Sous ensemble du jeu d'instruction ARM – Cortex – M3

ESIEE PARIS

Sous ensemble du jeu d'instruction – ARM – Cortex – M3

1. Opérations Arithmétiques

ADD: Add values and store result to register

Syntax	ADD{cond}{S} Rd, Rn, Op2
Description	Add Rn and Op2 and store result to Rd.
Condition Flags	If S is specified update flags: N, Z, C, V.

Example	ADDS R0,R2,R4
	//Add R2 and R4 and store result to R0, update flags
	ADD R4,R4,#0xFF00
	//Add value in 0xFF00 and R4 and store result in R4

Flexible Operand Op2

Several ARM instructions contain an **Op2** field that is called flexible operand. **Op2** may be one of the following:

- **#imm8_r**: 8-bit constant that may be rotated by an even number (0, 2, 4, 6, .., 30).
  Valid Examples: #0x12, #0xFF, #0x2FC0, #0xC000003F,
  Invalid Examples: #0x1FF, #0x1FE, #0x1234.
  Note: the Assembler uses Instruction Substitution (see below) in case that a constant cannot be represented.
- **Rm**: CPU register content of **Rm**.
  Example: R5
- **Rm, ASR #n**: Content of **Rm** shifted right with sign extension by #n bits; valid range for #n: 1 - 32.
  Example: R2, ASR #4
- **Rm, LSL #n**: Content of **Rm** shifted left with zero extension by #n bits; valid range for #n: 0 - 31.
  Example: R3, LSL #8
- **Rm, LSR #n**: Content of **Rm** shifted right with zero extension by #n bits; valid range for #n: 1 - 32.
  Example: R3, LSR #4
- **Rm, ROR #n**: Content of **Rm** rotated right by #n bits; valid range for #n: 1 - 31.
  Example: R2, ROR #6
- **Rm, RRX**: Content of **Rm** rotated right with C-flag extension by one bit.
  Example: R2, RRX
- **Rm, ASR Rs**: Content of **Rm** shifted right with sign extension by the content of Rs.
  Example: R2, ASR R3
- **Rm, LSL Rs**: Content of **Rm** shifted left with zero extension by the content of Rs.
  Example: R3, LSL R8
- **Rm, LSR Rs**: Content of **Rm** shifted right with zero extension by the content of Rs.
  Example: R3, LSR R4

- Rm, **ROR** Rs: Content of Rm rotated right by the content of Rs.
  Example: R2, ROR R6

2. Instructions arithmétiques

The instruction pairs ADD and SUB, ADC and SBC, AND and BIC, MOV and MVN, CMP and CMN, are equivalent except for the inversion or negation of **Op2**. If an**#imm8_r** value in **Op2** cannot be represented, the ARM Assembler inverts or negates the constant. If this constant can be represented in an**#imm8_r** value, a complementary instruction is automatically used.

ADC: Add with Carry

Syntax	ADC{cond}{S} Rd, Rn, Op2
Description	Add Rn and Op2 and Carry flag and store result to Rd. ADC is typical used for multi-word arithmetic.
Condition Flags	If S is specified update flags: N, Z, C, V.

Example	ADDS R0,R2,R4
	// add R2 + R4, store result to R0, set flags
	ADC R1,R3,R5
	// add R3 + R5 with carry from previous ADDS, store result to R1

SUB: Subtract registers

Syntax	SUB{cond}{S} Rd, Rn, Op2
Description	subtracts the value of Op2 from the value in Rn.
Condition Flags	If S is specified update flags: N, Z, C, V.
Example	SUBS R8,R6,#240 //R8=R6-240

SBC: Subtract with carry

Syntax	SBC{cond}{S} Rd, Rn, Op2
Description	synthesize multiword arithmetic.
Condition Flags	If S is specified update flags: N, Z, C, V.
Example	SUBS R0,R2,R4 SUB R4,R4,#5

MUL: Multiply (32-bit by 32-bit, bottom 32-bit result)

Syntax	MUL{cond}{S} Rd, Rm, Rs
Description	multiplies the values from Rm and Rs, and places the least significant 32 bits of the result in Rd.
Condition Flags	If S is specified: ▪ N and Z flags according to the result. ▪ the C flag in ARM architecture v4 and earlier will be corrupted. ▪ the C flag in ARM architecture v5 and later is not affected.
Example	MUL R10, R2, R5 //R10:= R2*R5

3. Opérations Logiques : ET, OU, OU Exclusif, opérations de décalage

AND: Logical AND operation

Syntax	AND{cond}{S} Rd, Rn, Op2
Description	Load Rd with logical AND of Rn with Op2. Rd := Rn AND Op2
Condition Flags	If S is specified, N, Z flags are updated. C flag may be updated by calculation of Op2.
Example	AND R9,R2,#0xFF00 // Load R9 with R2 and value in 0xFF00

ORR: Logical OR operation

Syntax	*ORR{cond}{S} Rd, Rn, Op2*
Description	*OR operations on the values in Rn and Op2.*
Condition Flags	*If S is specified, N, Z flags are updated. C flag may be updated by calculation of Op2.*

Example	*ORR R2, R0, R5* *// Rd = R0 or R5*

EOR: Logical Exclusive OR operation

Syntax	EOR {cond}Rd, Rn, Op2
Description	performs a logical Exclusive OR operation
Condition Flags	N and Z flags are updated. The C flag may be updated by calculation of Op2.
Example	EORS R0, R0, R3, ROR R6

ASR (Thumb)

Arithmetic (signed) Shift Right. The sign bit (bit position 31) is shift in on the right side.

Syntax	**ASR Rd, Rs** **ASR Rd, Rm, #imm**
Description	**ASR Rd, Rs** Arithmetic shift right value in Rd. Only the low byte in Rs is used as shift value. If shift value is 32, Rd is cleared. If shift value is greater than 32, Rd and C are cleared. **ASR Rd, Rm, #imm** Arithmetic shift right value in Rm and store result to Rd. **#imm** specifics a constant shift value in the range 1 - 31.
Registers	Supports only low registers (R0 - R7).
Condition Flags	Update N and Z; C is unaffected if shift value is zero, otherwise C contains last bit shifted out of Rd.
Example	ASR R0,R2,#6 // R0 = (signed) R2 >> 6 ASR R5,R2 // R5 = (signed) R5 >> R2, only the low byte in R2 is used.

TEQ: Bitwise Exclusive OR operation, result discarded. Used for conditional operations afterwards

Syntax	**TEQ{cond} Rn, Op2**
Description	Exclusive OR of the values Rn and Op2.
Condition Flags	N and Z flags are updated according the result. C flag may be updated during the calculation of Op2.

Example	TEQS R4, #3 //Test R4 for equality with 3

TST: Test

Syntax	TST{cond} Rn, Op2
Description	performs a bitwise AND operation on the value in Rn and the value of Op2. This is similar to the ANDS instruction, except that the result is discarded.
Condition Flags	N and Z flags are updated according the result. C flag may be updated during the calculation of Op2.
Example	TSTNE r1,r5,ASR r1

BIC: Bit Clear

Syntax	BIC{cond}{S} Rd, Rn, Op2
Description	Perform an AND operation on the bits in Rn with the complements of the corresponding bits in the value of Op2
Condition Flags	If S is specified, N, Z flags are updated. C flag may be updated by calculation of Op2.
Example	R0,R0 #0x1F //Clear mode bits

4. Instruction de comparaison – Instruction de branchement conditionnelle

CMP: Compare. Used in combination with conditional branch instructions

Syntax	CMP {cond} Rn, Op2
Description	subtracts the value of Op2 from the value in Rn (equals to the SUBS

	instruction with a discarded result). This instruction updates the condition flags, but do not place a result in a register.
Condition Flags	N, Z, C and V flags are updated.
Example	CMP R2, R9 //Subtract value of R9 from R2

B: Branch to label. Used to jump to a specific program location

Syntax	B{cond} label
Description	The jump distance must be within -252 to +258 bytes for conditional and ±2 KBytes for unconditional branch.
Condition Flags	not modified.
Example	CMP R1,#10 // compare R10 with #10 BEQ val_ok // jump to label val_ok val_ok: val_err: B val_err // jump to itself (loop forever)

5. Instructions lecture (Load)/écriture (Store) - mémoire

LDR: Load 32-bit word to Memory

Syntax	LDR{cond} Rd, [Rn] LDR{cond} Rd, [Rn, offset] LDR{cond} Rd, [Rn, offset]! LDR{cond} Rd, label LDR{cond} Rd, [Rn], offset
Description	LDR{cond} Rd, [Rn] (zero offset) Rn is used as address value. LDR{cond} Rd, [Rn, offset] (Pre-indexed offset) Rn and offset are added and used as address value. LDR{cond} Rd, [Rn, offset]{!} (Pre-indexed offset with update) Rn and offset are added and used as address value. The new address value is written to Rn. LDR{cond} Rd, label (Program-relative) The assembler calculates the PC offset and generates LDR{cond} Rd, [R15, offset].

	LDR{cond} Rd, [Rn], offset (Post-indexed offset) Rn is used as address value. After memory transfer, the offset is added to Rn.
Example	LDR R8,[R10] //loads r8 from the address in r10. LDRNE R2,[R5,#960]! //(conditionally) loads r2 from a word 960 bytes above the address in r5, and increments r5 by 960. LDR R0,localdata //loads a word located at label localdata

LDRB: Load register byte value to Memory

Syntax	LDR{cond}B Rd, [Rn] LDR{cond}B Rd, [Rn, offset] LDR{cond}B Rd, [Rn, offset]! LDR{cond}B Rd, label LDR{cond}B Rd, [Rn], offset
Description	LDR{cond}B Rd, [Rn] (zero offset) Rn is used as address value. LDR{cond}B Rd, [Rn, offset] (Pre-indexed offset) Rn and offset are added and used as address value. LDR{cond}B Rd, [Rn, offset]! (Pre-indexed offset with update) Rn and offset are added and used as address value. The new address value is written to Rn. LDR{cond}B Rd, label (Program-relative) The assembler calculates the PC offset and generates LDR{cond}B Rd, [R15, offset]. LDR{cond}B Rd, [Rn], offset (Post-indexed offset) Rn is used as address value. After memory transfer, the offset is added to Rn.

Example	LDR r8,[r10] //loads r8 from the address in r10.
	LDRNE r2,[r5,#960]! //(conditionally) loads r2 from a word
	//960 bytes above the address in r5, and
	//increments r5 by 960.
	STR r2,[r9,#consta-struc] //consta-struc is an expression evaluating
	//to a constant in the range 0-4095.
	STRB r0,[r3,-r8,ASR #2] //stores the least significant byte from
	//r0 to a byte at an address equal to
	//contents(r3) minus contents(r9)/4.
	//r3 and r8 are not altered.
	STR r5,[r7],#-8 //stores a word from r5 to the address
	//in r7, and then decrements r7 by 8.
	LDR r0,localdata //loads a word located at label local data

LDRH: Load register 16-bit halfword value to Memory. The address must be even for halfword transfers

Syntax	LDR{cond}H Rd, [Rn] LDR{cond}H Rd, [Rn, offset] LDR{cond}H Rd, [Rn, offset]! LDR{cond}H Rd, label LDR{cond}H Rd, [Rn], offset
Description	LDR{cond}H Rd, [Rn] (Zero offset) Rn is used as address value. LDR{cond}H Rd, [Rn, offset] (Pre-indexed offset) Rn and offset are added and used as address value. LDR{cond}H Rd, [Rn, offset]! (Pre-indexed offset with update) Rn and offset are added and used as address value. The new address value is written to Rn. LDR{cond}H Rd, label (Program-relative) The assembler calculates the PC offset and generates LDR{cond}H Rd, [R15, offset] LDR{cond}H Rd, [Rn], offset (Post-indexed offset) Rn is used as address value. After memory transfer, the offset is added to Rn.

Example	LDREQSH r11,[r6] //(conditionally) loads r11 with a 16-bit halfword
	//from the address in r6. Sign extends to 32 bits.
	LDRH r1,[r0,#22] //load r1 with a 16 bit halfword from 22 bytes
	//above the address in r0. Zero extend to 32 bits.
	STRH r4,[r0,r1]! //store the least significant halfword from r4
	//to two bytes at an address equal to contents(r0)
	//plus contents(r1). Write address back into r0.
	LDRSB r6,constf //load a byte located at label constf. Sign extend.

LDRD: Load register pair Rd and Rd+1 with double word (64-bit) value. Only the registers R0, R2, R4, R6, R8, R10, R12 are supported as Rd

Syntax	LDR{cond}D Rd, [Rn] LDR{cond}D Rd, [Rn, offset] LDR{cond}D Rd, [Rn, offset]! LDR{cond}D Rd, label LDR{cond}D Rd, [Rn], offset
CPU	ARM9E only
Description	LDR{cond}D Rd, [Rn] (zero offset) Rn is used as address value. LDR{cond}D Rd, [Rn, offset] (pre-index offset) Rn and offset are added and used as address value. LDR{cond}D Rd, [Rn, offset]! (pre-index offset with update) Rn and offset are added and used as address value. The new address value is written to Rn. LDR{cond}D Rd, label (pre-index with PC relative offset) The assembler calculates the PC offset and generates LDR{cond}D Rd, [R15, offset] LDR{cond}D Rd, [Rn], offset (post-index offset with update) Rn is used as address value. After memory transfer, the offset is added to Rn.
Example	LDRD R6,[R11] //Load R6 and R7 and use R11 as address value

MOV: Move constant or register to register. This instruction is also used for shift operations

Syntax	MOV{cond}{S} Rd, Op2
Description	copies the value of Op2 into Rd.
Condition Flags	If S is specified, N, Z flags are updated. C flag may be updated by calculation of Op2.
Example	MOV R5,#0x20 // load R5 with the constant 0x20 MOV R2,R3 // load R2 with the value in R3 MOV R4,R5, SHL #4 // load R4 with the value in R5 shift left by 4 bits

LDRD: Load register pair Rd and Rd+1 with double word (64-bit) value. Only the registers R0, R2, R4, R6, R8, R10, R12 are supported as Rd

Syntax	LDR{cond}D Rd, [Rn] LDR{cond}D Rd, [Rn, offset] LDR{cond}D Rd, [Rn, offset]! LDR{cond}D Rd, label LDR{cond}D Rd, [Rn], offset
CPU	ARM9E only
Description	LDR{cond}D Rd, [Rn] (zero offset) Rn is used as address value. LDR{cond}D Rd, [Rn, offset] (pre-index offset) Rn and offset are added and used as address value. LDR{cond}D Rd, [Rn, offset]! (pre-index offset with update) Rn and offset are added and used as address value. The new address value is written to Rn. LDR{cond}D Rd, label (pre-index with PC relative offset) The assembler calculates the PC offset and generates LDR{cond}D Rd, [R15, offset] LDR{cond}D Rd, [Rn], offset (post-index offset with update) Rn is used as address value. After memory transfer, the offset is added to Rn.
Example	LDRD R6,[R11] //Load R6 and R7 and use R11 as address value

STRD: Store register pair Rd and Rd+1 with double word (64-bit) value. Only the registers R0, R2, R4, R6, R8, R10, R12 are supported as Rd

Syntax	STR{cond}D Rd, [Rn] STR{cond}D Rd, [Rn, offset] STR{cond}D Rd, [Rn, offset]! STR{cond}D Rd, label STR{cond}D Rd, [Rn], offset
CPU	ARM9E only
Description	STR{cond}D Rd, [Rn] (zero offset) Rn is used as address value. STR{cond}D Rd, [Rn, offset] (pre-index offset) Rn and offset are added and used as address value. STR{cond}D Rd, [Rn, offset]! (pre-index offset with update) Rn and offset are added and used as address value. The new address value is written to Rn. STR{cond}D Rd, label (pre-index with PC relative offset) The assembler calculates the PC offset and generates STR{cond}D Rd, [R15, offset] STR{cond}D Rd, [Rn], offset (post-index offset with update) Rn is used as address value. After memory transfer, the offset is added to Rn.
Example	STRD R4,[R9,#24] //Store R4 in R9+24.

LDRD: Load register pair Rd and Rd+1 with double word (64-bit) value. Only the registers R0, R2, R4, R6, R8, R10, R12 are supported as Rd

Syntax	LDR{cond}D Rd, [Rn] LDR{cond}D Rd, [Rn, offset] LDR{cond}D Rd, [Rn, offset]! LDR{cond}D Rd, label LDR{cond}D Rd, [Rn], offset
CPU	ARM9E only
Description	LDR{cond}D Rd, [Rn] (zero offset) Rn is used as address value. LDR{cond}D Rd, [Rn, offset] (pre-index offset) Rn and offset are added and used as address value. LDR{cond}D Rd, [Rn, offset]! (pre-index offset with update) Rn and offset are added and used as address value. The new address value is written to Rn. LDR{cond}D Rd, label (pre-index with PC relative offset) The assembler calculates the PC offset and generates LDR{cond}D Rd, [R15, offset] LDR{cond}D Rd, [Rn], offset (post-index offset with update) Rn is used as address value. After memory transfer, the offset is added to Rn.
Example	LDRD R6,[R11] //Load R6 and R7 and use R11 as address value

STRD: Store register pair Rd and Rd+1 with double word (64-bit) value. Only the registers R0, R2, R4, R6, R8, R10, R12 are supported as Rd

Syntax	STR{cond}D Rd, [Rn] STR{cond}D Rd, [Rn, offset] STR{cond}D Rd, [Rn, offset]! STR{cond}D Rd, label STR{cond}D Rd, [Rn], offset
CPU	ARM9E only
Description	STR{cond}D Rd, [Rn] (zero offset) Rn is used as address value. STR{cond}D Rd, [Rn, offset] (pre-index offset) Rn and offset are added and used as address value. STR{cond}D Rd, [Rn, offset]! (pre-index offset with update) Rn and offset are added and used as address value. The new address value is written to Rn. STR{cond}D Rd, label (pre-index with PC relative offset) The assembler calculates the PC offset and generates STR{cond}D Rd, [R15, offset] STR{cond}D Rd, [Rn], offset (post-index offset with update) Rn is used as address value. After memory transfer, the offset is added to Rn.
Example	STRD R4,[R9,#24] //Store R4 in R9+24.

LDRH: Load register 16-bit halfword value to Memory. The address must be even for halfword transfers

Syntax	LDR{cond}H Rd, [Rn] LDR{cond}H Rd, [Rn, offset] LDR{cond}H Rd, [Rn, offset]! LDR{cond}H Rd, label LDR{cond}H Rd, [Rn], offset
Description	LDR{cond}H Rd, [Rn] (Zero offset) Rn is used as address value. LDR{cond}H Rd, [Rn, offset] (Pre-indexed offset) Rn and offset are added and used as address value. LDR{cond}H Rd, [Rn, offset]! (Pre-indexed offset with update) Rn and offset are added and used as address value. The new address value is written to Rn. LDR{cond}H Rd, label (Program-relative) The assembler calculates the PC offset and generates LDR{cond}H Rd, [R15, offset] LDR{cond}H Rd, [Rn], offset (Post-indexed offset) Rn is used as address value. After memory transfer, the offset is added to Rn.

Example	LDREQSH r11,[r6] //(conditionally) loads r11 with a 16-bit halfword
	//from the address in r6. Sign extends to 32 bits.
	LDRH r1,[r0,#22] //load r1 with a 16 bit halfword from 22 bytes
	//above the address in r0. Zero extend to 32 bits.
	STRH r4,[r0,r1]! //store the least significant halfword from r4
	//to two bytes at an address equal to contents(r0)
	//plus contents(r1). Write address back into r0.
	LDRSB r6,constf //load a byte located at label constf. Sign extend.

LDRSH: Load register signed halfword from Memory. The address must be even for halfword transfers

Syntax	LDR{cond}SH Rd, [Rn] LDR{cond}SH Rd, [Rn, offset] LDR{cond}SH Rd, [Rn, offset]! LDR{cond}SH Rd, label LDR{cond}SH Rd, [Rn], offset
Description	LDR{cond}SH Rd, [Rn] (Zero offset) Rn is used as address value. LDR{cond}SH Rd, [Rn, offset] (Pre-indexed offset) Rn and offset are added and used as address value. LDR{cond}SH Rd, [Rn, offset]! (Pre-indexed offset with update) Rn and offset are added and used as address value. The new address value is written to Rn. LDR{cond}SH Rd, label (Program-relative) The assembler calculates the PC offset and generates LDR{cond}SH Rd, [R15, offset] LDR{cond}SH Rd, [Rn], offset (Post-indexed offset) Rn is used as address value. After memory transfer, the offset is added to Rn.

Example	
	```
	LDREQSH r11,[r6]           //(conditionally) loads r11 with a 16-bit halfword
	                  //from the address in r6. Sign extends to 32 bits.
	LDRH r1,[r0,#22]  //load r1 with a 16 bit halfword from 22 bytes
	                  //above the address in r0. Zero extend to 32 bits.
	STRH r4,[r0,r1]!  //store the least significant halfword from r4
	                  //to two bytes at an address equal to contents(r0)
	                  //plus contents(r1). Write address back into r0.
	LDRSB r6,constf   //load a byte located at label constf. Sign extend.
	```

STRH: Store register 16-bit halfword value to Memory. The address must be even for halfword transfers

Syntax	STR{cond}H Rd, [Rn] STR{cond}H Rd, [Rn, offset] STR{cond}H Rd, [Rn, offset]! STR{cond}H Rd, label STR{cond}H Rd, [Rn], offset
Description	STR{cond}H Rd, [Rn] (zero offset) Rn is used as address value. STR{cond}H Rd, [Rn, offset] (Pre-indexed offset) Rn and offset are added and used as address value. STR{cond}H Rd, [Rn, offset]! (Pre-indexed offset with update) Rn and offset are added and used as address value. The address value is written to Rn. STR{cond}H Rd, label (Program relative) The assembler calculates the PC offset and generates STR{cond}H Rd, [R15], offset. STR{cond}H Rd, [Rn], offset (post-indexed offset) Rn is used as address value. After memory transfer, the offset is added to Rn.

Example	STRH r4,[r0,r1]! //store the least significant halfword from r4
	//to two bytes at an address equal to contents(r0)
	//plus contents(r1). Write address back into r0.

LDRB: Load register byte value to Memory

Syntax	LDR{cond}B Rd, [Rn] LDR{cond}B Rd, [Rn, offset] LDR{cond}B Rd, [Rn, offset]! LDR{cond}B Rd, label LDR{cond}B Rd, [Rn], offset
Description	LDR{cond}B Rd, [Rn] (zero offset) Rn is used as address value. LDR{cond}B Rd, [Rn, offset] (Pre-indexed offset) Rn and offset are added and used as address value. LDR{cond}B Rd, [Rn, offset]! (Pre-indexed offset with update) Rn and offset are added and used as address value. The new address value is written to Rn. LDR{cond}B Rd, label (Program-relative) The assembler calculates the PC offset and generates LDR{cond}B Rd, [R15, offset]. LDR{cond}B Rd, [Rn], offset (Post-indexed offset) Rn is used as address value. After memory transfer, the offset is added to Rn.

Example	LDR r8,[r10] //loads r8 from the address in r10.
	LDRNE r2,[r5,#960]! //(conditionally) loads r2 from a word
	//960 bytes above the address in r5, and
	//increments r5 by 960.
	STR r2,[r9,#consta-struc] //consta-struc is an expression evaluating
	//to a constant in the range 0-4095.
	STRB r0,[r3,-r8,ASR #2] //stores the least significant byte from
	//r0 to a byte at an address equal to
	//contents(r3) minus contents(r9)/4.
	//r3 and r8 are not altered.
	STR r5,[r7],#-8 //stores a word from r5 to the address
	//in r7, and then decrements r7 by 8.
	LDR r0,localdata //loads a word located at label localdata

LDRSB: Load register signed byte value to Memory. Only the registers R0, R2, R4, R6, R8, R10, R12 are supported as Rd

Syntax	LDR{cond}SB Rd, [Rn] LDR{cond}SB Rd, [Rn, offset] LDR{cond}SB Rd, [Rn, offset]! LDR{cond}SB Rd, label LDR{cond}SB Rd, [Rn], offset
Description	LDR{cond}SB Rd, [Rn] (Zero offset) Rn is used as address value. LDR{cond}SB Rd, [Rn, offset] (Pre-indexed offset) Rn and offset are added and used as address value. LDR{cond}SB Rd, [Rn, offset]! (Pre-indexed offset with update) Rn and offset are added and used as address value. The new address value is written to Rn. LDR{cond}SB Rd, label (Program-relative) The assembler calculates the PC offset and generates LDR{cond}SB Rd, [R15, offset] LDR{cond}SB Rd, [Rn], offset (Post-indexed offset) Rn is used as address value. After memory transfer, the offset is added to Rn.

Example	
	```
	LDREQSH r11,[r6]          //(conditionally) loads r11 with a 16-bit halfword
	                 //from the address in r6. Sign extends to 32 bits.
	LDRH r1,[r0,#22] //load r1 with a 16 bit halfword from 22 bytes
	                 //above the address in r0. Zero extend to 32 bits.
	STRH r4,[r0,r1]! //store the least significant halfword from r4
	                 //to two bytes at an address equal to contents(r0)
	                 //plus contents(r1). Write address back into r0.
	LDRSB r6,constf   //load a byte located at label constf. Sign extend.
	```

STRB: Store register byte value to Memory. Only the registers R0, R2, R4, R6, R8, R10, R12 are supported as Rd

Syntax	STR{cond}B Rd, [Rn] STR{cond}B Rd, [Rn, offset] STR{cond}B Rd, [Rn, offset] ! STR{cond}B Rd, label STR{cond}B Rd, [Rn], offset
Description	STR{cond}B Rd, [Rn] (zero offset) Rn is used as address value. STR{cond}B Rd, [Rn, offset] (Pre-indexed offset) Rn and offset are added and used as address value. STR{cond}B Rd, [Rn, offset]! (Pre-indexed offset with update) Rn and offset are added and used as address value. The address value is written to Rn. STR{cond}B Rd, label (Program-relative) The assembler calculates the PC offset and generates STR{cond}B Rd, [R15, offset]. STR{cond}B Rd, [Rn], offset (Post-indexed offset) Rn is used as address value. After memory transfer, the offset is added to Rn.
Example	STRB R8,[R10] // store r8 to memory address in r10. CMP R2,#10 // compare R2 with 10 STRNEB R2,[R5,#960]! // if R2 is not 10, store R2 to R5+960 and update the R5 with this address value

MOV: Move constant or register to register. This instruction is also used for shift operations

Syntax	MOV{cond}{S} Rd, Op2
Description	copies the value of Op2 into Rd.
Condition Flags	If S is specified, N, Z flags are updated. C flag may be updated by calculation of Op2.
Example	MOV R5,#0x20 // load R5 with the constant 0x20 MOV R2,R3 // load R2 with the value in R3 MOV R4,R5, SHL #4 // load R4 with the value in R5 shift left by 4 bits

MVN : Load register with inverted value

Syntax	MVN{cond}{S} Rd, Op2
Description	takes the value of Op2, performs a bitwise logical NOT operation on the value, and places the result into Rd.
Condition Flags	If S is specified, N, Z flags are updated. C flag may be updated by calculation of Op2.
Example	MVNE r11, #0xF000000B

STM: Store multiple registers

Syntax	STM{cond}mode Rn{!}, reglist{^}
Description	Stores any subset of the currently visible registers. This instruction supports all possible stacking modes, maintaining full or empty stacks which can grow up or down memory, and they are very efficient instructions for saving or restoring context, or for moving large blocks of data around main memory.
Condition Flags	If S is specified update flags: N, Z, C, V.
Example	STMDB r1!,{r3-r6,r11,r12} STMFD r13!,{r0,r4-r7,LR} ; Push registers including the stack pointer

STR: Store register 32-bit words to Memory. The address must be 32-bit word-aligned

Syntax	STR{cond} Rd, [Rn] STR{cond} Rd, [Rn, offset] STR{cond} Rd, [Rn, offset]! STR{cond} Rd, label STR{cond} Rd, [Rn], offset
Description	STR{cond} Rd, [Rn] (zero offset) Rn is used as address value. STR{cond} Rd, [Rn, offset] (Pre-indexed offset) Rn and offset are added and used as address value. STR{cond} Rd, [Rn, offset]! (Pre-indexed offset with update) Rn and offset are added and used as address value. The new address value is written to Rn. STR{cond} Rd, label (Program-relative) The assembler calculates the PC offset and generates STR{cond} Rd, [R15], offset. STR{cond} Rd, [Rn], offset (Post-indexed offset) Rn is used as address value. After memory transfer, the offset is added to Rn.

Example	LDR r8,[r10] //loads r8 from the address in r10.
	LDRNE r2,[r5,#960]! //(conditionally) loads r2 from a word
	//960 bytes above the address in r5, and
	//increments r5 by 960.
	STR r2,[r9,#consta-struc] //consta-struc is an expression evaluating
	//to a constant in the range 0-4095.
	STRB r0,[r3,-r8,ASR #2] //stores the least significant byte from
	//r0 to a byte at an address equal to
	//contents(r3) minus contents(r9)/4.
	//r3 and r8 are not altered.
	STR r5,[r7],#-8 //stores a word from r5 to the address
	//in r7, and then decrements r7 by 8.
	LDR r0,localdata //loads a word located at label localdata

SWP: Swap content of 32-bit word between register and memory

Syntax	SWP{cond} Rd, Rm, [Rn]
Description	Swap data between registers and memory.
Example	SWP R2,R3,[R4] // Load R2 with 32-bit word at address in R4 and store R3 to this memory location CMP R0,#55H // Compare R0 with 0x55, if equal SWPEQ R0,R0,[R1] // exchange memory content at address R1 with register R0

Note

- Non word-aligned addresses are handled in exactly the same way as an LDR and an STR instruction.

SWPB: Swap content of a byte between register and memory

Syntax	SWP{cond}B Rd, Rm, [Rn]
Description	Swap data between registers and memory.
Example	SWPB R2,R3,[R4] // Load R2 with unsigned byte at address in R4 and store R3 to this memory location CMP R0,#55H // Compare R0 with 0x55, if equal SWPEQB R0,R0,[R1] // exchange memory content at address R1 with byte in register R0

LDM: Load multiple registers from memory

Syntax	LDM{cond}mode Rn{!}, reglist{^}
Description	Loads any subset of the currently visible registers. This instruction supports all possible stacking modes, maintaining full or empty stacks which can grow up or down memory, and they are very efficient instructions for saving or restoring context, or for moving large blocks of data around main memory.
Condition Flags	If S is specified, N, Z, C and V flags are updated.
Example	LDMFD R13!, {R0,R4-R7,PC} //Pop the registers and return from subroutine

STM: Store multiple registers

Syntax	STM{cond}mode Rn{!}, reglist{^}
Description	Stores any subset of the currently visible registers. This instruction supports all possible stacking modes, maintaining full or empty stacks which can grow up or down memory, and they are very efficient instructions for saving or restoring context, or for moving large blocks of data around main memory.
Condition Flags	If S is specified update flags: N, Z, C, V.
Example	STMDB r1!,{r3-r6,r11,r12} STMFD r13!,{r0,r4-r7,LR} ; Push registers including the stack pointer