Welcome to the Class

Department of Computing and Information System

Computer Organization & Architecture

What Is An ISA?

- ISA (instruction set architecture)
  - A well-define hardware/software interface
  - The **"contract"** between software and hardware
    - **Functional definition** of operations, modes, and storage locations supported by hardware
    - **Precise description** of how to invoke, and access them
  - No guarantees regarding
    - How operations are implemented
    - Which operations are fast and which are slow and when
    - Which operations take more power and which take less

What Makes a Good ISA?

- **Programmability**
  - Easy to express programs efficiently?
- **Implementability**
  - Easy to design high-performance implementations?
  - More recently
    - Easy to design low-power implementations?
    - Easy to design high-reliability implementations?
    - Easy to design low-cost implementations?
- **Compatibility**
  - Easy to maintain programmability (implementability) as languages and programs (technology) evolves?
  - x86 (IA32) generations: 8086, 286, 386, 486, Pentium, PentiumII, PentiumIII, Pentium4,...

How Many Registers?

- Registers faster than memory, have as many as possible?
  - **No**
  - One reason registers are faster is that there are **fewer of them**
    - Small is fast (hardware truism)
  - Another is that they are **directly addressed** (no address calc)
    - More of them, means larger specifiers
    - Fewer registers per instruction or indirect addressing
  - **Not everything can be put in registers**
    - Structures, arrays, anything pointed-to
    - Although compilers are getting better at putting more things in
  - More registers means **more saving/restoring**

  - Upshot: trend to more registers: 8 (x86)→32 (MIPS) →128 (IA32)
    - 64-bit x86 has 16 64-bit integer and 16 128-bit FP registers

Significance of ISA

- An **instruction set architecture** (**ISA**) is an abstract model of a computer. It is also referred to as **architecture** or **computer architecture**.
- A realization of an ISA is called an *implementation*. An ISA permits multiple implementations that may vary in performance, physical size, and monetary cost (among other things); because the ISA serves as the interface between software and hardware.
- Software that has been written for an ISA can run on different implementations of the same ISA. This has enabled binary compatibility between different generations of computers to be easily achieved, and the development of computer families.
- For these reasons, the ISA is one of the most important abstractions

Instruction Set

An **instruction set** is a group of commands for a CPU in machine language. All CPUs have **instruction sets** that enable commands to the processor directing the CPU to switch the relevant transistors. Some **instructions** are simple read, write and move commands that direct data to different hardware.

Types of Instruction Set

- **Following are the instruction set architectures:**

- **Reduced Instruction Set Computer** (**RISC**)
- **Complex Instruction Set Computer** (**CISC**)
- Minimal instruction set computers (MISC)
- Very long instruction word (VLIW)
- Explicitly parallel instruction computing (EPIC)
- One instruction set computer (OISC)

Classifying Instruction Set

- **CISC = Complex Instruction Set Computer**
  - During the early years, memory was slow and expensive and the programming was done in assembly language.
  - Since memory was slow and instructions could by retrieved up to 10 times faster from a local ROM than from main memory
  - Programmers tried to put as many instructions as possible in a microcode.

Classifying Instruction Set

- **CISC = Complex Instruction Set Computer**
  - Large number of instruction format in instruction set architecture (ISA)
  - Large number of addressing schemes and instruction types
  - Varying size of instructions
  - Multiple cycle instruction execution

Classifying Instruction Set

- **RISC = Reduced Instruction Set Computer**
  - Earlier, computers used only 20% of the instructions. Making the other 80% unnecessary.
  - Reduce instruction set so they can execute their instructions very fast because the instructions are so simple.
  - RISC chips require fewer transistors, which makes them cheaper to design and produce.

Classifying Instruction Set

- **RISC = Reduced Instruction Set Computer**
  - Small number of fast instruction format
  - Fixed instruction format, fixed length
  - Fast and single cycle instruction execution
  - Large number of general purpose registers
  - Typical for load/store architectures

Advantages (RISC)

- **Speed**
  - Achieve 2 to 4 times the performance of CISC processor
- **Simpler Hardware**
  - Uses up much less chip space
  - Extra functions, such as memory management units or floating point arithmetic units, can also be placed on the same chip
  - can lower the per-chip cost dramatically
- **Shorter Design Cycle**
  - Can take advantage of other technological developments sooner than corresponding CISC designs, leading to greater leaps in performance between generations.

RISC Vs CISC

- CISC (complex instruction set computer)
  - VAX, Intel X86, Motorolla 680X0, etc
- RISC (reduced instruction set computer)
  - MIPS, DEC Alpha, SUN Sparc, IBM 801,ARM6, etc

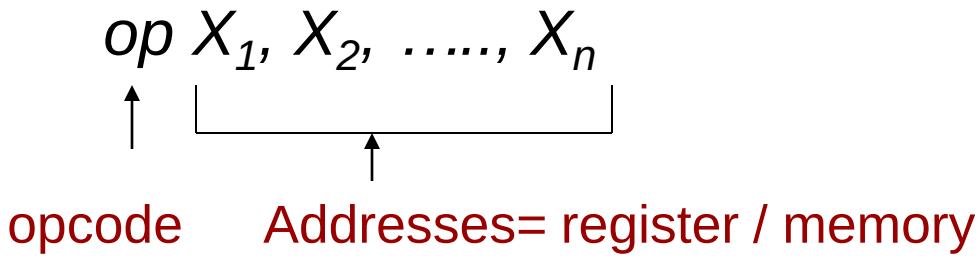
RISC Vs CISC

CISC	RISC
Variable length instruction	Single word instruction
Large number of format	Small number of format
Memory operands	Load/store architecture
Complex operations	Simple operations

Instruction Format

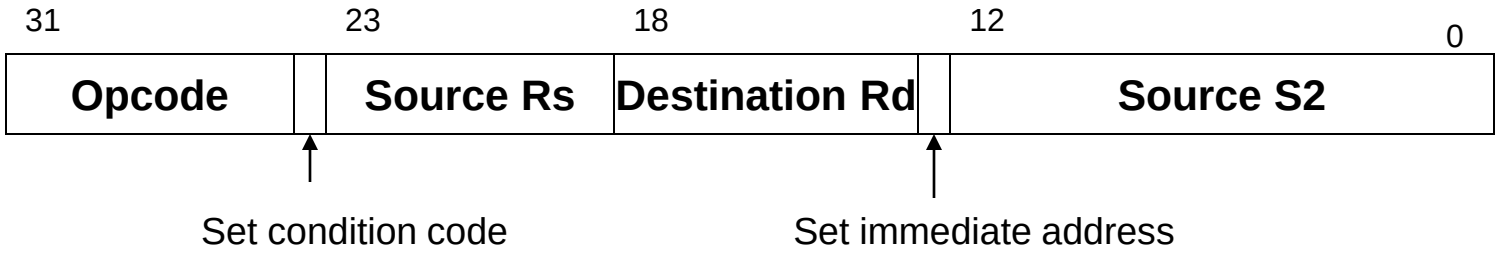
An instruction specifies

- the operation to be carried out by the processor.
- the set of operands to be used in the operation. It includes both the input data of the operation and the result that are produced.
- In assembly language notation, an instruction format can be written as



Instruction Format for RISC1

As memory addressing is restricted to load and store instructions, so the operands of most instructions are register addresses, Which are short and easy to accommodate in a one-word format.



The register-to-register operation form

Rd:=F(Rs, S2)

Rd – Destination register, Rs – First source register, S2 – Second source register

If bit 13 = 1, then S2 is interpreted as an immediate address

Operand Extension

- A CPU is designed to process data words and addresses of one specific length.
- Instruction contains operands fields that are shorter than standard word size.
- This problem is unavoidable in RISC instructions.
- Two methods to extend short operand values to full-size, signed or unsigned numbers:
  1. Sign Extension
  2. Zero Extension

Address Extension

- Suppose memory address is n-bit, we have to construct it from a short m-bit address field, where n > m.
- Zero extension does not allow m-bit address to refer to all 2^n possible addresses.
- The solution used is to treat the short memory address as a modifier. It is added (in zero-extended form) to a full length memory address stored in a CPU register, known as base register.
- This solution is used in both CISC and RISC processor.

Any Question???