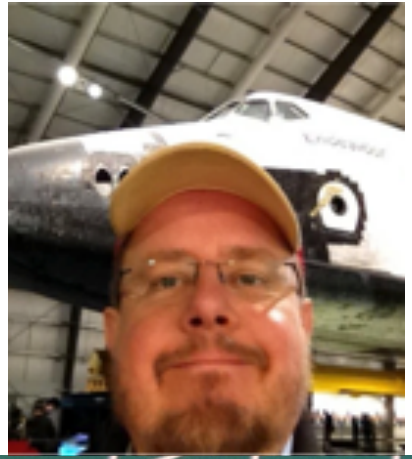
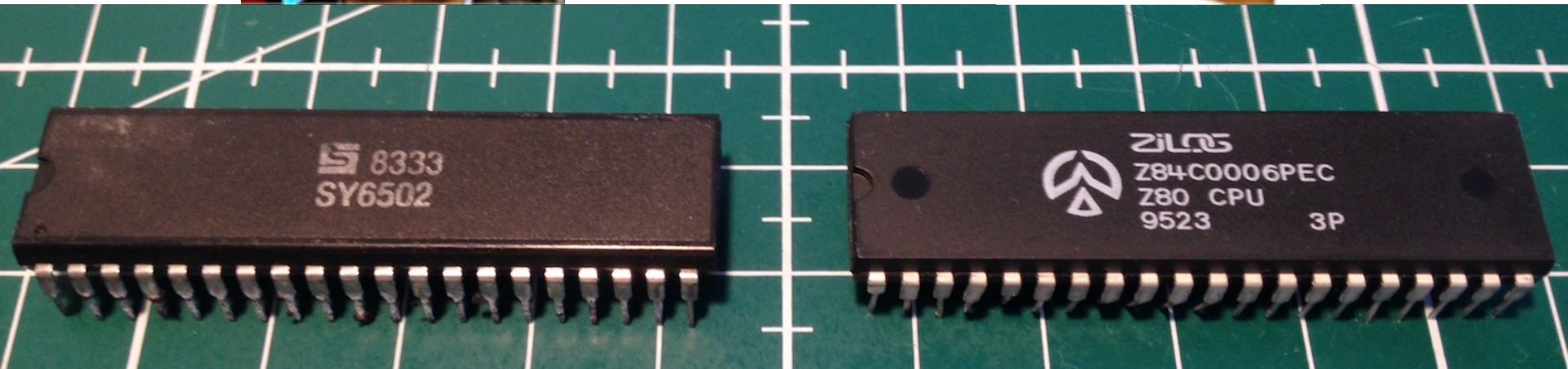
Building a TTL microcomputer without a microprocessor

Hackaday Conference Belgrade 2018

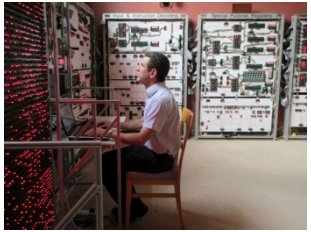
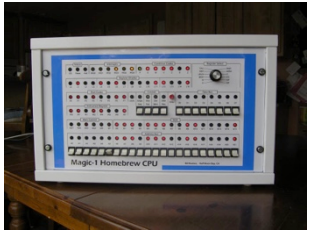
Marcel van Kervinck
Walter Belgers

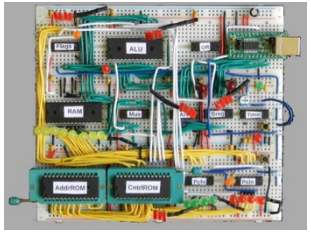
About us



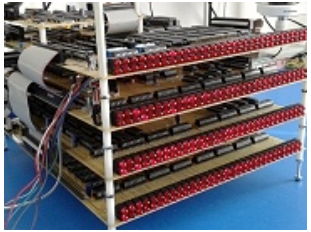
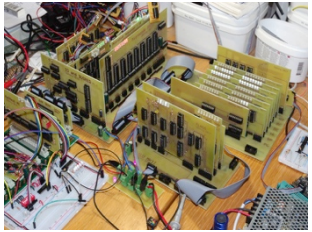
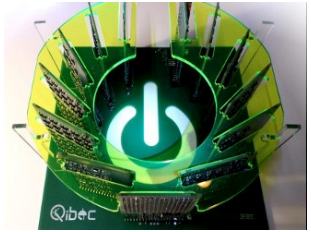


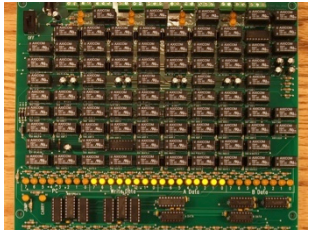
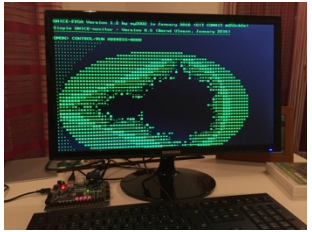
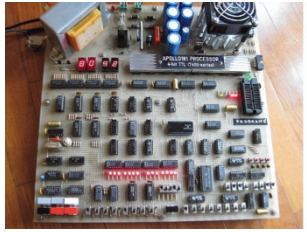
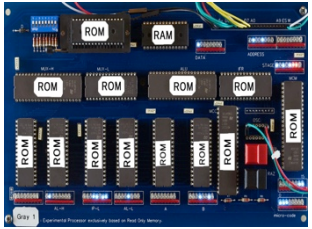
Building your own CPU

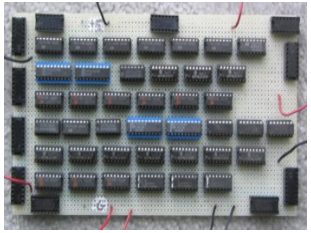
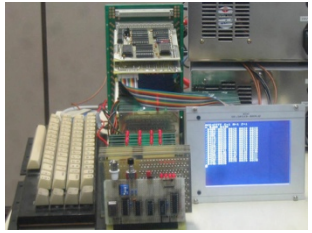
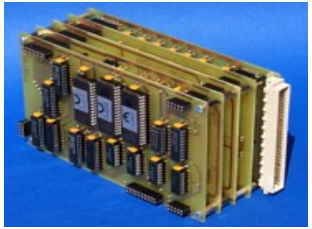








https://www.homebrewcpuring.org

Before you begin

What core building blocks?

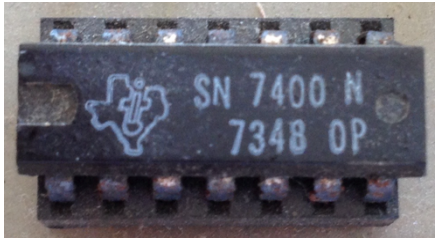
FPGA, SSI logic chips, NAND gates, discrete transistors, tubes, relays, steam punk, …

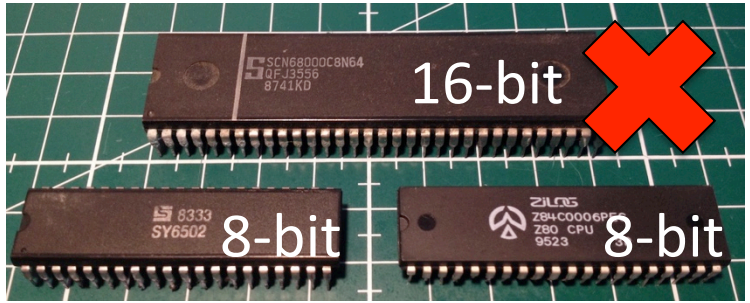
Data path size?

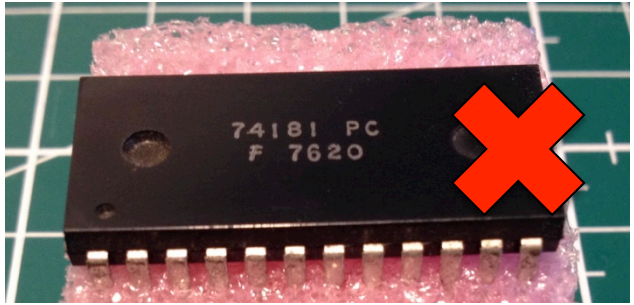
64 bits, 32 bits, 16 bits, 8 bits, 4 bits, 1 bit, other…

Standard ALU chips or custom?

74181 chips (4-bit ALU)?







Our choices:

7400 series logic "TTL"

8-bit system

No complex chips

Much more to consider

Harvard or Von Neumann?

Microprogramming or RISC?

Pipelining yes or no?

Existing instruction set or own?

Peripherals, extendibility, power, ..

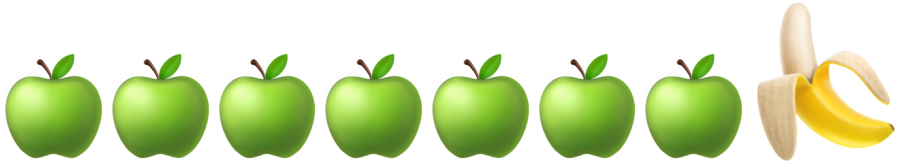
Time and budget?

1-2-3 days per week for 3m-6m-1yr

700–1000 euro to first PCB

It better be fun

Most important: what makes yours unique?



Ours is an exercise in minimalism

Rule 1 Absolutely <u>no complex logic chips</u>

74HC595 shift-register is “borderline OK”: ALUs, UARTs, are a no-go

Rule 2 <u>Single board</u> with 30-40 chip count

Same ballpark as Wozniak’s Break Out, early PC video cards or the “Ben Eater” breadboard type of computers

Rule 3 Still capable of <u>video games</u> with sound

Let software do the job of complex video and sound ICs

Bonus Nice <u>retro look</u> and hopefully still <u>somewhat useful</u>

Green 2-layer PCB, thick easy-to-follow traces, manual routing, through-hole components, some built-in games and can at least be a clock ☺

Pragmatism beats idealism

Be replicable and fit in the world of today

No obscure 1970s DRAM from E-Bay but standard 62256 SRAM

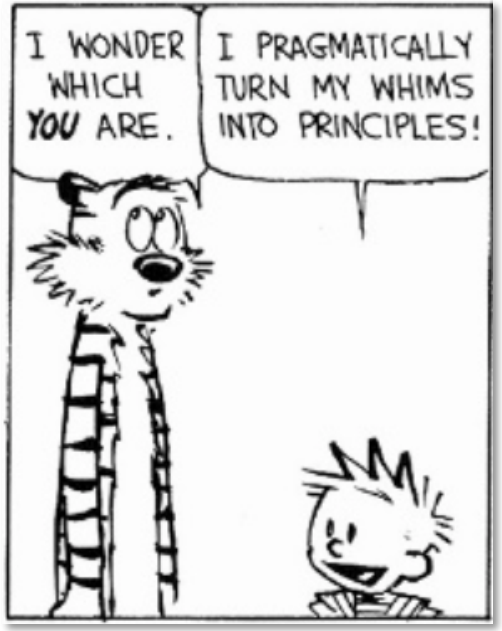
VGA video out and power over USB

Both are simple and commonly available

Switch from 74LS to 74HCT series for lower power

Operates on TTL levels using FETs inside → safe for USB ports

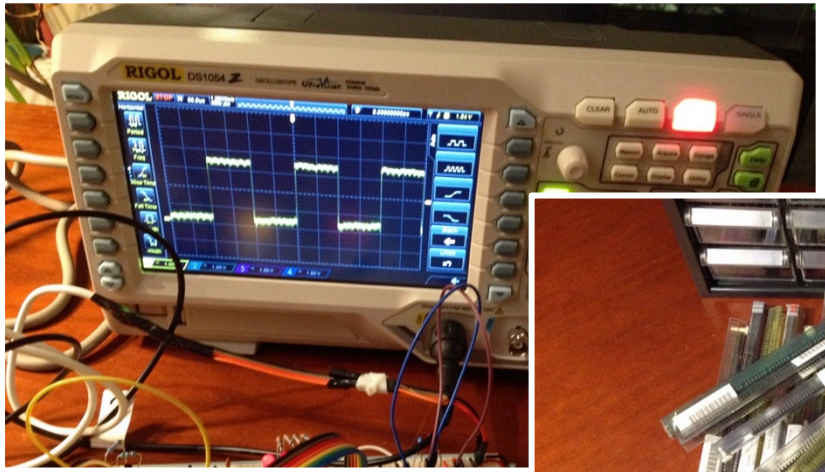
But also stay fully compatible with 74LS

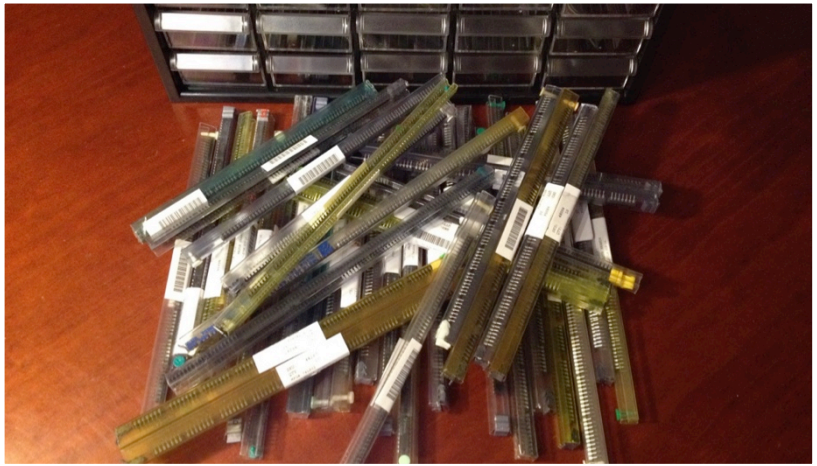


Get results: postpone stuff that threatens to drain your time

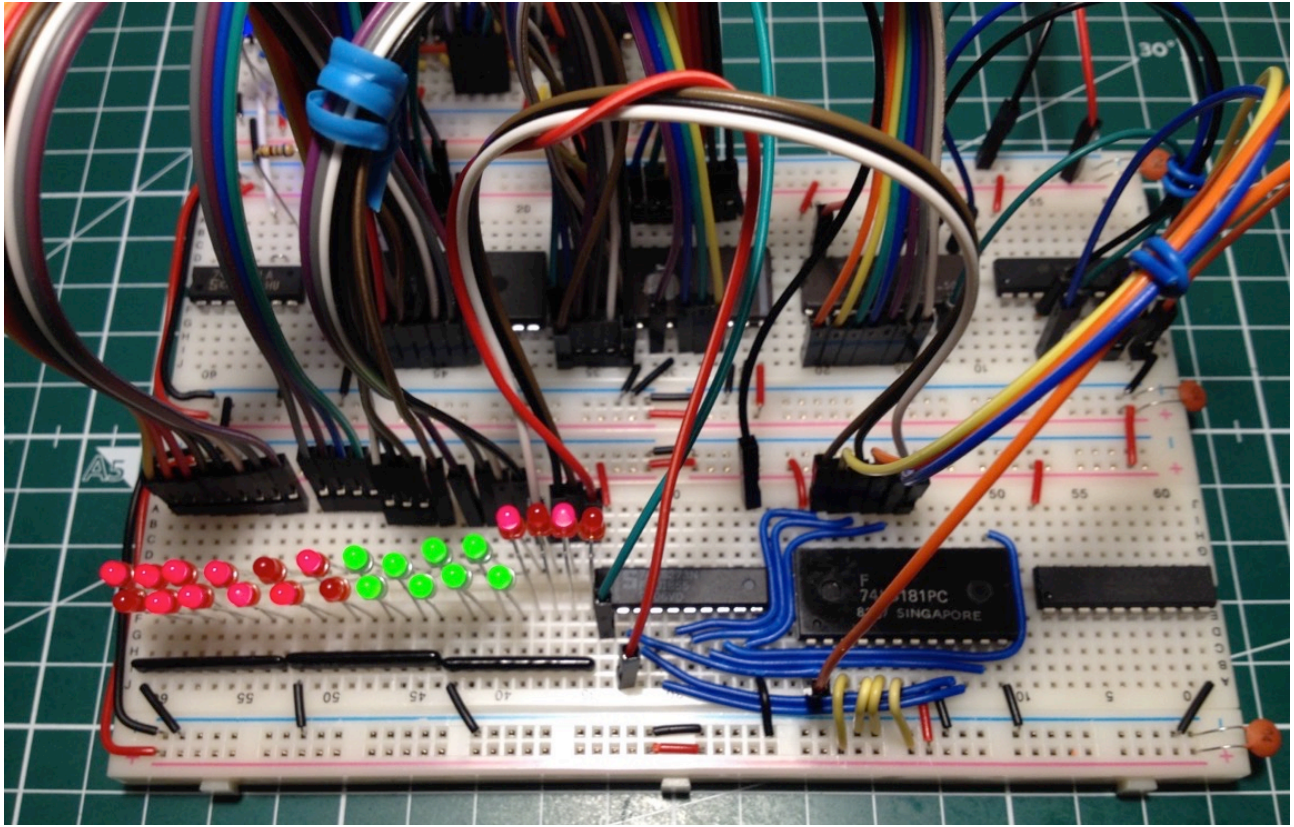
Accept to drop some ideals (later more on those)

Buy books, tools and hundreds of 7400-series chips …

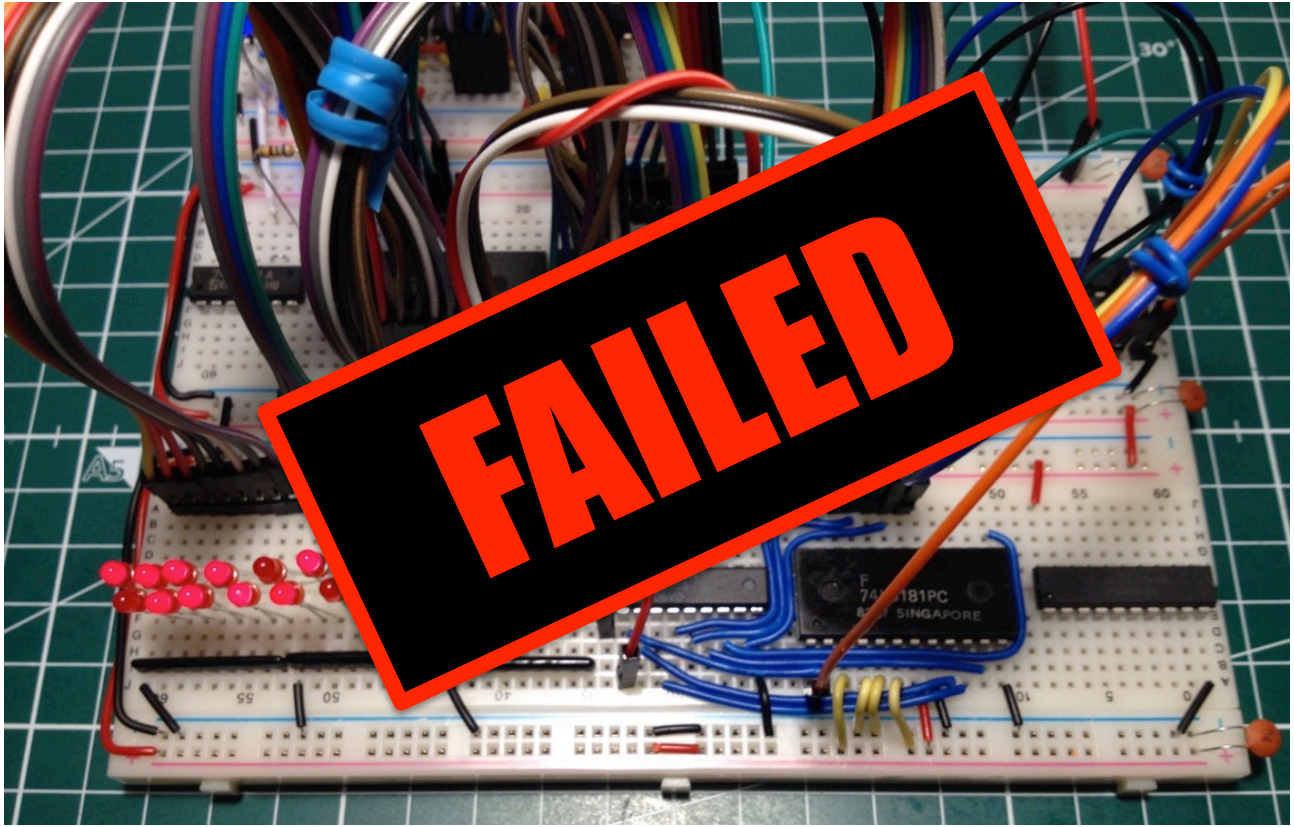




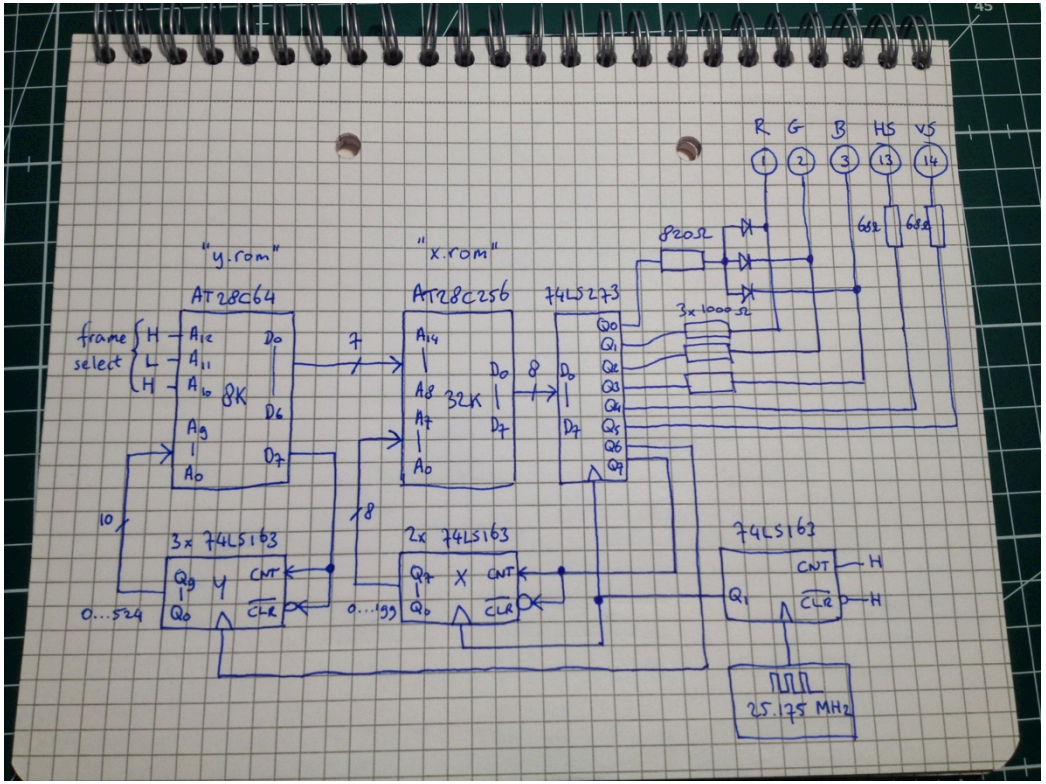
… and you can build a 4-bit computer!

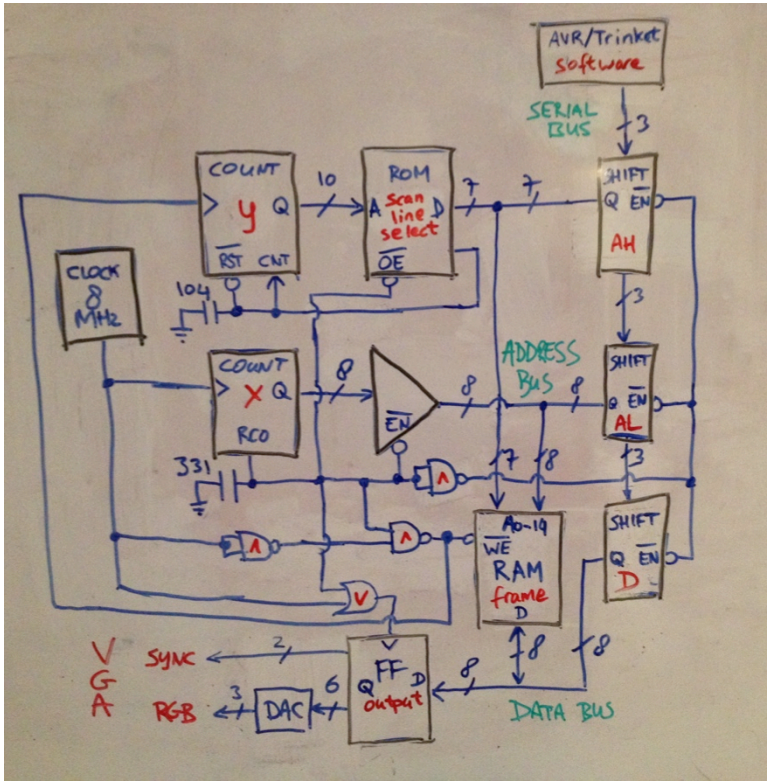


… and you can build a 4-bit computer!

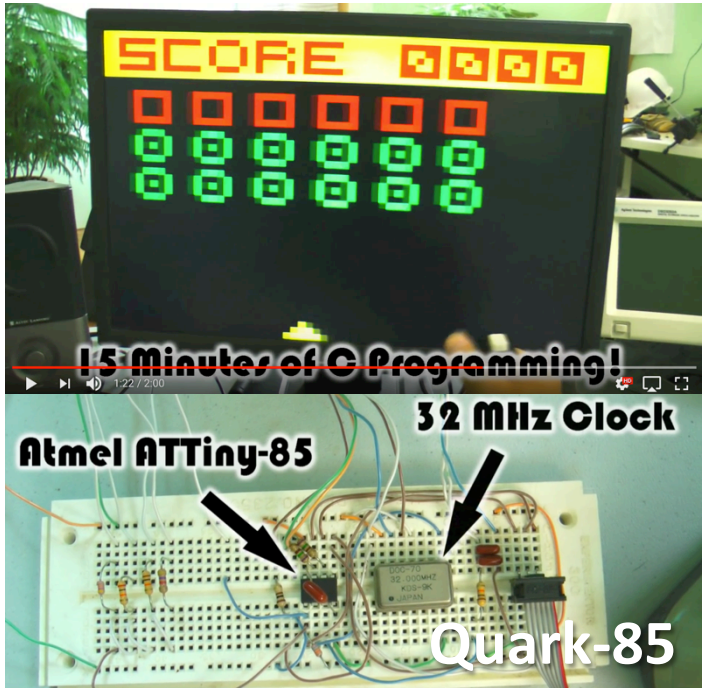


Document to help you think





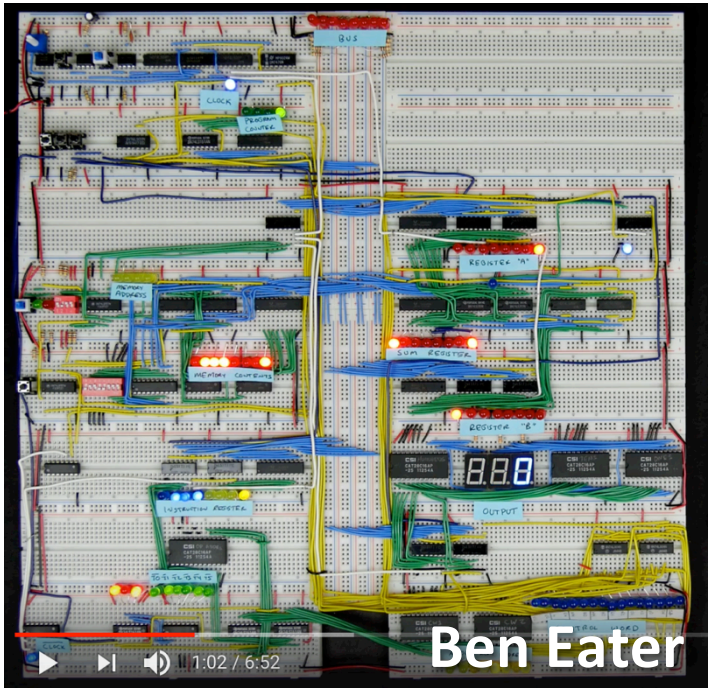
Look around for inspiration



ATtiny85 with 3 usable I/O lines, 512 bytes RAM does color VGA, 4 voice sound and joystick input. Software can bit-bang VGA!

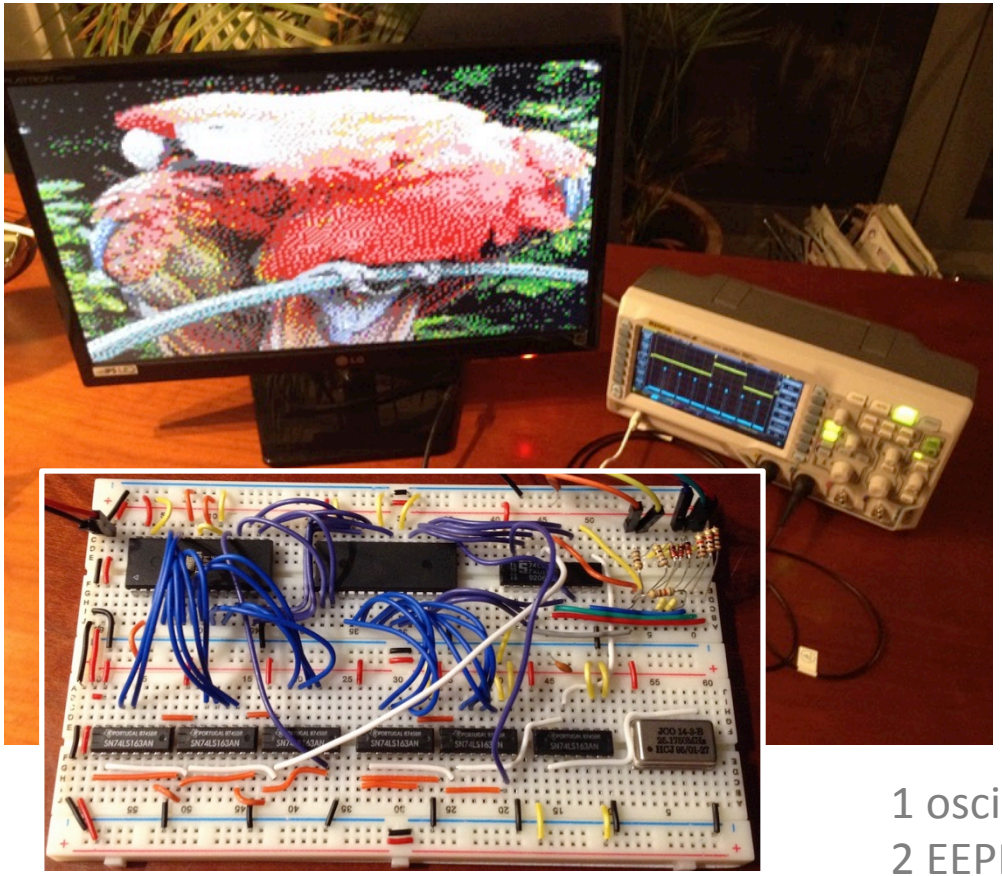
+

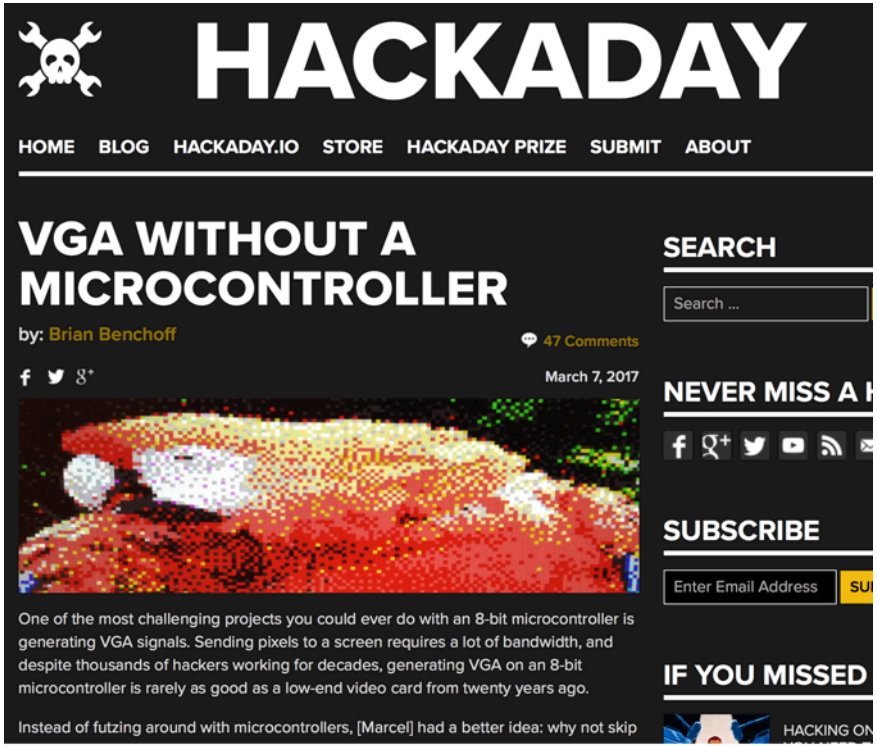
?



Breadboard computer based on text book SAP-1 design (“Simple As Possible”). Great educational YouTube series for 7400-series

Restart and make small concepts work first





HACKADAY

HOME BLOG HACKADAY.IO STORE HACKADAY PRIZE SUBMIT ABOUT

VGA WITHOUT A MICROCONTROLLER

by: Brian Benchoff

47 Comments

March 7, 2017

One of the most challenging projects you could ever do with an 8-bit microcontroller is generating VGA signals. Sending pixels to a screen requires a lot of bandwidth, and despite thousands of hackers working for decades, generating VGA on an 8-bit microcontroller is rarely as good as a low-end video card from twenty years ago.

Instead of futzing around with microcontrollers, [Marcel] had a better idea: why not skip

SEARCH

Search ...

NEVER MISS A

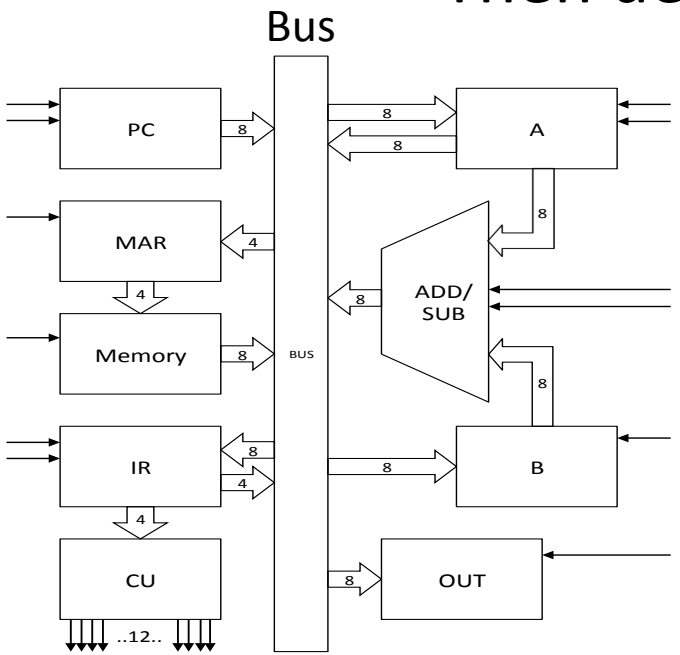
SUBSCRIBE

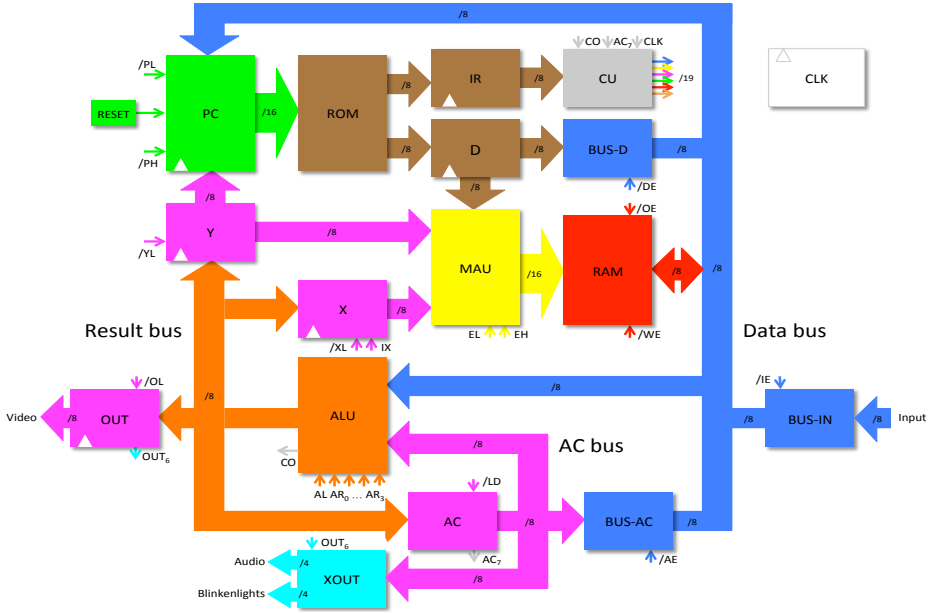
Enter Email Address

IF YOU MISSED

1 oscillator (25.175 MHz), 6 counters (4-bits),
2 EEPROM (8K + 32K) and 1 register (8-bits)

Then design the data flow





Reference design

- Von Neumann architecture
- 1 central bus is bottleneck: complexity↑
- Must be microcoded: speed↓↓

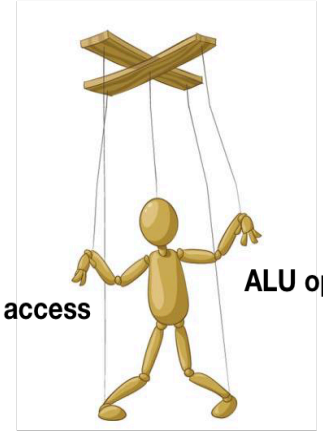
Our design

- Harvard architecture
- Split bus for efficiency: chip count↓
- Can do 1 instruction per cycle: speed↑↑

Control Unit last to define an instruction set

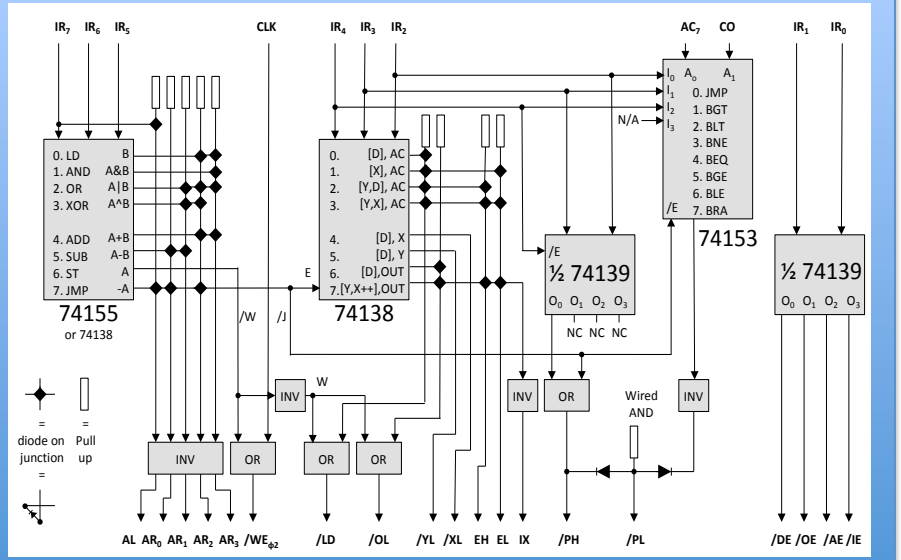
0	1	0	1	0	1	0	0

instruction
fixed length



Instruction and mode define what all units do

Map 8 instruction bits …

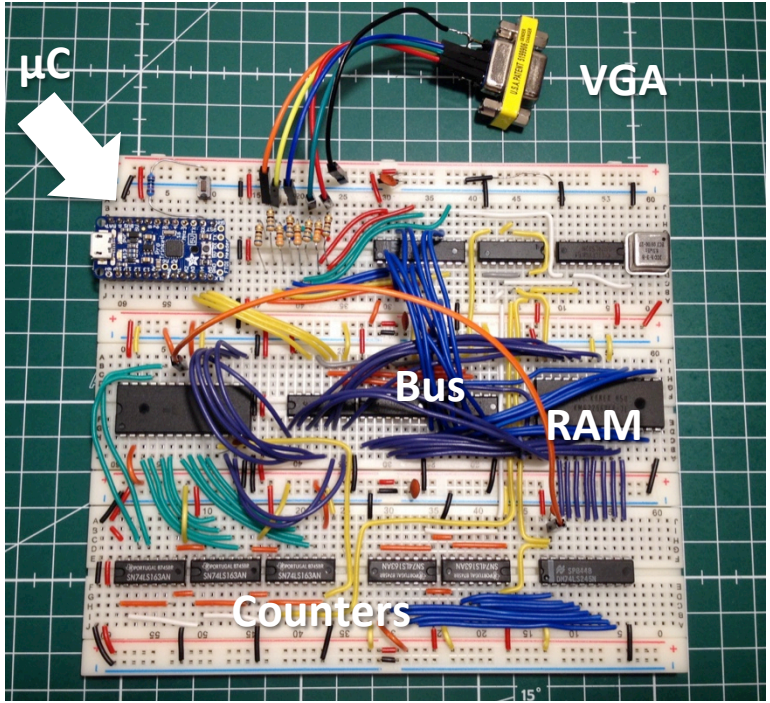


… to 19 control signals with
6 logic chips and 30 diodes

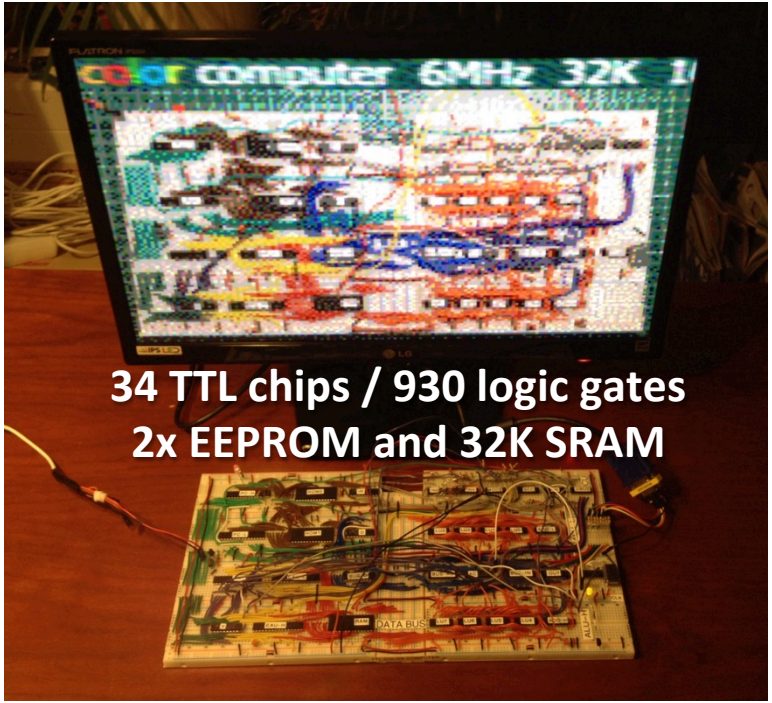
NOP	JMP
LD	BGT
AND	BLT
OR	BNE
XOR	BEQ
ADD	BGE
SUB	BLE
ST	BRA

16 native instructions,
32 modes (not all are useful)

Gradually extend scope when building



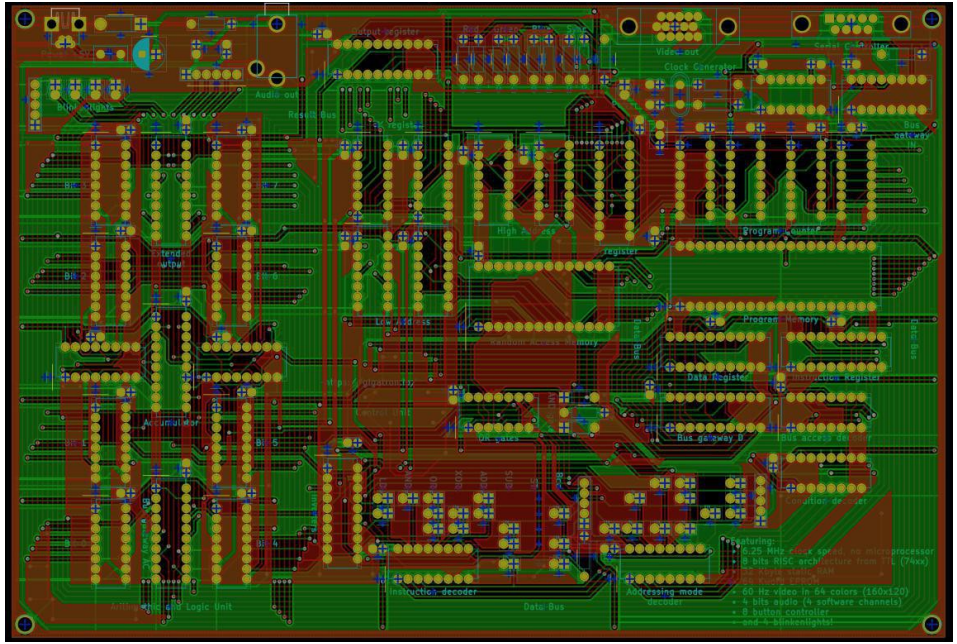
8 MHz breadboard dynamic VGA
from TTL logic and a 32K RAM.
A microcontroller to setup the RAM



Self-aware breadboard CPU at 6.3 MHz
with scrolling text (and a blinking LED).
“Look ma, no microcontroller!”

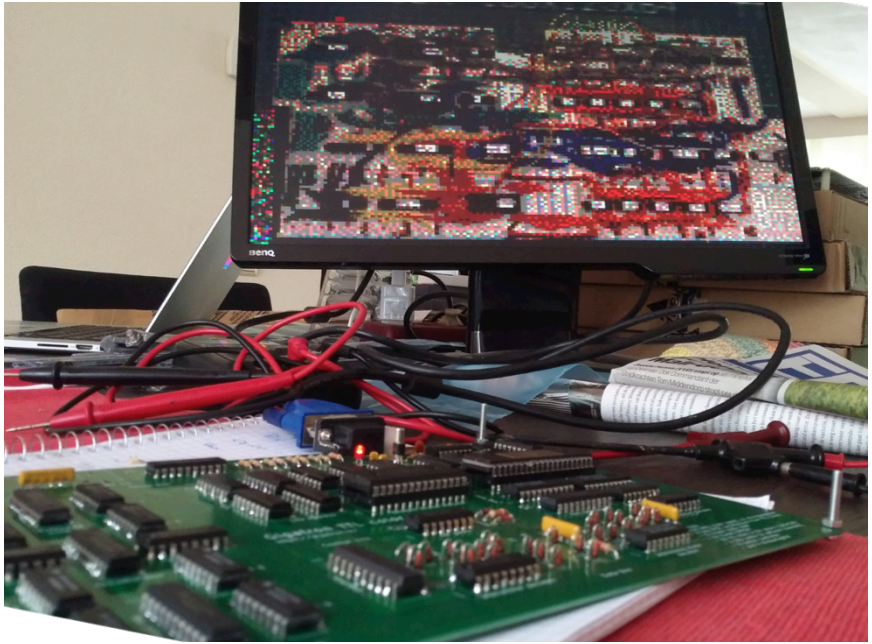
Learn to make a printed circuit board

About 10 weeks work



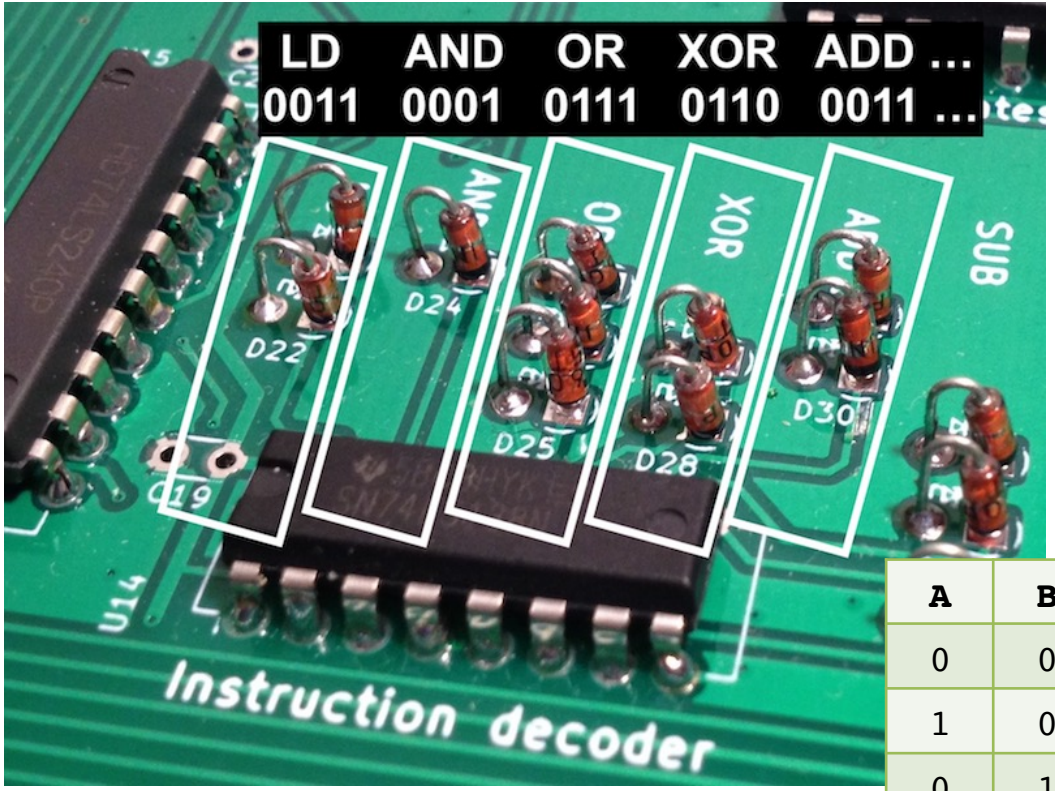
Done in Kicad4

First one could be brought to life \o/



PCB displaying an image of its prototype

Manually routing for interesting layout



For example: here the diodes visualize the truth tables for each operation

A	B	A∧B	A∨B	A≠B	B	~B
0	0	0	0	0	0	1
1	0	0	1	1	0	1
0	1	0	1	1	1	0
1	1	1	1	0	1	0

Do simple demos

Game with bricks, ball and bat

Application logic mixed with video generation loop.
No interrupts, so must count every instruction to keep VGA in sync. This is tedious.

Need a software stack

	On system	Offline
High level	**vCPU runs application code <u>from RAM</u>** SWEET16 inspired, Von Neumann! 34 self-timed instructions Can mix with native code	**GCL notation for vCPU programs** FALSE inspired (esoteric 1024-byte compiler) 😀Variables, if-then-else, loops, functions 😀Single pass, no linking stage: simple!
Low level	**Native code for hardware functions** Bit-bang VGA compatible signals, 4 channel sound, I/O, blinkenlights, reset, ROM as disk, and … an interpreted virtual CPU: **vCPU**	**Use Python itself as assembler** <u>Not</u> an assembler *in* Python! 😀Get a powerful macro assembler for free 🤫Python syntax for assembly **+ Emulators**

Loader

ROM disk

EPROM

Some programs we made with this

Minimalism at work

No standard instruction set

No interface adapter chips

No linear address space

No relative addressing

No flags register

No register file

No interrupts

No reset button

No timer chips

No sound chip

No video chip

No assembler

No compiler

No linker

Wise people stop here

"There is no product obscure enough that people are not interested in it."

Oscar "Obsolescence Guaranteed" Vermeulen

So we make it a kit! It sounds like fun and our friends ask for one…

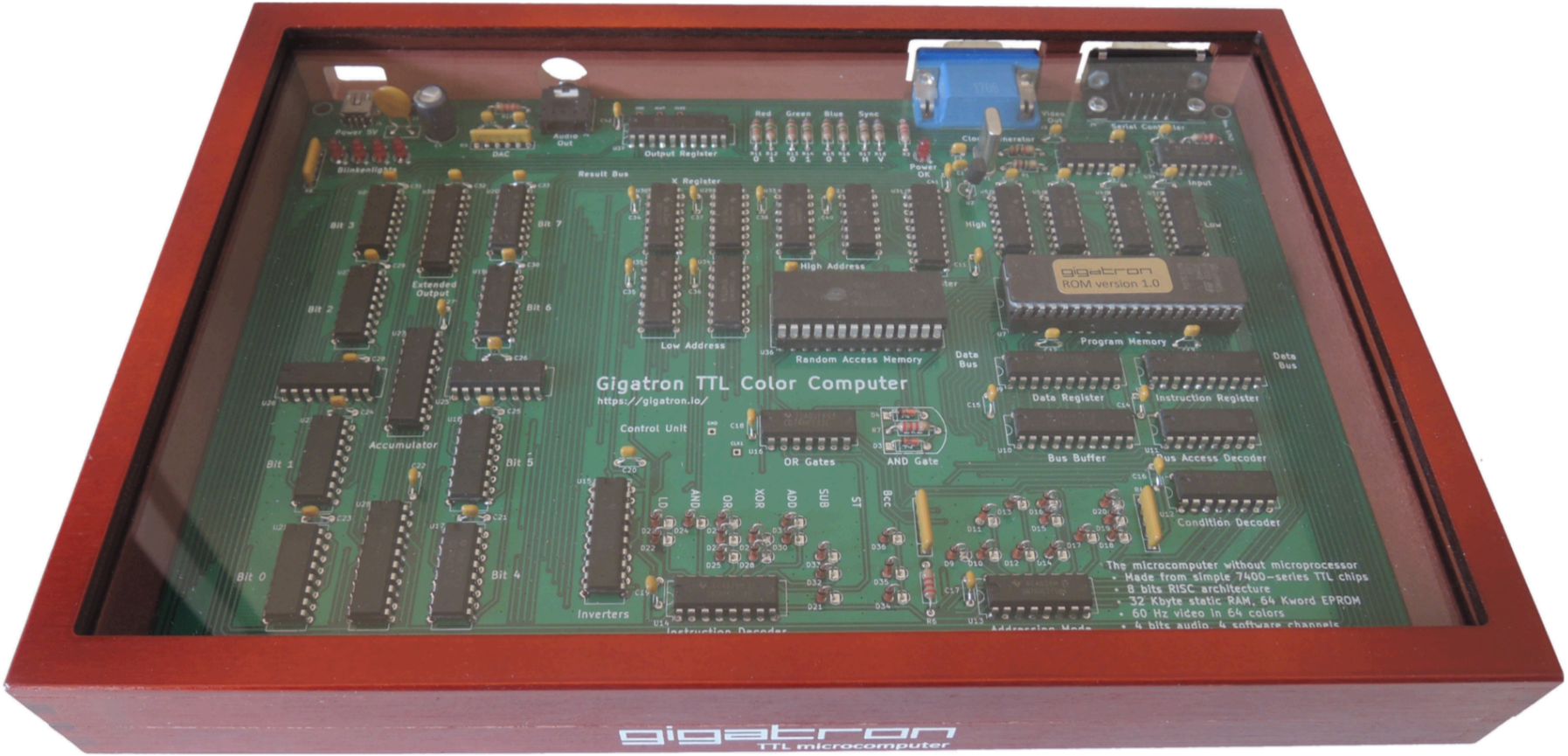
- Focus all efforts on 1st time right builds: assembly manual, videos, website
- The hardest part: stop working on new features for a while
- Talk a lot with other kit makers and potential users
- Find suppliers for quality parts
- All details matter
- Run beta-tests

Computer as a DIY soldering kit

Just need a soldering iron, a multi-meter and 3-4 hours of time to build. No oscilloscope needed

gigatron

TTL microcomputer

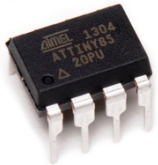

Gigatron TTL Color Computer
https://gigatron.io/
Power 5V
Blinkenlights
DAC
Audio Out
Output Register
Red
Green
Blue
Sync
Video Out
Serial Controller
Power OK
Input
Result Bus
X Register
Bit 3
Bit 7
Bit 2
Bit 6
Extended Output
Accumulator
Bit 1
Bit 5
Bit 0
Bit 4
Inverters
Low Address
High Address
Random Access Memory
Data Bus
High
Low
gigatron
ROM version 1.0
Program Memory
Data Register
Instruction Register
Bus Buffer
Bus Access Decoder
Condition Decoder
Control Unit
OR Gates
AND Gate
LD
AND
OR
XOR
ADD
SUB
ST
BCC
Instruction Decoder
Addressing Mode
The microcomputer without microprocessor
• Made from simple 7400-series TTL chips
• 8 bits RISC architecture
• 32 Kbyte static RAM, 64 Kword EPROM
• 60 Hz video in 64 colors
• 4 bits audio, 4 software channels
gigatron
TTL microcomputer

Not all ideals made it into “v1”

We wanted absolutely no rectangular PCB and no case from wood

We ended up with both and we’re quite happy with the result.

There is no *direct* keyboard hookup (yet)



Even the PS/2 protocol turns out to be an ugly beast.

So you must cheat a bit with hookup through a tiny µC. That works just fine.

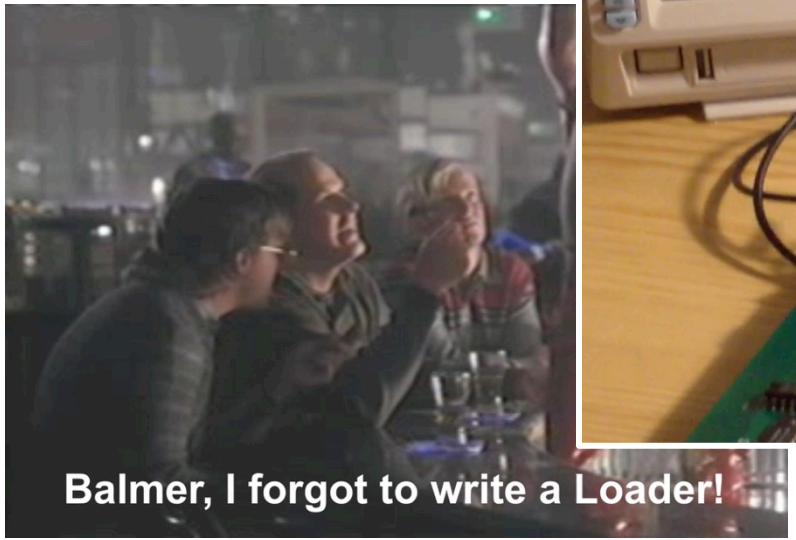
There is no built-in BASIC (yet)

Bill Gates doesn’t respond to our e-mails and it takes many weeks to write a BASIC.

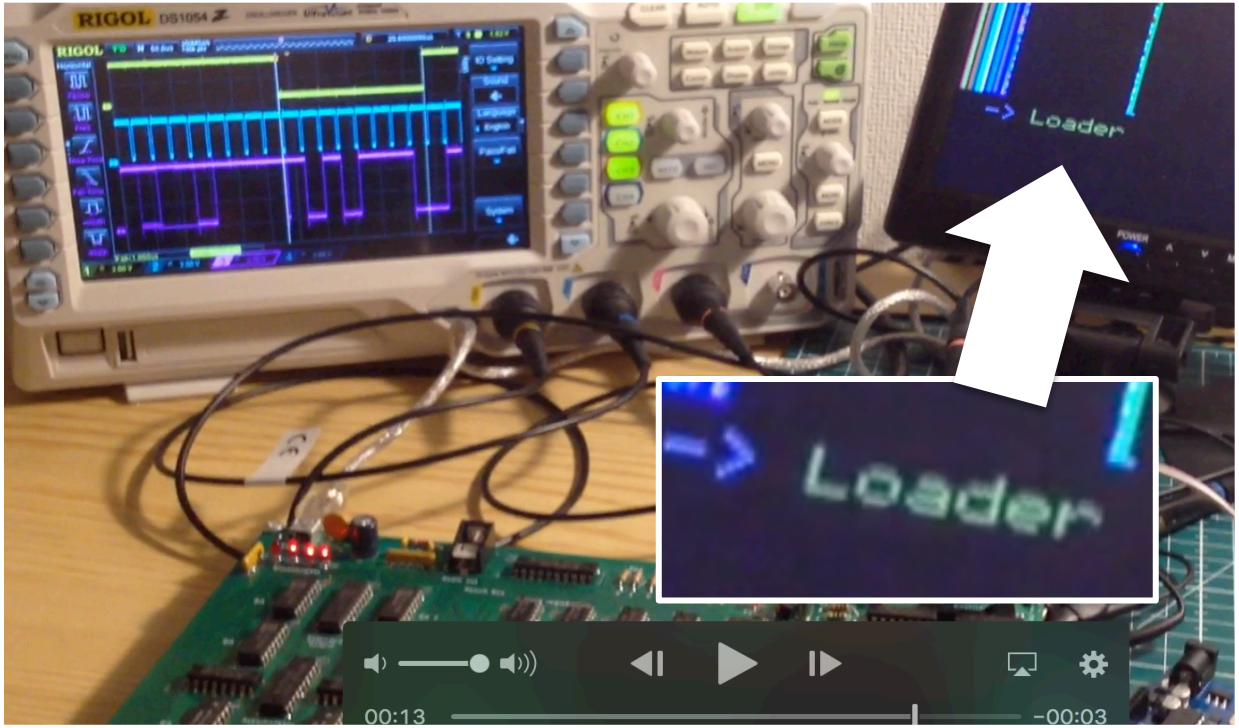
We would have liked a whole lot more blinkenlights

It makes sense at 1 millionth of the speed. Quite a few new parts needed for slow mode.

One thing we almost overlooked

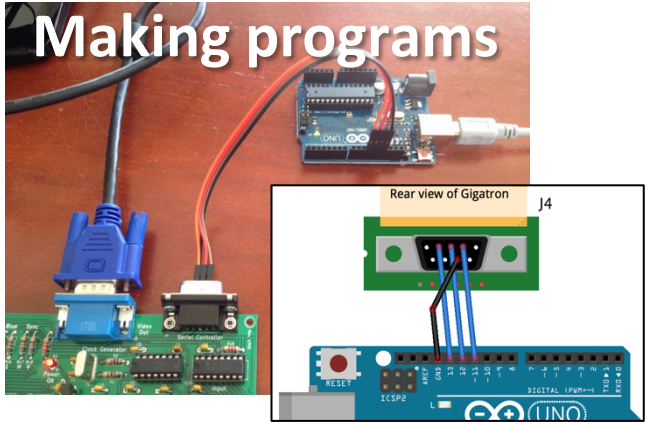


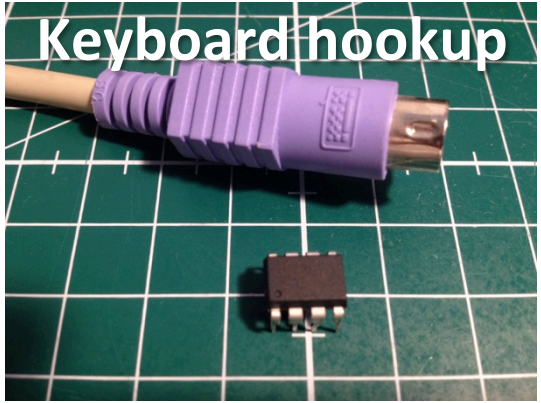
Pirates of Silicon Valley (movie)



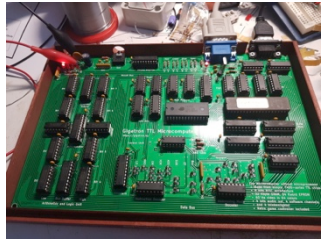
Loader is the escape hatch!

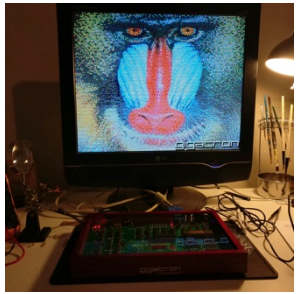
Much potential for true oldskool hacking!

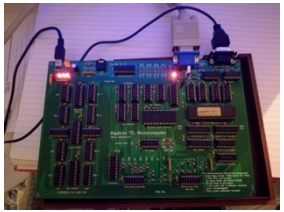


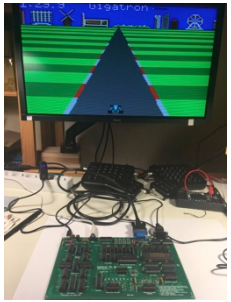


A new one is born every day







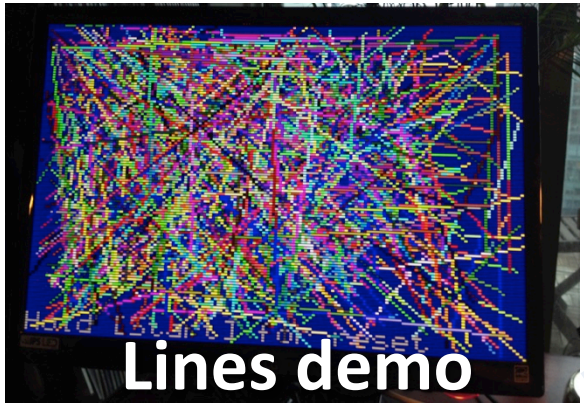


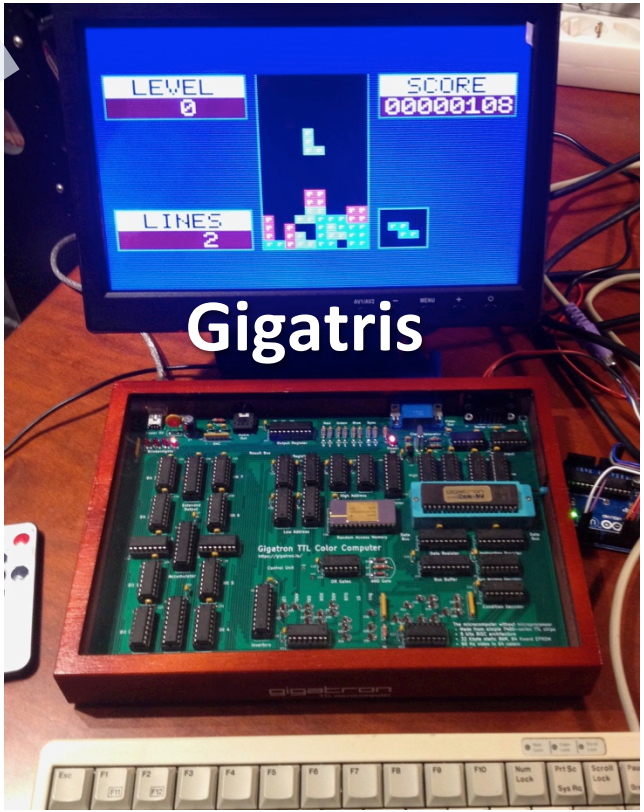
YouTube spreads the word

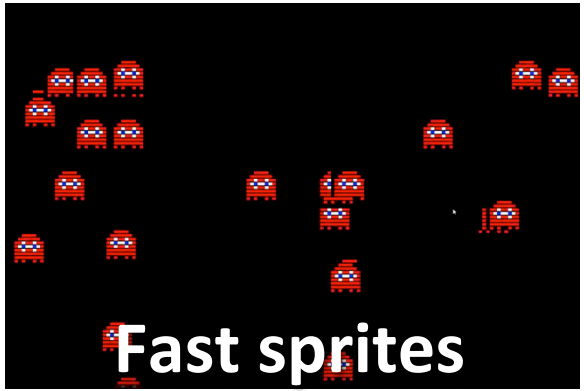
Buying more parts

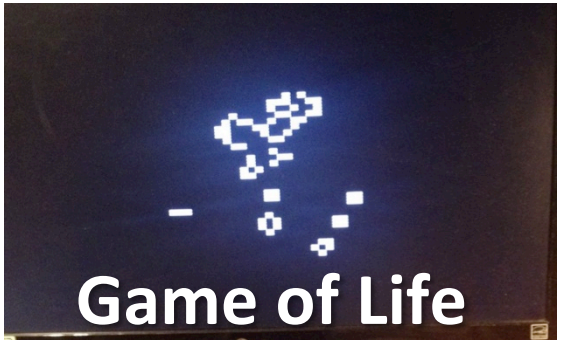
Tedious logistics

Community after first month





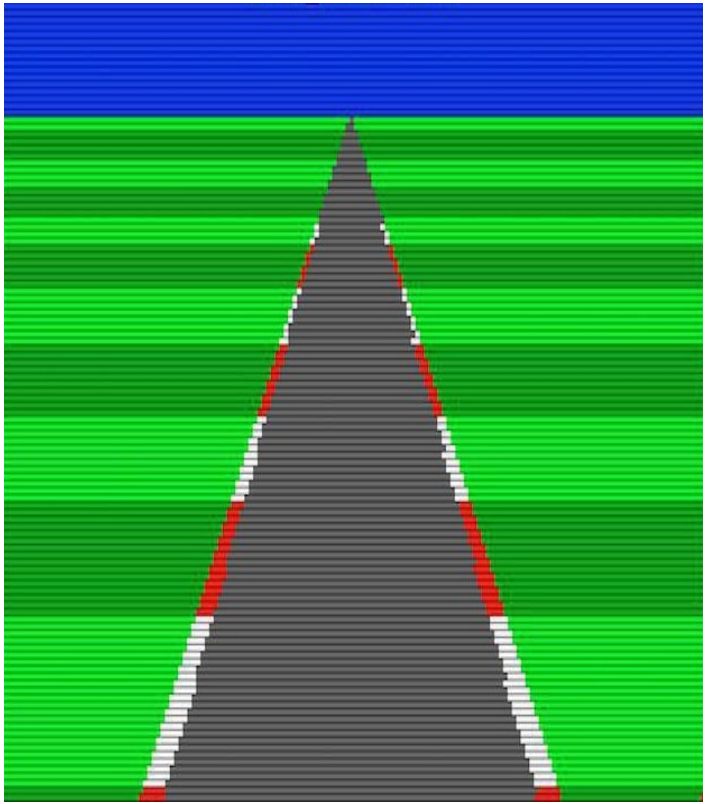




What's at the horizon?

1. Tutorials
   - Programming by simple examples
   - Hooking up a keyboard
   - Easy with tiny external µC to handle protocols
     Tougher nut to crack when allowing yourself at most 1 or 2 extra TTL chips
   - Consolidating your programs into an EPROM
2. Live hacking

   Monitor program. Onboard GCL interpreter?
3. Embedded BASIC?



Thank you for your attention!