Computer Architecture: Fundamentals, Tradeoffs, Challenges

Chapter 2: The ISA

Yale Patt

The University of Texas at Austin

Austin, Texas

Spring, 2023

Outline

- ***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***
  - ***First, a simple example: The LC-3b***
  - ***The x86, RISCV***
  - ***Vector Architecture***
- ***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.

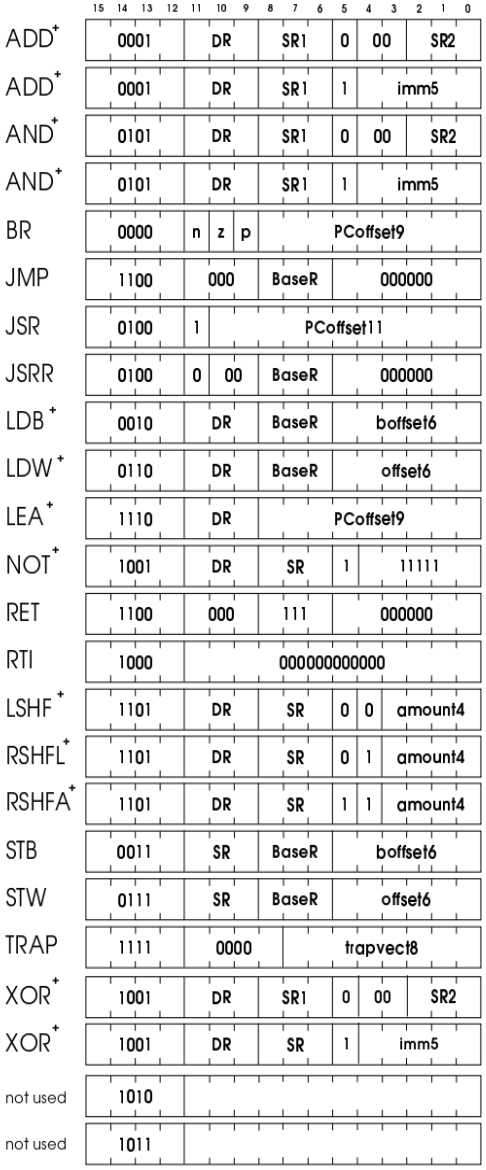
Another DIGRESSION (nugget)

- ***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 (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***
- ***Privilege: 2 levels, supervisor, user***
- ***Priority: 8 levels***
- ***Instruction format: fixed length, 16 bits***
- ***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***

The LC-3b Instruction Set

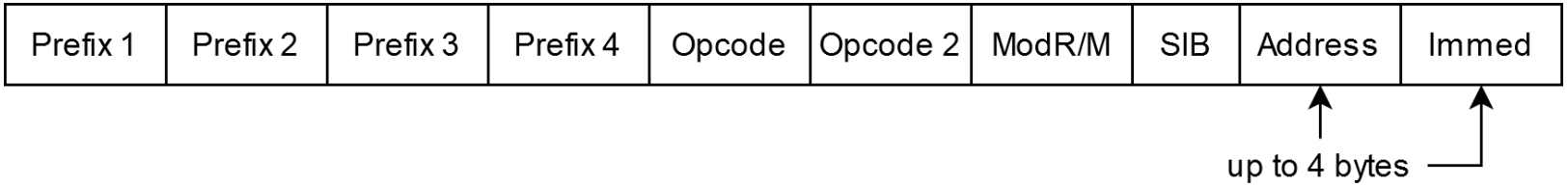


Characteristics (the LC-3b), continued

- ***Vector architecture (instructions, operands): no***
- ***Virtual memory specification: not yet!***
  - ***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, CHMD)***
- ***NOT the instruction cycle (It 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***
  - ***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

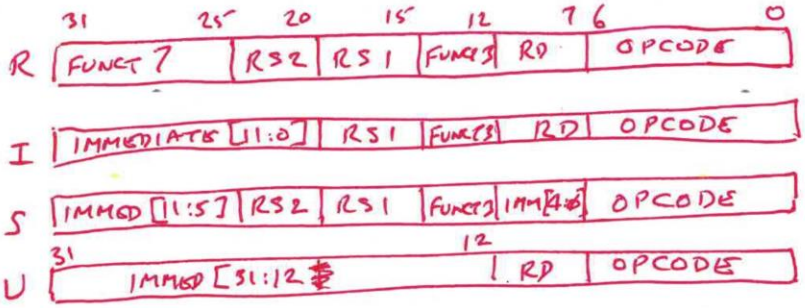
- ***The 5th chip from Professor David Patterson's group***
  - ***UC Berkeley***
  - ***Nothing (really) in common with their other four risc chips***
- ***Mostly handled by Professor Krste Asonovic***
- ***Major selling point: Open Source***
- ***Overall structure***
  - ***Multiple subset ISAs (Integer, 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***
  - ***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***

Vector Architecture

- ***Vector Registers***
  - ***Each register has multiple components***
- ***Vector Instructions***
  - ***Loads/Stores***
    - ***Multiple memory locations in one instruction***
    - ***Length register defines the number of components***
    - ***Stride register defines distance between successive memory locations***
  - ***Operates***
    - ***Operates operate component by component***
    - ***For example, C = A+B means Ci = Ai + Bi for all I***
    - ***Instruction is ADD V3, V1, V2***

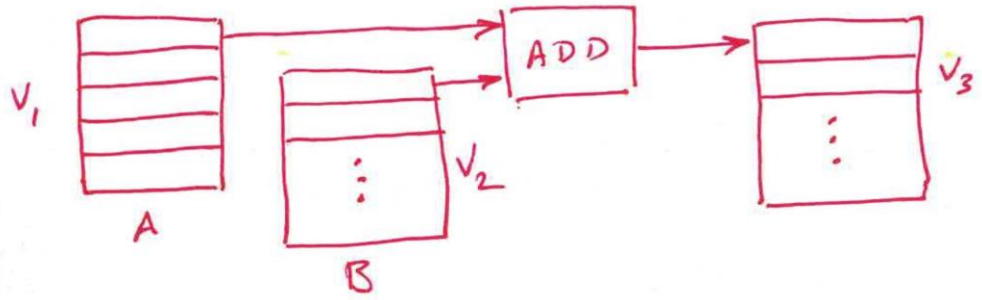
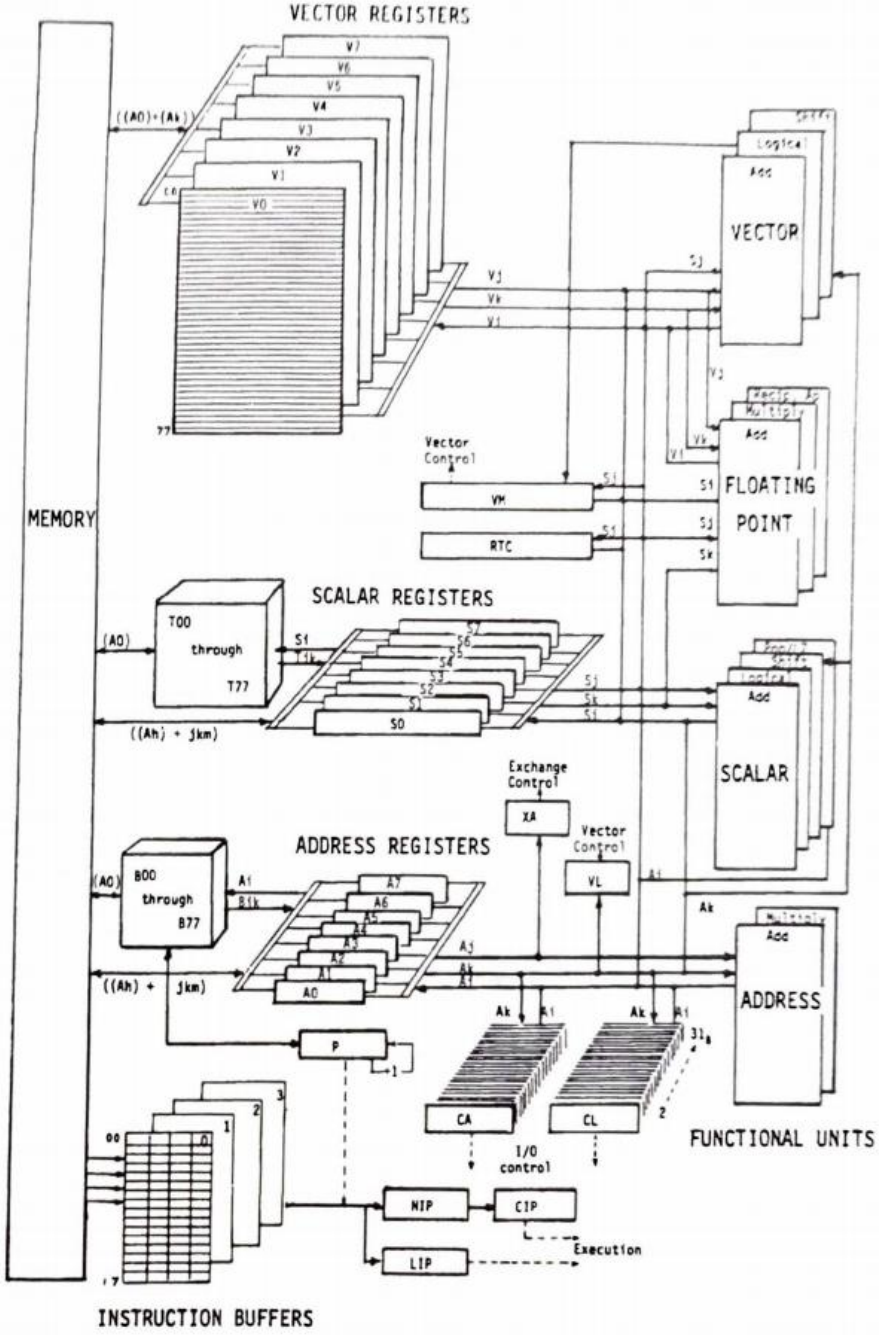


Fig. 5. Block diagram of registers.



Vector processing example

- ***The scalar code:***

  for i=1,50

  A(i) = (B(i)+C(i))/2; Vectorizable!

- ***The vector code:***

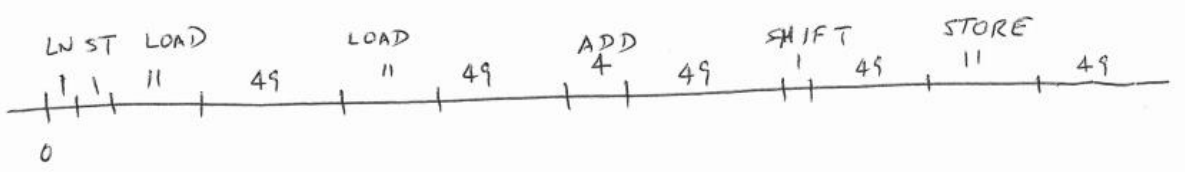
```
lvs 1                ; load vector stride
lvl  50              ; load vector length
vld  V0,B            ; load V0 from memory, starting at address B
vld  V1,C            ; load V1 from memory, starting at address C
vadd V2,V0,V1        ; V2 < --- V0 + V1
vshfr  V3,V2,1       ; V3 < --- V2 divided by 2 (shift right one bit)
vst   V3,A           ; store V3 to memory, starting at address A
```

Vector processing example (continued)

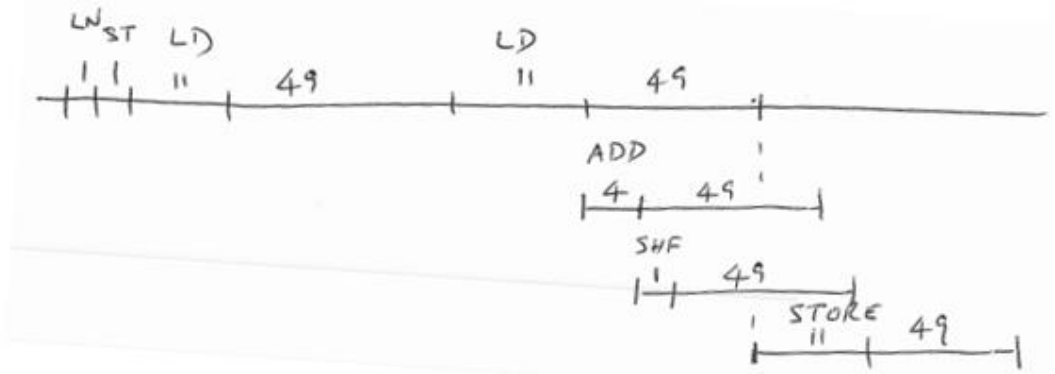
- ***Baseline: with a Scalar Processor:***
  - ***Loads/Stores take 11 cycles***
  - ***Add takes 4 cycles***
  - ***Shift takes 1 cycle***
  - ***Iteration Control takes 2 cycles***
- ***50 iterations of (LD, LD, Add, Shift, Store, Iteration Ctl)***
  - ***50 x (Load, Load, Add, Shift, Store, Iteration_Ctl)***
  - ***50 x (11 + 11 + 4 + 1 + 11 2) = 50 x 40 = 2000 clock cycles***

Vector processing example (continued)
Vector Processor Timing

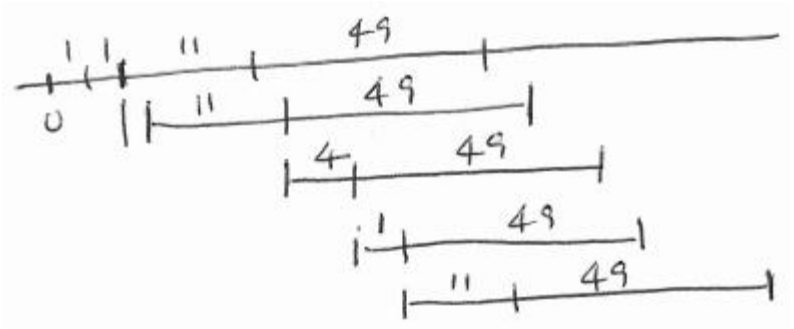
- **Vector code (no vector chaining): 285 clock cycles**



- **Vector code (with chaining): 182 clock cycles**

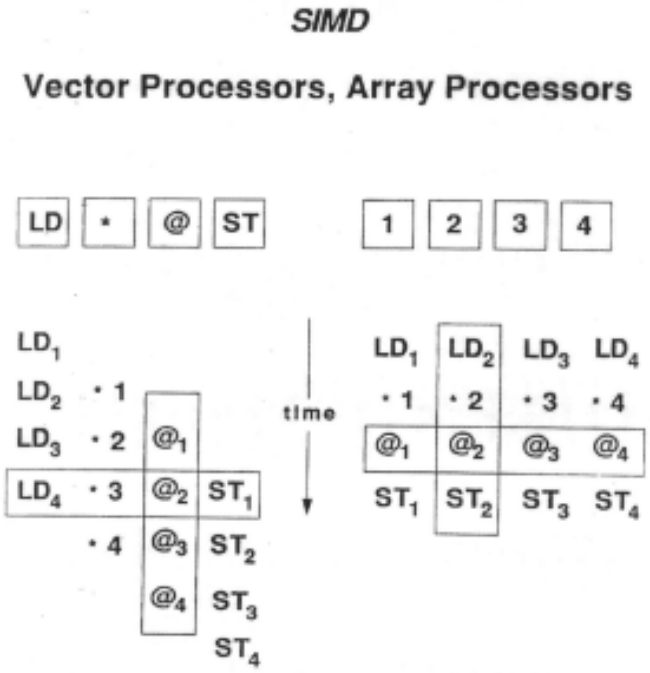


- **Vector code (with 2 load, 1 store port to memory):**



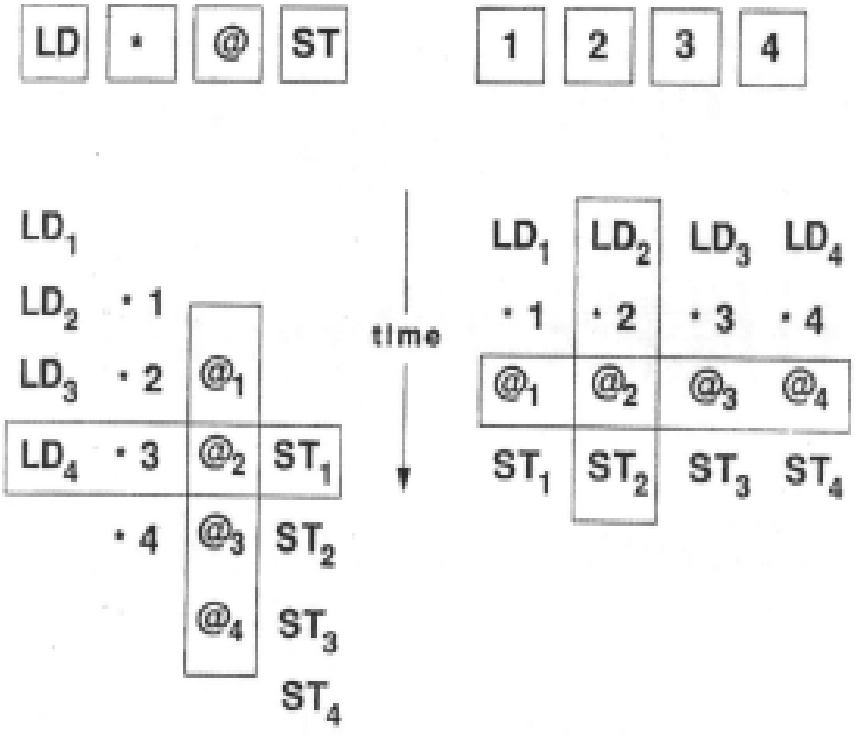
79 clock cycles

Important to note that SIMD can be either Vector Processors or Array Processors



SIMD

Vector Processors, Array Processors



Tradeoffs (with examples)

- ***Dynamic static interface (The semantic gap)***
- ***Some example instructions***
  - ***EDITPC, INDEX, AOBLEQ, LDCTX, CALL, FF,***
  - ***INSQUE/REMQUE, Triads, CHMD, PROBE***
- ***Security (at ISA: capability based ISAs)***
  - ***I432, IBM System 38, Data General Fountainhead***
- ***Predication***
  - ***X86: CMOV***
  - ***ARM: inst[31:28]***
  - ***THUMB: IT block***
- ***Support for multiple ISAs***
  - ***ARM: T bit in the status register***
  - ***VAX: Compatibility mode bit in the PSL***

Tradeoffs (with examples) continued

- ***Register set and size***
  - ***Many machine have 32 32-bit registers***
  - ***x86 now has 512-bit registers***
  - ***Itanium has one-bit predicate registers***
- ***Condition codes vs using a general purpose register***
  - ***MIPS, CDC6600***
- ***Rich instruction set vs Lean instruction set***
  - ***Hewlett Packard's RISC: HPPA has 140 instructions***
  - ***Orthogonal to RISC vs CISC***
- ***Load/Store vs Operate in the same instruction***
  - ***LC-3b, x86 are load/store***
  - ***x86 is not load/store***

Tradeoffs (with examples), continued

- ***Memory address space (keeps growing!)***
- ***Memory addressability***
  - ***Most memories: byte addressable (Data processing)***
  - ***Scientific machines: 64 bits (size of normal fl.pt. operands)***
  - ***Burroughs 1700: one bit (virtual machines)***
- ***Page Size (4KB vs more than one)***
  - ***Wasted space***
  - ***Longer access time***
  - ***Too many PTEs***
- ***I/O architecture***
  - ***Most today use memory mapped I/O***
  - ***Old days, special I/O instructions***
  - ***x86 still has both***

Tradeoffs (with examples), continued

- ***Compile time vs run time***
  - ***MIPS initially had NO hardware interlocks***
- ***Instruction format***
  - ***Most have fixed length, uniform decode***
  - ***x86 has variable length, with prefixes***
  - ***i432 had different bit size opcode***
- ***Word length***
  - ***VAX: 32 bits***
  - ***x86: initially 16 bits, then 32 bits, today 64 bits***
  - ***CRAY 1: 64 bits***
  - ***DEC System 20: 36 bits (LISP car, cdr for AI processing)***

Tradeoffs (with examples), continued

- ***Help for the programmer vs help fo the uarchitect***
  - ***Who gets the cushy job?***
  - ***Unaligned accesses***
  - ***Data Types***
  - ***Addressing modes***
- ***Unaligned access***
  - ***LC-3b does not allow unaligned access***
  - ***DEC: PDP-11 (no), VAX (yes), Alpha (no)***
- ***Data types (rich or lean)***
  - ***Integers, floats of various sizes***
  - ***Doubly-linked list, character string***
- ***Addressing modes (rich or lean)***
  - ***Indirect addressing***
  - ***Autoincrement, postdecrement***
  - ***SIB byte in x86***

Tradeoffs (with examples), continued

- ***VLIW vs …***
  - ***VLIW: compiler does it***
  - ***Superscalar: part of the microarchitecture***
- ***0,1,2,3 address machine (how many EXPLICIT)***
  - ***LC-3b: 3 address***
  - ***x86: 2 address***
  - ***VAX: both 2 and 3***
  - ***Old days: one address (registers were expensive)***
  - ***Stack machine: 0 address***
- ***Precise exceptions vs …***
  - ***Precise exceptions: today, everyone***
  - ***IBM 360/91: NO***
- ***Privilege modes***
  - ***Most ISA have two – supervisor and user***
  - ***VAX had four***

Obrigado!!