Some Other Instruction Set Architectures

Overview

- Alpha
- SPARC
- ARM
- i386

Alpha

- Processor architecture designed by Digital Equipment Corporation (DEC)
  - Purchased by Compaq
  - Purchased by HP
- The alpha is a 64-bit RISC processor similar to MIPS
- The alpha is also dead ☹
- The alpha processor handbook is a superb piece of documentation

Alpha

- “Alpha AXP is a 64-bit load/store RISC architecture that is designed with particular emphasis on the three elements that most affect performance: clock speed, multiple instruction issue, and multiple processors.”
- The first implementation issues 2 instructions per cycle.

Alpha

- 32 Integer registers, all 64 bits wide.
- 32 floating point registers, all 64 bits wide
- 2 Lock registers
- Processor cycle counter register

Alpha

- 32 bit instruction words

31 26	25 21	20 16	15 0
opcode	ra	rb	Memory_disp

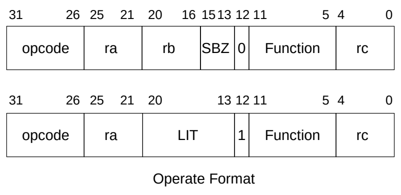
Memory Format

31 26	25 21	20 0
opcode	ra	Branch_disp

Branch Format

Alpha

- 32 bit instruction words



Alpha

- Alpha assembly convention has destination register last
  - add \$t0, 4, \$t1 # \$t1 = \$t0 + 4

Alpha

- Jump instruction uses a memory format encoding
  - Destination specified in Rb
  - Displacement field used to hint where the jump encoded in Rb will go, allowing early I-cache fill

Alpha

- s4add,s8add, s4sub, s8sub
  - Scaled addition/subtraction by 4 and 8 respectively
  - How would you do this in MIPS?

Alpha

- Conditional move integer (CMOVxx)
  - cmoveq \$t0, 4, \$t1 # \$t1 = 4 if \$t0 eq 0
  - cmovge \$t0, \$t1, \$t2 # \$t2 = \$t1 if \$t0 ge 0
  - How would you do this in MIPS?
- \$t1 = MAX(\$t1, \$t2)
  - cmplt \$t1, \$t2, \$t3
  - cmovne \$t3, \$t2, \$t1
- What is nice from an architecture point of view about the cmov instructions?

Alpha

- No divide instruction
  - Compiler must provide divide routines
- All memory accesses must be on a 64-bit word aligned boundary
  - Tedious when all you want is a single byte.
  - BWX extensions added in a later revision

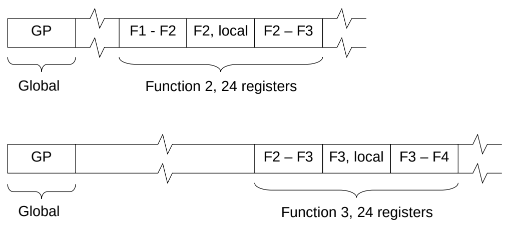
SPARC

- Designed at the same time as the MIPS
  - MIPS was a Stanford project
- SPARC descended from the RISC project at Berkeley
  - Scalable Processor ARChitecture
  - was more successful than the MIPS project.
  - MIPS (and other processors like it) are known as RISC processors.

SPARC

- Concept of register windows
  - processor has up to 128 registers
    - 32 are visible at any one time
    - 8 global
  - 8 local to current procedure
    - Store temporary variables, intermediate working
  - 16 shared with adjacent procedures
    - Used to pass parameters and return values between functions

SPARC

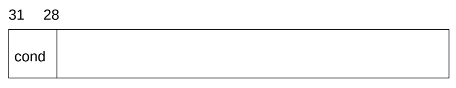


SPARC

- Register Windows
  - alternative to going through tedious procedure entry / exit sequences where registers are saved to a stack
  - MIPS designers leave precise register usage up to compilers

ARM

- Every instruction is executed conditionally
  - COND_AL: always
  - Remainder of conditions are arithmetic in nature



ARM

- Register operands for data instructions can be shifted any of four ways, with shift amount specified in register or immediate
  - Logical shift left
  - Logical shift right
  - Arithmetic shift right
  - Rotate right

ARM

- Thumb
  - Separate processor mode
  - Useful for making compact code
  - Instructions are 16 bits wide, operate on 32-bit values
  - Condition field present in only a few instructions
  - Reduced range for unconditional branches (2KB)
  - Reduced range for conditional branches (256 bytes)
- Jazelle
  - Proprietary mode for processor to accelerate java byte code
  - Useful in cell phones
  - Documentation for Jazelle not publically available

Intel i386

- registers are not general purpose
  - instructions expect their operands in specific registers
- destination can be either a memory location or a register
- complex instruction formats
  - somewhat restrictive too
  - in arithmetic instructions, the destination has to match one of the sources.

Intel i386

- CPU does not require aligned access
  - instruction length varies, 1 – 17 bytes.
- 80386 can access byte, 16-bit, and 32-bit parameters
  - most operations provide two parameter length modes
  - choice between 16bit and 32bit made with bit in code segment register.

Summary

- Choice of instruction set up to designer
  - More than one way to design RISC-like processor
- Has tradeoffs in terms of
  - Simplicity
  - Number of instructions required to execute common sequences of code
  - Ease of ISA implementation optimisation