Instruction Set

Instruction	Description	Function
SPECIAL INSTRUCTIONS		
HALT	The processor idles indefinitely	Do nothing indefinitely
NOP	The processor stalls for one clock cycle.	Do nothing for one clock cycle
MOVE INSTRUCTIONS		
MOV rA k	Move literal k to register rA	rA <= k
MOV rA rB	Move the value in register rB to register rA	rA <= rB
MOVE rA k	Move literal k to register rA if the ZERO flag is set.	If ZERO = 1, rA <= k
MOVE rA rB	Move the value in register rB to register rA if the ZERO flag is set.	If ZERO = 1, rA <= rB
MOVNE rA k	Move literal k to register rA if the ZERO flag is not set.	If ZERO = 0, rA <= k
MOVNE rA rB	Move the value in register rB to register rA if the ZERO flag is not set.	If ZERO = 0, rA <= rB
MOVC rA k	Move literal k to register rA if the CARRY flag is set.	If CARRY = 1, rA <= k
MOVC rA rB	Move the value in register rB to register rA if the CARRY flag is set.	If CARRY = 1, rA <= rB

MOVNC rA k	Move literal k to register rA if the CARRY flag is not set.	If CARRY = 0, rA <= k
MOVNC rA rB	Move the value in register rB to register rA if the CARRY flag is not set.	If CARRY = 0, rA <= rB
MOVL rA k	Move literal k to register rA if the SIGN flag is set.	If SIGN = 1, rA <= k
MOVL rA rB	Move the value in register rB to register rA if the SIGN flag is set.	If SIGN = 1, rA <= rB
MOVG rA k	Move literal k to register rA if the SIGN flag is not set.	If SIGN = 0, rA <= k
MOVG rA rB	Move the value in register rB to register rA if the SIGN flag is not set.	If SIGN = 0, rA <= rB
ALU INSTRUCTIONS		
ADD rA k	Add register rA with literal k	rA <= rA + k
ADD rA rB	Add register rA with register rB	rA <= rA + rB
ADDC rA k	Add register with literal k and CARRY	rA <= rA + k + CARRY
ADDC rA rB	Add register rA with register rB and CARRY	rA <= rA + rB + CARRY
AND rA k	And register rA with literal k	rA <= rA AND k
AND rA rB	And register rA with register rB	rA <= rA AND rB

CMP rA k	Compare register rA with literal k. Set CARRY and ZERO flags. Registers are unaffected.	If rA = k, ZERO <= 1 If rA < k, SIGN <= 1 If overflow or underflow, CARRY <= 1
CMP rA rB	Compare register rA with register rB. Set CARRY and ZERO flags. Registers are unaffected.	If rA = rB, ZERO <= 1 If rA < rB, SIGN <= 1 If overflow or underflow, CARRY <= 1
MVN rA k	Invert the bits of k and move the result to register rA	rA <= NOT k
MVN rA rB	Invert the bits of the value stored in register rB and move the result to register rA.	rA <= NOT rB
NOT rA k	Bitwise NOT register rA with literal k.	rA <= rA NOT k
NOT rA rB	Bitwise NOT register rA with register rB.	rA <= rA NOT rB
OR rA k	OR register rA with literal k.	rA <= rA OR k
OR rA rB	OR register rA with register rB.	rA <= rA OR rB
RL rA	Rotate register rA left.	rA <= {rA[6:0], rA[7]} CARRY <= rA[7]
RR rA	Rotate register rA right.	rA <= {rA[0], rA[7:1]} CARRY <= rA[0]
SL0 rA	Shift rA left, zero fill	rA <= {rA[6:0], 0} CARRY <= rA[7]
SL1 rA	Shift rA left, one fill	rA <= {rA[6:0], 1} CARRY <= rA[7]
SR0 rA	Shift rA right, zero fill	rA <= {0, rA[7:1]} CARRY <= rA[0]

SR1 rA	Shift rA right, one fill	rA <= {1, rA[7:1]} CARRY <= rA[0]
SUB rA k	Subtract register rA from literal k	rA <= rA – k
SUB rA rB	Subtract register rA from register rB	rA <= rA – rB
SUBC rA k	Subtract register rA from literal k and CARRY	rA <= rA – k – CARRY
SUBC rA rB	Subtract register rA from register rB and CARRY	rA <= rA – rB – CARRY
TEST rA k	Test bits in register rA against literal k. Update CARRY and ZERO flags. Registers are unaffected.	If (rA AND k) = 0, ZERO <= 1 CARRY <= odd parity of (rA AND k) SIGN <= (rA AND k)[7]
TEST rA rB	Test bits in register rA against register rB. Update CARRY and ZERO. Registers are unaffected.	If (rA AND rB) = 0, ZERO <= 1 CARRY <= odd parity of (rA AND rB) SIGN <= (rA AND rB)[7]
XOR rA k	Bitwise XOR register rA with literal k.	rA <= rA XOR k
XOR rA rB	Bitwise XOR register rA with register rB	rA <= rA XOR rB
BRANCHING INSTRUCTIONS		
JMP k	Jump to the address k	PC <= k
JMP rA	Jump to the address stored in register rA	PC <= rA
JE k	Jump to the address k if the ZERO flag is set.	If ZERO = 1, PC <= k

JE rA	Jump to the address stored in register rA if the ZERO flag is set.	If ZERO = 1, PC <= rA
JNE k	Jump to the address k if the ZERO flag is not set.	If ZERO = 0, PC <= k
JNE rA	Jump to the address stored in register rA if the ZERO flag is not set.	If ZERO = 0, PC <= rA
JL k	Jump to the address k if the SIGN flag is set.	If SIGN = 1, PC <= k
JL rA	Jump to the address stored in register rA if the SIGN flag is set.	If SIGN = 1, PC <= rA
JG k	Jump to the address k if the SIGN flag is not set.	If SIGN = 0, PC <= k
JG rA	Jump to the address stored in register rA if the SIGN flag is not set.	If SIGN = 0, PC <= rA
MEMORY INSTRUCTIONS		
FETCH rA k	Fetch the value stored at address k in RAM and store it in register rA	rA <= RAM[k]
FETCH rA rB	Fetch the value from the address stored in register rB from RAM and store it in register rA.	rA <= RAM[rB]
STORE rA k	Store the value in register rA at address k in RAM.	RAM[k] <= rA
STORE rA rB	Store the value in register rA at the address stored in register rB in RAM.	RAM[rB] <= rA

CALL & RETURN		
CALL <label>	Jump to the subroutine at address denoted by the text literal <label> and store the return address on the stack.	STACK[SP] <= PC + 1; PC <= ADDR[<label>]
RETURN	Jump to the address pointed to by the stack pointer	PC <= STACK[SP]

Notes:

- rA and rB are 8-bit registers
- k is an 8-bit literal
- ALU instructions update the status flags by the following, unless indicated otherwise:
    - ZERO = 1 if ALU result is 0, and ZERO = 1 otherwise
    - CARRY = 1 if overflow or underflow occurs, and CARRY = 0 otherwise
    - SIGN = MSB of result