Solution of Exercise Sheet 5

Exercise 1 (Memory Management)

1. Mark memory management methods that cause internal fragmentation to occur.

   ☒ Static partitioning
   ☐ Dynamic partitioning
   ☒ Buddy memory allocation

2. Mark memory management methods that cause external fragmentation to occur.

   ☐ Static partitioning
   ☒ Dynamic partitioning
   ☒ Buddy memory allocation

3. Explain how external fragmentation can be fixed.

   By defragmentation. For virtual memory, external fragmentation is irrelevant.

4. Mark the memory management method that searches for the block, which fits best.

   ☐ First Fit ☐ Next Fit ☒ Best fit ☐ Random

5. Mark the memory management concept that searches for a free block, starting from the beginning of the address space.

   ☒ First Fit ☐ Next Fit ☐ Best fit ☐ Random

6. Mark the memory management concept that fragments quickly the large area of free space at the end of the address space.

   ☐ First Fit ☒ Next Fit ☐ Best fit ☐ Random

7. Mark the memory management concept that selects randomly a free and appropriate block.

   ☐ First Fit ☐ Next Fit ☐ Best fit ☒ Random

8. Mark the memory management concept that searches for a free block, starting from the latest allocation.

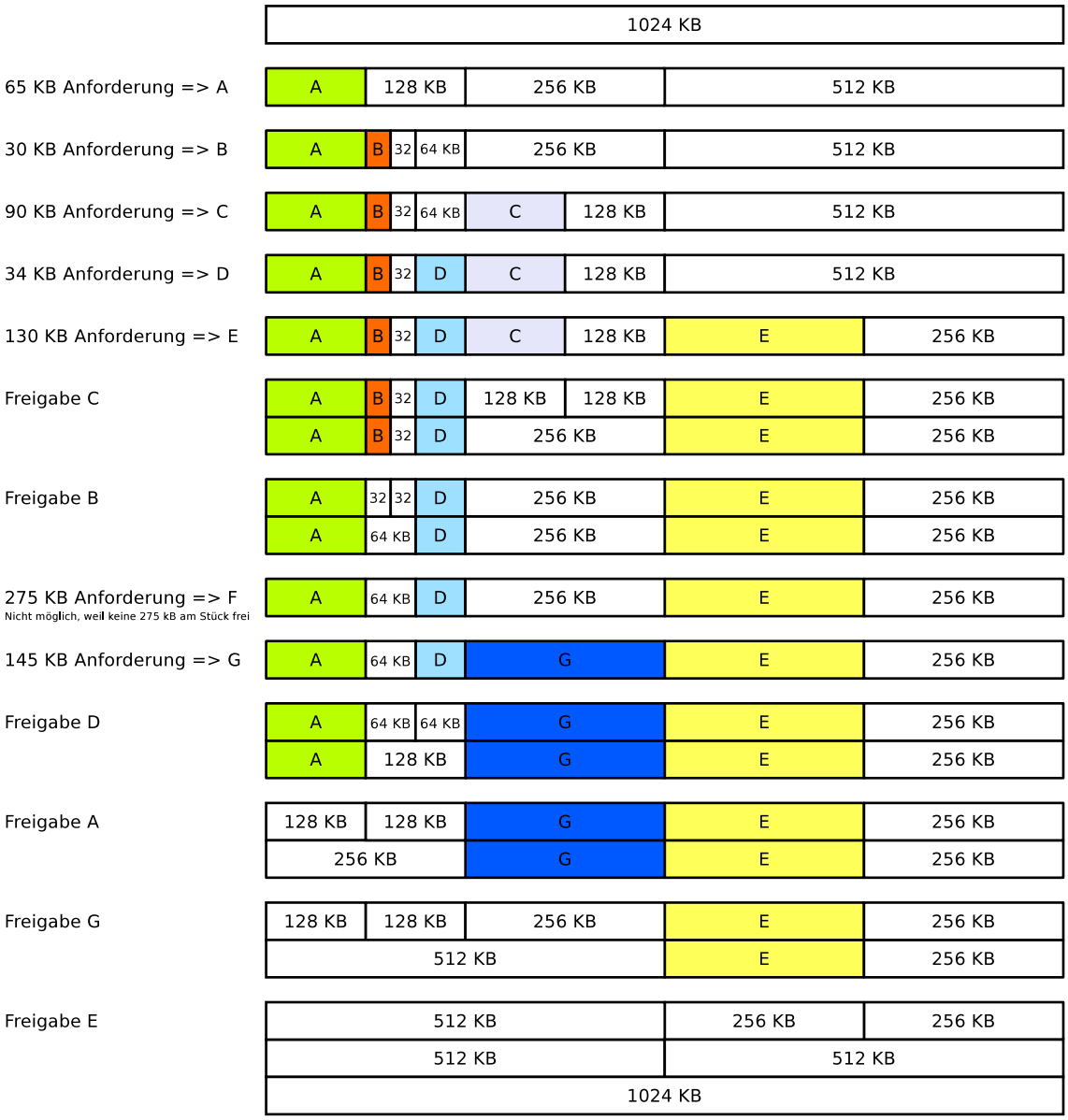
   ☐ First Fit ☒ Next Fit ☐ Best fit ☐ Random

9. Mark the memory management concept that produces many mini-fragments and is slow.

☐ First Fit ☐ Next Fit ☒ Best fit ☐ Random

Exercise 2 (Buddy Memory Allocation)

The Buddy method for allocating memory to processes shall be used for a memory with a capacity of 1024 kB. Perform the provided operations and give the occupancy state of the memory after each operation.



Exercise 3 (Real Mode and Protected Mode)

1. Describe the functioning of the real mode.

Each process can access the entire memory, which can be addressed.

2. Explain why it is impossible to use real mode for multitasking operation mode.

   It provides no memory protection.

3. Describe the functioning of the protected mode.

   Each process can only access its own virtual memory. Virtual memory addresses translates the CPU with the MMU into physical memory addresses.

4. Describe what virtual memory is.

   Each process has a separate address space. This address space is an abstraction of the physical memory. It implements virtual memory. It consists of logical memory addresses, which are numbered from address 0 upwards and it is independent from the storage technology used and the existing expansion options.

5. Explain, why virtual memory helps to better utilize the main memory.

   Processes do not need to be located in one piece inside the main memory. Therefore, the external fragmentation of the main memory is not a problem.

6. Describe what mapping is.

   The virtual memory is mapped to the physical memory.

7. Describe what swapping is.

   The process of relocating data from the main memory to the SDD/HDD and back.

8. Name the component of the CPU that is used to implement virtual memory.

   Memory Management Unit (MMU).

9. Describe the function of the component from subtask 8.

   Virtual memory addresses are translated into physical memory addresses by the CPU using the MMU.

10. Name a virtual memory concept.

    Paging.

11. Name the sort of fragmentation that does occur with the concept of subtask 10.

    Internal fragmentation. It can only occur in the last page of each process.

12. Explain what a page fault exception causes to occur.

    A process tries to access a page, which is not located in the physical main memory.

13. Describe the reaction of the operating system when a page fault exception occurs.

    The operating system handles the page fault exception by executing these steps:

    - *Allocate the page by using the controller and the device driver on the swap memory (SSD/HDD).*
    - *Copy the page into a free page of the main memory.*
    - *Update the page table.*
    - *Return control to the process. The process next tries to execute again the instruction that caused the page fault.*

14. Explain what an access violation exception or general protection fault exception causes to occur.

    A process tried to access a virtual memory address, which it is not allowed to access.

15. Describe the consequence (effect) of an access violation exception or general protection fault exception.

    In some legacy Windows operating systems, segmentation faults often caused system crashes and resulted in a blue screen. In Linux, the signal `SIGSEGV` *is returned as a result.*

16. Describe the content of the kernelspace.

    The operating system kernel and kernel extensions (drivers).

17. Describe the content of the userspace.

    The currently running process, which is extended with swap memory (Windows: page file).

Exercise 4 (Memory Management)

Please mark for each one of the following statements, whether the statement is true or false.

1. Real mode is suited for multitasking systems.

   ☐ True ☒ False

2. In protected mode, each process is executed in its own copy of the physical address space, which is protected from other processes.

   ☒ True ☐ False

3. When static partitioning is used, internal fragmentation occurs.

   ☒ True ☐ False

4. When dynamic partitioning is used, external fragmentation cannot occur.

   ☐ True ☒ False

5. With paging, all pages have the same length.

   ☒ True ☐ False

6. One advantage of long pages is little internal fragmentation.

   ☐ True ☒ False

7. A drawback of short pages is that the page table gets bigger.

   ☒ True ☐ False

8. When paging is used, the MMU translates the logical memory addresses into physical memory addresses.

   ☒ True ☐ False

9. Modern operating systems (for x86) operate in protected mode and use only paging.

   ☒ True ☐ False

Exercise 5 (Page Replacement Strategies)

1. Why is it impossible to implement the optimal replacement strategy OPT?

   Because it is not possible to predict the future and therefore the future request sequence is unknown.

2. Perform the access sequence with the replacement strategies Optimal, LRU, LFU and FIFO once with a cache with a capacity of 4 pages and once with 5 pages. Also calculate the hit rate and the miss rate for all scenarios.

Optimal replacement strategy (OPT):

Requests:	**1**	**3**	**5**	**4**	**2**	**4**	**3**	**2**	**1**	**0**	**5**	**3**	**5**	**0**	**4**	**3**	**5**	**4**	**3**	**2**	**1**	**3**	**4**	**5**
Page 1:	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	2	1	1	1	1
Page 2:		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Page 3:			5	5	2	2	2	2	2	2	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Page 4:				4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4

Hitrate: 15/24 = 0.625
Missrate: 9/24 = 0.375

Requests:	**1**	**3**	**5**	**4**	**2**	**4**	**3**	**2**	**1**	**0**	**5**	**3**	**5**	**0**	**4**	**3**	**5**	**4**	**3**	**2**	**1**	**3**	**4**	**5**
Page 1:	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Page 2:		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Page 3:			5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Page 4:				4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
Page 5:					2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1

Hitrate: 17/24 = 0.7083333
Missrate: 7/24 = 0.2916666

Replacement strategy Least Recently Used (LRU):

Requests:	**1**	**3**	**5**	**4**	**2**	**4**	**3**	**2**	**1**	**0**	**5**	**3**	**5**	**0**	**4**	**3**	**5**	**4**	**3**	**2**	**1**	**3**	**4**	**5**
Page 1:	1	1	1	1	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3
Page 2:		3	3	3	3	3	3	3	3	3	5	5	5	5	5	5	5	5	5	5	1	1	1	1
Page 3:			5	5	5	5	5	5	1	1	1	1	1	1	4	4	4	4	4	4	4	4	4	4
Page 4:				4	4	4	4	4	4	0	0	0	0	0	0	0	0	0	0	2	2	2	2	5
Queue:	1	3 1	5 3 1	4 5 3 1	2 4 5 3	4 2 5 3	3 4 2 5	2 3 4 5	1 2 3 4	0 1 2 3	5 0 1 2	3 5 0 1	5 3 0 1	0 5 3 1	4 0 5 3	3 4 0 5	5 3 4 0	4 5 3 0	3 4 5 0	2 3 4 5	1 2 3 4	3 1 2 4	4 3 1 2	5 4 3 1

Hitrate: 11/24 = 0.4583333%
Missrate: 13/24 = 0.5416666%

Requests:	**1**	**3**	**5**	**4**	**2**	**4**	**3**	**2**	**1**	**0**	**5**	**3**	**5**	**0**	**4**	**3**	**5**	**4**	**3**	**2**	**1**	**3**	**4**	**5**
Page 1:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2
Page 2:		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Page 3:			5	5	5	5	5	5	5	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
Page 4:				4	4	4	4	4	4	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Page 5:					2	2	2	2	2	2	2	2	2	2	4	4	4	4	4	4	4	4	4	4
Queue:	1	3 1	5 3 1	4 5 3 1	2 4 5 3 1	4 2 5 3 1	3 4 2 5 1	2 3 4 5 1	1 2 3 4 5	0 1 2 3 4	5 0 1 2 3	3 5 0 1 2	5 3 0 1 2	0 5 3 1 2	4 0 5 3 1	3 4 0 5 1	5 3 4 0 1	4 5 3 0 1	3 4 5 0 1	2 3 4 5 0	1 2 3 4 5	3 1 2 4 5	4 3 1 2 5	5 4 3 1 2

Hitrate: 14/24 = 0.583333%
Missrate: 10/24 = 0.416666%

Replacement strategy Least Frequently Used (LFU):

Requests:	**1**	**3**	**5**	**4**	**2**	**4**	**3**	**2**	**1**	**0**	**5**	**3**	**5**	**0**	**4**	**3**	**5**	**4**	**3**	**2**	**1**	**3**	**4**	**5**
Page 1:	1_1	1_1	1_1	1_1	2_1	2_1	2_1	2_2	2_2	2_2	2_2	2_2	2_2	2_2	2_2	2_2	2_2	2_2	2_2	2_3	2_3	2_3	2_3	2_3
Page 2:		3_1	3_1	3_1	3_1	3_1	3_2	3_2	3_2	3_2	3_2	3_3	3_3	3_3	3_3	3_4	3_4	3_4	3_5	3_5	3_5	3_6	3_6	3_6
Page 3:			5_1	5_1	5_1	5_1	5_1	5_1	1_1	0_1	5_1	5_1	5_2	5_2	5_2	5_2	5_3	5_3	5_3	5_3	5_3	5_3	5_3	5_4
Page 4:				4_1	4_1	4_2	4_2	4_2	4_2	4_2	4_2	4_2	4_2	0_1	4_1	4_1	4_1	4_2	4_2	4_2	1_1	1_1	4_1	4_1

Hitrate: 12/24 = 0.5
Missrate: 12/24 = 0.5

Requests:	**1**	**3**	**5**	**4**	**2**	**4**	**3**	**2**	**1**	**0**	**5**	**3**	**5**	**0**	**4**	**3**	**5**	**4**	**3**	**2**	**1**	**3**	**4**	**5**
Page 1:	1_1	1_1	1_1	1_1	1_1	1_1	1_1	1_1	1_2	1_2	1_2	1_2	1_2	0_1	0_1	0_1	0_1	0_1	0_1	0_1	1_1	1_1	1_1	1_1
Page 2:		3_1	3_1	3_1	3_1	3_1	3_2	3_2	3_2	3_2	3_2	3_3	3_3	3_3	3_3	3_3	3_4	3_4	3_5	3_5	3_5	3_6	3_6	3_6
Page 3:			5_1	5_1	5_1	5_1	5_1	5_1	5_1	0_1	5_1	5_1	5_2	5_2	5_2	5_2	5_3	5_3	5_3	5_3	5_3	5_3	5_3	5_4
Page 4:				4_1	4_1	4_2	4_2	4_2	4_2	4_2	4_2	4_2	4_2	4_2	4_3	4_3	4_3	4_4	4_4	4_4	4_4	4_4	4_5	4_5
Page 5:					2_1	2_1	2_1	2_2	2_2	2_2	2_2	2_2	2_2	2_2	2_2	2_2	2_2	2_2	2_2	2_3	2_3	2_3	2_3	2_3

Hitrate: 15/24 = 0.625
Missrate: 9/24 = 0.375

Replacement strategy FIFO:

Requests:	**1**	**3**	**5**	**4**	**2**	**4**	**3**	**2**	**1**	**0**	**5**	**3**	**5**	**0**	**4**	**3**	**5**	**4**	**3**	**2**	**1**	**3**	**4**	**5**
Page 1:	1	1	1	1	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3	5
Page 2:		3	3	3	3	3	3	3	1	1	1	1	1	1	4	4	4	4	4	4	4	4	4	4
Page 3:			5	5	5	5	5	5	5	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2
Page 4:				4	4	4	4	4	4	4	5	5	5	5	5	5	5	5	5	5	1	1	1	1

Hitrate: 11/24 = 0.4583333
Missrate: 13/24 = 0.5416666

Requests:	**1**	**3**	**5**	**4**	**2**	**4**	**3**	**2**	**1**	**0**	**5**	**3**	**5**	**0**	**4**	**3**	**5**	**4**	**3**	**2**	**1**	**3**	**4**	**5**
Page 1:	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Page 2:		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	1	1	1	1
Page 3:			5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	3	3	3
Page 4:				4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	5
Page 5:					2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

Hitrate: 15/24 = 0.625
Missrate: 9/24 = 0.375

3. What is the key message of Laszlo Belady's anomaly?

   FIFO produces worse results for certain access patterns with increased memory.

4. Show Belady's anomaly by performing the access sequence with the replacement strategy FIFO once with a cache with a capacity of 3 pages and once with 4 pages. Also calculate the hit rate and the miss rate for both scenarios.

Requests:	**3**	**2**	**1**	**0**	**3**	**2**	**4**	**3**	**2**	**1**	**0**	**4**
Page 1:	**3**	3	3	**0**	0	0	**4**	4	4	4	4	4
Page 2:		**2**	2	2	**3**	3	3	**3**	3	**1**	1	1
Page 3:			**1**	1	1	**2**	2	2	**2**	2	**0**	0

Hitrate: 3/12 = 25%
Missrate: 9/12 = 75%

Requests:	**3**	**2**	**1**	**0**	**3**	**2**	**4**	**3**	**2**	**1**	**0**	**4**
Page 1:	**3**	3	3	3	**3**	3	**4**	4	4	4	**0**	0
Page 2:		**2**	2	2	2	**2**	2	**3**	3	3	3	**4**
Page 3:			**1**	1	1	1	1	1	**2**	2	2	2
Page 4:				**0**	0	0	0	0	0	**1**	1	1

Hitrate: 2/12 = 16.66%
Missrate: 10/12 = 83.33%

Exercise 6 (Time-based Command Execution, Sorting, Environment Variables)

1. Create in your home directory a directory `NotImportant` and write a cron job, which erases the content of the directory `NotImportant` every Tuesday at 1:25 clock am.

   The output of the command should be appended to a file `EraseLog.txt` in your home directory.

   ```
   $ mkdir ~/NotImportant
   $ crontab -e
   ```

   Insert these lines:

   ```
   25 1 * * 2 rm -rfv /home/USERNAME/NotImportant/* >>
   /home/USERNAME/EraseLog.txt
   ```

2. Write a cron job, which appends a line at a file `Datum.txt` with the following format (but with the current values) every 3 minutes between 14:00 to 15:00 clock on every Tuesday in the month of November:

```
Heute ist der 30.10.2008
Die Uhrzeit ist 09:24:42 Uhr
**************************
```

```
$ crontab -e
```

Insert these lines:

```
*/3 14,15 * 11 * date+"Heute ist de %x%nDie Uhrzeit ist
%H:%M:%S Uhr%n**************************" >> Datum.txt
```

3. Write an at-job, which outputs at 17:23 today a list of the running processes.

> *You may have to install the command line tool* `at` *first.*
> *With Debian/Ubuntu this works with:*
> *`$ sudo apt update && sudo apt install at`*
> *With CentOS/Fedora/RedHat this works with:*
> *`$ sudo yum install at`*

```
$ at 1725 today
```

Insert these lines:

```
ps -r
```

4. Write an at-job, which outputs at December 24th at 8:15 am the text „It's christmas!“

```
$ at 0815 DEZ 25
```

Insert these lines:

```
echo "It's christmas!"
```

5. Create in your home directory a file `Kanzler.txt` with the following content:

```
Willy     Brandt    1969
Angela    Merkel    2005
Gerhard   Schröder  1998
KurtGeorg Kiesinger 1966
Helmut    Kohl      1982
Konrad    Adenauer  1949
Helmut    Schmidt   1974
Ludwig    Erhard    1963
```

```
$ echo "Willy     Brandt    1969" >> ~/Kanzler.txt
$ echo "Angela    Merkel    2005" >> ~/Kanzler.txt
```

```
$ echo "Gerhard   Schröder  1998" >> ~/Kanzler.txt
$ echo "KurtGeorg Kiesinger 1966" >> ~/Kanzler.txt
$ echo "Helmut    Kohl      1982" >> ~/Kanzler.txt
$ echo "Konrad    Adenauer  1949" >> ~/Kanzler.txt
$ echo "Helmut    Schmidt   1974" >> ~/Kanzler.txt
$ echo "Ludwig    Erhard    1963" >> ~/Kanzler.txt
```

6. Print out the file `Kanzler.txt` sorted by the first names.

   ```
   $ sort ~/Kanzler.txt
   ```

7. Print out the file `Kanzler.txt` sorted by the third letter of the last names.

   ```
   $ sort -k+2.4 ~/Kanzler.txt
   ```

8. Print out the file `Kanzler.txt` sorted by the year of the inauguration.

   ```
   $ sort -k3 ~/Kanzler.txt
   ```

9. Print out the file `Kanzler.txt` backward reverse sorted by the year of the inauguration and redirect the output into a file `Kanzlerdaten.txt`.

   ```
   $ sort -k3 -nr ~/Kanzler.txt > ~/Kanzlerdaten.txt
   ```

10. Create with the command `export` an environment variable `VAR1` and assign it the value `Testvariable`.

    ```
    $ export VAR01=Testvariable
    ```

11. Print out the value of `VAR1` in the shell.

    ```
    $ printenv VAR01
    ```

12. Erase the environment variable `VAR1`.

    ```
    $ unset VAR01
    ```