Instruction Set Architecture (ISA) - Part III

CS6133 - Computer Architecture I

Vikram Padman

NYU Polytechnic School of Engineering

vikram@poly.edu

Agenda

1. **Introduction** - Part I
2. **Classifying ISA** - Part I, II, III and IV
   - Operations in the instruction set - Part III
   - Instruction for control flow - Part III
3. **Activity**

Reading List

- "Computer Architecture - A Quantitative Approach" - Appendix A in Fifth Edition or Appendix B in Fourth Edition
- "Computer Organization and Design" - Chapter 2 in Fourth Edition or Third Edition
- "Digital Design and Computer Architecture" - Chapter 6

Operations in the Instruction Set

CPU Instructions could be categorized into the following five categories:

1. Computational
   1. Arithmetic
   2. Logical
   3. Compare
   4. Bit manipulation
2. Data transfer
3. Control
4. System
5. Specialized or Application Specific

Operations in the Instruction Set

Computational Instructions

Computational instructions are the fundamental instructions used to perform operation on data. Most, if not all, CPUs supports computational instructions.

1. **Arithmetic** - Perform calculation such as add, subtract, multiply, divide. Most modern CPUs have hardware support to perform calculation on floating point and decimal numbers.
2. **Logical** - Perform logical operation such as and, xor, or, not on data
3. **Compare** - Are instructions used to compare result of an arithmetic operation or contents of two register and set or unset a destination register. Example are Set equal (seq), Set Greater (sgt), Set Less than (slt), Set not equal (sne)
4. **Bit manipulation** - Used to shift or rotate the contents of a register right or left. Examples are: Shift left logical (sll), Shift right logical (srl), Rotate right (rol), Rotate right (ror)

Operations in the Instruction Set
Data Transfer Instructions

Data Transfer Instructions are used to copy data from CPU's internal registers to memory and vice-versa. Some CPU's support conditional data transfer instructions to only copy data when a specific condition is met. Examples of data transfer instructions are:

- **Load** - lw r1, [8] Copies a word (32 bits) from memory location 8 into CPU register r1
- **Store** - sw r1, [8] Copies a word (32 bits) from CPU register r1 into memory location 8
- **CMOVE** - cmove [8],r2 is an x86 instruction that copies memory location 8 into r2 if the result of a previous compare instruction was 0.

Operations in the Instruction Set

Control Instructions

Control instructions are used to change the order of instructions that are being executed by the CPU. A control instruction could be conditional or unconditional. Examples of control instructions are:

- **Jump** - j [8] unconditionally jump to memory location 8
- **Jump and Link** - jal [8] unconditionally jump to memory location 8 and store the return address in register ra
- **Branch** - b 8 unconditionally jump 8 instruction from the current position [1]
- **Branch and Link** bal 8 unconditionally jump 8 instruction from the current position and store the return address in register ra
- **Branch on Zero** - bnez r1, 8 jump 8 instruction from the current position if r1 = 0

[1] Note: The difference between jump and branch instructions are not consistent across all CPU architectures. The above is true on a MIPS CPU, but not on a x86 or 68K CPUs. Some CPUs only have jump instructions and they could be used in pc-relative or direct mode.

Operations in the Instruction Set

System Instructions

System Instructions are privileged instruction that are used by a multi-tasking operating system to maintain the state of a system, control and isolate processes, manage I/O devices and virtual memory. Example of system Instructions are:

- **LGDT** - Privileged x86 instruction used to load Global Descriptor Table(GDT).
- **LLDT** - Privileged x86 instruction used to load Local Descriptor Table(LDT).

A descriptor table contains memory maps and associated privilege levels.

Operations in the Instruction Set
Specialized or Application Specific Instructions

Specialized or Application Specific Instructions are used by scientific or graphics intensive applications. Many modern CPU's support many specialized instructions and execute them in a built-in co-processing modules. For example; MMX, SSE, 3Dnow are some popular extensions supported by x86 processor.

Operations in the Instruction Set

Top 10 x86 instructions

Rank	80x86 instruction	Integer average (% total executed)
1	load	22%
2	conditional branch	20%
3	compare	16%
4	store	12%
5	add	8%
6	and	6%
7	sub	5%
8	move register-register	4%
9	call	1%
10	return	1%
Total		96%

From "Computer Architecture – A Quantitative Approach" page A-16

Instruction for Control Flow

- Control Flow instruction are independent from the other types of operations presented in the previous section and could be categorized into:
  1. Jumps - Unconditionally goto an address or pc-relative address
  2. Conditional Branch - Goto to an address or pc-relative address when some condition is met
  3. Procedure call/return - Goto to an address while saving the current instructions location in some register.

Instruction for Control Flow
Evaluating Branch Conditions

There are essentially three popular way to evaluate a branch condition:

Name	Examples	How condition is tested	Advantages	Disadvantages
Condition code (CC)	80x86, ARM, PowerPC, SPARC, SuperH	Tests special bits set by ALU operations, possibly under program control.	Sometimes condition is set for free.	CC is extra state. Condition codes constrain the ordering of instructions since they pass information from one instruction to a branch.
Condition register	Alpha, MIPS	Tests arbitrary register with the result of a comparison.	Simple.	Uses up a register.
Compare and branch	PA-RISC, VAX	Compare is part of the branch. Often compare is limited to subset.	One instruction rather than two for a branch.	May be too much work per instruction for pipelined execution.

From "Computer Architecture – A Quantitative Approach" page A-19

Week 5 - Activity 3

Examine chapter 2, chapter 4 (page 67 - 73) and Appendix A in MIPS R4000's User's Manual[2], read Appendix J[3] and answer the following questions:

1. List and describe system and procedure call/return instruction supported by R4000 CPU.
2. How does MIPS CPU protect jump or branch into privileged/kernel memory?
3. Assume that R4000 does not have a floating point co-processor. Could a developer write an application that requires floating point data types? If yes, which instruction could she use and how?

[2] http://groups.csail.mit.edu/cag/raw/documents/R4400_Uman_book_Ed2.pdf

[3] http://booksite.mkp.com/9780123838728/references/appendix_j.pdf