ARM Cortex-A. Кеш памети. Йерархия на паметта. Инструкции за работа с кеш.

Автор: доц. д-р инж. Любомир Богданов

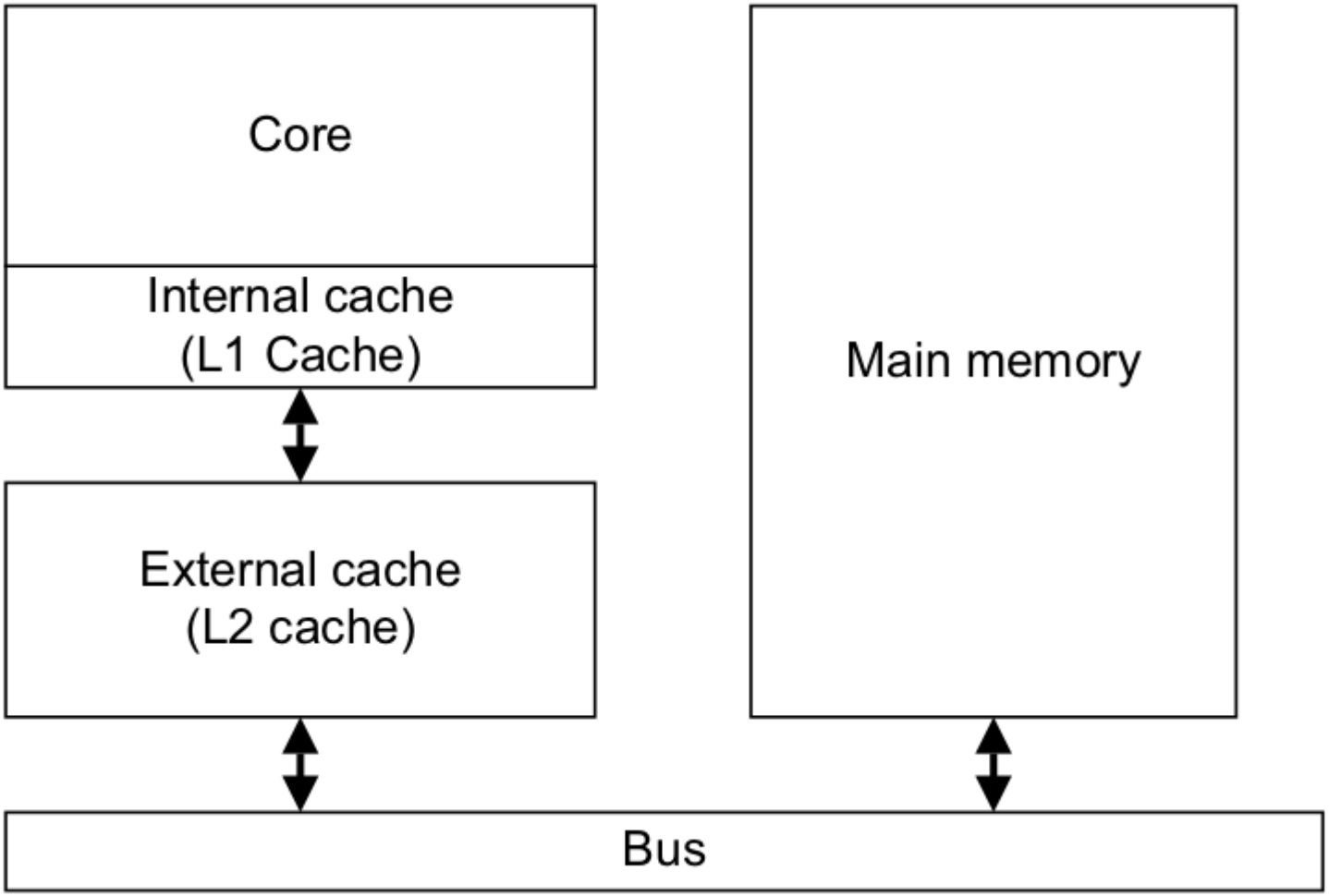
Съдържание

Кеш памети - въведение

*__Кеш памет__ – буферна памет за инструкции/данни, която съгласува бързото процесорно ядро с бавната основна памет (RAM и ROM).

*Думата кеш идва от френски език “cacher” – крия [1].
*От гледна точка на програмиста, кешът е скрит (прозрачен) – не се срещат инструкции за копиране на инструкции/данни от основната памет в кеша и обратно.
*Все пак има странични ефекти, за които програмиста е добре да знае.

Кеш памети - въведение



Кеш памети - въведение

*При първите версии на ARM, работната честота на процесора и паметта са били еднакви [1].
*Съвременните версии на ARM работят на честота много по-висока от паметта.
*On-chip SRAM може да достигна честотата на процесора, но тя е много **по-скъпа** от външната DRAM.
*В съвременните реализации, достъпът до външна DRAM става за **десетки и стотици** тактове на ядрото.

Кеш памети - въведение

*Мястото на кеша се намира между процесорното ядро и основната памет.

*Съдържа копия на елементи от основната памет.

*Достъпът до кеш е много по-бърз от достъпа до основната памет.

*Кешът **буферира данни/инструкции**, както и **адресите** на които се намират те.

*Когато ядрото иска да пише/чете, то първо проверява кеша за наличност на данните/инструкциите. Ако бъдат открити, ще бъдат използвани.

*Това подобрява:

-бързодействието

-консумираната енергия

Кеш памети - въведение

*Кеш паметите са с малки размери, в сравнение с основната памет.
*Чипове с големи кешове са скъпи чипове.

***Temporal locality** – свойството на програмите да достъпват данни в едни и същи обхвати от адреси за определен период от време.

***Spatial locality** - свойството на програмите да достъпват данни в съседни обхвати от адреси за определен период от време.

Кеш памети - въведение

Примери:
*цикли – един и същ код се изпълнява много пъти;
*функция – може да бъде извиквана много пъти;
*достъп до стек – стекът е малък регион от паметта.

Кеш памети - въведение

Изходен буфер (write buffer) – аналогичен на кеш, съгласува бързото ядро с бавната памет при инструкции за запис (store). Ядрото записва *адреси, данни и контролни* сигнали в хардуерен буфер, когато срещне тази инструкция.

Това позволява ядрото да продължи със следващите инструкции (освен ако не са също store) и не трябва да чака записът да завърши (бавен процес).

Кеш памети - въведение

*Поместването на кода в бърза памет го прави бърз на втора и всяка следваща итерация.

*На **първата итерация**, обаче, изпълнението е **бавно**, защото трябва да се изчака зареждането от основната памет.

*Достъпът до **периферията** (входно-изходните устройства на и извън чипа) трябва да бъде **некеширан**.

Кеш памети - въведение

Недостатъци:
*времето за изпълнение на програмата става недетерминистично => не са подходящи за hard real-time;
*достъпът до входно-изходни устройства изисква допълнителни операции;
*при повече от едно ядро, кешовете стават некохерентни;

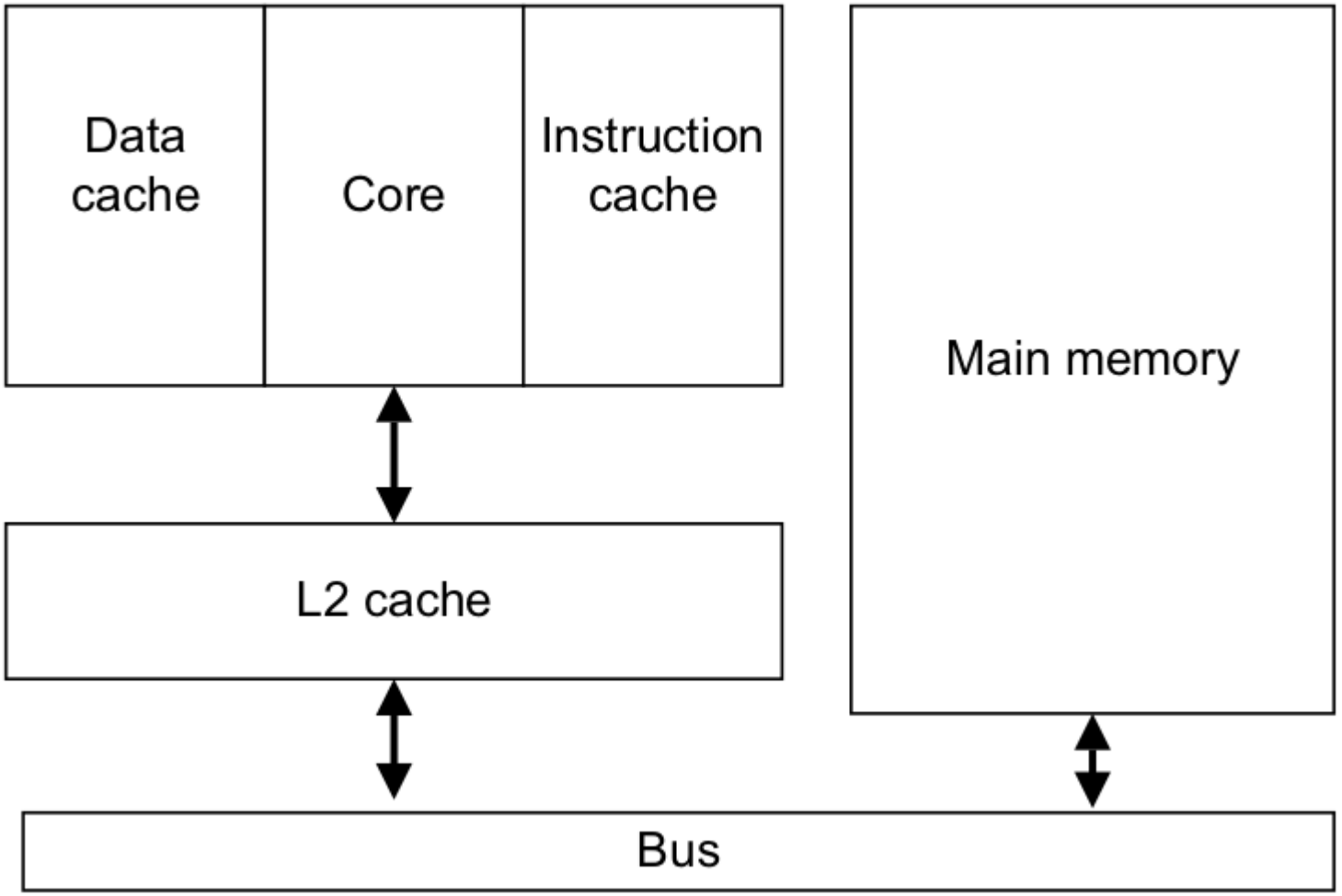
Кеш памети - въведение

В ARM Cortex-A микропроцесорите кеш на ниво 1 (level 1 cache, **L1**) е свързан по схема Харвард и е част от логиката на ядрото (вж сл. слайд).

L1i отговаря за **извличането на инструкции.**
L1d отговаря за **четенето/записите на данни** при load/store инструкции.

Типичен размер на L1 за ARM Cortex-A е 16 kB и 32 kB.

Кеш памети - въведение



Кеш памети - въведение

В ARM Cortex-A микропроцесорите кеш на ниво 2 (level 2 cache, **L2**) е свързан по схема фон Нойман. Може да е част от логиката на ядрото, но може и да е външен (за ядрото) модул – например да бъде споделен между две и повече ядра. L2 е по-бавен от L1.

L2 съдържа инструкции и данни.

Типичен размер на L2 за ARM Cortex-A е 256 kB, 512 kB, и 1024 kB.

Пример: ARM L2C-310 е L2 кеш за ARM Cortex-A процесори.

Кеш памети - въведение

Кеш стойност притежава следните полета:
*адрес
*данни
*статус информация

Кеш памети - въведение

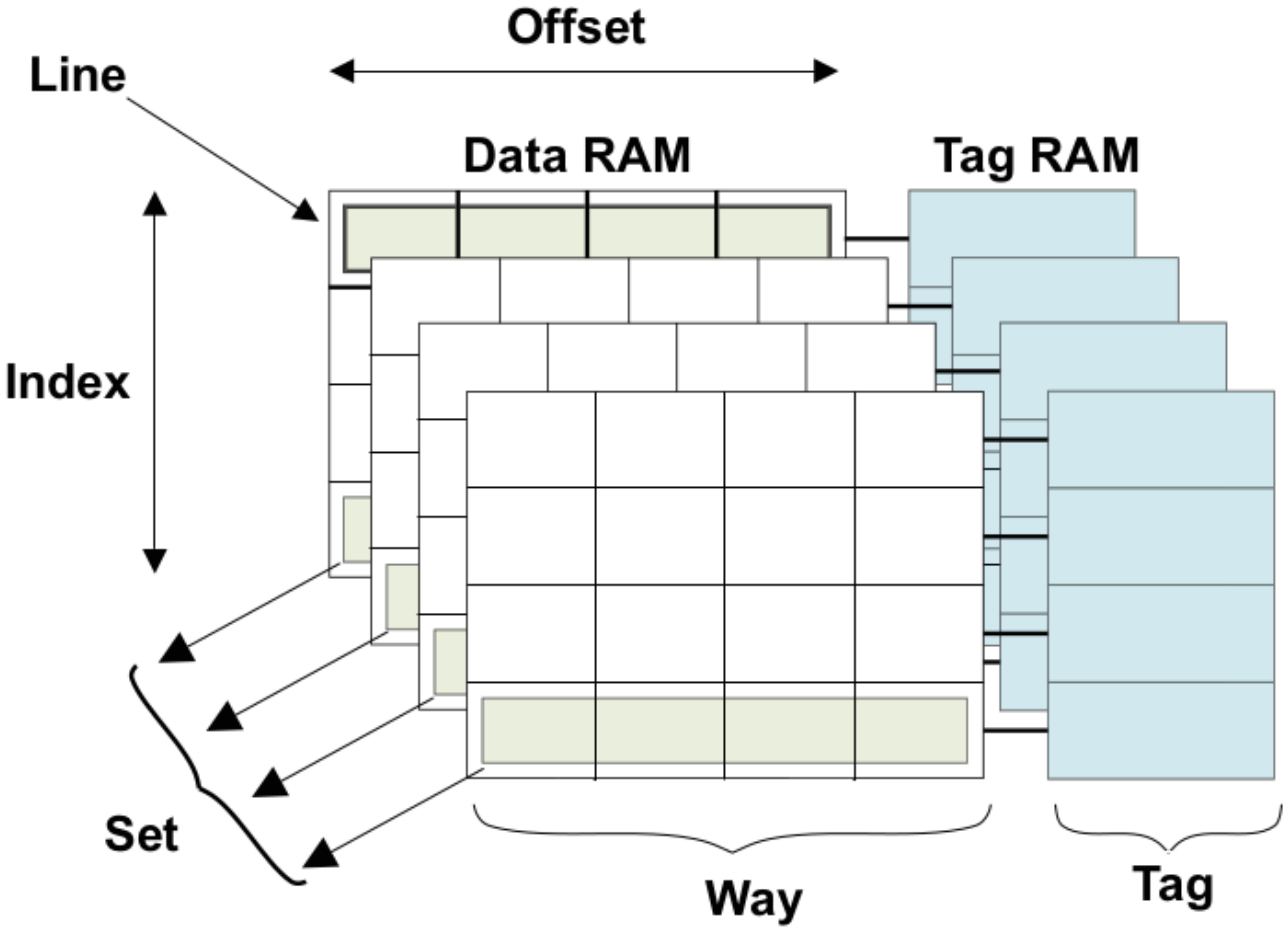


Figure 8-3 Cache terminology

Структура на ARM кеш памет [1].

Кеш памети - въведение

***Line** – блок от данни/инструкции на последователни адреси от основната памет, който е копиран в кеша. Кеш линията е най-малката неделима единица информация в кеш паметта.

***Index** – число, указващо къде в кеша може да бъде намерена дадена кеш линия.

***Way** – група от кеш линии.

***Set** – група от кеш линии, на един и същи индекс, но в различни кеш way групи.

***Tag** – част от адрес от основната памет, съхранен в кеш паметта, който указва къде в основната памет се намират данните, чийто копия са буферирани в кеш линията.

Кеш памети - въведение

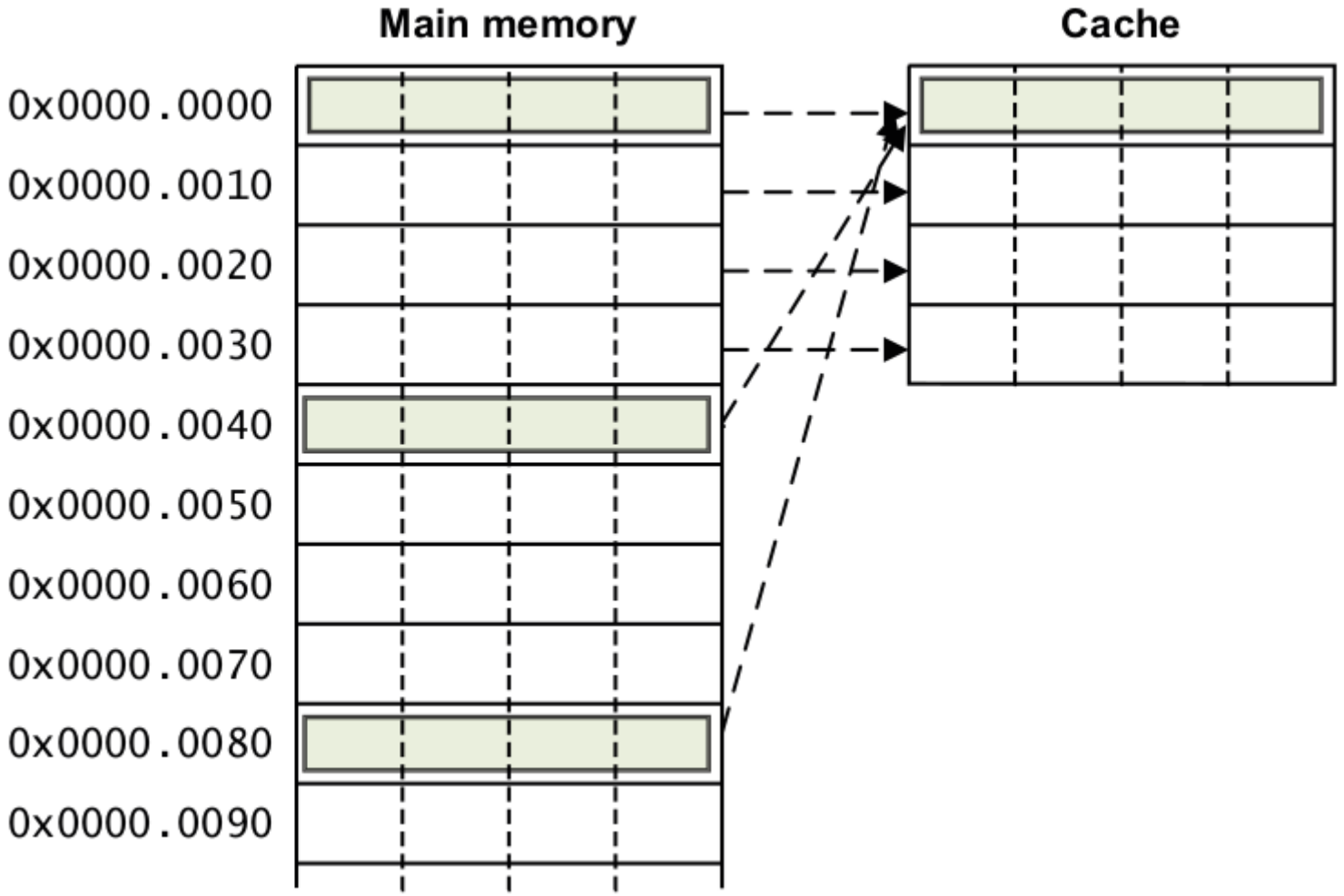
Видове кеш:
1. Директен (**direct mapped**)
2. Сет-асоциативен (**set associative**)

Кеш памети - въведение

Директен кеш (direct mapped) – адреси от основната памет се съпоставят с точно определени адреси от кеша.

Понеже основната памет е много по-голяма от кеша, много адреси от нея ще бъдат съпоставени (mapped) на **точно един адрес от кеша**.

Кеш памети - въведение



Пример: директен кеш с 4 думи във всяка линия и с общо 4 линии [1].

Кеш памети - въведение

Формирането на адреса при директния кеш от адреса в основната памет (на миналия слайд) става по следния начин [1]:

*битове[1:0] – избират байт от дума
*битове [3:2] - избират дума от линия
*битове [5:4] – избират линия
*битове [31:6] – кеш таг

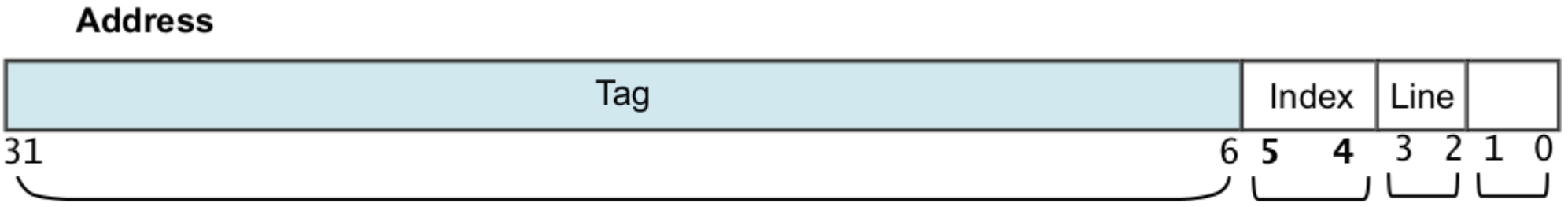


Figure 8-5 Cache address

Кеш памети - въведение

Когато дадена инструкция се опита да достъпи данни на определен адрес от основната памет, първо контролера на кеша проверява дали старшата част на адреса (битове [31:6]) съвпада с някой от кеш таговете.

Ако да, проверява се един бит наречен “**валиден бит**”, показващ дали данните отговарят на това, което има в основната памет. Ако да, казва се, че има **кеш попадение** (cache hit). Тогава, инструкцията ще получи данните си от кеша, а не от основната памет.

Кеш памети - въведение

Ако кеш тагът не съвпадне (събитието се нарича кеш пропуск, **cache miss**), трябва да се зареди друга линия от основната памет на мястото на настоящата (напомняне – една кеш линия отговаря за няколко адреса от основната памет), което ще принуди процесора да блокира (stall), докато завърши операцията.

Кеш памети - въведение

Проблем: ако няколко линии от основната памет се съпоставят само на едно място в кеша и ако тези линии участват в цикъл.

```
int i, result[10], arr1[10], arr2[10]; //линкерът дава
                                       //следните адреси
                                       //result = 0x00
                                       //arr1 = 0x40
                                       //arr2 = 0x80
for (i = 0 ; i < 10 ; i++) {
    result[i] = arr1[i] + arr2[i];
}
```

Кеш памети - въведение

НО от примерното съпоставяне на по-миналия слайд на адреси от основната памет към кеш линии се вижда, че всички тези адреси (0x00, 0x40, 0x80) сочат към една кеш линия, и процесорът ще е принуден да се обръща към основната памет при всеки достъп.

Процесорът ще работи все едно няма кеш.

Този проблем се нарича **thrashing**.

Кеш памети - въведение

Целият процес описан в стъпки:
*при първо четене от 0x40 ще настъпи кеш пропуск и контролерът на кеша ще зареди една линия – всичките думи от адрес 0x40 до адрес 0x4f;
*следващото четене е от адрес 0x80 – кеш пропуск, ще се зареди 0x80-0x8f на **мястото** на 0x40-0x4f;
*резултатът се записва в 0x00 – кеш пропуск, ще се зареди 0x00-0x0f на **мястото** на 0x80-0x8f;
*процесът ще се повтори във всичките 10 итерации на цикъла и производителността на програмата ще е ниска.

Кеш памети - въведение

За основни кешове на ARM не се използват директни кешове поради ниската производителност на програмата, която се получава в следствие на кеш пропуските.

Директни кешове се използват в BTAC, например в ARM1136 [1].

Кеш памети - въведение

Сет асоциативни кешове (set associative) – разделят се на блокове, наречени ways. Адрес от основната памет се съпоставя на way блок, а не на кеш линия. Индексът на адреса продължава да сочи към линия, но тук сочи към всички линии с този индекс от даден сет.

За настъпване на кеш съвпадение трябва да се обходят всички тагове на сет-а.

Кеш памети - въведение

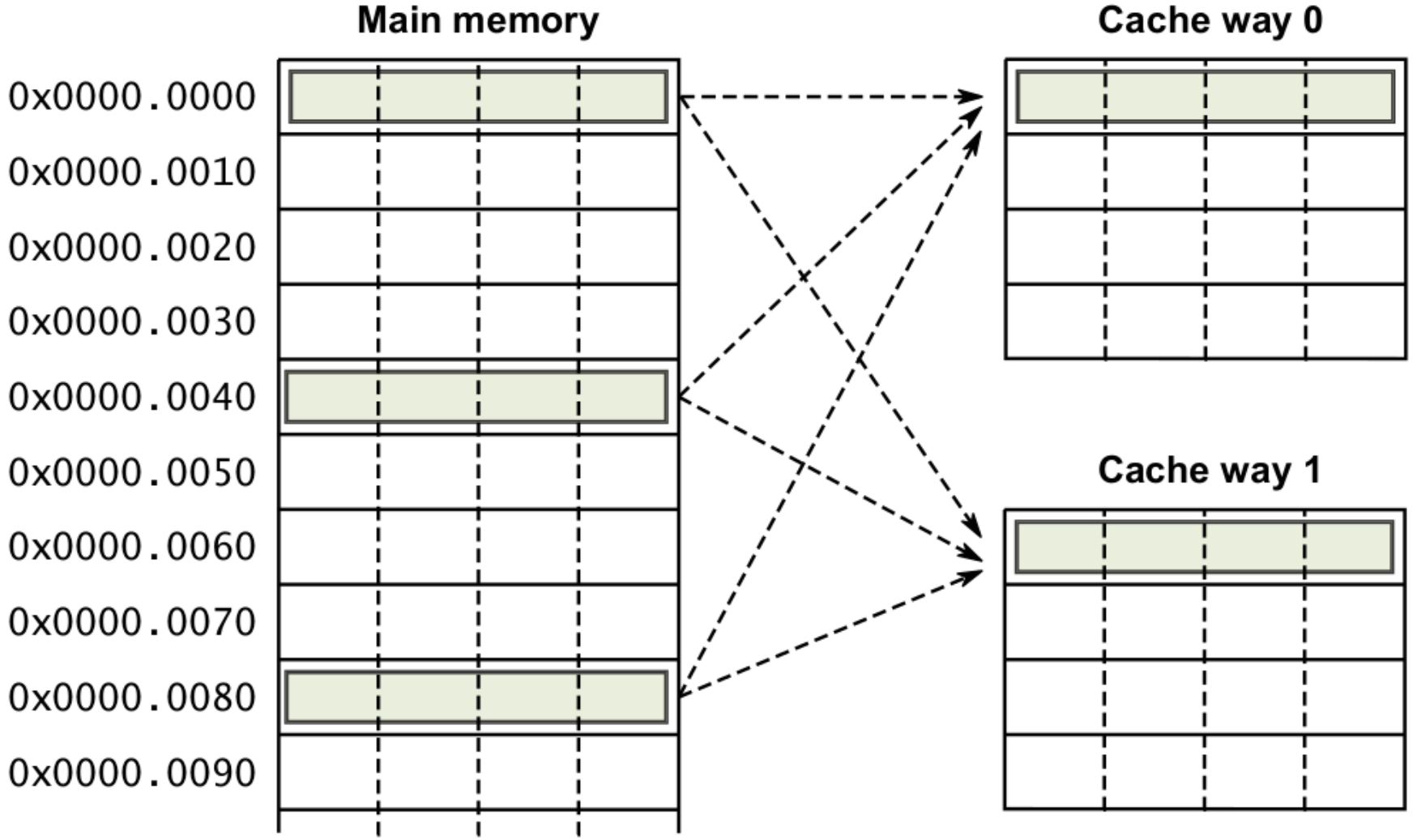


Figure 8-6 A 2-way set-associative cache

Пример: 2-way сет асоциативен кеш [1].

Кеш памети - въведение

По-голяма асоциативност, води до **по-малка вероятност** от настъпване на thrashing.

Напълно асоциативен кеш (fully associative cache) – кой да е адрес от основната памет може да бъде съпоставен (зареден) на коя да е линия от кеша. Много сложен хардуер, рядко се използва.

Повечето Level 1 кешове имат до 4-way блока.
Level 2 кешове, понеже са по-големи, се реализират с 8- и 16-way блока.

Кеш памети - въведение

Всички основни кешове на ARM Cortex са сет асоциативни.

Типично way блоковете са 2 или 4 броя, но има имплементации дори с повече.

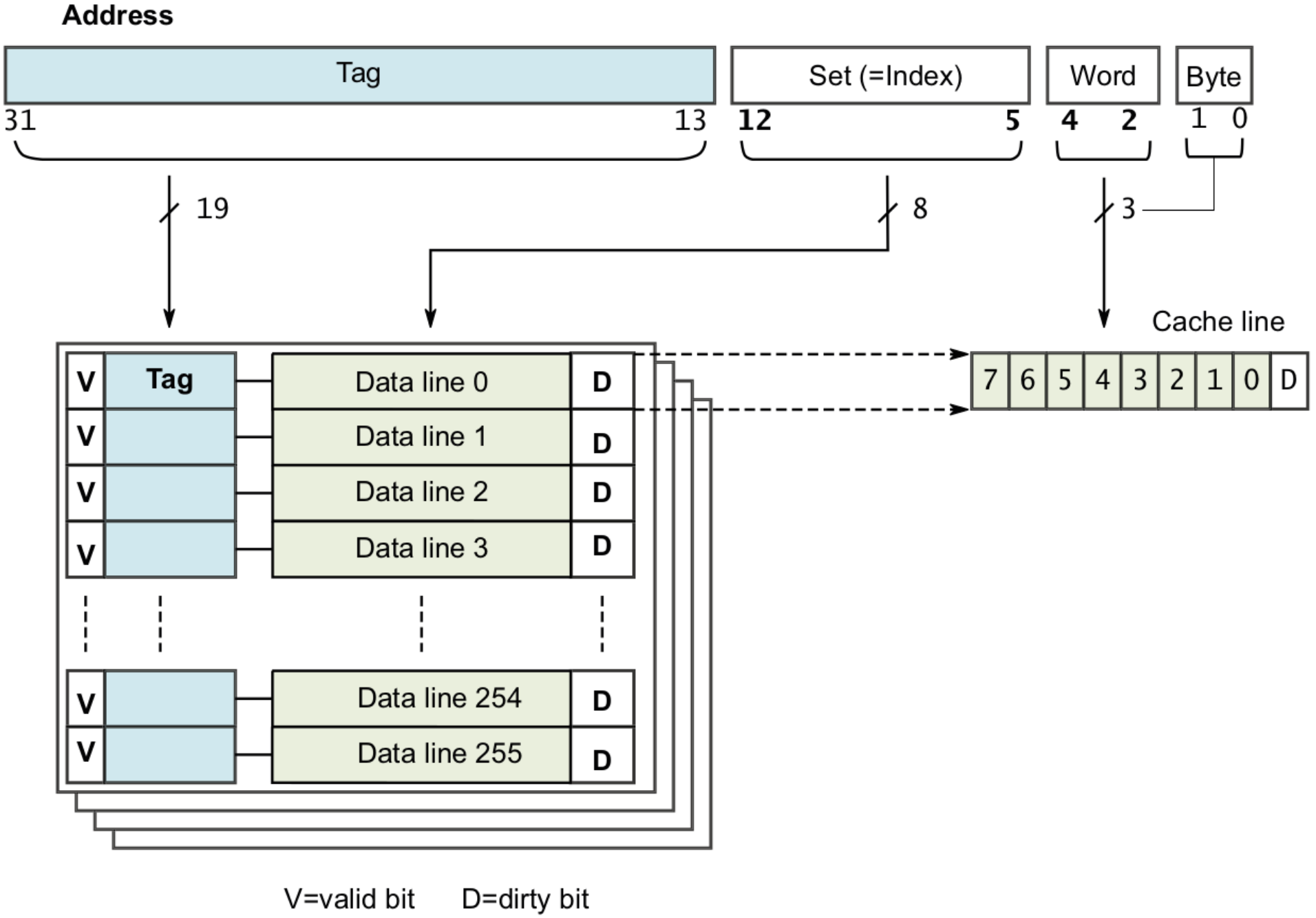
Кеш памети - въведение

Пример: на следващия слайд е показана структурна схема на 32 kB 4-way сет асоциативна даннова кеш памет с дължина на линията 8 думи.

Такъв вид кеш се използва в ARM Cortex-A7 и A9 процесори.

*битове [1:0] – индекс на байт от дума
*битове [4:2] – индекс на дума от линия
*битове [12:5] – индекс на линия в way блок
*битове [31:13] – адрес на таг

Кеш памети - въведение



Кеш памети - въведение

Кеш контролер – хардуерен блок, който управлява достъпите до кеш паметта. Той автоматично копира данни/инструкции от основната памет в кеша.

Кеш контролерът получава сигнали от ядрото за достъп (**requests**) до паметта. Той сравнява част от битовете на изискваният от ядрото адрес с наличните тагове. Процесът на срявняване се нарича **look-up.** Ако има такъв таг (**hit**), и ако линията е отбелязана като валидна, тогава четенето/записа ще станат от/в кеша, а не от основната памет.

Кеш памети - въведение

Ако няма такъв таг (**miss**), и ако линията е отбелязана като **невалидна**, достъпът трябва да бъде насочен към следващото ниво кеш или към основната памет. Възможно е да се предизвика и запис на нови данни/инструкции в кеш линията (**linefill**) от паметта, едновременно с което те ще бъдат предадени на ядрото.

Ядрото може да не изчака целия linefill да завърши. Това е възможно, понеже кеш контролерът достъпва първо критична дума (**critical word**) от кеш линията, т.е. точно тази която се изисква, а след това останалите от линията.

Кеш памети - въведение

Кеш схеми на опресняване:

*схема на зареждане (**allocation policy**) – указва събитието, което ще предизвика зареждане на данни/инстр. от основната памет в кеш;

*схема на изтриване (**replacement policy**) – указва, в коя линия да бъдат заредени новите данни/инстр., които ще изтрият старите; линията, която ще бъде изтрита се нарича **victim**;

*схема на запис (**write policy**) – указва какво събитие да стане при запис на данни от ядрото в кеша.

Кеш памети - въведение

Схеми на зареждане:
*Схема на зареждане при четене (**read allocate**) – кеш линия се зарежда при четене, в случай на кеш miss [a]. Ако при запис има кеш miss, кешът не се променя, а се отива към следващото ниво кеш/основна памет [1].

*Схема на зареждане при запис (**write allocate**) – кеш линия се зарежда при четен или запис, в случай на кеш miss.

Кеш памети - въведение

Кеш схеми на изтриване – индекс битовете от адреса избират кеш сет, а схемата на изтриване – конкретна линия от сета:

Ако victim линията съдържа валидни и мръсни данни, те трябва да бъдат записани обратно в основната памет, преди да бъдат изтрити (=eviction).

***Round-robin** (или още cyclic) - използва се брояч на way-модулите. Неговото число указва way-модул, който ще съдържа victim линията. При достигане на максималното число, започва да брои от нула.

Кеш памети - въведение

***Pseudo-random** – избира се кеш линия на произволен принцип. Използва се брояч, изполващ псевдослучаен закон.

***Least Recently Used** (LRU) – избира се кеш линия, която е най-рядко използвана.

Пример: повечето ARM Cortex използват round-robin и pseudo-random. ARM Cortex-A15 използва LRU.

Кеш памети - въведение

Схема на запис:

***Write-through** – записът в кеша предизвиква запис в основната памет. Така кеш и памет са **кохерентни**. *Бавно*

***Write-back** – записът в кеша не се отразява в основната памет. Това означава, че кеш линиите и основната памет имат различни данни (т.е. са **некохерентни**). Старите данни в основната памет се наричат (stale). Поради това, всяка линия от кеша има **dirty** бит, указващ че има разлика. При eviction, dirty битът се занулява, а данните в основната памет се уеднаквяват с кеш линията. *Бързо*

Кеш памети - въведение

***Write-buffer** – FIFO буфер приемащ данни/контролни сигнали за записи в паметта. Докато записът от FIFO буфера в основната памет завърши, процесорът може да изпълнява други инструкции. Това “скрива” бавното функциониране на паметта от процесора, **но само ако буферът не се напълни**. Ако буферът се напълни (програмата извършва чести записи), системата процесор-памет ще работи все едно няма такъв буфер [1].

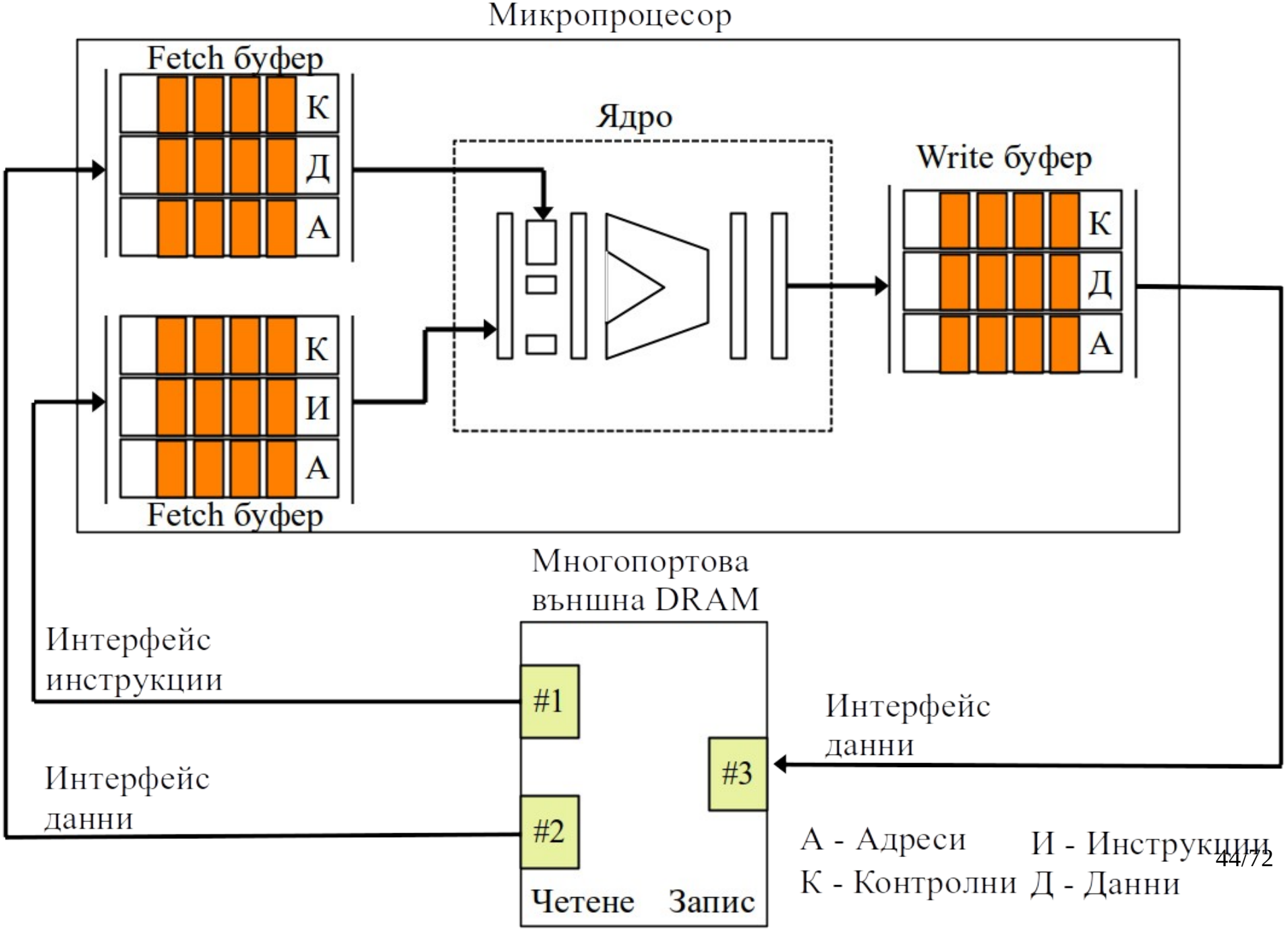
Този буфер трябва да се има предвид, когато се достъпват входно/изходни устройства.

Кеш памети - въведение

***Fetch-buffer** – FIFO буфер приемащ данни/контролни сигнали за четене от паметта. Когато процесорът опита да чете инструкции/данни това ще принуди fetch буфера да изтегли няколко последователни думи от основната памет. Това “скрива” бавното функциониране на паметта от процесора, **но само ако буферът не се напълни**. Ако буферът се напълни (програмата извършва четене често), системата процесор-памет ще работи все едно няма такъв буфер [1].

Този буфер трябва да се има предвид, когато се достъпват входно/изходни устройства.

Кеш памети - въведение



Кеш памети - въведение

Cache hit rate = cache hit / total memory requests, [%]

Cache miss rate = cache miss / total memory requests, [%]

Кеш памети - въведение

Изчистване на кеш (**flush**) – състои се от две операции:
***invalidation** на целия кеш / на кеш линия – битът clean се занулява на всички думи;

***clean** на целия кеш / на кеш линия – записване на dirty думите от кеш в основната памет, занулляване на dirty бита. Тази операция е валидна само за кеш с write-back схема.

Тези операции се извършват от CP15 копроцесора. Те са достъпни само за привилегирован код.

Кеш памети - въведение

Пример: когато ARM Cortex-A ядро промени данните в кеша и след това се използва DMA, за да прехвърли тези данни от едни адреси на други, кешът трябва да се clean-не.

Пример: ако DMA промени данните в основната памет, трябва да са опреснят копията им в кеша, което може да стане с invalidate-ване на съответните линии.

Характеристики на Cortex-A кеш

	Processor					
	Cortex-A5	**Cortex-A7**	**Cortex-A8**	**Cortex-A9**	**Cortex-A12**	**Cortex-A15**
L2 Cache	External	Integrated	Integrated	External	Integrated	Integrated
L2 Cache size	-	128KB to 1MB[a]	0KB to 1MB[a]	-	256KB to 8MB	512KB to 4MB[a]
Cache Implementation (Data)	PIPT	PIPT	PIPT	PIPT	PIPT	PIPT
Cache Implementation (Instruction)	VIPT	VIPT	VIPT	VIPT	VIPT	PIPT
L1 Cache size (data)[a]	4K to 64K[a]	8KB to 64KB[a]	16/32KB[a]	16KB/32KB/64KB[a]	32KB	32KB
Cache size (Inst)[a]	4K to 64K[a]	8KB to 64KB[a]	16/32KB[a]	16KB/32KB/64KB[a]	32KB or 64KB	32KB
L1 Cache Structure	2-way set associative (Inst) 4-way set associative (Data)	2-way set associative (Inst) 4-way set associative (Data)	4-way set associative	4-way set associative (Inst) 4-way set associative (Data)	4-way set associative (Inst) 4-way set associative (Data)	2-way set associative (Inst) 2-way set associative (Data)
L2 Cache Structure	-	8-way set associative	8-way set associative	-	16-way set associative	16-way associative

Характеристики на Cortex-A кеш

	Processor					
	Cortex-A5	**Cortex-A7**	**Cortex-A8**	**Cortex-A9**	**Cortex-A12**	**Cortex-A15**
Cache line (words)	8	8	16	8	-	16
Cache line (bytes)	32	64	64	32	64	64
Error protection	None	None	L2 ECC	None	L1 None, L2 ECC	Optional for L1 and L2

a. Configurable

Physically Indexed, Physically tagged, PIPT – тагът е физически адрес, индексът – също.

Virtually Indexed, Physically tagged, VIPT – тагът е физически адрес, индексът е виртуален адрес, даден от MMU. При смяна на виртуално-физическият мапинг на MMU, няма нужда да се invalidate-ват кешовете => пести се време.

Изчистване на ARM Cortex-A кеш

Асемблер за кеш flush:

```
setup_caches
      MRC p15, 0, r1, c1, c0, 0          ; Read System Control Register (SCTLR)
      BIC r1, r1, #1                     ; mmu off
      BIC r1, r1, #(1 << 12)             ; i-cache off
      BIC r1, r1, #(1 << 2)              ; d-cache & L2-$ off
      MCR p15, 0, r1, c1, c0, 0          ; Write System Control Register (SCTLR)
 ;----------------------------------------------
; 1.MMU, L1$ disable
;----------------------------------------------
     MRC p15, 0, r1, c1, c0, 0           ; Read System Control Register (SCTLR)
     BIC r1, r1, #1                      ; mmu off
     BIC r1, r1, #(1 << 12)              ; i-cache off
     BIC r1, r1, #(1 << 2)               ; d-cache & L2-$ off
     MCR p15, 0, r1, c1, c0, 0           ; Write System Control Register (SCTLR)
;----------------------------------------------
; 2. invalidate: L1$, TLB, branch predictor
;----------------------------------------------
     MOV     r0, #0
     MCR     p15, 0, r0, c7, c5, 0       ; Invalidate Instruction Cache
     MCR     p15, 0, r0, c7, c5, 6       ; Invalidate branch prediction array
     MCR     p15, 0, r0, c8, c7, 0       ; Invalidate entire Unified Main TLB
     ISB                                 ; instr sync barrier
;----------------------------------------------
; 2.a. Enable I cache + branch prediction
;----------------------------------------------
     MRC     p15, 0, r0, c1, c0, 0          ; System control register
     ORR     r0, r0, #1 << 12               ; Instruction cache enable
     ORR     r0, r0, #1 << 11               ; Program flow prediction
     MCR     p15, 0, r0, c1, c0, 0          ; System control register
;----------------------------------------------
```

Изчистване на ARM Cortex-A кеш

Асемблерната програма от миналия слайд може да бъде внедрена в код на C чрез **inline assembler** макроси.

ИЛИ

Ако се използва Linux, може да се извика вътрешната (intrinsic) функция от arch/arm/mm/cache-v7.s

void _ _clear_cache(**char** *beg, **char** *end);

Началния адрес beg се включва, но крайния адрес end не.

CP15 за ARM Cortex-A кеш

Копроцесор CP15 [2]:
*Регистър #0 – информация за вида на кеша
*Регистри #1, #7 и #9 – контрол на кеша
*Регистър #6 – статус на кеша

Копроцесор CP15 е **задължителен** за процесори с кеш и архитектури **>= ARMv6**

CP15 за ARM Cortex-A кеш

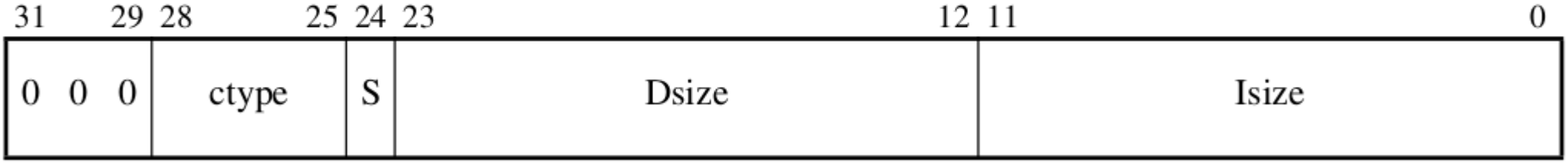
Регистър #0:
*само за четене
*описание на организацията на кеша
*размер на кеша

Пример: четене на регистър #0 от CP15

MRC p15, 0, Rd, c0, c0, 1 ; returns Cache Type register

CP15 за ARM Cortex-A кеш

The format of the Cache Type register is:



Ctype – вид на кеша, използва се следната таблица:

Table B6-1 Cache type values

ctype field	Method	Cache cleaning	Cache lock-down
0b0000	Write-through	Not needed	Not supported
0b0001	Write-back	Read data block	Not supported (deprecated in ARMv6)
0b0010	Write-back	Register 7 operations	Not supported (deprecated in ARMv6)
0b0110	Write-back	Register 7 operations	Format A
0b0111	Write-back	Register 7 operations	Format B (deprecated in ARMv6)
0b1110	Write-back	Register 7 operations	Format C
0b0101	Write-back	Register 7 operations	Format D

CP15 за ARM Cortex-A кеш

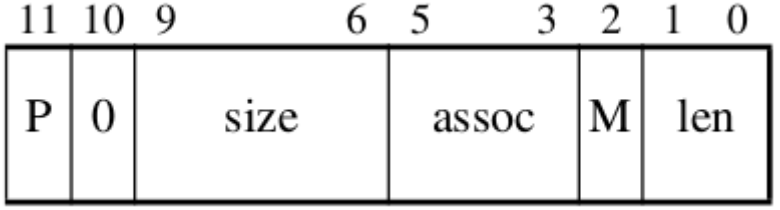
Кеш lockdown – софтуерно влияние върху решенията на кеш контролера за изтриване на линия (replacement policy) [3]. Данни, маркирани като locked остават в кеша и няма да бъдат изтрити, докато софтуера не ги отключи. Това позволява да се постигне много висок hit rate. Използва се в критичен код (напр. хендлери на прекъсванията и real time приложения).

S-бит – при 0 кешът е общ (unified), при 1 кешът е разделен на кеш за инструкции (i-cache) и кеш за данни (d-cache).

CP15 за ARM Cortex-A кеш

D-size – указва размер, дължина на линия и вид асоциативност на данновия кеш.
I-size – указва размер, дължина на линия и вид асоциативност на кеша за инструкции.

D- и I-size битовите полета имат следния формат:



P – при 0 няма ограничения при заделянето на битове 12 и 13 от виртуалния адрес, при 1 има ограничения.

CP15 за ARM Cortex-A кеш

Size и **M** битове – размер на кеша според следната таблица:

Table B6-2 Cache sizes

size field	Size if M == 0	Size if M == 1
0b0000	0.5KB	0.75KB
0b0001	1KB	1.5KB
0b0010	2KB	3KB
0b0011	4KB	6KB
0b0100	8KB	12KB
0b0101	16KB	24KB
0b0110	32KB	48KB
0b0111	64KB	96KB
0b1000	128KB	192KB

CP15 за ARM Cortex-A кеш

len – дължина на кеш линията според следната таблица:

Table B6-3 Cache line lengths

len field	Cache line length
0b00	2 words (8 bytes)
0b01	4 words (16 bytes)
0b10	8 words (32 bytes)
0b11	16 words (64 bytes)

CP15 за ARM Cortex-A кеш

assoc – вид асоциативност на кеша според следната таблица:

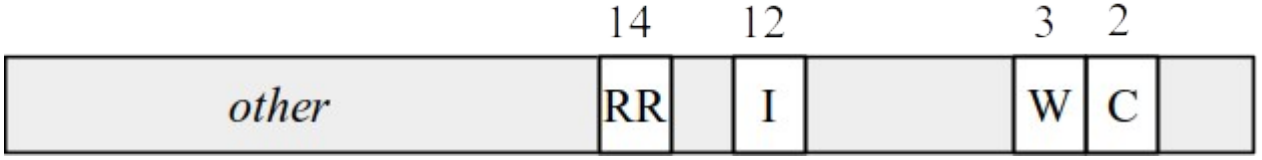
Table B6-4 Cache associativity

assoc field	Associativity if M == 0	Associativity if M == 1
0b000	1-way (direct-mapped)	cache absent
0b001	2-way	3-way
0b010	4-way	6-way
0b011	8-way	12-way
0b100	16-way	24-way
0b101	32-way	48-way
0b110	64-way	96-way
0b111	128-way	192-way

Ако имаме “cache absent”, всички останали битови полета в size са без значение.

CP15 за ARM Cortex-A кеш

Cache and write buffer control bits



Регистър #1:

C – при 0 кешът е забранен, при 1 кешът е разрешен. Ако се използва общ кеш, този бит важи за всички линии. Ако се използва разделен кеш, този бит важи за данновите линии.

W – при 0 се забранява изходния буфер (write buffer), при 1 – се разрешава. Ако се използва общ кеш, или ако кеша за инструкции не е включен в процесора, този бит е винаги нула и записите в него се игнорират. Ако изходния буфер не може да бъде забранен, този бит е винаги единици и записите в него се игнорират.

CP15 за ARM Cortex-A кеш

I - ако се използват разделени кешове, този бит разрешава (1)/забранява (0) кеша за инструкции. Ако се използва общ кеш, или ако не е включен i-cache, този бит е винаги 0 и записите в него се игнорират.
При **ресет**, този бит е **винаги 0**.

RR – при 0 кеш линиите използват обикновена стратегия на зареждане (например random), при 1 се използва предсказуема стратегия (например round-robin).

CP15 за ARM Cortex-A кеш

Регистър #7:
*Само за запис
*Управлява L1 кеш и изходните буфери (write buffers)
*Управлява prefetch буферите
*Управлява BTAC
*Отговаря за WFI (Wait For Interrupt) управлението на тактовия сигнал
*Повечето операции се изпълняват в privileged режим, някои могат и в user режим

Пример: запис в регистър 7

MCR p15,0,<Rd>,c7,<CRm>,<opcode2>

CP15 за ARM Cortex-A кеш

Операции, извършвани с регистър #7:

***clean** – (за write-back d-cache) операцията по отразяване промените на кеш линия в основната памет, ако кеш линията съдържа различаващи се (от основната памет) данни. След това кеш линията се маркира като clean с един бит.

***invalidate** – кеш линия се маркира като невалидна. В нея не може да настъпи кеш hit, докато данните от основната памет не се копират наново в кеш линията.

CP15 за ARM Cortex-A кеш

***prefetch** – данни от основната памет се копират в кеш линия. Това ще стане само ако не настъпи забранен достъп заради MMU/MPU (abort) и само ако регионът е маркиран като “с възможност за кеширане” (cacheable).

***prefetch flush** – занулляване на prefetch буфера за инструкции и зареждането му с актуални инструкции от L1 или iTCM. Използва се за самомодифициращ се код.

CP15 за ARM Cortex-A кеш

Регистър #9:

*управлява кеш lockdown-а
*ARM поддържат 4 формата за lockdown: A, B, C и D
*Формати A, B и C заключват way-блокове
*Формат D заключват думи от линия

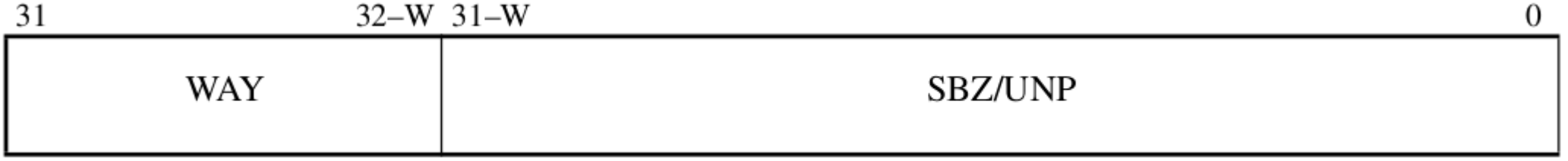
CP15 за ARM Cortex-A кеш

За достъп до регистър 9 се използват следните инструкции:

```
1MCR p15, 0, Rd, c9, c0, 0 ; write unified/data lockdown register
2MRC p15, 0, Rd, c9, c0, 0 ; read unified/data lockdown register
3MCR p15, 0, Rd, c9, c0, 1 ; write instruction lockdown register
4MRC p15, 0, Rd, c9, c0, 1 ; read instruction lockdown register
```

Lockdown блок – кеш way блок, който е бил заключен.

Регистър за управление на формат А:



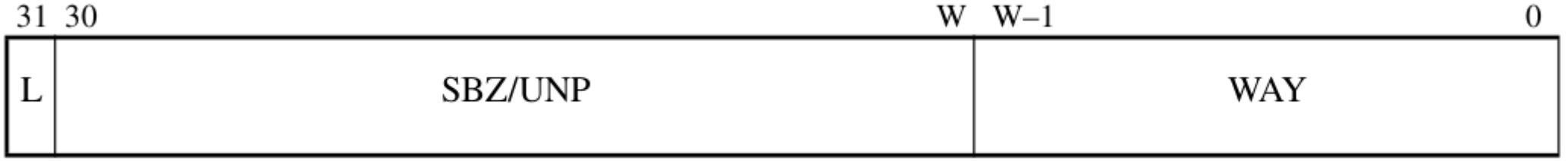
(SBZ, should be zero – винаги нула)

CP15 за ARM Cortex-A кеш

WAY – битово поле, указващо номер на lockdown блока. Този номер съвпада с номера на way блока.

Разделителната линия W = log2(ASSOCIATIVITY) и се закръглява до най-близкото цяло число.

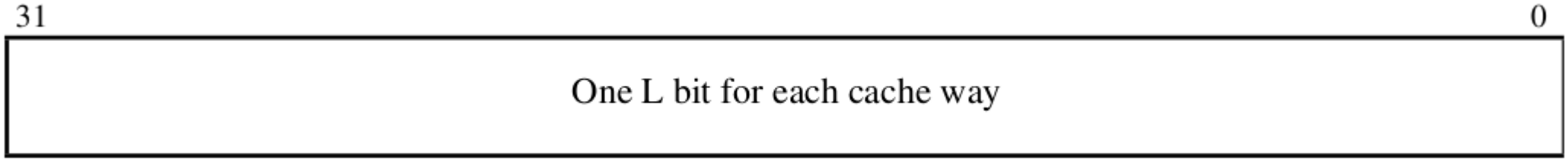
<u>Регистър за управление на формат B:</u> аналогичен на този за формат A.
L – записва се L=1 преди заключването на кеш линии от way блок, заключват се линиите, най-накрая L = 0.



CP15 за ARM Cortex-A кеш

Регистър за управление на формат C: позволява зареждането на кеш way да бъде пускано/спирано.

L (locking bit) – по един бит за всеки кеш way блок, 0 – зареждането става по default алгоритъма на системата, 1 – зареждането е забранено.

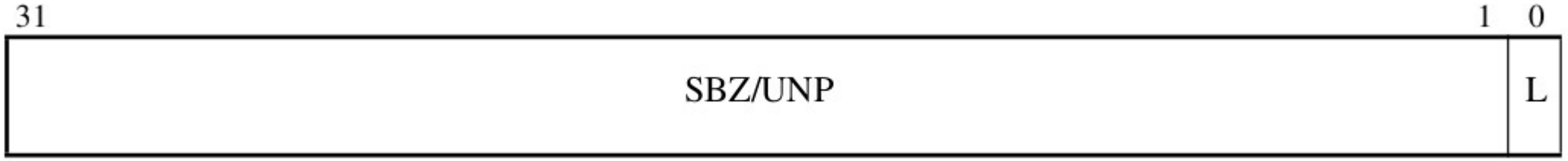


DSB инструкция трябва да бъде изпълнена преди да се пише в този регистър. Това подсигурява, че всички записи, които могат да предизвикат зареждане на линия, ще са приключили.

CP15 за ARM Cortex-A кеш

Регистър за управление на формат D: позволява зареждането на индивидуални думи от L1 кеш линия да бъде пускано/спирано.

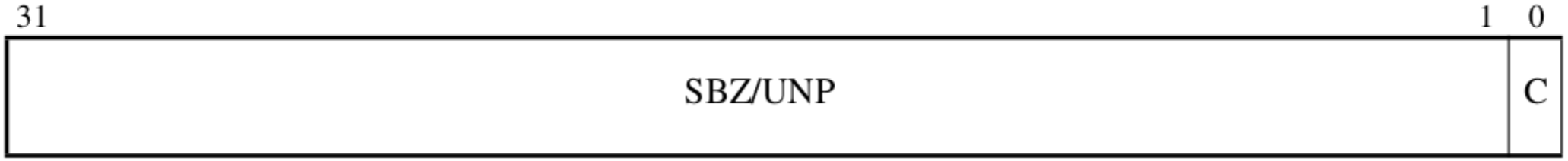
L (locking bit) – 0 – зареждането е разрешено, 1 – зареждането е забранено.



DCLR2 – Data or unified Cache Lockdown Register 2

CP15 за ARM Cortex-A кеш

Table B6-7 Cache Dirty Status register



Регистър #6:

*регистър само за четене

*съдържа само 1 бит

C – при 0 указва, че кешът е чист (clean), т.е. че не е настъпил hit на запис досега от последната операция по чистене. Логическа 1 означава, че кешът може да съдържа dirty линии.

CP15 за ARM Cortex-A кеш

Пример: критичен код трябва да спре прекъсванията и да изчисти кеша. Това гарантира работа с чист кеш.

```
1 ; interrupts are assumed to be enabled at this point
2 Loop1:
3             MOV R1, #0
4             MCR CP15, 0, R1, C7, C10, 0 ; Clean (for Clean & Invalidate
5                                         ; use "C7, C14, 0")
6             MRS R2, CPSR                ; Cache
7             CPSID iaf                   ; Disable interrupts
8             MRC CP15, 0, R1, C7, C10, 6 ; Read Cache Dirty Status Register
9             ANDS R1, R1, #01            ; Check if it is clean
10            BEQ UseClean
11            MSR CPSR, R2                ; Re-enable interrupts
12            B Loop1                     ; Clean the cache again
13 UseClean:
14            Do_Clean_Operations         ; Perform whatever operation relies on
15                                        ; the cache being clean/invalid.
16                                        ; To reduce impact on interrupt latency,
17                                        ; this sequence should be short
18            MCR CP15, 0, R1, C7, C6, 0  ; can use this "invalidate all" command to
19                                        ; optionally invalidate a "clean" loop.
20            MSR CPSR, R2                ; Re-enable interrupts
```

Литература

[1] “ARM Cortex-A Series Programmer’s Guide”, Version 4.0, ARM DEN0013D (ID012214), 2013.
[2] “ARM Architecture Reference Manual”, ARM DDI 0100l, ARM Ltd, 2005.
[3] Kapil Anand and Rajeev Barua, 20XX. "Instruction Cache Locking for Improving Embedded Systems Performance", ACM Trans. Embedd. Comput. Syst. V, N, Article A (January YYYY), 25 pages. DOI:http://dx.doi.org/10.1145/0000000.0000000

Външни връзки

[a] https://developerhelp.microchip.com/xwiki/bin/view/products/mcu-mpu/32bit-mcu/PIC32/mz-arch-cpu-overview/cache-defined/manage/