pyNES Documentation

Release 0.2.1

Guto Maia

November 02, 2017

Contents

Contents:

CHAPTER 1

The Legend of pyNES

There was a time when game cartridges were forged in the fire of mount doom itself. That great power was then trapped into a regular plastic shelf. Most of the secrets were sealed by fellowship of hardcore game programmers. Their names were concealed in end-game credits from games that were never supposed to be finished. Countless game lives were wasted in the first level, in fruitless attempts of unveiling their evil[1] spell.

That was what my curious and inventive mind believed for years, and still do so. As a kid, I used to play those games and always asked myself how they were done. I really wanted to experience some of the game design problems the pioneers once faced. Back then, they had to wage their own tools, hack the specs for game effects and layout the memory mapper circuits. I figure out, that to reach mount doom as equal, foremost, I had to forge my own hammer. I've decided trail their footmarks therefore I build PyNES: A Python ASM compiler for Nintendo 8 bits.

However as I strum steps progresses, the anvil didn't sound the same. Knowledge weight has changed. Internet made it all available and communities are helpful. Also, computer power had grown and programming languages evolved. I must go a further in each step of their challenges. PyNES is turning into a high-level compiler which will allow Nintendo games to be written mostly in Python. This lecture will explain the several hacks and drawbacks of such approach. And I must say, trying to compile a such evolved language to a such limited processor as the c6502 it's MADNESS. It's pyNES!

CHAPTER 2

`pynes.asm` — 6502 Assembly Instructions

CHAPTER 3

pynes.composer — Composer

PyNesVisitor

class pynes.composer.**PyNesVisitor**(*symbol_table=None*, **args*, ***kwargs*)

scope

new_symbol(*name*, ***kwargs*)

visit_ImportFrom(*node*)

visit_FunctionDef(*node*)

visit_Call(*node*)

visit_Assign(*node*)

visit_AugAssign(*node*)

get_symbol_table()

PyNesTransformer

class pynes.composer.**PyNesTransformer**(*symbol_table=None*, **args*, ***kwargs*)

CHAPTER 4

pynes.lib — extending pyNES

Function wrappers for external libraries

asm_def

class pynes.lib.**asm_def**(**args*, ***kwargs*)

A function decorator for an ASM Block function

Let's take a simple waitvblank function.

```
from pynes.lib import asm_def
from pynes.asm import BIT, BPL

@asm_def
def waitvblank():
    return (
        BIT + '$2002' +
        BPL + waitvblank()
    )

print waitvblank.as_function()
```

That must be translated to:

```
waitvblank:
BIT $2002
BPL waitvblank
RTS
```

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

p

A

G

N

P

S

V