# 《操作系统》课下作业 (OS-HW6)

中国人民大学　信息学院　崔冠宇　2018202147

**P421, 8.1** Suppose the page table for the process currently executing on the processor looks like the following. All numbers are decimal, everything is numbered starting from zero, and all addresses are memory byte addresses. The page size is 2,048 bytes.

| Virtual page number | Valid bit | Reference bit | Modify bit | Page frame number |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 7 |
| 1 | 0 | 0 | 0 | - |
| 2 | 0 | 0 | 0 | - |
| 3 | 1 | 0 | 0 | 6 |
| 4 | 1 | 1 | 0 | 0 |
| 5 | 1 | 0 | 1 | 3 |

**a.** Describe exactly how, in general, a virtual address generated by the CPU is translated into a physical main memory address.

**b.** What physical address, if any, would each of the following virtual addresses correspond to? (Do not try to handle any page faults, if any.)

(i) 6,204　(ii) 3,021　(iii) 9,000

**解.**

**a.** 可参考教材图 8.2:

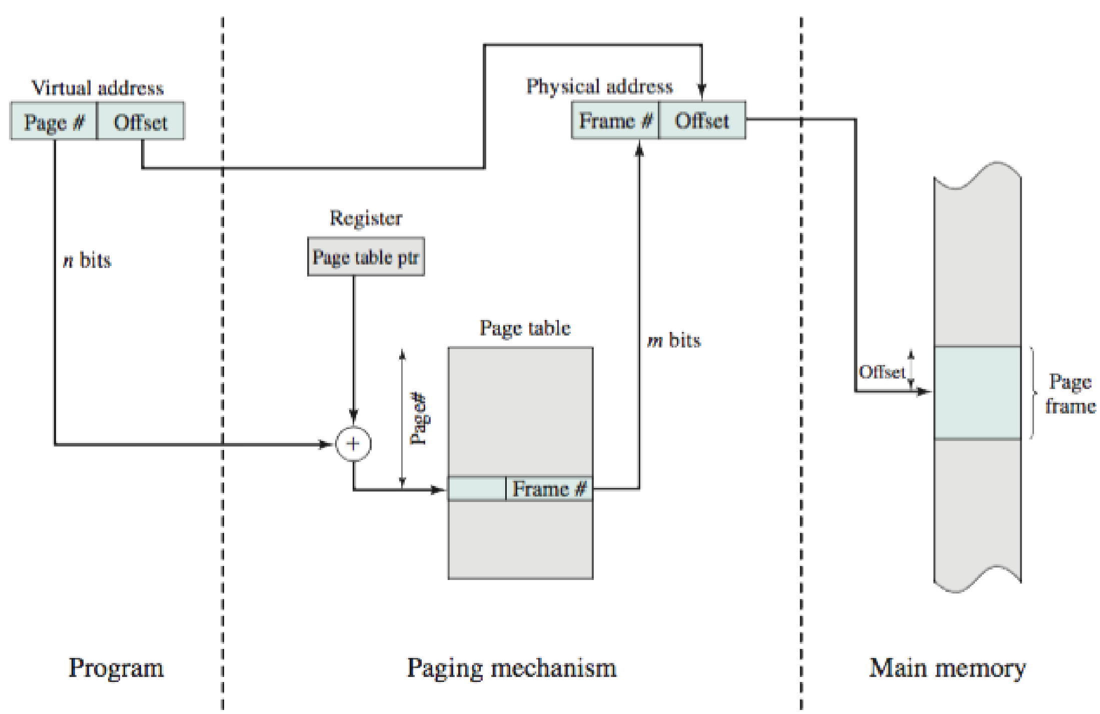


**Figure 8.2** Address Translation in a Paging System

根据页的大小, 将虚地址分为两部分: 页号和页内偏移量 (此处页大小为 2048 Bytes, 故低 11 位为页内偏移量). 将页表寄存器内的页表始址加上页号, 得到页表中对应的页表项, 然后取出页框号, 再与页面偏移量做拼接, 就得到了物理地址.

**b.** (i) $6204 = 3 \times 2048 + 60$, 查表得虚页号 3 对应的页框号为 6, 所以物理地址为 $6 \times 2048 + 60 = 12348$.
(ii) $3021 = 1 \times 2048 + 973$, 查表得虚页号 1 有效位为 0, 出现页错误.
(iii) $9000 = 4 \times 2048 + 808$, 查表得虚页号 4 对应的页框号为 0, 所以物理地址为 $0 \times 2048 + 808 = 808$.

**P422, 8.4** Consider the following page-reference string: a, b, d, c, b, e, d, b, d, b, a, c, b, c, a, c, f, a, f, d. Assume that there are 3 frames available and that they are all initially empty. Complete a figure, similar to Figure 8.14, showing the frame allocation for each of the following page replacement policies:

**a.** First-in-first-out

**b.** Optimal

**c.** Least recently used

Then, find the relative performance of each policy with respect to page faults.

**解.** 以下 F 表示需要置换的缺页中断, (F) 表示不需置换的缺页中断.

**a.** First-in-first-out: 先进先出, 即置换上一状态时进程页框中最先被调入的页.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| a | a | a | c | c | c | c | c | d | d |
| | b | b | b | b | e | e | e | e | e |
| | | d | d | d | d | d | b | b | b |
| (F) | (F) | (F) | F | | F | | F | F | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| d | d | b | b | b | b | b | b | b | d |
| a | a | a | a | a | a | f | f | f | f |
| b | c | c | c | c | c | c | a | a | a |
| F | F | F | | | | F | F | | F |

**b.** Optimal: 选择置换下次访问距当前时间最长的页.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| a | a | a | c | c | e | e | e | e | e |
| | b | b | b | b | b | b | b | b | b |
| | | d | d | d | d | d | d | d | d |
| (F) | (F) | (F) | F | | F | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| a | a | a | a | a | a | a | a | a | d |
| b | b | b | b | b | b | f | f | f | f |
| d | c | c | c | c | c | c | c | c | c |
| F | F | | | | | F | | | F |

**c.** Least recently used: 置换内存中最长时间未被引用的页.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| a | a | a | c | c | c | d | d | d | d |
| | b | b | b | b | b | b | b | b | b |
| | | d | d | d | e | e | e | e | e |
| (F) | (F) | (F) | F | | F | F | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| d | c | c | c | c | c | c | c | c | d |
| b | b | b | b | b | b | f | f | f | f |
| a | a | a | a | a | a | a | a | a | a |
| F | F | | | | | F | | | F |

从缺页错误的角度看, FIFO 方式在这种情况下产生了 10 + 3 次缺页, 最多; OPT 方式在这种情况下产生了 6 + 3 次缺页, 最少; LRU 方式在这种情况下产生了 7 + 3 次缺页, 总体较好.

**P423, 8.10** Suppose the virtual space accessed by memory is 6 GB, the page size is 8 KB, and each page table entry is 6 bytes. Compute the number of virtual pages that is implied. Also, compute the space required for the whole page table.

**解.** 虚空间大小为 6 GB = $1.5 \times 2^{32}$ Byte, 页大小为 8 KB = $2^{13}$ Byte. 故虚页数量为 6 GB / 8 (KB/页) = $1.5 \times 2^{19}$ 页; 页表大小为 $1.5 \times 2^{19}$ 页 $\times 6$ (Byte/页) = $1.125 \times 2^{22}$ Byte = 4.5 MB.

**P424, 8.17** Assume a task is divided into four equal-sized segments, and the system builds an eight-entry page descriptor table for each segment. Thus, the system has a combination of segmentation and paging. Assume also the page size is 2 kB.

**a.** What is the maximum size of each segment?

**b.** What is the maximum logical address space for the task?

**c.** Assume an element in physical location 00021ABC is accessed by this task. What is the format of the logical address that the task generates for it? What is the maximum physical address space for the system?

**解.**

**a.** 每段最大尺寸 = $8 \times 2$ KB = 16 KB. (= $2^{14}$ Byte, 即段内偏移占据 14 位)

**b.** 任务最大逻辑地址空间大小 = $4 \times 16$ KB = 64 KB. (= $2^{16}$ Byte, 即占据 16 位)

**c.** 段数目为 4, 占两位; 一个段内有 8 个页, 占 3 位; 每页 2KB, 占 11 位. 根据段页式系统的地址特点, 结合上面的分析, 逻辑地址的格式为:

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| 段号 | | 页号 | | | 页内偏移 | | | | | | | | | | |

由于物理地址为 00021ABC, 是 8 位 16 进制数, 相当于 32 位 2 进制数, 所以物理地址空间为 $2^{32}$ Byte = 4 GB.