# Exercise sheet 11 – Bus sequences

**Goals:**

- Program sequence and resulting bus cycles
- Cache influence on bus cycles
- Isolated I/O
- Memory mapped I/O

### Exercise 11.1: Program sequence and resulting bus cycles

Consider a 32-bit CPU **without caches**.

Given is following instruction-sequence:

```
Word 1:  Code for SUB R1, X          ; X = X - R1
Word 2:  Address of X
Word 3:  Code for ADD #4711, R2      ; R2 = R2 + 4711
Word 4:  Operand 4711                ; Direct operand
Word 5:  Code for MOVE (R0)+, (R1)   ; (R1) = (R0)
                                     ; R0 and R1 may contain addresses
                                     ; (R0)+: Post increment of R0
```

- 32 bit word in each memory line
- Rx stands for data- or address registers

*Hint: You may want to draw a table. A spreadsheet software (Excel, LibreOffice) or a paper is your friend.*

| Nr. | Master | Cycle | Comment | $\alpha$ | $\beta$ | $\gamma_1$ | $\gamma_2$ |
|---|---|---|---|---|---|---|---|
| 1 | | | | | | | |
| ... | | | | | | | |

(a) State a possible sequence of resulting bus cycles.

### Exercise 11.2: Cache influence on bus cycles

Consider a 32-bit CPU **with caches**.

State the changes for *exercise 11.1* resulting in the usage of different caches.

*Hint: Addresses of variables (direct addresses) and direct operands are considered as instructions.*

Consider following cases:

(a) Common cache for data and instructions (**perfectly filled**): Which cycles may be obsolete now? *Hint: Mark them with* $\alpha$*.*

(b) Cache for instructions (**perfectly filled**): Which cycles may be obsolete now? *Hint: Mark them with* $\beta$*.*

(c) Cache for data with *write through* (**perfectly filled**): Which cycles may be obsolete now? *Hint: Mark them with* $\gamma_1$*.*

(d) Cache for data with *write back* (**perfectly filled**): Which cycles may be obsolete now? *Hint: Mark them with* $\gamma_2$.

### Exercise 11.3: Isolated I/O with Tinkercad circuits (coding)

The idea is to continuously toggle the built-in LED of the Arduino Uno. For that the isolated I/O functions should be used.

*Hint: You may find the PIN Mapping, the ATMEGA 328 Datasheet, and the AVR Instruction Set Manual useful.*

(a) On the Arduino Uno, the built-in LED is on digital pin 13. On which physical pin is the digital pin 13 mapped and how is it called? Use the PIN Mapping for that.

(b) The physical pin is part of a register with 8 bits. How is this called and on which position in the 8 bit register is the physical pin mapped? You may use the ATMEGA 328 Datasheet to find this. *Hint: Look at page 72, section 13.4.2.*

(c) Which value do you have to write into this register, to enable (switch on)/disable (switch off) the built-in LED?

(d) Find the register address where the physical pin of the built-in LED is contained. You may again use the ATMEGA 328 Datasheet to find this. *Hint: Look at page 72, section 13.4.2.: the first HEX value.*

(e) Find an assembler instruction with which you can directly write to the I/O register. You may use the AVR Instruction Set Manual to find this. *Hint: You may have a look on page 134.*

(f) Create a new circuit using the *Starters Arduino: „Blink“*

(g) Copy the content of the `RA_exercises/sheet_11/io_prog_isolated_io_tinkercad/io_prog_isolated_io_tinkercad.ino` template into the code part of your Tinkercad circuit.

(h) Follow the TODOs in the code and use the already collected information about the registers, addresses, values, and assembler instructions to complete the code.

### Exercise 11.4: Memory mapped I/O with Tinkercad circuits (coding)

The idea is to continuously toggle the built-in LED of the Arduino Mega. For that memory mapped I/O should be used.

*Hint: You may find the PIN Mapping, the ATMEGA 328 Datasheet, and the AVR Instruction Set Manual useful.*

(a) Find the memory address of the register address where the physical pin of the built-in LED is connected. You may again use the ATMEGA 328 Datasheet to find this. *Hint: Look at page 72, section 13.4.2.: the second HEX value inside the parenthesis.*

(b) Find an assembler instruction with which you can write data from a register into the memory (data space/SRAM). You may use the AVR Instruction Set Manual to find this. *Hint: You may have a look on page 179.*

(c) Create a new circuit using the *Starters Arduino: „Blink“*

(d) Copy the content of the `RA_exercises/sheet_11/io_prog_memory_mapped_io_tinkercad/io_prog_memory_mapped_io_tinkercad.ino` template into the code part of your Tinkercad circuit.

(e) Follow the TODOs in the code and use the already collected information about the registers, addresses, values, and assembler instructions to complete the code.