ARM Cortex-A. Видове инструкции - част 2. Работа с копроцесори.

Автор: доц. д-р инж. Любомир Богданов

Съдържание

Инструкции за работа с копроцесори

ARM Cortex-A имат следните копроцесори†:

*Копроцесор за системен контрол (CP15)
*Копроцесор NEON за медийна обработка (CP11)
*Копроцесор VFP за вектори с плаваща запетая (CP10)
*Копроцесор за дебъгване (CP14)
*Копроцесор за Java байт-кодове

† От началото на съществуване на ARM до днес има 16 копроцесора, номерирани с CP0 - CP15

Инструкции за работа с копроцесори

Копроцесорите може да са **вътрешни** за процесора, т.е. да са вградени в микропроцесора.

ИЛИ

Копроцесорите може да са **външни** за процесора – тогава се използва **копроцесорен интерфейс** за достъп.

Инструкции за работа с копроцесори

Копроцесорите виждат същия поток от инструкции като основното ядро, но изпълняват само тези инструкции, за които са създадени [1].

Инструкции за един копроцесор могат да се изпълняват само от него, а не от други копроцесори.

Ако в системата няма копроцесор, който да изпълни дадена инструкция, тя ще генерира **Undefined Instruction изключение**.

Инструкции за работа с копроцесори

ARM процесорите имат три вида инструкции за **комуникация** с копроцесори [2]:

*инструкции за стартиране на **обработка** на данни от копроцесор;
*инструкции за **копиране (с преобразуване)** на данни от и в регистри на копроцесор;
*инструкции за **генериране на адреси** за ползване от копроцесорните Load & Store инструкции.

Инструкции за работа с копроцесори

Списък с копроцесорни инструкции:
1. CDP (CDP2) – извърши операция с Copr
2. MCR (MCR2) – копирай в Copr от ARM
3. MCRR† (MCRR2) – копирай в Copr от 2 ARM рег.
4. MRC (MRC2) - копирай в ARM от Copr
5. MRRC† (MRRC2) - копирай в 2 ARM рег. от Copr
6. LDC – копирай от памет в Copr
7. STC – копирай от Copr в памет

† ARMv5TE и нагоре

Инструкции за работа с копроцесори

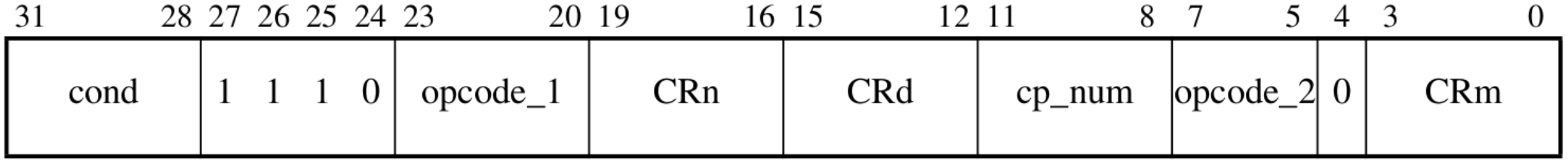
Забележка: на следващите слайдове, когато има два регистъра за източници и един за приемник се визира операция, която **комбинира и преобразува типове данни** на основното ядро и на копроцесора.

Пример:

120(int) + 530(int) = 650.00 (float)

Инструкции за работа с копроцесори

CDP



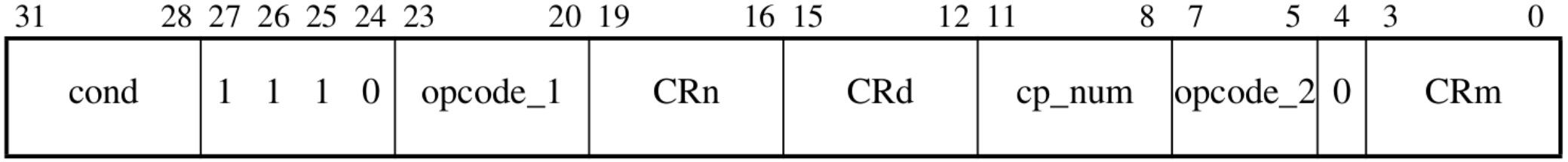
CDP (Coprocessor Data Processing) – стартира операция на копроцесор с номер cp_num, чийто регистри са различни от регистрите на основното ядро.

cond – условие (от статус регистъра), при което инструкцията ще се изпълни. Ако потребителя не е указал, използва се AL (execute Always);

opcode_1, opcode_2 – код на операцията за съответния копроцесор

Инструкции за работа с копроцесори

CDP



CRd – указва **регистър-приемник** на резултат от копроцесорното ядро;
CRn – указва **регистър-източник** за първи операнд от копроцесорното ядро;
CRm – указва **регистър-източник** за втори операнд от копроцесорното ядро;

Инструкции за работа с копроцесори

CDP2 – аналогична на CDP, но cond = 0b1111.

Table A3-1 Condition codes

Opcode [31:28]	Mnemonic extension	Meaning	Condition flag state
0000	EQ	Equal	Z set
0001	NE	Not equal	Z clear
0010	CS/HS	Carry set/unsigned higher or same	C set
0011	CC/LO	Carry clear/unsigned lower	C clear
0100	MI	Minus/negative	N set
0101	PL	Plus/positive or zero	N clear
0110	VS	Overflow	V set
0111	VC	No overflow	V clear
1000	HI	Unsigned higher	C set and Z clear
1001	LS	Unsigned lower or same	C clear or Z set
1010	GE	Signed greater than or equal	N set and V set, or N clear and V clear (N == V)
1011	LT	Signed less than	N set and V clear, or N clear and V set (N != V)
1100	GT	Signed greater than	Z clear, and either N set and V set, or N clear and V clear (Z == 0,N == V)
1101	LE	Signed less than or equal	Z set, or N set and V clear, or N clear and V set (Z == 1 or N != V)
1110	AL	Always (unconditional)	-
1111	-	See *Condition code 0b1111*	-

Инструкции за работа с копроцесори

Понеже cond = 0b1111 не е валидно условие, това позволява да се добавят още инструкции (които започват с 0b1111), които се различават от вече-съществуващите.

Те могат да се използват за бъдещо развитие за дадения копроцесор.

Тези инструкции, обаче, няма да могат да се изпълняват условно.

Инструкции за работа с копроцесори

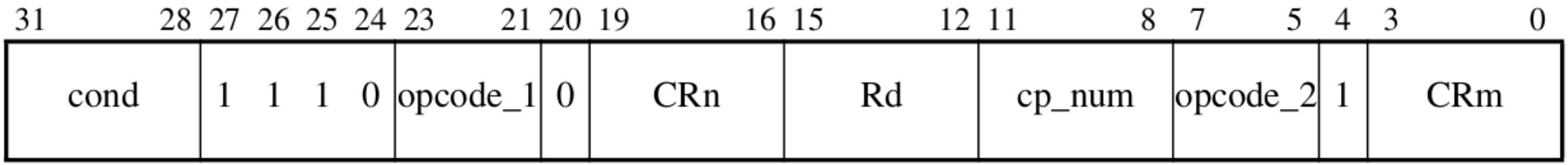
Пример:

CDP p**5**, 2, c12, c10, c3, 4

Инструкцията може да се преведе като:
*инструкция на копроцесор #**5** за обработка на данни
*opcode_1 = 2
*opcode_2 = 4
*регистър-приемник на резултат е регистър #12
*регистри-източници са #10 и #3

Инструкции за работа с копроцесори

MCR



MCR (Move to Coprocessor from ARM Register) – предава стойност на регистър от основното ядро (Rn) на копроцесор с номер cp_num. (**MCR2**, cond=0b1111)

Rd – указва **регистър-източник**, чиято стойност ще се използва като операнд от основното процесорно ядро;
CRn – указва **регистър-приемник** на резултат от копроцесорното ядро;
CRm – указва допълнителен **регистър-източник** или **регистър-приемник** от копроцесорното ядро;

Инструкции за работа с копроцесори

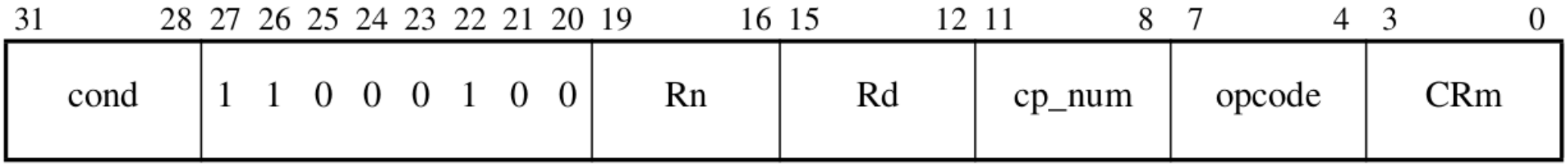
Пример:

MCR p**14**, 1, R7, c7, c12, 6

Инструкцията може да се преведе като:
*инструкция на копроцесор #**14** за прехвърляне на данни от основното ядро към копроцесора
*opcode_1 = 1
*opcode_2 = 6
*регистър-източник R7 от основното ядро
*регистри-приемници на резултат са регистри #7, #12

Инструкции за работа с копроцесори

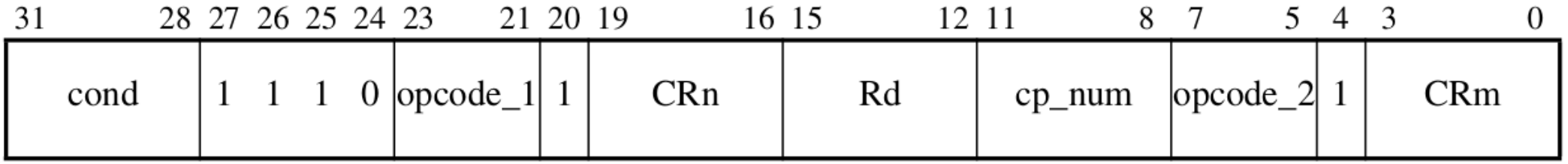
MCRR



MCRR (Move to Coprocessor from two ARM Registers) – предава стойност на два регистъра от основното ядро (Rn, Rd) на копроцесор с номер cp_num. (**MCRR2**, cond=0b1111)

Rd – указва **регистър-източник** за първи операнд от основното процесорно ядро;
Rn – указва **регистър-източник** за втори операнд от основното процесорно ядро;
CRm – указва **регистър-приемник** от копроцесорното ядро;

MRC



MRC (Move to ARM Register from Coprocessor) – предава стойност на регистър от копроцесор с номер cp_num към регистър от основното ядро (Rd) или опреснява битовете за условие от STATUS регистъра. (**MRC2**, cond=0b1111)

Rd – указва **регистър-приемник** от основното процесорно ядро, ако се избере R15 – опр. се статус флаговете;

CRn – указва **регистър-източник** за първи операнд от от копроцесорното ядро;

CRm – указва допълнителен **регистър-приемник** или **регистър-източник** от копроцесорното ядро;

Инструкции за работа с копроцесори

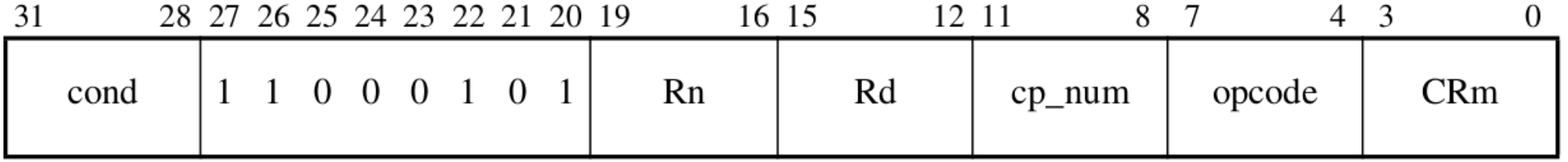
Пример:

MRC p**15**, 5, R4, c0, c2, 3

Инструкцията може да се преведе като:
*инструкция на копроцесор #**15** за прехвърляне на данни от копроцесора към основното ядро
*opcode_1 = 5
*opcode_2 = 3
*регистър-приемник R4 от основното ядро
*регистри-източници са регистри #0, #2 от копроцесора

Инструкции за работа с копроцесори

MRRC

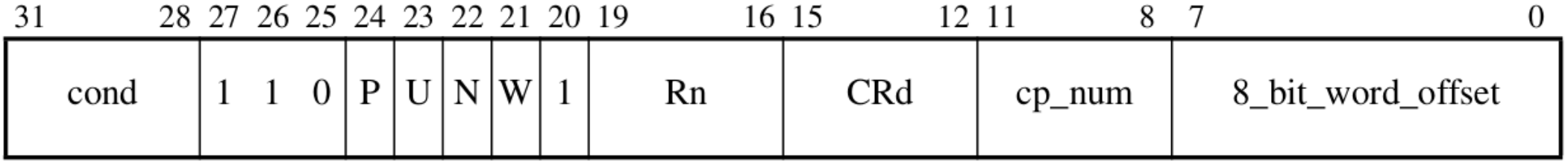


MRRC (Move two ARM Registers to Coprocessor Register) – предава стойност на два регистъра от основното ядро (Rn, Rd) на копроцесор с номер cp_num. (**MRRC2**, cond=0b1111)

Rd – указва **регистър-приемник** за първи операнд от основното процесорно ядро;
Rn – указва **регистър-приемник** за втори операнд от основното процесорно ядро;
CRm – указва **регистър-източник** от копроцесорното ядро;

Инструкции за работа с копроцесори

LDC

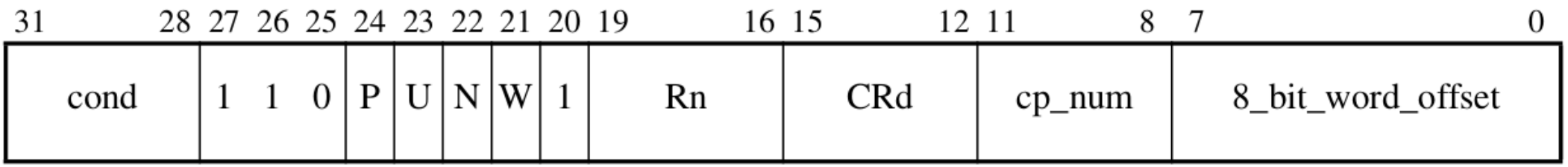


LDC (Load Coprocessor) – зарежда данни, разположени на последователни адреси, от паметта в регистър на копроцесор с номер cp_num.

Rn – указва регистър от основното процесорно ядро, който ще служи за **указател към адрес**;
8_bit_word_offset – указва отместване, което ще се добави към указателя на адреса;
CRd – указва **регистър-приемник** от копроцесорното ядро;

Инструкции за работа с копроцесори

LDC



P – вид на адресацията, 0 → пост-индексна, 1 → пред-индексна

U – знак на отместването, 0 → отрицателен, 1 → положителен

N – има различни функции при различните копроцесори

W – опресняване на указателя към адрес, 0 → остава непроменен, 1 → променя се със стойността на отместването

Бит 20 (отбелязва се също с L) – вид на копирането, 1 → от памет в копроцесор, 0 – от копроцесор в памет.

Инструкции за работа с копроцесори

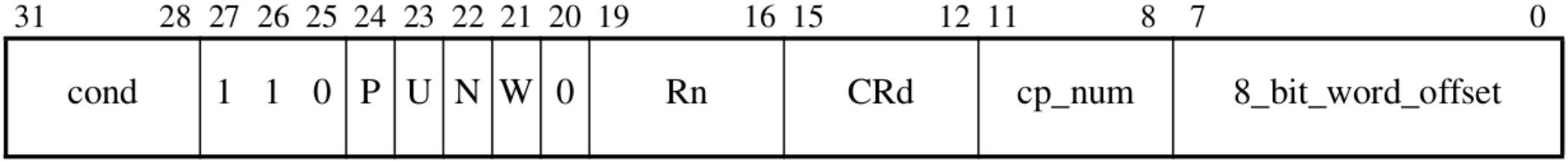
Пример:

LDC p**6**, CR1, [R4]

Инструкцията може да се преведе като:
*инструкция на копроцесор #**6** за копиране на данни от паметта в копроцесора;
*регистър R4 (от основното ядро), указващ базов адрес
*регистър-приемник #1 от копроцесорното ядро;

Инструкции за работа с копроцесори

STC



STC (Store Coprocessor) – записва данни от регистър на копроцесор с номер cp_num в паметта.

Rn – указва регистър от основното процесорно ядро, който ще служи за **указател към адрес**;
8_bit_word_offset – указва отместване, което ще се добави към указателя на адреса;
CRd – указва **регистър-източник** от копроцесорното ядро;

P, U, N, W, bit20 – аналогични на LDC

Инструкции за работа с копроцесори

Пример:

STC p**8**, CR7, [R2, #4]!

Инструкцията може да се преведе като:
*инструкция на копроцесор #**8** за копиране на данни от копроцесора в паметта;
*регистър R2 (от основното ядро), плюс 4, указва базовия адрес, на който ще се пише; след извършване на копирането R2 = R2 + 4
*регистър-източник #7 от копроцесорното ядро;

Приемник на стойността е паметта, указана от R2+4

Инструкции за работа с дебъг копроцесор

История на версиите на ARM debug interface (ADI)[3]:

*__ADI v1 & v2__ – предназначени за микропроцесорите ARM7TDMI, ARM9

*__ADI v3__ – преназначена за микропроцесора ARM10

*__ADI v4__ – първата версия, която се обвързва с набор от инструкции, а не с конкретно процесорно ядро. Наборът е ARMv6.

*__ADI v5__ – изцяло премахва зависимостта от процесорни ядра. Налага стандарт при даването на номера за версии на дебъг портовете.

*__ADI v6__ – позволява да се дебъгват всички части от една система през повече от един дебъг агент – например през външен хардуерен дебъгер и през вътрешен софтуерен дебъгер, изпълняван на самата система.

Инструкции за работа с дебъг копроцесор

Връзката с дебъгера може да бъде:

***JTAG/SWD** специализиран интерфейс

***PCIe, USB, Ethernet** стандартни интерфейси

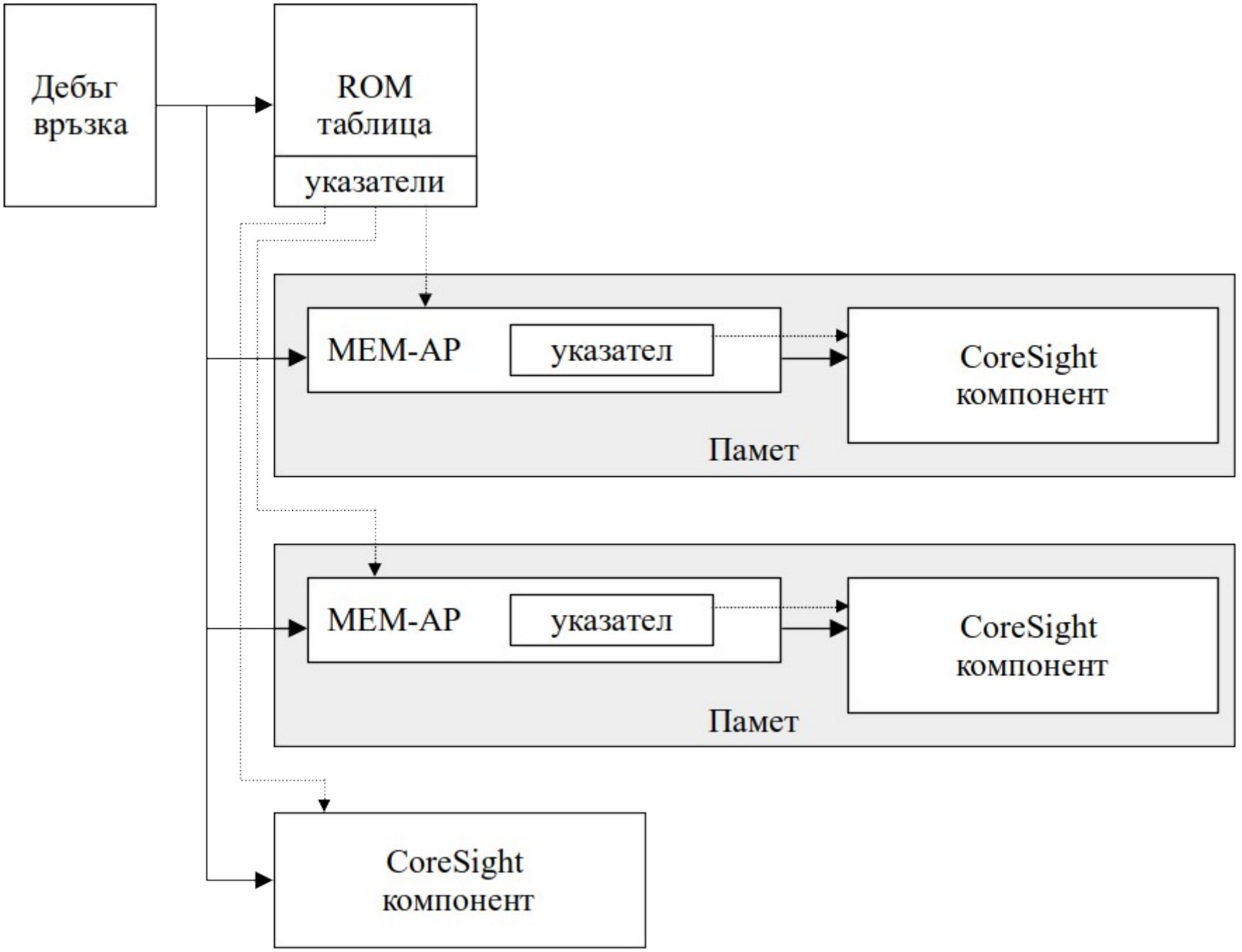
***Софтуерна**

Инструкции за работа с дебъг копроцесор

За да бъдат опознати елементите в системата и топологията ѝ, дебъгерът трябва да знае начален 64-битов адрес на първия елемент.

Първи елемент може да бъде:
***CoreSight дебъг** модул;
***AP** (Access Port) – преходник към друга система;
***MEM-AP** (Memory-Access Port) – преходник към памет;
***ROM таблица** – съдържа адреси на един или повече дебъг компоненти, които предстои да бъдат използвани;

Инструкции за работа с дебъг копроцесор



Инструкции за работа с дебъг копроцесор

AP (Access Port) – осигурява следната информация:
*дебъг регистрите на основния процесор
*регистрите на ETM
*ROM таблица
*памет
*класически JTAG модул

Инструкции за работа с дебъг копроцесор

DP (Debug Port) – осигурява общ интерфейс за достъп до информацията, предоставяна от AP. DP включва следните елементи:

***JTAG-DP** – IEEE1149.1-съвместим DBGTAP контролер;

***SW-DP** – Serial Wire използва два извода и пакетно-базиран протокол за четене и запис на регистри от системата;

***SWJ-DP** – Serial Wire/JTAG е комбинация от двата дебъг интерфейса и те споделят общи изводи, като дебъгерът избира кой от двата ще се използва в даден момент;

***DP регистри** – съдържат описание, с което може да се инициализира общия интерфейс за достъп до информация.

Инструкции за работа с дебъг копроцесор

Копроцесор #14 (CP14) е дебъг модула на ARM Cortex микропроцесорите.

На следващия слайд са показани някои от най-важните регистри на CP14. Това е **програмният модел** на този копроцесор.

Различните ARM Cortex-A процесори добавят също и други регистри, но те няма да бъдат разгледани тук.

Инструкции за работа с дебъг копроцесор

Table D3-1 CP14 debug register map

Binary address		Register number	Abbreviation [a]	CP14 debug register name
Opcode_2	CRm			
000	0000	0	DIDR	Debug ID Register
000	0001	1	DSCR	Debug Status and Control Register
000	0010-0100	2-4	-	RESERVED
000	0101	5	DTR	Data Transfer Register
000	0110	6	WFAR	Watchpoint Fault Address Register[b]
000	0111	7	VCR	Vector Catch Register
000	1000-1111	8-15	-	RESERVED
001-011	0000-1111	16-63	-	RESERVED
100	0000-1111	64-79	BVRy	Breakpoint Value Registers / RESERVED
101	0000-1111	80-95	BCRy	Breakpoint Control Registers / RESERVED
110	0000-1111	96-111	WVRy	Watchpoint Value Registers / RESERVED
111	0000-1111	112-127	WCRy	Watchpoint Control Registers / RESERVED

Инструкции за работа с дебъг копроцесор

За да се постави **точка на прекъсване (breakpoint)**, трябва да се достъпят два регистъра:
***BVR** (Breakpoint Value Register)
***BCR** (Breakpoint Control Register)

За да се постави **даннова точка на прекъсване (watchpoint)**, трябва да се достъпят два регистъра:
***WVR** (Watchpoint Value Register)
***WCR** (Watchpoint Control Register)

Инструкции за работа с дебъг копроцесор

Процесори, поддържащи набори ARMv6 и нагоре могат да **дебъгват на ниво нишка** от операционната система.

Това означава, че дебъг модула е **“thread-aware”**.

Механизъмът на задействане е описан на следващия слайд.

Инструкции за работа с дебъг копроцесор

*С BCR (WCR) се настройва thread-aware точка.
* След това в BVR (WVR) се записва ID на нишката, вместо адрес на инструкция (променлива). Тогава дебъг копроцесора ще следи за съвпадение с регистър Context ID (регистър #13) от CP15.
*Още една двойка BVR-BCR (WVR-WCR) се зарежда с виртуален адрес на инструкция (променлива) от дадената нишка.
***Дебъг събитие** се генерира, **само** когато има съвпадение на **Context ID и адрес** на инструкция (променлива).

Инструкции за работа с дебъг копроцесор

Структура на един BVR регистър.

Table D3-4 Breakpoint Value Registers bit definition

Bits	Read/write attributes	Reset value	Description
31:0	RW	- (UNPREDICTABLE)	Breakpoint Value

Инструкции за работа с дебъг копроцесор

Структура на един BCR регистър.

Table D3-5 Breakpoint Control Registers bit definition

Bits	Read/write attributes	Reset value	Description [a]
[0]	RW	0	Breakpoint enable. 0: disabled. 1: enabled.
[2:1]	RW	-	Supervisor Access, see *Supervisor Access, bits[2:1].*
[4:3]	UNP/SBZP	-	RESERVED
[8:5]	RW	-	Byte address select, see *Byte address select, bits[8:5]* on page D3-18.
[15:9]	UNP/SBZP	-	RESERVED
[19:16]	RW	-	Linked BRP number, see *Linked BRP number, bits[19:16]* on page D3-19.
[20]	RW	-	Enable linking, see *Enable linking, bit[20]* on page D3-19.
[22:21]	RW[b]	- (-)	Meaning of BVR, *Meaning of BVR, bits[22:21]* on page D3-19.
[31:23]	UNP/SBZP	-	RESERVED

Инструкции за работа с дебъг копроцесор

BCR[0] – разрешаване (1) или забраняване (0) на точката на прекъсване.

BCR[2:1] – точката на прекъсване може да сработи само в:
*00 – не се използва
*01 – привилегирован режим (privileged)
*10 – потребителски режим (user)
*11 – както в привилегирован, така и в потребителски (either)

Инструкции за работа с дебъг копроцесор

BCR[4:3] – не се използват

BCR[8:5] – разрешава генерирането на събитие от точката на прекъсване, ако процесорът се опита да изпълни инструкция от байт:

*0000 – точката никога няма да сработи

*0001 – точката ще сработи, ако процесорът се опита да изпълни инструкция от адрес (BVR & 0xFFFF.FFFC) + 0

*0010 – (BVR & 0xFFFF.FFFC) + 1

*0100 – (BVR & 0xFFFF.FFFC) + 2

*1000 – (BVR & 0xFFFF.FFFC) + 3

*1111 – всички байтове от адрес (BVR & 0xFFFF.FFFC)

Инструкции за работа с дебъг копроцесор

BCR[15:9] – не се използват

BCR[19:16] – номер на двойка BRP (Breakpoint Register Pair) регистри, с която да бъде свързана настоящата двойка регистри (двойка регистри → BVR + BCR)

BCR[20] – разреши свързването на настоящата двойка регистри с други двойки BRP:
*0 – забранено
*1 - разрешено

Инструкции за работа с дебъг копроцесор

BCR[22:21] – генериране на дебъг събитие при съвпадение на:
*00 – виртуален адрес на инструкцията, BVR[31:2] + BCR[8:5] съвпаднат с адреса на IVA магистралата;
*01 – Context ID, BVR се сравнява с регистър 13 от CP15
*10 - виртуален адрес на инструкцията, BVR[31:2] + BCR[8:5] само когато **не са равни** с адреса на IVA магистралата
*11 – не се използва

BCR[31:23] – не се използват

Инструкции за работа с дебъг копроцесор

За да е съвместим един дебъг модул с ADIv6 спецификацията, той трябва да има **минимум**:
*2 x BRP
*1 x WRP
*поне 1 x BRP да има възможност за Context ID сравнение

Инструкции за работа с дебъг копроцесор

Списък с инструкциите, поддържани от всички CP14 копроцесори на ARM Cortex-A са показани на следващия слайд [2].

Инструкции за работа с дебъг копроцесор

Table D3-1 Legal CP14 debug instructions

Binary address		Register number	Abbreviation	Legal instructions [a]
Opcode_2	CRm			
000	0000	0	DIDR	`MRC p14,0,Rd,c0,c0,0`

000	0001	1	DSCR	`MRC p14,0,Rd,c0,c1,0`
				`MRC p14,0,R15,c0,c1,0`
				`MCR p14,0,Rd,c0,c1,0`

000	0101	5	DTR	`MRC p14,0,Rd,c0,c5,0`
				`MCR p14,0,Rd,c0,c5,0`
				`STC p14,c5,<addressing_mode>`
				`LDC p14,c5,<addressing_mode>`

000	0110	6	WFAR	`MRC p14,0,Rd,c0,c6,0` `MCR p14,0,Rd,c0,c6,0`
000	0111	7	VCR	`MRC p14,0,Rd,c0,c7,0` `MCR p14,0,Rd,c0,c7,0`
100	0000-1111	64-79	BVR	`MRC p14,0,Rd,c0,CRm,4` `MCR p14,0,Rd,c0,CRm,4`
101	0000-1111	80-95	BCR	`MRC p14,0,Rd,c0,CRm,5` `MCR p14,0,Rd,c0,CRm,5`
110	0000-1111	96-111	WVR	`MRC p14,0,Rd,c0,CRm,6` `MCR p14,0,Rd,c0,CRm,6`
111	0000-1111	112-127	WCR	`MRC p14,0,Rd,c0,CRm,7` `MCR p14,0,Rd,c0,CRm,7`

Литература

[1] “ARM Architecture Reference Manual ARMv7-A and ARMv7-R edition”, ARM DDI 0406C ID112311, 2011.
[2] “ARM Architecture Reference Manual”, ARM DDI 0100l, ARM Ltd, 2005.
[3] “Arm Debug Interface Architecture Specification”, ADIv6.0, ARM IHI 0074D (ID030122), 2022

Външни връзки

[a]