MiscDev 002

by gynvael.coldwind//vx

002. Interpreter i kompilator BF

Cel główny

Stworzyć **interpreter** i **kompilator** ezoterycznego języka programowania Brainf*ck.

http://pl.wikipedia.org/wiki/Brainfuck

Cel główny

Interpreter
Samodzielny program interpretujący (wykonujący) skrypt BF.

Binder
Program łączony interpreter ze skryptem, tworzący plik wykonywalny

Kompilator
Program kompilujący skrypt na natywny plik wykonywalny

Kompilator optymalizujący
Jak wyżej, przy czym wynik jest dodatkowy zoptymalizowany

Język Brainf*ck

Pamięc

Tablica (zazwyczaj) 30k bajtów,
zainicjalizowana zerami

Język Brainf*ck

Wskaźnik

Jeden, nienazwany, wskazuje na element tablicy, można go „przesuwać”, oraz operować na elemencie tablicy na który wskazuje

Język Brainf*ck

Instrukcje

< >	przesuń wskaźnik w lewo/prawo
+ -	zwiększ/zmniejsz (o 1) wskazywany element
.	wyświetl znak o kodzie ASCII ze wskazywanego elem.
,	wczytaj kod znaku ASCII do wskazywanego elem.
[	skacze za dopełniający] jeżeli wskazywanym elementem jest 0
]	skacze go dopełniającego [

0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	...

Język Brainf*ck

Instrukcje

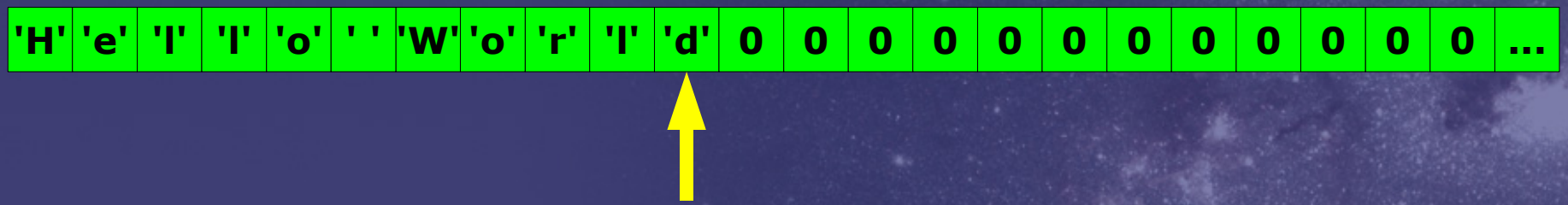
`< >`	`ptr++`	oraz	`ptr--`
`+ -`	`(*ptr)++`	oraz	`*(ptr--)`
`.`	`putchar(*ptr)`		
`,`	`*ptr = getchar()`		
`[`	`if(!*p) goto ]`	lub	`while(*p) {`
`]`	`goto [`	lub	`}`

Język Brainf*ck

Hello World!

Chcemy wypisać „Hello World!” w najprostszy możliwy sposób

(póki co skorzystamy z bfc.exe)



Interpreter

Ogólny zarys:

1. Dostaje nazwę pliku jako parametr
2. Wczytuje cały plik do pamięci (szybkość)
3. Znak po znaku wykonuje skrypt
4. Ignoruje znaki których nie zna (komentarze)

Interpreter

Plan działania:

1. Sprawdź ilość argumentów
2. Wczytaj plik
3. Wywołaj funkcję wykonującą skrypt
4. Wyjdź

Interpreter

Jak wykonać...

+ - < >

trywialne operacje na indeksie tablicy lub na elemencie pod tym indeksem

. ,

kwestia wywołania putchar i getchar

[ ]

tutaj trochę gorzej - trzeba poszukać w przód lub w tył nawias do pary - czyli miejsce skoku

Interpreter

Do dzieła!

Interpreter

Optymalizacja

Interpretacja dwuprzebiegowa

1. Przeskanuj źródło w celu znalezienia [ ], zapisz ich pozycję oraz poziom zagłębienia
2. Wykonaj skrypt korzystając z powyższych „notatek” do skoków

Optymalizacja skryptu

Przebieg zerowy:

0. Zamień wszystkie + < > - , . na pary:
OPERACJA, ILOŚĆ

Binder

Ogólny zarys:

1. Ma łączyć interpreter+skrypt do pliku .exe
2. Ma wykorzystywać kod interpretera
3. Powinien być w miarę najprostszy
4. Możemy założyć że skrypt nie będzie większy niż 64kb

Binder

Zasada działania:
(KISS)

1. Interpreter/binder ma mieć tablicę na skrypt, 64kb
2. Jeżeli tablica jest pusta, program jest binderem
3. Jeśli jest niepusta, program jest interpreterem i interpretuje skrypt z tablicy
4. Binder kopiuje sam siebie i wrzuca skrypt w tablicę w swojej kopii

Binder

Do dzieła!

Kompilator

Ogólny zarys:

1. Ma kompilować skrypt do pliku .exe
2. Ma tworzyć pliki wykonywalne typu PE
3. Plik PE powinien się prawidłowo wykonywać :)

Kompilator

Wykonanie:

1. Rama pliku PE:
- nagłówek MZ
- nagłówek PE
- importy (putchar i getchar z msvcrt.dll)
- sekcje (dwie sekcje, .text i .data)

2. Tłumaczenie BF na assembler?

```
+ - < >   inc, dec
, .       push+call
[ ]       jmp, call, ret
```

Kompilator

Do dzieła!

Kompilator optymalizujący

Optymalizacja

Łączenie instrukcji + - < >

Łączenie instrukcji , .

Do dzieła!

Podsumowanie

Interpretery
a
Bindery
a
Kompilatory

Podsumowanie

Kompilator kompilujący do assemblera
umożliwia to włączenie kodu w BF do projektu w C/C++

Większość kompilatorów języków kompilowanych do kodu maszynowego tworzy pliki assemblera, a nie pliki wykonywalne

Assemblera kompiluje/assembluje assembler :)

A pliki obiektowe łączy linker, który tworzy plik wykonywalny

Dziękuje za uwagę :)

http://re.coldwind.pl/
http://gynvael.coldwind.pl/