Operating Systems

Example solutions for Theoretical Exercise 4

Michael Engel

4.1 Replacement strategies

Perform and visualize (as shown in lecture 10) the access sequence with the replacement strategies FIFO, Optimal and LRU once with a memory with a capacity of 4 pages and once with 5 pages. Calculate the “hit rate” (accesses which did not result in a replacement operation) for all scenarios.

Request sequence: 1,3,5,4,2,4,3,2,1,0,5,3,5,0,4,3,5,4,3,2,1,3,4,5

4.1 Replacement strategies

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

4.1 Replacement strategies

Optimal:

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

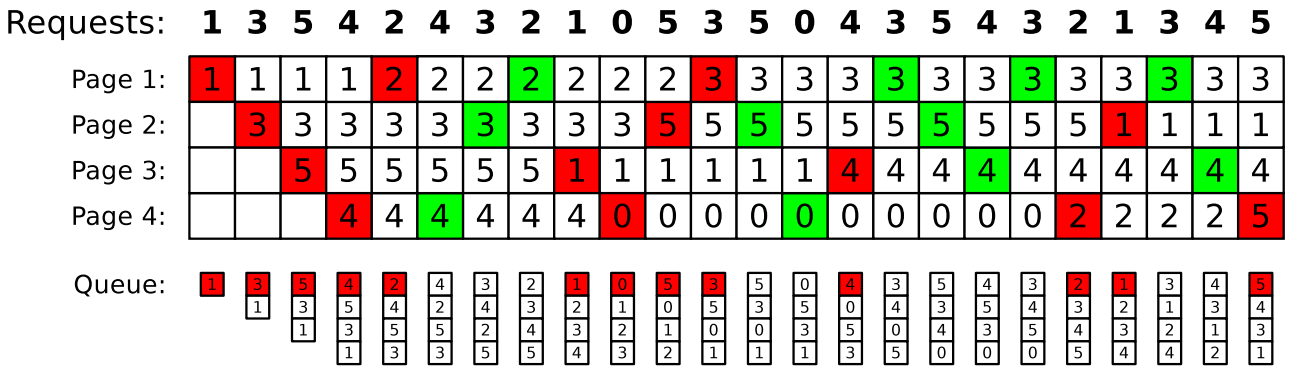
Hitrate: 15/24 = 0.625
Missrate: 9/24 = 0.375

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

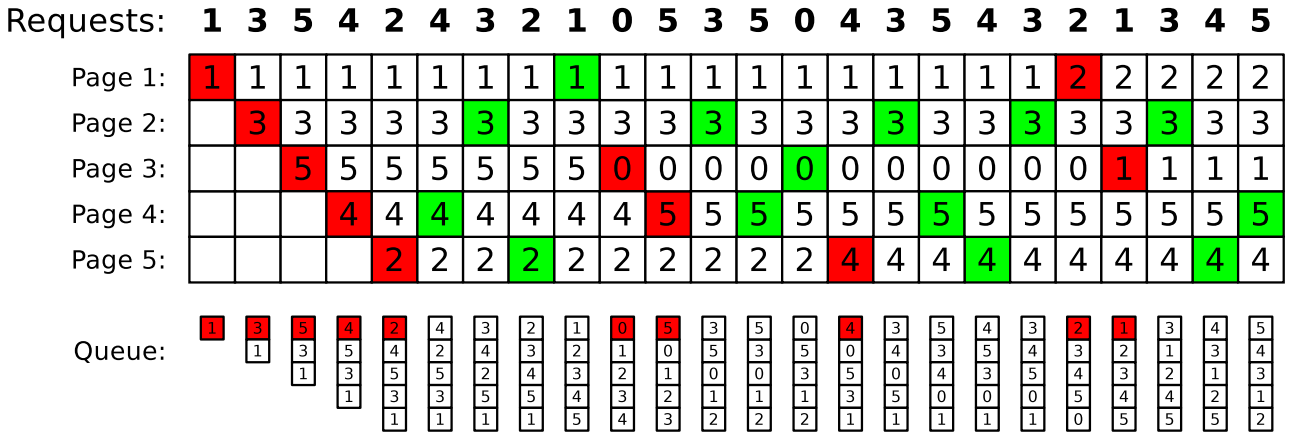
Hitrate: 17/24 = 0.7083333
Missrate: 7/24 = 0.2916666

4.1 Replacement strategies

LRU:



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



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

4.2 More replacement strategies

A computer has four page frames. The time of loading, time of last access, and the R (reference) and M (modified) bits for each page are as shown below (the times are in clock ticks):

Page	Loaded	Last ref.	R	M
0	126	280	1	0
1	230	265	0	1
2	140	270	0	0
3	110	285	1	1

Which pages will the algorithms FIFO, LRU and Second Chance replace? Explain your answer!

FIFO: Page 3 because it is loaded at 110 (First In)

LRU: Page 1 because is referenced at 265 (Least Recently)

2nd ch.: Page 2 because it is loaded at 140 and the reference bit is 0.

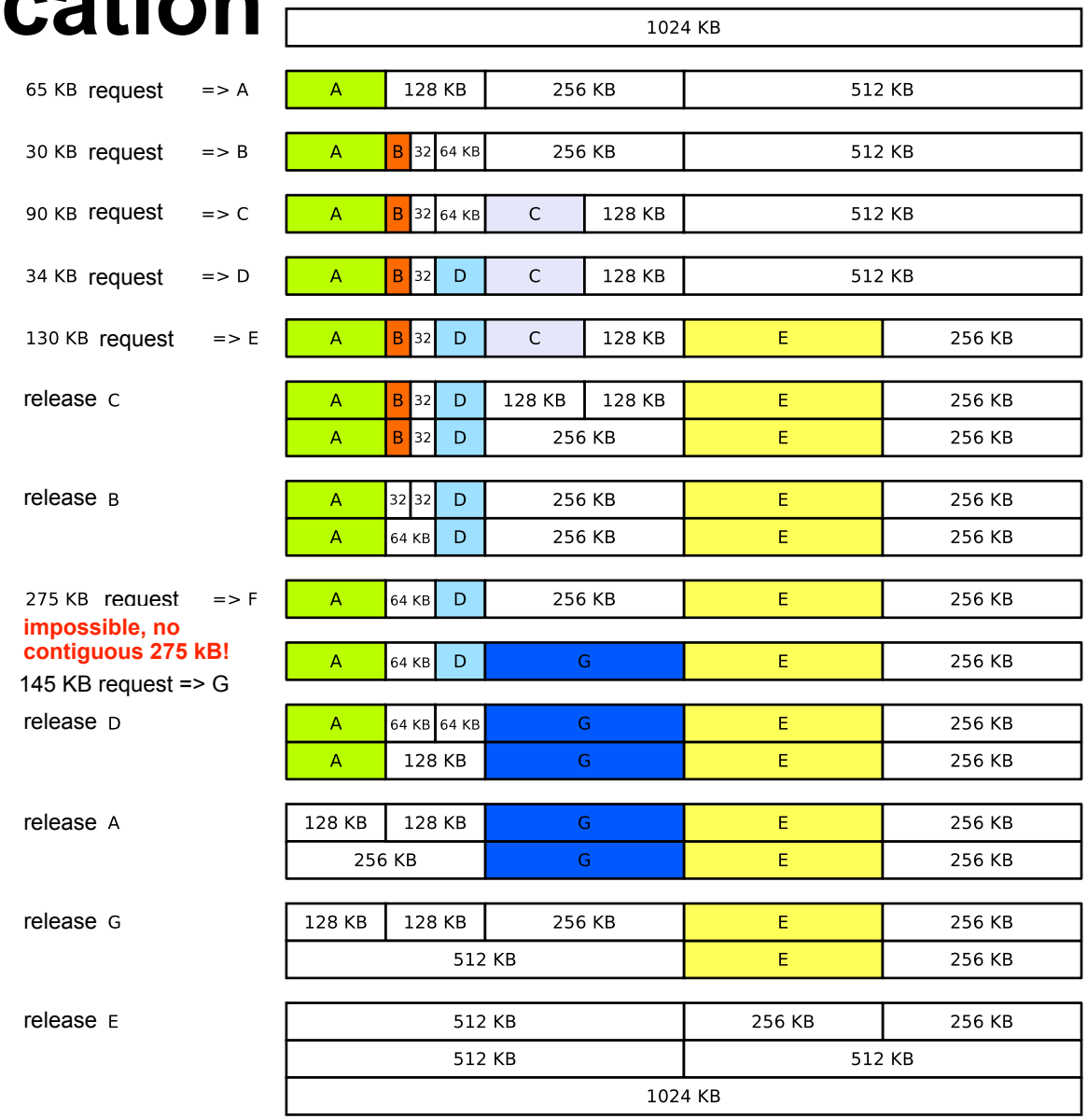
4.3 Buddy allocation

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. Indicate if an allocation cannot be satisfied.

a. Request 65 kB (A)
b. Request 30 kB (B)
c. Request 90 kB (C)
d. Request 34 kB (D)
e. Request 130 kB (E)
f. Release C
g. Release B
h. Request 275 kB (F)
i. Request 145 kB (G)
j. Release D
k. Release A
l. Release G
m. Release E

4.3 Buddy allocation

a. Request 65 kB (A)
b. Request 30 kB (B)
c. Request 90 kB (C)
d. Request 34 kB (D)
e. Request 130 kB (E)
f. Release C
g. Release B
h. Request 275 kB (F)
i. Request 145 kB (G)
j. Release D
k. Release A
l. Release G
m. Release E



4.4 Virtual memory

A machine has a physical memory with 2^{32} addressable bytes and a page size of 8 kB. Each process is allocated a virtual address space of 4 GB. Page table entries are 32 bits long. Page tables are kept in pageable memory.

a. Why is one-level paging inadequate for this system?

A one-level page table would imply that the first (and only) level of the table has space for all page entries.

# of pages: 4 GB / 8 kB = 2^{32} / 2^{13} = 2^{32-13} = 2^{19}

Each page requires a 32 bit = 4 byte entry in the page table.

Thus, the page table ***for each process*** would require:
2^{19} * 4 bytes = 2 MB.

This is quite a waste of memory…

4.4 Virtual memory

A machine has a physical memory with 2^{32} addressable bytes and a page size of 8 kB. Each process is allocated a virtual address space of 4 GB. Page table entries are 32 bits long. Page tables are kept in pageable memory.

b. Why is two-level paging sufficient?

The page table structure is a tradeoff between the size of the table and the overhead to look up a page entry in case of a TLB miss.

For 32 bit virtual addresses, level 0 and 1 page directories will have around 1024 entries or less (see part c), so a page directly will comfortably fit inside a page of the virtual memory system.

The sv32 page table structure of RISC-V (RV32) uses such a two-level page table with 4 kB pages and 2^{10}=1024 entries for level 0 and 1.

4.4 Virtual memory

A machine has a physical memory with 2^{32} addressable bytes and a page size of 8 kB. Each process is allocated a virtual address space of 4 GB. Page table entries are 32 bits long. Page tables are kept in pageable memory.

c. How many bits are needed to reference the outer page table and how many to reference the inner page table?
Explain your answer showing all appropriate arithmetic.

A two-level page table splits a virtual address into three parts, e.g.:

The offset is 13 bits (2^{13} = 8192 = 8 kB), so we need to split the remaining 32-13 = 19 bits into two halves

e.g. 10 bits for a level 1 entry and 9 for a level 2 entry

This would imply a level 1 page table size of 2^{10} * 4 bytes = 4096 b.
A level 2 page table would only use 2^{9} * 4 bytes = 2048 bytes

4.5 Paging and memory accesses

Consider the following 2D array (assume `sizeof(int)=8`):
`int X[32][32];`

Suppose that a system uses 4 pages of 512 byte page size each.

The X array is stored in row-major order (i.e., `X[0][1]` follows `X[0][0]` in memory).

Which of the following two code fragments will generate the lower number of page faults?

Compute the total number of page faults for each code fragment. Explain your calculation.

4.5 Paging and memory accesses

Consider the following 2D array (assume `sizeof(int)=8`):

```
int X[32][32];
```

Suppose that a system uses 4 pages of 512 byte page size each.

The X array is stored in row-major order (i.e., `X[0][1]` follows `X[0][0]` in memory).

Fragment 1:

```
for (int j=0; j<32; j++)
  for (int i=0; i<32; i++)
    X[i][j]++;
```

# of page faults:
A frame is 64 (=512/8) words
⇒ one row of the X array occupies half of a page (i.e., 32 words)

The entire array fits into 32 × 16/64 = 8 frames

The inner loop of the code steps through consecutive rows of X for a given column.

Thus every other reference to X[i][j] will cause a page fault.

⇒The total number of page faults will be 32 × 32/2 = 512.

4.5 Paging and memory accesses

Consider the following 2D array (assume `sizeof(int)=8`):
`int X[32][32];`

Suppose that a system uses 4 pages of 512 byte page size each.

The X array is stored in row-major order (i.e., `X[0][1]` follows `X[0][0]` in memory).

Fragment 2:

```
for (int i=0; i<32; i++)
  for (int j=0; j<32; j++)
    X[i][j]++;
```

# of page faults:
A frame is 64 (=512/8) words
⇒ one row of the X array occupies half of a page (i.e., 32 words)

The entire array fits into 32 × 16/64 = 8 frames

Fragment 2 will generate fewer page faults since the code has more ***spatial locality*** than Fragment 1.
The inner loop causes only one page fault for every other iteration of the outer loop.
⇒There will only be 16 page faults.