Université Sétif 1
Faculté des Sciences
Département d'informatique

2-ème année Inforamtique (année 2019)
Module Architecture des Ordinateurs
Enseignant Kara Abdelaziz

Less than 10 TTL chips from scratch 4-bits processor project

1. Introduction

The project for this year is relatively simple compared to those of the previous years, it consists to implement a minimalist 4-bit architecture of Harvard which contains all the basic components necessary for the design of a complete hardware architecture programmable with its own machine language, the architecture revolves around the main element which is the processor, supported by a RAM memory to contain the data and a ROM to contain the program. The architecture in question is described in the following section (Architecture description), it is a direct representation of the CHUMP processor (Cheap Homebrew Understandable Minimal Processor) described by *Dave Feinberg* in his paper **A Simple and Affordable TTL Processor for the Classroom**, The architecture represented here is an identical replica as that presented in the paper. The implementation is to be done either on the Logisim simulator by following step by step the construction of the different components, or a real physical construction from TTL logic circuits as described in the paper. Although before being able to start the construction, it is strongly advised to the student to deepen the basic concepts of micro-architecture (computer organization) by following the tutorial of Ben Eater used in the past years to build the processor SAP-1, or take inspiration from the Mic-2 architecture presented last year as a project.

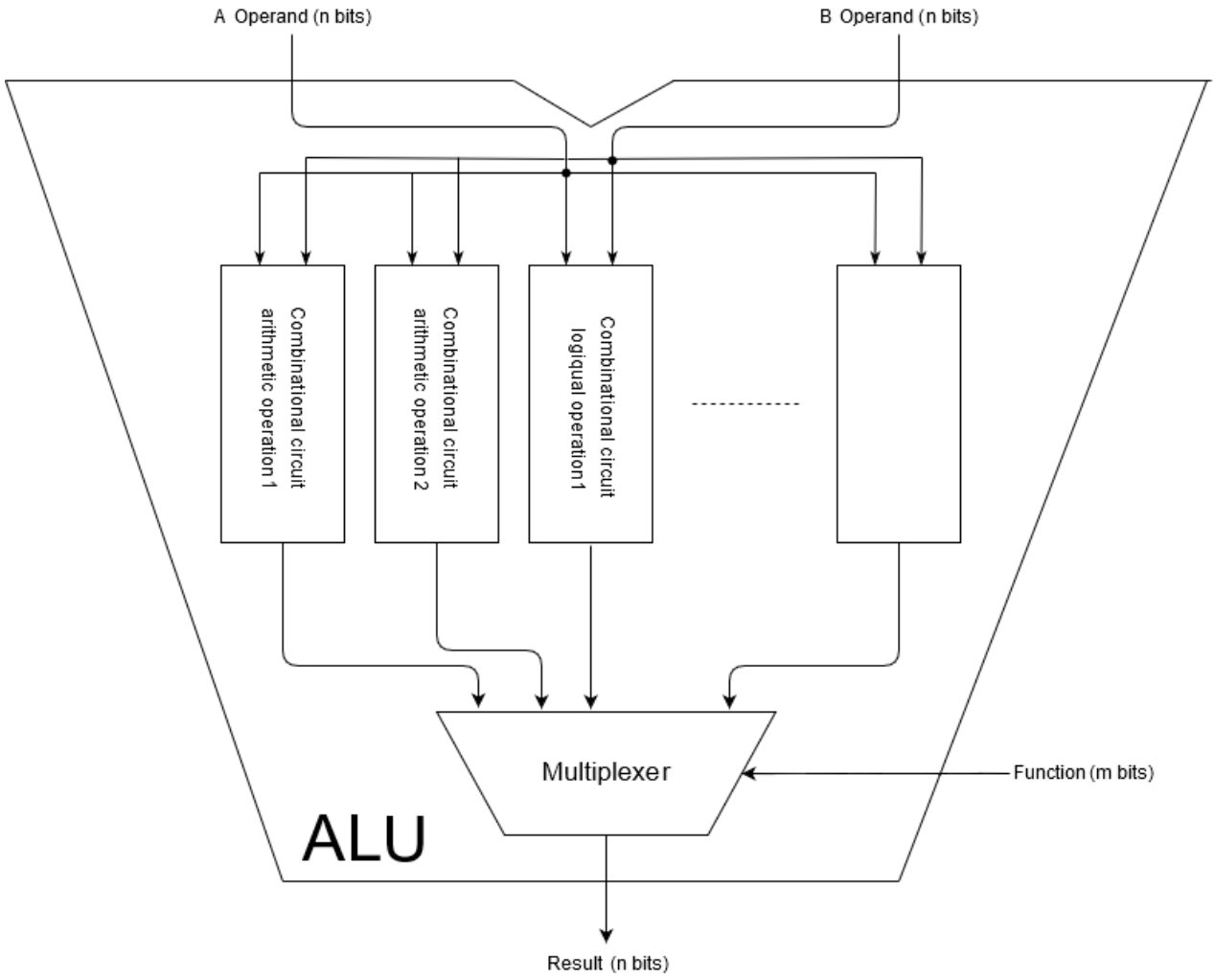
2. Architecture description

The construction of the architecture takes place over three stages, the first is the ALU (Arithmetic and Logic Unit), the second is the datapath (The path of instruction) and the third the CU (Control Unit). The design of a processor (because it represents almost all of the architecture) always follows these three stages, the ALU determines the arithmetic and logical operations that the processor can perform, the datapath is a network of paths in which each instruction takes its own way, it is the metaphor of a railway network in which the trains change the way by switches, the Multiplexers are responsable for making this work of routing of the paths in the processor. The CU's role is to recognize the instruction when it has entered the processor (this is called the decoding of the instruction) then to trace the appropriate path for it in the datapath by operating the Multiplexers and the controls of the various components on the path.

2.1. The ALU

For this year the ALU is based on a very well-known TTL chip, the 74181. It is a 4-bit ALU which comprises a total of 16 functions, for the physical realization it is necessary to refer to the datasheet of the circuit, or searching for more materials on the internet. For the implementation on a simulator two implementations are possible, either to use the ALU 74181 logic gates based diagram, available on the Internet or in its datasheet, and to reproduce it gate-by-gate on Logisim, or to create a new simplifed one based on the generic design of a ALU shown in the figure below, in which the ALU is presented in the form of several combinatorial circuits performing for each a single logical or arithmetic operation, all circuits receives the same two 4 bits operands A and B (n=4 for our case) and do all their calculations in parallel and return their results to the Multiplexer, a single result is chosen by the Multiplexer among the 8 proposed, the choice of the Multiplexer to let through the result of a specific circuit is taken in relation to the number encoded in binary on m bits (in our case m=3 for 8 operations, truly only 5 operations are needed) provided on the Multiplexer selector input. The table below summarizes the 3 bits binary encoding functions for the ALU. It should also be noted that the presented order of the 5 operations on the table is not fixed and can be changed or swapped, and the addition of other operations is also possible, although the 5 operations listed are mandatory for the proper working of CHUMP.

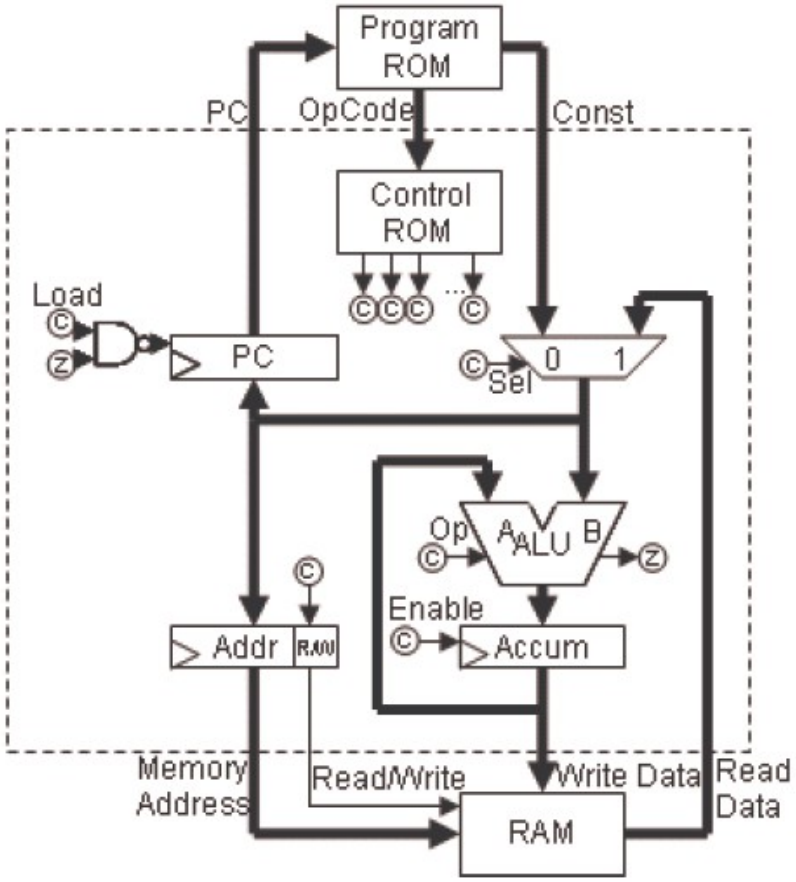
F	F2	F1	F0	ALU Output
0	0	0	0	A
1	0	0	1	B
2	0	1	0	A+B
3	0	1	1	A-B
4	1	0	0	0
5	1	0	1	...
6	1	1	0	...
7	1	1	1	…



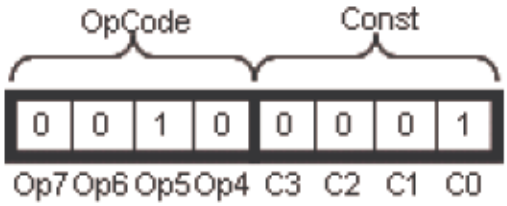
Of course it is possible to do some optimization to reduce the number of gates, It is not mandatory but you can take inspiration from the SAP-1 or Mic-2 ALU implemented past years, or from the ALU shown in the book **Digital Design and Computer Architecture** (section 5.2.4 page 248).

2.2. Datapath

As mentioned before, datapath is an interconnected network of paths, each instruction walk through its own path in this network. In a path we most often found the RAM, registers, combinational circuits (among others the UAL), Multiplexers, tristate buffers ... etc. The processor path is described in the figure below copied directly from the **A Simple and Affordable TTL Processor for the Classroom** paper. The datapath for this processor is quite simple, it is based on a RISC (Reduced Instruction Set Computer) architecture with a very small number of instructions, which allows the CHUMP processor to execute an entire unique instruction in a single cycle clock. Unlike SAP-1 and Mic-2 in which the instruction is cut into several pieces called micro-instructions, where each piece is executed on a single clock cycle.



The datapath in the CHUMP processor begins with the Program ROM, a 16-bytes memory that holds the program to be executed, each instruction is in an 8-bits memory cell, it consists of two parts: an OpCode indicating which operation the processor should perform followed by a constant/immediate value of 4 bits used as a data value, or as an address in RAM or even as an address which points in the program ROM. The binary representation of an instruction is like in the diagram below. Therefore, the CHUMP can only handle 4-bits data values, representing numbers in the narrow range of 0 to 15 (or -8 to 7 in 2-complement representation). Likewise, it can access only 16 RAM/ROM locations. Finally, the use of a 4-bits PC (Program Counter) means that programs cannot exceed 16 lines. The instruction set of CHUMP is of 7 different instructions, each instruction can take as operand 2 variants; an immediate found on the same instruction binary, or a value in the RAM memory, which leads In total of 14 instructions.



On the datapath diagram each component represents a single TTL circuit, which are connected by 4-bits buses represented by thick arrows. The number of registers used inside the CPU is 2 of 4 bits each, The Accum register, and the Addr register. The Accum register is called accumulator because of the way it works, it is a register involved in all the arithmetic operations of the processor, it is always connected to the first operand of the ALU, and always receives the result from the ALU. The 2nd operand on the other hand can take according to the instruction two possibilities, either the immediate or a value from the memory, determined by the multiplexer. The second Addr register is a register which holds the address of the cell pointed to in RAM. The 4-bits PC counter (Counter Counter) pointing to the instruction being executed in the ROM. The multiplexer 2-1 (4-bits) is used to choose between the immediate or the information in RAM, The output of the multiplexer can be used as operand B of the ALU, or as an address in the PC for instructions Jump, or as a RAM address in Addr register. And in the end, a 16-cells RAM with a memory word of 4 bits size.

2.3. Instructions Set

The CHUMP instructions set includes seven key operations, each in two versions: immediate and memory. For example, there is an ADD instruction to add a constant to the accumulator, and another ADD to add a memory value to the accumulator. The immediate 4-bits portion of the instruction is ignored by the seven memory commands. The following table describes the seven immediate instructions and the 7 memory instructions. The difference between the memory instructions and the immediate instructions is in the bit Op4 of the least significant bit in the instruction OpCode (see the diagram of the binary representation of the instruction).

Instruction	Type	OpCode	Fonctionnement	Description
LOAD	immediate	000|0	Accum ← imm ; PC++ ;	Load the accumulator with an immediate or a memory value.
	memory	000|1	Accum ← Mem[Addr] ; PC++ ;	
ADD	immediate	001|0	Accum ← Accum + imm ; PC++ ;	Add an immediate or a memory value to the accumulator.
	memory	001|1	Accum ← Accum + Mem[Addr]; PC++ ;	
SUB	immediate	010|0	Accum ← Accum - imm ; PC++ ;	Subtract an immediate or a memory value from the accumulator.
	memory	010|1	Accum ← Accum - Mem[Addr]; PC++ ;	
STORETO	immediate	011|0	Mem[Addr] ← Accum; PC++ ;	Save the value of the accumulator in the memory.
	memory	011|1	Mem[Addr] ← Accum ; PC++ ;	
READ	immediate	100|0	Addr ← imm ; PC++ ;	Load the memory address with an immediate or a memory value.
	memory	100|1	Addr ← Mem[Addr] ; PC++ ;	

GOTO	immediate	101|0	PC ← imm ;	Jump to the instruction with the address in the immediate or the memory value.
	memory	101|1	PC ← Mem[Addr] ;	
IFZERO	immediate	110|0	If (Accum == 0) PC ← imm ; else PC++ ;	Jump to the instruction with the address in the immediate or the memory value, if the accumulator is equal to zero.
	memory	110|1	If (Accum == 0) PC ← Mem[Addr] ; else PC++ ;	

Note : in the preceding table *imm* designates the immediate inside of the instruction, and *Mem[Addr]* the memory cell with the address in Addr.

Exemple : We can see on the listing below an overview of a small program in machine language and assembler, the program is very simple, it allows to increment +1 infinitely the memory cell in RAM with address 2.

```
0:  10000010    READ 2
1:  00010000    LOAD
2:  00100001    ADD 1
3:  01100010    STORETO 2
4:  10100000    GOTO 0
```

2.4. Control Unit (CU)

The control unit is the unit responsible for decoding the OpCode of the instruction and triggering the appropriate commands for the various components of the datapath, and thus allowing the instruction to follow its appropriate path in the datapath for its execution. Knowing that CHUMP is a RISC processor, the instruction is to be executed on a single clock cycle, a combinational circuit is used to act as a control unit, by capturing the OpCode of the instruction on 4 bits and firing instantly the control signals for each datapath component. The combinational circuit will be implemented in CHUMP by a ROM, the ROM in question is represented in the datapath diagram by the name Control ROM, it receives the 4 bits of OpCode as input and produces the control signals as output.

The components involved by the CPU control are those with the © sign on the datapath diagram: The Multiplexer, The ALU, The accumulator, RAM and the PC. The Multiplexer actually receives the selection command directly from the OpCode without going through the ROM, it is the bit Op4 with the least significant weight of the OpCode in the instruction which will decide whether to take the information from the Immediate or from RAM. The ALU 74181 needs 6 bits to chose a function, however it is necessary to find a way to use only 5 bits, due to the limitation of the ROM which only has 8 bits of memory word output, the 3 remaining bits are used by the other components. The RAM control in 1 bit for reading/writing, and the PC with 1 bit to choose between incrementing the PC or to load a new address, and the accumulator with 1 bit for its write command. Which makes 8 bits in total, matching the output size of the ROM.

The control signals for each instruction are listed in table below :

Instructions	ALU (5bit)	Accumulateur (1bit)	RAM (1bit)	JMP(1bit)
LOAD	B	write	read	PC++
ADD	A+B	write	read	PC++
SUB	A-B	write	read	PC++
STORETO	X	don't write	write	PC++
READ	X	don't write	read	PC++
GOTO	0	don't write	read	write if Z
IFZERO	A	don't write	read	write if Z

Note 1: The X value in the table indicates that the commend transmitted to the ALU is not important, cause its result will not be taken into account.

Note 2: The Z Signal is a 1-bit signal from the ALU, indicating whether the output result from the ALU is equal to zero or not.

It should be noted on the datapath diagram the implementation for the control signal of the PC counter that its signal is generated through a NAND gate, its role is to produce the value 0 (imply a Jump) only if the signal Z and the JMP signal are equal to 1, otherwise it is 1, whitch indicates to the PC to make an increment PC++. To make a jump the PC must be loaded by a new address coming from Mux 2-1. The NAND gate is used with the 2 instructions GOTO and IFZERO to implement the conditionnal jump. For GOTO instruction the jump is unconditional and the Z must always be 1, so the ALU must apply the function which outputs the value 0 (on the ALU 74181, this must be logic function 1, see the datasheet for more explanation). On the other hand for the IFZERO instruction, the ALU must pass the operand A coming from the accumulator to know if it is equal to zero or not, therefore apply the function A (it must be the function $\overline{A}$ for the 74181, see datasheet).

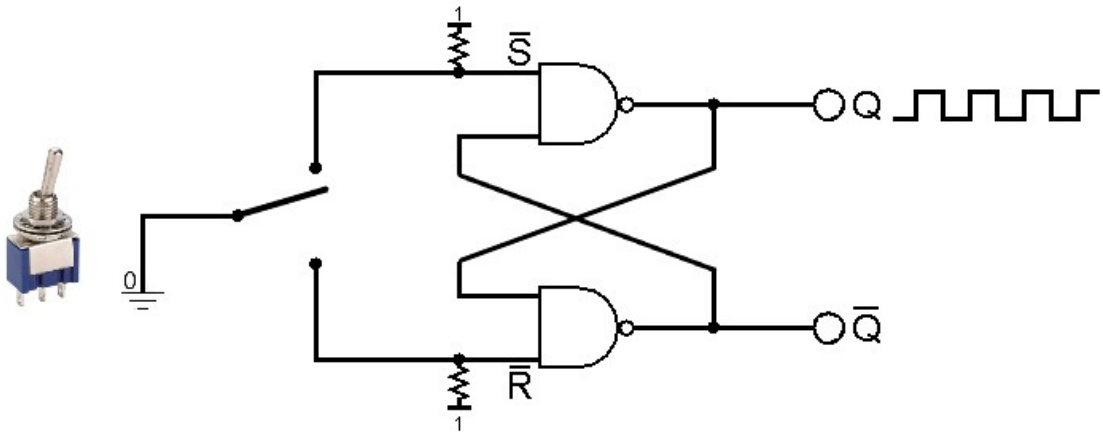
An equally important remark concerning the CHUMP processor is that the Addr register does not have a dedicated control signal, which implies that it is a register whom changes value with each rising edge of the clock, meaning for each instruction. This information is important to take into account when reading and writing in RAM, you must always be aware of what has been put in this register in the previous instruction, remember that this value comes directly from the output of the Multiplexer, so either immediate or RAM. This observation is only really true for reading from RAM, therefore a LOAD instruction must always be preceded by a READ instruction. On the other hand, an improvement has been put in place to avoid using 2 instructions for writing to RAM, it is to pass the RAM write signal through the Addr register (visible on the datapath diagram), and thus avoid adding a READ instruction before the STORETO instruction, the writing is actually done on the rising edge following the STORETO instruction. The Addr register must contain at least 5 bits of memory to contain the 4 address bits plus the RAM write signal.

2.5. The physical implementation of CHUMP

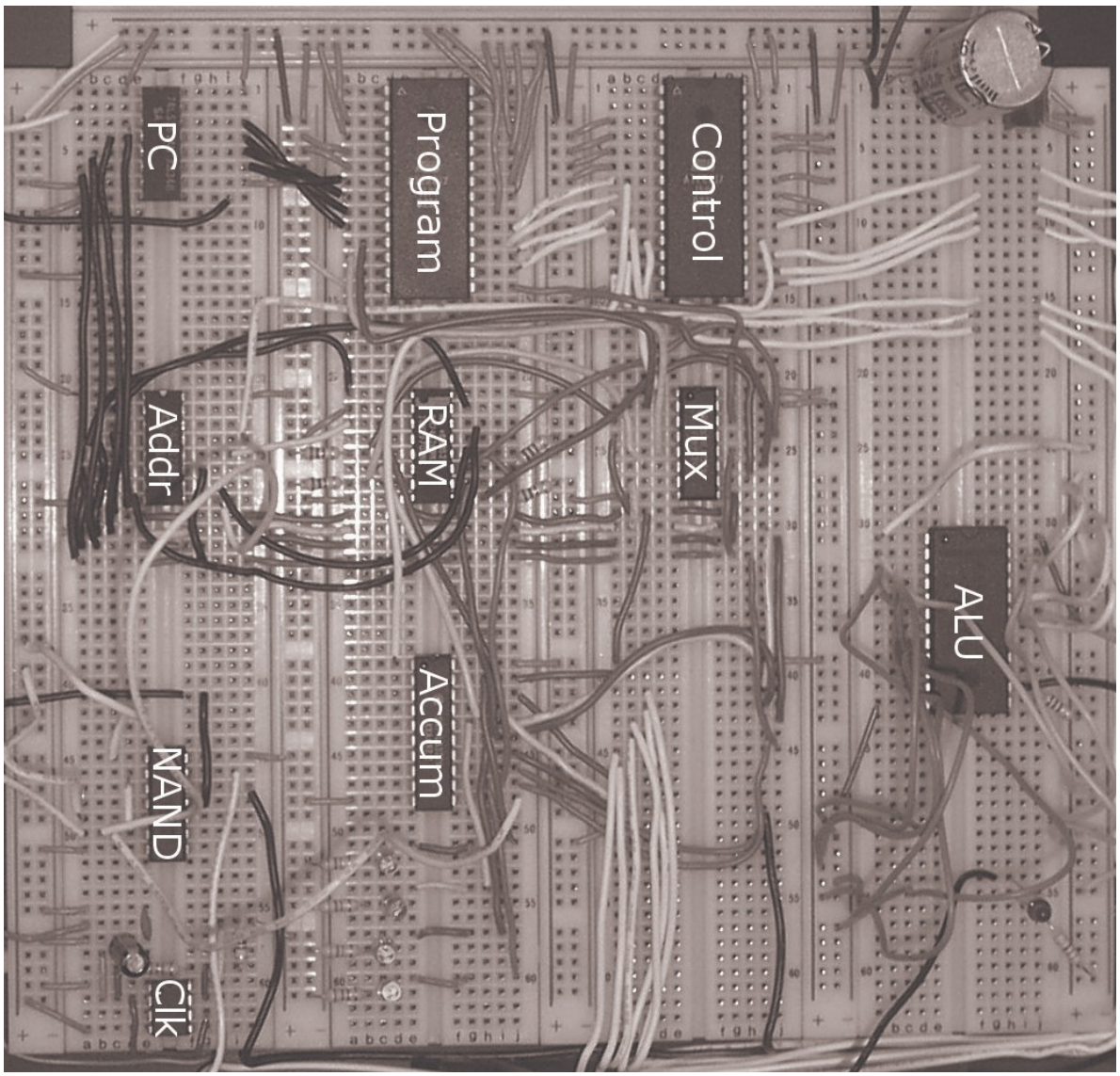
The physical implementation of CHUMP does not require more than 10 TTL circuits, it is quite simple, the list of components is on the listing below, it is a little different from those used in *Dave Feinberg*'s paper because of their spatial and temporal availability.

circuit	Description	Usage
7400	4 gates NAND	Clock and PC control
74157	Mux 2-1 (4 bits)	Multiplexer
74161	Counter (4 bits)	PC
74273	Register (8 bits)	Addr Register
74181	ALU (4 bits)	ALU
74377/74173	8 bits/4 bits Register	Accumulateur
HM61xx/HM62xx series	RAM	Data RAM
AT28xxx/AT27xxx series	ROM	two ROMs

Note : The xxx characters indicate different reference values for each model of RAM or ROM. The part xxx in the reference of a RAM/ROM generally designate their size.



A global representation of a physical implementation with the locations for each component and the various buses and electrical wires connecting them (captured from the paper) are shown in the image below. We can observe that this implementation uses a clock circuit not mentioned in the list, this is the famous 555. For our case we will use a simpler manual clock, its signal is generatd by a toggle switch, as shown in the image above, giving the user the option of alternating between 0 and 1 of the clock signal. The use of $\overline{R}\,\overline{S}$ Latch on the diagram will allow the elimination of the debounce problem, kind of micro-cuts in the signal which exists on all mechanical buttons or switches and will defect the clock signal (search more details on the internet). The clock signal will be generated on Q. The 2 NAND gates used to build the Latch will be salvaged from 7400 circuit, knowing that only one gate until now has been used and 3 are left.



وفقكم الله