



Boolean Algebra

In the following tasks you are asked to write a Boolean expression. The exclamation mark (!) should be used to indicate the Not-function, if there is any (e.g. $\overline{A} = !A$)

(4%)

Write the Boolean expressions equivalent to the truth table below and then reduce it as much as possible.

The truth table has two inputs, A and B, and one output, F.

A	B	F
0	0	1
0	1	0
1	0	1
1	1	1

$$F = \boxed{A + !B}$$

$$!a!b+a!b+ab$$

$$!a!b+a(!b+b)$$

$$!a!b+a$$

$$!b+a$$

CAO test exam (repetition lecture)

This is the assignment as it will look to the participant. Close this browser window to return to the assignment overview.

Write the Boolean expressions equivalent to the truth table below, and then reduce it as much as possible.

The truth table has three inputs, A, B and C, and one output F.

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$F = AB + C$$

$$\neg a \neg bc + \neg a bc + a \neg bc + ab \neg c + abc$$

$$\neg ac + a \neg bc + ab$$

$$\neg ac + a(b + \neg bc)$$

$$\neg ac + a(b + c)$$

$$\neg ac + ab + ac$$

$$Ab + c(\neg a + a)$$

$$Ab + c$$

Two's Complement

(3%)

Calculate the two's complement of the binary number 01101110_2 on an 8-bit processor.

Write your answer as a 8-bit binary number:

₂

(3%)

Calculate the two's complement of the hexadecimal number $2B8_{16}$ on a 10-bit processor.

Write your answer as a 10-bit binary number:

₂



01101110

1.com: 10010001

2. com: 10010010

2B8=1010111000

1. com: 0101000111

2. com: 0101001000

ALU

(7%)

The description below shows the `ORI` instruction copied from atmega2560 instruction set:

87. ORI – Logical OR with Immediate

87.1. Description

Performs the logical OR between the contents of register Rd and a constant, and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd \vee K$

Syntax:

(i) `ORI Rd,K`

Operands:

$16 \leq d \leq 31, 0 \leq K \leq 255$

Program Counter:

$PC \leftarrow PC + 1$

16-bit Opcode:

0110	KKKK	dddd	KKKK
------	------	------	------

The assembler translates the following instruction into a binary pattern, which is burned into the program memory.

`ORI R20, 49`

Write your answer as a 16-bit binary pattern without any spaces:

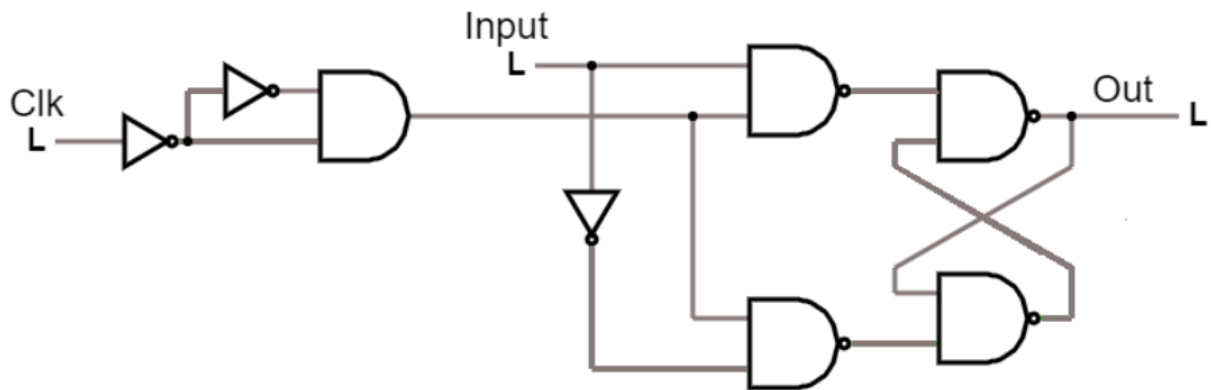
0110001101000001

8%

Which of the following elements is placed inside the CPU of the ATMEGA2560?

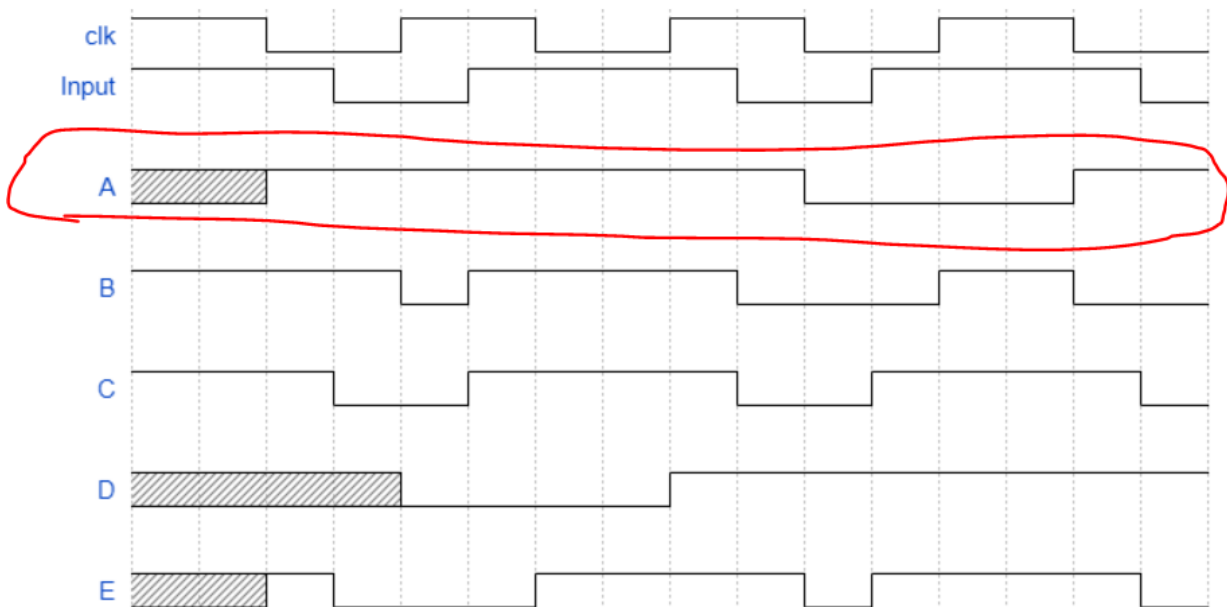
General purpose registers	<input checked="" type="radio"/> True	<input type="radio"/> False
SRAM	<input type="radio"/> True	<input checked="" type="radio"/> False
Flash memory	<input type="radio"/> True	<input checked="" type="radio"/> False
Program Counter	<input checked="" type="radio"/> True	<input type="radio"/> False
Input-output registers	<input type="radio"/> True	<input checked="" type="radio"/> False
Status Register	<input checked="" type="radio"/> True	<input type="radio"/> False
ALU	<input checked="" type="radio"/> True	<input type="radio"/> False
The Stack Pointer	<input checked="" type="radio"/> True	<input type="radio"/> False

The circuit below is a falling edge triggered data latch:



The timing diagram below shows the two signals to the data latch, clock (clk) and input.

Select the output line (A, B, C, D or E) which shows the correct data latch output (Out):



(3%)

A memory chip has 13 address pins.

How many memory locations are inside the chip?

$$2^{13} = 8192$$

Assembly language

(3%)

State the contents of R16 after the assembly code below has been executed:

```
LDI R16, 12
LDI R17, 19
MOV R16, R17
DEC R16
DEC R16
```

Write your response in decimal:

R16 =

(4%)

State the contents of R16 after the assembly code below has been executed:

```
LDI R16, 0x7F
ANDI R16, 0b00000110
ORI R16, 0b00100010
DEC R16
```

Write the value of R16 in decimal:

R16 =

(4%)

State the contents of R16 after the assembly code below has been executed:

```
LDI R16, 40  
LDI R17, 4  
MUL R16, R17  
LDI R17, 96  
MOV R16, R0  
ADD R16, R17
```

Write the value of R16 in decimal:

$R16 =$

(7%)

The following code is executed. When reaching the last NOP-instruction; how is the stack ordered? Order the stack correctly in the column to the right, such that the bottom row represent the bottom of the stack.

;initialize stack pointer

LDI R16, 0xFF

OUT SPL, R16

LDI R16, 0x21

OUT SPH, R16

LDI R16, 16

LDI R17, 20

LDI R18, 30

LDI R30, 40

PUSH R16

PUSH R17

PUSH R18

POP R17

POP R18

CALL FUNC

FUNC:

PUSH R17

PUSH R18

PUSH R30

NOP

Source



Target

≡ 40
≡ 20
≡ 30
≡ "Return address"
≡ 16



Delays

(6%)

How many clock cycles does it take to execute the assembly code below?

```
LDI R16, 100
LOOP:
NOP
DEC R16
NOP
BRNE LOOP
```

500

(6%)

How many clock cycles does it take to execute the assembly code below?

```
LDI R18, 10
LOOP2:
NOP
LDI R20, 25
LOOP1:
NOP
NOP
DEC R20
BRNE LOOP1
NOP
DEC R18
BRNE LOOP2
```

1300

Assembly Coding



(3%)

Write assembly code that implements this functionality:

Turn on an LED connected to PA7.

Turn off an LED connected to PA3.

All of PORT A has LEDs connected which are active high.



B <i>I</i> <u>U</u>   á
LDI R16, 0xFF OUT DDRA, R16 SBI PORTA, 7 CBI PORTA, 3
10000 Word limit

(4%)

Write assembly code that implements this functionality:

Add two 16 bit numbers, which are in the two register pairs R16, R17 and R30, R31

Output the least significant byte of the result to PORTB and the most significant byte of the result to PORTC.



B <i>I</i> <u>U</u>   á
LDI R18, 0xFF OUT DDRB, R18 OUT DDRC, R18 ADD R17, R31 ADC R16, R30 OUT PORTB, R17 OUT PORTC, R16
10000 Word limit

(4%)

Write assembly code that implements this functionality:

Read the input of PA4.

If the PA4 is high, set PA3, otherwise clear PA3.

B <i>I</i> <u>U</u>   á
CBI DDRA, 4 SBIC PINA, 4 SBI PORTA, 3 SBIS PINA, 4 CBI PORTA, 3
10000 Word limit

(8%)

Write a function that multiplies 2 8-bit numbers and returns the result as a 16-bit number.

The function shall be implemented using the Via Calling Convention. Comment your code as you see fit.

--

```

LDI R16, 0xFF; initializing the stack (This is not necessary to get full mark.)
OUT SPL, R16 ; initializing the stack
LDI R16, 0x21; initializing the stack
OUT SPH, R16 ; initializing the stack

ldi r26, 1 ; Original values of the working register (Should be the same after the function
has been executed) (This is also not necessary to get full mark.)
ldi r27, 2
ldi r18, 3
ldi r19, 4

push r16 ; 1. r16 does not matter. This is for allocating place to the output value

ldi r16, 10
ldi r17, 100
push r16 ; 1. Call setup
push r17 ; 1. Call setup

call AdderFunc ; 2. Call site
pop r17 ; 9. popping input values.
pop r16 ; 9. popping input values.
pop r17 ; 9. Retrieving output value.
stayhere:
rjmp stayhere

AdderFunc:
push r26 ; 3. saving working registers
push r27 ; 3. saving working registers
push r18 ; 3. saving working registers
push r19 ; 3. saving working registers

in r26, SPL ; 4. (Retrieving input values). Setting up the X-register
in r27, SPH ;
adiw r26, 10 ; 4 pushes from working registers, 3 from return address, 2 inputs, and 1 extra.
LD R18, -X ; 4. retrieving input value r16 = 10
LD R19, -X ; 4. retrieving input value r17 = 100

ADD r18, r19 ; 5. implementing the function body
adiw r26, 3 ; 5. updating the x-pointer to point at the right address.
st -X, r18 ; 6. Saving output value

pop r19; 7. restoring working registers
pop r18; 7. restoring working registers
pop r27; 7. restoring working registers
pop r26; 7. restoring working registers

ret ; 8. return from the function

```

Floating Point

The table below summarizes the IEEE 754 encoding for single-precision floating point values:

Single-Precision	Exponent = 8	Fraction = 23	Value
Normalized Number	1 to 254	Anything	$\pm(1.F)_2 \times 2^{E-127}$
Denormalized Number	0	nonzero	$\pm(0.F)_2 \times 2^{-126}$
Zero	0	0	± 0
Infinity	255	0	$\pm \infty$
NaN	255	nonzero	NaN

(4%)

A single precision floating point register contains the following bits:

11000000111000000000000000000000

What is the decimal value in the register?

-7

$$-(2^{(129-127)}) \times (1+0.5+0.25) = -7$$

(4%)

A single precision floating point register contains the following bits:

00111101110000000000000000000000

What is the decimal value in the register?

0.09375

$$-(2^{(123-127)}) \times (1+0.5) = 0.09375$$