

# Instruction Set Architecture Overview (1A)

---

Copyright (c) 2014 - 2018 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Please send corrections (or suggestions) to [youngwlim@hotmail.com](mailto:youngwlim@hotmail.com).

This document was produced by using LibreOffice.

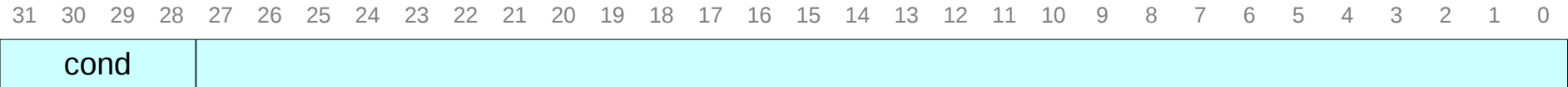
# Based on

---

ARM System-on-Chip Architecture, 2<sup>nd</sup> ed, Steve Furber

Rn	First operand register, Base Register
Rd	Source/Destination Register
S	Set Condition Codes
P	Pre/Post Index
U	Up/Down
B	Unsigned Byte/Word
W	Write-back (auto-index)
L	Load/Store

# Condition Code

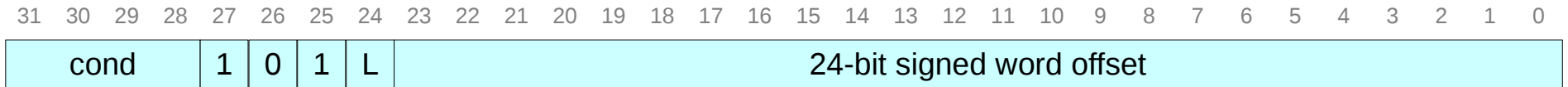


# ARM Condition Codes

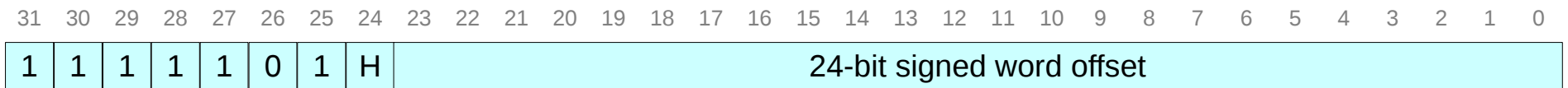
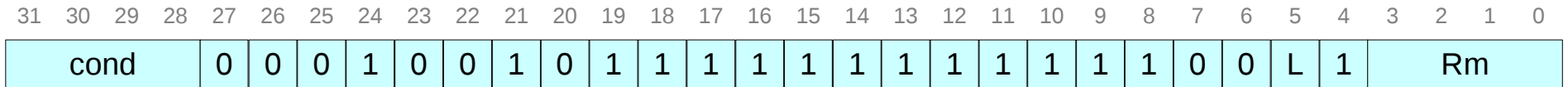
31 30 29 28

0	0	0	0	EQ	Equal / Equals zero	$Z \leftarrow 1$
0	0	0	1	NE	Not Equal	$Z \leftarrow 0$
0	0	1	0	CS/HS	Carry Set / unsigned High or Same	$C \leftarrow 1$
0	0	1	1	CC/LO	Carry Clear / unsigned Lower	$C \leftarrow 0$
0	1	0	0	MI	MInus / negative	$N \leftarrow 1$
0	1	0	1	PL	PLus / positive or zero	$N \leftarrow 0$
0	1	1	0	VS	oVerflow Set	$V \leftarrow 1$
0	1	1	1	VC	oVerflow Clear	$V \leftarrow 0$
1	0	0	0	HI	unsigned Higher	$C \leftarrow 1, Z \leftarrow 0$
1	0	0	1	LS	unsigned Lower or Same	$C \leftarrow 0, Z \leftarrow 1$
1	0	1	0	GE	signed Greater than or Equal	$N == V$
1	0	1	1	LT	signed Less Than	$N != V$
1	1	0	0	GT	signed Greater Than	$Z \leftarrow 0, N == V$
1	1	0	1	LE	signed Less than or Equal	$Z \leftarrow 1, N != V$
1	1	1	0	AL	ALways	any
1	1	1	1	NV	NeVer (do not use?)	none

# Branch and Branch with Link (B, BL)

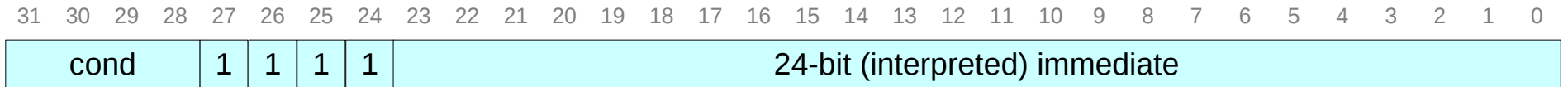


# Branch, Branch with Link and eXchange (BX, BLX)

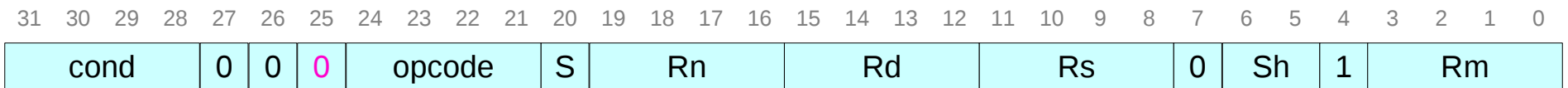
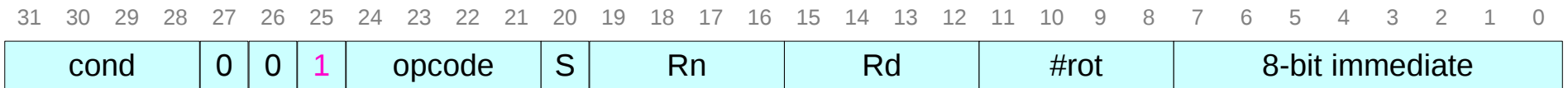




# SWI (Software Interrupt)



# Data Processing Instructions



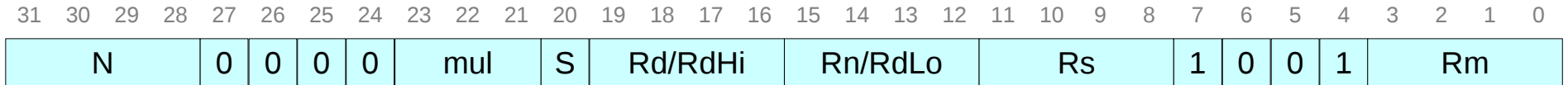
# Data Processing Instructions

31 30 29 28 27 26 25 24 23 22 21 20

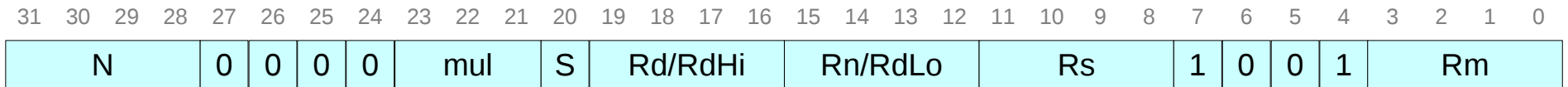
cond	0	0	#	0000	S
cond	0	0	#	0001	S
cond	0	0	#	0010	S
cond	0	0	#	0011	S
cond	0	0	#	0100	S
cond	0	0	#	0101	S
cond	0	0	#	0110	S
cond	0	0	#	0111	S
cond	0	0	#	1000	S
cond	0	0	#	1001	S
cond	0	0	#	1010	S
cond	0	0	#	1011	S
cond	0	0	#	1100	S
cond	0	0	#	1101	S
cond	0	0	#	1110	S
cond	0	0	#	1111	S

<b>AND</b>	Logical bit-wise AND	$Rd := Rn \text{ AND } Op2$
<b>EOR</b>	Logical bit-wise XOR	$Rd := Rn \text{ EOR } Op2$
<b>SUB</b>	Subtract	$Rd := Rn - Op2$
<b>RSB</b>	Reverse Subtract	$Rd := Op2 - Rn$
<b>ADD</b>	Add	$Rd := Rn + Op2$
<b>ADC</b>	Add with carry	$Rd := Rn + Op2 + C$
<b>SBC</b>	Subtract with carry	$Rd := Rn - Op2 + C - 1$
<b>RSC</b>	Reverse subtract with carry	$Rd := Op2 - Rn + C - 1$
<b>TST</b>	Test	$Rn \text{ AND } Op2$
<b>TEQ</b>	Test equivalence	$Rn \text{ EOR } Op2$
<b>CMP</b>	Compare	$Rn - Op2$
<b>CMN</b>	Compare negated	$Rn + Op2$
<b>ORR</b>	Logical bit-wise OR	$Rd := Rn \text{ OR } Op2$
<b>MOV</b>	Move	$Rd := Op2$
<b>BIC</b>	Bit clear	$Rd := Rn \text{ AND NOT } Op2$
<b>MVN</b>	Nive begated	$Rd := \text{NOT } Op2$

# Multiply Instructions

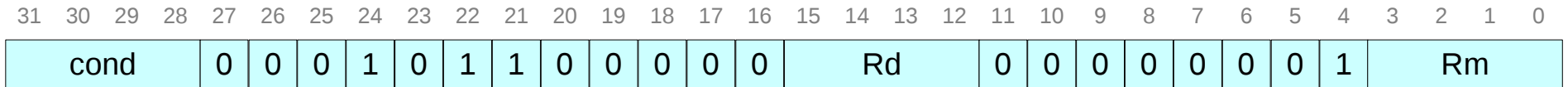


# Multiply Instructions

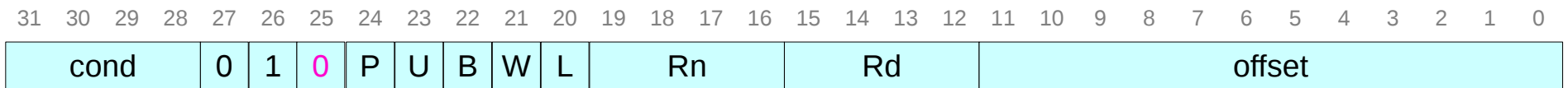


0	0	0	MUL	Multiply (32-bit result)	$Rd := (Rm * Rs)[31:0]$
0	0	1	MLA	Multiply-accumulate (32-bit result)	$Rd := (Rm * Rs)[31:0]$
1	0	0	UMULL	Unsigned multiply long	$RdHi.RdLo := Rm * Rs$
1	0	1	UMLAL	Unsigned multiply-accumulate long	$RdHi.RdLo += Rm * Rs$
1	1	0	SMULL	Signed multiply long	$RdHi.RdLo := Rm * Rs$
1	1	1	SMLAL	Signed multiply-accumulate long	$RdHi.RdLo += Rm * Rs$

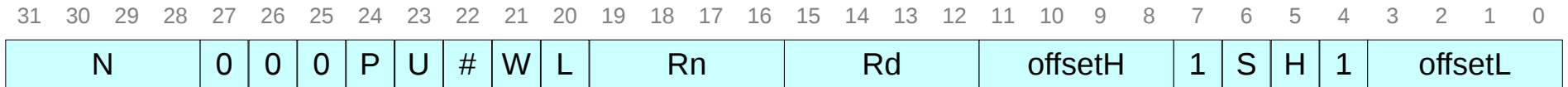
# CLZ (Count leading zeros)



# Single word and unsigned byte data transfer instructions

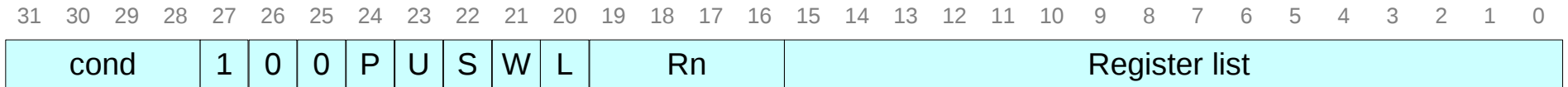


# Half-word and signed byte data transfer instructions





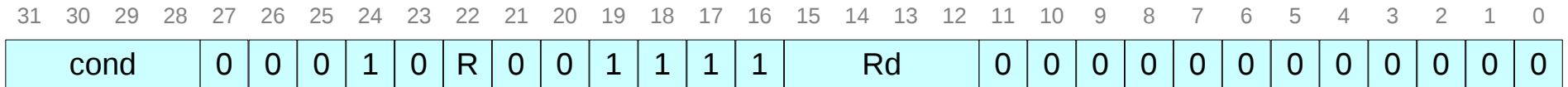
# Multiple register transfer instruction



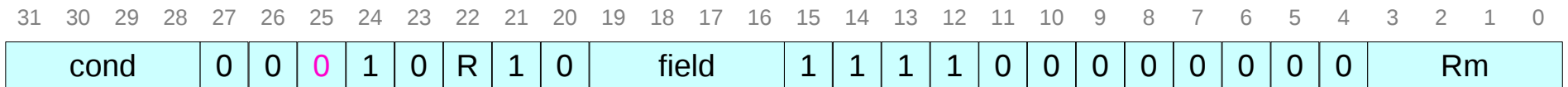
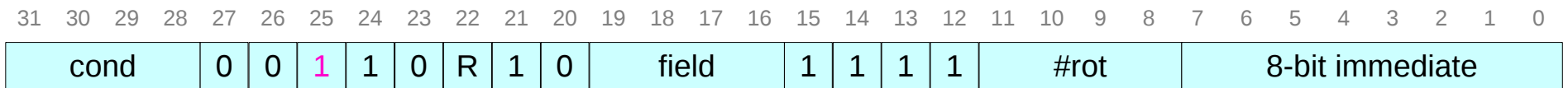
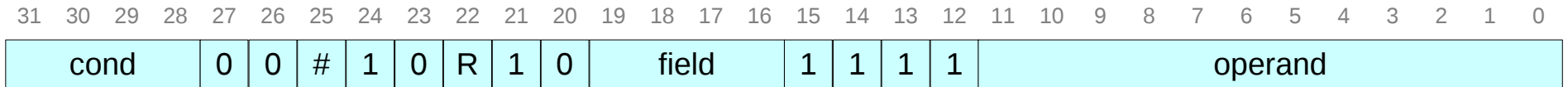
# Swap memory and register instruction (SWP)



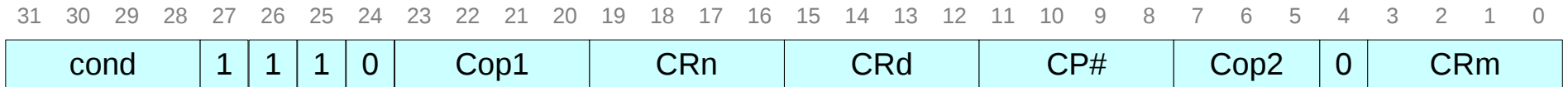
# Status register to general register transfer instructions



# General register to status register transfer instructions



# Coprocessor data operations



# Coprocessor data transfers



# Coprocessor register transfer

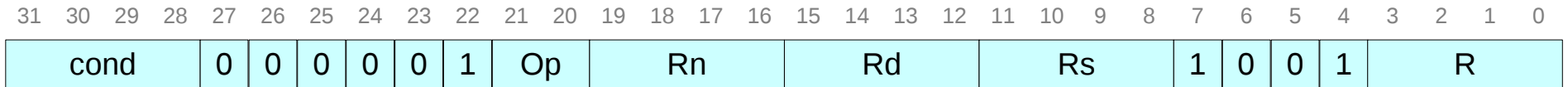


# Breakpoint instruction (BKPT)

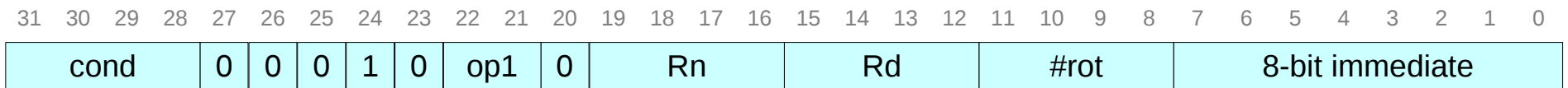
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	0	0	0	1	0	0	1	0	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	x	x	x	x



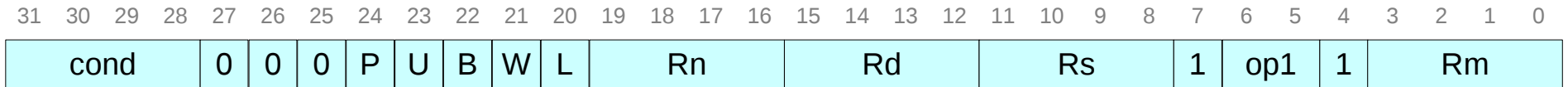
# Unused arithmetic instructions



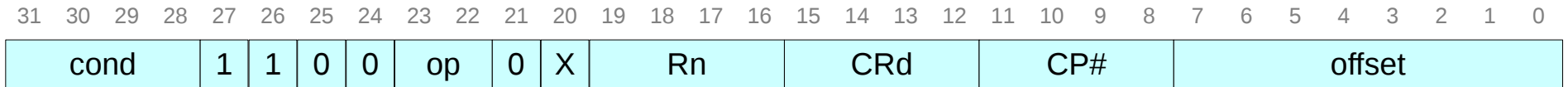
# Unused control instructions



# Unused load/store instructions



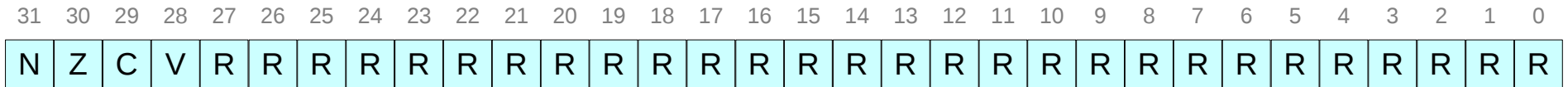
# Unused coprocessor instructions



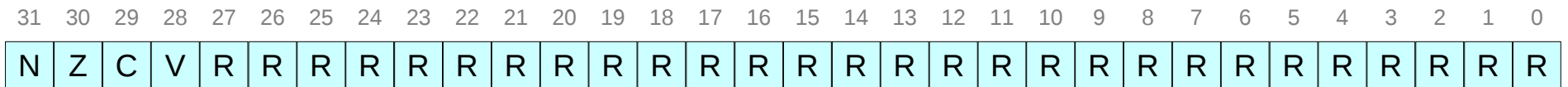
# Undefined instruction space

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
cond				0	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	X	X	X	X

# Multiple register transfer instruction



# ARM Exception Handling



## References

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