Архитектура компьютера

(Организация ЭВМ)

Компьютер

- Компьютер – устройство для решения задач, выполняющее команды
- Программа – последовательность команд
- Машинный язык – команды, выполняемые электронными компонентами компьютера
  - Сложить 2 числа
  - Проверить выполнение условия
  - Перейти к другой команде

Компьютер

- Электронная вычислительная машина (ЭВМ)

- **Происхождение**
- англ. ***computer*** (вычислитель; человек, выполняющий вычисления)
  = ***to compute*** (вычислять) + ***-er***
- лат. ***computare*** (сосчитать, складывать)
  = ***com-*** (вместе) + ***putare*** (считать)

Вычислительная машина

- Любые действия компьютера сводятся к действиям над числами
  - Вычисления
  - Редактирование текста
  - Распечатка на принтере
  - Обработка фотографий
  - Проигрывание музыки
  - Воспроизведение фильма
  - Компьютерные игры

Выполнение программы

- Программа
  - **Исходный текст** программы, понятный человеку
  - **Исполняемый код** – машинный язык, понятный компьютеру
- Выполнение программы
  - **Трансляция** – замена каждой команды исходного текста на набор машинных команд и создание новой программы на машинном языке
  - **Интерпретация** – пошаговая замена каждой команды исходного текста на набор машинных команд и пошаговое выполнение

Происхождение

- Трансляция
  - англ. ***to translate*** (переводить с одного языка на другой)
  - лат. ***translatus, transfero*** (переносить, перевозить)
    = ***trans-*** (пере-, через-) + ***latus*** (носить)
- Интерпретация
  - англ. ***to interpret*** (переводить устно; разъяснять)
  - лат. ***interpretari*** (толковать, переводить)
    = ***inter-*** (между) + санскр. ***prath-*** (распространять, продавать)

Выполнение программы

- Программа хранится на внешнем носителе (внешней памяти – диске, флешке) в виде файла
- Пользователь запускает программу на выполнение
- Программа копируется в оперативную память
- Процессор читает очередную команду из памяти
- Процессор выполняет прочитанную команду
- Процессор читает очередную команду из памяти
- …

Задание

- Схема устройства компьютера
- Процесс выполнения программы

Процессор

- Центральный процессор
- Центральное процессорное устройство (ЦПУ)
- Processor
- Central processing unit (CPU)
- Электронное устройство, выполняющее машинные инструкции (команды исполняемого кода программы):
  - Арифметические и логические операции
  - Операции ввода-вывода

Процессор

- **Происхождение**
- англ. ***processor*** (обрабатывающее устройство)
  = ***to process*** (обрабатывать) + ***-or***
- лат. ***processus*** (продвижение, движение вперед)
- лат. ***procedere*** (двигаться вперед)
- лат. ***pro-*** (вперед) + ***cedere*** (идти)

Food *Processor*

- Кухонный комбайн
- Устройство для ***обработки*** продуктов питания
  - Электродвигатель
  - Съемные насадки

- Photo by Donovan Govan
- Источник: https://commons.wikimedia.org/w/index.php?curid=119985

Компоненты процессора

1. **Устройство управления**
   (Блок управления)

   Управляет выполнением программы

2. **Арифметико-логическое устройство**

   (Функциональное устройство)

   (Исполнительное устройство)

   Выполняет команды программы

3. **Регистры**

   Хранят данные для выполнения текущей команды

Архитектура

1. Строительное искусство (проектирования, постройки и оформления зданий)
2. Строение, организация чего-либо

- Происхождение
  - от латинского *architectura*
  - от греческого *architekton*, где
  - *arkhi* – «старший» и *tekton* – «каменщик, строитель»

Абстракция

Уровни абстракции	**«Строительные блоки»**
Программное обеспечение	Прикладные программы
Операционная система	Драйверы устройств
Архитектура компьютера	Регистры, машинные инструкции и команды ассемблера
Микроархитектура	Контроллеры
Цифровые модули	АЛУ, память
Цифровые схемы	Вентили И, ИЛИ, НЕ
Аналоговые схемы	Усилители, фильтры
Полупроводниковые устройства	Диоды, транзисторы
Физика	Молекулы, атомы, электроны

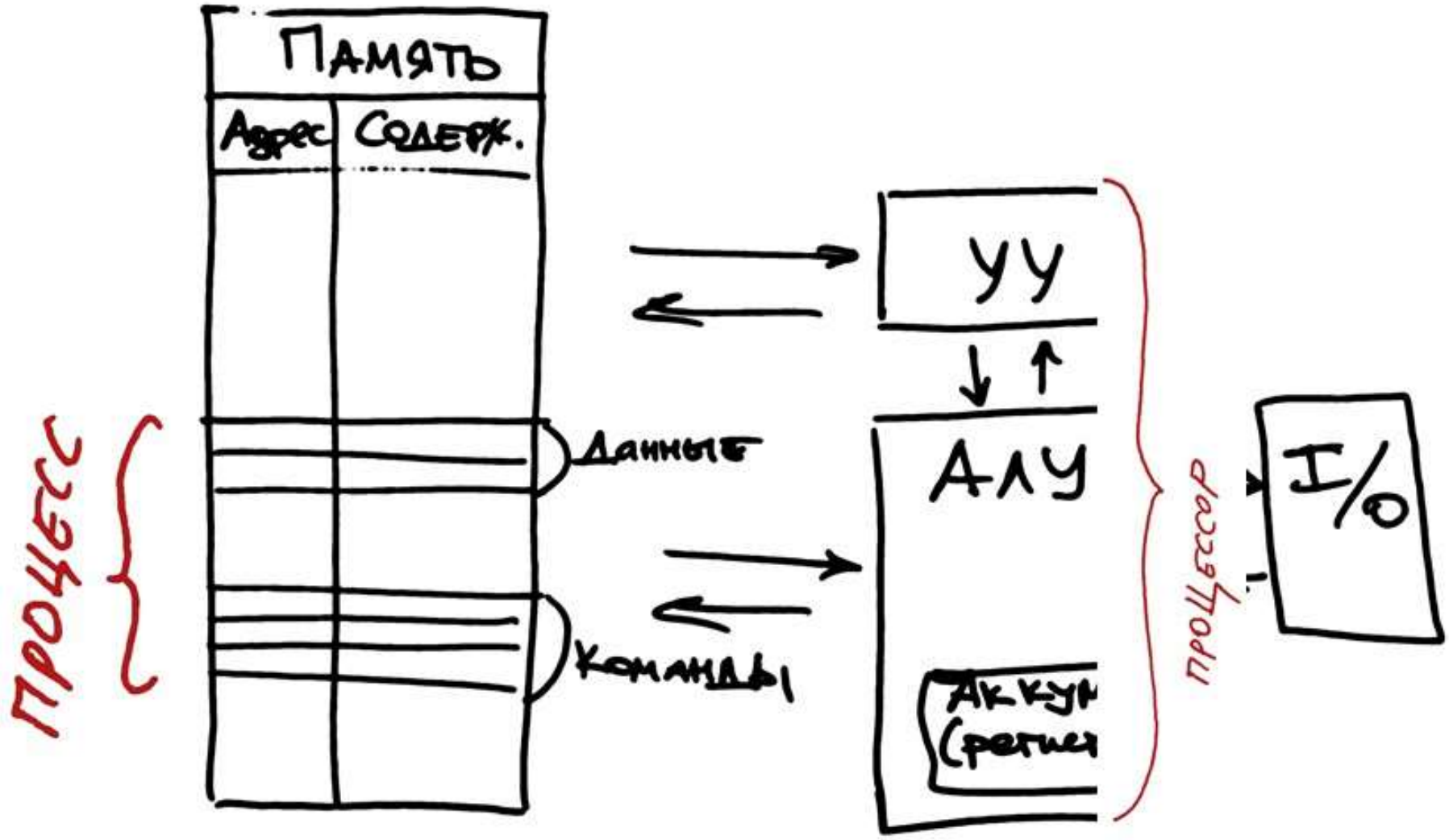
Архитектура компьютера

- Система команд
  - Машинный язык (ассемблер)
- Устройство (оборудование)
  - Аппаратура, «железо» (Аппаратный уровень)
- Специалист, работающий на одном уровне абстракции, должен представлять соседние уровни.
  - Программист
    - Прикладная программа <–> ОС и Архитектура процессора
  - Инженер-электронщик
    - Аналоговая схема <–> Цифровая схема и Транзисторы

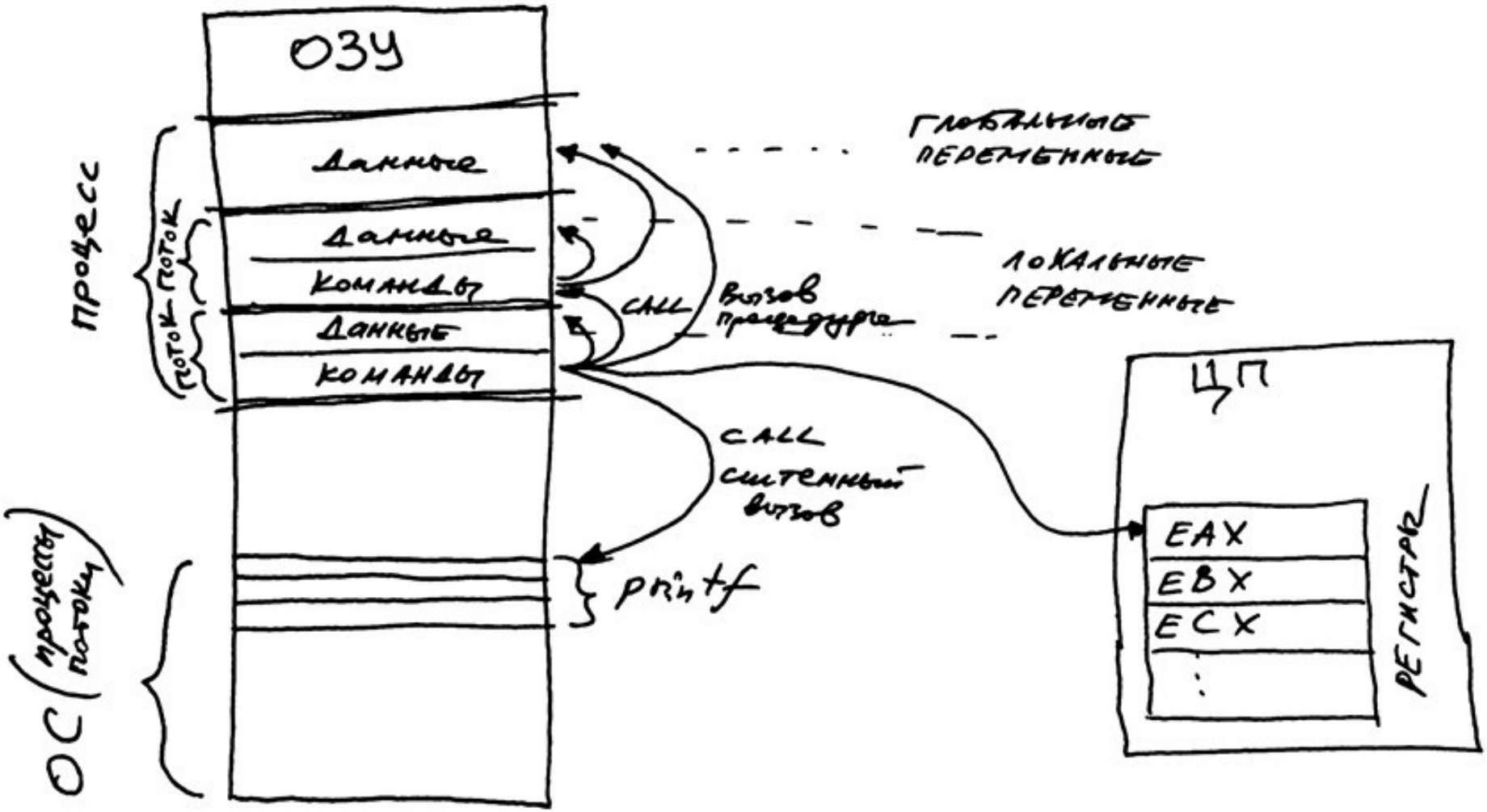
Архитектура фон Неймана

- John von Neumann (1903 – 1957)
  - Память
  - УУ
  - АЛУ
  - Аккумулятор
  - Ввод-вывод
- Принцип «хранимой программы»
  - Хранение команд и данных в одной памяти

Архитектура фон Неймана



Процессы и потоки



Машина Тьюринга (1936)

- абстрактный исполнитель (абстрактная вычислительная машина)
  - формализация понятия алгоритма
- лента, разделённая на ячейки
- управляющее устройство (головка записи-чтения)
  - перемещается по ленте
  - читает и записывает в ячейки
  - правила перехода (алгоритм)

Многоядерные процессоры

- MultiCore
- Параллельное выполнение программ
- Процессы и потоки

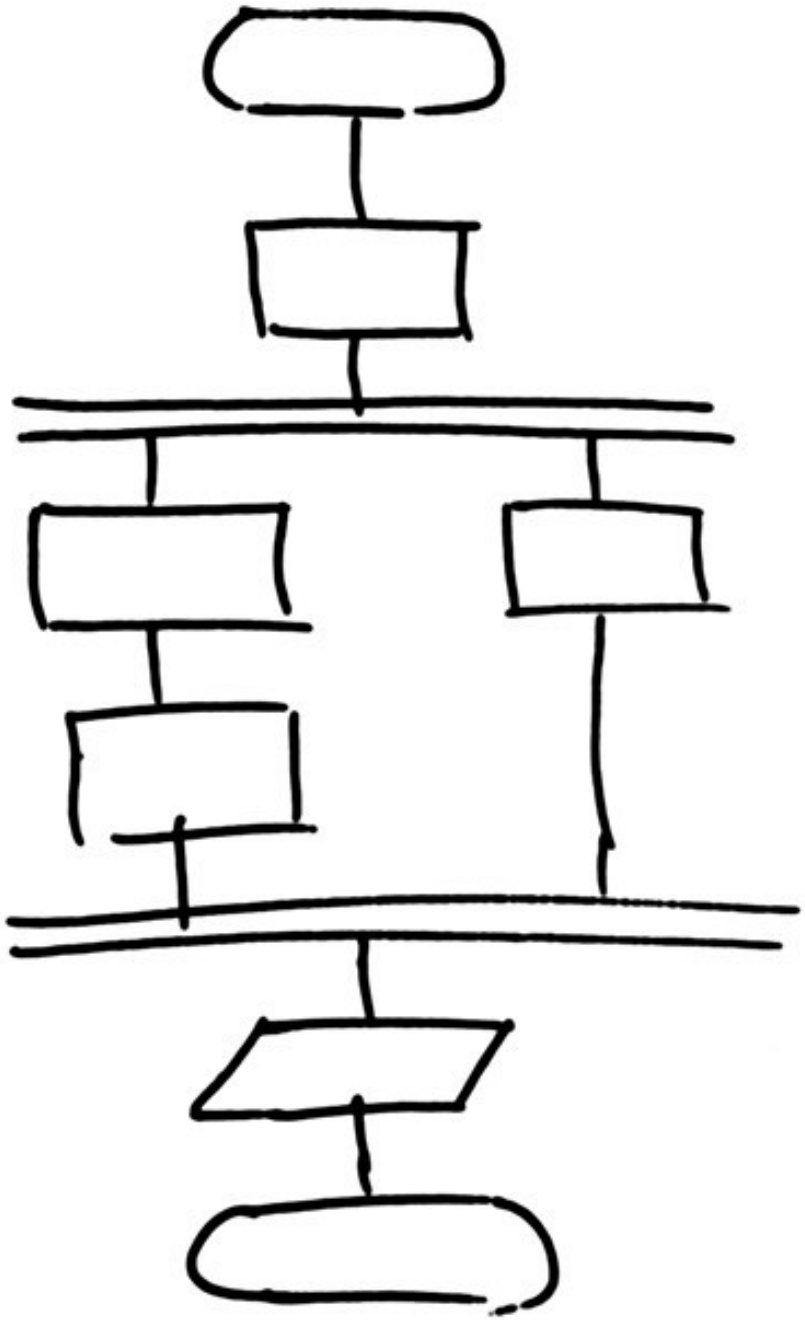
Схема алгоритма

ГОСТ 19.701-90 ЕСПД

Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения

ISO 5807:1985

Information processing -- Documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts



Процессы и потоки

- Программа (Program)
  - Исходный текст (*.C)
  - Исполняемый файл (*.EXE)
  - Последовательность команд (операторов)
  - Файл на диске
- Процесс (Process)
  - Программа в момент выполнения
  - Последовательность ячеек оперативной памяти
- Поток (Thread)
  - Часть программы, выполняемая на отдельном процессоре/ядре

Исходный текст (файл)

Process-Hello.c

This PC > D0P0dataE (E:) > Arkov > Process-Hello > Process-Hello

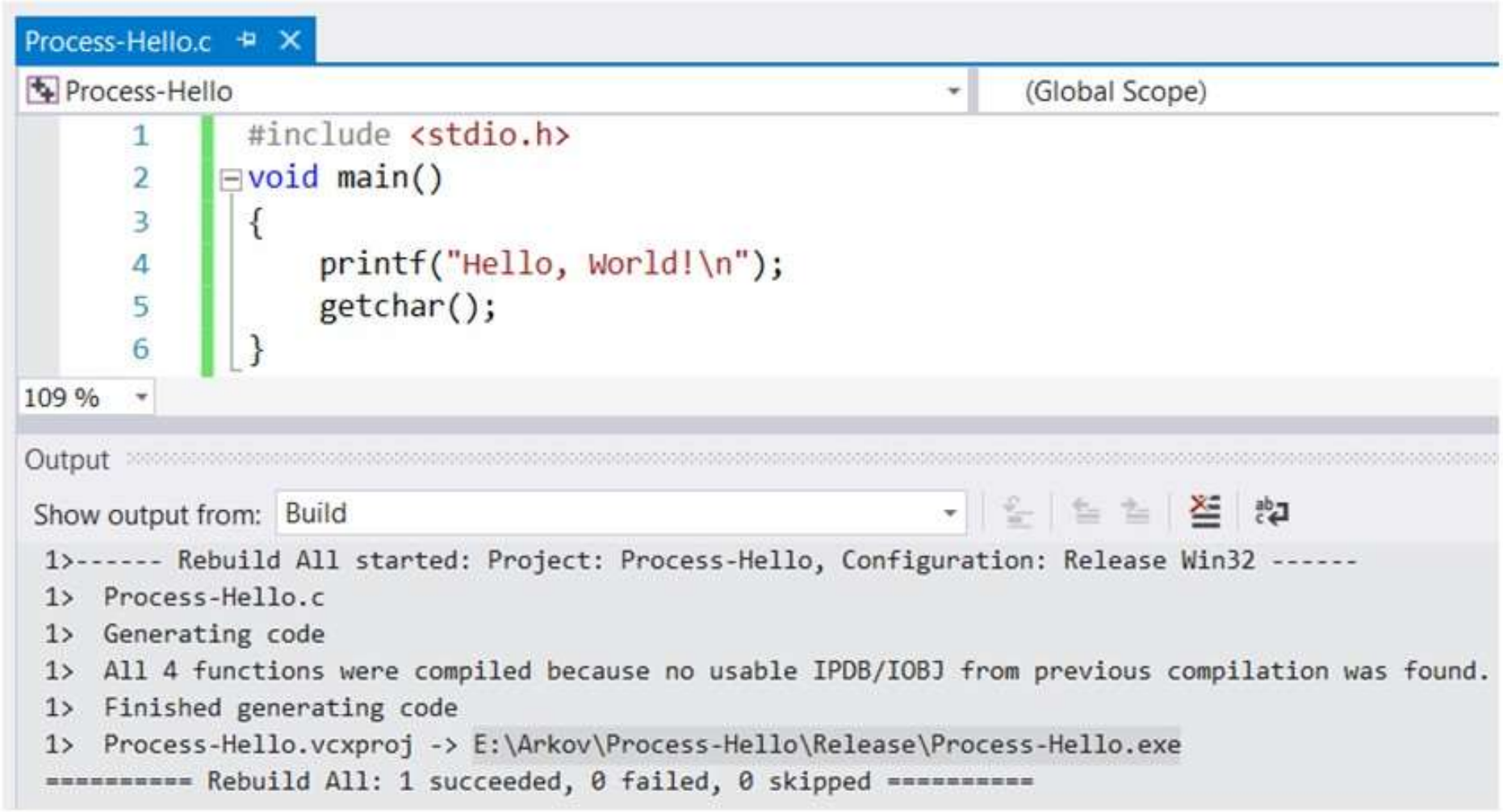
Name	Date modified	Type	Size
Process-Hello.c	2/12/2017 2:44 PM	C Source	1 KB

Process-Hello.c - Notepad

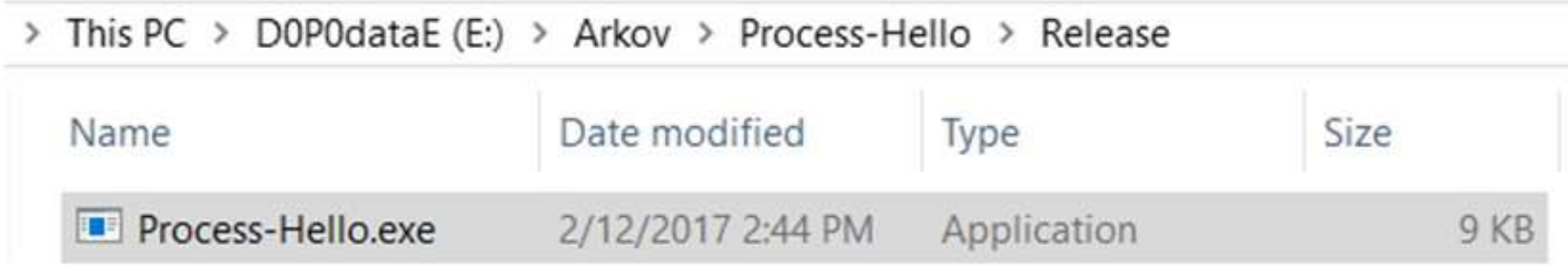
File Edit Format View Help

```
#include <stdio.h>
void main()
{
        printf("Hello, World!\n");
        getchar();
}
```

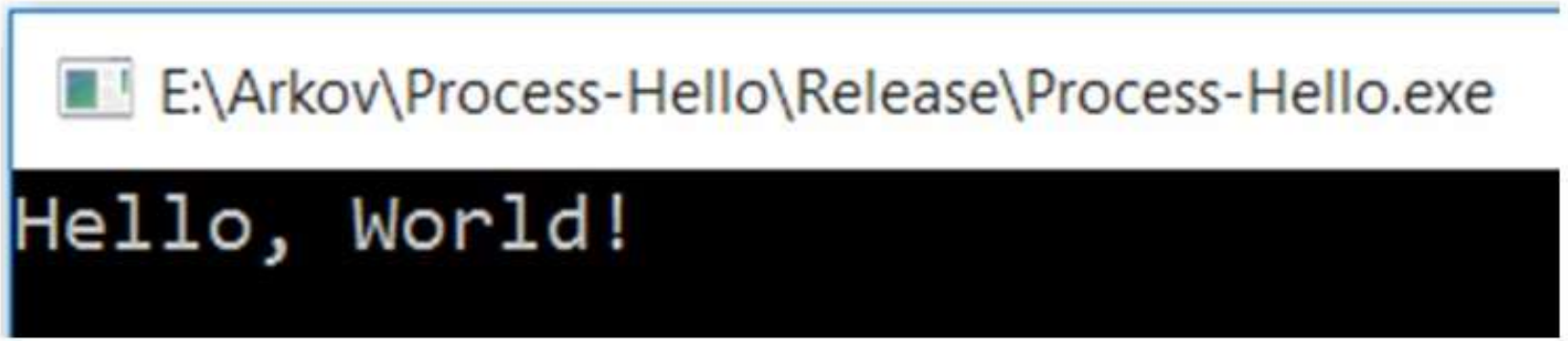
Компиляция (Hello.c -> Hello.exe)

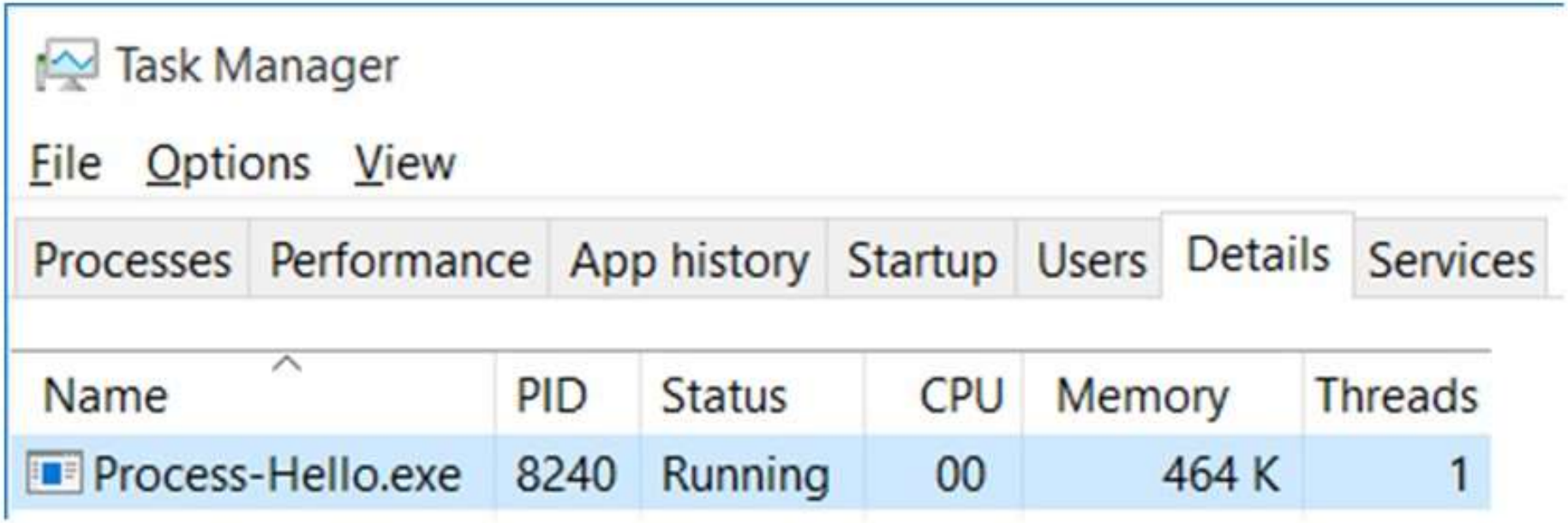


Исполняемый файл (Hello.exe)



Процесс (исполнение)





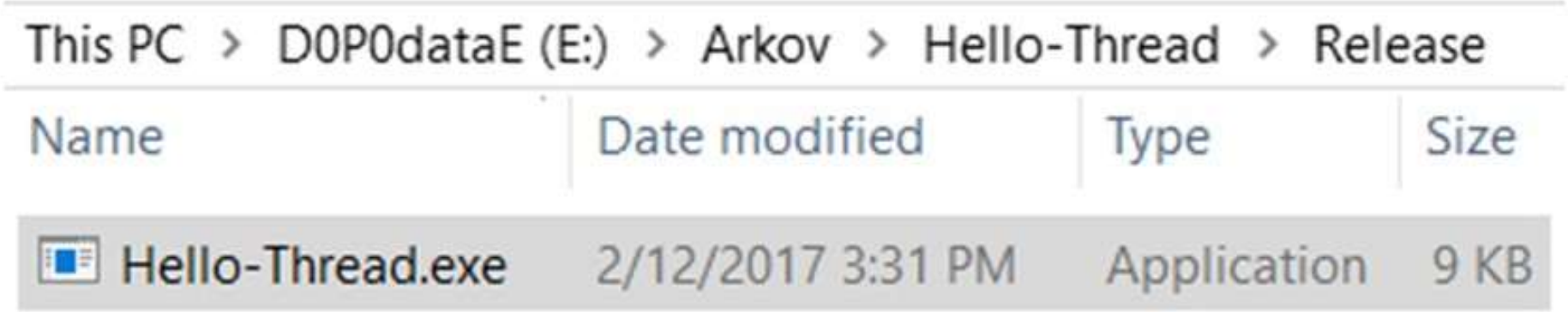
Поток (исходный текст)

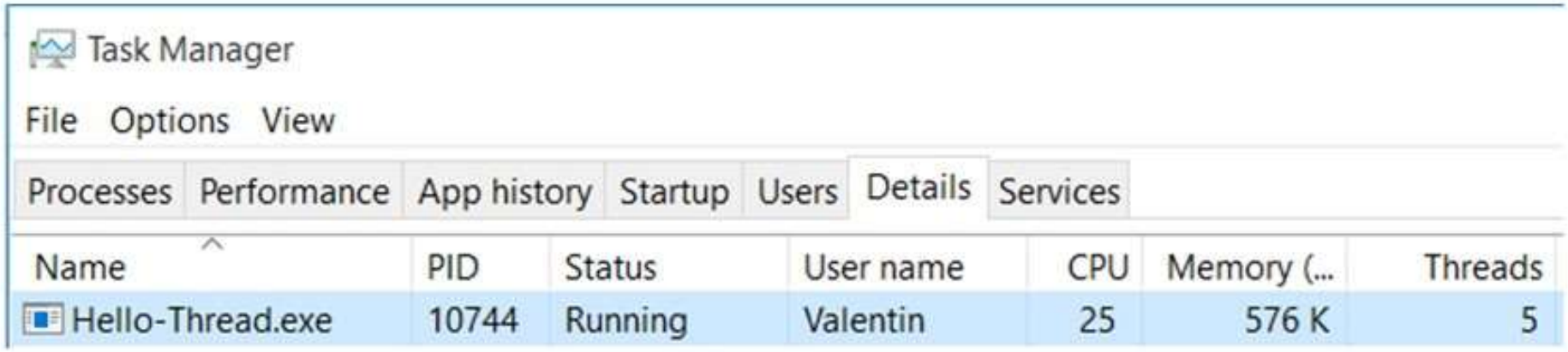
Hello-Thread.c

Hello-Thread | (Global Scope)

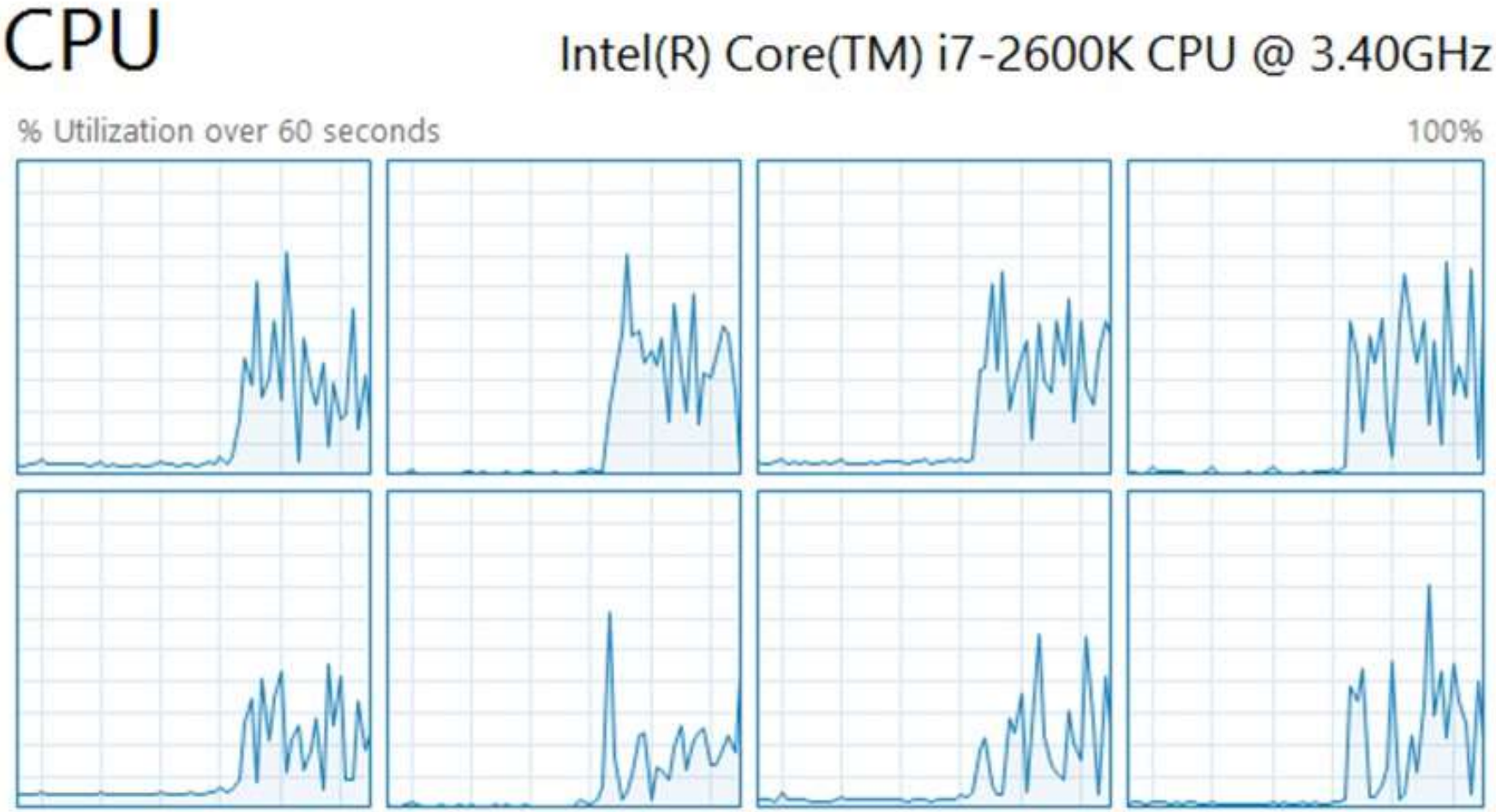
```
#include <windows.h>
#include <stdio.h>

void WINAPI ThreadProcedure()
{
    printf("Hello from Thread!\n");
    while (1);
}

void main()
{
    HANDLE Thread;
    Thread = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)ThreadProcedure, 0, 0, 0);
    printf("Hello from process!\n");
    while (1);
}
```

Поток (исполнение)





Поток (исполнение)



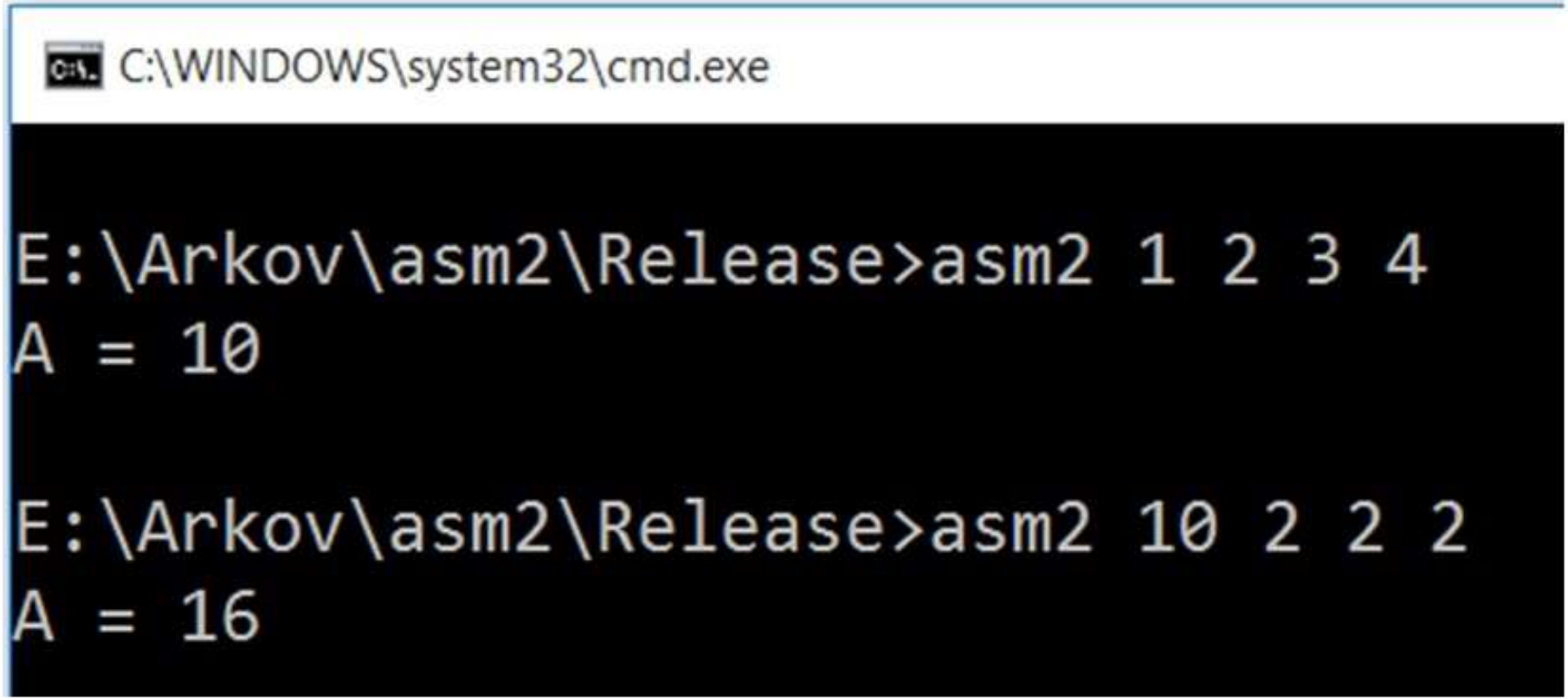
Исполняемый файл

- Исполняемый файл
  - *.EXE
  - Двоичный машинный код
  - Команды, понятные процессору
- Ассемблер
  - *.ASM
  - Текстовое представление машинного языка
  - Программа, понятная человеку
- Отладчик
  - DEBUG
  - *.EXE -> *.ASM
  - Изучение исполняемого кода

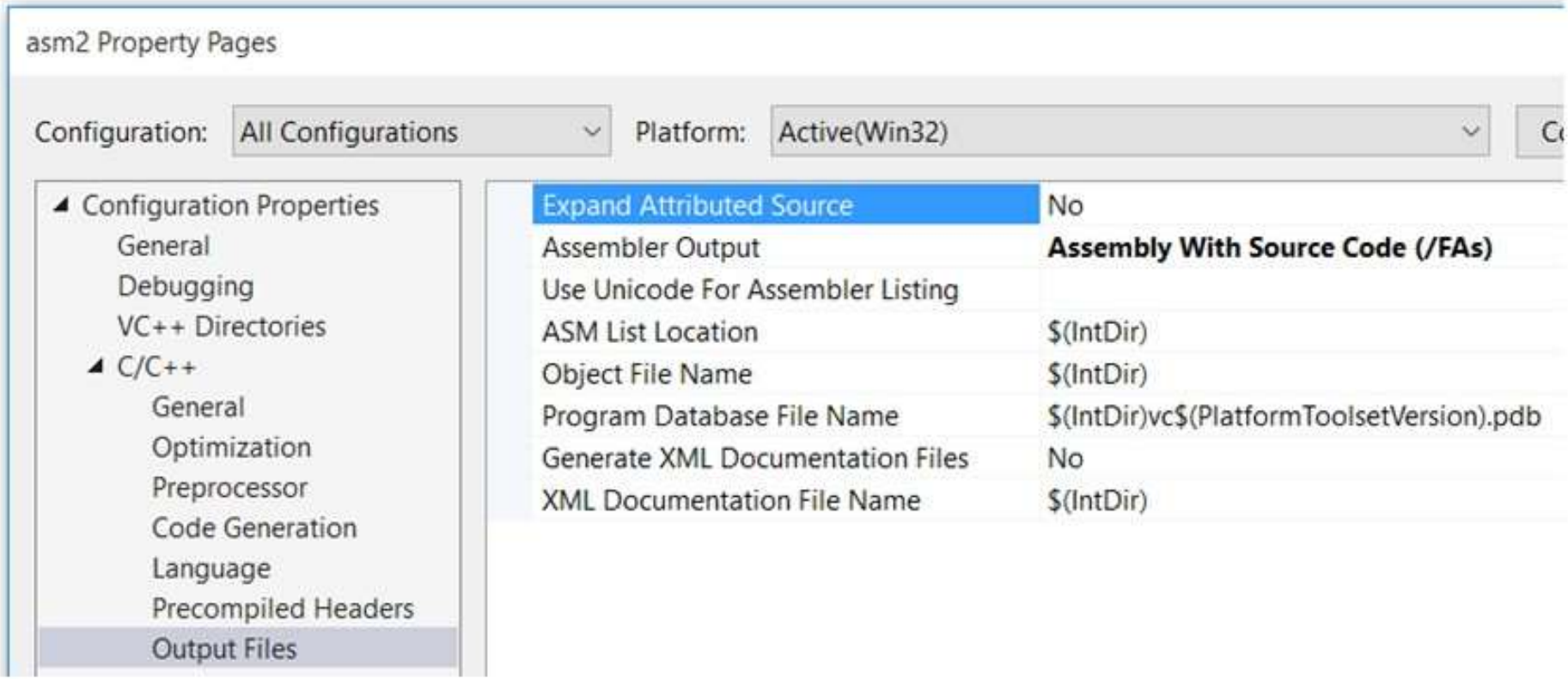
Исходный текст на языке Си: *.C

```
1     #include <stdio.h>
2
3     void main(int argc, char *argv[])
4     {
5         int A, B, C, D;
6         sscanf_s(argv[1], "%i", &A);
7         sscanf_s(argv[2], "%i", &B);
8         sscanf_s(argv[3], "%i", &C);
9         sscanf_s(argv[4], "%i", &D);
10        A = A + B + C + D;
11        printf("A = %i\n", A);
12    }
```

Выполнение *.EXE



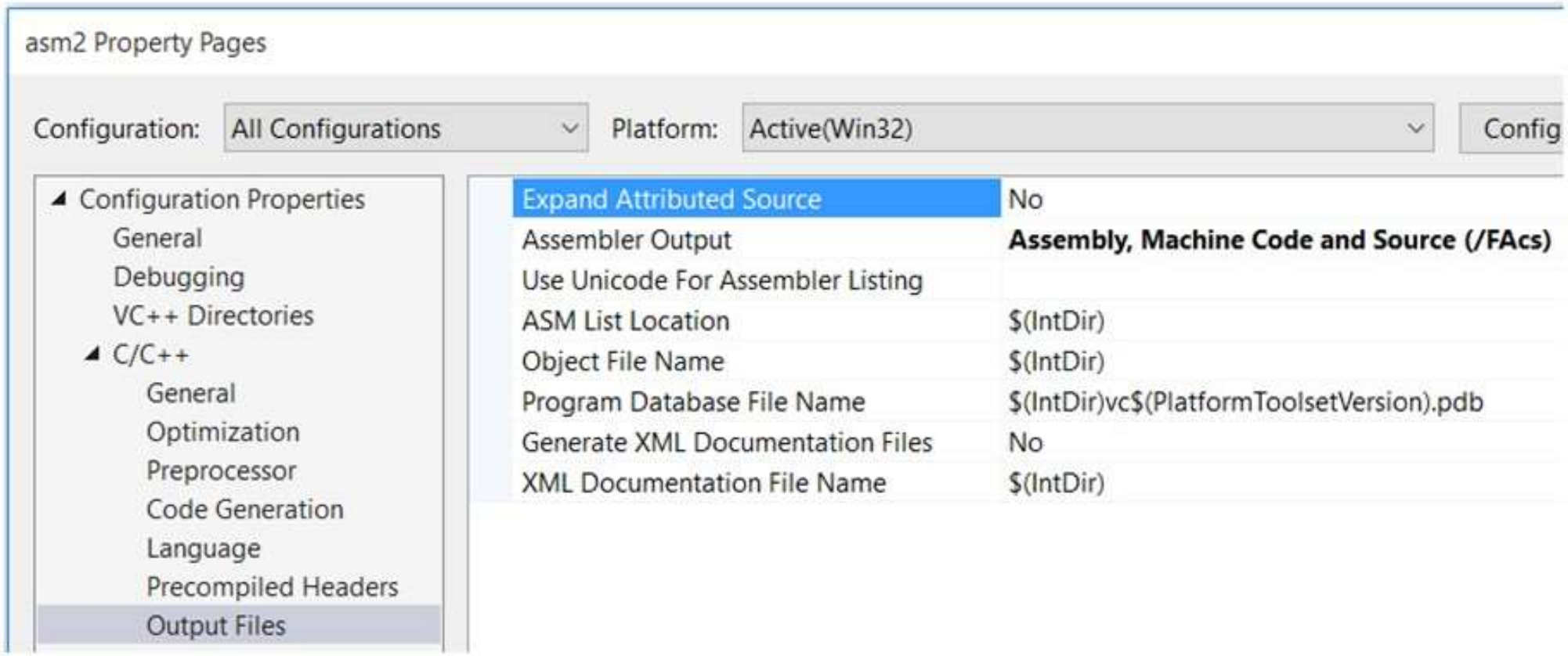
Генерируем ASM + Source



ASM + Source

```
; Listing generated by Microsoft (R) Optimizing Compiler

        TITLE   E:\Arkov\asm2\asm2\asm2.c
        call    _sscanf_s

; 10    :       A = A + B + C + D;

        mov     eax, DWORD PTR _D$[ebp]
        add     eax, DWORD PTR _C$[ebp]
        mov     ecx, DWORD PTR _A$[ebp]
        add     eax, DWORD PTR _B$[ebp]
        add     ecx, eax

; 11    :       printf("A = %i\n", A);

        push    ecx
```

Генерируем ASM + Code + Source



ASM + Code + Source

```
  00049 e8 00 00 00 00    call    _sscanf_s

; 10   :          A = A + B + C + D;

  0004e 8b 45 f8          mov     eax, DWORD PTR _D$[ebp]
  00051 03 45 0c          add     eax, DWORD PTR _C$[ebp]
  00054 8b 4d fc          mov     ecx, DWORD PTR _A$[ebp]
  00057 03 45 f4          add     eax, DWORD PTR _B$[ebp]
  0005a 03 c8             add     ecx, eax

; 11   :          printf("A = %i\n", A);

  0005c 51                push    ecx
```

Debug – Windows - Disassembly

```
013C18F5 E8 5B F7 FF FF       call        _sscanf_s (013C1055h)
013C18FA 83 C4 0C             add         esp,0Ch

        A = A + B + C + D;

013C18FD 8B 45 F8             mov         eax,dword ptr [A]
013C1900 03 45 EC             add         eax,dword ptr [B]
013C1903 03 45 E0             add         eax,dword ptr [C]
013C1906 03 45 D4             add         eax,dword ptr [D]
013C1909 89 45 F8             mov         dword ptr [A],eax

        printf("A = %i\n", A);

013C190C 8B 45 F8             mov         eax,dword ptr [A]
013C190F 50                   push        eax
```

dumpbin /disasm > asm2.txt

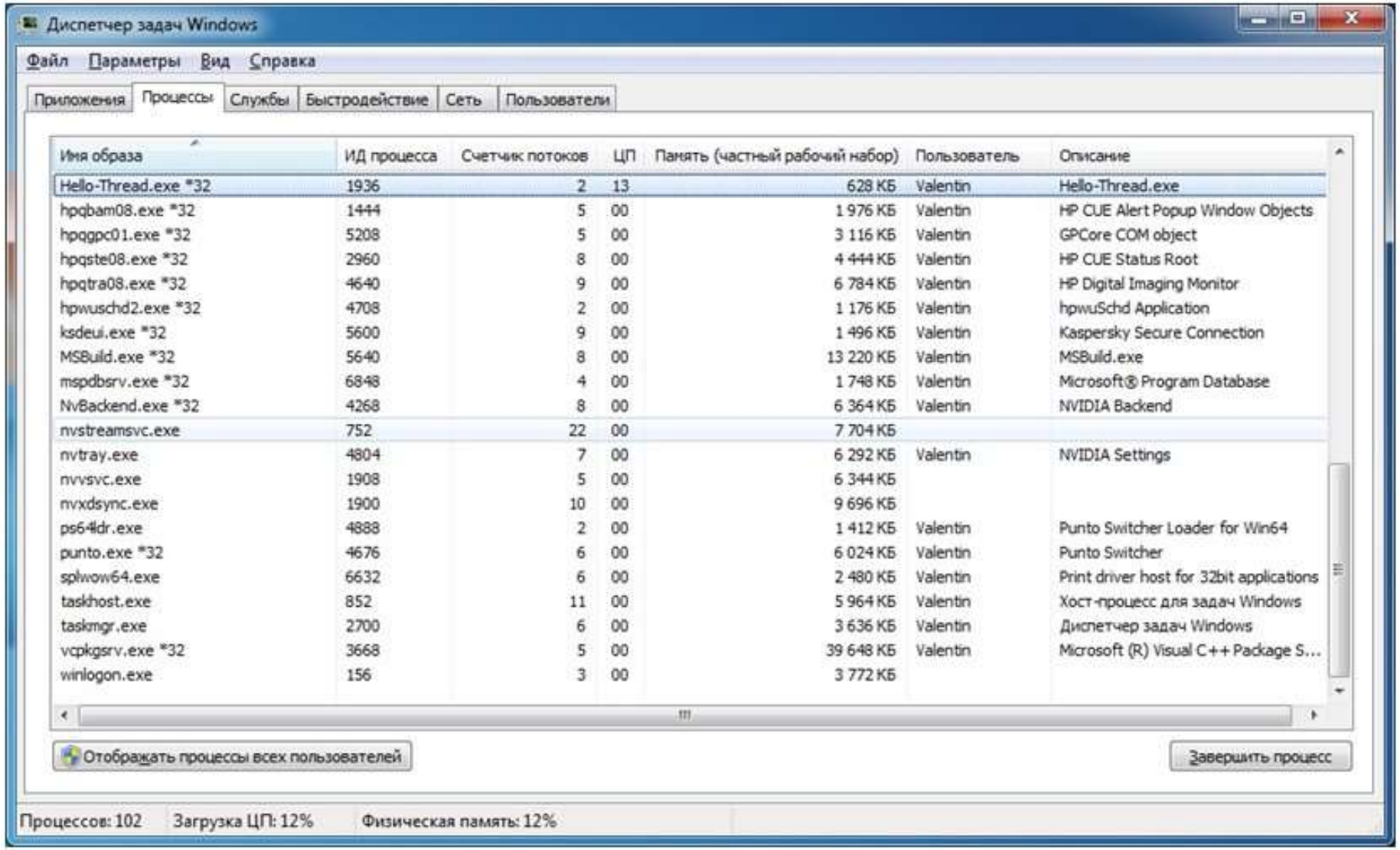
```
Microsoft (R) COFF/PE Dumper Version 14.00.23026.0
Copyright (C) Microsoft Corporation.  All rights reserved.

Dump of file asm2.exe

File Type: EXECUTABLE IMAGE
```

```
004010C9: E8 82 FF FF FF     call        _sscanf_s
004010CE: 8B 45 F8           mov         eax,dword ptr [ebp-8]
004010D1: 03 45 0C           add         eax,dword ptr [ebp+0Ch]
004010D4: 8B 4D FC           mov         ecx,dword ptr [ebp-4]
004010D7: 03 45 F4           add         eax,dword ptr [ebp-0Ch]
004010DA: 03 C8              add         ecx,eax
004010DC: 51                 push        ecx
```

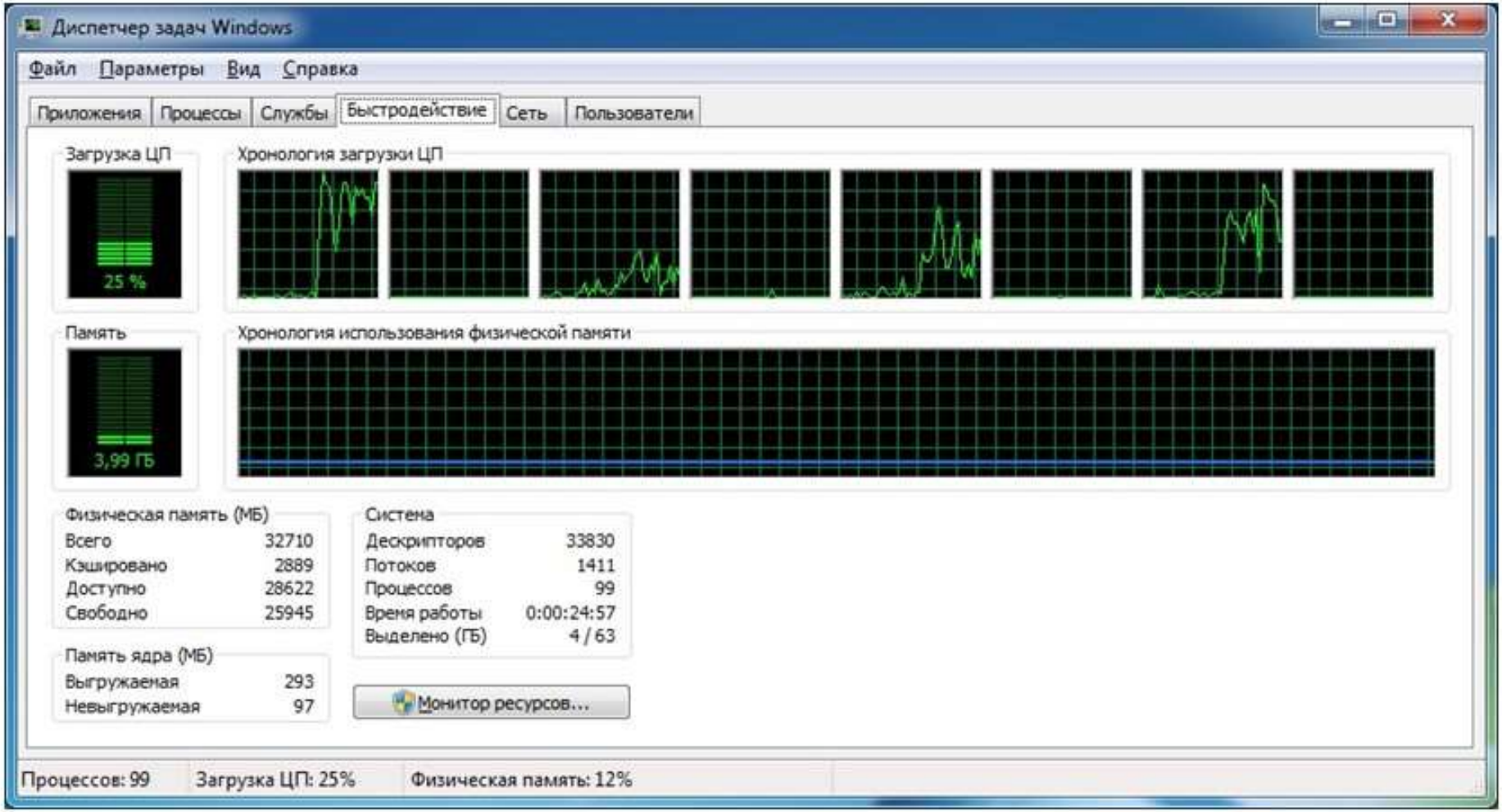
Thread + while(1)

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

void WINAPI ThreadProc()
{
    printf("Hello from Thread!\n");
    while(1);
}

void main()
{
    HANDLE Thread;
    Thread = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)ThreadProc, 0, 0, 0);
    printf("Hello from process!\n");
    while(1);
}
```

Thread + while(1)



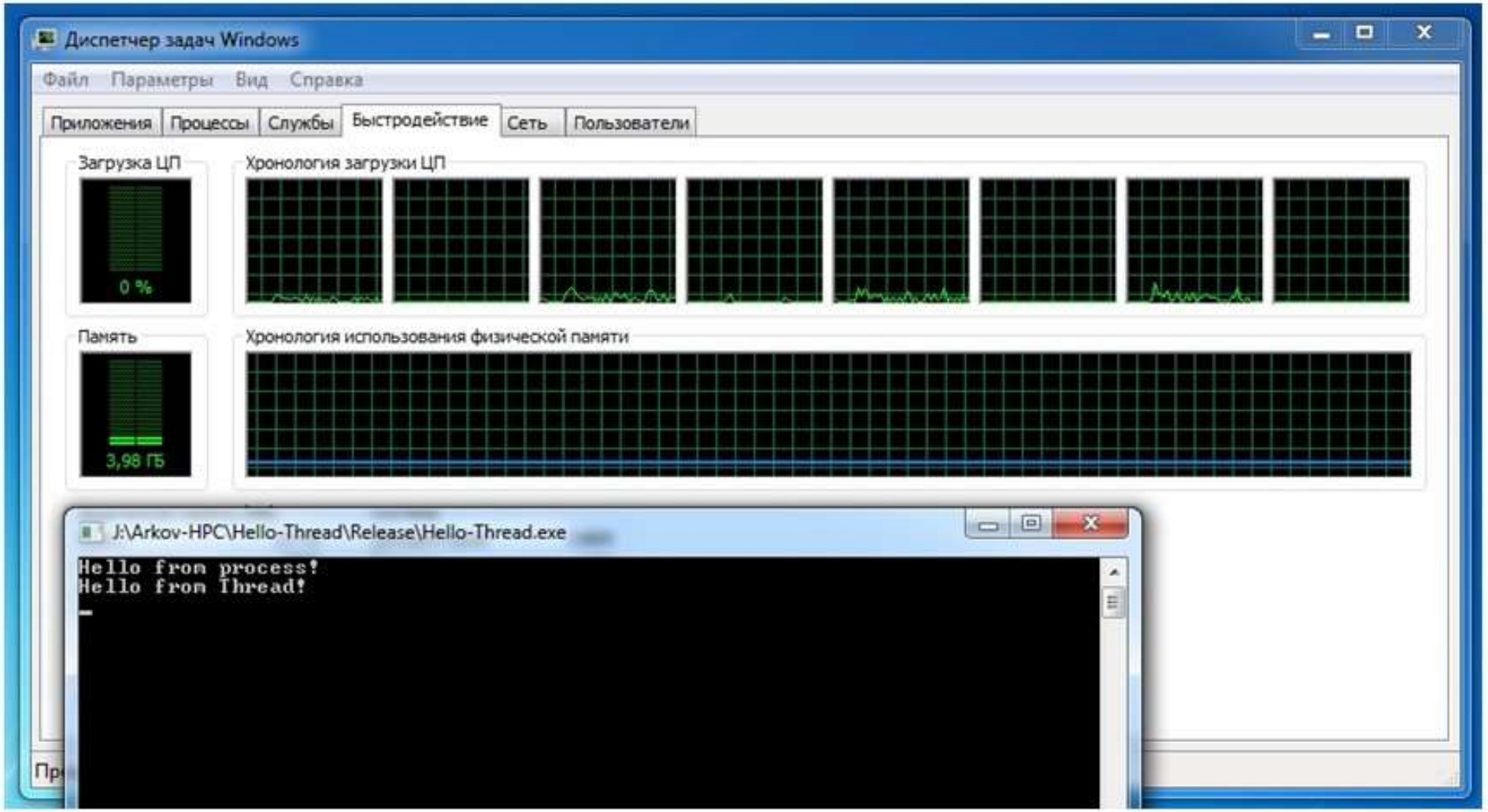
Thread + while(1)



Thread + getchar()

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

void WINAPI ThreadProc()
{
    printf("Hello from Thread!\n");
    getchar();
}

void main()
{
    HANDLE Thread;
    Thread = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)ThreadProc, 0, 0, 0);
    printf("Hello from process!\n");
    getchar();
}
```

Thread + getchar()



C + ASM

- Inline Assembly
- Ассемблерные вставки

```
#include <stdio.h>

void main(int argc, char *argv[])
{
    int A, B, C, D;
    sscanf_s(argv[1], "%i", &A);
    sscanf_s(argv[2], "%i", &B);
    sscanf_s(argv[3], "%i", &C);
    sscanf_s(argv[4], "%i", &D);
    __asm {
        MOV EAX, A;
        ADD EAX, B;
        ADD EAX, C;
        ADD EAX, D;
        MOV A, EAX;
    };
    printf("A = %i\n", A);
}
```

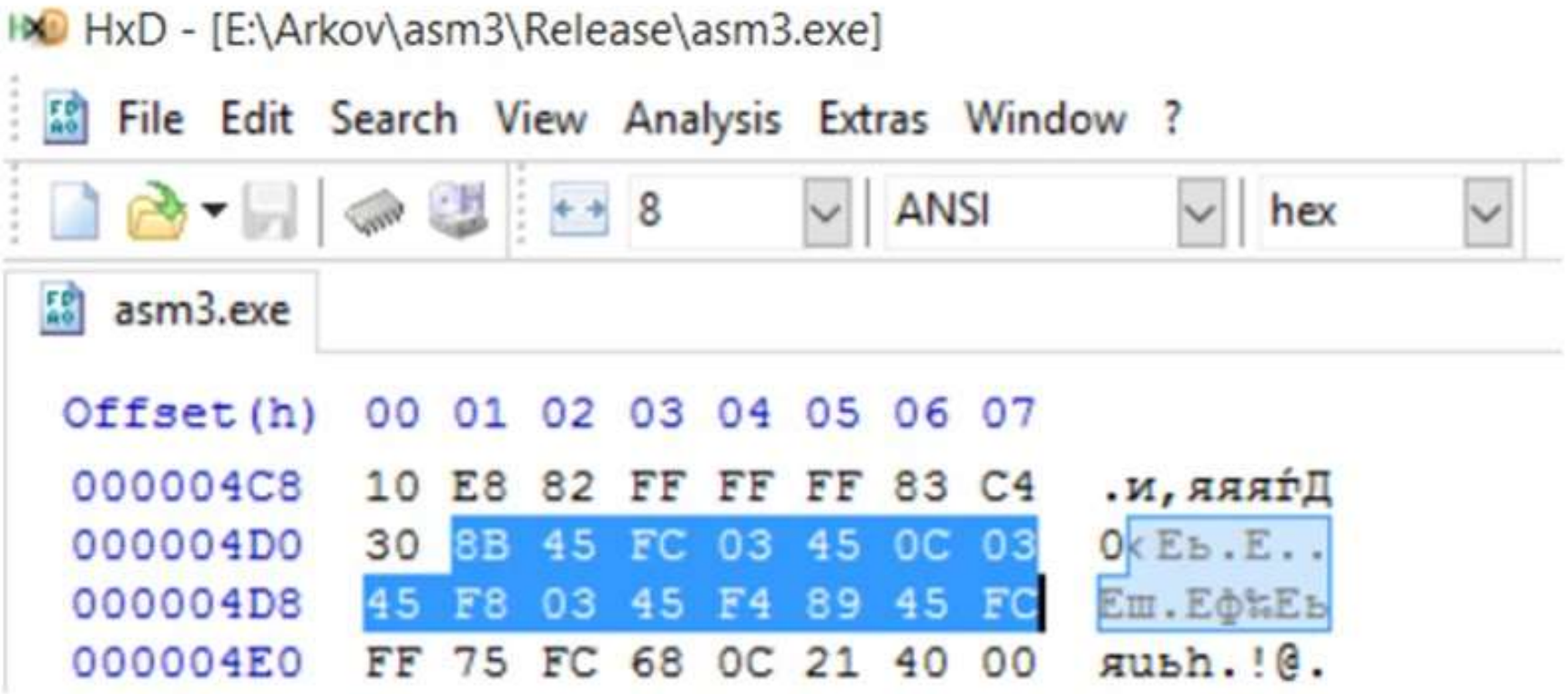
ASM + Code + Source

```
; 11    :          __asm {
; 12    :                  MOV EAX, A;
  00051 8b 45 fc            mov     eax, DWORD PTR _A$[ebp]

; 13    :                  ADD EAX, B;
  00054 03 45 0c            add     eax, DWORD PTR _B$[ebp]

; 14    :                  ADD EAX, C;
  00057 03 45 f8            add     eax, DWORD PTR _C$[ebp]

; 15    :                  ADD EAX, D;
  0005a 03 45 f4            add     eax, DWORD PTR _D$[ebp]

; 16    :                  MOV A, EAX;
  0005d 89 45 fc            mov     DWORD PTR _A$[ebp], eax
```

dumpbin/disasm > asm3.txt

```
Microsoft (R) COFF/PE Dumper Version 14.00.23026.0
Copyright (C) Microsoft Corporation.  All rights reserved.


Dump of file asm3.exe

File Type: EXECUTABLE IMAGE

  004010C9: E8 82 FF FF FF      call        _sscanf_s
  004010CE: 83 C4 30            add         esp,30h
  004010D1: 8B 45 FC            mov         eax,dword ptr [ebp-4]
  004010D4: 03 45 0C            add         eax,dword ptr [ebp+0Ch]
  004010D7: 03 45 F8            add         eax,dword ptr [ebp-8]
  004010DA: 03 45 F4            add         eax,dword ptr [ebp-0Ch]
  004010DD: 89 45 FC            mov         dword ptr [ebp-4],eax
  004010E0: FF 75 FC            push        dword ptr [ebp-4]
```

HxD Hex Editor: 8B 45 FC




```
8B 45 FC          mov         eax,dword ptr [ebp-4]
03 45 0C          add         eax,dword ptr [ebp+0Ch]
03 45 F8          add         eax,dword ptr [ebp-8]
03 45 F4          add         eax,dword ptr [ebp-0Ch]
89 45 FC          mov         dword ptr [ebp-4],eax
```

Адресация

- Сколько байт можно адресовать с помощью
  - 32-битного адреса?
  - 64-битного адреса?

Регистры

8 bit	16 bit	32 bit	64 bit
Byte	Word	Double Word	Quad Word
A	AX (AL, AH)	EAX	RAX
B	BX (BL, BH)	EBX	RBX
(R8B)	(R8W)	(R8D)	R8 … R15

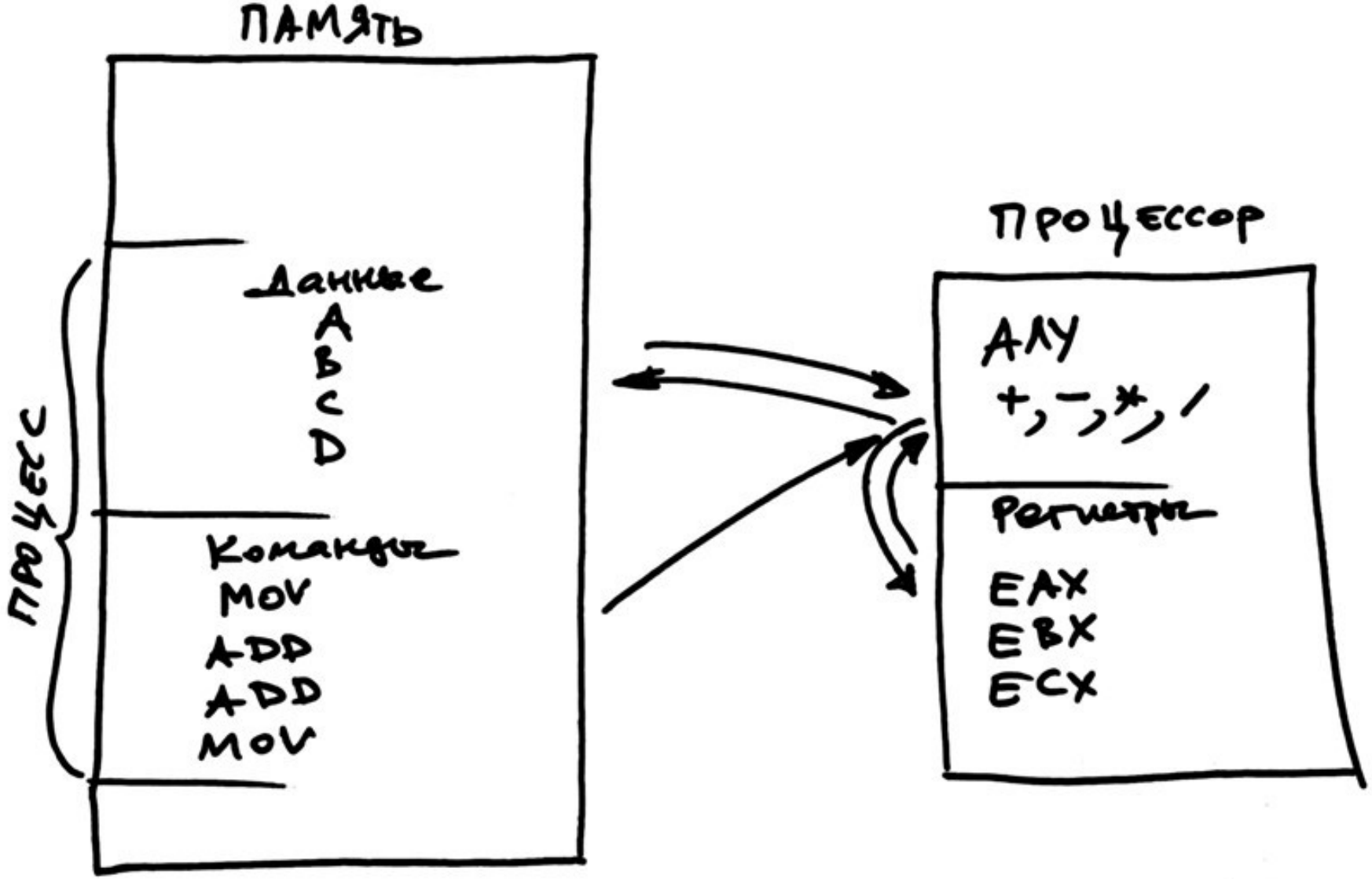
Названия единиц информации

byte	байт
word	(машинное) слово
double word	двойное слово
quad word	учетверённое слово слово учетверённой длины

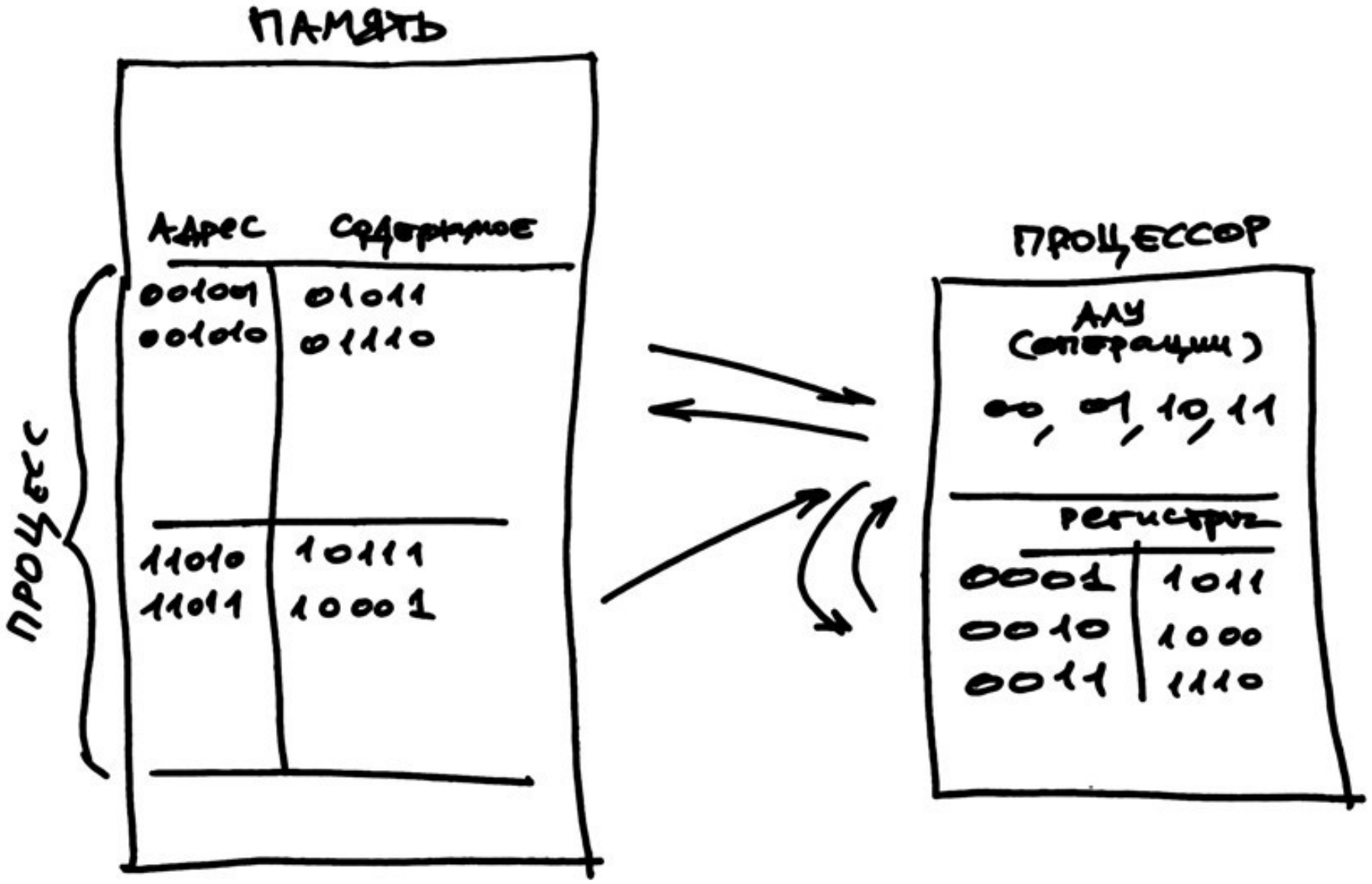
X64 ASM

```
  00058 e8 00 00 00 00    call    sscanf_s

; 10   :          A = A + B + C + D;

  0005d 8b 54 24 48       mov     edx, DWORD PTR A$[rsp]

; 11   :          printf("A = %i\n", A);

  00061 48 8d 0d 00 00
        00 00             lea     rcx, OFFSET FLAT:??_C@_07ODHHKHC
  00068 44 8b 44 24 58    mov     r8d, DWORD PTR D$[rsp]
  0006d 44 03 44 24 50    add     r8d, DWORD PTR C$[rsp]
  00072 44 03 44 24 20    add     r8d, DWORD PTR B$[rsp]
  00077 41 03 d0          add     edx, r8d
  0007a 89 54 24 48       mov     DWORD PTR A$[rsp], edx
  0007e e8 00 00 00 00    call    printf
```

Процесс (ассемблер)



Процесс (машинные коды)



Задание

- Нейман
- Тьюринг
- Принстонская и гарвардская архитектура

- Современные процессоры
  - intel
  - amd
  - ibm power
  - эльбрус

Виды ЭВМ

Вид	Цена	Назначение
Одноразовые	1	Открытки, RFID
Встроенные	10	Приборы
Игровые	100	Игры
Персональные	1000	Настольные, портативные, мобильные
Серверы	10000	Локальные сети и интернет
Большие машины	100000	Большие серверы (банки, ЦОДы)
Суперкомпьютеры	1000000	Сложные расчеты (погода, наука)

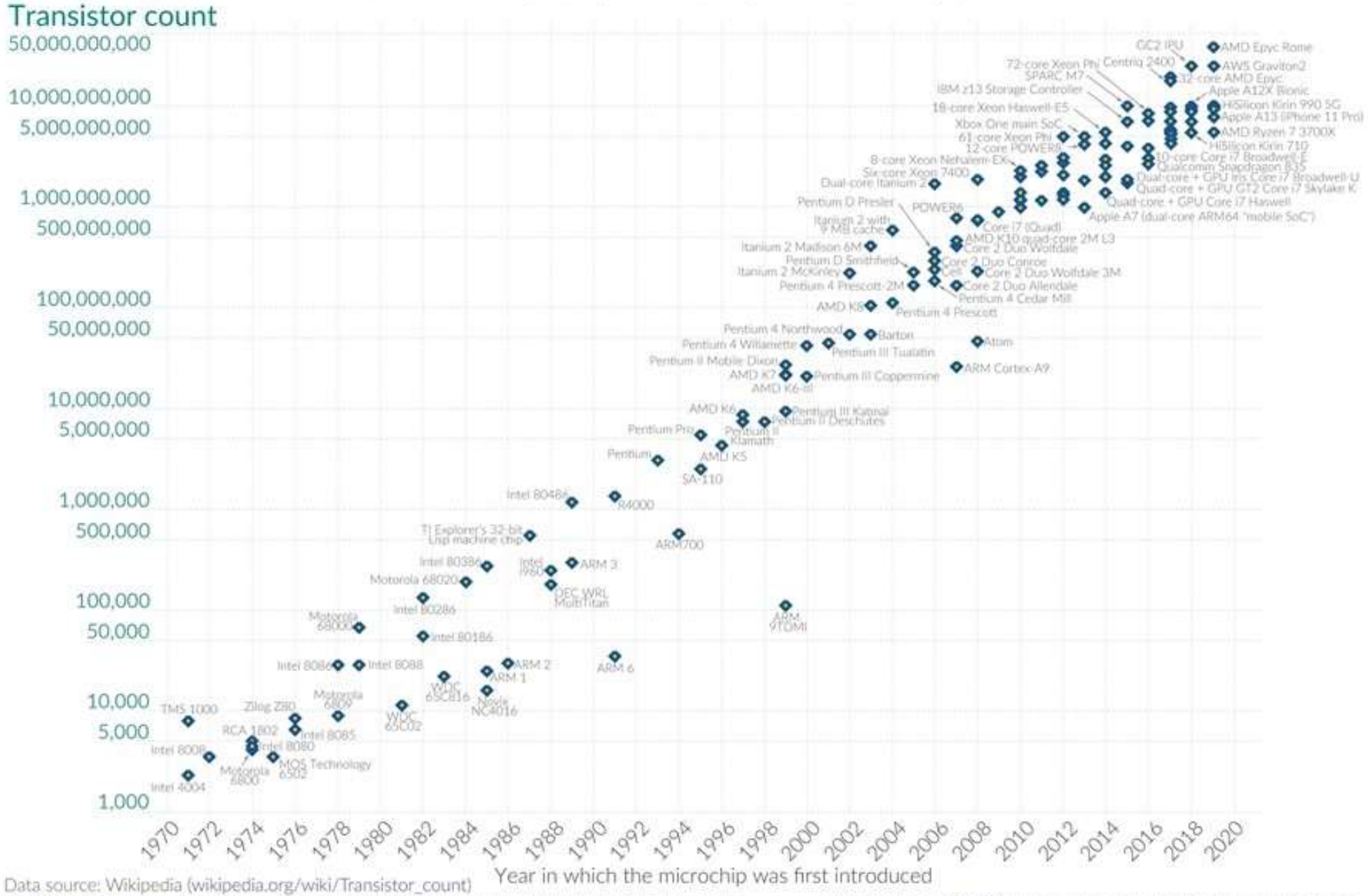
Закон Мура (Moore's law)

- «Закон» технологического прогресса
  - Число транзисторов на микросхеме удваивается каждые 1,5 – 2 года
- Экспоненциальный закон
  - Прямая линия в логарифмическом масштабе
- Гордон Мур (Gordon Moore)
  - Основатель Intel и Fairchild Semiconductor

Moore's Law: The number of transistors on microchips doubles every two years

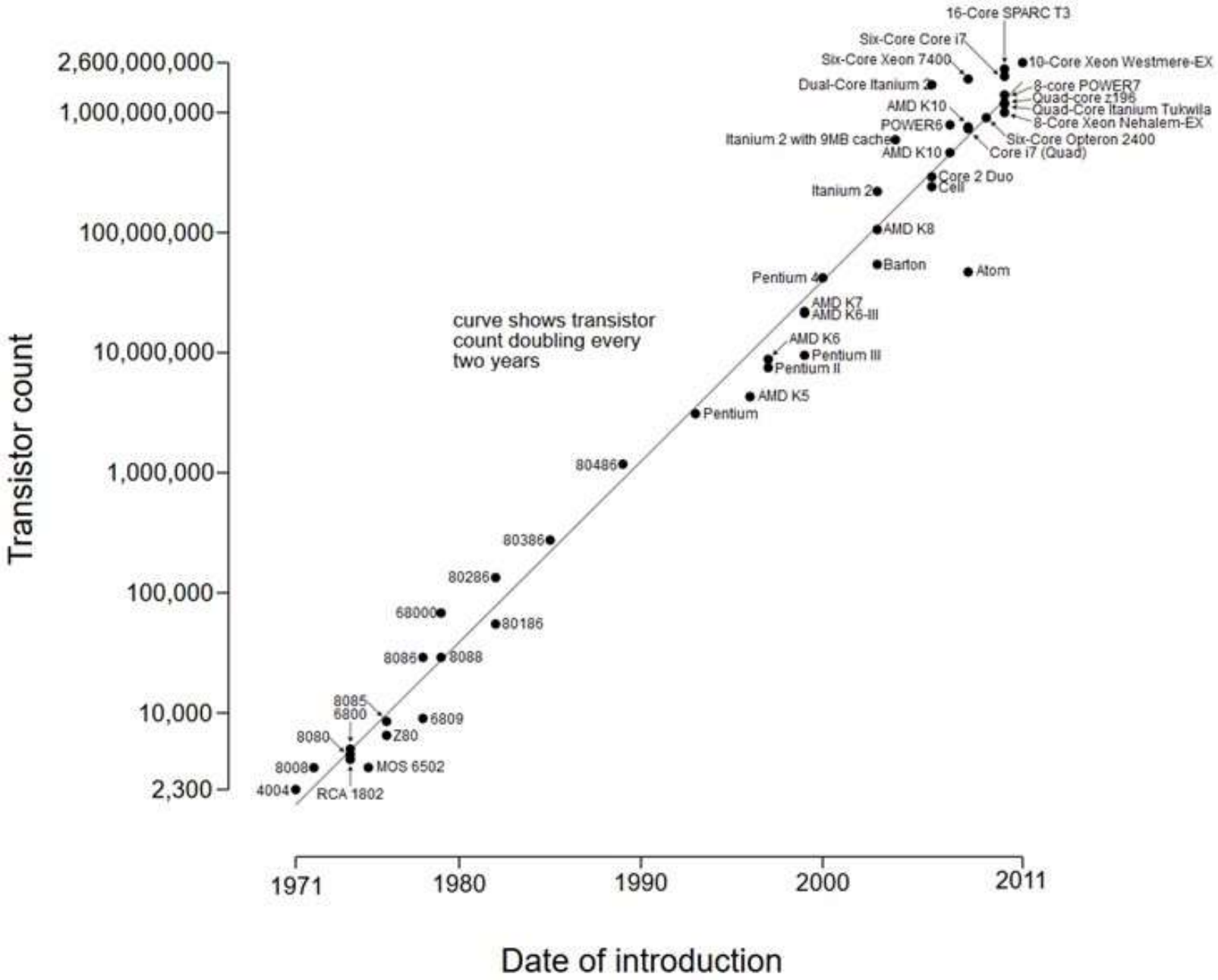
Our World in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.



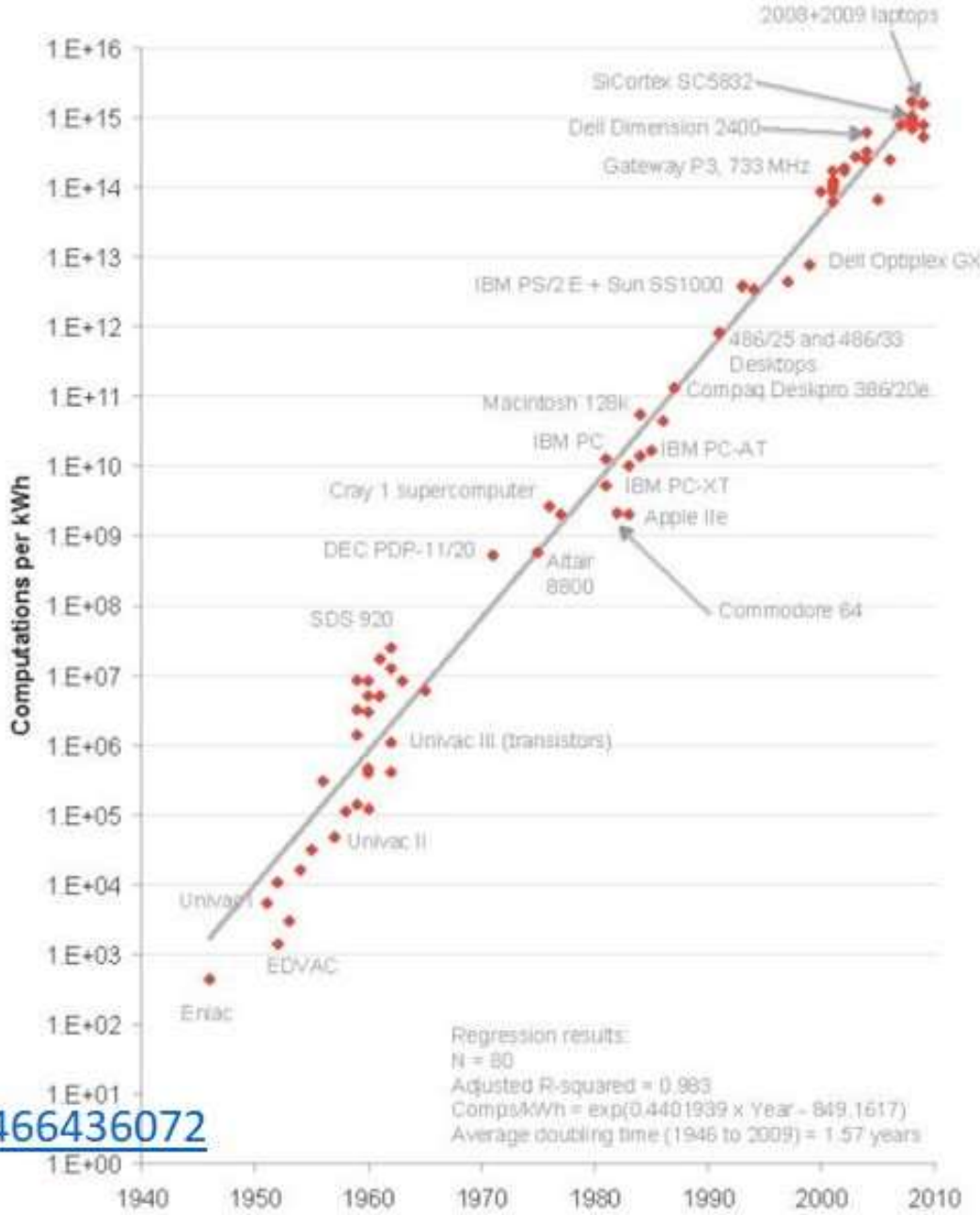
Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)

OurWorldinData.org – Research and data to make progress against the world's largest problems.



https://commons.wikimedia.org/wiki/File:Transistor_Count_and_Moore's_Law_-_2011.svg

Вычислений на один киловатт-час



Dr Jon Koomey

http://www.koomey.com/post/14466436072

Задание

- Закон масштабирования Деннарда
- Закон Гроша
- Zimmermann's Law

Закон Вирта (Wirth's law)

- Программы становятся медленнее более стремительно, чем компьютеры становятся быстрее.
- Программы перерастают компьютеры в размерах и медлительности
- Повышение производительности аппаратной части ещё не означает ускорения работы.
  - Никлаус Вирт (Niklaus Wirth)
    - Автор языка программирования Паскаль

Закон Гейтса (Gates' law)

- Программы становятся в два раза медленнее каждые полтора года, что компенсирует закон Мура
  - добавление функций
  - некачественный код
  - плохая организация работы

  - Билл Гейтс (Bill Gates)
    - основатель Microsoft

1-й закон ПО

- Программное обеспечение — это газ
  - Он распространяется и полностью заполняет резервуар, в котором находится
- Software is a gas; it expands to fill its container.

  - Натан Мирвольд (Nathan Myhrvold)
    - Chief Technology Officer, Microsoft

    - http://www.youtube.com/watch?v=gE4N0G9kElM
    - The future of software, the software industry, and Windows '47

Задание

- Раздувание программного обеспечения
- Software bloat
- Bloatware