

Assembly Language Instruction Set

Data transfer operations

Instruction	Description	Example	Example description
LDR R, M	Value in main memory address M loaded into register R	LDR R1, 100	Load value at memory location 100 into register R1
STR R, M	Value in register R stored in main memory address location M	STR R1, 100	Store value in R1 into main memory location 100
MOV R, #V	Copy data value #V register R	MOV R1, #12	Copy the number 12 into register R1.

The # refers to immediate addressing ie the value is the data.

Arithmetic operations

Instruction	Description	Example	Example description
ADD Ra, Rb, <operand>	Add values in registers in Rb and operand and load result in register Ra	ADD R1, R1, #102 ADD R1, R1, R2	Add 102 to the value in register R1 and store the value in register R1 Add the value stored in R2 and add to the value stored in R1 and output the result.
SUB Ra, Rb, <operand>	Subtract value in perand from register Rb load result in register Ra	SUB R2, R1, #102 SUB R2, R1, R3	Subtract 102 from the value in register R1 and store the result in register R2 subtract the value stored in R3 from the value in R1 and store the result in R2

The <operand> can be a register or a data value. The register is indicated by R and a data value is preceded by a #.

Logical shift operations

Instruction	Description	Example	Example description
LSL Ra, Rb, <operand>	Logical shift left value in register Ra by <operand> value and store in register Ra	LSL R1, R1, #2 LSL R1, R1, R2	Logical shift left value in R1 by 2 and store in register R1 Logical shift left value in R1 by value in R2 and store in register R1
LSR Ra, Rb, <operand>	Logical shift left value in register Ra by <operand> value and store in register Ra	LSR R1, R1, #2 LSR R1, R1, R2	Logical shift right value in R1 by 2 and store in register R1 Logical shift right value in R1 by value in R2 and store in register R1

Eg 3₁₀ << 2₁₀; 3₁₀ = 011₂; 01100₂ = 12₁₀

Logical operations

Instruction	Description	Example	Example description
AND Ra, Rb, <operand>	Bitwise AND operation between value in register Rb and <operand> and store result in Ra	AND R1, R1, #8	Bitwise AND operation between value in R1 and 8 ₁₀ (00001000 ₂) and store result in R1
ORR Ra, Rb, <operand>	Bitwise OR operation between value in register Rb and <operand> and store result in Ra	ORR R1, R1, #8	Bitwise OR operation between value in R1 and 8 ₁₀ (00001000 ₂) and store result in R1
EOR Ra, Rb, <operand>	Bitwise XOR operation between value in register Rb and <operand> and store result in Ra	EOR R1, R1, #8	Bitwise XOR operation between value in R1 and 8 ₁₀ (00001000 ₂) and store result in R1
MVN R, <operand>	Bitwise NOT operation on <operand> and store in R	MVN R1, #8	Bitwise NOT operation on 8 ₁₀ (00001000 ₂) and store result in R1 (11110111 ₂)

Control

Instruction	Description
CMP R, <operand>	Compare value in register R with <operand> value
B <label>	Branch to position <label>
BEQ <label>	Branch to position <label> if result of last comparison between R and <operand> was equal
BNE <label>	Branch to position <label> if result of last comparison was not equal between R and <operand>
BGT <label>	Branch to position <label> if R was greater than <operand> in the last comparison comparison
BLT <label>	Branch to position <label> if R was less than <operand> in the last comparison
HALT	Terminate execution of program

Examples

<i>Selection (if ...)</i> MOV R1, #10 CMP R1, #10 BNE end MOV R2, #20 end: HALT	<i>Selection (if.. else ..)</i> MOV R1, #10 CMP R1, #10 BEQ IF MOV R2, #20 B ELSE IF: MOV R2, #30 ELSE: HALT	<i>Iteration</i> MOV R0, #0 loop: ADD R0 R0 #1 CMP R0 #4 BNE loop HALT
---	---	--