

Cvičení 7

Příklad 1: Odsimulujte činnost následujícího programu pro stroj RAM, pokud jako vstup dostane posloupnost (délky 1) tvořenou jediným číslem 4, tj. vypište posloupnost jednotlivých konfigurací, kterými stroj RAM projde během tohoto výpočtu:

```

0:  $R_2 := \text{READ}()$ 
1:  $R_1 := 2$ 
2: goto 5
3:  $R_1 := R_1 * R_1$ 
4:  $R_2 := R_2 - 1$ 
5: if ( $R_2 > 0$ ) goto 3
6:  $\text{WRITE}(R_1)$ 
7: halt

```

Určete, co tento program počítá, tj. co vydá jako výstup, když na vstupu dostane číslo n .

Řešení: Pro hodnoty n , kde $n \geq 0$, vypíše na výstup hodnotu 2^{2^n} . Pro hodnoty n , kde $n < 0$, vypíše na výstup hodnotu 2.

Příklad 2: Určete, co dělá následující program pro stroj RAM, tj. popište detailně jeho činnost pro libovolný vstup a popište, co bude výstupem.

Poznámka: Pro přehlednost zde nejsou na rozdíl od předchozího příkladu explicitně uvedeny adresy instrukcí, ale jsou místo nich použita symbolická návěští.

$R_4 := 4$	$R_0 := [R_1]$
$R_3 := \text{READ}()$	$R_0 := R_0 + 1$
$R_1 := R_4 + R_3$	$[R_1] := R_0$
$R_0 := 0$	goto L ₂
L ₁ : if ($R_1 = R_4$) goto L ₂	L ₃ : $R_2 := 1$
$[R_1] := R_0$	L ₄ : if ($R_2 > R_3$) goto L ₅
$R_1 := R_1 - 1$	$R_1 := R_4 + R_2$
goto L ₁	$R_0 := [R_1]$
L ₂ : $R_2 := \text{READ}()$	$\text{WRITE}(R_0)$
if ($R_2 \leq 0$) goto L ₃	$R_2 := R_2 + 1$
if ($R_2 > R_3$) goto L ₃	goto L ₄
$R_1 := R_4 + R_2$	L ₅ : halt

Příklad 3: Pro každý z následujících problémů navrhnete program pro 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 $x + y$

Řešení:

```

R0 := READ ()
R1 := READ ()
R0 := R0 + R1
WRITE (R0)
halt

```

- b) VSTUP: celá čísla x, y (tj. $x, y \in \mathbb{Z}$)
 VÝSTUP: $\max\{x, y\}$

Řešení:

```

R0 := READ ()
R1 := READ ()
if (R0 ≥ R1) goto L1
R0 := R1
L1: WRITE (R0)
halt

```

- c) VSTUP: přirozené číslo n (tj. $n \in \mathbb{N}$)
 VÝSTUP: sekvence čísel $1, 2, \dots, n$

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

Řešení:

```

R1 := READ ()
R0 := 1
L1: if (R0 > R1) goto L2
      WRITE (R0)
      R0 := R0 + 1
      goto L1
L2: halt

```

- d) VSTUP: sekvence čísel $a_1, a_2, \dots, a_n, 0$, 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: Zápis $\prod_{i=1}^n a_i$ označuje součin $a_1 \cdot a_2 \cdot \dots \cdot a_n$. Pro $n = 0$ bude výstupem hodnota 1.

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

Příklad 4: 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 $F_0, F_1, F_2, \dots$ jsou definována následujícím rekurentním vztahem:

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

Řešení:

```

R1 := READ ()
R0 := 0
R2 := 0
R3 := 1
L1: if (R0 ≥ R1) goto L2
    R4 := R2 + R3
    R2 := R3
    R3 := R4
    R0 := R0 + 1
    goto L1
L2: WRITE (R2)
    halt

```

Příklad 5: Vezměme si následující Algoritmus 1. Vstupem tohoto algoritmu může být libovolné přirozené číslo n (tj. hodnoty proměnné n mohou být libovolná neomezeně velká přirozená čísla).

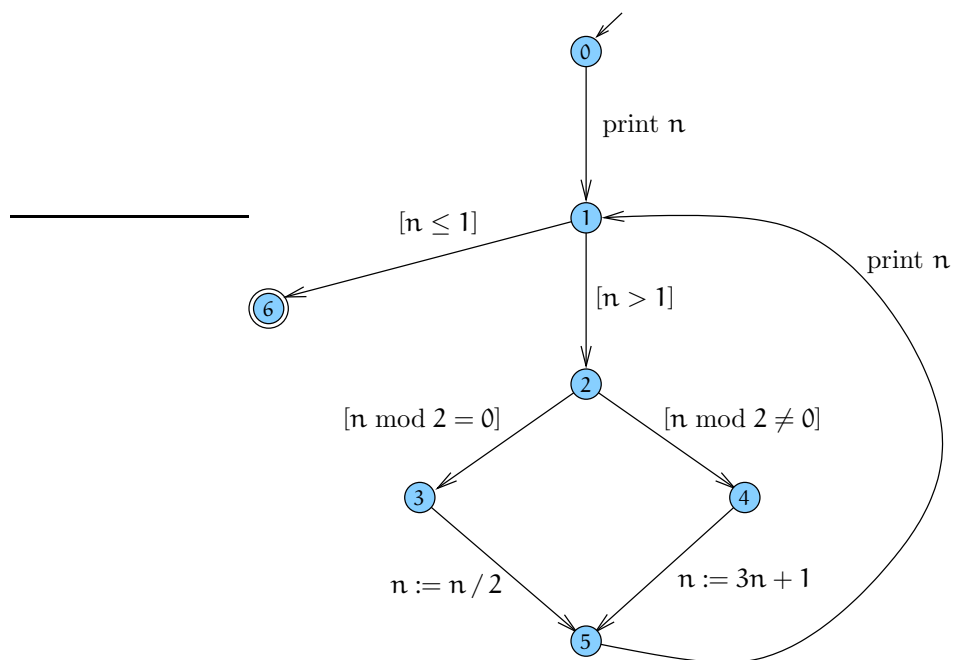
Algoritmus 1:

```

PRINTSEQ (n):
    print n
    while n > 1 do
        if n mod 2 = 0 then
            n := n / 2
        else
            n := 3 * n + 1
        print n

```

- Nakreslete graf řídicího toku tohoto algoritmu.
- Popište výpočet, který tento algoritmus provede, pokud jako vstup dostane číslo 5. Vypište posloupnost jednotlivých konfigurací při tomto výpočtu.
- Kolik kroků provede tento algoritmus, když jako vstup dostane číslo 7? Co bude výstupem?



Obrázek 1: Graf řídicího toku

d) Vytvořte program pro stroj RAM realizující činnost tohoto algoritmu.

Poznámka: Hodnotu proměnné n načte tento stroj RAM ze vstupu.

Řešení:

a) Graf řídicího toku je uveden na Obrázku 1.

b) Výpočet vypadá následovně

Krok	Stav	n
0	0	5
1	1	5
2	2	5
3	4	5
4	5	16
5	1	16
6	2	16
7	3	16
8	5	8
9	1	8
10	2	8
11	3	8
12	5	4
13	1	4
14	2	4
15	3	4
16	5	2
17	1	2
18	2	2
19	3	2
20	5	1
21	1	1
22	6	1

c) Výstupem bude posloupnost 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1.

Celkový počet kroků se určí následovně: Cyklem se projde celkem 16 krát. Při každém průchodu cyklem se provedou 4 kroky. Na začátku se provede mimo cyklus jeden krok a konci také jeden krok. Celkem se tedy provede $1 + 16 \cdot 4 + 1 = 66$ kroků.

d)

<pre> R2 := READ () WRITE (R2) L1: if (R2 ≤ 1) goto L4 R1 := ⌊R2 / 2⌋ R0 := R1 * 2 R0 := R2 - R0 if (R0 ≠ 0) goto L2 R2 := R1 goto L3 L2: R2 := R2 * 3 R2 := R2 + 1 L3: WRITE (R2) goto L1 L4: halt </pre>	nebo	<pre> R2 := READ () L1: WRITE (R2) if (R2 ≤ 1) goto L3 R1 := ⌊R2 / 2⌋ R0 := R1 * 2 R0 := R2 - R0 if (R0 ≠ 0) goto L2 R2 := R1 goto L1 L2: R2 := R2 * 3 R2 := R2 + 1 goto L1 L3: halt </pre>
--	------	---

Příklad 6: Uvažujme následující algoritmus 2 popsaný pseudokódem.

Algoritmus 2: Třídění přímým vkládáním

```

INSERTION-SORT ():
  n := READ ()
  for i := 0 to n - 1 do
    A[i] := READ ()
  for j := 1 to n - 1 do
    x := A[j]
    i := j - 1
    while i ≥ 0 and A[i] > x do
      A[i + 1] := A[i]
      i := i - 1
    A[i + 1] := x
  for i := 0 to n - 1 do
    WRITE (A[i])

```

- Nakreslete graf řídicího toku reprezentující tento pseudokód.
- Implementujte tento algoritmus jako program pro stroj RAM.
- Popište, jak různé části vámi vytvořeného kódu programu pro stroj RAM odpovídají jednotlivým hranám v grafu řídicího toku.

Řešení:

$R_5 := 7$	$R_1 := R_1 + 1$
$R_4 := \text{READ} ()$	$[R_1] := R_0$
$R_2 := 0$	$R_2 := R_2 - 1$
$L_1: \text{if } (R_2 \geq R_4) \text{ goto } L_2$	goto L_4
$R_0 := \text{READ} ()$	$L_5: R_1 := R_5 + R_2$
$R_1 := R_5 + R_2$	$R_1 := R_1 + 1$
$[R_1] := R_0$	$[R_1] := R_6$
$R_2 := R_2 + 1$	$R_3 := R_3 + 1$
goto L_1	goto L_3
$L_2: R_3 := 1$	$L_6: R_2 := 1$
$L_3: \text{if } (R_3 \geq R_4) \text{ goto } L_6$	$L_7: \text{if } (R_2 \geq R_4) \text{ goto } L_8$
$R_1 := R_5 + R_3$	$R_1 := R_5 + R_2$
$R_6 := [R_1]$	$R_0 := [R_1]$
$R_2 := R_3 - 1$	$\text{WRITE} (R_0)$
$L_4: \text{if } (R_2 < 0) \text{ goto } L_5$	$R_2 := R_2 + 1$
$R_1 := R_5 + R_2$	goto L_7
$R_0 := [R_1]$	$L_8: \text{halt}$
if $(R_0 \leq R_6) \text{ goto } L_5$	

Příklad 7: Popište, jak k libovolnému Turingovu stroji $\mathcal{M}$ vytvořit program pro stroj RAM realizující stejný algoritmus jako stroj $\mathcal{M}$. Uvažujte následující varianty Turingových strojů:

- Turingův stroj s jednou jednostranně nekonečnou páskou
- Turingův stroj s jednou oboustranně nekonečnou páskou
- Turingův stroj s více oboustranně nekonečnými páskami

Poznámka: Není třeba, abyste dané programy pro stroj RAM explicitně vytvářeli. Stačí, pokud slovně, ale dostatečně podrobně, popíšete, jak budou dané stroje RAM konkrétně fungovat.

Příklad 8: 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:

```

R1 := READ ()
R0 := READ ()
L1: if (R0 ≤ 0) goto L2
    R1 := ⌊R1 / 2⌋
    R0 := R0 − 1
    goto L1
L2: R0 := ⌊R1 / 2⌋
    R0 := R0 * 2
    R1 := R1 − R0
    WRITE (R1)
    halt

```

Příklad 9: 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 aritmetické instrukce s operacemi násobení a dělení. Můžete ovšem použít aritmetickou instrukci pro bitový posun doprava o jeden bit:

$$R_i := rshift(R_j)$$

která provede to samé, co by provedla instrukce $R_i := \lfloor R_j / 2 \rfloor$.

- Celkový počet instrukcí, který váš program provede, musí být polynomiální vzhledem k počtům bitů nutných pro zápis čísel x a y .
- Napadá vás nějaký způsob, jak vyřešit tento problém bez použití instrukce typu $R_i := rshift(R_j)$, tj. jak spočítat hodnotu $x \cdot y$ na stroji RAM, který má z aritmetickým operací k dispozici pouze sčítání a odčítání, tak, aby celkový počet kroků byl polynomiální vzhledem počtu bitů čísel x a y ?

Řešení:

```

R0 := READ ()
R1 := READ ()
R2 := 0
L1: if (R1 ≤ 0) goto L3
    R3 := rshift(R1)
    R4 := R3 + R3
    R4 := R1 - R4
    if (R4 = 0) goto L2
    R2 := R2 + R0
L2: R1 := R3
    R0 := R0 + R0
    goto L1
L3: WRITE(R2)
    halt
```