16-1

caching — uses a "cache"

cache k intermediates to cache k+1

register (k=0)

cache lines (or blocks) or transfer units ∈ cache

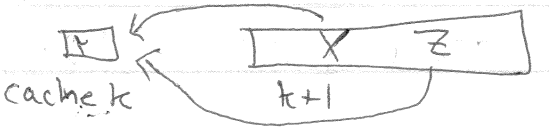
a stream (or trace) of address requests

hit — it's in cache

miss — it's not (ie forward request)

└ new request — a "cold" miss

"warm up" initial fill-up of cache



placement policy (map large stuff to small space)

└ conflict miss — you want address X, stored Y but Y contains Z

capacity (how many lines) working set (how much prog needs

└ capacity miss — wss > capacity

Locality — use information repeatedly

temporal — using a[0] now means you'll use a[0] again

spatial — using a[0] means you'll want a[1]

x = a[0] * 2 + a[0] * 3 + a[0] * a[0]

y = a[0]

movq %rax, addr

movntq %rax, addr

---

a[M][N]

```
for i = 0 ... M
  for j = 0 ... N
    sum += a[i][j]
```

→ a_{00} a_{01} a_{02} a_{10} a_{11} a_{12}

```
for j = 0 ... N
  for i = 0 ... M
```

if N > cache line size, every req is a miss

a_{00} a_{10} a_{01} a_{11} a_{20} a_{21}

6-2/ A cache is (S, E, B, m) size $= B \times E \times S$

m is the number of address bits $B = 2^b$

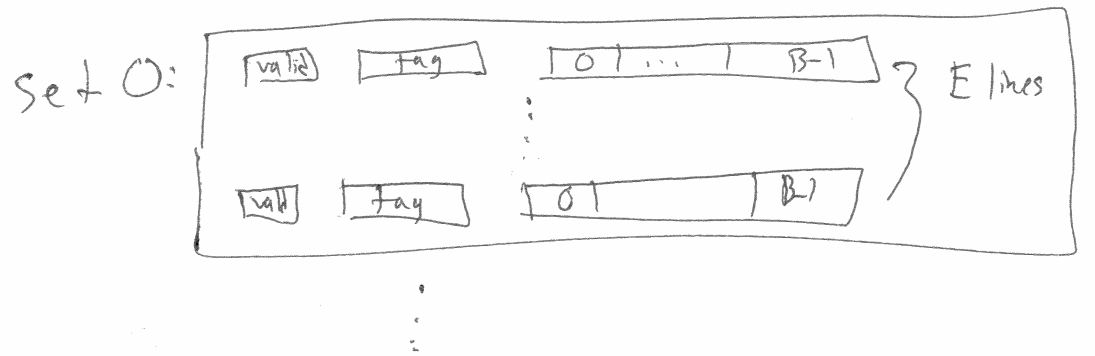
$M = 2^m$

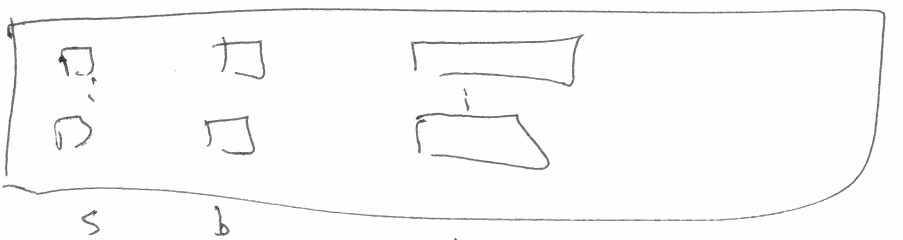
S is how many "sets" there are $(S = 2^s)$

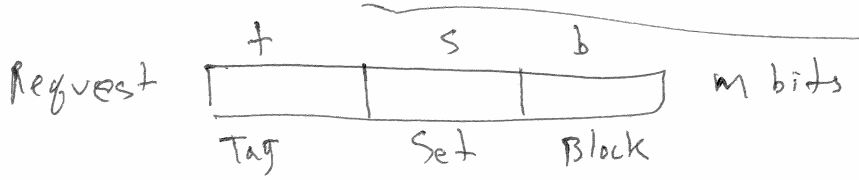
Each set has E lines

Each line has $B = 2^b$ bytes, a valid bit, and a tag

A tag is $m - (b+s)$ bits







$m = 4$ $S = 4$ $b = 1$ $E = 1$

addr = 0000 → set 0 → B_0

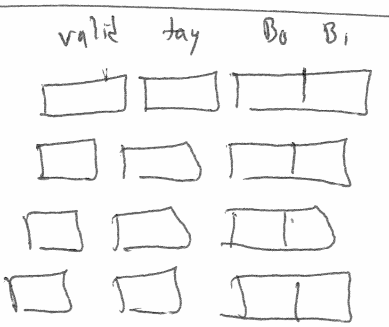
tag set block

0001 → set 0 → B_1

0010 → set 1

1000 → set 0 → B_0

1001 → set 0 → B_1



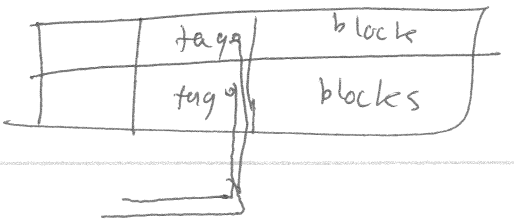
if valid is 0, then line isn't used yet

if tag matches tag bits, req is a hit

$E = 1$ – a direct-mapped cache

E > 1

Set 0:



E-way associate cache (E=4)

req tag

replacement policy during misses — LFU – frequency, LRU – recenty

S=1 but E>1 — fully associative