CMPE 311

Instruction Sets:

Characteristics and Functions

Addressing Modes

Slides modified from multiple sources

1. William Stallings Computer Organization and Architecture, 7th Edition
2.James Peckol, Embedded systems Design

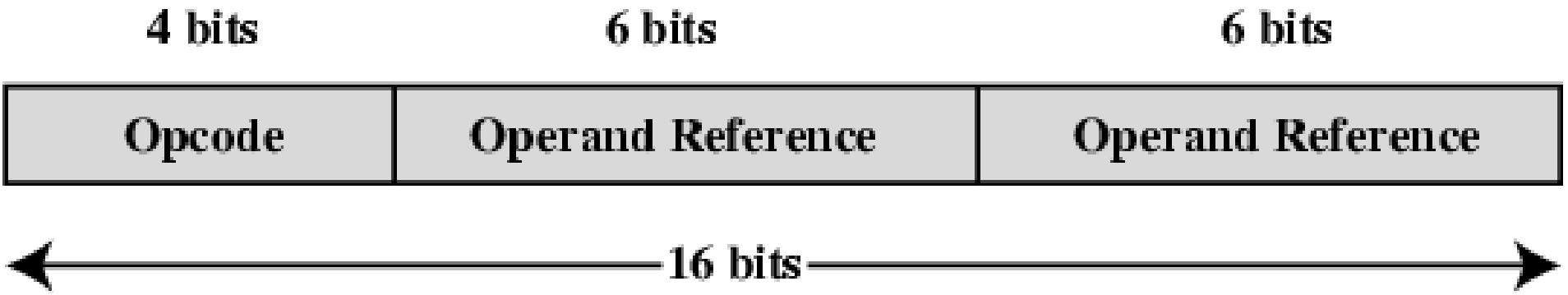
What is an Instruction Set?

- The complete collection of instructions that are understood by a CPU
- Machine Code
- Binary
- Usually represented by assembly codes

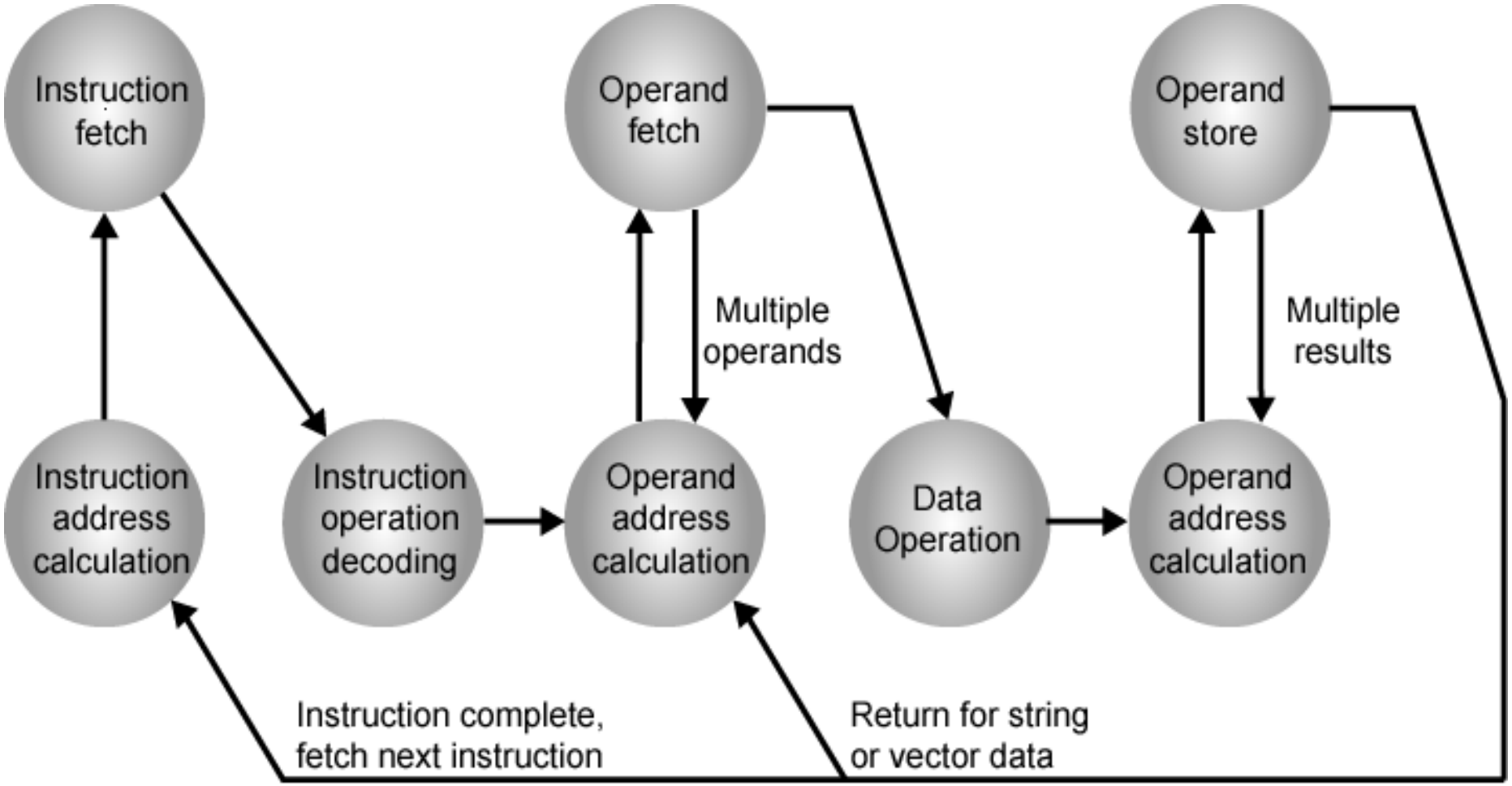
Elements of an Instruction

- Operation code (opcode)
  - Do this: ADD, SUB, MPY, DIV, LOAD, STOR
- Source operand reference
  - To this: (address of) argument of op, e.g. register, memory location
- Result operand reference
  - Put the result here (as above)
- Next instruction reference (often implicit)
  - When you have done that, do this: BR

Example: Simple Instruction Format (using two addresses)



Instruction Cycle State Diagram



Design Decisions (1)

- Operation
  - How many ops?
  - What can they do?
  - How complex are they?
- Data types (length of words, integer representation)
- Instruction formats
  - Length of op code field
  - Length and number of addresses (e.g., implicit addressing)

Design Decisions (2)

- Registers
  - Number of CPU registers available
  - Which operations can be performed on which registers? General purpose and specific registers
- Addressing modes (see later)
- RISC v CISC

Instruction Types

- Data transfer: registers, main memory, stack or I/O
- Data processing: arithmetic, logical
- Control: systems control, transfer of control

Data Transfer Instructions

- Are responsible for moving data around inside the processor as well as brining in data or sending data out
- Examples: Store, load, exchange, move, set, push, pop
- Each Instruction should have:
  - source and destination (memory, register, input/output port)
  - amount of data

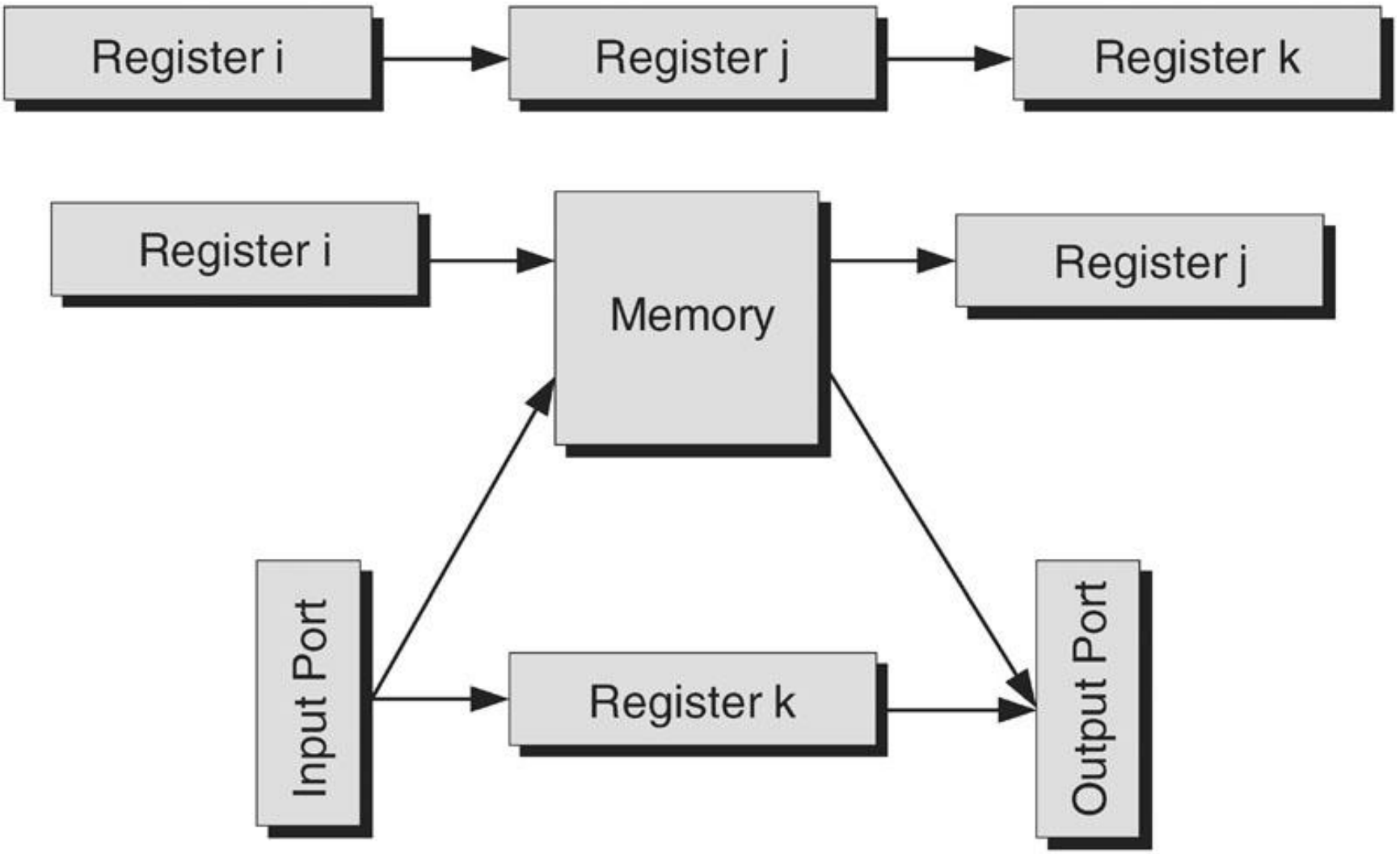

Register i
Register j
Register k
Register i
Memory
Register j
Input Port
Register k
Output Port

fig_01_16

Data Transfer Instructions Example

LD destination, source	Load—source operand transferred to destination operand can be either register or memory location.
ST source, destination	Store—source operand transferred to destination operand source must be a register and the destination must be memory.
MOVE destination, source	Transfer from register to register or memory to memory.
XCH destination, source	Interchange the source and destination operands.
PUSH/POP	Operand pushed onto or popped off of the stack.
IN/OUT destination, source	Transfer data from or to an input/output port.

Arithmetic

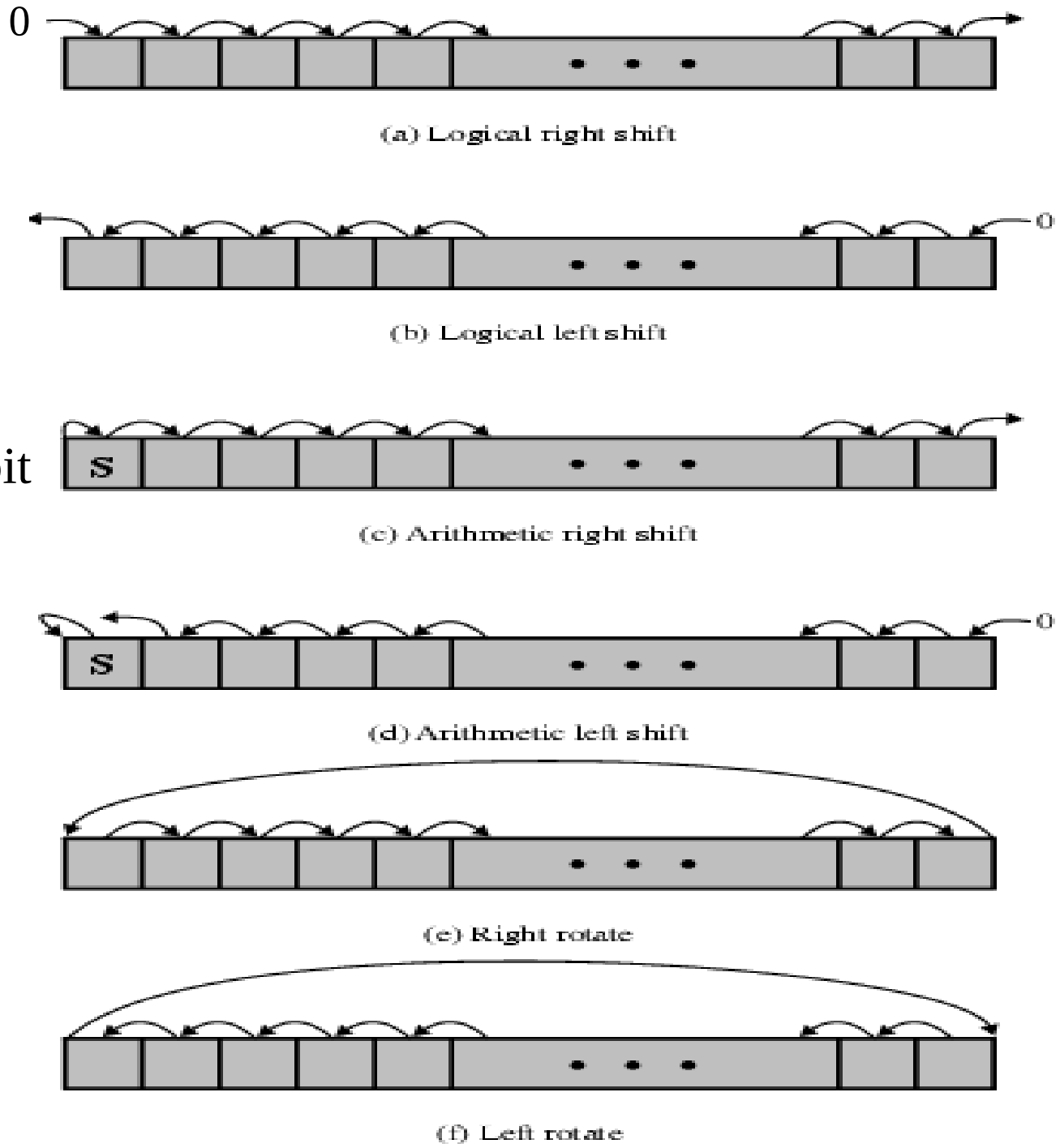
- Add, Subtract, Multiply, Divide for signed integer (+ floating point and packed decimal) – may involve data movement
- May include
  - Absolute (i.e |a|)
  - Increment (i.e a++)
  - Decrement (i.e a--)
  - Negate (i.e -a)

Logical

- Bitwise operations: AND, OR, NOT, XOR, CMP, SET
- Shifting and rotating functions, e.g.
  - logical right shift for unpacking: send 8-bit character from 16-bit word
  - arithmetic right shift: division and truncation for odd numbers
  - arithmetic left shift: multiplication without overflow

Different Shift Instructions

S is sign bit



Systems Control and Execution Flow

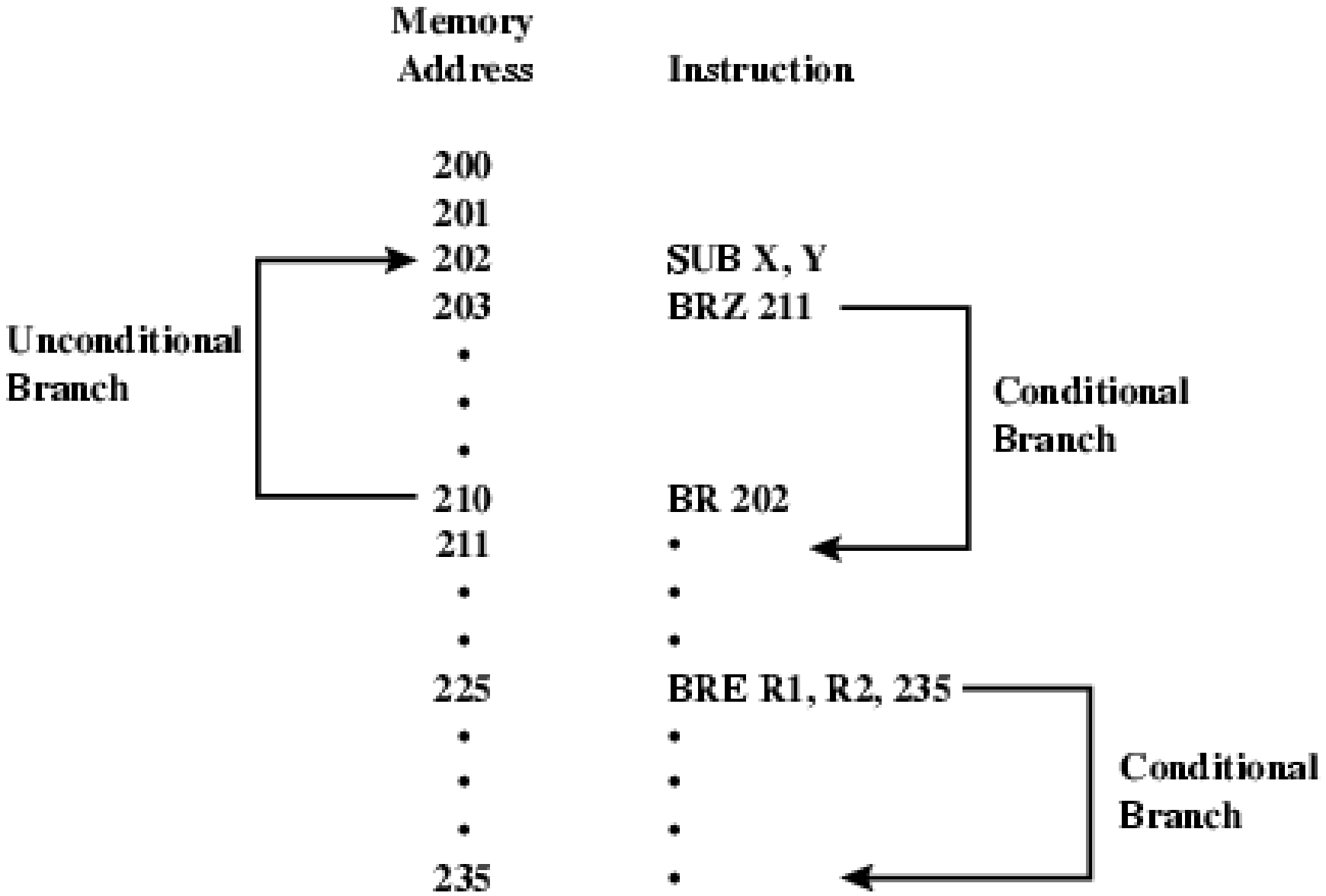
- The execution flow captures the order of evaluation/execution of each instruction
  - Sequential
  - Branch
  - Loop
  - Procedure or Function call

Branch

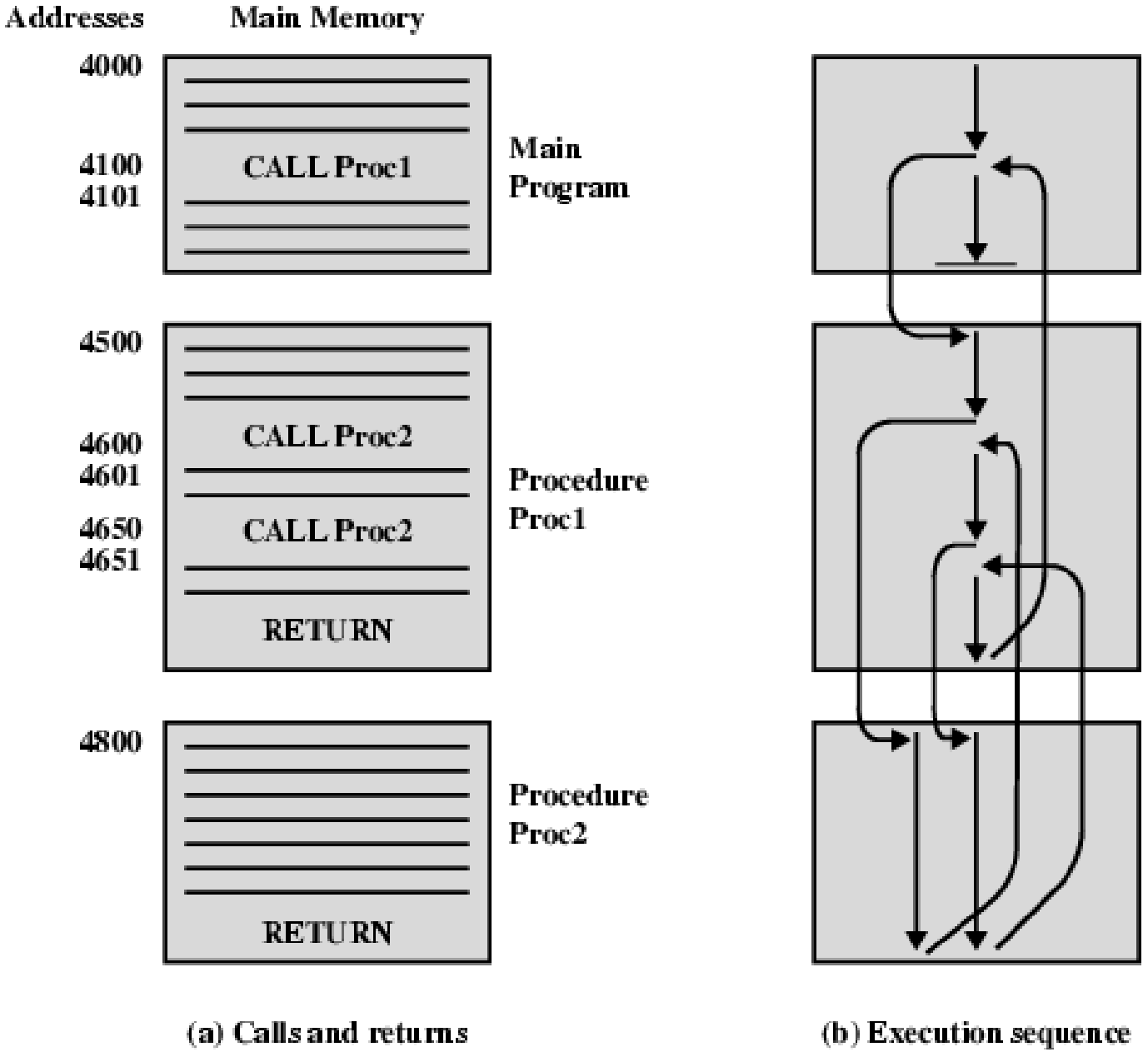
- Skip, e.g., increment and skip if zero:

  ISZ Reg1, cf. jumping out from loop
- Branch instructions: BRZ X (branch to X if result is zero), BRP X (positive), BRN X (negative), BRE X,R1,R2 (equal)
- Procedure (economy and modularity): call and return

Branch Instruction



Nested Procedure Calls



STI	- Store immediate
LDI / LOADI	- Load Immediate
MOVI	- Move Immediate

fig_01_20

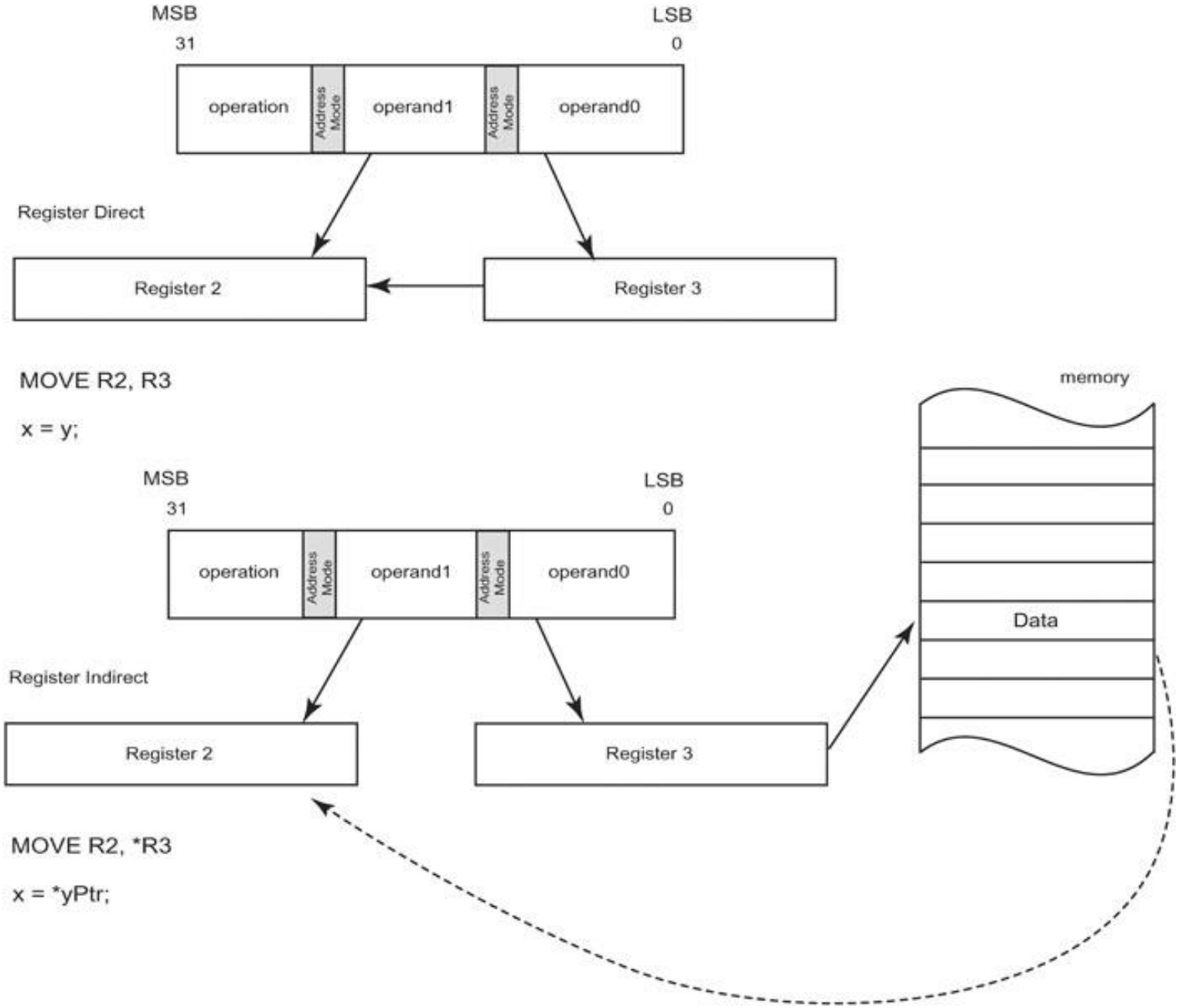

MSB
31
LSB
0
operation
Address Mode
operand1
Address Mode
operand0
Register Direct
Register 2
Register 3
MOVE R2, R3
x = y;
memory
MSB
31
LSB
0
operation
Address Mode
operand1
Address Mode
operand0
Data
Register Indirect
Register 2
Register 3
MOVE R2, *R3
x = *yPtr;

fig_01_22