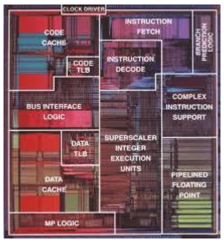
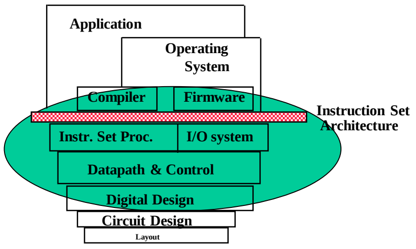
UTD

CS/SE 3340
Computer Architecture



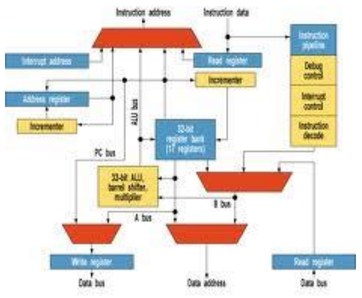


Comparing ISAs: MIPS v.s. ARM and x86

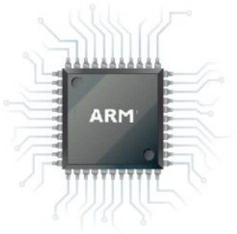
Adapted from slides by Profs. D. Patterson and J. Hennessey

Questions

- What are similarities between MIPS and ARM ISAs ?
- How instructions are encoded in ARM ISA?
- How has the x86 ISA been evolved?
- What/Why/How of microArchitecture in x86 ISA?
- What is the ARM story?



The Cortex M3's Thumbnail architecture looks like a conventional Arm processor. The differences are found in the Harvard architecture and the instruction decode that handles only Thumb and Thumb 2 instructions.



MIPS TECHNOLOGIES

intel CORE i7 inside

3

ARM & MIPS Similarities

- ARM: the most popular embedded core
- Similar basic set of instructions to MIPS

	ARM	MIPS
Date announced	1985	1985
Instruction size	32 bits	32 bits
Address space	32-bit flat	32-bit flat
Data alignment	Aligned	Aligned
Data addressing modes	9	3
Registers	15 × 32-bit	31 × 32-bit
Input/output	Memory mapped	Memory mapped

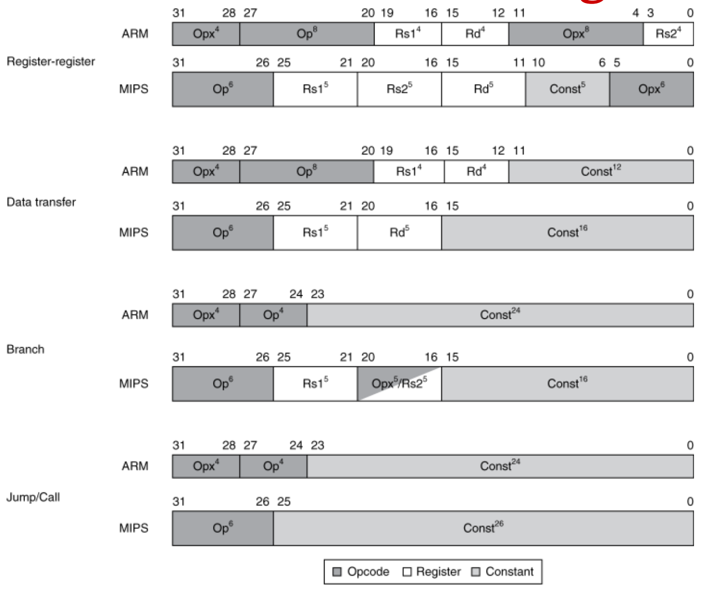
4

Compare and Branch in ARM

- Uses condition codes for result of an arithmetic/logical instruction
  - Negative, zero, carry, overflow
  - Compare instructions to set condition codes without keeping the result
- Each instruction can be conditional
  - Top 4 bits of instruction word: condition value
  - Can avoid branches over single instructions

5

Instruction Encoding



6

The Intel x86 ISA

- Evolution with backward compatibility
  - 8080 (1974): 8-bit microprocessor
    - Accumulator, plus 3 index-register pairs
  - 8086 (1978): 16-bit extension to 8080
    - Complex instruction set (CISC)
  - 8087 (1980): floating-point coprocessor
    - Adds FP instructions and register stack
  - 80286 (1982): 24-bit addresses, MMU
    - Segmented memory mapping and protection
  - 80386 (1985): 32-bit extension (now IA-32)
    - Additional addressing modes and operations
    - Paged memory mapping as well as segments

7

The Intel x86 ISA – cont'd

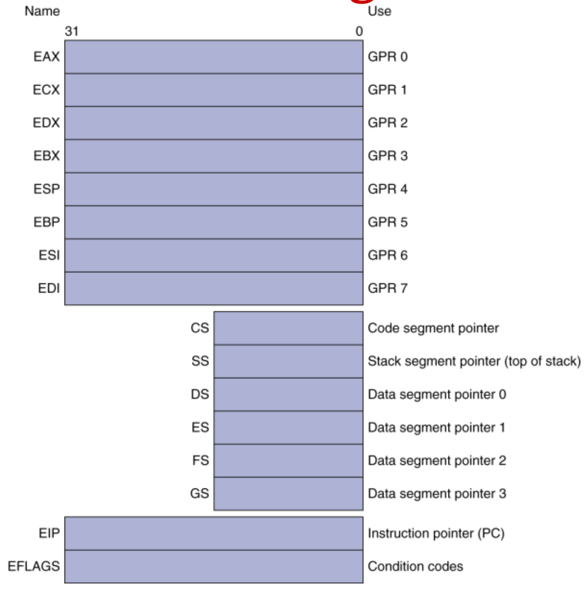
- Further evolution…
  - i486 (1989): pipelined, on-chip caches and FPU
    - Compatible competitors: AMD, Cyrix, …
  - Pentium (1993): superscalar, 64-bit datapath
    - Later versions added MMX (Multi-Media eXtension) instructions
    - The infamous FDIV bug
  - Pentium Pro (1995), Pentium II (1997)
    - New **microarchitecture** (see Colwell, *The Pentium Chronicles*)
  - Pentium III (1999)
    - Added SSE (Streaming SIMD Extensions) and associated registers
  - Pentium 4 (2001)
    - New **microarchitecture**
    - Added SSE2 instructions

8

The Intel x86 ISA – cont'd

- And further…
  - AMD64 (2003): extended architecture to 64 bits
  - EM64T – Extended Memory 64 Technology (2004)
    - AMD64 adopted by Intel (with refinements)
    - Added SSE3 instructions
  - Intel Core (2006)
    - Added SSE4 instructions, virtual machine support
  - AMD64 (announced 2007): SSE5 instructions
  - Intel declined to follow, instead…
  - Advanced Vector Extension (announced 2008)
    - Longer SSE registers, more instructions
- If Intel didn't extend with compatibility, its competitors would!
  - *Technical elegance ≠ market success*

Basic x86 Registers



Basic x86 Addressing Modes

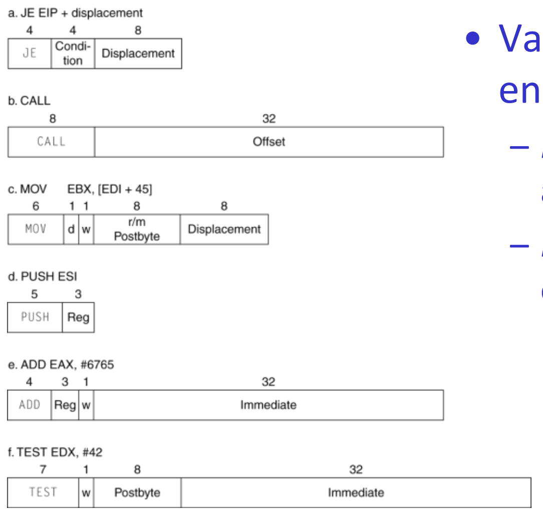
- Two operands per instruction

Source/dest operand	Second source operand
Register	Register
Register	Immediate
Register	Memory
Memory	Register
Memory	Immediate

- Memory addressing modes
  - Address in register
  - Address = R_{base} + displacement
  - Address = $R_{base} + 2^{scale} \times R_{index}$ (scale = 0, 1, 2, or 3)
  - Address = $R_{base} + 2^{scale} \times R_{index}$ + displacement

11

x86 Instruction Encoding



- Variable length encoding
  - *Postfix* bytes specify addressing mode
  - *Prefix* bytes modify operation
    - Operand length, repetition, locking, …

12

Implementing IA-32

- Complex instruction set makes implementation difficult
  - Hardware translates instructions to simpler micro-operations
    - Simple instructions: 1–1
    - Complex instructions: 1–many
  - *Micro-engine* similar to RISC
- Comparable performance to RISC
  - Compilers avoid complex instructions

13

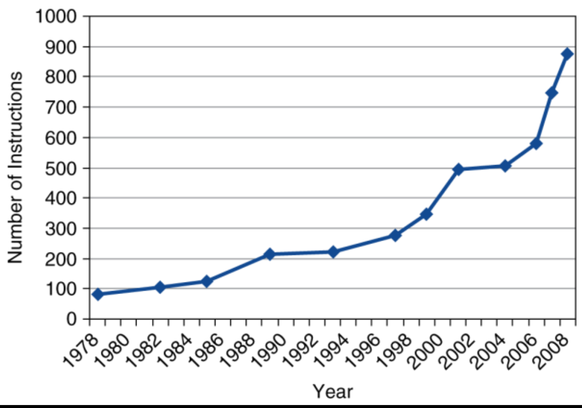
Fallacies

- Powerful instruction $\Rightarrow$ higher performance
  - Fewer instructions required
  - But complex instructions are hard to implement
    - May slow down all instructions, including simple ones
  - Compilers are good at making fast code from simple instructions
- Use assembly code for high performance
  - Used to be true, but modern compilers are getting better at dealing with modern processors
  - More lines of code $\Rightarrow$ more errors and less productivity

14

Fallacies

- Backward compatibility ⇒ instruction set doesn't change
  - But they do accrete more instructions



x86 instruction set

15

Pitfalls

- Sequential **words** are not at sequential addresses
  - *Increment by 4, not by 1 (MIPS)!*
- Keeping a *pointer* to an automatic variable after procedure returns
  - e.g., passing pointer back via an argument
  - Pointer becomes invalid when stack popped

16

Concluding Remarks

- Design principles
  1. *Simplicity favors regularity*
  2. *Smaller is faster*
  3. *Make the common case fast*
  4. *Good design demands good compromises*
- Layers of software/hardware
  - Compiler, assembler, hardware
- MIPS: typical of RISC ISAs
  - c.f. x86

17

Concluding Remarks

- Measure MIPS instruction executions in benchmark programs
  - Consider making the common case fast
  - Consider compromises

Instruction class	MIPS examples	SPEC2006 Int	SPEC2006 FP
Arithmetic	`add, sub, addi`	16%	48%
Data transfer	`lw, sw, lb, lbu, lh, lhu, sb, lui`	35%	36%
Logical	`and, or, nor, andi, ori, sll, srl`	12%	4%
Cond. Branch	`beq, bne, slt, slti, sltiu`	34%	8%
Jump	`j, jr, jal`	2%	0%

18

The Creators of ARM

Sophie & Steve

19