Processor Instruction Set

Op Codes

The four most significant bits of the instruction code represent the op-code for the instruction. These op-codes are defined as follows:

Op-code	Mnemonic	Instruction Format	Details
0000	NOP	NOP	
0001	ARITH_2OP	FUNC Rs1, Rs2, Rd	Valid FUNC values: ADD, ADDC, SUB, SUBB, AND, OR, XOR, XNOR
0010	ARITH_1OP	FUNC Rs1, Rd	Valid FUNC values: SHIFTL, SHIFTR, NOT, CP
0011	MOVI	FUNC Rd, #8-bit	Valid FUNC values: MOVIL, MOVIH
0100	ADDI	ADDI Rs1, Rd, #6-bit	
0101	SUBI	SUBI Rs1, Rd, #6-bit	
0110	LD	LD Rs1, Rd, #6-bit	
0111	ST	ST Rs1, Rd, #6-bit	
1000	BEQ	BEQ Rs1, Rs2, #6-bit	
1001	BGE	BGE Rs1, Rs2, #6-bit	
1010	BLE	BLE Rs1, Rs2, #6-bit	
1011	BC	BC #6-bit	
1100	J	J #12-bit	
1101	JL	JL #12-bit	
1110	INT	FUNC #4-bit	Valid FUNC values: INTE, INTD, INT_TRIG
1111	CONTROL	FUNC	Valid FUNC values: RET, STC, STB, HALT, RESET

In the instruction formats above and the Op-code detail below, the following conventions are observed:

- Register designations (Rs1, Rs2, Rd) represent Register File identifiers. These are always 3-bit indices
    - In the assembler, register ids are written with a dollar sign, e.g. "ADD $0 $1 $2"

- #N-bit – an N bit immediate value. May be signed (sign-extended) or unsigned (literal).
    - Example: Jump uses a sign-extended immediate, so J 0xFF8 would jump backwards, as it would be interpreted as "Jump -8". Note that the PC is still incremented, so "J 0x0" would proceed to the next instruction, but "J 0xFFE" would be an infinite loop.
    - On that note, "J 0xFFF" is not valid. The LSB of relative jump or branch commands should always be zero, as instructions are aligned to even addresses.
    - In the assembler, immediate values can be in any number of formats (anything that Java's Integer.decode will parse). Examples include:
        - ADDI $0 $0 1 // Increments Reg0 by 1
        - MOVIH $0 0x80 // MOVE 0x80 into the upper byte of Reg 0

- Instruction formats are consistent within a single op-code, but may vary between op-codes. The destination register location is consistent (all op-codes that store data in the register file do so with the three bits immediately following the op-code), but the source registers may be located in different bit fields. You will implement this in the instruction decode block.

Op-code Details

NOP

OP-CODE	Reserved
0000	000000000000

No operation.

ADD

OP-CODE	RD	RS1	RS2	FUNC
0001	Rd	Rs1	Rs2	000

Adds Rs1 and Rs2 and stores the sum in Rd.

ADDC

OP-CODE	RD	RS1	RS2	FUNC
0001	Rd	Rs1	Rs2	001

Adds Rs1 and Rs2 along with the Carry flag value and stores the sum in Rd.

SUB

OP-CODE	RD	RS1	RS2	FUNC
0001	Rd	Rs1	Rs2	010

Subtracts Rs2 from Rs1 and stores the difference in Rd.

SUBB

OP-CODE	RD	RS1	RS2	FUNC
0001	Rd	Rs1	Rs2	011

Subtracts Rs2 from Rs1 along with the Borrow flag value and stores the difference in Rd.

AND

OP-CODE	RD	RS1	RS2	FUNC
0001	Rd	Rs1	Rs2	100

Performs bitwise AND of Rs1 and Rs2 and stores the result in Rd.

OR

OP-CODE	RD	RS1	RS2	FUNC
0001	Rd	Rs1	Rs2	101

Performs bitwise OR of Rs1 and Rs2 and stores the result in Rd.

XOR

OP-CODE	RD	RS1	RS2	FUNC
0001	Rd	Rs1	Rs2	110

Performs bitwise XOR of Rs1 and Rs2 and stores the result in Rd.

XNOR

OP-CODE	RD	RS1	RS2	FUNC
0001	Rd	Rs1	Rs2	111

Performs bitwise XNOR of Rs1 and Rs2 and stores the result in Rd.

NOT

OP-CODE	RD	RS1	Reserved	FUNC
0010	Rd	Rs1	000	000

Performs bitwise NOT of Rs1 and stores the result in Rd.

SHIFTL

OP-CODE	RD	RS1	Reserved	FUNC
0010	Rd	Rs1	000	001

Shifts Rs1 by one place to the left and stores the result in Rd.

SHIFTR

OP-CODE	RD	RS1	Reserved	FUNC
0010	Rd	Rs1	000	010

Shifts Rs1 by one place to the right and stores the result in Rd.

CP

OP-CODE	RD	RS1	Reserved	FUNC
0010	Rd	Rs1	000	011

Copies Rs1 to Rd.

MOVIL

OP-CODE	RD	FUNC	IMMEDIATE (unsigned)
0011	Rd	0	#8-bit

Copies immediate value into lower byte of Rd.

MOVIH

OP-CODE	RD	FUNC	IMMEDIATE (unsigned)
0011	Rd	1	#8-bit

Copies immediate value into higher byte of Rd.

ADDI

OP-CODE	RD	RS1	IMMEDIATE (unsigned)
0100	Rd	Rs1	#6-bit

Adds a 6-bit unsigned value to Rs1 and stores the sum in Rd.

SUBI

OP-CODE	RD	RS1	IMMEDIATE (unsigned)
0101	Rd	Rs1	#6-bit

Subtracts a 6-bit unsigned value from Rs1 and stores the difference in Rd.

LD

OP-CODE	RD	RS1	IMMEDIATE (unsigned)
0110	Rd	Rs1	#6-bit

Loads Rd from the memory address given by [Rs1 + 6 bit unsigned value].

ST

OP-CODE	RD	RS1	IMMEDIATE (unsigned)
0111	Rd	Rs1	#6-bit

Stores Rd at the memory address given by [Rs1 + 6 bit unsigned value].
In spite of the name and position, Rd acts as a source register for this instruction – the value to be stored to memory.

BEQ

OP-CODE	RS1	RS2	IMMEDIATE (signed)
1000	Rs1	Rs2	#6-bit

Branches to the relative PC address given by the signed 6-bit immediate value if (Rs1 == Rs2).

BGE

OP-CODE	RS1	RS2	IMMEDIATE (signed)
1001	Rs1	Rs2	#6-bit

Branches to the relative PC address given by the signed 6-bit immediate value if (Rs1>= Rs2)

BLE

OP-CODE	RS1	RS2	IMMEDIATE (signed)
1010	Rs1	Rs2	#6-bit

Branches to the relative PC address given by the signed 6-bit immediate value if (Rs1 <= Rs2).

BC

OP-CODE	Reserved	Reserved	IMMEDIATE (signed)
1011	000	000	#6-bit

Branches to the relative PC address given by the signed 6-bit immediate value if the Carry flag is set.

J

OP-CODE	IMMEDIATE (signed)
1100	#12-bit

Jumps to the relative PC address given by the signed 12-bit immediate value

STC

OP-CODE	FUNC
1111	000000000001

Set the Carry flag

STB

OP-CODE	FUNC
1111	000000000010

Set the Borrow flag

RESET

OP-CODE	FUNC
1111	101010101010

Reset the CPU.

HALT

OP-CODE	FUNC
1111	111111111111

Halt the CPU