CISC Developments

Over Twenty Years

RJS 2/3/97

Overview

- Classic CISC design: Digital VAX
- VAX's RISC successor: PRISM/Alpha
- Intel's ubiquitous 80x86 architecture
  - 8086 through the Pentium Pro (P6)

CISC Designs

- Philosophy
  - Reduce code size
  - Machine language should match the semantics of high level language constructs
- Results
  - Complicated instructions which could take many cycles to execute

RISC Designs

- Philosophy
  - Simple, quick instructions.
- Results
  - Instruction dependencies can be easily determined.
  - Instructions can be sub-divided and pipelined.
  - High instruction throughput.

Digital VAX

- Mid-70's design
  - 32-bit architecture
  - Basically a general-purpose register machine
- CISC philosophy
  - Numerous addressing modes
  - Rich instruction suite

VAX: Addressing Modes

Register	r4
Base (displacement)	[r4 + offset]
Immediate	0xFFF0101
PC-relative	[PC] + offset
Deferred (indirect)	[[r3 + offset]]
Index (scaled)	[r3 + r4 * 8]

VAX: Addt'l Addressing Modes

- Byte, word, double word displacement
- Auto-increment/auto-decrement
  - Acesses memory and then increments/ decrements address

VAX: Instruction Encoding

- Operations: 1 byte
- Each operand must be encoded to specify the addressing mode
- Example:
  - Integer add: 3-19 bytes

VAX: Instructions

- Push
  - Push an item onto a stack
- insque
  - Insert an item onto a queue
- aobleq op1 op2
  - Add one to op1 and branch if equal to op2
- Special call/ret
  - Handles argument, stack setup automatically

CISC: Side effects

- Instruction basically has multiple results
  - Auto-increment
    - combines a load/store with an addition
  - Condition codes
    - Negative, Zero, oVerflow, Carry
    - Arithmetic instructions set these codes
    - Codes are used for conditional branches

Digital: Beyond VAX

- PRISM/Alpha
  - RISC processor
    - fixed size instructions
    - load/store architecture
    - out-of-order execution
    - speculation
- Backward Compatibility?
  - Recompilation: VAX --> Alphas

INTEL

- 1978: 8086
  - 16-bit, extended accumulator machine
- 1982: 80286 *backward compatible*
  - 24-bit address space
- 1985: 80386 *backward compatible*
  - 32-bit address space
- 1992: Pentium *backward compatible*
- 1996: P6 *backward compatible*

80x86: Overview

- Has moved closer to a general purpose register machine
- Segmented address space
- Instructions work on bytes, words, double words.
- Only four instructions added since 1989.
  - Three multiprocessing instructions
  - One conditional move

80x86: Segmented Addr Space

- Real Mode (8086)
  - Segment register is shift to the left 4 bits and the offet is added.
- Protected Mode (80286)
  - Segment register selects an index into the segment address table. (24 bits) Offset is added.
- Protected Mode (80386, 80486, Pentium)
  - Segment descriptor is 32 bits.

80x86: Addressing Modes

Absolute	[0xa0000000]
Register	[r4]
Based displacement	[r4 + offset]
Indexed	[r3 + r4]
Based indexed	[r3 + r4 + offset]
Base plus scaled index	[r3 + r4 * scale]
Base scaled displacement	[r3 + r4 * scale + offset]

80x86: Instruction Complications

- Instruction prefixes
  - Override default data size, segment registers
  - Lock the bus (i.e synchronization)
  - Repeat instruction until register CX counts to zero.
- Multiple segments complicate control-flow statements

80x86: Instruction Usage

- Instructions: 1 to 17 bytes
  - Integer programs: av=2.8
  - Floating point programs: av=4.1
- Most frequent addressing modes: Based displacement and Based scaled indexing.

Intel: Improving CISC

- Pentium: Leveraged RISC technology
  - 5-stage pipeline
  - Dual-issue
  - No branch prediction
  - No out-of-order execution
- Software basically needs recompilation

Pentium Pro (P6)

- A RISC processor running a CISC instruction set.
  - Three way issue
  - Speculative execution
  - out-of-order execution
- Superscalar pipeline
  - Allows higher clock rate while handling CISC

P6: Pipeline Stages 1 -4

- Stage 1
  - Determines next PC
- Stage 2-4
  - Fetch and mark instruction

P6: Pipeline Stages 5-6

- Instruction is decoded into a series of micro-ops (uops)
  - Three decoders
    - Two decoders handle simple instructions
    - Third decoder handles more complex cases
    - Falls through decoder to a special microcode area
  - Majority translate to < 4 uops.
  - Worst case: 204 uops

P6: Pipeline Stages 7-8

- Assign logical registers to physical registers
  - 80x86 machine instructions are limited to the basic x86 registers.
  - x86 architecture has 16 physical registers
  - P6 has 40 physical registers
- Prepared uops are passed to the reservation station and reorder buffer

P6: Pipeline Stages 9-10

- Reservation station dispatches uops to one of five parallel execution units.
  - Two integer, one load, one store, and one FPU
- Reorder buffer holds the uops
  - Status flags signal dependencies

P6: Pipeline Stage 11

- Execution
  - May overlap into stage 12

P6: Pipeline Stages 12-14

- Instructions are retired
  - Must wait until all uops that comprise the instruction are complete
  - Must handle precise exceptions

P6: Pipeline Performance

- 80x86 references memory more often than a typical RISC instruction set.
  - Floating point programs: 2-4x higher
  - Integer programs: 1.25 higher
- Superpipelining and branch prediction problems.
  - P6 uses 4-bit branch history

P6: Completely RISC?

- Can a compiler directly generate uops?
  - Bypass decoding phase
  - Leverage static scheduling techniques
  - Force competing chip makers to use same basic design.
    - Could reduce differentiating points between competitors.

References

- Hennesy and Patterson, “Computer Organization and Design: The Hardware/Software Interface”
- Hennesy and Patterson, “Computer Architecture: A Quantitative Approach”
- Bhandarkar, “Alpha Implementations and Architecture”
- Byte, April 1995, “Intel’s P6”
- http://www.intel/procs/ppro/info/isscc/index.htm