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

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
I1	0	1	op			k kkkk				Rs		Rd	Load, Store, Lea, with immediate
I2	op (100..110)			op		kkkk kkkk						Rd	Move, Compare, Add, Sub, And, Or, Load, Store, with immediate
J	1	1	1	0	cc	a aaaa aaaa							Conditional branch
	0	0	1	111	o	a aaaa aaaa							Subroutine call, Unconditional branch
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		111			xxx		Rd	Conditional set
	0	0	0	0	op		op (000..110)			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 one of the two prefix instructions. The prefix instructions contain a 'p_imm' 11 bit immediate field and a 'zero' bit that extend the functionality of core instructions in two possible ways:

- If the zero bit is disabled, the prefix instruction extends the immediate field 'imm' of the core instruction by replacing it with the result of the logical expression: (p_imm << 6) | (imm & 0b111111).

- If the zero bit is enabled, the prefix instruction additionally sets the first register operand of the core instruction to zero, without changing the underlying register.

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 core immediate is replaced by a 16 bit one
add Rs, K, Rd	pfix_k	add Rs, #K, Rd	Lea with long immediate. The 5 bit core immediate is replaced by a 16 bit one
Moves			
mov K, Rd	pfix_k	mov #K, Rd	Copy K into Rd. The 8 bit core immediate is replaced by a 16 bit one
sel%cc Rs, Rn, Rd	pfixz_k	sel%cc #0, Rn, Rd	Conditional select. Set Rd to zero if %cc matches SR flags, otherwise copy Rn to Rd. Register operand Rs on the core instruction is treated as 0
Branching and subroutines			
br%cc Label	pfix_k	br%cc Label	Conditional branch. Branch core 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 9 to 16 bit long adresses
Memory			
ld.w [Rs, K], Rd	pfix_k	ld.w [Rs, #K], Rd	Load word with immediate offset. The 5 bit core immediate is replaced by a 16 bit one
ld.w [Rs, K], Rd	pfixz_k	ld.w [&A], Rd	Load word with immediate absolute address. The 5 bit core immediate field is replaced by a 16 bit address. Register operand Rs is treated as zero

Instructions Summary

Category	Assembly Mnemonic	Description
Arithmetic, Logic		

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

	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 Rd, K, Rd 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 Rd, Rs	Compare
Data moves		
Moves	mov K, Rd mov Rs, Rd	Move
	mov SP, Rd	Move stack pointer
	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 [Rn, Rs], Rd ld.w [&A], Rd (*)	Load word from memory
	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, [Rn, Rs] st.w Rd, [&A]	Store word to memory
	st.b Rd, [Rs, K] st.b Rd, [Rn, Rs] st.b Rd, [&A] (*)	Store byte to memory

Category	Assembly Mnemonic	Description
Stack access	push Rd	Push to stack
	pop Rd	Pop from stack
Program memory	ld.w {Rs}, Rd	Program memory read
		(*) Prefixed instructions
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	0	pfix_k	-	Prefix immediate
	1	pfixz_k	-	Prefix immediate with zeroing
Load/store with immediate offset				
I1	000	lea_mr	add Rs, K, Rd	Add zero-extended K to Rs and store result in Rd
	001	movw_mr	ld.w [Rs, K], Rd	Load contents of word aligned memory address Rn+K into Rd.
	010	movzb_mr	ld.zb [Rs, K], Rd	Load zero-extended contents of byte memory address Rn+K into Rd
	011	movsb_mr	ld.sb [Rs, K], Rd	Load sign-extended contents of byte memory address Rn+Rs 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 byte truncated Rd in byte memory address Rn+K
	110	-	-	Reserved
	111	-	-	Reserved
				(*) K is a 6 bit immediate in the range 0..31 extendable to a 16 bit word with the prefix instruction
Load/store with absolute address				
I1		-	-	-
		movw_ar	ld.w [&A], Rd	Load contents of word aligned memory address A into Rd (
		movzb_ar	ld.zb [&A], Rd	Load zero-extended contents of byte memory address A into Rd
		movsb_ar	ld.sb [&A], Rd	Load sign-extended contents of byte memory address A into Rd
		movw_ra	st.w Rd, [&A]	Store Rd in word aligned memory address A
		movb_ra	st.b Rd, [&A]	Store lower byte of Rd in byte memory address A
		-	-	Reserved
		-	-	Reserved
				(*) These are emulated instructions through the prefix with zeroing instruction
Move, Compare, Add, Sub, And, Or, Load, Store, with immediate				
I2	100_00	mov_kr	mov K, Rd	Copy sign-extended K into Rd
	100_01	cmp_rk	cmp Rd, K	Compare Rd with sign-extended K and update SR flags
	100_10	add_kr	add Rd, K, Rd	Add zero-extended K to Rd and store result in Rd, update SR, AR
	100_11	sub_kr	sub Rd, K, Rd	Subtract zero-extended K from Rd and store in Rd, update SR, AR
	101_00	and_kr	and Rd, K, Rd	Logical AND zero-extended K with Rd and store result in Rd, update SR
	101_01	or_kr	or Rd, K, Rd	Logical OR zero-extended K with Rd and store result in Rd, update SR
	101_10	lea_sr	add SP, K, Rd	Add zero-extended K to SP and store result in Rd
	101_11	movw_sr	ld.w [SP, K], Rd	Load contents of word aligned memory address SP+K into Rd.
	110_00	movsb_sr	ld.sb [SP, K], Rd	Load sign-extended contents of byte memory address SP+K into Rd

Type	Opcode	Machine Name	Assembly Mnemonic	Description
	110_01	movzb_sr	ld.zb [SP, K], Rd	Load zero-extended contents of byte memory address SP+K into Rd
	110_10	movw_rs	st.w Rd, [SP, K]	Store Rd in word aligned memory address SP+K
	110_11	movb_rs	st.b Rd, [SP, K]	Store byte truncated Rd in byte memory address SP+K
	(*) K is a 8 bit immediate in the range 0..255, or -128..+127 extendable to a 16 bit word with the prefix instruction			
Conditional branch				
J	%cc	br_ck	br%cc Label	Conditional PC relative branch if %cc matches SR flags, otherwise proceed with the next instruction
Subroutine call, Unconditional branch				
J	0	jmp_k	jmp Label	PC relative unconditional branch to Label
	1	call_k	jsr Label	PC relative subroutine call to Label
Three register ALU operation				
R1	0000	add_rrr	add Rs, Rn, Rd	Rd = Rs + Rn, update SR, AR
	0001	adc_rrr	addc Rs, Rn, Rd	Rd = Rs + (Rn+AC), update SR, AR
	0010	sub_rrr	sub Rs, Rn, Rd	Rd = Rs - Rn, update SR, AR
	0011	subc_rrr	subc Rs, Rn, Rd	Rd = Rs - (Rn+!AC), update SR, AR
	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	-	-	Reserved
Load/store with register offset				
R1	1000	-	-	Reserved
	1001	movw_nr	ld.w [Rs, Rn], Rd	Load contents of word aligned memory at Rs+Rn into Rd
	1010	movzb_nr	ld.zb [Rs, Rn], Rd	Load zero-extended contents of byte memory at Rs+Rn into Rd
	1011	movsb_nr	ld.sb [Rs, Rn], Rd	Load sign-extended contents of byte memory address Rs+Rn into 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 byte truncated Rd in byte 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	cmp_rr	cmp Rd, Rs	Compare Rd with Rs and update SR flags
	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

Type	Opcode	Machine Name	Assembly Mnemonic	Description
	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	Rd = Rs >> 1
	001_001	lsl_rr	lsl Rs, Rd	Rd = Rs << 1
	010_001	asr_rr	asr Rs, Rd	Rd = Rs >> 1 (arithmetic shift right)
	011_001	-	-	Reserved
	100_001	-	-	Reserved
	101_001	neg_rr	neg Rs, Rd	Rd = -Rs, update SR, AR
	110_001	not_rr	not Rs, Rd	Rd = ~Rs, update SR
	111_001	-	-	Reserved
Reserved				
R2	000_010. .111_010	-	-	Reserved
	000_011. .111_011	-	-	Reserved
Branch/Call indirect				
R2	000_100	jmp_r	jmp Rd	Jump to Rd
	001_100	call_r	call Rd	Subroutine call to Rd
	010_100	push_r	push Rd	Decrement SP and store Rd onto the stack
	011_100	pop_r	pop Rd	Load Rd from the stack and increment SP
	100_100	-	-	Reserved
	101_100	-	-	Reserved
	110_100	mov_tr	mov SR, Rd	Copy Status Register to Rd (Not implemented)
	111_100	mov_rt	mov Rd, SR	Restore Status Register from Rd (Not implemented)
Reserved				
R2	000_101. .111_101	-	-	Reserved
Zero Operand Instructions				
R2	000_110	ret	ret	Return from subroutine
	001_110	reti	reti	Return from interrupt
	010_110	dint	dint	Disable interrups
	011_110	eint	eint	Enable interrupts
	100_110	halt	halt	Halts processor and sets it into program mode
	101_110	-	-	Reserved
	110_110	-	-	Reserved
	111_110			Reserved
Not Available				
R2	000_111. .111_111	-	-	Not Available