361-1-4201
Computer Architecture
Lecture 1: Fundamental Concepts and ISA
Lecturer: Dr. Guy Tel-Zur

Based on a course by Prof. Onur Mutlu

Carnegie Mellon University

Spring 2015, 1/14/2014

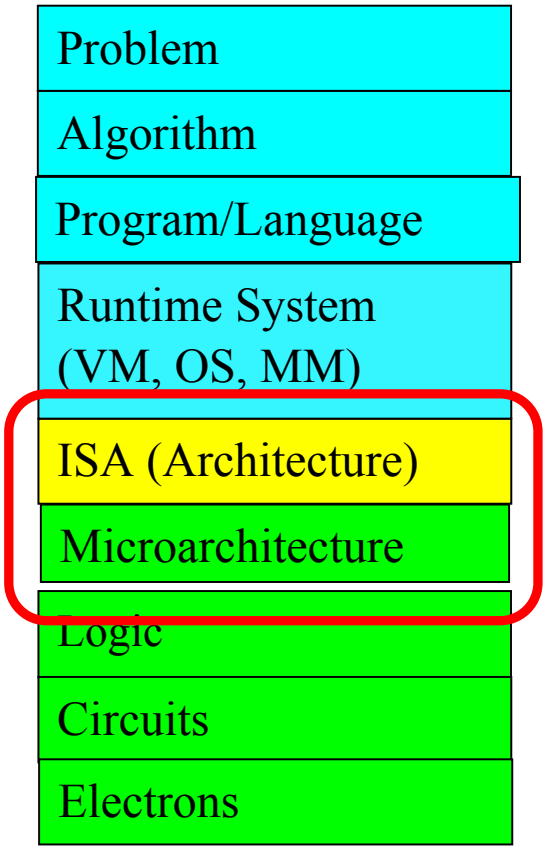
~~V 1.0, 19/03/2020~~
~~V 1.1, 3/3/2021~~
~~V 1.2 24/3/2022~~
V 1.3 8/3/2023

Aside: What Is An Algorithm?

- Step-by-step procedure where each step has three properties:
  - Definite (precisely defined)
  - Effectively computable (by a computer)
  - Terminates

אלגוריתם הוא דרך שיטתית וחד - משמעית לביצוע של משימה מסוימת, במספר סופי של צעדים. (ויקיפדיה, פברואר 2020).

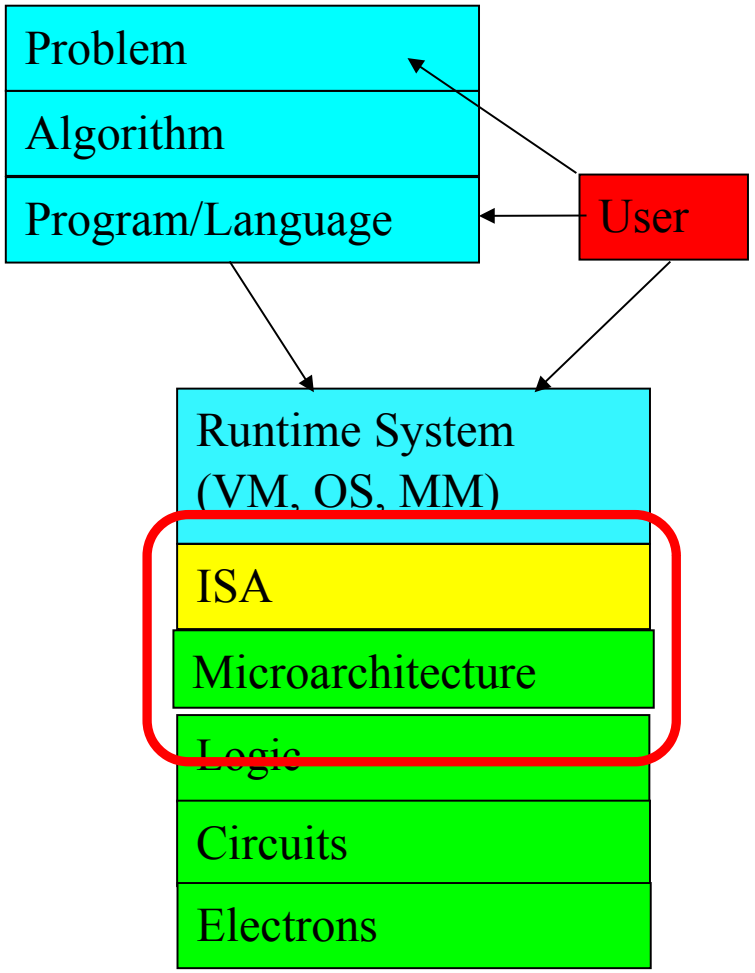
Computer Architecture in Levels of Transformation



- קריאה: Patt, "Requirements, Bottlenecks, and Good Fortune: Agents for Microprocessor Evolution," Proceedings of the IEEE 2001.

Levels of Transformation, Revisited

- A **user-centric** view: computer designed for users



- The entire stack should be optimized for user

What is Computer Architecture?

- The science and art of designing, selecting, and interconnecting hardware components and designing the hardware/software interface to create a computing system that meets functional, performance, energy consumption, cost, and other specific goals.

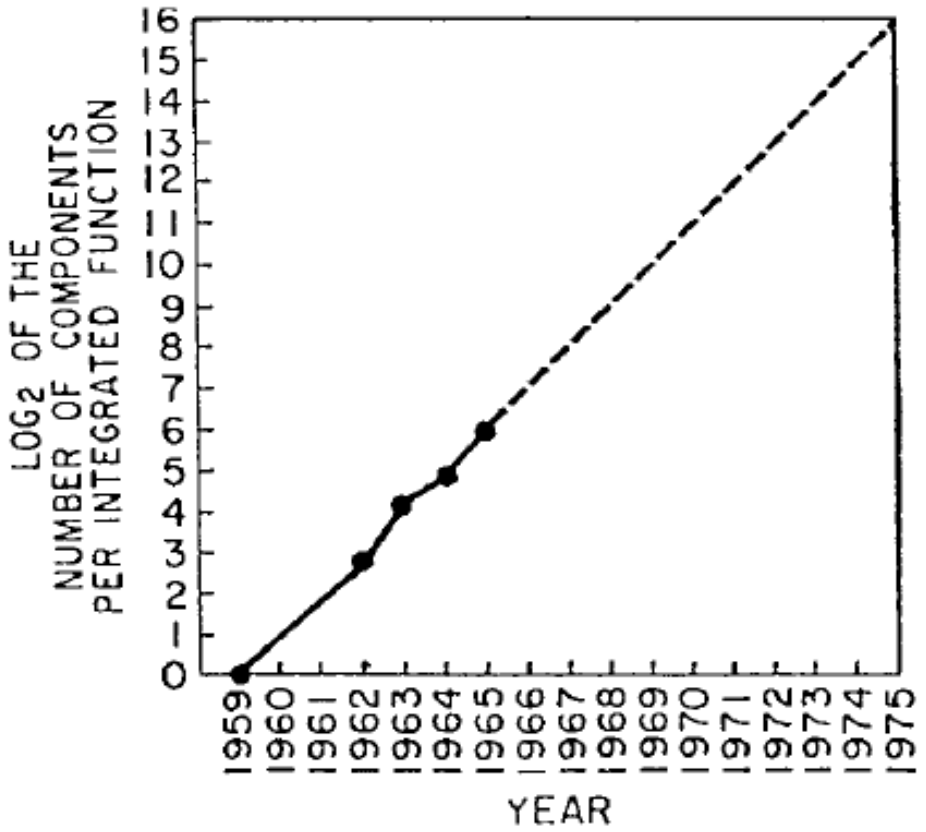
- We will soon distinguish between the terms ***architecture***, and ***microarchitecture***.

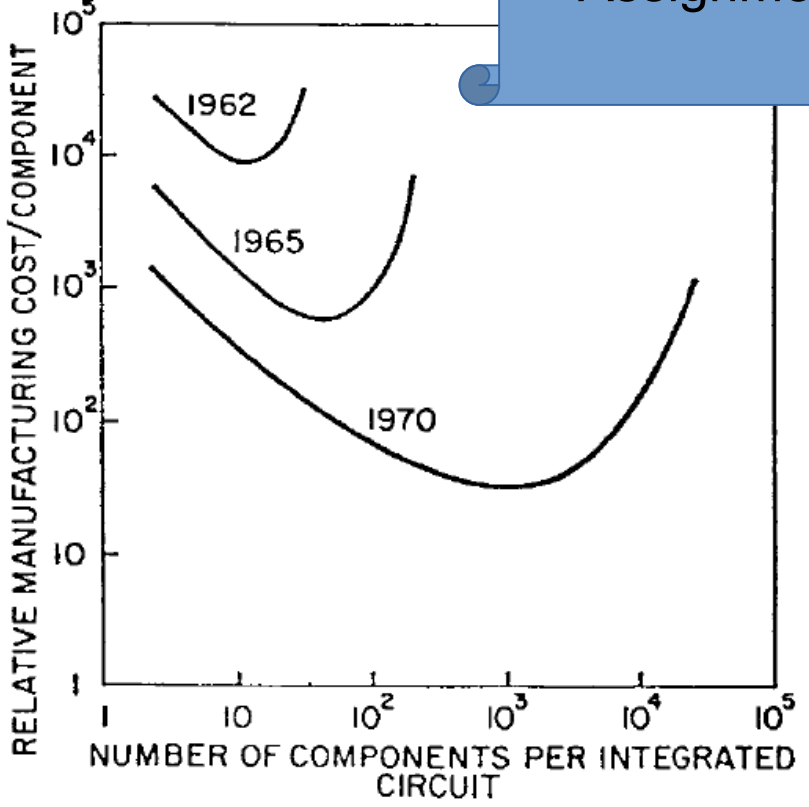
נגדיר אריכטקטורת מחשב:

המדע והאומנות לתכנן, לבחור ולקשור בין רכיבי חומרה כמו גם לתכנן ממשק בין החומרה לתכנה כדי ליצור מערכת מחשב שעומדת בדרישות הפונקציונאליות, בביצועים, בצריכת האנרגיה, בעלות ובמטרות ספציפיות נוספות.

An Enabler: Moore's Law

Reading Assignment!





לדחוס

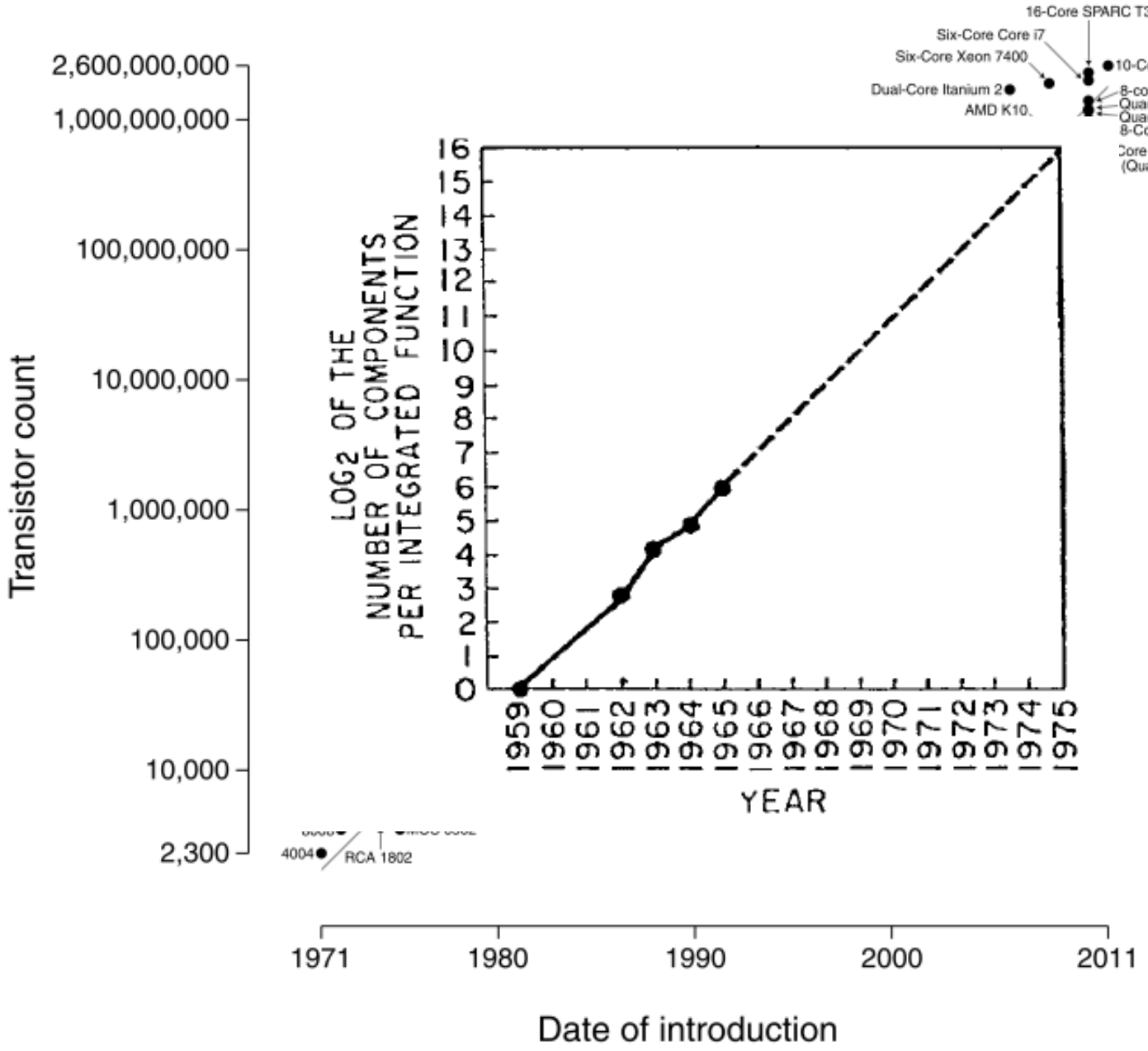
Moore, "Cramming more components onto integrated circuits," Electronics Magazine, 1965.

Component counts double every other year

המאמר ניתן להורדה גם מאתר ה- moodle. בסה"כ 3 עמודים.

Image source: Intel

Microprocessor Transistor Counts 1971-2011 & Moore's Law



Number of transistors on an integrated circuit doubles ~ every two years

Image source: Wikipedia

Required Reading

Reading Assignment!

- Moore, "Cramming more components onto integrated circuits," Electronics Magazine, 1965.
- Only 3 pages
- A quote:

  "With unit cost falling as the number of components per circuit rises, by 1975 economics may dictate squeezing as many as 65 000 components on a single silicon chip."
- Another quote:

  "Will it be possible to remove the heat generated by tens of thousands of components in a single silicon chip?"

המאמר ניתן להורדה גם מאתר ה- moodle. בסה"כ 3 עמודים.

What Do We Use These Transistors for?

Reading Assignment!

- Your readings for this week should give you an idea…

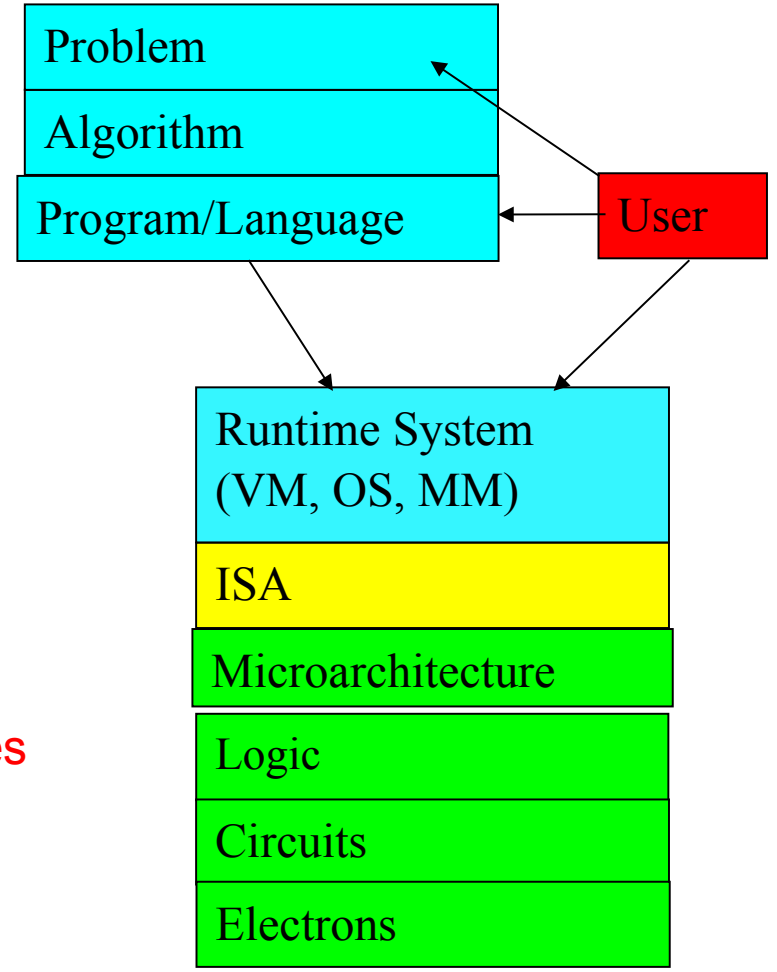
- **Patt, "Requirements, Bottlenecks, and Good Fortune: Agents for Microprocessor Evolution," Proceedings of the IEEE 2001.**

Computer Architecture Today (I)

- Today is a very exciting time to study computer architecture
- Industry is in a large **paradigm shift** (to **multi-core** and beyond) – many different potential system designs possible
- Many difficult problems *motivating* and *caused by* the shift
  - Power/energy constraints → multi-core?
  - Complexity of design → multi-core?
  - Difficulties in technology scaling → new technologies?
  - Memory wall/gap
  - במשך שנים שופרו ביצועי המעבדים, אך ביצועי הזיכרון שופרו הרבה פחות ולכן הזיכרון הפך להיות צוואר בקבוק!
  - Reliability wall/issues
  - Programmability wall/problem
  - Huge hunger for data and new data-intensive applications
- No clear, definitive answers to these problems

Computer Architecture Today (II)

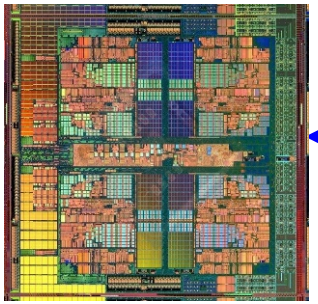
- These problems affect all parts of the computing stack – if we do not change the way we design systems



- No clear, definitive answers to these problems

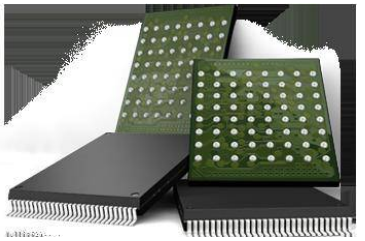
Computer Architecture Today (III)

- Computing landscape is very different from 10-20 years ago.
- Both UP (software and humanity trends) and DOWN (technologies and their issues), FORWARD and BACKWARD, and the resulting requirements and constraints.

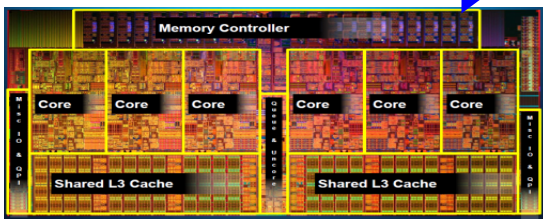


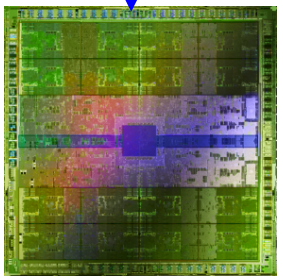
Heterogeneous Processors and Accelerators

Hybrid Main Memory



Persistent Memory/Storage





General Purpose GPUs

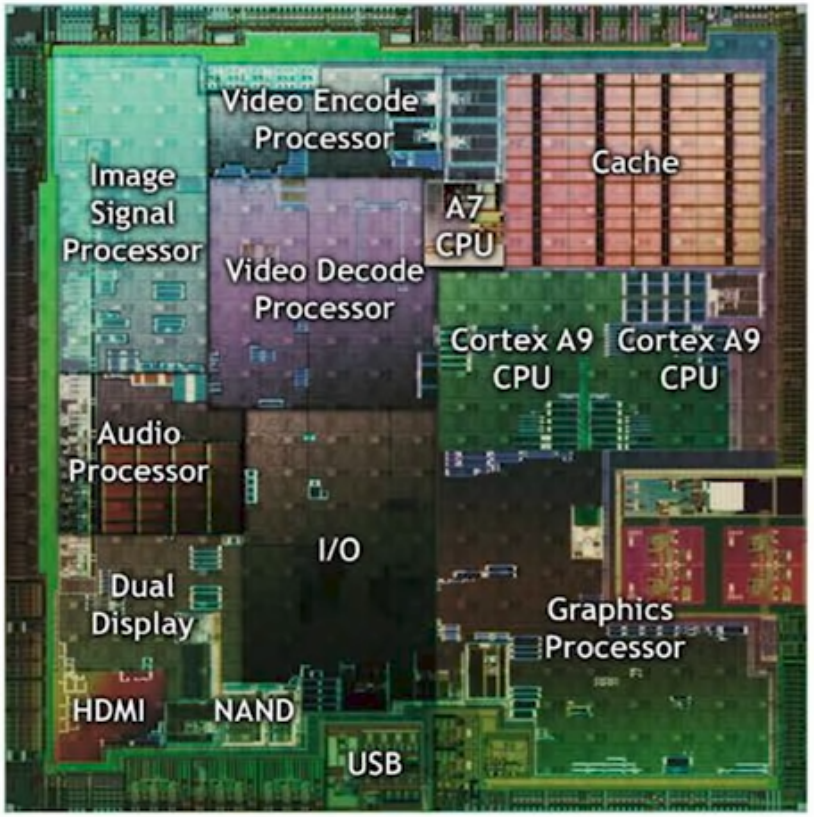
Every component and its interfaces, as well as entire system designs are being re-examined

כל מה שמוצג בתמונות בשקף לא היה קיים לפני 20 שנה

Computer Architecture Today (IV)

How many ISAs does a chip speak?

- Applications processor (usually ARM)
- Graphics processors
- Image processors
- Radio DSPs
- Audio DSPs
- Security processors
- Power-management processor
- *> dozen ISAs on some SoCs – each with unique software stack*



NVIDIA Tegra SoC

Credit: Dr. MeganWachs - Keynote RISC-V and FPGAs: "Open Source Hardware Hacking",
https://hackaday.com/2020/01/27/supercon-keynote-megan-wachs-breaks-down-risc-v/
https://www.youtube.com/watch?v=vCG5_nxm2G4

Computer Architecture Today (V)

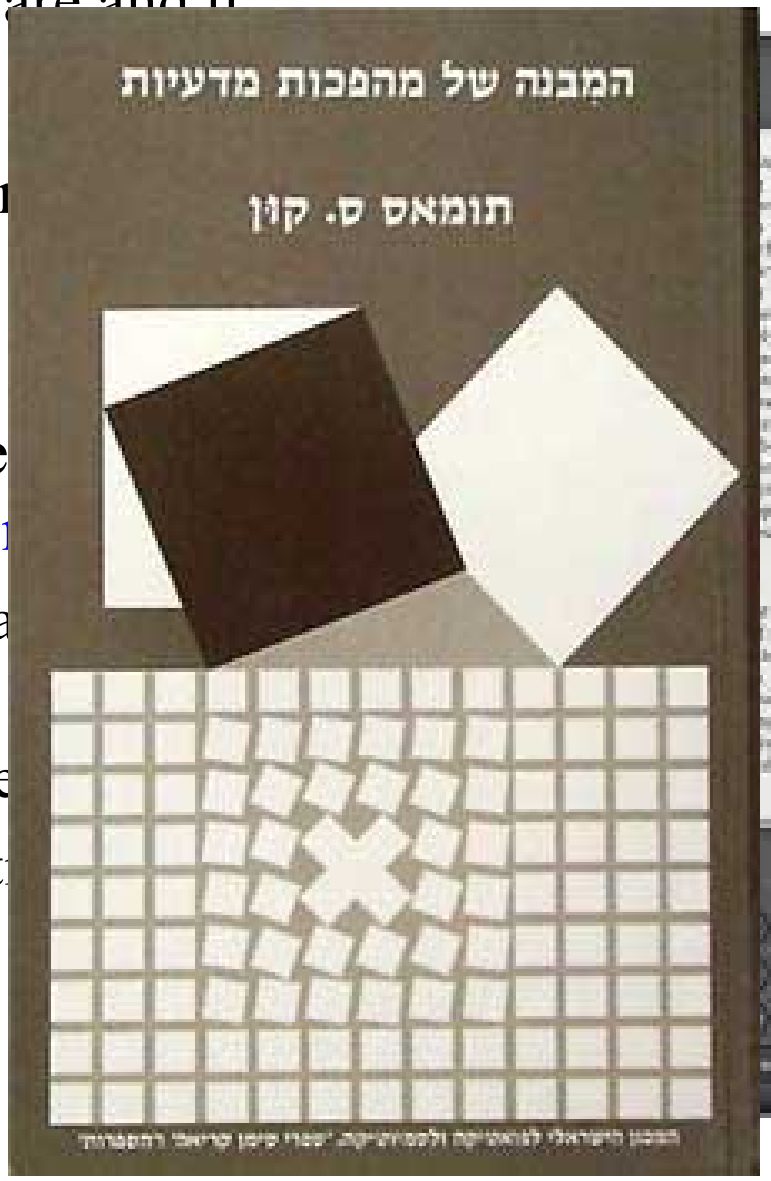
- You can **revolutionize** the way computers are built, if you **understand both the hardware and the software** (and change each accordingly)

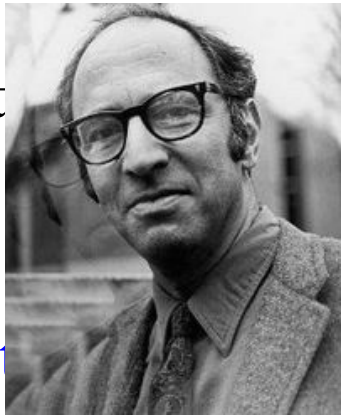
- You can ***invent new paradigms*** for computation, communication, and storage

- Recommended book: Thomas Kuhn, "The Structure of Scientific Revolutions" (1962)
  - Pre-paradigm science: no clear consensus in the field
  - Normal science: dominant theory used to explain/improve things (business as usual); exceptions considered anomalies
  - Revolutionary science: underlying assumptions re-examined

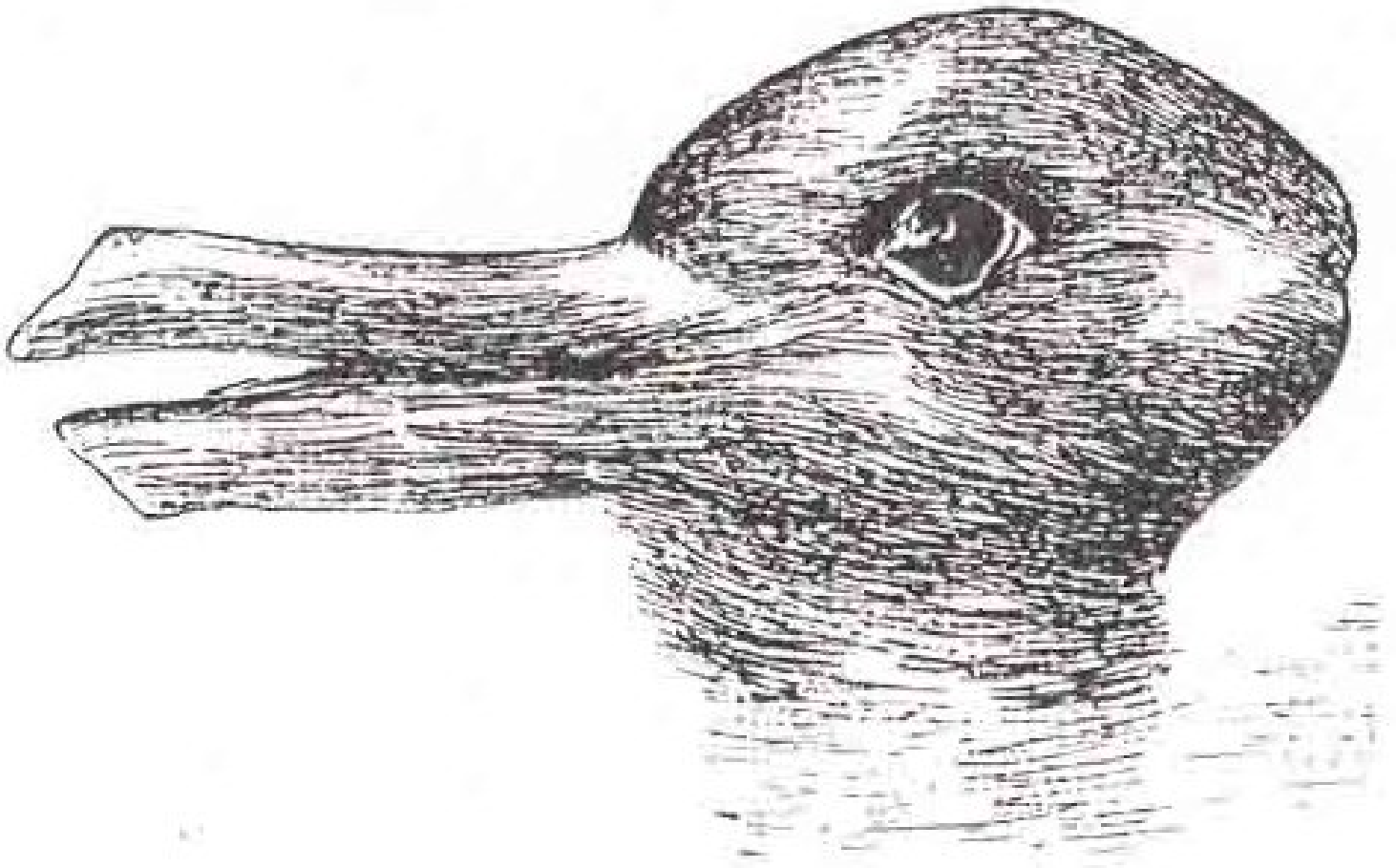
Computer Architecture Today (V)

- You can revolutionize the way computers are built, if you understand both the hardware and th[illegible] ccordingly)
- You can i[illegible] ommunicat[illegible] storage
- Recomme[illegible] en[illegible] Revolutio[illegible]
  - Pre-para[illegible]
  - Normal [illegible] ngs (business as usual); e[illegible]
  - Revolut[illegible]





Paradigm Shift



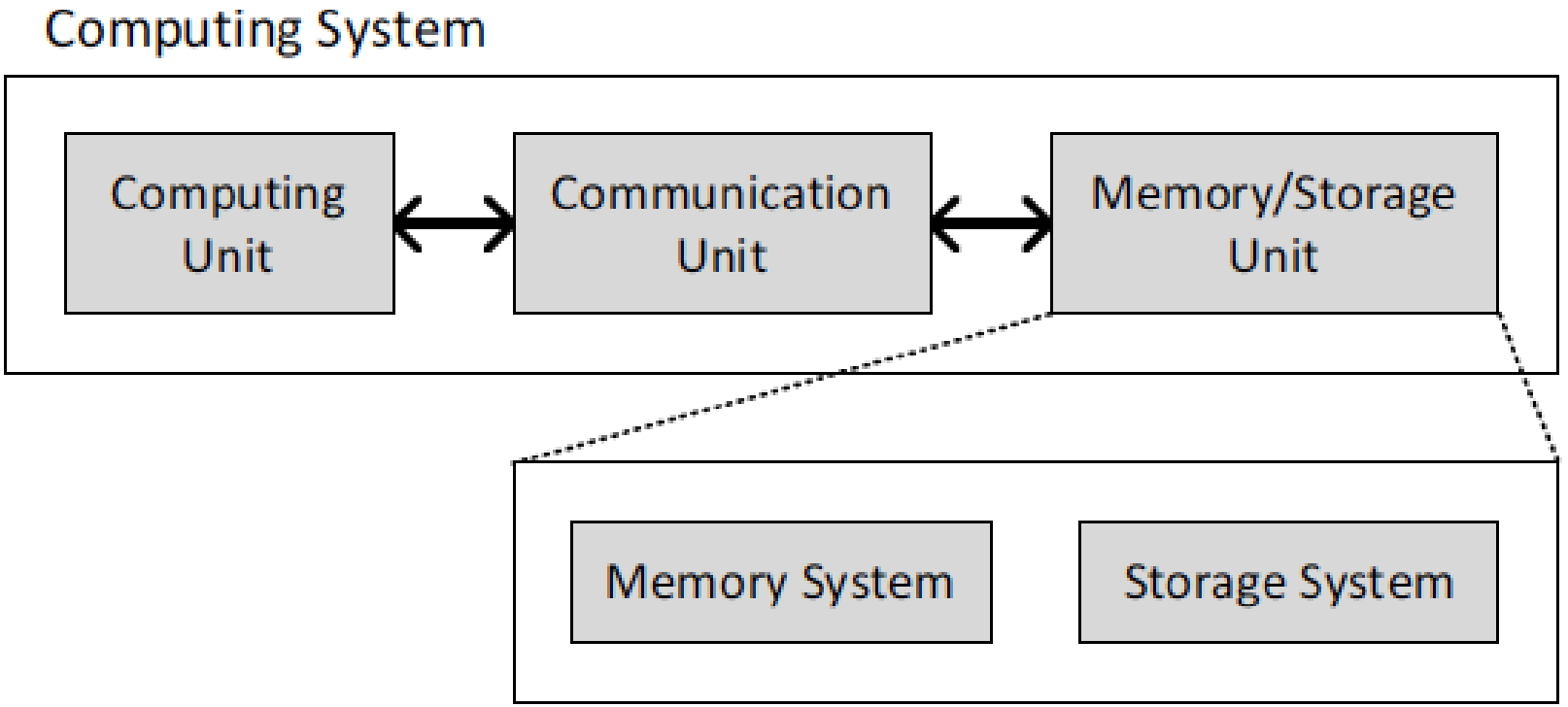
Fundamental Concepts

What is A Computer?

- Three key components
  - Computation (-→ datapath & control)
  - Communication (-→ input & output)
  - Storage (memory)

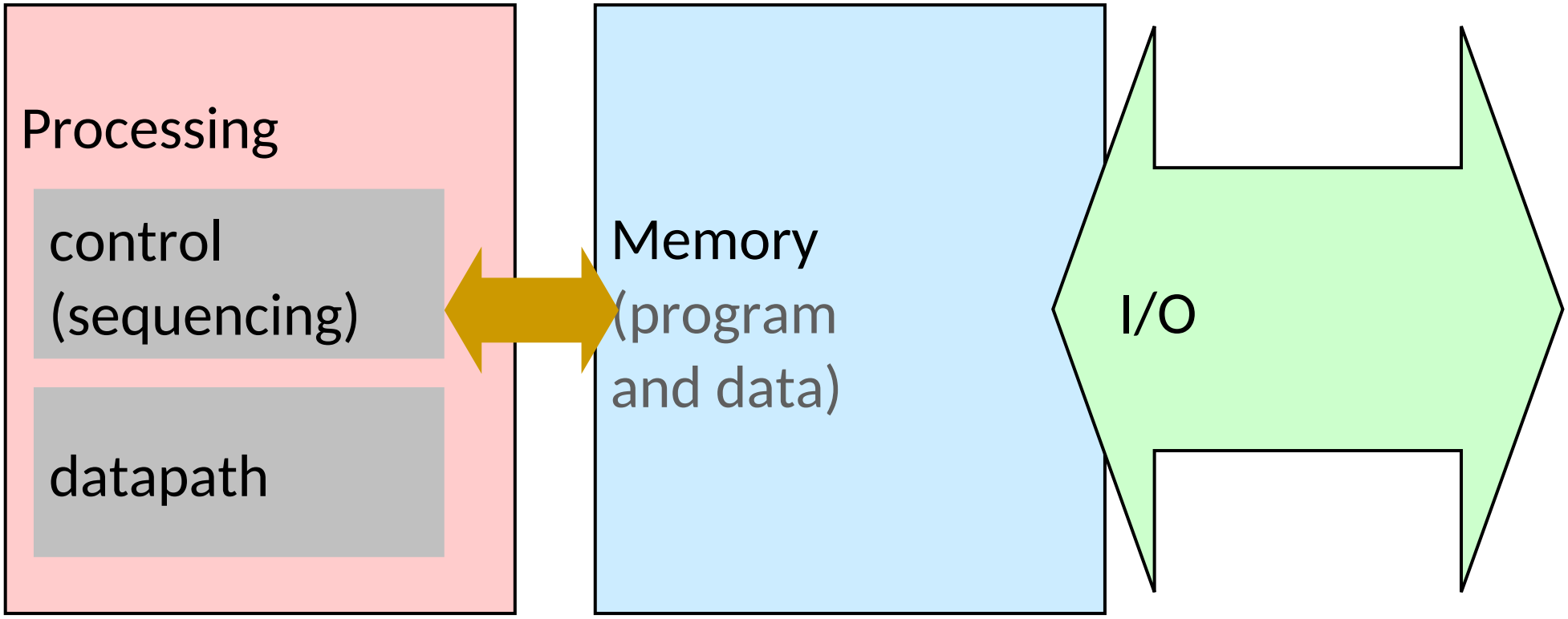
יחד עם זאת, ראינו במצגת הראשונה כי H&P מגדירים 5 מרכיבים:

1. Input
2. Output
3. Memory
4. Datapath
5. Control



What is A Computer?

- We will cover all three components



The Von Neumann Model/Architecture

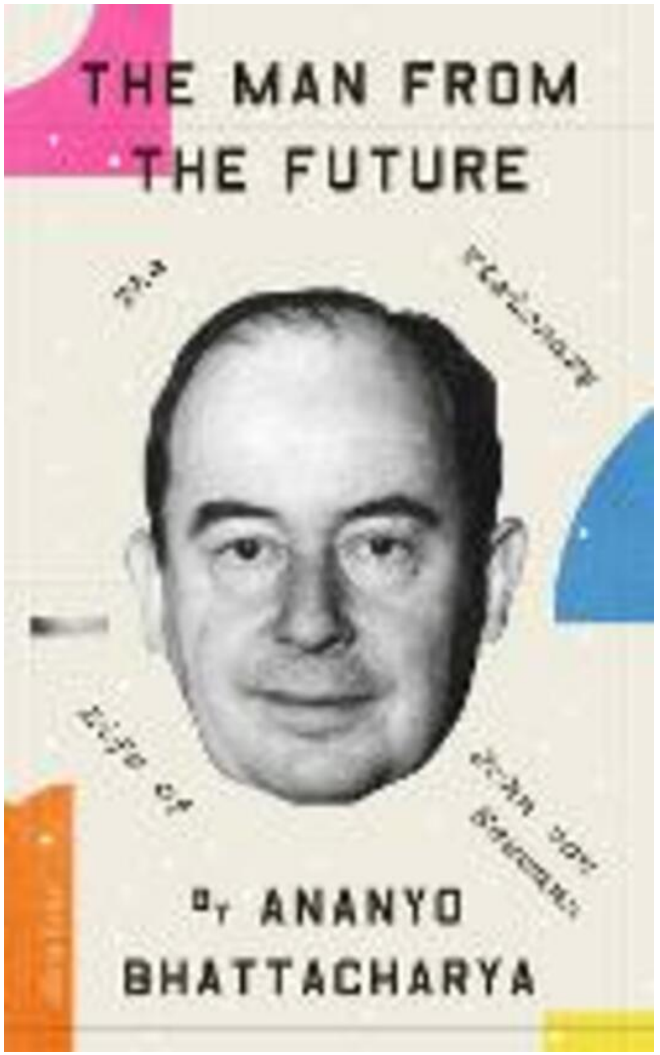
- Also called *stored program computer* (instructions in memory). Two key properties:

- **Stored program**
  - Instructions stored in a linear memory array
  - Memory is unified between instructions and data
    - The interpretation of a stored value depends on the control signals

When is a value interpreted as an instruction?

- **Sequential instruction processing**
  - One instruction processed (fetched, executed, and completed) at a time
  - Program counter (instruction pointer) identifies the current instr.
  - Program counter is advanced sequentially except for control transfer instructions

The Man from the Future: The Visionary Life of John von Neumann



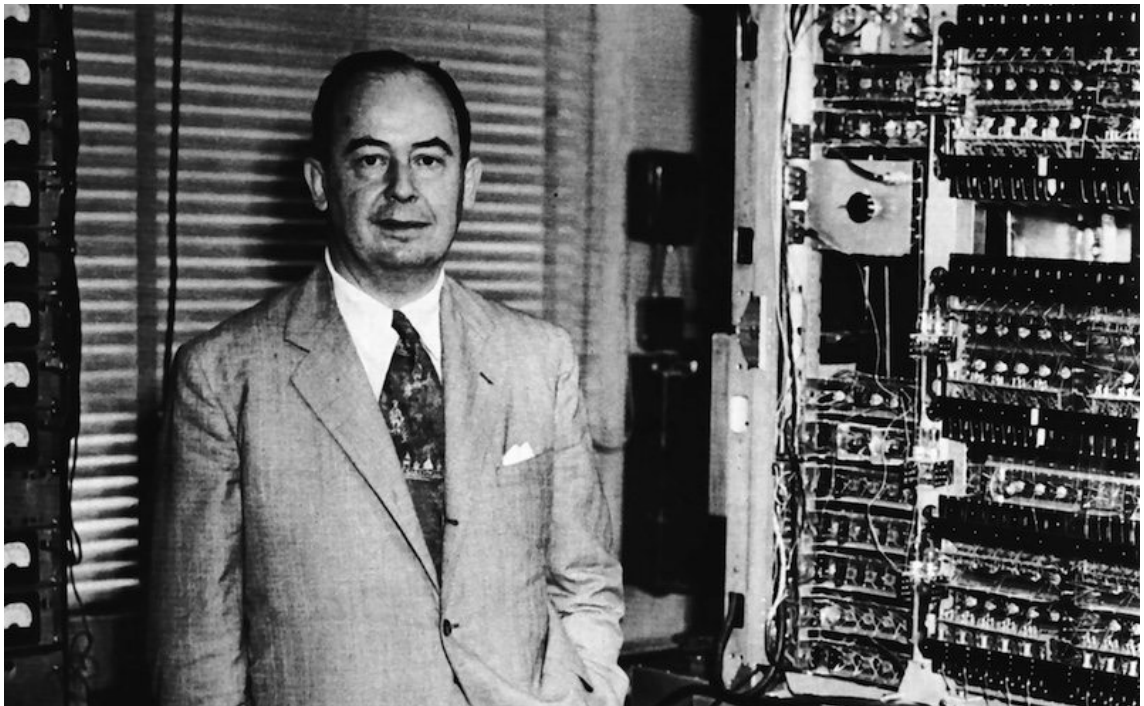
https://www.ias.edu/von-neumann
https://www.atomicarchive.com/resources/biographies/vonNeumann.html

The Von Neumann Model/Architecture

- Recommended reading – קריאה מומלצת (רשות)
  - Burks, Goldstein, von Neumann, Preliminary discussion of the logical design of an electronic computing instrument, 1946.
  - Patt and Patel book, Chapter 4, "The von Neumann Model"

- **Stored program**

- **Sequential instruction processing**



Chapter 4

Preliminary discussion of the logical design of an electronic computing instrument[1]

Arthur W. Burks / Herman H. Goldstine / John von Neumann

PART I

1. Principal components of the machine

1.1. Inasmuch as the completed device will be a general-purpose computing machine it should contain certain main organs relating to arithmetic, memory-storage, control and connection with the human operator. It is intended that the machine be fully automatic in character, i.e. independent of the human operator after the computation starts. A fuller discussion of the implications of this remark will be given in Sec. 3 below.

1.2. It is evident that the machine must be capable of storing in some manner not only the digital information needed in a given computation such as boundary values, tables of functions (such as the equation of state of a fluid) and also the intermediate results of the computation (which may be wanted for varying lengths of time), but also the instructions which govern the actual routine to be performed on the numerical data. In a special-purpose machine these instructions are an integral part of the device and constitute a part of its design structure. For an all-purpose machine it must be possible to instruct the device to carry out any computation that can be formulated in numerical terms. Hence there must be some organ capable of storing these program orders. There must, moreover, be a unit which can understand these instructions and order their execution.

1.3. Conceptually we have discussed above two different forms of memory: storage of numbers and storage of orders. If, however, the orders to the machine are reduced to a numerical code and if the machine can in some fashion distinguish a number from an order, the memory organ can be used to store both numbers and orders. The coding of orders into numeric form is discussed in 6.3 below.

1.4. If the memory for orders is merely a storage organ there must exist an organ which can automatically execute the orders stored in the memory. We shall call this organ the *Control*.

1.5. Inasmuch as the device is to be a computing machine there must be an arithmetic organ in it which can perform certain of the elementary arithmetic operations. There will be, therefore, a unit capable of adding, subtracting, multiplying and dividing. It will be seen in 6.6 below that it can also perform additional operations that occur quite frequently.

The operations that the machine will view as elementary are clearly those which are wired into the machine. To illustrate, the operation of multiplication could be eliminated from the device as an elementary process if one were willing to view it as a properly ordered series of additions. Similar remarks apply to division. In general, the inner economy of the arithmetic unit is determined by a compromise between the desire for speed of operation—a non-elementary operation will generally take a long time to perform since it is constituted of a series of orders given by the control—and the desire for simplicity, or cheapness, of the machine.

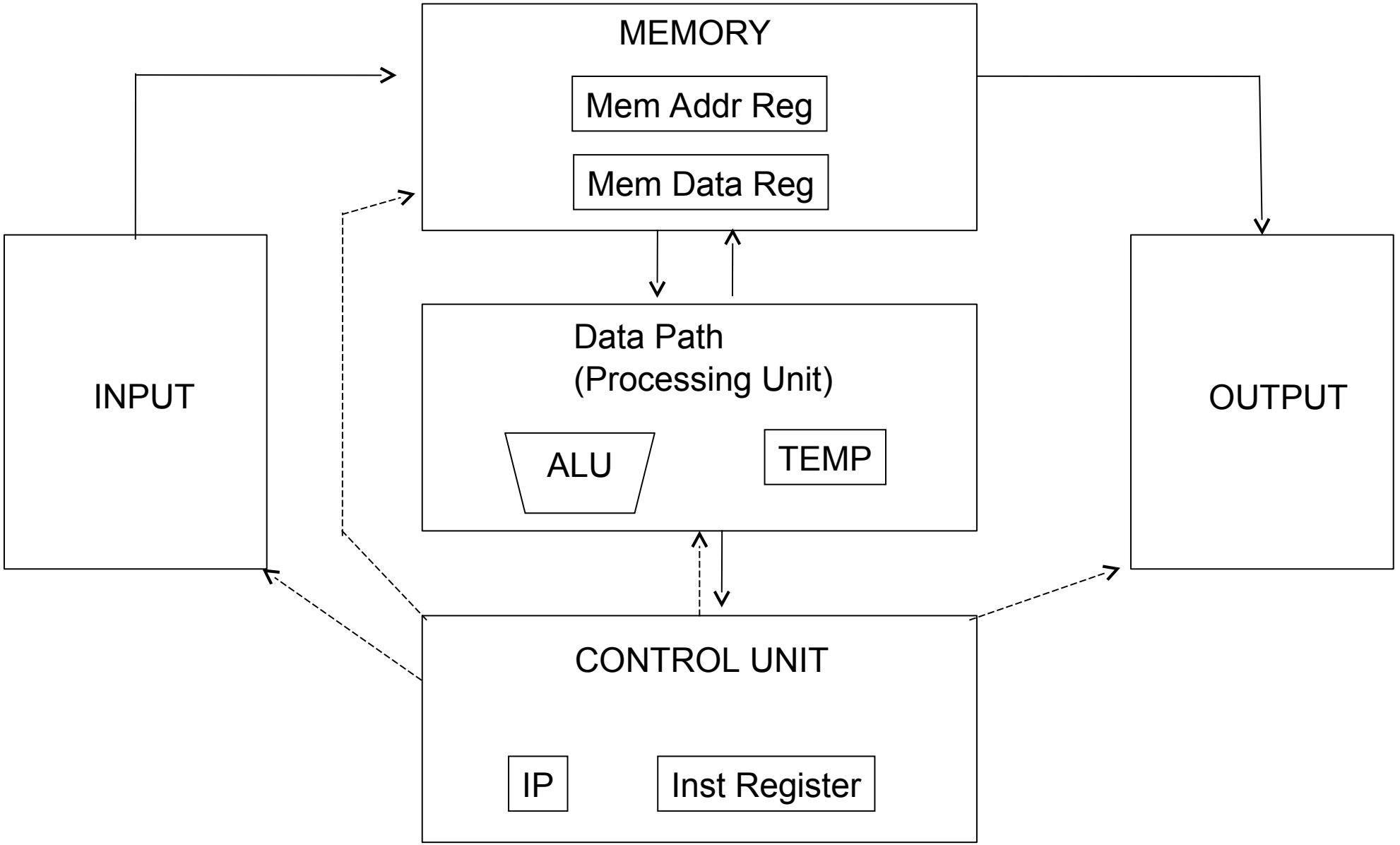
1.6. Lastly there must exist devices, the input and output organ, whereby the human operator and the machine can communicate with each other. This organ will be seen below in 4.5, where it is discussed, to constitute a secondary form of automatic memory.

2. First remarks on the memory

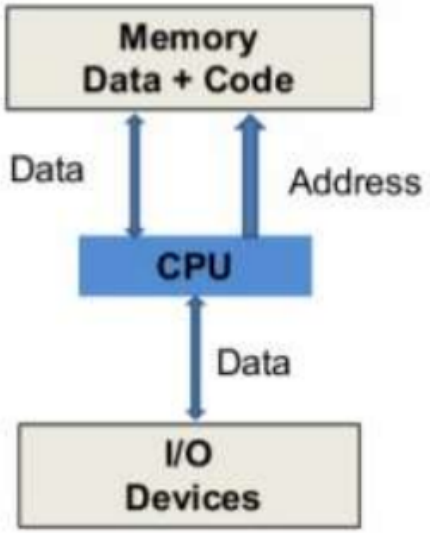
2.1. It is clear that the size of the memory is a critical consideration in the design of a satisfactory general-purpose computing

[1] From A. H. Taub (ed.), "Collected Works of John von Neumann," vol. 5, pp. 34–79, The Macmillan Company, New York, 1963. Taken from report to U. S. Army Ordnance Department, 1946. See also Bibliography Burks, Goldstine and von Neumann, 1962*a*, 1962*b*, 1963; and Goldstine and von Neumann 1963*a*, 1963*b*, 1963*c*, 1963*d*.

The Von Neumann Model (of a Computer)

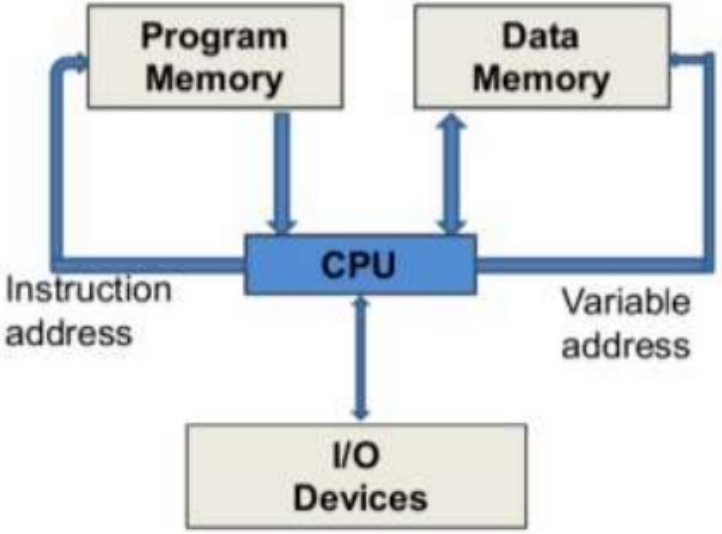


von Neumann vs. Harvard Architecture



von Neumann architecture
(unified memory for code & data)

Programs can be modified like data
More efficient use of memory space



Harvard architecture
(separate memories for code & data)

Better protection of programs
Higher aggregate memory bandwidth
Memory optimization for access type

Credit: Behrooz Parhami, University of California, Santa Barbar

The Von Neumann Model (of a Computer)

- Q: Is this the only way that a computer can operate?

- A: No.
- Qualified Answer: But, it has been the dominant way
  - i.e., the dominant paradigm for computing
  - for N decades

  - Multi Processor isn't Von Neumann model!

The Dataflow Model (of a Computer)

מודל אלטרנטיבי

- **Von Neumann model**: An instruction is fetched and executed in control flow order
  - As specified by the instruction pointer
  - Sequential unless explicit control flow instruction

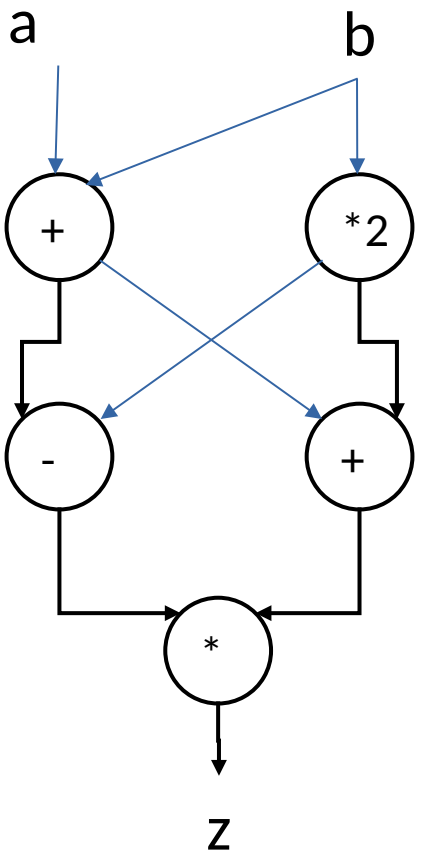
- **Dataflow model**: An instruction is fetched and executed in data flow order
  - i.e., when its operands are ready
  - i.e., there is **no instruction pointer**
  - Instruction ordering specified by **data flow dependence**
    - Each instruction specifies "who" should receive the result
    - An instruction can "fire" whenever all operands are received
  - Potentially many instructions can execute at the same time
    - **Inherently more parallel**

Von Neumann vs Dataflow

- Consider a Von Neumann program
  - What is the significance of the program order?
  - What is the significance of the storage locations?

```
v ← a + b;
w ← b * 2;
x ← v - w
y ← v + w
z ← x * y
```

Sequential



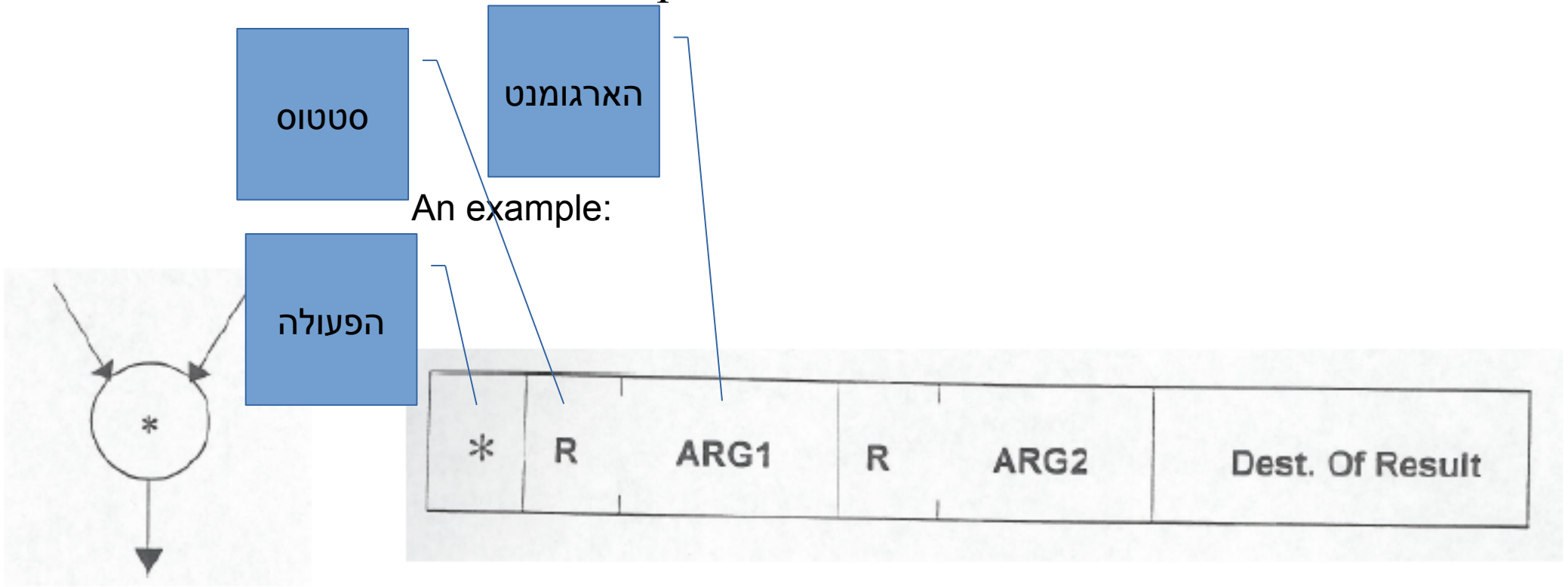
Dataflow

- Which model is more natural to you as a programmer?

More on Data Flow

- In a data flow machine, a program consists of data flow nodes
  - A data flow node fires (fetched and executed) when all it inputs are ready
    - i.e. when all inputs have tokens

- Data flow node and its ISA representation



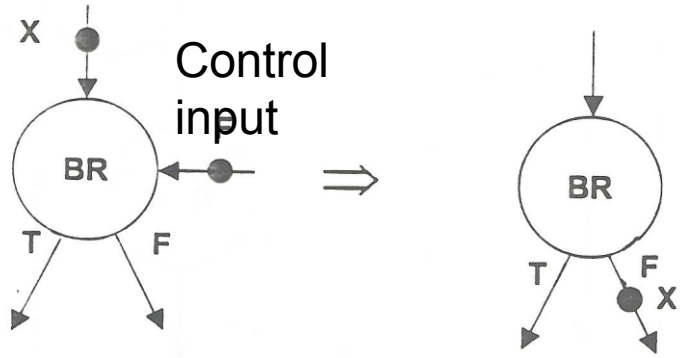
R=Ready bit

Data Flow Nodes

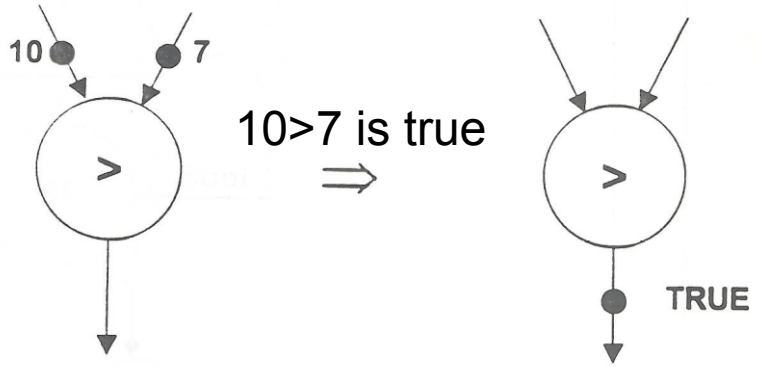
אפשר לבנות תחביר למערכת זו

*Conditional

Branch



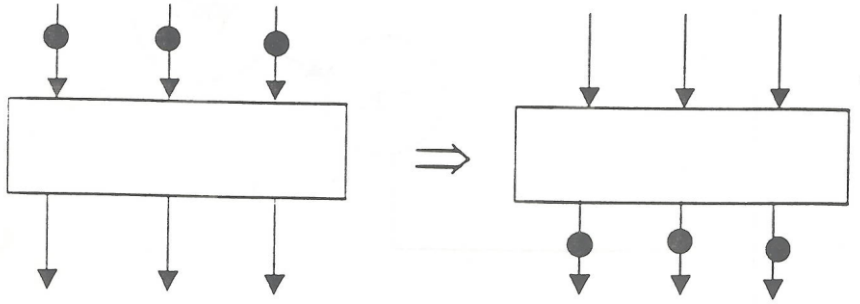
*Relational



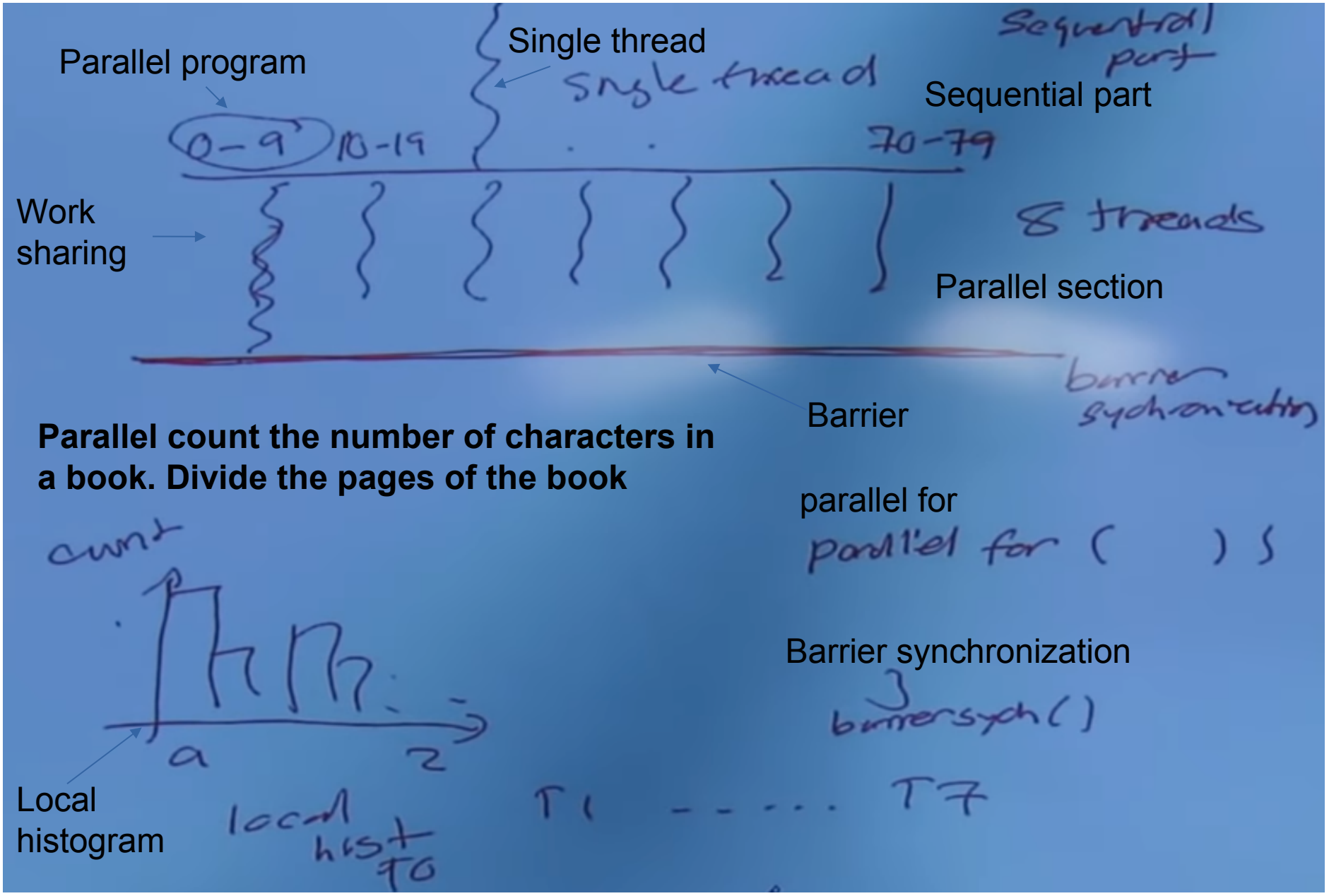
הרחבה בשקף הבא

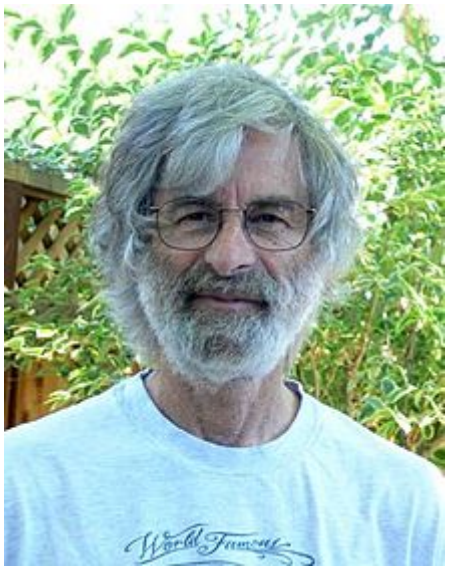
*Barrier Synch

(מחשב מקבילי)



Barrier sync. example





https://en.wikipedia.org/wiki/Leslie_Lamport

winner of the 2013 Turing Award

Operating Systems

R. Stockton Gaines Editor

Time, Clocks, and the Ordering of Events in a Distributed System

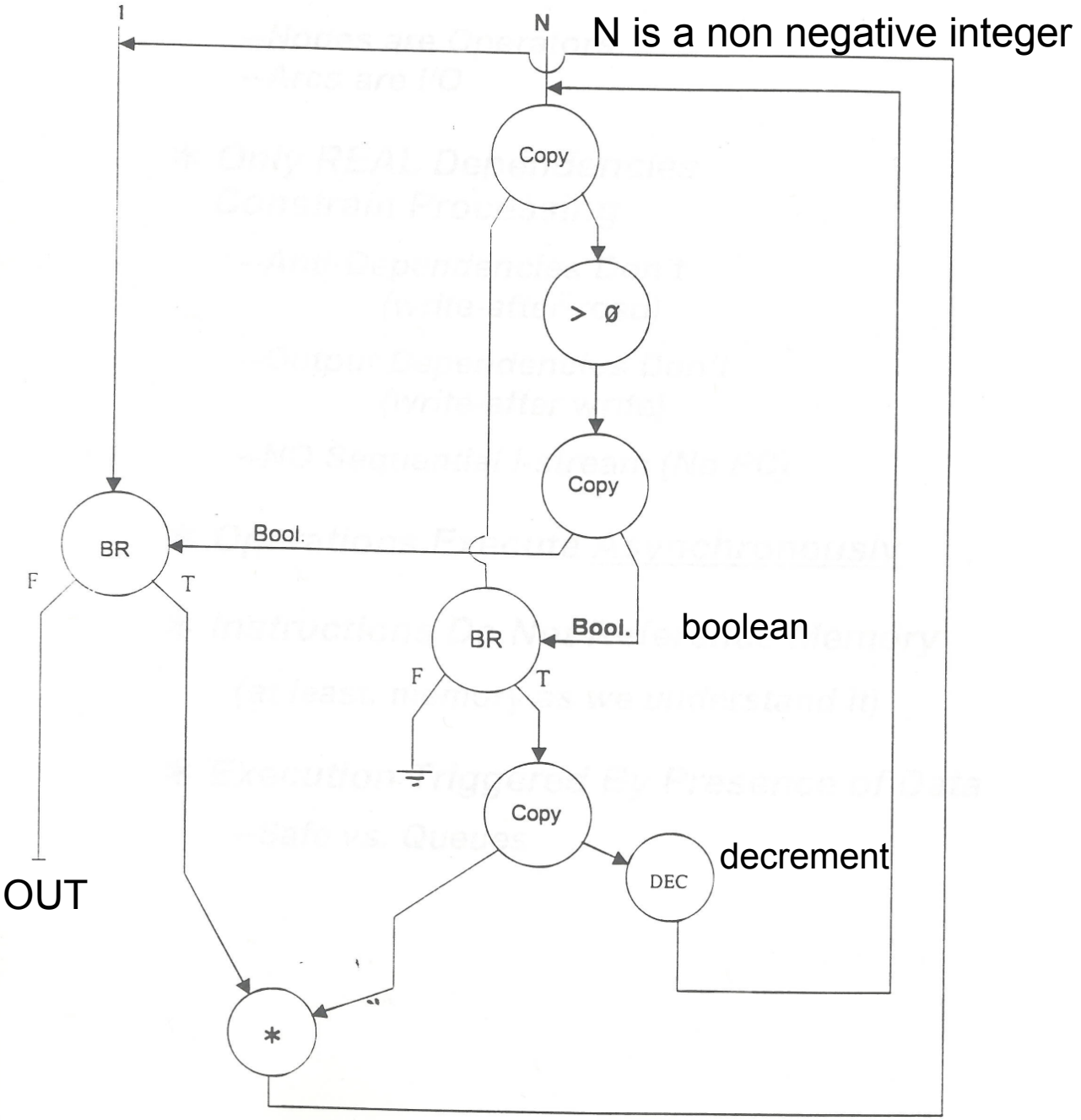
Leslie Lamport
Massachusetts Computer Associates, Inc.

Communications of the ACM

July 1978
Volume 21
Number 7

קריאת רשות

An Example Data Flow Program



ISA-level Tradeoff: Instruction Pointer

- Do we need an instruction pointer in the ISA?
  - **Yes**: Control-driven, sequential execution
    - An instruction is executed when the IP points to it
    - IP automatically changes sequentially (except for control flow instructions)
  - **No**: Data-driven, parallel execution
    - An instruction is executed when all its operand values are available (data flow)

- Tradeoffs: MANY high-level ones
  - Ease of programming (for average programmers)?
    - קל יותר לדבאג תכנית סדרתית
  - Ease of compilation?
  - Performance: Extraction of parallelism?
  - Hardware complexity?

מה ההשלכות של הבחירה שלנו?

ISA vs. Microarchitecture Level Tradeoff

- A similar tradeoff (control vs. data-driven execution) can be made at the microarchitecture level
- ISA: Specifies how the programmer sees instructions to be executed
  - Programmer sees a **sequential**, control-flow execution order vs.
  - Programmer sees a **data-flow** execution order

- Microarchitecture: How the underlying implementation actually executes instructions
  - **Microarchitecture** can execute instructions in any order as long as it obeys the semantics specified by the ISA when making the instruction results visible to software
    - Programmer should see the order specified by the ISA

Most of the ISAs available today are sequential!!!

The Von-Neumann Model

- All major *instruction set architectures* today use this model
  - x86, ARM, MIPS, SPARC, Alpha, POWER

Underneath (at the microarchitecture level), the execution model of almost all *implementations (or, microarchitectures)* is very different

- Pipelined instruction execution: *Intel 80486 uarch*
- Multiple instructions at a time: *Intel Pentium uarch*
- Out-of-order execution: *Intel Pentium Pro uarch*
- Separate instruction and data caches

- But, what happens underneath that is *not* consistent with the von Neumann model is *not* exposed to software
  - Difference between ISA and microarchitecture

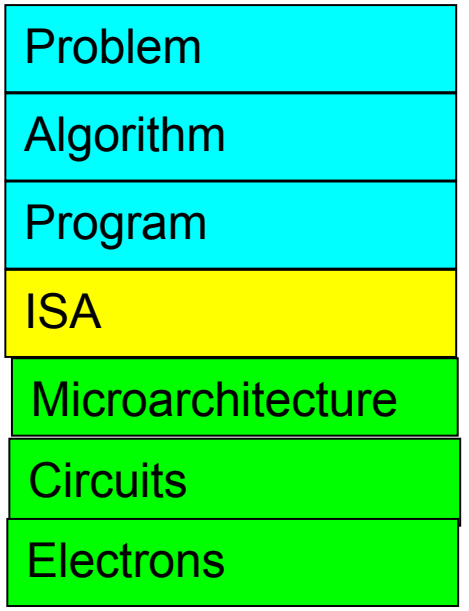
What is Computer Architecture?

- **ISA+implementation definition:** The science and art of designing, selecting, and interconnecting hardware components and designing the hardware/software interface to create a computing system that meets functional, performance, energy consumption, cost, and other specific goals.
- **Traditional (ISA-only) definition:** "The term *architecture* is used here to describe the attributes of a system as seen by the programmer, i.e., the conceptual structure and functional behavior as distinct from the organization of the dataflow and controls, the logic design, and the physical implementation." *Gene Amdahl*, IBM Journal of R&D, April 1964

ISA vs. Microarchitecture

- ISA
  - Agreed upon interface between software and hardware
    - SW/compiler assumes, HW promises
  - What the software writer needs to know to write and debug system/user programs
- Microarchitecture
  - Specific **implementation** of an ISA
  - Not visible to the software
- Microprocessor
  - **ISA, uarch**, circuits "Architecture" =ISA+microarchitecture

u = μ



ISA vs. Microarchitecture

- What is part of ISA vs. Uarch?
  - Gas pedal: interface for "acceleration"
  - Internals of the engine: implement "acceleration"

- Implementation (uarch) can be various as long as it satisfies the specification (ISA)
  - **Add instruction** vs. **Adder implementation**
    - Bit serial, ripple carry, carry lookahead adders are all part of microarchitecture
  - **x86** ISA has many implementations: 286, 386, 486, Pentium, Pentium Pro, Pentium 4, Core, …

- Microarchitecture usually changes faster than ISA
  - Few ISAs (x86, ARM, SPARC, MIPS, Alpha) but many uarchs
  - *Why?*

ISA

- Instructions
  - Opcodes, Addressing Modes, Data Types
  - Instruction Types and Formats
  - Registers, Condition Codes
- Memory
  - Address space, Addressability, Alignment
  - Virtual memory management
- Call, Interrupt/Exception Handling
- Access Control, Priority/Privilege
- I/O: memory-mapped vs. instr.
- Task/thread Management
- Power and Thermal Management
- Multi-threading support, Multiprocessor support

Intel® 64 and IA-32 Architectures
Software Developer's Manual

Volume 1:
Basic Architecture

Microarchitecture

- Implementation of the ISA under specific design constraints and goals
- Anything done in hardware without exposure to software
  - Pipelining
  - In-order versus out-of-order instruction execution
  - Memory access scheduling policy
  - Speculative execution
  - Superscalar processing (multiple instruction issue?)
  - Clock gating
  - Caching? Levels, size, associativity, replacement policy
  - Prefetching?
  - Voltage/frequency scaling?
  - Error correction?

קריאת חובה:

A New Golden Age for Computer Architecture

By
JOHN L. HENNESSY and
DAVID A. PATTERSON

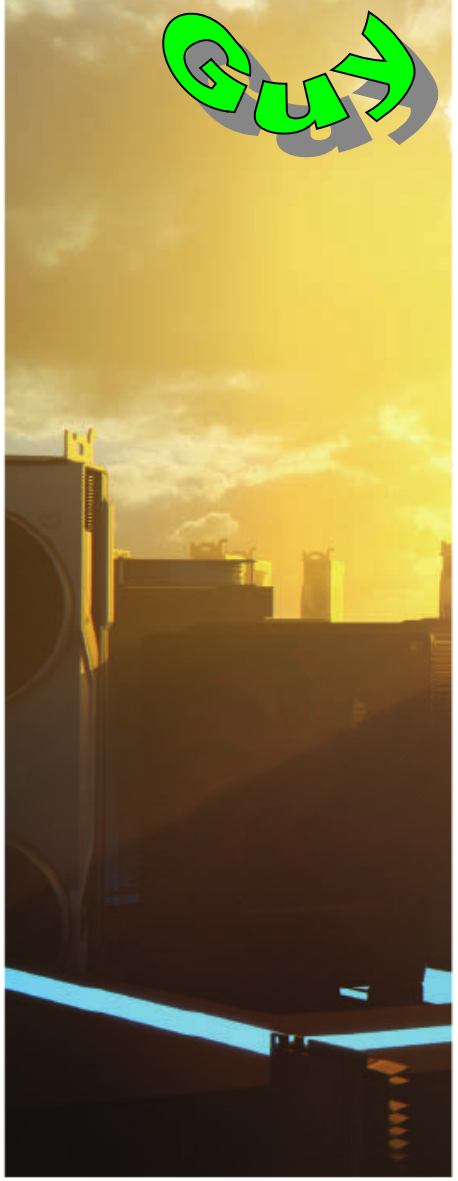
המאמר נמצא באתר המודל

DOI:10.1145/3282307

Innovations like domain-specific hardware, enhanced security, open instruction sets, and agile chip development will lead the way.

BY JOHN L. HENNESSY AND DAVID A. PATTERSON

A New Golden Age for Computer Architecture



WE BEGAN OUR Turing Lecture June 4, 2018[11] with a review of computer architecture since the 1960s. In addition to that review, here, we highlight current challenges and identify future opportunities, projecting another golden age for the field of computer architecture in the next decade, much like the 1980s when we did the research that led to our award, delivering gains in cost, energy, and security, as well as performance.

"Those who cannot remember the past are condemned to repeat it." —George Santayana, 1905

Software talks to hardware through a vocabulary called an instruction set architecture (ISA). By the early 1960s, IBM had four incompatible lines of computers, each with its own ISA, software stack, I/O system, and market niche—targeting small business, large business, scientific, and real time, respectively. IBM engineers, including ACM A.M. Turing Award laureate Fred Brooks, Jr., thought they could create a single ISA that would efficiently unify all four of these ISA bases.

They needed a technical solution for how computers as inexpensive as

» key insights

- Software advances can inspire architecture innovation.
- Elevating the hardware/software interface creates opportunities for architecture innovation.
- The marketplace ultimately settles architecture debates.

עד כאן מצגת זו