**Model Question Paper of Computer Architecture**

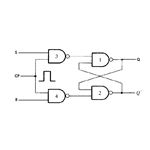
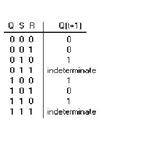
**MCA Semester I**

1. **What are filp flops? Explain RS Flipflop?**

A flip-flop is a type of circuit that has two states (i.e. on or off, 1 or 0) and are often used to store state information. By sending a signal to the flip-flop, the state can be changed. Flip-flops are used in a number of electronics, including computers and communications equipment.

**RS Flip Flop-**

The RS flip-flop consists of basic flip-flop circuit along with two additional [NAND gates](http://www.brighthubengineering.com/diy-electronics-devices/3595-and-nand-and-not-logic-gates-explained/) and a clock pulse generator. The clock pulse acts as an enable signal for the two inputs. The output of the gates 3 and 4 remains at logic “1" until the clock pulse input is at 0.This is nothing but the quiescent condition of the flip-flop.

[](http://img.bhs4.com/80/6/806660E43C2343613310F431E2FD4CCECB1C866A_large.jpg)[](http://img.bhs4.com/20/9/20950C750A70FC83D2BCDB3FBE6905EF1203828D_large.jpg)

Information from S and R is allowed to reach the output only when clock pulse goes to 1.

Let’s assume S=1, R=0 and CP=1. The set state is reached at this condition and since the clock pulse is 1, information from S and R is allowed to reach output.

From the truth table of NAND gate we can say that the output is 0 only when both the inputs are 1. In all the other case the output is 1.

So when S=1, R=0 &CP=1. Both the inputs to the gate 3 are 1 and hence its output is 0. This information (i.e.) 0 is passed to gate 1. Since one of the inputs of gate 1 is 0, we can say that the output Q=1. Since R=0, the output obtained at Q’=0.

1. **Explain the following addressing mode-  
   1) Register 2) Immediate  
   3) Direct 4) Register Indirect**

Addressing mode is the way of addressing a memory location in instruction. Microcontroller needs data or operands on which the operation is to be performed. The method of specifying source of operand and output of result in an instruction is known as addressing mode. 

1. **Register Addressing Mode**: In this addressing mode, the source of data or destination of result is Register. In this type of addressing mode the name of the register is given in the instruction where the data to be read or result is to be stored.  
   Example: ADD A, R5 ( The instruction will do the addition of data in Accumulator with data in register R5)
2. **Direct Addressing Mode**: In this type of Addressing Mode, the address of data to be read is directly given in the instruction. In case, for storing result the address given in instruction is used to store the result.  
   Example: MOV A, 46H ( This instruction will move the contents of memory location 46H to Accumulator)
3. **Register Indirect Addressing Mode**: In Register Indirect Addressing Mode, as its name suggests the data is read or stored in register indirectly. That is, we provide the register in the instruction, in which the address of the other register is stored or which points to other register where data is stored or to be stored.  
   Example: MOV A, @R0 ( This instruction will move the data to accumulator from the register whose address is stored in register R0 ).
4. **Immediate Addressing Mode** : In Immediate Addressing Mode , the data immediately follows the instruction. This means that data to be used is already given in the instruction itself.  
   Example: MOV A, #25H ( This instruction will move the data 25H to Accumulator. The # sign shows that preceding term is data, not the address.)
5. **Differentiate between the Cache memory and Virtual memory.**

Virtual memory is an abstraction of the main memory. It extends the available memory of the computer by storing the inactive parts of the content RAM on a disk when the content is required, it fetches it back to the RAM. Cache memory is used to store frequently accessed data in order to quickly access the data whenever it is required. They both are conceptually the same, however they mainly differ in the matter of implementation which results in different aspects like speed and control mechanism.

Comparison between Virtual Memory and Cache Memory:

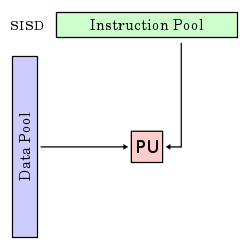
|  |  |  |
| --- | --- | --- |
|  | **Virtual Memory** | **Cache Memory** |
| Definition | Virtual memory is an abstraction of the main memory. It extends the available memory of the computer by storing the inactive parts of the content RAM on a disk. It fetches it back to the RAM when the content is required. | Cache memory is used to store frequently accessed data in order to quickly access the data whenever it is required. They both are conceptually the same; however they mainly differ in the matter of implementation. |
| Purpose | It extends the memory capacity of a computer beyond the one that is installed. | It reduces the amount of time needed to access the data. |
| Speed | It operates in the millisecond range. | It operates in the nanosecond range. |
| Control mechanism | Managed by the operating system | Managed automatically by the hardware |
| Component | It is a part of the hard drive (secondary storage). | Located on the processor itself |

1. **Explain flynn’s classification with suitable examples.**

Flynn's taxonomy is a classification of [computer architectures](https://en.wikipedia.org/wiki/Computer_architecture), proposed by [Michael J. Flynn](https://en.wikipedia.org/wiki/Michael_J._Flynn) in 1966. The classification system has stuck, and has been used as a tool in design of modern processors and their functionalities.

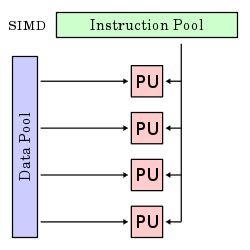
**SISD**

* Single Instruction, Single Data stream
* A sequential computer which exploits no parallelism in either the instruction or data streams. Examples of SISD architecture are the traditional single processor machines like a PC (currently manufactured PC's have multiple processors) or old mainframes.



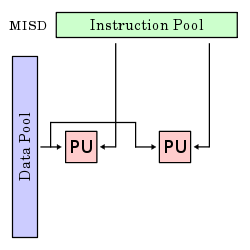
**SIMD**

* Single Instruction, Multiple Data streams
* Exploits multiple data streams against a single instruction stream to perform operations which may be naturally parallelized.
* For example, an array processor or GPU.
* Typical for splitting large data sets.



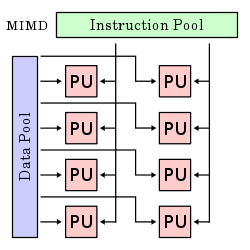
**MISD**

* Multiple Instruction, Single Data stream
* Multiple instructions operate on a single data stream. Uncommon architecture which is generally used for fault tolerance.
* Systolic Array: process data & pass on to next PU
* Examples include the Space Shuttle flight control computer.
* Least common



**MIMD**

* Multiple Instruction, Multiple Data streams
* Multiple autonomous processors simultaneously executing different instructions on different data.
* May send results to central location.



1. **State the limitations of Karnaugh Map?**

The Karnaugh map, also known as the K-map, is a method to simplify [boolean algebra](http://en.wikipedia.org/wiki/Boolean_algebra" \o "Boolean algebra) expressions.

Limitations : The K map does not necessarily "fail" for higher dimensions. The problem is that it is so difficult to visualize for more than five variables. A 4 variable K-map is 2 dimenisional and easy to visualize. A 5 variable is three dimensional, but is still manageable from a visualization standpoint, because the 2 states of the 5th variable only require visually moving from one plane to the next, without moving in the x or y directions of either plane. Just getting equations correct with more than 5 variables is difficult enough using the K map, much less considering an optimum set of terms ("core" prime implicants and "choice" prime implicants).

1. **Write short notes on a) interrupts b) I/O Interface**

**Interrupts**

An interrupt is a signal from a device attached to a computer or from a program within the computer that requires the [operating system](http://searchcio-midmarket.techtarget.com/definition/operating-system) to stop and figure out what to do next. Almost all personal (or larger) computers today are interrupt-driven - that is, they start down the list of computer [instructions](http://searchcio-midmarket.techtarget.com/definition/instruction) in one program (perhaps an application such as a word processor) and keep running the instructions until either (A) they can't go any further or (B) an interrupt signal is sensed. After the interrupt signal is sensed, the computer either resumes running the current program or begins running another program.

Basically, a single computer can perform only one computer instruction at a time. But, because it can be interrupted, it can take turns in which programs or sets of instructions that it performs. This is known as [multitasking](http://searchcio-midmarket.techtarget.com/definition/multitasking). It allows the user to do a number of different things at the same time. The computer simply takes turns managing the programs that the user starts. Of course, the computer operates at speeds that make it seem as though all of the user's tasks are being performed at the same time. (The computer's operating system is good at using little pauses in operations and user think time to work on other programs.)

An operating system usually has some code that is called an interrupt handler. The interrupt handler prioritizes the interrupts and saves them in a [queue](http://searchcio-midmarket.techtarget.com/definition/queue) if more than one is waiting to be handled. The operating system has another little program, sometimes called a [scheduler](http://searchcio-midmarket.techtarget.com/definition/queue), that figures out which program to give control to next.

In general, there are hardware interrupts and software interrupts. A hardware interrupt occurs, for example, when an I/O operation is completed such as reading some data into the computer from a tape drive. A software interrupt occurs when an application program terminates or requests certain services from the operating system. In a personal computer, a hardware interrupt request ([IRQ](http://searchcio-midmarket.techtarget.com/definition/IRQ)) has a value that associates it with a particular device.

**I/O Interface**

Interface is a shared boundary between two separate components of the computer system which can be used to attach two or more components to the system for communication purposes.

There are two types of interface:

1. CPU Inteface
2. I/O Interface

#### Input-Output Interface

Peripherals connected to a computer need special communication links for interfacing with CPU. In computer system, there are special hardware components between the CPU and peripherals to control or manage the input-output transfers. These components are called input-output interface units because they provide communication links between processor bus and peripherals. They provide a method for transferring information between internal system and input-output devices.

### Modes of Transfer

Data transfer between the central unit and I/O devices can be handled in generally three types of modes which are given below:

#### Programmed I/O-

Programmed I/O instructions are the result of I/O instructions written in computer program. Each data item transfer is initiated by the instruction in the program. Usually the program controls data transfer to and from CPU and peripheral. Transferring data under programmed I/O requires constant monitoring of the peripherals by the CPU.

#### Interrupt Initiated I/O-

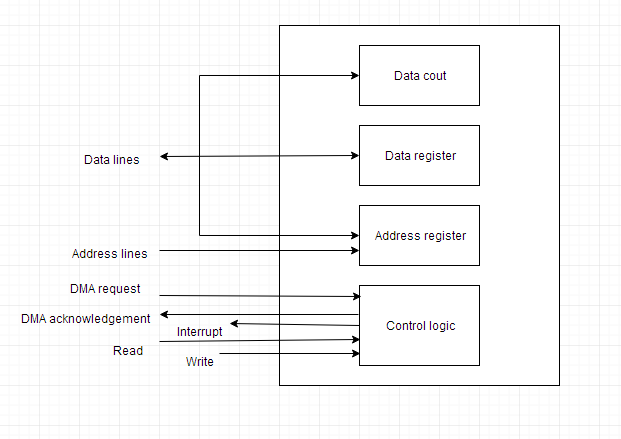
In the programmed I/O method the CPU stays in the program loop until the I/O unit indicates that it is ready for data transfer. This is time consuming process because it keeps the processor busy needlessly. This problem can be overcome by using interrupt initiated I/O. In this when the interface determines that the peripheral is ready for data transfer, it generates an interrupt. After receiving the interrupt signal, the CPU stops the task which it is processing and service the I/O transfer and then returns back to its previous processing task.

#### Direct Memory Access-

Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer. This technique is known as **DMA**.

In this, the interface transfer data to and from the memory through memory bus. A DMA controller manages to transfer data between peripherals and memory unit.

Many hardware systems use DMA such as disk drive controllers, graphic cards, network cards and sound cards etc. It is also used for intra chip data transfer in multicore processors. In DMA, CPU would initiate the transfer, do other operations while the transfer is in progress and receive an interrupt from the DMA controller when the transfer has been completed.



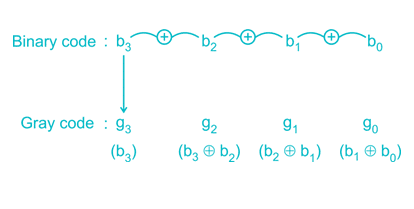
1. **Why does DMA have priority over the CPU when both request a memory transfer ?**

If you have a large block of data to be transferred over a system bus then DMA is called by the CPU. Reason is if CPU monitors this transfer of data block then CPU will be not doing its useful job and this reduces the performance of the system.

For this reason, now CPU gives authority to DMA then DMA will interrupt CPU two times before actual transfer and at end when complete transfer has finished. This allows CPU to do its useful work while transfer is going on. So DMA has to be given more priority then CPU if you don't want performance of the system to reduce.

1. **Write down the method of converting a binary number into its Gray Code equivalent. Also, give an example**

Let Binary code be b3  b2  b1  b0.Then the respective Gray Code can be obtained is as follows

*[](https://testbook.com/blog/wp-content/uploads/2015/11/binary-code-to-gray-code.png)*

i.e.

g3 = b3

g2 = b3 ⊕ b2

g1 = b2 ⊕ b1

g0 = b1 ⊕ b0

Example:

Binary Code: b3  b2  b1  b0= 1 1 1 0 Gray Code: g3g2  g1 g0

g3 = b3 = 1 *[](https://testbook.com/blog/wp-content/uploads/2015/11/Conversion-from-Binary-code-to-Gray-Code-Example.png)*

g2 = b3 ⊕ b2 = 1 ⊕ 1 = 0

g1 = b2 ⊕ b1 = 1 ⊕ 1 = 0

g0 = b1 ⊕ b0 =1 ⊕ 0 = 1

∴ Final Gray code: 1 0 0 1

## Conversion Table from Binary to Gray Code:

|  |  |  |
| --- | --- | --- |
| Decimal | Binary | Gray |
| 0 | 0000 | 0000 |
| 1 | 0001 | 0001 |
| 2 | 0010 | 0011 |
| 3 | 0011 | 0010 |
| 4 | 0100 | 0110 |
| 5 | 0101 | 0111 |
| 6 | 0110 | 0101 |
| 7 | 0111 | 0100 |
| 8 | 1000 | 1100 |
| 9 | 1001 | 1101 |
| 10 | 1010 | 1111 |
| 11 | 1011 | 1110 |
| 12 | 1100 | 1010 |
| 13 | 1101 | 1011 |
| 14 | 1110 | 1001 |
| 15 | 1111 | 1000 |

1. **What is an instruction cycle? Explain about each instruction cycle.**

An instruction cycle, also known as fetch-decode-execute cycle is the basic operational process of a computer. This process is repeated continuously by CPU from boot up to shut down of computer.

Following are the steps that occur during an instruction cycle:

a)Fetch the Instruction

The instruction is fetched from memory address that is stored in PC(Program Counter) and stored in the instruction register IR. At the end of the fetch operation, PC is incremented by 1 and it then points to the next instruction to be executed.

b)Decode the Instruction

The instruction in the IR is executed by the decoder.

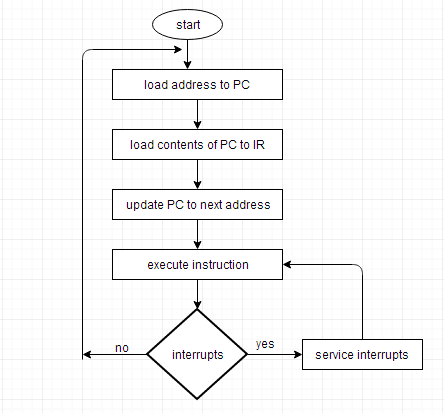
c)Read the Effective Address

If the instruction has an indirect address, the effective address is read from the memory. Otherwise operands are directly read in case of immediate operand instruction.

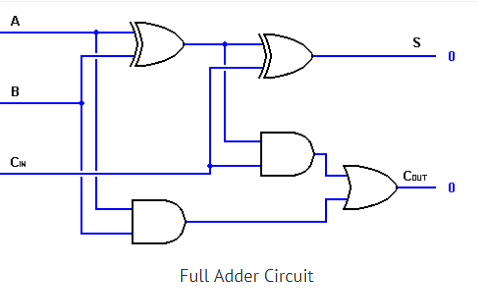
d)Execute the Instruction

The Control Unit passes the information in the form of control signals to thefunctional unit of CPU. The result generated is stored in main memory or sent to an output device.

The cycle is then repeated by fetching the next instruction. Thus in this way the instruction cycle is repeated continuously.



1. **Design a full adder with two half adders.**



 We can implement a full adder circuit with the help of two half adder circuits. The first half adder will be used to add A and B to produce a partial Sum. The second half adder logic can be used to add CIN to the Sum produced by the first half adder to get the final S output. If any of the half adder logic produces a carry, there will be an output carry. Thus, COUT will be an OR function of the half-adder Carry outputs.