Single cycle computer

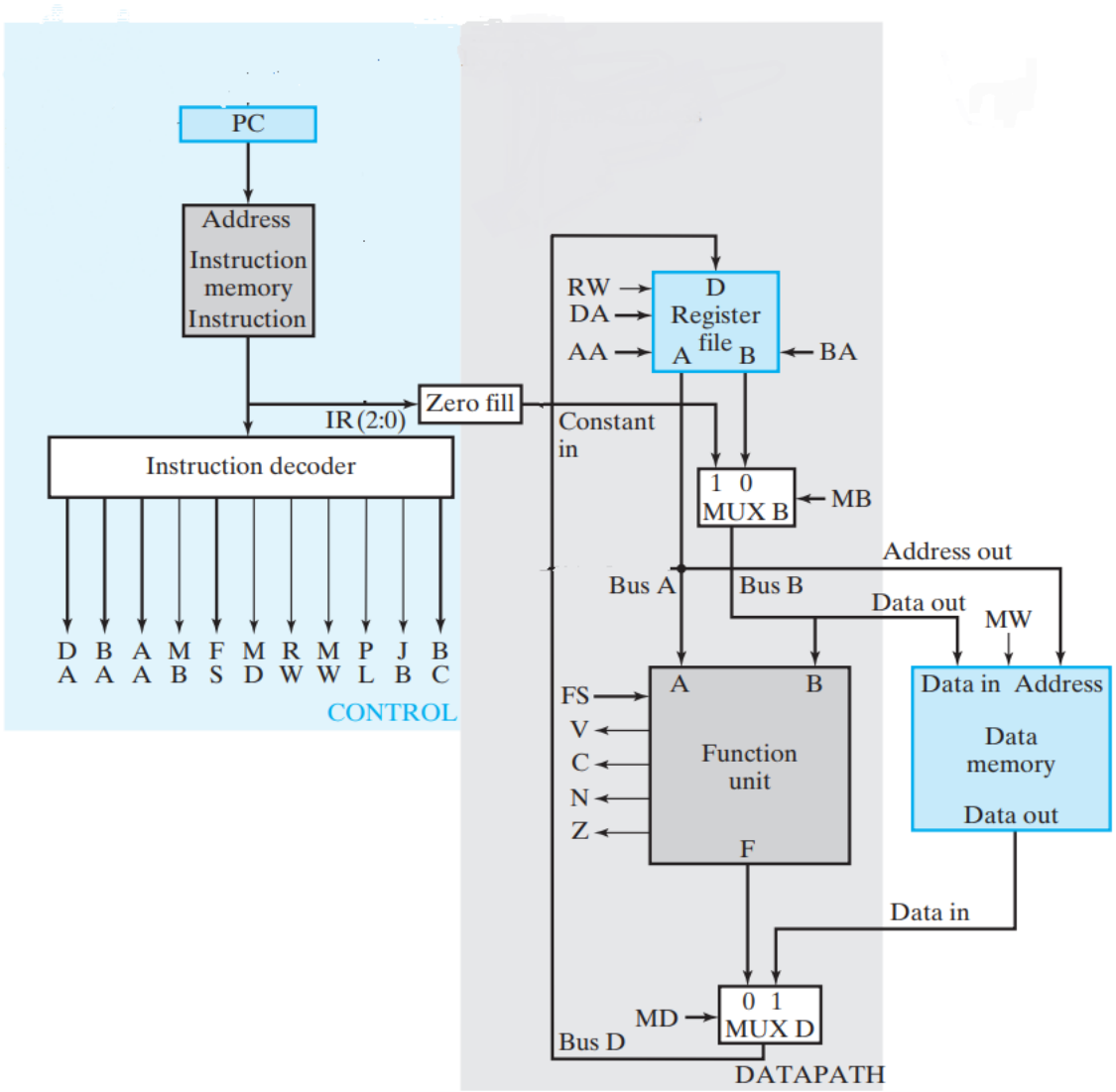
Supervisor: -

Dr. Khaled Elshafey

Single cycle computer

A single-cycle processor is a type of CPU design where each instruction is executed in a single clock cycle. This architecture simplifies control and instruction fetching, ensuring uniform execution time for all instructions but may result in lower efficiency due to potential idle time in some stages during each cycle.

Block Diagram for a Single-Cycle Computer:



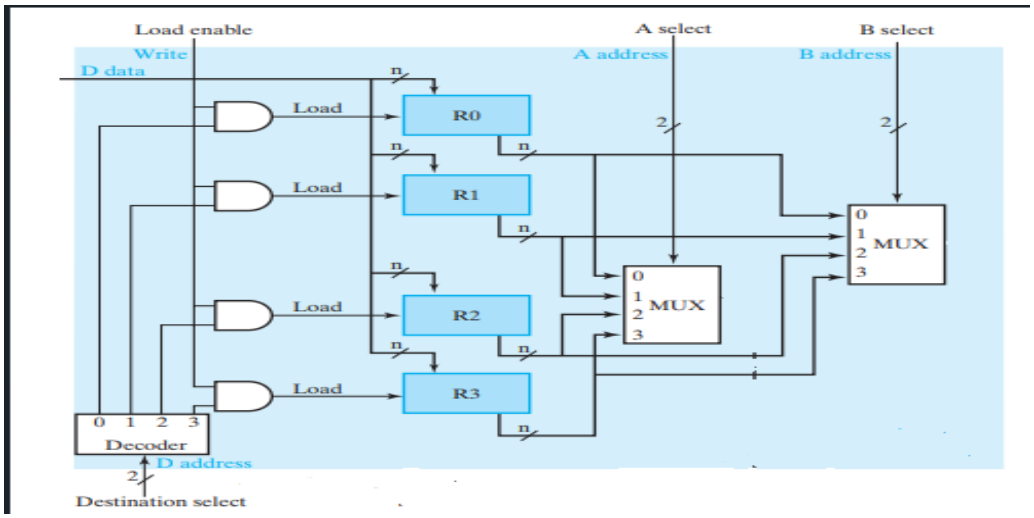
DATAPATHS:

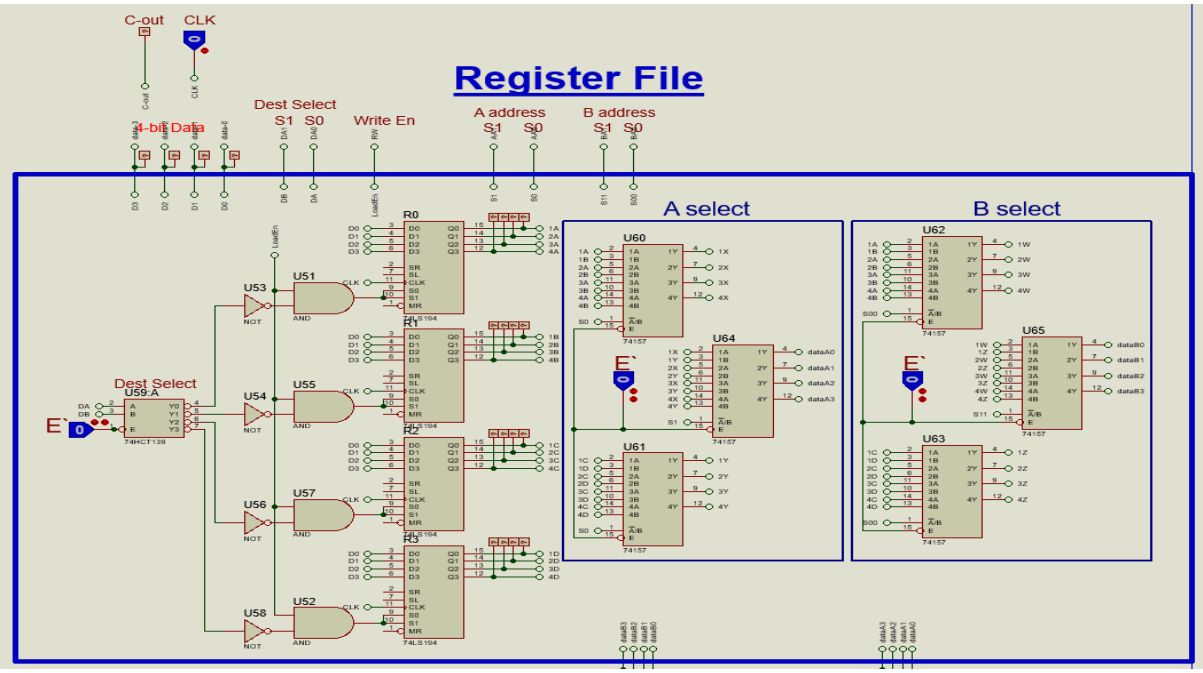
This component consists of a register file and a function unit. It manages the flow of data between the processor components. It consists of:-

1. Register file.
2. ALU (Arithmetic & Logic Functions)
3. Shifter

1- Register file

This component contains a set of registers (small, fast storage locations) that store data temporarily during instruction execution. It is also known as the register bank or the register set. The register file can store operands, results, or intermediate values. The register file is accessed by the CPU, which provides the register numbers to be read or written, and the ALU, which provides the data to be written.





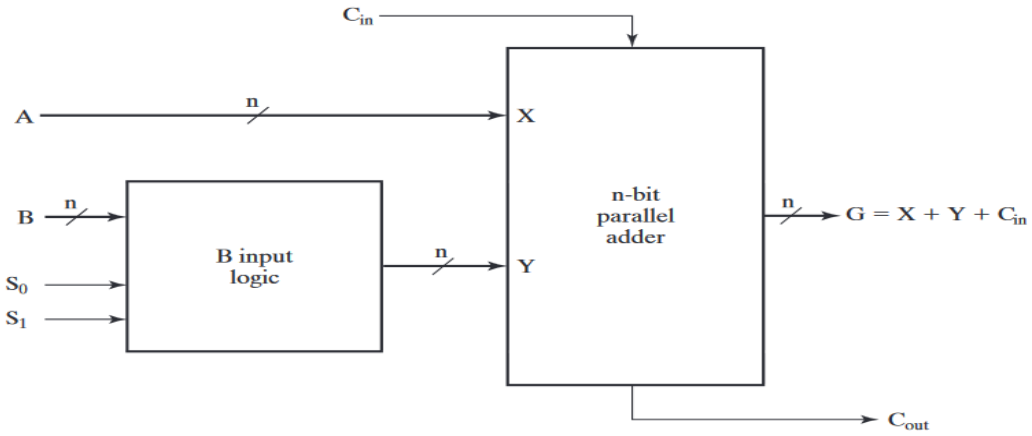
2- ALU (Arithmetic & Logic Functions) :

This component performs arithmetic and logic operations on data. It is also known as the execution unit or the functional unit. The ALU can perform operations such as addition, subtraction, multiplication, division, bitwise operations, comparison, and branching. The ALU receives operands from the register file or the sign-extend unit, and sends results to the register file or the data memory.

i- Arithmetic circuit :

The basic component of an arithmetic circuit is a parallel adder, which is constructed with a number of full-adder circuits connected in cascade , By controlling the data inputs to the parallel adder, it is possible to obtain different types of arithmetic operations.

Block Diagram of an Arithmetic Circuit :

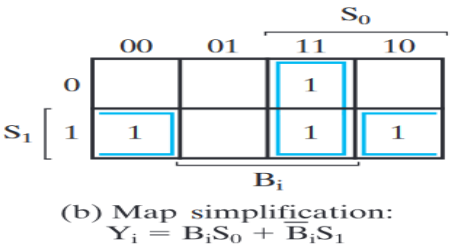


Function Table for arithmetic Circuit :

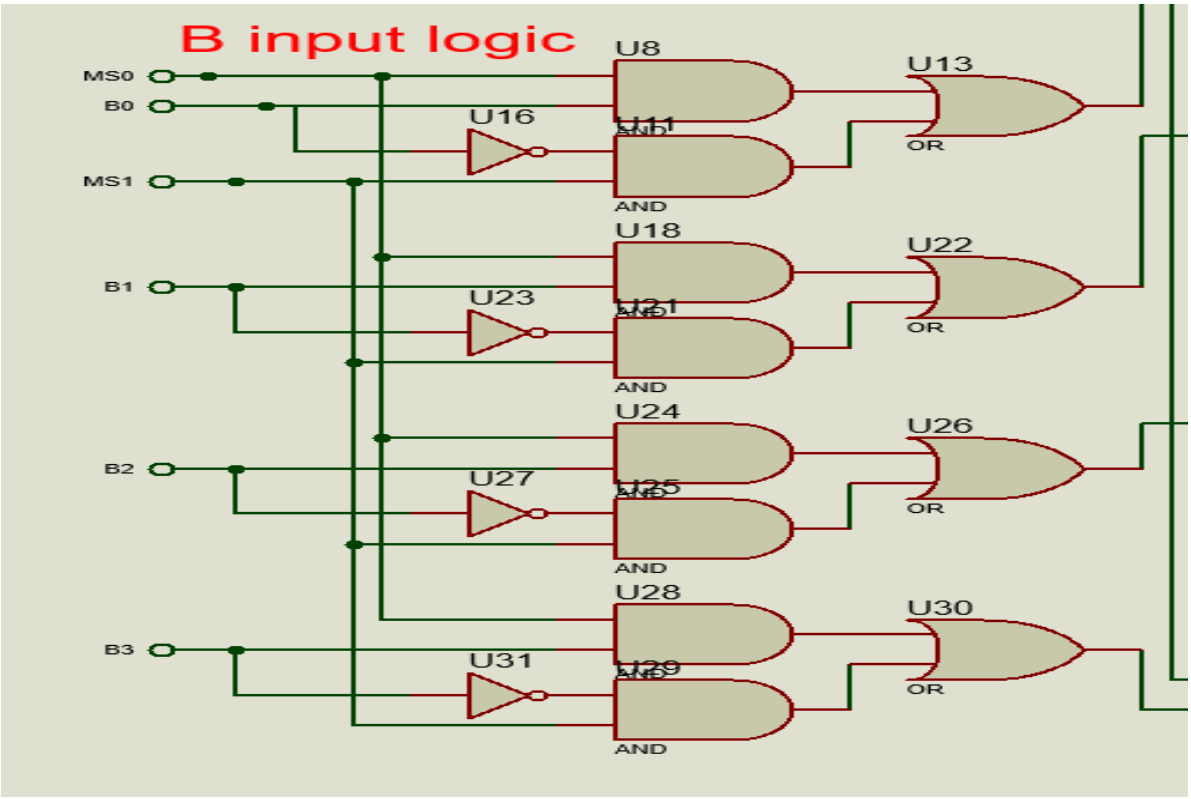
Select		Input	G = (X + Y + Cin)	
S1	S0	Y	Cin=0	Cin=1
0	0	all 0s	G = A (transfer)	G = A + 1 (increment)
0	1	B	G = A + B (add)	G = A + B + 1
1	0	B^-	G = A + B^-	G = A + B^- + 1 (subtract)
1	1	all 1s	G = A - 1 (decrement)	G = A (transfer)

B Input Logic for One Stage of Arithmetic Circuit :

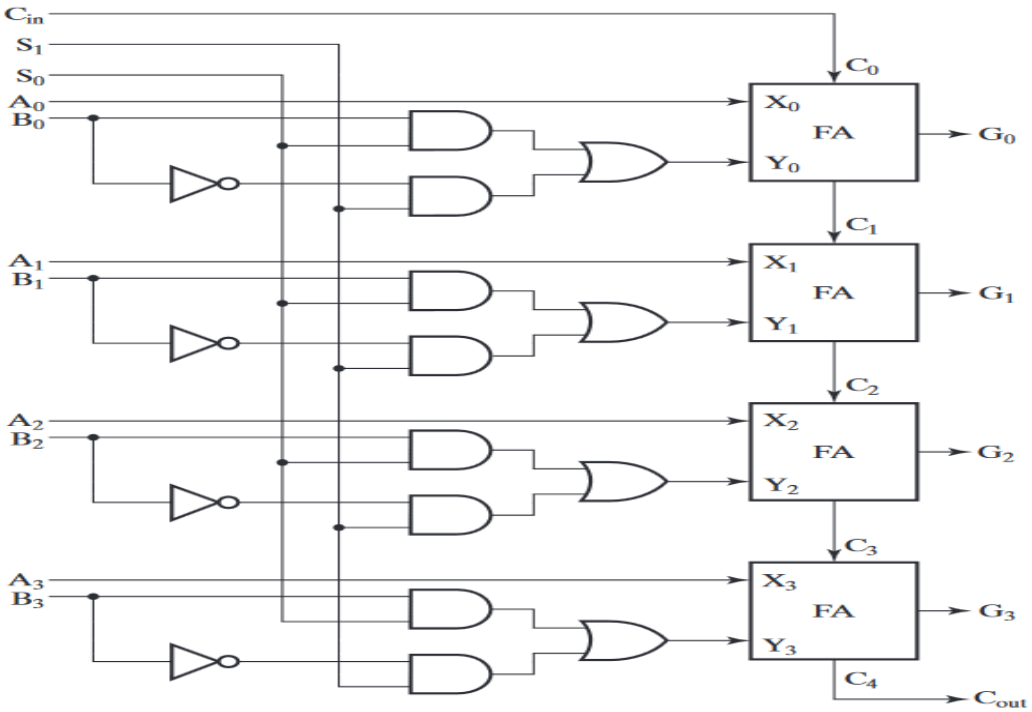
Inputs			Output	
S1	S0	Bi	Yi	
0	0	0	0	Y = 0
0	0	1	0	
0	1	0	0	Y =
0	1	1	1	Bi
1	0	0	1	Y=
1	0	1	0	Bi^{-}
1	1	0	1	Y =1
1	1	1	1	

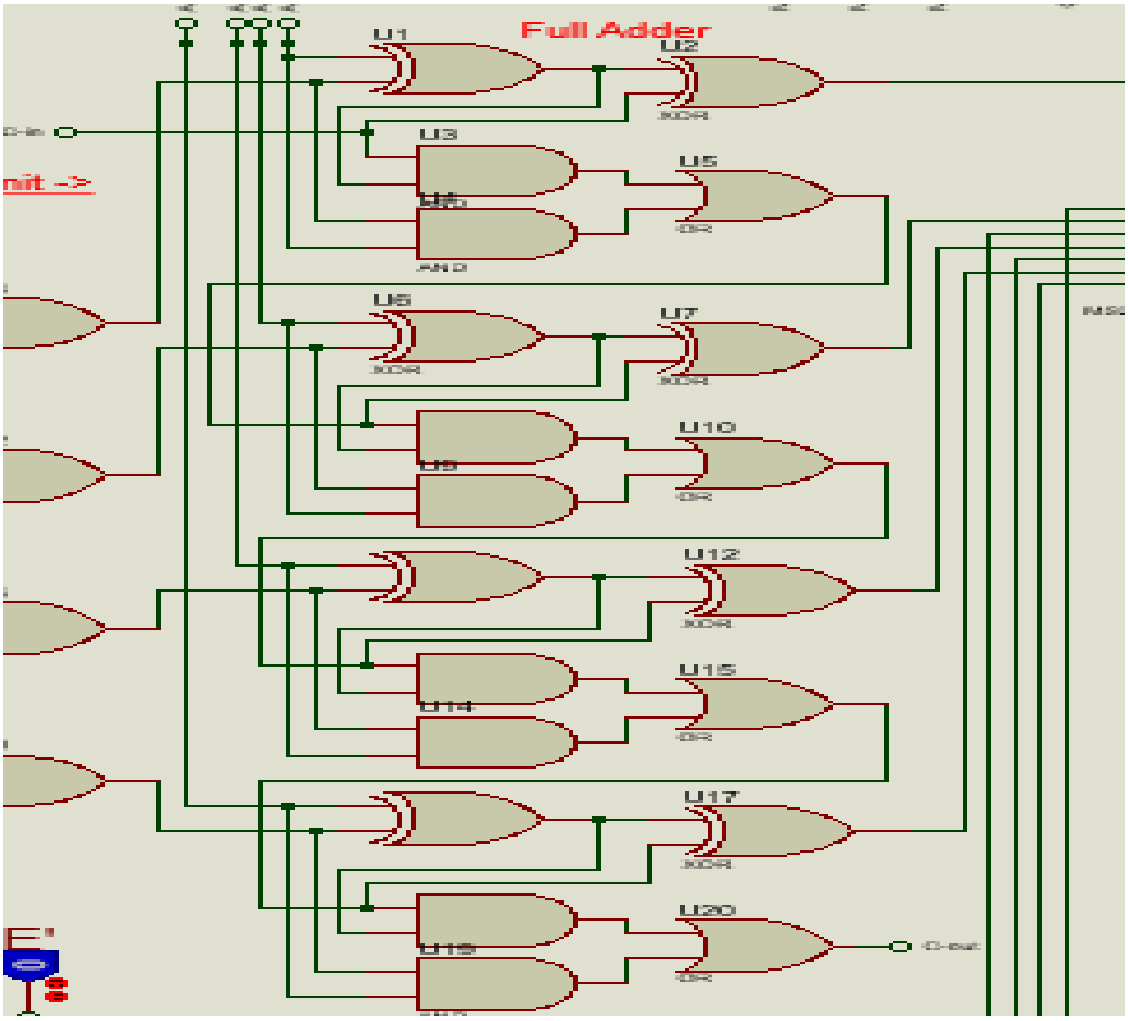


(b) Map simplification:
$Y_i = B_iS_0 + \overline{B}_iS_1$



Logic Diagram of a 4-bit Arithmetic Circuit

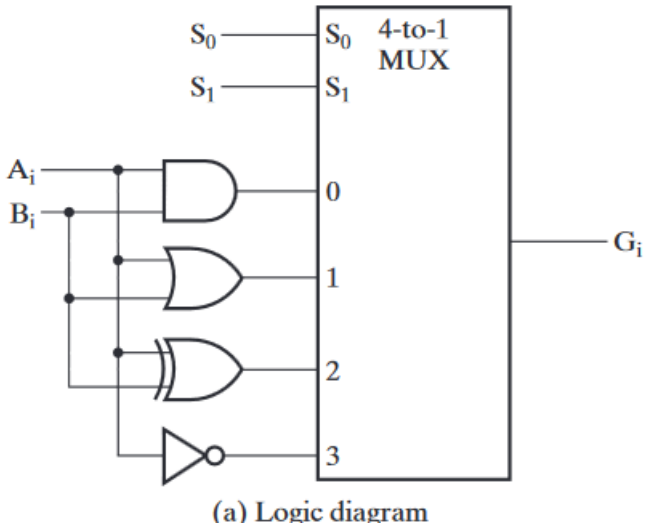




ii- Logic unit

S1	S0	Output	Operation
0	0	G = A . B	AND
0	1	G = A + B	OR
1	0	G = A ⊕ B	XOR
1	1	G = A^{-}	NOT

Logic table



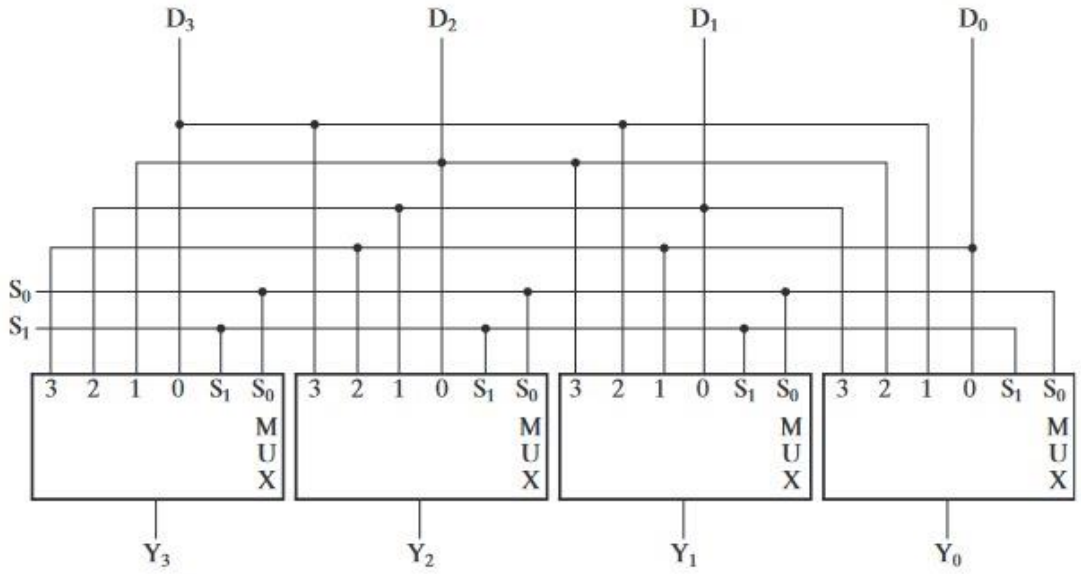
(a) Logic diagram

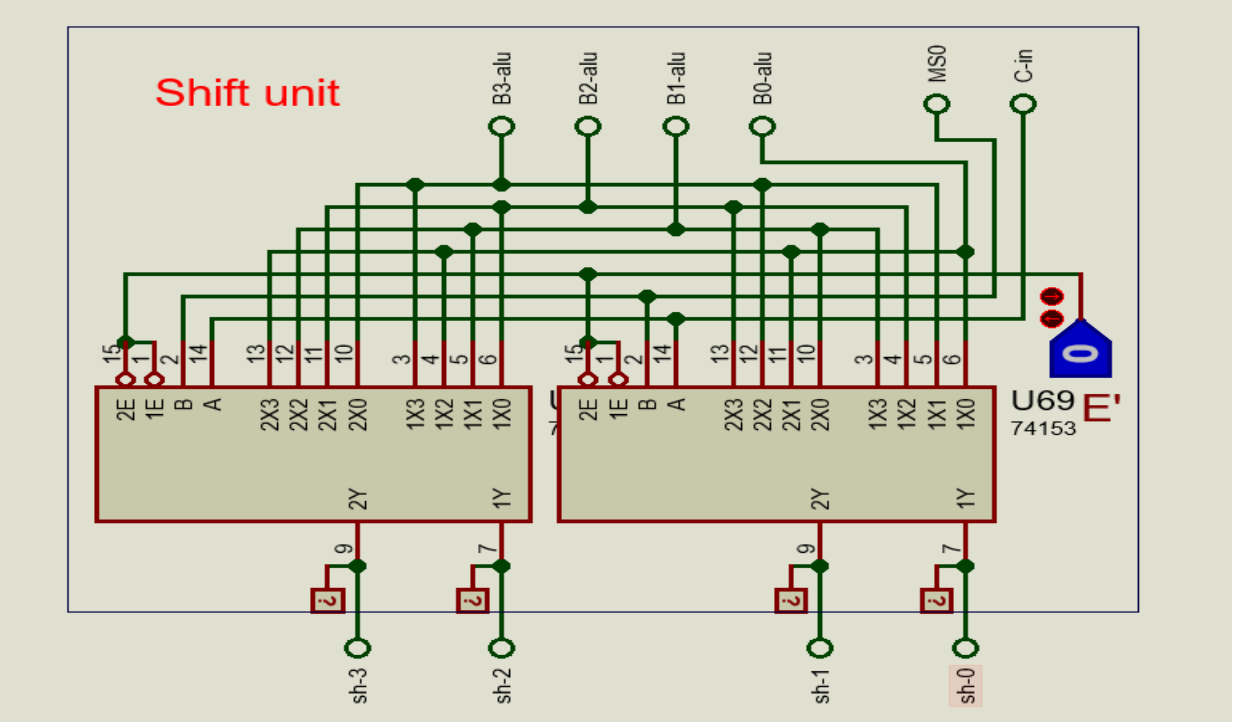
Function Table for ALU

Operation select					
S2	S1	S0	Cin	Operation	function
0	0	0	0	G = A	Transfer A
0	0	0	1	G = A+1	Increment A
0	0	1	0	G = A+B	Addition
0	0	1	1	G = A+B+1	Add with carry input of 1
0	1	0	0	G = A+B^{-}	A plus 1s complement of B
0	1	0	1	G = A+B^{-}+1	Subtraction
0	1	1	0	G = A-1	Decrement A
0	1	1	1	G = A	Transfer A
1	0	0	x	G = A . B	AND
1	0	1	X	G = A + B	OR
1	1	0	X	G = A⊕B	XOR
1	1	1	X	G = A^{-}	NOT(1s complement)

3) Shifter :

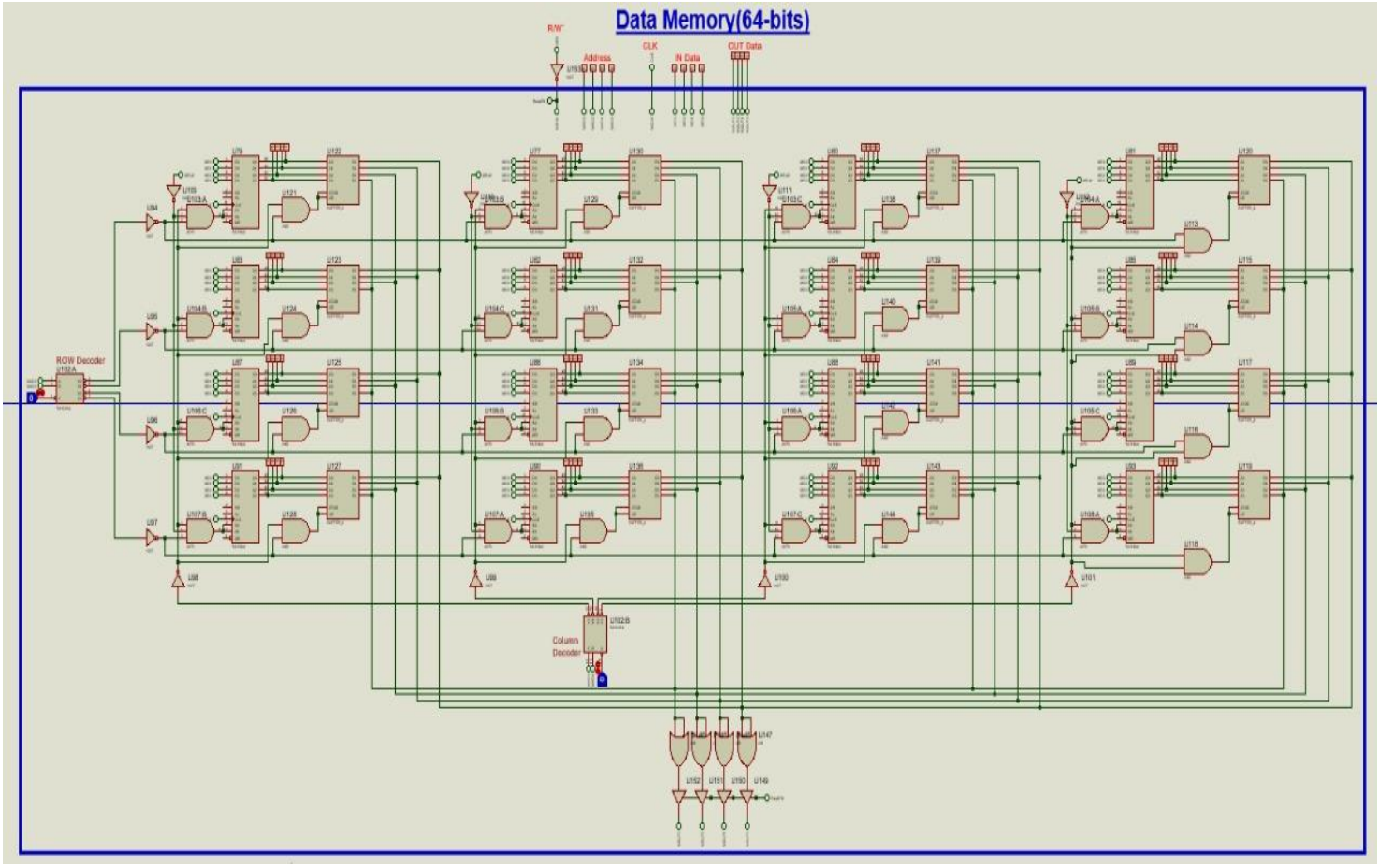
The shifter changes the value on bus B, putting the result on MUX input F. The basic shifter performs one of two main types of transformations on the data: shift-right and shift-left. In shifting to the left we only have.





Data memory :

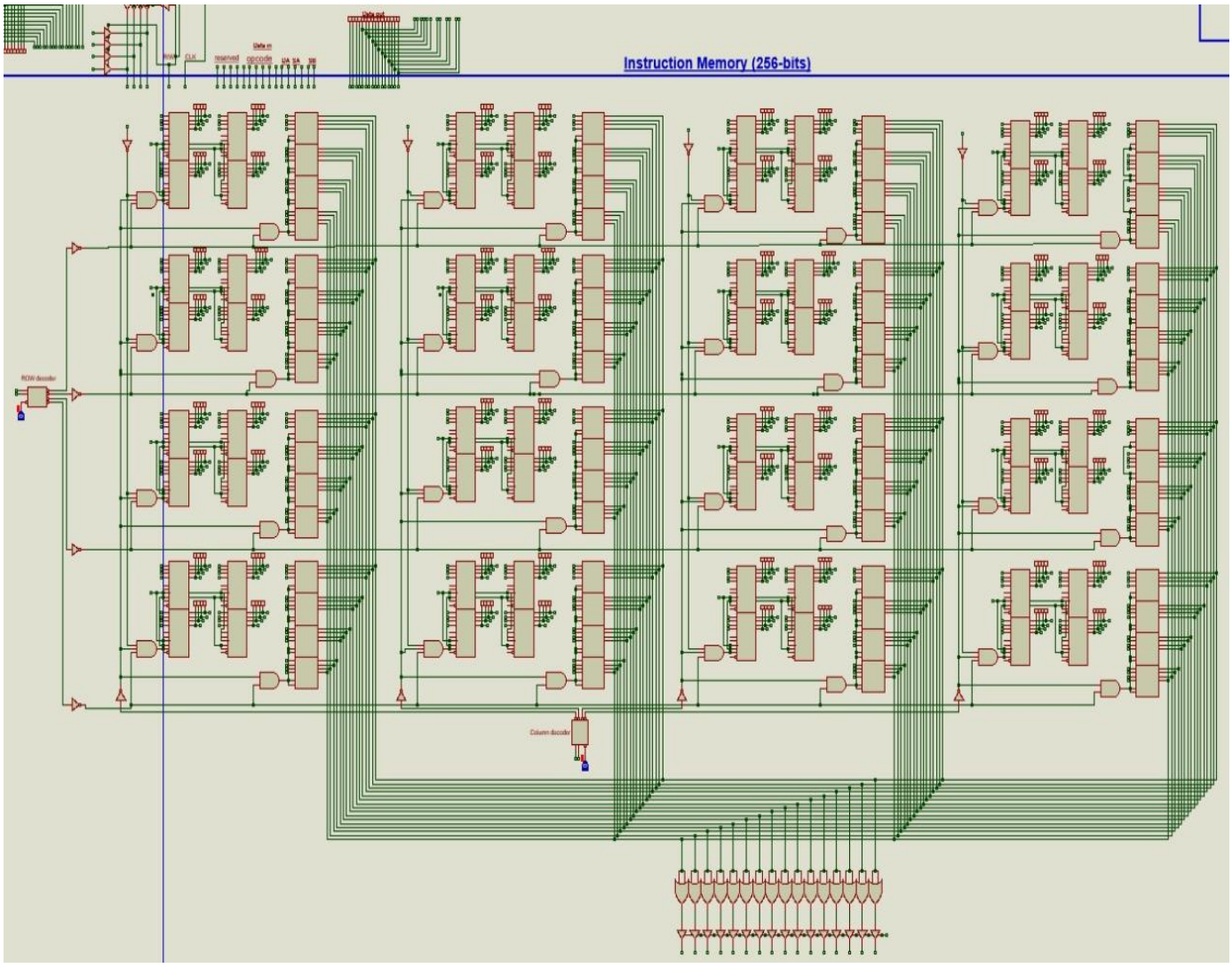
This component holds data that the processor uses for computation or stores results. It is also known as the data cache or the load/store unit. The data memory is usually organized as a random-access memory (RAM) that can be read or written by the processor. The data memory is accessed by the ALU, which provides the address and the data to be read or written.
Each cell is 4-bits , size is 2^4 x 4 .



Instruction memory :

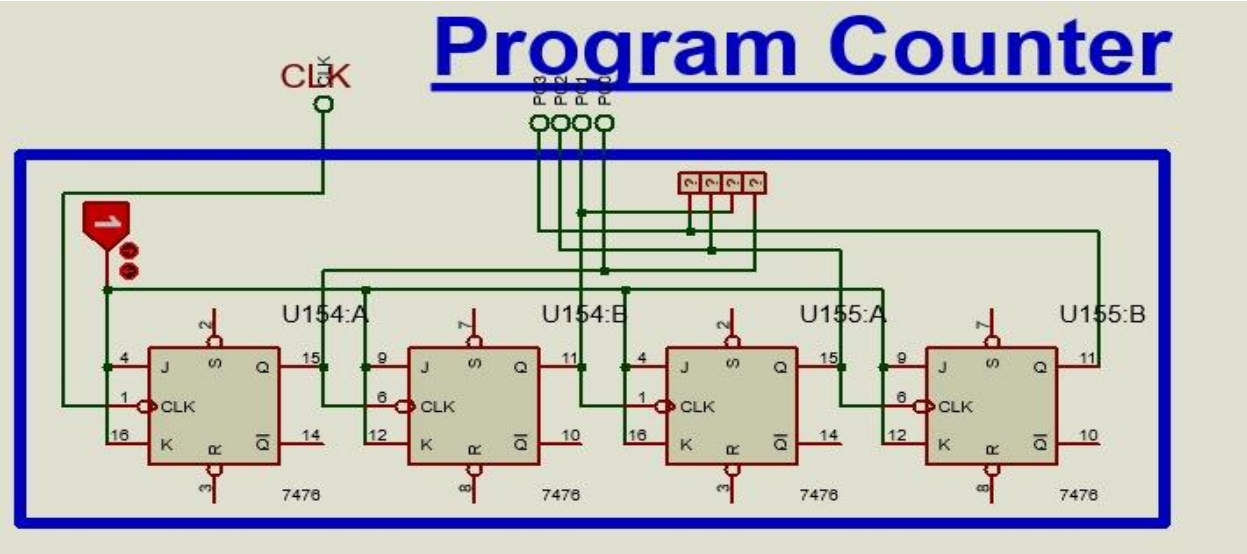
This component stores the program instructions that the processor fetches and executes. It is also known as the instruction cache or the instruction fetch unit. The instruction memory is usually organized as a sequential array of words, each containing an instruction. The instruction memory is accessed by the PC, which provides the address of the next instruction to be fetched.

Each cell is 16-bits , Size is 2^4 x 16 .



Program Counter(PC) :

This component is a CPU register that holds the address of the next instruction to be executed from memory. It is also known as the instruction pointer, the instruction address register, or the instruction counter. The PC is used to keep track of the current or next instruction in the program sequence. It is usually incremented after fetching an instruction, and can be modified by control transfer instructions such as jumps and branches. The PC works in combination with other registers, such as the instruction register and the memory address register, to identify and execute the current instruction.



Instruction decoder :

The instruction decoder is a component in a computer's CPU that translates a machine language instruction into a set of control signals. These signals direct the operation of the CPU's other components. The instruction decoder is used in the fetch-execute cycle of a computer's operation. The instruction decoder is connected to other components of the CPU, such as the control unit, the register file, and the data memory unit. The instruction decoder uses the instruction format to determine the operation to be performed.

There are three types of the instruction format: -

- Register format.
- Immediate format.
- Jump and Branch(unused in our project).

Register addressing mode

In register addressing mode, the operands exist in those registers that reside within a CPU. In this case, we select a specific register from a certain register field in the given instruction. Each field is capable of determining one register.

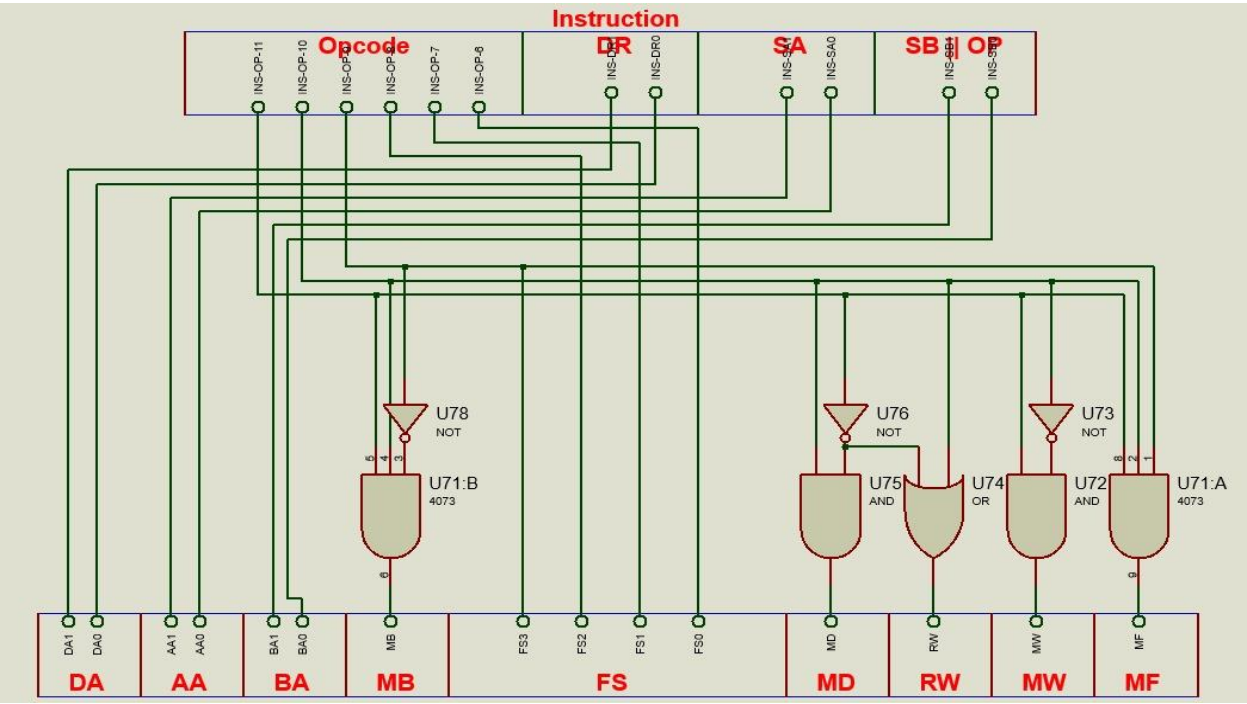
Immediate addressing mode

Immediate addressing mode involves specifying the operand directly in the instruction, rather than referring to a memory address or a register. This mode is commonly used for constants in assembly language programming.

Truth Table for Instruction Decoder Logic

Instruction Function Type	Instruction Bits			Control Word Bits				
	11	10	9	MB	MD	RW	MW	MF
Function-unit operations using registers	0	0	X	0	0	1	0	0
Memory read	0	1	X	0	1	1	0	0
Memory Write	1	0	X	0	X	0	1	0
Function-unit operation using register and constant	1	1	0	1	0	1	0	0
Shift unit	1	1	1	0	0	1	0	1

Circuit design :



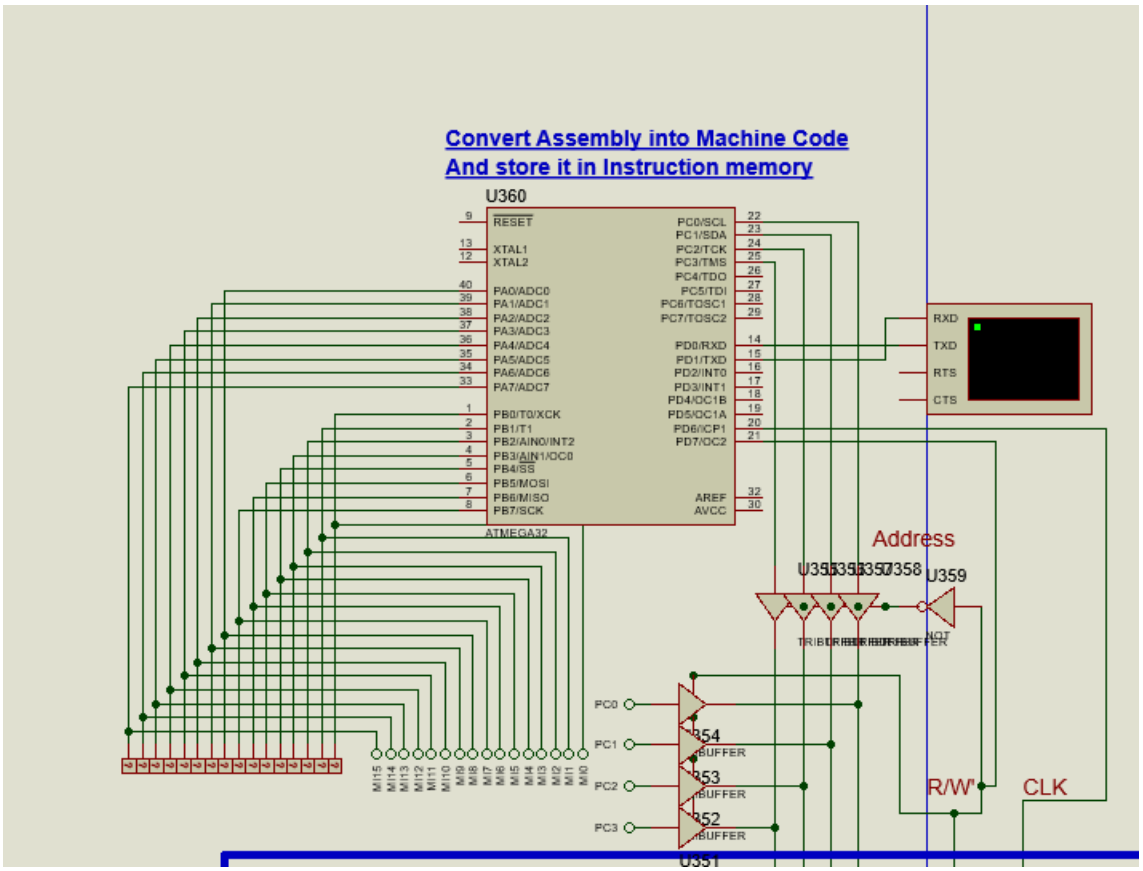
Instruction specification :

Instruction	Opcode	Mnemonic	Format	Description
No operation	000000	NOP	NOP	No operation
Move	000000	MOV	MOV RD,RA	RD←RA
Increment	000001	INC	INC RD,RA	RD←RA+1
Add	000010	ADD	ADD RD,RA,RB	RD←RA+RB
Subtract	000101	SUB	SUB RD,RA,RB	RD←RA-RB
Decrement	000110	DEC	DEC RD,RA	RD←RA-1
AND	00100x	AND	AND RD,RA,RB	RD←RA . RB
OR	00101x	OR	OR RD,RA,RB	RD←RA + RB
Exclusive OR	00110x	XOR	XOR RD,RA,RB	RD←RA ⊕ RB
NOT	00111x	NOT	NOT RD,RA	RD←$\overline{RA}$
Shift Left	111001	SHL	SHL RD,RA	RD←RA<<1
Load Immediate	110010	LDI	LDI RD,OP	RD←Operand
Add Immediate	110010	ADI	ADI RD,RA,OP	RD←RA+Operand
Subtract Immediate	110101	SUBI	SUBI RD,RA,OP	RD←RA-Operand
Load	010000	LD	LD RD,RA	RD←[RA]*
Store	100000	ST	ST RD,RA	[RD]*←RA

Assembler :

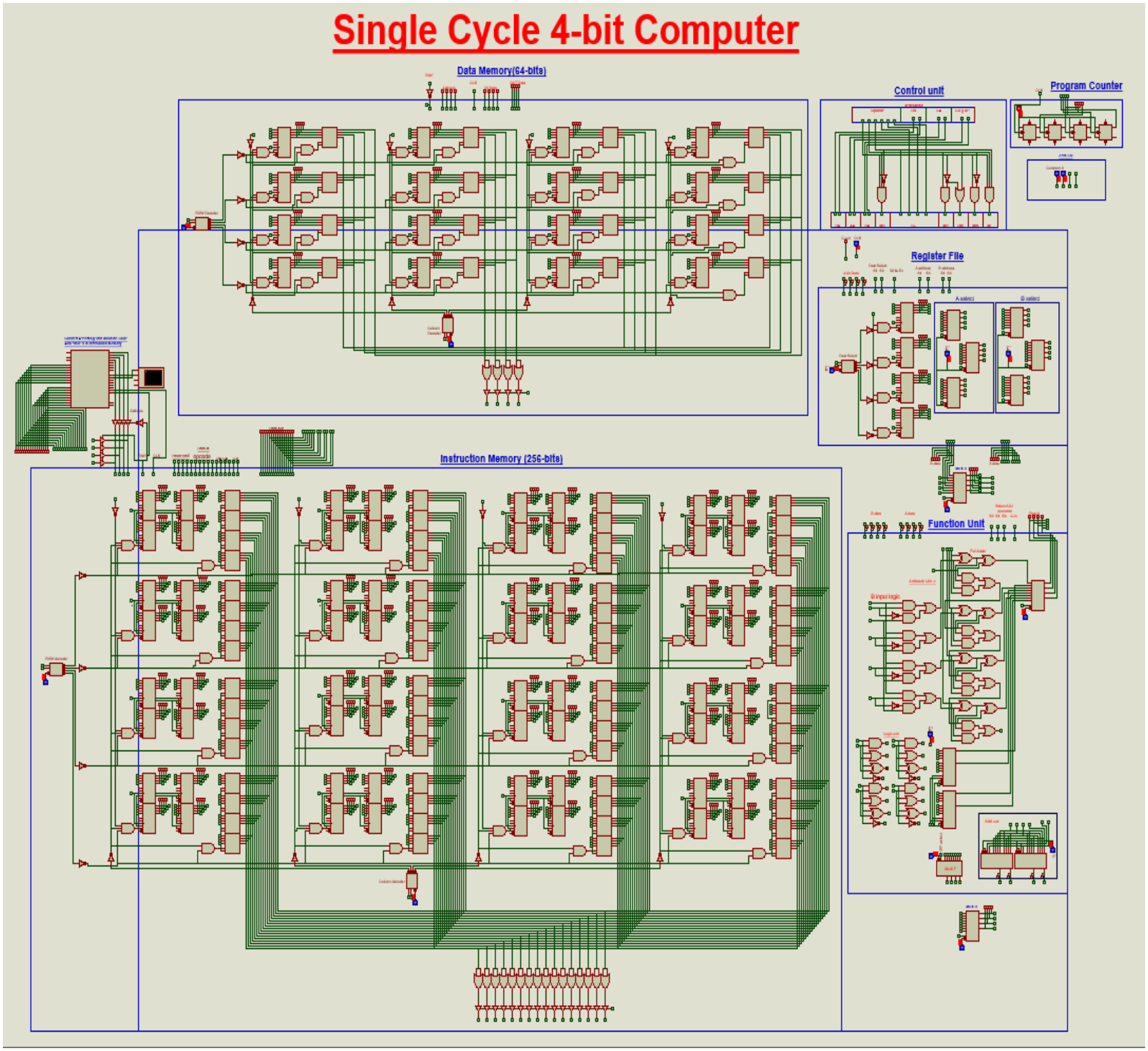
An assembler is a type of programming language translator that converts low-level assembly language code into machine code. Assembly language is a human-readable representation of the machine code instructions that a computer's central processing unit (CPU) can execute.

We used an atmega32 microcontroller to convert assembly instructions into machine code .



This microcontroller takes instructions, converts them into machine code, and stores them in instruction memory.

Full design :


Single Cycle 4-bit Computer
Data Memory(64-bits)
Control unit
Program Counter
Register File
Instruction Memory (256-bits)
Function Unit

Materials URL

https://drive.google.com/drive/folders/1svsPUgppsMZilctH2uqK962hW9mnTqDB?usp=drive_link