What is computation?

- Computation
  - Transformations (operations)
  - Conditionals
  - Data
- Correctness

How can you express computation?

- Basic ops
  - A=b+c
- Variables
- If…then.. While…
- Classes (data structures)
- Functions (recursion)
- Thread

Registers

- Fast
- Temporary storage
- Memory addressing

Memory addressing modes

- Memory
  - Direction: la <memory>
  - Memory: [$reg], [$reg+constant]
  - [$reg1 + $reg2]
    - A = b[I]
      - Reg1 = b
      - Reg2 = I
- Branches
  - PC relative
  - Absolute

Instructions

- Minimum set:
  - Nand, sub, branch-less-than-zero
- Real set:
  - Memory I/O
  - Jump, branches
  - Arithmetic

Return of the CISC

- Cryptographic instructions
  - Strange bit twiddles
- Domain specific processing
  - Silicon now is cheap / free
  - Bundle of computation common to domain
  - Different model
    - DSP
  - Don’t want to be on critical-path
  - Would like to compile for it

Encodings

- Desirables:
  - Uniformity
    - Less decoding time
  - Compactness
    - Potential down side
    - Less of everything
      - Less registers
      - Smaller constants
      - Optimizing for common case

When is CISC good?

- No compilers
- When RISC is over, go CISC
  - Everything old will be new again
- Slow memory
- Expensive memory
- Language specific computing SYMBOL, LISP machines
  - Limited languages

When is RISC good?

- Cheap memory
- Start again
- Logic is expensive
  - Area constraints
- Lower power because of less control logic
- You have compilers

What was 1980 like, for RISC?

- Conditions are right
  - Cheap enough memory
  - VLSI
    - Carp / Meade work (Cal-Tech)
- Studies that show VAX instructions were not being used
  - 65 of 100's of x86 instructions get used

What else might we want to express?

- Vector
  - Exposes parallelism
  - <a> = <b> + <c>
  - Exposes regularity
  - Reduces the need for speculation
- VLIW
  - Exposing parallelism
- Compiling for these?
  - Easier with global variables, no pointers