SISA:

Sajiv's Instruction Set Architecture

Immediate Flag

An immediate instruction holds data to be worked on, or stored, including a destination address (for immediate arithmetic instructions Rd = Rs), immediate flag is set high.

A non-immediate instruction holds only addresses of data to be worked on or moved to EEPROM/RAM, immediate flag is set low.

RAM Flag

Ram flag is set high when a RAM address is present in instruction, otherwise it is set low.

EEPROM Flag

EEPROM flag is set high when an EEPROM address is present in instruction, otherwise it is set low.

Loop Flags

There are 7 separate MARs, addressed from 001-111, when loop flags are set to 000, the program memory address will not be stored.

Instruction formats are written in pure binary.

Mnemonics:

#16: A value with maximum length of 16 bits (0 to 65535)

Rs: Source Register

Rd: Destination Register

RAMs: Source Ram Address

RAMd: Destination Ram Address

ROMs: Source ROM Address

ROMd: Destination ROM Address

Basic Instruction Structure

11 Bits					20 Bits	=31 Bits
00000-11111	0-1	0-1	0-1	000-111	20-Bit Operand	
5-Bit Opcode	IMM Flag	Ram Flag	EEPROM Flag	Loop Flags		

ADDIMM

Adds an immediate value to the contents of a register, stores the result in the same register.

Assembly Language:

ADDIMM Rd, #16

Operation:

Rd = Rs

Rd = Rs + #16

Example:

ADDIMM R16, 4004

00000	1	0	0	000-111	#16	0000-1111
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**Immediate Val**	**Rd**

ADD

Adds two registers together and stores the result into a register.

Assembly Language:

ADD Rd, R1, R2

Operation:

Rd = R1 + R2

Example:

ADD R1, R2, R3

00000	**0**	**0**	**0**	**000-111**	**0000-1111**	**0000-1111**	**8 Bits**	**0000-1111**
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**R1**	**R2**	**Don’t Cares**	**Rd**

SUBIMM

Subtracts an immediate value from a register, stores the result in that same register.

Assembly Language:

SUBIMM Rd, #16

Operation:

Rd = Rs

Rd = Rs - #16

Example:

SUBIMM Rd, 9000

00001	1	0	0	000-111	#16	0000-1111
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**Immediate Val**	**Rd**

SUB

Subtracts two registers and stores the result into a register.

Assembly Language:

SUB Rd, R1, R2

Operation:

Rd = R1 - R2

00001	0	0	0	1	1	X	0000-1111	0000-1111	8	0000-1111
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Write**	**Read**	**Loop**	**R1**	**R2**	**Don’t Cares**	**Rd**

MULIMM

Multiplies an immediate value and a register, stores the result in that same register.

Assembly Language:

MULIMM Rd, #16

Operation:

Rd = Rs

Rd = Rs * #16

00010	1	0	0	1	1	X	#16	0000-1111
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Write**	**Read**	**Loop**	**Immediate Val**	**Rd**

MUL

Multiplies two registers together and stores the result into a register.

Assembly Language:

MUL Rd, R1, R2

Operation:

Rd = R1 * R2

00010	0	0	0	1	1	X	0000-1111	0000-1111	8	0000-1111
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Write**	**Read**	**Loop**	**R1**	**R2**	**Don’t Cares**	**Rd**

DIVIMM

Divides a register by an immediate value, stores the result in that same register.

Assembly Language:

DIVIMM #16, Rd

Operation:

Rd = Rs

Rd = Rs / #16

00011	1	0	0	1	1	X	#16	0000-1111
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Write**	**Read**	**Loop**	**Immediate Val**	**Rd**

DIV

Divides R1 by R2 and stores the result into a register.

Assembly Language:

DIV Rd, R1, R2

Rd = R1 / R2

00011	0	0	0	1	1	X	0000-1111	0000-1111	8	0000-1111
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Write**	**Read**	**Loop**	**R1**	**R2**	**Don’t Cares**	**Rd**

INC

Adds 1 to a register.

Assembly Language:

INC Rd

Operation:

Rd = Rd + 1

Example:

INC R4

00100	0	0	0	000-111	0000-1111	0000-1111	8 Bits	0000-1111
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**Don’t Cares**	**Rd**	**Don’t Cares**	**Rd**

DEC

Subtracts 1 from a register.

Assembly Language:

DEC Rd

Operation:

Rd = Rd – 1

Example:

DEC R4

00101	**0**	**0**	**0**	**000-111**	**0000-1111**	**0000-1111**	**8 Bits**	**0000-1111**
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**Don’t Cares**	**Rd**	**Don’t Cares**	**Rd**

CLR

Sets the contents of a register to 0.

Assembly Language:

CLR Rd

00110	**0**	**0**	**0**	**000-111**	**0000-1111**	**0000-1111**	**8 Bits**	**0000-1111**
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**Don’t Cares**	**R2**	**Don’t Cares**	**Rd**

CMP

Compares two registers and sets status flags accordingly.

Assembly Language:

CMP R1, R2

00111	0	0	0	000-111	0000-1111	0000-1111	8 Bits	4 Bits
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**R1**	**R2**	**Don’t Cares**	**Don’t Cares**

LSL

Logically shifts a register to the left by one bit.

Assembly Language:

LSL Rd

01000	0	0	0	000-111	4 Bits	0000-1111	8 Bits	0000-1111
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**Don't Cares**	**R2**	**Don't Cares**	**Rd**

LSR

Logically shifts a register to the right by one bit.

Assembly Language:

LSR Rd

01001	0	0	0	000-111	4 Bits	0000-1111	8 Bits	0000-1111
Opcode	Imm Flag	Ram Flag	EEPROM Flag	Loop Bits	Don’t Cares	R2	Don’t Cares	Rd

LOGAND

Ands two registers together logically and sets status bits accordingly.

Assembly Language:

LOGAND R1, R2

01010	**0**	**0**	**0**	**000-111**	**0000-1111**	**0000-1111**	**8 Bits**	**4 Bits**
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**R1**	**R2**	**Don’t Cares**	**Don’t Cares**

LOGOR

ORs two registers together logically and sets status bits accordingly.

Assembly Language:

LOGOR R1, R2

01011	**0**	**0**	**0**	**000-111**	**0000-1111**	**0000-1111**	**8 Bits**	**4 Bits**
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**R1**	**R2**	**Don’t Cares**	**Don’t Cares**

LOGXOR

XORs two registers together logically and sets status bits accordingly.

Assembly Language:

LOGXOR R1, R2

01100	**0**	**0**	**0**	**000-111**	**0000-1111**	**0000-1111**	**8 Bits**	**4 Bits**
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**R1**	**R2**	**Don’t Cares**	**Don’t Cares**

BITAND

Performs an AND operation on each bit of two registers and stores the result into a register.

Assembly Language:

BITAND Rd, R1, R2

01101	**0**	**0**	**0**	**000-111**	**0000-1111**	**0000-1111**	**8 Bits**	**0000-1111**
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**R1**	**R2**	**Don’t Cares**	**Rd**

BITOR

Performs an OR operation on each bit of two registers and stores the result into a register.

Assembly Language:

BITOR Rd, R1, R2

01110	0	0	0	000-111	0000-1111	0000-1111	8 Bits	0000-1111
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**R1**	**R2**	**Don’t Cares**	**Rd**

BITXOR

Performs an XOR operation on each bit of two registers and stores the result into a register.

Assembly Language:

BITXOR Rd, R1, R2

01111	0	0	0	000-111	0000-1111	0000-1111	8 Bits	0000-1111
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**R1**	**R2**	**Don’t Cares**	**Rd**

ONE’S COMPLEMENT

Converts the contents of a register to its one’s complement representation.

Assembly Language:

ONES Rd

10000	0	0	0	000-111	4 Bits	0000-1111	8 Bits	0000-1111
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**Don’t Cares**	**R2**	**Don’t Cares**	**Rd**

TWO’S COMPLEMENT

Converts the contents of a register to its two’s complement representation.

Assembly Language:

TWOS Rd

10001	0	0	0	000-111	4 Bits	0000-1111	8 Bits	0000-1111
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**Don’t Cares**	**R2**	**Don’t Cares**	**Rd**

LOAD

Stores a 16-bit value into a register.

Assembly Language:

LOAD Rd, #16

10010	**1**	**0**	**0**	**000-111**	**#16**	**0000-1111**
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**Immediate Val**	**Rd**

MOVE

Moves the contents of a register to RAM.

Assembly Language:

MOV RAMd, Rs

10011	**0**	**1**	**0**	**000-111**	**4 bits**	**0000-1111**	**00000000-11111111**	**4 bits**
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**Don’t Cares**	**Rs**	**RAMd**	**Don’t Cares**

CPY

Copies the contents from one register to another register.

Assembly Language:

CPY Rd, Rs

10100	0	0	0	000-111	4 bits	0000-1111	8 Bits	0000-1111
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**Don’t Cares**	**Rs**	**Don’t Cares**	**Rd**

STORE REGISTER

Stores the contents of a register to EEPROM.

Assembly Language:

STR Rs, ROMd

10101	0	0	1	000-111	4 bits	0000-1111	00000000-11111111	4 bits
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**Don't Cares**	**Rs**	**ROMd**	**Don't Cares**

STORE RAM

Stores the contents of a RAM address to EEPROM.

Assembly Language:

STRR RAMs, ROMd

10101	0	1	1	000-111	00000000-11111111	00000000-11111111	4 bits
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**RAMs**	**ROMd**	**Don’t Cares**

INPUT

Stores the contents of the input register into a GPR.

Assembly Language

INPUT Rd

10111	**0**	**0**	**0**	**000-111**	**4 bits**	**4 bits**	**8 Bits bits**	**0000-1111**
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**Don’t Cares**	**Don’t Cares**	**Don’t Cares**	**Rd**

INPUT RAM

Stores the contents of a RAM address into a GPR.

Assembly Language:

INPUTRAM Rd, RAMs

10111	**0**	**1**	**0**	**000-111**	**4 bits**	**4 bits**	**00000000-11111111**	**0000-1111**
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**Don’t Cares**	**Don’t Cares**	**RAMs**	**Rd**

INPUT ROM

Stores the contents of a ROM address into a GPR.

Assembly Language:

INPUTROM Rd, ROMs

10111	0	0	1	000-111	4 bits	4 bits	00000000-11111111	0000-1111
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**Don’t Cares**	**Don’t Cares**	**ROMs**	**Rd**

OUTPUT

Stores a value from a GPR in the output register.

Assembly Language:

OUTPUT Rs

11000	0	0	0	000-111	4 bits	0000-1111	8 Bits	4 bits
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**Don’t Cares**	**Rs**	**Don’t Cares**	**Don’t Cares**

CLR STATUS

Clears the contents of the status register.

Assembly Language:

CLRSTS

11001	**0**	**0**	**0**	**000-111**	**4 bits**	**4 bits**	**8 Bits**	**4 bits**
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**Don’t Cares**	**Don’t Cares**	**Don’t Cares**	**Don’t Cares**

JUMP

Loads a previously saved memory address (from 1 of the 7 memory address registers) into the program counter.

Assembly Language:

JUMP

Example:

JUMP 2

11010	**0**	**0**	**0**	**000-111**	**4 bits**	**4 bits**	**8 Bits**	**0000-0110**
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**Don’t Cares**	**Don’t Cares**	**Don’t Cares**	**Loop Bits**

BRANCH IF EQUAL

Compares two registers and loads a previously saved memory address (from 1 of the 7 memory address registers) into the program counter if the two registers are equal.

Assembly Language:

BRNEQ

Example:

BREQ 3, R1, R2

11011	**0**	**0**	**0**	**000-111**	**0000-1111**	**0000-1111**	**8 Bits**	**0000-0110**
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**R1**	**R2**	**Don’t Cares**	**Loop Bits**

BRANCH IF NOT EQUAL

Compares two registers and loads a previously saved memory address (from 1 of the 7 memory address registers) into the program counter if the two registers are not equal.

Assembly Language:

BRNNEQ

Example:

BRNNEQ 5, R1, R2

11100	**0**	**0**	**0**	**000-111**	**0000-1111**	**0000-1111**	**8 Bits**	**0000-0110**
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**R1**	**R2**	**Don't Cares**	**Loop Bits**

BRANCH IF GREATER

Compares two registers and loads a previously saved memory address (from 1 of the 7 memory address registers) into the program counter if R1 is greater than R2.

Assembly Language:

BRNG

Example:

BRNG 5, R1, R2

11101	0	0	0	000-111	0000-1111	0000-1111	8 Bits	0000-0110
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**R1**	**R2**	**Don’t Cares**	**Loop Bits**

BRANCH IF LESS

Compares two registers and loads a previously saved memory address (from 1 of the 7 memory address registers) into the program counter if R1 is less than R2.

Assembly Language:

BRNL

Example:

BRNL 7, R13, R2

11110	**0**	**0**	**0**	**000-111**	**0000-1111**	**0000-1111**	**8 Bits**	**0000-0110**
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**R1**	**R2**	**Don’t Cares**	**Loop Bits**

NO OPERATION

The processor does nothing.

Assembly Language:

NOP

11110	**0**	**0**	**0**	**000-111**	**4 bits**	**4 bits**	**8 Bits**	**4 bits**
Opcode	**Imm Flag**	**Ram Flag**	**EEPROM Flag**	**Loop Bits**	**Don’t Cares**	**Don’t Cares**	**Don’t Cares**	**Don’t Cares**