ISA Assembler Format (4C)

Coprocessor Instructions

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using LibreOffice.

Young Won Lim
12/14/19

Based on

ARM System-on-Chip Architecture, 2nd ed, Steve Furber

Coprocessor Instruction

Coprocessor Data Transfers

LDC | STC {<cond>} {L} <CP#>, CRd, [Rn, <offset>] {!}

LDC | STC {<cond>} {L} <CP#>, CRd, [Rn], <offset>

Coprocessor Data Operations

CDP {<cond>} <CP#>, <Cop1>, CRd, CRn, CRm {, <Cop2>}

Coprocessor Register Transfers

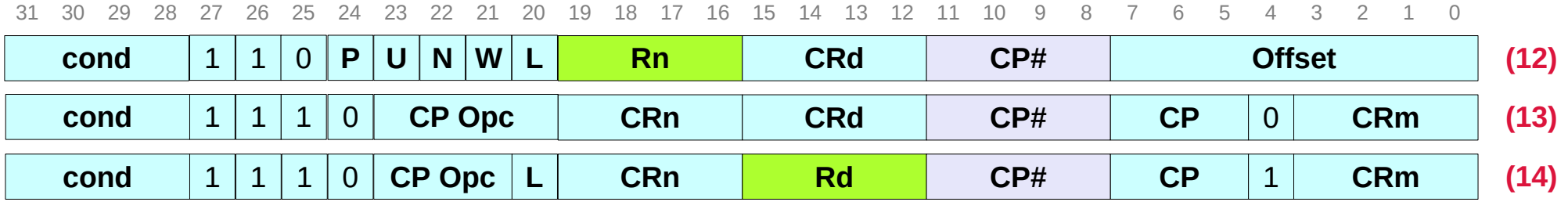
MRC {<cond>} <CP#>, <Cop1>, Rd, CRn, CRm {, <Cop2>}

MCR {<cond>} <CP#>, <Cop1>, Rd, CRn, CRm {, <Cop2>}

Coprocessor Instruction Mnemonics

L D C Load coprocessor	Load data from the memory into a coprocessor register	**CRd, [Rn, <offset>] {!}** **CRd, [Rn], <offset>**
S T C Store coprocessor	Store data from a coprocessor register to the memory	
C D P coprocessor Data Processing	Perform an operation over CRd and CRm and place the result in CRd	**CRd, CRn, CRm**
M R ⬅ C Move Reg coprocessor	**command** to **get** some data from a coprocessor	**Rd, CRn, CRm**
M C ⬅ R Move coprocessor Reg	**command** to **pass** some data to a coprocessor	

Coprocessor Instruction Encodings



Coprocessor Data Transfers

LDC | STC {<cond>} {L} <**CP#**>, CRd, [Rn, <offset>] {!}

LDC | STC {<cond>} {L} <**CP#**>, CRd, [Rn], <offset>

Coprocessor Data Operations

CDP {<cond>} <**CP#**>, <Cop1>, CRd, CRn, CRm {, <Cop2>}

Coprocessor Register Transfers

MRC {<cond>} <**CP#**>, <Cop1>, Rd, CRn, CRm {, <Cop2>}

MCR {<cond>} <**CP#**>, <Cop1>, Rd, CRn, CRm {, <Cop2>}

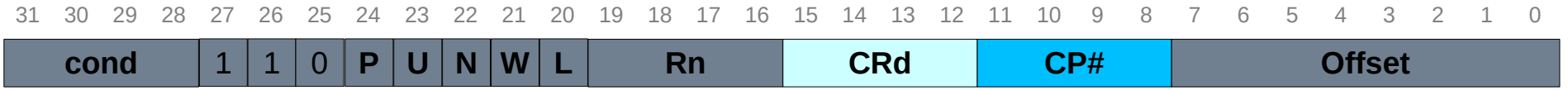
ARM Coprocessor Instruction Fields

<CP#>	Coprocessor number	
<Cop1>	Coprocessor operation 1	**<CP Opc>**
<Cop2>	Coprocessor operation 2	**<CP>**
CRd	Coprocessor Rd	
CRn	Coprocessor Rn	
CRm	Coprocessor Rm	

CP# : identifes a coprocessor (a number in [0, 15])
a coprocessor will ignore any instruction
that has an incorrect **CP#**

CP Opc (**Cop1**) and possible the **CP** (**Cop2**):
specified what operation the coprocessor should perform
on the contents of **CRn** and **CRm**, and place the result in **CRd**.

Coprocessor Access Fields



(12)

Coprocessor Data Transfer

CRd :=: [Rn,Offset]; CRd :=: [Rn], Offset

(13)

Coprocessor Data Operation

CRd :=: CRn <CP Opc, CP> CRm

(14)

Coprocessor Register Transfer

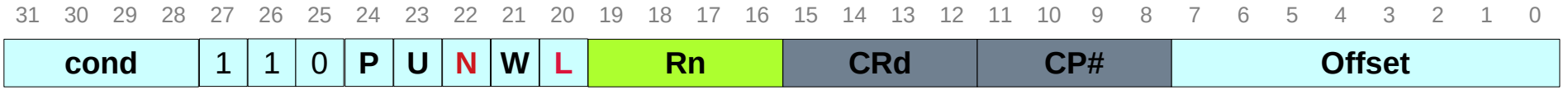
Rd :=: CRn <CP Opc, CP> CRm

P	Pre/Post Index
U	Up/Down
N	Transfer Length
W	Write-back (auto-index)
L	Load/Store

N (= **L** flag in assembler)

CP#	4-bit	Coprocessor number
COpc	4/3-bit	Coprocessor OP Code
CP	3-bit	Coprocessor Information
CRd	4-bit	Coprocessor Rd
CRn	4-bit	Coprocessor Rn
CRm	4-bit	Coprocessor Rm
Rn	4-bit	Base Reg
Rd	4-bit	Destination Reg

ARM Access Fields



(12)

Coprocessor Data Transfer

CRd :=: [Rn,Offset]; CRd :=: [Rn], Offset

(13)

Coprocessor Data Operation

CRd :=: CRn <CP Opc, CP> CRm

(14)

Coprocessor Register Transfer

Rd :=: CRn <CP Opc, CP> CRm

L=0 Store to memory **STC MCR**
L=1 Load from memory **LDC MRC**
N=0 Short Transfer
N=1 Long Transfer with **L** flag in assembler

Coprocessor Data Transfers

Preindex Coprocessor Data Transfer

LDC | STC {<cond>} {**L**} <CP#>, CRd, [Rn, <offset>] {!}

Postindex Coprocessor Data Transfer

LDC | STC {<cond>} {**L**} <CP#>, CRd, [Rn], <offset>

CP# : identifes a coprocessor (a number in [0, 15])

a coprocessor will ignore any instruction
that has an incorrect CP#

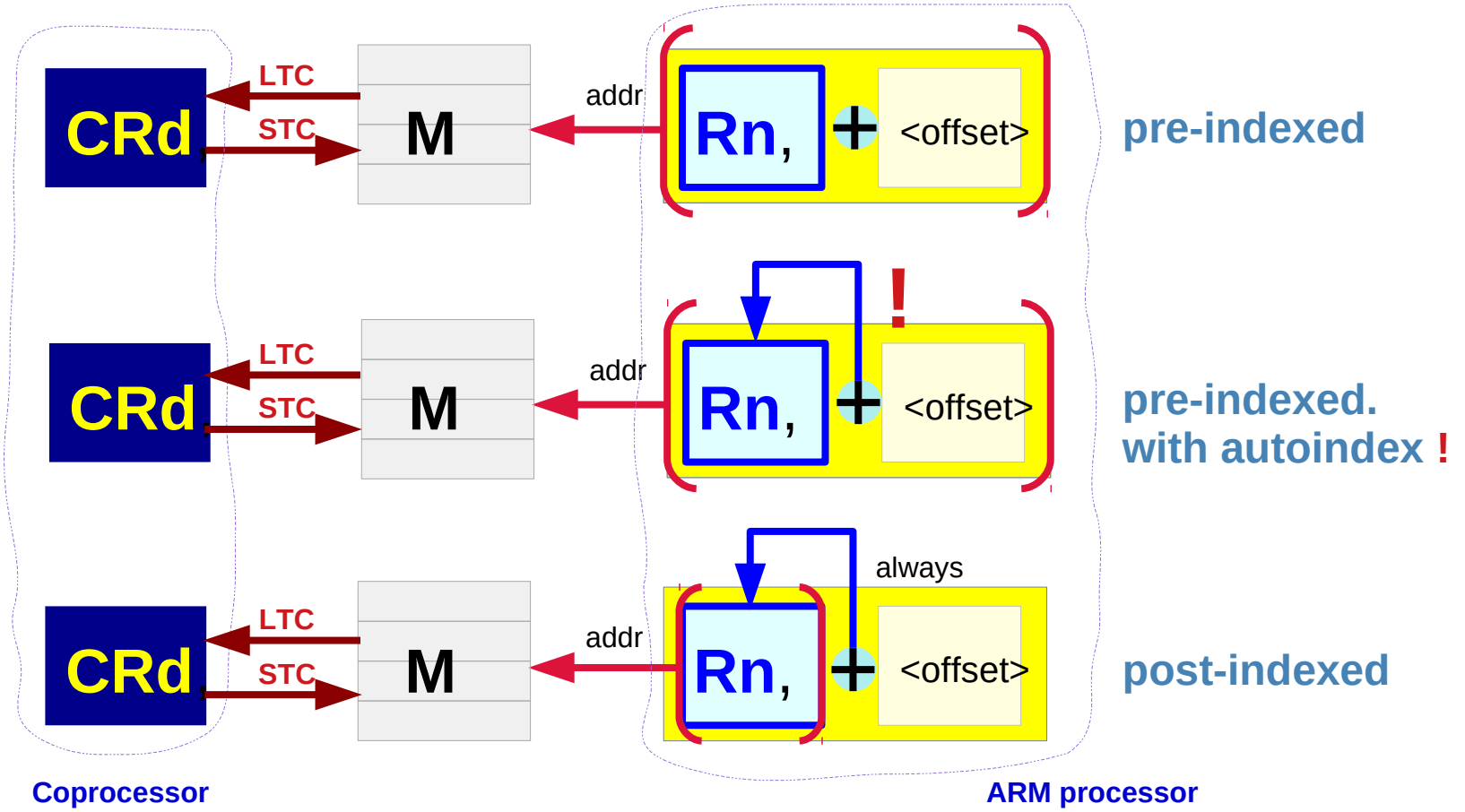
The **CRd** field and the **N** bit field in the encoding
contain information which may be interpreted in different ways
by different coprocessors,

CRd is the register to be transferred (the first register)
N bit selects transfer length options.

N=0 the transfer of a single register :: when **L** is present (long transfer)
N=1 the transfer of all the registers for context switching. :: otherwise

Pre-indexing and Post-indexing Coprocessor Instructions

CRd, [Rn, <offset>]	pre-indexed
CRd, [Rn, <offset>] !	pre-indexed with autoindex
CRd, [Rn], <offset>	post-indexed

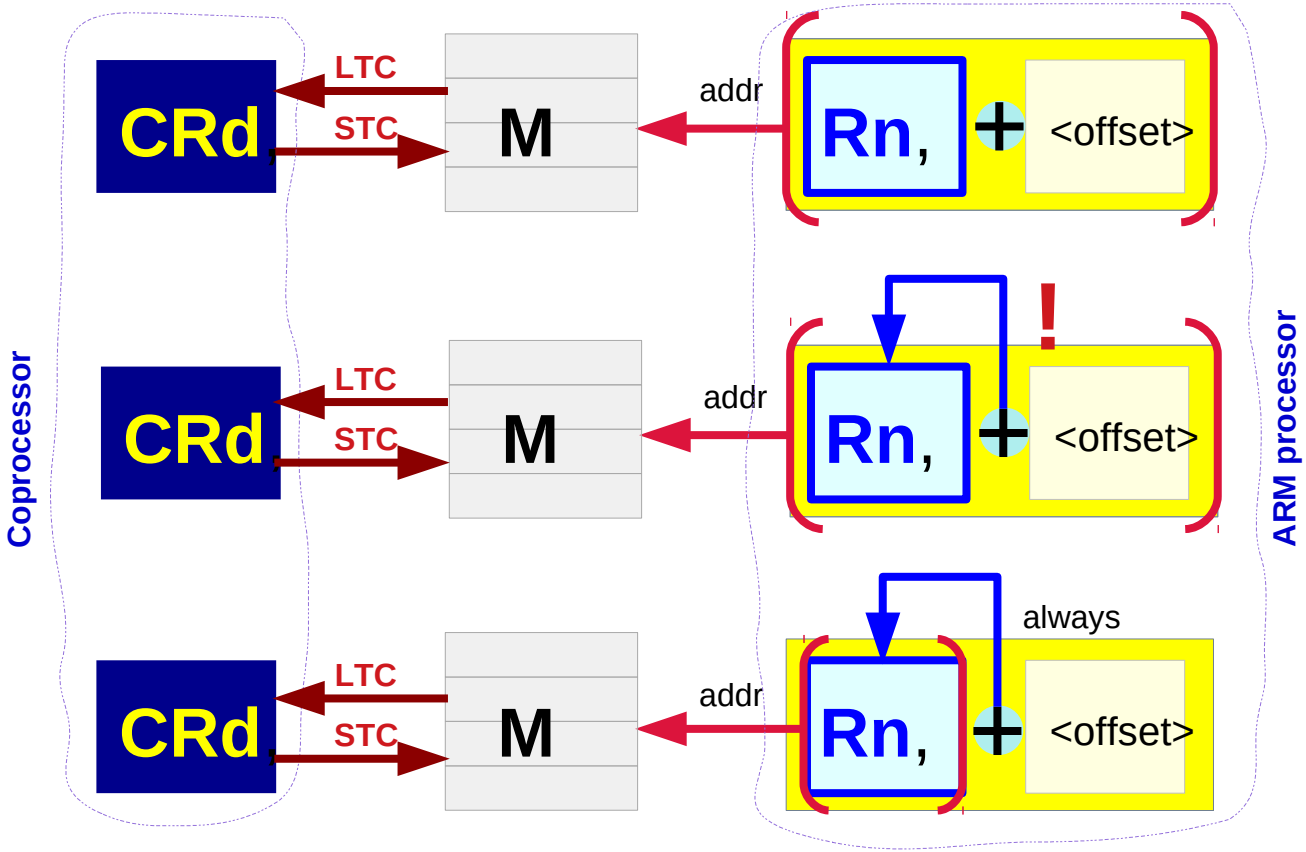


Pre-indexing and Post-indexing Coprocessor Instructions

Coprocessor Data Transfers

LDC | STC {<cond>} {L} <CP#>, CRd, [Rn, <offset>] {!}

LDC | STC {<cond>} {L} <CP#>, CRd, [Rn], <offset>



Pre-indexed
CRd, **[Rn**, <offset>**]**

pre-indexed.
with autoindex !
CRd, **[Rn**, <offset>**] !**

Post-indexed
CRd, **[Rn]**, <offset>

Coprocessor Data Transfers – Preindex

Preindex Coprocessor Data Transfer

LDC | STC {<cond>} {L} <CP#>, CRd, [Rn, <offset>] {!}

CP# : Coprocessor Number

```
LDC | STC              <CP#>, CRd, [Rn, <offset>]
LDC | STC <cond>       <CP#>, CRd, [Rn, <offset>]
LDC | STC              <CP#>, CRd, [Rn, <offset>] !
LDC | STC <cond>       <CP#>, CRd, [Rn, <offset>] !

LDC | STC          L   <CP#>, CRd, [Rn, <offset>]
LDC | STC <cond>   L   <CP#>, CRd, [Rn, <offset>]
LDC | STC          L   <CP#>, CRd, [Rn, <offset>] !
LDC | STC <cond>   L   <CP#>, CRd, [Rn, <offset>] !
```

Coprocessor Data Transfers – Postindex

Postindex Coprocessor Data Transfer

LDC | STC {<cond>} {L} <CP#>, CRd, [Rn], <offset>

CP# : Coprocessor Number

```
LDC | STC                <CP#>, CRd, [Rn], <offset>
LDC | STC <cond>         <CP#>, CRd, [Rn], <offset>
LDC | STC           L    <CP#>, CRd, [Rn], <offset>
LDC | STC <cond>    L    <CP#>, CRd, [Rn], <offset>
LDC | STC                <CP#>, CRd, [Rn], <offset> !
LDC | STC <cond>         <CP#>, CRd, [Rn], <offset> !
LDC | STC           L    <CP#>, CRd, [Rn], <offset> !
LDC | STC <cond>    L    <CP#>, CRd, [Rn], <offset> !
```

Coprocessor Data Operations

Coprocessor Data Processing Operations

CDP {<cond>} <CP#>, <Cop1>, CRd, CRn, CRm

CDP {<cond>} <CP#>, <Cop1>, CRd, CRn, CRm, <Cop2>

CP# : identifes a coprocessor (a number in [0, 15])

a coprocessor will ignore any instruction
that has an incorrect CP#

CP Opc (**Cop1**) and possible the **CP** (**Cop2**):

specified what operation the coprocessor should perform
on the contents of **CRn** and **CRm**, and place the result in **CRd**.

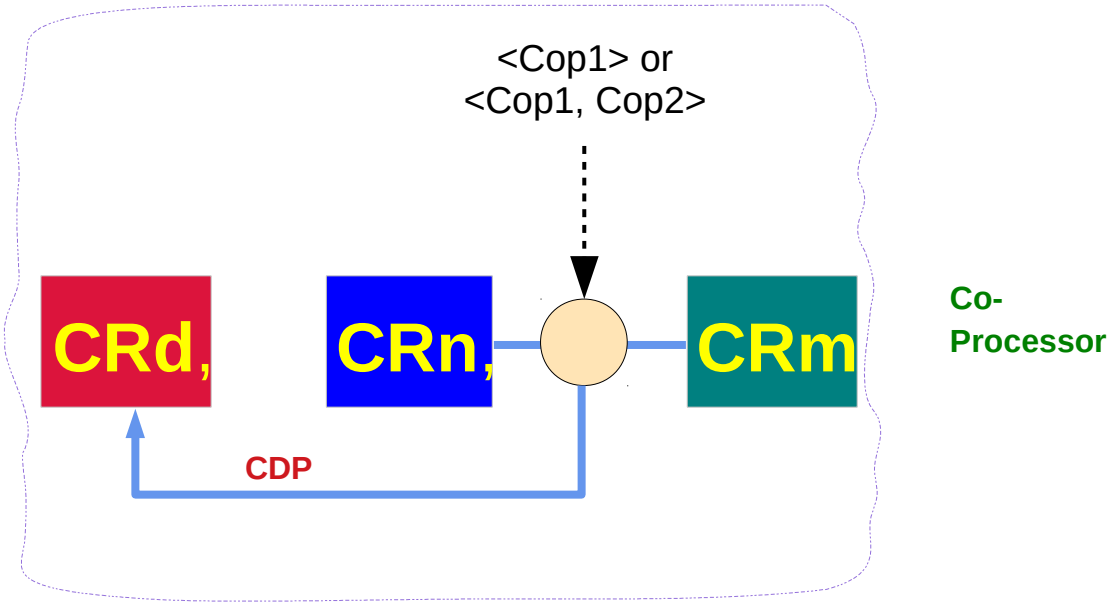
<expression1> evaluated to a constant and placed in the CP Opc field
<expression2> where present is evaluated to a constant and placed in the CP field

Coprocessor Data Operations

Coprocessor Data Processing Operations

CDP {<cond>} <CP#>, <Cop1>, CRd, CRn, CRm

CDP {<cond>} <CP#>, <Cop1>, CRd, CRn, CRm, <Cop2>



Coprocessor Data Operations

Coprocessor Data Processing Operations

CDP {<cond>} <CP#>, <Cop1>, CRd, CRn, CRm

CDP {<cond>} <CP#>, <Cop1>, CRd, CRn, CRm, <Cop2>

CP# : Coprocessor Number

CDP	<CP#>, <Cop1>, CRd, CRn, CRm
CDP	<CP#>, <Cop1>, CRd, CRn, Crm, <Cop2>
CDP <cond>	<CP#>, <Cop1>, CRd, CRn, CRm
CDP <cond>	<CP#>, <Cop1>, CRd, CRn, CRm, <Cop2>

Coprocessor Register Transfers (R←C, C←R)

Move to ARM Register from Coprocessor

MRC {<cond>} <CP#>, <Cop1>, Rd, CRn, CRm {, <Cop2>}

Move to Coprocessor from ARM Registers

MCR {<cond>} <CP#>, <Cop1>, Rd, CRn, CRm {, <Cop2>}

CP# : identifes a coprocessor (a number in [0, 15])
a coprocessor will ignore any instruction
that has an incorrect CP#

CP Opc (**Cop1**) and possible the **CP** (**Cop2**):
specified what operation the coprocessor should perform
on the contents of **CRn** and **CRm**, and place the result in **CRd**.

CRn is the coprocessor src / dst register of the transformation

CRm is a 2nd coprocessor register involved in some way
which depends on the particular operation specified.

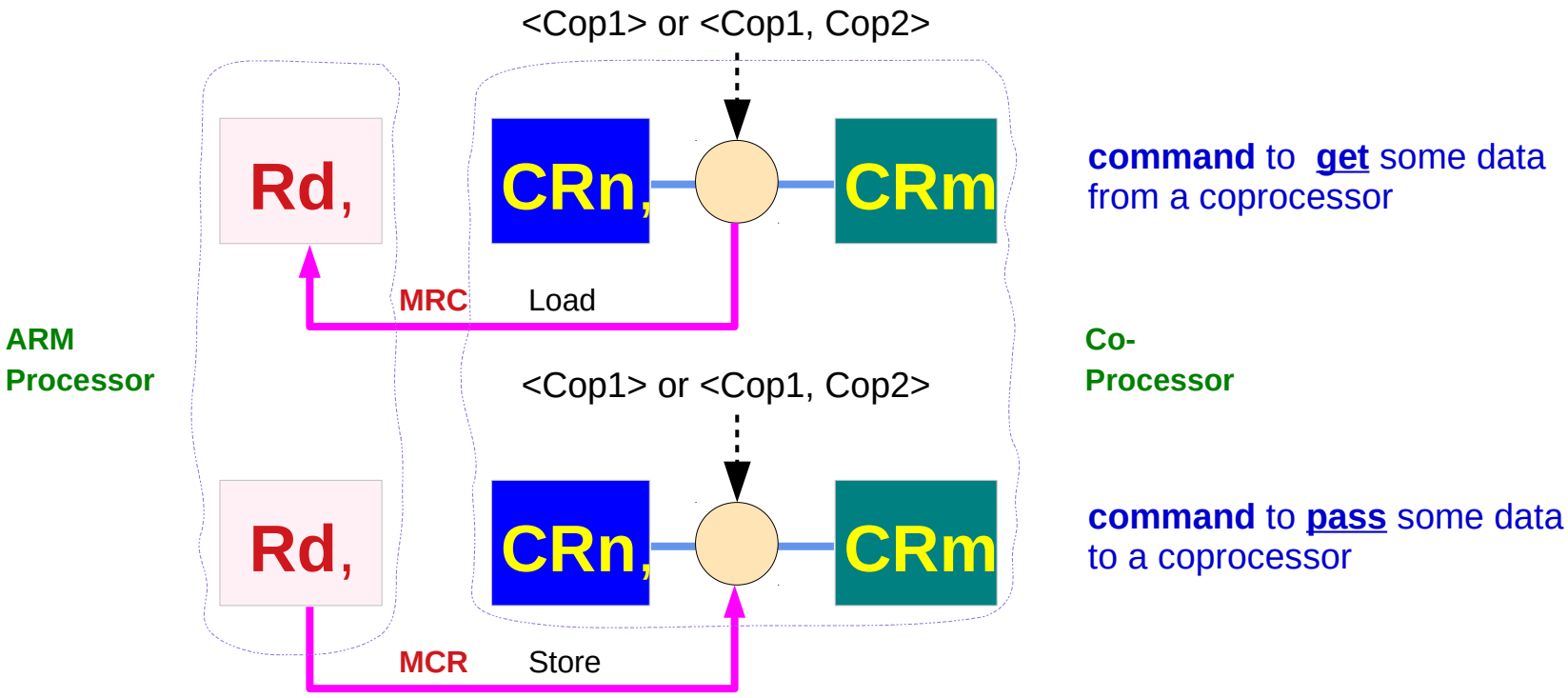
Coprocessor Register Transfers (R←C, C←R)

Move to ARM Register from Coprocessor

MRC {<cond>} <CP#>, <Cop1>, Rd, CRn, CRm {, <Cop2>}

Move to Coprocessor from ARM Registers

MCR {<cond>} <CP#>, <Cop1>, Rd, CRn, CRm {, <Cop2>}



command to **get** some data from a coprocessor

command to **pass** some data to a coprocessor

Coprocessor Register Transfers (R←C)

Move to ARM Register from Coprocessor

MRC {<cond>} <CP#>, <Cop1>, Rd, CRn, CRm

MRC {<cond>} <CP#>, <Cop1>, Rd, CRn, CRm, <Cop2>

CP# : Coprocessor Number

MRC	<CP#>, <Cop1>, Rd, CRn, CRm
MRC	<CP#>, <Cop1>, Rd, CRn, CRm, <Cop2>
MRC <cond>	<CP#>, <Cop1>, Rd, CRn, CRm
MRC <cond>	<CP#>, <Cop1>, Rd, CRn, CRm, <Cop2>

Coprocessor Register Transfers (C←R)

Move to Coprocessor from ARM Registers

MCR {<cond>} <CP#>, <Cop1>, Rd, CRn, CRm

MCR {<cond>} <CP#>, <Cop1>, Rd, CRn, CRm, <Cop2>

CP# : Coprocessor Number

```
MCR       <CP#>, <Cop1>, Rd, CRn, CRm
MCR       <CP#>, <Cop1>, Rd, CRn, CRm, <Cop2>
MCR <cond> <CP#>, <Cop1>, Rd, CRn, CRm
MCR <cond> <CP#>, <Cop1>, Rd, CRn, CRm, <Cop2>
```

Breakpoint Instruction

Breakpoint Instruction (BKPT)

BRK

References

[1] ftp://ftp.geoinfo.tuwien.ac.at/navratil/HaskellTutorial.pdf
[2] https://www.umiacs.umd.edu/~hal/docs/daume02yaht.pdf