Central Processing Unit Designs

Instructor: Saraju P. Mohanty

- CPU basics
- Design of a Complex Instruction Set Computer
- Design of a Reduced Instruction Set Computer
- High-Performance CPU concepts

Sources

- Logic and Computer Design Fundamentals by M. M. Mano and C. R. Kime.
- Computer Organization and Design: The Hardware/Software Interface by David A. Patterson and John L. Hennessy.
- Dr. Valavanis lectures

Central Processing Unit Design: Introduction

- CPU is the key component of a digital computer.
- CPU purpose:
  - Decode instructions received from memory
  - Perform operations such as, transfer, arithmetic, logic, etc.
  - Provides one or more buses for transferring instructions, data, and other information to and from the components connected to it.
- Two CPU designs will be discussed:
  - One for complex instruction set computer (CISC)
  - Other for reduced instruction set computer (RISC)

Central Processing Unit Design: Introduction

CPU is divided into two: datapath and control

Datapath: function unit, registers and internal buses that provide pathways for the transfer of information between the registers, the function unit, and other computer components.

Control Unit: has a program counter, instruction register, and control logic.

Datapath may or may not be pipelined.

Control unit may be hardwired or microprogrammed.

The Two designs:

A CPU for a CISC using a non-pipelined datapath with a microprogrammed control unit.

A CPU for a RISC using a pipelined datapath with a hardwired pipelined control unit.

Complex Instruction Set Computer (CISC) Design

Features

Non-pipelined datapath

Microprogrammed control unit.

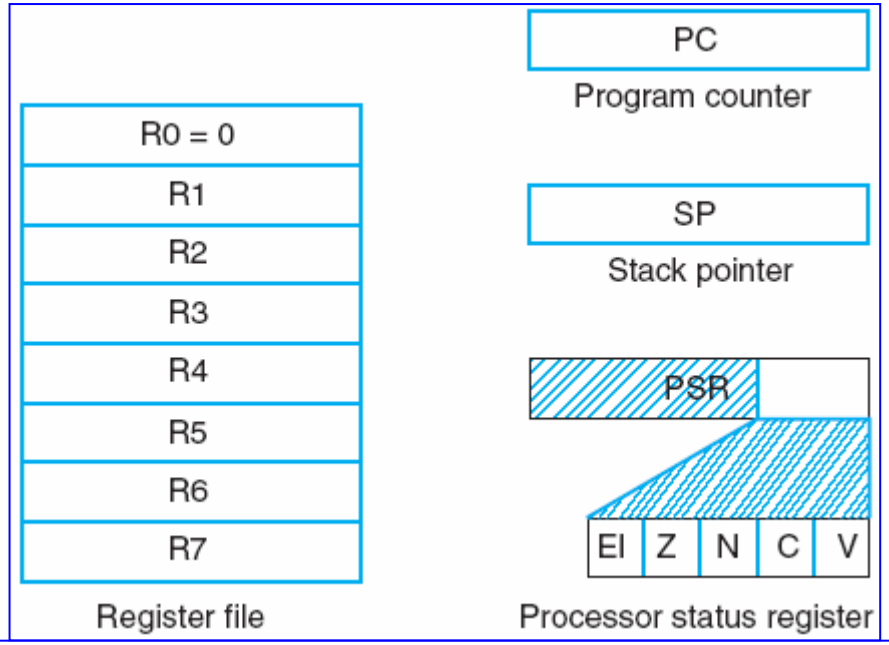
The CISC nature of the instruction set architecture is demonstrated by its memory-to-memory access for data manipulation instructions.

No of addressing modes: eight

Two instruction format lengths

Instructions that require significant sequences of operations for their execution.

CISC Design: Instruction Set Architecture

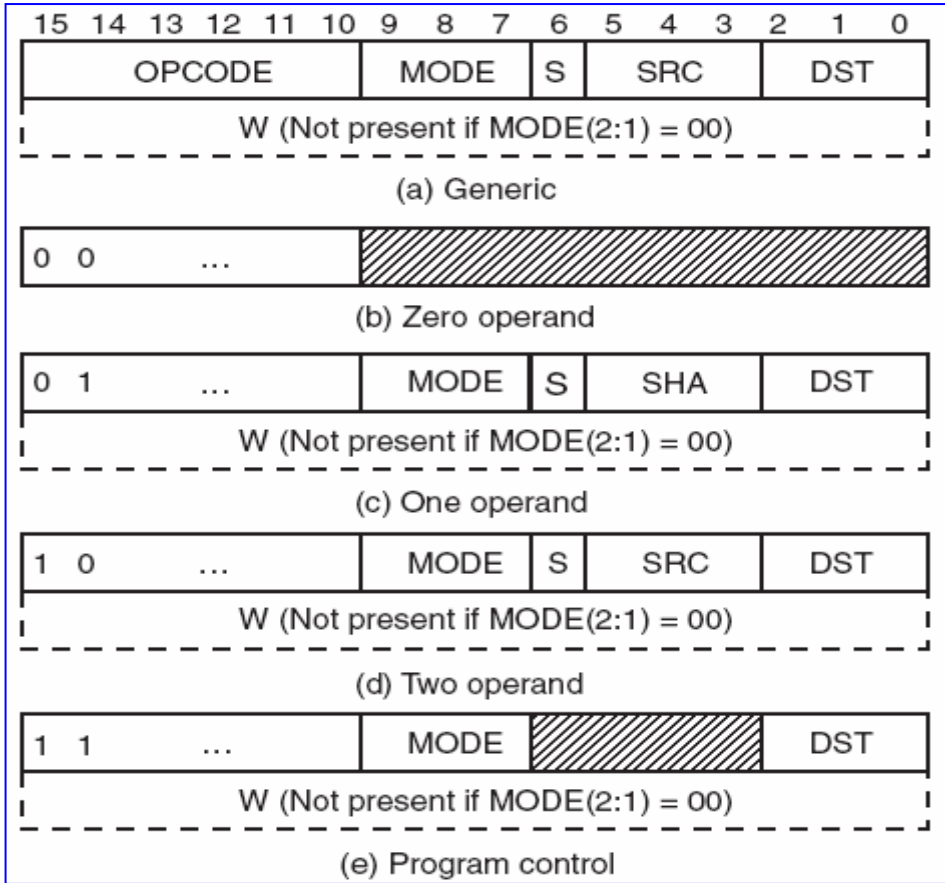


Eight 16-bit registers (R0 – R7).

- R0 always supplies the value zero when it is used as a source and discards the result when it is used as a destination.
- Also has a program counter PC and a stack pointer SP (memory stack is a part of the architecture).

The Process Status Register (PSR) contains information only in the rightmost 5 bits, other bits being zero. (EI is interrupt enable bit).

CISC Design: Instruction Formats



- Format has 5 fields.
- OPCODE specifies operation
- MODE and S determine address of the operands;
- SRC and DST are the 3-bit source /destination register address fields.
- Optional second word W that appears with some instructions as an operand or an address.

NOTE: For shifts, SRC becomes SHA.

- First 2 bits of MODE field specify 4 types of addressing modes : register, immediate, indexed, and relative to the PC.
- The third bit of MODE field specifies whether the address generated is direct (0) or indirect (1).

CISC Design: Datapath Organization

- Require more temporary storage for complex instructions spanning many clock cycles and performing complicated operations.
- Register file uses 16: – R0 to R7 are visible to the programmer, while R8 to R15 are used for temporary storage - hidden from the programmer.
- Standard locations are used for storing the operands and addresses used by execution microcode for almost all (most) instructions.
- R12 – Source Address (SA)
- R13 – Source Data (SD)
- R14 – Destination Address (DA)
- R15 – Destination Data (DD)

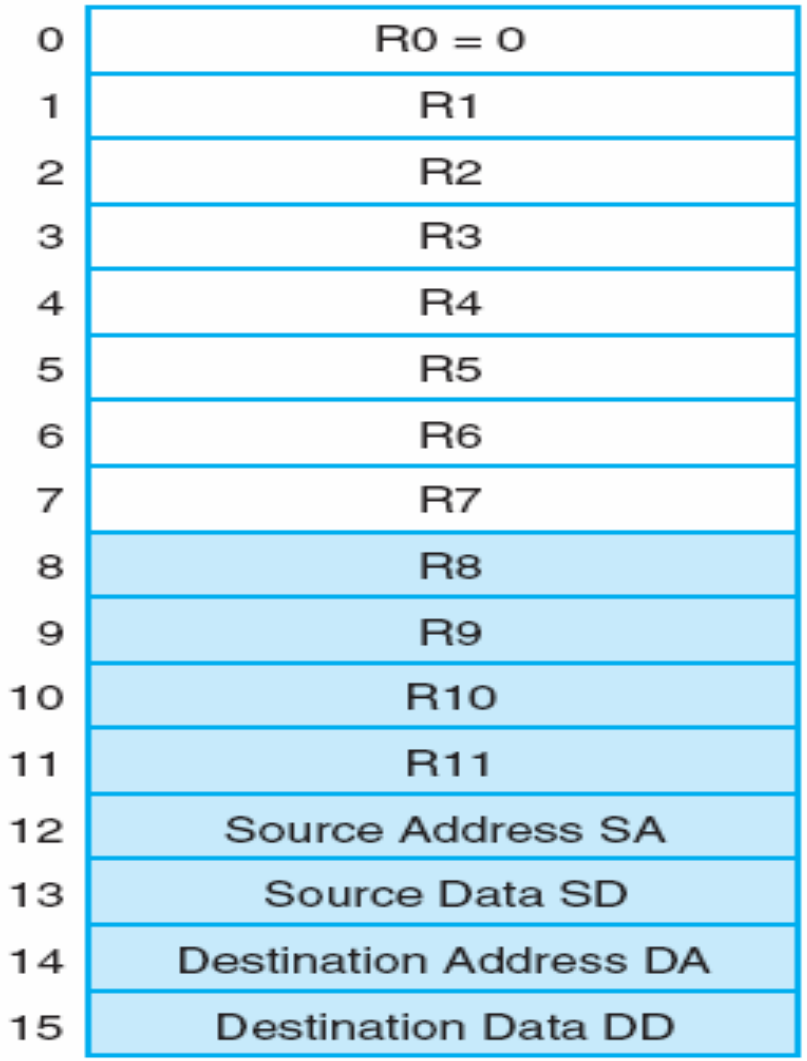
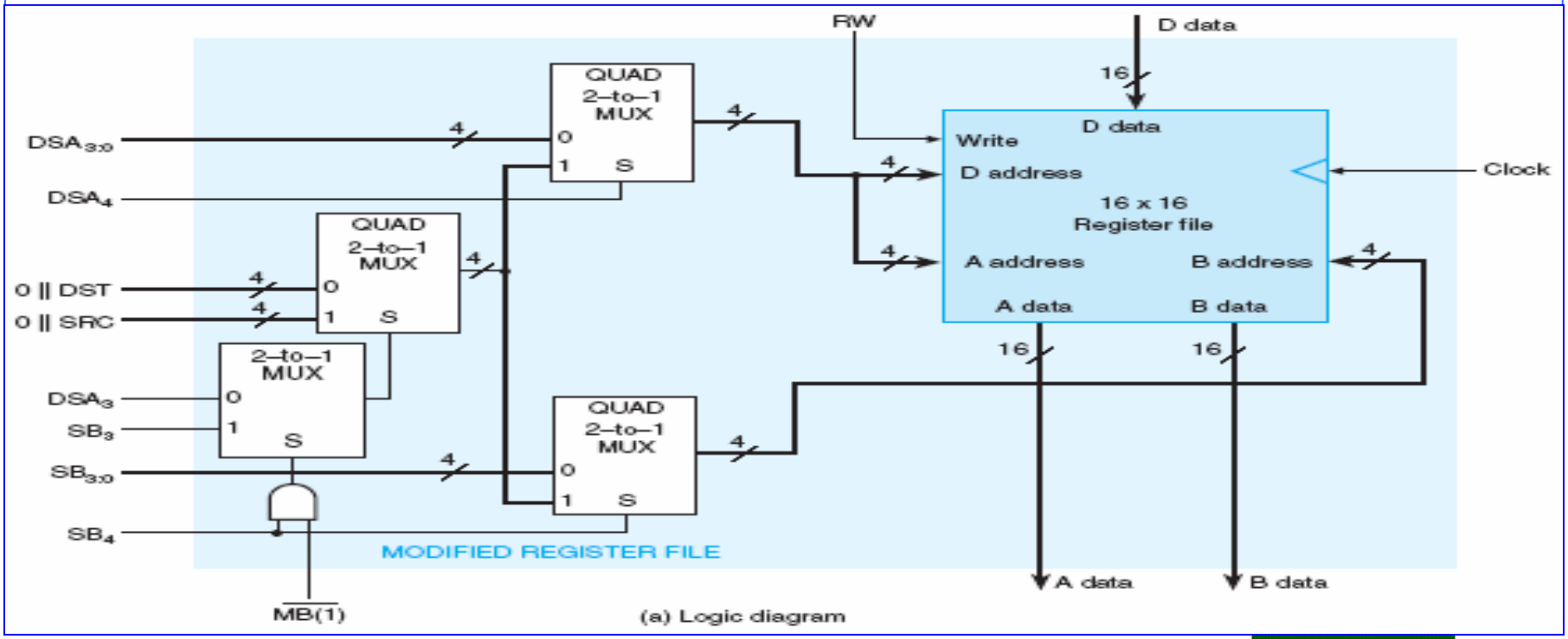


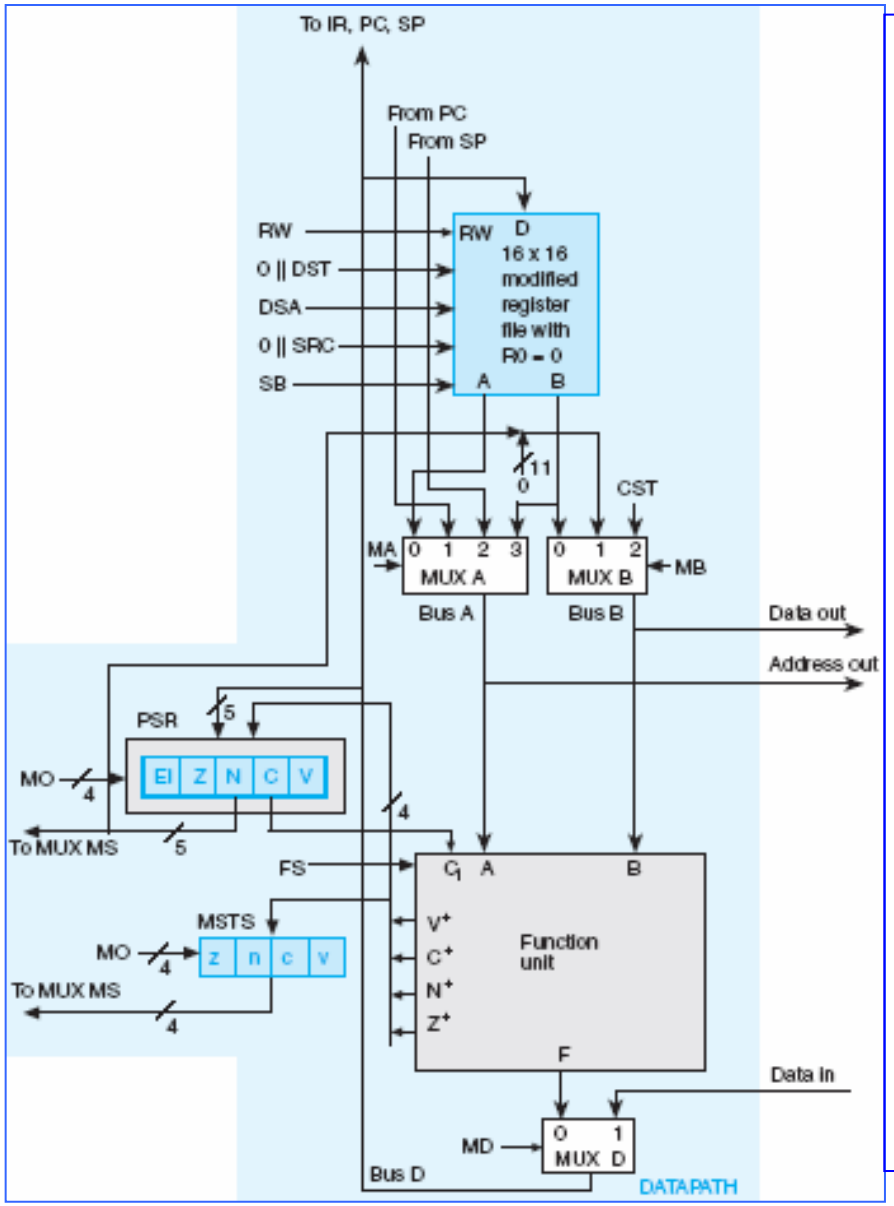
Fig. 10-3 Register File Map

CISC Design: Datapath Organization …..

- Cannot access the eight temporary registers based on the 3-bit register addresses available in the instruction.
- To deal with this problem,
  - first, 4-bit register addresses are provided from the microinstruction,
  - second, a microinstruction bit to choose between these addresses and those from the instruction.
- So, register file is modified



CISC Design: Datapath Organization



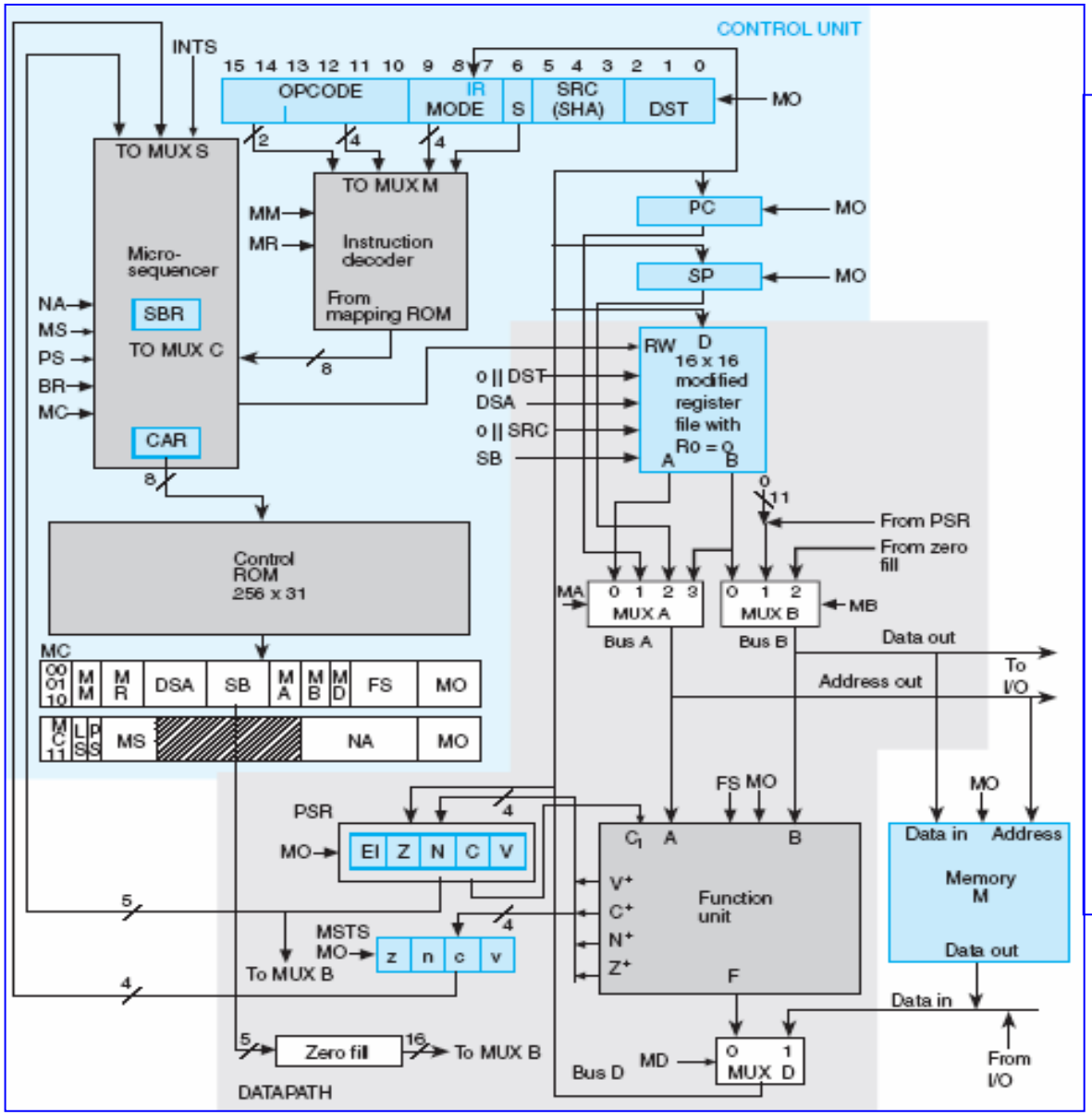
- It is necessary to load and store the contents of PC, SP and PSR.
- So, paths are established to and from the datapath buses.
- Five flip-flops (PSR) store program status bits EI, Z, N, C, and V.
- Another set of five flip-flop register (MSTS) is provided for temporary values of EI, Z, N, C, and V.
- MSTS is used by the microprogram routines to get the values of PSR without disturbing it.

CISC Design: Microprogrammed Control

Control consists of 4 principal parts:

- Control ROM – has 256 words of 31 bits each
- Three control registers – Instruction Register (IR), program counter (PC), and the stack pointer (SP).
- MicroSequencer – provides sequencing within the control unit. Contains 2 registers – control address register CAR, subroutine branch register, SBR.
- Instruction Decoder – consists of combinational logic and is also a next address source for the CAR.

CISC Design: CPU



There are 2 formats

Format A (MC is 00, 01, or 10) – perform data transfers and manipulation and decode instructions, also handle returns from a microsubroutine.

Format B (MC is 11) – change the flow of the microprogram by implementing branches and a microsubroutine call.

Reduced Instruction Set Computer (RISC) Design

Features

- Instruction set architecture is characterized by load/store memory access
- Four addressing modes
- Pipelined datapath and control unit.
- A single instruction format length, and instructions that require only elementary operations.
- Due to data and control hazards associated with pipelined designs, modifications are made to both the control unit and datapath.

RISC Design: Instruction Set Architecture

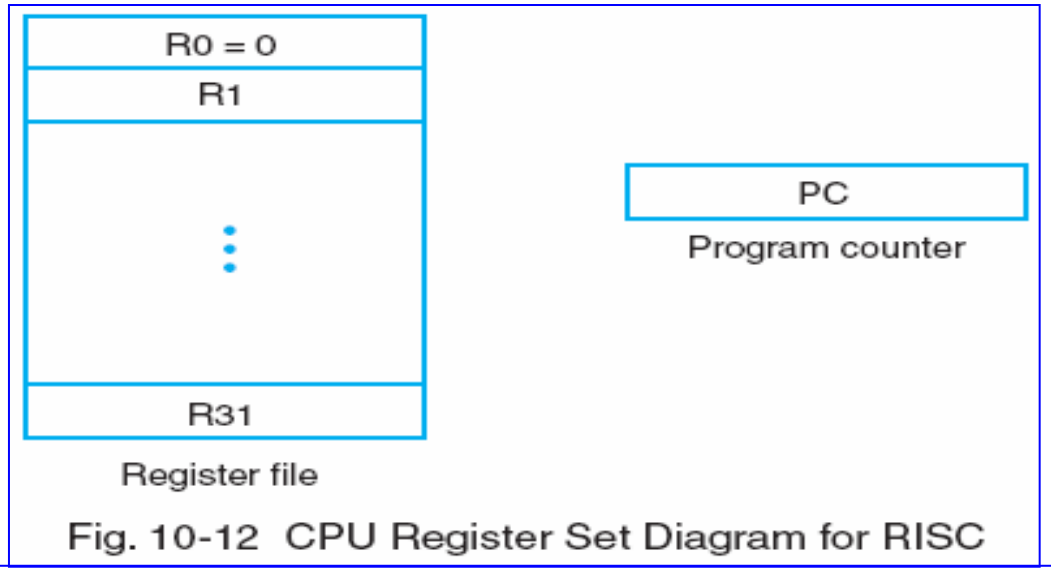


Fig. 10-12 CPU Register Set Diagram for RISC

- 32 registers, each 32 bits long.
- Register R0 always supplies the value zero when it is used as a source and discards the result when it is used as a destination.
- Size of the programmer-accessible register file is larger in the RISC than in the CISC because of the load/store instruction set architecture.
- No SP or PSR are provided, SP-based or PSR-based operations are implemented by sequences of instructions using register file.

RISC Design: CPU Instruction Formats

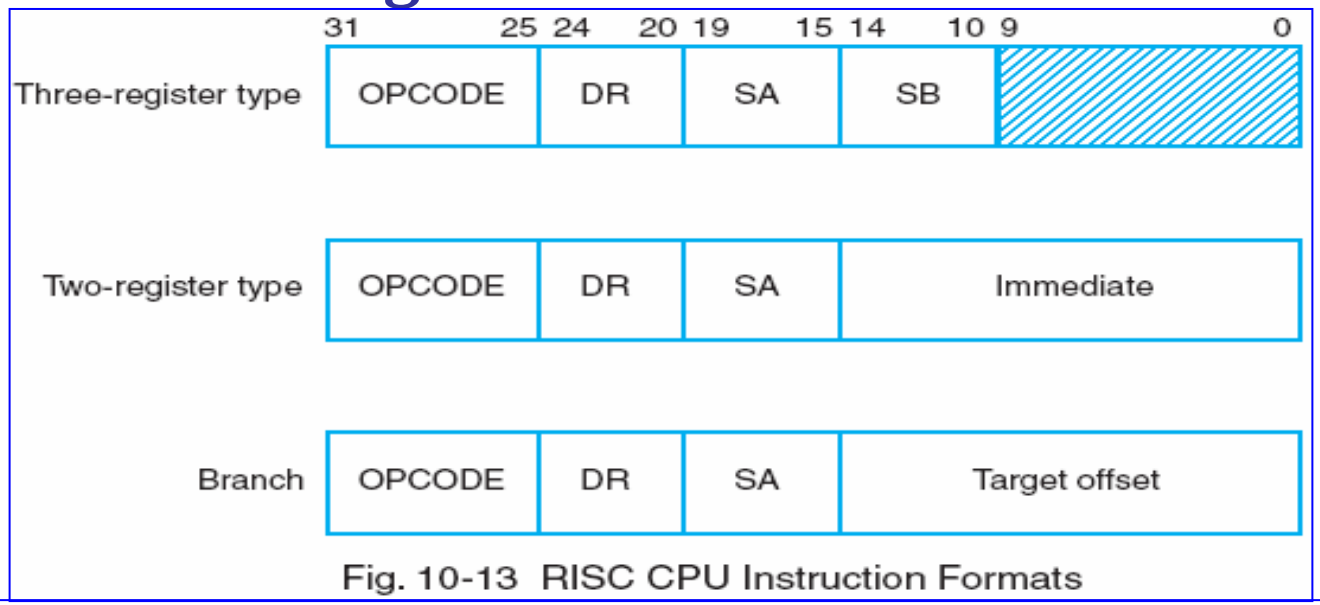


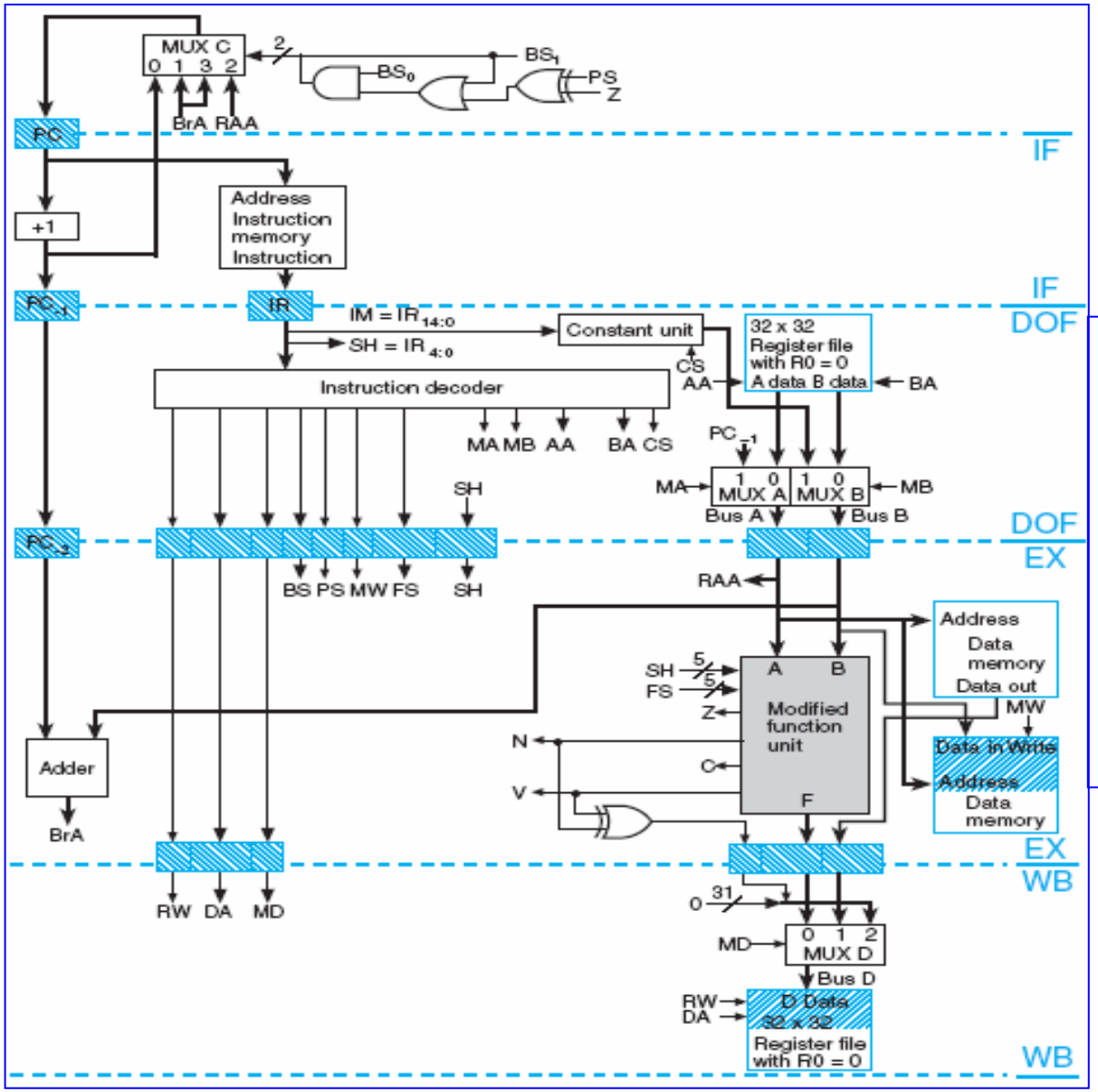
Fig. 10-13 RISC CPU Instruction Formats

- First format specifies 3 registers – two registers addressed by the 5-bit source register fields SA and SB contain two operands. Third register, addressed by a 5-bit destination register field DR, specifies the result location. 7-bit opcode provides for a maximum of 128 operations.
- Other 2 formats replace the second register with a 15-bit constant.
- In Two-register format, the constant acts as an immediate operand.
- In Branch format, the constant is used as a target offset.

RISC Design: Addressing Modes

- 4 addressing modes: Register, Register Indirect, Immediate, and Relative.
- The mode is specified by the operation code, rather than by a separate Mode field.
- The three-operand data manipulation instructions use register mode addressing.
- Register indirect applies only to the load and store instructions, the only instructions that access data memory.
- Instructions using the two-register format have an immediate value that replaces register address SB.
- Relative addressing applies exclusively to branch and jump instructions and so produces addresses only for the instruction memory.

RISC Design: Pipelined RISC CPU



- The zero fill is replaced by constant unit.
- Constant unit fill zero for CS=0 and sign extension for CS =1.

Pipelining RISC Design: Data Hazards

- Hazards are timing problems that arise because the execution of an operation in a pipeline is delayed by one or more clock cycles from the time at which the instruction containing the operation was fetched.
- If a subsequent instruction tries to use the result of the operation as an operand before the result is available, it uses the old or stale value.
- Remedy – Have the compiler or programmer generate the machine code to delay instructions so that new values are available.
- No-operation (NOP) instruction can be inserted between instructions to delay the respective reads relative to the writes to get the right value.
- Data forwarding is another solution to handle hazards.

Pipelining RISC Design: Data Hazards

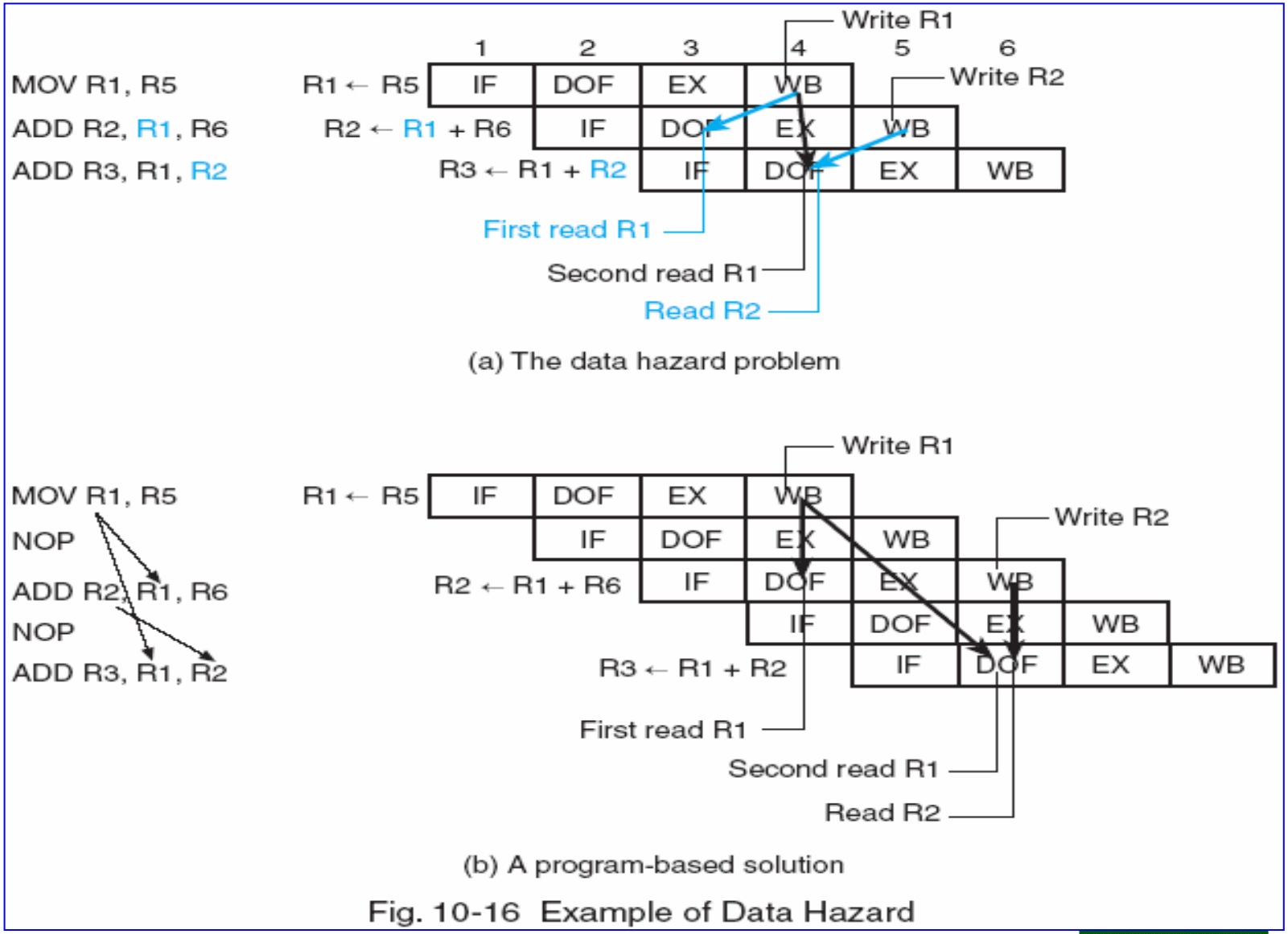


Fig. 10-16 Example of Data Hazard

Pipelining RISC Design: Data Hazard Stalls

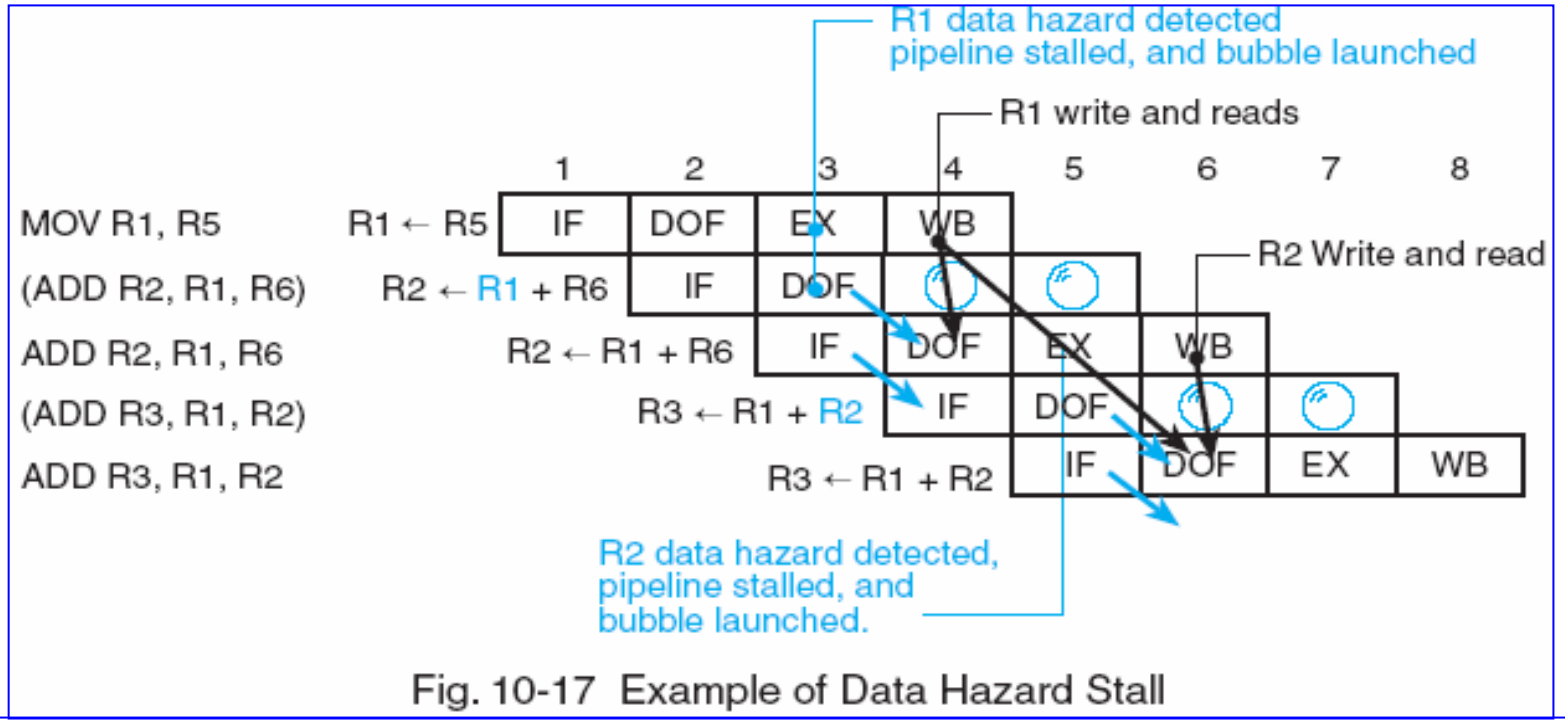
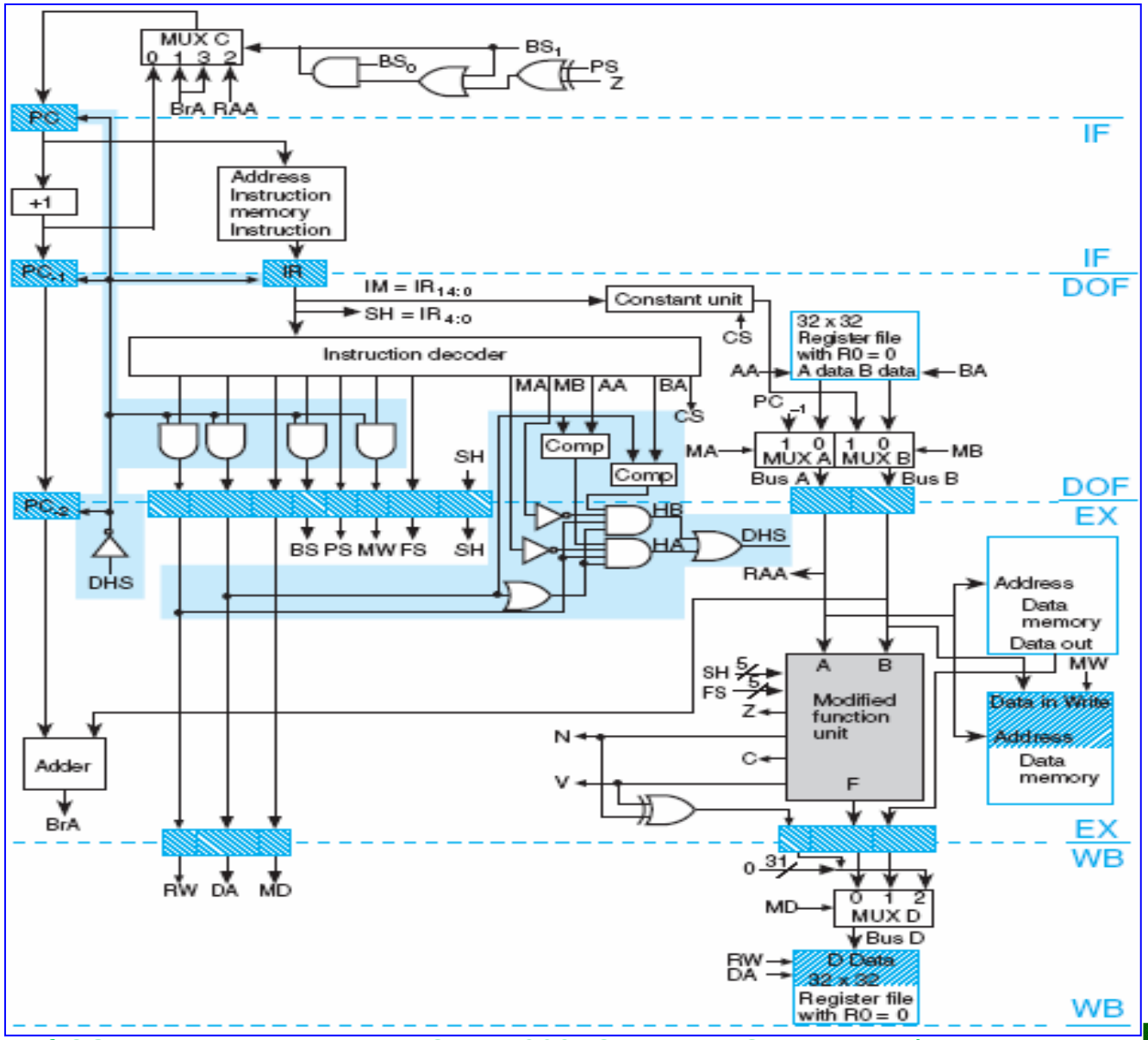


Fig. 10-17 Example of Data Hazard Stall

- Instead of the programmer or compiler placing NOPs, the hardware inserts NOPs automatically.
- When an operand is present at DOF stage, and value hasn't been written back yet, the associated execution and write-back stages are stalled for one clock cycle.

Pipelining RISC CPU with Data Hazard Stalls



High Performance CPU

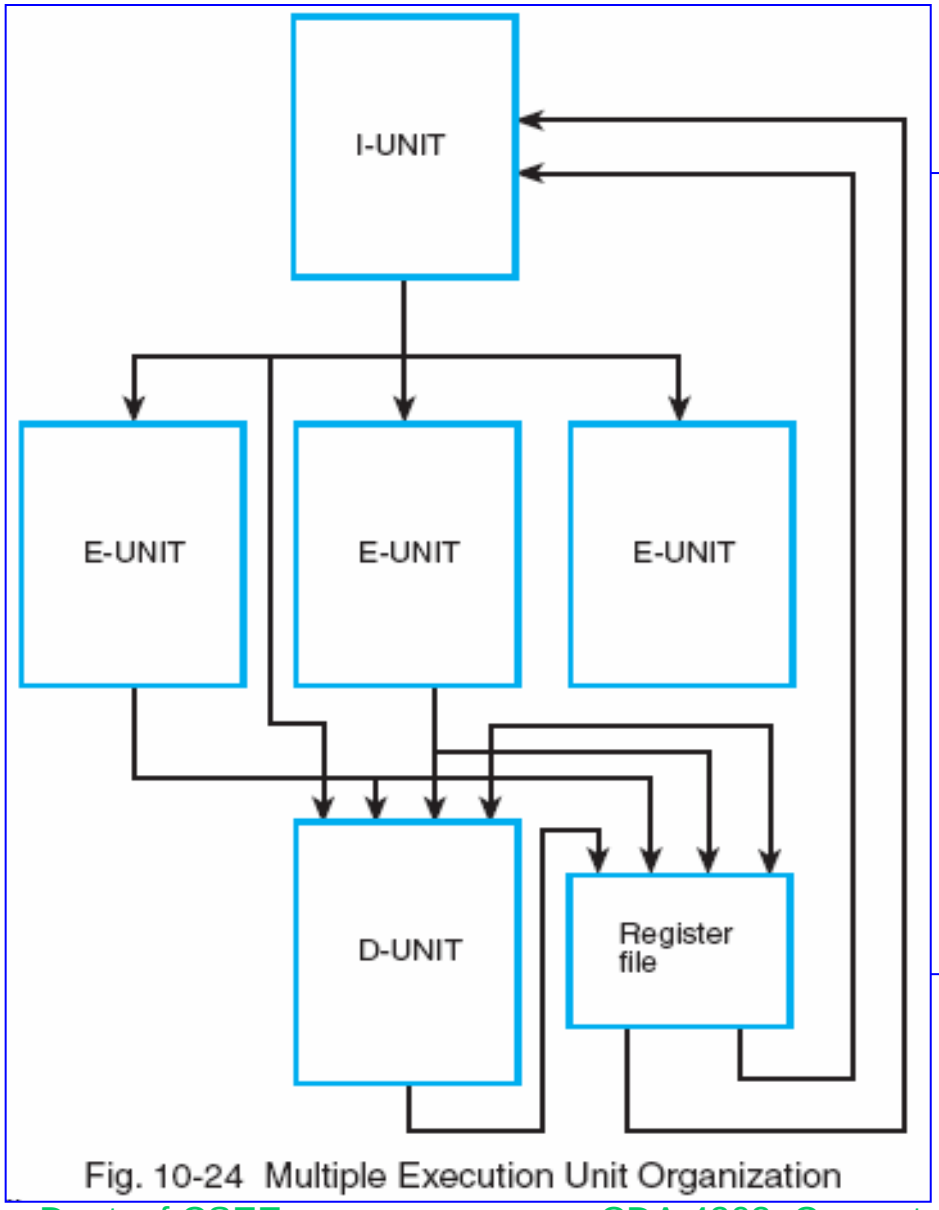
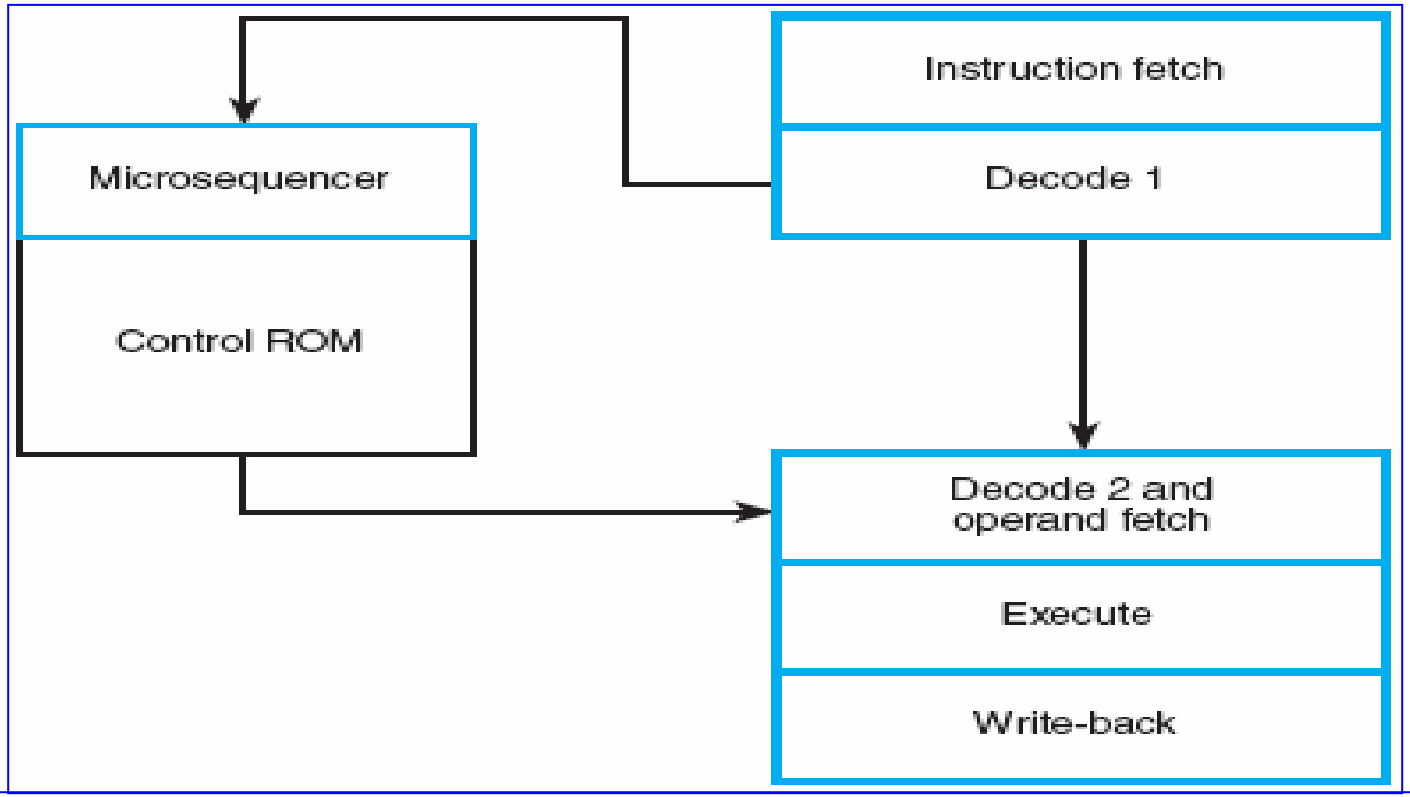


Fig. 10-24 Multiple Execution Unit Organization

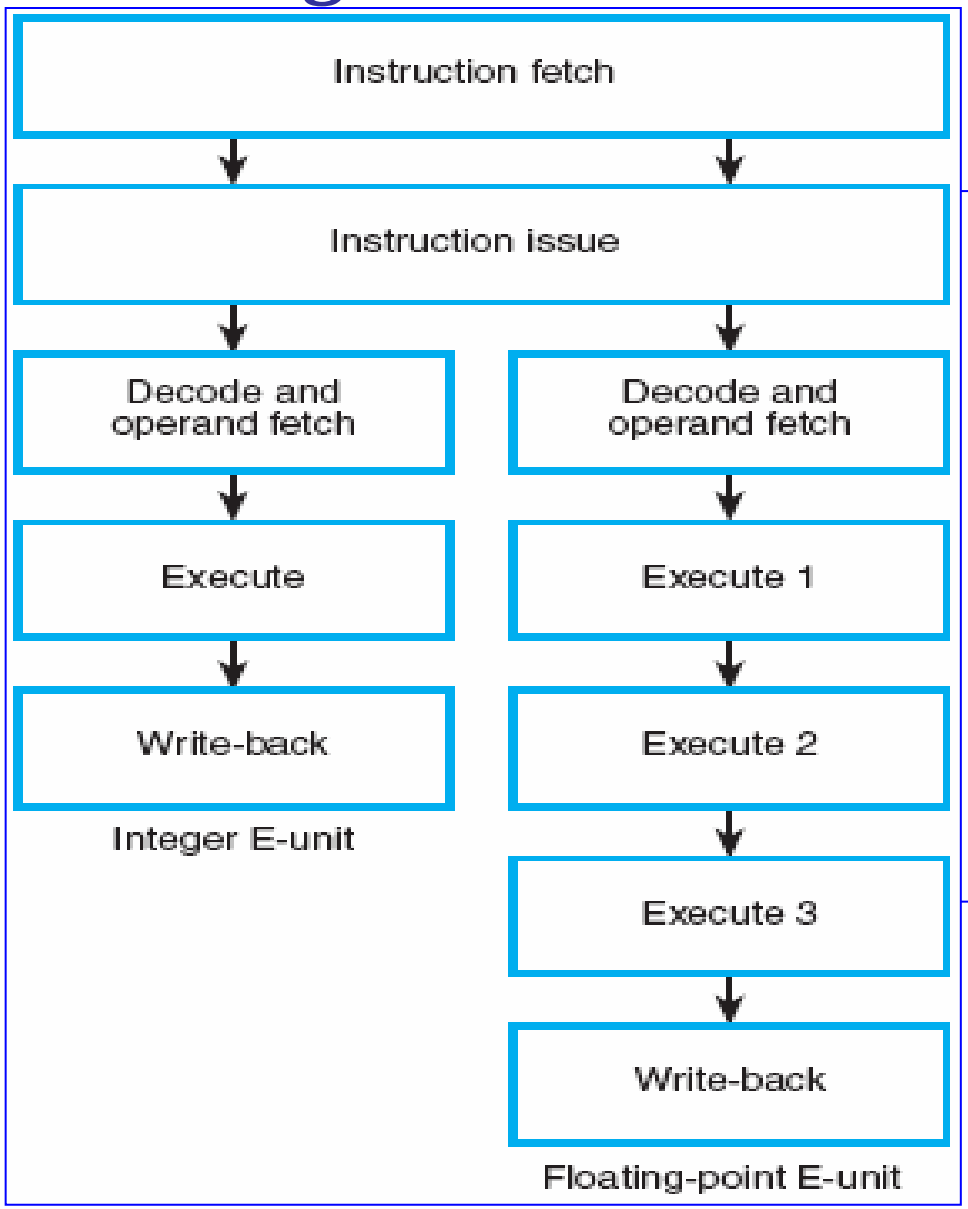
- When an operation takes multiple clock cycles to execute, the performance of the CPU can be substantially improved by having multiple execution units in parallel.
- I-unit handles instruction fetch, decode, operation fetch and branches.
- E-unit handles execution of an instruction after fetching operands.
- D-unit executes the memory write.

High Performance CPU (Combined CISC-RISC Organization)



To implement a CISC architecture and also achieve initiating and completing instructions close to one instruction per short RISC clock cycle, a pipelined data path and a combination of pipelined and micro programmed control is used.

High Performance CPU: Superscalar



- The goal is to have a peak rate of initiating instructions in excess of one instruction per clock cycle.
- The processor checks for hazards among instructions – in case of hazards the instructions are held for later execution.