1) The cache structure

1) cache direct mapped

64 KiB , 16 Byte blocks.

Byte addressed . 32 bit address

How many bits for tag, index and offset?

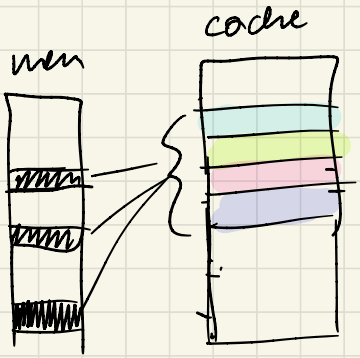
index : 12
offset : 4
tag : 16

offset := 4 bits

$64 \cdot 2^{10}$ lines of cache $\quad 2^6 \cdot 2^{10} = 2^{16}$ $\quad$ 16 bits of index + offset

index = 16 - 4 = 12

2) 4 way set associative instead



the "sets" are 4 less so I take 2 bits from index

index : 10
offset : 4
tag : 18

3) 0 bit because we can put the block in any line and we have to check the entire cache everytime we have to access it.

2) Hit/miss 2-way LRU CACHE

Size: 8 Byte

2-way set associative

LRU replacement.

request

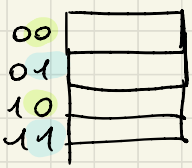
0110, 0000, 0010, 0001, 0011, 0100, 1001, 0000, 1010, 1111, 0111

address	set 0	set 1	H/M
0110	e/e	01/e	M
0000	00/e	01/e	M
0010	00/e	01/00	M
0001	00/e	01/00	H
0011	00/e	01/00	H
0100	00/01	01/00	M
1001	10/01	01/00	M
0000	10/00	01/00	M
1010	10/00	10/00	M
1111	10/00	10/11	M
0111	10/00	01/11	M

Mem

0110
0000
0010
0001
0011
0100
1001
0000
1010
1111
0111

cache



3) cache hit rate

cache direct mapped 32 Byte lines

2048 Byte size

have the following definitions:

```
struct point_color {
    int c;
    int m;
    int y;
    int K;
};
```

```
struct point_color square [16][16]
register int i, j
```

- sizeOf(int) = 4 (32 bit)
- square begins at address 0
- the cache is initially empty
- the only memory accesses are from entries to the square, ignore i and j
- addresses are 32 bit

what is the percentage of hits generated by the following snippets?

32 B lines so 8 ints per line

for each row we have 16 struct with 4 addresses inside each

therefore:

$$addr(sqr[i][j]) = 64i + 4j + \begin{cases} 0 & c \\ 1 & m \\ 2 & y \\ 3 & K \end{cases}$$

the determine in wich order the addresses are visited

Remember: I load 8 blocks per time

1)
```
for (i=0; i<16; i++){
    for (j=0; j<16; j++){
        sqr[i][j] c = 0
        sqr[i][j] m = 0
        sqr[i][j] y = 1
        sqr[i][j] K = 0
```

index	[0][0]	[0][0]	[0][0]	[0][0]	[0][1]	[0][1]	[0][1]	[0][1]	[0][2]	[0][2]
color	c	m	y	K	c	m	y	K	c	m
address	0	1	2	3	4	5	6	7	8	9
Hits	M	H	H	H	H	H	H	H	M	H

↑ when I get the miss I load the next 8 addr — they are already in cache

7/8 hit rate = 87,5 %

2)
```
for(i=0; i<16; i++){
    for(j=0; j<16; j++){
        sqr[i][i] c = 0
        sqr[i][i] m = 0
        sqr[i][i] y = 1
        sqr[i][i] K = 0
    }
}
```

$64i+4j+\begin{cases}0 & c\\ 1 & m\\ 2 & y\\ 3 & K\end{cases}$

index	[0][0]	[0][0]	[0][0]	[0][0]	[1][0]	[1][0]	[1][0]	[1][0]	[2][0]	[2][0]
color	c	m	y	K	c	m	y	K	c	m
address	0	1	2	3	64	65	66	67	128	129
Hits	M	H	H	H	M	H	H	H	M	H

3/4 Hit 75% Hit rate

3)
```
for(i=0; i<16; i++){
    for(j=0; j<16, j++){
        sqr[i][i] . y=1
    }
}
```

index	[0][0]	[1][0]	[2][0]	[3][0]
color	y	y	y	y
address	2	66	130	194
H	M	M	M	M

Hit rate 0%

4)
```
for(int i=0; i<16; i++){
    for(int j=0; j<16; j++){
        sqr[j][i] . c=0
        sqr[j][i] . m=0
        sqr[j][i] . K=0
}}
```

index	[0][0]	[0][0]	[0][0]	[1][0]	[1][0]	[1][0]	[2][0]	[2][0]	[2][0]
color	c	m	K	c	m	K	c	m	K
address	0	1	3	64	65	67	128	129	131
H	M	H	H	M	H	H	M	H	H

2/3 Hit 66,7 % Hit rate

4) Cache replacement

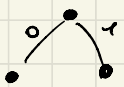
1) w_i most recently used word for $i=1, \dots k$ k = set size

With LRU only one word w_k can be replaced.

With pseudo-LRU which words can be replaced?

MRU - every w_i except w_1

Tree based - you cannot replace through w_1 to $w\log_2 k$. All $\log_2 k$ bits on the path to the element to replace X, X is only replaced when they all 0 again

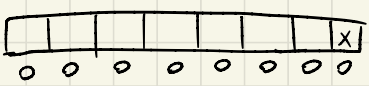


max 1 bit can be set to 0 per access.

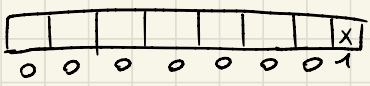
when passing a 1 the path to X is not followed any further

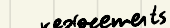
2) MRU

start

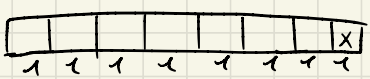


requested x

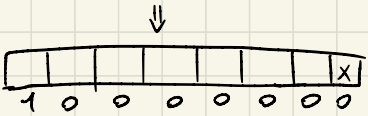




replace others



K-1 replacements



K-2 replacements before replacing X

$2K-3$ total replacements

tree based:

if $K=2$ it's LRU so maximum 1

if $K>2$ and I ask each time a "neighbor" of x it will never be replaced

5) Access time and hit rates

T_1 = access time to element in L1 cache

if not present in L1 cache then T_2 = time to tranfer to L1 cache

T_m → time to move from memory to L2 cache

suppose: $T_1 = 20ns$ $T_2 = 100ns$ $T_m = 400ns$

1) if we have hit rate = 0.9 L1 what is the average access time?
= 0.5 L2

90% 20ns $\quad 20 \cdot 0.3 = 18 +$

10% { 50% 100+20 ns $\quad 120 \cdot 0.05 = 6 +$

50% 400+100+20 $\quad 520 \cdot 0.05 = 26 +$

50 ns

2) L2 hit rate 0.6 and average time 33 ns
L1 hit rate = ?

$0 < p < 1$

p% 20

100-p% { 60% 100+20

40% 400+100+20

$p \cdot 20 + (1-p)(0.6(120) + 0.4(520)) = 33$

$20p + (1-p)(72 + 208) = 33$

$20p + (1-p)(280) = 33$

$20p + 280 - 280p = 33$

$-260p = 33 - 280$

$260p = 247$ $\quad p = \frac{247}{260} = 0,95$

Direct mapping vs set associative with random replacement

consider the following snippet:

```
float a[1024], b[1024]
for (i=0; i<1024; i++)
    sum += a[i] * b[i]
```

- cache size: 4 KByte
- Block size: 32 Byte >> 8 words per block
- word size: 4 Byte
- float: 4 Byte
- b is right after a in memory
- i and sum are registers

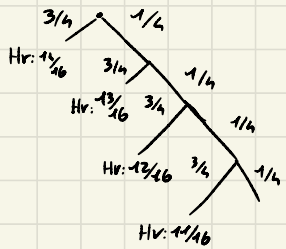
1) what is the hit rate if we assume direct mapping?

0% because if they are one right after the other they have the same index and then overwrite every time

2) 4 way set ass. random replacement, calculate circa the hit rate

every time one is inserted the other is replaced

left -> both are in cache
right -> one replaces the other



$$\sum_{i=0}^{} \frac{3}{4}\left(\frac{1}{4}\right)^{i} \cdot \frac{14-i}{16}$$

$$\left(\frac{3}{4}\cdot\frac{14}{16}\right)+\left(\frac{3}{4}\cdot\frac{1}{4}\cdot\frac{13}{16}\right)+\left(\frac{3}{4}\cdot\frac{1}{16}\cdot\frac{12}{16}\right)+\left(\frac{3}{4}\cdot\frac{1}{64}\cdot\frac{11}{16}\right)$$

$$= \frac{21}{32} + \frac{39}{256} + \frac{36}{1024} + \frac{33}{4096} =$$

$$= \frac{128(21)+16(39)+4(36)+33}{4096} = \frac{3489}{4096} = 0.85$$

85%

3) Same question but with array length = 256 and 1536

for 256 they are 1/4e of the cache apart corresponding elements are mapped to the same set

for 1536 they are 6/4e of the cache apart, that is a multiple of 4/4e = 1/4e, therefore as before, corresponding elements same set.

with direct mapping no more thrashing, so 7/8

with 4-way the same