IEEE EDUCON 2024
IEEE Global Engineering Education Conference

Hardware-Oriented Microprocessor Simulator

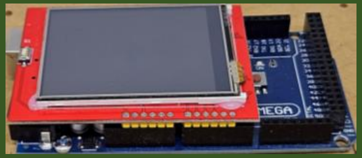
Panayotis (Panos) Papazoglou
Associate Professor

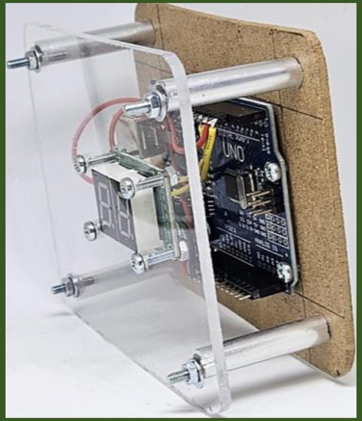
National and Kapodistrian University of Athens
dept. of Digital Arts and Cinema

An open-source project supported by

HELLENIC REPUBLIC
National and Kapodistrian University of Athens

ARCHIMEDES
CENTER FOR INNOVATION AND ENTREPRENEURSHIP





At a Glance

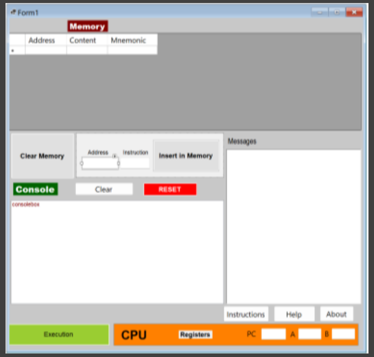
- Novel full-working Hardware-Oriented Microprocessor Simulator (HOMS)
- Educational kit for simulating microprocessors
- Based on the Arduino platform
- Supports custom educational scenarios
- Available as open-source project

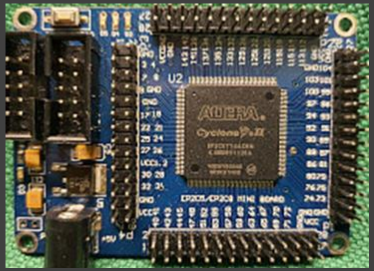
What happens today ?

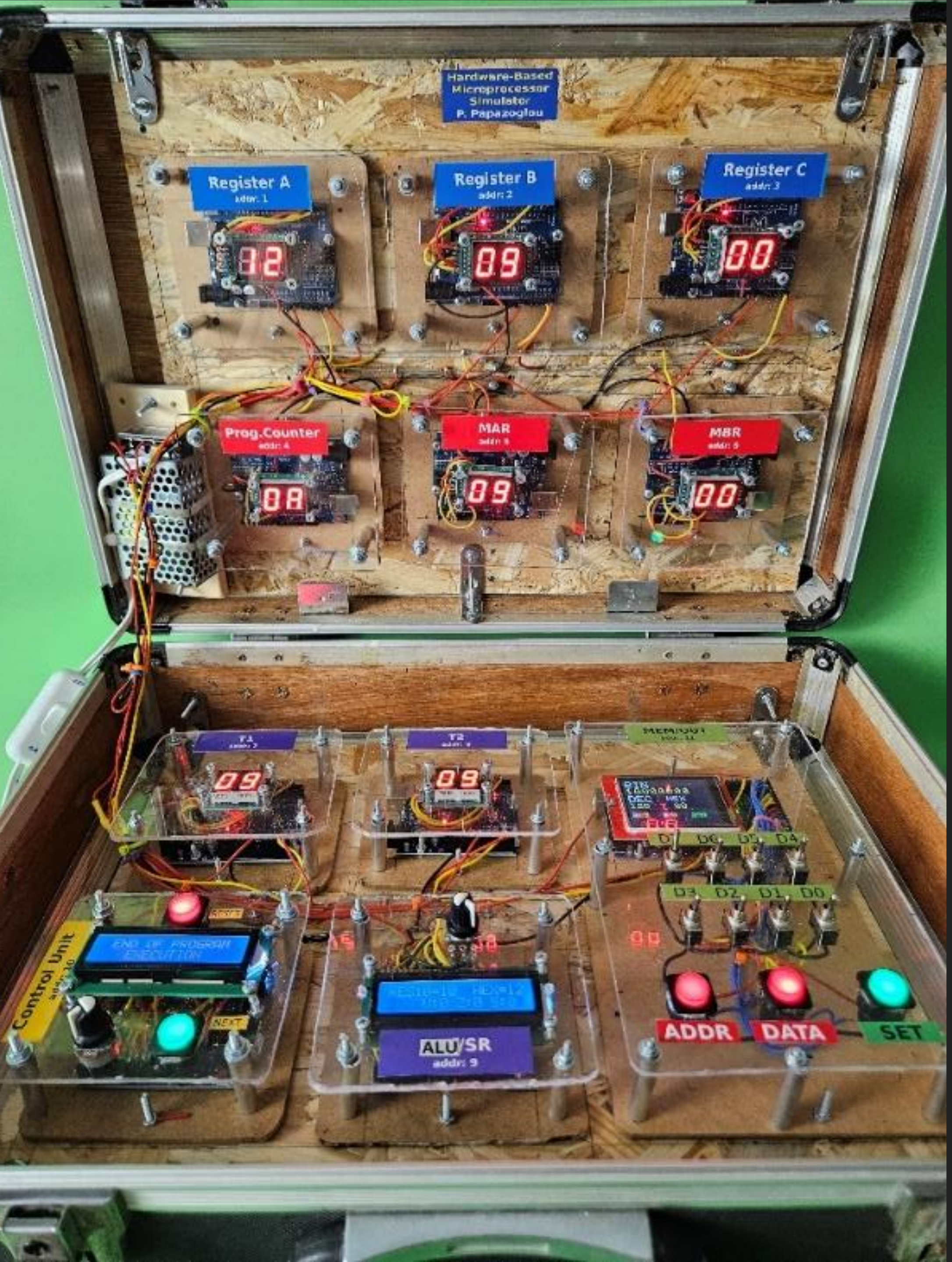
- **Simulator** tools are used in higher education for studying microprocessor architecture and programming

- **Hardware boards** like MPF-I (70s, 80s) are not used anymore [no internal point of view]

- **FPGA technology** (hardware and programming) is very complicated for the students [no internal point of view]



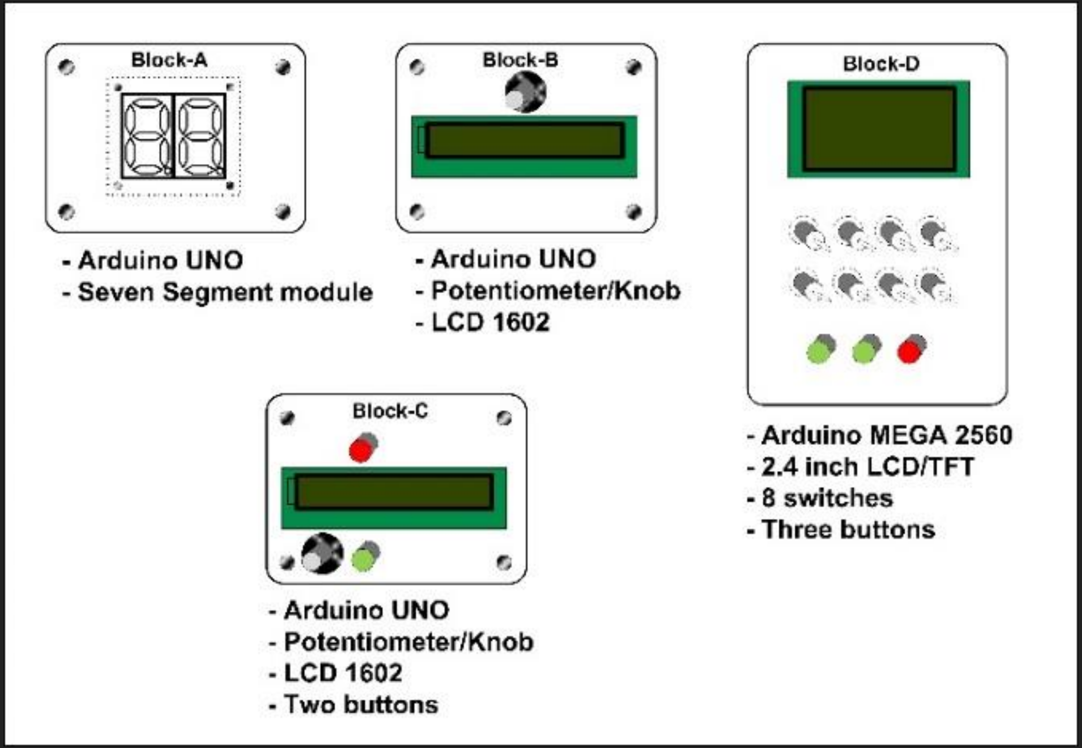




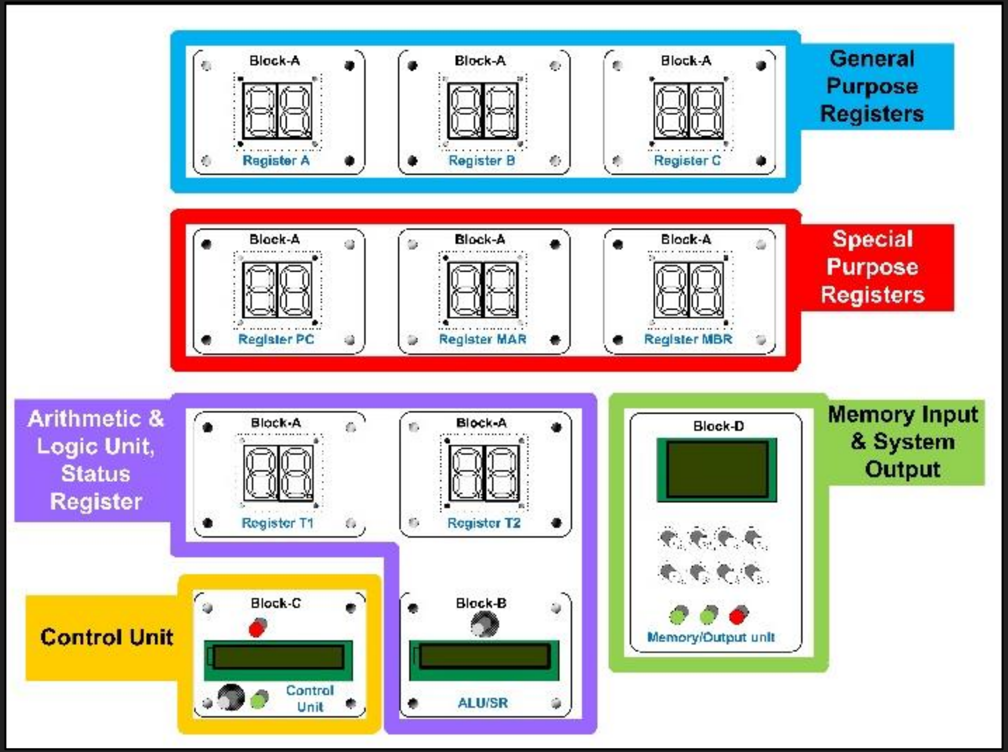
Tool basic features

- Customizable architecture
- Block reusability (building the simulator from identical blocks).
- Programmable functionality. Each block behavior is determined by the embedded software and represents a real microprocessor internal unit.
- Assembly instructions development
- Educational scenarios
- Easy reproduction
- Open-source hardware
- Mobility – Autonomous operation

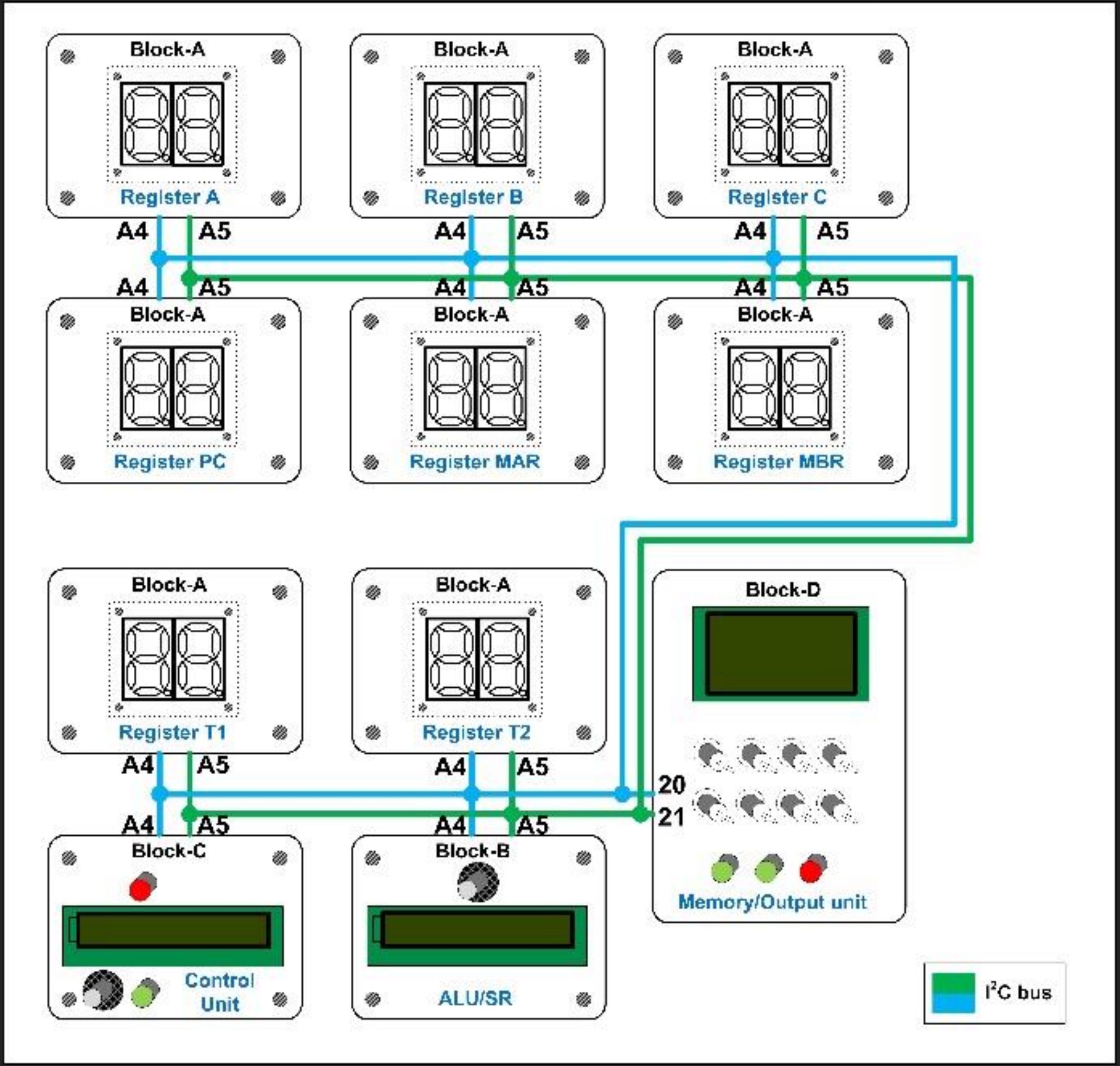
Types of blocks



Block organization



Block connectivity



Limitations

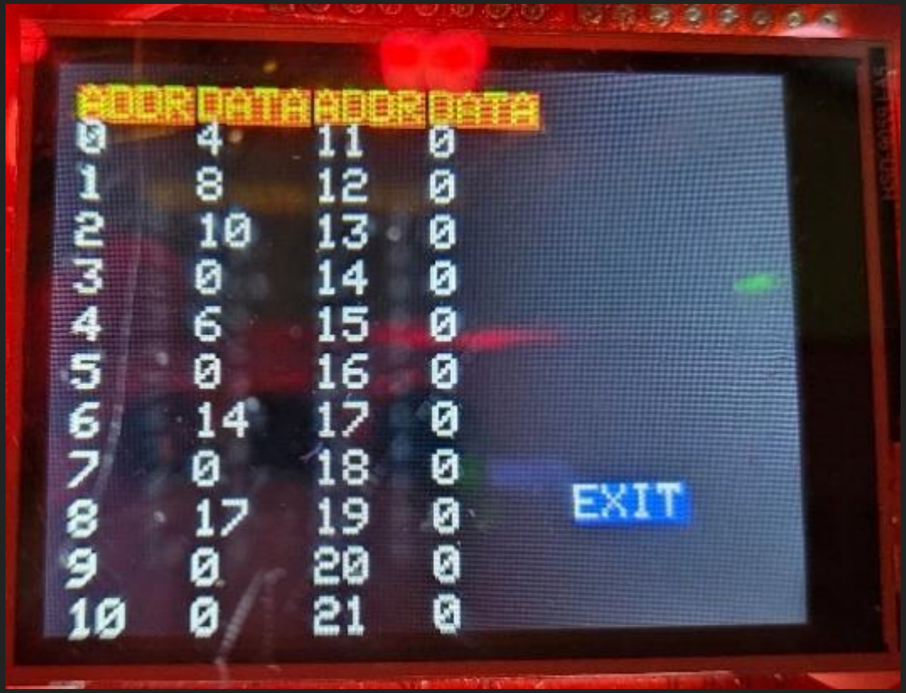
- It needs time to be built
- The current model does not support enough wiring flexibility to plug and unplug the needed blocks
- Much work must be done by teachers to develop suitable exercises and scenarios for the students
- Version 2 ?

SYSTEM VERIFICATION

Testing Program

Instruction	Byte code	Address (content) (in decimal)
LOD A,8	(dec) 04 08, (hex) 04 08	00ª (04), 01 (08)
INC A	(dec) 10 00, (hex) 0A 00	02ª (10), 03 (00)
MOV B,A	(dec) 06 00, (hex) 06 00	04ª (06), 05 (00)
ADD A,B	(dec) 14 00, (hex) 0E 00	06ª (14), 07 (00)
STOP	(dec) 17 00, (hex) 11 00	08ª (17), 09 (00)

Program in memory



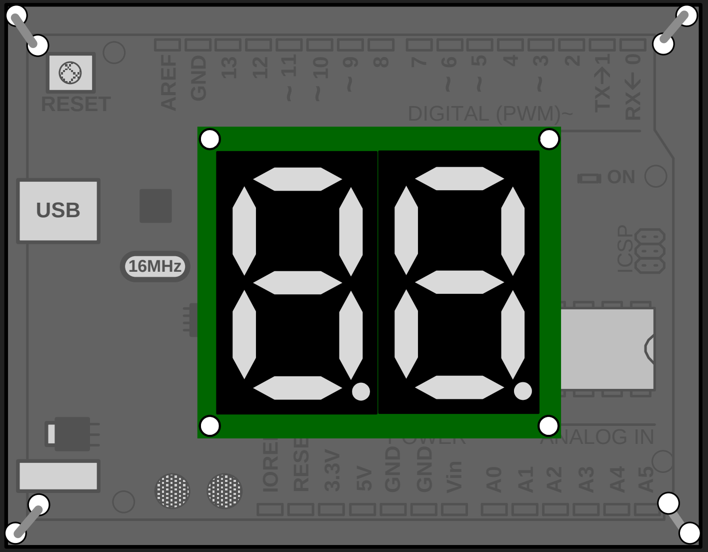
ADDR	DATA	ADDR	DATA
0	4	11	0
1	8	12	0
2	10	13	0
3	0	14	0
4	6	15	0
5	0	16	0
6	14	17	0
7	0	18	0
8	17	19	0
9	0	20	0
10	0	21	0

SYSTEM VERIFICATION

Execution sequence for the instruction ADD A,B

STEP 1: The PC shows the starting address of the instruction to be executed (ADD A,B)
STEP 2: The starting address of the instruction is stored in MAR register
STEP 3: The first instruction byte is fetched and is stored in MBR register
STEP 4: The MAR address is increased by one, to point to the next address where the second byte of the instruction is stored
S**TEP 5:** The second instruction byte is fetched and is stored in MBR register
STEP 6: The control unit decodes the instruction bytes and starts to execute the instruction
STEP 7: The content of register A is copied in the register T1 which is the first input of the ALU (Arithmetic and Logic Unit)
STEP 8: The content of register B is copied in the register T2 which is the second input of the ALU (Arithmetic and Logic Unit)
STEP 9: The addition T1+T2 is performed inside the ALU
STEP 10: The result is stored in register A
STEP 11: The content of the PC register is updated (increased by two) for pointing to the next instruction in memory

How it works (1) [indicative]



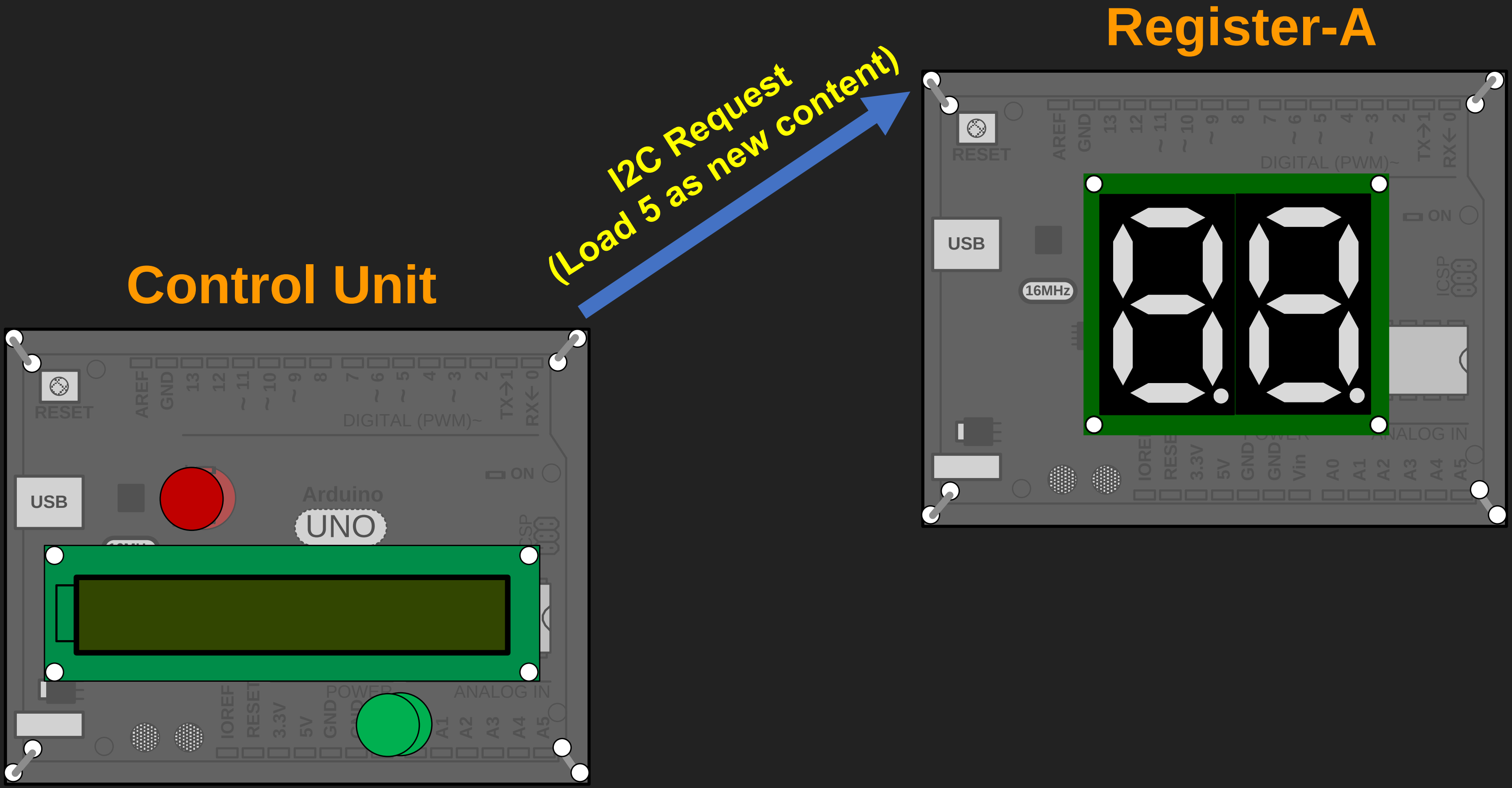
Every Register block

- Runs the same software
- Accepts requests from Control Unit (master unit), with or without answer
- Supports the operations:

READ = Read content
RESET = Set content to zero
SHIFT_L = Shift content left
SHIFT_R = Shift content right
DEC = Decrement content by one
INC = Increment content by one
LOAD = Load new content

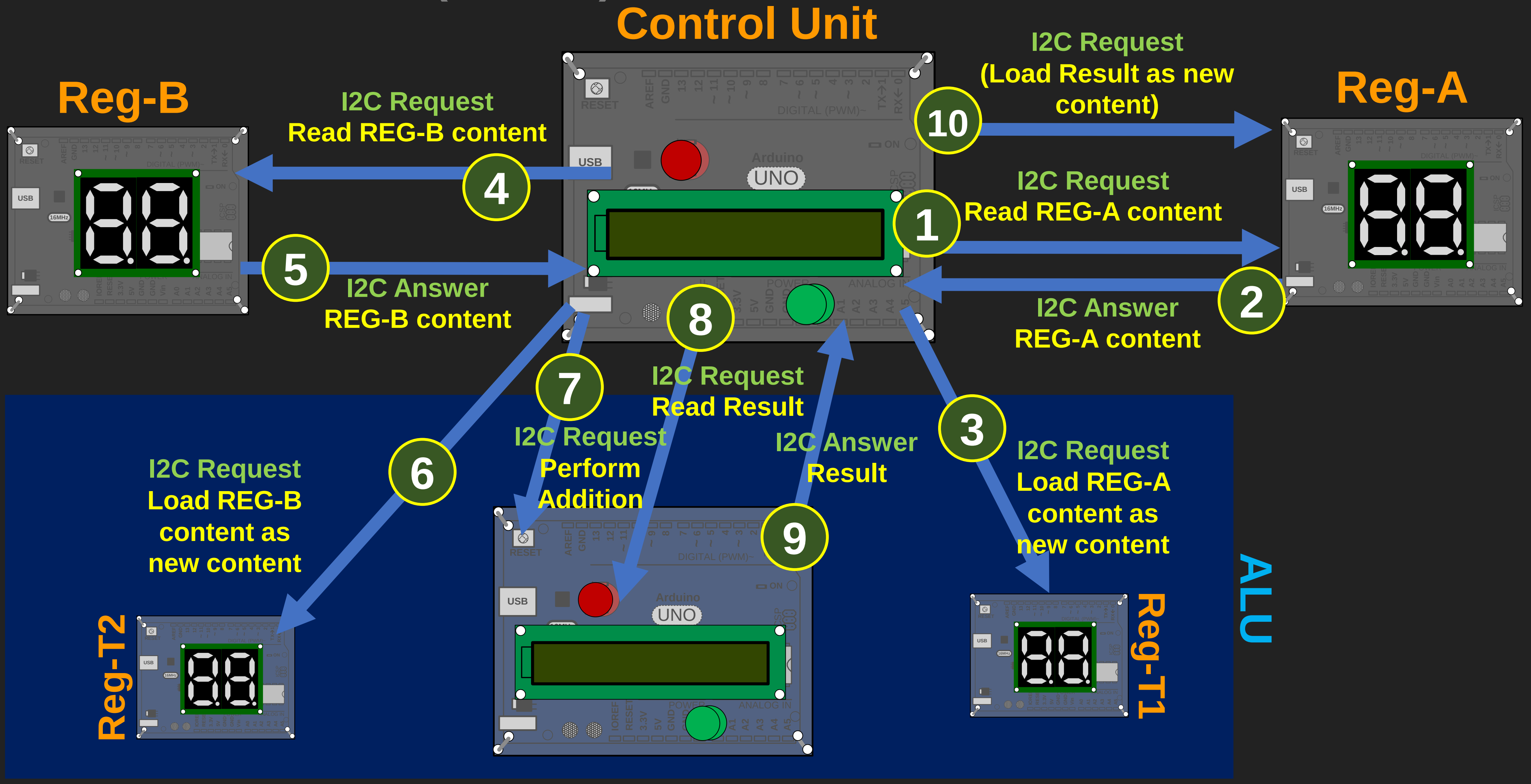
How it works (2) [indicative]

Executing the instruction MOV A,5



How it works (3) [indicative]

Preparing the execution of the instruction ADD A,B (A=A+B)


Control Unit
Reg-B
Reg-A
Reg-T2
Reg-T1
ALU
1 I2C Request Read REG-A content
2 I2C Answer REG-A content
3 I2C Request Load REG-A content as new content
4 I2C Request Read REG-B content
5 I2C Answer REG-B content
6 I2C Request Load REG-B content as new content
7 I2C Request Perform Addition
8 I2C Request Read Result
9 I2C Answer Result
10 I2C Request (Load Result as new content)

Dr. P. Papazoglou

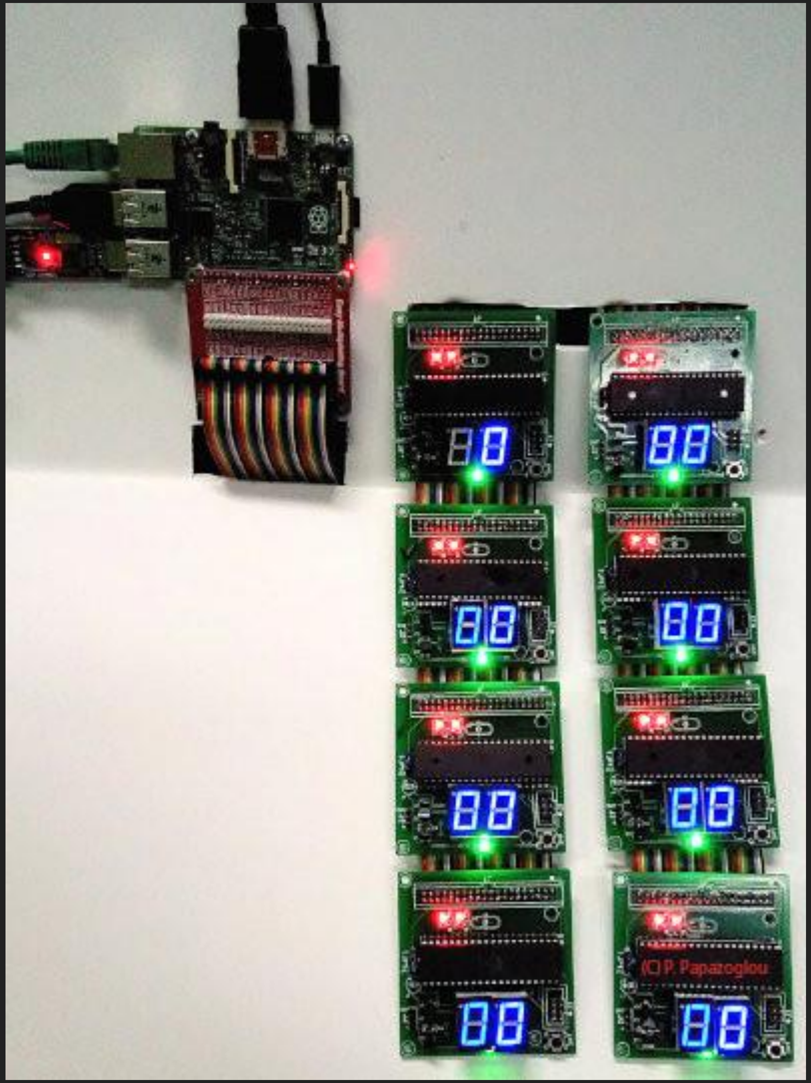
Demo video

Conclusions

- A unique simulator kit for studying and programming an educational experimental microprocessor has been presented.
- The proposed kit is fully functional and supports step by step execution of custom assembly instructions at hardware level.
- For organizing a full semester lab, multiple simulator kits must be reproduced.
- There is much work to be done for constructing and combining the needed internal blocks as well as the implementation of new assembly instructions.
- Based on the above concerns, plug-n-play PCBs must be designed for hosting microcontrollers, displays, buttons and replacing current model implementation.
- More educational scenarios must be also designed.

Previous work

- P.M.Papazoglou, D.A.Karras, A Hardware Based Novel Educational Methodology for Teaching Microprocessor Architectures Using Object Oriented Approach, International Review on Computers and Software (IRECOS), Vol 10, No 10 (2015)
- P.Papazoglou, A.Moschos, OpenHardSim: An Open Source Hardware Based Simulator for Learning Microprocessors, IEEE Global Engineering Education Conference (EDUCON 2017), 25-28 April 2017 Athens, Greece
- P.M.Papazoglou, A Hybrid Simulation Platform for Learning Microprocessors, Computer Applications in Engineering Education, 10.1002/cae.21921, (pp 655-674) WILEY, 2018
- Panayotis Papazoglou, Replacing Microprocessor Simulators: Developing Hybrid Technology in Modern Education, International Conference in Education, Research & Development, 25-28 Aug 2022, Burgas, Burlgaria



Thank you!

https://homs.panospapazoglou.gr/