The Challenges of Lecture Delivery of Arm X86, Cisc and Risc in the Teaching of Coursecsc303 (Computer Architecture) in the University of Jos, Nigeria: an Overview

Stephen MALLO, Jr

Department of Computer Science, Faculty of Natural Science, University of Jos, Nigeria

doi: 10.51505/IJAEMR.2022.7212 URL: http://dx.doi.org/10.51505/IJAEMR.2022.7212

Abstract

The trending definition of the computer architecture course involves the teaching/delivery of instruction set architecture design, micro architecture design, logic design, and implementation. Historically, the RISC vs. CISC wars raged in the 1980s when chip area and processor design complexity were the primary constraints and desktops and servers exclusively dominated the computing landscape (Patterson D.A & Ditzel D.R, 1980). Today and particularly in developing economies like Nigeria, energy and power pose the primary constraints to architecture design and the computing landscape is significantly different occasioned by growth in tablets and Smartphone's running ARM (a RISC ISA) is surpassing that of desktops and laptops running x86 (a CISCISA). Furthermore, the traditionally low-power ARM ISA is entering the high-performance server market, while the traditionally high-performance x86 ISA is entering the mobile low-power device market.ICT has given rise to a host of legal and ethical issues and challenges in the use of ICT for education. All array of both teaching and technical staff as well as students need to know to a reasonable extent about the issues and challenges in the use of ICT for education (Ogbomo. E.F, 2011). It has been severally been suggested in various academic climes of the need for teachers and students to be above reproach as succinctly put by Crossley. S.A. & McNamara, D.S, 2021 'in understanding the basic issues (effectiveness, cost, equity, and sustainability), as well as the challenges (infrastructure related challenges, capacity building challenges, challenges related to financing the cost of ICT use, to mention but few) surrounding the use of ICT in education and then apply those issues as principles in practice'.

The essence of this study is to attempt presenting the appropriate methods in which ARM X86, CISC and RISC can be taught with ease for students in Nigerian Universities to the effect that Institutions lack adequate number of computers and relevant ICT accessories and/or soft and hardwires in laboratories. The methods which was used initially was making use of Projectors for the students in their class rooms and the use of microphone speakers to teach the students in addition to having to cope with extended lectures periods as a result of downtimes. The hindrances notwithstanding, when this method was initiated students had a better theoretical understanding of ARM X86, CISC and RISC because as in agreement with the submission of Bhandarkar. D&Clark. W.D, 1991; in order to achieve results better comprehension and quicker understanding by Students, videos were deployed as lectures were on-going to augment the prospects and constraints of the teaching and delivery of computer architecture. When various methods (Wireless LAN (WLAN), computers/laptops, and other technologies were taken into cognizance, in the teaching and practical delivery of computer architecture, it was observed that the students' comprehension and understanding of the Reduced Instruction Sets of Computers and the Complex Instruction Sets of Computers showed remarkable improvement.

Keywords: RISC (Reduced Instruction Set of Computers), CISC (Complex Instruction Sets of Computers), ICT, Computer Architecture, ISA, Instruction Sets, x86 ARM, Micro Architecture.

Introduction

The First Draft of a Report on the EDVAC (Electronic Discrete Variable Automatic Computer), a 1945 paper by John von Neumann that described a logical element organization. ISA design, notably RISC vs. CISC, must be taken into consideration. When chip area and processor design complexity were the principal restraints in the 1980s and 1990s, ISA was a major worry (D. A. Pattersonetal, 2013). Computer architecture is a set of rules and procedures used in computer engineering to explain the functioning, organization, and implementation of computer systems. Some definitions of architecture refer to a computer's capabilities and programming paradigm rather than a specific implementation. It's debatable if the discussion was resolved on technical grounds. Regardless, both businesses thrived in the 1980s and 1990s. The ARM ISA (a RISC ISA) has dominated mobile and low-power embedded computing during the last decade, whereas the x86 ISA (a CISCISA) has dominated desktops and servers. Perhaps the most powerful influence on today's educational scene is technology. Many school districts are supporting higher levels of technology in the classroom by providing gear like tablets and PCs, improving internet connectivity, and creating computer literacy programs for both instructors and students. Although most instructors recognize the value of educational technology, they often find it difficult to integrate new technologies smoothly and effectively. Technological integration provides considerable obstacles to educators at all levels of school systems, from the acquisition of new technology equipment to the adaption of curricula and teaching practices to accommodate new educational resources. Depending on how one interacts with a computer system, there are various perspective points from which one might observe it. The concept of abstraction is critical for digging into the specifics that are required from a certain point of view (Johnson et al 2021). Depending on the user, computer systems can be viewed from many angles. The instruction set architecture (ISA) is a useful abstraction used by programmers to comprehend the processor's fundamental mechanics.ISA is a standard that defines the logic of a processor, as well as its personality. The ISA specifies the type of instructions as well as their meaning. If the semantics provided to those instructions are right, the chip manufacturer will have an easier time completing the processor's physical design implementation. At the ISA level, there are two types of processors: RISC and CISC (Sivarama. P & Dandamudi, 2016).

CISC computers use the strategy of doing more with each and every instruction, which increases their complexity. As a result, CISC processors are associated with a vast number of addressing modes. As a result, the instruction's execution time and length vary greatly. Also because CISC technique is rather sophisticated, it is being phased out in favor of the Reduced Instruction Set Computer (RISC) (Colwell. C.R & Hitchcock. Y. E, 1985). RISC does not imply that there are fewer instructions; rather, it suggests that all of the instructions follow the same simple and defined syntax. Inefficient processors are mostly caused by variations in instruction length. RISC architectures use a basic instruction format in addition to a fixed-length instruction size. The opcode and source operands fields in an instruction have fixed boundaries. This enables effective decoding and scheduling of instructions contained in the data. Other advantages of the RISC include easy addressing modes, register-to-register interactions (except for load and store instructions), fixed register lengths, simple operations, and a huge register set. MIPS Computer Systems Inc. created a RISC-based microprocessor architecture called MIPS. Fixed-length instructions, a simple decoded instruction format, memory accesses limited to load and store instructions, a hardwired control unit, a large general purpose register file, and operations executing within the microprocessor's registers are all part of the MIPS RISC microprocessor's architecture (Harris D.M & .Harris S.L, 2011). The problem, according to the literature on RISC-based processors, was to create a processor capable of operating quickly and executing even more instructions using the simplest of architectures and instruction sets. The RISC topology that would solve this problem was the MIPS architecture, which was a specific form of RISC architecture. As a result, this paper is single-cycle.

Computer architecture is a set of rules and procedures used in computer engineering to explain the functioning, organization, and implementation of computer systems. Some definitions of architecture refer to a computer's capabilities and programming paradigm rather than a specific implementation. (Alvan. C, 2021). In other words, computer architecture encompasses the design of instruction set architecture, microarchitecture, logic design, and implementation (John. H, & David. P, 2021). The first documented computer architecture can be found in Charles Babbage and Ada Lovelace's correspondence, which describes the analytical engine. In two patent applications for future projects, Konrad Zuse detailed how machine instructions may be stored in the same storage as data, i.e. the stored-program concept, when he built the computer Z1 in 1936. (Williams. F. C, 2009) John von Neumann's 1945 paper, First Draft of a Report on the EDVAC, which described an organization of logical elements; and Alan Turing's more detailed Proposed Electronic Calculator for the Automatic Computing Engine, which cited John von Neumann's paper, are two other early and important examples. The term "architecture" first appeared in computer literature in 1959, thanks to the efforts of Lyle R. Johnson and Frederick P. Brooks, Jr. of IBM's main research center's Machine Organization group. Johnson was given the opportunity to write a confidential research communication for Los Alamos National Laboratory regarding the Stretch, an IBM-developed supercomputer (at the time known as Los Alamos Scientific Laboratory). He stated that his description of formats, instruction kinds, hardware settings, and performance enhancements were at the level of "system architecture," a phrase that appeared more helpful than "machine organization" to express the degree of detail for discussing the luxuriously decorated computer (Lyle. J, 1960)."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," Brooks, a Stretch designer, wrote in Chapter 2 of a book called Planning a Computer System: Project Stretch (Werner. B, 1962). Brooks moved on to work on the IBM System/360 (now known as the IBM zSeries) computer line, where "architecture" became a term that defined "what the user needed to know." Later on, computer users began to use the phrase in a variety of less formal ways(Hans. D.H, 2004). The first computer designs were sketched out on paper and then incorporated into the final hardware. Later, computer architecture prototypes were physically manufactured in the form of transistor–transistor logic (TTL) computer, tested, and refined before committing to the final hardware form, such as the prototypes of the 6800 and the PA-RISC. Before committing to the final hardware form, new computer architectures are frequently "made," tested, and tweaked—within another computer architecture in a computer architecture simulator; or inside an FPGA as a soft microprocessor; or both—as of the 1990s (Schmalz. M.S, 2017).

The subcategories of computer architecture disciplines are organized into three major subcategories (Hennessy. L.J, & Patterson A.D, 2021). The instruction set architecture (ISA) specifies the word size, memory address modes, processor registers, and data type that a processor reads and acts on. Microarchitecture, commonly referred to as "computer organization," explains how a processor will implement the ISA (Phillip A.L, 2001). The size of a computer's CPU cache, for example, is a separate issue from the ISA (standard **Systems design)** which involves all other hardware components of a computing system, such as data processing that is not performed by the CPU (e.g., direct memory access), virtualization, and multiprocessing. Other computer architecture technologies exist. The following technologies are employed by larger businesses like Intel and are predicted to account for 1% of all computer architecture in 2002 (Hennessy L.J, 2002): Architectural layers that are more abstract than microarchitecture are referred to as macroarchitecture. Architecture of the assembly instruction set: A smart assembler can turn an abstract assembly language that is shared by a group of machines into a somewhat different machine language for distinct implementations. Programmer-visible macroarchitecture: higher-level language tools such as compilers may define a consistent interface or contract to programmers using them, abstracting differences between underlying ISA, UISA, and microarchitectures. For example, the C, C++, or Java standards define different programmer-visible macroarchitectures. **Microcode**: Microcode is software that converts instructions into instructions that may be executed on a chip. It wraps the hardware and presents a desired version of the hardware's instruction set interface. (Brinkley. S.J & Kollar. C.S, 1985). Chip designers have a variety of alternatives with this instruction translation facility: For example, a new better version of the chip can employ microcode to offer the exact same instruction set as the previous chip version, allowing all software written for that instruction set to execute on the new device without modification. Microcode, for example, can provide many instruction sets for the same underlying processor, allowing it to run a greater range of software. **UISA**: User Instruction Set Architecture refers to one of three subsets of the RISC CPU instructions provided by **PowerPC** RISC Processors. The UISA subset contains the RISC instructions that application developers are interested in. The other two subsets are the VEA (Virtual Environment Architecture) and OEA (Operating Environment Architecture) instructions, which are used by virtualization system developers and Operation System developers, respectively. (Bradford, F, 2005). **Pin architecture**: A microprocessor's hardware functions that should be provided to a hardware platform, such as the x86 pins A20M, FERR/IGNNE, or FLUSH. Also, messages that the processor should broadcast in order to invalidate external caches (emptied). Because external hardware can adapt to new encodings or move from a pin to a message, pin architecture functions are more adaptable than ISA functions. Even if the detailed approach changes, the term "architecture" fits because the functions must be offered for compatible systems.

Computer architecture is concerned with balancing a computer system's performance, efficiency, cost, and reliability. The balance of these competing elements can be illustrated using the example of instruction set architecture. This is because a single instruction can express some higher-level abstraction, more complicated instruction sets allow programmers to build more space-efficient programs such as the x 86 Loop instructions (Null. L, 2019). Longer and more complex instructions, on the other hand, take the processor longer to decode and can be more costly to implement. When instructions interact in unanticipated ways, the additional complexity of a big instruction set also offers more space for unreliability. Integrated circuit design, packaging, power, and cooling are all part of the process. To optimize the design, individuals will need to be knowledgeable with compilers, operating systems, logic design, and packaging (Milo.M, 2017). The interface between the computer's software and hardware is known as the instruction set architecture (ISA), and it may also be thought of as the programmer's view of the machine. Computers do not understand high-level programming languages such as Java, C++, or most programming languages used. A processor can only read instructions that are encoded numerically, typically as binary digits. Compilers and other software tools convert high-level languages into instructions that the processor can understand. The ISA describes objects in the computer that are available to a program, such as data types, registers, addressing modes, and memory, in addition to instructions. Instructions use register indexes (or names) and memory addressing modes to find these elements. A computer's ISA is typically detailed in a tiny instruction manual that explains how instructions are encoded. It may also define brief but ambiguous mnemonic names for the instructions. An assembler, a software development tool, can recognize the names (Lopez et al 2016). An assembler is a computer program that converts the ISA from a human-readable to a computer-readable format. Disassembles are also commonly used in debuggers and software applications to isolate and remedy errors in binary computer programs. Fig.1 shows the contents of the Computer Architecture Reference Microprocessors for RISC and CISC.

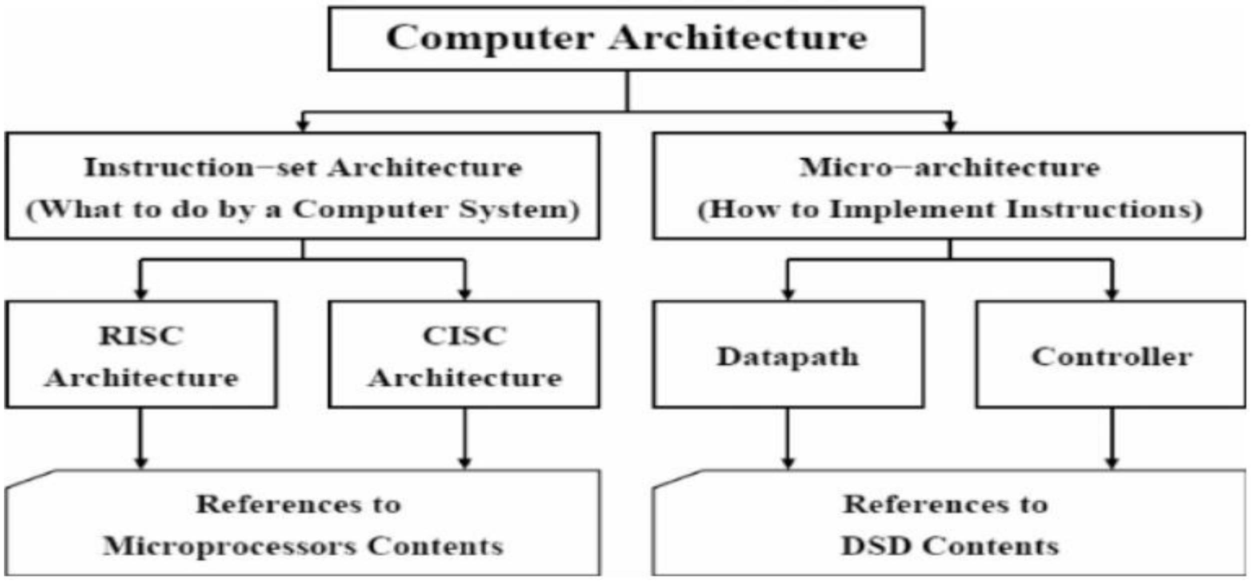

Figure 1: Computer Architecture Reference Microprocessors Contents for RISC & CISC
(Source: **After Muhammad Rashid, 1995**)

Materials and Methods

The implementation of Wireless LAN (WLAN), computers/laptops, and other technologies are used in the teaching and practical delivery of computer architecture etc. The **Wireless LAN** (**WLAN**)is a wireless computer network that connects two or more devices to form a local area network (LAN) inside a small area such as a home, school, computer laboratory, campus, or business building. This allows users to travel throughout the region while remaining connected to the network. A WLAheN can also enable access to the rest of the Internet via a gateway. Most current WLANs are marketed under the Wi-Fi brand name and are based on IEEE 802.11 standards. Due to their ease of installation and use, wireless LANs has become popular for use in the house. Commercial properties that provide wireless connection to their staff and customers are also using them. The **Computers /laptops**: Laptops consolidate all of the input/output components and functions of a desktop computer into a single unit, including the display screen, small speakers, keyboard, data storage device, optical disc drive, pointing devices (such as a touchpad or track pad), operating system, processor, and memory. The majority of current laptops have built-in webcams and microphones, and many also have touch screens. Laptops can be powered by an internal battery or an external power source such as an AC adapter. Hardware parameters, such as CPU speed and memory capacity, varies greatly across types, models, and price ranges. The **Desktop**: As a representation of desktop workloads, the use of SPECCPU2006 suite (www.spec.org). The SPECCPU2006 benchmark is a well-known desktop test that provides insight into core behavior. We discovered that the memory limited Cortex-A8, in particular, ran out of memory and execution was dominated by system effects for several benchmarks due to the huge memory footprint of the train and reference inputs. Instead, we publish results based on the test inputs, which for 10 of the 12 INT and 10 of the 17 FP benchmarks fit within the Cortex-memory A8's footprint. The **Mobile client**: This category presented challenges as mobile client chipsets typically include several accelerators and careful analysis is required to determine the typical workload executed on the programmable general-purpose core. The use of CoreMark (www.coremark.org), is widely used in industry white-papers, and two WebKit regression tests informed by the BBench study A. (Gutierrez, R & G. Dreslinski, 2011). BBench, a recently proposed Smartphone bench-mark suite, is "a web-page rendering benchmark comprising11 of the most popular sites on the internet today" by Authors. To avoid web-browser differences across the platforms, the use of the cross-platform WebKit with two of its built-in tests that mimic real-world HTML layout and performance scenarios for this study. The **Server**: The chosen server workloads informed by the Cloud-Suite workloads recently proposed by (Ferdman et al. 2012). Study characterizes server/cloud workloads into data analytics, data streaming, media streaming, software testing, web search, and web serving. The actual software implementations they provide are targeted for large memory-footprint machines intent is to benchmark the entire system and server cluster. This is unsuitable for study since in this research the aim is to isolate processor effects. Hence, picked implementations with small memory footprints and single-node behavior. To represent data-streaming and data-analytics, the use of three database kernels commonly used in database evaluation work.The **Applications**: Since both ISAs are touted as candidates for mobile clients, desktops, and servers, we consider a suite of workloads that span these. We use prior workload studies to guide our choice, and where appropriate we pick equivalent workloads that can run on our evaluation platforms. All workloads are single-threaded to ensure our single-core focus. The **Operating system**: Across all platforms, we run the same stable Linux 2.6 LTS kernel with some minor board-specific patches to obtain accurate results when using the performance counter subsystem. We useperf's1program sampling to find the fraction of time spent in the kernel while executing the SPEC benchmarks on all four boards; overheads were less than 5% for all but Gems FDTD and perlbench (both less than 10%) and the fraction of time spent in the operating system was virtually identical across platforms spanning ISAs. Intent to Keep OS effects as similar as possible across all platforms. is a small, portable personal computer (PC) with a thin LCD or LED computer screen mounted on the inside of the upper lid of the clamshell and an alphanumeric keyboard on the inside of the lower lid. The clamshell is opened up to use the computer. Laptops are folded shut for transportation, and thus are suitable for mobile use. Its name comes from lap, as it was deemed to be placed on a person's lap when being used. Although originally there was a distinction between laptops and notebooks (the former being bigger and heavier than the latter), as of 2014, there is often no longer any difference. Today, laptops are commonly used in a variety of settings, such as at work, in education, for playing games, web browsing, for personal multimedia, and general home computer use.The **Compiler**: This tool chain is built around a cross-compiler setup based on validatedgcc4.4. It may produce all binaries with the same front-end. All target-independent optimizations are enabled (O3), and machine-specific tweaking is turned off, resulting in a single set of ARM and x86 binaries. Because 64-bit ARM platforms are still in development focusing on 32-bit x86. THUMB instructions are disabled on ARM for a more RISC-like ISA.

Micro Architecture & Pipeline Organization of RISC Instruction Set Processors
As previously stated, the RISC microarchitecture is made up of five stages: retrieve, decode, and register choose, execute, and write back, all of which are pipeline stages of processors. The program counter (PC) is in charge of the instruction memory at the fetch stage, and it transmits this memory to the decoding stage for extraction. Once extracted, it passes on to the register select stage. The integer, floating point, and execution stages are then separated into three pieces allowing pipelining to take place. Finally, as illustrated in Fig. 2, the write back gets all three (3) executions and enables instruction in that order.

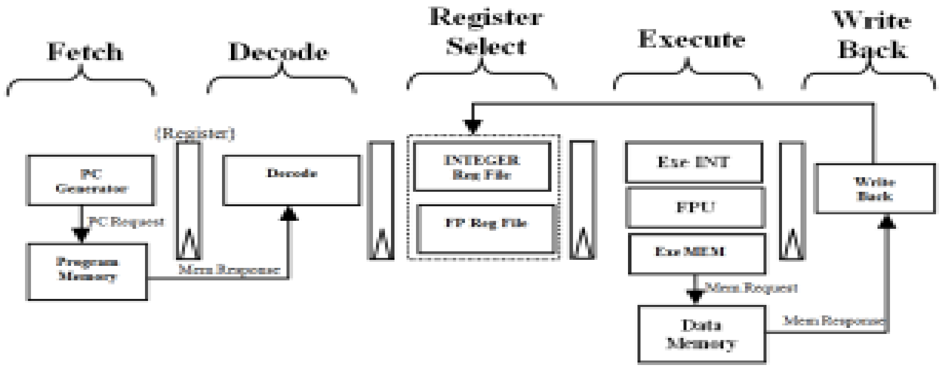

Figure 2: RISC Processors Architectural Top Level (Source: **After Sprunt and Kollar, 1985**)

The Fetch stage; It computes and works with two PCs during the fetch step. When the first PC is computed, it begins to predict the value of the second PC. It has a 32bit register in the first or current PC value, then a 32bit register in the second PC calculation, and once that is done, it adds 4 PC(+4) to the first PC. These predictions are made using a highly refined, or should I say sophisticated, branch prediction method. BTB and BP make up the branch prediction scheme. Branch target buffer is abbreviated as BTB, and branch prediction is abbreviated as BP. As a result, the multiplexer chooses whether to use PC (+4) or the projected PC in cases where the next PC number is unknown. Finally, both the PC and the next PC are sent to the decode phase.

The Decode stage: The instructions are sent from the fetch stage to this PC and the next PC, which receives or gets this instruction decoded from the program memory. In most circumstances, the decoder generates signals for future cases of instruction processing. Control signals are divided into three categories: Register Access, Register Update, and Pipeline. Note that these pipelines are those that have been scheduled for register access, which means that they are usually floating point or integer access, or in the worst-case scenario, NO ACCESS at all.

The Register Select Stage: It generates instructions from the decode stage that are acceptable at the register select phase, then selects operands from floating point or integer. Nonetheless, floating point has a 64-bit width, whereas integers have a 32-bit width. It contains three read ports, one write port, and a register file that is implemented in RAM Random Access Memory with delay at one clock cycle. The RISC processor of the register file accepts three major sources of address, namely (Read Address 1, Read Address 2, Read Address 3), which defines floating point or integer point access and returns a Register Access controlling signal. One regard to the decoding stage operand are usually passed on to the register select to either integer ALU. Computations are made in floating point execution pipeline or the memory address pipeline. In the execution stage schedules of instructions are made in FIFO which is intern part of the integer and memory.

The Execute stage: The execution stage consists of three (3) concurrent components that include integer arithmetic and logic operations, as well as FP computations and operand memory address. In integer execution is for the arithmetic and it consist of ADD, SUB, MUL and DIV while for the logical it consist of AND, OR, XOR and Shift operands accordingly. It is indeed important to remember that integer arithmetic performs computations for both conditional and unconditional branch and jump instructions.

The Write back stage: Concurrent or simultaneous reading from integer, floating point, or even memory units is referred to as write back. It normally commits the pipeline's instructions, then updates the register files and executes the stage. As a result, it reads from the top down. The MIPS architecture was chosen, and the design was tweaked to improve performance without sacrificing instruction count (Aboobacker Sidheeq. V.M, 2012). The time it takes for a processor to execute an instruction determines its performance (here execution means from fetching the opcode, then decoding of the opcode, then fetching the data, then performing the necessary operation and finally storing the result in the registers). The following statement, displayed in Fig.3, now gives the processor's execution time.

$$\textbf{Execution time} = \textit{(\# instructions) (Cycles per instructions) (Critical path delay)}$$

Figure 3: Executer time per clock cycle (John Neumann, 1945)

Challenges:

Separating the various implementation elements that are orthogonal to the ISA from the factors that are influenced or driven by the ISA is difficult in any ISA study. Chip process technology node, device optimization (high-performance, low-power, or low-standby power transistors), memory bandwidth, I/O device impacts, operating system, compiler, and workloads run are all aspects that are not ISA dependent. These problems are worsened when it comes to energy measurements and analysis, because chips that implement an ISA are mounted on boards, and distinguishing chip energy from board energy is difficult. Furthermore, some micro architectural elements may be determined by ISA-independent performance and application domain aims, while others may be governed by ISA-independent performance and application domain targets. Fig. 4 shows a high-level overview of this technique, including the four platforms, 26 workloads, and a set of parameters for each task on each platform. We use a variety of ISA implementations, with the ARM and x86 ISAs representing RISC and CISC, respectively. We give an in-depth and thorough analysis based on workloads from smart phones, desktops, and servers. The primary goal of this study is to see if and how the RISC vs. CISC ISA affects performance and power.

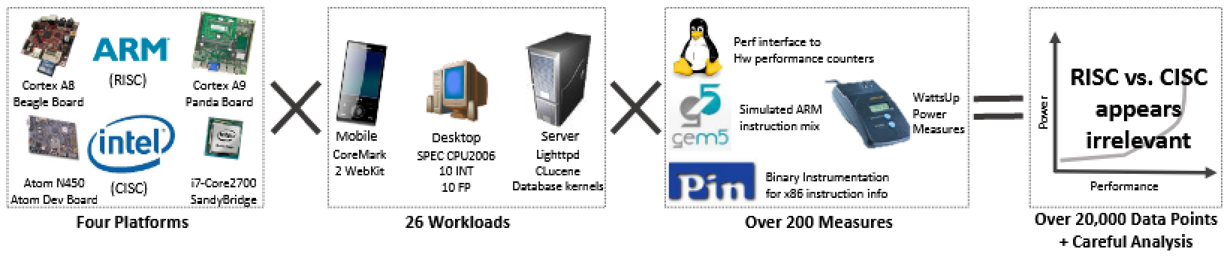
Cortex A8 Beagle Board
ARM (RISC)
Cortex A9 Panda Board
Atom N450 Atom Dev Board
intel (CISC)
i7-Core2700 SandyBridge
Four Platforms
Mobile CoreMark 2 WebKit
Desktop SPEC CPU2006 10 INT 10 FP
Server Lighttpd CLucene Database kernels
26 Workloads
Perf interface to Hw performance counters
Simulated ARM instruction mix
WattsUp Power Measures
Binary Instrumentation for x86 instruction info
Over 200 Measures
RISC vs. CISC appears irrelevant
Power
Performance
Over 20,000 Data Points + Careful Analysis

Figure 4: Formulary illustration of the CISCRISC & x86 (Source: **After Blem, Menon and Sankaralingam, 2013**)

Key Findings: The main findings from the study are:

1. Large performance gaps exist across the implementations.
2. Instruction count and mix are RISC vs. CISC ISA independent to first order.
3. Performance differences are generated by RISC vs. CISC ISA independent microarchitecture differences.
4. The energy consumption is again RISC vs. CISC ISA independent.
5. RISC vs. CISC ISA differences have implementation implications, but modern microarchitecture techniques render them moot; one ISA is not fundamentally more efficient.
6. ARM and x86implementationsare simply design points optimized for different performance levels.

Implications: The findings validate existing (or suspected) conventional wisdom while also adding value through quantification. Microarchitectural effects simply dominate ICT gear, performance, power, and energy consequences as a result of this. The ISA being RISC or CISC is mostly unimportant for today's mature microprocessor design world, according to the overall consequence of this work.

Conclusion:

In this paper, the debate between RISC and CISC is examined using recent ARM and x86 processors performing modern workloads to determine the impact of ISA on performance, power, and energy. The researcher, in course of lecture and practical delivery in the teaching of computer architecture encountered infrastructural and system problems, as well as failures and software/hardware bugs amongst others. The impact of ISAs on many sorts of workloads, such as workloads related to server and cloud computing, artificial intelligence, blockchain/cryptography, multimedia, and ultra-low-power systems, is one future avenue for this research. It will also be interesting to see if some ISAs are better suited to certain types of workloads. Furthermore, more different ISAs might be used to investigate the greatest possible performance variations amongst ISAs. The consideration for low power consumption in an environment such as Nigeria where power is a major constraint, it has become critical to deploy microprocessors. This suggests that it is critical to investigate the impact of ISAs on power consumption for a specific microarchitecture within the context of energy deficiency. Limitations in performance and power consumption between ISAs can therefore be used to create heterogeneous ISA multicore computers. Another issue to consider is the idea of developing models to estimate the impact of ISAs on certain applications. The machine learning or fuzzy reasoning approaches can be used to create these models. Such models can be used to help schedule applications over multiple cores in heterogeneous ISA multicore systems, as well as to improve ISAs. In this study, a five-stage 32-bit ISA instruction set containing MIPS, RISC, and CISC is investigated, as well as microarchitectures in stages. The fetch, decode, execute, register select, and write back phases are explained in terms of off grouping integers, and the floating point pipeline is verified as well. This is in addition to the creation of out-of-order floating point units and function modules derived from FPU requests, input decoders, schedule decoders, and scheduling instruction, among other factors.

References:

Aboobacker Sidheeq. V.M. (2012)"Four Stage Pipelined 16 bit RISC on Xilinx Spartan 3AN FPGA" in, International Journal of Computer Applications (0975888) Volume 48– No.6.

A. Gutierrez, R. G. Dreslinski, T. F. Wenisch, T. Mudge, A. Saidi,C. Emmons, and N. Paver. (2011). Full-system analysis and characterization of interactive smartphone applications. InIISWC '.

(Amy M. Johnson, Matthew E. Jacovina, Devin G. Russell, and Christian M. Soto. (2021) Challenges and solutions when using technologies in the classroom.

BSchmalz, M.S. (2017). "Organization of Computer Systems". UF CISE.

Crossley, S.A.&McNamara, D.S. (2021). Adaptive Educational Technologies for Literacy Instruction. Taylor & Francis, Routledge: NY.

Clements, Alan. Principles of Computer Hardware. (2003). Architecture describes the internal organization of a computer in an abstract way; that is, it defines the capabilities of the computer and its programming model. You can have two computers that have been constructed in different ways with different technologies but with the same architecture. (Fourth Ed.). p. 1

David Money Harris and Sarah.L.Harris, (2019). "Architecture" in, Digital Design and Computer Architecture, 3rd ed., Graphic World Publishing Services, chapter 6, pp. 291-317.

D. Bhandarkar and D. W. Clark. (1991).Performance from architecture: comparing a RISC and a CISC with similar hardware organization. In ASPLOS '.

D. A. Patterson and D. R. Ditzel. (1980).The case for the reduced instruction set computer. SIGARCH Comp. Arch. News, 8(6).

Esoswo Francisca Ogbomo. (2021) ISSUES AND CHALLENGES IN THE USE OF INFORMATION COMMUNICATION TECHNOLOGY (ICTs) IN EDUCATION.

Emily Blem, Jaikrishnan Menon, and Karthikeyan Sankaralingam. (2013). Power Struggles: Revisiting the RISC vs. CISC Debate on Contemporary ARM and x86 Architectures.

Frey, Brad. (2005). "PowerPC Architecture Book, Version 2.02". IBM Corporation.

Grover, S., & Pea, R. (2013).Using a discourse-intensive pedagogy and Android's App Inventor for introducing computational concepts to middle school students. Proceedings of the 44th SIGCSE technical symposium on Computer science education. ACM

Hennessy, John; Patterson, David. (2019). Computer Architecture: A Quantitative Approach. This task has many aspects, including instruction set design, functional organization, logic design, and implementation. (Fifth ed.). p. 11.

Johnson, Lyle (2016). "A Description of Stretch"(PDF). *p. 1.*

John L. Hennessy and David A. Patterson. (2019). Computer Architecture: A Quantitative Approach (Third ed.). Morgan Kaufmann Publishers.

Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., Thomas, L. (2004). A multi-national study of reading and tracing skills in novice programmers. Working Group Reports from ITiCSEon Innovation and Technology in Computer Science Education, Leeds, United Kingdom. 119–150.

Laplante, Phillip A. (2001). Dictionary of Computer Science, Engineering, and Technology. CRC Press. pp. 94–95. ISBN0-8493-2691-5.

Lopez, M., Whalley, J., Robbins, P., &amp; amp; Lister, R. (2008). Relationships between reading, tracing and writing skills in introductory programming. Proceedings of the Fourth International Workshop on Computing Education Research, Sydney, Australia. 101–112.

M. de Kruijf, S. Nomura, and K. Sankaralingam. (2010) Relax: An architectural framework for software recovery of hardware faults. InISCA '.

M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee,D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Fal-safi. (2012). Clearing the clouds: a study of emerging scale-out work-loads on modern hardware. InASPLOS.

M. J. Flynn, C. L. Mitchell, and J. M. Mulder. (1987). And now a case for more complex instruction sets. Computer, 20(9).

Martin, Milo. (2013). "What is computer architecture?" (PDF). *UPENN.*

M. de Kruijf, S. Nomura, and K. Sankaralingam. (2010). Relax: An architectural framework for software recovery of hardware faults. InISCA '.

Muhammad Rashid. (2015). International Journal of Engineering Education 31(1):141-153.

Neumann, John (1945). First Draft of a Report on the EDVAC. p. 9.

Null, Linda (2019). The Essentials of Computer Organization and Architecture (5th ed.). Burlington, MA: Jones & Bartlett Learning. p. 280. ISBN9781284123036.

R. Colwell, C. Y. Hitchcock, III, E. Jensen, H. Brinkley Sprunt, and C. Kollar. (1985). Instruction sets and beyond: Computers, complexity, and controversy. Computer, 18(9):8–19.

Sivarama P. Dandamudi, (2020). "Introduction", in Guide to RISC Processors, Springer, chapter 1, pp.3-11.

Sivarama P. Dandamudi, (2005). Design Issues" in Guide to RISC Processors, Springer, chapter 2, pp.13-36.

Williams, F. C.; Kilburn, T. (2010). "Electronic Digital Computers" Nature, ***162*** *(4117): 487, Bibcode: 1948Natur.162.487W doi10.1038/162487a0, S2CID 4110351.*