KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN MẠNG MÁY TÍNH VÀ TT DỮ LIỆU

Hướng dẫn Lab 10.1 – Giả lập bộ nhớ ảo.

Cấp phát bộ nhớ là một trong những thành phần cốt lõi của một Hệ điều hành. Trong phương pháp cấp phát liên tục, một tiến trình sẽ được nạp vào một lỗ trống với một giải thuật thích hợp, Hệ điều hành cập nhật danh sách các lỗ trống để quản lý.

Mục tiêu	Lý thuyết liên quan	Tài nguyên
Phân trang	Ch9.3.1 Basic Method (Paging) Ch9.3.2.1 Translation Look-Aside Buffer. Ch9.5 Swapping Ch10.4.2 FIFO Page Replacement Ch10.4.4 LRU Page Replacement Ch10.4.5.2 Second-Chance Algorithm	https://github.com/Trantin84/LAB_IntroOS (mã nguồn ví dụ). Sử dụng image Ubuntu 16 / 18
TLB		
Page Replacement		

Yêu cầu sinh viên: Hiểu và thực thi các đoạn mã đã cung cấp. Áp dụng lý thuyết hiện thực bộ nhớ ảo với chức năng dịch địa chỉ, tra cứu TLB, thay thế trang bằng các giải thuật, swapping bộ nhớ ảo.

Đánh giá sinh viên: Hỏi đáp các vấn đề lý thuyết. Kỹ năng thực hành. Bài tập.

Yêu cầu nộp bài: các tập tin mã nguồn .c và tập tin khả thực thi .out của các Yêu cầu trong buổi thực hành và Bài tập cuối hướng dẫn trong thời gian cho phép của giảng viên.

Preferences

[1] Abraham Silberschatz, Peter B. Galvin, Greg Gagne, [2018], Operating System Concepts, 10th edition, John Wiley & Sons, New Jersey.

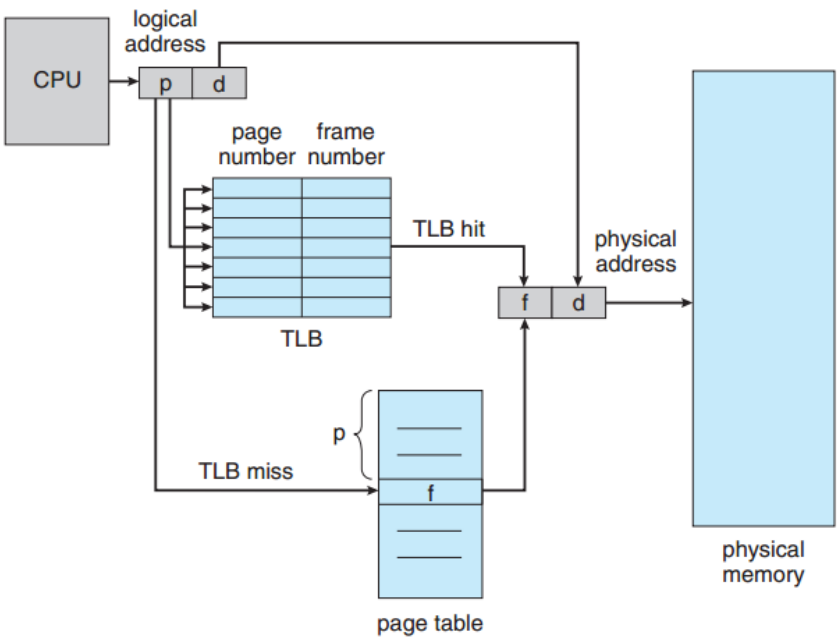
Programming Problems of Chapter 9, 10.

Yêu cầu 1: Cài đặt bộ nhớ ảo với tham số như sau: (***Programming Projects: Designing a Virtual Memory Manager, sách [1] trang P-51 (p.566 PDF)***)

Vùng nhớ ảo có kích thước 2^{16} = 65,536 bytes. Địa chỉ ảo 16 bits chia làm hai phần: phần số trang dài 8-bit và phần offset cũng dài 8-bit. Vì thế:

- Có 2^8 dòng trong Bảng trang.
- Một trang có kích thước 2^8 = 256 bytes.
- TLB có 16 chỉ mục.
- Một khung có kích thước 2^8 bytes.
- Bộ nhớ vật lý có 256 khung.
- Kích thước bộ nhớ vật lý là 256 khung × 256 bytes / khung = 65,536 bytes.
- Vùng nhớ ảo được hiện thực bởi tập tin BACKINGSTORE.bin nằm trên đĩa cứng.

Hướng dẫn: Tải về tập tin ***LAB_IntroOS/LAB_10/task1_1_FIFO_page.c***, xem xét các cấu trúc dữ liệu.



Hình 1. Dịch địa chỉ với phần cứng TLB.

- Dòng 16 đến 18 là cài đặt theo mô tả.

- Dòng 19 là số khung cấp cho tiến trình (có thể từ 1 đến 256).

- Dòng 20 thể hiện bộ nhớ chính. Mỗi vị trí lưu trữ 1 byte (đại diện là 1 kí tự).

- Dòng 22 là cấu trúc 1 chỉ mục trong Bảng trang, mỗi chỉ mục bao gồm chỉ số khung, bit khả dụng (được cài đặt khi trang nạp vào bộ nhớ và được xoá khi trang bị Swap Out) và bit dirty (dùng để xác định có cần Swap Out hay không). Dòng 33 khai báo Bảng trang gồm số lượng chỉ mục mong muốn.

- Dòng 27 là cấu trúc 1 chỉ mục trong TLB. Do một quan hệ (p, f) có thể lưu vào chỉ mục bất kỳ nên cần thêm phần tử ghi số khung f (nhiều hơn so với chỉ mục của Bảng trang). Dòng 34 khai báo TLB có số lượng chỉ mục mong muốn.

```
16      #define PAGE_TABLE_SIZE 256
17      #define OFFSET 256
18      #define TLB_SIZE 16
19      int FRAME_SIZE = 100;
20      char MEM[PAGE_TABLE_SIZE * OFFSET];

22      struct PageEntry {
          bool bVaild;
          int iFrame;
          bool bDirty;   };

27      struct TLBEntry {
          int iPage;
          int iFrame;
          bool bVaild;
          bool bDirty;    };

33      struct PageEntry PageTable[PAGE_TABLE_SIZE];
34      struct TLBEntry TLB[TLB_SIZE];
```

Yêu cầu 2: Dịch địa chỉ ảo sang địa chỉ vật lý, có sử dụng TLB.

Hướng dẫn: Tải về tập tin ***LAB_IntroOS/LAB_10/task1_1_FIFO_page.c***, biên dịch và thực thi theo giải thích bên dưới.

(tham khảo *Hình 2. Dịch đỉa chỉ với phần cứng TLB*.)

- Dòng 70 tìm số trang p sau khi bỏ đi phần d (trong bộ p | d của địa chỉ ảo), việc bỏ đi phần d được tiến hành bằng cách shift phải 8 bits, (hay là chia cho 256).

- Dòng 71 tìm số offset d từ 8 bits cuối trong bộ p | d, có được từ mặt nạ 0b0000 0000 1111 1111 hoặc 0x00FF, (hay là chia lấy dư cho 256).

- Dòng 73 là hành vi tra cứu trong TLB. TLB thiết kế nằm trong phần cứng CPU và số chỉ mục cũng rất ít (ở đây chỉ có 16 dòng) nên hành vi tra cứu sẽ rất nhanh. Nếu tìm thấy p trong một chỉ mục nào đó của TLB, giá trị khung f được tìm thấy (iFrame) và địa chỉ vật lý f |d được tính tại dòng 76.

- Nếu không tìm thấy trong TLB, hệ thống buộc phải tra cứu trong Bảng trang (dòng 81), nằm trong bộ nhớ chính và số chỉ mục có thể lên đến hàng ngàn dòng (trong hệ thống này là 256 dòng). Hành vi tra cứu này rất chậm so với TLB.

- Trong trường hợp trang chưa được nạp vào bộ nhớ, cả bảng TLB lẫn Bảng trang đều sẽ không tìm thấy, việc xử lý lỗi trang sẽ thảo luận trong yêu cầu tiếp theo.

```
70      int iPage = iVirtualAdd >> 8;
71      int iOffset = iVirtualAdd & 0x00FF;

73      int iFrame = fTLBLook(iPage);
74      if (iFrame != -1) {
75        printf(" TLB hit  ");
76        int PhysicAdd = iFrame * OFFSET + iOffset;
77        printf(" ... ", PhysicAdd, MEM[PhysicAdd]);

80        printf(" TLB_miss ");
81        iFrame = fPageTable(iPage);
82        if (iFrame != -1) { }

87      //Page fault
```

Hướng dẫn thực thi.

- Nếu đối số thứ hai là số nguyên: chuỗi truy xuất bộ nhớ được sinh ra ngẫu nhiên với số lượng truy vấn là giá trị đối số.

- Nếu đối số thứ hai là tên tập tin: chuỗi truy xuất được đọc từ tập tin. Tập tin bao gồm chuỗi các số nguyên, cách nhau bởi khoảng trắng.

```
$ gcc -o memory.out task1_1_FIFO_page.c
$ ./memory.out 500 48
$ ./memory.out data.txt 65
```

*** Kiểm chứng độc lập**: tập tin BACKINGSTORE.bin đã được nạp các kí tự từ a đến z và lặp lại. Vì vậy một địa chỉ ảo @ sẽ truy cập kí tự tương ứng như sau:

@ MOD 26 = 0 → kí tự 'a'.

@ MOD 26 = 1 → kí tự 'b'.

…

@ MOD 26 = 25 → kí tự 'z'.

Yêu cầu 3: Tìm nạn nhân bằng giải thuật FIFO (giải thuật Clock).

Hướng dẫn:

* Thuật toán FIFO cũng được áp dụng cho TLB, nhưng cần lưu ý rằng giải thuật cho TLB và cho Khung bộ nhớ không nhất thiết phải giống nhau.

- Với thuật toán FIFO, các vị trí thay thế sẽ "xoay vòng" (lý do tên gọi Clock được dùng). Dòng 36 và 37 khai báo một giá trị khởi đầu là 0 và sẽ tăng 1 sau mỗi lần thay thế xảy ra (tại dòng 131 và 132), khi vị trí thay thế đến tận cùng của Khung bộ nhớ (hay TLB) thì quay về 0 (phép tính % được sử dụng).

- Khi đã xác định nạn nhân, trang đó không còn khả dụng, bit Khả dụng sẽ bị xoá (Dòng 115 và 117).

- Nếu nạn nhân có thao tác ghi (tức là dữ liệu có thay đổi), trang đó cần ghi lại ra vùng nhớ ảo. Việc này được xác định bằng bit Dirty (hay bit Modified) (dòng 119). Nếu không có gì thay đổi, tức là nội dung vẫn như bên ngoài Vùng nhớ ảo, việc ghi ra sẽ không cần thiết và rất mất thời gian I/O với HDD.

- Sau đó trang cần thiết được Swap In (dòng 121).

- Dòng 122 – 125 là cập nhật Bảng trang. Trong khi dòng 127 – 129 là cập nhật TLB.

36	`int CLOCK_TLB = 0;`
37	`int CLOCK_PTB = 0;`

```
112     printf(" Victim=%d ", CLOCK_PTB);
114     for (int i = 0; i < PAGE_TABLE_SIZE; i++)
115       if (PageTable[i].iFrame == CLOCK_PTB)
                PageTable[i].bVaild = false;
116     for (int i = 0; i < TLB_SIZE; i++)
117       if (TLB[i].iFrame == CLOCK_PTB)
                TLB[i].bVaild = false;

119     if (PageTable[iPage].bDirty == true) fSwapOut(iPage);
121     fSwapIn(iPage, CLOCK_PTB * OFFSET);

122     PageTable[iPage].iFrame = CLOCK_PTB;
123     PageTable[iPage].bVaild = true;
125     PageTable[iPage].bDirty = false;

127     TLB[CLOCK_TLB].bVaild = true;
128     TLB[CLOCK_TLB].iFrame = CLOCK_PTB;
129     TLB[CLOCK_TLB].iPage = iPage;

131     CLOCK_PTB = (CLOCK_PTB + 1) % FRAME_SIZE;
132     CLOCK_TLB = (CLOCK_TLB + 1) % TLB_SIZE;
```

Yêu cầu 4: Xử lý thao tác Swap In.

Hướng dẫn: Mặc dù địa chỉ ảo chúng ta cần tìm chỉ dẫn đến 1 byte, nhưng thao tác của hệ thống tính trên 1 trang (ở đây là 256 bytes), cho việc Swap Out hay Swap In được diễn ra với khối 256 bytes (hằng số OFFSET), khối này khai báo ở dòng 99.

- Dòng 97 có tham số iPage là trang cần đọc, ví trí bắt đầu đọc chính là (iPage * OFFSET), vì vậy con trỏ đọc di chuyển khoảng cách này (tính từ đầu tập tin) (Dòng 104).

- Dòng 98 – 100 tạo con trỏ quản lý tập tin (tượng trưng cho Vùng nhớ ảo).

- Dòng 97 có tham số iMemPosition là vị trí bắt đầu ghi vào Bộ nhớ chính, chính là Số khung nạn nhân * OFFSET.

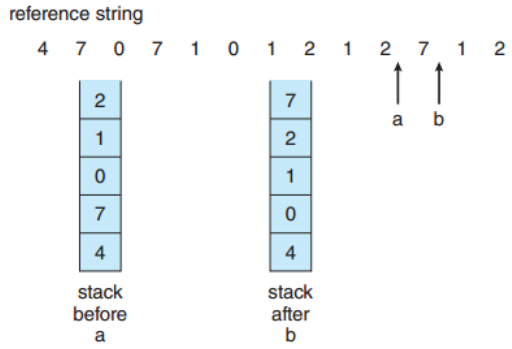
- Dòng 105 đọc một trang. và dòng 106 ghi nó vào Bộ nhớ chính.

```
97      void fSwapIn(int iPage, int iMemPosition) {
98        FILE * fp;
99        char str[OFFSET];
100       if ((fp = fopen("BACKINGSTORE.bin", "r")) == NULL) {

104       fseek(fp, iPage * OFFSET, SEEK_SET);
105       fgets(str, OFFSET, fp);
106       for (int i = 0; i < OFFSET; i++)
                    MEM[iMemPosition + i] = str[i];
```

Bài tập

1. Xây dựng hàm tìm nạn nhân và thay thế trang bằng thuật toán **LRU**.
   Gợi ý: Thuật toán này còn tên gọi là Stack. Một Stack có kích thước bằng số Khung và ghi lại các Trang được tham khảo theo quy tắc: Trang mới nạp vào được ghi lên đỉnh Stack và trang Tham khảo lại được nổi lên đỉnh Stack; Khi cần tìm Nạn nhân, chỉ số dưới đáy Stack chính là nạn nhân cần tìm.



Hình 2. Stack quản lý trang tham khảo gần đây.

2. Xây dựng hàm tìm nạn nhân và thay thế trang bằng thuật toán **Second Chance.**
   Gợi ý: Thuật toán dựa trên FIFO nhưng cần hiện thực thêm một cột bit "Reference", mỗi Trang mới nạp hay Được tham khảo lại thì bit này được cài đặt. Mỗi khi con trỏ CLK_PTB tìm đến, nếu Trang có bit tham khảo, bit tham khảo bị xoá, con trỏ di chuyển tiếp. Nhưng nếu Trang không có bit tham khảo, Trang đó là nạn nhân.

3. Hiện thực hàm tiến hành **Swap Out** trang ra bộ nhớ ảo.
   Gợi ý: Ngược lại với Swap In, hàm Swap Out sẽ đọc Bộ nhớ chính MEM[] một đoạn dài bằng OFFSET (1 trang) và ghi vào tập tin BACKINGSTORE.bin