Э.А. АЛЕКСАНЯН

ПРОТОТИП МОДЕЛИ ИЗУЧЕНИЯ И ИССЛЕДОВАНИЯ ЭЛЕКТРОННЫХ СИСТЕМ В СРЕДЕ ВИРТУАЛЬНОЙ РЕАЛЬНОСТИ

Рассматривается прототип модели виртуальной реальности, позволяющий пользователю проводить в виртуальной реальности такие научно-лабораторные эксперименты, которые по каким-то причинам невозможно проводить в реальной среде или которые опасны либо очень дороги. Представленная модель предназначена для создания и использования как образовательного инструмента для студентов, так и инструмента проектирования для промышленности или в качестве виртуальной электрической среды для широкой публики.

Ключевые слова: электрические схемы, виртуальная реальность, лабораторное исследование, электрическая среда.

E.A. ALEKSANYAN

STUDY AND RESEARCH OF THE MODEL PROTOTYPE OF ELECTRONIC SYSTEMS IN VIRTUAL REALITY ENVIRONMENT

A prototype of virtual reality model is considered, allowing the user to perform such research-laboratory experiments using Virtual Reality, which for some reasons, are impossible, hazardous and very expensive to perform in the real environment. The presented model is intended to create and use both as an educational tool for students and as a design tool for industry, as well as a virtual electrical environment for the general public.

Keywords: electrical circuits, virtual reality, laboratory experiment, electrical environment.

ՀՏԴ 681.3

Ա. Կ. ՍԱՂԱԹԵԼՅԱՆ, Հ. Վ. ԾԱՌՈՒԿՅԱՆ, Գ. Ա. ՇԱՀՆԱԶԱՐՅԱՆ

ԲԱԶՄԱՊՐՈՑԵՍՈՐԱՅԻՆ ՀԱՄԱԿԱՐԳԵՐՈՒՄ ՔԵՇ-ՀԻՇՈՂՈՒԹՅԱՆ ԿՈՀԵՐԵՆՏՈՒԹՅՈՒՆԸ, ՀՍԿՈՂ ԱՐՁԱՆԱԳՐՈՒԹՅՈՒՆՆԵՐԻ ԱՊԱՐԱՏԱՅԻՆ ԵՎ ԾՐԱԳՐԱՅԻՆ ԻՐԱԳՈՐԾՈՒՄՆԵՐԻ ՀԱՄԵՄԱՏՈՒՄԸ

Հետազոտված են ժամանակակից բազմապրոցեսորային համակարգերում առաջացող քեշ-հիշողությունների կոհերենտության խնդիրները և այդ դեպքում կիրառվող հսկող արձանագրությունների արդի տեսակները: Դիտարկված են նաև մշակված հսկող արձանագրությունների բլոկի ապարատային և ծրագրային իրագործումների համեմատումները արագագործությունների տեսանկյունից:

Առանցքային բառեր. բազմապրոցեսորային համակարգեր, քեշ-հիշողության կոհերենտություն, MESI, MOESI, MESIF հսկող արձանագրություններ, ապարատային և ծրագրային իրագործումների համեմատում:

Ժամանակակից հաշվողական համակարգերում արագագործության բարձրացման նպատակով կիրառվում են MIMD (Multiple Instruction stream, Multiple

Data stream) բազմապրոցեսորային համակարգեր: Այդպիսի համակարգերում կարևորվում է հիշող սարքերում (մասնավորապես օպերատիվ և քեշ հիշողությունների) տվյալներին արագ հասանելիության ապահովումը: Այս դեպքում հիշողությունների միջև տվյալների պատշաճ փոխանակման կազմակերպումն առաջացնում է քեշերի կոհերենտության խնդիր, որի լուծումն իրականացվում է հսկող արձանագրությունների միջոցով[1,2,4]:

Հետևապես՝ բազմապրոցեսորային համակարգերում քեշ-հիշողության կոհերենտության հետազոտումը, հսկող արձանագրությունների բլոկի մշակումը և իրականացումը կարևոր է և արդիական:

Բազմապրոցեսորային ՀՀ-ում հիշողության կազմակերպումները. Պրոցեսորի և հիշողության արագագործության տարբերությունները խնդիրներ են առաջացնում նույնիսկ միապրոցեսորանի համակարգերում, իսկ բազմապրոցեսորանի համակարգերում առաջանում է ևս մեկը՝ մի քանի պրոցեսորների՝ հիշողությանը միաժամանակ դիմումի կազմակերպումը: Կախված բազմապրոցեսորային համակարգերում հիշողության կազմակերպման եղանակից՝ տարբերում են առանձնացված (shared memory) և բաշխված (distributed memory) հիշողությամբ հաշվողական համակարգեր:

Առանձնացված հիշողությամբ համակարգերում (այն հաճախ անվանում են նաև համատեղ կամ ընդհանուր հիշողությամբ) **ՀՀ**-երի հիշողությունը դիտվում է որպես ընդհանուր ռեսուրս, և պրոցեսորներից յուրաքանչյուրն ունի լիարժեք հասանելիություն բոլոր հասցեային տարածքներին: Առանձնացված հիշողությամբ համակարգերն անվանում են **խիստ կապված համակարգեր (closely coupled systems):** Նման ՀՀ-երի կառուցումը տեղ է գրավում ինչպես **SIMD (**Single Instruction stream, Multiple Data stream), այնպես էլ **MIMD** դասերում [1,3]:

Բաշխված հիշողությամբ համակարգերում յուրաքանչյուր պրոցեսորի համար առանձնացվում (տրվում) է սեփական հիշողություն: Պրոցեսորները միավորվում են ցանցում և անհրաժեշտության դեպքում կարող են իրենց հիշողությունից տվյալներ փոխանակել մեկը մյուսին, այսպես կոչված, «հաղորդագրություն» փոխանցելով: Այսպիսի հաշվողական համակարգն անվանում են **թույլ կապված (loosely coupled system)**: Թույլ կապված համակարգերը նույնպես հանդիպում են ինչպես **SIMD,** այնպես էլ **MIMD** դասերում:

Ինչպես առանձնացված, այնպես էլ բաշխված հիշողությունների շրջանակներում իրականացվում են հիշողության համակարգերի մի քանի ճարտարապետություններ [1,3,6].

• ***Ֆիզիկապես առանձնացված (համատեղ) հիշողություն,*** երբ բոլոր պրոցեսորներն ունեն միակ հասցեային տարածքի հասանելիության հավասար

հնարավորություններ. յուրաքանչյուր պրոցեսորի մուտքի հասանելիությունը հիշողությանը իրականացվում է միատեսակ և ունի միևնույն տևողությունը: Այդպիսի հաշվողական համակարգերն անվանում են **հիշողության միատեսակ մուտքի հնարավորությամբ համակարգեր** կամ **UMA** (Uniform Memory Access):

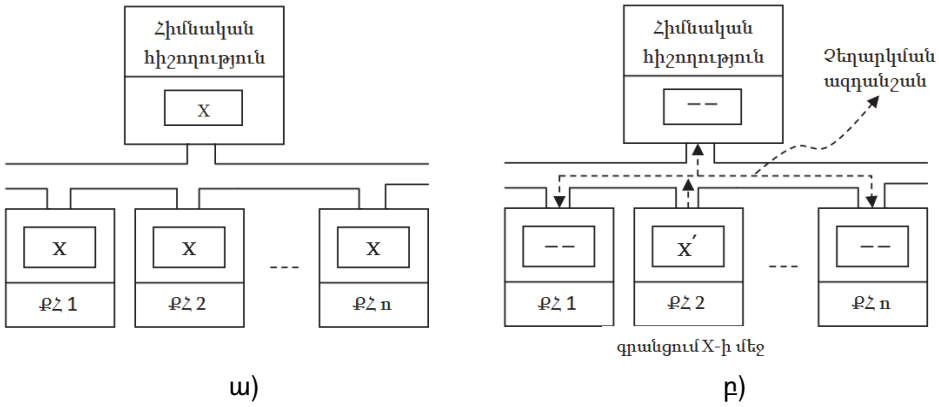
• ***Ֆիզիկապես բաշխված առանձնացված (համատեղ) հիշողություն,*** երբ հիշողությանը հասանելիությունն արտահայտվում է այն փաստով, որ պրոցեսորը մուտք է գործում իր տեղային հիշողություն ուղղակիորեն (սա շատ ավելի արագ է, քան մուտքը հեռավոր հիշողություն՝ քոմփյութերի կամ ցանցի միջոցով): Այդպիսի հաշվողական համակարգերն անվանում են **հիշողության ոչ միատեսակ մուտքի հնարավորությամբ համակարգեր** կամ NUMA (Non-Uniform Memory Access): Այս դասում անհրաժեշտ է առանձնացնել նաև COMA (Cache Only Memory Architecture) ճարտարապետությունը, երբ բոլոր պրոցեսորների քեշերի ամբողջությունը դիտվում է որպես համակարգի գլոբալ հիշողություն, և բացակայում է սեփական գլոբալ հիշողությունը [4,5]

Քանի որ COMA ճարտարապետության մեջ տվյալները փոխանցվում են պրոցեսորի տեղային քեշ, որը վերջին անգամ է դրանց հարցում արել, ուստի COMA հիշողությամբ ՀՀ-ի արտադրողականությունը զգալիորեն գերազանցում է NUMA հիշողությամբ ՀՀ-ին: Ֆիզիկապես բաշխված հիշողությամբ համակարգերում հիմնական հիշողության նույն բլոկի պատճենների պահպանման համաձայնության խնդիր է առաջանում: Պատճենների ոչ միարժեքությունն ի հայտ է գալիս, երբ պրոցեսորներից մեկն իր քեշում փոխում է ինֆորմացիան: Այս պարագայում մյուս պրոցեսորների քեշերի նմանատիպ պատճենները պետք է համապատասխան կերպով ուղղվեն կամ անվավեր ճանաչվեն: Այս խնդիրը հայտնի է որպես **քեշ-հիշողության կոհերենտության խնդիր**, իսկ քեշ-հիշողության կոհերենտության խնդիրները լուծող ապարատային միջոցները անվանում են քեշ-հիշողության կոհերենտության արձանագրություններ [6, 8]:

Ինչպես հայտնի է, բազմապրոցեսորանի համակարգերում հիմնական և քեշ-հիշողություններում տվյալների պատճենների նույնականացման համար ընդունված է երկու՝ *չեղարկումով գրանցում* կամ *թարմացումով գրանցում* ռազմավարություններից մեկը: Չեղարկմամբ գրանցման տարբերակում, եթե պրոցեսորներից մեկն իր քեշի բլոկներից մեկում փոփոխություն է կատարում, այդ բլոկի բոլոր պատճեներն այլ տեղային քեշերում չեղարկվում են, այսինքն՝ անվավեր են ճանաչվում: Դրա համար փոփոխված բլոկի վավերականության բիթը մյուս բոլոր քեշերում տեղակայվում է 0: Չեղարկմամբ գրանցման գաղափարը ներկայացված է նկ. 1-ում, որտեղ ցույց է տրված հիշողության համակարգի նախնական վիճակը, երբ **x** փոփոխականի պատճենն առկա է բոլոր քեշերում (նկ. 1ա),

ինչպես նաև պրոցեսոր 2-ում **x**-ի նոր արժեքի գրանցումից հետո նրա վիճակը (նկ. 1բ):

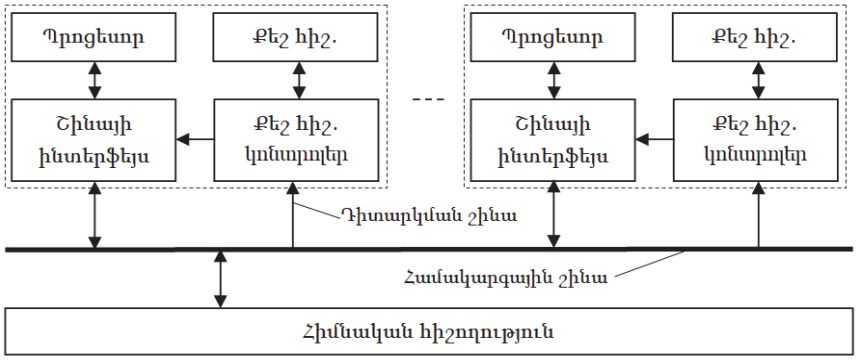
Դիտարկման (հսկող) արձանագրություններ: Դիտարկման արձանագրության մեջ բազմապրոցեսորային համակարգերում բոլոր քեշերի կոհերենտության ապահովման պատասխանատվությունն ընկած է քեշերի կոնտրոլերների վրա:



ա) բ)

Նկ. 1. Չեղարկմամբ գրանցում. ա) նախնական վիճակ, բ) պրոցեսոր 2-ի քեշում x-ի արժեքը՝ փոփոխելուց հետո

Համակարգերում, որտեղ իրականացվում են դիտարկման արձանագրությունները, յուրաքանչյուր տեղային քեշ-կոնտրոլեր պարունակում է *շինային հետևող բլոկ* (նկ. 2), որը հետևում է ընդհանուր շինայի վրա բոլոր տրանզակցիաներին, մասնավորապես՝ ղեկավարում է բոլոր գրանցման օպերացիաները:



Նկ. 2. Դիտարկման արձանագրությամբ քեշերի կոհերենտությունն ապահովող համակարգի կառուցվածքը

Ժամանակակից բազմապրոցեսորային համակարգերում կիրառում են մի քանի տեսակի դիտարկման արձանագրություններ, որոնցից ուշադրության են

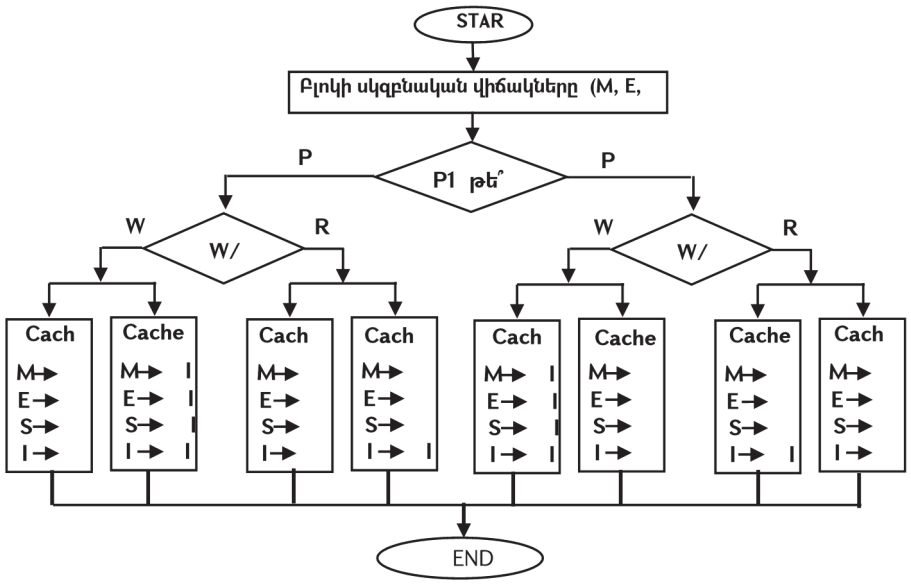
արժանի MESI, MOESI և MESIF տարբերակները: Քեշերում պահվող բլոկներին տրամադրվում են բիթեր, համաձայն որոնց որոշվում է, թե ինչ կարգավիճակում է տվյալ քեշում այդ բլոկը [4,7]:

MESI արձանագրություն: MESI արձանագրության ցանկացած զույգ քեշերի համար տվյալ քեշի տողի թույլատրելի վիճակները կարելի է ներկայացնել հետևյալ աղյուսակային տեսքով:

	M	E	S	I
M	✗	✗	✗	✓
E	✗	✗	✗	✓
S	✗	✗	✓	✓
I	✓	✓	✓	✓

Modified – տվյալ բլոկը փոփոխված է և առկա է միայն այդ քեշում:
Exclusive – տվյալ բլոկի ճիշտ տարբերակն առկա է միայն մեկ քեշում (որտեղ կատարվել է վերջին փոփոխությունը) և ՕՀ-ում:
Shared – տվյալ բլոկի ճիշտ տարբերակը կա ՕՀ-ում, ինչպես նաև երկու կամ ավելի քեշերում:
Invalid – այս բլոկի տվյալներն անվավեր են:

MESI արձանագրությունը կարելի է ներկայացնել նաև անցումների դիագրամի միջոցով [4,9,11,12]: Այդ անցումների գրաֆի հիման վրա մշակվել է ալգորիթմի բլոկ-սխեմա, որը ներկայացված է նկ. 3-ում: Պարզության և բլոկերի հնարավորինս ավելի քիչ վիճակների փոփոխությունների դիտարկման համար MESI արձանագրության կիրառմամբ քեշերի կոհերենտության խնդրի լուծումը մշակվել է երկու պրոցեսորային համակարգի համար:



Նկ. 3. MESI արձանագրության ալգորիթմի բլոկ - սխեման

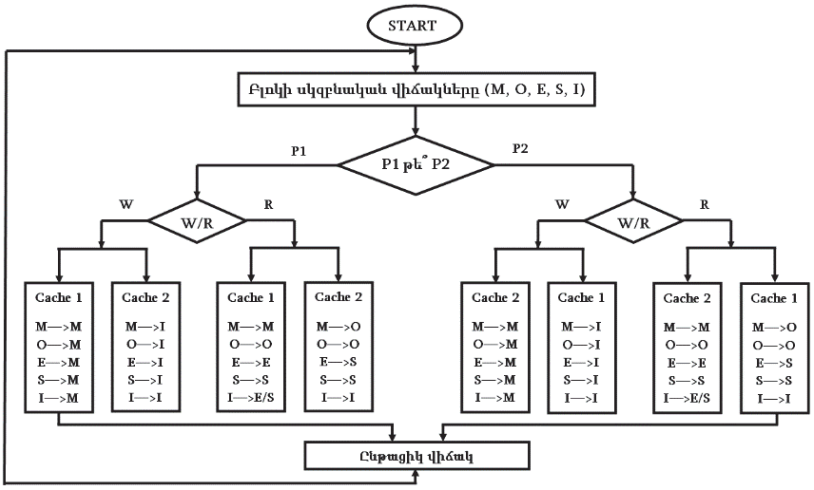
MOESI արձանագրություն: Ժամանակակից բազմապրոցեսորանի համակարգերում քեշ-հիշողությունների կոհերենտության խնդիրների լուծման համար օգտագործվում է նաև **MOESI** արձանագրությունը: Ի հավելումն MESI արձանագրության չորս ընդհանուր վիճակների՝ գոյություն ունի հինգերորդը՝ O (Owned)-*«Սեփական»*, որը ներկայացնում է ինչպես *փոփոխված*, այնպես էլ *shared* տվյալներ: Սա թույլ է տալիս խուսափել փոփոխված տվյալները հիմնական հիշողության մեջ հետ գրանցելու անհրաժեշտությունից՝ նախքան դրանց տարածումը: Իսկ երբ, վերջիվերջո, տվյալները պետք է հետ գրվեն հիշողությունում, հետ գրանցումը կարող է հետաձգվել: Որպեսզի դա հնարավոր լինի, պետք է հնարավոր լինի տվյալների քեշից քեշ ուղիղ փոխանցումը:

MOESI արձանագրության ցանկացած զույգ քեշերի համար տվյալ քեշի տողի թույլատրելի վիճակները կարելի է ներկայացնել հետևյալ աղյուսակային տեսքով:

	M	O	E	S	I
M	✗	✗	✗	✗	✓
O	✗	✗	✗	✓	✓
E	✗	✗	✗	✗	✓
S	✗	✓	✗	✓	✓
I	✓	✓	✓	✓	✓

Այս արձանագրությունը թույլ է տալիս խուսափել քեշի փոփոխված տողի՝ հիմնական հիշողությունում հետադարձ գրանցումից, երբ մեկ այլ պրոցեսոր փորձում է ընթերցել այն: Փոխարենը, Owned վիճակը պրոցեսորին թույլ է տալիս փոփոխված տվյալներն ուղղակիորեն տրամադրել մյուս պրոցեսորին: Սա հարմար է, երբ երկու պրոցեսորների միջև հաղորդակցման հապաղումը և թողունակությունը զգալիորեն ավելի լավն են, քան հիմնական հիշողության հետ կապվածները:

MOESI արձանագրությունը նույնպես կարելի ներկայացնել անցումների դիագրամի միջոցով [4,10,11]: Այդ անցումների գրաֆի հիման վրա մշակվել է ալգորիթմի բլոկ-սխեմա, որը ներկայացված է նկ. 4-ում:



Նկ. 4. MOESI արձանագրության անցումների դիագրամը

MESIF արձանագրություն: MESI արձանագրությունը լրացնում է Intel-ի շինայի հիման վրա հին ճարտարապետությունը և նրբորեն ապահովում կոհերենտությունը: Մասնավորապես, երբ պրոցեսորը հարցում է անում քեշի տողը, որը պահվում է բազմաթիվ վայրերում, յուրաքանչյուր պրոցեսոր կարող է արձագանքել տվյալներով: Սակայն հայցող պրոցեսորին անհրաժեշտ է տվյալների միայն մեկ պատճեն, այնպես որ համակարգը որոշակի թողունակություն է վատնում:

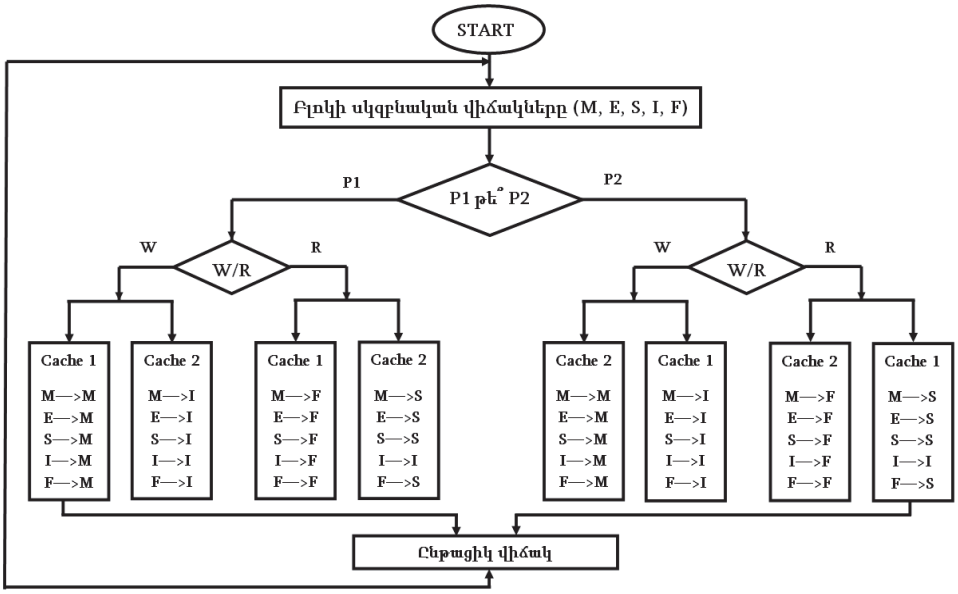
Intel-ը այս խնդրի լուծման համար MESI ստանդարտ արձանագրությանը ներառել է լրացուցիչ *«Forwarding» (F)* վիճակ՝ փոխելով *Shared (S)* վիճակի դերը: MESIF արձանագրությունում քեշի տողի միայն մեկ օրինակ կարող է լինել F վիճակում, և այդ օրինակը միակն է, որը կարող է պատճենվել:

MESIF արձանագրության ցանկացած զույգ քեշերի համար տվյալ քեշի տողի թույլատրելի վիճակները կարելի է ներկայացնել աղյուսակի տեսքով:

	M	E	S	I	F
M	✗	✗	✗	✓	✗
E	✗	✗	✗	✓	✗
S	✗	✗	✓	✓	✓
I	✓	✓	✓	✓	✓
F	✗	✗	✓	✓	✗

Մյուս քեշերը կարող են ունենալ տվյալները, բայց նրանք կգտնվեն *shared* վիճակում և չեն կարող պատճենվել: Այլ կերպ ասած, F վիճակում գտնվող քեշի տողն օգտագործվում է ցանկացած ընթերցման հարցմանը պատասխանելու համար, մինչդեռ քեշի S վիճակում գտնվող տողերն այդ ժամանակ լռում են: Սա F վիճակում գտնվող տողը դարձնում է առաջնային իրեն հավասարների շրջանում, երբ պատասխանում է դիտարկման հարցումներին:

Երբ առկա են տվյալների բազմակի պատճեններ հարցումներին պատասխանելու համար, քեշի մեկ տողի նշանակումը թրաֆիկն էապես կրճատում է: Անցումային գրաֆի հիման վրա մշակված է MESIF արձանագրության բլոկ-սխեման ներկայացված է նկ. 5-ում:



Նկ. 5. MESIF արձանագրության բլոկ-սխեման

Ներկայացված հսկող արձանագրությունների ալգորիթմների բլոկ-սխեմաները մշակված են երկպրոցեսորանի համակարգերի համար, սակայն կատարվել են նաև հաշվարկներ քեշ-հիշողության բլոկների վիճակների քանակի վերաբերյալ n-պրոցեսորանի համակարգի համար, որոնց տվյալները ներկայացված են աղյուսակում: Օրինակ՝ երկպրոցեսորանի ՀՀ-ում MESI արձանագրության կիրառման դեպքում քեշ-հիշողության բլոկների վիճակների քանակը 6-ն է (բացառվում են երկու քեշերում տարբեր բլոկների միաժամանակ M և E վիճակները) և այլն:

Աղյուսակ

Հսկող արձանագրություն	Երկպրոցեսորանի համակարգ	n-պրոցեսորանի համակարգ
MESI	6	$n + 2^n$,
MOESI	10	$n + 2^n(1 + \frac{n}{2})$
MESIF	8	$2^n(1 + \frac{n}{2})$

Հիմնվելով քեշի կոհերենտության ապահովման համար դիտարկված հսկող արձանագրությունների աշխատանքների սկզբունքների և մշակված ալգորիթմների բլոկ-սխեմաների վրա՝ կատարվել են ապարատային իրագործումները սարքերի նկարագրման Verilog լեզվով, գնահատվել են մշակված սարքերի արագագործությունները: Սարքերի սիմուլյացիան իրականացվել է EDA Playground առցանց սիմուլյատորի կիրառմամբ 2,5 *ԳՀց* հաճախականության սինքրոազդանշանի համար, որի նպատակով գեներացվել է սինքրոազդանշան 0,4նվ պարբերականությամբ (always #0,2 clk=~clk): Համաձայն ստացված արդյունքների՝ քեշի կոհերենտության ապահովման համար դիտարկված հսկող արձանագրությունների ապարատային իրագործման արագագործությունը կազմում է 8–10 նվ, կախված արձանագրության տեսակից (MESI, MOESI, MESIF):

Կատարվել են նաև այս ալգորիթմների ծրագրային իրագործումները Java ծրագրավորման լեզվով, և գնահատվել ծրագրային իրագործումների արագագործությունն ու համեմատումը ապարատային իրագործման արագագործության հետ:

Ծրագրի աշխատանքն իրականացվել է 2,5 *ԳՀց* հաճախականությամբ պրոցեսորի վրա: Ծրագրի աշխատանքի սկզբում քեշ-հիշողության բլոկների վիճակի փոփոխությունները (ընդունենք դրանք որպես մեկ տակտ) կատարվել են բավականին մեծ ժամանակում՝ մոտավորապես, միջին տվյալներով, մեկ միլիոն նանովայրկյանի ընթացքում: Սակայն մի քանի տակտ հետո նկատվել է ժամանակի շատ կտրուկ նվազում, և սկսած այդ պահից, մեկ տակտի կատարման ժամանակը միջինում կազմել է 500 նվ: Սա բացատրվում է ծրագրի կատարման ընթացքում կոմպիլյատորի, մասնավորապես՝ **JIT** (Just In Time) կոմպիլյատորով կատարվող օպտիմացիայով: Ինչպես հայտնի է, Java-ով գրված ծրագրային կոդը **javac** կոմպիլյատորի միջոցով ձևափոխվում է բայթ կոդի, որն այնուհետև **JVM**-ի (Java Virtual Machine) միջոցով թարգմանվում է Ասեմբլերի կոդի և իրականացվում պրոցեսորի վրա: JIT-ով իրականացվող օպտիմացումները կատարվում են հենց Ասեմբլերի մակարդակով, այդ պատճառով դիտվում է ծրագրի կատարման ժամանակի կտրուկ նվազում:

Սակայն, համեմատելով ապարատային իրագործման հետ, արագագործության տարբերությունը շատ զգալի է. այս դեպքում մոտավորապես 50 անգամ ավելի բարձր է: Սա պայմանավորված է նաև այն փաստով, որ Java-ն չի համարվում արագագործ ծրագրային լեզու՝ համեմատած այլ լեզուների հետ (C/C++):

Եզրակացություն.

• Հետազոտվել և մշակվել են ժամանակակից բազմապրոցեսորային համակարգերում քեշ-հիշողության կոհերենտությունն ապահովող MESI, MOESI, MESIF հսկող արձանագրությունների ապարատային և ծրագրային իրագործումները:

• Կատարված են բոլոր սարքերի ապարատային իրագործումները Verilog HDL-ով և ծրագրայինները՝ Java լեզվով:

• Կատարվել է այդ իրագործումների արագագործությունների համեմատում:

Ելնելով վերը նշվածից՝ կարելի է եզրակացնել, որ մշակված սարքերի արագագործությունները գերազանցում են նույն ալգորիթմներով մշակված ծրագրային իրագործումների արագագործությունները մոտավորապես 50 անգամ: Բազմապրոցեսորային համակարգերի քեշ-հիշողությունների կոհերենտության խնդրի լուծման համար կիրառվող դիտարկող արձանագրությունների ապարատային իրագործումը փաստացի կրճատում է պրոցեսոր-հիշողություն արագագործությունների տարբերությունը:

ԳՐԱԿԱՆՈՒԹՅԱՆ ՑԱՆԿ

1. **Антонов А. С.** Введение в параллельные вычисления.-М.: НИВЦ МГУ, 2012.-69 с.
2. **Баранов С. И.** Архитектура компьютеров.-Минск: Новое знание, 2016.-559 с.
3. **Воеводин В.** Параллельные вычисления.-СПб.: БХВ-Петербург, 2012.-608 с.
4. **Столлингс У.** Структурная организация и архитектура компьютерных систем.-5-е изд.-М.: Изд. дом “Вильямс”, 2012.-896 с.
5. **Таненбаум Э.** Архитектура компьютера.-5-е изд.-СПб.: Питер, 2007. - 843 с.
6. **Цилькер Б. Я., Орлов С. А**. Организация ЭВМ и систем.-СПб.: Питер, 2014.-668 с.
7. **Цилькер Б. Я., Пятков В.П.** Архитектура вычислительных систем. - Рига: TSI, 2011.-249 с.
8. **Archibald, J.A.** The Cache Coherence Problems in Shared-Memory Multi-processors: Technical Report.-University of Washington, Feb. 2007. - 233p.
9. **Berger, A.** Hardware and Computer Organization: The Software Perspective.-Elsevier, 2012.-255p.
10. **Hennessy J.L., Patterson, D.A.** Computer Organization and Design: The Hardware/ Software Interface.-4th Edition.-Morgan Kaufmann Publishers, San Francisco, CA, USA, 2013.-306p.
11. http://db4.sbras.ru/elbib/data/show_page.phtml?77+854+1213
12. https://www.realworldtech.com/common-system-interface/5/

А.К. САГАТЕЛЯН, А.В. ЦАРУКЯН, Г.А. ШАХНАЗАРЯН

СРАВНЕНИЕ АППАРАТНОЙ И ПРОГРАММНОЙ РЕАЛИЗАЦИЙ ПРОТОКОЛОВ НАБЛЮДЕНИЯ КОГЕРЕНТНОСТИ КЭШ-ПАМЯТИ В МНОГОПРОЦЕССОРНЫХ СИСТЕМАХ

Исследованы проблемы когерентности кэш-памяти в многопроцессорных системах и современные типы используемых при этом наблюдательных протоколов, таких как протоколы MESI, MOESI и MESIF. Выполнены сравнения быстродействия этих алгоритмов при аппаратной и программной реализациях.

Ключевые слова: многопроцессорные системы, когерентность кэш-памяти, протоколы MESI, MOESI и MESIF, аппаратная и программная реализации.

A.K. SAGHATELYAN, H.V. TSARUKYAN, G.A. SHAHNAZARYAN

CACHE MEMORY COHERENCE, COMPARISON OF HARDWARE AND SOFTWARE OF THE MONITORING PROTOCOLS IN MULTIPROCESSOR SYSTEMS

The cache memory coherence problems in multiprocessor systems and the modern types of monitoring protocols such as MESI, MOESI and MESIF implemented in that case are studied, comparisons of the speed of those algorithms at hardware and software implementations are also carried out.

Keywords: multiprocessor systems, coherence of cache memory, MESI, MOESI, MESIF controlling (or monitoring) protocols, comparison of hardware and software implementations.

ՀՏԴ 681.3

Ա.Ժ. ՄՈՄՋՅԱՆ

ԲՈՒԺՄԱՆ ԳՈՐԾԸՆԹԱՑԻՆ ԱՋԱԿՑՈՂ ՀԵՌԱՀԱՂՈՐԴԱԿՑԱԿԱՆ ՀԱՄԱԿԱՐԳԻ ՏՎԻՉՆԵՐԻ ՀԵՏԱԶՈՏՈՒՄԸ ԵՎ ԸՆՏՐՈՒԹՅՈՒՆԸ

Հետազոտվել են մեթաբոլիկ համախտանիշով հիվանդությունների բուժման գործընթացին աջակցող համակարգի IoT ցանցի միջոցով հիվանդների տվյալների մշտադիտարկման տվիչներ: Որոշվել են դրանց ճշգրտությունը, անվնասությունը, API-ները և SDK-ները, և կատարվել է համապատասխան ընտրություն:

Առանցքային բառեր. խելացի կրիչներ, տվիչներ, IoT, SDK, MKM-HealthDataSDK:

Ներածություն: Հիվանդների արդյունավետ մշտադիտակումը առանցքային դերակատարություն ունի հիվանդների առողջական վիճակին առավելագույնս հարմարեցված բուժման պլանի մշակման և հետևաբար՝ լավագույն