Cvičení 10

Příklad 1: Zjistěte, co dělá následující stroj RAM:

```
          LOAD   =1
          STORE  3
          READ
  loop:   JZERO  output
          STORE  2
          LOAD   3
          ADD    0
          STORE  3
          LOAD   2
          SUB    =1
          JUMP   loop
output:   LOAD   3
          WRITE
          HALT
```

Řešení: Přečte ze vstupu číslo n. Pokud je $n \geq 0$, vypíše hodnotu 2^n a zastaví se. V opačném případě (tj. pokud $n < 0$) se výpočet stroje nikdy nezastaví.

Příklad 2: Pro každý z následujících problémů navrhněte stroj RAM, který ho řeší.

Poznámka: Při konstrukci stroje nemusíte řešit chybná data na vstupu, která neodpovídají zadání.

a) Vstup: celá čísla x, y (tj. $x, y \in \mathbb{Z}$)
Výstup: hodnota $x + y$

b) Vstup: celá čísla x, y (tj. $x, y \in \mathbb{Z}$)
Výstup: $\max\{x, y\}$

Řešení:

```
        READ
        STORE  2
        READ
        STORE  3
        SUB    2
        JGTZ   L1
        LOAD   2
        JUMP   L2
 L1:    LOAD   3
 L2:    WRITE
        HALT
```

c) Vstup: přirozené číslo n (tj. $n \in \mathbb{N}$)

VÝSTUP: sekvence čísel $1, 2, \ldots, n$

Poznámka: Pro $n = 0$ bude sekvence na výstupu prázdná.

Řešení:

```
        READ
        STORE  2
        LOAD   =0
 loop:  STORE  3
        SUB    2
        JZERO  end
        LOAD   3
        ADD    =1
        WRITE
        JUMP   loop
  end:  HALT
```

d) VSTUP: sekvence čísel $a_1, a_2, \ldots, a_n, 0$, kde $n \geq 0$ a $a_i \in \mathbb{Z} - \{0\}$ pro $1 \leq i \leq n$
VÝSTUP: $\prod_{i=1}^{n} a_i$

Poznámka: Pro $n = 0$ bude výstupem hodnota 1.

e) VSTUP: sekvence čísel $a_1, a_2, \ldots, a_n, 0$, kde $n \geq 0$ a $a_i \in \mathbb{Z} - \{0\}$ pro $1 \leq i \leq n$
VÝSTUP: sekvence čísel $a_n, a_{n-1}, \ldots, a_1$

Příklad 3: Sestavte program pro stroj RAM, který přečte ze vstupu číslo n a vypíše na výstup n-té Fibonacciho číslo F_n. Můžete předpokládat, že číslo n na vstupu je nezáporné (tj. nemusíte řešit situaci, kdy $n < 0$). Připomeňme, že Fibonacciho čísla $F_0, F_1, F_2, \ldots$ jsou definována následujícím rekurentním vztahem:

$$F_n = \begin{cases} 0 & \text{pro } n = 0 \\ 1 & \text{pro } n = 1 \\ F_{n-1} + F_{n-2} & \text{pro } n > 1 \end{cases}$$

Příklad 4: Sestavte program pro stroj RAM, který přečte ze vstupu dvě čísla x a k a na výstup vypíše hodnotu k-tého bitu čísla x (tj. 0 nebo 1), přičemž bity jsou číslovány od 0 a 0-tý bit je nejméně významný bit. Můžete předpokládat, že $x \geq 0$ a $k \geq 0$ (tj. nemusíte řešit situace, kdy $x < 0$ nebo $k < 0$).

Řešení:

```
        READ
        STORE  3
        READ
loop:   JZERO  end
        STORE  2
        LOAD   3
        DIV    =2
        STORE  3
        LOAD   2
        SUB    =1
        JUMP   loop
 end:   LOAD   3
        DIV    =2
        ADD    0
        STORE  4
        LOAD   3
        SUB    4
        WRITE
        HALT
```

Příklad 5: Navrhněte program pro stroj RAM, který načte ze vstupu dvě čísla x a y (můžete předpokládat, že $x \geq 0$ a $y \geq 0$) a na výstup vypíše jejich součin $x \cdot y$. Aby to nebylo tak jednoduché, musíte navíc dodržet následující omezení:

- Ve vašem programu *nesmíte* použít instrukce `MUL` a `DIV`. Můžete ovšem použít novou speciální instrukci `RSHIFT`, která má stejný význam jako `DIV =2`.
- Celkový počet instrukcí, který váš program provede, musí být polynomiální vzhledem k počtům bitů nutných pro zápis čísel x a y.