Basics of Computer Systems

Talk by Peter Manshausen

Introduction

Contents

- Models for understanding computers: The Calculator Model, the File Clerk Model
- Hardware: Basic Building Blocks of Computers: Input/Output, Processor, Memory, Buses, Integrated Circuits, Example: Apple iPad 2, Computer Speed and Clock
- Software: Fundamental Computing Concepts, Stored Program Computer, Example: Code Stream, Below the Program, Program Hierarchy, Booting

Forms of Computing

- PCs
- Servers
- Supercomputers
- Embedded systems
- Personal Mobile Devices (PMDs): Smartphones, Tablets, Smartwatches
- Warehouse Scale Computers providing flexible (cloud-) services

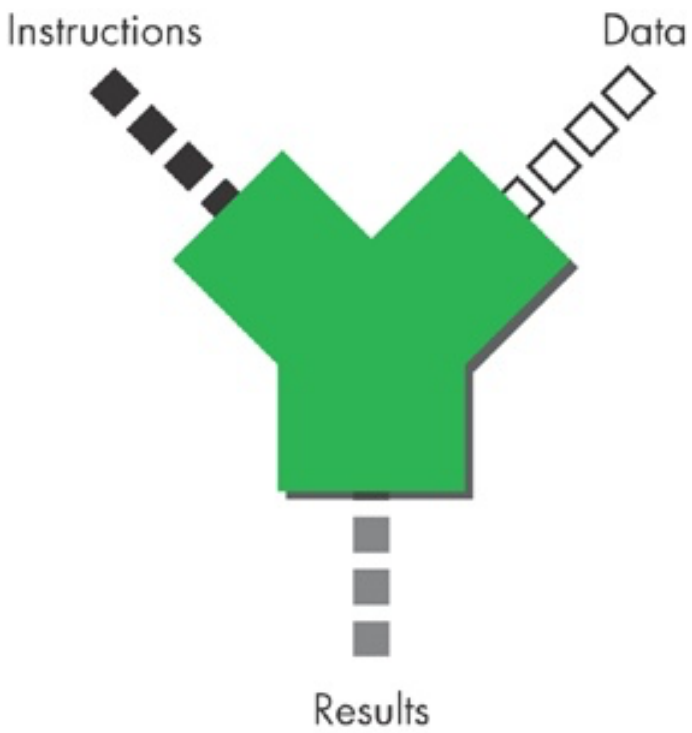
Motivation

- understanding computers on an abstract level to apply that model to every computer system we encounter

- to do this:
  - choose a model
  - categorize building blocks
  - basic understanding of programs

The Calculator Model

- input of data (numbers) and instructions (arithmetic operators like addition and multiplication)
- data is processed according to instructions
- results are displayed



The File Clerk Model

- read-modify-write
- „dumber but faster“

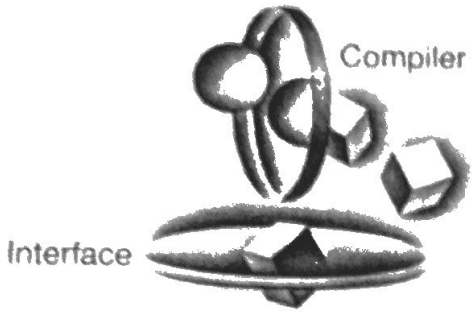
- binary: ones and zeroes as machine language
- transistors as tiny switches

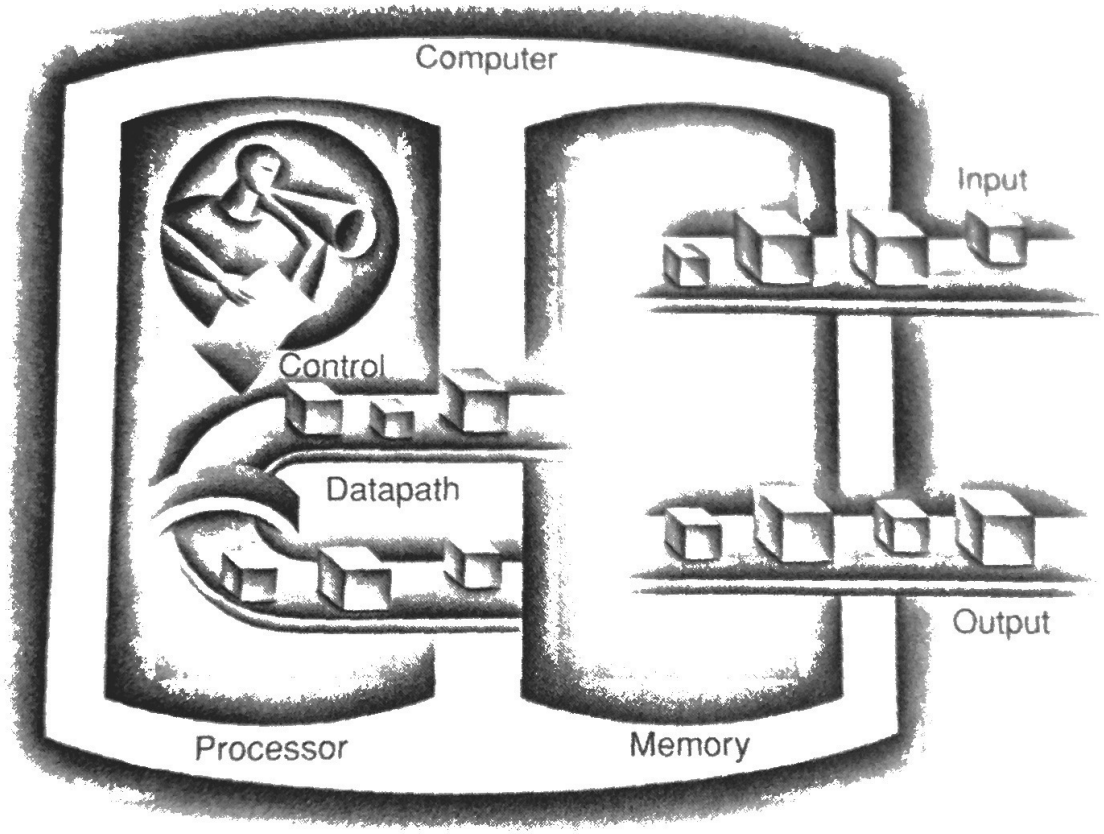
A Computer's basic building blocks

- Input/Output (I/O)
- Datapath
- Control Unit
- Memory (the database)

- Transmission: buses

Von-Neumann-Scheme





Input/Output devices

- mouse and keyboard (input)
- buttons (input)
- microphones (input)
- camera (input)
- gyroscope and accelerometer (input)

- Displays (mainly output, touchscreens also input)
- Speakers (output)

Processor 1: CU

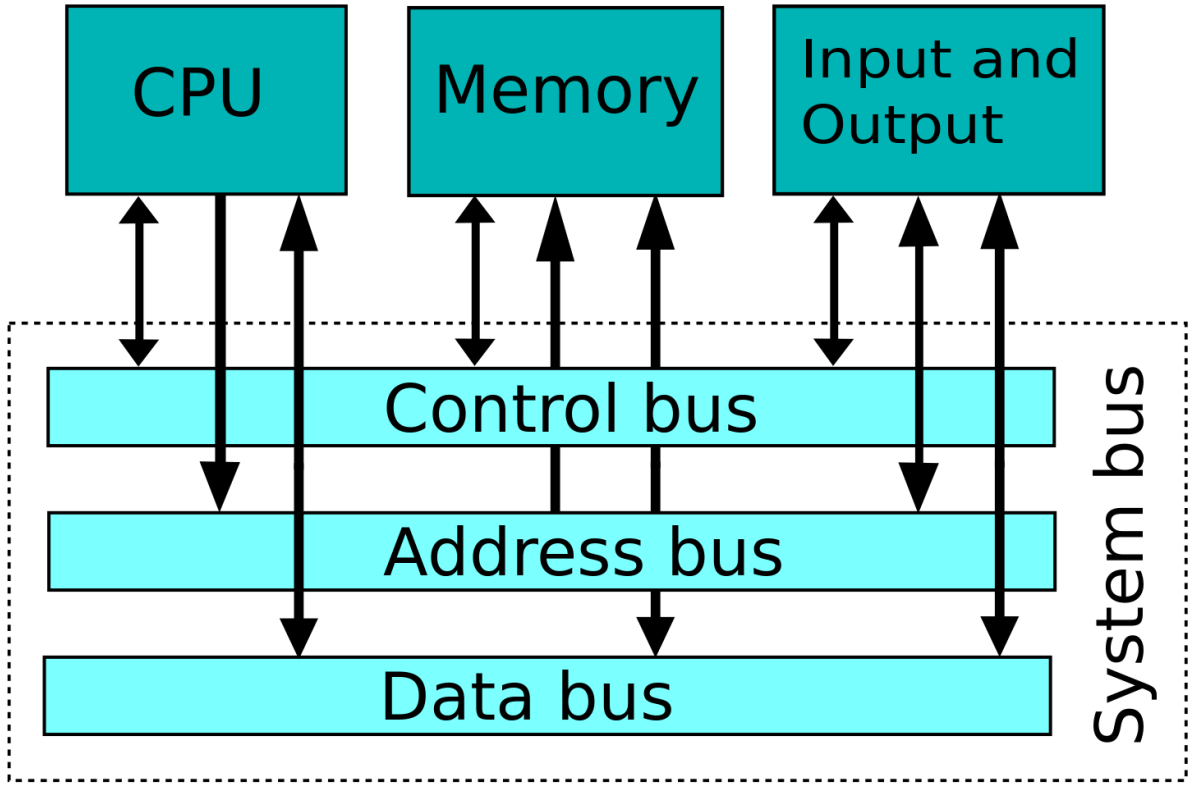
- Control Unit (CU)
- orchestrates execution of instructions
- performs operations dealing with main memory such as loading data (`load`) or storing data (`store`)

Processor 2: Datapath

- Inside Central Processing Unit (CPU):
- ALU (arithmetic logic unit) which performs arithmetic (addition: `add`) and logical operations (negation: `not`, conjunction: `and`)
- registers:
  - small but very fast memory inside CPU
  - physically close to ALU
  - thus speed of computer is decoupled from speed of memory

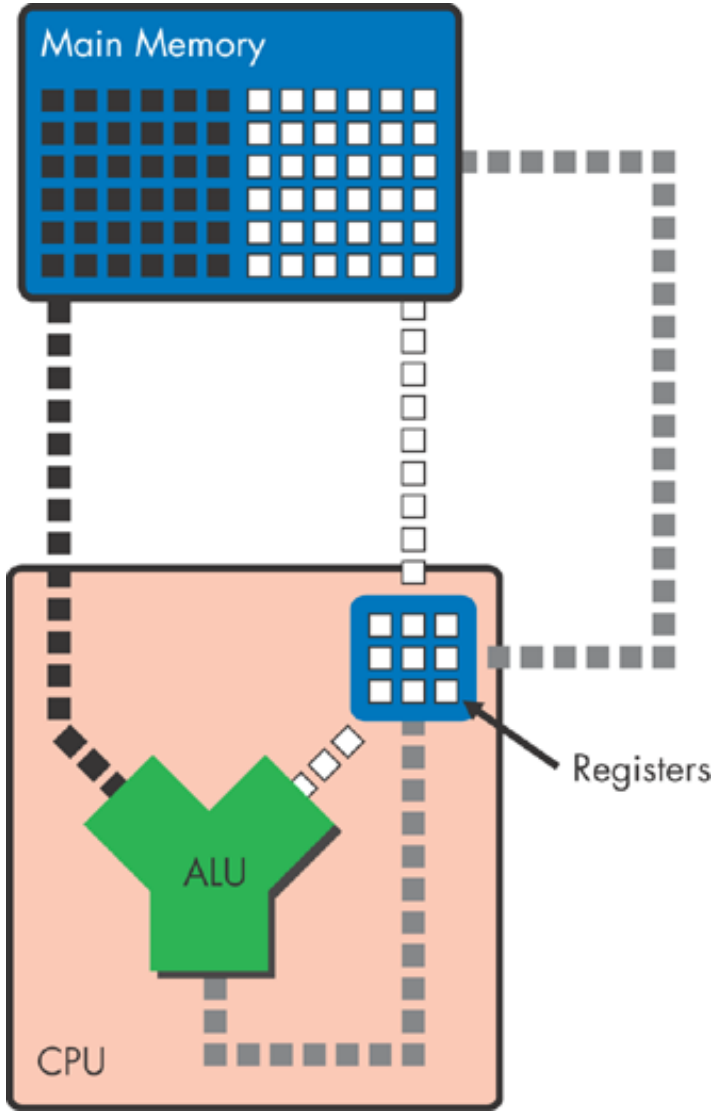
Buses

- system bus connects processor to the other computer components
- sometimes counted to datapath
- transmits information like:
  - data that is operated on
  - storage addresses in memory
  - control instructions for bus itself



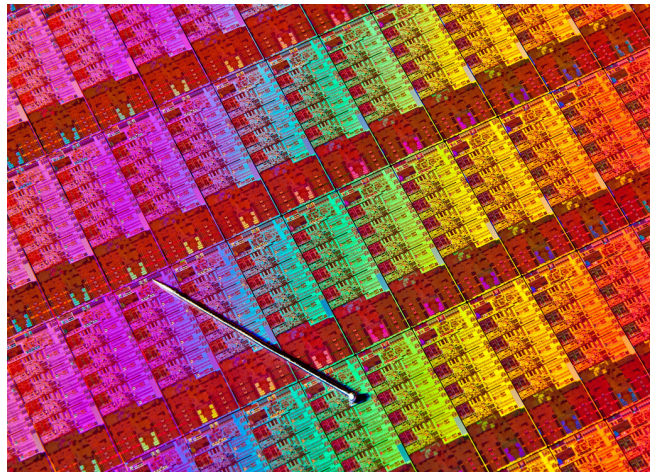
(Main) Memory

- registers store only small amount of data
- primary storage is needed for programs and data that is not currently operated on
- main memory physically farther away from CPU and slower
- almost always a kind of RAM (random access memory as opposed to sequential access)
- data is lost when power is switched off (volatile memory)
- secondary storage is non-volatile



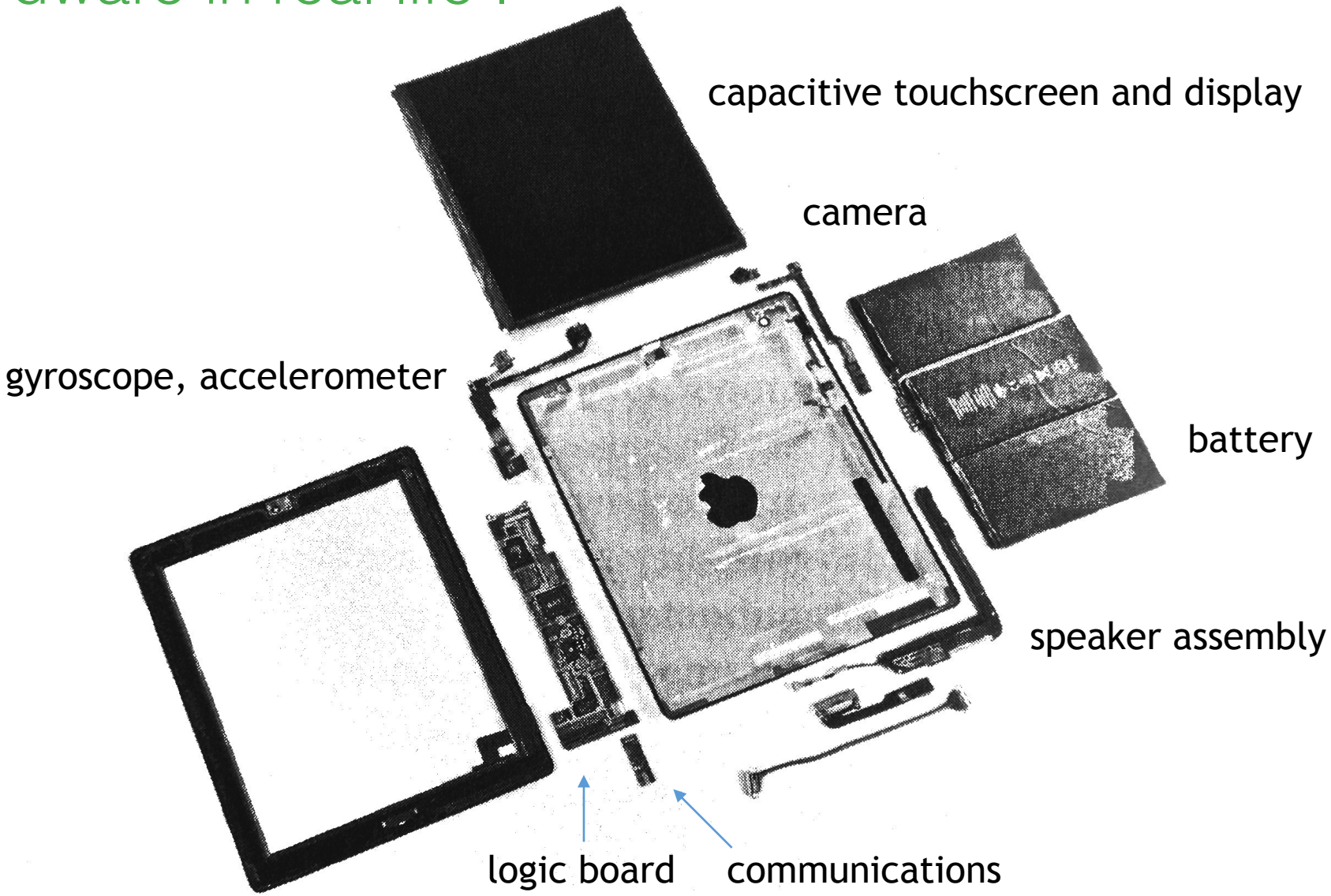
Chips: Integrated Circuits

- Almost all inner parts of computers are made up of transistors.
- ICs pack large amount of transistors on a single wafer.
- Moores Law (1965): number of transistors per chip doubles every two years, transistor density proportional to computing resources
- semiconductor basis (silicon, Si)
- Si can be made conductor, insulator or switch



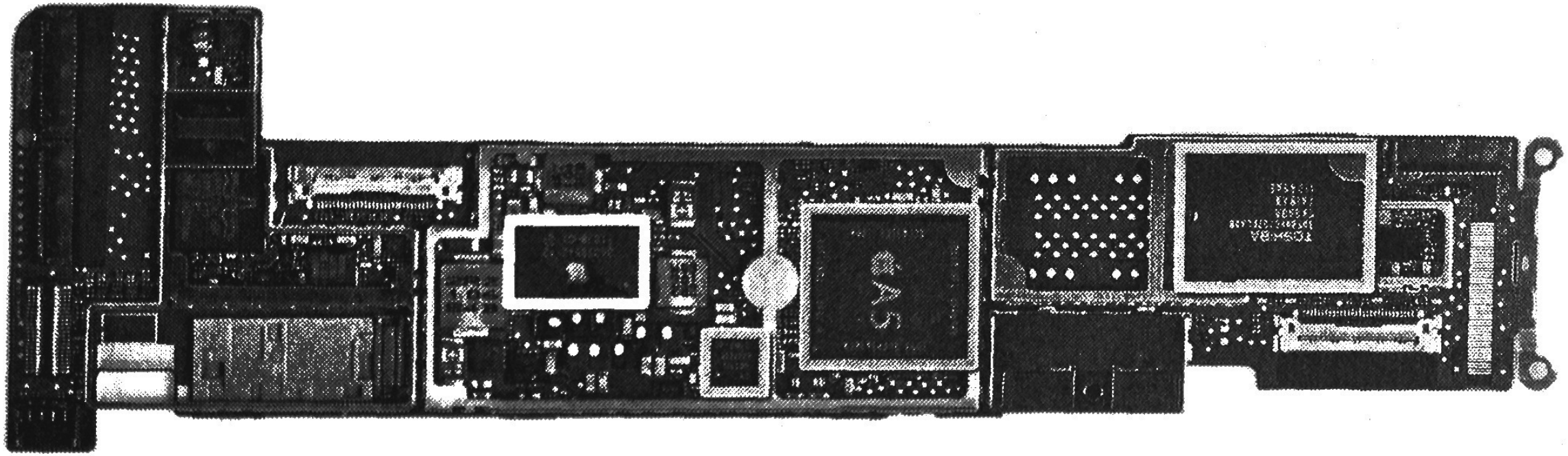
Example: Hardware in real life I

Apple iPad 2



Example: Hardware in real life II

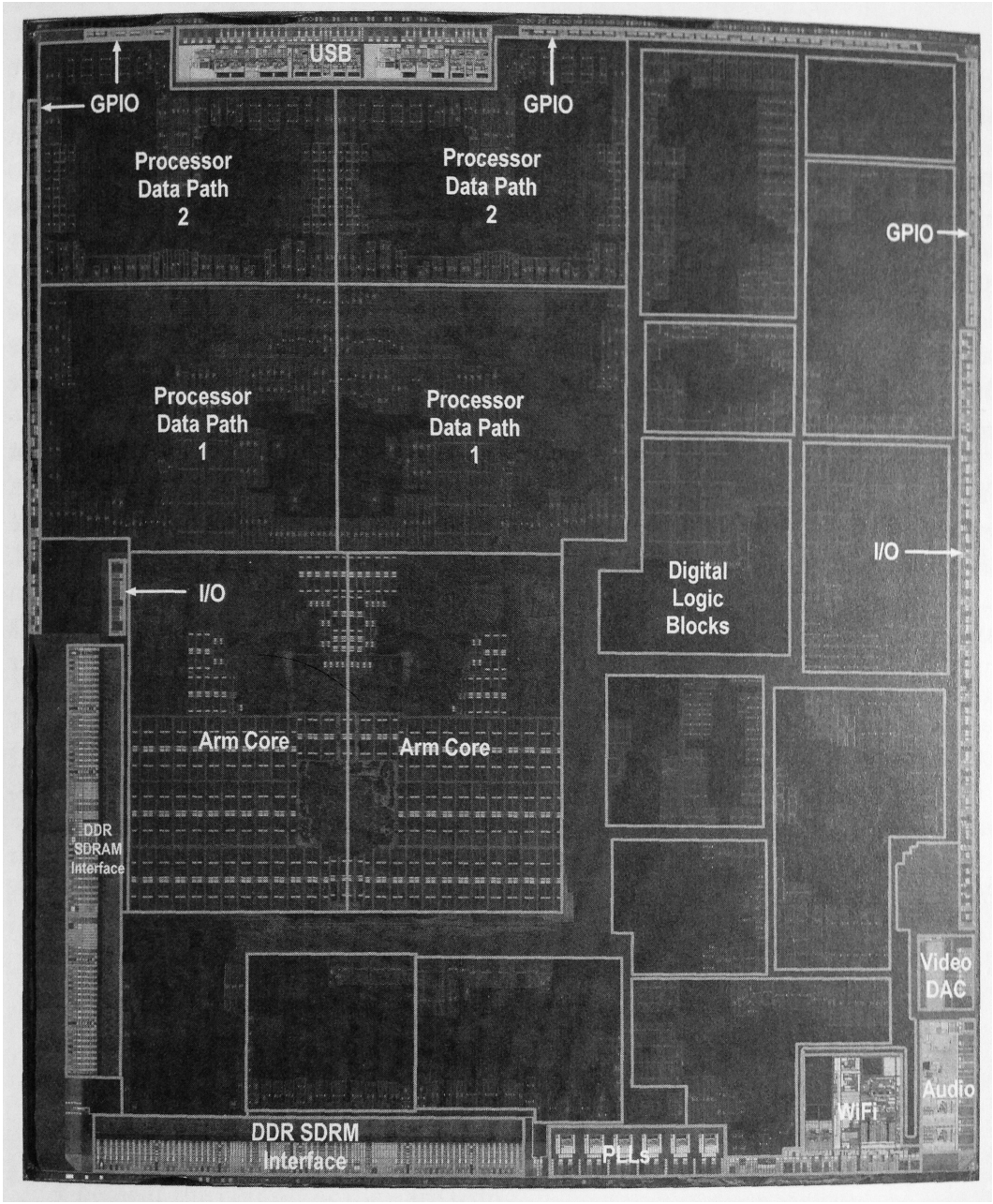
logic board



secondary storage processor and main memory

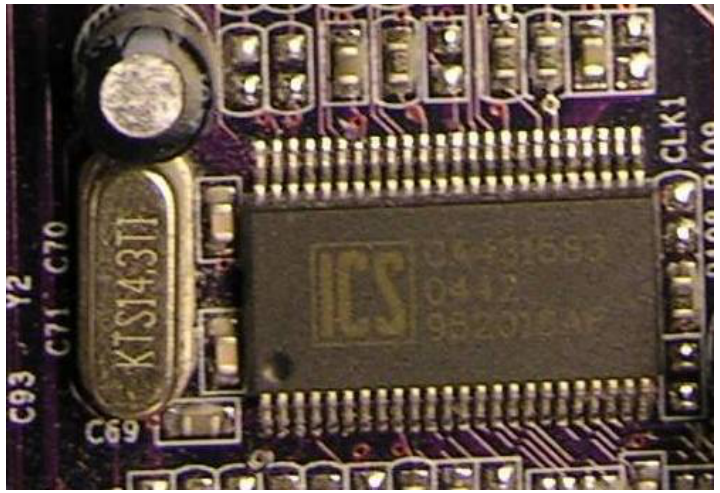
Example: Hardware in real life III

processor integrated circuit in the A5



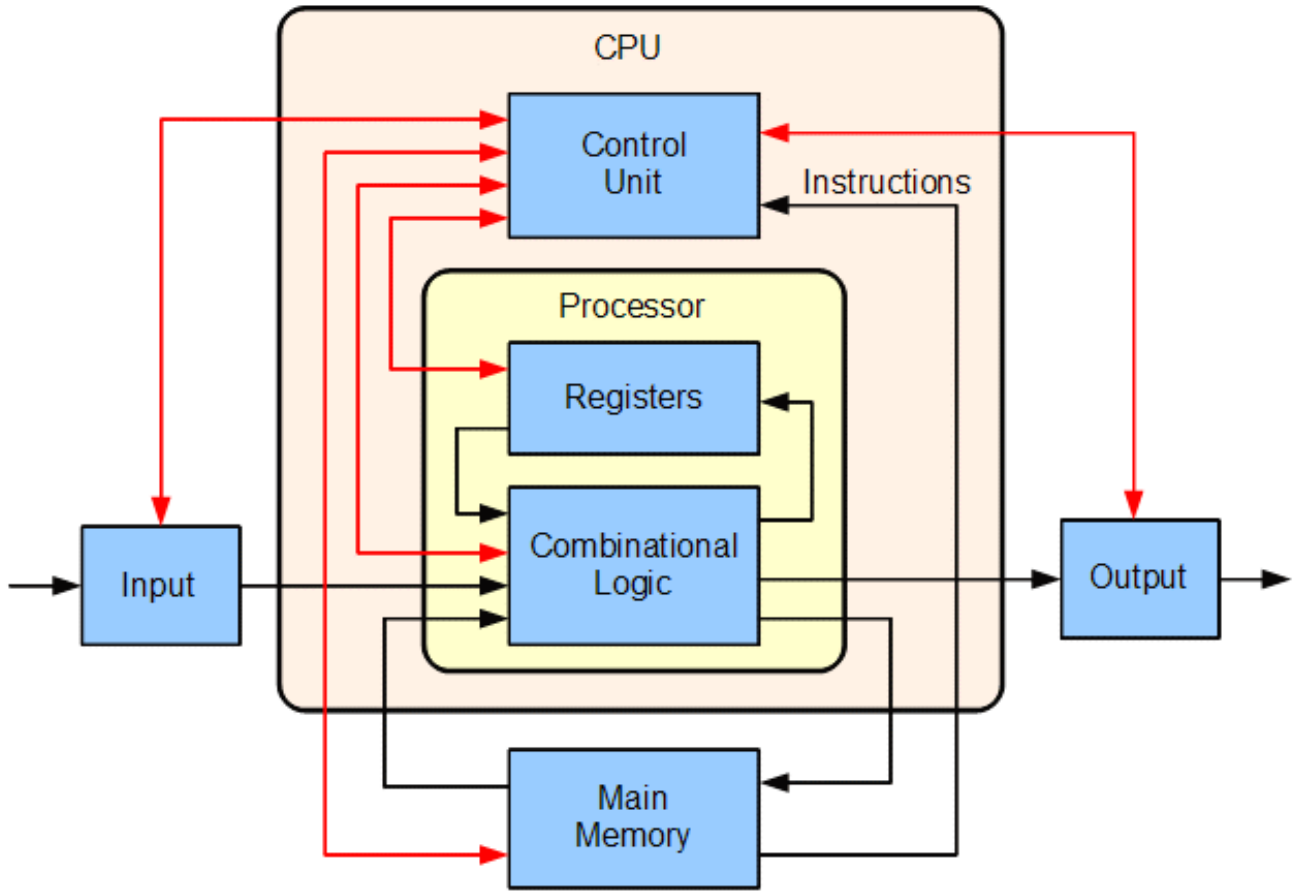
Computer Speed

- clock cycle: processor works at predefined speed, one (simple) operation per unit time
- clock rate given by an external „clock generator“ using piezoelectric resonance circuit (quartz)
- speed of processors has grown more than speed of other components
- components ordered by their respective speed:
  - CPU
  - main memory
  - I/O devices such as sound card



Stored Program Computer

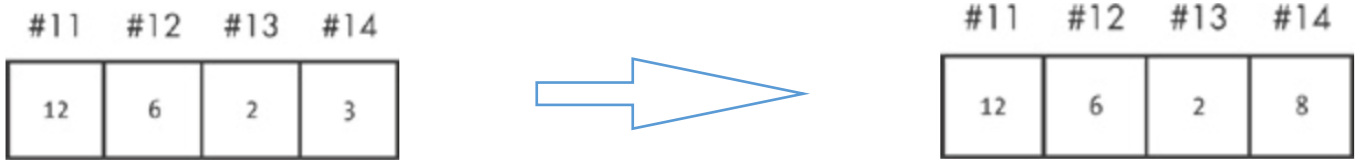
- program instructions in read-write RAM
- as opposed to computing machines requiring manual input or have non-exchangeable program instructions
- SPCs can be used for multiple purposes without changing the hardware

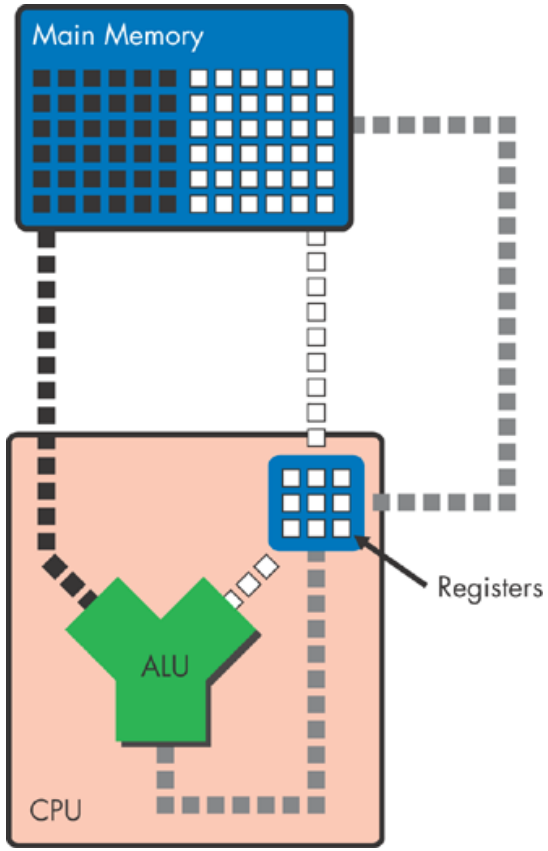


Example: Code Stream

- arithmetic instruction format: `instruction source1, source2, destination`
- memory instruction format: `instruction source, destination`
- example program:

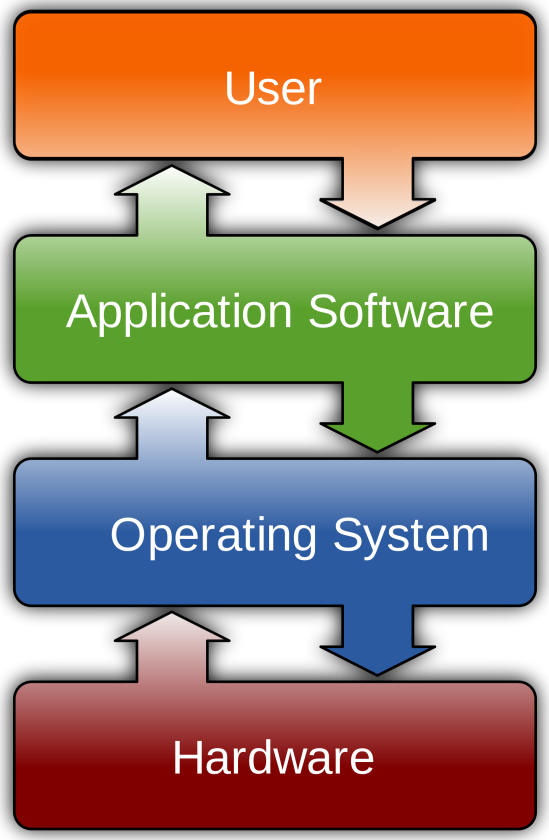
Line	Code	Comments
1	`load #12, A`	Read the contents of memory cell `#12` into register `A`.
2	`load #13, B`	Read the contents of memory cell `#13` into register `B`.
3	`add A, B, C`	Add the numbers in registers `A` and `B` and store the result in `C`.
4	`store C, #14`	Write the result of the addition from register `C` into memory cell `#14`.





Below the Program

- How do instructions in a programming language get translated into simple instructions in binary?
- Portable languages: How do instructions get interpreted according to the varying requirements of different architectures?
- multiple layers of program over hardware:



Program Hierarchy

- application software
- systems software:
    - compiler (programming language —> assembly language)
    - operating system (handles I/O, allocates storage, manages shared use of computer by multiple applications)
    - assembler (assembly language —> binary)
- hardware

Excursus: Booting

- How can a computer start at all? „(pulling itself up by its own bootstraps)“
- processor needs instruction how to start the operating system but on starting, memory is empty
- solution: default state of the processor is to get first instructions from specific address in (read-only-)memory where so called BIOS(-program) is stored
- BIOS runs basic tests of functionality and then directs to the bootloader(-program) which in turn loads the OS

Take Home Points

- File clerk model: computers read, modify and write data
- computer components: I/O, memory, CPU (datapath, CU), buses
- CPU controls and processes data flow
- CPU is working at the speed of the clock
- storage divided into registers, primary and secondary storage (hierarchical)
- programs executed in layers: application, compiler, OS, assembler (hierarchical)