CSE 544: Lecture 12

Wednesday, 5/8/2002

Hash table indexes,

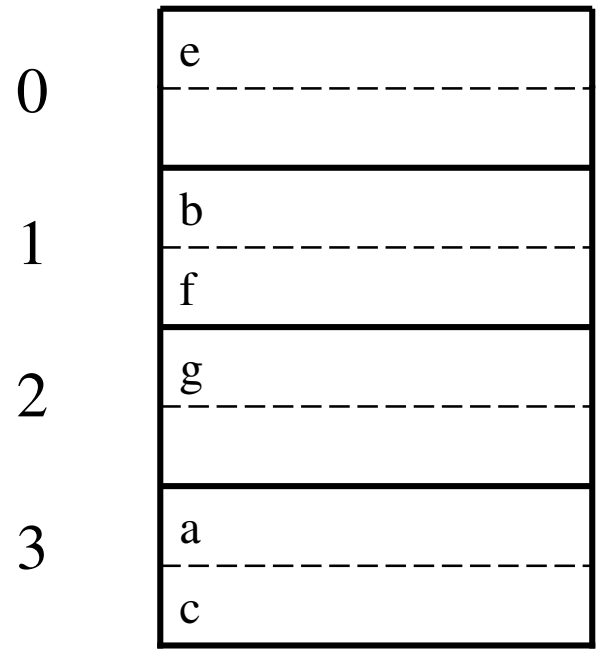
physical operators

Hash Tables

- Secondary storage hash tables are much like main memory ones
- Recall basics:
  - There are n *buckets*
  - A hash function f(k) maps a key k to {0, 1, …, n-1}
  - Store in bucket f(k) a pointer to record with key k
- Secondary storage: bucket = block, use overflow blocks when needed

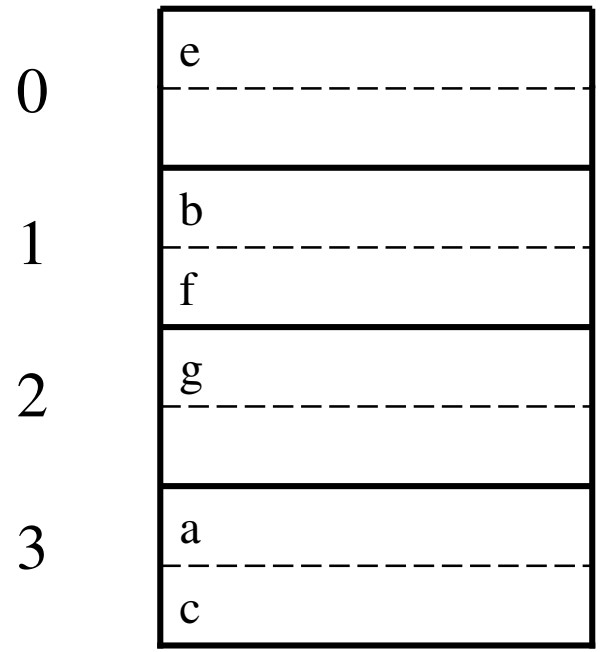
Hash Table Example

- Assume 1 bucket (block) stores 2 keys + pointers
- h(e)=0
- h(b)=h(f)=1
- h(g)=2
- h(a)=h(c)=3



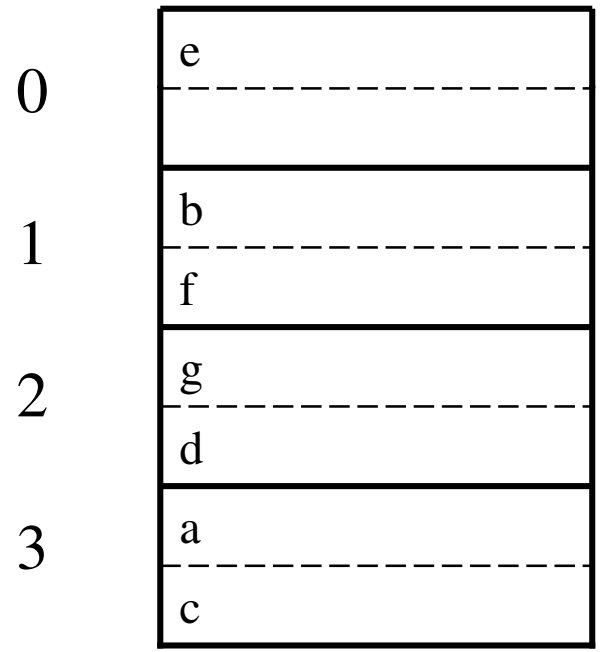
Searching in a Hash Table

- Search for a:
- Compute h(a)=3
- Read bucket 3
- 1 disk access



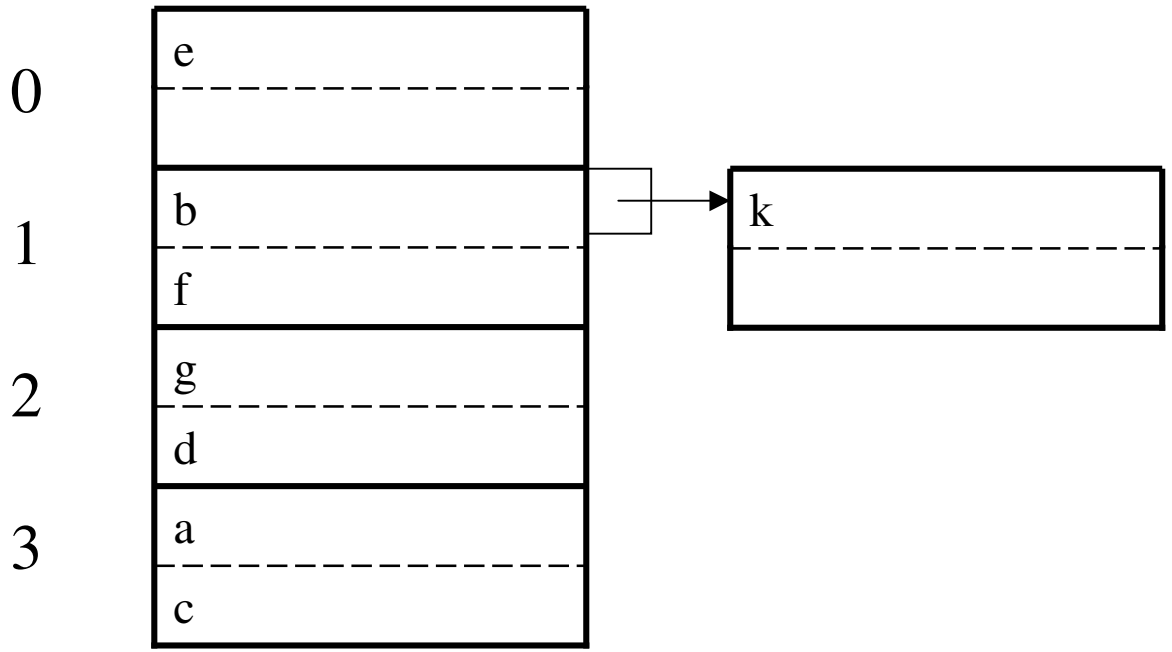
Insertion in Hash Table

- Place in right bucket, if space
- E.g. h(d)=2



Insertion in Hash Table

- Create overflow block, if no space
- E.g. h(k)=1



- More over-flow blocks may be needed

Hash Table Performance

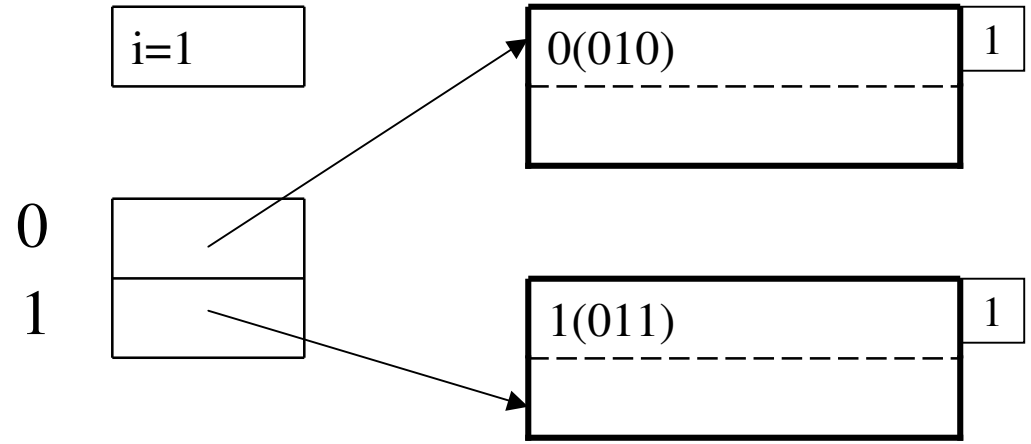
- Excellent, if no overflow blocks
- Degrades considerably when number of keys exceeds the number of buckets (I.e. many overflow blocks).

Extensible Hash Table

- Allows has table to grow, to avoid performance degradation
- Assume a hash function h that returns numbers in $\{0, \ldots, 2^k - 1\}$
- Start with $n = 2^i << 2^k$, only look at first i most significant bits

Extensible Hash Table

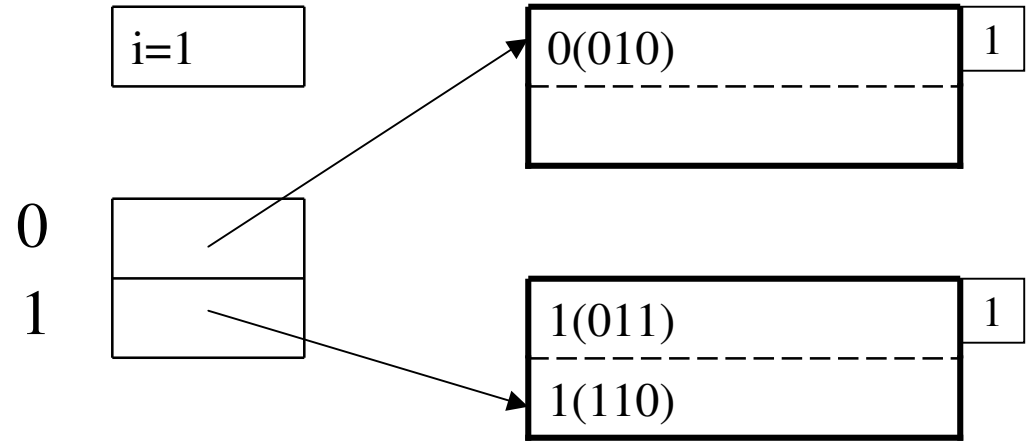
- E.g. i=1, n=2, k=4



- Note: we only look at the first bit (0 or 1)

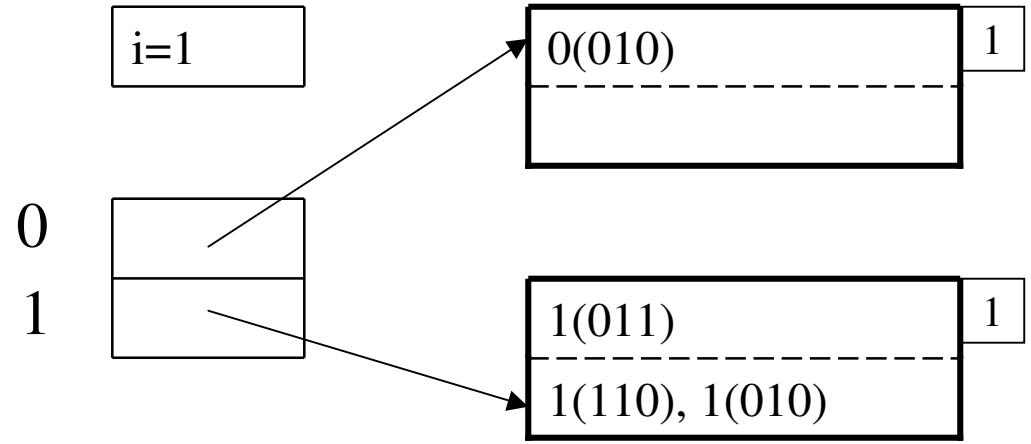
Insertion in Extensible Hash Table

- Insert 1110



Insertion in Extensible Hash Table

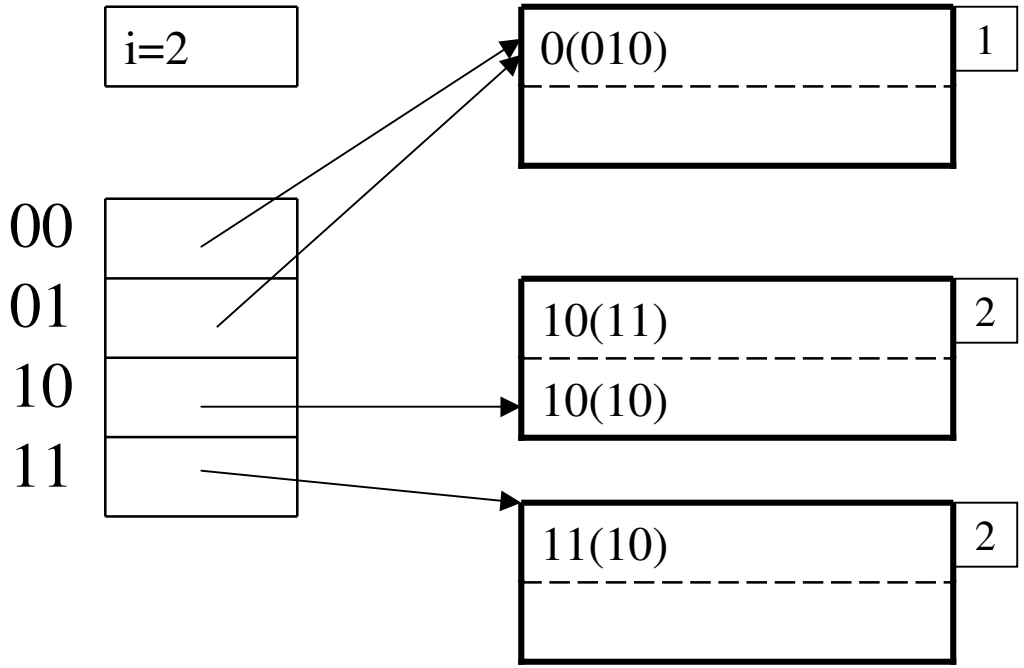
- Now insert 1010



- Need to extend table, split blocks
- i becomes 2

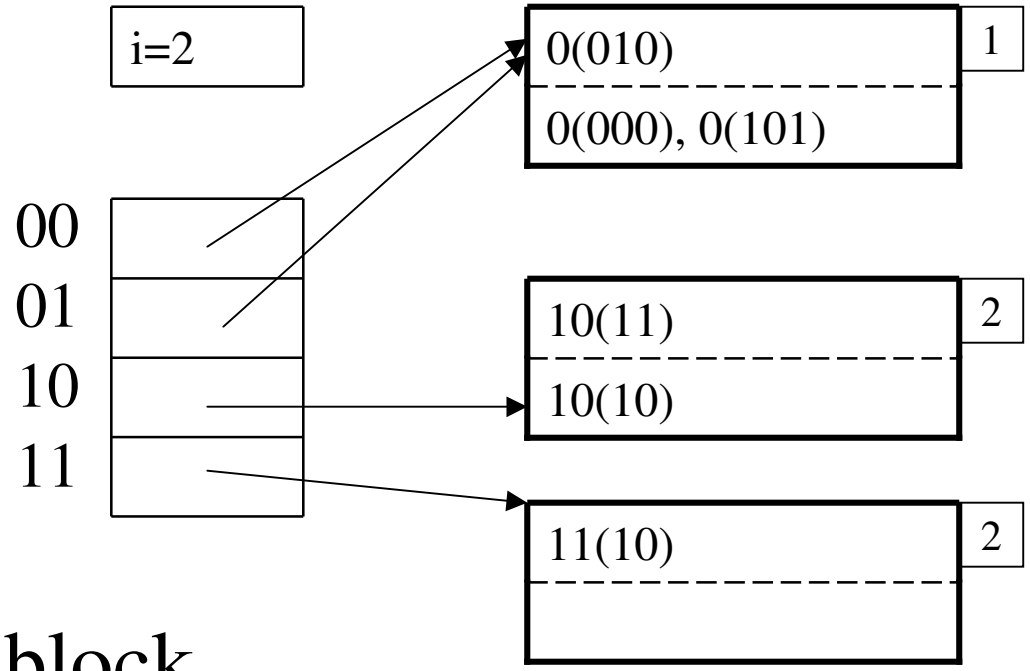
Insertion in Extensible Hash Table

- Now insert 1110



Insertion in Extensible Hash Table

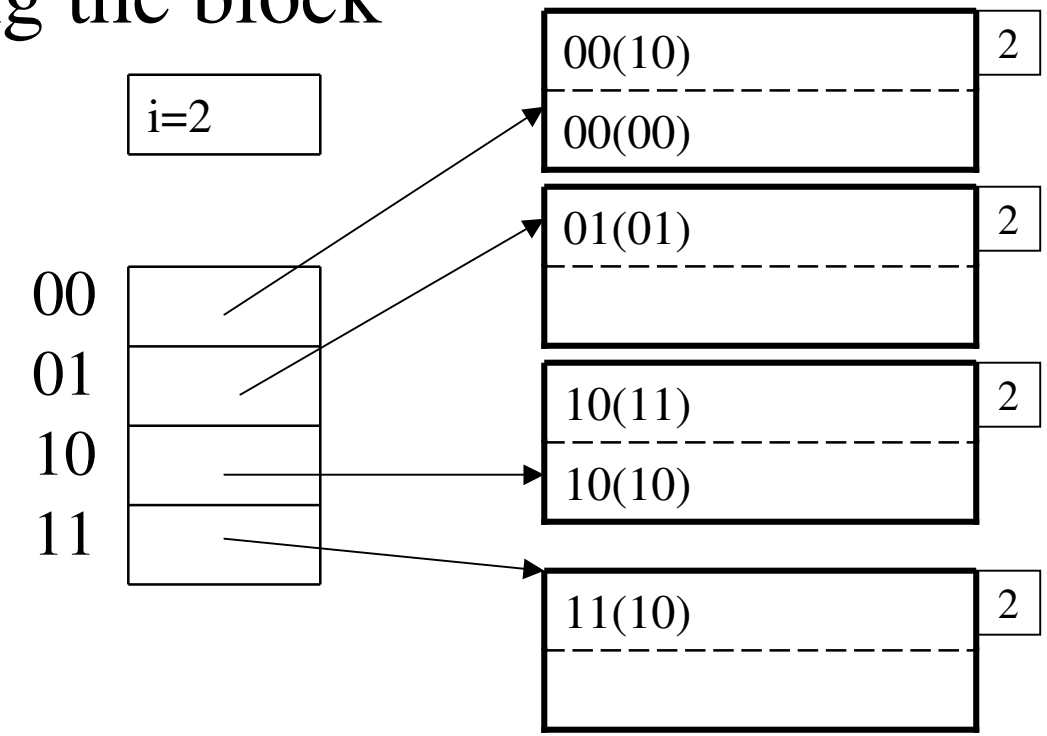
- Now insert 0000, then 0101



- Need to split block

Insertion in Extensible Hash Table

- After splitting the block



Performance Extensible Hash Table

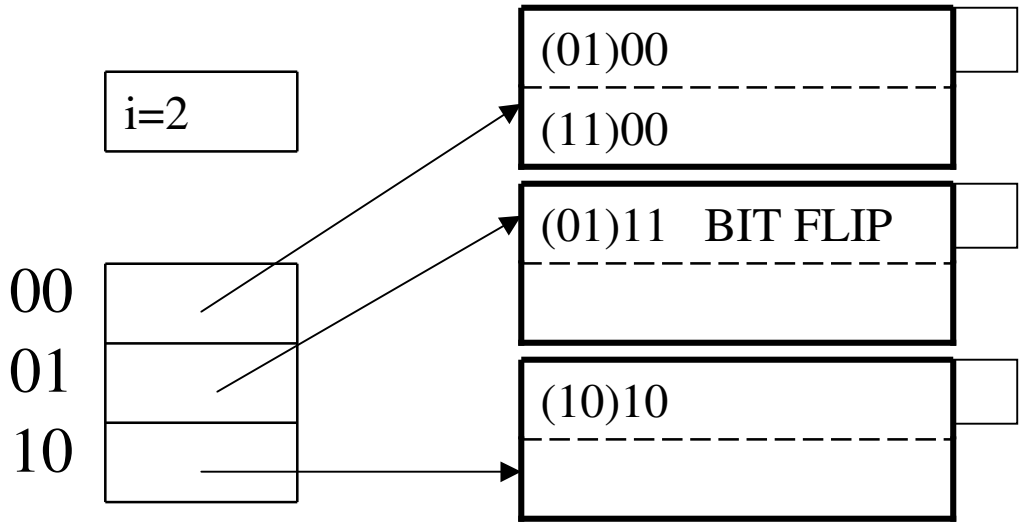
- No overflow blocks: access always one read
- BUT:
  - Extensions can be costly and disruptive
  - After an extension table may no longer fit in memory

Linear Hash Table

- Idea: extend only one entry at a time
- Problem: n= no longer a power of 2
- Let i be such that $2^i <= n < 2^{i+1}$
- After computing h(k), use last i bits:
  - If last i bits represent a number > n, change msb from 1 to 0 (get a number <= n)

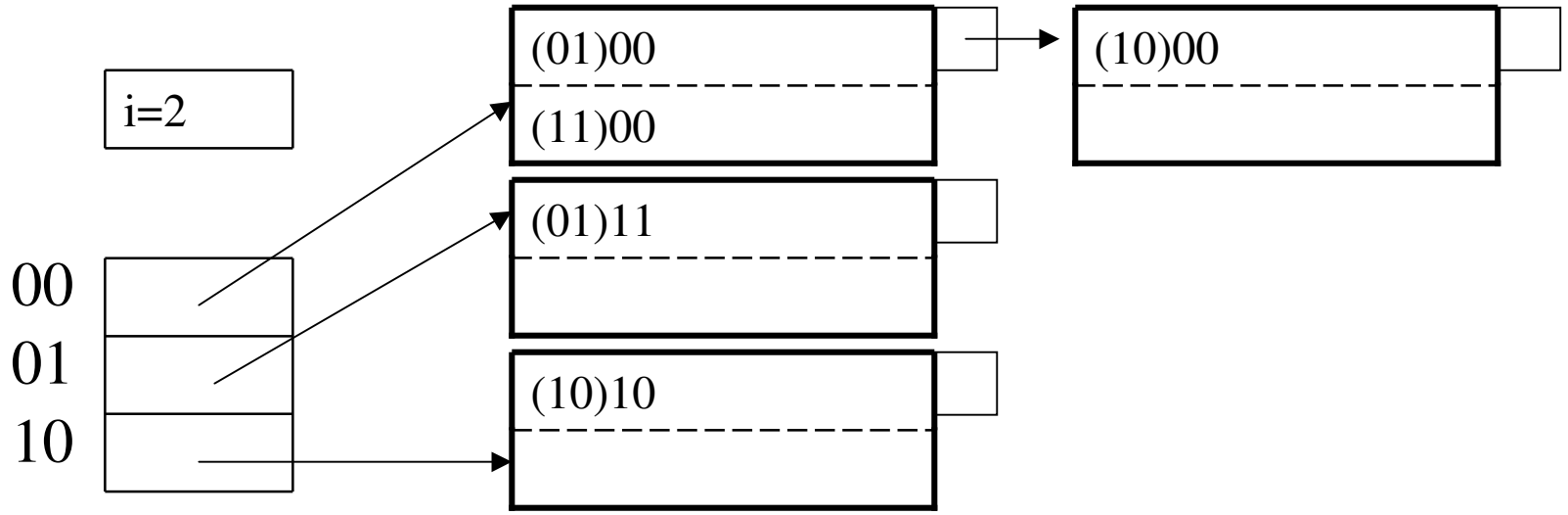
Linear Hash Table Example

- N=3



Linear Hash Table Example

- Insert 1000: overflow blocks…

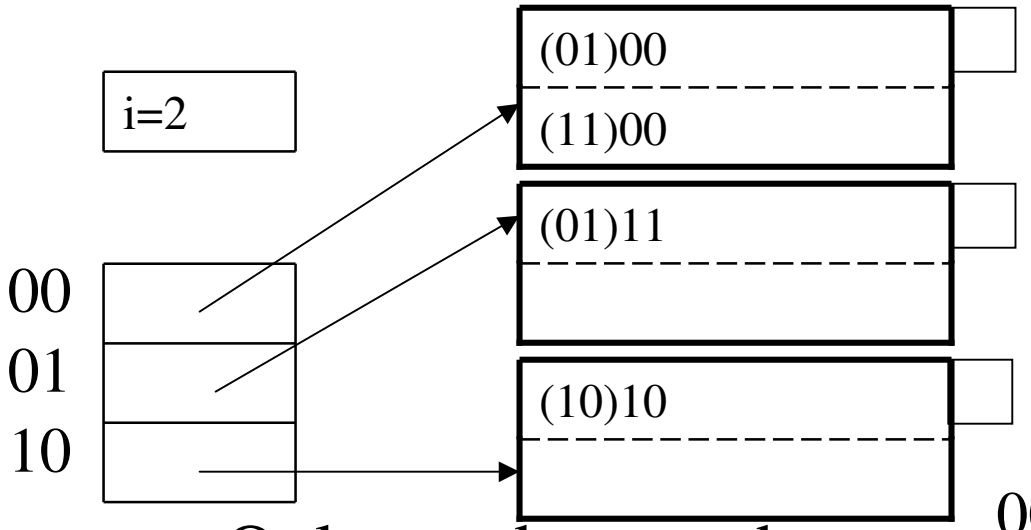


Linear Hash Tables

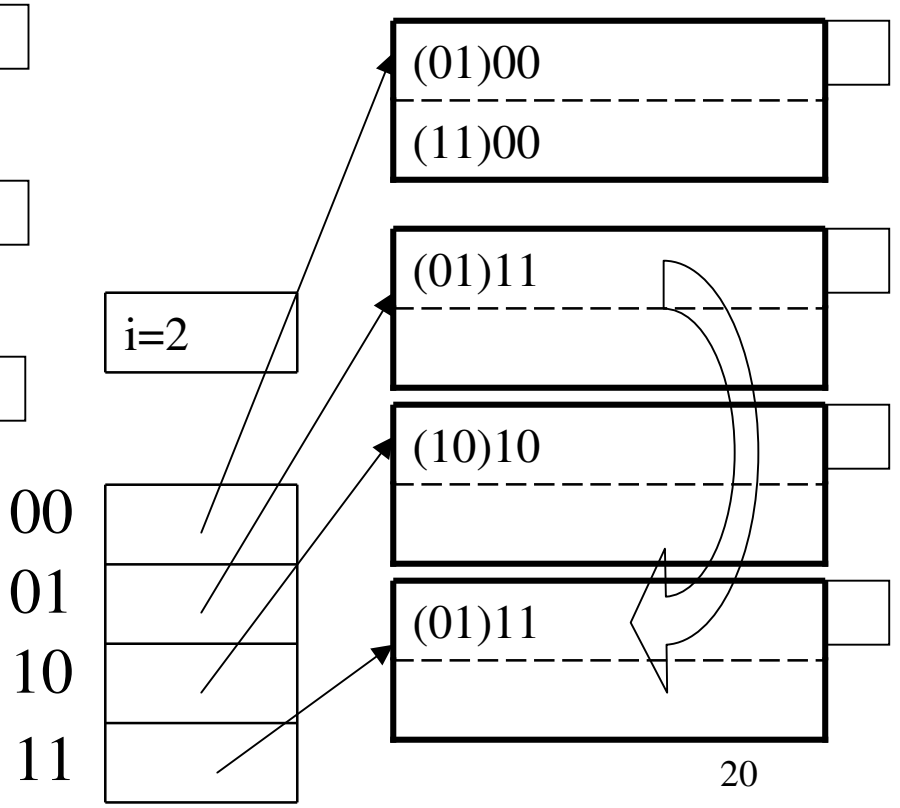
- Extension: independent on overflow blocks
- Extend n:=n+1 when average number of records per block exceeds (say) 80%

Linear Hash Table Extension

- From n=3 to n=4



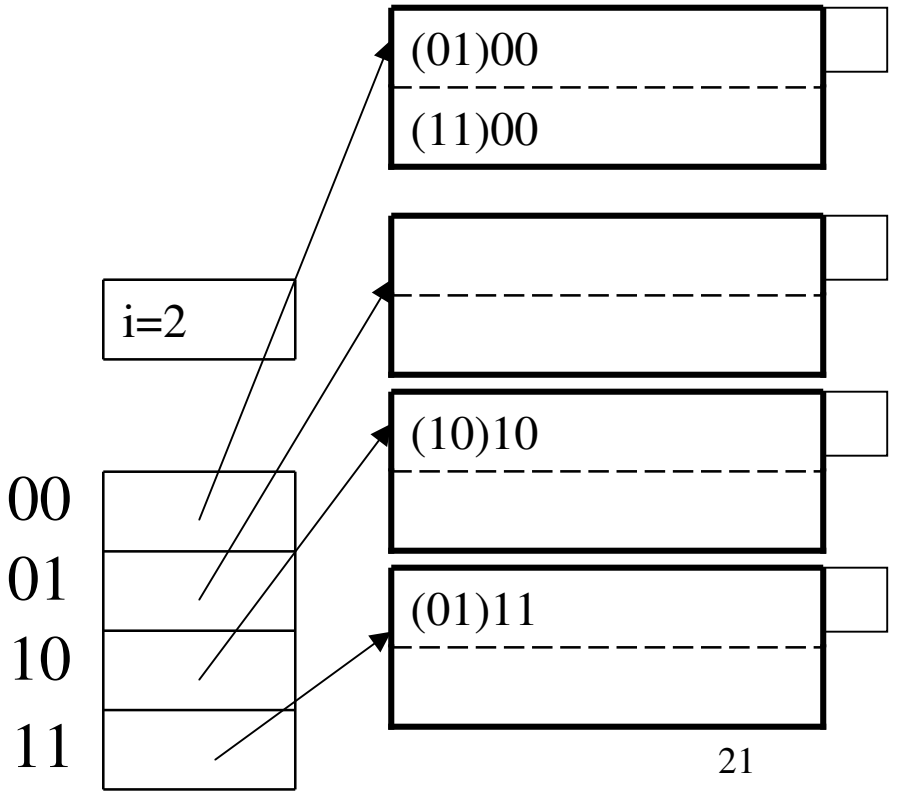
- Only need to touch one block (which one ?)



Linear Hash Table Extension

- From n=3 to n=4 finished

- Extension from n=4 to n=5 (new bit)
- Need to touch every single block (why ?)



Discussion of Physical Operators

The following discussion is based mostly on:

Goetz Graefe

Query evaluation techniques for large databases

Discussion of Physical Operators

General questions:

- What is the difference between *physical* algebra and *logical* algebra ?

- What is the *iterators model* ?

Discussion of Physical Operators

Mergesort questions:

- What are the two methods for creating the level-0 runs ? Describe their pros and cons.

Discussion of Physical Operators

Suppose we only allow two passes in *merge-sort*:
(1) build level 0 runs, (2) merge them

(XMLTK sorts this way)

Assume M = 128MB, page size = 4kB

- What is the largest relation size we can sort ?
- How does this change if we double M (to 256MB) ?
- How does this change if double the page size (to 8kB) ?

What conclusions do you draw from this slide ?

Discussion of Physical Operators

Suppose we only allow two passes in *merge-join*:
(1) build level 0 runs for R and S,
(2) merge and join them

Assume M = 128MB, page size = 4kB

- Write down the condition on the size of R and/or S that allows us to do merge-join in two passes

Discussion of Physical Operators

Consider *partitioned hash-join* (described in the book, not the paper).

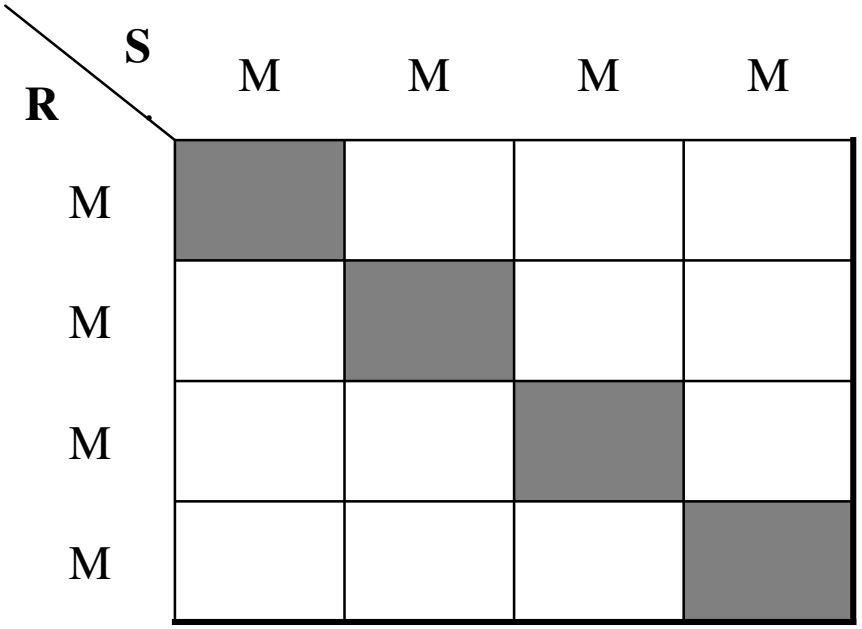
Assume M = 128MB, page size = 4kB

- Write down the condition on the size of R and/or S that allows us to do partitioned hash-join

Discussion of Physical Operators

Block nested loop join v.s. partitioned hash-join.

Assuming R=S. When is one better than the other ?



Discussion of Physical Operators

Hybrid hash join

- This is difficult.
  What does it really buy us ?

Discussion of Physical Operators

More questions on joins

- What is an antisemijoin ?

Discussion of Physical Operators

Object oriented databases and pointers

- Comment on the following statement:

> Object-oriented databases have little need for joins. Foreign keys are usually replaced with pointers, like in:
>
> **Person(ssn, name, deptid),**
> **Department(name)**
>
> **deptid** is a pointer to a **department**. A join in the relational model is now replaced with a traversal of a physical pointer, which is far more efficient.

Discussion of Physical Operators

Parallel databases

- Describe briefly the following three forms of parallelism:
  - *Interquery* parallelism
  - *Interoperator* parallelism
  - *Intraoperator* parallelism
- Assuming you implement a huge, single-user database and buy a parallel machine with 128 nodes. Which form of parallelism has best potential for speedup ?
- Describe *speedup* and *scaleup*

Discussion of Physical Operators

More questions:

- What are NF^2 relations ?
- Suppose R is on node 1, and S is on node 2. Describe the following distributed join computation methods:
  - Semijoins
  - Bloom filters