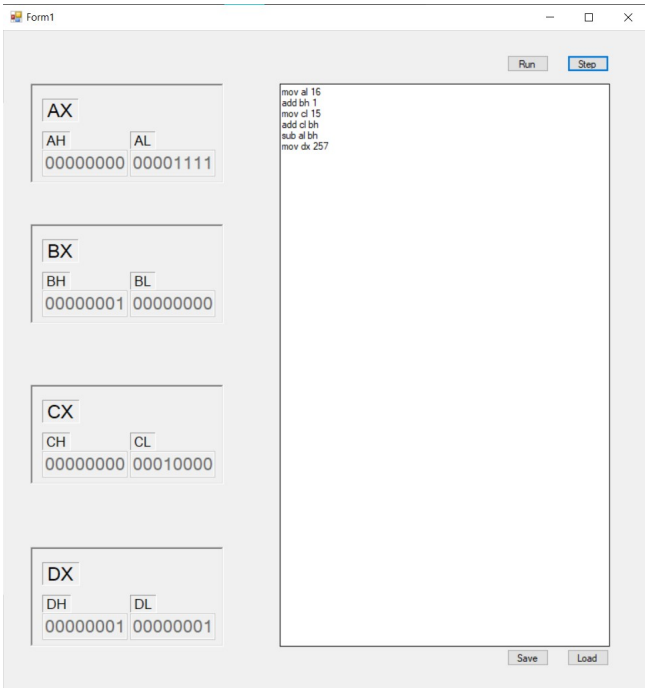
Sprawozdanie do zad. 4. w ramach laboratorium Organizacja Systemów Komputerowych

Paweł Szczepański (188640), Igor Szczyrbak (188661)

8 czerwca 2024

1 Założenia programu

Aplikacja służy do symulowania pracy mikroprocesora o czterech rejestrach ogólnego przeznaczenia, zezwalający na instrukcje MOV, ADD i SUB. W pole tekstowe po prawej użytkownik wpisuje kod separowany spacjami lub enterem. Przycisk "Run"powoduje wykonanie programu, a przycisk "Step"wykonanie kolejnego kroku programu. Przyciski "Save"oraz "Load"służą do odpowiednio zapisywania i wczytywania kodu.



Rysunek 1: Główny ekran aplikacji

2 Rozwiązania programowe

2.1 Wykonanie kodu

Tekst wpisany w pole tekstowe jest zczytywany do kolejki znaków (Queue<char>), która następnie jest opróżniania w pętli. Aplikacja traktuje spacje oraz entery jako separatory poleceń. Znaki z kolejki są zapisywane w zmiennej tekstowej (string), a po napotkaniu separatora zapisane słowo jest interpetowane i usuwane. Najpierw program rozpoznaje polecenie MOV, ADD lub SUB, następnie rejestr, następnie rejestr lub liczbę, po czym szuka nowego polecenia. Dane te są zapisywane w obiekcie klasy Order. Która wykonuje kolejne operacje na rejestrach.

```
//Zapis do kolejki code
foreach (char c in (codeBox.Text.ToLower() + " "))
    code.Enqueue(c);

int orderPart = 0; //Rozkaz, rejestr, czy drugi argument?
string word = ""; //Odczytywany rozkaz
char newChar;     //Odczytywany znak
Order order = new Order(registers);

while (code.Count() > 0)
{
    newChar = code.Dequeue();
    if (newChar == ' ' || newChar == '\n')
    {
        if (word != "")   //Czy słowo nie jest puste? na
                          //wypadek podwójnej spacji
        {
            switch (orderPart) //Rozkaz, rejestr, czy drugi
                               //argument?
            {
                case 0: //Rozkaz
                    switch (word) //Jaki rozkaz?
                    {
                        //case "mov", "sub" oraz "add"
                        default:
                            //Błąd!
                            return;
                            break;
                    }
                    break;

                case 1: //Rejestr
                    if (word[0] == 'a' ||'b' || 'c' || 'd')
                    {
                        //Zapisz nazwę rejestru
                        order.reg1Name = word[0];
```

```
                        switch (word[1])
                        {
                            //Przypadki x, h i l
                            default:
                                //Błąd!
                                return;
                                break;
                        }
                    }
                    else{
                        //Błąd!
                        return;}
                    break;

                case 2: //Drugi argument
                    //Podano rejestr
                    if (word[0] == 'a' ||'b' || 'c' || 'd')
                    {
                        order.mode = Order.modes.register;
                        order.reg2Name = word[0];
                        switch (word[1])
                        {
                            //Przypadki x, h i l
                            default:
                                //Błąd!
                                return;
                                break;
                        }
                    }
                    else   //Podano liczbę
                    {
                        order.mode = Order.modes.instant;
                        try {
                            order.instantValue =
                                Convert.ToInt32(word);}
                        catch {
                            //Błąd!
                            return;}
                    }
                    order.Perform();
                    break;
            }
            orderPart = (orderPart + 1) % 3;
            word = "";
        }
    }
    else
        word += newChar;
}
```

2.2 Zapisywanie i wczytywanie kodu

Kod jest zapisywany jako plik .txt. Do nawigacji w pamięci komputera wykorzystywane są obiekty SaveFileDialog oraz LoadFileDialog.

```
    private void save_Click(object sender, EventArgs e)
{
    Stream myStream;

    saveFileDialog1.Filter = "txt files (*.txt)|*.txt|All files
        (*.*)|*.*";
    saveFileDialog1.ShowDialog();

    if (saveFileDialog1.ShowDialog() == DialogResult.OK)
    {
        if ((myStream = saveFileDialog1.OpenFile()) != null)
        {
            string dataasstring = codeBox.Text; //your data
            byte[] info = new
                UTF8Encoding(true).GetBytes(dataasstring);
            myStream.Write(info, 0, info.Length);

            // writing data in bytes already
            byte[] data = new byte[] { 0x0 };
            myStream.Write(data, 0, data.Length);

            myStream.Close();
        }
    }
}
```

3 Dyskusja osiągniętych wyników

Program spełnia założenia projektowe: wykonuje zadane operacje, pozwala na zapisywanie/wczytywanie kodu i wykrywa niepoprawnie wpisany kod.