

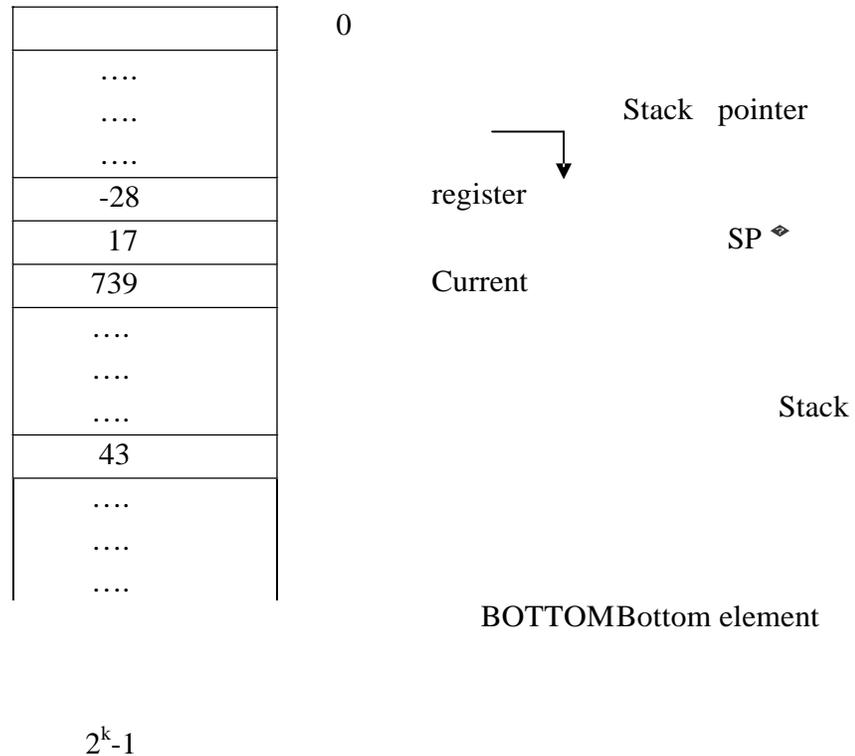
## STACKS AND QUEUES

A computer program often needs to perform a particular subtask using the familiar subroutine structure. In order to organize the control and information linkage between the main program and the subroutine, a data structure called a stack is used. This section will describe stacks, as well as a closely related data structure called a queue.

Data operated on by a program can be organized in a variety of ways. We have already encountered data structured as lists. Now, we consider an important data structure known as a stack. A stack is a list of data elements, usually words or bytes, with the accessing restriction that elements can be added or removed at one end of the list only. This end is called the top of the stack, and the other end is called the bottom. Another descriptive phrase, last-in-first-out (LIFO) stack, is also used to describe this type of storage mechanism; the last data item placed on the stack is the first one removed when retrieval begins. The terms push and pop are used to describe placing a new item on the stack and removing the top item from the stack, respectively.

Fig b shows a stack of word data items in the memory of a computer. It contains numerical values, with 43 at the bottom and -28 at the top. A processor register is used to keep track of the address of the element of the stack that is at the top at any given time. This register is called the stack pointer (SP). It could be one of the general-purpose registers or a register dedicated to this function.

Fig b A stack of words in the memory



Another useful data structure that is similar to the stack is called a queue. Data are stored in and retrieved from a queue on a first-in-first-out (FIFO) basis. Thus, if we assume that the queue grows in the direction of increasing addresses in the memory, which is a common practice, new data are added at the back (high-address end) and retrieved from the front (low-address end) of the queue.

There are two important differences between how a stack and a queue are implemented. One end of the stack is fixed (the bottom), while the other end rises and falls as data are pushed and popped. A single pointer is needed to point to the top of the stack at any given time. On the other hand, both ends of a queue move to higher addresses as data are added at the back and removed from the front. So two pointers are needed to keep track of the two ends of the queue.

Another difference between a stack and a queue is that, without further control, a queue would continuously move through the memory of a computer in the direction of higher addresses. One way to limit the queue to a fixed region in memory is to use a circular buffer. Let us assume that memory addresses from BEGINNING to END are assigned to the queue. The first entry in the queue is entered into location BEGINNING, and successive entries are appended to the queue by entering them at successively higher addresses. By the time the back of the queue reaches END, space will have been created at the beginning if some items have been removed from the queue. Hence, the back pointer is reset to the value BEGINNING and the process continues. As in the case of a stack, care must be taken to detect when the region assigned to the data structure is either completely full or completely empty.

Source : <http://elearningatria.files.wordpress.com/2013/10/cse-iv-computer-organization-10cs46-notes.pdf>