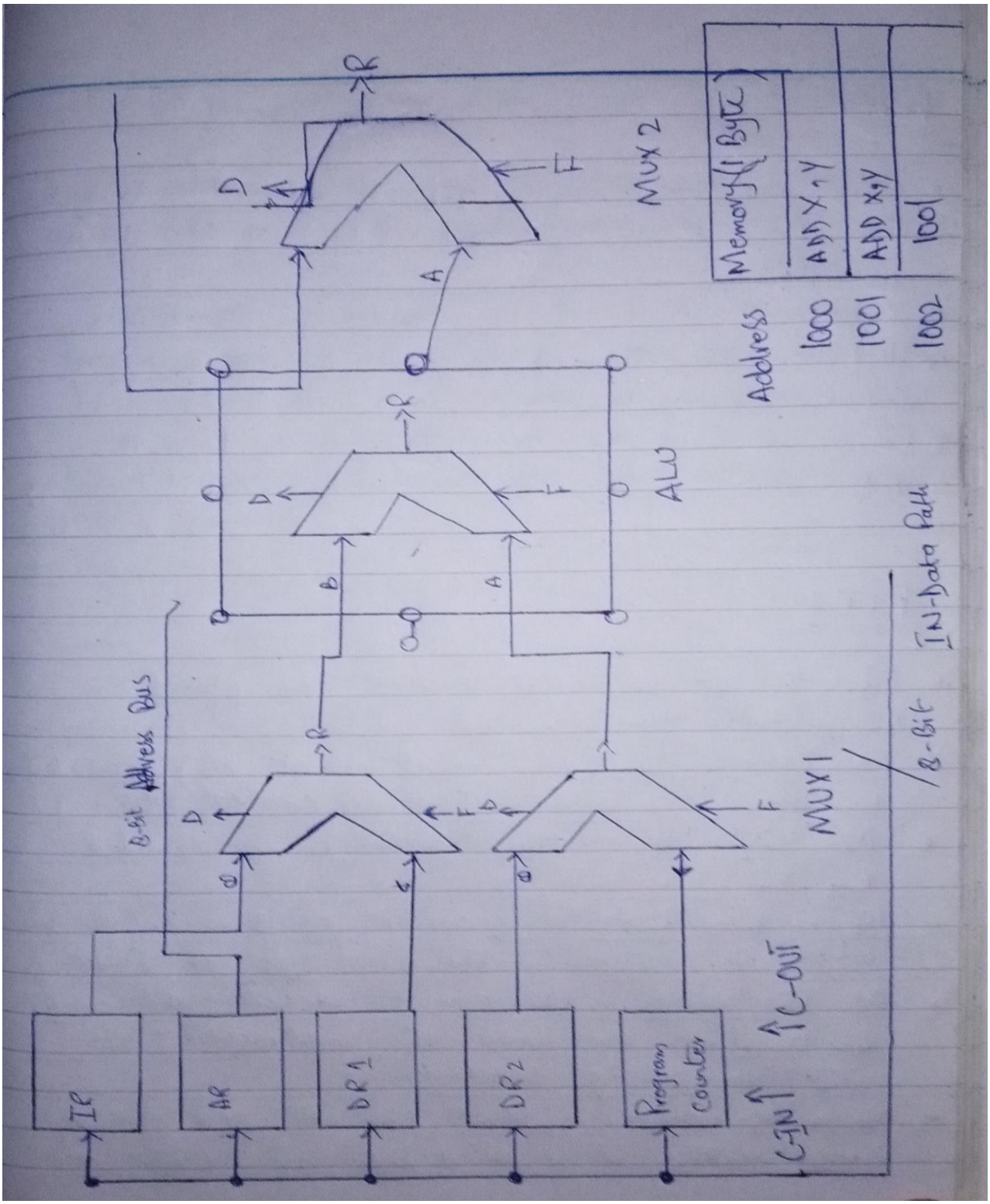
Q1. Differentiate between different generations of computer (in terms of performance and architecture).

Differences of	1st Generation Computer	2nd Generation Computer	3rd Generation Computer	4th Generation Computer	5th Generation Computer
Architecture	The first-generation computer used **Vacuum tubes** for circuitry and magnetic drums for memory and was huge in size. They were very expensive to operate and in addition to using a great deal of electricity, generated a lot of heat, which was often the cause of malfunctions.	**Transistors** replaced vacuum tubes and accompanied in the second generation of computers. Predecessors. Though the transistor still generated a great deal of heat that subjected the computer to damage, it was a vast improvement over the vacuum tube. Second-generation computers still relied on	The development of the integrated circuit was the hallmark of the third generation of computers. Transistors were miniaturized and placed on **silicon chips**, called semiconductors , which drastically increased the speed and efficiency of computers.	The **microprocessor** brought the fourth generation of computers, as thousands of integrated circuits were built onto a single silicon chip. The Intel 4004 chip, developed in 1971, located all the components of the computer—from the central processing unit and memory to input/output controls—on a single chip.	Fifth generation computing devices, based on **artificial intelligence**, are still in development, though there are some applications, such as voice recognition, that are being used today.
Performance	These computers could only solve one problem at a time.	These computers were smaller, faster, cheaper, more energy-efficient and more reliable than their first-generation.	These computers were smaller and cheaper than their forerunner.	These computers became more powerful.	These computers are super fastest in speed and very small in size.

Q2. Draw and describe functionality of data path architecture?



FETCH:-

* Since we have assumed that the instruction is 2-bytes then the instruction register (IR) should also be 2-bytes.
* X, when we assumed is already in the data register 1.
Now how the CPU know to fetch the ADD instruction from memory locations 1000 and 1001?
* Pc must hold the memory address 1000 to fetch first byte of the ADD i instruction.

FETCH:-

* First the Pc value 1000 must be placed on the address bus via AR.
* Then Read control signal must be generated by the CPU and placed on control bus.
* At same time Pc counter is incremented by 1.
* Next RAM puts the content of 1000 on the bus which is stored in the first byte of the IR.
* At this stage CPU cannot decode the complete instruction because the instruction is not completely fetched.
* Now question is that how the CPU will know whether it is 1 byte or 2 bytes or

s bytes instruction?

* The first byte of the instruction must indicate that there is another byte to be fetched as well or in other words it is a 2 bytes instruction.

FETCH:-

* The instruction has got two parts OP-code and Operand.
* The first byte of the instruction must consist of op-code part and this op-code must indicate the size of the instruction too.
* So after getting the first byte of the instruction the CPU decodes the first byte of the instruction to know if there is another byte to be fetched.
* In our example the instruction consists of two bytes, so after decoding the first byte the CPU knows that there is another byte to fetched too.

So all the same steps of fetching the first byte will be repeated to fetch the second byte of the instruction from memory location 1001.

* The second byte of instruction will be stored in second byte of IR

Decode:-

* Now Complete the decoding of the instruction takes place.
* After decoding the instruction the CPU knows the location of x & y.
* Location of x ~~&~~ y is DR1 and y is 1002.
* Now CPU will fetch data (not instruction) from 1002.
* The Address of y will be placed on address bus via AR.
* Again the Read signal will be sent by CPU to the memory via control bus (Pc is not recommended this time)
* Memory will send the contents of 1002 to CPU via data bus which is stored in DR2 (not IR).
* Now ~~PC~~ CPU has values to be added in DR1 & DR2 (x & y respectively)

Execute:

* Content of DR1 is to content of DR2 and result is stored in to DR1.

Q3. Brief the following:

- **Indirect Addressing**
- **Register Addressing**
- **Machine Cycle**
- **State and clock cycle**

❖ Indirect Addressing

In Indirect addressing, address field in the instruction carries the memory location or register where productive address of operand is present. It requires two memory access. It is further classified into two categories: Register Indirect, and Memory Indirect.
Example:
LOAD R1, @500
Above-mentioned instruction is used to load the data of memory location stored at memory location 500 to register R1. In other words, we can say, effective address is stored at memory location 500.

❖ Register Addressing

Every instruction incorporates operands, the operands can be a memory location, a processor register or an I/O device. The instruction which uses processor **registers** to express operands are the instruction in **register addressing**.

Here the effective address is a register where the value of the operand is present.

EA=R

Below we have two instructions as our examples for register addressing.

Add R4, R3

Load R3, R2

In the examples above, the Add instruction uses registers to represent both of its operands. Likewise, the Load instruction also uses registers to represent both of its operands. So, the instruction above uses register addressing to describe the address of the operand.

❖ Machine Cycle

Machine cycle is the term of time period consumption of processor to fetch data or instruction from memory. The machine cycle is a four-process cycle that includes reading interpreting the machine language, executing the code and then storing that code.

1. **Fetch** - Retrieve an instruction from the memory.

2. **Decode** - Translate the retrieved instruction into a series of computer commands.
3. **Execute** - Execute the computer commands.
4. **Store** - Send and write the results back in memory.

❖ State and clock cycle

During a state, the processor performs one or a set of simultaneous micro-operations as determined by the control signals

A clock cycle, or simply a cycle is a single electronic pulse of a CPU. During each cycle, a CPU can perform a basic operation such as fetching an instruction, accessing memory or writing data. Since only simple commands can be performed during each cycle, most CPU processes require multiple clock cycles.