CA Research Report

During the course of this paper, we are going to explain the fundamental aspects of A computer architecture with its corresponding instruction sets and methods of execution. Different models of architectures go through various changes during the years, but hold the same concept of tackling the execution of a program.this paper focuses on tackling the important aspects of how a processor works and some modifications made to minimize overhead and possible faults and redundancy.

Processors:

A processor is a combination of circuits, mainly transistors and latches, that receive and compute the basic instructions that operate a computer. A processor's main job is to execute the following commands: fetch, decode, execute and writeback. Processors have many different types and different type of classifications as they can be grouped according to functionalities, models or other attributes.

The basic elements of a processor include:

- ALU (Arithmetic Logic Unit): the part responsible for carrying out the operations on the operands (the quantity on which an operation is to be done) received from the instructions.
- FPU (Floating Point Unit): a type of coprocessor, which is a special type of processor, that is in charge of executing mathematical operations quickly and efficiently
- Registers: Registers can be thought of as an inventory where data is stored; they hold instructions that as well as supply operands to ALU and store the results of the computations.
- Cache: It is a type of memory (to be discussed more thoroughly in this paper) and its presence aids in severely cutting down the time required to access the main memory.[1]

Types of processors:

- Uniprocessors
    - Single
- Multiprocessors
    - Dual core
    - Triple core

- Quad core

Uni processors and Multiprocessors are types of processors, classified according the number of cores in a processor. The single/uniprocessor, as its name suggests, contains only one core. Any other non single processor is considered a multiprocessor and it can have from to 2 to any amount of cores. Currently the highest number of cores in a processor is Intel's 72-core x86 Knights Landing CPU that is used for supercomputing.[2][3]

A processor can also have multiple cores on one chip, this is known as a microprocessor. One of the advantages of microprocessors is its speed; For instance, a Microprocessor with 3 gigahertz, is capable of performing 3 billion tasks per second.
A special type of microprocessors is the coprocessor; its main functionality is to aid the capabilities of the primary processor. In a MIPS architecture (to be discussed later), there can be more than one coprocessor and each coprocessor has a certain functionality.[4][5]

Processor Architecture

All these processors are used in a processor architecture. A processor architecture, or a computer architecture, is a specification describing how hardware and software technologies interact to create a computer platform or system. It is basically a set of rules stating how computer software and hardware are joined together and interact to make a computer work.[6]

Types of Processor Architecture:

- Harvard Architecture
- Von Neumann Architecture

Harvard

The Harvard architecture type has machine instructions and data in separate memory units that are connected by different busses. Due to this, computers designed with Harvard architecture are able to run a program and access data independently; data transfers and instruction fetches can be performed at the same time. As per this strict separation, this architecture is considered very complicated. The processing unit can complete an instruction in one cycle if appropriate pipelining strategies are in place. Pipelining is an implementation technique that is used when multiple instructions overlap in execution. The computer pipeline is divided instages and each stage completes a part of an instruction in parallel.

Von Neumann

In the Von Neumann architecture type, date and instructions are both stored in the same memory unit and there is only a single bus. If a Von Neumann machine wants to perform an

operation on some data in memory, it first has to move the data across the bus into the CPU. After the computation is done, it moves the output of the computation back to the memory across the same bus. This means the processing unit needs two clock cycles to complete an instruction; one cycle for fetching and decoding the instruction from the memory, another for retrieving the data from the memory. Due to the fact that the bus is needed to move data regularly in both directions, the amount of data the bus can transfer at one time determines how fast the Von Neumann architecture can be as well as its throughput. The Von Neumann bottleneck occurs when data taken in or out of memory must wait while the current memory operation is completed. That is, if the processor just completed a computation and is ready to perform the next, it has to write the finished computation into memory, which requires the bus, before it can fetch new data out of memory, which also requires the bus.

Harvard vs Von Neumann

Due to Von Neumann having only one memory unit and one bus, its design is considered relatively simple and the development of the ALU. Hence, the production cost is relatively low. Harvard, on the other hand, has a high development cost and high production cost because of the two busses and two memory units used. In Von Neumann, an instruction is executed in two cycles but only one in Harvard. Also, in Harvard, data transfers and instructions can be performed simultaneously but this is not possible in Von Neumann.[7][8][9]

Memory and Cache

Memories are classified into different levels, two of which are the primary and the secondary storage units (a cache is typically in-between these two levels).

Primary storage unit:

Any primary storage unit typically holds data that is directly accessed by the computer's CPU (Central Processing Unit). An example of which is the RAM (Random Access Memory), which is a high-speed, volatile storage unit that is accessed with minimal delay since the RAM is connected directly to the CPU via a “Memory bus”. Primary storage therefore typically stores the data loaded by active programs.[10]

Secondary storage unit:

A hard disk drive is an example of a secondary storage; a hard disk drive is not directly connected to the CPU so instead of accessing data in a direct manner, any secondary storage unit sends and receives data via an I/O (Input/output) bus, which may pass through a cache or another memory level before being processed.[10]

Types of primary storage:

1. RAM: a RAM operates on electric current, hence, after the computer has been shut down, every data stored on the RAM is deleted. There are two types of RAM:
    a. DRAM: A dynamic random access memory is used in the main memory and has a simple design that is composed of a few capacitors and transistors. Due to its power leakage, this type of memory needs power refresh circuitries.
    b. SRAM: A static random access memory which is faster, smaller and consumes lower power in comparison to the DRAM, but is more expensive (SRAM's on the contrary are used in the cache).
2. ROM: a Read Only Memory retains its contents even if the device loses power. You cannot change the data on it, but rather just read it. ROM is a more reliable form of storage and it will often boot instructions and other mission-critical data.

3. Cache memory, also called CPU memory, is a high-speed static random access memory (SRAM) that a computer microprocessor can access more quickly than it can access regular random access memory (RAM) since it is physically closer to the processor than the RAM. A cache is mainly used to reduce execution time of an instruction/program. A cache operates by storing some information if the main memory. Due to its location that is closer to the processor than the main memory, fetching data stored in the cache is done in a faster fashion than fetching it directly from the main memory. If a record/data is not stored in the cache, it is retrieved from the main memory, passed to the CPU for executing and then saved in the Cache so that the next time it is fetched it would require less time. Caches are classified into levels (typically 3). A relatively easy analogy is to compare the types of storage units (primary, secondary, tertiary etc...) with the levels of the cache (level 1, level 2 level 3...). Level 1 cache is the smallest of all the cache levels (typically 2KB-64KB) and thus, the f fastest of them all. Data is searched for in Level 1 cache, if not found, computer searches for the target of interest in level 2 cache, followed by level 3 cache (if it is missed from its previous level).[11][12]

Instruction Set Architectures (ISA)

Instruction Set Architecture is the set of instructions supported by a processor. An ISA serves as the interface between software and hardware. An ISA defines everything a machine language programmer is required to know in order to program a computer. Every ISA is different in terms of what what they define. Its components include the ISA supported data types, the states (registers and memories), their semantics (memory consistency and addressing nodes), the instruction set and the input and output model.[13]

ISAs can be classified in many ways. One of the classifications is according to architectural complexity: RISC and CISC.

A complex instruction set computer (CISC):

CISC has many specialised instructions and each instruction can execute several low-level operations such as loading and storing from memory or arithmetic operations. Due to single instructions having the capability of executing several low level operations, this instruction set lead to smaller program sizes, hence less RAM is needed and it the frequency of the calls made to main memory is lower. It also supports high level languages and assembly language.

However, CISC does have its drawbacks. Due a single instruction having many operations, there is a great overhead in decoding instruction which in turn causes slower execution time. There is also a case where complex architecture lead to having problems when attempting to enhance it. It can be significantly harder to improve it using a complex instruction so instead, a set of simple instructions are used in place. Also due to its complex design, there are several cases where not all the registers are used which is considered a form of inefficiency. One of the main issues of CISC is the fact that instructions are so unique complexity-wise which makes them not general enough to be used frequently. The low frequency use of these instructions suggests that their presence is not entirely justified.[14]

A well known example of CISC is “x86” which is based on the Intel 8086 microprocessor and its 8088 variant which was introduced in 1978 as a fully 16-bit extension of Intel's 8-bit 8080 microprocessor. The 32-bit extension came into existence in 1985 and later the 64-bit in 2003; these are all considered generations of the architecture. The x86 architecture is a variable instruction length with emphasis on backward compatibility and its type is register-memory. Byte-addressing is enabled and words are stored in memory with little-endian byte order. Memory access to unaligned addresses is allowed for all valid word sizes. A dedicated floating point processor, a microprocessor, with 80-bit internal registers was developed for the original 8086.[15]

A reduced instruction set computer (RISC):

RISC is a simplicity-based design with a fixed size of instructions. It is relatively much simpler than CISC and smaller with only 32 registers. Of those 32 registers, 3 are fixed for operands for every arithmetic operation. While the instruction length is fixed, instruction format changes depending on the kinds of instructions. These instructions are considered simple and general. Another defining trait of RISC is that accessibility of memory where only specific instructions are allowed to access memory but not most instructions. Its fixed instruction length as well as as its simple encoding simplifies fetching, decoding and issue logic to a great extent. However this greatly decreases its code density which can be a considered a drawback in embedded computing. RISC Architectures are mainly used for cellular telephones and tablet

computers and its continued success in this fields greatly justifies its use as opposed to its minimal use for desktops as the x86 remains the dominant processor type.

Examples of RISC include ARM and MIPS. ARM, introduced in 1985, is mainly used in smartphones and tablet computers. Since the 21st century, it has dominated the market for low end mobile systems. It has 32 and 64 bit generations that have been introduced in 2011 and its type is register-register.

MIPS (Microprocessor without Interlocked Pipelined Stages), is developed by MIPS Computer Systems and it has had multiple versions throughout the years ever since its introduction in 1985 as MIPS I with 5 versions as well as 6 releases for MIPS 32/64. Its latest release is the MIPS32/64 Release 6 in 2014. Its encoding is fixed and its type is register-register. The MIPS architecture also has several extensions each used for a specific purpose. For example, the MIPS-3D is dedicated to common 3D tasks.

Originally, MIPS was designed for general-purpose computing and during the 1980s and 1990s, MIPS processors for personal, workstation, and server computers were used by many companies. Adding to that, several video game consoles such as the Nintendo 64, Sony PlayStation, PlayStation 2, and PlayStation Portable used MIPS processors. Nowadays however, with the diminished use of RISC in desktops, MIPS is generally used in embedded systems such as residential gateways and routers.

- MIPS I: Has thirty-two 32-bit general-purpose registers.
- MIPS II: Removed the load delay slot and added several sets of instructions. For shared-memory multiprocessing, the Synchronize Shared Memory, Load Linked Word, and Store Conditional Word instructions were added.
- MIPS III: Added a support for 64-bit for memory addressing and integer operations. It also added backward compatibility with MIPS II
- MIPS IV: It was designed to mainly improve floating-point performance. To improve access to operands and an indexed addressing mode for Floating Point loads and stores were added.
- MIPS V: It was designed to improve the performance of 3D graphics transformations.
- MIPS32/64: In order to refocus in embedded systems, MIPS32/64 were introduced:
    - MIPS32/64 Release 1: based on MIPS II, added conditional moves, prefetch instructions, and other features from the R4000 and R5000 families of 64-bit processors
    - MIPS32/64 Release 6 added the following instructions:
        - Unconditional branches (BC) and branch-and-link (BALC) with a 26-bit offset.
        - Conditional branch on zero/non-zero with a 21-bit offset.
        - Full set of branch-and-link which compare a register against zero.

        - Instructions to load 16-bit immediates at bit position 16, 32 or 48, allowing to easily generate large constants and many many more were added.

    - MIPS32/64 Release 6 removed the following instructions for being not frequently used:
        - Some conditional moves.
        - Branch likely instructions (deprecated in previous releases).
        - Integer overflow trapping instructions with 16-bit immediate and other instructions were removed.[15]

Datapath

Datapath is a collection of functional units such as arithmetic logic units or multipliers, that perform data processing operations, registers, and buses. A datapath needs a control unit (CU) to aid in the processing of instructions. Datapaths could be combined together to form larger datapaths using multiplexers.

The datapath determines the instruction classes and formats in an ISA. accordingly, different instruction formats require different datapaths. Some of which are explained as follows[16]:

- Load/Store datapath: the load/store datapath uses instructions where the offset denotes a memory address relative to the memory segment where the PC(program counter) currently points to. In this datapath, an instruction is fetched from the instruction memory, executed and (read from the register values) and the result is applied to the data memory as an address. In order to compute the memory address, the MIPS ISA specification for example, says that we have to sign-extend the 16-bit offset to a 32-bit signed value. This is done using the sign extender. A load/store datapath performs the following action in order of succession:
    - Register Access: takes input from the register, to implement the instruction, data, or address fetch step of the fetch-decode-execute cycle.
    - Memory address calculation: combines the base address and the offset to produce an actual memory address using the help of the ALU and the sign extender
    - Read/Write from/to Memory: takes data from/inputs data to the memory
    - Write into register: puts data into memory and executes the instruction of interest.
- Arithmetic/Logical instructions (R-format datapaths): the datapath for the arithmetic operations is fairly straightforward and requires the register file and the ALU. the ALU accepts an input from the register file, executes the instruction and the register file is written to by the ALU result output.
- Branch/Jump datapath: the branch datapath uses instructions where its offset is a 16-bit address used for determining the branch address using relative addressing.By *branching to the desired location*, the ALU adds a sign-extended

offset to the program counter. The offset is shifted left 2 bits to allow for word addressing. Thus, to branch to the target address, the lower 26 bits of the PC are replaced with the lower 26 bits of the instruction shifted left 2 bits.instruction in the branch/jump datapath follows the following order to fully execute an instruction:

- Register Access: takes input from the register, to implement the instruction, data, or address fetch step of the fetch-decode-execute cycle.

- Calculate Branch Target: the ALU calculates the branch target address to be ready for the branch if it is taken.
- Evaluate Branch Condition and Jump: the condition is evaluated and the decision of branching or not is taken accordingly. If the condition is met, the program jumps to the calculated branch target address and the PC is incremented to fetch the next instruction.

Instruction Pipelining

instruction pipelining is a method commonly used for implementing instruction-level parallelism(the execution of several instructions in a parallel manner) within a single processor. Pipelining keeps the processor busy with some instruction by dividing incoming instructions into a series of sequential steps.In a pipelined datapath, the known datapath is subdivided into several stages: instruction fetch, Instruction decode, execution and address calculation, memory access, and writeback. An easy analogy to visualize pipelining is to consider each instruction being executed independently. After the completion of each stage, registers are needed to store the information produced from the previous cycle.[17]

The use of pipelining allows for reducing the cycle time of a processor and increasing the instruction throughput.If pipelining is used, the CPU Arithmetic logic unit can be designed faster, but more complex.

Reference list:

1. https://whatis.techtarget.com/definition/processor
2. https://www.pchardware.co.uk/processors.php
3. https://www.extremetech.com/extreme/171678-intel-unveils-72-core-x86-knights-landing-cpu-for-exascale-supercomputing
4. http://scanftree.com/microprocessor/Advantages-&-Disadvantages
5. Soubra, H. Computer System Architecture Lecture 4
6. https://www.computersciencedegreehub.com/faq/what-is-computer-architecture/
7. https://www.microcontrollertips.com/whats-the-difference-between-von-neumann-and-harvard-architectures/
8. http://web.cs.iastate.edu/~prabhu/Tutorial/PIPELINE/pipe_title.html
9. http://www.differencebetween.net/technology/difference-between-von-neumann-and-harvard-architecture/
10. https://pc.net/helpcenter/answers/primary_and_secondary_storage
11. https://blog.storagecraft.com/primary-vs-secondary-storage-difference/
12. https://specialties.bayt.com/en/specialties/q/305444/what-is-the-difference-between-l1-l2-and-l3-cache-memory/
13. Pugh, Emerson W.; Johnson, Lyle R.; Palmer, John H. (1991). *IBM's 360 and Early 370 Systems*. MIT Press. ISBN 0-262-16123-0.
14. https://web.archive.org/web/20150714180129/http://www.ijirs.com/vol2_issue-6/59.pdf
15. https://booksite.elsevier.com/9780124077263/downloads/historial%20perspectives/section_4.16.pdf
16. https://www.cise.ufl.edu/~mssz/CompOrg/CDA-proc.html
17. http://www.cs.fsu.edu/~zwang/files/cda3101/Fall2017/Lecture8_cda3101.pdf