MIPS, ARM and SPARC- an Architecture Comparison

Sarah El Kady, Mai Khater, and Merihan Alhafnawi

Abstract—**This paper provides an insightful comparison between three of the most popular and widely-used Reduced Instruction Set Architecture (RISC) processors- MIPS, ARM and SPARC. In order for the reader to acknowledge the differences between those three architectures and fully understand the significance of each one, a comparison between the three architectures is very important. Given that MIPS, ARM and SPARC are used in many different applications, this paper provides a comparison to be able to cover as many software and hardware applications used by these three architectures as possible. After reading this paper, the reader would have had enough information on the three RISC processors, the differences between them, and where they are most effectively used. This in turn will help in determining which ISA of the selected three is most suitable for the reader's required application.**

Index Terms—**registers, memory access time, stack implementation, architecture, exception handling.**

I. Introduction

One sometimes finds it challenging to choose which Instruction Set Architecture to use for implementing a specific hardware and/or software application. This paper attempts to cover as many applications as possible by providing a comparison between three ISAs that are used in different fields. With the lack of some significant information about specific processors like SPARC and the rapid change of the hardware technologies in the market, how can one decide which processor is the most efficient performance-wise and cost-wise with one's desired application? Accordingly, this paper will provide sufficient information to help make this decision. Crucial as it is, programmers need to decide on which factors are most important in their application. That is, which factors, if increased or decreased, will affect the performance, the cost or other factors of the application negatively or positively? Through knowing information about the three ISA and acknowledging their differences, programmers are then able to make this decision confidently, which could result in bettering their application and its usability. This information will be demonstrated through a comparative study between three of the most renowned ISAs worldwide: MIPS, ARM, and SPARC. This paper presents the interesting features and background of MIPS, ARM, and SPARC by being divided into sections that highlight the main differences between each of them.

Manuscript received March 15, 2014; revised April 03, 2014. This work was supported by the Office of the Dean of the Undergraduate Studies by The American University in Cairo.

S. El Kady is with the Department of Computer Science and Engineering, The American University in Cairo, Cairo, Egypt (E-mail: sarahelkady@aucegypt.edu).

M. Khater is with he Department of Computer Science and Engineering, The American University in Cairo, Cairo, Egypt (E-mail: maimagdi@aucegypt.edu).

M. Alhafnawi is with the Department of Computer Science and Engineering, The American University in Cairo, Cairo, Egypt (E-mail: merihan@aucegypt.edu).

II. History

This section provides an introduction to the three ISAs through speaking briefly about their history (MIPS, ARM and SPARC, respectively). The Microprocessor without Interlocked Pipeline Stages known as MIPS is one of many RISC processors. RISC processors commonly use a load/store architecture where the only instructions that can deal with the memory are load and store. MIPS was invented in the early 1980s in Stanford University. When researchers started to develop MIPS, it was to support embedded systems and connectivity. However, currently, it started to support the mobile market as well. Multiple versions of MIPS have been developed over the years; starting from MIPS I up to MIPS V. In addition, there were other two types of MIPS architectures developed: MIPS-32 and MIPS-64.

The ARM processor was developed by a British company called Acorn Computer in 1985. The companys target back then was low cost PCs. Later, Acorn introduced an advanced RISC machines and changed ARM from (Acorn RISC Machines) to Advanced RISC Machines. Now, ARM is a leading architecture in many market segments, especially cost sensitive embedded systems. [1]

The scalable Processor Architecture known as SPARC ISA was developed by Sun Microsystems in 1987. SPARC has a wide range of CPU implementation compatibility, many compilers tool, a licensable standard UNIX operating systems, and window system and graphical user interface (GUI) and many other features. In addition to that, it is considered a solution for high-performance [2].

III. Applications and Market

This section is very helpful in determining where MIPS, ARM and SPARC are used and where their places are in the market.

A. MIPS

Unlike SPARC, MIPS has many market applications. All these applications are cheaper when a processor like MIPS is used in embedded systems and networking where complex processes take place.

One of the advantages of MIPS is that it has many open source implementations. This makes MIPS a more popular choice among beginners who are trying to code in assembly language. However, the simplicity of MIPS ISA compared to other architectures is what makes it the most popular choice among beginners. Thus, many educational institutions use it to teach their students about assembly language before introducing them to other ISAs.

MIPS Management (MM) is a MIPS application that ensures that programs running on MIPS are performing well [3]. Nowadays, MM ensures lower demands on system resources by reducing the amount of CPU resources used. This, in turn reduces the cost of running applications using CPU cycles that are not in use which makes it very beneficial in avoiding faults. For example, it is very helpful in coding online transactions. In addition to that, MIPS has an MMU (Memory Management Unit) chip that helps in utilizing the memory resulting in the reduction of unnecessary memory loss and cost of memory consumption. Lately, MIPS has been integrated closely with networking and mobile applications because of its effective costs and less power consumption [4].

MIPS has started to slightly fall behind ARM starting 2011 due to the fact that ARM had worked hard to develop itself in the area of mobile application development - the market interest nowadays. However, MIPS was revived by targeting the Chinese market and dedicating more core processors.

B. ARM

Many believe that ARM has secured its future for many years ahead of its time. This is due to the wide spread of its processors in all areas of technology and its strategy in updating these processors to meet the new functionalities demanded by the market.

As the high performance, low implementation size and low power consumption are key features of all ARM architectures, ARM has developed two main profiles: the ARMv8-A architecture profile for high performance markets such as mobile and enterprise, and the ARMv8-R architecture profile for embedded applications in automotive and industrial control. In addition to that, ARM has two different Cortex series: firstly, the Cortex-M series which offers an ideal solution for most embedded applications. Secondly, the Cortex-A series which is widely used in telemedicine, security and avionics [1].

Keeping up with the markets demands, ARM has developed multiple extensions supporting java acceleration. Those extensions include Jazelle, security (TrustZone), Single Instruction Multiple Data (SIMD), and advanced SIMD (NEON) technologies.

Education-wise, the widespread of ARM's processors and ISAs in the market forced many universities to change some of their courses curriculum and include ARM for students to acquire the necessary up-to-date knowledge.

C. SPARC

Unlike MIPS and ARM, SPARC is not used for educational purposes. It is mainly used by programmers and computer architects who deal with server applications and lower level programming. SPARC was owned by Sun Microsystems for almost 25 years until it was acquired by ORACLE Corporation in 2010. During those 25 years, Sun published many applications for the SPARC based computer. In its early years, Sun released the SPARCstation1 which tripled the performance of the first SPARC based single processor. Later on, Sun developed Solaris which is the operating system for SPARC computers. In 2004, SPARC launched the Dual-Core UltraSPARC IV which was the first multi-core SPARC processor. Following that, it launched the first UltraSPARC of the T-series which was an 8-core system on a chip. In 2010, when SPARC was acquired by ORACLE, SPARC was commonly used in SUN ORACLE Station. Since then, ORACLE worked very hard on the SPARC processors to improve the utilization of the system and its efficiency and that was shown in the new versions of the M-Series and T-series UltraSPARC that was just launched. In addition to that, ORACLE launched different types of servers. For instance, SPARC servers, which run on Solaris and have a very high-performance. Those are: Sun Blade Servers, which integrate both x86 and SPARC-based servers, and Sun Netra Carried-Grade Servers which are designed for the 4G infrastructure [5].

IV. Architectural Differences

This section highlights the major distinctions among the three processors in the architectural context.

A. MIPS

MIPS provides low overhead and power efficiency. Sir Hossein Yassaie, CEO of Imagination Technology, pointed out, in the Imagination Press in 2013 in Seoul, that MIPS is less complicated than other processors since it uses less instructions when achieving a specific task, making it a less power consuming processor [6].

B. ARM

One of the features that distinguishes ARM is its very dense 16-bit compressed instruction set "Thumb" that executes instructions unconditionally. Many of Thumb's instruction formats are less regular than those of ARM's. Also, one of ARM's architectural differences is that the cores used can switch between two execution states: ARM or the 16-bit thumb. Some ARM cores introduce the Thumb-2 instruction set, which is basically a mixture of 32 bit and 16 bit instructions maintaining the instruction density yet keeping it more flexible [7], [8].

C. SPARC

There are three main differences among SPARC and other RISC architectures, such as MIPS and ARM. The first architectural difference is the Register File Model, which unlike SPARC's peers, is not a flat 32-register. Quoting Ben J. Catanzaro It is a set of overlapping register windows arranged as a circular buffer. This circular file model is very efficient in the handling of dynamic linking. The second architectural difference is Annulling Delayed Branches which means that the instruction that is in the delay slot of the branch (unless it is a branch instruction) will be copied in the delay slot and will not be executed until the branch is taken. Last but not least is SPARC's Software Environment which is considered a low-cost RISC computer that is based on open standards [9].

TABLE I
REGISTERS CONVENTION COMPARISON

	MIPS	ARM	SPARC
Number of Registers / width	32 registers/32-bits	37 registers/32-bits	37 registers/32-bits
Register Division	Registers are reserved for special operations. • Two-special purpose registers: Hi/LO: they store the results of the integer multiply and divide instructions • 30-General Purpose registers: from \$0 to \$31. 1) \$0: hardwired to zero 2) \$31: link registers. 3) \$29: stack pointer.	Registers are divided into groups: • 31 General-purpose Registers: – Unbanked registers: from $r0$ to $r7$ – Banked registers: from r8 to r14 • Status Registers: 1) Current Program Status Register (CPSR) 2) the last five are called saved program status register (SPSR)	Registers are divided into four groups: • In Registers: from $\%i0$ to $\%i7$. • Global Registers: from $\%g0$ to $\%g7$. $\%g0$ is always hard-wired to zero. • Local Registers: from $\%l0$ to $\%l7$. They are user freely in any code. • Out Registers: from $\%o0$ to $\%o7$

V. REGISTERS CONVENTION

This section highlights the major distinctions among the three processors in the architectural context. To elaborate more on the registers conventions of each ISA, we present Table I which compares the registers from three perspectives: Number of Registers, Registers width, and registers division. Note that MIPS registers are referred to with a dollar sign (\$), ARM registers are referred to with the real register number and SPARC registers are referred to with the percentage sign (%).

VI. OPERATING MODES

This section covers the different operating modes for the three different ISAs. Each operating mode is allowed to access certain registers and to use certain instructions.

A. MIPS

MIPS has only two operating modes: the kernel (supervisor) mode and the user mode. In the kernel mode, when the status bit is set to 0, the operating mode is switched to the kernel mode and can access and change all registers. This mode has the privilege over other modes and gets switched to in case of an error, interruption, exception or at power up. In the user mode, when the status bit is 1, the operating mode switches to the user mode. This mode is accessed by users and has a lower privilege than that of the kernel mode. It also prevents different users from interfering with one another [10].

B. ARM

ARM has seven basic operating modes, starting with the processor mode which has two operating modes under it, the user mode and the privileged mode. The user mode is where most applications' contents or operating systems' tasks run. The privileged mode has under it two operating modes, the system mode and the exception mode: the system mode is the privileged mode which uses the same registers as used by the user mode. The exception mode has five operating modes under it: the first operating mode is the Supervisor (SVC) which is entered under two conditions, when on rest or when Software Interrupt Instruction (SWI) is executed, the second mode is Abort (ABT) mode which is used to handle memory access violations, the third mode is the Undefined (UND) mode which is used to handle undefined instructions, the fourth mode is the Interrupt (IRQ) mode which is entered when a high priority interrupt is raised. Last but not least, the Fast Interrupt (FIQ) mode which is entered when a high priority interrupt is raised [11].

C. SPARC

SPARC executes the instructions in only two modes: the supervisor mode and the user mode. The supervisor mode is used to access the processor state registers, such as the Window Invalid Mask (WIM), and the input/output devices. Also, in the supervisor mode, there is a full access to the memory and its system tables making it privileged in almost all architectures. The user mode is used to write to and read from the processor state register and has a limited access to the memory since a load or a store instruction cannot be executed in it. Additionally, to know whether the processor is in the user mode or in the supervisor mode, the processor status word is used to determine the current state of the processor [12] [13].

VII. ADDRESSING MODES

This section compares the different addressing modes used in each ISA since the method the address is calculated with differs slightly from one ISA to another.

A. MIPS

MIPS support five addressing modes:

Register Addressing: This mode is mainly used in calculating the effective address of the jump register (jr) instruction.
Immediate Addressing: This mode does not access memory and thus is relatively faster than other modes. The immediate is of size equal to 16-bits.
PC-Relative Addressing: This mode is used to determine when the branch instruction occurs by summing the offset

value with the PC.

Pseudo-direct Addressing: This mode is used in the jump instruction where the value of the offset is 6-bits and the target of the instruction jumped to is 26-bits. The upper four bits of the PC and the least two significant bits, which are 00, are all concatenated with the 26-bit immediate resulting in a 32-bit instruction.

Base Addressing: This mode is used in store word and load word instructions. It is known as the indirect addressing since the register acts as a pointer to some memory location whose address could be found in the register [10].

B. ARM

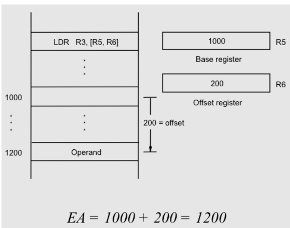
ARM supports multiple addressing modes, including modes that allow direct bit shifting. There are four main addressing modes to calculate the effective address:

Pre-indexed Addressing: In this mode, the source/destination address is stored in a register offset by another value. Figure 1 (a), shows how the load operation is used to calculate the pre-indexed effective address.

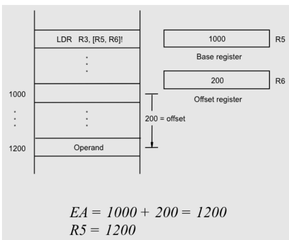
Pre-indexed Addressing with Write Back: In this mode, it is sometimes useful to save the new address in a register. To indicate that this effective address is being written back, add an exclamation mark (!) at the end of the load instruction. Figure 1 (b) describes the operation of the write back.

Post-Indexed Addressing: This mode is similar to Pre-Indexed Addressing with Write back. However, the address is modified and saved only after the load/store operation. Figure 1 (c) illustrates the effective address calculation in the post-indexed addressing mode [14].

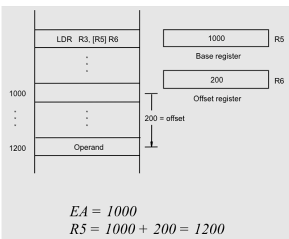
Program Counter Relative Addressing: This mode allows the ARM architecture developers to address memory relative to the Program Counter (r15).



(a) Pre-indexed addressing mode



(b) Pre-indexed addressing mode with write back



(c) Post-indexed addressing mode

Fig. 1. ARM Addressing Modes

C. SPARC

SPARC supports two addressing modes to compute the effective address: the register indirect with index mode and the register indirect with immediate addressing mode.

The Register indirect with index: This mode computes the effective address by adding the contents of the base register to those of the index register. The effective address cannot be equal to only the base register but the index register could be made equal to zero and by that the effective address will be equal to the base register. E.g.: Ld [%o1], %o2

The Register indirect with immediate: This mode computes the effective address by sign extending the 13-bit immediate to 64 bits and then adds the contents of the base register to it [7]. The effective address could be made equal to the base register by making the constant equal to zero. E.g.: Ld [%o1+30], %o2

VIII. Conditional Execution

This section compares the three ISAs with respect to their abilities to execute instructions conditionally. Conditional execution saves most of the CPU time as it does not execute an instruction unless its flag is set to the appropriate value (i.e. no unneeded instructions will be executed).

A. ARM

In ARM, almost all instructions contain a condition field which determines whether the CPU will execute those instructions or not. Non-executed instructions consume one cycle. NOP (No Operation) instruction cannot be collapsed; it should complete its cycle normally. This feature in ARM allows very dense in-line coding without branches. The time penalty of not executing several conditional instructions is frequently less than that of branching overheading. This conditional execution is implemented using 4 bits (Condition field).

In code, all instructions can be made to execute conditionally by adding a postfix that sets the condition field with the appropriate bits to execute a certain condition. For instance, if you want to execute the ADD instruction only if the zero flag is set, then the instruction will be written as ADDEQ r0, r1, r2 and will be translated to the following: if zero flag is set, then r0=r1+r2. Otherwise, do not execute the addition. The default is that data processing instructions (i.e. Arithmetic instructions) do not affect the condition flags, but if programmers want to update the condition flag, they can use the letter S after the instruction. For instance, the previous instruction will be ADDS r0, r1, r2 which will be translated to r0=r1+r2 and then set the desired flag. Data processing instructions are the largest family of instructions sharing the same instruction format which support the RISC architecture. These instructions include arithmetic operations, comparisons, logical operations, and data movement between registers [15].

B. SPARC

SPARC has the condition code update opportunity. A conditional flag is determined by a single bit when the instruction is encoded. There are four conditional flags: Z (Zero), N (Negative), C (Carry), and V (over flow). The zero flag (Z bit) is used when the result in the operation is zero and is diminished when the result is otherwise. Similarly, the Negative flag (N bit) is set when the result of the operation is negative and removed when the result is positive. The

ISBN: 978-988-19252-7-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

overflow flag (V bit) is used when the signed integer result cannot be represented in 32 bits and is cleared when the result can be stored in 32 bits. Finally, the carry flag (C bit) is set when the operation generates a carry out of the most significant bit, and cleared otherwise. To use any of the condition codes, add ”cc” to the end of the arithmetic operation. For example, if you want to add the carry bit, you can use this instruction: ADDc Rs1, Rs2, Rd but if you want to add the carry bit and check on something before adding the two registers, you can use this instruction:

ADDccc Rs1, Rs2, Rd

IX. Exception Handling

This section shows what happens in case of the occurrence of an exception and how each ISA handles it.

A. MIPS

Exception handling in MIPS is supported by the coprocessor 0. When an interruption occurs, the content of the PC gets stored in the Exception Program Counter (EPC). Then PC gets the new value which is the address of the instruction to which the interruption is going to be handled. Afterwards, the operating mode switches from the user mode to the kernel mode. Finally, after handling the exception, the return address is incremented by 4 to avoid executing the same instruction again and the PC gets the address of this instruction and continues executing normally [16].

There are four main registers that handle exceptions in MIPS: register 8 (BadVAdder) which holds the memory address at which the exception occurred, register 12 (Status) which enables the bits, masks the interruption and states when an interruption happens, register 13 (Cause) which holds the type of exception happened and pending interrupt bits, and register 14 (EPC) which has the address of the instruction that caused the exception [17].

B. ARM

ARM exception handling is only made in the ARM mode; however, for Thumb-2 cores, the switch between the modes is not needed. When an interrupt occurs, the core copies the Current Program Status Register (CPSR) into the Saved Program Status Register (SPSR) then sets the appropriate CPSR bits to change to ARM state, change to exception mode and disable interrupts. Finally, the PC is set to the interrupt vector address [8].

C. SPARC

In SPARC, when an exception or a trap occurs, the trap enable known as (ET) gets cleared and the processor switches to the supervisor mode by changing the current processor execution state to be stored in the previous state bit. Then, the current window pointer (CWP) is decremented and the program counter (PC), NPC and Processor State Register (PSR) are saved in the first three local registers (%l0 to %l2) and the code of the trap handling is stored in %l3 to %l7.

X. Stack Implementation

This section starts with a brief description of what the stack is and then proceeds to a Table 2 where the comparison, in terms of stack implementation, is shown in details.

A stack is a data structure that is used for the storage of data. The data can be saved or obtained via operations done on the stack, named stack pushing and stack popping, respectively. The following two figures (Figures 2 and 3) illustrate the mechanism of pushing and popping [18].

TABLE II
Stack Implementation Comparison

	MIPS	ARM	SPARC
Direction of the Stack growth	The Stack grows downward	ARM supports both ascending and descending stack	The stack grows downward
Push and Pop Instructions	Does not support push/pop instructions. Instead, it manipulates the stack pointer (register 29.	No explicit push/pop instructions but it can manipulate the stack pointer to do so	No explicit push/ pop instructions but can be implemented using the stack pointer(sp)

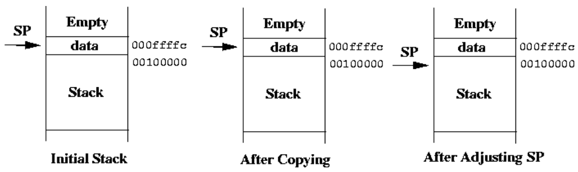


Fig. 2. Stack popping. The stack pointer (SP) copies the data, in the address given, in a register first, and then moves the stack pointer.

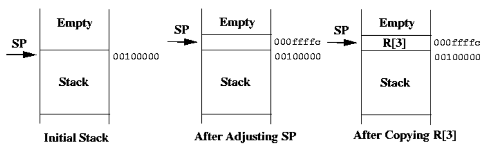


Fig. 3. Stack pushing. The stack pointer (SP) is moved in order to place the contents of register 3 (R[3]) into the address that the stack pointer is pointing to.

XI. Conclusion

Due to the incredible pace in which new technologies in computer architecture emerge, more development and enhancements are predicted to happen in the near future. This will make the choice of which ISA to use harder as the options would be increasing and the specifications of all the ISAs would become similar to some extent. Given that finding a resource that would address these three particular ISAs in a comparative study is very hard, this paper was written to provide a tool for the reader not only to see the differences between MIPS, ARM, and SPARC, but also to be able to choose which processor to use for the required job at hand.

ISBN: 978-988-19252-7-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

ACKNOWLEDGMENT

Our greatest gratitude goes to Professor Khaled El Ayat for his insightful observations and extensive help and support in revising this paper.

We would like to express our sincere thanks to Dr. Yousra El Kabani for her assistance in reviewing our paper and her constant encouragement and help.

Thanks to Professor Howaida Ismail for providing us with all the needed knowledge and for her incredible suggestions in improving this paper.

We would finally like to thank Professor Mohamed Moustafa for providing us with helpful comments.

REFERENCES

[1] A. I. Center., *ARM Information Center. N.p., n.d.*
[2] B. J. Catanzaro, Ed., *The SPARC Technical Papers*. New York, NY, USA: Springer-Verlag New York, Inc., 1991.
[3] C. Corporation, *Best Practices For Improving Application Performance and Lowering Cost by Managing MIPS.*, 2012.
[4] L. Kwiatkowski and C. Verhoef, *Reducing operational costs through MIPS management*. Department of Computer Science, Vrije Universiteit Amsterdam, 2012.
[5] O. CORPORATION, *ORACLE SPARC 25 Years of SPARC Innovation.*, 2014.
[6] J. Bae, *Imagination Technologies brings MIPS Architecture to the fore.*, 2013.
[7] S. P. Dandamudi, *Guide to RISC Processors for Programmers and Engineers*. Springer Science+Business Media, Inc., 2005.
[8] M. McDermott, *EE382N-4 Embedded Systems Architecture The ARM Instruction Set Architecture." Lecture. EE345M/EE380L Course Material, Spring 2014*. The University of Texas at Austin, 2014.
[9] O. CORPORATION, *Sun Servers, Integrated Systems, ORACLE*. Oracle.com., 2014.
[10] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.
[11] A. Denault, *Other Architectures (ARM, IA-64,etc) Lecture. Index of CS 573*. School of Computer Science, McGill, 2005.
[12] R. P. Paul, *SPARC Architecture, Assembly Language Programming, and C*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1999.
[13] D. L. Weaver and T. Germond, *The SPARC Architecture Manual*, 9th ed. Englewood Cliffs, NJ, USA: PTR Prentice Hall, 2003.
[14] M. Shalan, *ARM ISA*. Department of Computer Science and Engineering, The American University in Cairo, 2013.
[15] Samsung, *S3C4510B*. Samsung.
[16] D. Chiarulli, *4a: Exception and Interrupt handling in the MIPS architecture*. University of Pittsburg, 2014.
[17] R. Teodorescu, *Instruction Set Architecture of MIPS Processor Presentation B*. The Ohio State University, 2008.
[18] C. Lin, *Understanding the Stack*. Department of Computer Science, University of Maryland, 2003.

AUTHORS