Lecture 7: Instruction Set Architectures - IV

- Announcements:
  - Readings for today: 3.2 (int), 3.6 (float), B.9 (memory)
  - EXTENSION ON HW #2 – Now due Feb 20.
- Last Time
  - Graphics processor ISA
  - Naming Storage
  - Various kinds of register storage
  - Alignment
- Today
  - Quick review
  - Memory addressing (cont'd)

Last Lecture - Naming Storage in ISAs

- Memory
  - Addresses in instruction
  - Addresses computed by instructions
- General Registers
  - Operands to instructions
- Special registers
  - Status, condition codes, floating-point codes
  - Operands to special instructions

Last Lecture – ISA Variety

- Graphics processor instruction set
- Various register organizations:
  - Accumulator, Index, General Registers, Stack machines

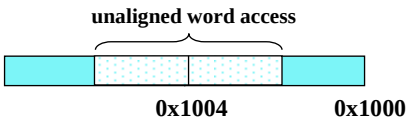
Memory Organization

Memory Organization

- Four components specified by ISA:
  - Smallest addressable unit of memory (byte? halfword? word?)
  - Maximum addressable units of memory (doubleword?)
  - Alignment
  - Endianness
- Already talked about addressing modes last time

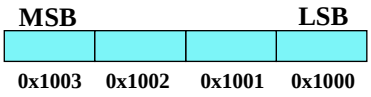
Alignment

- Some architectures restrict addresses that can be used for particular size data transfers!
  - Bytes accessed at any address
  - Halfwords only at even addresses
  - Words accessed only at multiples of 4

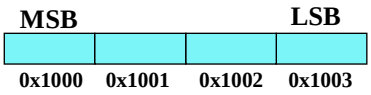


Endianness

- How are bytes ordered within a word?
  - Little Endian (Intel/DEC)



  - Big Endian (IBM/Motorola)

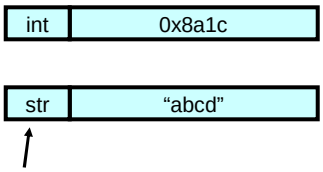


  - Today - most machines can do either (configuration register)

Data Types

Data Types

- How the contents of memory and registers are interpreted
- Can be identified by
  - Tag (data itself)
  - Use (instructions)
- Driven by application
  - Signal processing
    - 16-bit fixed point (fraction)
  - Text processing
    - 8-bit characters
  - Scientific computing
    - 64-bit floating point
- Most *general purpose* computers support several types
  - 8, 16, 32, 64-bit
  - signed and unsigned
  - fixed and floating



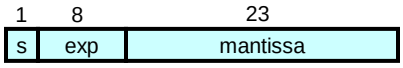
Examples of tags (ie. Symbolics machine)

Example: 32-bit Floating Point

- Type specifies mapping from bits to real numbers (plus symbols)
  - format
    - S, 8-bit exp, 23-bit mantissa
  - interpretation
    - mapping from bits to abstract set

$$v = (-1)^S \times 2^{(E-127)} \times 1.M$$

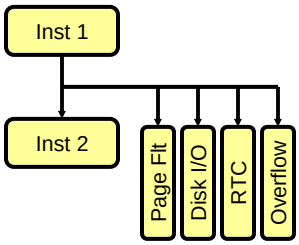
  - operations
    - add, mult, sub, sqrt, div



Exceptions

Control - Exceptions/Events

- Implied branch after every instruction
  - Internal events
    (called faults or exceptions)
    - arithmetic overflow
    - page fault
  - External events
    (called interrupts)
    - completion of I/O operations

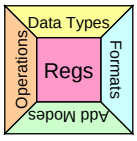


- What happens????
  - Save PC (in special register)
  - Jump to exception address
    (taken from table)
  - When done: Jump to original PC + 4

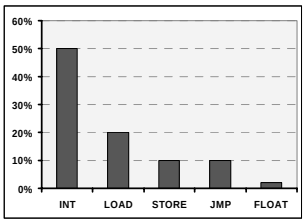
General ISA design principles

Principles of Instruction Set Design

- Keep it simple (KISS)
  - complexity
    - increases logic area
    - increases pipe stages
    - increases development time
  - evolution tends to make kludges
- Orthogonality (modularity)
  - simple rules, few exceptions
  - all ops on all registers

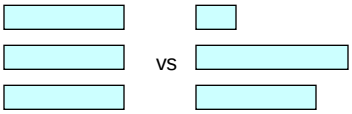


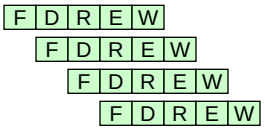
- Frequency
  - make the common case fast
    - some instructions (cases) are more important than others



Principles of Instruction Set Design (part 2)

- Generality
  - not all problems need the same features/instructions
  - principle of *least surprise*
  - performance should be easy to predict
- Cost effectiveness
  - design ISA to permit efficient implementation
    - today
    - 10 years from now





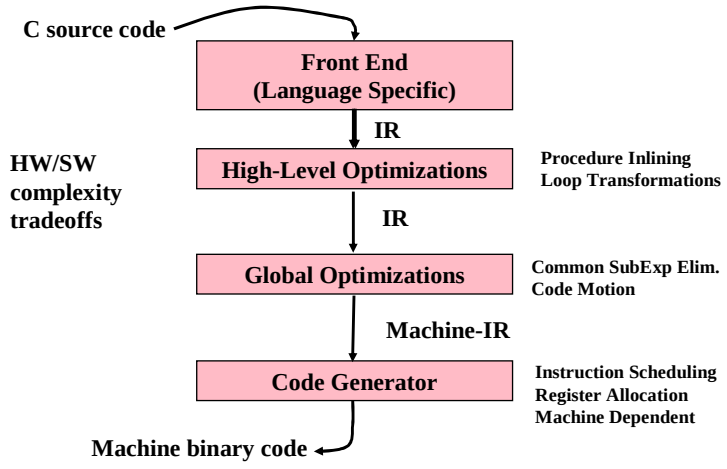
CISC vs RISC

- What is a RISC?
  (Reduced Instruction Set Computer)
  - no firm definition
  - generally includes
    - general registers
    - fixed 3-address instruction format
    - strict load-store architecture
    - simple addressing modes
    - simple instructions
  - Examples
    - DEC Alpha
    - MIPS
  - Advantages
    - good compiler target
    - easy to implement/pipeline
- CISC (Complex Instruction-Set Computer)
  - CISC ≡ ¬RISC
  - may include
    - variable length instructions
    - memory-register instructions
    - complex addressing modes
    - complex instructions
      - CALLP, EDIT, …
  - Examples
    - DEC VAX, IBM 370, x86
  - Advantages
    - better code density
    - legacy software

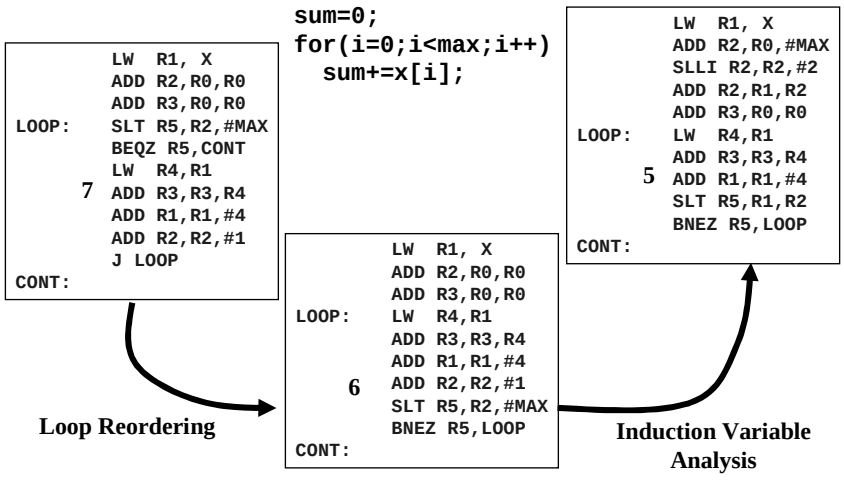
Compilers

Performance = application +
compiler +
hardware

Role of the Optimizing Compiler



Example: Loop Optimization



Architect can help Compiler Writer

- Simplify, Simplify, Simplify
  - Feature difficult to use, it won't be used….Less is More!
- Regularity
  - Common set of formats, few special cases
- Primitive, not solutions
  - CALLS vs. Fast register moves
- Make performance tradeoffs simple
- Ultimately, the ISA will *not* be perfect

Compiler can help Microarchitecture

- Instruction Scheduling
  - Instruction Level Parallelism
- Resource Allocation
  - Registers (minimize spills/restores to and from memory)
- Memory optimizations
  - Cache conscious data organization
  - Code layout
- Etc……

Summary of first part of lecture

- ISA principles
- Compiler/ISA interaction
- Next Time
  - Simple processor datapath
  - Reading assignment – P&H 5.1-5.4

On blackboard:
Begin discussing implementation of MIPS architecture

1) What parts are available to us?
2) What does the machine need to do?
   (For three major types of instruction)