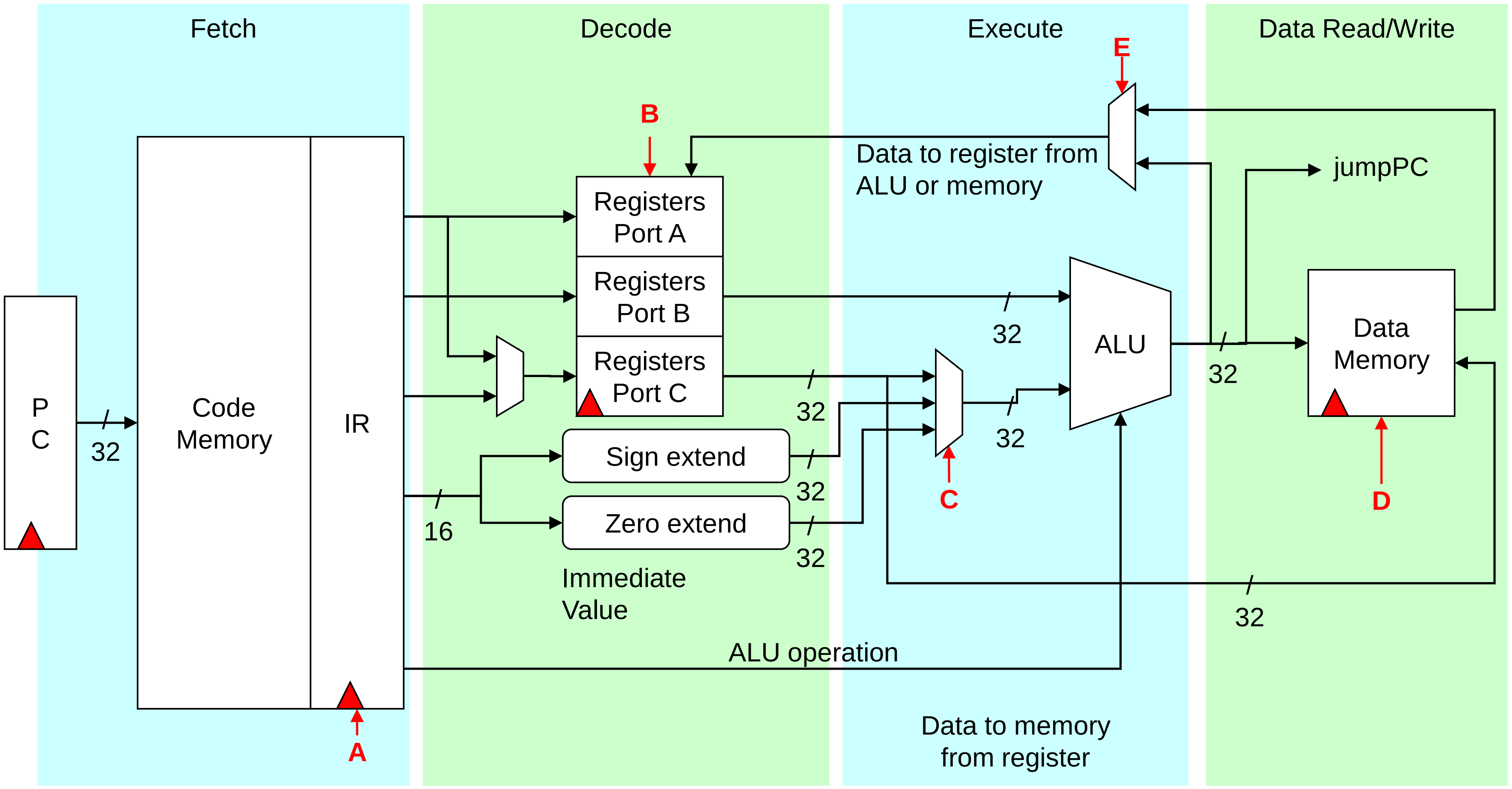
CPU32 Project

Overview

- “Classic” RISC machine
  - Load-store architecture (op r,mem)
  - Register-register operations
  - 3-address machine (op r1,r2,r3)
  - R0 = constant zero
- Non-pipelined implementation
  - Multi-cycle
  - Fetch-Decode-Execute-Data Read/Write
- Impractical instruction set!
  - Only supports 32-bit operands
  - Limited set of instructions



Control Signals

▲ Clocked element

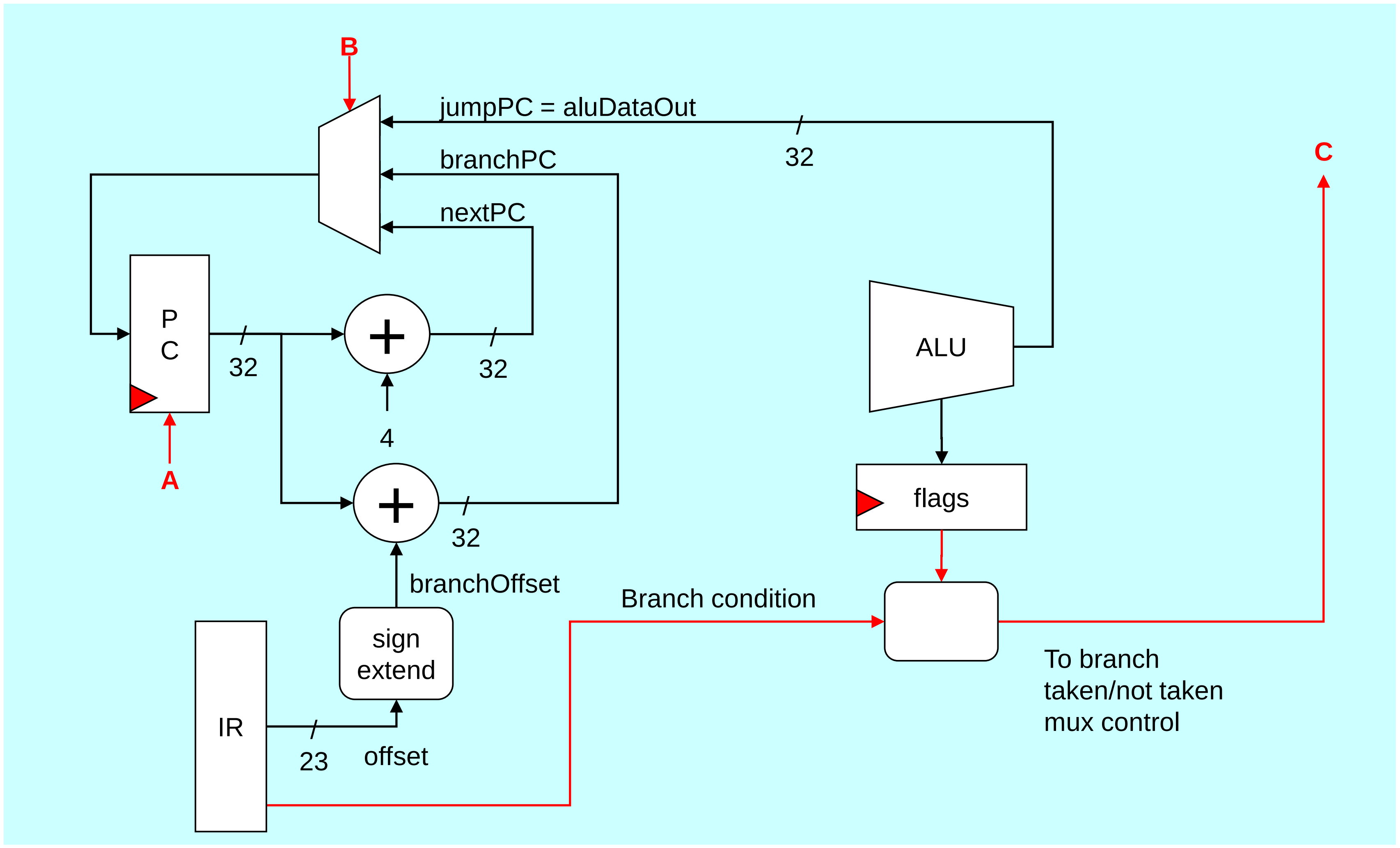
A. Code memory control (**loadIR**)

B. Register control (**regAWrite**)

C. Immediate/register operand select (inherent control)

D. Data memory control (**dataMemWrite**)

E. Register write data (**RegASource**)



<u>Control Signals</u>

A. PC load control (**loadPC**)

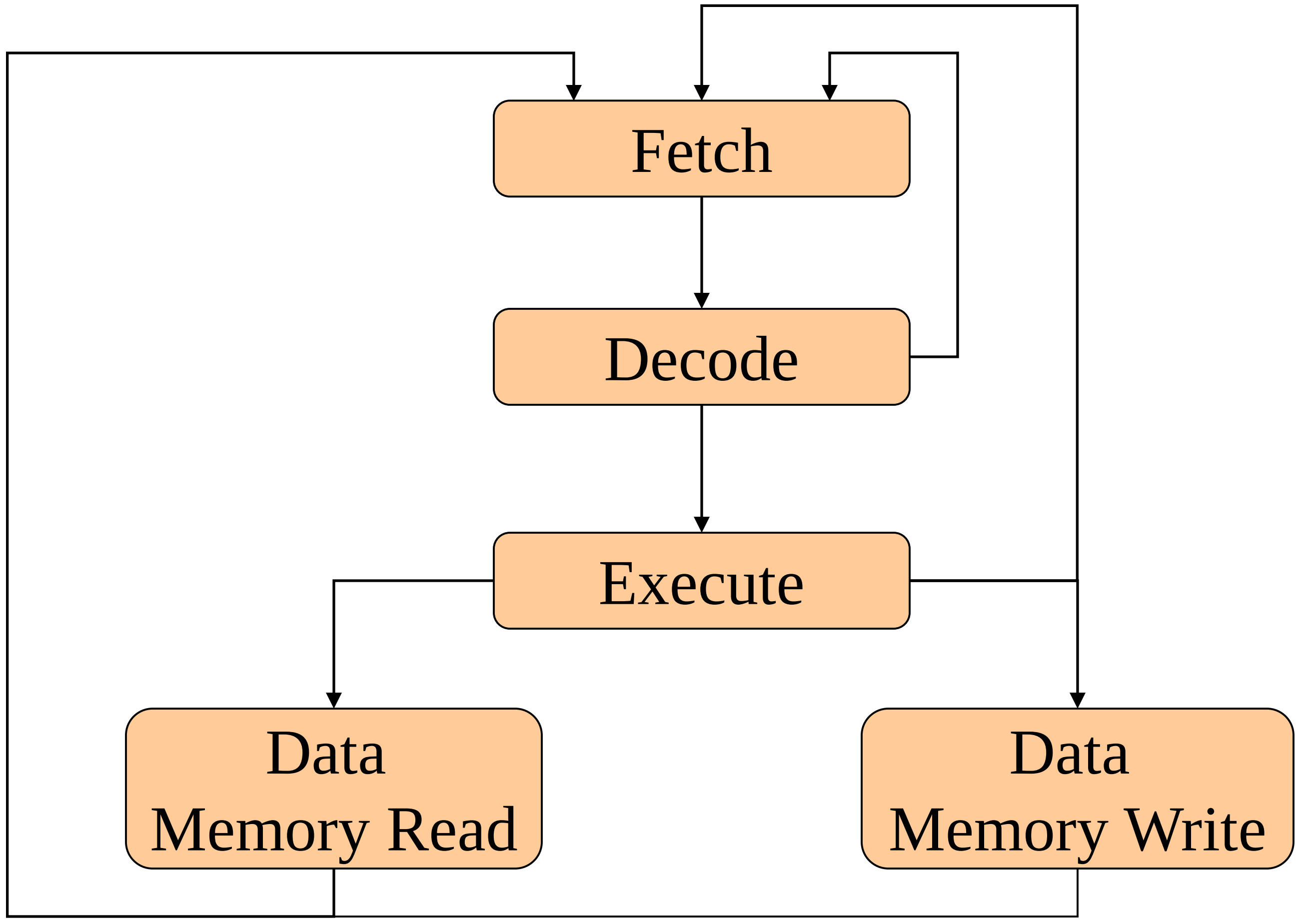
B. PC source control (**pcSource**)

C. Branch true/false

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Op			ALU Func			Reg A					Reg B					Reg C																RegA <- RegB op RegC	
0			0 (add)			1-31					0-31					0-31																ADD Ra,Rb,Rc	
0			1 (sub)			1-31					0-31					0-31																SUB Ra,Rb,Rc	
0			2 (and)			1-31					0-31					0-31																AND Ra,Rb,Rc	
0			3 (or)			1-31					0-31					0-31																OR Ra,Rb,Rc	
0			4 (xor)			1-31					0-31					0-31																EOR Ra,Rb,Rc	
0			5 (swap)			1-31					X					0-31																SWAP Ra,Rc	
0			6 (---)			1-31					0-31					0-31																	
0			7 (mul)			1-31					0-31					0-31																MUL Ra,Rb,Rc	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Op			ALU Func			Reg A					Reg B					15							Immediate Value								0	RegA <- RegB op sex/zex(Immed)	
1			0 (add)			1-31					0-31					0 <-> 65535																ADD Ra,Rb,#dddd	zero extend
1			1 (sub)			1-31					0-31					0 <-> 65535																SUB Ra,Rb,#dddd	zero extend
1			2 (and)			1-31					0-31					0 <-> 65535																AND Ra,Rb,#dddd	zero extend
1			3 (or)			1-31					0-31					0 <-> 65535																OR Ra,Rb,#dddd	zero extend
1			4 (xor)			1-31					0-31					-32768 <-> 32767																EOR Ra,Rb,#dddd	sign extend
1			5 (swap)			1-31					X					0 <-> 65535																MOVH Ra,#dddd	zero extend
1			6 (---)			1-31					0-31					0 <-> 65535																	
1			7 (mul)			1-31					0-31					0 <-> 65535																MUL Ra,Rb,#dddd	zero extend
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Op			X	Size		Reg A					Reg B					15							Offset								0	RegA <- mem(RegB + sex(Immed))	
2			X	X		1-31					0-31					-32768 <-> 32767																LD.b Ra,dddd(Rb)	
																																LD.b Ra,(Rb)	dddd = 0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Op			X			Reg A					Reg B					15							Offset								0	PC <- RegB + sex(Offset)	
2			X			0					0-31					-32768 <-> 32767																JMP dddd(Rb)	
																																JMP (Rb)	dddd=0
																																JMP dddd	Rb = R0
																																RTS	Rb = R31,dddd=0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Op			X	Size		Reg C					Reg B					15							Offset								0	mem(RegB + sex(Offset)) <- RegC	
3			X	X		0-31					0-31					-32768 <-> 32767																ST.b Rc,dddd(Rb)	
																																ST.b Rc,(Rb)	dddd = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Op			X		Conditon				Offset																							if (cond) PC <- PC + offset	
4			X		2-15				-2097152 <-> 2097151																								
21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Op			X						Offset																							PC <- PC + offset	
4			X		0				-2097152 <-> 2097151																							BRA Offset	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Op			X						Offset																							Reg31 <- PC, PC <- PC + offset	
4			X		1				-2097152 <-> 2097151																							BSR Offset	

Multi-Cycle Stages



CPUPackage.vhd

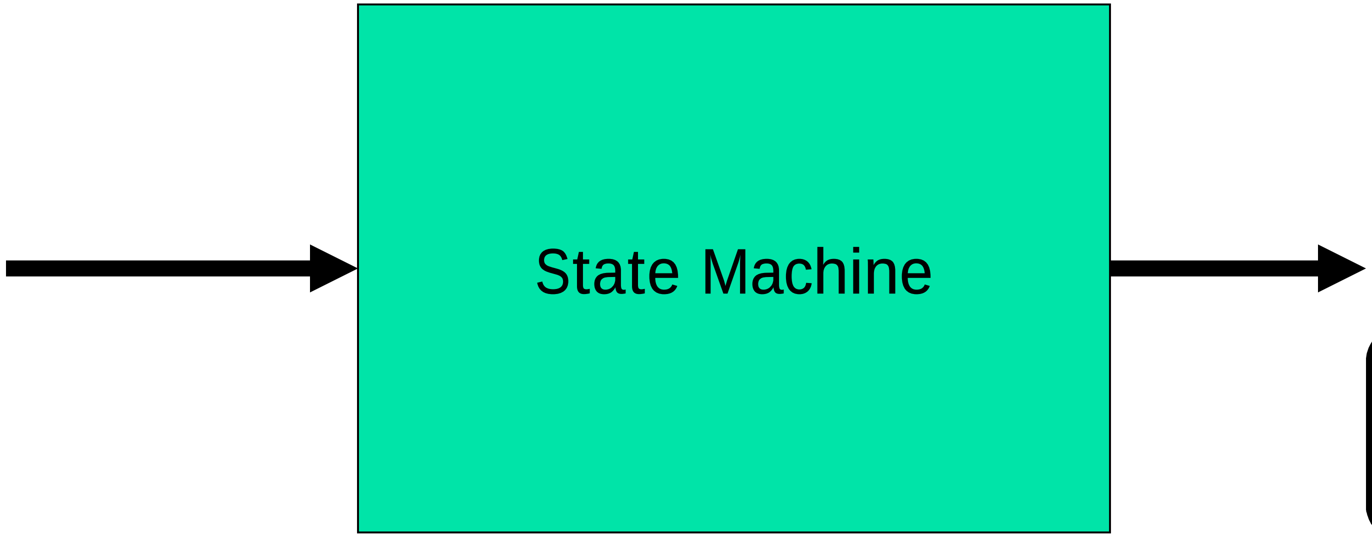
- Provides useful definitions and functions that are shared by multiple modules.
  - Definitions for field values of instruction
    - Alu operations (add, sub ...)
    - Branch conditions (beq, bne ...)
  - Functions to extract fields from instruction
    - ir_op() – opcode field
    - ir_aluOp() – alu operation field etc.
  - Extension function
    - Sex(), Zex()

CPU32.vhd

- Main data paths
  - Code memory
  - Instruction Register
  - Register File (32 32-bit registers)
  - ALU
  - Data Memory
- Program counter data paths
  - PC
  - Next address calculation
  - PC multiplexing.

Control.vhd

- Control State Machine
  - Decodes the instruction
  - Controls the data paths
  - Note: Some simple control is done in CPU32.vhd



ALU.vhd

- Implements a range of functions on one or two operands.
  - Arithmetic
    - Add, sub
  - Logical
    - And, or, xor
  - Misc
    - Pass operand through
    - Swap register halves

ALU.vhd

- Combined with R0, which always contains zero, we can obtain:
  - Sub R1,R0,R2 = Negate – R1 <= (0-R2)
  - Add R1,R0,R2 = Move – R1 <= (0+R2)
- Used with the same register
  - Zero – (Rn-Rn)
- Misc
  - eor R2,R2,#-1 = Complement
    - Note sign extended!

Memories

- CodeMemory (Read only = ROM)
  - Contents are loaded from a file when the VHDL is synthesized or a simulation is loaded
- DataMemory (Read/write)
- Registers
  - 3-Port memory
    - 1 Write port A
    - 2 Read ports B & C