Computer System Architecture

CSEN601

Supervisor: Assoc. Prof. Dr. Hassan Soubra

Department: Media Engineering & Technology

Project Report Milestone 1

Team Name: Newbies

Team Members:

Islam Nasr 40-4936

Kariman Hossam 40-1398

Hana Kamal 40-1183

Gasser Khaled 40-18970

Ahmed Salah 40-1316

1. Different Architectures Models of Processors

i. Overview about processors

Any system is composed of a CPU, memory, and the input/output devices. A CPU or a processor is the central component of any computer. It is essential because it is responsible for every single action the computer does. A processor determines many things. For instance, what kind of operating systems can be used and which software packages can run. The cost of any machine depends on how powerful and new the processor is. The way that a processor is designed is called its architecture. A processor's architecture model has a huge effect on its performance. There are some types of architecture models of processors found. The prominent types are Von Neumann, Harvard, RISC, CISC architectures.

ii. Von Neumann Architecture

Von Neumann architecture is all about storing instructions and data in the same memory. The Von Neumann architecture is characterized by the memory, the ALU, the control unit and the input/output devices. All parts are connected by one bus. Firstly, the Von Neumann uses a single processor. As above-mentioned, the Von Neumann uses one memory for both the instructions and data which is called the stored program concept. The memory is addressed linearly which means that there's a single sequential address for each location in the memory. Since the Von Neumann architecture has only one bus, the operations should get scheduled because they cannot get performed at the same time. This concept is referred to as the Von Neumann's bottleneck. Moreover, a program consists of instructions which are executed in order. The sequence of execution can only be altered using conditional or unconditional jumps. The processor will need two clock cycles to execute an instruction. In the first cycle, the processor fetches and decodes an instruction from the memory. In the second cycle, the data is retrieved from the memory. Nowadays, the Von Neumann is the basic architecture for many computers. It is better for desktop computers, laptops and workstations.

iii. Harvard Architecture

The Harvard architecture basically uses two memories; one for the instructions (ROM) and the other for the memory (RAM). Both of the memories can have different sizes. Different buses are used to access data and instructions which makes. Since there are two memories, parallel access is allowed. In other words, instructions and data can be fetched at the same time. Having a separate bus for the instructions made it possible to fetch other instructions while decoding and executing the current one. In Harvard architecture, the processor needs only one clock cycle to execute an instruction since separate buses are used. This model is used primarily for small embedded computers and signal processing.

iv. Von Neumann vs. Harvard

A Von Neumann's design is simple and cheaper as the control unit gets the data and instructions in the same way from the memory at a time. On the other hand, a control unit for two buses is complex and more expensive. In Von Neumann when instructions are stored in the same memory as the data, they can be overwritten due to an error in a program. In contrast, in the Harvard architecture the program cannot write itself. Von Neumann's model memory organization is in the hands of programmers which allows them to utilize the memory's whole capacity. In addition, the space is not wasted as the space in the instruction memory can be used by the data memory and vice versa. While in the Harvard model, the free space in the data memory cannot be used for instructions and vice versa. Parallel execution is not allowed in Von Neumann because instructions are processed in a serial manner but it is allowed in Harvard. Being unable to fetch data and instructions at the same time in Von Neumann makes the speed of execution slower. In Harvard, the processor fetches data and instructions simultaneously; thus, the speed of execution is faster.

v. CISC Architecture

CISC is Complex Instruction Set Computer. Its purpose is to reduce the number of instructions; therefore, the number of clock cycles per

instruction increases. Small number of general-purpose registers are used because instructions operate directly on the memory. More transistors are needed for decoding and the execution time is very high. Since instructions are complex, they are larger than one-word size. CISC has multiple addressing modes; hence, this leads to variable-length instructions. For instance, if an operand is in the memory the length increases as we have to specify the memory address which consumes more bits. Basically, the CISC minimizes the number of instructions and sacrifices the number of clock cycles. The CISC architecture is used in applications like security systems and home automation. System/360, VAX, PDP-11, Motorola 68000 family, AMD and Intel x86 CPUs are examples of CISC microprocessors.

vi. RISC Architecture

RISC is Reduced Instruction Set Computer. It has a few instructions in the instruction set; therefore, operates faster. An instruction will take one clock cycle to get executed. An instruction fits in one-word; hence, all instructions are about the same length. RISC has a large number of registers to prevent having to interact a lot with the memory. The decoding of instructions is simple; therefore, the execution time is less. RISC supports pipelining which is a way that processes instructions more efficiently as it allows simultaneous execution of them. Basically, the RISC increases the number of instructions to reduce the cycles. The RISC architecture is used in applications like video and image processing. Alpha, ARC, ARM, AVR, MIPS, PA-RISC, PIC, Power Architecture, and SPARC are examples of RISC microprocessors.

vii. CISC vs. RISC

The RISC processors have a small set of instructions with few addressing modes. However, the CISC processors have a large set of instructions with many addressing modes. In RISC, an instruction is executed per clock cycle. On the other hand, CISC have some special and other instructions that take more than one clock cycle to get executed. As a result, the execution time is less in RISC than in CISC. Also, the RISC architecture needs more RAM. RISC focuses on software over hardware while CISC focuses on hardware. In RISC

the code size is larger as there is a small instruction set. Many lines of code in RISC can correspond to only one line of code in CISC. In CISC, many transistors are used to store the complex instructions, while in RISC more transistors are spent on memory registers. RISC is a complex compiler design, but CISC is an easy one.

viii. Conclusion

The Von Neumann architecture is actually similar the Harvard model. The only difference is that it uses one bus to transfer data and fetch instructions. This factor limits the performance. Contrarily, the Harvard model uses two memories for the data and instructions. To conclude, the Von Neumann is commonly used over Harvard as it is cheaper to implement. When speed is favored, Harvard is used. Moving on, we cannot differentiate between CISC and RISC architectures because each of them is better in a specific application. The thing that matters is how fast a processor can execute the instructions that it is given and how well do they run.

2- Comparison between different types of Memory and different Cache levels, and their impact on efficiency and complexity.

In computer architecture, memory is the section of the computers' hardware integrated circuit which stores information that's needed for the computing functionalities. Memory is usually made of semiconductors that are silicon-based transistors used mainly in computers but also in numerous digital and electronic devices. Memory is divided into different categories and more sub-categories, mainly volatile and non-Volatile.

Volatile mainly means changeable or unstable, so volatile memory is the type of computer memory that demands power in order to maintain the information that is stored on it. Almost all of the random access memory is volatile except the CMOS RAM used in the BIOS. Primary storage demands very high access read and write speeds which is what volatile memory is typically used for. However, when the user needs to save any data for a long time they will use a permanent volatile memory, such as a hard drive to avoid data loss. RAM is the internal memory of the computer that stores all relevant information that's needed for the moment, like storing data, programs and program results. RAM is divided into 2 two types, Static RAM (SRAM) and Dynamic RAM (DRAM).

The word static indicates that memory retains its content as long as power is supplied. SRAM chips use a matrix of capacitors and no transistors to store the data; it is used as cache memory as it tends to be small and extremely fast.

Dynamic RAM however, not only does it need electricity but it also needs to be continually refreshed in order for it to maintain the data. The DRAM chips are made up of memory cells which are composed of a capacitor and a transistor. DRAM is used for most system memory as it is small and cheap, when compared to SRAM.

Non-volatile memory is the type of computer memory that, unlike volatile memory, maintains its state as well the information whether it is supplied with power or not. Examples of non-volatile memories include ROM (read-only-memory), hard disks, optical disks and flash memory. It can be classified into tradition non-volatile disk storage and storage in non-volatile memory chips like SSDs and EEPROM.

Non-volatile data storage is categorized into 2 categories, mechanically addressed systems: hard disks, magnetic tapes, optical disks) and electrically addressed systems (ROM). Electrically addressed systems are way faster, however they have limited capacity and are much more expensive than mechanically addressed systems which are slower. However, no matter how fast the different types of non-volatile memories get, they will still be slower than volatile type memories.

Electrically addressed memory that consists of semiconductor non-volatile memory is further categorized based on the type of writing mechanism. Programmable read-only memory can be re-programmed after manufacture however they of course require a special kind of programming and cannot be altered inside the target system. Data is stored by physically burning storage sited on the device itself. However, EEPROM which stands for Electronically Erasable Programmable Read Only Memory can be changed multiple times, but the difference is that it can be edited by applying voltage to erase and write to the memory. Both of these types are not easily changed so they can't be a viable usage option for storing data that needs to be altered frequently.

Flash memory is somewhat similar to EEPROM but it differs in the manner that it can only erase one block or "page" at a time. It is a solid-state chip that retains stored data without the need for a power source. Moreover, they have considerably higher capacities than EEPROM, making these chips a viable option for BIOS chips that store some of the operating system information on PCs.

Coming to mechanically addressed systems, they use a system that consists of a contact structure, "head" that reads and writes data on a designated storage medium. Since circuitry layout is not a problem for data density, the size of storage is usually much larger that electrically addressed systems. However, as the access time depends on the physical location of the data on the storage medium itself, mechanically addressed systems tend to be slower for read and write; however they are the best fit for secondary storage usage as they have the best cost per data bit as data that is on the secondary storage doesn't need to be accessed very frequently and doesn't require fast data access times.

Hard disks use a rotating disk to store data, which supports random data access, unlike tapes which store data on a long strip of tape, where data is just a

sequence of bits. Hard disks technology is similar to magnetic tapes, where a magnetic material is layered onto a high precision aluminum disk. The disk head can move to any point on the platter almost instantly compared to tape. With tape, the tape touches the head, but with disk, the head never touches the platter. Disk platters move at up to 7500 cm/s.

A cache, in computing, is a data storing technique that provides the ability to access data or files at a higher speed. They are implemented in software as well as hardware. It serves as an intermediary storage media between primary storage appliance and the hardware device to reduce delay in accessing the data. Cache works in hardware and software performing similar functionalities. In the physical hardware form, it is a small area of internal memory that stores important, frequently accessed data that is expected to be requested by the CPU.

For caches to be cost-effective and to allow for efficient use of data, caches are relatively small. Caches have proven themselves in multiple areas of computing because typical software applications access data with a high degree of locality of reference. These access strategies utilize temporal locality, where data is requested that has been requested recently, and spatial locality where data is requested that is stored physically nearby data that has already been requested.

The use of throughput incurs a significant latency for access where prediction and prefetching that can guess where future reads will come from and be ready for requests before they even happen, and if done correctly, latency is bypassed.

Using cache also increases throughput, by assembling multiple fine grain transfers into larger more efficient requests. In the case of DRAM for instance, this can be done by having a bigger data bus.

Hardware implements cache as a block of memory that stores data that is likely to be used again temporarily. CPUs tend to use the cache very often. So does web browsers. When a user/process wants to access data on the memory, that is probably available on the cache, it first check the cache, if the data found was containing the same tag as the requested data then the data in the cache is use instead. This is called a cache hit. The other situation when a data is queried and the found data doesn’t have the same tag, this is called a miss. During a miss scenario, the data that is currently resent there gets replaces with the new

requested data, to be ready for the next access. However, different algorithms will decide on this replacement strategy differently.

Small memory on or close to the CPU tends to operate faster than larger main memory. CPUs since the 1980s mostly use one or more different types of caches with different levels. For example, Graphical Processing Unit Cache (GPU Cache), Translation lookaside buffer, disk cache and web cache.ss

A very common example of caching is in a Web browser, where a website's HTML, images, CSS, Javascript, etc is cached locally so that a page will load faster after its first hit.

3- Comparison between different Instruction Set Architectures (ISAs) in terms of structure as well as on decisions on including or excluding a number of instructions.

Different instruction set architectures are known nowadays. We will consider in our research some of the main ISA languages. The following ISA will be considered:

-ARM (Advanced RISC Machines)

-X86 Architecture

-MIPS ISA Architecture

ARM Instruction Set Architecture:

ARM it was developed by British company and it licenses ARM to other companies. Instruction set architecture has six major versions of instructions. The first 3 versions of ARM included original 26-bit architecture. After that they introduced a 32-bit architecture. The ARM ISA has 31 internal registers (from x0-x30) , in addition to a stack pointer and a zero register. The stack pointer and the zero register both are used as register number 31, depending on the instruction it is used as either a stack pointer or as a zero register. The return address is stored in the x30 register. When a branch is made using instructions like BL or BLX the link address is hold in the link register which is register 14. The registers are 64-bit registers, the lower 32 bits are accessed by accessing w0-w31. Usually if operations will use the lower 32 bits only it clears the upper 32 bits. ARM ISA also contains 32 registers for 128-bit vector registers or floating point registers. If these register are used as vectors different arrangements are supported which maybe 64 bits or 128 bits. This speeds up the multiple loop iterations. ARM have about 442 instructions which are organized in table of four levels which include main encoding, instruction group, decode group and instruction.

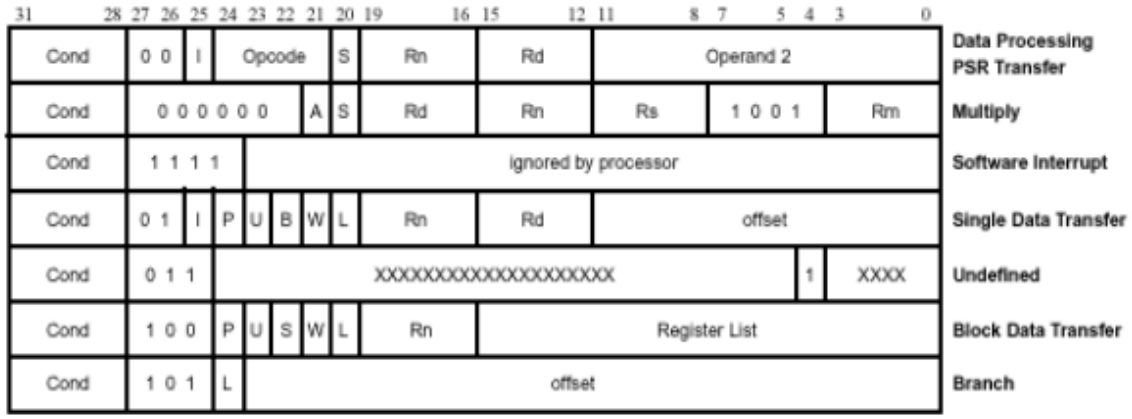
The ARM provides control over the shifter and the ALU in almost all data-processing instruction which makes the use of the ALU and the shifter as high as possible. It also provides auto-increment/decrement addressing mode, load and

store multiple instructions to maximize the throughput of the data. Finally it maximizes execution throughput using conditional execution of approximately all of the instructions.

The ARM have seven types of exceptions supported which are the following: Reset, attempted execution of an undefined instruction, call to operating system is made by software interrupt instructions (SWI), prefetch abort instruction for memory abort. Data access memory abort, normal interrupt (IRQ) and fast interrupt (FIQ).

Version 3 of the ARM processor mainly added two new processor modes in order to make it possible to use exception instructions effectively. Version 4 added half-word load and store instructions, instructions to load half-word and sign extended bytes. Instruction to transfer to thumb state. Version 5 finally added instruction to count the leading zeros which allow efficient integer division and interrupt routines with priorities. Version 5 also added instructions for software breakpoint. It also tightened the definition of the setting of flags by multiply instruction.

ARM used Thumb Instructions which enables strong arithmetic operations, long branch range and large address space. The following figure shows how ARM ISA divides the instruction set.



X86 Architecture:

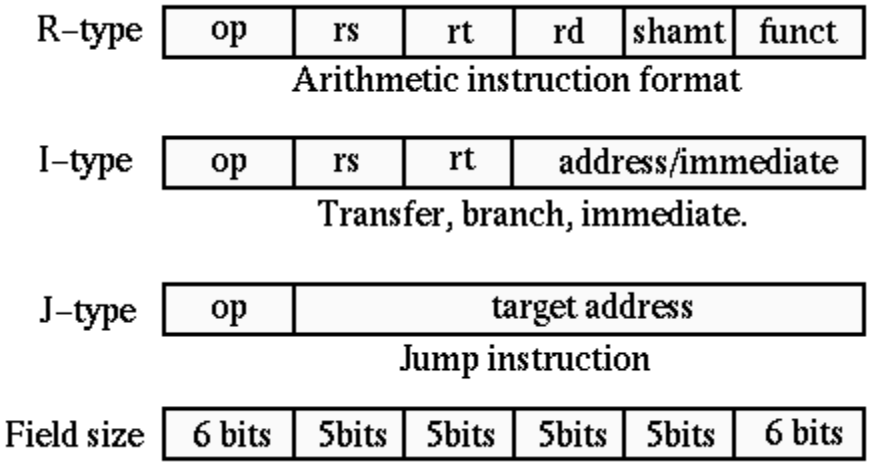
X86 was developed by Intel Corporation for the computer processors. X86 defines how processors can handle different instructions passed from software programs and operating system and then executes it. X86 instruction started as a 16-bit 8086 processor in the late 70's. Nowadays it is supporting 32 bit processor compatibles. X86 ISA is a CISC ISA (Complex Instruction Set Architecture). X86 instruction was up to 15 bytes instructions. Because it is CISC ISA it has a widely varying instruction length. Because it is a CISC instructions many instructions are being added frequently. The x86-64 allowed two modes of operations the 64-bit mode and the compatibility mode. It also added a new mode of paging which is the 4-level paging mode. This allowed supporting large amount of physical and virtual memory than before.

X86 have data transfer instruction (like MOV with different uses depending on the inputs, MOVZX zero extension which extends upper half of destination cells to zeros, XCHG which exchanges the value of two operand, etc…). X86 have arithmetic instructions (like INCrement, DECrement, ADD,SUB, NEG instructions, etc…). X86 have Jump and loop instructions (like JMP, LOOP [example or instruction], etc…). It also have conditional processing instructions (like status flag, not instruction, and instruction, or instruction, etc…). X86 have Test instructions, CMP instruction (compare), and the conditions. Also it have conditional jump instructions, conditional loops instruction, conditional structure instructions, shift and rotate instructions, multiplying and division instructions and implementing arithmetic expressions.

It have 8 General purpose registers which are the accumulator register (AX), counter register (CX), data register (DX), base register (BX), stack pointer register (SP), stack base pointer register (BP), source index register (SI), and destination index register. The x86 architecture is little-endian; means that the least significant bytes are written first.

MIPS Architecture:

MIPS is a simple RISC architecture ISA. It changed over time introducing more instructions and deleting some instructions which found that they were of no use. It was developed by MIPS Computer System Incorporation, which was founded in 1984 by group of Stanford University researchers which included Chriss Rowen and John Hennessey. MIPS stands for microprocessor without interlock pipelined stages. MIPS consisted of 32 registers. The $zero register is used for default value 0, $at register which is register number 1 which is reserved for use by the assembler. Then we have the $v0-$v1 registers for the outputs. $a0-$a3 registers are for the procedures (passing parameters). $t0-$t9 are used for local storage and it is saved for the caller. $s0-$s7 is used for local storage also but which is less likely to be overwritten and more likely to be used than $t registers. $k0-$k1 registers are reserved for the kernel operations, $gp is for the global pointer, $sp is the stack pointer, $fp is the frame pointer and the $ra is the register saving the return address. MIPS was released in five versions I,II,III,IV,V. Early releases of MIPS where 32 bits but then 64 bits versions where released. MIPS are used in embedded systems. MIPS CPU instructions are divided into arithmetic instructions, logical instructions, data transfer, conditional and jump instructions. It is of 3 types R-type, I-type, J-type instructions. Which are divided as in the following picture:



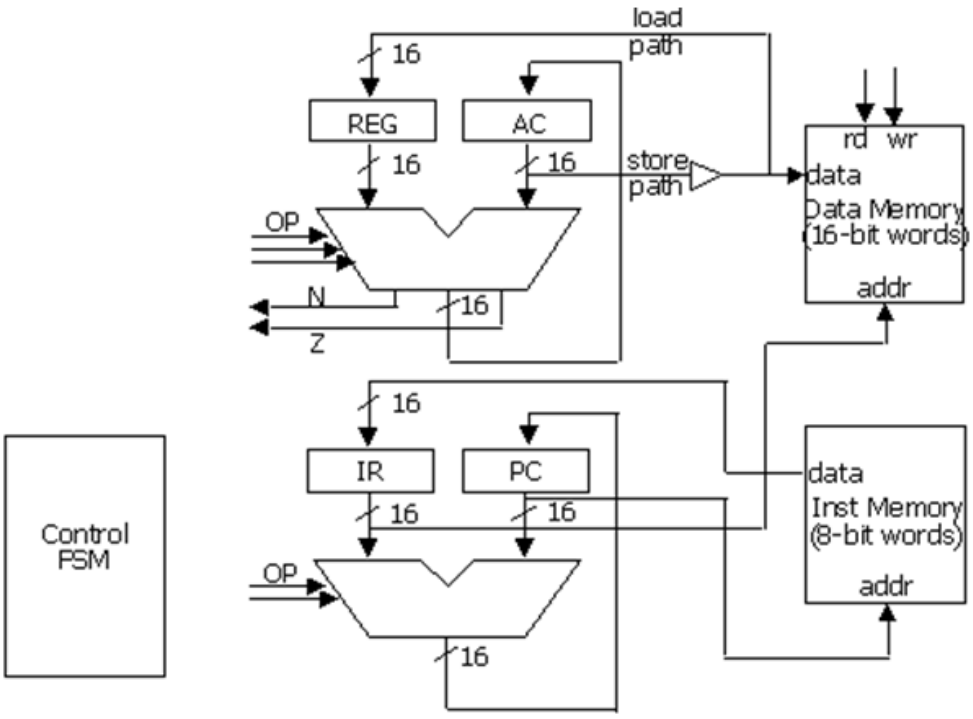
4- Comparison between data path of different Computer Architectures:

The data path is also known to be the data section which is one of the two sections of the CPU (the data section and the control section). The data section contains the ALU and the registers. The data path utilizes the control signal supplied by the control section. The data is transferred between the ALU and the registers and from one register to another. The data path executes specific functions on the data. We can also create a larger data path by joining one or more data paths by using multiplexers. In order to enhance the effectiveness and reduce the power consumption we can engrave the data path on fabrics and make them reconfigurable which hence allows the data paths to be configured at runtime.

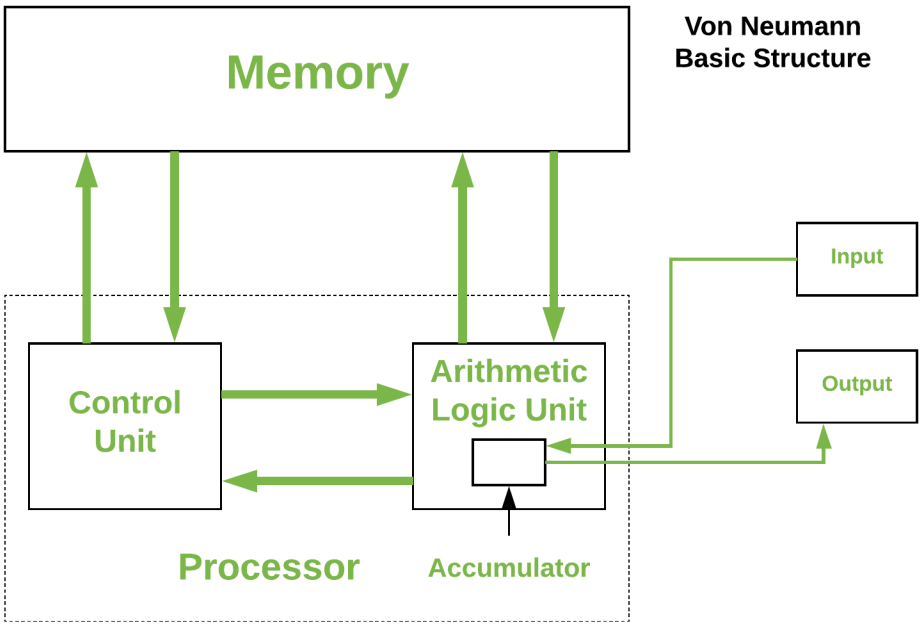
There are different types of computer architectures as discussed before, two major types are

1- *Harvard computer architecture*
2- *Von Neumann computer architecture*

The data path of the Harvard computer architecture:



The data path of Von Neumann computer architecture:

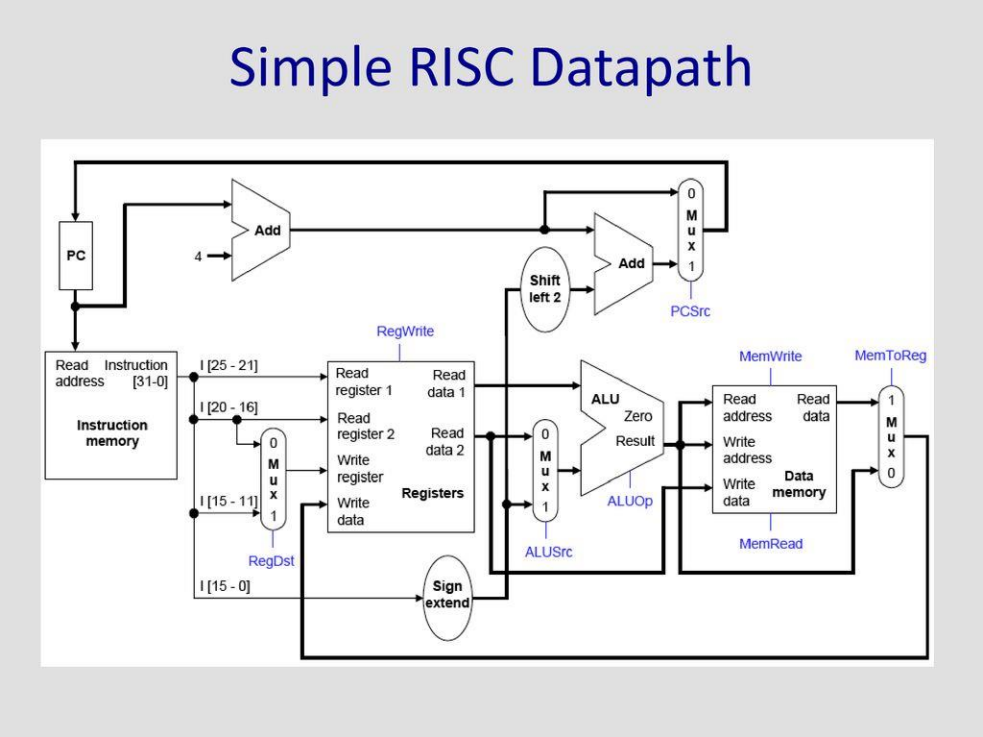


As illustrated in the figures above the von Neumann data path has only one bus which hence contains one ALU and one Memory for both data and instructions. On the other hand, the Harvard data path has two buses which hence contain two ALUs and two memories (the data memory and the instructions memory) each separated.

The von Neumann processor takes two clock cycles to fully perform a single instruction, in the first clock cycle the instruction is fetched from the memory and decoded while in the second clock cycle the processor gets the required data from the memory. Therefore, pipelining instructions is not applicable in the von Neumann computer architecture.

However, the Harvard processor takes only one clock cycle to fully perform a single instruction as it has two memories. Therefore, pipelining instructions is applicable in the Harvard computer architecture

The data path of the RISC processor:



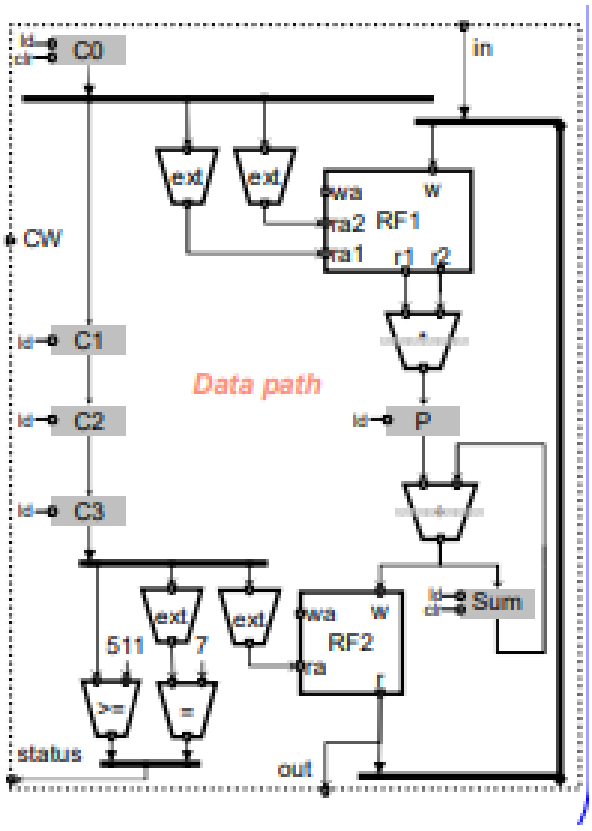
Since that the data paths of the RISC processors are very thorough as seen in the figure above as the operations in every stage are straightforward and clear, therefore, a single instruction is performed in no more than one clock cycle which includes the fetch and the decode making it easy to construct an instruction pipeline for the RISC processors. The design had a five stage execution instruction pipeline. The five stages include:

1- *Instruction fetch*
2- *Instruction decode*
3- *Execute*
4- *Memory access*
5- *Writeback*

On the other hand, in the CISC processor the instructions' format and size fluctuate meaning that the only way to know that you are done fetching an instruction is when you start decoding it. This would result in a variation of the number of clock cycles taken by each instruction to be performed as some instructions may take only one cycle to be fetched and decoded and others may take a lot more than one cycle and this result in a difficulty in the instruction pipeline process of the CISC processor.

The data path of NISC processor:

Shortcut for No Instruction Set Computer, it provides the advantages where it defines a simpler controller, it adds on performance level, and it is finally easier to design. The NISC instruction path is generated automatically for applications. Each word which is a control word in NISC can perform up to 3 regular RISC instructions.



Above we have an example of a NISC data path customized for Discrete Cosine transform implementation. Data path in NISC can be extended without limitations by adding reconfigurable data path. NISC data also does not limit parallelism unlike RISC.

References:

https://hal.archives-ouvertes.fr/hal-01864517/document

http://www.toves.org/books/arm/

http://vision.gel.ulaval.ca/~jflalonde/cours/1001/h17/docs/arm-instructionset.pdf

http://www.cecs.uci.edu/technical_report/TR04-12.pdf

https://searchstorage.techtarget.com/definition/cache-memory

https://www.ijraset.com/fileserve.php?FID=5135

https://www.quora.com/What-is-the-difference-between-the-Von-Neumann-architecture-and-the-Harvard-architecture

https://www.microcontrollertips.com/whats-the-difference-between-von-neumann-and-harvard-architectures/

https://www.cse.wustl.edu/~lu/cse467s/slides/dsp.pdf

http://paginapessoal.utfpr.edu.br/gortan/aoc/transparencias/t01_intro_aoc/literatura/Wiki_Risc_Cisc_Harvard_VonNeumann.pdf/at_download/file

http://ijarcsms.com/docs/paper/volume4/issue7/V4I7-0014.pdf