CS 203A
Advanced Computer Architecture

Lecture 2

Instruction Set Principles

Instructor: Jun Yang

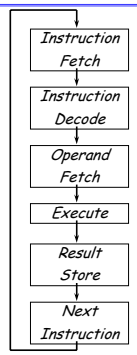
Outline

- ISA classification
- Address Modes
- Operand type and size
- Types of operations
- Non-data related program control instructions (Branches)

Introduction

- Desktop computing emphasizes on price-performance.
  - Code size is hardly considered
- Servers—Database, file server, Web appl. time-sharing appl.
  - treat integer op.'s and character string op.'s more important than FP op.'s.
- Embedded applications value cost and power.
  - Code size and low power are both important.
  - Some classes of instructions are optional to reduce chip cost.
  - Real-time performance requirement often means to guarantee worst-case perf.
  - Hand-optimized code exists for critical kernels. Not generate-able by any compiler.
- Current trend for desktops/servers is to add media instr. and the embedded processors are beginning to run more general-purpose code.

Instruction Set Architecture (ISA)



What must be specified?

- **Instruction Format or Encoding**
  - How is it decoded?
- **Location of operands and result**
  - Where other than memory?
  - How many explicit operands?
  - How are memory operands located?
- **Data type and Size**
- **Operations**
  - What are supported
- **Control instruction**
  - Jumps, conditions, branches

ISA Classification

Most real machines are hybrids of these:

Stack:

0 address	`add`	**tos ← tos + next**

Accumulator (1 register):

1 address	`add A`	**acc ← acc + mem[A]**
1+x address	`addx A`	**acc ← acc + mem[A + x]**

Register-Memory:

2 address	`add A B`	**A ← A + B (mem[B])**
3 address	`add A B C`	**A ← B + C (mem[B/C])**

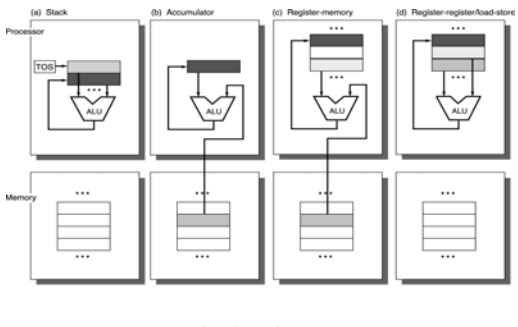
Load/Store (register-register):

3 address	`add Ra Rb Rc`	**Ra ← Rb + Rc**
	`load Ra Rb`	**Ra ← mem[Rb]**
	`store Ra Rb`	**mem[Rb] ← Ra**

Comparison:

Bytes per instruction? Number of Instructions? Cycles per instruction?

ISA Classification



Comparing Number of Instructions

Code sequence for (C = A + B) for four classes of instruction sets:

Stack	Accumulator	Register (register-memory)	Register (load-store)
Push A	Load A	Load R1,A	Load R1,A
Push B	Add B	Add R1,B	Load R2,B
Add	Store C	Store C, R1	Add R3,R1,R2
Pop C			Store C,R3

General Purpose Registers (GPR) Dominate

- After 1980, almost all machines use general purpose registers
- Advantages of registers
  - Registers are faster than memory
  - Registers are easier for a compiler to use
    - E.g., (A*B) – (C*D) – (E*F) can do multiplies in any order vs. stack
  - Registers can hold variables
    - Memory traffic is reduced, so program is sped up (since registers are faster than memory)
  - Code density improves (since register named with fewer bits than memory location)

Two characteristics of GPR architectures

Number of memory addresses	Maximum number of operands allowed	Type of architecture	Examples
0	3	Register-register	Alpha, ARM, MIPS PowerPC, SPARC, Trimedia TM5200
1	2	Register-memory	IBM 360/370, Intel 80x86, Motorola 68000
2	2	Memory-memory	VAX
3	3	Memory-memory	VAX

Advantages ? Disadvantages? (Read Figure 2.4 on Pg95!)

Outline

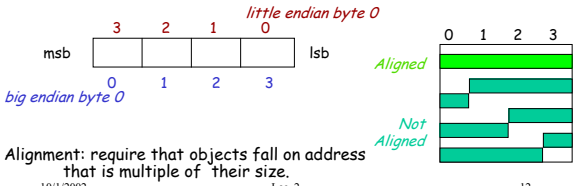
- ISA classification
- **Address Modes**
- Operand type and size
- Types of operations
- Non-data related program control instructions (Branches)

Memory Addressing

- Since 1980, almost every machine uses addresses to level of 8-bits (byte)
- Two questions for design of ISA:
  - Since one could read a 32-bit word as four loads of bytes from sequential byte address of as one load word from a single byte address, how do byte addresses map onto words?
  - Can a word be placed on any byte boundary?

Addressing Objects: Endianess and Alignment

- Big Endian: address of most significant byte = word address (xx00 = Big End of word)
  - IBM 360/370, Motorola 68k, MIPS, Sparc, HP PA
- Little Endian: address of least significant byte = word address (xx00 = Little End of word)
  - Intel 80x86, DEC Vax, DEC Alpha (Windows NT)



Alignment: require that objects fall on address that is multiple of their size.

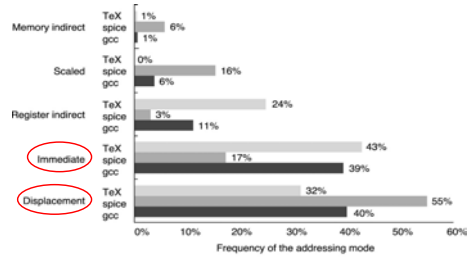
Addressing Modes

Addressing mode	Example	Meaning
Register	Add R4,R3	R4← R4+R3
Immediate	Add R4,#3	R4 ← R4+3
Displacement	Add R4,100(R1)	R4 ← R4+Mem[100+R1]
Register indirect	Add R4,(R1)	R4 ← R4+Mem[R1]
Indexed / Base	Add R3,(R1+R2)	R3 ← R3+Mem[R1+R2]
Direct or absolute	Add R1,(1001)	R1 ← R1+Mem[1001]
Memory indirect	Add R1,@(R3)	R1 ← R1+Mem[Mem[R3]]
Post-increment	Add R1,(R2)+	R1 ← R1+Mem[R2]; R2 ← R2+d
Pre-decrement	Add R1,-(R2)	R2 ← R2-d; R1 ← R1+Mem[R2]
Scaled	Add R1,100(R2)[R3]	R1 ← R1+Mem[100+R2+R3*d]

Addressing Modes for Signal Processing

- Besides Immediate, Displacement, Register indirect and Direct modes, DSP supports instructions that perform circular buffer access and Fast Fourier Transformation (FFT).
  - Due to frequency of these types of operations!

Usage of Addressing Mode



Addressing Mode Summary

- Data addressing modes that are important:
  - Register-register, displacement, immediate.
- Displacement size should be 12 to 16 bits
  - This range account for 75% to 99% of the displacement.
- Immediate size should be 8 to 16 bits
  - This range captures 50% to 80% of the immediates.

Outline

- ISA classification
- Address Modes
- **Operand type and size**
- Types of operations
- Non-data related program control instructions (Branches)

Type and Size of Operands

- Encoding of Type Information
  - In the operand itself: tags in LISP machines.
  - In the instruction: as with all normal machines.
- Common Types
  - character, half-word, integer, single-precision double-precision fp.
  - Until 1980s fp format was manufacturer specific, now IEEE Floating-Point Standard 754.
- Other Types
  - binary-coded decimals (4 bits representing 0-9) and packed decimals (2 decimals per byte).
- Operands for Media and Signal Processing
  - Vertex (x, y, z-coordinates) and Pixels (RGBA) - 32 bits.
  - Fixed point data types vs. floating point data in DSP.

Outline

- ISA classification
- Address Modes
- Operand type and size
- **Types of operations**
- Non-data related program control instructions (Branches)

Typical Operations (little change since 1960)

Data Movement	Load (from memory) Store (to memory) memory-to-memory move register-to-register move input (from I/O device) output (to I/O device) push, pop (to/from stack)
Arithmetic	integer (binary + decimal) or FP Add, Subtract, Multiply, Divide
Shift	shift left/right, rotate left/right
Logical	not, and, or, set, clear
Control (Jump/Branch)	unconditional, conditional
Subroutine Linkage	call, return
Interrupt	trap, return
Synchronization	test & set (atomic r-m-w)
String	search, translate
Graphics (MMX)	parallel subword ops (4 16bit add)

Top 10 80x86 Instructions

° **Rank instruction Integer Average Percent 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%**

° **Simple instructions dominate instruction frequency**

Outline

- ISA classification
- Address Modes
- Operand type and size
- Types of operations
- **Non-data related program control instructions (Branches)**