fasm - README

Introduction

fasm is an assembler for the ***Fonzie***[1] virtual machine. You can use it to build binaries in the *Delvecchio* format. These binaries can be loaded and executed by the machine.

File format

The file format is similar to most x86 assemblers. You can define two sections in your source code:

- **.data**: *DWORDs* stored in the memory of the machine
- **.code**: instructions

The *.data* section is optional.

.data section

You can define variables in the *.data* section. The maximum length of a variable name is 16, the minimum length is one character. Letters, numbers and underscore are allowed in a variable name, but it cannot start with a number.

A valid *.data* section looks the following way:

```
.data
    f00_=900  ; first value
    _bar=2000 ; second value
```

As you can see here each instruction can end with a comment. Comments are introduced by a semicolon.

.code section

fasm uses *Intel syntax*. The *mnemonics* below are supported:

- mov
- inc
- dec

[1] Fonzie

- sub
- add
- mul
- div
- and
- or
- rnd
- ret
- cmp
- je
- jne
- jge
- jg
- jle
- jl
- call
- ret

You find a description of all available instructions in the ***Fonzie*** documentation.

This example shows how a complete source file can look like:

```
.data
    foo=900

.code
    mov a0, [foo] ; copy 900 to register a0
    mov a1, 100   ; copy 100 to register a1
    add a0, a1    ; add register a0 and a1, result is stored in r
    mov a0, r     ; copy r to a0
    cmp a0, a1    ; compare a1 to a0
    jl foobar     ; jump to subroutine if a1 is less than a1
    ret           ; without a return the subroutine foobar would be executed

foobar:
    mov a0, a1    ; copy a1 to a0
    ret
```

Label names follow the same rules as variable names.

Building fasm

fasm is written in *C#*. The easiest way to build the executable is to compile it with *MonoDevelop* or *Microsoft Visual Studio*.

If you want to build ***fasm*** with the *Mono* compiler you can also use the *Makefile*.