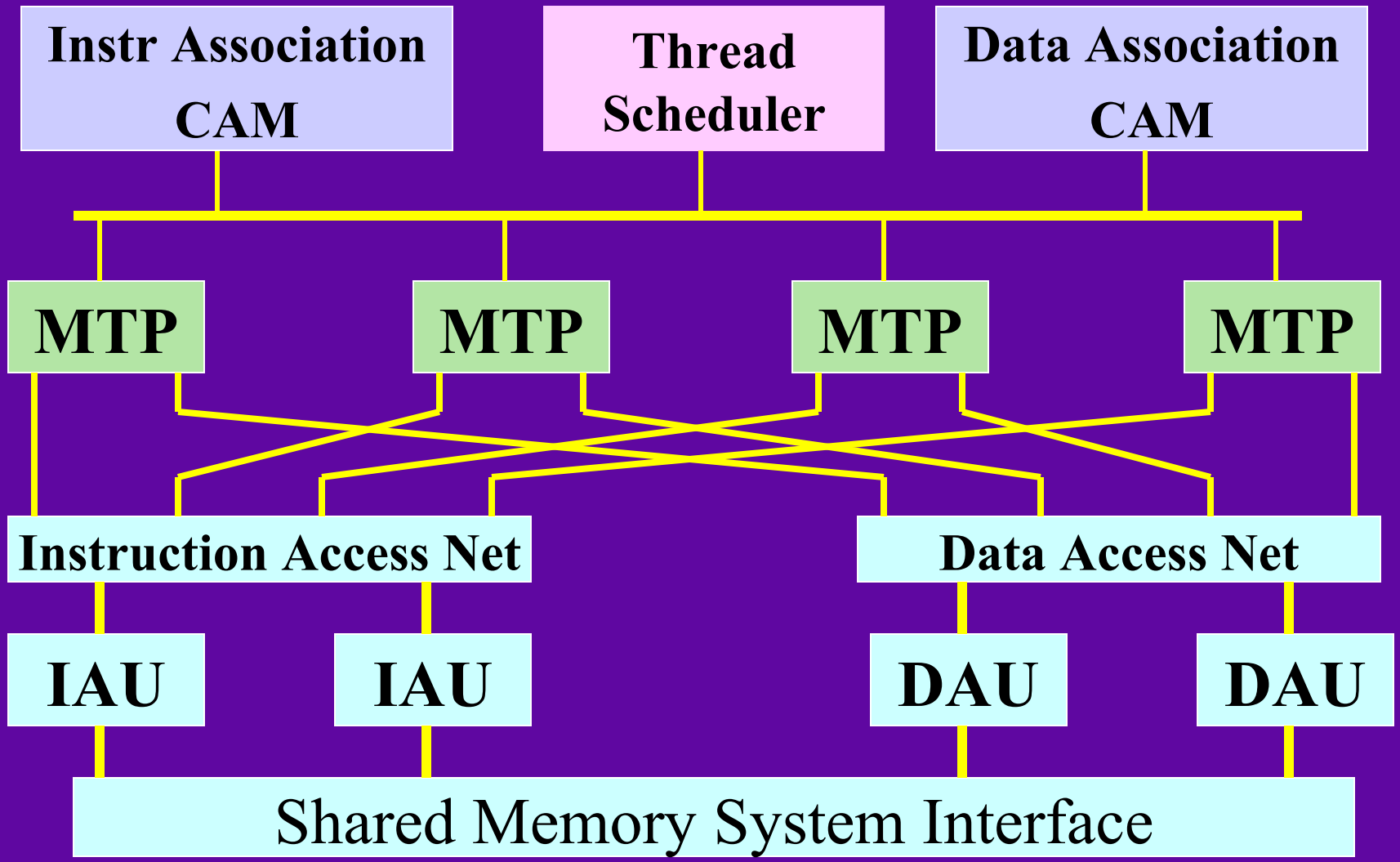
# Content Addressable Memory Using B-Tree

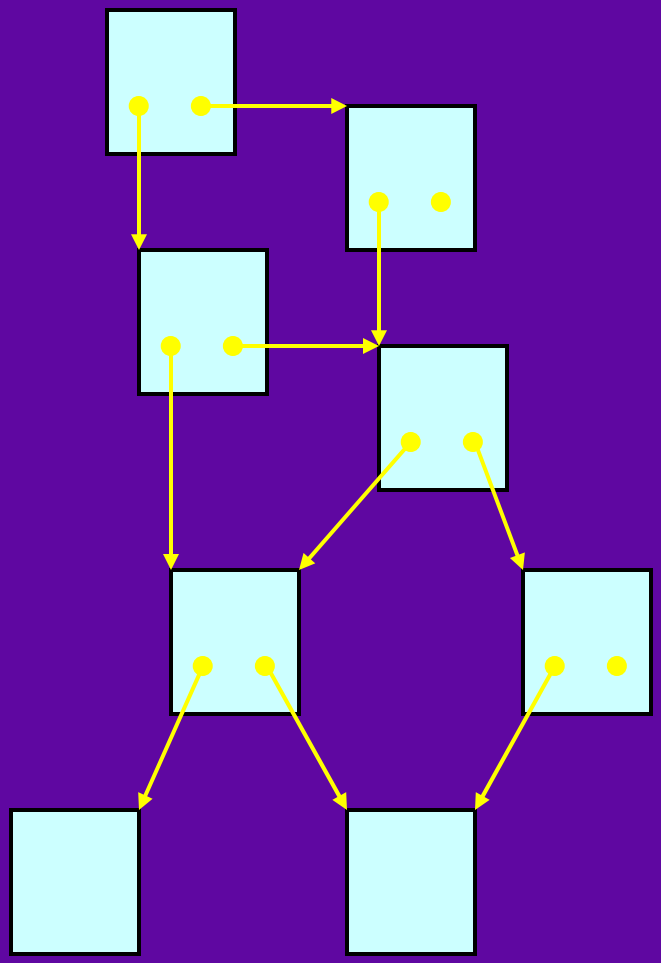
Jack Dennis
MIT Computer Science
and
Artificial Intelligence Laboratory

# Multi-Thread, Multi-Core Chip

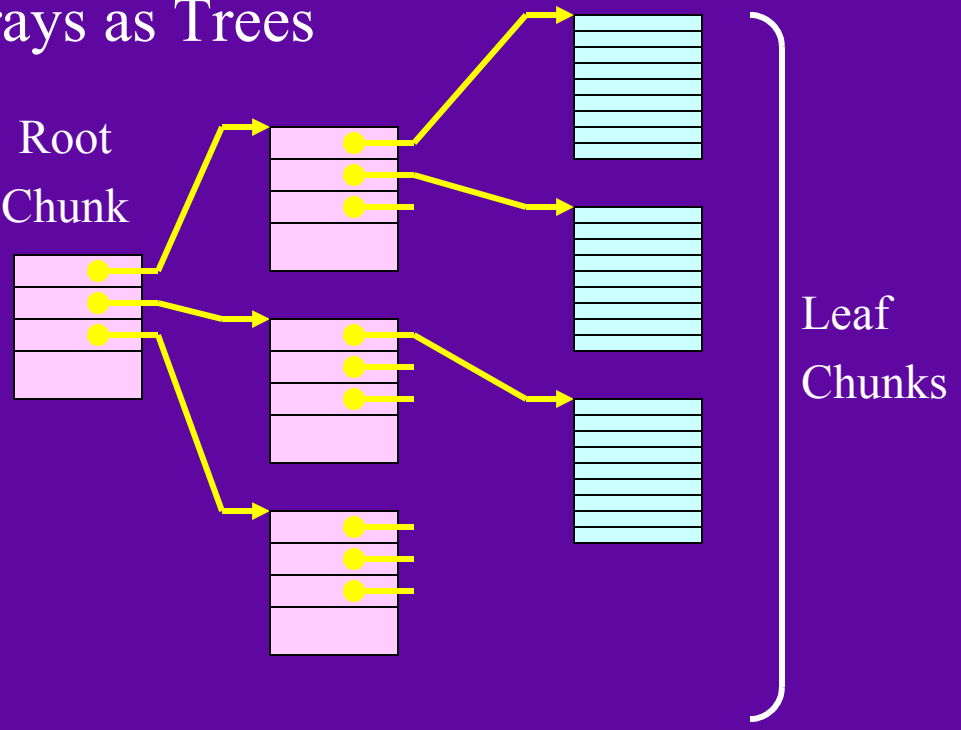


# Data Structures: The Heap as a DAG

- Cycle-Free Heap



- Arrays as Trees




- Fan-out as large as 16
- Three levels yields 4096 elements (longs)

# B-Trees

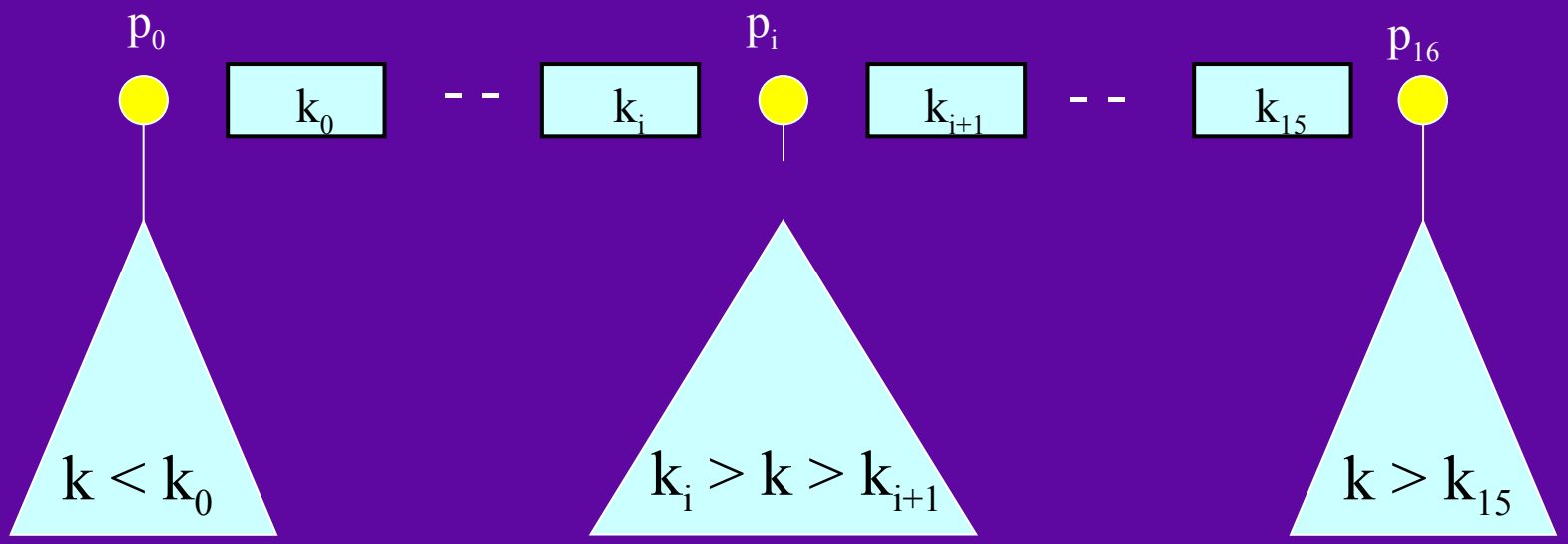
A Data Structure that represents a collection of keys

In an Order Z B-Tree, each node holds at least Z keys and associated pointers to subtrees.

An order Z B-Tree of depth h can hold at least

$$(Z^h * 2) - 1 \quad \text{keys}$$

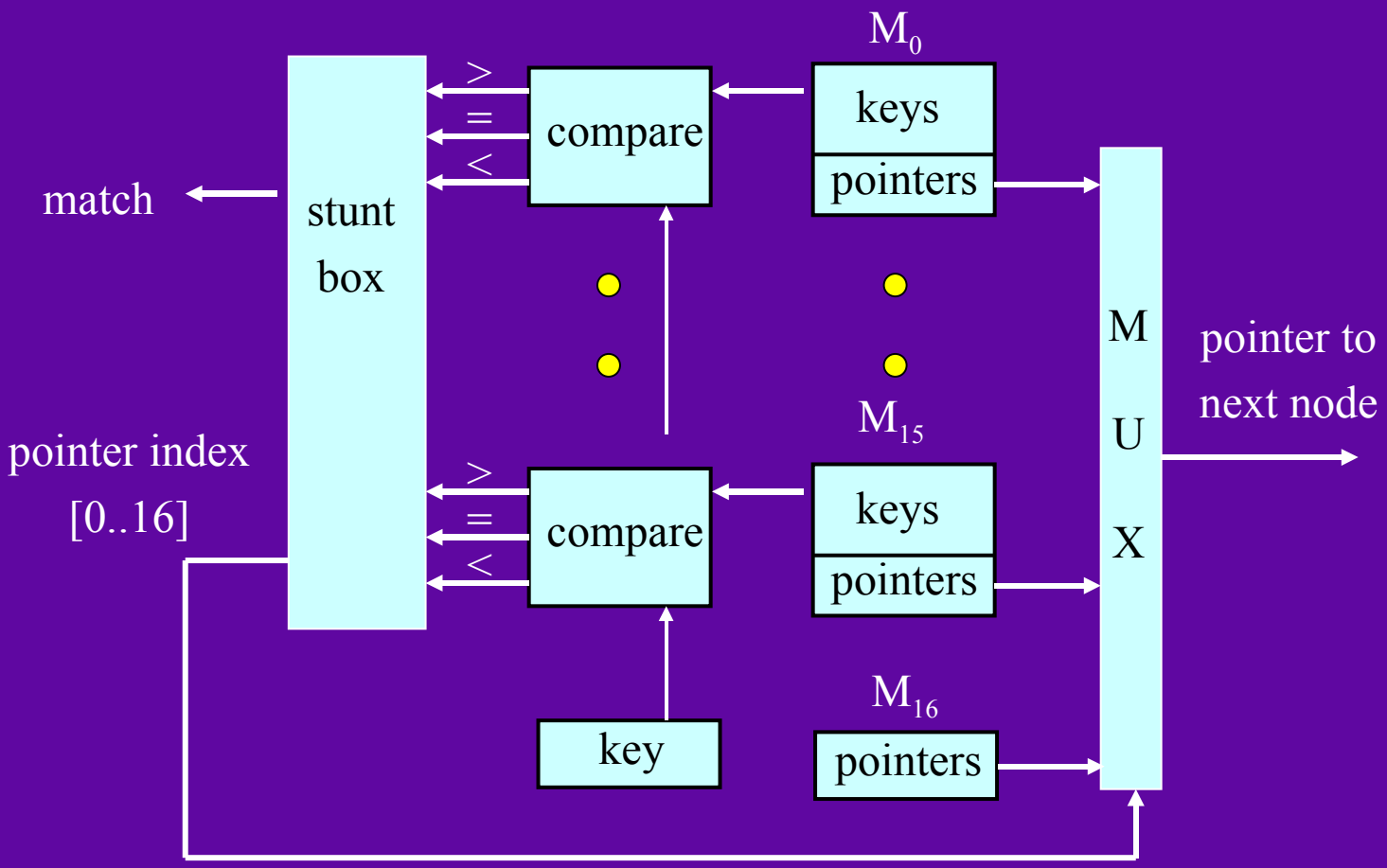
# A B-Tree Node




Invariants (Order Z B-Tree; Z = 8 above):

- Number of keys at a node: at least Z; at most Z*2
- Exception: The root node may contain as few as zero keys.
- All leaf nodes are at the same level.
- All subtree pointers in a leaf node are null (and only those)

# A Hardware B-Tree CAM



# Insert

1. Search the B-Tree for the new key; failure if found.
2. If the leaf node is full it must be split into two nodes, each holding Z keys. The median key of the set of keys held by the found leaf node and the key to be inserted is inserted into the parent node, continuing up the tree.
3. To do insert in a single pass, split any full node encountered during the search to anticipate the need.

# Delete

1. Search the B-Tree for the given key; failure if not found.
2. To do delete a key from a leaf node, it may be necessary to move in a key from a neighboring node; if this is not possible, the leaf node may be joined with one of its neighbors and the pointer to the lost node deleted from the parent
3. A few more complications arise.

# B-Tree Node Rearrangement