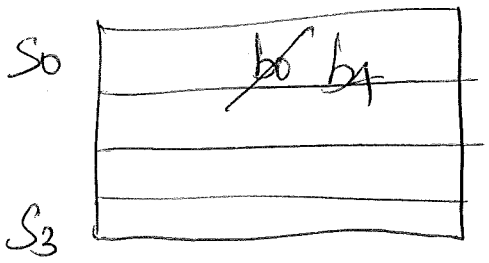
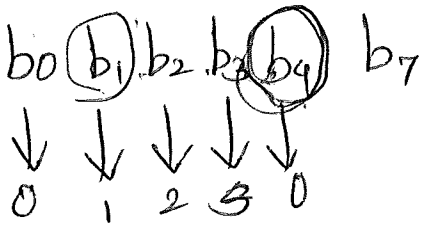
CS 354 - Lecture 13





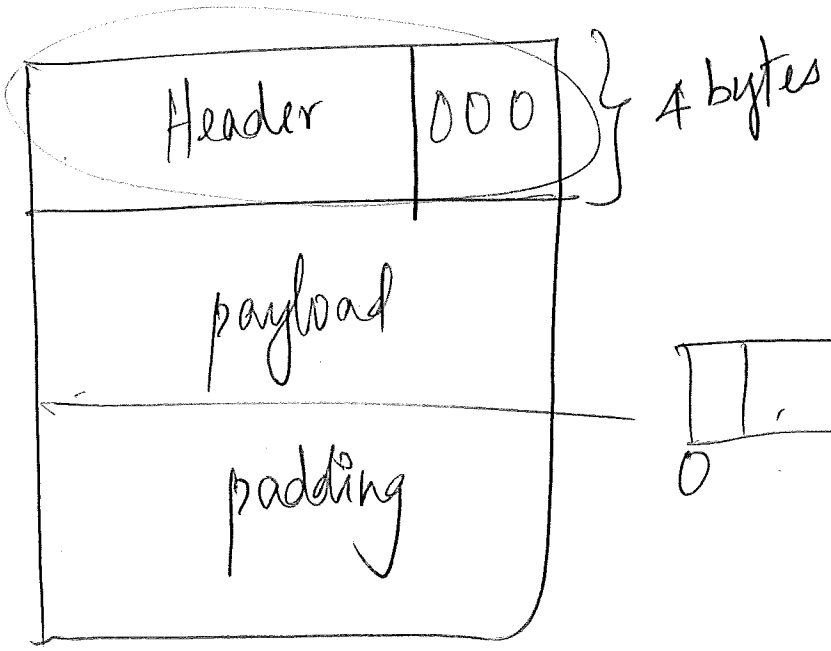
Direct Mapped Cache

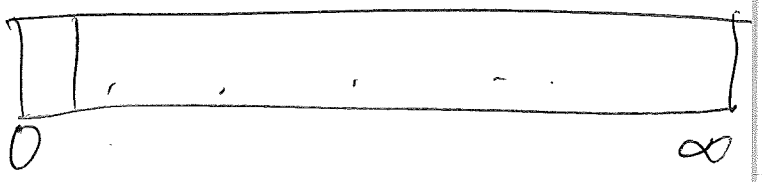
b0 b1 b2 b3 (b4)

Cold C . C . C conflict miss!

cache warmed up.

block

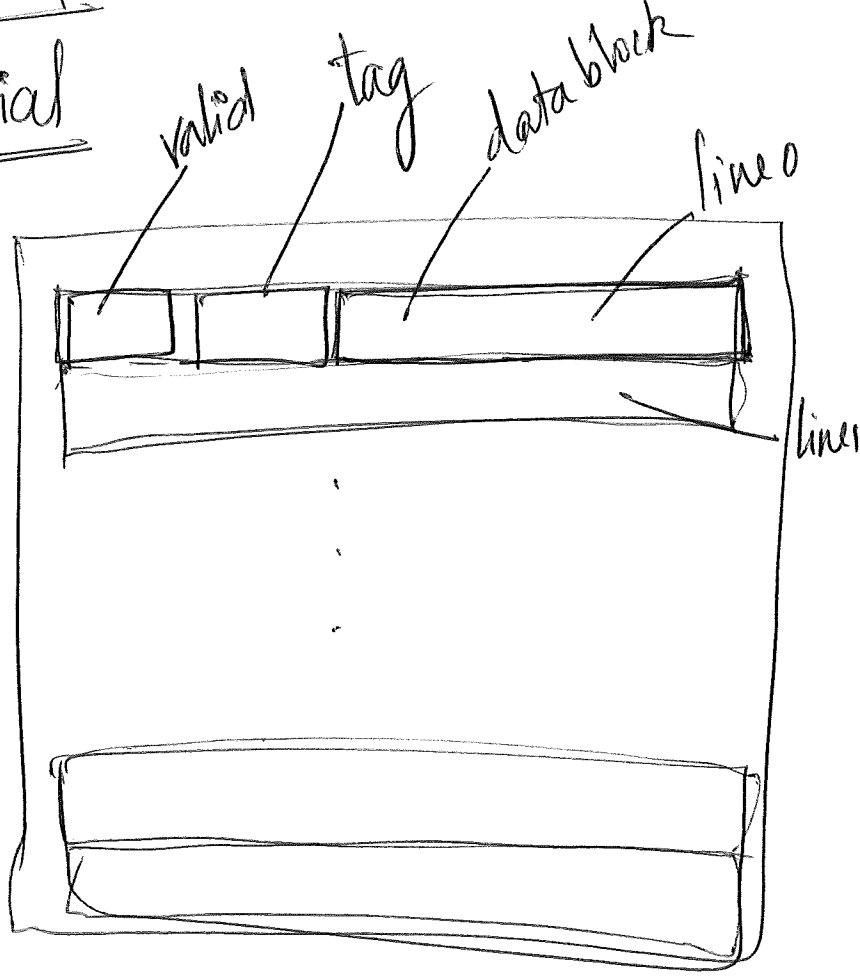


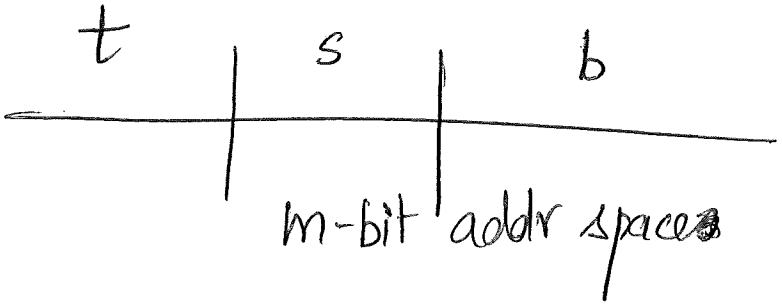


Dynamic Memory Allocation

Caches

- Locality
  - Temporal
  - Spatial
- Organization
  - S_0
    - Direct Mapped Cache
      - E = 1
    - S_3
      - Set Associative Cache
      - Fully Associative Cache
      - Writes





Linking

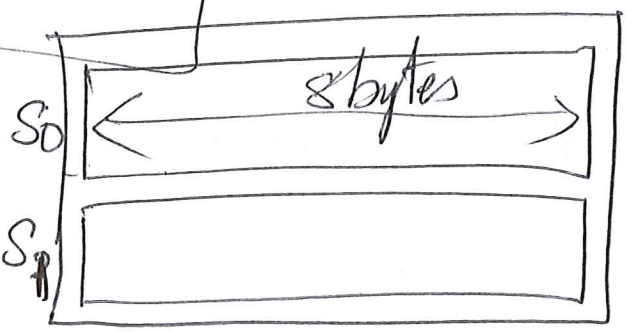
Virtual Memory

Associative Caches

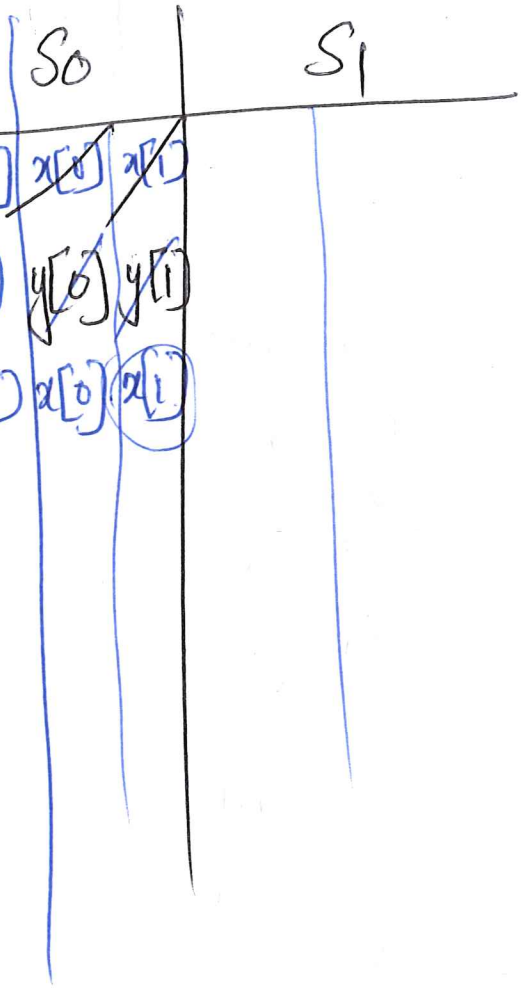
```
for(i=0; i<8; i++)
    sum += x[i] * y[i];
```

cache block size (B) = 8 bytes

Cache size (C) = 16 bytes



Elem	Addr	Set index
x[0] (block 0)	0	0
x[1]	4	0
x[2]	8	1
x[3]	12	1
y[0]	16	0
y[1]	320	0
y[2]	24	1
y[3]	28	1



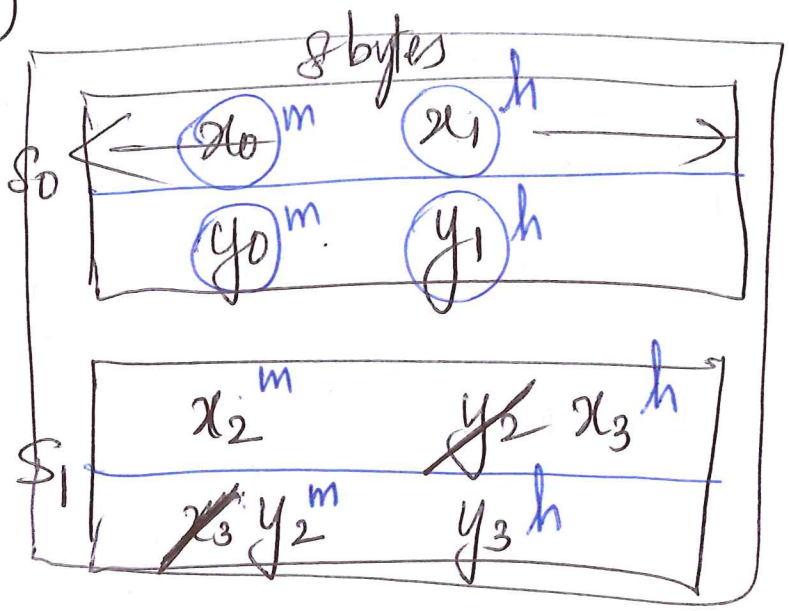
④

$C = 32$ bytes

$B = 8$ bytes

$E = 2$

cache lines



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

E-way set associative cache

2-way " " "

1 byte memory addr.

x_0 m

y_0 m

x_1 h

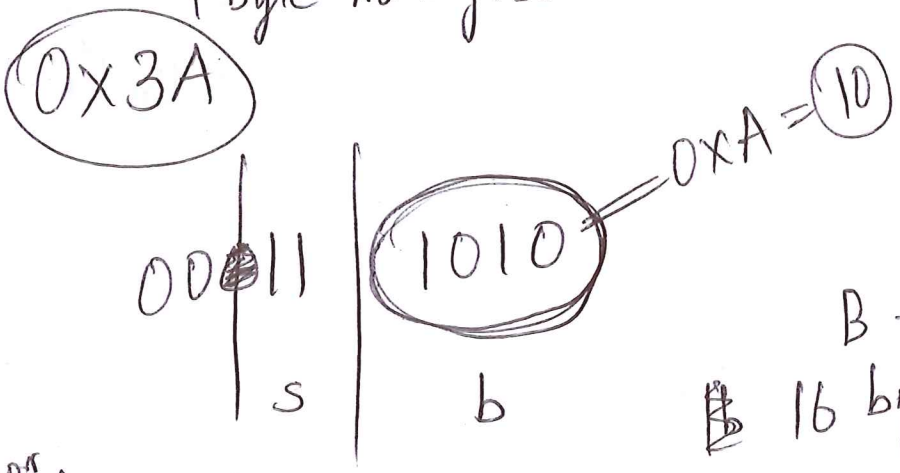
y_1 h

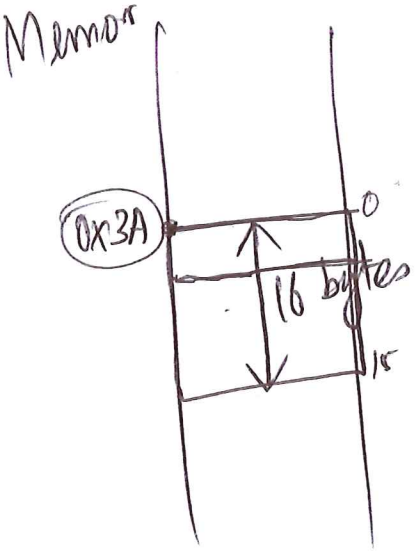
x_2 m

y_2 m

x_3 h

y_3 h





Assume $x_0 \ldots x_7$

$y_0 \ldots y_7$

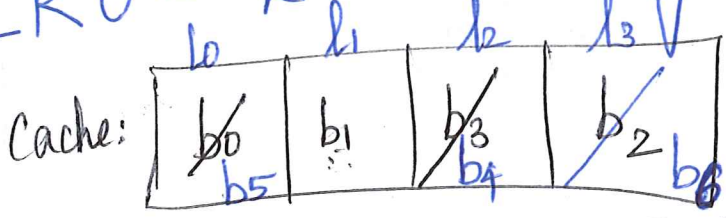
Read x_4 → [x_4 x_5] → set 0

Q: What to replace in set 0?

[x_0 x_1] OR [y_0 y_1]

Cache Replacement Policies

1. Random
2. LRU – Least Recently Used.



1 set
4 lines

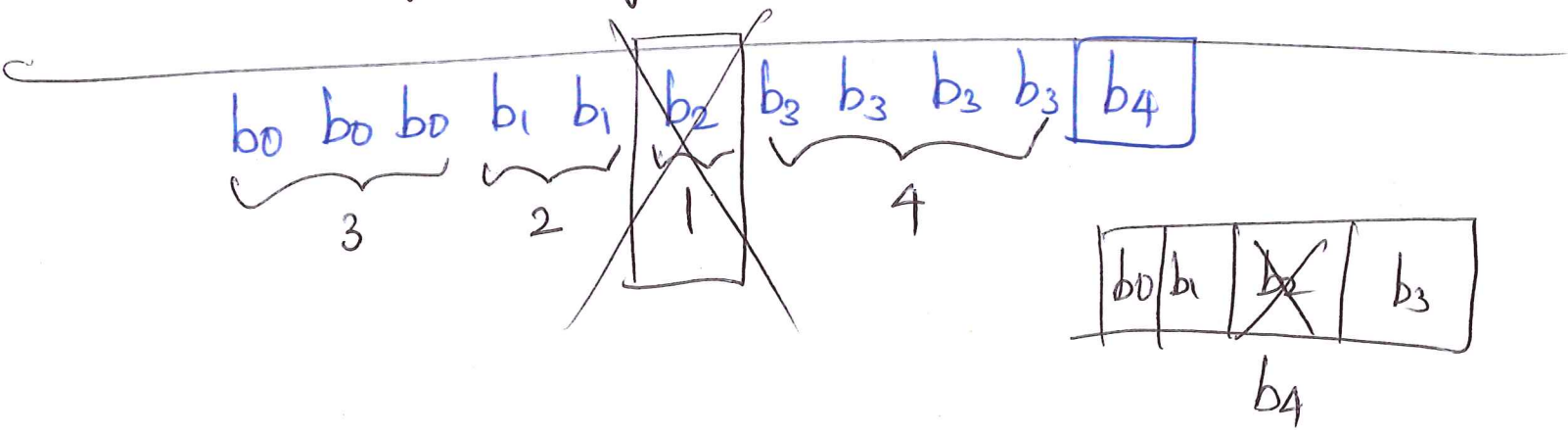
b_0	b_1	b_3	b_1	b_0	b_2	b_1	b_4	b_5	b_4	b_6
m	m	m	h	h	m	h	m	m	h	

* victim block

needs an eviction. victim = b_3.

b_0 b_0 b_0 b_0 b_0 . . . . b_1 b_2 b_3 b_4 b_5

3. ~~Most~~ Least frequently used. (~~MFU~~ LFU)



Fully-Associative Cache.

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

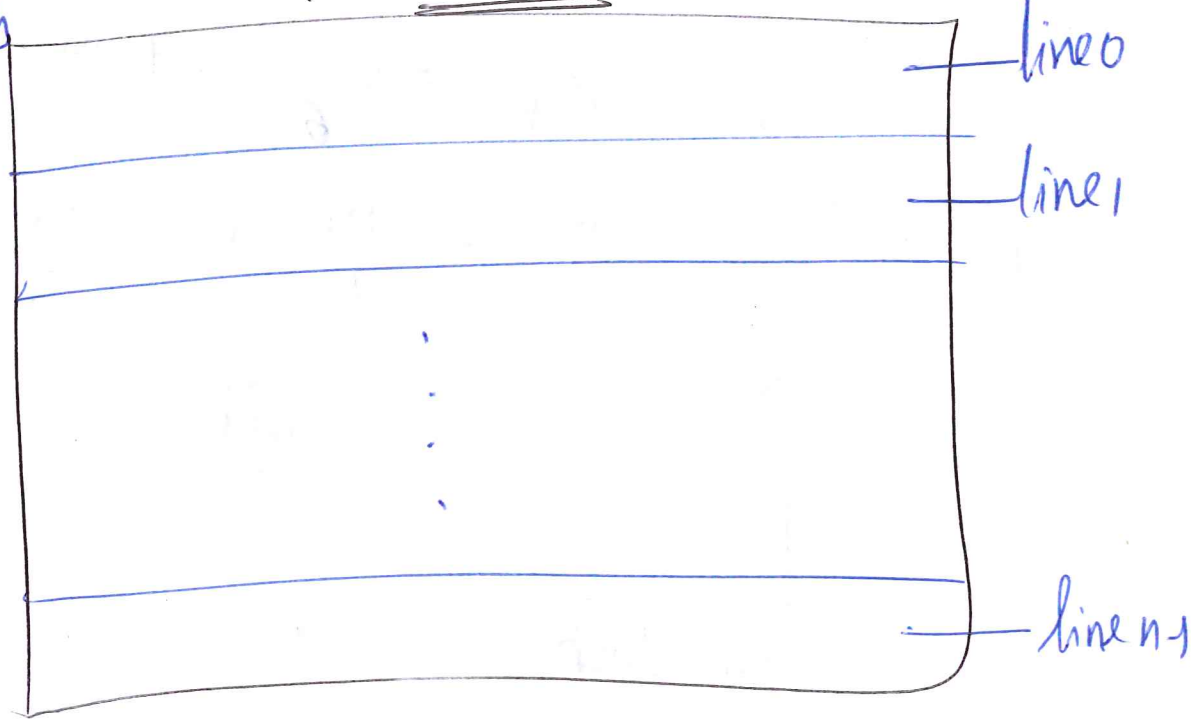
$E = 1$ Direct Mapped

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

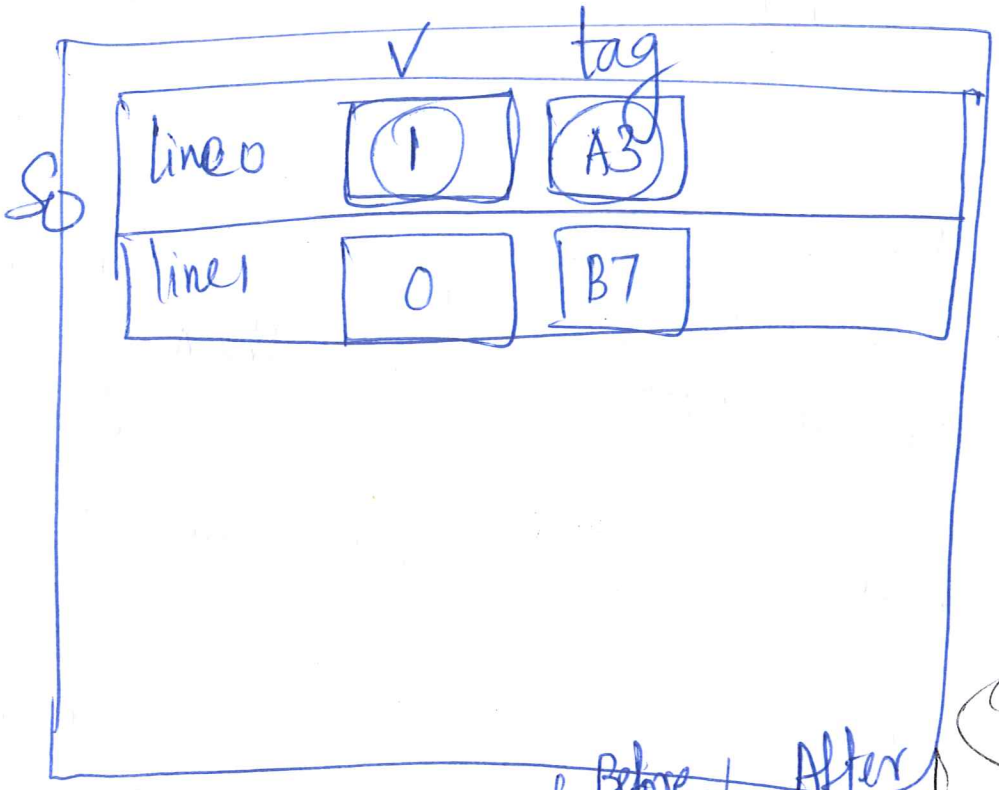
$C = 64$ bytes

$B = 8$ bytes

$E = 64/8 = 8$ lines

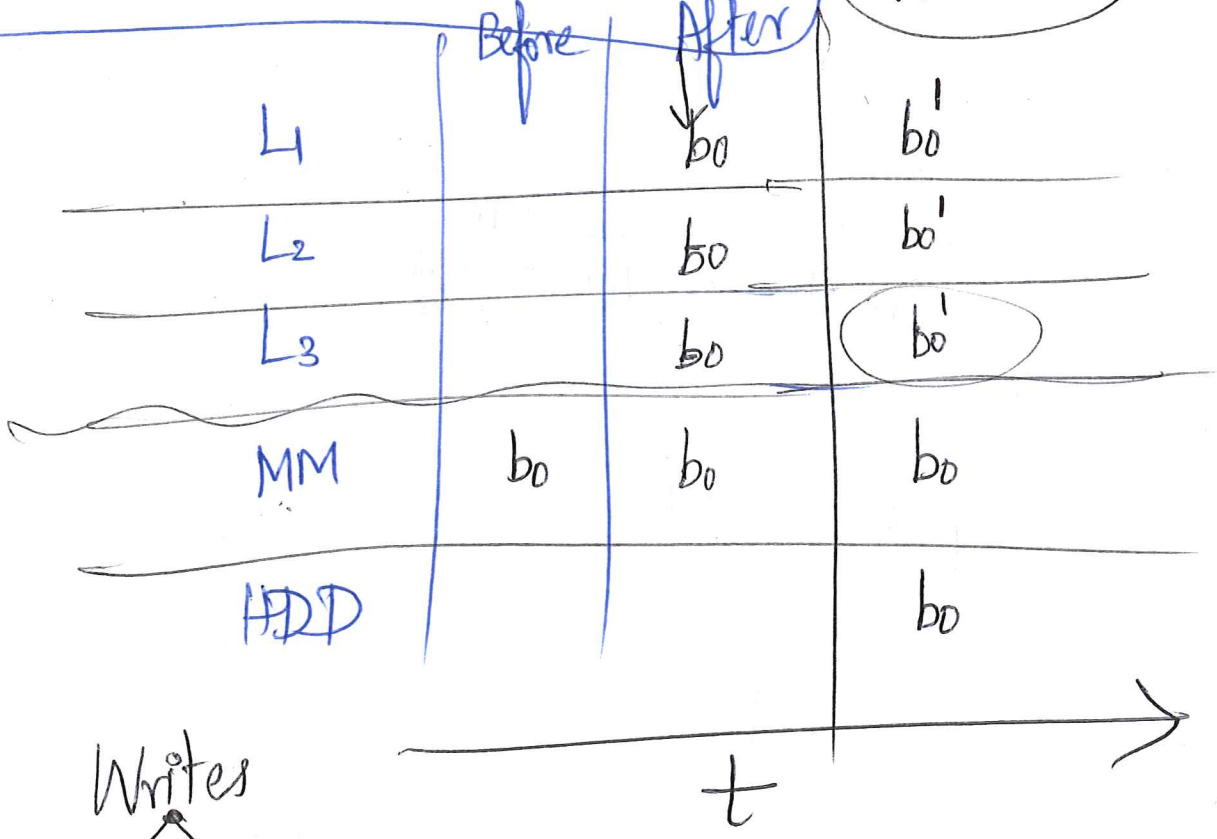


So



Write b_0

Writes



Writes

- Write through
- Write back

(8)

Writes

cache

MM bb

write allocate

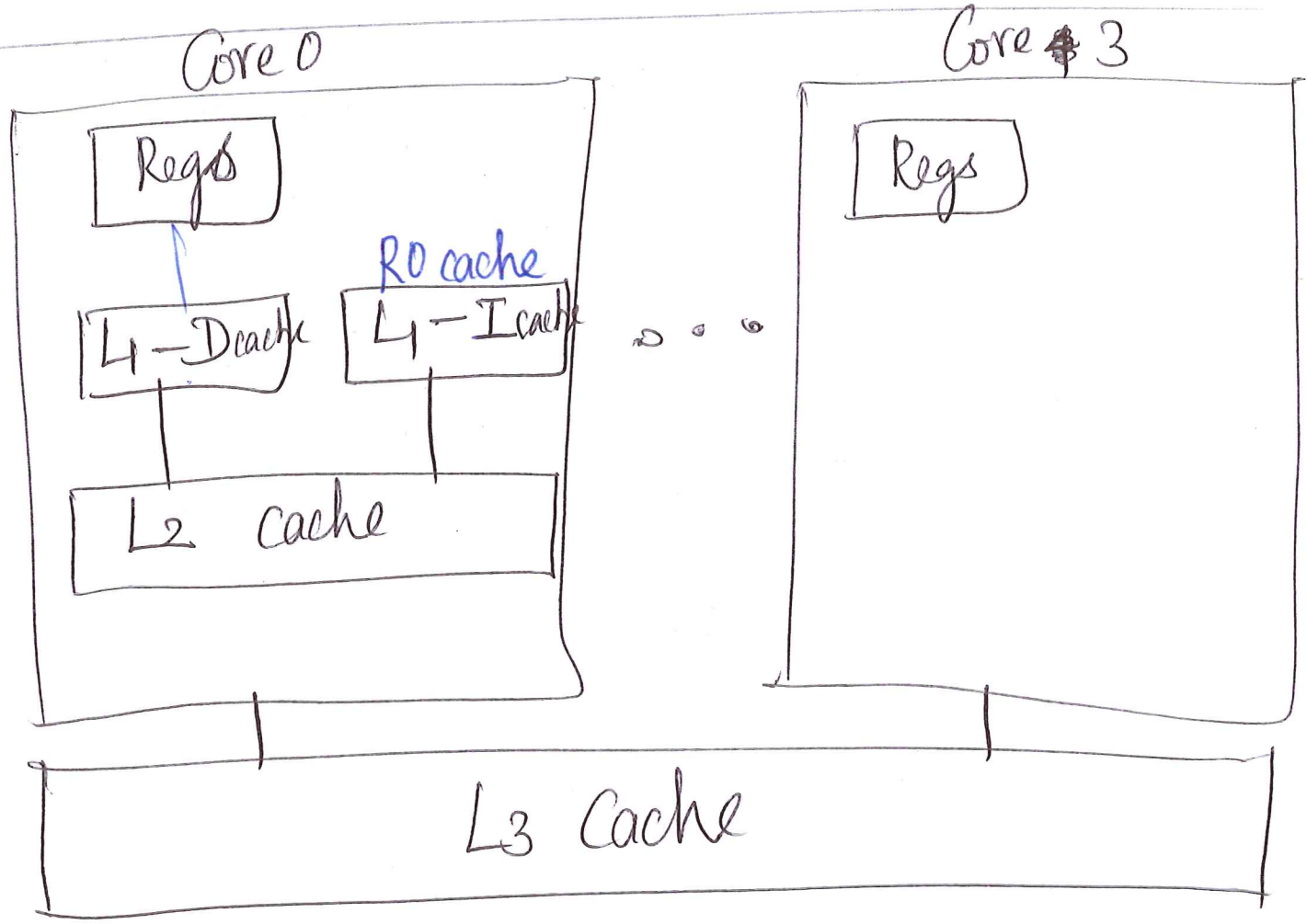
Load block in cache and writes in cache

write back

no-write allocate

write happens directly in main memory

write through caches.



Cache-Friendly.

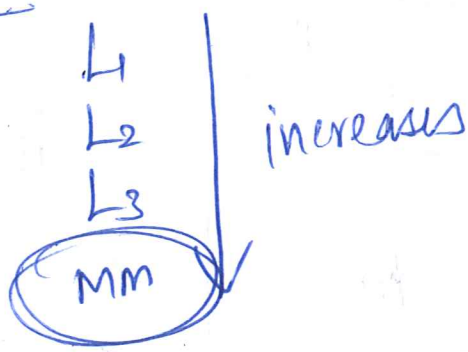
1. miss rate = $\frac{\#\text{misses}}{\#\text{references}}$

2. hit rate = 1 − miss rate.

3. hit time – time to deliver a word in the cache to the CPU.

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

4. miss penalty



transpose (10)

$$M = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad M^T = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

src dst

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

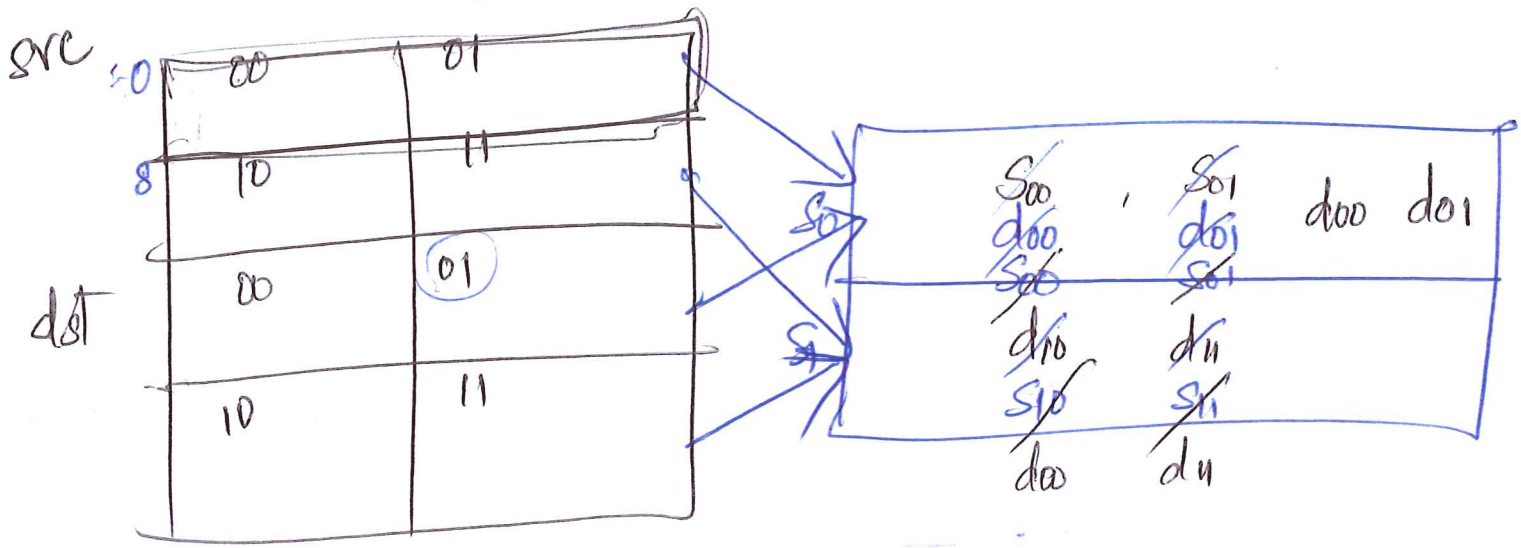
- src starts at 0. dest at addr 16.
- cache - direct mapped, write through, write-allocate.
- with a block size of 8 bytes.
- C = 16 bytes → C = 32 bytes

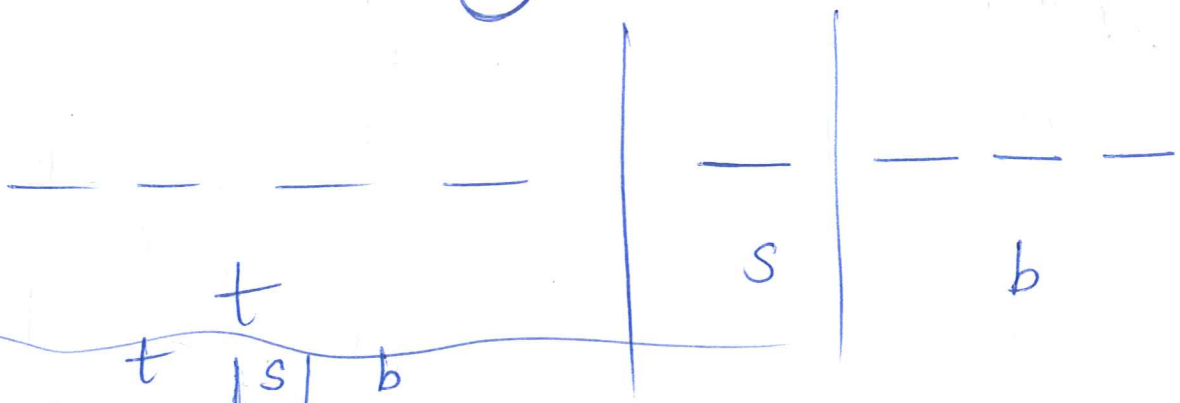
src

	0	1
0	m	m
1	m	h

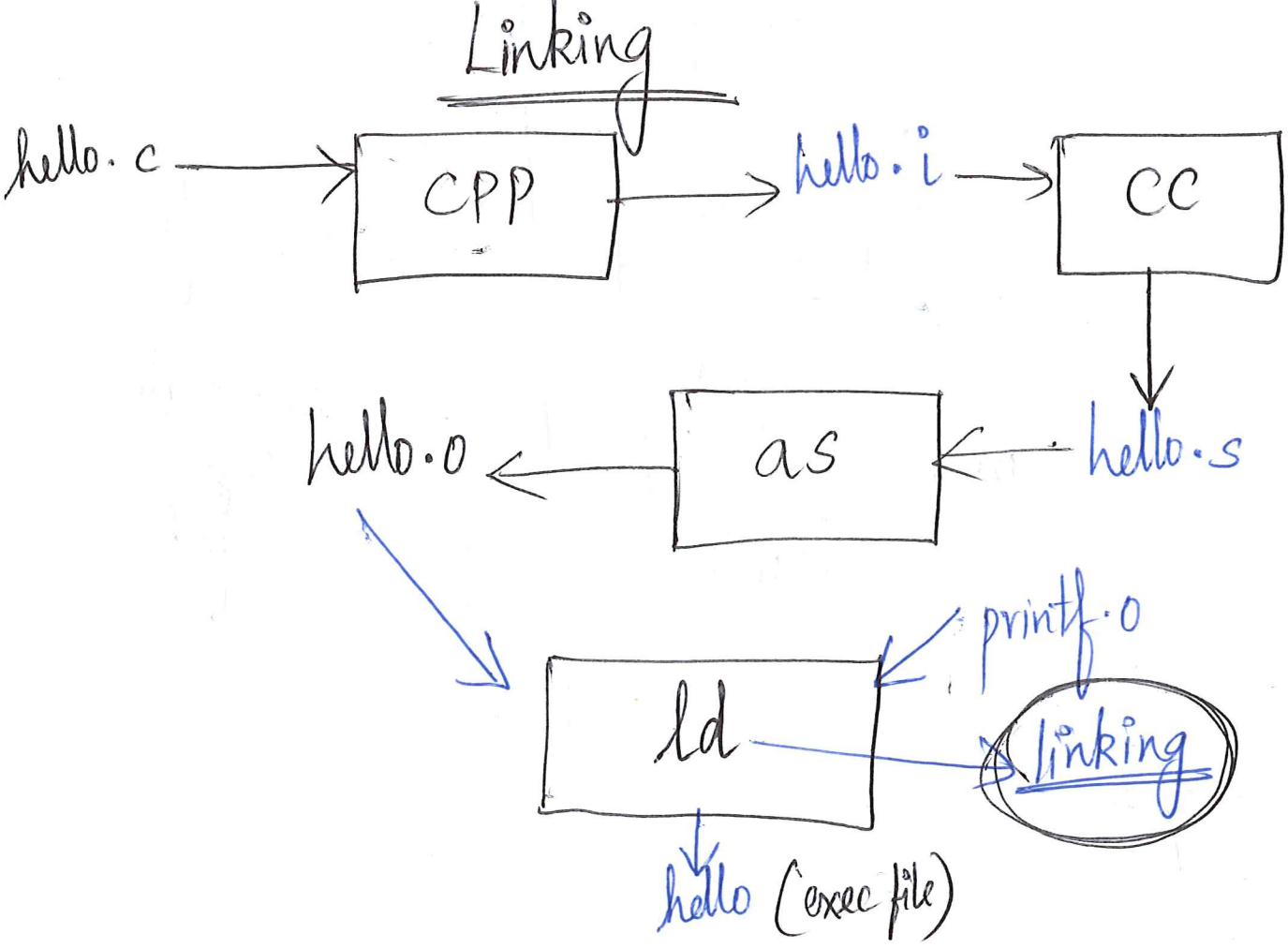
dst

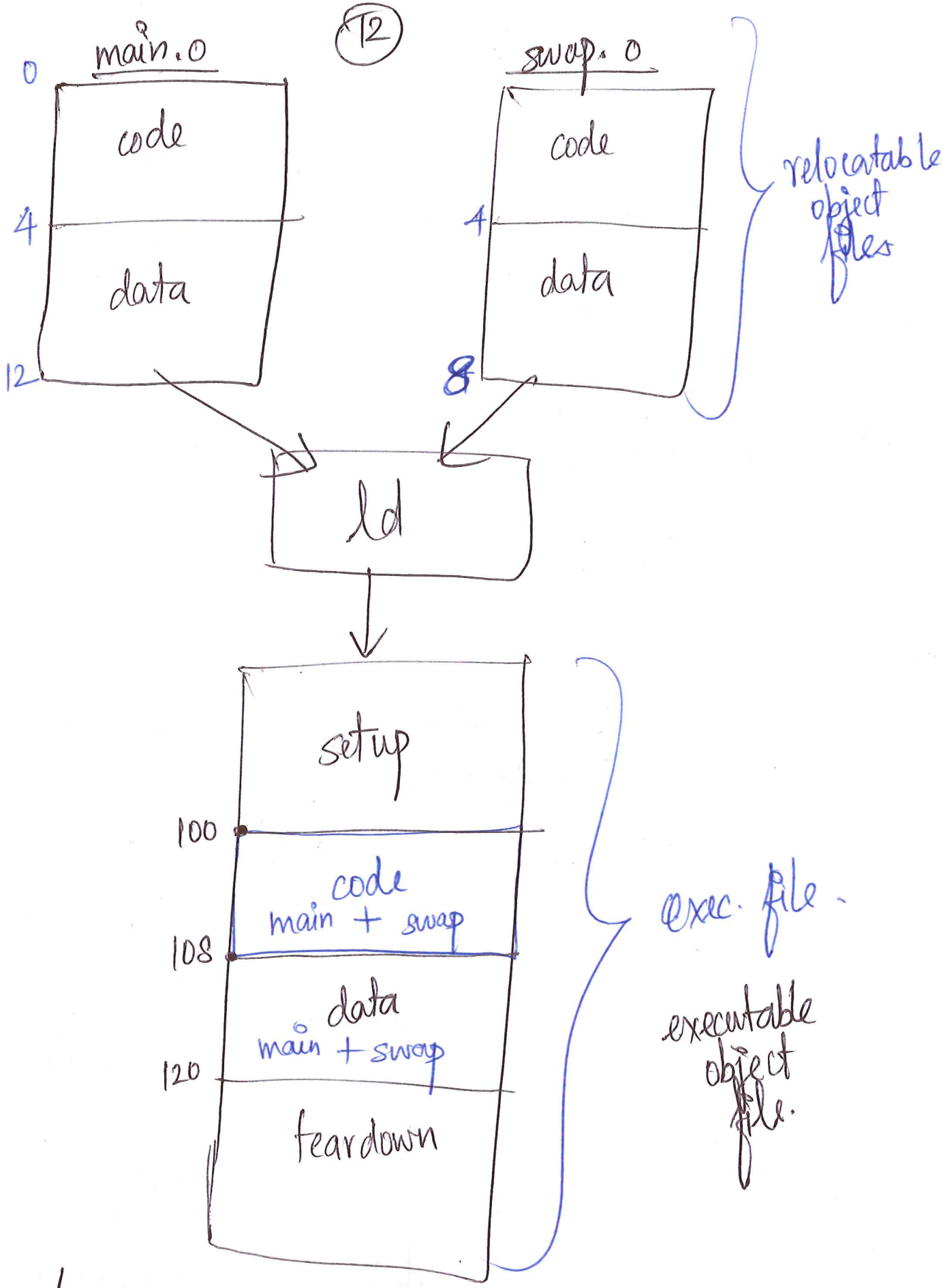
	0	1
0	m	m
1	m	m





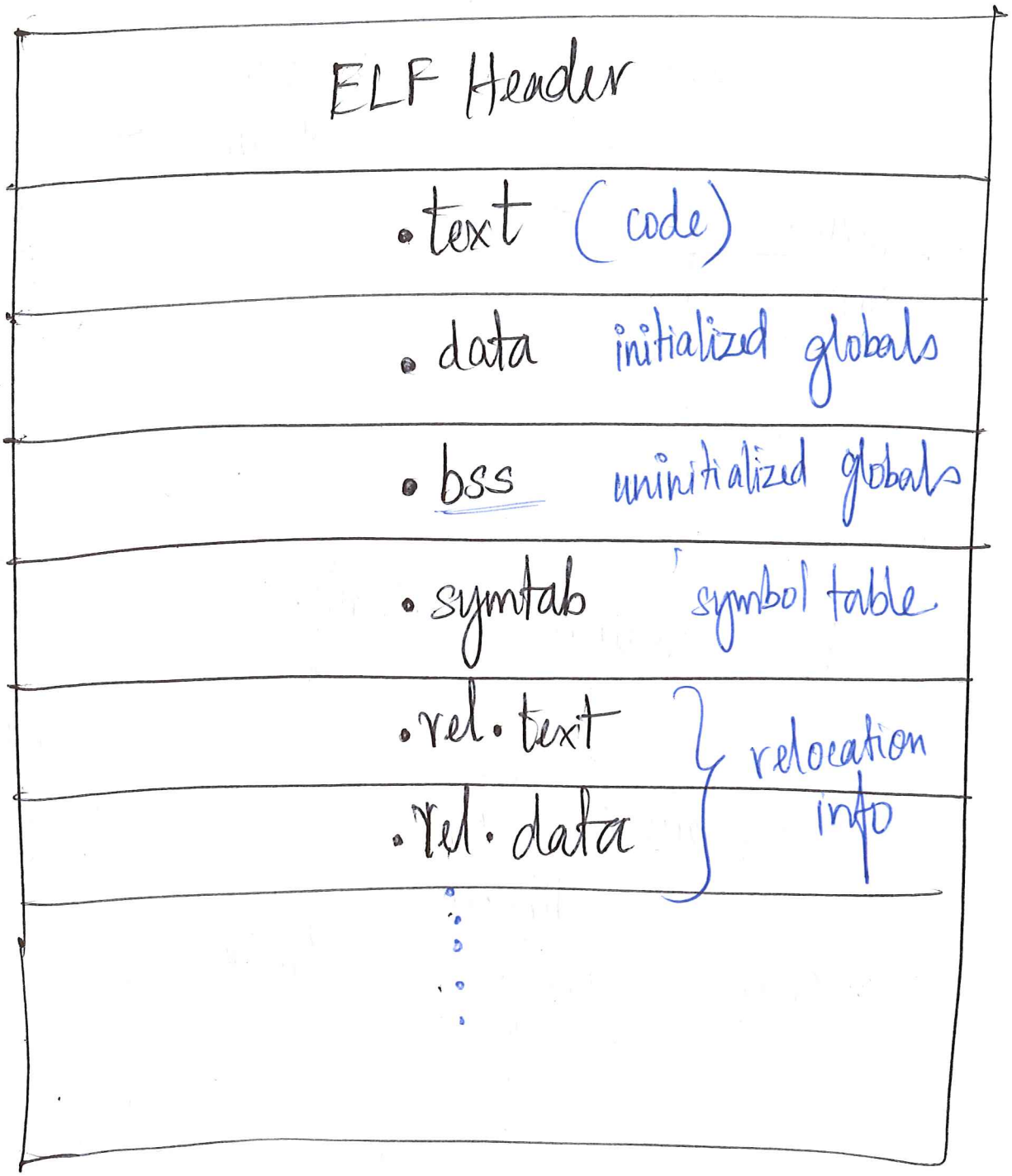
src	t	s	b
0:	0 000	0	000
⋮			
7:	0 000	0	111
8:	0000	1	000
16:	0001	0	000
24:	0001	1	000




main.o
0
code
4
data
12
swap.o
code
4
data
8
relocatable object files
ld
setup
100
code
main + swap
108
data
main + swap
120
teardown
exec. file.
executable object file.

> ./p

ELF - Executable & Linkable Format



(14)

Linking

- Static
  - libc.a
- Dynamic
  - libc.so → shared object
  - DLLs
    - Dynamic Link Libraries

main.c

```
printf()
scanf()
atoi()
```

> ld main.o printf.o ... atoi.o

> ld main.o libc.o

archive

libc.a

- printf.o → object modules
- scanf.o

libc.o

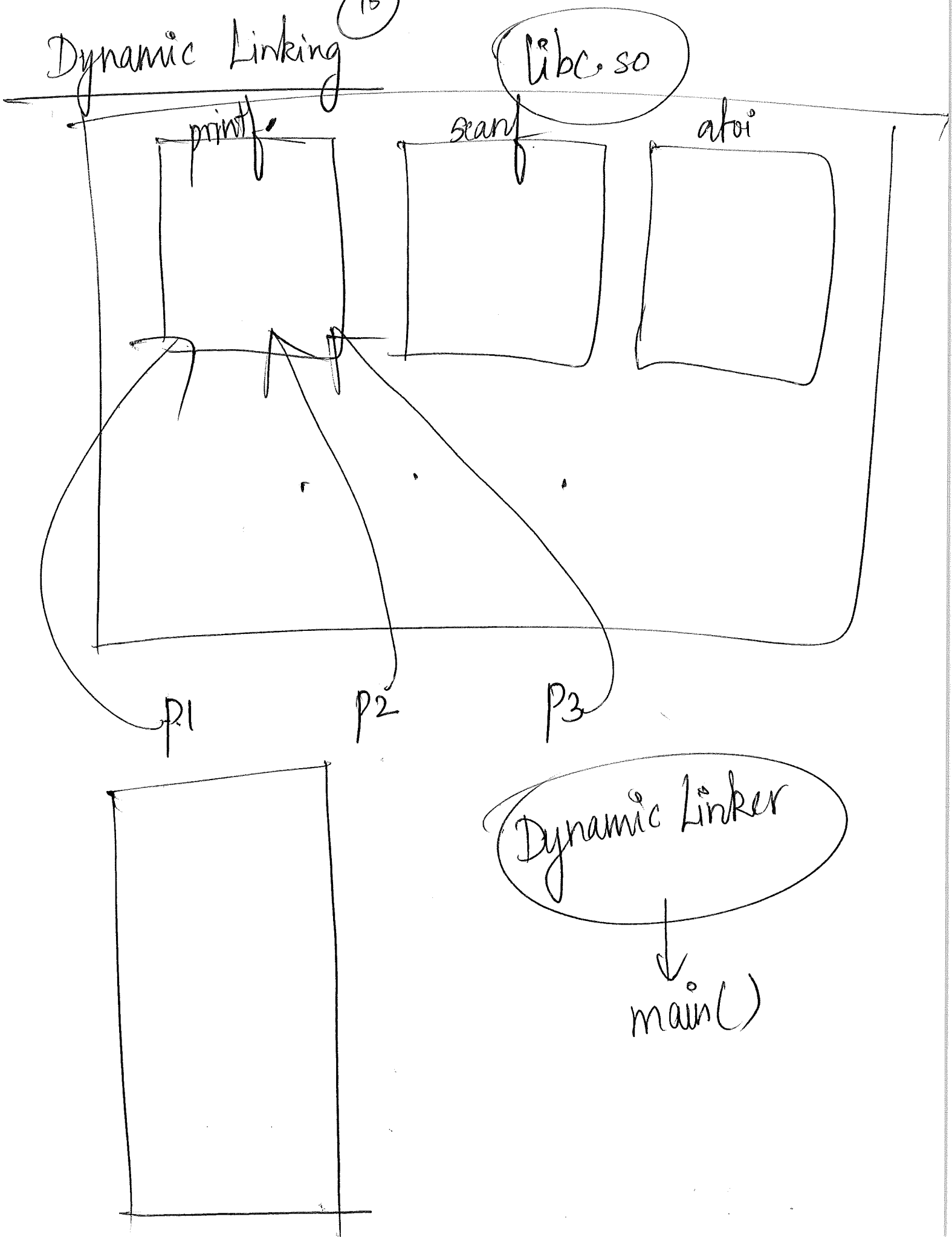
- printf.o
- ...
- 100 do.o

> ld main.o /usr/lib/libc.a

exec

```
main:
printf:
scanf:
atoi:
```

Dynamic Linking



C = S x E x B
= 8 x 4 x 2 = 64 bytes

E = 4

S = 8

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.

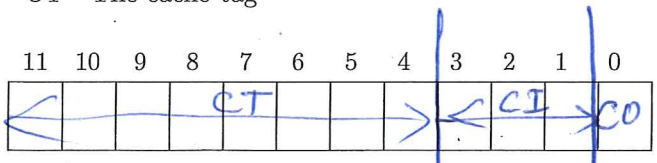
In the following tables, **all numbers are given in hexadecimal**. The contents of the cache are as follows:

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 3 1 1 1 0/1

0xAFE, 0xAFF

tag

0x3AE, 0x3AF

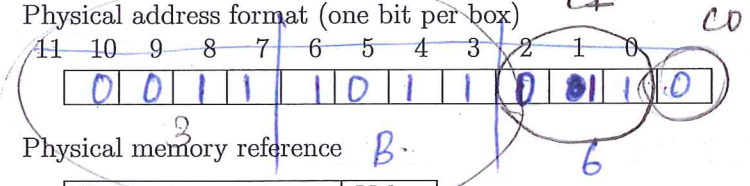
Part 2

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)	0x0
Cache Index (CI)	0x3
Cache Tag (CT)	0x3B
Cache Hit? (Y/N)	Y
Cache Byte returned	0x 66

Part 3

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