Registers

Register	Description
R0	16 bit, General Purpose
R1	16 bit, General Purpose
R2	16 bit, General Purpose
R3	16 bit, General Purpose
R4	16 bit, General Purpose
R5	16 bit, General Purpose
R6	16 bit, General Purpose
SP	16 bit, Stack Pointer
PC	16 bit, Program Pointer
Status I V S C Z AC AZ	Status Register I: Interrupt flag V, S, C, Z: Compare flags AC, AZ: Arithmetic flags (SR refers to the compare flags subset) (AR refers to the arithmetic flags subset)
All registers are 16 bit. ALU operations are 16 bit. 8 bit operations are not natively supported except for extended loads and truncated stores. The Status Register is split into two sets, the SR set and the AR set. Instructions may update SR only or both SR and AR.	

Condition Codes Summary

Encoding	Machine Name	Alt Names	SR Flags	Description
000	eq	z	Z	Equal than. Zero
001	ne	nz	!Z	Not equal. Not zero
010	uge	hs, c	C	Unsigned greater than or equal. Carry
011	ult	lo, nc	!C	Unsigned less than. Not carry
100	ge	-	S == V	Signed greater than or equal
101	lt	-	S != V	Signed less than
110	ugt	hi	C && !Z	Unsigned greater than
111	gt	-	(S == V) && !Z	Signed greater than
-	ule	ls	!C || Z	Unsigned less than or equal Implemented as the opposite of ugt
-	le	-	(S != V) || Z	Signed less than or equal Implemented as the opposite of gt

Opcode Summary, by opcode number

Pattern	Encoding																Description
T1	1	1	1	o	*aaaa aaaa aaaa*												Relative Call/Jump
T2	1	1	0	cc				*aa aaaa aaaa*									Conditional branch
T3	1	0	op			kkkk kkkk							Rd				Move, Compare, ALU immediate
T4	0	1	op			kk kkkk				Rs			Rd				Load/store with immediate offset
T5	0	0	1	op			Rn			Rs			Rd				Three register ALU operation, Load/store with register offset
T6	0	0	0	cc		1	Rn			Rs			Rd				Conditional select
T7	0	0	0	cc		0	1	1	1	x	x	x	Rd				Conditional set
T8	0	0	0	op		0	1	1	0	x	x	x	x	x	x		Zero Operand Instructions, Call absolute
T9	0	0	0	op		0	1	0	o	x	x	x	Rd				Push/Pop, Branch/Call Indirect, Move word immediate, Load/store with absolute address
T10	0	0	0	op		0	0	op		Rs			Rd				Two register ALU operation, Move, Compare, ALU operation, Load/Store with word offset
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NOP instruction, emulated through 'mov r0, r0'

Instructions Summary

	Encoding	Machine Name	Assembly Instruction	Description
Relative Call/Jump				
T1	0	jmprel	jmp Label	PC relative unconditional branch to Label
	1	callrel	jsr Label	PC relative subroutine call to Label
Conditional branch				
T2	%cc	br%cc	br%cc Label	Conditional PC relative branch if %cc matches SR flags, otherwise proceed with the next instruction
Move, Compare, ALU immediate				
T3	000	movkr	mov K, Rd	Copy sign-extended K into Rd
	001	cmpkr	cmp Rd, K	Compare Rd with sign-extended K and update SR flags
	010	addkr	add Rd, K, Rd	Add zero-extended K to Rd and store result in Rd, update SR, AR
	011	subkr	sub Rd, K, Rd	Subtract zero-extended K from Rd and store in Rd, update SR, AR
	100	andkr	and Rd, K, Rd	Logical AND zero-extended K with Rd and store result in Rd, update SR
	101	orkr	or Rd, K, Rd	Logical OR zero-extended K with Rd and store result in Rd, update SR
	110	xorkr	xor Rd, K, Rd	Logical XOR zero-extended K with Rd and store result in Rd, update SR
	111	-	-	Reserved
Load/store with immediate offset				
T4	00	mov16mr	ld.w [Rs, K], Rd	Load contents of word aligned memory address Rn+K into Rd.
	01	movs8mr	ld.sb [Rs, K], Rd	Load sign-extended contents of byte memory address Rn+K into Rd
	10	mov16rm	st.w Rd, [Rs, K]	Store Rd in word aligned memory address Rn+K
	11	mov8rm	st.b Rd, [Rs, K]	Store byte truncated Rd in byte memory address Rn+K
	(*) K is a 6 bit displaced signed immediate in the range -16..<80			
Three register ALU operation				
T5	0000	addrrr	add Rn, Rs, Rd	Rd = Rn + Rs, update SR, AR
	0001	adcrrr	addc Rn, Rs, Rd	Rd = Rn + (Rs+AC), update SR, AR
	0010	subrrr	sub Rn, Rs, Rd	Rd = Rn - Rs, update SR, AR
	0011	subcrrr	subc Rn, Rs, Rd	Rd = Rn - (Rs+!AC), update SR, AR
	0100	orrrr	or Rn, Rs, Rd	Rd = Rn | Rs, update SR
	0101	andrrr	and Rn, Rs, Rd	Rd = Rn & Rs, update SR
	0110	xorrrr	xor Rn, Rs, Rd	Rd = Rn ^ Rs, update SR
	0111	-	-	Reserved
Load/store with register offset				
T5	1000	-	-	
	1001	mov16nr	ld.w [Rn, Rs], Rd	Load contents of word aligned memory at Rn+Rs into Rd
	1010	mov8znr	ld.zb [Rn, Rs], Rd	Load zero-extended contents of byte memory at Rn+Rs into Rd
	1011	mov8snr	ld.sb [Rn, Rs], Rd	Load sign-extended contents of byte memory address Rn+Rs into Rd
	1100	mov16rn	st.w Rd, [Rn, Rs]	Store Rd in word aligned memory address Rn+Rs

	Encoding	Machine Name	Assembly Instruction	Description
	1101	mov8rn	st.b Rd, [Rn, Rs]	Store byte truncated Rd in byte memory address Rn+Rs
	1110	-	-	
	1111	-	-	
Conditional select				
T6	%cc	sel%cc	sel%cc Rn, Rs, Rd	Conditional select. Copy Rn to Rd if %cc matches SR flags, otherwise copy Rs to Rd
Conditional set				
T7	%cc	set%cc	set%cc Rd	Conditional set. Move 1 to Rd if %cc matches SR flags, otherwise move 0 to Rd
Zero Operand Instructions				
T8	000	ret	ret	Return from subroutine
	001	reti	reti	Return from interrupt
	010	dint	dint	Disable interrups
	011	eint	eint	Enable interrupts
	100	-	-	
	101	-	-	
	110	-	-	
	111	calladr A	call &A	Call to subroutine with absolute address (A is in the next word)
Branch/Call indirect				
T9	000 0	jmpreg	jmp Rd	Jump to Rd
	001 0	callreg	call Rd	Subroutine call to Rd
	010 0	push	push Rd	Decrement SP and store Rd onto the stack
	011 0	pop	pop Rd	Load Rd from the stack and increment SP
	100 0	-	-	
	101 0	-	-	
	110 0	movSr	mov S, Rd	Copy Status Register to Rd
	111 0	movrS	mov Rd, S	Restore Status Register from Rd
Move Immediate, Load/store with absolute address				
T9	000 1	leaar K	mov K, Rd	Copy K into Rd (K is in the next word)
	001 1	mov16ar A	ld.w [&A], Rd	Load contents of word aligned memory address A into Rd (A is in the next word)
	010 1	mov8zar A	ld.zb [&A], Rd	Load zero-extended contents of byte memory address A into Rd (A is in the next word)
	011 1	mov8sar A	ld.sb [&A], Rd	Load sign-extended contents of byte memory address A into Rd (A is in the next word)
	100 1	mov16ra A	st.w Rd, [&A]	Store Rd in word aligned memory address A (A is in the next word)
	101 1	mov8ra A	st.b Rd, [&A]	Store lower byte of Rd in byte memory address A (A is in the next word)
	110 1	-	-	-
	111 1	-	-	-
Two register Move, Compare, ALU operation				
	000 00	movrr	mov Rs, Rd	Copy Rs to Rd
	001 00	cmprr	cmp Rs, Rd	Compare Rd with Rs and update SR flags

	Encoding	Machine Name	Assembly Instruction	Description
T10	010 00	zext	zext Rs, Rd	Move zero-extended Rs low byte to Rd
	011 00	sext	sext Rs, Rd	Move sign-extended Rs low byte to Rd
	100 00	bswap	bswap Rs, Rd	Move the swapped bytes of Rs to Rd
	101 00	sextw	sextw Rs, Rd	Sets Rd to all ones if Rs is negative, or zero otherwise
	110 00	-	-	-
	111 00	mov16pr	ld {Rs}, Rd	Load Program Memory
Two Register ALU Operation				
T10	000 01	lsr	lsr Rs, Rd	Rd = Rs >> 1
	001 01	lsl	lsl Rs, Rd	Rd = Rs << 1
	010 01	asr	asr Rs, Rd	Rd = Rs >> 1 (arithmetic shift right)
	011 01	-	-	-
	100 01	-	-	-
	101 01	neg	neg Rs, Rd	Rd = -Rs, update SR
	110 01	not	not Rs, Rd	Rd = ~Rs, update SR
Load/store with word offset				
T10	000 10	leaMrK	add Rs, #K, Rd	Move Rs+K into Rd (K is in the next word). This is equivalent to 'load effective address'
	001 10	mov16MrK	ld.w [Rs, #K], Rd	Load contents of word aligned memory address Rs+K into Rd (K is in the next instruction word)
	010 10	mov8zMrK	ld.zb [Rs, #K], Rd	Load zero-extended contents of byte memory address Rs+K into Rd (K is in the next instruction word)
	011 10	mov8sMrK	ld.sb [Rs, #K], Rd	Load sign-extended contents of byte memory address Rs+K into Rd (K is in the next instruction word)
	100 10	mov16rMK	st.w Rd, [Rs, #K]	Store Rd in word aligned memory address Rs+K (K is in the next instruction word)
	101 10	mov8rMK	st.b Rd, [Rs, #K]	Store lower byte of Rd in byte memory address Rs+K (K is in the next instruction word)
	110 10	-	-	-
	111 10	-	-	-
Reserved (Two register with large immediate)				
T10	xxx 11	ReservedK	-	Reserved