int / word = 4
double = 8
char = 1

stack - local vars
heap - dynamic mem
data - global / static

%d int
%f float
%p pointer address
%s string
%i integer

$S = 2^s$ number of sets
E = number of lines per set
$B = 2^b$ block size
$m = \log_2(M)$ number of add bits

smaller
faster
costlier

CPU
L0 Registers
L1 cache
L2 cache
L3 cache
L4 Main Mem
L5 Local secondary storage
L6 Network storage

larger
slower
cheaper

$M = 2^m$ max # of unique mem addresses

$s = \log_2(S)$ number of set idx bits

$b = \log_2(B)$ # of block offset bits

$t = m - (s+b)$ tag

$C = B \times E \times S$ cache size

9 VirtualMem

virtual memory: uses main memory as cache for address space on disk
- CPU executes, virtual address is translated to physical address, main mem is accessed, sends to memory
address space is made up of non-neg addresses, $N = 2^n$

for more virtual mem we use heap's dynamic mem allocator. grows upward and maintains break which points to the top
- block allocated / free

realloc adjusts
- malloc - allocates block from heap and returns pointer, NULL if error
- calloc - allocates / initializes memory to 0

increment brk - sets to spec address
- sbrk can grow / shrink heap by adding to brk. returns old brk value otherwise -1
- free, free allocated blocks, needs beginning

fragmentation - unused memory can't satisfy req
- internal frag, block is larger than payload
- external frag, enough mem but it's broken up

a block has a
- header, contains block size, header / padding and if alloc'd or free.

first fit - starts at beginning, chooses 1st fit, fails if end
next fit - starts at most recent allocated, then 1st fit, fails if two wrap
best fit - examines every free block, chooses sm size, fails if end

external frag - enough heap mem available, but it's divided into small blox
internal frag - heap mem is used for padding
false fragmentation

solution:
- immediate coalescing, merging anytime adj are free
- differed coalescing, waiting to coalesce

block size	a/f
payload if allocated	001 allocated 000 free
padding if needed	
block size	a/f

↑ footers / boundary tags is header replica

when using explicit free list, you can use
- LIFO, inserts newly freed block at beginning
- maintain address order, address of block < successor
  - requires time, but better utilization

Segregated Free list - to reduce alloc time
you can have lots of free lists which holds blocks of same sz
- partition by size classes

simple segregation - one FFL for each block sz
- no HDR just footer

filted segregation - one free list for each size

movb - move byte
movw - move word
movl - move double word

movsbw - move sign extended byte → word
movsbl - byte → double word
movswl - word → double word

movzbw - zero extended → word
movzbl → double word
movzwl → word → double word

pushl → push double word
popl → pop double word

6 Memory Hierarchy, utilize locality - access storage at a level more often than a lower level

- temporal locality - mem location referenced again multiple times
- spatial locality - mem location is referenced and the nearby mem is likely to be referenced
  - the smaller the stride, better spatial locality
  - smaller loop, greater iterations, better locality

data is copied between levels in block-size transfer units, levels can have diff block sizes

cache hit - when you use k to find a data object from level k+1, and it's then cached it's a hit - k fetches from k+1, possibly overwriting a block

cache miss - if k doesn't have d
- caches can miss if empty (cold cache)

conflict miss - multiple memory location map to the same cache line / set
- direct-mapped, each address → 1 cache line
- set-associative, each address → set
- fully-associative, any block can be stored in any line

Cache writing
- write-through - writes to cache and main memory
- write-back - writes only to cache, then main mem is updated when evicted

cache miss policies
- write allocate - on miss, loads data into cache and writes to main mem
- no write allocate, write directly to main mem bypassing cache

cache format
$C = (S, E, B, m)$

Chapter 3: Machine level Representation

assembly language, low level programming for hardware

char - byte - b - 1, short, word, w, 2
int / long int, double word, 2

IA32 has 8 registers that start with %e but then have diff names

Ex (%esp, %edx, 4), %eax
- adds esp to the value in edx multiplied by 4

leal - Load effective address, variant of movl, reads mem → regist

CF: carry flag - most recent op → sig bit
ZF: zero flag - yields zero
SF: sign flag → neg val
OF: Overflow flag - two compliment overflow neg or pos