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 Counter
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

Opcodes Summary

Pattern	Type	Encoding	Description
P1	T1	1 1 1 o aaaa aaaa aaaa	Relative Call/Jump
P2	T2	1 1 0 cc aa aaaa aaaa	Conditional branch
P3	T3	1 0 op kkkk kkkk Rd	Move, Compare, ALU immediate
P1	T4	0 1 op kk kkkk Rs Rd	Load/store with immediate offset
P4	T5	0 0 1 op Rn Rs Rd	Three register ALU operation, Load/store with register offset
P5	T6	0 0 0 cc 1 Rn Rs Rd	Conditional select
P6	T7	0 0 0 cc 0 1 1 1 x x x Rd	Conditional set
P7	T8	0 0 0 op 0 1 1 0 x x x x x x	Zero Operand Instructions, Call absolute
P7	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
P7	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

Category	Assembly Mnemonic	Description
Arithmetic, Logic		

Arithmetic	add Rd, K, Rd
	add Rn, Rs, Rd
	add Rs, #K, Rd
		Add

	addc Rn, Rs, Rd	Add with carry
	sub Rd, K, Rd sub Rn, Rs, Rd	Subtract
	subc Rn, Rs, Rd	Subtract with carry
	neg Rs, Rd	Negate
	zext Rs, Rd	Zero extend byte
	sext Rs, Rd	Sign extend byte
	sextw Rs, Rd	Sign extend word
Logic	and Rd, K, Rd and Rn, Rs, Rd	Logical And
	or Rd, K, Rd or Rn, Rs, Rd	Logical Or
	xor Rd, K, Rd xor Rn, Rs, Rd	Exclusive logical Or
	not Rs, Rd	Logical Not
Shifts	asr Rs, Rd	Arithmetic shift right
	lsr Rs, Rd	Logical shift right
	lsl Rs, Rd	Logical shift left
Comparison	cmp Rd, K cmp Rd, Rs	Compare
Data moves		

Category	Assembly Mnemonic	Description

Moves	mov K, Rd
	mov #K, Rd
	mov Rs, Rd
		Move

	bswap Rs, Rd	Byte swap
Conditional moves	sel%cc Rn, Rs, Rd	Select
	set%cc Rd	Set
Memory access		
Memory load	ld.w [Rs, K], Rd ld.w [Rs, #K], Rd ld.w [Rn, Rs], Rd ld.w [&A], Rd	Load word from memory
	ld.sb [Rs, K], Rd ld.sb [Rs, #K], Rd ld.sb [Rn, Rs], Rd ld.sb [&A], Rd	Load sign extended byte from memory
	ld.zb [Rs, #K], Rd ld.zb [Rn, Rs], Rd ld.zb [&A], Rd	Load zero extended byte from memory
Memory store	st.w Rd, [Rs, K] st.w Rd, [Rs, #K] st.w Rd, [Rn, Rs] st.w Rd, [&A]	Store word to memory
	st.b Rd, [Rs, K] st.b Rd, [Rs, #K] st.b Rd, [Rn, Rs] st.b Rd, [&A]	Store byte to memory
Stack access	push Rd	Push to memory stack
	pop Rd	Pop from memory stack
Program memory	ld.w {Rs}, Rd	Program memory read
Branching and subroutines		
Branch instructions	jmp Label jmp Rd	Unconditional branch
	br%cc Label	Conditional branch
Subroutine instructions	jsr Label call Rd call &A	Call to subroutine
	ret	Return from subroutine
Interrupts		
Interrupt instructions	dint	Disable interrupt
	eint	Enable interrupt
	reti	Return from interrupt
	halt	Halts processor

Instructions Summary, by opcode

	Encoding	Machine Name	Assembly Mnemonic	Description
Relative Call/Jump				
T1	0	jmp_k	jmp Label	PC relative unconditional branch to Label
	1	call_k	jsr Label	PC relative subroutine call to Label
Conditional branch				
T2	%cc	br_ck	br%cc Label	Conditional PC relative branch if %cc matches SR flags, otherwise proceed with the next instruction
Move, Compare, ALU immediate				
T3	000	mov_kr	mov K, Rd	Copy sign-extended K into Rd
	001	cmp_rk	cmp Rd, K	Compare Rd with sign-extended K and update SR flags
	010	add_kr	add Rd, K, Rd	Add zero-extended K to Rd and store result in Rd, update SR, AR
	011	sub_kr	sub Rd, K, Rd	Subtract zero-extended K from Rd and store in Rd, update SR, AR
	100	and_kr	and Rd, K, Rd	Logical AND zero-extended K with Rd and store result in Rd, update SR
	101	or_kr	or Rd, K, Rd	Logical OR zero-extended K with Rd and store result in Rd, update SR
	110	xor_kr	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	movw_mr	ld.w [Rs, K], Rd	Load contents of word aligned memory address Rn+K into Rd.
	01	movsb_mr	ld.sb [Rs, K], Rd	Load sign-extended contents of byte memory address Rn+K into Rd
	10	movw_rm	st.w Rd, [Rs, K]	Store Rd in word aligned memory address Rn+K
	11	movb_rm	st.b Rd, [Rs, K]	Store byte truncated Rd in byte memory address Rn+K
	(*) K is a positive immediate in the range 0 ..< 64			
Three register ALU operation				
T5	0000	add_rrr	add Rn, Rs, Rd	Rd = Rn + Rs, update SR, AR
	0001	adc_rrr	addc Rn, Rs, Rd	Rd = Rn + (Rs+AC), update SR, AR
	0010	sub_rrr	sub Rn, Rs, Rd	Rd = Rn - Rs, update SR, AR
	0011	subc_rrr	subc Rn, Rs, Rd	Rd = Rn - (Rs+!AC), update SR, AR
	0100	or_rrr	or Rn, Rs, Rd	Rd = Rn | Rs, update SR
	0101	and_rrr	and Rn, Rs, Rd	Rd = Rn & Rs, update SR
	0110	xor_rrr	xor Rn, Rs, Rd	Rd = Rn ^ Rs, update SR
	0111	-	-	Reserved
Load/store with register offset				
T5	1000	-	-	
	1001	movw_nr	ld.w [Rn, Rs], Rd	Load contents of word aligned memory at Rn+Rs into Rd
	1010	movzb_nr	ld.zb [Rn, Rs], Rd	Load zero-extended contents of byte memory at Rn+Rs into Rd
	1011	movsb_nr	ld.sb [Rn, Rs], Rd	Load sign-extended contents of byte memory address Rn+Rs into Rd
	1100	movw_rn	st.w Rd, [Rn, Rs]	Store Rd in word aligned memory address Rn+Rs

	Encoding	Machine Name	Assembly Mnemonic	Description
	1101	movb_rn	st.b Rd, [Rn, Rs]	Store byte truncated Rd in byte memory address Rn+Rs
	1110	-	-	
	1111	-	-	
Conditional select				
T6	%cc	sel_crrr	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_cr	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	halt	halt	Halts processor and sets it into program mode
	101	-	-	
	110	-	-	
	111	call_a	call &A	Call to subroutine with absolute address (A is in the next word)
Branch/Call indirect				
T9	000 0	jmp_r	jmp Rd	Jump to Rd
	001 0	call_r	call Rd	Subroutine call to Rd
	010 0	push_r	push Rd	Decrement SP and store Rd onto the stack
	011 0	pop_r	pop Rd	Load Rd from the stack and increment SP
	100 0	-	-	
	101 0	-	-	
	110 0	mov_sr	mov S, Rd	Copy Status Register to Rd
	111 0	mov_rs	mov Rd, S	Restore Status Register from Rd
Move Immediate, Load/store with absolute address				
T9	000 1	mov_lr lea_ar	mov #K, Rd mov &A, Rd	Copy K into Rd (K is in the next word)
	001 1	movw_ar	ld.w [&A], Rd	Load contents of word aligned memory address A into Rd (A is in the next word)
	010 1	movzb_ar	ld.zb [&A], Rd	Load zero-extended contents of byte memory address A into Rd (A is in the next word)
	011 1	movsb_ar	ld.sb [&A], Rd	Load sign-extended contents of byte memory address A into Rd (A is in the next word)
	100 1	movw_ra	st.w Rd, [&A]	Store Rd in word aligned memory address A (A is in the next word)
	101 1	movb_ra	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	mov_rr	mov Rs, Rd	Copy Rs to Rd
	001 00	cmp_rr	cmp Rd, Rs	Compare Rd with Rs and update SR flags

	Encoding	Machine Name	Assembly Mnemonic	Description
T10	010 00	zext_rr	zext Rs, Rd	Move zero-extended Rs low byte to Rd
	011 00	sext_rr	sext Rs, Rd	Move sign-extended Rs low byte to Rd
	100 00	bswap_rr	bswap Rs, Rd	Move the swapped bytes of Rs to Rd
	101 00	sextw_rr	sextw Rs, Rd	Sets Rd to all ones if Rs is negative, or zero otherwise
	110 00	-	-	-
	111 00	movw_pr	ld.w {Rs}, Rd	Load Program Memory
Two Register ALU Operation				
T10	000 01	lsr_rr	lsr Rs, Rd	Rd = Rs >> 1
	001 01	lsl_rr	lsl Rs, Rd	Rd = Rs << 1
	010 01	asr_rr	asr Rs, Rd	Rd = Rs >> 1 (arithmetic shift right)
	011 01	-	-	-
	100 01	-	-	-
	101 01	neg_rr	neg Rs, Rd	Rd = -Rs, update SR, AR
	110 01	not_rr	not Rs, Rd	Rd = ~Rs, update SR
	111 01	-	-	-
Load/store with word offset				
T10	000 10	add_rlr lea_qr	add Rs, #K, Rd	Move Rs+K into Rd (K is in the next word). This is equivalent to 'load effective address'
	001 10	movw_qr	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	movzb_qr	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	movsb_qr	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	movw_rq	st.w Rd, [Rs, #K]	Store Rd in word aligned memory address Rs+K (K is in the next instruction word)
	101 10	movb_rq	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	Reserved	-	Reserved