Ridiculously Simple Computers
(& simulation thereof)

Bruce Jacob

Cyber Science Department
United States Naval Academy

Disclaimer: The views you are about to hear are the ravings of a madman and do not necessarily represent the views or opinions of the U. S. Naval Academy, Department of the Navy, or Department of Defense (DoD) or any of its components.

Bruce Jacob

United States
Naval Academy

My Goal = Teaching Platform**

Fundamental understanding

- **Distillation of components to essence**
- **Everything is there to serve a purpose (education => explain that purpose)**

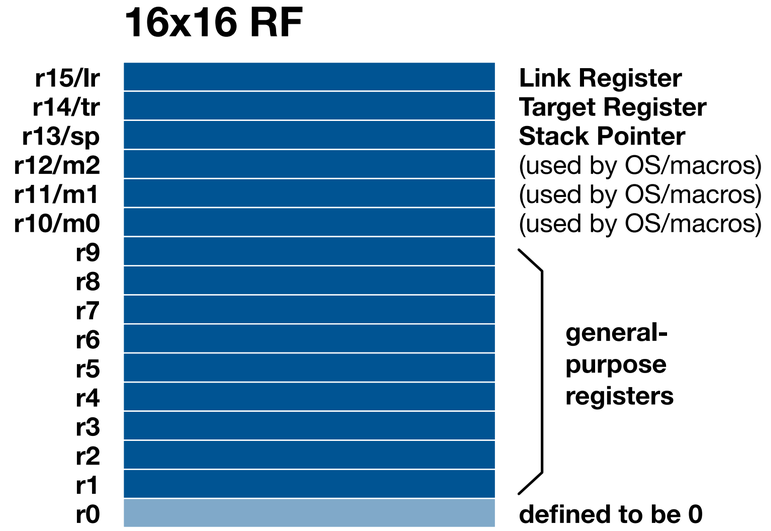
Observations:

Same-ish goals as HW/SW Co-Design

**Results echo yesterday's panel session

RiSC-16 — Student Model

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD	0000				reg A				reg B				reg C			
SUB	0001				reg A				reg B				reg C			
AND	0010				reg A				reg B				reg C			
OR	0011				reg A				reg B				reg C			
JALR	0111				reg A				reg B				0000			



Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDI	1000				reg A				reg B				0000				imm16															
LW	1001				reg A				reg B				0000				imm16															
SW	1010				reg A				reg B				0000				imm16															
BEQ	1011				reg A				reg B				0000				imm16															
HALT	1111				1110				1111				1111				imm16															

9 Instructions
+ HALT

Bruce Jacob

United States
Naval Academy

RiSC-16 + Compiler + Kernel

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD	0000				reg A				reg B				reg C			
SUB	0001				reg A				reg B				reg C			
AND	0010				reg A				reg B				reg C			
OR	0011				reg A				reg B				reg C			
—	0100				reg A				reg B				reg C			
—	0101				reg A				reg B				reg C			
—	0110				reg A				reg B				reg C			
JALR	0111				reg A				reg B				0000			
BOOL	0111				reg A				reg B				0001			
INV	0111				reg A				reg B				0010			
S.LT	0111				reg A				reg B				1100			
S.LTE	0111				reg A				reg B				1101			
S.EQ	0111				reg A				reg B				1110			
TRAP	0111				0000				1111				1111			

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDI	1000				reg A				reg B				0000			
SHRI	1000				reg A				reg B				0001			
LW	1001				reg A				reg B				0000			
LW.PA	1001				reg A				reg B				0001			
SW	1010				reg A				reg B				0000			
SW.PA	1010				reg A				reg B				0001			
BEQ	1011				reg A				reg B				0000			
BNE	1011				reg A				reg B				0001			
JALI	1011				reg A				0000				0010			
—	1100				reg A				reg B				reg C			
—	1101				reg A				reg B				reg C			
—	1110				reg A				reg B				reg C			
RFI	1111				0001				1111				1111			
HALT	1111				1110				1111				1111			

20 Instructions
+ HALT

Bruce Jacob

United States
Naval Academy

C Compilers ~~for~~ by Dummies

`int` is the only data type bahahahahaha …

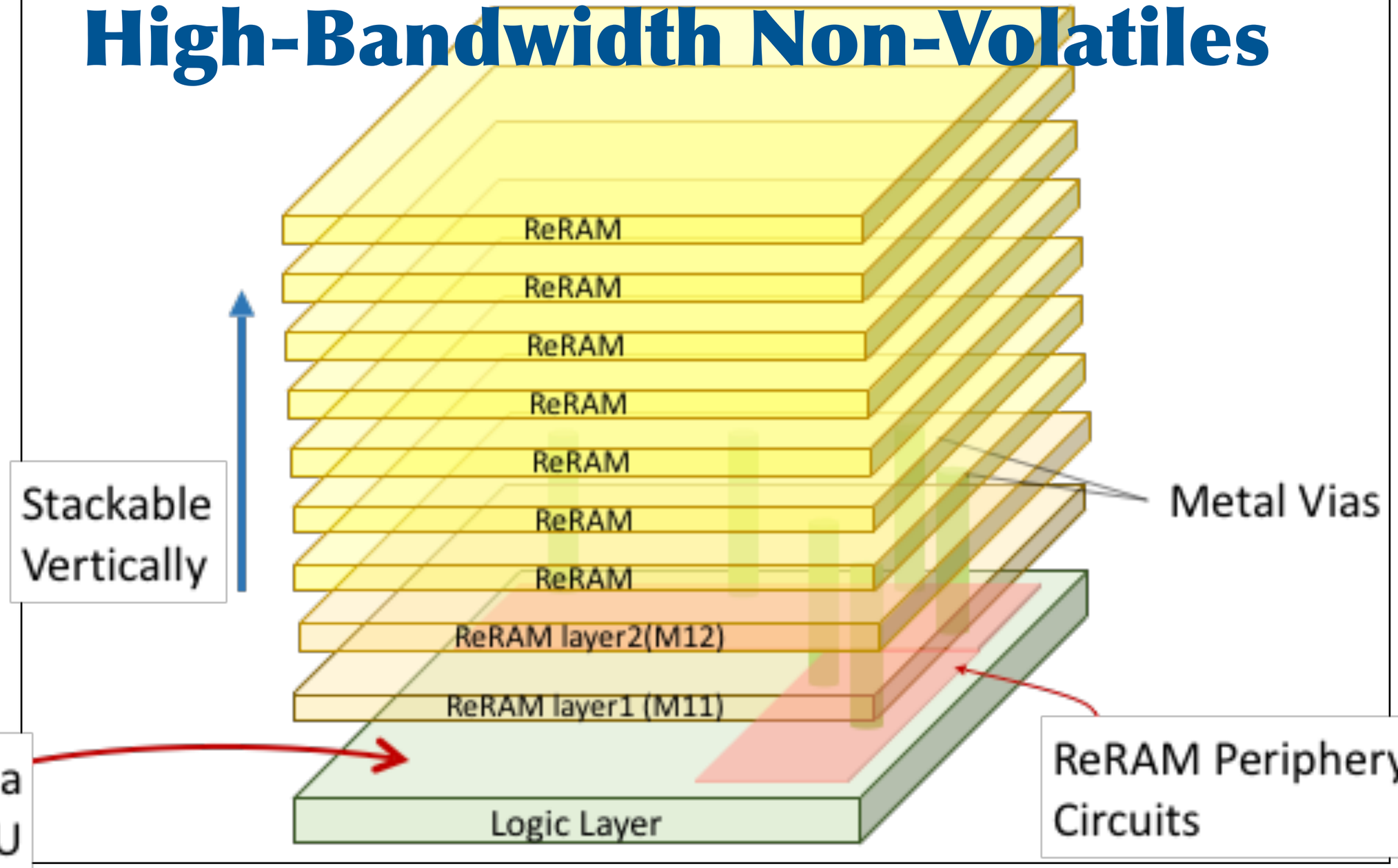
```
int ip = &data_structure;
*ip = value;

int variable;    <- 'variable' = its address
int array[5];    <- 'array' = its address
int ip = &array; <- what does ip[3] mean?

=> ip[3] refers to (&ip)+3, not (*ip)+3

   int ip = &array;
   ip->[3] = value;

etc …
```

ridiculously simple but ~1000 lines of code

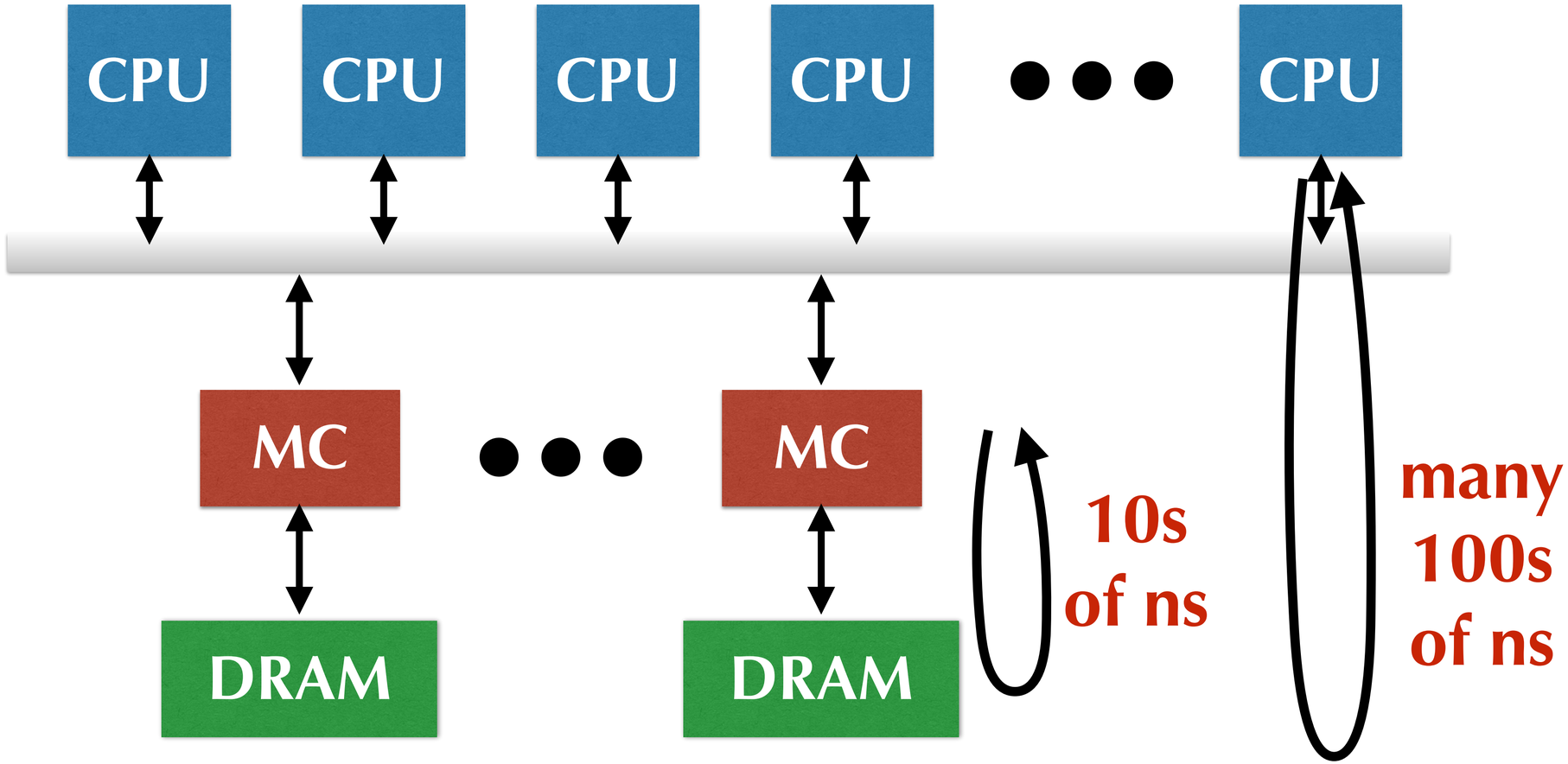
Kernels … w/ ***Sustainability Hook!!!***

- **Future of main memory = nonvolatile**
  (price per bit + stay-alive power)
- **… it will therefore be BIG (TBs)**
- **… it is also likely to be on-die**
  (monolithic integration)
- **… with many, many access points and 100s of cores using them**
  (multicore is here to stay)

High-Bandwidth Non-Volatiles



Memory Latency 101



Limit = *core-to-controller* ratio

(<1 is good, 1–2 is okay, >2 is bad)**

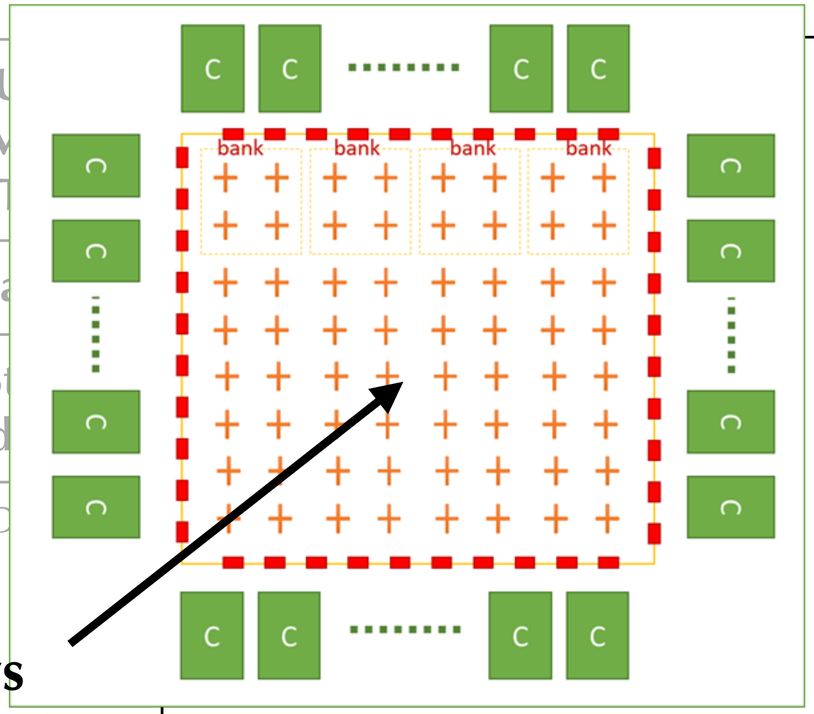
** adjusted by the memory technology's access latency

RIDICULOU
SIM
COMPUT

Bruce Ja

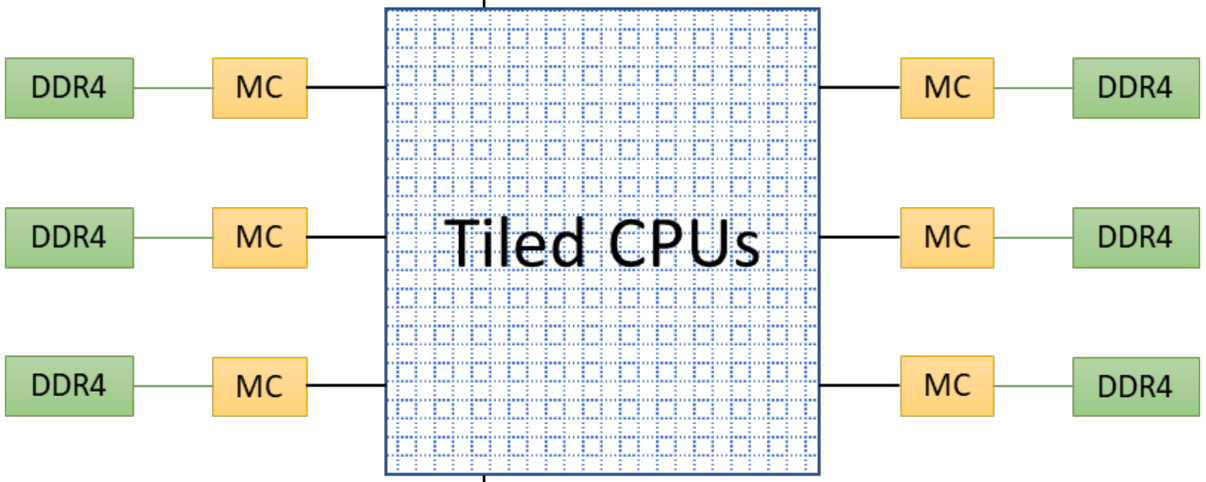
United St
Naval Acad

SLID



ReRAM Arrays
NoC-connected

ReRAM Architecture

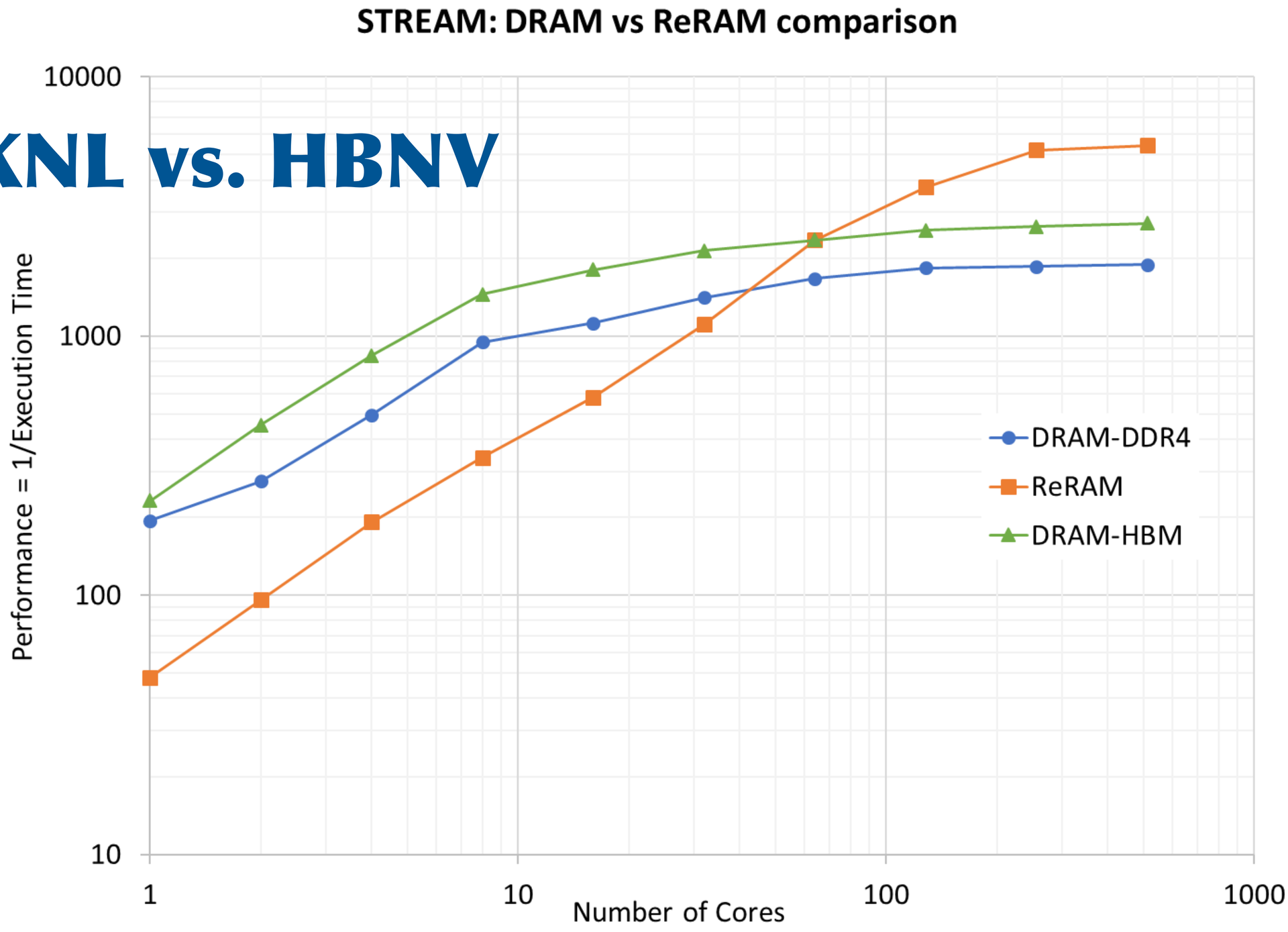


DRAM Architecture

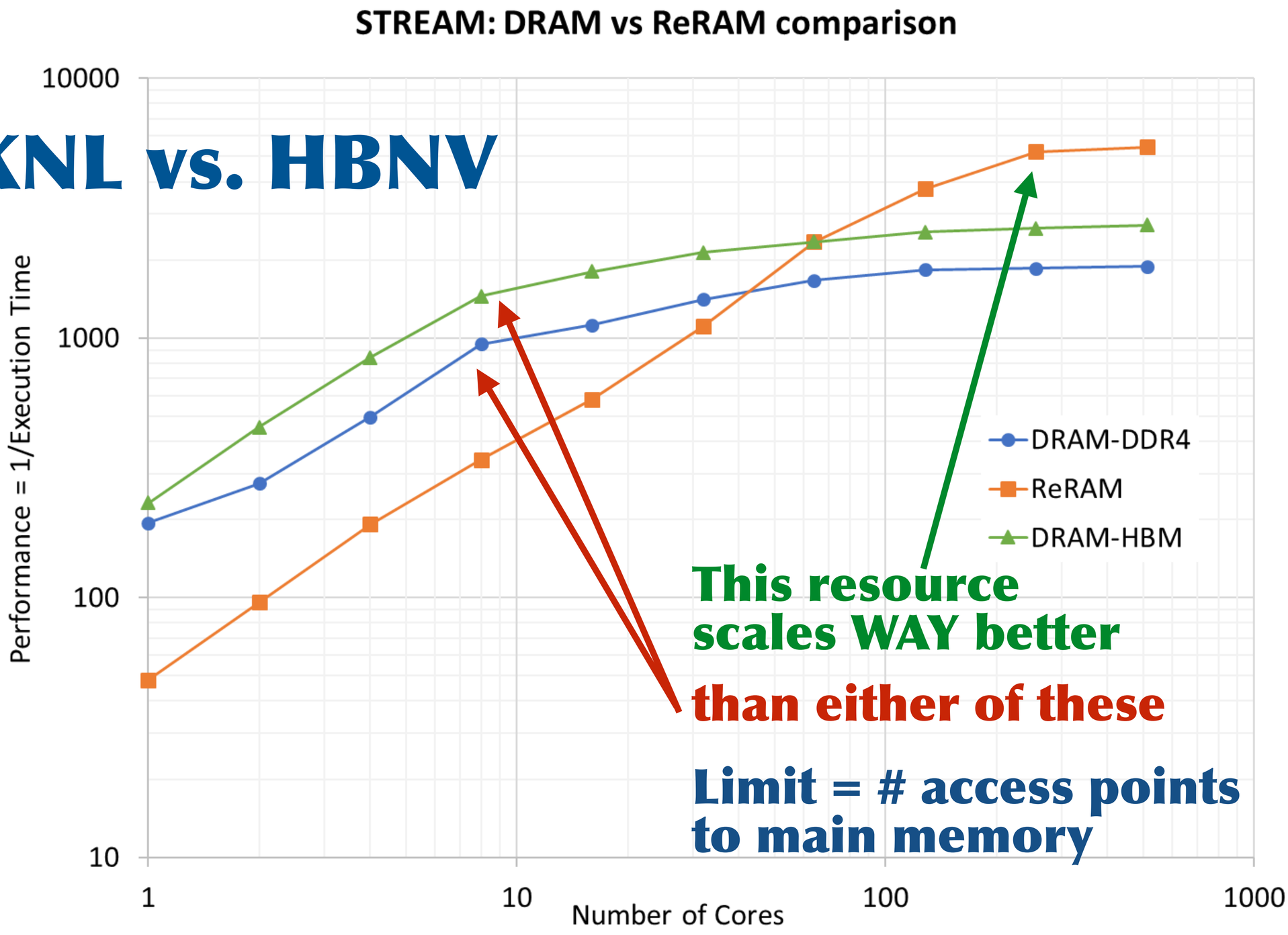
KNL vs. HBNV

	DRAM	ReRAM
# Mem Controllers	6	**1000**
L1 Cache size	32KB	32KB
L2 Cache size	1MB	1MB
Main Memory Parameters		
Mem Latency	DRAMsim3 simulated DDR4 & HBM	SST Messier **Read: 200ns** **Write: 1000ns**
request_width (access granularity)	**64 Bytes** bus-width = 8B burst-length = 8	**8 Bytes** bus-width = 8B burst-length = 1
Topology	Mesh	Mesh

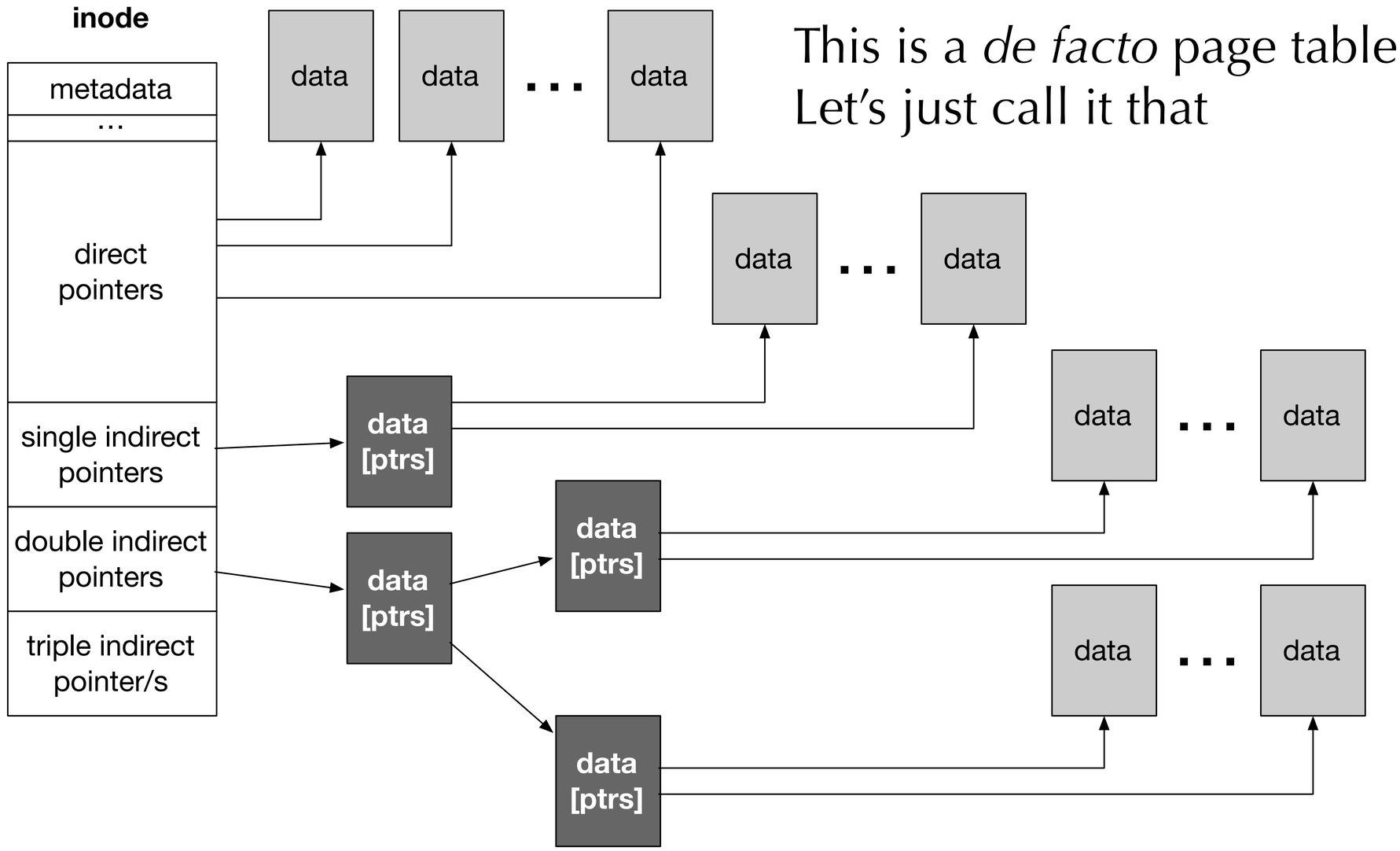
KNL vs. HBNV


STREAM: DRAM vs ReRAM comparison
Performance = 1/Execution Time
10000
1000
100
10
1
10
100
1000
Number of Cores
DRAM-DDR4
ReRAM
DRAM-HBM

KNL vs. HBNV


STREAM: DRAM vs ReRAM comparison
Performance = 1/Execution Time
10000
1000
100
10
1
10
100
1000
Number of Cores
DRAM-DDR4
ReRAM
DRAM-HBM
This resource scales WAY better
than either of these
Limit = # access points to main memory

Use load/store for Everything

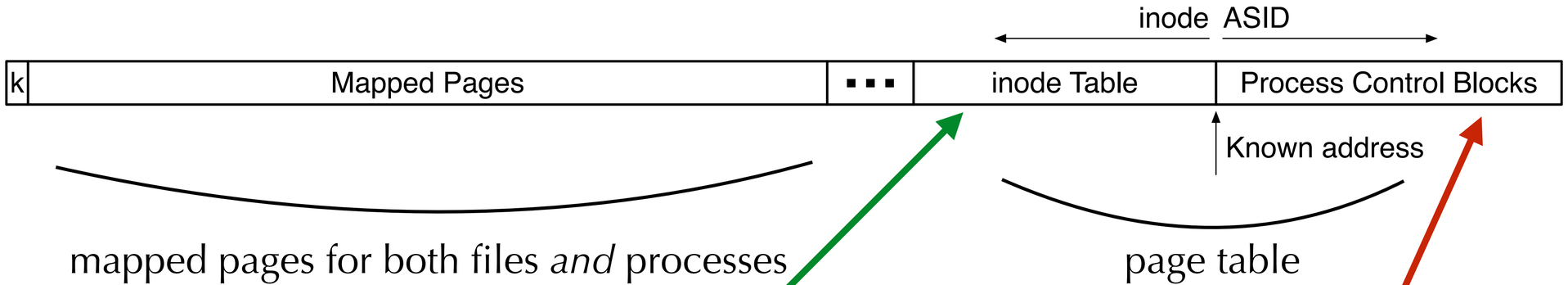


This is a *de facto* page table
Let's just call it that

Use load/store for Everything

Existing non-volatile space:

Proposal: it becomes something like this:



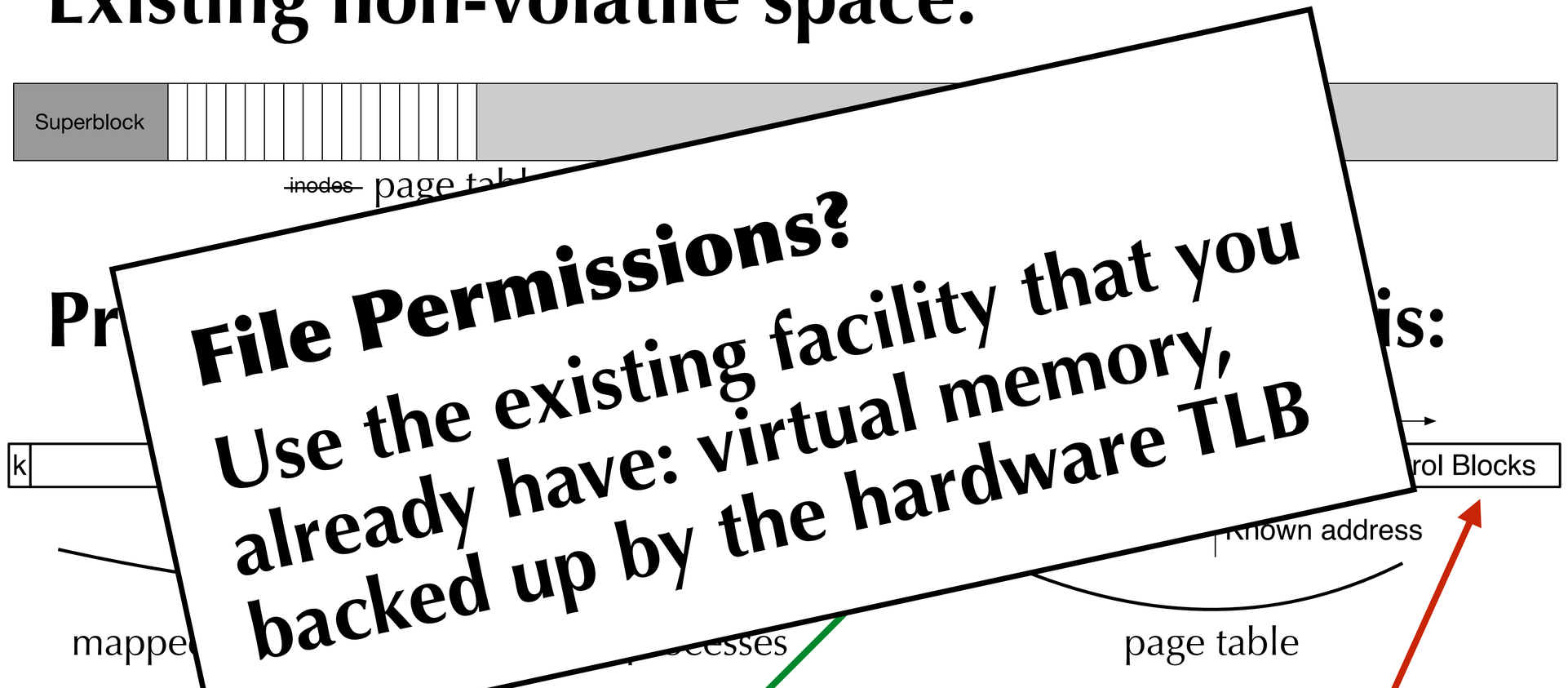
Map both file space and process space

Bruce Jacob

United States
Naval Academy

Use load/store for Everything

Existing non-volatile space:



File Permissions?
Use the existing facility that you already have: virtual memory, backed up by the hardware TLB

Map both file space and process space

Observations: Simplicity

Pseudocode for process invocation:

```
// input: inode of the file to execute

new_process(inode){
   choose ASID
   initialize a PCB (e.g., PC=0, SP=0xFFFF)
   put ASID onto runQ
   return from interrupt
}
```

You jump directly into the raw executable

Bruce Jacob

United States
Naval Academy

Observations / Conclusions

Verilator + Verilog CPU + kernel + shell
is doable in real time

(most of you guys already knew this)

[demo]

Bruce Jacob

United States
Naval Academy

Observations / Conclusions

Simplicity — Kernel in ~1000 lines of C

VM + preemptive multitasking + named files + int-driven I/O

Merged file system + virtual memory system (way less code)

Jumping into binaries is kinda cool :D

—> New Way to Use Modeling/Simulation

Running in first-person as opposed to batch-mode enables exploratory design, vs. design-space exploration

Study ANY microarchitecture, in ANY amount of detail, running ANY operating system … in real-time on your laptop

Shameless Plug

The Design and Security of Modern Computing Systems

How a Computer Is Built, From Ground Up
How It Is Designed To Provide Security
How its Defenses Can Be Overrun

Bruce **Jacob**
William **Casey**
Anthony **Melaragno**

United States Naval Academy
Cyber Science Department

RIDICULOUSLY SIMPLE COMPUTERS

Bruce Jacob

United States Naval Academy

SLIDE 17

Another Shameless Plug

www.memsys.io

Washington DC Sep/Oct, 2024

Call For Papers www.memsys.io Call For Papers

MEMSYS 2024

The International Symposium on Memory Systems ❖ Fall 2024, Washington DC

Important Dates

Abstract Deadline: 2 June, 2024
Submission: 16 June, 2024
Notification: 15 July, 2024
Camera-Readv: 1 September, 2024

ats

gth of an
ything from 1–
for example:

ers
pers

t, ACM 'sigconf' proceedings template, blind submission (no authors listed), up to 16 pages long

Organizers

Bruce Jacob (Naval Academy)
Kenneth Wright (AMD)
Chen Ding (U. Rochester)
Hameed Badawy (NMSU)
uce Childers (U. of Pittsburgh)
etmar Fey (U. Erlangen)
chael Jantz (University of Tennessee)
n Dreslinski (University of Michigan)
arc Reichenbach (U. Rostock)
Zhang (ICT)
nathan Beard (Google)
shan Chishti (Intel)
uce Christenson (Intel)
vid Donofrio (TTCL)
omas Vogelsang (Rambus)
hn Leidel (Tactical Computing Labs)
un F Rodrigues (Samsung)
bert Trout (Sadram)
lliam Wang (ARM)
tar Radojkovic (BSC)
rbert Wehn (RPTU)
atthias Jung (U. Würzburg)
ndy Elsasser (Rambus)
aya Gokhale (LLNL)
Hammond (NNSA)
zhak Birk (Technion)
ke Ignatowski (AMD)

Memory-device manufacturing, memory-architecture design, and the use of memory technologies by application software all profoundly impact today's and tomorrow's computing systems, in terms of their performance, function, reliability, predictability, power dissipation, and cost. Existing memory technologies are seen as limiting in terms of power, capacity, and bandwidth. Emerging memory technologies offer the potential to overcome both technology- and design-related limitations to answer the requirements of many different applications. Our goal is to bring together researchers, practitioners, and others interested in this exciting and rapidly evolving field, to update each other on the latest state of the art, to exchange ideas, and to discuss future challenges. *Please visit **memsys.io** for more information.*

Areas of Interest

Previously unpublished papers containing significant novel ideas and technical results are solicited. Papers that focus on system, software, and architecture level concepts specifically memory-related, i.e. topics outside of traditional conference scopes, will be preferred over others (e.g., the desired focus is away from pipeline design, processor cache design, prefetching, data prediction, etc.). Symposium topics include, but are not limited to, the following:

- Memory-system design from both hardware and software perspectives
- Memory failure modes and mitigation strategies
- Memory and system safety and security issues
- Disaggregated Memory (e.g. CXL)
- Memory for embedded and autonomous systems (e.g., automotive)
- Operating system design for hybrid/nonvolatile memories
- Technologies including, DRAM, FLASH, NVM etc.
- Memory-centric programming models, languages, optimization
- Compute-in-memory and compute-near-memory technologies
- Data-movement issues and mitigation techniques
- Algorithmic & software memory-management techniques
- Emerging memory technologies, their controllers, and novel uses
- Interference at the memory level across datacenter applications
- In-memory databases and NoSQL stores
- Post-CMOS scaling efforts and memory technologies to support them, including cryogenic, neural, and heterogeneous memories
- Negative results, validation of results, invalidation of results

To reiterate, papers that focus on topics *outside* the scope of traditional architecture conferences will be preferred over others.

Submissions and Presentations

Our primary goal is to showcase interesting ideas that will spark conversation between disparate groups—to get applications people, operating systems people, system architecture people, interconnect people and circuits people to talk to each other. We accept extended abstracts, position papers, and/or full research papers, and each accepted submission is given a 20-minute presentation time slot.

We intend to publish all papers in the ACM Digital Library.

Venue

The conference will be in Washington DC area.

Thank You!

Questions?

Bruce Jacob

bjacob@usna.edu

But Wait, How Does argv Work?

Pseudocode for process invocation:

```
// input: inode of the file to execute

new_process(inode, phys_arg_page){
   choose ASID
   initialize a PCB (e.g., PC=0, SP=0xFFFF)
   put ASID onto runQ
   map( VIRTUAL_ARG_PAGE, phys_arg_page );
   return from interrupt
}
```

The *shell* reads input line into ARG_PAGE**, calls** detach(ARG_PAGE) **& then** new_proc()