Proposal New Cache Coherence Protocol to Optimize CPU Time through Simulation Caches

Dr. Luma Fayeq Jalil
Department of Computer Sciences, University of Technology /Baghdad
Dr. Maha Abdulkareem .H. Al-Rawi
Department of Computer Sciences, University of Technology /Baghdad
Abeer Diaa Al-Nakshabandi
Distribution office At Ministry of Electricity/Baghdad
Email:abeerdiaaphd@gmail.com

Received on:11/5/2016 & Accepted on:20/10/2016

ABSTRACT

The cache coherence is the most important issue that rapidly affected the performance of a multicore processor as a result of increasing the number of cores on chip multiprocessors and the shared memory program that will be run on these processors. "Snoopy protocols" and "directory based protocols" are two types of protocols that are used to achieve coherence between caches. The main objective of these Protocols is to achieve consistency and validation of the data value in the caches of a multi core processor so that any reading of a memory address via any caches will returns the latest data written to that address.

In this paper, a new protocol has been designed to solve a problem of a cache coherence that combines the two schemes of coherency: snooping and directory depending on the states of MESI protocol. The MESI protocol is a version of the snooping cache protocol which based on four (Modified, Exclusive, Shared, Invalid) states that a block in the cache memory can have. The proposed protocol has the same states of MESI protocol but the difference is in laying the directory inside a shared cache instead of main memory to make the processor more efficient by reducing the gap between fast CPU and slow main memory.

Keywords: Cache coherence problem, snooping protocol, Directory-Based cache Protocols, MESI, Cache Simulator, Dev. C++, Multiprocessor, shared memory.

INTRODUCTION

Shared memory is the hardware part that supported by many modern computer systems and multicore chips. Each of the processor cores in a shared memory system may read and write a single address space [1]. But in designing shared system, one of the most important problems appears which is called coherence problem. The coherence problem results when the caches are laid in recent computers between processor and main memory to solve the contention problem as trying to access a shared memory at the same time which then causes performance degradation [2].

Two hardware-based protocols are used to solve coherence problem that appeared in a shared memory system of a multicore processor with caches that can store multiple copies of memory blocks simultaneously which are "**Snooping protocols**" and "**Directory-based protocols**" [3,4,5].

Memory Hierarchy

The basic idea to overcome the problem of increasing the gap between a fast CPU and a slow RAM is in using a hierarchy of memories as in figure (1). Each level speedier, more expensive and smaller, the closer it is to CPU, to feed the CPU with the required data [6].

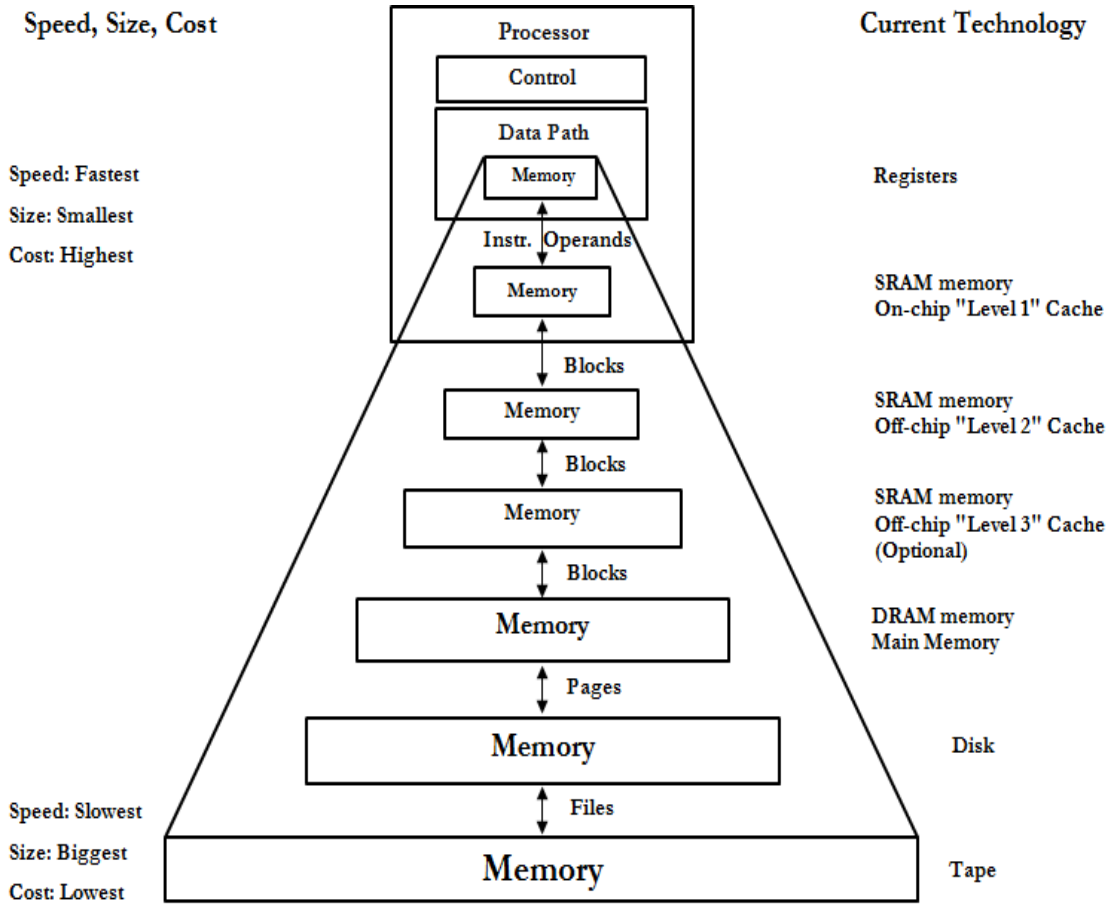

Figure 1: The Memory Hierarchy [7, 8 , 9]

Protocols for Cache Coherence

Two hardware-based protocols to coherence the caches in multiprocessor systems are used which include**:-**

Snooping Protocol

To solve the problem of cache coherence by snoopy protocol, the central bus is used as a "broadcast medium" which make the transactions on bus visible to all` caches **[11,12].** As a result the cache controllers of all processors can observe all memory accesses (figure 2) **[5,7]:**

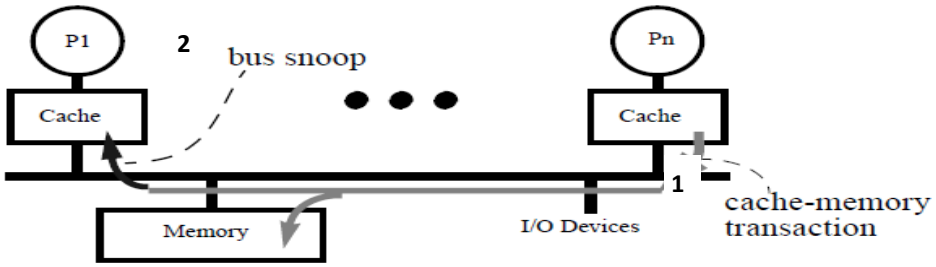

Figure 2: Snoopy Protocol **[13]**

These protocols are called "**update-based protocols**" when updated is performed directly by the cache controllers. Also "**invalidation-based protocols"** occur when the cache block that match memory block is invalidated and as a result main memory must update next read **[3].**

Directory Based Protocol

The ability of scaling in directory based schemes is better than snooping because it does not depend upon a shared bus for communication. The directory which can be central or distributed keeps state of all memory block shared between processors and then the cache controller uses point-to-point messages looking up directory instead of observing shared broadcast to get memory block state **[3, 10]**. (Figure 3). Although the directory-based protocols will likely have to be employed for multi core architectures of the future, there exist a drawbacks that appears in a directory which are: storage overhead, frequent indirections, and are more prone to design bugs **[14,15,16]**.

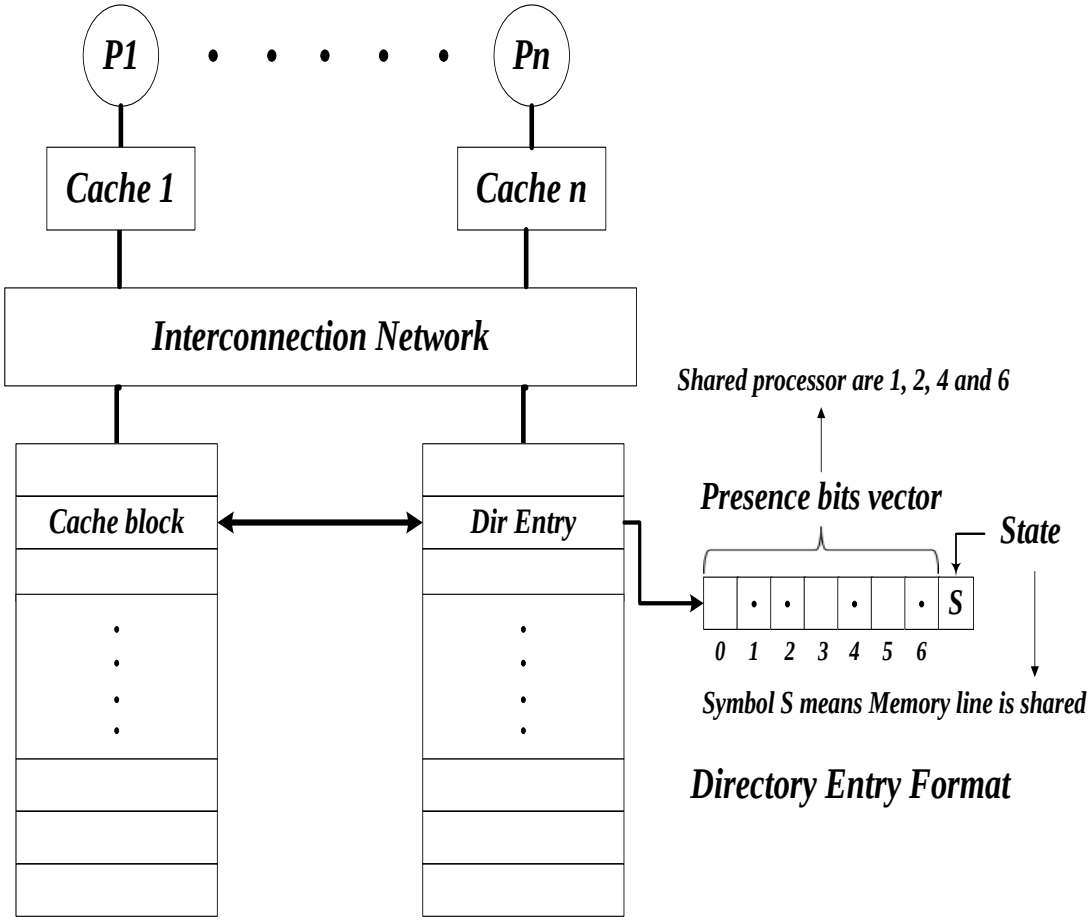

Figure 3: Directory Based Protocols **[6]**

Preprocessing steps to proposal protocol

A Preprocessing steps before beginning proposal protocol are illustrated as in Figure 4, Figure 5, Figure6 respectively:

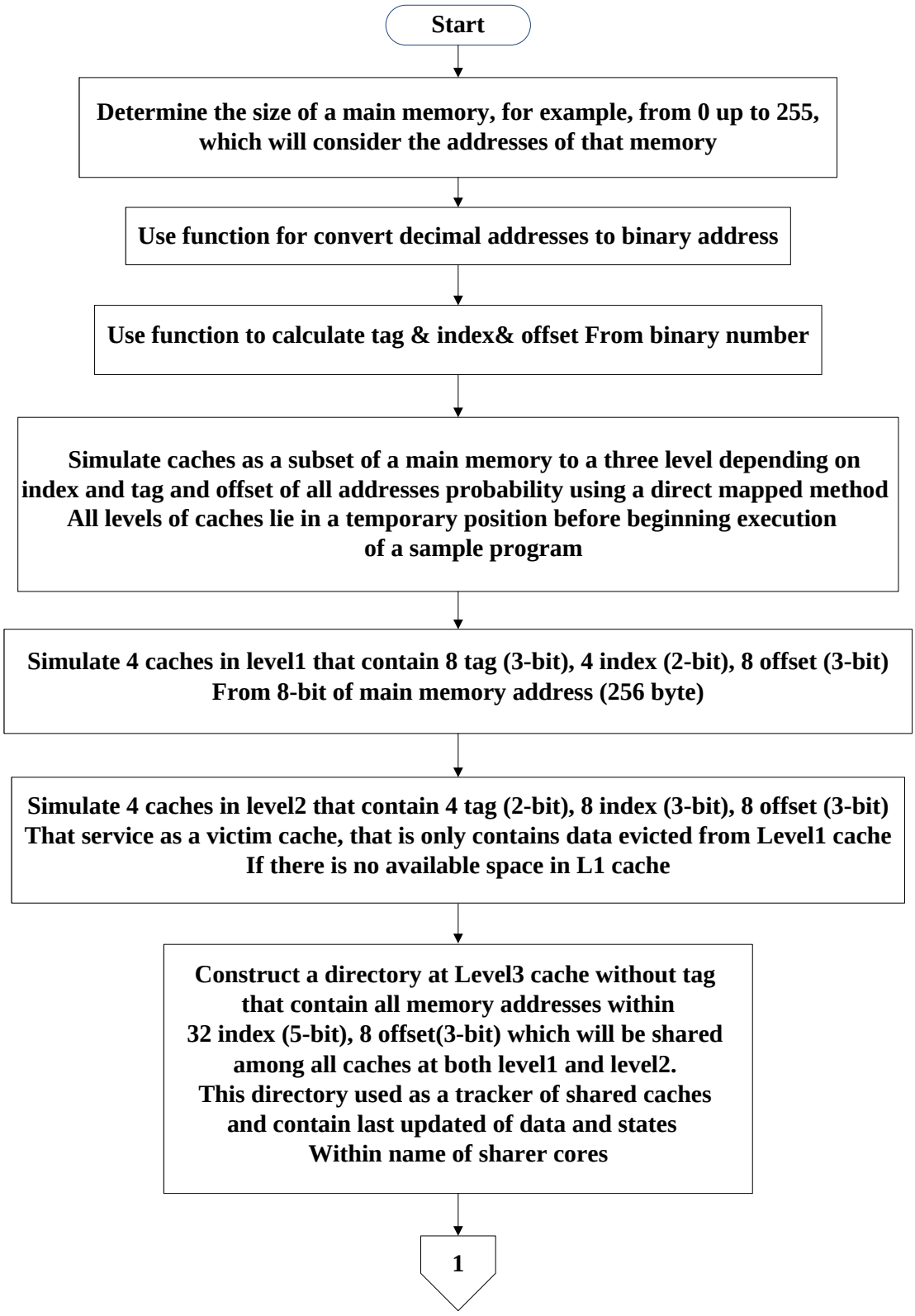

Figure 4: preprocessing steps of a proposed protocol

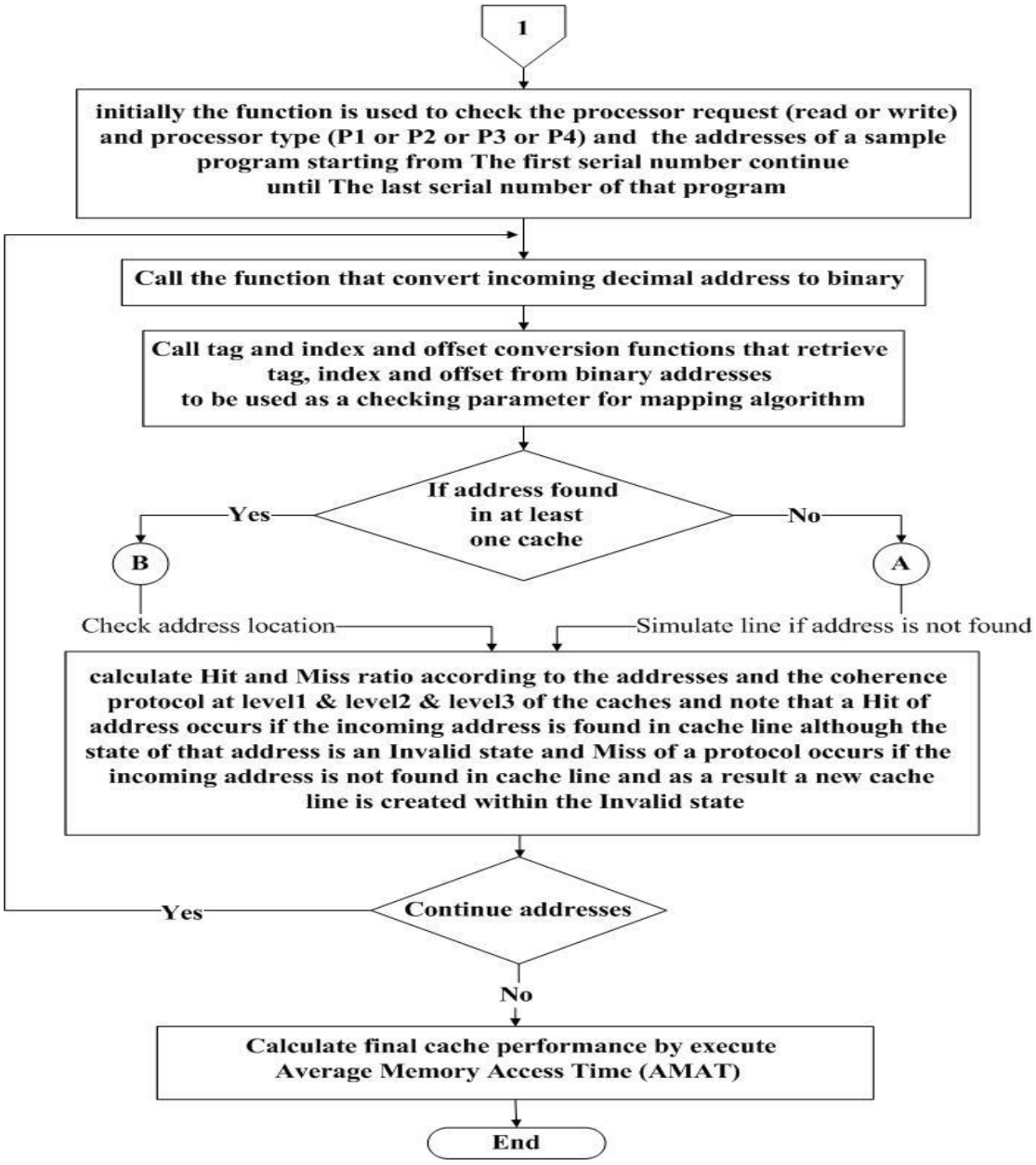

Figure 4: cont.

Direct Mapped Method

It is the simplest technique which maps each block of main memory into only one possible cache line [16]. The mapping is expressed using equation 1

$$i = j \ modulo \ L \quad where$$

$$i = cache\ line\ number \quad \text{---------- (1)}$$
$$j = main\ memory\ block\ number$$
$$L = number\ of\ lines\ in\ the\ cache$$

Cache block = address modulo number of words in cache line ---------- (2)

Total number of memory blocks = number of main memory addresses / number of words in cache line ---------- (3)

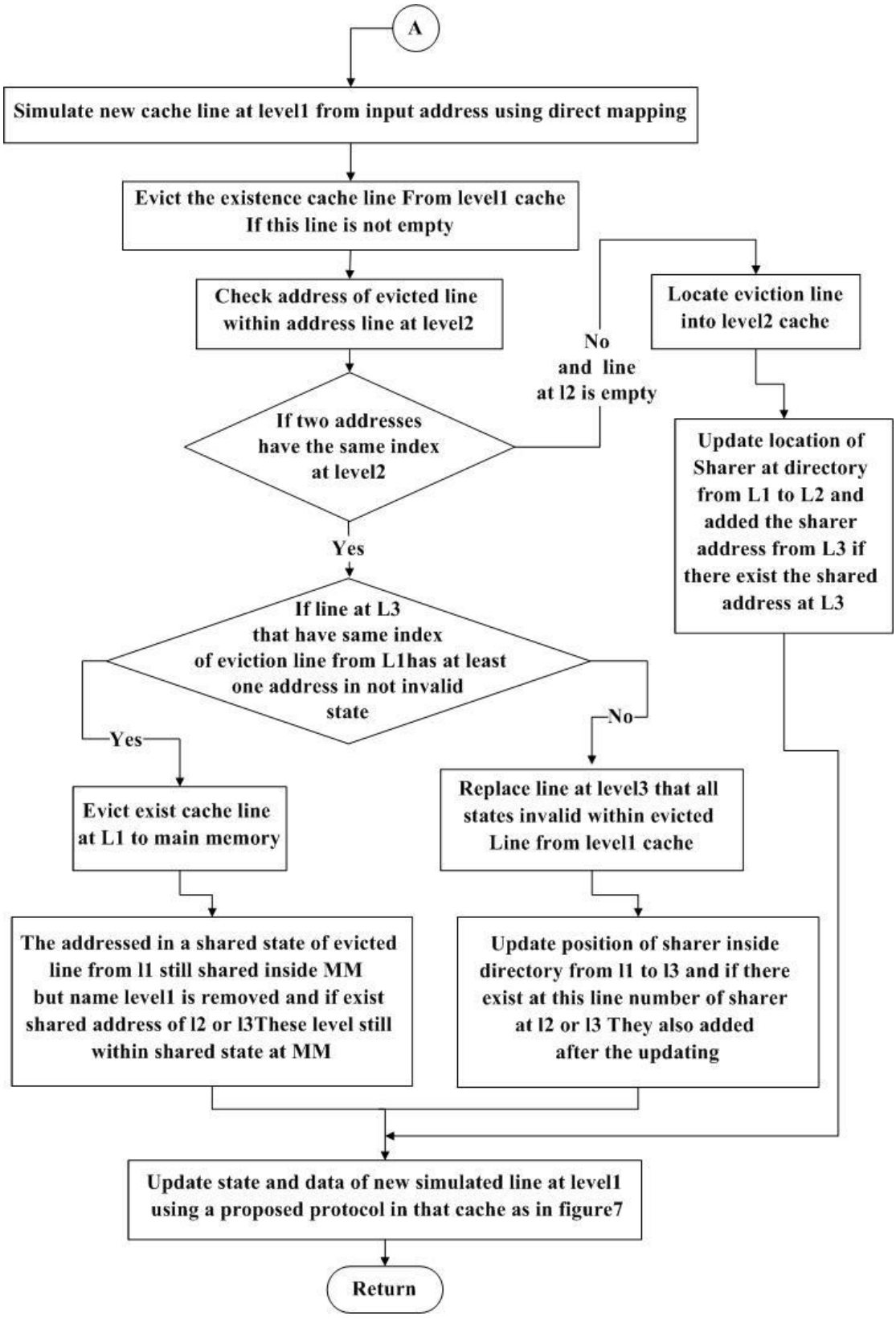

Figure 5: Simulation process using Direct Mapped Method In the case of not existence the address

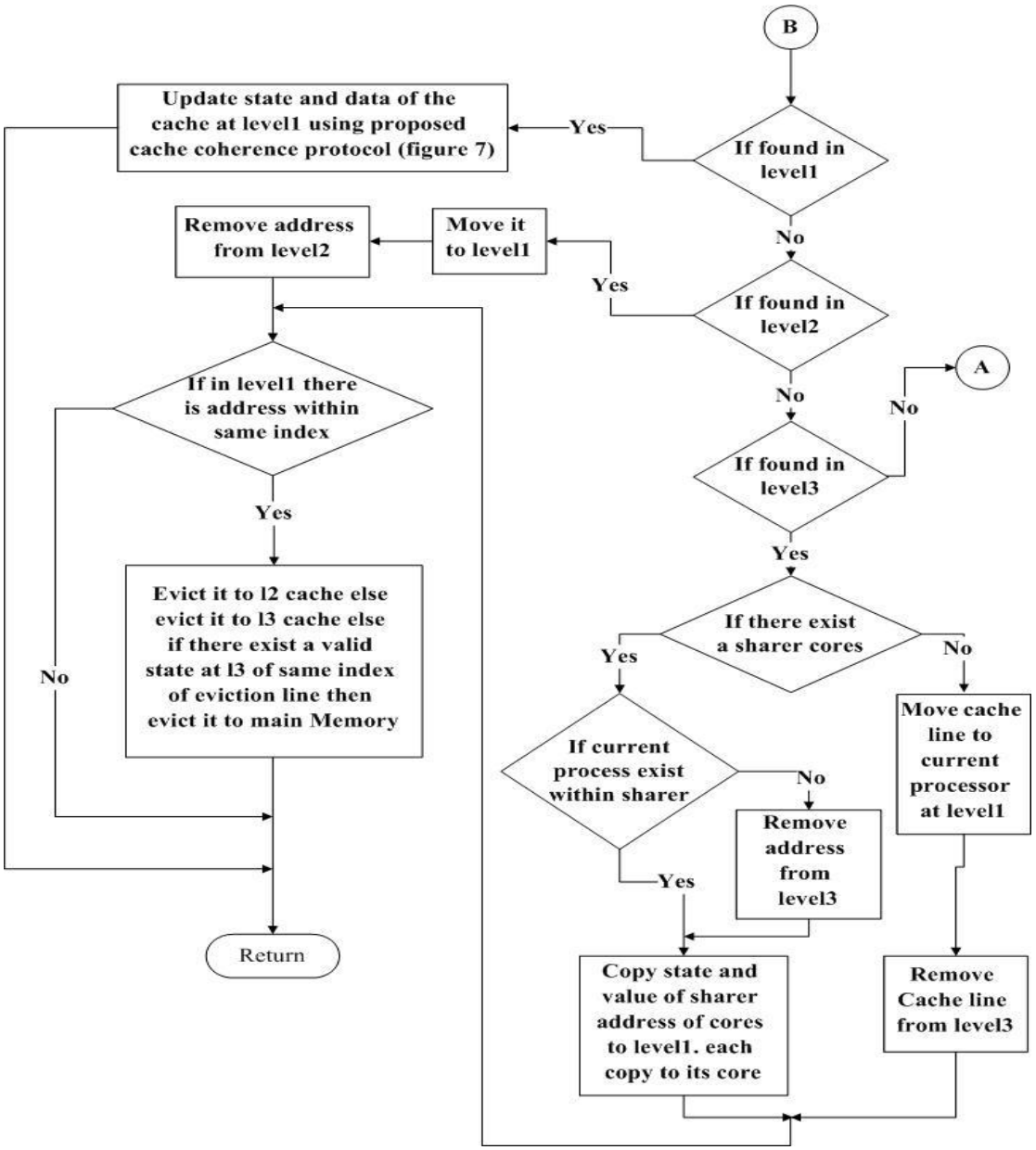

Figure 6: Simulation process using Direct Mapped Method In the case of existence the address

Transition State Diagram of proposal protocol

A cache line in each cache of proposal protocol can be in one of the following states as in Figure 7:-

Modified: the data owned by one processor, but it is dirty; must respond to any read/write request

Exclusive: the data owned by one processor and it is clean; no need to inform others about further changes

Shared: cached in more than one processors and memory is up-to-date

Invalid: The block has been invalidated (possibly on the request of someone else)

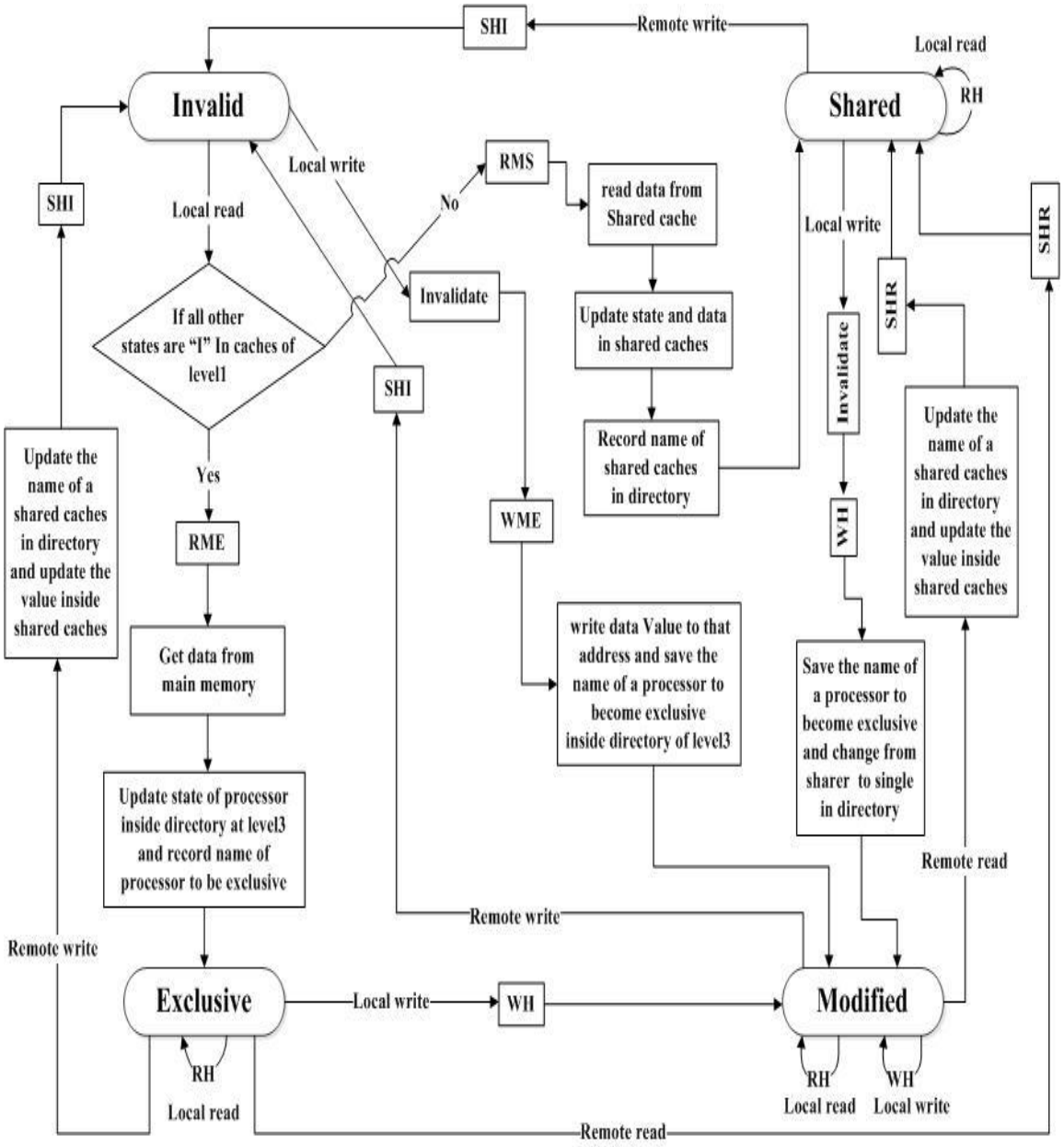

Figure 7: The state diagram of proposal protocol within shared directory at L3

The abbreviate symbols of these buses are as follow:

Bus transaction:
Invalidate = Broadcast Invalidate
Events:
RH = Read Hit
RMS = Read Miss, Shared
RME = Read Miss, Exclusive
WH = Write Hit
WM = Write Miss
WME = Write Miss, Exclusive
SHR = Snoop Hit on Read
SHI = Snoop Hit on Invalidate

Measuring Cache Performance

Time CPU lapses in the implementation of the program as well as in waiting inside the memory, so CPU time is calculated as in the following equations [6, 7]:

$$CPU\ time = (CPU\ execution\ clock\ cycles + Memory\ stall\ clock\ cycles) * clock\ cycle\ time \quad --(4)$$

$$CPU\ time = IC * \left(CPI_{Execution} + \frac{Mem\ Access}{Inst.} * Miss\ rate * Miss\ penality\right) * cycle\ time \quad --(5)$$

$$CPU\ time = IC * \left(\frac{ALUOPS}{Inst.} * CPI_{ALUOPS} + \frac{Mem\ Access}{Instruction} * AMAT\right) * cycle\ time \quad --(6)$$

$$\text{AMAT} = L1\ Hit\ time * L1\ Hit\ rate + L1\ Miss\ penality * L1\ miss\ rate \quad --(7)$$

$$L1\ Miss\ penality = Access\ time\ of\ L2 = L2\ Hit\ \ time * L2\ Hit\ rate + L2\ Miss\ penality * L2\ Miss\ rate \quad --(8)$$

$$L2\ Miss\ penality = Access\ time\ of\ L3 = L3\ Hit\ \ time * L3\ Hit\ rate + L3\ Miss\ penality * L3\ Miss\ rate \quad --(9)$$

$$L3\ Miss\ penality = Access\ time\ of\ Main\ Memory \quad --(10)$$

$$\text{AMAT} = L1\ hit\ time * L1\ Hit\ rate + (L2\ Hit\ time * L2\ Hit\ rate + (L3\ Hit\ time * L3\ Hit\ rate + Access\ time\ of\ Main\ Memory\ * L3\ Miss\ rate) * L2\ Miss\ rate) * L1\ Miss\ rate \quad --(11)$$

Where

IC = **I**nstruction **C**ounter
CPI = **C**lock cycle **P**er **I**nstruction
AMAT = **A**verage **M**emory **A**ccess **T**ime

- ***Hit*** -- the referenced information is in the cache.
- ***Miss*** -- the referenced information is not in cache, and must be read from MM
- ***Hit time*** – is how long it takes data to be sent from the cache to the processor. This is usually fast, on the order of 1-5 clock cycles at Level1, of 10-20 clock cycles at Level2, of 30-40 clock cycles at Level3, of 50-100 clock cycles at main memory.
- ***Miss penalty*** – is the time to copy data from main memory to the cache. This often requires dozens of clock cycles (at least).
- ***Hit ratio*** -- percentage of time the data is found in the higher cache.
- ***Miss ratio*** – is the percentage of misses and equal (100 - hit ratio).

The Experiment Result Using DEV C++ Language

Binary Representation

Binary representation is one of a necessary preprocessing steps used to convert decimal addresses to binary address in order to obtain tag and index and offset of each binary address so as to facilitate the work of a mapping algorithm. In a proposed protocol, Main Memory has 8-bit to represent the address. So, the addresses of memory have 256 addresses as in table 1.

Table 1: Binary Representation of a Memory Addresses as Tag and Index and Offset

Memory address	Binary representation								Decimal number		
	tag			index		offset			tag	index	offset
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0	0	1
2	0	0	0	0	0	0	1	0	0	0	2
3	0	0	0	0	0	0	1	1	0	0	3
.	.	.	.	.	.	.	. .	.	.	.	.
.	.	.	. .	.	.	.	.	.	.	.	.
255	1	1	1	1	1	1	1	1	7	3	7

Cache Simulation using Direct Mapped Function

The mapping of a memory addresses into L1 caches using direct mapped function are as follow:-

a- Number of cache block at level1 obtains from equation 3
Total number of memory blocks = 256 / 8 = 32 block

b- initially cache block are determines from equation 2 as follows
i.e. the cache block 20 contain the cache line from address 160 to address 167 and obtained by dividing these address to 8
i.e. the cache block 1 contain the cache line from address 8 to address 15 and obtained by dividing these address to 8
i.e. the cache block 18 contain the cache line from address 144 to address 151 and obtained by dividing these address to 8
i.e. the cache block 27 contain the cache line from address 216 to address 223 and obtained by dividing these address to 8

c- Then the direct mapped function as in equation 1 is applied to these blocks to obtain index that will be used in simulation of caches at level 1as follow:

Cache block 0, 4, 8, 12, 16, 20, 24, 28 are mapped to index 0
Cache block 1, 5, 9, 13, 17, 21, 25, 29 are mapped to index 1
Cache block 2, 6, 10, 14, 18, 22, 26, 30 are mapped to index 2
Cache block 3, 7, 11, 15, 19, 23, 27, 31 are mapped to index 3

These steps are repeated to simulate caches at level2 and level3 but different is in number of cache line and tag that they are specified initially

Proposal protocol Results

Before applying the proposal protocol, a binary function is used to convert addresses of input sample program to binary address and then other functions are used to obtain tag, index and offset from a binary address. Table2 list binary addresses that are used in a sample program exist in table3.

Table 2: Binary Representation of Input Addresses Using Proposal Protocol

address representation			cache level1			cache level2			cache level3		
Seq	binary no	address	tag	index	offset	Tag	index	offset	tag	index	offset
1	01000000	64	2	0	0	1	0	0	0	8	0
2	01000100	68	2	0	4	1	0	4	0	8	4
3	01000110	70	2	0	6	1	0	6	0	8	6

The results of applying a proposed protocol on a sample program are listed in table7. Initially all states of input sample program are invalid and also all the values equal to zero.

Table 3: The Results of a Proposed Protocol on a Sample Program

Seq	core name	core request	data	cache line			Event	Bus operation	Sharer Cores	Description
				address	state	value				
1	P1	writes	20	64	M	20	WME	invalidate		I → M
2	P3	writes	7	68	M	7	WME	invalidate		I → M
3	P2	reads		64	S	20	RMS	update directory	P1&P2 at L1	I → S
4	P2	writes	77	70	M	77	WME	invalidate		I → M
5	P4	reads		68	S	7	RMS	Update directory	P3&p4 at L1	I → S
6	P1	reads		70	S	77	RMS	Update directory	P1&P2at L1	I → S
7	P2	writes	45	68	S	45	WME	invalidate		I → M
8	P2	reads		70	S	77	RH		P1&P2at L1	S → S
9	P4	writes	99	64	M	99	WME	Invalidate		I → M
10	P1	writes	35	64	M	35	WME	Invalidate & Update directory		I → M
11	P3	reads		64	S	35	RMS	Update directory	P1&P3at L1	I → S
12	P1	writes	80	70	M	80	WH	invalidate		S → M
13	P3	reads		68	S	45	RMS	Update directory	P2&P3at L1	I → S
14	P2	reads		70	S	80	RMS	Update directory	P1&P2 at L1	I → S
15	P1	writes	54	70	M	54	WH	invalidate		S → M
16	P2	reads		68	S	45	RH		P2&P3 at L1	S → S
17	P4	reads		70	S	54	RMS	Update directory	P1&P4 at L1	I → S
18	P1	writes	33	70	M	33	WME	invalidate		I → M
19	P3	reads		70	S	33	RMS	Update directory	P1&P3 at L1	I → S
20	P3	reads		64	S	35	RH		P1&P3 at L1	S → S

Cache Performance Result

Cache performance can be measured by counting a program execution cycles that include cache Hit time and a memory stall cycles which result from cache misses. Suppose that after depending on the clock speed of the central processor, it takes: 7 ns to access data in L1 cache, 17 ns to access data in L2 cache, 30 ns to access data in L3 cache, 80 ns to access data in Main Memory.

Calculate hit and miss ratio according to both addresses and also to proposed protocol:

- In using proposed protocol1

The proposed protocol in figure 7 are applied on a sample program, then hit and miss ratio results are appears at level1 caches only because the input addresses use only 3 address and all these addresses appear at the same cache line. Hit and miss ratio calculate from table 3 as follow:

Hit ratio at L1 = (no. of hit in level1/ total no. of address)*100 = (5/20)*100= 25%

Miss ratio in L1 =100-Hit ratio =100-25= 75%

Hit and Miss ratio in level2 and Level3 are not exist

2- in using addresses

First address equal to miss and all other addresses are equal to hit because the same line contains all addresses and when this line is fetch then the first address take from main memory and the rest addresses appears. So, Hit and miss ratio is calculated as follow:

Hit ratio at L1 = (19/20) * 100 = 95 %

Miss ratio at L1 = 100 – 95 = 5 %

Hit ratio = (hit ratio in protocol + hit ratio in address) / 2 = (25 + 95) / 2 = 60 %

Miss ratio = (miss ratio in protocol + miss ratio in address) / 2 = (75 + 5) / 2 = 40 %

Finally Average Memory Access Time (AMAT) is applied as in equation11

The Comparison between MESI and proposed Protocol

- In MESI *cache coherence Protocols* the directory that keep track of shared data is located in main memory but in a proposed protocol the directory is located in a shared cache level3. As a result the efficiency is increased by reducing a gap between a fast CPU and a slow main memory.
- The write through and write back has been translated from main memory into level3 shared cache, so the disadvantages of write through in uses more memory bandwidth is reduced, and the disadvantages of write back of making the main memory inconsistent with cache also reduced. The different between MESI and proposed protocol in using sample program that are shown in table 3 are as follow:

Steps 3, 5, 6, 11, 13, 14, 17, 19 are write back addresses of a previous modified state to main memory as a result of a remote read. And step 10 is write back modified address line to main memory as a result of remote write. But in using proposed protocol these steps return to update the directory at level3 instead of access to main memory.

Conclusion and Future Works

A new idea is proposed in this research to achieve cache coherency. The reason behind the development of coherency protocol is that this protocol effectively affects the efficiency of the processor in multi-core computer systems.

In future work the number of caches at level1 and level2 are tried to be increased and also modifying in one of the states and also increase associativity in using mapping algorithm. All these idea are proposed in order to reduce access to main memory**.**

REFERENCES

[1] J. Sorin Daniel & D. Hill Mark & A. Wood David, "A Primer on Memory Consistency and Cache Coherence ", A Publication in the Morgan & Claypool Publishers series, 2011.

[2] El-Rewini Hesham & Abd-El-Barr Mostafa, "ADVANCED COMPUTER ARCHITECTURE AND PARALLEL PROCESSING ", Published by John Wiley & Sons, Inc., Hoboken, New Jersey. Published simultaneously in Canada, 2005.

[3] Rauber Thomas & R¨unger Gudula, "Parallel Programming For Multicore and Cluster Systems ", Published by Springer-Verlag Berlin and Heidelberg GmbH & Co. KG, Berlin ,2013.

[4] Saparon Azilah, and Bt Razlan Fatin Najihah, " Cache Coherence Protocols in Multi-Processor", International conference on Computer Science and Information Systems (ICSIS'2014) Oct 17-18, 2014 Dubai (UAE) .

[5] Zaghloul Soha S., et.al., " Index-Based Cache Coherence Protocol ", Journal of communication and Compter vol. 11pp. 479-483 2014.

[6] A. Patterson David & L. Hennessy John, "Computer Organization and Design the Hardware / software interface ", Elsevier Inc., 2005.

[7] A. Patterson David & L. Hennessy John, "Computer Architecture A quantitative approach ", *Morgan Kaufmann* is an imprint of Elsevier, 2012.

[8] Stalling William, "Computer organization and architecture designing for performance ", Printed in the United States of America by Pearson Education, Inc., Upper Saddle River, New Jersey, 07458, 2010.
[9] Hwang Kai & A. Briggs Faye, "Computer architecture and parallel processing", copyright by McGraw-Hill, Inc. in New York St. Louis San Francisco, London, Paris, 1985.
[10] Moyer Bryon, "Real World Multicore Embedded Systems", Elsevier Inc., United States of America, 2013.
[11] Tiwari Anoop, " Performance Comparison of Cache Coherence Protocol on Multi-Core Architecture", Department of Computer Science and Engineering National Institute of Technology Rourkela Rourkela, Odisha, 769008, India, 2014.
[12] Al-Hothali Samaher, Soomro Safeeullah , et.al.," Snoopy and Directory Based Cache Coherence Protocols: A Critical Analysis", *Journal of Information & Communication Technology* Vol. 4, No. 1, (Spring 2010) 01-10.
[13] Culler David & Singh Jaswinder Pal & Gupta Anoop, "Parallel Computer Architecture A Hardware/Software Approach, scalability, programmability", Morgan Kaufmann Publishers., 1997.
[14] Pugsley Seth H., et.al., "SWEL: Hardware Cache Coherence Protocols to Map Shared Data onto Shared Caches",19th International Conference on Parallel Architectures and Compilation Techniques (PACT-19), Vienna, September 2010.
[15] Martin Milo M. K., "Formal Verification and its Impact on the Snooping versus Directory Protocol Debate", International Conference on Computer Design (ICCD 05) IEEE Computer Society Publishing Services, 2005.
[16] Ros Alberto and Jimborean Alexandra, "A Dual-Consistency Cache Coherence Protocol", IEEE 29th International Parallel and Distributed Processing Symposium IPDPS, **pp. 1119-1128, USA**, 2015.