środa, 13 stycznia 2016

Wiktor Adamski

Zadanie dodatkowe na pracownię SO – sprawozdanie

„Maszyna niezawodna nie może być inteligentna.”
Alan Turing

Treść zadania

Napisz program emulujący maszynę RAM.

Specyfikacja maszyny RAM

Maszyna ram to abstrakcyjny model uproszczonego komputera, składającego się z taśm (wejściowej i wyjściowej), komórek pamięci zdolnych przechowywać liczby całkowite, oraz licznika i listy rozkazów do wykonania. Dla potrzeb programu przyjmujemy, że liczby, na których operujemy mieszczą się w przedziale od -2 147 483 648 do 2 147 483 647 (zakres liczb `int32`), a komórki pamięci są indeksowane liczbami naturalnymi, gdzie wyszczególniamy komórkę o adresie 0, jako akumulator wykorzystywany w obliczeniach. Pojedynczy rozkaz ma następującą budowę:

```
[:etykieta] rozkaz [etykieta|argument|=argument|^argument]
```

Gdzie

- `etykieta` – ciąg znaków alfanumerycznych wykorzystywany w rozkazach skoku
- `rozkaz` – jeden z rozkazów z poniższej tabeli
- `argument` – liczba całkowita (domyślnie adres komórki pamięci, poprzedzona ^ oznacza adresowanie pośrednie, = stałą)

W poniższej tabeli `P[n]` oznacza komórkę (zawartość komórki) pamięci o adresie n, `'/'` dzielenie całkowitoliczbowe, a `'%'` operację modulo.

Rozkaz	Argument	Efekt
READ	n	`P[n] :=` wartość z taśmy wejściowej
	^n	`P[P[n]] :=` wartość z taśmy wejściowej
WRITE	n	Zapisanie `P[n]` na taśmie wyjściowej.
	=n	Zapisanie `n` na taśmie wyjściowej.
	^n	Zapisanie `P[P[n]]` na taśmie wyjściowej.
LOAD	n	`P[0] := P[n]`
	=n	`P[0] := n`
	^n	`P[0] := P[P[n]]`
STORE	n	`P[n] := P[0]`
	^n	`P[P[n]] := P[0]`
ADD	n	`P[0] := P[0] + P[n]`
	=n	`P[0] := P[0] + n`
	^n	`P[0] := P[0] + P[P[n]]`
SUB	n	`P[0] := P[0] - P[n]`
	=n	`P[0] := P[0] - n`
	^n	`P[0] := P[0] - P[P[n]]`
MULT	n	`P[0] := P[0] * P[n]`
	=n	`P[0] := P[0] * n`
	^n	`P[0] := P[0] * P[P[n]]`

DIV	n	P[0] := P[0] / P[n]
	=n	P[0] := P[0] / n
	^n	P[0] := P[0] / P[P[n]]
MOD	n	P[0] := P[0] % P[n]
	=n	P[0] := P[0] % n
	^n	P[0] := P[0] % P[P[n]]
JUMP	etykieta	Przekazanie sterowania do instrukcji oznaczonej etykieta.
JZERO	etykieta	Przekazanie sterowania do instrukcji oznaczonej etykieta, jeśli P[0] = 0.
JGTZ	etykieta	Przekazanie sterowania do instrukcji oznaczonej etykieta, jeśli P[0] > 0.
JLTZ	etykieta	Przekazanie sterowania do instrukcji oznaczonej etykieta, jeśli P[0] < 0.
STOP		Zakończenie wykonywania programu.

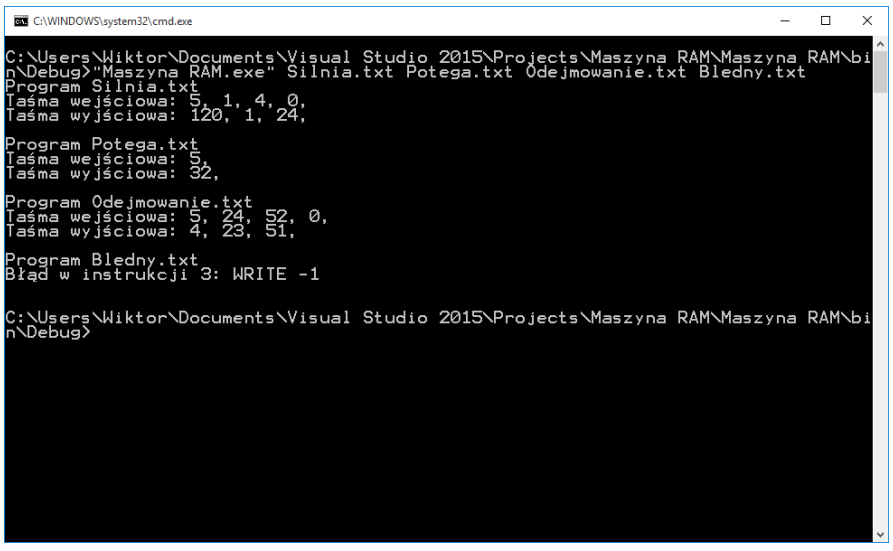
Program, który nie jest zakończony instrukcją STOP nadal wykona się poprawnie. Jeśli któraś z instrukcji wykona niedozwoloną operację (np. odwoła się do ujemnego indeksu, czy nieistniejącej etykiety) zostanie przerwany, wraz z informacją, w której linijce kodu wystąpił błąd.

Opis programu

Program został napisany w języku C#. Obiektowość tegoż języka ułatwiła modularyzację kodu – każdy z rozkazów jest osobną klasą. Ułatwia to rozszerzanie maszyny o dodatkowe rozkazy. W argumentach wywołania należy napisać nazwy plików zawierające programy maszyny ram, po jednym programie na plik. W każdym pliku, dwie pierwsze linijki muszą zawierać odpowiednio liczbę komórek pamięci wymaganą do działania programu, oraz zawartość taśmy wejściowej, w postaci liczb oddzielonych znakami spacji. W kodzie programu można umieszczać komentarze, po napotkaniu znaku '//' cała pozostała zawartość linijki zostanie pominięta przy czytaniu programu. Jeśli program zakończy się poprawnie, zawartość obu taśm zostanie wypisana na ekranie.

Testowanie programu

Razem z programem dołączyłem zestaw trzech programów maszyny ram (pliki Odejmowanie.txt, Silnia.txt, Potega.txt), które służyły do sprawdzania działania zarówno czytania programów, jak i poprawności implementacji maszyny. Osobno zostało sprawdzone działanie każdego z rozkazów, a także reakcja maszyny na błędy w programie.



Uwagi praktyczne

Ponieważ programy pisane w C# są kompilowane do postaci kodu pośredniego, interpretowanego przez platformę .NET, dostarczona wersja skompilowana (plik `Maszyna RAM.exe`), powinna uruchomić się na każdym komputerze z zainstalowaną ww. platformą (praktycznie każdy komputer z zainstalowanym systemem Windows, od wersji XP włącznie). W przeciwnym przypadku wystarczy skompilować plik `Program.cs`, który zawiera punkt wejścia dla całego programu.