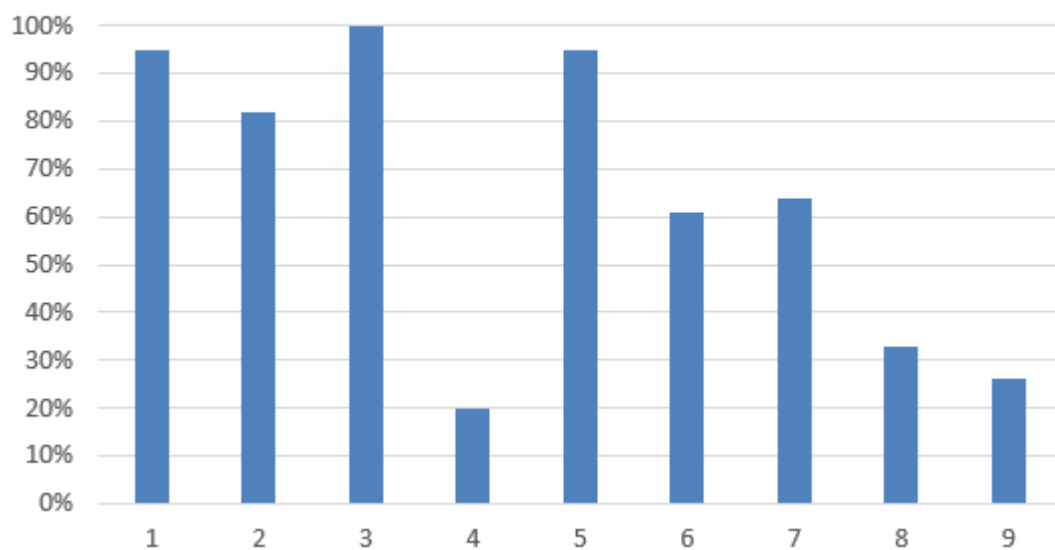
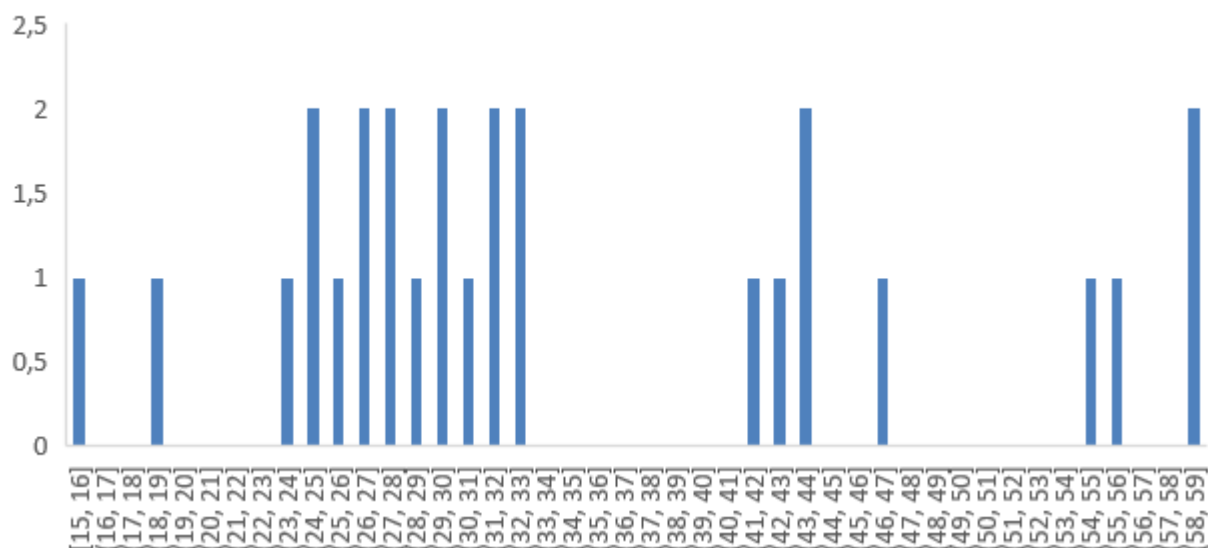


Statistika zkoušky 23. června 2023**Úspěšnost u otázek 1 až 9**

Procenta udávají průměrný počet bodů vztahený k maximu z otázky


Četnost bodů z písemky

Zde
nepište

4

$$Y = \dots \dots | \dots \dots | \dots \dots | \dots \dots | \dots \dots |$$
$$Y = \dots \dots | \dots \dots | \dots \dots | \dots \dots | \dots \dots |$$
$$Y = \dots \dots | \dots \dots | \dots \dots | \dots \dots | \dots \dots |$$

6

f1=  **B**
A
C

6

4

y_1 ☐

 $y_2 \quad \square$ v3 ☐

y4 ☐

4

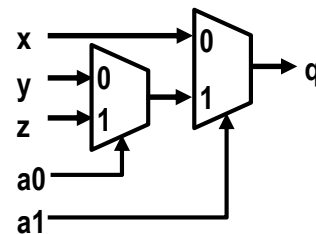
6

6

--	--	--	--	--

end function;

5. Použijte pouze hradla AND, NAND, OR, NOR a inventory NOT.
Zapojte s nimi zapojte schéma vlevo.



8

6. Obvod z otázky 5 nahoře vytvořte ve VHDL nejjednodušším možným způsobem.

```
library ieee; use ieee.std_logic_1164.all; use ieee.numeric_std.all;
entity yyy is port( x,y,z, a1,a0: in std_logic; q: out std_logic);
end entity;
architecture dataflow of yyy is
```

```
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
```

```
end architecture;
```

6

7. C program hledá **minimum** v poli **data**.

```
int data[] = { 0, 1, 2, 3, 4, -5, -6, -7, 8, 9 };  
int min = INT_MAX; // =2147483647 (v <limits.h>)  
for (int i = 0; i < sizeof(data)/sizeof(int); i++) // i<10  
{ if (data[i] < min) min = data[i];  
}
```

Kolik v něm bude chybných skokových predikcí za předpokladu, že for-loop s konstantními mezemi se úsporně přeložila cyklem do-while a procesor používá pouze:

1bitové prediktory, které měly svůj výchozí stav Not-Taken, NT, misses=.....

2bitové prediktory, které měly svůj výchozí stav WT, Weakly Taken, misses=.....

8. 32-bitový procesor čte data z adres 0x14, 0x94, 0x14, 0x94, 0x114, 0x14

Nechť jeho cache má jenom **128 bytů a** je organizovaná jako 2-cestná

částečně asociativní o délce řádku **1 slovo**. Používá se v ní nahrazovací algoritmus LRU.

Napište cache miss a nakreslete finální stav ovlivněných setů cache a jaké adresy budou v nich načtené spolu s jejich tagy.

9. **Prémiová otázka, u níž se uzná jen plně funkční řešení:** Vytvořte 3bitový synchronní Johnsonův čítač, který s každou náběžnou hranou clk provede operaci: při RESET='1' má Q="000".

Při RESET='0' a DN=0, čítá vpřed, tedy má výstupy Q = 000,001, 011,111,110, 100, 000, 001... atd.

Je-li RESET='0' a DN=1, čítá stejnou posloupnost opačným směrem.

```
library ieee; use ieee.std_logic_1164.all; use ieee.numeric_std.all;
```

```
entity JC3 is port (clk, DN, RESET: in std_logic;
```

```
                  Q:out std_logic_vector(2 downto 0));
```

```
end entity;
```

```
architecture rtl of JC3 is
```

```
end architecture;
```