this post was submitted on 02 Nov 2016

78 points (94% upvoted)

shortlink: `https://redd.it/5as91q`

[2016-11-02] Challenge #290 [Intermediate] Blinking LEDs

78

submitted 9 months ago * by jnazario 2 0

Description

Mark saw someone doing experiments with blinking LEDs (imagine something like this) and became fascinated by it. He wants to know more about it. He knows you are good with computers, so he comes to you asking if you can teach him how it works. You agree, but as you don't have any LEDs with you at the moment, you suggest: "Let's build an emulator with which we can see what's happening inside". And that's today's challenge.

1st Part

The 1st part should be easy, even though the description is rather verbose. If you want more challenge try the 2nd part afterwards.

Our system has 8 LEDs, we represent their state with a text output. When all LEDs are off, it is printed as string of eight dots "........". When a led is on, it is printed as "*". LED-0 is on the right side (least significant bit), LED-7 is on the left side. Having LEDs 0 and 1 on and all others off is written as "......**"

On input you get a sequence of lines forming a program. Read all lines of the input (detect EOF, or make the first line contain number of lines that follow, whichever is more convenient for you). Afterwards, print LED states as they are whenever the program performs an out instruction.

Each line is in the following format:

```
<line>: <whitespace> <instruction> |
        <empty>

<instruction> : ld a,<num> |
                out (0),a
```

<whitespace> is one or more of characters " " or "\t". <num> is a number between 0 and 255.

Instruction ld a,<num> sets internal 8-bit register A to the given number. Instruction out (0),a updates the LEDs according to the current number in A. The LED-0's state corresponds to bit 0 of number in A, when that number is represented in binary. For example, when A = 5, the LED state after out instruction is ".....*.*".

You should output the LED states after each out instruction.

Challenge input 1:

```
  ld a,14
  out (0),a
  ld a,12
  out (0),a
  ld a,8
  out (0),a

  out (0),a
  ld a,12
  out (0),a
  ld a,14
  out (0),a
```

```
....**..
....***.
```

2nd Part

We will extend our programming language, so that we can do more updates without writing out instruction for each of them. We will have loops.

Each line has the following format:

```
<line>: <whitespace> <instruction> |
        <label>                     |
        <empty>

<instruction> : ld a,<num> |
                ld b,<num> |
                out (0),a  |
                rlca       |
                rrca       |
                djnz <labelref>
```

<label> is a sequence of characters a-z A-Z _ terminated with one character ":". <labelref> is a sequence of characters a-z A-Z _ (it corresponds to some label minus the trailing ":").

Instruction ld b,<num> sets a number to register B. Instruction rlca rotates bits in register A one position to the left, in circle (i.e. bit 0 goes to bit 1, bit 1 to bit 2, and bit 7 to bit 0). Instruction rrca rotates bits in register A one position to the right, in circle. Instruction djnz <labelref> (decrement and jump if not zero) subtracts one from the value of register B and if the new value of register B is not zero then the processing of instructions continues at the line containg label corresponding to the <labelref>. You can assume that in the input text <label> is always given before the corresponding <labelref> (i.e. jumps go backwards).

You should output the LED states after each out instruction.

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

need help? message the moderators

MODERATORS

rya11111	3	1
nint22	1	2
Cosmologicon	2	3
Elite6809	1	1
XenophonOfAthens	2	1
jnazario	2	0
Godspiral	3	3
Blackshell	2	0
fvandepitte	0	0
G33kDude	1	1

about moderation team »

```
  out (0),a
  ld a,60
  out (0),a
  ld a,24
  out (0),a
  djnz triple
```

Challenge Output 2:

```
.******.
..****..
...**...
.******.
..****..
...**...
.******.
..****..
...**...
```

Challenge Input 3:

```
  ld a,1
  ld b,9

loop:
  out (0),a
  rlca
  djnz loop
```

Challenge Output 3:

```
.......*
......*.
.....*..
....*...
```

Challenge Input 4:

```
    ld a,2
    ld b,9

  loop:
    out (0),a
    rrca
    djnz loop
```

Challenge Output 4:

```
......*.
.......*
*.......
.*......
..*.....
...*....
....*...
.....*..
......*.
```

Credit

This challenge was suggested by /u/lukz in /r/dailyprogrammer_ideas, many thanks! If you have a challenge idea please share it and there's a good chance we'll use it.

59 comments **share** **report**

all 59 comments

sorted by: **best**

< > discussions in r/dailyprogrammer X

43 · 24 comments

[17-08-21] Challenge #328 [Easy] Latin Squares

ode representation with the labels resolved to PC-relative offsets. The parser is the bulk of

y 30 lines.

```
#include <stdlib.h>
#include <string.h>

#define MAX_LABEL_LENGTH 16
#define WHITESPACE ", \n"

enum op { NOP, LD_A, LD_B, OUT, RLCA, RRCA, DJNZ };

struct ins {
    char label[MAX_LABEL_LENGTH];
    enum op op;
    union {
        int v;
        char ref[MAX_LABEL_LENGTH];
    } operand;
};

static int
program_load(struct ins *ins)
{
    char line[256];
    int n = 0;
    while (fgets(line, sizeof(line), stdin)) {
        char *mnemonic = strtok(line, WHITESPACE);
        if (!mnemonic)
            continue;
        if (strcmp(mnemonic, "ld") == 0) {
            char *reg = strtok(NULL, WHITESPACE);
                                  ULL, WHITESPACE);
                                  'a' ? LD_A : LD_B;
                                  i(value);
                                  , "out") == 0) {
        } else if (strcmp(mnemonic, "rcla") == 0) {
            ins[n].op = RLCA;
        } else if (strcmp(mnemonic, "rrca") == 0) {
            ins[n].op = RRCA;
        } else if (strcmp(mnemonic, "djnz") == 0) {
            ins[n].op = DJNZ;
            strcpy(ins[n].operand.ref, strtok(NULL, WHITESPACE));
        } else {
            memcpy(ins[n].label, mnemonic, strlen(mnemonic) - 1);
        }
        n++;
    }

    /* Resolve jump targets */
    for (int i = 0; i < n; i++) {
        if (ins[i].op == DJNZ) {
            for (int j = 0; j < n; j++)
                if (strcmp(ins[j].label, ins[i].operand.ref) == 0) {
                    ins[i].operand.v = j - i;
                    break;
                }
        }
    }
    return n;
}

static struct ins program[4096];
```

```
    /* Execute program */
    int pc = 0;
    unsigned a = 0;
    int b = 0;
    while (pc < n) {
        switch (program[pc].op) {
            case NOP:
                break;
            case LD_A:
                a = program[pc].operand.v;
                break;
            case LD_B:
                b = program[pc].operand.v;
                break;
            case OUT:
                for (int i = 7; i >= 0; i--)
                    putchar((a >> i) & 1 ? '*' : '.');
                putchar('\n');
                break;
            case RLCA:
                a = a << 1 | a >> 7;
                break;
            case RRCA:
                a = a >> 1 | a << 7;
                break;
            case DJNZ:
                if (--b)
                    pc += program[pc].operand.v - 1;
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
}
```

permalink embed

[–] lukz 2 0 3 points 9 months ago

This will be one of the most efficient solutions here.

Initially I thought this challenge can increase awareness of assembly. And now I see it can also bring awareness of bytecode interpreters and how they are useful (efficiency).

permalink embed parent

[–] skeeto - 9 8 3 points 9 months ago

Simple bytecode virtual machines have a surprising number of applications, even where the input isn't a formal programming language. If a program takes a specific sequence of actions repeatedly based on run-time input (e.g. a data structure layout specification, a user-supplied regexp, an output template), rather than parse and interpret the raw input on each iteration, the program could instead compile the input to a bytecode program and execute it. It's much faster, and the intermediate bytecode form is easier to debug.

permalink embed parent

[–] danielbiegler 3 points 9 months ago

That's awesome. I really do like your solution a lot!

permalink embed parent

[–] marchelzo 2 points 9 months ago

Wouldn't it make more sense to use a `char []` for the program? Each `struct ins` has an unused (after assembly) 16-byte buffer, and then an additional 0-16 bytes wasted by the union.

permalink embed parent

[–] skeeto - 9 8 1 point 9 months ago

You're right that there's still data attached that doesn't have a purpose during execution. The labels and references could be stored

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

nps are resolved. Or there could be an intermediate representation, not directly executable, with extra

s an optimizer (to, say, remove sequential "ld a" instructions), it would transform that intermediate

[–] **primaryobjects** 7 points 9 months ago

Javascript

I went overboard and created a full ReactJs web app out of this challenge. :) Little css light bulbs light-up in animated fashion, as the program runs. Cute!

Demo | GitHub

```
export var LEDManager = {
  registers: { a: null },

  dec2bin: function(dec) {
    return (dec >>> 0).toString(2);
  },

  padLeft: function(nr, n, str) {
    return Array(n-String(nr).length+1).join(str||'0')+nr;
  },

  compile: function(program) {
    var result = { result: true, diodes: [], errors: [] };

    // Break program into lines.
    var lines = program.split(/[\r\n]/g);
    for (var i=0; i < lines.length; i++) {
      var line = lines[i].toLowerCase();
      var lineNum = parseInt(i, 10) + 1;

      if (line.length > 0) {
        // Break line into parts.
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
[ ,]/g);
{
s[0];
1];
[2];
```

```
        var terminator = parts[3];

        if (terminator === '|') {
          switch (instruction) {
            case 'ld': {
              // ld a,5 |
              LEDManager.registers[paramName] = paramValue;
              break;
            }
            case 'out': {
              // out (0),a |
              var value = LEDManager.registers[paramValue];
              if (value) {
                var binary = LEDManager.padLeft(LEDManager.dec2bin(value), 8);
                var bits = binary.split('');
                var diodes = [];

                // Set bits to 0 or 1 in diodes array.
                for (var j=0; j < bits.length; j++) {
                  diodes[j] = parseInt(bits[j], 10);
                }

                // Add to output array.
                result.diodes.push(diodes);
              }
              else {
                result.result = false;
                result.errors.push('Error on line ' + lineNum + ': Null parameter \'' + paramValue + '\
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
lse;
```

```
                    result.errors.push('Error on line ' + lineNum + ': Invalid instruction.');
                  }
                };
              }
              else {
                result.result = false;
                result.errors.push('Error on line ' + lineNum + ': Invalid parameter. Use format: a,8');
              }
            }
            else if (line.indexOf('|') === -1) {
              result.result = false;
              result.errors.push('Error on line ' + lineNum + ': Missing line terminator \'|\'.');
            }
            else {
              result.result = false;
              result.errors.push('Error on line ' + lineNum + ': Invalid instruction length.');
            }
          }
        }

        return result;
      }
    };
```

permalink **embed**

[–] **eMkaQQ** 3 points 9 months ago

part 1 is pretty easy with **PL/SQL** if we use table for inputs and cursor to fetch it

```
declare
```

< > discussions in r/dailyprogrammer X

43 · 24 comments

[17-08-21] Challenge #328 [Easy] Latin Squares

```
    a_num number;
    a_bin varchar2(8);
begin
    for cmd in program_lines
    loop
        if ltrim(cmd.line) like 'ld a,%' then
            a_bin := null;
            a_num := substr(cmd.line,instr(cmd.line,',')+1);
            while a_num > 0 loop
                a_bin := mod(a_num,2) || a_bin;
                a_num := trunc(a_num/2);
            end loop;
            a_bin := lpad(a_bin,8,'0');
            a_bin := replace(a_bin,'0','.');
            a_bin := replace(a_bin,'1','*');
        end if;

        if ltrim(cmd.line) like 'out (0),a' then
            dbms_output.put_line(a_bin);
        end if;
    end loop;
end;
```

I will try to expand it to part 2. As a begginer I'm open to any sugestions.

permalink **embed**

[–] **eMkaQQ** 2 points 9 months ago

and there is second part

< > discussions in r/dailyprogrammer X

43 · 24 comments

[17-08-21] Challenge #328 [Easy] Latin Squares

```
    a_num number;
    b_num number;
    a_bin varchar2(8);
begin
    select count(*)
      into lines
      from challenge_input;

    while i<=lines
    loop
        select line
          into cmd
          from challenge_input
         where nr = i;

        if ltrim(cmd) like 'ld b,%' then
            b_num := substr(cmd,instr(cmd,',')+1);
        elsif ltrim(cmd) like 'ld a,%' then
            a_bin := null;
            a_num := substr(cmd,instr(cmd,',')+1);
            while a_num > 0
            loop
                a_bin := mod(a_num,2) || a_bin;
                a_num := trunc(a_num/2);
            end loop;
            a_bin := lpad(a_bin,8,'0');
            a_bin := replace(a_bin,'0','.');
            a_bin := replace(a_bin,'1','*');
                                  out (0),a' then
                                  e(a_bin);
                                  rlca' then
                                  in,2) || substr(a_bin,1,1);
                                  rrca' then
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
                a_bin := substr(a_bin,8) || substr(a_bin,1,7);
            elsif ltrim(cmd) like 'djnz%' then
                b_num := b_num - 1;
                if b_num > 0 then
                    select nr
                      into i
                      from challenge_input
                     where line like substr(ltrim(cmd),6)||'%';
                end if;
            end if;

            i:=i+1;
        end loop;
    end;
```

permalink **embed** **parent**

[–] nevec71 3 points 9 months ago

C#. I only just started learning C#, any feedback is welcome.

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BlinkingLeds
{
    class Program
                   ] args)

                   ;
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
            string sFile = @"LedInstructions.txt";
            string[] sLines = File.ReadAllLines(sFile);

            foreach (string sLineItem in sLines)
            {
                ExecCommand(ledA, sLineItem.Trim());
            }

        }

        private static void ExecCommand(Led myLed, string sCommand)
        {
            string[] sAllowedCommands = { "ld a", "out(0)" };
            string[] sCommandsIn = sCommand.Split(',');

            switch (sCommandsIn[0])
            {
                case "ld a":
                    myLed.State = int.Parse(sCommandsIn[1]);
                    break;

                case "out (0)":
                    myLed.Display();
                    break;

                default:
                    break;
            }
```

```
{
    public class Led
    {
        public Led()
        {
            _state = 0;
        }

        protected int _state;

        public int State
        {
            get
            {
                return _state;
            }
            set
            {
                _state = value;
            }
        }

        public void Display()
        {
            string sReg = "";
            for (int i = 7; i >= 0; i--)
            {
                                          (Math.Pow(2, i))) != 0)
```

```
                    {
                        sReg += ".";
                    }
                }
                Console.WriteLine(sReg);
            }
        }
    }
```

permalink **embed**

[–] **taindissa_work** 2 points 9 months ago

This is a really cool introduction to assembly and low level memory storage.

permalink **embed**

[–] **asterite** 2 points 9 months ago

Crystal/Ruby, 1st part:

```
input = <<-INPUT
  ld a,14
  out (0),a
  ld a,12
  out (0),a
  ld a,8
  out (0),a

  out (0),a
  ld a,12
  out (0),a
  ld a,14
  case line
  when /ld\s+a,(\d+)/
    a = $1.to_i
  when /out \(0\),a/
    7.downto(0) { |i| print a & (1 << i) == 0 ? '.' : '*' }
    puts
  end
end
```

https://play.crystal-lang.org/#/r/1d8a

permalink **embed**

[–] **Vectorious** 2 points 9 months ago

Rust

```
use std::collections::HashMap;
use std::fs::File;
use std::io::BufReader;
use std::io::prelude::*;

#[derive(Debug)]
enum Register { A, B }

#[derive(Debug)]
enum Instruction {
    LD(Register, u8),
    OUT,
    RLCA,
    RRCA,
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
#[derive(Default)]
struct CPU {
    a: u8,
    b: u8,
    pc: usize,
    label_map: HashMap<String, usize>,
    rom: Vec<Instruction>,
    running: bool,
}

impl CPU {
    pub fn new(rom: Vec<Instruction>) -> CPU {
        CPU {
            label_map: CPU::find_labels(&rom),
            rom: rom,
            running: false,
            ..Default::default()
        }
    }

    fn find_labels(rom: &Vec<Instruction>) -> HashMap<String, usize> {
        rom.iter().enumerate().filter_map(|(idx, ins)| {
            match ins {
                &Instruction::Label(ref label) => Some((label.to_owned(), idx)),
                _ => None,
            }
        }).collect()
    }
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
                match reg {
                    &Register::A => self.a = num,
                    &Register::B => self.b = num,
                }
            }
            OUT => {
                led(self.a);
            }
            RLCA => {
                self.a = (self.a >> 7) | (self.a << 1);
            }
            RRCA => {
                self.a = (self.a << 7) | (self.a >> 1);
            }
            DJNZ(ref label) => {
                self.b -= 1;
                if self.b != 0 {
                    self.pc = self.label_map[label];
                }
            }
            Label(_) => {}
            EOF => {
                self.running = false;
            }
        }
        self.pc += 1;
    }
```

```
        while self.running {
            self.execute();
        }
    }
}

fn led(a: u8) {
    println!("{}", format!("{:08b}", a).replace("0", ".").replace("1", "*"))
}

fn interpret(line: Result<String, std::io::Error>) -> Option<Instruction> {
    use Instruction::*;

    if let Ok(line) = line {
        if line.starts_with(|c| c == ' ' || c == '\t') {
            let mut split = line.trim().split_whitespace();
            let instruction = match split.next().unwrap() {
                "ld" => {
                    let mut ld_split = split.next().unwrap().split(',');
                    let reg = match ld_split.next().unwrap() {
                        "a" => { Register::A }
                        "b" => { Register::B }
                        e => { panic!("Unknown register: {}", e) }
                    };
                    LD(reg, ld_split.next().unwrap().parse().unwrap())
                }
                "out" => { OUT }
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
t().unwrap().to_owned())
```

```
                e => { panic!("Unknown instruction: {}", e) }
            };
            Some(instruction)
        } else if !line.is_empty() {
            Some(Label(line.split(':').next().unwrap().to_owned()))
        } else {
            None
        }
    } else {
        None
    }
}

fn main() {
    let f = File::open("in").expect("'in' not found.");
    let mut rom: Vec<Instruction> = BufReader::new(f).lines().map(interpret).filter_map(|i| i).collect();
    rom.push(Instruction::EOF);
    let mut cpu = CPU::new(rom);
    cpu.run();
}
```

permalink **embed**

[–] **IceDane** 0 0 2 points 9 months ago

Haskell

Should support forward jumps since it does two passes. One to collect all the labels and one pass to resolve them into absolute "addresses" in the CPU.

```
import Data.Char
adP
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
import Data.List
import Data.Maybe
import Data.Array
import Data.Word
import Data.Bits
import qualified Data.Map.Strict as M

data Register
    = A
    | B
    deriving Show

data LabelRef = Ref String | Abs Int
    deriving (Show, Ord, Eq)

-- Uninhabited types to be used as parameters
-- to Instruction, to ensure on the type level
-- that we do not try to run programs where the
-- symbols (labels) have not been resolved
data Resolved
data Unresolved

data Instruction a
    = LD Register Word8
    | Out Register
    | RLCA
    | RRCA
    | Label LabelRef
    { ip      :: Int
    , end     :: Int
    , program :: Array Int (Instruction Resolved)
    , a       :: Word8
    , b       :: Word8
    }

main :: IO ()
main = do
    input <- lines <$> getContents
    let  parsed = parseProgram input
    case parsed of
        Nothing ->
            putStrLn "error: Could not parse program"
        Just p -> do
            -- Collect label information, and then convert labels
            -- into absolute references ("addresses" in the CPU)
            let resolved = resolveLabels (collectLabels p) p
                cpu      = initializeCPU resolved
            run cpu

collectLabels :: [Instruction Unresolved] -> M.Map LabelRef Int
collectLabels = foldl' collect M.empty . zip [0..]
where
    collect m (i, Label l) = M.insert l i m
    collect m _ = m

resolveLabels :: M.Map LabelRef Int -> [Instruction Unresolved] -> [Instruction Resolved]
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
licit'.
ved -> Instruction Resolved
$ m M.! l)
```

```
    resolve (Label l) = Label (Abs $ m M.! l)
    resolve (Out r)   = Out r
    resolve (LD x y)  = LD x y
    resolve RLCA      = RLCA
    resolve RRCA      = RRCA



initializeCPU :: [Instruction Resolved] -> CPU
initializeCPU resolved =
    CPU
    { ip      = 0
    , end     = length resolved - 1
    , a       = 0
    , b       = 0
    , program = listArray (0, length resolved - 1) resolved
    }

run :: CPU -> IO ()
run = evalStateT go
where
    -- Retrieve the next instruction
    next = do
        i <- gets ip
        gets (\s -> program s ! i)

    -- Run the next instruction
    go = do
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
we're at the end
```

```
            i <- gets ip
            e <- gets end
            when (i <= e) $ do
                modify (\s -> s { ip = i + 1 })
                go
        interpret (LD A v) = modify (\s -> s { a = v }) >> step
        interpret (LD B v) = modify (\s -> s { b = v }) >> step
        interpret (Out r)  = printLED r >> step
        interpret RLCA     = modify (\s -> s { a = rotateL (a s) 1 }) >> step
        interpret RRCA     = modify (\s -> s { a = rotateR (a s) 1 }) >> step
        interpret (DJNZ (Abs v)) = do
            b' <- gets b
            when (b' > 1) $
                modify (\s -> s { ip = v, b = b' - 1 }) >> go
        interpret _ = step

        printLED :: Register -> StateT CPU IO ()
        printLED A = gets a >>= liftIO . printBinary
        printLED B = gets b >>= liftIO . printBinary

        printBinary n = putStrLn $  map (\x -> conv $ bit x .&. n) $ reverse [0..7]
        conv n | n > 0 = '*'
            | otherwise = '.'

    --------------------------------------------------------------------------------
    -- Parsing
    --------------------------------------------------------------------------------
    registerP :: ReadP Register
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
ed)
```

```
ldP = do
    void $ string "ld"
    skipSpaces
    r <- registerP
    void $ string ","
    n <- manyTill (satisfy isDigit) eof
    return $ LD r (read n)

outP :: ReadP (Instruction Unresolved)
outP = do
    void $ string "out (0),"
    r <- registerP
    return $ Out r

rlcaP :: ReadP (Instruction Unresolved)
rlcaP = string "rlca" >> return RLCA

rrcaP :: ReadP (Instruction Unresolved)
rrcaP = string "rrca" >> return RRCA

labelP :: ReadP (Instruction Unresolved)
labelP = do
    lbl <- manyTill (satisfy isAlpha) (char ':')
    return $ Label (Ref lbl)

djnzP :: ReadP (Instruction Unresolved)
djnzP = do
    void $ string "djnz"
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
ha) eof

[Instruction Unresolved]
```

```
parseProgram =
    mapM (fmap fst . listToMaybe . readP_to_S instrP) . filter (not . null)
where
    instrP = skipSpaces *> choice parsers
    parsers =
        [ ldP
        , outP
        , rlcaP
        , rrcaP
        , djnzP
        , labelP
        ]
```

permalink **embed**

[–] **jnazario** 2 0 [S] 1 point 9 months ago

hey /u/IceDane just an FYI your submission was flagged as possible spam. you may want to check if you're shadowbanned due to the IP you used. i approved your submission, this looked like a FP. however i would hate to see you miss out on engagement due to that error.

permalink **embed** **parent**

[–] **IceDane** 0 0 1 point 9 months ago

Thanks! I am pretty sure I'm not shadowbanned and I was simply posting from home. Thanks for the heads up, though. I'll look into it!

permalink **embed** **parent**

[–] **smokeyrobot** 2 points 9 months ago*

Here is my Java version. I don't liek the String parsing that I did for int -> bit strings but I didn't know anything elegant in Java that would help me. I tried BitSets but management was still a pain. My solution would also easily extend to more registers and more instructions

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

gestions would be great.

```
led;
```

```
public class LedTest {

    static final int ASCII_PAD = 97;
    static int[] registers = new int[2];
    static int mark = 0;

    public static void main(String[] args){

        String[] instructionSet = {"  ld a,14",
                "  out (0),a",
                "  ld a,12",
                "  out (0),a",
                "  ld a,8",
                "  out (0),a",
                "  out (0),a",
                "  ld a,12",
                "  out (0),a",
                "  ld a,14",
                "  out (0),a"};

        String[] instructionSet2 = {"  ld b,3",
                "",
                "triple:",
                "  ld a,126",
                "  out (0),a",
                "  ld a,60",
                "  out (0),a",
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
{"  ld a,1",
                "  ld b,9",
                "",
                "loop:",
                "  out (0),a",
                "  rlca",
                "  djnz loop"};

        String[] instructionSet4 = {"  ld a,2",
                "  ld b,9",
                "",
                "loop:",
                "  out (0),a",
                "  rrca",
                "  djnz loop"};

        System.out.println("--------- Instruction Set 1 ---------");
        executeInstructions(instructionSet, instructionSet);

        System.out.println("--------- Instruction Set 2 ---------");
        executeInstructions(instructionSet2, instructionSet2);

        System.out.println("--------- Instruction Set 3 ---------");
        executeInstructions(instructionSet3, instructionSet3);

        System.out.println("--------- Instruction Set 4 ---------");
        executeInstructions(instructionSet4, instructionSet4);
    }

                                             s(String[] instructionSet, String[] originalInstructionSet){

                                             tionSet){
                                             )){
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
                        String[] r = instr.split(",");
                        r[0].charAt(r[0].length() - 1);
                        registers[r[0].charAt(r[0].length() - 1)-ASCII_PAD] = Integer.parseInt(r[1]);
                        break;
                    case OUTPUT:
                        System.out.println(outputRegister(registers, instr.charAt(instr.length()-1)-ASCII_PAD
                        break;
                    case LABELREF:
                        mark = index;
                        break;
                    case ROTATELEFT:
                        registers['a'-ASCII_PAD] = rotateBits(registers['a'-ASCII_PAD], true);
                        break;
                    case ROTATERIGHT:
                        registers['a'-ASCII_PAD] = rotateBits(registers['a'-ASCII_PAD], false);
                        break;
                    case DECREMENT_JUMP:
                        if(registers[1] - 1 > 0){
                            registers[1]--;
                            executeInstructions((String[])
                                    Arrays.copyOfRange(originalInstructionSet, mark + 1, originalInstructionSe
                        }
                        break;
                    default:
                        break;
                }
                index++;
            }
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
tring s){
```

```
                return Instruction.LOAD;
            } else if (s.contains("out")){
                return Instruction.OUTPUT;
            } else if (s.contains(":")){
                return Instruction.LABELREF;
            }  else if (s.contains("djnz")){
                return Instruction.DECREMENT_JUMP;
            }  else if (s.contains("rlca")){
                return Instruction.ROTATELEFT;
            }  else if (s.contains("rrca")){
                return Instruction.ROTATERIGHT;
            }
            return Instruction.UNKNOWN;
        }

        static String outputRegister(int[] regs, int regNum){

            String output = convertToEightBitString(regs[regNum]);
            output = output.replaceAll("0", ".").replaceAll("1", "*");
            return output;
        }

        static enum Instruction{
            LOAD,
            OUTPUT,
            LABELREF,
            ROTATELEFT,
            ROTATERIGHT,
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
olean left){
        String bits = convertToEightBitString(a);
        String newBits = null;

        if(left){
            newBits = bits.substring(1, bits.length()).concat(bits.substring(0,1));
        } else {
            newBits = bits.substring(bits.length()-1, bits.length()).concat(bits.substring(0, bits.length
        }
        return Integer.parseInt(newBits, 2);
    }

    static String convertToEightBitString(int num){
        String empty = "00000000";
        String bits = Integer.toBinaryString(num);
        return empty.substring(0, empty.length() - bits.length()) + bits;
    }
}
```

permalink **embed**

[–] **itsme86** 2 points 9 months ago*

C#

I might have over-engineered this a bit. Works for all challenges and even supports forward jumps too, so yay! Compiles the program into an intermediary bytecode that's used for executing the program.

```
class Program
{
    static void Main(string[] args)
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
am path: ");
```

```
            programPath = Console.ReadLine();
        }
        else
        {
            programPath = args[0];
        }

        byte[] compiled = new Assembler().Assemble(File.ReadAllLines(programPath));
        CpuContext context = new CpuContext(compiled);
        InstructionFactory factory = new InstructionFactory();
        while (context.InstructionPointer < context.Code.Length)
        {
            Instruction instruction = factory.MakeInstruction(context.Code[context.InstructionPointer++])
            instruction.Execute(context);
        }
    }
}

public enum Instructions : byte
{
    Load,
    Out,
    RLCA,
    RRCA,
    DJNZ
}

public abstract class Instruction
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
get; }

Code(AssemblerContext context);
puContext context);
```

```
}

public class LoadInstruction : Instruction
{
    public override Regex Regex => new Regex(@"^\s*ld\s*(?<reg>[ab]),(?<val>\d+)$", RegexOptions.Compiled

    public override byte[] GetByteCode(AssemblerContext context)
    {
        byte reg = (byte)(context.RegexMatch.Groups["reg"].Value[0] - 'a');
        int val = int.Parse(context.RegexMatch.Groups["val"].Value);
        if (val < 0 || val > 255)
            throw new Exception("Value must be between 0 and 255.");
        return new[] { (byte)Instructions.Load, reg, (byte)val };
    }

    public override void Execute(CpuContext context)
    {
        byte val = context.Code[context.InstructionPointer + 1];
        if (context.Code[context.InstructionPointer] == 0)
            context.A = val;
        else
            context.B = val;

        context.InstructionPointer += 2;
    }
}

public class OutInstruction : Instruction
```

< > discussions in r/dailyprogrammer X

43 · 24 comments

[17-08-21] Challenge #328 [Easy] Latin Squares

```
 new Regex(@"^\s*out\s*\(0\),(?<reg>[ab])$", RegexOptions.Compiled);

Code(AssemblerContext context)
```

```
            byte reg = (byte)(context.RegexMatch.Groups["reg"].Value[0] - 'a');
            return new[] { (byte)Instructions.Out, reg };
        }

        public override void Execute(CpuContext context)
        {
            byte val = context.Code[context.InstructionPointer++] == 0 ? context.A : context.B;

            StringBuilder sb = new StringBuilder(8);
            for (int mask = 1 << 7; mask > 0; mask >>= 1)
                sb.Append((val & mask) == 0 ? '.' : '*');

            Console.WriteLine(sb.ToString());
        }
    }

    public class RlcaInstruction : Instruction
    {
        public override Regex Regex => new Regex(@"^\s*rlca$", RegexOptions.Compiled);

        public override byte[] GetByteCode(AssemblerContext context)
        {
            return new[] { (byte)Instructions.RLCA };
        }

        public override void Execute(CpuContext context)
        {
            context.A = (byte)((context.A << 1) | (context.A >> 7));
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
truction
```

```
    public override Regex Regex => new Regex(@"^\s*rrca$", RegexOptions.Compiled);

    public override byte[] GetByteCode(AssemblerContext context)
    {
        return new[] { (byte)Instructions.RRCA };
    }

    public override void Execute(CpuContext context)
    {
        context.A = (byte)((context.A >> 1) | (context.A << 7));
    }
}

public class DjnzInstruction : Instruction
{
    public override Regex Regex => new Regex(@"^\s*djnz\s*(?<label>[A-Za-z_]+)$", RegexOptions.Compiled);

    public override byte[] GetByteCode(AssemblerContext context)
    {
        string label = context.RegexMatch.Groups["label"].Value;
        context.JumpList.RegisterJump((ushort)(context.InstructionStart + 1), label);
        return new byte[] { (byte)Instructions.DJNZ, 0, 0 };
    }

    public override void Execute(CpuContext context)
    {
        if (--context.B == 0)
            context.InstructionPointer += 2;

nter = (ushort)((context.Code[context.InstructionPointer] << 8) | contex
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
public class JumpList
{
    private readonly List<Jump> _jumps = new List<Jump>();
    private readonly Dictionary<string, ushort> _lookup = new Dictionary<string, ushort>();

    public void RegisterJumpPoint(string label, ushort offset)
    {
        _lookup[label] = offset;
    }

    public void RegisterJump(ushort source, string target)
    {
        _jumps.Add(new Jump(source, target));
    }

    public void Finalize(byte[] code)
    {
        foreach (Jump jump in _jumps)
        {
            ushort offset;
            if (!_lookup.TryGetValue(jump.Target, out offset))
                throw new Exception("Invalid jump to non-existent label " + jump.Target);
            code[jump.Source] = (byte)(offset >> 8);
            code[jump.Source + 1] = (byte)(offset & 0xFF);
        }
    }

    private class Jump
```

< > discussions in r/dailyprogrammer X

43 · 24 comments

[17-08-21] Challenge #328 [Easy] Latin Squares

```
; }
; }

 string target)
```

```
        {
            Source = source;
            Target = target;
        }
    }
}

public class InstructionFactory
{
    private readonly Instruction[] _instructions;
    private readonly Dictionary<Instructions, Instruction> _opcodeLookup = new Dictionary<Instructions, In

    public InstructionFactory()
    {
        Instruction loadInstruction = new LoadInstruction();
        Instruction outInstruction = new OutInstruction();
        Instruction rlcaInstruction = new RlcaInstruction();
        Instruction rrcaInstruction = new RrcaInstruction();
        Instruction djnzInstruction = new DjnzInstruction();

        _instructions = new [] { loadInstruction, outInstruction, rlcaInstruction, rrcaInstruction, djnzIn

        _opcodeLookup[Instructions.Load] = loadInstruction;
        _opcodeLookup[Instructions.Out] = outInstruction;
        _opcodeLookup[Instructions.RLCA] = rlcaInstruction;
        _opcodeLookup[Instructions.RRCA] = rrcaInstruction;
        _opcodeLookup[Instructions.DJNZ] = djnzInstruction;
    }
```

< > discussions in r/dailyprogrammer X

43 · 24 comments

[17-08-21] Challenge #328 [Easy] Latin Squares

```
tion(byte opcode)

alue((Instructions)opcode, out instruction))
```

```
            throw new Exception("Unknown opcode " + opcode);
        return instruction;
    }

    public Instruction MatchInstruction(string text, AssemblerContext context)
    {
        var res = _instructions.Select((i, idx) => new { Instruction = i, Match = i.Regex.Match(text) }).F
        if (res == null)
            throw new Exception("Parse error " + text);

        context.RegexMatch = res.Match;
        return res.Instruction;
    }
}

public class AssemblerContext
{
    public JumpList JumpList { get; } = new JumpList();

    public Match RegexMatch { get; set; }
    public ushort InstructionStart { get; set; }
}

public class Assembler
{
    private readonly Regex _labelPattern = new Regex(@"^\s*(?<label>[A-Za-z_]+):$", RegexOptions.Compiled)

    public byte[] Assemble(string[] lines)
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
new AssemblerContext();
= new InstructionFactory();
```

```
        using (MemoryStream stream = new MemoryStream())
        {
            foreach (string line in lines)
            {
                if (string.IsNullOrWhiteSpace(line))
                    continue;

                Match labelMatch = _labelPattern.Match(line);
                if (labelMatch.Success)
                {
                    context.JumpList.RegisterJumpPoint(labelMatch.Groups["label"].Value, context.Instruct
                    continue;
                }

                Instruction instruction = factory.MatchInstruction(line, context);
                byte[] bytes = instruction.GetByteCode(context);
                stream.Write(bytes, 0, bytes.Length);

                context.InstructionStart = (ushort)stream.Position;
            }

            compiled = stream.ToArray();
        }

        context.JumpList.Finalize(compiled);

        return compiled;
    }
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
er { get; set; }
```

```
    public byte[] Code { get; }

    public byte A { get; set; }
    public byte B { get; set; }

    public CpuContext(byte[] code)
    {
        Code = code;
    }
}
```

permalink embed

[–] adrian17 1 4 1 point 9 months ago*

Quick Python with decorators and regex-based matching.

Edit: just noticed that it's impossible to jump forward with the current approach (since the labels are read at the time of execution) :/

```
import re

instructions = {}

def instruction(regex):
    def decorator(func):
        instructions[regex] = func
        return func
    return decorator

class Machine:
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
        self.line = 0
        self.labels = {}
        self.lines = lines

    @instruction(" +ld (\w),(\d+)")
    def load(self, name, num):
        self.regs[name] = int(num)

    @instruction(" +out \(0\),a")
    def out(self):
        val = self.regs['a']
        result = ""
        for num in range(8):
            result = ("*" if val & (1<<num) else ".") + result
        print(result)

    @instruction(" +rlca")
    def rlca(self):
        self.regs['a'] = ((self.regs['a'] << 1) & 255) | (self.regs['a'] >> 7)

    @instruction(" +rrca")
    def rrca(self):
        self.regs['a'] = (self.regs['a'] >> 1) | ((self.regs['a'] << 7) & 255)

    @instruction("([a-zA-Z_]+):")
    def make_label(self, label):
        self.labels[label] = self.line
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
]+)")

s[label]
```

```
    def run(self):
        while self.line < len(lines):
            line = self.lines[self.line]
            for regex, func in instructions.items():
                match = re.match(regex, line)
                if match:
                    func(self, *match.groups())
            self.line += 1

lines = open("input.txt").read().splitlines()
machine = Machine(lines)
machine.run()
```

permalink embed

[–] lukz 2 0 2 points 9 months ago

TIL how decorators are written. Interesting feature.

permalink embed parent

[–] quakcduck 2 points 9 months ago

What is &amp and &lt?

permalink embed parent

[–] adrian17 1 4 1 point 9 months ago

oops, my <> characters were killed when I edited the comment on mobile. Fixed.

permalink embed parent

[–] quakcduck 1 point 9 months ago

ew that they represented an ampersand and less than sign but I wasn't sure why you used them in your

anks.

< > discussions in r/dailyprogrammer X

43 · 24 comments

[17-08-21] Challenge #328 [Easy] Latin Squares

Your post taught me how decorators work and thanks to that, I have a nice system now. Instead of making specific regexes, I made a general one that either matched a function with arguments or a tag.

Other than that, I only looked at your out function and everything else is made by me. (Mine used bin so it was a bit ugly) I'll probably finish rrca and rlca tommorow.

https://drive.google.com/open?id=0BwEEUbCO_OlceGFOYWZsT1Q4UU0

permalink embed parent

[–] adrian17 1 4 1 point 9 months ago

Why not post the whole code in a new comment?

permalink embed parent

[–] _HyDrAg_ 1 point 9 months ago

I was in a rush. It was swill wip but now it's done so I'm posting it.

permalink embed parent

[–] Daanvdk 1 0 1 point 9 months ago

Python 3: Loads the program from stdin as it goes so in theory should be able to execute infinitely long programs and in practice be able to execute your program directly while you're typing it in the command line. Also checks the input for errors and gives detailed error information when an error occurs.

```
from sys import stdin
from string import ascii_letters

def byteToBits(byte):
    return tuple(byte // 2**(7-i) % 2 for i in range(8))

def bitsToByte(bits):
                                      i] else 0 for i in range(8))


                                      (0)
```

< > discussions in r/dailyprogrammer x

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
for n, line in enumerate(stdin):
    if line.strip() != "":
        if line.lstrip() == line:
            if line.strip()[-1] == ":":
                label = line.strip()[:-1]
                if all(c in ascii_letters + "_" for c in label):
                    labels[line.strip()[:-1]] = len(program)
                else:
                    print("ERROR in line {}: {}\n`{}` is not a correct label, only a-z, A-Z and _ are all
                    break
            else:
                print("ERROR in line {}: {}\nInstruction missing whitespace or label missing colon.".forma
                break
        else:
            instruction, *args = line.split()
            if len(args) > 1:
                print("ERROR in line {}: {}\nUnexpected input after instruction.".format(n, line.strip())
                break
            args = args[0].split(",") if len(args) == 1 else []
            if instruction == "ld":
                if len(args) != 2:
                    print("ERROR in line {}: {}\nWrong number of arguments to ld, expected 2 got {}.".for
                    break
                if args[0] not in ["a", "b"]:
                    print("ERROR in line {}: {}\n`{}` is not an existing register, you can only use `a` a
                    break
                try:
                    args[1] = int(args[1])
                                         or args[1] > 255:
                                        eError()

                                        n line {}: {}\n`{}` is not a valid byte value, only integers from 0 unt
                program.append(("ld", args[0], int(args[1])))
            elif instruction == "out":
                if len(args) != 2:
                    print("ERROR in line {}: {}\nWrong number of arguments to out, expected 2 got {}.".for
                    break
                if args[0] != "(0)":
                    print("ERROR in line {}: {}\n`{}` is not a correct first argument for out, only `(0)`
                    break
                if args[1] != "a":
                    print("ERROR in line {}: {}\n`{}` is not a correct register for out, only `a` is allo
                    break
                program.append(("out", args[0], args[1]))
            elif instruction in ["rlca", "rrca"]:
                if len(args) != 0:
                    print("ERROR in line {}: {}\nWrong number of arguments to {}, expected 0 got {}.".form
                    break
                program.append((instruction,))
            elif instruction == "djnz":
                if len(args) != 1:
                    print("ERROR in line {}: {}\nWrong number of arguments to djnz, expected 1 got {}.".fo
                    break
                if not all(c in ascii_letters + "_" for c in args[0]):
                    print("ERROR in line {}: {}\n`{}` is not a correct label, only a-z, A-Z and _ are all
                    break
                if args[0] not in labels:
                    print("ERROR in line {}: {}\n`{}` is not an existing label.".format(n, line.strip(),
                    break
                program.append(("djnz", args[0]))
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
ne {}: {}\n`{}` is either an incorrect instruction or a label that has

:
```

```
                if program[ptr][1] == "a":
                    a = byteToBits(program[ptr][2])
                else:
                    b = byteToBits(program[ptr][2])
            elif program[ptr][0] == "out":
                print("".join("*" if led else "." for led in a))
            elif program[ptr][0] == "rlca":
                a = a[1:8] + a[0:1]
            elif program[ptr][0] == "rrca":
                a = a[7:8] + a[0:7]
            elif program[ptr][0] == "djnz":
                b_ = (bitsToByte(b) - 1) % 256
                if b_ != 0:
                    b = byteToBits(b_)
                    ptr = labels[program[ptr][1]] - 1
            ptr += 1
```

permalink **embed**

[–] **lukz** 2 0 2 points 9 months ago

At first I am a bit surprised how much effort you put into error reporting even though the challenge does not ask it. But it makes sense. If it would be a learning platform for beginners you want them to be told exactly what they input wrongly.

permalink **embed** **parent**

[–] **gfixler** 2 points 9 months ago

Upvoted for error message love.

permalink **embed** **parent**

[–] **asterite** 1 point 9 months ago

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
    triple:
      ld a,126
      out (0),a
      ld a,60
      out (0),a
      ld a,24
      out (0),a
      djnz triple
    CHALLENGE_2
  inputs << <<-CHALLENGE_3
      ld a,1
      ld b,9

    loop:
      out (0),a
      rlca
      djnz loop
    CHALLENGE_3
  inputs << <<-CHALLENGE_4
      ld a,2
      ld b,9

    loop:
      out (0),a
      rrca
      djnz loop
    CHALLENGE_4
```

```
record RRCA
record DJNZ, label : String

alias Instruction = LD_A | LD_B | OUT_A | RLCA | RRCA | DJNZ

inputs.each_with_index do |input, input_index|
  instructions = [] of Instruction
  labels = {} of String => Int32

  # Parse instructions and labels
  input.lines.each do |line|
    case line
    when /([a-zA-Z_]+):/     then labels[$1] = instructions.size
    when /ld\s+a,(\d+)/      then instructions << LD_A.new($1.to_u8)
    when /ld\s+b,(\d+)/      then instructions << LD_B.new($1.to_u8)
    when /out \(0\),a/       then instructions << OUT_A.new
    when /rlca/              then instructions << RLCA.new
    when /rrca/              then instructions << RRCA.new
    when /djnz ([a-zA-Z_]+)/ then instructions << DJNZ.new($1)
    end
  end

  puts "Challenge #{input_index + 2}:"
  puts

  # Execute program
  a = 0_u8
  b = 0_u8
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
        a = inst.value
      when LD_B
        b = inst.value
      when OUT_A
        7.downto(0) { |i| print a & (1 << i) == 0 ? '.' : '*' }
        puts
      when RLCA
        a = a << 1 | a >> 7
      when RRCA
        a = a >> 1 | a << 7
      when DJNZ
        b -= 1
        if b != 0
          i = labels[inst.label]
          next
        end
      end
      i += 1
    end

    puts
  end
```

https://play.crystal-lang.org/#/r/1d8b

permalink **embed**

[–] **marchelzo** 1 point 9 months ago*

Ty

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
t, Djnz, Halt;
```

```
let labels = {};

while let $line = read() {
        match line {
                /^(\w+):$/ ~> [_, label]        => { labels[label] = code.len(); },
                /^\s+ld a,(\d+)$/ ~> [_, num]   => { code.push(LoadA(int(num))); },
                /^\s+ld b,(\d+)$/ ~> [_, num]   => { code.push(LoadB(int(num))); },
                /^\s+djnz (\w+)$/ ~> [_, label] => { code.push(Djnz(label));     },
                /^\s+out \(0\),a$/              => { code.push(Out);             },
                /^\s+rlca$/                     => { code.push(Rlca);            },
                /^\s+rrca$/                     => { code.push(Rrca);            },
                _                               => {                             }
        }
}

code.push(Halt);

let [pc, a, b] = [0, 0, 0];

let bits = (..8).reverse!();
let set? = i -> and(a, shiftLeft(1, i)) != 0;

while match code[pc++] {
        LoadA(num)  => { a = num;                                                   },
        LoadB(num)  => { b = num;                                                   },
        Rlca        => { a = and(255, shiftLeft(a, 1) + and(a, 128) / 128);         },
        Rrca        => { a = and(255, shiftRight(a, 1) + shiftLeft(and(a, 1), 7));  },
        Djnz(label) => { if (--b != 0) pc = labels[label];                          },
                                  s.map(i -> '*' if set?(i) else '.').sum());      }
```

[deleted]

[–] **marchelzo** 2 points 9 months ago

It's a programming language that I designed. You can find the implementation here: https://github.com/marchelzo/ty

I'm basically the only user, and I haven't published any documentation or made a home page or anything for it, so it's pretty hard to find on Google.

permalink **embed**

[–] [deleted] 9 months ago

[deleted]

[–] **marchelzo** 1 point 9 months ago*

Thanks! I'm glad you like it. The implementation is kind of messy for two reasons:

1. I didn't know much about compiler or VM implementation when I started writing it.
2. The language was designed as I went, so lots of things were changed without proper refactoring, and due TO the organic growth, the implementation is kind of unnatural.

But if you do decide to check it out, and have any questions, feel free to message me.

permalink **embed**

[–] **quakcduck** 1 point 9 months ago

How does designing a language work? Is it just making a compiler and VM using another language?

permalink **embed** **parent**

[–] **marchelzo** 2 points 9 months ago

Pretty much. Mine is in C, and there's also a bunch of built-in functions that are implemented in C. I'm not sure if you'd consider those part of the VM or not.

permalink **embed** **parent**

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
(define-namespace-anchor na)
(define ns (namespace-anchor->namespace na))

(define regs (make-hash '((a . 0)
                          (b . 0))))

(define jumps (make-hash))

(define (ld reg num)
  (hash-set! regs reg num)
  #f)

(define (out n reg)
  (displayln (regexp-replaces
              (~r (hash-ref regs reg) #:base 2 #:min-width 8 #:pad-string "0")
              '([#rx"0" "."] [#rx"1" "*"])))
  #f)

(define (rotate-a l r)
  (define a (hash-ref regs 'a))
  (hash-set! regs 'a (bitwise-and (bitwise-ior (arithmetic-shift a l)
                                               (arithmetic-shift a r))
                                  #b11111111)))
(define (rlca)
  (rotate-a 1 -7)
  #f)
  (define b (sub1 (hash-ref regs 'b)))
  (hash-set! regs 'b b)
  (if (zero? b) #f (hash-ref jumps sym)))

(define (evaluate ops)
  (unless (empty? ops)
    (define next (apply (eval (caar ops) ns) (cdar ops)))
    (evaluate (if next next (cdr ops)))))

(define (parse lines)
  (define (loop ls ops)
    (cond
      [(empty? ls) ops]
      [(not (non-empty-string? (car ls))) (loop (cdr ls) ops)]
      [(string-suffix? (car ls) ":")
       (hash-set! jumps (string->symbol (string-trim (car ls) ":")) ops)
       (loop (cdr ls) ops)]
      [else (loop (cdr ls)
                  (cons
                   (call-with-input-string
                    (string-append "("
                                   (string-replace (car ls) #rx"\\(|\\)|," " ")
                                   ")")
                    read)
                   ops))]))
  (loop (reverse lines) '()))

(evaluate (parse (port->lines (current-input-port))))
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
import collection.mutable.Map
object LEDBlinker extends App {
  val ldA = """ld a,(\d{1,3})""".r
  val ldB = """ld b,(\d{1,3})""".r
  val djnz = """djnz (\w+)""".r
  val label = """(\w+):""".r
  val labels = Map.empty[String, Int]
  var line, a, b = 0
  val lines = File(args(0)).lines.toSeq.map(_.trim)

  while (line < lines.length) {
    lines(line) match {
      case ldA(num) => a = num.toInt
        line += 1
      case ldB(num) => b = num.toInt
        line += 1
      case "rlca" => val astr = a.toBinaryString.reverse.padTo(8, '0').reverse
        a = Integer.parseInt(astr.tail + astr.head, 2)
        line += 1
      case "rrca" => val astr = a.toBinaryString.reverse.padTo(8, '0').reverse
        a = Integer.parseInt(astr.last + astr.init, 2)
        line += 1
      case label(text) => labels(text) = line
        line += 1
      case djnz(text) => b -= 1
        if (b != 0) line = labels(text)
        else line += 1
      case "out (0),a" => println(a.toBinaryString.replace('0', '.').replace('1', '*').reverse.padTo(8, '
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

Going for small code instead of sanity here, as you can see from my rlca and rrca implementations.

permalink **embed**

[–] [deleted] 9 months ago*

[deleted]

[–] **smokeyrobot** 1 point 9 months ago

FYI Notepad++ does vertical selection so you can space entire blocks of code by pressing and holding alt+tab and using up/down keys to extend cursor. Super handy for mundane code editing like adding quotes to long strings etc. I can't get to your Github link but I posted my Java solution here and would be curious to see the comparison.

permalink **embed**

[–] [deleted] 9 months ago*

[deleted]

[–] **smokeyrobot** 1 point 9 months ago

I pasted mine without issue. The link is probably fine just filtered out by my proxy.

permalink **embed**

[–] **ASpueW** 1 point 9 months ago*

Rust

```
use std::str::FromStr;
use std::io::Write;
use std::io::stdout;

type Res<T> = Result<T, &'static str>;

#[derive(Debug, Copy, Clone)]
enum Code{ LdA(u8), LdB(u8), Rrca, Rlca, Djnz, Label, Out }
```

< > discussions in r/dailyprogrammer X

43 · 24 comments

[17-08-21] Challenge #328 [Easy] Latin Squares

```
<Code>{
,") { (&inp[5..]).trim().parse::<u8>()
```

```
                                                    .map(|val| Code::LdA(val))
                                                    .map_err(|_| "ld a command value parsing failed") }
        else if inp.starts_with("ld b,") { (&inp[5..]).trim().parse::<u8>()
                                                        .map(|val| Code::LdB(val))
                                                        .map_err(|_| "ld b command value parsing failed")
        else if inp == "out (0),a" { Ok(Code::Out) }
        else if inp == "rrca" { Ok(Code::Rrca) }
        else if inp == "rlca" { Ok(Code::Rlca) }
        else if inp.starts_with("djnz") { Ok(Code::Djnz) }
        else if inp.ends_with(':') { Ok(Code::Label) }
        else{ Err("code parsing failed") }
    }
}

struct Program{
    mem: Vec<Code>,
    pc: usize,
    reg: [u8;2],
    lbl: Option<usize>
}

impl Program{
    fn new(mem:Vec<Code>) -> Program{
        Program{pc:0, mem:mem, reg:[0;2], lbl:None }
    }

    fn step<W>(&mut self, dst:W) -> bool
    where W: Write
```

```
ed().map(|code| {

{ self.reg[0] = val; self.pc + 1 }
{ self.reg[1] = val; self.pc + 1 }
```

```
                Code::Out => { show(self.reg[0], dst); self.pc + 1 }
                Code::Rrca => { self.reg[0] = self.reg[0].rotate_right(1); self.pc + 1 }
                Code::Rlca => { self.reg[0] = self.reg[0].rotate_left(1); self.pc + 1 }
                Code::Djnz => {
                    self.reg[1] -= 1;
                    if self.reg[1] != 0 { self.lbl.expect("label undefined") }else{ self.pc + 1 }
                }
                Code::Label => { self.lbl = Some(self.pc + 1); self.pc + 1 }
            };
            true
        })
        .unwrap_or(false)
    }

    fn run<W>(&mut self, mut dst:W)
    where W: Write
    {
        while self.step(dst.by_ref()) {}
    }
}

impl FromStr for Program{
    type Err = &'static str;
    fn from_str(inp:&str) -> Res<Program>{
        inp.lines()
            .map(|line| line.trim())
            .filter(|line| !line.is_empty())
            .map(|line| line.parse::<Code>())

                                         w(mem))
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
fn show<W>(val:u8, mut dst:W)
where W: Write
{
    for mask in (0..8).map(|x| 0x80u8 >> x ) {
        write!(dst.by_ref(), "{}", if val & mask == 0 {'.'} else {'*'}).unwrap();
    }
    writeln!(dst, "").unwrap();
}



fn main() {
    let sample ="   ld b,3

                    triple:
                      ld a,126
                      out (0),a
                      ld a,60
                      out (0),a
                      ld a,24
                      out (0),a
                      djnz triple
                    ";
    match sample.parse::<Program>() {
        Ok(mut cpu) => cpu.run(stdout()),
        Err(e) => println!("ERR: {}", e)
    };
    for (&smp, &chk) in SAMPLES.iter().zip(RESULTS){
        let mut cpu:Program = smp.parse().unwrap();
        let mut res:Vec<u8> = Vec::new();
        cpu.run(&mut res);
        assert_eq!(res, chk);
    }
}

static SAMPLES:&'static [&'static str] = &[
" ld a,14
  out (0),a
  ld a,12
  out (0),a
  ld a,8
  out (0),a

  out (0),a
  ld a,12
  out (0),a
  ld a,14
  out (0),a
",
" ld b,3

triple:
  ld a,126
  out (0),a
  ld a,60
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
" ld a,1
  ld b,9

loop:
  out (0),a
  rlca
  djnz loop
",
" ld a,2
  ld b,9

loop:
  out (0),a
  rrca
  djnz loop
"
];

static RESULTS:&'static [&'static [u8]] = &[
b"....***.
....**..
....*...
....*...
....**..
....***.
",
b".******.
..****..
```

```
..****..
...**...
",
b".......*
......*.
.....*..
....*...
...*....
..*.....
.*......
*.......
.......*
",
b"......*.
.......*
*.......
.*......
..*.....
...*....
....*...
.....*..
......*.
"
];
```

permalink embed

[–] Specter_Terrasbane 1 point 9 months ago

Python 2, both parts

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
ict
```

```
        self._a = self._b = 0
        self._instructions = map(self._parse, program.splitlines())
        self._labels = {}

    def _parse(self, line):
        attempts = ((func, pattern.match(line)) for (func, pattern) in LedState._CMDS.iteritems())
        return next((func, match.group(1)) for func, match in attempts if match)

    def _loadA(self, n, line_no):
        self._a = int(n) % 256
        return line_no + 1

    def _loadB(self, n, line_no):
        self._b = int(n) % 256
        return line_no + 1

    def _out(self, __, line_no):
        print ''.join('.*'[(self._a >> i) & 1] for i in xrange(7, -1, -1))
        return line_no + 1

    def _rlca(self, __, line_no):
        self._a = ((self._a << 1) % 256) + ((self._a >> 7) & 1)
        return line_no + 1

    def _rrca(self, __, line_no):
        self._a = (self._a >> 1) + ((self._a & 1) << 7)
        return line_no + 1

                               ):
                                1)
                                if self._b else line_no + 1

                               ):
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
            ret = self._labels[label] = line_no + 1
            return ret

        def _nop(self, __, line_no):
            return line_no + 1

        def _err(self, line, line_no):
            print 'Unknown command on line {}: {}'.format(line_no, line)
            return -1

        def run(self):
            i, n = 0, len(self._instructions)
            while 0 <= i < n:
                func, arg = self._instructions[i]
                i = func(self, arg, i)

        _CMDS = OrderedDict((
            (_loadA, re.compile(r'\A\s*ld a,(\d+)\Z')),
            (_loadB, re.compile(r'\A\s*ld b,(\d+)\Z')),
            (_out, re.compile(r'\A\s*out \(0\),a()\Z')),
            (_rlca, re.compile(r'\A\s*rlca()\Z')),
            (_rrca, re.compile(r'\A\s*rrca()\Z')),
            (_djnz, re.compile(r'\A\s*djnz ([a-zA-Z_]+)\Z')),
            (_label, re.compile(r'\A\s*([a-zA-Z_]+):\Z')),
            (_nop, re.compile(r'\A()\Z')),
            (_err, re.compile(r'\A(.*)\Z')),
        ))
```

< > discussions in r/dailyprogrammer X

)

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

Python 3, all challenges. This was a fun, but I hate Python's lack of case handling like Java has.

+/u/CompileBot Python3

```
# Blinking LEDs from Daily Programmer, Challenge 290
# By: den510


def main(input_one, input_two, input_three, input_four):
    output_binary, output_binary_two, output_binary_three, output_binary_four = generate_output(input_one)
    print("Challenge One:")
    emulate_lights(output_binary)
    print("\nChallenge Two:")
    emulate_lights(output_binary_two)
    print("\nChallenge Three:")
    emulate_lights(output_binary_three)
    print("\nChallenge Four:")
    emulate_lights(output_binary_four)


def generate_output(data):
    return_list, instructions, a, b = [], data.splitlines(), '', 0
    for line in instructions:
        if line and line[0] == line[1] == ' ':
            if 'ld a' in line:
                a = format(int(line.split(',')[1]), '08b')
            elif 'ld b' in line:
                b = int(line.split(',')[1])
            elif 'out' in line:
                                    nd(a)
```

< > discussions in r/dailyprogrammer X

43 · 24 comments

[17-08-21] Challenge #328 [Easy] Latin Squares

```
            elif 'rlca' in line:
                a = rca(a, 'l')
        if line and line[0] != ' ':
            trigger, triggered = line, False
            while b > 0:
                for second_line in instructions:
                    if second_line == trigger:
                        triggered = True
                    if triggered:
                        if 'ld a' in second_line:
                            a = format(int(second_line.split(',')[1]), '08b')
                        elif 'out' in second_line:
                            return_list.append(a)
                        elif 'djnz' in second_line:
                            b -= 1
                            triggered = False
                        elif 'rrca' in second_line:
                            a = rca(a, 'r')
                        elif 'rlca' in second_line:
                            a = rca(a, 'l')
            break
    return return_list


def emulate_lights(data):
    for line in data:
        print(line.replace('0', '.').replace('1', '*'))
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
            return_string += binary[i-1]
        if direction == 'l':
            return_string += binary[i-len(binary)+1]
    return return_string if direction == 'l' or direction == 'r' else binary


challenge_one = """  ld a,14\n  out (0),a\n  ld a,12\n  out (0),a\n  ld a,8\n  out (0),a\n
  out (0),a\n  ld a,12\n  out (0),a\n  ld a,14\n  out (0),a"""
challenge_two = """  ld b,3
\ntriple:\n  ld a,126\n  out (0),a\n  ld a,60\n  out (0),a\n  ld a,24\n  out (0),a\n  djnz triple"""
challenge_three = """  ld a,1\n  ld b,9\n\nloop:\n  out (0),a\n  rlca\n  djnz loop"""
challenge_four = """  ld a,2\n  ld b,9\n\nloop:\n  out (0),a\n  rrca\n  djnz loop"""
main(challenge_one, challenge_two, challenge_three, challenge_four)
raise SystemExit()
```

permalink **embed**

[–] **CompileBot** 1 point 9 months ago

Output:

```
Challenge One:
....***.
....**..
....*...
....*...
....**..
....***.

Challenge Two:
 ******
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
.******.
..****..
...**...

Challenge Three:
.......*
......*.
.....*..
....*...
...*....
..*.....
.*......
*.......
.......*

Challenge Four:
......*.
.......*
*.......
.*......
..*.....
...*....
....*...
.....*..
......*.
```

source | info | git | report

permalink **embed** **parent**

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

ava has.

n can do it with a dictionary that maps the control variable to a function.

```
    pass

def op_print():
    pass

switch = {
    "load": op_load,
    "print": op_print
}
command = 'load'  # from parser
switch[command]()  # This is a function call
```

edit: This solution is a complete implementation.

permalink **embed** **parent**

[–] **den510** 1 point 9 months ago

Thanks! I had seen something similar, but hadn't wanted to create methods for everything. Good to know this is an option though :)

permalink **embed** **parent**

[–] **thtoeo** 1 point 9 months ago

C#. Created little variation which allows using more variables and using commands on all variables. Allows also to use loop inside loop.

Commands:

```
ld <var>,<value> : loads value to variable
out (0),<var> : prints variable state
rlc <var> : shifts variable bits to left
rrc <var> : shifts variable bits to right
loop <var> / djnz : loops contents <var> times, djnz ends loop
```

Example 1 (loop with rotate):

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
loop b
  out (0),a
  rlc a
djnz

OUTPUT:

.......*
......*.
.....*..
....*...
...*....
..*.....
.*......
*.......
.......*
```

Example 2 (loop inside loop):

```
INPUT:

ld a,1
ld b,3
ld c,7
ld d,15

loop b
    out (0),a
    out (0),d
djnz

OUTPUT:

.......*
......**
.....***
.....***
.....***
....****
.......*
......**
.....***
.....***
.....***
....****
.......*
......**
.....***
.....***
.....***
....****
```

Code:

```
public static void RunCommands(string commands)
{
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
.NewLine }, StringSplitOptions.None)
```

```
            .Where(x => x.Length > 0)
            .ForEach(x => machine.Execute(x));
    }

    public class Machine
    {
        private readonly Dictionary<string, byte> _memory = new Dictionary<string, byte>();
        private readonly List<Commands.Loop> _loops = new List<Commands.Loop>();

        public byte GetMemory(string key)
        {
            if (_memory.ContainsKey(key))
            {
                return _memory[key];
            }

            const byte array = new byte();
            _memory.Add(key, array);
            return array;
        }

        public void SetMemory(string key, byte value)
        {
            if (_memory.ContainsKey(key))
            {
                _memory[key] = value;
            }
            else
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
);
```

```
    public void Execute(string line)
    {
        var parameters = line.Split(' ');

        ICommand command = null;

        switch (parameters[0])
        {
            case "ld":
                command = new Commands.Load(parameters[1]);
                break;

            case "out":
                command = new Commands.Out(parameters[1]);
                break;

            case "rlc":
                command = new Commands.Rlc(parameters[1]);
                break;

            case "rrc":
                command = new Commands.Rrc(parameters[1]);
                break;

            case "loop":
                command = new Commands.Loop(parameters[1]);
                break;

                                                Last();
                                                ;
                                                );
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
        }

        if (command != null)
        {
            if (_loops.Count > 0)
            {
                _loops[_loops.Count - 1].AddCommand(command);
            }

            var loop = command as Commands.Loop;

            if (loop != null)
            {
                _loops.Add(loop);
            }
            else
            {
                command.Execute(this);
            }
        }
    }
}

public interface ICommand
{
    void Execute(Machine machine);
}
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
ey;
```

```
        private readonly byte _value;

        public Load(string input)
        {
            var parameters = input.Split(',');
            _key = parameters[0];
            _value = (byte)int.Parse(parameters[1]);
        }

        public void Execute(Machine machine)
        {
            machine.SetMemory(_key, _value);
        }
    }

    public class Out : ICommand
    {
        private readonly string _key;

        public Out(string input)
        {
            var parameters = input.Split(',');
            _key = parameters[1];
        }

        public void Execute(Machine machine)
        {
            var value = machine.GetMemory(_key);
                                                lder();

                                                ; i--)

                                                 (1 << i)) != 0 ? "*" : ".");
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
            }

            Console.WriteLine(sb.ToString());
        }
    }

    public class Rlc : ICommand
    {
        private readonly string _key;

        public Rlc(string input)
        {
            _key = input;
        }

        public void Execute(Machine machine)
        {
            var value = machine.GetMemory(_key);
            machine.SetMemory(_key, (byte)(value << 1 | value >> 7));
        }
    }

    public class Rrc : ICommand
    {
        private readonly string _key;

        public Rrc(string input)
        {
```

```
e machine)
```

```
                var value = machine.GetMemory(_key);
                machine.SetMemory(_key, (byte)(value << 7 | value >> 1));
            }
        }

        public class Loop : ICommand
        {
            private readonly string _key;
            private readonly List<ICommand> _commands = new List<ICommand>();

            public Loop(string input)
            {
                _key = input;
            }

            public void AddCommand(ICommand command)
            {
                _commands.Add(command);
            }

            public void Execute(Machine machine)
            {
                RunCommands(machine, false);
            }

            public void RunCommands(Machine machine, bool reset)
            {
                var count = machine.GetMemory(_key);
                for (var i = 0; i < count; i++)
                {
                    _commands.ForEach(x =>
                    {
                        var loop = x as Loop;
                        if (loop != null)
                        {
                            loop.RunCommands(machine, true);
                        }
                        else
                        {
                            x.Execute(machine);
                        }
                    });
                }
            }
        }
    }
```

permalink **embed**

[–] **Minolwa** 1 point 9 months ago

Functional Python3 w/ Closures

- Implemented as an interpreter
- Infinite amount of registers allowed
- All registers may be shifted
- Command line input is the file to be interpreted

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
registers = dict()

def load(register, num):
    registers[register] = num

def out(register):
    def padleft(string, size=8):
        if len(string) >= size:
            return string
        else:
            return padleft('0' + string, size)

    def tosymbol(char):
        if char == '0':
            return '.'
        elif char == '1':
            return '*'

    binary = padleft(str(bin(registers[register]))[2:])
    print(''.join([tosymbol(x) for x in binary]))

def jump(tag):
    global counter
    if registers['b'] > 1:
        counter = tags[tag]
        registers['b'] -= 1
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
 func(registers[register], 1)
] < 1:
r] = 128
```

```
        elif registers[register] > 128:
            registers[register] = 1

    return realshift

commands = {
    'ld': load,
    'out': out,
    'djnz': jump,
    'rlc': shift(operator.lshift),
    'rrc': shift(operator.rshift)
}
tags = dict()
counter = 0

def startprogram(commandlist):
    global counter

    def mapints(string):
        if string.isdigit():
            return int(string)
        else:
            return string

    def execute(command):
        command = (command[0], [mapints(integer) for integer in command[1]])
        if command[1][0] == '(0)':
            command = (command[0], [command[1][1]])
                              *command[1])

                              ndlist):
                              ounter])
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
def interpret(filelines):
    def recordshift(info):
        args = '1'
        if info[0].startswith('r'):
            args = (info[0][-1:])
            info[0] = info[0][:3]
        return info, args

    def recordinstruction(fileline):
        info = fileline[2:].split(' ')
        if len(info) > 1:
            args = info[1].split(',')
        else:
            info, args = recordshift(info)
        return info[0], args

    def recordtag(fileline):
        tags[fileline[:-1]] = filelines.index(fileline) - 2

    def interpretline(fileline):
        if fileline.startswith(' '):
            return recordinstruction(fileline)
        else:
            recordtag(fileline)

    commands_in_file = [interpretline(fileline) for fileline in filelines]
    commands_in_file = [x for x in commands_in_file if x is not None]
                          file)
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
                          [1]), 'r') as f:
```

```
    lines = [line.replace('\n', '') for line in lines]
    interpret(lines)
```

permalink embed

[–] Scroph 0 0 1 point 9 months ago

D (dlang) solution. Most of the work is done inside the Cpu structure. Throws if it runs into an unknown instruction or an undefined label :

```
import std.stdio;
import std.string;
import std.format;

int main(string[] args)
{
    if(args.length < 2)
    {
        writeln("Usage : ", args[0], " <program>");
        return 0;
    }
    auto cpu = Cpu(File(args[1]));
    while(true)
        if(!cpu.cycle)
            break;
    return 0;
}

struct Cpu
{
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
] dispatcher;
```

```
    this(File fh)
    {
        pc = 0;
        string line;
        while((line = fh.readln) !is null)
            if(line.strip.length)
                program ~= line.strip;
        dispatcher["ld"] = &ld_cb;
        dispatcher["out"] = &out_cb;
        dispatcher["rlca"] = &rlca_cb;
        dispatcher["rrca"] = &rrca_cb;
        dispatcher["djnz"] = &djnz_cb;
    }

    bool cycle()
    {
        if(pc >= program.length)
            return false;
        string instruction = program[pc];
        auto space = instruction.indexOf(" ");
        string opcode = space != -1 ? instruction[0 .. space] : instruction;
        auto callback = dispatcher.get(opcode, &generic_cb);
        callback(instruction);
        pc++;
        return true;
    }
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
]] = pc;
```

```
                throw new Exception("Unknown instruction : ", line);
        }

        void ld_cb(string line)
        {
            char reg;
            int n;
            line.formattedRead("ld %c,%d", &reg, &n);
            registers[reg] = n;
        }

        void out_cb(string line)
        {
            for(int mask = 0b10000000; mask; mask >>= 1)
                write(registers['a'] & mask ? '*' : '.');
            writeln;
        }

        void rrca_cb(string line)
        {
            bool rightmost = registers['a'] & 1 ? true : false;
            registers['a'] >>= 1;
            if(rightmost)
                registers['a'] |= 1 << 7;
        }

        void rlca_cb(string line)
        {
                                                  'a'] & (1 << 7) ? true : false;
```

```
    void djnz_cb(string line)
    {
        if(--registers['b'])
        {
            string label = line[line.indexOf(" ") + 1 .. $];
            if(label !in labels)
                throw new Exception("Unknown label : " ~ label);
            pc = labels[label];
        }
    }
}
```

permalink **embed**

[–] **Haizan** 1 point 9 months ago

C. I wanted an excuse to play around with some code generation. So this code builds a function from machine code and executes that.

Code is here it got a bit long for a comment.

This supports different sizes for the LED array (8/16/32) and has an alternate print function. Instead of printing ever "out" on a separate line it animates the changes on one line.

permalink **embed**

[–] **altorelievo** 1 point 9 months ago*

Recursive method with Python

```
#!/usr/bin/env python


def parse_program(fname):
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
nes():
ne.split()
```

```
                if not instruction: continue
                elif len(instruction) < 2:
                    program.append(instruction)
                else:
                    program.append(instruction[:1] +
                                   instruction[1].split(','))
    except OSError:
        print('File {} does not exist'.format(fname))
    return program

def set_register(n):
    r = bin(int(n))[2:]
    return '0' * (8 - len(r)) + r

print_register = lambda r: print(r.replace('0', '.').replace('1', '*'))
left_rotate = lambda r: r[1:] + r[0]
right_rotate = lambda r: r[-1] + r[:-1]

def run_program(program, registers):
    labels = {}
    for line_no, line in enumerate(program):
        if line[0].endswith(':'):
            labels[line[0].strip(':')] = program[line_no:]
        elif line[0] == 'ld':
            if line[1] == 'a':
                registers['a'] = set_register(line[2])
            else:
                registers['b'] = int(line[2])
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
ters[line[2]]
rs[LED])

_rotate(registers['a'])
```

```
        elif 'rlc' in line[0]:
            registers['a'] = left_rotate(registers['a'])
        elif line[0] == 'djnz':
            registers['b'] -= 1
            if registers['b'] < 1: continue
            run_program(labels[line[1]], registers)
        else:
            print('Invalid instruction {}'.format(str(line)))
            return


if __name__ == '__main__':
    LED = '(0)'
    base_file_name = 'input{}.txt'
    for i in range(1, 5):
        print('Running program-{}'.format(i))
        program1 = parse_program('input{}.txt'.format(i))
        run_program(program1, {LED:0})
```

permalink **embed**

[–] **_HyDrAg_** 1 point 9 months ago*

Python 3. The idea of using regex and decorators comes from /u/adrian17. Currently reads labels on init. It could be easily edited to read them when it's interpreting, but it would lose the ability to jump forwards. Because it reads labels in advance, each label name should be unique.

```
import re


commands = {}
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
        return f
    return decorator

# Note: closes the file given to it.
class Masinka:
    def __init__(self, file):
        self.regs = {
            "a": 0,
            "b": 0
        }
        self.labels = {}
        self.pattern = re.compile(r'\s+(\w+) ?((?:[\w()]+,?)*)')
        self.label_pattern = re.compile(r'([A-Za-z_]+):')
        self.leds = "."*8
        self.file = file.readlines()
        file.close()
        # Labels read in advance for jumping forwards.
        for index, line in enumerate(self.file):
            regex = self.label_pattern.match(line)
            if regex:
                self.labels[regex.groups()[0]] = index

    @naming("ld")
    def set(self, reg, num):
        self.regs[reg] = int(num)

    @naming("out")
    def out(self, _, reg): # _ is the (0) argument to the out command
```

```
gs[reg] & 2**bit else ".") + leds
ate of leds for some reason...
```

```
            print(leds)

        # I added this command to show how easy this is to extend.
        # It's not useful in any way since it's a primitive and inefficient
        # version of assembly but that's not the point.
        @naming("add")
        def add(self, reg, num):
            if num.isalpha():
                n = self.regs[num]
            else:
                n = int(num)
            self.regs[reg] += n


        def shift(self, reg, direction):
            orig = 2**7
            dest = 1
            if direction == "r":
                orig, dest = dest, orig

            if self.regs[reg] & orig:
                set_bit = True
            else:
                set_bit = False

            if direction == "r":
                self.regs[reg] >>= 1
            else:
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
    @naming("rlca")
    def shift_left(self, reg="a"):
        self.shift(reg, "l")

    @naming("rrca")
    def shift_right(self, reg="a"):
        self.shift(reg, "r")

    @naming("djnz")
    def djnz(self, label):
        # It's nice that this jumps to the label and then the index increment
        # gets it to the first command directly without wasting time resetting
        # the label.
        # This comment only made sense before I changed the labels to only
        # be set on init.
        self.regs["b"] -= 1
        if self.regs["b"] != 0:
            self.index = self.labels[label]

    def run(self):
        # This seems like too much nesting but the alternatives like doing
        # 'if not regex: return None' seem worse.
        self.index = 0
        while self.index < len(self.file):
            line = self.file[self.index]
            regex = self.pattern.match(line)
            if regex:
                command, args = regex.groups()
                                ne

                                nd](self, *args.split(","))
                                be rewritten
```

< > discussions in r/dailyprogrammer X

43 · 24 comments

[17-08-21] Challenge #328 [Easy] Latin Squares

```
                    commands[command](self)

            self.index += 1


m = Masinka(open("290.txt", "r"))
m.run()
```

permalink **embed**

[–] **demreddit** 1 point 9 months ago*

Python 3.5. I built a little menu based program that includes some basic options like viewing the current led program and repeating it if desired, along with of course creating a new one. Sorry, no built in editing! I have no doubt this program is very breakable, but I got the challenge outputs to work and I'm extremely happy for that! Any comments or criticisms are welcome and appreciated. :)

```
def buildCommandList():
    '''
    Returns a list of lists and dictionaries containing commands.
    '''
    progInputList = []
    progInputListList = []

    def buildLoop(loopName, progInputList = []):
        while True:
            progInput = input('>>> ')
            if progInput == '':
                progInput = '\n'
                progInputList.append(progInput)
            elif progInput[0] == ' ':
                                   ' djnz ' + loopName[0:-1]:
                                   t.append(progInput)
                                   oopName: progInputList}
                                   ic
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
                    progInputList.append(progInput)
            elif progInput[0] != ' ' and progInput[-1] == ':':
                progInputList.append(buildLoop(progInput, progInputList = []))
            else:
                print("Invalid input.")

    while True:
        progInput = input('>>> ')
        if progInput == '':
            progInput = ''
            progInputList.append(progInput)
        elif progInput[0] == ' ':
            if progInput == ' end':
                progInputList.append(progInput)
                progInputListList.append(progInputList)
                break
            else:
                progInputList.append(progInput)
        elif progInput[0] != ' ' and progInput[-1] == ':':
            progInputListList.append(progInputList)
            progInputList = []
            progInputListList.append(buildLoop(progInput))
        else:
            print("Invalid input.")

    return progInputListList

def viewCommandList(L, commandList = ''):

                                             aries containing commands.
                                             ll commands in the form of a line-broken string.
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
        if type(i) == str:
            if i == ' end':
                commandList += i
            else:
                commandList += i + '\n'
        elif type(i) == list:
            commandList += viewCommandList(i, commandList = '')
        elif type(i) == dict:
            for k in i.keys():
                commandList += k + '\n'
            commandList += viewCommandList(i.values(), commandList = '')

    return commandList

def interpretCommandList(L):
    '''
    L: A list of lists and dictionaries containing commands.
    Returns each individual command in L and applies to makeLights function for final output.
    '''
    for i in L:
        if type(i) == str:
            if i[0:6] == ' ld a,':
                global a
                a = int(i[6:])
            elif i[0:6] == ' ld b,':
                global b
                b = int(i[6:])
            elif i == ' rlca':
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
                if a > 1:
                    a = a>>1
                elif a == 1:
                    a = 128
            elif i == ' out (0),a':
                print(makeLights(a))
            elif i == '' or i == ' end' or ' djnz' in i:
                pass
            else:
                print('Invalid input.')
        elif type(i) == list:
            interpretCommandList(i)
        elif type(i) == dict:
            for n in range(b):
                interpretCommandList(i.values())

def makeLights(n):
    '''
    n: An integer between 0 and 255
    Returns a light pattern, mapped to 8 digit binary conversion of n.
    '''
    if n < 0 or n > 255:
        return 'Out of range.'
    bin_n = str(bin(n))[2:]
    bin_n = ('0' * (8 - len(bin_n))) + bin_n
    leds = ''
    for c in bin_n:
        if c == '0':
```

```
def led_program():
    '''
    A very basic menu for building led programs, viewing them, and running them. Sorry, no built in editi
    '''
    while True:
        toDo = input("What to do? 'n' = new program, 'v' = view current program, 'a' = run current progra
        if toDo == 'n':
            currentCommandList = buildCommandList()
            interpretCommandList(currentCommandList)
        elif toDo == 'v':
            try:
                print(viewCommandList(currentCommandList))
            except:
                print('No current command list.')
        elif toDo == 'a':
            try:
                interpretCommandList(currentCommandList)
            except:
                print('No current command list.')
        elif toDo == 'q':
            break
        else:
            print("That's not an option.")

led_program()
```

permalink **embed**

[–] **marcelo_rocha** 1 point 9 months ago

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
  ld a,8
  out (0),a

  out (0),a
  ld a,12
  out (0),a
  ld a,14
  out (0),a""";

const prog2 = """  ld b,3

triple:
  ld a,126
  out (0),a
  ld a,60
  out (0),a
  ld a,24
  out (0),a
  djnz triple""";

const prog3 = """  ld a,1
  ld b,9

loop:
  out (0),a
  rlca
  djnz loop""";
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
  rrca
  djnz loop""";

final rxLabel = new RegExp(r'(\w+):');
final rxLoad = new RegExp(r'\s+ld\s(a|b),(\d+)');
final rxOut = new RegExp(r'\s+out\s\(0\),a');
final rxJump = new RegExp(r'\s+djnz\s(\w+)');
final rxRot = new RegExp(r'\s+(rlca|rrca)');

void interpret(List<String> program) {
  var match;
  int ra = 0, rb = 0;
  int pc = 0;
  while(pc < program.length) {
    String line = program[pc];
    if ((match = rxLoad.firstMatch(line)) != null) {
      var v = num.parse(match.group(2));
      match.group(1) == 'a' ? ra = v: rb = v;
    }
    else if ((match = rxOut.firstMatch(line)) != null) {
      print(ra.toRadixString(2).padLeft(8, "0").replaceAll("0", ".").replaceAll("1", "*"));
    }
    else if ((match = rxRot.firstMatch(line)) != null) {
      if(match.group(1) == 'rlca') {
        ra = ((ra << 1) & 0xff) | ((ra >= 0x80 ? 1 : 0));
      }
      else {
        ra = (ra >> 1) | (ra.isOdd ? 0x80: 0);
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
Match(line)) != null) {

.group(1) + ':');
```

```
      }
    }
    pc++;
  }
}

void main() {
  var program = new List<String>();
  new RegExp('.+').allMatches(prog1).forEach((m)=>(program.add(m.group(0))));
  interpret(program);
}
```

permalink **embed**

[–] **[deleted]** 1 point 9 months ago*

Python. Works if there are no unlabeled instructions after the first unlabeled block which the challenge inputs contain.

Input filename and recursion depth are acquired via command line arguments.

```
import sys


def LEDConv(x):
    if x is '0':
        return '.'
    elif x is '1':
        return '*'

def binStrConv(a):
    binrep = bin(a)[2:]
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
r x in zeroList])
```

```
def LEDRep(a):
    binrep = binStrConv(a)
    return ''.join([LEDConv(x) for x in binrep])

def rlca(a):
    binrep = binStrConv(a)
    rotated = binrep[1:] + binrep[0]
    return int(rotated, 2)

def rrca(a):
    binrep = binStrConv(a)
    lastIndex = len(binrep) - 1
    rotated = binrep[lastIndex] + binrep[0:lastIndex]
    return int(rotated, 2)

def process(a, b, allLines, label, countdown):
    lines = allLines[label]
    if countdown is 0:
        return a,b
    for line in lines:
        if len(line) is 1:
            continue
        splitLines = line.split()
        command = splitLines[0]
        if command == "ld":
            data = splitLines[1]
            target,val = data.split(",")
        elif command == "out":
            print(LEDRep(a))

        elif command == "rlca":
            a = rlca(a)

        elif command == "rrca":
            a = rrca(a)

        elif command == "djnz":
            nextLabel = splitLines[1]
            return process(a, b, allLines, nextLabel, countdown - 1)
    return a,b

def main():
    filename = sys.argv[1]
    depth = sys.argv[2]
    ofile = open(filename, 'r')
    chunks = {}
    chunks["None"] = []
    currentLabel = "None"
    orderLabels = ["None"]
    for line in ofile:
        if ':' in line:
            label = line.split(':')[0]
            chunks[label] = []
            currentLabel = label
            orderLabels.append(label)

                                  ne"

                                  l].append(line)
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
    a,b = 0,0
    a,b = process(0, 0, chunks, "None", 1)
    for label in orderLabels:
        if label is not "None":
            a,b = process(a, b, chunks, label, depth)
    ofile.close()

if __name__ == '__main__':
    main()
```

permalink **embed**

[–] Rasaford 1 point 8 months ago

Java Had some seriously hard to find bugs on this one. Nevertheless here it is:

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Stack;

public class LEDs {

    private Map<String, byte[]> register = new HashMap<>();
    private Stack<Integer> label = new Stack<>();

    public void parse(String in)
    {
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
rrayList<>(Arrays.asList(in.split("(\\s{2,})|\\n")));

s.singleton(""));
size(); i++)
```

```
            String[] command = cmd.get(i).split("(.\\(+)|(\\).)|(\\s+)|(,)");
            String arg = command[0];
            if (command.length > 1 && !register.containsKey(command[1]))
                register.put(command[1], new byte[8]);
            switch (arg)
            {
            case "ld":
                setLED(command[1], command[2]);
                break;
            case "out":
                print(command[2], Integer.parseInt(command[1]));
                break;
            case "rrca":
                rrca("a");
                break;
            case "rlca":
                rlca("a");
                break;
            case "djnz":
                if (djnz("b"))
                    i = label.pop();
                break;
            default:
                if (arg.endsWith(":"))
                    label.push(i-1);
                break;
            }
        }
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
int start)

er.get(key);
```

```
        for (int i = 0; i < brightness.length - start; i++)
        {
            if (brightness[i] > 0)
                System.out.print("*");
            else
                System.out.print(".");
        }
        System.out.println();
    }

    private void setLED(String key, String num)
    {
        byte[] brightness = new byte[8];
        String bin = Integer.toBinaryString(Integer.parseInt(num));
        int i = brightness.length - 1;
        int j = bin.length() - 1;
        while (i >= 0 && j >= 0)
        {
            brightness[i] = (byte) Integer.parseInt(bin.substring(j, j + 1));
            i--;
            j--;
        }
        register.replace(key, brightness);
    }

    // right shift
    private void rrca(String... keys)
    {



                                  t(key);
                                  3];
                                  );
            for (int i = 1; i < out.length; i++)
            {
                out[i] = a[i - 1];
            }
            out[0] = a[7];
            register.replace(key, out);
        }
    }

    // left shift
    private void rlca(String... keys)
    {
        for (String key : keys)
        {
            byte[] a = register.get(key);
            byte[] out = new byte[8];
            // register.remove(key);
            for (int i = 0; i < out.length - 1; i++)
            {
                out[i] = a[i + 1];
            }
            out[7] = a[0];
            register.replace(key, out);
        }

    }

    private boolean djnz(String... keys)
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
.get(key);
oyOf(leds, leds.length);
```

```
            int sum = 0;
            for (int i = leds.length - 1; i >= 0; i--)
            {
                if (leds[i] == 1)
                {
                    out[i] = (byte) (1 - leds[i]);
                    break;
                } else
                {
                    out[i] = (byte) (1 - leds[i]);
                }
            }
            for (byte i : out)
            {
                sum += i;
            }
            if (sum == 0)
                return false;
            register.replace(key, out);
        }
        return true;
    }
}
```

permalink **embed**

[–] **lisbonant** 1 point 8 months ago

Go

Feedback always welcome. This was trickier than expected, and I'm sure it's more fragile than I'd like - but all challenge cases pass as

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
    "bufio"
    "fmt"
    "log"
    "os"
    "strconv"
    "strings"
)

const inFile = "infile.dat"

var (
    regA, regB int
    labels     map[string]int
)

func displayLights(a int) {
    bits := getBits(a)

    for _, v := range bits {
        if v == "1" {
            fmt.Print("*")
        } else {
            fmt.Print(".")
        }
    }
    fmt.Print("\n")
}
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

l

```
        tokens := strings.Split(v, " ")

        if tokens[0] == "out" {
            //display output
            displayLights(regA)
        } else if tokens[0] == "ld" {
            //update given register
            activeReg := strings.Split(tokens[1], ",")[0]
            update, e := strconv.Atoi(strings.Split(tokens[1], ",")[1])
            if e != nil {
                log.Fatalln(e)
            }
            if activeReg == "a" {
                if update < 0 || update > 255 {
                    log.Fatalln("a must satisfy 0 <= a < 256")
                }
                regA = int(update)
            } else if activeReg == "b" {
                regB = int(update)
            } else {
                log.Fatalln("No such register")
            }
        } else if tokens[0] == "rlca" {
            regA = rotateC(regA, "left")
        } else if tokens[0] == "rrca" {
            regA = rotateC(regA, "right")
        } else if len(tokens) == 0 {
            continue
                                         nz" {


                                         label
                                         tokens[1]])
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
                }
            } else {
                //any other word is assumed to be a label, store line number of label
                //don't store the colon
                split := strings.Split(tokens[0], ":")
                //get line number
                var line int
                for k, v := range i {
                    if v == tokens[0] {
                        line = k
                    }
                }
                labels[split[0]] = line
            }
        }
    }

    func getBits(a int) []string {
        bin := strconv.FormatUint(uint64(a), 2)
        bin = leftPad2Len(bin, "0", 8)
        tmp := strings.Split(bin, "")
        var bits []string
        for _, v := range tmp {
            bits = append(bits, v)
        }
        return bits
    }
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
string, overallLen int) string {

 - len(padStr)) / len(padStr))
dStr, padCountInt) + s
verallLen):]
```

```
}

//rotateC rotates register r in direction d, either "right" or "left"
func rotateC(r int, d string) int {
    bits := getBits(r)
    if d == "left" {
        t := bits[0]
        for i := 1; i < len(bits); i++ {
            bits[i-1] = bits[i]
        }
        bits[len(bits)-1] = t
    } else if d == "right" {
        t := bits[len(bits)-1]
        for i := len(bits) - 2; i >= 0; i-- {
            bits[i+1] = bits[i]
        }
        bits[0] = t
    }
    temp := strings.Join(bits, "")
    ret, e := strconv.ParseInt(temp, 2, 0)
    if e != nil {
        log.Fatalln(e)
    }
    return int(ret)
}

func main() {
    instructions := []string{}
    labels = make(map[string]int)

    scanner := bufio.NewScanner(f)
    for scanner.Scan() {
        instructions = append(instructions, scanner.Text())
    }

    execute(instructions, 0)

}
```

permalink **embed**

[–] **JusticeMitchTheJust** 1 point 7 months ago

Very late to the party, but here's a **Java 8** solution

+/u/CompileBot java

```
import java.io.*;
import java.util.*;
import java.util.function.*;
import java.util.regex.*;
import java.util.stream.*;

class Leds{

    public enum ACTION {
        LD(Pattern.compile("^\\s*ld ([a-z]),(\\d*)"),
                input -> REG.put(input.get(0), Byte.valueOf(input.get(1)))),
                                    s*out \\(0\\),([a-z])"),

                                    et(input.get(0));
                                    ;
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
        RLCA(Pattern.compile("^\\s*rlca"),
                input -> REG.put("a", toByte(Integer.rotateLeft(REG.get("a"), 1)))),
        RRCA(Pattern.compile("^\\s*rrca"),
                input -> {
                    Byte a = (byte) (REG.get("a"));
                    byte rolled = (byte) ((((a & 0xff) >> 1) + ((a & 0x01) == 1 ? 1 << 7 : 0)));
                    REG.put("a", rolled);
                }),
        LABEL(Pattern.compile("^\\s*(.*):"),
                input -> { }),
        DJNZ(Pattern.compile("^\\s*djnz (.*)"),
                input -> {
                    Byte b = REG.get("b");
                    REG.put("b", (byte) (b - 1));
                    if (b > 1) {
                        IntStream.range(0, PROGRAM.size())
                                .filter(index -> (PROGRAM.get(index).action == LABEL && PROGRAM.get(index)
                                .findFirst()
                                .ifPresent(index -> PC = index);
                    }
                });

        public Pattern pattern;
        public Consumer<List<String>> consumer;

        private ACTION(Pattern pattern, Consumer<List<String>> consumer) {
            this.pattern = pattern;
            this.consumer = consumer;

                                      num) {
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
            return (byte) ((tmp & 0x80) == 0 ? tmp : tmp - 256);
        }

        public static final HashMap<String, Byte> REG = new HashMap<>();
        public static Byte LEDS = 0x00;
        public static final List<ParsedAction> PROGRAM = new LinkedList<>();
        public static int PC = 0;

        public static Optional<ParsedAction> parseAction(String line) {
            return Arrays.stream(ACTION.values())
                    .filter(action -> action.pattern.matcher(line).matches())
                    .map(action -> {
                        Matcher m = action.pattern.matcher(line);
                        List<String> params = new ArrayList<>();
                        m.find();

                        return new ParsedAction(action, IntStream.range(1, m.groupCount()+1)
                                .mapToObj(index -> m.group(index))
                                .collect(Collectors.toList())
                        );
                    })
                    .findFirst();
        }

        public static void outputLEDs() {
            for (int i = 7; i >= 0; i--) {
                System.out.print((LEDS >> i & 0x01) == 1 ? "*" : ".");
            }
```

< > discussions in r/dailyprogrammer X

43 · 24 comments

[17-08-21] Challenge #328 [Easy] Latin Squares

```
[] args) {
        new BufferedReader(new InputStreamReader(System.in)).lines()
                .sequential()
                .map(Leds::parseAction)
                .filter(Optional::isPresent)
                .map(Optional::get)
                .forEachOrdered(action -> PROGRAM.add(action));

        for (PC = 0; PC < PROGRAM.size(); PC++) {
            ParsedAction action = PROGRAM.get(PC);
            action.perform();
        }
    }

    public static class ParsedAction {

        public ACTION action;
        public List<String> params;

        public ParsedAction(ACTION action, List<String> params) {
            this.action = action;
            this.params = params;
        }

        public void perform() {
            action.consumer.accept(params);
        }
    }
}
```

```
  out (0),a
  ld a,8
  out (0),a

  out (0),a
  ld a,12
  out (0),a
  ld a,14
  out (0),a

    ld b,3

triple:
  ld a,126
  out (0),a
  ld a,60
  out (0),a
  ld a,24
  out (0),a
  djnz triple

  ld a,1
  ld b,9

loop:
  out (0),a
  rlca
  djnz loop
```

```
    out (0),a
    rrca
    djnz loop2
```

permalink **embed**

[–] **CompileBot** 1 point 7 months ago

Output:

```
....***.
....**..
....*...
....*...
....**..
....***.
.******.
..****..
...**...
.******.
..****..
...**...
.******.
..****..
...**...
.......*
......*.
.....*..
....*...
...*....
  *
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
*.......
.*......
..*.....
...*....
....*...
.....*..
......*.
```

source | info | git | report

permalink **embed** **parent**

[–] **dpforyou** 1 point 7 months ago

C# with TDD and like, functional tables or something:

```
public class LEDSimulator
{
    private class Instruction
    {
        public Instruction(string command, Func<string, int, string, int, int> action)
        {
            Command = command;
            Action = action;
        }

        public string Command { get; set; }
        public Func<string, int, string, int, int> Action { get; set; }
    }

    public string Run(string input)
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
];

ary<string, int>();
```

```
        var output = new StringBuilder();

        var instructions = new[]
        {
            new Instruction(
                "ld a,",
                (line,i,instruction,instructionPos) =>
                {
                    var num = Convert.ToInt16(line.Substring(instructionPos + instruction.Length));
                    var start = Convert.ToString(num, 2).ToCharArray().Select(x => x == '1').ToArray();
                    numA = new bool[8 - start.Length].Concat(start).ToArray();
                    return -1;
                }
            ),
            new Instruction(
                "ld b,",
                (line,i,instruction,instructionPos) =>
                {
                    numB = Convert.ToInt16(line.Substring(instructionPos + instruction.Length));
                    return -1;
                }
            ),
            new Instruction(
                ":",
                (line,i,instruction,instructionPos) =>
                {
                    var colonMatch = Regex.Match(line, @"\s+(\w+)\:");
                    if(colonMatch.Groups.Count > 1)

                                    = colonMatch.Groups[1].Value;
                                    (label, i);
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
                    }
                ),
                new Instruction(
                    "djnz ",
                    (line,i,instruction,instructionPos) =>
                    {
                        int jumpTarget = -1;
                        if (labels.TryGetValue(line.Substring(instructionPos + instruction.Length), out jumpTa
                        {
                            if (numB > 1)
                            {
                                numB--;
                                return jumpTarget;
                            }
                        }
                        return -1;
                    }
                ),
                new Instruction(
                    "out (0),a",
                    (line,i,instruction,instructionPos) =>
                    {
                        output.Append((output.Length > 0 ? Environment.NewLine : "") + String.Join("", numA.Se
                        return -1;
                    }
                ),
                new Instruction(
                    "rlca",
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
n,instructionPos) =>

mA[0];
ip(1).Concat(new[] { first }).ToArray();
```

```
                return -1;
            }
        ),
        new Instruction(
            "rrca",
            (line,i,instruction,instructionPos) =>
            {
                var last = numA[numA.Length-1];
                numA = new[] { last }.Concat(numA.Take(numA.Length-1)).ToArray();

                return -1;
            }
        ),
    };

    var lines = Regex.Split(input, Environment.NewLine);

    RunInstructionsOnLines(lines, instructions);

    return output.ToString();
}

private static void RunInstructionsOnLines(string[] lines, Instruction[] instructions)
{
    for (int i = 0; i < lines.Length; i++)
    {
        var line = lines[i];

                              on in instructions)

                               = line.IndexOf(instruction.Command);
                               > -1)
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
                        var newI = instruction.Action(line, i, instruction.Command, instructionPos);
                        if (newI > -1)
                            i = newI;
                        break;
                    }
                }
            }
        }
    }

    [TestClass]
    public class Challenge290
    {
        private static readonly string nl = Environment.NewLine;

        [TestMethod]
        public void Can_Do_8()
        {
            Assert.AreEqual("....*...", new LEDSimulator().Run("ld a,8" + nl + "out (0),a"));
        }

        [TestMethod]
        public void Can_Do_14()
        {
            Assert.AreEqual("....***.", new LEDSimulator().Run("ld a,14" + nl + "out (0),a"));
        }

        [TestMethod]
```

```
            ld a,12
            out (0),a
            ld a,8
            out (0),a

            out (0),a
            ld a,12
            out (0),a
            ld a,14
            out (0),a";

        var expected = Expect(@"
            ....***.
            ....**..
            ....*...
            ....*...
            ....**..
            ....***.");

        Assert.AreEqual(expected, new LEDSimulator().Run(instructions));
    }

    [TestMethod]
    public void Can_Do_Input_2()
    {
        var instructions = @"
            ld b,3
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
                ld a,24
                out (0),a
                djnz triple";

        var expected = Expect(@"
            .******.
            ..****..
            ...**...
            .******.
            ..****..
            ...**...
            .******.
            ..****..
            ...**...");

        Assert.AreEqual(expected, new LEDSimulator().Run(instructions));
    }

    [TestMethod]
    public void Can_Do_Input_3()
    {
        var instructions = @"
          ld a,1
          ld b,9

        loop:
          out (0),a
          rlca
```

```
                .....*..
                ....*...
                ...*....
                ..*.....
                .*......
                *.......
                .......*");

            Assert.AreEqual(expected, new LEDSimulator().Run(instructions));
        }

        [TestMethod]
        public void Can_Do_Input_4()
        {
            var instructions = @"
                ld a,2
                ld b,9

            loop:
                out (0),a
                rrca
                djnz loop";

            var expected = Expect(@"
            ......*.
            .......*
            *.......
            .*......
```

< > discussions in r/dailyprogrammer X

43 · 24 comments
[17-08-21] Challenge #328 [Easy] Latin Squares

```
            Assert.AreEqual(expected, new LEDSimulator().Run(instructions));
        }

        private object Expect(string input)
        {
            return String.Join(Environment.NewLine, Regex.Split(input, Environment.NewLine).Select(x => x.Trir
        }
    }
```

permalink **embed**