William Stallings
Computer Organization
and Architecture
8th Edition

Chapter 10

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

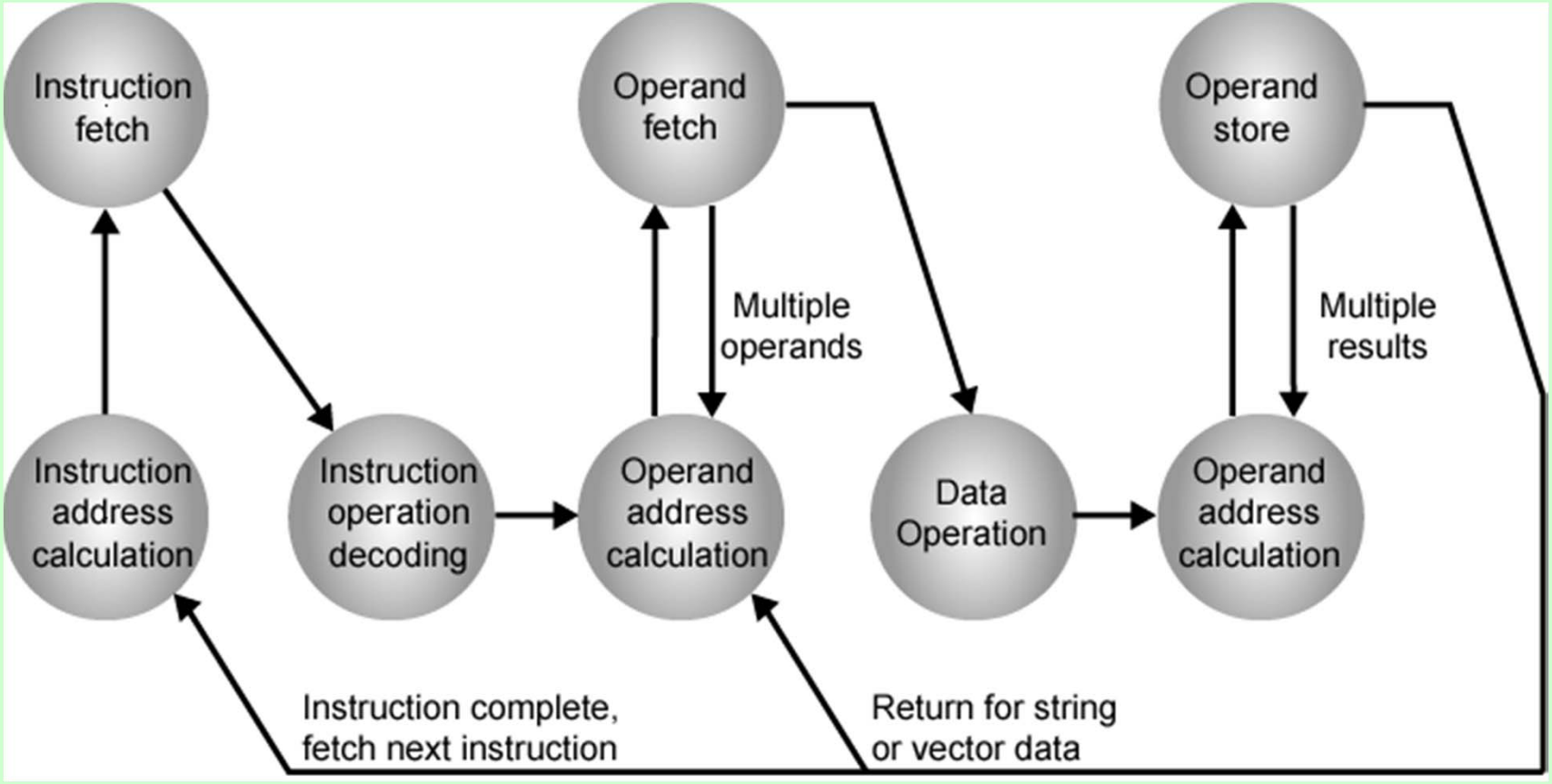
Elements of an Instruction

- 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...

Where have all the Operands Gone?

- Long time passing….
- (If you don’t understand, you’re too young!)
- Main memory (or virtual memory or cache)
- CPU register
- I/O device

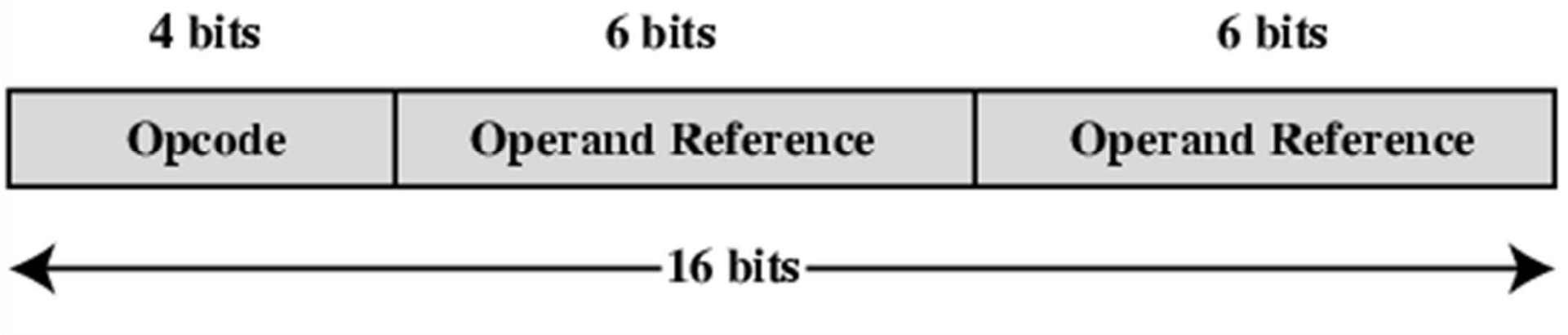
Instruction Cycle State Diagram



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;
  - May be a forth - next instruction (usually implicit)
  - Not common
  - Needs very long words to hold everything

Number of Addresses (a)

$$Y = \frac{A - B}{C + (D \times E)}$$

Instrução	Comentário

Number of Addresses (b)

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

Number of Addresses (b)

$$Y = \frac{A - B}{C + (D \times E)}$$

Instrução	Comentário

Number of Addresses (c)

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

Number of Addresses (c)

$$Y = \frac{A - B}{C + (D \times E)}$$

Instrução	Comentário

Number of Addresses (d)

- 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 (later...)
- RISC v CISC

Types of Operand

- Addresses
- Numbers
  - —Integer/floating point
- Characters
  - —ASCII etc.
- Logical Data
  - —Bits or flags
- (Aside: Is there any difference between numbers and characters? Ask a C programmer!)

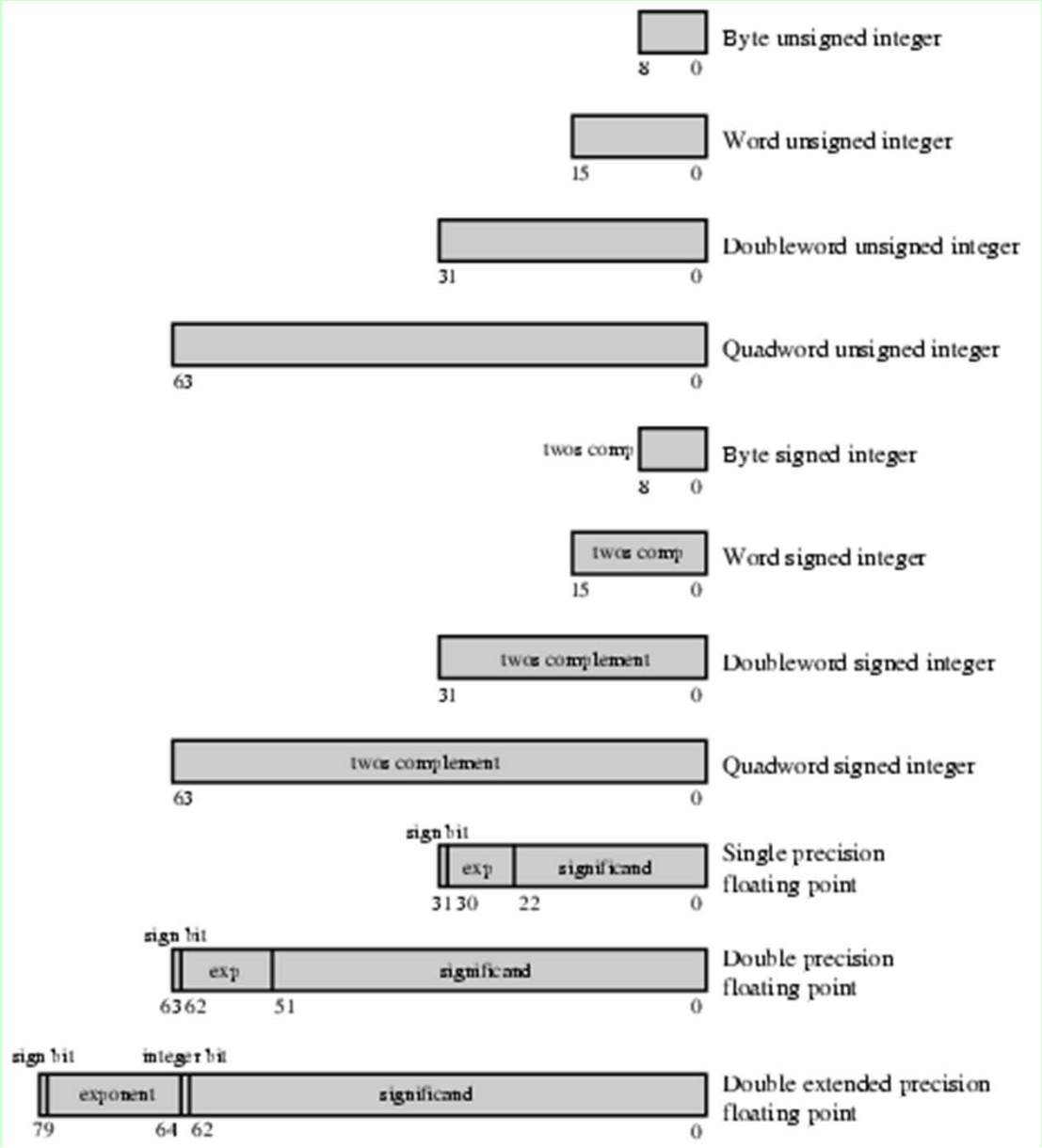
x86 Data Types

- 8 bit Byte
- 16 bit word
- 32 bit double word
- 64 bit quad word
- 128 bit double quadword
- Addressing is by 8 bit unit
- Words do not need to align at even-numbered address
- Data accessed across 32 bit bus in units of double word read at addresses divisible by 4
- Little endian

SMID Data Types

- Integer types
  - Interpreted as bit field or integer
- Packed byte and packed byte integer
  - Bytes packed into 64-bit quadword or 128-bit double quadword
- Packed word and packed word integer
  - 16-bit words packed into 64-bit quadword or 128-bit double quadword
- Packed doubleword and packed doubleword integer
  - 32-bit doublewords packed into 64-bit quadword or 128-bit double quadword
- Packed quadword and packed qaudword integer
  - Two 64-bit quadwords packed into 128-bit double quadword
- Packed single-precision floating-point and packed double-precision floating-point
  - Four 32-bit floating-point or two 64-bit floating-point values packed into a 128-bit double quadword

x86 Numeric Data Formats



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)

Logical

- Bitwise operations
- AND, OR, NOT

Conversion

- E.g. Binary to Decimal

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
  - Ring 0 on 80386+
  - 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
  - ISZ Register1
  - Branch xxxx
  - ADD A
- Subroutine call
  - c.f. interrupt call