Instruction Set Architecture

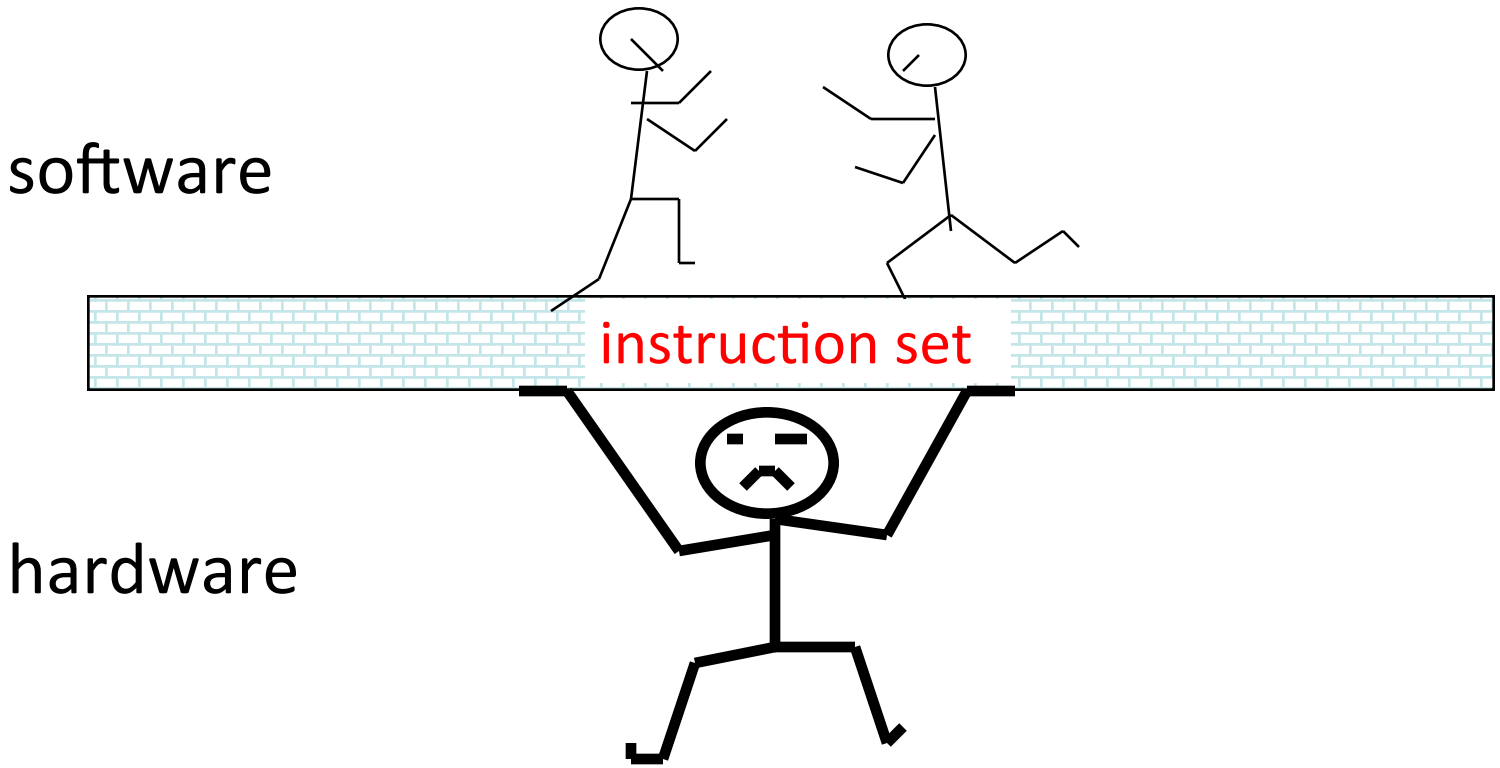
Virendra Singh
Computer Architecture and Dependable Systems Lab
Department of Electrical Engineering
Indian Institute of Technology Bombay
http://www.ee.iitb.ac.in/~viren/
E-mail: viren@ee.iitb.ac.in

EE-309: Microprocessors

Lecture 18 (27 Aug 2015)

CADSL

Instruction Set Architecture (ISA)



ISA-level Tradeoffs: Semantic Gap

- Where to place the ISA? Semantic gap
  - Closer to high-level language (HLL) → Small semantic gap, complex instructions
  - Closer to hardware control signals? → Large semantic gap, simple instructions

- RISC vs. CISC machines
  - RISC: Reduced instruction set computer
  - CISC: Complex instruction set computer
    - FFT, QUICKSORT, POLY, FP instructions?
    - VAX INDEX instruction (array access with bounds checking)

ISA-level Tradeoffs: Semantic Gap

- Simple compiler, complex hardware vs. complex compiler, simple hardware
  - Caveat: Translation (indirection) can change the tradeoff!

- Burden of backward compatibility

- Performance?
  - Optimization opportunity: Example of VAX INDEX instruction: who (compiler vs. hardware) puts more effort into optimization?
  - Instruction size, code size

Small versus Large Semantic Gap

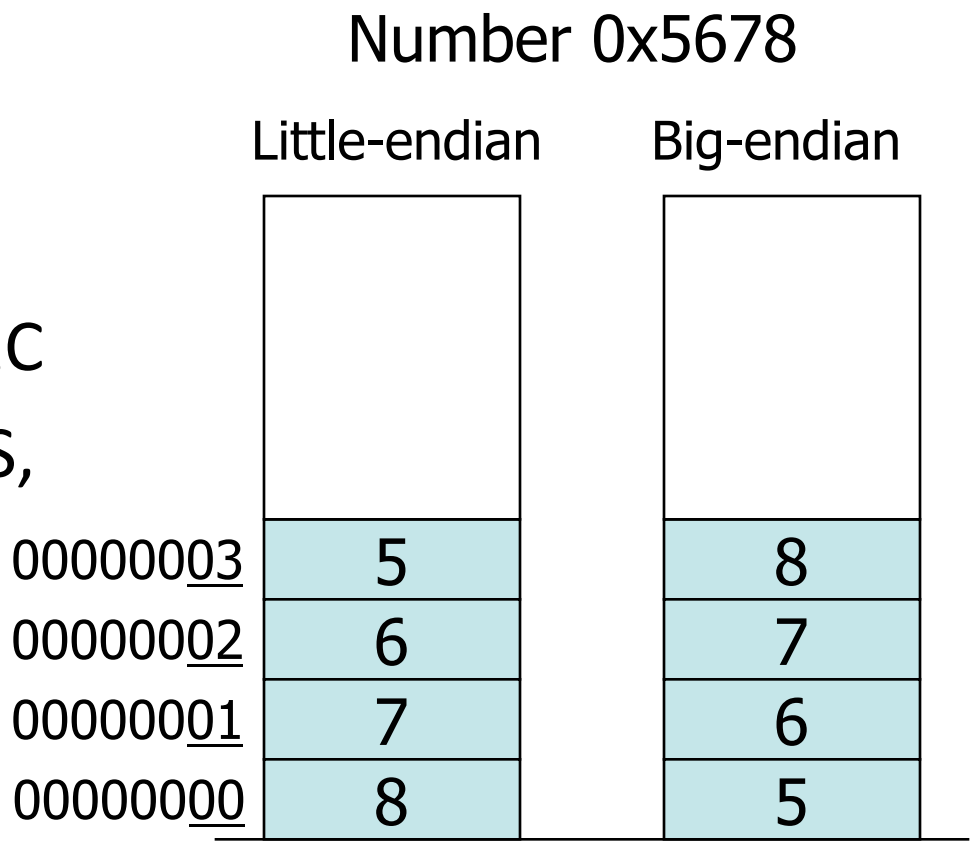
- CISC vs. RISC
  - Complex instruction set computer → complex instructions
    - Initially motivated by compact code generation
  - Reduced instruction set computer → simple instructions
    - John Cocke, mid 1970s, IBM 801
      - Goal: enable better compiler control and optimization
- RISC motivated by
  - Memory stalls (no work done in a complex instruction when there is a memory *stall*?)
  - Simplifying the hardware → lower cost, higher frequency
  - Enabling the compiler to optimize the code better

Memory Address

- Interpreting memory address
  - Big Endian
  - Little Endian
- Instruction misalignment
- Addressing mode

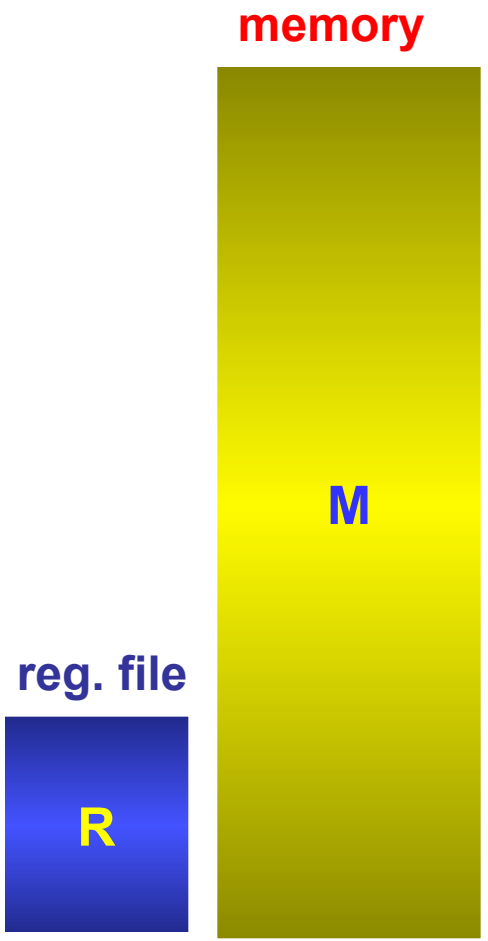
Little or Big: Where to Start?

- Byte ordering: Where is the first byte?
- Big-endian : IBM, SPARC, Mororola
- Little-endian: Intel, DEC
- Supporting both: MIPS, PowerPC



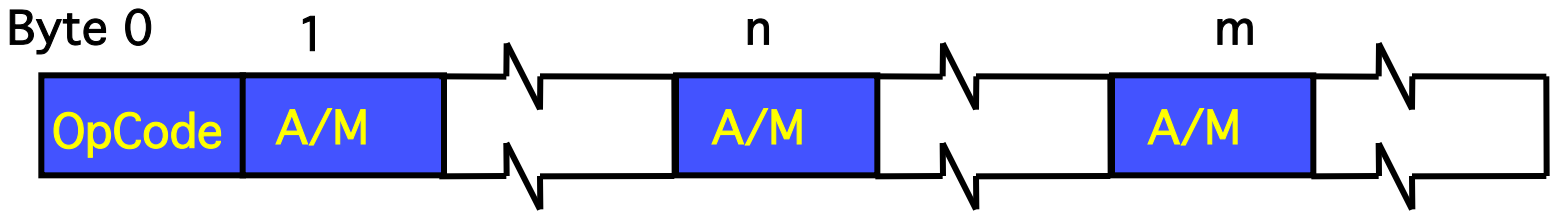
Kinds of Addressing Modes

Addressing Mode	value in [ ] is the operand
• Register direct	[Ri]
• Immediate (literal)	v
• Direct (absolute)	M[v]
• Register indirect	M[[Ri]]
• Base+Displacement	M[[Ri] + v]
• Base+Index	M[[Ri] + [Rj]]
• Scaled Index	M[[Ri] + [Rj]*d + v], e.g. d=8
• Autoincrement	M[[Ri]+1]
• Autodecrement	M[[Ri] - 1]
• Memory Indirect	M[M[Ri]]



VAX-11

Variable format, 2- and 3-address instructions



- 32-bit word size, 16 GPR (4 reserved)
- Rich set of addressing modes (apply to any operand)
- Rich set of operations
  - bit field, stack, call, case, loop, string, poly, system
- Rich set of data types
- Condition codes

Thank You