Computer Architecture & Organization

- A *digital computer* is a machine that can solve problems by carrying out instructions given to it.
- A sequence of instructions describing how to perform a certain task is called a *program*.

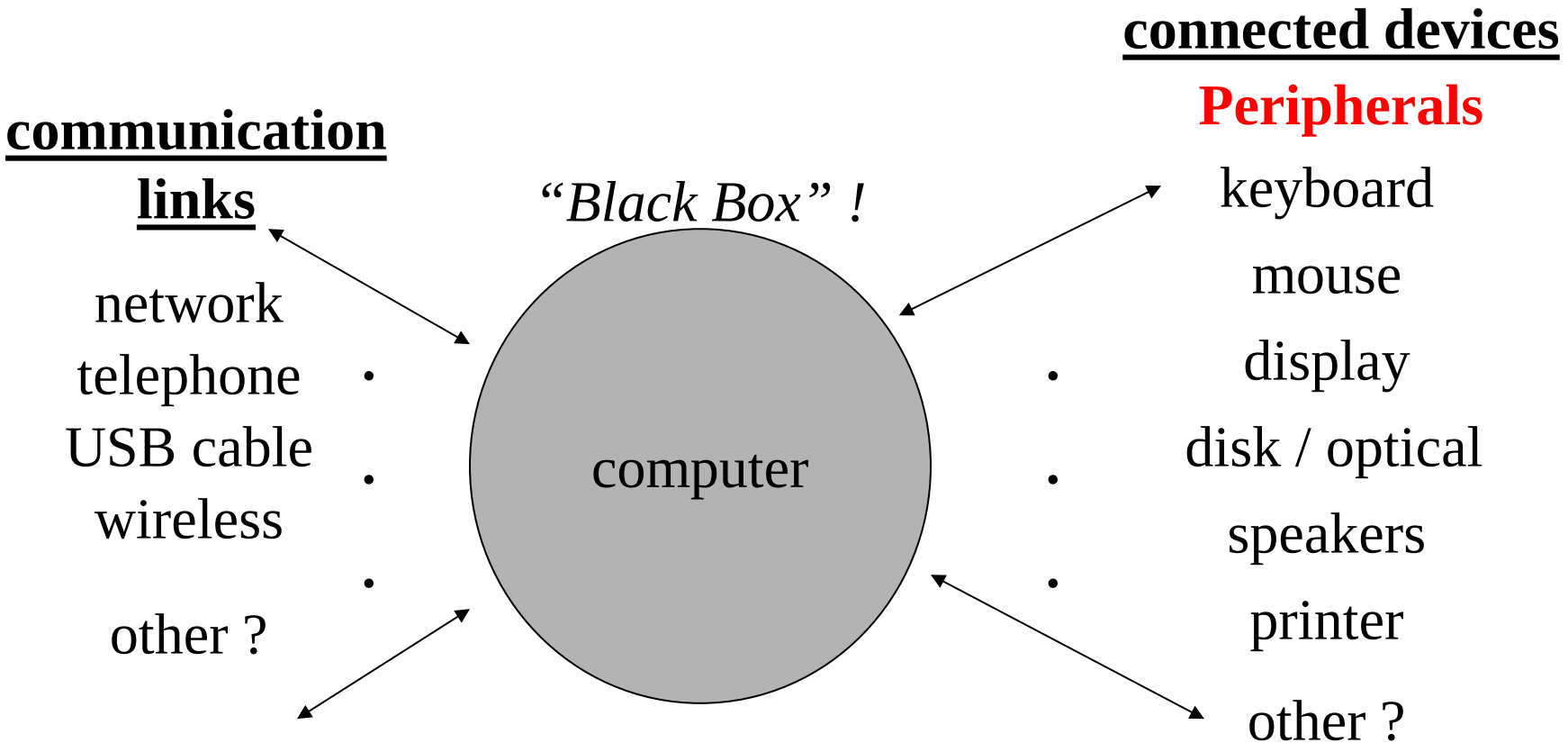
Architecture → attributes visible to the programmer

- Instruction set, number of bits used for data representation, I/O mechanisms, addressing techniques, etc.
- e.g. Is there a multiply instruction?

Organization → how features are implemented
"under the hood"

- e.g. Is there a hardware multiply unit or is it done by repeated addition?

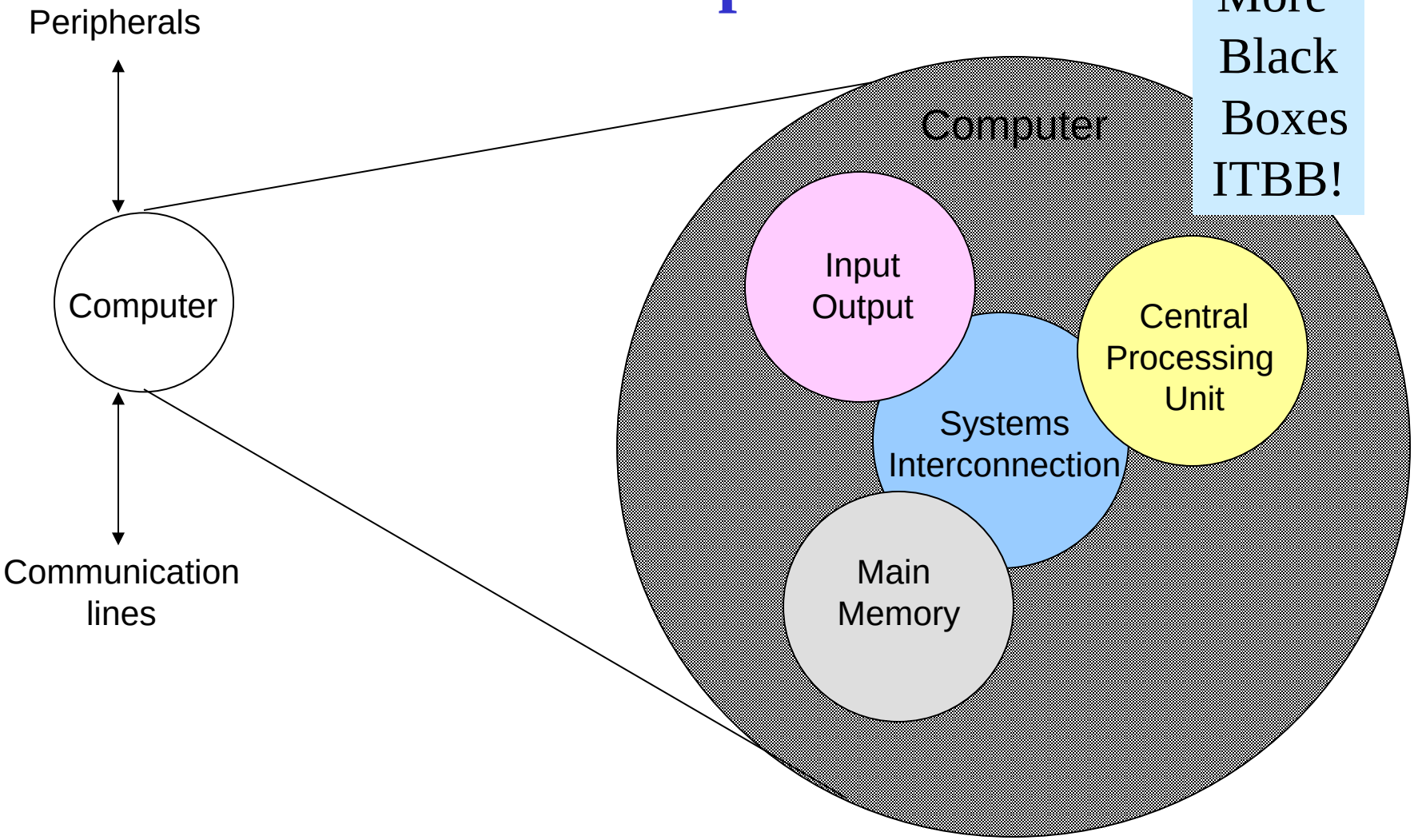
What Should I already know re Computer Arch & Org ?



SYSC 2001 will look inside the black box ! (ITBB)

peripherals and comm^{n} links are outside black box (SYSC3601)

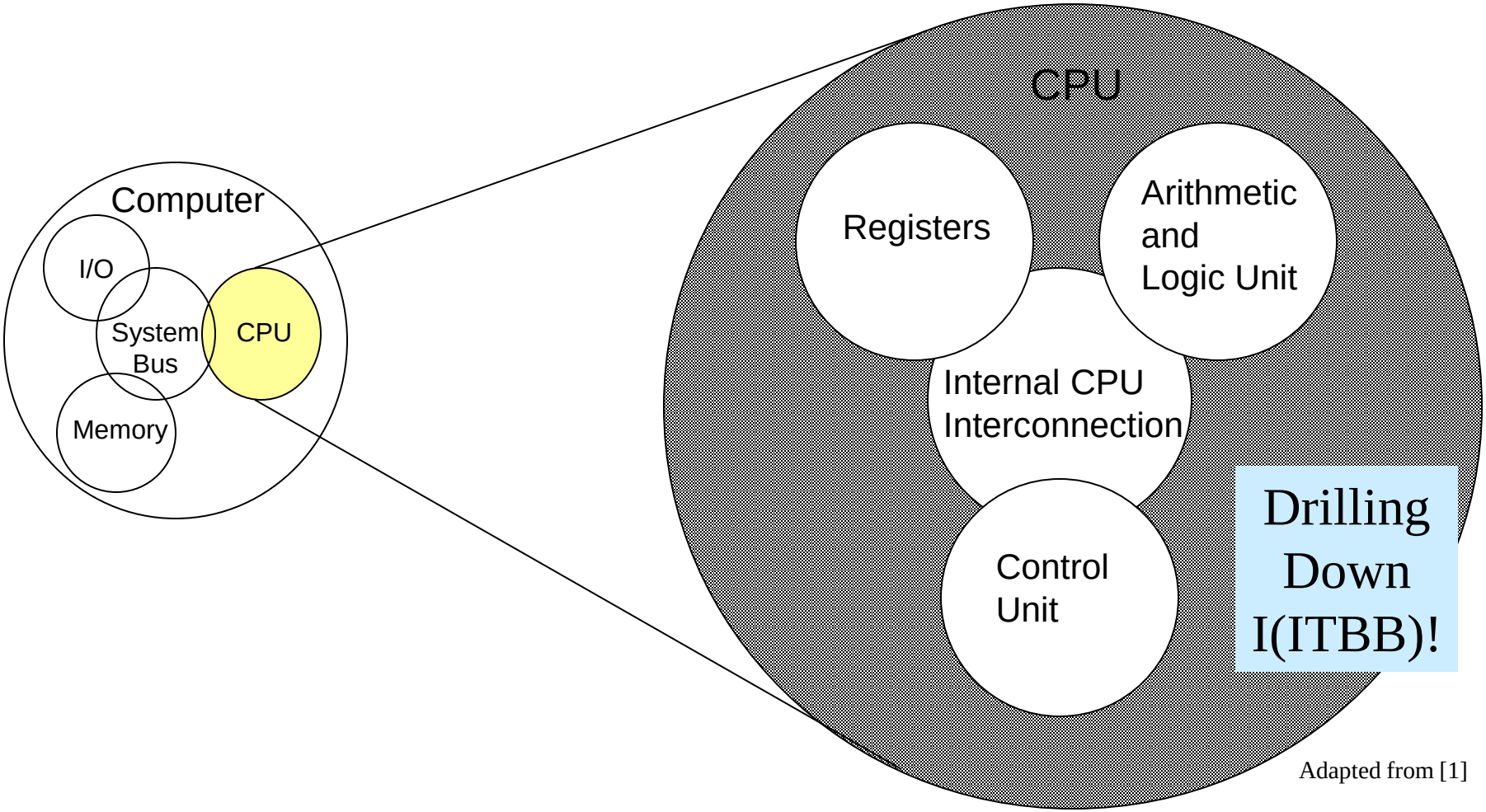
Structure - Top Level ITBB



More Black Boxes ITBB!

Adapted from [1]

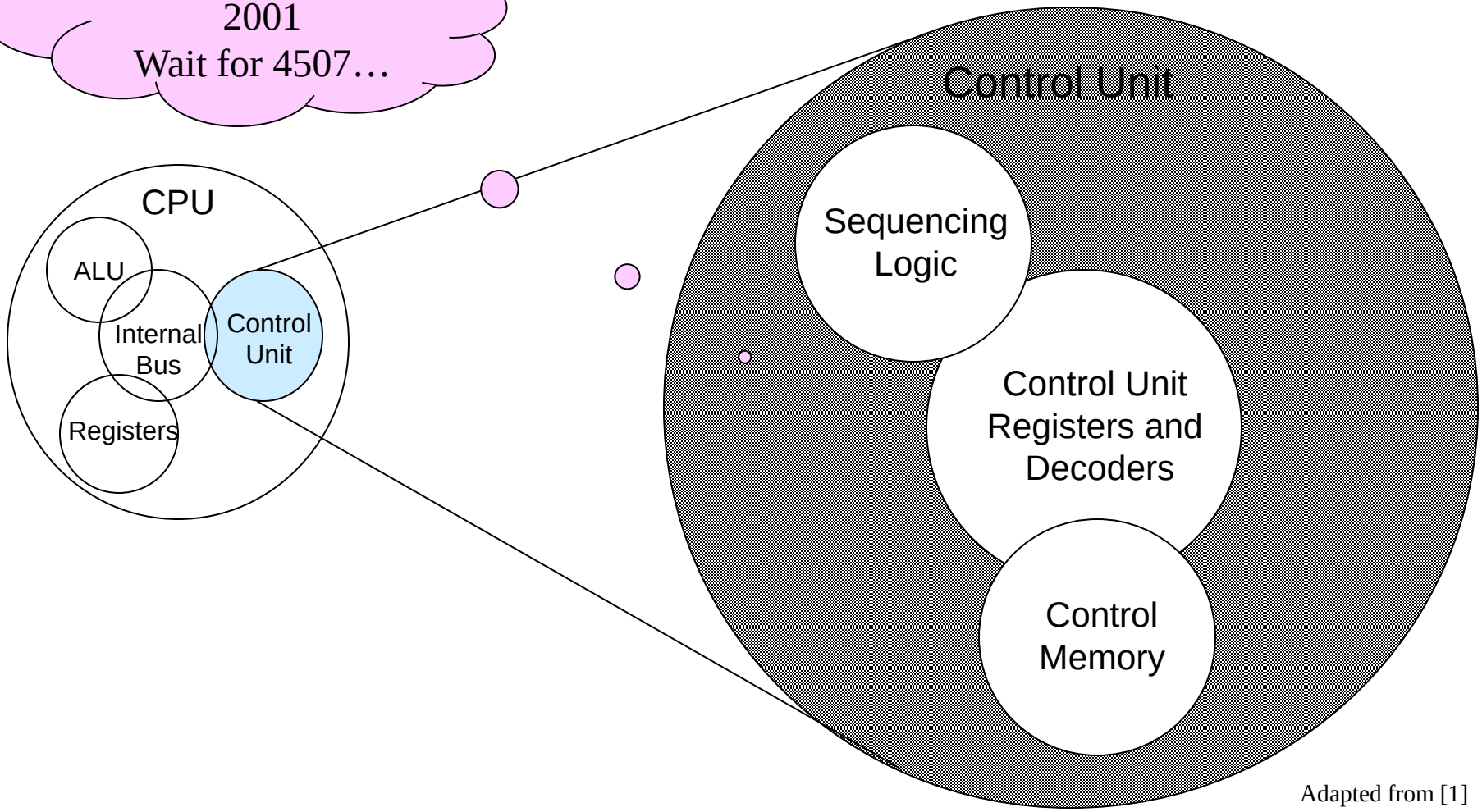
Structure - The CPU



Adapted from [1]

Structure - The Control Unit

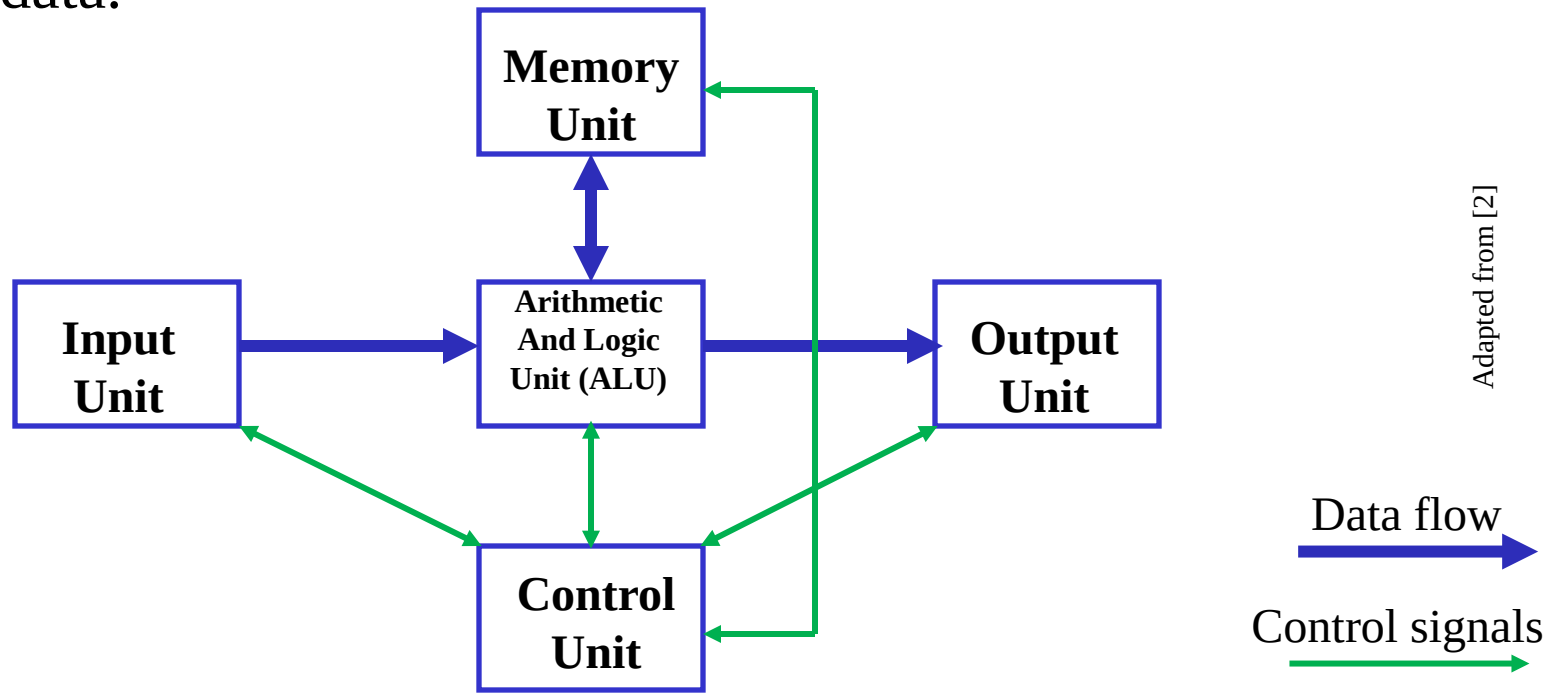
Too deep for SYSC 2001
Wait for 4507…



Adapted from [1]

The von Neumann Model

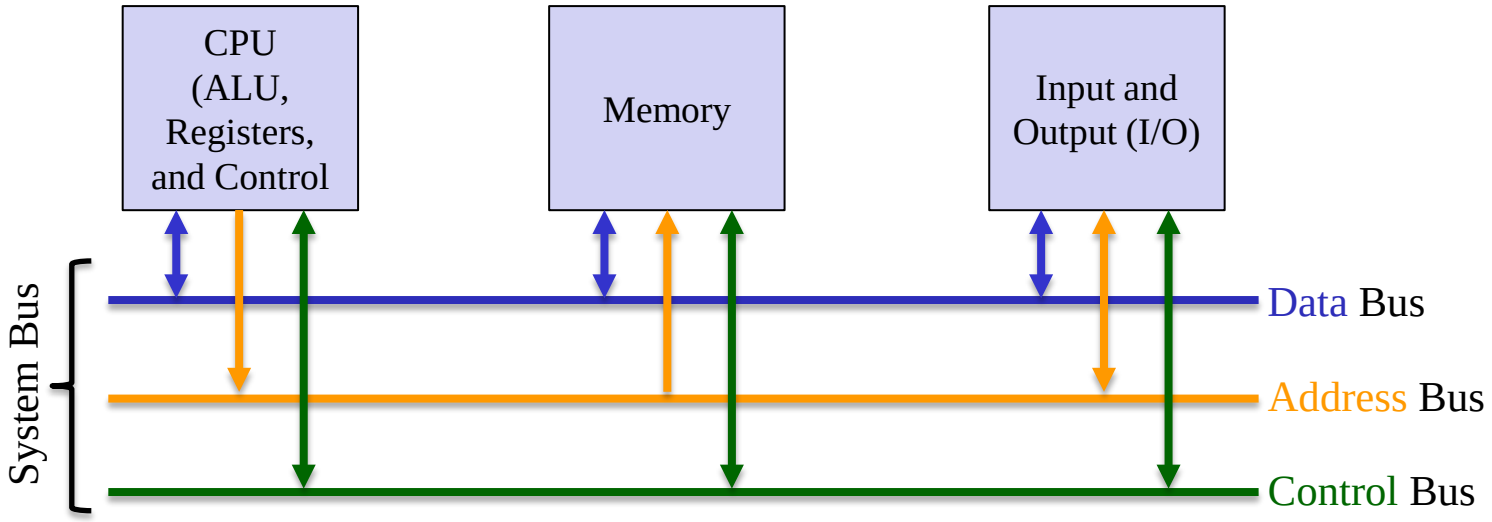
- The von Neumann model consists of five major components: arithmetic and logic unit (ALU), control unit, memory unit, input unit, and output unit.
- The stored program computer: a program is stored in the computer's memory along with the data, and can be changed as if it is data.



Adapted from [2]

The System Bus Model

- The system bus model is a refinement of the von Neumann model:
  - CPU (ALU and Control), memory, and input/output unit.
- The *system* bus provides communication among components
  - Made up of: data bus, the address bus, and the control bus.
  - Optional: *power bus*, separate *I/O bus*.

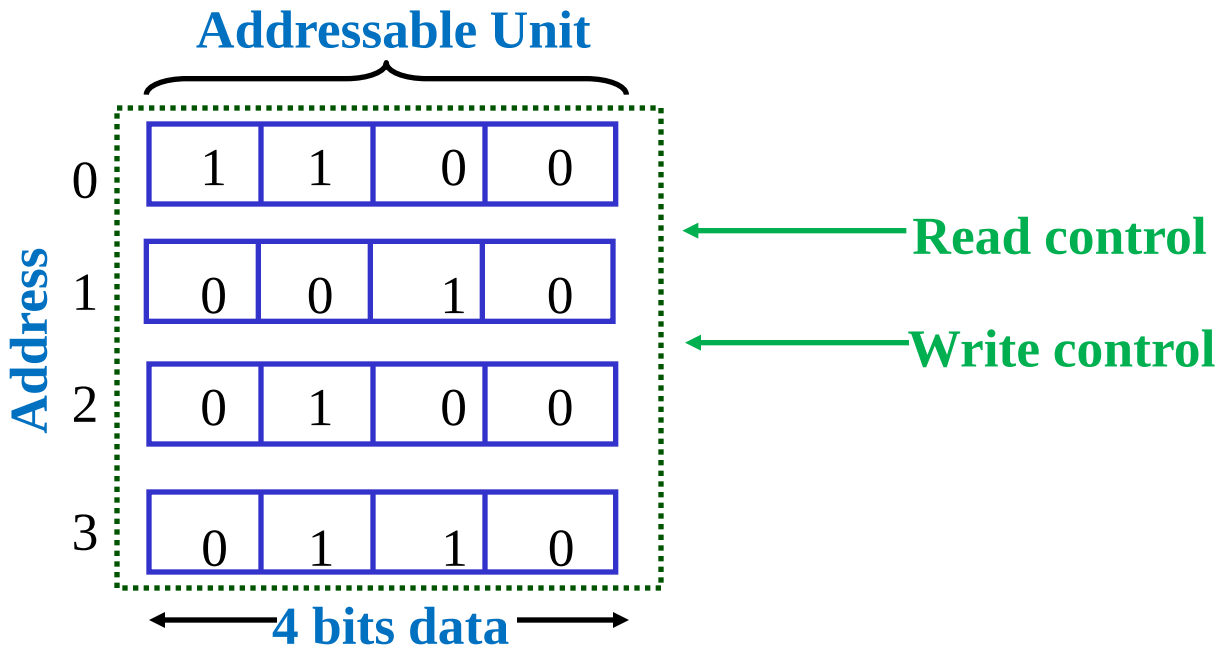


Adapted from [2]

The System Bus Model (Cont'd)

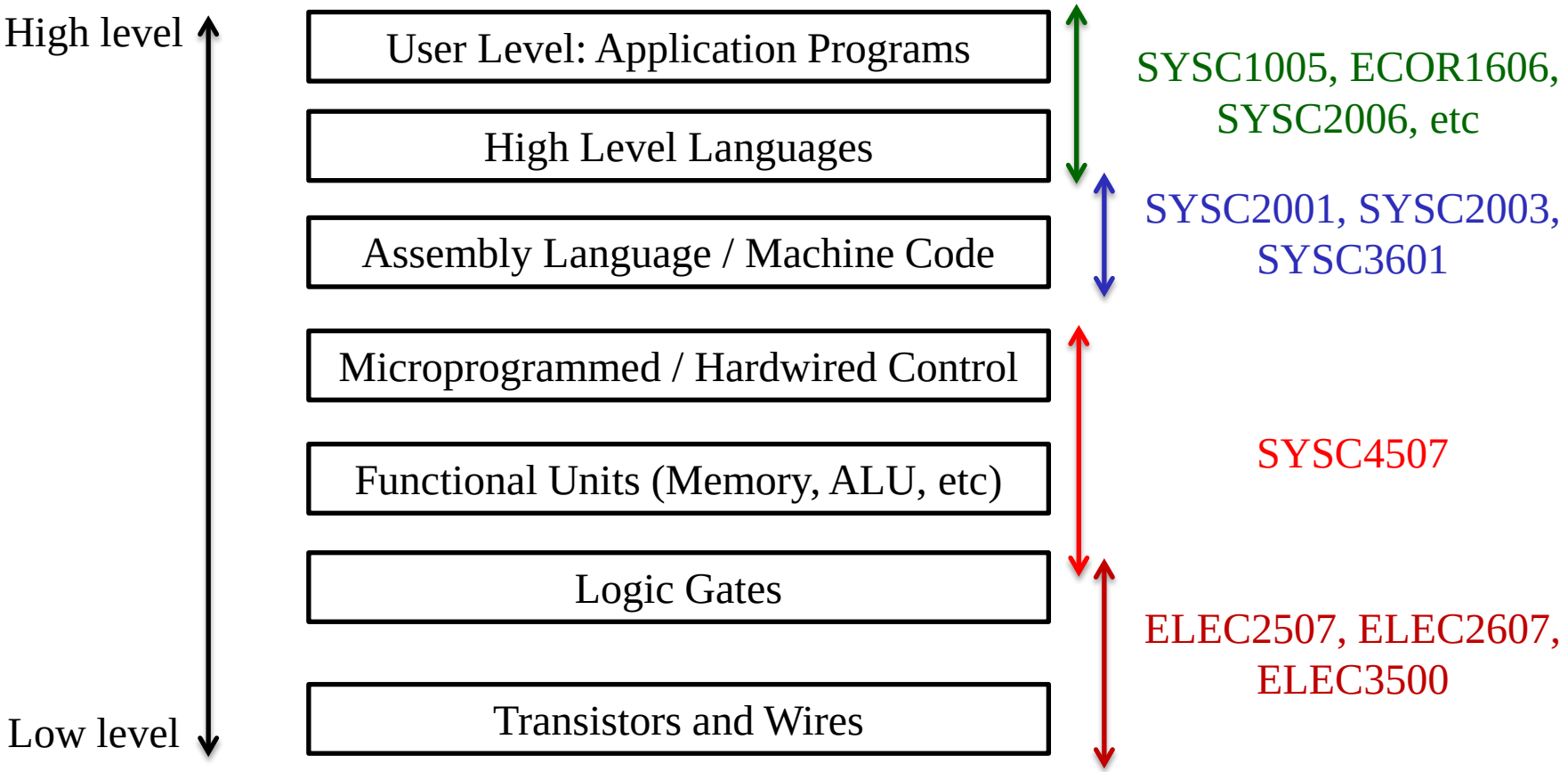
- Buses are made up of collections of wires that are grouped by function.
- The data bus moves data among the system components. It may be bidirectional or unidirectional (data-in bus and a data-out bus).
- The address bus contains the address of the communication entities. In case of memory unit, the address bus contains the address of the memory location where the data is to be read from or written to.
- The control bus can be thought of as coordinating access to the data bus and to the address bus.

4 × 4 Memory Unit



Levels of Machine View

- There are a number of levels on which we view a computer (the exact number is open to debate) from the *user level* down to the *transistor level*.
- Progressing from the top level downward, the levels become less abstract as more of the internal structure of the computer becomes visible.



Levels of Machine View (Cont'd)

- User level: the user sees the computer through a program that runs on it, and little (if any) of its internal structure is visible.
- High-Level language level: the programmer sees the data types and instructions of the high-level language but needs no knowledge of how those data types and instructions are actually implemented in the machine. It is the role of the compiler to map those to the actual computer hardware.
- Machine language level: set of fundamental instructions that the machine can execute. Machine language deals with hardware issues such as registers and the transfer of data between them.
  - The compiler translates the source code to the actual machine instructions, referred to as *machine language* or machine code.
  - Machine code is just a collection of 1's and 0's, sometimes referred to as machine binary code, or just binary code.
  - The collection of machine instructions for a given machine is referred to as the instruction set of that machine.

Levels of Machine View (Cont'd)

- Assembly language level: Alphanumeric equivalent of the machine code. It provides ordinary language mnemonics for each machine code, such as *MOVE data, Acc*. Easier than 1's and 0's.
  - Assembler: is a program that translates the assembly language program into the machine language.

- Control level: the control unit interprets the machine instructions or codes one by one, causing the specified action, such as register transfer, to occur. There are two ways to implement the control unit:
  - Hardwiring: the control signals that affect the register transfers are generated from a block of digital logic components. It is fast.
  - Microprogramming: is a small program written in an even lower-level language, and implemented in the hardware, whose job is to interpret the machine language instructions. It is also referred to as firmware because it spans both the hardware and software. Firmware is executed by a microcontroller that executes the actual microinstructions. It is slower but simpler.

Levels of Machine View (Cont'd)

- Functional unit level: includes internal CPU registers, the ALU, memory, etc.
- Logic gates level: the logic gates make up the functional units. They implement the lowest-level logical operations upon which the computer's functioning depends.
- Transistor's and wires level: electrical components such as transistors and wires make up the logic gates. At this level, the functioning of the computer is concerned with electrical properties such as voltage, current, and signal propagation delays.

  - Interaction between levels: the distinctions within and between levels are frequently blurred. For instance, for a CPU without a floating point unit, floating point instructions can be replaced by a sequence of machine language instructions that emulate the floating point instructions using the existing integer instructions.

Instruction Set Architecture

- Instruction set architecture (ISA):
  - Collection of instruction set + functional units
  - Corresponds to the Assembly/machine code level
    - Between the high-level language view, where little or none of the machine hardware is visible, and the control level, where machine instructions are interpreted as register transfer actions at the functional unit level.
- The *computer architect* views the system at all levels driven by performance requirements, such as speed of program execution, and cost constraints.
  - Designs ISA, and the hardware for implementing the instructions
  - Designs for performance at the lowest cost
  - Uses performance tools (benchmarks) to see if the goals are met

Brief History of Computer Evolution

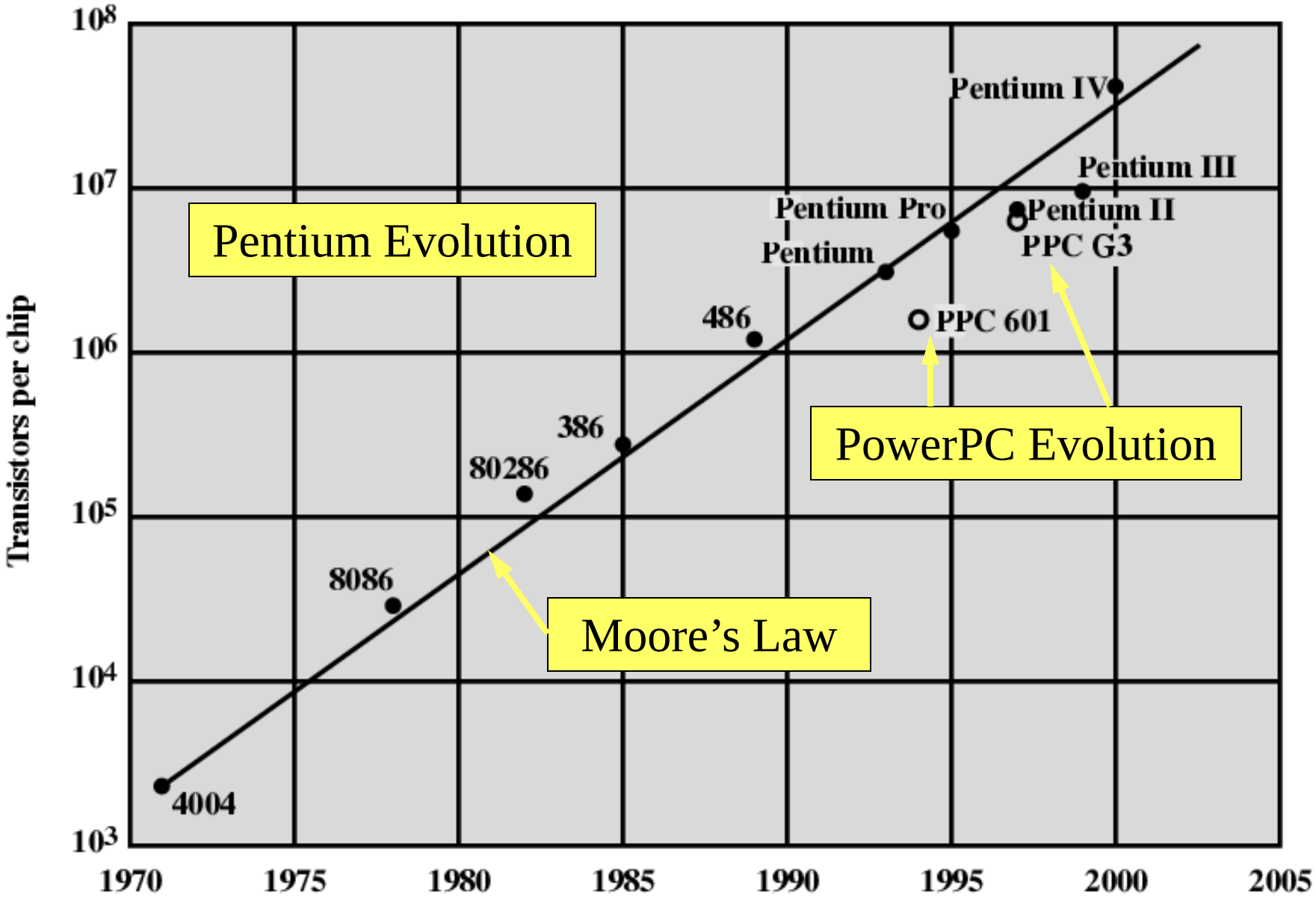
Two phases:

1. before VLSI 1945 – 1978
   - ENIAC
   - IAS
   - IBM
   - PDP-8

   see text discussion, Ch 2
2. VLSI 1978 → present day
   - microprocessors !

VLSI = Very Large Scale Integration

Growth in CPU Transistor Count



Speeding Up the Processor

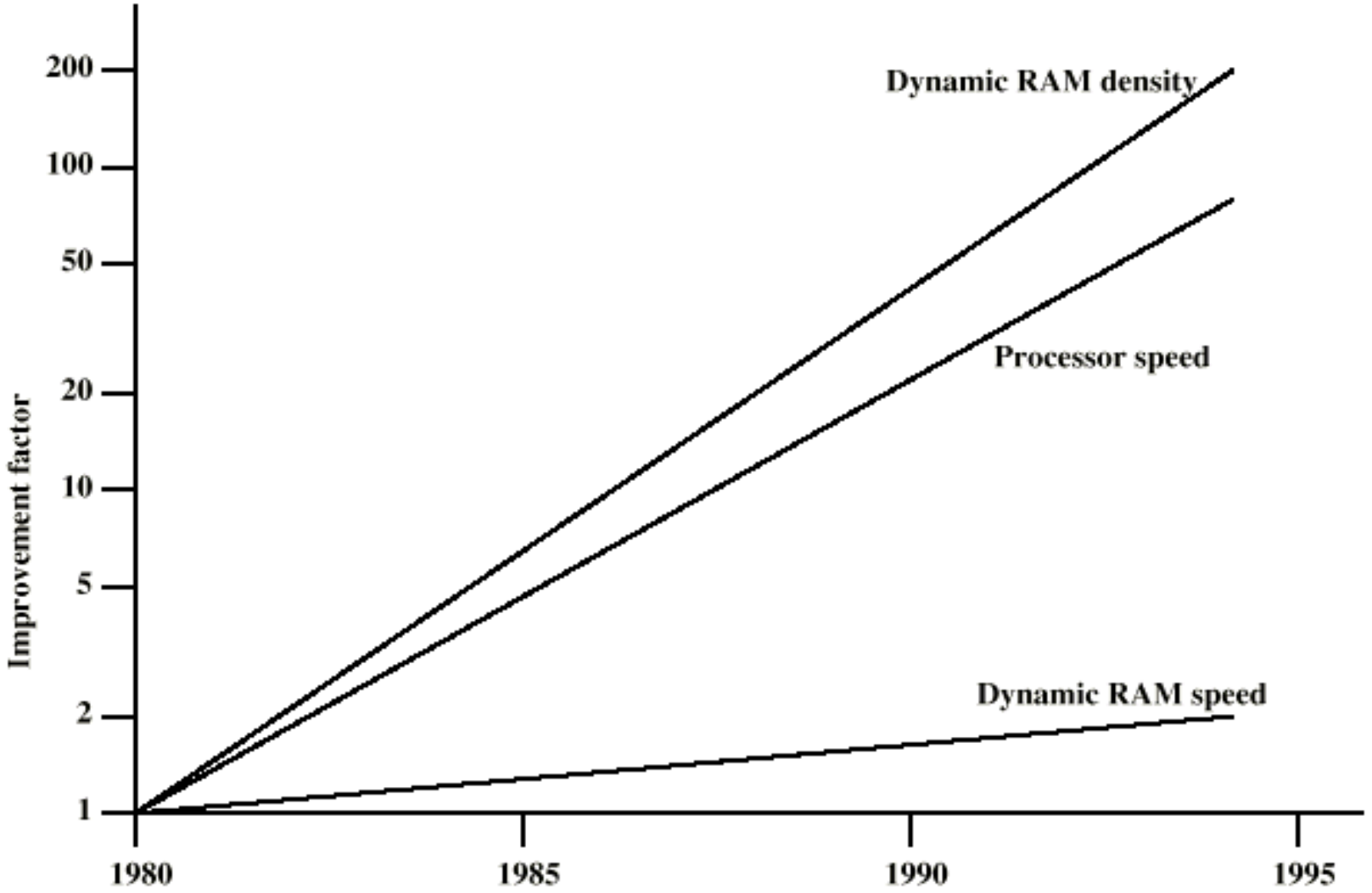
- Pipelining
- On board cache
- On board L1 & L2 cache
- Branch prediction
- Data flow analysis
- Speculative execution

we'll see some of these as the course progresses

But Performance Mismatch!

- Processor speed increased
- Memory capacity increased
- Memory speed lags behind (and increasing slower than) processor speed

DRAM and Processor Characteristics



Some Solutions

- Increase number of bits retrieved at one time
  - Make DRAM “wider” rather than “deeper”
- Change DRAM interface
  - Cache
- Reduce frequency of memory access
  - More complex cache, and cache on chip
- Increase interconnection bandwidth
  - High speed buses
  - Hierarchy of buses

References

[1] W. Stallings, Computer Organization & Architecture, 9th Edition, Peason-Prentice Hall, 2012

[2] M. J. Murdocca and V.P Heuring, “Principles of Computer Architecture”, Prentice Hall, 2000