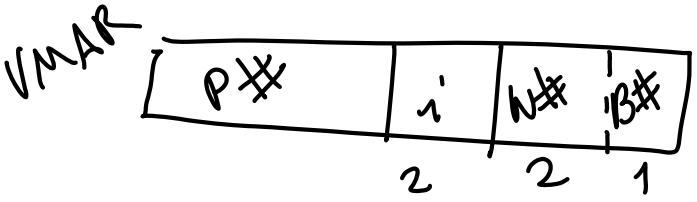
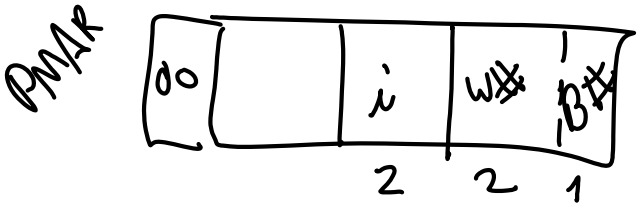


$2^5_B = 32B$



Mem 14-bit A

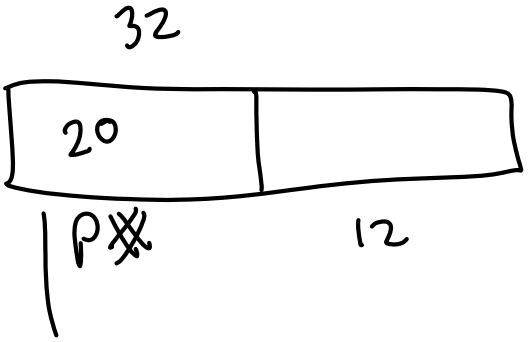
$\rightarrow 2^{14} B = 16kB$

VMEM 16-bit A

$\rightarrow 2^{16} = 64kB$

$2^{14}\left(\frac{bL}{8B}\right) = 2^{11} = 2k\ bL$

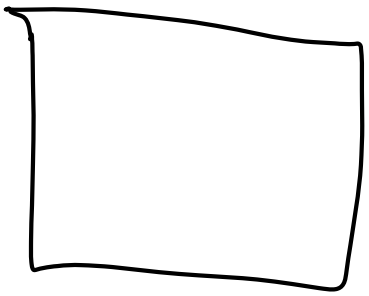
$2^{16}\left(\frac{bL}{8B}\right) = 2^{13} = 8k\ bL$



16-bit Addr,
Byte addrbl.

words → 2B

8B blocks → 2B/W 4W/bL



2-way SA

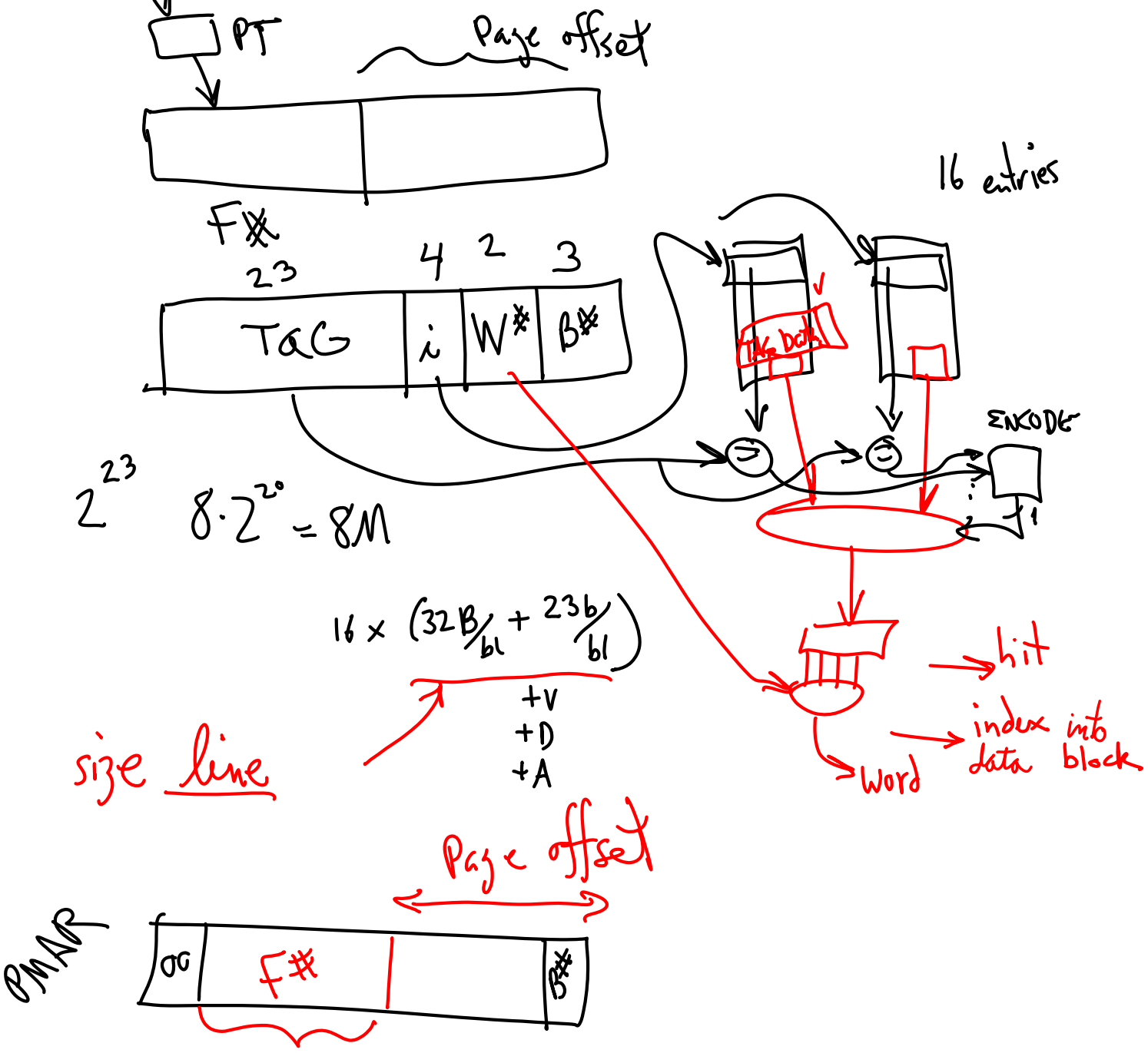
4 entries → i 2bits

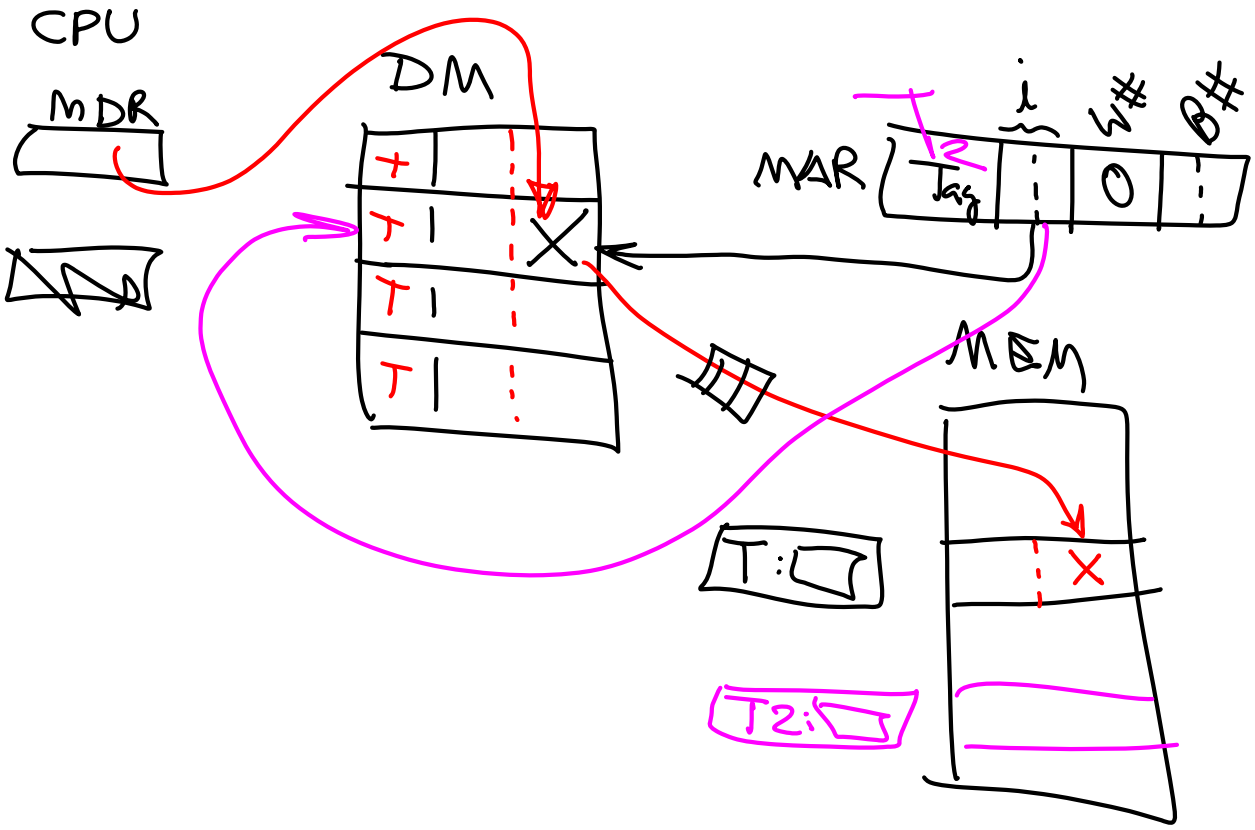
$2^{14} B\left(\frac{W}{2B}\right) = 2^{13} W = 8kW$

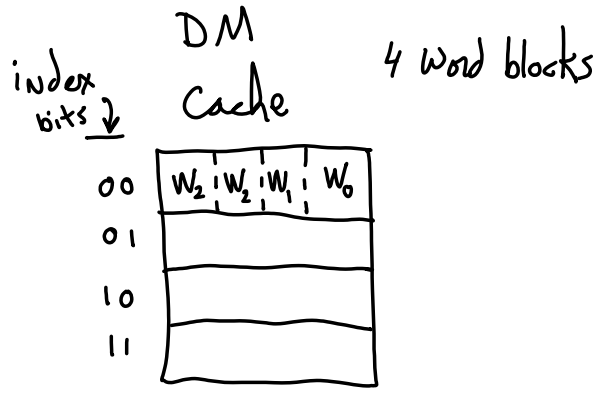
$2^{16} B\left(\frac{W}{2B}\right) = 2^{15} = 32kW$

Pages

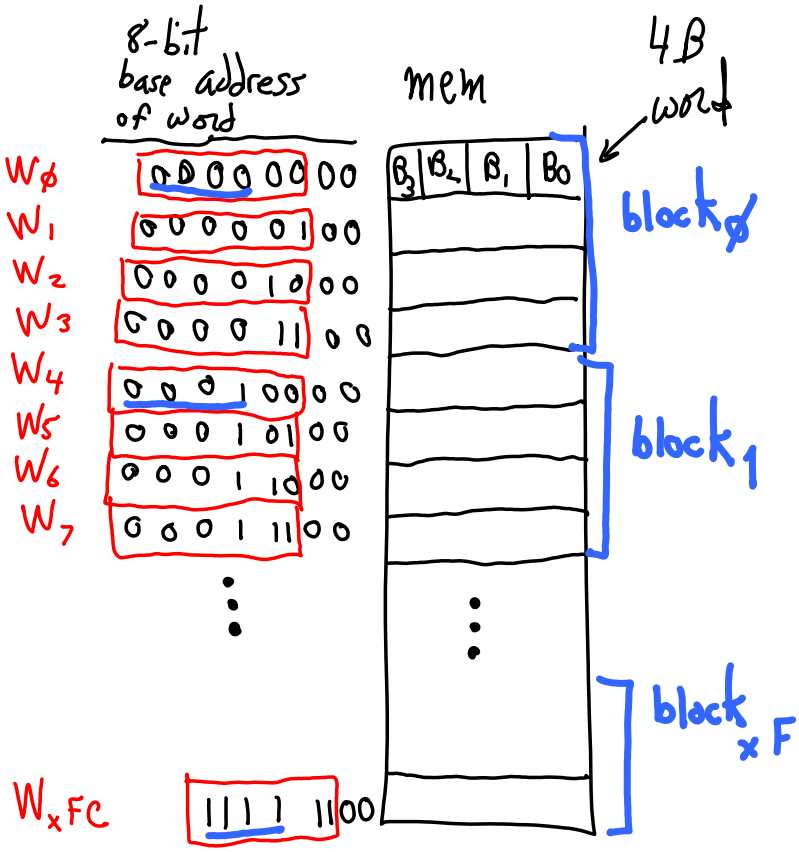
$2^{12} = 4kB$


PT
Page offset
16 entries
F#
23
4
2
3
TAG
i
W#
B#
TAG DATA
ENCODE
2^{23}
$8 \cdot 2^{20} = 8M$
$16 \times (32B/bl + 23b/bl)$
+V
+D
+A
size line
hit
index into data block
word
Page offset
PMAR
00
F#
B#





4 word blocks

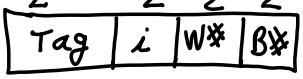


(8-bit address, byte addressable) → 2^8 B Memory

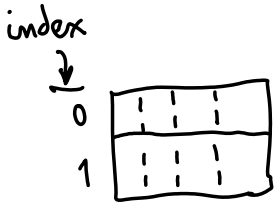
2^8 B mem $\left(\frac{W}{4B}\right) = 2^6$ W mem

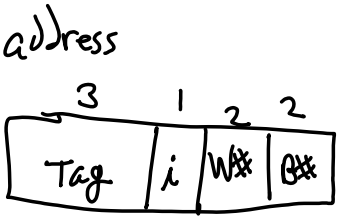
2^6 W mem $\left(\frac{\text{block}}{4W}\right) = 2^4$ block memory

assign address bits

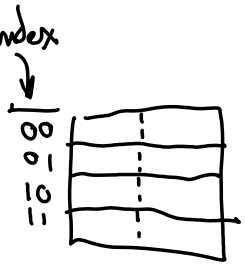


a) Suppose fewer cache entries

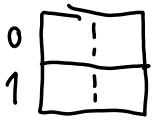




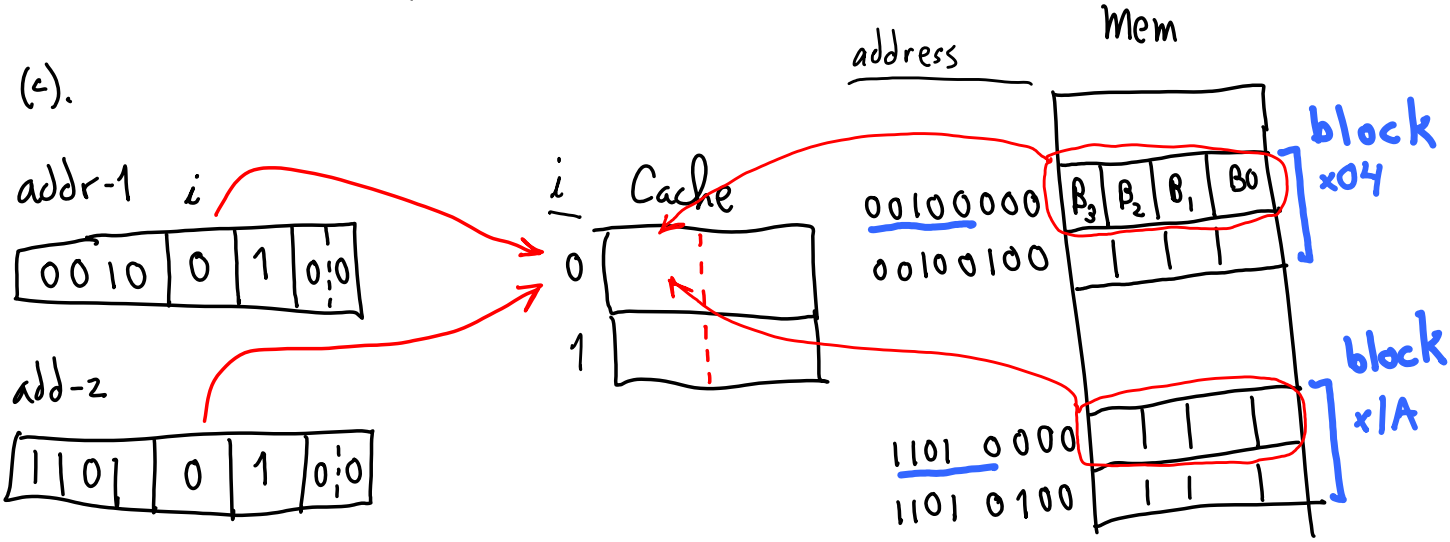
b) Suppose smaller blocks



c) Suppose a) and b)



Suppose (c).

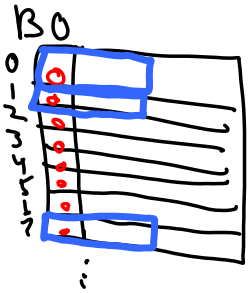


---

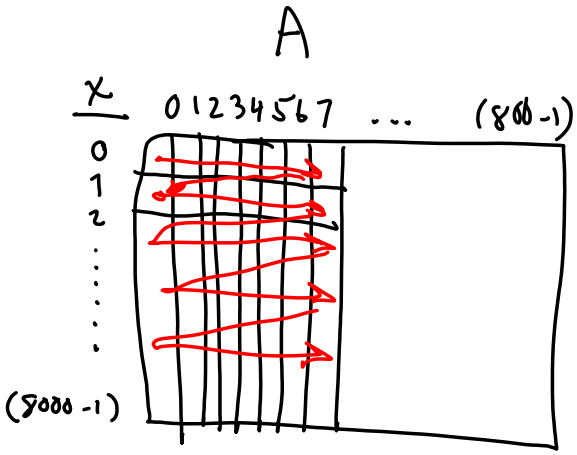
x 0 → (8000-1)

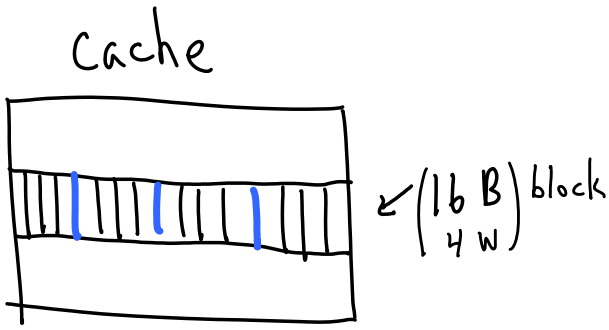
y 0 → 7

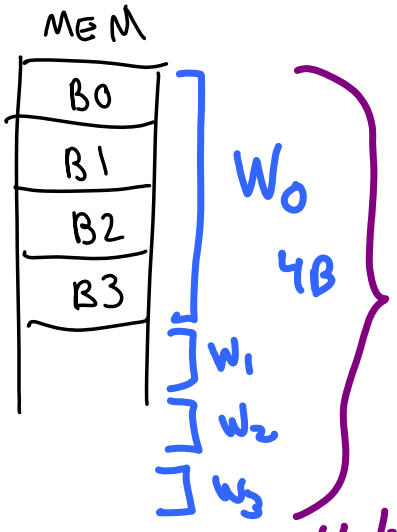
$$A[x][y] = B[y][0] + A[y][x]$$

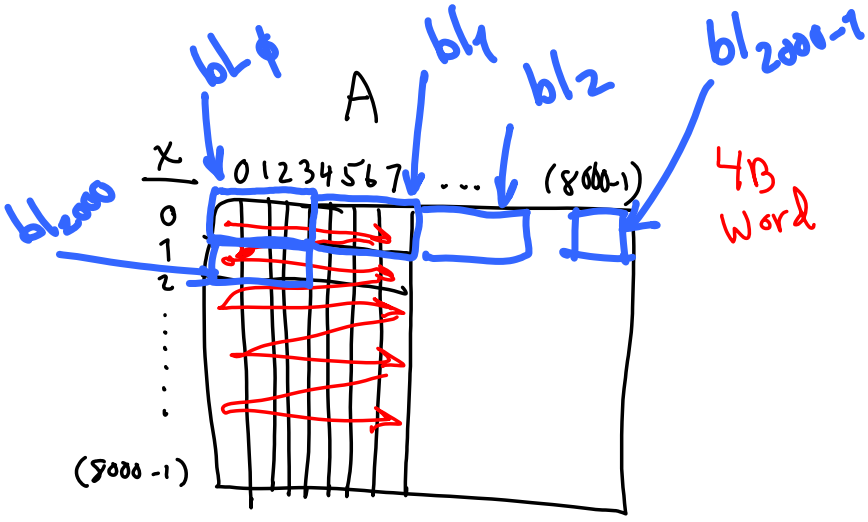


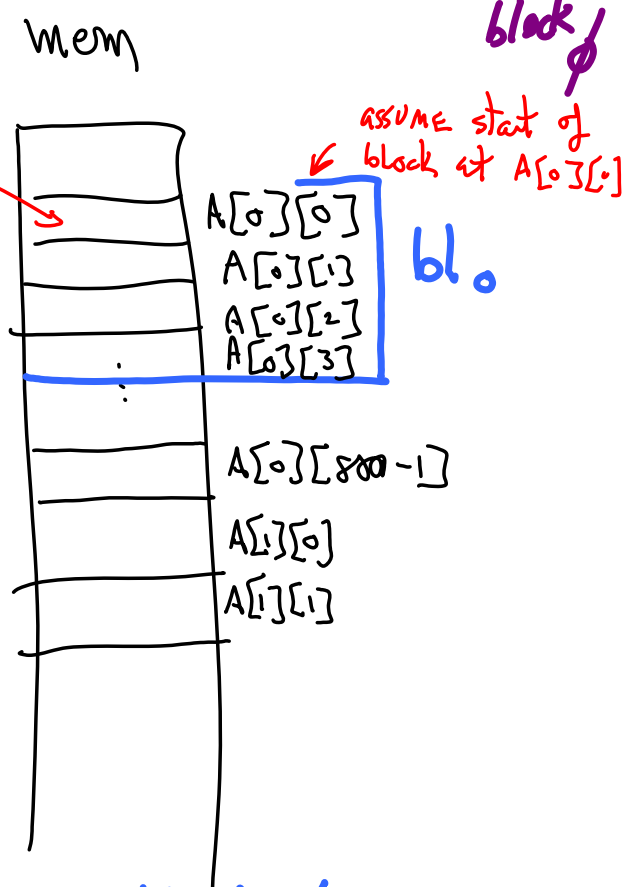
8 blocks



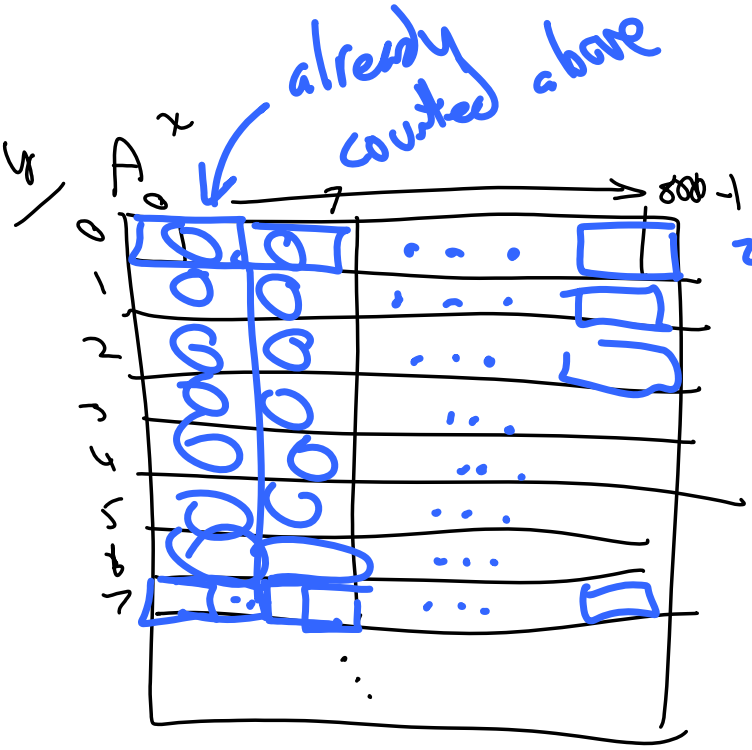




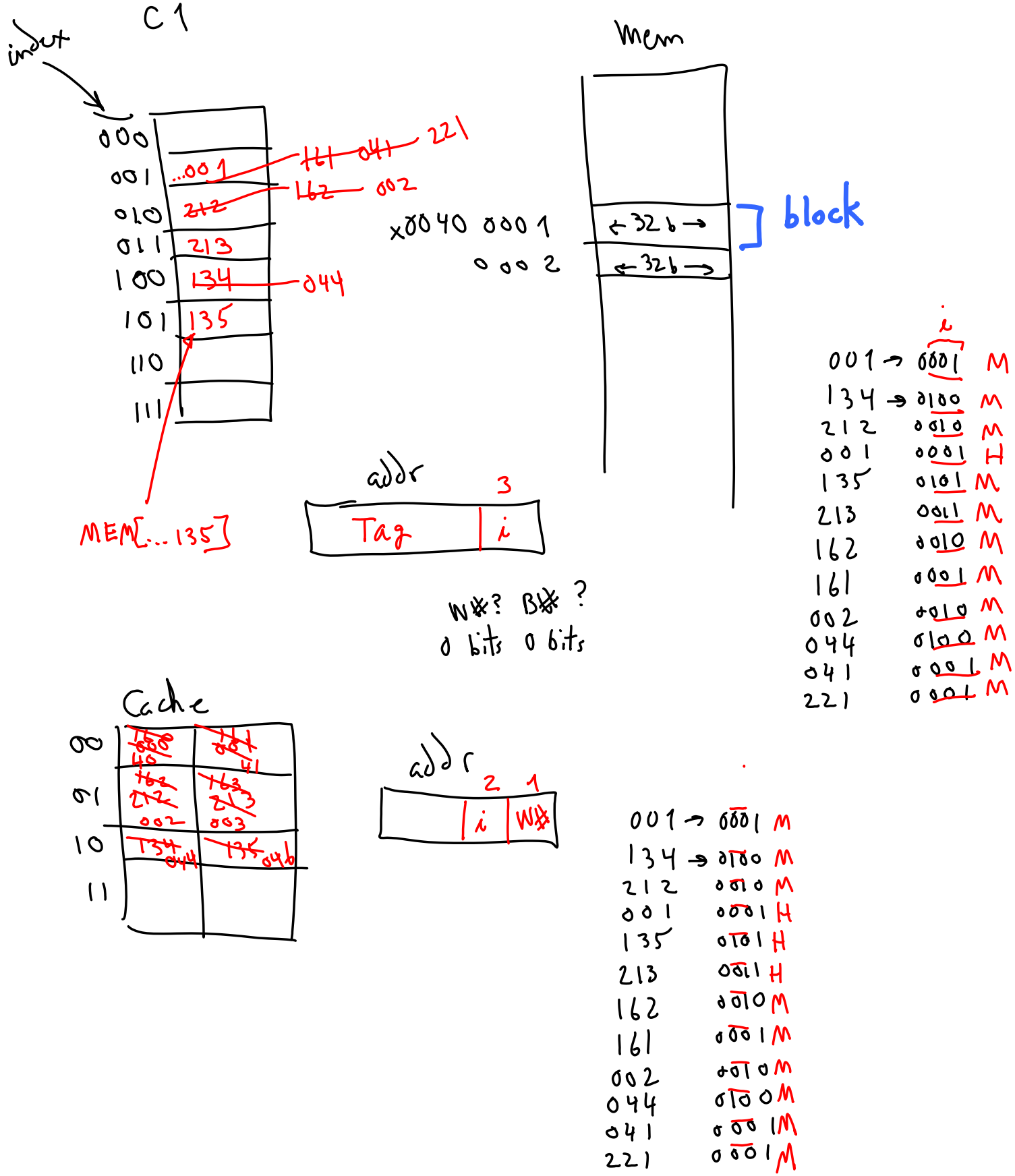


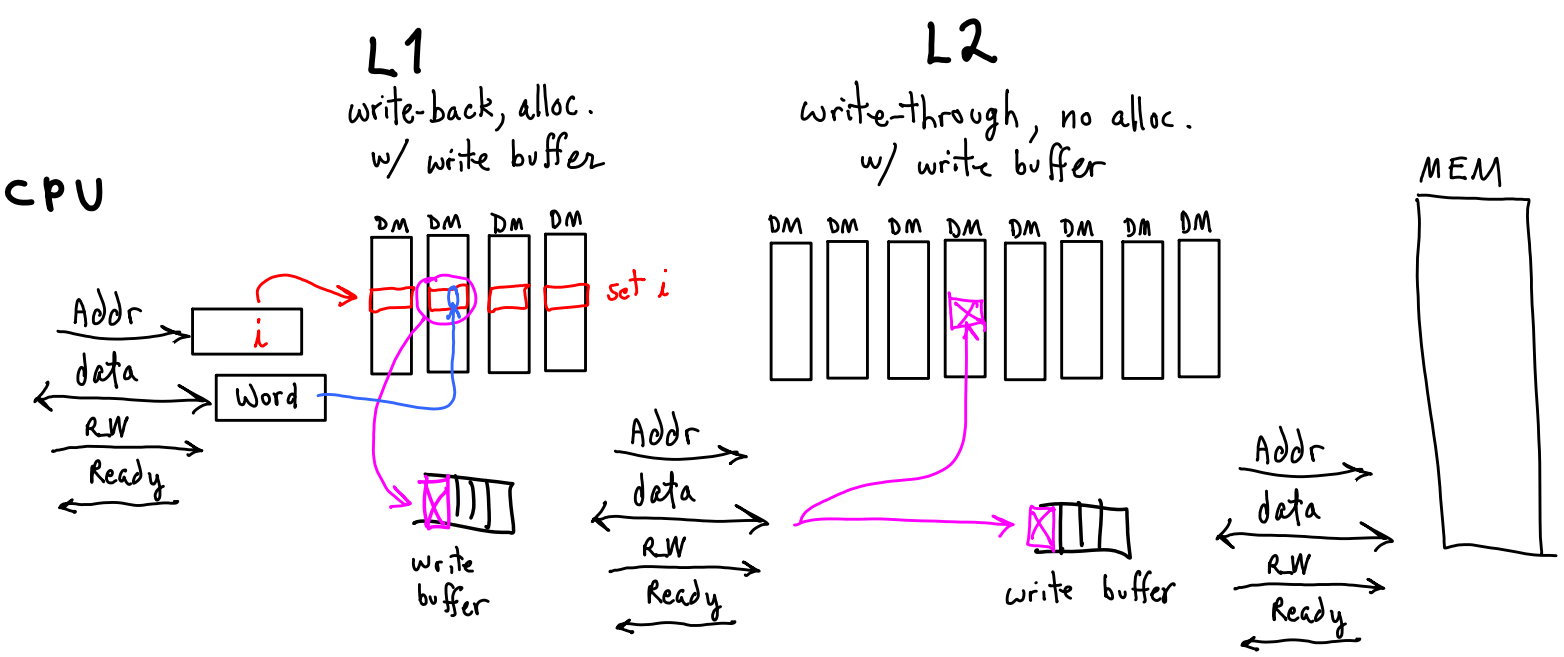


$\left(\frac{2 \text{ blocks}}{\text{row}}\right)$ 8000 rows = 16,000 blocks



2000 blocks/row
x
8 rows
= 16,000 blocks
- 16 blocks
(counted twice)


C 1
index
000
001
010
011
100
101
110
111
...001
212
213
134
135
MEM[... 135]
Mem
block
addr
3
Tag
i
WB? BB?
0 bits 0 bits
Cache
00
01
10
11
addr
2 1
i
WB
001 → 0001 M
134 → 0100 M
212 0010 M
001 0001 H
135 0101 M
213 0011 M
162 0010 M
161 0001 M
002 0010 M
044 0100 M
041 0001 M
221 0001 M



L1: (pick victim from set i ◎)

* (Write victim if dirty) (Send to L2 via buffer) { [Addr] = [TAG, i, W** = 0, B** = 0] = address of victim
[data block of victim] }

** (Send read request to L2) [Addr] = address from CPU
[R_W = Read]

(Send L1.Ready to CPU)

(On L2.Ready)

(get data block from L2.data, put it + tag in victim's entry)

(write word from CPU.data to block, set dirty)

* L2: (write victim block to buffer)
(if hit for victim's Addr, write victim block to cache)

** L2: (if hit for Addr, send block to L1)
(else)
(Send [Addr, R_W = Read] to Mem)
(Wait Mem.Ready)
(write block to cache)
(Send block to L1)