

Fonzie - README

Introduction

Fonzie is a small virtual machine. It can be used to test algorithms in a virtual environment, for example.

Fonzie has 8 registers, a stack and two segments: one for data and another one for code. The stack is used only internally.

The default stack size is 256 *bytes*. The data segment can have 256 *bytes* and the code segment 2048 *bytes*.

Registers

Fonzie has the following 8 registers:

- *A0..A3*: used in mathematical instructions
- *R*: stores the result of mathematical instructions
- *IP*: points to the location of the current executing statement
- *SP*: points to the top of the stack
- *FL*: stores various flags
- *EX*: stores exceptions

The only supported datatype is *DWORD*. **Fonzie** stores bytes in *big endian* format.

OPCODES / Instructions

Results of mathematical instructions are stored in the *R* register.

Comparing two *DWORDs* you find the result of the operation in the *FL* register.

The following list contains all available *opcodes*:

- **01** (*MOV_REG_REG*): copy *DWORD* in second register to first one
- **02** (*MOV_REG_ADDR*): copy *DWORD* in memory to register
- **03** (*MOV_ADDR_REG*): copy *DWORD* in register to memory
- **04** (*MOV_REG_DWORD*): copy *DWORD* to register
- **05** (*INC*): increment *DWORD* in register
- **06** (*DEC*): decrement *DWORD* in register
- **07** (*SUB_REG_REG*): subtract value in second register from value in first one
- **08** (*SUB_REG_ADDR*): subtract value in memory from value in register

- **09** (*SUB_REG_DWORD*): subtract *DWORD* from value in register
- **10** (*ADD_REG_REG*): add values from two registers
- **11** (*ADD_REG_ADDR*): add value in register and value in memory
- **12** (*ADD_REG_DWORD*): add value in register and *DWORD*
- **13** (*MUL_REG_REG*): multiply values from two registers
- **14** (*MUL_REG_ADDR*): multiply value in register and value in memory
- **15** (*MUL_REG_DWORD*): multiply value in register and *DWORD*
- **16** (*DIV_REG_REG*): divide value in second register from value in first register
- **17** (*DIV_REG_ADDR*): divide value in memory from value in register
- **18** (*DIV_REG_DWORD*): divide *DWORD* from value in register
- **19** (*RET*): return from (sub)routine
- **20** (*CMP_REG_ADDR*): compare value in memory to value in register
- **21** (*CMP_REG_REG*): compare value in second register to value in first one
- **22** (*JE*): jump if compared values are equal
- **23** (*JNE*): jump if compared values aren't equal
- **24** (*JGE*): jump if second value is greater than or equal to first one
- **25** (*JG*): jump if second value is greater than first one
- **26** (*JLE*): jump if second value is less than or equal to first one
- **27** (*JL*): jump if second value is less than first one
- **28** (*CALL*): call subroutine

The exceptions below may occur during operation:

- **01**: a specified address is invalid
- **02**: a specified register is invalid
- **03**: stack overflow
- **04**: carry over
- **05**: division by zero
- **06**: a given *opcode* is invalid

Exceptions are stored in the *EX* register.

Returning from the main routine the virtual machine halts.

Building Fonzie / Usage

Type in the following command to build **Fonzie**:

```
make
```

Having a *little endian* system the additional option below is necessary:

`-DLITTLE_ENDIAN`

If you have a x86 compatible CPU the following option turns on some optimizations:

`-DARCH_X86`

You can use the executable to run binaries in the *Delvecchio* format. To build such binaries please have a look at the ***fasm***¹ project.

¹[fasm](#)