Description	UAL code				Bits																Flags			
	construction	operandes			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	C	V	N	Z
Logical**S**hift**L**eft	LSL S	<Rd>,	<Rm>,	#<imm5>	0	0	0	0	0	imm5					Rm			Rd			x		x	x
Logical**S**hift**R**ight	LSR S	<Rd>,	<Rm>,	#<imm5>	0	0	0	0	1	imm5					Rm			Rd			x		x	x
Arithmetic**S**hift**R**ight	ASR S	<Rd>,	<Rm>,	#<imm5>	0	0	0	1	0	imm5					Rm			Rd			x		x	x
Shift.add.sub.mov																								
Add register	ADD S	<Rd>,	<Rn>,	<Rm>	0	0	0	1	1	0	0	Rm			Rn			Rd			x	x	x	x
Sub stract register	SUB S	<Rd>,	<Rn>,	<Rm>	0	0	0	1	1	0	1	Rm			Rn			Rd			x	x	x	x
Add 3-bit immediate	ADD S	<Rd>,	<Rn>,	<#imm3>	0	0	0	1	1	1	0	imm3			Rn			Rd			x	x	x	x
Sub stract 3-bit immediate	SUB S	<Rd>,	<Rn>,	<#imm3>	0	0	0	1	1	1	1	imm3			Rn			Rd			x	x	x	x
Move	MOV S	<Rd>,	# <imm8>		0	0	1	0	0	Rd					imm8								x	x
Compare Immediate	CMP	<Rd>,	# <imm8>		0	0	1	0	1	Rd					imm8						x	x	x	x
Add Immediate	ADD S	<Rd>,	# <imm8>		0	0	1	1	0	Rd					imm8						x	x	x	x
Sub Immediate	SUB S	<Rd>,	# <imm8>		0	0	1	1	1	Rd					imm8						x	x	x	x
Data processing											op code													
Bitwise **AND**	AND S	<Rdn>,	<Rm>		0	1	0	0	0	0	0	0	0	0	Rm			Rdn			0		x	x
Exclusive **OR**	EOR S	<Rdn>,	<Rm>		0	1	0	0	0	0	0	0	0	1	Rm			Rdn			0		x	x
Logical**S**hift**L**eft	LSL S	<Rdn>,	<Rm>		0	1	0	0	0	0	0	0	1	0	Rm			Rdn			x		x	x
LogicalShift**R**ight	LSR S	<Rdn>,	<Rm>		0	1	0	0	0	0	0	0	1	1	Rm			Rdn			x		x	x
Arithmetic**S**hift**R**ight	ASR S	<Rdn>,	<Rm>		0	1	0	0	0	0	0	1	0	0	Rm			Rdn			x		x	x
Add with **C**arry	ADD S	<Rdn>,	<Rm>		0	1	0	0	0	0	0	1	0	1	Rm			Rdn			x	x	x	x
Sub stract with **C**arry	SUB S	<Rdn>,	<Rm>		0	1	0	0	0	0	0	1	1	0	Rm			Rdn			x	x	x	x
Rotate **R**ight	ROR S	<Rdn>,	<Rm>		0	1	0	0	0	0	0	1	1	1	Rm			Rdn			x		x	x
Set flag on bitwise and	TST	<Rn>,	<Rm>		0	1	0	0	0	0	1	0	0	0	Rm			Rn			0		x	x
Reverse **S**ub strucs from 0	RSB S	<Rd>,	<Rn>,	#0	0	1	0	0	0	0	1	0	0	1	Rn			Rd			x	x	x	x
Compare **R**egisters	CMP	<Rn>,	<Rm>		0	1	0	0	0	0	1	0	1	0	Rm			Rn			x	x	x	x
Compare **N**egative	CMN	<Rn>,	<Rm>		0	1	0	0	0	0	1	0	1	1	Rm			Rn			x	x	x	x
Logical **OR**	ORR S	<Rdn>,	<Rm>		0	1	0	0	0	0	1	1	0	0	Rm			Rdn			0		x	x
Multiply two Registers	MUL S	<Rdm>,	<Rn>,	<Rdm>	0	1	0	0	0	0	1	1	0	1	Rn			Rdm					x	x
Bit **C**lear	BIC S	<Rdn>,	<Rm>		0	1	0	0	0	0	1	1	1	0	Rm			Rdn			0		x	x
Bitwise NOT	MVN S	<Rd>,	<Rm>		0	1	0	0	0	0	1	1	1	1	Rm			Rd			0		x	x
Load/Store									op code															
Store **R**egister	STR	<Rt>	[SP,	# <offset>]	1	0	0	1	0	Rt					imm8									
Load **R**egister	LDR	<Rt>	[SP{,	# <offset>}]	1	0	0	1	1	Rt					imm8									
Miscellaneous 16-bit instructions																								
Add immediate to SP	ADD	SP,	{SP},	#<offset>	1	0	1	1	0	0	0	0	0	imm7										
Sub stract immediate from SP	SUB	SP,	{SP},	#<offset>	1	0	1	1	0	0	0	0	1	imm7										
Conditionnal Branch	B																							
égalité	BEQ	<label>			1	1	0	1	0	0	0	0	imm8								Z == 1			
différence	BNE	<label>			1	1	0	1	0	0	0	1	imm8								Z == 0			
retenue	BCS	<label>			1	1	0	1	0	0	1	0	imm8								C == 1			
pas de retenue	BCC	<label>			1	1	0	1	0	0	1	1	imm8								C == 0			
négatif	BMI	<label>			1	1	0	1	0	1	0	0	imm8								N == 1			
positif ou nul	BPL	<label>			1	1	0	1	0	1	0	1	imm8								N == 0			
dépassement de capacité	BVS	<label>			1	1	0	1	0	1	1	0	imm8								V == 1			
pas de dépacement de capacité	BVC	<label>			1	1	0	1	0	1	1	1	imm8								V == 0			
supérieur (non signé)	BHI	<label>			1	1	0	1	1	0	0	0	imm8								C == 1 et Z == 0			
inférieur ou égal (non signé)	BLS	<label>			1	1	0	1	1	0	0	1	imm8								C == 0 ou Z == 1			
superieur ou égal (signé)	BGE	<label>			1	1	0	1	1	0	1	0	imm8								N == V			
inférieur (signé)	BLT	<label>			1	1	0	1	1	0	1	1	imm8								N != V			
supérieur (signé)	BGT	<label>			1	1	0	1	1	1	0	0	imm8								Z == 0 et N == V			
inférieur ou égal (signé)	BLE	<label>			1	1	0	1	1	1	0	1	imm8								Z == 1 ou N != V			
toujours vrai	B ou BAL	<label>			1	1	0	1	1	1	1	0	imm8											
toujours faux	BF	<label>			1	1	1	1	1	1	1	1	imm8											
branche non conditionnelle	B	<label>			1	1	1	0	0	imm11														