OPCODES for 16-bit RISC microprocessor.

GENERAL:

5 bit opcode
3 bit operand fields (Rd, Rs1, Rs2) to specify one of the 8 general purpose registers.
Immediates are restricted in length to the number of bit positions remaining in a 16 bit instruction word.
ALL instructions are 16 bits in length.
Shift immediates can be restricted to 2 bits.

INSTRUCTIONS:

Rd - destination
Rs1 - source 1
Rs2 - source 2
- 5/8/11-bit immediate

Table 1: OPCODES

Name	Opcode	Format	Notes
SEQ (SetIfEq)	00000	SEQ Rd Rs1 Rs2	Set register Rd to 1 if the values in Rs1 and Rs2 are equal, else set to 0.
ADD	00001	ADD Rd Rs1 Rs2	Add registers Rs1 and Rs2 and store the result in Rd.
SGT (SetIfGtr)	00010	SGT Rd Rs1 Rs2	Set register Rd to 1 if Rs1 > Rs2, else set to 0.
ADDI	00011	ADDI Rd Rs #	Store in register Rd the value Rs + the 5-bit **sign-extended** immediate.
SUB	00100	SUB Rd Rs1 Rs2	Compute Rs1 - Rs2 and store the result in Rd.
SLT (SetIfLes)	00101	SLT Rd Rs1 Rs2	Set register Rd to 1 if Rs1 < Rs2, else set to 0.
Unused	00110		
OR	00111	OR Rd Rs1 Rs2	Store in register Rd the bitwise OR of Rs1 and Rs2.
ORI	01000	ORI Rd Rs #	Store in register Rd the bitwise OR of Rs and the 5-bit **zero-extended** immediate.
AND	01001	AND Rd Rs1 Rs2	Store in register Rd the bitwise AND of Rs1 and Rs2.
ANDI	01010	ANDI Rd Rs #	Store in register Rd the bitwise AND of Rs and the 5-bit **zero-extended** immediate.
XOR	01011	XOR Rd Rs1 Rs2	Store in register Rd the bitwise XOR of Rs1 and Rs2.
XNOR	01100	XNOR Rd Rs1 Rs2	Store in register Rd the bitwise XNOR of Rs1 and Rs2.
NOT	01101	NOT Rd Rs	Store in register Rd the bitwise complement Rs.
SRA (ShftRgtArith)	01110	SRA Rd Rs #	Store in register Rd the **sign-extended** value of Rs shifted to the right by the 2-bit immediate. (An immediate of 0 does not shift the operand).

Table 1: OPCODES

Name	Opcode	Format	Notes
SRL (ShftRghtLogic)	01111	SRL Rd Rs #	Store in register Rd the **zero-extended** value of Rs shifted to the right by the 2-bit immediate.
SLL (ShftLeftLogic)	10000	RLL Rd Rs #	Store in register Rd the **zero-extended** value of Rs shifted to the left by the 2-bit immediate.
SW (StoreWord)	10001	SW 0 Rs Raddr	Store to memory at address Raddr the value in Rs. (You can assume that address is an even number and the compiler inserts 3 zero bits for Rd).
MUL (Multiply)	10010	MUL Rd Rs1 Rs2	Multiply the lower 8 bits of Rs1 and Rs2 and store the result in Rd.
Unused	10011		
LW (LoadWord)	10100	LR Rd 0 Raddr	Load the word value at memory address Raddr into Rd. (You can assume that address is an even number and the compiler inserts 3 zero bits for Rs).
LBI (LoadBytImmed)	10101	LBI Rd #	Store the 8-bit **sign-extended** immediate into register Rd after sign extending it.
LBIU (LoadBytIUnsign)	10110	LBIU Rd #	Write the 8-bit **zero-extended** immediate into register Rd.
LHI (LoadHighImmed)	10111	LHI Rd #	Write the 8-bit immediate into the upper 8 bits of register Rd and optionally clear the low order 8 bits.
BEQZ (BrIfEqZero)	11000	BEQZ Rs #	Set PC to PC + 8-bit **sign-extended** immediate if the value in register Rs is zero.
BNEZ (BrIfNotZero)	11001	BNEZ Rs #	Set PC to PC + 8-bit **sign-extended** immediate if the value in register Rs is non-zero.
BC (BrIfCarrySet)	11010	BC #	Set PC to PC + 11-bit **sign-extended** immediate if the carry out is set.
BO (BrIfOvrflowSet)	11011	BO #	Set PC to PC + 11-bit **sign-extended** immediate if the overflowis set.
NOP (NoOperation)	11100	NOP	Do nothing.
J (Jump)	11101	J #	Set the PC to PC + 11-bit **sign-extended** immediate.
JR (JumpToRegister)	11110	JR 0 Rs	Set the PC to the value in register Rs.
JALR (JumpAndLink)	11111	JALR Rd #	Rd=PC, PC=PC+#. Save the current value of PC (which points to the NEXT instruction) to register Rd and set PC to PC + 8-bit **sign-extended** immediate.

NOTES: All PC-relative addresses are sign-extended to allow both forward and backward branches.

'Unused' opcodes may be optionally encoded for other instructions, e.g. LLI (LoadLowImmed preserving the high order bits of destination register).

You can assume all addresses (whether for instruction or data) are word aligned.

You must increment PC by 2 by the end of the 2nd clock cycle so it is available for PC-relative branch additions such as BEQZ.