bmalloc

Filip Pizlo
Apple

- What is bmalloc?
- How the GC uses bmalloc
- Geoff’s bmalloc slides

What is bmalloc?

- Immix-style bump/line allocator
- Used for virtually all memory allocation in WebKit
- Fastest malloc for WebKit
- My manager’s year-old solo project

GC and bmalloc

- GC requests memory from bmalloc.
- 16KB blocks are allocated using bmalloc's memalign.
- GC objects larger than about 8KB are allocated directly from bmalloc.

elasticity

Geoff's slides

BMALLOC


MALLOC

BMALLOC

u

m

p

WHY MALLOC?

WHY MALLOC?

NanoMalloc

ListHashSetNodeAllocator

TCMalloc

System Malloc

x-alloc

dispatch continuations

RenderArena

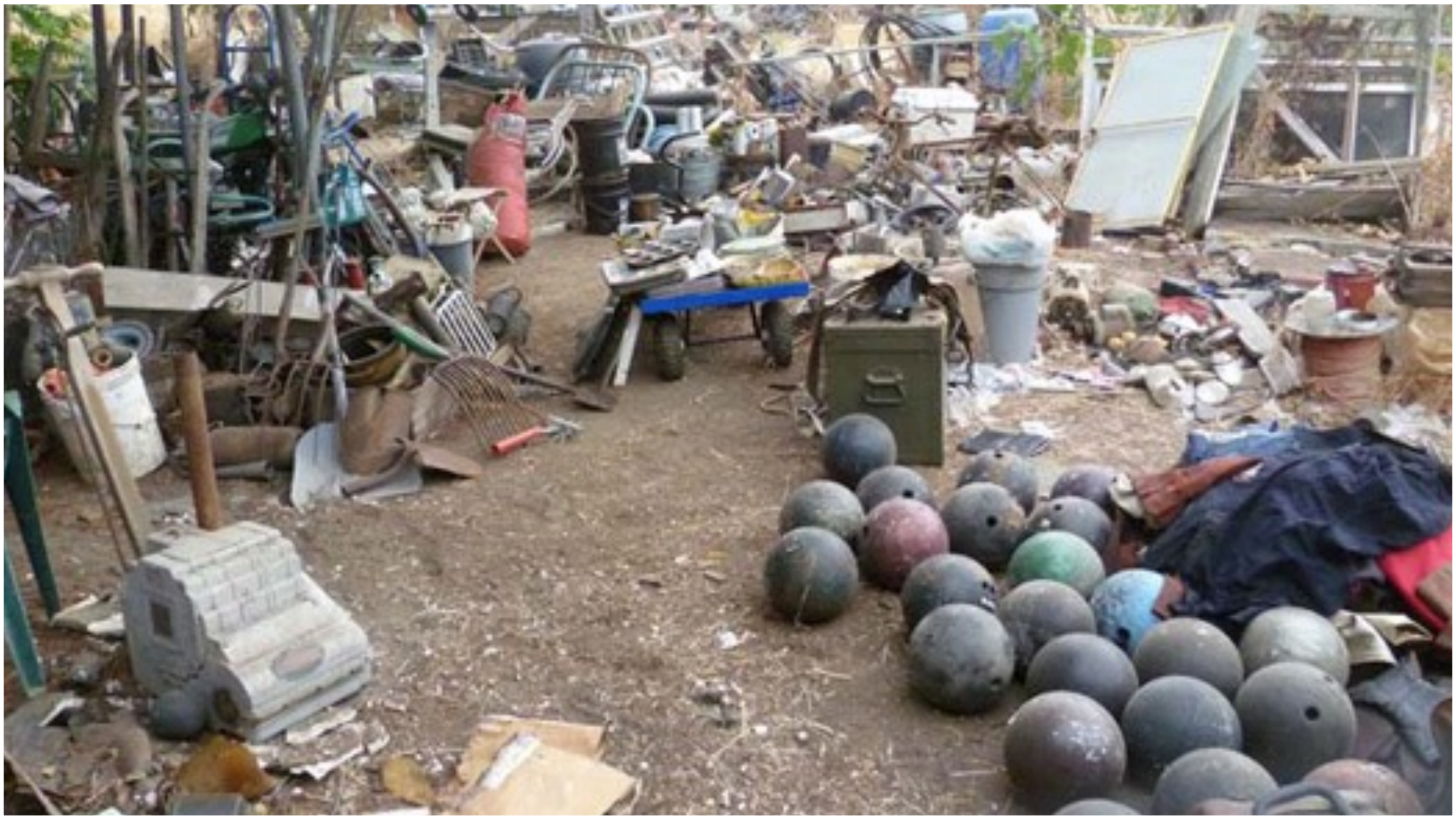
BumpPointerAllocator

BlockAllocator

TPoolAllocator

MetaAllocator

HOARDERS



HOARDERS

HOARDERS

HOARDERS

HOARDERS


System Malloc
ListHashSetNodeAllocator
TCMalloc
dispatch continuations
NanoMalloc
x-alloc
BumpPointerAllocator
BlockAllocator
RenderArena
MetaAllocator
TPoolAllocator

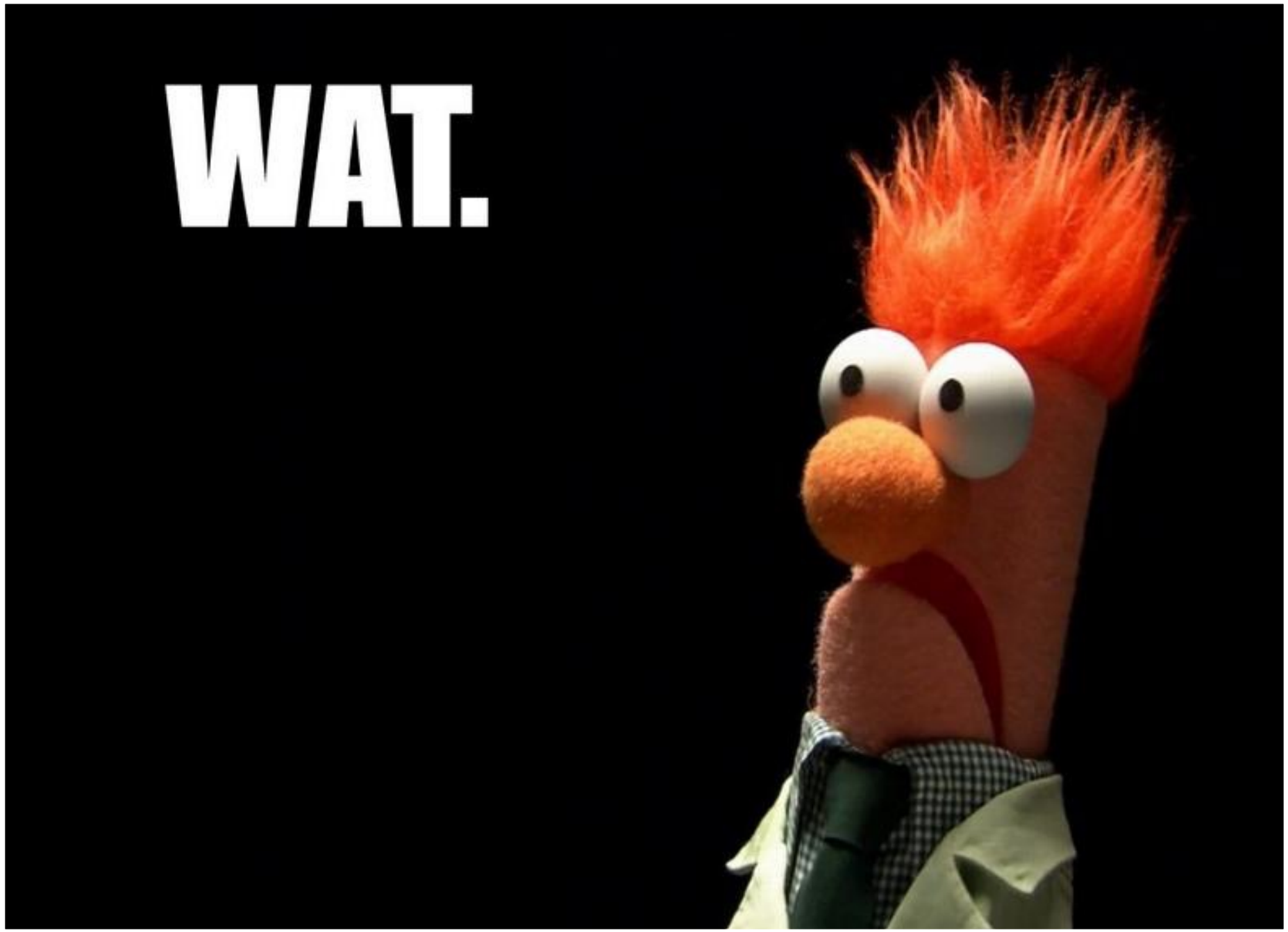
HOARDERS

GARBAGE COLLECTION

"Performance of the [garbage] collector is typically competitive with malloc/free… it may in some cases significantly outperform malloc/free."

–Hans Boehm

GARBAGE COLLECTION



GARBAGE COLLECTION

GARBAGE COLLECTION

GARBAGE COLLECTION

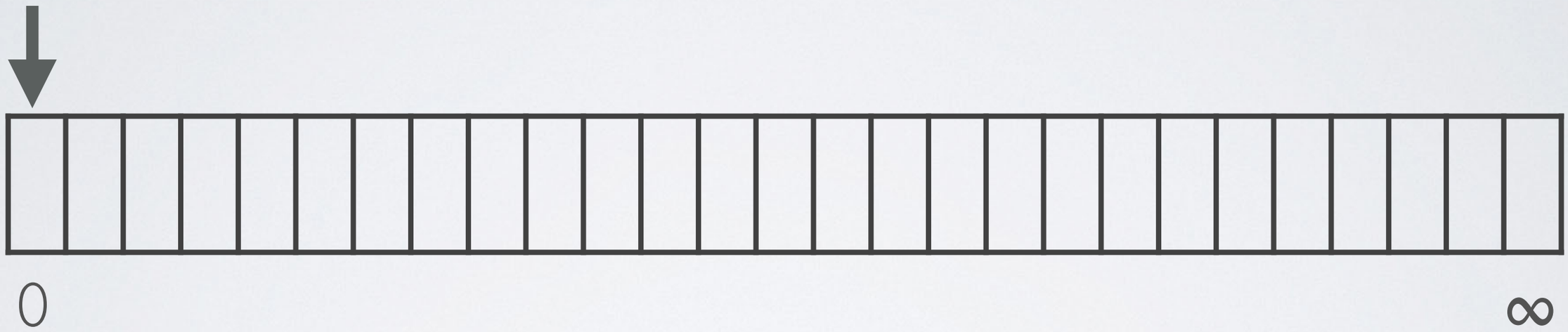
TOPICS

- Memory allocation
- Traditional malloc
- bmalloc

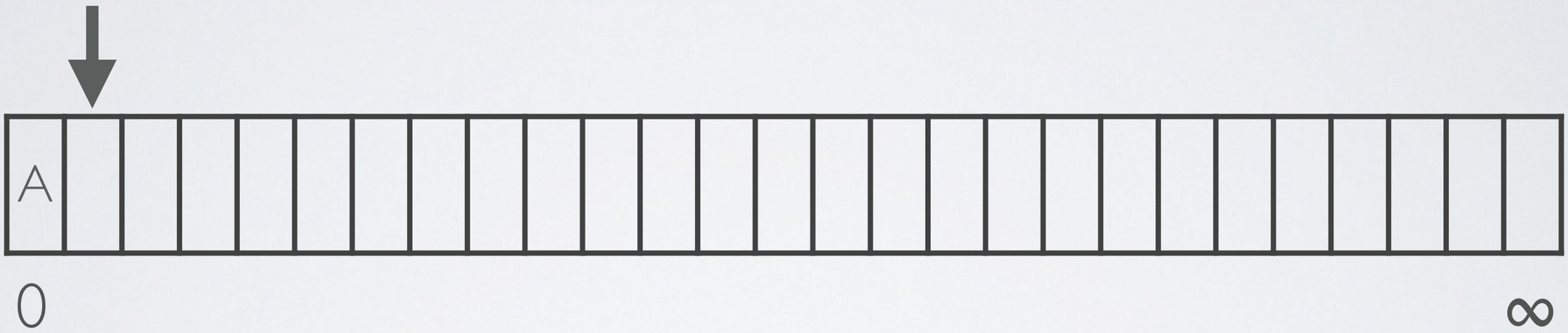
MEMORY ALLOCATION

MEMORY ALLOCATION

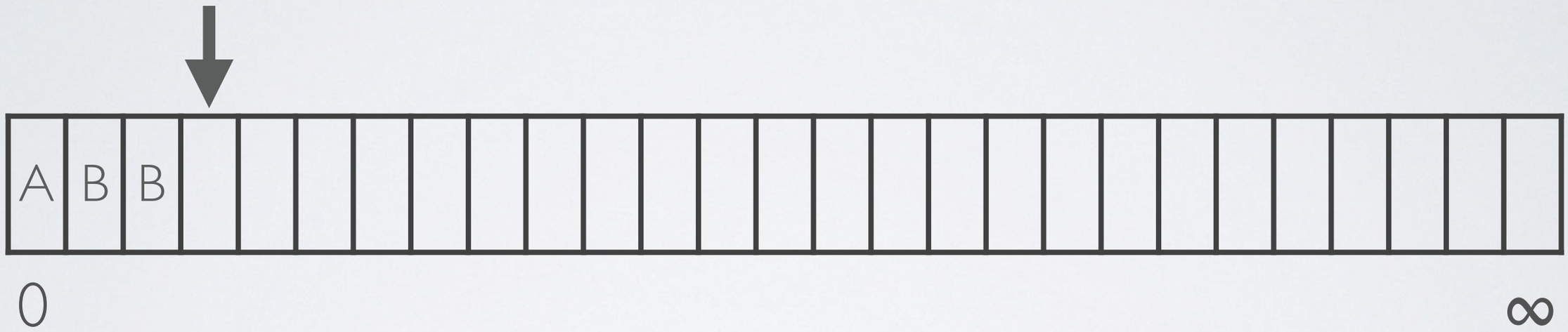
MEMORY ALLOCATION



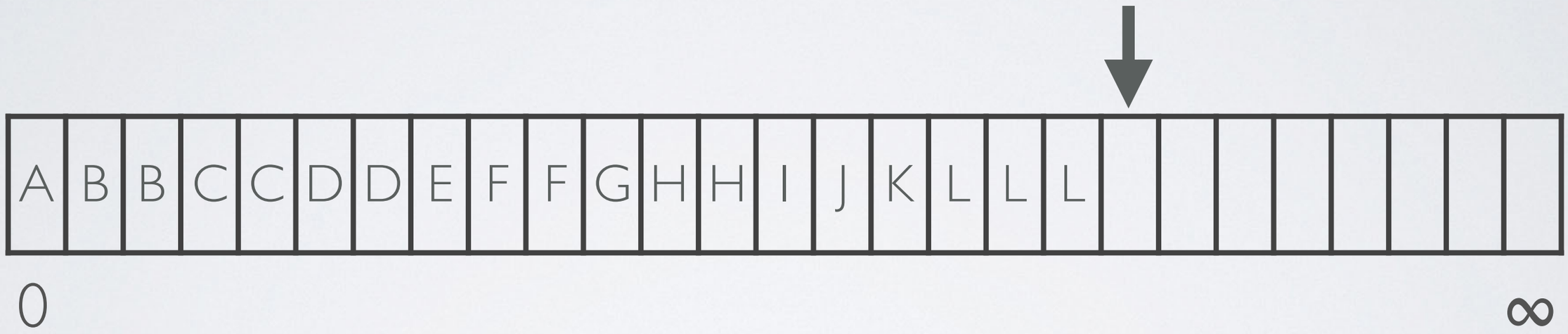
MEMORY ALLOCATION



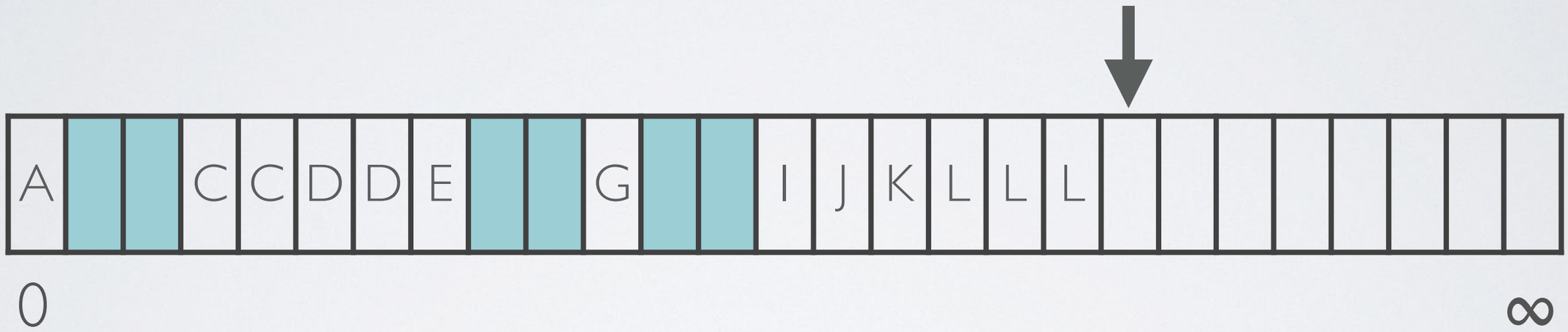
MEMORY ALLOCATION



MEMORY ALLOCATION

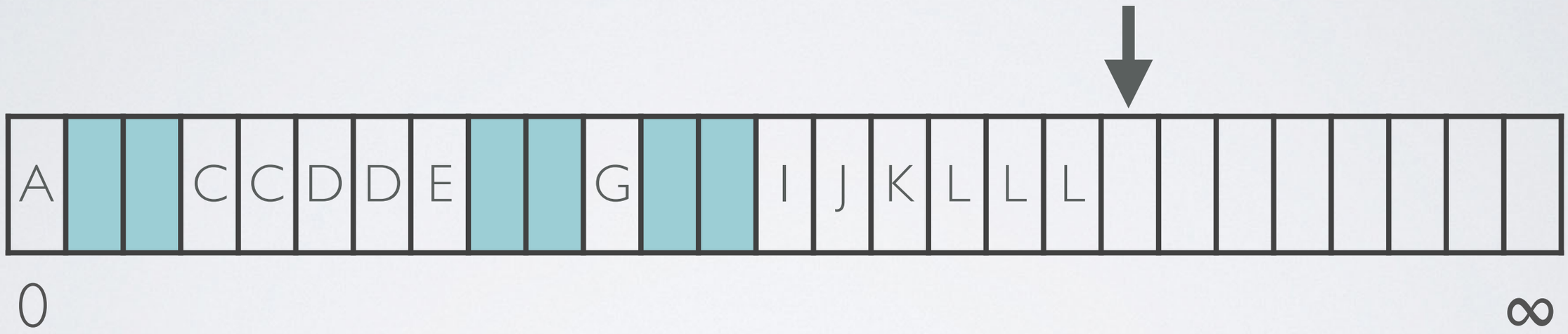


MEMORY ALLOCATION

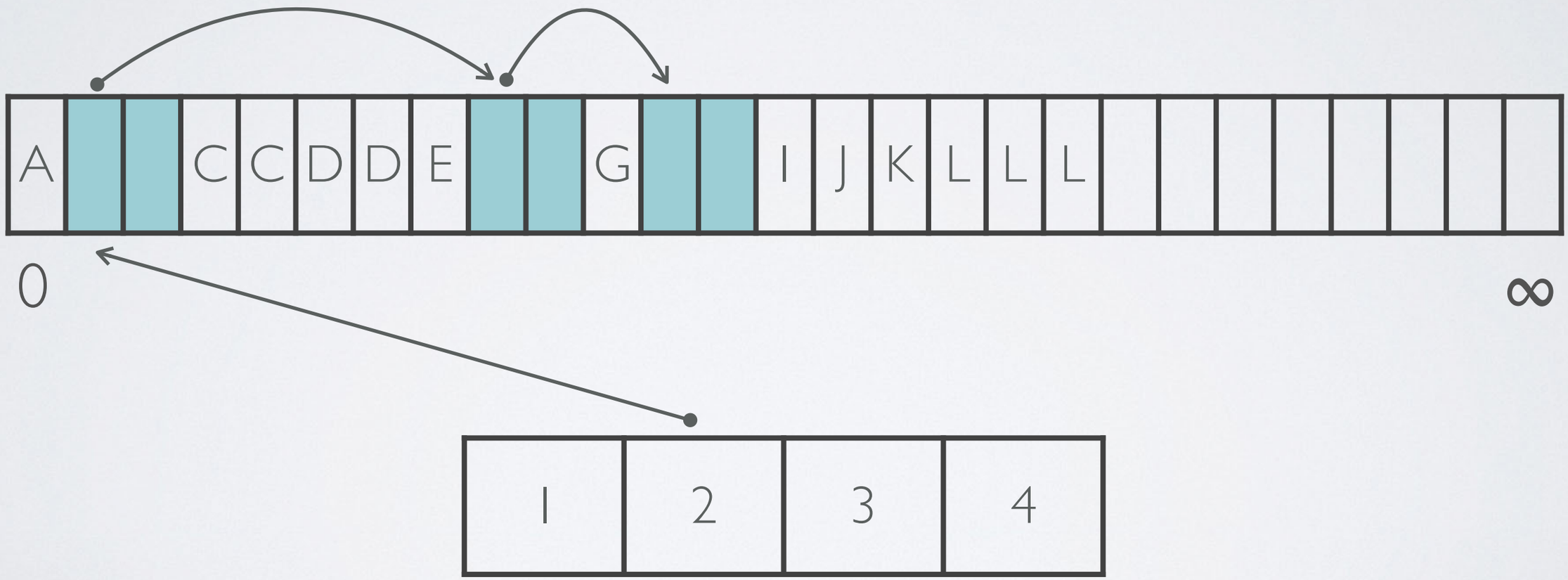


TRADITIONAL MALLOC

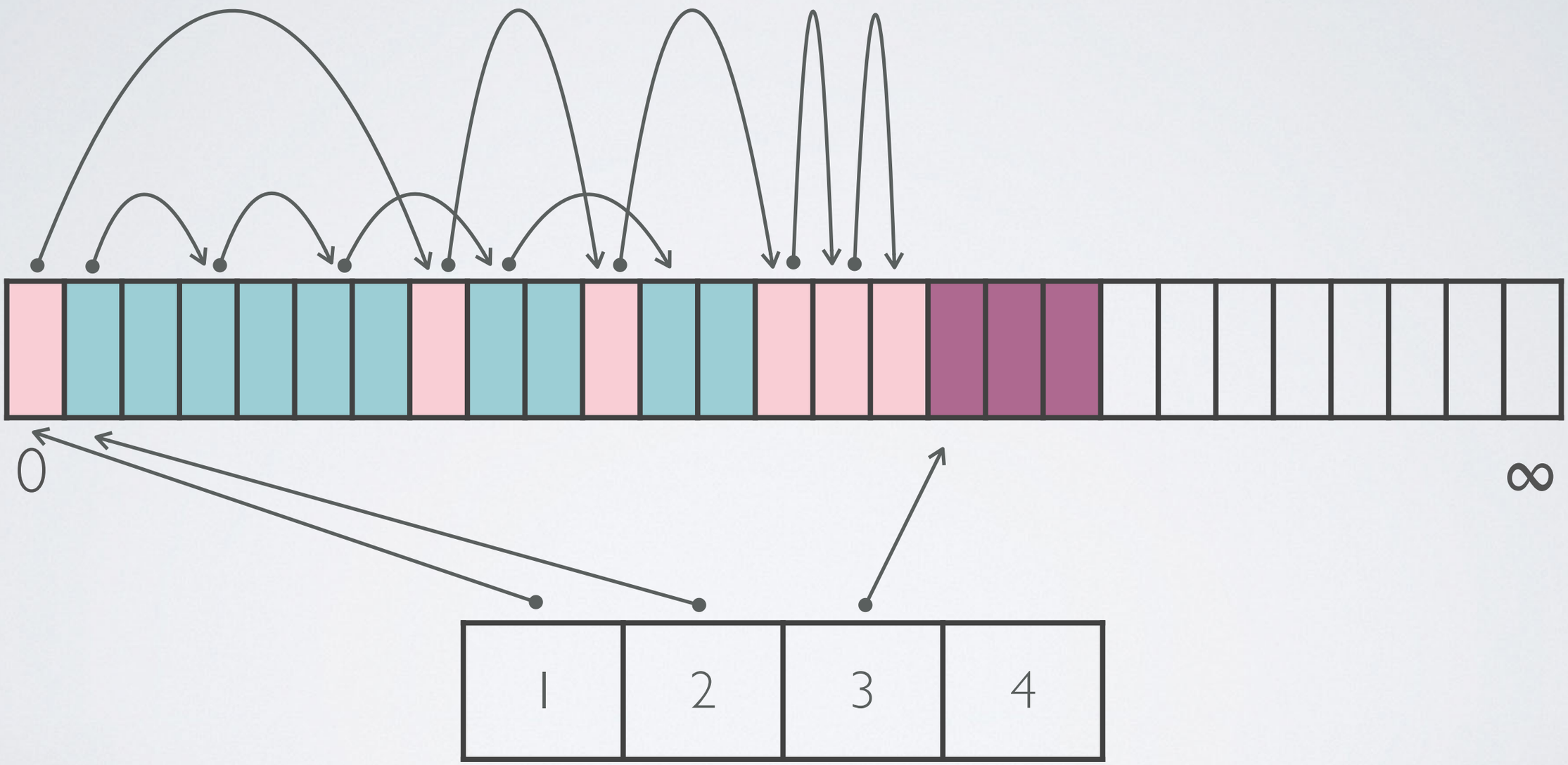
TRADITIONAL MALLOC



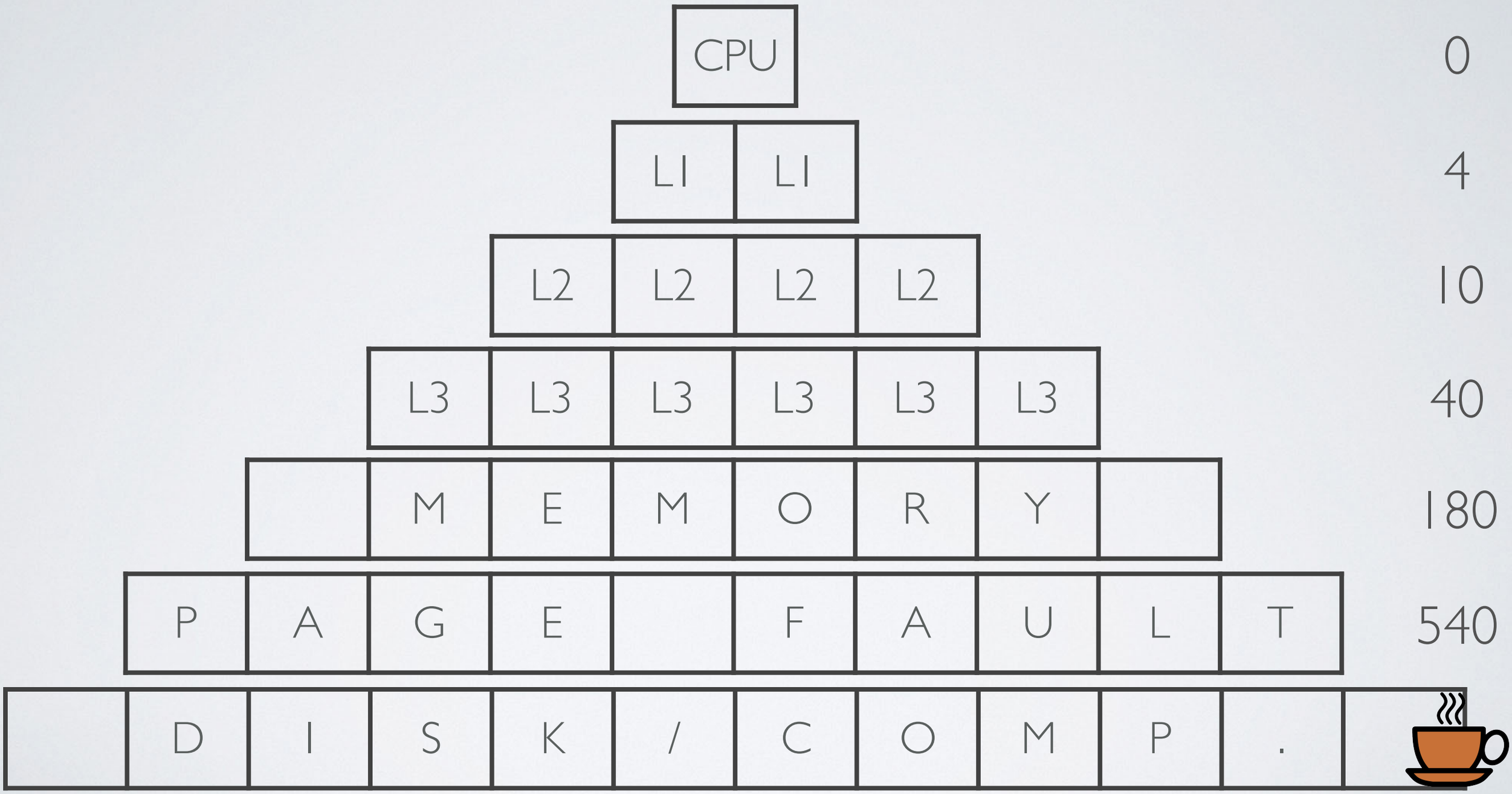
FREE LISTS



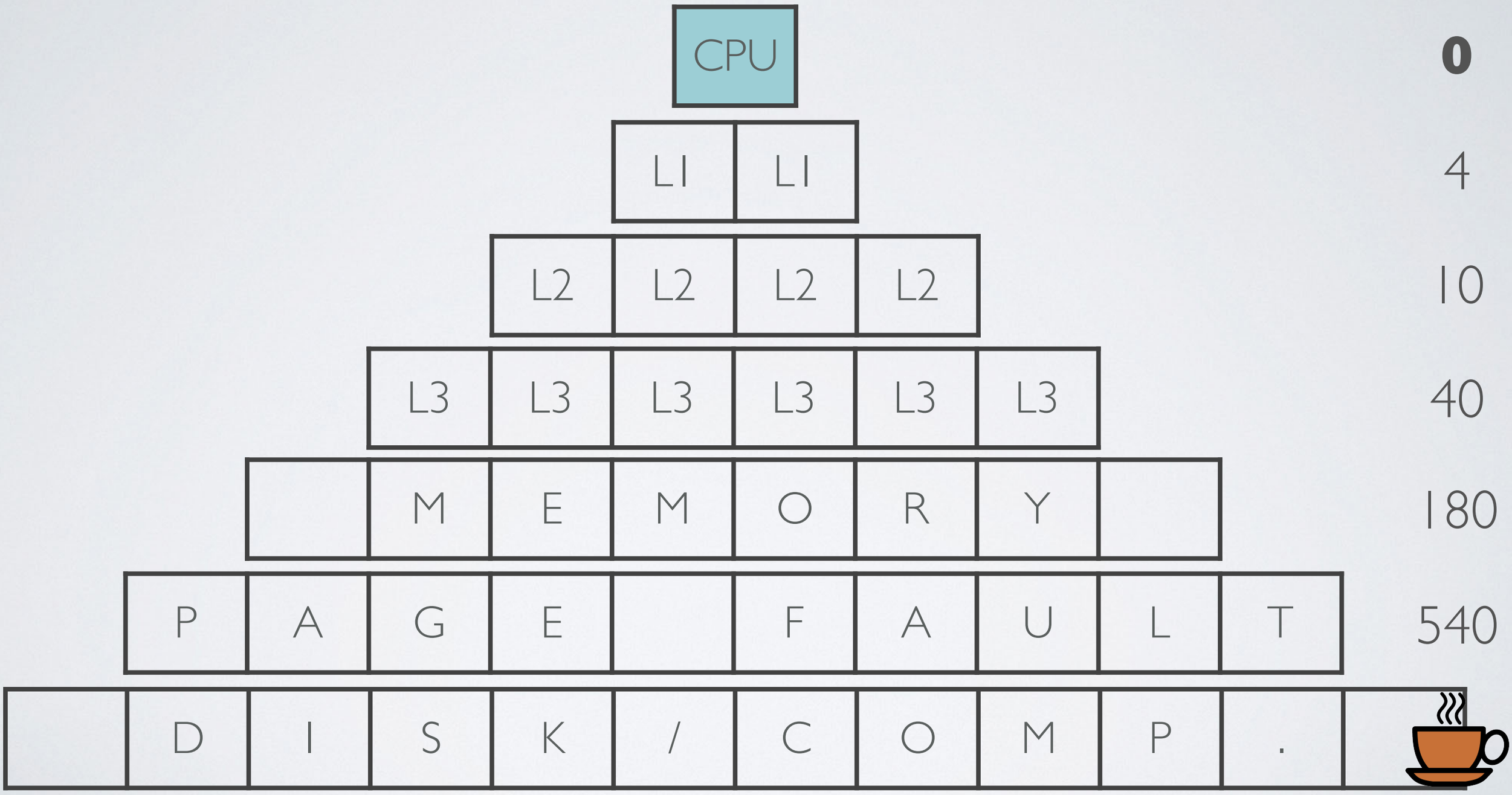
FREE LISTS



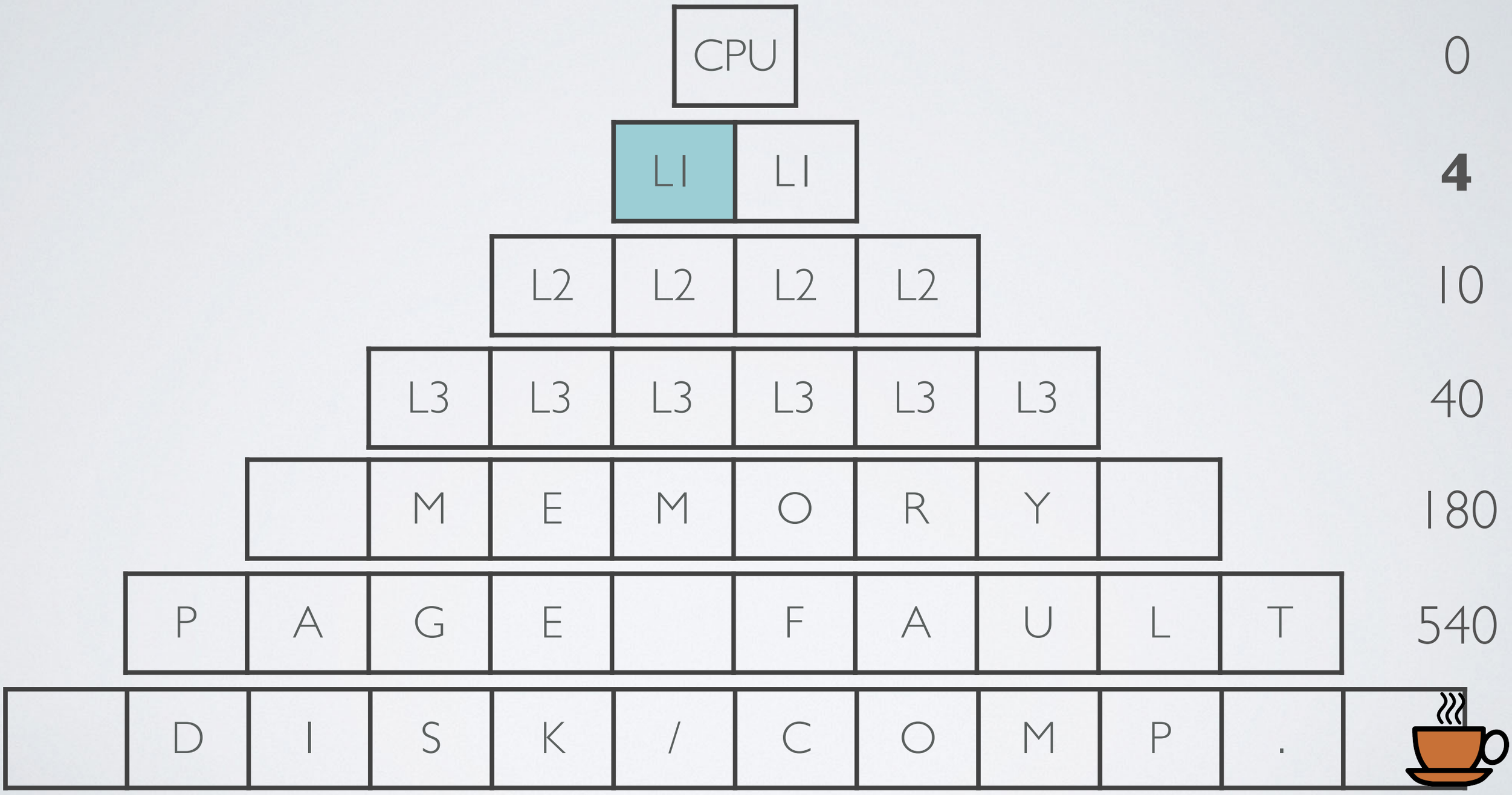
MEMORY MOUNTAIN



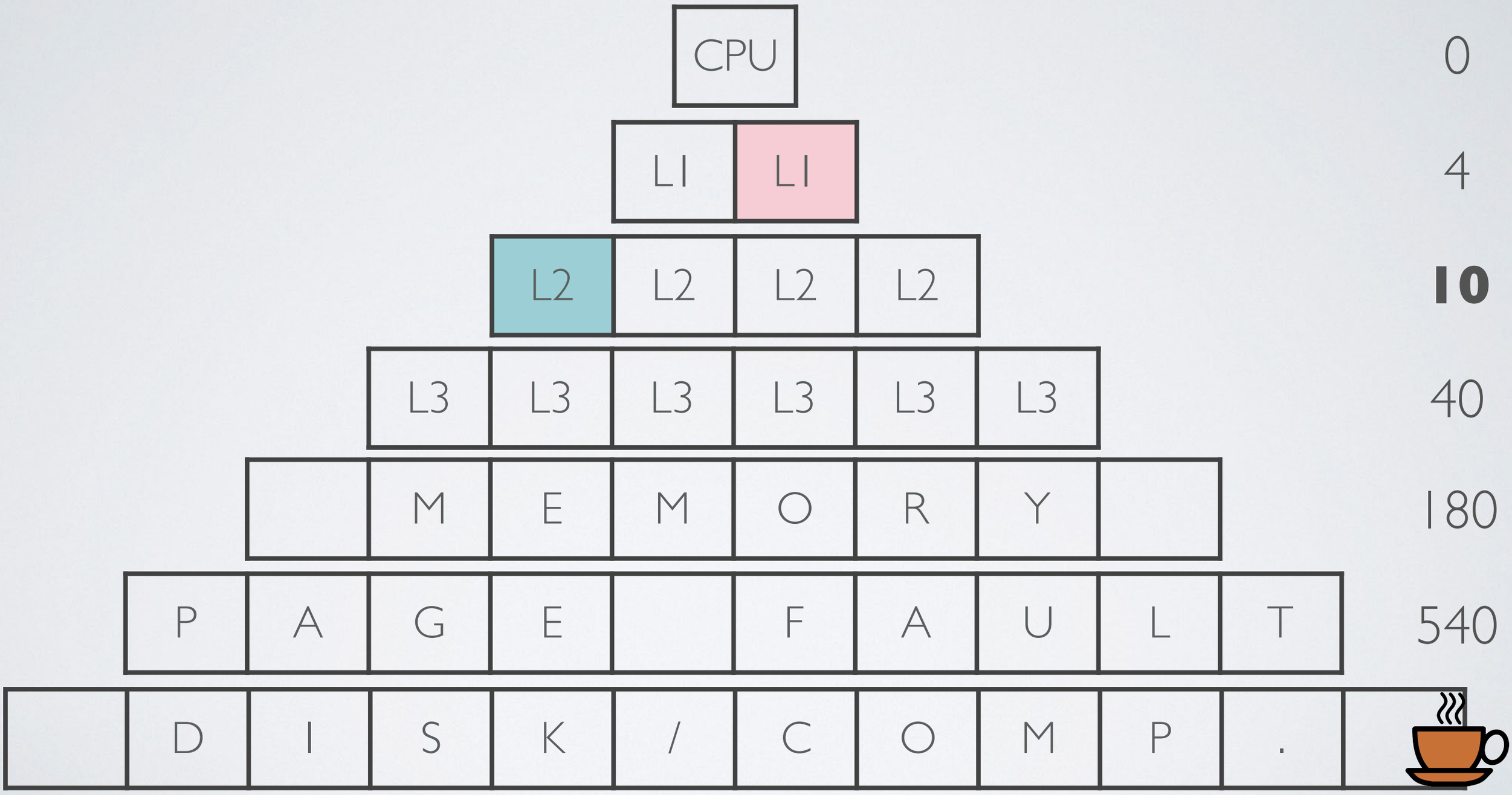
MEMORY MOUNTAIN



MEMORY MOUNTAIN



MEMORY MOUNTAIN



0

4

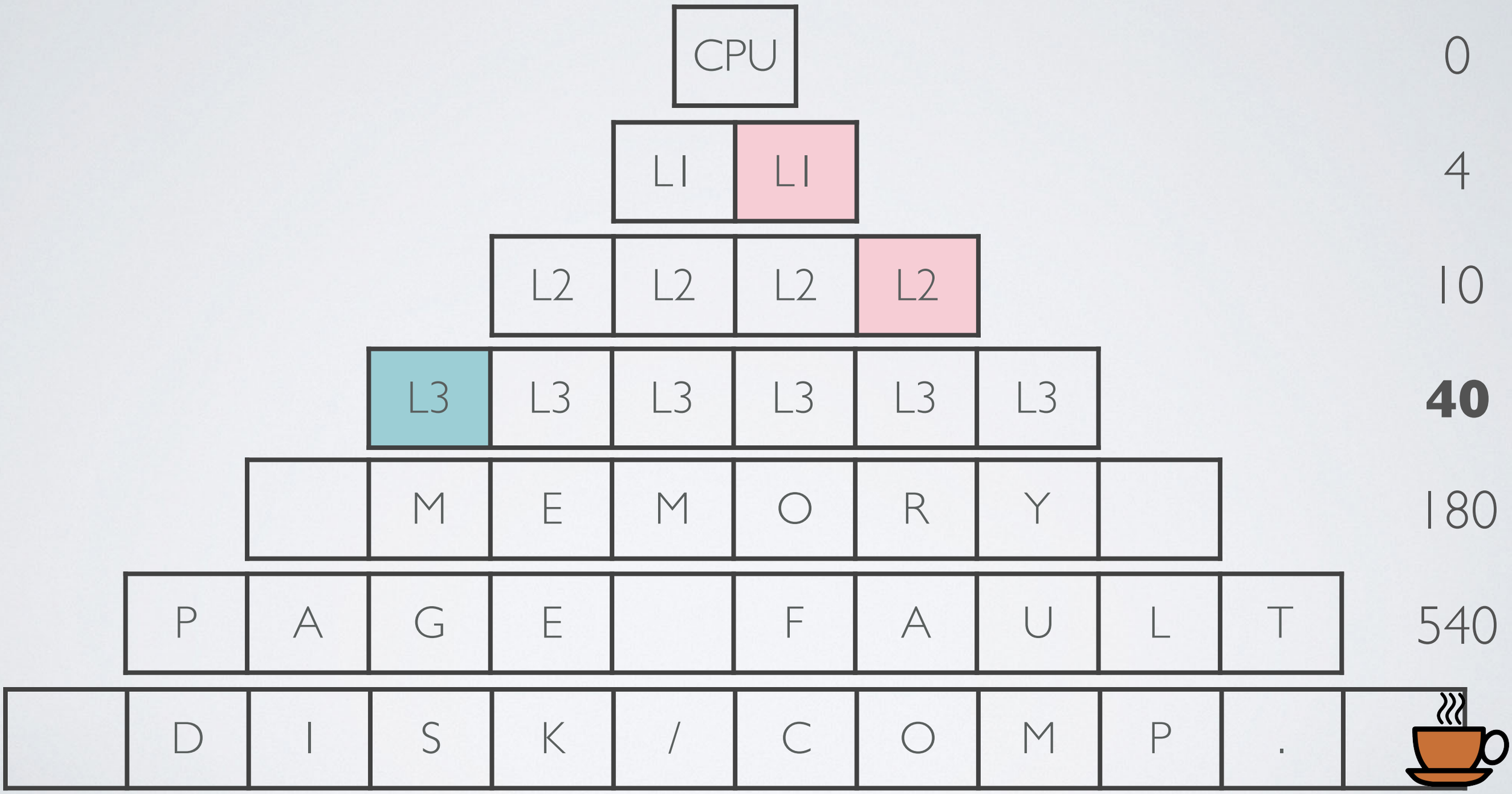
10

40

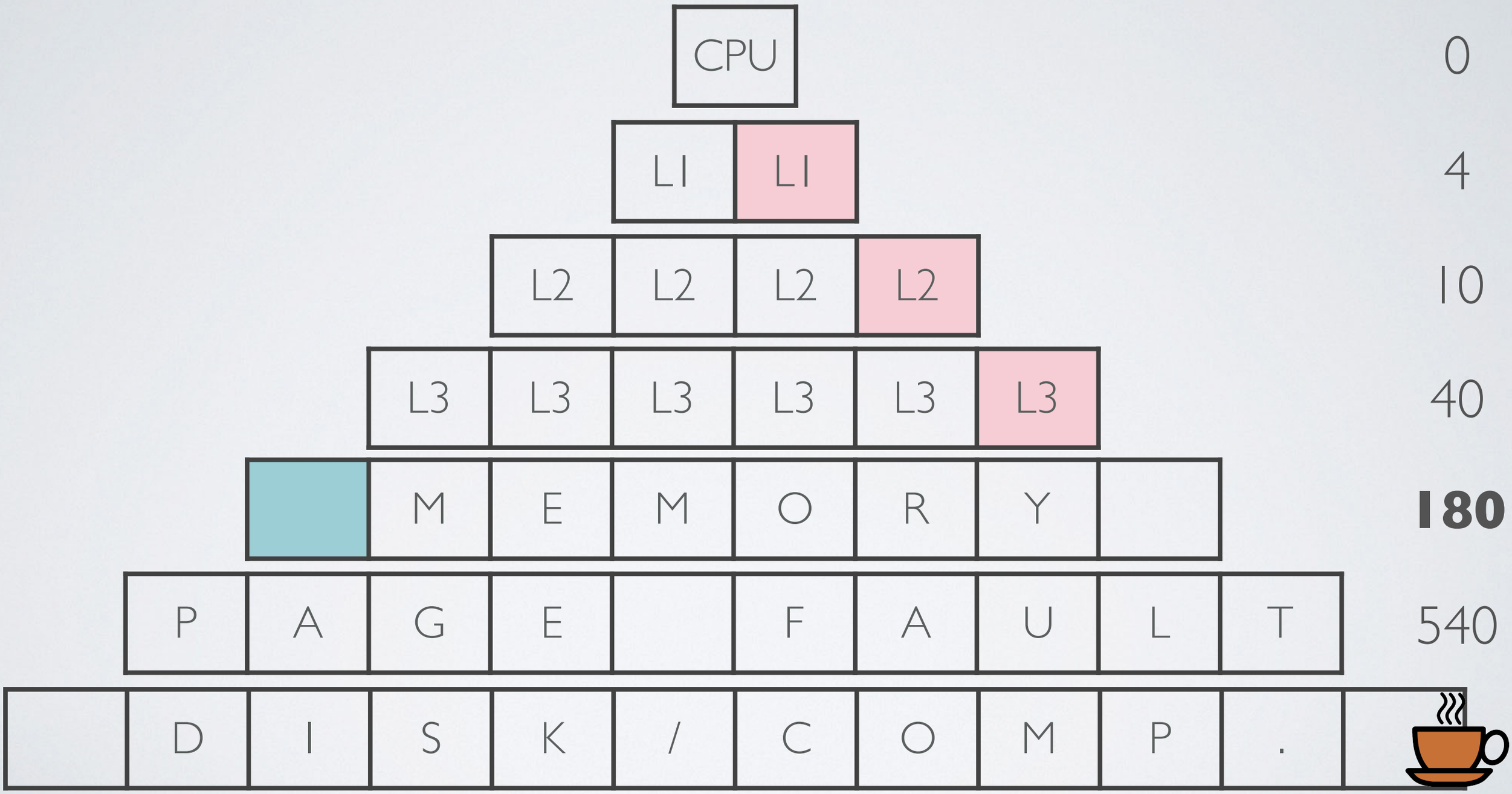
180

540

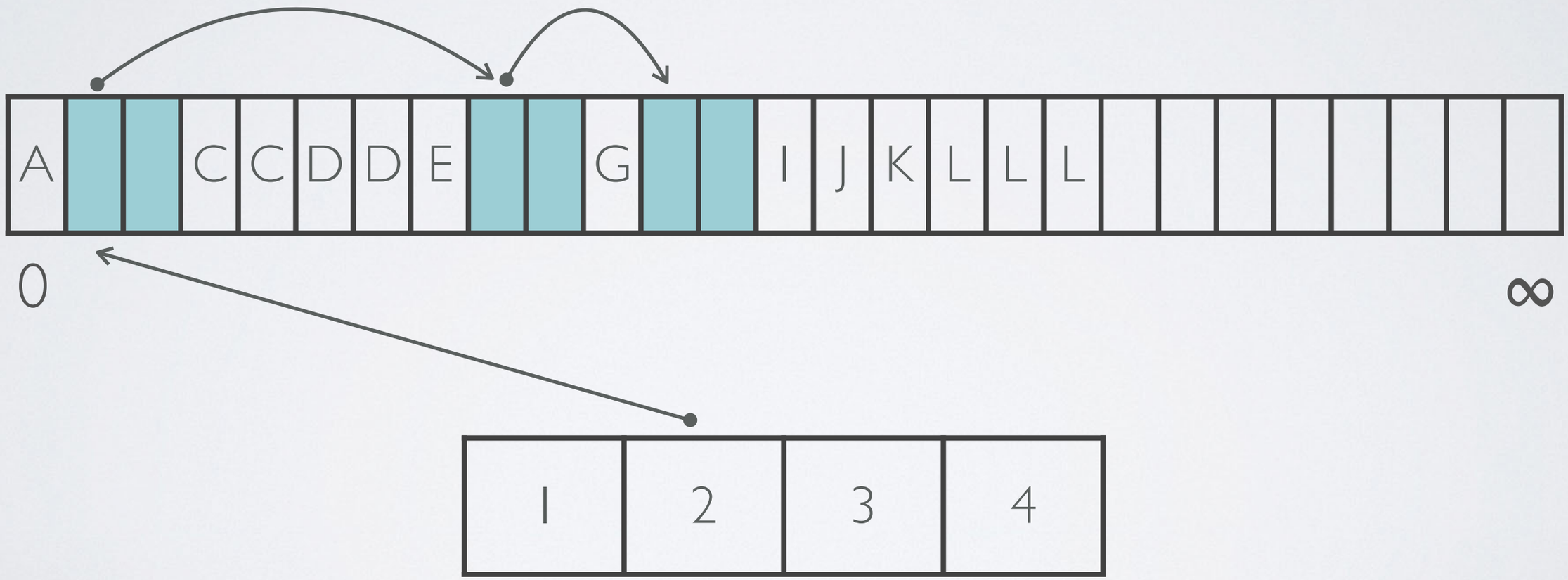
MEMORY MOUNTAIN



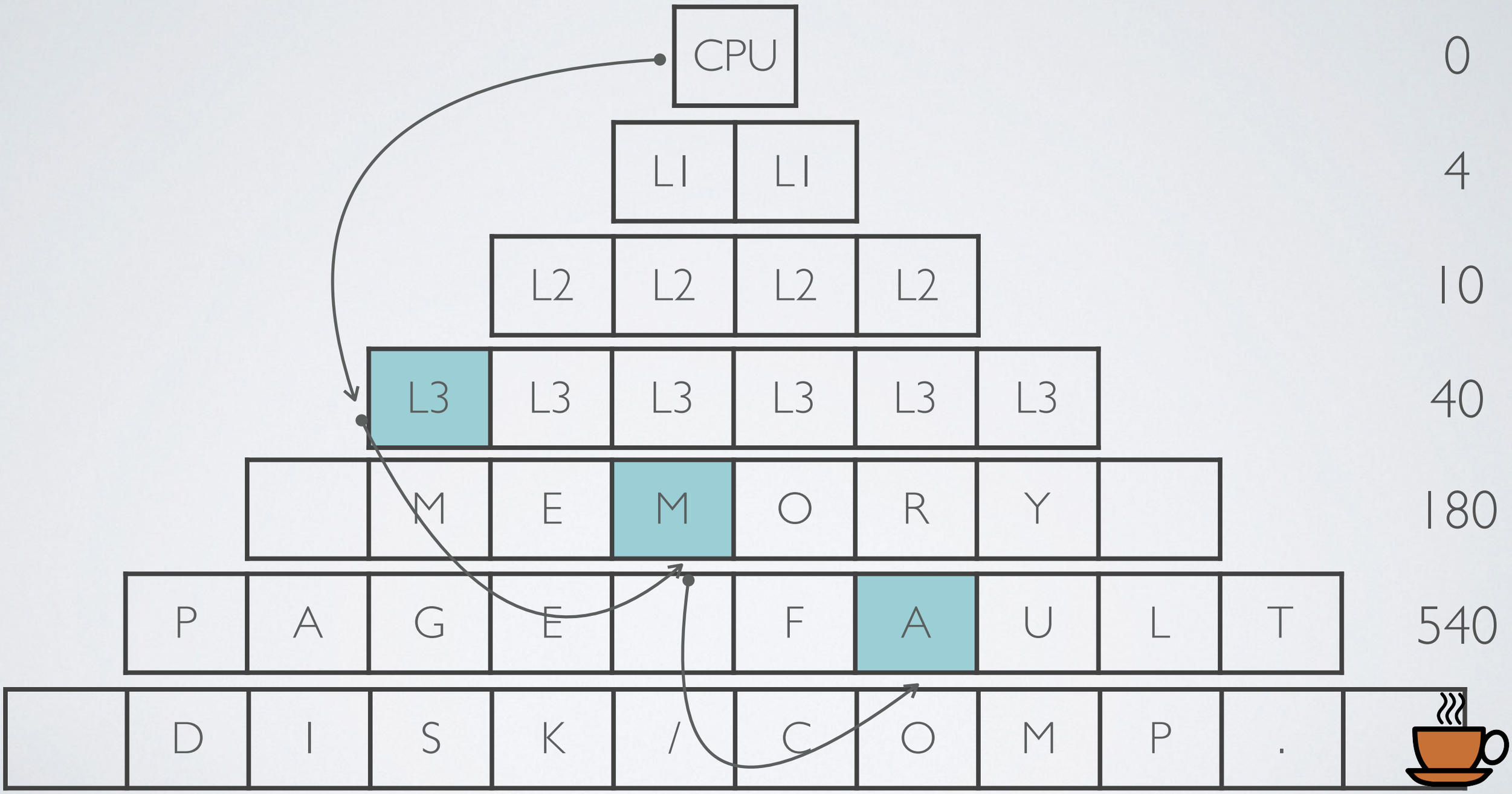
MEMORY MOUNTAIN



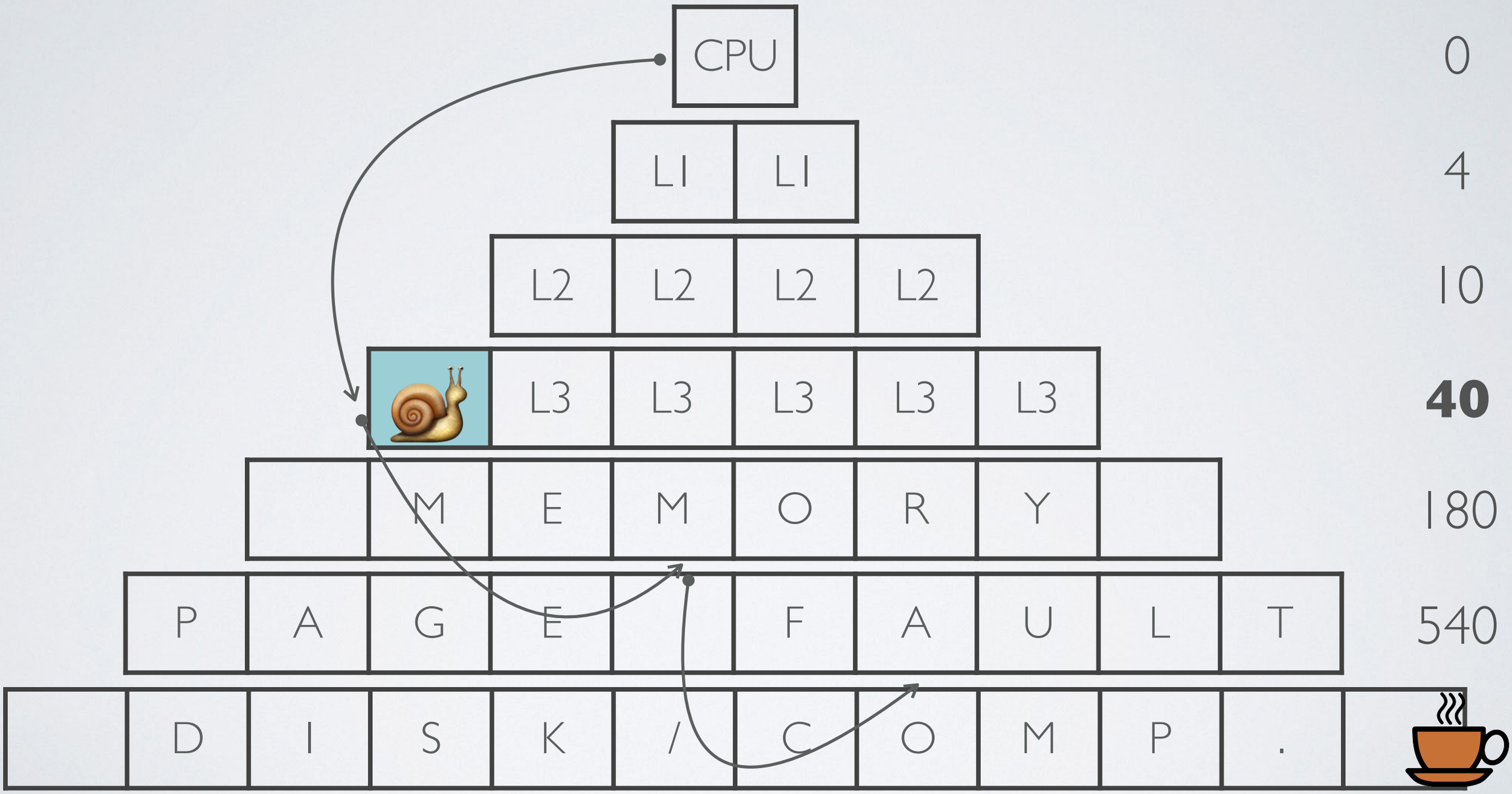
FREE LISTS



FREE LISTS


CPU
L1
L1
L2
L2
L2
L2
L3
L3
L3
L3
L3
L3
M
E
M
O
R
Y
P
A
G
E
F
A
U
L
T
D
I
S
K
/
C
O
M
P
.
0
4
10
40
180
540

FREE LISTS



0

4

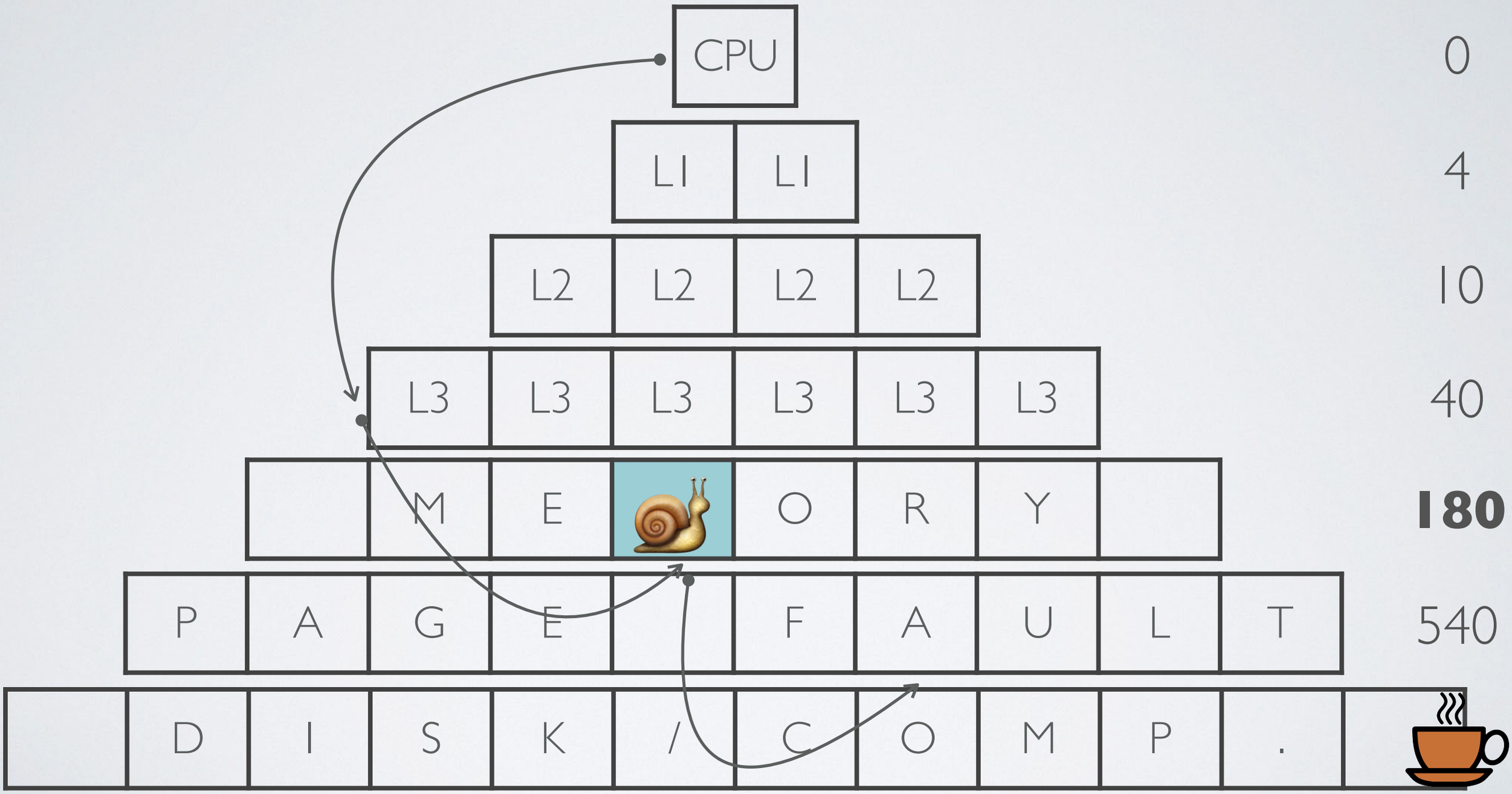
10

40

180

540

FREE LISTS



0

4

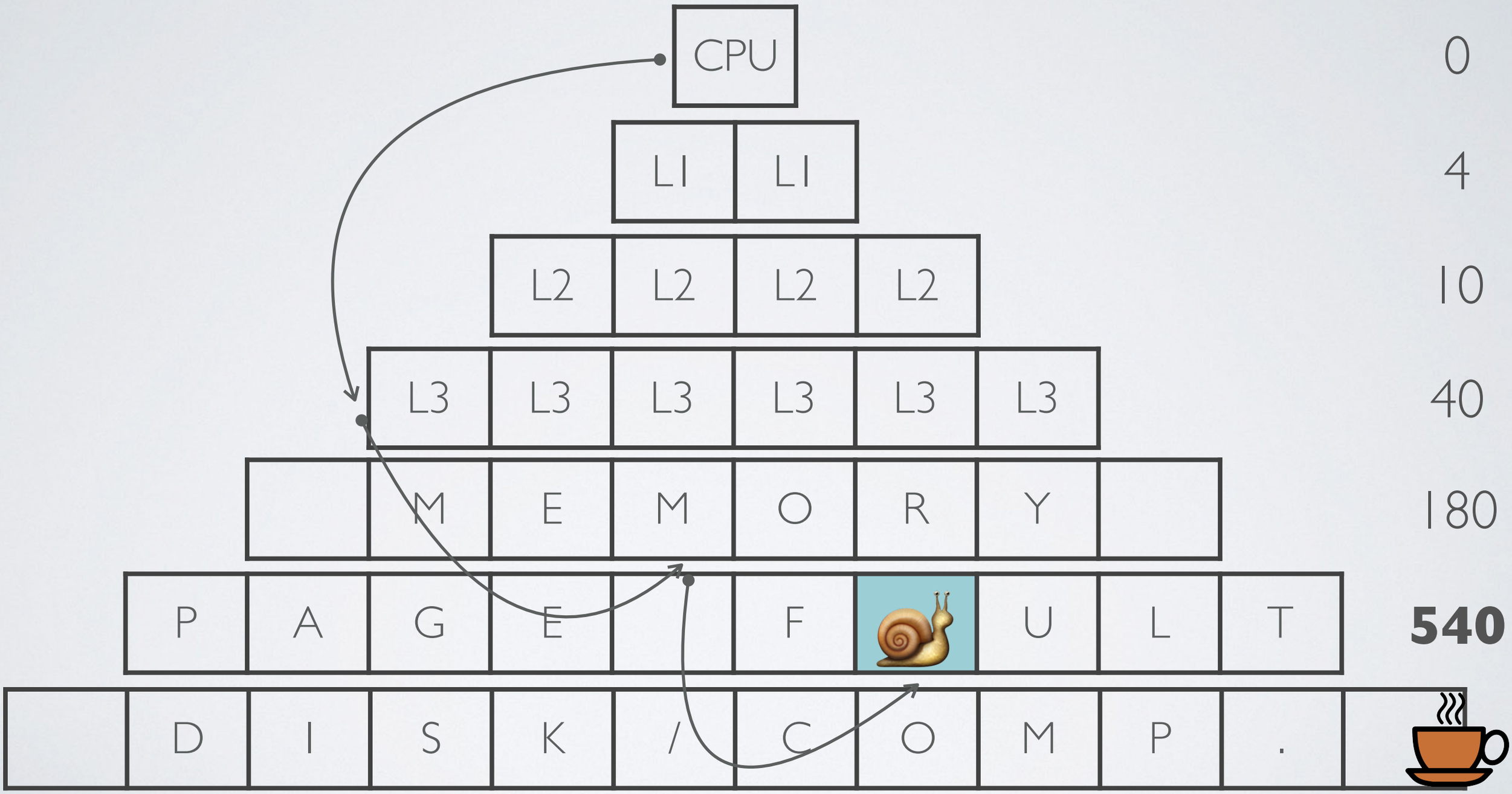
10

40

180

540

FREE LISTS



0

4

10

40

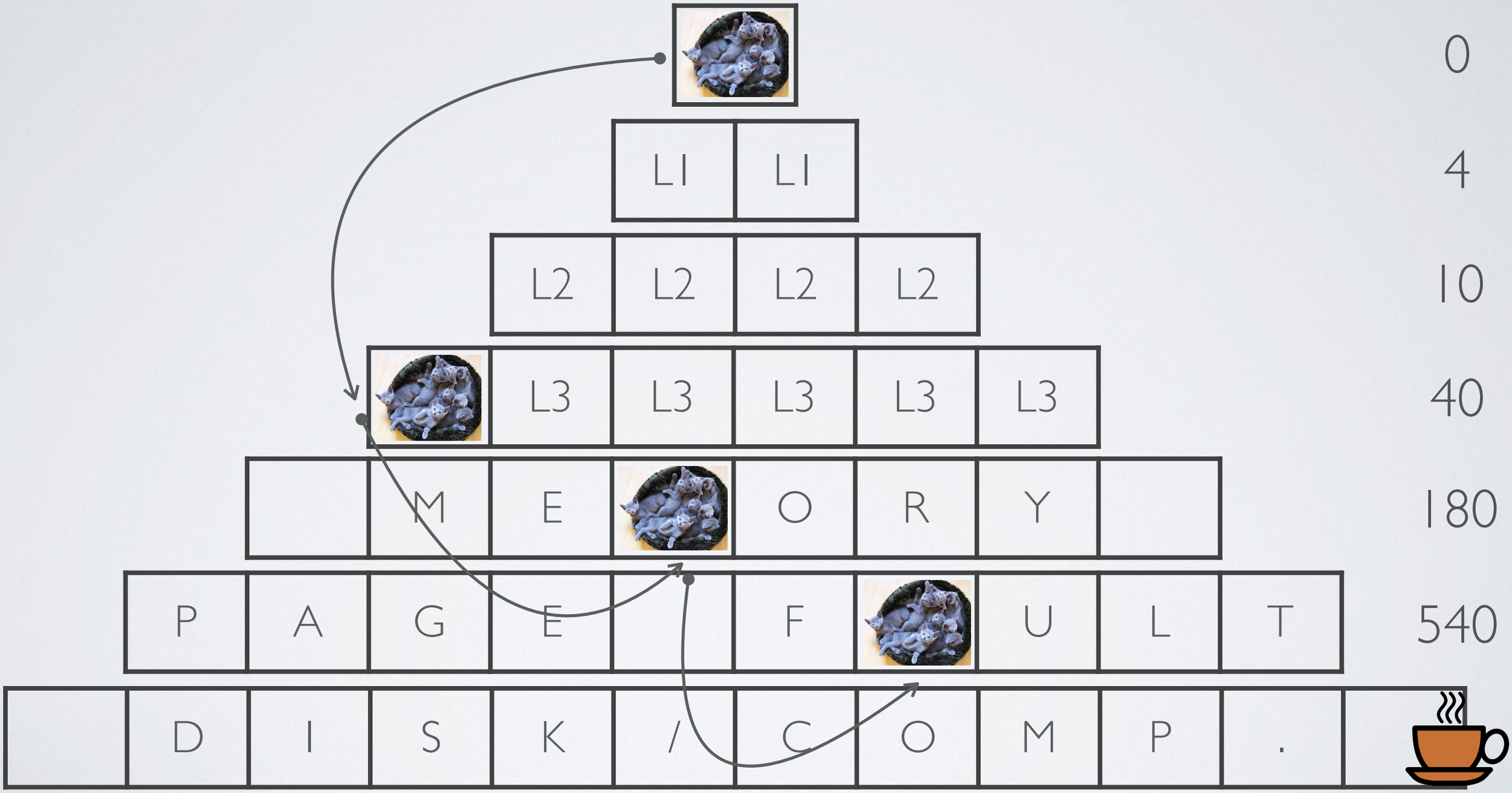
180

540

GARBAGE COLLECTION

GARBAGE COLLECTION

GARBAGE COLLECTION



GARBAGE COLLECTION

BMALLOC

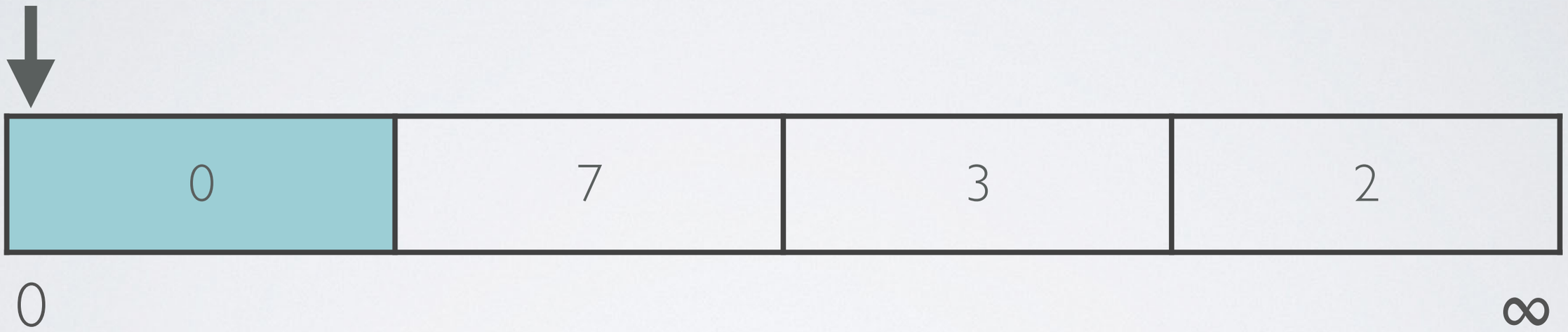
BMALLOC

BMALLOC

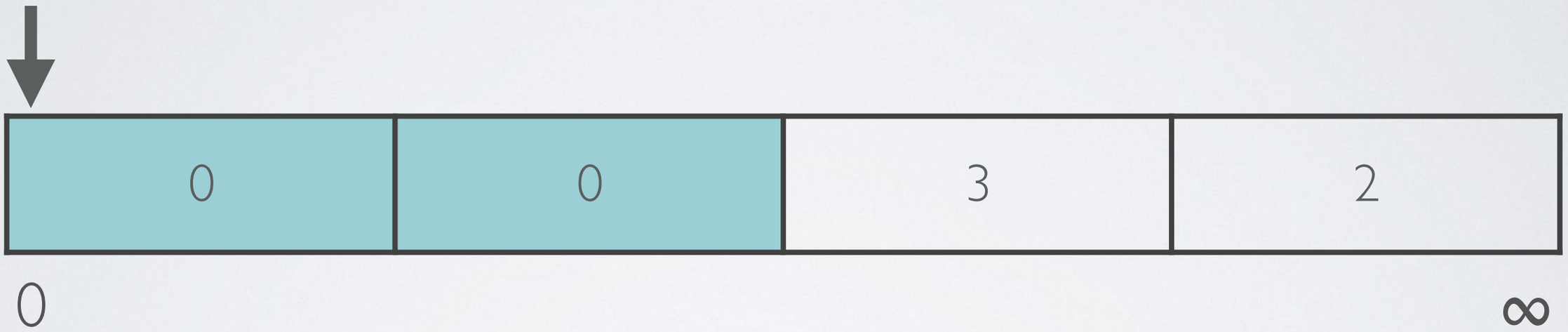
LINES

LINES

LINES



LINES

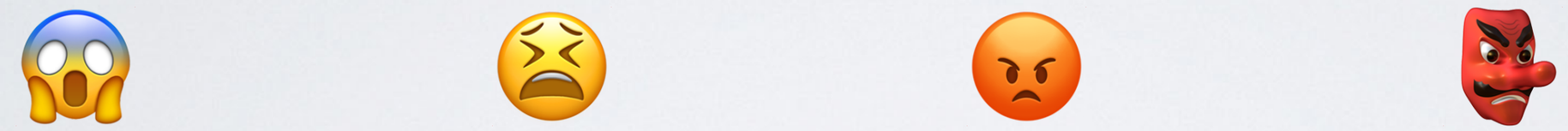


“But what about the crumbs?!”

–Everyone ever

CRUMBS

CRUMMY CRUMBS



0

∞

DON’T FEAR THE CRUMBS

DON'T FEAR THE CRUMBS

- Most objects die young

DON'T FEAR THE CRUMBS

- Most objects die young
- Objects allocated together die together

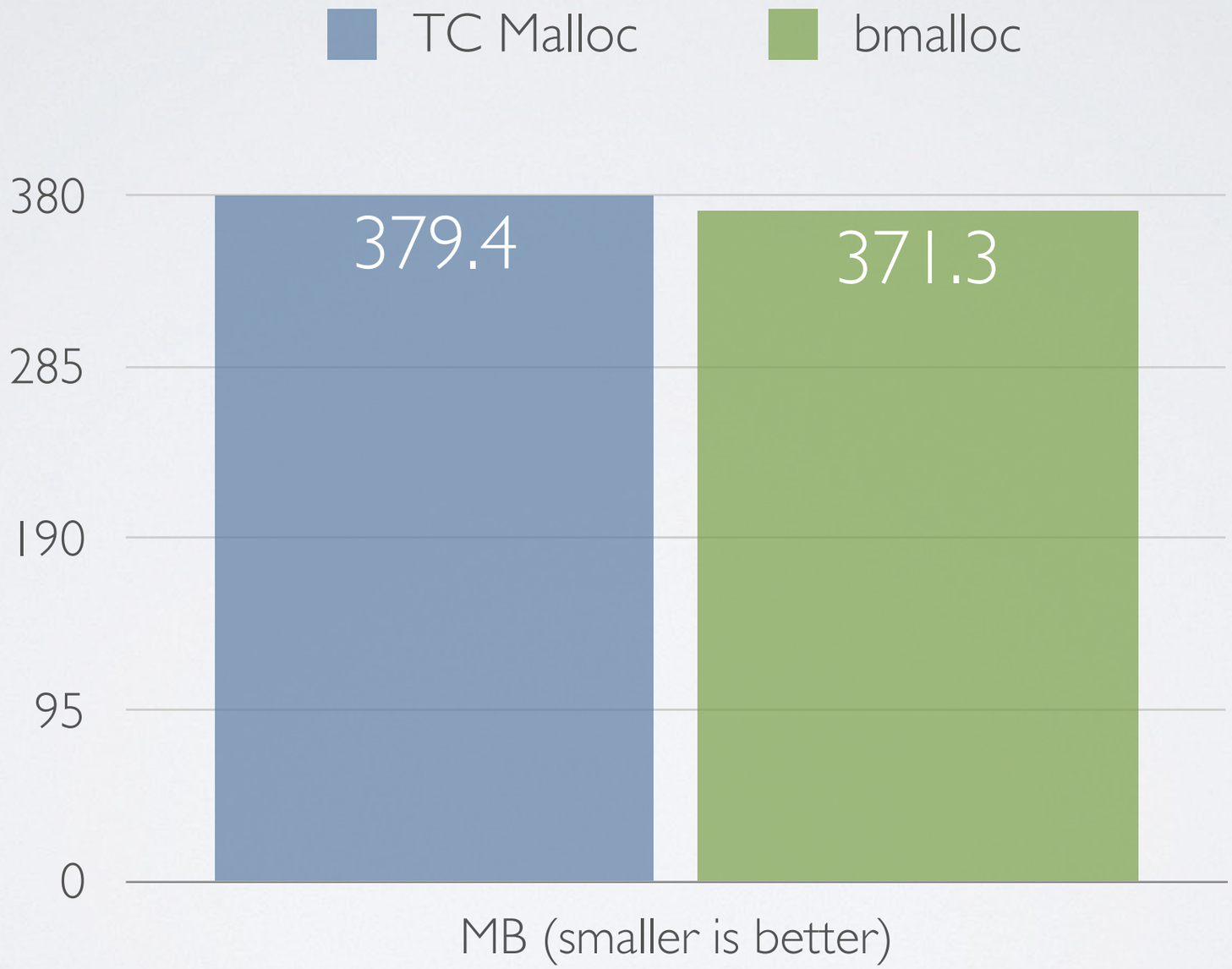
DON'T FEAR THE CRUMBS

- Most objects die young
- Objects allocated together die together
- Objects of the same type die together

DON'T FEAR THE CRUMBS

- Most objects die young
- Objects allocated together die together
- Objects of the same type die together
- Large objects matter most

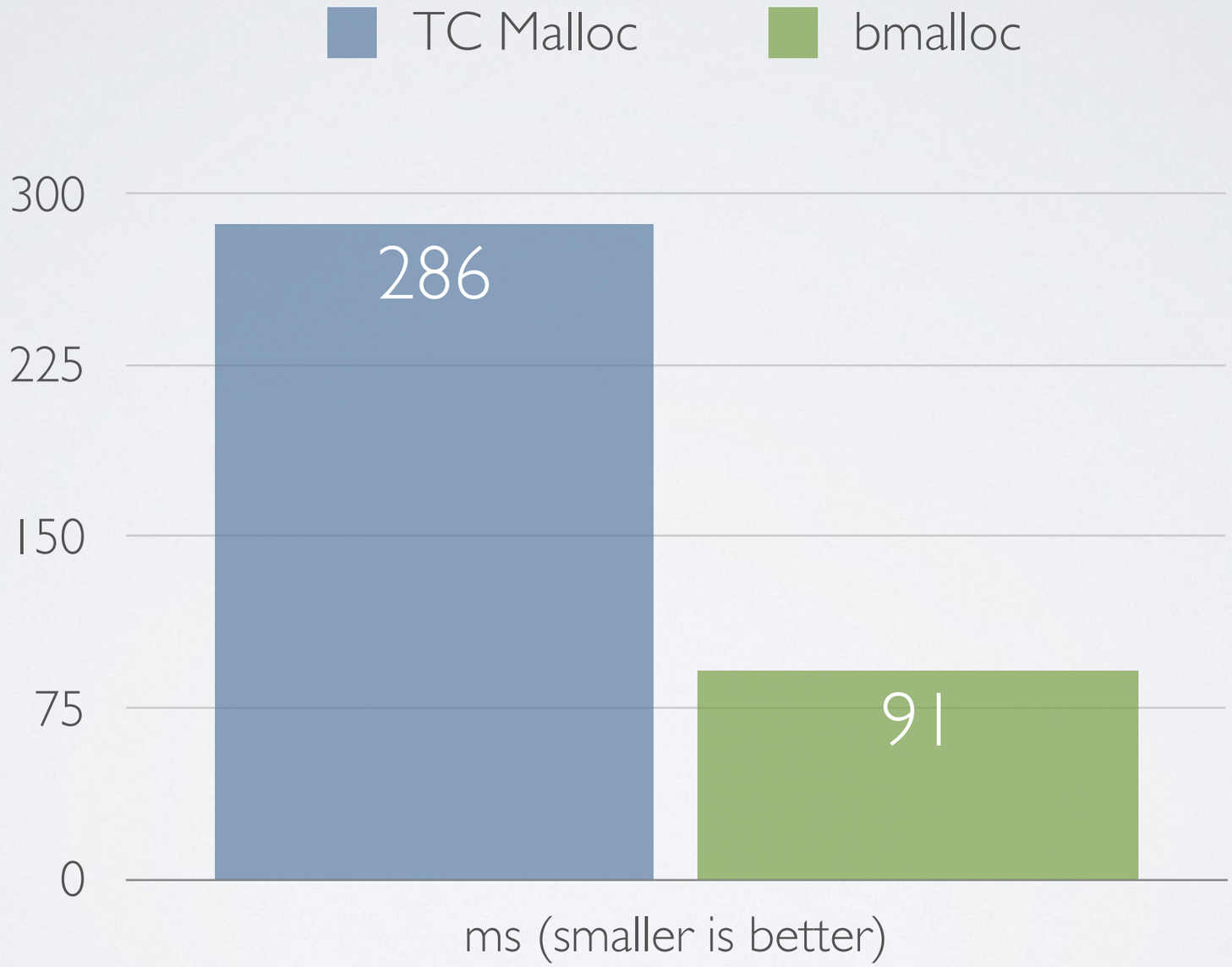
MEMBUSTER "POST"



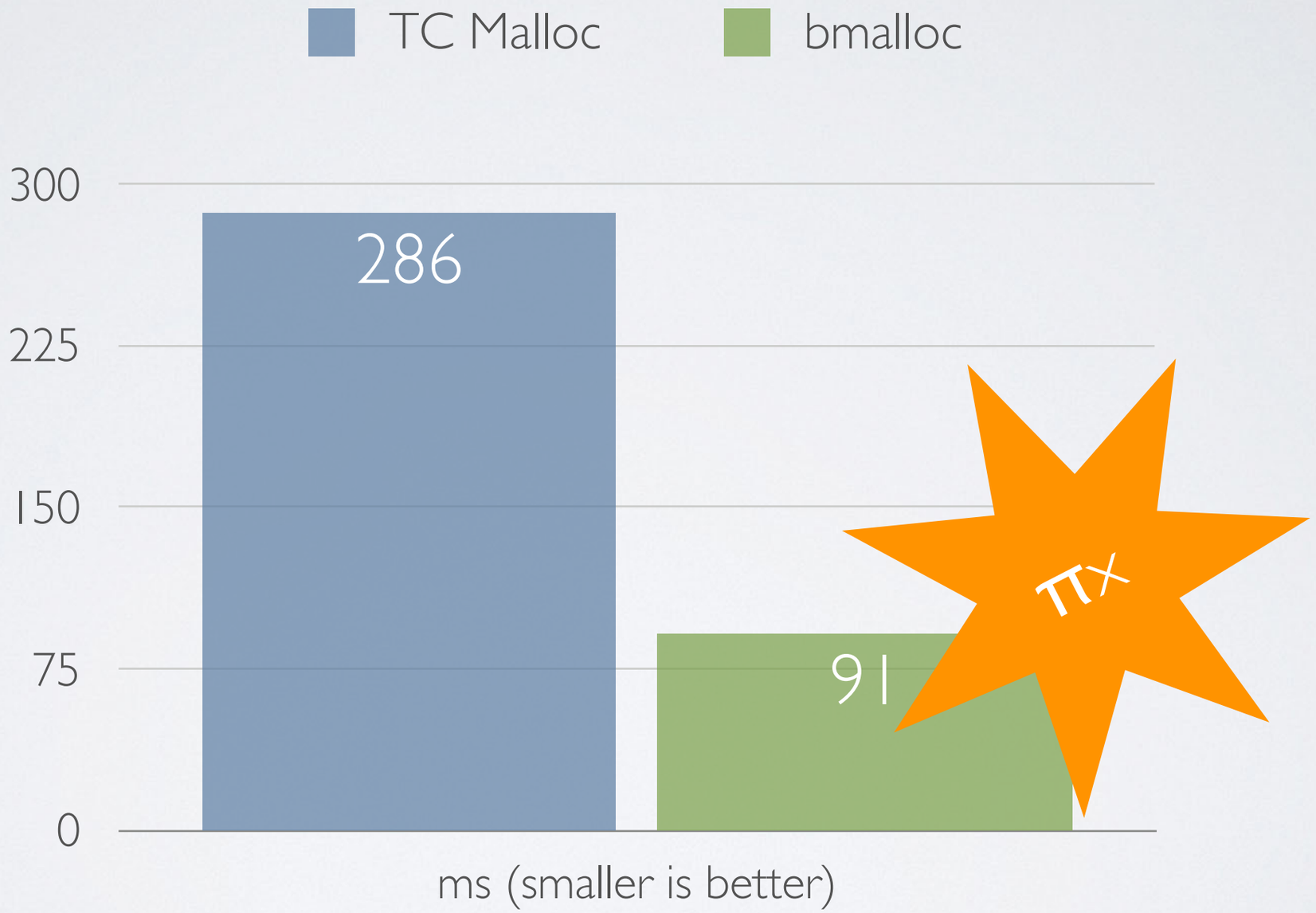
"But what about the **crumbs**?!"

–andersca

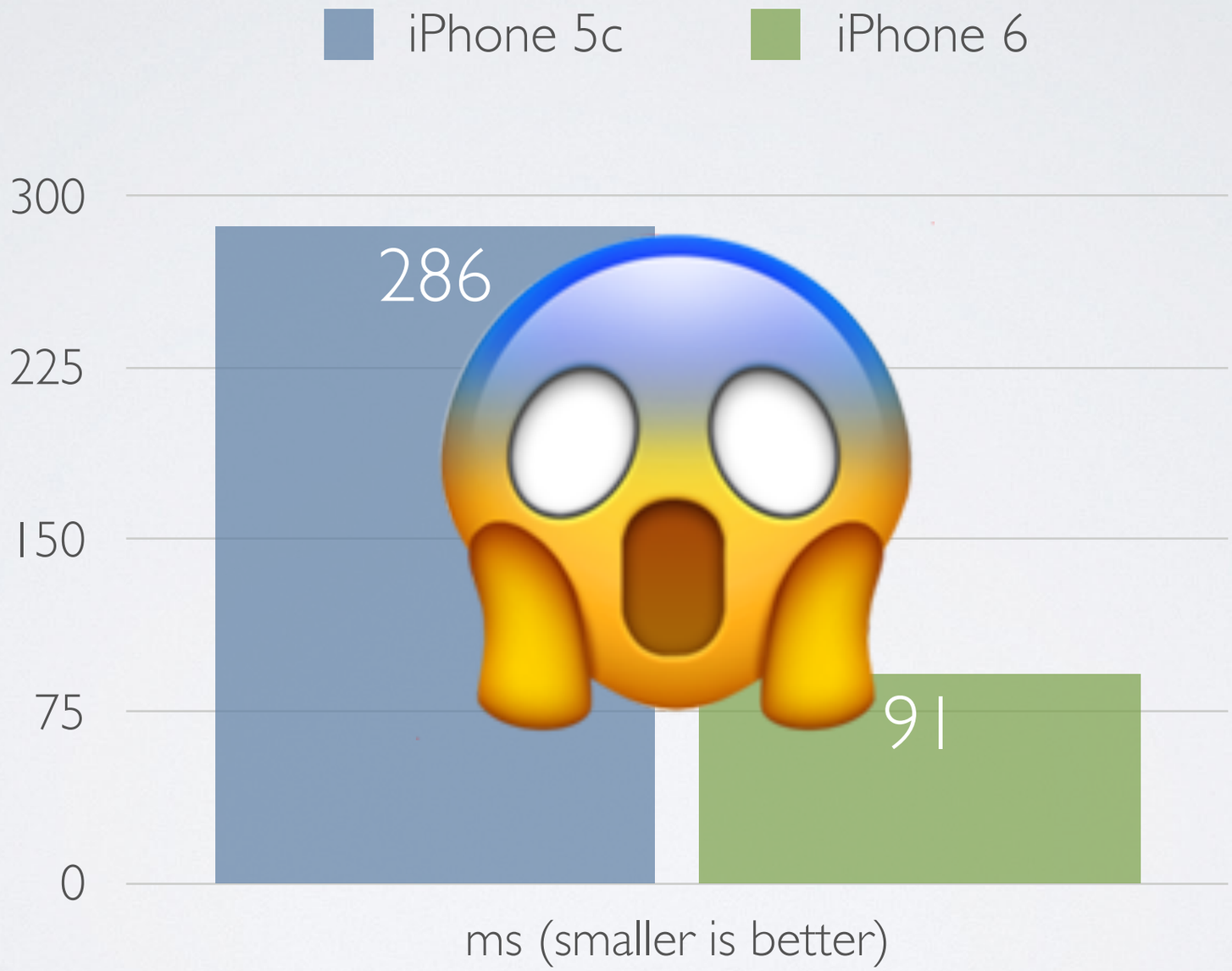
FRAGMENT_ITERATE



FRAGMENT_ITERATE



FRAGMENT_ITERATE



20 MALLOC BENCHMARKS

- churn
- list_allocate
- tree_allocate
- tree_churn
- fragment_iterate
- medium
- big
- facebook
- reddit
- flickr
- theverge
- message_one
- message_many
- churn (parallel)
- list_allocate (parallel)
- tree_allocate (parallel)
- tree_churn (parallel)
- fragment_iterate (parallel)
- medium (parallel)
- big (parallel)

20 MALLOC BENCHMARKS

Category	Speedup
reddit	1.65x
Average	2.11x
Max (list_allocate)	4.09x

30 WEBKIT BENCHMARKS

- DOM
- Page load
- JS frameworks
- CSS
- Layout
- Animation
- Parsing
- Window resize

30 WEBKIT BENCHMARKS

Category	Speedup
PLT	1.05x
Average	1.10x
Max (GetElement)	1.43x

DIDN’T WORK

DIDN'T WORK

- Threads

DIDN'T WORK

- Threads
- Atomics

DIDN'T WORK

- Threads
- Atomics
- Regions

DIDN'T WORK

- Threads
- Atomics
- Regions
- Bitvectors

DIDN'T WORK

- Threads
- Atomics
- Regions
- Bitvectors
- Trees

DIDN'T WORK

- Threads
- Atomics
- Regions
- Bitvectors
- Trees
- Sorting

DIDN'T WORK

- Threads
- Atomics
- Regions
- Bitvectors
- Trees
- Sorting
- Hashing

NEXT STEPS

NEXT STEPS

- Remove all the allocators!

NEXT STEPS

- Remove all the allocators!
- System tools

NEXT STEPS

- Remove all the allocators!
- System tools
- Moar clients

NEXT STEPS

- Remove all the allocators!
- System tools
- Moar clients
- Poor locality is a fixable bug

Agenda

- ~~Introduction~~
- ~~JavaScriptCore~~
- ~~Efficient Mark-Sweep~~

  - *30 minute break*

- ~~Concurrent GC~~
- ~~bmalloc~~
- WTF::Lock