COLLEGE OF ENGINEERING

FACULTY OF ELECTRICAL AND COMPUTER ENGINEERING

DEPARTMENT OF ELECTRICAL ENGINEERING

COE 381: INTRODUCTION TO MICROPROCESSORS

GROUP PROJECT REPORT

DESIGNING A 16 BIT MICROPROCESSOR

MARCH 2021

GROUP MEMBERS

SAMUEL DANIEL UDEH	3558718
KINGSLEY OWUSU SARKODIE	3558818
TAKYI PRINCE KWAKU	3559318
NATHANIEL AWONU OFORI	3550118
APPIAGYEI EUGENE ADOM	3548818
JAMES ASIAMAH	3549818

ABSTRACT

Microprocessor is any of a type of a miniature electronic device that contains the arithmetic, logic and control circuiting necessary to perform the functions of a digital computer's CPU. It has ALU, control unit, registers, bus systems and clock to perform computational tasks.

It's also a programmable device that takes in numbers, performs on them arithmetic or logic operations according to the program stored in memory and then produces other numbers as result.

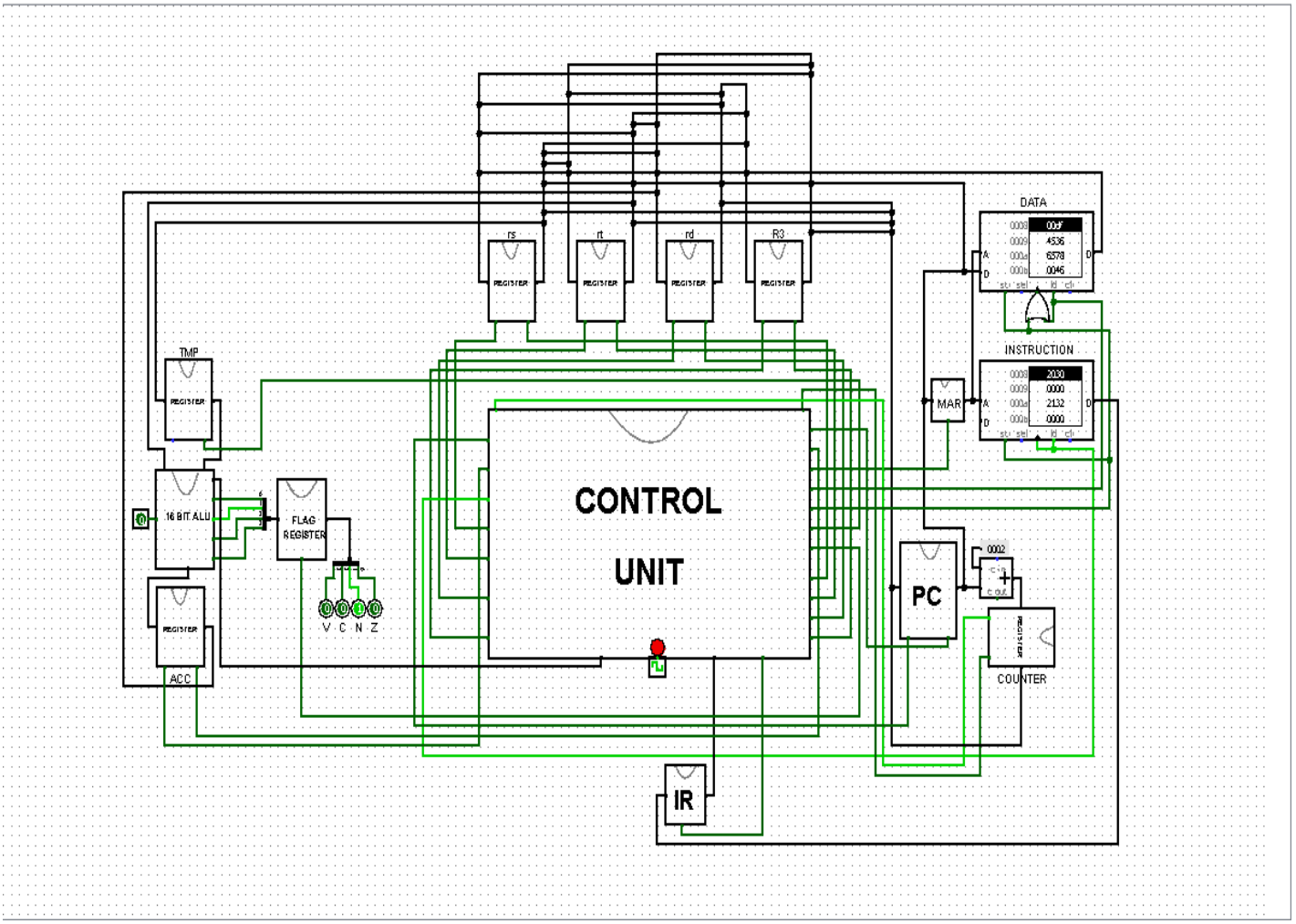
The microprocessor follows a sequence or a pattern in order to execute an instruction this is called the instructions cycle. Firstly, it fetches the information from memory, decodes it and finally executes it. A microprocessor is termed as either 4-bit, 8-bit, 16-bit, 32-bit or 16-bit based on the number of bits of data it can work on at a time. A 16-bit microprocessor is a computer processor that can process 16 bits of data in one clock cycle.

ABOUT OUR 16-BIT MICROPROCESSOR

Our microprocessor has registers, the control unit and the Arithmetic and Logic Unit with two memories (RAM) – this is our **data path**. Our 16-bit microprocessor has 16 address bits as its memory is also 16 bit allowing for 2^{16} (65536) memory locations which can hold 16-bits of data. The size of an instruction is 16-bit wide. The design of our microprocessor was based on the Harvard Architecture, which has the data memory separated from the instruction memory. This processor can fetch information from memory given an instruction. It can decode and execute an instruction. Some of the instructions that our processor can execute are LOAD, STORE, arithmetic and logic operations, jump instructions and move data between registers.

However, there are certain operation our processor cannot execute; operations involving fractions or decimal.

The general view of our 16bit microprocessor is showed below:



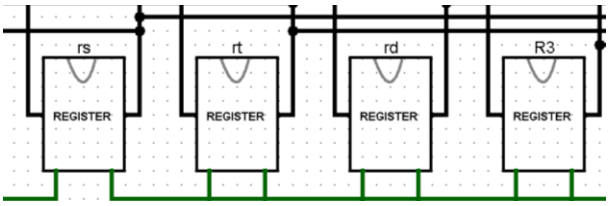
Going into the components of our processor…

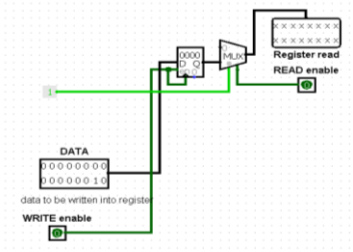
The Registers

Our microprocessor is limited to only 4 general-purpose registers, 1 temporary register, the accumulator, the flag or status register, the memory address register (MAR), the instruction register (IR) and the program counter (PC) or the instruction address register (IAR). Let us look at what each of them do.

❖ *The general purpose registers*

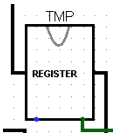
These registers do a little more work than any other registers. They hold operands for operation, they can also hold address where data can be loaded from or stored into, in the data memory.





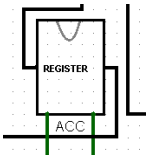
❖ *The temporary register*

In our design, this register hold one of the operands during an ALU operation.



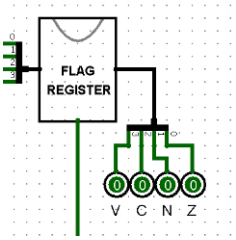
❖ *The accumulator*

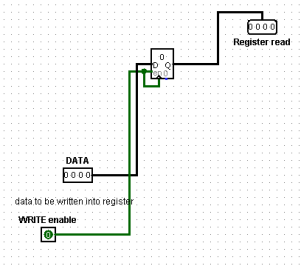
Here in our design, the accumulator holds the result or the output of our ALU operations.



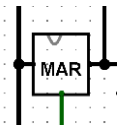
❖ *The flag or status register*

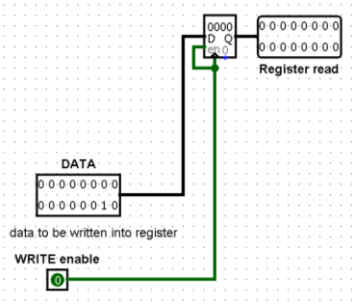
This register hold the flags from our ALU operations. Flags such as CARRY (C), OVERFLOW (V), NEGATIVE (N) and ZERO (Z).



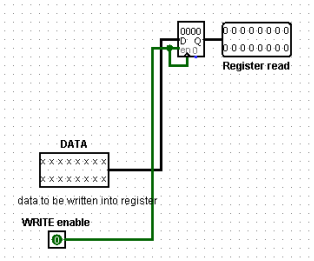


❖ *The memory address register*
This register holds the address of any data or instruction to be fetched from memory.

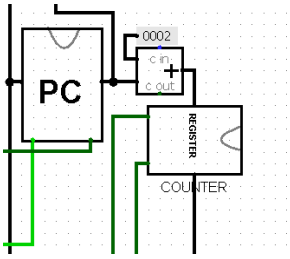


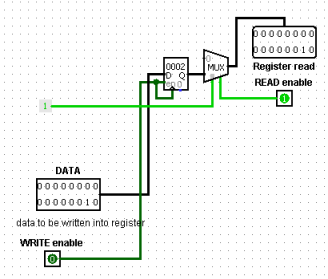


❖ *The instruction register*
The instruction register holds the current instruction being executed by the processor.



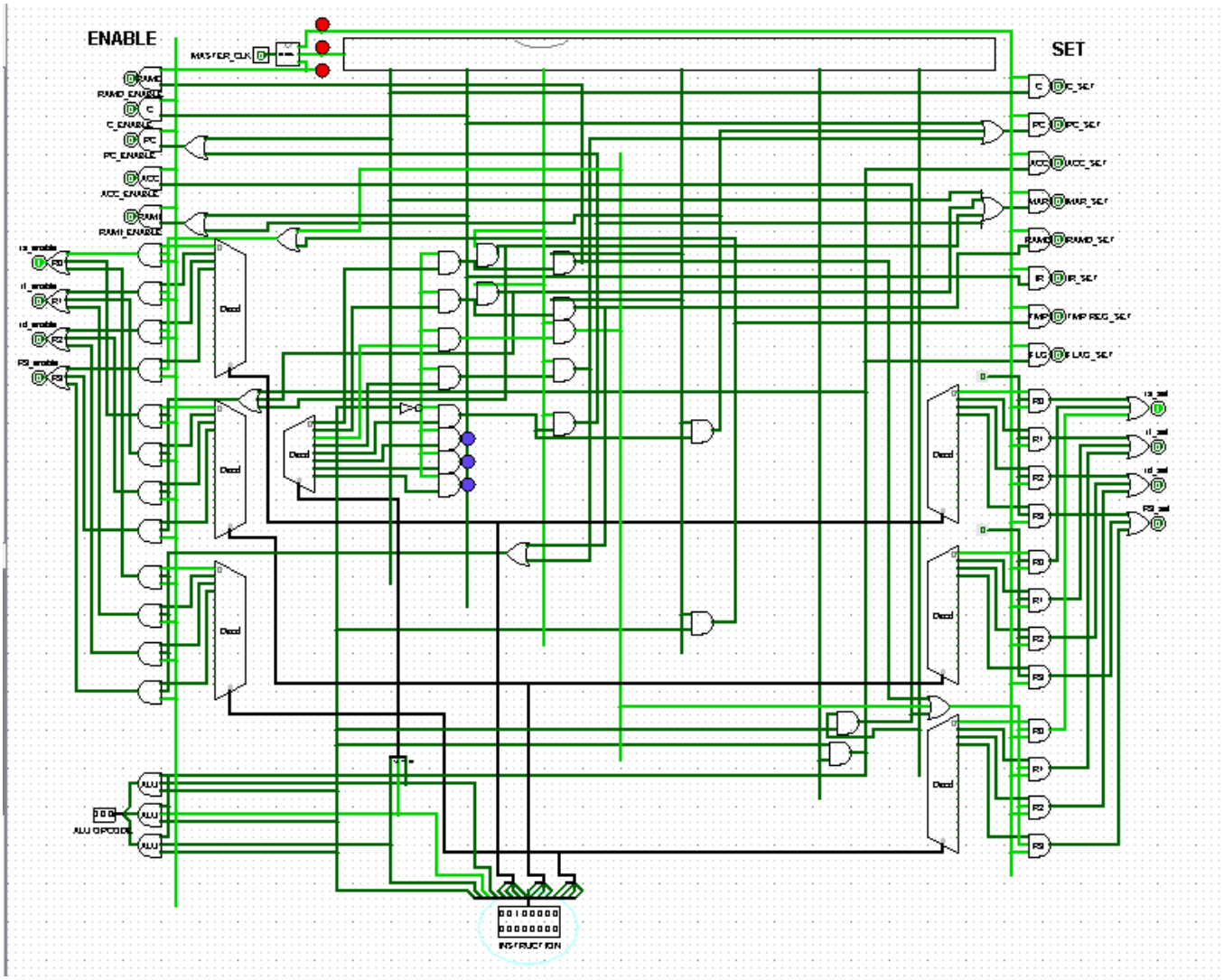
❖ *The program counter*
This register holds the address of the next instruction to be executed.





The Control Unit

Our control unit has a decoder which decodes the instruction available in the instruction register and sends signals (SET or ENABLE) to the respective components of the processor including the RAM for an instruction to be executed. The pictorial view of our control unit is shown below:

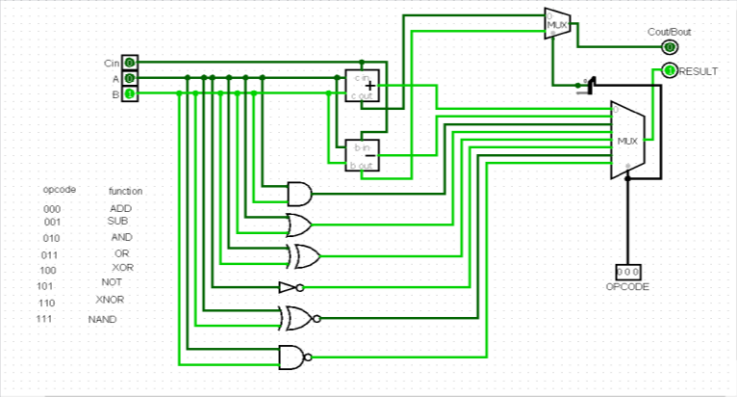


The Arithmetic and logic Unit (ALU)

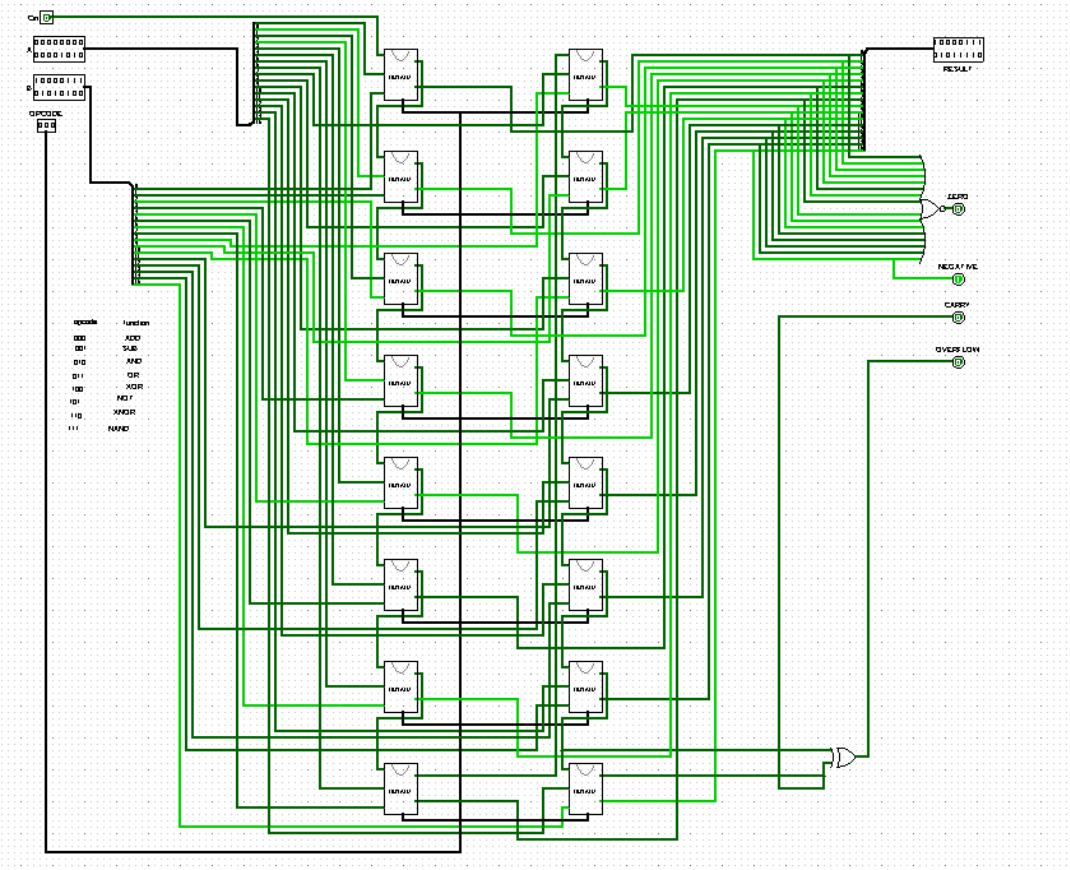
The **ALU** is a digital circuit that performs arithmetic functions like add, subtract and also logic operations AND, OR, XOR to mention just a few.

Below are the images of our 1-bit and 16-bit ALU:

1-bit ALU



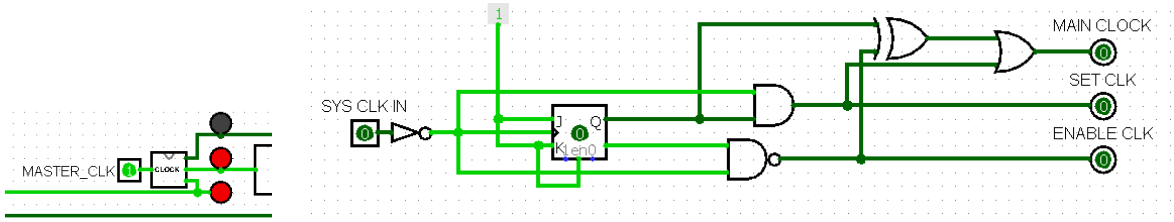
16-bit ALU



The Clock

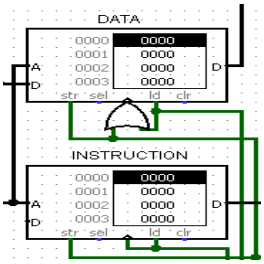
The speed at which the processor can carry out instructions is called the clock speed. This is controlled by a clock. With every tick of the clock, the CPU fetches and executes one instruction. The clock speed is measured in cycles per second,

and one cycle per second is known as 1 Hertz. Our processor uses a maximum of 6 clock cycles to execute an instruction. This clock is used by the control unit to either SET or ENABLE a component. SET CLK is the SET signal and ENABLE CLK is the ENABLE signal.



The Memory

Since our processor uses the Harvard Architecture, we have two memories; the data memory and the instruction memory. The data memory stores the data to be used for operations as the instruction memory stores the instructions to be executed by the processor. Both memories are 16-bits having 16-bit address and 65536 memory locations each.



Instruction Set Architecture (ISA)

Each processor chip implements and supports a set of binary commands that it can decode and execute. This set of commands is hard-wired into the microprocessor's circuitry during its manufacturing process. This set of commands implemented by the processor is called the Instruction Set Architecture. The ISA forms a boundary between the hardware and software.

REGISTER NAME	USAGE	VALUE ASSIGNED
rs	General purpose	0000
rt	General purpose	0001
rd	General purpose	0010
R3	General purpose	0011

INSTRUCTION TYPE	INSTRUCTION	OPCODE
Arithmetic	ADD	1000
	SUB	1001
Logic	AND	1010
	OR	1011
	XOR	1100
	NOT	1101
	XNOR	1110
	NAND	1111
Data transfer	LOAD	0000
	STORE	0001
	MOVE	0010
Jump	JMP	0100
	JMPR	0011

For any instruction type, the first set of 4 bits represents the opcode. Example, **1000** xxxx xxxx xxxx which means addition.

For arithmetic and logic instruction, the second, third and last sets of 4 bits are register designations. The second and third sets of 4 bits are used as operands and the last set of 4 bits is used as destination register. Example, **1000 0000 0001 0010** (**8012** in hex) means add content in rs (0000) to rt (0001) and store result in rd (0010).

For data transfer instruction, the second set of 4 bits is not used. Example, 0001 **xxxx** xxxx xxxx.

For LOAD instruction, the third set of 4 bits contains the address of the memory location of data to load into the desired register. The last set of 4 bits is the desired

register. Example, **0000 0000 0000 0011** (**0003** in hex) means load data from memory into R3 (0011) using the address contained in rs (0000).

For STORE instruction, the third set of 4 bits contains the address of the memory location where the data in the desired register is to be stored. The last set of 4 bits is the desired register. Example, **0001 0000 0001 0000** (**1010** in hex) means store data in rs (0000) into memory using the address contained in rt (0001).

For MOVE instruction, the third set of 4 bits refers to the register which has the data you want to move. The last set of 4 bits refers to the register where you want to move the data into. Example, **0010 0000 0011 0010** (**2032** in hex) means move content in R3 (0011) into rt (0010).

For the JUMP instructions, JMP means jump to the next address in the program counter. JMPR means jump to the instruction having the address contained in a particular register.

For JMP, the second, third and last sets of 4 bits are not used. Example, **0100 0000 0000 0000** (**4000** in hex) means jump to the next instruction in the program counter.

For JMPR, the second and third sets of 4 bits are not used but the last set of 4 bits refers to the register which has the address of the instruction to jump to. Example, **0011 0000 0000 0011** (**3003** in hex) means jump to the instruction have the address contained in R3 (0011).

References

- *But How Do It Know by J. Clark Scott*
- *YouTube videos*