Instruction_Decoder_TruthTable.txt

instruction encoding 15 … 0	assembler	A1	A2	A3	always	PCS	MemToReg	MemW	ALUSrc	ImmSrc	RegW	RegSrc	ALUControl	FlagW	Example			
0 0 0 0 0 0 0 0 0 n n n d d d	MOVS <Rd>,<Rn>	Rn		Rd	1	0	0	0	0	xx	1	000	011	10	000c	movs	r4, r1	
0 0 0 1 1 0 0 m m m n n n d d d	ADDS <Rd>,<Rn>,<Rm>	Rn	Rm	Rd	1	0	0	0	0	xx	1	000	100	11	18d5	adds	r5, r2, r3	
0 0 0 1 1 0 1 m m m n n n d d d	SUBS <Rd>,<Rn>,<Rm>	Rn	Rm	Rd	1	0	0	0	0	xx	1	000	101	11	1ace	subs	r6, r1, r3	
0 0 1 0 0 d d d i i i i i i i i	MOVS <Rd>, #<imm8>			Rd	1	0	0	0	1	00	1	100	000	10	2156	movs	r1, #86	
0 0 1 1 0 dn dr dr i i i i i i i i	ADDS <Rdn>,#<imm8>	Rdn		Rdn	1	0	0	0	1	00	1	100	100	11	3203	adds	r2, #3	
0 0 1 1 1 dn dr dr i i i i i i i i	SUBS <Rdn>,#<imm8>	Rdn		Rdn	1	0	0	0	1	00	1	100	101	11	3b10	subs	r3, #0x10	
1 0 0 1 1 t t t i i i i i i i i	LDR <Rt>,[<SP>,#<imm8>]	SP		Rt	1	0	1	0	1	01	1	110	100	00	9f01	ldr	r7, [sp, #4]	
1 0 0 1 0 t t t i i i i i i i i	STR <Rt>,[<SP>,#<imm8>]	SP		Rt	1	0	0	1	1	01	0	110	100	00	9401	str	r4, [sp, #4]	
1 1 0 1 c c c c i i i i i i i i	Bcc #<simm8>	PC			0	1	0	0	1	10	0	001	100	00	d0fe	beq	-2	self loop, not taken
1 1 1 0 0 i i i i i i i i i i i	B #<simm11>	PC			1	1	0	0	1	11	0	001	100	00	e7fe	b	-2	self loop, unconditional

1 for unconditional execution

1 for computed PC from Mem

stor e to Mem

1 for immediate

writer to register

FlagW1 for updating N and Z (Flags3:2), and FlagW0 for updating C and V (Flags1:0).

20fb

ALUControl

0	0	0	B
0	0	1	-B
0	1	0	0
0	1	1	A
1	0	0	A+B
1	0	1	A-B
1	1	0	A&B
1	1	1	A|B

ImmSrc

0	0	imm8
0	1	imm8 << 2
1	0	simm8 << 1
1	1	simm11 << 1

RegSrc

	0 0	use RA1 for A1
	0 1	use PC for A1
	1 0	use SP for A1
0		use individual bits for RAi
1		use bits 10..8 for all RAi

D-Flipflop (MS)
(used for all registers)

D	WE	R	CLK	Q'	
0	0	0	x	Q	no change
1	0	0	x	Q	no change
x	x	1	x	0	static reset
0	1	0	↑	0	edge triggered write
1	1	0	↑	1	edge triggered write

ROM

A	D	
a	ROM[a]	static read

RAM

C	WE	OE	A	Din	Dout	
x	x	1	a		RAM[a]	static read
↑	1	x	a	RAM[a] = Din		rising edge triggered write

MUX Placement	Purpose
ALU Input A	PC or Register File output
ALU Input B	Register File output or Immediate value
Write-back Stage	ALU output or Data Memory output
Memory Address Selection	PC or ALU output
PC Update Selection	PC + 4 or Branch Target Address

conditional branches

1 1 0 1 c c c c i i i i i i i i

Branch conditions

Code	Condition	Meaning	Status of Flags
0 0 0 0	EQ	Equal	Z==1
0 0 0 1	NE	Not Equal	Z==0
0 0 1 0	CS or HS	Unsigned Higher or Same (or Carry Set)	C==1
0 0 1 1	CC or LO	Unsigned Lower (or Carry Clear)	C==0
0 1 0 0	MI	Negative (or Minus)	N==1
0 1 0 1	PL	Positive (or Plus)	N==0
0 1 1 0	VS	Signed Overflow	V==1
0 1 1 1	VC	No signed Overflow	V==0
1 0 0 0	HI	Unsigned Higher	(C==1) && (Z!=0)
1 0 0 1	LS	Unsigned Lower or same	(C==0) || (Z==0)
1 0 1 0	GE	Signed Greater Than or Equal	N==V
1 0 1 1	LT	Signed Less Than	N!=V
1 1 0 0	GT	Signed Greater Than	(Z==0) && (N==V)
1 1 0 1	LE	Signed Less Than or Equal	(Z==1) || (N!=V)
1 1 1 0	AL	Always executed	true
1 1 1 1	NV	Never executed	false

A B

data processing (not implemented yet)

0 1 0 0 0 0 c c c c m m m dn dr dn

0 0 0 0	ANDS	bitwise (logical) AND	Rdn := Rdn & Rm
0 0 0 1	EORS	bitwise (logical) exclusive OR	Rdn := Rdn ^ Rm
0 0 1 0	LSLS	logical shift left (unsigned)	Rdn := Rdn << Rm
0 0 1 1	LSRS	logical shift right (unsigned)	Rdn := Rdn >> Rm
0 1 0 0	ASRS	arithmetic shift right (signed)	Rdn := Rdn ASR Rm
0 1 0 1	ADCS	add with carry	Rdn := Rdn + Rm + C-bit
0 1 1 0	SBCS	sub with carry	Rdn := Rdn - Rm - NOT C-bit
0 1 1 1	RORS	rotate right	Rdn := Rdn ROR Rm
1 0 0 0	TST	test (like AND), no result but flags	NZCV := NZCV(Rdn & Rm)
1 0 0 1	RSBS	reverse subtract (from 0), NEG	Rdn = -Rm
1 0 1 0	CMP	compare (like SUB), no result but flags	NZCV := NZCV(Rdn - Rm)
1 0 1 1	CMN	compare negative (like RSB)	NZCV := NZCV(Rdn + Rm)
1 1 0 0	ORRS	bitwise (logical) OR	Rdn := Rdn | Rm
1 1 0 1	MULS	multiply	Rdn := Rm * Rdn
1 1 1 0	BICS	bitwise bit clear	Rdn := Rdn & ~Rm
1 1 1 1	MVNS	bitwise (logical) NOT	Rdn := ~Rm