Evaluating the Impact of Replacement Policies on the Cache Performance and Energy Consumption in Different Multicore Embedded Systems

Sajjad Rostami-Sani, Mojtaba Valinataj, Amir-Hossein Khojir-Angasi

Abstract**—**The cache has an important role in the reduction of access delay between a processor and memory in high-performance embedded systems. In these systems, the energy consumption is one of the most important concerns, and it will become more important with smaller processor feature sizes and higher frequencies. Meanwhile, the cache system dissipates a significant portion of energy compared to the other components of a processor. There are some elements that can affect the energy consumption of the cache such as replacement policy and degree of associativity. Due to these points, it can be inferred that selecting an appropriate configuration for the cache is a crucial part of designing a system. In this paper, we investigate the effect of different cache replacement policies on both cache's performance and energy consumption. Furthermore, the impact of different Instruction Set Architectures (ISAs) on cache's performance and energy consumption has been investigated.

Keywords**—**L1-cache, energy consumption, replacement policy, Instruction set architecture, multicore processor.

I. Introduction

EMBEDDED systems are basically application-specific systems. These system products are widely used in mobile devices such as smartphones and tablet PCs. In these devices, caches have been widely used to fill the performance gap between memories and processors. Thus, the information evoked from the memory is kept into a quick and small cache system. In the case of recalling the same items, the processor directly searches the cache as a high-speed memory instead of the whole main memory, and as the result, the system performance improves significantly via using the cache memory [1]. In addition, embedded systems are usually produced as portable devices or embedded in devices in which the required energy for processing is provided by batteries. Therefore, the energy consumption is an important issue in these systems. One of the modules that consume a lot of energy is the cache. Thus, if the energy consumption of the cache decreases, the total energy consumed by the system will decrease as well [2]. Hence, computer architects should investigate the power consumption of the caches with different configurations and choose the most optimum configuration for it to alleviate the cache power consumption as much as it is possible. In a cache, there are several parts that could be effective on its energy consumption such as the cache's replacement policy and the degree of associativity. The impact of different replacement policies on cache's energy consumption is outstanding; also, these policies are effective in the case of system's performance [3]-[7]. Regarding to these facts and Fig. 1, choosing an appropriate replacement policy for the cache especially in multicore systems is a critical part of a system designer's task. In this work, we investigate the effect of different replacement policies on caches with different configurations.

The rest of this paper is organized as follows. Section II reviews some existing works that are about the evaluating the impact of the cache replacement policies on the system's performance. The structure of the cache and the effect of its different parts on its performance are given in Section III. Section IV details the simulation environment and presents the experimental results for the evaluation of the cache with different replacement policies and configuration. Finally, some conclusions are drawn in Section V.

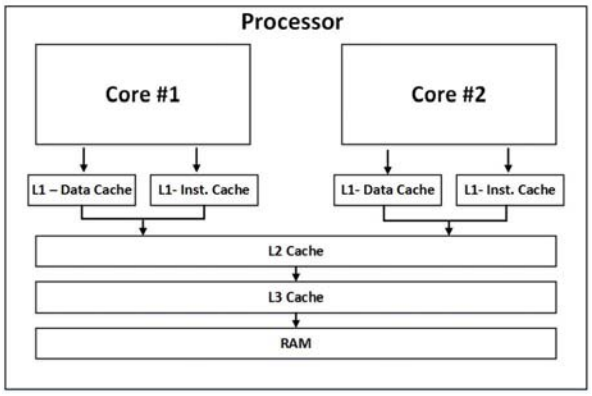


Fig. 1 The structure of the cache in multicore systems (2-core processor)

II. Related Works

There exist a variety of previous works focused on the effect of cache replacement policies and different configurations on the system's performance. However, to the best of our knowledge, none of the previous works focus on issues such as energy consumption and the impact of ISAs.

In [8], Al-Zoubi et al. tried to find the replacement policy as

Sajjad Rostami-Sani is with the Department of Electrical and Computer Engineering, Ryerson University Toronto, Canada (e-mail: srostamisani@ryerson.ca).
Mojtaba Valinataj is with the Department of Computer Engineering, Babol Noshirvani University of Technology, Babol, Iran (e-mail: m.valinataj@nit.ac.ir).
Amir-Hossein Khojir-Angasi is with the Department of Electrical Engineering, Golestan University, Gorgan, Iran (e-mail: a.khojirangasi@gmail.com).

close to optimal as possible; they thoroughly explored the design space of existing replacement mechanisms using SimpleScalar toolset, across a wide range of cache sizes and organizations. In order to better understand the behavior of different policies, they introduced new measures, such as a cumulative distribution of cache hits in the LRU stack. They also dynamically monitored the number of cache misses, per each 100000 instructions. In [9], they analyzed various factors of effects on cache performance, and then carried out the simulation experiment of cache performance based on SimpleScalar tools set and SPEC2000 benchmark suite. They compared the effects on cache performance when L1-cache, L2-cache capacity, and replacement methods are changed separately, and then they analyzed the experimental results in detail.

In [10], Grund et al. presented an efficient method to estimate the miss ratio using a stochastic model. The model takes into account the parameters of the cache architecture and a concise characterization of the software's locality. In that work, they considered the replacement policy as an important component of the cache architecture. To this goal, they introduced policy tables as a concise representation of replacement policies. In [11], Zahran at first showed that local replacement policies may not always be the correct way to go for obtaining the most efficient cache hierarchy. Second, they proposed several global replacement policies and discussed their behavior with several benchmarks using a cycle accurate simulator. Third, they showed that for some benchmarks, the global replacement schemes do not perform much better than their local counterparts, and they discussed the characteristics of an application that can benefit from the global schemes.

III. The Effective Factors of Cache

A. Cache Size

The size of the cache has an important impact on its performance (hit and miss ratio). By increasing the cache size, the amount of data that can be kept on the cache will increase and therefore, the hit rate will grow and then, it would have a positive effect on system's performance. Unfortunately, by growing the cache, the energy that could be consumed by the cache will increase and the system's overall energy consumption will increase significantly. Due to these facts, one of the greatest tasks of the system's designer is to find the best size for the cache to convince both performance and energy.

B. Associativity Degree

The associativity degree is the number of cache mirrors that can save the data. The structure of the cache could be uniform or not. In the case of uniform structure, all data are kept in a single cache and all operation will be done on it. On the other hand, the data are saved on various sub-structures. The greater the number of cache mirrors, the cache's miss rate will decrease and therefore, its performance will increase; but by increasing the mirrors, more comparators and gates are needed, thus the energy consumption will increase. These points clearly illustrate the importance of choosing the optimum number for associativity degree.

C. Instruction Set Architecture

One of the most crucial parts of a system is its ISA that plays a key role in the performance and energy consumption of the whole system. The ARM and X86 are two different ISAs that are widely used in modern systems and each of these ISAs has its advantages and disadvantages. The ARM is based on RISC architecture and it is prominent for its low-power attitude that could be an outstanding characteristic especially in embedded systems. On the other hand, the x86 is well-known for its CISC architecture and its high performance. With the increasing demands for new devices with different goals, the issue of ISA selection became a challenging issue for system's designers.

D. Replacement Policy

The data that are kept on the cache is originally from memory. Adding different data on the cache is a simple operation until the cache has some empty rooms. But, when the cache be filled and there is not any place for new data, the system needs a replacement algorithm for finding a room for new data. In the case of new data coming in a full cache, these new data should be replaced by some old data. The operation of choosing the appropriate data for substitution is done by replacement policies. There are variant replacement policies such as LFU, MRU, FIFO, and Random that work in different manners.

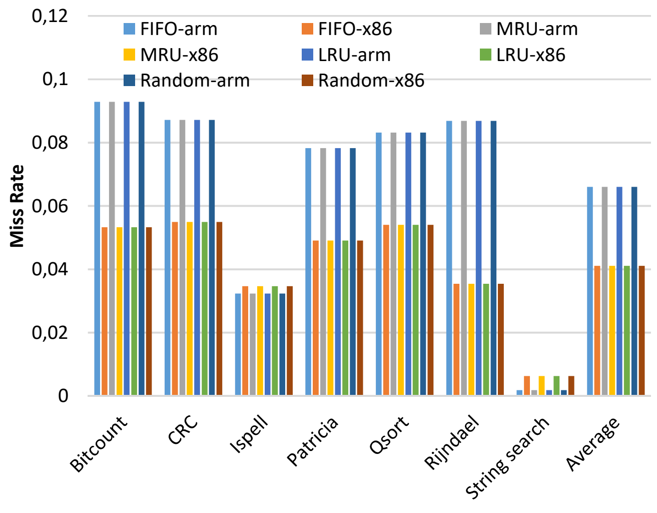
IV. Experimental Results

In this section, in the beginning, the simulators that are used for evaluating the performance and energy consumption will be introduced and then, the benchmark that is used will be discussed. In the second part, the results of simulations would be presented and we will discuss the impact of different parameters on cache's performance and energy consumption. In this work, the results belong to Level 1 D-cache and cache's miss rate is used as an indicator of performance.

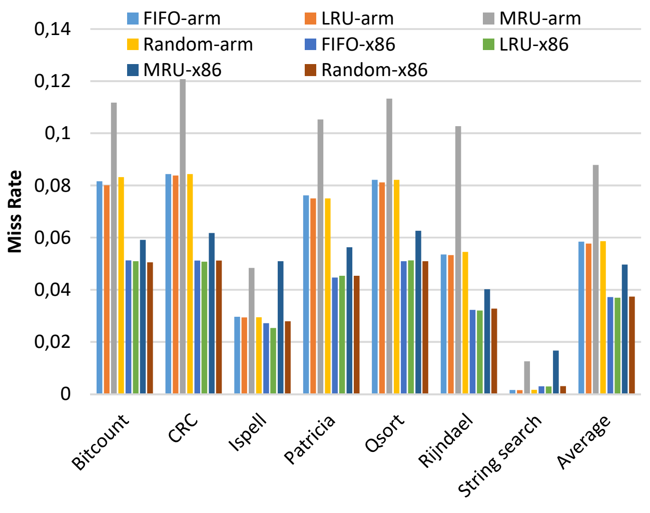
A. Simulation Setup

In this work, the Gem5 simulator [12] is utilized to evaluate the performance of the cache. This simulator can simulate a whole system with different devices and an operating system in two modes, full system mode (FS mode) and syscall imitation mode (SE mode). There are various levels of support for executing the processors including ARM, Alpha, Power, MIPS, SPARC, and 64-bit x86 binaries on CPU models consist of two simple single CPI models, an out of order model (O3), and an in-order model. In our simulations, Gem5 is arranged to the ARM and X86 versions, which are typical embedded processors. The parameters of the main system are shown in Table I.

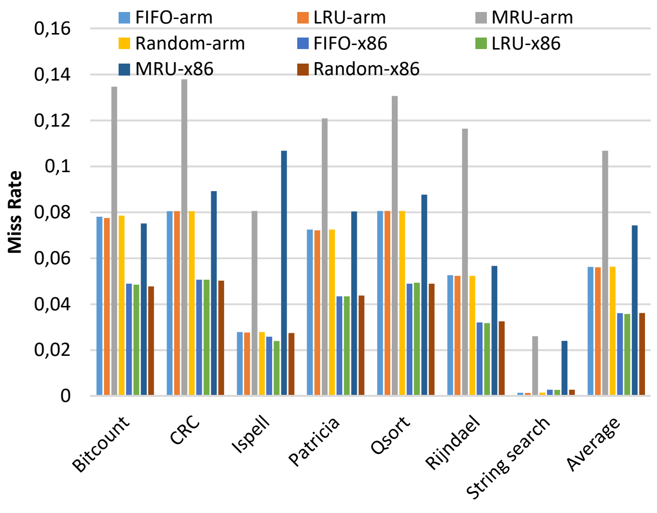
The CACTI 6.5 cache model is used in this paper with 22 nm technology size to estimate the dynamic energy consumption of the proposed method [13], [14]. CACTI can evaluate the energy consumption of cache memory system based on the cache memory parameters such as the block size, cache memory size, degree of associativity, technology size and etc. In this paper, the dynamic energy information of the proposed method is obtained from CACTI.



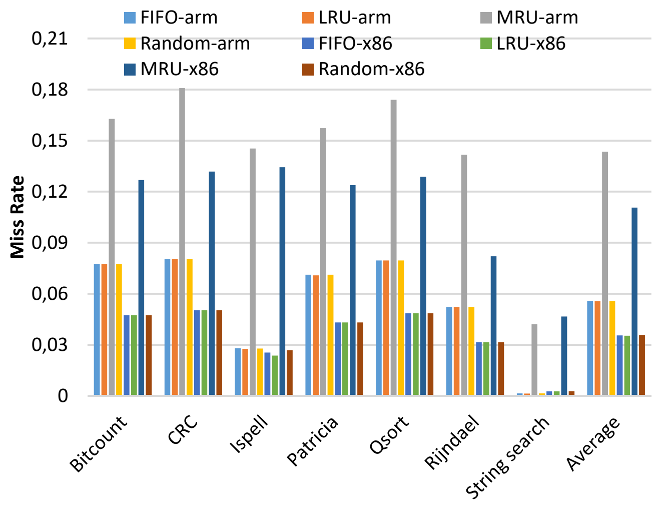
(a) Direct-map



(b) 2-way



(c) 4-way



(d) 8-way

Fig. 2 The effect of different replacement policies on performance of the cache (cache size: 16 KB)

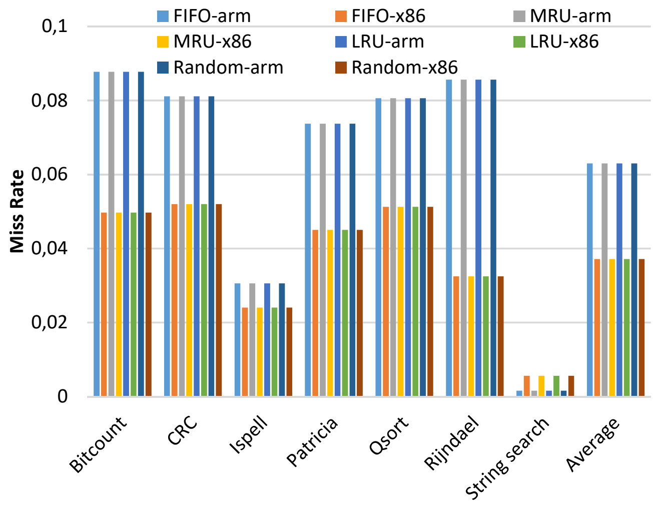
TABLE I
SYSTEM CONFIGURATION PARAMETERS USED IN THE SIMULATIONS

Parameter	Value
ISA	ARM, X86
CPU clock speed	2.4 GHz
Core #	4
Instruction cache size	16 KB
Data cache size	16 KB, 32 KB, 64 KB
Associativity	Direct map, 2-way, 4-way, 8-way
Technology size	22 nm
Replacement policy	LRU, MRU, FIFO, Random
Block size	64 bytes
L2 cache size	512 KB
L3 cache size	8 MB

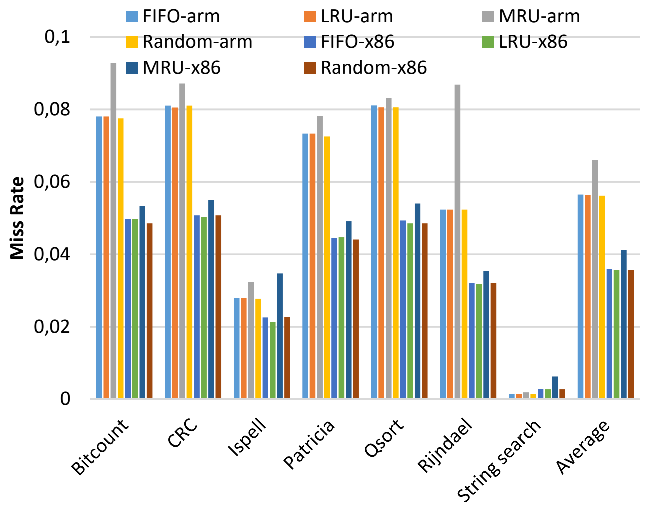
TABLE II
SELECTED PROGRAMS FROM DIFFERENT CATEGORIES OF MIBENCH

Category	Program	Description
Automotive	Bitcount	An algorithm that checks the bit manipulation
	Qsort	A well-known sorting algorithm
Network	Patricia	A data structure that is used in trees with sparse nodes
	Dijkstra	An algorithm for finding the shortest path
Office	Ispell	A fast spell checker
	Stringsearch	A program that searches for given words
Security	Blowfish	A block cipher by a length key
	Rijndael	An algorithm for standard encryption
Telecomm	Crc	A program for cyclic redundancy check
	FFT	A program for fast Fourier transform

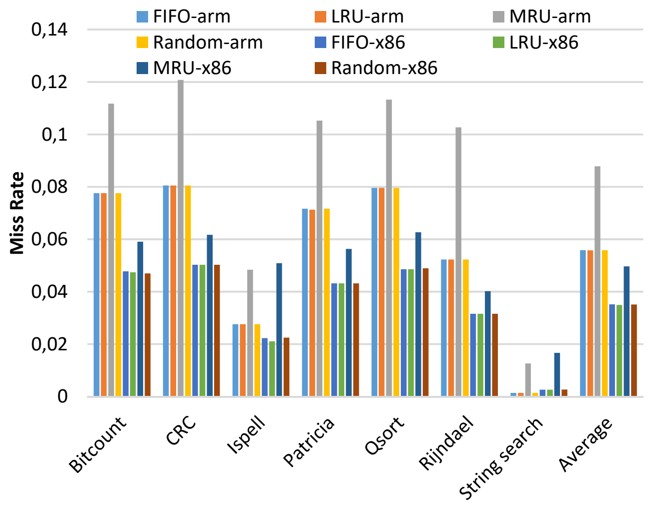
The application programs used for all simulations are from the MiBench benchmark [15]. This benchmark is composed of different categories that support various applications of embedded systems. The details of utilized categories and programs from the MiBench benchmark are depicted in Table II. All the programs are compiled in cross compiler manner with the static compiler flag.



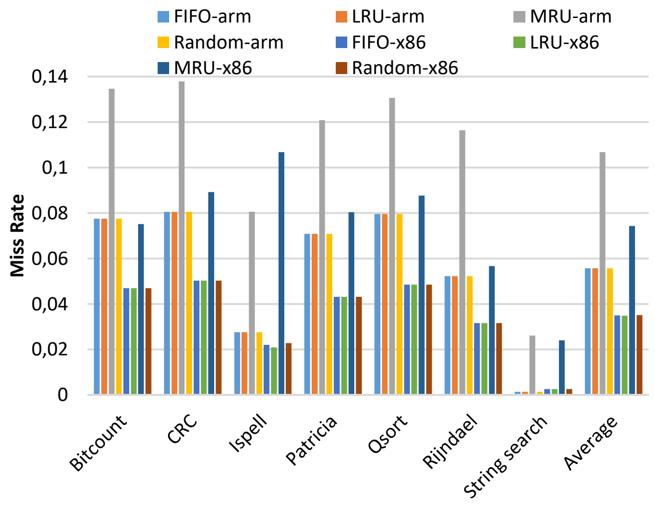
(a) Direct-map



(b) 2-way



(c) 4-way



(d)8-way

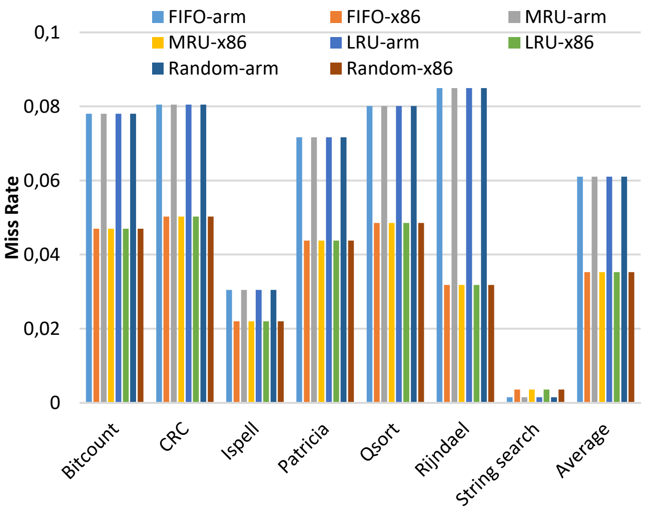
Fig. 3 The effect of different replacement policies on performance of the cache (cache size: 32 KB)

B. Simulation Results

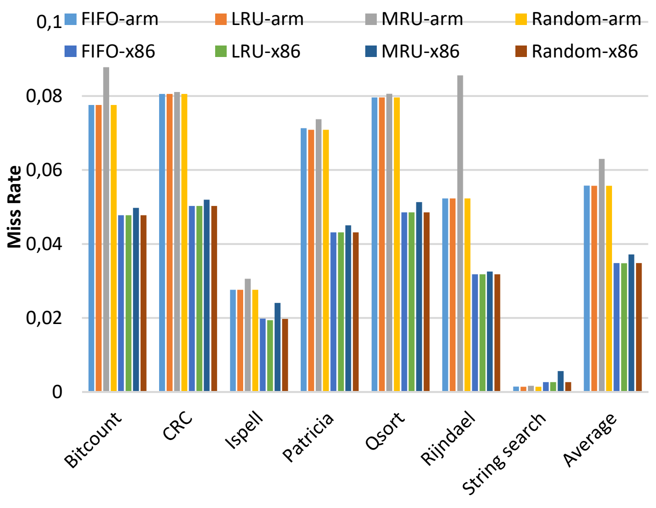
In this section, at first, the effect of the different replacement policies on the performance in the form of cache miss rate is estimated. Then, their effect on the cache energy consumption with different configuration is evaluated. In these sections, the simulations are investigated in both x86 and ARM.

1) Replacement policy Versus Performance

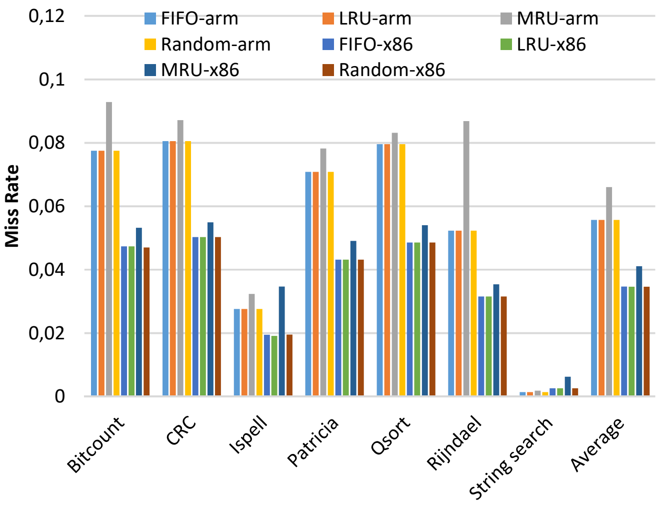
Fig. 2 shows the miss rate of different benchmark programs in 22 nm feature size for a 16 KB L1 data cache with different degrees of associativity (Direct-map, 2-way, 4-way, and 8-way). In this figure, despite the fact that the replacement policies have different impacts on programs' performances with different configurations, some general results can be stated. Due to Fig. 2, the replacement policies are ineffective on the performance of Direct-map caches. The reason is that, in Direct-map caches, the exact location of each data is predefined and the data cannot be kept in other places and therefore, this conclusion can be reached that in Direct-map caches, there is no difference between various replacement policies. By increasing the degree of associativity, the changes in the acache's performance are perceptible. As it can be inferred from this figure, caches with higher associativity degrees have better performance in comparison with caches with lower degrees of associativity. By observing the results, it can be found that some of the replacement policies have better effects on a cache's performance. For example, with regard to the results, in most of the cases, when the cache is simulated with an LRU replacement policy, the system's performance is better than other policies. But, it can be worth mentioning that the LRU is not the best policy in all cases. As an instance, in a 4-way cache, in the case of the Patricia program, the results of the Random replacement policy are better than other policies. Based on these facts, it cannot be concluded that a replacement policy is the best for all caches with different configurations; but, it is obvious that the results of the LRU policy are outstanding. The impact of different replacement policies of the cache on its performance with 32 KB and 64 KB as cache size is shown in Figs. 3 and 4, respectively.



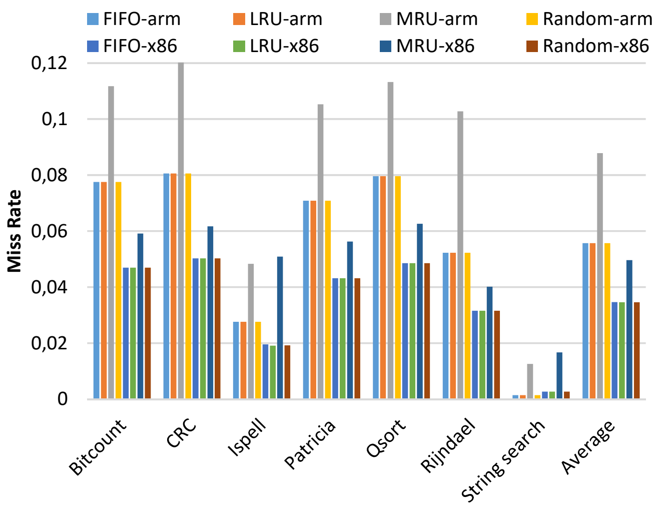
(a) Direct-map



(b) 2-way



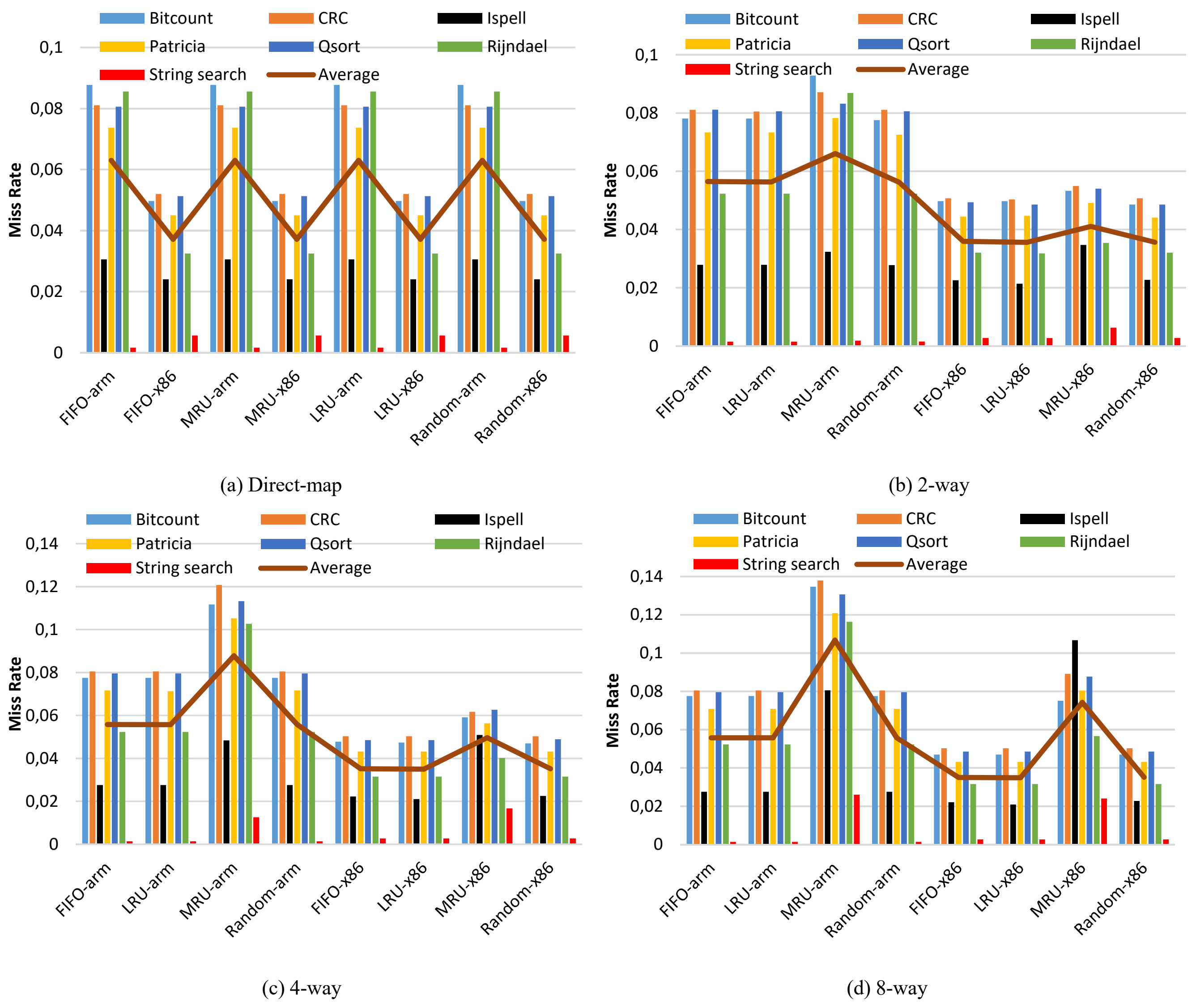
(c) 4-way



(d) 8-way

Fig. 4 The effect of different replacement policies on performance of the cache (cache size: 64 KB)

To evaluate the effect of the different replacement policies on the performance of the cache with different ISAs, the cache is simulated considering different replacement policies (FIFO, LRU, MRU, and Random) with different cache associativity degrees from Direct-map to 8-way and various ISAs (ARM and x-86). As shown in Fig. 5, the ISA has a direct impact on miss rate of the cache and when the ISA is changed from ARM to x86, the performance is altered significantly; the reason is that, the ARM is RISC-based, which means that it is focused on keeping the instructions simple and therefore, more instructions are needed for doing tasks. Unlike the ARM, the x86 that is CISC-based is focused on performing complex instructions with high flexibility. This figure indicates that by changing the replacement policy from ARM to x86, the performance will increase remarkably. For example, when a 2-way cache with FIFO policy is simulated, the average amount of miss rate is about 0.056, by changing the ISA, the miss rate changed to 0.035 that is a significant reduction. With regard to these facts and figures, it can be concluded that x86 could be a good choice for system's designers that want to have the best performance in their systems. However, in some cases, the energy consumption is a crucial element that designer should consider that. The next sub-section will discuss the impact of replacement policies and ISAs on the energy consumption of the systems.



(a) Direct-map

(b) 2-way

(c) 4-way

(d) 8-way

Fig. 5 The effect of different ISAs and replacement policies on performance of the cache

TABLE III
THE IMPACT OF DIFFERENT REPLACEMENT POLICIES ON CACHE'S ENERGY CONSUMPTION
(A) LRU AND FIFO AS REPLACEMENT POLICY

MiBench Programs	Cache energy consumption (nJ)															
	LRU								FIFO							
	1-way		2-way		4-way		8-way		1-way		2-way		4-way		8-way	
	ARM	X86	ARM	X86	ARM	X86	ARM	X86	ARM	X86	ARM	X86	ARM	X86	ARM	X86
Bit count	55.3	69.2	56.3	70.6	64.7	76.7	67.5	78.1	55.3	69.2	56.4	70.7	64.8	76.7	67.9	78.5
CRC	51.3	64.4	52.3	65.6	61.5	72.7	64.2	73.9	51.3	64.4	52.3	65.7	61.5	72.7	64.3	74
Ispell	196.7	1394.4	199.8	1417.3	220.7	1550	256.7	1586.5	196.7	1394.4	199.8	1418.1	221.6	1550.4	266.4	1586.8
Patricia	68.33	86.3	69.6	87.9	75.4	90.7	78.5	92.5	68.3	86.3	69.6	87.9	75.4	90.7	79.5	92.5
Qsort	55.5	69.4	56.5	70.7	64.9	76.8	68.6	78.2	55.4	69.3	56.5	70.7	64.9	76.8	69.6	78.2
Rijndael	89.8	111	87.8	113.2	90.4	119.4	94.5	127.3	89.8	111	87.8	113.2	90.4	119.4	96.2	128.1
String search	4374.6	1895.9	4462.6	1913.8	4648.2	2078.5	4781.1	2135.4	4374.6	1895.9	4462.7	1913.8	4648.2	2078.5	4781.1	2137.9

(B) MRU AND RANDOM AS REPLACEMENT POLICY

Cache energy consumption (nJ)																
MiBench Programs	MRU								Random							
	1-way		2-way		4-way		8-way		1-way		2-way		4-way		8-way	
	ARM	X86	ARM	X86	ARM	X86	ARM	X86	ARM	X86	ARM	X86	ARM	X86	ARM	X86
Bit count	55.3	69.2	57.1	70.8	65.4	77.1	66.7	78.6	55.3	69.2	56.4	70.6	64.8	76.7	65.5	78.1
CRC	51.3	64.4	52.4	65.8	61.8	72.9	63.4	74.4	51.3	64.4	52.3	65.7	61.5	72.7	62.2	73.9
Ispell	196.7	1394.4	200.6	1426.4	212.4	1467.6	218.9	1514.1	196.7	1394.4	199.8	1417.9	201.7	1550.4	206.4	1686.6
Patricia	68.3	86.3	69.9	88.2	75.8	91.1	88.2	93.2	68.3	86.3	69.6	87.9	75.4	90.7	76.5	92.5
Qsort	55.5	69.4	56.6	70.9	65.1	77.1	66.7	78.7	55.5	69.4	56.5	70.7	64.9	76.8	65.6	78.2
Rijndael	89.8	111	91.6	113.3	92.7	119.8	94.6	124.6	89.8	111	87.8	113.2	90.4	119.4	92	121.8
String search	4374.6	1895.9	4464.1	1939.9	4649.8	2084.1	4792.4	2152.4	4374.7	1895.9	4462.7	1931.8	4648.2	2078.5	4781	2135.4

2) Energy versus Replacement Policy

Table III illustrates the effect of replacement policy and system's ISA on a cache's energy consumption and indicates that the policy has a direct impact on the energy consumption of the cache. Based on this table, it can be inferred that when the policy is changed from a specific one (LRU) to another, in most cases, the energy consumption increased. For example, in a 2-way cache, when the policy is changed from LRU to MRU, the cache's energy consumption increased in most of the programs. Based on this table and the system's configuration, the best policy for the cache could be selected. Another important point that can be reached based on Table III is that the ISA plays a crucial role in the energy consumption of the cache. As it can be inferred from this table, in the case of x86 as the ISA, the energy consumption is much more than the ARM. Earlier, it was discussed the positive effect of the x86 ISA on system's performance but, due to this table, it can be concluded that the x86 is not the best choice in the case of energy and the ARM is far better than it.

V. CONCLUSION

In this paper, the effect of various replacement policies on cache's performance and energy consumption has been evaluated. The results indicate that none of the cache's policy is the best in all cases, but, in most of the cache configurations, the LRU policy is the best. Due to this paper's results, a system's designer can choose the most optimum replacement policy with regard to the cache's configuration. Moreover, the results could be useful for designers to decide between ARM and x86 as the system's ISA. Based on the simulations, the ARM could be a good choice in the case of energy consumption, but, the results of x86 is more convincing in the case of performance.

REFERENCES

[1] J. L. Hennessy, D. A. Patterson. "Computer architecture: a quantitative approach." Elsevier; 2011.
[2] H. Esmaeilzadeh, T. Cao, X. Yang, S. Blackburn, K. McKinley, "Looking back to the language and hardware revolutions: measured power, performance, and scaling", Proc. 16th Int'l. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2011.
[3] A. Vakil-Ghahani, S. Mahdizadeh-Shahri, M. Lotfi-Namin, et al. "Cache replacement policy based on expected hit count", IEEE computer architecture letters, 2017.
[4] R. Olanrewaju, A. Baba, B. Khan, et al. "A study on performance evaluation of conventional cache replacement algorithms: A review", International conference on parallel, distributed and grid computing (PDGC), 2016.
[5] D. Swain, S. Marar, N. Motwani, et al. "CWRP: An efficient and classical weight ranking policy for enhancing cache performance", International conference on image information processing (ICIIP), 2017.
[6] G. Einziger, R. Friedman, B. Manes, "TinyLFU: A highly efficient cache admission policy", ACM Transactions on storage (TOS), 2017.
[7] J. Reineke, D. Grund, C. Berg, R. Wilhelm. "Timing predictability of cache replacement policies", Real-Time Systems, Volume 37, Issue 2, 2007.
[8] H. Al-Zoubi, A. Milenkovic, M. Milenkovic. "Performance Evaluation of cache Replacement Policies for the SPEC CPU2000 Benchmark Suite", In Proc. of the 42nd ACM Southeast Conf, April 2004.
[9] M. Hai-feng, Y. Nian-min, F. Hong-bo. "Cache Performance Simulations and Analysis under Simplescalar Platform", International conference on new trends in information and service science, 2009.
[10] D. Grund, J. Reineke, "Estimating the Performance of Cache Replacement Policies", IEEE international conference on formal methods and models for Co-Design, 2008.
[11] M. Zahran, "Cache Replacement Policy", Proc. Annual Workshop on Duplicating, Deconstructing, and Debunking (WDDD) held in conjunction with the international Symposium on Computer Architecture (ISCA), 2007.
[12] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, et al. "The gem5 simulator", ACM SIGARCH Computer Architecture News, 39 (2) (2011) 1-7.
[13] N. Muralimanohar, R. Balasubramonian, N. P. Jouppi. CACTI 6.0: A tool to model large caches. HP Laboratories, 22-31, 2009.
[14] CACTI, An integrated cache and memory access time, cycle time, area, leakage, and dynamic power model, Available: <http://www.hpl.hp.com/research/cacti>.
[15] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, R. B. Brown. MiBench: a free, commercially representative embedded benchmark suite. IEEE International Workshop on Workload Characterization, WWC-4. pp. 3-14, 2001.