

Interpreter pseudoassemblera

SPRAWOZDANIE

Łukasz Basiński

Spis treści

Instrukcja obsługi interpretera.	2
Struktura programu. Szczegóły działania interpretera.	4
Rejestry	4
Program.....	4
Odwołania do pamięci i skoki	4
Dostępne instrukcje.	5

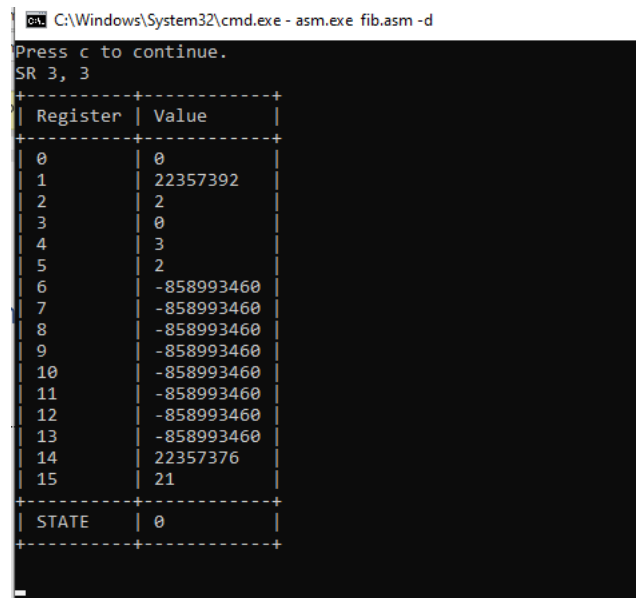
Instrukcja obsługi interpretera.

Interpreter uruchamiamy komendą: `asm.exe <filename.asm> (-d)`

Parametr `-d` określa czy chcemy uruchomić interpreter w trybie debugowania – wtedy będziemy mogli swobodnie przeglądać zawartość rejestrów, pamięci oraz aktualnie wykonywaną instrukcję. Jeśli nie zostanie dodany ten parametr to do pamięci, instrukcji i rejestrów będziemy mieli dostęp dopiero po wykonaniu całego programu.

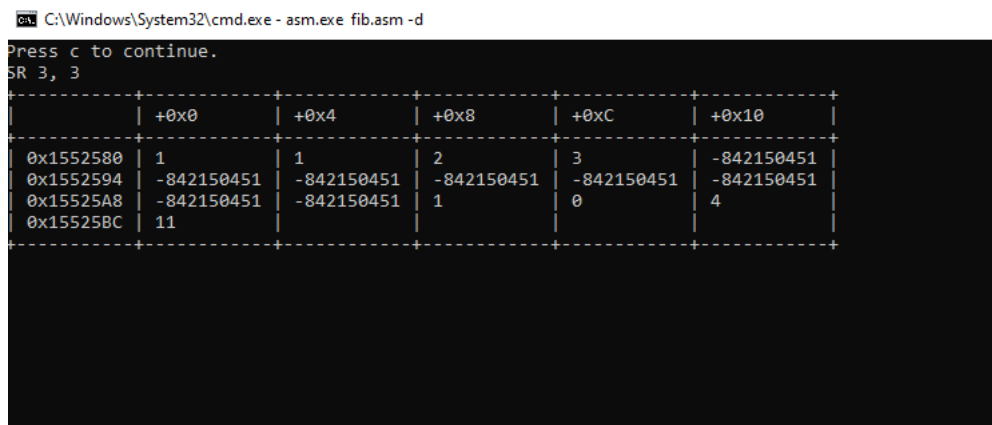
Użytkownik podczas pracy w trybie debugowania ma możliwość wykorzystania jednego z 3 trybów:

1. Przeglądu rejestru – wyświetlana jest zawartość wszystkich rejestrów (łącznie z rejestrem stanu)



Register	Value
0	0
1	22357392
2	2
3	0
4	3
5	2
6	-858993460
7	-858993460
8	-858993460
9	-858993460
10	-858993460
11	-858993460
12	-858993460
13	-858993460
14	22357376
15	21
STATE	0

2. Przegląd pamięci – wyświetlanie zawartości wszystkich zadeklarowanych komórek pamięci



	+0x0	+0x4	+0x8	+0xC	+0x10
0x1552580	1	1	2	3	-842150451
0x1552594	-842150451	-842150451	-842150451	-842150451	-842150451
0x15525A8	-842150451	-842150451	1	0	4
0x15525BC	11				

3. Przegląd instrukcji wraz z podkreśleniem aktualnie wykonywanej

```
C:\Windows\System32\cmd.exe - asm.exe fib.asm -d
Press c to continue.
1  TAB      DC      12*INTEGER(1)
2  JEDEN    DC      INTEGER(1)
3  ZERO     DC      INTEGER(0)
4  CZTERY   DC      INTEGER(4)
5  DZIESIEC DC      INTEGER(11)
6          LA      1          TAB
7          L       0          ZERO
8          L       4          JEDEN
9          L       5          JEDEN
10         ST      4          0(1)
11         A       1          CZTERY
12         ST      5          0(1)
13         A       1          CZTERY
14         L       2          JEDEN
15         SR      3          3
16 FIB      AR      3          4
17         AR      3          5
18         ST      3          0(1)
19         A       1          CZTERY
20         LR      5          4
21         LR      4          3
22         SR      3          3
23         A       2          JEDEN
24         C       2          DZIESIEC
25         JN      FIB
26         PRT     TAB
```

Między ekranami można się przełączać wciskając liczby odpowiednio 1, 2 lub 3.

Aby przejść do następnej instrukcji należy wcisnąć klawisz *c*.

Aby wyjść z interpretera po zakończeniu wykonywania programu należy wprowadzić *q* i wcisnąć ENTER.

Struktura programu. Szczegóły działania interpretera.

Rejestry

Przy implementacji można korzystać z 16 rejestrów, w których można zapisać liczbę całkowitą (ze znakiem, 4-bajtową). Do użycia jednak zaleca się korzystanie z pierwszych 14 (od 0 do 13), ponieważ rejestr 14 zawiera początek sekcji danych, a 15 numer aktualnie wykonywanej instrukcji. Interpreter zawiera jeszcze jeden rejestr – rejestr stanu programu – który nie jest dostępny dla użytkownika. Przechowuje on wynik ostatniej operacji arytmetycznej lub porównania. Przyjmuje on wartości: 0, w przypadku, gdy wynikiem ostatniej operacji było zero, 1 gdy wynik ostatniej operacji był dodatni, 2 gdy wynik ostatniej operacji był ujemny lub 3 gdy wystąpił błąd.

Program

Podczas implementacji przyjęto następującą strukturę wprowadzanego programu:

- 1) Na początku deklarowane są wszystkie zmienne potrzebne w dalszej części programu. Jest to wymagane, ponieważ nie ma możliwości dodania pamięci w trakcie wykonania.
- 2) Po sekcji danych zapisywane są wszystkie instrukcje, zgodnie z kolejnością wykonywania.

Odwołania do pamięci i skoki

Do pamięci zadeklarowanej w sekcji danych można odwoływać się poprzez:

- 1) Podawanie ich etykiet do odpowiednich instrukcji (np. A 1, <słowo>).
- 2) Wczytanie do rejestru adresu komórki pamięci (w przypadku alokacji więcej niż jednego elementu pod jednym słowem, wczytany zostanie adres pierwszego elementu). Następnie możemy się odwołać do takiej pamięci podając jako argument <przesunięcie>(<rejestr>). Np. W rejestrze 1 mamy załadowany adres tablicy TAB. Wtedy do jej pierwszego elementu odwołamy się poprzez 0(1), natomiast do drugiego 4(1) (4 ponieważ integer jest 4-bajtowy). Do drugiego elementu możemy się odwołać dodając 4 do rejestru 1 i odwołując się poprzez 0(1).

UWAGA: interpreter domyślnie nie zeruje wartości rejestrów ani pamięci!

W instrukcjach skoku należy podać etykietę zadeklarowaną w programie – nie ma to znaczenie czy jest on przed czy po aktualnie wykonywanej instrukcji.

Do wyświetlania tabel użyto biblioteki (OpenSource) libfort.

Interpreter został przetestowany na systemie operacyjnym Windows 10 oraz kompilatorze Visual Studio 2017.

Dostępne instrukcje.

Zaimplementowany pseudo-assembler zawiera następujące instrukcje¹:

Typ operacji	Sposób interpretacji	Opis
Dodawanie	(etykieta) A <rejestr>, <pamięć>	Dodaje do rejestru wartość podanej komórki pamięci.
	(etykieta) AR <rejestr1>, <rejestr2>	Dodaje do rejestru1 wartość rejestru2. Poprzednia wartość rejestru1 jest tracona.
Odejmowanie	(etykieta) S <rejestr>, <pamięć>	Odejmuje od rejestru wartość podanej komórki pamięci.
	(etykieta) SR <rejestr1>, <rejestr2>	Odejmuje od rejestru1 wartość rejestru2. Poprzednia wartość rejestru1 jest tracona.
Mnożenie	(etykieta) M <rejestr>, <pamięć>	Mnoży wartość rejestru przez wartość podanej komórki pamięci.
	(etykieta) MR <rejestr1>, <rejestr2>	Mnoży wartość rejestru1 przez wartość rejestru 2. Poprzednia wartość rejestru1 jest tracona.
Dzielenie	(etykieta) D <rejestr>, <pamięć>	Dzieli wartość rejestru przez wartość podanej komórki pamięci. Jeśli wartość komórki pamięci wynosi 0, to zachowana zostaje poprzednia wartość rejestru i ustawiana zostaje flaga rejestru stanu (ustawiona zostaje 3).
	(etykieta) DR <rejestr1>, <rejestr2>	Dzieli wartość rejestru1 przez wartość rejestru 2. Poprzednia wartość rejestru1 jest tracona. Jeśli wartość komórki pamięci wynosi 0, to zachowana zostaje poprzednia wartość rejestru i ustawiana zostaje flaga rejestru stanu (ustawiona zostaje 3).
Porównanie	(etykieta) C <rejestr>, <pamięć>	Porównuje wartość rejestru z wartością komórki pamięci. Jeśli wartość rejestru jest większa od wartości komórki pamięci to rejestr stanu jest ustawiany na 1, jeśli są równe na 0, jeśli jest mniejsza to na 2.
	(etykieta) CR <rejestr1>, <rejestr2>	Porównuje wartość rejestru1 z wartością rejestru2. Jeśli wartość rejestru1 jest większa od wartości rejestru2i to rejestr stanu jest ustawiany na 1, jeśli są równe na 0, jeśli jest mniejsza to na 2.

¹ Pola opcjonalne ujęto w nawiasach okrągłych, natomiast wymagane w nawiasach trójkątnych.

Skoki	(etykieta) J <etykieta2>	Skok bezwarunkowy do etykiety2.
	(etykieta) JZ <etykieta2>	Skok warunkowy do etykiety2, jeśli rejestr stanu jest ustawiony na 0.
	(etykieta) JP <etykieta2>	Skok warunkowy do etykiety2, jeśli rejestr stanu jest ustawiony na 1 (wynik ostatniej operacji był dodatni).
	(etykieta) JN <etykieta2>	Skok warunkowy do etykiety2, jeśli rejestr stanu jest ustawiony na 2 (wynik ostatniej operacji był ujemny).
Odwołania do pamięci	(etykieta) L <rejestr>, <pamięć>	Wczytuje wartość komórki pamięci do rejestru.
	(etykieta) LA <rejestr>, <pamięć>	Wczytuje adres komórki pamięci do rejestru (w przypadku alokacji więcej niż jednego elementu pod jednym słowem, wczytany zostanie adres pierwszego elementu).
	(etykieta) LR <rejestr1>, <rejestr2>	Wczytuje wartość rejestru2 do rejestru1. Poprzednia wartość rejestru1 jest tracona.
	(etykieta) ST <rejestr>, <pamięć>	Zapisuje wartość rejestru do komórki pamięci.
Alokacja pamięci	<słowo> DC INTEGER(<wartość>)	Alokuje 4-bajtową komórkę pamięci i przypisuje do niej wartość
	<słowo> DC <ile>*INTEGER	Alokuje <ile> komórek pamięci (każda 4 bajtowa).
	<słowo> DS INTEGER	Alokuje 4-bajtową komórkę pamięci
	<słowo> DS <ile>*INTEGER	Alokuje <ile> komórek pamięci (każda 4 bajtowa).
Inne	(etykieta) PRT <słowo>	Wyświetla wartość(wartości) podanej komórki pamięci.
	(etykieta) RD <słowo>	Wczytuje wartość(wartości) do podanej komórki pamięci.

UWAGA: rejestr stanu zmieniają także operacje arytmetyczne.