Instruction Set Architecture (ISA) specifications

Acknowledgement: this ISA is taken from the Architecture Course Project at Cairo University Faculty of Engineering.

A) Registers

R[0:7]<15:0> ; Eight 16-bit general purpose register
PC<15:0> ; 16-bit program counter
SP<15:0>; 16-bit stack pointer
CCR<2:0> ; condition code register
Z<0>:=CCR<0> ; zero flag, change after arithmetic, logical, or shift operations
N<0>:=CCR<1> ; negative flag, change after arithmetic, logical, or shift operations
C<0>:=CCR<2> ; carry flag, change after arithmetic or shift operations.

B) Input-Output

IN.PORT<15:0> ; 16-bit data input port
OUT.PORT<15:0> ; 16-bit data output port
INTR.IN<0> ; a single, non-maskable interrupt
RESET.IN<0> ; reset signal

C) Mnemonics convention

Rsrc1 ; 1st operand register
Rsrc2 ; 2nd operand register
Rdst ; result register field
Imm ; Immediate Value 16-bits unless stated otherwise.

Take Care that Some instructions will Occupy more than one memory location

The following table contains the instructions of the ISA and their function.

Mnemonic	**Function**
NOP	PC ← PC + 1
SETC	C ←1, make sure other flags don't get affected
CLRC	C ←0, make sure other flags don't get affected

NOT Rdst, Rsrc1	NOT value stored in register Rsrc and store it in Rdst
	R[Rdst] ← 1's Complement(R[Rsrc1]);
	If (1's Complement(R[Rsrc1]) = 0): Z ←1; else: Z ←0;
	If (1's Complement(R[Rsrc1]) < 0): N ←1; else: N ←0
	Don't change carry flag

INC Rdst, Rsrc1	Increment value stored in Rsrc1
	R[Rdst] ←R[Rsrc1] + 1;
	If ((R[Rsrc1] + 1) = 0): Z ←1; else: Z ←0;
	If ((R[Rsrc1] + 1) < 0): N ←1; else: N ←0
	Update carry flag as appropriate

DEC Rdst,Rsrc1	Decrement value stored in Rsrc1
	R[Rdst] ←R[Rsrc1] - 1;
	If ((R[Rsrc1] - 1) = 0): Z ←1; else: Z ←0;
	If ((R[Rsrc1] - 1) < 0): N ←1; else: N ←0
	Update carry flag as appropriate

OUT Rsrc1	OUT.PORT ← R[Rsrc1]
IN Rdst	R[Rdst] ←IN.PORT
MOV Rdst, Rsrc1	Move value from register Rsrc1 to register Rdst **DON'T change flags**
ADD Rdst, Rsrc1,Rsrc2	Add the values stored in registers Rsrc1, Rsrc2 and store the result in Rdst and updates carry If the result =0 then Z ←1; else: Z ←0; If the result <0 then N ←1; else: N ←0 **Update carry flag as appropriate**
IADD Rdst, Rsrc1, Imm	Add the values stored in registers Rsrc1 to Immediate Value and store the result in Rdst and updates carry If the result =0 then Z ←1; else: Z ←0; If the result <0 then N ←1; else: N ←0 **Update carry flag as appropriate**
SUB Rdst, Rsrc1,Rsrc2	Subtract the values stored in registers **Rsrc1-Rsrc2** and store the result in Rdst and updates carry If the result =0 then Z ←1; else: Z ←0; If the result <0 then N ←1; else: N ←0 **Update carry flag as appropriate**
AND Rdst, Rsrc1,Rsrc2	AND the values stored in registers Rsrc1, Rsrc2 and store the result in Rdst If the result =0 then Z ←1; else: Z ←0; If the result <0 then N ←1; else: N ←0 **Don't change carry flag**
OR Rdst, Rsrc1,Rsrc2	OR the values stored in registers Rsrc1, Rsrc2 and store the result in Rdst

	If the result =0 then Z ←1; else: Z ←0; If the result <0 then N ←1; else: N ←0 **Don't change carry flag**
PUSH Rsrc1	DataMemory[SP] ← R[Rsrc]; SP-=1
POP Rdst	SP+=1; R[Rdst] ← DataMemory[SP];
LDM Rdst, Imm	Load immediate value (16 bit) to register Rdst R[Rdst] ← Imm<15:0>
LDD Rdst, Rsrc1	Load value from memory address Rsrc1 R[Rdst] ← DataMemory [R[Rsrc1]];
STD Rsrc2, Rsrc1	Store value that is in register Rsrc1 to memory location Rsrc2 DataMemory [R[Rsrc2]] ←R[Rsrc1];
JZ Rdst	Jump if zero If (Z=1): PC ←R[Rdst]; (Z=0)
JC Rdst	Jump if negative If (C=1): PC ←R[Rdst]; (C=0)
JMP Rdst	Jump PC ←R[Rdst]
CALL Rdst	(DataMemory[SP] ← PC + 1; sp-=1; PC ← R[Rdst])
RET	sp+=1, PC ← DataMemory [SP]
RTI	sp+=1; PC ← DataMemory [SP]; **Then restore Flags**