INTERPRETER JĘZYKA APO

Prezentowany program jest interpreterem fikcyjnego języka asemblerowego APO zaimplementowanym w języku Java 7.
Interpreter jest uruchamiany w konsoli / wierszu polecenia i tamże działa.

Program składa się z klasy odpowiedzialnej za interakcję z użytkownikiem oraz pakietu zawierającego właściwy interpreter.
Interpreter dokonuje alokacji pamięci, parsowania programu *.apo* oraz wykonania tegoż programu.

Alokacja pamięci polega na zarezerwowaniu określonego obszaru pamięci (domyślnie 1 kB, ale użytkownik sam może określić rozmiar) przeznaczonego na potrzeby działania programu.
Parsowanie polega na wczytaniu kolejnych linii programu *.apo* i sformatowaniu ich w postaci listy rozkazów.
Wykonanie programu polega na pobieraniu kolejnych instrukcji z listy rozkazów i realizacji operacji zadanych przez program.
Interpreter informuje również użytkownika o błędach podczas wykonania programu.

Po zakończeniu działania programu interpreter zamyka się automatycznie.

INSTRUKCJA OBSŁUGI INTERPRETERA JĘZYKA APO:

Interpreter należy uruchomić w środowisku obsługującym Java 7.

Aby uruchomić interpreter należy otworzyć konsolę / wiersz polecenia.
Następnie wejść do folderu, w którym znajduje się program (klasa Interpreter).

Wpisać polecenie 'javac Interpreter.java'.
Program jest gotowy do działania.

W celu wykonania programu napisanego w języku APO należy wpisać polecenie *'java Interpreter nazwa.apo'*.
Podczas działania programu interpretowany może być tylko jeden program.

Oprócz wykonania programu możliwa jest opcja alokacji x kB pamięci.
W tym celu należy wpisać polecenie *'java Interpreter nazwa.apo x'*.
Domyślnie interpreter alokuje 1 kB pamięci.

Należy pamiętać, że programy *.apo* powinny znajdować się w tym samym folderze co plik *Interpreter.java*.

Po wydaniu polecenia interpretacji użytkownik zostanie poinformowany o rozmiarze zaalokowanej pamięci:
memory>> 1 kB allocated
oraz o wyniku parsowania:
parsing>> done

Następnie program dokona właściwej interpretacji.

W przypadku napotkania na operację wejścia (RDINT bądź RDCHR) zostanie wyświetlony komunikat o możliwości dokonania wpisu:
input>>

W przypadku wystąpienia wyjątku w programie *.apo* komunikat o wyjątku zostanie poprzedzony komendą:
error>>

Po pomyślnym zakończeniu pracy interpretera bądź wystąpieniu błędu program automatycznie kończy działanie.

MIŁEJ PRACY!

INSTRUKCJA OBSŁUGI JĘZYKA APO

Podstawowe informacje:

Każda instrukcja powinna być umieszczona w osobnym wierszu.
Nazwy instrukcji piszemy WIELKIMI LIETRAMI.
Nazwy zmiennych i etykiet piszemy małymi literami.
Argumenty dołączane do instrukcji oddzielamy od siebie dowolnym ciągiem białych znaków.
Zmienna 'zero' jest zmienną specjalną - przechowuje wartość 0 i nie można jej modyfikować.
Komentarze poprzedzamy znakiem '#'.

Zestaw instrukcji:

1. instrukcje arytmetyczno-logiczne
ADD x y z - do zmiennej x przypisz sumę wartości zmiennych y i z
ADDI x y c - do zmiennej x przypisz sumę wartości zmiennej y i stałej c
SUB x y z - do zmiennej x przypisz różnicę wartości zmiennych y i z
SUBI x y c - do zmiennej x przypisz różnicę wartości zmiennej y i stałej c
MUL x y z - do zmiennej x przypisz iloczyn wartości zmiennych y i z
MULI x y c - do zmiennej x przypisz iloczyn wartości zmiennej y i stałej c
DIV x y z - do zmiennej x przypisz iloraz wartości zmiennych y i z
DIVI x y c - do zmiennej x przypisz iloraz wartości zmiennej y i stałej c
SHLT x y c - do zmiennej x przypisz wynik przesunięcia bitowego w lewo wartości zmiennej y o c bitów
SHRT x y c - do zmiennej x przypisz wynik przesunięcia bitowego w prawo wartości zmiennej y o c bitów
SHRS x y c - do zmiennej x przypisz wynik przesunięcia bitowego w prawo z zachowaniem znaku wartości zmiennej y o c bitów
AND x y z - do zmiennej x przypisz wynik operacji bitowej AND na wartościach zmiennych y i z
ANDI x y c - do zmiennej x przypisz wynik operacji bitowej AND na wartości zmiennej y i stałej c
OR x y z - do zmiennej x przypisz wynik operacji bitowej OR na wartościach zmiennych y i z
ORI x y c - do zmiennej x przypisz wynik operacji bitowej OR na wartości zmiennej y i stałej c
XOR x y z - do zmiennej x przypisz wynik operacji bitowej XOR na wartościach zmiennych y i z
XORI x y c - do zmiennej x przypisz wynik operacji bitowej XOR na wartości zmiennej y i stałej c
NAND x y z - do zmiennej x przypisz wynik operacji bitowej NAND na wartościach zmiennych y i z
NOR x y z - do zmiennej x przypisz wynik operacji bitowej NOR na wartościach zmiennych y i z

2. instrukcje skoku
JUMP e - skocz bezwarunkowo do etykiety e
JPEQ x y e - jeżeli wartości zmiennych x i y są równe, to skocz do etykiety e
JPNE x y e - jeżeli wartości zmiennych x i y są różne, to skocz do etykiety e
JPLT x y e - jeżeli wartość zmiennej x jest mniejsza od wartości zmiennej y, to skocz do etykiety e
JPGT x y e - jeżeli wartość zmiennej x jest większa od wartości zmiennej y, to skocz do etykiety e

3. instrukcje działające na pamięci
LDW x a - załaduj do zmiennej x słowo długości 32 bit spod adresu a (adres musi być liczbą podzielną przez 4)
LDB x a - załaduj do zmiennej x bajt pamięci spod adresu a
STW x a - wartość zmiennej x załaduj jako pojedyncze słowo długości 32 bit do adresu a (adres musi być liczbą podzielną przez 4)
STB x a - wartość zmiennej x załaduj jako pojedynczy bajt do adresu a (wartość zmiennej x musi mieścić się w jednym bajcie)

4. instrukcje wejścia/wyjścia
PTINT x - wypisz wartość zmiennej x jako liczbę całkowitą
PTCHR x - wypisz wartość zmiennej x jako znak
RDINT x - wczytaj wartość zmiennej x jako liczbę całkowitą
RDCHR x - wczytaj wartość zmiennej x jako znak