Архитектура и организация компьютера

Алексей Владыкин

СПбГУ ИТМО

8 сентября 2010

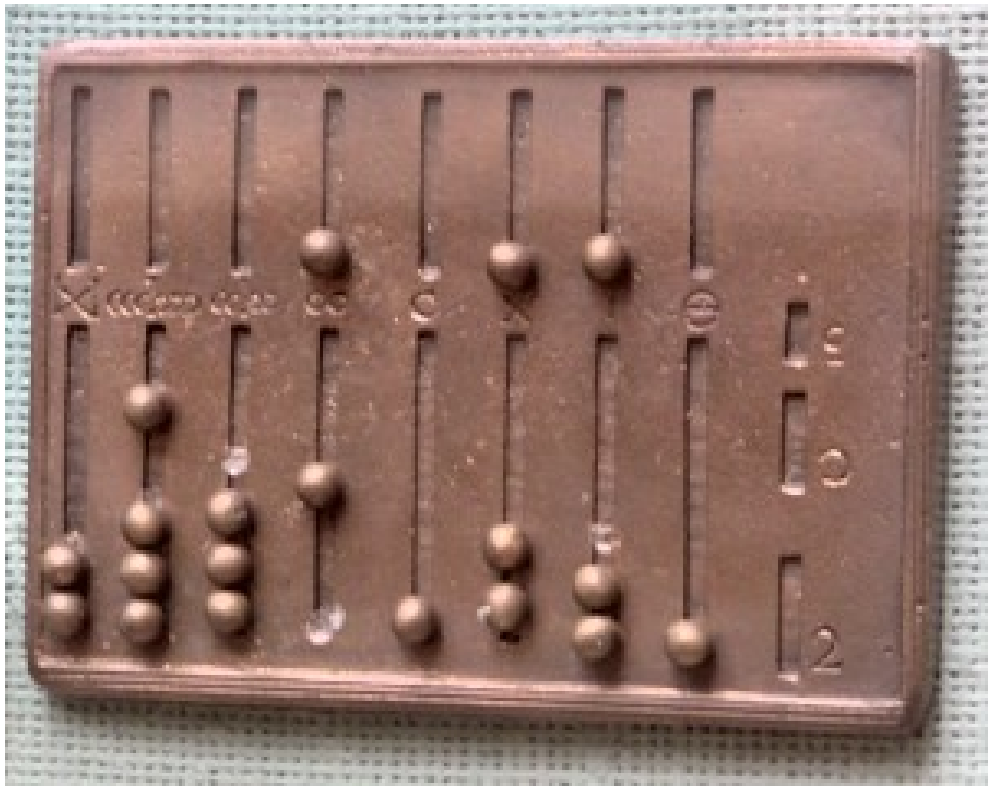
Зачем это знать?

- Фундамент для изучения программирования
- Глубокое понимание происходящего внутри программ
- Оптимизация производительности

- Пример: обход матрицы по строкам/столбцам

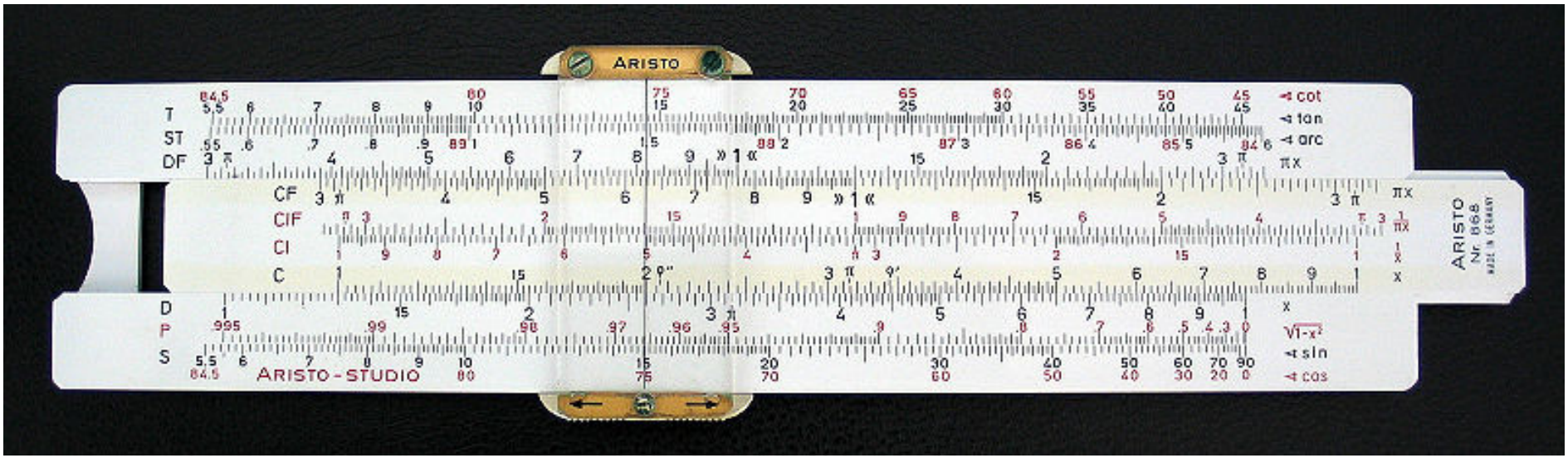
Немного истории

3 тыс. до н. э. Абак. Древний Вавилон.



Немного истории

XVII в. Логарифмы, логарифмическая линейка.
Механические счетные машины Паскаля и Лейбница.

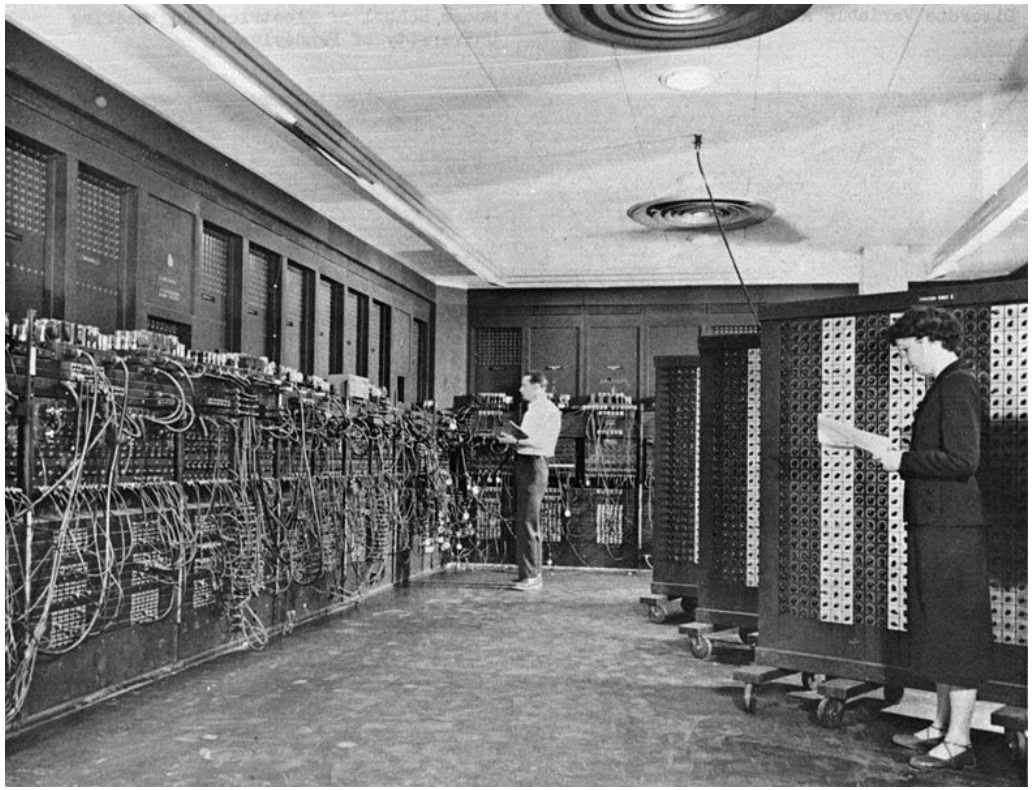


Немного истории

Начало XIX в. Механические машины на перфокартах.

Немного истории

1940-е гг. Большие специализированные машины (ABC, Mark I, . . .).
ENIAC — полностью электронная машина.



Немного истории

1950-е гг. Мэйнфреймы. Изобретение и применение транзисторов.

Немного истории

1980-е гг. Персональные компьютеры.

Немного истории

1990-е гг. Ноутбуки, смартфоны, КПК.

Архитектура фон Неймана

- Джон фон Нейман — венгро-американский математик, внесший большой вклад в информатику.

- Программа и данные находятся в общей памяти, программа и данные неотличимы друг от друга
- Команды зачитываются из памяти и исполняются последовательно, за исключением условных и безусловных переходов
- Используется двоичная система счисления

- Первые компьютеры с этой архитектурой — 1940-е гг.
- Альтернатива — гарвардская архитектура.

Двоичная система счисления

- Бит (bit, от «binary digit») — единица информации: 0 или 1
- Байт (byte) — 8 бит, минимальная адресуемая ячейка памяти
- Разрядность процессора определяет максимальную ширину операндов (чисел и адресов в памяти)
  - 32-битный процессор: 1, 2, 4 байта
  - 64-битный процессор: 1, 2, 4, 8 байт

Центральный процессор (CPU)

- Целочисленная арифметика.
- Арифметика над числами с плавающей точкой.
- Копирование данных из ОЗУ в регистр и обратно (в архитектуре x86 — через северный мост).
- Обмен данными с внешними устройствами (в архитектуре x86 — через южный мост).

Центральный процессор (CPU)

- Регистры
- Кэш-память
- Декодер команд
- Конвейер команд
- Предсказатель переходов

Assembler

- Низкоуровневый машинный код в мнемоничном текстовом виде
- Под каждую архитектуру свой ассемблер
- Пример: Hello World на NASM для Linux/x86

```
SECTION .data
msg: db "Hello, world",10
len: equ $-msg

SECTION .text
global _start
_start: mov edx, len
        mov ecx, msg
        mov ebx, 1      ; stdout
        mov eax, 4      ; write
        int 0x80

        mov ebx, 0
        mov eax, 1      ; exit
        int 0x80
```

Загрузка компьютера

- BIOS, поиск загрузочных устройств
- Загрузчик с выбранного загрузочного устройства
- Операционная система, драйверы устройств
- Прикладные программы

Исполнение программы

```
pc = startAddress;
do {
    instruction = memory.instructionAt(pc);
    execute(instruction, &pc);
};
```

- PC — program counter
- Инструкции имеют разную длину (x86)
- PC увеличивается на длину инструкции или явно устанавливается инструкцией перехода

Многозадачность

- Планировщик задач (операционная система)
- На одном ЦП:
  - задачи исполняются поочередно в соответствии с их приоритетами;
  - состояние процессора (контекст), соответствующее каждой задаче, сохраняется в оперативной памяти;
  - при смене задачи происходит переключение контекста.
- На разных ЦП:
  - возможно честное параллельное исполнение.

Рекомендуемая литература


Зубков С.
Ассемблер для DOS, Windows и UNIX.
М.: ДМК Пресс; СПб.: Питер, 2006. — 608 с.


Одинцов И.
Профессиональное программирование. Системный подход. — 2-е изд.
СПб: БХВ-Петербург, 2004. — 624 с.: ил. // Главы 6 и 7.


Таненбаум Э.
Архитектура компьютера. — 5-е изд.
СПб.: Питер, 2007. — 848 с.


Таненбаум Э.
Современные операционные системы. — 2-е изд.
СПб.: Питер, 2007. — 1040 с.


Drepper U.
What every programmer should know about memory.
http://lwn.net/Articles/250967/