# Example

```
int A[16][128]={all values};
int B[16][128]={all values};
int C[16];

int main() {
        int tmp,i,j ;

        for (i=0;i<16;i++) {
                tmp=0;
                for (j=0;j<128;j++)
                        tmp+=A[i][j]*B[i][j];
                C[i]=tmp;
        }
return 0;
}
```

# Example: 1) Memory accesses

```
int main() {
  int tmp,i,j ;

  for (i=0;i<16;i++) {
       tmp=0;
       for (j=0;j<128;j++)
       tmp+=A[i][j] * B[i][j];
       C[i]=tmp;
       }
  return 0;
}
```

```
ldr     r9, [r8, r1, asl 2]
ldr     r4, [r7, r1, asl 2]
mla     r0, r4, r9, r0
add     r1, r1, #1
cmp     r1, #128
bne     lni-j
add     r5, r5, #1
cmp     r5, #16
str     r0, [r6, #4]!
bne     lni-i
```

Memory reads: 128+128

Memory write

Memory accesses

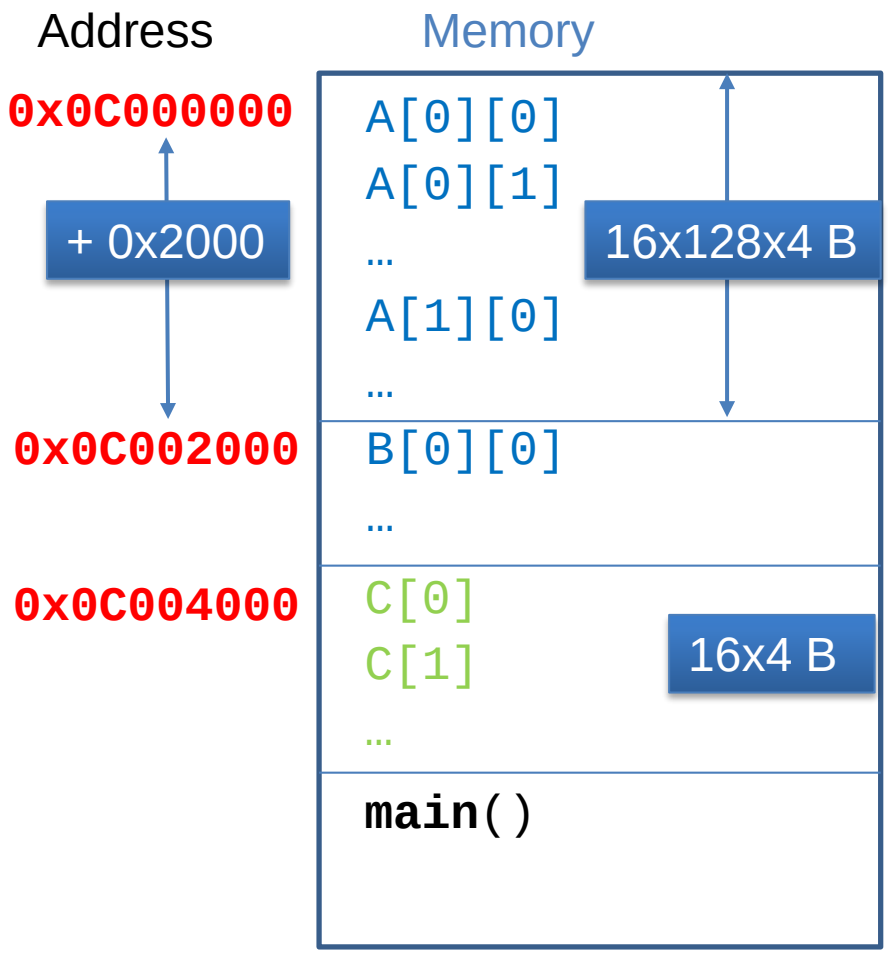
Accesses for fetching instructions are not considered in this example.

# Example: 2) Memory layout

```
int A[16][128]={values};
int B[16][128]={values};
int C[16];
int main() {
        return 0; }
```

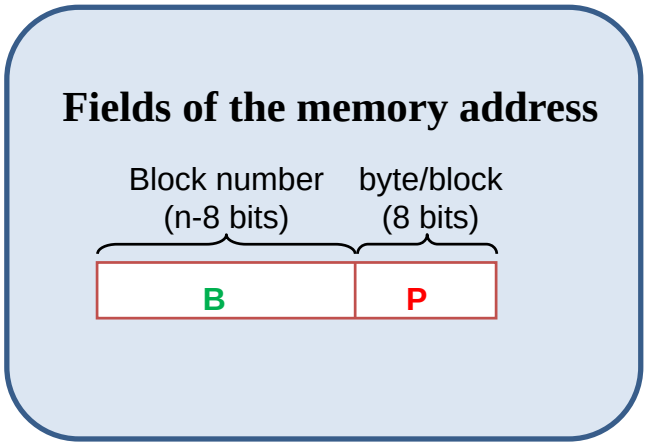
**Linker script**

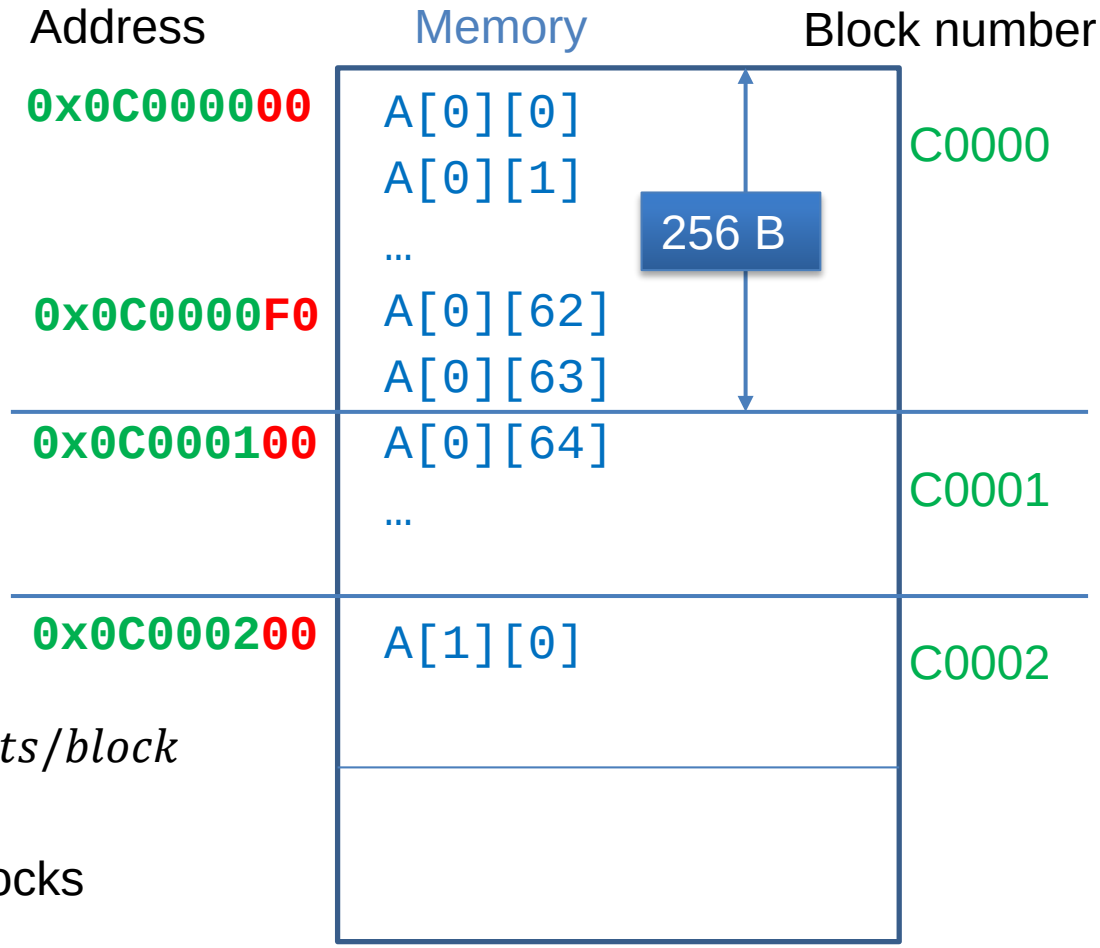
```
SECTIONS {
. = 0x0C000000;
.data : {
*(.data)
.bss : {
*(.bss)
.text : {
  *(.text) }
```



# Example: 3) Memory blocks

**Assume a cache memory with block of 256B**



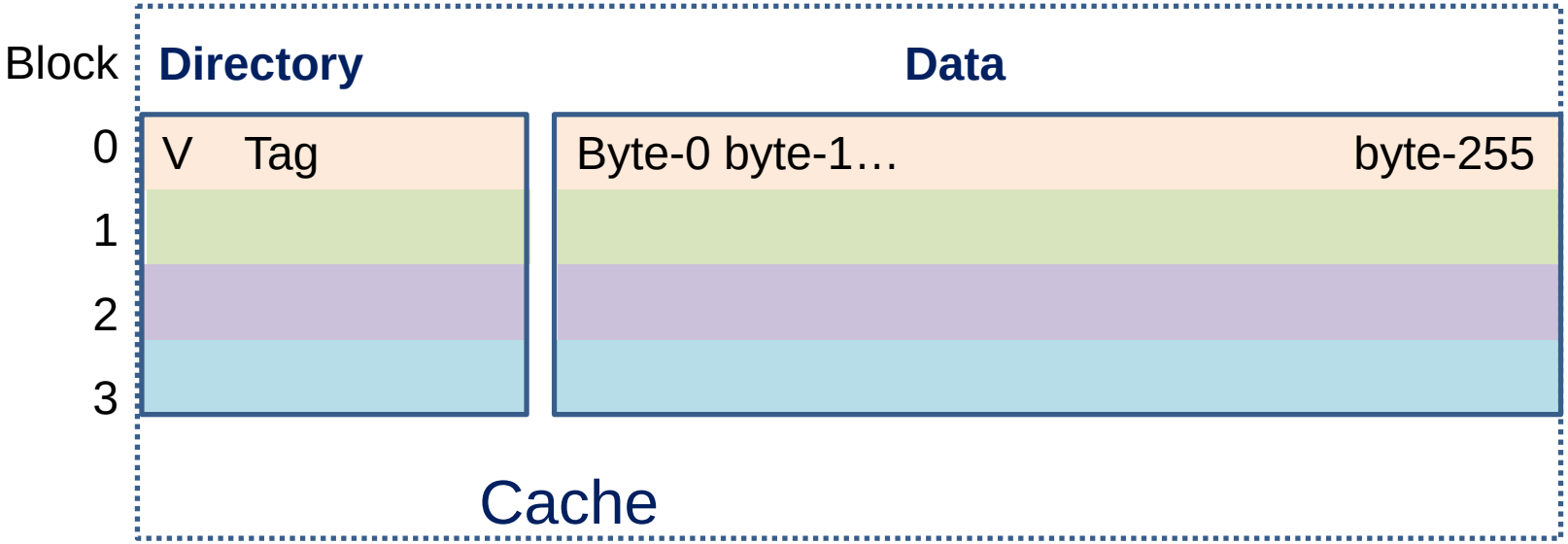

$$\frac{256\ bytes/block}{4\ bytes/element} = 64\ elements/block$$

Matriz A needs 32 memory blocks

# Example: 4) Cache structure

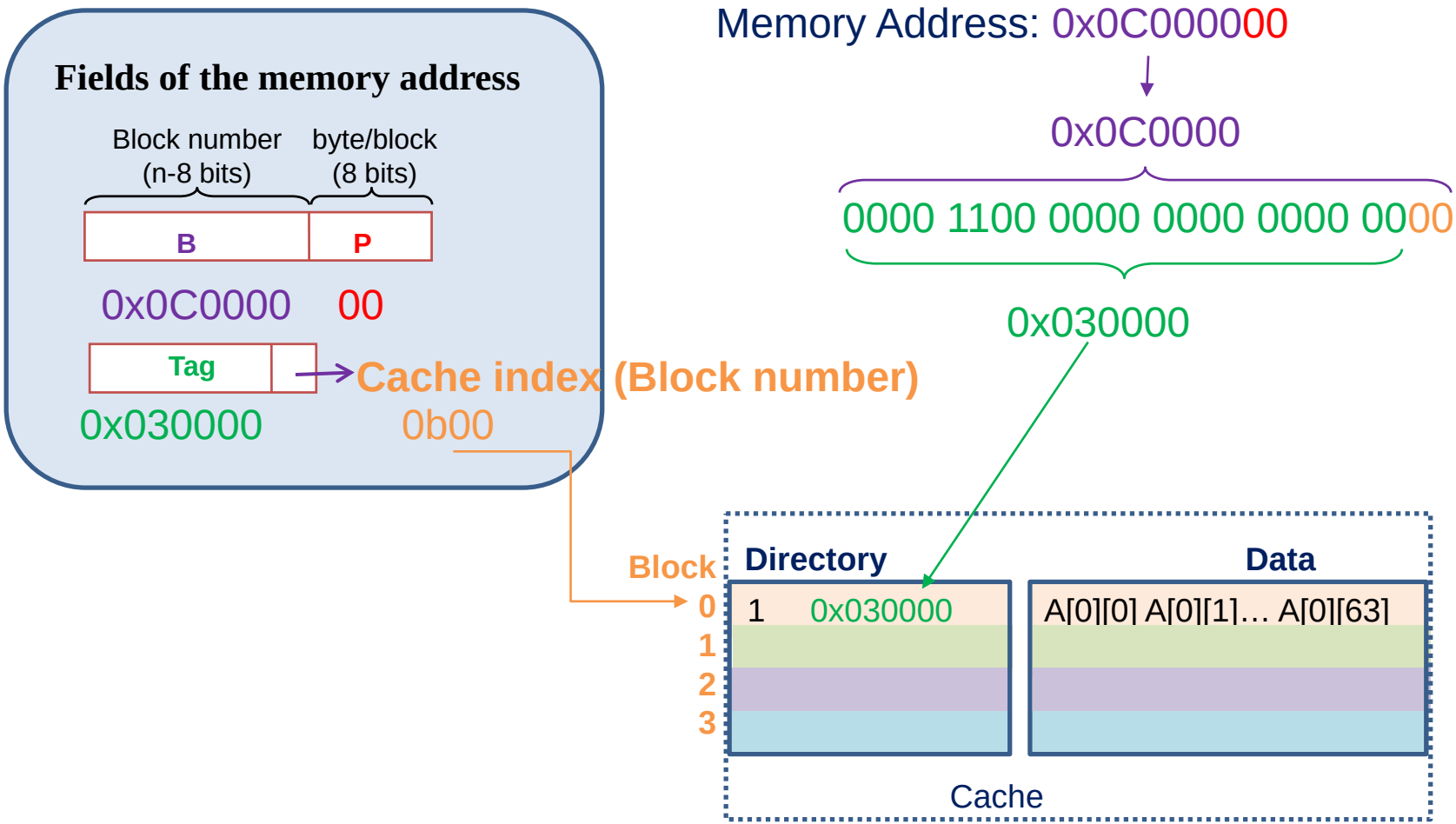
**Assume a cache memory of 1KB with block of 256B**

$$\frac{1024\ bytes}{256\ bytes/block} = 4\ blocks$$



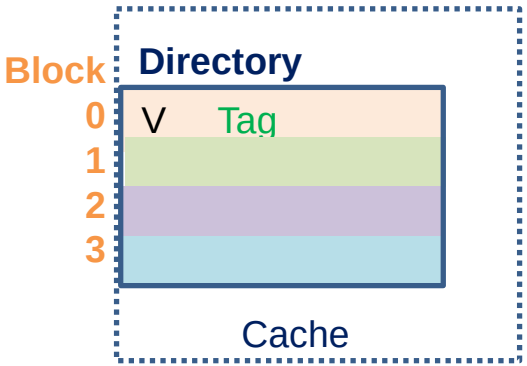
# Example: 5) Direct mapped cache

**Assume a cache memory of 1KB with block of 256B**
**Where in the cache is A[0][0] stored?**



# Example: 5) Direct mapped cache (2)

**Assume a cache memory of 1KB with block of 256B**
**Cache block for each data**




# Example: 6) Cache hit

**Assume we want to read A[0][1], and the cache contents are:**

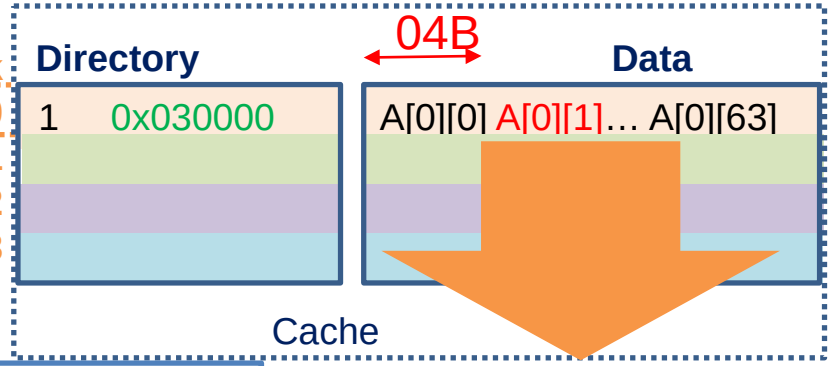
Memory Address: 0x0C000004

1. Directory search:
   0x0C0000

0000 1100 0000 0000 0000 0000

0x030000

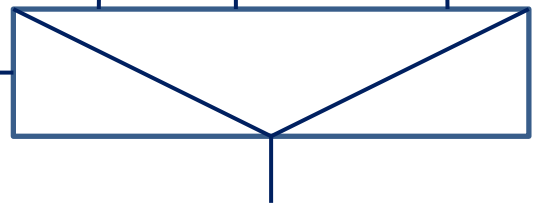
Block




2.Cache hit because:
- V=1 and
- Tags are equal.

3. Read block 0

4.Select the word: 04



# Example: 7) Cache miss

**Assume we want to read B[0][0], and the cache contents are:**

Memory Address: 0x0C002000

1. Directory search: 0x0C0020

0000 1100 0000 0000 0010 0000

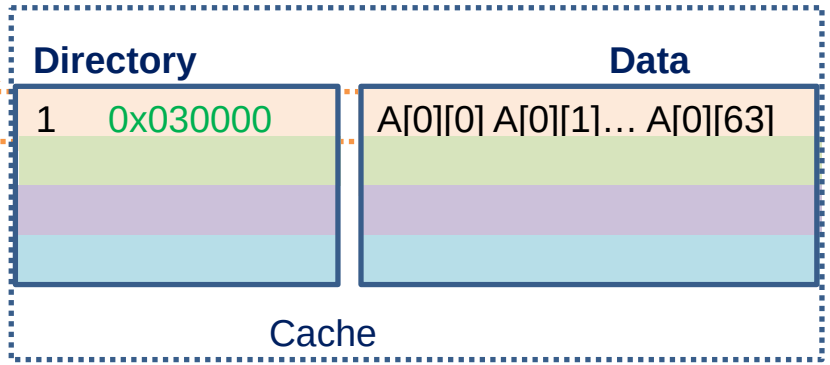
0x030008

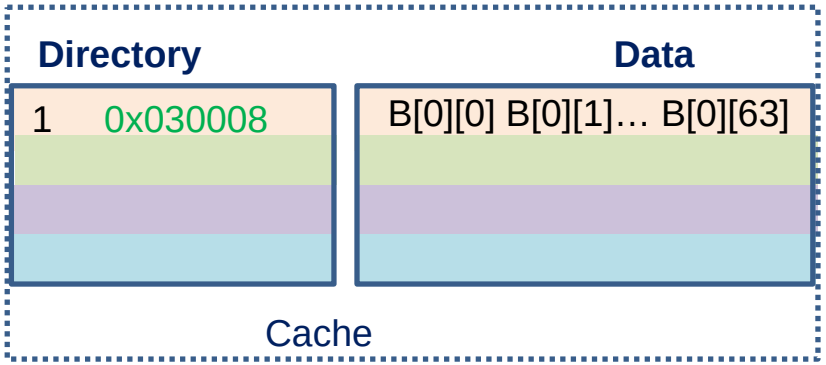
2. Cache miss

3.The block has to be fetched from memory

4. Read block 0

5.Select the word: 00



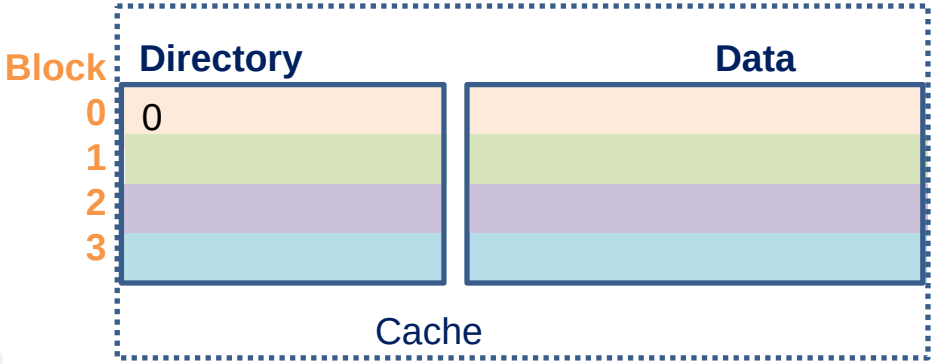
# Example: accesses in a direct mapped cache (1)

```
for (i=0;i<16;i++) {
  tmp=0;
  for (j=0;j<128;j++)
       tmp+=A[i][j] * B[i][j];
  C[i]=tmp; }
```

i=0,j=0

1. Read A[0][0]
   Memory Address: 0x0C000000
   Block: 0
   Tag: 0x030000

Cache miss (v=0)




Cold miss or compulsory miss:
First Access to the block

# Example: accesses in a direct mapped cache (2)

i=0,j=0

1. Read A[0][0]

2. Read B[0][0]

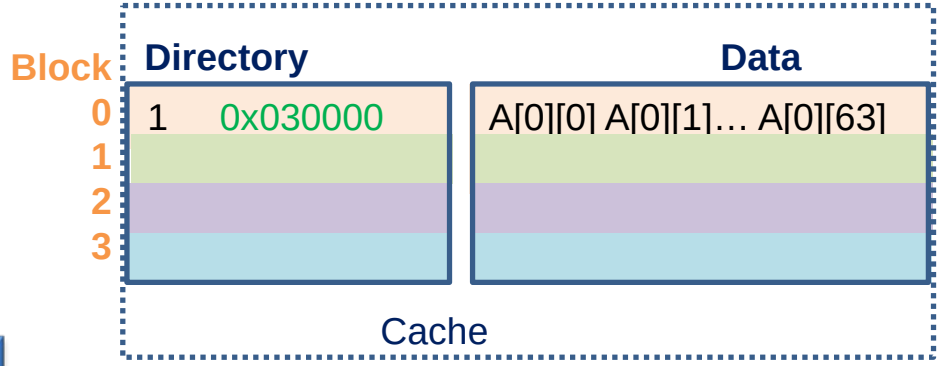
   Memory Address: 0x0C000000

   Block: 0

   Tag: 0x030008

```
for (i=0;i<16;i++) {
  tmp=0;
  for (j=0;j<128;j++)
       tmp+=A[i][j] * B[i][j];
  C[i]=tmp; }
```

Cache miss (different tags)



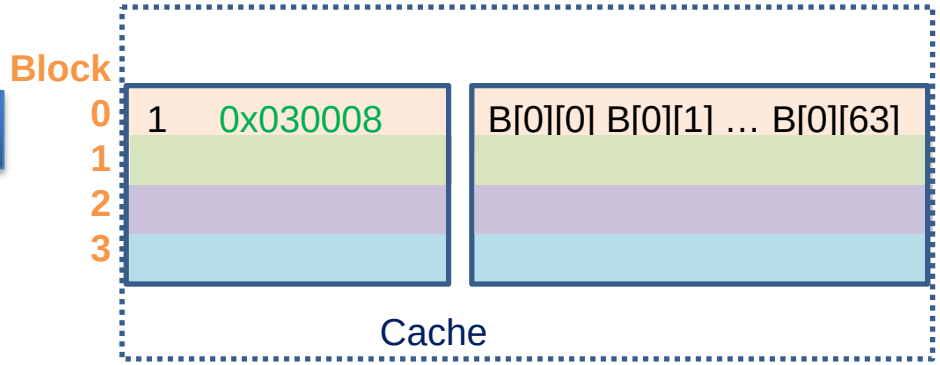

Conflict:
2 memory blocks are mapped to the same cache block

# Example: accesses in a direct mapped cache (3)

i=0,j=1
1. Read A[0][1]
Memory Address: 0x0C000004
Block: 0
Tag: 0x030000

```
for (i=0;i<16;i++) {
  tmp=0;
  for (j=0;j<128;j++)
       tmp+=A[i][j] * B[i][j];
  C[i]=tmp; }
```

Cache miss (different tags)




Internal loop
Due to conflicts, all reads are misses: 128x2 read misses

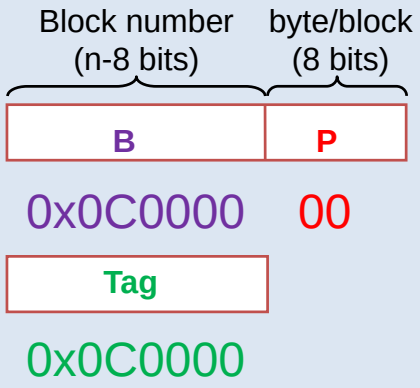
# Example: 8) Fully associative cache

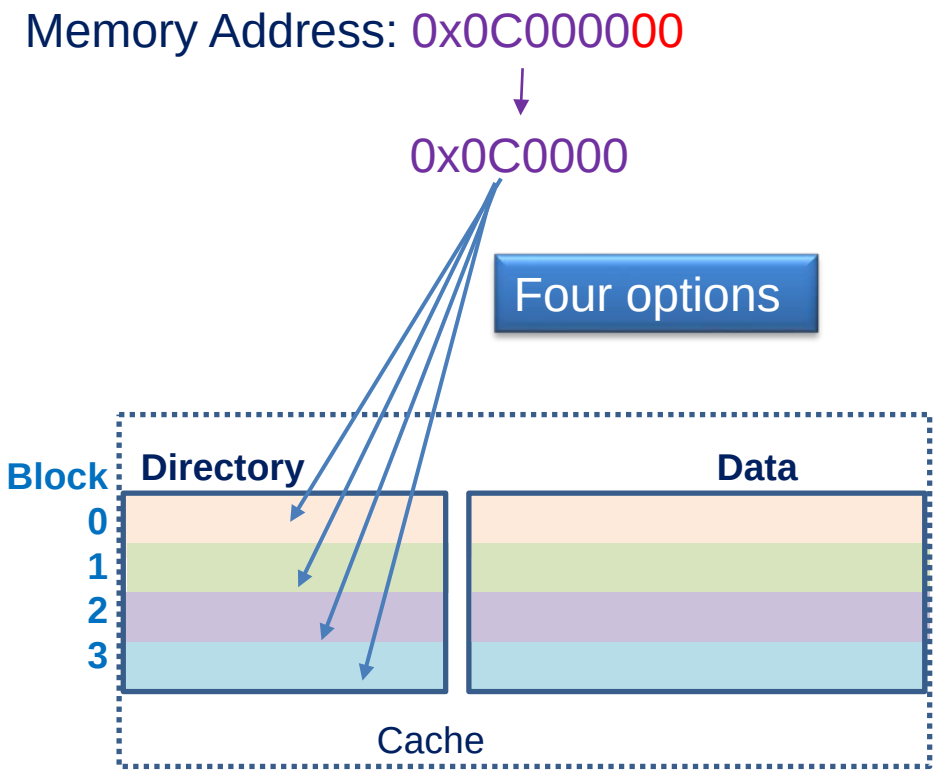
**Assume a cache memory of 1KB with block of 256B and fully associative**

**Where in the cache is A[0][0] stored?**

**ANYWHERE**

**Fields of the memory address**



# Example: accesses in a fully associative cache (1)

i=0,j=0

1. Read A[0][0]
   Memory Address: 0x0C000000
   Tag: 0x0C0000
   Cache miss (V=0)

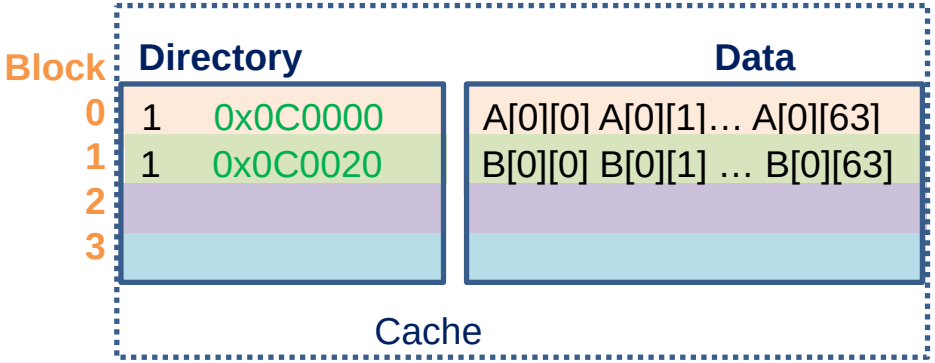
2. Read B[0][0]
   Memory Address: 0x0C002000
   Tag: 0x0C0020
   Cache miss (V=0)

i=0,j=1

1. Read A[0][1]
   Memory Address: 0x0C000004
   Cache hit

2. Read B[0][1]
   Memory Address: 0x0C002004
   Cache hit

```
for (i=0;i<16;i++) {
  tmp=0;
  for (j=0;j<128;j++)
       tmp+=A[i][j] * B[i][j];
  C[i]=tmp; }
```



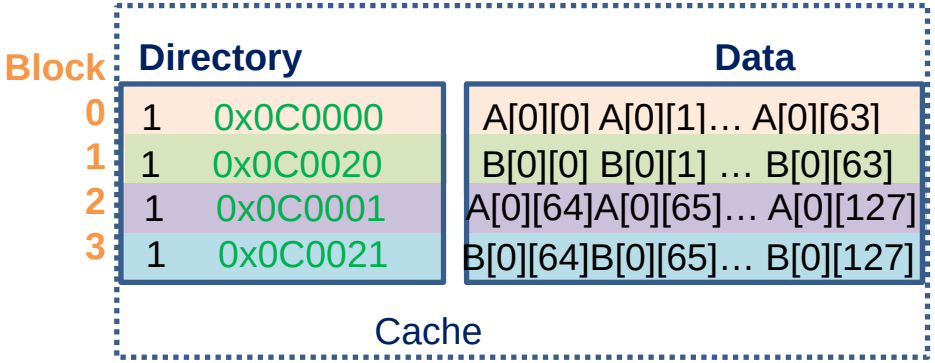
# Example: accesses in a fully associative cache

i=0,j=64

1. Read A[0][64]
   Memory Address: 0x0C000100
   Tag: 0x0C0001
   Cache miss (V=0)

2. Read B[0][64]
   Memory Address: 0x0C002100
   Tag: 0x0C0021
   Cache miss (V=0)

```
for (i=0;i<16;i++) {
  tmp=0;
  for (j=0;j<128;j++)
       tmp+=A[i][j] * B[i][j];
  C[i]=tmp; }
```




Internal loop
Only compulsory misses: 4 read misses, one for each memory block

# Example: 9) Set associative cache

**Assume a cache memory of 1KB with block of 256B and 2 ways**
**Where in the cache is A[0][0] stored?**

$$\text{Number of sets} = \frac{Number\ of\ blocks}{Blocks/Set} = 2$$

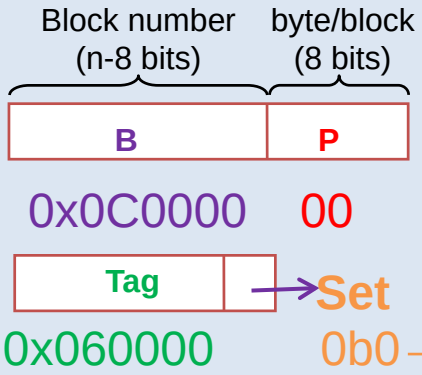
Memory Address: 0x0C000000

0x0C0000

0000 1100 0000 0000 0000 0000

0x060000

**Fields of the memory address**




Two options