Chapter 1 : Computer architecture - Wikipedia

*The structure of the PUMA computer systems; overview and the central processor [Ralph Grishman] on theinnatdunvilla.com *FREE* shipping on qualifying offers. This is a reproduction of a book published before*

History[edit] The first documented computer architecture was in the correspondence between Charles Babbage and Ada Lovelace , describing the analytical engine. When building the computer Z1 in , Konrad Zuse described in two patent applications for his future projects that machine instructions could be stored in the same storage used for data, i. Johnson and Frederick P. Buchholz, by writing, [8] Computer architecture, like other architecture, is the art of determining the needs of the user of a structure and then designing to meet those needs as effectively as possible within economic and technological constraints. As of the s, new computer architectures are typically "built", tested, and tweakedâ€”inside some other computer architecture in a computer architecture simulator ; or inside a FPGA as a soft microprocessor ; or bothâ€”before committing to the final hardware form. The ISA defines the machine code that a processor reads and acts upon as well as the word size , memory address modes , processor registers , and data type. Microarchitecture , or computer organization describes how a particular processor will implement the ISA. System Design includes all of the other hardware components within a computing system. Data processing other than the CPU, such as direct memory access DMA Other issues such as virtualization , multiprocessing , and software features. There are other types of computer architecture. A smart assembler may convert an abstract assembly language common to a group of machines into slightly different machine language for different implementations Programmer Visible Macroarchitecture: The hardware functions that a microprocessor should provide to a hardware platform, e. Also, messages that the processor should emit so that external caches can be invalidated emptied. Pin architecture functions are more flexible than ISA functions because external hardware can adapt to new encodings, or change from a pin to a message. The term "architecture" fits, because the functions must be provided for compatible systems, even if the detailed method changes. Definition[edit] The purpose is to design a computer that maximizes performance while keeping power consumption in check, costs low relative to the amount of expected performance, and is also very reliable. For this, many aspects are to be considered which includes instruction set design, functional organization, logic design, and implementation. The implementation involves integrated circuit design, packaging, power, and cooling. Optimization of the design requires familiarity with compilers, operating systems to logic design, and packaging. Please help improve this section by adding citations to reliable sources. Unsourced material may be challenged and removed. March Main article: A processor only understands instructions encoded in some numerical fashion, usually as binary numbers. Software tools, such as compilers , translate those high level languages into instructions that the processor can understand. Besides instructions, the ISA defines items in the computer that are available to a programâ€”e. Instructions locate these available items with register indexes or names and memory addressing modes. The ISA of a computer is usually described in a small instruction manual, which describes how the instructions are encoded. Also, it may define short vaguely mnemonic names for the instructions. The names can be recognized by a software development tool called an assembler. An assembler is a computer program that translates a human-readable form of the ISA into a computer-readable form. Disassemblers are also widely available, usually in debuggers and software programs to isolate and correct malfunctions in binary computer programs. ISAs vary in quality and completeness. A good ISA compromises between programmer convenience how easy the code is to understand , size of the code how much code is required to do a specific action , cost of the computer to interpret the instructions more complexity means more hardware needed to decode and execute the instructions , and speed of the computer with more complex decoding hardware comes longer decode time. Memory organization defines how instructions interact with the memory, and how memory interacts with itself. During design emulation software emulators can run programs written in a proposed instruction set. Modern emulators can measure size, cost, and speed to determine if a particular ISA is meeting its goals. Microarchitecture Computer organization helps optimize performance-based products. For example, software engineers need to know the processing power of processors. They may need to optimize software in order to gain the most performance for the lowest price. For example, in a SD card, the designers might need to arrange the card so that the most data can be processed in the fastest possible way. Computer organization also helps plan the selection of a processor for a particular project. Multimedia projects may need very rapid data access, while virtual machines may need fast interrupts. Sometimes certain tasks need additional components as well. For example, a computer capable of running a virtual machine needs virtual memory hardware so that the memory of different virtual computers can be kept separated. Computer organization and features also affect power consumption and processor cost. Implementation Once an instruction set and micro-architecture are designed, a practical machine must be developed. This design process is called the implementation. Implementation is usually not considered architectural design, but rather hardware design engineering. Implementation can be further broken down into several steps: Logic Implementation designs the circuits required at a logic gate level Circuit Implementation does transistor -level designs of basic elements gates, multiplexers, latches etc. Physical Implementation draws physical circuits. The different circuit components are placed in a chip floorplan or on a board and the wires connecting them are created. Design Validation tests the computer as a whole to see if it works in all situations and all timings. Once the design validation process starts, the design at the logic level are tested using logic emulators. However, this is usually too slow to run realistic test. Most hobby projects stop at this stage. The final step is to test prototype integrated circuits. Integrated circuits may require several redesigns to fix problems. Design goals[edit] The exact form of a computer system depends on the constraints and goals. Computer architectures usually trade off standards, power versus performance, cost, memory capacity, latency latency is the amount of time that it takes for information from one node to travel to the source and throughput. Sometimes other considerations, such as features, size, weight, reliability, and expandability are also factors. The most common scheme does an in depth power analysis and figures out how to keep power consumption low, while maintaining adequate performance. Performance[edit] Modern computer performance is often described in IPC instructions per cycle. This measures the efficiency of the architecture at any clock frequency. Since a faster rate can make a faster computer, this is a useful measurement. Older computers had IPC counts as low as 0. Simple modern processors easily reach near 1. Superscalar processors may reach three to five IPC by executing several instructions per clock cycle. Counting machine language instructions would be misleading because they can do varying amounts of work in different ISAs. This refers to the cycles per second of the main clock of the CPU. However, this metric is somewhat misleading, as a machine with a higher clock rate may not necessarily have greater performance. As a result, manufacturers have moved away from clock speed as a measure of performance. Other factors influence speed, such as the mix of functional units , bus speeds, available memory, and the type and order of instructions in the programs. There are two main types of speed: Latency is the time between the start of a process and its completion. Throughput is the amount of work done per unit time. Interrupt latency is the guaranteed maximum response time of the system to an electronic event like when the disk drive finishes moving some data. Performance is affected by a very wide range of design choices â€” for example, pipelining a processor usually makes latency worse, but makes throughput better. Computers that control machinery usually need low interrupt latencies. These computers operate in a real-time environment and fail if an operation is not completed in a specified amount of time. For example, computer-controlled anti-lock brakes must begin braking within a predictable, short time after the brake pedal is sensed or else failure of the brake will occur. Benchmarking takes all these factors into account by measuring the time a computer takes to run through a series of test programs. Often the measured machines split on different measures. For example, one system might handle scientific applications quickly, while another might render video games more smoothly. Low-power electronics Power efficiency is another important measurement in modern computers. A higher power efficiency can often be traded for lower speed or higher cost. Modern circuits have less power required per transistor as the number of transistors per chip grows. However the number of transistors per chip is starting to increase at a slower rate. Therefore, power efficiency is starting to become as important, if not more important than fitting more and more transistors into a single chip. Recent processor designs have shown this emphasis as they put more focus on power efficiency rather than cramming as many transistors into a single chip as possible. Shifts in market demand[edit] Increases in publicly released refresh rates have grown slowly over the past few years, with respect to vast leaps in power consumption reduction and miniaturization demand. This has led to a new demand for longer battery life and reductions in size due to the mobile technology being produced at a greater rate.

DOWNLOAD PDF THE STRUCTURE OF THE PUMA COMPUTER SYSTEMS; OVERVIEW AND THE CENTRAL PROCESSOR.

Chapter 2 : Ralph Grishman (Author of Computational Linguistics)

Project objectives and history are summarized first. Then the overall system structure is briefly described. Next, the central processor is discussed in some detail. Then the microprograming language of the PUMA is described; this chapter can serve as an introduction for the microprogramer. Finally.

The design flow for an SoC aims to develop this hardware and software at the same time, also known as architectural co-design. Most SoCs are developed from pre-qualified hardware component IP core specifications for the hardware elements and execution units , collectively "blocks", described above, together with software device drivers that may control their operation. Of particular importance are the protocol stacks that drive industry-standard interfaces like USB. The hardware blocks are put together using computer-aided design tools, specifically electronic design automation tools; the software modules are integrated using a software integrated development environment. Once the architecture of the SoC has been defined, any new hardware elements are written in an abstract hardware description language termed register transfer level RTL which defines the circuit behavior, or synthesized into RTL from a high level language through high-level synthesis. These elements are connected together in a hardware description language to create the full SoC design. The logic specified to connect these components and convert between possibly different interfaces provided by different vendors is called glue logic. Functional verification and Signoff electronic design automation Chips are verified for logical correctness before being sent to a semiconductor foundry. Bugs found in the verification stage are reported to the designer. Traditionally, engineers have employed simulation acceleration, emulation or prototyping on reprogrammable hardware to verify and debug hardware and software for SoC designs prior to the finalization of the design, known as tape-out. This is used to debug hardware, firmware and software interactions across multiple FPGAs with capabilities similar to a logic analyzer. In parallel, the hardware elements are grouped and passed through a process of logic synthesis , during which performance constraints, such as operational frequency and expected signal delays, are applied. This generates an output known as a netlist describing the design as a physical circuit and its interconnections. These netlists are combined with the glue logic connecting the components to produce the schematic description of the SoC as a circuit which can be printed onto a chip. This process is known as place and route and precedes tape-out in the event that the SoCs are produced as application-specific integrated circuits ASIC. Optimization goals[edit] Systems-on-chip must optimize power use , area on die , communication, positioning for locality between modular units and other factors. Optimization is necessarily a design goal of systems-on-chip. If optimization was not necessary, the engineers would use a multi-chip module architecture without accounting for the area utilization, power consumption or performance of the system to the same extent. Common optimization targets for system-on-chip designs follow, with explanations of each. In general, optimizing any of these quantities may be a hard combinatorial optimization problem, and can indeed be NP-hard fairly easily. Therefore, sophisticated optimization algorithms are often required and it may be practical to use approximation algorithms or heuristics in some cases. Additionally, most SoC designs contain multiple variables to optimize simultaneously , so Pareto efficient solutions are sought after in SoC design. Oftentimes the goals of optimizing some of these quantities are directly at odds, further adding complexity to design optimization of systems-on-chip and introducing trade-offs in system design.

Chapter 3 : Full text of "The structure of the PUMA computer systems; overview and the central processor"

Excerpt from The Structure of the Puma Computer System: Overview and the Central Processor In our system, as in the Control Data , separate processors have been provided for executing user programs and managing peripherals.

Chapter 4 : What is a CPU - Central Processing Unit? Webopedia Definition

DOWNLOAD PDF THE STRUCTURE OF THE PUMA COMPUTER SYSTEMS; OVERVIEW AND THE CENTRAL PROCESSOR.

Gross Structure of the PUMA Computer System 6 4. Centr al processor structure As we noted earlier, this project was prompted by develop- ments in digital integrated circuit technology. The specific family of circuits we have used in our central processor is 10, series ECL (emitter-coupled logic).

Chapter 5 : System on a chip - Wikipedia

You can read The Structure of the Puma Computer Systems Overview And the Central Processor by Ralph Grishman in our library for absolutely free. Read various fiction books with us in our e-reader.

Chapter 6 : Ralph Grishman: used books, rare books and new books @ theinnatdunvilla.com

The Structure of the Puma Computer Systems; Overview and the Central Processor by Ralph Grishman starting at $ The Structure of the Puma Computer Systems; Overview and the Central Processor has 2 available editions to buy at Alibris.