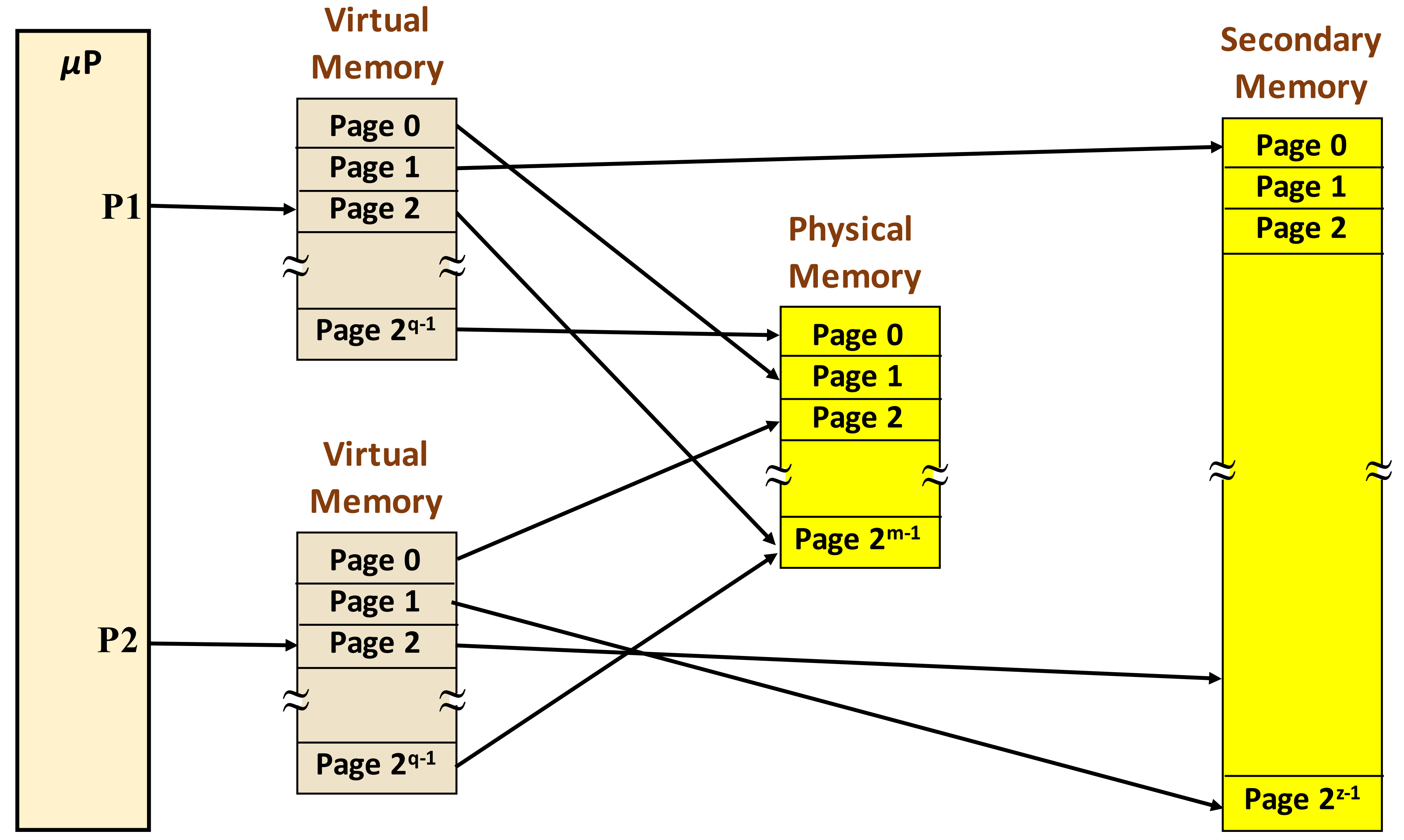
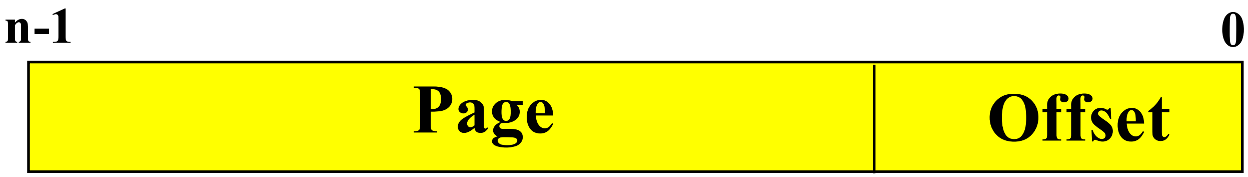
# Virtual Memory
# Address Translation



# Virtual Address

- A virtual address is broken into two fields:



  - *Page refers to a specific block of bytes in memory (virtual or physical).*
  - *Offset specifies a byte within a page.*

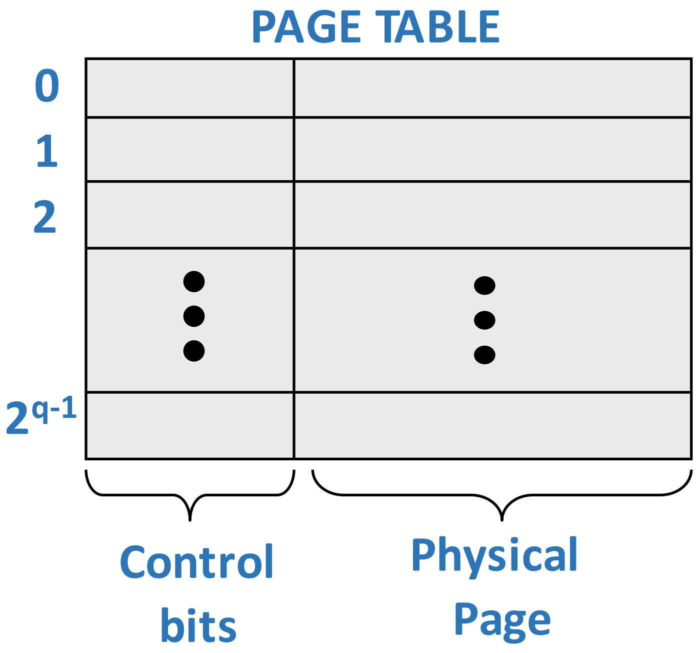
- The number of bits of the offset and the page depends on the size of the pages.

- For a $2^k$-byte page there are $2^{n-k}$ ($2^q$) pages. Thus the Offset field is **k** bits and the Page field is **n-k** bits.

- For a 32-bit address and 4K-byte page there are
  - *$2^{20}$ ($2^{32-12}$) pages and $2^{12}$ bytes per page*
  - *12 bits for the Offset field and 20 bits for the Page field*

# Page Table

- An array with an entry for each page of the virtual space



PAGE TABLE

0

1

2

$2^{q-1}$

Control bits

Physical Page

- Each entry has a series of bits known as control bits and the physical page number corresponding to the virtual page if it resides in physical memory.

- The page table is constructed and managed by the operating system.

# Control Bits

**Valid** Indicates if the virtual page is in physical memory.

**Dirty** Indicates if the corresponding physical page has been written while in physical memory.

**Read** Indicates if the page can be read by the program that is trying to access it.

**Write** Indicates if the page can be written by the program that is trying to access it.

**Execute** Indicates if the code store in the page can be executed by the program that is trying to access it.

# Page Table Base Register

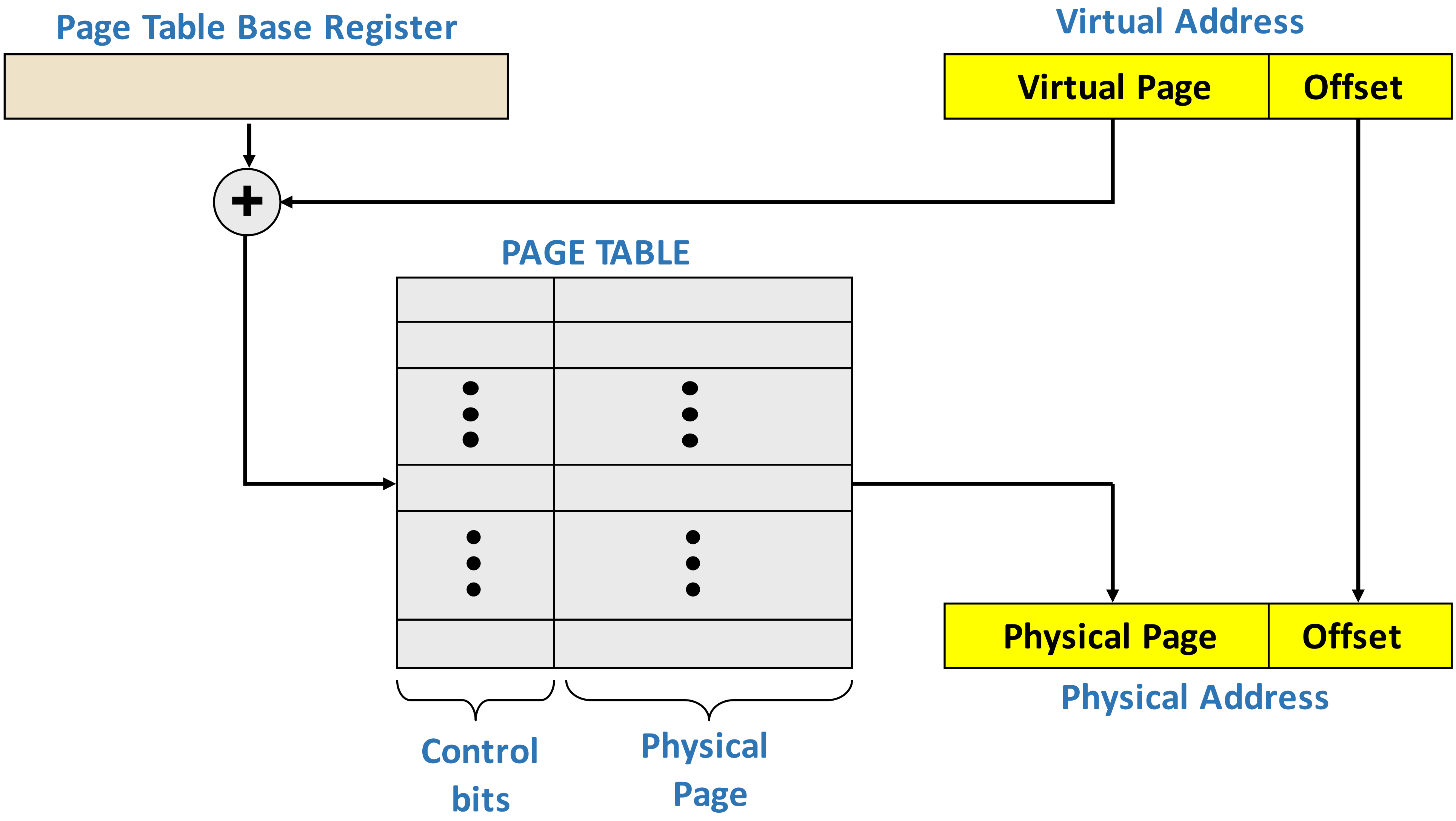
- Each program (process) assumes that it has the whole virtual memory space ($2^{32}$ bytes in case of the 32-bit ARM architecture)

- Each process has its own page table in primary memory

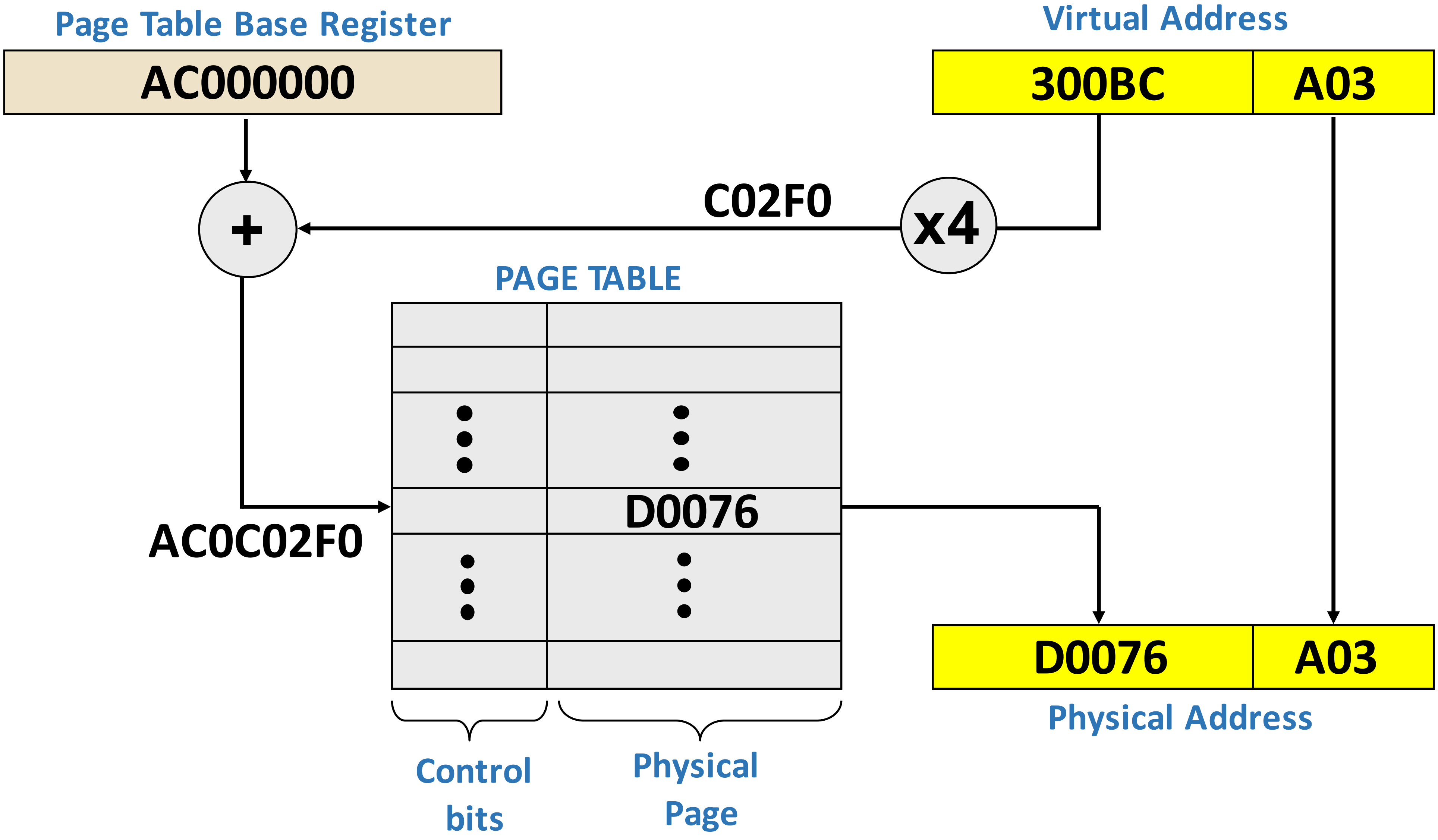
- The location where the first entry of the page table is located in memory is specified by a **Page Table Base Register (PTBR)**

- The **Page Table Base Register** is managed under a privileged operation mode by the operating system (writing to this register is a privileged operation)

# Virtual to Physical Address Translation



# Example: $2^{32}$-byte virtual memory, 4K-byte page, 4-byte page entry



# Another Example: $2^{16}$-byte virtual memory, 4K-byte page, 2-byte page entry, 8-page physical memory

In which physical address is the page table entry for virtual page 4 located ?

If PTBR = 1010100000011100:

= 4x2 + 1010100000011100
= 1000 + 1010100000011100
= 1010100000100100

If PTBR = 0000000000000000:

= 4x2 + 0000000000000000
= 1000 + 0000000000000000
= 0000000000001000

Page Table

| | V | D | R | W | E | Physical Page |
|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 | 1 | 0110 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0111 |
| 2 | 0 | 0 | 0 | 1 | 1 | 0001 |
| 3 | 1 | 1 | 0 | 1 | 0 | 0011 |
| 4 | 0 | 0 | 0 | 1 | 0 | 0001 |
| 5 | 0 | 1 | 1 | 1 | 1 | 0001 |
| 6 | 0 | 0 | 1 | 1 | 0 | 0001 |
| 7 | 1 | 1 | 1 | 1 | 1 | 0101 |
| 8 | 0 | 0 | 1 | 0 | 0 | 0001 |
| 9 | 1 | 1 | 1 | 0 | 0 | 0010 |
| 10 | 0 | 1 | 0 | 0 | 1 | 0001 |
| 11 | 0 | 1 | 1 | 0 | 0 | 0001 |
| 12 | 0 | 1 | 0 | 1 | 1 | 0001 |
| 13 | 1 | 0 | 1 | 0 | 0 | 0000 |
| 14 | 0 | 1 | 1 | 0 | 0 | 0001 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0001 |

V - Valid
D - Dirty
R - Read
W - Write
E - Execute

# Another Example: $2^{16}$-byte virtual memory, 4K-byte page, 2-byte page entry, 8-page physical memory

For which virtual pages an access will result in a page fault?

**Answer:**

**2, 4, 5, 6, 8, 10, 11, 12, 14, 15**

Page Table

| | V | D | R | W | E | Physical Page |
|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 | 1 | 0110 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0111 |
| 2 | 0 | 0 | 0 | 1 | 1 | 0001 |
| 3 | 1 | 1 | 0 | 1 | 0 | 0011 |
| 4 | 0 | 0 | 0 | 1 | 0 | 0001 |
| 5 | 0 | 1 | 1 | 1 | 1 | 0001 |
| 6 | 0 | 0 | 1 | 1 | 0 | 0001 |
| 7 | 1 | 1 | 1 | 1 | 1 | 0101 |
| 8 | 0 | 0 | 1 | 0 | 0 | 0001 |
| 9 | 1 | 1 | 1 | 0 | 0 | 0010 |
| 10 | 0 | 1 | 0 | 0 | 1 | 0001 |
| 11 | 0 | 1 | 1 | 0 | 0 | 0001 |
| 12 | 0 | 1 | 0 | 1 | 1 | 0001 |
| 13 | 1 | 0 | 1 | 0 | 0 | 0000 |
| 14 | 0 | 1 | 1 | 0 | 0 | 0001 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0001 |

V - Valid
D - Dirty
R - Read
W - Write
E - Execute

# Another Example: $2^{16}$-byte virtual memory, 4K-byte page, 2-byte page entry, 8-page physical memory

**From which physical pages is the process allowed to read?**

**Answer:**

**6, 5, 2, 0**

Page Table

| | V | D | R | W | E | Physical Page |
|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 | 1 | 0110 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0111 |
| 2 | 0 | 0 | 0 | 1 | 1 | 0001 |
| 3 | 1 | 1 | 0 | 1 | 0 | 0011 |
| 4 | 0 | 0 | 0 | 1 | 0 | 0001 |
| 5 | 0 | 1 | 1 | 1 | 1 | 0001 |
| 6 | 0 | 0 | 1 | 1 | 0 | 0001 |
| 7 | 1 | 1 | 1 | 1 | 1 | 0101 |
| 8 | 0 | 0 | 1 | 0 | 0 | 0001 |
| 9 | 1 | 1 | 1 | 0 | 0 | 0010 |
| 10 | 0 | 1 | 0 | 0 | 1 | 0001 |
| 11 | 0 | 1 | 1 | 0 | 0 | 0001 |
| 12 | 0 | 1 | 0 | 1 | 1 | 0001 |
| 13 | 1 | 0 | 1 | 0 | 0 | 0000 |
| 14 | 0 | 1 | 1 | 0 | 0 | 0001 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0001 |

V - Valid
D - Dirty
R - Read
W - Write
E - Execute

# Another Example: $2^{16}$-byte virtual memory, 4K-byte page, 2-byte page entry, 8-page physical memory

To which physical pages is the process allowed to write?

Answer:

6, 7, 3, 5

Page Table

| | V | D | R | W | E | Physical Page |
|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 | 1 | 0110 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0111 |
| 2 | 0 | 0 | 0 | 1 | 1 | 0001 |
| 3 | 1 | 1 | 0 | 1 | 0 | 0011 |
| 4 | 0 | 0 | 0 | 1 | 0 | 0001 |
| 5 | 0 | 1 | 1 | 1 | 1 | 0001 |
| 6 | 0 | 0 | 1 | 1 | 0 | 0001 |
| 7 | 1 | 1 | 1 | 1 | 1 | 0101 |
| 8 | 0 | 0 | 1 | 0 | 0 | 0001 |
| 9 | 1 | 1 | 1 | 0 | 0 | 0010 |
| 10 | 0 | 1 | 0 | 0 | 1 | 0001 |
| 11 | 0 | 1 | 1 | 0 | 0 | 0001 |
| 12 | 0 | 1 | 0 | 1 | 1 | 0001 |
| 13 | 1 | 0 | 1 | 0 | 0 | 0000 |
| 14 | 0 | 1 | 1 | 0 | 0 | 0001 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0001 |

V - valid
D - Dirty
R - Read
W - Write
E - Execute

# Another Example: $2^{16}$-byte virtual memory, 4K-byte page, 2-byte page entry, 8-page physical memory

**Of which physical pages is the process allowed to execute code?**

**Answer:**

**6, 5**

**Page Table**

|  | V | D | R | W | E | Physical Page |
|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 | 1 | 0110 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0111 |
| 2 | 0 | 0 | 0 | 1 | 1 | 0001 |
| 3 | 1 | 1 | 0 | 1 | 0 | 0011 |
| 4 | 0 | 0 | 0 | 1 | 0 | 0001 |
| 5 | 0 | 1 | 1 | 1 | 1 | 0001 |
| 6 | 0 | 0 | 1 | 1 | 0 | 0001 |
| 7 | 1 | 1 | 1 | 1 | 1 | 0101 |
| 8 | 0 | 0 | 1 | 0 | 0 | 0001 |
| 9 | 1 | 1 | 1 | 0 | 0 | 0010 |
| 10 | 0 | 1 | 0 | 0 | 1 | 0001 |
| 11 | 0 | 1 | 1 | 0 | 0 | 0001 |
| 12 | 0 | 1 | 0 | 1 | 1 | 0001 |
| 13 | 1 | 0 | 1 | 0 | 0 | 0000 |
| 14 | 0 | 1 | 1 | 0 | 0 | 0001 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0001 |

V - Valid
D - Dirty
R - Read
W - Write
E - Execute

# Another Example: $2^{16}$-byte virtual memory, 4K-byte page, 2-byte page entry, 8-page physical memory

**Which physical pages, that the process is allowed to access, have been written?**

**Answer:**

**7, 3, 5, 2**

Page Table

| | V | D | R | W | E | Physical Page |
|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 | 1 | 0110 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0111 |
| 2 | 0 | 0 | 0 | 1 | 1 | 0001 |
| 3 | 1 | 1 | 0 | 1 | 0 | 0011 |
| 4 | 0 | 0 | 0 | 1 | 0 | 0001 |
| 5 | 0 | 1 | 1 | 1 | 1 | 0001 |
| 6 | 0 | 0 | 1 | 1 | 0 | 0001 |
| 7 | 1 | 1 | 1 | 1 | 1 | 0101 |
| 8 | 0 | 0 | 1 | 0 | 0 | 0001 |
| 9 | 1 | 1 | 1 | 0 | 0 | 0010 |
| 10 | 0 | 1 | 0 | 0 | 1 | 0001 |
| 11 | 0 | 1 | 1 | 0 | 0 | 0001 |
| 12 | 0 | 1 | 0 | 1 | 1 | 0001 |
| 13 | 1 | 0 | 1 | 0 | 0 | 0000 |
| 14 | 0 | 1 | 1 | 0 | 0 | 0001 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0001 |

V - Valid
D - Dirty
R - Read
W - Write
E - Execute

# Another Example: $2^{16}$-byte virtual memory, 4K-byte page, 2-byte page entry, 8-page physical memory

Which is the corresponding physical address?

For virtual address:

$\boxed{1001}\boxed{000111011111}$

Answer:

$\boxed{0010}\boxed{000111011111}$

Page Table

| | V | D | R | W | E | Physical Page |
|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 | 1 | 0110 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0111 |
| 2 | 0 | 0 | 0 | 1 | 1 | 0001 |
| 3 | 1 | 1 | 0 | 1 | 0 | 0011 |
| 4 | 0 | 0 | 0 | 1 | 0 | 0001 |
| 5 | 0 | 1 | 1 | 1 | 1 | 0001 |
| 6 | 0 | 0 | 1 | 1 | 0 | 0001 |
| 7 | 1 | 1 | 1 | 1 | 1 | 0101 |
| 8 | 0 | 0 | 1 | 0 | 0 | 0001 |
| 9 | 1 | 1 | 1 | 0 | 0 | 0010 |
| 10 | 0 | 1 | 0 | 0 | 1 | 0001 |
| 11 | 0 | 1 | 1 | 0 | 0 | 0001 |
| 12 | 0 | 1 | 0 | 1 | 1 | 0001 |
| 13 | 1 | 0 | 1 | 0 | 0 | 0000 |
| 14 | 0 | 1 | 1 | 0 | 0 | 0001 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0001 |

V - Valid
D - Dirty
R - Read
W - Write
E - Execute

# Another Example: $2^{16}$-byte virtual memory, 4K-byte page, 2-byte page entry, 8-page physical memory

Which is the corresponding physical address?

For virtual address:

**1101 111111011100**

Answer:

**0000 111111011100**

Page Table

| | V | D | R | W | E | Physical Page |
|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 | 1 | 0110 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0111 |
| 2 | 0 | 0 | 0 | 1 | 1 | 0001 |
| 3 | 1 | 1 | 0 | 1 | 0 | 0011 |
| 4 | 0 | 0 | 0 | 1 | 0 | 0001 |
| 5 | 0 | 1 | 1 | 1 | 1 | 0001 |
| 6 | 0 | 0 | 1 | 1 | 0 | 0001 |
| 7 | 1 | 1 | 1 | 1 | 1 | 0101 |
| 8 | 0 | 0 | 1 | 0 | 0 | 0001 |
| 9 | 1 | 1 | 1 | 0 | 0 | 0010 |
| 10 | 0 | 1 | 0 | 0 | 1 | 0001 |
| 11 | 0 | 1 | 1 | 0 | 0 | 0001 |
| 12 | 0 | 1 | 0 | 1 | 1 | 0001 |
| 13 | 1 | 0 | 1 | 0 | 0 | 0000 |
| 14 | 0 | 1 | 1 | 0 | 0 | 0001 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0001 |

V - Valid
D - Dirty
R - Read
W - Write
E - Execute

# Another Example: $2^{16}$-byte virtual memory, 4K-byte page, 2-byte page entry, 8-page physical memory

**Which is the corresponding physical address?**

**Answer:**

**1110100111000000**

**Physical address:**

**Not in physical memory**

**Page Table**

| | V | D | R | W | E | Physical Page |
|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 | 1 | 0110 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0111 |
| 2 | 0 | 0 | 0 | 1 | 1 | 0001 |
| 3 | 1 | 1 | 0 | 1 | 0 | 0011 |
| 4 | 0 | 0 | 0 | 1 | 0 | 0001 |
| 5 | 0 | 1 | 1 | 1 | 1 | 0001 |
| 6 | 0 | 0 | 1 | 1 | 0 | 0001 |
| 7 | 1 | 1 | 1 | 1 | 1 | 0101 |
| 8 | 0 | 0 | 1 | 0 | 0 | 0001 |
| 9 | 1 | 1 | 1 | 0 | 0 | 0010 |
| 10 | 0 | 1 | 0 | 0 | 1 | 0001 |
| 11 | 0 | 1 | 1 | 0 | 0 | 0001 |
| 12 | 0 | 1 | 0 | 1 | 1 | 0001 |
| 13 | 1 | 0 | 1 | 0 | 0 | 0000 |
| 14 | 0 | 1 | 1 | 0 | 0 | 0001 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0001 |

Invalid

V - Valid
D - Dirty
R - Read
W - Write
E - Execute

# Page Fault

- Takes place when a virtual page is not in physical memory (valid bit = 0) or when there is a violation of the access permission (Write, Read or Execute bit equal zero).

- Generates a exception that enters a privileged mode an transfers control to the operating system. (Prefetch Abort and Data Abort exceptions in ARM).

# Operating System Intervention on a Page Fault

1. Context Switch - Stops the faulting process and let another process to run (changes the content of the Page Table Base Register).

2. Uses an algorithm to determine a victim in physical memory to place the faulting page.

3. If the victim has been written (dirty bit = 1), instructs an I/O port to initiate the transfer of the victim page to secondary memory.

4. After the victim is transferred, instructs and I/O port to initiate the transfer of the faulting page to physical memory into the space previously occupied by the victim page.

5. After the page is transferred, places the faulting process back into the execution queue.