

---

---

ELEC 374

Digital Systems Engineering

Winter 2018

CPU Design Project Tutorial

Ahmad Afsahi

Course web site: <http://post.queensu.ca/~afsahi/ELEC-374/index.htm>

# Instruction Set Architecture

---

- **Instruction Set Architecture (ISA):** is an abstract interface between the hardware and the lowest level software of a machine that encompasses all necessary information to write a machine language program that will run correctly, including instructions, registers, memory access, I/O, etc.
  - It involves decisions regarding: registers, memory addressing, addressing modes, instruction operands, available operations, control flow instructions, and instruction encoding.
- **Evaluating ISA:**
  - **Design-time metrics:**
    - ❖ Can it be implemented? With what performance, and at what costs (design, fabrication, test, packaging), with what power, and with what reliability?
    - ❖ Can it be programmed? Ease of compilation?
  - **Static Metrics:**
    - ❖ How many bytes does the program occupy in memory?
  - **Dynamic Metrics:**
    - ❖ How many instructions are executed? How many bytes does the processor fetch to execute the program?
    - ❖ How many clocks are required per instruction?
    - ❖ How “lean” a clock is practical?

# RISC Design Principles

---

## 1. Simplicity favours regularity

- Fixed size instructions
- Instructions to have (mostly) the same number of operands
- Opcode always a few bits of an instruction

## 2. Smaller is faster

- Limited instruction set
- Limited number of registers in register file
- Limited number of addressing modes

## 3. Good design demands good compromises

- Fixed size instructions → requiring different (but small number of) kinds of instruction formats

## 4. Make the common case fast

- Arithmetic operands from the register file (Load-Store machine)
- Allow instructions to contain immediate operands

# Designing a Simple RISC Computer (Mini SRC)

---

- Mini SRC is a variant of SRC in Lab Reader

- Processor State

PC<31..0>:	32-bit Program Counter (PC)
IR<31..0>:	32-bit Instruction Register (IR)
R[0..15]<31..0>:	Sixteen 32-bit registers named R[0] through R[15]
R[0..7]<31..0>:	Eight general-purpose registers
R[8..11]<31..0>:	Four Argument Registers
R[12..13]<31..0>:	Two Return Value Registers
R[14]<31..0>:	Stack Pointer (SP)
R[15]<31..0>:	Return Address Register (RA)
HI<31..0>:	32-bit HI Register to keep the high-order word of a Multiplication product, or the Remainder of a Division operation
LO<31..0>:	32-bit LO Register to keep the low-order word of a Multiplication product, or the Quotient of a Division operation

## Registers

**R0 – R15**

**PC**

**HI**

**LO**

# Designing a Simple RISC Computer (Mini SRC) - Cont'd

---

## ◦ Memory State

Mem[0..511]<31..0>:	512 words (32 bits per word) of memory
MDR<31..0>:	32-bit memory data register
MAR<31..0>:	32-bit memory address register

## ◦ I/O State

In.Port<31..0>:	32-bit input port
Out.Port<31..0>:	32-bit output port
Run.Out:	Run/halt indicator
Stop.In:	Stop signal
Reset.In:	Reset signal

# Mini SRC ISA Overview

## ◦ Instruction Categories:

- Arithmetic and Logical
- Load and Store
- Jump and Branch
- Input and Output
- Special and Miscellaneous

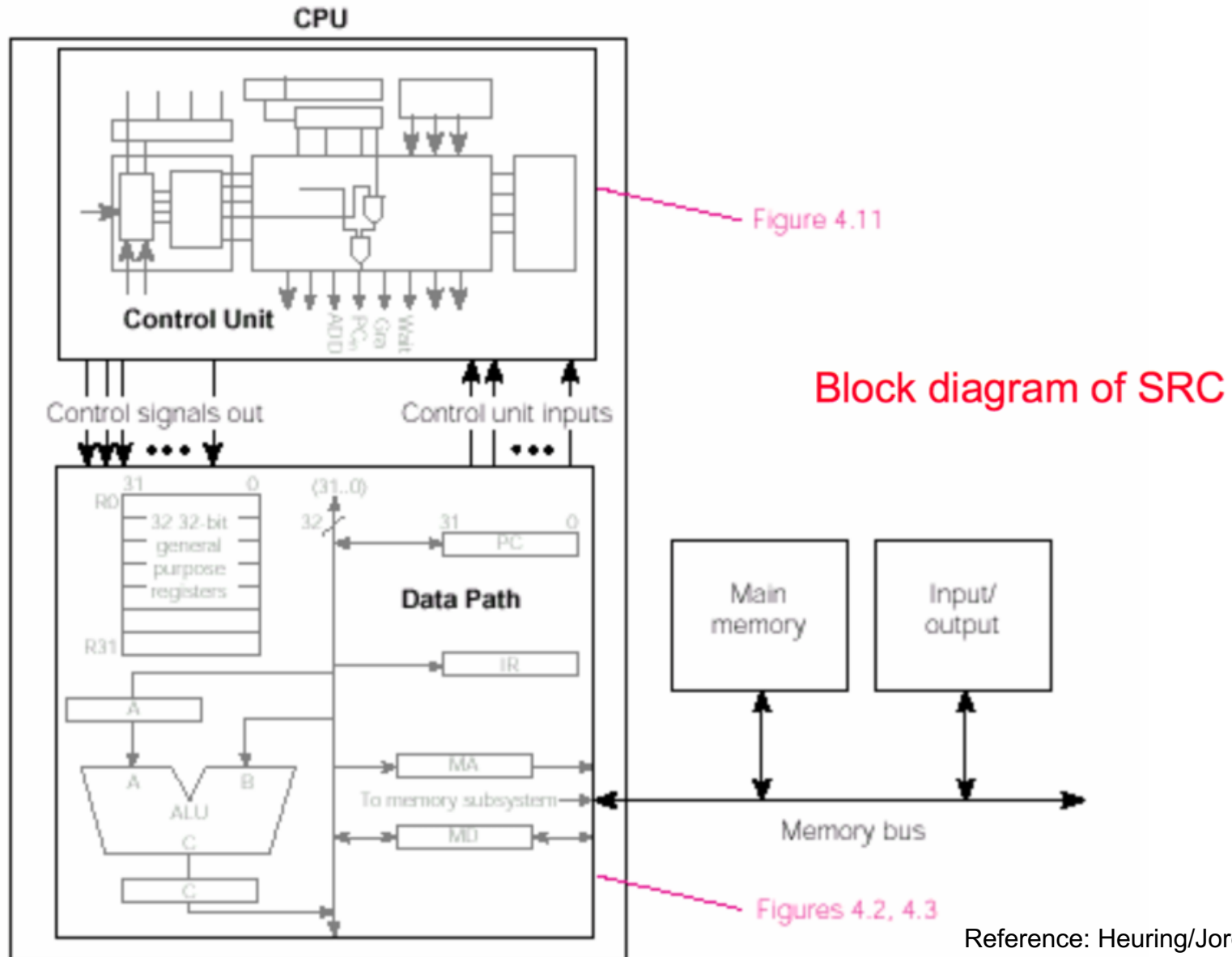
## ◦ Addressing Modes:

- Direct/Absolute
- Register
- Register indirect
- Indexed
- Immediate
- Relative

## ◦ Instruction Formats:

Name		Fields				Comments
Field size	31..27 5 bits	26..23 4 bits	22..19 4 bits	18..15 4 bits	14..0 15 bits	All instructions are 32-bit long
R-Format	OP	Ra	Rb	Rc	Unused	Arithmetic/Logical
I-Format	OP	Ra	Rb	Constant C/Unused		Arithmetic/Logical; Load/Store; Imm.
B-Format	OP	Ra	C2	Constant C		Branch
J-Format	OP	Ra	Unused			Jump; Input/Output; Special
M-Format	OP	Unused				Misc.

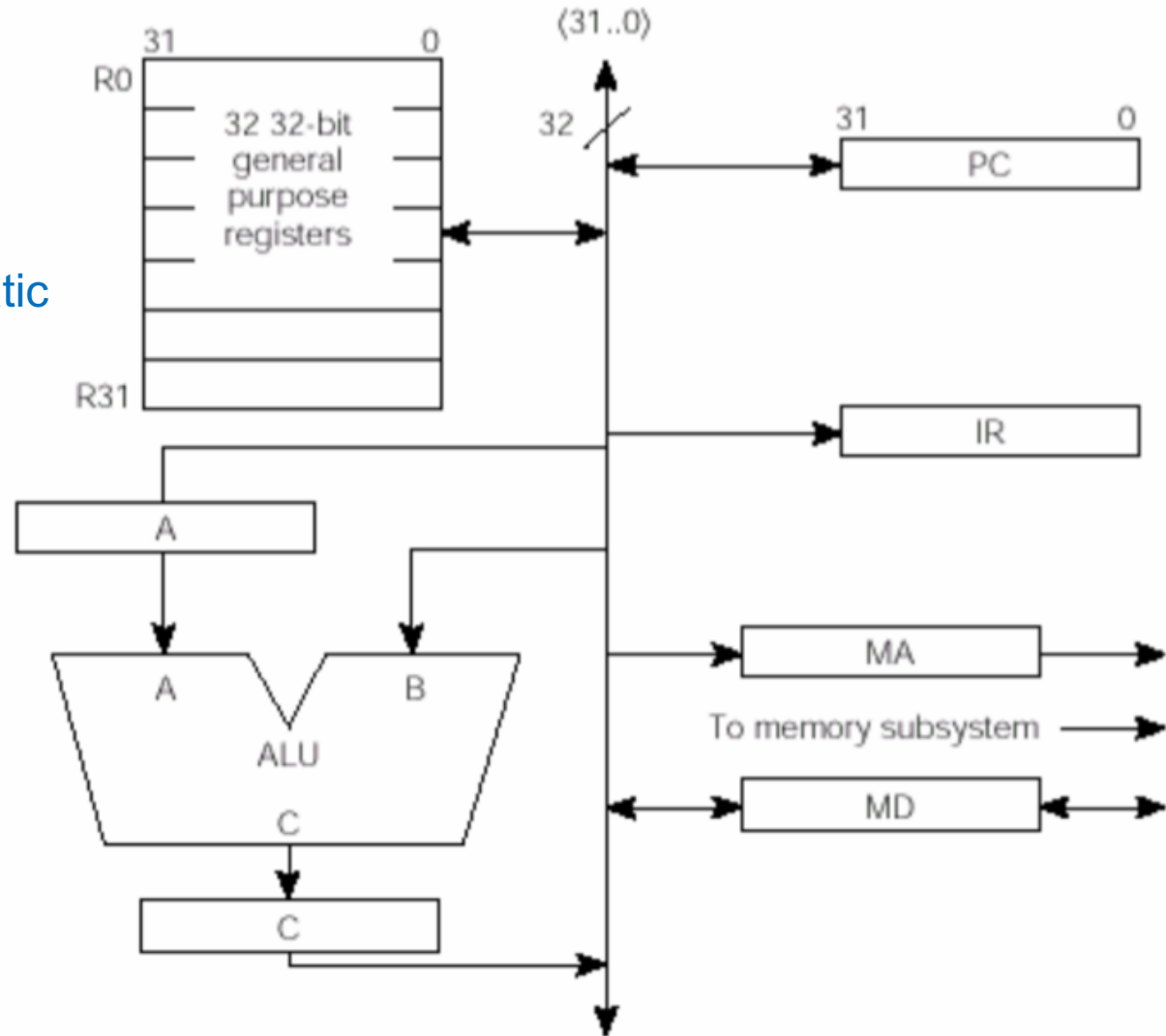
# SRC: Simple RISC Computer



Reference: Heuring/Jordan, Ch. 4  
([Lab Reader](#) on the course website)

# SRC: Simple RISC Computer (Cont'd)

You may design your Mini SRC using schematic capture, HDL, or in a mixed-mode fashion



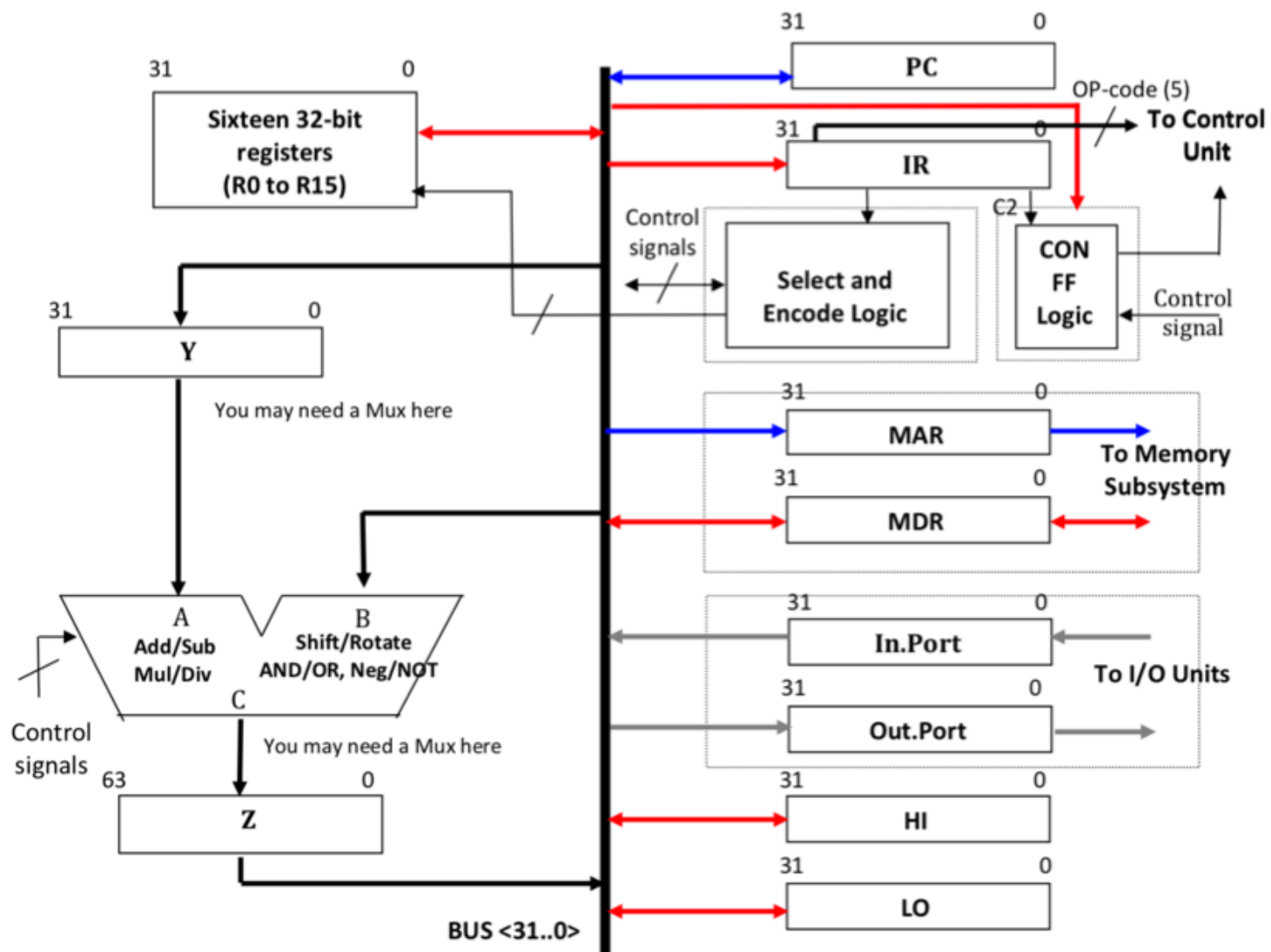
Reference: Heuring/Jordan, Ch. 4  
([Lab Reader](#) on the course website)

High-level view of the 1-Bus SRC design



# Mini SRC Datapath Design

## ◦ Lab Phase 1 and Phase 2



# Mini SRC Datapath Design (Cont'd)

---

## ◦ Design Steps in Phase 1:

- Design registers R0 to R15, PC, IR, Y, Z, MAR, HI, and LO
- Design the bidirectional Bus, and connect the registers that you designed in Step 1 to the Bus
- Design the MDR register and connect it to the BUS. Leave the rest of the Memory Subsystem (connection to the memory module) for Phase 2
- Design your ALU
  - ❖ Start with simple ALU operations such as AND, OR, and NOT. Then,
  - ❖ Design the more involved operations such as ADD/SUB, MUL, and DIV circuitry. Finally, design the rest of the ALU operations.
- Functionally simulate your datapath design in Phase 1
- Submit Phase 1 report (one per group)
- Get the required design operations done first
  - ❖ You may leave any other advanced design techniques (bonus marks) that you learned in class for later during the term

## Mini SRC Datapath Design (Cont'd)

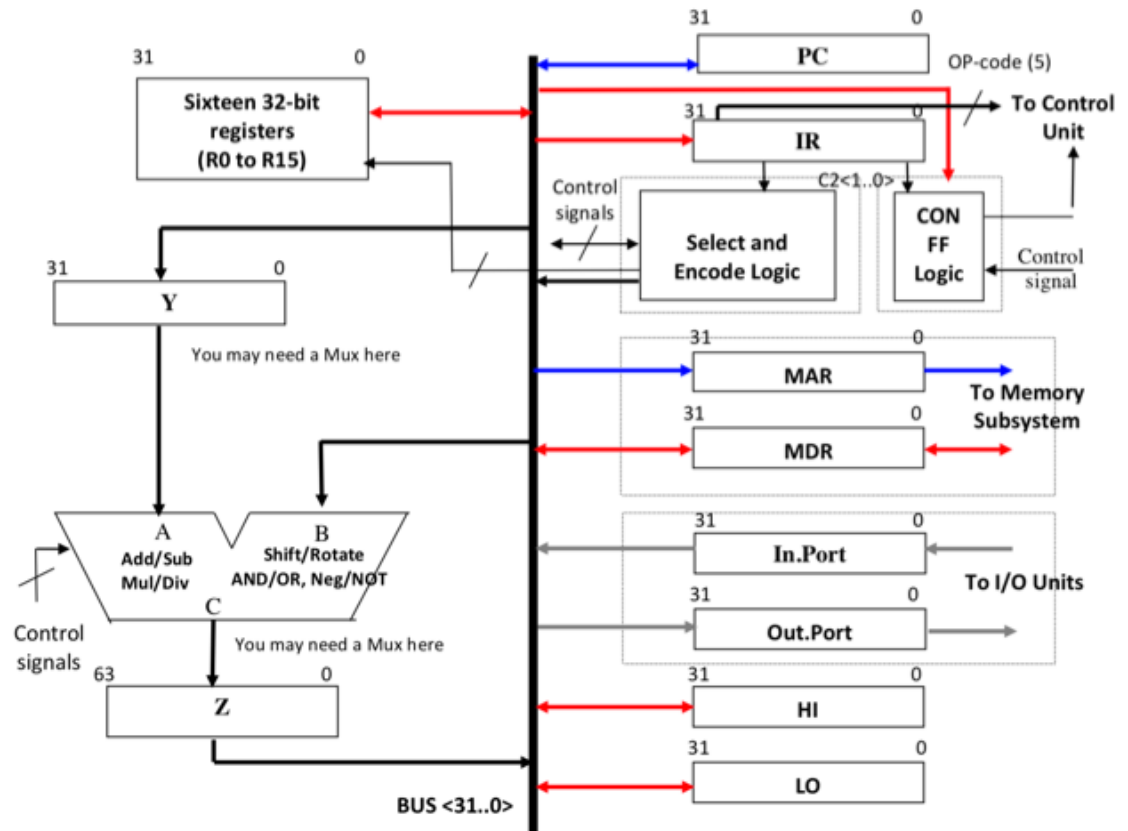
- Control sequence for add instruction:

### Control Sequence:

<u>Step</u>	<u>Control Sequence</u>
T0	PCout, MARin, IncPC, Zin
T1	Zlowout, PCin, Read, Mdatain[31..0], MDRin
T2	MDRout, IRin
T3	R2out, Yin
T4	R3out, ADD, Zin
T5	Zlowout, R1in

Diagram showing sixteen 32-bit registers (R0 to R15) with a red arrow pointing to the right.

Control signals are mostly generated in Phase 3; some in Phase 2.



# Mini SRC Datapath Design (Cont'd)

---

## ◦ Design Steps in Phase 2:

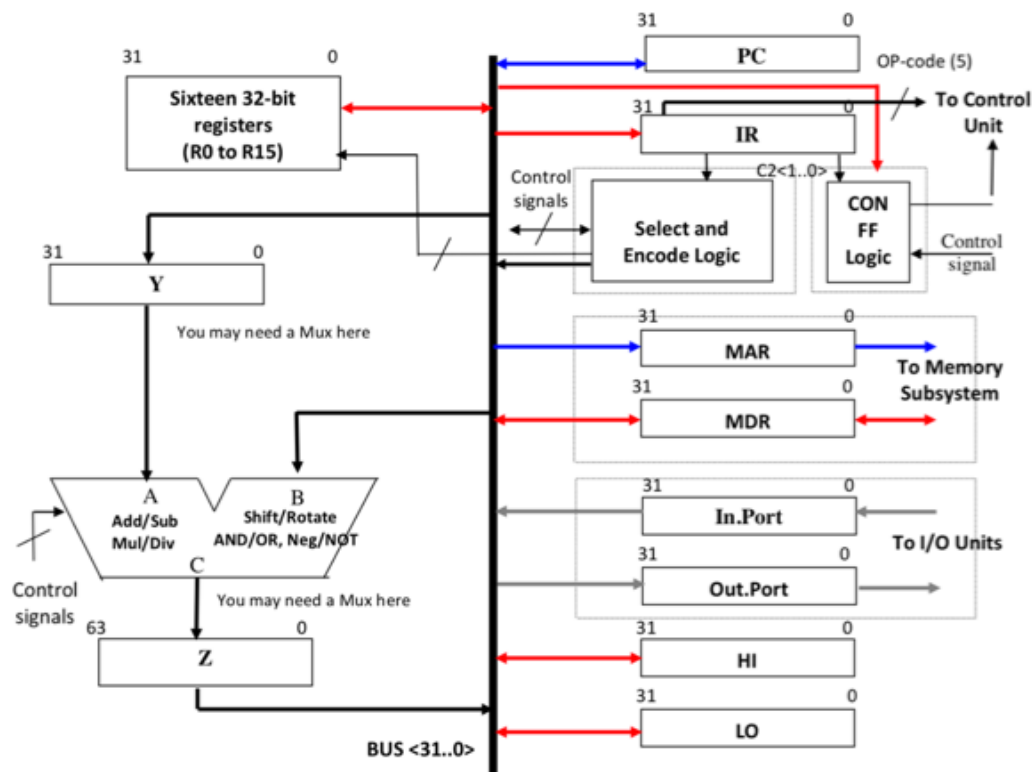
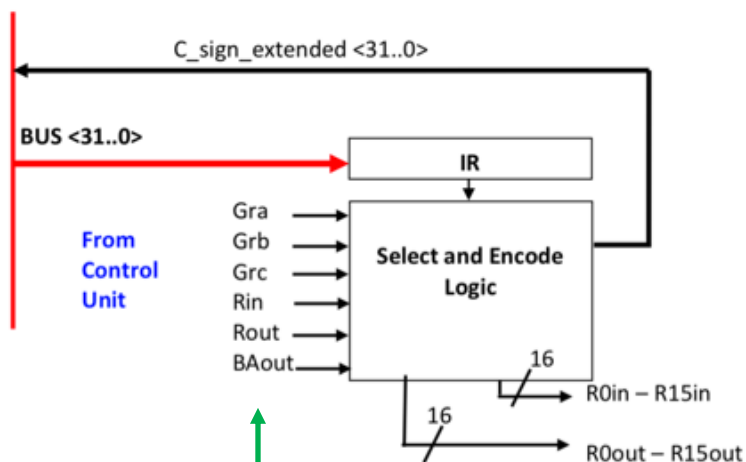
- Design the remaining parts of the Mini SRC datapath including
  - ❖ “Select and Encode” logic,
  - ❖ “CON FF” logic,
  - ❖ “Memory Subsystem”
  - ❖ “Input/Output” ports
- Functionally simulate your datapath design in Phase 2
- Submit Phase 2 report (one per group)
- Get the required design operations done first
  - ❖ You may then go after other advanced design techniques (bonus marks)

# Mini SRC Datapath Design (Cont'd)

## Control sequence for Load Indexed instruction:

### Control Sequence: ld

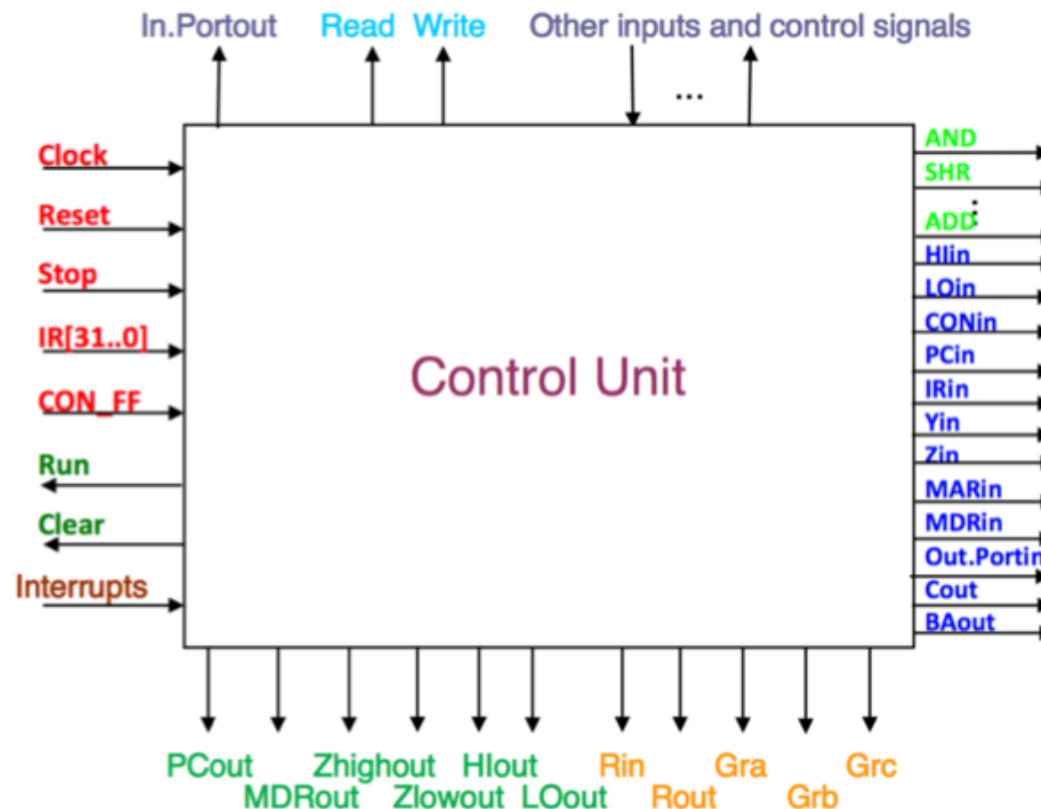
Step	Control Sequence
T0	PCout, MARin, IncPC, Zin
T1	Zlowout, PCin, Read, Mdatain[31..0], MDRin
T2	MDRout, IRin
T3	Grb, BAout, Yin
T4	Cout, ADD, Zin
T5	Zlowout, MARin
T6	Read, Mdatain[31..0], MDRin
T7	MDRout, Gra, Rin



# Mini SRC Datapath Design (Cont'd)

## ◦ Design Steps in Phase 3:

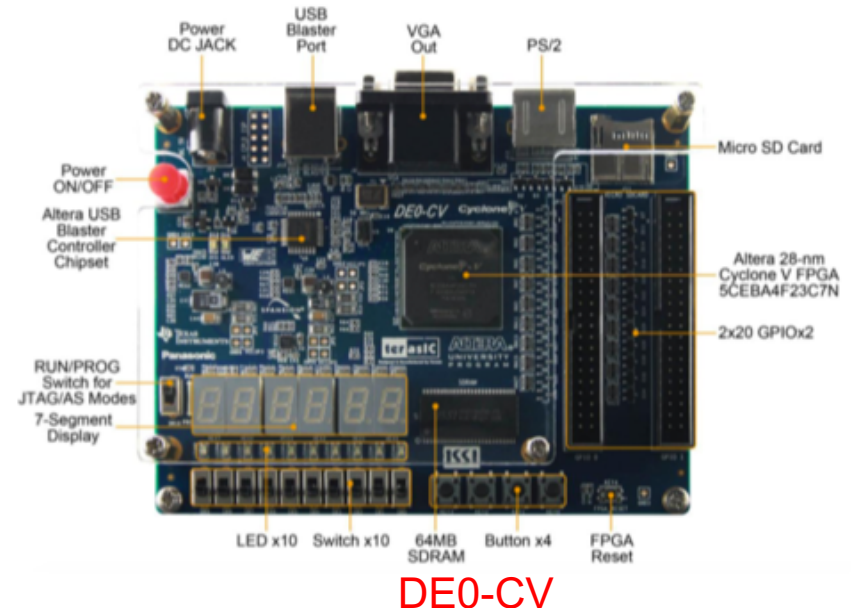
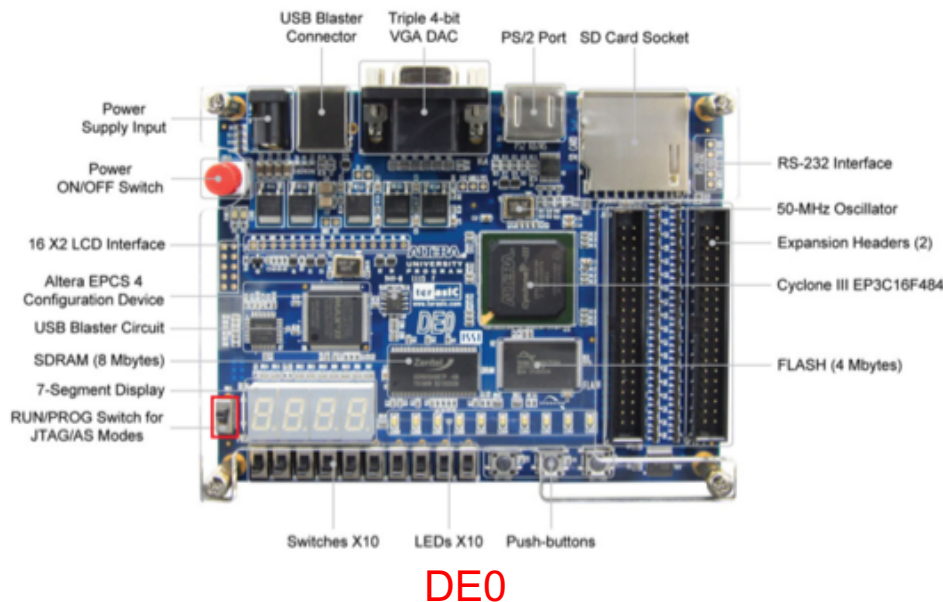
- Design the Control Unit of Mini SRC in VHDL or Verilog
- Functionally simulate your Control unit using a given test program
- Submit Phase 3 report (one per group)
- Continue your other designs for bonus mark



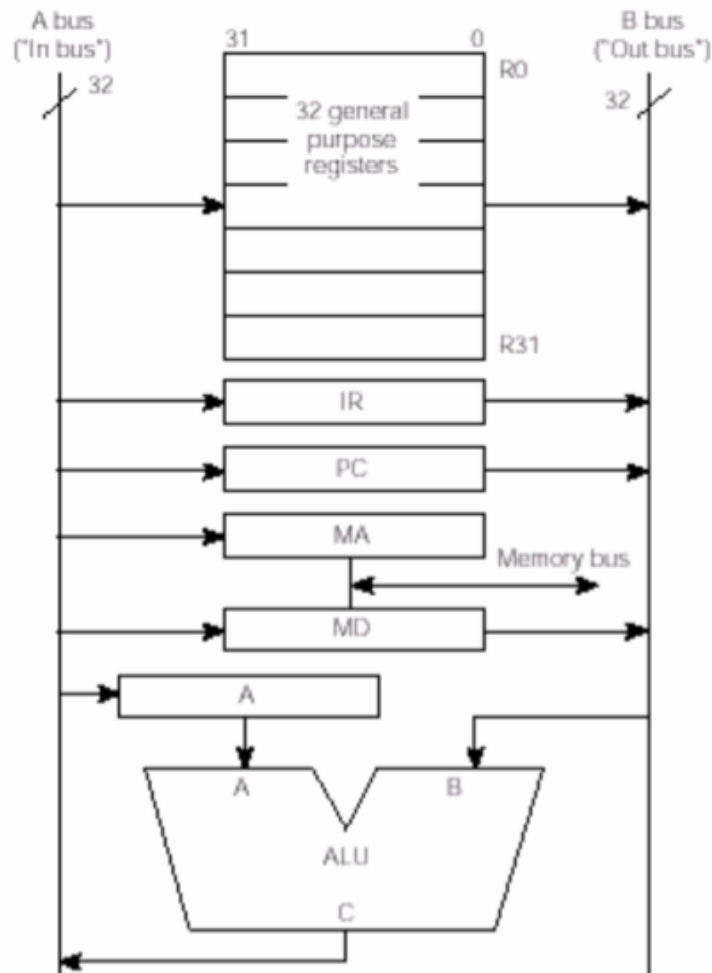
# Mini SRC Datapath Design (Cont'd)

## ° Design Steps in Phase 4:

- Familiarize yourself with the DE0 (DE0-CV) development boards, its I/O pins, display units, clock circuitry, etc.
- Functionally simulate your Mini SRC processor that you have designed in Phases 1 through 4 using a given test program
- Implement and verify the operation of Mini SRC that you have designed and simulated on DE0 development board.
- Submit a detailed Final Report of your CPU Design Project.



# A potential 2-bus Mini SRC



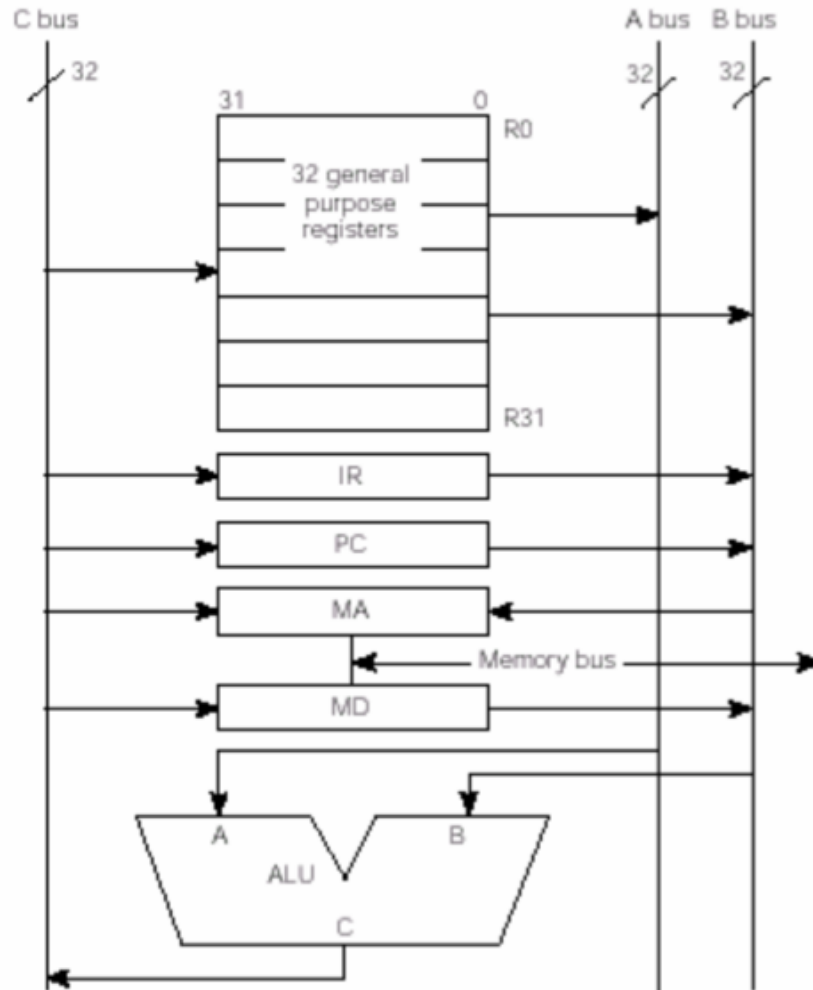
- **Bus A carries data going into registers**
- **Bus B carries data being gated out of registers**
- **ALU function  $C = B$  is used for all simple register transfers**

## The 2-Bus SRC Microarchitecture

Reference: Heuring/Jordan, Ch. 4  
([Lab Reader](#) on the course website)



# A potential 3-bus Mini SRC



- **A-bus is ALU operand 1, B-bus is ALU operand 2, and C-bus is ALU output**
- **Note MA input connected to the B-bus**

## The 3-Bus SRC design

Reference: Heuring/Jordan, Ch. 4  
([Lab Reader](#) on the course website)

# General Guidelines

---

- Identify your Lab Partner in advance (groups cannot be changed later)
- For each phase, collaborate with your Lab Partner
- Divide up the required job for each phase between the group members
- While someone is working on the project with the machine in the Lab, the other group member should work on his/her own laptop and then coordinate things
  
- Read the descriptions of Lab Phases carefully
- Consult the Lab Reader on the course website
- Refer to the Tutorial document on Altera Quartus II
- Consult the Lecture Slides on Arithmetic, VHDL, and Verilog
- Refer to DE0 and DE0-CV User Manuals
- Read the additional reference material on VHDL, Verilog, etc., in the tutorial and on the course website
- Contact TAs and Instructor during Lab hours