# The Q1 Emulator Project

- Brief intro to Q1
  - Company
  - About Q1
- The emulator project
  - How to (possibly) write an emulator
  - Emulator architecture
- Emulator Demo

Dansk
Datahistorisk
Forening

# Acknowledgements & Data sources


Peter Andersen – rom images, documentation

Achim Baqué – original floppy disks

Alex Burke – hw investigations, curation

Poul-Henning Kamp – reading the floppies

Ivan Kosarev – z80/8080 emulator

Mattis Lind – floppy images, chargen ROMs

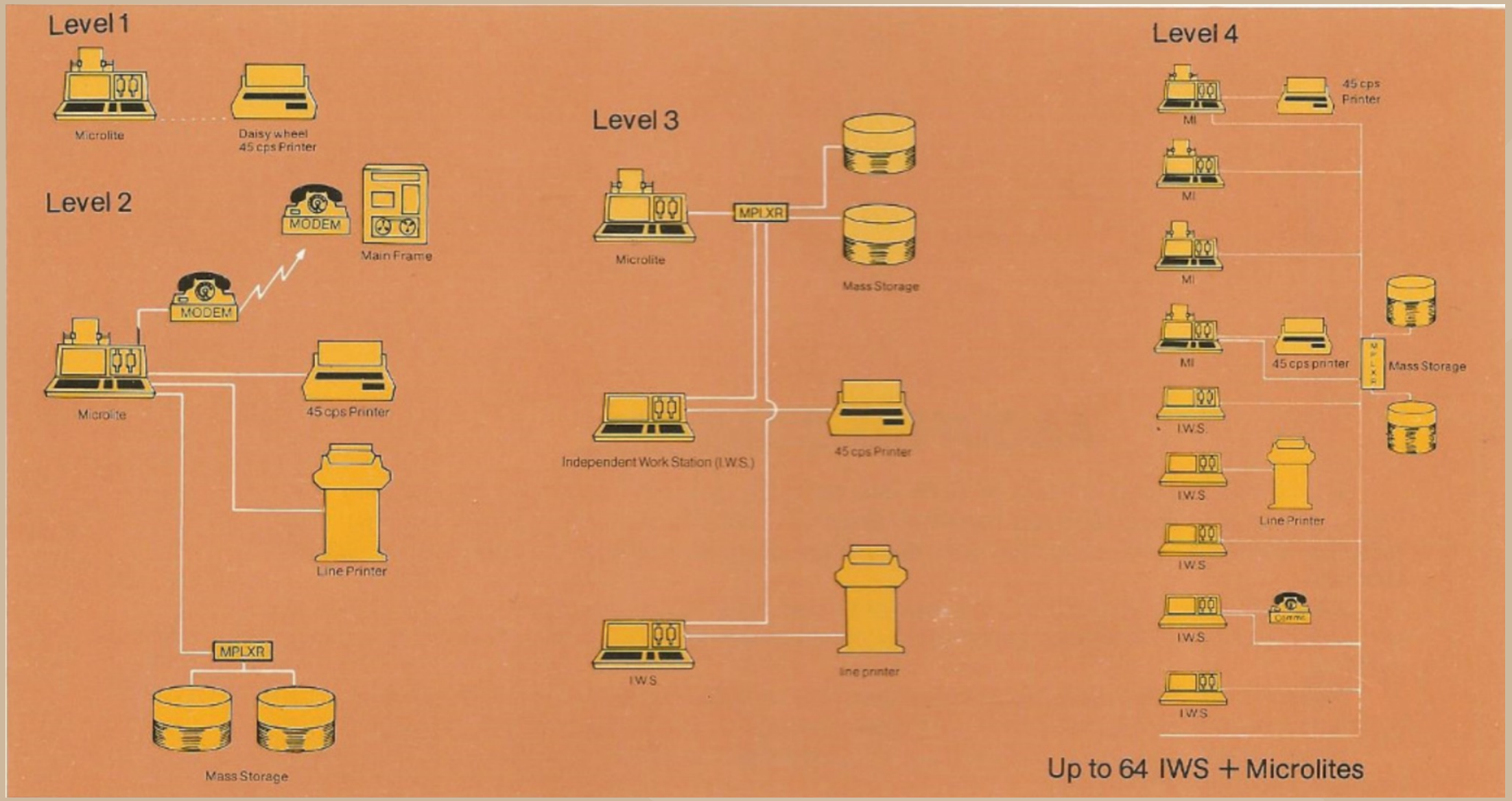
Karl-Wilhelm Wacker – Q1 knowledge

*"Being conscious is the central fact of human experience. Yet, it is not presently known what consciousness is and what it does. For example, Physicalism, the currently dominant theory of knowledge takes the position that the non-conscious brain can do anything that the conscious brain can do"*

-Daniel Alroy

# About Q1

- Q1 Corporation
  - Founded by Daniel Alroy
  - 1972 to ~1990
  - Self promoted as world's first
- Used as business machine
  - Computer terminal
  - Data entry
  - Word processing
  - Payroll/accouting


Level 1
Microlite
Daisy wheel
45 cps Printer
Level 2
MODEM
Main Frame
MODEM
Microlite
45 cps Printer
Line Printer
MPLXR
Mass Storage
Level 3
MPLXR
Microlite
Mass Storage
Independent Work Station (I.W.S.)
45 cps Printer
IWS
line printer
Level 4
MI
45 cps
Printer
MI
MI
MI
MI
45 cps printer
Mass Storage
I.W.S.
I.W.S
Line Printer
I.W.S
I.W.S
I.W.S
Up to 64 IWS + Microlites

- Known customers
  - NASA – work order management
  - Aroskraft – swedish powerplant
- No operating system
  - line editor
  - entered text interpreted as program name to execute
  - Very limited error handling
- Programs mainly written in PL/I
  - Compiled into pseudo-machine code
  - assembler programs also supported

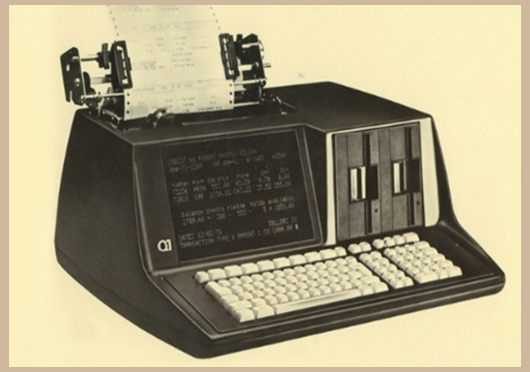
Q1/LMC (1972) 8008
1 x 80 characters
integrated printer

Q1/Basic Office Machine
12 x 40 characters
integrated floppies

Q1/Lite (1974) Z80
12 x 80 characters
integrated floppies

... I have also seen display widths of 47 chars ...

... and Q1/T and Q1/C product names ...



Q1/Lite (1974) 8080
12 x 80 characters
integrated printer + floppies


Q1/MicroLite (197x)
Z80
24 x 80 characters
integrated floppies + harddrives

... presumably a fourth generation existed using the 68000 processor ...

electronics today 1980

## Micro Age 78

Sheridan College is hosting Micro Age 78, to be held June 1 and 2 at the Oakville Campus.

More than 30 exhibitors will display their latest microcomputer equipment (some not even on the market yet) and demonstrate uses for such fields as small business management, word processing, environmental management, education and process control.

The exhibition/conference, will run from 10 a.m. to 10 p.m. each day. Among the speakers is Daniel Alroy, President of Q1 Corporation, which manufactured the world's first microcomputer system in 1972 and was the first manufacturer to deliver magnetic bubble memory (1977).

Admission
person (fre
College is lo
Oakville, On
QEW. For fu
the Oakville
823-9730, ex

from sales brochure 1980

systems.

The aim of Q1 is to reduce the complexities of the computer world to a scale totally manageable by all forms of commercial enterprise – no matter how large or small; whatever their line of business.

Many "firsts" have been achieved by Q1 Corporation including designing the world's first microcomputer system in 1972.

Today installations have been successfully accomplished in 20 countries stretching across the globe.

datamation 1979

**the technological leader in microcomputer systems, is looking for a few outstanding people.**

Since 1972, when we developed and delivered the world's first microcomputer system, we have maintained the technological leadership in this rapidly growing industry. Our company is now planning a joint venture

**Q1 Corporation**

## Leader in Microcomputer Systems

- In 1972, Q1 Corporation developed, manufactured and delivered the world's first microcomputer system to Litcom, a division of Litton Industries.
- Q1 Corporation was the first to introduce microcomputer systems with flexible diskette drives for external storage, which are now becoming the industry standard.
- Q1 was the first to make available on a low-cost microcomputer system the powerful PL/1 programming language used on the far more expensive IBM 370 systems.

Dansk
Datahistorisk
Forening

# Emulator Challenge

How to write an emulator

The Q1 emulator architecture

# SYSTEMS & PERIPHERALS

## Bits & Pieces

### Input to Key-to-Tape Unit Done by Selectric

PLAINVIEW, N.Y. – A key-to-cassette system provides full teletypewriter signal capability and outputs keyboard data from an IBM Selectric directly onto a magnetic tape cassette, according to the developer, Varisystems Corp.

Designed as an alternative to punched paper tape, the "Reporter" eliminates the need for keypunching, tape tearing, chadding, spooling and other activities normally associated with paper tape, the firm's spokesman said.

The tape drive controls a 200-foot magnetic tape cassette which can store up to 645K bits of data each, in blocked format up to 256 characters. Tape speed averages 31 in./sec, the spokesman said.

Special command keys on the Selectric enable the operator to output codes used on- or off-line to drive peripherals, central processors or phototypesetting equipment, the spokesman added.

Cost of the system is $3,470 with delivery in 60 days from 207 Newtown Road, 11803.

## Minicomputer Allows PL/I Programming

NEW YORK – Q1 Corp. has a desk-top minicomputer costing around $20,000 which features a PL/I language capability.

The Q1/T is available in 4K-, 8K-, 12K- and 16K-byte configurations with the PL/I compiler requiring about 8K bytes – of which 5K are resident, a firm spokesman said.

The other language offered, a system assembler, can be linked to PL/I programs so assembler code can be used as subroutines, he asserted.

The present storage medium is special magnetic cards which are inserted into the side of the unit. Each of these cards can hold 64 tracks – 160 char./track – of data for a total of 10K byte/card which can be accessed by programs, the spokesman added.

A disk subsystem for larger storage, the spokesman said, will soon be released.

Output from the system is via a 30 char./sec printer that incorporates proportional spacing and offers users upper- and lower-case options with a 158-character total capability, the spokesman said.

Since the printer also has forward and reverse capabilities, it can be used as a plotting unit, he added.

Included in the keyboard entry console is an electronic display with 80 character display positions. This display is used to visually edit and correct programs or data, the spokesman stated.

In many cases the Q1/T provides faster response than larger, batch-oriented machines, such as the IBM 360/30 and System/3, since card punching and sequential tape storage are eliminated, the spokesman asserted.

The Q1/T is specifically aimed at commercial data processing and text-processing users with applications such as order, entry, inventory, receivables, payables and general ledger, the spokesman said.

In programming a typical business application – such as payroll – the use of PL/I permits a reduction of up to 90% in programming cost, compared to using machine language, the spokesman claimed.

Q1 Corp. will provide users with application and operating software as well as provide personnel training programs and operating aids, for the Q1/T, the spokesman stated.

Future enhancements include a mag tape system so that Q1/T output can be transferred and used by larger processors, and a 64K byte version of the system.

## Future System Design Possible With Switch

By Michael Weinstein
Of the CW Staff

ZURICH, Switzerland – IBM scientists have unwrapped an experimental switching device that is reportedly much faster – by a factor of at least 20 – than

## Small Calculators Add Memory For Preset Programming Tasks

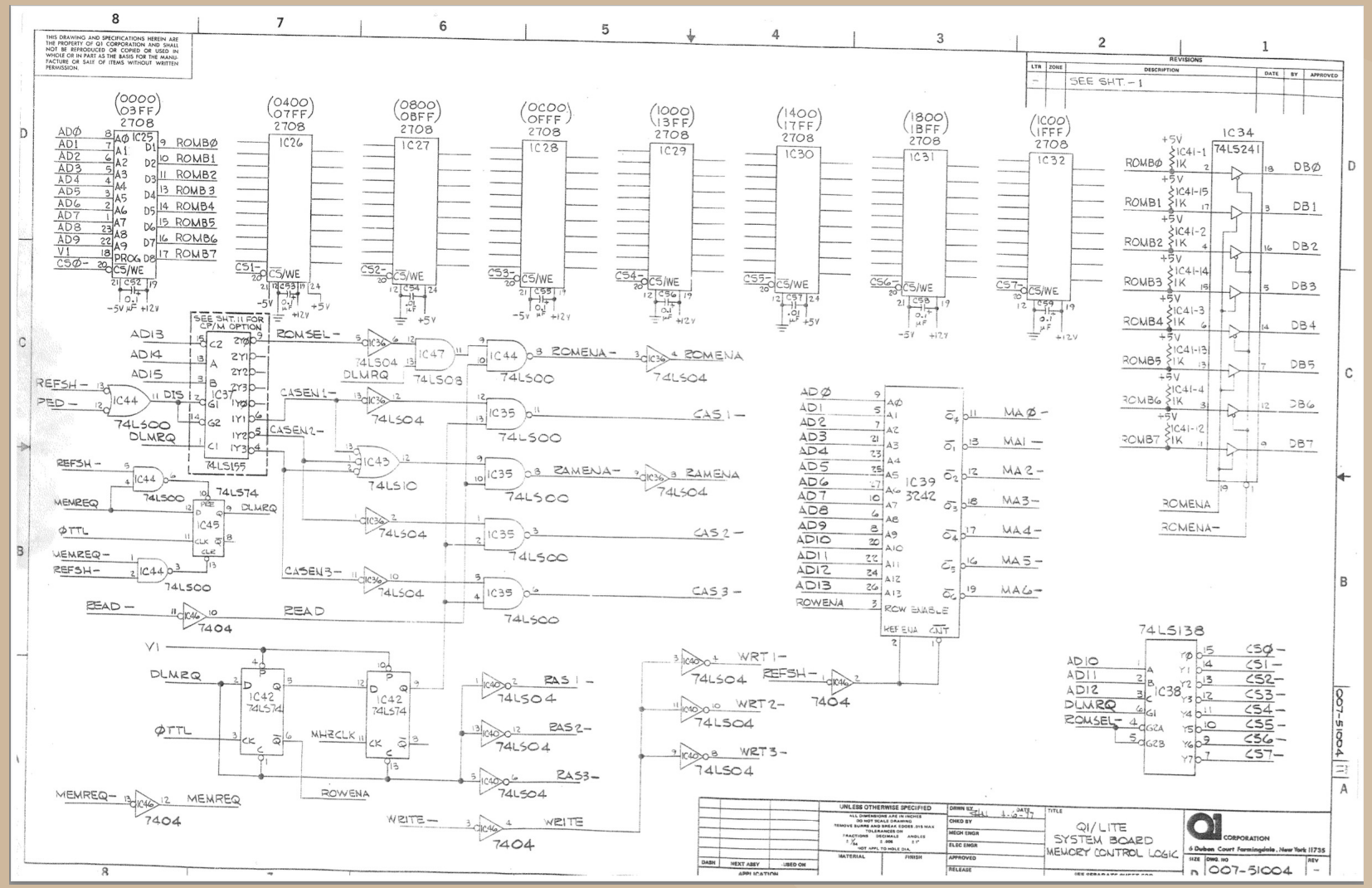
Burroughs and Hewlett-Packard have on a standard duplex calculator."

## Basic skeleton of an emulator

```
while (running) {
    disassembly(pc)
    in = cpu.getInstruction(pc)
    pc = cpu.handleInstruction(in)
    keyboardInput()
}
```

## Utilities

- loader
  - initialises RAM, loads ROM images, and code snippets, sets the initial program counter
- runner
  - steps through instructions one at a time
  - prints relevant information (pc, regs, disassembly, annotations)




Memory locations for the ROM files were identified from the Q1/Lite mainboard schematics

## Adding io

```
void handleOut(addr, data) {
   // printf("%d (%c)", addr, data);
   return;
}

void handleIn(addr, data) {
   if (addr == disk_status)
      return 0x42;
   ...
}
```

## Idea

Initially we ignore any output the program writes, or possibly just print the address and the value (both integer and ascii)

However on input we stop the emulator, inspect the disassembly and the returns what seems to lead us closer to the end goal

At this stage I found myself alternating between

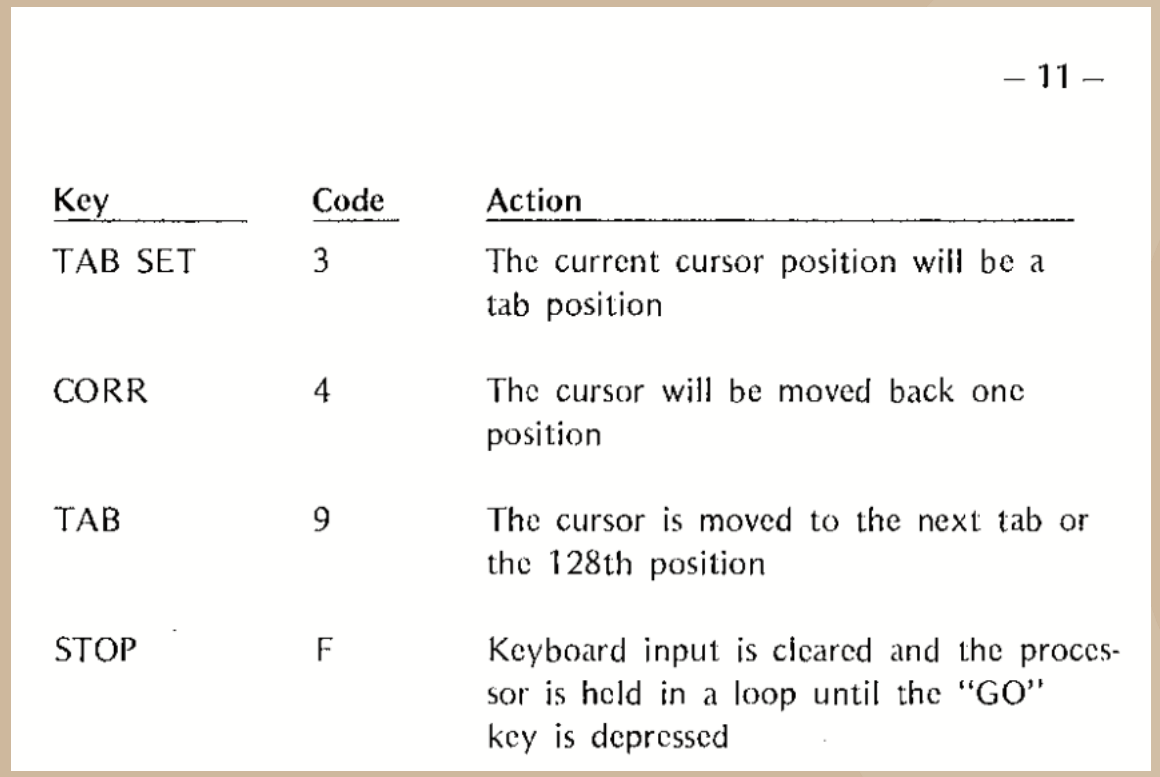
- inspecting the ROMs
- inspecting the output
- inspecting the disassembly
- adding annotations
- reading available manuals
- modifying IO handling

Using this approach relatively quickly took me to the point where the Q1 had initialised its RAM structures such as jump vectors, keyboard buffer and file descriptors.

And had sent the following text to the display

**Q1-Lite klar til brug**

Here it was obvious that keyboard input was needed.



– 11 –

| Key | Code | Action |
|---|---|---|
| TAB SET | 3 | The current cursor position will be a tab position |
| CORR | 4 | The cursor will be moved back one position |
| TAB | 9 | The cursor is moved to the next tab or the 128th position |
| STOP | F | Keyboard input is cleared and the processor is held in a loop until the "GO" key is depressed |

Sparse documentation obtained from

'Q1 ROS Users Manual'

Not all keycodes are documented (for example the GO key)

Function keys were eventually identified by trial and error

## Keyboard

```
def int38(self, key):
    io.keyin = key
    oldpc = pc
    sp -= 2
    mem.writeu16(sp, oldpc)
    pc = 0x38 # int vector


...
if ch == kc.ikey("TAB"):
    int38(kc.okey("TAB"))
```

## Idea

occasionally check for keyboard input. And when a key is pressed fake an interrupt.

Luckily Q1 uses ASCII

Next up is the display

Display

Address 03,04

The display is an output device. It is a fully buffered electronic unit, capable of displaying up to 12 lines of data under program control. Each line contains a minimum of 47 character positions. Each character code is seven bits long. When a data byte is written from the processor to the display, the high-order bit is ignored. After one line is filled, the next character will automatically appear in position 1 of the next line.

To write a character on the display, address the device and output the first character. Control instructions may be issued to reset the display to character position 1 of line 1, nondistructively blank or restore the display, or step the display to the next character position.

OUT,03
Writes a character on the display

IN,04
Status
Bit 6 = 0 for 12 line = 1 for 6 line
Bit 5 = 1 for LITE; = 0 for LMC

```
Bit 7 = 1    display 'busy'
Bit 4 = 1    40 character
Bit 3 = 1    80 character
```

OUT,04
Display Control
The control bit assignments for the A-register are:

7 6 5 4 3 2 1 0

Bit 0 - Reset: resets display to leftmost position of line 1

Bit 1 - Blank: display is blanked, but display buffer is not erased

Bit 2 - Unblank: buffer contents restored to display

Bit 3 - Step: character position is advanced one space to the right or to position 1 of the next line if prior position was at the end of a line

This is the entire documentation for the display!

from "Q1 ASM IO addresses usage"

**Display**

```
def handle_display_out(val):
    display.data(chr(val))
    display.update()

def update(self):
    msg = cursor_pos
    for l in buffer:
        msg += ''.join(l)
    udp.send(msg)
```

**Idea**

Display is a buffer of x *y bytes + current cursor position.

Upon transmit to display:

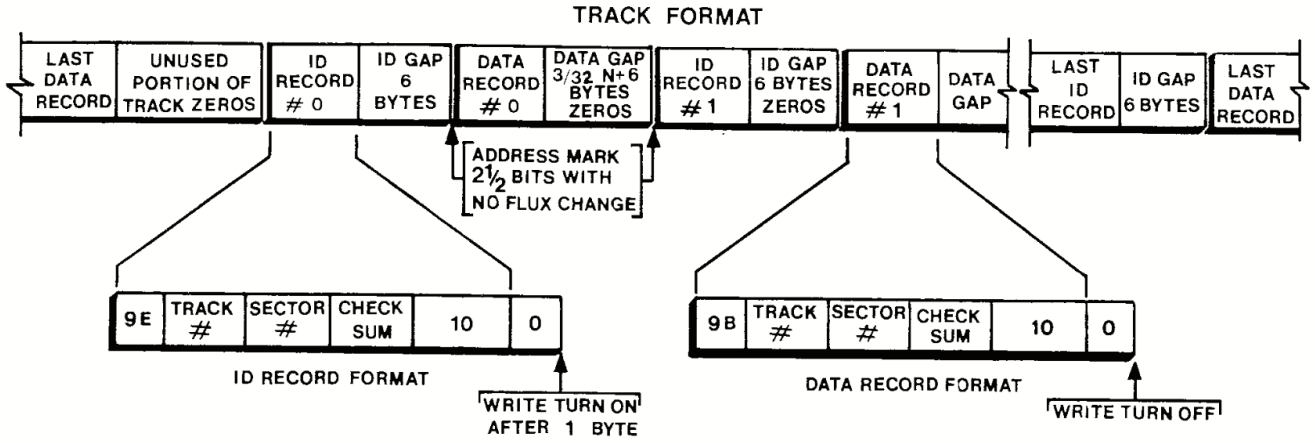
1) update the buffer
2) send buffer to display emulator

Once keyboard and display was 'working' it became clear that anything the user typed was interpreted as a program to be loaded from disk.

This required a floppy emulator. By far the most challenging aspect so far.

Luckily two-three disk images existed (DDF, Datormuseum)

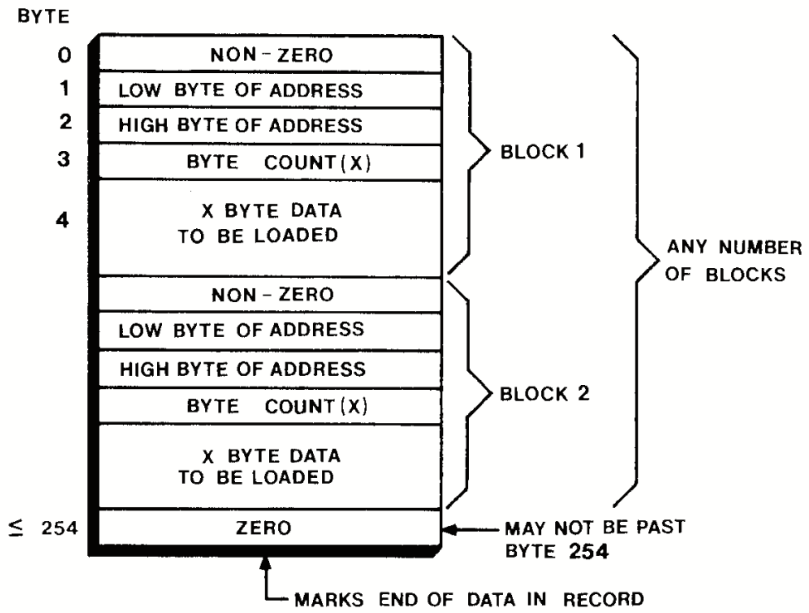
Unfortunately the documentation was sparse – and inaccurate




| BYTE | | |
|---|---|---|
| 0 | RECORD NUMBER | ZERO ON INDEX |
| 2<br>4<br>6<br>8 | NAME OF FILE<br>IN ASCII<br>PADDED WITH<br>BLANKS ON RIGHT | ONLY THIS<br>PORTION (FILE NAME)<br>NEEDS TO BE IN MEMORY<br>BEFORE CALL TO OPEN |
| A | NUMBER OF RECORDS | ENFILE DATA. EXPANDED TO MAXIMUM BY CALL TO WRITE |
| C | RECORD LENGTH | REFERED TO AS N ABOVE |
| E | RECORDS / TRACK; DISK # | DISK # IS ZERO ON INDEX |
| 10 | FIRST TRACK # | |
| 12 | LAST TRACK # | MOST SIGNIFICANT BIT IS SET IF PROTECTED |
| 14 | UNUSED | |
| 16 | RECORD # BEFORE LAST OPERATION | ZERO ON INDEX. USED FOR REWRITE AND RETRIES |

FILE DESCRIPTION
AS IT APPEARS IN MEMORY AND ON INDEX

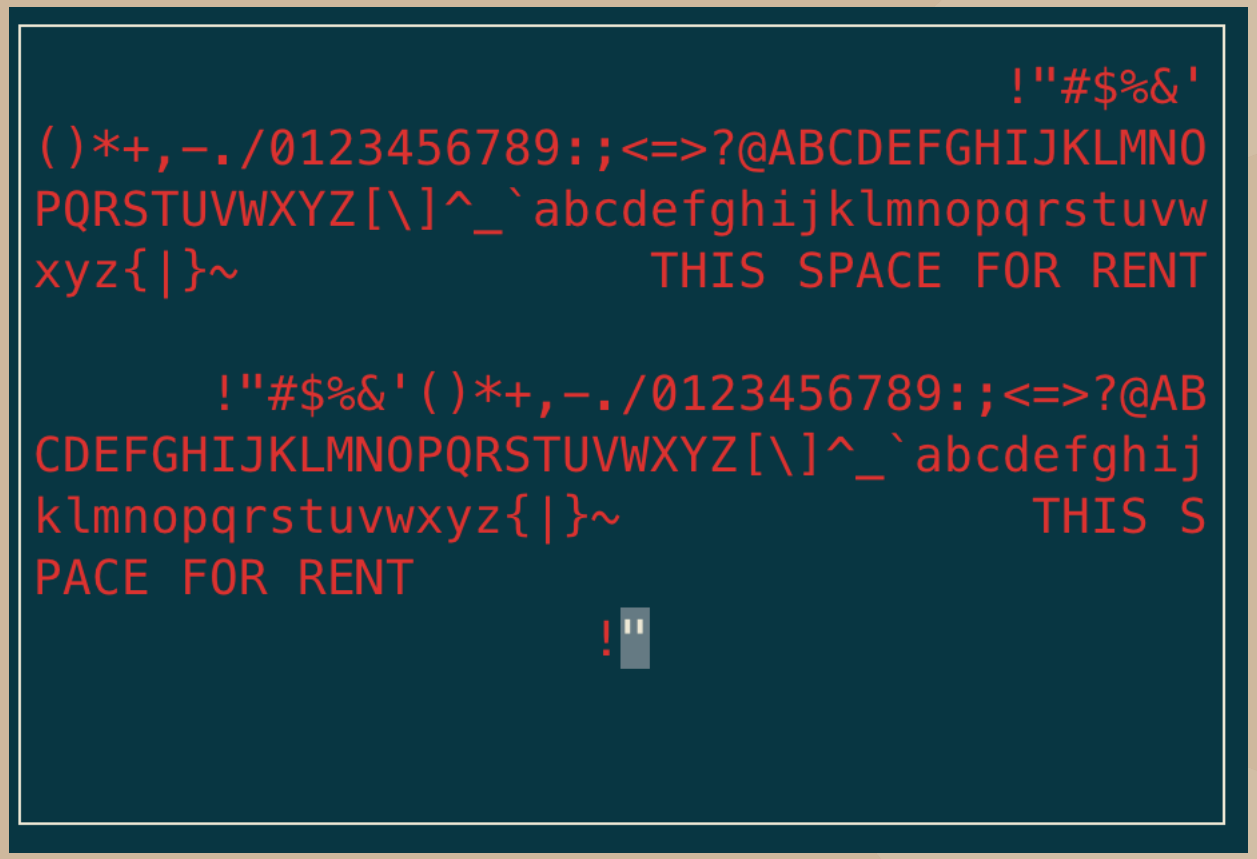
FIGURE 2




LOADER RECORD FORMAT

Entire documentation for the disk format (incorrect) and the program loader.

.... the long and windind road ....




Debug disk (felsökningsdiskett) decoded by Mattis Lind was very useful for bringup

**SCR** was the first program succesfully loaded and executed.

z80 assembler (luckily not PL/I pseudo-machine code)

Project now stalled due to lack of disks/programs

# Then on the same day (30 November) we increased the number of available floppy disks by about a factor 100!

**To:**'Morten Jagd Christensen' <mortenjc@jcaps.com>

Hi Morton,

Today I packed the disks and ship it on Monday. I am pleased to support your project.
I trust in you. It is risky to ship the disks. They could be lost or damaged.

Please take care of it! After copying the disks please send it back to my address

Cheers,
Achim

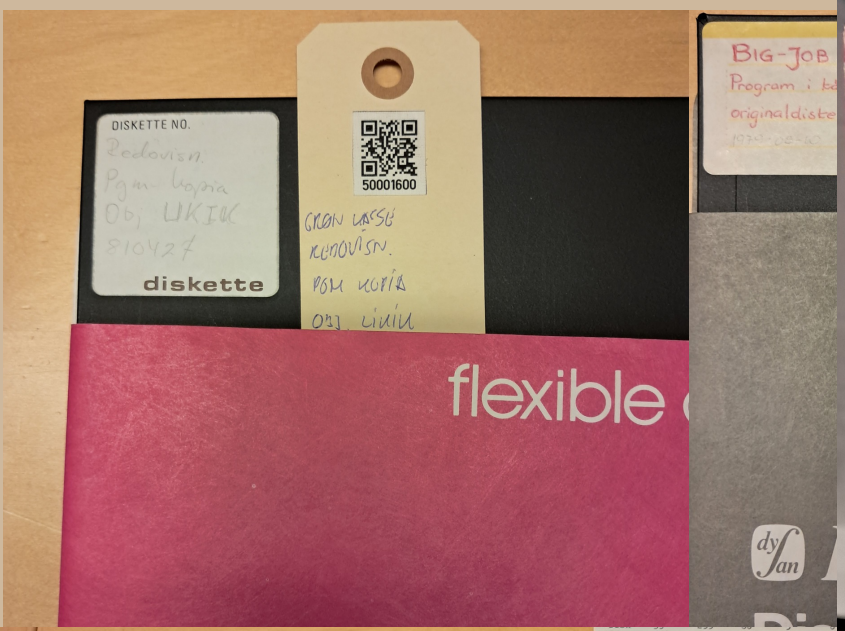
Best regards

Morten

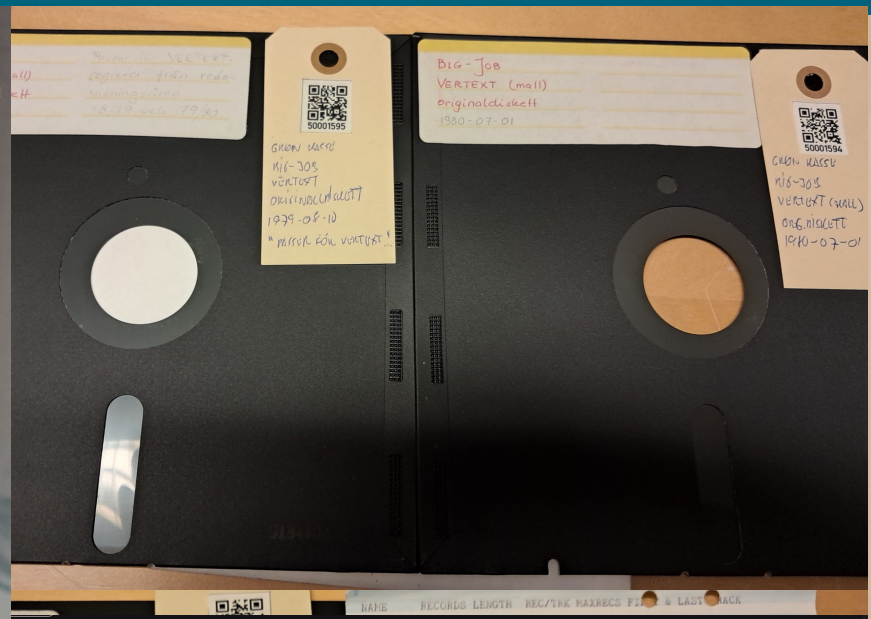
On 30/11/2024 16.25, Mattis Lind wrote:

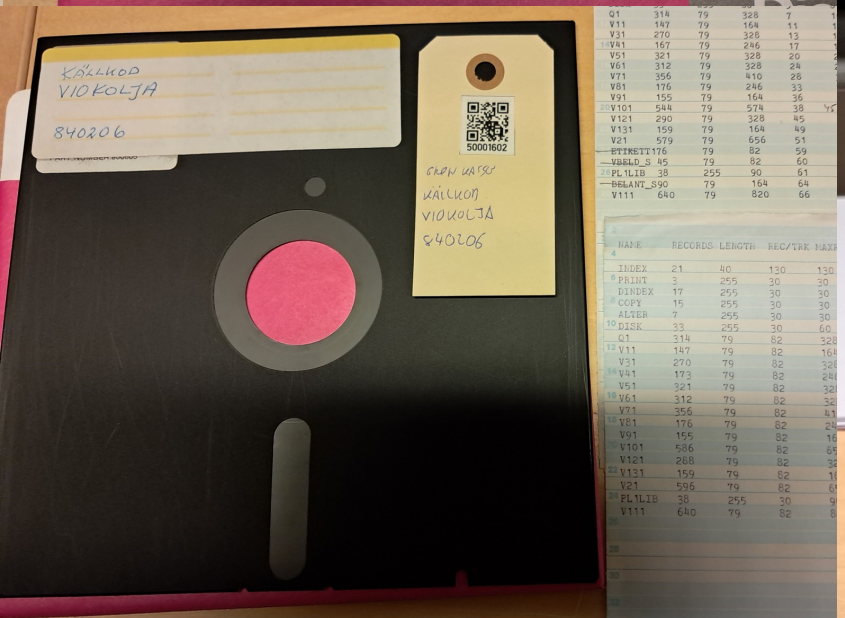
> Hello all!
>
> I have uploaded pictures of around 40 disks with content that appears to be interesting. In total there is between 100 and 200 disks.
>
> Q1 disks MSAB – Google Drive
> drive.google.com

# Dansk Datahistorisk Forening

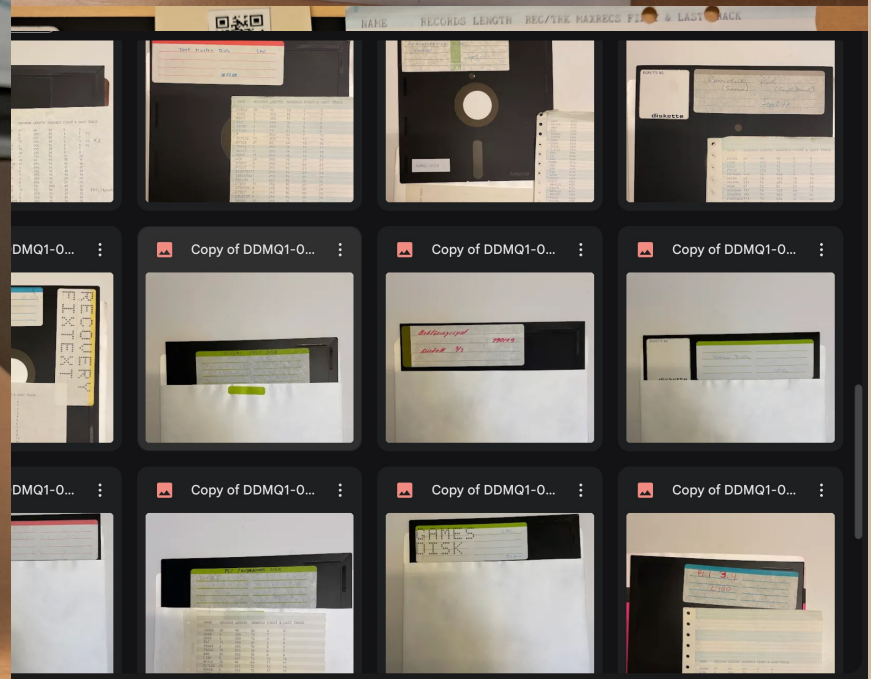




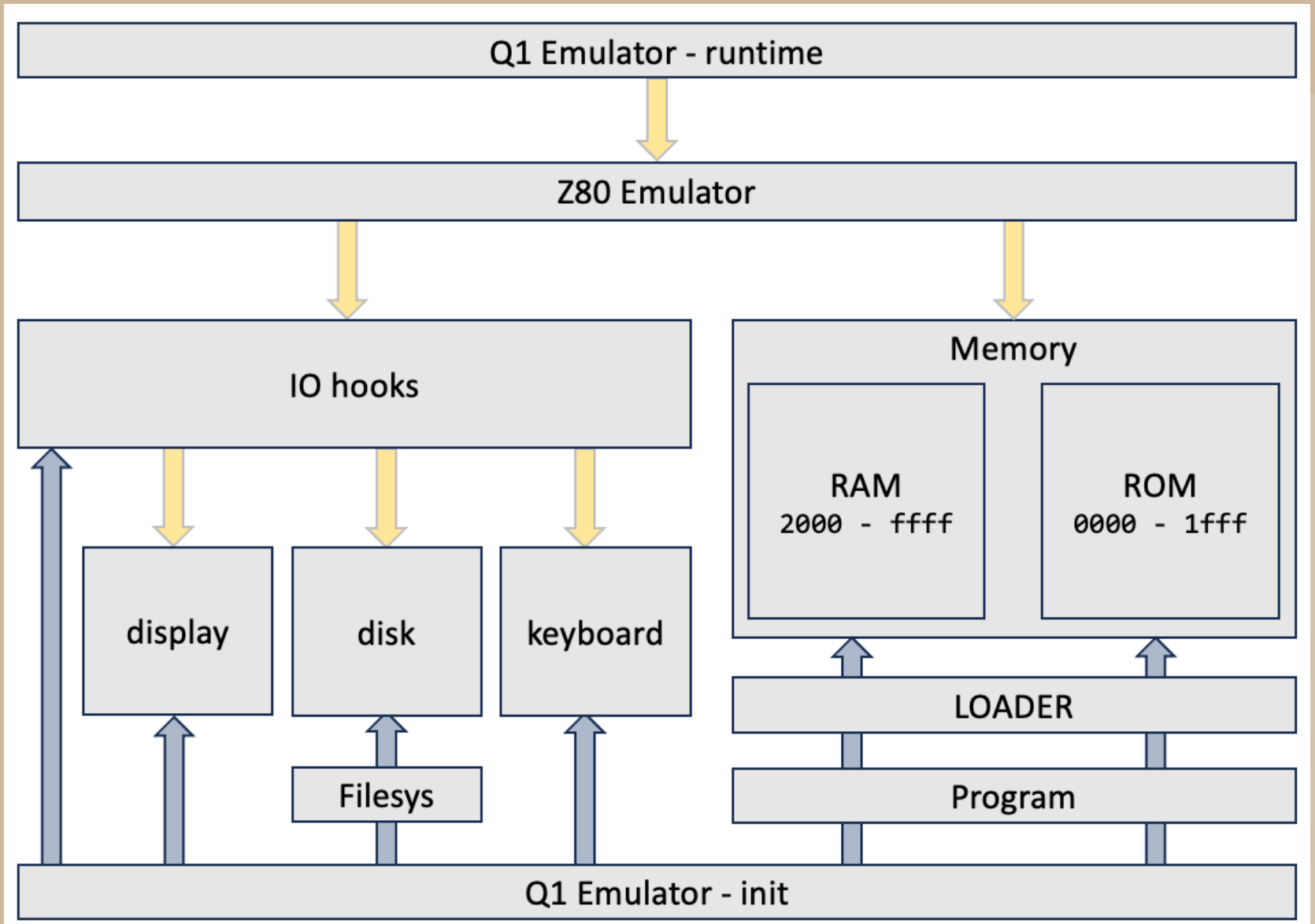




# Emulator Architecure




Using a z80 emulator by Ivan Kosarev
found a few bugs
helped improve python bindings

instead of

getinstruction()
handleinstruction()

I now use

step(1)



**CPU abstraction**

reset(), halt(), exit()

z80 emulator - step()

instruction decode

backtrace

**Memory abstraction**

clear()

load()

hexdump()

read8(), read16()

write8(), write16()

```
write to ROM (0x0a62), pc 6298 warning, no effect ...
...
write to ROM (0x098e), pc 2376 warning, no effect ...

0xfdfdfdfd at 2926, exiting ...

########### HEXDUMP 0x2000 - 0x10000 ###################################
4090 fd fd fd fd fd fd fd fd fd fd fd fd 00 fd fd fd  ................
....
fe50 20 4c 45 4e 47 54 48 20 28 6b 65 79 20 69 73 20   LENGTH (key is
fe60 70 61 72 74 20 6f 66 20 20 64 61 74 61 20 72 65  part of  data re
fe70 63 6f 72 64 29 3a 20 45 4e 54 45 52 20 4b 45 59  cord): ENTER KEY
....
########### HEXDUMP END #################################################

093e 21 9c 40     ; ld hl, 0x409c            | sp=0990, a=fb    bc=00e5, de=3642, hl=409c,
0937 34           ; inc (hl)                 | sp=0990, a=fb    bc=00e5, de=3642, hl=409c,
0941 35           ; dec (hl)                 | sp=0990, a=fb    bc=00e5, de=3642, hl=409c,
0939 20 03        ; jr nz, 0x93e             | sp=0990, a=fb    bc=00e5, de=3642, hl=409c,
0936 dd 34 00     ; inc (ix + 0x0)           | sp=0990, a=fb    bc=00e5, de=3642, hl=409c,
093e 21 9c 40     ; ld hl, 0x409c            | sp=0990, a=fb    bc=00e5, de=3642, hl=409c,
0937 34           ; inc (hl)                 | sp=0990, a=fb    bc=00e5, de=3642, hl=409c,
0941 35           ; dec (hl)                 | sp=0990, a=fb    bc=00e5, de=3642, hl=409c,
0939 20 03        ; jr nz, 0x93e             | sp=0990, a=fb    bc=00e5, de=3642, hl=409c,
exiting...
```

warnings from **memory** module

exit caused by (invalid) memory pattern

hexdump of all modified memory

backtrace of the last 9 instructions
(if step size is 1, else garbage)

```
jdc = {
    "descr": "Combined Q1 image from IC25-IC32",
    "start": 0x0000,
    "data": [
        ["file", "roms/JDC/IC25.BIN", 0x0000],
        ["snippet", [0xff], 0x1000] # force different disk access path
    ],
    "funcs" : {
       0x01f8: "01f8 clear RAM from 4089 to 40ff",
       0x4086: "4086 wait_for_kbd_or_printer()"
    },
    "pois" : {
        0x0033: 'PROCH',
    },
    "known_ranges" : [
        [0x0000, 0x003e, 'jump tables'],
        [0x01de, 0x01e4, 'interrupt routine()'],
        [0x01e5, 0x020e, 'start()'],
        [0x020f, 0x0278, 'Main program loop'],
        [0x0279, 0x0292, 'load and run program'],
    ],
        "pl1" : {
            0x1c12: 'INST 09 - compare character strings',
            0x1c37: 'INST 07 - compare binary numbers',
        }
}
```

## Program abstraction

Used by loader to initialise memory with programs and data

and by emulator and disassembler for annotations and debugging

```
> python3 disassemble.py -a

loading program: Combined Q1 image from IC25-IC32
loaded 1024 bytes from roms/JDC/IC25.BIN at address 0000h
loaded 1024 bytes from roms/JDC/IC26.BIN at address 0400h
...
loaded 1024 bytes from roms/JDC/IC32.BIN at address 1c00h

;jump tables
0000 c3 e5 01     ; jp 0x1e5              | reset START
0003 c3 77 00     ; jp 0x77               | TOSTR
...
003b 00           ; nop                   |
003c c3 67 07     ; jp 0x767              | SHIFTY

;Return Address (copied to 4080)
003f c3 3f 00     ; jp 0x3f               |

;JP to interrupt routine (copied to 4083)
0042 c3 b1 02     ; jp 0x2b1              |

;JP to wait-for-keyboard-or-printer
0045 c3 15 08     ; jp 0x815              |

;UNEXPLORED
0048 c9           ; ret                   |
0049 c9           ; ret                   |
004a c9           ; ret                   |
004b c9           ; ret                   |
004c c3 62 03     ; jp 0x362              |
004f c3 34 07     ; jp 0x734              |
```

## Program abstraction

disassambly example

uses known_ranges, pois

Helps identify poorly understood or unexplored regions

# Disk image generation – from floppy tools

```
# 5000xxx2/50001610/50001610.ft_cache
data = [
    [   # Track 0
        [0x9e, 0x00, 0x00],
        [0x9b, 0x00, 0x00, 0x49, 0x4e, 0x44, 0x45, 0x58, 0x20, 0x20, 0x20, 0x
        #      0000494e4455820202011002800820000000000000000000000000000000
        [0x9e, 0x00, 0x01],
        [0x9b, 0x00, 0x00, 0x44, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x
        #      0000442020202020200900ff001e000100010000000000000000000000000
        [0x9e, 0x00, 0x02],
        [0x9b, 0x00, 0x00, 0x50, 0x52, 0x49, 0x4e, 0x54, 0x20, 0x20, 0x20, 0x
        #      00005052494e542020200300ff001e000200020000000000000000000000
```

# Disk image generation – hand crafted

```
def pl1(txt):
    return [0x9b] + [ ord(x) for x in txt] + [0x20 for x in range(79 - len(txt))]

data = [
    [ 0x9e, 0x04, 0x00], pl1('X1 = 1;'),
    [ 0x9e, 0x04, 0x01], pl1('X2 = 1;'),
    [ 0x9e, 0x04, 0x02], pl1('LOOP: X3 = X2 + X1;'),
    [ 0x9e, 0x04, 0x03], pl1("PUT FILE(DISPLAY) EDIT (X3) (P'ZZZZZZZZZZ9');"),
    [ 0x9e, 0x04, 0x04], pl1('X1 = X2;'),
    [ 0x9e, 0x04, 0x05], pl1('X2 = X3;'),
    [ 0x9e, 0x04, 0x06], pl1('IF (X3 < 4807526977) THEN GO TO LOOP;'),
    [ 0x9e, 0x04, 0x07], pl1('GET SKIP LIST(A);'),
    [ 0x9e, 0x04, 0x08], pl1('END;'),
```

**Shortcomings**

| | |
|---|---|
| display | prints only ascii (not Q1 ROMS) |
| keyboard | ascii, Fn keys working |
| serial comms | unexplored |
| printer | rudimentary support |
| floppy disk | only simple write operations |
| hard drive | unexplored |
| timer | some support, tested by RTC program |
| interrupts | mostly untested, keyboard uses fake interrupts |

## References

datamuseum.dk/wiki/Q1_Microlite

### Emulator

q1-lite-emulator.readthedocs.io/en/latest/

github.com/Datamuseum-DK/Q1-Emulator

### ROMs and documentation

www.peel.dk/Q1/

www.thebyteattic.com/p/q1.html

www.1000bit.it/database2.asp?id=698

github.com/MattisLind/q1decode

technikum29.de/de/geraete/Q1_lite/Q1_lite_Details.php

# Demo