

Ziron Programming Language

Burgas Free University
2012.2012

Introduction

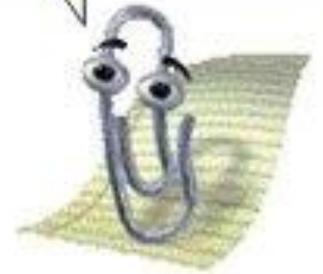


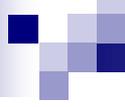
- Personal project.
- Under development since early 2011.
- Input is assembled into machine code.
- The name “Ziron” is a mix of OverHertz and Iron (a tough metal).
- Similar syntax to Assembly, C and Pascal.
- High performance assembler.

It looks like you're stupid.

Story

- The beginning.
- During a project with debugging problems.
- Compiler assembling incorrect code.
- Hard to find fault in high level compilers.
- Solution: writing inline assembly.
- Under development
Approx. 8 months total





The problems of many modern day Programmers

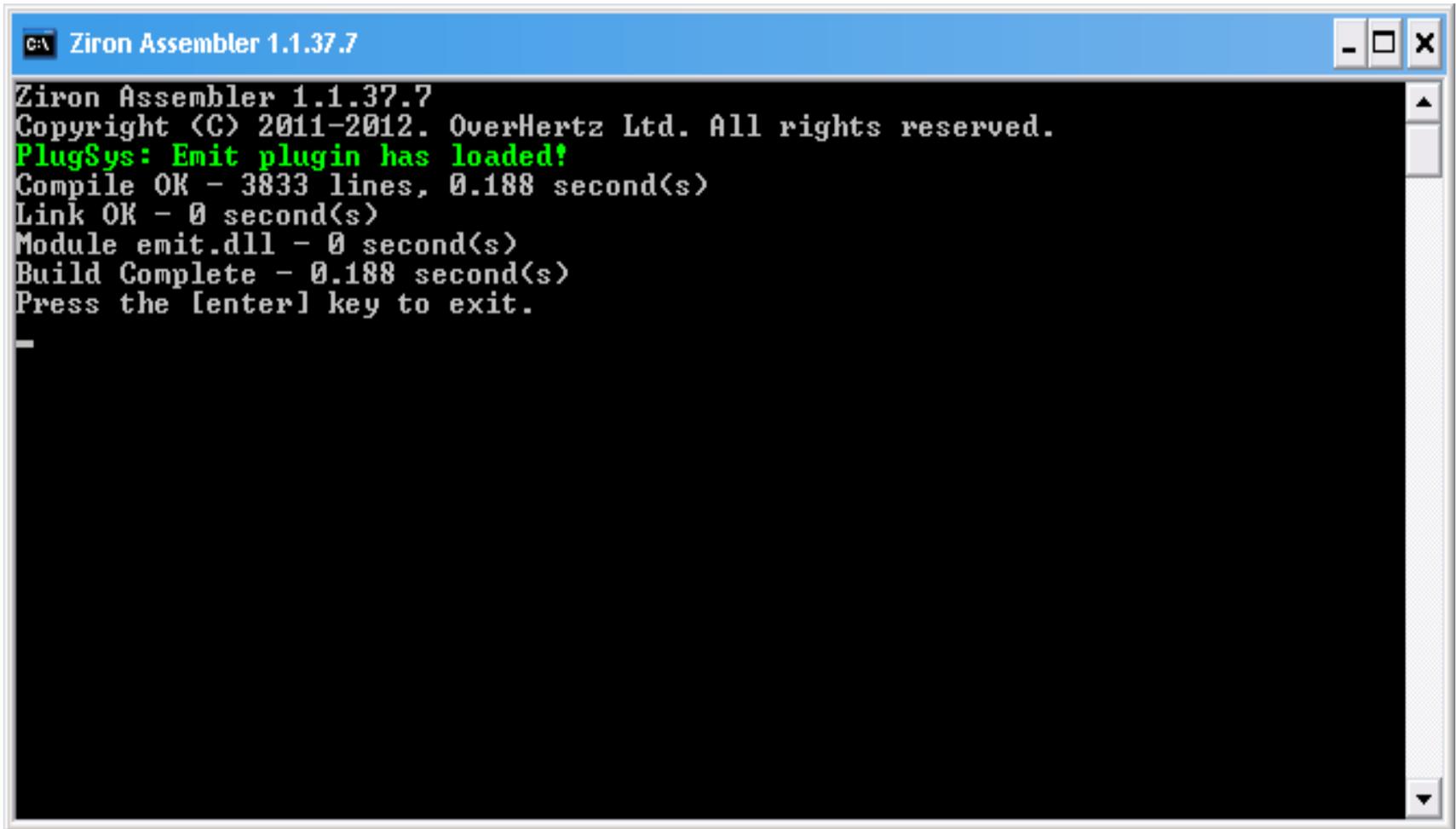
- Lazy
- Rely solely on pre-written libraries.
- Missing out on their true potential.
- Expect too much from hardware manufacturers.
- Slowing down progression of Technologies.

Philosophy

- Designed for multiple programming styles.
- Bring high and low level code together.
- Making assembly easier to understand.
- For high performance optimised code.
- Small foot-print. (Using minimal resources)
- Closer to hardware level.
- Non-strict data and return types.



Ziron Assembler



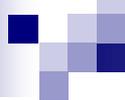
```
C:\> Ziron Assembler 1.1.37.7

Ziron Assembler 1.1.37.7
Copyright (C) 2011-2012. OverHertz Ltd. All rights reserved.
PlugSys: Emit plugin has loaded!
Compile OK - 3833 lines, 0.188 second(s)
Link OK - 0 second(s)
Module emit.dll - 0 second(s)
Build Complete - 0.188 second(s)
Press the [enter] key to exit.
-
```



Features

- OOP (Object Orientated Programming)
- Inline macro system.
- Easy to learn syntax similar to C and Pascal.
- Plug-in system and API.
- Common calling conventions such as stdcall, fastcall etc.
- High level block structures such as if, while, repeat and case statements.
- Custom data types using blocks (structs), typedef and enumeration.
- Syntax error reporting.
- Built-in linker. (output executable file format)



Syntax

- High and low level syntax.
- High level syntax similar to C and Pascal.
- Low level syntax can be written directly inline.
(examples)
- Semicolon is often used to denote the end of a statement.
- Curly braces are used to group statements.
- Supports default parameters.

Hello, World! (console application)

```
program WIN32CUI 'Hello World';
```

```
//included files
```

```
#include 'console.zir';
```

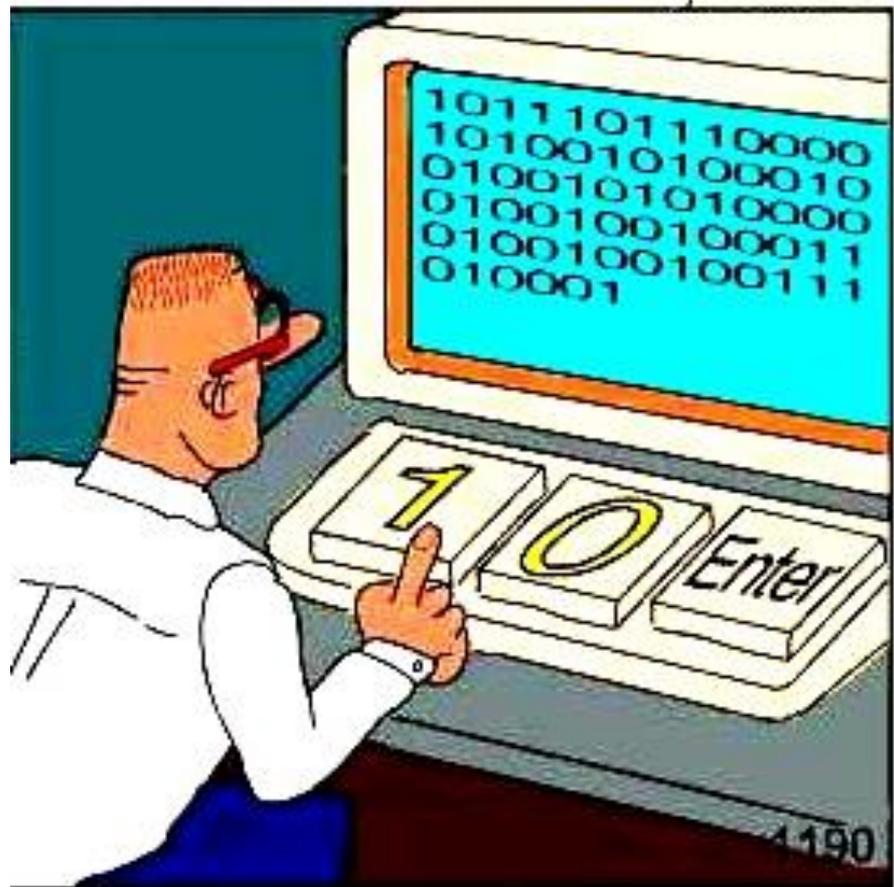
```
//show the hello world message and wait for a keypress
```

```
wait_key('Hello, World!');
```

```
//exit the program gracefully.
```

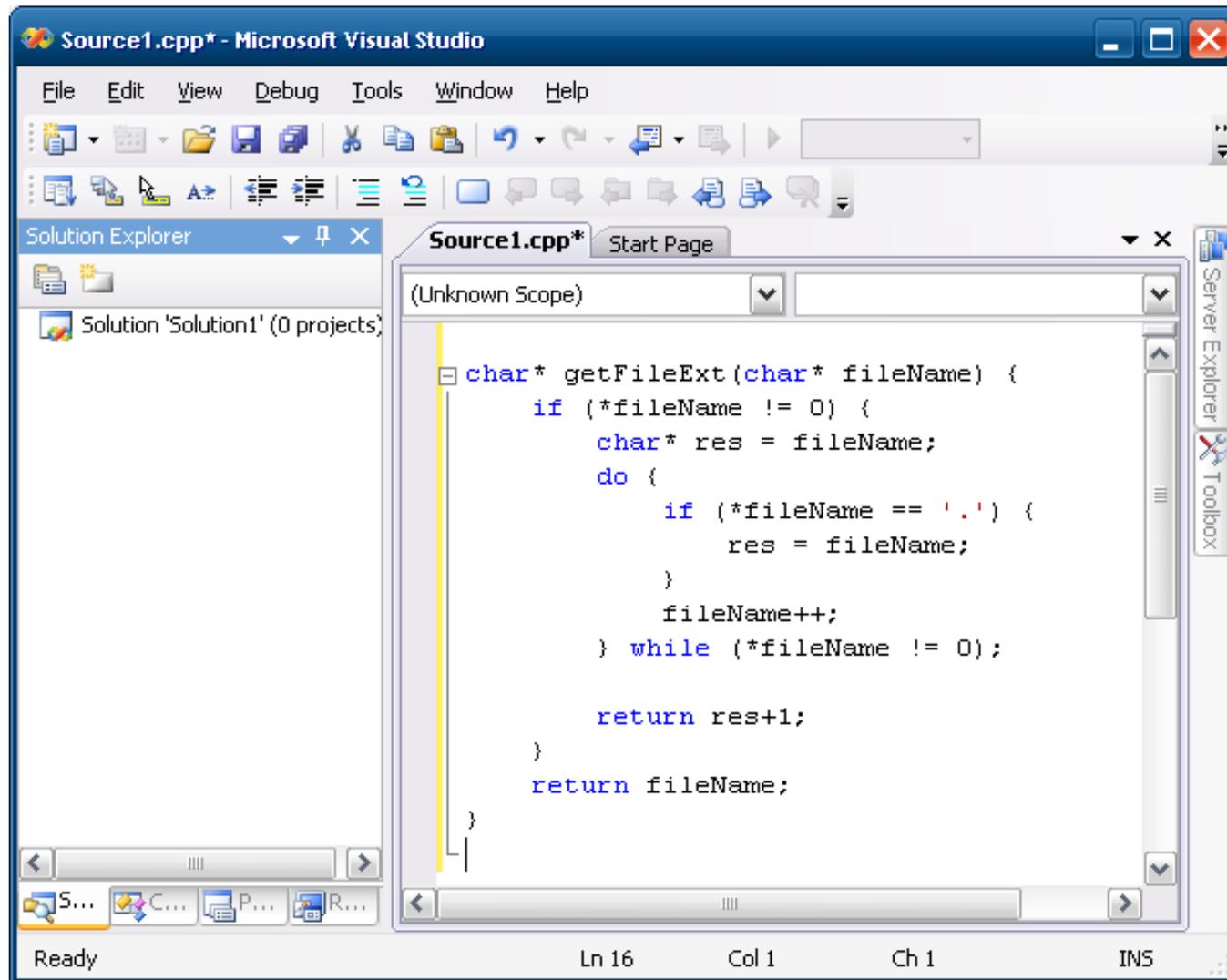
```
ExitProcess(0);
```

Comparison and Optimisation



REAL Programmers code in BINARY.

C function

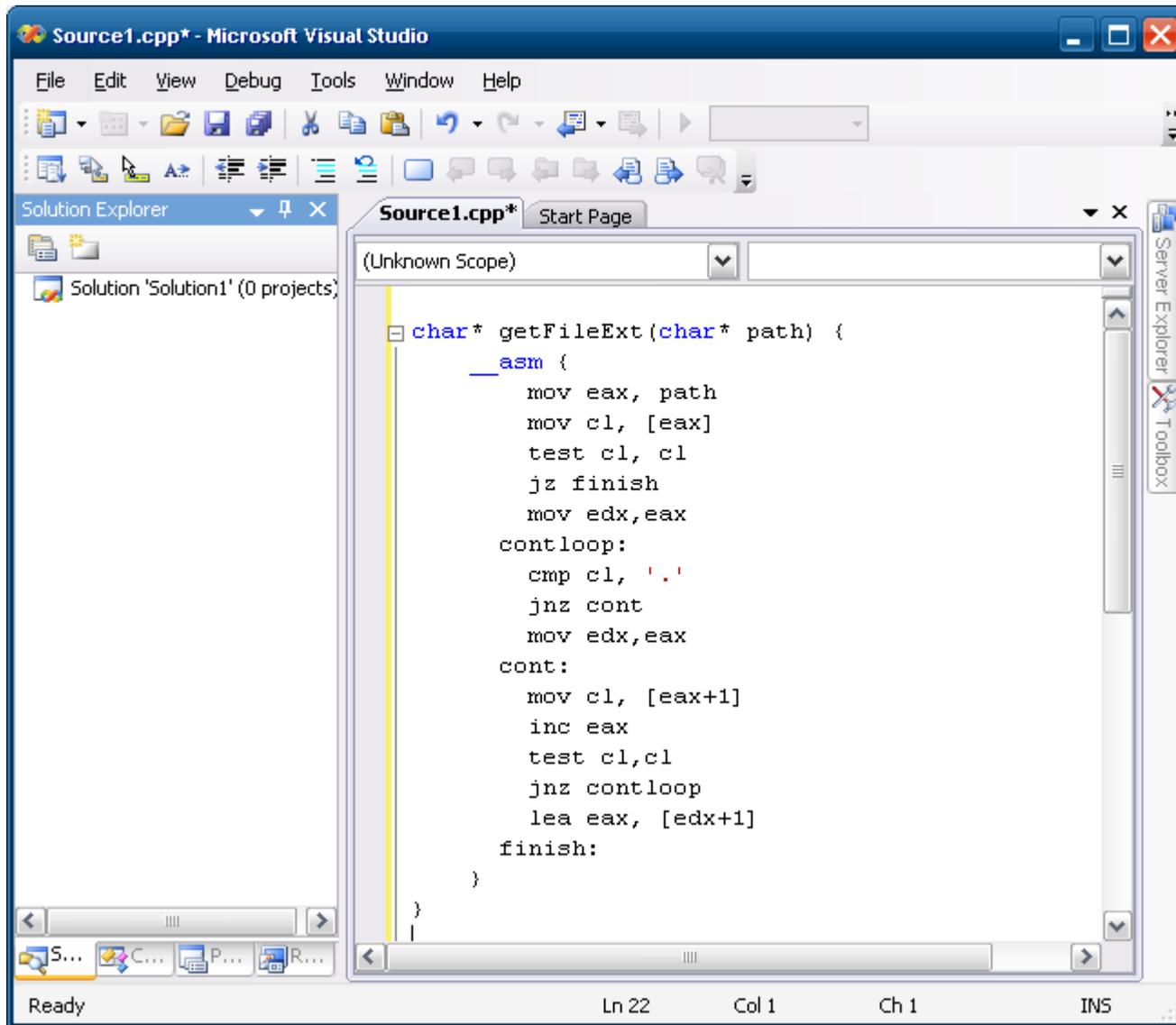


The screenshot shows the Microsoft Visual Studio IDE with a C++ source file named Source1.cpp. The code defines a function `getFileExt` that takes a `char*` parameter `fileName` and returns a `char*`. The function logic is as follows:

```
char* getFileExt(char* fileName) {  
    if (*fileName != 0) {  
        char* res = fileName;  
        do {  
            if (*fileName == '.') {  
                res = fileName;  
            }  
            fileName++;  
        } while (*fileName != 0);  
  
        return res+1;  
    }  
    return fileName;  
}
```

The status bar at the bottom of the window indicates the current cursor position: Ln 16, Col 1, Ch 1, INS.

Inner code disassembled

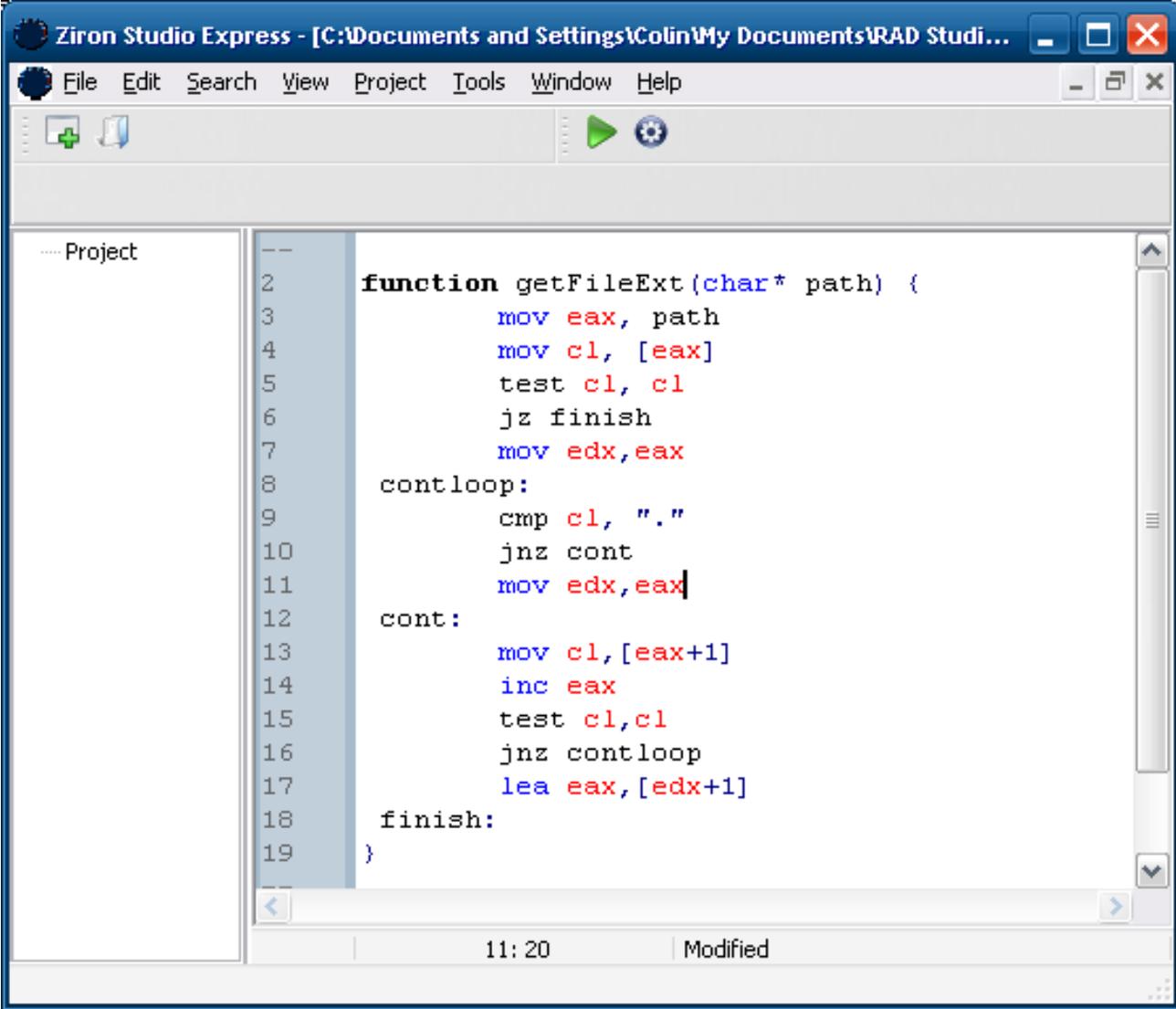


The screenshot shows the Microsoft Visual Studio IDE with the file 'Source1.cpp*' open. The Solution Explorer on the left shows a solution named 'Solution1' with 0 projects. The main editor window displays the disassembly of the function 'char* getFileExt(char* path)'. The disassembly code is as follows:

```
char* getFileExt(char* path) {  
    __asm {  
        mov eax, path  
        mov cl, [eax]  
        test cl, cl  
        jz finish  
        mov edx, eax  
        contloop:  
        cmp cl, '.'  
        jnz cont  
        mov edx, eax  
        cont:  
        mov cl, [eax+1]  
        inc eax  
        test cl, cl  
        jnz contloop  
        lea eax, [edx+1]  
    finish:  
    }  
}
```

The status bar at the bottom indicates 'Ready', 'Ln 22', 'Col 1', 'Ch 1', and 'INS'.

Ziron equivalent

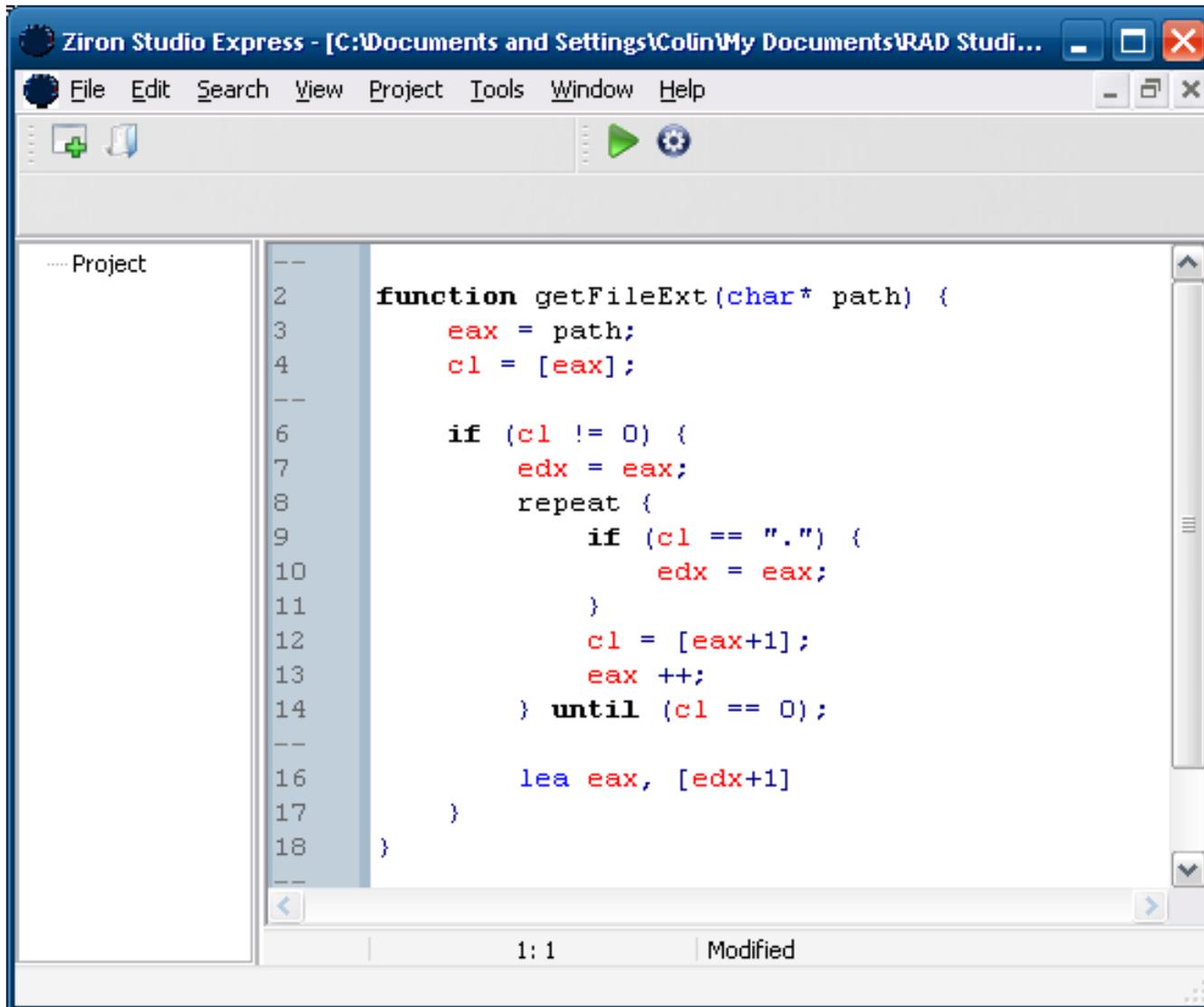


The screenshot shows the Ziron Studio Express interface. The title bar reads "Ziron Studio Express - [C:\Documents and Settings\Colin\My Documents\RAD Studi...". The menu bar includes "File", "Edit", "Search", "View", "Project", "Tools", "Window", and "Help". Below the menu bar is a toolbar with icons for a plus sign, a document, a play button, and a gear. The main workspace is divided into two panes. The left pane is titled "Project" and is currently empty. The right pane displays assembly code for a function named "getFileExt". The code is as follows:

```
2  function getFileExt(char* path) {
3      mov eax, path
4      mov cl, [eax]
5      test cl, cl
6      jz finish
7      mov edx, eax
8      contloop:
9          cmp cl, "."
10         jnz cont
11         mov edx, eax
12     cont:
13         mov cl, [eax+1]
14         inc eax
15         test cl, cl
16         jnz contloop
17         lea eax, [edx+1]
18     finish:
19 }
```

At the bottom of the window, the status bar shows the time "11: 20" and the word "Modified".

Conversion to high level



The screenshot shows the Ziron Studio Express IDE. The main window displays assembly code for a function named `getFileExt`. The code is as follows:

```
--
2  function getFileExt(char* path) {
3      eax = path;
4      cl = [eax];
--
6      if (cl != 0) {
7          edx = eax;
8          repeat {
9              if (cl == ".") {
10                 edx = eax;
11             }
12             cl = [eax+1];
13             eax ++;
14         } until (cl == 0);
--
16         lea eax, [edx+1]
17     }
18 }
```

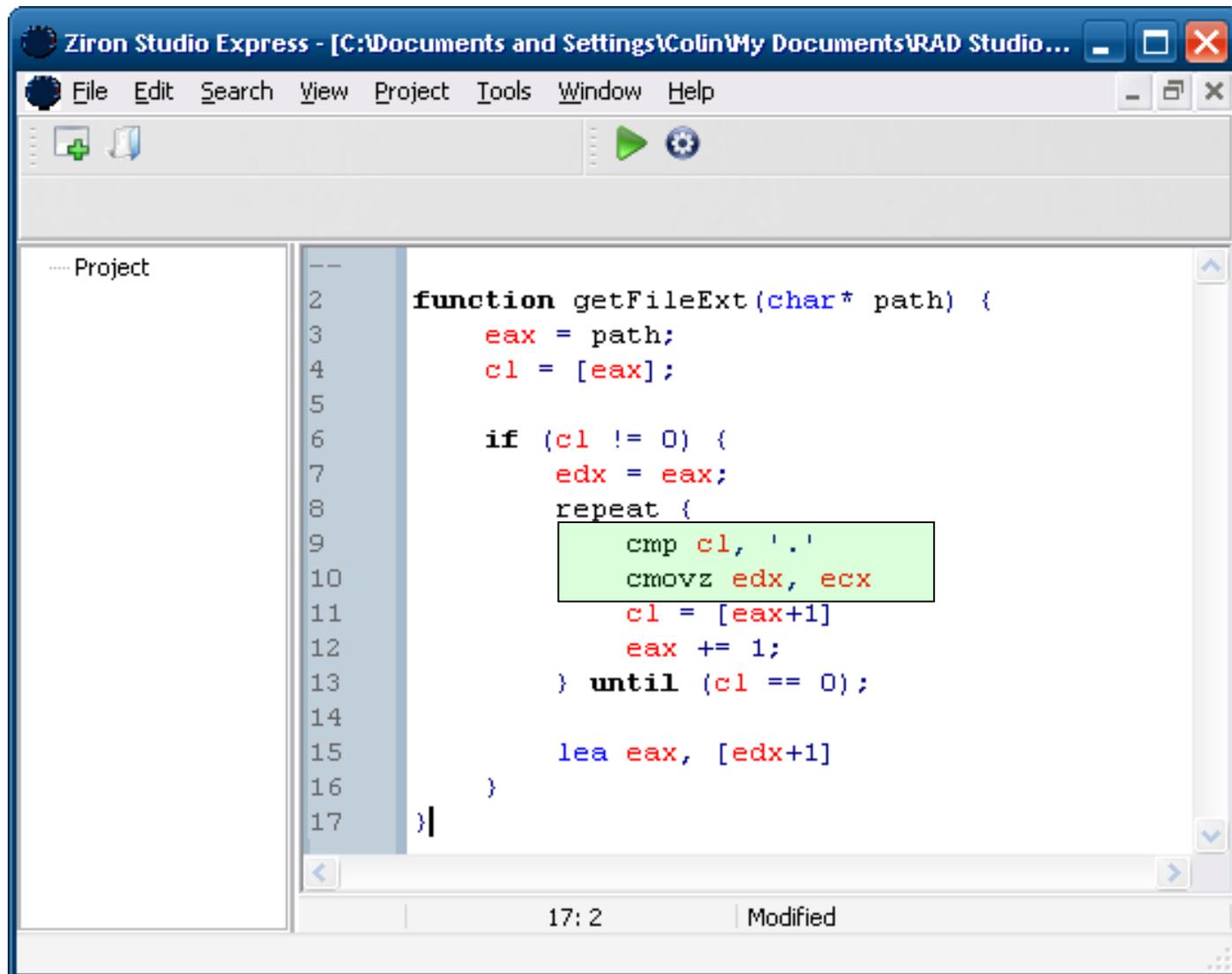
The status bar at the bottom of the window shows "1: 1" and "Modified".

Optimisation (1/2)

```
1  --
2  function getFileExt(char* path) {
3      eax = path;
4      cl = [eax];
5  --
6      if (cl != 0) {
7          edx = eax;
8          repeat {
9              if (cl == ".") {
10                 edx = eax;
11             }
12             cl = [eax+1];
13             eax ++;
14         } until (cl == 0);
15     --
16     lea eax, [edx+1]
17 }
18 }
```

1: 1 Modified

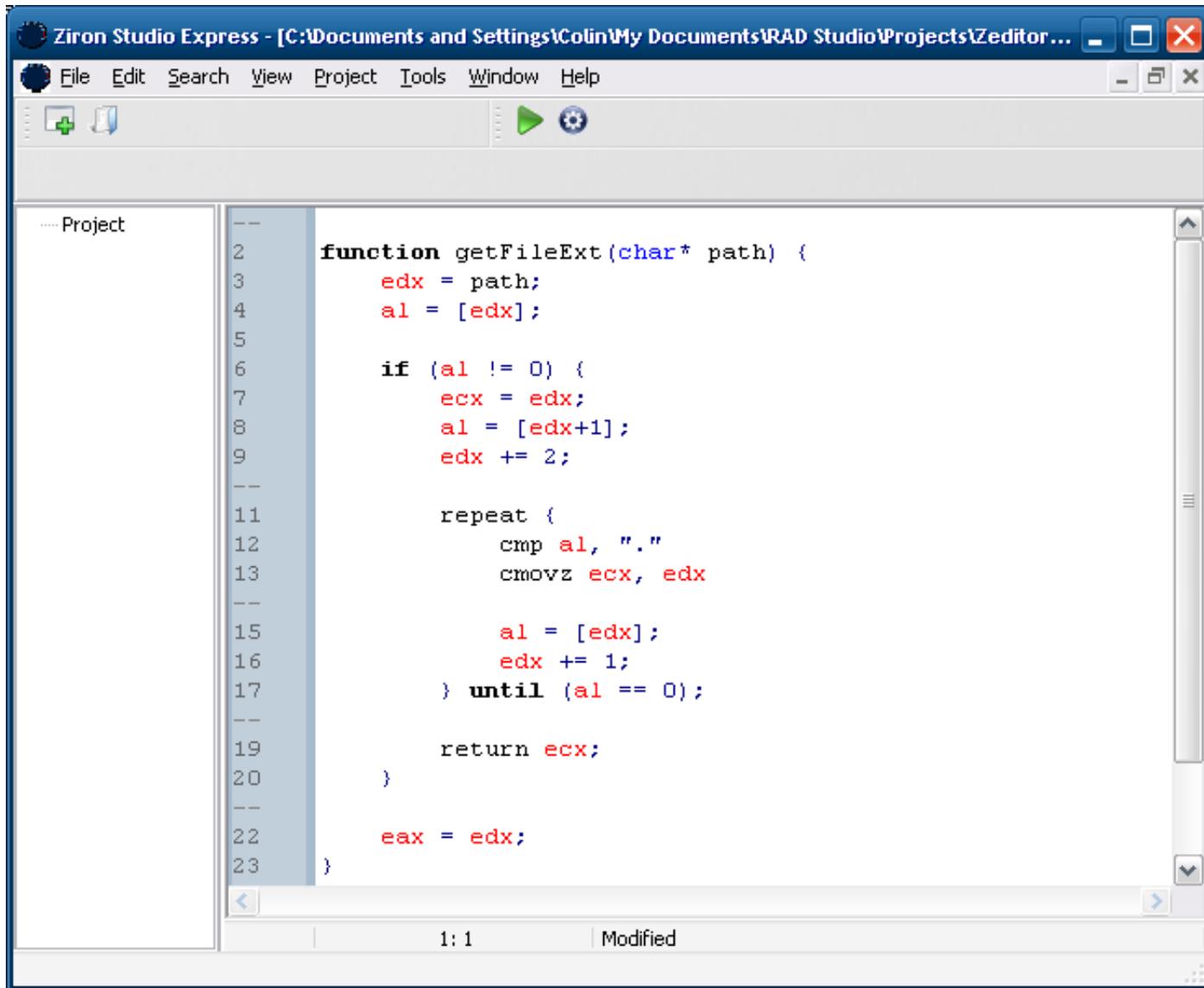
Optimisation (2/2)



```
1  --
2  function getFileExt(char* path) {
3      eax = path;
4      cl = [eax];
5
6      if (cl != 0) {
7          edx = eax;
8          repeat {
9              cmp cl, '.'
10             cmovz edx, ecx
11             cl = [eax+1]
12             eax += 1;
13         } until (cl == 0);
14
15         lea eax, [edx+1]
16     }
17 }
```

17: 2 | Modified

Slightly better!

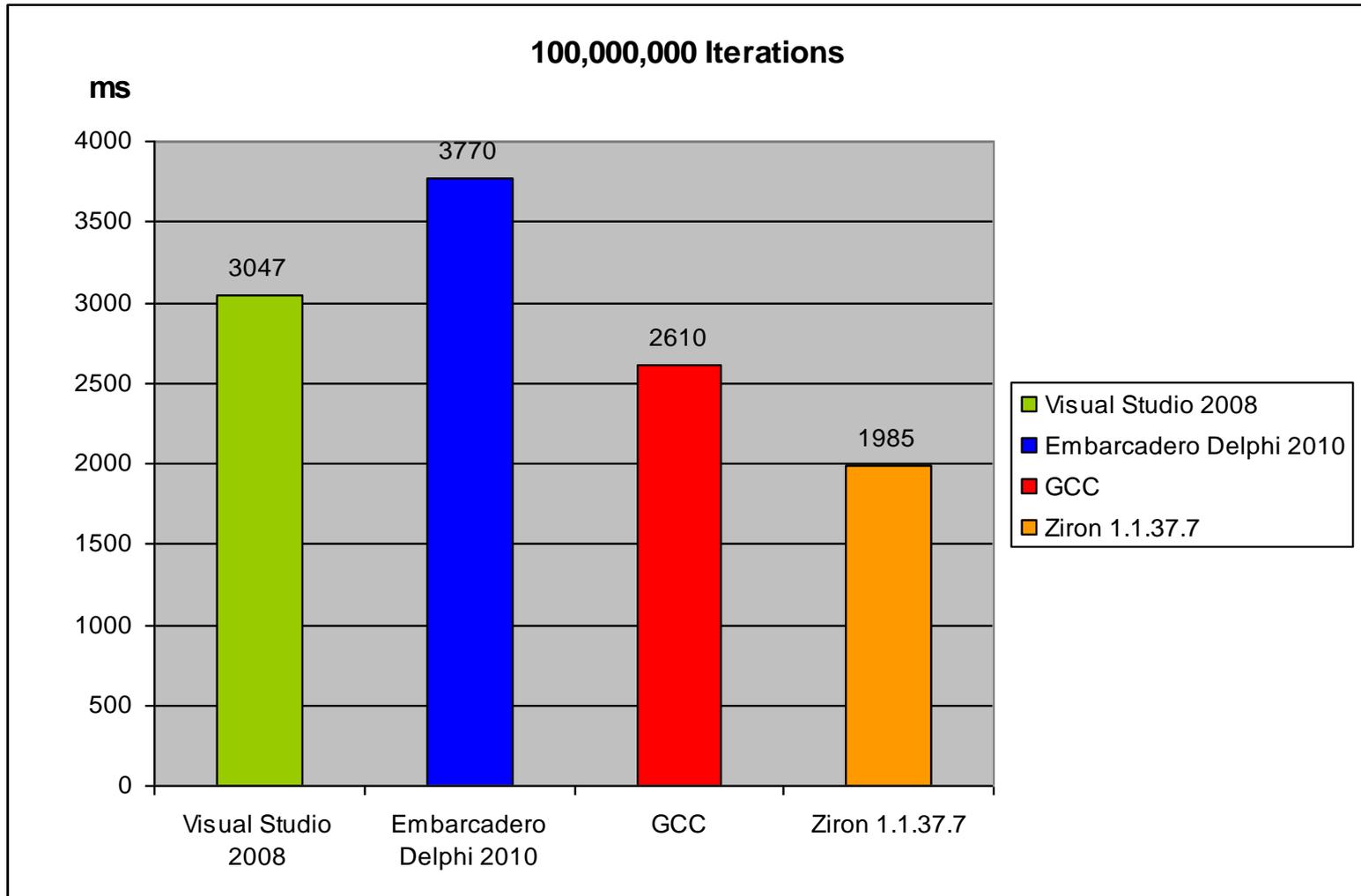


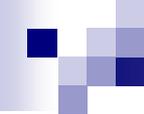
The screenshot shows the Ziron Studio Express IDE. The title bar reads "Ziron Studio Express - [C:\Documents and Settings\Colin\My Documents\RAD Studio\Projects\Zeditor...". The menu bar includes "File", "Edit", "Search", "View", "Project", "Tools", "Window", and "Help". The main editor window displays the following C code:

```
2  function getFileExt(char* path) {
3      edx = path;
4      al = [edx];
5
6      if (al != 0) {
7          ecx = edx;
8          al = [edx+1];
9          edx += 2;
10
11         repeat {
12             cmp al, "."
13             cmovz ecx, edx
14
15             al = [edx];
16             edx += 1;
17         } until (al == 0);
18
19         return ecx;
20     }
21
22     eax = edx;
23 }
```

The status bar at the bottom shows "1: 1" and "Modified".

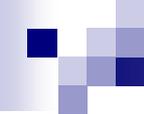
Statistics of our function in different Compilers / Languages





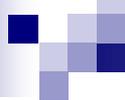
Current development

- Latest version: 1.1.38.1
- Supported architecture: x86
- Support website and Forum:
<http://www.codeziron.com>
- Large library of pre-written functions.
- Documentation available on website.



The future of Ziron

- Ziron 2
- Open for suggestions.
- Modular.
- Support for other platforms via plug-ins, Linux, Mac-OS, Android.
- Better documented features.



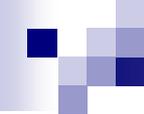
Conclusion

- Ziron provides programmers a tool to optimise their software in a simple manner.
- Writing assembly can help you understand the workings of the CPU
- Ziron can help you to learn assembly language with ease.
- Has a simple syntax that is easy to learn by many programmers.



How to get involved!

- Register at the Code::Ziron Forum.
- Learn the syntax and language features.
- Write plug-ins to extend the features of the language and assembler.
- Submit bug reports and feature requests.
- If you would like to know or learn more, please ask at the forums!



Thank you!

visit the website at:

<http://www.codeziron.com>