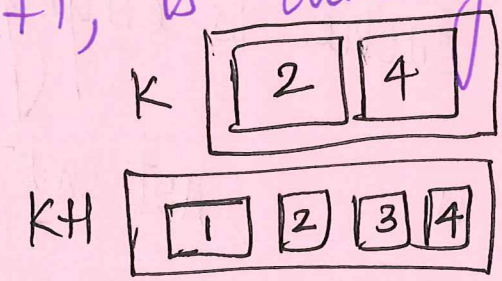
Lecture 24 - Cache Memories

Idea 1:

Every storage device is a "cache" to the (at level K) slower and larger storage device (at level K+1).

Cache Hits

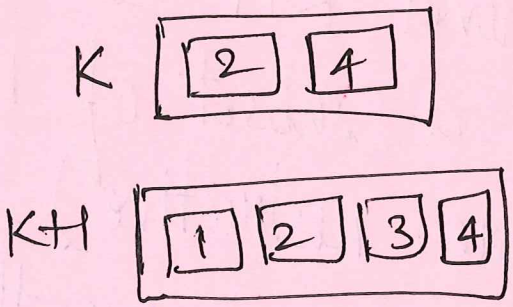
The data object d, that the program needs from level K+1, is already available at level K.



read block 2
⇒ cache hit.

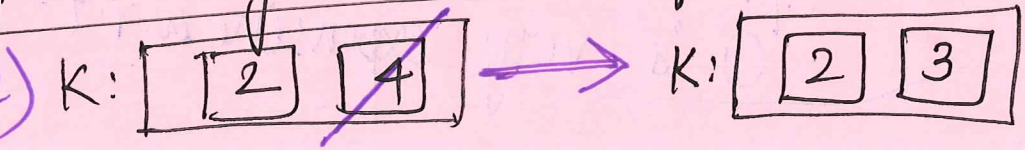
Cache Misses

if the data object d is not cached at level K
⇒ cache miss.



Read block 3 ⇒ cache miss.

State after reading block 3 from K+1



Important terminology

1. replacing / evicting a block
2. victim block
3. replacement policy — eg. random, LRU, etc.
4. placement policy — where to place the block in the cache?

Kinds of Cache Misses

1. Compulsary misses (or) cold misses.

if the cache is empty ⇒ cold cache.

not empty ⇒ warm cache.

Placement policy

1. Any block from level K+1 can be stored in any block at level K. (most flexible)
2. A particular block at level K+1 can be placed/stored only in a subset of blocks at level K. (more restrictive)

H/W caches (L1, L2, L3) use this since ① is very expensive w.r.t. searching a block.

Kind of Cache Misses (cont.)

2. Conflict miss

Even though the cache is large enough to hold the referenced data objects, but because they map to the same cache block, the cache keeps missing.

eg. Reading block 0 & 8 repeatedly in the ~~prev~~ fig.

page (12) Lecture 23.

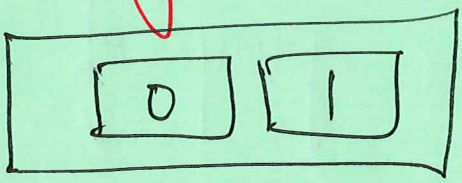
3. Capacity misses

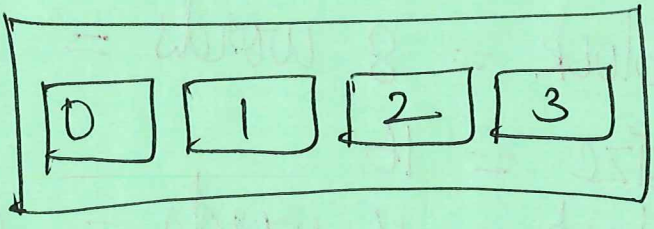
if (size of working set > cache size)

$\Rightarrow$ capacity miss.

The cache is just too small to handle this particular working set.

eg.





if working set is 0, 1, 2 in the same order again and again.

$\Rightarrow$ capacity miss.

Cache Management

Logic to manage a cache – H/W, S/W or both.

Cache type	Managed By
eg. Registers	Compiler
L1, L2, L3	Hardware Logic. (built into the caches)
DRAM Main Memory	Operating System + Addr. translation H/W on the CPU.

Blocks

On 32-bit computers,

word size = 32 bits or 4 bytes.

> Block = a group of words

eg. if block size = 8
⇒ 1 block = 8 words = 32 bytes.

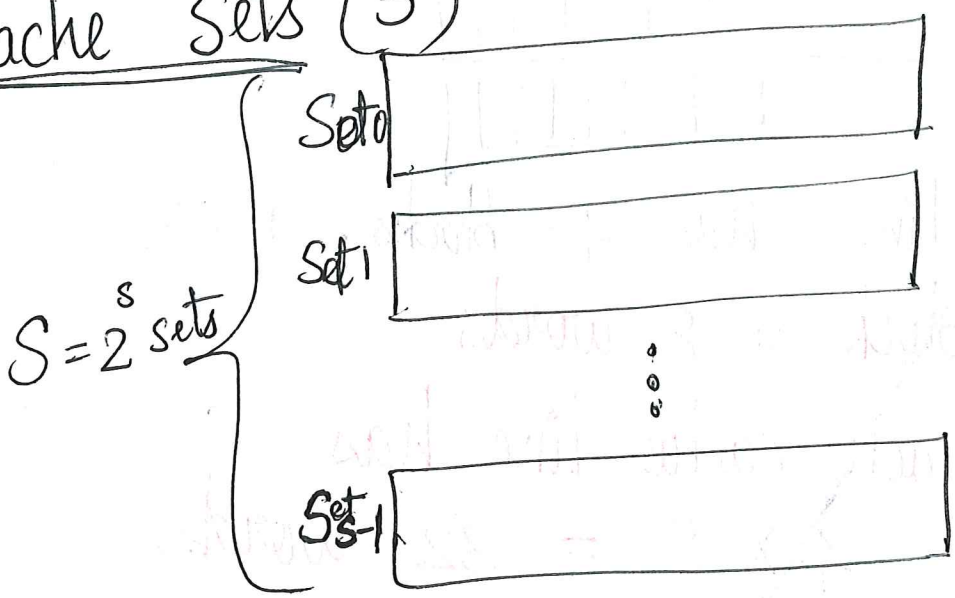
if block size = 16
⇒ 1 block = 16 words = 64 bytes.

Cache Memories

each memory address = m bits

unique addresses $M = 2^m$

Cache Sets (S)

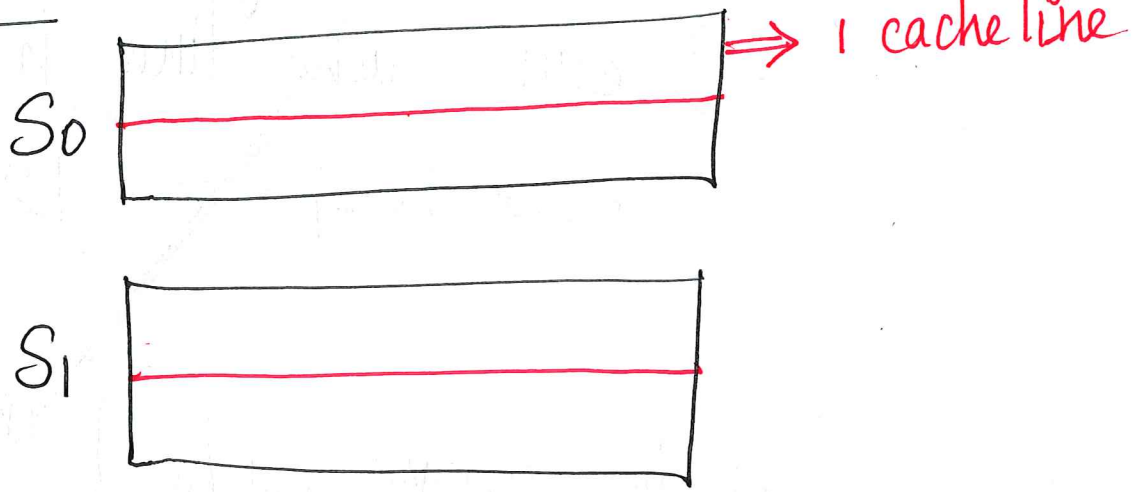


no. of cache sets

$S = 2^s$

~~where~~

Cache Lines (E)

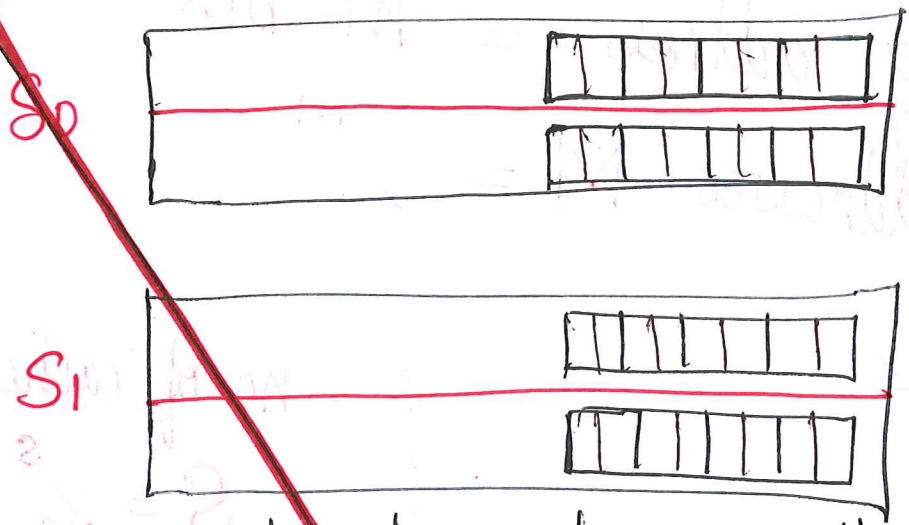


Each Set has 2 cache lines in the above example.

(ie) $E = 2$

Total cache lines in the above cache = 4

Cache block



Each cache line has ~~4 blocks.~~ 1 block.

if 1 block = 8 words

⇒ each cache line has 8 words

~~4 × 8 = 32 words~~

if 1 word = 4 bytes

⇒ each cache line has,

4 × 8 = 32 bytes

~~32 × 4 = 128 bytes~~

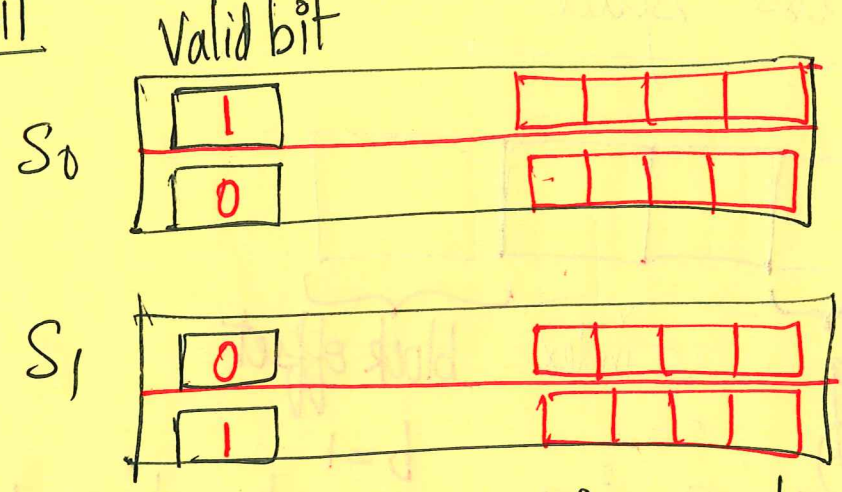
of memory per cache line.

⇒ Size of the above cache

= ~~128~~ 32 bytes × 4 cache line

= ~~512~~ 128 bytes

Valid bit



if valid bit = 1 $\Rightarrow$ The cache line contains meaningful information.

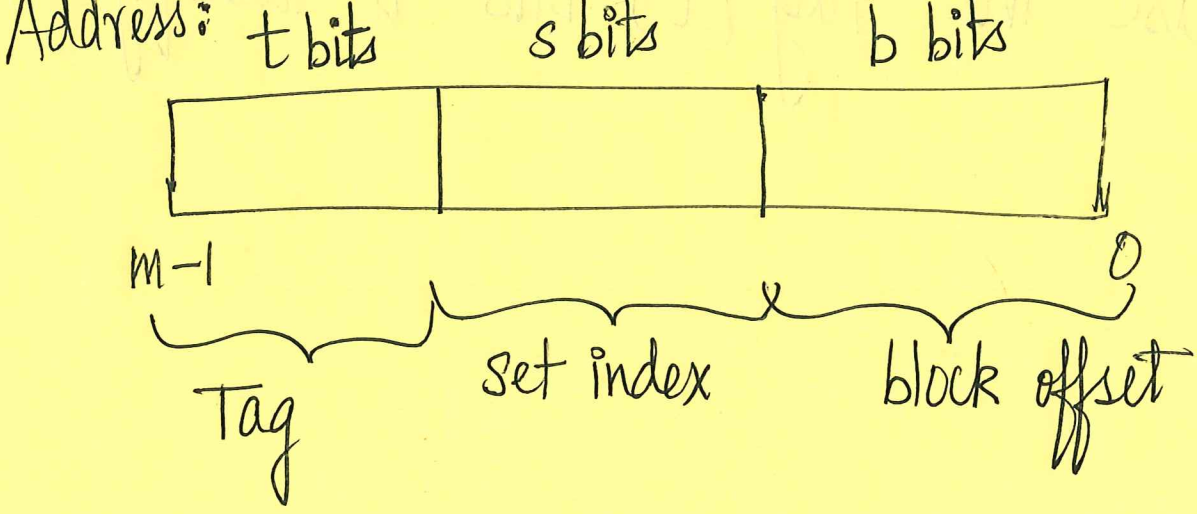
if valid bit = 0 $\Rightarrow$ either cold cache
(or)
cache-entry invalidated.

Tag bit

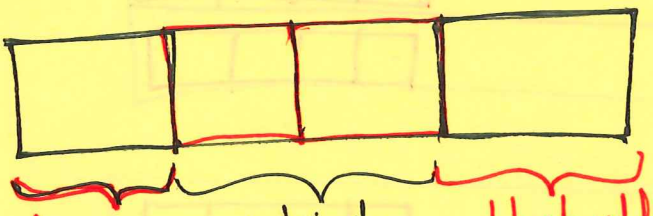
$$t = m - (b+s)$$

t - no. of. tag bits.

To uniquely identify the block stored in the cache line



4-bit address space



tag (1-bit). set index: $s = 2$, $S = 2^2 = 4$ ⇒ 4 sets. block offset: $b = 1$, $B = 2^b = 2^1 = 2$ bytes ⇒ 1 block = 2 bytes.

To find a word in a cache memory

1. Find the set in which the word can be.
   use the set index bits to find this.
2. Find the cache line (if any) within the set which contains the word.
   use the tag (t) bits to identify this.

A line in the set contains the word

1. iff the valid bit = 1.

and

2. The tag bits in the line match the tag bits in the address A.

Direct Mapped Cache

No. of cache lines per set = 1.

ie. $E = 1$

eg. $(S, E, B, m)^{(10)} = (4, 1, 2, 4)$

Cache has

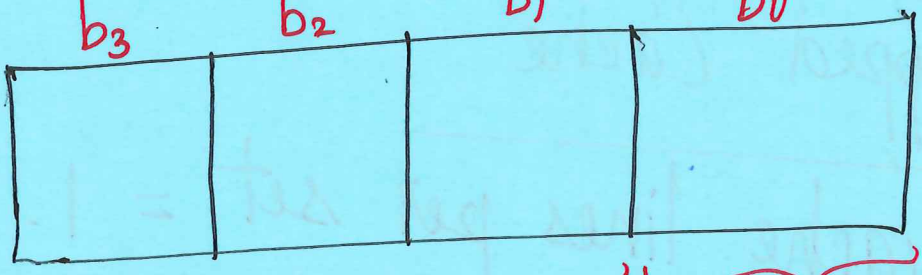
4 sets

1 line per set

2 bytes per block

4-bit addresses.

4-bit Address:



Tag | set index | Block offset

Assumption: 1 word = 1 byte

Caches in action

* Enumerate the 4-bit address space

Address (decimal)	Tag bits (t=1)	Index bits (s=2)	Offset bits (b=1)	Block number (decimal)
0	0	00	0	0
1	0	00	1	0
2	0	01	0	1
3	0	01	1	1
⋮	⋮	⋮	⋮	⋮
15	1	11	1	7

set

Set	Valid	Tag (2)	block[0]	block[1]
0	0			
1	0			
2	0			
3	0			

1. Read word at address 0.
2. Read word at address 1.
3. " " " " 13.
4. " " " " 8
5. " " " " 0.