Obrigado!!

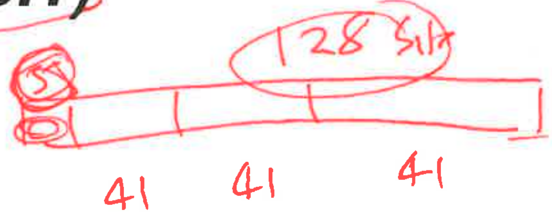
Tradeoffs (with examples), continued

- VLIW vs …
  - VLIW: compiler does it
  - Superscalar: part of the microarchitecture

VLIW VS Superscalar

What is the same?
What is different?

- 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

Almost everyone

← Special permission from IBM Hq.

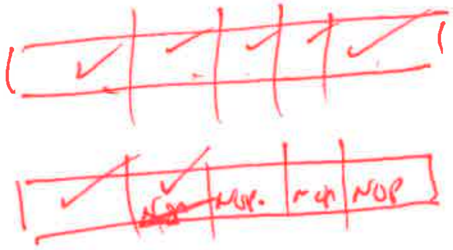
Cydrome
Cydra 5

Multiflow

- Privilege modes
  - Most ISA have two – supervisor and user
  - VAX had four

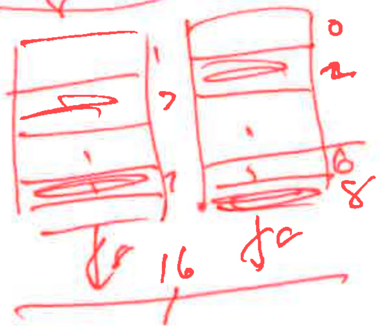
DIV X, Y, Z

ADD Y, –, –



Tradeoffs (with examples), continued

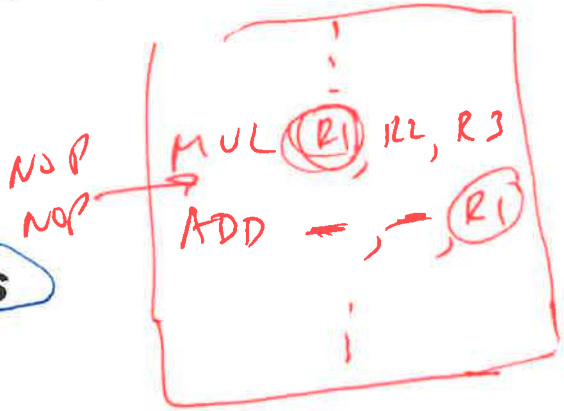
- *Help for the programmer vs help fo the uarchitect* 8
  - *Who gets the cushy job?*
  - *Unaligned accesses*
  - *Data Types*
  - *Addressing modes*



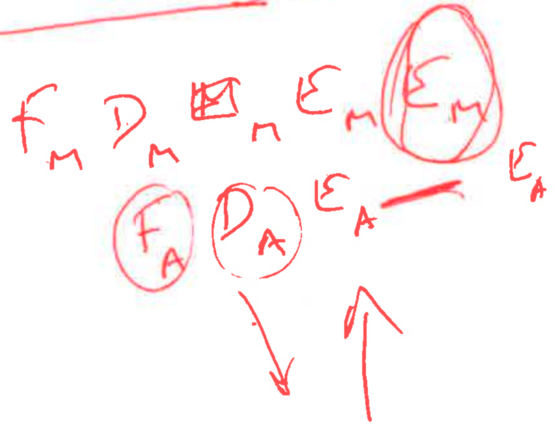
- *Unaligned access*
  - *LC-3b does not allow unaligned access*
  - *DEC: PDP-11 (no), VAX (yes), Alpha (no)* ← Same company! How come?
- *Data types (rich or lean)*
  - *Integers, floats of various sizes*
  - *Doubly-linked list, character string*
- *Addressing modes (rich or lean)*
  - *Indirect addressing*
  - *Autoincrement, postdecrement*
  - *SIB byte in x86*

Tradeoffs (with examples), continued

- **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*



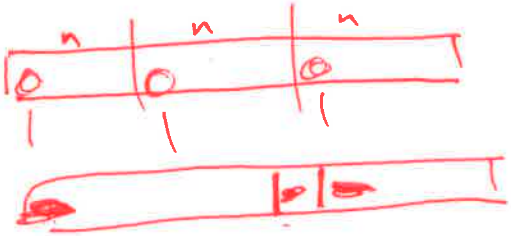
- **Word length** what does word length specify.
  - *VAX: 32 bits*
  - *x86: initially 16 bits, then 32 bits, today ~~64~~ bits* 512 bits
  - *CRAY 1: 64 bits*
  - *DEC System 20: 36 bits (LISP car, cdr for AI processing)*

SCOREBOARD

MIPS

24f

$$P = \frac{1}{\cancel{X} \times \text{CPI} \times t}$$



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)*** – Faruk's PhD dissertation
  - *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*

4K, 2M, 1G

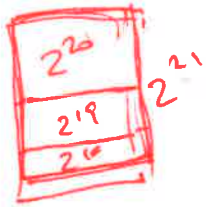
INTEL MOTOR

Guvenilir

ISCA 2019

$2^n, n \geq 12$

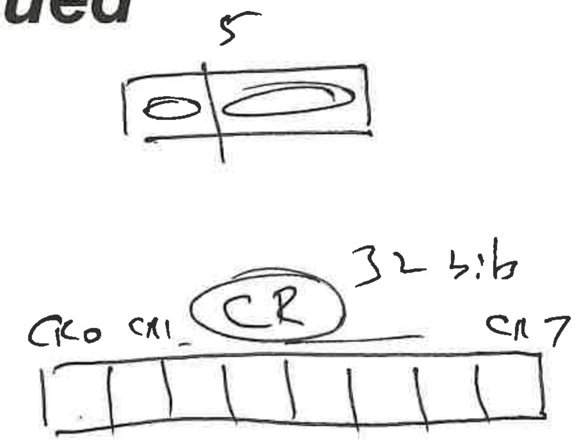
$\frac{2^{20}}{2^{12}} = 2^8$



MARK HILL

$2^{20} + 2^{19} + 2^{18}$

Tradeoffs (with examples) continued

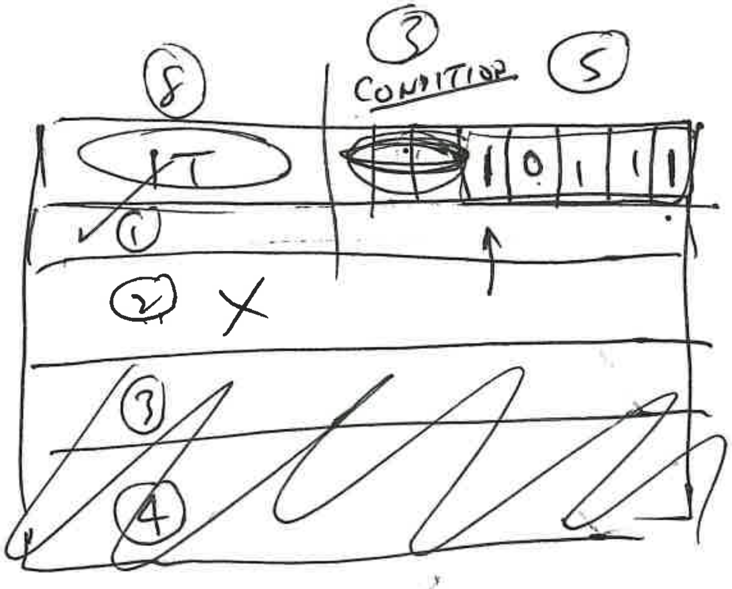


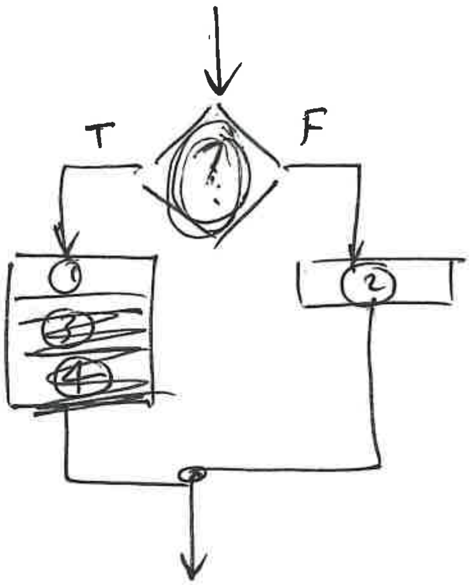
- ***Register set and size***
  - ***Many machine have 32 32-bit registers***
  - ***x86 now has 512-bit registers***
  - ***Itanium has one-bit predicate registers***
- **Condition codes vs using a general purpose register**
  - ***MIPS, CDC6600***

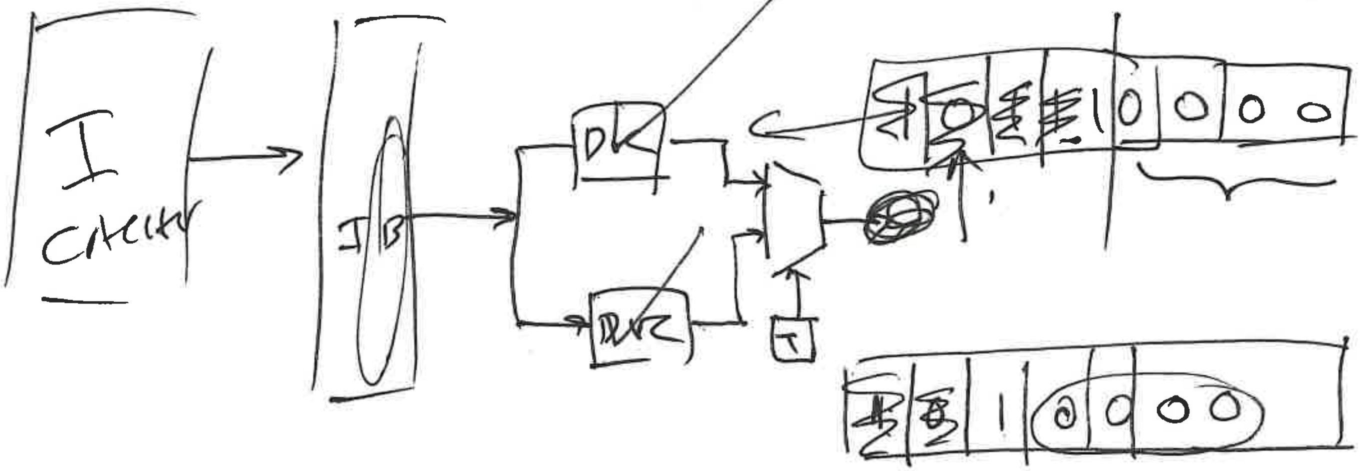
RS 6000 John Cocke IBM 801 ASPLOS 1982

- ***Rich instruction set vs Lean instruction set***
  - ***Hewlett Packard's RISC: HPPA has 140 instructions***
  - ***Orthogonal to RISC vs CISC*** — Register Window vs Set of GPRs
- ***Load/Store vs Operate in the same instruction***
  - ***LC-3b, ~~x86~~ are load/store***
  - ***x86 is not load/store***

Load/Store had a window of value.
Out of order changed that.



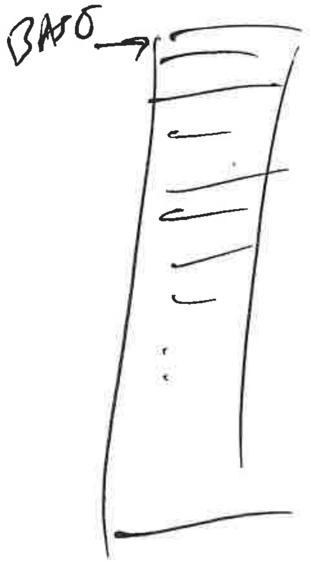


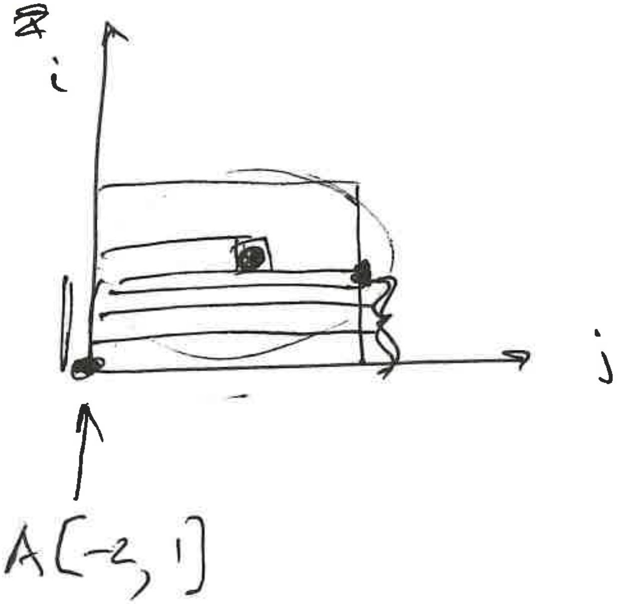


x86

~~Pentium~~ ~~Decode~~ IBM

Power PC = 615

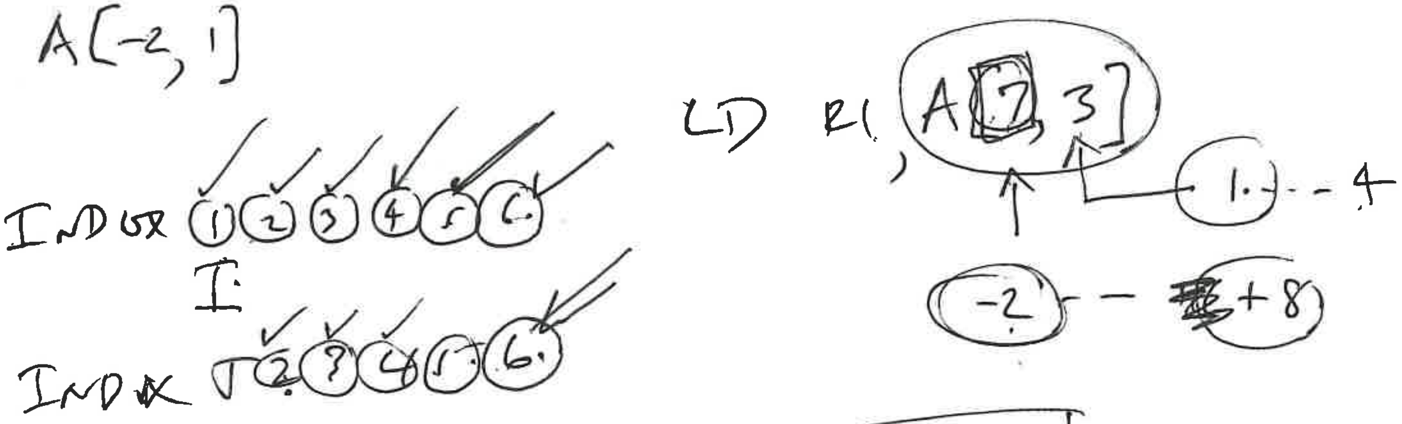




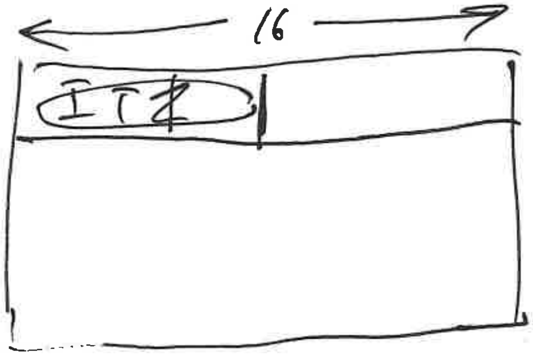
~~A[I, J~~

A[i, j]

DOPE VECTOR



LD R1, A



INDEX I
INDEX J

A

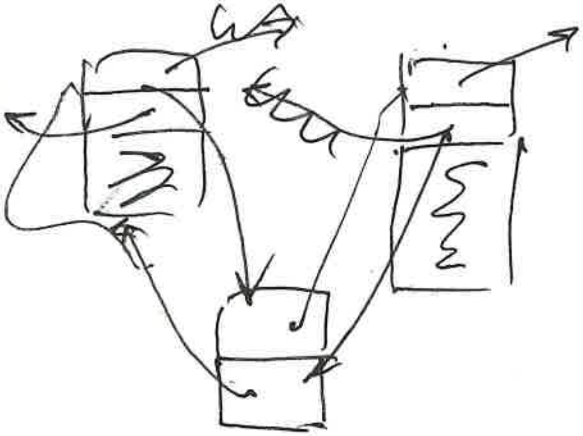
LD R1, A

INDEX 2
" 7
" 9
LD R1, A

A[2, 7, 9]

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

ALL Failed.

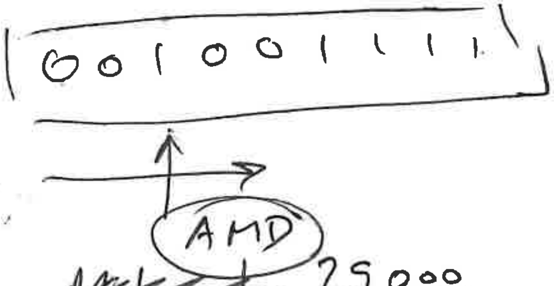
Why?

They ran like a dog!

- Predication
  - X86: CMOV
  - ARM: inst[31:28]
  - THUMB: IT block

4921

$49.

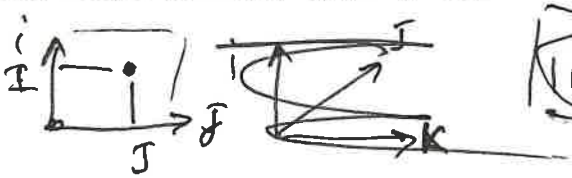


~~Motorola~~ 29000

- Support for multiple ISAs
  - ARM: T bit in the status register
  - VAX: Compatibility mode bit in the PSL

PPC with x86

LD R1, A

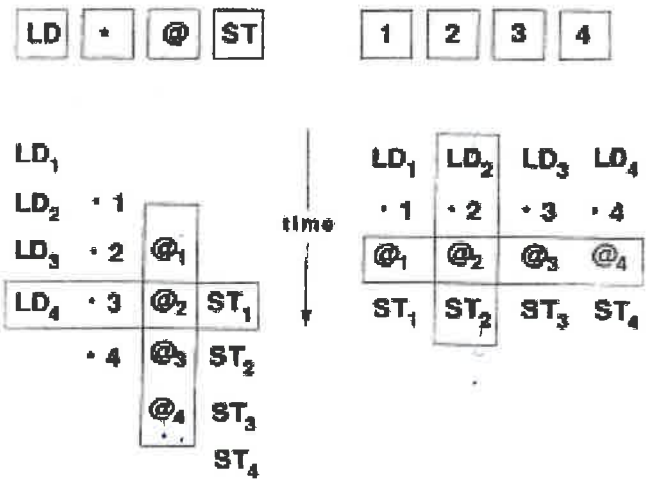


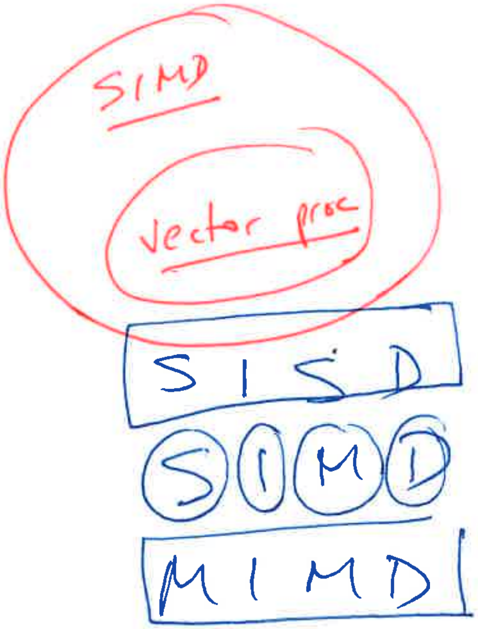
INDEX 1 2 3 4 5 6

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

SIMD

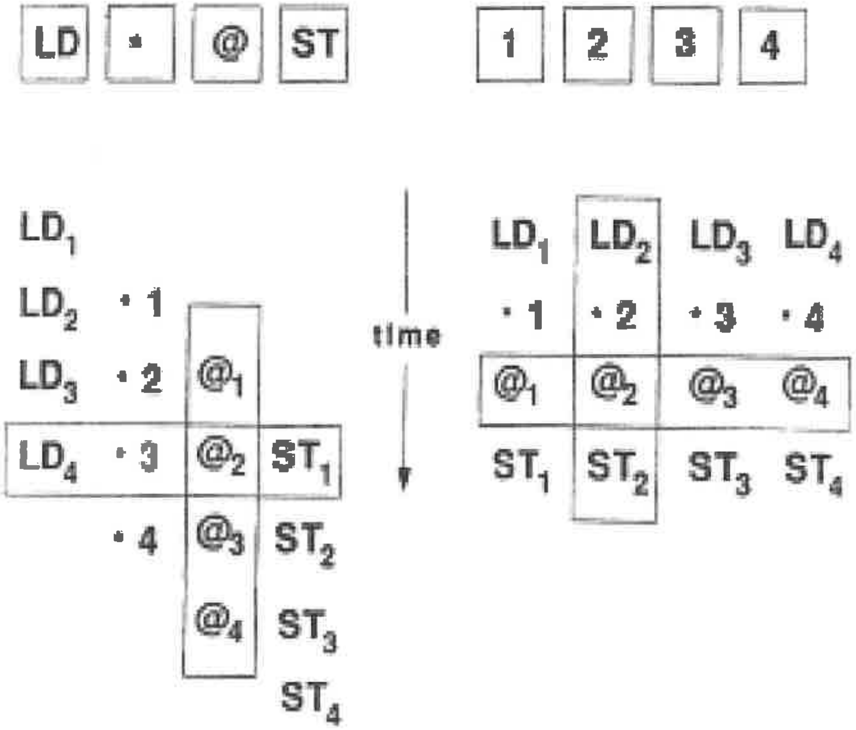
Vector Processors, Array Processors





SIMD

Vector Processors, Array Processors



LD	*	@	ST

1	2	3	4

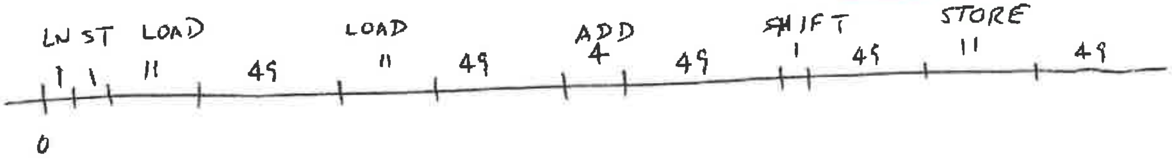
time

LD_1			
LD_2	*1		
LD_3	*2	$@_1$	
LD_4	*3	$@_2$	ST_1
	*4	$@_3$	ST_2
		$@_4$	ST_3
			ST_4

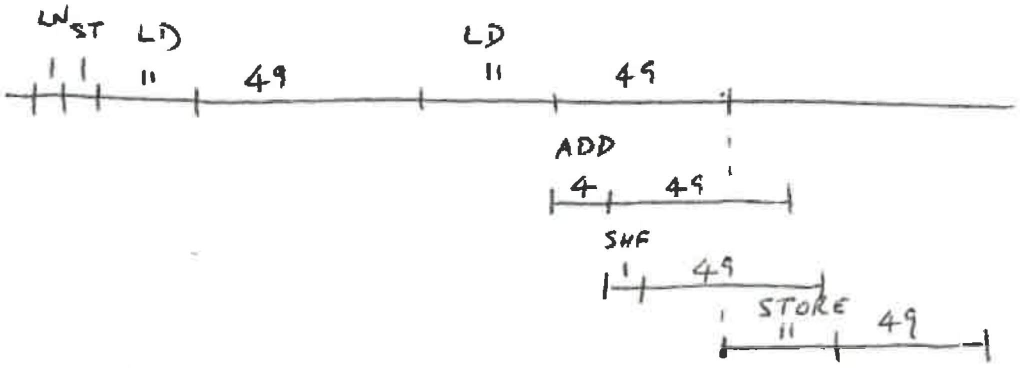
LD_1	LD_2	LD_3	LD_4
*1	*2	*3	*4
$@_1$	$@_2$	$@_3$	$@_4$
ST_1	ST_2	ST_3	ST_4

Vector processing example (continued)
Vector Processor Timing

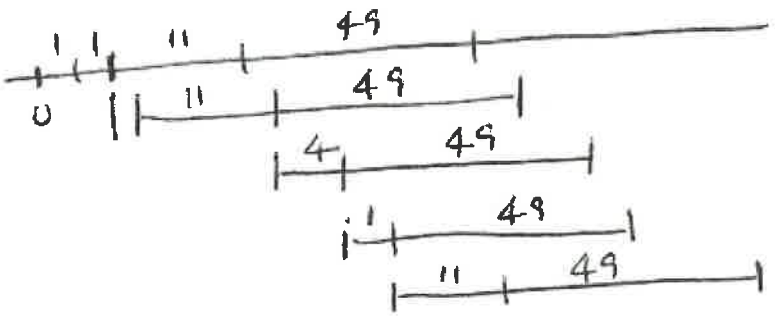
- Vector code (no vector chaining): 285 clock cycles



- Vector code (with chaining): 182 clock cycles



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



79 clock cycles

Vector processing example

- *The scalar code:*

```
for i=1,50
A(i) = (B(i)+C(i))/2; Vectorizable!
```

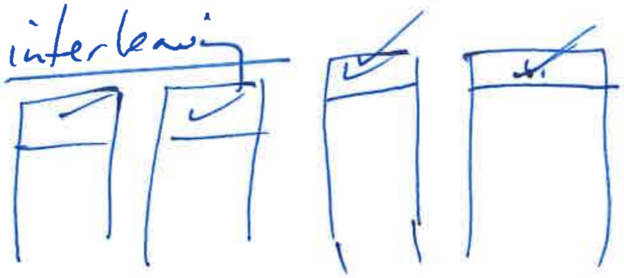
- *The vector code:*

```
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

How much interleaving

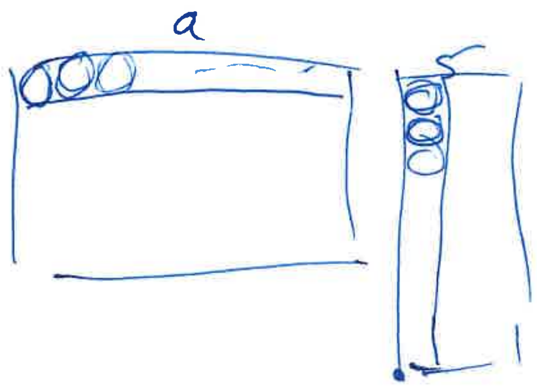


- **50 iterations of (LD, LD, Add, Shift, Store, Iteration Ctl)**
  - 50 x (Load, Load, Add, Shift, Store, Iteration_Ctl)
  - 50 x (11 + 11 + 4 + 1 + 11 2) = 50 x 40 = 2000 clock cycles

Vector Architecture

- **Vector Registers**
  - Each register has multiple components
- **Vector Instructions**
  - Loads/Stores
    - Multiple memory locations in one instruction ← What if you encounter a Page Fault
    - 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

FPD



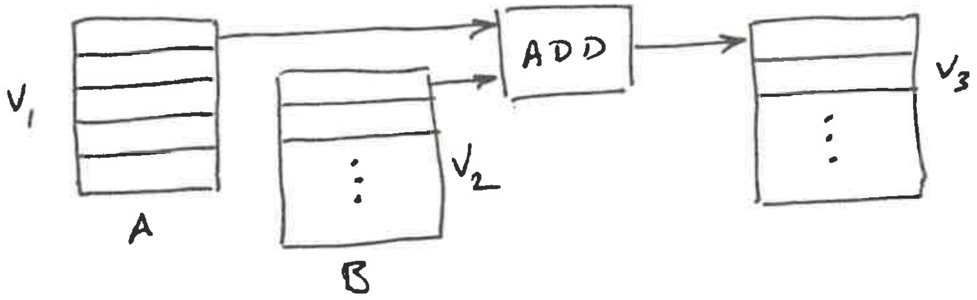
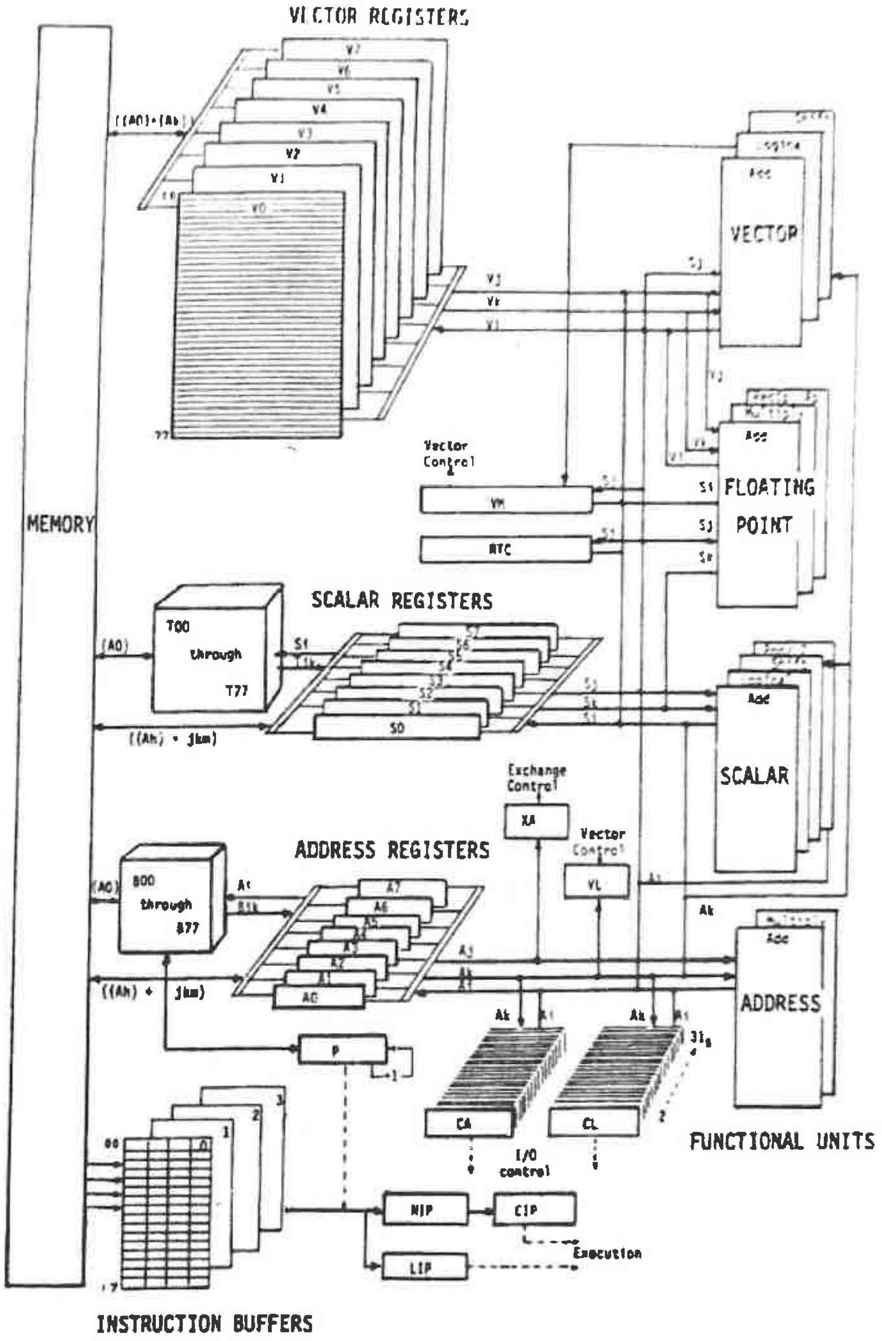


Fig. 5 Block diagram of registers

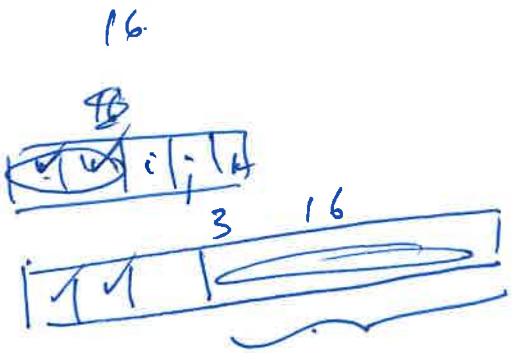


CRAY I

Russell, Comm ACM 1978

$\frac{A}{B}$

$\frac{1}{B} \times A$



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*

Many ISAs do that

  - *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*

Flynn : IBM 360 PDP-11

	31	25	20	15	12	7	6 0
R	FUNCT 7	RS2	RS1	FUNCT3	RD		OPCODE
I	IMMEDIATE [11:0]		RS1	FUNCT3	RD		OPCODE
S	IMMED [11:5]	RS2	RS1	FUNCT3	IMM[4:0]		OPCODE
U	31 IMMED [31:12]			12	RD		OPCODE

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

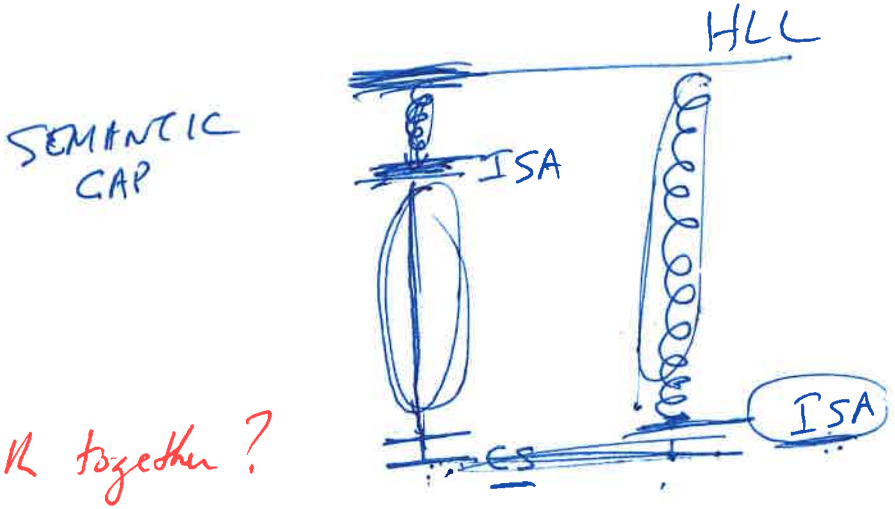
MAC (A, B, C)

$(A \times B) + C$

RISCV (characteristics)

A Buffet

- **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 ← Why both together?
  - V extension: Vector operations
  - E extension: Embedded Controller (RV32E)
  - G extension: A system, really (IMAFD)



AOB LE

RISCV

- *The 5th chip from Professor David Patterson's group*
  - *UC Berkeley*
  - *Nothing (really) in common with their other four risc chips*

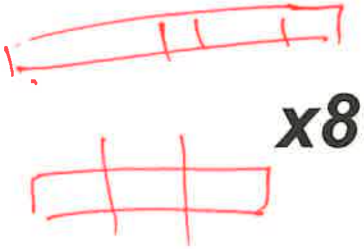
1,2. Simple
3. Add Smalltalk
4. Add cache coherence

- *Mostly handled by Professor Krste Asonovic*

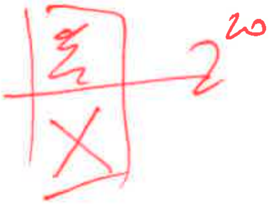
- *Major selling point: Open Source*

I asked Jim Keller: Why RISCV?

- *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*



x86



$$\frac{2^{21}}{2^{12}}$$

FARUK GUVENILIR

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

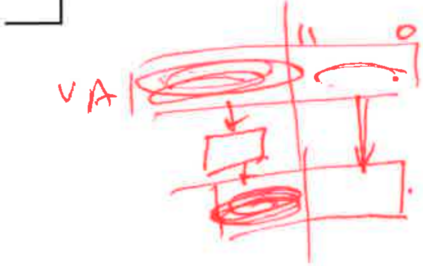
Prefix 1	Prefix 2	Prefix 3	Prefix 4	Opcode	Opcode 2	ModR/M	SIB	Address	Immed

up to 4 bytes

*

4KB
2MB
1GB

- *Characteristics*
  - *Rich set of addressing modes*
  - *Two-address machine*
  - *SSE extension* → Now AVX
  - *Not load/store*
  - *Three page sizes (4KB, 2MB, 1GB)* ⇒ Three TLBs BUT Faruk Guvenilir
  - *Register sizes: 8b, 16b, 32b, 64b, 128b, …*
  - *Example: AH, Ax, EAX, …*
  - *Memory: Byte addressable, 64 bit address space*, One bit addressable.



↑
Why?

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~~ That is part of the uarch)*

The LC-3b Instruction Set

<table>
<tr><th></th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>ADD$^+$</td><td colspan="4">0001</td><td colspan="3">DR</td><td colspan="3">SR1</td><td>0</td><td colspan="2">00</td><td colspan="3">SR2</td></tr>
<tr><td>ADD$^+$</td><td colspan="4">0001</td><td colspan="3">DR</td><td colspan="3">SR1</td><td>1</td><td colspan="5">imm5</td></tr>
<tr><td>AND$^+$</td><td colspan="4">0101</td><td colspan="3">DR</td><td colspan="3">SR1</td><td>0</td><td colspan="2">00</td><td colspan="3">SR2</td></tr>
<tr><td>AND$^+$</td><td colspan="4">0101</td><td colspan="3">DR</td><td colspan="3">SR1</td><td>1</td><td colspan="5">imm5</td></tr>
<tr><td>BR</td><td colspan="4">0000</td><td>n</td><td>z</td><td>p</td><td colspan="9">PCoffset9</td></tr>
<tr><td>JMP</td><td colspan="4">1100</td><td colspan="3">000</td><td colspan="3">BaseR</td><td colspan="6">000000</td></tr>
<tr><td>JSR</td><td colspan="4">0100</td><td>1</td><td colspan="11">PCoffset11</td></tr>
<tr><td>JSRR</td><td colspan="4">0100</td><td>0</td><td colspan="2">00</td><td colspan="3">BaseR</td><td colspan="6">000000</td></tr>
<tr><td>LDB$^+$</td><td colspan="4">0010</td><td colspan="3">DR</td><td colspan="3">BaseR</td><td colspan="6">boffset6</td></tr>
<tr><td>LDW$^+$</td><td colspan="4">0110</td><td colspan="3">DR</td><td colspan="3">BaseR</td><td colspan="6">offset6</td></tr>
<tr><td>LEA$^+$</td><td colspan="4">1110</td><td colspan="3">DR</td><td colspan="9">PCoffset9</td></tr>
<tr><td>NOT$^+$</td><td colspan="4">1001</td><td colspan="3">DR</td><td colspan="3">SR</td><td>1</td><td colspan="5">11111</td></tr>
<tr><td>RET</td><td colspan="4">1100</td><td colspan="3">000</td><td colspan="3">111</td><td colspan="6">000000</td></tr>
<tr><td>RTI</td><td colspan="4">1000</td><td colspan="12">000000000000</td></tr>
<tr><td>LSHF$^+$</td><td colspan="4">1101</td><td colspan="3">DR</td><td colspan="3">SR</td><td>0</td><td>0</td><td colspan="4">amount4</td></tr>
<tr><td>RSHFL$^+$</td><td colspan="4">1101</td><td colspan="3">DR</td><td colspan="3">SR</td><td>0</td><td>1</td><td colspan="4">amount4</td></tr>
<tr><td>RSHFA$^+$</td><td colspan="4">1101</td><td colspan="3">DR</td><td colspan="3">SR</td><td>1</td><td>1</td><td colspan="4">amount4</td></tr>
<tr><td>STB</td><td colspan="4">0011</td><td colspan="3">SR</td><td colspan="3">BaseR</td><td colspan="6">boffset6</td></tr>
<tr><td>STW</td><td colspan="4">0111</td><td colspan="3">SR</td><td colspan="3">BaseR</td><td colspan="6">offset6</td></tr>
<tr><td>TRAP</td><td colspan="4">1111</td><td colspan="4">0000</td><td colspan="8">trapvect8</td></tr>
<tr><td>XOR$^+$</td><td colspan="4">1001</td><td colspan="3">DR</td><td colspan="3">SR1</td><td>0</td><td colspan="2">00</td><td colspan="3">SR2</td></tr>
<tr><td>XOR$^+$</td><td colspan="4">1001</td><td colspan="3">DR</td><td colspan="3">SR</td><td>1</td><td colspan="5">Imm5</td></tr>
<tr><td>not used</td><td colspan="4">1010</td><td colspan="12"></td></tr>
<tr><td>not used</td><td colspan="4">1011</td><td colspan="12"></td></tr>
</table>

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*

Another trade-off

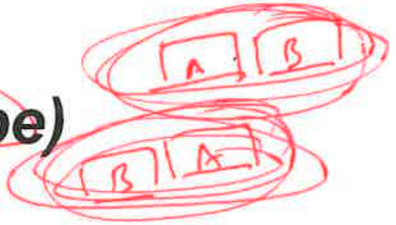
- *Privilege: 2 levels, supervisor, user*
- *Priority: 8 levels*

Highest level?

MACHINE CHECK
POWER FAIL

111
110
:
000

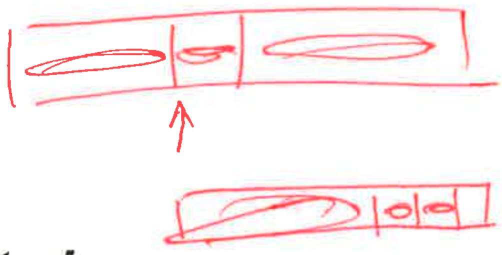
- *Instruction format: fixed length, 16 bits*
- *Endian-ness: little endian*
- *Instructions (opcode, addressing mode, data type)*
  - *Opcode (14 opcodes, including XOR, SHF, LDB)*
  - *Addressing modes (PC-relative, Register + offset)*
  - *Data types (2's complement 16 bit integers, bit vector)*
  - *Three-address machine*



NOT	DR	SR	1111,
[illegible]	✓	✓	2ND Source

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

PORTABILITY

IBM 360
PDP-11
x86
VAX

Mike Flynn

- 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!"

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.

What is the ISA?

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

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

Outline

- *What is it?*
  - *The interface between hardware and software*
  - *A specification* Contract
  - *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)*

Computer Architecture: Fundamentals, Tradeoffs, Challenges

Chapter 2: The ISA

Yale Patt

The University of Texas at Austin

Austin, Texas
Spring, 2023