# *Computer Architecture*
# *Random Thoughts After 60 years in the trenches*

***Yale Patt***
***The University of Texas at Austin***

***Computer Engineering Seminar***
***Academia Sinica, Taipei***
***November 13, 2026***

# *My thoughts*

- *Computer Architecture was dead for 30 years*
- *Work on Challenging Problems (Avoid LPUs)*
- *Scholarship versus Research*
- *Can we believe the claims*
- *Do we need to include a baseline in our papers?*
- *Moore's Law, Dennard Scaling gone. What does it mean?*
- *The Transformation Hierarchy)*
- *Top Down or Bottom up?*
- *Are you a Hardware Person or a Software Person?*
- *What is the CORRECT first course in computing for ECE?*
- *The CORRECT 1st computing course for all engineers*

# *My thoughts (continued)*

- ***Heterogeneity or Homogeneity – What is the answer?***
- ***The temptation to add features***
- ***Why much better branch prediction accuracy does not yield much better IPC?***
- ***Quantum***
- ***AI, the elephant in the room***
- ***Some run-time stuff close to the metal***
- ***Security***
- ***RISCV***
- ***The Apple Approach***
- ***The microprocessor of 2036***
- ***The vonNeumann Machine model is not dead***
- ***Be generous in giving credit***

*Computer Architecture was dead for 30 years*

# *What we accomplished in those 30 dead years*

- ***Branch Prediction***
- ***Speculative execution (and recovery)***
- ***Special execution units (FP,  Graphics, MMX)***
- ***Out of order execution (and in-order retirement)***
- ***Pipelining with hiccups (R/S, ROB)***
- ***Wide issue***
- ***Trace cache***
- ***SMT, SSMT***
- ***Soft errors***
- ***Energy as a first class consideration***

***Clearly Computer architecture was not dead for 30 years***

# *Examine what people say*

***If someone who should know better says something that appears absurd, he may have a different agenda.***

***Don't accept on face value what is said, regardless who is saying it. Challenge what does not make sense.***

## *Work on Challenging Problems (Avoid LPUs)*

## *Work on Challenging Problems (Avoid LPUs)*

- *LPU = Least Publishable Unit*
- *With LPUs you will succeed and publish lots of papers*
- *But nobody will care.*
- *A guarantee you will never move the needle*

- *If you want to move the needle,*
  *work on challenging problems*

# *Scholarship vs Research*

# *Scholarship vs Research*

- ***Most disciplines view research as a way to understand***
  - ***I would call that scholarship***
- ***Engineers see the world differently***
  - ***Step 1 in research is scholarship, shedding light on what is.***
  - ***But there needs to be a Step 2: Build a widget***
- ***No credit for step 1 if you don't come up with a step 2***
  - ***Some dismissed HPS because of our bad checkpoint scheme***
  - ***I would argue Step 1 is the more important step***
  - ***Develop the insights as to what the problem is***
  - ***Publish that and invite others to join us fin finding the widget***
  - ***Others call new knowledge research. We call it scholarship.***
- ***To be research we need to have  a widjit of value***
  - ***Without which the paper is rejected***
  - ***Seznec's AHEAD predictor was unlucky, he had a bad widjit***

***Can we believe the claims?***

# *Can we believe the claims?*

- ***What should be provided to justify the claims?***
  - ***The whole experimental set-up***
  - ***Or, do we simply trust the authors' word?***
- ***Disclosing too much makes life easier for followers***
  - ***Is that fair to the person who first had the idea?***
  - ***But that is how science moves the needle***
- ***Too many comparch results are never replicated***
  - ***No carrot for replicating someone else's results***
  - ***Will we ever be a science until that happens?***

***Do we need to include a baseline***

# *Do we need to include a baseline*

- ***Anecdotal data from recruiting junior faculty lately***
  - ***Too many job talks give performance numbers***
  - ***But no reference to state-of-the-art before the candidate***

- ***Would you be impressed if I told you***

***I can run the Marine Corps Marathon***
***1000 times faster than your grandmother?***

***Your work must be measured on the basis of how much better you do than the state-of-the-art did before you came along.***

***Moore's Law, Dennard Scaling gone. What's it mean?***

## *Moore's Law, Dennard Scaling gone. What's it mean?*

- *No more faster transistors*
- *No more increase in the number of transistors*
- *We won't even be able to have them all on at once*
- *Counting on 3d stacking is silly.*
- *So how can we stay on the performance curve?*

# *The Transformation Hierarchy*

- ***MIT says: Plenty of room at the top***
  - ***Exploit algorithms, languages, compilers***
- ***I say: Still plenty of room at the bottom***
  - ***50 billion transistor chips before we are done***

***Both of us are referring to***
***my Transformation Hierarchy***

Problem

---

***Algorithm***

---

***Program***

---

***ISA (Instruction Set Architecture)***

---

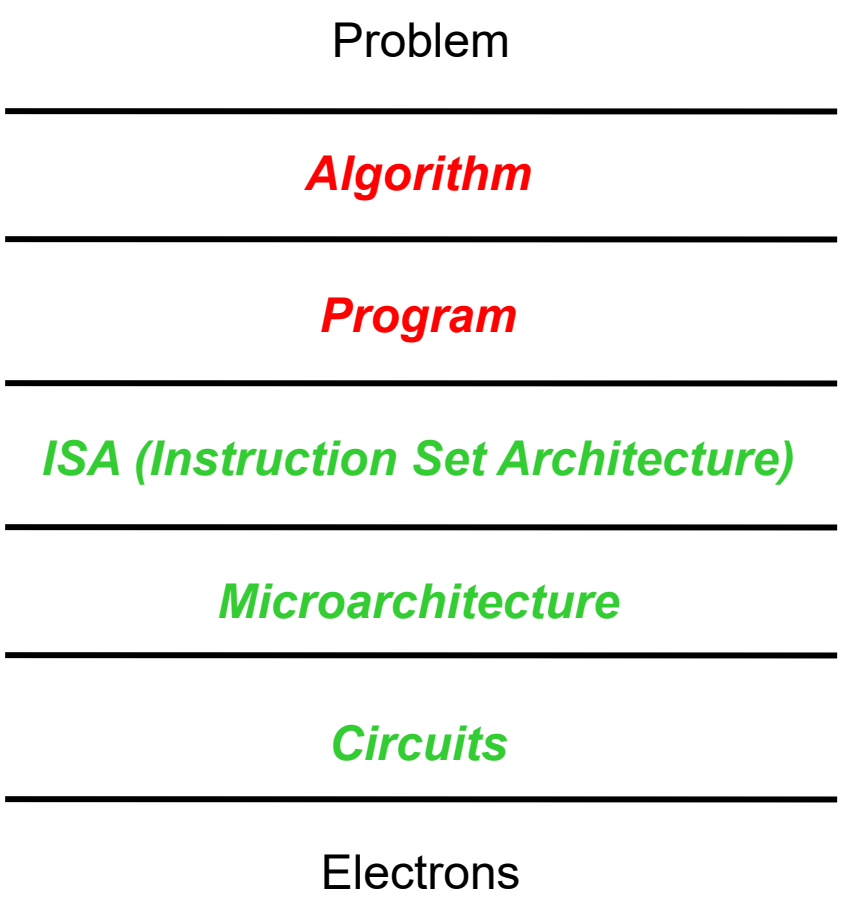
***Microarchitecture***

---

***Circuits***

---

Electrons


Problem
Algorithm
Program
ISA (Instruction Set Architecture)
Microarchitecture
Circuits
Electrons

# *Top-down or Bottom-up*

# *Top-down or Bottom-up*

- ***Bottom-up for understanding***
  - ***Because you are learning***
  - ***And it is helpful to build on what you know***
- ***Top-down for design***
  - ***Because you already understand the components***
  - ***And bottom-up would yield the old design with a pimple on it.***

***Bottom up or Top down: Don't be a Top-up!***

*Are you a Hardware Person or a Software Person?*

# *Hardware or Software (Don't limit your education)*

- *My first semester at Berkeley (1979)*
  - *I taught Data Structures and Computer Organization*
  - *Are you a hardware person or a software person*
  - *I'm a person*
- *We do not know what you will need to know downstream*
- *Google's Tensor Processing Unit for inferencing*
  - *The problem is machine learning*
  - *Cliff Young is an architect (PhD at Harvard in the 1990s)*
  - *Took HT Kung's course*
  - *Google's TPU has a systolic array*

# An example: Find First (1st bit set in a bit vector) AMD 29000 (AMD's RISC machine)

- Do it in Software

```
        LD  R1, #15
        LD R2,A          ; R2 contains the bit vector
AGAIN   BRn FINISH
        ADD R1, R1 - #1
        ADD R2,R2,R2
        BRnzp AGAIN
FINISH  HALT             ; R1 contains bit number
A       .BLKW            ; The bit vector, loaded first
```

  - More than 30 clock cycles
- Do it in Hardware
  - A fraction of one clock cycle: a priority encoder

***And, while we are at it:***

***The CORRECT first course in Computing***

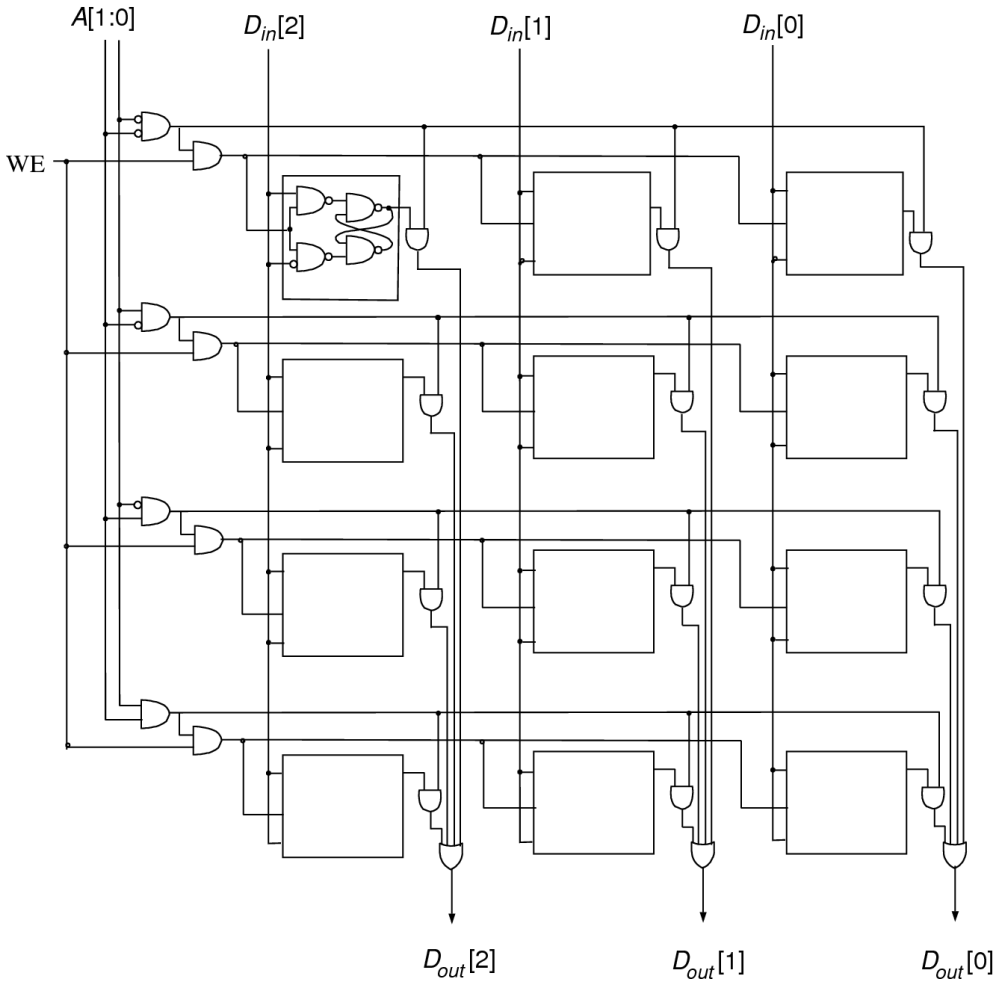
***for ECE majors***

# *The CORRECT first course in computing for ECE majors*

- ***The "motivated bottom up approach"***
  - ***Build on what students know, minimize memorizing***
  - ***Start with the transistor, then gates, structures, FSM, and computer***
- ***Programming AFTER they know how the computer works***
  - ***Students understand how the computer processes their programs***
  - ***Students are better able to debug their own programs***
  - ***Not the case for students taking "Programming in HLL X"***
- ***The ECE curriculum***
  - ***Each relevant course deals with some aspect of the computer***
  - ***The correct 1st course builds the foundation from day one.***
- ***About 400 adopters do it my way***
  - ***About 5,000 professors still doing it wrong***

# *The Freshman Course*
# *(At Michigan we called it EECS 100)*
# *(At Texas it is called ECE306)*

- ***Start with what they "know"***
  - ***The transistor as light switch***
  - ***Not quantum mechanics!***
- ***Choose a computer model that is simple (The LC-3)***
  - ***As the genius said: simple, but still rich***
- ***Continually build on what they know***
- ***Continually raising the level of abstraction***
- ***Memorizing as little as absolutely necessary***
- ***Trying very hard to not introduce magic***


A[1:0]
$D_{in}$[2]
$D_{in}$[1]
$D_{in}$[0]
WE
$D_{out}$[2]
$D_{out}$[1]
$D_{out}$[0]

# The LC-3 ISA



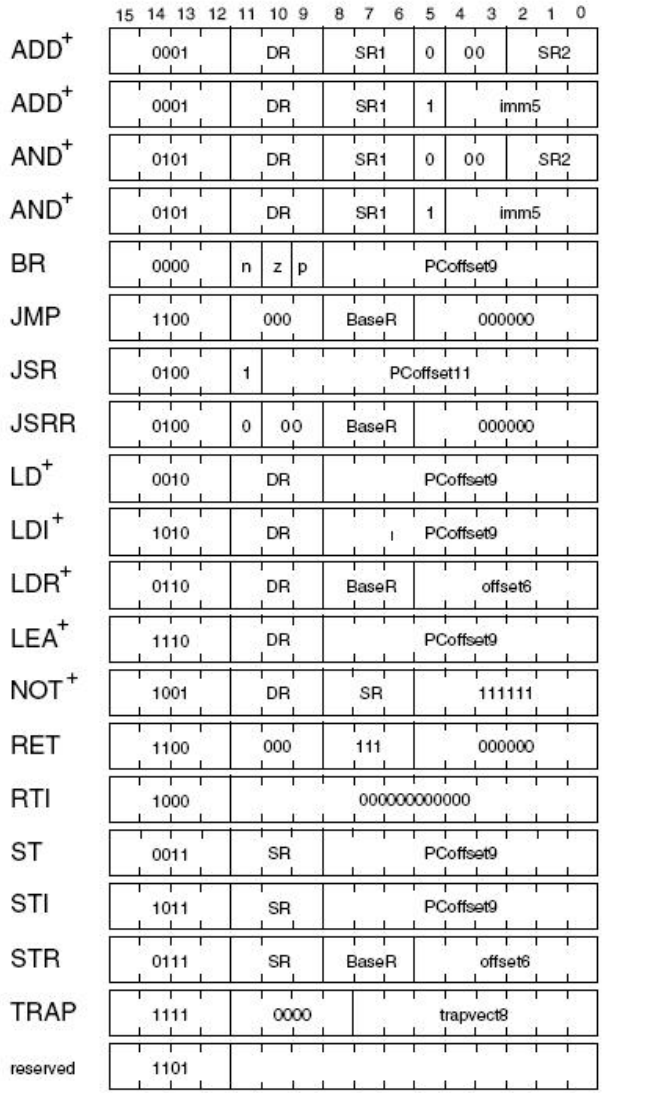



Figure A.2 Format of the entire LC-3 instruction set. Note: + indicates instructions that

# The State Machine

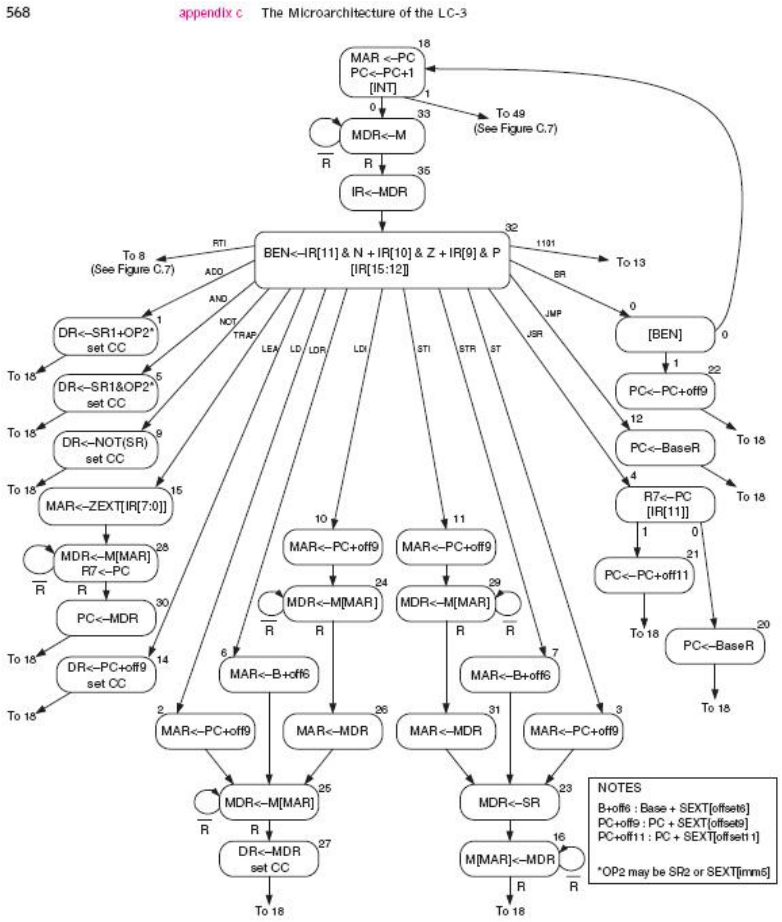



Figure C.2 A state machine for the LC-3

# The Data Path



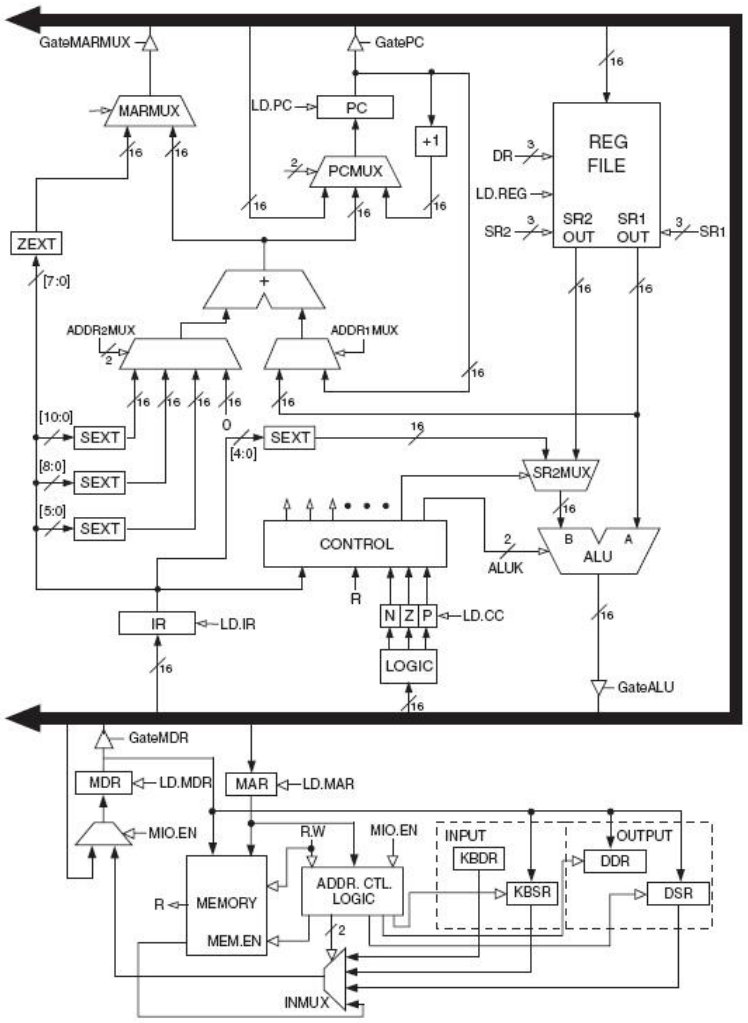



Figure C.3 The LC-3 data path

# *…And, actually, for all engineering students*

- ***Programming in X is the wrong answer***
  - *Approach almost always top-down, leads to excessive memorizing*
  - *Provides no insight into important tools (Matlab, for example)*
  - *No insight into how embedded processor connects in the system*
  - *No insight into Design, Tradeoffs, State (bedrock of all engineering)*
- ***Why "Intro to Computing" is essential to all eng'g curricula***
  - *Serious "Intro to Computing" emphasizes Design, Tradeoffs, State*
  - *No time for Word, Excel, Web Surfing (they'll get that on their own)*
  - *An engineer must have a deeper relationship with computers*
- ***What the engineer needs to know vis-à-vis computers***
  - *Just about every system today comes with an embedded computer*
  - *An engineer needs to know how the computer works*
  - *How the numbers are represented, how an algorithm works*
  - *i.e, to be able to design the system, to be able to exploit this "tool"*

## *After the LC-3, What ISA comes next*

- ***The LC-3b***
  - *Byte-addressable memory*
  - *A Coincidence: It closely resembles RISCV i32*
  - *Except it has 16 bit instructions, 14 opcodes*

- ***Additions to the LC-3***
  - *Added a serious Shift instruction*
  - *Added XOR (compatible with LC-3 NOT)*
  - *Removed LDI and STI*
  - *Added LD and ST byte*

# *Homogeneity or Heterogeneity: what is the answer?*

## *Heterogeneity is the answer*

- ***HIGHER PERFORMANCE AND LESS ENERGY***
  - ***Not a tradeoff***
  - ***You win on both***

- ***But you must select hetero judiciously***

  ***You can not make everything heterogeneous***

# *Why did it take so long to catch on*
# *There were plenty of examples*

- *The PDP 11/60 had writable control store*
- *Morphcore*
  - *Enabled tailoring each structure to what is needed*
  - *Heavyweight for ILP, Mickey mouse array for TLP*
  - *One Mickey mouse for No LP!*
- *American football -- Eleven players on the field*
  - *How many quarterbacks?*
- *What took the rest of the world so long to catch on?*

*Hetero creates the temptation to add features*

## *Heterogeneity provides a bounty*

- *Higher performance structures at lower energy cost*
- *It is easy to get carried away*
- *BUT if you select correctly the functions to accelerate*
- *And if you can turn off the power when not in use,*
- *You can exploit heterogeneity*

## *Hetero creates the temptation to add features*

- ***A consequence of designing for the specific need***
  - ***Higher performance***
  - ***Less energy consumed***
- ***Which begs the question:***
  - ***Should we add the features to the ISA***
  - ***Should we add the features to the microarchitecture***

## *Some features have been added to the ISA, Others were added to the microarchitecture*

- *In the microarchitecture*
  - *Branch prediction*
  - *Wider issue*
  - *Predicated execution*
  - *Extra memory pipes*
  - *FPGAs (big increase in flexibility at small cost?)*
- *In the ISA*
  - *Predicated execution*
  - *Unaligned access*
  - *Register windows*
  - *Delayed branch*

## …*so I must say: BE CAREFUL*

- ***Add something to the microarchitecture: No problem!***
  - ***If a bad idea, discard it on the next implementation***
- ***Add something to the ISA***
  - ***You are probably stuck with it forever***
- ***Delayed branch and register windows***
  - ***Classic examples!***

# *What next for Heterogeneity?*

# *What next for Hetero*

- ***Chiplets***
  - ***An additional degree of freedom for the substrate***
- ***Expose the microarchitecture to the compiler***
  - ***Better performance at the expense of portability***
  - ***Could help us cure cancer, predict tsunamis***
    - *:*

# *When I suggest exposing the microarchitecture to the compiler*

- ***I get enormous pushback:***
  - ***What about portability?***
  - ***Without portability, the economics isn't there***
  - ***Industry won't do it***

- ***I say portability be damned!***
  - ***Some things are too important to have a price tag***
  - ***If industry won't do it, perhaps governments will***
  - ***But if we can get performance on very important problems***
  - ***By combining hetero uarch with exposing it to the compiler,***
  - ***I think that is worth pursuing***

***Why much better branch prediction accuracy does not yield much better IPC***

## *Why much better branch prediction accuracy does not yield much better IPC*

- *Most bottlenecks are due to multiple causes*
- *Fixing one cause may yield no visible improvement*
- *When the last cause is fixed, BINGO!*
  - *I keep hoping that someday reviewers will actually get it.*

# *Quantum*

# *Quantum*

- ***The paradigm is intriguing***
  - ***It needs to be studied, but not at the current funding level***
- ***When will it be able to do useful work?***
  - ***How many qbits will it take to do something useful?***
  - ***How long will a qbit have to be stable to be useful?***
  - ***What temperature will be required?***
- ***What will it be useful for?***
  - ***The poster child has been RSA encryption***
  - ***But crypto people are now doing post quantum crypto***
  - ***Maybe quantum will be useful for simulations of chemistry***
  - ***But at what level of funding?***
- ***Still the hype continues***
  - ***Funding levels are absurd, faculty positions are plentiful***
  - ***No country wants to be left behind***
  - ***What happens if the bubble bursts?.***

# *AI, the elephant in the room*

# *AI, the elephant in the room*

- *AI consumes more and more resources*
- *No question some of it is producing amazing results*
- *Other stuff is just brute force consuming energy*
- *But that brute force may take an unintended path*
- *Producing unintended new knowledge*
- *AI can easily absorb all the energy available*
- *Therein lies the dilemna*
- *Energy is a scarce resource*
- *But could produce unexpected benefits*
- *Our challenge: What if anything should we do.*

# *Some run-time stuff close to the metal*

# *Some run-time stuff close to the metal*

- ***Low level threads resident on chip***
  - ***Helper threads to improve chip infrastructure***
    - ***Branch Prediction***
    - ***Cache Replacement Policy***
  - ***Helper threads to dynamically recompile***
- ***Both can result in higher performance***

# *Security*

# *Security*

??

RISCV

# *RISCV*

- *The magic word is "open source"*
- *Will architects accept the open source*
- *Or will each architect wish to grow his/her own*
- *Mini-ISAs provide some slack*
- *With 13 mini-ISAs, 8000 possible ISAs*
  - *One must be i32 or i64 or i128*
  - *Most opt for i32; 37 opcodes, 32bit address space*
  - *Why no pedagogical ISA for initial training?*
- *For example, the LC-3b – 14 opcodes, 16bit address*
- *Much more effective than the i32*

# *The Apple Approach*

# *The Apple Approach*

- ***Apple owns all parts of the system***
- ***Vertically integrated***
- ***Apple does not have to pay for interfacing***
- ***Which in Apple's case is unnecessary***
- ***Yielding higher performance***

# *The Microprocessor of 2036*

# *The microprocessor of 2036*

- ***Lots of Accelerators, probably some with FPGAs***
  - ***Power turned off when not in use (like the Refrigerator in the NFL)***
- ***Chiplets – the substrate yields another degree of freedom***
- ***Pay attention to interconnect,***
  - ***Lest you throw way all the benefit the accelerators deliver***
- ***The vonNeumann machine model alive and well***
- ***Part of the run-time system close to the metal***
- ***SSMT (helper threads) for monitoring, dynamic compiling***
- ***At least two distinct interfaces to the metal***
  - ***One for those who get it, and one for the clueless***

*The vonNeumann Machine Model is not dead*

# *The vonNeumann Machine model is not dead*

- ***The microprocessor will have accelerators***
  - *Each can do its own thing, creating chaos*
  - *UNLESS something keeps them in order*

- ***The instruments in a world class orchestra***
  - *Require a conductor to maintain order*
  - *The on-chip accelerators need something to keep order*
  - *The vonNeumann machine is their conductor*

***Be generous in giving credit***

# *Be generous in giving credit: Always recognize others' contributions*

- ***Branch prediction, Helper threads***
  - ***Branch prediction: Kimming So, IBM***
  - ***Helper threads: Michel Dubois, USC***

- ***If you think you're the only smart person, take a nap***
  - ***It never hurts to give credit***

# *Finally,*

- ***Don't be afraid to exploit the microarchitecture to solve important problems.***
- ***Be careful about adding features to the ISA. They will be with you for a very long time.***
- ***Don't be a "top-up" person. Understand the underlying structures so you can exploit them more effectively.***

***Xie Xie!***