

CS 241 Discussion Section
(12/1/2011)

Tradeoffs

- When do you:
 - Expand
 - Increase total memory usage
 - Split
 - Make smaller chunks (avoid internal fragmentation)
 - Coalesce
 - Make bigger chunks (avoid external fragmentation)

Basic Allocator Mechanisms

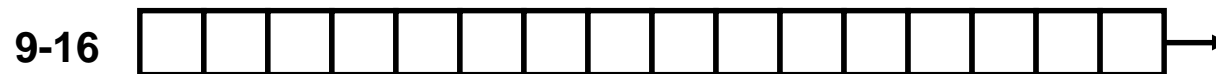
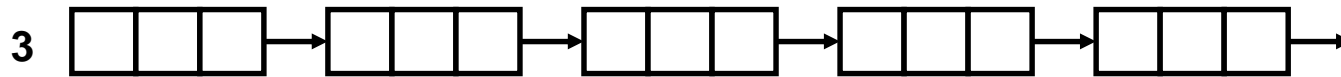
- Sequential fits (implicit or explicit single free list)
 - best fit, first fit, or next fit placement
- Tradeoffs
 - Expand: No fit
 - Split: Threshold
 - Coalesce: Immediate or Deferred

Basic allocator mechanisms

- Segregated free lists
 - simple segregated storage -- separate heap for each size class
 - segregated fits -- separate linked list for each size class
- Tradeoffs
 - Expand: No big blocks
 - Split: No “right” sized blocks
 - Coalesce: Immediate or Deferred

Segregate Storage

- Each size “class” has its own collection of blocks



- Often have separate collection for every small size (2,3,4,...) FAST
- For larger sizes typically have a collection for each power of 2 EFFICIENT

Simple segregated storage

- Separate heap and free list for each size class
- No splitting
- To allocate a block of size n :
 - if free list for size n is not empty,
 - allocate first block on list (note, list can be implicit or explicit)
 - if free list is empty,
 - get a new page
 - create new free list from all blocks in page
 - allocate first block on list
 - constant time
- To free a block:
 - Add to free list
 - If page is empty, return the page for use by another size (optional)
- Tradeoffs:
 - fast, but can fragment badly

Segregated fits

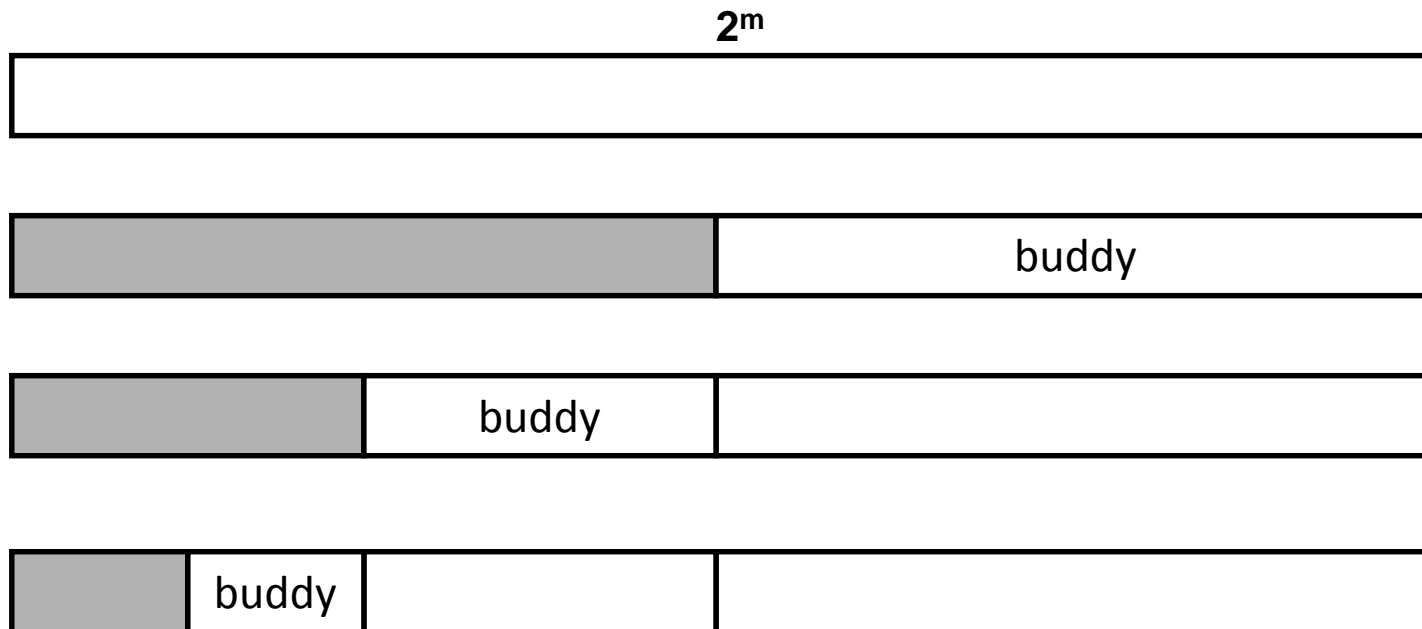
- Array of free lists, each one for some size class
- To allocate a block of size n :
 - search appropriate free list for block of size $m > n$
 - if an appropriate block is found:
 - split block and place fragment on appropriate list (optional)
 - if no block is found, try next larger class
 - repeat until block is found
- To free a block:
 - coalesce and place on appropriate list (optional)
- Tradeoffs
 - faster search than sequential fits (i.e., log time for power of two size classes)
 - controls fragmentation of simple segregated storage
 - coalescing can increase search times
 - deferred coalescing can help

Buddy systems

- Special case of segregated fits.
 - all blocks are power of two sizes
- Basic idea:
 - Heap is 2^m words
 - Maintain separate free lists of each size 2^k , $0 \leq k \leq m$.
 - Requested block sizes are rounded up to nearest power of 2.
 - Originally, one free block of size 2^m .

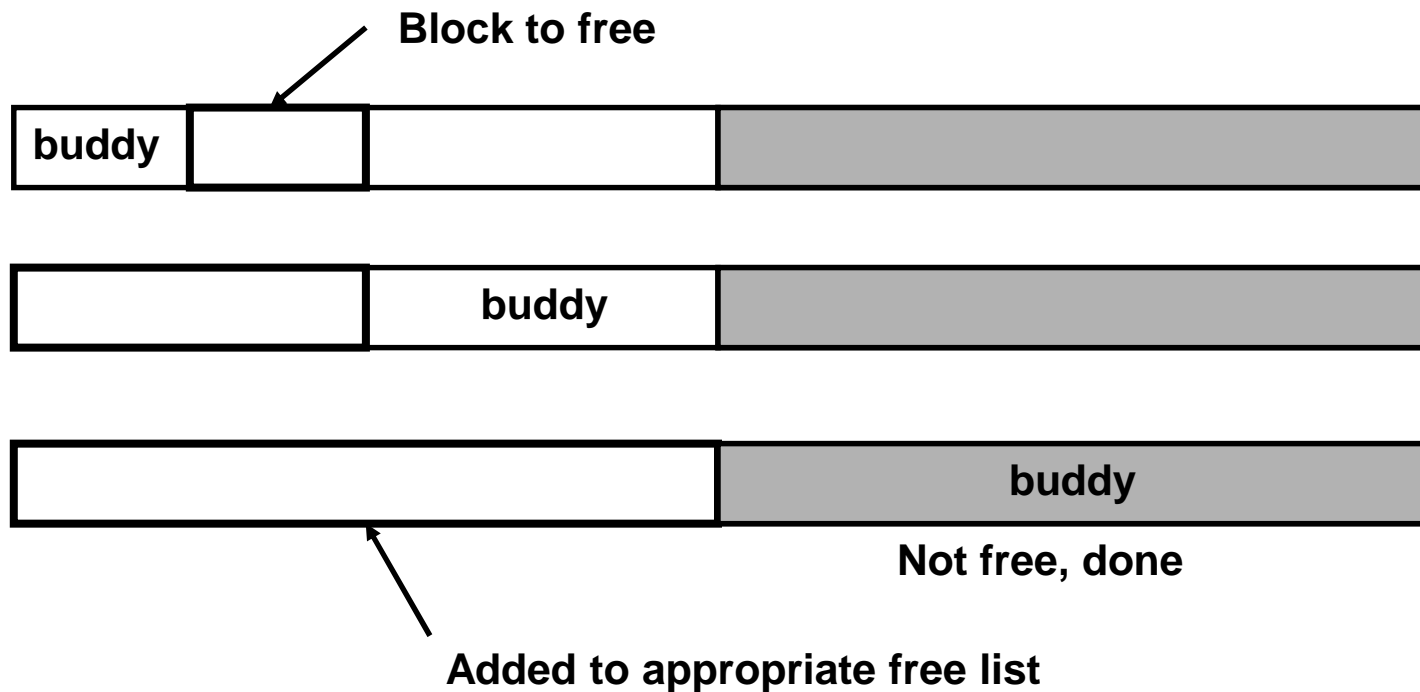
Buddy systems (cont)

- To allocate a block of size 2^k :
 - Find first available block of size 2^j s.t. $k \leq j \leq m$.
 - if $j == k$ then done.
 - otherwise recursively split block until $j == k$.
 - Each remaining half is called a “buddy” and is placed on the appropriate free list



Buddy systems (cont)

- To free a block of size 2^k
 - continue coalescing with buddies while the buddies are free



Buddy systems (cont)

- Key fact about buddy systems:
 - given the address and size of a block, it is easy to compute the address of its buddy
 - e.g., block of size 32 with address `xxx...x00000` has buddy
`xxx...x10000`
- Tradeoffs:
 - fast search and coalesce
 - subject to internal fragmentation

Internal fragmentation

- Internal fragmentation is wasted space inside allocated blocks:
 - minimum block size larger than requested amount
 - e.g., due to minimum free block size, free list overhead
 - policy decision not to split blocks
 - e.g., buddy system
 - Much easier to define and measure than external fragmentation.

Other Sources of Wisdom

- Many implementations and algorithms online...
- All work should be your own!
- Good Luck