

PSEUDOKÓD

Adam Klepáč

12. října 2022

Gymnázium Evolution Jižní Město

I. Předmluva

Počítačový program

CPU

Operace

Typy instrukcí

Program

CPU

Co je CPU?



CPU (**C**entral **P**rocessing **U**nit) je elektrický obvod, který vykonává **instrukce** tvořící **počítačový program**.

Program

Operace

Fetch

Fetch

CPU **vyzvedne** instrukci z programu.

- Instrukce je v programu uložena jako posloupnost nul a jedniček.
- Poloha (adresa) instrukce je dána čítačem (program counter). Postaru se mu někdy říká 'hlava'.
- Čítač uchovává adresu poslední instrukce a po přičtení se posune o délku (v bitech) zpracovávané instrukce.

Decode

Decode

CPU přeloží načtenou instrukci.

- Prvních pár bitů v instrukci obvykle značí operaci, která se má provést.
- Zbývající bity jsou pak například adresy operandů v paměti.

Execute

Execute

CPU **provede** přeloženou instrukci.

- V závislosti na architektuře CPU, instrukce obsahují buď jedinou akci nebo posloupnost akcí.
- Výsledek je uložen ve vnitřní paměti CPU. Uložení do vnější paměti (třeba RAM nebo disk) musí být obsahem nějaké další instrukce.

Program

Instrukce

Paměťové operace

- **SET**: Nastav blok vnitřní paměti na danou hodnotu.

Paměťové operace

- **SET**: Nastav blok vnitřní paměti na danou hodnotu.
- **COPY**: Zkopíruj hodnotu z bloku vnitřní paměti do jiného bloku.

Paměťové operace

- **SET**: Nastav blok vnitřní paměti na danou hodnotu.
- **COPY**: Zkopíruj hodnotu z bloku vnitřní paměti do jiného bloku.
- **READ/WRITE**: Zapisuj data nebo je čti z připojených zařízení.

Početní a logické operace

- Sečti/odečti/vynásob/vyděl spolu hodnoty ve dvou blocích paměti.

Početni a logické operace

- Sečti/odečti/vynásob/vyděl spolu hodnoty ve dvou blocích paměti.
- Konjunkce/disjunkce dvou uložených hodnot.

Početni a logické operace

- Sečti/odečti/vynásob/vyděl spolu hodnoty ve dvou blocích paměti.
- Konjunkce/disjunkce dvou uložených hodnot.
- Porovnej spolu dvě uložené hodnoty.

Početní a logické operace

- Sečti/odečti/vynásob/vyděl spolu hodnoty ve dvou blocích paměti.
- Konjunkce/disjunkce dvou uložených hodnot.
- Porovnej spolu dvě uložené hodnoty.
- Operace na desetinných číslech (floating point arithmetic).

Řídící (control flow) operace

- **Odboč** na jiné místo v programu a vykoněj tamější instrukce.

Řídící (control flow) operace

- **Odboč** na jiné místo v programu a vykonej tamější instrukce.
- **Podmínečně odboč** na jiné místo v programu a vykonej tamější instrukce.

Řídící (control flow) operace

- **Odboč** na jiné místo v programu a vykonej tamější instrukce.
- **Podmínečně odboč** na jiné místo v programu a vykonej tamější instrukce.
- **Zavolej** jiný blok kódu, uchovav následující instrukci jako místo návratu.

II. Pseudokód

Definice a příklady

Definice

Příklady

Základní koncepty

Proměnné

Podmínky

Cykly

Procedury

Definice a příklady

Definice

Pseudokód

Pseudokód je **neformální** zápis počítačového programu/algoritmu.

Definice a příklady

Příklady

Sčítání pod sebou

```
input :  $A, B \in \mathbb{N}$            //  $A, B$  mají stejný počet číslic
output:  $A + B$ 

1  $c \leftarrow 0$ 
2  $i \leftarrow 0$ 
3  $r \leftarrow 0$ 
4 while  $A > 0$  and  $B > 0$  do
5    $x \leftarrow A \bmod 10$ 
6    $y \leftarrow B \bmod 10$ 
7    $r \leftarrow r + (x + y + c \bmod 10) \cdot 10^i$ 
8   if  $x + y + c \geq 10$  then
9      $c \leftarrow 1$ 
10  else
11     $c \leftarrow 0$ 
12   $A \leftarrow \lfloor A/10 \rfloor$ 
13   $B \leftarrow \lfloor B/10 \rfloor$ 
14   $i \leftarrow i + 1$ 
15 return  $r$ 
```

Přihlášení uživatele

```
input  : user, pass      // uživatelské jméno, heslo
output: true/false      // přihlášení úspěšné/neúspěšné

1 if user není uložen v databázi then
2   | return false
3 if pass neodpovídá heslu uloženému pod user v databázi then
4   | return false
5 return true
```

Základní koncepty

Proměnné

Co to je proměnná?

Proměnná

Proměnná je písmeno nebo slovo, které **představuje nějakou hodnotu**. Tato hodnota může být na různých místech programu různá.

Základní koncepty

Podmínky

Co to je podmínka?

Podmínka

Podmínka je **logický výraz** určující, která sada následujících instrukcí v programu se má provést.

Příklad 1: absolutní hodnota

input : $x \in \mathbb{R}$

output: $|x|$

1 **if** $x \geq 0$ **then**

2 **return** x

3 **else**

4 **return** $-x$

Příklad 2: písmeno ve slově

```
input : word          // nějaké slovo
output: true/false    // slovo obsahuje/neobsahuje písmeno 'b'

1 if word obsahuje 'b' then
2   | return true
3 else
4   | return false
```

Základní koncepty

Cykly

Co je to cyklus?

Cyklus

Cyklus je posloupnost instrukcí, která se v průběhu algoritmu **několikrát za sebou opakuje**, pokaždé s jinými vstupními daty.

For cyklus (uzavřený cyklus)

For cyklus

Cyklus, který se opakuje postupně pro **všechny prvky z nějaké konečné množiny**.

Příklad 1: násobení množiny dvěma

input : A // množina čísel

output: B // množina obsahující všechny prvky A vynásobené 2

1 $B \leftarrow \{\}$

2 **for** $x \in A$ **do**

3 $B \leftarrow B \cup \{2 \cdot x\}$

4 **return** B

Příklad 2: všechna menší prvočísla

input : $n \in \mathbb{N}$ // přirozené číslo udávající rozsah
output: P // množina prvočísel menších než n

```
1  $P \leftarrow \{\}$ 
2 for  $x$  from 1 to  $n$  do
3   if  $x$  je prvočíslo then
4      $P \leftarrow P \cup \{x\}$ 
5 return  $P$ 
```

While cyklus (otevřený cyklus)

While cyklus

Cyklus, který se opakuje, **dokud je splněna nějaká podmínka**.

Příklad 1: kdo neumí pseudokód

input : IVTstudent // jméno studenta IVT
output: $n \in \mathbb{N}$ // počet napsaných algoritmů

```
1  $n \leftarrow 0$ 
2 while IVTstudent neumí psát pseudokód do
3   | IVTstudent napíše algoritmus v pseudokódu.
4   |  $n \leftarrow n + 1$ 
5 return n
```


Příklad 2: rozklad na prvočísla

input : $n \in \mathbb{N}$ // přirozené číslo k rozložení

output: $\{(p_i, m_i)\}$ // posloupnost dvojic (prvočísla, mocnina)

```
1  $p_0 \leftarrow 2$ 
2  $m_0 \leftarrow 0$ 
3  $i \leftarrow 0$ 
4 while  $n > 1$  do
5   while  $p_i$  dělí  $n$  do
6      $n \leftarrow n / p_i$ 
7      $m_i \leftarrow m_i + 1$ 
8    $p_{i+1} \leftarrow$  nejbližší prvočísla větší než  $p_i$ 
9    $m_{i+1} \leftarrow 0$ 
10   $i \leftarrow i + 1$ 
11 return  $\{(p_i, m_i)\}$ 
```

Základní koncepty

Procedury

Co je to procedura?

Procedura

Procedura je **blok kódu**, který mohu (s různými vstupními daty) provést během algoritmu několikrát. Vlastně je to jakýsi malý algoritmus uvnitř algoritmu.

Příklad 1: věková skupina

Procedure věk(d, m, r)

input : datum narození // den, měsíc, rok

output: věk // věk člověka (v letech)

- 1 $D, M, R \leftarrow$ současný den, současný měsíc, současný rok
- 2 $V \leftarrow R - r$

Příklad 1: věková skupina

Procedure věk(d, m, r)

input : datum narození // den, měsíc, rok

output: věk // věk člověka (v letech)

- 1 $D, M, R \leftarrow$ současný den, současný měsíc, současný rok
- 2 $V \leftarrow R - r$
- 3 **if** ($M < m$) **or** ($M = m$ **and** $D < d$) **then**
- 4 $V \leftarrow V - 1$
- 5 **return** V

Příklad 1: věková skupina (pokr.)

input : datum narození // den, měsíc, rok

output: věková skupina // dítě/dospělý/důchodce

1 $V \leftarrow \text{věk}(\text{den}, \text{měsíc}, \text{rok})$

Příklad 1: věková skupina (pokr.)

input : datum narození // den, měsíc, rok
output: věková skupina // dítě/dospělý/důchodce

- 1 $V \leftarrow \text{věk}(\text{den}, \text{měsíc}, \text{rok})$
- 2 **if** $V \leq 18$ **then**
- 3 **return** dítě

Příklad 1: věková skupina (pokr.)

input : datum narození // den, měsíc, rok
output: věková skupina // dítě/dospělý/důchodce

```
1 V ← věk(den, měsíc, rok)
2 if V ≤ 18 then
3   | return dítě
4 else if 18 < V < 65 then
5   | return dospělý
```


Příklad 1: věková skupina (pokr.)

input : datum narození // den, měsíc, rok
output: věková skupina // dítě/dospělý/důchodce

```
1 V ← věk(den, měsíc, rok)
2 if V ≤ 18 then
3   | return dítě
4 else if 18 < V < 65 then
5   | return dospělý
6 else
7   | return důchodce
```

Příklad 2: spamming

input : list = {(name, e-mail)} // seznam (jmě, e-mail)

output: nic

1 list \leftarrow seřad'(list, name)

Příklad 2: spamming

input : list = {(name, e-mail)} // seznam (jmě, e-mail)

output: nic

```
1 list ← seřad'(list, name)
2 for (name, e-mail) ∈ list do
3   exc ← ! * 1000
4   subject ← VYHRÁLI JSTE 300,000 Kč exc
5   body ← Vážený/á/é name, ...
6   pošli(e-mail, subject, body)
```

Díky za pozornost.