16-Bit CPU ISA

Marco A. Zuniga

Table 1: Register Type

ADD rc, rb, ra	0000(15:12), rc(11:8), rb(7:4), ra(3:0) Add value in register ra to register rb and save result in register rc.
SUB rc, rb, ra	0001(15:12), rc(11:8), rb(7:4), ra(3:0) Subtract value in register ra from register rb and save result in rc.
AND rc, rb, ra	0010(15:12), rc(11:8), rb(7:4), ra(3:0) AND contents in registers rb and ra and save result in register rc.
OR rc, rb, ra	0011(15:12), rc(11:8), rb(7:4), ra(3:0) OR contents in registers rb and ra and save result in register rc.
SLT rc, rb, ra	0100(15:12), rc(11:8), rb(7:4), ra(3:0) Set register rc to 1 if value in rb is less than value in ra. Else 0.

Table 2: Immediate 1 (1 Register) Type

LDI ra, #	0101(15:12), ra(11:8), #(7:0) Load # into bits (7:0) of register ra.
J Label	0110(15:12), ra(11:8), #(7:0) Concatenate bits[7:0] which correspond to 'Label' to bits[15:8] of PC.
JLR Label	0111(15:12), ra(11:8), #(7:0) Set register $ra to PC + 1 and concatenate bits[7:0] which correspond to 'Label' to bits[9:8] of PC.
JR $ra	1000(15:12), ra(11:8), #(7:0) Load PC with value in $ra.
HALT	1001(15:12), ra(11:8), #(7:0) Halt the program.

Table 3: Immediate 2 (2 Registers) Type

LDR rb, #(ra)	1010(15:12), rb(11:8), ra(7:4), #(3:0) Load value at memory location ra with offset # into register rb.
STR rb, #(ra)	1011(15:12), rb(11:8), ra(7:4), #(3:0) Store value in register rb to memory location ra with offset #.
BEQ rb, ra, Label	1100(15:12), rb(11:8), ra(7:4), #(3:0) Branch to location 'Label' if value of register rb equals value in register ra. Bits [9:4] of PC will be appended to #.
BNE rb, ra, Label	1101(15:12), rb(11:8), ra(7:4), #(3:0) Branch to location 'Label' if value of register rb does not equal value in register ra. Bits [9:4] of PC will be appended to #.

Registers

Table 4: User-Programmable Registers

Register	**Purpose**
$s0-$s4	Saved data/addresses
$t0-$t5	temporary data/addresses
$arg0-$arg1	Memory locations for arguments
$v0	Return value
$zero	Constant 0
$ra	Return address(Recommended to not modify)