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
R7	16 bit, General Purpose
R8	16 bit, General Purpose
SP	16 bit, Stack Pointer
PC	16 bit, Program Pointer
SR I V S C Z	8 bit, Status Register
All general purpose registers are 16 bit long. ALU operations are 16 bit. 8 bit operations are not natively supported except for extended loads and truncated stores.	

Opcode Summary

<table>
<tr><th></th><th colspan="16">Encoding</th><th>Description</th></tr>
<tr><td>T1</td><td>1</td><td>1</td><td colspan="3">op</td><td colspan="5">k kkkk</td><td colspan="3">Rs</td><td colspan="3">Rd</td><td>Load/store with immediate offset</td></tr>
<tr><td>T2</td><td>1</td><td>0</td><td colspan="4">op</td><td colspan="10">aa aaaa aaaa</td><td>Conditional branch</td></tr>
<tr><td>T3</td><td>0</td><td>1</td><td>1</td><td>o</td><td colspan="12">aaaa aaaa aaaa</td><td>Relative Call/Jump</td></tr>
<tr><td>T4</td><td>0</td><td>1</td><td>0</td><td colspan="2">op</td><td colspan="8">kkkk kkkk</td><td colspan="3">Rd</td><td>Move, Compare, Add, Subtract immediate</td></tr>
<tr><td>T5</td><td>0</td><td>0</td><td>1</td><td colspan="4">op</td><td colspan="3">Rn</td><td colspan="3">Rs</td><td colspan="3">Rd</td><td>Three register ALU operation,
Load/store with register offset</td></tr>
<tr><td rowspan="2">T6</td><td>0</td><td>0</td><td>0</td><td>1</td><td>o</td><td colspan="8">kkkk kkkk</td><td colspan="3">Rd</td><td>SP relative load/store</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td colspan="8">kkkk kkkk</td><td colspan="3">Rd</td><td>PC relative load</td></tr>
<tr><td>T7</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td colspan="2">op</td><td colspan="8">kkkk kkkk</td><td>Add/Subtract offset to SP</td></tr>
<tr><td>T8</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td colspan="4">op</td><td>x</td><td colspan="3">Rd</td><td>Push/Pop, move SP Register, Branch/Call Indirect,
Move immediate, Load/store with absolute address</td></tr>
<tr><td>T9</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>x</td><td colspan="3">op</td><td>x</td><td>x</td><td>x</td><td>x</td><td>Zero Operand Instructions</td></tr>
<tr><td>T10</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td colspan="3">op</td><td colspan="3">Rs</td><td colspan="3">Rd</td><td>Two register Move, Compare, ALU operation</td></tr>
</table>

Instructions Summary

	Encoding	Machine Name	Assembly Instruction	Description
Load/store with immediate offset				
T1	00x	mov16mr	ld.w [#K, Rs], Rd	Load contents of word aligned memory address K+Rs into Rd
	010	movz8mr	ld.zb [#K, Rs], Rd	Load zero-extended contents of byte memory address K+Rs into Rd
	011	movs8mr	ld.sb [#K, Rs], Rd	Load sign-extended contents of byte memory address K+Rs into Rd
	10x	mov16rm	st.w Rd, [#K, Rs]	Store Rd in word aligned memory address K+Rs
	11x	mov8rm	st.b Rd, [#K, Rs]	Store lower byte of Rd in byte memory address K+Rs
Conditional branch				
T2	0000	blt	blt label	Branch PC relative if S != V (signed less than)
	0001	ble	ble label	Branch PC relative if (S != V) || Z (signed less or equal)
	0010	bge	bge label	Branch PC relative if S == V (signed greater than or equal)
	0011	bgt	bgt label	Branch PC relative if (S == V) && !Z (signed greater than)
	0100	bult	bult label blo label bnc label	Branch PC relative if !C (unsigned lower) (not carry)
	0101	bule	bule label bls label	Branch PC relative if !C || Z (unsigned lower or same)
	0110	buge	buge label bhs label bc label	Branch PC relative if C (unsigned higher or same) (carry)
	0111	bugt	bugt label bhi label	Branch PC relative if C && !Z (unsigned higher)
	1000	beq	beq label bz label	Branch PC relative if Z (equal) (zero)
	1001	bne	bne label bnz label	Branch PC relative if !Z (not equal) (not zero)
	1010	bs	bs label	Branch PC relative if N (negative)
	1011	bns	bv label	Branch PC relative if V (overflow)
	1100	-	-	-
	1101	-	-	-
	1110	-	-	-
	1111	-	-	-
Relative Call/Jump				
T3	0	jmprel	jmp label	PC relative unconditional branch
	1	callrel	call label	PC relative subroutine call
Move, Compare, Add, Subtract immediate				
T4	00	movkr	mov #K, Rd	Copy sign-extended K into Rd
	01	cmpkr	cmp #K, Rd	Compare Rd with sign-extended K
	10	addkr	add #K, Rd	Add zero-extended K to Rd

	Encoding	Machine Name	Assembly Instruction	Description
	11	subkr	sub #K, Rd	Subtract zero-extended K from Rd
Three register ALU operation				
T5	0000	addrrr	add Rn, Rs, Rd	Rd = Rn+Rs, update flags
	0001	adcrrr	addc Rn, Rs, Rd	Rd = Rn+(Rs+C), update flags
	0010	subrrr	sub Rn, Rs, Rd	Rd = Rn-Rs, update flags
	0011	subcrrr	subc Rn, Rs, Rd	Rd = Rn-(Rs+!C), update flags
	0100	orrrr	or Rn, Rs, Rd	Rd = Rn | Rs, update flags
	0101	andrrr	and Rn, Rs, Rd	Rd = Rn & Rs, update flags
	0110	xorrrr	xor Rn, Rs, Rd	Rd = Rn ^ Rs, update flags
	0111	-	-	Reserved
Load/store with register offset				
T5	100x	mov16nr	ld.w [Rn, Rs], Rd	Load contents of word aligned memory address Rn+Rs into Rd
	1010	mov8znr	ld.zb [Rn, Rs], Rd	Load zero-extended contents of byte memory address Rn+Rs into Rd
	1011	mov8snr	ld.sb [Rn, Rs], Rd	Load sign-extended contents of byte memory address Rn+Rs into Rd
	110x	mov16rn	st.w Rd, [Rn, Rs]	Store Rd in word aligned memory address Rn+Rs
	111x	mov8rn	st.b Rd, [Rn, Rs]	Store lower byte of Rd in byte memory address Rn+Rs
SP relative load/store				
T6	0	mov16qr	ld.w [#K, SP], Rd	Load contents of stack memory address K+SP into Rd
	1	mov16rq	st.w Rd, [#K, SP]	Store Rd in stack memory address K+SP
PC Relative load				
T6	0	mov16pr	mov.w [#K, PC], Rd	Load contents of program memory address K+PC into Rd
Add/Subtract offset to SP				
T7	00	addks	add #K, SP	Add zero-extended K to Rd
	01	subks	sub #K, SP	Subtract zero-extended K from Rd
	-	-	-	-
Push/Pop, move SP Register, Branch/Call indirect				
T8	0000	push	push Rd	Decrement SP and push Rd onto the stack
	0001	pop	pop Rd	Pop Rd from the sta
	0010	movsr	mov SP, Rd	Copy SP into Rd
	0011	movrs	mov Rd, SP	Copy Rd into SP
	0100	jmpreg	jmp Rd	Jump to Rd
	0101	callreg	call Rd	Subroutine call to Rd
	0110			
	0111	-	-	-
Move Immediate, Load/store with absolute address				
T8	1000	movKr	mov.w #K, Rd	Copy K into Rd (K is in next instruction word)
	1001	mov16ar	ld.w [#A], Rd	Load contents of word aligned memory address A into Rd (A is in next instruction word)
	1010	mov8zar	ld.zb [#A], Rd	Load zero-extended contents of byte memory address A into Rd (A is in next instruction word)
	1011	mov8sar	ld.sb [#A], Rd	Load sign-extended contents of byte memory address A into Rd (A is in next instruction word)

	Encoding	Machine Name	Assembly Instruction	Description
	110x	mov16ra	st.w Rd, [#A]	Store Rd in word aligned memory address A (A is in next instruction word)
	111x	mov8ra	st.b Rd, [#A]	Store lower byte of Rd in byte memory address A (A is in next instruction word)
Zero Operand Instructions				
T9	000	ret	ret	Return from subroutine
	001	reti	reti	Return from interrupt
	010	dint	dint	Disable interrups
	011	eint	eint	Enable interrupts
Two register Move, Compare, ALU operation				
T10	000	movrr	mov Rs, Rd	Copy Rs.w to Rd.w
	001	cmprr	cmp Rs, Rd	compare Rd.w with Rs and set flags
	010	zext	zext Rs, Rd	Zero extend Rs low byte into Rd
	011	sext	sext Rs, Rd	Sign extend Rs low byte into Rd
	100	lsr	lsr Rs, Rd	1 bit shift right of Rs into Rd and update flags
	101	lsl	lsl Rs, Rd	1 bit shift left of Rs into Rd update flags
	110	asr	asr Rs, Rd	1 bit signed shift right of Rs into Rd and update flags
	111	swapb	swapb Rs, Rd	Moves the swapped bytes of Rs into Rd