**Metode de proiectare a asamblorului aferent Procesorului Didactic (PD)**

1. **Specificatii de proiectare generale**

Asamblorul aferent simulatorului didactic trebuie proiectat in prima faza ca si o aplicatie de sine statatoare, capabila sa efectueze:

1. Selectarea fisierului ASM;
2. Parsarea acestui fisier;
3. Izolarea fiecarui element (token) relevant din codul sursa;
4. Obtinerea codului obiect (cod masina) pe baza token-urilor.

In momentul in care asamblorul functioneaza corect, acesta va fi integrat in simulatorul PD (aplicatia finala).

**Exemplu:**

Sa presupunem ca fisierul ASM contine urmatorul cod sursa:

MOV R2,R3

Eticheta: ADD R1,R2(7)

SUB (R5)15,4

DEC R7

JZ Eticheta

ADD R12,R9

Asamblorul va parsa aceasta secventa si va trebui sa obtina in pasul (c) urmatoarele token-uri:

MOV

R2

R3

Eticheta

ADD

R1

R2

7

SUB

R5

15

4

DEC

R7

JZ

Eticheta

ADD

R12

R9

Pe baza acestor token-uri aplicatia va trebui sa obtina codul masina afferent PD.

1. **Interfata aplicatiei date ca exemplu**

Aplicatia este scrisa in limbajul C#, folosind mediul de dezvoltare Micorsoft Visual Studio Express 2012 (disponibil gratuit pe internet). Aplicatia (fig. 2.1) realizeaza doar pasii (a), (b) si (c), pasul (d) nefiind implementat deoarece acest task trebuie sa ramana in sarcina studentilor.

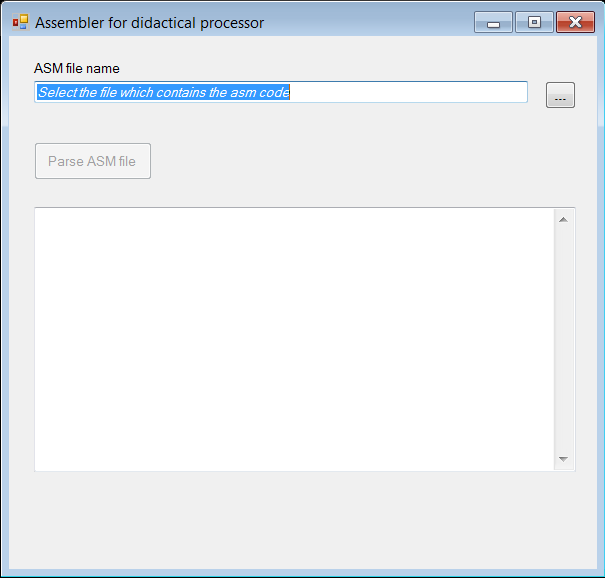


Fig. 2.1. Interfata asamblorului aferent PD.

Interfata este relativ simpla si intuitiva. Utilizatorul va trebui sa selecteze un fisier care contine cod de asamblare specific PD (fig. 2.2.).

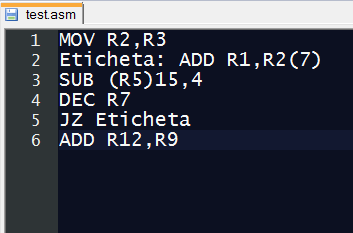


Fig. 2.2. Exemplu de cod scris in limbajul de asamblare al PD

Dupa selectia fisierului, utilizatorul va trebui sa actioneze butonul “Parse ASM file”, actiune care va conduce la obtinerea token-urilor aferente codului (fig. 2.3.).

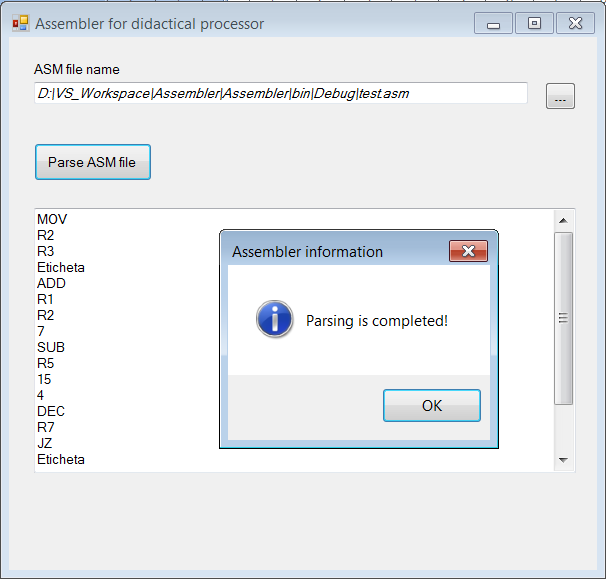


Fig. 2.3. Obtinerea token-urilor aferente codului ASM.

Codul sursa al acestei aplicatii este prezentat/comentat in anexa si este disponibil complet in fisierul Assembler\_VS2012.zip.

1. **Indicatii de proiectare**

In dezvoltarea asamblorului trebuie avut in vedere ca acest cod va reprezenta un modul (o clasa) in simulatorul final. Recomandarea este ca studentii sa aiba in vedere un model de clase bine definit. O abordare optima ar fi utilizarea unei singure clase care va fi instantiate in simulatorul final ca si un obiect numit Assembler.

**Anexa**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Project: Simulator of an microprogrammed Didactical Processor (DP)

\* Module: Assembler for DP

\* Filename: AssemblerMainForm.cs

\* Users: Students in Computer Science (3rd year of study)

\* Creator: Horia V. Caprita

\* Date: 10.03.2015

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.IO;

using Microsoft.VisualBasic.FileIO;

namespace Assembler

{

public partial class AssemblerMainForm : Form

{

private String filename;

public AssemblerMainForm()

{

InitializeComponent();

}

/\* Function used to obtain the ASM filename (\*.asm) \*/

private String getFileName(String filter)

{

try

{

/\* Local variable used to store the filename \*/

String fileNameWithPath = "";

/\* Instantiate an OpenFileDialog \*/

OpenFileDialog of = new OpenFileDialog();

/\* Set the filter \*/

of.Filter = filter;

/\* Get the working directory \*/

of.InitialDirectory = Path.GetFullPath("..\\Debug");

of.RestoreDirectory = true;

/\* Display the Open File dialog \*/

if (of.ShowDialog() == DialogResult.OK)

{

/\* Get only the filename with full path \*/

fileNameWithPath = of.FileName;

/\* Get only the filename without path \*/

filename = of.SafeFileName;

}

/\* Return the filename with complete path \*/

return fileNameWithPath;

}

catch (Exception e)

{

MessageBox.Show(e.Message);

return null;

}

}

private void ParseFileButton\_Click(object sender, EventArgs e)

{

try

{

/\* local variable used for debugging only \*/

int lineCounter = 0;

/\* List which will store each token (element) read from ASM file \*/

List<String> asmElements = new List<String>();

/\* Create a parser object used for ASM file

REMEMBER: this parser can be used for all kind of text files!!!

\*/

TextFieldParser parser = new TextFieldParser(filename);

/\* Reinitialize the Text property of OutputTextBox \*/

OutputTextBox.Text = "";

/\* Define delimiters in ASM file \*/

String[] delimiters = {":",","," ","(",")"};

/\* Specify that the elements in ASM file are delimited by some characters \*/

parser.TextFieldType = FieldType.Delimited;

/\* Set-up the specified delimiters \*/

parser.SetDelimiters(delimiters);

/\* Parse the entire ASM file based on previous specifications\*/

while (!parser.EndOfData)

{

/\* Read an entire line in ASM file

and split this line in many strings delimited by delimiters \*/

string[] asmFields = parser.ReadFields();

/\* Store each string as a single element in the list

if this string is not empty \*/

foreach (string s in asmFields)

{

if (!s.Equals(""))

{

asmElements.Add(s);

}

}

/\* Counting the number of lines stored in ASM file \*/

lineCounter++;

}

/\* Close the parser \*/

parser.Close();

/\* If the file is empty, trigger a new exception which

in turn will display an error message \*/

if (lineCounter == 0)

{

Exception exc = new Exception("The ASM file is empty!");

throw exc;

}

else

{

/\* Display every token in OutputTextBox \*/

foreach (String s in asmElements)

{

OutputTextBox.Text += s + Environment.NewLine;

}

/\* Display an information about the process completion \*/

MessageBox.Show("Parsing is completed!", "Assembler information", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

catch (Exception exc)

{

MessageBox.Show(exc.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void OpenFileButton\_Click(object sender, EventArgs e)

{

try

{

/\* String used to be displayed in ASMFileTextBox \*/

String filename = "";

/\* Reinitialize the Text property of OutputTextBox \*/

OutputTextBox.Text = "";

/\* Take the filename selected by user \*/

filename = getFileName("ASM file for didactical processor(\*.asm)|\*.asm");

/\* Display the filename in ASMFileTextBox \*/

ASMFileTextBox.Text = filename != null ? filename : ASMFileTextBox.Text;

/\* Enable/Disable the ParseFileButton depending of user choice \*/

if (!filename.Equals(""))

{

ParseFileButton.Enabled = true;

}

else

{

ParseFileButton.Enabled = false;

}

}

catch (Exception exc)

{

MessageBox.Show(exc.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}