ARM Instruction References

Z. Gu

Fall 2025

Acknowledgement: Lecture slides based on Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C, University of Maine https://web.eece.maine.edu/~zhu/book/

Common ARM Instructions

Type of Instruction	ARM Assembly	Register Transfer Language Description
Memory Access (Load and Store)	LDR r4, Mem	[r4] ← [Mem] ; Mem is a global variable label
	STR r4, Mem	[Mem] ← [r4]
	LDR r4, [r3]	[r4] ← [[r3]] ; register indirect
	STR r4, [r3, #4]	[[r3] + 4] ← [r4] ; register indirect with offset
Move	MOV r4, r2	[r4] ← [r2]
	MOV r4, #10	[r4] ← 10 ; 8-bit literal, can be shifted
Load Address	ADR r4, Mem	[r4] ← load address of label Mem
Arithmetic Instruction	ADD r4, r2, r3	[r4] ← [r2] + [r3]
	MUL r4, r2, r3	[r4] ← [r2] * [r3] (32-bit product)
	SUB r4, r2, r3	[r4] ← [r2] - [r3]
Compare (sets condition codes)	CMP r4, r2	
Conditional Branch	BGT LABEL (BGE, BLT, BLE, BEQ, BNE)	Branch to LABEL based on condition codes
Unconditional Branch	B LABELAlways Branch to LABEL	

Type of Instruction	ARM Assembly	Register Transfer Language Description
ARM Logical Instructions	AND r4, r2, r3	[r4] ← [r2] (bit-wise AND) [r3]
	AND r4, r2, #0xFF000000	[r4] ← [r2] (bit-wise AND) FF000000
	ORR r4, r2, r3	[r4] ← [r2] (bit-wise OR) [r3]
	EOR r4, r2, r3	[r4] ← [r2] (bit-wise XOR) [r3]
	BIC r4, r2, r3	[r4] ← [r2] (bit-wise AND) (NOT [r3]) (clear bits set in r3)
	MOVN r4, r2	[r4] ← (NOT) [r2] (Flip all bits)
ARM Shift and Rotate Instructions	MOV r4, r5, LSL #3	r4 ← logical shift left r5 by 3 positions. (Shift in zeros)
	MOV r4, r5, LSL r6	r4 ← logical shift left r5 by the number of positions specified in register r6
	MOV r4, r5, LSR #3	r4 ←logical shift right r5 by 3 positions. (Shift in zeros)
	MOV r4, r5, ASR #3	r4 ← arithmetic shift right r5 by 3 positions. (Shift with sign-extend)
	MOV r4, r5, ROR #3	r4 ← rotate right r5 by 3 positions. (Circulate shift)
	AND r4, r5, r6, LSL #2	Shifts can operate on 3rd register operand of arithmetic or logical instruction, e.g., r4 ← r5 AND (logical shift left r6 by 8 positions)

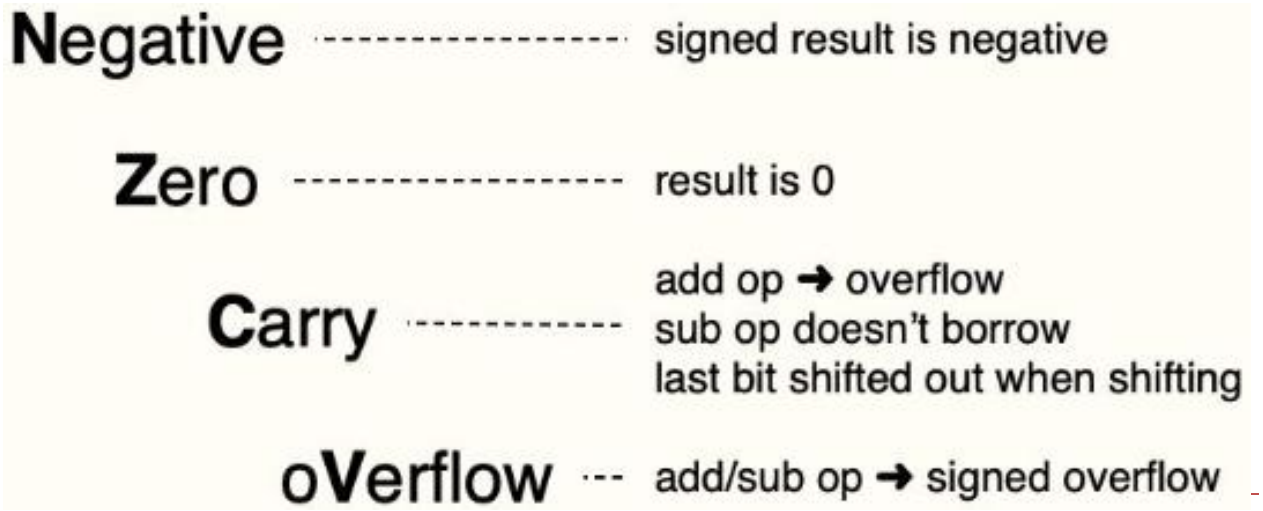
Ch6 Summary: Condition Codes

Suffix	Description	Flags tested
EQ	EQual	Z=1
NE	Not Equal	Z=0
CS/HS	Unsigned Higher or Same	C=1
CC/LO	Unsigned LOwer	C=0
MI	MInus (Negative)	N=1
PL	PLus (Positive or Zero)	N=0
VS	oVerflow Set	V=1
VC	oVerflow Cleared	V=0
HI	Unsigned HIgher	C=1 & Z=0
LS	Unsigned Lower or Same	C=0 or Z=1
GE	Signed Greater or Equal	N=V
LT	Signed Less Than	N!=V
GT	Signed Greater Than	Z=0 & N=V
LE	Signed Less than or Equal	Z=1 or N!=V
AL	ALways	

Note AL is the default and does not need to be specified

NZCV Flags in xPSR

- **N** (negative) – set (1) when a signed result is negative, otherwise cleared (0).
- **Z** (zero) – set (1) when a result is 0, otherwise cleared (0).
- **C** (carry) – set (1) when an add-based operation produces an overflow, when a subtract-based operation doesn't require a borrow; when shifting, holds the last bit that's been shifted out; otherwise cleared (0).
- **V** (overflow) – set (1) when an add- or subtract-based operation generates a signed overflow, otherwise cleared (0).



Ch6 Summary: Branch Instructions

	Instruction	Description	Flags tested
Unconditional Branch	**B** *label*	Branch to label	
Conditional Branch	**BEQ** *label*	Branch if **EQ**ual	`Z = 1`
	BNE *label*	Branch if **N**ot **E**qual	`Z = 0`
	BCS/BHS *label*	Branch if unsigned **H**igher or **S**ame	`C = 1`
	BCC/BLO *label*	Branch if unsigned **LO**wer	`C = 0`
	BMI *label*	Branch if **MI**nus (Negative)	`N = 1`
	BPL *label*	Branch if **PL**us (Positive or Zero)	`N = 0`
	BVS *label*	Branch if o**V**erflow **S**et	`V = 1`
	BVC *label*	Branch if o**V**erflow **C**lear	`V = 0`
	BHI *label*	Branch if unsigned **HI**gher	`C = 1 & Z = 0`
	BLS *label*	Branch if unsigned **L**ower or **S**ame	`C = 0 or Z = 1`
	BGE *label*	Branch if signed **G**reater or **E**qual	`N = V`
	BLT *label*	Branch if signed **L**ess **T**han	`N != V`
	BGT *label*	Branch if signed **G**reater **T**han	`Z = 0 & N = V`
	BLE *label*	Branch if signed **L**ess than or **E**qual	`Z = 1 or N = !V`

Ch6 Summary: Conditionally Executed

Add instruction	Condition	Flag tested
ADDEQ r3, r2, r1	Add if EQual	Add if Z = 1
ADDNE r3, r2, r1	Add if Not Equal	Add if Z = 0
ADDHS r3, r2, r1	Add if Unsigned Higher or Same	Add if C = 1
ADDLO r3, r2, r1	Add if Unsigned LOwer	Add if C = 0
ADDMI r3, r2, r1	Add if Minus (Negative)	Add if N = 1
ADDPL r3, r2, r1	Add if PLus (Positive or Zero)	Add if N = 0
ADDVS r3, r2, r1	Add if oVerflow Set	Add if V = 1
ADDVC r3, r2, r1	Add if oVerflow Clear	Add if V = 0
ADDHI r3, r2, r1	Add if Unsigned HIgher	Add if C = 1 & Z = 0
ADDLS r3, r2, r1	Add if Unsigned Lower or Same	Add if C = 0 or Z = 1
ADDGE r3, r2, r1	Add if Signed Greater or Equal	Add if N = V
ADDLT r3, r2, r1	Add if Signed Less Than	Add if N != V
ADDGT r3, r2, r1	Add if Signed Greater Than	Add if Z = 0 & N = V
ADDLE r3, r2, r1	Add if Signed Less than or Equal	Add if Z = 1 or N = !V

Ch8 ARM Procedure Call Standard

Register	Usage	Subroutine Preserved	Notes
r0	Argument 1 and return value	No	If return has 64 bits, then r0:r1 hold it. If argument 1 has 64 bits, r0:r1 hold it.
r1	Argument 2	No	
r2	Argument 3	No	If the return has 128 bits, r0-r3 hold it.
r3	Argument 4	No	If more than 4 arguments, use the stack
r4	General-purpose V1	Yes	Variable register 1 holds a local variable.
r5	General-purpose V2	Yes	Variable register 2 holds a local variable.
r6	General-purpose V3	Yes	Variable register 3 holds a local variable.
r7	General-purpose V4	Yes	Variable register 4 holds a local variable.
r8	General-purpose V5	Yes	Variable register 5 holds a local variable.
r9	Platform specific/V6	Yes	Usage is platform-dependent.
r10	General-purpose V7	Yes	Variable register 7 holds a local variable.
r11	General-purpose V8	Yes	Variable register 8 holds a local variable.
r12 (IP)	Intra-procedure-call register	No	It holds intermediate values between a procedure and the sub-procedure it calls.
r13 (SP)	Stack pointer	Yes	SP has to be the same after a subroutine has completed.
r14 (LR)	Link register	No	Receives return address on BL call to procedure
r15 (PC)	Program counter	N/A	Do not directly change PC