Optimizations:
"optimization blocker" – side effect (such as a ++ in a func)
Code motion-identifying computation performed multiple times but whose value never changes
Inlining – substituting body of function in place of function call (disadv-reduces modularity, readability)
Loop unrolling – reduces loop overhead
Reduce Procedure Calls
Eliminating Unneeded Memory References

$$S = \frac{1}{(1-\alpha) + {}^{\alpha}/_{k}}$$

Amdahl's Law
S = speed up
a = fraction being sped up
*%of time consumed before optimization
k = speed up factor

$$Disk\ Capacity = \frac{\#bytes}{sector} \cdot \frac{avg\ \#\ sector}{track} \cdot \frac{\#tracks}{surface} \cdot \frac{\#surfaces}{platter} \cdot \frac{\#platters}{disk}$$

seek time – delay of positioning arm
+ rotational latency – waiting for sector to pass under head
+ transfer time (1/RPM * 1/(avg # sect/track) * 60sec/min)
= data access time

Caches:
Temporal – items that have been accessed recently
Special – cache items that are nearby
Cache misses
1. compulsory miss (cold miss)
   by the first reference to a datum
2. conflict cache miss
   could have been avoided, had the cache not evicted an entry earlier
3. capacity cache miss ->thrashing
   occur regardless of associativity or block size, solely due to the finite size of the cache

(S,E,B,m):
S: # of sets
E: lines per set
B: #blocks of cache/ line

Valid	Tag	Blocks	>	>	>	>	>		

Procedures:
%eax, %edx, %ecx – caller save registers (P)
%ebx, %esi, %edi – callee save registers (Q)
P assumes value won't change

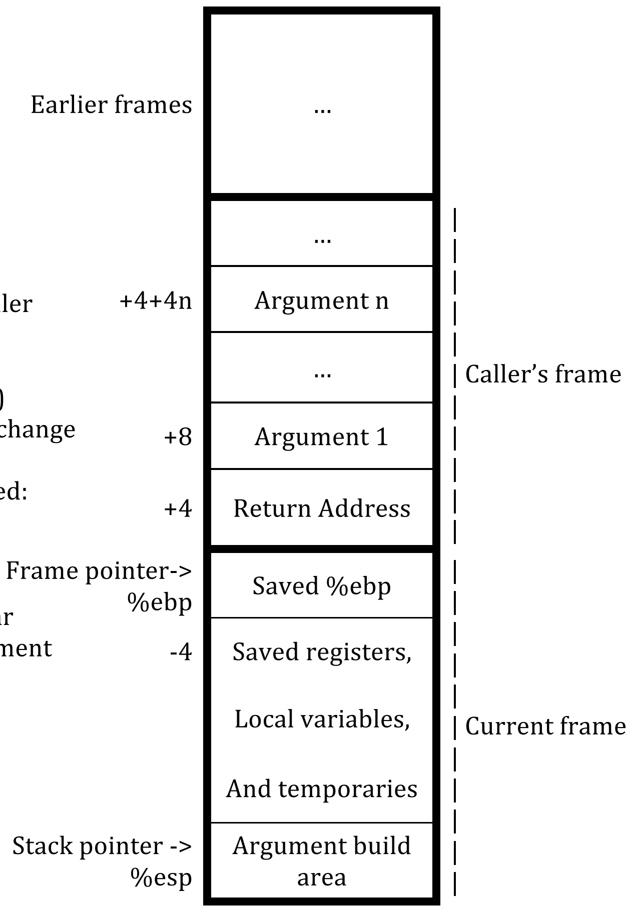
How are structs returned:
As a hidden pointer

Structs passed:
Sequentially… as regular arguments w/ each element passed

Linking:
Def: process of collecting & code and data into a single file to be loaded-copied- in memory and executed
1. compile time
2. load time
3. run time

Static linking
1. symbol resolution
2. relocation
   a. merges code & data
   b. takes assembler's code & data which starts at address 0
      updates all references to reflect new position

Why linkers?
1. time efficiency (e.g. compile)
2. space efficiency (e.g. libraries)


Earlier frames
…
…
+4+4n
Argument n
…
+8
Argument 1
+4
Return Address
Caller's frame
Frame pointer-> %ebp
Saved %ebp
-4
Saved registers,
Local variables,
And temporaries
Current frame
Stack pointer -> %esp
Argument build area