Register Transfer and Datapath Structures

COE608: Computer Organization and Architecture

Dr. Gul N. Khan

http://www.ee.ryerson.ca/~gnkhan

Electrical and Computer Engineering

Ryerson University

Overview

- Introduction to Register Transfer
- Design of Datapaths and Processor Control
- Datapath Representation
- Datapath Interconnection Strategies and Register Transfer Operations
  - Point-to-Point Transfer
  - Bus-based Transfer
  - Multiple Busses
- Typical ALU-Datapath Diagrams

Chapter 4 of the text book
Chapter 7 "Logic and Computer Design" by Mano & Kime

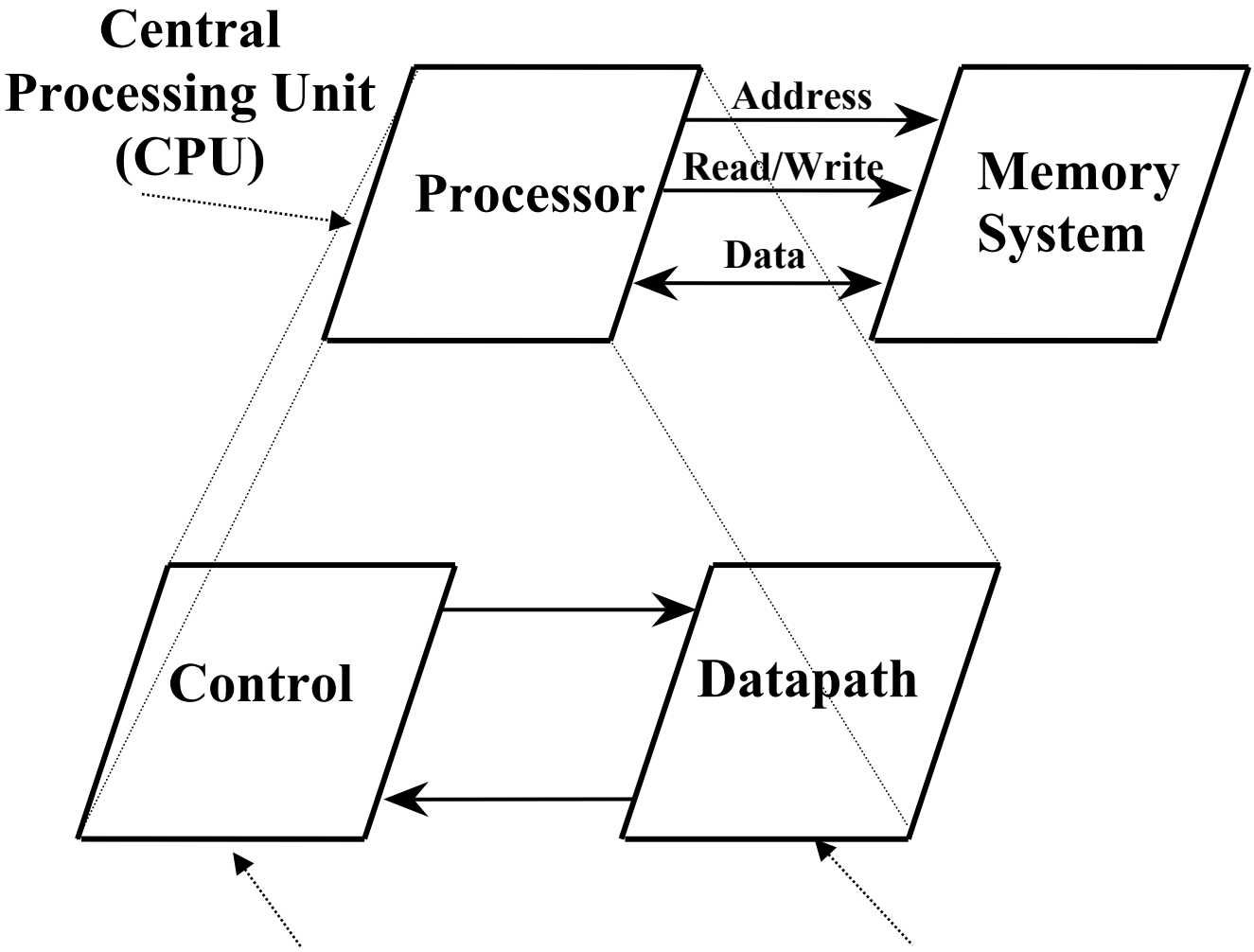
Motivation

Computer Design Example: By Using Combinational and Sequential Logic Circuit Design Procedure.

- **Computer**
- **Processing Unit**
- **Control =** Finite State Machine
  - ♦ Inputs
  - ♦ Outputs
  - ♦ Instruction Interpretation = Instruction Fetch, Decode, Execute
- **Datapath**
  - ♦ Functional Units = ALU, Multipliers, Dividers, etc.
  - ♦ Registers = Program Counter, Shifters, Storage Registers

In addition to registers, datapath contains the digital logic consisting of buses, MUXes, decoders and processing circuits.

Structure of a Computer



Instruction

Instruction fetch & interpretation

Execution Unit

Functional Units and Registers

Instruction Sequencing

Example Instruction: Add R_X to R_Y and place result in R_Z.

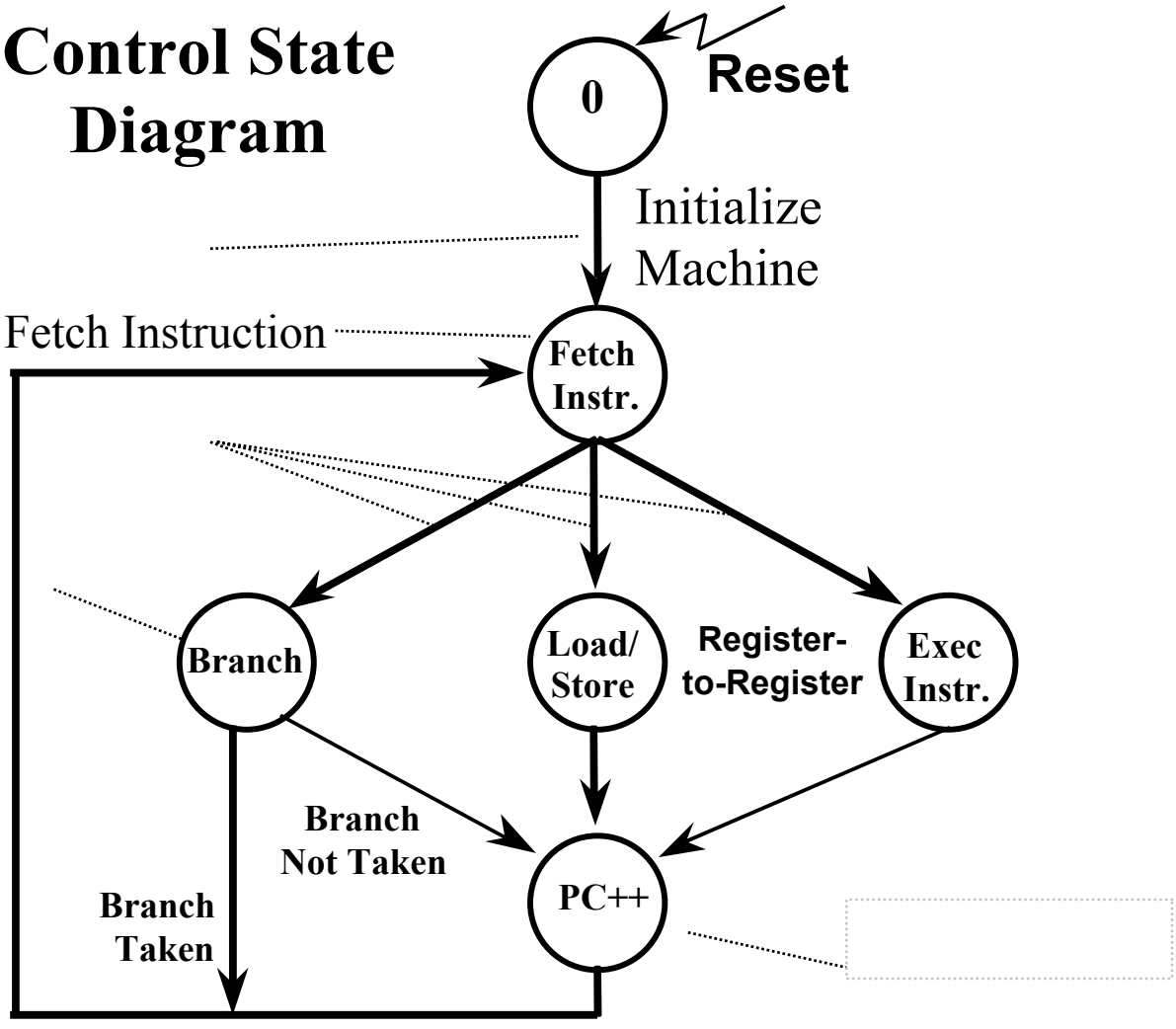
- **Fetch:** Get the Add instruction from memory to Instruction Register (IR)
- **Decode Instruction**
  - Instruction in IR is an ADD
  - Source operands are in R_X, R_Y
  - Destination operand is in R_Z
- **Execute Instruction**
  - Move R_X, R_Y to ALU
  - Set up ALU to perform ADD function
  - ADD R_X to R_Y
  - Move ALU result to R_Z

Instruction Types

- **Data Manipulation**
  Add, Subtract, etc.
- **Data Staging**
  Load/Store data to/from memory
  Register-to-register move
- **Control**
  Conditional/unconditional branches.
  Subroutine calls and returns.

Control Unit

Elements of the Control Unit
Finite State Machine Elements:

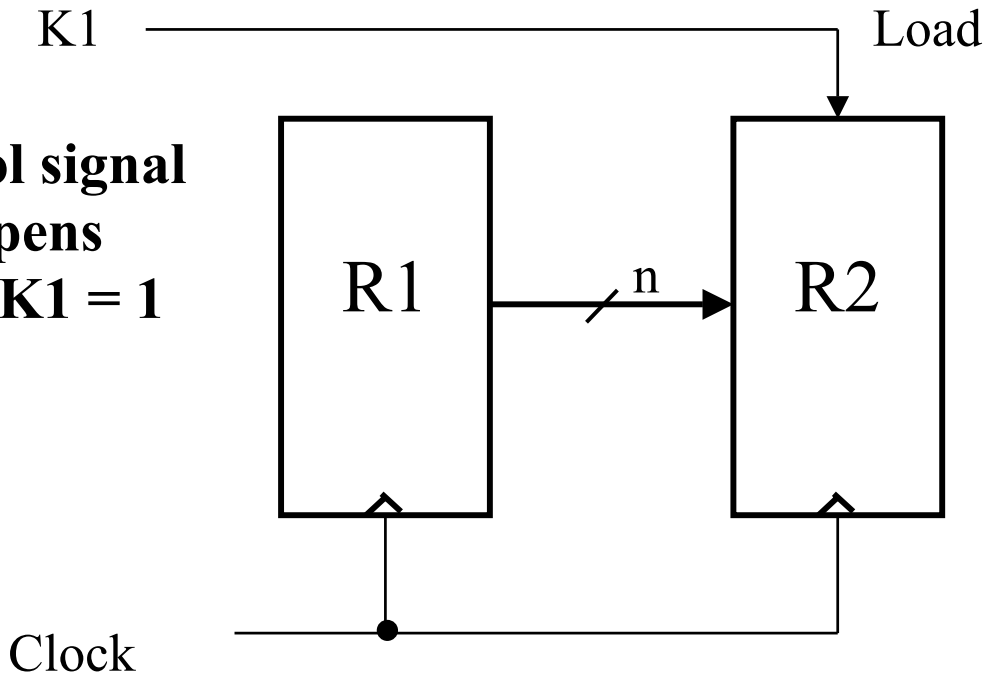


RT: Register Transfer

Data transfer from one register to another is represented in symbolic form as:

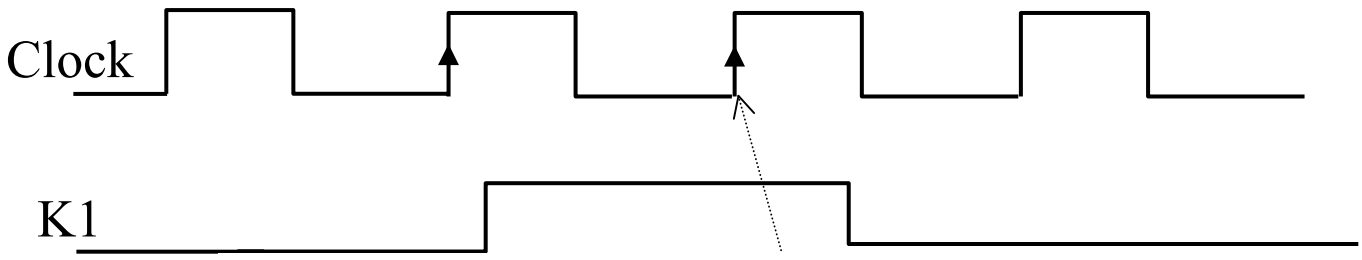
R2 <= R1;

Source Register: R1
Destination Register: R2



K1 is a control signal
Transfer happens
when K1 = 1

If (K1=1) then (R2 <= R1) or K1: R2 <= R1



Register Transfer

Basic Symbols for RT Operations

Symbol	Description	Examples
Letters and numerals	Denotes a register	AR, R2, DR, IR, PC
Parentheses	Denotes a part of the register	R2(1), R2(7:0)
Arrow	Denotes transfer of data	R2 <= R1
Comma	Separate and simultaneous transfers	R2 <= R1, R0 <= R1, R4 <= R3
Square Brackets	Specifies an address for memory	DR <=M[ADR]

Datapath

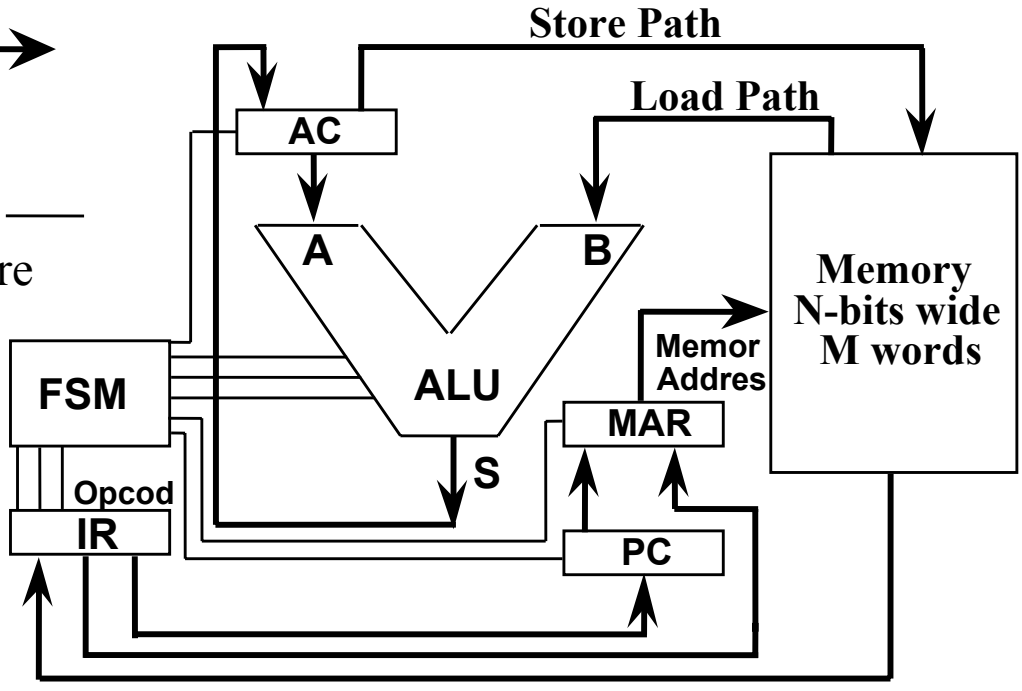
Datapath (Registers + Functional Units) Design

There is a trade-off between datapath complexity and control unit complexity.

Register Transfer View
Single Accumulator Machine

Data Flow →
Represented by arrowhead lines

Control Flow —
Control signals are represented by thin lines.



AC <= AC <op> Mem

MAR *Memory Address Register*

RT: Register Transfer

Trace an Instruction

AC <= AC + Mem<address>

1) Instruction Fetch

- Move PC to MAR
- Initiate a memory read sequence
- Move data from memory to IR

2) Instruction Decode

- Op code bits of IR are input to control FSM
- Rest of the IR bits encode the operand address

3) Operand Fetch

- Move operand address from IR to MAR
- Initiate a memory read sequence

4) Instruction Execute

- Data available on load path
- Move data to ALU input
- Configure ALU to perform ADD operation
- Move S result to AC

5) House-keeping

- Increment PC to point at next instruction

Register Transfer

Control Unit generates control signals

- to transfer data from one register to another.

Register Transfer and Micro-Operations

Instruction Fetch:

MAR <= PC ;
Memory Read;
IR <= Memory;

Instruction Decode:

IF IR<op code> = ADD_FROM_MEMORY THEN

Instruction Execution:

MAR <= IR<addr>;
Memory Read;
ALU B <= Memory;
ALU A <= AC ;
ALU ADD;
AC <= ALU S;

Assert Control Signal for

PC <= PC+1;

Micro-Operations

Micro-operations are the elementry operations performed on Registers.

- Usually performed in one clock cycle.
- Multiple (independent) micro-operations can be performed in one clock cycle.
- One micro-operation for each control point.

✓Control units provide control signals to sequence micro-operations in a prescribed manner.

✓RT operations can be decomposed into one or more micro-operations.

Register Transfer	***Micro-operations***
$AC_1 <= AC_1 + AC_2$	
PC <= 0	$0 \rightarrow PC$
PC <= PC+1	$PC+1 \rightarrow PC$
Address Bus <= MAR	$MAR \rightarrow$ Address Bus

Datapath Interconnection Strategies

Tradeoffs between datapath/control complexity and amount of parallelism supported by hardware.

How to interconnect hardware resources of the datapath?

Case Study:

Consider 4 general-purpose registers that must be able to exchange their contents.

- Methods of interconnection.
- Datapath support for register-to-register swap.

RT notation for swap instruction

SWAP(Ri, Rj)

Register-to-Register Communication

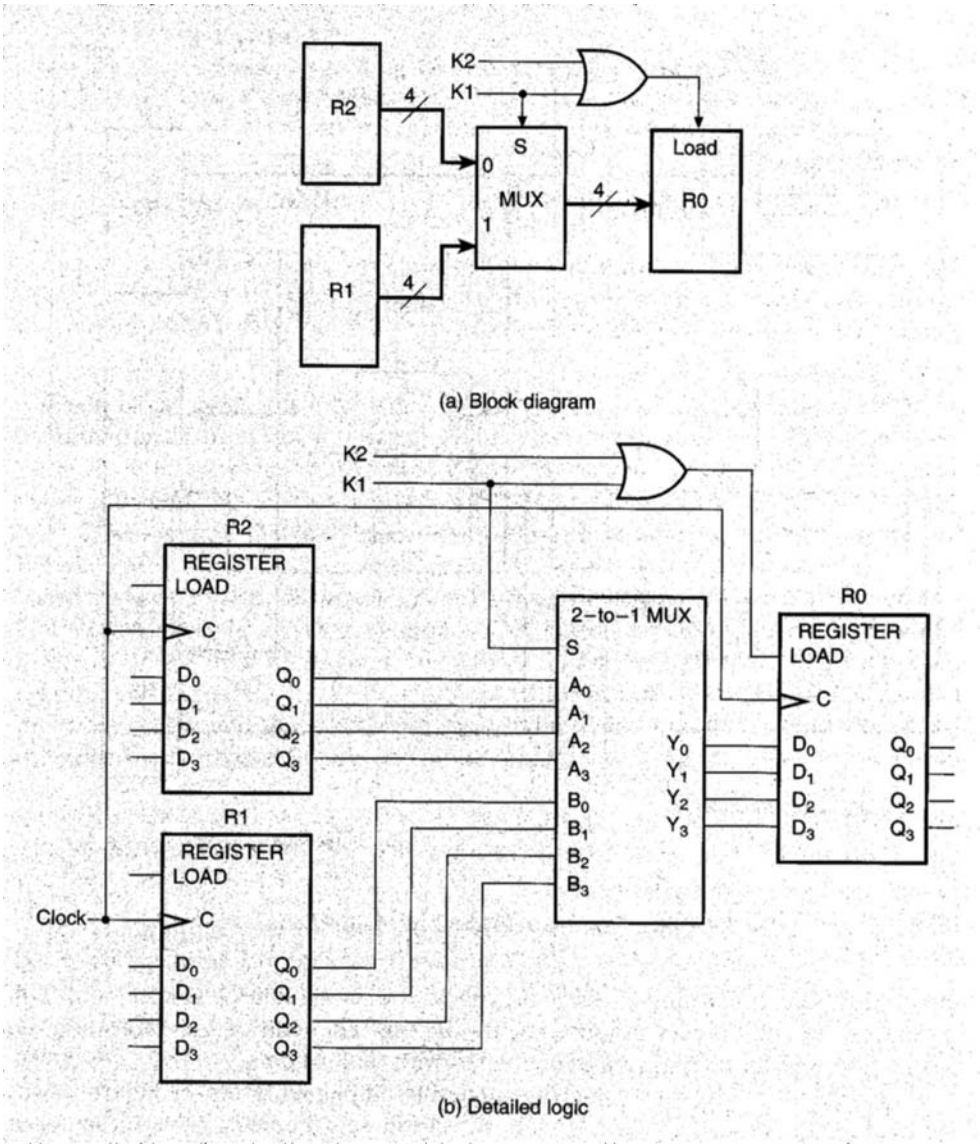
Point-to-Point Connection

MUX Based Data Transfer

If (K1 = 1) then (R0 <= R1) **else**
If (K2 = 1) then (R0 <= R2)

K1 and K2 control signals that control RT as:

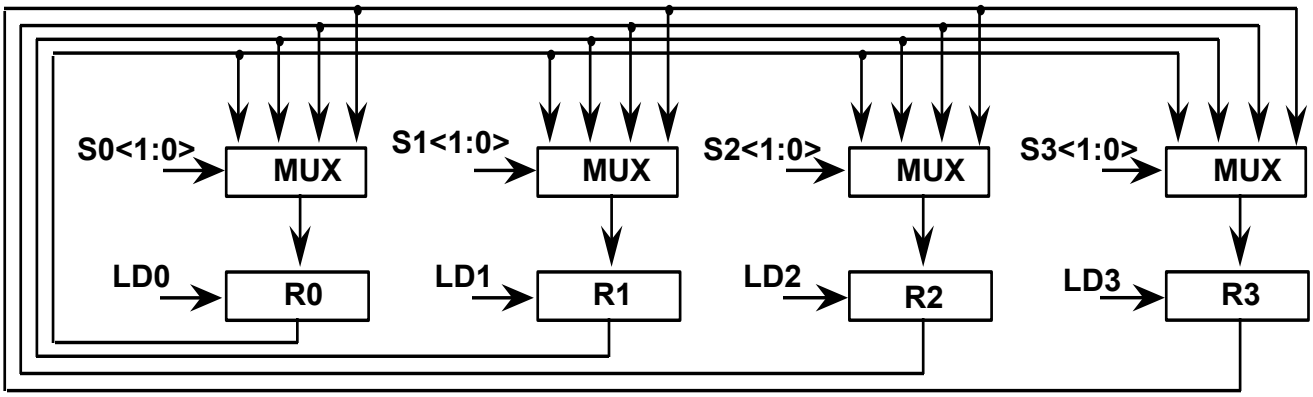
K1: R0 <= R1, K1K2: R0 <= R2



- Dedicated paths need large amount of logic.
- Excessive number of interconnections.

Point-to-Point Connection

Four registers interconnected via 4:1 MUXes and point-to-point connections



Edge-triggered N-bit registers controlled by LDi control signals

N x 4:1 Multiplexers per register that are controlled by Si<1:0> control signals.

Control of Register SWAP Operation

SWAP(R1, R2) Control Signals

01 → S2<1:0>;

10 → S1<1:0>;

1 → LD2;

1 → LD1;

Point-to-Point Connections

Multiple Register Transfers:

R0 <= R1 and R3 <= R2

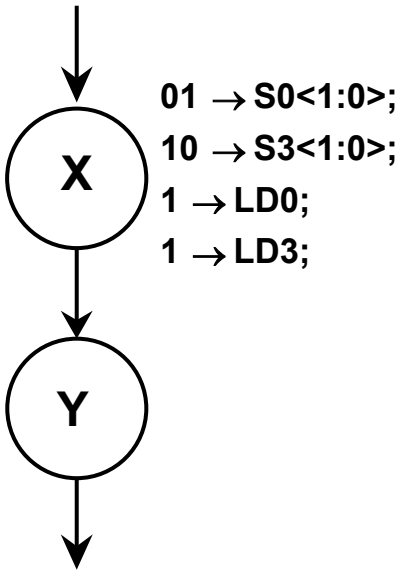
01 → S0<1:0>;

10 → S3<1:0>;

1 → LD0;

1 → LD3;

Control Signals Assertion



Enter state X:

On entering state Y:

Advantages and Disadvantages

Bus-Based Transfer

Bus is a set of common lines driven by selection logic.

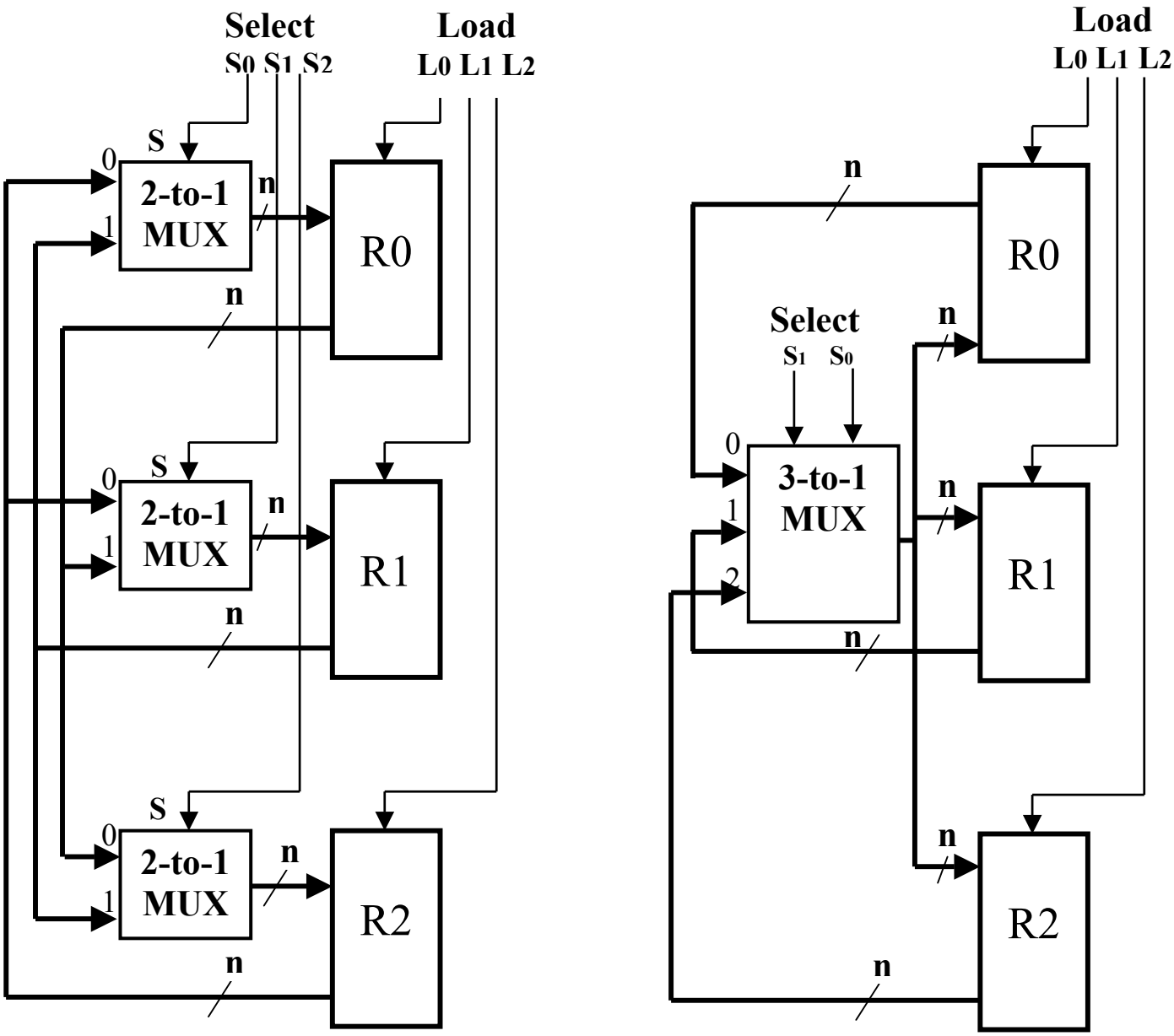
- Bus-based transfer is an efficient scheme that uses shared transfer path.
- Control signals select a single source and one or multiple destinations on a clock cycle for which the transfer takes place.

Transfers between 3-Registers

Dedicated Multiplexers

Single Bus System

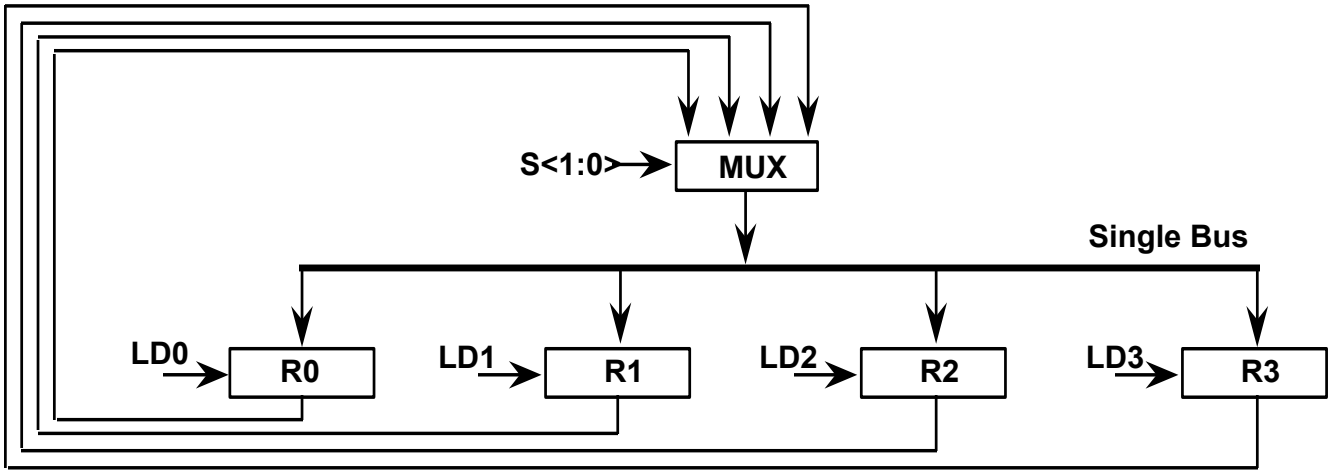
MUX and Bus Transfer



Single Bus Register Transfers

Register Transfer	CONTROL SIGNALS				
	Select		Load		
	S_1	S_0	L_2	L_1	L_0
R0 <= R2					
R0 <= R1, R2 <= R1					
R0 <= R1, R1 <= R0					

Single Bus Interconnection



- Register MUX blocks replaced by a single block.
- 25% hardware cost of the previous alternative.
- Shared set of pathways is called a BUS.

RT Operations R0 <= R1 and R3 <= R2

State X: (R0 <= R1)
01 → S<1:0>;
1 → LD0;

(R3 <= R2)
State Y: 10 → S<1:0>;
1 → LD3;

Datapath no longer supports two simultaneous transfers!
Two control states are required for transfers.

Single Bus Interconnection

SWAP Operation

- A special TEMP register must be introduced ("Register 4")

State X: **(R4 <= R1)**

Control Signals

Three states are required rather than one! plus extra register and wider MUX

More control states because this datapath supports less parallel activity

State Y: **(R1 <= R2)**

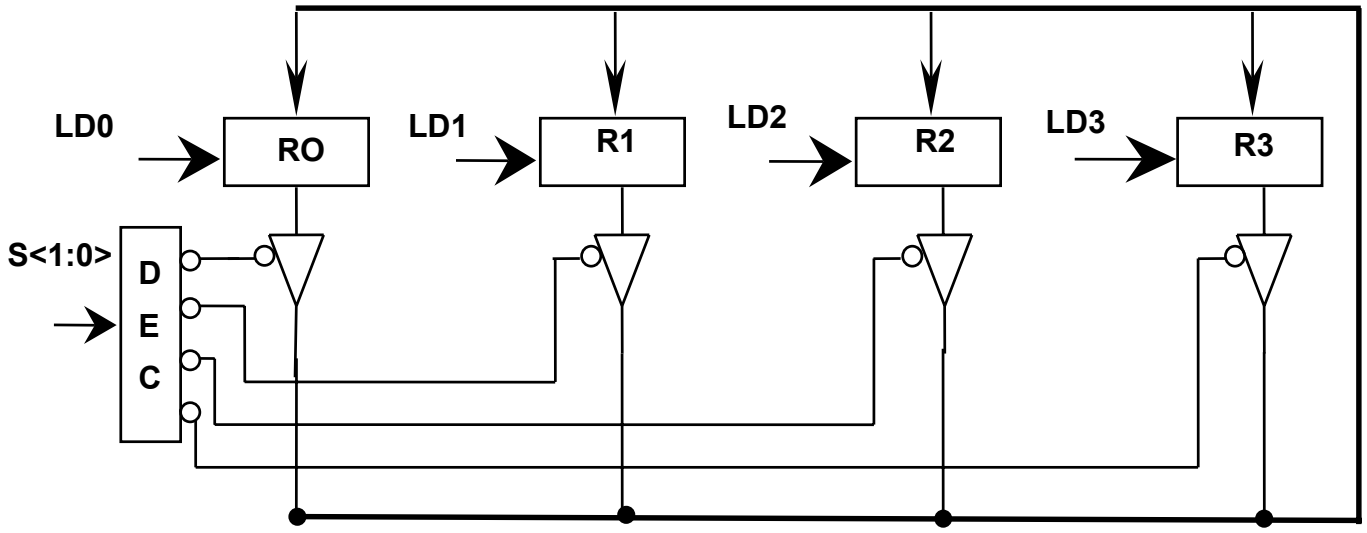
Control Signals

Engineering choices made based on how frequently multiple transfers take place at the same time

State Z: **(R2 <= R4)**

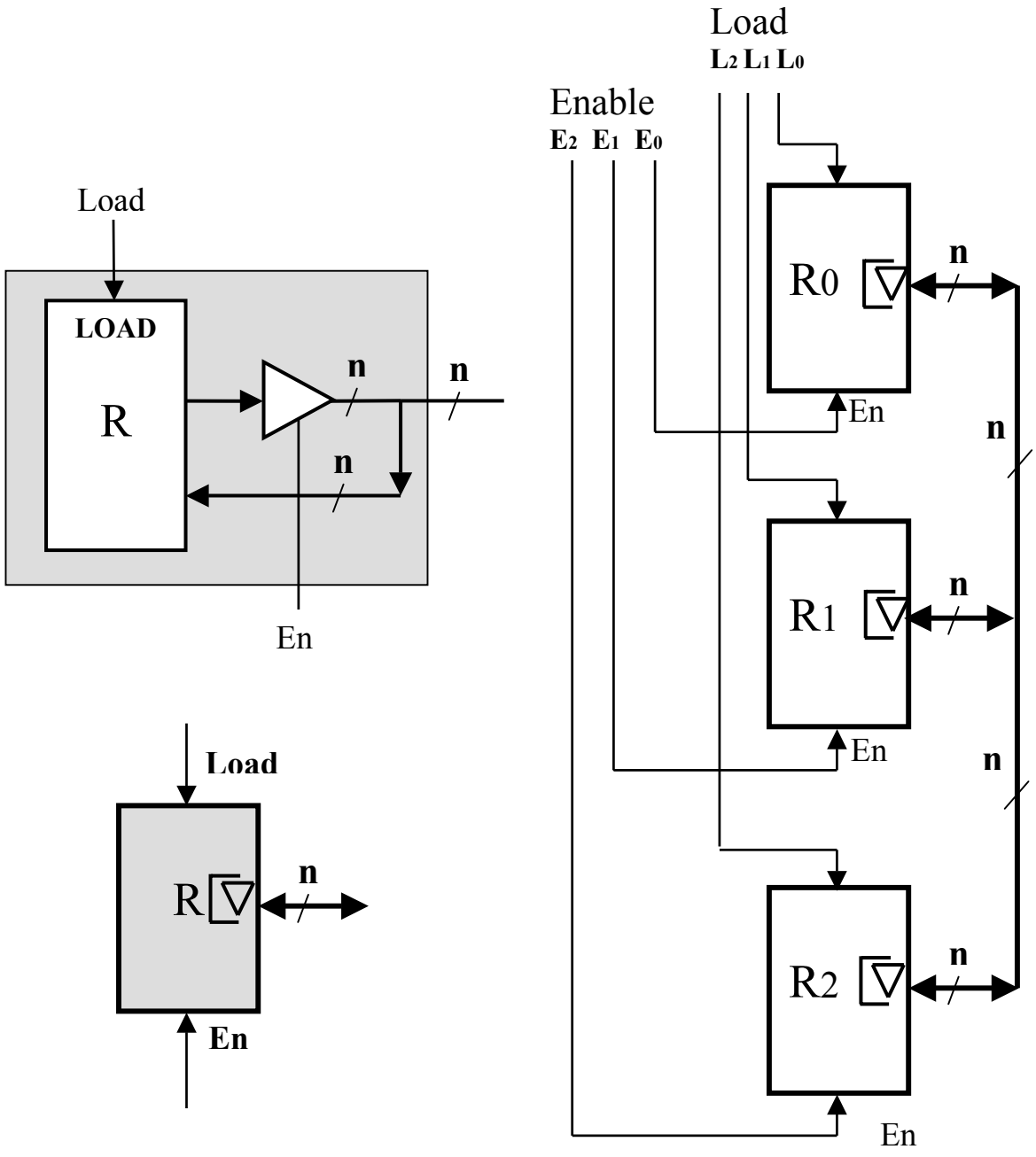
Alternatives to Multiplexers

Tri-state buffers as an interconnection scheme



Decoder decodes the input control lines S<1:0> and generates one select signal to enable only one tri-state buffer.

Three-State Bus



Three-state bus using registers with bi-directional lines

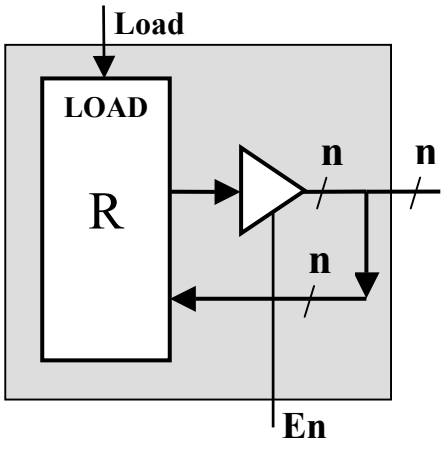
Three-State Bus

Three-state buffer based bus has the potential of reduction in the number of connections.

Many 3-state buffer outputs can be connected together to form one-bit line of a bus.

Main Advantages:

- 3-state based bus has one level of logic gates as compared to MUX that may require multiple-level OR gates due to limited fan-in.
- 3-state buffers provide fast buses with multiple sources.
- Reduced (half) data connections to the set of register blocks for each bit of the bus.



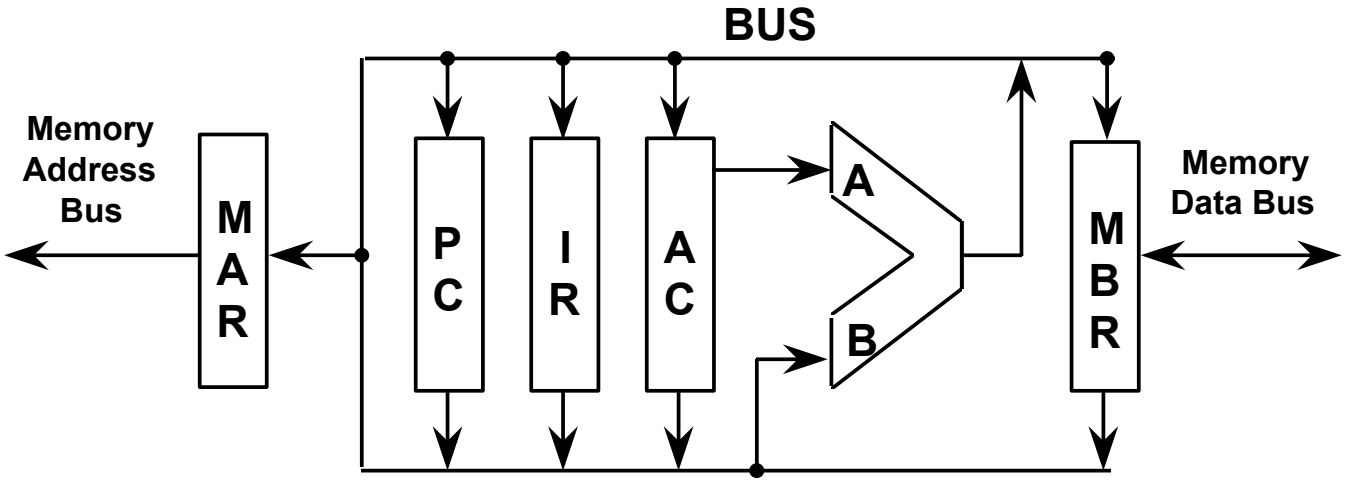
3-state buses are suitable for interconnection between different physical packages.

Bus-based Interconnection

Real datapaths are a compromise between extremes.

Register Transfer Diagram

Single Bus Design



Register Transfer Operations

BUS <= PC
BUS <= IR
BUS <= AC
BUS <= MBR
BUS <= ALU Result

PC <= BUS
IR <= BUS
AC <= BUS
MBR <= BUS
ALU B <= BUS
MAR <= BUS

ALU <= AC
("hardwired")

Single Bus

Example:

Register Transfer for Single Bus Design

Instruction Interpretation for "ADD Mem[X]"

Fetch Operand

Cycle 1: BUS <= IR<operand address> ;
MAR <= BUS ;

Cycle 2: Memory Read;
MBR <= Databus ;

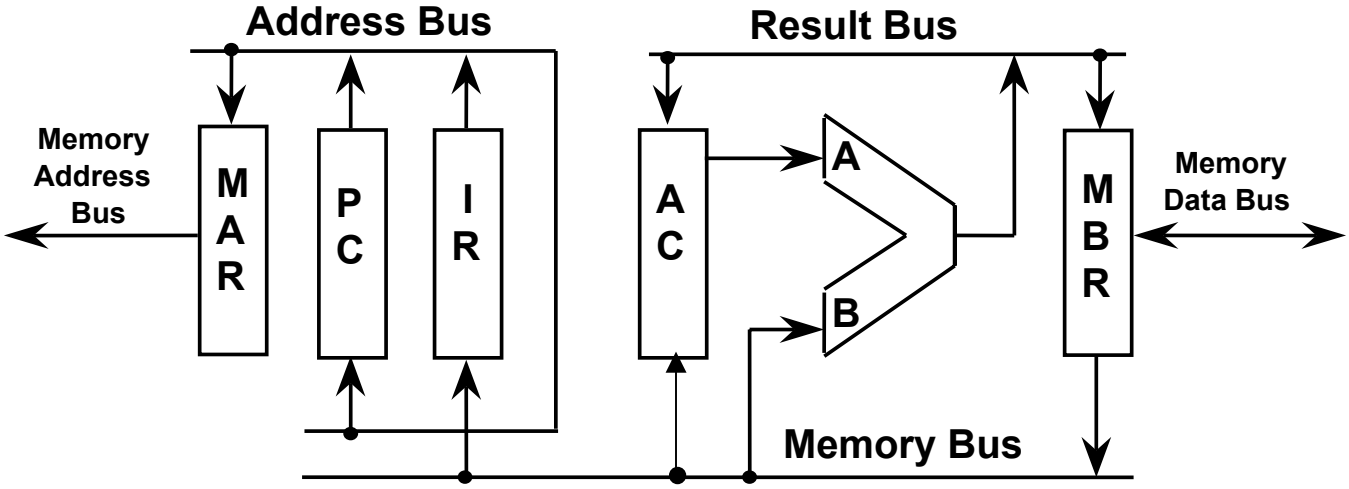
Perform ADD

Cycle 3: BUS <= MBR ;
ALU B <= BUS ;
ALU A <= AC ;
ADD;

Write Result BUS <= ALU Result ;
AC <= BUS ;

Multiple Buses

Three Bus Design - Supports more Parallelism



Single bus replaced by three buses:

- MBUS: Memory Bus
- RBUS: Result Bus
- ABUS: Address Bus

Instruction Interpretation for "ADD Mem[X]"

Fetch Operand

Cycle 1:

ABUS <= IR<operand address>;
MAR <= ABUS;

Cycle 2:

Memory Read;
MBR <= Databus;

Perform ADD

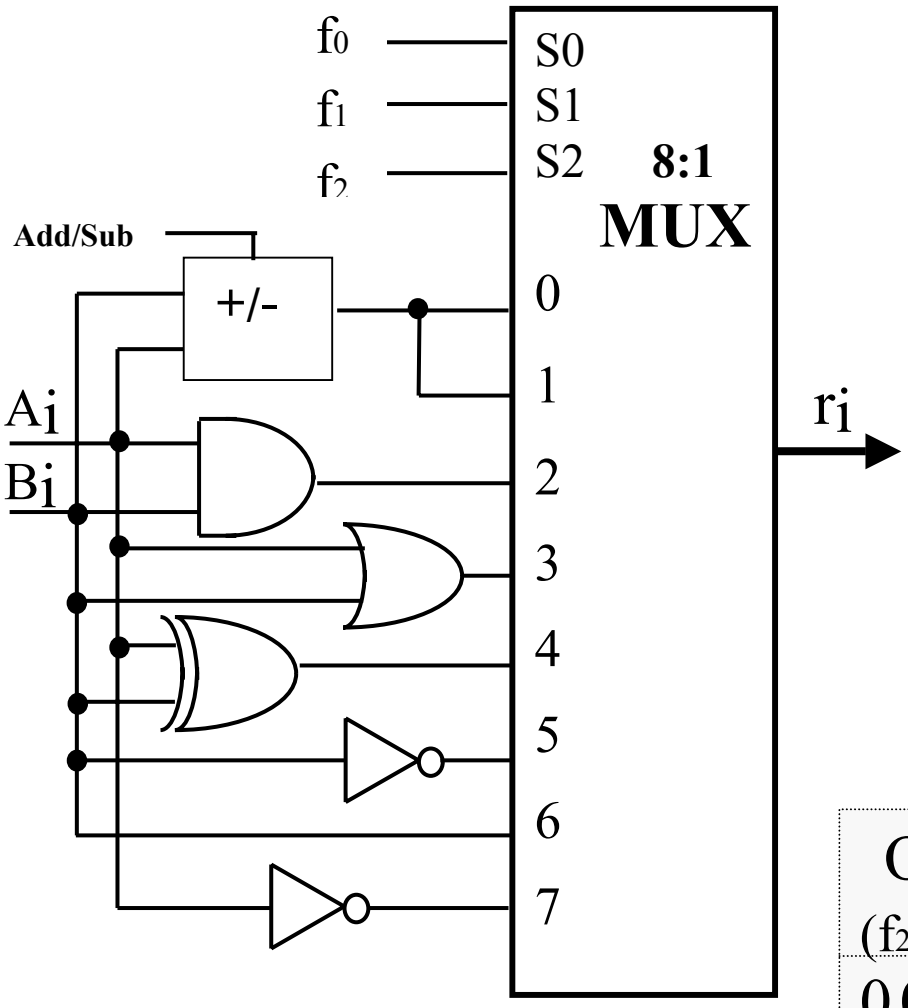
Cycle 3:

MBUS <= MBR;
ALU B <= MBUS;
ALU A <= AC;
ADD;

Write Result

RBUS <= ALU Result;
AC <= RBUS ;

ALU



Add/Sub = f_0

Op (f_2 f_1 f_0)	Result, r_i
0 0 0	$A_i + B_i$
0 0 1	$A_i - B_i$
0 1 0	$A_i \cdot B_i$
0 1 1	$A_i \parallel B_i$
1 0 0	$A_i \oplus B_i$
1 0 1	$\overline{A_i}$
1 1 0	A_i
1 1 1	$\overline{B_i}$

32-bit ALU (VHDL Code)

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity alu is
  port(
       A,B  : in  std_logic_vector(31 downto 0);
       OP   : in  std_logic_vector(2 downto 0);
            Ri    : out  std_logic_vector(31 downto 0));
  end alu;
architecture alu of alu is
       signal tmp : std_logic_vector(31 downto 0);
begin
process(OP, A, B)
begin
 case OP is
   when "000" => tmp <= A + B;
   when "001" => tmp <= A - B;
   when "010" => tmp <= A and B;
   when "011" => tmp(31 downto 0)  <= A or B ;
   when "100" => tmp <= A xor B ;
   when "101" => tmp <= not A ;
   when "110" => tmp <=  A ;
   when "111" => tmp <= not B ;
   when others => tmp <= "00000000000000000000000000000000";
 end case;
 Ri <= tmp ;
end process;
end alu;
```

Datapath

Block diagram of a datapath containing registers, buses, MUXes, decoders and processing units.