Instruction Sets:
Characteristics and Functions

What is an Instruction Set?

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

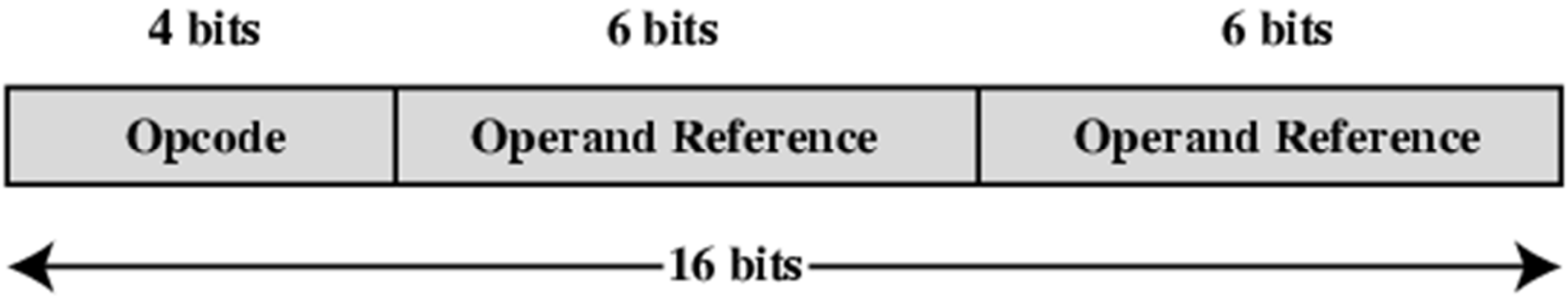
Elements of an Instruction

- Operation code (Op code)
  - —Do this
- Source Operand reference
  - —To this
- Result Operand reference
  - —Put the answer here
- Next Instruction Reference
  - —When you have done that, do this...

Instruction Representation

- In machine code each instruction has a unique bit pattern
- For human consumption (well, programmers anyway) a symbolic representation is used
  - e.g. ADD, SUB, LOAD
- Operands can also be represented in this way
  - ADD A,B

Simple Instruction Format



Instruction Types

- Data processing
- Data storage (main memory)
- Data movement (I/O)
- Program flow control

Number of Addresses (a)

- 3 addresses
  - —Operand 1, Operand 2, Result
  - —a = b + c;
  - — ADD a, b, c
  - —May be a forth - next instruction (usually implicit)
  - —Not common
  - —Needs very long words to hold everything

Number of Addresses (b)

- 2 addresses
  - —One address doubles as operand and result
  - —a = a + b
  - — ADD a, b
  - —Reduces length of instruction
  - —Requires some extra work
    - – Temporary storage to hold some results

Number of Addresses (c)

- 1 address
  - —Implicit second address
  - —Usually a register (accumulator)
  - —Common on early machines

Number of Addresses (d)

Instruction	Comment
SUB Y, A, B	Y ← A − B
MPY T, D, E	T ← D × E
ADD T, T, C	T ← T + C
DIV Y, Y, T	Y ← Y ÷ T

(a) Three-address instructions

Instruction	Comment
MOVE Y, A	Y ← A
SUB Y, B	Y ← Y − B
MOVE T, D	T ← D
MPY T, E	T ← T × E
ADD T, C	T ← T + C
DIV Y, T	Y ← Y ÷ T

(b) Two-address instructions

Instruction	Comment
LOAD D	AC ← D
MPY E	AC ← AC × E
ADD C	AC ← AC + C
STOR Y	Y ← AC
LOAD A	AC ← A
SUB B	AC ← AC − B
DIV Y	AC ← AC ÷ Y
STOR Y	Y ← AC

(c) One-address instructions

Figure 10.3 Programs to Execute $Y = \frac{A - B}{C + (D \times E)}$

Number of Addresses (e)

- 0 (zero) addresses
  - All addresses implicit
  - Uses a stack
  - e.g. push a
  - push b
  - add
  - pop c

  - c = a + b

How Many Addresses

- More addresses
  - More complex (powerful?) instructions
  - More registers
    - Inter-register operations are quicker
  - Fewer instructions per program
- Fewer addresses
  - Less complex (powerful?) instructions
  - More instructions per program
  - Faster fetch/execution of instructions

Design Decisions (1)

- Operation repertoire
  - How many ops?
  - What can they do?
  - How complex are they?
- Data types
- Instruction formats
  - Length of op code field
  - Number of addresses

Design Decisions (2)

- Registers
  - Number of CPU registers available
  - Which operations can be performed on which registers?
- Addressing modes

Types of Operand

- Addresses
- Numbers
  - —Integer/floating point
- Characters
  - —ASCII etc.
- Logical Data
  - —Bits or flags

Specific Data Types

- General - arbitrary binary contents
- Integer - single binary value
- Ordinal - unsigned integer
- Unpacked BCD - One digit per byte
- Packed BCD - 2 BCD digits per byte
- Near Pointer - offset within segment
- Bit field
- Byte String
- Floating Point

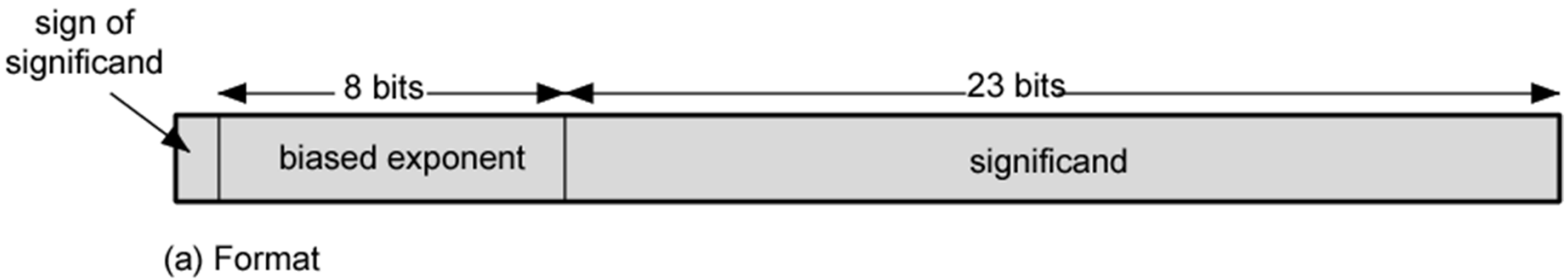
Integer Representation

- Only have 0 & 1 to represent everything
- Positive numbers stored in binary
  - e.g. 41=00101001
- Sign-Magnitude
- Two's compliment

Real Numbers

- Numbers with fractions
- Could be done in pure binary
  - $1001.1010 = 2^4 + 2^0 + 2^{-1} + 2^{-3} = 9.625$
- Where is the binary point?
- Fixed?
  - Very limited
- Moving?
  - How do you show where it is?

Floating Point



(a) Format

- +/- .significand x $2^{exponent}$
- Misnomer
- Point is actually fixed between sign bit and body of mantissa
- Exponent indicates place value (point position)
- IEEE 754 single format
- IEEE 754 double format (64bits = 1+11+52)

Types of Operation

- Data Transfer
- Arithmetic
- Logical
- Conversion
- I/O
- System Control
- Transfer of Control

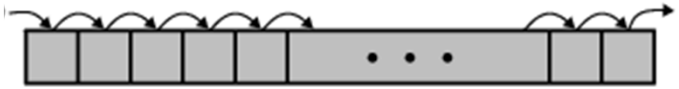
Data Transfer

- Specify
  - Source
  - Destination
  - Amount of data
- May be different instructions for different movements
  - e.g. IBM 370
- Or one instruction and different addresses
  - e.g. VAX

Arithmetic

- Add, Subtract, Multiply, Divide
- Signed Integer
- Floating point
- May include
  - Increment (a++)
  - Decrement (a--)
  - Negate (-a)

Shift and Rotate Operations

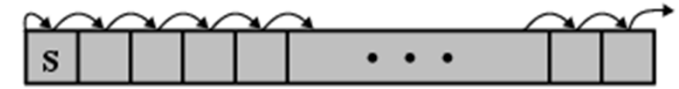


(a) Logical right shift

SHR

(b) Logical left shift

SHL

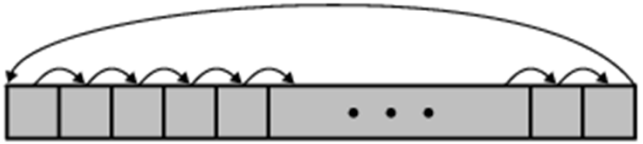


(c) Arithmetic right shift

ASR

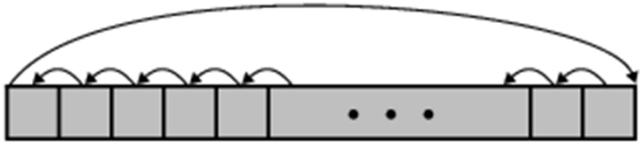
(d) Arithmetic left shift

ASL



(e) Right rotate

ROR



(f) Left rotate

ROL

Logical

- Bitwise operations
- AND, OR, NOT

Input/Output

- May be specific instructions
- May be done using data movement instructions (memory mapped)
- May be done by a separate controller (DMA)

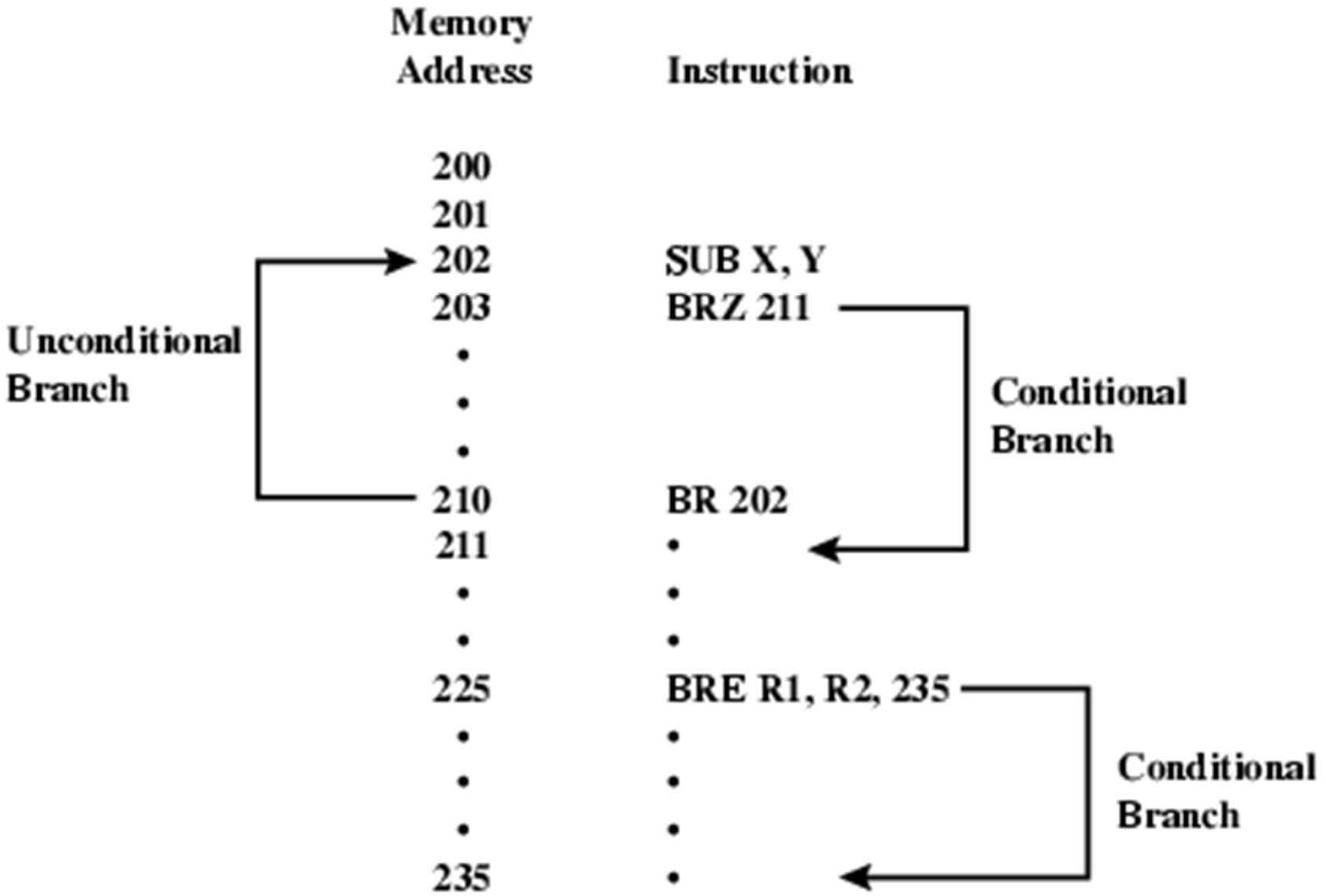
Systems Control

- Privileged instructions
- CPU needs to be in specific state
  - Kernel mode
- For operating systems use

Transfer of Control

- Branch
  - e.g. branch to x if result is zero
- Skip
  - e.g. increment and skip if zero
- Conditional Instruction
  - ISZ Register1
  - Branch xxxx
  - BNZ xxxx
  - BP xxxx
- Subroutine call
  - interrupt call

Branch Instruction



Nested Procedure Calls