

Cache Exercises, Virtual Memory

The heart of the recent hit game *SimAquarium* is a tight loop that calculates the average position of 256 algae. You are evaluating its cache performance on a machine with a 1024-byte direct-mapped data cache with 16-byte blocks ($B = 16$). You are given the following definitions:

```
1 struct algae_position {
2     int x;
3     int y;
4 };
5
6 struct algae_position grid[16][16];
7 int total_x = 0, total_y = 0;
8 int i, j;
```

You should also assume the following:

- `sizeof(int) == 4`.
- grid begins at memory address 0.
- The cache is initially empty.
- The only memory accesses are to the entries of the array `grid`. Variables `i`, `j`, `total_x`, and `total_y` are stored in registers.

Determine the cache performance for the following code:

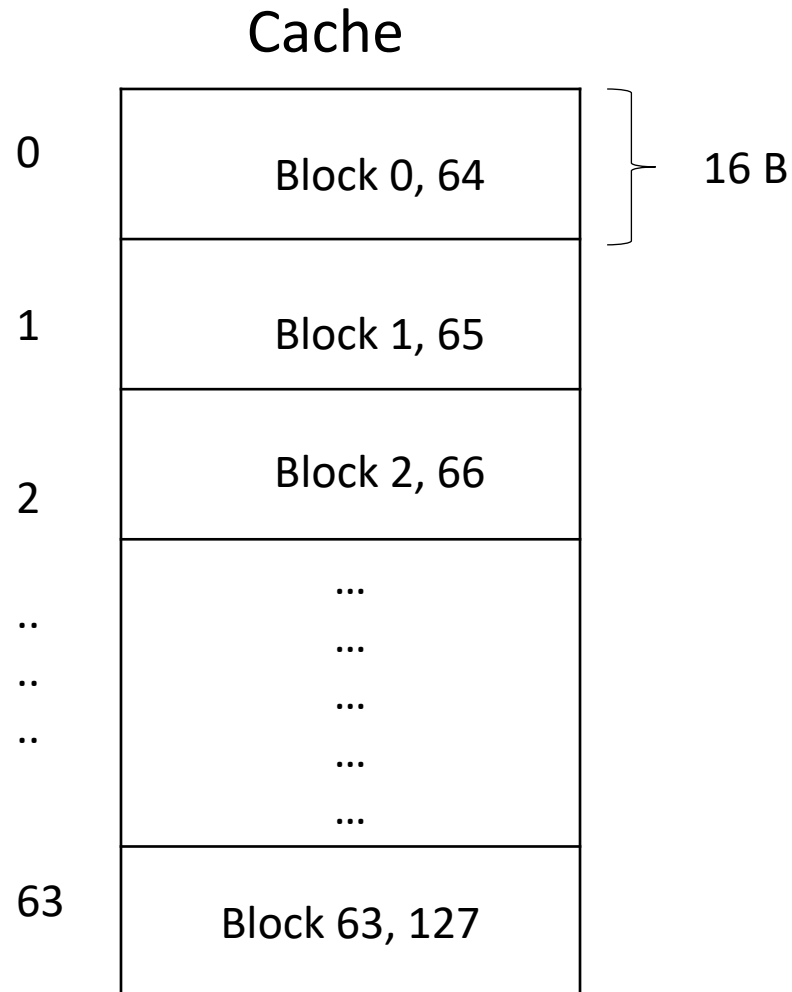
```
1     for (i = 0; i < 16; i++) {
2         for (j = 0; j < 16; j++) {
3             total_x += grid[i][j].x;
4         }
5     }
6
7     for (i = 0; i < 16; i++) {
8         for (j = 0; j < 16; j++) {
9             total_y += grid[i][j].y;
10        }
11    }
```

Randal E. Bryant and David R. O'Hallaron. 2010. Computer Systems: A Programmer's Perspective (2nd. ed.). Addison-Wesley Publishing Company, USA.

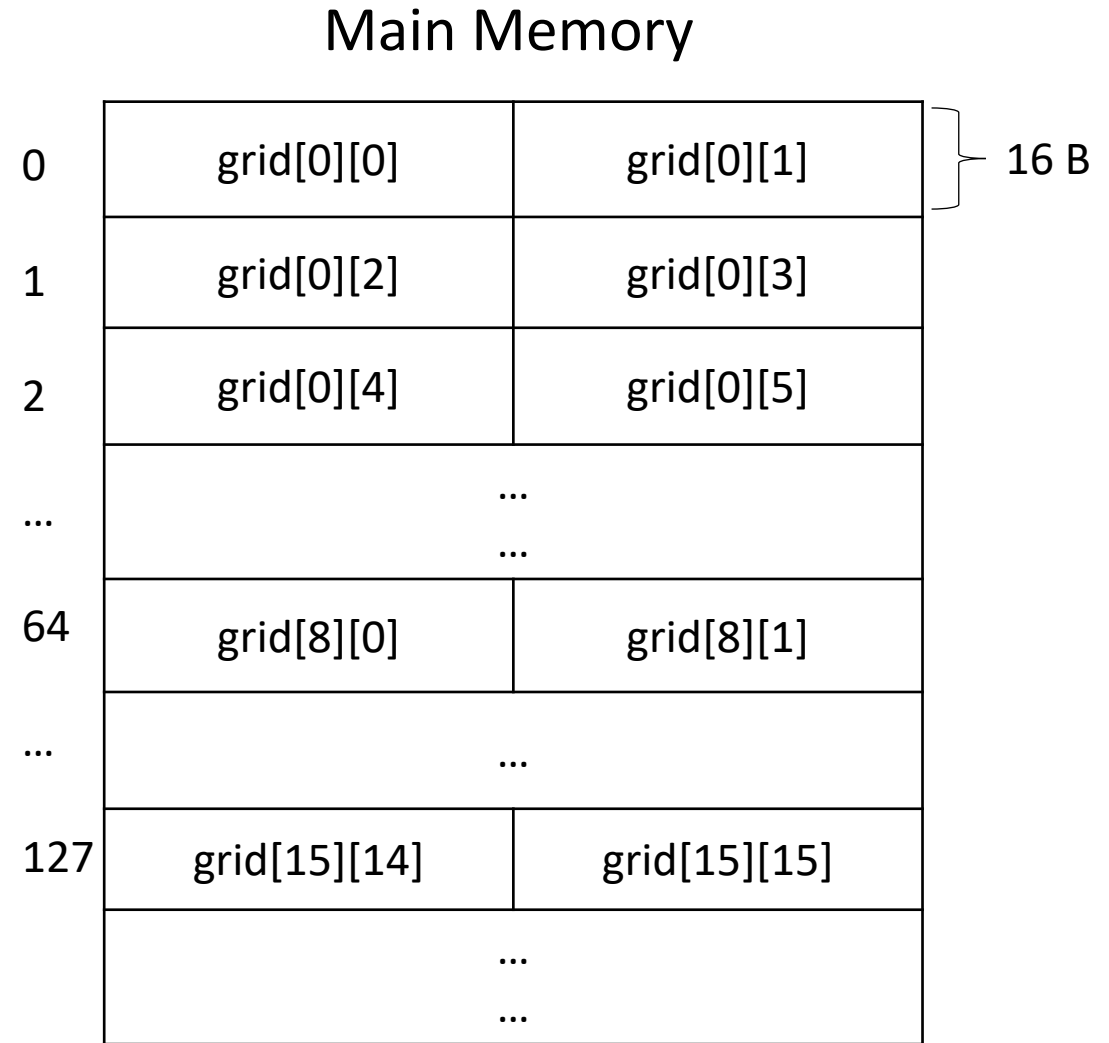
Exercise 1

- What is the **total number of reads**?
- What is the **total number of reads that miss** in the cache?
- What is the **miss rate**?
- Will **increasing the Cache size** improve the performance?
- Will **increasing the Block size** improve performance?

Visualize the Cache and Main Memory

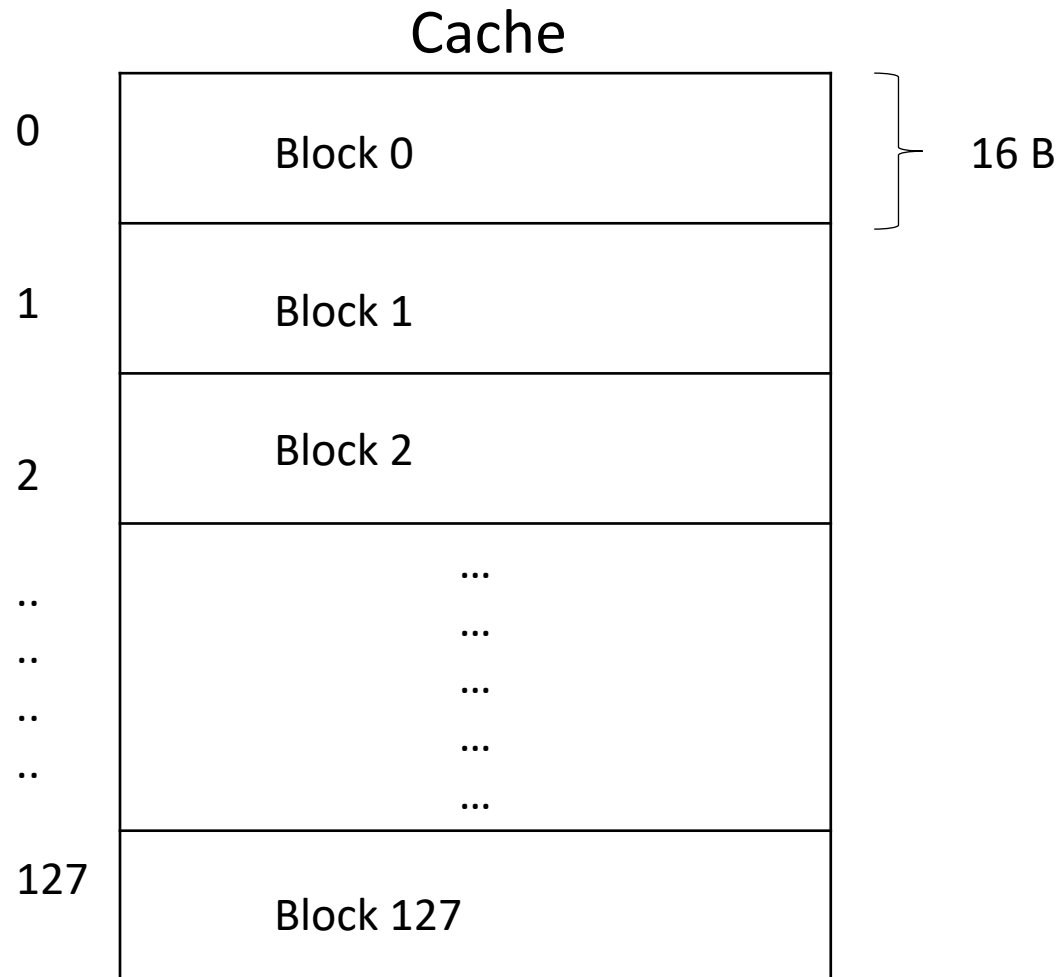


Sequence of accesses
//1st nested for loop
sequence
grid[0][0] -> miss
grid[0][1] -> hit
...
grid[8][0] -> miss
grid[8][1] -> hit
...
grid[15][14] -> miss
grid[15][15] -> hit
//2nd nested for loop
sequence
grid[0][0] -> miss
grid[0][1] -> hit
...
grid[15][0] -> miss
grid[15][1] -> hit



50% MISS RATE

If we double the cache



Sequence of accesses

//1st nested for
sequence

grid[0][0] -> miss

grid[0][1] -> hit

...

grid[8][0] -> miss

grid[8][1] -> hit

...

grid[15][14] -> miss

grid[15][15] -> hit

//2nd nested for
sequence

grid[0][0] -> hit

grid[0][1] -> hit

...

grid[8][0] -> hit

grid[8][1] -> hit

...

grid[15][0] -> hit

grid[15][1] -> hit

25% MISS RATE

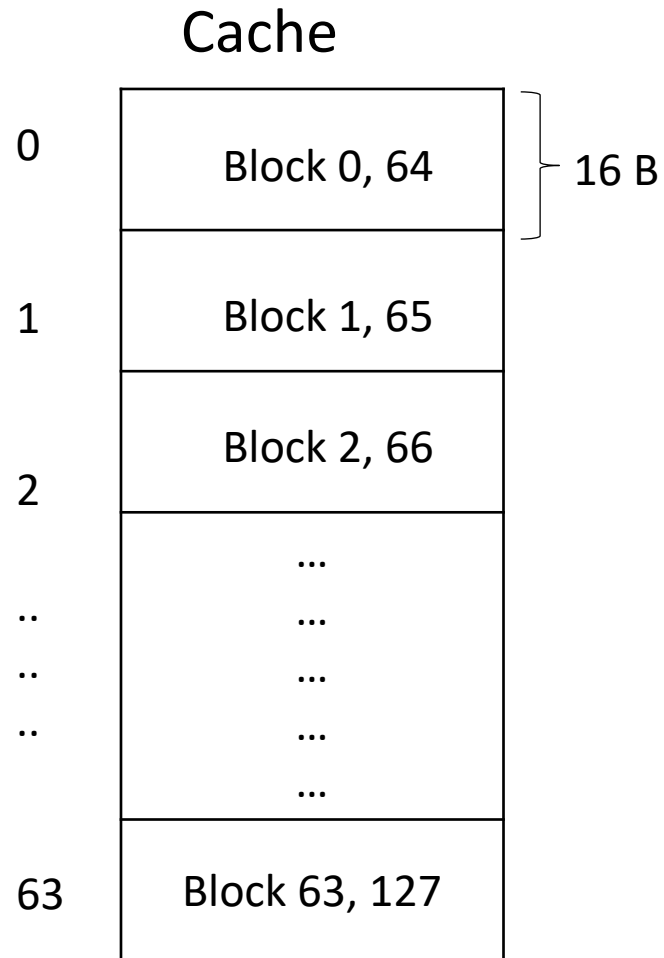
- What is the **total number of reads**?
 - 512
- What is the **total number of** reads that **miss** in the cache?
 - 256
- What is the **miss rate**?
 - 50%
- Will **doubling the Cache size** improve the performance?
 - Yes
 - What will the number of misses in this case?
 - 128 misses for the first nested for loop sequence. 0 misses for the next nested for loop sequence.
- Will **increasing the Block size** improve performance?
 - Yes

Exercise 2

```
1      for (i = 0; i < 16; i++){  
2          for (j = 0; j < 16; j++) {  
3              total_x += grid[j][i].x;  
4              total_y += grid[j][i].y;  
5          }  
6      }
```

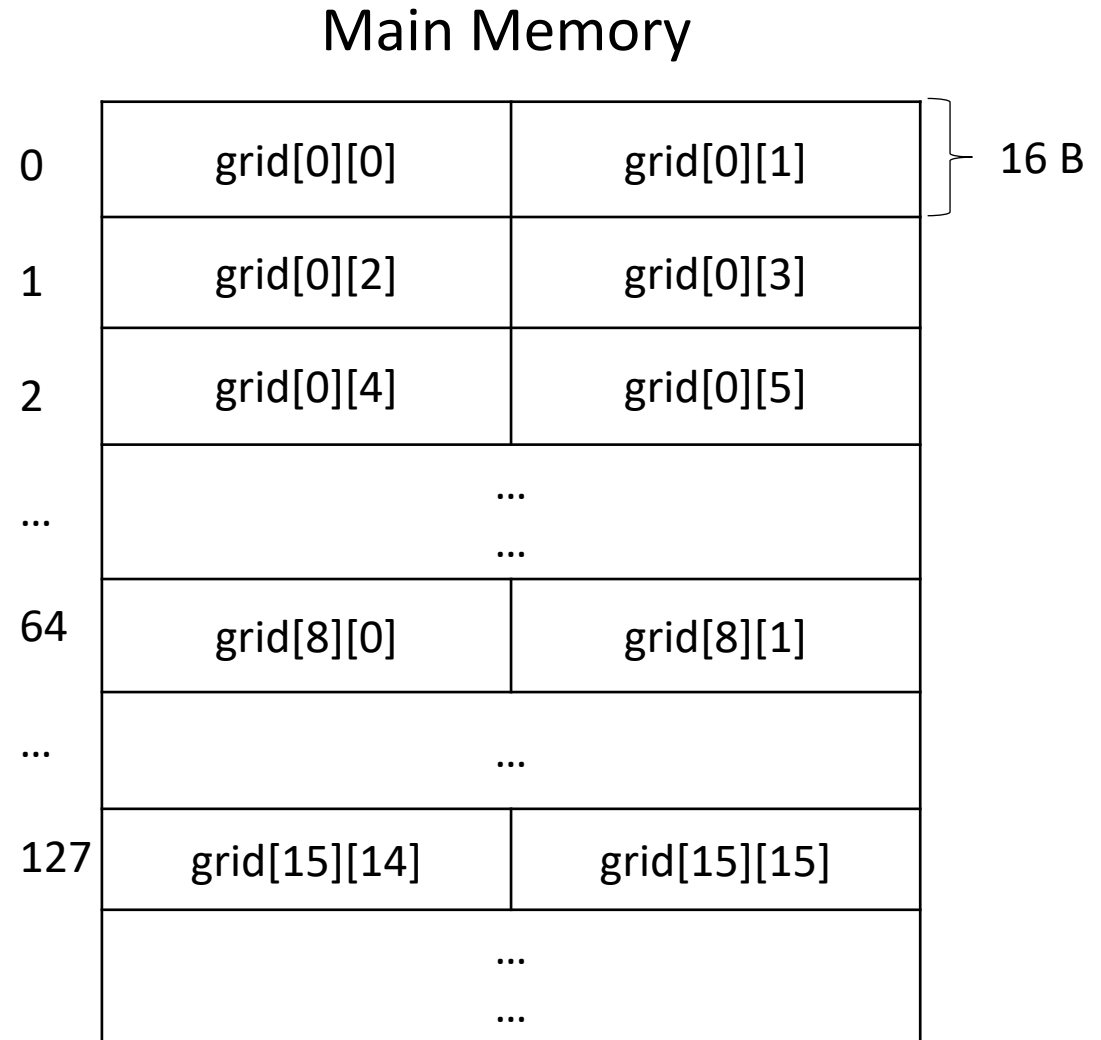
- A. What is the total number of reads?
- B. What is the total number of reads that miss in the cache?
- C. What is the miss rate?
- D. What would the miss rate be if the cache were twice as big?

Visualize the Cache and Main Memory

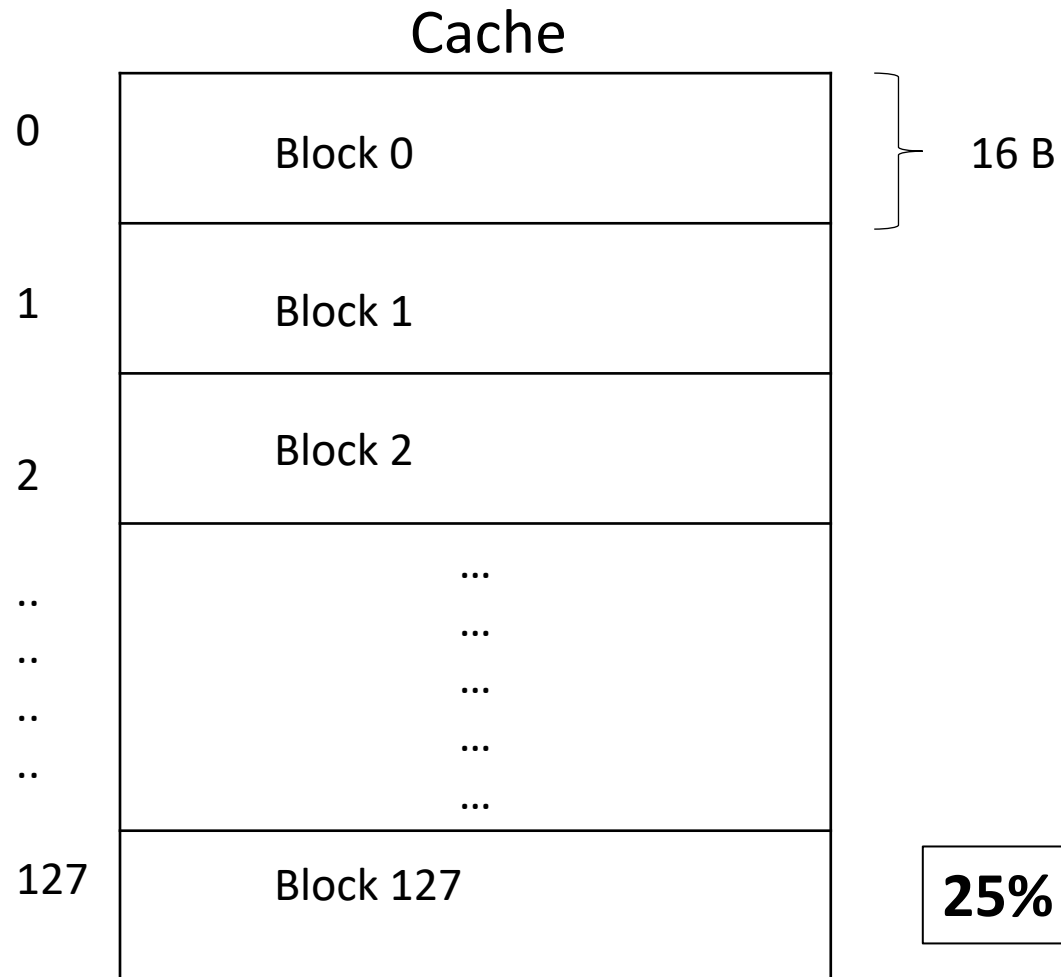


Sequence of accesses

grid[0][0] -> miss
grid[0][0] -> hit
grid[1][0] -> miss
grid[1][0] -> hit
...
grid[8][0] -> miss
grid[8][0] -> hit
...
grid[15][0] -> miss
grid[15][0] -> hit
grid[0][1] -> miss
grid[0][1] -> hit
...
grid[15][1] -> miss
grid[15][1] -> hit
...



If we double the cache



25% MISS RATE

Sequence of accesses

grid[0][0] -> miss
grid[0][0] -> hit
grid[1][0] -> miss
grid[1][0] -> hit
...
grid[8][0] -> miss
grid[8][0] -> hit
...
grid[15][0] -> miss
grid[15][0] -> hit
grid[0][1] -> hit
grid[0][1] -> hit
...
grid[15][1] -> hit
grid[15][1] -> hit
grid [0][2] -> miss
grid [0][2] -> hit
...
grid[0][3] -> hit
grid[0][3] -> hit
...

Exercise 2 - Answers

```
1      for (i = 0; i < 16; i++){  
2          for (j = 0; j < 16; j++) {  
3              total_x += grid[j][i].x;  
4              total_y += grid[j][i].y;  
5          }  
6      }
```

- A. What is the total number of reads? 512
- B. What is the total number of reads that miss in the cache? 256
- C. What is the miss rate? 50%
- D. What would the miss rate be if the cache were twice as big? 25%

Exercise 3 – Try to solve it

```
1      for (i = 0; i < 16; i++){
2          for (j = 0; j < 16; j++) {
3              total_x += grid[i][j].x;
4              total_y += grid[i][j].y;
5          }
6      }
```

Sequence of
accesses

grid[0][0]

grid[0][0]

grid[0][1]

grid[0][1]

...

grid[15][14]

grid[15][14]

grid[15][15]

grid[15][15]

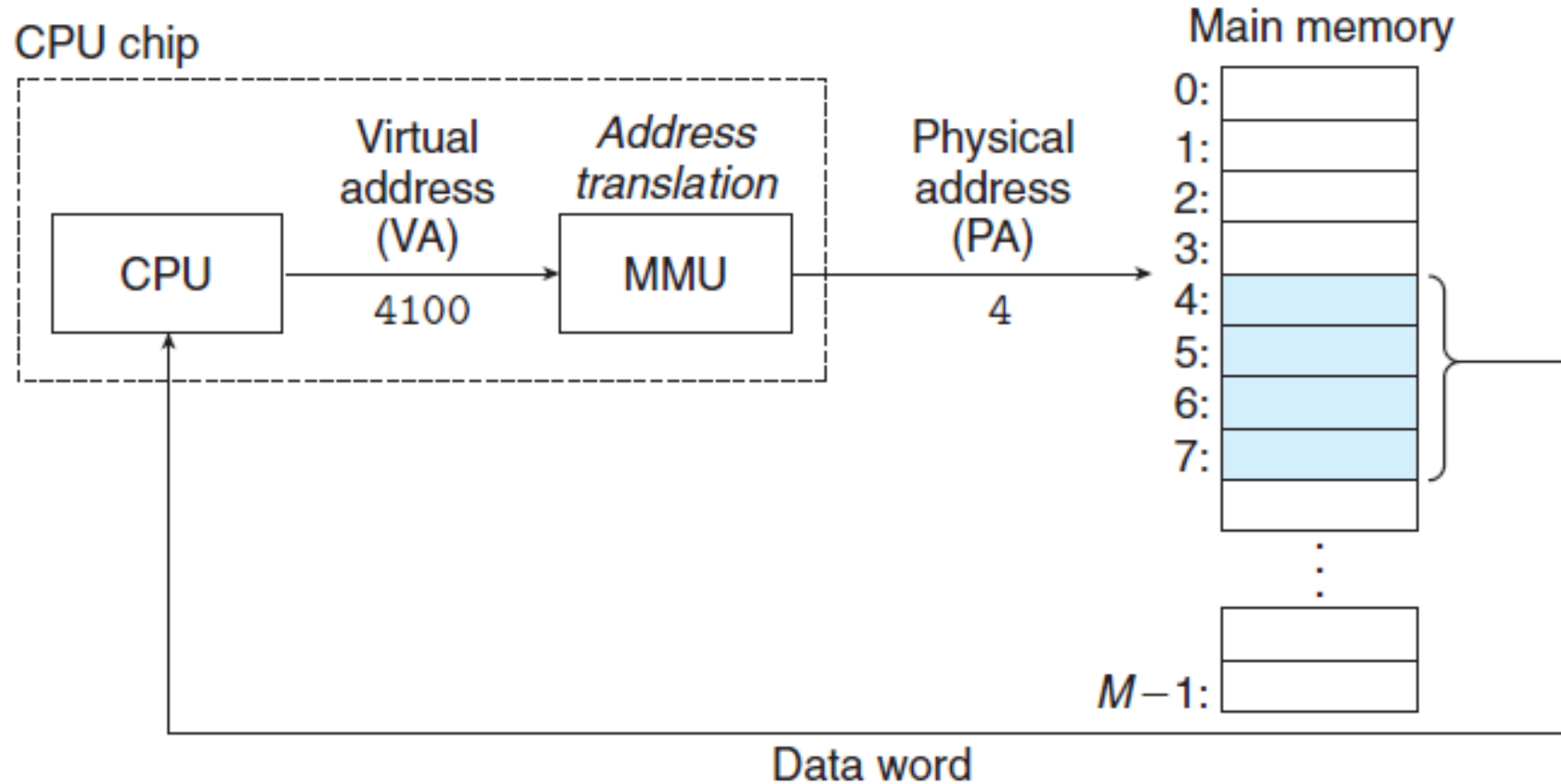
- A. What is the total number of reads?
 - B. What is the total number of reads that miss in the cache?
 - C. What is the miss rate?
 - D. What would the miss rate be if the cache were twice as big?
-

Exercise 3 – Answers

```
1      for (i = 0; i < 16; i++){  
2          for (j = 0; j < 16; j++) {  
3              total_x += grid[i][j].x;  
4              total_y += grid[i][j].y;  
5          }  
6      }
```

- A. What is the total number of reads? 512
 - B. What is the total number of reads that miss in the cache? 128
 - C. What is the miss rate? 25%
 - D. What would the miss rate be if the cache were twice as big? No change
-

Virtual Memory



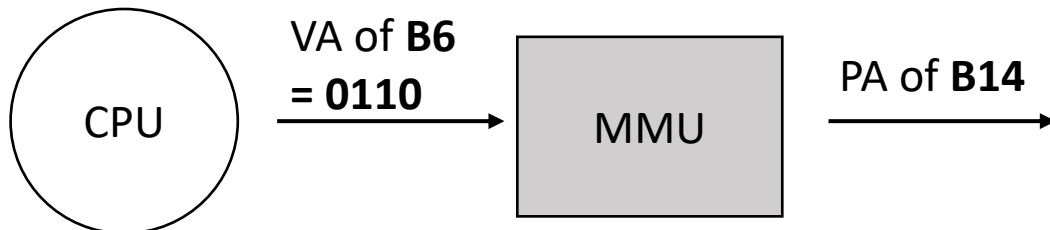
Consider a Scenario

- Main Memory
 - Size of Main Memory = 64 Bytes
 - Frame Size = 4 Bytes
 - Number of Frames = $64/4 = 16$
- Process
 - Size of Process = 16 Bytes
 - Page Size = 4 Bytes
 - Number of Pages = $16/4 = 4$

Paging

How does the MMU get the Physical address of B14 in Main Memory?!!

Process P1				
Page 0	B0	B1	B2	B3
Page 1	B4	B5	B6	B7
Page 2	B8	B9	B10	B11
Page 3	B12	B13	B14	B15



Main Memory				
Frame 0	B0	B1	B2	B3
Frame 1	B4	B5	B6	B7
Frame 2	B8	B9	B10	B11
Frame 3	B12	B13	B14	B15
Frame 4	B16	B17	B18	B19
Frame 5	B20	B21	B22	B23
...				
...				
...				
Frame 14	B56	B57	B58	B59
Frame 15	B60	B61	B62	B63

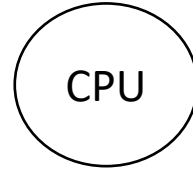
Process P1

Process P1

Page 0	B0	B1	B2	B3
Page 1	B4	B5	B6	B7
Page 2	B8	B9	B10	B11
Page 3	B12	B13	B14	B15

Page Table of Process P1

Page 0	Frame 2
Page 1	Frame 3
Page 2	Frame 4
Page 3	Frame 5



VA of B6
= **0110**



PA of B14
= **001110**

Main Memory

Frame 0	B0	B1	B2	B3
Frame 1	B4	B5	B6	B7
Frame 2	B8	B9	B10	B11
Frame 3	B12	B13	B14	B15
Frame 4	B16	B17	B18	B19
Frame 5	B20	B21	B22	B23
	
Frame 14	B56	B57	B58	B59
Frame 15	B60	B61	B62	B63

Process P1

How many bits is the Physical Address?

Related Links

- You can refer to the following links to know more about virtual memory, how frames are allocated to processes and about the page replacement policies.
 - [Allocation of frames in Operating Systems](#)
 - [Page Replacement Algorithms in Operating Systems](#)
 - [Virtual Memory](#)

Some Important Byte Conversions

- $1\ KB = 2^{10}B$
- $1\ MB = 2^{20}B$
- $1\ GB = 2^{30}B$
- $1\ TB = 2^{40}B$

Example:

- $2^{34}B$
 - $2^{30} * 2^4 B = 2^4 GB = 16 GB$

Exercise

- Physical Address Space (PAS)
 - Size of Main Memory
- Virtual Address Space (VAS)
 - Size of the process

- PAS = 128 KB
- VAS = 128 KB
- How many bits for PA?
 - 17
- How many bits for VA?
 - 17

Exercise

- Physical Address Space (PAS)
 - Size of Main Memory
- Virtual Address Space (VAS)
 - Size of the process

- PAS = 128 KB
- VAS = 128 KB
- Page Size = 4 KB
- How many bits for the page offset?
 - 12 bits

Exercise

- Physical Address Space (PAS)
 - Size of Main Memory
- Virtual Address Space (VAS)
 - Size of the process

- PAS = 128 KB
- VAS = 128 KB
- Page Size = 4 KB
- How many bits for the page number?
 - $(17 - 12)$ bits = 5 bits
- How many Pages?
 - $2^5 = 32$

Exercise

- Physical Address Space (PAS)
 - Size of Main Memory
- Virtual Address Space (VAS)
 - Size of the process

- PAS = 128 KB
- VAS = 128 KB
- Page Size = 4 KB
- How many bits for the Frame number?
 - $(17 - 12)$ bits = 5 bits
- How many Frames?
 - $2^5 = 32$

Exercise

- Physical Address Space (PAS)
 - Size of Main Memory
- Virtual Address Space (VAS)
 - Size of the process

- PAS = 128 KB
- VAS = 128 KB
- Page Size = 4 KB
- How many bits for a Page Table Entry (PTE)?
 - 5 bits

**For this example, we assume that a page table entry contains only the bits to represent the Frame Number.*

Exercise

- Physical Address Space (PAS)
 - Size of Main Memory
- Virtual Address Space (VAS)
 - Size of the process

- PAS = 128 KB
- VAS = 128 KB
- Page Size = 4 KB
- What is the Size of the Page Table?
 - (Number of Pages * Size of a Page Table Entry) = $2^5 * 5$ bits

**For this example, we assume that a page table entry contains only the bits to represent the Frame Number.*

Exercise 2

\$For this example, we assume that a page table entry contains only the bits to represent the Frame Number.

VAS	PAS	VA	PA	Page Size	Page Offset	# of Virtual Pages	# of Physical Pages(or frames)	PTE\$	Size of Page Table
1 MB					10		256		

Exercise 2

\$For this example, we assume that a page table entry contains only the bits to represent the Frame Number.

VAS	PAS	VA	PA	Page Size	Page Offset	# of virtual Pages	# of Physical Pages (or Frames)	PTE\$	Size of Page Table
1 MB				2^{10}	10		256		

Exercise 2

\$For this example, we assume that a page table entry contains only the bits to represent the Frame Number.

VAS	PAS	VA	PA	Page Size	Page Offset	# of Virtual Pages	# of Frames	PTE\$	Size of Page Table
1 MB	# of Frames * Frame Size = 256 KB			2^{10}	10		256		

Exercise 2

\$For this example, we assume that a page table entry contains only the bits to represent the Frame Number.

VAS	PAS	VA	PA	Page Size	Page Offset	# of Virtual Pages	# of Frames	PTE\$	Size of Page Table
1 MB = 2^{20} Bytes	# of Frames * Frame Size = 256 KB = 2^{18} Bytes	20 bits	18 bits	2^{10}	10		256		

Exercise 2

\$For this example, we assume that a page table entry contains only the bits to represent the Frame Number.

VAS	PAS	VA	PA	Page Size	Page Offset	# of Virtual Pages	# of Frames	PTE\$	Size of Page Table
1 MB = 2^{20} Bytes	# of Frames * Frame Size = 256 KB = 2^{18} Bytes	20 bits	18 bits	2^{10}	10	$2^{(20-10)}$ = 2^{10}	256		

Exercise 2

\$For this example, we assume that a page table entry contains only the bits to represent the Frame Number.

VAS	PAS	VA	PA	Page Size	Page Offset	# of Virtual Pages	# of Frames	PTE\$	Size of Page Table
1 MB = 2^{20} Bytes	# of Frames * Frame Size = 256 KB = 2^{18} Bytes	20 bits	18 bits	2^{10}	10	$2^{(20-10)}$ = 2^{10}	256 = 2^8	8 bits	

Exercise 2

\$For this example, we assume that a page table entry contains only the bits to represent the Frame Number.

VAS	PAS	VA	PA	Page Size	Page Offset	# of Virtual Pages	# of Frames	PTE\$	Size of Page Table
1 MB = 2^{20} Bytes	# of Frames * Frame Size = 256 KB = 2^{18} Bytes	20 bits	18 bits	2^{10}	10	$2^{(20-10)}$ = 2^{10}	256 = 2^8	8 bits	$2^{10} * 8 \text{ bits}$

TLB

Virtual Address



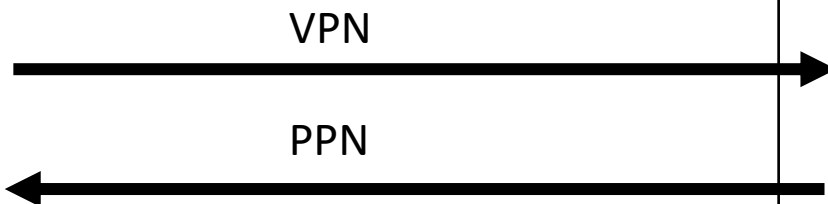
Physical Address



Main Memory/Cache

Page Table

VPN 1	PTE
VPN 2	PTE
VPN 3	PTE
...	..
...	..

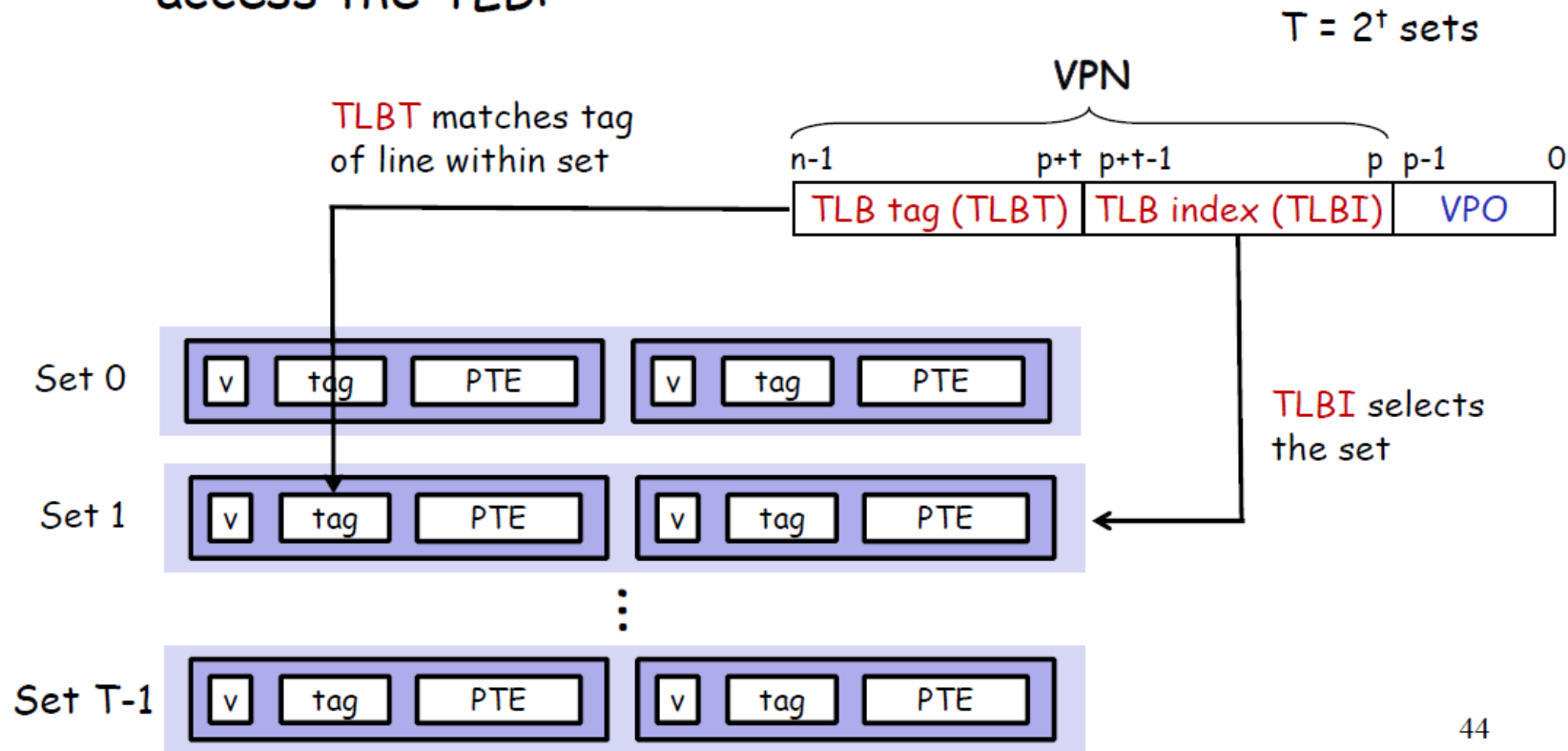


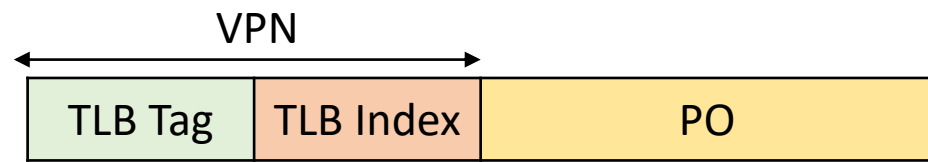
Speeding up Translation with a TLB

- Page table entries (PTEs) are cached in L1 like any other memory word
 - PTEs may be evicted by other data references
 - PTE hit still requires a small L1 delay
- Solution: *Translation Lookaside Buffer* (TLB)
 - Small set-associative hardware cache in MMU
 - Maps virtual page numbers to physical page numbers
 - Contains complete page table entries for small number of pages

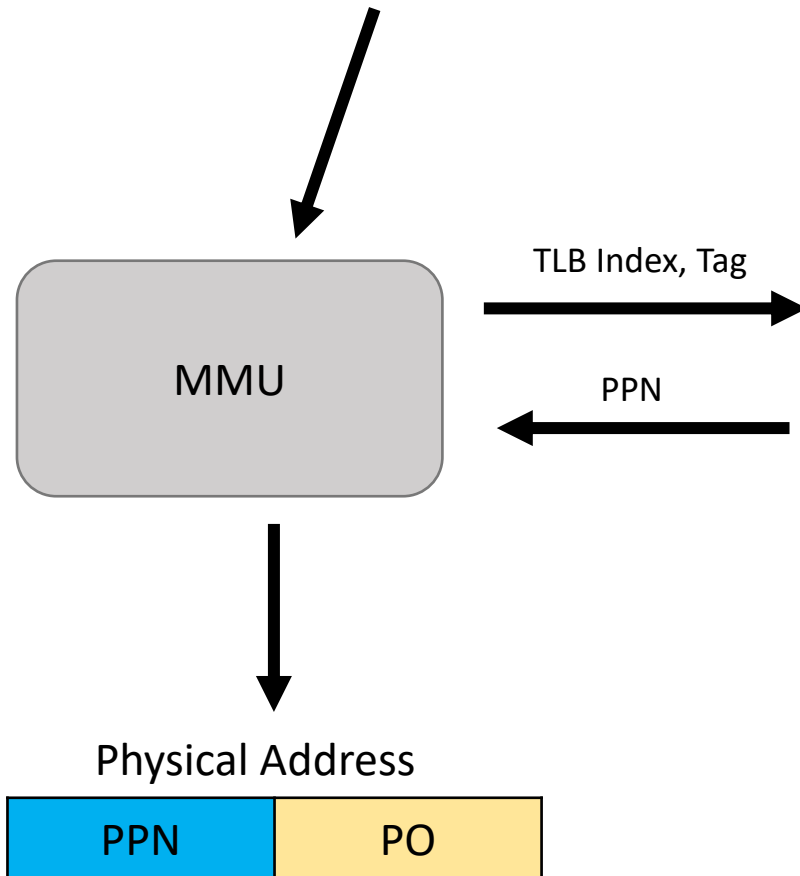
Accessing the TLB

- MMU uses the VPN portion of the virtual address to access the TLB:

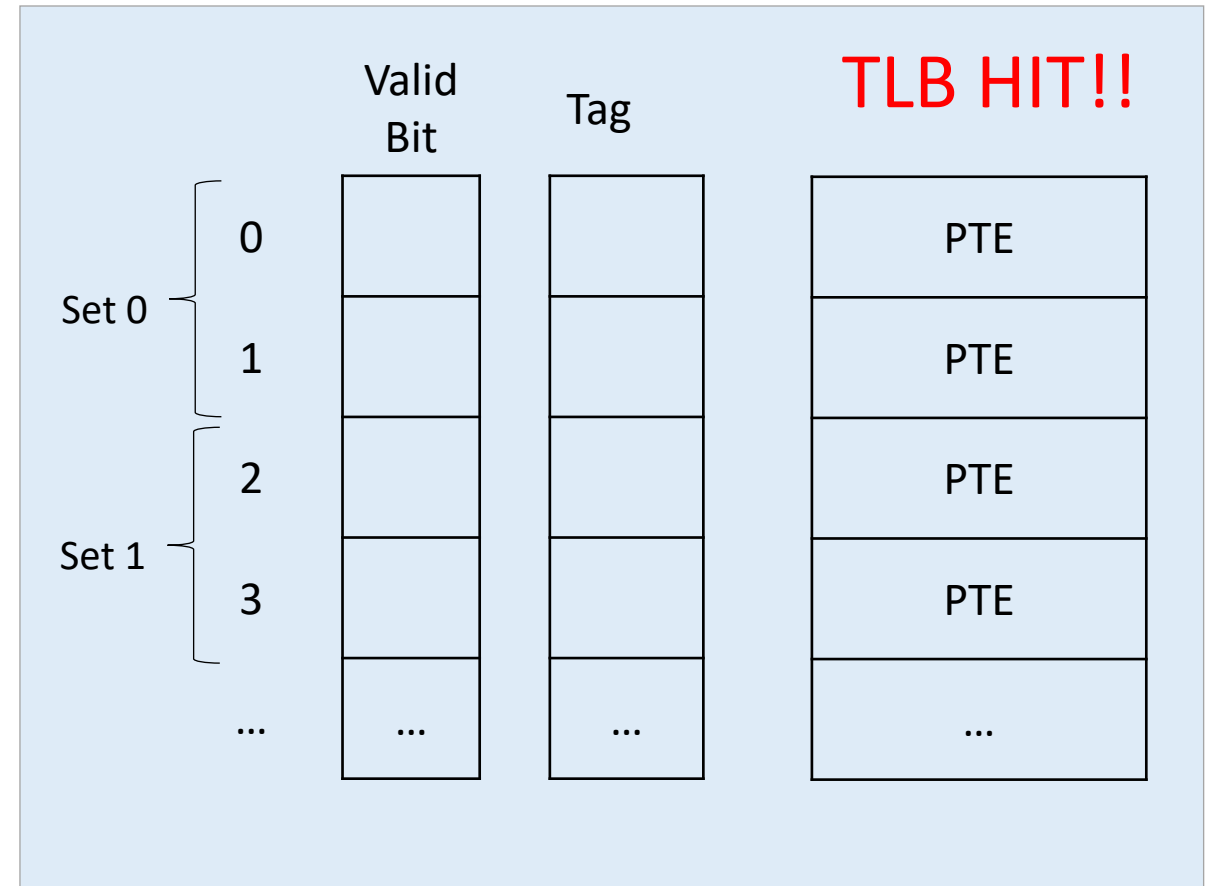




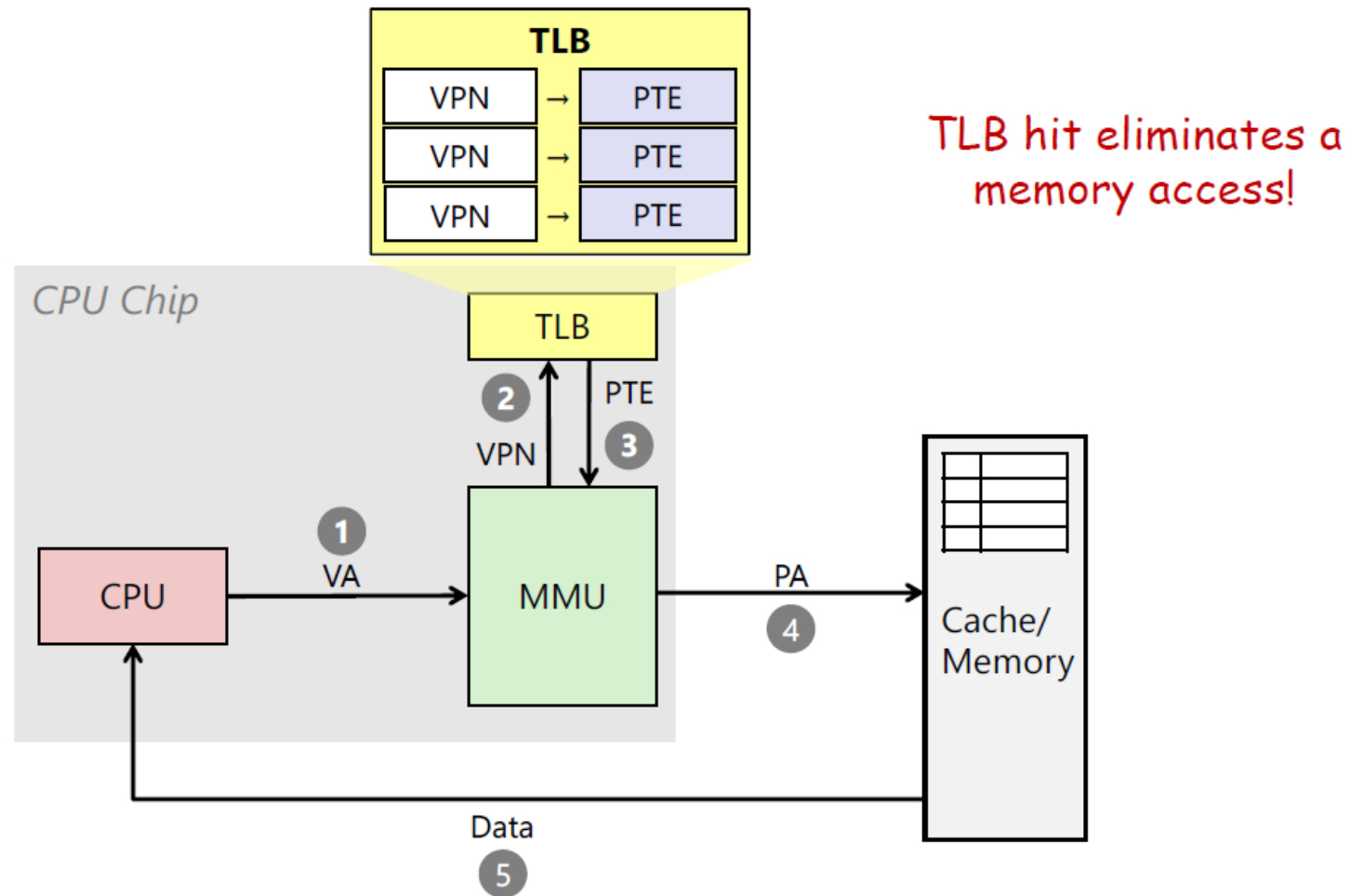
Virtual Address



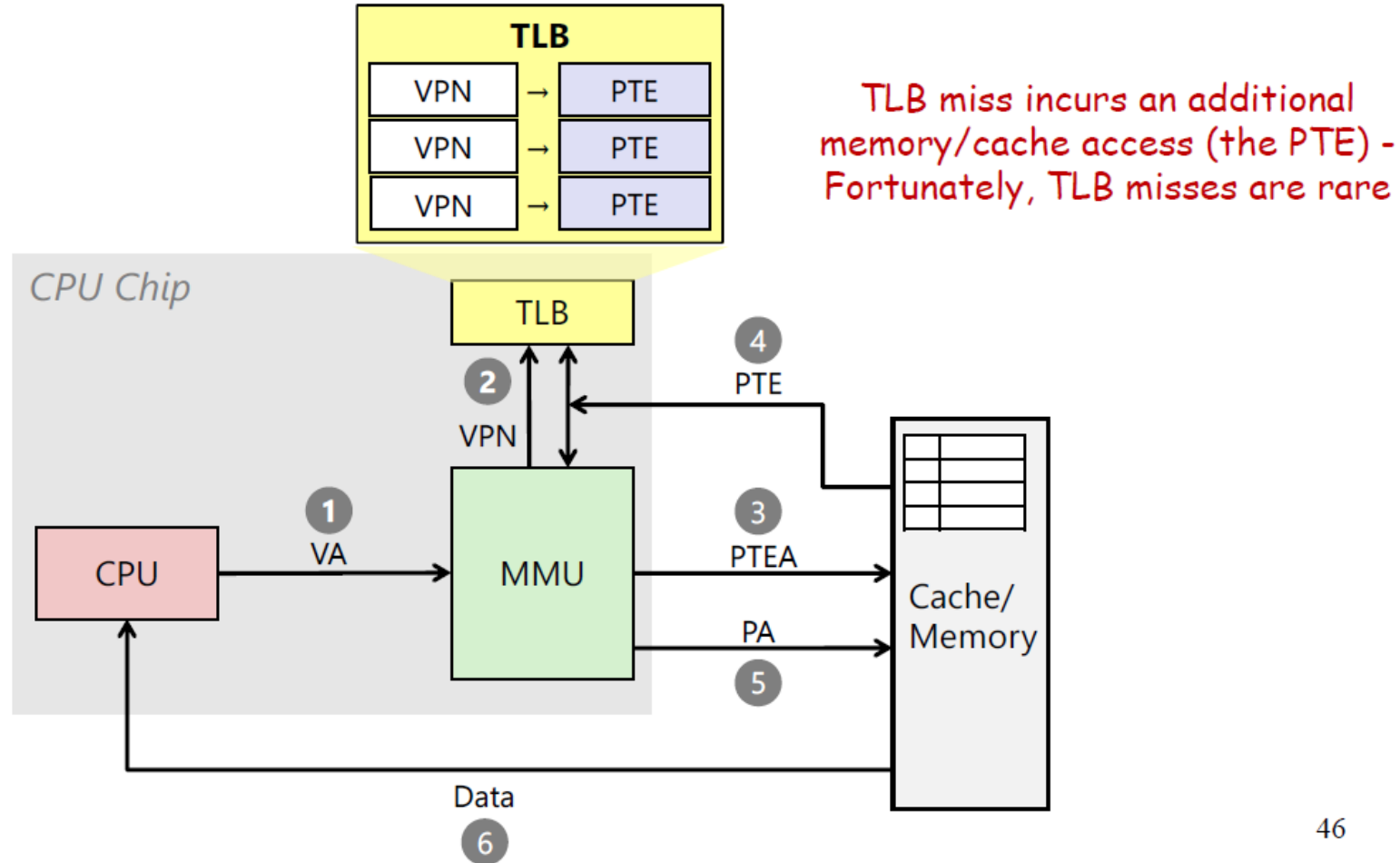
2-way set associative **TLB**



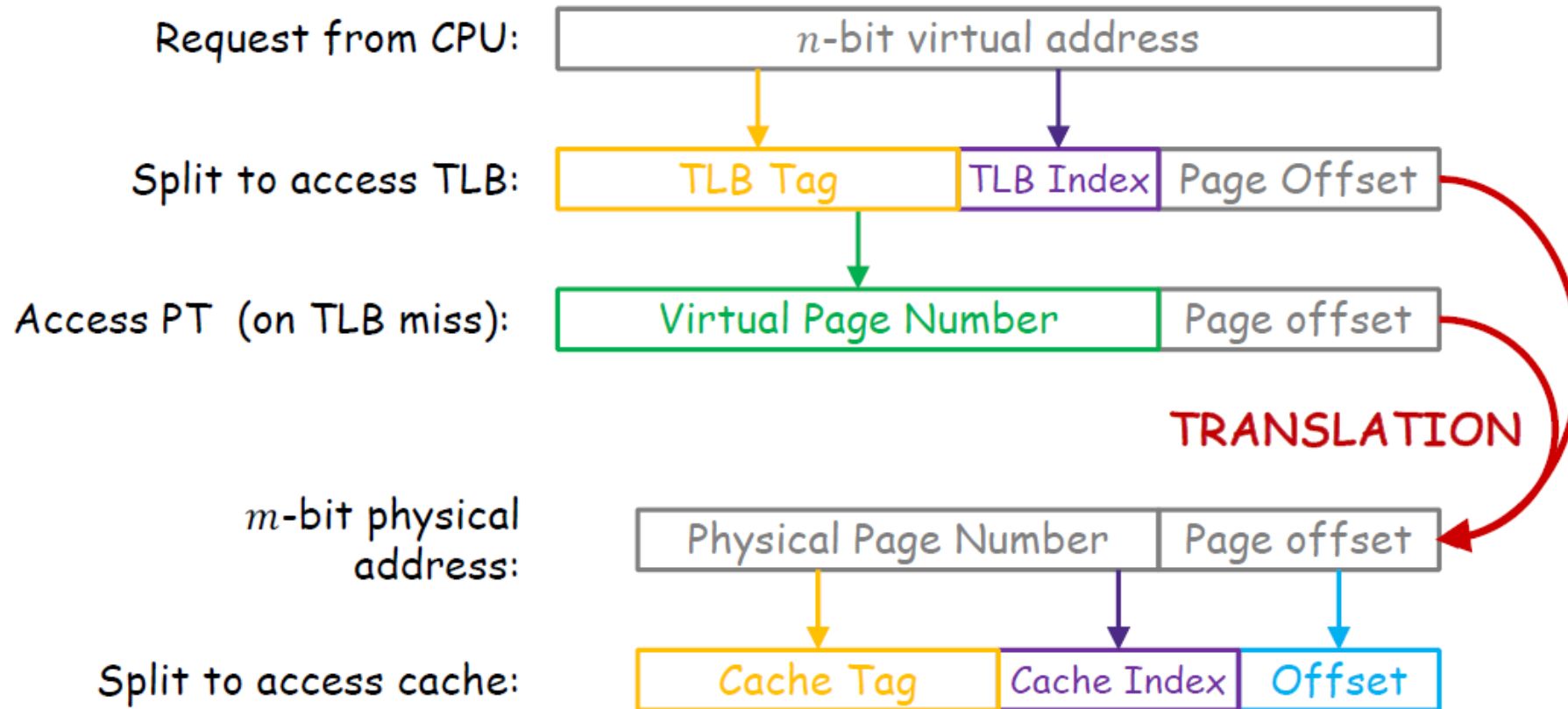
TLB Hit



TLB Miss



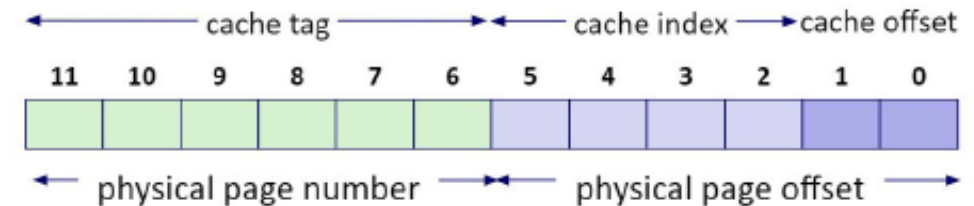
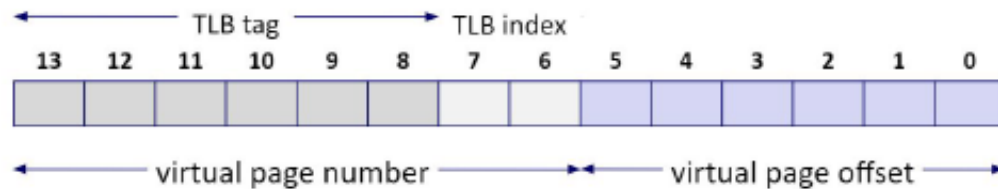
Address Manipulation



Cache Tutorial

A Small Example

Suppose we have a simple memory system with **14-bit** virtual addresses, **12-bit** physical addresses, and a page size of **64 bytes**. The TLB has **16 entries** in total and is **4-way** set associative. The cache is direct-mapped with **16 sets** and a block size of **4 bytes**.



Current State of Memory System:

TLB:

Set	Tag	PPN	V	Tag	PPN	V	Tag	PPN	V	Tag	PPN	V
0	03	–	0	09	0D	1	00	–	0	07	02	1
1	03	2D	1	02	–	0	04	–	0	0A	–	0
2	02	–	0	08	–	0	06	–	0	03	–	0
3	07	–	0	03	0D	1	0A	34	1	02	–	0

Page table (partial):

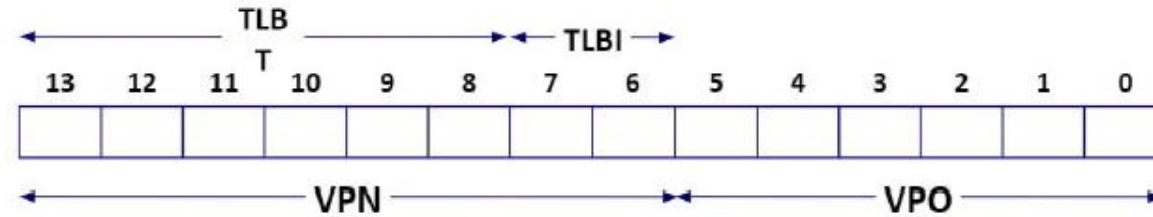
VPN	PPN	V	VPN	PPN	V
0	28	1	8	13	1
1	–	0	9	17	1
2	33	1	A	09	1
3	02	1	B	–	0
4	–	0	C	–	0
5	16	1	D	2D	1
6	–	0	E	–	0
7	–	0	F	0D	1

Cache:

Index	Tag	V	B0	B1	B2	B3	Index	Tag	V	B0	B1	B2	B3
0	19	1	99	11	23	11	8	24	1	3A	00	51	89
1	15	0	–	–	–	–	9	2D	0	–	–	–	–
2	1B	1	00	02	04	08	A	2D	1	93	15	DA	3B
3	36	0	–	–	–	–	B	0B	0	–	–	–	–
4	32	1	43	6D	8F	09	C	12	0	–	–	–	–
5	0D	1	36	72	F0	1D	D	16	1	04	96	34	15
6	31	0	–	–	–	–	E	13	1	83	77	1B	D3
7	16	1	11	C2	DF	03	F	14	0	–	–	–	–

Memory Requests

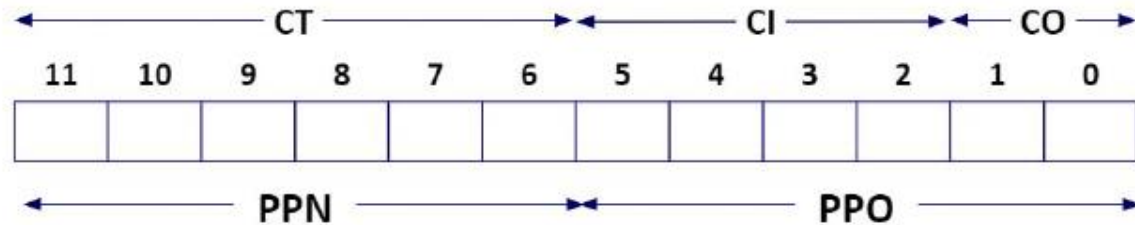
1. Virtual Address: 0x0ACA



VPN ____ TLBT ____ TLBI ____

TLB Hit? ____ Page Fault? ____ PPN ____

Physical Address:

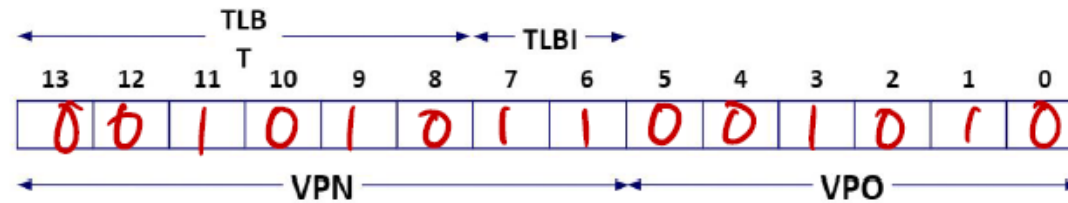


CT ____ CI ____ CO ____ Cache Hit? ____ Data (byte) ____

NOTE: It is just a coincidence that the PPN is the same width as the cache tag.

Memory Requests

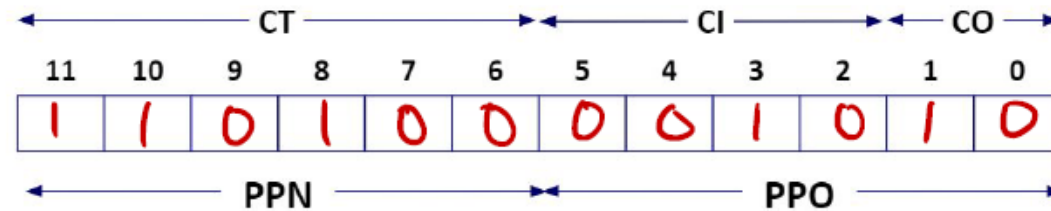
1. Virtual Address: 0x0ACA



VPN 0x2B TLBT 0x0A TLBI 0x3

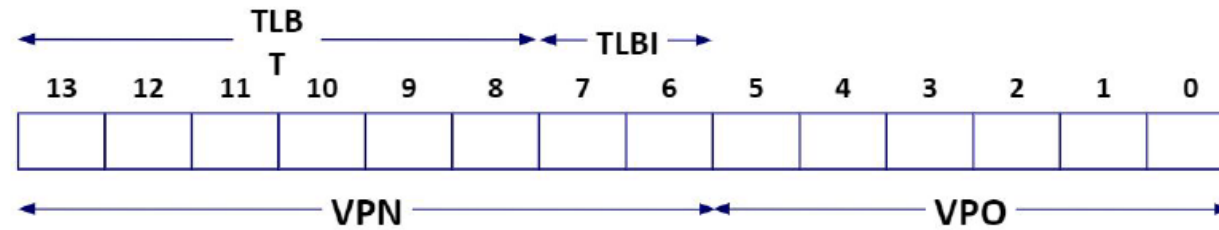
TLB Hit? Yes Page Fault? No PPN 0x34

Physical Address:



CT 0x34 CI 0x2 CO 0x2 Cache Hit? No Data (byte) 11

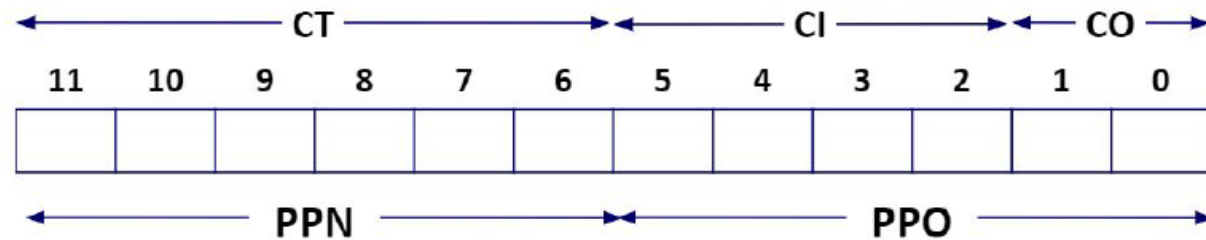
2. Virtual Address: 0x015D



VPN ____ TLBT ____ TLBI ____

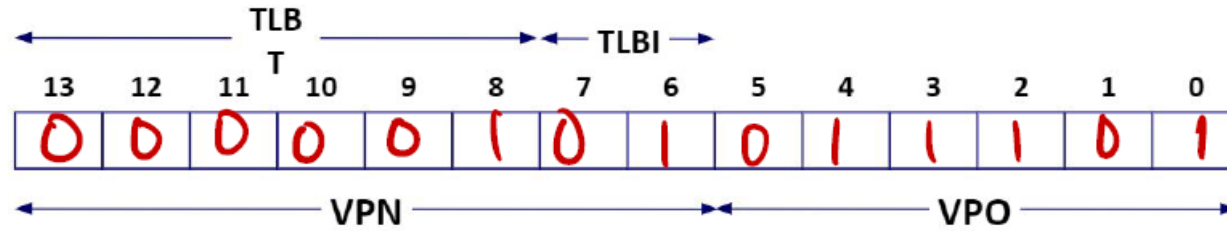
TLB Hit? ____ Page Fault? ____ PPN ____

Physical Address:



CT ____ CI ____ CO ____ Cache Hit? ____ Data (byte) ____

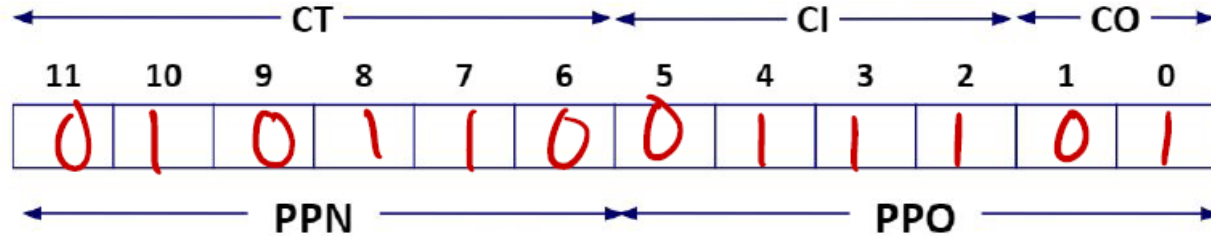
2. Virtual Address: 0x015D



VPN 0x05 TLBT 0x01 TLBI 0x1

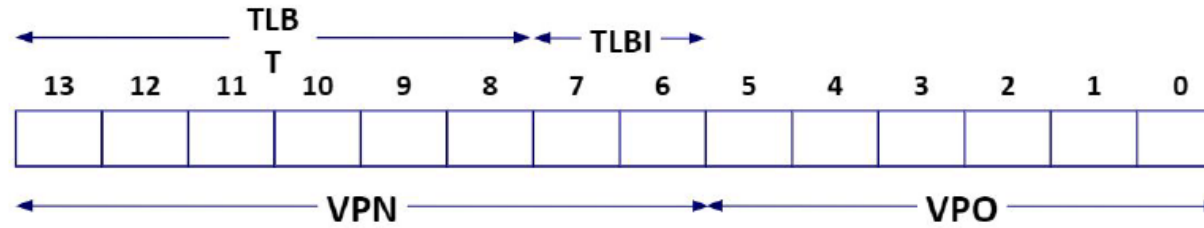
TLB Hit? No Page Fault? No PPN 0x16

Physical Address:



CT 0x16 CI 0x7 CO 0x1 Cache Hit? Yes Data (byte) 0xC2

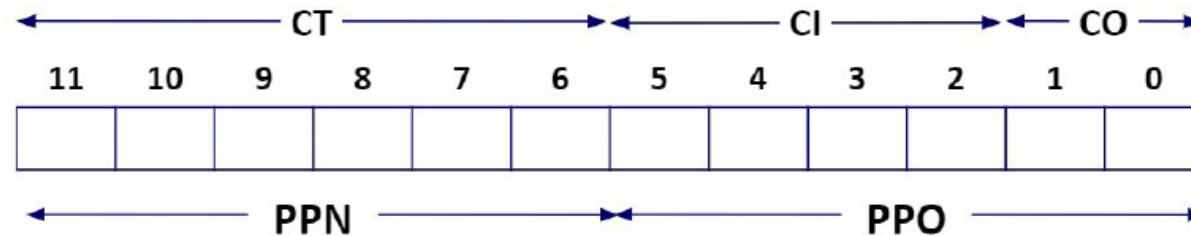
3. Virtual Address: 0x0273



VPN ____ TLBT ____ TLBI ____

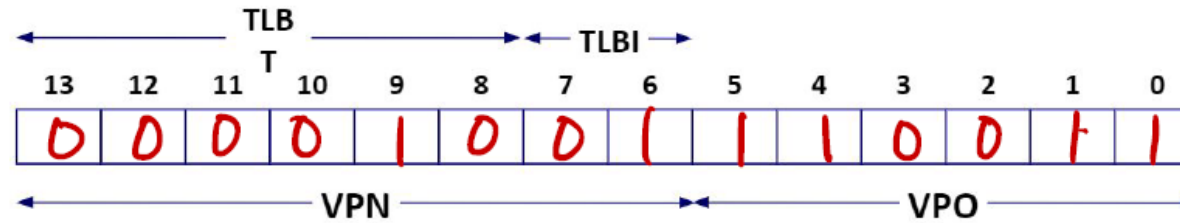
TLB Hit? ____ Page Fault? ____ PPN ____

Physical Address:



CT ____ CI ____ CO ____ Cache Hit? ____ Data (byte) ____

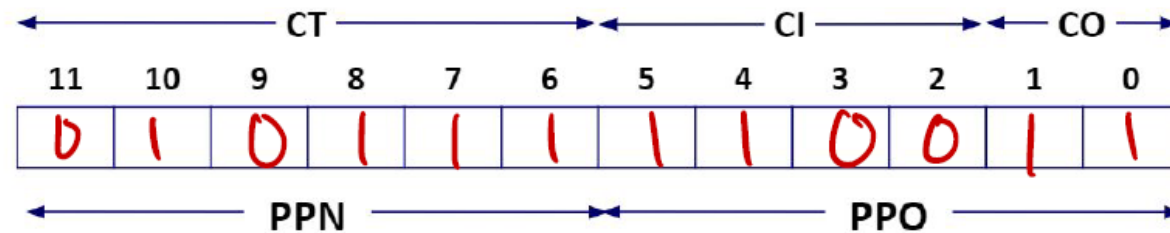
3. Virtual Address: 0x0273



VPN 0x09 TLBT 0x02 TLBI 0x01

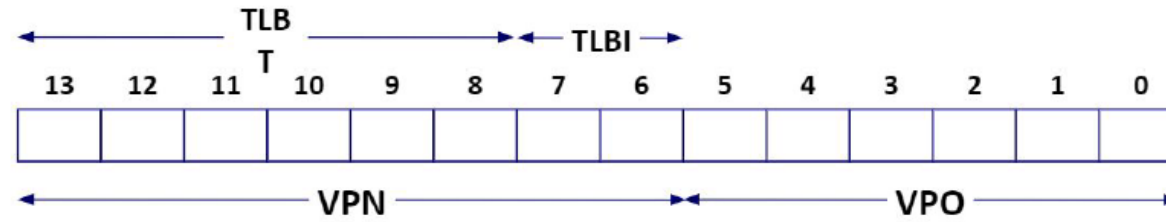
TLB Hit? No Page Fault? No PPN 0x17

Physical Address:



CT 0x17 CI 0xC CO 0x3 Cache Hit? No Data (byte) 11

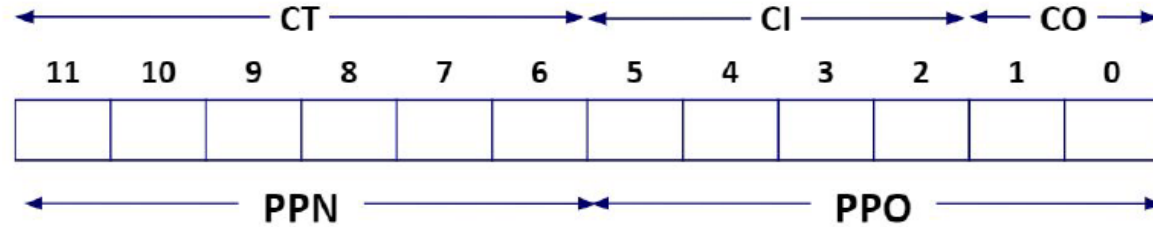
4. Virtual Address: 0x036B



VPN ____ TLBT ____ TLBI ____

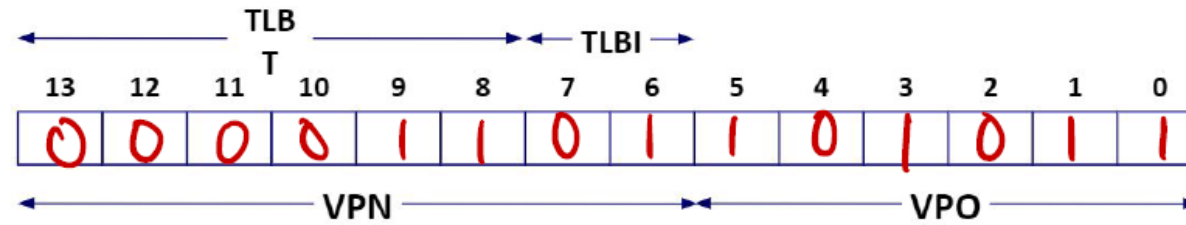
TLB Hit? ____ Page Fault? ____ PPN ____

Physical Address:



CT ____ CI ____ CO ____ Cache Hit? ____ Data (byte) ____

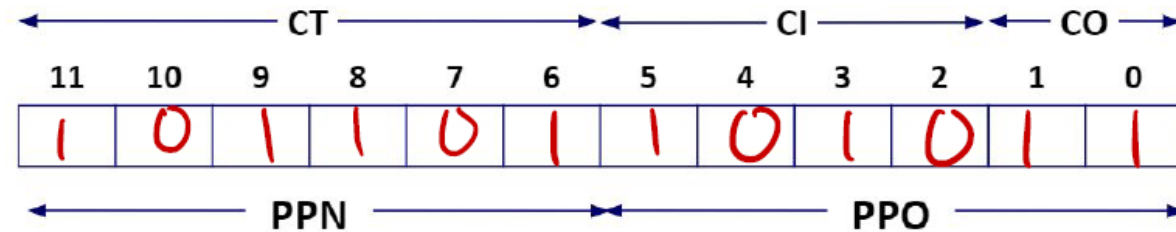
4. Virtual Address: 0x036B



VPN 0x0D TLBT 0x03 TLBI 0x1

TLB Hit? Yes Page Fault? No PPN 0x2D

Physical Address:



CT 0x2D CI 0xA CO 0x3 Cache Hit? Yes Data (byte) 0x3B

Virtual Memory Table

VA width (n)	PA width (m)	Page size (P)	VPN width	PPN width	Bits in PTE (assume V, D, R, W, X)
32	32	16 KiB			
32	26			13	
	32		21		22
		32 KiB	25		26
64			48		29

Virtual Memory Table

VA width (n)	PA width (m)	Page size (P)	VPN width	PPN width	Bits in PTE (assume V, D, R, W, X)
32	32	16 KiB	18	18	23
32	26	8 KiB	19	13	18
36	32	32 KiB	21	17	22
40	36	32 KiB	25	21	26
64	40	64 KiB	48	24	29

$p = \log_2 P$, VPN width = $n - p$, PPN width = $m - p$,
 bits in PTE = PPN width + 5.

References

- https://www.youtube.com/watch?v=2i2N_Qo_FyM&list=PLEbnTDJUrIf_BnzJkkN_J0Tl3iXtL8vq
- <https://www.geeksforgeeks.org/multilevel-paging-in-operating-system/>
- <https://ravindrababuravula.com/index.php>