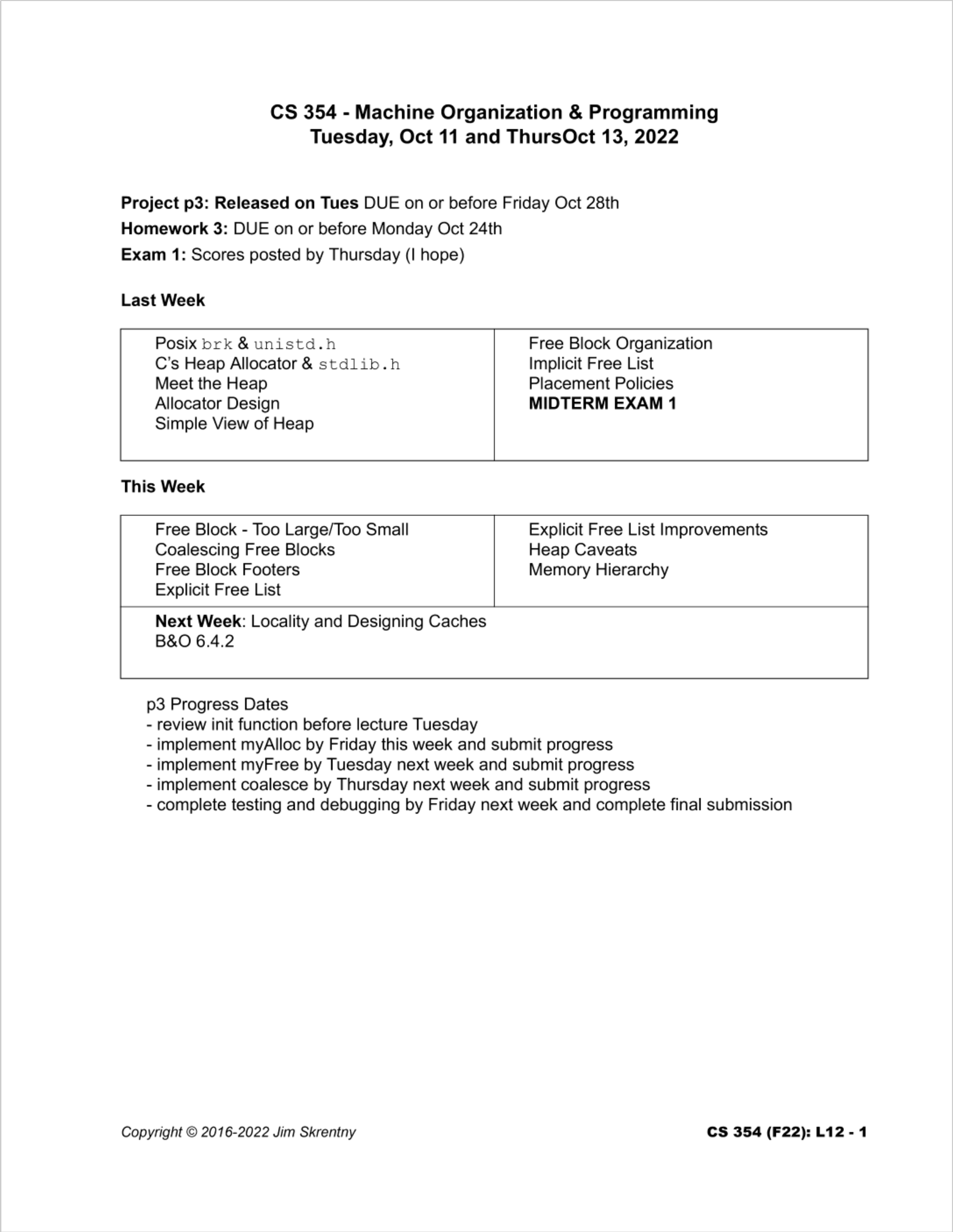
outlineL12-w6TR-student

Monday, October 17, 2022 7:25 PM

outlineL12-w6TR-student



CS 354 - Machine Organization & Programming
Tuesday, Oct 11 and ThursOct 13, 2022

Project p3: Released on Tues DUE on or before Friday Oct 28th

Homework 3: DUE on or before Monday Oct 24th

Exam 1: Scores posted by Thursday (I hope)

Last Week

Posix `brk` & `unistd.h`
C’s Heap Allocator & `stdlib.h`
Meet the Heap
Allocator Design
Simple View of Heap

Free Block Organization
Implicit Free List
Placement Policies
MIDTERM EXAM 1

This Week

Free Block - Too Large/Too Small
Coalescing Free Blocks
Free Block Footers
Explicit Free List

Explicit Free List Improvements
Heap Caveats
Memory Hierarchy

Next Week: Locality and Designing Caches
B&O 6.4.2

p3 Progress Dates
- review init function before lecture Tuesday
- implement myAlloc by Friday this week and submit progress
- implement myFree by Tuesday next week and submit progress
- implement coalesce by Thursday next week and submit progress
- complete testing and debugging by Friday next week and complete final submission

Free Block - Too Large/Too Small

What happens if free block chosen is bigger than the request?

- Use Entire Block

— mem util: more internal fragmentation

+ thruput: Fast + simple code

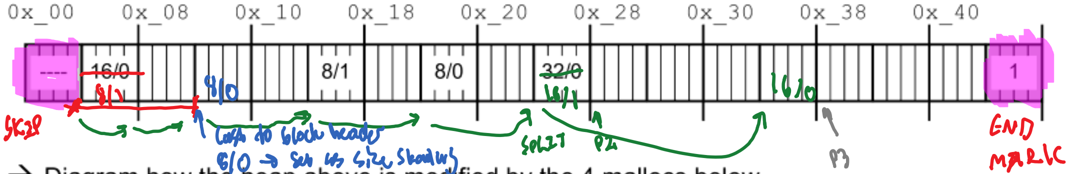
- Split Block (1st block – allocated one) 2nd block – free block

p3 solution

+ mem util: less internal fragmentation

— thruput: more heap blocks, slower

Run 4: Heap First-Fit Allocation with Splitting



→ Diagram how the heap above is modified by the 4 mallocs below.
For each, what address is assigned to the pointer?
If there is a new free block, what is its address and size?

		PTR	NEW FREE BLOCK	
1) `p1 = malloc(sizeof(char));`	4+1+3= 8 +1=9	0x_08	0x_06	
2) `p2 = malloc(11 * sizeof(char));`	4+11+1=16	0x_28	0x_34	Return 0x0
3) `p3 = malloc(2 * sizeof(int));`	4+8=12+4=16	0x_38	no split – sum	↑
4) `p4 = malloc(5 * sizeof(int));`	4+20+0=24	N/A	N/A	ALLOC FAILS

What happens if there isn't a large enough free block to satisfy the request?

1st. Coalesce ('merge') free blocks that are adjacent "delayed coalesce" (wait to do until fail)

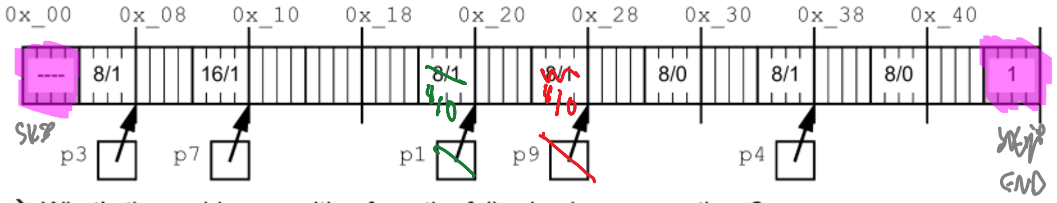
→ Can allocated blocks be moved out of the way to create larger free areas?

2nd. Ask kernel for more memory for heap (not p3)

3rd. Return NULL = Fail, 0x0 (p3)

Coalescing Free Blocks

Run 5: Heap Freeing without Coalescing



→ What's the problem resulting from the following heap operations?

🟥 1) `free(p9); p9 = NULL;`
🟩 2) `free(p1); p1 = NULL;`
■ 3) `p1 = malloc(4 * sizeof(int));` 4+16+4=24 → P3 = fail

technically enough space

Problem? FAULTS FRAGMENTATION

there is enough free contiguous space - but is divided into small blocks

Solution? COALESCE FREE BLOCKS → coalesce free blocks with previous and next if possible

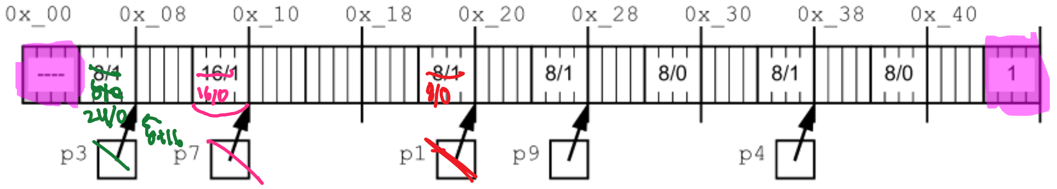
immediate:

delayed: coalesce in alloc() function if needed, as you search for free block

(P3) coalesce by called function

Run 6: Heap Freeing with Immediate Coalescing

On paper



→ Given the heap above, what is the size in bytes of the freed heap block?

1) `free(p7); p7 = NULL;` free 16 → look at next/prev bytes. Since they are both alloc'd, no coalescing.

→ Given a pointer to a payload, how do you find its block header?

0x_10 0x_0C ptr - sizeof(HDR)

→ Given a pointer to a payload, how do you find the block header of the NEXT block?

ptr - sizeof(HDR + block size)

curr_block_size

char * S.F.=1

⚹ *Use type casting* to set correct scale factor

void * S.F.=1

→ Given the modified heap above, what is the size in bytes of the freed heap block when immediate coalescing is used? 0 + 8+16 = 24

2) `free(p3); p3 = NULL;`
3) `free(p1); p1 = NULL;` $\boxed{24}$ + 8+0 = 32 (next) - nothing

→ Given a pointer to a payload, how do you find the block header of the PREVIOUS block?

(ptr - 4) - prev_block_size

Free Block Footers

❋ *The last word of each free block* is a footer containing free block size

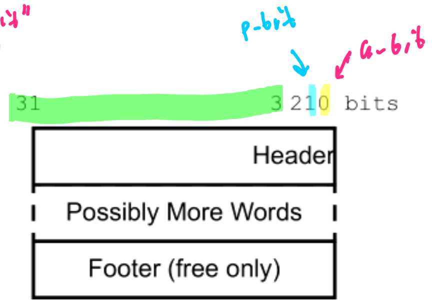
→ Why don't allocated blocks need footers? not free, can't be coalesced

→ If only free blocks have footers, how do we know if previous block will have a footer?

Have to track it, add a "p-bit" somewhere if pbit=1, Prev block alloc'd pbit=0 free

❋ *Free and allocated block headers* also encode a "p-bit"

Layout 2: Heap Block with Headers & Free Block Footers



How can we check p-bit
size_status & mask

→ What integer value will the header have for an allocated block that is:

1) 8 bytes in size and prev. block is free?

1001 = 9 = 8/01

2) 8 bytes in size and prev. block is allocated?

1011 = 11 = 10/11

3) 32 bytes in size and prev. block is allocated?

100011 = 35 = 32/11

4) 64 bytes in size and prev. block is free?

1000001 = 65 = 64/01

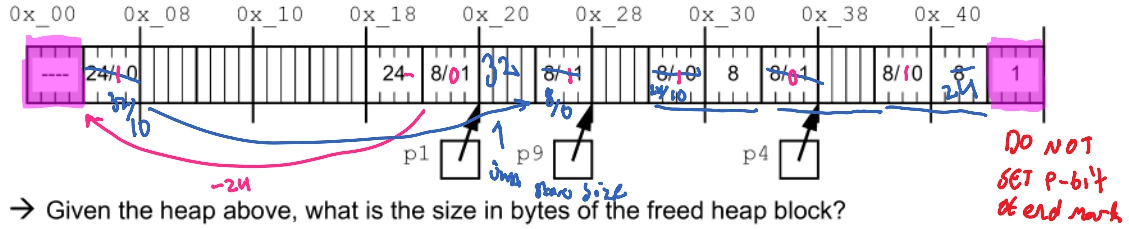
→ Given a pointer to a payload, how do you get to the header of a previous block if it's free?

1. set to HDR, ptr-4

2. Check p-bit if p-bit is Ø, can coalesce with previous

3. get previous block header (ptr-4) - prev_block_size

Run 7: Heap Freeing with Immediate Coalescing using p-bits and Footers



→ Given the heap above, what is the size in bytes of the freed heap block?

1) `free(p1); p1 = NULL;` 24 + 8 + 0 = 32
prev cur next

→ Given the modified heap above, what is the size in bytes of the freed heap block?

2) `free(p4); p4 = NULL;` 8 + 8 + 8 = 24
prev curr next

❋ *Don't forget to update* p-bit of next block, footer of free block

➢ Is coalescing done in a fixed number of steps (constant time) or is it dependent on the number of heap blocks (linear time)? → O(1)

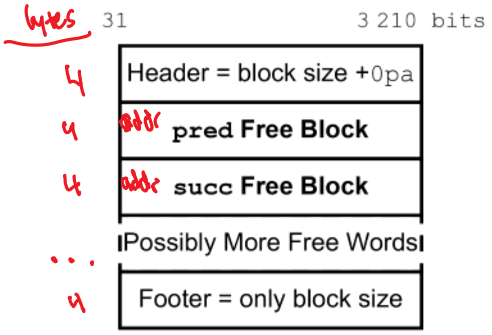
Copyright © 2016-2022 Jim Skrentny

Explicit Free List

- *An allocator using an explicit free list* only keeps list of free blocks

This list can be integrated and stored in the heap, by specifying a special layout for free blocks.

Explicit Free List Layout: Heap Free Block with Footer



Free Block Links

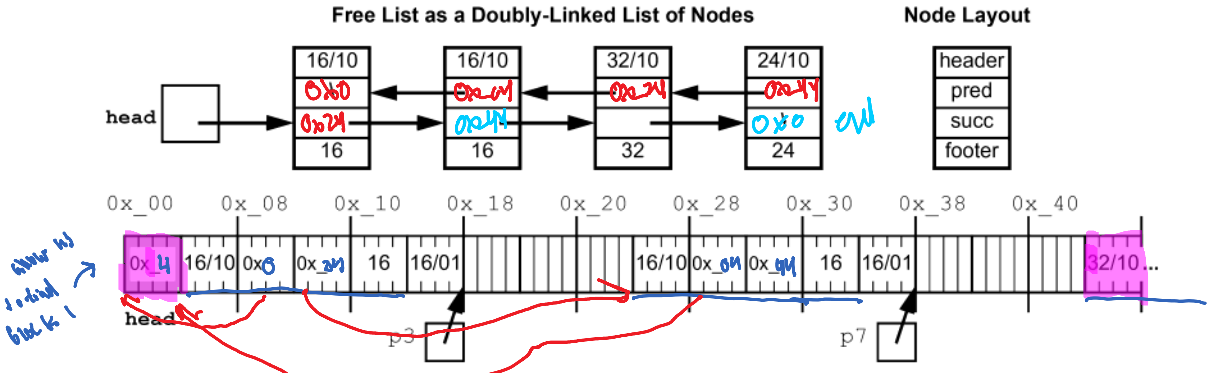
pred address of previous free block

succ address of next free block

4+8=12+~~4~~4=16

16/1 = 17

→ Complete the addresses in the partially shown heap diagram below.



→ Why is a footer still useful?

Faster coalescing with a previous adjacent block

→ Does the order of free blocks in the free list need to be the same order as they are found in the address space? NO

Explicit Free List Improvements

Free List Ordering

address order: Maintain order of free blocks from low to high addr.

+ malloc with FF Better memory utilization than last-in

— free Slightly slower, O(n) n = free blocks, search for spot to insert

• last-in order: "place most recently freed block at start of EFL

— malloc with FF slower, must go through most recent blocks

+ free much faster, O(1), link at start
O(1) coalesce using footers

Free List Segregation

Keep an array of free list ~ separate list for each block size

SM, MD, LG lists
(16-32) (64-256) (512+)

malloc chooses correct free list to get block

simple segregation: One free list for each block size

structure simple, no need for header, only need block successor address

+ malloc FAST! Just pick right list → get first free block from correct list

if free list is empty ~ ask for more heap from OS, divide into needed sizes
could coalesce from smaller size, or split a longer free block

+ free Fast. O(1) can link freed block to start of the E.F.L.

— problem Internal frag. — no splitting
External frag. — no coalescing

fitted segregation: One explicit free list for each size range (used by gcc)

+ memory util → as good as best fit
+ throughput → search only part of the heap

fitting use first fit of appropriate free list, if fail search next larger list.

splitting put new free block in appropriate list

coalescing put coalesced block into appropriate list.

Heap Caveats

p1 Block → p2

Consecutive heap allocations don't result in contiguous payloads!

→ Why? payloads are interspersed with padding and heap structs

placement policies and heap structure, algs for malloc + free

Don't assume heap memory is initialized to 0!

OS initially clears heap for security

But recycled heap will have old data from process + have heap struct data

Do free all heap memory that your program allocates!

→ Why are memory leaks bad? Slowly they kill the heap by cluttering heap performance with garbage blocks

→ Do memory leaks persist when a program ends? NO!

Don't free heap memory more than once! UNDEFINED BEHAVIOR

SEG FAULT

→ What is the best way to avoid this mistake?

free(ptr); ptr=null;

Don't read/write data in freed heap blocks!

→ What kind of error will result? Intermittent error

Don't change heap memory outside of your payload!

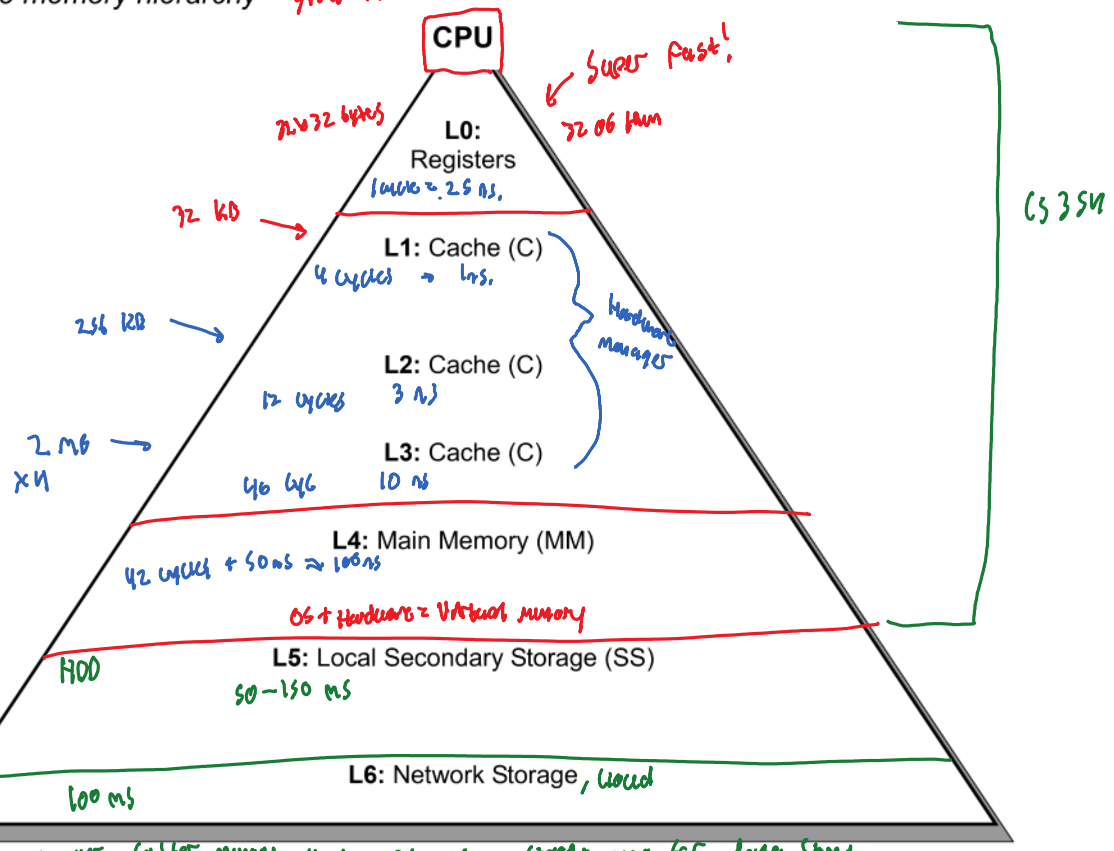
→ Why? Can trash heap struct of another block

Do check if your memory intensive program has run out of heap memory!

→ How? Always check malloc, calloc, realloc return values are not NULL;

Memory Hierarchy

- *The memory hierarchy* gives illusion of having lots of fast memory.



Cache smaller, faster memory that acts as a staging area for data stored in larger slower memory

Memory Units

x86 | x64

word: size used by CPU transfer between CI — 4 byte | 8 byte

block: size used by Cache transfer between cache levels + memory — 32 bytes | 64 bytes

page: size used by MM transfer between MM + SS — 4 KB | 4 KB

Memory Transfer Time

cpu cycles: used to transfer time

latency: mem access time (delay) - to get first

Copyright © 2016-2022 Jim Skrentny