

Technical Report

COMP-2660

COMPUTER ARCHITECTURE ii

Anatomy of a CPU

*Author:*

Ishrat Sikder

*Submitted to:*

Ikjot Saini

October 17, 2022

*Anatomy of a CPU*

Ishrat Sikder

University of Windsor

sikderi@uwindsor.ca

1. Introduction

A Central Processing Unit or the CPU is essentially the brain of a computer. Similar to a human brain, it consists of various parts which work together to process information. Like the lobes of a brain that are responsible for different tasks, the CPU also contains components accountable for bringing in, transfer, store and perform actions with the information it has. Different devices serve different purposes, meaning, their CPU varies depending on the operations it must perform. However, the fundamentals of any CPU come down to the same basics and functions. Without the CPU no electronic device can run any programs; as a result, the CPU plays a crucial role in the making of any computer. As CPUs are the head of a computer, understanding the roles each component play and how they work together as a whole can help us comprehend how the computer performs the tasks programmers assign through the use of higher-level languages.

To start off, the basic anatomy of a CPU contains 6 sections, they are the control unit, arithmetic logic unit (ALU), registers, caches, buses, and lastly the clocks. Each of these works together to run various programs and perform necessary tasks. From a single-core Intel 4004 chip to multi-core Intel i9 chip, how far have we come and how did this all start 40 years ago?

1. Importance

Since the release of the first processor 40 years ago, computers have evolved to being an essential part of our lives. Even just 20 years ago not many people had easy access to computers, whereas life without phones, laptops is unimaginable to many around the world now. We have dealt with how human written instructions can produce programs, but how does the computer work with these instructions to provide the desired results? To understand this, we must first look at how the CPU works individually with other parts of a computer, and then together as a whole. Having an understanding of how the smallest of the parts function in sync with others helps programmers and developers to perform their tasks far more efficiently.

Even as users of computers, having basic knowledge of what each element do and how the internals components function can help understand what features are necessary to accomplish the desired results. Business organizations can easily identify which types of computers will be most cost effective and provide all essential properties, artists and gamers can select handy but easy to set-up devices with incredible graphics card, and general users can choose one within budget but provide basic latest features. Computer users can also understand where or what the issue and is and troubleshoot accordingly.

1. technological details
   1. *The CPU Blueprint: The ISA*

When analyzing any CPU, one of the primary parts to look into is the Instruction Set Architecture (ISA). It’s the blueprint of how CPUs operate and the way each internal system should interact with others. Similar to how various breeds of cats exist under the same species, there exists numerous types of ISAs onto which CPUs can be built. The most common types being the *x86* (found in desktops and laptops) and *ARM* ­or *Advanced RISC Machine* (found in embedded mobile devices).

An ISA specifies the instructions a CPU can process, its interactions with memory and cache, and how tasks are distributed in different stages of processing and more.

The following are the major sections of a CPU in the order instructions takes to be executed. Depending on the type of CPU the path of instructions set may vary, however the following sections covers a generalized route. We’ll focus on the most basic design of a single-core processor which was first revealed, and later introduce some of the more modern components used today.

* 1. *Control Unit and Datapath*

The internal structure of any CPU can be divided into two: the control unit and datapath. Like a car and its driver, the control unit is the engine of the computer running the machine. Whereas, the datapath is the driver controlling different aspects of the engine to keep it operating.

As the name suggests, the datapath is where the data flows during a process. The datapath receives the inputs, processes them and forwards them to the appropriate location. The control unit directs the datapath on how to operate the car like a driver does. Depending on the instruction set, the datapath will map the signals to reach different parts, activate or deactivate different areas while monitoring the state of the CPU.

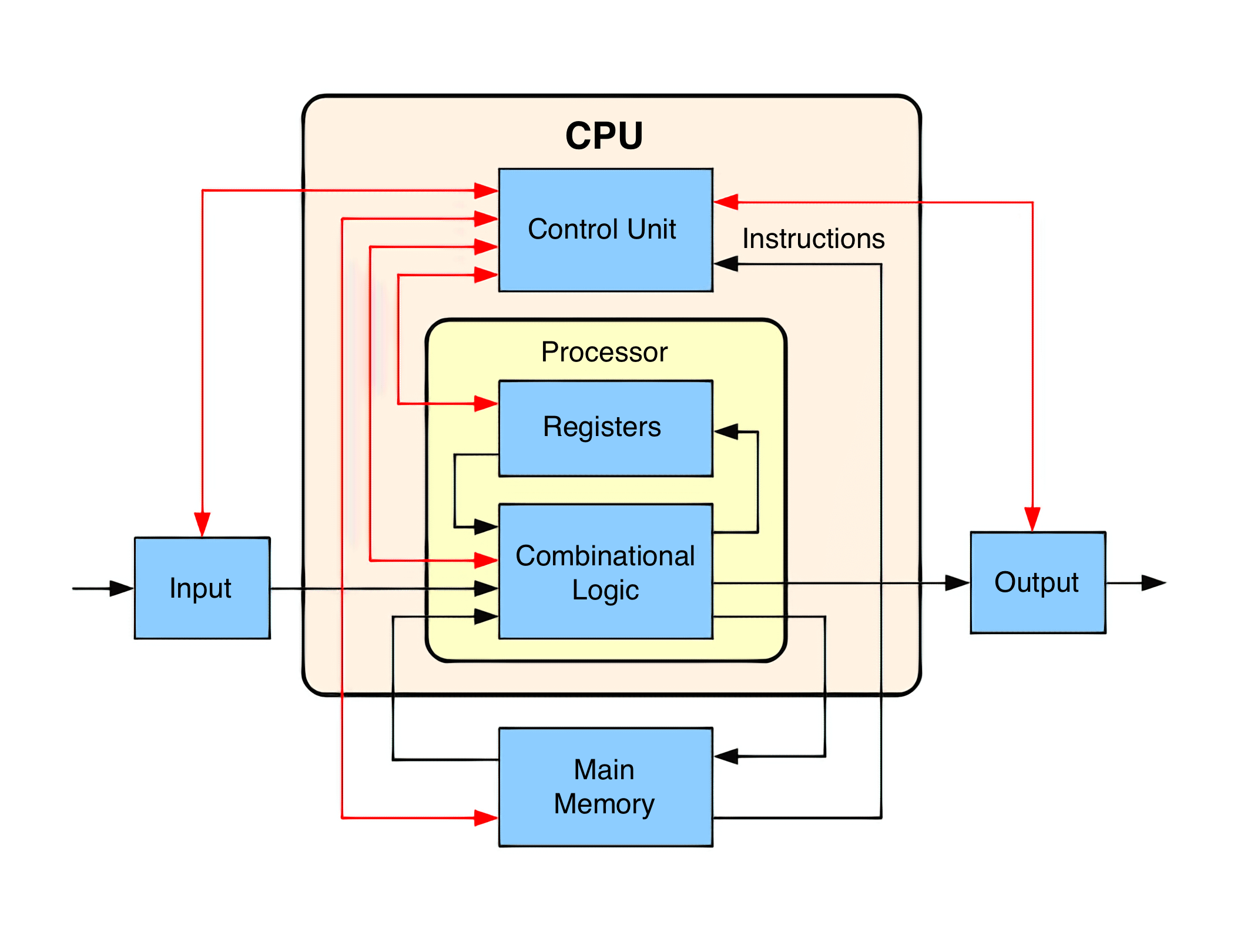


Fig. 1. Block diagram of basic CPU (Black lines indicate data flow, and red lines indicate control flow)

* 1. *The Instruction Cycle - Fetch*

The first thing a CPU must do is figure out what instructions to execute next and transfer them from memory into the CPU. Instructions are produced by the compiler and are unique to the CPU’s ISA. ISAs share most common type of instructions such as load, store, add, subtract and others, but there are other special instructions unique to each ISA. The control unit knows where each signals have to be routed for all the instruction types.

When someone runs an executable file on Windows for example, the code for the program is moved into memory and the CPU is informed what address the first instruction begins at. The CPU constantly maintains an internal register which stores the memory address of the next instruction to be executed, and this is called the Program Counter (PC).

Once the program has figured out where to begin, the next thing to do in the instruction cycle is to receive the first instruction. The instruction is moved from the memory and onto the CPU’s instruction register, known as the *fetch* stage.

* 1. *The Instruction Cycle - Decode*

When the CPU receives any instruction, it needs to identify the type of instruction to be executed, which is the *decoding* stage. Each instruction comes with a set of bits called the Opcode that tells the CPU how to interpret it. It’s like how file extensions are used to classify the type of file being used, such as *.docx* being a word processing file, *.png* being image file and so forth. The manner in which data is organized in each file type differs, it is the reason behind why the machine must know the data type to interpret correctly.

The instruction decoding stage may become complex depending on the complexity of the ISA itself. RISC-V for example, may only contain few instructions while others such as x64 or x86 have thousands more. This is why decoding stage is the most challenging and also consumes the largest amount of space in the CPU. Some of the most common instruction decoded by CPU are memory, arithmetic and branch instruction among others.

* 1. *The Instruction Cycle – Execute and the ALU*

As mentioned before, the CPU commonly decodes three types of instructions: memory, arithmetic and branch. Let’s start off with the arithmetic instruction, the easiest one the three to understand. The arithmetic instructions are put into something called an Arithmetic Log Unit (ALU) for processing. An ALU is a circuit which takes two inputs – binary or integer – along with a control signal to produce an output as a result of the operation. Think of a standard calculator, to add two numbers for example, you enter two numbers and addition operation to calculate the sum. In the case of an ALU however, the operand is determined by the instruction opcode sent to the ALU by the Control Unit (CU) rather than a user. ALUs can also perform the following bitwise operations such as AND, OR, NOT and XOR. Aside from result output, ALU also outputs a status flag to the CU of the last performed calculation. The status flags are positive, negative, zero or overflow.

Even though ALU is primary used for arithmetic and logical calculations, it could also be used for memory and branch instructions. For instance, the CPU could be required to calculate the memory location as a result of some arithmetic operation, or even calculate the offset to increment counter needed by a branch instruction.

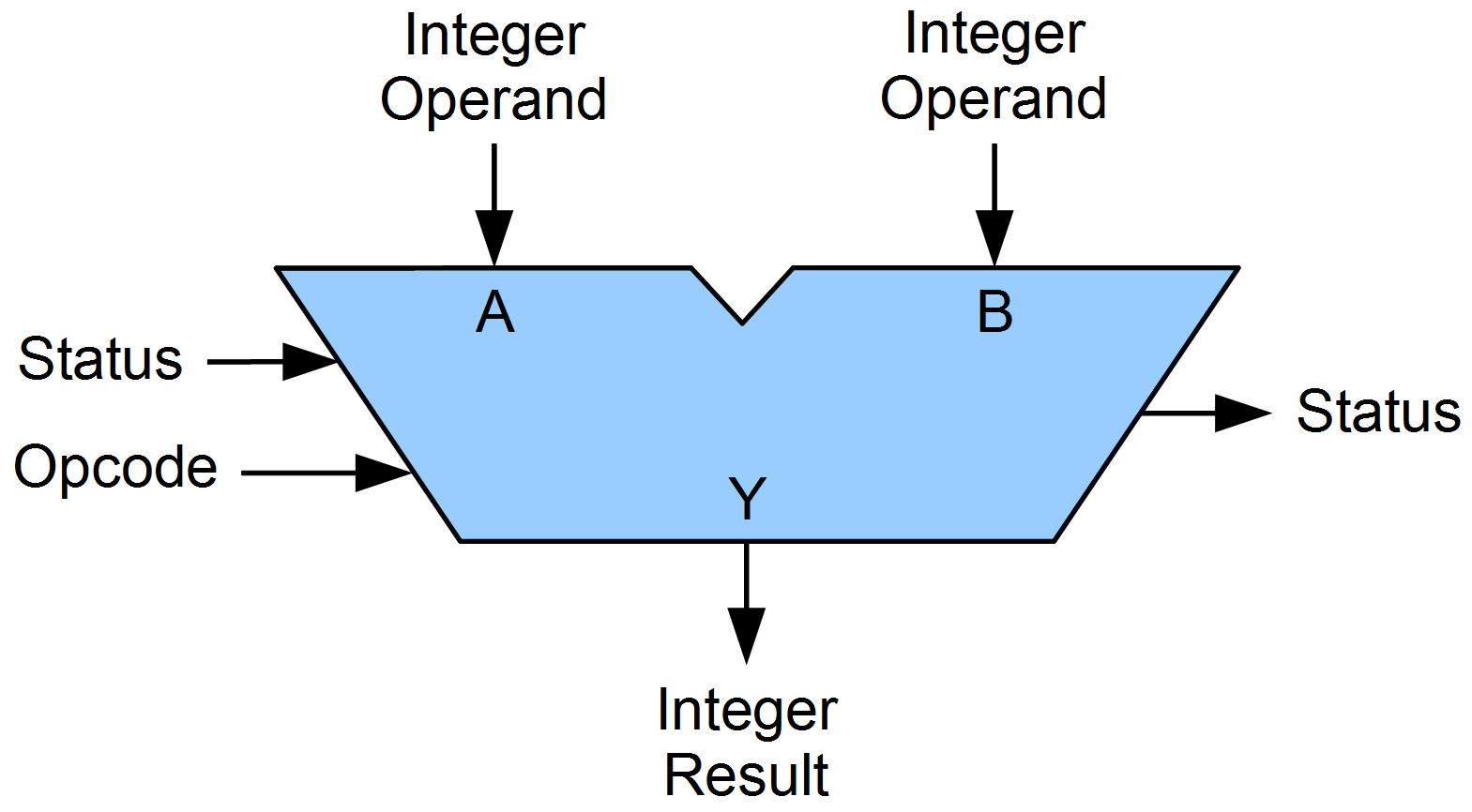


Fig. 2. Arithmetic Logic Unit (ALU)

* 1. *The Instruction Cycle – Memory Instructions and Hierarchy*

The next type of instruction that CPUs decode is memory instructions. Before we begin, it’s important to understand the concept of *Memory Hierarchy.* This represents the relationship between caches, RAM and the hard drive. When a CPU receives a memory instruction for some data which is yet to be present locally in its registers, it goes down the memory hierarchy until found. Most modern CPUs used today have three levels of caches – L1, L2, L3. Starting with L1, the CPU searches and continues until L3. L1 is the smallest and the fasted of the three. It is also generally divided into two portions: one for data and instructions for the other. It’s important to remember that like data, instructions have to be fetched from memory before execution.

As L1 is very small in size, it usually goes a few hundred KB at most. If the CPU fails to find the data it moves onto L2, which goes up few MB in size. Again, if failed CPU goes to the last level, L3 which may be few tens of MB. If the CPU still cannot find the data it moves to RAM, and lastly the hard drive. As the CPU travels along the hierarchy, the storage space as well as the latency rises by a large magnitude.

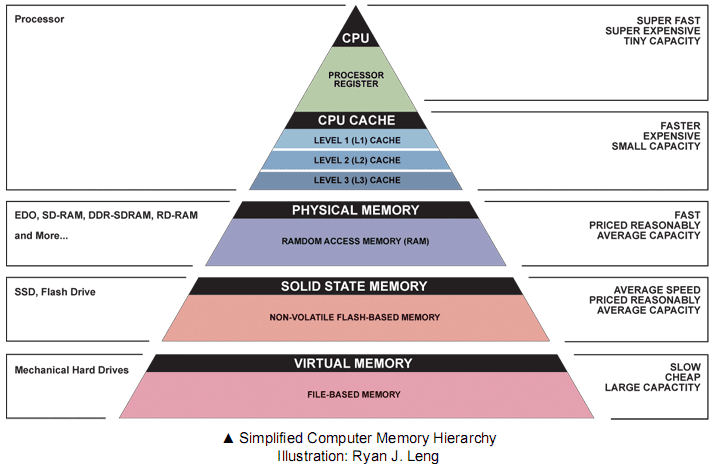


Fig. 3. Memory hierarchy

Once the CPU is successful in finding the data, it travels back up to be easily accessible by CPU for future use, that is, to the registers as it can be read in just mere cycle or two.

* 1. *The Instruction Cycle – Branch and Jump Instructions*

The last of the three aforementioned instruction type is the branch instruction. Modern programs jump around here and here constantly and a CPU will rarely ever execute over a dozen consecutive instructions without a branch. Branch instructions derive from programming elements such as if-statements, for-loops and return statements. They are called branch instructions as they interrupt the flow of the program execution to switch to a different part. Jump instructions on the other hand are instructions which transfer the program sequence to the described memory address depending on the presence or absence of a condition.

Branch instructions, especially conditional branches, are particularly tricky for CPU to manage since it may not determine the result of a branch until it starts the next instruction while executing multiple instructions simultaneously. To understand why that is the case, we need to understand pipelining. We know that each step of the execution cycle will take several cycles to complete, meaning during the instruction fetching stage the ALU sits idling. To minimize time consumption and maximize CPU’s efficiency, each stage in instruction cycle is divided in a process called pipelining.

A simple way to explain pipelining is thinking of doing laundry. Assume you have two loads to do and washing/drying take an hour each. If you are to wash & dry first load and move onto the second, it would take 4 hours. But if you start washing second load while first load dries, the time is reduced down to 3 hours.

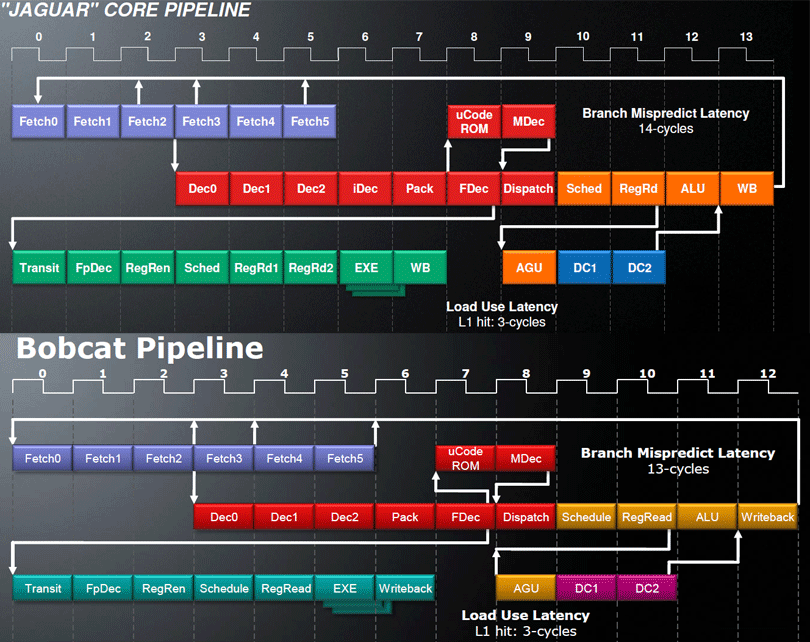


Fig. 4. AMD’s Jaguar and Bobcat microarchitecture pipelines

As seen above, if we are unaware that an instruction is a branch until long later, we will have already started executing all the previous instructions that may become invalid if the branch is taken. In order to overcome this issue, CPUs are built to have a very complex structures called the branch predictors. Similar to machine learning, these try and guess whether a branch will be taken. In simple terms, they track and study previous branch to estimate if the next branch will be taken or not. They have an accuracy rate of 95%+ for modern branch predictors.

Once the output of a branch is determined, the program counter is incremented and CPU proceeds to execute the next instruction. In the case of a mispredicted branch, the CPU dumps out every instruction prior to the mispredicted one and restarts execution.

* 1. *Accelerators*

A demanding feature found in modern CPUs are task-specific accelerators. As the name suggest, their job is to speed up the particular process to make the CPU much more efficient. Common types of accelerators include: performance, encryption, graphics, hardware and more.

Despite CPU’s ability to already do the aforementioned tasks, having a dedicated unit for each vastly increases the magnitude for improved performance. With the rise of various types of accelerators, the size of CPU may shrink to a tiny portion of the chip.

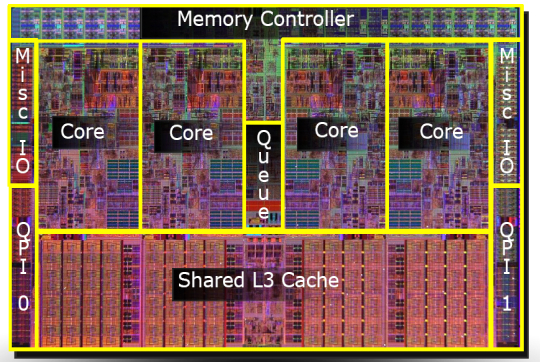


Fig. 5. Intel’s Nehalem microarchitecture CPU

The above picture is a Nehalem microarchitecture CPU from Intel core earlier series. Note that the majority of the chip is taken by cores and caches. Let’s look at a chip from recent years.

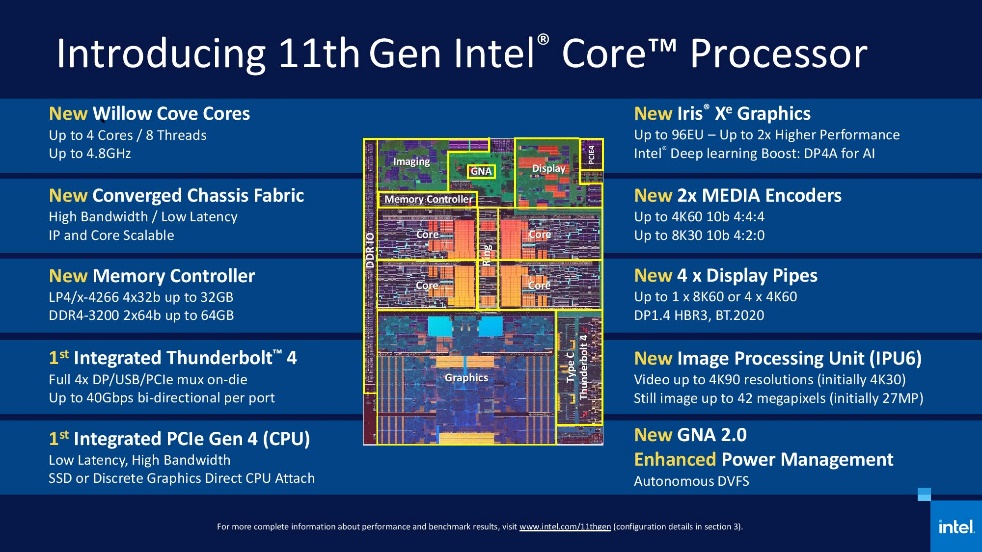


Fig. 6. Intel core i11 chip die

This is the internal structure of Intel core i11 chip. We can see the cores decreasing in size and more space is dedicated to different types of accelerators and other interfaces.

* 1. *Multicore CPUs*

We have previously mentioned about the journey of CPU starting from single chips to the latest being multicores. Multicore CPU is essentially connecting together several CPUs together, and allowing a single physical processor to incorporate the core logic of multiple processors. A single integrated circuits known as die are used to hold these processors. Each of the chips have their own design limits, and designers add different accelerators to boost performance. But multicore chips have several issues to keep in mind as well. Firstly, adding more processors to cores magically doesn’t improve device performance. The OS and the applications must coordinate program instructions to make use of individual cores. This is done through parallel computing, using threads to different cores in the chips. Secondly, multicore doesn’t mean the performance is multiplied by the number of cores used. Even though multicore CPUs allow the PC run several processes simultaneously with greater each, it can also cause congestions in the system due to overloading.

1. discussion

It is my first time learning in depth about how the CPU operates. How the most important parts carry out their roles to keep the machine running as demanded. The concept of memory hierarchy was surely a new face as I have in fact not heard of it previously. It was fascinating to see how fast they can execute an instruction as the highest level, by just a cycle or two which is just a fraction of a second. This reminded of how computer users get impatient even within such a short frame of time. I had learned about sockets, and multithreading, which is why realized how important such function is as they can relief the load on the said computer component without disturbing the performance. Reading through various article also cleared some assumptions I had regarding multicore CPUs. I had thought that adding more cores would raise the performance level in equal magnitude but the reality turned out to be much more complicated. I used to believe that CPUs are improved by adding some larger cores that can take more, never occurred that the cores are shrunk and accelerators are added. Accelerators made me imagined them to be catalysts in a chemical reaction, it was fun to see both are similar – they both speed up the process. Overall learning more about the brain of electronic devices helped me gain lots of knowledge about computer hardware which I usually don’t dig much into. Though due to time constrain it was not possible to explore more, I am pleased to have learnt a little bit of everything through this research paper.

1. conclusion

This article skimmed through the important concepts and simplified the CPU, but this is merely the tip of the iceberg. Now that we have developed some idea about the different functions each part of the CPU play, purchasing laptops or computers should seem easier than before for those who have very little idea about what 8 MB cache, 4 cores, 8 threads so and so mean under the tech specs.

We have seen how the little chips inside the CPU bring in data, process them, and produce some output to be used by both the machine and user. We have seen how the CPU can filter data and instructions put into it and create a road map for them to run through. As we dug deeper into the internal structures, we came to notice the similarities of a CPU and human brain. The development and improvement of CPU is strikingly similar to the development of a human brain. Single core chips are like a young child’s brain, it can process only so many information. Whereas multicore chips are like adults managing work, housework, children all at once. Just as how human brain assess what should be done when fed some information, the CPU also organizes the data and information into separate chambers, and executes the command just like brain signal informing a human what to do. It is incredible how far technology has evolved to ease performing daily functions through machines by which we have grown heavily dependent on.

References

1. *Arithmetic Logic Unit.* Wikipedia.
2. *CPU Block diagram.* Wikipedia.
3. D. Both, “The Central Processing Unit (CPU): Its components and functionality,” *Enable Sysadmin*, 03-Aug-2022. [Online]. Available: https://www.redhat.com/sysadmin/cpu-components-functionality. [Accessed: 15-Oct-2022].
4. D. Kanter, *AMD Jaguar and Bobcat basic pipeline*. Real World Technologies, 2014.
5. I. Cutress, *Intel 11th gen Core processor* . AnandTech, 2020.
6. R. Leng, *Computer Memory Hierarchy*. Google Sites.
7. R. Shrout, *Die shot of Intel Nehalem architecture processor*. PC Perspective , 2008.
8. S. J. Bigelow, “What is a multicore processor and how does it work?,” *SearchDataCenter*, 07-Mar-2022. [Online]. Available: https://www.techtarget.com/searchdatacenter/definition/multi-core-processor. [Accessed: 15-Oct-2022].
9. W. Gayde, “Anatomy of a CPU,” *TechSpot*, 06-Apr-2020. [Online]. Available: https://www.techspot.com/article/2000-anatomy-cpu/. [Accessed: 15-Oct-2022].
10. “What are accelerators in the context of computing hardware?,” *Premio Inc*, 15-Mar-2021. [Online]. Available: https://premioinc.com/blogs/blog/performance-accelerators-in-the-context-of-computing-hardware. [Accessed: 15-Oct-2022].