Boolean Algebra

16. juni 2022 14:17

A*A=a
A*!A=0
A*1=A
A*0=0

A+A=A
A+!A=1
A+1=1
A+0=A

A+!A*B=A+B
!(A+B) = !A*!B
!(A*B)=!A+!B
!A+A*!B = !A+!B

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:

10010010 $_2$

(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:

0101001000 $_2$

Next

01101110

1.com: 10010001

2. com: 10010010

2B8=1010111000

1. com: 0101000111
2. com: 0101001000

Com1 -> invers of the const (binary/hex converted to binary/dec converted to binary)

Com2-> Com1 + 1

ALU

18. marts 2022 10:49

2.1 TASK: Fill out Table 1

Instruction nr.	Assembly	Binary code	Explanation
1	LDI R17, 101 ;	1110 0110 0001 0101	
2	LDI R18, 153 ;	1110 1001 0010 1001	
3	ADD R17, R18 ;	0000 1111 0001 0010	

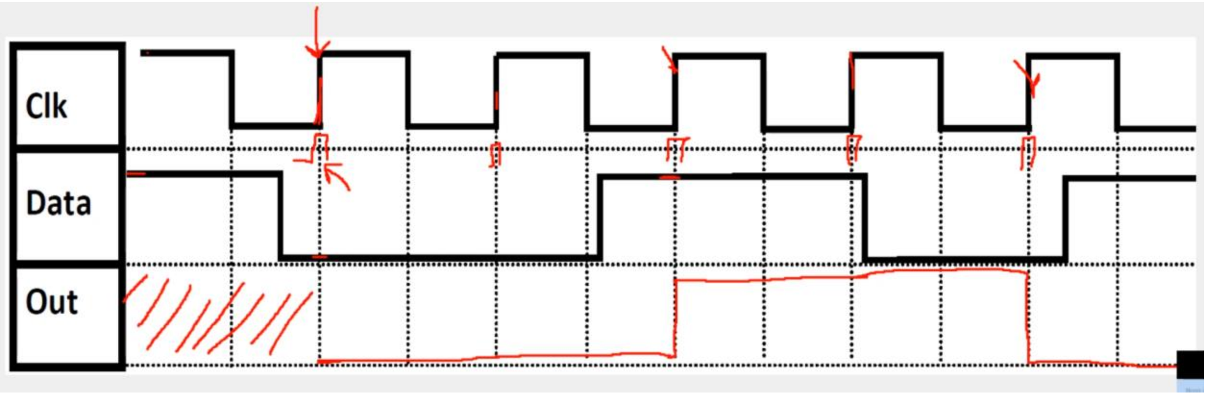
Table 1

1101	KKKK	dddd	KKKK

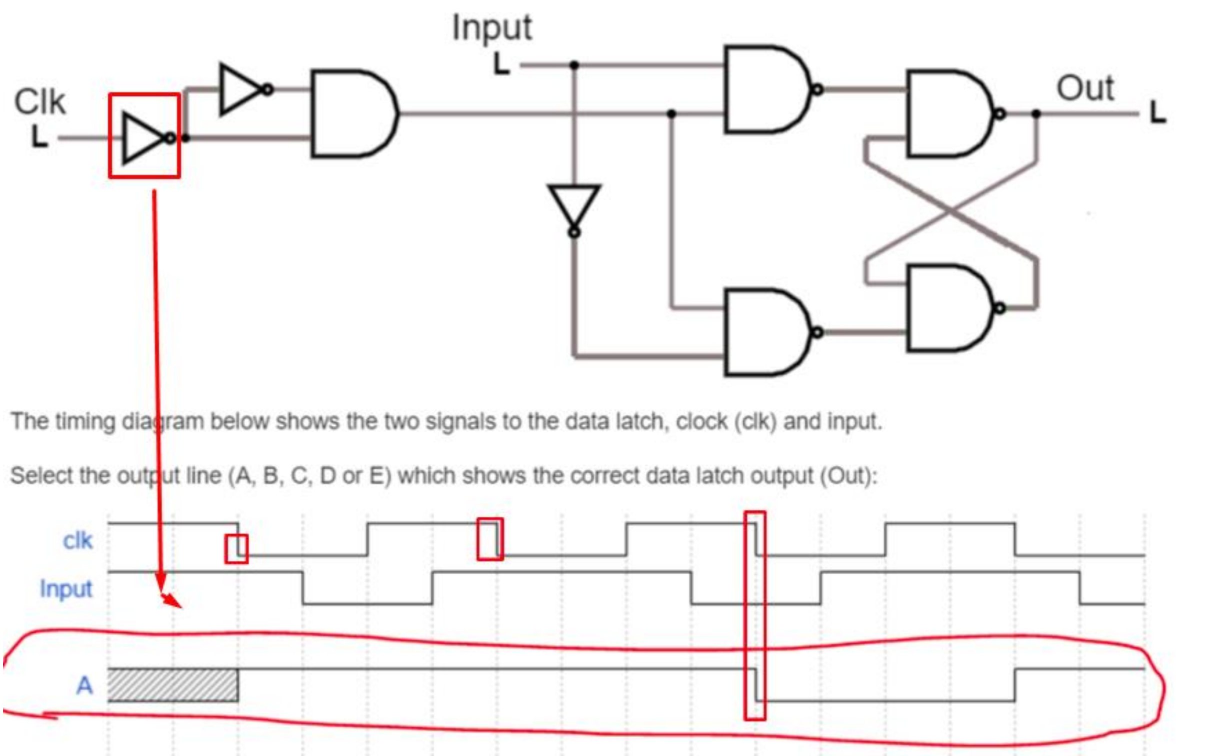
LDI = 1101 - get from manual stuff
ADD = 0000
KKKKKKKK is constant split in two parts => ex: 101 => 110 0101 => 0110 ...ddddd....0101
R17 => 10001 => dddd=0001

Clock

^^WHEN clock is going from 0->1



^^WHEN Clock is going from 1->0 because of NOT gate



Asambly

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

=> convert R16 into binary
=> ANDI will be 01111111 (8bit presentation of 7F) AND(*) 0000110
=> ORI 00000110 (response from ANDI) OR (+) 00100010
=> 00100110 (response from ORI) - 1 = 00100101=>37 decimal

Clock cycle

```
LDI R18, 10      1
LOOP2:
NOP              1 - 1
LDI R20, 25      1
LOOP1:
NOP 1            - 1
NOP 2 1          - 1
DEC R20 1        - 1
BRNE LOOP1 2/1   - 2
NOP 1            - 1
DEC R18 1        - 1
BRNE LOOP2 2     - 2
```

loop1 = 5·25

$$loop2 = 10 \cdot (\overset{125}{loop1} + 5) = 10 \cdot 130 = 1300$$

I. R18 is part of LOOP 2 because after break nothing happen
II. R25 is part of LOOP 1 because after break nothing happen with R25

->LOOP 1 = 5 cycle * 25 times from R25 => 125 cycles
->LOOP 2 = 10 times * (125+5 cycle) => 10*130 cycle = 1300

Floating point IEEE 754

16. juni 2022 19:00

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

EX

1	10000001	11000000000000000000000
S +- (1 is -) (0 is +)	E (from binary to decimal)	F (1.f1f2...fn) in binary => 1.11

=> -(2^(129-127))*(1.75)=-7 | where 129 is E |F = 1.11 base 2 and 1.75 base 10 | S = (-1)^S
where (-1)*1 = -1

Mandatory Assignment

Table of content

1 Initialize port A as an output. Send 0xAA to port A

Solution

```
Ldi r16, 0xff
Out ddra, r16
Ldi r16 0xaa
Out porta, r16
```

2 Set bit 4 in ddra with out disturbing the remaining bits.

Solution

```
Sbi ddra, 4
```

3 Clear bit 3 in ddrb without disturbing the remaining bits.

Solution

```
Cbi ddrb, 3
```

4 Set bit 1,3,5 and 7 in DDRA without disturbing the remaining bits.

Solution

```
in r16, ddra
ori r16, 0b10101010
out ddra, r16
```

5 Clear bit 0, 1, 2 and 3 in DDRA without disturbing the remaining bits.

Solution

```
in r16, ddra
andi r16, 0b11110000
out ddra, r16
```

6 A switch is placed on PA3. If it is set then multiply r16 and r17 and send the result to portb (most significant byte) and portc (least significant byte). If PA3 is cleared add r16 and r17 and send the result to portb.

Solution

```
cbi ddra, 3 ; clear bit in ddra register to ensure that the switch is an input.
ldi r18, 0xff
out ddrb, r18 ; make this port an output.
out ddrc, r18 ; make this port an output.

sbic pina, 3 ; skip next instruction the bit is 0
rjmp bitcleared
add r16, r17
out portb, r16
rjmp forever

bitcleared:
mul r16, r17
out portb, r1
out portc, r0

forever:
rjmp forever ; stay here
```

7 Make a multiplier function using VIA calling convention

Solution

```
LDI R16, 0xFF; initializing the stack
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)
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
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 multiplierFunc ; 2. Call site
pop r17 ; 9. poping input values.
pop r16 ; 9.  poping input values.
pop r17 ; 9. Retrieving output value.
pop r16 ; 9. Retrieving output value.
stayhere:
rjmp stayhere

multiplierFunc:
push r0  ; 3. saving working registers
push r1  ; 3. saving working registers
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. (Retrienving input values). Setting op the X-register
in r27, SPH ;
adiw r26, 12 ; 6 pushes from working registers, 3 from return adress, 2 inputs,
and 1 ekstra.
LD R18, -X  ; 4. retrieving input value r16 = 10
LD R19, -X  ; 4. retrieving input value r17 = 100

mul r18, r19 ; 5. implementing the function body
adiw r26, 4  ; 5. updating the x-pointer to point at the right adress.
st -X,r0 ; 6. Saving output value 1
st -X,r1 ; 6. Saving output value 2

pop r19; 7. restoring working registers
pop r18; 7. restoring working registers
pop r27; 7. restoring working registers
pop r26; 7. restoring working registers
pop r1 ; 7. restoring working registers
pop r0 ; 7. restoring working registers

ret    ; 8. return from the function
```

Mandatory Assignment

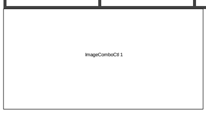
Table of content

1 From Boolean expression to truth table

1.1 Fill out the truth table from the following Boolean expression:

$Out = A\bar{B}\,\bar{C} + BC$

A	B	C	Out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



1.2 Fill out the truth table from the following Boolean expression:

$Out = \bar{B}$

A	B	C	Out
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1

1	0	1	1
1	1	0	0
1	1	1	0

1.3 Fill out the truth table from the following Boolean expression:

$Out = 1$

A	B	C	Out
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

1.4 Fill out the truth table from the following Boolean expression:

$Out = C(A + B)$

A	B	C	Out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0

1	0	1	1
1	1	0	0
1	1	1	1

1.5 Write down the truthtable from the following Boolean expression.

$Out = B(A + C) + \bar{B}\,\bar{C}\,\bar{A}$

Solution

A	B	C	Out
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

2 From truth table to Boolean expression

2.1 Write down the Boolean expression described by the truthtable and simplify it as much as possible.

A	B	C	Out
0	0	0	0
0	0	1	0

0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

SOLUTION

$$\bar{A}BC + A\bar{B}C + ABC$$

$$\bar{A}BC + AC(B + \bar{B})$$

$$\bar{A}BC + AC$$

$$C(\bar{A}B + A)$$

$$C(B + A)$$

$$CB + CA$$

2.2 Write down the Boolean expression described by the truthtable and simplify it as much as possible.

A	B	C	Out
0	0	0	1

0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

SOLUTION

$$\bar{a}\bar{b}\bar{c} + \bar{a}bc + a\bar{b}\bar{c} + abc$$

$$\bar{b}\bar{c}(a + \bar{a}) + bc(\bar{a} + a$$

$$\bar{b}\bar{c} + bc$$

2.3 Write down the Boolean expression described by the truthtable and simplify it as much as possible.

A	B	C	Out
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1

1	1	1	0

Solution

$$\bar{a}\bar{b}c + \bar{a}b\bar{c} + ab\bar{c}$$

$$\bar{a}\bar{b}c + b\bar{c}(\bar{a} + a)$$

$$\bar{a}\bar{b}c + b\bar{c}$$

2.4 Write down the Boolean expression described by the truthtable and simplify it as much as possible.

A	B	C	Out
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Solution

$$\bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + a\bar{b}\bar{c} + a\bar{b}c + ab\bar{c}$$
$$\bar{a}\bar{b}(c + \bar{c}) + a\bar{b}(\bar{c} + c) + ab\bar{c}$$
$$\bar{a}\bar{b} + a\bar{b} + ab\bar{c}$$
$$\bar{a}\bar{b} + a(\bar{b} + b\bar{c})$$
$$\bar{a}\bar{b} + a(\bar{b} + \bar{c})$$
$$\bar{a}\bar{b} + a\bar{b} + a\bar{c}$$
$$\bar{b}(\bar{a} + a) + a\bar{c}$$
$$\bar{b} + a\bar{c}$$

2.5 Write down the Boolean expression described by the truthtable and simplify it as much as possible.

A	B	C	Out
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Solution

$$\bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}b\bar{c} + \bar{a}bc + a\bar{b}\bar{c} + a\bar{b}c + ab\bar{c}$$

$$\bar{a}\bar{b}(c + \bar{c}) + \bar{a}b(\bar{c} + c) + a\bar{b}(\bar{c} + c) + ab\bar{c}$$

$$\bar{a}\bar{b} + \bar{a}b + a\bar{b} + ab\bar{c}$$

$$\bar{a}(b + \bar{b}) + a(\bar{b} + b\bar{c})$$

$$\bar{a} + a(\bar{b} + \bar{c})$$

$$\bar{a} + a\bar{b} + a\bar{c}$$

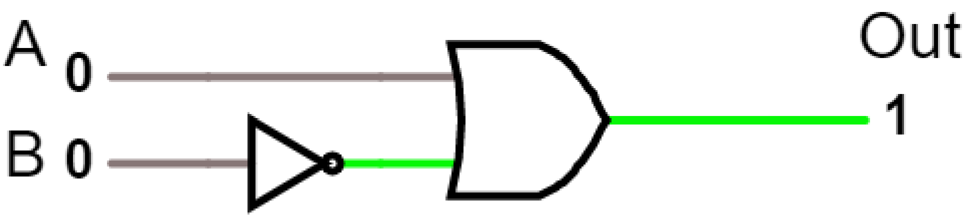
$$\bar{a} + \bar{b} + \bar{c}$$

3 Boolean expression to circuit.

3.1 Draw the logic circuit described by the following Boolean expression

$Out = A + \bar{B}$

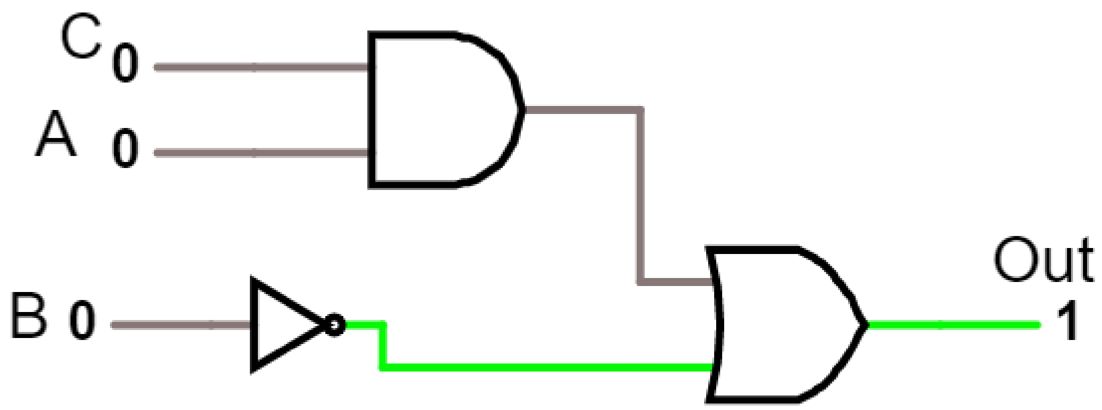
Solution



3.2 Draw the logic circuit described by the following Boolean expression

$Out = CA + \bar{B}$

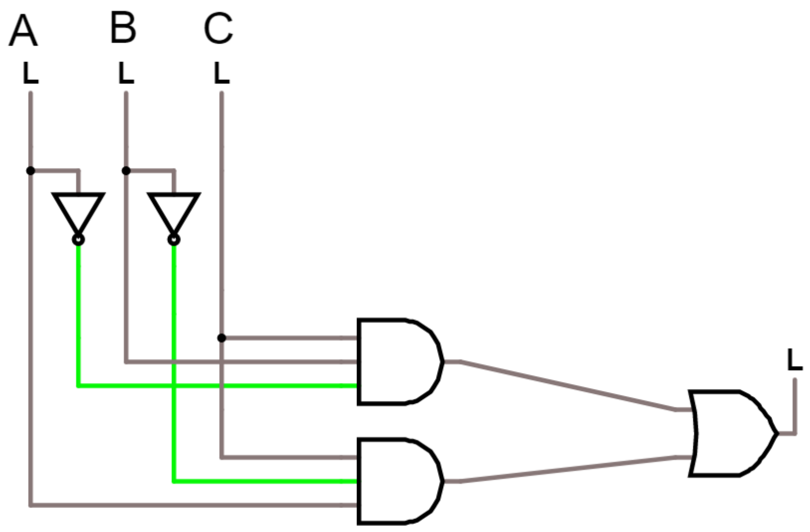
Solution



3.3 Draw the logic circuit described by the following Boolean expression

$Out = A\bar{B}C + \bar{A}BC$

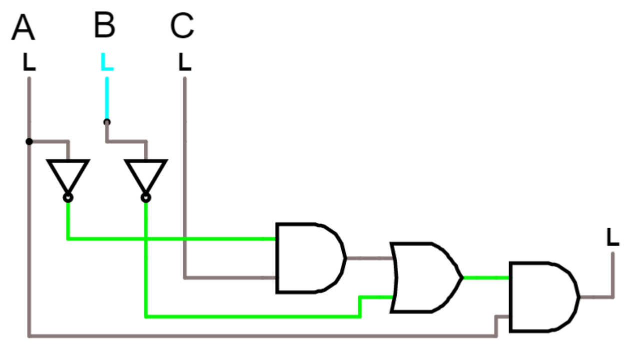
Solution



3.4 Draw the logic circuit described by the following Boolean expression

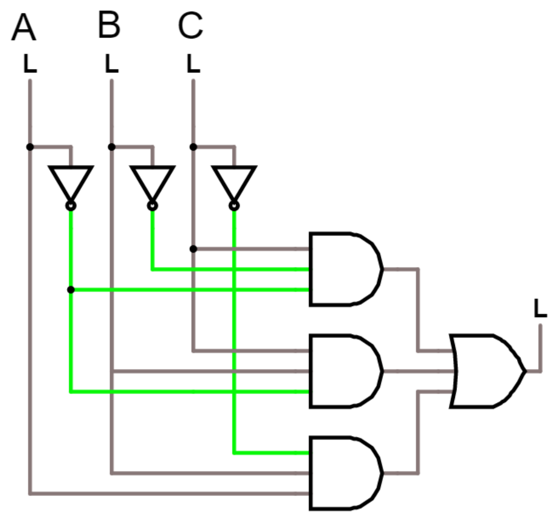
$Out = A(\bar{B} + \bar{A}\,C)$

Solution



4 From circuit to Boolean expression

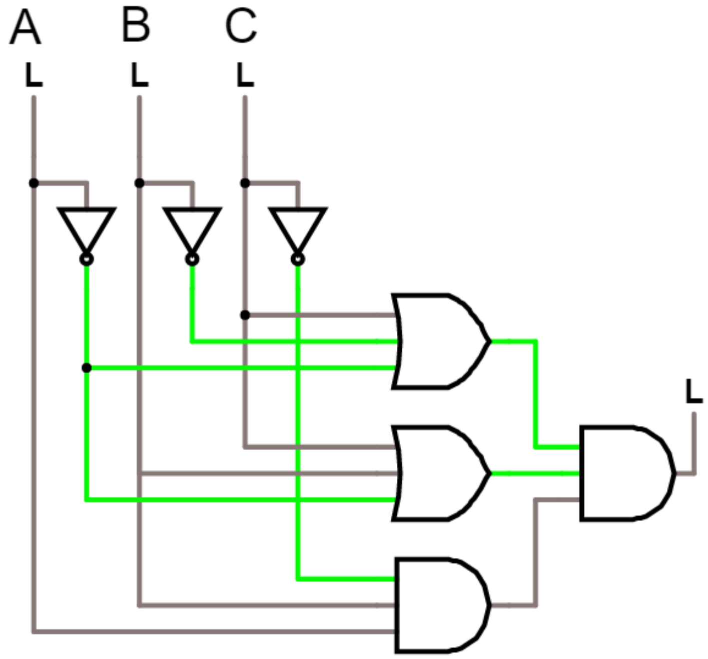
4.1 Write down the Boolean expression derived from the logical circuit below:



Solution

$\overline{A}\,\overline{B}C + \overline{A}BC + AB\overline{C}$

4.2 Write down the Boolean expression derived from the logical circuit below:

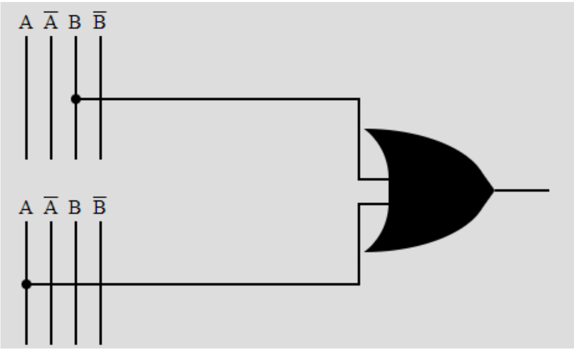


Solution

$(\bar{A} + \bar{B} + C)(\bar{A} + B + C)(AB\bar{C})$

5 From circuit to truth table

5.1 Write down the truthtable from the following circuit:

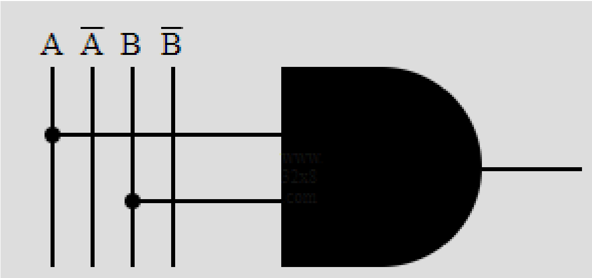


A	B	Out
0	0	
0	1	
1	0	
1	1	

Solution

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

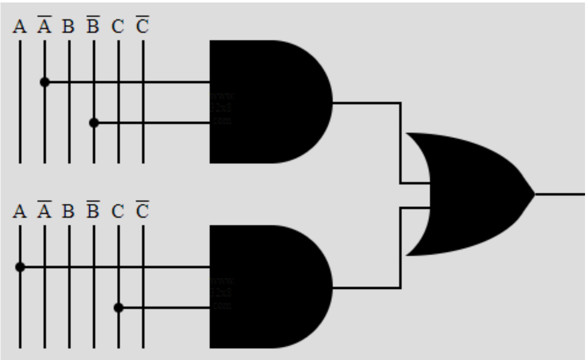
5.2 Write down the truthtable from the following circuit:



Solution

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

5.3 Write down the truthtable from the following circuit:

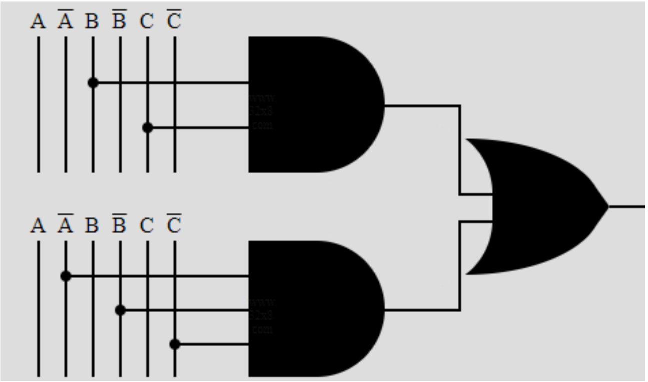


A	B	C	Out
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Solution

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

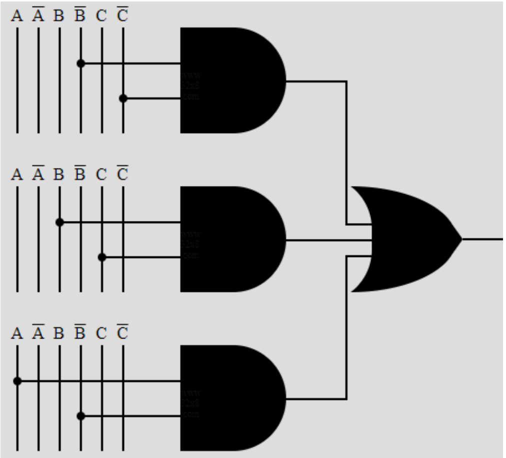
5.4 Write down the truthtable from the following circuit:



Solution

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

5.5 Write down the truthtable from the following circuit:



Solution

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Mandatory Assignment

Table of content

1 Calculate the delays

1.1 Calculate the delay. The clock frequency is 16MHz

```
delay:
ldi r20, 86
loop1:
dec r20
brne loop1
```

Solution

$$1+86\cdot 3-1=258 \qquad \frac{258}{16\ MHz}=16.125\ \mu s$$

1.2 Calculate the delay. The clock frequency is 16MHz

```
delay:
ldi r20, 100
loop1:
nop
dec r20
brne loop1
nop
nop
```

Solution

$$1+100\cdot 4-1+2=402 \qquad \frac{402}{16\ MHz}=25.125\ \mu s$$

1.3 Calculate the delay. The clock frequency is 16MHz

```
delay:
```

```
ldi r20, 200
loop1:
nop
dec r20
nop
nop
brne loop1
nop
nop
```

Solution

$$1+200\cdot 6-1+2=1202 \qquad \frac{1202}{16\ MHz}=75.125\ \mu s$$

1.4 Calculate the delay. The clock frequency is 16MHz

```
delay:
ldi r18, 180
loop2:
ldi r20, 199
loop1:
dec r20
brne loop1
dec r18
brne loop2
```

Solution

$$loop1:=1+3\cdot 199-1=597$$

$$delay:=(597+3)\cdot 180=108000 \qquad \frac{108000}{16\ MHz}=6.75\ ms$$

1.5 Calculate the delay. The clock frequency is 16MHz

```
delay:
ldi r18, 11
loop2:
nop
ldi r20, 15
```

```
loop1:
nop
nop
dec r20
nop
brne loop1
nop
nop
dec r18
brne loop2
nop
nop
nop
```

Solution

$$loop1 := 1 + 6 \cdot 15 - 1 = 90$$

$$delay := (90 + 6) \cdot 11 + 3 = 1059 \qquad \frac{1059}{16\ MHz} = 66.188\ \mu s$$

1.6 Calculate the delay. The clock frequency is 16MHz

```
delay:
ldi r16, 14
loop3:
ldi r18, 11
loop2:
ldi r20, 15
loop1:
dec r20
brne loop1
dec r18
brne loop2
dec r16
brne loop3
```

Solution

$$loop1 := 3 \cdot 15 = 45$$

$$loop2 := (45 + 3) \cdot 11 = 528$$

$$delay := (528 + 3) \cdot 14 = 7434 \qquad \frac{7434}{16\ MHz} = 464.625\ \mu s$$

1.7 Calculate the delay. The clock frequency is 16MHz

```
delay:
ldi r16, 14
loop3:
nop
ldi r18, 110
loop2:
nop
ldi r20, 150
loop1:
dec r20
brne loop1
nop
nop
dec r18
brne loop2
dec r16
nop
brne loop3
nop
nop
nop
```

Solution

$$loop1 := 3 \cdot 150 = 450$$

$$loop2 := (450 + 6) \cdot 110 = 50160$$

$$delay := (50160 + 5) \cdot 14 + 3 = 702313 \qquad \frac{702313}{16\ MHz} = 43.895\ ms$$

2 Create delays

2.1 Your microcontroller is connected to a 16MHz clock. Create a delay that is around $10\mu s$ (+- 5%):

Solution

$$clocks := 10\ \mu s \cdot 16\ MHz = 160$$

You could choose a loop of 5 clocks and run it 32 times ($32 * 5 = 160$)

```
delay:
ldi r16, 32
loop1:
nop
nop
dec r16
brne loop1
```

2.2 Your microcontroller is connected to a 16MHz clock. Create a delay that is around $168\mu s$ (+- 5%):

Solution

$$clocks := 168\ \mu s \cdot 16\ MHz = 2688$$

If you divide this number with 16, you get a round number: $\frac{2688}{16} = 168$

So make a loop that take 16 clocks and run it 168 times:

```
delay:
ldi r16, 168
loop1:
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
```

```
nop
nop
nop
dec r16
brne loop1
```

2.3 Your microcontroller is connected to a 16MHz clock. Create a delay that is around $1ms$ (+- 5%):

Solution

$$clocks := 1 \cdot ms \cdot 16 \cdot MHz = 16000$$

One way of doing this is to create an innerloop that takes 495 clocks. Then an outer loop that add 5 clocks, and then runs 32 times.

$$(495 + 5) * 32 = 16000$$

```
delay:
ldi r17, 32
loop2:
ldi r16, 99
loop1:
nop
nop
dec r16
brne loop1
nop
nop
dec r17
brne loop2
```

1 Calculate the 2. Compliment of the following numbers:

- 0b 01101110

  10010010
- 0b 10111001

  1000111
- 0b 1111 1111

  00000001

2 Fill out the table below:

Binairy number	HEX	Decimal
0b110110	0x36	54
0b 1111 0000	0xF0	240
0b11111	0x1F	31
0x11001000	0xC8	200
0b1010 1010	0xAA	170
0b1010 1010	0x AA	170

3 Calculate the following:

$0xFF - 0b11010000 - \$2F \ = 0$

Mandatory Assignment

Table of content

1 1-bit register triggering.

Figure 1, Figure 2, Figure 3 and Figure 4 shows, 1-bit registers. Fill out the table below:

Trigger	
High level	Figure XX
Low level	Figure XX
Rising edge	Figure XX
Falling edge	Figure XX

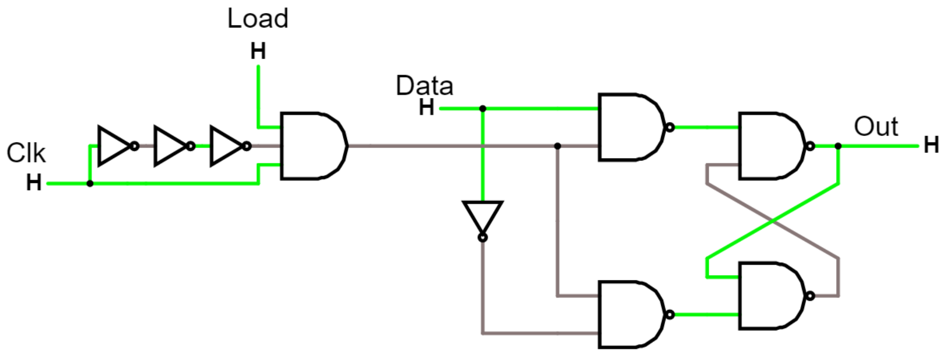


Figure 1: 1-bit register. [Simulation]

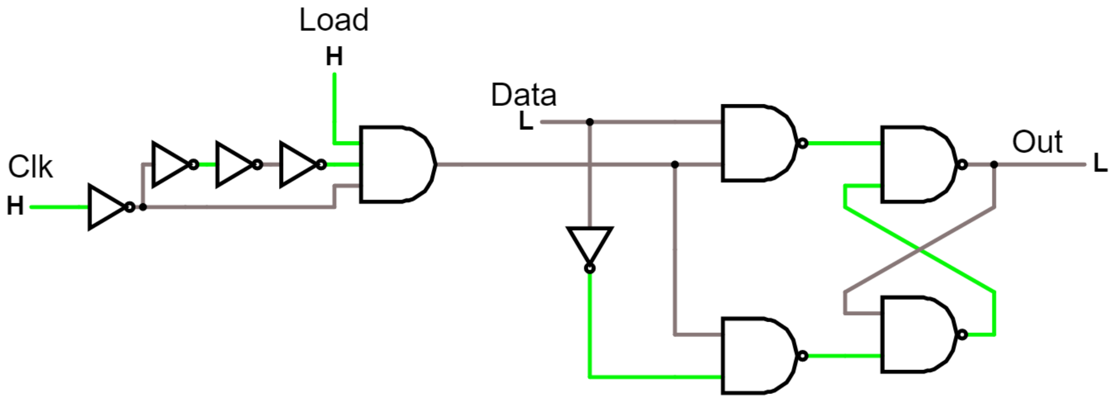


Figure 2: 1-bit register. [Simulation]

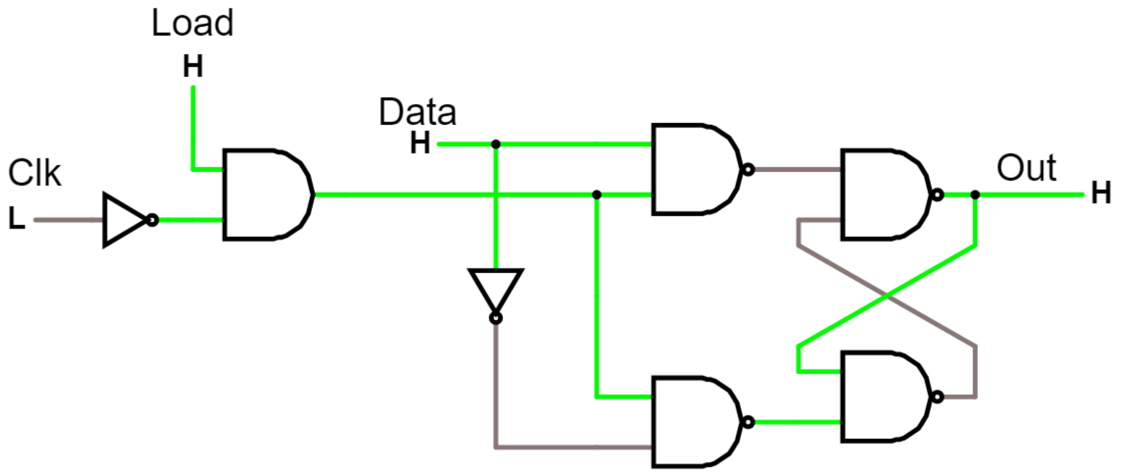


Figure 3: 1-bit register. Simulation

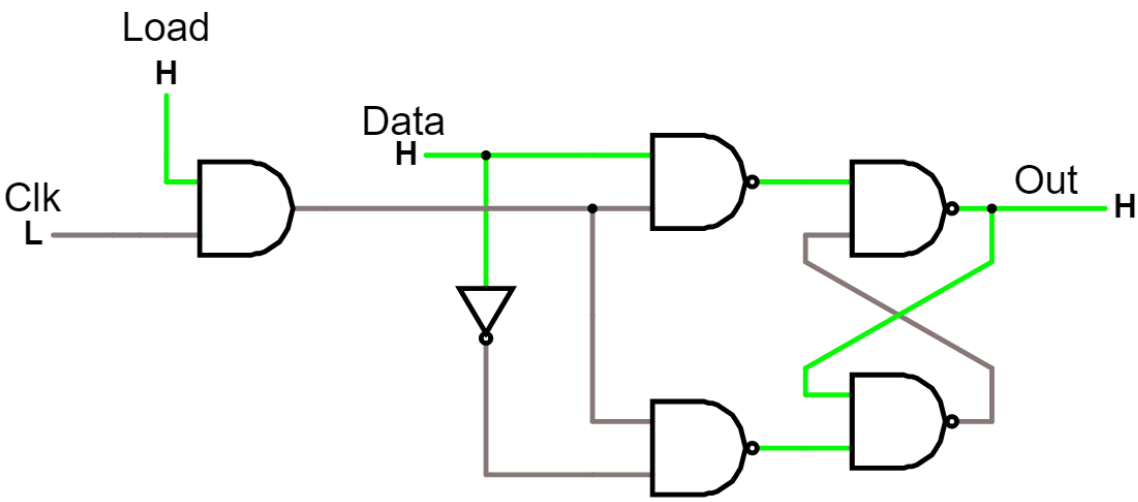


Figure 4: 1-bit register. Simulation

Solution

Trigger	
High level	Figure 4
Low level	Figure 3
Rising edge	Figure 1
Falling edge	Figure 2

2 Latches

2.1 How are the latch on Figure 9 triggered?

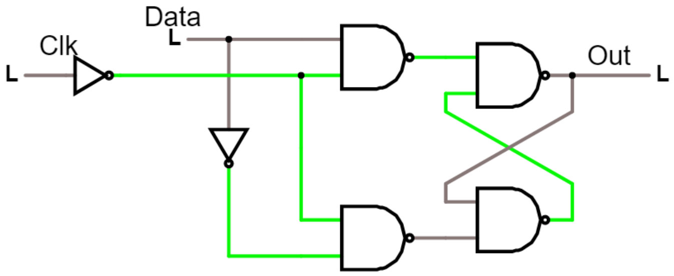


Figure 5 [Simulation]

Solution

Low level triggered

2.1.1 Draw 'Out' on Figure 6

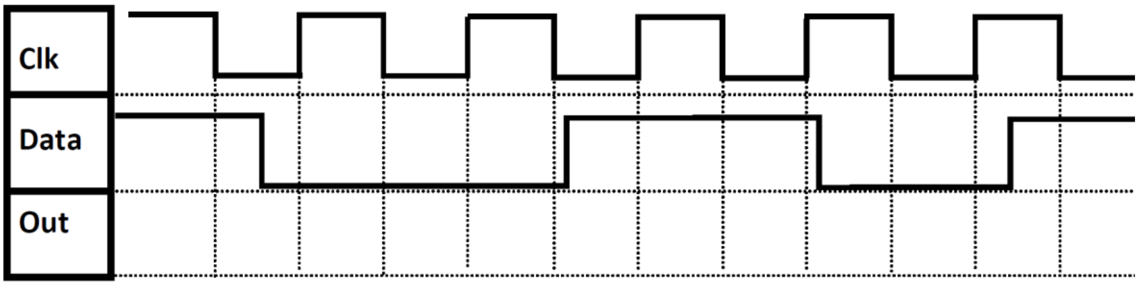
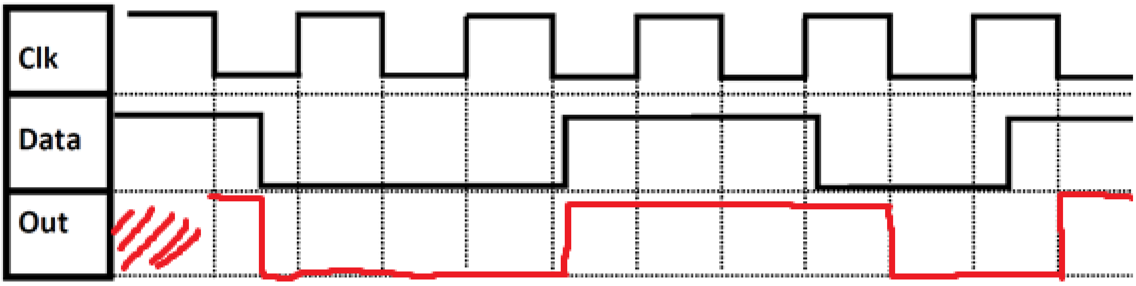


Figure 6

Solution



2.2 How are the latch on Figure 7 triggered?

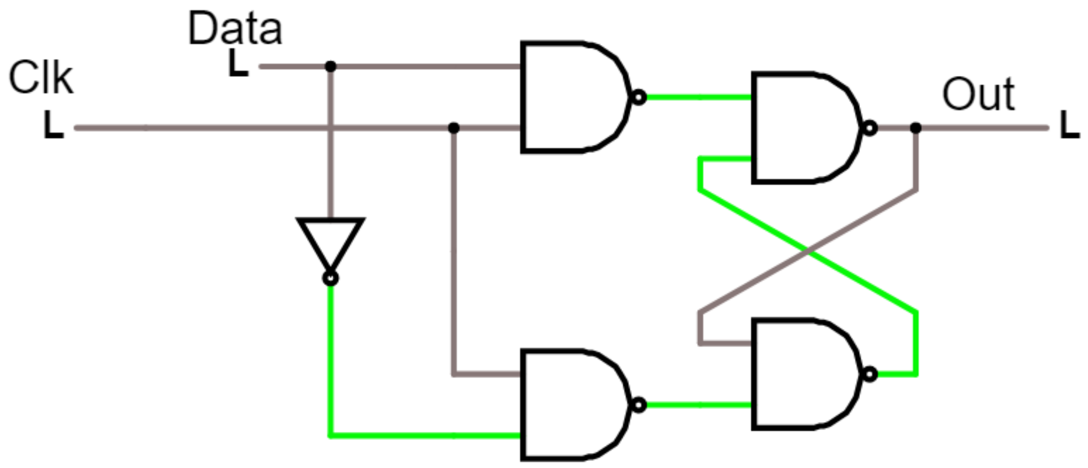


Figure 7. [*Simulation*]

Solution

High level triggered

2.2.1 Draw 'Out' on Figure 8

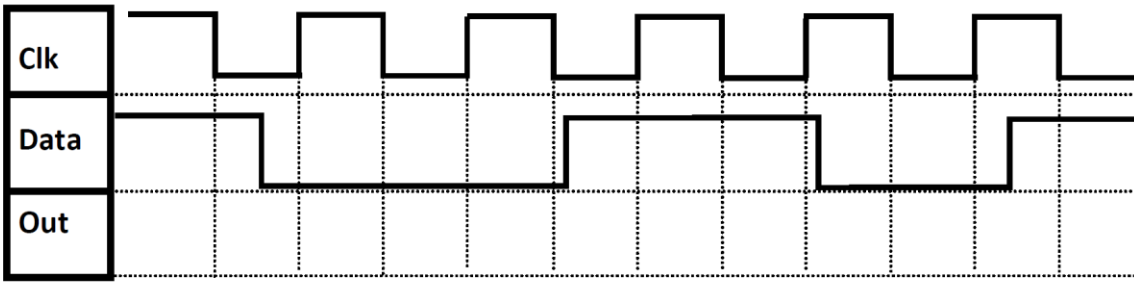
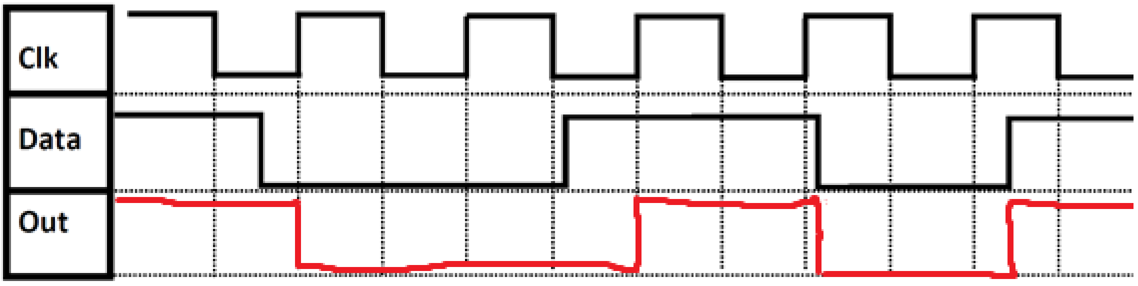


Figure 8

Solution



2.3 How are the latch on Figure 9 triggered?

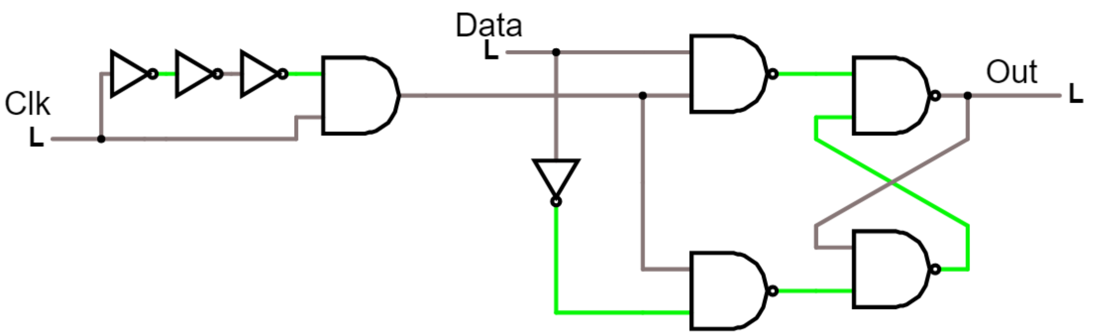


Figure 9: [Simulation]

Solution

Rising Edge

2.3.1 Draw 'Out' on Figure 10,

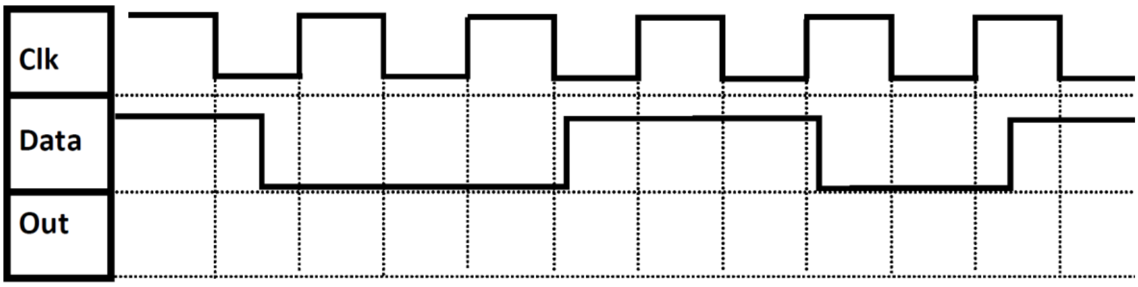
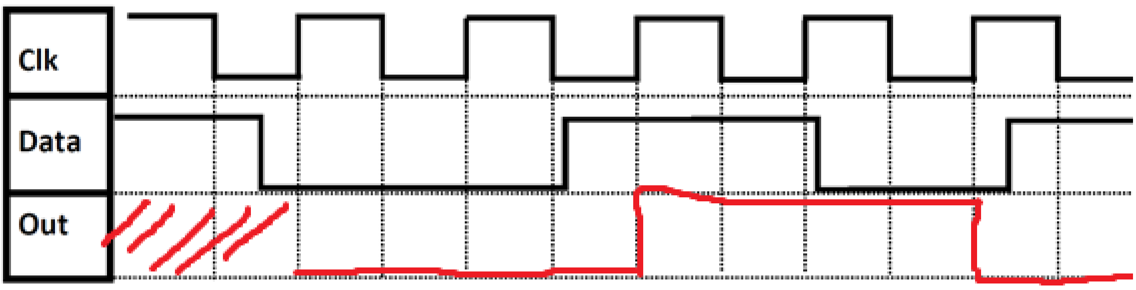


Figure 10

Solution



2.4 How are the latch on Figure 11 triggered?

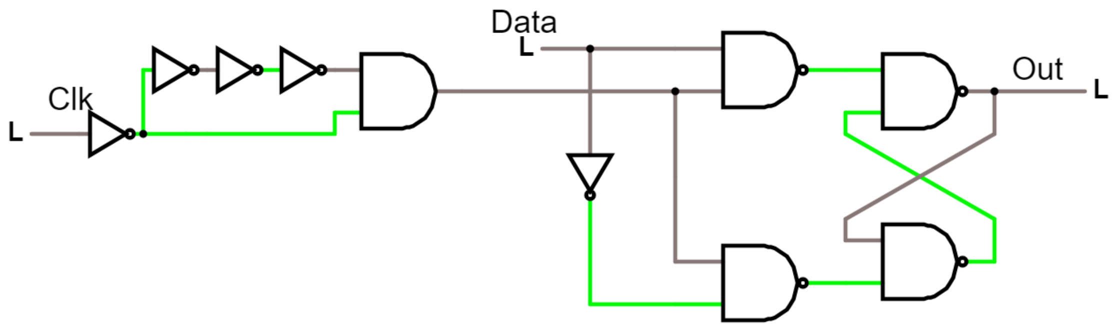


Figure 11: [Simulation]

Solution

Falling Edge

2.4.1 Draw 'Out' on Figure 12.

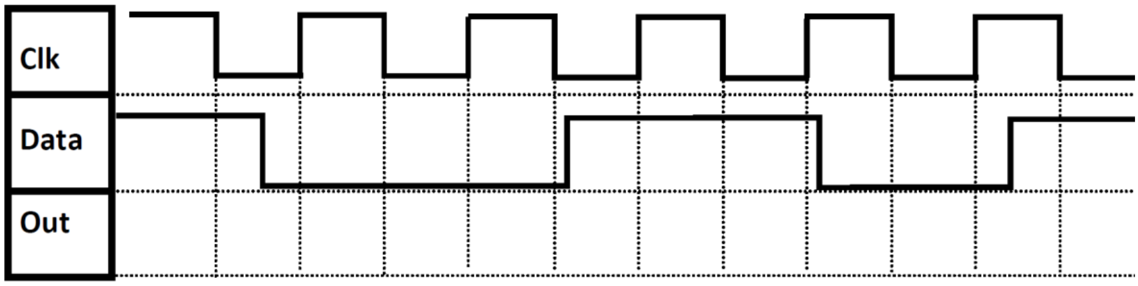
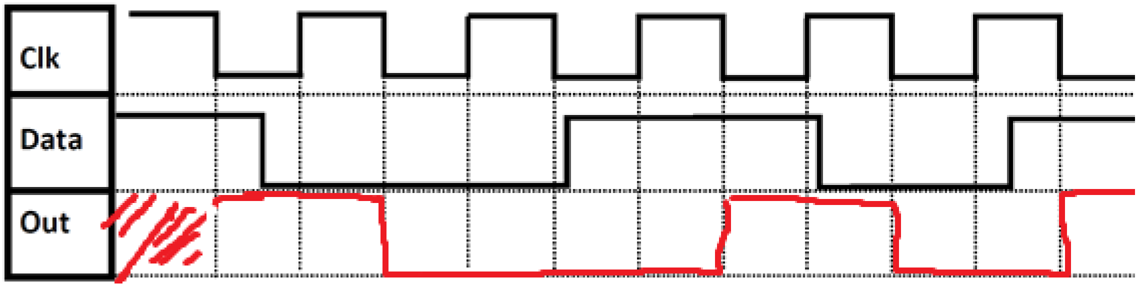


Figure 12

Solution



3 1-bit register

3.1 1-bit register High level triggered

A 1-bit register which is High-level triggered can be seen on Figure 13

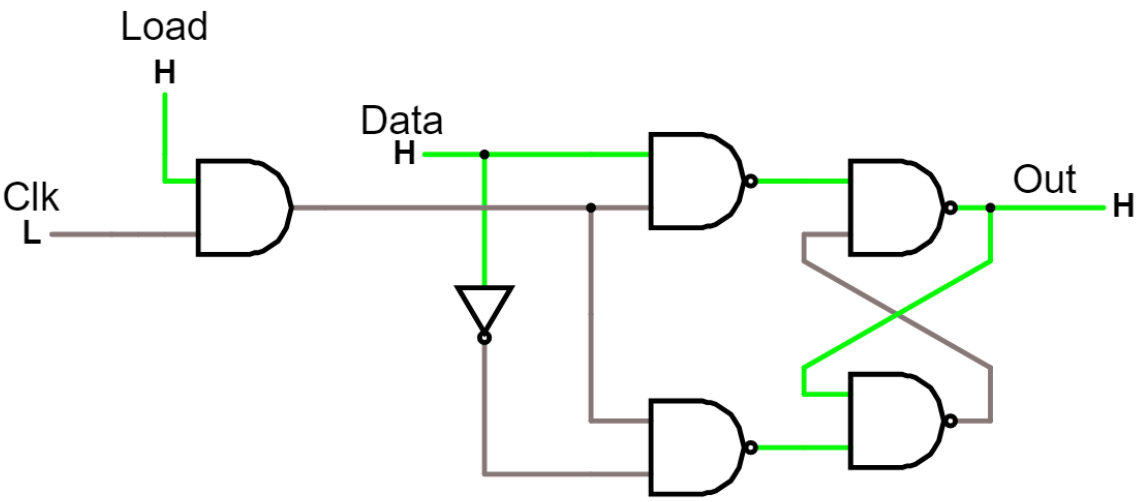


Figure 13: 1-bit register. [Simulation]

TASK: Draw 'Out' on Figure 14

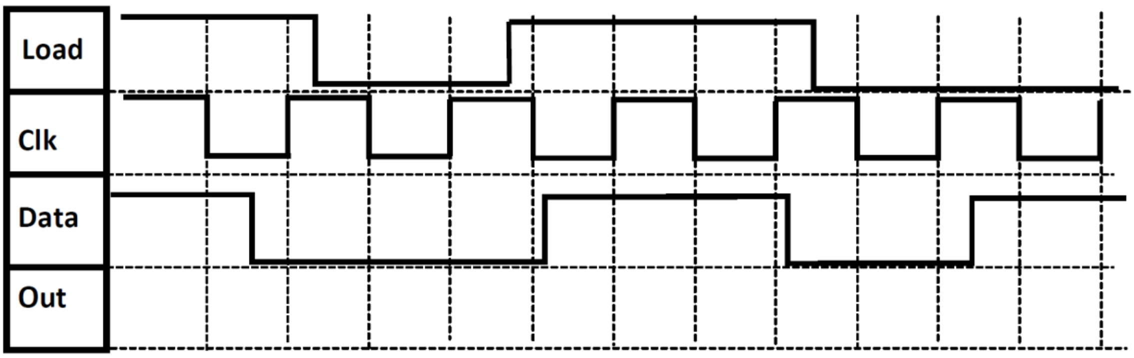
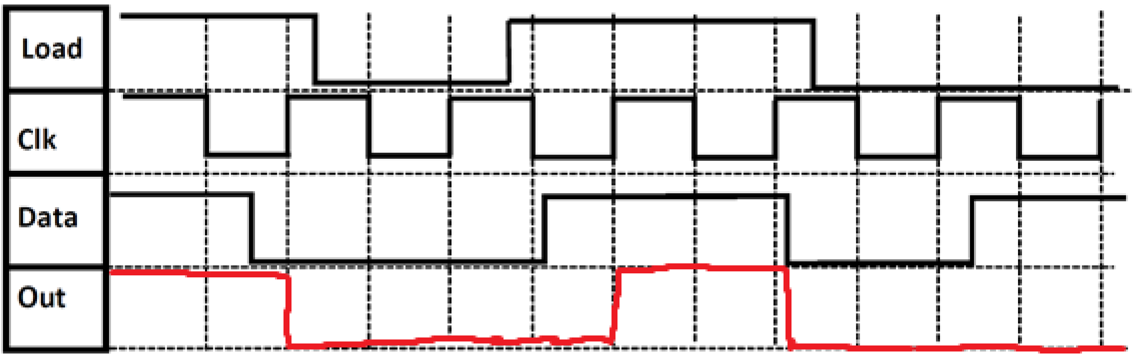


Figure 14

SOLUTION



3.2 1-bit register Low level triggered

A 1-bit register which is low-level triggered can be seen on Figure 15

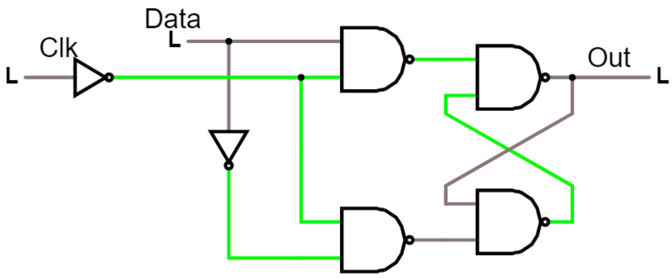


Figure 15 [Simulation]

TASK: Draw 'Out' on Figure 16

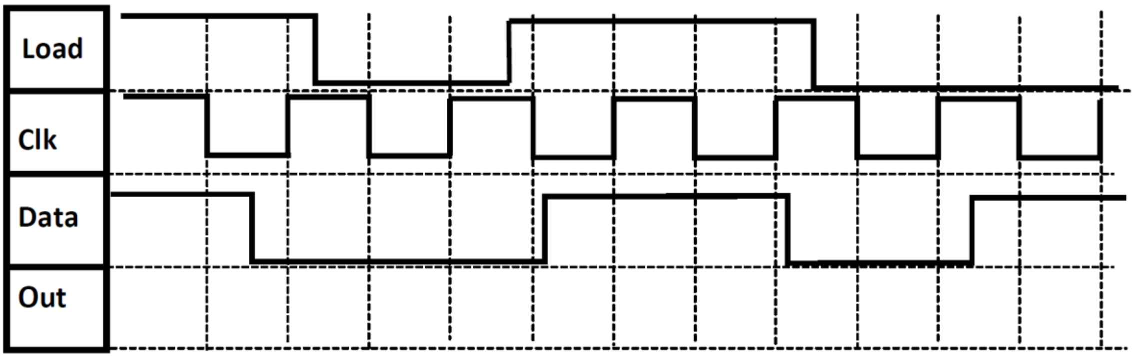
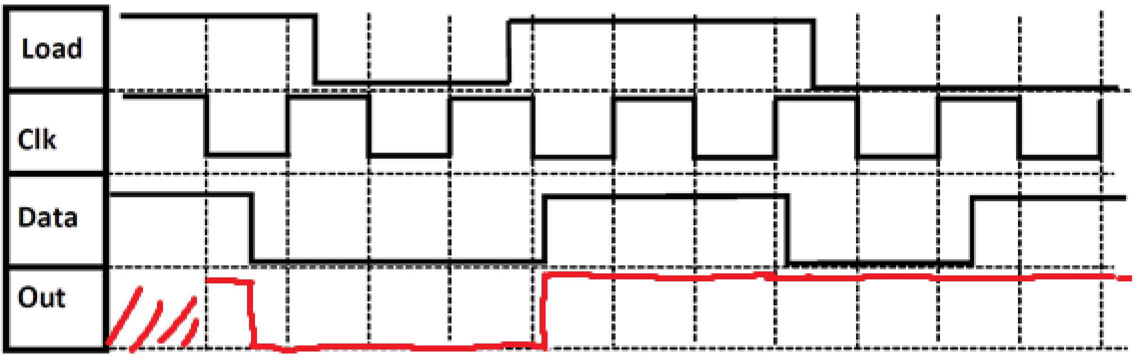


Figure 16

SOLUTION



3.3 1-bit register rising edge triggered

A 1-bit register which is rising edge triggered can be seen on Figure 17

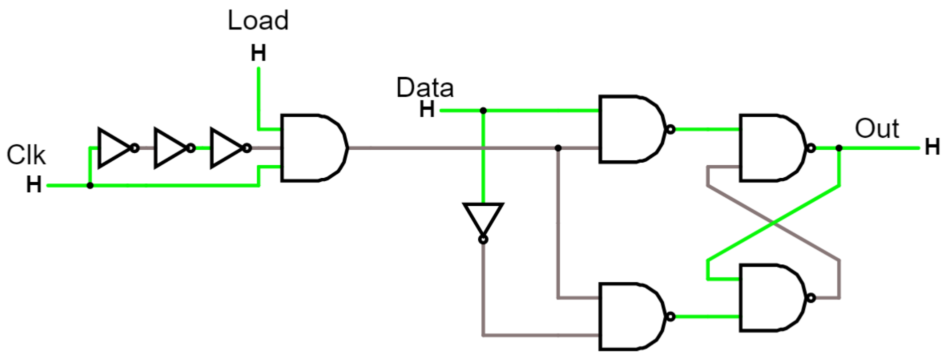


Figure 17: [Simulation]()

TASK: Draw 'Out' on Figure 18

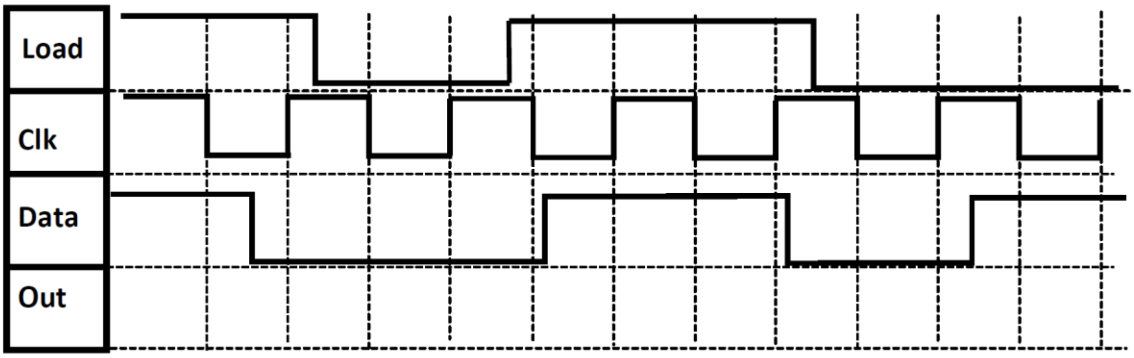
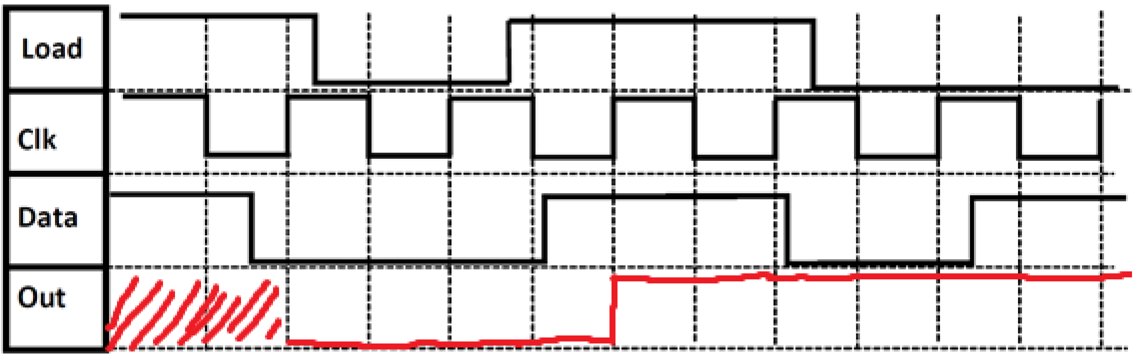


Figure 18

SOLUTION



3.4 1-bit register falling edge triggered

A 1-bit register which is falling edge triggered can be seen on Figure 19.

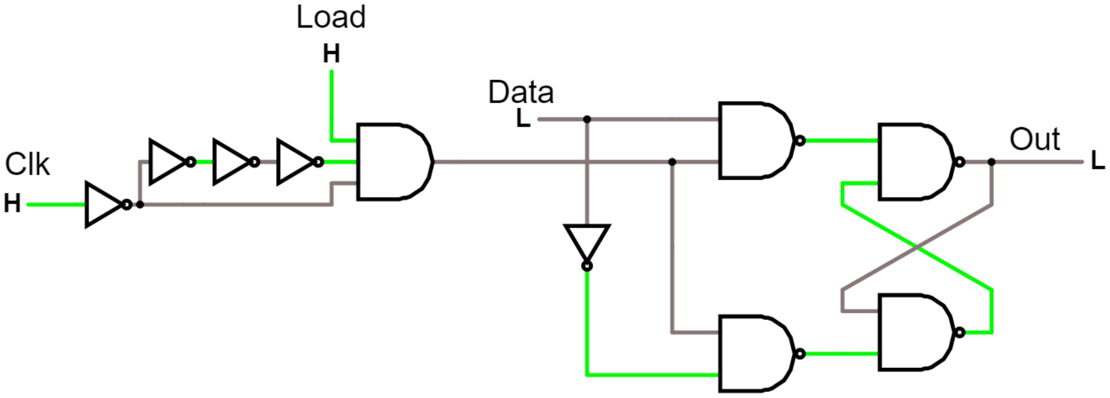


Figure 19: 1-bit register. [Simulation]

TASK: Draw ‘Out’ on Figure 20

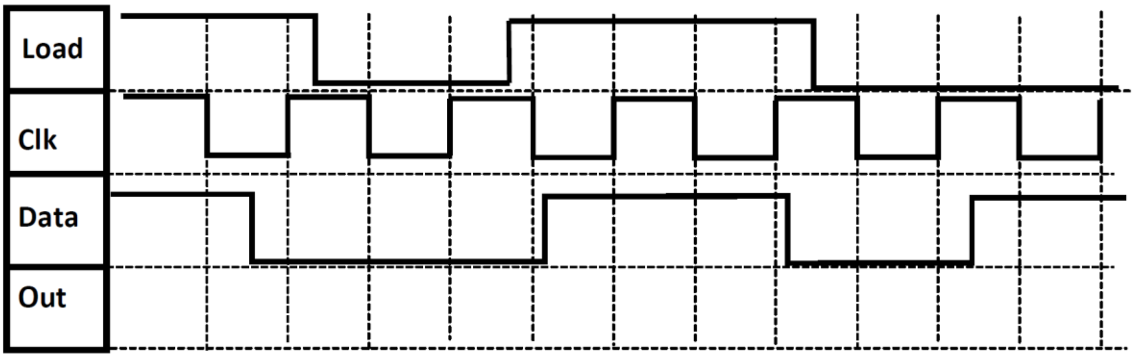
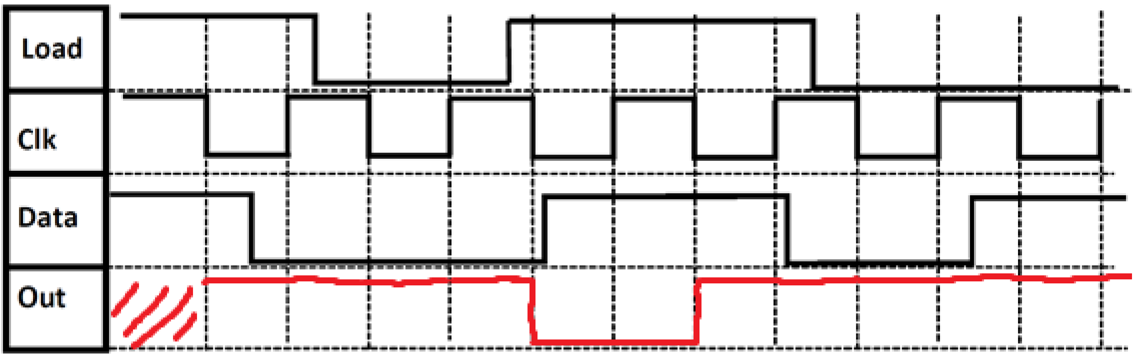


Figure 20

SOLUTION



Exercise 10 – Floating point

Table of content

1 Normalize the following decimal numbers

Denormalized	Normalized
1024	$1.024 * 10^3$
1000000	$1 * 10^6$
0.000898	$8.98 * 10^{-4}$
$0.5 * 10^{-3}$	$5 * 10^{-4}$
$0.5 * 10^3$	$5 * 10^2$
$131 * 10^{-3}$	$1.31 * 10^{-1}$
$131 * 10^3$	$1.31 * 10^5$
213.456	$2.13456 * 10^2$

2 Normalize the following binary numbers

(The exponent can be written in decimal)

Denormalized	Normalized
0.0011	$1.1 * 2^{-3}$
1100	$1.1 * 2^3$
111.0011	$1.110011 * 2^2$
$0.0011 * 2^3$	$1.1 * 2^0 = 1.1$
$0.0011 * 2^{-3}$	$1.1 * 2^{-6}$
$11001.01 * 2^3$	$1.100101 * 2^7$
$11001.01 * 2^{-3}$	$1.100101 * 2^1$

3 Write the following numbers as decimal number

3.1 A single precision Floating point register contains the following bits:

11000000011010000000000000000000000

What’s the decimal number in the register?

Solution

S (sign)	1=negetiv
Exponent=	10000000 = biased(128) 128-127=1
Fraction =	1.1101

$$Frac := 2^0 + 2^{-1} + 2^{-2} + 2^{-4} = 1.813$$

$$DecimalNumber := -Frac \cdot 2^1 = -3.625$$

3.2 A single precision Floating point register contains the following bits:

01001000010000000000000000000000

Whats the decimalnumber in the register?

Solution

S (sign)	0=positive
Exponent=	10010000 = biased(144) 144-127=17
Fraction =	1.1 = 1.5

$$frac := 2^0 + 2^{-1} = 1.5$$

$$DecimalNumber := frac \cdot 2^{17} = 196608$$

3.3 A single precision Floating point register contains the following bits:

01111111110100100000000000000000

Whats the decimalnumber in the register?

Solution

NaN (Not a Number)

3.4 A single precision Floating point register contains the following bits:

01111111100000000000000000000000

What's the decimalnumber in the register?

Solution

According to the standard when all Exponential bits are set and all the Fraction bits are cleared, the result are infinite.

3.5 A single precision Floating point register contains the following bits:

11111111100000000000000000000000

Whats the decimalnumber in the register?

Solution

-infinite.

3.6 A single precision Floating point register contains the following bits:

00001000010000000000000000000000

Whats the decimalnumber in the register?

Solution https://www.h-schmidt.net/FloatConverter/IEEE754.html

S (sign)	0=positive
Exponent=	00010000 = biased(16) 16-127=-111
Fraction =	1.1 = 1.5

$$frac := 2^0 + 2^{-1} = 1.5$$

$$DecimalNumber := frac \cdot 2^{-111} = 5.778 \cdot 10^{-34}$$

3.7 What is the biggest number (less than infinite) you can write in a single precision float register?

Solution

01111111011111111111111111111111

S (sign)	0=positive
Exponent=	11111110 = biased(254) 254-127=127
Fraction =	11111111111111111111111= almost 2

$$frac := 2 - 2^{-23} = 2$$

$$DecimalNumber := frac \cdot 2^{127} = 3.403 \cdot 10^{38}$$

3.8 What is the smallest number that are bigger than zero you can write in a single precision float register?

Solution

00000000000000000000000000000001

Since the Exponential only contains Zeros, the fraction is denormalized (meaning it does not assume a 1, in the beginning)

S (sign)	0=positive
Exponent=	00000000 = biased(0) -126 (and denormalized)
Fraction =	00000000000000000000001= 2^-23

$$Frac := 2^{-23} = 1.192 \cdot 10^{-7}$$

$$Decimal := Frac \cdot 2^{-126} = 1.401 \cdot 10^{-45}$$

4 In a single precision floating point register the exponential bits contains the bits 11111110. What is the resolution?

Solution

$$2^{-23} \cdot 2^{254-127} = 2.028 \cdot 10^{31}$$

5 Calculate 4096+0.5 using IEEE standard for single precision floating point.

Solution

	Fraction (binary)	Exponential (decimal)
4096	1.00000000000000000000000	12
0.5	1.00000000000000000000000	-1

Since the difference is 13

	Fraction (binary)	Exponential (decimal)
4096	1.00000000000000000000000	12
1	0.00000000000010000000000	12

Then they are added together

results	1.00000000000010000000000	12

$$\left(1+2^{-13}\right)\cdot 2^{12}=4096.5$$

6 Calculate 2^25+1 using IEEE standard for single precision floating point.

Solution

33554432=2^25

So all the fraction bits are zero.

	Fraction (binary)	Exponential (decimal)
33554432	1.00000000000000000000000	25
1	1.00000000000000000000000	0

1 needs to be bit-shiftet 24 times to the right:

	Fraction (binary)	Exponential (decimal)
2^25	1.00000000000000000000000	25
1	0.0000000000000000000000001	25

Then they are added together

results	1.0000000000000000000000001	25

The last to bits are truncated. The results in decimal is then: $1 * 2^{25} = 33554432$

The result can be verified with the following Java code (float in java is 32bits where double is 64bit):

```
float x = 33554432;
float y = 1;
float z=x+y;
System.out.println(z);
```

Or python code:

```
import numpy as np
x = np.float32(33554432)
y = np.float32(1)
z=x+y
print(z)
```

7 Calculate 1hour + 1 microsecond using IEEE standard for single precision floating point.

The unit could be seconds.

	Fraction (binary)	Exponential (decimal)
3600s	1.11000010000000000000000	11
10^{-6}s	1.00001100011011100000000	-20

Solution

Since the difference is 31, the microsecond is shiftet 31 times to the right and there for completely disappears.

32 bit does not have good enough resolution to this equation.

Can be verified with the following java code

```
float x = 3600f;
float y = 0.000001f;
float z=x+y;
System.out.println(z);
```