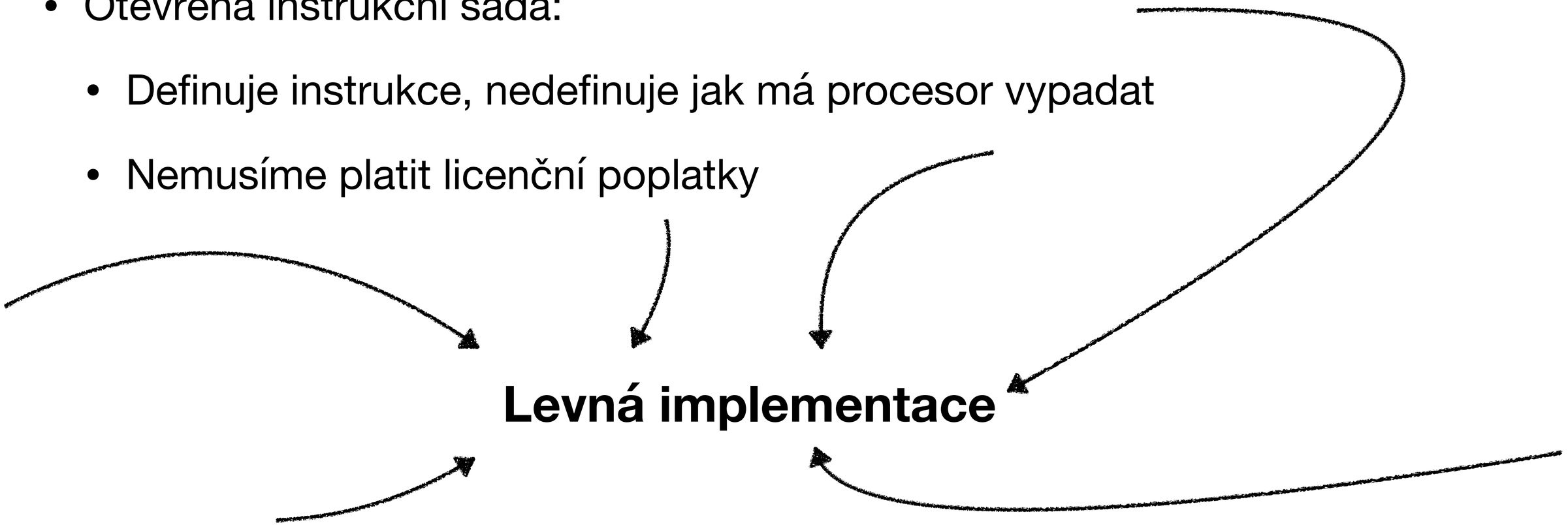
RISC-V - Aneb proč se nebát pochopit moderní architekturu

Filip Szkandera 2025

RISC-V obecně

- Relativně “nová” - vyvinuta v roce **2014** v Berkley
- Otevřená instrukční sada:
  - Definuje instrukce, nedefinuje jak má procesor vypadat
  - Nemusíme platit licenční poplatky



RISC

- RISC - **Reduced** Instruction Set Computer
  - RISC - Designová filozofie
  - RISC-V - Specifická implantace založená na filozofii RISCu

VS

CISC

- CISC - **Complex** Instruction Set Computer

Co se vlastně redukovalo?

Proč by se to jinak jmenovalo “redukovaná sada”?

- Procesory s “redukovanou” sadou **nejsou “hloupější”**, ale **optimalizovanější**
- Instrukce no-op může být u RISC architektury separátní instrukce
- U RISC-V se no-op enkóduje jako operace sčítání:
  - Přičteme nulu k nule [add zero, zero, zero]

Instrukční sady

Pro 32-bitovou architekturu

- Základní instrukční sada: **RV32I**
- Můžeme přidat rozšiřující instrukční sady pro různé funkce
- Některé z nich jsou “frozen”, jiné se ještě mohou změnit

Rozšiřující sady

M	Multiplication
A	Atomics
F	Floating point
B	Bit manipulation
V	Vector operations
C	Kompresované instrukce

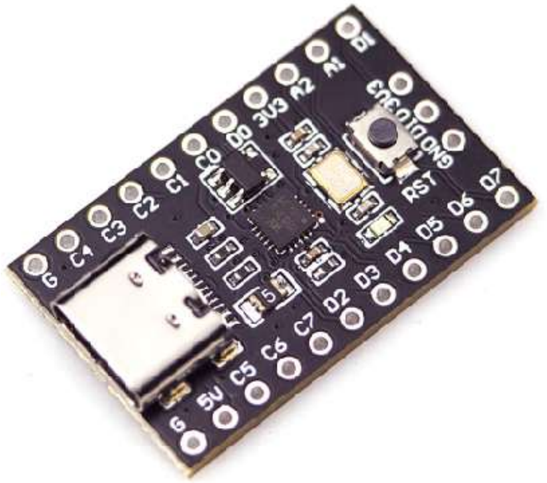
RV32I instrukce

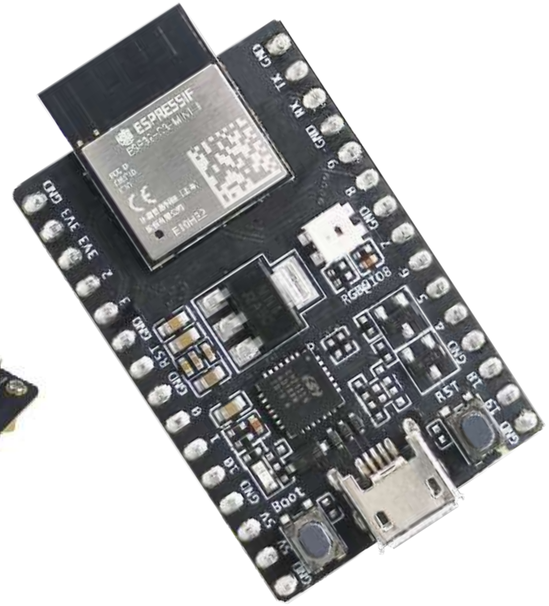
- Spousta instrukcí sdílí společné sekce:
  - rs1, rs2 a rd jsou vždy na stejném místě
  - Immediate instrukce se “snaží” mít co nejvíce stejných bitů na stejné pozici

		31 25	24 20	19 15	14 12	11 7	6 0
LUI	U	imm[31:12]				rd	0110111
AUIPC	U	imm[31:12]				rd	0010111
JAL	J	imm[20] imm[10:1]	imm[11]	imm[19:12]		rd	1101111
JALR	I	imm[11:0]		rs1	000	rd	1100111
BEQ	B	imm[12] imm[10:5]	rs2	rs1	000	imm[4:1] imm[11]	1100011
BNE	B	imm[12] imm[10:5]	rs2	rs1	001	imm[4:1] imm[11]	1100011
BLT	B	imm[12] imm[10:5]	rs2	rs1	100	imm[4:1] imm[11]	1100011
BGE	B	imm[12] imm[10:5]	rs2	rs1	101	imm[4:1] imm[11]	1100011
BLTU	B	imm[12] imm[10:5]	rs2	rs1	110	imm[4:1] imm[11]	1100011
BGEU	B	imm[12] imm[10:5]	rs2	rs1	111	imm[4:1] imm[11]	1100011
LB	I	imm[11:0]		rs1	000	rd	0000011
LH	I	imm[11:0]		rs1	001	rd	0000011
LW	I	imm[11:0]		rs1	010	rd	0000011
LBU	I	imm[11:0]		rs1	100	rd	0000011
LHU	I	imm[11:0]		rs1	101	rd	0000011
SB	S	imm[11:5]	rs2	rs1	000	imm[4:0]	0100011
SH	S	imm[11:5]	rs2	rs1	001	imm[4:0]	0100011
SW	S	imm[11:5]	rs2	rs1	010	imm[4:0]	0100011
ADDI	I	imm[11:0]		rs1	000	rd	0010011
SLTI	I	imm[11:0]		rs1	010	rd	0010011
SLTIU	I	imm[11:0]		rs1	011	rd	0010011
XORI	I	imm[11:0]		rs1	100	rd	0010011
ORI	I	imm[11:0]		rs1	110	rd	0010011
ANDI	I	imm[11:0]		rs1	111	rd	0010011
SLLI	I	0000000	shamt	rs1	001	rd	0010011
SRLI	I	0000000	shamt	rs1	101	rd	0010011
SRAI	I	0100000	shamt	rs1	101	rd	0010011
ADD	R	0000000	rs2	rs1	000	rd	0110011
SUB	R	0100000	rs2	rs1	000	rd	0110011
SLL	R	0000000	rs2	rs1	001	rd	0110011
SLT	R	0000000	rs2	rs1	010	rd	0110011
SLTU	R	0000000	rs2	rs1	011	rd	0110011
XOR	R	0000000	rs2	rs1	100	rd	0110011
SRL	R	0000000	rs2	rs1	101	rd	0110011
SRA	R	0100000	rs2	rs1	101	rd	0110011
OR	R	0000000	rs2	rs1	110	rd	0110011
AND	R	0000000	rs2	rs1	111	rd	0110011

Kde jej můžeme najít

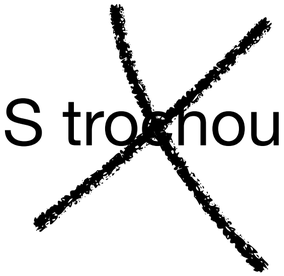
- Mikroprocesory:
  - (WCH) - CH32V003
  - (Espressif) - ESP32-C3, ESP32-H2, …
- SBC, počítače:
  - (SiFive) - SiFive Unmatched





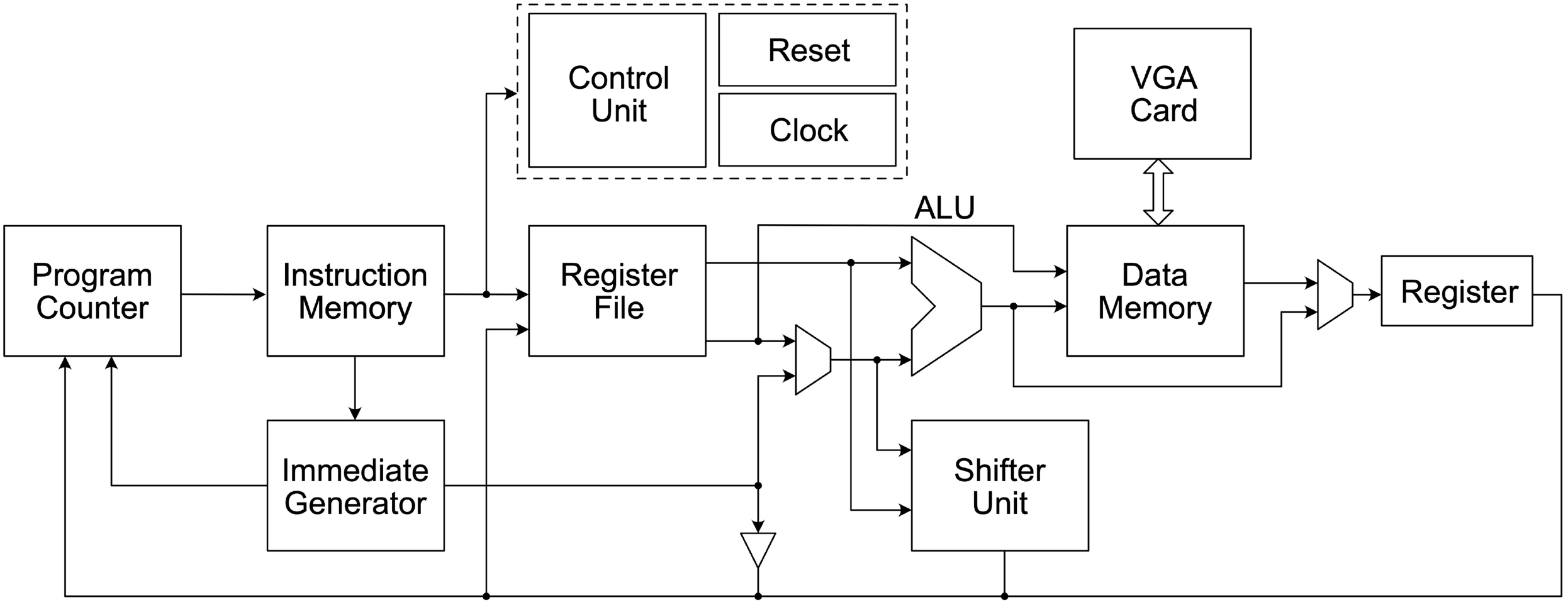
https://nuttx.apache.org/docs/latest/platforms/risc-v/esp32c3-legacy/boards/esp32c3-devkit/index.html
https://www.cnx-software.com/wp-content/uploads/2023/02/Cheap-CH32V003-RISC-V-development-board.jpg
https://external-content.duckduckgo.com/iu/?u=https://www.phoronix.net/image.php?id=sifive-riscv-unmatched&image=sifive_unmatched_1_show&f=1&nofb=1&ipt=4a43dd67e9ca47807e7de19a1061cd023b6cf4c8a5207e8481951063b2e5cd07

Lze si postavit vlastní RISC-V?

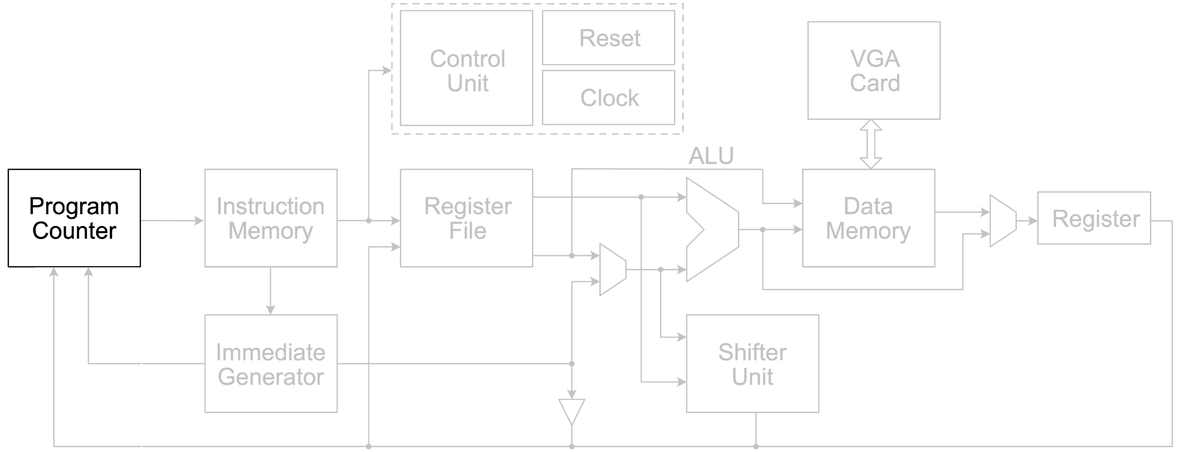


S trochou trpělivosti….

Blokový diagram

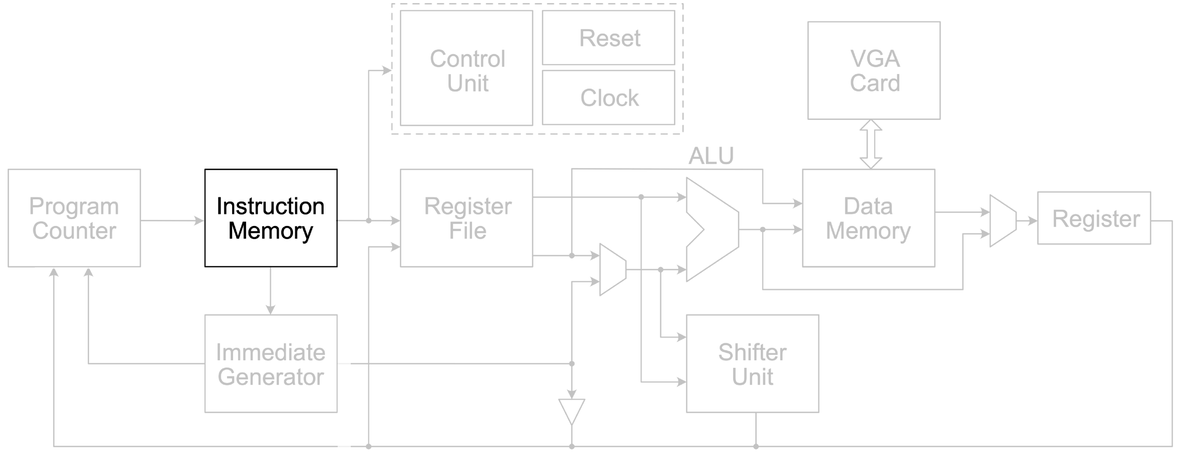


Program Counter

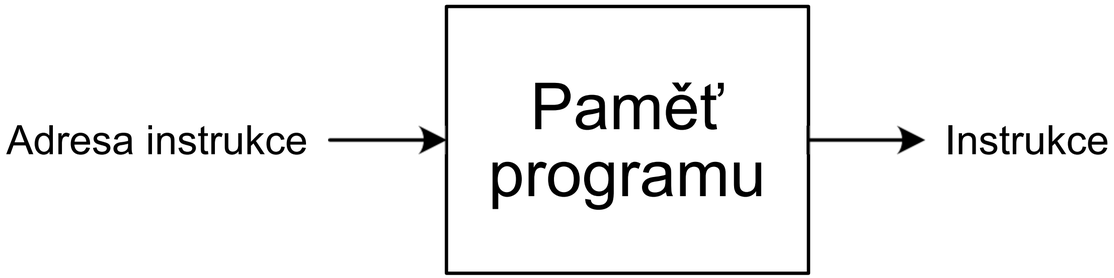


- Uchovává adresu právě vykonávané instrukce
- Obsahuje “malou ALU” pro výpočet:
  - Nadcházející instrukce (+ 4)
  - Skoku [jal, jalr]

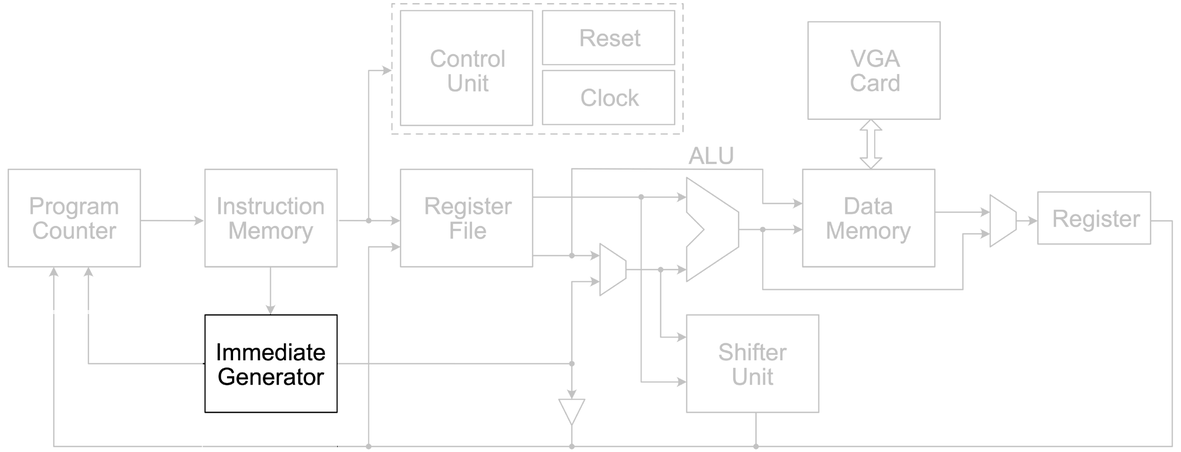
Instruction Memory



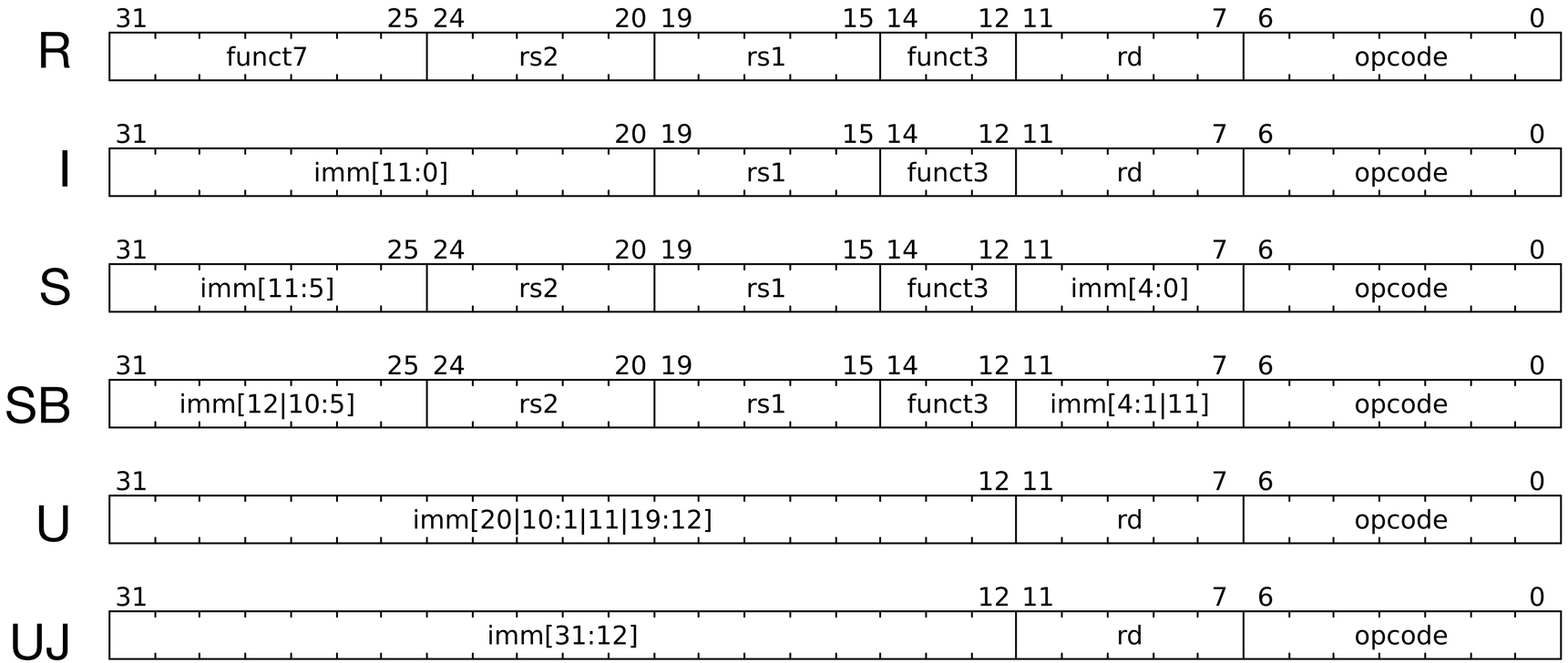
- Paměť instrukcí - uchovává celý program
- Sestavená z EEPROM pamětí (“non volatile”)



Immediate Generator

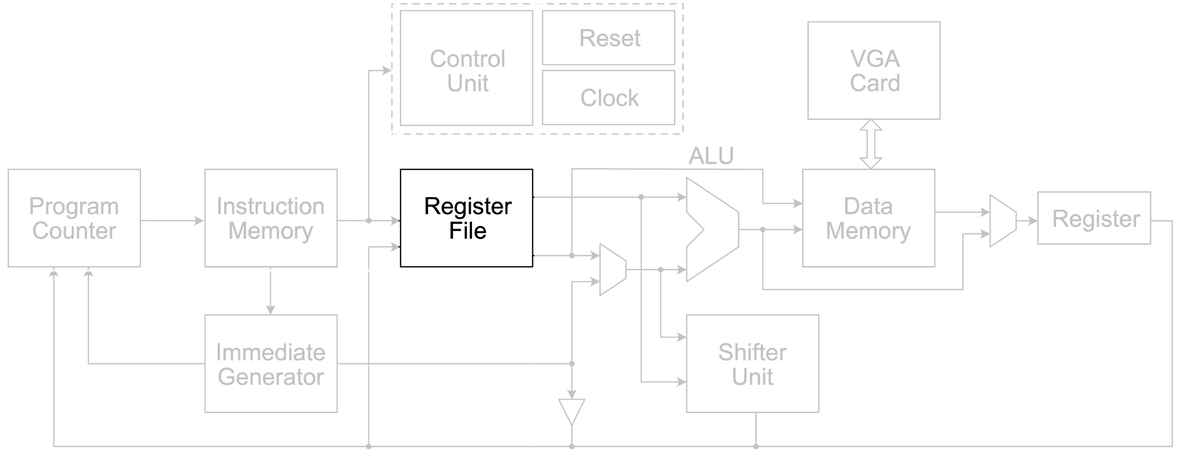


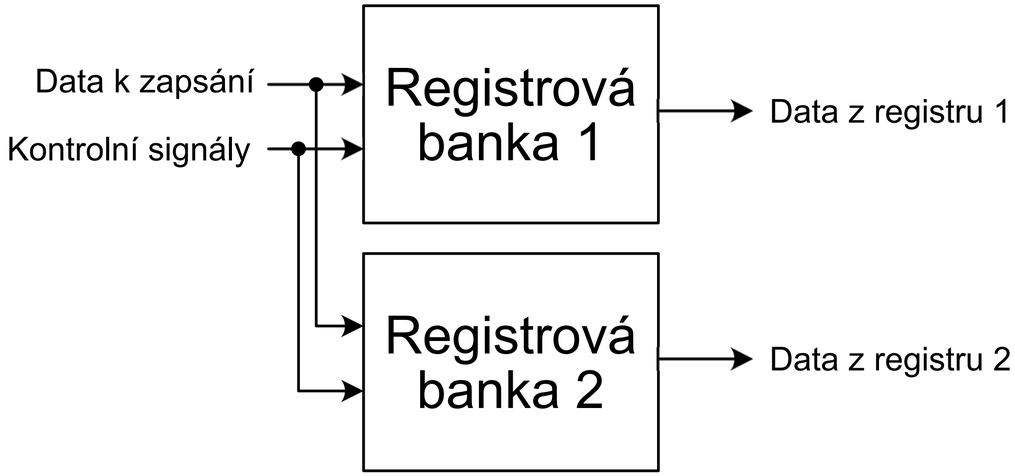
- “Vytahuje” konstanty z instrukce dle formátu;



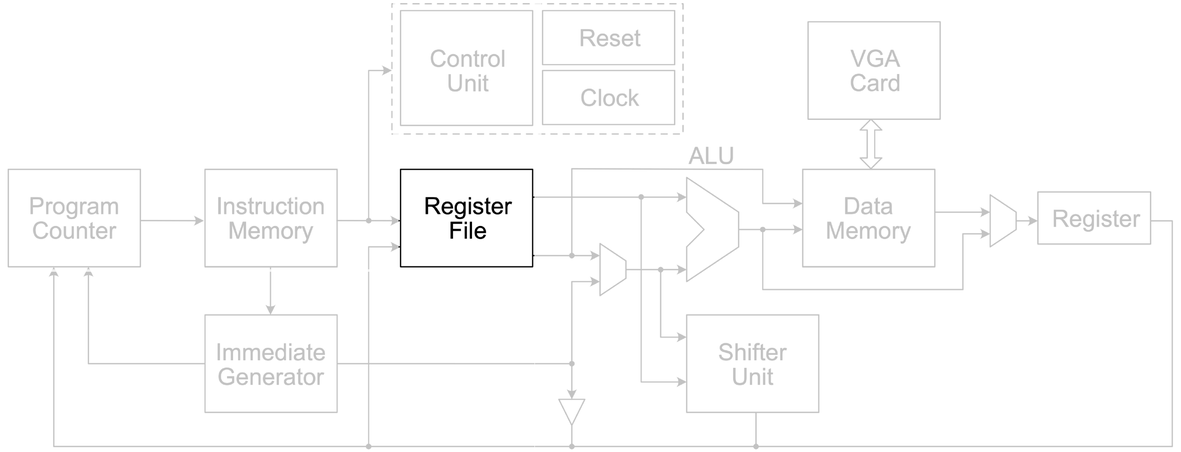
Register File

- Uchovává 32 registrů (po 32 bitech)
- Můžeme vyčítat až dva registry v jednu chvíli nezávisle
- Všechny registry jsou identické, s výjimkou registru x0
- Některé registry jsou předurčené specifickému použití, není to však HW zajištěno





Register File



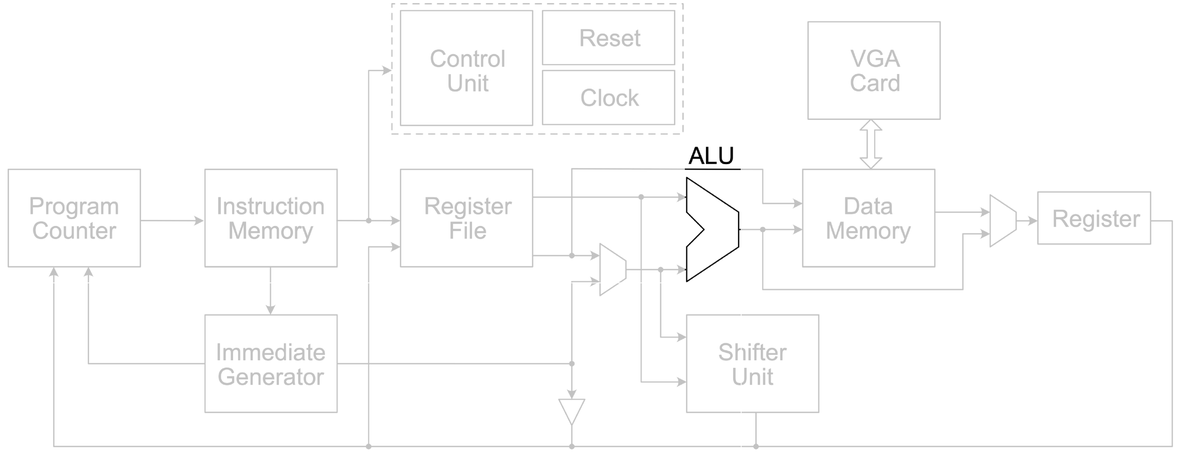
Výpis registrů (RV32I):

x0	Zero
x1	Return address
x2	Stack pointer
x3	Ground pointer
x4	Thread Pointer
x5	Temporary 0
x6	Temporary 1
x7	Temporary 2
x8	Save 0
x9	Save 1
x10	Function argument / return value 0
x11	Function argument / return value 1
x12	Function argument / return value 2
x13	Function argument / return value 3
x14	Function argument / return value 4
x15	Function argument / return value 5

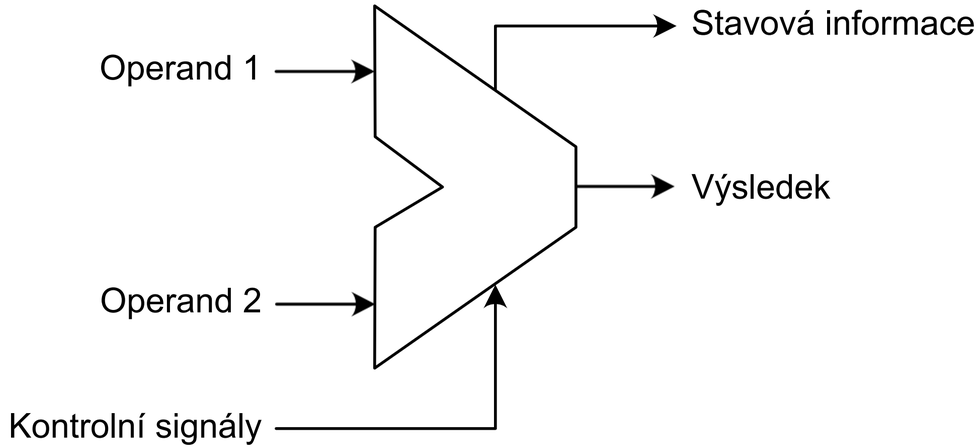
x16	Function argument / return value 6
x17	Function argument / return value 7
x18	Save 2
x19	Save 3
x20	Save 4
x21	Save 5
x22	Save 6
x23	Save 7
x24	Save 8
x25	Save 9
x26	Save 10
x27	Save 11
x28	Temporary 3
x29	Temporary 4
x30	Temporary 5
x31	Temporary 6

ALU

Aritmeticko - logická jednotka



- Řeší aritmeticko-logické problémy a podmínky (branch)
- Řeší podmínky (například pro podmíněné skoky)

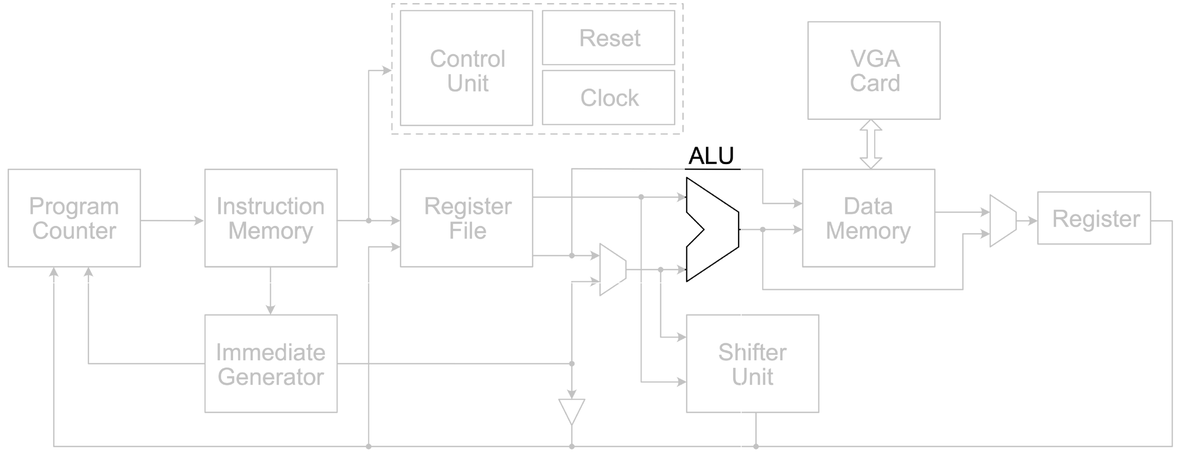


Aritmeticko-logické instrukce

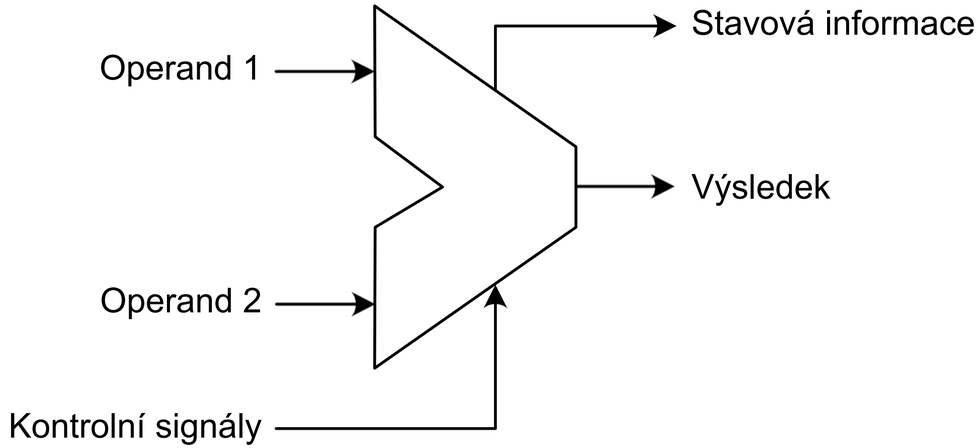
add	Addition
addi	Addition (of immediate value)
sub	Subtraction
sll	Shift left logical
slli	Shift left logical (by immediate value)
xor	XOR
xori	XOR (with immediate value)
srl	Shift right logical
srli	Shift right logical (by immediate value)
sra	Shift right arithmetic
srai	Shift right arithmetic (by immediate value)
or	OR
ori	OR (with immediate value)
and	AND
andi	AND (with immediate value)

ALU

Aritmeticko - logická jednotka



- Řeší aritmeticko-logické problémy a podmínky (branch)
- Řeší podmínky (například pro podmíněné skoky)



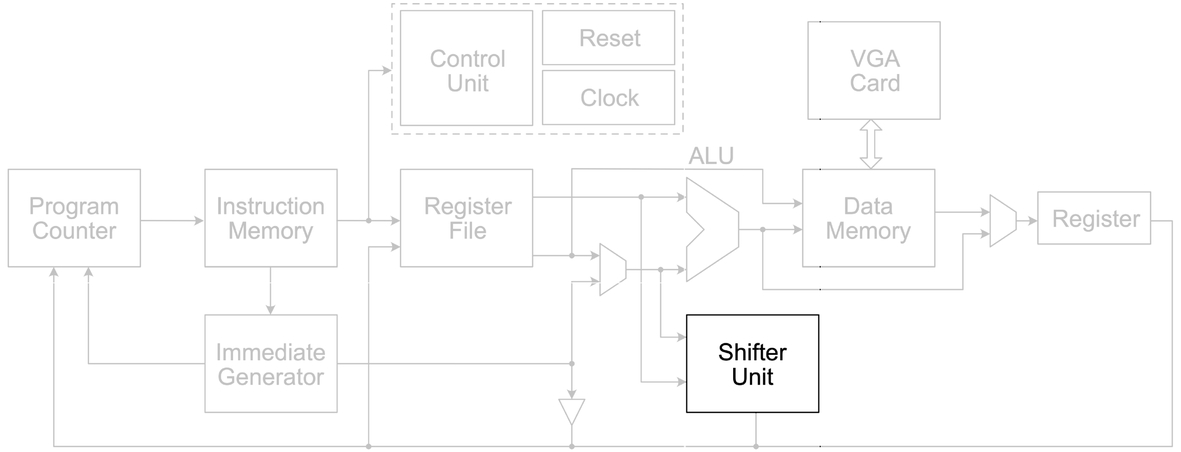
Podmíněné skoky

beq	branch (if) equal
bne	branch (if) not equal
blt	branch (if) less than
bge	branch (if) greater (or) equal
bltu	branch (if) less than unsigned
bgeu	branch (if) greater (or) equal unsigned

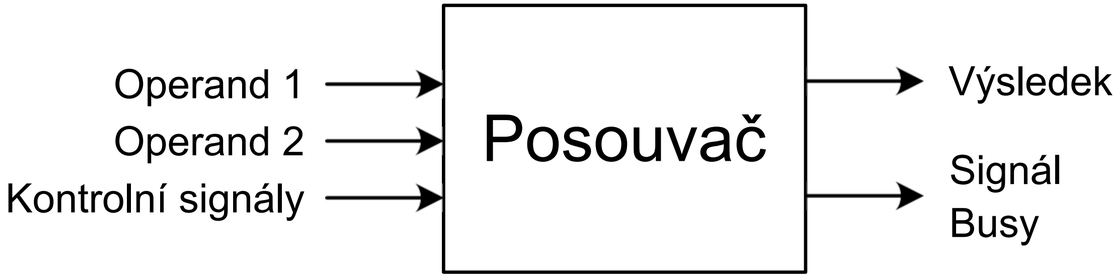
Nastavovací instrukce

slt	Set (if) less then
slti	Set (if) less then (immediate)
sltu	Set (if) less then (unsigned)
sltiu	Set (if) less then (immediate, unsigned)

Shifter Unit



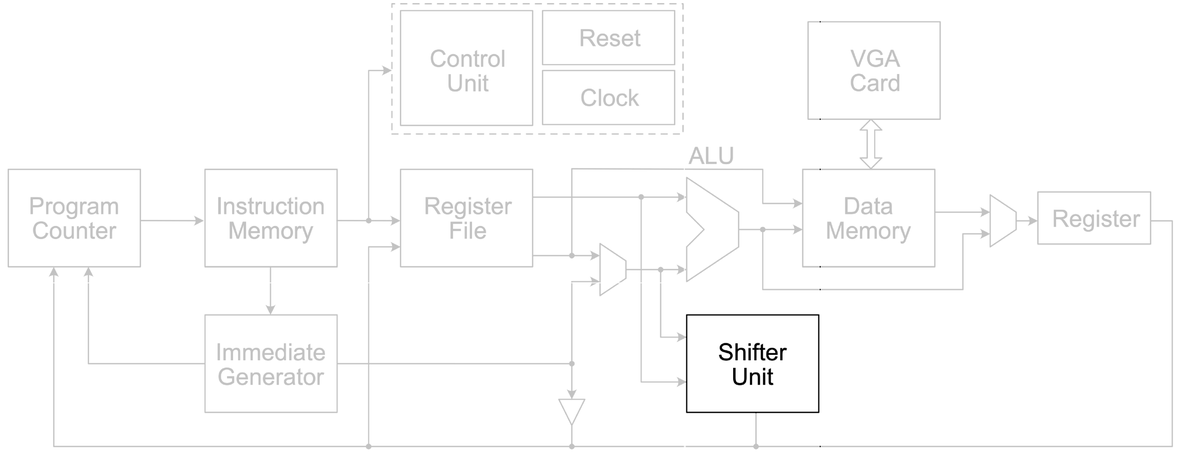
- Posouvá operand o danou hodnotu:
  - Doprava (aritmeticky / logicky)
  - Doleva (pouze logicky)



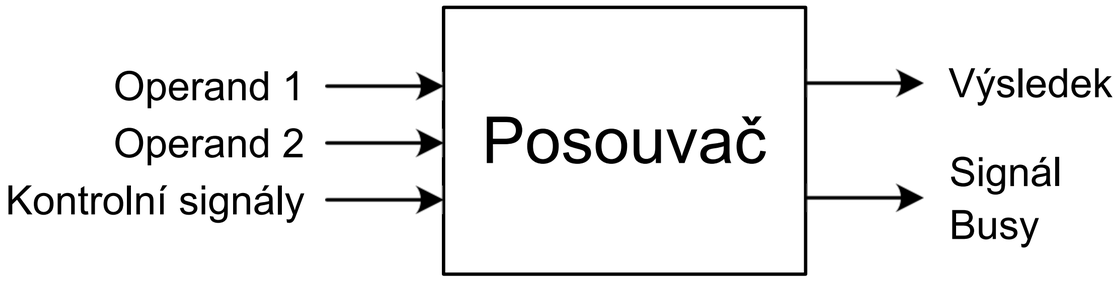
Aritmeticko-logické instrukce

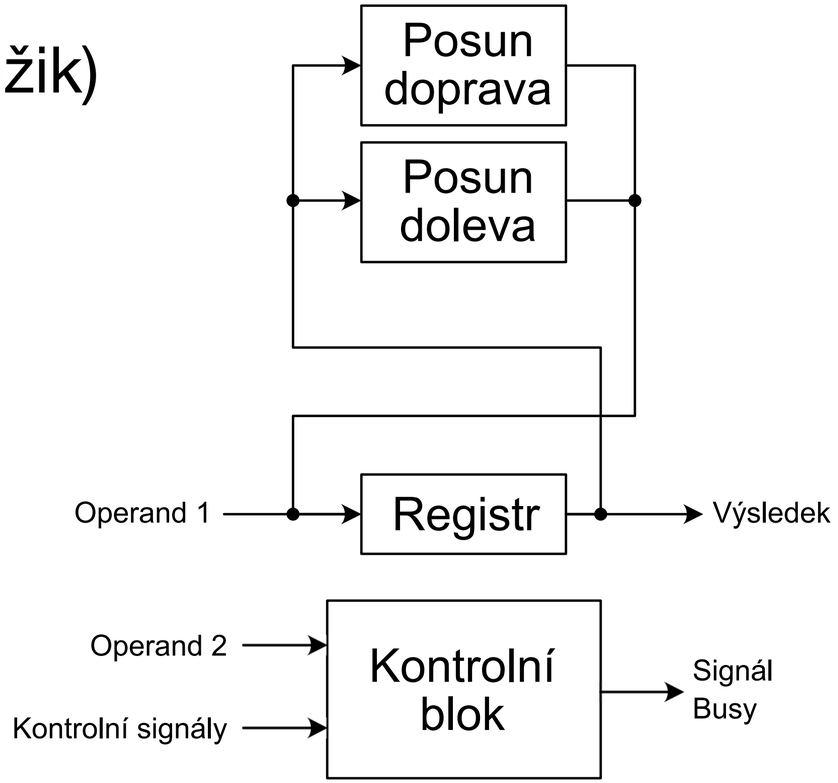
add	Addition
addi	Addition (of immediate value)
sub	Subtraction
sll	Shift left logical
slli	Shift left logical (by immediate value)
xor	XOR
xori	XOR (with immediate value)
srl	Shift right logical
srli	Shift right logical (by immediate value)
sra	Shift right arithmetic
srai	Shift right arithmetic (by immediate value)
or	OR
ori	OR (with immediate value)
and	AND
andi	AND (with immediate value)

Shifter Unit

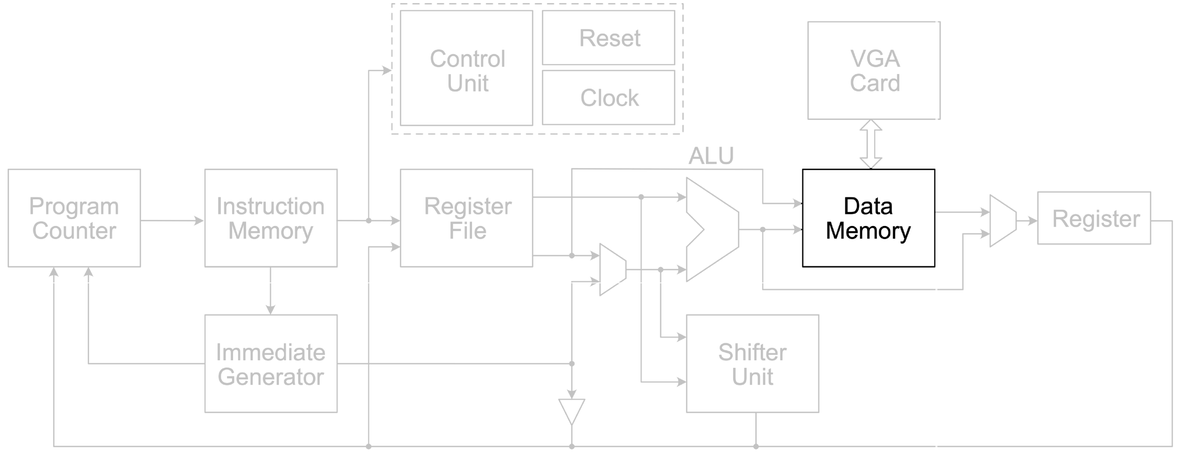


- Dvě možnosti implementace:
  - Paralelní (celá operace se vykoná v jeden okamžik)
  - Postupná (postupně posouváme o 1)

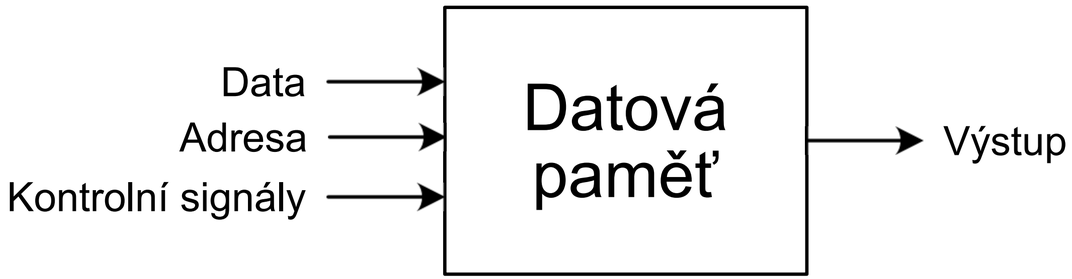




Data Memory



- Ukládá data do paměti (dle sektoru)
- Můžeme pracovat s byty (8 bit), půlslovem (16 bit) a slovem (32 bit)
- Byte steering logic - ukládaný byte můžeme “přesunout” na 3 další pozice

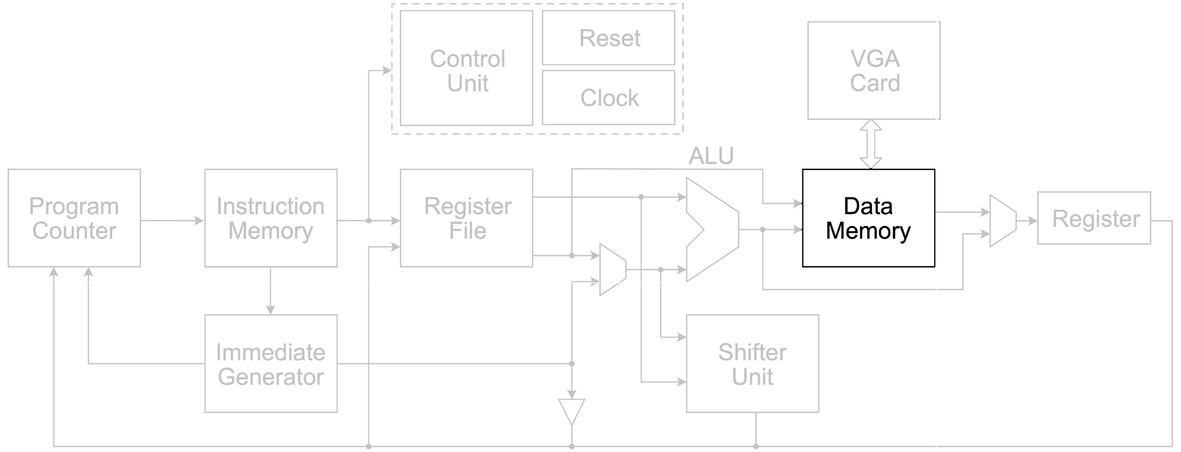


Paměťové instrukce

lb	load byte
lbu	load byte unsigned
lb	load byte
lhu	load half unsigned
lw	load word
sb	store byte
sh	store half
sw	store word

Data Memory

- Speciální rozsahy - RAM, V-RAM, SR



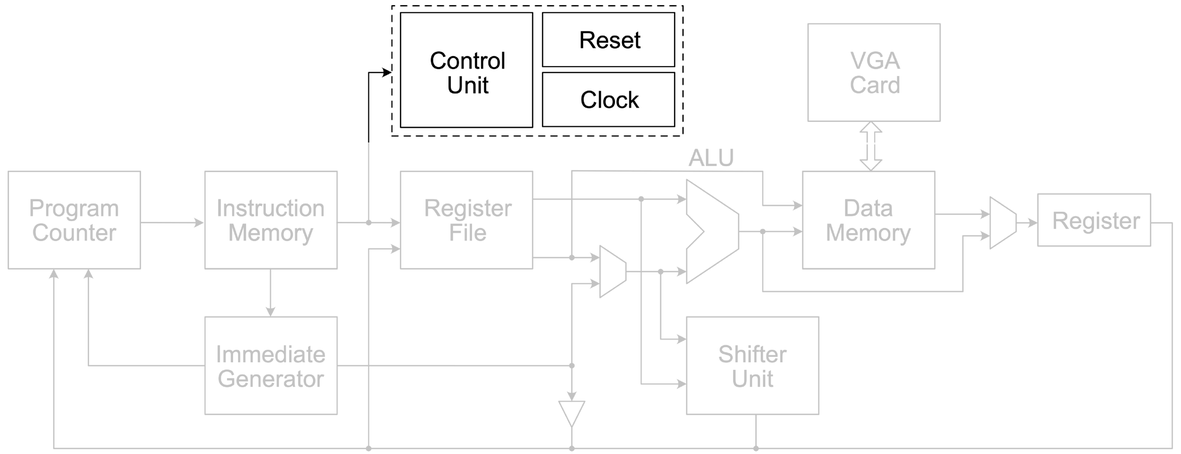
Speciální registry

Adresa [hex]	Vstup/Výstup	Popis
0x80000000	Výstup	Displej
0x80000001	Výstup	VGA_Enable
0x80000004	Vstup	Port A
0x80000005	Vstup	Port B
0x80000006	Výstup	Port C
0x80000007	Výstup	Port D

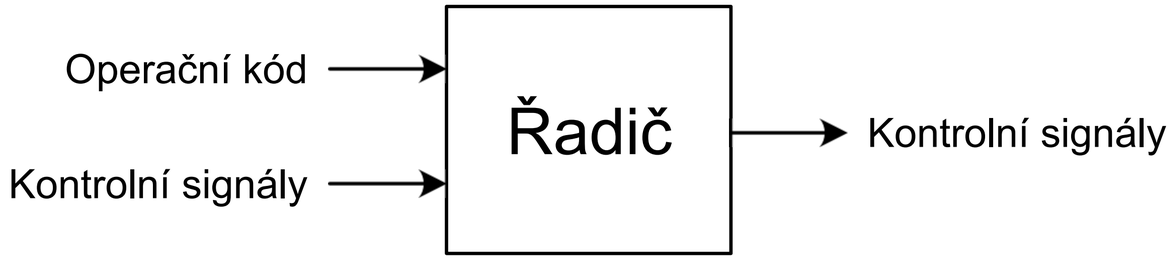
Rozsahy paměti

Adresový rozsah	Typ
0x00000000 ... 0x0001FFFF	RAM
0x00020000 ... 0x3FFFFFFF	Rezervovaný prostor
0x40000000 ... 0x400007FF	V-RAM
0x40000800 ... 0x7FFFFFFF	Rezervovaný prostor
0x80000000 … 0x8000000F	Speciální Registry
0x80000010 ... 0xFFFFFFFF	Rezervovaný prostor

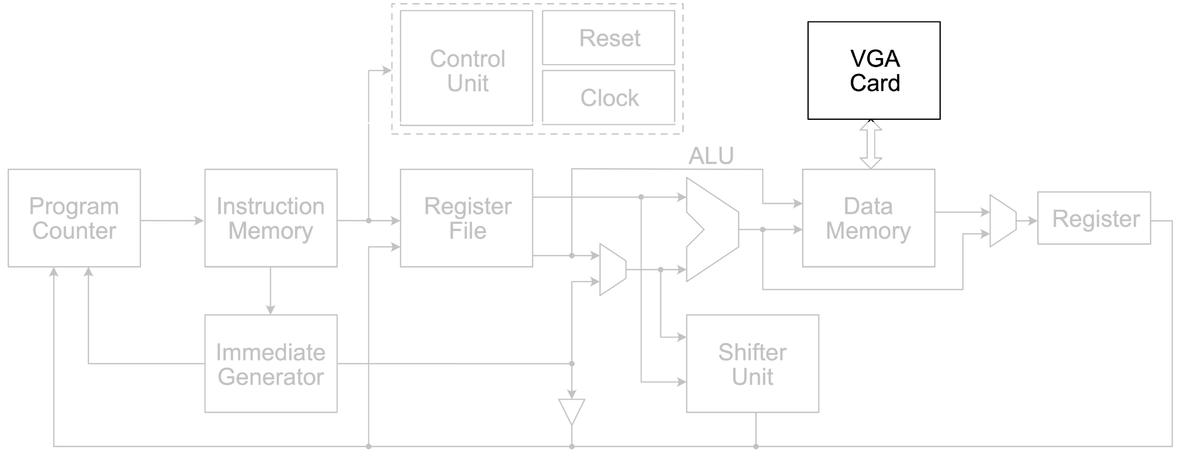
Control Unit



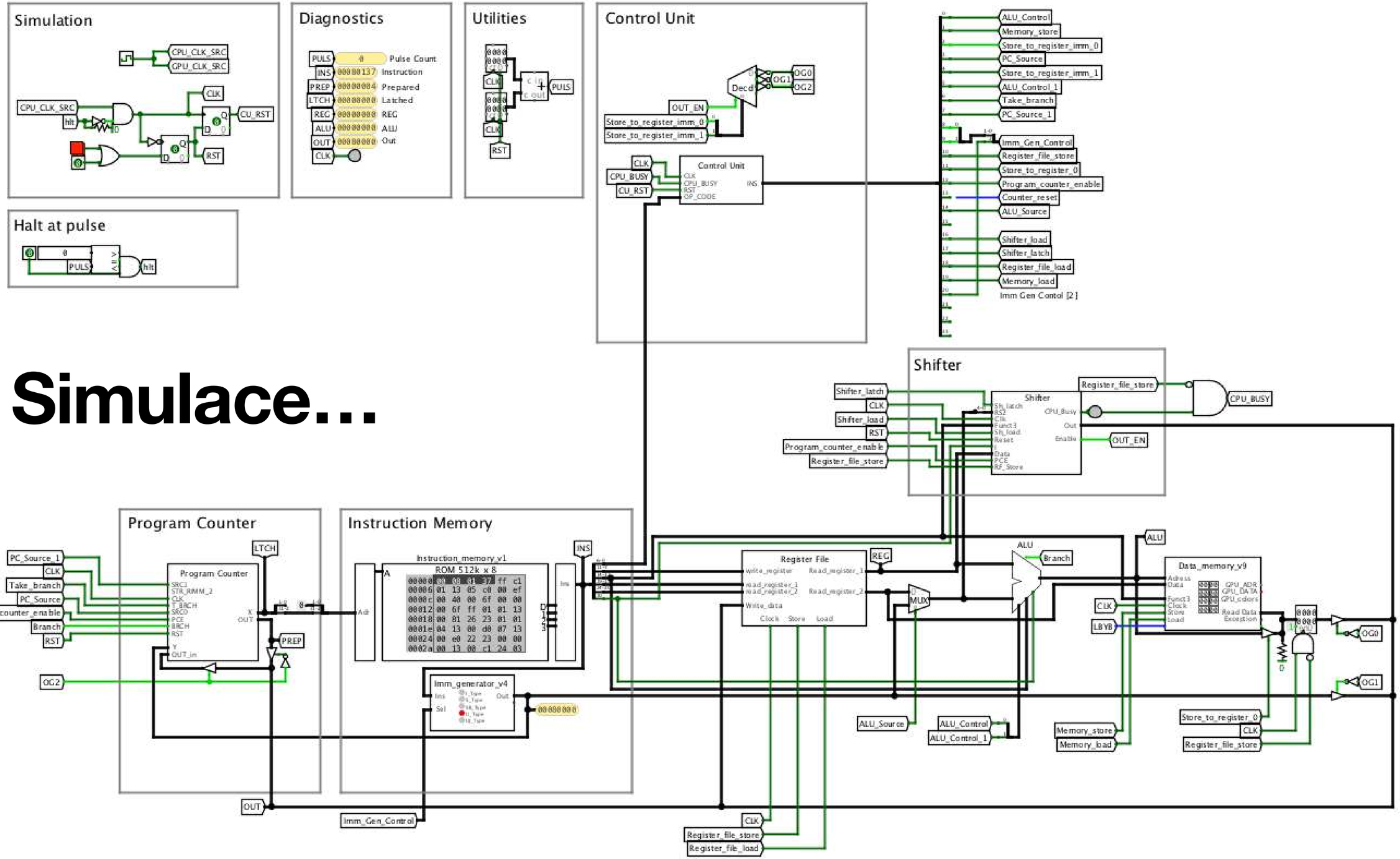
- Generuje kontrolní signály v závislosti na op-kódu z každé instrukce
- Mikroinstrukce - Risc-V procesory by je technicky neměly potřebovat



VGA Card

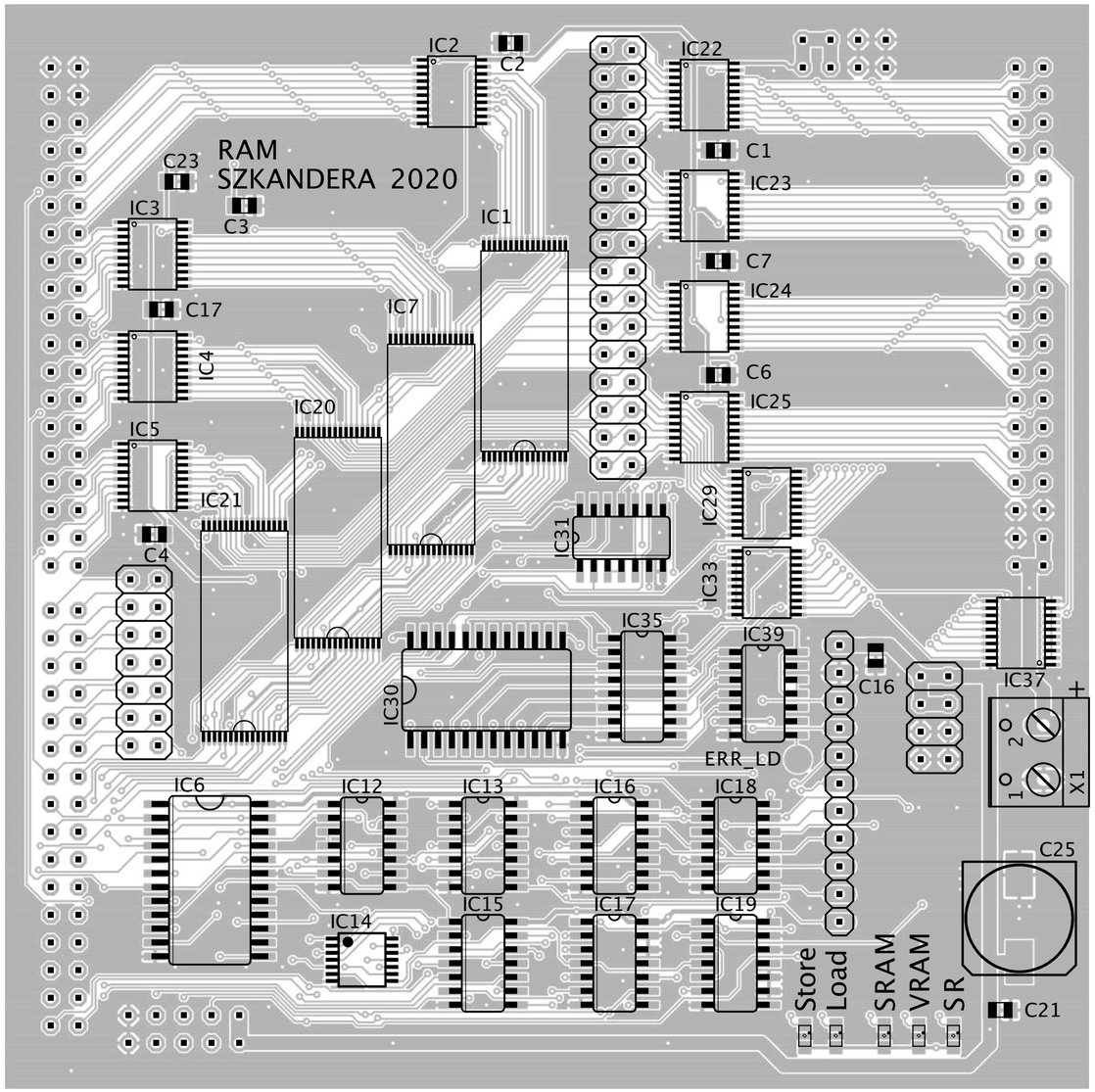


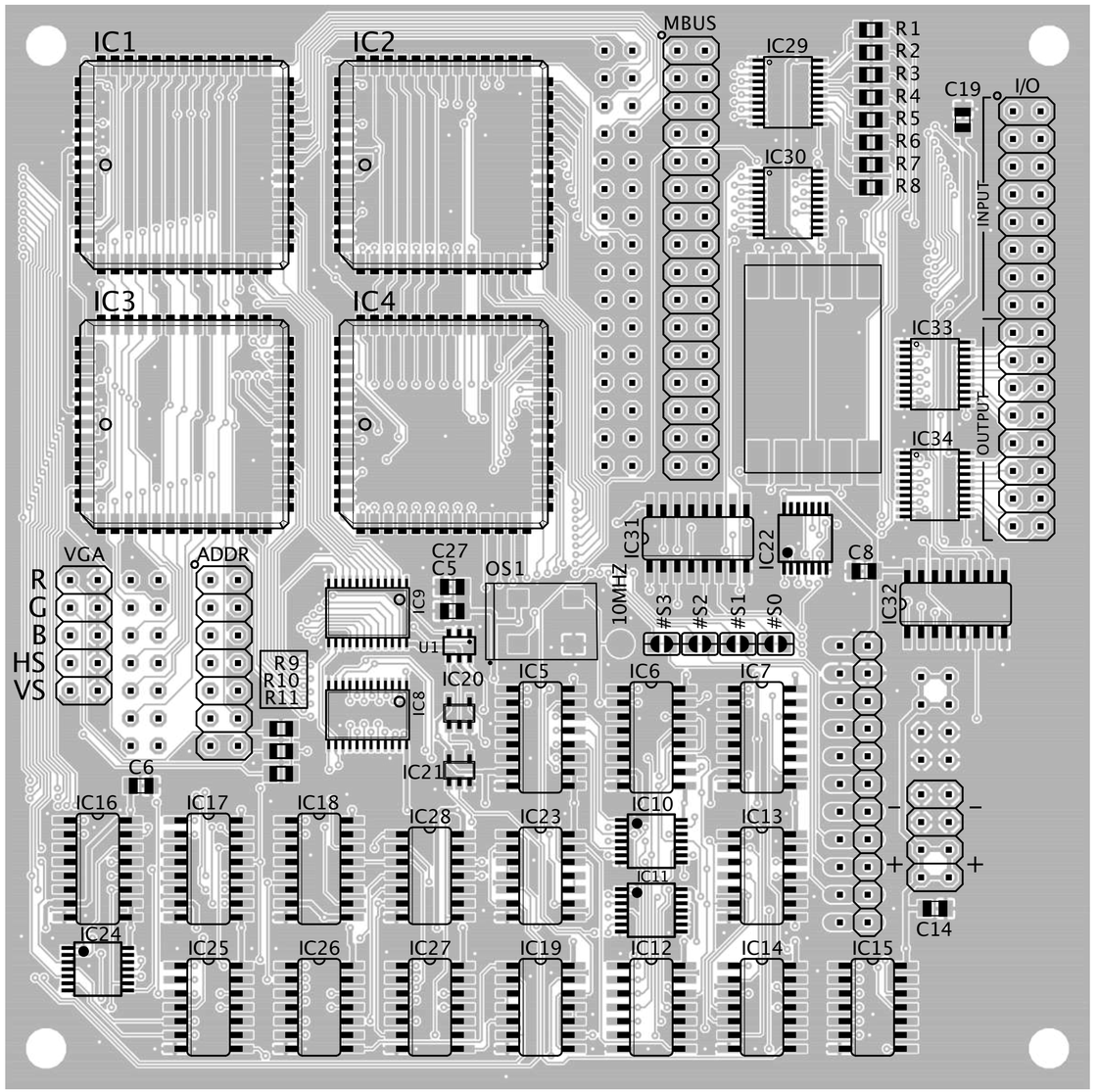
- Generuje VGA signál
- Používá dual-port paměti pro display-buffer
- 200 x 150px, pouze černobílé


Simulation
CPU_CLK_SRC
GPU_CLK_SRC
CLK
hlt
CU_RST
RST
Diagnostics
PULS
Pulse Count
INS
00000137
Instruction
PREP
00000004
Prepared
LTCH
00000000
Latched
REG
00000000
REG
ALU
00000000
ALU
OUT
00000000
Out
CLK
Utilities
PULS
RST
Control Unit
Decd
OG0
OG1
OG2
OUT_EN
Store_to_register_imm_0
Store_to_register_imm_1
CLK
CPU_BUSY
CU_RST
Control Unit
INS
ALU_Control
Memory_store
Store_to_register_imm_0
PC_Source
Store_to_register_imm_1
ALU_Control_1
Take_branch
PC_Source_1
Imm_Gen_Control
Register_file_store
Store_to_register_0
Program_counter_enable
Counter_reset
ALU_Source
Shifter_load
Shifter_latch
Register_file_load
Memory_load
Imm Gen Contol [2]
Halt at pulse
PULS
hlt
Shifter
Shifter_latch
CLK
Shifter_load
RST
Program_counter_enable
Register_file_store
Register_file_store
CPU_BUSY
OUT_EN
Program Counter
LTCH
PC_Source_1
CLK
Take_branch
PC_Source
Program_counter_enable
Branch
RST
PREP
OG2
OUT
Instruction Memory
Instruction_memory_v1
ROM 512k x 8
INS
Imm_generator_v4
00000000
Imm_Gen_Control
Register File
REG
MUX
ALU
Branch
ALU_Source
ALU_Control
ALU_Control_1
CLK
Register_file_store
Register_file_load
Data_memory_v9
CLK
LBYB
Memory_store
Memory_load
Store_to_register_0
CLK
Register_file_store
OG0
OG1

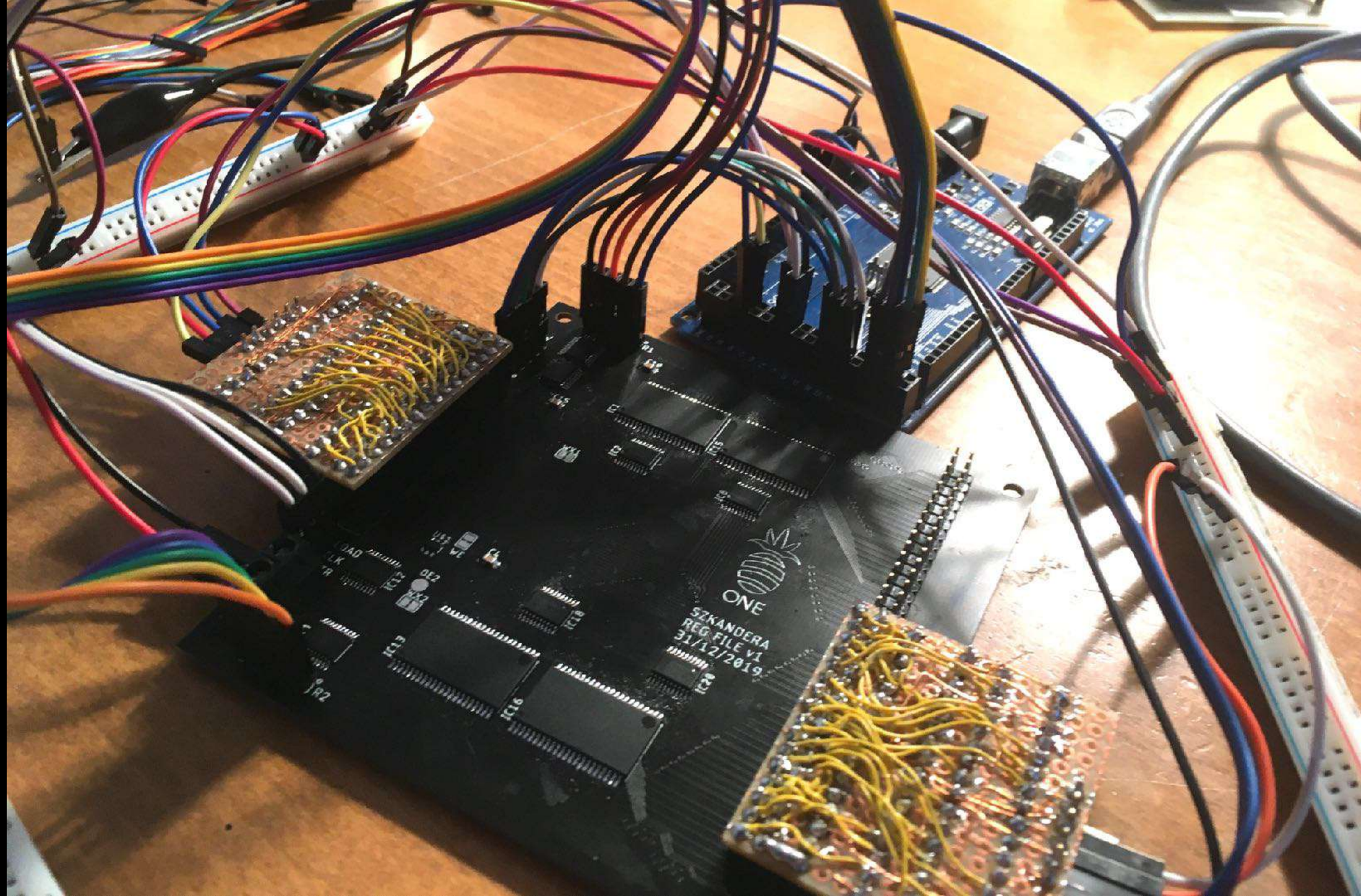
Simulace…

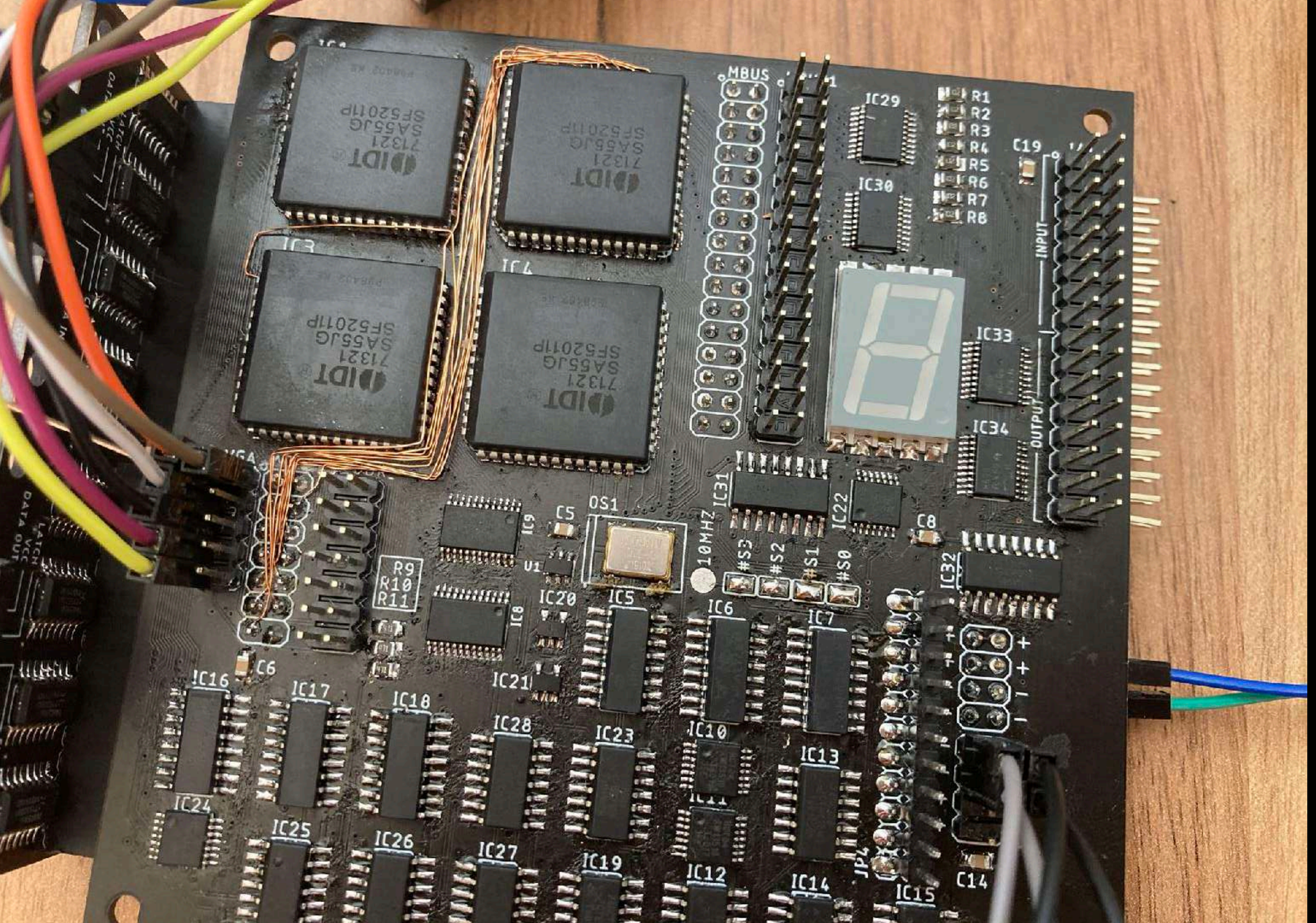
Návrh PCB


RAM
SZKANDERA 2020
IC1
IC2
IC3
IC4
IC5
IC6
IC7
IC12
IC13
IC14
IC15
IC16
IC17
IC18
IC19
IC20
IC21
IC22
IC23
IC24
IC25
IC29
IC30
IC31
IC33
IC35
IC37
IC39
C1
C2
C3
C4
C6
C7
C16
C17
C21
C23
C25
X1
ERR_LD
Store
Load
SRAM
VRAM
SR


IC1
IC2
IC3
IC4
MBUS
IC29
IC30
IC31
IC32
IC33
IC34
R1
R2
R3
R4
R5
R6
R7
R8
C19
I/O
INPUT
OUTPUT
VGA
ADDR
R
G
B
HS
VS
C27
C5
OS1
10MHZ
#S3
#S2
#S1
#S0
IC22
C8
IC9
U1
R9
R10
R11
IC8
IC20
IC21
C6
IC5
IC6
IC7
IC16
IC17
IC18
IC28
IC23
IC10
IC11
IC13
IC24
IC25
IC26
IC27
IC19
IC12
IC14
IC15
C14

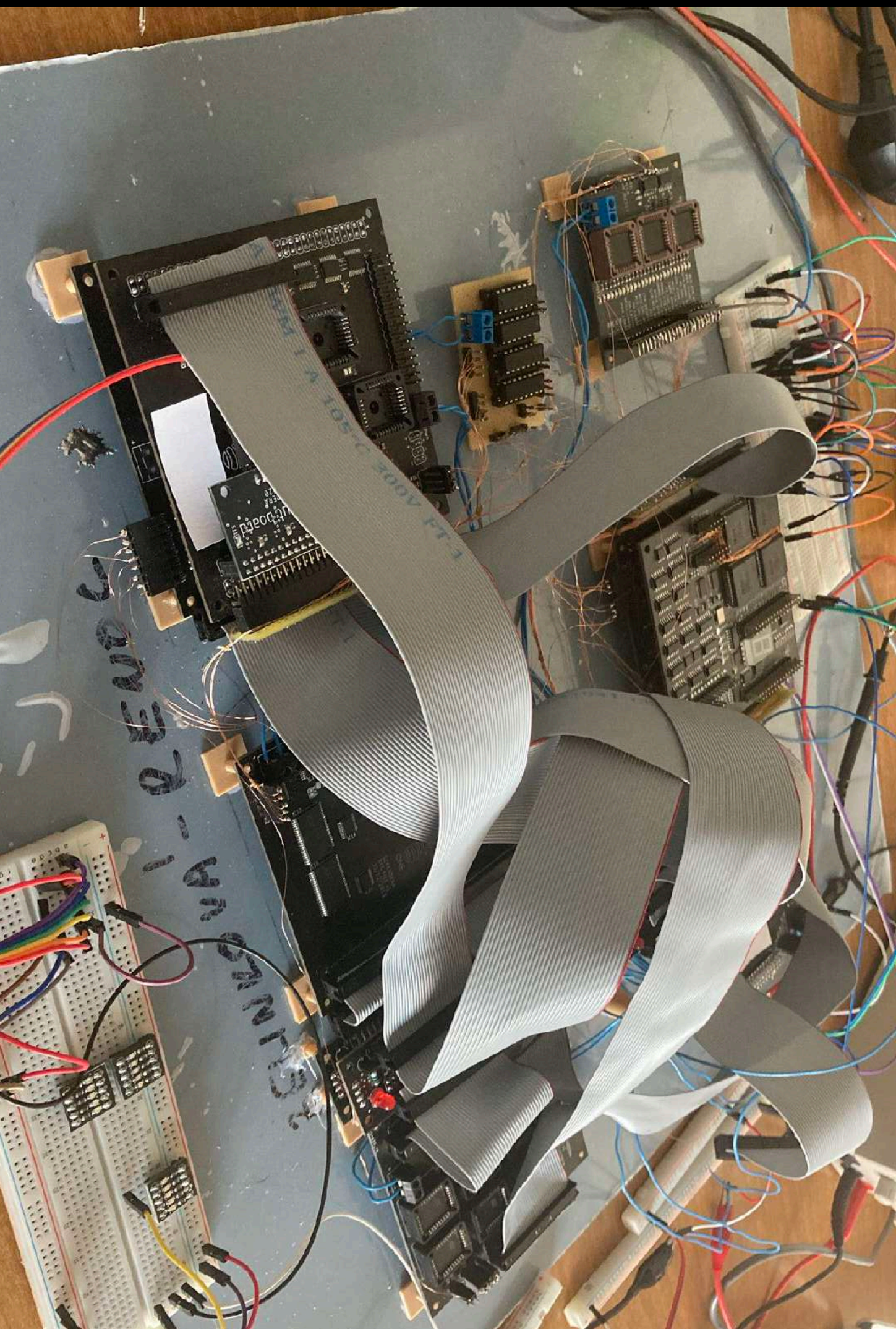
Testování

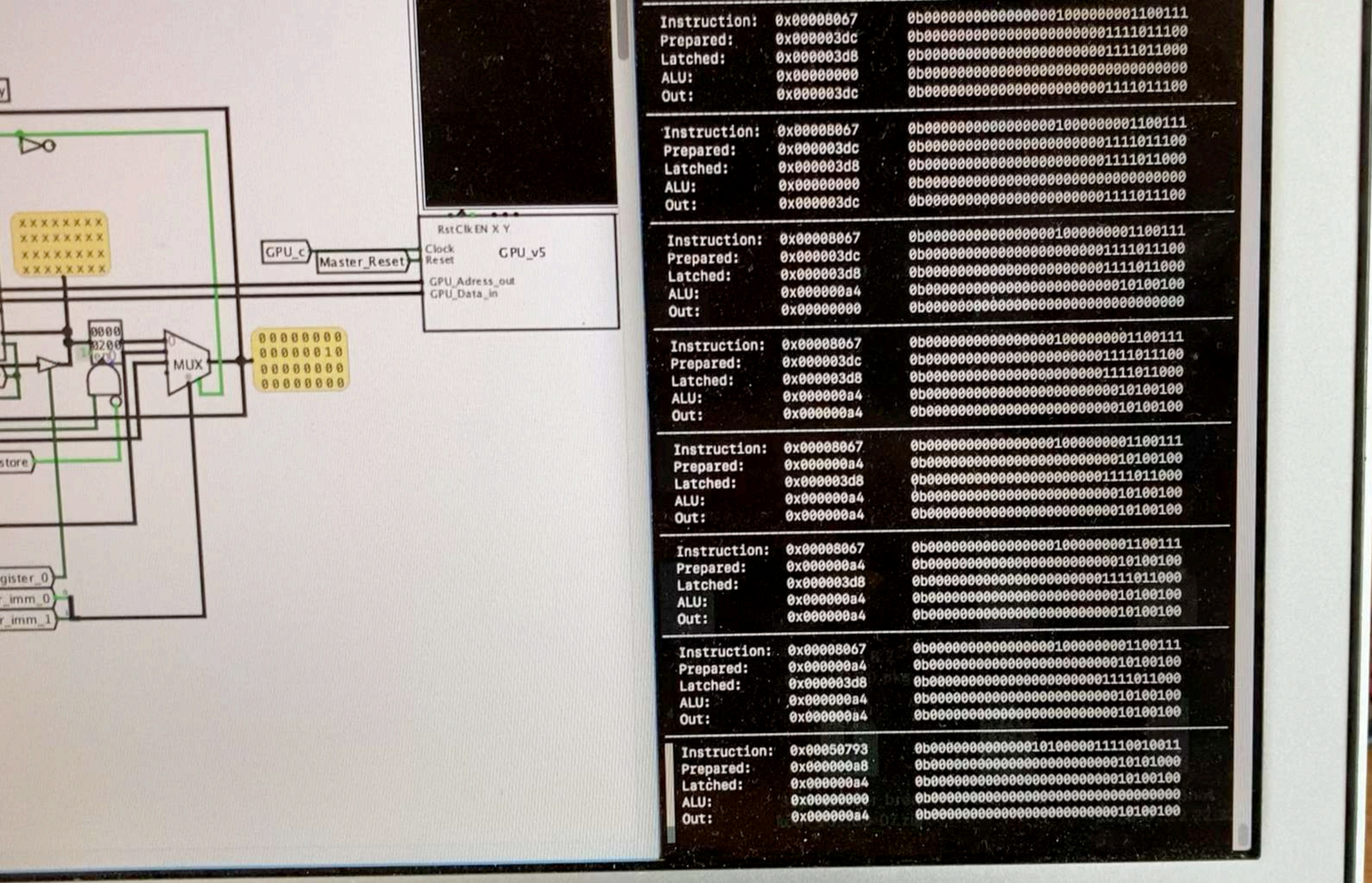



MBUS
IC29
IC30
R1
R2
R3
R4
R5
R6
R7
R8
C19
INPUT
OUTPUT
IC33
IC34
IC31
IC22
C8
OS1
10MHZ
#S3
#S2
#S1
#S0
IC32
IC20
IC5
IC6
IC7
C5
C6
IC16
IC17
IC18
IC21
IC28
IC23
IC10
IC13
IC24
IC25
IC26
IC27
IC19
IC12
IC14
IC15
C14
JP2
R9
R10
R11
IC8
IC9
V1

Oprava chyb

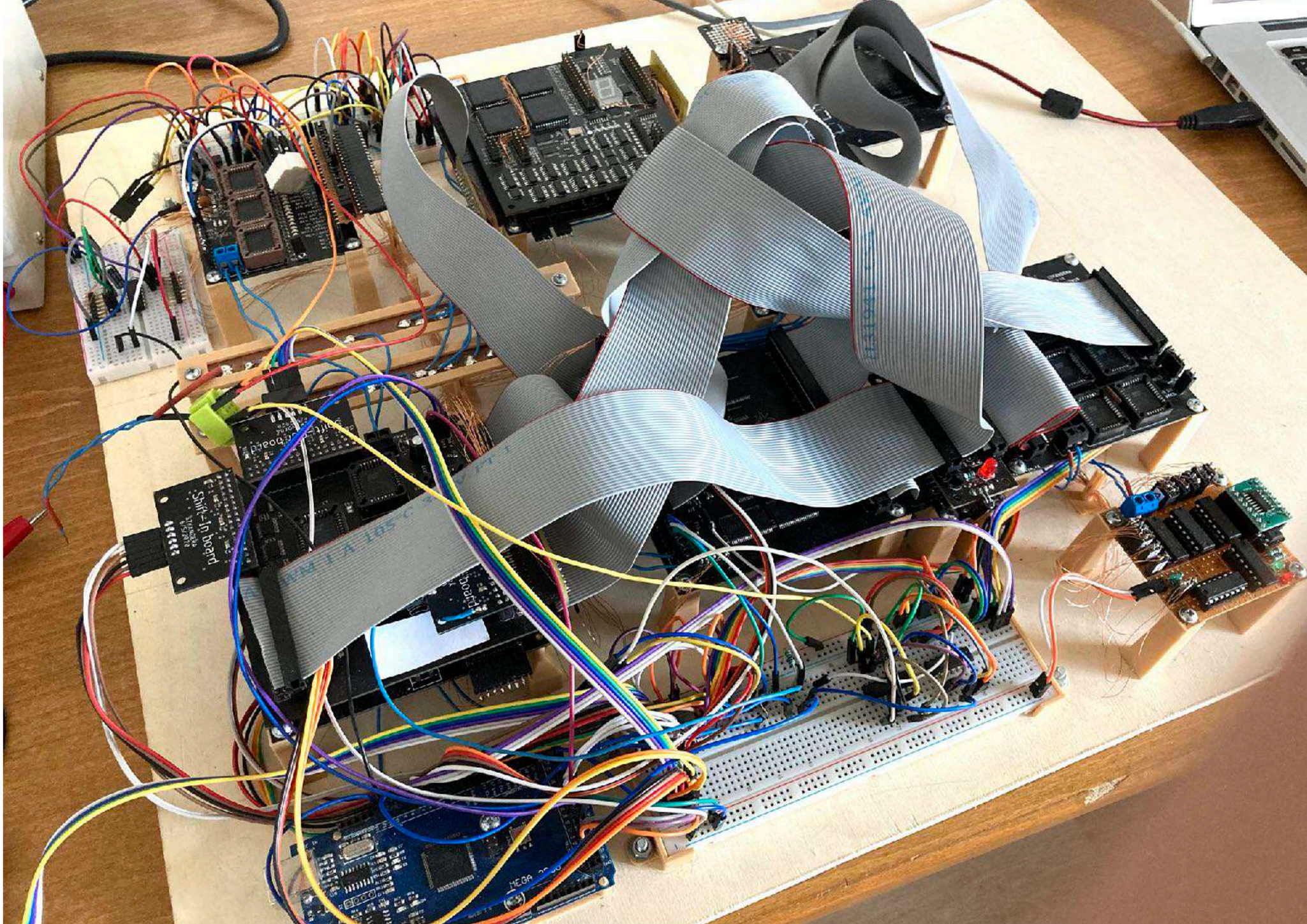
Rozložení PCB





```
Instruction:  0x00008067   0b00000000000000001000000001100111
Prepared:     0x000003dc   0b00000000000000000000001111011100
Latched:      0x000003d8   0b00000000000000000000001111011000
ALU:          0x00000000   0b00000000000000000000000000000000
Out:          0x000003dc   0b00000000000000000000001111011100

Instruction:  0x00008067   0b00000000000000001000000001100111
Prepared:     0x000003dc   0b00000000000000000000001111011100
Latched:      0x000003d8   0b00000000000000000000001111011000
ALU:          0x00000000   0b00000000000000000000000000000000
Out:          0x000003dc   0b00000000000000000000001111011100

Instruction:  0x00008067   0b00000000000000001000000001100111
Prepared:     0x000003dc   0b00000000000000000000001111011100
Latched:      0x000003d8   0b00000000000000000000001111011000
ALU:          0x000000a4   0b00000000000000000000000010100100
Out:          0x00000000   0b00000000000000000000000000000000

Instruction:  0x00008067   0b00000000000000001000000001100111
Prepared:     0x000003dc   0b00000000000000000000001111011100
Latched:      0x000003d8   0b00000000000000000000001111011000
ALU:          0x000000a4   0b00000000000000000000000010100100
Out:          0x000000a4   0b00000000000000000000000010100100

Instruction:  0x00008067   0b00000000000000001000000001100111
Prepared:     0x000000a4   0b00000000000000000000000010100100
Latched:      0x000003d8   0b00000000000000000000001111011000
ALU:          0x000000a4   0b00000000000000000000000010100100
Out:          0x000000a4   0b00000000000000000000000010100100

Instruction:  0x00008067   0b00000000000000001000000001100111
Prepared:     0x000000a4   0b00000000000000000000000010100100
Latched:      0x000003d8   0b00000000000000000000001111011000
ALU:          0x000000a4   0b00000000000000000000000010100100
Out:          0x000000a4   0b00000000000000000000000010100100

Instruction:  0x00008067   0b00000000000000001000000001100111
Prepared:     0x000000a4   0b00000000000000000000000010100100
Latched:      0x000003d8   0b00000000000000000000001111011000
ALU:          0x000000a4   0b00000000000000000000000010100100
Out:          0x000000a4   0b00000000000000000000000010100100

Instruction:  0x00050793   0b00000000000001010000011110010011
Prepared:     0x000000a8   0b00000000000000000000000010101000
Latched:      0x000000a4   0b00000000000000000000000010100100
ALU:          0x00000000   0b00000000000000000000000000000000
Out:          0x000000a4   0b00000000000000000000000010100100
```

Znovu a lépe


Shift-in board

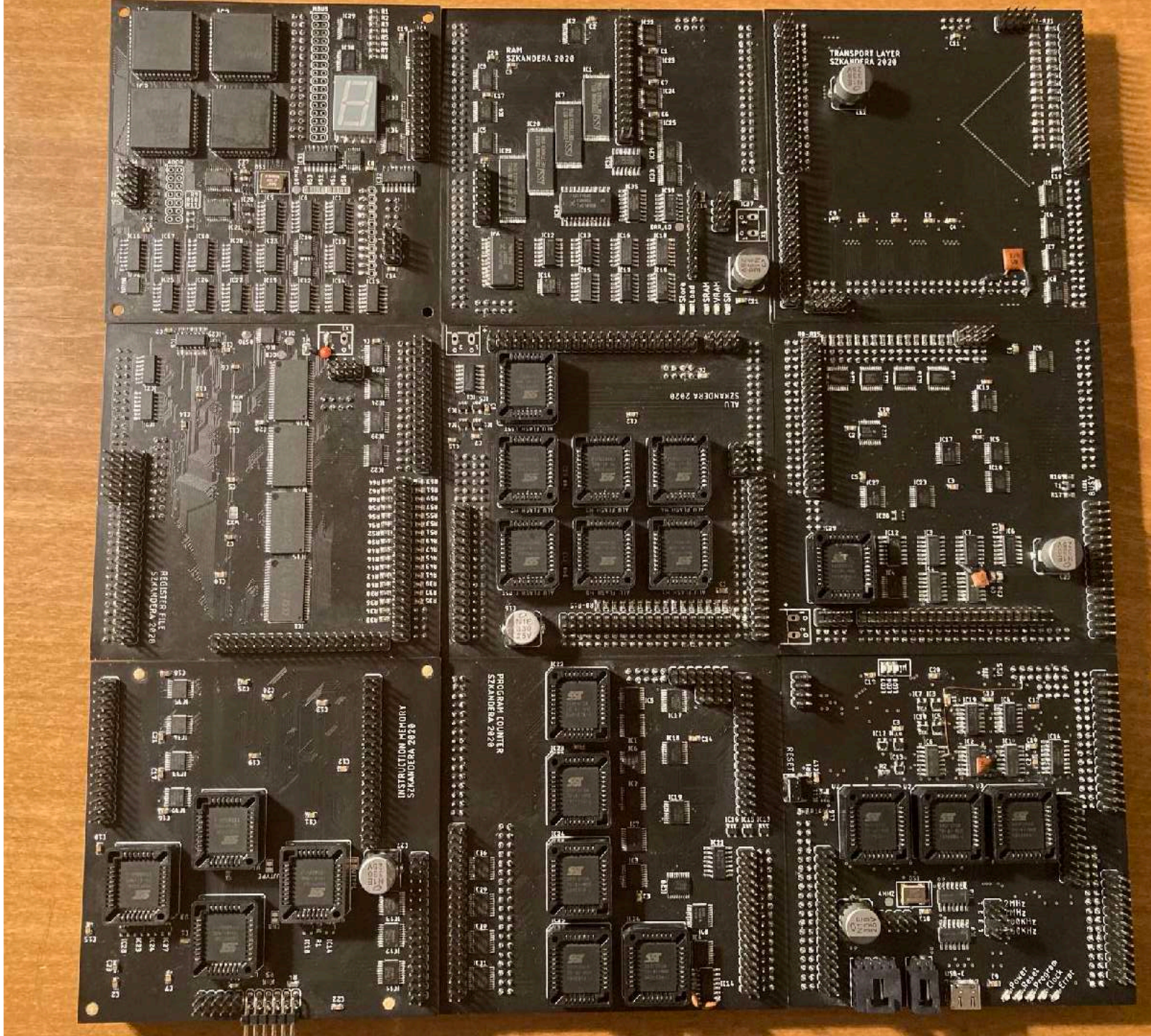

PINEAPPLE ONE OPTIMIZED??
PINEAPPLE ONE OPTIMIZED??
PINEAPPLE ONE OPTIMIZED??
PINEAPPLE ONE OPTIMIZED??
PINEAPPLE ONE OPTIMIZED??
PINEAPPLE ONE OPTIMIZED??
PINEAPPLE ONE OPTIMIZED??
PINEAPPLE ONE OPTIMIZED??
PINEAPPLE ONE OPTIMIZED??
PINEAPPLE ONE OPTIMIZED??
PINEAPPLE ONE OPTIMIZED??
PINEAPPLE ONE OPTIMIZED??
PINEAPPLE ONE OPTIMIZE

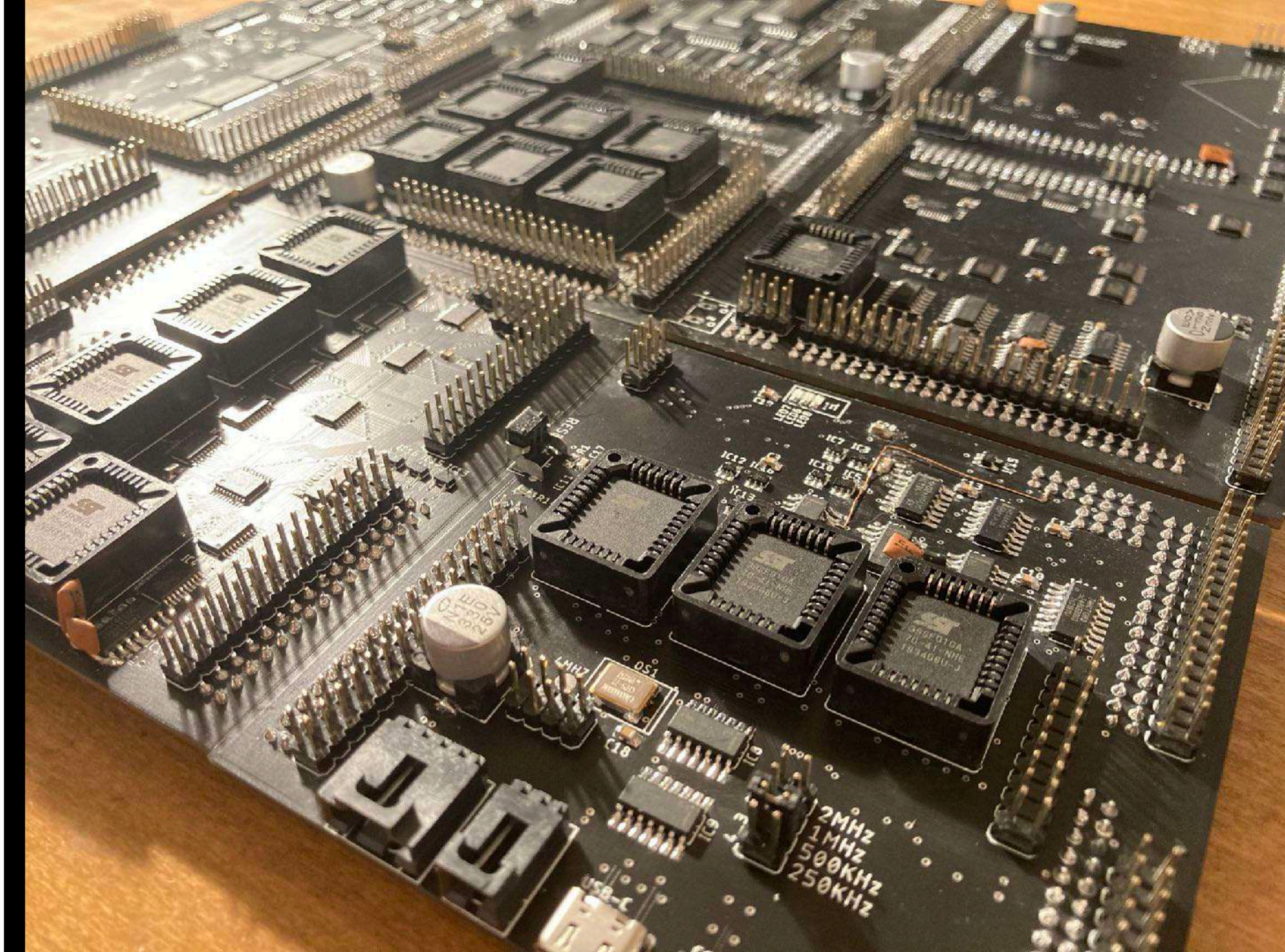
Why ...

Redesign

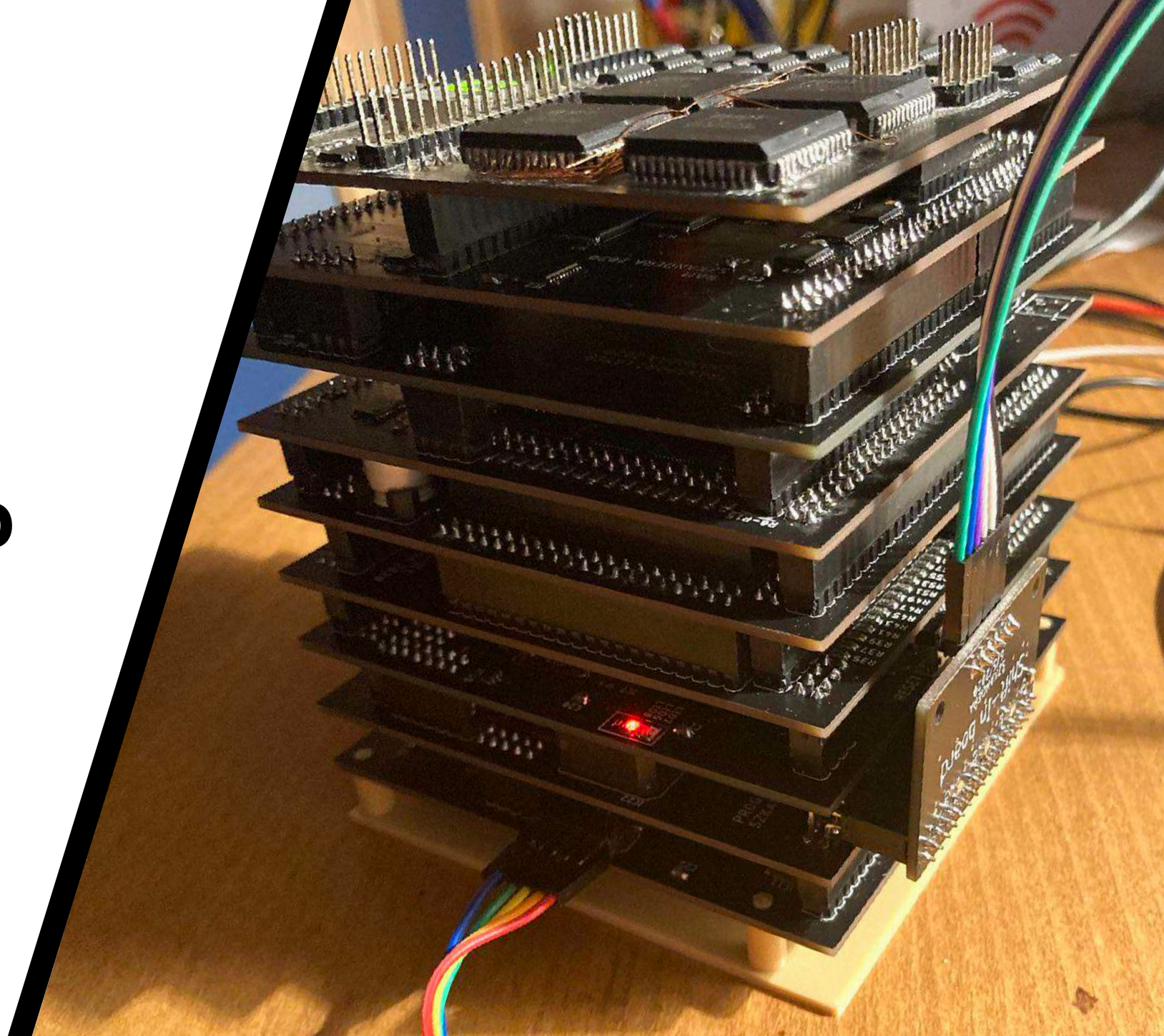
Redesign

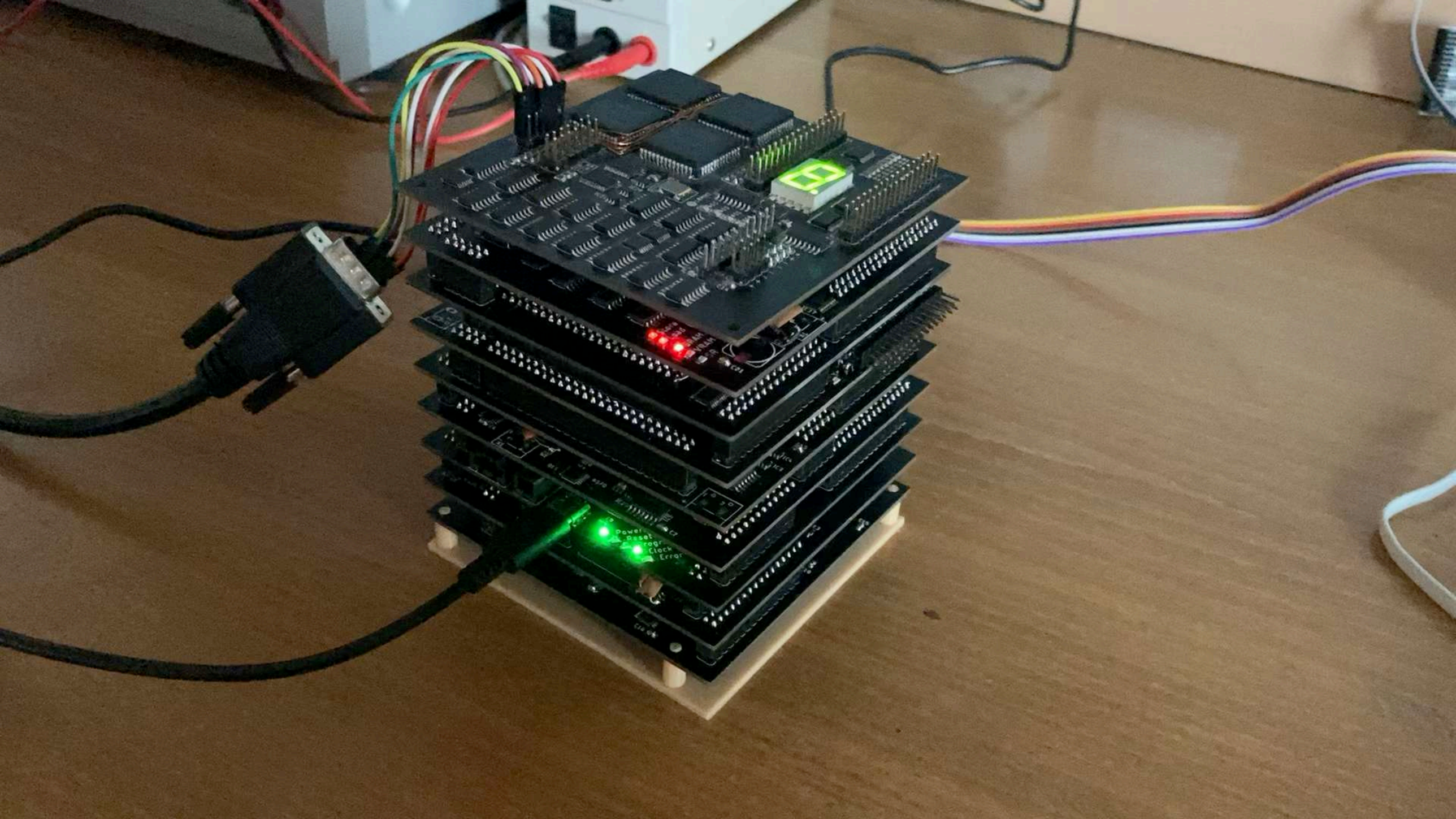
VGA Card	Program Memory	Transport Layer
Register File	ALU	Shifter
Instruction Memory	Program Counter	Control Unit

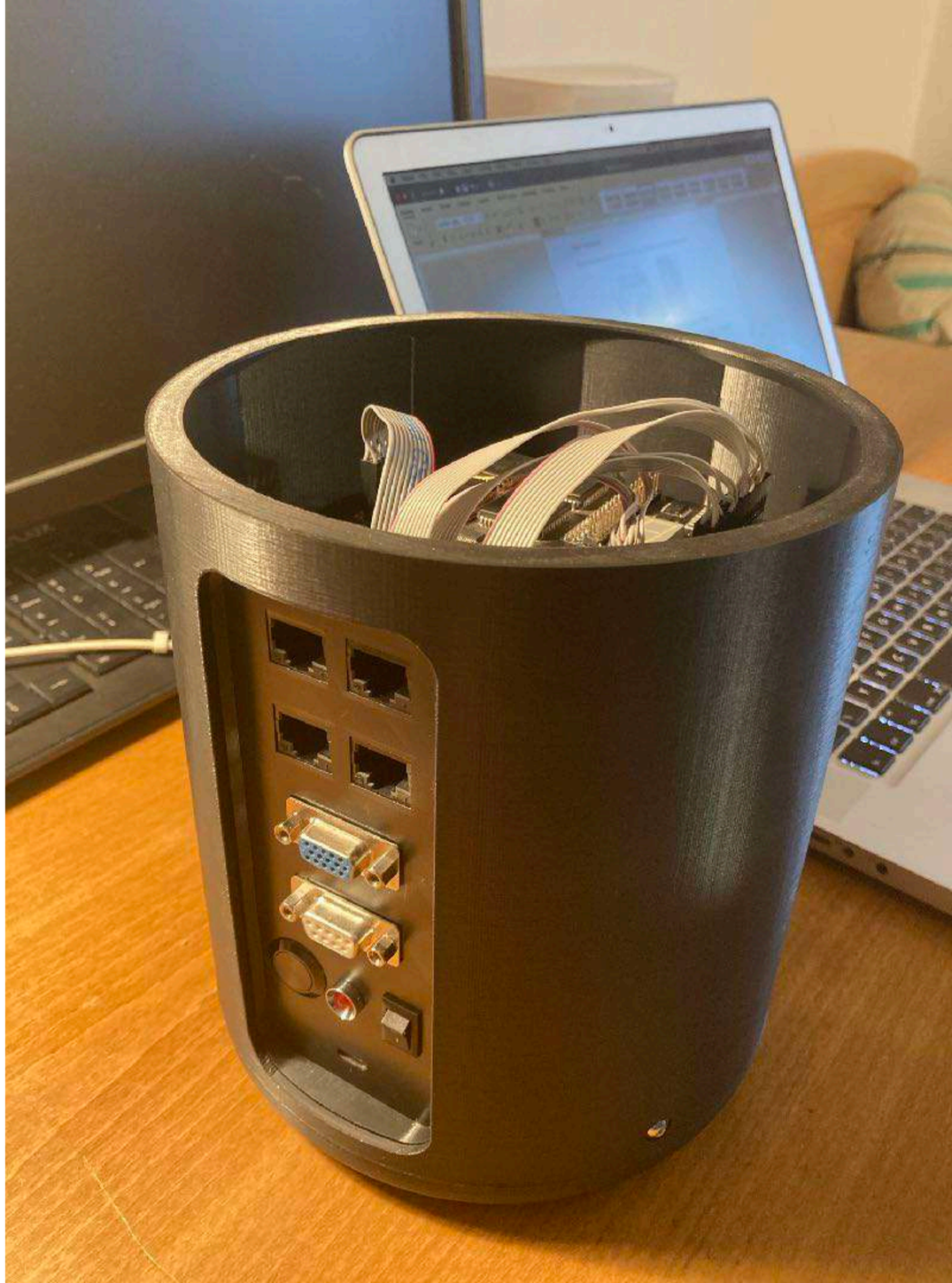



2MHz
1MHz
500KHz
250KHz
USB-C

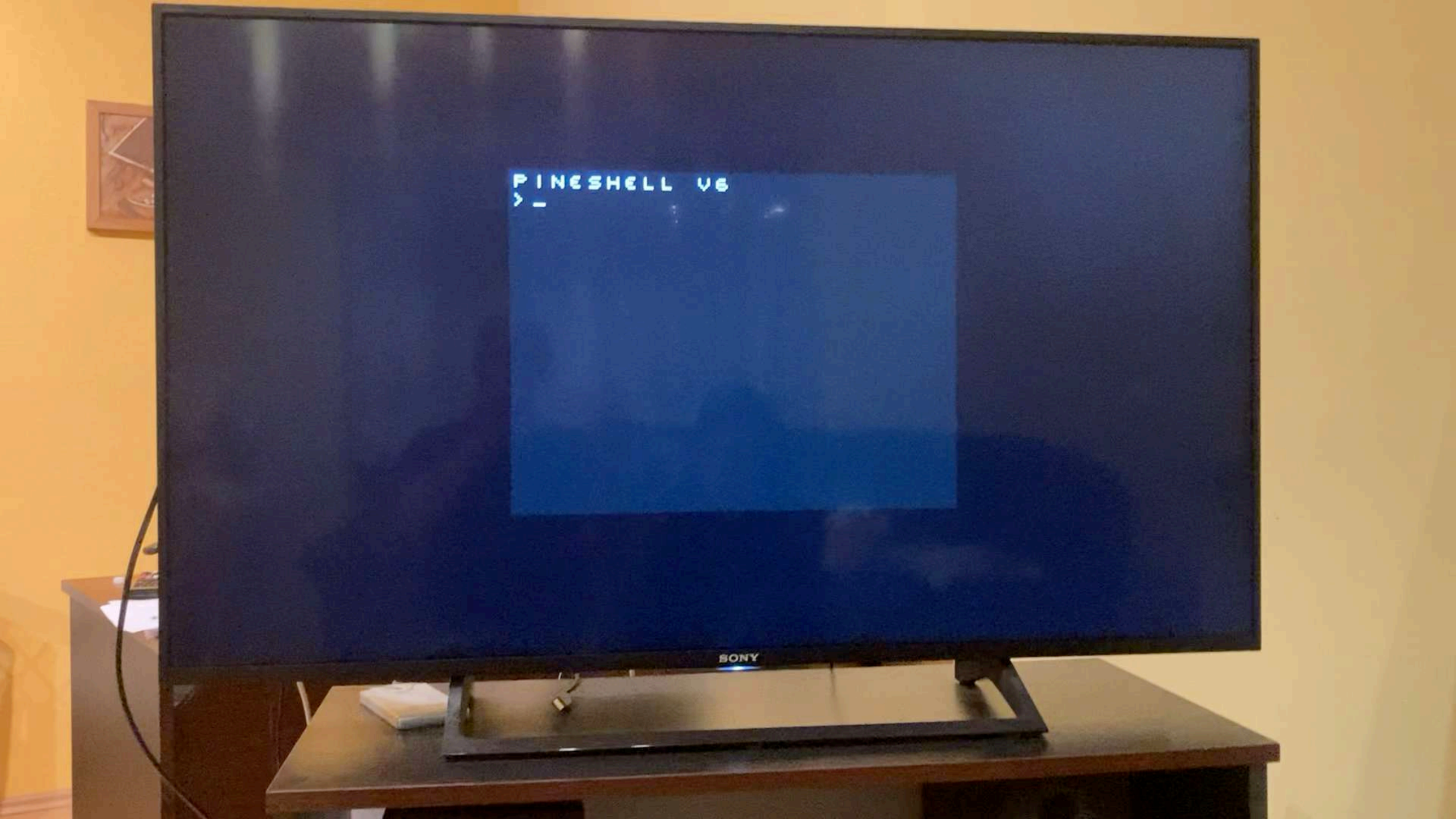
Finální stack-up



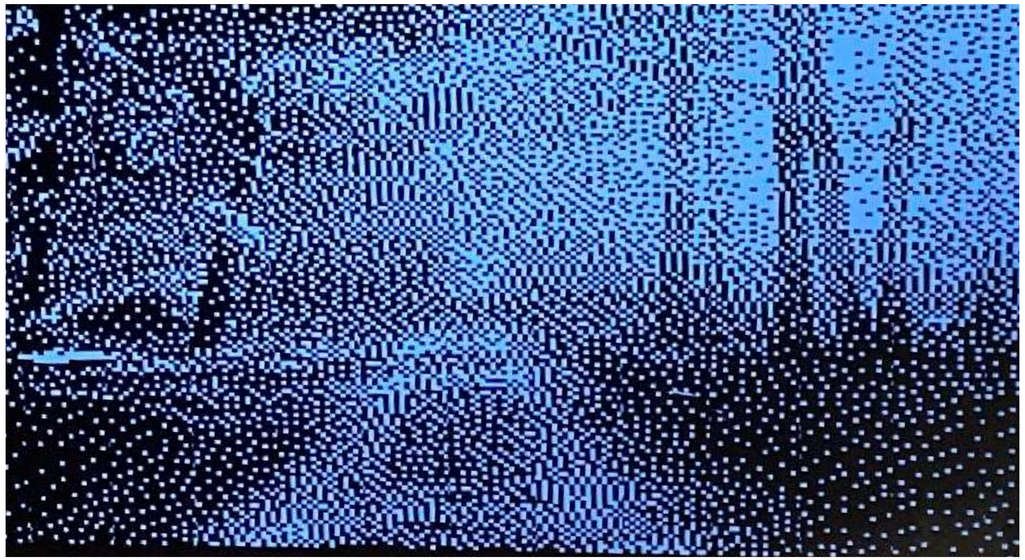

Power
Reset
Clock
Error



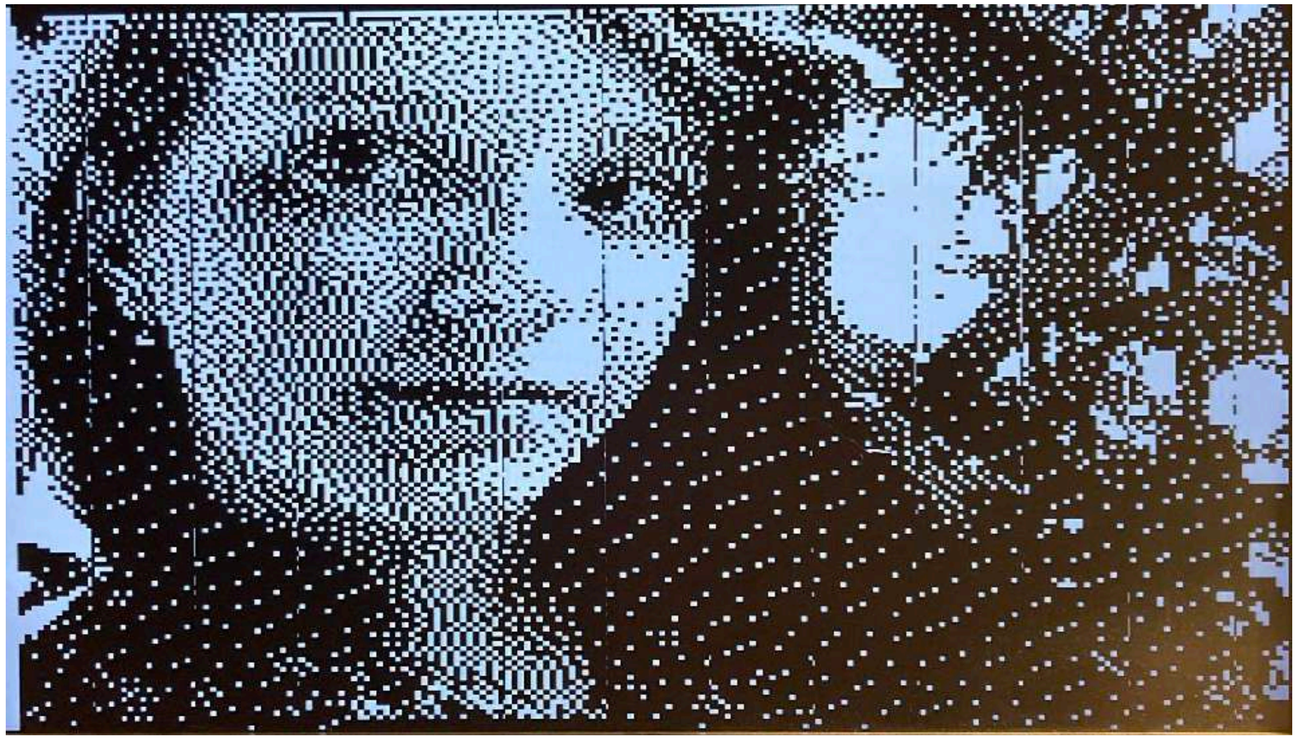
Poslední detaily…


PINESHELL V6
>_
SONY

I malé rozlišení zvládne …



Dithering



Děkuji za pozornost