

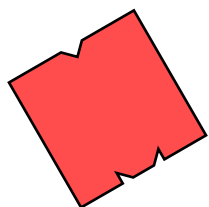
Architectures for Real-Time Caching

Bruce Jacob

**Electrical & Computer Engineering
University of Maryland, College Park**

OUTLINE:

- **Caches vs. Tagless SRAMs**
- **Data Placement and Its Complexity**
- **Solutions to the Problem**



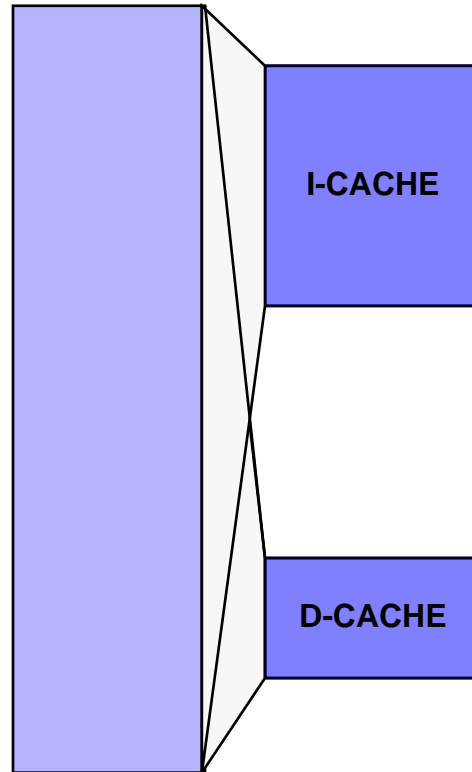
Why Traditional Caches Suck



NON-DETERMINISM

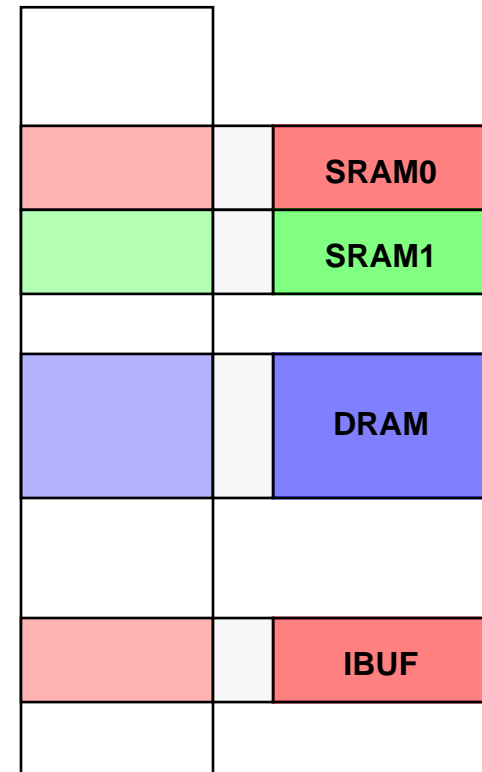
Cache vs. Tagless SRAM

UNIFORM
ADDRESS
SPACE



Traditional Caches

NON-UNIFORM
ADDRESS
SPACE



Tagless SRAMs

**MAIN DIFFERENCE: SRAM requires
EXPLICIT MANAGEMENT**

Tagless SRAMs

Addressing is impediment:

CONTIGUITY must be preserved

DISTANCE BETWEEN OBJECTS
must be preserved

Multiply-accumulate requires
TWO DISJOINT DATA SPACES

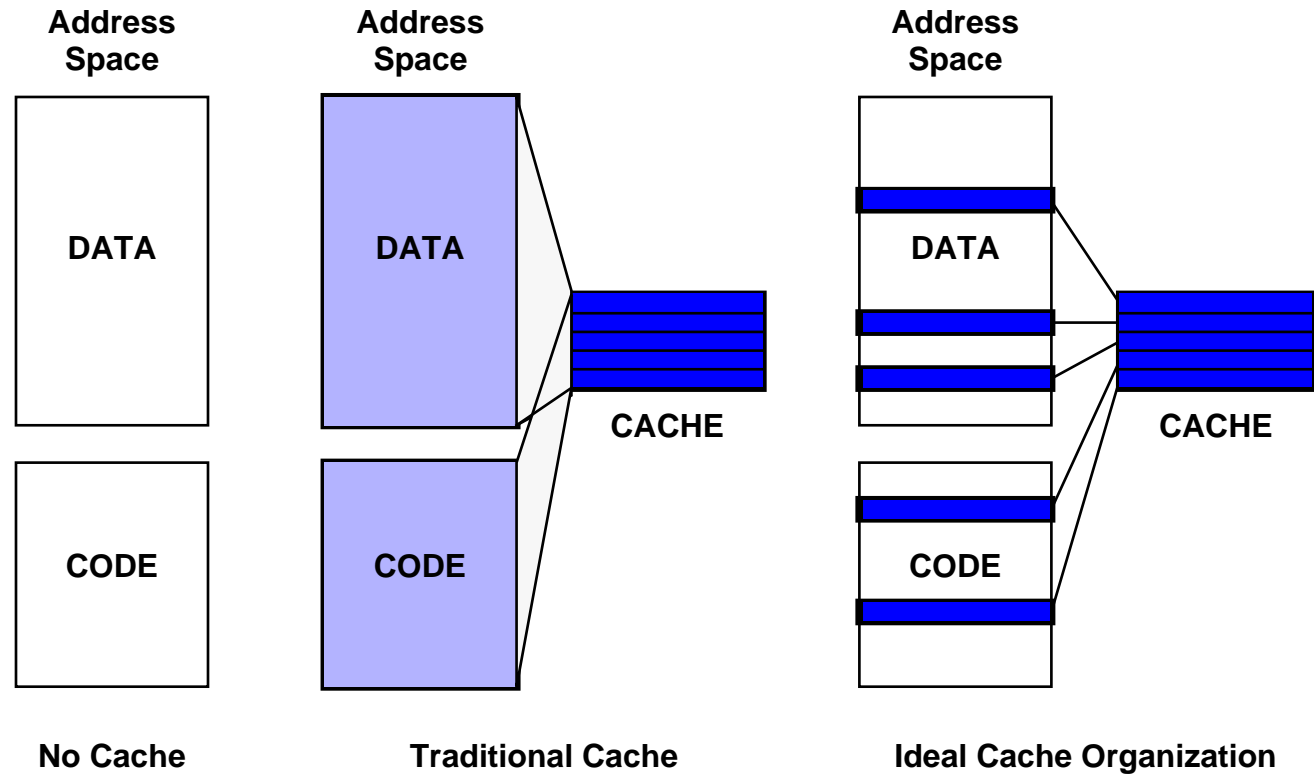
Bottom Line:

Access to memory **NON-ORTHOGONAL**
Separate spaces are **DISJOINT**

Result: **COMPILATION IS HARD**

Trend: **UNIFORM ADDRESS SPACES**




WHAT WE WANT



No Cache

Traditional Cache

Ideal Cache Organization

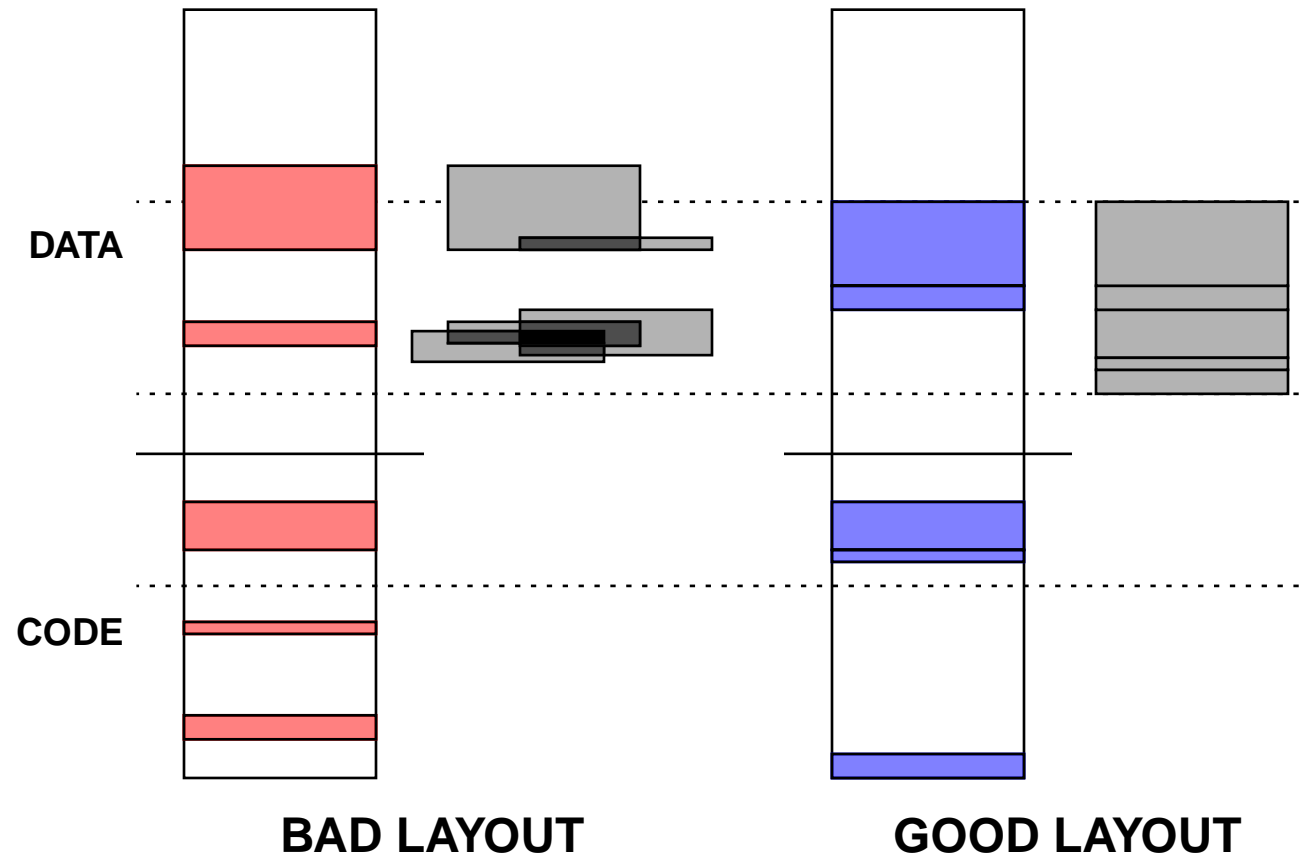
-  Guaranteed slow access-time
-  Statistically fast access-time
-  Guaranteed fast access-time

WHY IT'S DIFFICULT

DATA NAME => DATA PLACEMENT

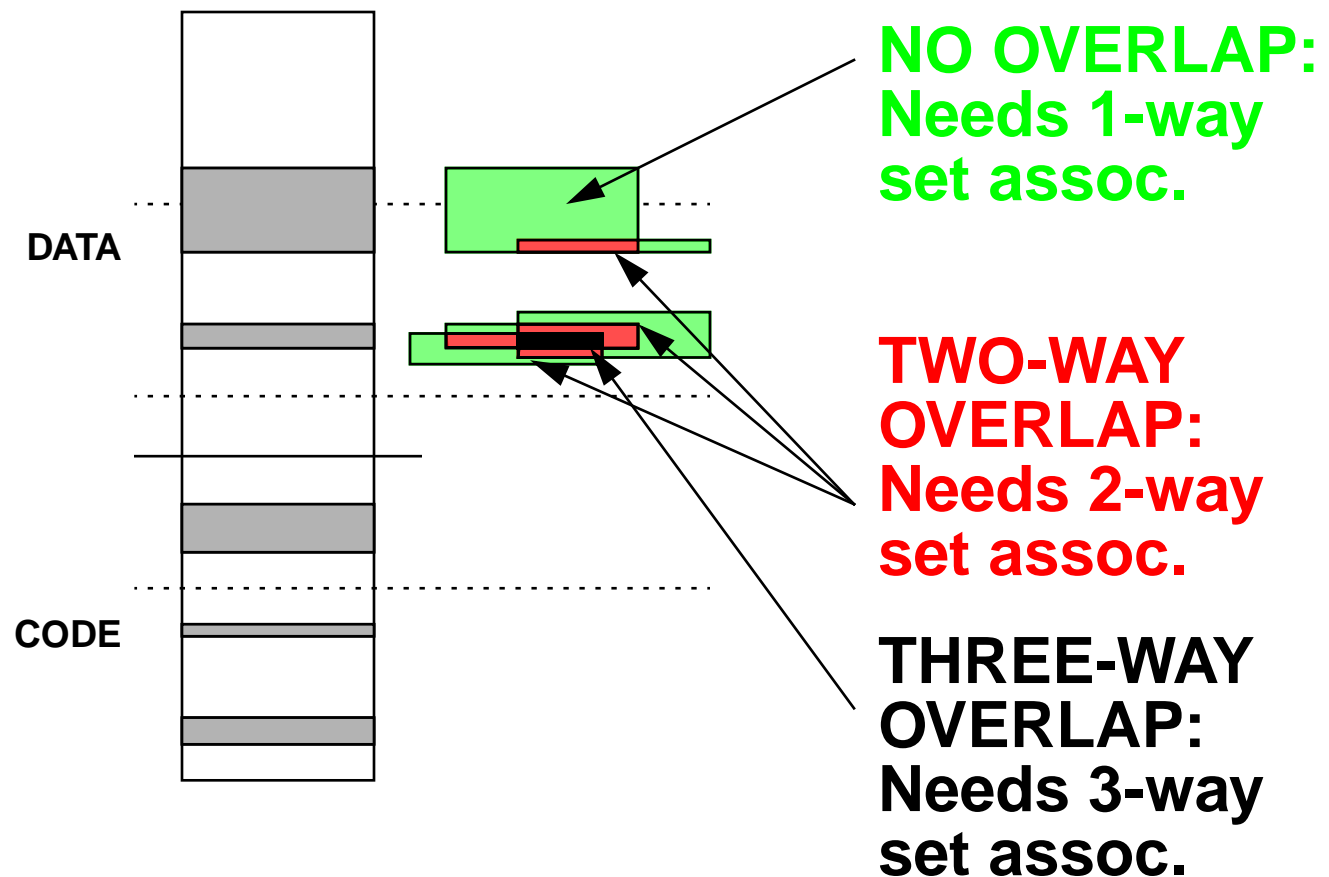
Must Group Data & Instructions

So as to Minimize Cache Conflicts



Solution #1

A BIG, HIGHLY ASSOCIATIVE CACHE
+ ability to PIN DOWN CACHE LINES



Solution #1

- **Choose items to cache,
Bring each into the cache,
Pin each down**
- **Can CACHE/NOT-CACHE adjacent items**
- **Must know CACHE ORGANIZATION
at COMPILE TIME
(not huge issue for embedded systems)**
- **SIMPLEST, but perhaps
MOST EXPENSIVE solution
(big & highly associative ...)**

Main Issue: Data Placement

PROBLEM:

To reduce size/associativity of cache,
must **LOCATE** data in **MEMORY SPACE**
so as to **ALIGN IN CACHE** (conflict-free)

This task is **NP-complete**.

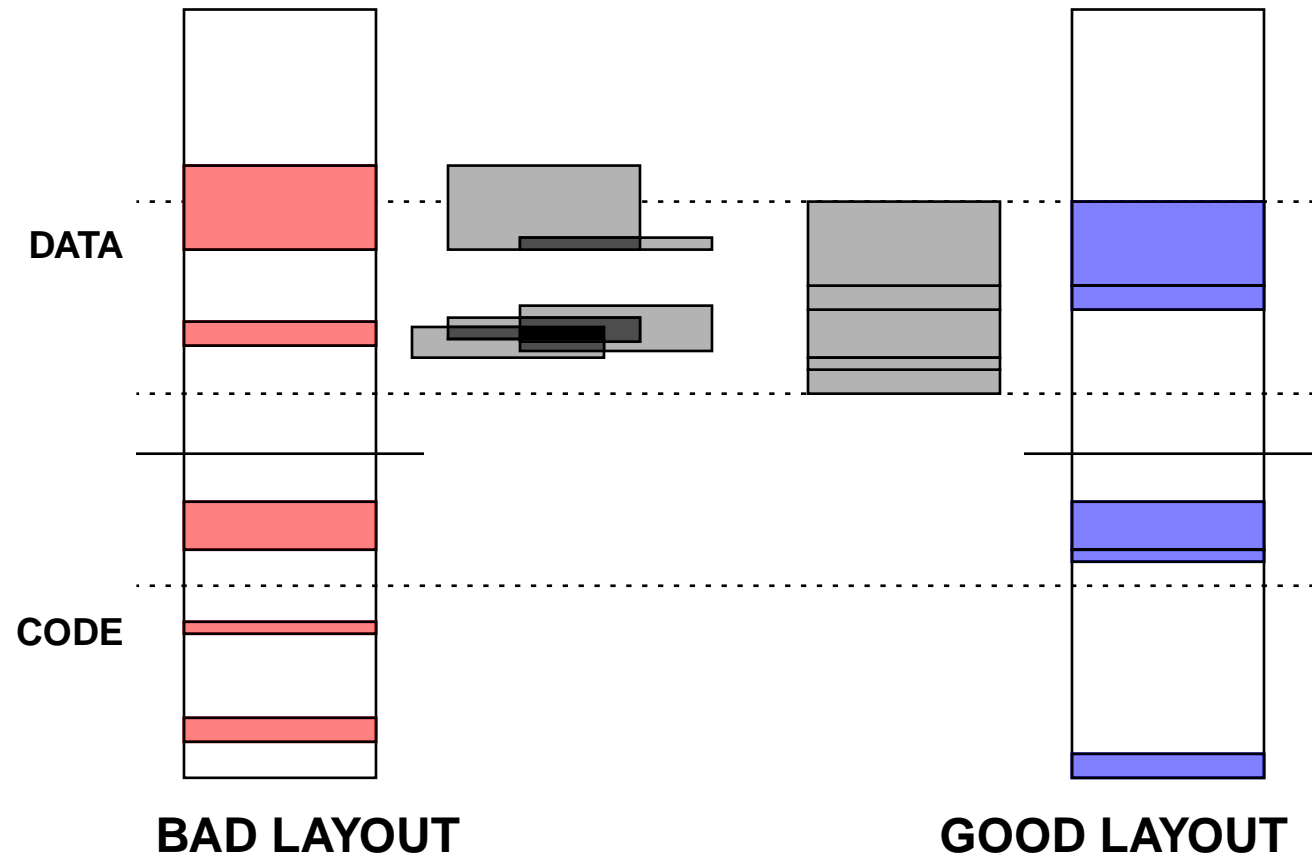
GOALS:

- Disassociate **NAME** and **PLACEMENT**
- Fine-grained code/data relocation
at granularity of **TLB page**
or (better) **cache line**

Enter Virtual Addressing

Disassociates NAME from PLACE

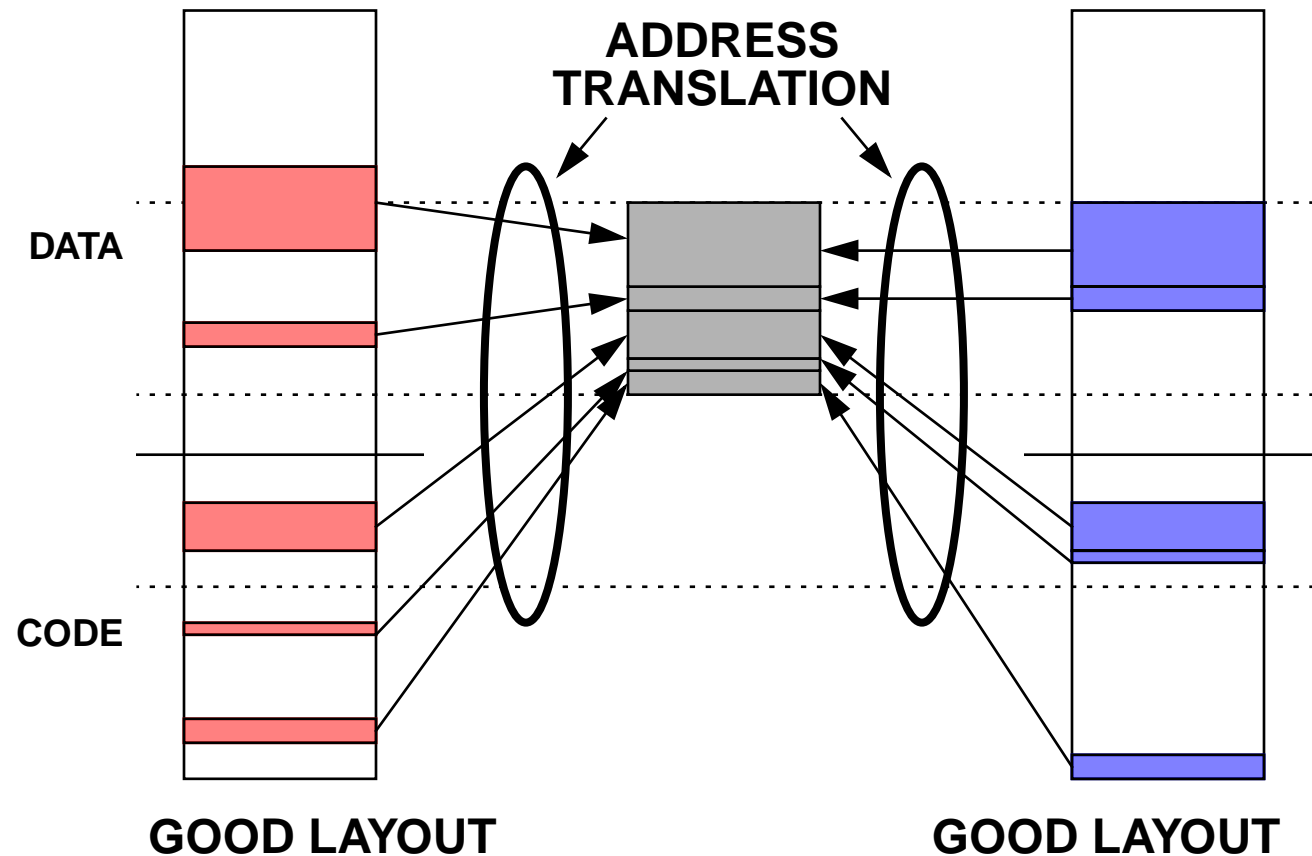
Allows you to go from **THIS**:



Enter Virtual Addressing

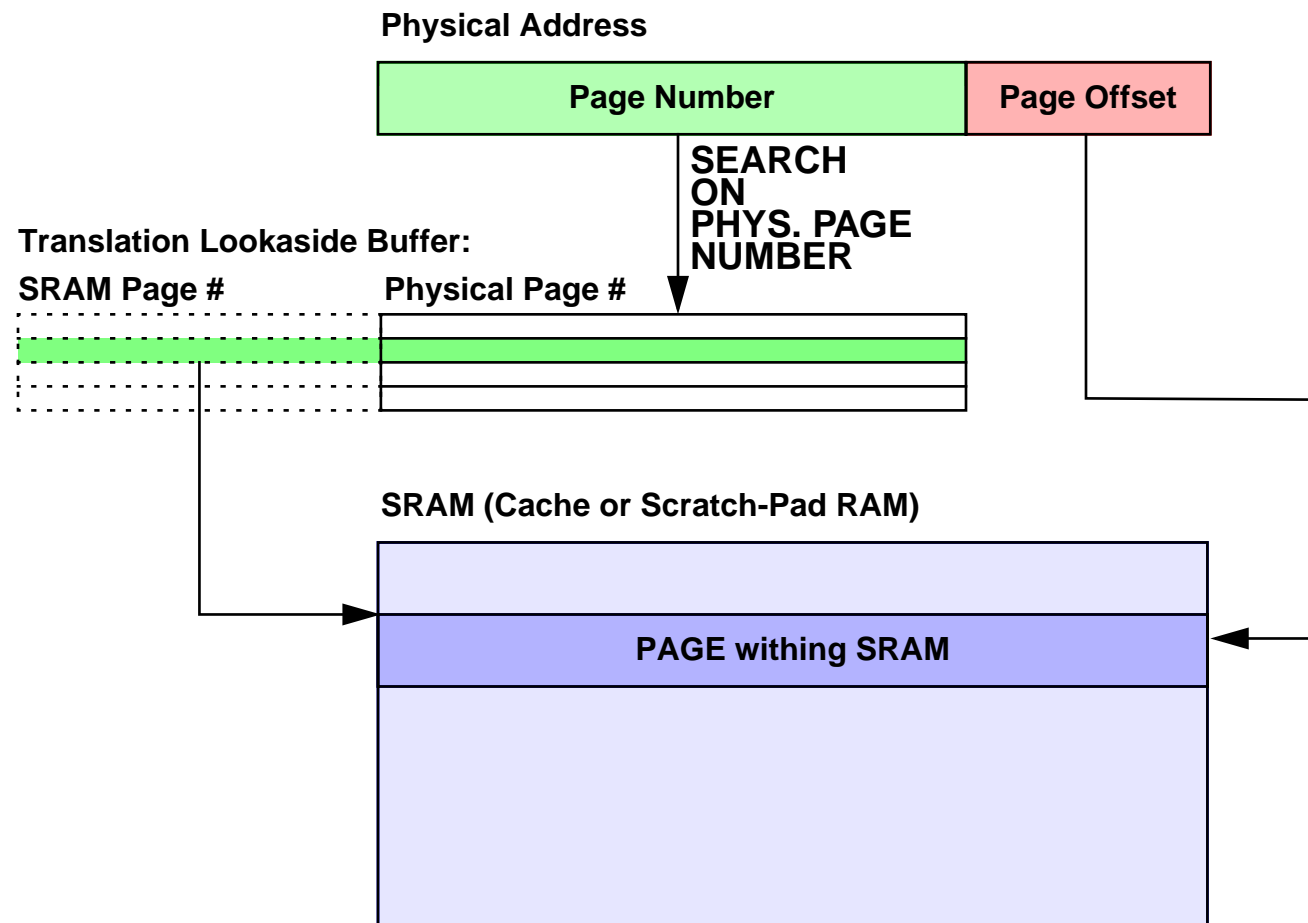
Disassociates NAME from PLACE

... to **THIS**:



Real-Time TLB Organization

Works with either **CACHE** or **SCRATCH-PAD**



Solution #2

Fully-Associative Real-Time TLB + SW-Managed Tagless SRAM

- TLB must fully map SRAM
(8KB SRAM, 256-byte page => 32 entries)
- Can place ANY 256-byte page
ANYWHERE in the SRAM
- Benefit: **simple SRAM design**
- Drawback: **fully assoc. TLB**

**Similar to software-managed FA cache
with very large lines**

Variations on Solution #2

WANT A **LARGER CACHE?**

- Larger TLB
- Larger Page Size

WANT A **SMALLER TLB?**

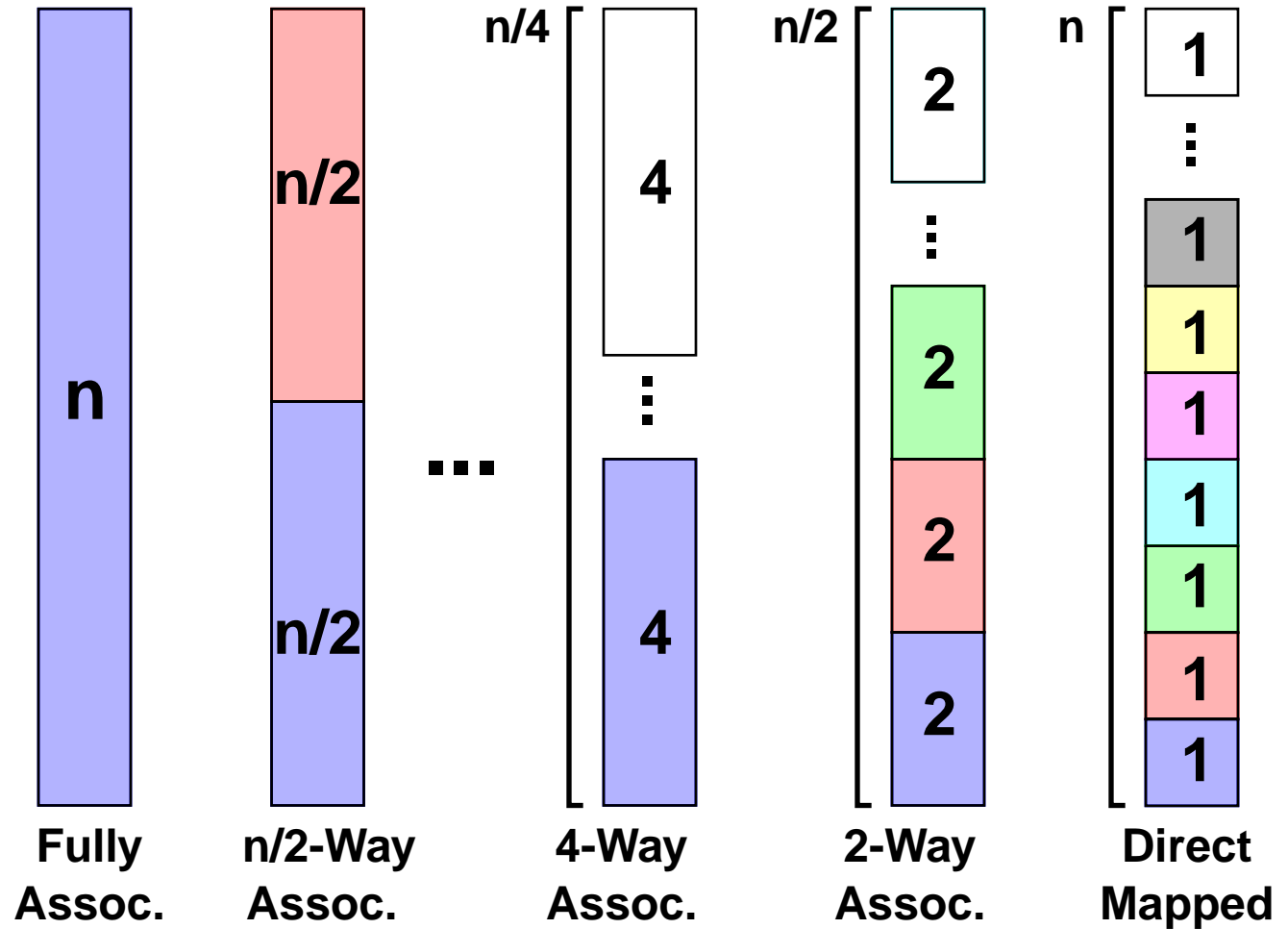
- Smaller Cache
- Larger Page Size

WANT **LESS ASSOCIATIVITY?**

- That's a little more involved ...

Set-Associative RT-TLBs

Associativity vs. the Memory Space



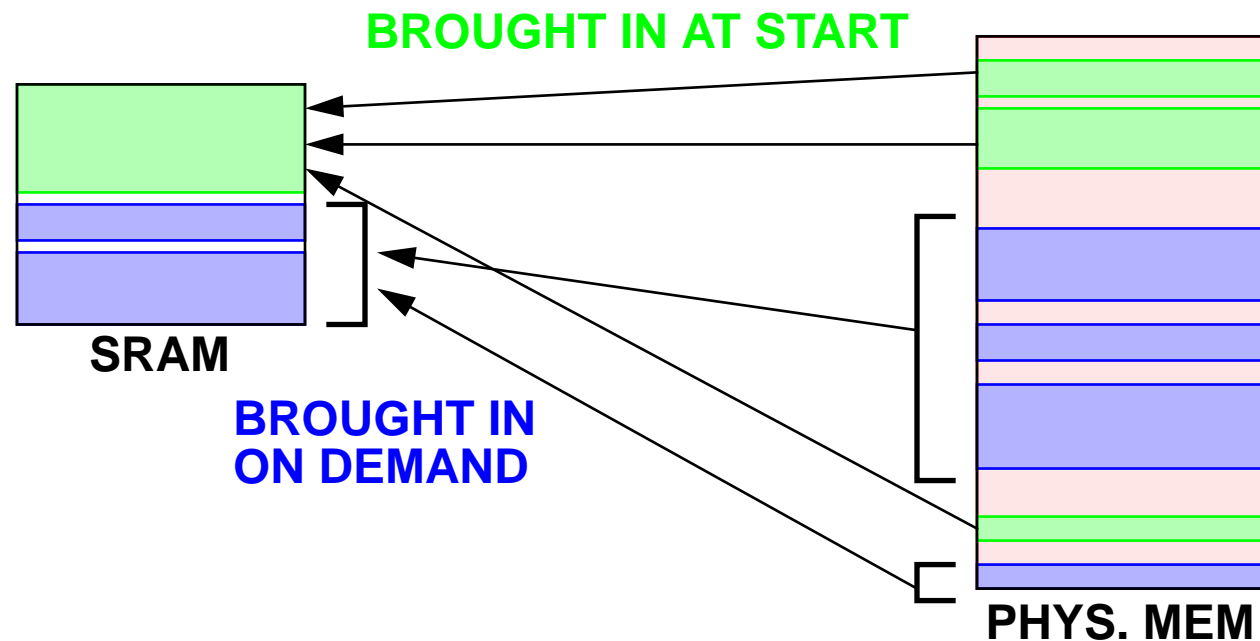
$n =$ entries in TLB

Solution #3

What if SRAM Still Too Small?

(i.e. — previous solution reduces **CONFLICT** problems, not **CAPACITY** problems)

Real-Time SRAM-Management



Real-Time SRAM Management

CLASSIFY ALL CODE & DATA:

- **MUST ALWAYS REMAIN CACHED**
- **MUST NEVER BE CACHED**
- **EXHIBITS PERIODIC LOCALITY**
(i.e. loop code & data)

FOR PERIODIC ITEMS:

- **Add code at beginning to set up TLB**
- **Add code at end to unmap TLB
and write out any dirty values**

Real-Time SRAM Management

RESULTS:

- **VM-style extending of SRAM space into DRAM space via demand-paging**
- **PROACTIVE demand-paging, not REACTIVE demand-paging**
- **Deterministic memory performance for all references**
- **Slight overhead in code size & execution**

Summary

UNIFORM MEMORY SPACES:

- **Provide Orthogonal Look at Memory**
- **Cache Architectures Exhibit Non-Deterministic Performance**

NON-UNIFORM MEMORY SPACES:

- **Non-Orthogonal Memory Map**
- **Caches Offer Deterministic Performance (at the Price of Explicit Management)**

TREND IS TOWARD UNIFORM SPACES

- **Easier to Program & Compile for ...**

Summary, cont'd

REAL-TIME CACHE ARCHITECTURES:

- Really Big, Highly Associative Caches
- Virtual Addressing w/ RT-TLB
- Real-Time SRAM Management

URL (CASES talk slides, white paper, etc.):

<http://www.ece.umd.edu/~blj/>

