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	α	β	γ_1	γ_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:

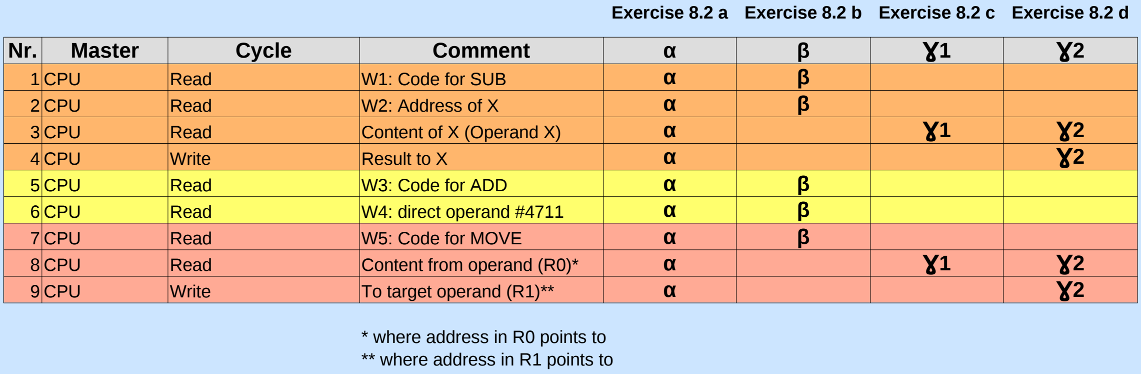
(a) Common cache for data and instructions (**perfectly filled**): Which cycles may be obsolete now? *Hint: Mark them with α.*

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

(c) Cache for data with *write through* (**perfectly filled**): Which cycles may be obsolete now? *Hint: Mark them with γ_1.*

(d) Cache for data with *write back* (**perfectly filled**): Which cycles may be obsolete now? *Hint: Mark them with* γ_2.

Proposal for solution: One combined solution for *exercises 11.1 and 11.2* .



				Exercise 8.2 a	Exercise 8.2 b	Exercise 8.2 c	Exercise 8.2 d
Nr.	**Master**	**Cycle**	**Comment**	**α**	**β**	**ɣ1**	**ɣ2**
1	CPU	Read	W1: Code for SUB	α	β		
2	CPU	Read	W2: Address of X	α	β		
3	CPU	Read	Content of X (Operand X)	α		ɣ1	ɣ2
4	CPU	Write	Result to X	α			ɣ2
5	CPU	Read	W3: Code for ADD	α	β		
6	CPU	Read	W4: direct operand #4711	α	β		
7	CPU	Read	W5: Code for MOVE	α	β		
8	CPU	Read	Content from operand (R0)*	α		ɣ1	ɣ2
9	CPU	Write	To target operand (R1)**	α			ɣ2

* where address in R0 points to
** where address in R1 points to

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.

Proposal for solution: It is mapped on physical pin 19 which is called PB5.

(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.*

Proposal for solution: The register is called PORTB and PB5 is mapped on bit 5 (numbered from 0 to 7). You may find this on page 72 of the ATMEGA 328 Datasheet.

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

Proposal for solution: To enable (switch on) the built-in LED we have to write a one into bit 5 of the PORTB register: 0b00100000 (binary) = 0x20 (hex). To disable (switch off) the built-in LED we have to write a zero into bit 7 of the PORTB register: 0b00000000 (binary) = 0x00 (hex).

(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.*

Proposal for solution: The PORTB register address is 0x05. You may find this on page 72 of the ATMEGA 328 Datasheet.

(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.*

Proposal for solution: The OUT instruction on page 134 is the right one.

(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.

Proposal for solution:

```
//TODO: set a "1" for ON and a "0" for OFF for the right bit into the register.
//use an appropriate number representation for that.
//".equ" is nearly similar to the C-preprocessor statement "#define"
asm (".equ ON,  0b00100000"); //value for enable (switch on)
asm (".equ OFF, 0b00000000"); //value for disable (switch off)

void setup() {
  //The built-in LED is connected to digital PIN 13, so we set it as an output
  pinMode(13, OUTPUT);
}

void loop() {
  //Switch LED on
  asm("ldi r16, ON"); //load the immediate value ON into r16
  asm("out 0x05,r16"); //use out to write the r16 content to the I/O address 0x05.

  delay(1000); //wait a second

  //Switch LED off
  asm("ldi r16, OFF"); //load the immediate value OFF into r16
  asm("out 0x05,r16"); //use out to write the r16 content to the I/O address 0x05.

  delay(1000); //wait a second
}
```

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.*

Proposal for solution: The memory address is 0x25. You may find this on page 72 of the ATMEGA 328 Datasheet.

(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.*

Proposal for solution: The STS instruction on page 179 is the right one.

(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.

Proposal for solution:

```
//".equ" is nearly similar to the C-preprocessor statement "#define"
asm (".equ ON,  0b00100000"); //value for enable (switch on)
asm (".equ OFF, 0b00000000"); //value for disable (switch off)

void setup() {
  //The built-in LED is connected to digital PIN 13, so we set it as an output
  pinMode(13, OUTPUT);
}

void loop() {
  //Switch on
  asm("ldi r16, ON"); //load the immediate value ON into r16
  asm("sts 0x25, r16"); //use sts to write the r16 content to the memory address 0x25.

  delay(1000); //wait a second

  //Switch off
  asm("ldi r16, OFF"); //load the immediate value OFF into r16
  asm("sts 0x25, r16"); //use sts to write the r16 content to the memory address 0x25.

  delay(1000); //wait a second
}
```