ISA Binary Encoding (2A)

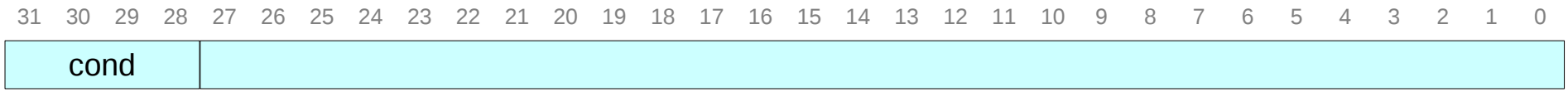
Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using LibreOffice.

Based on

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

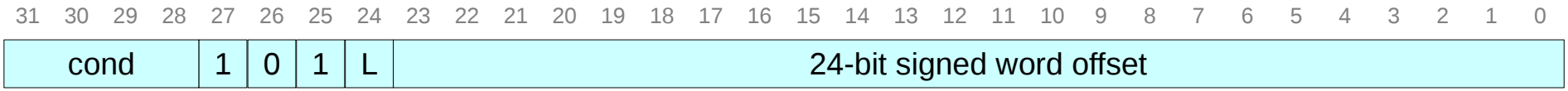
Condition Code



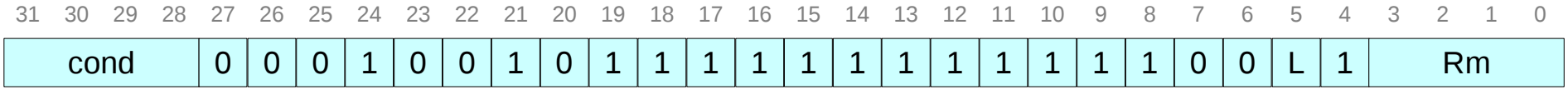
ARM Condition Codes

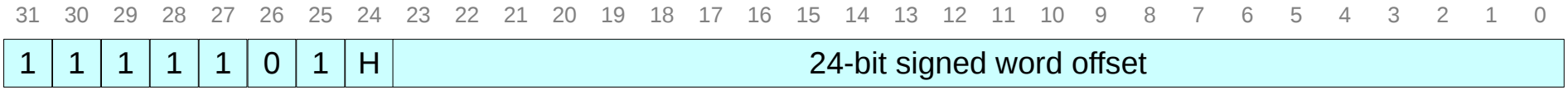
31	30	29	28			
0	0	0	0	EQ	EQual / EQuals zero	Z←1
0	0	0	1	NE	Not Equal	Z←0
0	0	1	0	CS/HS	Carry Set / unsigned High or Same	C←1
0	0	1	1	CC/LO	Carry Clear / unsigned Lower	C←0
0	1	0	0	MI	MInus / negative	N←1
0	1	0	1	PL	PLus / positive or zero	N←0
0	1	1	0	VS	oVerflow Set	V←1
0	1	1	1	VC	oVerflow Clear	V←0
1	0	0	0	HI	unsigned HIgher	C←1, Z←0
1	0	0	1	LS	unsigned Lower or Same	C←0, Z←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←0, N==V
1	1	0	1	LE	signed Less than or Equal	Z←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)

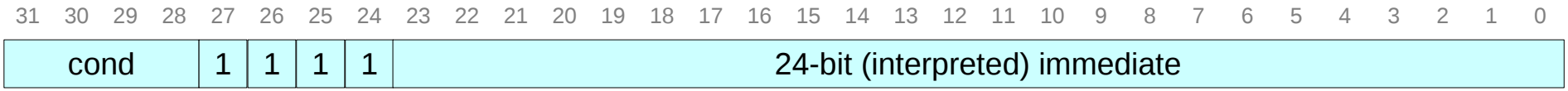


Branch, Branch with Link and eXchange (BX, BLX)

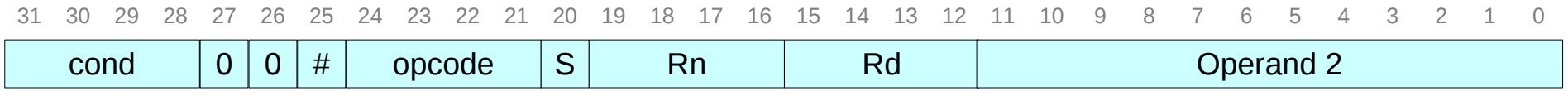


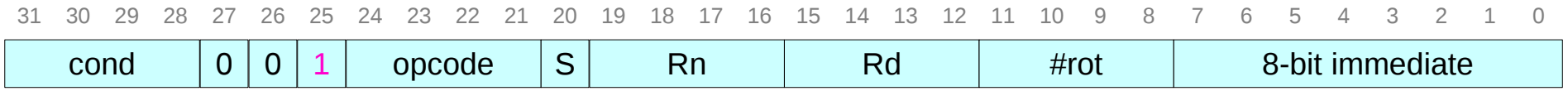


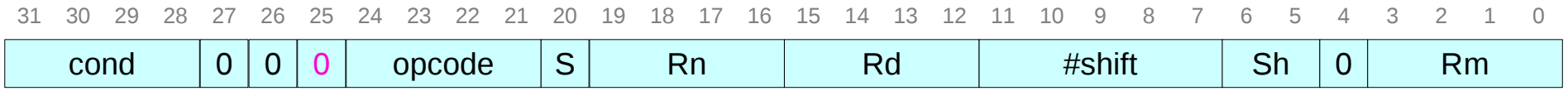
SWI (Software Interrupt)

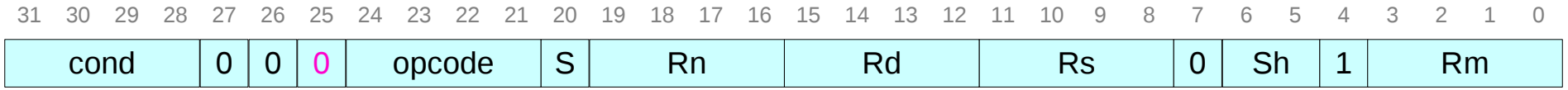


Data Processing Instructions







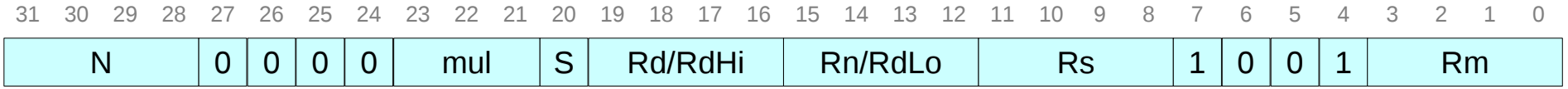


Data Processing Instructions

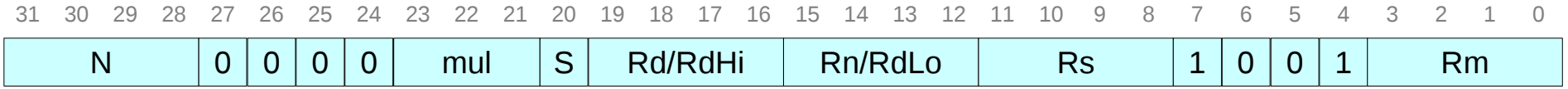
31	30	29	28	27	26	25	24	23	22	21	20
cond				0	0	#	0000				S
cond				0	0	#	0001				S
cond				0	0	#	0010				S
cond				0	0	#	0011				S
cond				0	0	#	0100				S
cond				0	0	#	0101				S
cond				0	0	#	0110				S
cond				0	0	#	0111				S
cond				0	0	#	1000				S
cond				0	0	#	1001				S
cond				0	0	#	1010				S
cond				0	0	#	1011				S
cond				0	0	#	1100				S
cond				0	0	#	1101				S
cond				0	0	#	1110				S
cond				0	0	#	1111				S

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

Multiply Instructions

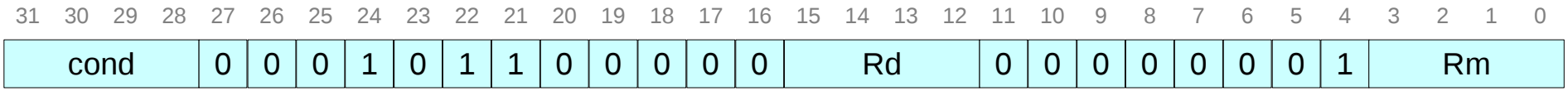


Multiply Instructions

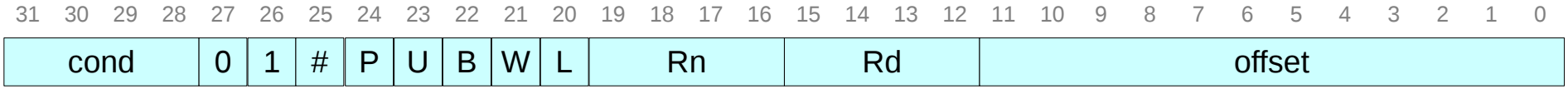


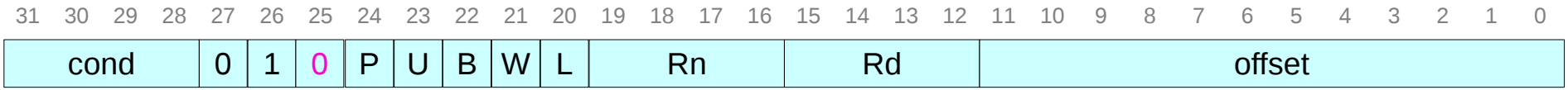
0	0	0	MUL	Multiply (32-bit result)	Rd := (Rm * Rs)[31:0]
0	0	1	MLA	Multiply-accumulate (32-bit result)	Rd := (Rm * Rs)[31:0]
1	0	0	UMULL	Unsigned multiply long	RdHi.RdLo := Rm * Rs
1	0	1	UMLAL	Unsigned multiply-accumulate long	RdHi.RdLo += Rm * Rs
1	1	0	SMULL	Signed multiply long	RdHi.RdLo := Rm * Rs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

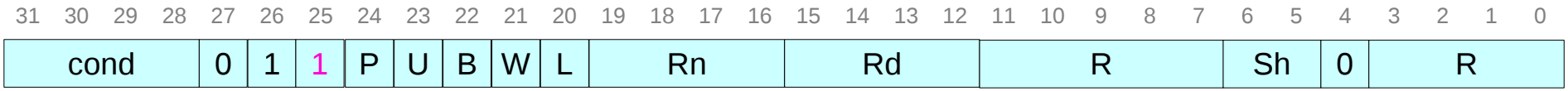
CLZ (Count leading zeros)



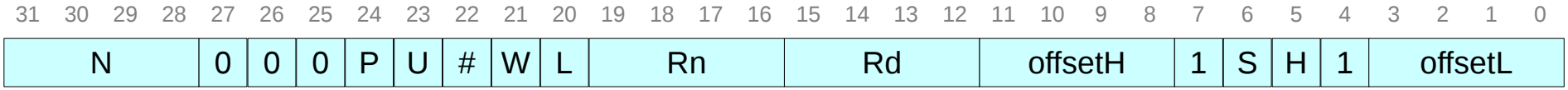
Single word and unsigned byte data transfer instructions

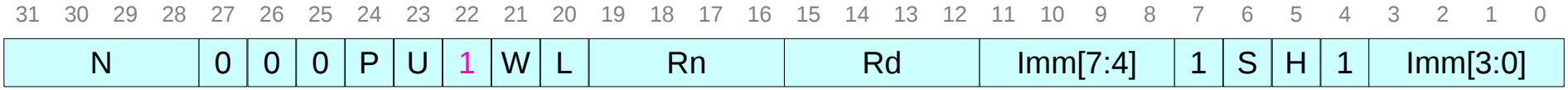


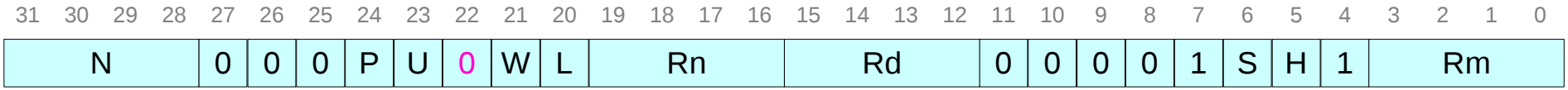




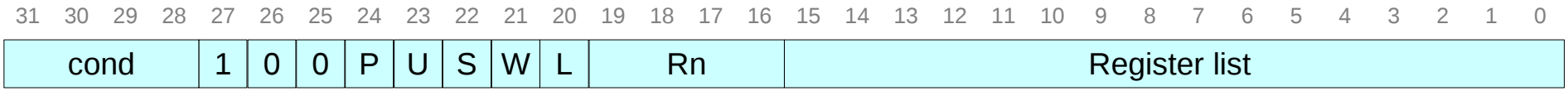
Half-word and signed byte data transfer instructions



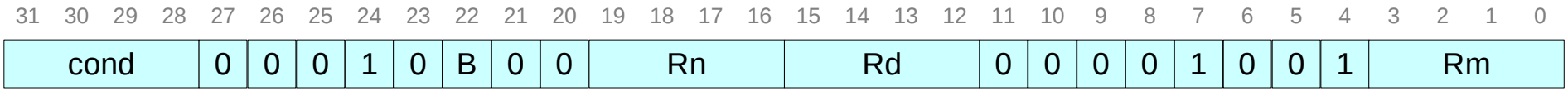




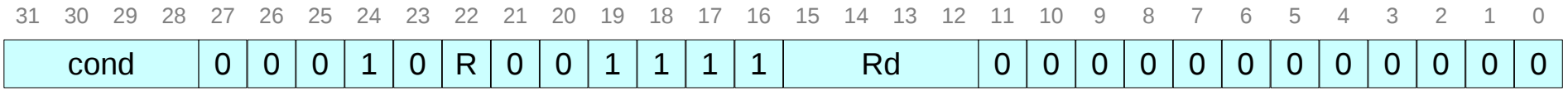
Multiple register transfer instruction



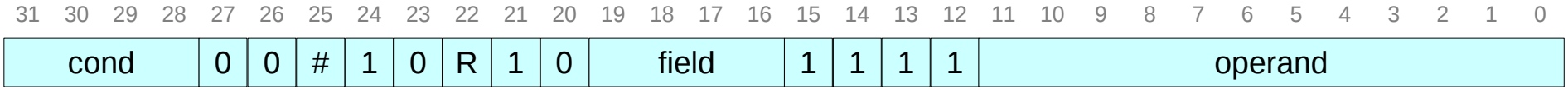
Swap memory and register instruction (SWP)

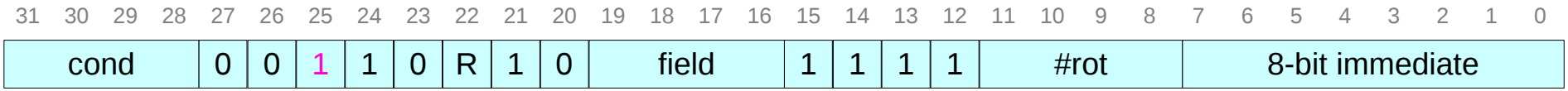


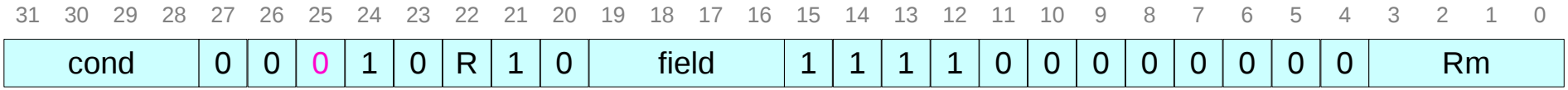
Status register to general register transfer instructions



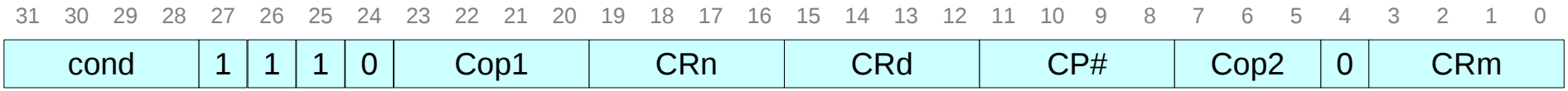
General register to status register transfer instructions



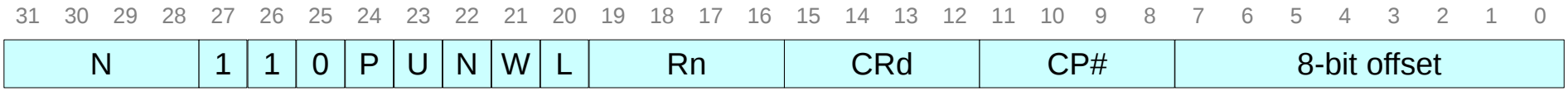




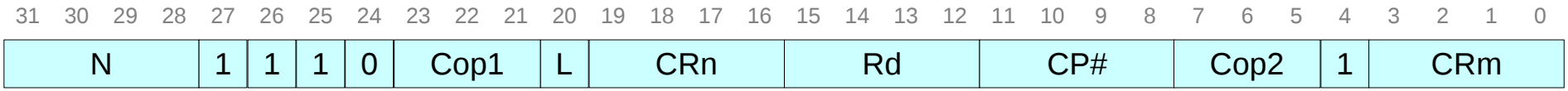
Coprocessor data operations



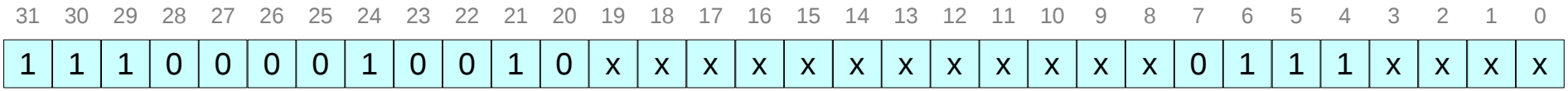
Coprocessor data transfers



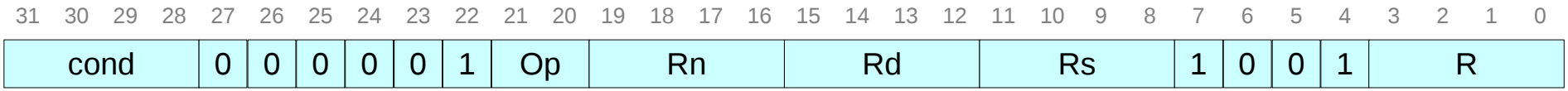
Coprocessor register transfer



Breakpoint instruction (BKPT)

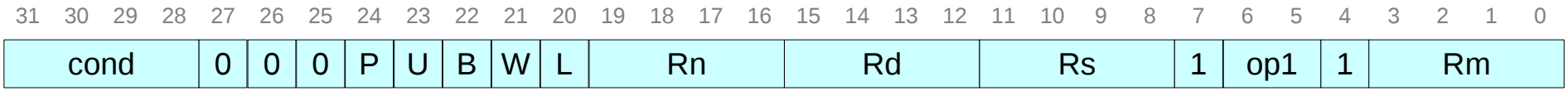


Unused arithmetic instructions

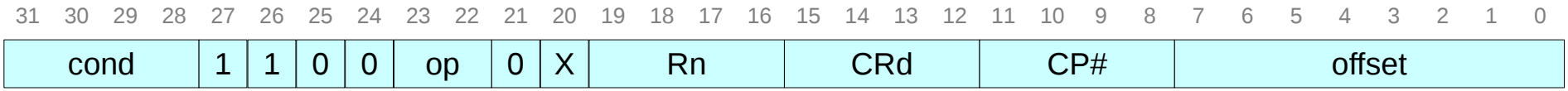


Unused control instructions

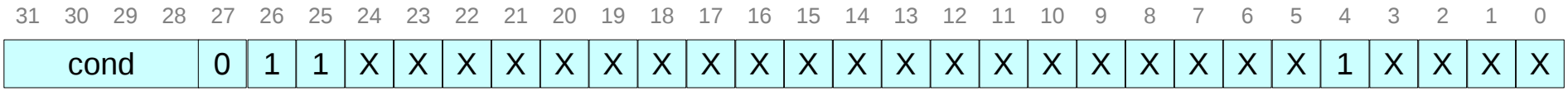
Unused load/store instructions



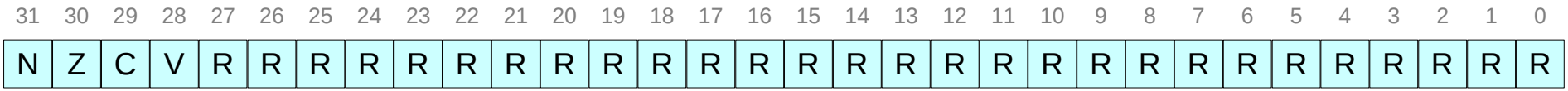
Unused coprocessor instructions



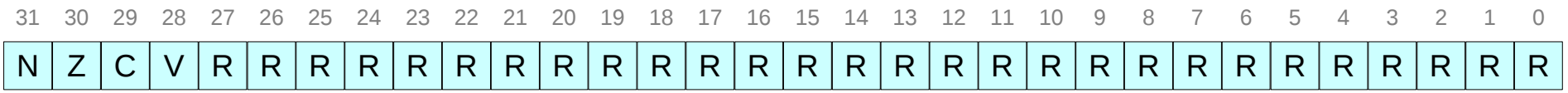
Undefined instruction space



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