Operating Systems 2016/17
Assignment 9

Prof. Dr. Frank Bellosa
Dipl.-Inform. Marc Rittinghaus

Submission Deadline: Monday, January 9th, 2017 – 23:59

A new assignment will be published every week, right after the last one was due. It must be completed before its submission deadline.

The assignments must be filled out online in ILIAS. Handwritten solutions are no longer accepted. You will find the online version for each assignment in your tutorial's directory. **P-Questions** are programming assignments. Download the provided template from ILIAS. Do not fiddle with the compiler flags. Submission instructions can be found on the first assignment.

In this assignment you will get familiar with segmentation-based address translation.

T-Question 9.1: Segmentation

a. What is the benefit of using virtual addresses instead of working directly on physical addresses? **1 T-pt**

b. How does the MMU determine where the segment table is located in memory? **1 T-pt**

c. Assume a system with 16-bit virtual addresses that supports four different segments, which uses the following segment table: **4 T-pt**

Segment Number	Base	Limit
0	0x1000	0x15f3
1	0x43ab	0x0300
2	0x25f3	0x1000
3	0x8000	0x00FF

Complete the following table and explain briefly how you derived your solution for each row in the table.

Virtual Address	Segment Number	Offset	Valid?	Physical Address
0xA019				
	1	0x0200		
0xC0DE				
			yes	0x25f3

P-Question 9.1: MMU for Segmentation

Download the template **p1** for this assignment from ILIAS. You may only modify and upload the file `mmu.c`.

Consider a system that uses a segmentation-based 32 bit virtual to 32 bit physical address translation. The MMU performs the translation for a virtual address by interpreting the high 3 bits as index into a segment table and the remaining bits as offset.

In this question you will implement the MMU in software.

a. Write a function that performs the translation from a virtual to a physical address. Your implementation should fulfill the following requirements: **3 P-pt**

- Splits the value pointed to by `address` into the segment index and offset.
- Performs a look up in the segment table `table` and, if the supplied address is valid, calculates the physical address.
- On success, writes the physical address into the variable pointed to by `address` and returns 0, -1 otherwise.
- Also returns -1 if the table is not configured.

```
int translateSegmentTable(uint32_t *address);
```

b. The MMU should be extended with a translation lookaside buffer (TLB), that keeps up to `TLB_SIZE` entries, each one representing an exact translation (e.g., virtual address `0x0000ABCD` → physical address `0x00A6ABCD`). The TLB should use the least-recently-used (LRU) replacement strategy, based on a counter that is incremented on every access to the TLB (e.g., adding new entries, performing a lookup, etc.). Define the necessary data structures to represent the TLB and its entries and write a function that clears the whole TLB. *Hints:* Since the TLB size is known at compile time, you can use static arrays where needed. **1 P-pt**

```
void flushTLB(void);
```

c. Write a function that adds a new translation to the cache. The function should apply the LRU strategy to find and replace an entry if no free entries exist. *Hints:* The LRU strategy selects the entry as victim that has not been used for the longest time. **1 P-pt**

```
void addToTLB(uint32_t virtual, uint32_t physical);
```

d. Write a function that performs a lookup for the supplied virtual address in the TLB. On success, the function should pass the physical address by updating the value pointed to by `address` and return 0. If a matching translation does not exist, the function should return -1. *Hints:* Remember that the TLB uses the LRU strategy for replacement. **1 P-pt**

```
int translateTLB(uint32_t *address);
```

Total: 6 T-pt 6 P-pt