ACCESO A MEMORIA

Mnemonic	Operands	Description		Flags
LDR {H,SH,B,SB}	Rd, [Rn]	Load Register with data	Rd = [Rn]	-
LDR {H,SH,B,SB}	Rd, [Rn,#offset]	Load Register with data	Rd = [Rn+off]	-
LDR {H,SH,B,SB}	Rd, [Rn],#offset	Load Register with data and increment	Rd = [Rn] Rn = Rn + off	-
LDR {H,SH,B,SB}	Rd, [Rn,#offset]!	Load Register with data and increment	Rn = Rn + off | (Rd = [Rn + off]) Rd = [Rn] | (Rn = Rn + off)	-
LDR {H,SH,B,SB}	Rd, [Rn,<Op2>]	Load Register with data	Rd = [Rn+<Op2>]	-
STR {H,B}	Rt, [Rn]	Store Register data	[Rn] = Rt	-
STR {H,B}	Rt, [Rn,#offset]	Store Register data	[Rn+off] = Rt	-
STR {H,B}	Rt, [Rn,<Op2>]	Store Register data	[Rn+<Op2>] = Rt	-
PUSH {Rt1,Rt2,....}	reglist	Push registers onto stack		-
POP {Rt1,Rt2,....}	reglist	Pop registers from stack		-
ADR	Rd, label	Set Rd equal to the address at label	Rd = <label> (Rd no SP ni PC)	-
MOV, MOVS	Rd, <Op2>	Set Rd equal to Op2	Rd = Operand2	N,Z,C
MOV {W,T}	Rd, #imm16	Set Rd equal to imm16	Rd[15:0] = imm16, Rd[31:16] = 0, imm16 range 0-65535	
MVN, MVNS	Rd, <Op2>	Set Rd equal to -Op2	Rd = 0xFFFFFFFF EOR Operand2	N,Z,C

SET DE INSTRUCCIONES BÁSICO

{ }	Encierra operandos opcionales.
Rd	Especifica el registro de destino.
	Si se omite Rd, el registro de destino es Rn.
Rn	Especifica el registro que contiene el primer operando.
Rm	Especifica el registro que contiene el segundo operando.
Rt	Especifica cualquier registro.
#n	Puede ser cualquier valor de 0 a 31, o de 1 a 32.
#offset	Cualquier valor de -255 a 4095.
label	Cualquier dirección dentro de la memoria ROM.
#imm12	Cualquier valor de 0 a 4095.
#imm16	Cualquier valor de 0 a 65535.
H	Dato de 16 bits sin signo.
SH	Dato de 16 bits con signo.
B	Dato de 8 bits sin signo.
SB	Dato de 8 bits con signo.
W	Parte baja [15:0].
T	Parte alta [31:16].

<Op2>	Es un segundo operando flexible
	#imm8
	Rm
	Rm, LSL Rs
	Rm, LSL #n 1<= n <=31
	Rm, LSR Rs
	Rm, LSR #n 1<= n <=32
	Rm, ASR Rs
	Rm, ASR #n 1<= n <=32
	Rm, ROR Rs
	Rm, ROR #n 1<= n <=31

Cortex M4

ZAS2018-I

ARITMÉTICAS Y LÓGICAS

Mnemonic	Operands	Description		Flags
ADD, ADDS	{Rd,} Rn,<Op2>	Add	Rd = Rn + Operand2	N,Z,C,V
ADD, ADDS	{Rd,} Rn ,#imm12	Add	Rd = Rn + imm12, imm12 range 0-4095	
SUB, SUBS	{Rd,} Rn,<Op2>	Subtract	Rd = Rn – Operand2	N,Z,C,V
SUB, SUBS	{Rd,} Rn ,#imm12	Subtract	Rd = Rn - imm12, imm12 range 0-4095	N,Z,C,V
RSB, RSBS	{Rd,} Rn,<Op2>	Reverse Subtract	Rd = Operand2 – Rn	N,Z,C,V
RSB, RSBS	{Rd,} Rn ,#imm12	Reverse Subtract	Rd = imm12 – Rn	N,Z,C,V
CMP	Rn, <Op2>	Compare	Update PSR flags on Rn – Operand2	N,Z,C,V
CMN	Rn, <Op2>	Compare Negative	Update PSR flags on Rn + Operand2	N,Z,C,V
MUL, MULS	{Rd,} Rn,Rm	Multiply, 32-bit result	Rd = Rn * Rm	N,Z
MLA	Rd, Rn,Rm, Ra	Multiply with Accumulate, 32-bit result	Rd = Ra + (Rn * Rm)	N,Z
MLS	Rd, Rn,Rm, Ra	Multiply and Subtract, 32-bit result	Rd = Ra – (Rn * Rm)	-
UDIV	{Rd,} Rn,Rm	Unsigned Divide	Rd = Rn / Rm	-
SDIV	{Rd,} Rn,Rm	Signed Divide	Rd = Rn / Rm	-

AND, ANDS	{Rd,} Rn,<Op2>	Logical AND	Rd = Rn & Operand2	N,Z,C
ORR, ORRS	{Rd,} Rn,<Op2>	Logical OR	Rd = Rn | Operand2	N,Z,C
ORN, ORNS	{Rd,} Rn,<Op2>	Logical OR NOT	Rd = Rn | (~ Operand2)	N,Z,C
EOR, EORS	{Rd,} Rn,<Op2>	Exclusive OR	Rd = Rn ^ Operand2	N,Z,C
BIC, BICS	{Rd,} Rn,<Op2>	Bit Clear	Rd = Rn & (~ Operand2)	N,Z,C
LSL, LSLS	Rd, Rm,Rs	Logical Shift Left (unsigned)	Rd = Rm << Rs (Same as MOV{S} Rd, Rm, LSL Rs)	N,Z,C
LSL, LSLS	Rd, Rm,#n	Logical Shift Left (unsigned)	Rd = Rm << n (Same as MOV{S} Rd, Rm, LSL n)	N,Z,C
LSR, LSRS	Rd, Rm,Rs	Logical Shift Right (unsigned)	Rd = Rm >> Rs (Same as MOV{S} Rd, Rm, LSR Rs)	N,Z,C
LSR, LSRS	Rd, Rm,#n	Logical Shift Right (unsigned)	Rd = Rm >> n (Same as MOV{S} Rd, Rm, LSR n)	N,Z,C
ASR, ASRS	Rd, Rm,Rs	Arithmetic Shift Right (signed)	Rd = Rm >> Rs (Same as MOV{S} Rd, Rm, ASR Rs)	N,Z,C
ASR, ASRS	Rd, Rm,#n	Arithmetic Shift Right (signed)	Rd = Rm >> n (Same as MOV{S} Rd, Rm, ASR n)	N,Z,C
ROR, RORS	Rd, Rm,Rs	Rotate Right	Rd = Rm >> Rs (Same as MOV{S} Rd, Rm, ROR Rs)	N,Z,C
ROR, RORS	Rd, Rm,#n	Rotate Right	Rd = Rm >> #n (Same as MOV{S} Rd, Rm, ROR n)	N,Z,C
RRX, RRXS	Rd, Rm	Rotate Right with Extend (only 1 bit)	Rd = RRX(Rm) (Same as MOV{S} Rd, Rm, RRX)	N,Z,C

CONTROL DE FLUJO

Mnemonic	Operands	Description		Flags
B	label	Branch to label	PC = label. label is this instruction ±32MB	-
BX	Rm	Branch indirect to location specified by Rm	PC = Rm	-
BL	label	Branch to subroutine at label	LR = address of next instruction, PC = label. label is this instruction ±32MB.	-
BLX	label o Rm	Branch to subroutine indirect specified by Rm o label	LR = address of next instruction, PC = Rm[31:1]. O PC = label	-

Mnemonic	Operands	Meaning	Condition Flags
BEQ	label o Rm	Equal	Z = 1
BNE	label o Rm	Not equal	Z = 0
BCS	label o Rm	Higher or same, unsigned	C = 1
BHS	label o Rm	Higher or same, unsigned	C = 1
BCC	label o Rm	Lower, unsigned	C = 0
BLO	label o Rm	Lower, unsigned	C = 0
BMI	label o Rm	Negative	N = 1
BPL	label o Rm	Positive or zero	N = 0
BVS	label o Rm	Overflow	V = 1
BVC	label o Rm	No overflow	V = 0
BHI	label o Rm	Higher, unsigned	C = 1 and Z = 0
BLS	label o Rm	Lower or same, unsigned	C = 0 or Z = 1
BGE	label o Rm	Greater than or equal, signed	N = V
BLT	label o Rm	Less than, signed	N != V
BGT	label o Rm	Greater than, signed	Z = 0 and N = V
BLE	label o Rm	Less than or equal, signed	Z = 1 and N != V

INSTRUCCIONES ESPECIALES

Mnemonic	Operands	Description		Flags
CPSID	i	Change Processor State, Disable Interrupts		-
CPSIE	i	Change Processor State, Enable Interrupts		-