RAILS Instruction Set Developer Guide

Instruction Operand Types

3 Operand Instruction

Op Code 4 bit	A operand 4 bit	B operand 4 bit	C operand 4 bit

Immediate Instruction

Op Code 4 bit	Immediate 8 bit	C operand 4 bit

Instructions / Descriptions

Name	Op code	Operand Type	Description
ADD	0000	3 Operand	A + B = C
ADDC	0001	3 Operand	A + B + (previous operation carry out) = C
SUB	0010	3 Operand	A - B = C
SWB	0011	3 Operand	A - B - (previous operation carry out) = C
NAND	0100	3 Operand	A NAND B = C
RSFT	0101	NULL	Logical right shift A, stores in C.
LIMM	0110	Immediate	Stores Immediate in C.
LD	0111	NULL	Loads data from ram address *A and stores at C.
LDIM	1000	Immediate	Loads data from ram address IMM and stores at C.
ST	1001	NULL	Stores B in ram address *A.
STIM	1010	Immediate	Stores B in ram address IMM.
BEQ	1011	Immediate	Branch to IMM if r15 and C are equal.
BGT	1100	Immediate	Branch to IMM if r15 is greater than C.
JMPL	1101	NULL	Jumps to *A and stores the current instruction address + 1 in C.
IN	1110	NULL	Loads data from i/o port A and stores at C.
OUT	1111	NULL	Stores B in i/o port A.

Description syntax

*X	This is a pointer, meaning you are using the contents of the register as an address. That could be either an address for ram or the program counter. For example if 23 is in register 2 and you jump to register 2, the pc will be set to 23. If you LD from register 2 then ram address 23 will be stored in C.
rX	This is an abbreviation for “register”, so r15 is just short for “register 15”.

Pseudo Instructions

NOP	This operation does nothing and is replaced by ADD r0 r0 r0
MOV	This moves data from one register to another. It is replaced by ADD source r0 destination
JMP	This jumps to imm by doing BEQ imm r15, since r15 will obviously equal itself it will always jump
EXIT	This stops all execution and the clock. It is replaced by JMPL r0, r0. It also resets the PC. This puts the cpu in a state where it is ready to be started again using the power button on the control panel.

Memory

Register File	The register file is 16 bytes of dual read registers. • Register 0 is a constant 0. Writing to address 0 results in nothing being saved. Reading from it will always be 0. • Register 15 will always be used in branch instructions with C
Input/Output Ports	My implementation has 8 ports you can read / write to. Since the address is a 4 bit value if you made your own cpu using this isa you could go up to 16 ports. On mine any address above 0-7 it just ignores.
Data Storage	Data storage is 256 bytes of RAM.
Instruction Storage	Instructions are stored in 512 bytes of ROM, since instructions are 16 bit that means you have a max of 256 instructions.

Path Diagrams

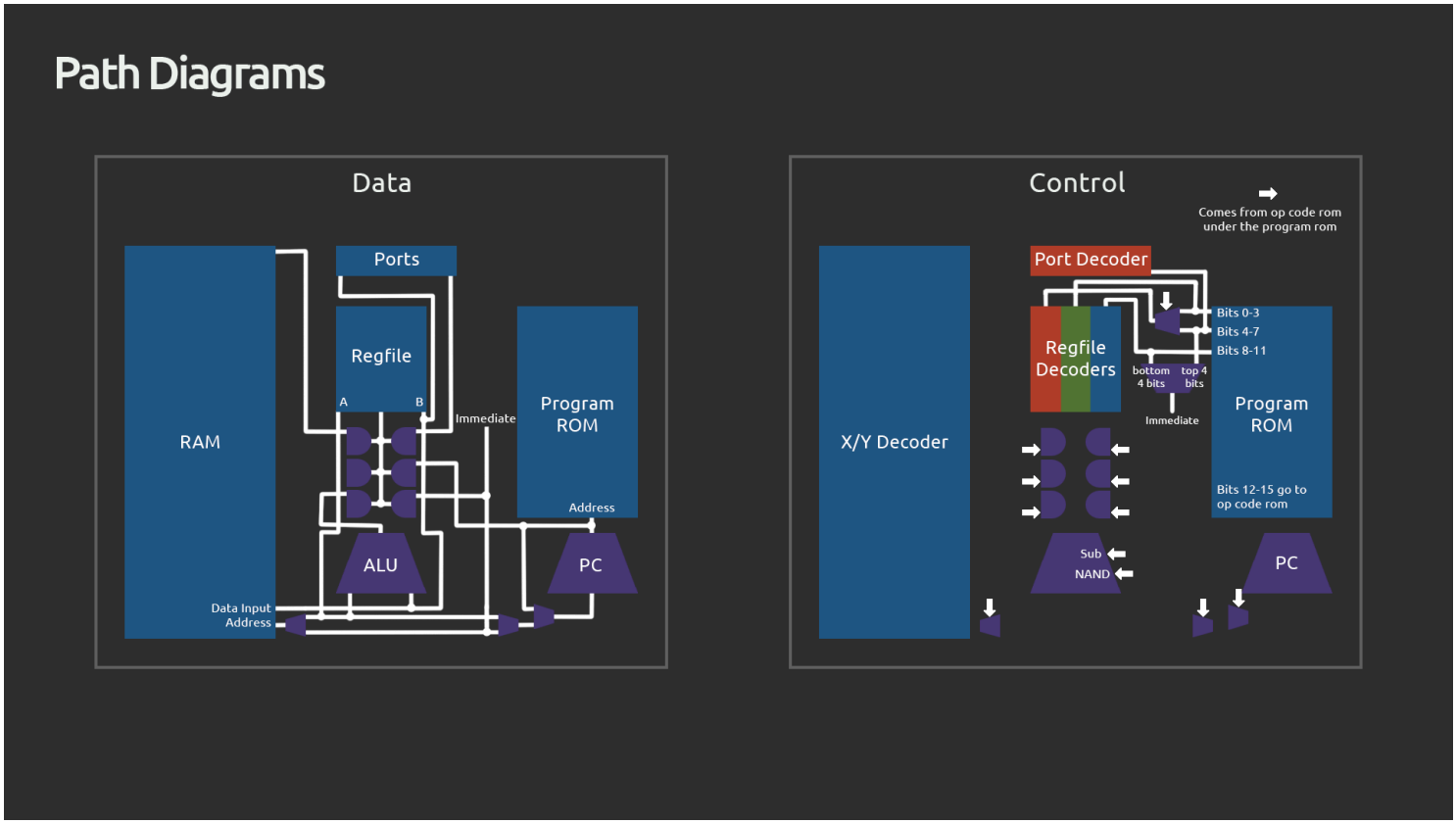

*diagram might be missing a few labels etc

[link to above pic]

Pipeline

Buffered 2 stage*

Assembly / Assembler

Test

Emulator

Notes

- This ISA was developed with the following traits in mind.
    - Newcomer Friendly.
    - Made for 8 bit word size.
    - Easy to understand.
    - Easy to implement.
    - Be fully functional / capable of general computation.