# Chapter 8

## — 例题与作业

北京郵電大學

# Examples

- Paging中逻辑地址-物理地址-页表结构及地址变换
  - 例1. Fig. 8.9
  - 例2，例3，例4, 例5
- 例6. EAT in TLB-based paging
- Paging with segmentation
  - 例7，例8，例9
- 多级页表结构
  - 例10 Fig. 8.12, 例11
- 例12. 内存扩展-CPU利用率-吞吐量
- 补充作业

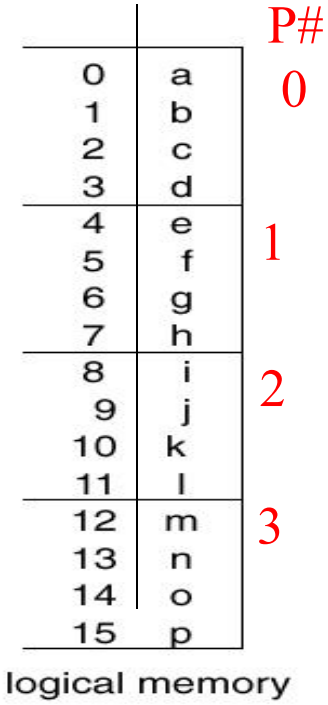
作业：

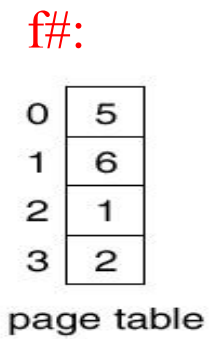
8.3, 8.9, 8.12

补充作业

# 例1. 逻辑地址-物理地址变换

- page size 4 bytes
- Logical space
  - 4 pages =16 bytes=$2^4$ B
  - logical address consists of m=**4** bits, i.e. **, **
- Physical memory
  - 8 frames =32 bytes=$2^5$B
  - physical address consists of **5** bits, i.e. ***, **




e.g. m=4, n=2

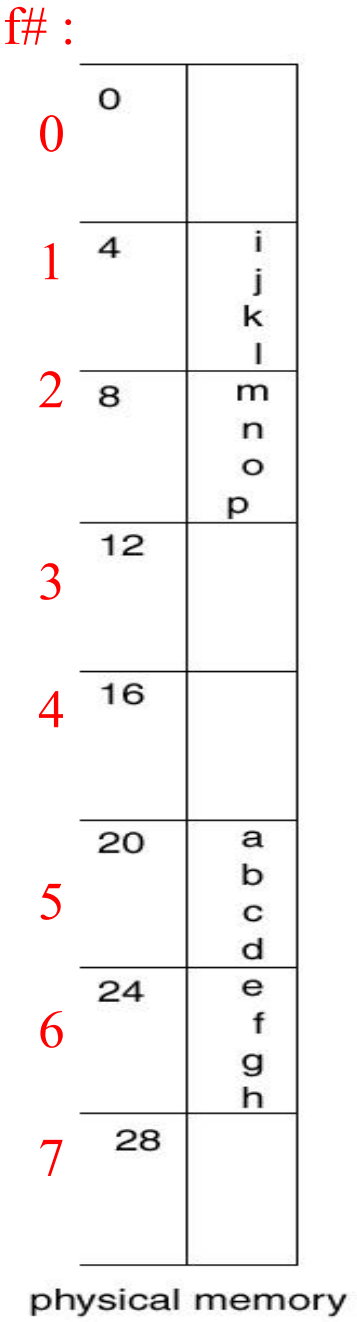



Fig.3.9

# 例1.

- Address mapping in Fig. 8.9
  - e.g.1 logical address 0, i.e. 00,00

    page number=00,  page offset=00

    page table:  page 0 is allocated to frame 5, that is, 101

    physical address : 10100, or: 5 * 4  + 0 =20

  - e.g.2  logical address 9, i.e.  10,01

    page number=2, i.e, 10,  page offset=1, i.e. 01

    page table:  page 2 is allocated to frame 1, that is, 001

    physical address :  1* 4  + 1 =5, or: 00101

# 例2

- On a system using fixed partitions with sizes $2^8$, $2^{24}$ and $2^{64}$ bytes, how many bits must the limit register have?

- Answer
  - 64 bits in the limit register

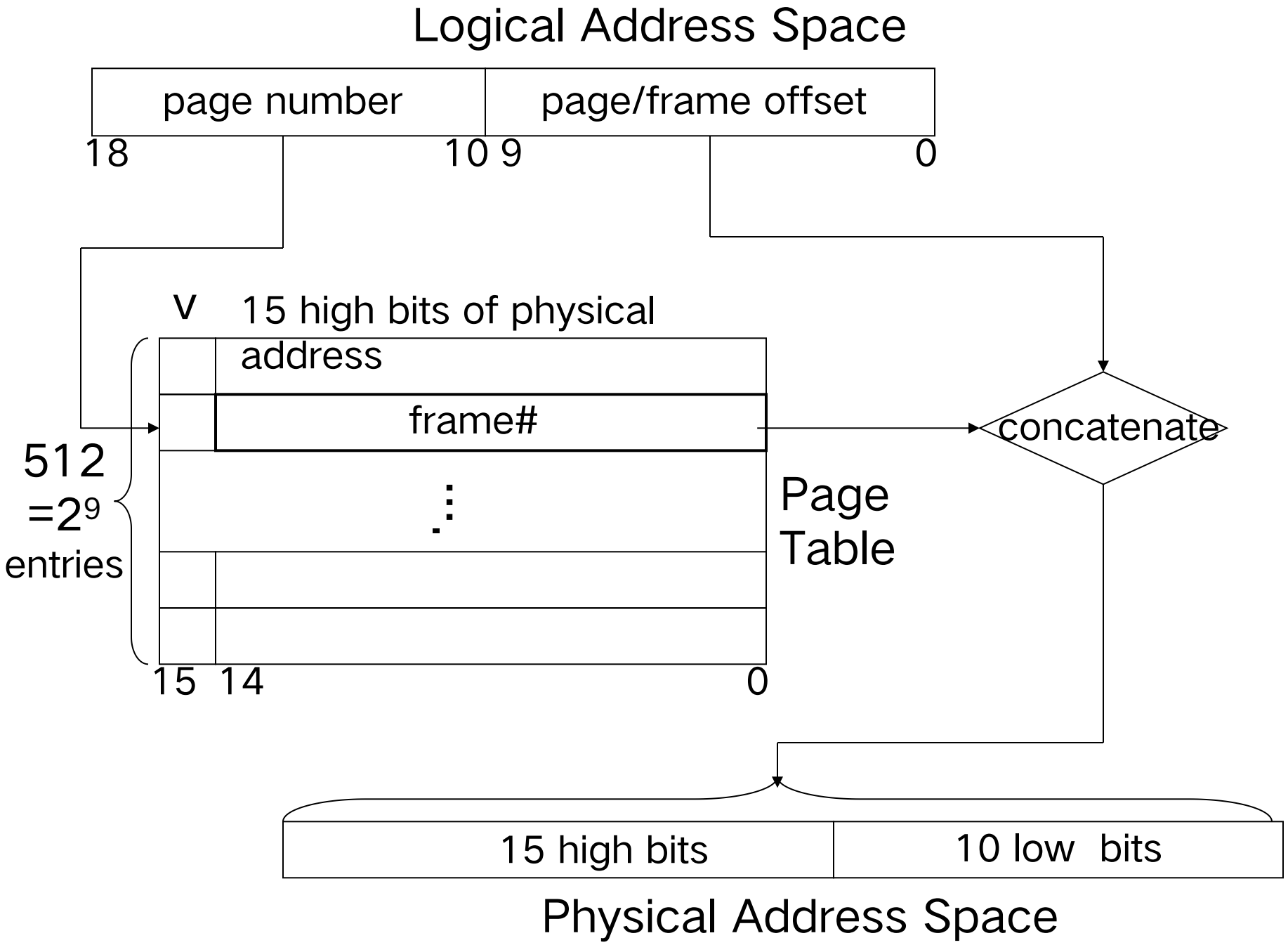
# 例3

- On a simple (not virtual memory ) paging system with $2^{24}$ bytes of physical memory, 256 pages of the logical address space, and a page size of $2^{10}$ bytes, how many bits are in a logical address?

- Answer
  - the logical address space contains 256 ($=2^8$) pages of $2^{10}$ bytes each, making the total logical address space $2^{10} \times 2^8 = 2^{18}$ bytes
  - an 18-bit address is required to cover a $2^{18}$-byte address space
  - note: the size of the logical address space is less than that of physical memory

# 例4

- On a *simple* paging system with a page table containing 512 entries of 16 bits (including one valid/invalid bit) each, and a page size of 1024 (i.e.$2^{10}$ ) bytes
  - (a) what is the size of the logical address space?
    - $2^{19}$ = the number of pages * the size of pages
      = 512*1024 (bytes)
  - (b) how many bits are there in a logical address?
    - 19 = $\log_2$(the size of logical address space)
      = $\log_2$(512 * the size of pages)=$\log_2$($512*2^{10}$) (bits)
  - (c) how many bits in the physical address specify the frame number?
    - 15= the length of entries in the page table -1
      =16 -1 (bits)


Logical Address Space
page number
page/frame offset
18
10 9
0
V
15 high bits of physical address
frame#
512
=2^9
entries
Page Table
concatenate
15
14
0
15 high bits
10 low bits
Physical Address Space

# 例4

- (d) how many bits in the physical address specify the offset within the frame?
  - 10 = $\log_2$(the size of a frame) = $\log_2$(1024)  (bits)
- (e ) what is the size of the physical address space ?
  - $2^{25}$ bytes = the number of frames * the size of frames
    = $2^{(16-1)}$ *$2^{10}$ bytes
- (f) how many bits are there in a physical address?
  - 25 = $\log_2$(the size of physical memory)
    = $\log_2$ ($2^{25}$ bytes ) bits

# 例5

- On a simple paging system with $2^{32}$ bytes of physical memory, $2^{12}$ pages of the logical address space, and page size of $512=2^9$ bytes
  - (a) how many bits are in a logical address?
    - 21 = $\log_2$(the size of logical address space)

      = $\log_2$($2^{12}$ pages * page size ) = $\log_2$($2^{12}$ * $2^9$)
  - (b) how many bytes are there in a frame?
    - 512 bytes, the same as that in a page
  - (c) how many bits in the physical address specify the frame number?
    - 23 = $\log_2$(physical memory size$2^{32}$) − $\log_2$(page size)

      = $\log_2$($2^{32}$) - $\log_2$($2^9$)

# 例5

- (d) how many entries are in the page table (how long is the page table) ?
  - $2^{12}$, the number of pages in the logical address space
- (e) how many bits in the logical address specify the page number?
  - 12 = $\log_2$(the number of pages) =$\log_2(2^{12})$
- (f) how many bits in the logical address specify the offset within a page?
  - 9 = $\log_2$(the size of pages) =$\log_2(2^9)$

# 例6. EAT in TLB-based paging

- Hit ratio α – percentage of times that a page number is found in the TLB
- E.g. It is supposed
  - the hit ratio is 80%
  - it takes 20 ns to search the TLB and 100ns to access memory, then
  - a mapped-memory access takes 120 ns when the page number is in TLB, because it takes 20ns to search TLB and 100 ns to access memory
  - if the page number is not in TLB (20 ns), the we must first access the memory for the page table and frame number (100ns), then access the desired the page in memory (100ns), for a total 220 ns
  - by weighting each case by its probability, we obtain the EAT as

$$EAT = 0.80 \times (20+100) + 0.20 \times (20+ 100 + 100)$$
$$= 140 \text{ ns}$$

# 例7

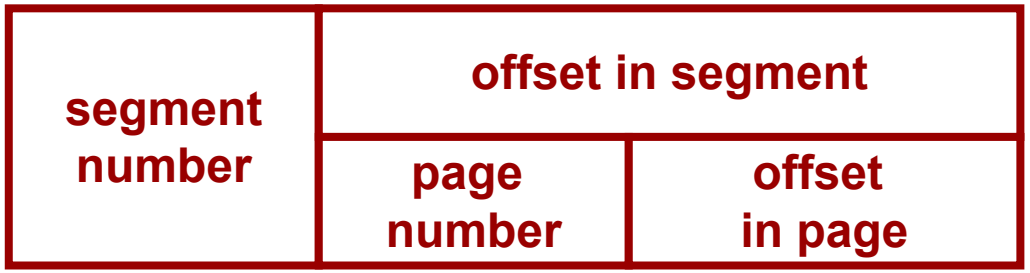
- On a system of segmentation with paging, the logical address space consists of up to 16 segments where each segment can be up to $2^{16}$ bytes long. The hardware pages each segment into 512-byte pages

  how many bits in the logical address specify the following ?

  - (a) segment number
    - 4 = $\log_2$ (the number of segments)= $\log_2(16)$
  - (b) page number
    - 7 = $\log_2$(the number of pages)

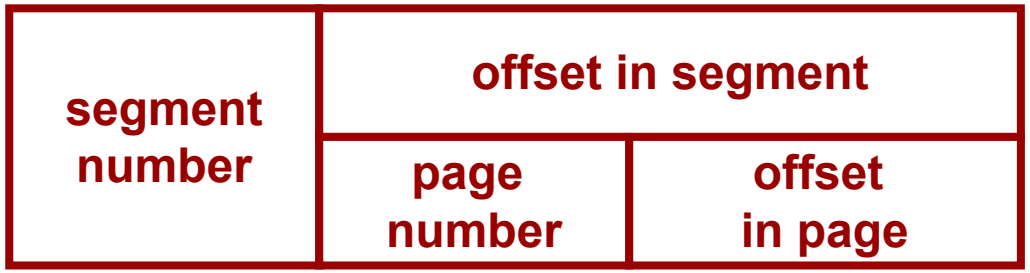
      = $\log_2$(the size of segments / the size of pages )

      = $\log_2( 2^{16} / 2^9)$



# 例7

- (c) offset within pages
  - 9 = $\log_2$(the size of pages) = $\log_2$ 512
- (d) entire logical address
  - 20 = the bits for segment number
    + the bits for page number
    + the bits for offset within pages




or: 20 = $\log_2$(the size of the logical address space)

= $\log_2$(the number of segments * the size of segments)

= $\log_2(16 * 2^{16})$

# 例8

- Consider a system using segmentation with paging management scheme, whose physical memory is of $2^{35}$ bytes. The logical address space consists of up to 8 segments. Each segment can be up to $2^{13}$ pages, and the page size of 512 bytes,
  - (1) How many bits are there in the entire logical address?
    - the entire logical address consists of segmentation and offset in segmentation, and offset in segmentation is divided into page number and page offset, so entire logical address has

$\log2\ (8) + \log2(2^{13}) + \log2\ (512) = 3 + 13 + 9 = 25$ bits

# 例8

- (2) How many bits in the logical address specify the page number?
  - each segment consists of up to $2^{13}$ pages, so $\log_2(2^{13}) = 13$ bits in the logical address specify the page number
- (3) What is the size of a frame?
  - the size of a frame is equal to that of a page, i.e. 512 bytes

- (4) How many bits in the physical address specify the frame number?
  - the total size of physical memory is $2^{35}$ bytes, and each frame is $2^9$ = 512 bytes, so there are 26 = (35 − 9) bits in the physical address specify the frame number

# 例8

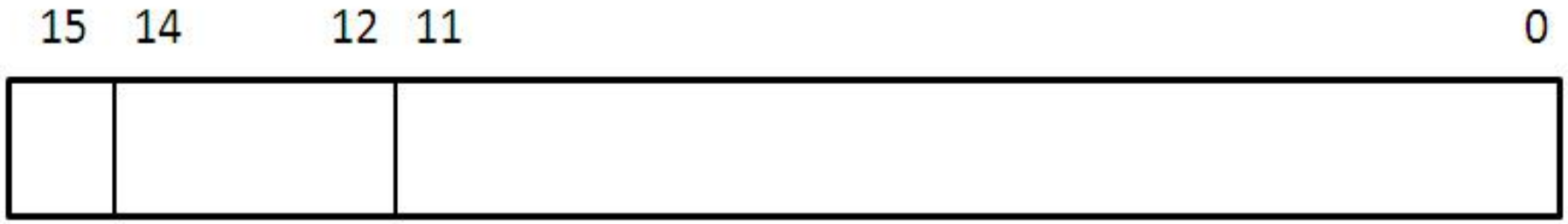
- (5) How many bits in the physical address specify the frame offset?
  - the number of bits in the physical address specifying the frame offset depends on the size of the frame, so

    $\log_2 (2^9) = 9$ bits

    in the physical address specify the frame offset
- (6) How many entries are there in the page table for each segmentation, i.e. how long is the page table for each segmentation?
  - each entry in the page table corresponds to a page in the segment, each segment can be up to $2^{13}$ pages, so there are $2^{13}$ entries in the page table for each segment

# 例9

- A system using segmentation with paging has a 16-bit logical address space with 2 segments per process and a page size of $2^{12}$ bytes
  - (1) Draw a figure to indicate the components of a logical address, and specify how many bits are there in each component




段号：1位，bit15
页号：3位，bit12~bit14
页内地址：12位，bit0~bit11

# 例9

- (2) If the content of the segment and page tables is specified below (all values binary).

Segment table

| page table | length |
|---|---|
| Pointer to page table 0 | 100 |
| Pointer to page table 1 | 101 |

Page table 0

| Frame | valid/invalid |
|---|---|
| 001100 | 1 |
| 011101 | 1 |
| 100110 | 0 |
| 001010 | 1 |
| 111010 | 1 |
| 110110 | 0 |
| 101011 | 0 |
| 001011 | 0 |

Page table 1

| Frame | valid/invalid |
|---|---|
| 010100 | 1 |
| 110101 | 0 |
| 110100 | 1 |
| 011001 | 1 |
| 001001 | 1 |
| 110011 | 0 |
| 000101 | 0 |
| 100010 | 0 |

For the following binary logical addresses, indicate what physical address they would be translated into, or if they would a page fault.

(a) 0, 001, 010001010111
(b) 0,100, 010011111111
(c) 1, 011, 010011000111
(d) 1, 110, 001011000111

(2) answers

Segment table

| page table | length |
|---|---|
| Pointer to page table 0 | 100 |
| Pointer to page table 1 | 101 |

4页

5页

Page table 0

| #Page | Frame | valid/invalid |
|---|---|---|
| 000 | 001100 | 1 |
| 001 | 011101 | 1 |
| 010 | 100110 | 0 |
| 011 | 001010 | 1 |
| 100 | 111010 | 1 |
| 101 | 110110 | 0 |
| 110 | 101011 | 0 |
| 111 | 001011 | 0 |

Page table 1

| P# | Frame | valid/invalid |
|---|---|---|
| 000 | 010100 | 1 |
| 001 | 110101 | 0 |
| 010 | 110100 | 1 |
| 011 | 011001 | 1 |
| 100 | 001001 | 1 |
| 101 | 110011 | 0 |
| 110 | 000101 | 0 |
| 111 | 100010 | 0 |

(a) 0 001 010001010111 ==> 011101, 010001010111

(b) 0 100 010011111111 ==> 111010, 0100 1111 1111

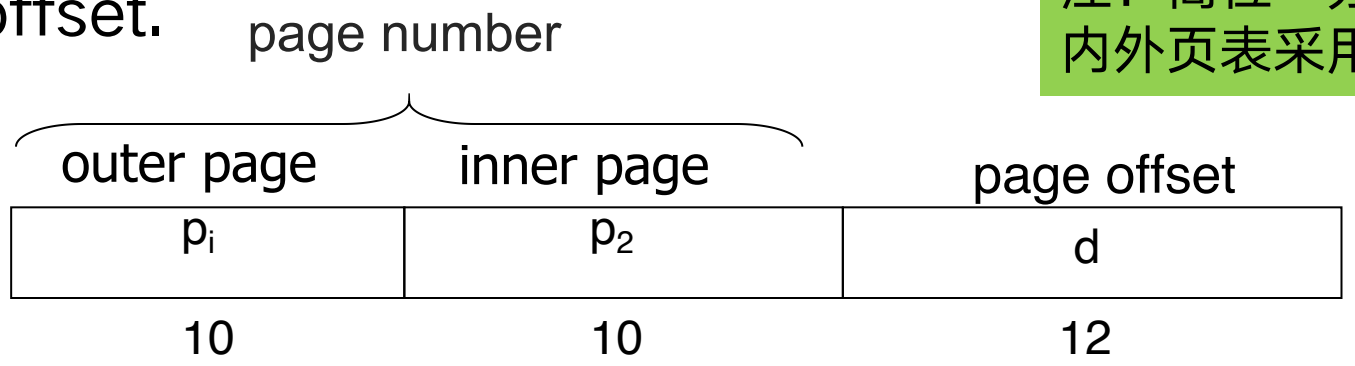
(c) 1 011 010011000111 ==> 011001, 010011000111

(d) 1 110 001011000111 ==> page fault

# 例10 Fig. 8.12

- A logical address (on 32-bit machine with 4K page size) is divided into:
  - a page number consisting of 20 bits
  - a page offset consisting of 12 bits
- The page table itself is also paged, the page number is further divided into
  - a 10-bit page number,
  - a 10-bit page offset.

注：高位一分为二，各10bits，内外页表采用相同大小的chunk




the number of entries in the table: $2^{20}=1M=2^{10}*2^{10}=1K*1K$

$1M=2^{20}$ entries in the page table is divided into $2^{10}$ chunks, and each chunk contains $2^{10}$ entries and located in a contiguous space

The page table occupies $(1+2^{10})$ chunks, $(1+2^{10}) * 2^{10} * 4 = (4K + 4M)$ Bytes

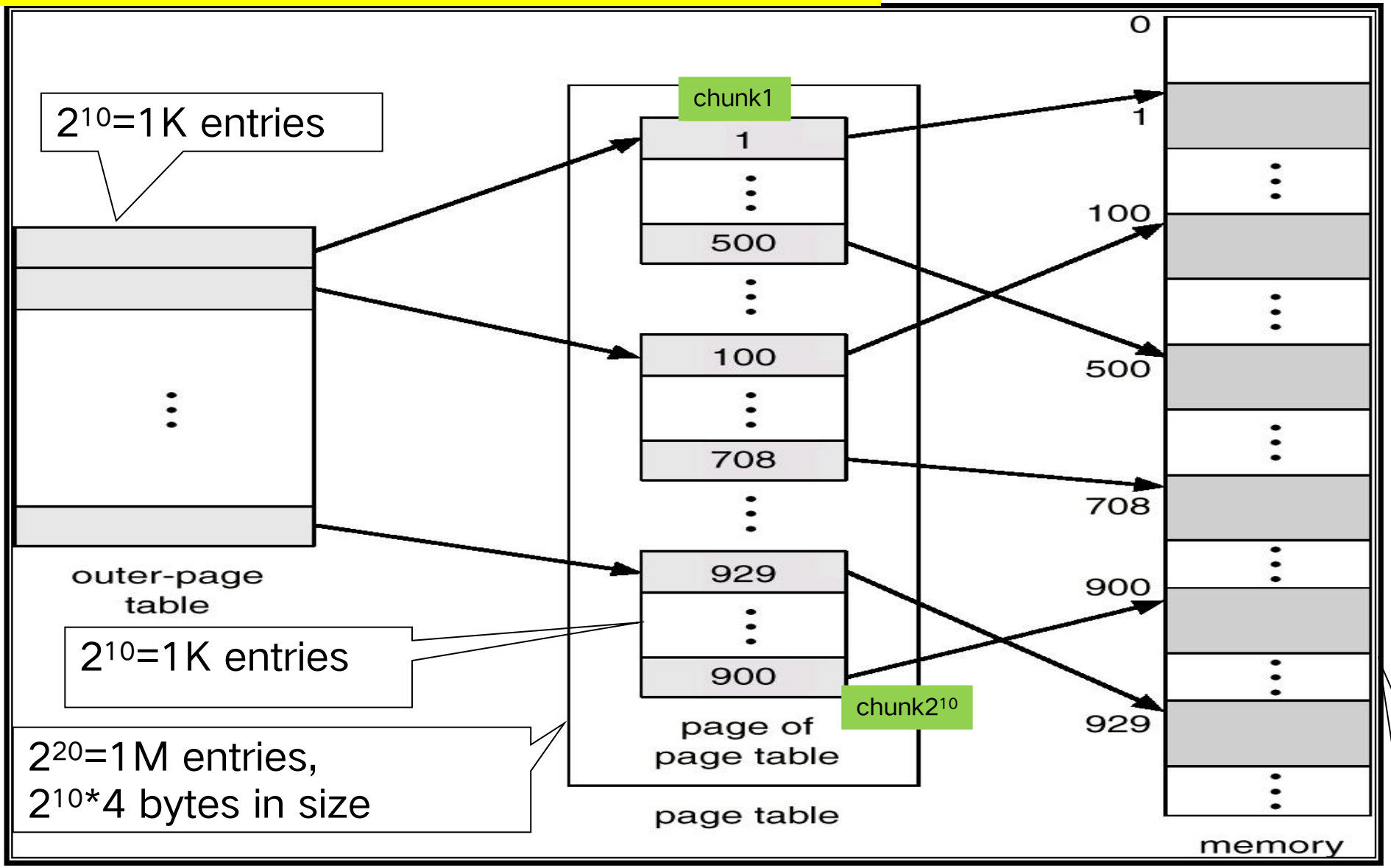



Fig.8.12. Two-level page-table scheme

$2^{20}*4K$ =4 GB

注：内外页表采用相同大小的chunk

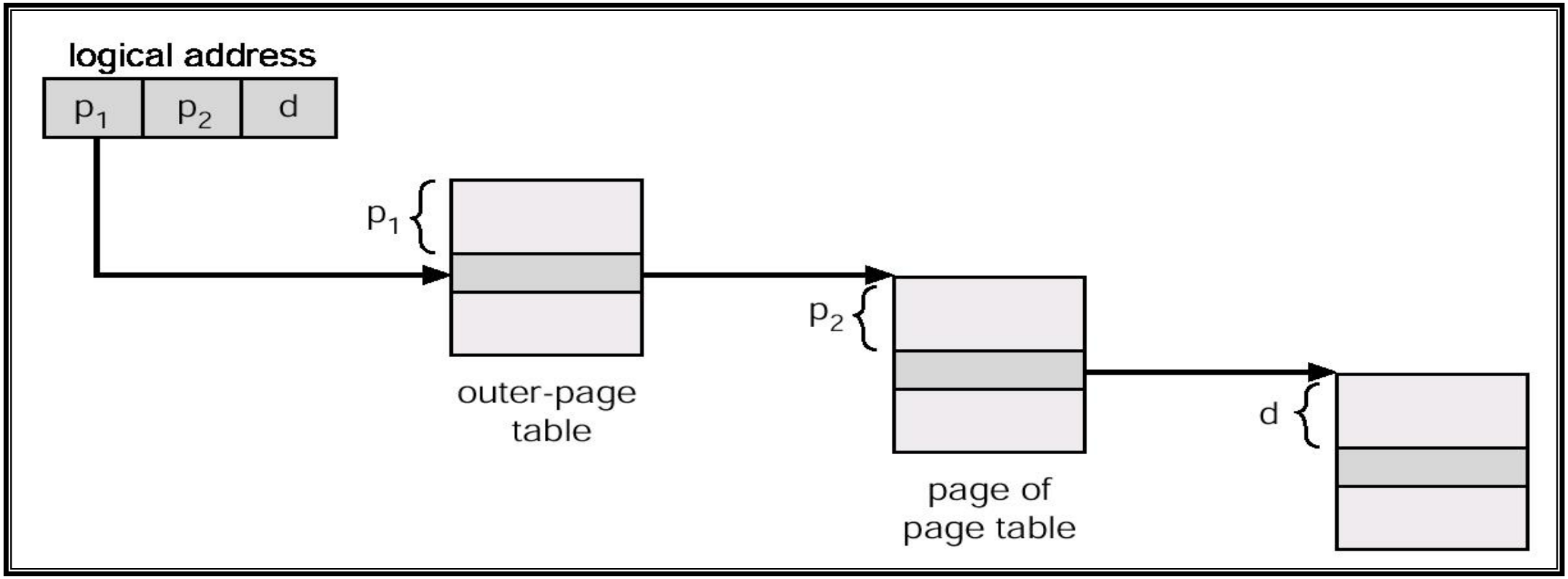
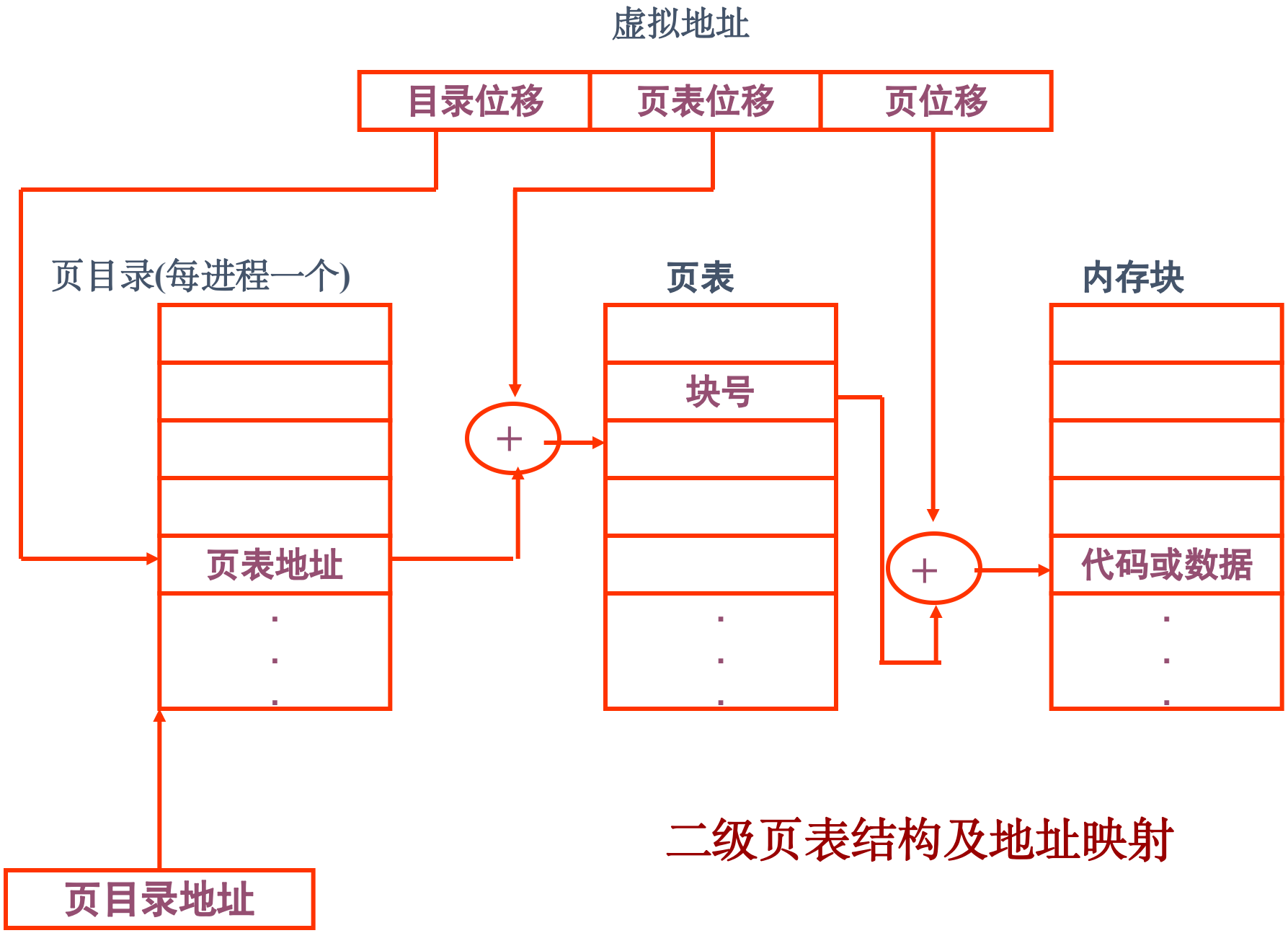



Fig.8.13 Address-translation scheme for a two-level 32-bit paging architecture




二级页表结构及地址映射

# 例11

- 某计算机采用two-level paging的分页管理机制，按字节编址，页大小为$2^{10}$bytes，页表项大小为2字节，逻辑地址结构为

| 页目编号/outer page | 页号/inner page | 页内偏移/offset |
|---|---|---|

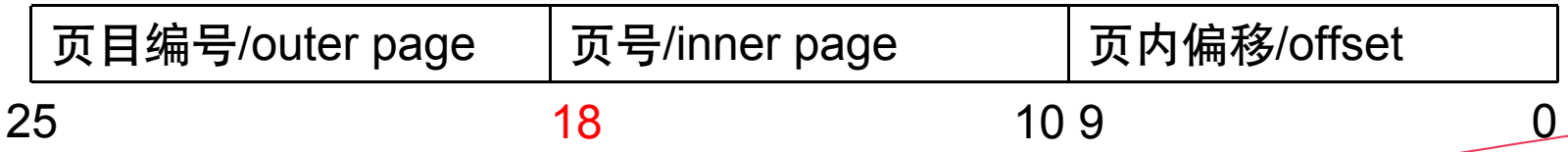
- 逻辑地址空间大小为$2^{16}$pages

- 问：表示整个逻辑地址空间的页目录表中包含表项个数至少是：B

A. 64　**B. 128**　C. 256　D. 512

# 例11

- 页大小为$2^{10}$bytes，页表项大小为2字节，
- 逻辑地址空间大小为$2^{16}$pages，共$2^{10}*2^{16}$bytes=$2^{26}$bytes, 逻辑地址26bits



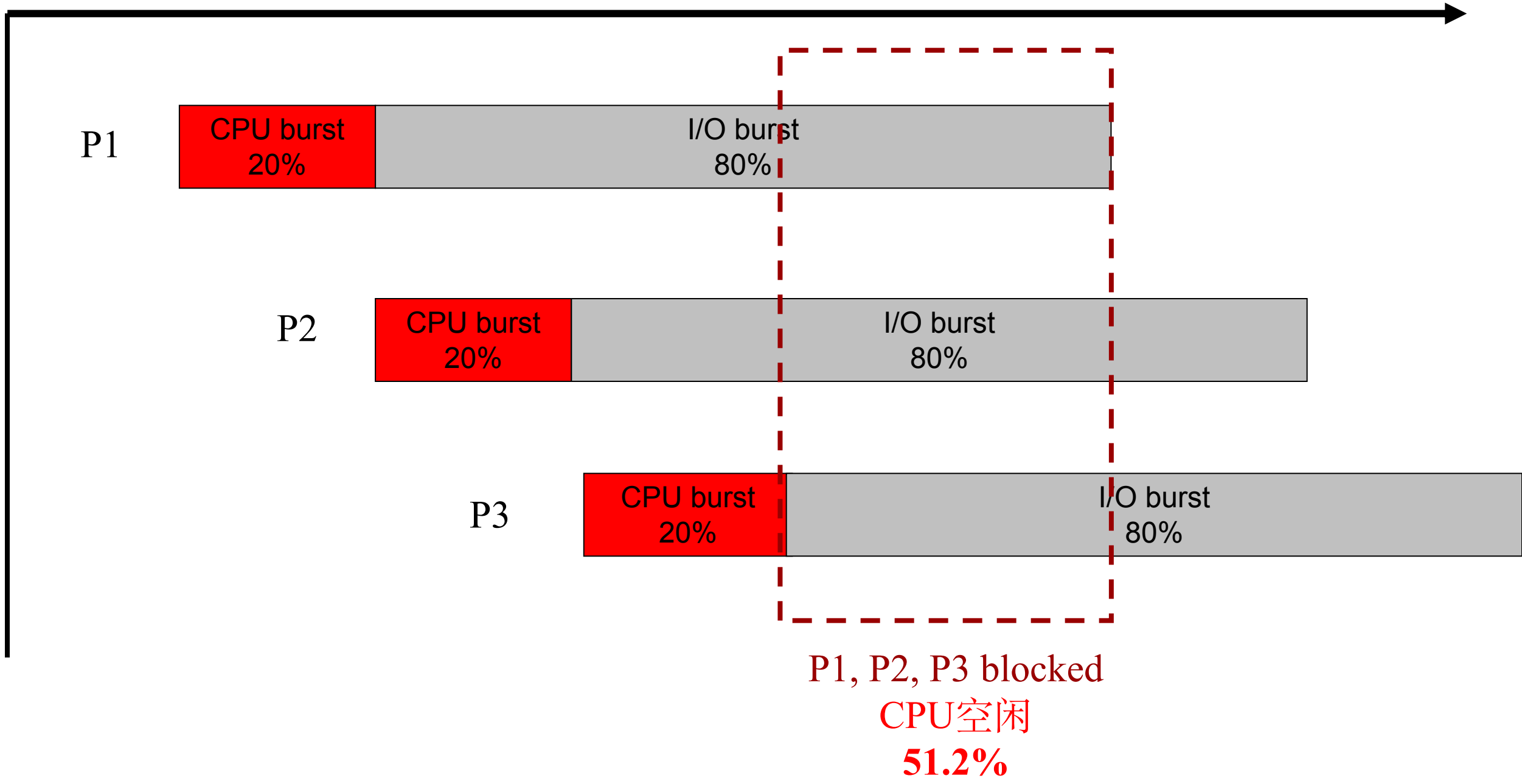

- 假设：内页表中多个chunk、外页表的1个chunk均采用1个page大小（$2^{10}$bytes ）的内存块来存储页表项，而1个页表项占2个字节，故
  - 一个chunk中可以存储$2^{10}/2=512=2^9$个页表项，所以逻辑地址中内页号所占的地址位数为9位。由此推出，外页所占位数为7位，外页有128个表项；
  - 共有$2^{16}$个pages，一个内页表chunk可以存储512个page的物理地址/frame首地址高位，需要$2^{16}/512=2^7=128$个内页表chunk，故外页表中应有128个表项指向内页中的每个chunk。

# 例12 内存扩展-CPU利用率-吞吐量

- 1个多道程序设计系统采用实存管理技术并发执行多个大小均为300K的程序。假设各个进程在执行过程中有80%的I/O等待时间（即进程等待I/O的时间占其生命周期时长的80%）
  - (1) 当系统内存为900K时，有多大比例的cpu时间被浪费掉了？cpu利用率是多少？为什么?
  - (2) 若再增加300K内存，cpu利用率可提高多少？
  - (3) 假设系统内每个进程在其生命周期内处于running态执行程序指令的总时间均相同。当系统内存由900K增至1200K时，系统**吞吐量**可提高多少

## 例12 内存扩展-CPU利用率-吞吐量

- (1) 当系统内存为900K时, 系统可容纳3个进程同时并发执行
  - 根据题意，进程在执行过程中有80%的I/O等待时间, 表明进程等待I/O的时间占其生命周期时长的80%, 进程处于waiting状态的概率为0.8
  - 对系统内并发执行的n个进程, 当这n个进程同时处于I/O等待时, cpu是空闲的
  - 系统内3个并发进程同时处于I/O等待态而导致CPU空闲的概率为$(80\%)^3=0.512$
  - 因此，系统有51.2%的CPU时间被浪费掉了，系统CPU的利用率为(1—51.2%)＝48.8%。


P1
CPU burst
20%
I/O burst
80%
P2
CPU burst
20%
I/O burst
80%
P3
CPU burst
20%
I/O burst
80%
P1, P2, P3 blocked
CPU空闲
51.2%

## 例12 内存扩展-CPU利用率-吞吐量

- (2) 若再增加300K内存，系统可容纳 4 个进程同时并发执行
  - 此时，CPU的利用率为 $(1— (80\%)^4) = (1— 0.4096) = 0.5904$

所以，若再增加300K内存，cpu利用率可提高：

(0.5904—0.488) / 0.488 = 20.98%

## 例12 内存扩展-CPU利用率-吞吐量

- (3)根据题意，设系统内每个进程在其生命周期内处于running态执行程序指令(CPU burst)的总时间均为T
  - 当系统内存为900K时, 系统可容纳3个进程并发执行。如果3个进程全部并发执行完所需时间为$L_3$，则：

    CPU的利用率=3T / $L_3$=48.8%,

    $L_3$ = 3T / (48.8%),

    ，系统吞吐量$Th_3$=3/ $L_3$= (48.8%) / T

## 例12 内存扩展-CPU利用率-吞吐量

- 当系统内存由900K增至1200K时，系统可容纳 4 个进程同时并发执行。设4个进程全部并发执行完所需时间为$L_4$，则:

  CPU的利用率 = 4T / $L_4$=59.04%,

  $L_4$=4T / (59.04%),

  系统吞吐量$Th_4$=4/ $L_4$= (59.04%) / T

- 因此，若再增加300K内存，系统吞吐量可提高:

  [(59.04%) / T —(48.8%) / T] / [(48.8%) / T]

  = (0.5904—0.488) / 0.488 = 20.98%

  = 20.98%

CPU利用率提高→系统吞吐量提高

# 补充作业

- On a *simple* paging system with a page table containing 1024 entries of 13 bits (including one valid/invalid bit) each, and a page size of 4096 bytes
  - (a) what is the size of the logical address space?
  - (b) how many bits are there in a logical address?
  - (c) how many bits in the physical address specify the frame number?
  - (d) how many bits in the physical address specify the offset within the frame?
  - (e ) what is the size of the physical address space ?
  - (f) how many bits are there in a physical address?

要求：参照例4，画图