Modern Computer Architecture using different Technique

Jay Pankajkumar Kania
Surat, Gujarat,
India

ABSTRACT
Computer architecture has been and will always significantly influenced by the underlying trends and capability of hardware and software technologies. Current computer architecture research maintains a bias for the past in focuses to desktop and server application. A distinct computer area, personal mobile computing, will, in my opinion, play a major role in propelling technology in the coming decade. This area will have its own set of needs for microprocessors, which may shift the focus of computer architecture. The evolution of computer architecture from electromechanical relays through vacuum tubes to transistors to integrated circuits has resulted in fundamentally different trade-offs. Additional software advancements include the shift in programming paradigms from machine code to assembly language to high-level procedural language to object-oriented language. The impact on these technology on computer system architecture in past and in future will be explored and projected in future.

General Terms
Computer Architecture

Keywords
Computer Architecture, Computer Organization, Processor Architectures, 64-bit Architecture, x86 Architecture, Computer Memory Architecture: UMA, NUMA, NORMA, HyperTransport (HT)

1. INTRODUCTION
We In the nearly 65 years after the first general-purpose electronic computer was built, computer technology has advanced tremendously. A mobile computer nowadays may be purchased for less than $500 and has more performance, main memory, and disc storage than a computer purchased for $1 million in 1985. This fast advancement is due to advancements in computer manufacturing technologies as well as breakthroughs in software development.

Themost common form of computer architecture is von Neumann architecture, which is still in use today. John von Neumann, a mathematician, suggested this in 1945. It defines the arithmetic logic unit, control unit, registers, data and instruction memory, an input/output interface, and external storage functions of an electronic computer with its CPU.As a consequence, computer architecture guides the design of a computer and specifies the logical interface that programming languages and compilers target.[1]

Themix of functional units that make up the system, as well as the structure of their interconnection, are determined by the organization density and speed of digital switches, as well as the density and access time of digital storage, are essential hardware technologies that influence computer systems.

Thedesign of the building Semantics refers to the meaning of what systems do when they are directed by users, as well as how their functional components are coordinated to operate together. The system's instruction set architecture (ISA) is an essential manifestation of semantics. The ISA is a logical (typically binary) representational encoding of the fundamental set of different operations that a computer architecture may do, and it is used to determine whether a computer architecture is capable of performing those operations. There are different type of architecture System Design, Instruction Set Architecture and Microarchitecture.

TheProcessors are the most important part of the computer architecture systems as it is a component which everything is centered around. A processor is an electronic part of device which is capable of manipulating the data in a specified way by the sequence of instructions which are also known as the Machine Codes. The Sequence of instructions may be altered to suit the application and computers are programmable. Various computer architecture designs have been created to accelerate data flow, allowing for more data processing. The CPU is at the center of the basic design, with a primary memory and an input/output system on each side of the CPU. Almost all architectures define fundamental instructions that operate on memory or registers, such as add, subtract, and branch.[1]

Theconcept of embedded systems and the different processors designs can be known with the help of the ARM Architecture concept used for the understanding of the computer systems works with different operating systems. Many things depend on Computer Architecture and Organization understanding like performance analysis, practical software, parallel software, embedded systems, game programming, databases, accelerators, GPGPU computing, compiler optimization.

ThisPaper is branched into various parts to explain the process and the architecture of the computer systems along with the different methodologies and the synopsis and conclusions shows the future of computer operating systems.

2. RELATED WORK

All Energy dissipation is quickly becoming a major design consideration in current microprocessors. Microprocessors with a few million transistors and tens of watts of power dissipation are common, restricting their utility in portable applications and making heat removal in dense structures challenging. The portable device market is growing, as is the number of gadgets available.

Developers create generic embedded systems as off-the-shelf (OTS) components that can be used in a variety of smart products. These designs must integrate both generality and completeness due to the unknown nature of their applications. In terms of structure and functionality, specialization entails departures from both of these qualities. An application-

specific system or a procedure is the product of extreme specialization.[2]

In spite of the fact that there have been exploratory computer architectures, the von Neumann design proceeds to be the common engineering for computers. It is critical that in a field where innovative changes are so quick that the general computer engineering is essentially unaltered since 1951.Today's CISC and RISC designs are consistent with the wide characteristics that characterize a von Neumann computer.

Computer architectures describe the methods of interconnection for a computer's hardware parts, as well as the manner of information flow and process shown. Various computer design configurations are being created to speed up the transfer of information, allowing for more processing. The central processor is at the centre of the basic architecture, with a primary memory and an input/output system on each side of the central processor. In the future, low-density construction will be promoted. Various alternative computing structures are projected. Brownian computers enable the flight of element particles to follow a stochastic process through the device, the speed of computation (and dissipation) being proportional to the gradient of Associate in Nursing applied force.[3]

The program in programming language adheres to the one-line principle for a single command. every instruction will incorporate many elements (fields), specifically the Label half, the mnemonic half, the quantity half which might be over one and therefore the last half is that the comment section.To tell apart every of those elements, The subsequent partial provisions area unit made:[3]

1. Each half is separated by an area or TAB, particularly for quantity's that area unit over one each operand separated by commas.
2. These elements don't need to all get on a line, if there's one half that's not there, then a space or TAB as a section extractor should still be written.
3. The label half is written ranging from the primary letter of the road, if the road doesn't contain label, then the label is replaced with an area or TAB, that is, as an extractor between the label half and therefore the mnemotechnical half.

The label represents the program-memory number of the instruction on the corresponding line. At the time of writing the JUMP instruction, this label is written in the operand to indicate the designated program-memory number. As a result, the Label always reflects the program memory number and must appear at the start of the instruction line. Besides Label, there is also a symbol, which is a name to represent a particular value and the value represented can be anything, not necessarily a program-memory number.

Symbol is written in the same way as Label, it must start at the first letter of the instruction line. Mnemonic (meaning something that is easy to remember) is a command abbreviation, there are two kinds of mnemonics, namely mnemonics which are used as instructions to control the processor, for example ADD, MOV, DJNZ and so on. There is also a mnemonic that is used to regulate the work of the Assembler program, for example ORG, EQU or DB, the mnemonic to regulate the work of the Assembler program is called an 'Assembler Directive'.

The operand is the part that lies behind the mnemonic part, which is a trap for the mnemonic. If an instruction is likened to a command sentence, then the mnemonic is the subject (verb)and the operand is the object (noun) of the command sentence. Depending on the type of instruction, the operand can be any number of things. In the JUMP instruction.[4]

The operand is in the form of a label representing the memory number of the intended program, for example Start, in instructions for data transfer / processing, the operand can be a symbol that represents the data, for example ADD A, # Offset Many instructions whose operands are registers of the processor, for example MOV A, R1. There are even instructions that don't have an operand, for example, RET. Comments are part that just as a note, do not affect the processor nor affect the work of the Assembler program, but this section is very important for documentation purposes.

3. METHODOLGY

3.1 Computing Systems Architecture

A computer system is essentially a machine that automates complex operations. It should maximize performance and reduce costs also as power consumption. The different components within the computing system Architecture are Input Unit, Output Unit, Storage Unit, Arithmetic Logic Unit, Control Unit etc.

A microprocessor may be a processor implemented (usually) on one, microcircuit. With the exception of certain big supercomputers, virtually all current processors are microprocessors, and the two names are frequently used interchangeably. Common microprocessors in use today are the Intel Pentium series, Freescale/IBM PowerPC, MIPS, ARM, and therefore the Sun SPARC, among others. A microprocessor is usually also referred to as a CPU (Central Processing Unit).

Microcontrollers are integrated circuits with a CPU, memory, and specific I/O devices that are designed for use in embedded systems. Buses that are part of the same integrated circuit link the CPU and its I/O. Microcontrollers are available in a broad range of sizes and forms. [5]

They vary from the tiniest PICs and AVRs to integrated PowerPC processors with built-in I/O.[5]

The data is sent from the input unit to the ALU. The calculated data also flows from the ALU to the output unit. Data is continually sent from the storage unit to the ALU and back. All of the other components, as well as their data, are controlled by the control unit.

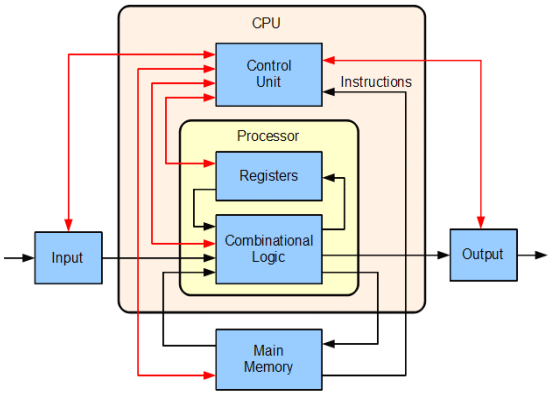

Figure 1: Block diagram of a basic uniprocessor CPU

Details about all the computer system units are –[6]

3.1.1 Input Unit

The input unit provides data to the pc system from the surface. So, basically it links the external environment with the pc. It receives data from input devices, transforms it to machine language, and feeds it into the computer system. The most popular input devices include keyboards, mice.

3.1.2 Output Unit

The output unit displays the outcomes of computer processes to users, i.e., it connects the computer to the outside world. The majority of the output data is in the form of audio or video. Monitors, printers, speakers, and headphones are examples of output devices.

3.1.3 Storage Unit

A storage unit is a collection of computer components that are used to store data. It has historically been split into two categories: primary storage and secondary storage. Primary storage, often known as main memory, is the memory that the CPU has direct access to. The CPU does not have direct access to secondary or external storage. Before the CPU may use data from auxiliary storage, it must be transferred to the first storage. Secondary storage retains an enormous quantity of knowledge indefinitely.

3.1.4 Arithmetic Logic Unit

The arithmetic logic unit performs all computations connected with the computer system. It is capable of performing operations such as addition, subtraction, multiplication, and division. When computations must be done, the control unit transmits data from the storage unit to the arithmetic logic unit. The central processing unit is made up of the arithmetic logic unit and therefore the control unit.

3.1.5 Control Unit

This unit, which controls all of the opposing units of the computer system, is known as the *"central systema nervosum"*. It moves data across the computer as needed, including from the storage unit to the central processor unit and vice versa.

The control unit also specifies how the memory, input/output devices, and arithmetic logic unit should operate.

3.2 Conversionof High-level Language to Machine Code

The procedures for converting a program from a high-level language to machine code and then running it. A compiler initially converts the high-level code to assembly code. The assembler translates assembly code to machine code and stores it in an object file.

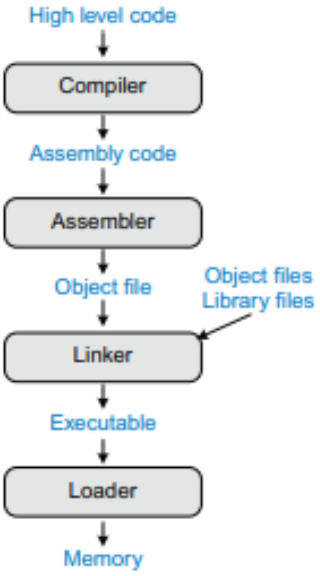

Figure 2: High level code flow diagram

To create a full executable program, the linker mixes machine code with code from libraries and other files, determining the correct branch addresses and variable locations. Most compilers, in practice, complete all three phases of compilation, assembly, and linking. Finally, the program is loaded into memory and executed by the loader.[7]

4. ARCHITECTURE (64-BIT)

A 32-bit architecture allows an application to access up to 232 bytes (4 GB) of memory directly. The shift to 64-bit architectures, which can access huge quantities of memory, was spearheaded by large computer servers. Eventually came personal computers, and then mobile gadgets. 64-bit architectures can also be quicker in some cases since they can transfer more data with a single instruction.

Many architectures merely increased the size of their general-purpose registers from 32 to 64 bits, but ARMv8 added a new instruction set to help with peculiarities. For complicated applications, the traditional instruction set lacks adequate general-purpose registers, necessitating expensive data transfer between registers and memory.Keeping the PC in R15 and the SP in R13 makes the processor implementation more difficult, because applications frequently require a register with the value 0.[8]

The ARMv8 instructions are still 32 bits long, and the instruction set resembles that of ARMv7, albeit with a few flaws fixed. The register file in ARMv8 has been increased to 31 64-bit registers (named X0–X30), and the PC and SP are no longer included in the general-purpose registers. The link register is designated by the number X30.

It's worth noting that there is no X31 register; instead, the zero register (ZR) is hardwired to 0. Loads and stores always

utilize 64-bit addresses, but data-processing instructions can work with 32-bit or 64-bit values. The condition field is deleted from most instructions to create way for the additional bits used to define source and destination registers.

Branches, on the other hand, can still be conditional. ARMv8 significantly improves exception handling, increases the amount of advanced SIMD registers, and includes cryptographic instructions for AES and SHA. The instruction encodings are complicated and do not fit neatly into a few categories.

ARMv8 processors start up in 64-bit mode after a reset. By changing a bit in a system register and raising an exception, the CPU can switch to 32-bit mode. When the exception is thrown, it switches back to 64-bit mode.[9]

Some 64-bit architectures, like x86-64, have more general-purpose registers than 32-bit ones (although this is not due specifically to the word length). Because the processor does not have to acquire data from the cache or main memory if the data can fit in the available registers, tight loops can run much faster.

64-bit Architecture high level code example:

```
int a, b, c, d, e;
for (a = 0; a < 100; a++)
{
   b = a;
   c = b;
   d = c;
   e = d;

}
```

If a processor can only retain two or three values or variables in registers, it will have to transfer some data between memory and registers to process variables d and e as well; this is a time-consuming operation. A processor with enough registers to retain all values and variables can loop over them without having to transfer data between registers and memory for each iteration.

The major drawback of 64-bit architectures is that the same data takes up more memory space in comparison to 32-bit systems (due to longer pointers and possibly other types, and alignment padding). This raises a process's memory needs, which might have an impact on how efficiently the processor cache is used.

One approach to deal with this is to keep a partly 32-bit model, which is often effective. The z/OS operating system, for example, follows this strategy by forcing program code to be stored in 31-bit address spaces (the high order bit is not utilized in address computation on the underlying hardware platform), whereas data objects can be stored in 64-bit areas.

ARMv8 processors start up in 64-bit mode after a reset. By changing a bit in a system register and raising an exception, the CPU can switch to 32-bit mode. When the exception is thrown, it switches back to 64-bit mode.

5. X86 ARCHITECTURE

x86 architecture microprocessors are used in almost all personal computers nowadays. x86, often known as IA-32, is a 32-bit architecture created by Intel. AMD also sells microprocessors that are compatible with the x86 architecture.

The x86 architecture has a complicated history that dates back to 1978, when Intel released the 16-bit 8086 CPU. For IBM's initial personal computers, the 8086 and its cousin, the 8088, were chosen. Intel released the 32-bit 80386 microprocessor in 1985, which was backward compatible with the 8086, allowing it to execute software designed for older PCs. x86 processors are CPU architectures that are compatible with the 80386. x86 processors such as the Pentium, Core, and Athlon are well-known.

Over the years, several departments at Intel and AMD have crammed more instructions and capabilities into the outdated architecture. The end product is considerably less attractive than ARM. However, because software compatibility trumps technological beauty, x86 has been the de facto PC standard for more than two decades. Every year, over 100 million x86 processors are sold. This massive market supports more than $5 billion in annual research and development to keep the processors developing.

A Complex Instruction Set Computer (CISC) architecture is exemplified by the x86 architecture. Unlike RISC systems like ARM, each CISC instruction may perform more work. CISC architectures often require fewer instructions in program.

When RAM was much more costly than it is now, the instruction encodings were chosen to be more compact in order to preserve memory; instructions are varying in length and are frequently fewer than 32 bits. Complicated instructions are more complex to decipher and execute at a slower rate as a result.

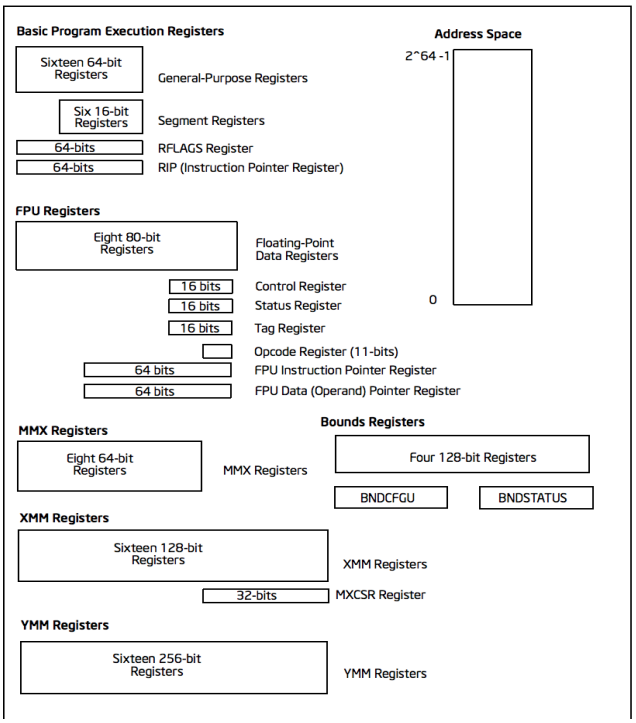

Figure 3: x86 Architecture[10]

The x86 architecture is discussed in this section. The objective isn't to turn you become an x86 assembly language programmer, but to highlight some of the parallels and differences between the two architectures.

6. ALGORITHMIC PARALLELISM

Task/process parallelism can boost a computer's throughput by allowing more tasks/processes to be performed in the same

amount of time. Algorithmic parallelism aims to reduce the time it takes to solve a single issue or job, yet it may actually reduce throughput. Algorithmic parallelism entails revisiting the underlying methods used to solve a problem on a computer in order to devise a new technique that can be efficiently divided across multiple separate processors to work on the problem simultaneously.

The degree of sharing necessary between processors is the basic limit of parallelism and scalability. Sharing can occur at the instruction, data, or I/O levels. Sharing can also occur in actual implementations because buses, cache lines, or other implementation artefacts are shared across processors but are not sharing limitations imposed by the algorithms.[11]

7. COMPUTER MEMORY ARCHITECTURE

Three techniques are being used in computer system design to decrease the amount of sharing at the hardware level while still providing an efficient environment for the present state of the art in applications and operating systems. Uniform memory access (UMA), non-uniform memory access (NUMA), and no remote memory access (NoRMA) are the names given to these three types of multiprocessors.

7.1 Uniform Memory Access

This is the sharing paradigm used by the majority of commercially successful symmetric multiprocessors (SMPs). They are distinguished by all processors having equal access to all memory and I/O. They generally run a single copy of the operating system that has been modified to utilize parallelism and to offer users and all uniprocessor program the impression of a uniprocessor.

While UMA MP technology is fairly established, it does have certain limitations when it comes to using algorithmic parallelism. Because memory, I/O, and OS are all shared by all CPUs, there is a large degree of implicit sharing, limiting the scalability of these systems. This issue simply worsens when the speed of each individual CPU increases. Additional hardware improvements will be made to speed up instances when the algorithms themselves need sharing. Despite these heroic attempts, the scalability limit of UMA systems will remain in the low double digits.[13]

UMA MP systems suffer from system dependability issues in addition to performance scaling issues. Because memory, I/O, and the operating system are shared by all processors, any failure in any component has a significant probability of bringing the entire system down. As a result, lowering the degree of sharing can enhance a system's availability.

7.2 Non-Uniform Memory Access

NUMA machines share a shared address space among all processors, memory, and I/O; nevertheless, access time from any particular processor to parts of the system's memory and I/O is noticeably longer. Memory and I/O with the quickest access time are sometimes referred to as "local" to that CPU. Memory and I/O that is located further away is referred to as "global" or "remote."

NUMA designs will work well as long as most resources can be accessed locally. In reality, in NUMA systems, the average proportion of distant accesses is the ultimate determinant of scalability. As a result, the scalability of NUMA systems is heavily dependent on the application's and operating system's capacity to decrease sharing.

Non-shared data (or instructions) exist exclusively locally by default. The difficulty is in decreasing the amount of data that is updated and read by several processors. This migration necessitates the relocation of the related local data. Because the operating system and hardware are attempting to control this at run time using just run-time data. There are two ways to designing an operating system for NUMA systems. The initial step is to adapt an operating system built for UMA machines to handle memory replication and remote/local optimizations. Many operating system functions will eventually be replicated to more distributed ways, such as having local task dispatch queues coordinated by an overarching global task dispatch.[12]

This method has the advantage of beginning from a design position where no sharing occurs and gradually adding sharing and collaboration through global services as needed. The disadvantage is that because not all services are global, migrating from UMA machines is difficult. Furthermore, upgrading the operating system to allow it to collaborate on global resource control is typically not straightforward.

7.3 No Remote Memory Access

Memory, I/O, and operating system copies are not shared between NORMA computers. Individual nodes in a NORMA machine communicate with one another via transmitting messages. High performance NORMA machines concentrate on increasing communication bandwidth and latencies between nodes, particularly at the application level.[14]

Global services and resource allocation are supplied by a layer of software that coordinates the operations of the many copies of the operating system, similar to the second method stated for NUMA operating system design. From a hardware standpoint, the NORMA method is clearly the most scalable, and it has shown this capacity for applications that have been designed to have restricted sharing as well. Most apps, however, have yet to make this change. NORMA machines are beginning to offer "remote memory copy" capabilities to aid in this complex process and enhance performance.

Both machines are made up of nodes having local memory and I/O that are linked to other nodes through a fabric. Data is transmitted from one node to another in a NUMA computer when a cache miss to a distant node occurs. Data is transmitted between nodes in a NORMA machine at the request of an application (with a PUT or GET command).

The essential distinction is whether the transfer was started by hardware (a NUMA cache miss) or software (a NORMA PUT/GET). Because of these commonalities, several current research efforts are aiming to develop computers that can support both at the same time.

8. HIGH PERFORMANCE COMPUTING SYSTEM ARCHITECTURE

Initially, most HPC systems were built using specialized hardware. By the late 1990s, HPC system designers realized that utilizing huge arrays of off-the-shelf processors might enhance performance and save time to deployment. HPC system designs, as well as some of the most recent HPC system architectures based on huge compute arrays, sometimes known as massively parallel computing, are discussed.[15]

8.1 Large Compute Nodes

Cray computers were built in the shape of a circular rack, with

the circuit board connectors near to each other in the middle of the ring to decrease cable lengths and, hence, signal propagation delays. Rather than the bigger high-density CMOS processor chips that were available in the early 1980s. Throughout the 1990s, other firms, such as Convex Computer Corporation, were creating high-performance computers employing huge processor boards constructed with various types of GaAs gate arrays.

These high-performance processors were assembled on a huge board to create the functional components of a computing node, with each system having several racks of these boards. However, the days of building supercomputers from a few big high-performance computing nodes came to an end in the late 1990s, as new techniques based on massive arrays of off-the-shelf CPUs came to the fore.[18]

9. HYPER TRANSPORT

In 2001, numerous firms, including AMD, Apple, and NVIDIA, created the Hyper Transport (HT) Consortium, and by 2003, many new products, including devices from AMD and others, had appeared implementing this new interface standard.

The same year, the HT Consortium published a standard for multiprocessor communication on a CPU board through an interconnect mechanism known as a system bus. This HT bus is a point-to-point connection that uses 2-32 bits per link and has a throughput of up to 51.2G data per second.

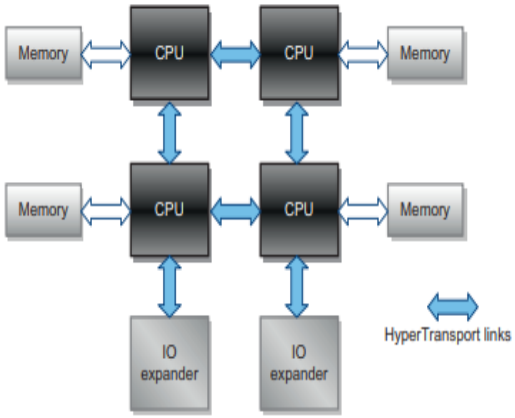

Figure 4: Hyper transport links for data transport between CPU set and I/O devices[17]

To transport data between devices, HT employs packets, which are made up of several 32-bit words. Packets are tiny and contain a command field as well as additional optional fields such as address information and data. Because the packet overhead is kept minimal, the connection bandwidth may be effectively utilized.[16]

Software is a critical component of HPC. Massive parallel processing necessitates close coordination between all of the computing capabilities in the array, and it is here that software programmers spend the most of their effort optimizing code. One important aspect of their work is to maximize data transmission between computer nodes via the switch fabric.

MPI offers an application programming interface (API) that contains the syntax and semantics for core library functions that allow various forms of communications between routines running on individual processors or processor cores. Although communication can take place between process pairs, MPI also enables gather and reduce operations, as well as graph-like logical procedures and other monitoring and control activities.[17]

OEMs creating big parallel computers can use MPI to give their customers with higher level distributed communication procedures. MPI is a higher layer protocol that operates on top of lower layer protocols like sockets and TCP/IP.

10. CONCLUSION

The challenges related to building higher and higher architectures are continuously there. The ever-increasing performance and decreasing prices, make computers a lot and more cost-effective and, in turn, accelerates extra software system and hardware developments that fuel this method even a lot. As new architectures come up, a lot of refined algorithms are run on them, and these algorithms keep stringent a lot of, and the method continues.

Computer designs have developed to maximize the use of underlying hardware and software technologies to attain higher levels of performance. By leveraging the growth in density of digital switches and storage, computer performance has improved faster than the underlying improvement in performance of the digital switches from which they are constructed.

Parallelism at three levels has been used: instruction-level parallelism, task/process-level parallelism, and algorithmic parallelism. These three levels of parallelism are not mutually exclusive and will most likely be utilized in tandem to increase system performance in the future.

Assembly language may be a low-level artificial language employed in programming computers, microprocessors, small controllers, and different programmable devices. Assembly language implements the illustration of code within the kind of symbols that area unit relatively more graspable to humans. programming language typically supports specifically for one or several specific sorts of pc architectures. As such, the movability of the assembly language cannot match different high-level programming languages.

The Computer Architecture helps the operating systems to work accordingly and manages the system architecture of the various kind of methodologies which are used for the betterment of the Work-Flow for different combinational optimization scenarios and the use of higher-level operating systems. The different scenarios are explained in the methodology of the research paper.

11. FUTURE WORK

Computer Architecture is the field that designs computers, which sets the establishment for the whole IT industry. One space of huge freedom in computer design is the manycore challenge. The objective is to connect computers that make it simple to compose programs that are productive, convenient, right, and scale as the quantity of centers per microchip increments – as simple as it has been to compose programs for conventional PCs. On the off chance that this innovation permits programming to utilize numerous straightforward force productive centers rather than a solitary force hungry center, this will reset the establishment for the IT business for basically the following 30 years.

A new opportunity has been created for a new computer architecture that improves computer security and privacy, which causes error in the IT field. Computer engineers could

remove many of the vulnerabilities of today's computers if they were not bound by the old requirements of compatibility with today's computers. They could also provide and upgrade it to new features to make it easier to build fast, secure, efficient virtual machines which can make it easier and safer for software to migrate between various operating systems.

A third opportunity can be created about the computers that will remove the performance hindrances from new, highly productive programming environments such Ruby or Python. For instance, the Ruby on Rails allows software engineers to concoct astounding new computer applications in only 1000 to 2000 lines of code—components of 10 to 100 not exactly ordinary methodologies.[19]

12. ACKNOWLEDGEMENTS

The author wishes to thank the IJCA for the content and support for authors. Also, the editor for handling the submission during the pandemic,and the anonymous referees for the diligent reading.

13. REFERENCES

[1] Vieri, C. J., 1995. Pendulum: A Reversible Computer Architecture. Thesis in Massachusetts Institute of Technology, Cambridge, MA.

[2] Rau, B. R., Schlansker, M. S.: "Embedded computer architecture and automation"; Computer, Vol. 34, No. 4 (2001), pp. 75–83. http://doi.org/10.1109/2.917544

[3] I. R. V. ENGLANDER, Architecture of computer hardware, Systems Software, and networking: An information technology approach: JOHN WILEY, 2020.

[4] P. J. Fortier and H. E. Michel, *Computer Systems Performance Evaluation and prediction*. Burlington, MA: Digital Press, 2003.

[5] A. N. Napitupulu, I. E. Simorangkir, N. Brahmana, L. Sitorus, S. U. A. Togatorop, and D. Sitompul, "LITERATURE STUDY OF THE LANGUAGE PROGRAMMING ASSEMBLY ON COMPUTER ARCHITECTURE," *Academia*, 2021. [Online]. Available: https://www.academia.edu/49257248/LITERATURE_STUDY_OF_THE_LANGUAGE_PROGRAMMING_ASSEMBLY_ON_COMPUTER_ARCHITECTURE. [Accessed: 2021].

[6] "Assembly language," *Cleverism*, 11-Aug-2016. [Online]. Available: https://www.cleverism.com/skills-and-tools/assembly-language/. [Accessed: 2021].

[7] P. Prakash, "Low level vs high level language - difference between lowand high level language," *Codeforwin*, 13-Oct-2018. [Online]. Available: https://codeforwin.org/2017/05/low-level-vs-high-level-language-difference-low-high-level-language.html. [Accessed: 2021].

[8] R. Dajan, "Chapter 7 introduction to LC-3 assembly language," *SlideServe*, 18-Sep-2014. [Online]. Available: https://www.slideserve.com/ramla/chapter-7-introduction-to-lc-3-assembly-language. [Accessed: 2021].

[9] P. Knaggs, ARM Book: ARM Assembly Language Programming. 2016.

[10] *Assembly programming tutorial*. [Online]. Available: https://www.tutorialspoint.com/assembly_programming/index.htm. [Accessed: 2021].

[11] R. E. Gonzalez, "XTENSA: A configurable and Extensible Processor," *IEEE Micro*, vol. 20, no. 2, pp. 60–70, March-April 2000, doi: 10.1109/40.848473.

[12] A. K. Sharma, Programmable logic handbook: Plds, CPLDs, and fpgas. New York, NY: MacGraw-Hill, 1998.

[13] D. W. Knapp, Behavioral synthesis: Digital System design using the synopsys behavioral compiler. Upper Saddle River, NJ: Prentice Hall, 1996.

[14] J. P. Elliott, Understanding behavioral synthesis: A practical guide to high-level design. Boston, MA: Springer US, 2012.

[15] W. F. Lee, *VHDL: Coding and logic synthesis with Synopsys*. San Diego, CA: Academic Press, 2000.

[16] R. F. Sechler and G. F. Grohoski, "Design at the system level with VLSI CMOS," *IBM Journal of Research and Development*, vol. 39, no. 1.2, pp. 5–22, Jan. 1995.

[17] R. Groves, "18th CERN School of Computing," in 1995 CERN school of computing: Arles, France, 20 August - 2 September 1995: Proceedings, 1995, vol. 18, pp. 147–159.

[18] G. Lee, "High-Performance ComputingNetworks," *Cloud Networking*, pp. 179–189, Jun. 2014.

[19] S. Furst, "System/ software architecture for Autonomous Driving Systems," *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pp. 31–32, May 2019.