Power Differences Between RISC and CISC In Contemporary Microprocessors

California State University, San Bernardino
CSE595 Independent Study Course Project–Fall 2013
Rodolfo Gutierrez

***Abstract*—The landscape of the computing industry has changed drastically over the past decade. Mobile devices now account for an increasingly large market share and this has created an interesting turning point in terms of processor complexity design choice for mobile device engineers. Traditionally low-power ARM processors (RISC ISA) have begun to make their way into the high-performance server market, while traditionally high-performance x86 processors (CISC ISA) have entered the low-power mobile market. Consequently, the question of whether the ISA plays a significant role in the power efficiency of either group of microprocessors seems relevant now more than ever. After analyzing empirical studies aimed at distinguishing the power differences in both, RISC and CISC ISAs in contemporary microprocessors, I have arrived at the conclusion that neither ISA is inherently more or less energy efficient. The significance of the ISA appears to simply be an engineering design preference aimed at achieving different levels of performance.**

I. Introduction

The earliest studies aimed at showing the potential advantages of RISC and CISC architectures began in the 1980's and 1990's when processors appeared to be constrained by physical chip size as well as ISA. The question still remains if the debate ever really settled on any technical issues, however, both ISAs thrived during this time. For the past decade, ARM has dominated the low-power embedded systems and mobile market, while x86 has dominated the desktop and server market. Now that both, ARM and x86, have begun to make their way into new markets, ARM in the high-performance server market and x86 in the low-power embedded systems and mobile market, the role of the ISA in the power efficiency of both processors seems relevant now more than ever.

While the empirical studies I considered in preparation for this paper performed an exhaustive analysis of each ISA in several ARM and x86 processors, I was particularly concerned with the analysis of the ARM Cortex and Intel Atom processors. These two processors are most appealing to the RISC vs CISC debate for the following two reasons: they have very similar levels of performance and have begun to crossover into each other's previously controlled market (eg. mobile and servers). To give perspective to this debate, in the two proceeding sections I will present architectural comparisons between ARM Cortex and Intel Atom. I will conclude with an analysis of the empirical evidence that points to neither ISA in either processor holding a potential advantage in terms of power efficiency.

II. Architecture Overview

A. ARM Cortex

ARM is a RISC machine, also called a "Load/Store" type architecture [5]. Historically, the emphasis of RISC machines has been on simplicity and efficiency. Some potential advantages and features of RISC machines are the following [3]:

- emphasis on register-oriented operations
- instructions that primarily execute in a single cycle
- simple LOAD/STORE instructions for memory access
- limited addressing modes
- pipelined instruction cycle

The ARM processor adheres to all of the characteristics stated above, however, ARM has a unique mechanism called Thumb or Thumb2 which essentially allows the limited ARM instruction set to be more competitive with other architectures by improving code density through a compressed 16-bit form of the original 32-bit instruction set [5]. Unfortunately, this improved instruction set has not proven to significantly reduce power consumption. In order to address that issue ARM looked into an optimization of its micro-architecture. The optimization addresses one commonly identified source of power consumption in integrated circuits [5]:

- Short circuit: Power dissipated in momentary short circuits during complementary PMOS/NMOS switch transitions

To address short circuit power dissipation, ARM implemented a cache-related micro-architecture technique, which essentially takes advantage of the sequential nature of many cache operations. Consider the following classic cache controller design:

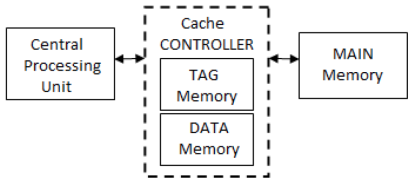

Fig. 1. Classic L1 Cache Design [5]

The level 1 cache shown above is internal to the microprocessor and is a small, high-performance memory which stores copies of data from the most frequently used main memory locations. The purpose of this is to reduce the average time the microprocessor accesses the main memory [1][6]. Latency is much less when memory accesses are cached memory locations, therefore if most memory accesses are stored in cache, then the average latency of memory will be much closer to cached latency than to that of main memory latency [1][6]. When the processor needs to read or write to a main memory location, it first checks to see if there is a copy of the data stored in cache, if there is, the processor instantly reads from or writes to cache. This is much faster than reading or writing to main memory. Therefore, as clock rates go high, the the L1 cache is required when performance, which is measured in terms of CPI or clocks per instruction must be maintained [6].

The cache is made up of two types of high-speed RAM or random access memory, labeled DATA and TAG. Effectively, DATA, measured in bytes, contains the minimum number of words that are copied to cache when it is updated (many ARM implementations use a 32-byte line) [6]. Each line in DATA has a corresponding location in TAG, which is indexed to the same depth in array. The TAG location provides the address representing where in memory the line in DATA is from [6]. When there is a request for main memory data, the cache controller compares the address in the TAG memory to the same address in main memory. If there is a match, then this is what is called a 'hit' which means the cache controller can satisfy the memory request by reading or writing lines recently copied to cache.

Two potential optimizations are apparent when considering an implementation that will increase hit rates. The first is to increase the cache DATA memory size. The larger the size, the higher the likely-hood that there will be a hit when there is a memory request. However, with this optimization comes a drawback: increasing cache DATA memory size will increase power consumption. The second alternative appears to be the most effective and it is what ARM has opted for when designing the ARM11 micro architecture, which is present in the ARM Cortex. The idea is to reduce full cache reads, especially for the instruction cache, which reduces power consumption [6]. ARM11 micro-architecture allows three successive cache data misses to occur without an instruction pipeline stall, that is of course assuming there are not any data dependencies that occur after the first miss. This technique reduces the number of pipeline stalls and the associated cache reloads.

In short, a good L1 cache design reduces power consumption by reducing the number of transistors required to implement the cache memory and the number of clock cycles the CPU requires to complete an instruction [6] A comparison to the Intel Atom in the following section will illustrate that the significant power reduction techniques do not occur at the instruction set level but rather at the micro-architecture level.

B. Intel Atom

The Intel Atom processor is a CISC machine. CISC processors are most commonly found in PCs and servers. While details vary, some typical characteristics and potential advantages of CISC processors are the following [3]:

- richer instruction sets
- emphasis on object-oriented operations
- individual instructions that often require dozens of RISC instructions to represent (e.g floating point representation)
- extensive and highly sophisticated addressing modes
- pipelined instruction cycles on many recent machines

The Atom processor employs all of these CISC characteristics, however, in order to reduce power consumption, Intel also looked to an optimization of its micro-architecture [4]. Until recently, the Intel Atom rested on the Bonnell micro-architecture. One key characteristic of the Bonnel was that it used in-order execution. Briefly, the general CPU pipeline functions the following way: the PC or program counter within the CPU points to the address in memory of the next program to be executed. The instructions for that program are then decoded into an understood format. Once decoded, all necessary operands are fetched from memory, the operands are then paired with their corresponding instruction, then they are issued for execution. The results are committed to memory, the next instruction is then fetched and the process repeats [4].

The architecture of the Bonnell completes this pipeline in order. The obvious draw back is that many of the steps within the pipeline are dependent on having the operands immediately available–this is not always possible. Required operands are often located in main memory, hundreds of clock cycles away from the CPU. During these cases, the efficiency of the pipeline drops because no work is being done until those operands are available. Therefore, out-of-order execution architectures attempt to fix this inefficiency by allowing some instructions to execute before others that are stalled waiting for data [4].

An out-of-order implementation comes with the penalty of increased chip size which translates to more power consumption. This is the reason the ARM11 micro-architecture has opted to keep its in-order design. However, Intel has the advantage of having a vertical integration business model, which primarily means Intel not only designs, but also fabricates its own chips [2]. Intel leads the industry in fabrication technology, which essential means Intel is usually a full generation ahead of its competition in terms of chip size. Intel has taken advantage of this and it is one of the key reasons Intel has been able to implement an out-of-order micro-architecture in its Atom chip. The power consumption that is increased in implementing out-of-order execution is compensated by the smaller chip size.

III. Comparative Analysis

The two previous examples of optimizations ARM and Intel have opted for have been to the micro-architecture. This should be some sort of indication that ARM and Intel have found that to significantly decrease power consumption, the microprocessors have to be optimized at the micro-architecture level. The following section will present empirical evidence that suggests this.

A. *Key Power Imacts of ISA*

Determining the key impacts of the ISA resulted in a variety of challenges which involved separating out the ISA-independent factors that contribute to power consumption. ISA-independent factors include: chip process technology, memory bandwidth, operating system, compiler and workloads executed [1]. These ISA-independent factors compound when considering an analysis of energy consumption since micro-processors implementing an ISA are part of a larger system on board, therefore separating out chip energy from board energy provided an additional challenge [1]. In addition, some micro-architecture features are required by the ISA while others are not, therefore this was another challenge.

In order to separate ISA-independent factors, the main study used for this paper considered several chips for each ISA with similar micro-architectures, use the same operating system and compiler front-ends and construct workloads that do not significantly rely on the operating system [1].

B. *Infrastructure*

The Beagleboard (Cortex-A8), Pandaboard (Cortex-A9) and the Atom board, were the three platforms used to test these two processors. All boards include processors with similar micro-architectural features like issue-width, caches, main memory and technology nodes [1]. The specifications can be viewed below:

	32/64b x86 ISA		ARMv7 ISA	
Architecture	Sandybridge	Atom	Cortex-A9	Cortex-A8
Processor	Core 2700	N450	OMAP4430	OMAP3530
Cores	4	1	2	1
Frequency	3.4 GHz	1.66 GHz	1 GHz	0.6 GHz
Width	4-way	2-way	2-way	2-way
Issue	OoO	In Order	OoO	In Order
L1 Data	32 KB	24 KB	32 KB	16 KB
L1 Inst	32 KB	32 KB	32 KB	16 KB
L2	256 KB/core	512 KB	1 MB/chip	256 KB
L3	8 MB/chip	—	—	—
Memory	16 GB	1 GB	1 GB	256 MB
SIMD	AVX	SSE	NEON	NEON
Area	216 mm^2	66 mm^2	70 mm^2	60 mm^2
Tech Node	32 nm	45 nm	45 nm	65 nm
Platform	Desktop	Dev Board	Pandaboard	Beagleboard
Products	Desktop	Netbook Lava Xolo	Galaxy S-III Galaxy S-II	iPhone 4, 3GS Motorola Droid

Fig. 2. Processor Comparison [1]

As for the operating system, across all platforms, Linux 2.6 LTS kernel was used. For the compiler, the study used gcc 4.4.

C. *Applications*

The influences of ARM and Atom processors in mobile and servers are the two categories this paper is most interested in. Therefore, it is important to note the applications that were used in both categories to benchmark the processors. For the mobile category, BBench, a web-page rendering benchmark which comprises 11 of the most popular websites on the web today was used [2]. The server category uses two benchmark applications, one for web search (CLucene) and one for web serving (Lighttpd) [2].

D. *Performance Analysis*

The data in this section will present the key findings from the main study, which include the performance, power and the trade-offs between them [1].

1) Instruction Count Comparison: The following chart shows the benchmarks that were run on CLucene. Specifically, the part of the chart we are most interested is the mobile and server comparison between ARM and x86 instructions. The key finding from this chart shows that the instruction count is similar accross the ISAs, which implies the gcc compiler pics RISC-like instructions [1]. Another key finding, according to the study, suggests that ”the x86 ISA overheads, if any, are overcome by micro-architecture” [1].

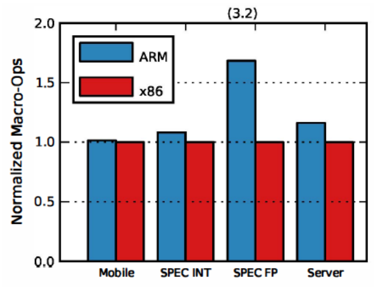

Fig. 3. Macro-Ops [1]

2) Instruction Format Mix: For the instruction mix sizes for mobile and server shown in the chart below, two key findings are presented. One is that the average ARM and x86 binary instruction sizes are similar for mobile workloads, which suggests ARM and x86 have similar code densities. Another finding is that the x86 executed instructions are approximately 25 percent shorter than ARM instructions.

Table 6. Instruction Size Summary.

		(a) Binary Size (MB)		(b) Instruction Length (B)	
		ARM	x86	ARM	x86
Mobile	Minimum	0.02	0.02	4.0	2.4
	Average	0.95	0.87	4.0	3.3
	Maximum	1.30	1.42	4.0	3.7
Desktop INT	Minimum	0.53	0.65	4.0	2.7
	Average	1.47	1.46	4.0	3.1
	Maximum	3.88	4.05	4.0	3.5
Desktop FP	Minimum	0.66	0.74	4.0	2.6
	Average	1.70	1.73	4.0	3.4
	Maximum	4.75	5.24	4.0	6.4
Server	Minimum	0.12	0.18	4.0	2.5
	Average	0.39	0.59	4.0	3.2
	Maximum	0.47	1.00	4.0	3.7

Fig. 4. Instruction Mix [1]

3) Power Analysis: Looking at the mobile portion of the chart below, when frequency and technology are scaled to 1GHz and 45nm, which essentially limits the Cortex A8 as the unit of measure, the ISA appears irrelevant for the Cortex A8, A9 and Atom. The Intel Atom and the Cortex A9 fall within 60 percent of the power consumed by the A8. Surprisingly, the Atom is slightly less power hungry than the A9.

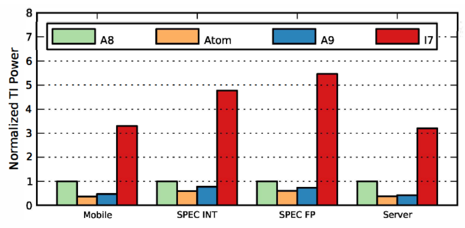

Figure 12. Tech. Independent Avg. Power Normalized to A8.

Ratio	Mobile	SPEC INT	SPEC FP	Server
Atom to A8	0.6	0.6	0.6	0.6
i7 to A9	7.0	6.1	7.4	7.6

Fig. 5. Power Analysis [1]

Lastly, the following chart illustrates the power/performance trade-offs of each processor. Micro-architecture has a high energy cost when performance is low. Regardless the energy efficiency or ISA of either processors, higher performance processors require more energy than low performance processors [1].

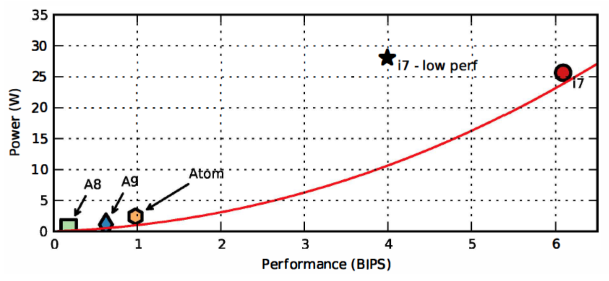

Fig. 6. Power/Performace Trade-offs [1]

IV. Conclusion

Company optimization trends, as well as empirical evidence suggest that the ISA does not play as significant a role in the power consumption of a processor as conventional wisdom might lead one to believe. For today's mature microprocessor design world, the ISA of a processor, RISC or CISC seems largely irrelevant.

References

[1] Blem, E.; Menon, J.; Sankaralingam, K., "Power struggles: Revisiting the RISC vs. CISC debate on contemporary ARM and x86 architectures," High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on , vol., no., pp.1,12, 23-27 Feb. 2013 doi: 10.1109/HPCA.2013.6522302

[2] Bourzac, K.; Guy, E., "Intel inside...your smartphone [2013 Tech To Watch]," Spectrum, IEEE , vol.50, no.1, pp.56,62, Jan. 2013 keywords: Companies;Games;Graphics;Industries;Microarchitecture;Mobile communication;Program;processors;, doi: 10.1109/MSPEC.2013.6395314

[3] George, A.D., "An overview of RISC vs. CISC," System Theory, 1990., Twenty-Second Southeastern Symposium on , vol., no., pp.436,438, 11-13 Mar 1990 doi: 10.1109/SSST.1990.138185 keywords: computer architecture;reduced instruction set computing;CISC;RISC;complex-instruction-set-computer;computer architecture;computer processor technology;reduction-instruction-set-computer;Computer aided instruction;Delay;Educational institutions;Instruction sets;Microprocessors;Optimizing compilers;Pipelines;Process design;Reduced instruction set computing;Registers,

[4] Gargini, P.A., "The global route to future semiconductor technology," Circuits and Devices Magazine, IEEE , vol.18, no.2, pp.13,17, Mar 2002 doi: 10.1109/101.994854 keywords: electronics industry;integrated circuit manufacture;integrated circuit technology;monolithic integrated circuits;semiconductor technology;2001 ITRS;ASIC;DRAM;,

[5] Lal Shimpi. Anand, May 6, 2013. Intel's Silvermont Architecture Revealed: Getting Serious About Mobile. Available: http://www.anandtech.com/show/6936/intels-silvermont-architecture-revealed-getting-serious-about-mobile/2

[6] Neagoe, T.; Karjala, E.; Banica, L., "Why ARM processors are the best choice for embedded low-power applications?," Design and Technology in Electronic Packaging (SIITME), 2010 IEEE 16th International Symposium for , vol., no., pp.253,258, 23-26 Sept. 2010 doi: 10.1109/SIITME.2010.5650194 keywords: cache storage;embedded systems;low-power electronics;microprocessor chips;ARM architecture;ARM processor;,