#### **ARM Instructions**

	A R.VI Instructions			
-		$\mathtt{ADD} cd \mathtt{S}^\dagger$	reg, reg, arg	add
		$\mathtt{SUB}cd\mathtt{S}$	reg, reg, arg	subtract
		$\mathtt{RSB}\mathit{cd}\mathtt{S}$	reg, reg, arg	subtract reversed operands
		$\mathtt{ADC}cd\mathtt{S}$	reg, reg, arg	add both operands and carry flag
	tic	$\mathtt{SBC}cd\mathtt{S}$	reg, reg, arg	subtract both operands and adds carry flag $-1$
	me	$\mathtt{RSC}\mathit{cd}\mathtt{S}$	reg, reg, arg	reverse subtract both operands and adds carry flag $-1$
	Arithmetic	$\mathtt{MUL}cd\mathtt{S}$	$reg_d, reg_m, reg_s$	multiply $reg_m$ and $reg_s$ , places lower 32 bits into $reg_d$
	$\mathbf{Ar}$	$\mathtt{MLA}cd\mathtt{S}$	$reg_d, reg_m, reg_s, reg_n$	places lower 32 bits of $reg_m \cdot reg_s + reg_n$ into $reg_d$
		$\mathtt{UMULL}\mathit{cd}\mathtt{S}$	$reg_{lo}, reg_{hi}, reg_m, reg_s$	multiply $reg_m$ and $reg_s$ place 64-bit unsigned result into $\{reg_{hi}, reg_{lo}\}$
		$\mathtt{UMLAL}cd\mathtt{S}$	$reg_{lo}, reg_{hi}, reg_m, reg_s$	$\text{place unsigned } reg_m \; \cdot \; reg_s \; + \; \{reg_{hi}, \; reg_{lo}\} \; \text{into} \; \{reg_{hi}, \; reg_{lo}\}$
			$reg_{lo}, reg_{hi}, reg_m, reg_s$	multiply $reg_m$ and $reg_s$ , place 64-bit signed result into $\{reg_{hi}, reg_{lo}\}$
		$\mathtt{SMLAL}cd\mathtt{S}$	$reg_{lo}, reg_{hi}, reg_m, reg_s$	place signed $reg_m \cdot reg_s + \{reg_{hi}, reg_{lo}\}$ into $\{reg_{hi}, reg_{lo}\}$
-	e	$\mathtt{AND}cd\mathtt{S}$	reg, reg, arg	bitwise AND
	itwis logic	ORR cd S	reg, reg, arg	bitwise OR
	Bitwise logic		reg, reg, arg	bitwise exclusive-OR
		BICcdS	$reg, reg_a, arg_b$	bitwise $reg_a$ AND (NOT $arg_b$ )
		$\mathtt{CMP}cd$	reg, arg	update flags based on subtraction
	Comp- arison	$\mathtt{CMN}cd$	reg, arg	update flags based on addition
	Col	$\mathtt{TST}\mathit{cd}$	reg, arg	update flags based on bitwise AND
		TEQcd	reg, arg	update flags based on bitwise exclusive-OR
Data mov	ement.	$\mathtt{MOV}cd\mathtt{S}$	reg, arg	copy argument
Dava III o		MVNcdS	reg, arg	copy bitwise NOT of argument
		LDR $cd$ $B$	reg, mem	loads word/ byte/ half from memory into a register
	Memory access	$\mathtt{STR}\mathit{cd}\hspace{.01in}\mathtt{B}$	reg, mem	stores word/ byte/ half to memory from a register
	1emor; access	$\mathtt{LDMcd}um$	$reg!,\ mreg$	loads into multiple registers
	a K	$\mathtt{STMcd}um$	$reg!,\ mreg$	stores multiple registers
		$\mathtt{SWP}cd\mathtt{B}$	$reg_d, reg_m, [reg_n]$	copies $reg_m$ to memory at $reg_n$ , old value at address $reg_n$ to $reg_d$
		Bcd	$imm_{24}$	branch to $imm_{24}$ words away
	lch	BLcd	$imm_{24}$	copy PC to LR, then branch
	Branch- ing	$BX\mathit{cd}$	reg	copy $reg$ to PC, and exchange instruction sets (T flag := $reg[0]$ )
		SWIcd	$imm_{24}$	software interrupt
			+ .	11.1 0 1 1 1 1 1 0 1 10 1/01 . \

 $<sup>^{\</sup>dagger}$  S = set condition flags

cd:	condition	code
cu.	Containion	couc

cd: condition code				
AL	or omitted	always	(ignored)	
ΕQ		equal	Z = 1	
NE		not equal	$\mathbf{Z} = 0$	
CS		carry set (same as HS)	C = 1	
CC		carry clear (same as LO)	C = 0	
ΜI		minus	N = 1	
PL		positive or zero	N = 0	
VS		overflow	V = 1	
VC		no overflow	V = 0	
HS		unsigned higher or same	C = 1	
LO		unsigned lower	C = 0	
ΗI		unsigned higher	$C = 1 \wedge Z = 0$	
LS		unsigned lower or same	$C = 0 \lor Z = 1$	
GΕ		signed greater than or equal	N = V	
LT		signed less than	$N \neq V$	
GΤ		signed greater than	$Z = 0 \wedge N = V$	
LΕ		signed less than or equal	$Z = 1 \vee N \neq V$	

## um: update mode

		-
FA /	IA	ascending, starting from reg
EA /	IB	ascending, starting from $reg + 4$
FD /	DB	descending, starting from reg
ED /	DA	descending, starting from $reg - 4$

# reg: register

R0	to	R15	register	according to number
SP			register	13
LR			register	14
РC			$\operatorname{register}$	15

### arg: right-hand argument

	0 0
$\#imm_8$	immediate on 8 bits, possibly rotated right
reg	register
reg, $shift$	register shifted by distance

## shift: shift register value

LSL	$\#imm_{5}$	shift left 0 to 31
LSR	$\#imm_{5}$	logical shift right 1 to 32
ASR	$\#imm_{5}$	arithmetic shift right 1 to 32
ROR	$\#imm_{5}$	rotate right 1 to 31
RRX		rotate carry bit into top bit
LSL	reg	shift left by register
LSR	reg	logical shift right by register
ASR	reg	arithmetic shift right by register
ROR	reg	rotate right by register

## mem: memory address

	· ·
[ $reg$ , $\#\pm imm_{12}$ ]	reg offset by constant
[ $reg$ , $\pm reg$ ]	reg offset by variable bytes
[ $reg_a$ , $\pm reg_b$ , $shift$ ]	$reg_a$ offset by shifted variable $reg_b$ †
[ $reg$ , $\#\pm imm_{12}$ ] !	update reg by constant, then access memory
[ $\mathit{reg}$ , $\pm \mathit{reg}$ ]!	update reg by variable bytes, then access memory
[ $reg$ , $\pm reg$ , $shift$ ] !	update $reg$ by shifted variable, then access memory $^{\dagger}$
[ $reg$ ] , $\#\pm imm_{12}$	access address reg, then update reg by offset
[ $reg$ ] , $\pm reg$	access address reg, then update reg by variable
[ $reg$ ] , $\pm reg$ , $shift$	access address $reg$ , then update $reg$ by shifted variable $^{\dagger}$

 $<sup>^{\</sup>ddagger}$  B = byte, can be replaced by H for half word(2 bytes)