**Chapter 01**

Table of Contents

[1.1. Organization and Architecture 2](#_Toc48979506)

[1.2. Structure and Function 3](#_Toc48979507)

[1.3. A Brief () History of Computers 12](#_Toc48979508)

[First Generation 12](#_Toc48979509)

[Second Generation Computers 18](#_Toc48979510)

[Third Generation Computers 19](#_Toc48979511)

[Later Generations 22](#_Toc48979512)

[1.4 RISC and CISC Architecture Processors 23](#_Toc48979513)

[1.5 Embedded Systems 24](#_Toc48979514)

[The Internet of Things 24](#_Toc48979515)

[Application and Dedicated Processors 25](#_Toc48979516)

[Microprocessors and Microcontrollers 25](#_Toc48979517)

[1.7 Cloud Computing 25](#_Toc48979518)

## 1.1. Organization and Architecture

In the previous lecture, we looked at the process through which a computer prints a specific command, print(a+5). We can look at the process in a different manner as well. We can say that the computer takes two memory registers, or D Flip Flops, one containing the value of a and another containing the value 5, and passes these to an adder. The result from the adder is then passed to the display. As you can see, this view on the problem goes through the physical devices that makes the result possible.

Computer Architecture refers to the attributes of a system visible to a programmer, ones that have a direct impact on the logical execution of a program. Computer architecture is also often referred to as Instruction Set Architecture or ISA.

Computer Organization refers to the operational units and interconnections that makes the architectural specifications possible, such as the bits that represent data, input and output mechanisms and memory addressing techniques. This information is mostly invisible to the programmer, since they deal with the architecture part.

For example, it is an architectural design issue whether a computer will have a specific multiply instruction, but it is an organizational design issue whether the multiply instruction will make use of a special multiply unit, or just use the adder unit repeatedly. The organizational decision will be based on how often the instruction is used, and the cost and size of the unit.

The difference in organization and architecture, although usually invisible, is very important. Computer models from the same family usually have improved organizational abilities over the years, thus allowing them to be faster but cheaper, while maintaining the same overall architecture, thus allowing newer computers to still run older programs. In microcomputers, which do not need to have backwards compatibility, the improvements in organization allow more powerful and complex architectures.

## 1.2. Structure and Function

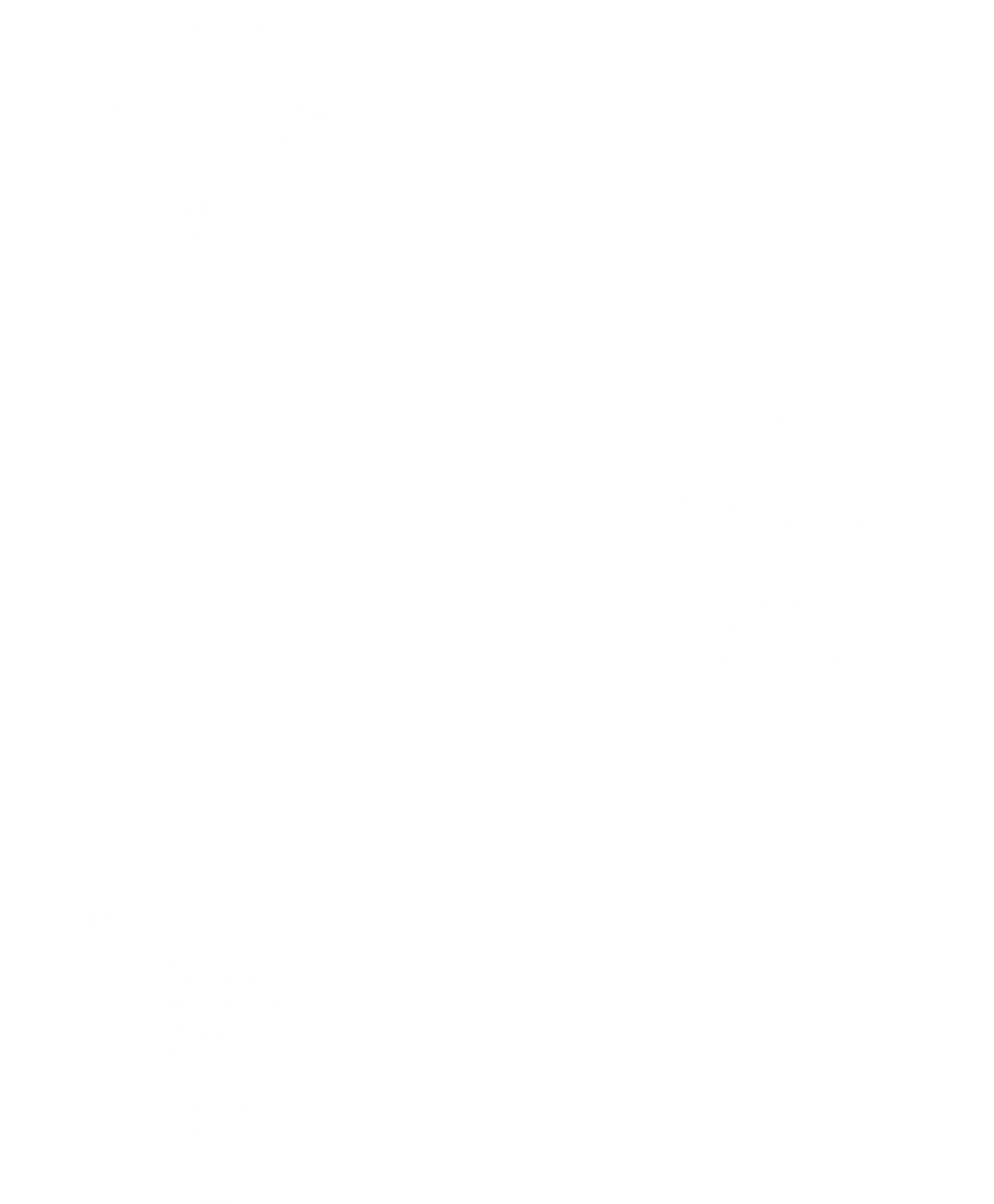
A complex system such as a computer generally has a hierarchical structure. This makes both the design and the description easy, since only a particular level needs to be dealt with at one time. The structure refers to the way in which components are related, and the functions are the operations of each individual component with regard to the structure.

There are two ways to describe a hierarchical structure, top-down and bottom-up. As an example, think of the president of a country trying to determine the country’s population. He/she would inform the people in charge of districts, who would instruct city mayors, who would get employees to go door to door collecting information. This is a top-down structure. This is much easier to dictate and describe than it would be to go door-to-door, instructing people to inform their mayor’s office about how many people live there, then instructing the mayors to inform district officials and so on, a bottom-up approach.

There are generally only four functions a computer can perform:

* Data Processing
* Data Storage - short-term storage used while processing data and long-term storage for the user
* Data Movement - retrieval and delivery of data to and from peripherals
* Control - management of resources and orchestration of functions in response to instructions

We will now look at the structure of a computer, starting with a traditional computer with a single processor with a microprogrammed control unit.



The computer itself is composed of 4 components:

* Central Processing Unit (CPU) – The processor, controls the operations of the computer and performs data processing functions. Recent computers have more than one. It is a physical piece of silicon.
* Main Memory – Long term data storage
* I/O – moves data between computer and outside environment
* System Interconnections – Mechanisms that allow communications between the other three. A common example is the system bus, which consists of a number of conducting wires.

We will be looking at all these components in depth, but for now we will only discuss the CPU. It in turn consists of four parts:

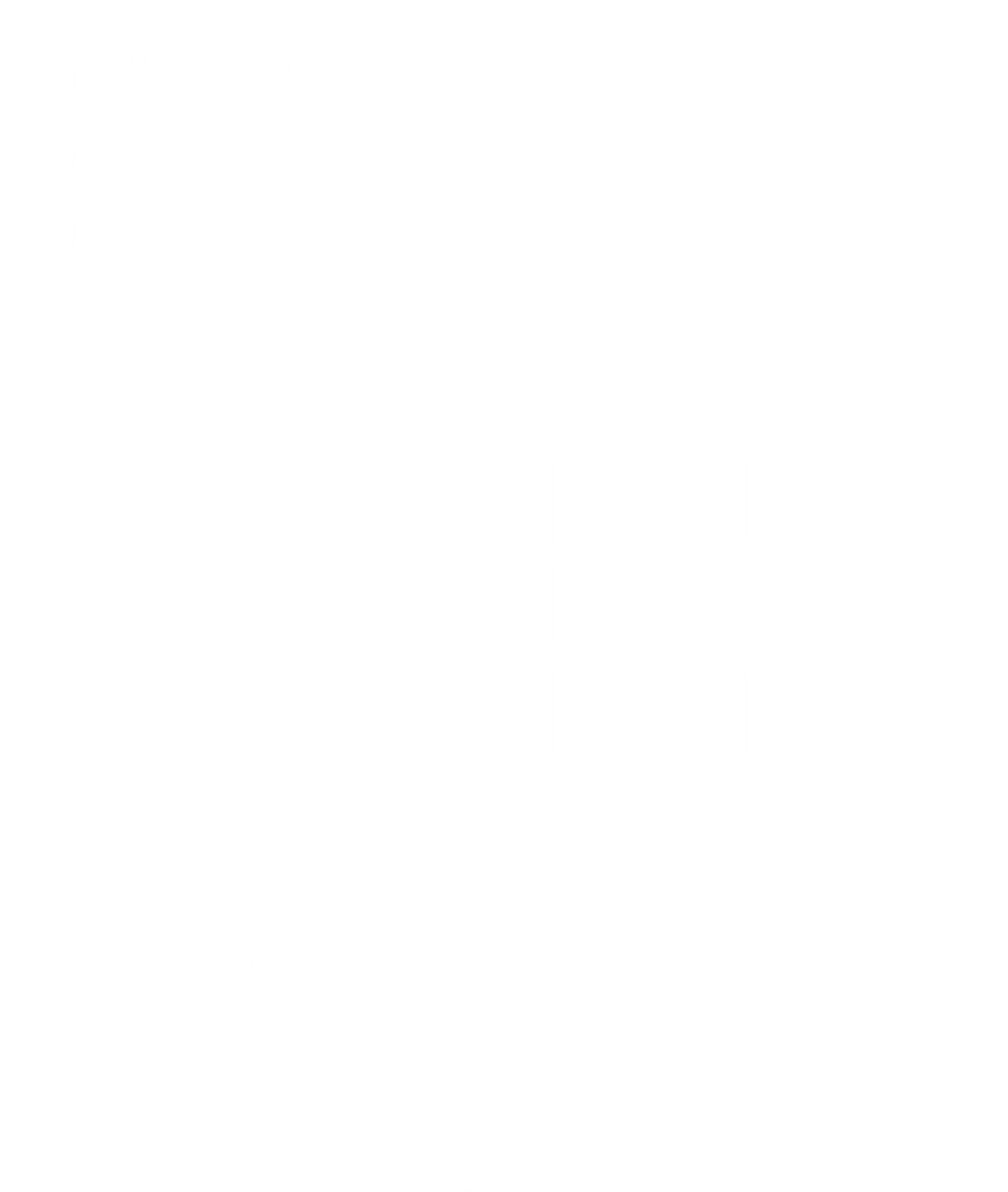
* Control Unit – Controls the CPU, and thus the computer
* Arithmetic and Logic Unit (ALU) – Performs data processing functions
* Registers – Provides internal storage to CPU (used for temporary storage while performing functions)
* CPU Interconnections – Provides communication between the control unit, the ALU and registers.

There are several ways to implement the control unit, one common way being a microprogrammed implementation. Basically, a microprogrammed control unit operates by executing microinstructions that define the functionality of the control unit.

Computers usually have more than one processor, and when these processors are all on a single chip, it is said to be a multicore processor, with each processor being called a core.

We also have cache memory as layers between the processor and the main memory. The cache memory stores data that is likely to be used next. The closer the layer is to the core, the smaller and faster it is.

Most computers are housed on a motherboard. A motherboard is a printed circuit board or PCB, which is a rigid, flat board that holds and interconnects chips and other electronic components. The board is made of layers that interconnect components with copper pathways etched into the board. The main PCB is called the motherboard while smaller ones, that can be plugged into slots in the main one, are called expansion boards. We can now discuss the motherboard and its components.



The main elements on the motherboards are chips, also known as integrated circuits or ICs. These are piece of semiconducting material upon which electronic circuits and logic gates are fabricated.

The motherboard contains slots for:

* Processor Chip – Multicore processors have multiple individual cores
* Memory Chips
* I/O Controller Chips

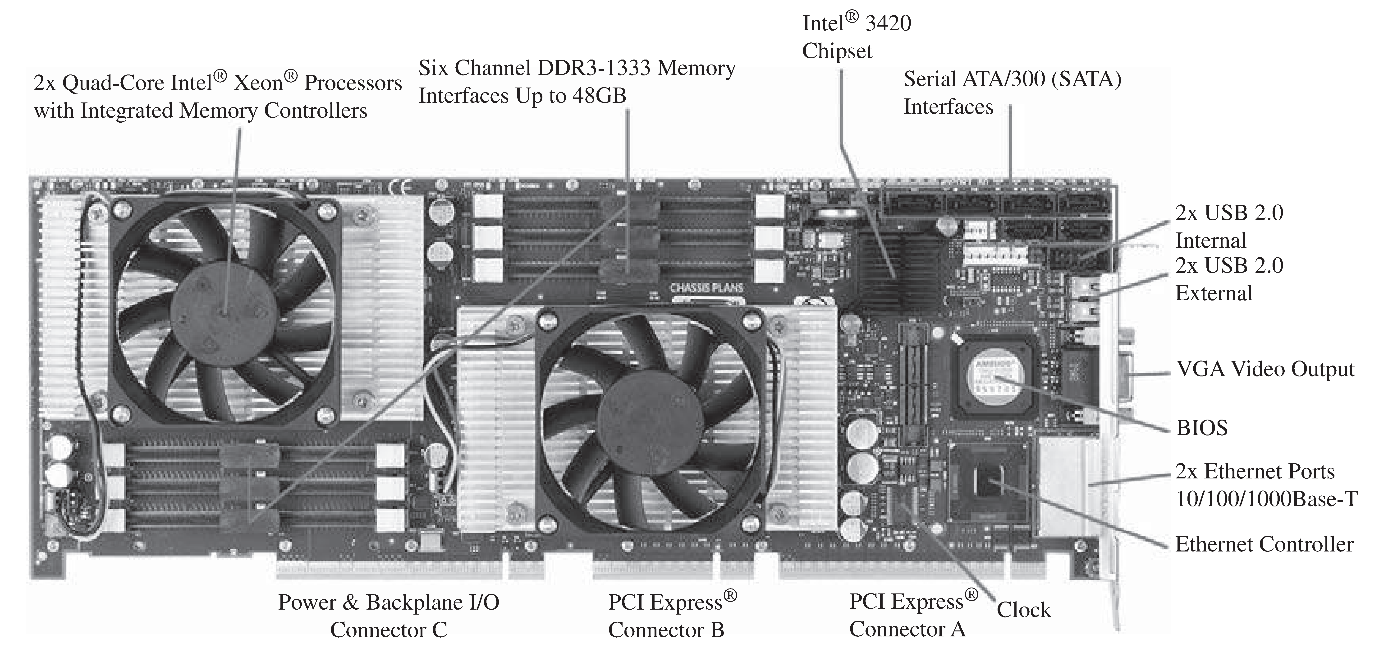
There are also expansion slots that allow the inclusion of expansion boards for more components. Thus, the modern computer connects a few individual chip components with each chip containing a few thousand to hundreds of millions of transistors.

Next, we look at the processor chip. The one shown has 8 cores and an L3 cache. Some logic is required to control operations between the cores and the cache and between the cores and the external circuitry of the motherboard. Unlike the figure, the cores in the processor all have access to the entirety of the L3 cache via the control circuits.

A single core contains:

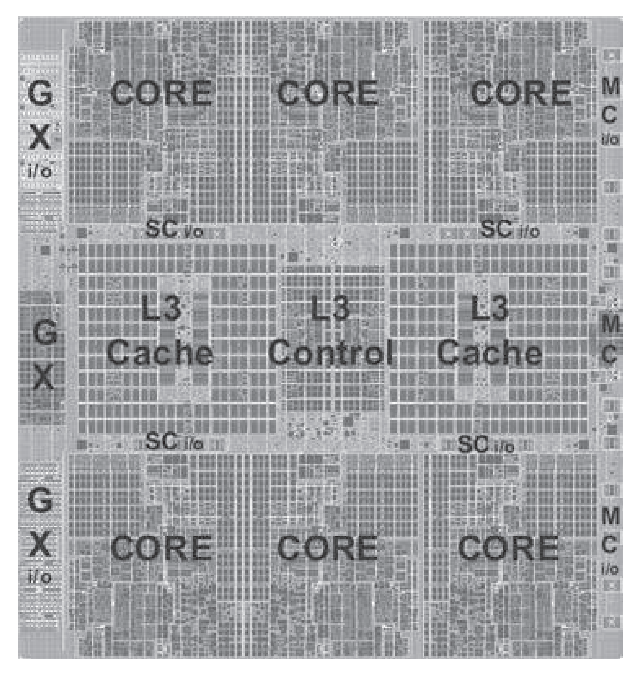
* Instruction Logic – It fetches instructions and decodes each to determine the instruction operation and the memory locations of any operands
* ALU – Performs the operation specified by an instruction
* Load/Store Logic – Manages the transfer of data to and from the main memory via cache
* L1 Cache – Has two parts, the instruction cache that transfers instructions to and from the main memory, and the data cache for the transfer of operands and results
* L2 Cache – Same function as L1 Cache; present in modern computers; may be combined

The following is the image of an actual motherboard with two processors:



* PCI Express Slots – For high-end display adapters and additional peripherals; will be discussed at length later
* Ethernet Controller and Ports – For network connections
* USB Sockets – For peripheral devices
* Serial ATA (SATA) Sockets – For connection to disk memory
* Memory Interfaces – For DDR (Double Data Rate) main memory chips
* Intel 3420 Chipset – An I/O controller for direct memory access operations between peripherals devices and main memory

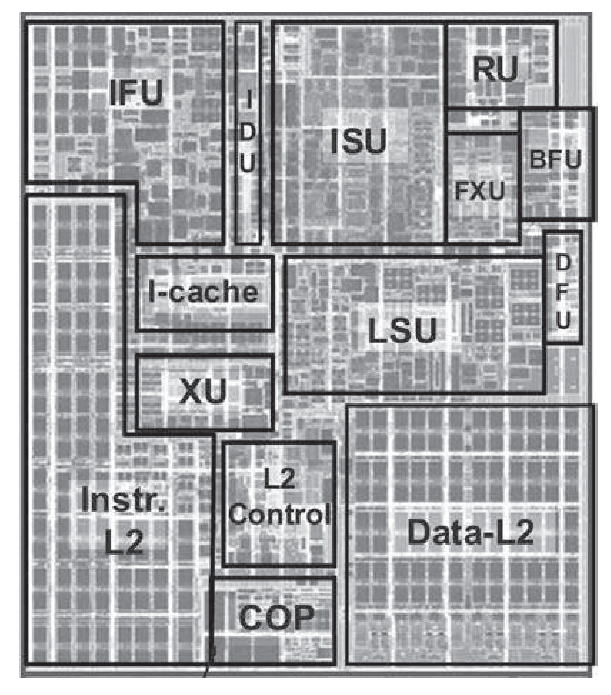
Next, we look at the processor chip:



This look almost exactly like the diagram we saw earlier. Additionally, we have:

* L3 Control Logic - Controls traffic between the L3 cache and the cores and the L3 cache and the external environment.
* Storage Control (SC) – Logic between cores and the L3 cache
* Memory Control (MC) – Controls access to memory outside the chip
* GX I/O Bus – Controls interface to the channel adapters accessing the I/O

And finally, we have the core:

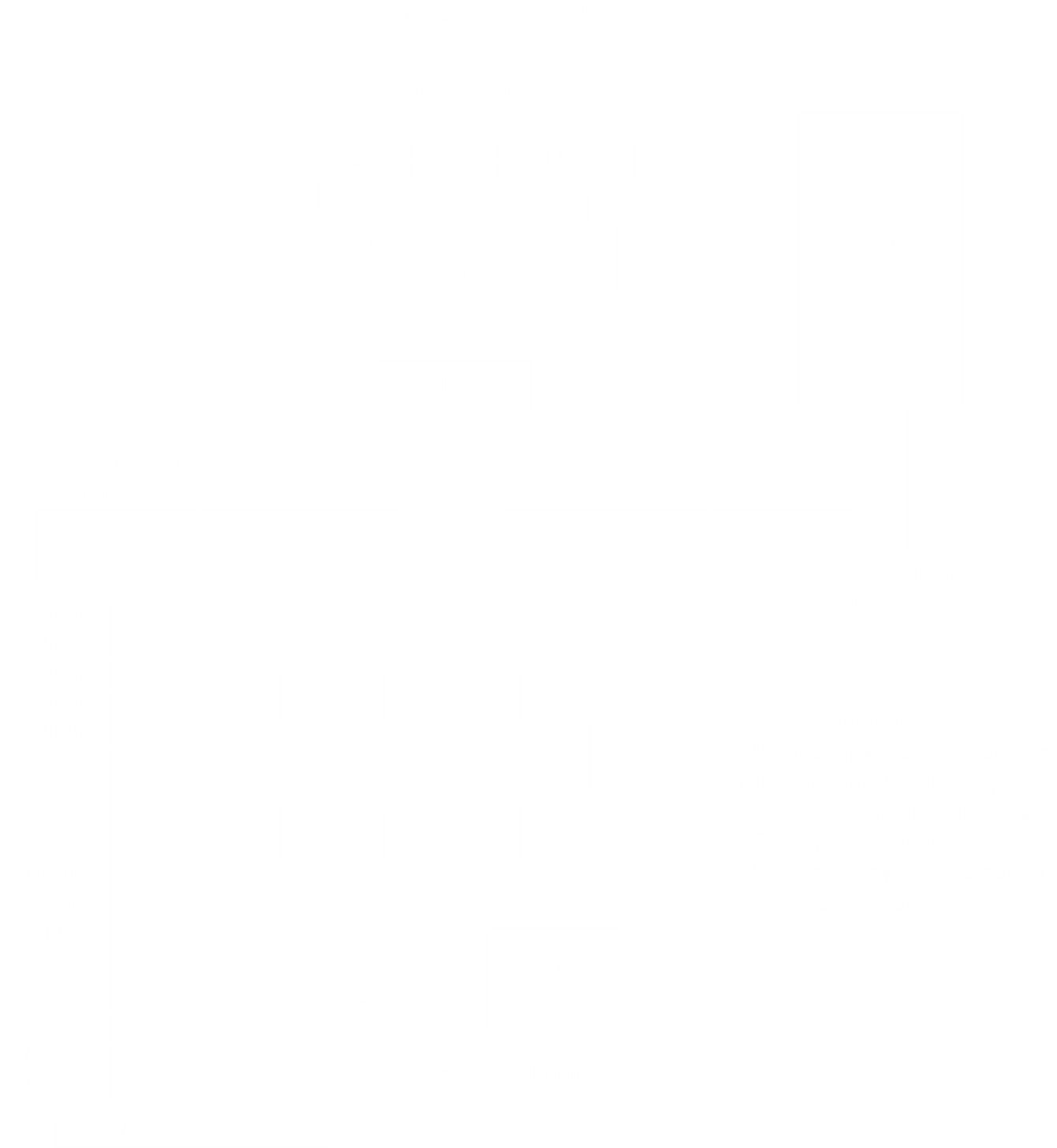


* Instruction Sequence Unit (ISU) – Determines the sequence in which instructions are executed in what is referred to as a superscalar architecture
* Instruction Fetch Unit (IFU) – Logic for fetching instructions
* Instruction Decode Unit (IDU) – Fed from the IFU buffers; responsible for parsing and decoding all z/Architecture operation codes
* Load-Store Unit (LSU) – Contains 96 KB L1 data cache; manages data traffic between L2 data cache and functional execution units; responsible for handling all types of operand accesses of all lengths, modes and formats as defined in z/Architecture.
* Translation Unit (XU) – Translates logical addresses from instructions into physical addresses in the main memory; contains Translation Lookaside Buffer (TLB) to speed up memory access
* Fixed Point Unit (FXU) – Executes fixed point arithmetic operations
* Binary Floating-Point Unit (BFU) – Handles all binary and hexadecimal floating-point operations and fixed-point multiplication operations
* Decimal Floating-Point Unit (DFU) – Handles fixed-point and floating-point operations on decimal numbers
* Recovery Unit (RU) – Keeps a copy of the complete state of the system including registers; collects hardware fault signals; manages hardware recovery actions
* Dedicated Co-Processor (COP) – Responsible for data compression and encryption functions
* I-Cache – 64 KB L1 instruction cache allowing IFU to prefetch instructions before they are needed
* L2 Control – Control logic managing traffic between L2 caches
* Data L2 – 1 MB L2 data cache for all memory traffic other than instructions
* Instr L2 – 1 MB L2 instruction cache

## 1.3. A Brief (🤣) History of Computers

### First Generation

The first generation of computers used vacuum tubes for digital logic elements and memory. We will be looking at the most famous one, the IAS computer. A fundamental design approach first implemented in the IAS computer is the stored program concept. The idea is attributed to John von Neumann, who first published it in 1945 as the Electronic Discrete Variable Computer (EDVAC). He began designing it with colleagues in 1946 at the Princeton Institute for Advanced Studies and completed the work in 1952.



The IAS computer consists of:

* Main Memory – Stores both data and instructions
* ALU – Capable of operating on binary data
* Control Unit – Interprets instructions in memory and causes them to be executed
* I/O Equipment – Operated by control unit

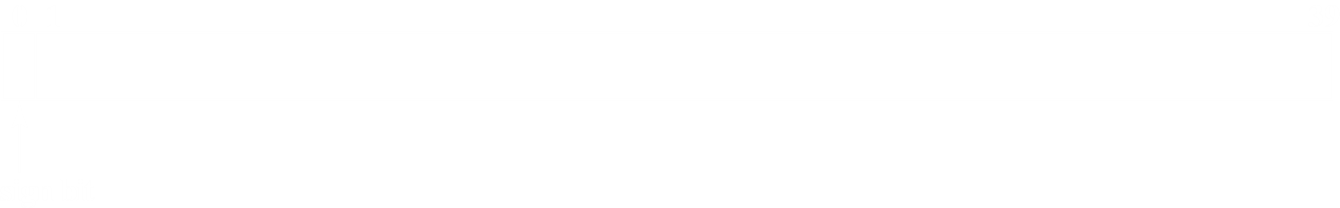
Neumann made a few key notes about this general structure:

* Since this was primarily a computer, it would need a special part to perform arithmetic operations. He called this the Central Arithmetic (CA).
* He recognized that the device required a unit that would carry out instructions; he called it the control centre (CC).
* He knew that complicated sequences of instructions would need considerable memory (M)
* He compared the CC, CA and M to the associative neurons in the human body, and input and output devices to the sensory and motor neurons. He also said that the sensory and motor contact would need to be maintained with an outside recording medium of the device (R).
* He noted that a part would be needed to transfer information from R to C or M as input (I). He also said it would be best to transfer information to M and not directly to C.
* He noted that parts would be needed to transfer information from C and M into R as output (O) and again said it would be best to transfer from M and not directly from C.

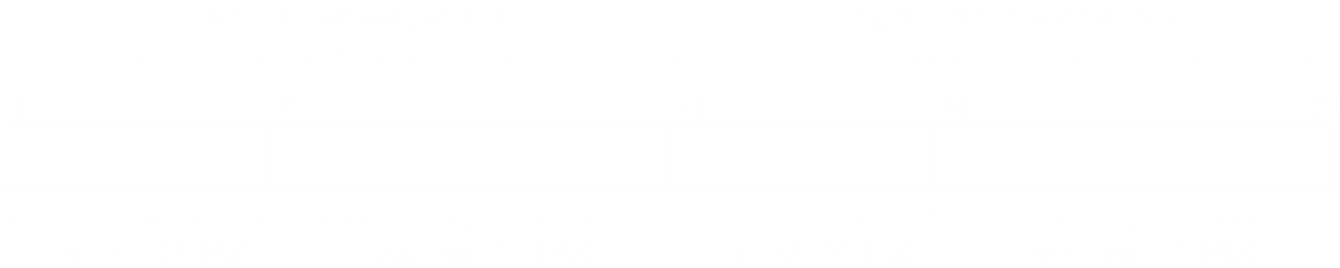
All of today’s computers follow this general structure and are known as von Neumann machines

The memory of the IAS consisted of 4096 storage locations, or words, of 40 bits each that stored both instructions and data. Numbers are represented in binary form and each instruction is a binary code.

Each number is a sign bit and a 39-bit value. This is called a number word.



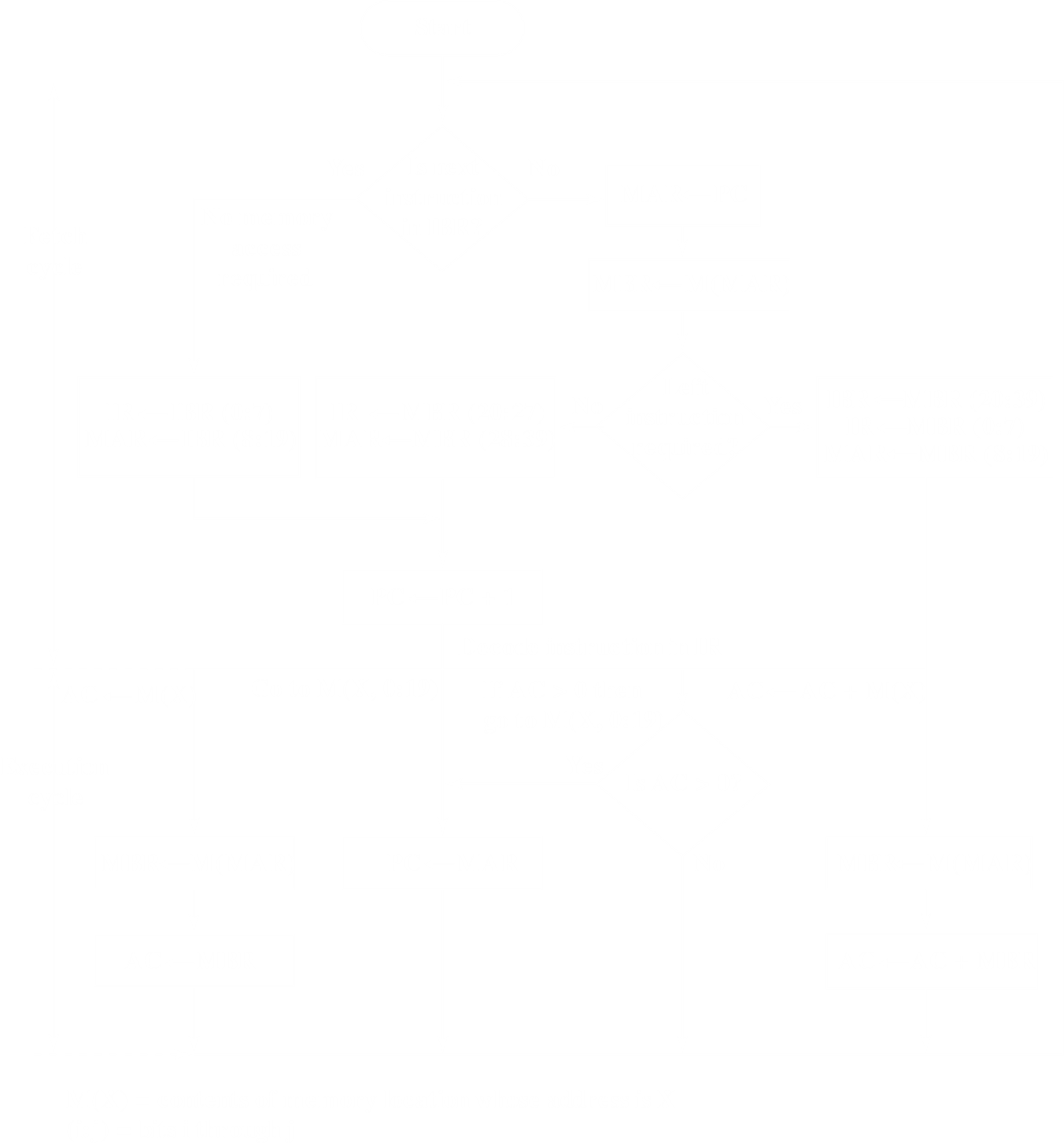
Alternatively, a word may contain 2 20-bit instructions, each with an 8-bit operation code (opcode) specifying the operation to be performed, and a 12-bit address designating one of the words in memory.



The control unit operates the IAS by fetching instructions from memory and executing them one by one. The [diagram for the IAS](#ias_diagram) given earlier shows that both the control unit and the ALU contain storage locations called registers.

* Memory Buffer Register (MBR) – Used to receive, store or send a word from memory or the I/O unit
* Memory Address Register (MAR) – Specifies the address in memory of the word to be written from or read into the MBR
* Instruction Register (IR) – Contains the 8-bit opcode instruction being executed
* Instruction Buffer Register (IBR) – Employed to temporarily hold the right-hand instruction from a word in memory
* Program Counter (PC) – Contains the address of the next instruction pair to be fetched from memory
* Accumulator (AC) and Multiplier Quotient (MQ) – Employed to temporarily hold operands and results of ALU operations. Multiplying 2 40-bit numbers, which gives an 80-bit number, will cause the most significant 40-bits to be stored in AC and the least significant 40-bits to be stored in MQ

The IAS operates by repetitively performing the instruction cycle shown below:



Each instruction cycle consists of two subcyles. In the fetch cycle, the opcode of the next instruction is loaded into the IR and the address part is loaded into the MAR. The instruction may come from the IBR, or directly from the memory by loading a word into the MBR and then to the IBR, IR and MAR. The indirection is caused due to the fact that there is only one register used to specify the address in memory for a read or write and only one register for the source or destination. This was done to simplify the electronics.

Once the opcode is in the IR, the execute cycle begins. Control circuitry interprets the opcode and executes the instruction by sending out appropriate control signals, making the data move or the operation be performed by the ALU.

The IAS computer has 21 instructions, listed below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction Type** | **Opcode** | **Symbolic Representation** | **Description** |
| Data Transfer | 00001010 | LOAD MQ | Transfer contents of register MQ to the accumulator AC |
| 00001001 | LOAD MQ, M(X) | Transfer contents of memory location X to MQ |
| 00100001 | STOR M(X) | Transfer contents of accumulator to memory location X |
| 00000001 | LOAD M(X) | Transfer M(X) to the accumulator |
| 00000010 | LOAD -M(X) | Transfer -M(X) to the accumulator |
| 00000011 | LOAD |M(X)| | Transfer absolute value of M(X) to the accumulator |
| 00000100 | LOAD -|M(X)| | Transfer -|M(X)| to the accumulator |
| Unconditional Branch | 00001101 | JUMP M(X, 0:19) | Take next instruction from left half of M(X) |
| 00001110 | JUMP M(X, 20:39) | Take next instruction from right half of M(X) |
| Conditional Branch | 00001111 | JUMP + M(X, 0:19) | If number in the accumulator is non-negative, take next instruction from left half of M(X) |
| 00010000 | JUMP + M(X, 20:39) | If number in the accumulator is non-negative, take next instruction from right half of M(X) |
| Arithmetic | 00000101 | ADD M(X) | Add M(X) to AC; put the result in AC |
| 00000111 | ADD |M(X)| | Add |M(X)| to AC; put the result in AC |
| 00000110 | SUB M(X) | Subtract M(X) from AC; put the result in AC |
| 00001000 | SUB |M(X)| | Subtract |M(X)| from AC; put the remainder in AC |
| 00001011 | MUL M(X) | Multiply M(X) by MQ; put most significant bits of result in AC; put least significant bits in MQ |
| 00001100 | DIV M(X) | Divide AC by M(X); put the quotient in MQ and the remainder in AC |
| 00010100 | LSH | Multiply accumulator by 2; that is, shift left one-bit position |
| 00010101 | RSH | Divide accumulator by 2; that is, shift right one-bit position |
| Address Modify | 00010010 | STOR M(X, 8:19) | Replace left address field at M(X) by 12 rightmost bits of AC |
| 00010011 | STOR M(X, 28:39) | Replace right address field at M(X) by 12 rightmost bits of AC |

* Data Transfer – Move data between memory and ALU or between two ALU registers
* Unconditional Branch – Normally, the control unit executes instructions in sequence from memory. The sequence can be changed with these instructions, which allows repetitive operations.
* Conditional Branch – The branch can be made dependent on a condition, allowing decision point.
* Arithmetic – Operations performed by the ALU
* Address Modify – Permits addresses to be computed in the ALU and then inserted into instructions stored in memory, thus allowing the program considerable addressing flexibility.

The table does not contain I/O instructions, and is given in symbolic, easy-to-read form. In binary, each instruction must follow the format of an instruction word. The opcode specifies which of the 21 instructions is to be executed and the address portion specifies which of the 4096 memory locations is to be involved.

### Second Generation Computers

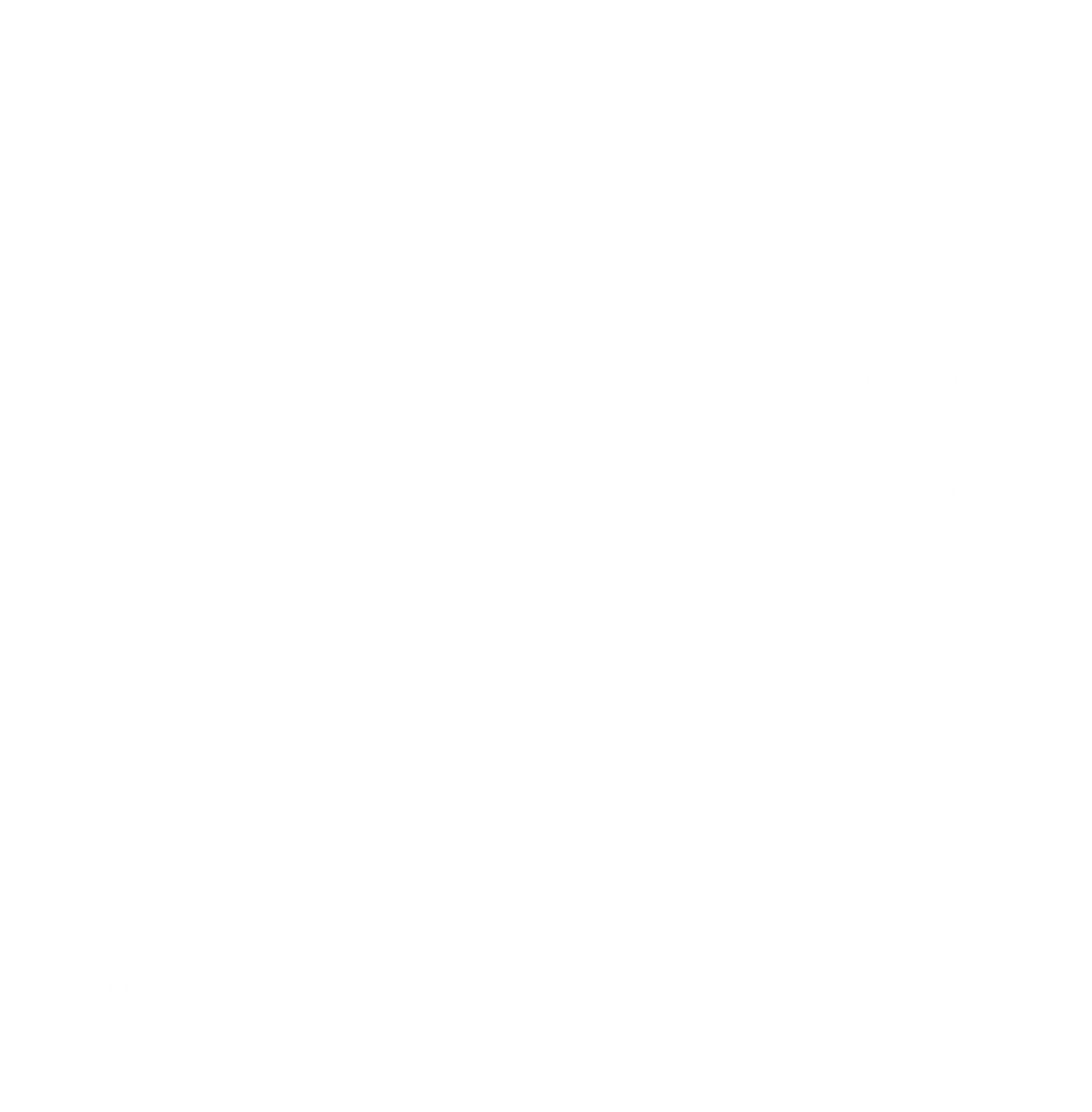
In the second generation, the major change was the replacement of the vacuum tube by the transistor. Transistors are smaller, cheaper or generate less heat. A transistor is also made entirely of silicon, removing the need for wires, metal plates, glass capsules or a vacuum.

The second generation also saw the usage of more complex arithmetic and logic units and control units, high level programming languages and system software. System software provided the ability to load programs, move data and held libraries to common computations, essentially the equivalent of the modern-day operating system.

There was also the Instruction Buffer Register. Two consecutive instructions were loaded at a time for each instruction fetch, with the second one being stored in the IBR. Except for when instructions branched, this reduced the operation time by 50%.

Data channels dealt with I/O operations. These were modules that had their own processors and instruction sets, so the CPU was no longer bothered with detailed I/O instructions. Alongside this was the multiplexor, which acted as the central termination points for data channels, CPU and memory, scheduling memory access.

The IBM 7094 is the perfect example of all these features.



### Third Generation Computers

The third generation saw the introduction of integrated circuits. Before their invention, electronic equipment was made of discrete components, literally individual transistors, resistors and capacitors soldered or wired onto a circuit board. With transistor numbers reaching the hundred of thousands, the manufacturing processing became difficult and expensive. ICs combined all of these things, along with their associated circuitry, into a smaller package, making computers smaller, more reliable and more efficient.

The only things we really need to make a computer are logic gates and memory cells. The four fundamental functions of a computer are covered by these. Data storage is done by the memory cells, data processing by the logic gates, data movement by the circuitry and control by the logic gates as well. The gates and memory cells are in turn made of components like transistors and capacitors.

The integrated circuit takes the fact that each of those components are made of separate pieces of silicon, and turns it into multiple components lying on a single piece of silicon. Connections are made through a process called metallization. A single wafer of silicon is divided into a matrix of tiny squares, each having the same identical circuit pattern, called chips. Each chip has many memory cells and/or gates plus input and output attachments. The chips are packaged in a housing for protection, with pins for attachment to other devices. Multiple packages are interconnected on a printed circuit board (PCB) to produce large and complex circuits.

The number of gates per chip has been growing. We began with Small Scale Integration, at 3 to 30 gates per chip. Medium Scale and Large-Scale Integrations each increased this number 10-fold. By Very Large-Scale Integration, we had reached numbers greater than 3000, and pushing for Ultra Large-Scale Integration, with even greater numbers. The growth follows Moore’s Law, which states the number of transistors in a chip doubles every year while costs halve. This has held true for a long time, but recently the time has increased to 18 months.

The consequences of this growth are that costs have fallen dramatically, operating speeds have increased since logic and gates are placed closer, decreasing electrical path length and computers have become smaller, use less power and have more circuitry, which means there are fewer, unreliable inter-chip connection.

The IBM System/360 is a good example of the consequences of the change. It was incompatible with older machines, which was bad news, but it was necessary to break out of older constraints and produce systems with ICs. The decision paid off financially and technically. They produced the first family of computers, with members only having a difference in execution time. Many of its features became standard on other large computers.

The IBM System/360 had the following characteristics:

* Similar instruction set so all any member of the family could run a program made by another member
* Similar operating systems
* Increasing speed going to higher family members
* Increasing number of I/O ports going to higher family members
* Increasing memory size going to higher family members
* Increasing cost going to higher family members due to the complexity it supported

Another computer built around the same time is the PDP-8 by Digital Equipment Corporation (DEC). It was so small that it could be places on a lab bench, and cheap enough for individuals to buy. It also had something that became universal for microcomputers, the bus structure.

The bus structure, originally called the Omnibus, was a common set of signal paths use by all the components, and controlled by the CPU. This made everything much smaller, gave the CPU less work to do, and allowed for extra components to be added to the system.



### Later Generations

After the third generation, further improvements came too rapidly and too often to separate things into different generations. Some of the prominent improvements include:

* Use of semiconductor IC technology in memory construction. Earlier, memory was made from tiny rings of ferromagnetic materials with a magnetic core. These were fast, but expensive and bulky and used destructive readout, meaning just reading the data stored would destroy it. This made it necessary to install circuits that would restore the data as soon as it had been extracted. The first semiconductor memory on the other hand, was about the size of a single core and held 256 bits of memory. It was non-destructive and much faster. However, it was a little more expensive.
* As the density of elements on a processor chip increased, fewer and fewer chips were needed to produce a single computer processor. In 1971, Intel created the first chip with all the components of a CPU on it, the Intel 4004 microprocessor. It could only add 4-bit numbers, and multiply through repeated addition, but it was the beginning of the microprocessor evolution. In 1972, Intel brought an 8-bit microprocessor, the Intel 8008, but even this, like its predecessor, was built for a specific application. In 1974, the Intel 808 came, which was a general-purpose microprocessor with greater speed and a richer instruction set. Then came the 16-bit 8086, and finally 32-bit microprocessors starting with the one made by HP and Bell Lab in 1981.

1.4 RISC and CISC Architecture Processors (No clue where this information is coming from. It is not from the book.)

The RISC Architecture Processor represents the Reduced Instruction Set Computer. It utilizes small and highly optimized instruction sets, that need a single clock cycle to be executed. This is also known as LOAD/STORE Architecture. Since instructions are simplified, program code becomes longer, so software needs to be optimized. The addressing modes are simpler, opcodes are smaller, there are fewer parameters in code and everything is cheaper, faster and more power efficient. However, more registers are needed to reduce memory access. Compiler and hardware design is simpler and pipelining is easier to implement for simple instructions. RISC is used in mobile phones, smart devices and ARM processors.

CISC represents the Complex Instruction Set Computer. The main goal is to complete the task using fewer instruction lines. Programming code is smaller, but instructions are complex. Hardware and decoding can also be complex. Multiple clock cycles are needed for each instruction execution. There are more addressing modes and more memory access for operands. There are fewer registers in the processor. Pipelining is difficult to implement for complex instructions. CISC supports microprogramming/micro coding, i.e. low-level code where instruction are groups of microprogram codes and are executed accordingly, defining how the microprocessor executes instructions. They are inbuilt in the main memory of the processor and are faster than memory access. CISC is used in laptops, general purpose PCs and x86 processors.

Both RISC and CISC perform the decode and execute phase in instruction execution. RISC is highly efficient, but uses more lines of code, whereas CISC has more lines of instructions. RISC is simple and cost-efficient, being used in mobiles, while CISC is used in PCs. Nowadays however, a hybrid of the two is used.

## 1.5 Embedded Systems

Embedded systems refer to the electronics and software that is inside a product as opposed to a standalone computer. This includes everything from smart watches to smart washing machines. Such computers are often tightly coupled with the outside environment, detecting things like temperature and pulse rate and making real time calculations in order to perform activities based on those detections. An embedded system may have the following additional components:

* Variety of interfaces that allow system to interact with external environment
* Human interface to interact with user
* Diagnostic port to diagnose system being controlled
* Application specific software
* Optimized energy use, code size, execution time, weight, dimensions and cost
* Embedded operation system that contains the existing OS adapted for embedded use or an OS intended solely for embedded use
* Deeply embedded operating system that uses the microcontroller rather than the microprocess and is dedicated for a specific purpose

### The Internet of Things

This refers to the expanding interconnections of smart devices. The theme is the embedding of short-range mobile transceivers into a wide array of gadgets and everyday items, enabling new forms of communication between people and things, and between themselves. This is of course primarily driven by embedded devices, while the interconnection is supported through the cloud system.

### Application and Dedicated Processors

An application processor is able to execute complex operation systems like Linux and Windows. They are general purpose. An example of an embedded application processor is the smart phone.

A dedicated processor is dedicated to a small number of specific tasks required by the host device.

### Microprocessors and Microcontrollers

A microprocessor includes registers, an ALU, a control unit, instruction processing logic and multiple cores and cache memory.

A microcontroller is a single chip containing a processor, ROM, RAM, clock and I/O control unit units. It is a computer on a chip. It is slower than a microprocessor, does not allow human interaction, is embedded on a device and is programmed for a specific task.

## 1.7 Cloud Computing

Cloud computing is an internet connected infrastructure, also known as enterprise cloud computing since it is mostly used by large organizations for IT operations. Individual users also rely on cloud computing to backup, sync and share data using personal cloud computing.

With cloud computing, it is possible to provide economies of scale, professional network management and professional security management, all things that attract companies. Users only pay for storage capacity and services that they need.

Cloud computing is a model for enabling easily available, convenient, on-demand network access to a shared pool of different types of computing resources like networks, servers or applications. It can be quickly provided and released with minimal management effort or service provider interaction. The user does not have the hassle of setting up a database system, acquiring hardware, performing maintenance or backing up data, since all of this is handled by the cloud service.

Cloud networking is another term that refers to the networks and its management that allows cloud computing. This service mostly relies on the internet, but high-performance, highly reliable networking can also be provided. This basically bypasses the internet and uses dedicated private network facilities to provide all services on behalf of the internet. More generally, cloud networking involves providing services like cloud access, linking with data centres, use of firewalls and network security.

Cloud storage is a part of cloud networking, consisting of database storage and database applications. It allows small businesses and individuals to take advantage of database applications without having to buy, maintain or manage the storage assets.

Cloud services are provided using one of the following models:

* Software as a Service (SaaS) – Application software is provided that runs on and is accessible from the cloud, using interfaces as simple as a Web browser.
* Platform as a Service (PaaS) – A platform is provided on which the clients software can run. There are also a few development tools that allow the development of new applications.
* Infrastructure as a Service (IaaS) – The customer is given access to the underlying cloud infrastructure and is provided with virtual machines and other abstract hardware and operation systems.