**NAME:** NWACHUKWU EMMANUEL IFECHUKWU

**MATRIC NO:** 17/SCI01/052

**1951-1960 -** The long-lasting languages evolved over this decade are Fortran and COBOL and laid down the foundation for further evolution. Assembly Language- the first in this list is a low-level programming language and works closer to hardware and specific to computer architecture.

**1961-1970** - The most important language came out during this period is SNOBOL the successor of COBOL developed at AT & T Bell Laboratories by David J. Farber, Ralph E. Grisworld, and Ivan P. Polonsky. Simula is another important language since Simula 67 developed in this series was first Object Oriented Language (OOL). By the end of this decade, PASCAL came out as another promising programming language introducing structured programming.

**1971-1980** – This decade introduced C which is still being widely used to develop programs specific to hardware. It’s one of the most popular languages and still in use by major industries. SQL emerged as the first languages specific to manage data. C++ was released by the closure of this decade adding OOPS to its predecessor C.

**1981-1990** – MATLAB, FoxPro and Objective C evolved as major languages during this decade where MATLAB was widely used for algorithm and visualizations. FoxPro is a mix of programs having DBMS concepts and it’s also Object Oriented Programming Language. Object C is an object-oriented programming language and majorly used by Apple for OS X and IOS development.

**1991-2000** – This decade introduced very interesting languages starting with Python which is widely used as an alternate of PHP and Java. Java was released by the mid of this decade and soon became the de-facto standard of the enterprise with its inbuilt security features.

Ruby is another important language released in 1995 and attracted programmers to give their hands on it. The Rails framework opens up doors to website and application development by simplifying repetitive tasks.

PHP was also released during the same year i.e. 1995 and became the default choice for web development due to the easy learning curve. JavaScript and ActionScript have also emerged as frontend languages focusing on the UI part.

**2001-2010** – C# was introduced by Microsoft at the beginning of this decade and became the programmers’ choice for Desktop and Web application development. It’s object-oriented and the system is somewhat closer to Java. Go and Rust were released by the end of this decade focusing on system programming and competent to each other. Golang is majorly used for developing Microservices. Learning Rust is a bit difficult whereas Go is easy to learn.

**RISC (reduced instruction set computer):** RISC is a type of microprocessor architecture that utilizes a small, highly-optimized set of instruction, rather than a more specialized set of instructions often found in other types of architectures. The main distinguishing feature of RISC is that the instruction set is optimized for a highly regular instruction pipeline flow. Characteristics- Most RISC architectures have fixed-length instructions (commonly 32 bits) and a simple encoding, which simplifies fetch, decode, and issue logic considerably. For any given level of general performance, a RISC chip will typically have a far fewer transistors dedicated to the core logic which originally allowed designers to increase the size of the register et and increase internal parallelism. Simple instructions are used in RISC architecture.

* RISC helps and supports few simple data types and synthesize complex data types.
* RISC utilizes simple addressing modes and fixed length instructions for pipelining.
* RISC permits any register to use in any context.
* One Cycle Execution Time.
* The amount of work that a computer can perform is reduced by separating “LOAD” and “STORE” instructions.
* RISC contains Large Number of Register in order to prevent various number of interactions with memory.

**ADVANTAGES**

* RISC (Reduced Instruction Set Computing) architecture has a set of instruction, so high-level language compilers can produce more efficient code.
* It allows freedom of using the space on microprocessors because of its simplicity.
* Many RISC processors use the registers for passing arguments and holding the local variables.
* RISC functions use only a few parameters, and the RISC processors cannot use the call instructions, and therefore, use a fixed length instruction which is easy to pipeline.
* The speed of the operation can be maximized and the execution time can minimized.

**DISADVANTAGES**

* Mostly, the performance of the RISC processors depends on the programmer or compiler as the knowledge of the compiler plays a vital role while changing the CISC code to a RISC code.
* While rearranging the CISC code to a RISC code, termed as a code expansion, will increase the size. And, the quality of this code expansion will again depend on the compiler, and also on the machine’s instruction set.
* The first level cache of the RISC processor is also a disadvantage of the RISC, in which these processors have large memory caches on the chip itself. For feeding the instructions, they require very fast memory systems.

**CISC (Complex Instruction Set Computer):** The CISC approach attempts to minimize the number of instructions per program, sacrificing the number of cycles per instruction. Computers based on the CISC architecture are designed to decrease the memory cost. Because, the large programs need more storage, thus increasing the memory cost and large memory becomes more expensive. To solve these problems, the number of instructions per program can be reduced by embedding the number of operations in a single instruction, thereby making the instructions more complex.

**Examples of CISC:**

IBM370/168 – It was introduced in 1970. CISC design is a 32 bit processor and four 64-bit floating point registers.

VAX 11/780 – CISC design is a 32-bit processor and it supports many numbers of addressing modes and machine instructions which is from Digital Equipment Corporation.

Intel 80486 – It was launched in the year 1989 and it is a CISC processor, which has instructions varying lengths from 1 to 11 and it will have 235 instructions.

**Characteristics of CISC:**

* Instruction decoding logic will be complex.
* One instruction is required to support multiple addressing modes.
* Less chip space is enough for general purpose registers for the instructions that are operated directly on memory.
* Various CISC designs are set up two special registers for the stack pointer, handling interrupts, etc.
* MUL is referred to as a “complex instruction” and requires the programmer for storing functions.

**ADVANTAGES**

* Microprogramming is easy assembly language to implement, and less expensive than hard wiring a control unit.
* The ease of microcoding new instruction allowed designers to make CISC machines upwardly compatible.
* As each instruction became more accomplished, fewer instructions could be used to implement a given task.

**DISADVANTAGES**

* The performance of the machine slows down due to the amount of clock time taken by different instructions will be dissimilar.
* Only 20% of the existing instructions is used in a typical programming event, even though there are various specialized instruction in reality which are not even used frequently.
* The conditional codes are set by the CISC instructions as a side effect of each instruction which takes time for this setting- and, as the subsequent instruction changes the condition code bits- so, the compiler has to examine the condition code bits before this happens.

**VLIW (Very Long Instruction Word) architecture**: VLIW refers to instruction set architectures designed to exploit Instruction Level Parallelism (ILP). VLIW describes a computer processing architecture in which a language compiler or pre-processor breaks program instruction down into basic operations that can be performed by the processor in parallel (that is, at the same time). These operations are put into a very long instructions word which the processor can then take apart without further analysis, handing each operation to an appropriate functional unit.