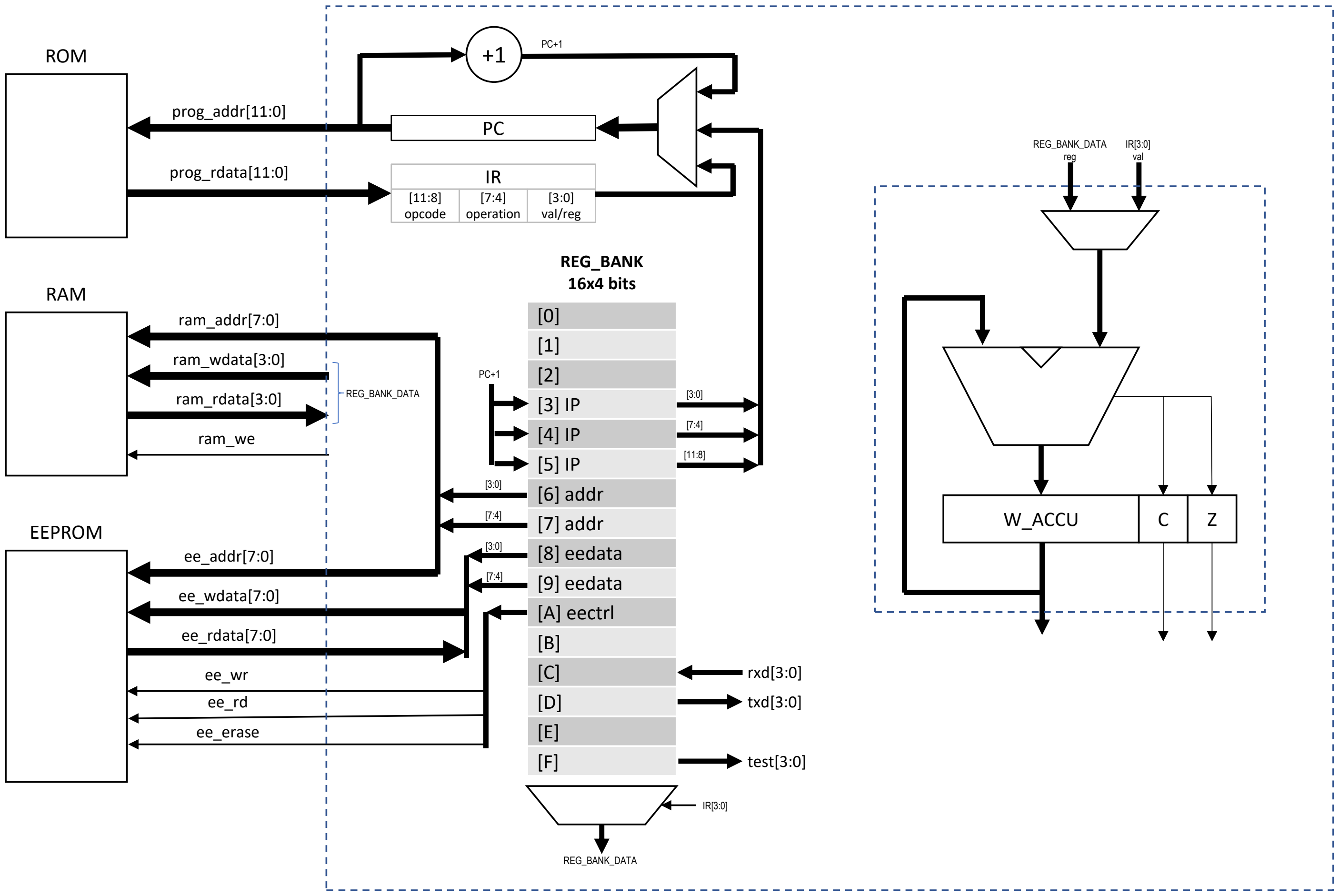
RISC4B

Features

RISC based architecture
True atomic 1 clock cycle per instruction
4 bits datapath
12 bits instruction length
16 x 4 bits registers
4 wires test block
4k x 12 bits program memory
Addressable 256 x 4 bits data memory
Addressable 256 x 8 bits EEPROM

Overview



PC: Program Counter

IR: Instruction Register

IP: Instruction pointer

Register Map

Reg	**Function**	**[3]**	**[2]**	**[1]**	**[0]**
h0	R0				
h1	R1				
h2	R2				
h3	IP0	IP[3]	IP[2]	IP[1]	IP[0]
h4	IP1	IP[7]	IP[6]	IP[5]	IP[4]
h5	IP2	IP[11]	IP[10]	IP[9]	IP[8]
h6	RAM/EE Address 0	ADDR[3]	ADDR[2]	ADDR[1]	ADDR[0]
h7	RAM/EE Address 1	ADDR[7]	ADDR[6]	ADDR[5]	ADDR[4]
h8	EED0	DATA[3]	DATA[2]	DATA[1]	DATA[0]
h9	EED1	DATA[7]	DATA[6]	DATA[5]	DATA[4]
hA	EECTRL		EEER	EERD	EEWR
hB	-				
hC	RxD	RxDI	RxStart	RxDValid	RxACK
hD	TxD	TxDO	TxStard	TxDValid	TxACK
hE	-				
hF	TEST				

ISA - Instruction Set Architecture

Instruction	IR			C/Z	Binary operations with direct value
	11 :8	7 :4	3 :0		
ADDI	*h9*	*hC*	Val	XX	W = W + Val
ADDCI	*h9*	*hF*	Val	XX	W = W + Val + Carry
SUBI	*h9*	*hA*	Val	XX	W = W – Val
SUBCI	*h9*	*h9*	Val	XX	W = W – Val – (1 – Carry)
ANDI	*h9*	*h4*	Val	XX	W = W and Val
ORI	*h9*	*h7*	Val	XX	W = W or Val
XORI	*h9*	*h6*	Val	XX	W = W xor Val
PASSI	*h9*	*h5*	Val	XX	W = Val
CMPI	*h9*	*hB*	Val	XX	see Notes
					Binary operations with indirect value
ADD	*h8*	*hC*	Reg	XX	W = W + Reg
ADDC	*h8*	*hF*	Reg	XX	W = W + Reg + Carry
SUB	*h8*	*hA*	Reg	XX	W = W – Reg
SUBC	*h8*	*h9*	Reg	XX	W = W – Reg – (1 – Carry)
AND	*h8*	*h4*	Reg	XX	W = W and Reg
OR	*h8*	*h7*	Reg	XX	W = W or Reg
XOR	*h8*	*h6*	Reg	XX	W = W xor Reg
PASS	*h8*	*h5*	Reg	XX	W = Reg
CMP	*h8*	*hB*	Reg	XX	see Notes
					Unary operations with direct value
INCI	*hB*	*h4*	Val	XX	W = Val + 1
DECI	*hB*	*h6*	Val	XX	W = Val – 1
NOTI	*hB*	*h0*	Val	XX	W = not Val
SHRI	*hB*	*hE*	Val	XX	W = shr Val
SHLI	*hB*	*hC*	Val	XX	W = shl Val
RRCI	*hB*	*hA*	Val	XX	W = rrc Val
RLCI	*hB*	*h8*	Val	XX	W = rlc Val
					Unary operations with indirect value
INC	*hA*	*h4*	Reg	XX	W = Reg + 1
DEC	*hA*	*h6*	Reg	XX	W = Reg – 1
NOT	*hA*	*h0*	Reg	XX	W = not Reg
SHR	*hA*	*hE*	Reg	XX	W = shr Reg
SHLI	*hA*	*hC*	Reg	XX	W = shl Reg
RRC	*hA*	*hA*	Reg	XX	W = rrc Reg
RLC	*hA*	*h8*	Reg	XX	W = rlc Reg
					Jump and move instructions
JMPC	CC	Ad	Ad	--	If CC then PC[7:0]=Ad else PC=PC+1
JMPIC	*hF*	*hF*	CC	--	If CC then PC=IP else PC=PC+1
CALL	*hC*	Ad	Ad	--	PC[7:0] = Ad[7:0] ; IP=PC+1
CALLI	*hF*	*hF*	*hE*	--	PC=IP ; IP=PC+1
MOVI	*hD*	Reg	Val	--	Reg=Val
MOVW	*hF*	*h2*	Reg	--	Reg=W
STO	*hF*	*h0*	Reg	--	RAM(#Reg7/Reg6) = Reg *
LOAD	*hF*	*h1*	Reg	--	Reg = RAM(#Reg7/Reg6) *
NOP	*hF*	*hF*	*hF*	--	

CC = Condition; Ad = Address; Reg = Register address; Val = Absolute Value.
CMP (and CMPI) does not affect register W, but only Zero and Carry registers (used for comparisons)
SHL / SHR = shift left / right; 0's are inserted.
RLC / RRC = a rotate left / right; Carry is inserted.

Conditions

CC		If condition	
NC	Non Conditional	-	B000
EQ_ZS		Z=1	B111
NE_ZC		Z=0	B011
GE_CS	Greater or equal	C=1	B001
GT	Greater than	C=1 & Z=0	B010
LE	Less or equal	C=0 & Z=1	B110
LT_CC	Less than	C=0	B101