Igor Marzyński, Jan Mikicki

**Laboratorium Architektury Systemów Komputerowych**

Zadanie 3

Celem zadania było stworzenie aplikacji utylizującej interfejs graficzny do interakcji z uzytkownikiem. Elementy grafiki miały być w pewnym stopniu konfigurowalne (np. zmiana motywu). Przeznaczenie aplikacji nie było wyspecyfikowane, wybrano zatem kalkulator z zegarem podobny np. do kalkulatora systemowego Windows.

Program został napisany w języku C# przy pomocy interfejsu programowania graficznych aplikacji Windows Forms i środowiska Visual Studio.

1. Założenia szczegółowe

Podczas realizacji zadania należało ustalić według jakich zasad będzie odbywała się interakcja z aplikacją i jakie ograniczenia są nałożone na program:

* każda instukcja zaczyna się od nowej linii
* instrukcja może być pisana małymi lub dużymi literami
* argumenty są oddzielane przecinkiem
* pierwszy argument to rejestr na którym wykonywana jest operacja
* drugi argument to rejestr z którego pobierana jest wartość (adresowanie rejestrowe) lub liczba (adresowanie bezpośrednie)
* dozwolone systemy liczbowe to:
  + dziesiętny - domyślny np. ‘24’
  + szesnastkowy – argument zaczynający się od ‘0x’ np. ‘0x1f’
  + dwójkowy – argument kończący się literą ‘b’ np. ‘10011b’
* nie używany jest kod znaku, tzn. liczba ujemna zostanie przekonwertowana na zasadzie kodu U2, zostanie jednak zinterpretowana jako liczba dodatnia: -32768 zostanie zapisane w rejestrze jako 32768
* liczba większa niż 16-bitowa zostanie obcięta do 16 najmłodszych bitów

2. Opis przyjętych rozwiązań programowych

* rejestry AX, BX, CX oraz DX zrealizowano jako instancje klasy ‘Register’ posiadającej zmienne ‘H’ i ‘L’, które przechowują wartości rejestrów składowych oraz pola ‘HL’, które przechowuje wartość całego rejestru
* wartość zmiennej ‘HL’ obliczana jest na podstawie wzoru:  
  this.HL = (this.H << 8) + this.L;  
  i jest przechowywana wewnętrznie jako integer
* zbyt duża liczba konwertowana jest na ciąg znaków w postaci binarnej, po czym 16 (lub 8 w przypadku rejestru składowego) najmłodszych bitów jest konwertowane z powrotem na liczę typu integer:   
   String bin = Convert.ToString(value, 2);  
   bin = bin.Substring(bin.Length - 16);  
   this.HL = Convert.ToInt32(bin, 2);
* kod pisanego programu wpisywany jest w pole TextBox i wczytywany w pętli linijka po linijce – następna linijka zostanie wczytana dopiero po wykonaniu poprzedniej instrukcji:  
    
   for (int i = 0; i < textBox1.Lines.Length; i++){  
   String line = textBox1.Lines[i];  
   if (!String.IsNullOrEmpty(line))   
   { …
* praca krokowa polega na sprawdzeniu dostępności linii i wykonaniu zawartej instrukcji jednorazowo po każdym naciśnięciu guzika (bez pętli) – aktualnie wykonywana instrukcja pojawia się w okienku na dole ekranu
* zapisywanie i wczytywanie zawartości TextBox’a do pliku odbywa się przy pomocy obiektów StreamWriter oraz StreamReader z przestrzeni System.IO
* przesyłanie i konwersja argumentów możliwa jest dzięki szeregowi funkcji klasy ‘Register’ jak i funkcji samego programu – z uwagi na wybraną przez nas realizację rejestrów składowych (jako zmienne w rejestrze głównym) często do ich obsługi wymagana jest oddzielna funkcja

3. Dodatkowe funkcje i dyskusja osiągniętych wyników

Poza spełnieniem założeń ogólnych i szczegółowych z punktu 1 dodano opcję przełączania widoku pomiędzy 3 obsługiwanymi systemami liczbowymi – dziesiętnym, szesnastkowym i binarnym. Widok przełącza się w dowolnej chwili za pomocą trzech przycisków wyboru.

Dodatkowo w menu paskowym, oprócz funkcji zapisywania i wczytywania można znaleźć opcję ‘Pomoc’, która wyświetli m.in. dozwolone operacje i kody mnemotechniczne.

Z uwagi na decyzję o nieużywaniu bitu znaku przedział poprawnie obsługiwanych liczb to   
<0, 65535> a nie < -32768, 32767> - jest to jednak kwestia umowna i może podlec zmianie w przyszłej wersji programu.

Jako że format używany wewnętrznie do przechowywania liczb to Int32 wpisywana liczba nie może przekroczyć wartości 2,147,483,647 – jakakolwiek liczba pomiędzy 65535 i 2,147,483,647 zostanie zaakceptowana i przycięta do ostatnich 8 lub 16 najmłodszych bitów.

Dzięki obsłudze wystąpień pustych linii w kodzie można zostawiać przerwy.