Green Sheet ISA - Vectorial Pipeline CPU

System

	Id			Null
	Type	Op	Null	
	31:30	29:28	27:25	24:0
NOP	00	00	000	00000000000000000000000000
COM	00	01	000	00000000000000000000000000
END	00	10	000	00000000000000000000000000

Data register-register / vectorial-vectorial

	Id			Registers			Null
	Type	Op	IS	RD	RA	RB	
	31:30	29:27	26:25	24:21	20:17	16:13	12:0
ADDV	01	000	00	xxxx	xxxx	xxxx	00000000000000
MOVV	01	001	00	xxxx	xxxx	xxxx	00000000000000
XORV	01	010	00	xxxx	xxxx	xxxx	00000000000000
ORV	01	011	00	xxxx	xxxx	xxxx	00000000000000
SHRV	01	100	00	xxxx	xxxx	xxxx	00000000000000
SHLV	01	101	00	xxxx	xxxx	xxxx	00000000000000
CMPV	01	110	00	xxxx	xxxx	xxxx	00000000000000

Data register-register / vectorial-scalar

	Id			Registers			Null
	Type	Op	IS	RD	RA	RB	
	31:30	29:27	26:25	24:21	20:17	16:13	12:0
ADDS	01	000	01	xxxx	xxxx	xxxx	00000000000000
MOVS	01	001	01	xxxx	xxxx	xxxx	00000000000000
XORS	01	010	01	xxxx	xxxx	xxxx	00000000000000
ORS	01	011	01	xxxx	xxxx	xxxx	00000000000000
SHRS	01	100	01	xxxx	xxxx	xxxx	00000000000000
SHLS	01	101	01	xxxx	xxxx	xxxx	00000000000000
CMPS	01	110	01	xxxx	xxxx	xxxx	00000000000000

Data register-immediate / vectorial-scalar

	Id			Registers			Null
	Tipo	Op	IS	RD	RA	Immediate	
	31:30	29:27	26:25	24:21	20:17	16:9	8:0
ADDS	01	000	11	xxxx	xxxx	xxxxxxxx	000000000
MOVS	01	001	11	xxxx	xxxx	xxxxxxxx	000000000
XORS	01	010	11	xxxx	xxxx	xxxxxxxx	000000000
ORS	01	011	11	xxxx	xxxx	xxxxxxxx	000000000
SHRS	01	100	11	xxxx	xxxx	xxxxxxxx	000000000
SHLS	01	101	11	xxxx	xxxx	xxxxxxxx	000000000
CMPS	01	110	11	xxxx	xxxx	xxxxxxxx	000000000

Memoria

	Id				Addressing			Null
	Type	Op	Null	IS	RD	RA	Immediate	
	31:30	29	28:27	26:25	24:21	20:17	16:9	8:0
LDRV	10	0	00	00	xxxx	xxxx	xxxxxxxx	000000000
LDRS	10	0	00	01	xxxx	xxxx	xxxxxxxx	000000000
STRV	10	1	00	00	xxxx	xxxx	xxxxxxxx	000000000
STRS	10	1	00	01	xxxx	xxxx	xxxxxxxx	000000000

Control

	Id			Null	Instr number	Null
	Type	Op	Null		Immediate	
	31:30	29:28	27:25	24:17	16:9	8:0
JMP	11	00	000	00000000	xxxxxxxx	000000000
JEQ	11	01	000	00000000	xxxxxxxx	000000000
JLT	11	10	000	00000000	xxxxxxxx	000000000

INSTRUCCIÓN	MODO	EJEMPLO	SINTAXIS	COMPORTAMIENTO	OP CODE	TIPO
NOP	no aplica	NOP	NOP	PC ← PC + 4; Flags ← 0	0	system
COM	no aplica	COM	COM	COMFlag ← 1	1	
END	no aplica	END	END	ENDFlag ← 1	2	
ADDV	registro	ADDV RV[1:8], RV[1:8], RV[1:8]	ADDV Rd, Ra, Rb	Rd ← Ra + Rb	0	dato
ADDS	registro	ADDS RV[1:8], RV[1:8], RS[1:6]	ADDS Rd, Ra, Rb	Rd ← Ra + Rb		
ADDS	inmediato	ADDS RV[1:8], RV[1:8], #5	ADDS Rd, Ra, Imm	Rd ← Ra + Imm		
MOVV	registro	MOVV RV[1:8], RV[1:8]	MOVV Rd, Rb	Rd ← Rb	1	
MOVS	registro	MOVS RS[1:6], RS[1:6]	MOVS Rd, Rb	Rd ← Rb		
MOVS	inmediato	MOVS RS[1:6], #5	MOVS Rd, Rb	Rd ← Imm		
XORV	registro	XORV RV[1:8], RV[1:8], RV[1:8]	XORV Rd, Ra, Rb	Rd ← Ra ^ Rb	2	
XORS	registro	XORS RV[1:8], RV[1:8], RS[1:6]	XORS Rd, Ra, Rb	Rd ← Ra ^ Rb		
XORS	inmediato	XORS RV[1:8], RV[1:8], #5	XORS Rd, Ra, Imm	Rd ← Ra ^ Imm		
ORV	registro	ORV RV[1:8], RV[1:8], RV[1:8]	ORV Rd, Ra, Rb	Rd ← Ra | Rb	3	
ORS	registro	ORS RV[1:8], RV[1:8], RS[1:6]	ORS Rd, Ra, Rb	Rd ← Ra | Rb		
ORS	inmediato	ORS RV[1:8], RV[1:8], #5	ORS Rd, Ra, Imm	Rd ← Ra | Imm		
SHRV	registro	SHRV RV[1:8], RV[1:8], RV[1:8]	SHRV Rd, Ra, Rb	Rd ← Ra >> Rb	4	
SHRS	registro	SHRS RV[1:8], RV[1:8], RS[1:6]	SHRS Rd, Ra, Rb	Rd ← Ra >> Rb		
SHRS	inmediato	SHRS RV[1:8], RV[1:8], #5	SHRS Rd, Ra, Imm	Rd ← Ra >> Imm		
SHLV	registro	SHLV RV[1:8], RV[1:8], RV[1:8]	SHLV Rd, Ra, Rb	Rd ← Ra << Rb	5	
SHLS	registro	SHLS RV[1:8], RV[1:8], RS[1:6]	SHLS Rd, Ra, Rb	Rd ← Ra << Rb		
SHLS	inmediato	SHLS RV[1:8], RV[1:8], #5	SHLS Rd, Ra, Imm	Rd ← Ra << Imm		
CMPV	registro	SHRV RV[1:8], RV[1:8]	SHRV Ra, Rb	Ra – Rb; ALUFLags ← NZ	6	
CMPS	registro	SHRS RV[1:8], RS[1:6]	SHRS Ra, Rb	Ra – Rb; ALUFLags ← NZ		
CMPS	inmediato	SHRS RV[1:8], #5	SHRS Ra, Imm	Ra – Imm; ALUFLags ← NZ		
LDRV	registro	LDRV RV[1:8], [RV[1:8]]	LDR Rd, [Ra]	Rd ← MEM[Ra + 0]	0	memoria
LDRV	inmediato	LDRV RV[1:8], [RV[1:8], #8]	LDR Rd, [Ra, Imm]	Rd ← MEM[Ra + Imm]		
LDRS	registro	LDRS RS[1:6], [RV[1:8]]	LDR Rd, [Ra]	Rd ← MEM[Ra + 0]		
LDRS	inmediato	LDRS RS[1:6], [RV[1:8], #8]	LDR Rd, [Ra, Imm]	Rd ← MEM[Ra + Imm]		
STRV	registro	STRV RV[1:8], [RV[1:8]]	STR Rd, [Ra]	MEM[Ra + 0] ← Rd	1	
STRV	inmediato	STRV RV[1:8], [RV[1:8], #8]	STR Rd, [Ra, Imm]	MEM[Ra + Imm] ← Rd		
JMP	incondicional	JMP _loop	JMP LABEL	PC ← Imm	0	control
JEQ	condicional	JEQ _loop	JEQ LABEL	PC ← Imm if Z=1 else PC+4	1	
JLT	condicional	JLT _loop	JLT LABEL	PC ← Imm if N=1 else PC+4	2	

INSTRUCCIÓN	NOMBRE	DESCRIPCIÓN/PROPÓSITO
NOP	*No Operation*	Esta instrucción no hace nada durante la ejecución. Se utiliza con fines de temporización, para prevenir riesgos y para invalidar una instrucción existente, como un salto, como objetivo de una instrucción de ejecución.
COM	*Communication*	Esta instrucción activa la bandera de COM. Se utiliza como indicador para iniciar la comunicación con el arduino.
END	*End*	Esta instrucción activa la bandera de END. Se utiliza como indicador para indicar la finalización del algoritmo.
ADD	*Add*	Esta instrucción realiza la suma de dos valores y los almacena en un registro destino.
MOV	*Move*	Esta instrucción almacena el valor de un registro o inmediato recibido en un registro destino.
XOR	*Xor*	Esta instrucción realiza la operación lógica xor de dos valores y los almacena en un registro destino.
OR	*Or*	Esta instrucción realiza la operación lógica or de dos valores y los almacena en un registro destino.
SHR	*Shift right*	Esta instrucción realiza la operación de desplazamiento a la derecha en un valor y lo almacena en un registro destino.
SHL	*Shift left*	Esta instrucción realiza la operación de desplazamiento a la izquierda en un valor y lo almacena en un registro destino.
CMP	*Compare*	Esta instrucción realiza la resta de dos valores y actualiza las banderas de la ALU acorde al resultado que obtiene.
LDR	*Load*	Esta instrucción guarda en el primer registro de la instrucción el valor ubicado en la dirección de memoria efectiva conformada por el valor del segundo registro adicionado a un indice que puede ser indicado con un inmediato u otro registro. Si no se indica un indice, se asume un cero y la dirección efectiva es igual al segundo registro.
STR	*Store*	Esta instrucción guarda el valor del primer registro de la instrucción en la dirección de memoria efectiva conformada por el valor del segundo registro adicionado a un indice que puede ser indicado con un inmediato u otro registro. Si no se indica un indice, se asume un cero y la dirección efectiva es igual al segundo registro.
JMP	*Jump*	Esta instrucción realiza un salto a la primera instrucción después de la etiqueta que se indica.
JEQ	*Jump equal*	Esta instrucción comprueba si la bandera Z (cero) se ha accionado, si es así realiza un salto a la primera instrucción después de la etiqueta correspondiente, sino continua con la siguiente instrucción. **NOTA:** La instrucción CMP debe ser utilizada antes de emplear esta instrucción, en caso contrario podría tener un funcionamiento incierto.
JLT	*Jump less than*	Esta instrucción comprueba si la bandera N (negativo) se ha accionado, si es así realiza un salto a la primera instrucción después de la etiqueta correspondiente, sino continua con la siguiente instrucción. **NOTA:** La instrucción CMP debe ser utilizada antes de emplear esta instrucción, en caso contrario podría tener un funcionamiento incierto.

REGISTROS	DESCRIPCIÓN/PROPÓSITO
R0 - R9	Se emplean diez registros de propósito general.