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
SP	16 bit, Stack Pointer
PC	16 bit, Program Counter
Status I V S C Z	Status Register I: Interrupt flag V, S, C, Z: Condition flags
All registers are 16 bit. ALU operations are 16 bit. 8 bit operations are not natively supported except for extended loads and truncated stores. Registers R0 through R7 are general purpose. Registers SP and PC are not in the general set, they use special instructions.	

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

Instruction Formats

Type	Encoding				Description
P	opcode (5)	immediate (11)			Prefix
I1	opcode (5)	immediate (5)	Rs (3)	Rd (3)	Two registers with immediate
I2	opcode (5)	immediate (8)		Rd (3)	One register with immediate
J	opcode (7)	address (9)			No registers with immediate address
R1	opcode (7)	Rn (3)	Rs (3)	Rd (3)	Three registers
R2	opcode (10)		Rs (3)	Rd (3)	Zero, One or Two registers

Opcodes Summary

Type	Encoding												Description
P	1	1	1	1	o	aaa aaaa aaaa							Prefix, call immediate
I1	1	1	op (000..101)			k kkkk			Rs		Rd		Load, Store, Lea, with register+immediate
I2	op (01010..10111)					kkkk kkkk					Rd		Move, Compare, Add, Sub, And, Or, Load, Store, with immediate
J	0	1	0	0	cc	a aaaa aaaa							Conditional branch
	0	0	1	111	o	a aaaa aaaa							Unconditional branch, Add SP
R1	0	0	1	op (0000..1101)			Rn		Rs		Rd		Three register ALU operation, Load/store with register offset
	0	0	0	1	cc		Rn		Rs		Rd		Conditional select
R2	0	0	0	0	cc		11	x	xxx		Rd		Conditional set
	0	0	0	0	op		op (000..011)		Rs/xxx		Rd/xxx		Two register ALU operation, Move, Compare, Push/Pop, Jump/Call Indirect, Zero Operand Instructions,
	0	0	0	0	0	0	0	0	0	0	000	000	NOP instruction, emulated through 'mov r0, r0'

Prefixed instructions

The prefixed instructions are assembler emulated instructions that are made of core instructions preceded by a prefix instruction. The prefix instruction contains a 'p_imm' 11 bit immediate field that expands the functionality of core instructions. The prefix instruction extends the immediate field 'imm' of the next instruction by replacing it with the result of the logical expression: (p_imm << 5) | (imm & 0b11111), thus providing a full 16 bit immediate range to the prefixed instruction.

The following non exhaustive list shows several examples of prefix instruction transformations:

Core Instruction	Prefix	Prefixed Instruction	Description
Arithmetic, Logic			
add Rd, K, Rd	pfix_k	add Rd, #K, Rd	Add with long immediate. The 8 bit embedded immediate is replaced by a 16 bit one
and Rd, K, Rd	pfix_k	and Rd, #K, Rd	And with long immediate. The 8 bit embedded immediate is replaced by a 16 bit one
lea Rs, K, Rd	pfix_k	lea Rs, #K, Rd	Lea with long immediate. The 5 bit embedded immediate is replaced by a 16 bit one
Moves			
mov K, Rd	pfix_k	mov #K, Rd	Copy K into Rd. The 8 bit embedded immediate is replaced by a 16 bit one
Branching and subroutines			
br%cc Label	pfix_k	br%cc Label	Conditional branch. Branch instruction reach is extended from 9 to 16 bit long offsets
call Label	pfix_k	call &Label	Subroutine call. Call instruction reach is extended from 11 to 16 bit addresses
Memory			
ld.w [Rs, K], Rd	pfix_k	ld.w [Rs, #K], Rd	Load word with immediate offset. The 5 embedded immediate is replaced by a 16 bit one
ld.w [A], Rd	pfix_k	ld.w [&A], Rd	Load word with immediate absolute address. The 8 bit embedded immediate field is replaced by a 16 bit address

Instructions Summary

Category	Assembly Mnemonic	Description
Arithmetic, Logic		
Arithmetic	add Rd, K, Rd add Rs, Rn, Rd	Add
	lea Rs, K, Rd lea SP, K, Rd add SP, K, SP	Add address
	addc Rs, Rn, Rd	Add with carry
	sub Rd, K, Rd sub Rs, Rn, Rd	Subtract
	subc Rs, Rn, 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 Rs, Rn, Rd	Logical And
	or Rs, Rn, Rd	Logical Or
	xor Rs, Rn, 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 Rs, Rn	Compare
	cmpc Rs, Rn	Compare with carry
Data moves		

Moves	mov K, Rd
	mov Rs, Rd
	mov Rs, SP
		Move

	bswap Rs, Rd	Byte swap
Conditional moves	sel%cc Rs, Rn, Rd	Select
	set%cc Rd	Set
Memory access		
Memory load	ld.w [Rs, K], Rd ld.w [Rs, Rn], Rd ld.w [&A], Rd ld.w [SP, K], Rd	Load word from memory
	ld.w {Rs}, Rd	Load word from program memory
	ld.sb [Rs, K], Rd ld.sb [Rs, Rn], Rd ld.sb [&A], Rd ld.sb [SP, K], Rd	Load sign extended byte from memory
	ld.zb [Rs, K], Rd ld.zb [Rs, Rn], Rd	Load zero extended byte from memory

Category	Assembly Mnemonic	Description
Memory store	st.w Rd, [Rs, K] st.w Rd, [Rs, Rn] st.w Rd, [&A] st.w Rd, [SP, K]	Store word to memory
	st.b Rd, [Rs, K] st.b Rd, [Rs, Rn] st.b Rd, [&A] st.b Rd, [SP, K]	Store byte to memory
Stack Pointer specific		
Push/Pop	push Rd	Push to stack
	pop Rd	Pop from stack
Branching and subroutines		
Branch instructions	jmp Label jmp Rd	Unconditional branch
	br%cc Label	Conditional branch
Subroutine instructions	call Label call Rd	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

Type	Opcode	Machine Name	Assembly Mnemonic	Description
Prefix immediate				
P	1	pfix_k	-	Prefix immediate
	0	call	call &Label	Call immediate
				(*) Label is a 11 bit immediate extensible to a 16 bit word with the prefix instruction
Load/store with immediate offset				
I1	000	lea_mr	lea Rs, K, Rd	Add zero-extended K to Rs, store result in Rd
	001	movw_mr	ld.w [Rs, K], Rd	Load word at aligned memory address Rn+K, store in Rd
	010	movzb_mr	ld.zb [Rs, K], Rd	Load byte at memory address Rn+K, zero-extend into Rd
	011	movsb_mr	ld.sb [Rs, K], Rd	Load byte at memory address Rn+K, sign-extend into Rd
	100	movw_rm	st.w Rd, [Rs, K]	Store Rd in word aligned memory address Rn+K
	101	movb_rm	st.b Rd, [Rs, K]	Store lower byte of Rd in memory address Rn+K
	110	-	-	Not Available
	111	-	-	Not Available
				(*) 'K' is a 5 bit immediate in the range 0..31 extensible to a 16 bit word with the prefix instruction
Move, Compare, Add, Sub, And, Load, Store, with immediate				
I2	01010	mov_kr	mov K, Rd	Copy sign-extended K into Rd
	01011	cmp_rk	cmp Rd, K	Compare Rd with sign-extended K and update SR flags
	01100	add_kr	add Rd, K, Rd	Add zero-extended K to Rd, store result in Rd, update SR
	01101	sub_kr	sub Rd, K, Rd	Subtract zero-extended K from Rd, store in Rd, update SR
	01110	and_kr	and Rd, K, Rd	AND zero-extended K with Rd, store in Rd, update SR
	01111	movw_ar	ld.w [&A], Rd	Load word at aligned memory address A, store in Rd
	10000	movsb_ar	ld.sb [&A], Rd	Load byte at memory address A, sign-extend into Rd
	10001	movw_ra	st.w Rd, [&A]	Store Rd in word aligned memory address A
	10010	movb_ra	st.b Rd, [&A]	Store lower byte of Rd in memory address A
	10011	lea_qr	lea SP, K, Rd	Add zero-extended K to SP, store result in Rd
	10100	movw_qr	ld.w [SP, K], Rd	Load word at aligned memory address SP+K, store in Rd.
	10101	movsb_qr	ld.sb [SP, K], Rd	Load byte at memory address SP+K, sign-extend into Rd
	10110	movw_rq	st.w Rd, [SP, K]	Store Rd in word aligned memory address SP+K
	10111	movb_rq	st.b Rd, [SP, K]	Store lower byte of Rd in memory address SP+K
				(*) 'K' is a 8 bit immediate in the range 0..255, or -128..+127 extensible to a 16 bit word with the prefix instruction
Conditional branch, Unconditional branch				
J	%cc	br_ck	br%cc Label	Conditional PC relative branch if %cc matches SR flags, otherwise proceed with the next instruction
	0	add_kq	add SP, K, SP	Add signed immediate to SP
	1	jmp_k	jmp Label	PC relative unconditional branch to Label
				(*) 'Label' and 'K' are 9 bit signed immediates extensible to a 16 bit word with the prefix instruction
Three register ALU operation				
	0000	cmp_rr	cmp Rs, Rn	Compare Rs with Rn and update SR flags

Type	Opcode	Machine Name	Assembly Mnemonic	Description
R1	0001	cmpc_rr	cmpc Rs, Rn	Compare Rs with Rn and update SR flags
	0010	sub_rrr	sub Rs, Rn, Rd	Rd = Rs - Rn, update SR
	0011	subc_rrr	subc Rs, Rn, Rd	Rd = Rs - (Rn+C), update SR
	0100	or_rrr	or Rs, Rn, Rd	Rd = Rs | Rn, update SR
	0101	and_rrr	and Rs, Rn, Rd	Rd = Rs & Rn, update SR
	0110	xor_rrr	xor Rs, Rn, Rd	Rd = Rs ^ Rn, update SR
	0111	adc_rrr	addc Rs, Rn, Rd	Rd = Rs + (Rn+C), update SR
Load/store with register offset				
R1	1000	add_rrr	add Rs, Rn, Rd	Rd = Rs + Rn, update SR
	1001	movw_nr	ld.w [Rs, Rn], Rd	Load word at aligned memory address Rs+Rn, store in Rd
	1010	movzb_nr	ld.zb [Rs, Rn], Rd	Load byte at memory address Rs+Rn, store in Rd
	1011	movsb_nr	ld.sb [Rs, Rn], Rd	Load byte at memory address Rs+Rn, store in Rd
	1100	movw_rn	st.w Rd, [Rs, Rn]	Store Rd in word aligned memory address Rs+Rn
	1101	movb_rn	st.b Rd, [Rs, Rn]	Store lower byte of Rd in memory address Rn+Rs
	1110	-	-	Not Available
	1111	-	-	Not Available
Conditional select				
R1	%cc	sel_crrr	sel%cc Rs, Rn, Rd	Conditional select. Copy Rs to Rd if %cc matches SR flags, otherwise copy Rn to Rd
Conditional set				
R2	%cc	set_cr	set%cc Rd	Conditional set. Move 1 to Rd if %cc matches SR flags, otherwise move 0 to Rd
Two register Move, Compare, ALU operation				
R2	000_000	mov_rr	mov Rs, Rd	Copy Rs to Rd
	001_000	mov_rq	mov Rs, SP	Copy Rs to SP
	010_000	zext_rr	zext Rs, Rd	Move zero-extended Rs low byte to Rd
	011_000	sext_rr	sext Rs, Rd	Move sign-extended Rs low byte to Rd
	100_000	bswap_rr	bswap Rs, Rd	Move the swapped bytes of Rs to Rd
	101_000	sextw_rr	sextw Rs, Rd	Sets Rd to all ones if Rs is negative, or zero otherwise
	110_000	-	-	Reserved
	111_000	movw_pr	ld.w {Rs}, Rd	Load Program Memory
Two Register ALU Operation				
R2	000_001	lsr_rr	lsr Rs, Rd	Logical shift right. Bit 0 is shifted to the C Flag. Bit 15 is set to zero.
	001_001	lsrc_rr	lsrc Rs, Rd	Shift Right through carry. Bit 0 is shifted to the C Flag. The old C flag is shifted to bit 15
	010_001	asr_rr	asr Rs, Rd	Arithmetic shift right. Bit 0 is shifted to the C Flag. bit 15 is preserved
	011_001	-	-	Reserved
	100_001	-	-	Reserved
	101_001	neg_rr	neg Rs, Rd	Rd = 0 - Rs, update SR
	110_001	not_rr	not Rs, Rd	Rd = ~Rs, update SR
	111_001	-	-	Reserved
	(*) Left shifts are implemented with the add and addc instructions			
Branch/Call indirect				

Type	Opcode	Machine Name	Assembly Mnemonic	Description
R2	000_010	jmp_r	jmp Rd	Jump to Rd
	001_010	call_r	call Rd	Subroutine call to Rd
	010_010	push_r	push Rd	Decrement SP and store Rd onto the stack
	011_010	pop_r	pop Rd	Load Rd from the stack and increment SP
	100_010	-	-	Reserved
	101_010	-	-	Reserved
	110_010	mov_sr	mov SR, Rd	Copy Status Register to Rd (Not implemented)
	111_010	mov_rs	mov Rd, SR	Restore Status Register from Rd (Not implemented)
Zero Operand Instructions				
R2	000_011	ret	ret	Return from subroutine
	001_011	reti	reti	Return from interrupt
	010_011	dint	dint	Disable interrups
	011_011	eint	eint	Enable interrupts
	100_011	halt	halt	Halts processor and sets it into program mode
	101_011	-	-	Reserved
	110_011	-	-	Reserved
	111_011	-	-	Reserved
Reserved				
R2	000_100. .111_100	-	-	Reserved
	000_101. .111_101	-	-	Reserved
Not Available				
R2	000_110. .111_110	-	-	Not Available
	000_111. .111_111	-	-	Not Available