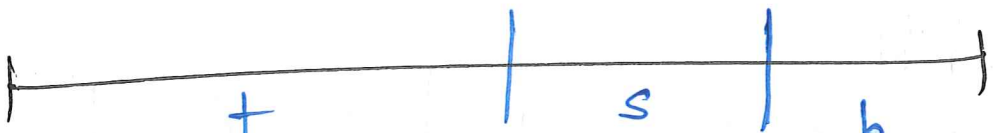
CS 354 - Lecture 12

Plan:

1. Cache
   - Set associative cache
   - fully " "
   - Writes
   - Cache friendly code.
2. Intro to OS.
   - CPU Virtualization
   - Memory "

Review

m-bit address



t: tag

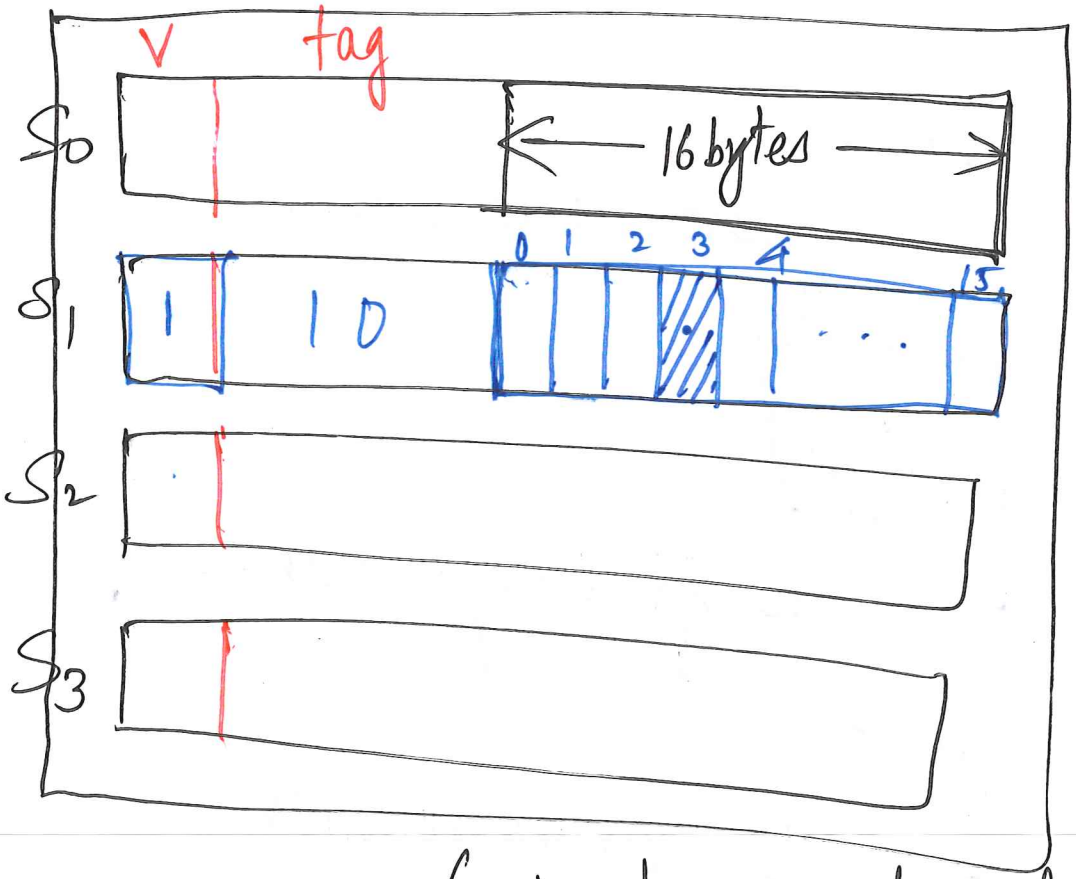
s: set index, $S = 2^s$

b: Block offset, $B = 2^b$

Cache size

$$C = S \times E \times B$$

E: cache lines



E = 1 (direct mapped cache).

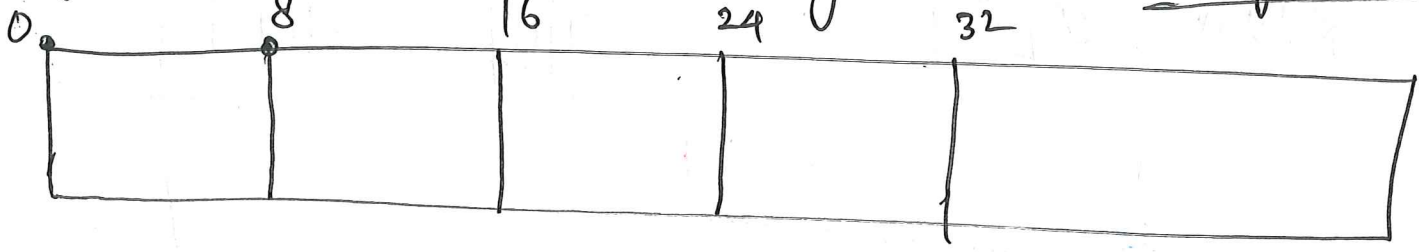
B = 16 bytes

C = S x E x B = 4 X 1 X 16 = 64 bytes

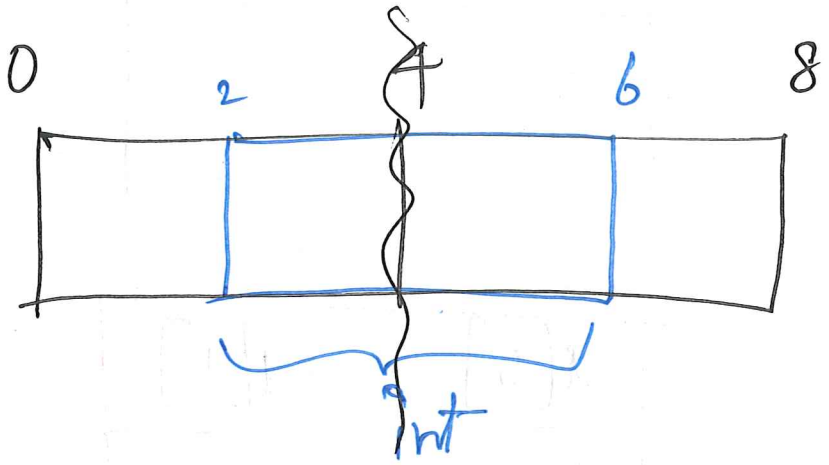
addr = 8 bits

1 0	0 1	0 0 1 1
t	s	b

Dyn. mem. allocator: ③ Alignment = 8 bytes



malloc (n)



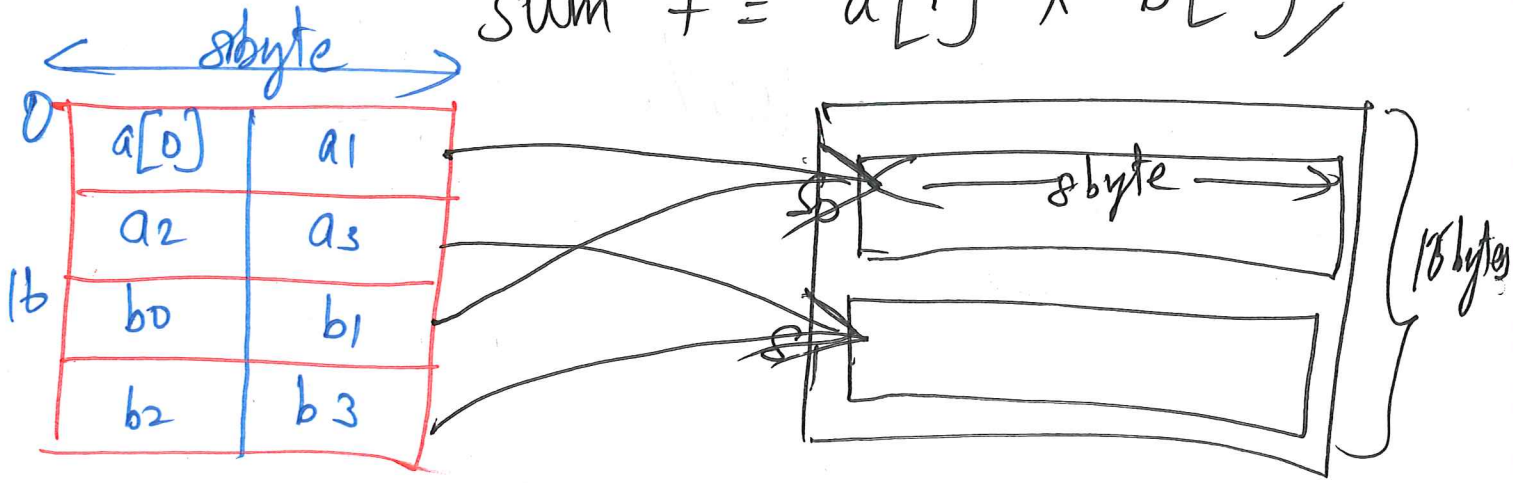
char - 1

short - 2

int - 4

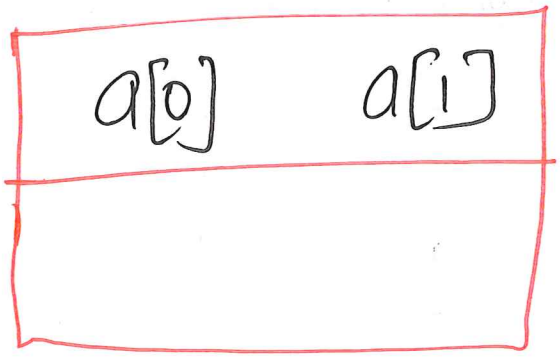
long long - ~~4 or~~ 8

```
int dot_prod (int a[4], int b[4]) {
    int sum;
    for (int i = 0; i < 4; i++)
        sum += a[i] * b[i];
}
```

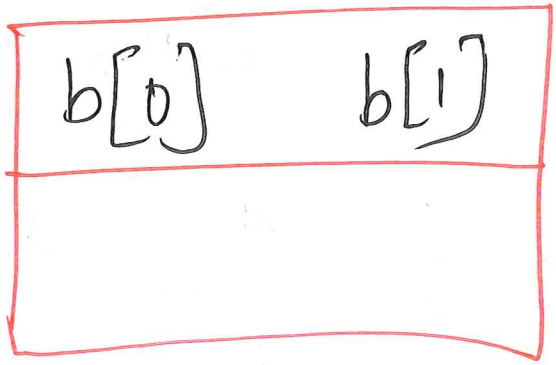


(4)

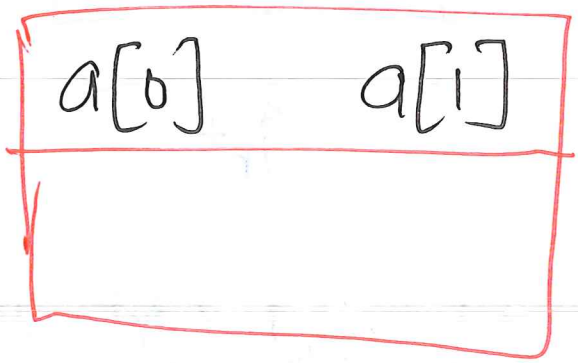
1) Read a[0]



2) Read b[0]



3) Read a[1]



$$\text{miss rate} = \frac{\#\ \text{misses}}{\#\ \text{refs.}} = \frac{8}{8}$$

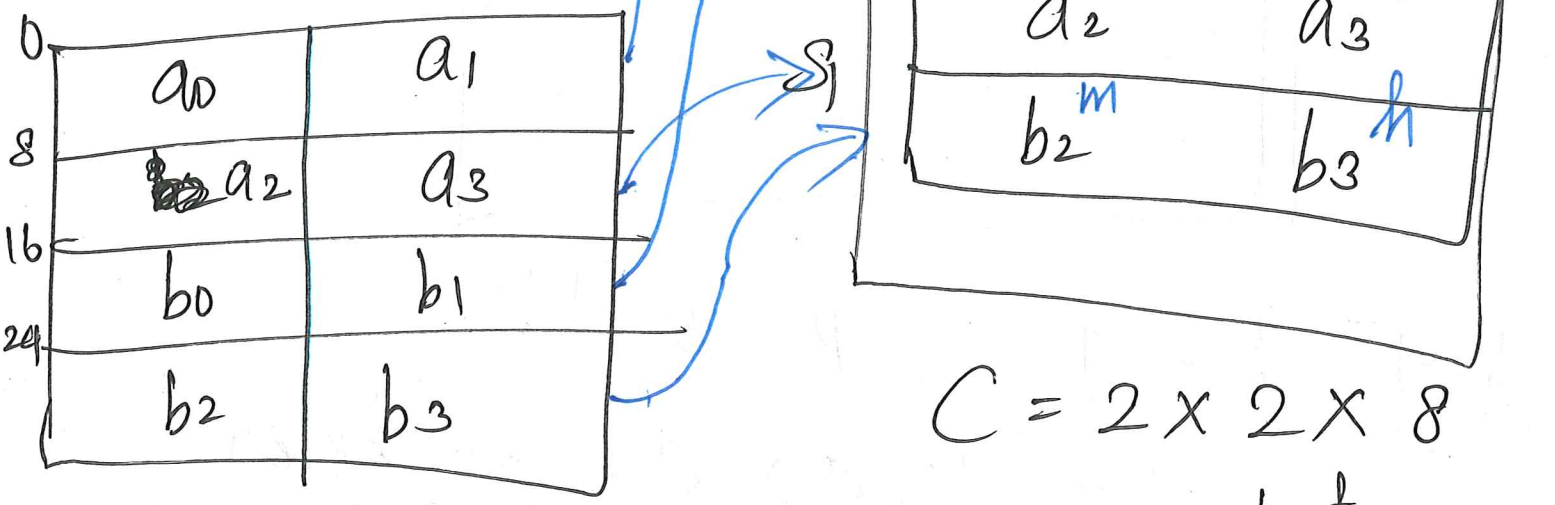
$$= 100\%.$$

Set associative cache

E = 2

2 - way set associative cache

E



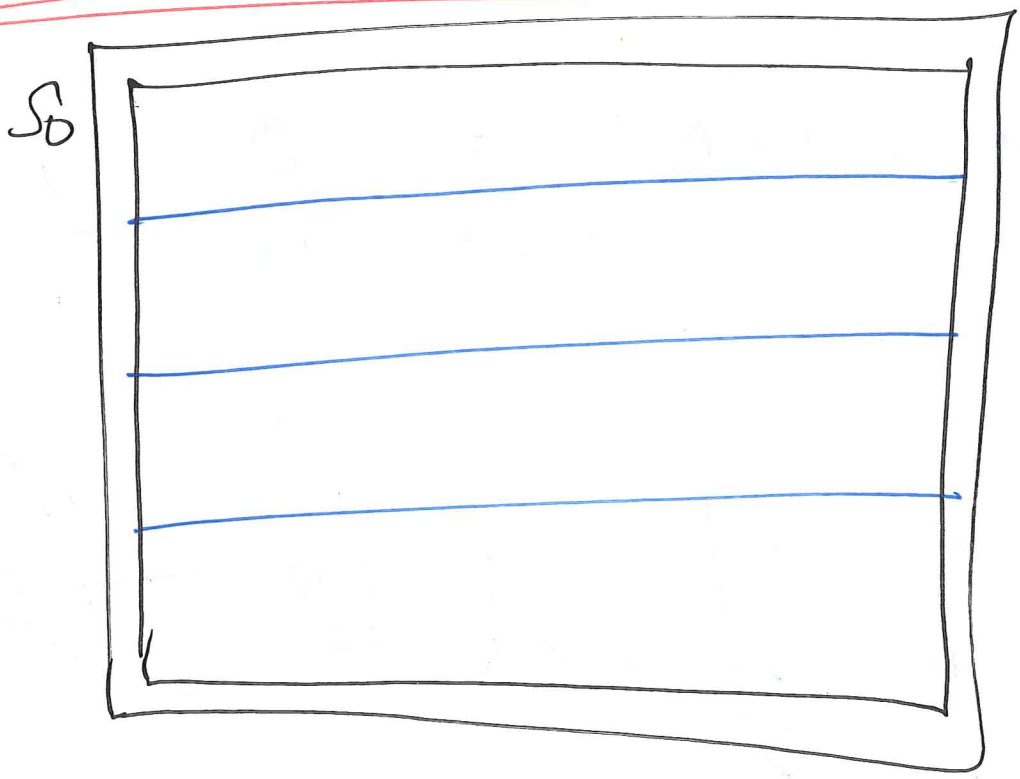
$C = 2 \times 2 \times 8$

$= 32$ bytes

miss rate $= \frac{4}{8} = 50\%$

$$1 < E < \frac{C}{B}$$

Fully associative cache (8)



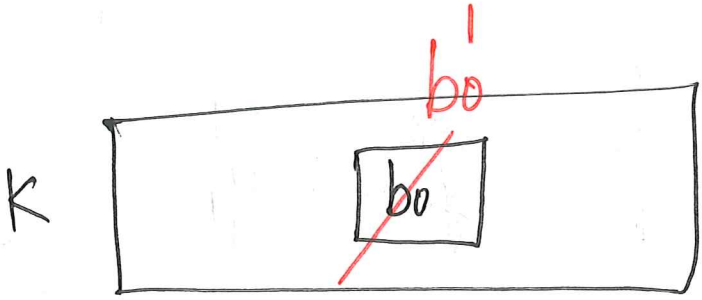
$S = 1 \quad E = 4 \quad B = 8$

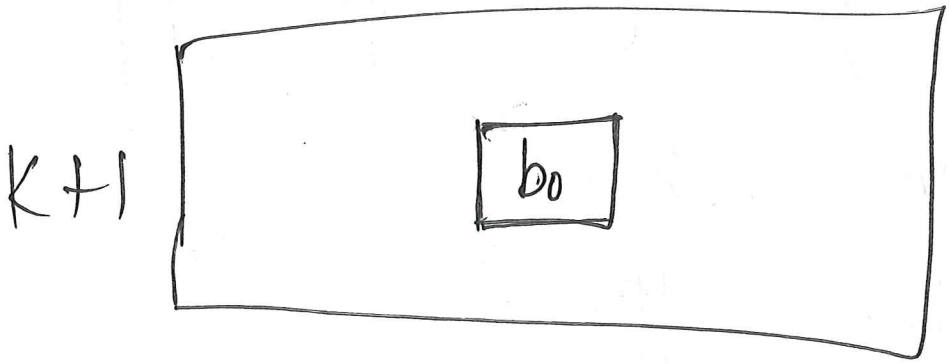
$C = 1 \times 4 \times 8 = \underline{32 \text{ bytes.}}$

$$E = \frac{C}{B}$$

"Main memory" is a fully associative cache.

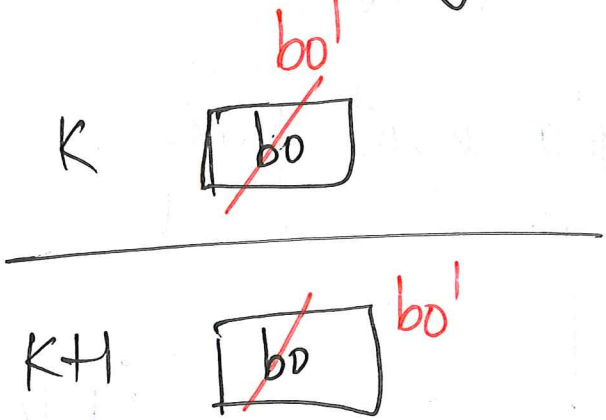
Writes



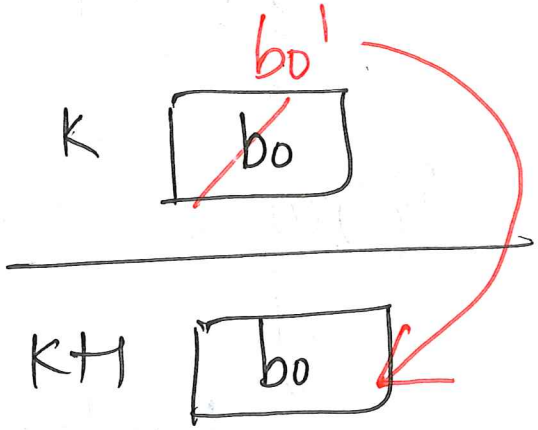


Writes

Write through

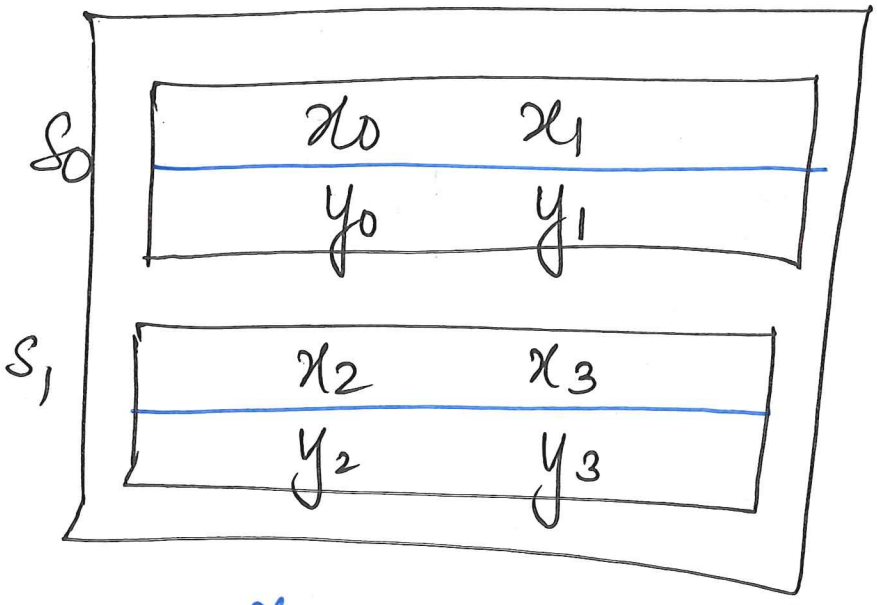


Write back.



b0' written to K+1 only if eviction happens.

Evictions



Assume: $x_0 \ldots x_7$

$y_0 \ldots y_7$

Read x_4 x_5 → maps to s_0.

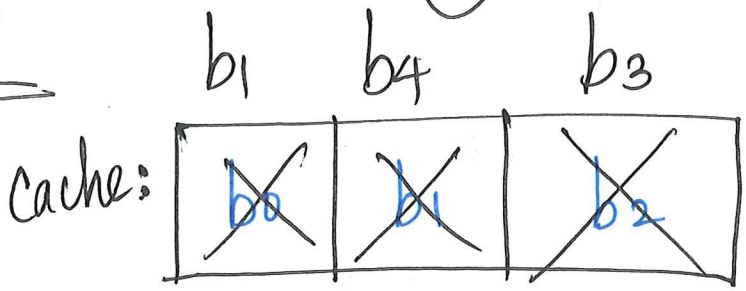
What are we going to evict?

x_0 x_1 y_0 y_1

Policy: Cache replacement policy.

- Random
- LRU — Least recently used
- LFU — Frequently..

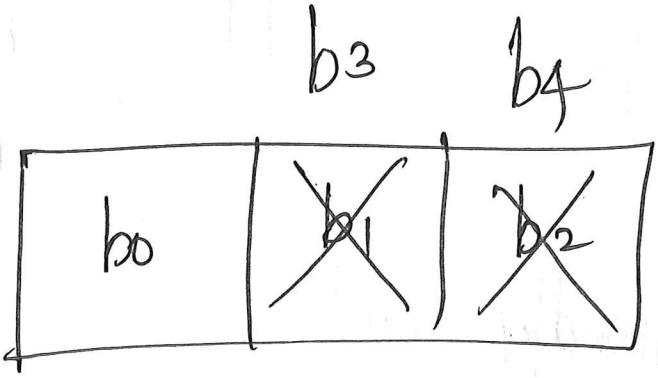
LRU



References: b0 b1 b2 b1 b0 b3 b4 b1

m m m h h m m m

LFU



Ref: b0 b0 b0 b1 b2 b1 b2 | b3 b3 b3 b3 | b4

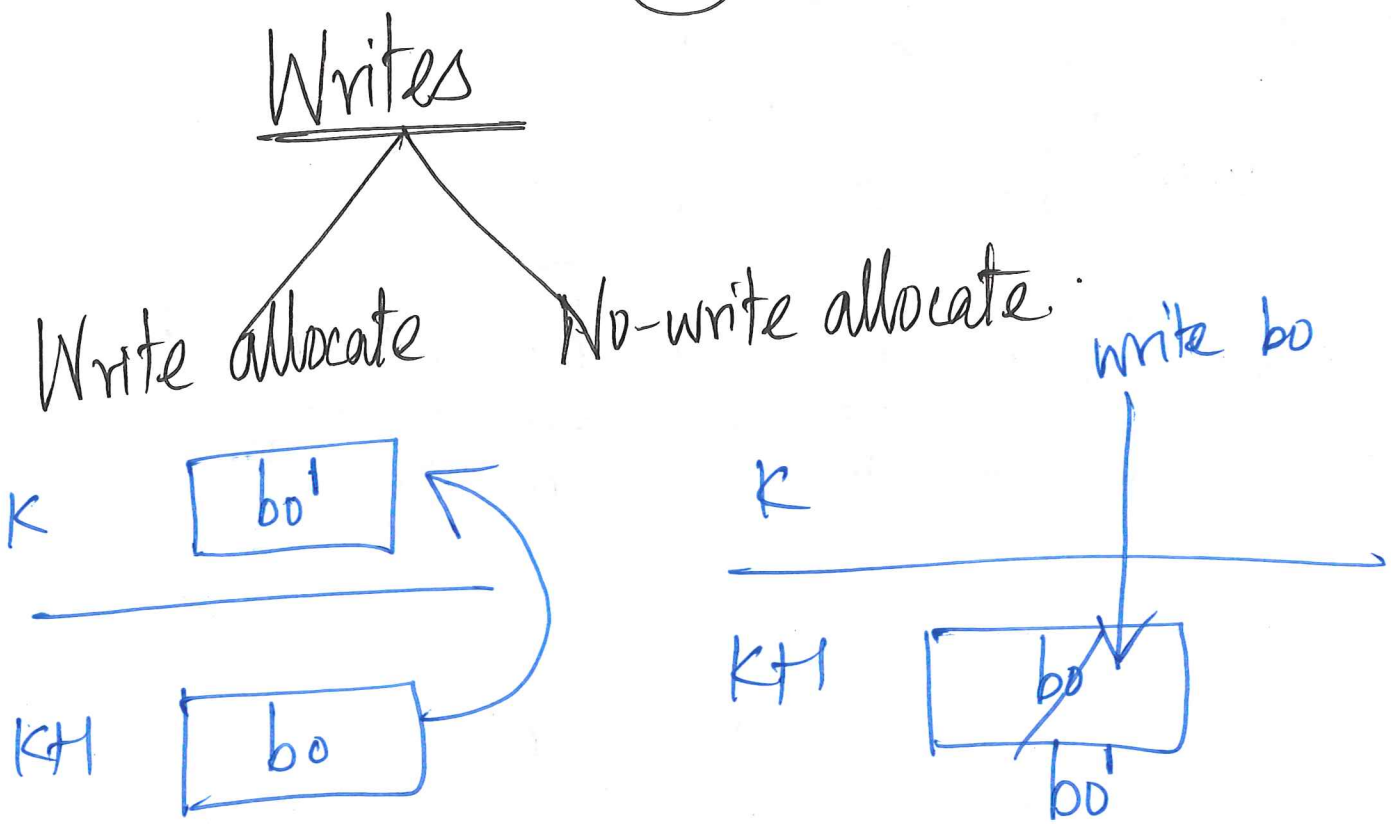
b0 - 3

b1 - 2

b2 - 2

b3 - 4

Writes



Cache Friendly Code

1. miss rate = $\frac{\#\text{ misses}}{\#\text{ references}}$

2. hit rate = 1 − miss rate

3. hit time = $t_{\text{set selection}} + t_{\text{line iden.}} + t_{\text{word iden.}}$

4. miss penalty.

Transpose

$$M_{(src)} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad M^t_{(dst)} = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

```
for ( i=0; i< 2; ++i )
    for(j=0; j<2; ++j)
        dst[j][i] = src[i][j];
```

* src starts at 0; dst starts at 16.
* Cache - direct mapped, write through, write allocate.
* B = 8 bytes

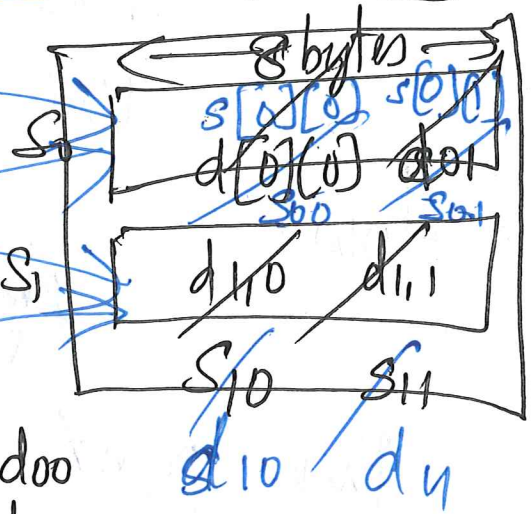
C = 16 bytes

src

0,0 m	0,1 m
1,0 m	1,1 h

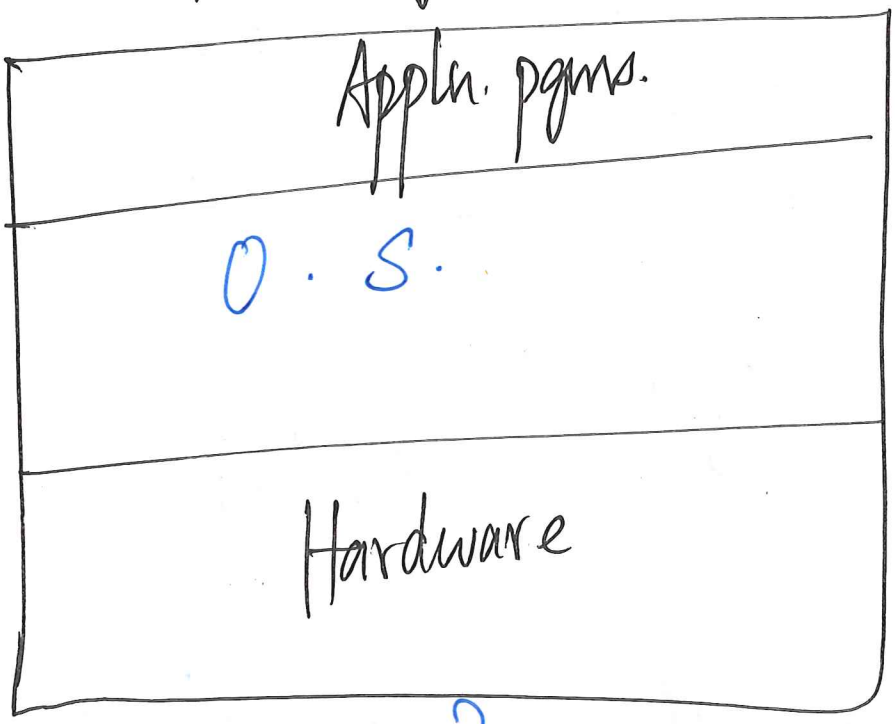
dst

0,0 m	0,1 m
1,0 m	1,1 m



$s_{00} \rightarrow d_{00}$
$s_{01} \rightarrow d_{10}$
$s_{10} \rightarrow d_{01}$
$s_{11} \rightarrow d_{11}$

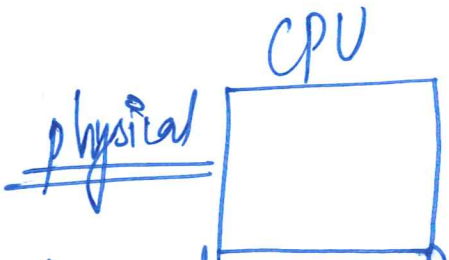
Intro to Operating Systems.



What is an OS?

⟹ software that makes the machine much easier to use.

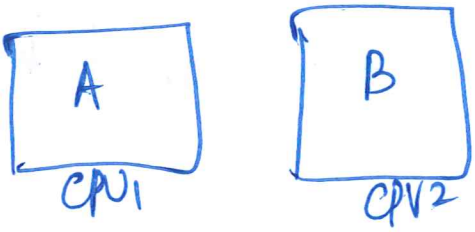
How?

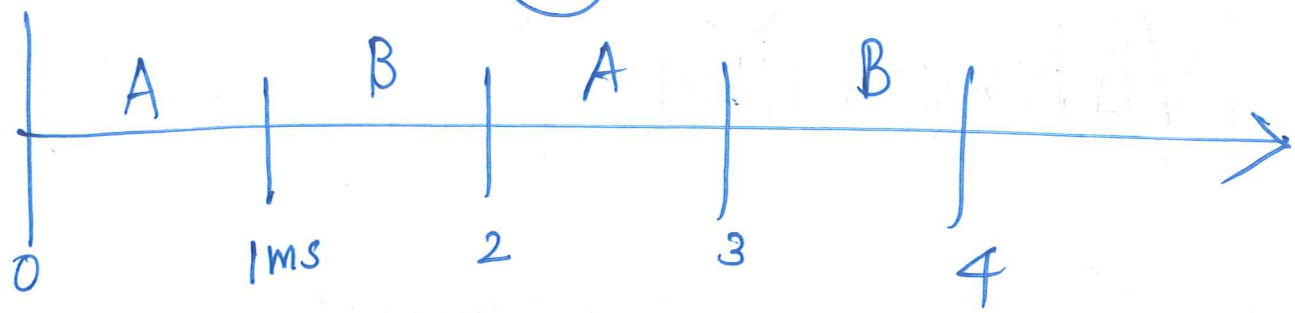


1 CPU

Run pgm A and pgm B on CPU.

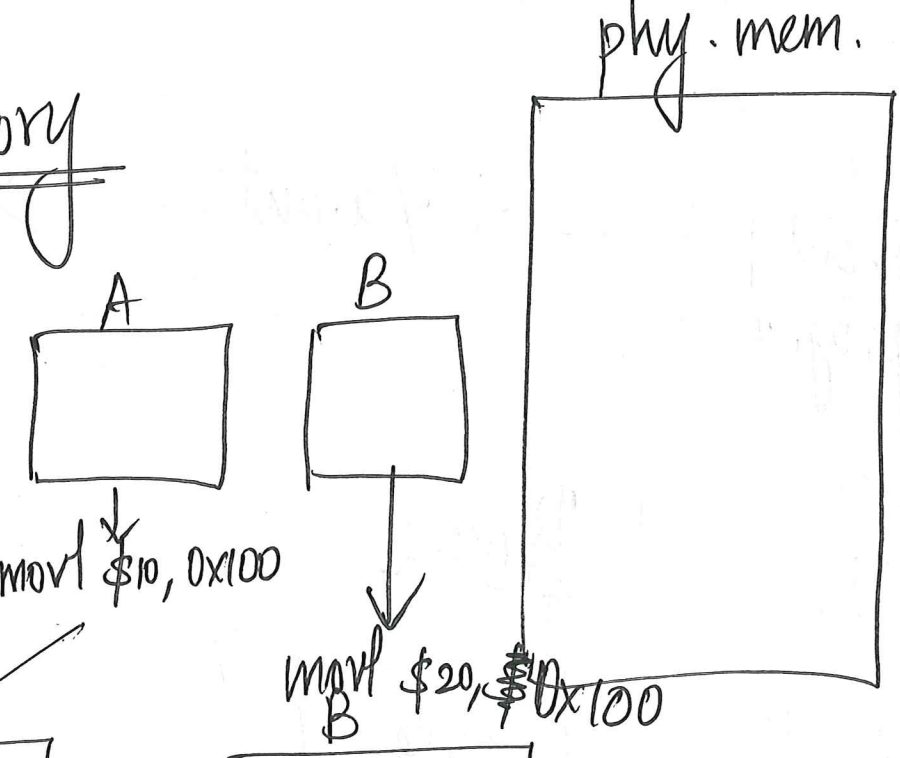
Illusion: virtuals CPUs





idea: time-sharing

⇒ memory



"memory virtualization"

14

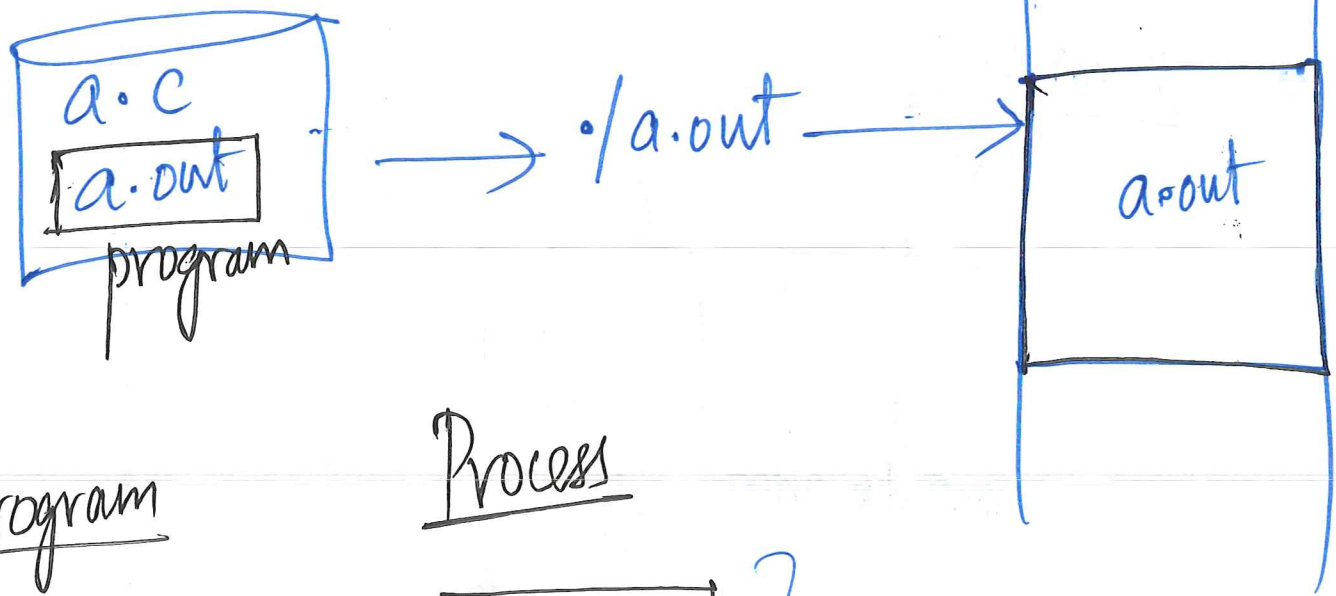
Virtualization

CPU

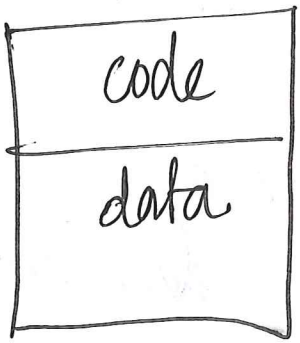
memory

Process – basic abstraction.

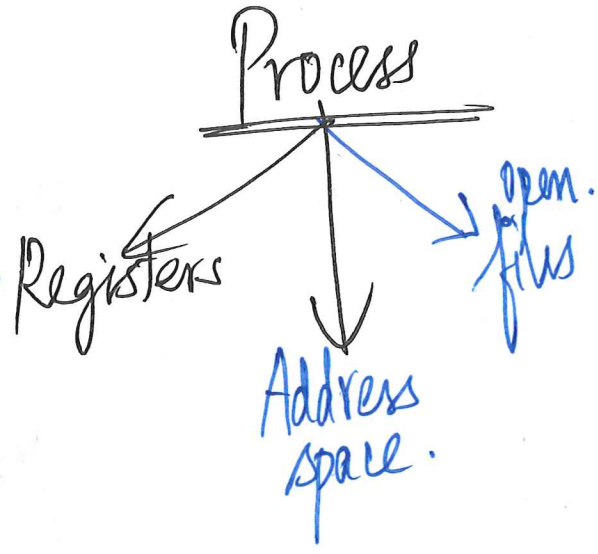
Memory.

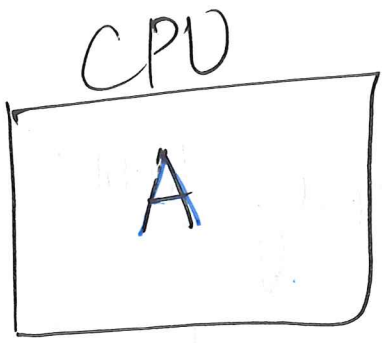


Program



Process



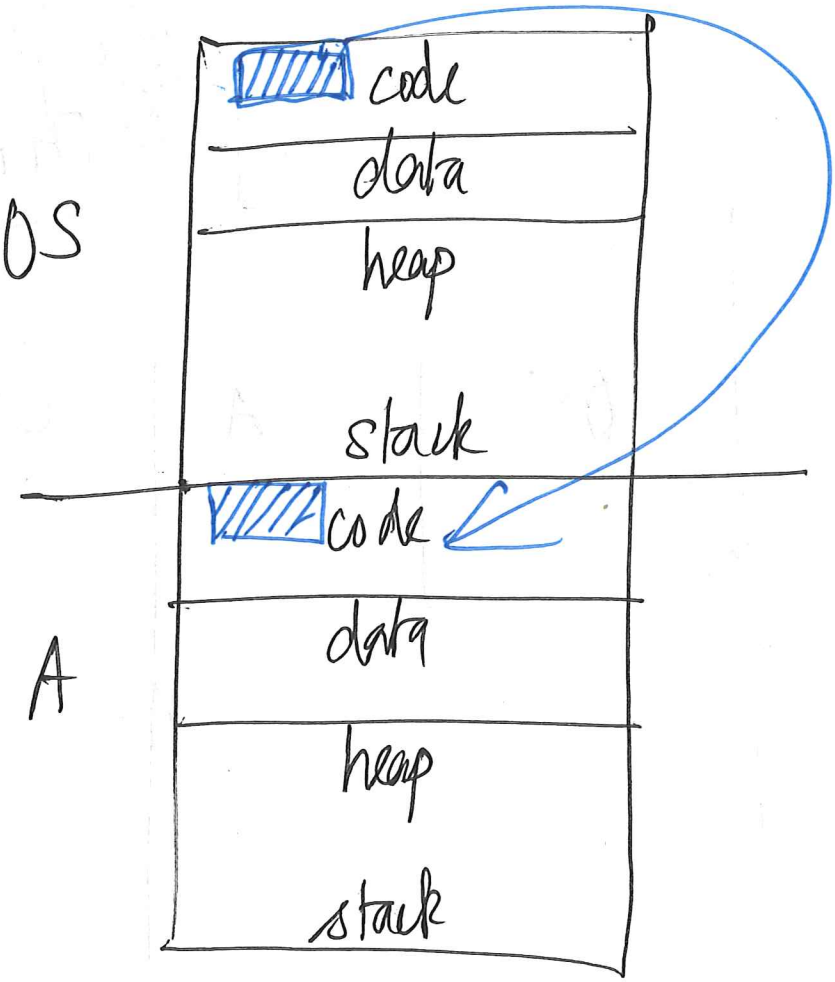


How do you switch to B?

choose proc A to run.

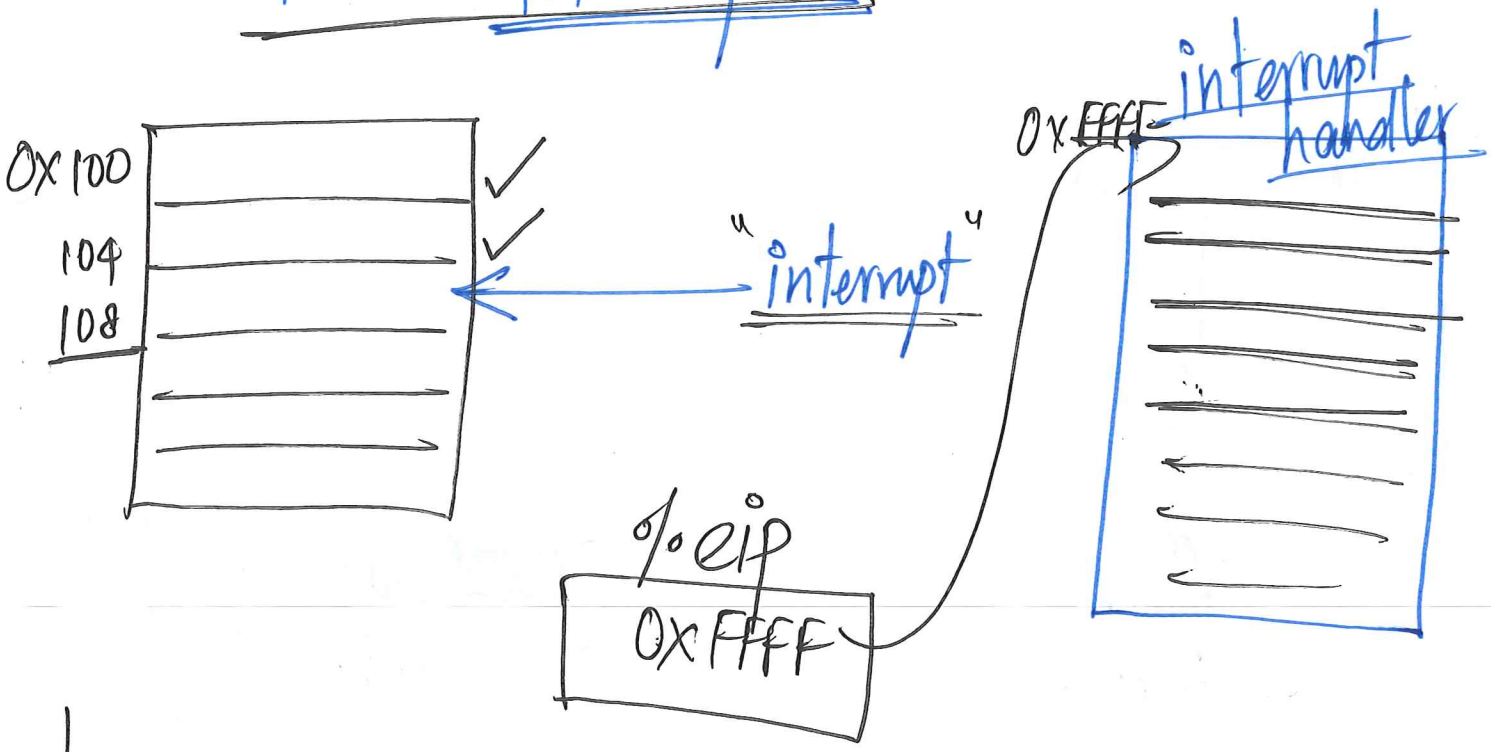
"yield();" X

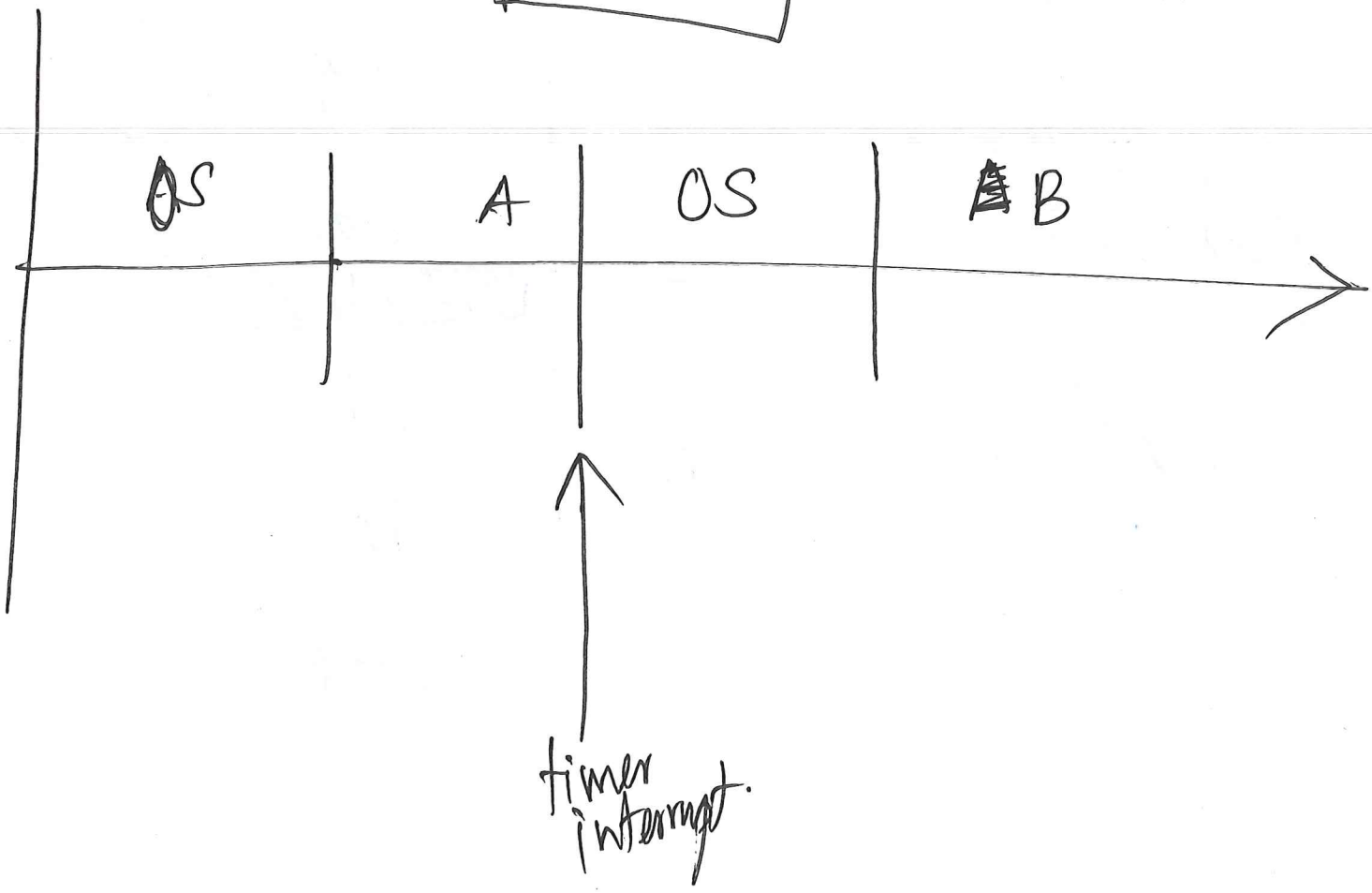
while(1)
;



Hardware support

"timer interrupt" — every 10 m-s.





CPU Virtualization (17)

1. Mechanisms. → "timer interrupt", mode bit
2. Policies

@boot: OS tells the hardware the location of the interrupt handlers.

OS ~~setup~~ starts the timer interrupt. (10 ms)

---

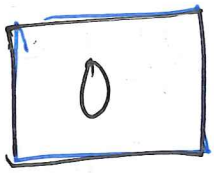
timer interrupt: can be switched on / off.

change the addr of the timer interrupt

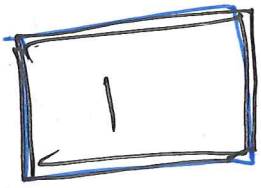
What if a user process switches off the timer interrupt?

Hardware support

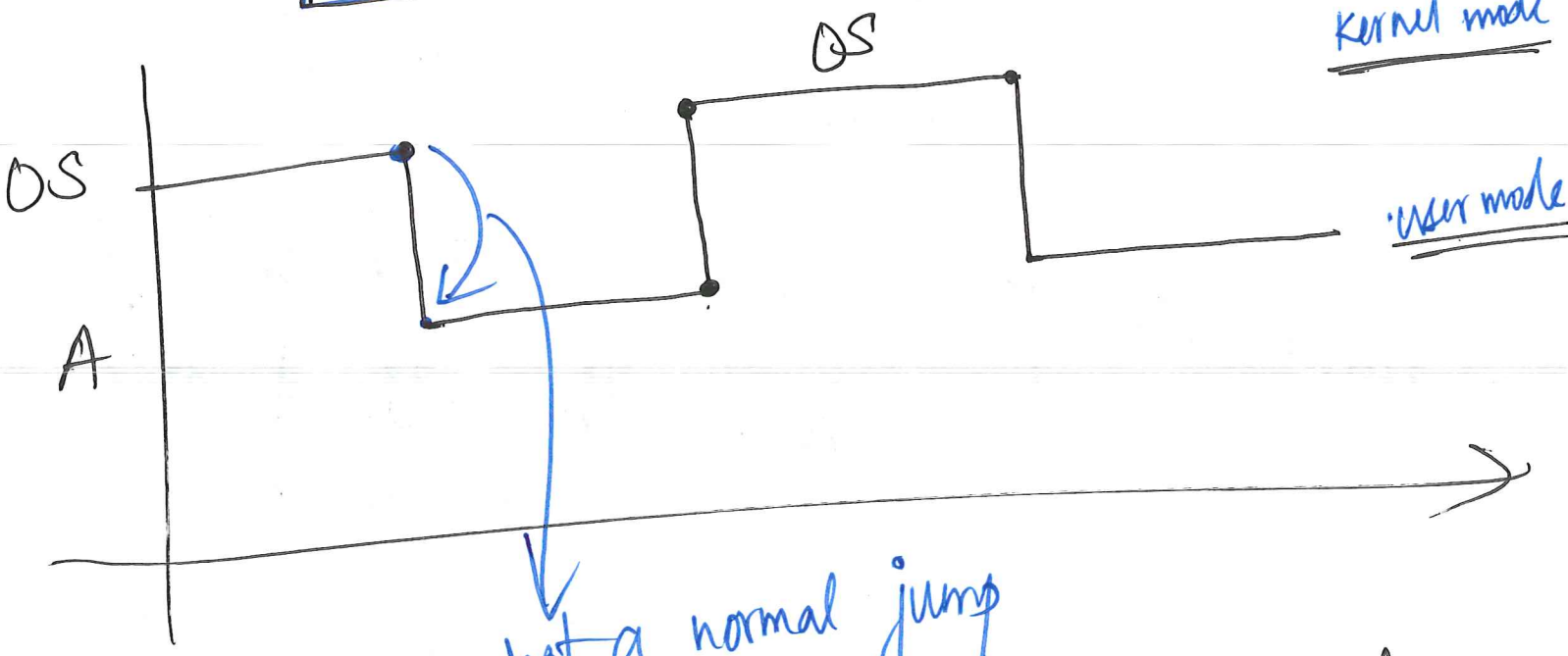
"privilege mode" (status register).



→ user mode



→ "kernel" mode

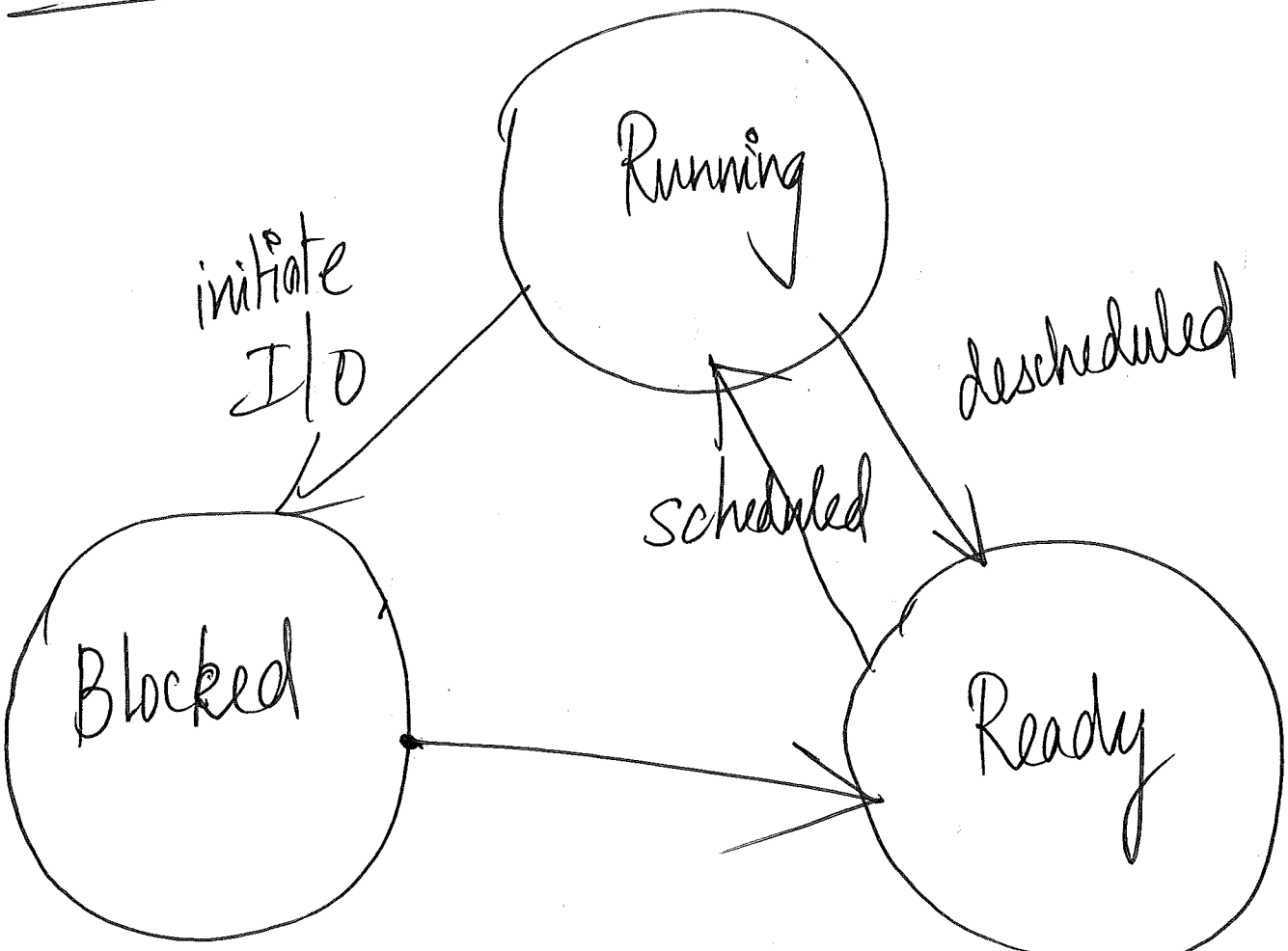


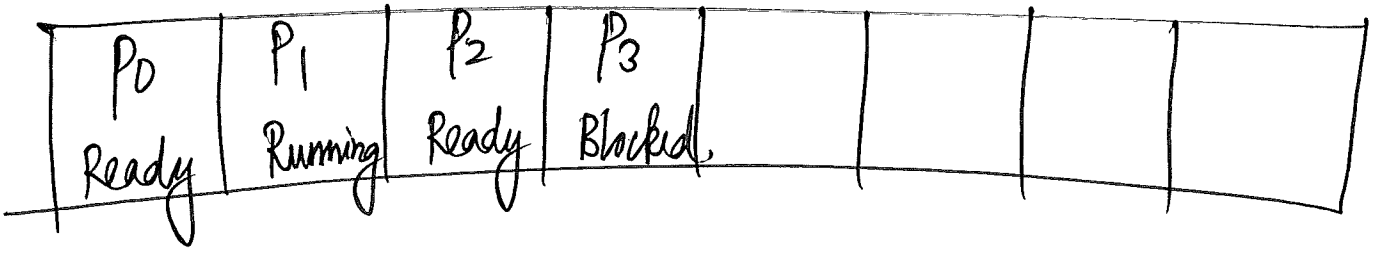
not a normal jump

jumps to user code

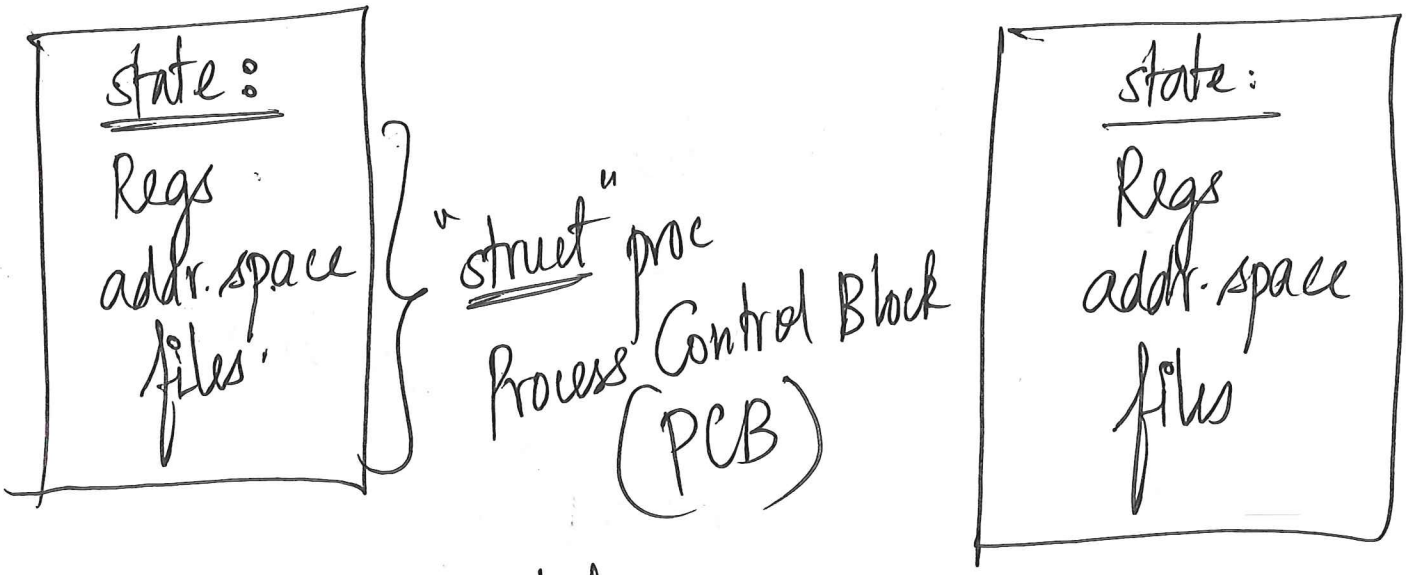
decreases ~~raises~~ the privilege level to ~~kernel~~ user mode.

Process states





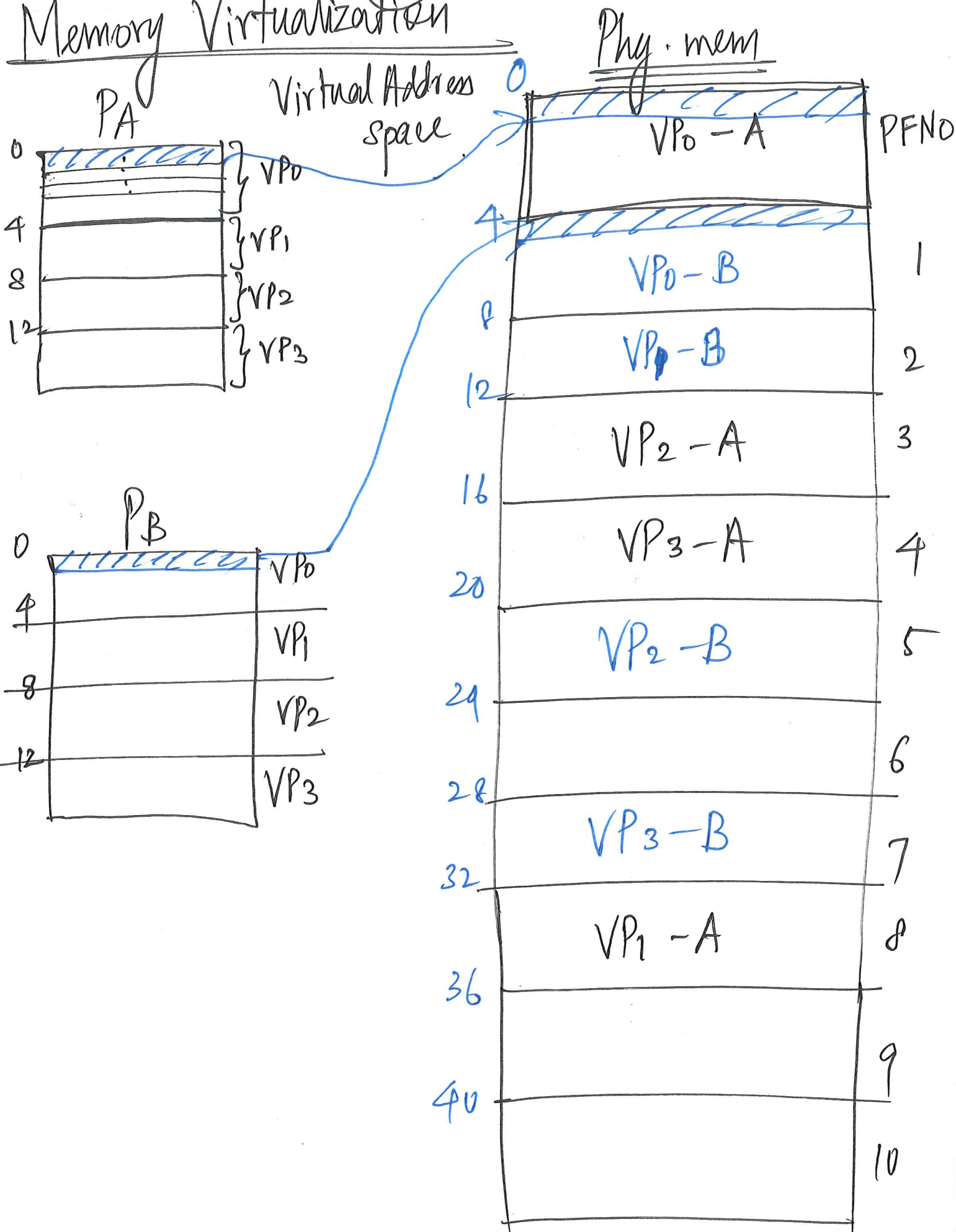
A OS B



Context Switch

1. Save the state of P_A.
2. Restore the state of P_B (that was previously saved).

Memory Virtualization



CS 354: Intro to Computer Systems (Spring 2018)

Lecture 13 - Set Associative Caches

The following problem concerns basic cache lookups.

- The memory is byte addressable.
- Memory accesses are to **1-byte words** (not 4-byte words).
- Physical addresses are 12 bits wide.
- The cache is 4-way set associative, with a 2-byte block size and 32 total lines.

E = 4

B = 2

S = 8

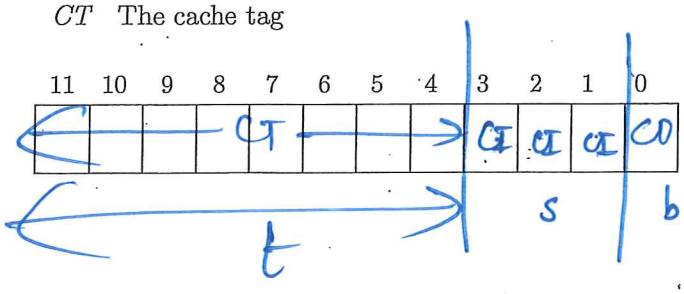
In the following tables, **all numbers are given in hexadecimal.** The contents of the cache are as follows:

4-way Set Associative Cache																
Index	Tag	Valid	Byte 0	Byte 1	Tag	Valid	Byte 0	Byte 1	Tag	Valid	Byte 0	Byte 1	Tag	Valid	Byte 0	Byte 1
0	29	0	34	29	87	0	39	AE	7D	1	68	F2	8B	1	64	38
1	F3	1	0D	8F	3D	1	0C	3A	4A	1	A4	DB	D9	1	A5	3C
2	A7	1	E2	04	AB	1	D2	04	E3	0	3C	A4	01	0	EE	05
3	3B	0	AC	1F	E0	0	B5	70	3B	1	66	95	37	1	49	F3
4	80	1	60	35	2B	0	19	57	49	1	8D	0E	00	0	70	AB
5	EA	1	B4	17	CC	1	67	DB	8A	0	DE	AA	18	1	2C	D3
6	1C	0	3F	A4	01	0	3A	C1	F0	0	20	13	7F	1	DF	05
7	0F	0	00	FF	AF	1	B1	5F	99	0	AC	96	3A	1	22	79

Part 1

The box below shows the format of a physical address. Indicate (by labeling the diagram) the fields that would be used to determine the following:

CO The block offset within the cache line
CI The cache index
CT The cache tag



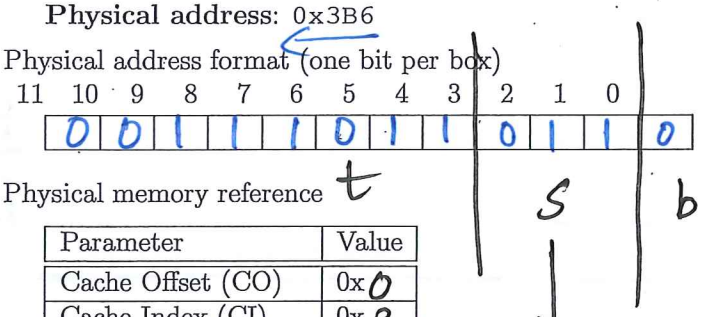
Part 2

For the given physical address, indicate the cache entry accessed and the cache byte value returned **in hex**. Indicate whether a cache miss occurs.

If there is a cache miss, enter "-" for "Cache Byte returned".

Physical address: `0x3B6`

Physical address format (one bit per box)



Physical memory reference

Parameter	Value
Cache Offset (CO)	0x 0
Cache Index (CI)	0x 3
Cache Tag (CT)	0x 3B
Cache Hit? (Y/N)	Y
Cache Byte returned	0x 66

tag = 0011 1011

0x 3B

Part 3

In the 4-way Set Associative Cache given above, list all the hex memory addresses that will **hit** in **Set 7**.

0x AF | 1 1 1 _

0xAFE, 0xAFF

0x 3AE, 0x 3AF