# Computer Organization & Architecture

# Chapter 2 – Instruction Formats

Zhang Yang　张杨
cszyang@scut.edu.cn

Autumn 2021

## Content of this lecture

- Instruction Formats (补充内容)
  - Machine Instruction and Instruction Set
  - Elements of A Machine Instruction
  - Instruction Length
  - Instruction Format
  - Address Code Filed Format
  - Opcode Field Format (Reference Book 5.3 in “Structured Computer Organization”, 6th ed., Andrew S. Tanenbaum)
  - Example of Instruction Format

## Machine Instruction and Instruction Set

- Machine Code
  - A set of binary codes that are recognized and executed directly by a particular processor.
- Machine Instruction
  - An individual machine code is called a Machine Instruction.
  - E.g. the machine instruction to add 1 to the value in accumulator AC is 01001100
  - Usually represented by assembly codes.
- Instruction Set
  - The collection of different machine instructions that the processor can execute.

## Elements of A Machine Instruction (1)

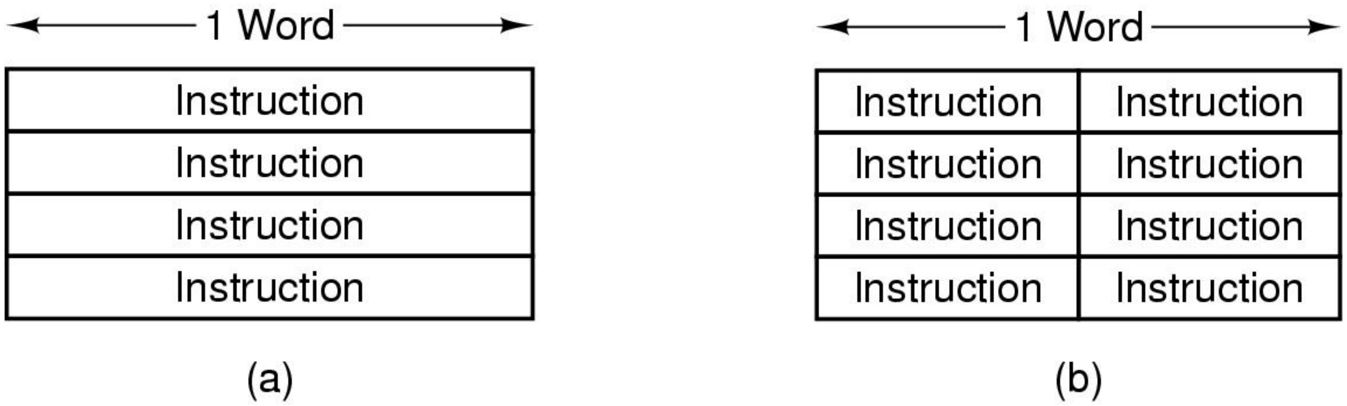
- Operation code
  - Specify the operation to be performed (e.g.,ADD, I/O), expressed as a binary code.
- Source operand reference
  - Operands required for the instruction are specified.
- Result operand reference
  - Where should the result of the operation be placed?
  - Source and result operands can be in one of three areas:
    - Main or virtual memory or cache
    - Processor register
    - I/O device

## Elements of A Machine Instruction (2)

- Next instruction reference
  - How / where is the next instruction to be found.
    - In most cases, this is not explicitly stated in the instruction.
    - Next instruction is the one that logically follows the current one in the program (sequential / linear progression through the program).

# Instruction Length (1)

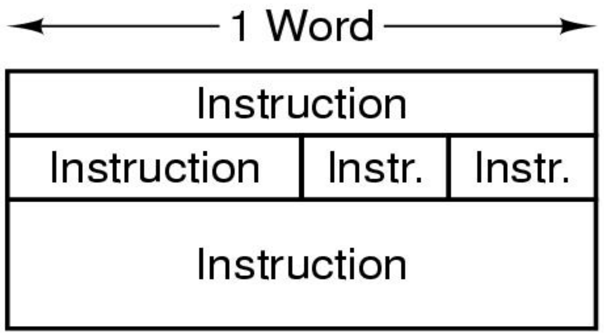
- Fixed Length
  - All instructions have the same length



  - Advantages
    - Simple to decode, reducing the amount of decode logic required and the latency of the decode logic.
    - A processor that uses this encoding can easily predict the location of the next instruction to be executed (assuming that the current instruction is not a branch). This makes it easier for the processor to use pipelining.

## Instruction Length (2)

- Variable Length
  - Instructions may be many different lengths



(c)

  - Advantage
    - Each instruction takes only as much space in memory as it requires. So it can reduce the amount of space taken up by a program.

## Instruction Length (3)

- Variable Length (ctd.)
  - Disadvantages
    - It greatly increases the complexity of the logic required to decode instructions.
    - The hardware cannot predict the location of the next instruction until the current instruction has been decoded enough to know how long the current instruction is.

## Instruction Length (4)

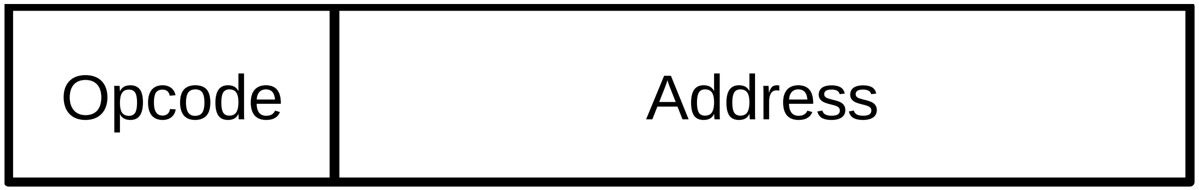
- Fixed Length vs. Variable Length
  - Fixed-length encodings are more common in recent architectures.
  - Variable-length encodings are mainly used in architectures where there is a large variance between the amount of space required for the longest instruction in the ISA and the average instruction in the ISA.
    - Stack architecture
    - CISC architecture

## Instruction Length (5)

- Methods to Reduce Instruction Length
  - If the operand is to be used several times, it can be moved into a register. (If an operand is to be used only once, putting it in a register is not worth it.)
  - Specify one or more operands implicitly.

## Instruction Format (1)

- Within the computer, each instruction is represented by a sequence of binary bits.
- The bits of the instruction are divided into groups called fields.
- Two Fields



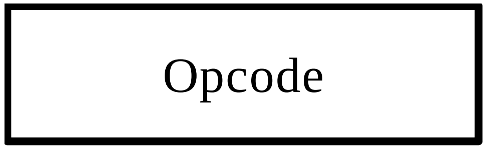
  - Operation Code Field
    - Specify the operation to be performed.
  - Address Code Field
    - Specify the source and result operand address.

## Instruction Format (2)

- Instruction Design Criteria
  - Short instructions are better than long ones.
  - Sufficient room in the instruction format to express all the operations desired.
  - Number of bits in the address field.

# Address Code Field Format (1)

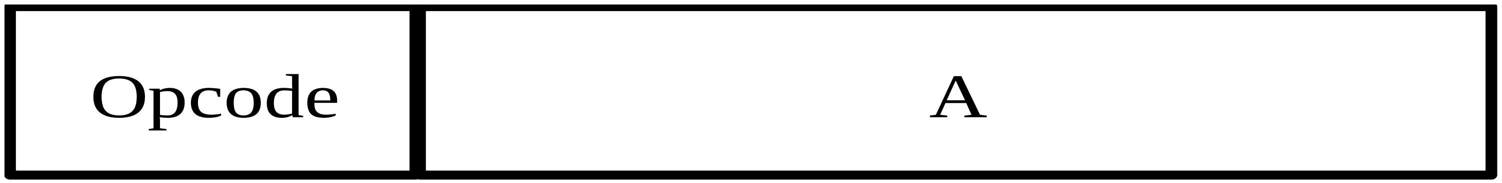
- Zero-Address Instruction



  - Contains no address fields
  - Source and result operands are both implicit.
  - E.g. push, pop, halt

## Address Code Field Format (2)

- One-Address Instruction



  - Operation 1: OP [A] → A
  - Operation 2: [AC] OP [A] → AC
    - Typically, it is understood implicitly that a second operand is in the accumulator of the processor.

## Address Code Field Format (3)

- Two-Address Instruction

| | | |
|---|---|---|
| Opcode | A 1 | A2 |

  - Most common in commercial computers.
  - Each address field can specify either a processor register or a memory word.
  - Operation: [A1] OP [A2] $\rightarrow$ A2
  - A1, A2: Source operand address
  - A2: Result operand address

## Address Code Field Format (4)

- Three-Address Instruction

| | | | |
|---|---|---|---|
| Opcode | A1 | A2 | A3 |

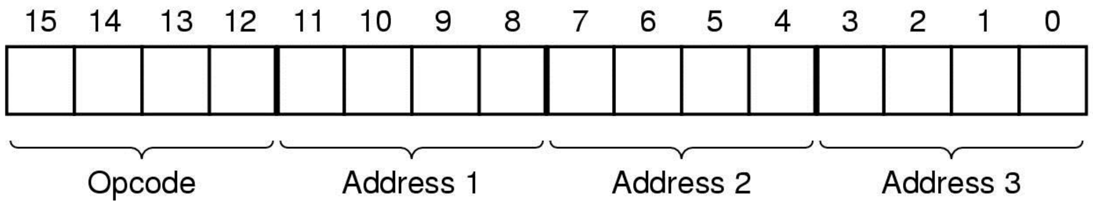
  - Each address field can specify either a processor register or a memory word.
  - Operation: [A1] OP [A2] → A3
  - A1, A2: Source operand address
  - A3: Result operand address

## Opcode Field Format (1)

- Fixed-length Opcode
  - Instruction length: n + k bits
  - *k* bits opcode and *n* bits operand address
    - Allows for $2^k$ different operations
    - Allows for $2^n$ addressable memory cells
  - *k-1* bit opcode and *n+1* bit address
    - Half as many instructions but twice the addressable memory
  - *k+1* bit opcode and *n-1* bit address
    - Twice as many instructions but half the addressable memory

# Opcode Field Format (2)

- Variable-length Opcode (Expanding Opcode)
  - Example: Instruction length is 16-bit, operand address is 4-bit.
    - This might be reasonable on a machine that has 16 registers on which all arithmetic operations take place.
    - One design would be a 4-bit opcode and three addresses in each instruction, giving 16 three-address instructions.

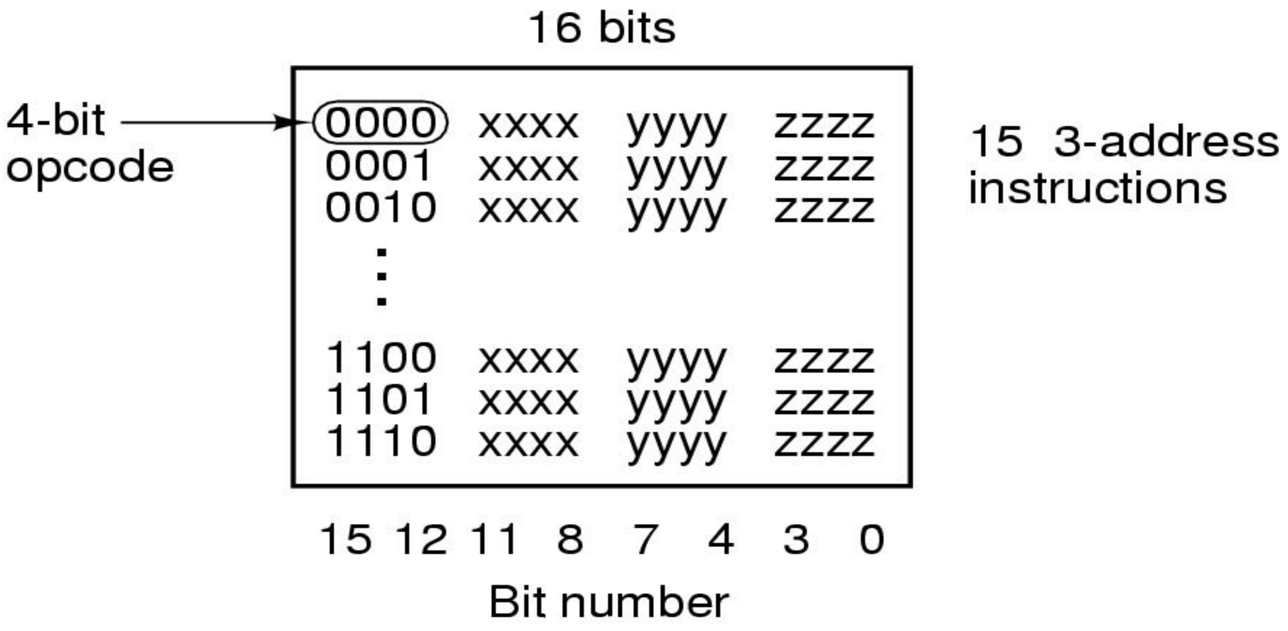


## Opcode Field Format (3)

- Variable-length Opcode (Expanding Opcode) (ctd.)
  - Example: Instruction length is 16-bit, operand address is 4-bit. (ctd.)
    - Suppose the designers need:
      - 15 three-address instructions
      - 14 two-address instructions
      - 31 one-address instructions
      - 16 zero-address instructions
    - How should we design the instruction format?

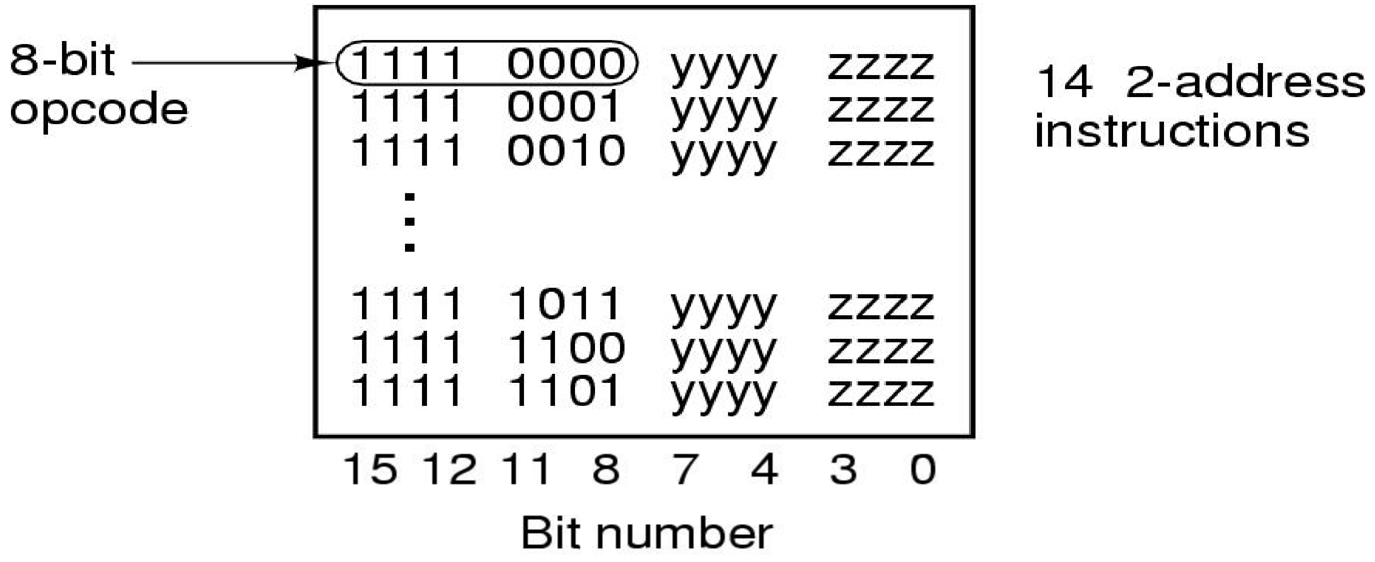
# Opcode Field Format (4)

- Variable-length Opcode (Expanding Opcode)
  - Example: (ctd.)
    - Three-address instruction
      - 4-bit opcode 0000 – 1110 (15 – 12 bit)



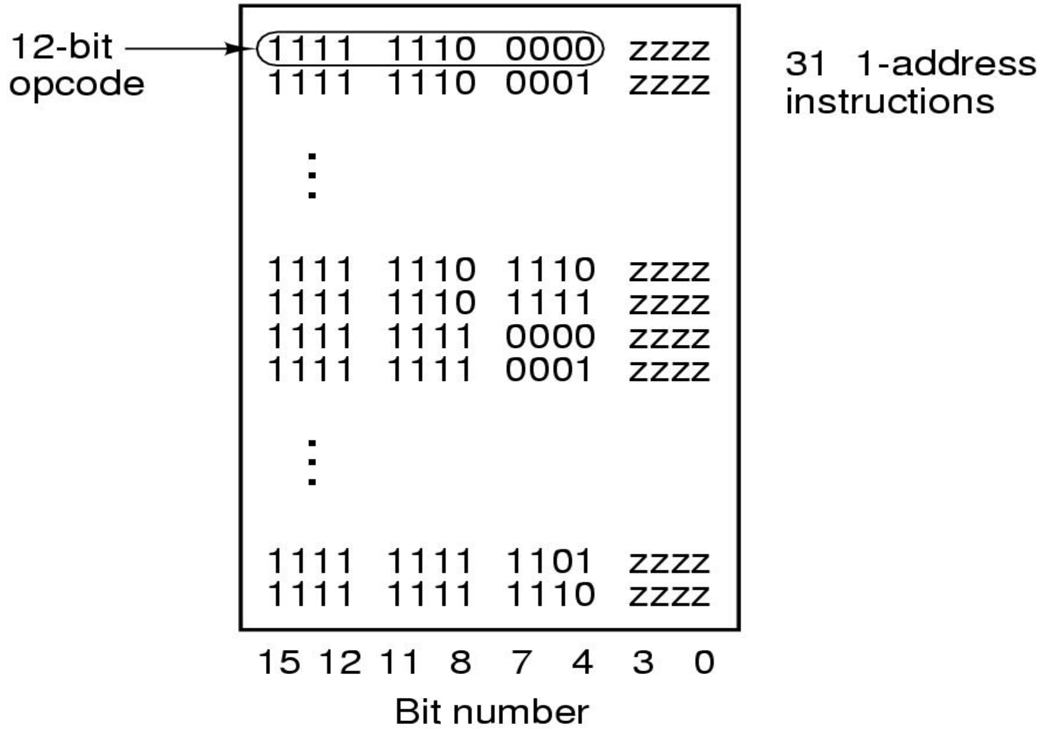
# Opcode Field Format (5)

- Variable-length Opcode (Expanding Opcode)
  - Example: (ctd.)
    - Two-address instruction
      - 8-bit opcode 11110000 – 11111101 (15 – 8 bit)
      - 1111 (15 – 12 bit) 0000 – 1101 (11 – 8 bit)



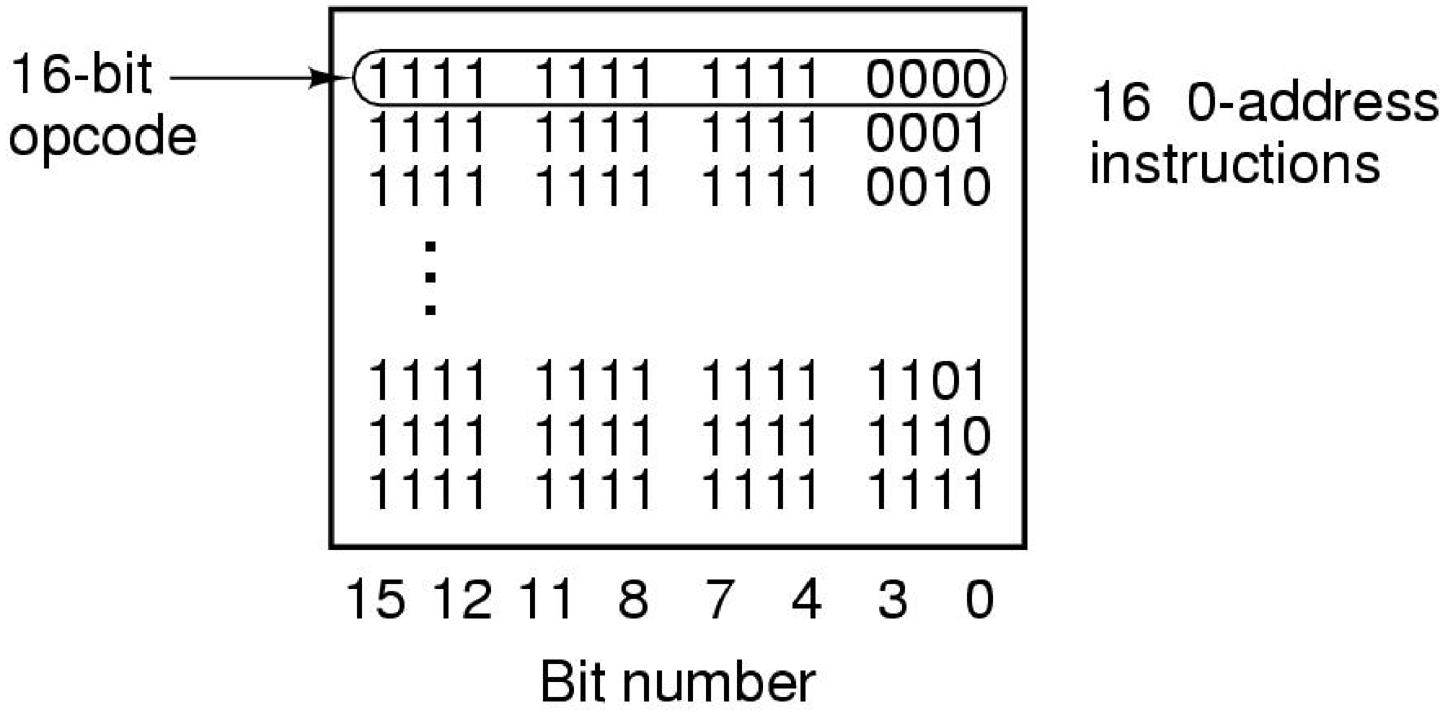
# Opcode Field Format (6)

- Variable-length Opcode (Expanding Opcode)
  - Example: (ctd.)
    - One-address instruction
      - 12-bit opcode 111111100000 – 111111111110 (15 – 4 bit)



# Opcode Field Format (7)

- Variable-length Opcode (Expanding Opcode)
  - Example: (ctd.)
    - Zero-address instruction
      - 16-bit opcode 1111111111110000 - 1111111111111111

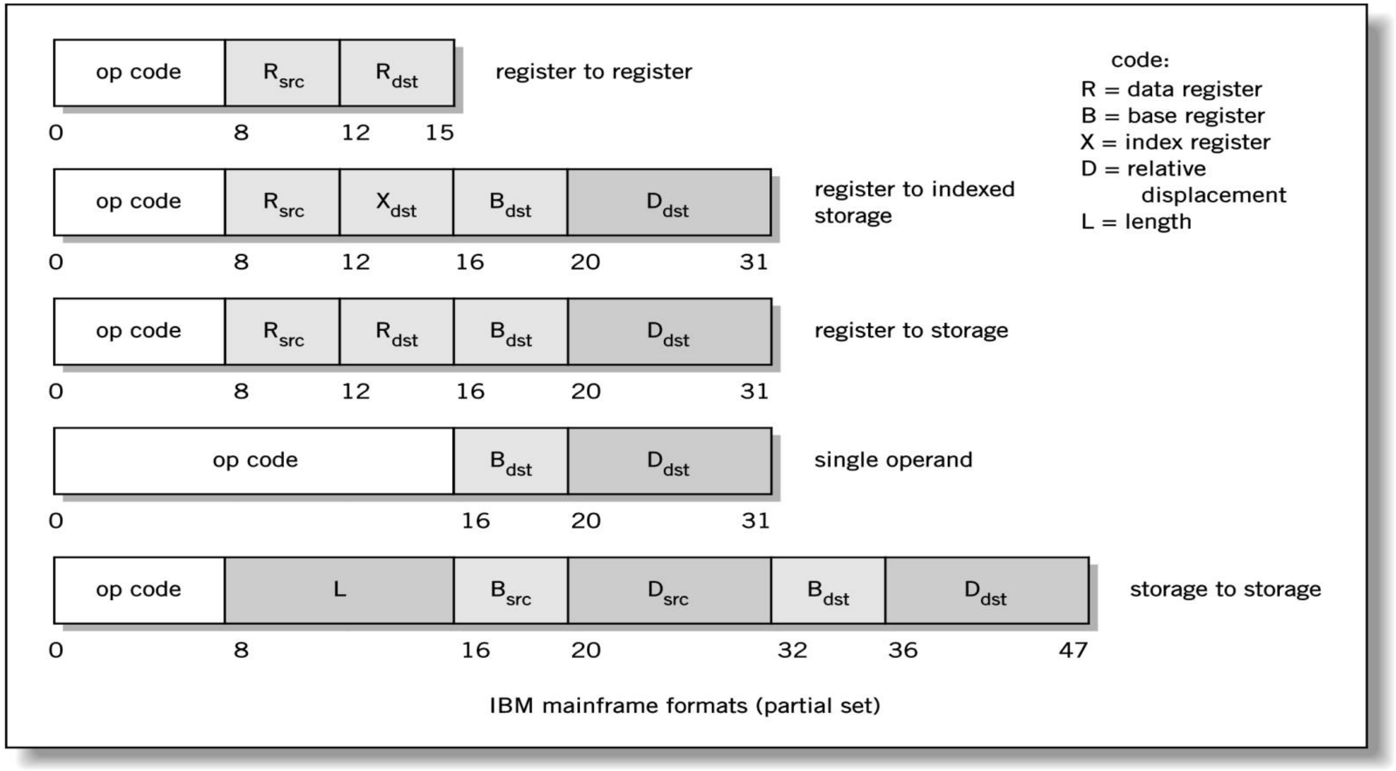


## Opcode Field Format (8)

- Variable-length Opcode (Expanding Opcode)
  - Summary
    - Expanding Opcodes demonstrates a trade-off between space for Opcode and space for other information.
    - Carrying variable Opcode to an extreme, it is possible to minimize the average instruction length by encoding every instruction to minimize bits needed.
    - However, this will result in instructions of various sizes that are not even aligned on byte boundaries.
    - Expanding Opcodes is typically employed at the byte level like the above example.

## Example of Instruction Format (1)

- Figure



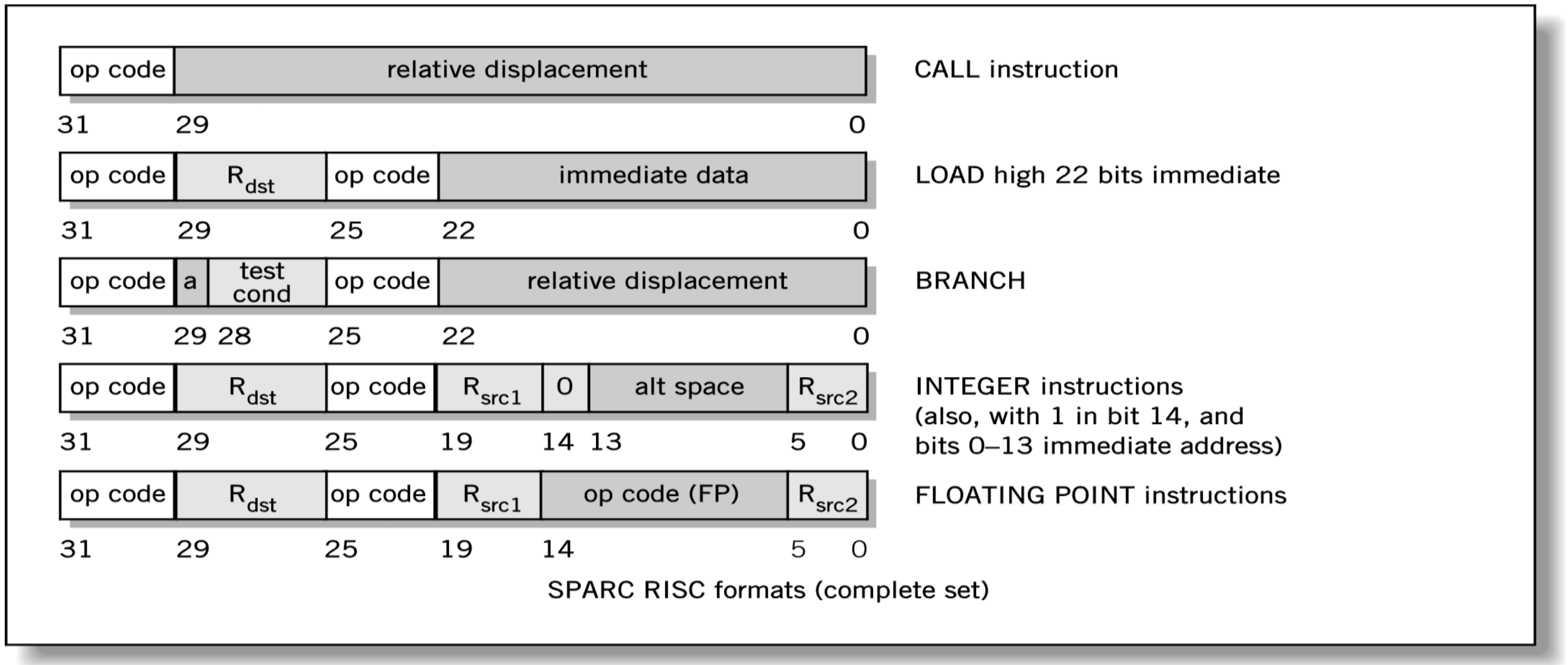
IBM mainframe formats (partial set)

(Figure continues on next slide)

Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition
Chapter 7, Figure 07-15

# Example of Instruction Format (2)

- Figure (ctd.)



SPARC RISC formats (complete set)

Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition
Chapter 7, Figure 07-15 continued

## Summary

- 知识点: Instruction Format
  - 理解What is Instruction Set
  - 理解Elements of An Instruction
  - 了解Instruction Length
  - 掌握Address Field Format
  - 掌握Opcode Field Format (Expanding Opcode)

## Exercise (1)

- 1. Assume that a computer has a 16-bit instruction format. If the opcode field is 4-bit and it applies fixed-length opcode, how many instructions does this computer have at most?
  - A.64
  - B.16
  - C.32
  - D.128

# Exercise (2)

- 2. In a computer’s instruction set, there are three types of instructions: one-, two-, three-address instructions. Assume that the fixed length of opcode is 7-bit. If there are m one-address instructions and n two-address instructions, how many three-address instructions can be designed at most?
  - A. $2^7$- m
  - B. $2^7$- m+n
  - C. $2^7$- m - n
  - D. $2^7$- n

## Homework (1)

- 补充题
  - 1. Consider a processor with 64 registers and an instruction set of size 15. Each instruction has five distinct fields, namely, opcode, two source register identifiers, one destination register identifier, and a 12-bit immediate value. Each instruction must be stored in memory in a byte-aligned fashion. If a program has 100 instructions, what is the amount of memory (in bytes) consumed by the program text?

## Homework (2)

- 补充题
  - 2. Design a 16-bit instruction format using expanding opcode for the following: 14 three-address instructions, 30 two-address instructions, 30 one-address instructions, 32 zero-address instructions. Assume that there are 16 registers and operands are placed in register.