Competitive Exams: Components of Computers

Input and Output

An input device is used to feed, or rear or input, programs and data into the computer. An output device is used by the computer to pass back, or write or print or output, the results of a computation to the user.

A computer will usually have several input and output (I/O) devices; these are collectively Icon as peripherals.

The Function Unit

In order to execute a program, the computer will need to do various operations on the data. These operations are carried out by the Function Unit.

A typical computer instruction might specify the addition of two numbers by the Function Unit. Inside the computer, each number is represented by a bit pattern, and the Function Unit is actually a device for processing these bit patterns to produce a third bit pattern which represents the sum of the two numbers.

Input and Output Instructions

In the I/O instructions the address field holds a binary number, called a device code, which specifies which of several possible I/O devices is to be used in the operation. Thus, the I/O instructions are rather similar to memory-reference instructions, except that they give the address of a device to be used in the operation, rather than the address of a Memory location.

If ID is the device code of a particular input device, the read, or input, instruction cause a character, or, more precisely, the code for the character, to be moved from the input device to the least-significant byte of the ACC. If OD is the device code of a particular output device, the write, or output, instruction causes the character code in the least-significant byte of the ACC to be moved to the output device, which might, for example, print the corresponding character, depending on the particular device used.

The skip instruction, SKP, is used to check whether an I/O device is busy doing an I/O operation; it is required for synchronization purpose, because of the great difference in speed between the CPU (which can execute about one million instructions per second) and peripherals (which can be tens of thousands of time slower) .

The first two instructions in the program

- LDA FIRST
- STA 256

initialise location 256 to contain the instruction LDA

The store instruction places the address I in location 256; however, because the operation code for LDA is 0000, and because the address has not more than twelve significant bits, so that its most-significant four bits are ′ 0, it is also a load instruction which references the first of the integers to be summed. This load instruction is set up so that the program can be re-used to sum integers in locations given by the address placed in location FIRST.

The next two instructions

- ADD N
- STA 256

store, in a temporary-storage location, the address of the last of the integers to be summed. This is the address in location FIRST + the value of n and is used when checking for the end of summing.

The next two instructions

- LDA 266
- STA SUM

initialize the value of the sum to 0.

Location 256 is shown to contain 0 to begin with, but this is overwritten by the STA instruction in location 251, and subsequently contains LDA instructions which reference the integers to be summed. Thus the first time the instruction in this location is executed it is

LDA I

and the three instructions

- LDA I ADD
- SUM STA SUM

add the first of the integers to the value of sum, which has previously been initialised to zero.

The three instructions

- LDA 256 ADD
- 267 STA 256

increment the address of the LDA instruction, jn location 256 so that it tc a location which contains the value of a variable, thus causing the variable's value to be executed as if it were an instruction. No such error can be made in Pascal, because the programmer is relieved of the low-level task of storage allocation. In Pascal, it is impossible for variable values or contents to be used as instructions. Also, constants which are explicitly written into a program, such as 1 in

x:= x + 1

cannot be asigned to, because they are treated differently to variables, and constant identifiers cannot be used in the same way as variable identifiers.