ADChips® SE3208 Instruction Set
Quick Reference Card

Key to Tables	
%Rc	Shift count register (GPR)
%Ri	Index register (GPR)
%Rs	Source register (GPR)
%Rd	Destination register (GPR)
imm	Immediate value (Maximum 32bit)
imm#	Immediate value (Maximum #bit)

<label>	Assembler label
/	Exclusively used
|	Concatenation
@unit	at unit
C	Carry flag
[address]	32bit data from address

Operation		V	Assembler	SR Mod.	Action
Load	Word		LD (%Ri/%SP, imm), %Rd		%Rd = [%Ri/%SP + imm]
	Byte		LDBU (%Ri/%SP, imm), %Rd		%Rd = ZeroExtent[Byte from (%Ri/%SP + imm)]
	signed		LDB (%Ri/%SP, imm), %Rd		%Rd = SignExtent[Byte from (%Ri/%SP + imm)]
	Half word (short)		LDSU (%Ri/%SP, imm), %Rd		%Rd = ZeroExtent[Short from (%Ri/%SP + imm)]
	signed		LDS (%Ri/%SP, imm), %Rd		%Rd = SignExtent[Short from (%Ri/%SP + imm)]
Load Multiple	Pop		POP <reg list>		while (all regs in reg list is poped) <lower num unpoped register in reg list> = [%SP], SP = SP + 4
Store	Word		ST %Rs, (%Ri/%SP, imm)		[%Ri/%SP + imm] = %Rs
	Byte		STB %Rs, (%Ri/%SP, imm)		[%Ri/%SP + imm][7:0] = %Rs[7:0]
	Half word (short)		STS %Rs, (%Ri/%SP, imm)		[%Ri/%SP + imm][15:0] = %Rs[15:0]
Store Multiple	Push		PUSH <reg list>		while (all regs in reg list is poped) [%SP = %SP -4] = <higher num unpushed register in reg list>
Move	Move		MOV %Rs, %Rd		%Rd = %Rs
	immediate		LDI imm, %Rd		%Rd = imm
	with add		LEA (%Rs/%SP, imm), %Rd/%SP		%Rd/%SP = %Rs/%SP + imm
Arithmetic	Add		ADD %Rs1, %Rs2/imm, %Rd	C Z S V	%Rd = %Rs1 + %Rs2/imm
	with carry		ADC %Rs1, %Rs2/imm, %Rd	C Z S V	%Rd = %Rs1 + %Rs2/imm + C
	Subtract		SUB %Rs1, %Rs2/imm, %Rd	C Z S V	%Rd = %Rs1 - %Rs2/imm
	with carry		SBC %Rs1, %Rs2/imm, %Rd	C Z S V	%Rd = %Rs1 - %Rs2/imm – C
	Multiply		MUL %Rs1, %Rs2, %Rd	V	%Rd = %Rs1 * %Rs2
Logical	Test		TST %Rs1/imm, %Rs2	Z S	Update SR flag on %Rs1 AND %Rs2
	AND		AND %Rs1, %Rs2/imm, %Rd	Z S	%Rd = %Rs1 AND %Rs2/imm
	OR		OR %Rs1, %Rs2/imm, %Rd	Z S	%Rd = %Rs1 OR %Rs2/imm
	XOR		XOR %Rs1, %Rs2/imm, %Rd	Z S	%Rd = %Rs1 XOR %Rs2/imm
	NEG(2's Complement)		NEG %Rs, %Rd	C Z S V	%Rd = - %Rs
Compare	Compare		CMP %Rs1, %Rs2/imm	C Z S V	Update SR flag on %Rs1 - %Rs2/imm
Shift	Arithmetic shift right		ASR %Rc/imm_5, %Rd	C Z S	{%Rd >> (%Rc/imm_5), %Rd@(MSB) = %Rd@(MSB)}
	Logical shift right		LSR %Rc/imm_5, %Rd	C Z S	%Rd >> (%Rc/imm_5)
	Arithmetic shift left		ASL %Rc/imm_5, %Rd	C Z S	%Rd << (%Rc/imm_5)
Format Conversion	Extension from byte to word		EXTB %Rd	Z S	SignExtent(%Rd[7:0])
	Extension from short to word		EXTS %Rd	Z S	SignExtent(%Rd[15:0])
SR control	Set a bit in SR		SET imm_4		SR.bit<imm_4> = 1 depend on processor mode
	Clear a bit in SR		CLR imm_4		SR.bit<imm_4> = 0 depend on processor mode

ADChips® SE3208 Instruction Set
Quick Reference Card

Operation		V	Assembler	SR Mod.	Action
Branch	Jump		`JMP <label>`		PC = address of <Label>
	on overflow clear		`JNV <label>`		if (V == 0) PC = address of <Label>
	on overflow set		`JV <label>`		if (V == 1) PC = address of <Label>
	on sign clear / positive or zero		`JP <label>`		if (S == 0) PC = address of <Label>
	on sign set / negative		`JM <label>`		if (S == 1) PC = address of <Label>
	on non-zero / not equal		`JNZ <label>`		if (Z == 0) PC = address of <Label>
	on zero / equal		`JZ <label>`		if (Z == 1) PC = address of <Label>
	on carry clear / unsigned higher or equal		`JNC <label>`		if (C == 0) PC = address of <Label>
	on carry set / unsigned lower		`JC <label>`		if (C == 1) PC = address of <Label>
	on signed greater		`JGT <label>`		if ((Z+S^V) == 0) PC = address of <Label>
	on signed less		`JLT <label>`		if ((S^V) == 1) PC = address of <Label>
	on signed greater or equal		`JGE <label>`		if ((S^V) == 0) PC = address of <Label>
	on signed less or equal		`JLE <label>`		if ((Z+S^V) == 1) PC = address of <Label>
	on unsigned higher		`JHI <label>`		if ((C+Z) == 0) PC = address of <Label>
	on unsigned lower or equal		`JLS <label>`		if ((C+Z) == 1) PC = address of <Label>
	register indirection		`JR %Rs`		PC = %Rs
	Call		`CALL <label>`		{%SP = PC, PC = address of <Label>}
	register indirection		`CALLR %Rs`		{%SP = PC, PC = %Rs}
Coprocessor	Move to GPR from coproc		`MVFC %Rs@CP`		%R0 = %Rs@CP
	Move to coproc from GPR		`MVTC %Rd@CP`		%Rd@CP = %R0
Soft Interrupt	Software Interrupt		`SWI imm_4`		Software interrupt processor exception
Halt			`HALT imm 4`		Halt for low power

* All immediate values use signed number except Shift Operations, SET, CLR, HALT, SWI and GETC/EXEC.