Computer Organization & Architecture

Chapter 2 – Instructions & Instruction Sequencing

Zhang Yang 张杨
cszyang@scut.edu.cn

Autumn 2025

Content of this lecture

- 2.3 Instruction and Instruction Sequencing
  - Four Types of Instructions
  - Register Transfer Notation
  - Assembly-Language Notation
  - RISC and CISC Instruction Sets
  - Introduction to RISC Instruction Sets
  - Instruction Execution and Straight-Line Sequencing
  - Branching

Four Types of Instructions

- Four Types of Instructions
  - Data transfers between the memory and the processor registers
  - Arithmetic and logic operations on data
  - Program sequencing and control
  - I/O transfers

Register Transfer Notation

- Memory Location: LOC, PLACE, A, VAR2
- Processor Register: R0, R5
- I/O Subsystem Register: DATAIN, OUTSTATUS
- The contents of any location are denoted by placing square brackets around its name: R2 ← [LOC]
- Register Transfer Notation: R4 ← [R2] + [R3]
    - The righthand side of an RTN expression always denotes a value, and the left-hand side is the name of a location where the value is to be placed, overwriting the old contents of that location.

Assembly-Language Notation

- Example1: Load R2, LOC
  - The transfer from memory location LOC to processor register R2.
- Example2: Add R4, R2,R3
  - Registers R2 and R3 hold the source operands, while R4 is the destination.

Mnemonics

- Assembly languages for different processors often use different mnemonics for a given operation.
- E.g.

Load	LD
Store	STR or ST

RISC and CISC Instruction Sets (1)

- Design a computer starting by defining ISA.
  - ISA is instruction set architecture.
    - Defines registers.
    - Defines data transfer modes (instructions) between registers, memory and I/O.
    - There should be *sufficient* instructions to efficiently translate any program for machine processing.
- Next, define instruction set format – binary representation used by the hardware.
  - Variable-length vs. fixed-length instructions

RISC and CISC Instruction Sets (2

- Types of ISA
  - Reduced instruction set computer (RISC)
    - Small set of instructions (typically 32).
    - Simple instructions, each executes in one clock cycle – ***REALLY? Well, almost.***
    - Effective use of pipelining.
    - Example: ARM
  - Complex instruction set computer (CISC)
    - Many instructions (several hundreds).
    - An instruction takes many cycles to execute.
    - Example: Intel Pentium

Introduction to RISC Instruction Sets (1)

- Two key characteristics of RISC instruction sets
  - Each instruction fits in a single word.
  - A *load/store architecture* is used, in which
    - Memory operands are accessed only using Load and Store instructions.
    - All operands involved in an arithmetic or logic operation must either be in processor registers, or one of the operands may be given explicitly within the instruction word.

Introduction to RISC Instruction Sets (2)

- Load Instruction Format
  - Load *destination*, *source*
  - Load *processor_register*, *memory_location*
- Store Instruction Format
  - Store *source*, *destination*
- Add Instruction Format
  - Add *destination*, *source1*, *source2*

Introduction to RISC Instruction Sets (3)

- Example: C = A + B

  $C \leftarrow [A] + [B]$

  - Load R2, A
  - Load R3, B
  - Add R4, R2, R3
  - Store R4, C

Instruction Execution and Straight-Line Sequencing (1)

- Example:

$C = A + B$

$C \leftarrow [A] + [B]$

Assume that

- The word length is 32 bits
- The memory is byte-addressable
- A desired memory address can be directly specified in Load and Store instructions.

Instruction Execution and Straight-Line Sequencing (2)

- Straight-line Sequencing

  [PC]= i

  [PC]= i +4

  …

  - Instruction Fetch
  - Instruction Execute

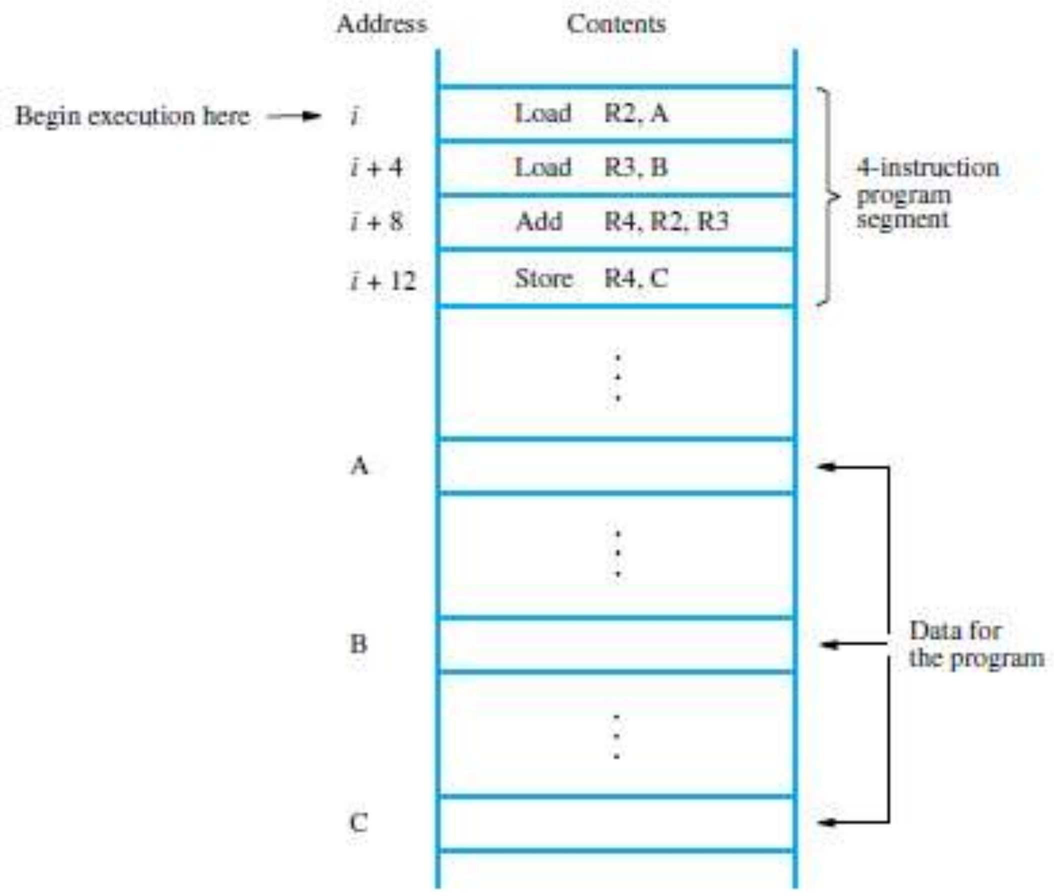

Figure 2.4 A program for C ← [A] + [B].

Instruction Execution and Straight-Line Sequencing (3)

- Branching
  - Adding a list of n numbers
    - Separate Load and Add Instructions

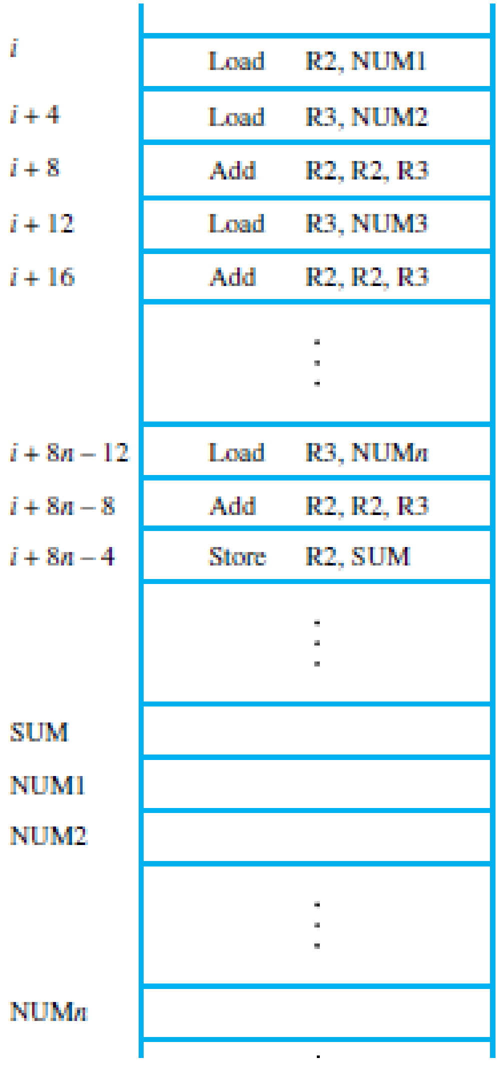

Figure 2.5 A program for adding n numbers.

Instruction Execution and Straight-Line Sequencing (4)

- Branching (ctd.)
  - Adding a list of n numbers
    - A program loop.
    - Branch Instructions
      - Load a new address into the PC.
      - Branch Target
    - Conditional Branch
      - Compare the contents of two registers.
        Branch_if_[R4]>[R5] Loop
      - Condition Codes

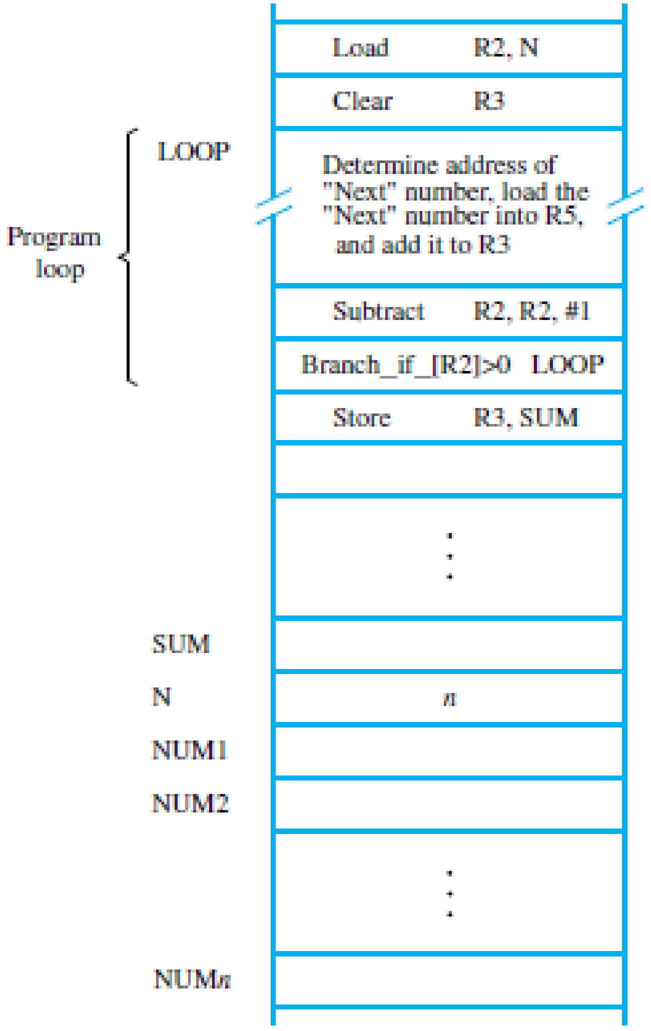

Figure 2.6 Using a loop to add n numbers.