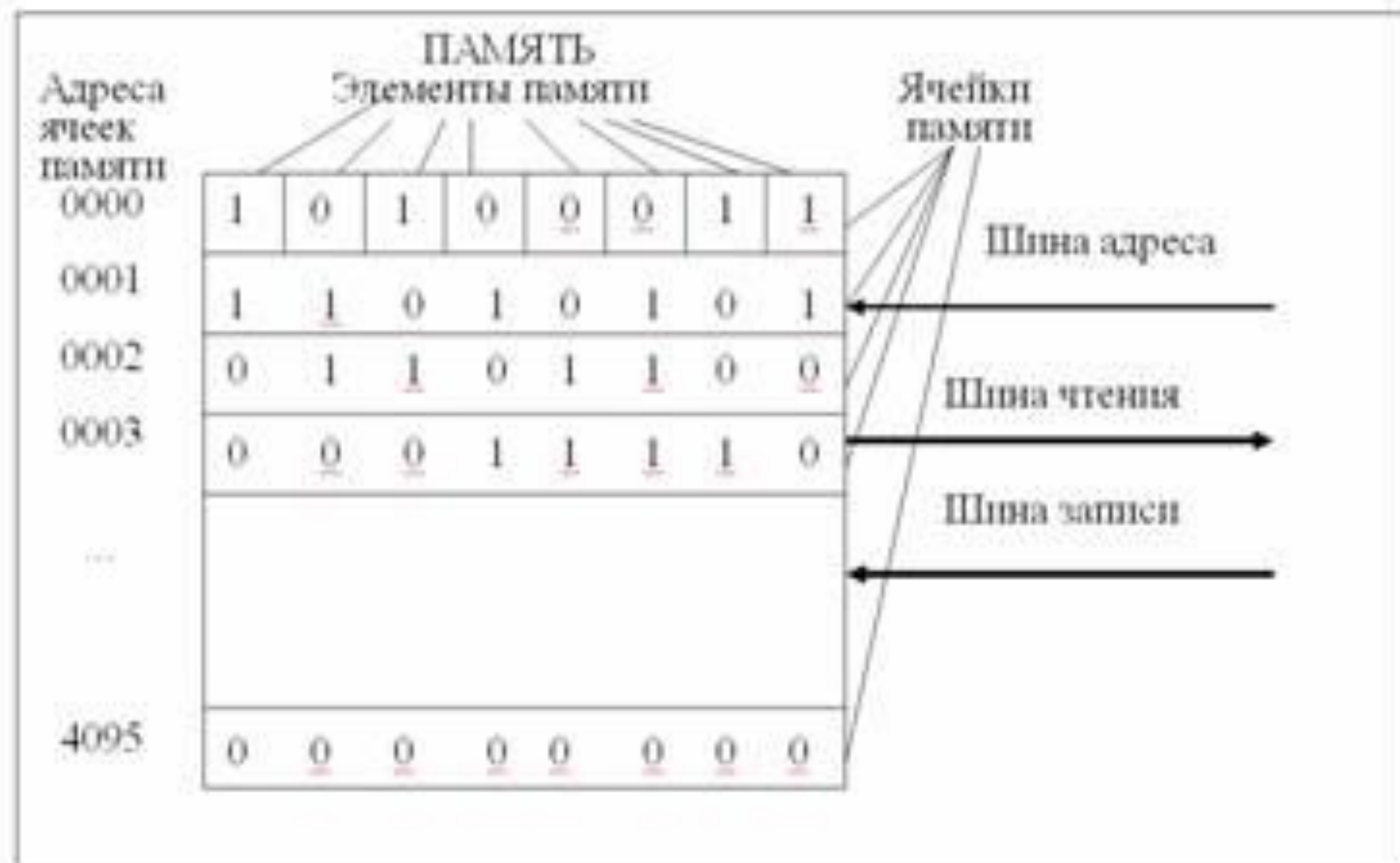


Рис 4.2 Организация оперативной памяти ЭВМ

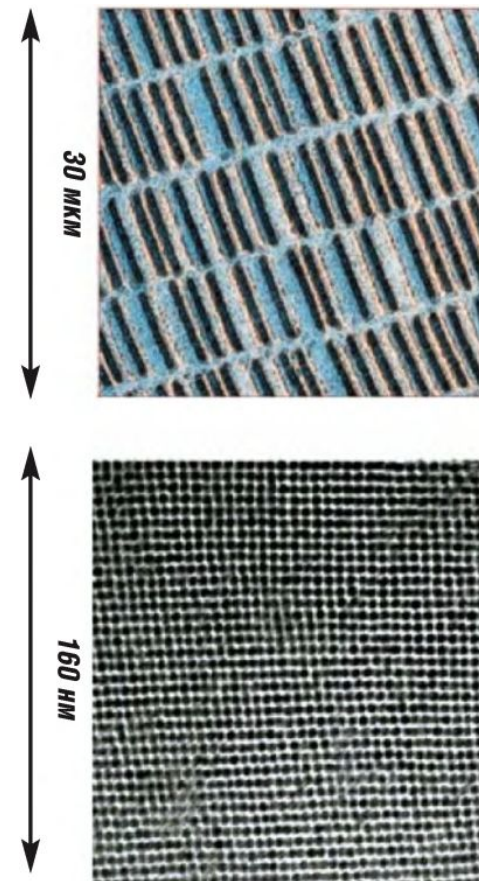
DEKSI Hard Disk Manager

PROFESSIONAL



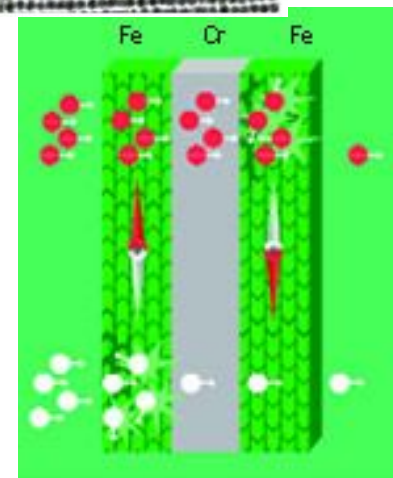
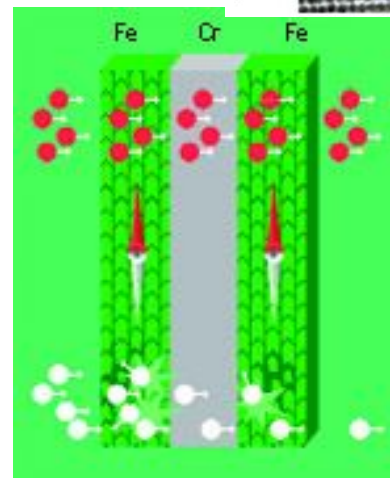
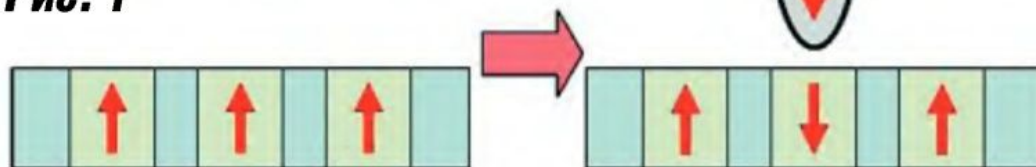
v2.90
(3586)

DEK
Software International



Намагниченная игла

Рис. 1



устройства хранения информации

«**Внешняя память**» – это память, которая используется для постоянного хранения данных по окончании или до начала работы компьютера. Во внешнем запоминающем устройстве данные хранятся блоками, именуемыми **файлами**. Файлы могут быть произвольного размера и хранить любую информацию, в том числе и программы.

С одним компьютером может быть соединено любое количество внешних запоминающих устройств.

