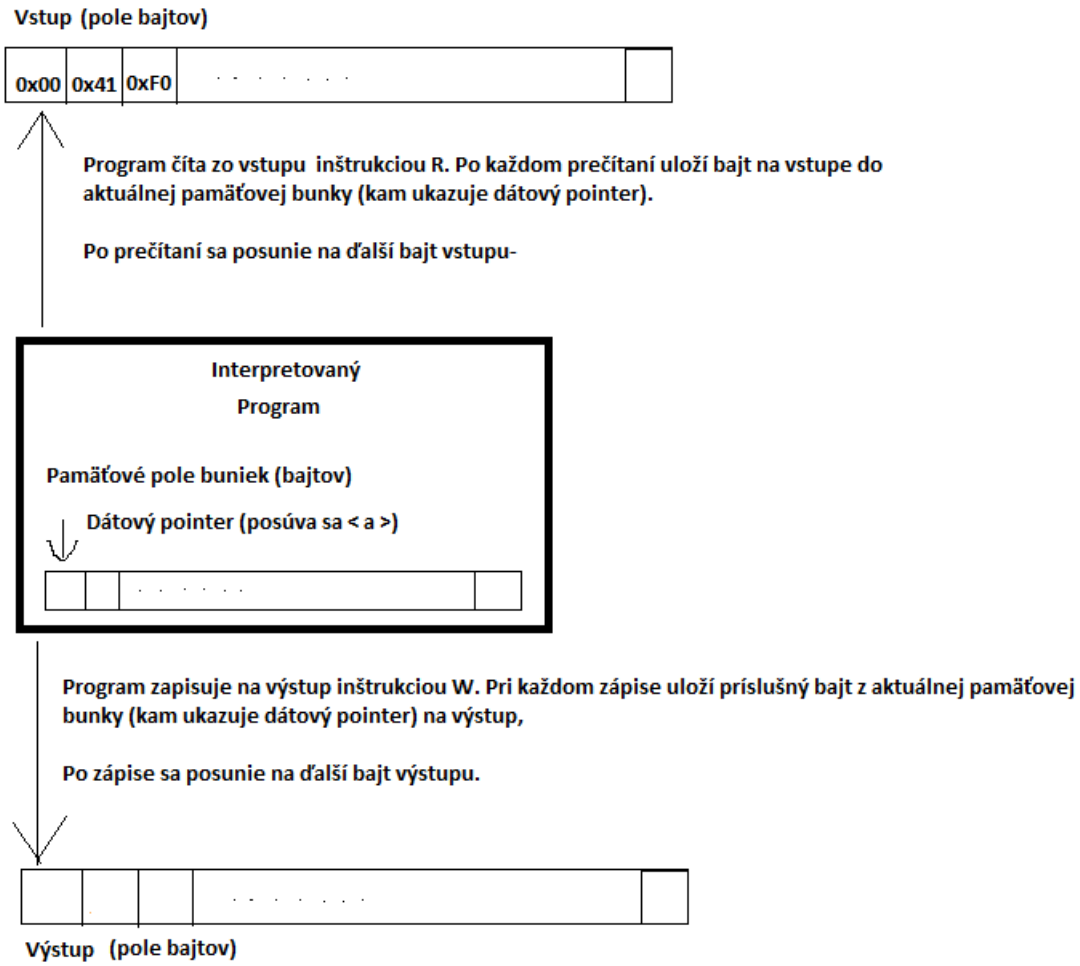
Zadanie č. 1

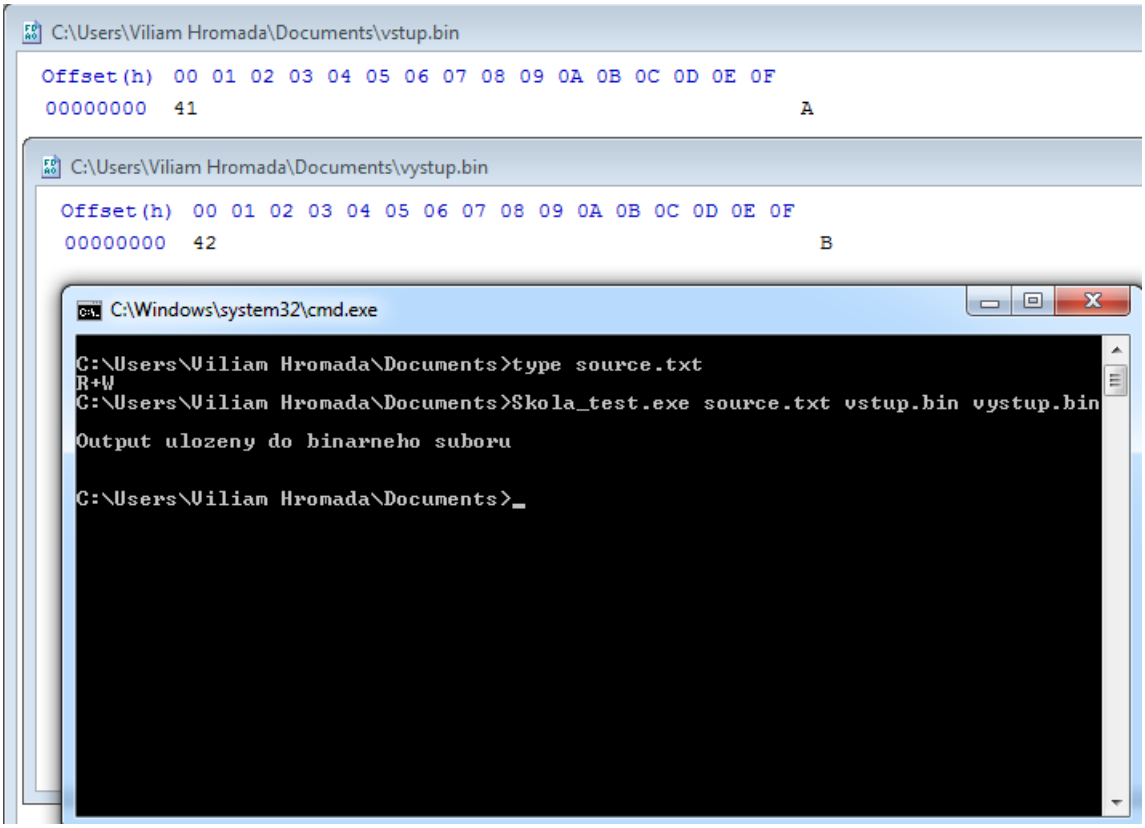
- Naprogramujte program v C, C++, Java, ktorý interpretuje zdrojový kód programovacieho jazyka popísaného nižšie. Váš program tento zdrojový kód vykoná (interpretuje) v súlade s platnými inštrukciami jazyka. Ak zdrojový kód obsahuje syntaktické chyby, je potrebné o tom minimálne vypísať **chybovú hlášku**!
- V súlade s popisom programovacieho jazyka a jeho fungovania je potrebné zabezpečiť pre daný program vstup a výstup.
- Vstupom Vášho programu bude teda:
  - Zdrojový kód pre nižšie popísaný programovací jazyk.
  - Pole bajtov predstavujúce vstup programu, zapísaného v zdrojovom kóde.
- Výstupom Vášho programu bude:
  - Výstup programu, zapísaného v zdrojovom kóde, t.j. výstupné pole bajtov.
- To, ako načítate zdrojový kód a vstupy interpretovaného programu je na Vás (t.j. či budú zadávané počas behu programu, alebo sa zadajú najprv do nejakého súboru a následne sa odtiaľ načítajú, pričom súbory budú argumenty programu).
- **Pre jednoduchosť uvažujte**, že vstup interpretovaného programu je známy už pred jeho vykonávaním, t.j. najprv sa zadá vstup pre interpretovaný program a až potom sa tento program začne vykonávať.
- Deadline zadania je 6. marec 2016, 23:59, t.j. polnoc z nedele 6.3.2016 na pondelok, 7.3.2016. Do AIS odovzdajte **zdrojový kód** Vášho programu (interpretera) daného programovacieho jazyka. Taktiež odovzdajte jednoduchú dokumentáciu - **dokument docx/pdf**, popisujúcu Váš interpreter - ako Vaša aplikácia funguje, t.j. ako **načítava vstupy a ako vypisuje výstupy** - aby ju bolo možné jednoducho otestovať a **v akom jazyku je naprogramovaná**.
- Za naprogramovanie a vytvorenie dokumentácie interpretera je 10 bodov.
- **Dodržanie syntaxe daného programovacieho jazyka** a **individuálna práca na zadaní** je NUTNOU podmienkou udelenia viac ako 0 bodov.

- (**BONUS za 5 bodov**): Vymyslite nejaký jednoduchý program a naprogramujte ho v danom programovacom jazyku. Odovzdajte jeho zdrojový kód a napíšte, čo Váš program robí. **Musíte** v ňom využiť inštrukcie : $<, >, +, -, [,]$,R, W.

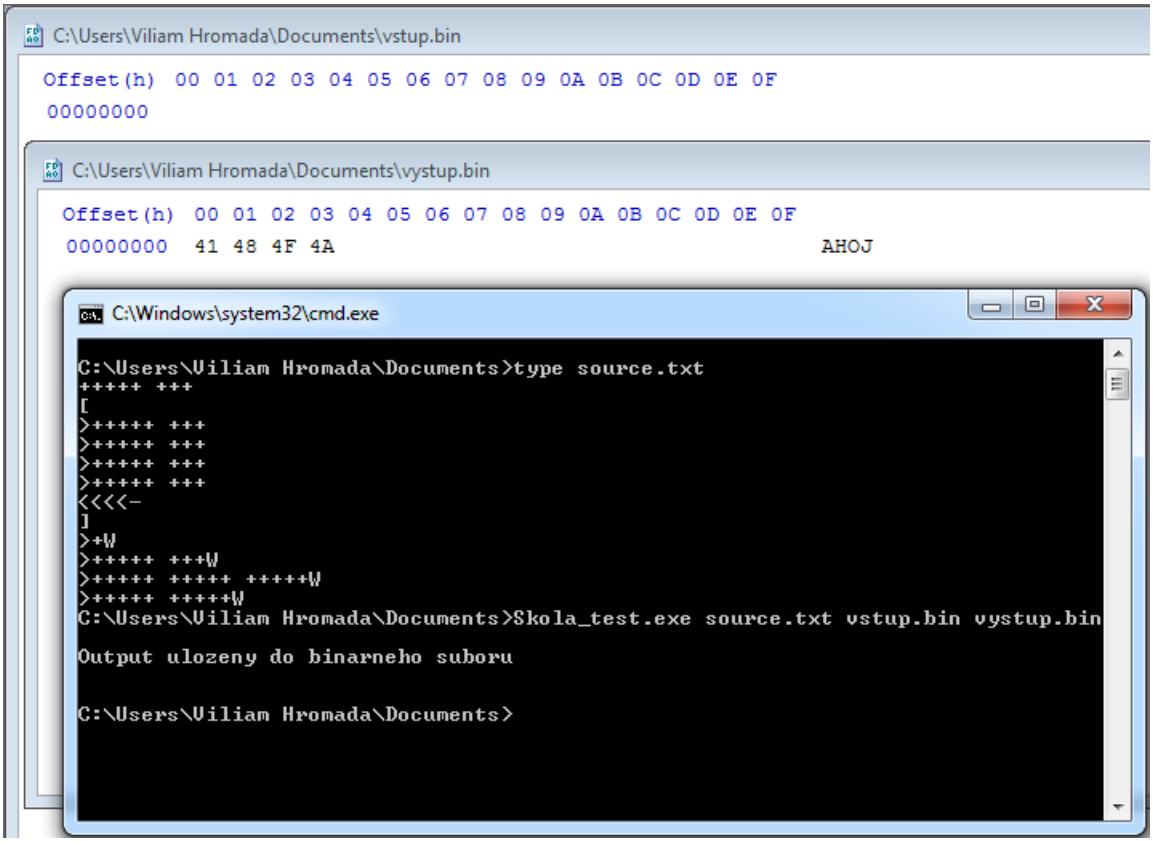
Výpočtový model interpretovaného jazyka je nasledovný:



Obr. 1: Výpočtový model interpretovaného jazyka



Obr. 2: Screenshot spustenia programu, zdrojový kód interpretovaného programu je v súbore *source.txt* a tvoril argument hlavného programu (zadania), vstup a výstup sú reprezentované binárnymi súbormi, tiež argumenty programu



Obr. 3: Screenshot spustenia programu, zdrojový kód interpretovaného programu je v súbore *source.txt* a tvoril argument hlavného programu (zadania), vstup a výstup sú reprezentované binárnymi súbormi, tiež argumenty programu

Je daný nasledujúci programovací jazyk:

- Programovací jazyk obsahuje 10 rôznych inštrukcií.
- Interpretácia jazyka sa vykonáva sekvenčne (t.j. inštrukcia za inštrukciou).
- Keď interpreter spracuje všetky inštrukcie, program končí, t.j. koniec programu je hneď za poslednou inštrukciou.
- Neznáme symboly sa **ignorujú**. T.j. čokoľvek, čo nie je platným príkazom, je komentár.
- Exekúcia programu sa vykonáva nad **poľom buniek**, z ktorých každá má veľkosť 1 bajt (t.j. nad poľom bajtov). Každá bunka teda môže obsahovať hodnoty od 0 po 255 (8 bitov). Uvažujme pole buniek o veľkosti 100 000 buniek (t.j. 100 000 bajtov). Na začiatku sú všetky bunky nastavené na hodnotu 0x00.
- Program má tzv. **dátový ukazovateľ** (dátový pointer), ktorý sa hýbe medzi týmito bunkami a označuje aktuálnu bunku, nad ktorou sa vykonávajú operácie (inštrukcie). Na začiatku programu tento ukazovateľ ukazuje na prvú bunku poľa (prvý bajt). Vie sa posúvať doľava alebo doprava. Ak sa posúva vľavo z prvej bunky, tak sa cyklicky dostane na poslednú bunku. A naopak, posun vpravo z poslednej bunky je posunom na prvú bunku.
- Program taktiež podporuje **vstup** a **výstup**. Oba predstavujú prúdy bajtov. Zo vstupného prúdu bajtov vie program pomocou príslušnej inštrukcie načítať 1 bajt. Taktiež vie, pomocou príslušnej inštrukcie, zapísať 1 bajt do výstupného prúdu bajtov.
- Jazyk je case-sensitive! T.j. nemožno si zamieňať veľké a malé písmená.

Tabuľka príkazov programu:

<	dekrementácia dátového pointera
>	inkrementácia dátového pointera
R	načítaj bajt zo vstupu a ulož ho na pozíciu, kde ukazuje dátový pointer
W	zapíš bajt do výstupu z pozície, kde ukazuje dátový pointer
+	inkrementuj (zvýš o 1) hodnotu bajtu na pozícii, kde ukazuje dátový pointer ak je hodnota bajtu 0xFF, tak po zvýšení bude jeho hodnota 0x00
-	dekrementuj (zníž o 1) hodnotu bajtu na pozícii, kde ukazuje dátový pointer ak je hodnota bajtu 0x00, tak po znížení bude jeho hodnota 0xFF
N	nastav aktuálny bajt, na ktorý ukazuje dátový pointer, na nulový bajt 0x00
!	zneguj aktuálny bajt, na ktorý ukazuje dátový pointer (znak inštrukcie je výkričník) napr. !(0xFF) = 0x00, !(0xAB) = 0x54, t.j. zmeň každý bit v bajte z 0 na 1 a naopak.
[	ak **je hodnota** na pozícii, kde ukazuje dátový pointer **nula**(t.j. 0x00), skoč na inštrukciu za príslušným "]", inak pokračuj na ďalšiu inštrukciu
]	ak **nie je hodnota** na pozícii, kde ukazuje dátový pointer **nula** (t.j. 0x00), skoč na inštrukciu za príslušným "[", inak pokračuj na ďalšiu inštrukciu

Príkazy „["a „] "sú párové, t.j. ku každému „["musí existovať „] "a naopak.
De facto predstavujú cyklus.

Nezabudnite na to, že príkazy „["a „] "**môžu byť aj vnorené**!!!

Príklady programov:

1) Program, ktorý načíta 1 bajt zo vstupu, inkrementuje ho o 1 a vypíše na obrazovku:

```
R+W
```

Vysvetlenie: *R* načíta 1 bajt zo vstupu do pozície, kam ukazuje dátový pointer (t.j. na začiatku je to prvá bunka). + inkrementuje hodnotu, kam ukazuje dátový pointer (t.j. kam sa v predchádzajúcom kroku uložil bajt zo vstupu). *W* následne túto novú inkrementovanú hodnotu pošle na výstup (znovu ako obsah bunky, t.j. 1 bajt). V prípade, že bol vstup povedzme bajt s hodnotou 65 (t.j. v ASCII písmeno *A*), tak sa na výstup zapíše bajt s hodnotou 66 (t.j. písmeno *B*).

Ukážka:
Vstup: A
Výstup: B

Vstup: Z
Výstup: [

2) Program, ktorý nastaví prvých 5 buniek na bajty s hodnotami 0x01, 0x02, 0x03, 0x04 a 0x05.:

```
+>++>+++>++++>+++++
```

Vysvetlenie: + inkrementuje tú bunku, na ktorú ukazuje dátový pointer vždy o 1. Preto prvá bunka bude mať po + hodnotu 1. Následne sa pomocou > posunie dátový pointer na vedľajšiu bunku vpravo. Tá má zase hodnotu 0, preto ju inkrementujeme 2-krát ++ aby sme sem dostali hodnotu 2. Toto opakujeme, kým nenaplníme prvých 5 buniek bajtami s hodnotami 0x01, 0x02, 0x03, 0x04, 0x05.

3) Program, ktorý vypíše na výstup znak A (znak A má s ASCII tabuľke dekadickú hodnotu 65):

```
+++++ +++++ +++++ +++++ +++++ +++++
+++++ +++++ +++++ +++++ +++++ +++++
+++++ W
```

Iná varianta (s cyklom):

```
+++++ +++
[
  >+++++ +++
  <-
]
>+W
```

Vysvetlenie: Prvý program proste inkrementuje prvú bunku 65-krát a následne daný bajt vypíše na obrazovku.

Druhý program najprv nastaví prvú bunku na hodnotu 8. Inštrukcia [znamená, že sa testuje, či je bunka, kam ukazuje dátový pointer (t.j. momentálne prvá) rovná nule. Ak by bola, program skočí za príslušnú opačnú zátvorku]. Keďže nie je, (lebo je rovná 8), program pokračuje s ďalšou inštrukciou. Tá hovorí, aby sa dátový pointer posunul vpravo o 1 pozíciu. Následne sa táto bunka 8-krát inkrementuje. V tomto momente sú teda v poli buniek prvé 2 bunky (prvé 2 bajty) rovné 0x08 a 0x08. Na ďalšom riadku je prvou inštrukciou posun dátového pointera vľavo (t.j. späť na prvú bunku) a jej dekrementácia o 1. Teraz sú prvé 2 bunky rovné 0x07 a 0x08. Nasleduje inštrukcia], ktorá otestuje, či je aktuálna bunka rovná nule. Keďže aktuálne je dátový pointer na prvej bunke a tam je bajt 0x07, tak sa program znovu vracia na inštrukciu za príslušnou opačnou zátvorkou [. Takto vlastne docielime, že vždy, keď sa prvý bajt dekrementuje o 1, tak druhý bajt sa inkrementuje o 8. Po 8-iteráciách bude prvý bajt 0x00 a druhý bajt 0x41 (dekadicky 65). Keď inštrukcia $-$ dekrementuje prvý bajt a ten dosiahne hodnotu nula, ďalšou je znovu inštrukcia]. Tá testuje, či je bajt, na ktorý ukazuje dátový pointer, nula. Keďže je, tak sa už nevracia znovu na inštrukciu za zátvorkou [, ale pokračuje ďalej, t.j. vyjde zo slučky von na ďalšie inštrukcie $> +W$. T.j. posunie sa znovu na 2. bunku, inkrementuje ju o 1 (čím má teraz hodnotu 0x41) a vypíše ju na výstup. Keďže bajt s hodnotou 0x41 je znak A, program de facto na výstup vypíše znak A.

4) Program, ktorý vypíše text AHOJ (resp. uloží do výstupného prúdu bajtov reťazec bajtov 0x41 0x48 0x4F 0x4A - v decimálnej reprezentácii 65 72 79 74).

```
+++++ +++
[
  >+++++ +++
  >+++++ +++
  >+++++ +++
  >+++++ +++
  <<<<-
]
>+W
>+++++ +++W
>+++++ +++++ +++++W
>+++++ +++++W
```