System software

SicTools

SicTools

- SicTools is
  - developer toolchain for educational computer SIC/XE
  - simulator and virtual machine
  - assembler, linker, loader, etc.

http://jurem.github.io/SicTools/

SicTools

- SicDemos is
  - accompanying project of SicTools
  - containes examples of programs for SIC/XE

https://github.com/jurem/SicDemos

Simulator

- Installation and basic instructions
  - go to the http://jurem.github.io/SicTools/ webpage
  - read and follow the instructions

- Result of installation
  - Java program: `sictools.jar`
  - includes
    - simulator `sic.Sim`
    - assembler `sic.Asm`

Simulator

- Use
  - `java -jar sictools.jar`
  - `java -jar sictools.jar examples/balls.asm`
  - `java -jar sictools.jar examples/balls.obj`
- Options
  - loading and observing asm/obj code
  - execution: start, step, stop
  - views: cpu, mem, screen, …

Simulator

- SIC/XE virtual machine
  - registers
  - memory
  - stream I/O devices
  - memmory-mapped I/O devices

Simulator

- Stream input / output devices
  - devices numbered from 0 to 255
    - instructions `RD`, `WD` in `TD`
  - devices 0, 1, and 2 are mapped to
    - 0 standard input
    - 1 standard output
    - 2 standard output for errors
  - all other numbers are mapped to
    - files (in the current working directory) with a name *NUM*`.dev`
    - writing to device AA writes (and creates) file `AA.dev`.

Simulator

- Textual screen
  - monochromatic black&white
  - textual: can only show characters (ASCII)
  - screen origin: 0xB800
  - default size: 80 x 25
  - representation: *row-major*
    - characters (cells) are stored in sequence
    - address of the cell (x, y) is

      address = screen_origin + y * cols + x

Simulator

- Color graphic screen
  - origin address: 0xA000
  - default size: 64 x 64
  - representation: *row-major*
    - pixels are stored sequentially
    - the address of a pixel on coordinate (x, y) is

      $$\text{address} = \text{screen\_origin} + y * \text{cols} + x$$

    - each pixel is represented with one byte: IRGB (iirrggbb)
      - intensity 0, 1, 2, 3 = 20, 40, 60, 80
      - color: (R, G, B) * amp

Simulator

- Keyboard
  - stores character code of the last keypress
  - the memory location is 0xC000
  - keyboard window must have focus

Simulator

- Settings
  - frequency
    - default: 100 Hz
      - textual / graphical programs needs higer frequencies
    - max: 100,000,000 Hz = 100 MHz

Assembler

- Usage
  - via simulator
    - „Load asm“ - turbo principle
  - via standardn input
    - `java -cp sictoolss.jar sic.Asm <<< "start LDA 42"`
  - via file (as argument)
    - `java -cp sictools.jar sic.Asm SicDemos/balls.asm`

Assembler

- Switches
  - help: -help, -h
  - reference:
    - `java -cp sictools.jar sic.Asm -refshort`
    - `java -cp sictools.jar sic.Asm -reflong`
  - object file format
    - `java -cp sictools.jar sic.Asm -obj-slack` *code.asm*

Assembler

- Via file as argument
  - generates several files
  - *name*`.obj` – object (machine) code
  - *name*`.lst` – anotated source code (listing)
  - *name*`.log` – log file

Assembler

- Object file
  - stores machine code
  - SIC/XE obj format
    - hex encoding
    - records H, T, E, ...

Assembler

- Listing file
  - containes source and object code
  - line format
    - address + object code + instructions + operands

Assembler

- Log file
  - for each section
    - list of symbols
    - list of literals
    - list of relocations