Computer Architecture Technologies – Past, Present and Future

Bhavneet Soni

Author's Note

With the advent of technology and exponential growth in manufacturing and scientific know how we have seen tremendous growth in computing power available. To think of we have more computing power in the palm of our hands greater than the combined computing power of the Apollo missions that got man on the moon, A modern day IPhone 6 has enough computing power to run 120 Million Apollo Rockets Guidance module simultaneously (Ledak, 2015). We have been able to harness all this power with the help of better Computing power and Architecture.

Computer Architecture Technologies – Past, Present and Future

Ever since the human race could use machines to automate calculating tasks computing has undergone quite a few paradigm shifts from mechanical computing devices to electro-mechanical devices, to electrical devices all the way to electronic devices. We have progressed from mechanical devices such as Abacus in early centuries, calculating clock in 16th century, Pascaline in early 17century to Difference engine and Punched Card tabulating machine in early 1900s. Computing power saw an exponential jump with the advent of Vacuum tubes and transistor technology in the mid-1900s, and microprocessor chips in later part of the 20th century. In 1965 Gordon Moore, founder of Intel stipulated that number of transistors per square inch of an Integrated Circuit will double every year.

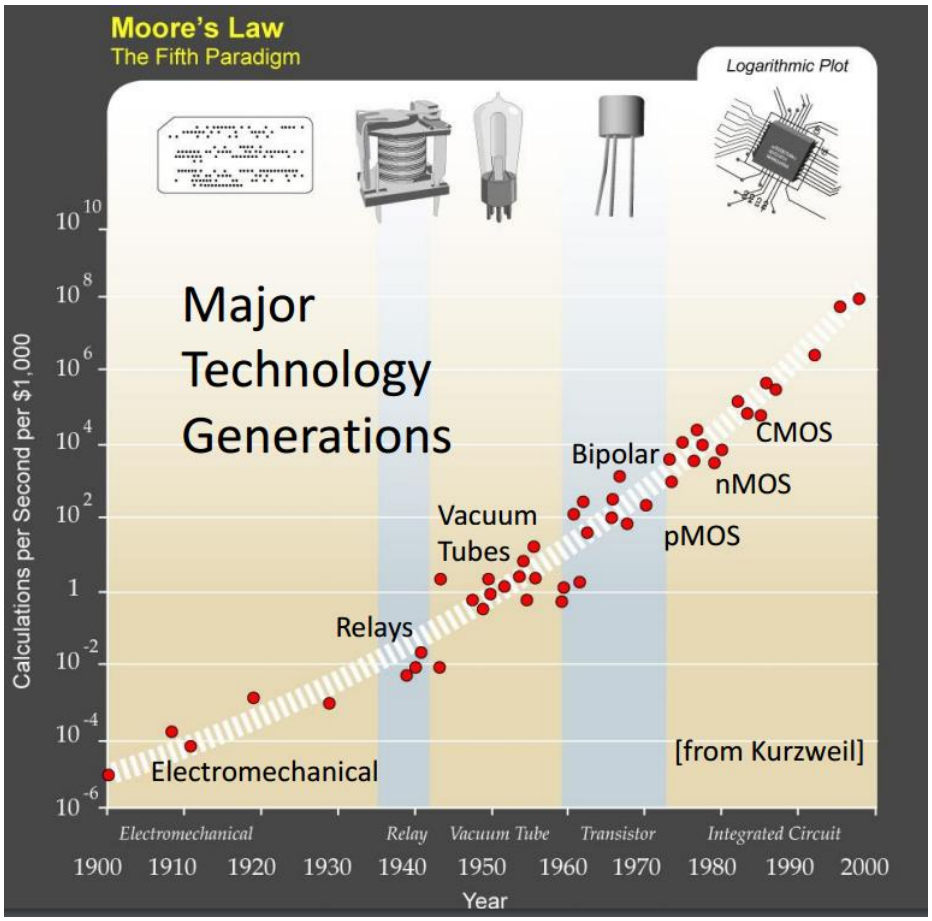

Figure 1- Cost of computing over period of time (Wentzlaff, 2013)

Figure 1 shows a logarithmic graph, extension of Moore's Law of how the cost of computing power progressed over time with respect to cost of computing. Computing has changed from Mainframes running in big rooms during 1950s and 1960s to Minicomptuers 1970 and 1980 to PC and Web 1990 and 2000 to Cloud Computing where all the software and data are being run of big computer farms and content is delivered to the end user over the web. Computer industry is gearing itself for yet another big technological leap divulging into Quantum computing and Nano biological computing. With increasing complexity and computing power arise the need for better computer architecture.

Computer Architecture can be defined as the design of implementation layers that lets us interact and meaningfully process information using a machine aka Computer. Different interaction layers are designed to meet the end user requirements of desired computing prowess in a reliable and efficient way. Figure 2 shows the different subcategories between the actual physics behind the computing and the application, layers Instruction Set Architecture, Microarchitecture and Register-Transfer Level primarily defined as computer architecture.

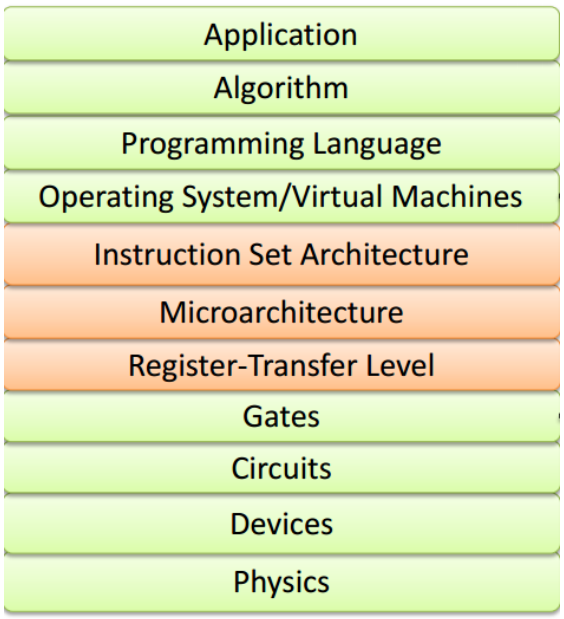

Figure 2 Abstraction layers between The Physics and the actual application (Wentzlaff, 2013)

Since Primary objective in computing is to maximize the output and minimize the cost, a good architecture should support a wide range of implementations while keeping a low cost, should be easy to program and must be compatible with older generation (backward compatibility) with room to overlap with the new generation processors (forward compatibility). In this report we will discuss how the different layers of Computer Architecture has evolved over a period of time. We will discuss namely the abstraction layers

- Instruction Set Architecture
- Micro Architecture
- Register Transfer Level

Instruction Set Architecture

The lowest level of abstraction visible to a programmer is Instruction Set Architecture (ISA), compiler uses ISA to provide commands to the processor, it tells the processor what it needs to do. ISA specifies what storage mechanisms are available to the processor and how to access them. ISA consists of addressing modes, instructions, native data types, registers, memory architecture, interrupt, and exception handling, and external I/O. As the complexity and demand for higher number of operations ISA has also become more complex and advanced.

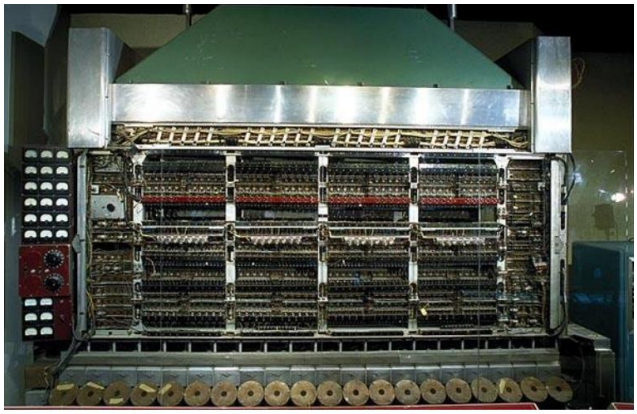

Figure 3IAS Machine designed by John von Newuman -Princeton NJ 1952

Accumulator Architecture – 1949

Computers in early 1950s had Accumulator Architecture, it had a single register for arithmetic operations which acted as source to store both data and result of the operation, system will take the instruction to execute an operation on data stored at a certain address and store the resulting data at the same location examples of such architecture are EDSAC in 1949, IBM 701 in 1953 and DEC PDP-8 in 1965.

General Purpose Register Architecture – 1963

Accumulator architecture evolved in to multiple register architecture, the extra registers were included to store array references for data transfer instructions and also act as separate accumulators to perform arithmetic operations, to point to top of stack or act as program counter. IBM 360 developed in 1964 used this type of architecture where one operand was stored in memory. Another variation of this architecture was used in CDC 6600(1963) where operand is stored in the register.

Compact code and Stack Architectures – 1960's

Limited memory and inefficient compilers led to the development of Compact Code stack architectures, operands were stored in stack and once the arithmetic operation is completed result is place back on the stack. This design effectively simplified compilers and eliminated register allocation hence dependency on memory space.

High Level Language Computer Architectures – 1960's

In early computer systems operating systems were written in assembly language, which had a huge code density making it difficult to work with and lead to the need for making hardware more like programming language. Source code is directly executable with minimal processing supporting constructs such as procedure calls, loop control, and complex addressing

modes, allowing data structure and array accesses to be combined into single instructions. This type of architecture was quite popular during 1960s and 1970 but gave way to other architectures.

Vector processors-1970's

This type of architecture dominated the era between 1970- 1980, it utilized the power a a large number os simple processors being controlled by a single master CPU. A single common instruction is fed to all the ALUs with different data points achieving massive parallel processing. It was mainly implemented in super computers.

Complex Instruction Set Computer Architecture – 1980's

Problems with having the compilers and programs had to be coded by hand one instruction at a time there was need to program more and more instructions. Instead of writing, many instructions CISC utilized single instructions that can execute several low level instructions such as one complex instruction will do 3 simple tasks namely - load from memory, do an arithmetic operation and store the result back in memory. Many of these instructions are complicated combination instructions such as loops. In general, more complicated or specialized instructions are inefficient in hardware, and in a typically CISC architecture the best performance can be obtained by using only the most simple instructions from the ISA. Some of the best examples of CISC ISAs are the Motorola 68k and Intel x86 architectures.

RISC Architecture – 1980's

Reduced instruction set computing was realized in the late 1970s by IBM but faced commercial uncertainty. Stanford University and University of California, Berkeley mainly developed concept of RISC architecture and brought it to forefront. It brought high performance with simplified instruction set, it uses small highly optimized set of instructions. Most RISC architectures have 32 bits fixed-length instructions and a simple encoding. It has simplified fetch, decode, and issue logic. Memory is only accessed through specific instruction and not accessible to other

instructions. By reducing, the number of address modes and breaking down multi-cycle instructions into multiple single-cycle instructions several advantages were realized, such as easier to write compilers, fast running programs that did simple operations and control over clock rate. While research at Stanford lead to the development of Microprocessor without Interlocked Pipeline Stages (MIPS) architecture, research at UCB pathed way for Scalable processor architecture (SPARC) developed by Microsystems in 1987.

Very long instruction word (VLIW)

VLIW architecture uses instruction level parallelism where program can execute multiple instructions simultaneously. It's a logical extension of the RISC architecture it can execute more than one instruction at a time (a characteristic referred to as superscalar). This type of architecture leverage software's capability to manage complexity instead of the hardware, leading to smaller, cheaper, and power efficient processors

Explicitly parallel instruction computing (EPIC)-1997

By early 1990s it was realized that RISC were reaching their limits and started looking at VLIW approach to achieve higher performance. EPIC is a 64-bit microprocessor instruction set designed by Intel and Hewlett Packard. While CISC and RICS architectures used 32 bit registers, branch prediction, memory latency and implicit parallelism, EPIC utilizes more efficient approach to ISA using speculative loading, predication and explicit parallelism. Intel Architecture-64 is an EPIC based architecture.

Micro Architecture

Microarchitecture, or computer organization describes how a particular processor will implement the ISA. Microarchitecture defines the constituent parts of the processor and how they

interact and work together for meaning full computation. It deals with the design of data path for efficient computing. Most modern MA uses pipeline architecture design that allows for multiple instruction to overlap in execution. Another important aspect of MA is the execution units which include arithmetic logic units (ALU), floating point units(FPU) load/store units, branch perdition etc. Progression in MA has been focused on to have the capability to execute more instructions in parallel, thus increasing the effective execution speed of a program. MA has seen advancement in Pipeline depth, number of pipelines, cache size, silicon area, peak power, execution ordering, bus widths, ALU widths over the period of time. Few of the techniques that have been implemented in recent times to maximize the benefits of MA are discussed below.

Instruction pipelining

Legacy processor designs would execute instructions in sequence which cause a lot of resource left idle at any one step, instruction decoding resource would be idle to wait for execution and so on. Data Pipelines allow for a number of instructions to be executed by the processor at the same time. RISC ISA had made it easier and simpler to design and construct these pipelines.

Cache

As the technological advances in manufacturing and microchips MA has started increasing add more cache which is simply very fast memory that can be accessed right on the processor.

Branch prediction and Speculative execution

As clock speed increase so does the stalling waiting for data to arrive, branch prediction makes an educated guess on which branch should be taken and by watching the past branches makes choices able to predict accurately to which branch to take.

Superscalar

With ever increasing logic gates and transistors on a chip it became possible to execute multiple instructions simultaneously know as superscalar.

Out-of-order execution

Increased cache reduced the time to wait for the data to be fetched from memory, there might be some instruction in the program whose data is already available in the cache allows for the process to be executed on the data in cache out of turn.

Multiprocessing, multithreading and simultaneously multithreading

Multithreading allows for multiple programs to be executed simultaneously in the same cycle.

Transport triggered architecture (TTA)

In this type of architecture CPU directly controls the internal transport buses of a processor.

Register Transfer Level

RTL is an abstraction layer that models the simulation of digital signals between hardware registers and logical operations performed on them. Based on synchronous logic and contains three primary pieces namely, registers which hold state information, combinatorial logic which defines the nest state inputs and clocks that control when the state changes.

Recent advances have been in programmable storage, 3D memory and co-processor technologies. Over the last couple of decades growing disparity between processor and off-chip memory speeds particularly in multicore processor machines has been deterrent to the growth in computing prowess. 3D packaged memory where two or more RAM chips are stacked one on top of each other and connected using ThroughSilicon Vias (TSV) is used to increase the

memory capacity, getting possibility of integrating compute logic directly with one or more memory layers opening up the possibility of near-memory computations. As research in Processing In Memory (PIM) evolves it is inevitable that we will have fully-programmable memory architectures, where application developers can run custom code directly near memory

Similar to programmable SSDs, with 3D packaged memory the provenance collection system running on the CPU will lack visibility into the computations and data transformation that have taken place on the compute logic close to memory, thereby making provenance collection incomplete. Collecting provenance in this computational environment improves the completeness of provenance.

References

Giri, S. (n.d.). Retrieved from academia.edu: https://www.academia.edu/2502719/HISTORY_OF_INSTRUCTION_SET_ARCHITETURES

Groves, F. D. (1995). Brief History of Computer Architecture Evolution and Future Trends. *18th CERN School of Computing* (pp. 147 - 159). Arles, France: CERN.

Ledak, P. (2015, Jan). Retrieved from quora.com: https://www.quora.com/How-much-more-computing-power-does-an-iPhone-6-have-than-Apollo-11-What-is-another-modern-object-I-can-relate-the-same-computing-power-to

Wentzlaff, D. (2013). Department of Electrical Engineering - Princeton University. *Computer Architecture - Introduction, Instruction Set Architecture, and Microcode*.