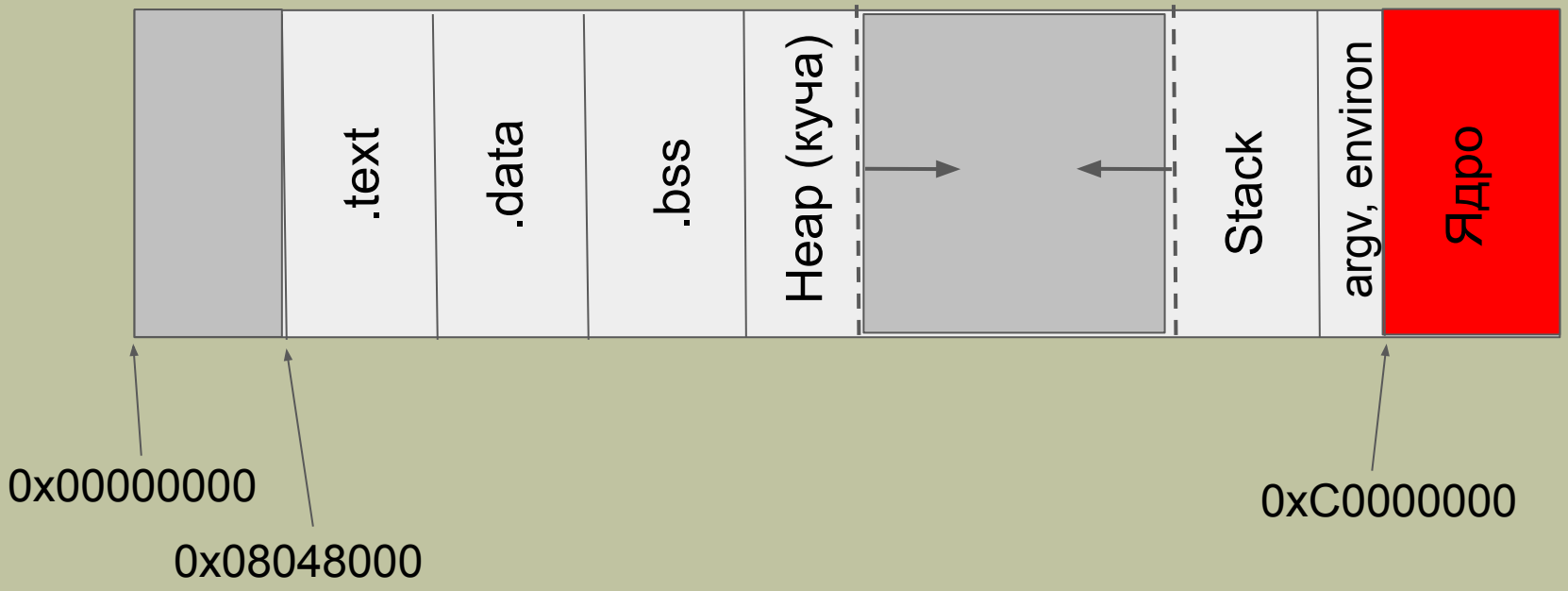
Лекция 7

- Образ программы в памяти.
- Механизм виртуальной памяти.

Распределение памяти процесса в Linux/x86-32.



```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

char Buffer[1<<25]={1};
int odds[] = {1, 3, 5, 7 };

int f(int x){
 static char buffer[1<<25]={1};
 return x*x*x;
}
```

.data

.data

.data

```
Lecture7> ls -ltrh
-rwxr-xr-x 1 malkov users
73M Oct 18 17:06 lab7
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

char Buffer[1<<25]={0};
int odds[] = {1, 3, 5, 7 };

int f(int x){
 static char buffer[1<<25];
 return x*x*x;
}
```

.bss

.data

.bss

```
Lecture7> ls -ltrh
-rwxr-xr-x 1 malkov users
13K Oct 18 17:11 lab7
```

```
int main(int argc, char *argv[]){
 int n = 3;
 static char mbuf[1<<23]={1};
 char *p;
 p = malloc(1<<21);

 printf("f(%d)=%d\n", n, f(n));
 pause();

 free(p);
 return 0;
}
```

.data

Heap

```
int main(int argc, char *argv[]){
 int n = 3;
 static char mbuf[1<<23];
 char *p;
 p = malloc(1<<21);

 printf("f(%d)=%d\n", n, f(n));
 pause();

 free(p);
 return 0;
}
```

.bss

Heap

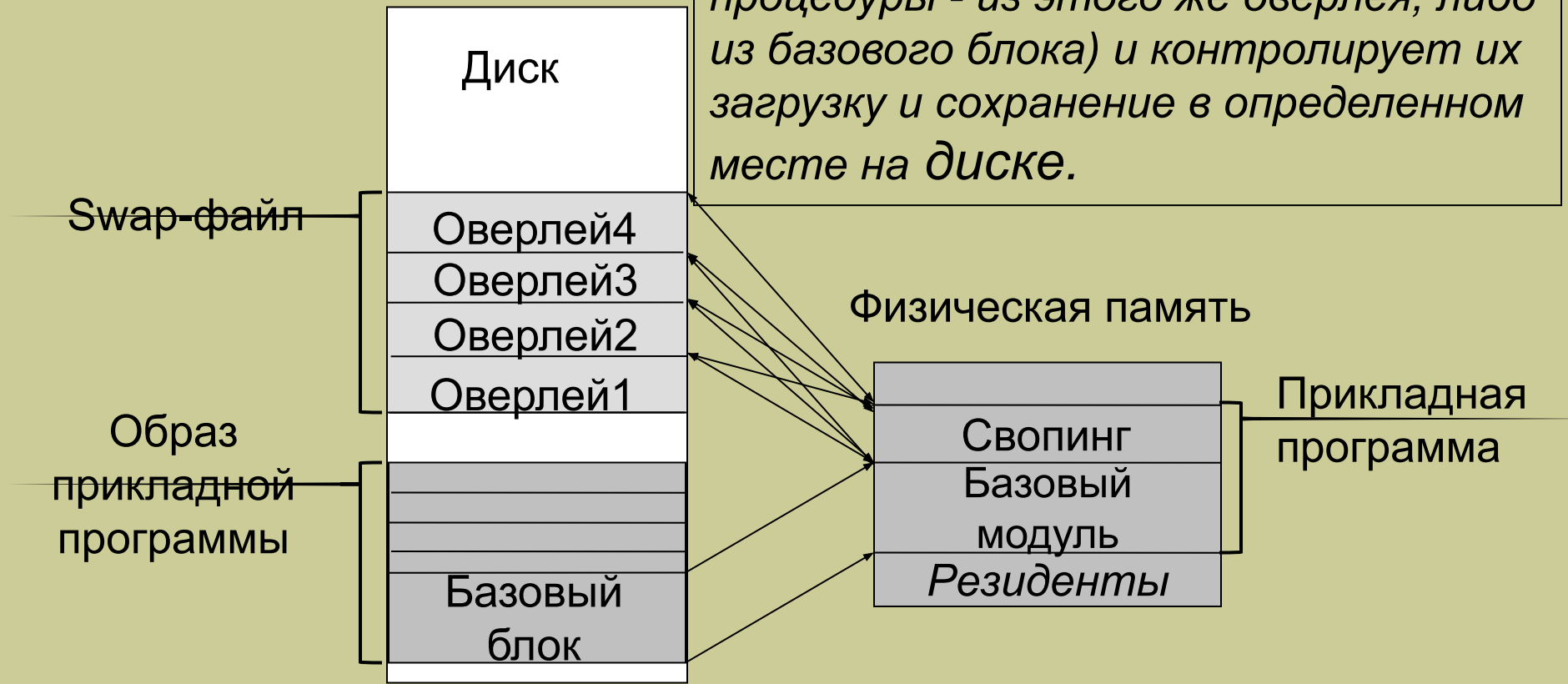
```
/Lecture7> cat /proc/12565/maps
00400000-00401000 r-xp 00000000 08:13 1096819760     ~/lab7
00600000-00601000 r--p 00000000 08:13 1096819760     ~/lab7
00601000-00602000 rw-p 00001000 08:13 1096819760     ~/lab7
00602000-04e02000 rw-p 00000000 00:00 0
05d20000-05d41000 rw-p 00000000 00:00 0              [heap]
7f0fb29e4000-7f0fb2be5000 rw-p 00000000 00:00 0
7f0fb2be5000-7f0fb2d96000 r-xp 00000000 00:2d 18863 /lib64/libc-2.26.so
……
7f0fb31c6000-7f0fb31c7000 rw-p 00026000 00:2d 18855 /lib64/ld-2.26.so
7f0fb31c7000-7f0fb31c8000 rw-p 00000000 00:00 0
7ffe1c5f7000-7ffe1c619000 rw-p 00000000 00:00 0       [stack]
……..
```

```
/Lab7> cat /proc/12868/maps
00400000-00401000 r-xp 00000000 08:13 1074618870                         ~/Lab7/lab5
00600000-00601000 r--p 00000000 08:13 1074618870                         ~/Lab7/lab5
00601000-00602000 rw-p 00001000 08:13 1074618870                         ~/Lab7/lab5
00f38000-00f59000 rw-p 00000000 00:00 0                                  [heap]
7f9f6ba9a000-7f9f6bc4b000 r-xp 00000000 00:2d 18863                      /lib64/libc-2.26.so
……….
7f9f6be55000-7f9f6be56000 r-xp 00000000 08:13 1074618869                 ~/liblab5.so
7f9f6be56000-7f9f6c055000 ---p 00001000 08:13 1074618869                 ~/liblab5.so
7f9f6c055000-7f9f6c056000 r--p 00000000 08:13 1074618869                 ~/liblab5.so
……
7f9f6c057000-7f9f6c07c000 r-xp 00000000 00:2d 18855
/lib64/ld-2.26.so
……
7fff8249b000-7fff824bd000 rw-p 00000000 00:00 0                          [stack]
7fff825eb000-7fff825ef000 r--p 00000000 00:00 0                          [vvar]
7fff825ef000-7fff825f1000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 --xp 00000000 00:00 0                  [vsyscall]
```

Организация памяти

Оверлейная модель

Программист разбивает программу на оверлеи (например, вызов процедуры - из этого же оверлея, либо из базового блока) и контролирует их загрузку и сохранение в определенном месте на диске.



Виртуальная память.

[Виртуальное] адресное пространство – некоторая последовательность чисел. Код программы может ссылаться на адреса этого числового диапазона.

Существует некоторая схема отображения виртуальных адресов на адреса физической памяти.

Технология страничной организации памяти

Пример: машинный код позволяет адресовать 64K байт памяти, физическая память составляет 4K.

Поделим адресное пространство на 16 областей (страниц) по 4K и установим следующее соответствие:

физический адрес = виртуальный адрес % 4K;

номер области (страницы) = виртуальный адрес / 4K.

При ссылке по виртуальному адресу A
- содержимое физической памяти сохраняется на диске;
- область с номером A/4K загружается в память;
- произойдет обращение по адресу физической памяти A%4K.

Современные реализации страничной организации памяти

Каждому процессу выделяется адресное пространство (например в 32 разрядной Windows числа от нуля до 0xFFFFFFFF).

Адресное пространство разбивается на страницы размером, обычно (в зависимости от ОС) от 512 байт до 64K.

Физическая память разбивается на области (*страничные кадры* (фреймы, блоки, слоты)) размером в страницу.

Таблица страниц устанавливает соответствие между страницами и страничными кадрами.

Виртуальная страница	Страничный кадр	Бит присутствия
…	…	…
7	0	0
6	3	1
5	4	1
4	0	0
3	2	1
2	0	0
1	0	1
0	1	1

Физическая память	**Кадр**
Виртуальная страница 5	4
Виртуальная страница 6	3
Виртуальная страница 3	2
Виртуальная страница 0	1
Виртуальная страница 1	0

Отображением виртуальной памяти на физические адреса занимается диспетчер виртуальной памяти –*VMM (Virtual Memory Management*).

Аппаратной реализацией *VMM* является *MMU (Memory Management Unit),* расположенный на чипе процессора.

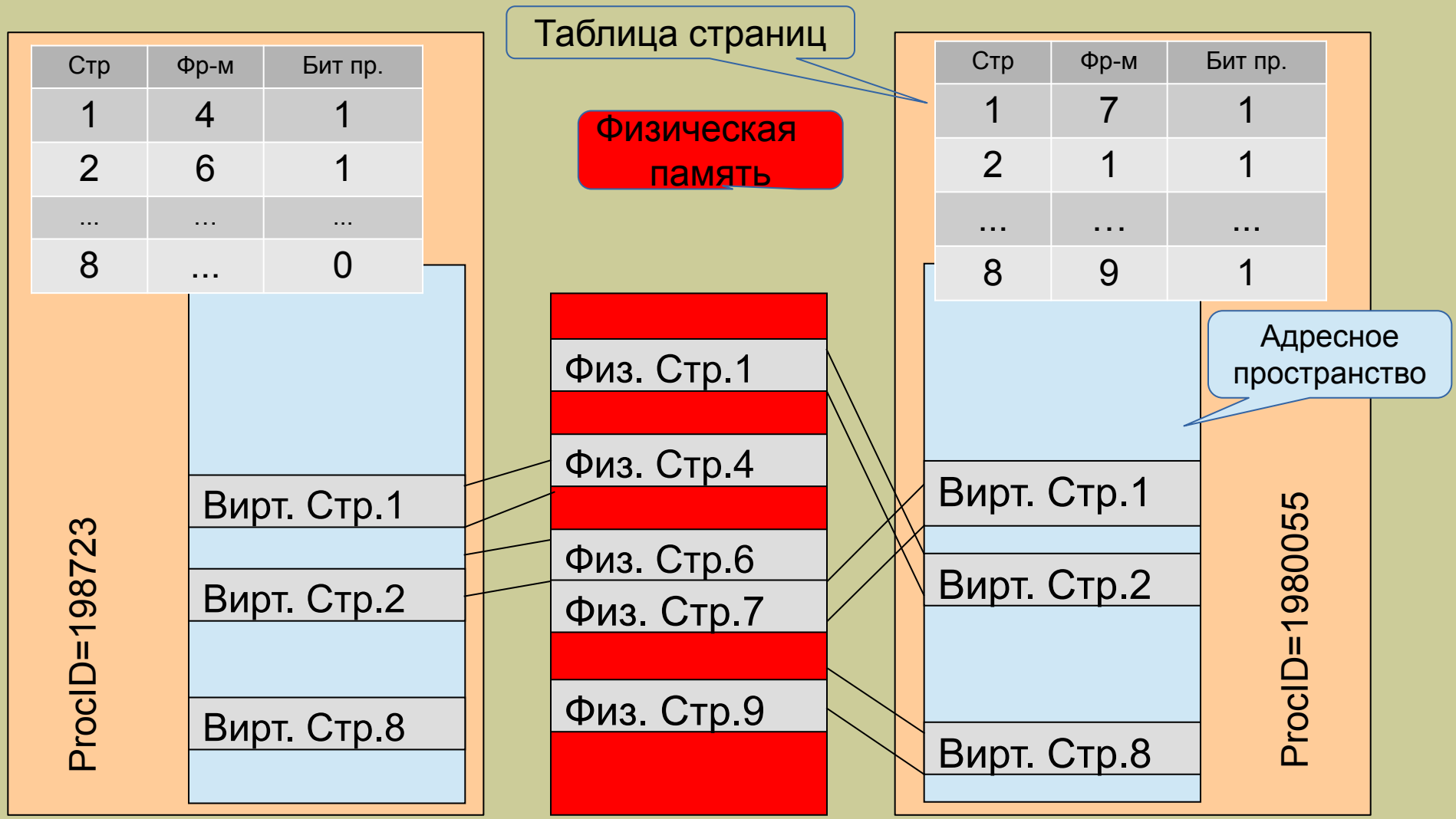
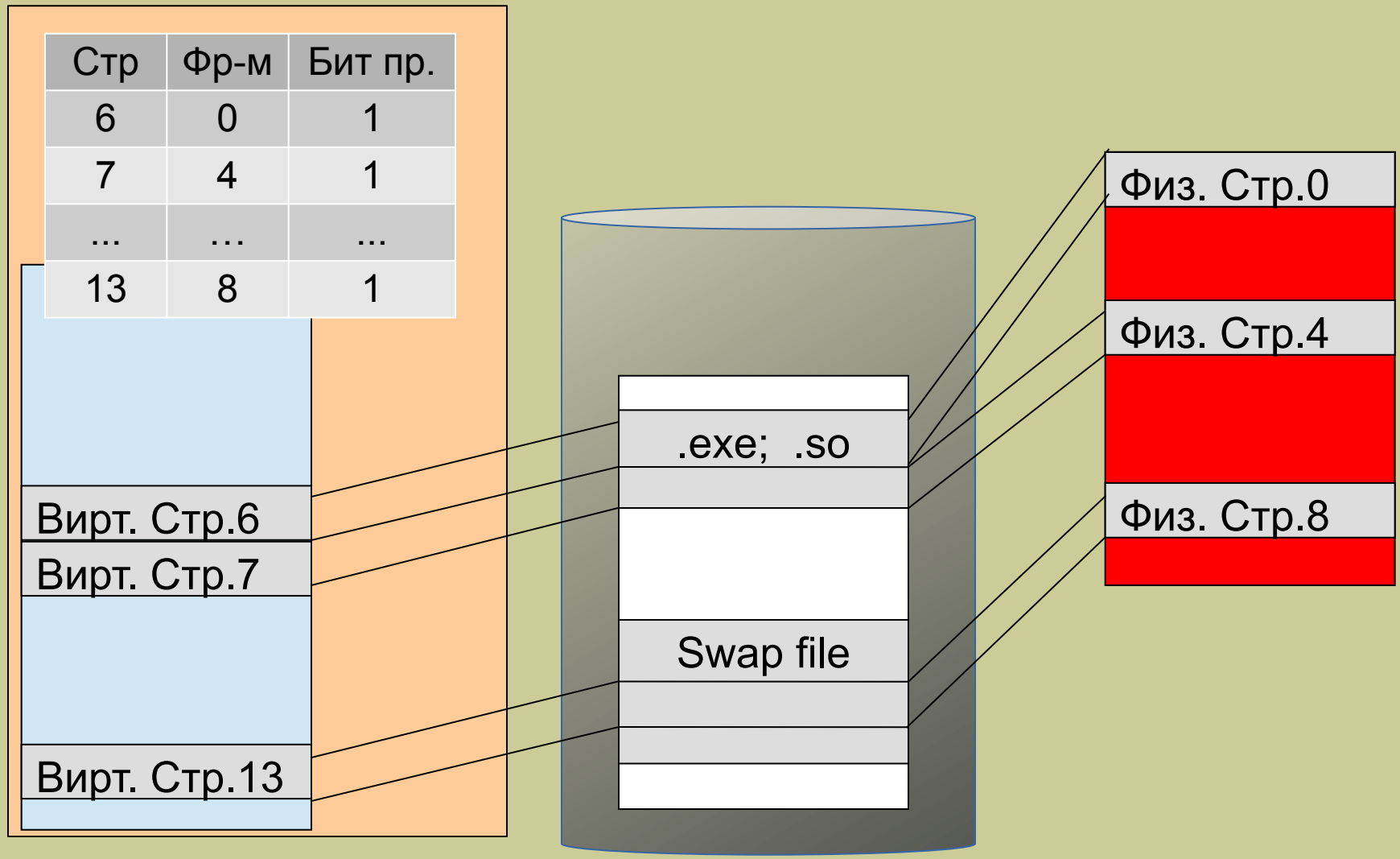


Таблица страниц
Физическая память
Стр
Фр-м
Бит пр.
1
4
1
2
6
1
...
...
...
8
...
0
Вирт. Стр.1
Вирт. Стр.2
Вирт. Стр.8
ProcID=198723
Физ. Стр.1
Физ. Стр.4
Физ. Стр.6
Физ. Стр.7
Физ. Стр.9
Стр
Фр-м
Бит пр.
1
7
1
2
1
1
...
…
...
8
9
1
Адресное пространство
Вирт. Стр.1
Вирт. Стр.2
Вирт. Стр.8
ProcID=1980055

Вызов страниц по требованию. При обращении к адресу страницы, которой нет в основной памяти (бит присутствия 0), генерируется исключение – ***ошибка отсутствия страницы*** (промах). Обработка этого исключения – считывается нужная страница с диска, в таблице страниц делается соответствующая запись и команда повторяется.

Политика замещения страниц. Существует множество алгоритмов удаления (как правило, с последующим сохранением на диске) страниц из физической памяти. Например: *LRU (Least Recently Used)* – удаляется дольше всего не использовавшаяся страница; FIFO *(First –in First out)* – алгоритм очереди.


Стр
Фр-м
Бит пр.
6
0
1
7
4
1
...
…
...
13
8
1
Вирт. Стр.6
Вирт. Стр.7
Вирт. Стр.13
.exe; .so
Swap file
Физ. Стр.0
Физ. Стр.4
Физ. Стр.8