CS143A Principles on Operating Systems Discussion 04:

Instructor: Prof. Anton Burtsev

TA: Saehanseul Yi (Hans)

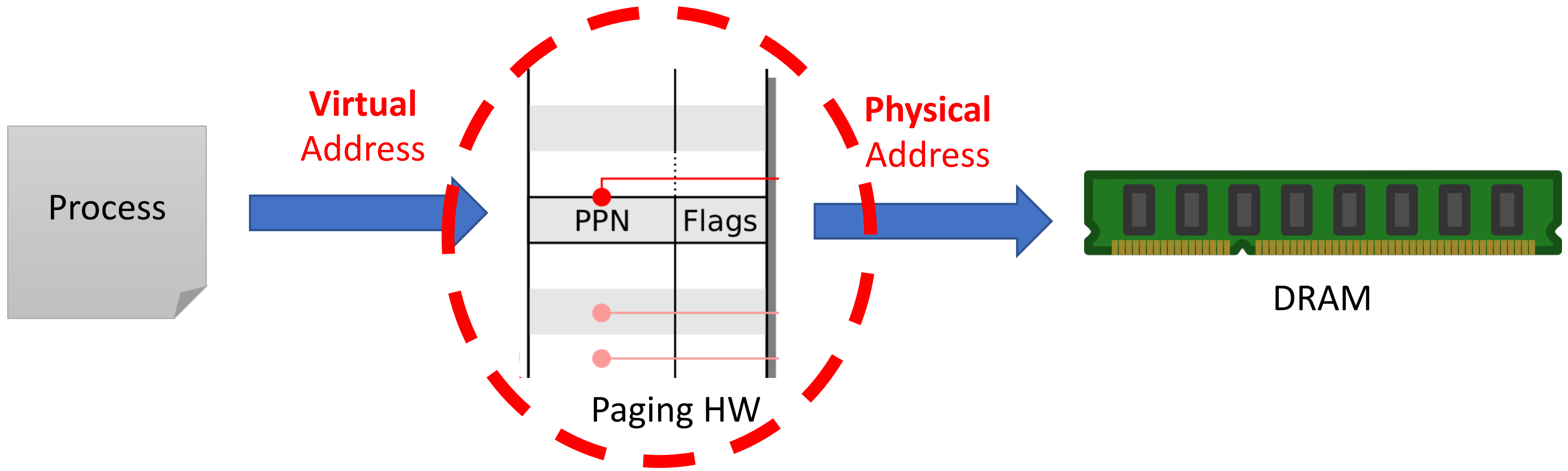
Oct 25, 2019 **Noon**

Agenda

- Segmentation
- Paging
- A simple address translation example

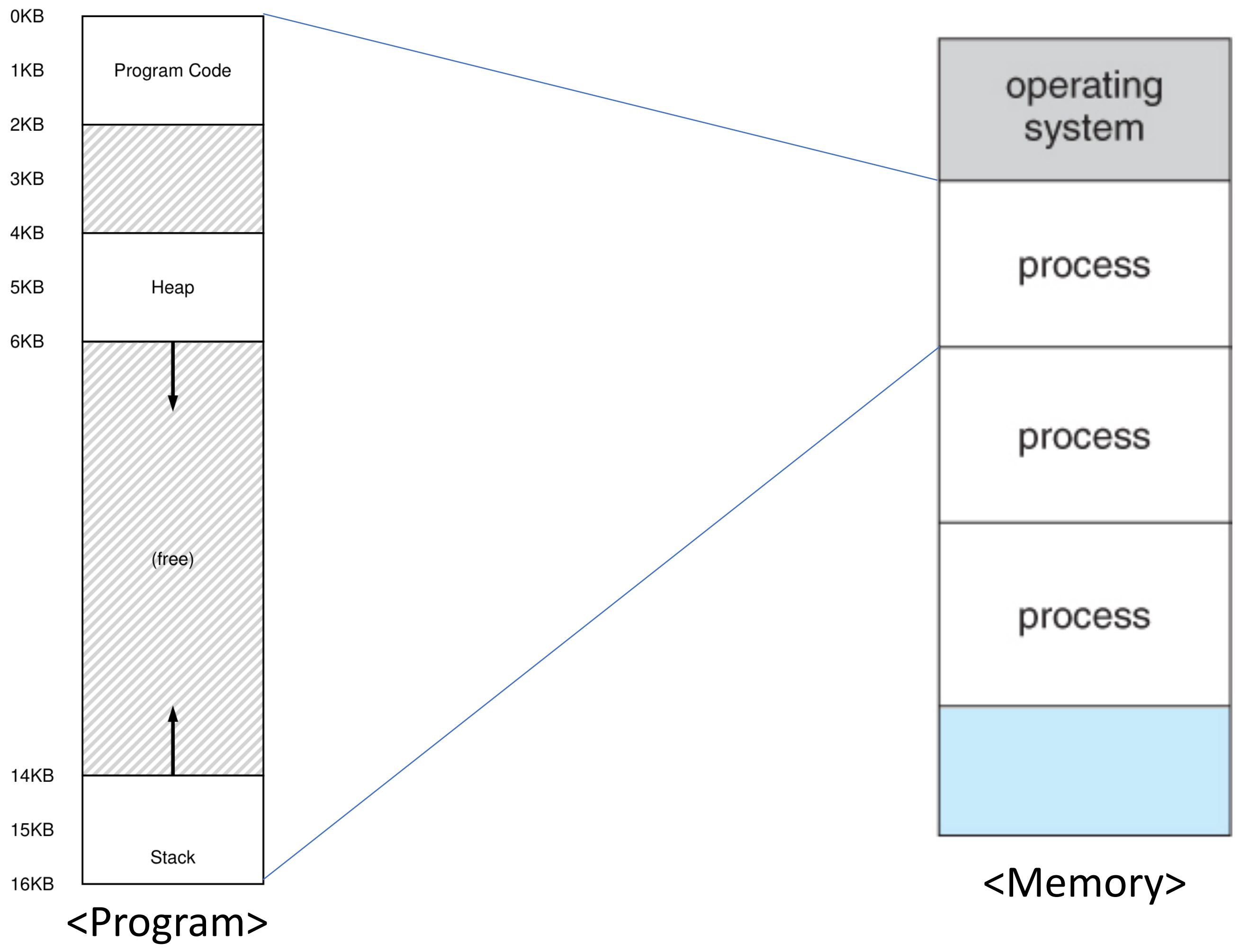
Memory Address Overview

- DRAM: byte-addressable
  e.g. address of 4th bytes in the memory: 0x04
  address of 5th bytes in the memory: 0x05
- A process(instructions) uses only virtual address



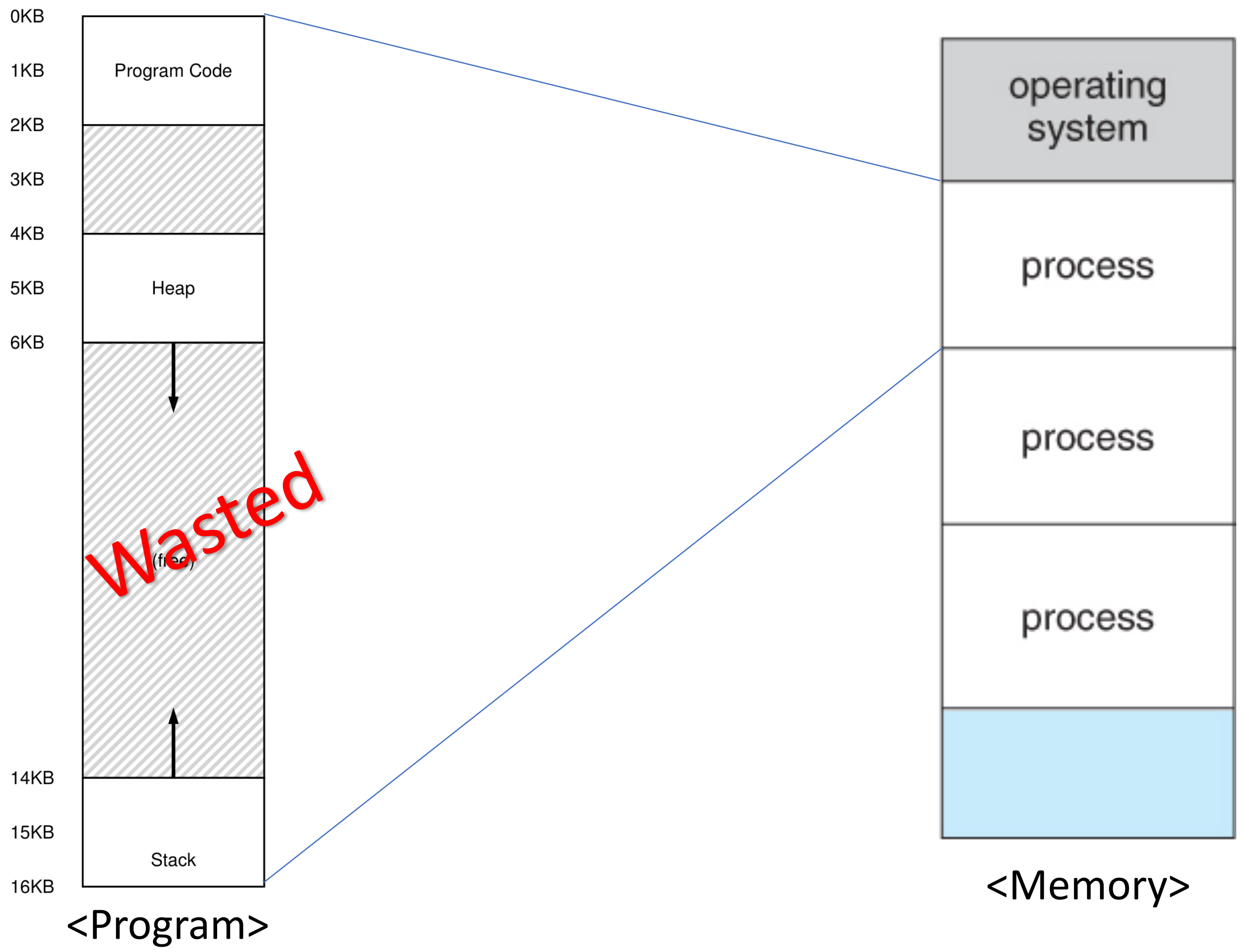
Segmentation

- Multiprogram era
- Multiple programs in the memory



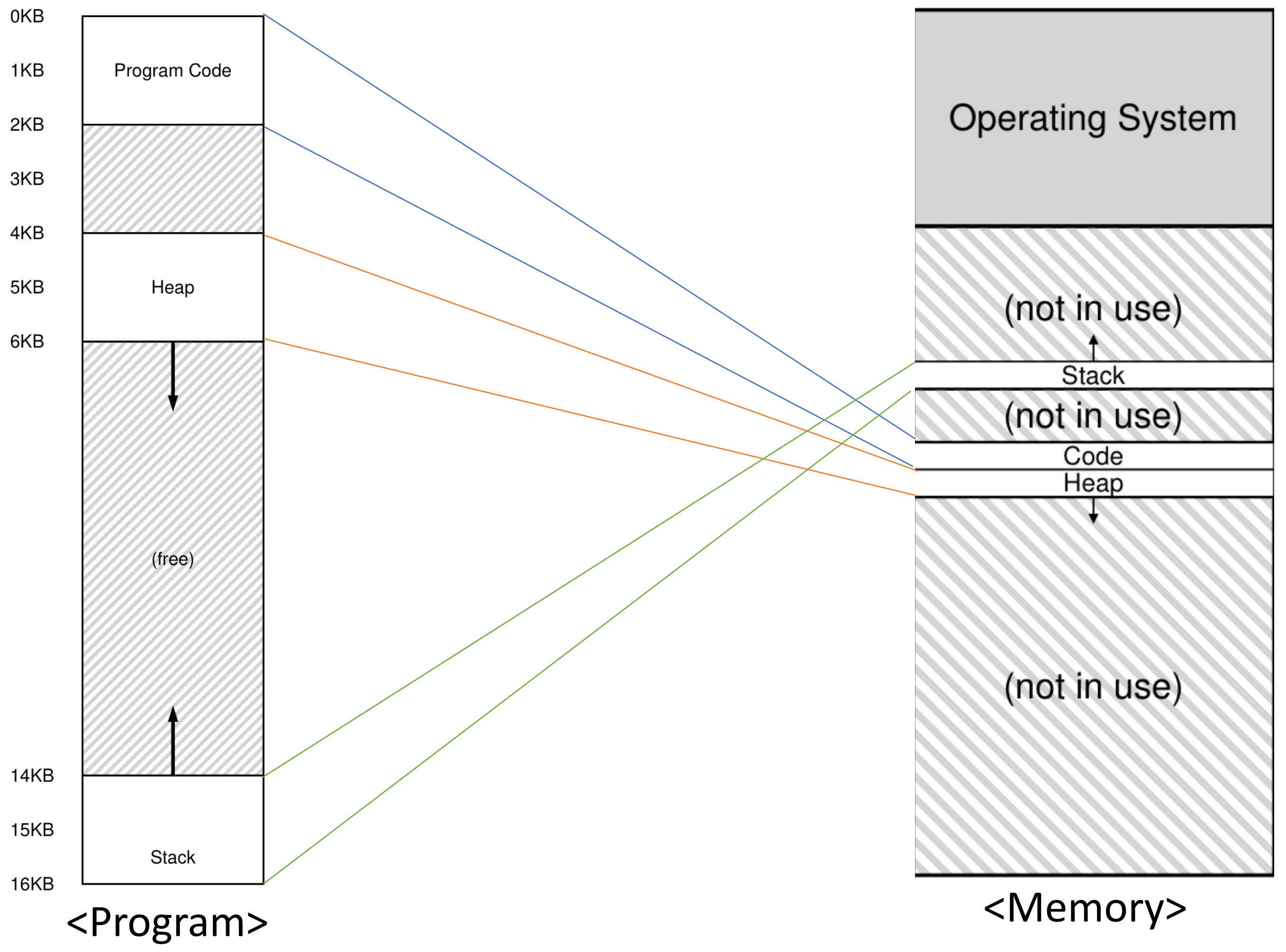
Segmentation

- Multiprogram era
- Multiple programs in the memory
- The free space between heap and stack is wasted



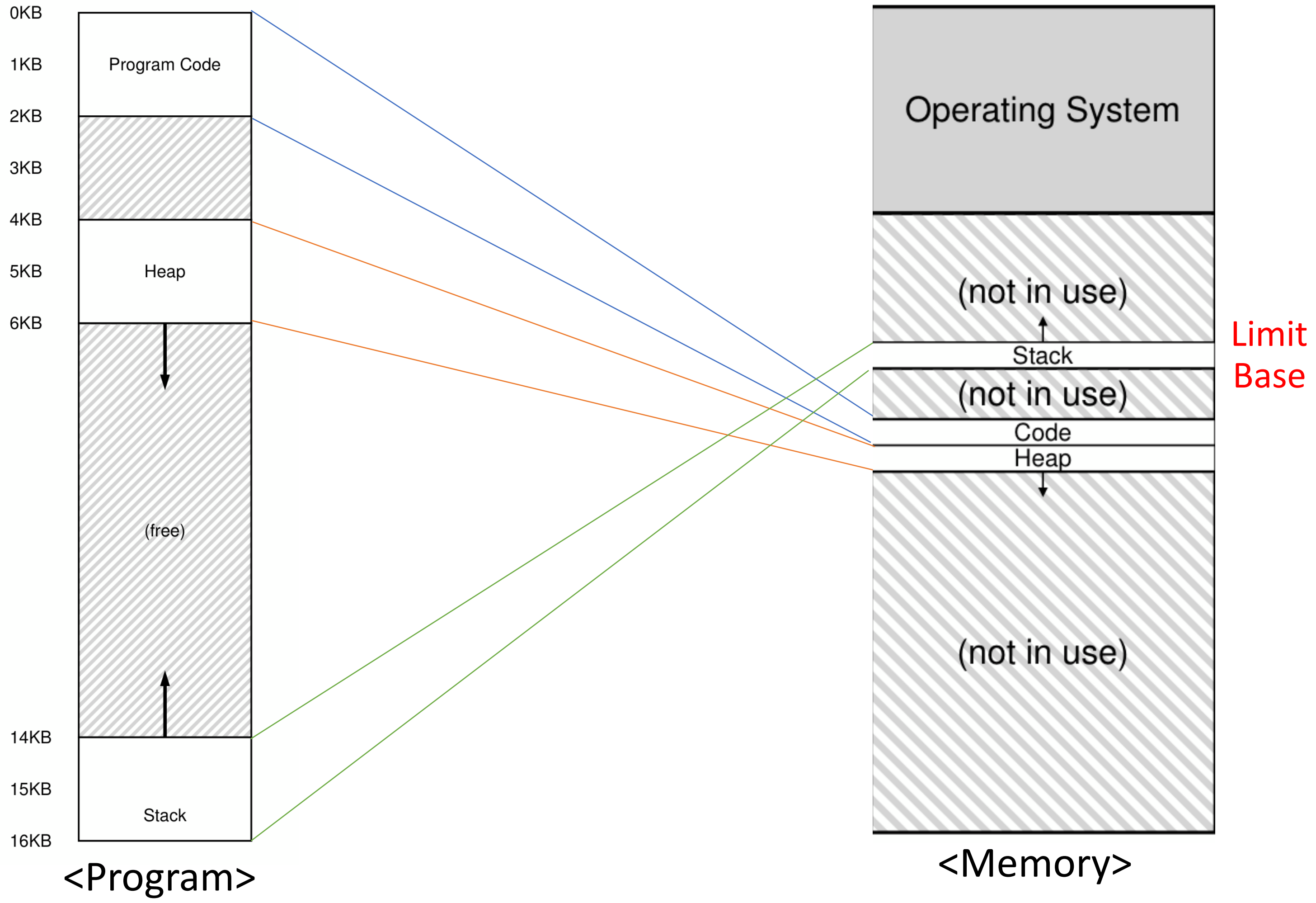
Segmentation

- Multiprogram era
- Multiple programs in the memory
- The free space between heap and stack is wasted
- Place each segment(code, stack, heap, …) into different memory region



Segmentation

- Multiprogram era
- Multiple programs in the memory
- The free space between heap and stack is wasted
- Place each segment(code, stack, heap, …) into different memory region

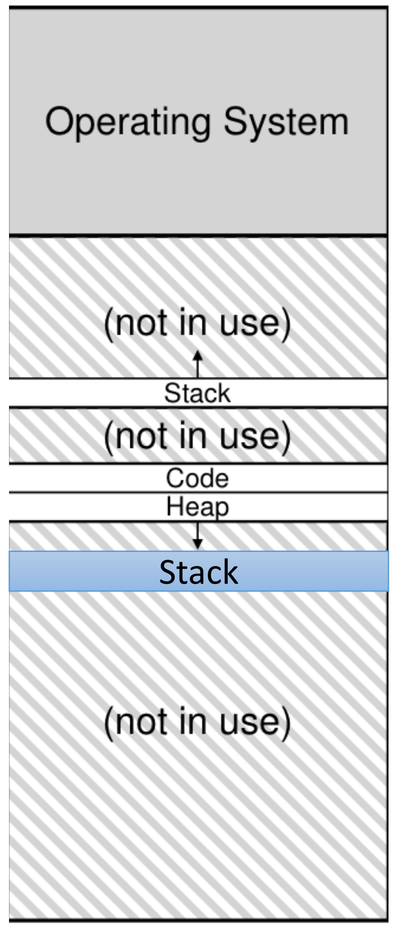


Global Descriptor Table (GDT)

- OS has one GDT
- Each program will receive a number of different segments
- Segment information is stored as a **segment descriptor** in GDT (32bits BASE + 20 bits LIMITS + 12 bits FLAGS)
- **Segment** = BASE + LIMITS
- **Flags** = Writable? Privilieged? …
- **Segment registers**--CS(Code Segment), DS(Data Segment), ..– contains the index of *segment descriptor* in GDT

Paging

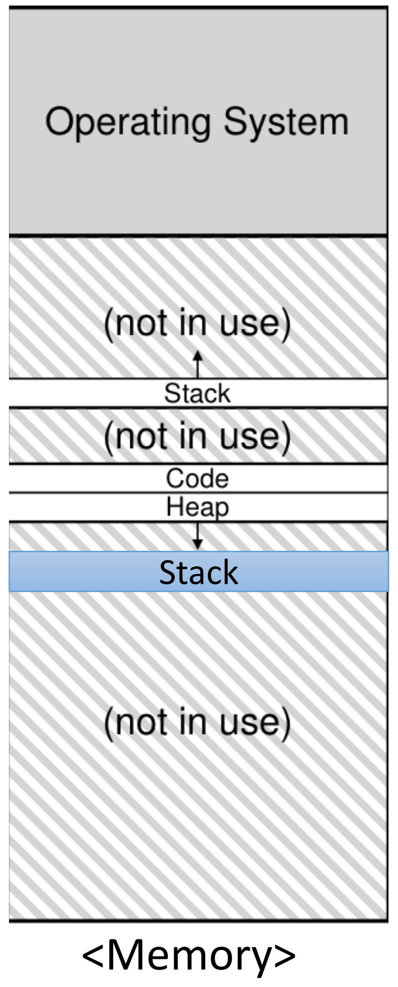
- What if segments are about to be overlapped?



<Memory>

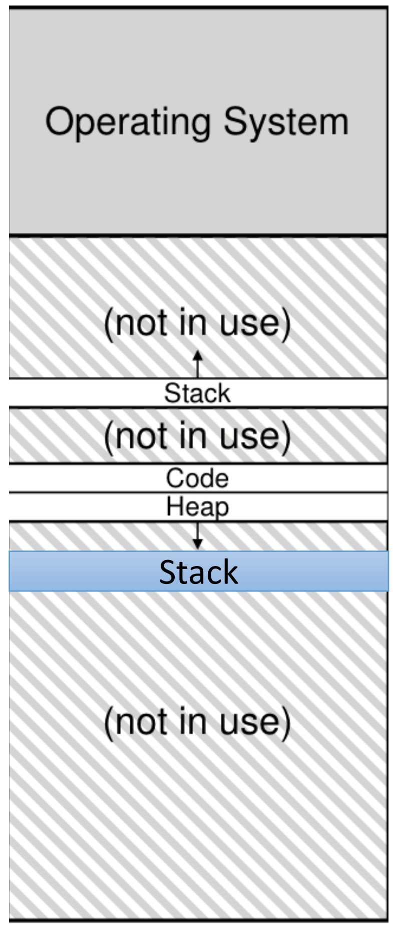
Paging

- What if segments are about to be overlapped?
- OS should find a free **contiguous** memory region and move the segment
- **Fragmentation** occurs
  - Sometimes the small space between segments is not large enough for a new segment → wasted
  - Moving the segment costs a lot (lots of memory operations)



Paging

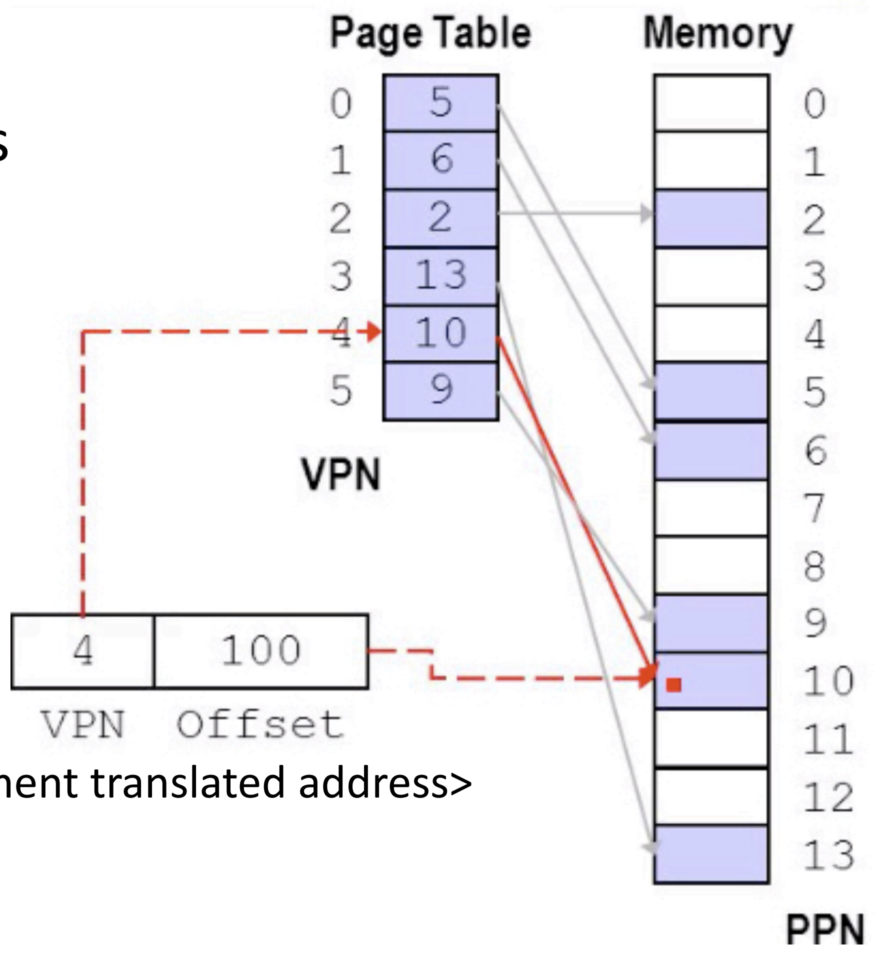
- What if segments are about to be overlapped?
- OS should find a free **contiguous** memory region and move the segment
- **Fragmentation** occurs
  - Sometimes the small space between segments is not large enough for the new segment → wasted
  - Moving the segment costs a lot (lots of memory operations)
- Solution?
  - Divide memory into **many fixed-size regions (pages)** and allocate this to segment dynamically
  - by paging hw



<Memory>

Flat Page Table

- Page table entry contains physical page address
- A virtual address contains page table index



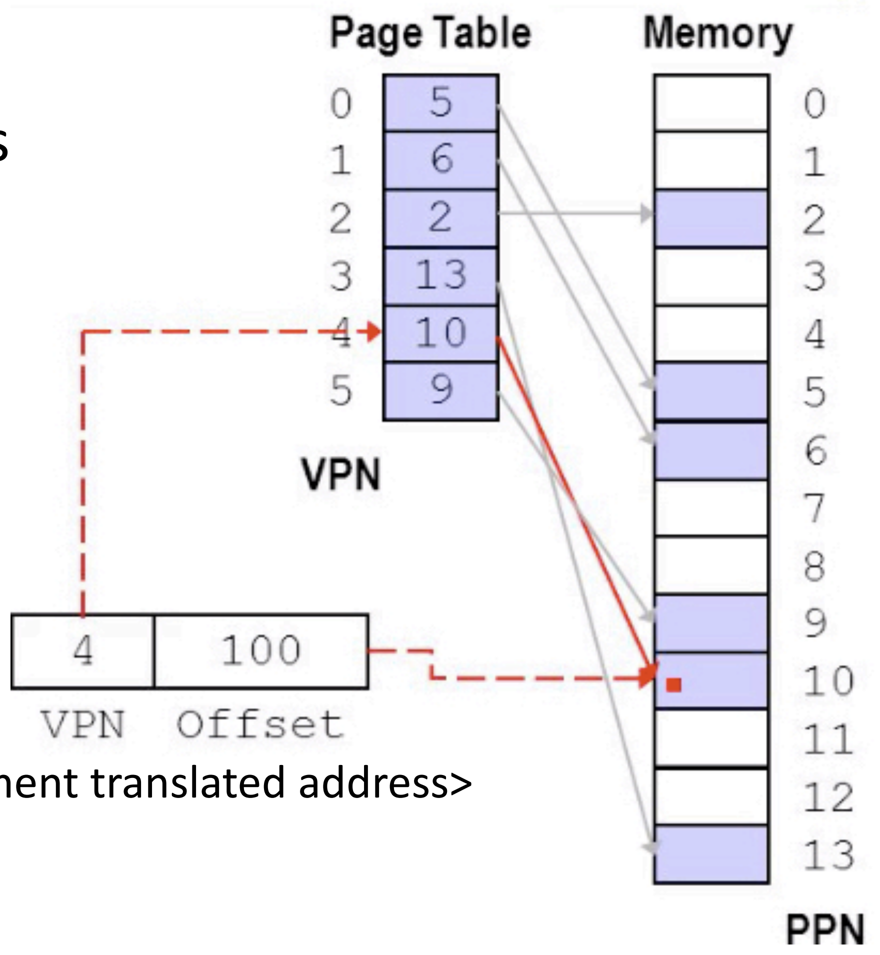
<Segment translated address>

<Memory>

Flat Page Table

- Page table entry contains physical page address
- A virtual address contains page table index
- 32-bit address can represent 4GB space
  4GB = 4kb * 1 million
  4 bytes(page table entry) * 1 million = **4MB**

- Page table is stored in memory
- Each program has its own page table
- 100 programs running ?

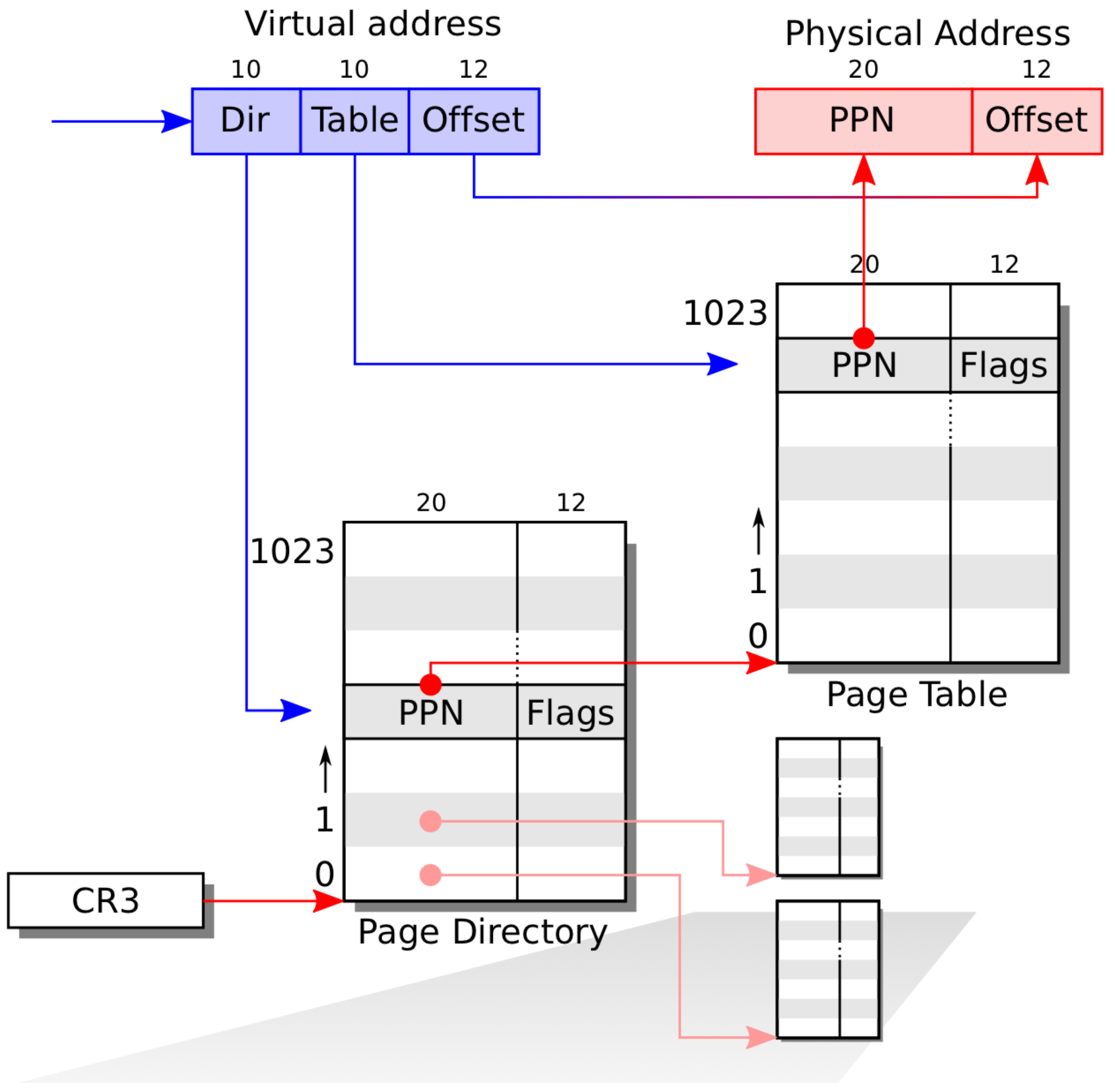


<Segment translated address>

<Memory>

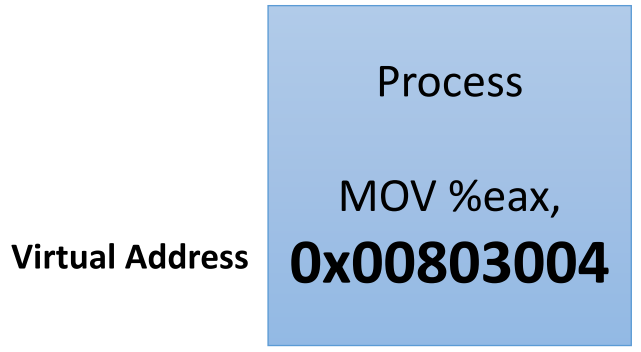
Paging Hardware

- x86 page table = an array of 2^20 **Page Table Entries (PTEs)**
- PTE: 20-bint **Physical Page Number (PPN)**
- Top 20 bits of virtual address = index of page table
- **Page Directory**: contains reference to page table
- **Page Fault**: PTE_P(PAGE PRESENT) is not set



A Simple Example

- x86, 4k page
- Logical address 0x803004 → Physical address 0x8004
- Physical address of Page Directory: 0x5000
- Physical address of the page table involved: 0x8000
- entry[1] in Global Descriptor Table: 0x1000000, 2GB
- DS register: 0x8
- **Draw the diagram of process translation**



```
MOV %eax,
0x00803004
```

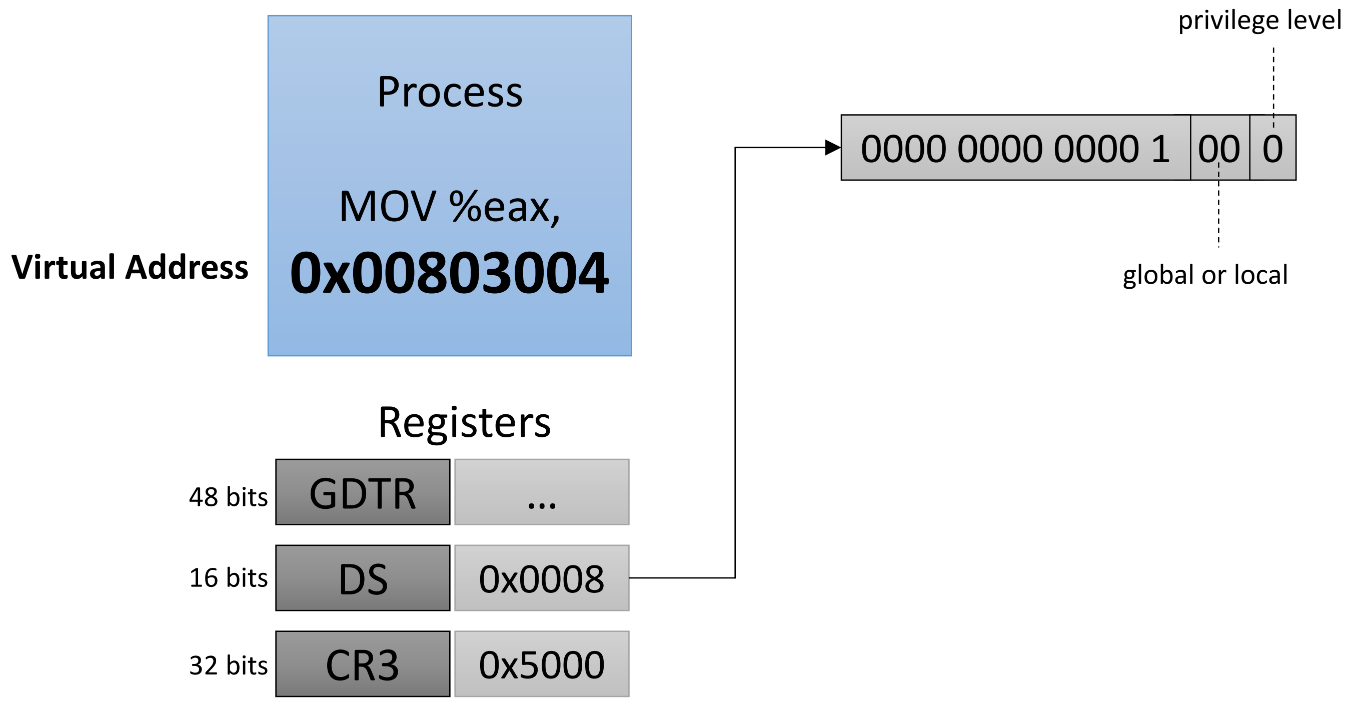
Process

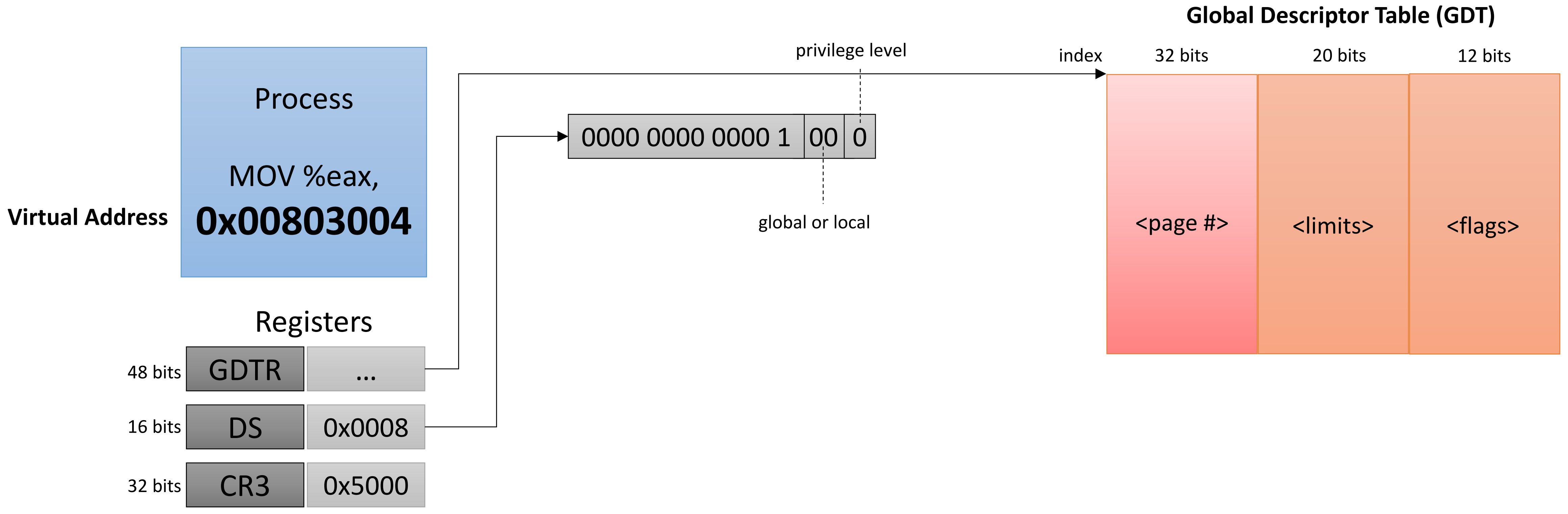
MOV %eax,

Virtual Address **0x00803004**

Registers

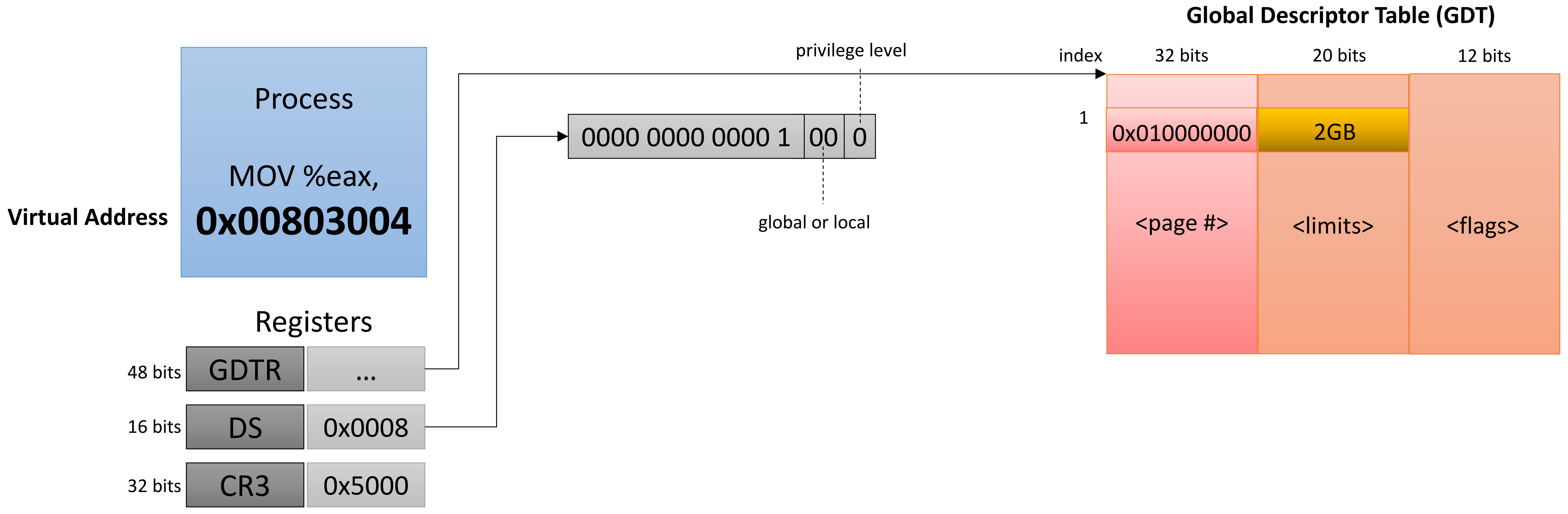
48 bits	GDTR	…
16 bits	DS	0x0008
32 bits	CR3	0x5000

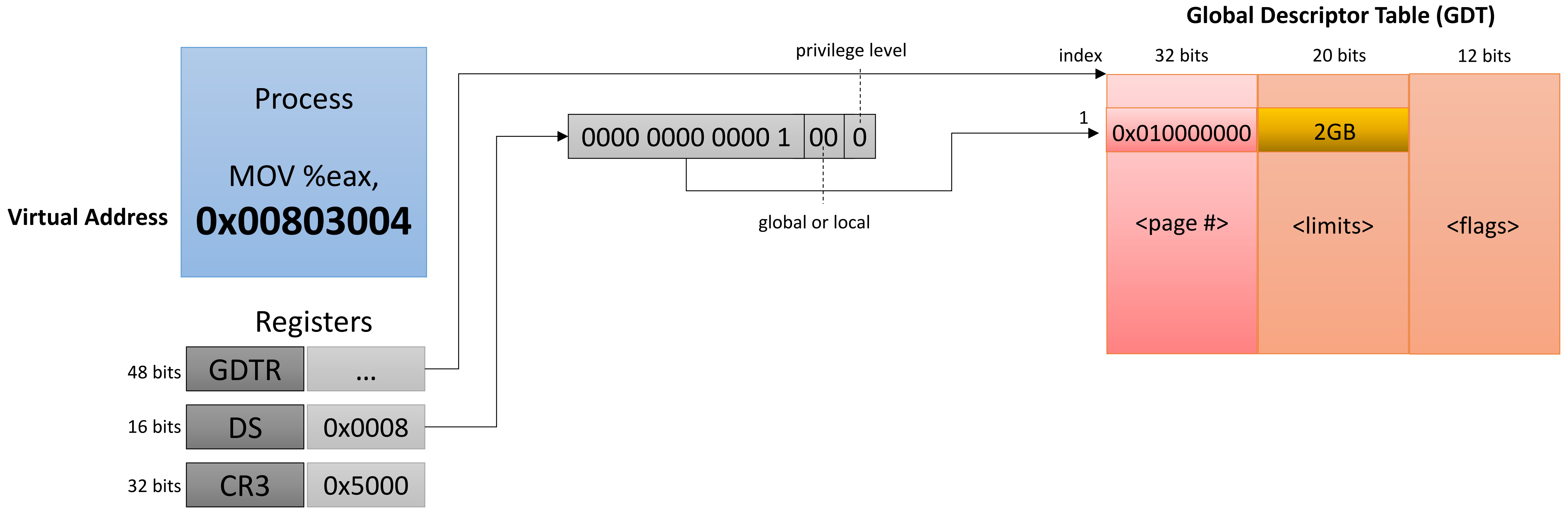

Process
MOV %eax,
Virtual Address
0x00803004
Registers
48 bits
GDTR
...
16 bits
DS
0x0008
32 bits
CR3
0x5000
privilege level
0000 0000 0000 1
00
0
global or local

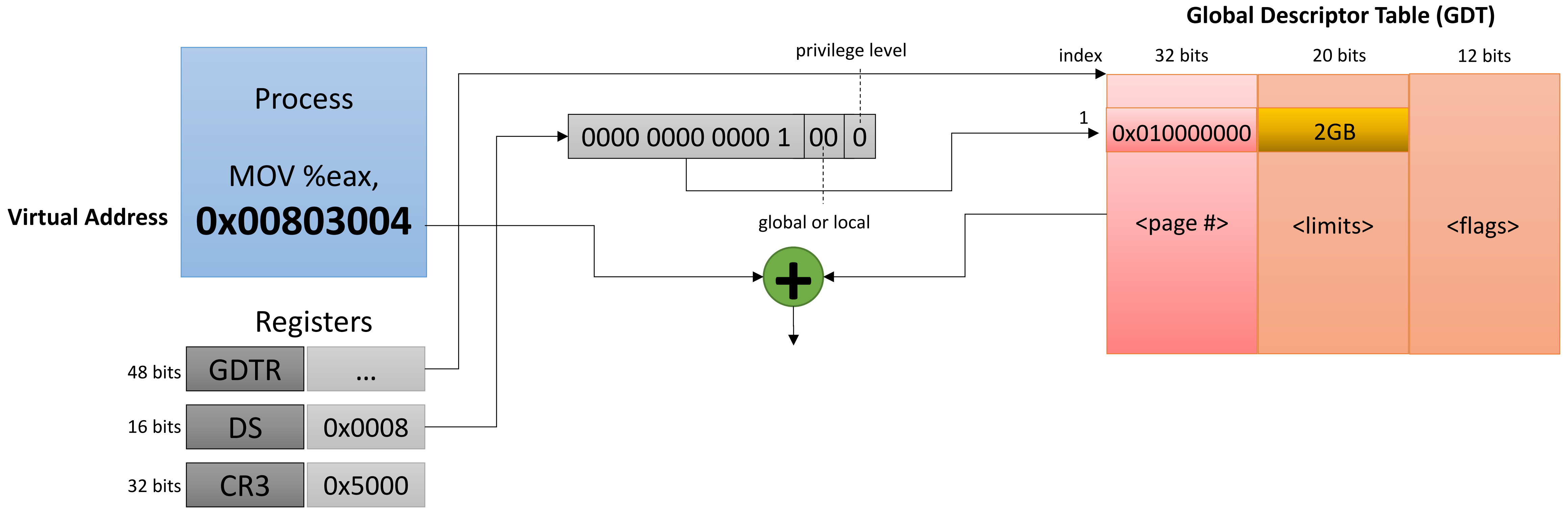

Global Descriptor Table (GDT)
Process
MOV %eax,
Virtual Address
0x00803004
privilege level
index
32 bits
20 bits
12 bits
0000 0000 0000 1
00
0
global or local
<page #>
<limits>
<flags>
Registers
48 bits
GDTR
...
16 bits
DS
0x0008
32 bits
CR3
0x5000

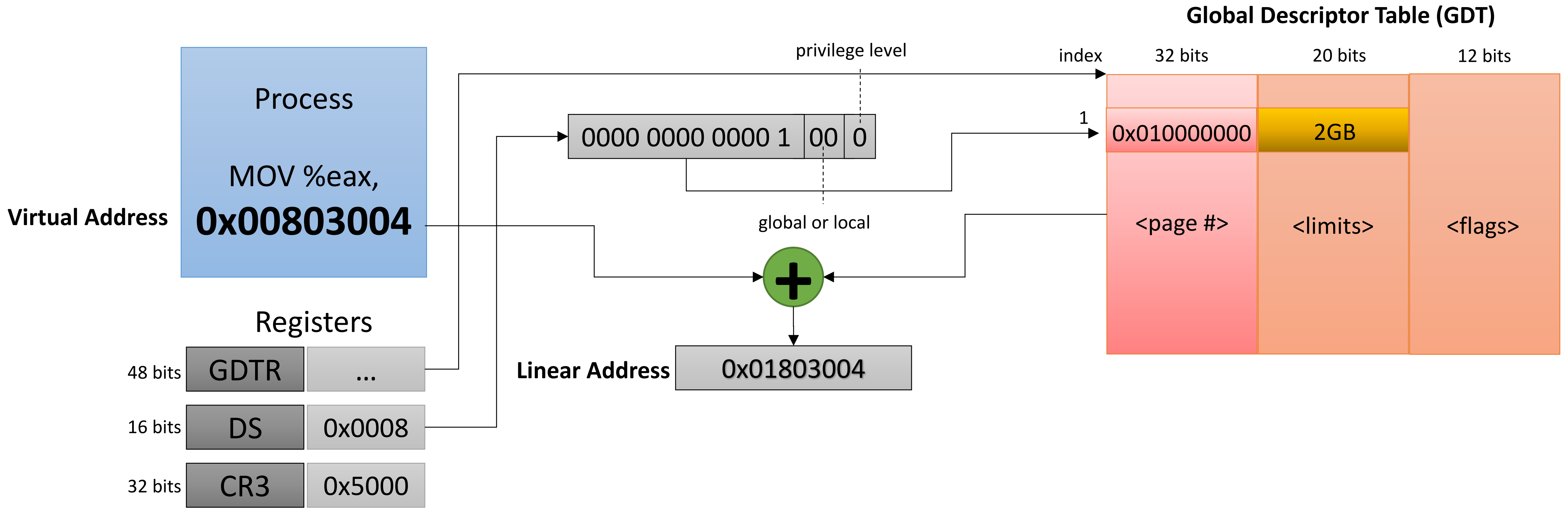
A Simple Example

- x86, 4k page
- Logical address 0x803004 → Physical address 0x8004
- Physical address of Page Directory: 0x5000
- Physical address of the page table involved: 0x8000
- entry[1] in Global Descriptor Table: 0x1000000, 2GB
- DS register: 0x8
- **Draw the diagram of process translation**

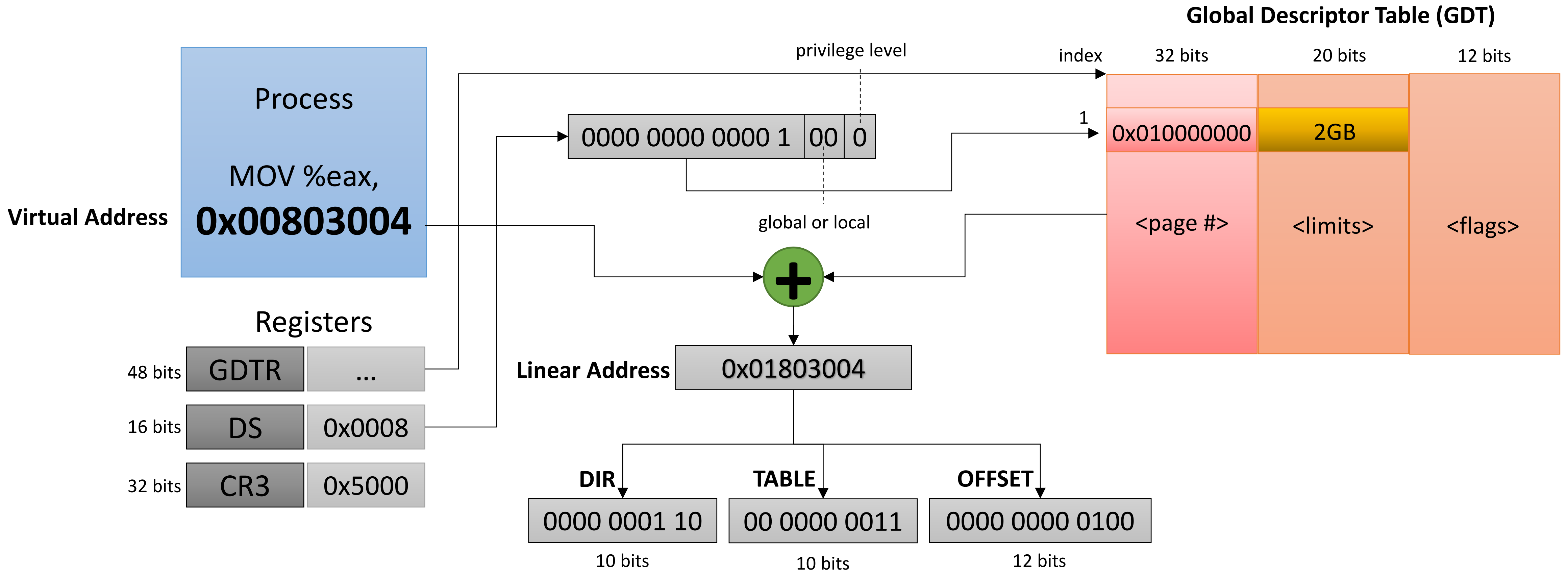

Global Descriptor Table (GDT)
Process
MOV %eax,
Virtual Address
0x00803004
Registers
48 bits
GDTR
...
16 bits
DS
0x0008
32 bits
CR3
0x5000
privilege level
0000 0000 0000 1
00
0
global or local
index
32 bits
20 bits
12 bits
1
0x010000000
2GB
<page #>
<limits>
<flags>

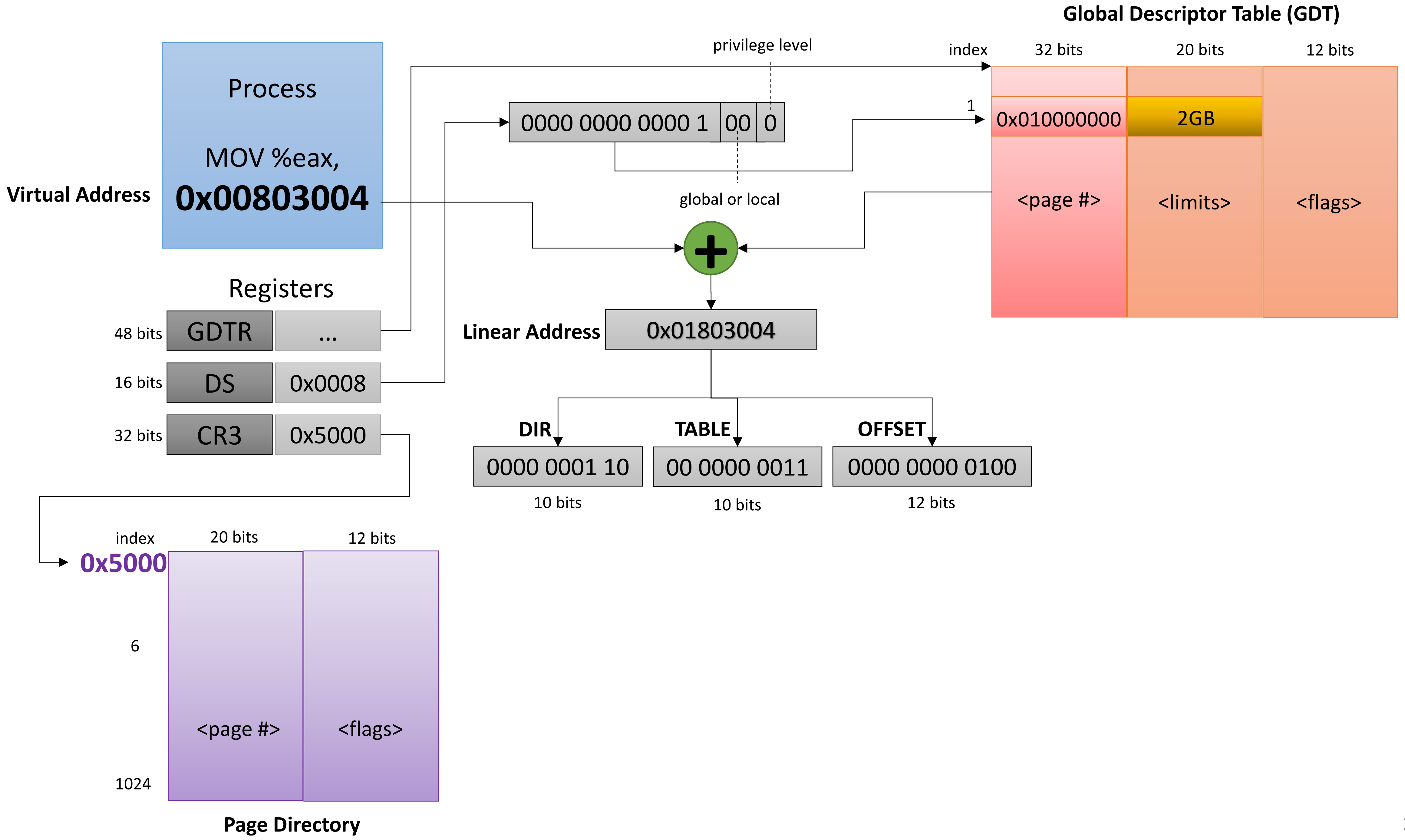

Global Descriptor Table (GDT)
privilege level
index
32 bits
20 bits
12 bits
Process
MOV %eax,
Virtual Address
0x00803004
0000 0000 0000 1
00
0
1
0x010000000
2GB
global or local
<page #>
<limits>
<flags>
Registers
48 bits
GDTR
...
16 bits
DS
0x0008
32 bits
CR3
0x5000


Global Descriptor Table (GDT)
privilege level
index
32 bits
20 bits
12 bits
Process
MOV %eax,
Virtual Address
0x00803004
0000 0000 0000 1
00
0
1
0x010000000
2GB
global or local
<page #>
<limits>
<flags>
Registers
48 bits
GDTR
...
16 bits
DS
0x0008
32 bits
CR3
0x5000

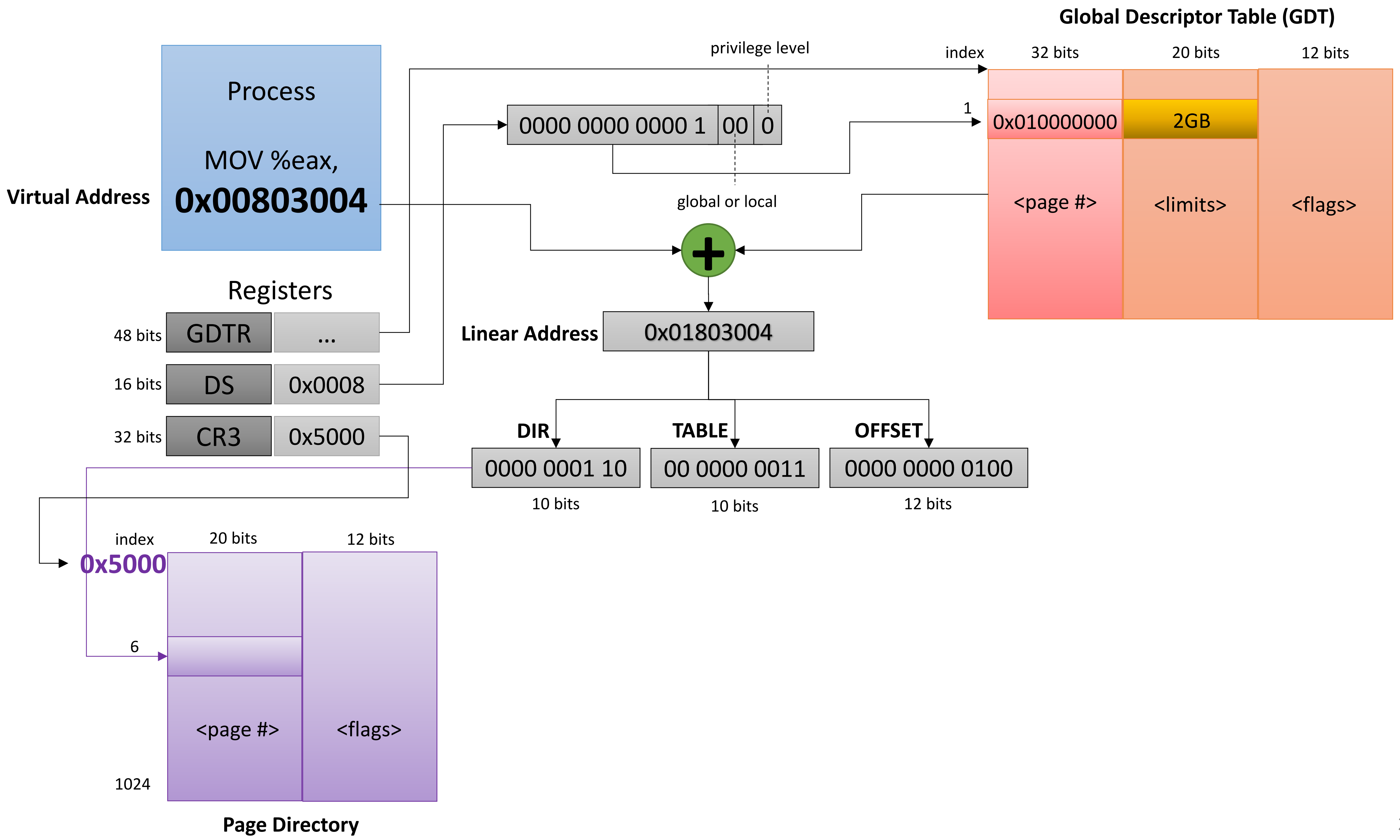


Global Descriptor Table (GDT)
privilege level
index
32 bits
20 bits
12 bits
Process
MOV %eax,
0x00803004
Virtual Address
0000 0000 0000 1
00
0
1
0x010000000
2GB
global or local
<page #>
<limits>
<flags>
Registers
48 bits
GDTR
...
16 bits
DS
0x0008
32 bits
CR3
0x5000
Linear Address
0x01803004


Global Descriptor Table (GDT)
privilege level
index
32 bits
20 bits
12 bits
Process
MOV %eax,
0x00803004
Virtual Address
0000 0000 0000 1
00
0
1
0x010000000
2GB
global or local
<page #>
<limits>
<flags>
Registers
48 bits
GDTR
...
16 bits
DS
0x0008
32 bits
CR3
0x5000
Linear Address
0x01803004
DIR
TABLE
OFFSET
0000 0001 10
00 0000 0011
0000 0000 0100
10 bits
10 bits
12 bits

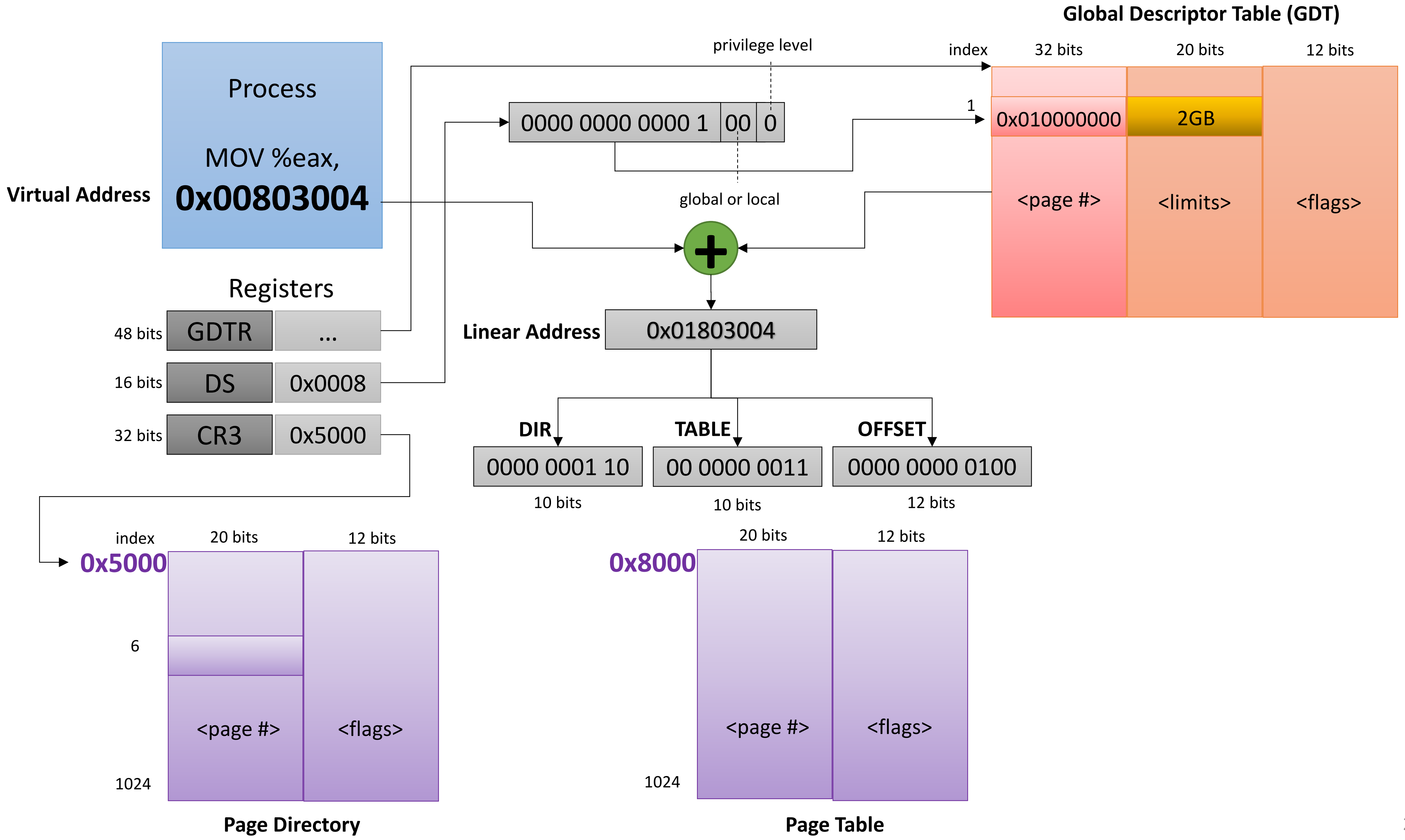


Global Descriptor Table (GDT)
privilege level
index
32 bits
20 bits
12 bits
Process
MOV %eax,
Virtual Address
0x00803004
0000 0000 0000 1
00
0
1
0x010000000
2GB
global or local
<page #>
<limits>
<flags>
Registers
48 bits
GDTR
...
Linear Address
0x01803004
16 bits
DS
0x0008
32 bits
CR3
0x5000
DIR
TABLE
OFFSET
0000 0001 10
00 0000 0011
0000 0000 0100
10 bits
10 bits
12 bits
index
20 bits
12 bits
0x5000
6
<page #>
<flags>
1024
Page Directory

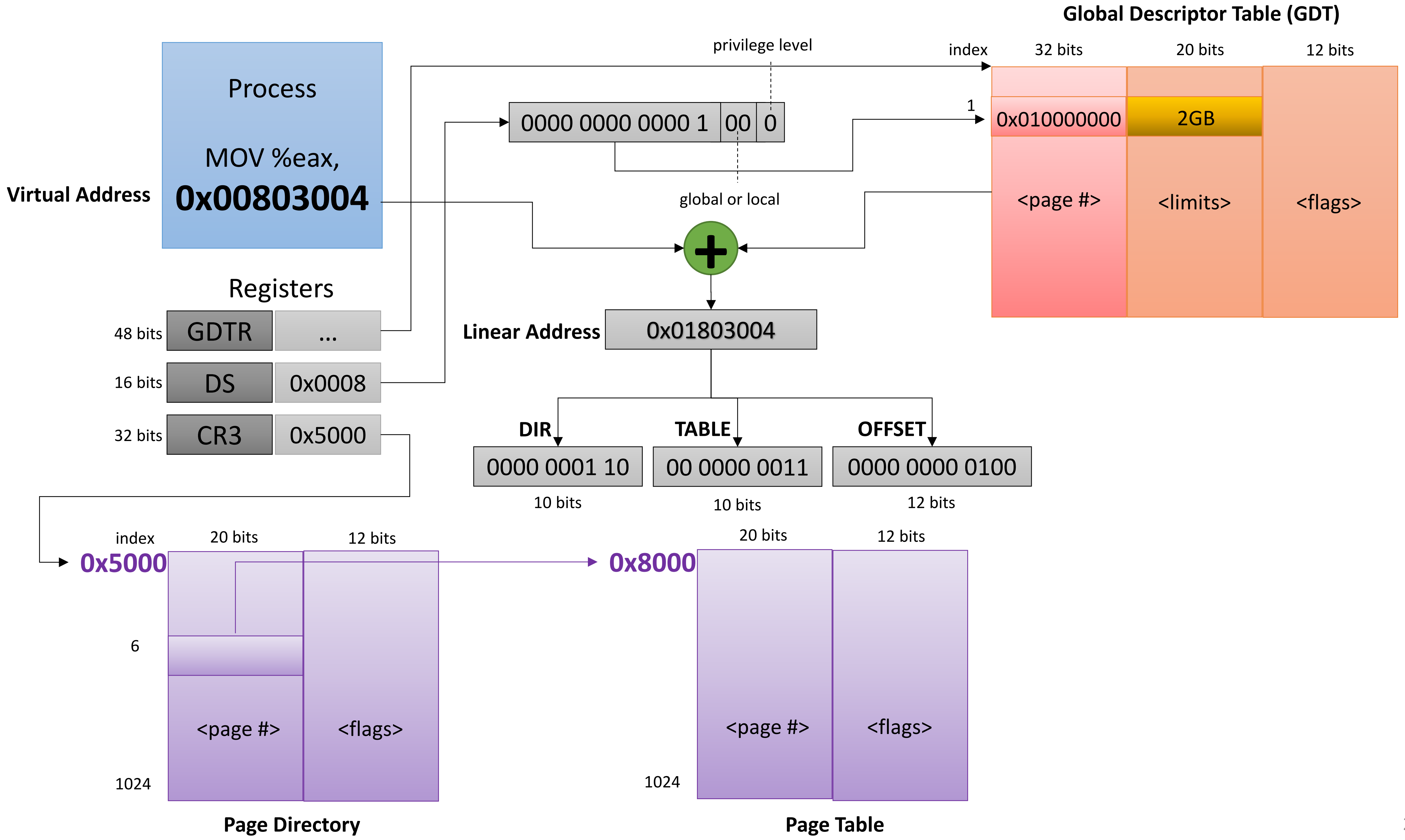

Global Descriptor Table (GDT)
privilege level
index
32 bits
20 bits
12 bits
Process
MOV %eax,
0x00803004
Virtual Address
0000 0000 0000 1
00
0
1
0x010000000
2GB
global or local
<page #>
<limits>
<flags>
Registers
48 bits
GDTR
...
16 bits
DS
0x0008
32 bits
CR3
0x5000
Linear Address
0x01803004
DIR
TABLE
OFFSET
0000 0001 10
00 0000 0011
0000 0000 0100
10 bits
10 bits
12 bits
index
20 bits
12 bits
0x5000
6
<page #>
<flags>
1024
Page Directory

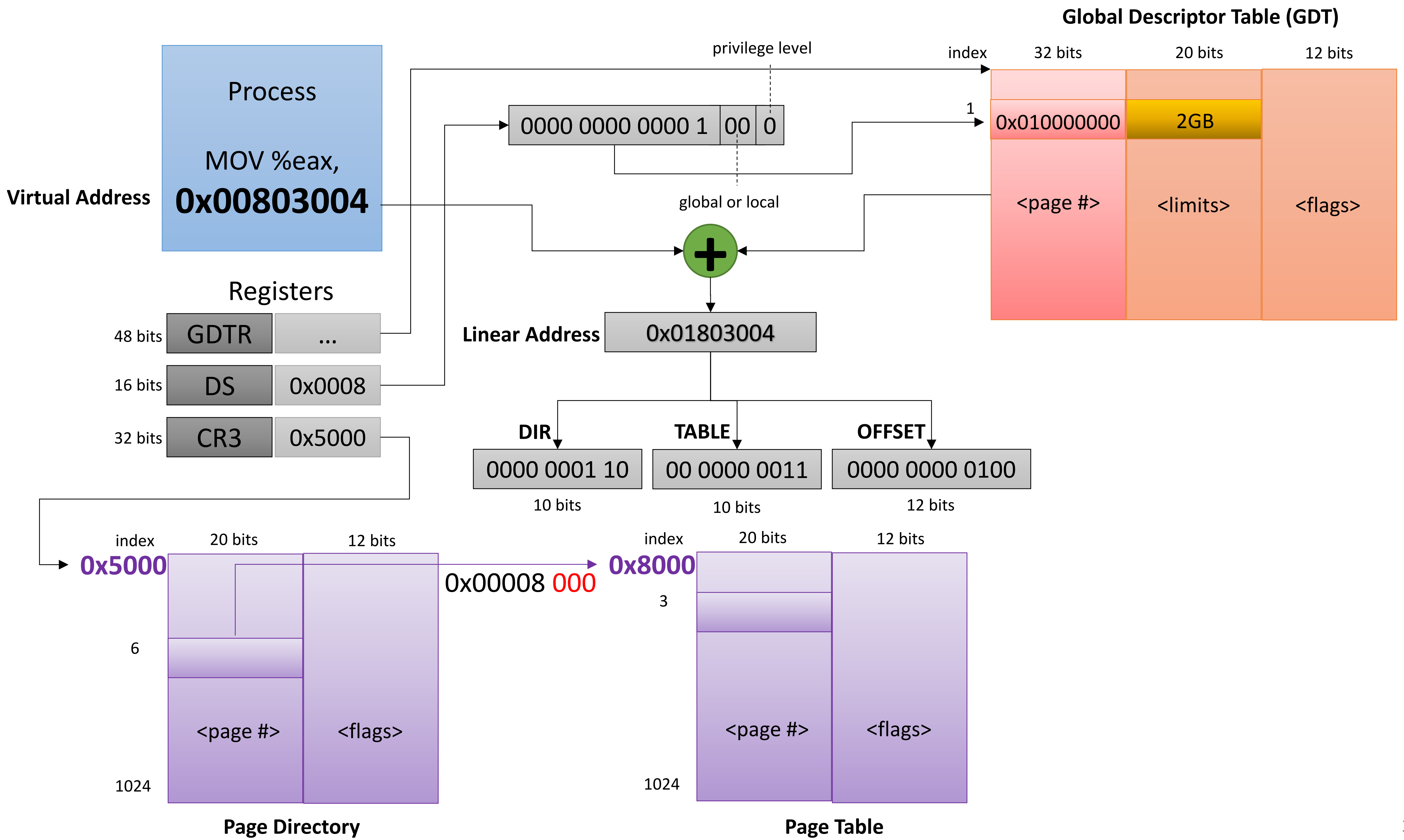
A Simple Example

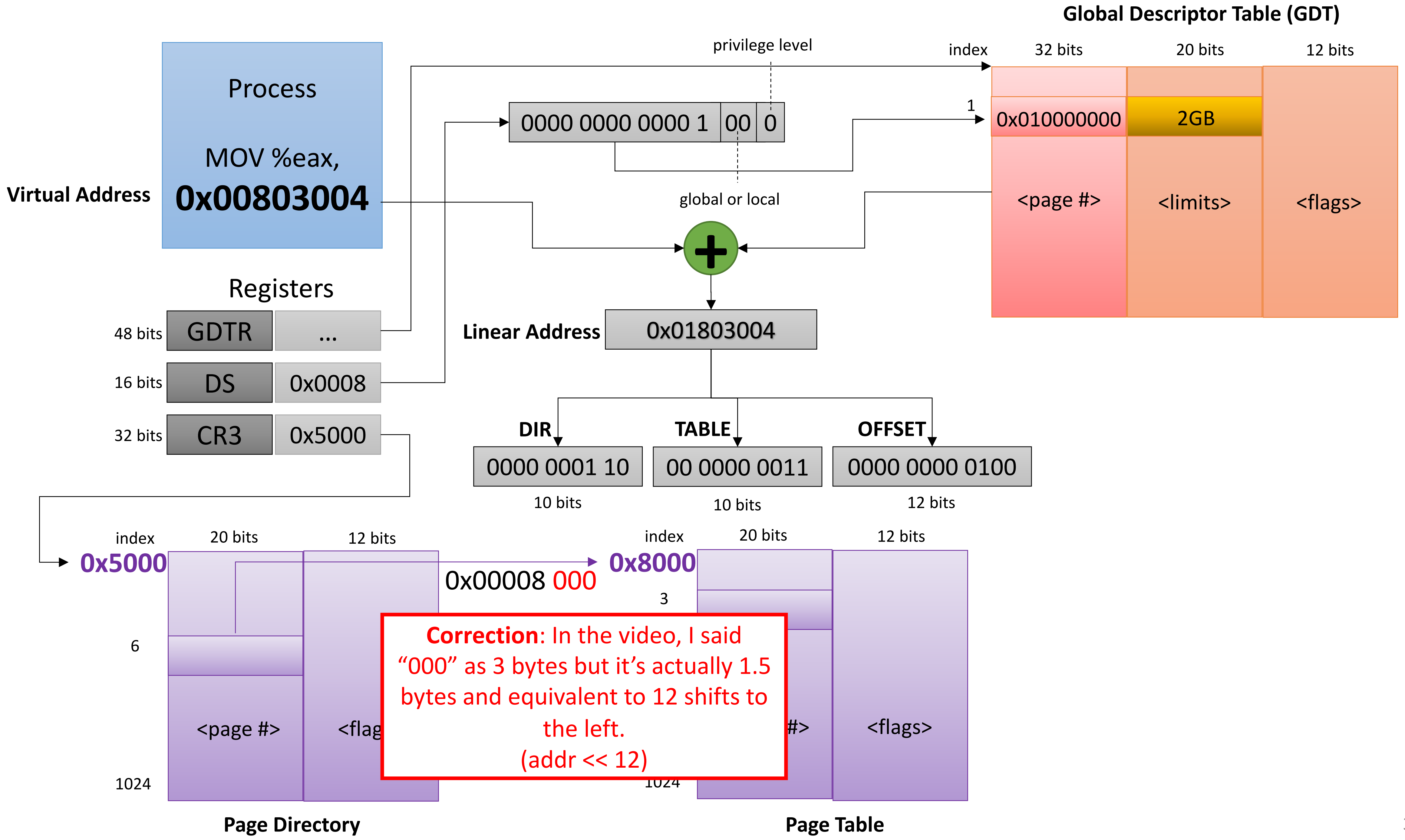
- x86, 4k page
- Logical address 0x803004 → Physical address 0x8004
- Physical address of Page Directory: 0x5000
- Physical address of the page table involved: 0x8000
- entry[1] in Global Descriptor Table: 0x1000000, 2GB
- DS register: 0x8
- **Draw the diagram of process translation**



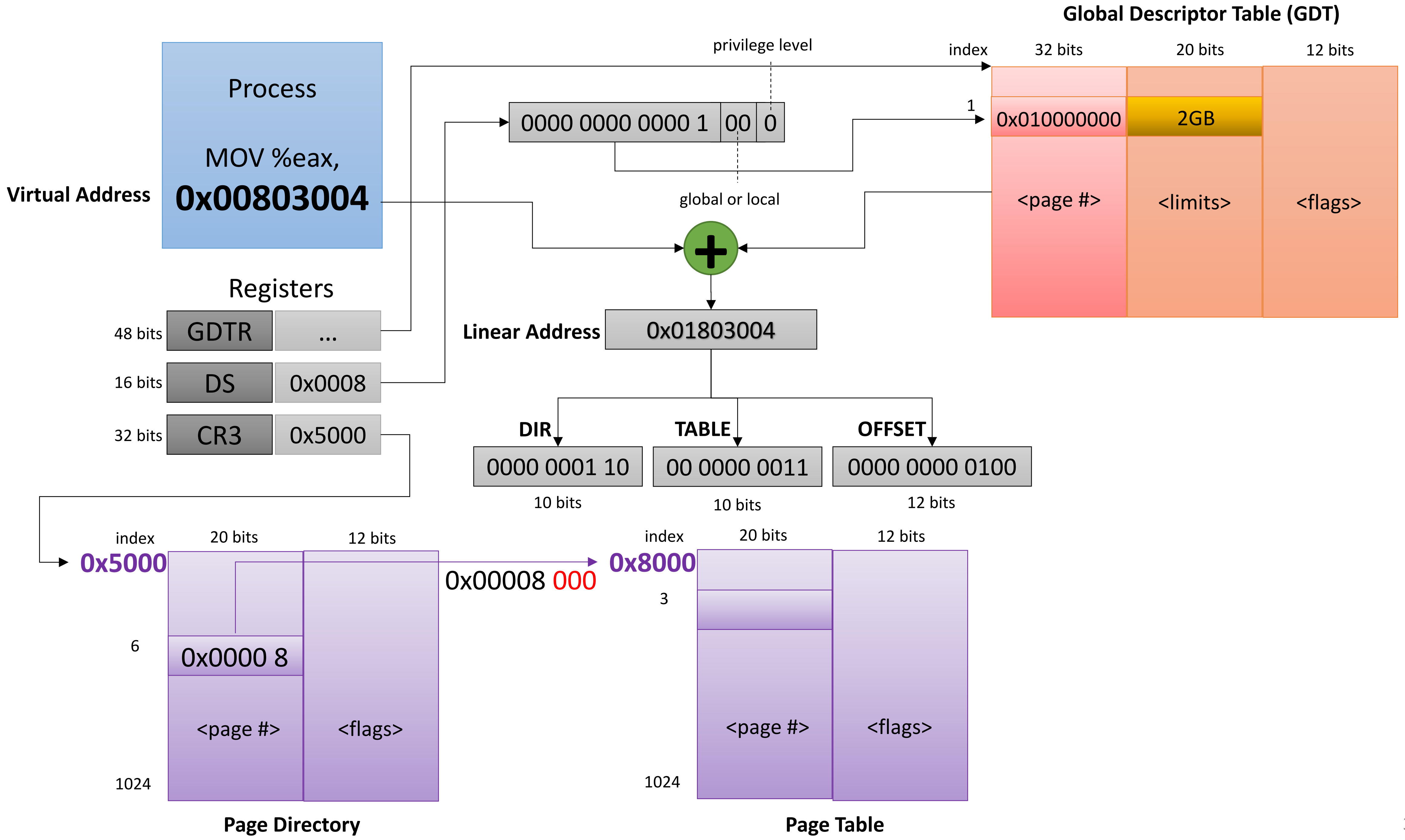
Global Descriptor Table (GDT)
index
32 bits
20 bits
12 bits
1
0x010000000
2GB
<page #>
<limits>
<flags>
privilege level
0000 0000 0000 1
00
0
global or local
Process
MOV %eax,
0x00803004
Virtual Address
Registers
48 bits
GDTR
...
16 bits
DS
0x0008
32 bits
CR3
0x5000
Linear Address
0x01803004
DIR
TABLE
OFFSET
0000 0001 10
00 0000 0011
0000 0000 0100
10 bits
10 bits
12 bits
index
20 bits
12 bits
0x5000
6
<page #>
<flags>
1024
Page Directory
20 bits
12 bits
0x8000
<page #>
<flags>
1024
Page Table


Global Descriptor Table (GDT)
index
32 bits
20 bits
12 bits
privilege level
Process
MOV %eax,
Virtual Address
0x00803004
0000 0000 0000 1
00
0
1
0x010000000
2GB
global or local
<page #>
<limits>
<flags>
Registers
48 bits
GDTR
...
16 bits
DS
0x0008
32 bits
CR3
0x5000
Linear Address
0x01803004
DIR
TABLE
OFFSET
0000 0001 10
00 0000 0011
0000 0000 0100
10 bits
10 bits
12 bits
index
20 bits
12 bits
0x5000
6
<page #>
<flags>
1024
Page Directory
0x8000
20 bits
12 bits
<page #>
<flags>
1024
Page Table


Global Descriptor Table (GDT)
privilege level
index
32 bits
20 bits
12 bits
Process
MOV %eax,
0x00803004
Virtual Address
0000 0000 0000 1
00
0
1
0x010000000
2GB
global or local
<page #>
<limits>
<flags>
Registers
48 bits
GDTR
...
16 bits
DS
0x0008
32 bits
CR3
0x5000
Linear Address
0x01803004
DIR
TABLE
OFFSET
0000 0001 10
00 0000 0011
0000 0000 0100
10 bits
10 bits
12 bits
index
20 bits
12 bits
0x5000
6
1024
<page #>
<flags>
0x00008 000
index
20 bits
12 bits
0x8000
3
1024
<page #>
<flags>
Page Directory
Page Table

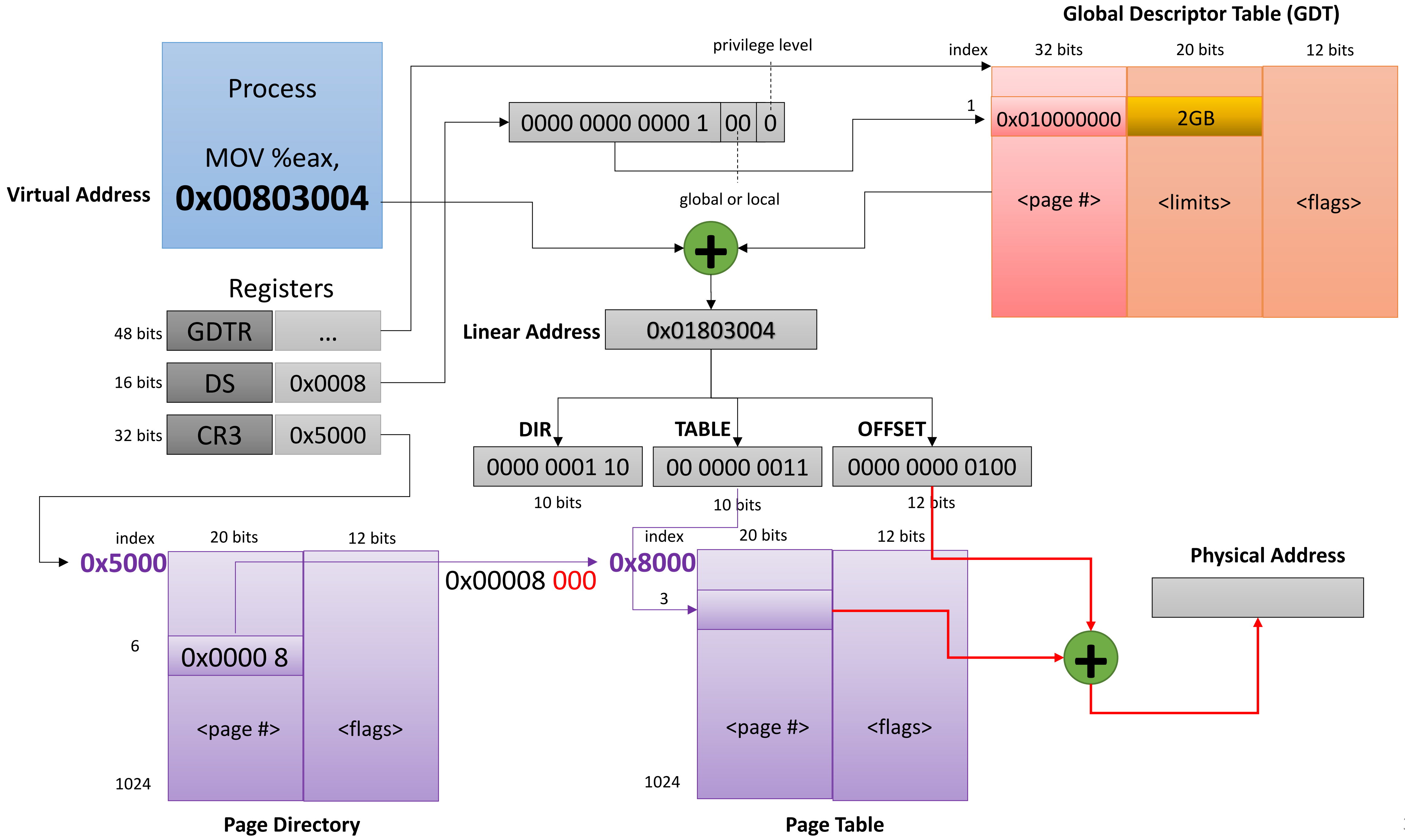


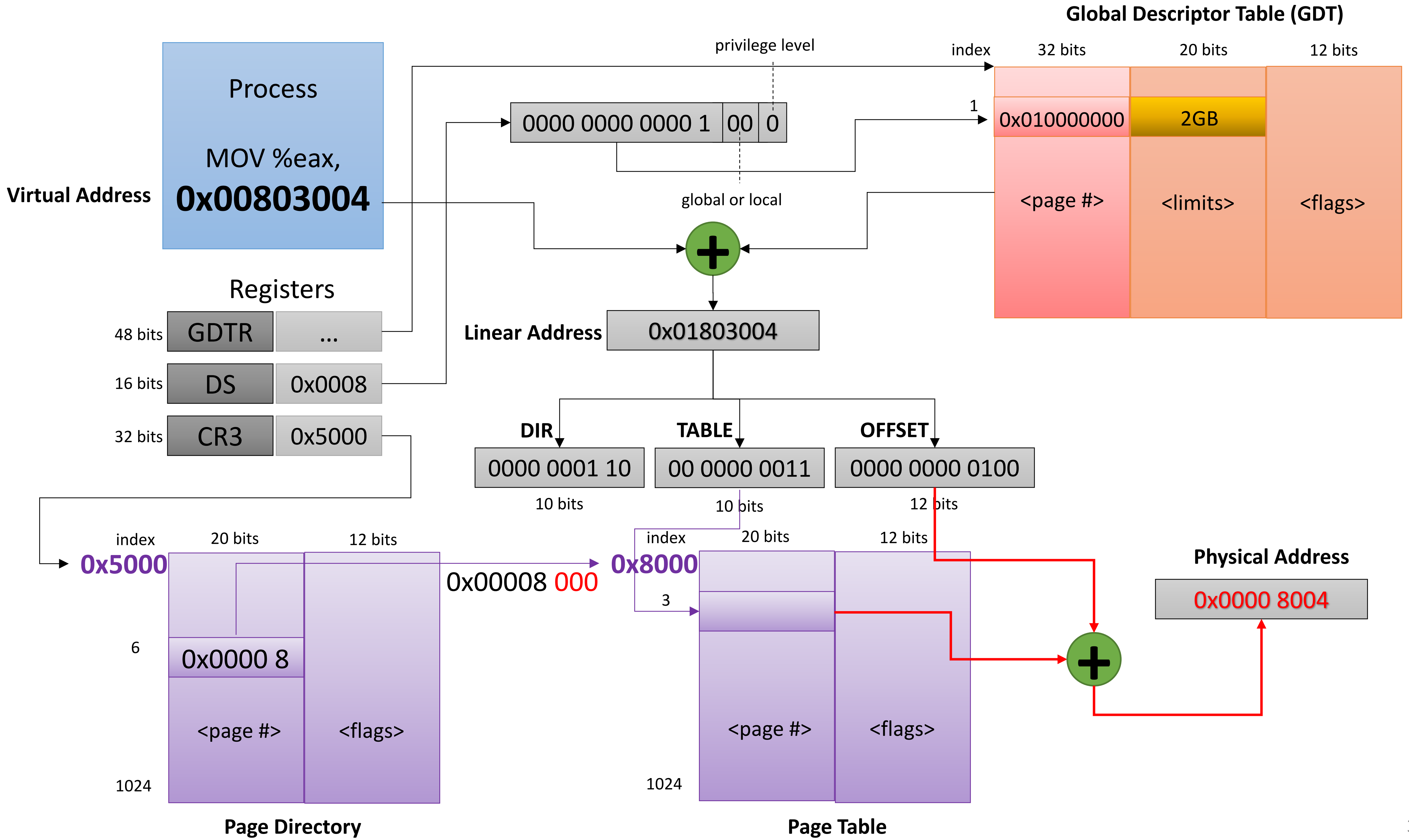
Correction: In the video, I said "000" as 3 bytes but it's actually 1.5 bytes and equivalent to 12 shifts to the left.
(addr << 12)

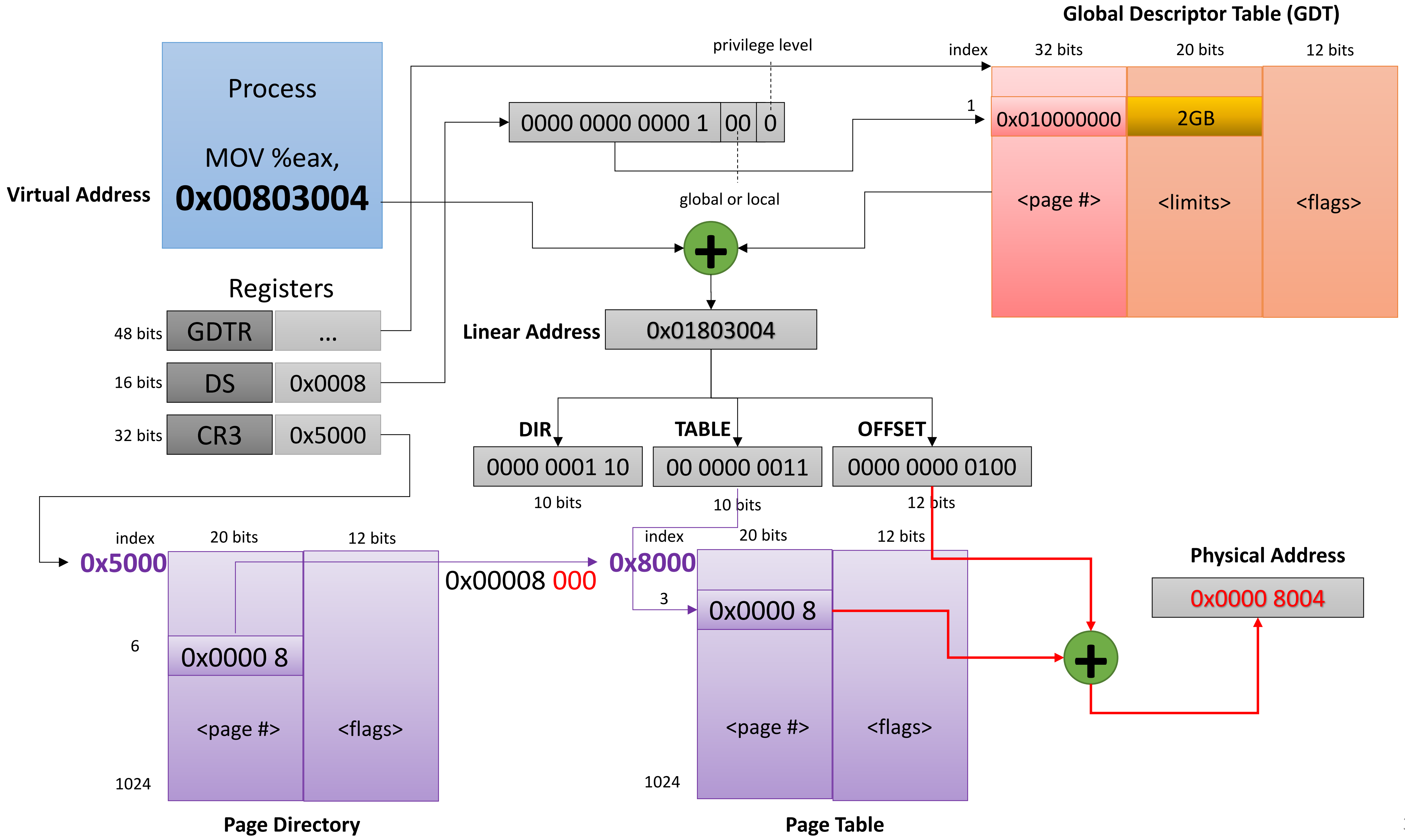

Global Descriptor Table (GDT)
index
32 bits
20 bits
12 bits
1
0x010000000
2GB
<page #>
<limits>
<flags>
privilege level
0000 0000 0000 1
00
0
global or local
Process
MOV %eax,
Virtual Address
0x00803004
Registers
48 bits
GDTR
...
16 bits
DS
0x0008
32 bits
CR3
0x5000
Linear Address
0x01803004
DIR
TABLE
OFFSET
0000 0001 10
00 0000 0011
0000 0000 0100
10 bits
10 bits
12 bits
index
20 bits
12 bits
0x5000
6
0x0000 8
<page #>
<flags>
1024
Page Directory
0x00008 000
index
20 bits
12 bits
0x8000
3
<page #>
<flags>
1024
Page Table

A Simple Example

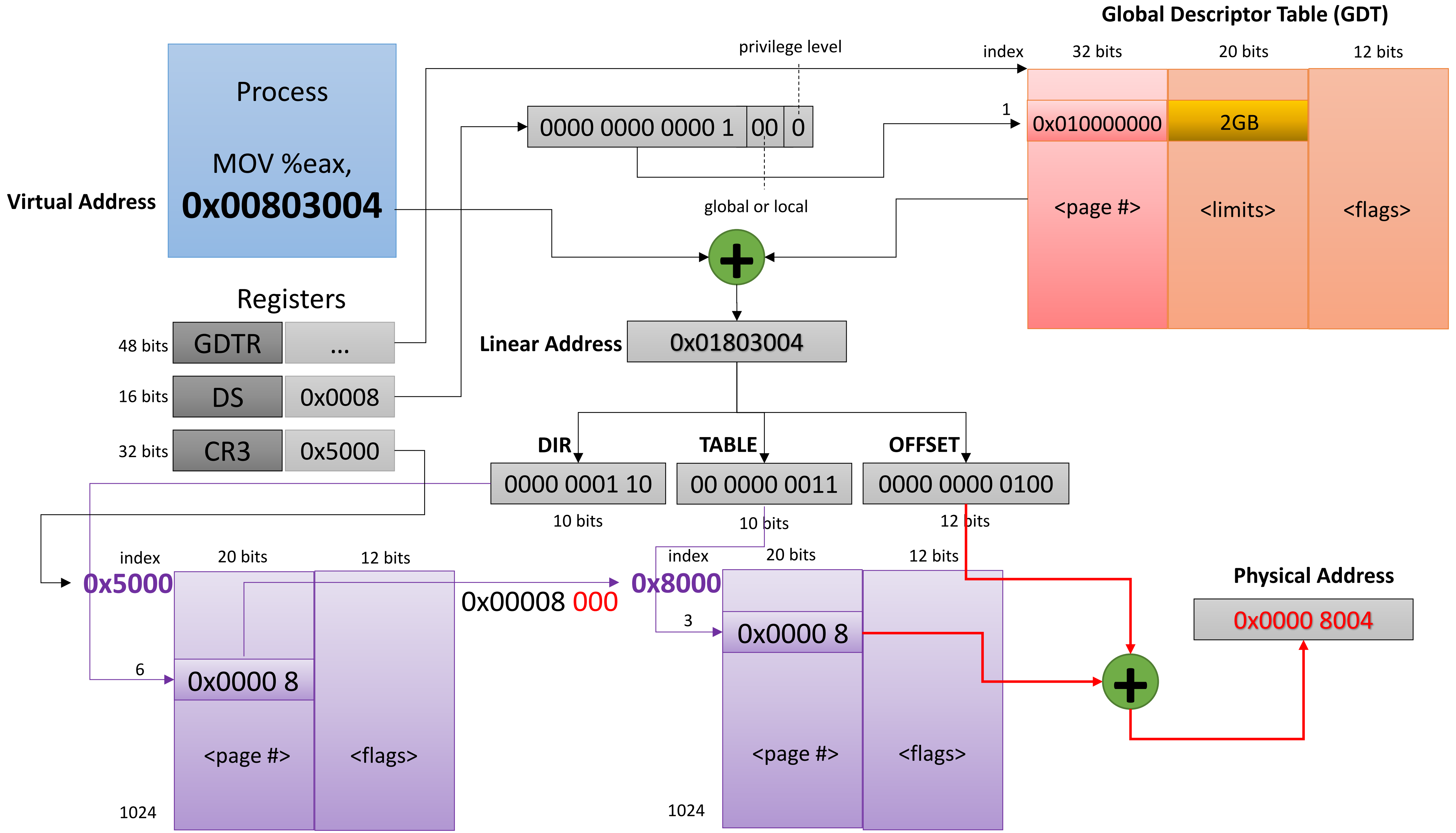
- x86, 4k page
- Logical address 0x803004 → Physical address 0x8004
- Physical address of Page Directory: 0x5000
- Physical address of the page table involved: 0x8000
- entry[1] in Global Descriptor Table: 0x1000000, 2GB
- DS register: 0x8
- **Draw the diagram of process translation**


Global Descriptor Table (GDT)
privilege level
index
32 bits
20 bits
12 bits
Process
MOV %eax,
0x00803004
Virtual Address
0000 0000 0000 1
00
0
1
0x010000000
2GB
global or local
<page #>
<limits>
<flags>
Registers
48 bits
GDTR
...
16 bits
DS
0x0008
32 bits
CR3
0x5000
Linear Address
0x01803004
DIR
TABLE
OFFSET
0000 0001 10
00 0000 0011
0000 0000 0100
10 bits
10 bits
12 bits
index
20 bits
12 bits
0x5000
6
0x0000 8
<page #>
<flags>
1024
0x00008 000
index
20 bits
12 bits
0x8000
3
<page #>
<flags>
1024
Physical Address
Page Directory
Page Table


Global Descriptor Table (GDT)
privilege level
index
32 bits
20 bits
12 bits
Process
MOV %eax,
0x00803004
Virtual Address
0000 0000 0000 1
00
0
1
0x010000000
2GB
global or local
<page #>
<limits>
<flags>
Registers
48 bits
GDTR
...
16 bits
DS
0x0008
32 bits
CR3
0x5000
Linear Address
0x01803004
DIR
TABLE
OFFSET
0000 0001 10
00 0000 0011
0000 0000 0100
10 bits
10 bits
12 bits
index
20 bits
12 bits
0x5000
6
0x0000 8
<page #>
<flags>
1024
0x00008 000
0x8000
index
20 bits
12 bits
3
<page #>
<flags>
1024
Physical Address
0x0000 8004
Page Directory
Page Table



Global Descriptor Table (GDT)
privilege level
index
32 bits
20 bits
12 bits
Process
MOV %eax,
0x00803004
Virtual Address
0000 0000 0000 1
00
0
1
0x010000000
2GB
global or local
<page #>
<limits>
<flags>
Registers
48 bits
GDTR
...
16 bits
DS
0x0008
32 bits
CR3
0x5000
Linear Address
0x01803004
DIR
TABLE
OFFSET
0000 0001 10
00 0000 0011
0000 0000 0100
10 bits
10 bits
12 bits
index
20 bits
12 bits
0x5000
6
0x0000 8
<page #>
<flags>
1024
0x00008 000
index
20 bits
12 bits
0x8000
3
0x0000 8
<page #>
<flags>
1024
Physical Address
0x0000 8004
Page Directory
Page Table



Page Directory

Page Table