Evolution of ISA's

- ISA's have changed over computer "generations".
- A traditional way to look at ISA complexity encompasses:
  - Number of addresses per instruction
  - Regularity/size of instruction formats
  - Number of addressing types

Number of addresses per instruction

- First computer: 1 memory address + implied accumulator
  - Names like EMIAC, EDVAC, SEAC, MANIAC (AC=accumulator)
- Then 1 memory address + "index" registers (for addressing operands)
- Followed by 1 memory address + "general registers" (for addressing and storing operands)
  - IBM System/360 and followers
- Then 2 or 3 memory addresses + general registers
- Then N memory addresses + general registers
  - DEC PDP-11 and VAX
- Also "0-address" computers, or "stack computers"
- In which category is MIPS, and more generally RISC machines?

Addresses per instruction

- RISC machines
  - Load-store and branches: 1 memory address + 2 registers
  - All other 3 registers or 2 registers + immediate
- CISC machines
  - Most of them: Two addresses
    (destination ← source op. destination)
  - One operand is a register; other is either register, immediate, or given by memory address
  - Some special instructions (string manipulation) can have two memory addresses

Regularity of instruction formats

- Started with fixed format (ease of programming in "machine language"; few instructions)
- Then more flexibility (assembler/compiler): three or four instruction formats, not necessarily the same length
- Then strive for memory compactness. Complex, powerful, variable length instructions (x86)
- Back to regular instruction sets: few formats, instructions of the same length (memory is cheap; instructions must be decoded fast)

Number of instruction formats

- RISC: three or four (instructions have same length)
- CISC
  - Several formats, each of fixed (but maybe different) length
  - Variable length instructions (depends on opcode, addressing of operands etc. Intel x86 instructions from 1 to 17 bytes)
    - Instruction encoding via “specifiers”

Addressing modes

- In early machines: immediate, direct, indirect
- Then index registers
- Then index + base (sum of 2 registers instead of -- or in addition to -- index +displacement)
- All kinds of additional modes (indirect addressing, auto-increment, combinations of the above etc.)
- In general RISC
  - Immediate, indexed, and sometimes index + base (IBM Power PC)
- CISC
  - Anything goes...

The Ultimate CISC - VAX-11

- ISA defined late 70's. Last product mid 80's
- Over 200 instructions
  - Some very powerful: "polynomial evaluation", procedure calls with register saving and frame set-up etc
- Complex addressing modes

A sample of VAX addressing modes

- Immediate (with even some small f-p constants)
- Direct (register) One instruction for each I-unit type
- Indirect (deferred)
- Autodecrement (and autoincrement) . The register is incremented by the I-unit type before (after) the operand is accessed
- Displacement (like MIPS indexed)
- Index like displacement but offset depends on the I-unit
- Combination of the above and more

Examples

- CLRL register (clears a whole register)
- CLRB register (clears the low byte of the register)
- CLRL (register) clears memory word whose add. is in reg.
- CLRL (register)+ as above but then register is incr. by 4
- CLRL @(register)+ as above with 1 more level of indirection (register points to address of address)
- CLRL offset(register) offset mult.by 4 for L, by 1 for B etc
- CLRL offset[register] similar but use offset + 4 * register
- CLRL 12(R4)+[R1] clear word at add. R4 + 12*4 + R1*4 and add 4 to R4

Intel x86: the largest number of CPU's in the world

- ISA defined early 80's
- Compatibility hurts:
  - 16 to 32-bit architecture to 64-bit as of 2004
  - Paucity of general-purpose registers -- only 8
- Addressing relies on segments (code, data, stack)
- Lots of different instruction formats
- Lots of addressing modes (less than the VAX though!)
- But … over 400(?) millions CPU's in the world and growing
  - 90% (?)of the market if you don't count embedded processors

X86 instruction encoding

- Opcode 1 or 2 bytes (defined by one bit in first byte)
- First byte following opcode is operand specifier
  - e.g., 2 registers
  - 1 register and the next byte specifies base and index register for a memory address for second operand and next byte specifies a displacement etc
  - etc.
- No regularity in instruction set

MIPS is not the only RISC

- MIPS family outgrowth of research at Stanford (Hennessy)
- DEC (Compaq,HP) Alpha had its roots in MIPS
  - Alas, discontinued
- Sun family outgrowth of research at Berkeley (Patterson)
- IBM Power/PC family outgrowth of research at IBM Watson (Cocke)
- HP Precision architecture
- more ...

Recent trends for high-end servers

- 32-bit architectures are becoming 64-bit architectures
  - already in Dec Alpha, some HP-PA, in AMD and most recent Intel
  - CISC ISA but RISC-like implementation
- A "new" type of instruction format
  - VLIW (Very Long Instruction Word) or EPIC (Explicitly Parallel Instruction Computing)
    - Intel-HP Itanium(IA-64); much relaince on compilers
- Multithreaded architectures (Tera; SMT is a UW invention; Hyperthreading in some recent Intel processors)
- More than one processor on a chip – CMP (IBM Power 4)
- Embedded systems become "systems on a chip"

Current trends in RISC

- Not that "restricted"
  - instructions for MMX (multimedia) and graphics
  - instructions for multiprocessing (synchronization, memory hierarchy)
- Design is becoming more complex
- Execute several instructions at once (multiple ALU's)
  - Speculative execution (e.g., guess branch outcomes)
  - Execute instructions out-of-order (but "commit" them in program order)
- Ultimate goal is (was?) speed
  - Reliability, power-awareness are becoming important