

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА
Факультет прикладної математики та інформатики

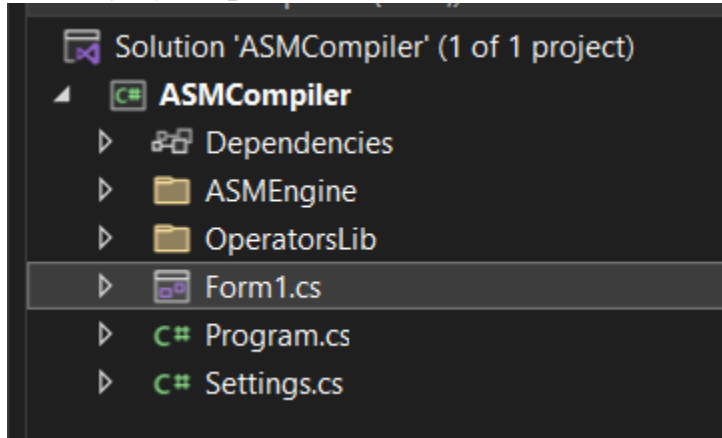
Кафедра дискретного аналізу

Операційні системи та системне програмування
Лабораторна робота №4

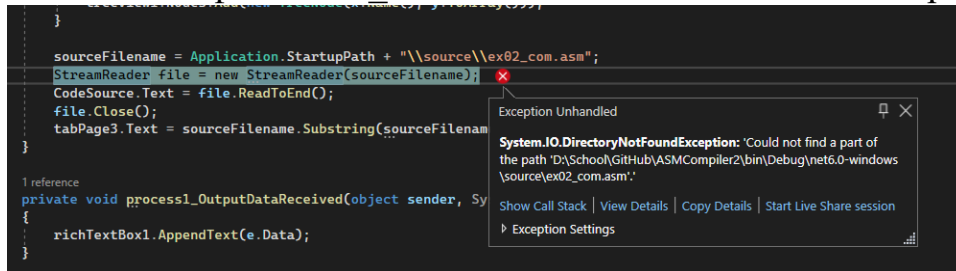
Виконав
Студент групи ПМІ-43
Заречанський Олексій
Викладач
Доц. Черняхівський Володимир

2023

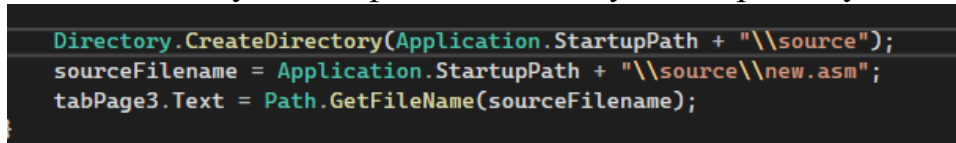
1. Я вирішив перевикористати зарефакторивши вже наявний source code даний в лабораторній роботі. З форм файлів видно що це віндовс форми, тому створю для них новий проект та рішення в візуал студію, поміщу туди файли.



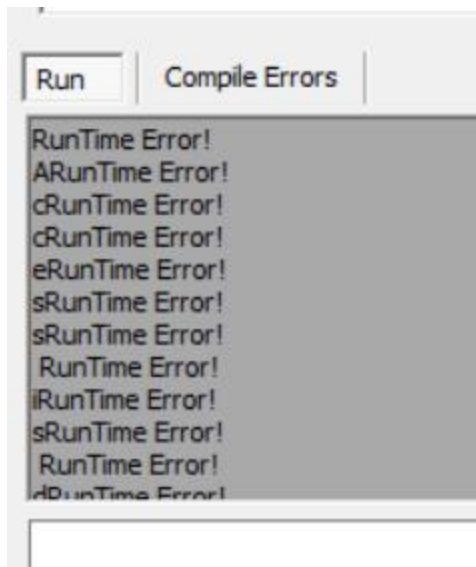
2. При спробі запуску бачу першу помилку що необхідно мати обов'язково файл ex02_com.asm в папці source в папці з програмою.



3. Не будемо зчитувати текст файлу якого може не бути, проте можемо дати йому нове ім'я щоб його можна було зберегти як новий файл. Також перепису спосіб отримання імені файлу на кращий. Так як папки в якій буде цей файл може не бути то пропишу її створення.



4. При спробі поклацати на кнопки отримую помилку в консоль, але вона виглядає неправильно, так ніби RunTime Error вставляється після кожного символу, перепису метод запису даних сюди.



Так як це відбувалось асинхронно то зміню методи на асинхроні, але які працюють набагато простіше, зчитуючи рідери до кінця одною асинхронною операцією.

```
1 reference
async Task ErrorBeginReadAsync()
{
    if(_errorReader is not null)
    {
        var text = await _errorReader.ReadToEndAsync();
        RunTextBox.AppendText("Runtime Error!\n");
        RunTextBox.AppendText(text);
    }
}

1 reference
async Task OutputBeginReadAsync()
{
    if(_outputReader is not null)
    {
        var text = await _outputReader.ReadToEndAsync();
        RunTextBox.AppendText(text);
        Debug.WriteLine(string.Format("Result of running file {0}:\n{1}", _sourceFilename, text));
    }
}
```

Відповідно для уникнення попереджень що я не очікую кінця виконання цього, при їх виклику я буду показувати що відкидаю результати

```
// Begin async read on standard error and output
_ = ErrorBeginReadAsync();
_ = OutputBeginReadAsync();
```

5. Як вже можна було побачити на попередніх скріншотах я також перейменував більшість об'єктів на формі, для простішого

орієнтування в них, ось як приклад декілька нових назв

```
CompileAndRunButton System.Windows.Forms.Button  
CompileButton System.Windows.Forms.Button  
CompileErrorsTextBox System.Windows.Forms.RichTextBox  
CompileResults System.Windows.Forms.DataGridView
```

6. Після переписання методу запису в текстовий результат стали непотрібні оці змінні, витру їх

```
StreamWriter inputWriter = null;  
byte[] errorBuffer = new byte[1];  
byte[] outputBuffer = new byte[1];  
StringBuilder outputString = new StringBuilder();
```

7. Перенесу решту змінних файлу над конструктор, дам їм модифікатори доступу, та перейменую згідно з конвенціями

```
private Process? _cmdProcess;  
private StreamReader? _errorReader;  
private StreamReader? _outputReader;  
private StreamWriter? _inputWriter;  
  
private string? _sourceFilename;  
private string? _binFilename;  
private ProcessStartInfo _processStartInfo;
```

8. В файлі багато функцій з малої, хоча за конвенціями повинні бути з великої, перейменую

```
2 references  
private void build()  
{
```

```
2 references  
private void Build()  
{  
    CompileErrorsTextBox.Text = "";  
    CompileResultsTextBox.Text = "";
```

9. Поперейменовую автозгенеровані функції по типу button1_click на щось зрозуміліше

```
1 reference  
private void RunButtonClick(object sender, EventArgs e)  
{  
    Run();  
}
```

10. По всьому файлу заміню спосіб отримання назви файлу на кращий та зрозуміліший візуально

```
// binFilename = UserSettings.Default.WorkingDirectory + UserSettings.Default.DefaultFilename;  
binFilename = Settings.WorkDir + sourceFilename.Substring(sourceFilename.LastIndexOf("\\") + 1);  
binFilename = binFilename.Remove(binFilename.IndexOf('.'));  
  
_binFilename = Settings.WorkDir + Path.GetFileNameWithoutExtension(_sourceFilename);
```

11. Зроблю додаткові відступи між усіма блоками If, try, та циклів.

```
foreach (Operator x in Operators.DataBase)  
{  
    List<TreeNode> y = new List<TreeNode>();  
    foreach (Format z in x.RegistredFormats)  
    {  
        y.Add(new TreeNode(z.FormatLine));  
    }  
    treeView1.Nodes.Add(new TreeNode(x.Name(), y.ToArray()));  
}
```

```
foreach (Operator x in Operators.DataBase)  
{  
    List<TreeNode> y = new();  
  
    foreach (Format z in x.RegistredFormats)  
    {  
        y.Add(new TreeNode(z.FormatLine));  
    }  
  
    treeView1.Nodes.Add(new TreeNode(x.Name(), y.ToArray()));  
}
```

Також як видно у рядку з створенням нового списку я забрав непотрібні повторні задання типу змінних.

12. Так як кожного разу як ми пробуємо заранити файл створюється процес з тими самими параметрами окрім імені файлу, то я переніс задання цих параметрів в конструктор, а змінну в скоуп класу.

```
_processStartInfo = new("cmd.exe")  
{  
    CreateNoWindow = true,  
    ErrorDialog = false,  
    RedirectStandardError = true,  
    RedirectStandardInput = true,  
    RedirectStandardOutput = true,  
    UseShellExecute = false  
};
```

13. Через це що це єдиний рядок де задається binfilename, то після першого створення файлу, навіть вибравши інший файл, при спробі його зберегти буде записуватись файл з тим самим іменем, виправлю

це.

```
if (binFilename == null)
// binFilename = UserSettings.Default.WorkingDirectory + UserSettings.Default.DefaultFilename;
binFilename = Settings.WorkDir + sourceFilename.Substring(sourceFilename.LastIndexOf("\\") + 1);
binFilename = binFilename.Remove(binFilename.IndexOf('.'));
```

Тепер ім'я буде визначатись кожного разу залежно від імені вибраного файлу.

```
_binFilename = Settings.WorkDir + Path.GetFileNameWithoutExtension(_sourceFilename);
```

14. Позбудує можливих NullReferenceException та заберу попередження з аналізатору коду.

```
if(_sourceFilename is not null)
{
    File.WriteAllText(_sourceFilename, CodeSource.Text);
    OpenedFileTab.Text = Path.GetFileName(_sourceFilename);
}
```

15. Для усунення попереджень з аналізатору коду заміню as на cast

```
..Text = CompileError:("\n",
..AppendText((e1.InnerException as CompileError).Line
rException as CompileError).LineNumber != -1)
CompileErrorsTextBox.AppendText(
    ((CompileError)e1.InnerException).LineNumber.ToString("0000")
    + " : "
    + ((CompileError)e1.InnerException).Message);
```

16. Заміню всі зчитування, записування в файл простішими для розуміння функціями, які працюють так само.

```
2 references
private void saveToFile(string filename)
{
    FileStream f = new FileStream(filename, FileMode.Create);
    StreamWriter file = new StreamWriter(f);
    file.Write(CodeSource.Text);
    file.Flush();
    file.Close();
    f.Close();
}

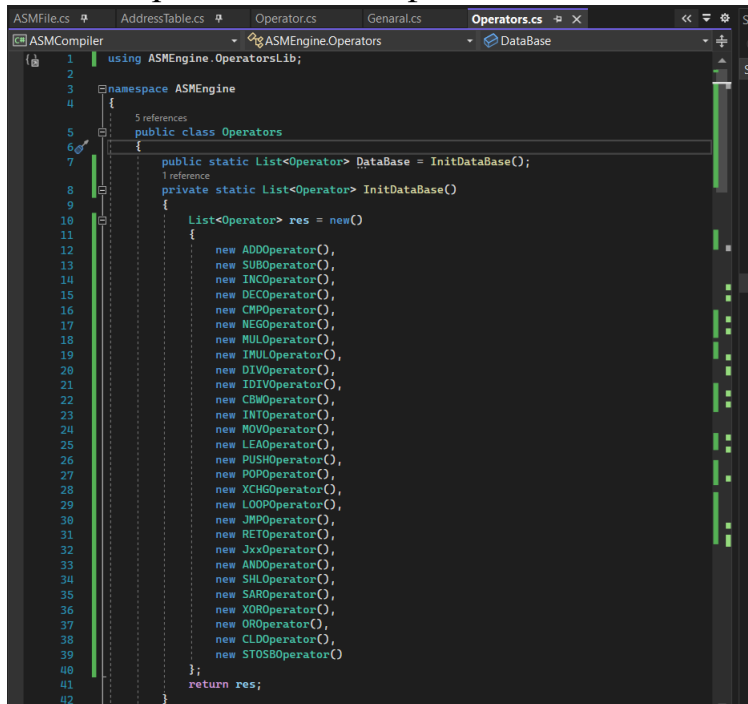
_sourceFilename = saveFileDialog1.FileName;
File.WriteAllText(_sourceFilename, CodeSource.Text);
OpenedFileTab.Text = Path.GetFileName(_sourceFilename);
```

17. Помічаю в формі процес, який не використовується, використовується процес що в методі Run, тому видаю цей зайвий компонент, атакож метод який мав записувати дані в текстбокс при отриманні даних з процесу.

```
OutputTabControl System.Windows.Forms.TabControl
process1 System.Diagnostics.Process
RunButton System.Windows.Forms.Button

0 references
private void process1_OutputDataReceived(object sender, DataReceivedEventArgs e)
{
    RunTextBox.AppendText(e.Data);
}
```

18. Це все що стосуються зміни в формі, проте були зміни і в інших файлах, проте там вони переважно лише візуальні, тобто проміжки між блоками, між функціями, обов'язкова наявність лапок в умовах та циклах, забрані непотрібні повторні задання типів, покращений синтаксис створення нових об'єктів, тощо, ось для прикладу показані зміненні рядки в деяких файлах:



```
1 using ASMEngine.OperatorsLib;
2
3 namespace ASMEngine
4 {
5     public class Operators
6     {
7         public static List<Operator> DataBase = InitDataBase();
8         private static List<Operator> InitDataBase()
9         {
10             List<Operator> res = new()
11             {
12                 new ADDOperator(),
13                 new SUBOperator(),
14                 new INCOperator(),
15                 new DECOperator(),
16                 new CMPOperator(),
17                 new NEGOperator(),
18                 new MULOperator(),
19                 new IMULOperator(),
20                 new DIVOperator(),
21                 new IDIVOperator(),
22                 new CBWOperator(),
23                 new INTOperator(),
24                 new MOVOperator(),
25                 new LEAOperator(),
26                 new PUSHOperator(),
27                 new POPOperator(),
28                 new XCHGOperator(),
29                 new LOOPOperator(),
30                 new JMPOperator(),
31                 new RETOperator(),
32                 new JxxOperator(),
33                 new ANDOperator(),
34                 new SHLOperator(),
35                 new SAROperator(),
36                 new XOROperator(),
37                 new OROperator(),
38                 new CLDOperator(),
39                 new STOSBOperator()
40             };
41             return res;
42         }
43     }
44 }
```

```
1 using System;
2 using System.Collections.Generic;
3 using System.Text;
4
5 namespace ASMEngine
6 {
7     public abstract class Operator
8     {
9         public List<Format> RegisteredFormats;
10        internal int length;
11        internal int address;
12        internal bool hasAddress = false;
13        protected abstract void initFormats();
14        public virtual string Name()
15        {
16            return "";
17        }
18        public virtual bool TryToCreate(string line)
19        {
20            return line.ToUpper() == Name();
21        }
22
23        internal Operand Op1;
24        internal Operand Op2;
25        internal string CodeLine;
26
27        public Operator()
28        {
29            RegisteredFormats = new List<Format>();
30            initFormats();
31        }
32    }
33}
```

```
1 namespace ASMEngine
2 {
3     public class Line
4     {
5         public string LineNumber;
6         public string Address;
7         public string Code;
8         public string Source;
9
10        public Line(string LineNumber, string Address, string Code, string Source)
11        {
12            this.LineNumber = LineNumber;
13            this.Address = Address;
14            this.Code = Code;
15            this.Source = Source;
16        }
17    }
18
19    public class ASMAFile
20    {
21        private List<string> data;
22        private List<Operator> parsed;
23
24        private List<string> segments;
25        private List<int> relocatable;
26        private int codeLength;
27        private int stackSize;
28
29        public bool MakeComFile = false;
30        private string hex = "";
31
32        public ASMAFile(List<string> data)
33        {
34            segments = new List<string>();
35            relocatable = new List<int>();
36            this.data = data;
37            Clean();
38            parsed = new List<Operator>();
39            Compile();
40            OutCodesToFile("log.txt");
41        }
42    }
43}
```

(Зелені смуги справа показують зміни по всій довжині файлу)

19. Єдина додаткова некосметична зміна - переписаний клас AddressTable, який зберігав словники string, string конвертуючи, при тому конвертуючи отримані числа в текст для того щоб їх там зберігати, тому я переписав його щоб він одразу працював з інтовими числами, без додаткових конвертацій.

```
public class AddressTable
{
    private Dictionary<string, int> data;

    2 references
    public Dictionary<string, int> Data
    {
        get { return data; }
    }

    1 reference
    public AddressTable()
    {
        data = new();
    }

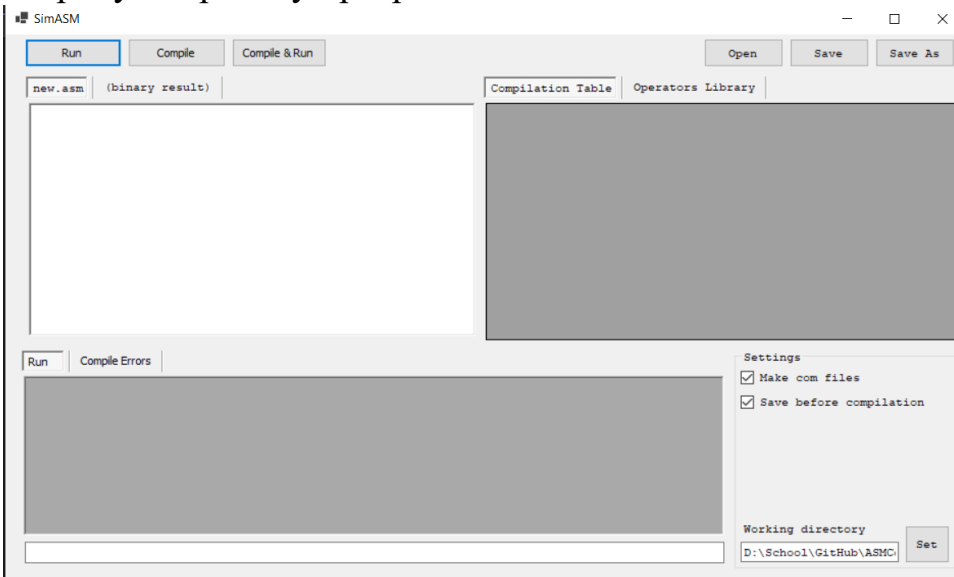
    6 references
    public void Add(string name)
    {
        if (!string.IsNullOrEmpty(name))
        {
            name = name.ToUpper();
            data.TryAdd(name, -1);
        }
    }

    5 references
    public void Set(string name, int offset)
    {
        if (!data.ContainsKey(name.ToUpper()))
        {
            throw CompileError.VariableNotDefined(name.ToUpper(), -1);
        }

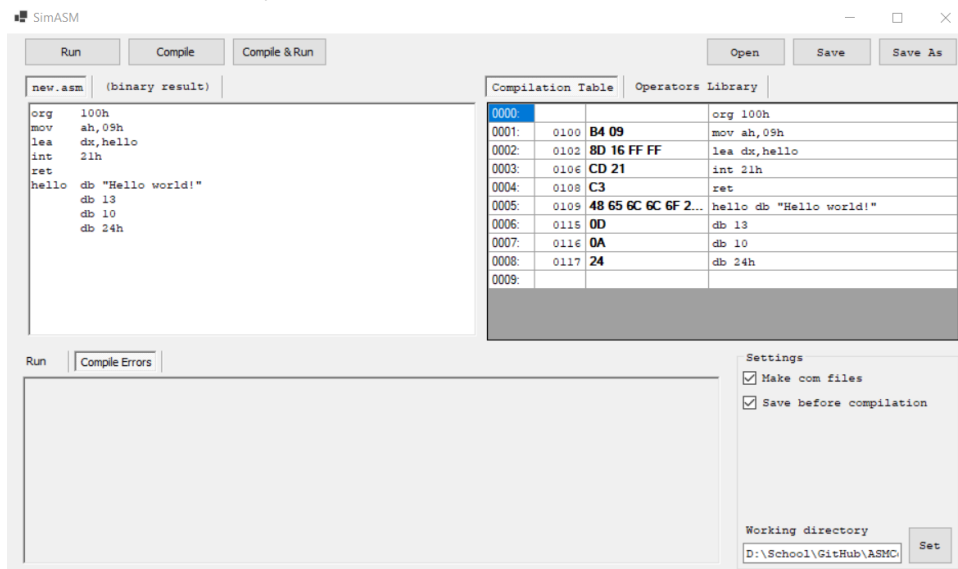
        data[name] = offset;
    }

    6 references
    public string GetOffset(string name, int offset, int bytes)
    {
        if(data.TryGetValue(name.ToUpper(), out int q))
        {
            return Convertors.DecimalToBase(Convertors.TryNegative(q + offset, bytes), 2);
        }
        else
        {
            throw CompileError.VariableNotDefined(name.ToUpper(), -1);
        }
    }
}
```

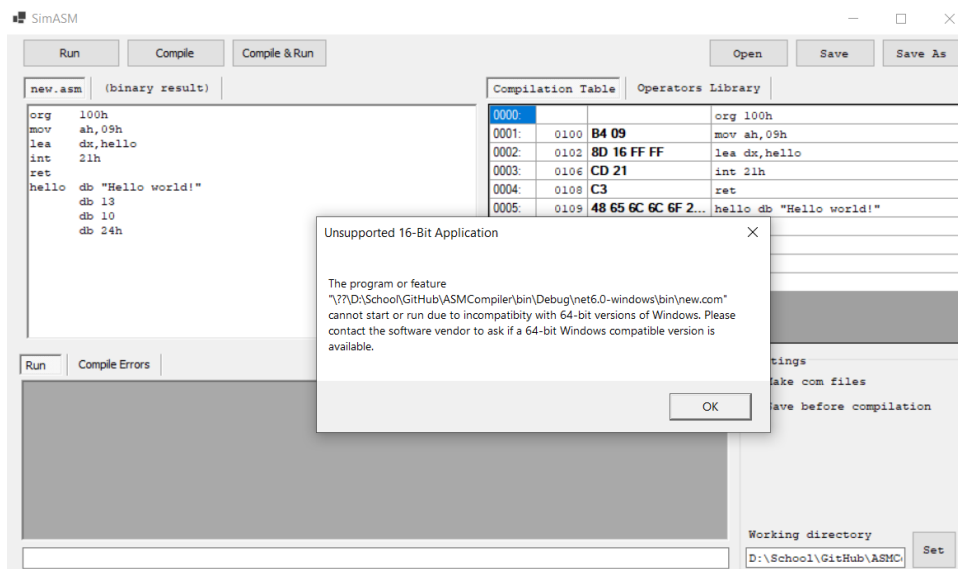
20. Спробуємо роботу програми

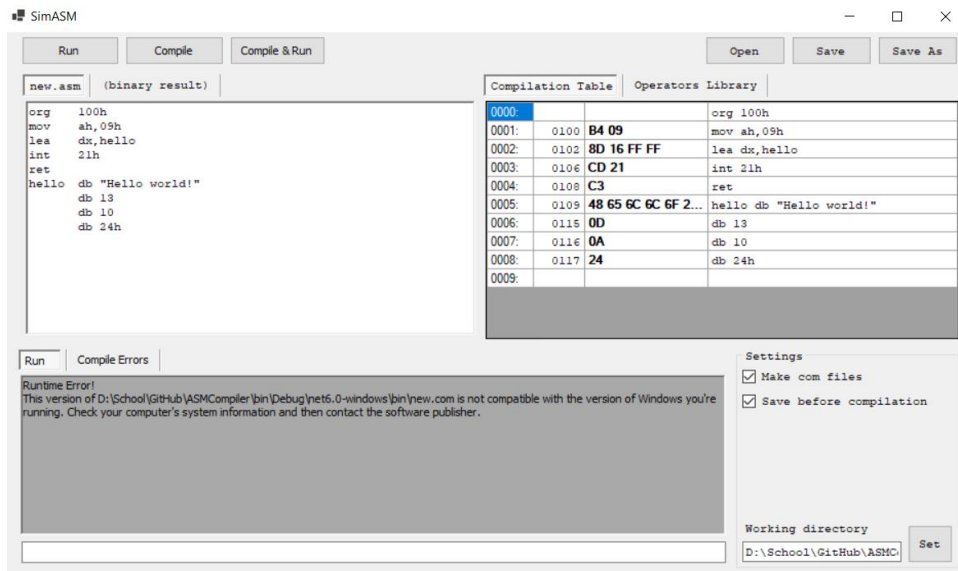


21. Запишемо сюди вміст іншого файлу не відкриваючи його і спробуємо скомпілювати - успішно.



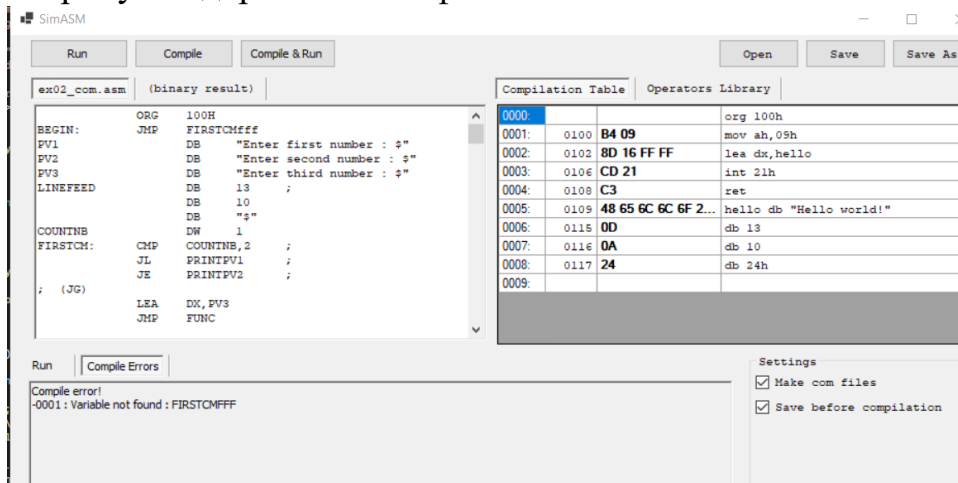
22. Хотілось би також заранити, проте на моїй в мене 64-розрядна система.



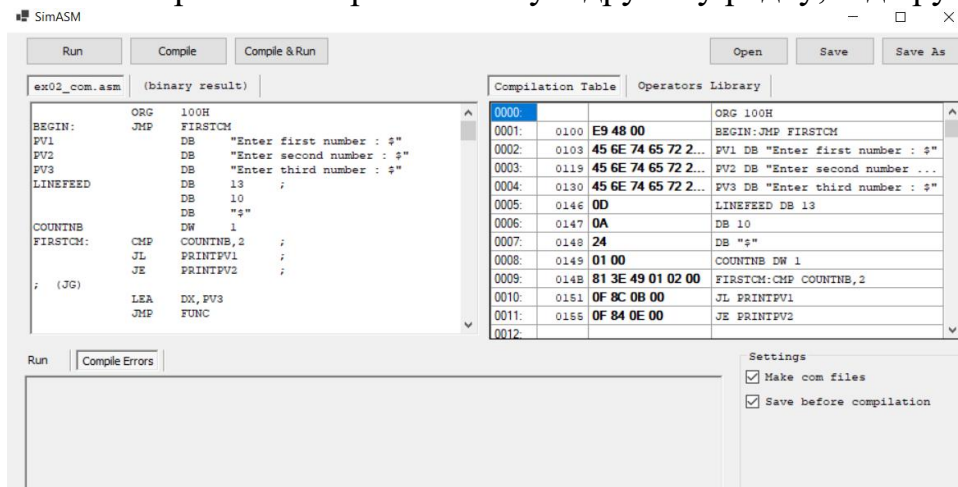


Як видно тепер помилка записується нормально.

23. Спробую відкрити інший файл та скомпілювати

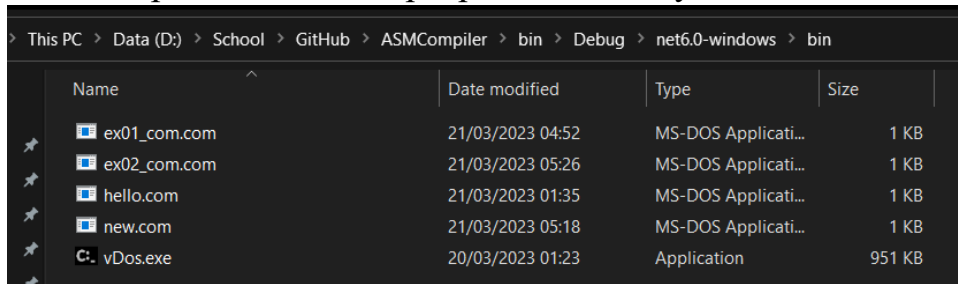


Виникла проблема через помилку в другому рядку, підітру зайві літери



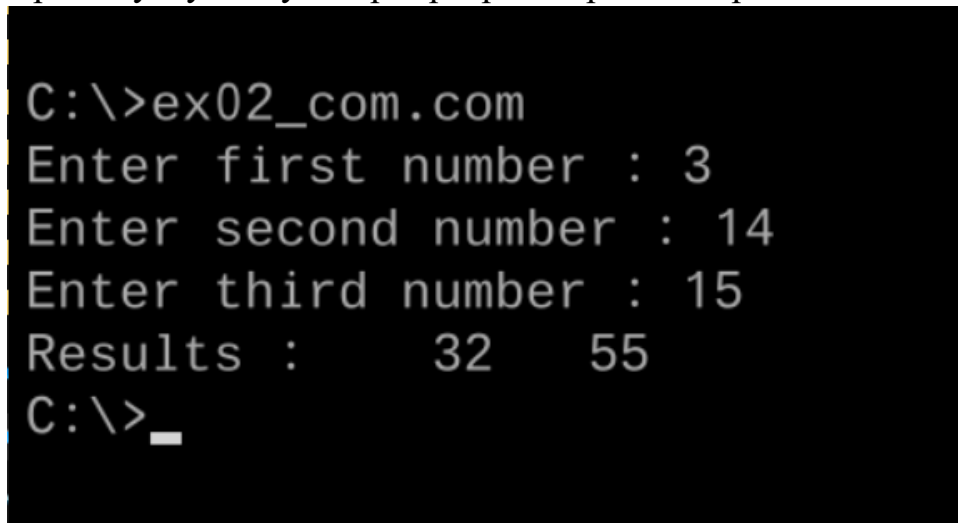
Як видно тепер табличка відображається і немає помилок.

24. Натисну кнопку зберегти і спробую запустити файл емулятором.
Файл зберігся в папці з програмою в папку bin



| Name | Date modified | Type | Size |
|--------------|------------------|---------------------|--------|
| ex01_com.com | 21/03/2023 04:52 | MS-DOS Applicati... | 1 KB |
| ex02_com.com | 21/03/2023 05:26 | MS-DOS Applicati... | 1 KB |
| hello.com | 21/03/2023 01:35 | MS-DOS Applicati... | 1 KB |
| new.com | 21/03/2023 05:18 | MS-DOS Applicati... | 1 KB |
| C:\vDos.exe | 20/03/2023 01:23 | Application | 951 KB |

При запуску в емуляторі програма працює коректно



```
C:\>ex02_com.com
Enter first number : 3
Enter second number : 14
Enter third number : 15
Results :      32      55
C:\>_
```