# *Computer Architecture: Fundamentals, Tradeoffs, Challenges*

# *Chapter 2: The ISA*

*Yale Patt*

*The University of Texas at Austin*

*Austin, Texas*

*Spring, 2025*

# *Outline*

- ***The ISA -- What is it?***
  - ***The interface between hardware and software***
  - ***A specification***
  - ***NOT microarchitecture***
  - ***NOT just the instruction set***
- ***The Instruction***
  - ***The atomic unit of processing***
  - ***Changes the state of the machine***
- ***Characteristics of an ISA (using LC-3b as an example)***
- ***The LC-3b instruction set***
- ***Other ISAs***
  - ***X86***
  - ***RISCV***
- ***ISA Tradeoffs (with examples)***

# *What is the ISA?*

- ***A specification***
  - ***The interface between hardware and software***

- ***A contract***
  - ***What the software demands***
  - ***What the hardware agrees to deliver***

# *NOT Microarchitecture*

- ***Architecture***
  - ***Software Visible***
  - ***Address Space, Addressability***
  - ***Opcodes, Data Types, Addressing Modes***
  - ***Privilege, Priority***
  - ***Support for Multiprocessors (e.g., TSET)***
  - ***Support for Multiprogramming (e.g., LDCTX)***
- ***Microarchitecture***
  - ***Not Software Visible***
  - ***Caches (although this has changed, …sort of)***
  - ***Branch Prediction***
  - ***The instruction cycle***
  - ***Pipelining***

***DIGRESSION (nugget): You have a brilliant idea, and It requires a change to the ISA or to the uarchitecture.***

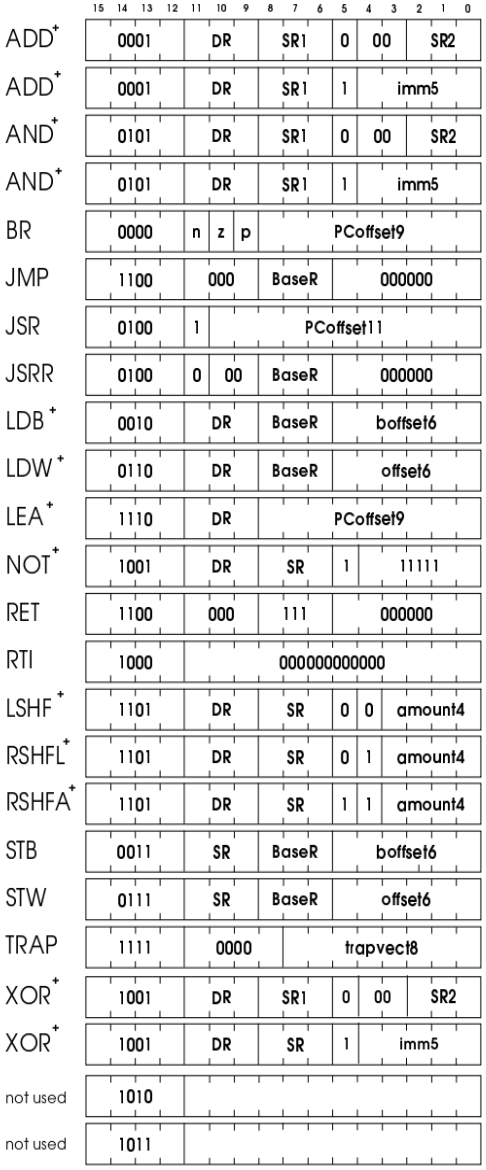
# *Another DIGRESSION (nugget)*

- ***The pure distinction between ISA vs uarchitecture***
  - ***ISA is visible to the software***
  - ***Microarchitecture is "underneath the hood"***
- ***BUT some have noticed that…***
  - ***If you let the compiler know how the ISA is implemented,***
  - ***i.e., if you break the walls between the transformation levels,***
  - ***You can produce better code for that implementation***
  - ***…at the expense of compatibility***
- ***Today, with the impending demise of Moore's Law,***
  - ***Computer Architecture is looking for ways to still be relevant***
  - ***I have been preaching: Break the layers!***
  - ***MIT recent white paper: "There is plenty of room at the top!"***

# Characteristics of an ISA (e.g., the LC-3b)

- Processor State (memory, registers)
  - Memory addressability: byte
  - Memory address space: 2^16
  - Registers: 8 GPR, Condition Codes N, Z, P
  - Word length: 16 bits
  - Program Counter (Actually, Instruction pointer)
  - Process Status Register (contains Privilege, Priority, CC)
- Privilege: 2 levels, supervisor, user
- Priority: 8 levels
- Instruction format: fixed length, uniform decode. 16 bits
  - Flynn's observation: Two or three lengths better than all the same
- Endian-ness: little endian
- Instructions (opcode, addressing mode, data type)
  - Opcode (14 opcodes, including XOR, SHF, LDB)
  - Addressing modes (PC-relative, Register + offset)
  - Data types (2's complement 16 bit integers, bit vector)
  - Three-address machine (Load/Store ISA)

# *The LC-3b Instruction Set*



# *Pragmas (aka Pseudo-ops)*

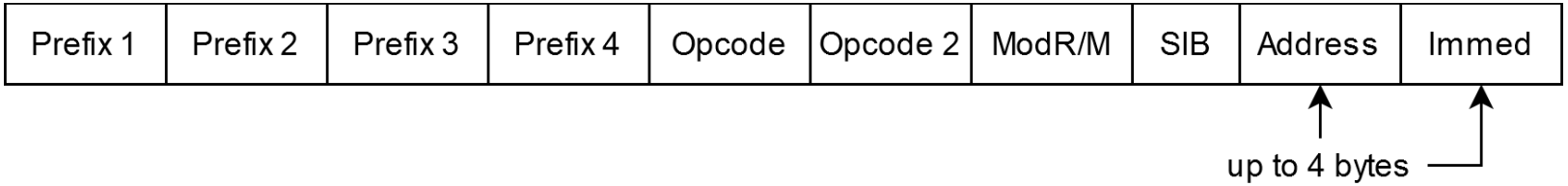
- ***Not part of the instruction set***
  - ***Not executed by the computer***
- ***Messages from the programmer to the translater***
  - ***Necessary for the translater to produce object code***
  - ***Opportunity for programmer to affect performance (e.g., hints)***

# *Characteristics of an ISA (continued)*

- *Vector architecture (instructions, operands)*
  - *Not part of the LC-3b*
- *Virtual memory specification: not yet part of LC-3b!*
  - *Address space*
  - *Translation mechanism*
  - *Protection*
  - *Page size*
- *System architecture*
  - *State to deal with: trap vector table, interrupt vector table*
  - *Interrupt, exception handling*
  - *Instructions for the O/S to use (RTI)*
- *NOT the instruction cycle (that is part of the uarch)*

# *x86*

- ***Variable length instruction (one byte to 16 bytes)***




- ***Characteristics***
  - ***Rich set of addressing modes***
  - ***Two-address machine***
  - ***SSE extension (originally, MMX)***
  - ***Not load/store***
  - ***Three page sizes (4KB, 2MB, 1GB)***
  - ***Register sizes: 8b, 16b, 32b, 64b, 128b, …***
  - ***Example: AH, Ax, EAX, …***
  - ***Memory: Byte addressable, 64 bit address space***

# *RISCV*

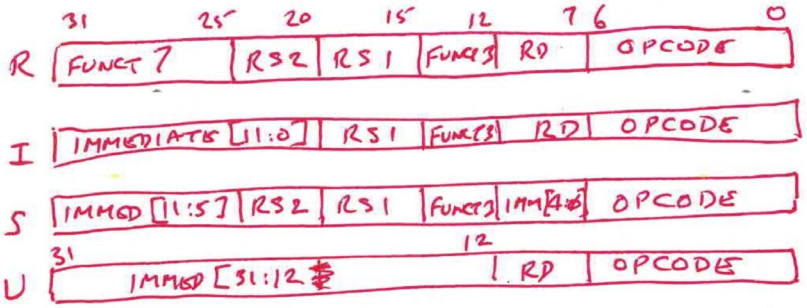
- ***The 5th chip from Professor David Patterson's group***
  - ***UC Berkeley***
  - ***Nothing (really) in common with their other four risc chips***
- ***Mostly handled by Professor Krste Asonovic***
- ***Major selling point: Open Source***
- ***Overall structure***
  - ***Multiple subset ISAs (Integer, Float, MUL/DIV, Atomic, etc.)***
  - ***Designers build their own system, picking and choosing***
  - ***MUST contain one of the Integer subsets (32-bit or 64-bit)***
  - ***The rest (extensions) are up to the designer***

# *RISCV (characteristics)*

- ***The subsets***
  - ***Integer: 32-bit (RV32I), 64-bit (RV64I), 128-bit (RV128I)***
  - ***Float extension: 32-bit (RV32F), 64-bit (RV64D), 128-bit (RV128Q)***
  - ***M extension: Integer MUL/DIV***
  - ***A extension: Atomic instructions***
  - ***L extension: Decimal float***
  - ***C extension: Compressed***
  - ***B extension: Bit manipulation***
  - ***J extension: Dynamically translated***
  - ***T extension: Transactional memory***
  - ***P extension: Packed SIMD***
  - ***V extension: Vector operations***
  - ***E extension: Embedded Controller (RV32E)***
  - ***G extension: A system, really (IMAFD)***

# RISCV Characteristics (continued)

- *RV32I (32 bit Integer data type)*
  - *47 distinct opcodes (*
    - *loads, stores, shifts, arith, logic, compare, branch, jump, synch, count*
  - *32 GPRs (x0 to x31, x0=0, x1 used for call return linkage)*
  - *Also contains a PC*
  - *32 bit instructions*
    - *Can be extended by a multiple of n bits*
    - *Mixture allows for unaligned access*
    - *Also allows for 16 bit instructions, but then restricted to 8 registers*
  - *4 basic instruction formats*




  - *Other*
    - *Little endian*
    - *Load/Store*
    - *No predication*
    - *Conditional branches use GPRs, not condition codes*

# *ISA Tradeoffs (examples)*

- ***Dynamic-Static Interface (The Semantic Gap)***
  - ***ISA closer to the HLL***
  - ***ISA closer to the control signals***
- ***ISA-level CAPABILITY (aka "Security")***
  - **instructions included privilege needed**
  - **Intel 432, IBM System 38, Data General Fountainhead**
  - **All failed! Why? Performance was very slow**
- ***PREDICATION***
  - ***x86 CMOV, ARM inst[31:28], THUMB IT block***
- ***Support for multiple ISAs* on the same chip**
  - ***ARM T bit, VAX Compatibility mode bit***
  - ***Both in the PSR, making it part of the ISA.***

# *ISA Tradeoffs (continued)*

- ***REGISTER SET -- how many reg, how many bits each***
  - ***Many machines: 32 32-bit registers***
  - ***More registers save spills to memory, but longer context switch***
  - ***x86 now has 512-bit registers***
  - ***Itanium has 1-bit predicate registers***
  - ***Register window (SPARC ISA) vs set of gprs***
- ***Condition codes vs gprs. (MIPS, CDC6600 used gprs)***
  - ***Serialization of code vs getting something done for nothing***
  - ***John Cocke RS6000 -- 8 sets of 4 condition codes each***
- ***RICH INSTRUCTION SET vs. LEAN INSTRUCTION SET***
  - ***Hewlett-Packard: HPPA RISC had 140 opcodes***
  - ***So much for RISC being a small set of instructions!***

# *ISA Tradeoffs (continued)*

- ***Do we include complex instructions (faster, but more gates)***
  - ***EDITPC -- for COBOL***
  - ***INDEX -- for Fortran***
  - ***AOBLEQ -- For loop***
  - ***LDCTX/SAVCTX -- to swap in/out context needed for next quantum***
  - ***Test and Set (an Atomic operation) – for multiprocessing***
  - ***CALL -- formal procedure call***
    - ***more work, but cleaner interface to libraries***
    - ***In addition to JSR (quick and simple)***
  - ***FF -- find first***
  - ***INSQUE/REMQUE -- doubly linked list data type***
  - ***TRIADS – example: MAC (multiply accumulate)***
  - ***CHMD -- to change privilege mode***

# *ISA Tradeoffs (continued)*

- ***INST SET Orthogonal vs Tailoring to each opcode***
- ***LOAD/STORE vs NOT Load/Store***

  ***Load/Store: Can't access storage and operate in same instruction***

  - ***LC-3b is load/store, x86 is not***
  - ***Load/Store had more importance before o-o-o processing***
- ***MEMORY ADDRESS SPACE -- keeps getting larger***
- ***MEMORY ADDRESSABILITY -- depends on needs***
  - ***Most memories: byte addressable (for data processing)***
  - ***Scientific machines -- 64 bits***
  - ***Burroughs 1700 -- one bit (needed for virtual machines)***

# *ISA Tradeoffs (continued)*

- ***VLIW (compile time) vs. Superscalar (do it at runtime)***
  - ***Compile time in ISA, runtime is uarchitecture (Superscalar)***
  - ***VLIW: Hard to find enough instructions to fill the long word,***
  - ***VLIW is fast because there are no dependencies at runtime.***
  - ***Examples Multiflow, cydra five***
  - ***Itanium 3 in one VLIW, but allows for dependencies***

- ***0,1,2,3 address machines. 3 addresses are needed.***

  ***How many have to be explicit?***
  - ***0. stack machine: pop, pop, compute, push***
  - ***1. one accumulator when reg were expensive AC ← AC + Memory***
  - ***2. two. X86 (Rd ← Rd op Rx)***
  - ***3. LD/ST all three are explicit Rd ← Rs1 op Rs2 (LC-3b)***
  - ***VAX ISA had both 2-address and 3-address***

# *ISA Tradeoffs (continued)*

- ***Word length (max size processed as a unit)***
  - ***VAX 32 bits, x86 initially 16 bits, then 32, today 64;***
  - ***CRAY-1 64bits***
  - ***DEC System 20, an AI Machine. LISP's car, cdr needed 36 bits***
- ***Help for the programmer or help for the microarchitect***
  - ***Aligned vs. unaligned accesses easier – who gets the easy job?***
  - ***PDP11 NO, VAX YES, ALPHA NO***
  - ***Data types; int, float, bitvector, char string, doubly linked list***
  - ***Addressing modes; indirect, index, autoinc/autodec/ SIB byte***

# *ISA Tradeoffs (continued)*

- *PAGE SIZE*
  - *(4KB vs more than one size*
  - *Intel has 3three: 4KBm 2MB, 1GB)*
  - *Faruk has all sizes  2^n for n greater or equal to 12*
    - *Efficient use of TLB, vs. Wasted space*
    - *Better use of TLB (fewer PTEs) vs. wasted space*
    - *Too many PTEs vs longer access time*
- *I/O architecture*
  - *Today most use memory mapped I/O with LD,ST*
  - *OLD days, special I/O instructions*
  - *x86 still has both*
- *Compile time vs run time*
  - *MIPS use of NOPS rather than hardware interlocks*

# *ISA Tradeoffs (continued)*

- ***Synch vs Asynch***
- ***Instruction format:***
  - ***Most: fixed length, uniform decode (easier multiple decode)***
  - ***X86: variable length more work done per instruction)***
  - ***Intel 432: different sizes for different opcodes (Nightmare!)***
- ***SPEED DEMONS vs. BRAINIACS***
  - ***Alpha: Speed demon, 200 MHz;***
  - ***Pentium: Brainiac, 66MHz***

***Obrigado!!***