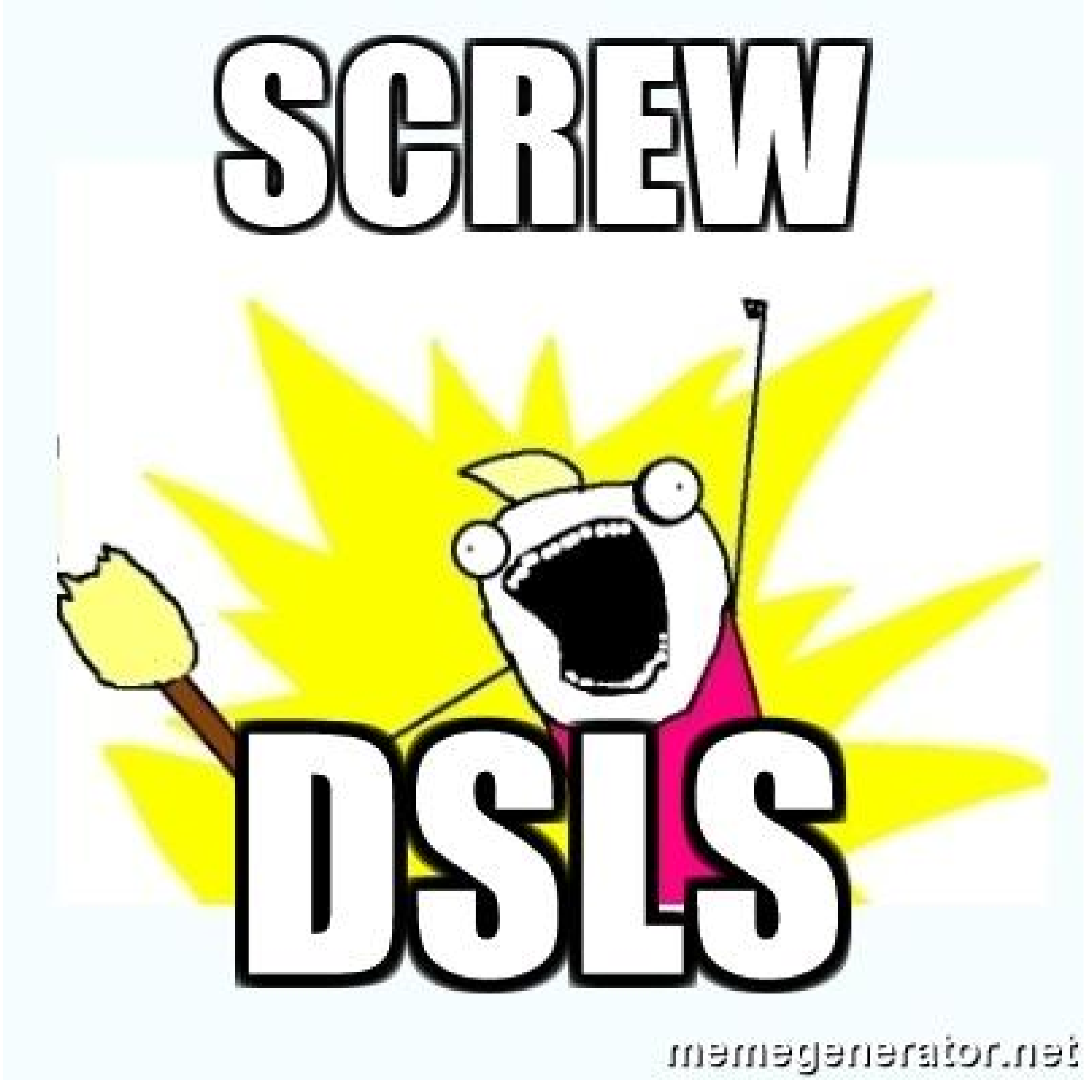
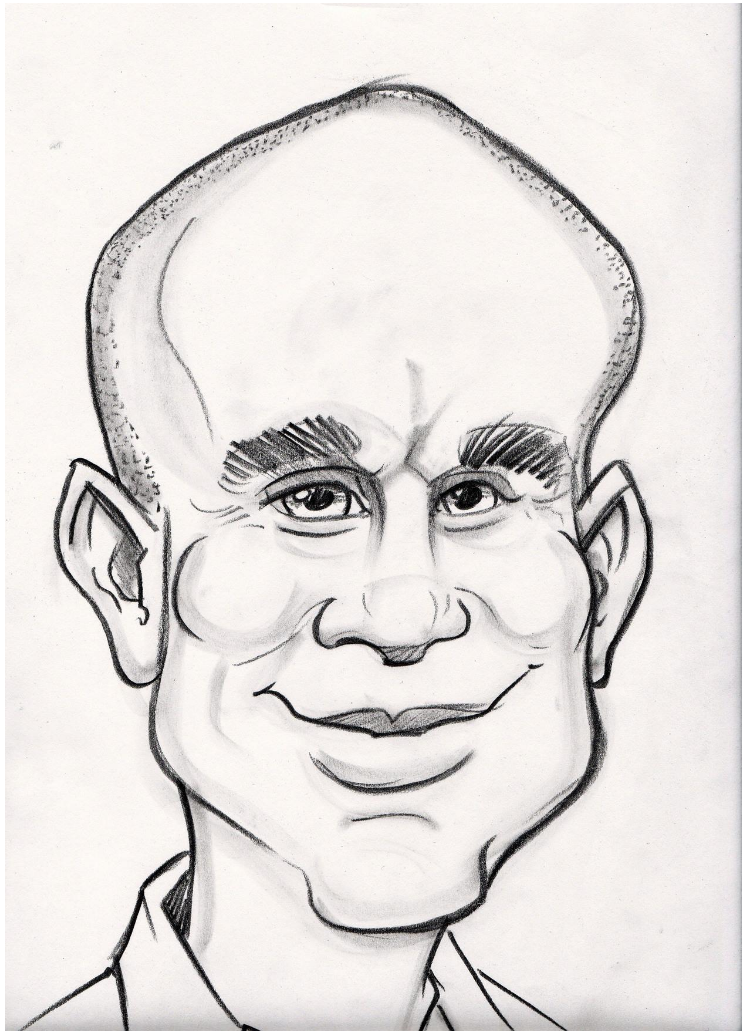

SCREW
DSLS
memegenerator.net

Miki
Tebeka

@tebeka



CEO, CTO,
UFO ...

353Solutions

Domain

Specific

Language

```
00000000: 7F45 4C46 0201 0100 0000 0000 0000 0000  .ELF............
00000010: 0300 3E00 0100 0000 E01D 0600 0000 0000  ..>.............
00000020: 4000 0000 0000 0000 88EE 3200 0000 0000  @.........2.....
00000030: 0000 0000 4000 3800 0700 4000 1B00 1A00  ....@.8...@.....
00000040: 0100 0000 0500 0000 0000 0000 0000 0000  ................
00000050: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000060: 7C66 2C00 0000 0000 7C66 2C00 0000 0000  |f,.....|f,.....
00000070: 0000 2000 0000 0000 0100 0000 0600 0000  .. .............
00000080: 7068 2C00 0000 0000 7068 4C00 0000 0000  ph,.....phL.....
00000090: 7068 4C00 0000 0000 1085 0600 0000 0000  phL.............
000000a0: C892 0900 0000 0000 0000 2000 0000 0000  .......... .....
000000b0: 0200 0000 0600 0000 D084 2C00 0000 0000  ..........,.....
000000c0: D084 4C00 0000 0000 D084 4C00 0000 0000  ..L.......L.....
000000d0: 1002 0000 0000 0000 1002 0000 0000 0000  ................
000000e0: 0800 0000 0000 0000 0400 0000 0400 0000  ................
```

`MOV AX, BX`	`MOV(AX, BX)`
`LOOP1:`	`LABEL('LOOP1')`
`; Comment`	`# Comment`

ASM

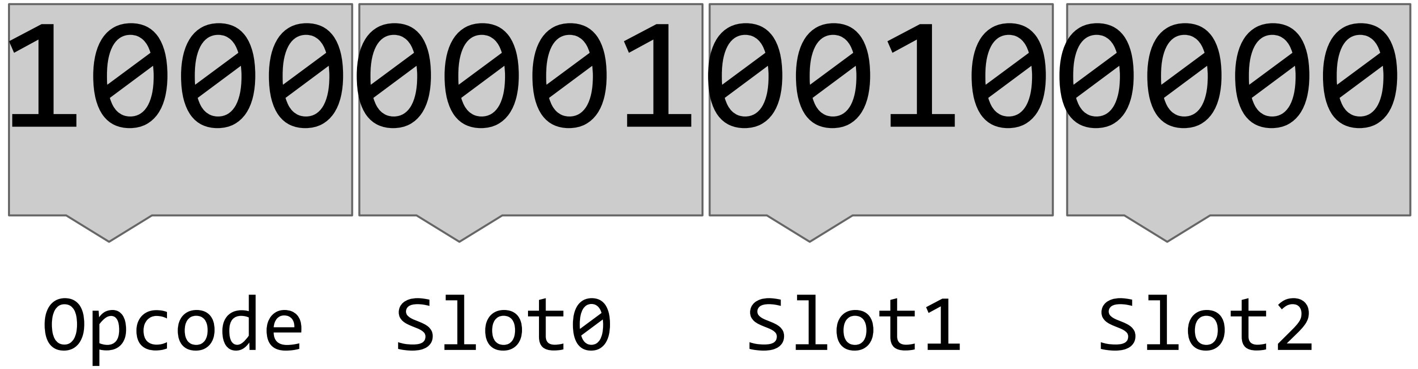
python

powered

A Toy Assembly

- Instructions are 16 bits
- 4 MSB are opcode
- Rest are arguments (4bit each)
- R0-R7 general registers
- MOV, CMP
- ADD, SUB
- JMP, JMPE

MOV(R1, 2)



TALK IS CHEAP
SHOW ME THE CODE
memegenerator.net

```
MOV(R1, 1)
MOV(R2, 1)
LABEL('LOOP')
CMP(R0, 0)
JMPE('EXIT')
MOV(R3, R1)
MOV(R1, R2)
ADD(R2, R2, R3)
SUB(R0, R0, 1)
JMP('LOOP')
LABEL('EXIT')
```

```
MOV(R1, 1)
MOV(R2, 1)
LABEL('LOOP')
CMP(R0, 0)
JMPE('EXIT')
MOV(R3, R1)
MOV(R1, R2)
ADD(R2, R2, R3)
SUB(R0, R0, 1)
JMP('LOOP')
LABEL('EXIT')
```

```python
a, b = 1, 1

while n:


    a, b = b, a + b



    n -= 1


return a
```

```
>>> from fib import fib
>>> from dis import dis
>>> dis(fib)
  4           8 SETUP_LOOP              34 (to 44)
        >>   10 LOAD_FAST                0 (n)
             12 LOAD_CONST               2 (0)    while n:
             14 COMPARE_OP               4 (>)
             16 POP_JUMP_IF_FALSE       42
  5          18 LOAD_FAST                2 (b)
             20 LOAD_FAST                1 (a)        a, b = b, a + b
             22 LOAD_FAST                2 (b)
             24 BINARY_ADD
             26 ROT_TWO
             28 STORE_FAST               1 (a)
             30 STORE_FAST               2 (b)
  6          32 LOAD_FAST                0 (n)
             34 LOAD_CONST               1 (1)
             36 INPLACE_SUBTRACT                      n -= 1
             38 STORE_FAST               0 (n)
             40 JUMP_ABSOLUTE           10
```

```python
class OpCodes(IntEnum):
    ADD_II = 0
    ADD_IR = 1
    ADD_RI = 2
    ADD_RR = 3
    SUB_II = 4
   ...
```

```
program = [] # List of instructions
labels = {}  # name -> location
instructions = {} # name -> class


def instruction(cls):
    """Decorator to register instruction"""
    instructions[cls.__name__] = cls
    return cls
```

```python
class REG:
    def __init__(self, code):
        self.code = code

    def __repr__(self):
        name = self.__class__.__name__
        return f'{name}({self.code!r})'
```

```python
@instruction
def LABEL(name):
    if name in labels:
        line = line_info(depth=2)
        raise ASMError(
            f'duplicate label - {name!r}', line)
    labels[name] = len(program)
```

```python
class ASM(ABC):
    def __init__(self):
        self.line = line_info()
        self.name = self.__class__.__name__
        program.append(self)

    @abstractmethod
    def bits(self):
        return 0
```

```
def code(self, opcode, slot1, slot2=0, slot3=0):
        return \
            (opcode << Shifts.Code) | \
            (self.slot_bits(slot1) << Shifts.Slot0) | \
            (self.slot_bits(slot2) << Shifts.Slot1) | \
            (self.slot_bits(slot3) << Shifts.Slot2)

def slot_bits(self, slot):
        if isinstance(slot, int):
            return slot & 0xF
        return slot.code
```

```python
def __str__(self):
    val = self.bits() & 0xFFFF
    return f'{val:016b}'
```

```python
@instruction
class JMP(ASM):
    opcode = OpCodes.JMP

    def __init__(self, target):
        super().__init__()
        self.target = target
```

```python
def bits(self):
        target = self.target
        if isinstance(target, str):
            target = labels[target]

        return self.code(self.opcode, target)

    def __repr__(self):
        return f'{self.name}({self.target!r})'
```

```
@instruction
class JMPE(JMP):
    opcode = OpCodes.JMPE
```

```python
@instruction
class MOV(ASM):
    def __init__(self, dest, src):
        super().__init__()
        self.dest = dest
        self.src = src

    def bits(self):
        opcode = OpCodes.MOV_I if isinstance(self.src, int) else \
            OpCodes.MOV_R
        return self.code(opcode, self.dest, self.src)

    def __repr__(self):
        return f'{self.name}({self.dest!r}, {self.src!r})'
```

```
def asm_compile(infile):
    program.clear()

    env = instructions.copy()
    for i in range(num_regs):
        env[f'R{i}'] = REG(i)

    code = infile.read()
    exec(code, env, {})
    return program
```

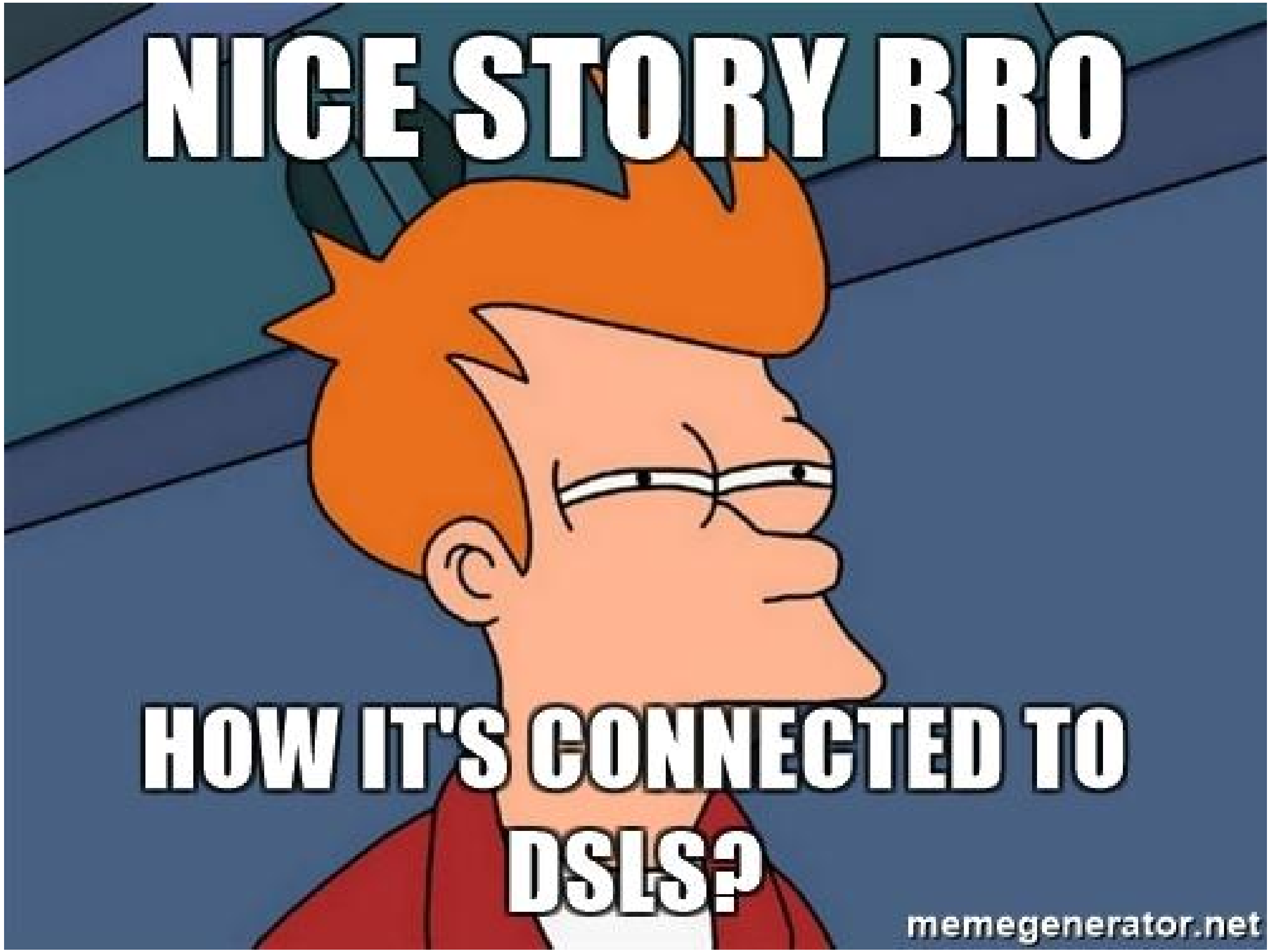
```python
try:
        program = asm_compile(args.infile)
except ASMError as err:
    fname = args.infile.name
    raise SystemExit(
        f'error: {fname}:{err.line}: {err}')

for inst in program:
    print(inst, file=args.out)
```

```
$ python asm.py fib.asm
1000000100100000
1000001000010000
1100000000000000
1011100100000000
1001001100010000
1001000100100000
0011001000100011
0110000000000001
1010001000000000
```

I CAN HAZ
MACROS?
memegenerator.net

Already There

```
def SWAP(r1, r2):
    # Swaps two registers (uses R9)
    MOV(R9, r1)
    MOV(r1, r2)
    MOV(r2, R9)
```


NICE STORY BRO
HOW IT'S CONNECTED TO DSLS?
memegenerator.net

To Write A DSL You Need To

- Be a language designer
- Implements tools
- Document
- ...

Dumb

Stupid

Language

Just Use Python™

Another Example: Configuration

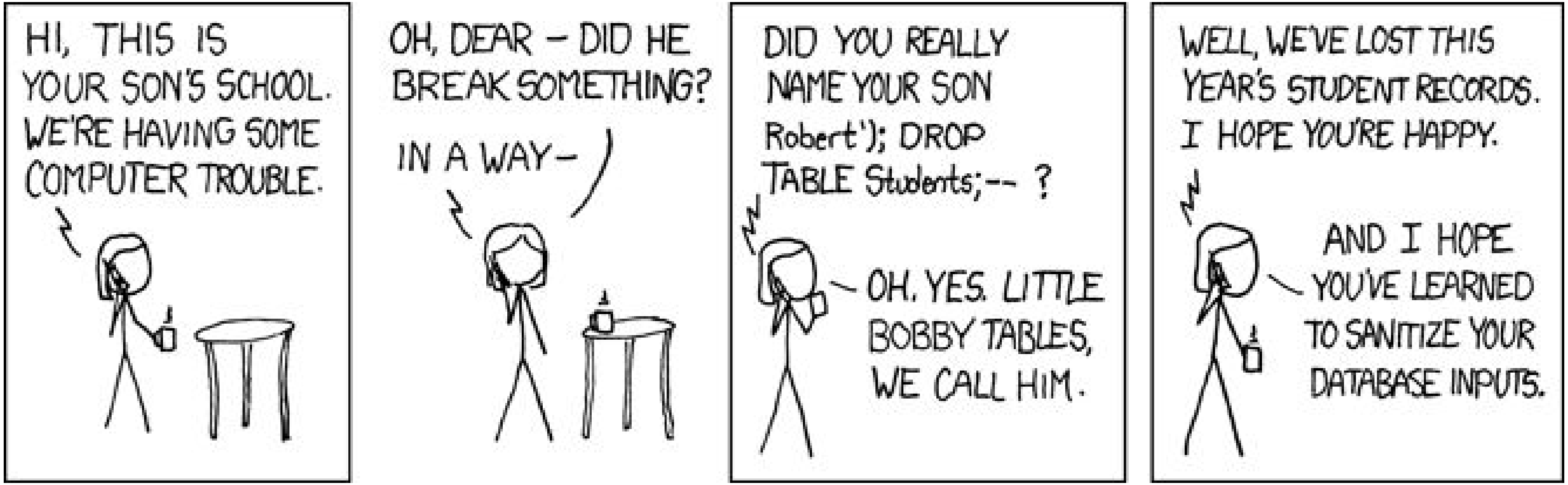
- JSON?
- YAML?
- ini?
- TOML?
- DSL?

Just Use Python™

```python
# config.py
port = 8080
db_host = 'db1.353solutions.com'

def _load_rc():
    from os import path, environ

    default = path.expanduser('~/.config/353/config')
    cfg_file = environ.get('CONFIG_FILE', default)
    if path.isfile(cfg_file):
        with open(cfg_file) as fp:
            exec(fp.read(), globals())

_load_rc()
del _load_rc
```

```
$ python app.py
DB HOST = db1.353solutions.com
PORT = 8080

$ cat overrides
port = 9999

$ CONFIG_FILE=${PWD}/overrides python app.py
DB HOST = db1.353solutions.com
PORT = 9999
```

DUDE
SECURITY!
imgflip.com


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

`yaml.safe_load`

anyone?

trust

no

one

Before you roll your own dumb, stupid language

Just Use Python™

github.com/tebeka/talks/tree/master/screw-dsls