INSTRUCTION SET

Instructions are 13-bits wide. The first 5 bits are opcodes, and the latter 8-bits are data bits. The 5 opcode bits have the following meaning:

(1) Im / Immediate. 1 if data bits are going straight to ALU, 0 if data bits are an address to be stored/loaded
(2) LS / Load Store. 1 if first pass going to store, 0 if going to load1
(3) Ret Return. This is used for indirect addressing. 1 means that data will be 'returning' to either load2 or store for a second time.
(4) ALU1. ALU control bit 1
(5) ALU2. control bit 2

The ALU receives two control bits for a total of 4 operations:

Controls bits	Operation	Description
00	Add	Adds data to acc
01	Sub	Subtracts data from acc
10	Branch	Feeds data to pc-out if acc is not 0
11	Clear	Sets acc to 0

This gives us the following instructions:

Instruction	Im	LS	Ret	ALU1	ALU2	Description
Store	0	1	0	x	x	Stores acc at memory location of data bits
StoreIndr	0	1	1	x	x	Loads data, then stores acc at that location
Add	0	0	0	0	0	Loads data adds to acc
Addi	1	x	x	0	0	Adds data to acc
AddIndr	0	0	1	0	0	Indirectly adds data to acc
Sub	0	0	0	0	1	Loads data subs from acc
Subi	1	x	x	0	1	Subs data from acc
SubIndr	0	0	1	0	1	Indirectly subs data from acc
Clear	1	x	x	1	1	Sets acc to 0
Branch	1	x	x	1	0	Relative branch if acc is not 0

Notice there is no load operation. This can be done in two instructions with $Clear, AddX$. Also, the architecture leaves the responsibility of avoiding control hazards to the programmer. This means a nop must be performed after every branch. This can be done with a *addi* 0 instruction.