Intel® XScale™ Microarchitecture Assembly Language Quick Reference Card
ARM Instruction Set

Operation		Assembler	S updates	Action	Notes
Move	Move	MOV{cond}{S} Rd, <Oprnd2>	N Z C	Rd := Oprnd2	
	NOT	MVN{cond}{S} Rd, <Oprnd2>	N Z C	Rd := 0xFFFFFFFF EOR Oprnd2	
	SPSR to register	MRS{cond} Rd, SPSR		Rd := SPSR	
	CPSR to register	MRS{cond} Rd, CPSR		Rd := CPSR	
	register to SPSR	MSR{cond} SPSR_<fields>, Rm		SPSR := Rm (selected bytes only)	
	register to CPSR	MSR{cond} CPSR_<fields>, Rm		CPSR := Rm (selected bytes only)	
	immediate to SPSR	MSR{cond} SPSR_<fields>, #<immed_8r>		SPSR := immed_8r (selected bytes only)	
	immediate to CPSR	MSR{cond} CPSR_<fields>, #<immed_8r>		CPSR := immed_8r (selected bytes only)	
Arithmetic	Add	ADD{cond}{S} Rd, Rn, <Oprnd2>	N Z C V	Rd := Rn + Oprnd2	
	with carry	ADC{cond}{S} Rd, Rn, <Oprnd2>	N Z C V	Rd := Rn + Oprnd2 + Carry	
	saturating	QADD{cond} Rd, Rm, Rn		Rd := SAT(Rm + Rn)	Sticky. No shift/rotate.
	double saturating	QDADD{cond} Rd, Rm, Rn		Rd := SAT(Rm + SAT(Rn * 2))	Sticky. No shift/rotate.
	Subtract	SUB{cond}{S} Rd, Rn, <Oprnd2>	N Z C V	Rd := Rn - Oprnd2	
	with carry	SBC{cond}{S} Rd, Rn, <Oprnd2>	N Z C V	Rd := Rn - Oprnd2 - NOT(Carry)	
	reverse subtract	RSB{cond}{S} Rd, Rn, <Oprnd2>	N Z C V	Rd := Oprnd2 - Rn	
	reverse subtract with carry	RSC{cond}{S} Rd, Rn, <Oprnd2>	N Z C V	Rd := Oprnd2 - Rn - NOT(Carry)	
	saturating	QSUB{cond} Rd, Rm, Rn		Rd := SAT(Rm - Rn)	Sticky. No shift/rotate.
	double saturating	QDSUB{cond} Rd, Rm, Rn		Rd := SAT(Rm - SAT(Rn * 2))	Sticky. No shift/rotate.
	Multiply	MUL{cond}{S} Rd, Rm, Rs	N Z C	Rd := (Rm * Rs)[31:0]	
	accumulate	MLA{cond}{S} Rd, Rm, Rs, Rn	N Z C	Rd := ((Rm * Rs) + Rn)[31:0]	
	unsigned long	UMULL{cond}{S} RdLo, RdHi, Rm, Rs	N Z C V	RdHi,RdLo := unsigned(Rm * Rs)	
	unsigned accumulate long	UMLAL{cond}{S} RdLo, RdHi, Rm, Rs	N Z C V	RdHi,RdLo := unsigned(RdHi,RdLo + Rm * Rs)	
	signed long	SMULL{cond}{S} RdLo, RdHi, Rm, Rs	N Z C V	RdHi,RdLo := signed(Rm * Rs)	
	signed accumulate long	SMLAL{cond}{S} RdLo, RdHi, Rm, Rs	N Z C V	RdHi,RdLo := signed(RdHi,RdLo + Rm * Rs)	
	signed 16 * 16 bit	SMULxy{cond} Rd, Rm, Rs		Rd := Rm[x] * Rs[y]	No shift/rotate.
	signed 32 * 16 bit	SMULWy{cond} Rd, Rm, Rs		Rd := (Rm * Rs[y])[47:16]	No shift/rotate.
	signed accumulate 16 * 16	SMLAxy{cond} Rd, Rm, Rs, Rn		Rd := Rn + Rm[x] * Rs[y] Sticky.	No shift/rotate.
	signed accumulate 32 * 16	SMLAWy{cond} Rd, Rm, Rs, Rn		Rd := Rn + (Rm * Rs[y])[47:16]	Sticky. No shift/rotate.
	signed accumulate long 16 * 16	SMLALxy{cond} RdLo, RdHi, Rm, Rs		RdHi,RdLo := RdHi,RdLo + Rm[x] * Rs[y]	No shift/rotate.
	Count leading zeroes	CLZ{cond} Rd, Rm		Rd := number of leading zeroes in Rm	
DSP CP0	Multiply with internal accumulate	MIA{cond} acc0, Rm, Rs		acc0 = (Rm[31:0] * Rs[31:0])[39:0] + acc0[39:0])	
		MIAPH{cond} acc0, Rm, Rs		acc0 = sign_extend(Rm[31:16] * Rs[31:16]) + sign_extend(Rm[15:0] * Rs[15:0]) + acc0[39:0]	
		MIAxy{cond} acc0, Rm, Rs		acc0[39:0] = sign_extend(Rm[x] * Rs[y]) + acc0[39:0]	
	Accumulator move to	MAR{cond} acc0, RdLo, RdHi		acc0[39:32] = RdHi[7:0] acc0[31:0] = RdLo[31:0]	
	Accumulator move from	MRA{cond} RdLo, RdHi, acc0		RdHi[31:0] = sign_extend(acc0[39:32] RdLo[31:0] = acc0[31:0]	
Logical	Test	TST{cond} Rn, <Oprnd2>	N Z C	Update CPSR flags on Rn AND Oprnd2	
	Test equivalence	TEQ{cond} Rn, <Oprnd2>	N Z C	Update CPSR flags on Rn EOR Oprnd2	
	AND	AND{cond}{S} Rd, Rn, <Oprnd2>	N Z C	Rd := Rn AND Oprnd2	
	EOR	EOR{cond}{S} Rd, Rn, <Oprnd2>	N Z C	Rd := Rn EOR Oprnd2	
	ORR	ORR{cond}{S} Rd, Rn, <Oprnd2>	N Z C	Rd := Rn OR Oprnd2	
	Bit Clear	BIC{cond}{S} Rd, Rn, <Oprnd2>	N Z C	Rd := Rn AND NOT Oprnd2	
	No operation	NOP		R0 := R0	Flags not affected.
	Shift/Rotate				See Table Operand 2.
Compare	Compare	CMP{cond} Rn, <Oprnd2>	N Z C V	Update CPSR flags on Rn - Oprnd2	
	negative	CMN{cond} Rn, <Oprnd2>	N Z C V	Update CPSR flags on Rn + Oprnd2	
Branch	Branch	B{cond} label		R15 := label	label within ±32Mb
	with link	BL{cond} label		R14 := R15-4, R15 := label	label within ±32Mb
	and exchange	BX{cond} Rm		R15 := Rm, Change to Thumb if Rm[0] is 1	
	with link and exchange (1)	BLX label		R14 := R15 - 4, R15 := label	Cannot be conditional.
	with link and exchange (2)	BLX{cond}		Rm R14 := R15 - 4, R15 := Rm[31:1]	
Load	Word	LDR{cond} Rd, <a_mode2>		Rd := [address]	
	User mode privilege	LDR{cond}T Rd, <a_mode2P>			
	branch (and exchange)	LDR{cond} R15, <a_mode2>		R15 := [address][31:1], Change to Thumb if [address][0] is 1	
	Byte	LDR{cond}B Rd, <a_mode2>		Rd := ZeroExtend[byte from address]	
	User mode privilege	LDR{cond}BT Rd, <a_mode2P>			
	signed	LDR{cond}SB Rd, <a_mode3>		Rd := SignExtend[byte from address]	
	Halfword	LDR{cond}H Rd, <a_mode3>		Rd := ZeroExtent[halfword from address]	
	signed	LDR{cond}SH Rd, <a_mode3>		Rd := SignExtend[halfword from address]	
Load Multiple	Pop, or Block data load	LDM{cond}<a_mode4L> Rd{!}, <reglist-pc>		Load list of registers from [Rd]	
	return (and exchange)	LDM{cond}<a_mode4L> Rd{!}, <reglist+pc>		Load registers, R15 := [address][31:1]	
	and restore CPSR	LDM{cond}<a_mode4L> Rd{!}, <reglist+pc>^		Load registers, branch and exchange	Exception modes only.
	User mode registers	LDM{cond}<a_mode4L> Rd, <reglist-pc>^		Load list of User mode registers from [Rd]	Privileged modes only.
Load Double		LDRD			
Store	Word	STR{cond} Rd, <a_mode2>		[address] := R	
	User mode privilege	STR{cond}T Rd, <a_mode2P>		[address] := Rd	
	Byte	STR{cond}B Rd, <a_mode2>		[address][7:0] := Rd[7:0]	
	User mode privilege	STR{cond}BT Rd, <a_mode2P>		[address][7:0] := Rd[7:0	
	Halfword	STR{cond}H Rd, <a_mode3>		[address][15:0] := Rd[15:0]	
Store Multiple	Push, or Block data store	STM{cond}<a_mode4S>Rd{!}, <reglist>		Store list of registers to [Rd]	
	User mode registers	STM{cond}<a_mode4S> Rd{!}, <reglist>^		Store list of User mode registers to [Rd]	Privileged modes only.
Store Double		STRD			
Swap	Word	SWP{cond} Rd, Rm, [Rn]		temp := [Rn], [Rn] := Rm, Rd := temp	
	Byte	SWP{cond}B Rd, Rm, [Rn]		temp := ZeroExtend([Rn][7:0]), [Rn][7:0] := Rm[7:0], Rd := temp	
Coprocessor	Move to ARM reg from coproc	MRC{cond} p<cpnum>, <op1>, Rd, CRn, CRm, <op2>			
		MRRC{cond} p<cpnum>, <op1>, <Rd>, <Rn>, <CRn>			
	Move to coproc from ARM reg	MCR{cond} p<cpnum>, <op1>, Rd, CRn, CRm, <op2>			
		MCRR{cond} p<cpnum>, <op1>, <Rd>, <Rn>, <CRn>			
	Load coprocessor	LDC{cond} p<cpnum>, CRd, <a_mode5>		CRd := [address]	
	Store coprocessor	STC{cond} p<cpnum>, CRd, <a_mode5>		[address] := CRd	
Software interrupt		SWI{cond} <immed_24>		Software interrupt processor exception	24-bit value.
Breakpoint		BKPT <immed_16>		Prefetch abort or enter debug state	Cannot be conditional.
Pre-load		PLD <a_mode2>		Pre-load cache line	

Intel® XScale™ Microarchitecture Assembly Language Quick Reference Card
Thumb Instruction Set

Operation		Assembler	Action	Notes
Move	Immediate	MOV Rd, #<immed_8>	Rd := immed_8	8-bit immediate value.
	Lo to Lo	MOV Rd, Rm	Rd := Rm	
	Hi to Lo, Lo to Hi, Hi to Hi	MOV Rd, Rm	Rd := Rm	Not Lo to Lo
Arithmetic	Add	ADD Rd, Rn, #<immed_3>	Rd := Rn + immed_3	3-bit immediate value.
	Lo and Lo	ADD Rd, Rn, Rm	Rd := Rn + Rm	
	Hi to Lo, Lo to Hi, Hi to Hi	ADD Rd, Rm	Rd := Rd + Rm	Not Lo to Lo
	immediate	ADD Rd, #<immed_8>	Rd := Rd + immed_8	8-bit immediate value.
	with carry	ADC Rd, Rm	Rd := Rd + Rm + C-bit	
	value to SP	ADD SP, #<immed_7*4>	SP := SP + immed_7 * 4	9-bit immediate value (word-aligned).
	form address from SP	ADD Rd, SP, #<immed_8*4>	Rd := SP + immed_8 * 4	10-bit immediate value (word-aligned).
	form address from PC	ADD Rd, PC, #<immed_8*4>	Rd := (PC AND 0xFFFFFFFC) + immed_8 * 4	10 bit immediate value (word-aligned).
	Subtract	SUB Rd, Rn, Rm	Rd := Rn - Rm	
	immediate 3	SUB Rd, Rn, #<immed_3>	Rd := Rn - immed_3	3-bit immediate value.
	immediate 8	SUB Rd, #<immed_8>	Rd := Rd - immed_8	8-bit immediate value.
	with carry	SBC Rd, Rm	Rd := Rd - Rm - NOT C-bit	
	value from SP	SUB SP, #<immed_7*4>	SP := SP - immed_7 * 4	9-bit immediate value (word-aligned).
	Negate	NEG Rd, Rm	Rd := - Rm	
	Multiply	MUL Rd, Rm	Rd := Rm * Rd	
	Compare	CMP Rn, Rm	update CPSR flags on Rn - Rm	Can be Lo to Lo, Lo to Hi, Hi to Lo, or Hi to Hi.
	negative	CMN Rn, Rm	update CPSR flags on Rn + Rm	
	immediate	CMP Rn, #<immed_8>	update CPSR flags on Rn - immed_8	8-bit immediate value.
	No operation	NOP	R8 := R8	lags not affected.
Logical	AND	AND Rd, Rm	Rd := Rd AND Rm	
	Exclusive OR	EOR Rd, Rm	Rd := Rd EOR Rm	
	OR	ORR Rd, Rm	Rd := Rd OR Rm	
	Bit clear	BIC Rd, Rm	Rd := Rd AND NOT Rm	
	Move NOT	MVN Rd, Rm	Rd := NOT Rm	
	Test bits	TST Rn, Rm	update CPSR flags on Rn AND Rm	
Shift / Rotate	Logical shift left	LSL Rd, Rm, #<immed_5>	Rd := Rm << immed_5	5-bit immediate shift. Allowed shifts 0-31.
	LSL Rd, Rs	Rd := Rd << Rs		
	Logical shift right	LSR Rd, Rm, #<immed_5>	Rd := Rm >> immed_5	5-bit immediate shift. Allowed shifts 1-32.
	LSR Rd, Rs	Rd := Rd >> Rs		
	Arithmetic shift right	ASR Rd, Rm, #<immed_5>	Rd := Rm ASR immed_5	5-bit immediate shift. Allowed shifts 1-32.
	ASR Rd, Rs	Rd := Rd ASR Rs		
	Rotate right	ROR Rd, Rs	Rd := Rd ROR Rs	
Branch	Conditional branch	B{cond} label See Table Condition Field (ARM side). AL not allowed.	R15 := label	label must be within -252 to +258 bytes
	Unconditional branch	B label	R15 := label	label must be within ±2Kb of current instruction.
	Long branch with link	BL label label must be within ±4Mb of current instruction.	R14 := R15 - 2, R15 := label	Encoded as two Thumb instructions.
	Branch and exchange	BX Rm	R15 := Rm AND 0xFFFFFFFE	Change to ARM state if Rm[0] = 0.
	Branch with link and exchange	BLX label	R14 := R15 - 2, R15 := label	Encoded as two Thumb instructions.
	Change to ARM	label must be within ±4Mb of current instruction.		
	Branch with link and exchange	BLX Rm	R14 := R15 - 2, R15 := Rm AND 0xFFFFFFFE	
	Change to ARM if Rm[0] = 0			
Software Interrupt		<immed_8>	Software interrupt processor exception	8-bit immediate value encoded in instruction.
Breakpoint		BKPT <immed_8>	Prefetch abort or enter debug state	
Load	with immediate offset, word	LDR Rd, [Rn, #<immed_5*4>]	Rd := [Rn + immed_5 * 4]	
	halfword	LDRH Rd, [Rn, #<immed_5*2>]	Rd := ZeroExtend([Rn + immed_5 * 2][15:0])	Clears bits 31:16
	byte	LDRB Rd, [Rn, #<immed_5>]	Rd := ZeroExtend([Rn + immed_5][7:0])	Clears bits 31:8
	with register offset, word	LDR Rd, [Rn, Rm]	Rd := [Rn + Rm]	
	halfword	LDRH Rd, [Rn, Rm]	Rd := ZeroExtend([Rn + Rm][15:0])	Clears bits 31:16
	signed halfword	LDRSH Rd, [Rn, Rm]	Rd := SignExtend([Rn + Rm][15:0])	Sets bits 31:16 to bit 15
	byte	LDRB Rd, [Rn, Rm]	Rd := ZeroExtend([Rn + Rm][7:0])	Clears bits 31:8
	signed byte	LDRSB Rd, [Rn, Rm]	Rd := SignExtend([Rn + Rm][7:0])	Sets bits 31:8 to bit 7
	PC-relative	LDR Rd, [PC, #<immed_8*4>]	Rd := [(PC AND 0xFFFFFFFC) + immed_8 * 4]	
	SP-relative	LDR Rd, [SP, #<immed_8*4>]	Rd := [SP + immed_8 * 4]	
	Multiple	LDMIA Rn!, <reglist>	Loads list of registers	Always updates base register.
Store	with immediate offset, word	STR Rd, [Rn, #<immed_5*4>]	[Rn + immed_5 * 4] := Rd	
	halfword	STRH Rd, [Rn, #<immed_5*2>]	[Rn + immed_5 * 2][15:0] := Rd[15:0]	Ignores Rd[31:16]
	byte	STRB Rd, [Rn, #<immed_5>]	[Rn + immed_5][7:0] := Rd[7:0]	Ignores Rd[31:8]
	with register offset, word	STR Rd, [Rn, Rm]	[Rn + Rm] := Rd	
	halfword	STRH Rd, [Rn, Rm]	[Rn + Rm][15:0] := Rd[15:0]	Ignores Rd[31:16]
	byte S	STRB Rd, [Rn, Rm]	[Rn + Rm][7:0] := Rd[7:0]	Ignores Rd[31:8]
	SP-relative, word	STR Rd, [SP, #<immed_8*4>]	[SP + immed_8 * 4] := Rd	
	Multiple	STMIA Rn!, <reglist>	Stores list of registers	Always updates base register.
Push	Push	PUSH <reglist>	Push registers onto stack	Full descending stack.
	Push with link	PUSH <reglist, LR>	Push LR and registers on to stack	
Pop	Pop	POP <reglist>	Pop registers from stack	
	Pop and return	POP <reglist, PC>	Pop registers, branch to address loaded to PC	
	Pop and return with exchange	POP <reglist, PC>	Pop, branch, and change to ARM state if address[0] = 0	

All Thumb registers are Lo (R0-R7) except where specified. Hi registers are R8-R15.

Intel® XScale™ Microarchitecture Assembly Language Quick Reference Card
Key to Tables

CODES

{cond}	Refer to Table Condition Field {cond}
<Oprnd2>	Refer to Table Operand 2
<fields>	Refer to Table PSR fields
{S}	Updates condition flags if S present
Sticky	Sticky flag, updates on overflow (no S option), read and reset using MRS and MSR
x,y	B meaning half-register [15:0], or T meaning [31:16]
<immed_8r>	A 32-bit constant, formed by right-rotating an 8-bit value by an even number of bits
<immed_8*4>	A 10-bit constant, formed by left-shifting an 8-bit value by two bits
<a_mode2>	Refer to Table Addressing Mode 2
<a_mode2P>	Refer to Table Addressing Mode 2 (Post-indexed only)
<a_mode3>	Refer to Table Addressing Mode 3
<a_mode4L>	Refer to Table Addressing Mode 4 (Block load or Stack pop)
<a_mode5>	Refer to Table Addressing Mode 5
<reglist>	A list of registers, enclosed in braces ({ and })
{!}	Updates base register after data transfer if ! present

CONDITION FIELD {COND}

Mnemonic	Description
EQ	Equal
NE	Not equal
CS / HS	Carry Set / Unsigned higher or same
CC / LO	Carry Clear / Unsigned lower
MI	Negative
PL	Positive or zero
VS	Overflow
VC	No overflow
HI	Unsigned higher
LS	Unsigned lower or same
GE	Signed greater than or equal
LT	Signed less than
GT	Signed greater than
LE	Signed less than or equal
AL	Always (normally omitted)

OPERAND 2

Immediate value	#<immed_8r>	
Logical shift left immediate	Rm, LSL #<immed_5>	Allowed shifts 0-31
Logical shift right immediate	Rm, LSR #<immed_5>	Allowed shifts 1-32
Arithmetic shift right immediate	Rm, ASR #<immed_5>	Allowed shifts 1-32
Rotate right immediate	Rm, ROR #<immed_5>	Allowed shifts 1-31
Register	Rm	
Rotate right extended	Rm, RRX	
Logical shift left register	Rm, LSL Rs	
Logical shift right register	Rm, LSR Rs	
Arithmetic shift right register	Rm, ASR Rs	
Rotate right register	Rm, ROR Rs	

PSR FIELDS (USE AT LEAST ONE SUFFIX)

Suffix	Meaning	
c	Control field mask byte	PSR[7:0]
f	Flags field mask byte	PSR[31:24]
s	Status field mask byte	PSR[23:16]
x	Extension field mask byte	PSR[15:8]

ARM ADDRESSING MODES

ADDRESSING MODE 2 - WORD AND UNSIGNED BYTE DATA TRANSFER

Pre-indexed	Immediate offset	[Rn, #+/-<immed_12>]{!}	
	Zero offset	[Rn]	Equivalent to [Rn,#0]
	Register offset	[Rn, +/-Rm]{!}	
	Scaled register offset	[Rn, +/-Rm, LSL #<immed_5>]{!}	Allowed shifts 0-31
		[Rn, +/-Rm, LSR #<immed_5>]{!}	Allowed shifts 1-32
		[Rn, +/-Rm, ASR #<immed_5>]{!}	Allowed shifts 1-32
		[Rn, +/-Rm, ROR #<immed_5>]{!}	Allowed shifts 1-31
		[Rn, +/-Rm, RRX]{!}	
Post-indexed	Immediate offset	[Rn], #+/-<immed_12>	
	Register offset	[Rn], +/-Rm DB	
	Scaled register offset	[Rn], +/-Rm, LSL #<immed_5>	Allowed shifts 0-31
		[Rn], +/-Rm, LSR #<immed_5>	Allowed shifts 1-32
		[Rn], +/-Rm, ASR #<immed_5>	Allowed shifts 1-32
		[Rn], +/-Rm, ROR #<immed_5>	Allowed shifts 1-31
		[Rn], +/-Rm, RRX	

ADDRESSING MODE 2 (POST-INDEXED ONLY)

Post-indexed	Immediate offset	[Rn], #+/-<immed_12>	
	Zero offset	[Rn]	Equivalent to [Rn],#0
	Register offset	[Rn], +/-Rm	
	Scaled register offset	[Rn], +/-Rm, LSL #<immed_5>	Allowed shifts 0-31
		[Rn], +/-Rm, LSR #<immed_5>	Allowed shifts 1-32
		[Rn], +/-Rm, ASR #<immed_5>	Allowed shifts 1-32
		[Rn], +/-Rm, ROR #<immed_5>	Allowed shifts 1-31
		[Rn], +/-Rm, RRX	

ADDRESSING MODE 3 - HALFWORD AND SIGNED BYTE DATA TRANSFER

Pre-indexed	Immediate offset	[Rn, #+/-<immed_8>]{!}	
	Zero offset	[Rn]	Equivalent to [Rn,#0]
	Register	[Rn, +/-Rm]{!}	
Post-indexed	Immediate offset	[Rn], #+/-<immed_8>	
	Register	[Rn], +/-Rm	

ADDRESSING MODE 5 - COPROCESSOR DATA TRANSFER

Pre-indexed	Immediate offset	[Rn, #+/-<immed_8*4>]{!}	
	Zero offset	[Rn]	Equivalent to [Rn,#0]
Post-indexed	Immediate offset	[Rn], #+/-<immed_8*4>	
Unindexed	No offset	[Rn], {8-bit copro. option}	

ADDRESSING MODE 4 - MULTIPLE DATA TRANSFER

Block load	
IA	Increment After
IB	Increment Before
DA	Decrement After
DB	Decrement Before
Block store	
IA	Increment After
IB	Increment Before
DA	Decrement After
DB	Decrement Before
Stack pop	
FD	Full Descending
ED	Empty Descending
FA	Full Ascending
EA	Empty Ascending
Stack push	
EA	Empty Ascending
FA	Full Ascending
ED	Empty Descending
FD	Full Descending

Intel Access

Developer's Site	developer.intel.com
I/O Building Blocks Home Page	developer.intel.com/design/iio
Intel® XScale™ Microarchitecture	developers.intel.com/design/XScale
Other Intel Support	developer.intel.com/design/litcentr
Intel Literature Center	(800) 548-4725 7a.m. to 7 p.m. CST (U.S. and Canada) International locations please contact your local sales office.
General Information Hotline	(800) 628-8686 or (916) 356-3104 5 a.m. to 5 p.m. PST

For more information, visit the Intel Web site at: **developer.intel.com**

UNITED STATES AND CANADA
Intel Corporation
Robert Noyce Bldg.
2200 Mission College Blvd.
P.O. Box 58119
Santa Clara, CA 95052-8119
USA

EUROPE
Intel Corporation (UK) Ltd.
Pipers Way
Swindon
Wiltshire SN3 1RJ
UK

ASIA-PACIFIC
Intel Semiconductor Ltd.
32/F Two Pacific Place
88 Queensway, Central
Hong Kong, SAR

JAPAN
Intel Kabushiki Kaisha
P.O. Box 115 Tsukuba-gakuen
5-6 Tokodai, Tsukuba-shi
Ibaraki-ken 305
Japan

SOUTH AMERICA
Intel Semicondutores do Brazil
Rue Florida, 1703-2 and CJ22
CEP 04565-001 Sao Paulo-SP
Brazil

Printed on Recycled Paper

Printed in USA/0801/5K/IL11139 KG