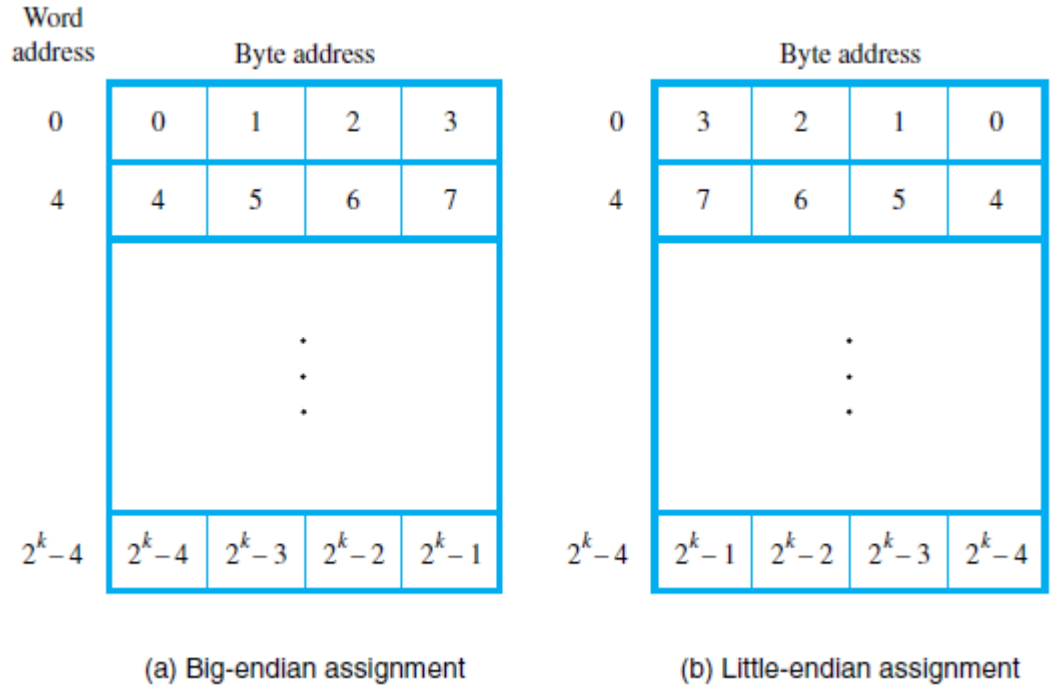
COMPUTER ORGANIZATION AND ARCHITECTURE

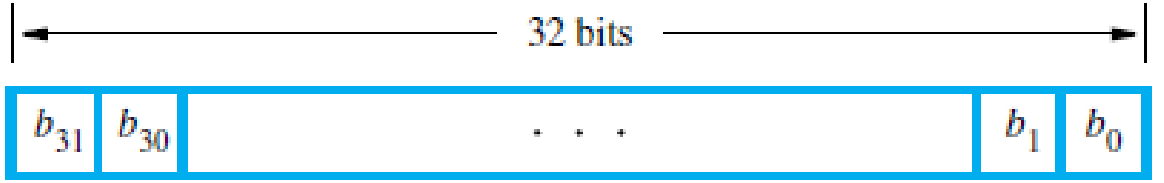
Course Code : CSE 2151

Credits : 04

BYTE ADDRESSABILITY

- Two ways that byte addresses can be assigned across words
  - Big-Endian: lower byte addresses are used for the most significant bytes of the word
  - Little-Endian: opposite ordering. Lower byte addresses are used for the less significant bytes of the word
    - Example: 15326→ 1, 53 and 26
    - Big-Endian: Little-Endian:
    - Address: Value Address: Value
    - 1000 : 01 1000 : 26
    - 1001 : 53 1001 : 53
    - 1002 : 26 1002 : 01
- The words "more significant" and "less significant" are used in relation to the powers of 2 assigned to bits when the word represents a number.





WORD ALIGNMENT

- Word locations have aligned addresses if they begin at a byte address i.e., a multiple of the number of bytes in a word.
  - 16-bit word: word addresses: 0, 2, 4,….
  - 32-bit word: word addresses: 0, 4, 8,….
  - 64-bit word: word addresses: 0, 8,16,….
- If words begin at arbitrary byte address, words are said to have unaligned addresses.

ACCESSING NUMBERS AND CHARACTERS

- A number usually occupies one word and can be accessed in the memory by specifying its word address.
- Individual characters can be accessed by their byte address.

MEMORY OPERATIONS

- Load (or Read or Fetch)
  - Copy the content. The memory content doesn't change.
  - Address – Load
  - Registers can be used
- Store (or Write)
  - Overwrite the content in memory
  - Address and Data – Store
  - Registers can be used

INSTRUCTION AND INSTRUCTION SEQUENCING

- "Must-Perform" Operations
  - 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

- Identify a location by a symbolic name standing for its hardware binary address (LOC, R0,…)
- Contents of a location are denoted by placing square brackets around the name of the location
- e.g. R1←[LOC]

  R3 ←[R1]+[R2]
- Register Transfer Notation (RTN)

ASSEMBLY-LANGUAGE NOTATION

- Represent machine instructions and programs.
- Move R1,LOC → R1←[LOC]
- Add R3, R1, R2 → R3 ←[R1]+[R2]

RISC AND CISC INSTRUCTION SET

- Most important characteristics that distinguish different computers is the nature of their instructions
- Reduced Instruction Set Computers (RISC)
  - Higher performance can be achieved if each instruction occupies exactly one word in memory, and all operands needed to execute a given arithmetic or logic operation specified by an instruction are already in processor registers
  - Various operations needed to process a sequence of instructions are performed in "pipelined" fashion
  - Number of different types of instructions may be included in the instruction set of a computer.
- Complex Instruction Set Computers (CISC)
  - An alternative to RISC
  - makes use of more complex instructions which may span more than one word of memory, and
  - may specify more complicated operations.

RISC INSTRUCTION SETS

- Two key characteristics of RISC instruction sets are:
  1. Each instruction fits in a single word.
  2. 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.
- At the start of execution of a program, all instructions and data used in the program are stored in the memory of a computer.
- Processor registers do not contain valid operands at that time.
- Load instructions: Operands expected to be in processor registers before they can be used by an instruction, are transferred from memory location into the registers.
- Load instructions are of the form
  - Load destination, source
- In general,
  - Load processor_register, memory_location

RISC INSTRUCTION SETS

- C = A + B
- C ← [A] + [B]
- Assembly:
  - Load R2, A
  - Load R3, B
  - Add R4, R2, R3
  - Store R4, C
- Three-Address Instructions
  - Add destination, source1, source2
- The Store instruction is of the form
  - Store source, destination

INSTRUCTION EXECUTION AND STRAIGHT-LINE SEQUENCING

- a possible program segment for the task, C=A+B, as it appears in the memory of a computer.
- Straight-line sequencing: using the information in the PC to fetch and execute instructions, one at a time, in the order of increasing addresses.
- Executing a given instruction is a two-phase procedure.
  - Instruction fetch (First phase)
  - Instruction execute (Second phase)

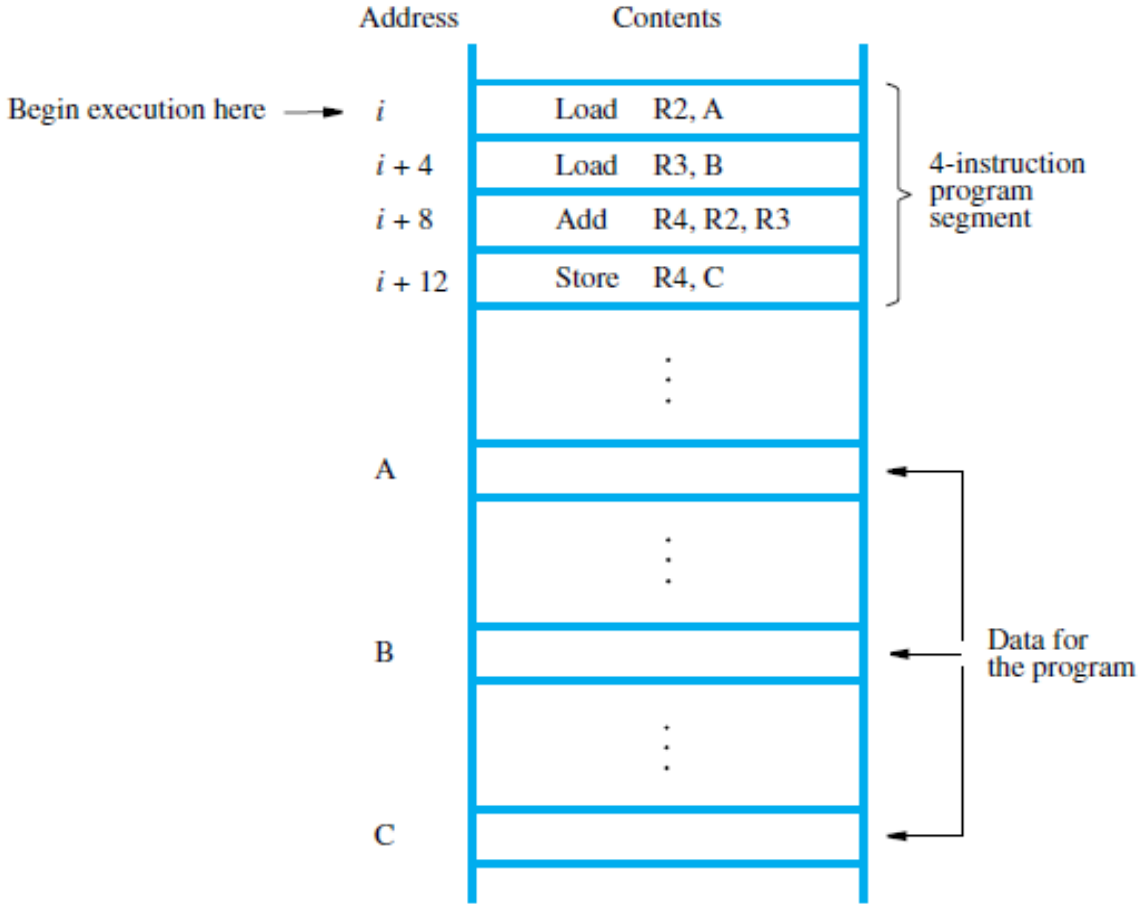


Figure 2.4 A program for C ← [A] + [B].

TOPICS COVERED FROM

- Textbook 1:
  - Chapter 2: 2.1, 2.2, 2.3