Instructions and Micro-operations:

Instructions typically refer to a CPU command or trigger to perform particular operations / functions.
Whereas, micro-operations are detailed low-level instructions used in some designs, to implement complex machine instructions.

ISA

- Stands for Instruction Set Architecture.
- ISA is a part of an abstract model of a computer that defines how the CPU is controlled by the software. The ISA acts as an interface between the hardware and software, specifying both what the processor is capable of doing as well as how it gets done.

Differences between memory mapped I/O and isolated I/O

Isolated I/O	Memory Mapped I/O
Memory and I/O have separate address space	Both have same address space
All address can be used by the memory	Due to addition of I/O addressable memory become less for memory

Separate instruction control read and write operation in I/O and Memory	Same instructions can control both I/O and Memory
In this I/O address are called ports.	Normal memory address are for both
More efficient due to separate buses	Lesser efficient
Larger in size due to more buses	Smaller in size
It is complex due to separate separate logic is used to control both.	Simpler logic is used as I/O is also treated as memory only.

VHDL program for **The Boolean equation A + B'C + A'C + BC'**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity boolean_equ is
   port ( a,b,c : in std_logic;
         y: out std_logic
         );
end boolean_equ;

architecture bool_equ_arch of boolean_equ is
```

```
begin
  y <= a or (not(b) and c) or (not(a) and c) or (not(c) and b);
end bool_equ_arch;
```

Wallace Tree;
A digital circuit that multiplies two integers.

RTL

- Stands for Register Transfer Language.
- It is used to describe the micro-operations transfer among registers. It is a kind of intermediate representation (IR) that is very close to assembly language.The term “Register Transfer” can perform micro-operations and transfer the result of operation to the same or other register.

Eight Shift operations of RTL
Example : X = 110001010001

a. Logical Shift
    - Logical Shift Left : Each bit in the register is shifted to the left one by one in this shift micro-operation. The most significant bit (MSB) is moved outside the register, and the place of the least significant bit (LSB) is filled with 0.

(MSB) 1 1 0 0 0 1 0 1 0 0 0 1 (LSB)

1 0 0 0 1 0 1 0 0 0 1 0 (<= 0)

- Logical Shift Right: Each bit in the register is shifted to the right one by one in this shift micro-operation. The least significant bit (LSB) is moved outside the register, and the place of the most significant bit (MSB) is filled with 0.

(MSB) 1 1 0 0 0 1 0 1 0 0 0 1 (LSB)

0 1 1 0 0 0 1 0 1 0 0 0

b. Arithmetic Shift

- Arithmetic Shift Left : The arithmetic shift left micro-operation is the same as the logical shift left micro-operation. Each bit in the register is shifted to the left one by one in this shift micro-operation. The most significant bit (MSB) is moved outside the register, and the place of the least significant bit (LSB) is filled with 0.

(MSB) 1 1 0 0 0 1 0 1 0 0 0 1 (LSB)

1 0 0 0 1 0 1 0 0 0 1 0 (<= 0)

- Arithmetic Shift Right : Each bit in the register is shifted to the right one by one in this shift micro-operation. The least significant bit (LSB) is moved outside the register, and the place of the most significant bit (MSB) is filled with the previous value of MSB.

(MSB) 1 1 0 0 0 1 0 1 0 0 0 1 (LSB)

1 1 1 0 0 0 1 0 1 0 0 0

c. Circular Shift.

- Circular Shift Left : Each bit in the register is shifted to the left one by one in this shift micro-operation. After shifting, the least significant bit (LSB) place becomes empty, so it is filled with the value at the most significant bit (MSB).

(MSB) 1 1 0 0 0 1 0 1 0 0 0 1 (LSB)

1 0 0 0 1 0 1 0 0 0 1 1

- Circular Shift Right : Each bit in the register is shifted to the right one by one in this shift micro-operation. After shifting, the most significant bit (MSB) place becomes empty, so it is filled with the value at the least significant bit (LSB).

(MSB) 1 1 0 0 0 1 0 1 0 0 0 1 (LSB)

1 1 1 0 0 0 1 0 1 0 0 0

d. Decimal Shift

The decimal shift was developed specifically for BCD representation. It acts like a linear shift, except it shifts one digit, or four bits, instead of just 1 bit.
For X= 1001 0111, the decimal shift left results in the value 0111 0000, and the decimal shift right produces the value 0000 1001.

Programmed I/O

Programmed I/O is one of the three techniques we use for I/O transfer. The other two methods for the same are *interrupted I/O* and (*direct memory access*) DMA.
Programmed I/O is a technique or approach that we use to transfer data between the processor and the I/O module.

If we talk of programmed I/O and interrupted I/O, it is the responsibility of the *processor* to control the transfer from I/O to main memory as input and from main memory to I/O as output. On the other way, the DMA does not involve a processor, the main memory and I/O exchange their data directly.

What is the Need?

To understand the need and functioning of the programmed I/O consider a scenario of human and computer interaction where the human types something on the keyboard. Now the processor has to save this data into the memory and has to display the same data on the display device.

With the programmed I/O this entire functioning is regulated with the help of a program. This action of transferring the typed characters from the keyboard to memory and then to the display module must happen at the right time.

Let us take this in more detail the input from the keyboard is accepted when the key is being pressed on the keyboard by the user. And the output can be sent to the display module only when the module is ready to accept it.

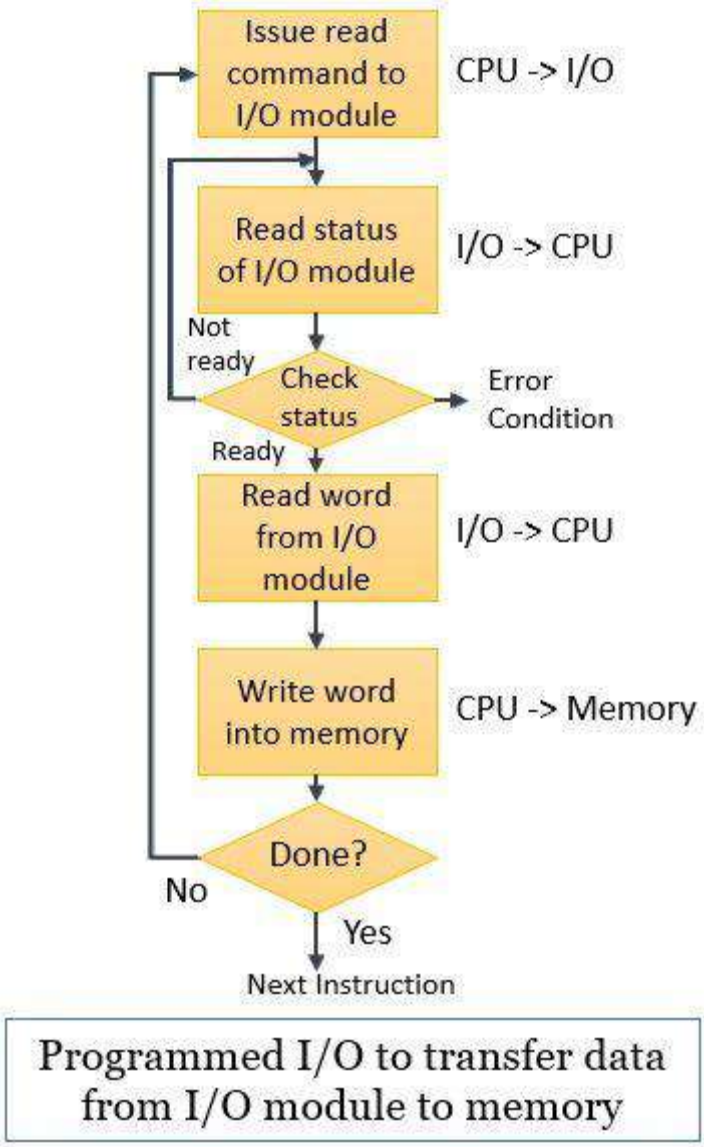
Here the speed at which the data is transferred from the keyboard to the computer memory depends on the typing speed of the computer. On the other hand, the speed at which the output data from memory is transferred to the display module is very high.

The speed at which the character can be transmitted and displayed on the display module is still much slower if compared to the speed of the processor. So, to overcome the difference in the speed of the processor and I/O device a mechanism must be implemented to synchronize the transfer of data between the processor and I/O modules. This is we require programmed I/O

Functioning of programmed I/O

Consider the situation that the processor is busy executing any program. Meanwhile, it encounters an *I/O instruction*. To execute the encountered instruction the process supply an appropriate *I/O command* to the corresponding I/O module. Accepting the issued command, the I/O module performs the desired task and sets some appropriate bits of its I/O status registers. As we have seen in our previous content, the bus structure that each I/O interface has a set of registers.

Further, the I/O module does not notify the processor that it has performed the desired task. Moreover, it's the processors' responsibility to periodically check the status of the I/O module till it finds that the I/O has successfully completed the desired task.



Programmed I/O to transfer data from I/O module to memory

If you have observed the function of the programmed I/O it involves two things the I/O command that is provided by the processor to the I/O module and the I/O instruction that is encountered and executed by the processor.

Cache Mapping Notes:

https://byjus.com/gate/cache-mapping-notes/

Booth Algorithm VIdeos:

https://www.youtube.com/watch?v=DHhcnjEKEFo

https://www.youtube.com/watch?v=cWfaw7b3jKY

Paging vs Segmentation.

Paging is a memory management technique in which process address space is broken into blocks of the same size called pages and placed in the main memory.

Segmentation is a memory management technique in which each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions.

Sr. No.	Key	Paging	Segmentation
1	Memory Size	In Paging, a process address space is broken into fixed sized blocks called pages.	In Segmentation, a process address space is broken in varying sized blocks called sections.
2	Accountability	Operating System divides the memory into pages.	Compiler is responsible to calculate the segment size, the virtual address and actual address.
3	Size	Page size is determined by available memory.	Section size is determined by the user.

4	Speed	Paging technique is faster in terms of memory access.	Segmentation is slower than paging.
5	Fragmentation	Paging can cause internal fragmentation as some pages may go underutilized.	Segmentation can cause external fragmentation as some memory block may not be used at all.
6	Logical Address	During paging, a logical address is divided into page number and page offset.	During segmentation, a logical address is divided into section number and section offset.
7	Table	During paging, a logical address is divided into page number and page offset.	During segmentation, a logical address is divided into section number and section offset.
8	Data Storage	Page table stores the page data.	Segmentation table stores the segmentation data.

Method for solving Cache Coherence Problem:

Video url : https://www.youtube.com/watch?v=_qdHpUdTPBM

1. Write Update-Write through.
2. Write Update-Write back.
3. Write Invalidate - Write through
4. Write Invalidate - Write back