Exercise sheet 10 – Associative memory

Goals:

- Associative memory (Cache, TLB)
- Virtual memory on a 64 bit system

Exercise 10.1: Cache (theoretical)

Consider an architecture with the following details:

- 16 bit architecture
- Memory address: 0x00014494
- Data: 0xAFFE

(a) State the cache entry, considering a data size of a *word*. *Hint: Check out the size of a word.*

Proposal for solution:

Key	Value
Real address	**Data**
0x00014494	0xAFFE

Word → 2 bytes

(b) State the cache entry, considering a cache line size of *256 bytes*.

Proposal for solution:

Key	Value					
Real address	**Data**					
0x000144	0x??	…	0xAF	0xFE	…	0x??
Position	#00		#94	#95		#FF

(c) Exactly state the position of the value (0xAFFE) inside the 256 byte data entry.

Proposal for solution: Check out the solutions for the previous part.
0xAFFE is 2 bytes, which is the size of a word for the given 16 bit architecture.
For a block size of 256 bytes, 0xAF is at position #94, and 0xFE on #95, respectively (for those who wants to know it in detail: for a BE architecture!).

(d) What is happening if the cache is empty and we try to access the cache line?

Proposal for solution:
First: Load the data for the whole cache line from the memory into the cache.
Then: The data can be accessed through the cache, e.g. loaded into a register.

Exercise 10.2: TLB 1

Consider an architecture for page addressing similar to the 1 level page table in the lecture about the VM/MMU, but with a 32 bit architecture.

- The TLB *(translation lookaside buffer)* contains the following entry:

Key	Value
0x00009	0x00002

- The page table contains the following entries:

Page table offset	Value
...	
0x0000A	0x00003
0x00009	0x00002
0x00008	0x00001
...	

(a) Is the page table and the TLB consistent?

Proposal for solution: TLB-Key: 0x00009
TLB-Value: 0x00002
Page-Table-Entry 0x00009 contains 2
Yes they are consistent.

(b) State the virtual address, which is mapped to the 32 bit real address 0x00002AAA.

Proposal for solution: The *value* in the TLB is the real address, the corresponding *key* is the virtual address, the offset can be extracted from the given real address.
Offset: 0xAAA
Frame number: 0x00002
Page number: 0x00009
Virtual address: 0x00009AAA

Exercise 10.3: TLB 2

Consider an architecture for page addressing similar to the 2 level page table in the lecture for a 32 bit architecture with a 4 KiB page/frame size.
Given:

- Virtual address: 0x1202F494
- Real address: 0x00014494

(a) State the entry in the TLB for the given addresses.

Proposal for solution:

Page base address	Frame base address
0x1202F	0x00014

(b) What is happening if the TLB is empty and we try to access the page?

Proposal for solution:

- Load page table(s)
- Lookup inside page table(s)
- Address translation
- Store address into TLB