ARM Instruction Set: Quick Reference Card

	Instruction	Function	Comments
Arithmetic	**add** Rd, Rn, Op2	Rd = Rn + Op2	Rd: registro destino
	sub Rd, Rn, Op2	Rd = Rn – Op2	Rn: registro fuente
	rsb Rd, Rn, Op2	Rd = Op2 – Rn	Op2: segundo operando
	cmp Rn, Op2	(Rn – Op2) → Flags	actualiza flags NZCV
	adc Rn, Rn, Op2	Rd = Rn + Op2 + FC	suma con flag de carry
	mul Rd, Rm, Rs	Rd = Rm * Rs	Rs,Rm: registros fuente
	umull RdLo, RdHi, Rm, Rs	RdHiLo = Rm * Rs (unsigned)	RdLo: 32 bits bajos
	smull RdLo, RdHi, Rm, Rs	RdHiLo = Rm * Rs (signed)	RdHi: 32 bits altos
	mla Rd, Rm, Rs, Rn	Rd = (Rm * Rs) + Rn	multiplicar i acumular
Logical	**mov** Rd, Op2	Rd = Op2	copia 32 bits
	mvn Rd, Op2	Rd = not Op2	NOT de todos los bits
	and Rd, Rn, Op2	Rd = Rn and Op2	AND bit a bit
	orr Rd, Rn, Op2	Rd = Rn or Op2	OR bit a bit
	eor Rd, Rn, Op2	Rd = Rn xor Op2	Exclusive Or
	bic Rd, Rn, Op2	Rd = Rn and not Op2	Bit Clear
	tst Rn, Op2	(Rn and Op2) → FlagZ	Test bits de Rn a 1 en Op2
Branch	**b** Dir	PC = Dir	**B**ranch (saltar)
	bl Dir	LR = PC - 4; PC = Dir	**B**ranch with **L**ink
	beq Dir	(A == B) **Después de cmp A, B**	**(Branch if) EQ**ual
	bne Dir	(A != B)	**N**ot **E**qual
	bhi Dir	(A > B) naturales	**HI**gher
	blo Dir	(A < B) naturales	**LO**wer
	bhs Dir	(A >= B) naturales	**H**igher or **S**ame
	bls Dir	(A <= B) naturales	**L**ower or **S**ame
	bgt Dir	(A > B) enteros	**G**reater **T**han
	blt Dir	(A < B) enteros	**L**ess **T**han
	bge Dir	(A >= B) enteros	**G**reater or **E**qual
	ble Dir	(A <= B) enteros	**L**ess or **E**qual
Load/Store	**ldr** Rd, [Md1]	Rd = Mem32[Md1]	**LoaD R**egister (word)
	str Rd, [Md1]	Mem32[Md1] = Rd	**ST**ore **R**egister (word)
	ldrb Rd, [Md1]	Rd = Up24(Mem8[Md1])	b: byte, h: half-word
	ldsb Rd, [Md2]	Rd = Ext24(Mem8[Md2]) (signed)	UpX: X bits altos a 0
	strb Rd, [Md1]	Mem8[Md1] = Low8(Rd)	ExtX: X bits altos a signo
	ldrh Rd, [Md2]	Rd = Up16(Mem16[Md2])	LowX: trunca X bits bajos
	ldsh Rd, [Md2]	Rd = Ext16(Mem16[Md2]) (signed)	MemX: acceso mem. X bits
	strh Rd, [Md2]	Mem16[Md2] = Low16(Rd)	Md1,Md2: *Addressing Modes*
Stack	**push** {reg_list(n)}	sp = sp – n*4; Mem32[sp + i*4] = reg_list(i);	apilar lista registros
	pop {reg_list(n)}	reg_list(i) = Mem32[sp + i*4]; sp = sp + n*4	desapilar lista registros

Registers		
r0-r12	data registers	datos en general
r13	Stack Pointer	puntero de pila (SP)
r14	Link Register	registro de enlace (LR)
r15	Program Counter	contador del programa (PC)

Conditional suffix		Predicated Instructions
Suffix	**Description**	**Examples**
EQ	Equal	***beq, addeq, subeq, moveq...***
NE	Not equal	***bne, addne, subne, movne...***
CS/HS	Carry Set/Higher or Same	***bhs, addhs, subhs, movhs...***
CC/LO	Carry Clear/LOwer	***bcc, addcc, subcc, movcc...***
MI	MInus (negative)	***bmi, addmi, submi, movmi...***
PL	PLus (positive or zero)	***bpl, addpl, subpl, movpl...***
VS	oVerflow Set	***bvs, addvs, subvs, movvs...***
VC	oVerflow Clear	***bvs, addvc, subvc, movvc...***
HI	HIgher	***bhi, addhi, subhi, movhi...***
LS	Lower or Same	***bls, addls, subls, movls...***
GE	Greater or Equal	***bge, addge, subge, movge...***
LT	Less Than	***blt, addlt, sublt, movlt...***
GT	Greater Than	***bgt, addgt, subgt, movgt...***
LE	Less or Equal	***ble, addle, suble, movle...***
AL	ALways	***bal, addal, subal, moval...***

Addressing Modes		
[Rb]	registro base	Md1,Md2
[Rb, #offset]	reg. base + offset	Md1,Md2
[Rb, Ri]	reg. base + reg. índice	Md1,Md2
[Rb,Ri,disp #nbits]	reg. base + reg. índice desplazado (núm.bits)	Md1

Displacement Type Op2 / Index register		
lsl	**L**ogical **S**hift **L**eft	desplazar n bits izquierda
lsr	**L**ogical **S**hift **R**ight	desplazar n bits derecha
asr	**A**ritmetic **S**hift **R**ight	desplazar n bits derecha con extensión de signo
ror	**RO**tate **R**ight	rotar n bits derecha con realimentación del bit de menos peso

santiago.romani@urv.cat

GAS Directive Set: Quick Reference Card

	Directive	Description	Comments
Sections	.bss	sección de datos no inicializados	no ocupa espacio en fichero objeto
	.data	sección de datos inicializados	sí ocupa espacio en fichero objeto
	.section *secname*	sección del programa	*secname* = {".text",".rodata", ".sbss",...}
	.text	sección de código	instrucciones de lenguaje máquina
Modif.asm	.align *pot2*	alineación dirección de ensamblado	siguiente dirección será múltiplo de 2^*pot2*
	.arm	siguiente código tipo ARM	32 bits por instrucción
	.thumb	siguiente código tipo THUMB	16 bits por instrucción
	.thumb_func	siguiente función tipo THUMB	implica código THUMB
Files	.end	final del fichero actual	no es obligatoria
	.global *symbol*	declara un símbolo como global	*symbol* será visible en otros ficheros fuente
	.include *fit.i*	incluye fichero definiciones	el fichero no debería incluir código ni datos
Data	.ascii *strings...*	inserta string sin centinela	vector con los códigos ASCII del string
	.asciz *strings...*	inserta string con centinela	el centinela es un 0 después del string
	.byte *values...*	inserta valores de 8 bits	naturales o Ca2
	.hword *values...*	inserta valores de 16 bits	naturales o Ca2
	.word *values...*	inserta valores de 32 bits	naturales o Ca2
	.space *num_bytes*	reserva un espacio en memoria	inserta *num_bytes* ceros
	.fill *num, size, val*	rellena un espacio en memoria	inserta *num* entradas de *size* bytes con *val*

Markers	Description	Example
@;	inicio de comentario (hasta final linea)	@;esto es un comentario
#	inicio de operando inmediato	mov r0, #0x3F00
,	separación de una secuencia de valores	.byte 042, -34, 0xFF, 'z'
:	declara una etiqueta (dirección memoria)	vector:
.L	antecede a una etiqueta local	.Lbucle:
=	referencia relativa a dir./valor de 32 bits	ldr r0, =vector
[]	delimitan direccionamiento a memoria	ldr r1, [r4, r12, lsl #2]
{}	delimitan lista de registros	push {r0,r1,r2,lr}
-	separador de rango de registros	pop {r0-r2,pc}
' '	delimitan un carácter ASCII	add r1, #'x'
" "	delimitan un string	.asciz "Esto es un string\n"

Data Type	Base	Prefix	Digits	Example
Entero decimal	10		0-9	34
Entero hexadecimal	16	0x o 0X	0-9, A-F	0x1AF7
Entero octal	8	0	0-7	042
Entero binario	2	0b o 0B	0-1	0B00100111

\meta	Description	ASCII
\0	código 0 (centinela strings)	0
\t	tabulador	9
\n	salto de línea	10
\"	comillas	34
\\	barra invertida	92
num	código *num* (octal)	num_8
\x*num*	código *num* (hexadecimal)	num_{16}

santiago.romani@urv.cat