ISA Binary Encoding (5A)

Young Won Lim
11/5/19

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

This document was produced by using LibreOffice.

Young Won Lim
11/5/19

Based on

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

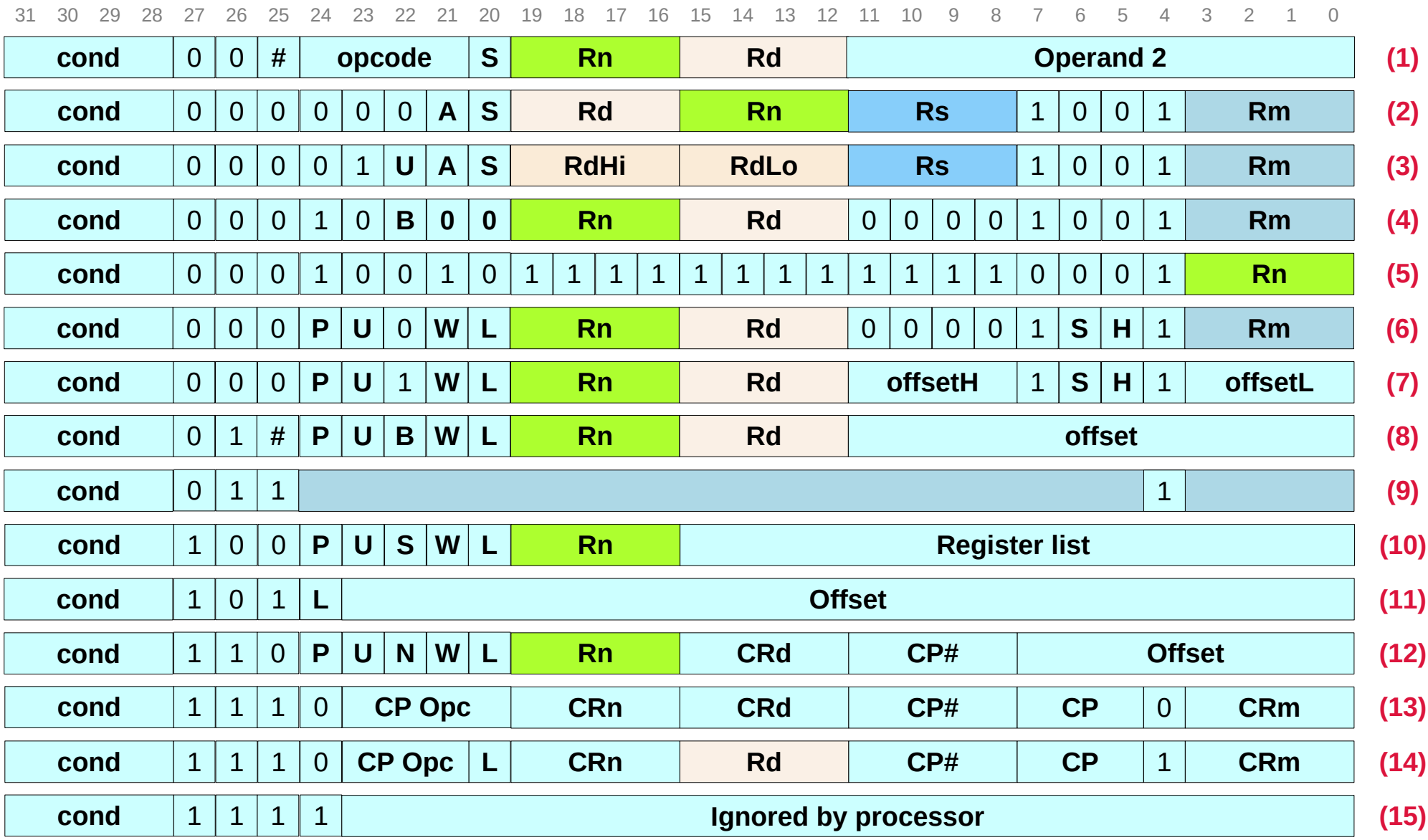
Rn	1st Operand Reg / Base Reg
Rm	2nd Operand Reg / Operand Reg / Offset Reg / Source Reg
Rd	Left most Reg : Source / Destination Reg
S	Set Condition Codes / Signed / Restore PSR and force user bit
P	Pre/Post Index
U	Up/Down
B	Unsigned Byte/Word
H	Half-word Address
L	Link / Load/Store
W	Write-back (auto-index)
Opcode	4-bit op codes
Sh	Shift type
R	CPSR/SPSR

Rn	1st Operand Reg / Base Reg
Rm	2nd Operand Reg / Operand Reg / Offset Reg / Source Reg
Rd	Source / Destination Reg
S	Set Condition Codes / Signed / Restore PSR and force user bit
B	Unsigned Byte/Word
H	Half-word Address
L	Link / Load/Store
T	Selects the user view in the non-usermodes

<cond>	Conditions for conditional execution
<shift>	Shift type and the shift amount (except RRX)
CPSR	Current Program Status Register
SPSR	Saved Program Status Register
RdHi	The most significant 32-bits
RdLo	The least significant 32-bits

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

All listings (1)



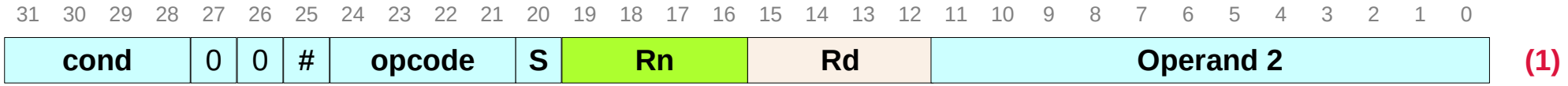
All listings (2)

(1)	Data Processing / PSR Transfer	Rd := Rn <op> operand2 (shifted)
(2)	Multiply	Rd := Rm * Rs + Rn
(3)	Multiply Long	RdHi : RdLo := Rm * Rs + RdHi : RdLo
(4)	Single Data Swap	Rd := [Rn]; [Rn]:= Rm
(5)	Branch and Exchange	PC := Rn; (Rn[0]=1 Thumb, else ARM)
(6)	Halfword Data Transfer: register offset	Rd :=: [Rn, Rm]; Rd :=: [Rn], Rm
(7)	Halfword Data Transfer: immediate offset	Rd :=: [Rn, Offset]; Rd :=: [Rn], Offset
(8)	Single Data Transfer	Rd :=: [Rn, Offset]; Rd :=: [Rn], Offset
(9)	Undefined	
(10)	Block Data Transfer	[Rn, i] :=: {Register List}
(11)	Branch	R14 := PC+8; PC := Offset
(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
(15)	Software Interrupt	

All listings (3)

(1)	Data Processing / PSR Transfer	I, Opcode, S, Rn, Rd, Operand2
(2)	Multiply	A, S, Rd, Rn, Rs, Rm
(3)	Multiply Long	U, A, S, RdHi, RdLo, Rs, Rm
(4)	Single Data Swap	B, Rn, Rd, Rm
(5)	Branch and Exchange	Rm
(6)	Halfword Data Transfer: register offset	P,U, W, L, Rn, Rd, S, H, Rm
(7)	Halfword Data Transfer: immediate offset	P,U, W, L, Rn, Rd, Offset, S, H, Offset
(8)	Single Data Transfer	P, U, B, W, L, Rn, Rd, Offset
(9)	Undefined	
(10)	Block Data Transfer	P, U, S, W, L, Rn, Register List
(11)	Branch	L, Offset
(12)	Coprocessor Data Transfer	P, U, N, W, L, Rn , CRd, CP#, Offset
(13)	Coprocessor Data Operation	CP Opc, CRn, CRd, CP#, CP, CRm
(14)	Coprocessor Register Transfer	CP Opc, L, CRn, Rd, CP# CP, CRm
(15)	Software Interrupt	Software Interrupt

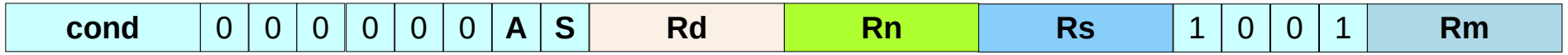
1. Data Processing Instructions



(1)

Data Processing / PSR Transfer

Rd := Rn <op> operand2 (shifted)



(2)

Multiply

Rd := Rm * Rs + Rn

(3)

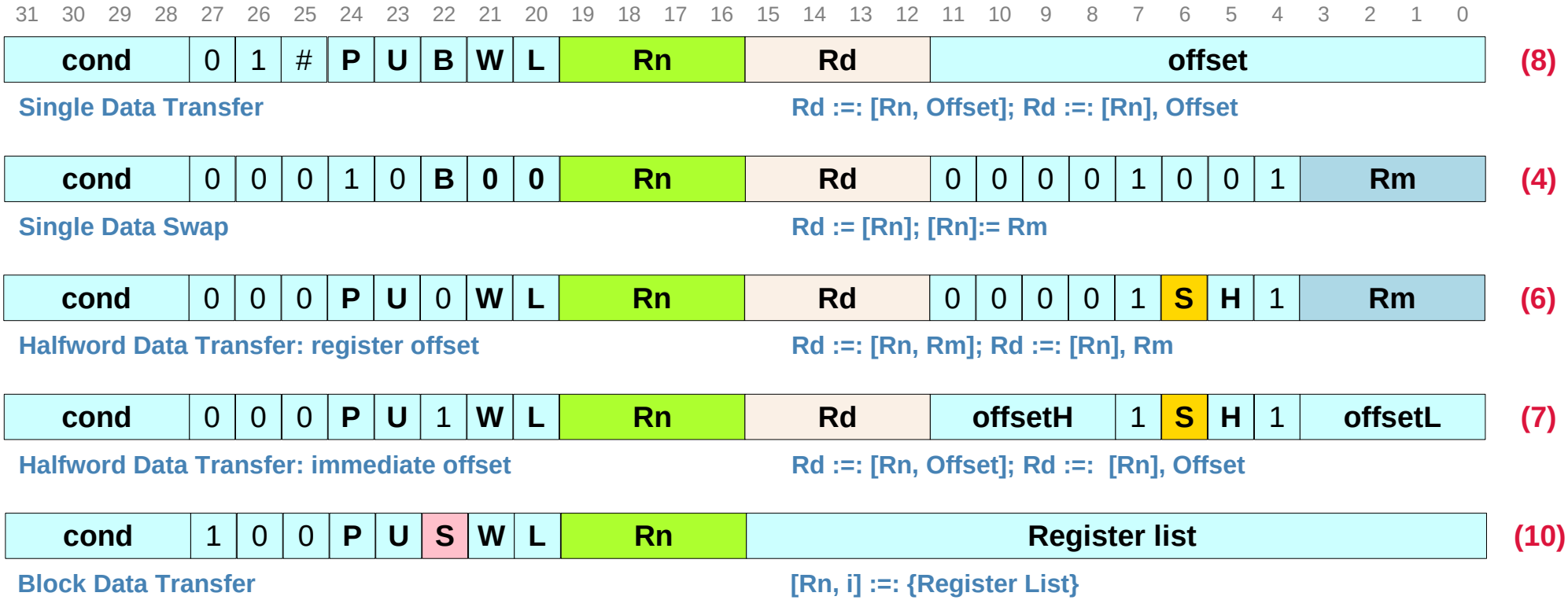
Multiply Long

RdHi : RdLo := Rm * Rs + RdHi : RdLo

#	Immediate Operand
S	Set Condition Codes
A	Accumulate
U	Unsigned
B	Unsigned Byte/Word

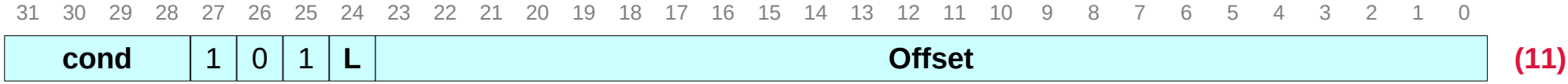
Rn	Operand (1st / Adder) / Base Reg
Rm	Operand (Multiplicand) / Source Reg
Rs	Operand (Multiplier) Reg
Rd	Destination Reg
RdHi	Destination Reg
RdLo	Destination Reg

2. Data Transfer Instructions



P	Pre/Post Index	**Rn**	Base Reg
U	Up/Down	**Rm**	Offset Reg
B	Unsigned Byte/Word	**Rd**	Source/Destination Reg
W	Write-back (auto-index)	**S**	Signed
L	Load/Store	**H**	Half-word Address
S	PSR & force user mode		

3. Branch and Branch Exchange



(11)

Branch **R14 := PC=8; PC := Offset**

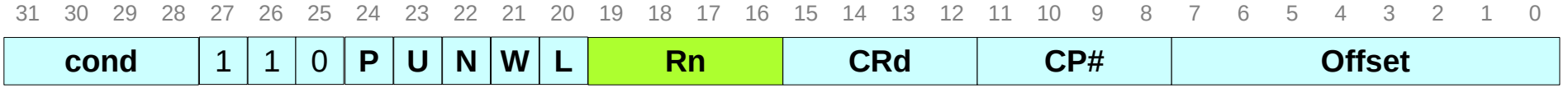
(5)

Branch and Exchange **PC := Rn; (Rn[0]=1 Thumb, else ARM)**

L Link

Rn Operand Reg

4. Coprocessor Instructions



(12)

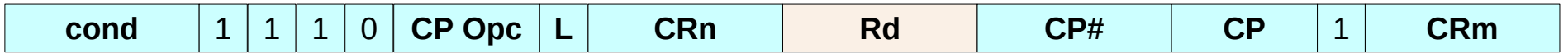
Coprocessor Data Transfer

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

(13)

Coprocessor Data Operation

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



(14)

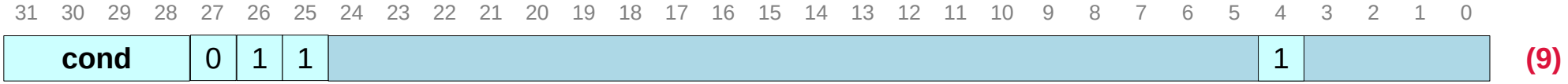
Coprocessor Register Transfer

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

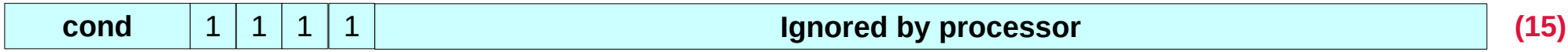
P	Pre/Post Index
U	Up/Down
N	Transfer Length
W	Write-back (auto-index)
L	Load/Store

<CP#>	Coprocessor number
<Cop1>	Coprocessor operation 1
<Cop2>	Coprocessor operation 2
CRd	Coprocessor Rd
CRn	Coprocessor Rn
CRm	Coprocessor Rm
Rn	Base Reg
Rd	Destination Reg

5. Miscellaneous Instructions

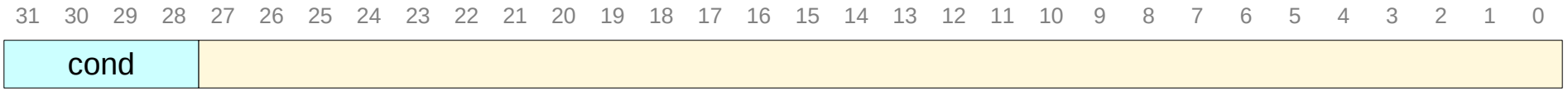


Undefined



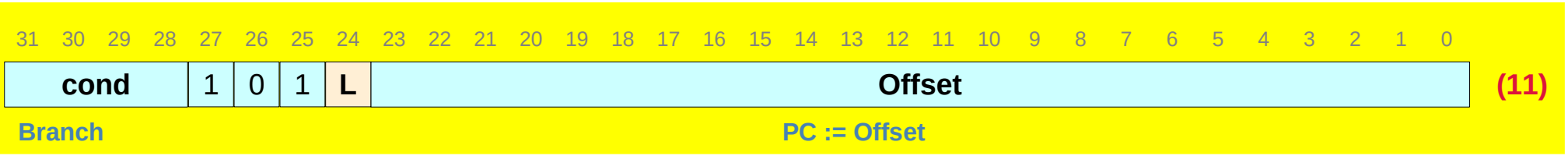
Software Interrupt

Condition Codes



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 **PC := Offset**

cond	1	0	1	0	24-bit signed word offset

Branch **B{<cond>} <target address>** **PC := Offset**

cond	1	0	1	1	24-bit signed word offset

Branch with Link **BL{<cond>} <target_address>** **R14 := PC+8; PC := Offset**

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

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	0	0	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	**0**	1	**Rn**				(5)

Branch and Exchange — **PC := Rn; (Rn[0]=1 Thumb, else ARM)**

cond				0	0	0	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	**0**	1	**Rn**

Branch and Exchange — **BX{<cond>} Rn** — **PC := Rn; (Rn[0]=1 Thumb, else ARM)**

cond				0	0	0	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	**1**	1	**Rn**

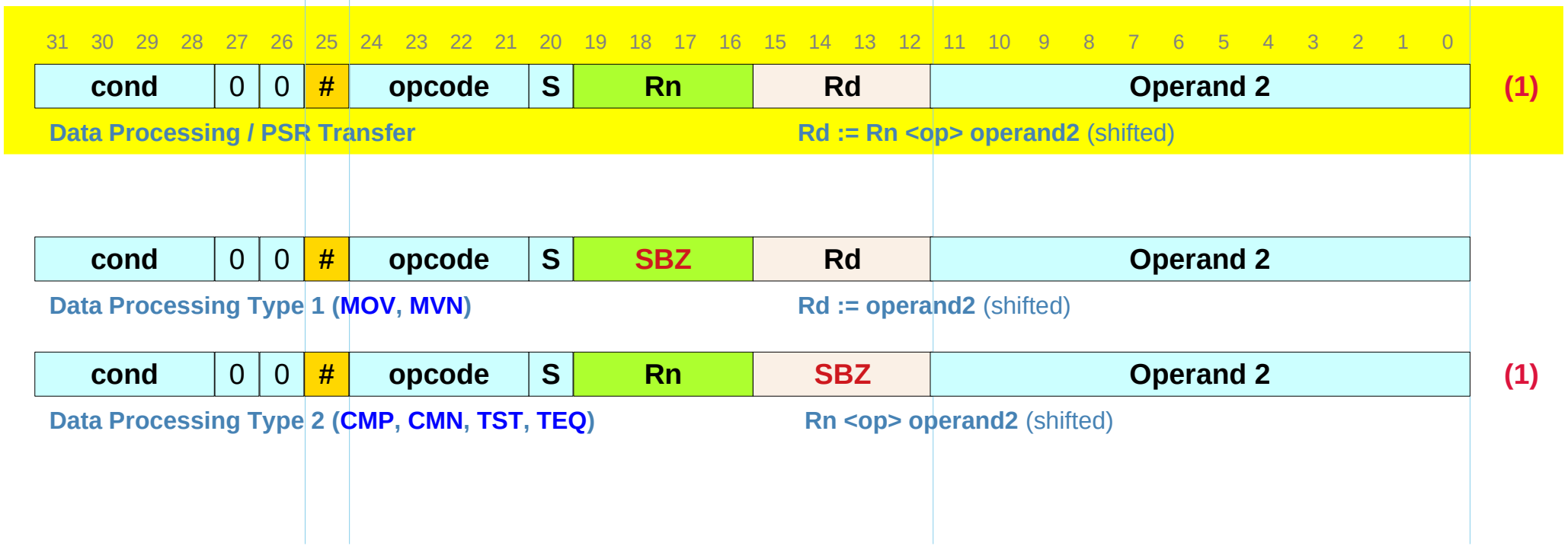
Branch with Link and Exchange — **BLX{<cond>} Rn** — **R14 := PC+8; PC := Rn; (Rn[0]=1 Thumb, else ARM)**

1	**1**	**1**	**1**	1	0	1	**H**	**Offset**

Branch with Link and Exchange — **BLX{<cond>} <address>** — **PC := Offset**

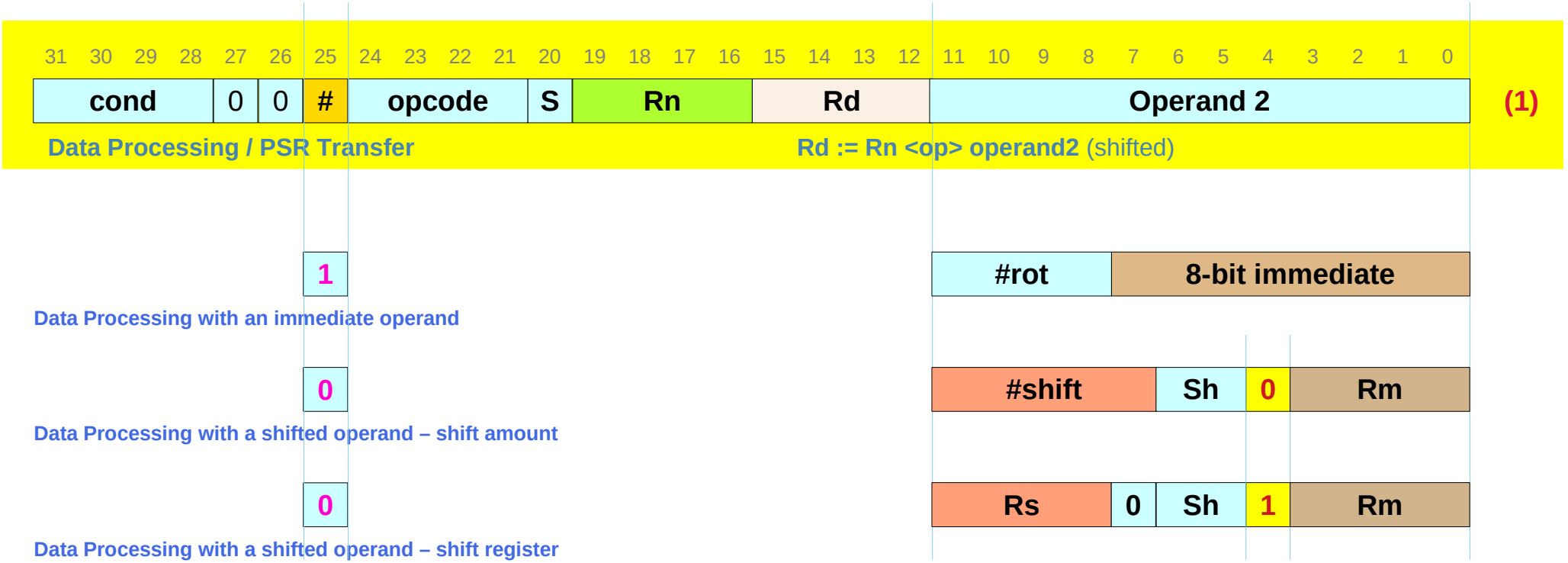
H Half Word Address

Data Processing Instructions



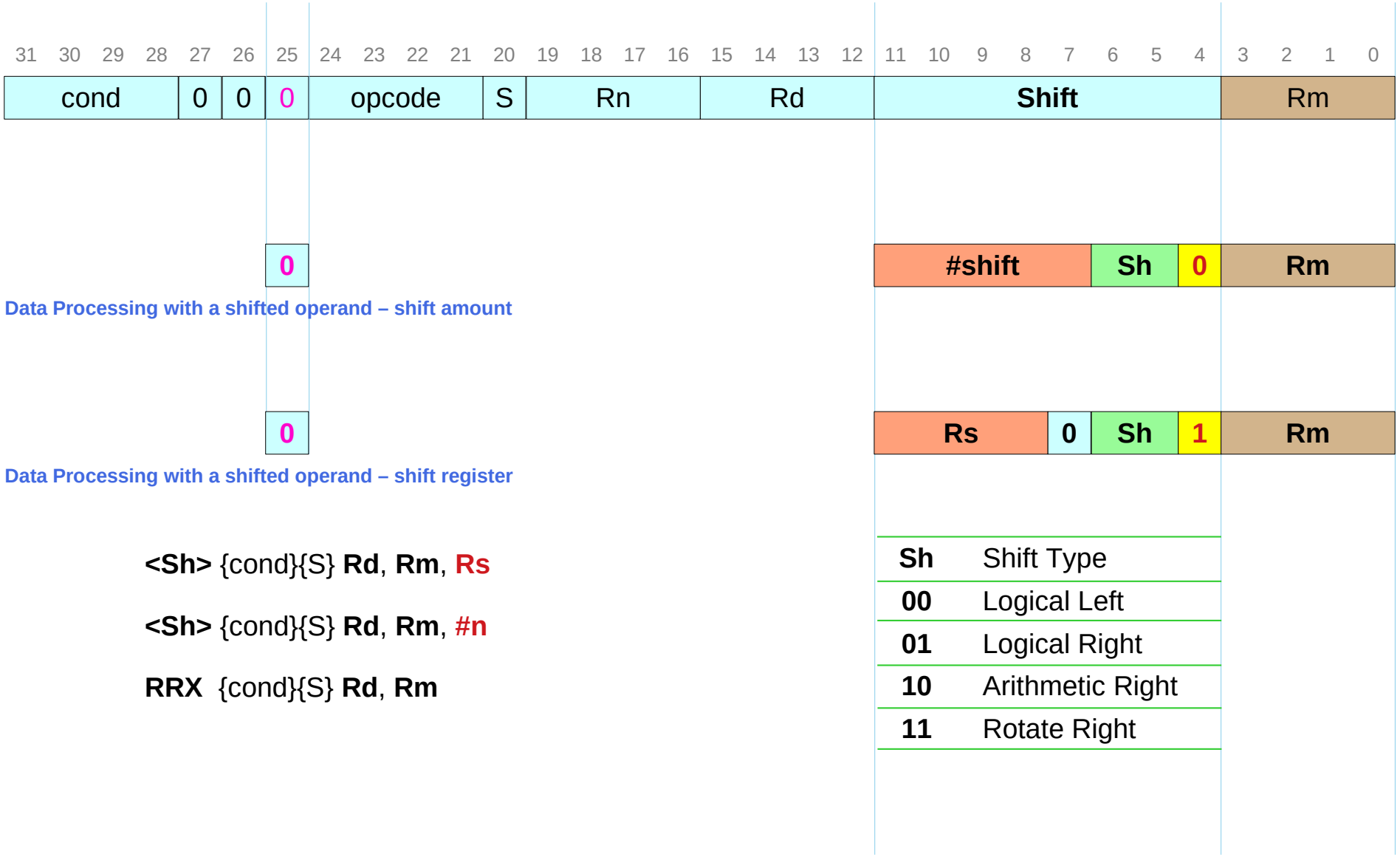
#	Immediate Operand	**Rn**	1st Operand Reg
S	Set Condition Codes	**Rm**	2nd Operand Reg
rot#	Shift applied to the 8-bit immeidate	**Rs**	Shift (amount) Reg
#shift	Shift amount (5-bit immediate)	**Rd**	Destination Reg
Sh	Shift Type	**SBZ**	Should Be Zero

Data Processing Instructions



#	Immediate Operand	**Rn**	1st Operand Reg
S	Set Condition Codes	**Rm**	2nd Operand Reg
rot#	Shift applied to the 8-bit immeidate	**Rs**	Shift (amount) Reg
#shift	Shift amount (5-bit immediate)	**Rd**	Destination Reg
Sh	Shift Type		

Data Processing Instructions – a shifted operand



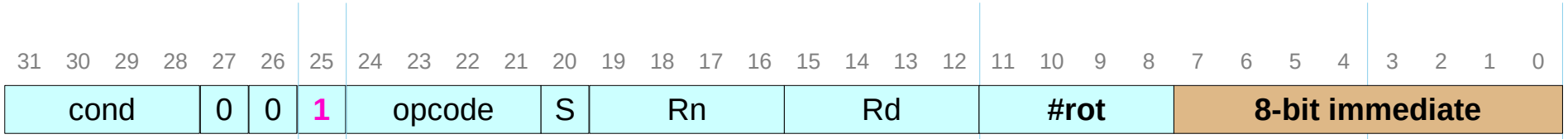
<Sh> {cond}{S} **Rd**, **Rm**, **Rs**

<Sh> {cond}{S} **Rd**, **Rm**, **#n**

RRX {cond}{S} **Rd**, **Rm**

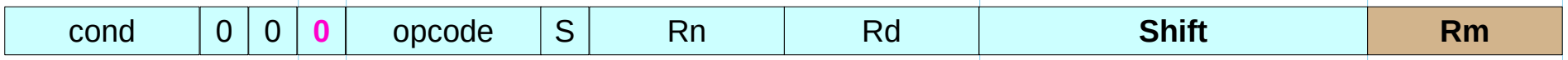
Sh	Shift Type
00	Logical Left
01	Logical Right
10	Arithmetic Right
11	Rotate Right

Data Processing Instructions – operand2 examples



Data Processing with an immediate operand

0x**A**_**05** // rotate 0x05 right by 20 (=2 * 0x**A**)
0x**B**_**14** // rotate 0x14 right by 22 (=2 * 0x**B**)
0x**C**_**50** // rotate 0x50 right by 24 (=2 * 0x**C**)



Data Processing with a shifted operand – shift amount

ADD r3, r2, **r1**, **LSL #3** ; r3 := r2 + **r1***2^3
MOV r0, **r0**, **LSR #2** ; r0 := **r0** / 2^2

Data Processing with a shifted operand – shift register

ADD r5, r5, **r3**, **LSL r2** ; r5 := r5 + **r3***2^r2

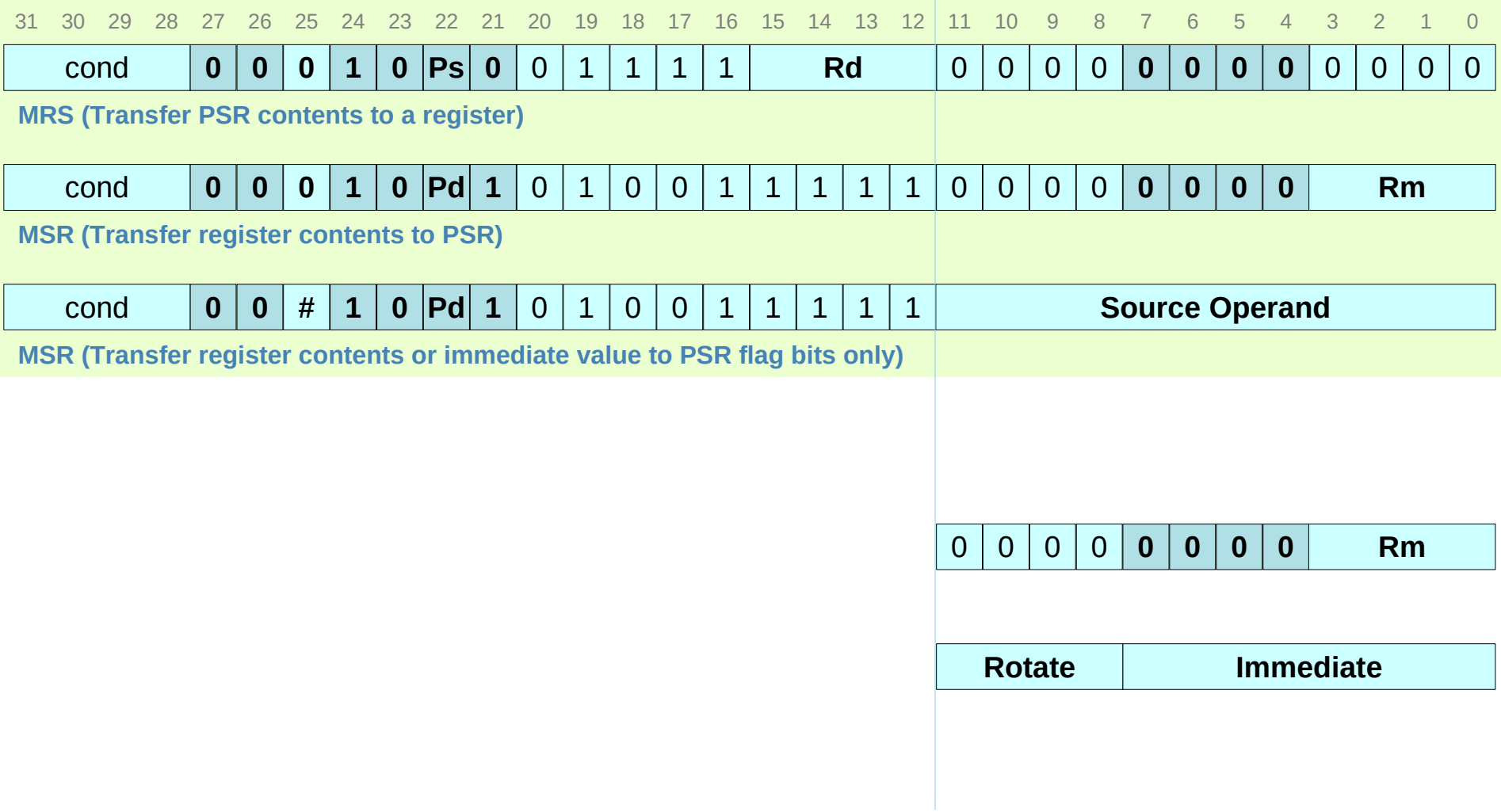
Data Processing Opcode

31	30	29	28	27	26	25	24	23	22	21	20
cond				0	0	I	**opcode**				S

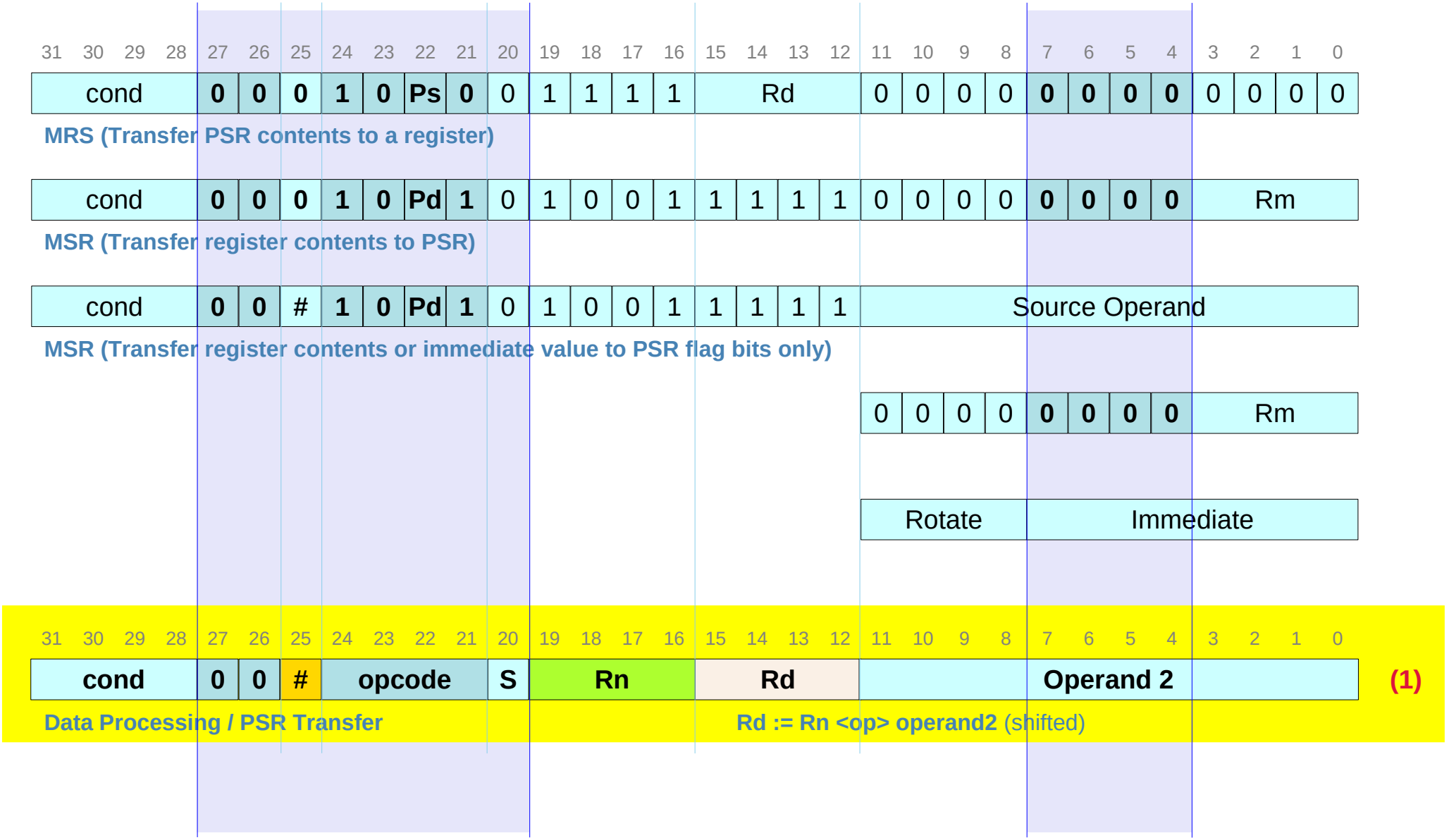
cond	0	0	#	**0000**	S	
cond	0	0	#	**0001**	S	
cond	0	0	#	**0010**	S	
cond	0	0	#	**0011**	S	
cond	0	0	#	**0100**	S	
cond	0	0	#	**0101**	S	
cond	0	0	#	**0110**	S	
cond	0	0	#	**0111**	S	
cond	0	0	#	**1000**	S	
cond	0	0	#	**1001**	S	
cond	0	0	#	**1010**	S	
cond	0	0	#	**1011**	S	
cond	0	0	#	**1100**	S	
cond	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	Set CC on Rn **AND** Op2
TEQ	Test equivalence	Set CC on Rn **EOR** Op2
CMP	Compare	Set CC on Rn **–** Op2
CMN	Compare negated	Set CC on 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

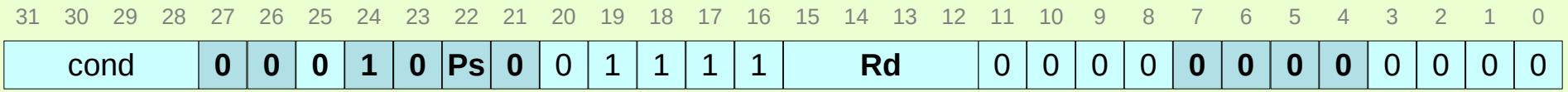
PSR Transfer (MRS, MSR)



PSR Transfer (MRS, MSR) – decoding

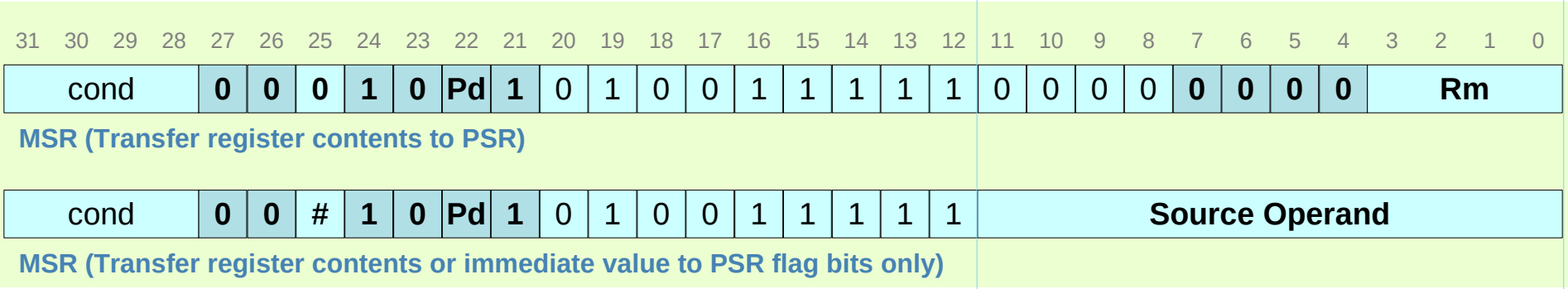


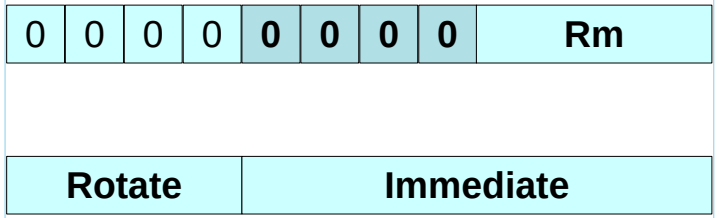
Status register to general register transfer instructions



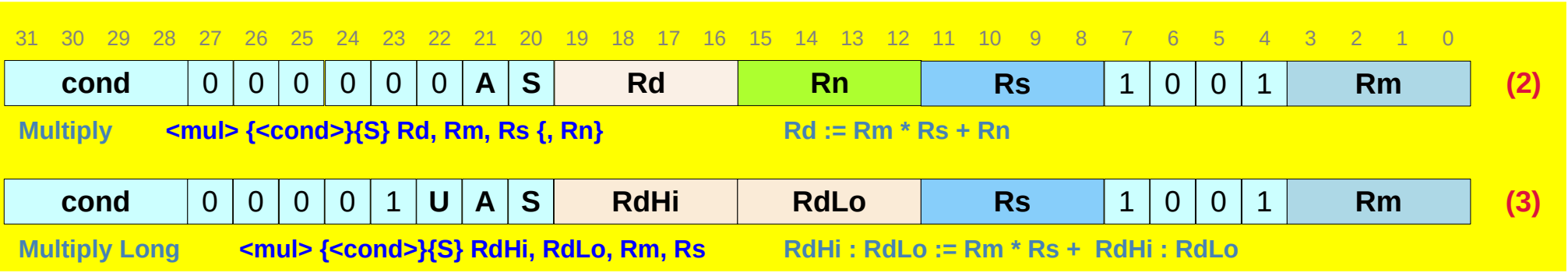
MRS (Transfer PSR contents to a register)

General register to status register transfer instructions





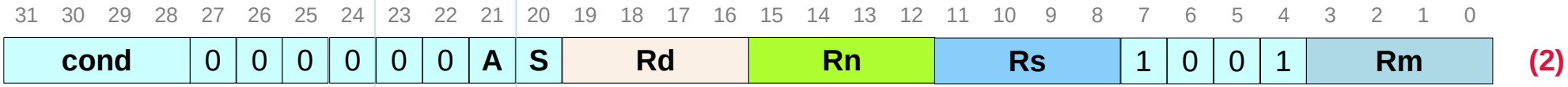
Multiply Instructions



S	Set Condition Codes
A	Accumulate
U	Unsigned

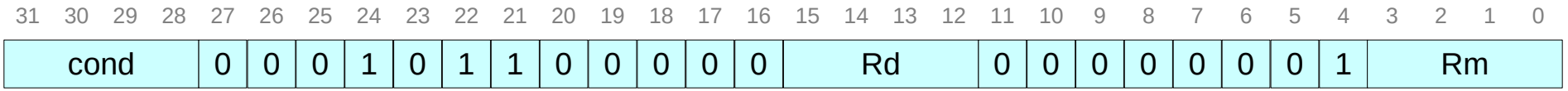
Rn	Accumulator Operand
Rm	Operand (Multiplicand)
Rs	Operand (Multiplier)
Rd	Destination
RdHi	Destination (Hi Word)
RdLo	Destination (Lo Word)

Multiply Instruction Opcode



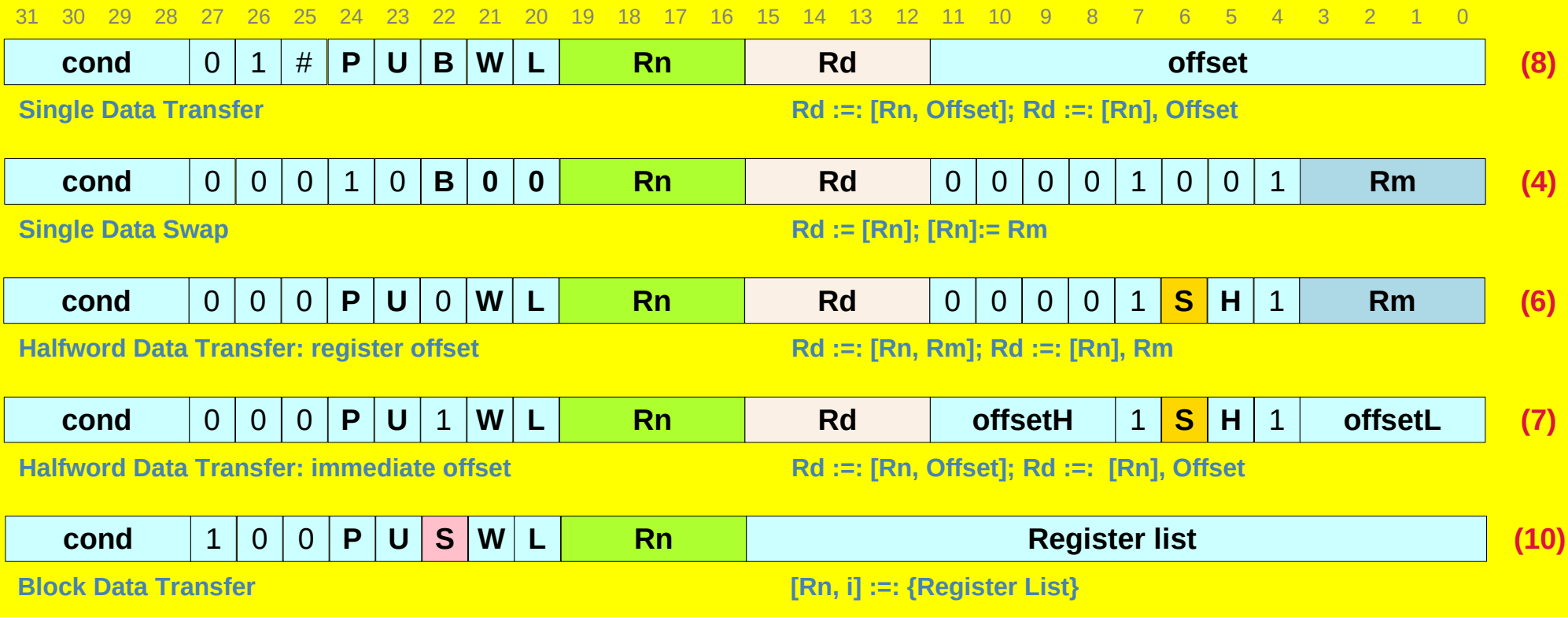
0	0	0	**MUL**	Multiply (32-bit result)	Rd := (Rm * Rs)[31:0]
0	0	1	**MLA**	Multiply-accumulate (32-bit result)	Rd := (Rm * Rs + Rn)[31:0]
0	1	0			
0	1	1			
1	0	0	**UMULL**	Unsigned multiply long	RdHi.RdLo := Rm * Rs
1	0	1	**UMLAL**	Unsigned multiply-accumulate long	RdHi.RdLo += Rm * Rs
1	1	0	**SMULL**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

CLZ (Count leading zeros)



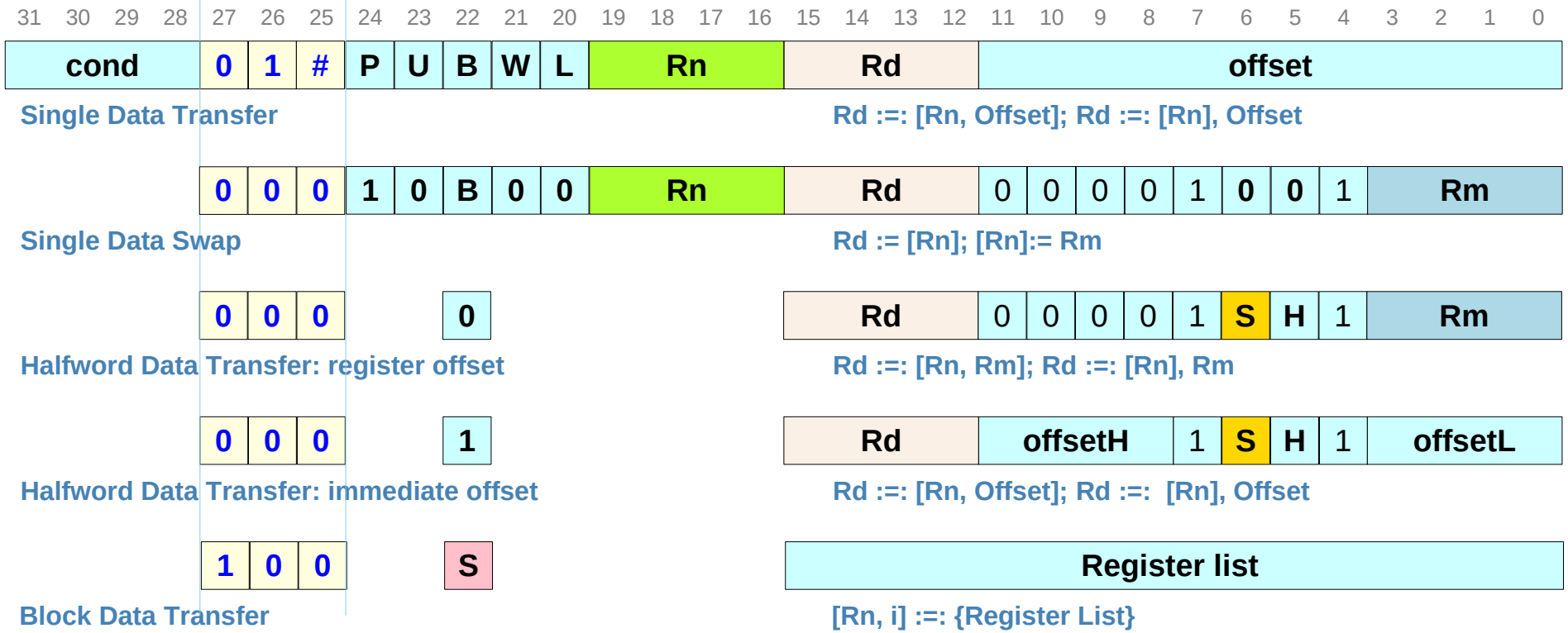
CLZ

Data Transfer Instructions



P	Pre/Post Index	**Rn**	Base Reg
U	Up/Down	**Rm**	Offset Reg
B	Unsigned Byte/Word	**Rd**	Source/Destination Reg
W	Write-back (auto-index)	**S**	Signed
L	Load/Store	**H**	Half-word Address
S	PSR & force user mode		

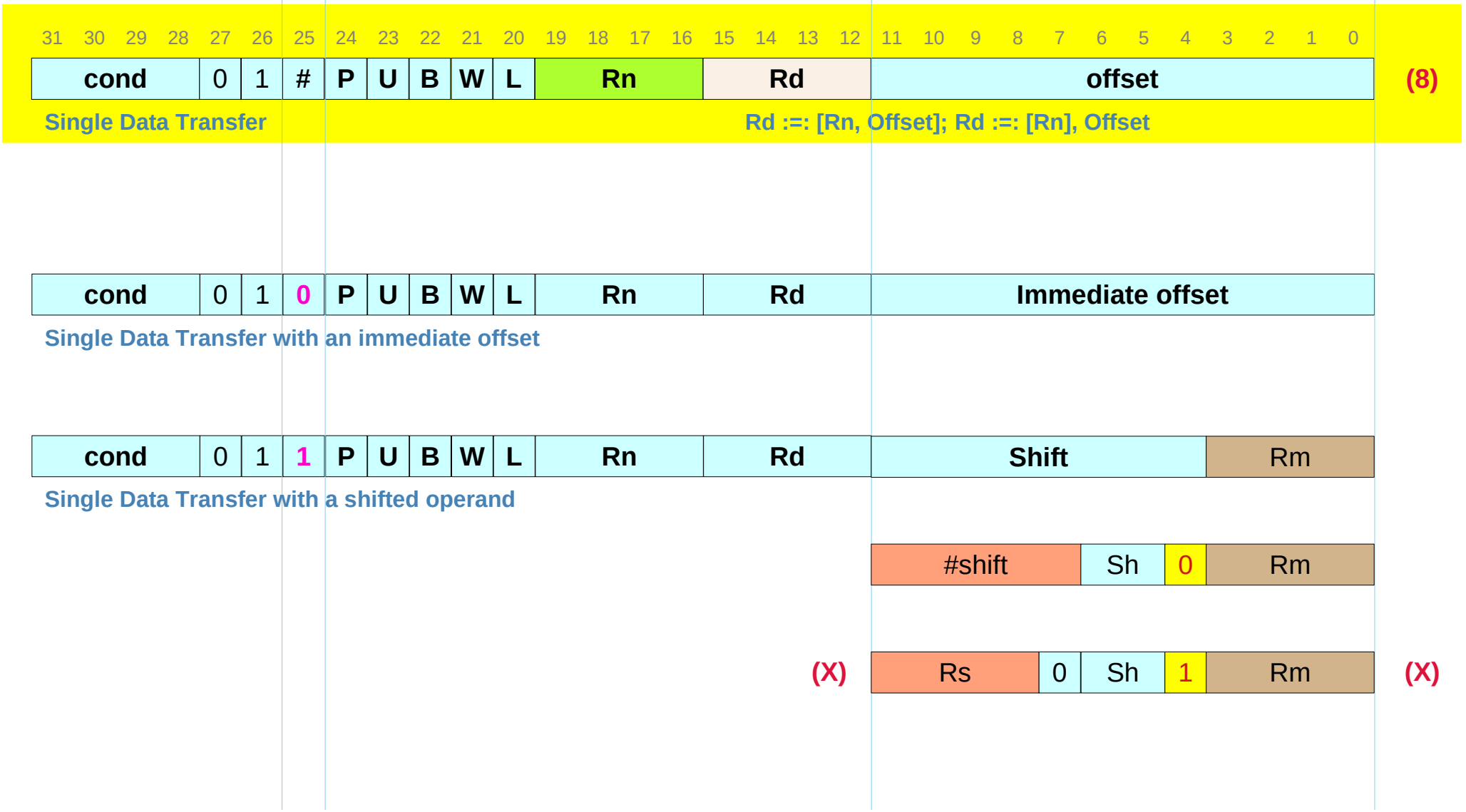
Data Transfer Instructions



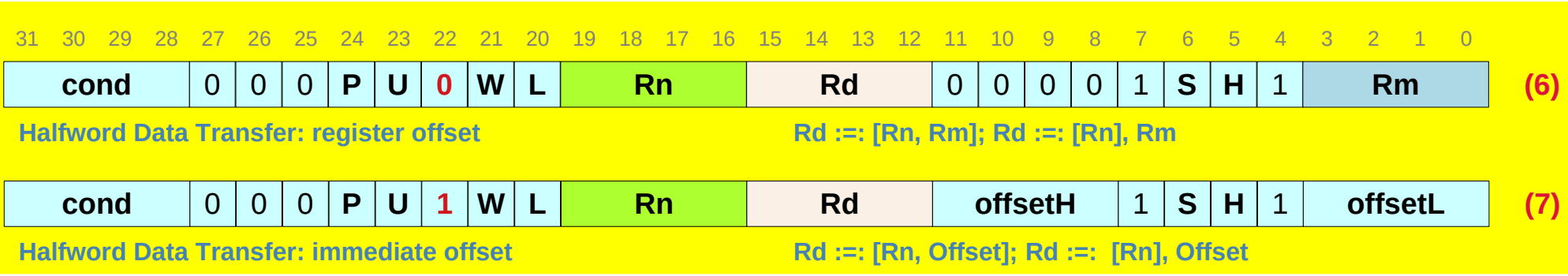
P	Pre/Post Index
U	Up/Down
B	Unsigned Byte/Word
W	Write-back (auto-index)
L	Load/Store
S	PSR & force user mode

Rn	Base Reg
Rm	Offset Reg
Rd	Source/Destination Reg
S	Signed
H	Half-word Address

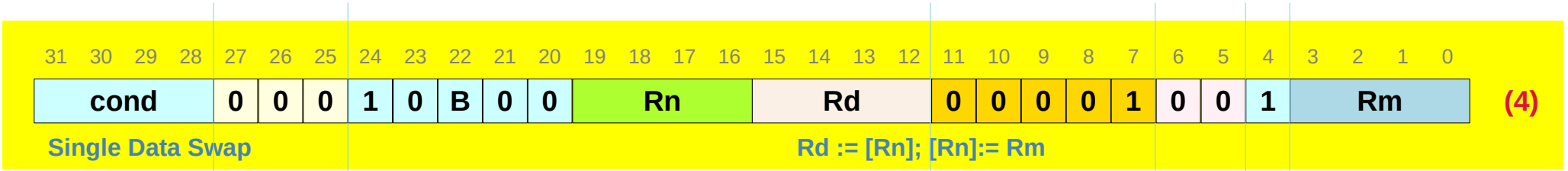
Single word data transfer instructions

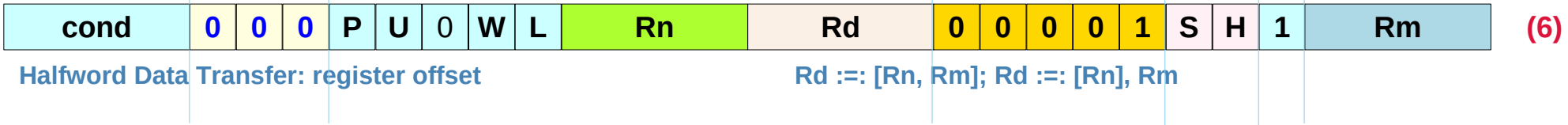


Half-word data transfer instructions



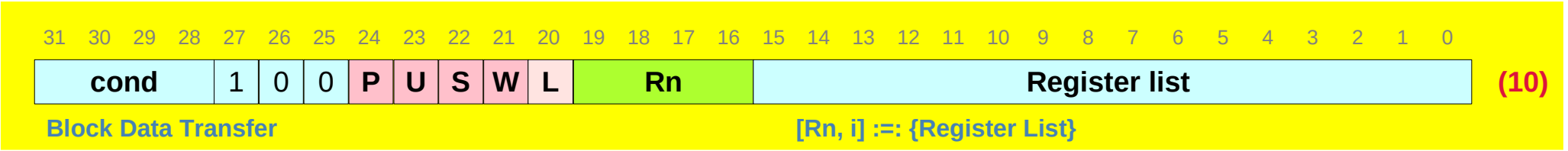
Swap memory and register instruction (SWP)





SH	
00	SWP instruction
01	Unsigned Halfwords
10	Signed Byte
11	Signed Halfwords

Block data transfer instruction



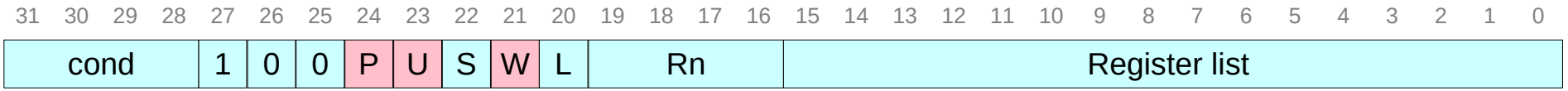
P=1 : before **U=1** : Increment **W=1** : auto index **L=1** : **LDM**
P=0 : after **U=0** : Decrement **W=0** : no auto index **L=0** : **STM**

P : Pre/Post **U : Up/Down** **W : Write-back (auto index)** **S :** PSR & force user mode

IB	**PU=11**	**DA**	**PU=00**
IA	**PU=01**	**IA**	**PU=01**
DB	**PU=10**	**DB**	**PU=10**
DA	**PU=00**	**IB**	**PU=11**

Name	Stack	Block	L	P	U
Pre-Increment Load	**LDMED**	**LDMIB**	1	1	1
Post-Increment Load	**LDMFD**	**LDMIA**	1	0	1
Pre-Decrement Load	**LDMEA**	**LDMDB**	1	1	0
Post-Decrement Load	**LDMFA**	**LDMDA**	1	0	0
Pre-Increment Store	**STMFA**	**STMIB**	0	1	1
Post-Increment Store	**STMEA**	**STMIA**	0	0	1
Pre-Decrement Store	**STMFD**	**STMDB**	0	1	0
Post-Decrement Store	**STMED**	**STMDA**	0	0	0

Block data transfer instruction examples



IB	**PU=11**
IA	**PU=01**
DB	**PU=10**
DA	**PU=00**

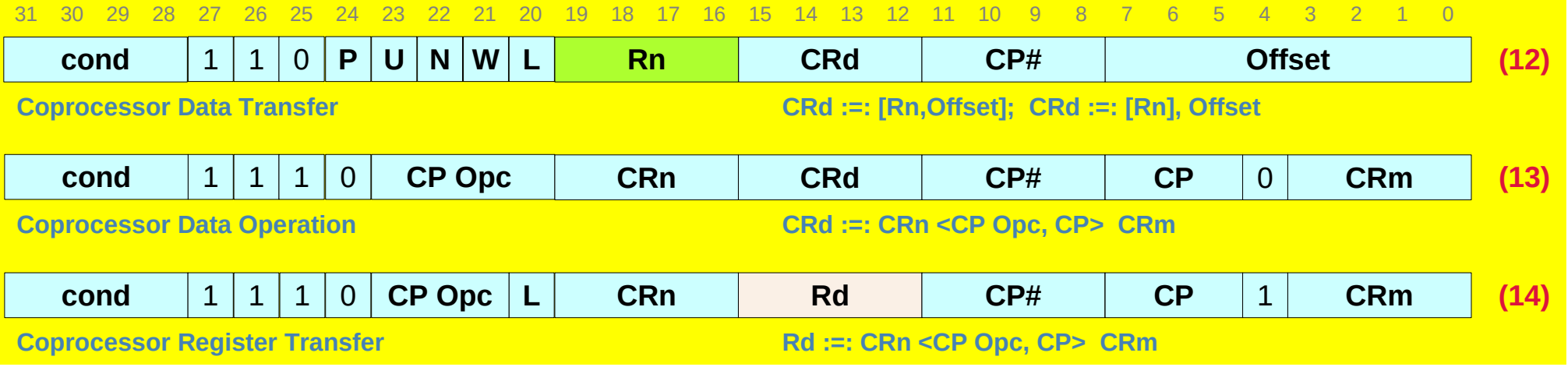
```
STMIB R8! {R0,R1,R4}      LDMDA R8! {R0,R1,R4}
STMFA R8! {R0,R1,R4}      LDMFA R8! {R0,R1,R4}

STMIA R8! {R0,R1,R4}      LDMDB R8! {R0,R1,R4}
STMEA R8! {R0,R1,R4}      LDMEA R8! {R0,R1,R4}

STMDB R8! {R0,R1,R4}      LDMIA R8! {R0,R1,R4}
STMFD R8! {R0,R1,R4}      LDMFD R8! {R0,R1,R4}

STMDA R8! {R0,R1,R4}      LDMIB R8! {R0,R1,R4}
STMED R8! {R0,R1,R4}      LDMED R8! {R0,R1,R4}
```

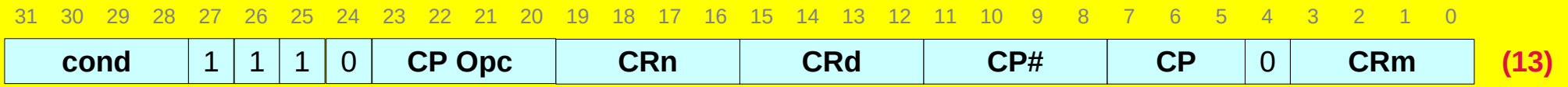
Coprocessor Instructions



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

<CP#>	Coprocessor number
<Cop1>	Coprocessor operation 1
<Cop2>	Coprocessor operation 2
CRd	Coprocessor Rd
CRn	Coprocessor Rn
CRm	Coprocessor Rm
Rn	Base Reg
Rd	Destination Reg

Coprocessor data operations

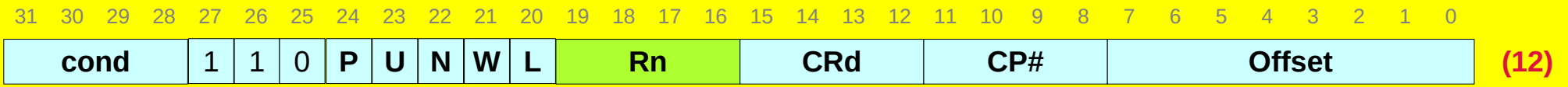


(13)

Coprocessor Data Operation

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

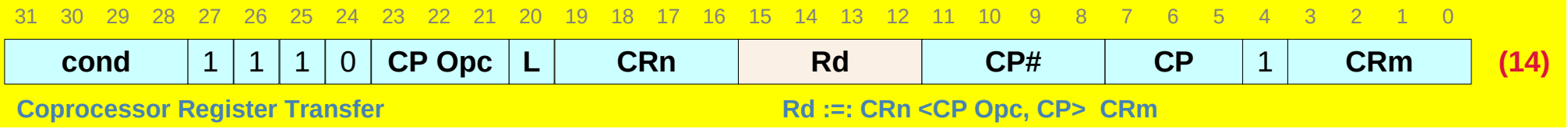
Coprocessor data transfers



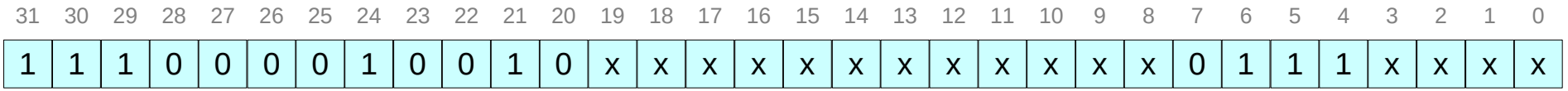
Coprocessor Data Transfer

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

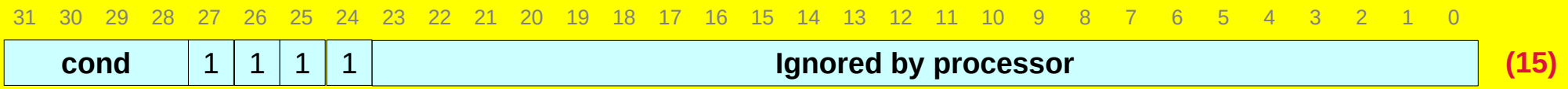
Coprocessor register transfer



Breakpoint instruction (BKPT)

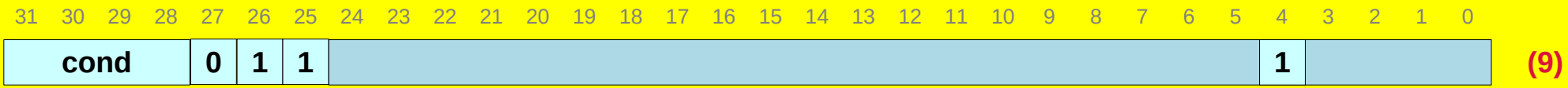


SWI (Software Interrupt)

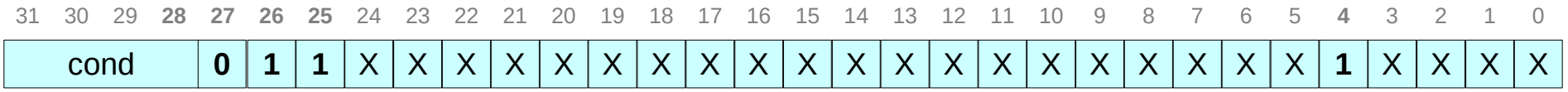


Software Interrupt SWI {<cond>} <24-bit immediate>

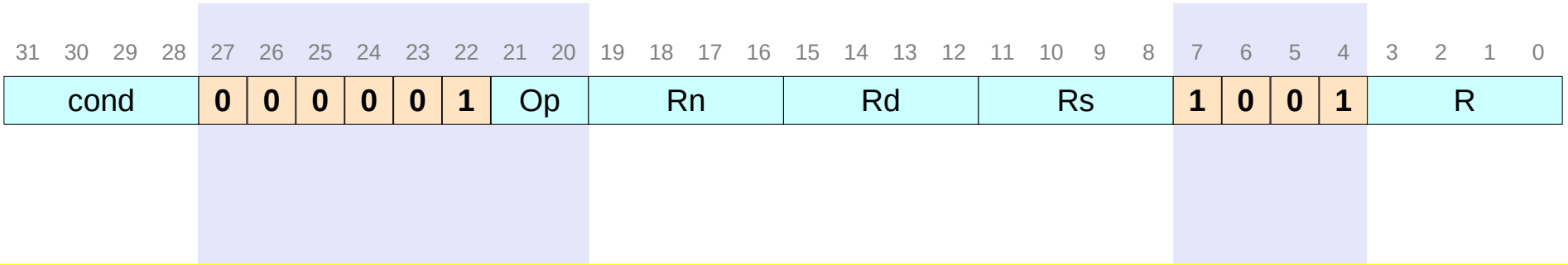
Undefined instrucitons

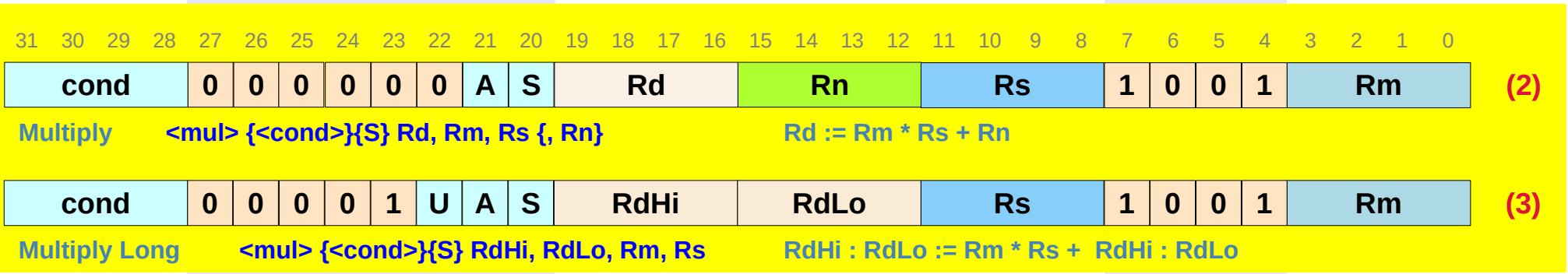


Undefined

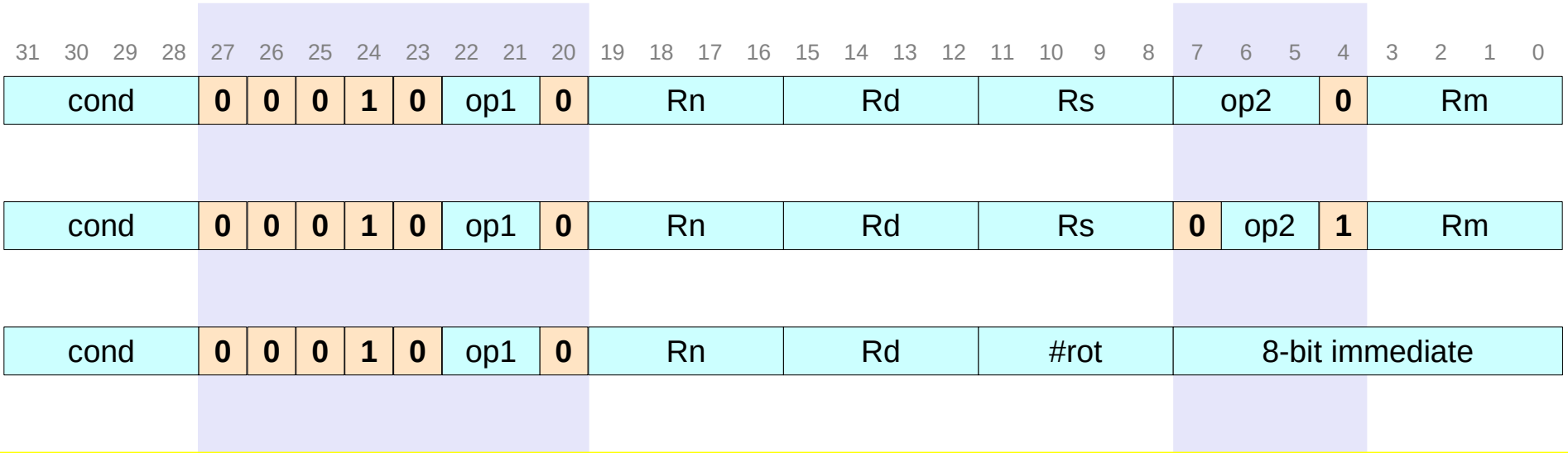


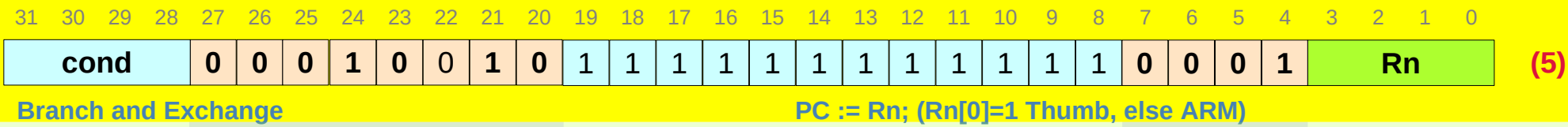
Unused arithmetic instructions



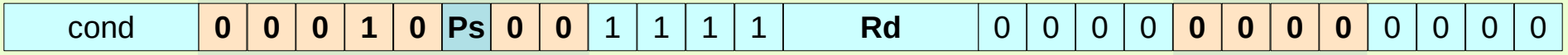


Unused control instructions

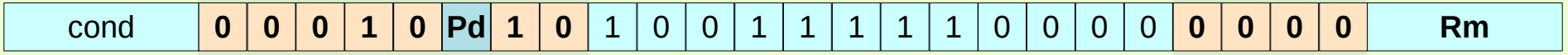




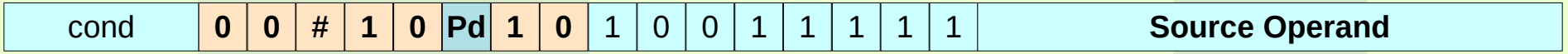
Branch and Exchange — **PC := Rn; (Rn[0]=1 Thumb, else ARM)**



MRS (Transfer PSR contents to a register)

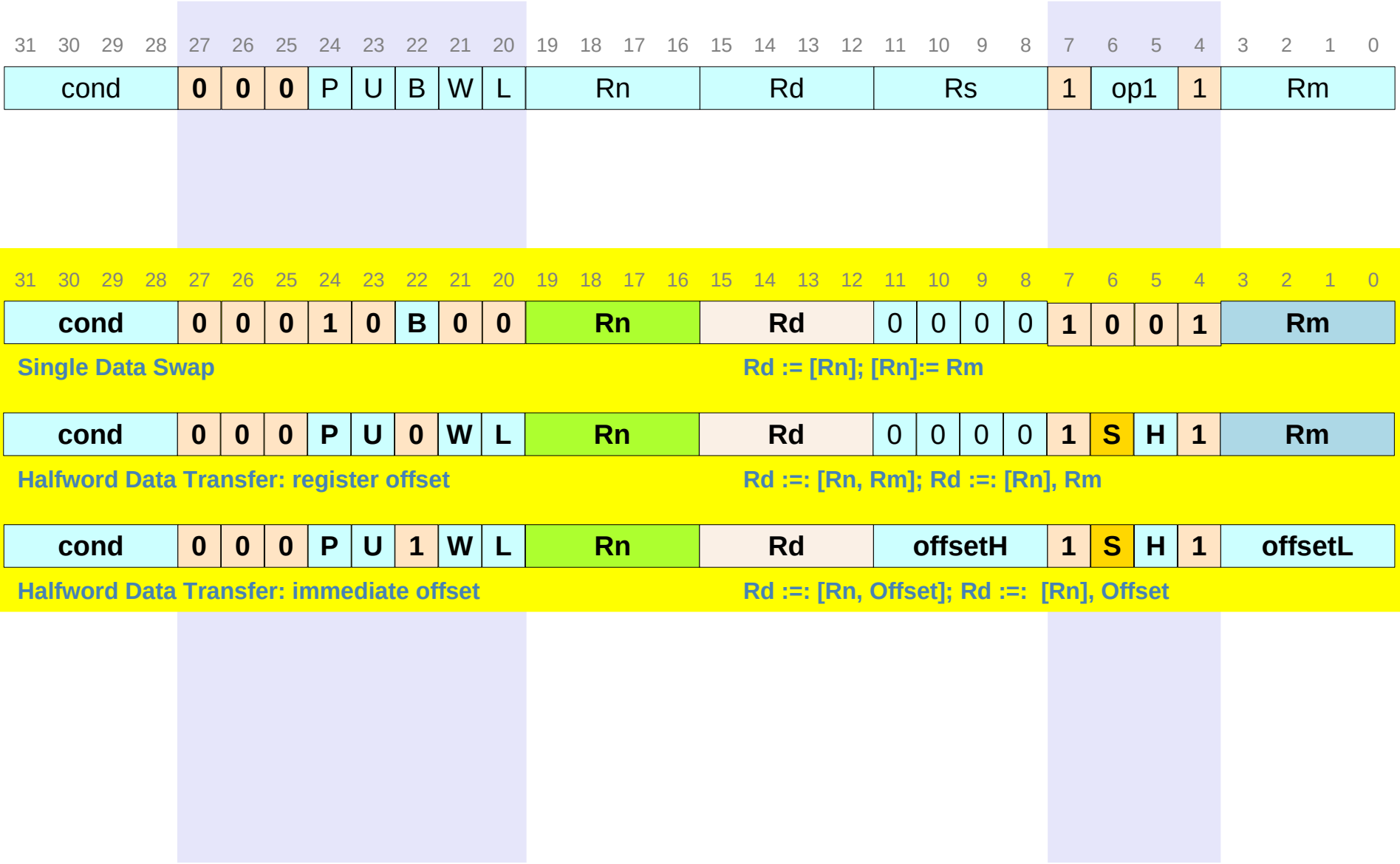


MSR (Transfer register contents to PSR)

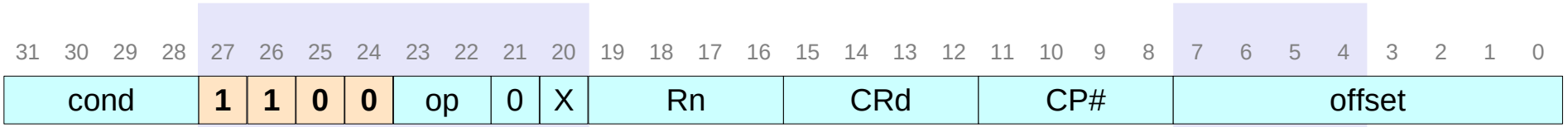


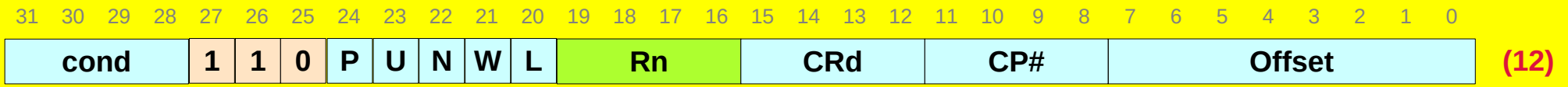
MSR (Transfer register contents or immediate value to PSR flag bits only)

Unused load/store instructions

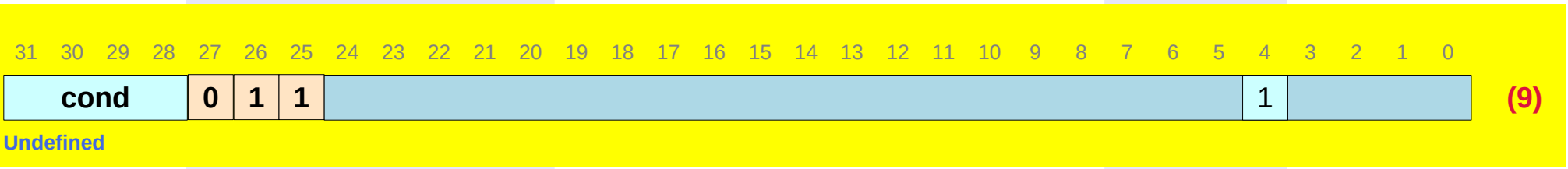


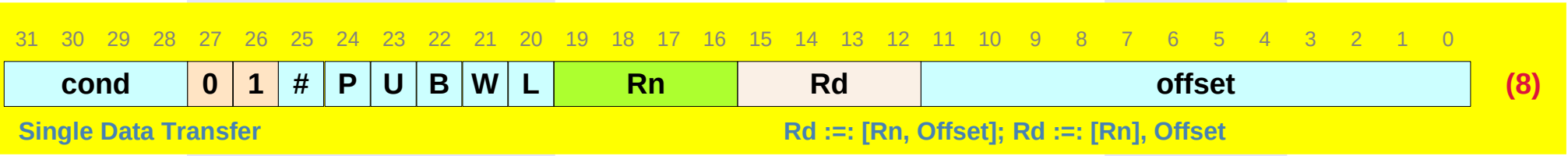
Unused coprocessor instructions



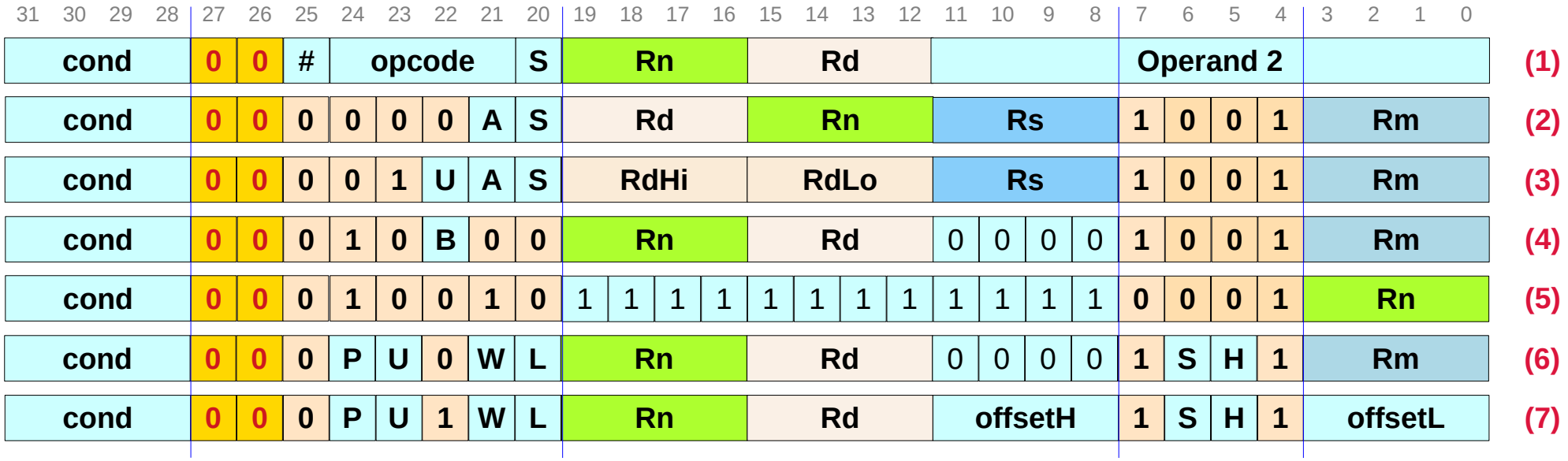


Undefined instruction space



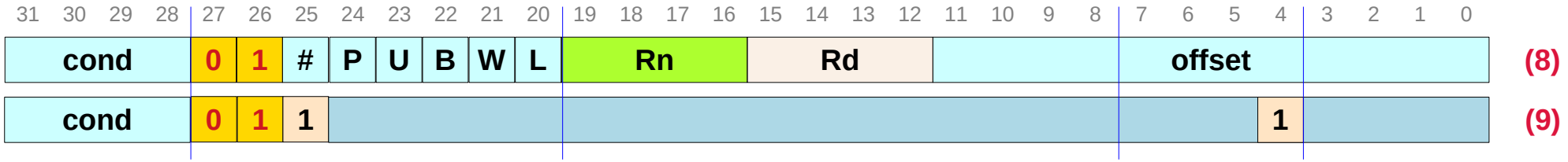


Decoding Instructions (1)



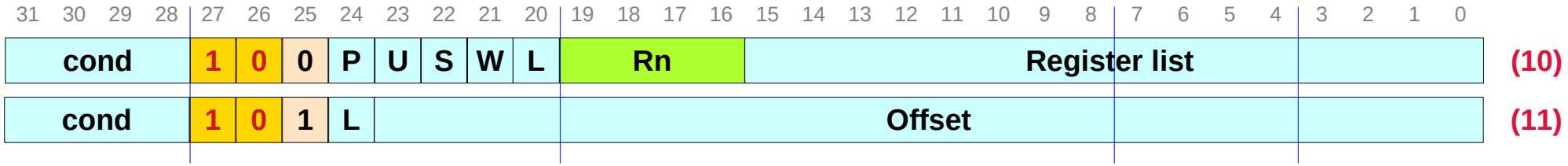
(1)	Data Processing / PSR Transfer	Rd := Rn <op> operand2 (shifted)
(2)	Multiply	Rd := Rm * Rs + Rn
(3)	Multiply Long	RdHi : RdLo := Rm * Rs + RdHi : RdLo
(4)	Single Data Swap	Rd := [Rn]; [Rn]:= Rm
(5)	Branch and Exchange	PC := Rn; (Rn[0]=1 Thumb, else ARM)
(6)	Halfword Data Transfer: register offset	Rd :=: [Rn, Rm]; Rd :=: [Rn], Rm
(7)	Halfword Data Transfer: immediate offset	Rd :=: [Rn, Offset]; Rd :=: [Rn], Offset

Decoding Instructions (2)



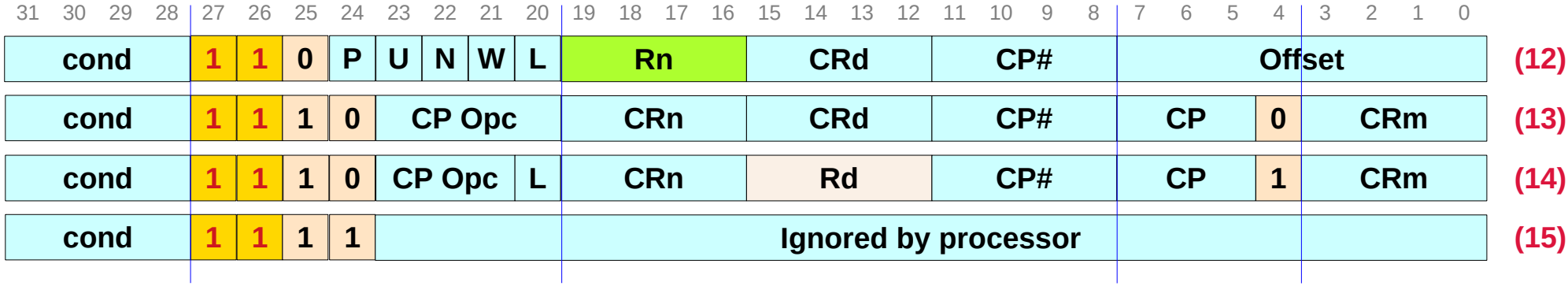
(8)	Single Data Transfer	Rd :=: [Rn, Offset]; Rd :=: [Rn], Offset
(9)	Undefined	

Decoding Instructions (3)



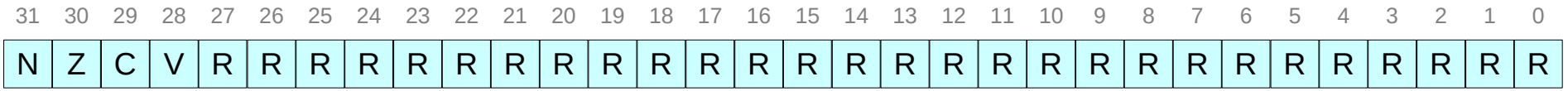
(10)	Block Data Transfer	[Rn, i] :=: {Register List}
(11)	Branch	R14 := PC+8; PC := Offset

Decoding Instructions (4)

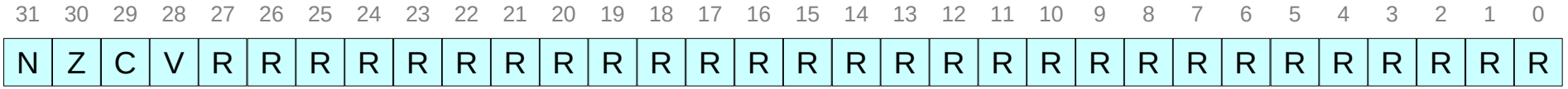


(12)	Coprocessor Data Transfer	CRd :=: [Rn,Offset]; CRd :=: [Rn], Offset
(13)	Coprocessor Data Operation	CRd :=: CRn <CP Opc, CP> CRm
(14)	Coprocessor Register Transfer	Rd :=: CRn <CP Opc, CP> CRm
(15)	Software Interrupt	

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