Computer Language Compiler
Next Generation Hardware pre Quantum

There has been an excellent progressive path of computer language inventions for fifty years. As someone who appreciates llvm, the stringent languages that implement it typically remain within their language specification and rarely offer forgiving code. After surveying the many computers languages I know, plus ones I have read about, the inception of a language always seems to give an effort to simplifying problem description . In recent years object oriented component technologies, data storage technology, and parallelism have propelled market performance.

As a multi feature computing language with modern hardware support one must never forget the simplicity of c++, the mother. It seems that all computings bare a heart of it. The expression format is usual for any language. Kernig and Richie were good visionaries to offer such tools amongst COBOL. Forth offering a library approach along with Prolog as AI barely scratched the surface. With the necessity of database formation a problem hand coding intelligence and NLP led ways to machine learning.

Within the c++ language, the base keywords offer boolean logic, memory management, data types, functions, objects, conditional iteration are sound digital logic constructs. Yet c++ lacks in the base standard many advances of next generation languages. A person would have to implement the feature. Often, these languages are crafted in c++.

Some nice features can be added to a c++ compiler target using preprocessing in the build chain. This essentially is a program which translates the source to a complete c++ source file. The technique is offered but rarely accepted in industry product approaches as a concept. Often developers wish to write code that works and have it done quickly.

A focus of c++ programming also is that objects are to be named well. Yet because good names occupy a namespace already one must find a variation in the name to have a usable system. String for example. Many times this is overplayed and programming concepts remain quirky and oddball within the language due to the name used. Often what really simplifies language is specific to what is being described in a section of the algorithm

For example a set of coordinates with vertice indices would be nicely described as a spreadsheet grid interface with numbers. Or a modeling program is used to generate the test data.

While this information is inlined within the program binary the development pattern, program data intelligence, and data input stream can be designed and tested as a single source or included file is essential to research development. As an integrated development environment facility, patterns of source editing can be very concise to datatype, record data structure, and also easily adapt the program to use external data.

- Visual interface language for gui
- Data structure hard typing with inline parser that is executed as preprocess.
- Auto tuple on return
- Auto data type
- Database
- Cgi web html css library
- Multi threaded data types
- Ide support of object linking
- Cross process objects using stl
- Audio playback recording

Browser API design as a more efficient model.

The internal operating system functions should also be adapted through a llvm installation process. The design of llvm based objects, are ones that resolve communication at compile time. A major key concept to the summarization logic are parameters, data structure format and calling convention. At the OS layer, this mingles to large scale designed systems. For example, the spiral of code yarn from Text to display. At even the xcb layer, the necessary parameter calling convention to drivers is intermingled with critical data management. The memory operations are distinct for all aspects of video system state mode and device type.

The hardware abstraction layer HAL that forms a soft concept from the OS API to the device driver, is a catch and dispatch to appropriate hardware, port data format or software emulation. Typically programs query capabilities of hardware and perform state management that directs the program's logic.

With an llvm enabled architecture, truly native network compiling is possible as a configurable pipeline build service for os image. GCC the other large older mainstay has assembly features for the few lower level bootup of Intel microprocessors. The necessity of an ordered flush mnemonic exists such that planning without supporting previous generation is inconsequential. Compiler technology and encoding technology is only manifest more complex by unordered sets. The SOC and algorithmic processing of CISC offers inroads for embedded computing devices. With high clock frequencies of crystals (modern clock forms?) the granularity of instructions offers enhancements necessary to next generation subsystems. Typically the front side bus is used for expansion in these types of legacy upgrades. Yet with onboard CPU to parallel processors, the communication and task offload to specialized multicore processors can save time

As an gui application platform, the W3C naming convention summarizes years of technology growth that were implemented not as straight forward. The process of refinement comes after usage.

Video buffer markup data structure is the compounding facet of rendering technology not yet available. The font render, low level layout and image composite makes graphical interface projection accessible to a dynamic market. That is a major technical reliance of the OS API is object or actor rendering. OpenVG is the market progression before this as stepping stones must pebble. Yet with a capable hardware vector process with stage compositing, additions may be accumulated. System on chip designs featuring open format inclusion for processor sound processing, and the central processing unit to utilize the gui rendering video system can redesign technology usage and accessibility. OpenVG acceleration is currently seen obtusely in research forms using opengl.

CPU, protected ring memory management, port, interrupt processing, DMA, and chip based permanent storage offer a bounty of kernel code rearchiture. That is focused hardening of the computing pipeline for software support. Types of enhancements algorithmically such as data queueing for high volume transaction have been also laid out in esoteric Linux kernels such as DragonFly. The speed of Gentoo is also capturing, especially without the GUI.

The Wayland architecture speeds support for a plethora of devices and offers a hardware compositor. As a hardened concept with unrolled logic can be applied to functions necessary. The CppUX design offers a discrete and more focused codebase for UI. At the OS Layer, compositing, and event handling can be incorporated as a application API designed like opengl.

Hardware designs of modern day are heavily based in legacy technology even just down to Von Neumann boot discovery. The platform that applications and algorithms execute on are sorely matched due to legacy technology concepts. Needless reallocation of blocks and copies by not incorporating indirect alias with block copy instructions that allow skips or linear jumps. The data communication layer to storage managers rely on distinct data type meanings that are often not handled natively. The capability to enhance the ALU for numerical processing without including legacy floating type cleans and simplifies the design for easier types of multi core parallelism. That is strictly computation for a newer type of miniaturized core. Register usage within the CPU is often the greatest speed increase as types of calling convention on the platform use it for parameter communication. Thread switching also causes constant copy of the context when an increase in register memory can ease. Having the ability for the CPU to access specific registers is noted within the internal instruction deciding mechanism. By simplifying the registers to be bank oriented, a program or core thread might never have the need to be paused as registers are accessed more due to cache usage. A thread context switch is merely a register bank switch for the time sharing algorithm. Less spills, less context copy offers faster system performance.

The capability of large ram storage offers that searches performed need analysis to determine inclusion within hardware.

Standard Library in processor ring protected cache constant to the hardware layer for program execution can offer reduction of RAM swap.

Hardware parallel memory management to provide waitstate serial logical access.

Coprocessor independent memory access for read using multilayer chip design.

Transaction based storage in parallel within consumed memory block signaling for round buffer management. Database enhanced disk storage systems for next generation record access. The integer parallel alu for vector based match. As an indexed table with necessary parameters in registers for the instruction, The problem is within the pattern match logic which has to be allowed to be complex for good advancement.

The indirect pointer heap provides no dirty copy block with a memory queue operating in a security ring. The progression from one ring to the next is a system function of data lifecycle to permanent storage. The disk adaptor can assume uninterrupted independent read access of a large multi bank image from RAM using a memory coprocessor. with register memory for bus transfer.

Hardware memory queue processing in round buffer can supply multimedia and event system well. The instruction counter and program memory balanced to provide serialization of coprocessor dispatch logic Rules of decoding can provide metrics which note advancement synchronization.

Storage key and hash calculations algorithmically a block process for coprocessor instruction. Most data base storage systems rely on these numeric values for order. A process of data storage which incorporated advanced indexing of format specific characteristics can be furthered by co processors in text, image, optical recognition, audio, and video.

A very intelligent algorithm is often supplied with a database and subsequent analysis of the data. The basic tiny block problem for these concepts to work is based upon a comparison routine. A logical less than operation which returns a boolean value. The other functions are deduced from the existence I remember reading somewhere. The cpp sort is the standard Library sort.

The test function is magnified to a large set of processor functions. Jump and jump to a comparison is a very detailed mnemonic when encoded has numerous variations based on data type and size. Programs rely on these functions to pass

control to other logic conditionally. The address of the next instruction is also within the payload of the instruction. RIP or relative instruction pointer are additions or subtractions from the current IP.

A new question to solve is how to make the comparison faster. The function has to be run multiple times across the dataset. Many times data hashing is used.