APPENDIX : ARM Assembly Instructions

Key to Tables	
{cond}	Refer to Table **Condition Field {cond}**
<Oprnd2>	Refer to Table **Oprnd2**
{field}	Refer to Table **Field**
S	Sets condition codes (optional)
B	Byte operation (optional)
H	Halfword operation (optional)
T	Forces address translation. Cannot be used with pre-indexed addresses
<a_mode1>	Refer to Table **Addressing Mode 1**
<a_mode2>	Refer to Table **Addressing Mode 2**
<a_mode3>	Refer to Table **Addressing Mode 3**
<a_mode4>	Refer to Table **Addressing Mode 4**
<a_mode5>	Refer to Table **Addressing Mode 5**
<a_mode6>	Refer to Table **Addressing Mode 6**
#32_Bit_Immed	A 32-bit constant, formed by right-rotating an 8-bit value by an even number of bits

Operation		Assembler	S updates	Action
Move	Move	MOV{cond}{S} Rd, <Oprnd2>	N Z V	Rd:= <Oprnd2>
	NOT	MVN{cond}{S} Rd, <Oprnd2>	N Z V	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{field}, Rm		SPSR:= Rm
	register to CPSR	MSR{cond} CPSR{field}, Rm		CPSR:= Rm
	immediate to SPSR flags	MSR{cond} SPSR_f, 32_Bit_Immed		SPSR:= #32_Bit_Immed
	immediate to CPSR flags	MSR{cond} CPSR_f, 32_Bit_Immed		CPSR:= #32_Bit_Immed
ALU	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
	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)
	Multiply	MUL{cond}{S} Rd, Rm, Rs		Rd:= Rm * Rs
	accumulate	MLA{cond}{S} Rd, Rm, Rs, Rn,	N Z	Rd:= (Rm * Rs) + Rn
	unsigned long	UMULL{cond}{S} RdLo, RdHi, Rm, Rs	N Z	RdHi:= (Rm*Rs)[63:32] RdLo:= (Rm*Rs)[31:0]
	unsigned accumulate long	UMLAL{cond}{S} RdLo, RdHi, Rm, Rs	N Z	RdLo:=(Rm*Rs)+RdLo RdHi:=(Rm*Rs)+RdHi+ CarryFrom((Rm*Rs)[31:0]+RdLo))
	signed long	SMULL{cond}{S}, RdLo, RdHi, Rm, Rs	N Z	RdHi:= signed(Rm*Rs)[63:32] RdLo:= signed(Rm*Rs)[31:0]
	signed accumulate long	SMLAL{cond}{S}, RdLo, RdHi, Rm, Rs	N Z	RdHi:=signed(Rm*Rs)+RdHi+ CarryFrom((Rm*Rs)[31:0]+RdLo))
	Compare	CMP{cond} Rd, <Oprnd2>	N Z C V	CPSR flags:= Rn - <Oprnd2>
	negative	CMN{cond} Rd, <Oprnd2>	N Z C V	CPSR flags:= Rn + <Oprnd2>
	Logical			
	Test	TST{cond} Rn, <Oprnd2>	N Z C	CPSR flags:= Rn AND <Oprnd2>
	Test equivalence	TEQ{cond} Rn, <Oprnd2>	N Z C	CPSR flags:= 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>
	Shift/Rotate			See Table **Oprnd2**

Operation		Assembler	Action
Branch	Branch with link and exchange instruction set	B{cond} label BL{cond} label BX{cond} Rn	R15:= address R14:=R15, R15:= address R15:=Rn, T bit:= Rn[0]
Load	Word with user-mode privilege Byte with user-mode privilege signed Halfword signed Multiple Block data operations Increment Before Increment After Decrement Before Decrement After Stack operations and restore CPSR User registers	LDR{cond} Rd, <a_mode1> LDR{cond}T Rd, <a_mode2> LDR{cond}B Rd, <a_mode1> LDR{cond}BT Rd, <a_mode2> LDR{cond}SB Rd, <a_mode3> LDR{cond}H Rd, <a_mode3> LDR{cond}SH Rd, <a_mode3> LDM{cond}IB Rd{!}, <regs>{^} LDM{cond}IA Rd{!}, <regs>{^} LDM{cond}DB Rd{!}, <regs>{^} LDM{cond}DA Rd{!}, <regs>{^} LDM{cond}<a_mode4>Rd{!}, <registers> LDM{cond}<a_mode4> Rd{!}, <registers+pc> LDM{cond}<a_mode4> Rd, <registers>^	Rd:= [address] Rd:= [byte value from address] Loads bits 0 to 7 and sets bits 8-31 to 0 Rd:= [signed byte value from address] Loads bits 0 to 7 and sets bits 8-31 to bit 7 Rd:= [halfword value from address] Loads bits 0 to 15 and sets bits 16-31 to 0 Rd:= [signed halfword value from address] Loads bits 0 to 15 and sets bits 16-31 to bit 15 Stack manipulation (pop)
Store	Word with user-mode privilege Byte with user-mode privilege Halfword Multiple Block data operations Increment Before Increment After Decrement Before Decrement After Stack operations User registers	STR{cond} Rd, <a_mode1> STRT{cond} Rd, <a_mode2> STRB{cond} Rd, <a_mode1> STRBT{cond} Rd, <a_mode2> STR{cond}H Rd, <a_mode3> STM{cond}IB Rd{!}, <registers>{^} STM{cond}IA Rd{!}, <registers>{^} STM{cond}DB Rd{!}, <registers>{^} STM{cond}DA Rd{!}, <registers>{^} STM{cond}<a_mode5> Rd{!}, <regs> STM{cond}<a_mode5> Rd{!}, <regs>^	[address]:= Rd [address]:= byte value from Rd [address]:= halfword value from Rd Stack manipulation (push)
Swap	Word Byte	SWP{cond} Rd, Rm, [Rn] SWP{cond}B Rd, Rm, [Rn]	
Software Interrupt		SWI #24_Bit_Value	24-bit immediate value

Field	
Suffix	**Sets**
_c	Control field mask bit (bit 3)
_f	Flags field mask bit (bit 0)
_s	Status field mask bit (bit 1)
x	Extension field mask bit (bit 2)

Oprnd2	
Immediate value	#32_Bit_Immed
Logical shift left	Rm LSL #5_Bit _Immed
Logical shift right	Rm LSR #5_Bit _Immed
Arithmetic shift right	Rm ASR #5_Bit _Immed
Rotate right	Rm ROR #5_Bit _Immed
Register	Rm
Logical shift left	Rm LSL Rs
Logical shift right	Rm LSR Rs
Arithmetic shift right	Rm ASR Rs
Rotate right	Rm ROR Rs
Rotate right extended	Rm RRX

Addressing Mode 1	
Immediate offset	[Rn, #+/-12_Bit_Offset]
Register offset	[Rn, +/-Rm]
Scaled register offset	[Rn, +/-Rm, LSL #shift_imm]
	[Rn, +/-Rm, LSR #shift_imm]
	[Rn, +/-Rm, ASR #shift_imm]
	[Rn, +/-Rm, ROR #shift_imm]
	[Rn, +/-Rm, RRX]
Pre-indexed offset	
Immediate	[Rn, #+/-12_Bit_Offset]!
Register	[Rn, +/-Rm]!
Scaled register	[Rn, +/-Rm, LSL #shift_imm]!
	[Rn, +/-Rm, LSR #shift_imm]!
	[Rn, +/-Rm, ASR #shift_imm]!
	[Rn, +/-Rm, ROR #shift_imm]!
	[Rn, +/-Rm, RRX]!
Post-indexed offset	
Immediate	[Rn], #+/-12_Bit_Offset
Register	[Rn], +/-Rm
Scaled register	[Rn], +/-Rm, LSL #shift_imm
	[Rn], +/-Rm, LSR #shift_imm
	[Rn], +/-Rm, ASR #shift_imm
	[Rn], +/-Rm, ROR #shift_imm
	[Rn, +/-Rm, RRX]

Addressing Mode 3 – Signed Byte and Halfword Data Transfer	
Immediate offset	[Rn, #+/-8_Bit_Offset]
Pre-indexed	[Rn, #+/-8_Bit_Offset]!
Post-indexed	[Rn], #+/-8_Bit_Offset
Register	[Rn, +/-Rm]
Pre-indexed	[Rn, +/-Rm]!
Post-indexed	[Rn], +/-Rm

Condition Field {cond}	
Suffix	**Description**
EQ	Equal
NE	Not Equal
CS	Unsigned higher or same
CC	Unsigned lower
MI	Negative
PL	Positive or zero
VS	Overflow
VC	No overflow
HI	Unsigned higher
LS	Unsigned lower or same
GE	Greater or equal
LT	Less than
GT	Greater than
LE	Less than or equal
AL	Always

Addressing Mode 2	
Immediate offset	[Rn, #+/-12_Bit_Offset]
Register offset	[Rn, +/-Rm]
Scaled register offset	[Rn, +/-Rm, LSL #shift_imm]
	[Rn, +/-Rm, LSR #shift_imm]
	[Rn, +/-Rm, ASR #shift_imm]
	[Rn, +/-Rm, ROR #shift_imm]
	[Rn, +/-Rm, RRX]
Post-indexed offset	
Immediate	[Rn], #+/-12_Bit_Offset
Register	[Rn], +/-Rm
Scaled register	[Rn], +/-Rm, LSL #shift_imm
	[Rn], +/-Rm, LSR #shift_imm
	[Rn], +/-Rm, ASR #shift_imm
	[Rn], +/-Rm, ROR #shift_imm
	[Rn, +/-Rm, RRX]

Addressing Mode 4			
Addressing Mode		**Stack Type**	
IA	Increment After	FD	Full Descending
IB	Increment Before	ED	Empty Descending
DA	Decrement After	FA	Full Ascending
DB	Decrement Before	EA	Empty Ascending

Addressing Mode 5			
Addressing Mode		**Stack Type**	
IA	Increment After	EA	Empty Ascending
IB	Increment Before	FA	Full Ascending
DA	Decrement After	ED	Empty Descending
DB	Decrement Before	FD	Full Descending

END OF PAPER