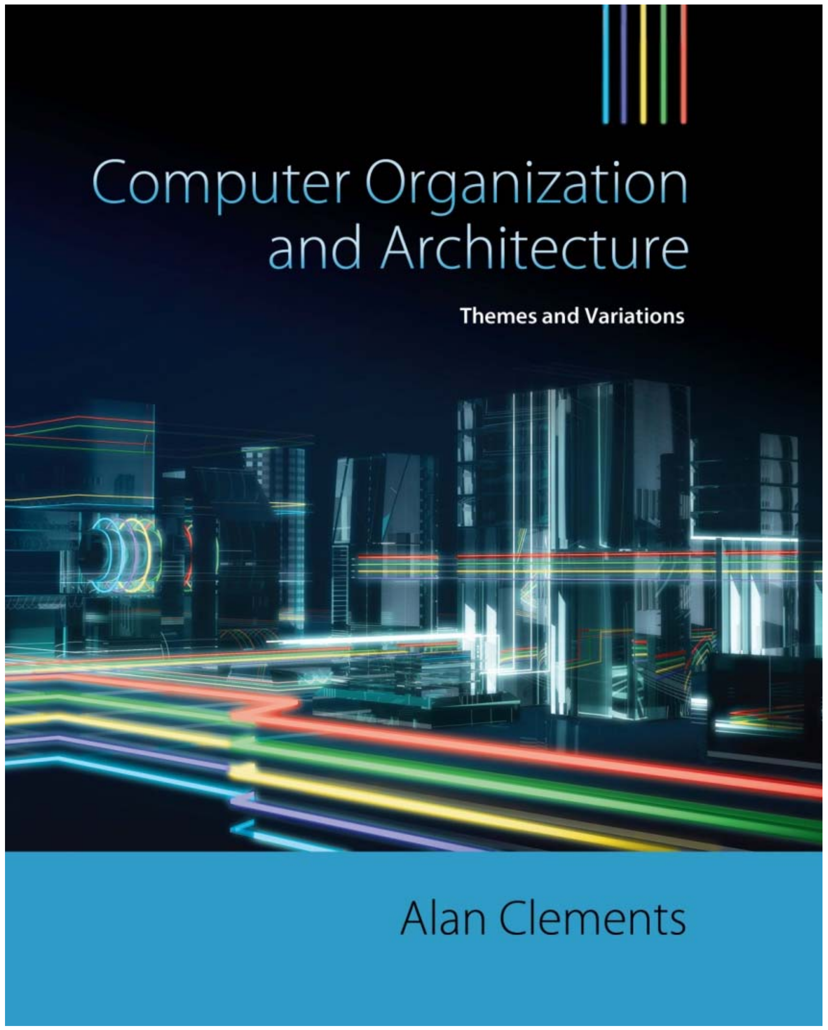
Part 0xF

CHAPTER 3

Architecture and Organization



CENGAGE Learning

ARM Assembly Instructions Summary

ADC{cond}{S} {Rd,}Rn,Op2	Add with carry	Rd ← Rn + Op2 + Carry
ADD{cond}{S} {Rd,}Rn,Op2	Add	Rd ← Rn + Op2
MLA{cond}{S} Rd, Rm,Rs,Rn	Multiply Accumulate	Rd ← (Rm × Rs) + Rn
MUL{cond}{S} Rd, Rm,Rs	Multiply	Rd ← Rm × Rs
MOV{cond}{S} Rd,Op2	Move register or constant	Rd ← Op2
NEG{cond}{S} Rd,Rn	Negate the value in a register	Rd ← - Rn
RSB{cond}{S}{ Rd,}Rn,Op2	Reverse Subtract	Rd ← Op2 - Rn
RSC{cond}{S} {Rd,}Rn,Op2	Reverse Subtract with Carry	Rd ← Op2 - Rn - 1 + Carry
SBC{cond}{S} {Rd,}Rn,Op2	Subtract with Carry	Rd ← Rn - Op2 - 1 + Carry
SUB{cond}{S} {Rd,}Rn,Op2	Subtract	Rd ← Rn - Op2

ARM Assembly Instructions Summary

AND{cond}{S} {Rd,}Rn,Op2	AND	Rd ← Rn *AND* Op2
BIC{cond}{S} {Rd,}Rn,Op2	Bit Clear	Rd ← Rn *AND NOT* Op2
ORR{cond}{S} {Rd,}Rn,Op2	OR	Rd ← Rn *OR* Op2
EOR{cond}{S} {Rd,}Rn,Op2	Exclusive OR	Rd ← Rn ⊕ Op2
MVN{cond}{S} Rd,Op2	Move not	Rd ← 0xFFFFFFFF ⊕ Op2

ARM Assembly Instructions Summary

CMN{cond} Rn,Op2	Compare Negative	CPSR flags $\leftarrow$ Rn + Op2
CMP{cond} Rn,Op2	Compare	CPSR flags $\leftarrow$ Rn - Op2
TEQ{cond} Rn,Op2	Test bitwise equality	CPSR flags $\leftarrow$ Rn $\oplus$ Op2
TST{cond} Rn,Op2	Test bits	CPSR flags $\leftarrow$ Rn *AND* Op

ARM Assembly Instructions Summary

B{cond} address	Branch	R15 $\leftarrow$ address
BL{cond} address	Branch with Link	R14 $\leftarrow$ R15, R15 $\leftarrow$ address

ARM Assembly Instructions Summary

ADR{cond}Rd,label	Load address	Rd ← The address of the label
STR{cond}{B} Rd,address	Store register to memory	[address] ← Rd
LDR{cond}{B} Rd,address	Load register from memory	Rd ← [address]
LDR{cond} Rd,=expr	Load a 32-bit immediate value	Rd ← expr
LDR{cond} Rd,=label	Load a 32-bit address	Rd ← The address of the label

ARM Assembly Instructions Summary

LDM{cond}{IA|IB|DA|DB}{cond} Rn{!},reglist	Load Multiple registers/Stack pop
LDM{cond}{FD|FA|ED|EA}{cond} Rn{!},reglist	Load Multiple registers/Stack pop
STM{cond}{IA|IB|DA|DB}}{cond} Rn{!},reglist	Store Multiple registers/Stack push
STM{cond}{FD|FA|ED|EA}}{cond} Rn{!},reglist	Store Multiple registers/Stack push

ARM Assembly Instructions Summary

ADC{cond}{S} {Rd,}Rn,Op2	Add with carry	Rd ← Rn + Op2 + Carry
ADD{cond}{S} {Rd,}Rn,Op2	Add	Rd ← Rn + Op2
AND{cond}{S} {Rd,}Rn,Op2	AND	Rd ← Rn *AND* Op2
ADR{cond}Rd,label	Load address	Rd ← The address of the label
B{cond} address	Branch	R15 ← address
BIC{cond}{S} {Rd,}Rn,Op2	Bit Clear	Rd ← Rn *AND NOT* Op2
BL{cond} address	Branch with Link	R14 ← R15, R15 ← address
CMN{cond} Rn,Op2	Compare Negative	CPSR flags ← Rn + Op2
CMP{cond} Rn,Op2	Compare	CPSR flags ← Rn - Op2
EOR{cond}{S} {Rd,}Rn,Op2	Exclusive OR	Rd ← Rn ⊕ Op2
LDM{cond}{IA|IB|DA|DB}{cond} Rn{!},reglist		Load Multiple registers/Stack pop
LDM{cond}{FD|FA|ED|EA}{cond} Rn{!},reglist		Load Multiple registers/Stack pop
LDR{cond}{B} Rd,address	Load register from memory	Rd ← [address]
LDR{cond} Rd,=expr	Load a 32-bit immediate value	Rd ← expr
LDR{cond} Rd,=label	Load a 32-bit address	Rd ← The address of the label
MLA{cond}{S} Rd, Rm,Rs,Rn	Multiply Accumulate	Rd ← (Rm × Rs) + Rn
MOV{cond}{S} Rd,Op2	Move register or constant	Rd ← Op2
MUL{cond}{S} Rd, Rm,Rs	Multiply	Rd ← Rm × Rs
MVN{cond}{S} Rd,Op2	Move not	Rd ← 0xFFFFFFFF ⊕ Op2
NEG{cond}{S} Rd,Rn	Negate the value in a register	Rd ← - Rn
NOP	No operation	No operation
ORR{cond}{S} {Rd,}Rn,Op2	OR	Rd ← Rn *OR* Op2
RSB{cond}{S}{ Rd,}Rn,Op2	Reverse Subtract	Rd ← Op2 - Rn
RSC{cond}{S} {Rd,}Rn,Op2	Reverse Subtract with Carry	Rd ← Op2 - Rn - 1 + Carry
SBC{cond}{S} {Rd,}Rn,Op2	Subtract with Carry	Rd ← Rn - Op2 - 1 + Carry
STM{cond}{IA|IB|DA|DB}}{cond} Rn{!},reglist		Store Multiple registers/Stack push
STM{cond}{FD|FA|ED|EA}}{cond} Rn{!},reglist		Store Multiple registers/Stack push
STR{cond}{B} Rd,address	Store register to memory	[address] ← Rd
SUB{cond}{S} {Rd,}Rn,Op2	Subtract	Rd ← Rn - Op2
TEQ{cond} Rn,Op2	Test bitwise equality	CPSR flags ← Rn ⊕ Op2
TST{cond} Rn,Op2	Test bits	CPSR flags ← Rn *AND* Op2

{S} ➔ Update condition flags if S present
{cond} ➔ *(to be omitted for unconditional execution)*
Refer to the table below for the meaning of the {cond} field.

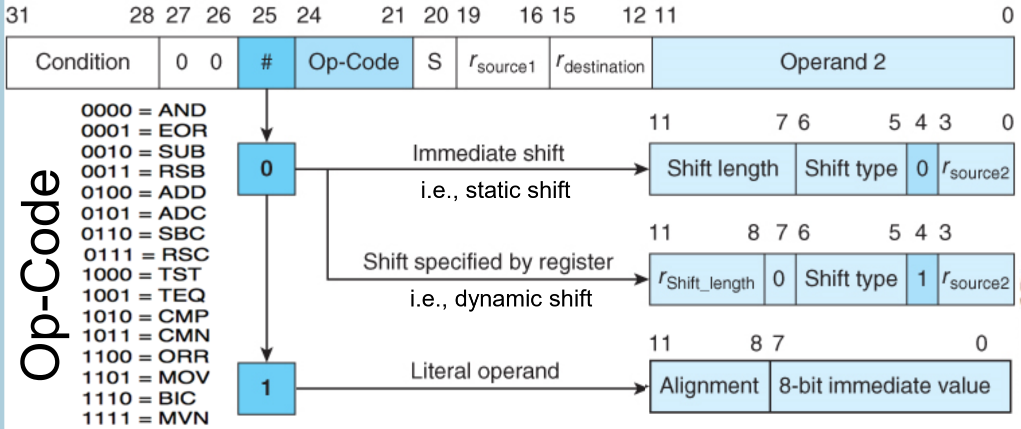
ARM Assembly Instructions Summary

Meaning of {condition} field

Encoding	Mnemonic	Branch on Flag Status	Execute on Condition
0000	EQ	Z set	Equal (i.e., zero)
0001	NE	Z clear	Not equal (i.e., not zero)
0010	CS	C set	Unsigned higher or same
0011	CC	C clear	Unsigned lower
0100	MI	N set	Negative
0101	PL	N clear	Positive or zero
0110	VS	V set	Overflow
0111	VC	V clear	No overflow
1000	HI	C set and Z clear	Unsigned higher
1001	LS	C clear or Z set	Unsigned lower or same
1010	GE	N set and V set, or N clear and V clear	Greater or equal
1011	LT	N set and V clear, or N clear and V set	Less than
1100	GT	Z clear and N set and V set, or Z clear and N clear and V clear	Greater than
1101	LE	Z set, or N set and V clear, or N clear and V set	Less than or equal
1110	AL		Always (default)
1111	NV		Never (reserved)

Instruction Encoding Formats

FIGURE 3.26 Encoding the ARM's data processing instructions



0000	= AND
0001	= EOR
0010	= SUB
0011	= RSB
0100	= ADD
0101	= ADC
0110	= SBC
0111	= RSC
1000	= TST
1001	= TEQ
1010	= CMP
1011	= CMN
1100	= ORR
1101	= MOV
1110	= BIC
1111	= MVN

Shift type

00	= logical left
01	= logical right
10	= arithmetic right
11	= rotate right

FIGURE 3.41 Encoding ARM's branch and branch-with-link instructions

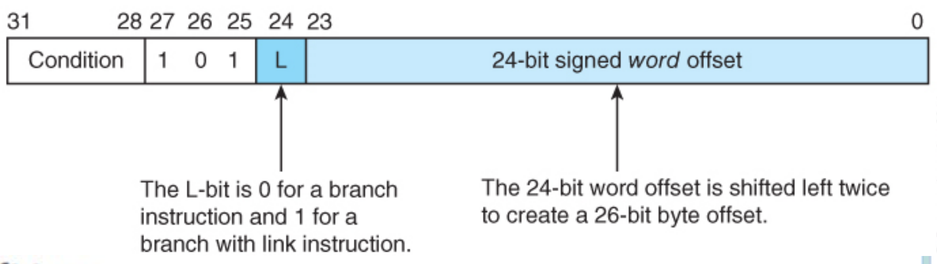

31 28 27 26 25 24 23 0
Condition
1 0 1
L
24-bit signed word offset
The L-bit is 0 for a branch instruction and 1 for a branch with link instruction.
The 24-bit word offset is shifted left twice to create a 26-bit byte offset.

FIGURE 3.39 Format of ARM's load and store instructions

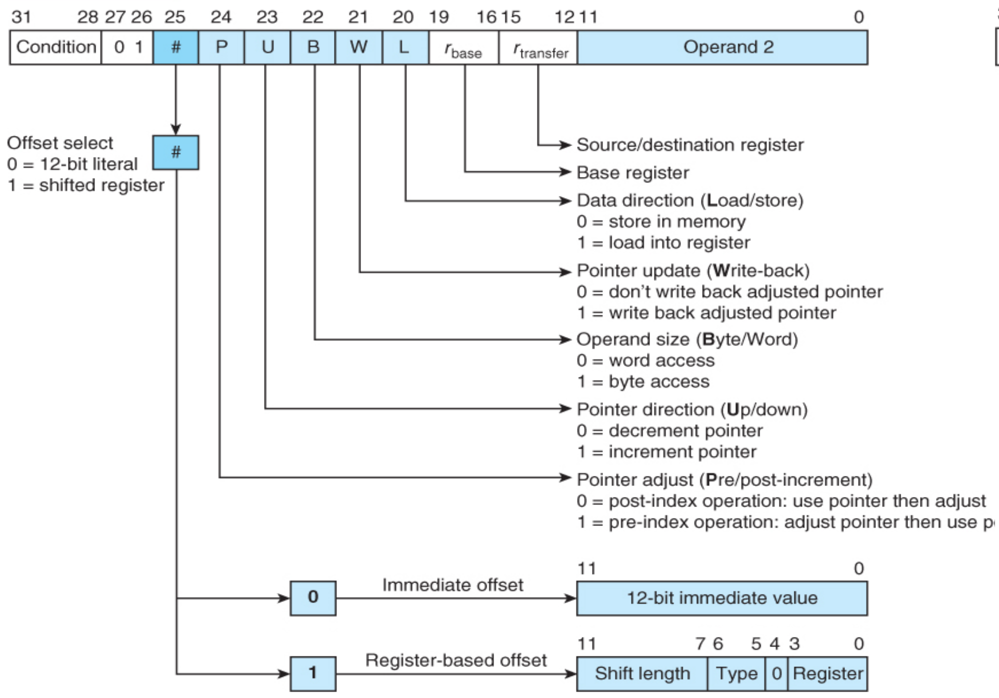
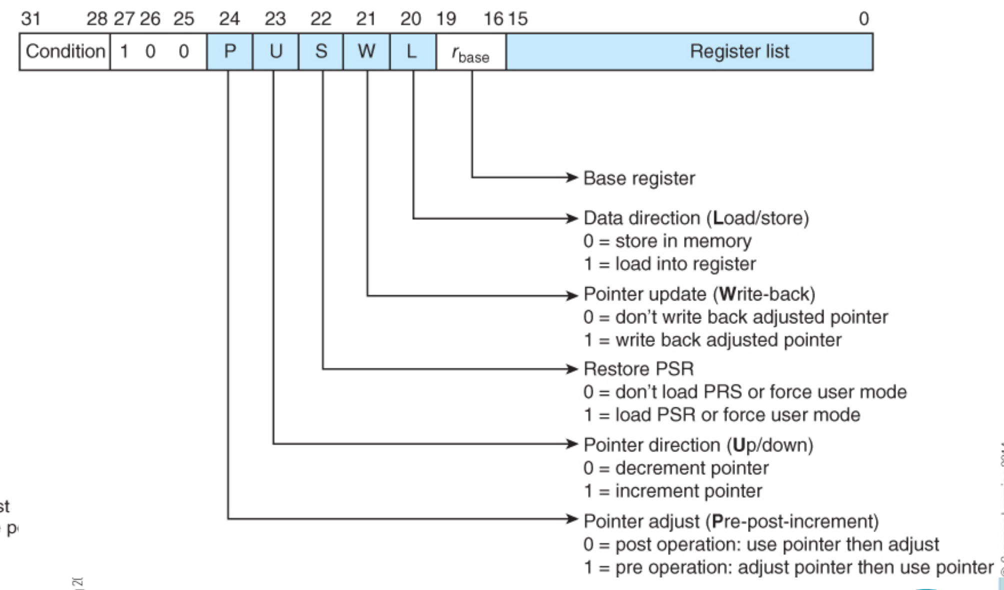


FIGURE 3.58 Encoding ARM's block move instructions


31 28 27 26 25 24 23 22 21 20 19 16 15 0
Condition
1 0 0
P
U
S
W
L
r_base
Register list
Base register
Data direction (Load/store)
0 = store in memory
1 = load into register
Pointer update (Write-back)
0 = don't write back adjusted pointer
1 = write back adjusted pointer
Restore PSR
0 = don't load PRS or force user mode
1 = load PSR or force user mode
Pointer direction (Up/down)
0 = decrement pointer
1 = increment pointer
Pointer adjust (Pre-post-increment)
0 = post operation: use pointer then adjust
1 = pre operation: adjust pointer then use pointer

Conversion Tables

2^0	=	1
2^1	=	2
2^2	=	4
2^3	=	8
2^4	=	16
2^5	=	32
2^6	=	64
2^7	=	128
2^8	=	256
2^9	=	512
2^{10}	=	1024 (Kilo)
2^{11}	=	2048
2^{12}	=	4096
2^{13}	=	8192
2^{14}	=	16384
2^{15}	=	32768
2^{16}	=	65536
2^{17}	=	131072
2^{18}	=	262144
2^{19}	=	524288
2^{20}	=	1048576 (Mega)

$(0)_{16}$	=	$(0)_{10}$	=	$(0000)_2$
$(1)_{16}$	=	$(1)_{10}$	=	$(0001)_2$
$(2)_{16}$	=	$(2)_{10}$	=	$(0010)_2$
$(3)_{16}$	=	$(3)_{10}$	=	$(0011)_2$
$(4)_{16}$	=	$(4)_{10}$	=	$(0100)_2$
$(5)_{16}$	=	$(5)_{10}$	=	$(0101)_2$
$(6)_{16}$	=	$(6)_{10}$	=	$(0110)_2$
$(7)_{16}$	=	$(7)_{10}$	=	$(0111)_2$
$(8)_{16}$	=	$(8)_{10}$	=	$(1000)_2$
$(9)_{16}$	=	$(9)_{10}$	=	$(1001)_2$
$(A)_{16}$	=	$(10)_{10}$	=	$(1010)_2$
$(B)_{16}$	=	$(11)_{10}$	=	$(1011)_2$
$(C)_{16}$	=	$(12)_{10}$	=	$(1100)_2$
$(D)_{16}$	=	$(13)_{10}$	=	$(1101)_2$
$(E)_{16}$	=	$(14)_{10}$	=	$(1110)_2$
$(F)_{16}$	=	$(15)_{10}$	=	$(1111)_2$

ASCII Table

‘0’	➔	0x30
‘1’	➔	0x31
‘2’	➔	0x32
...		
‘8’	➔	0x38
‘9’	➔	0x39
...		
‘A’	➔	0x41
‘B’	➔	0x42
‘C’	➔	0x43
‘D’	➔	0x44
‘E’	➔	0x45
‘F’	➔	0x46
...		
‘X’	➔	0x58
‘Y’	➔	0x59
‘Z’	➔	0x5A
...		
‘a’	➔	0x61
‘b’	➔	0x62
‘c’	➔	0x63
‘d’	➔	0x64
‘e’	➔	0x65
‘f’	➔	0x66
...		
‘x’	➔	0x78
‘y’	➔	0x79
‘z’	➔	0x7A