System call :-

The purpose of a system call is to request the operating system kernel to perform some activity.

→ System call provides an interface between a process & the OS.

→ System calls are generally available as assembly language instructions, but now-a-days, these are available as high level languages like C, PERL etc.

→ There are different types of systemcalls for performing different kinds of task.

~~Pro~~ System calls for Process control

(i) end, abort
(ii) load, execute
(iii) create process, terminate process
(iv) allocate & free memory for process

System calls for File manipulation

(i) create file, delete file
(ii) open, close
(iii) read, write
(iv) getfileattributes, set file attributes

System calls for Device Management

(i) request device, release device
(ii) read, write
(iii) get device attribute, set-deviceattribute
(iv) logically attach or detach device

System calls for information maintenance

(i) gettime or getdate, settime or setdate
(ii) get systemdata, set systemdata
(iii) get process, file or deviceattributes
(iv) set process, file or " "

System call for communication

(I) create, delete communication connection
(II) send msg, receive msg
(III) open connection, close connection, ~~accep~~ accept connection
(IV) read message, write message
(V) attach or detach remote devices

Suppose we want to ~~read~~ create a directory. The OS have a routine for ~~reading~~ creating a directory. To execute this we have to make a system call.

→ The steps of system call are:-

1. The system service is requested.
- If the process is running a user program in user mode & it needs a system service, then it has to execute a trap instruction to transfer control to the OS i.e switch from user mode to kernel mode.

2. The OS then finds out what the calling process wants by inspecting the parameters.

→ Three general methods are used to pass parameters to the OS

(i) Passing the parameters in registers

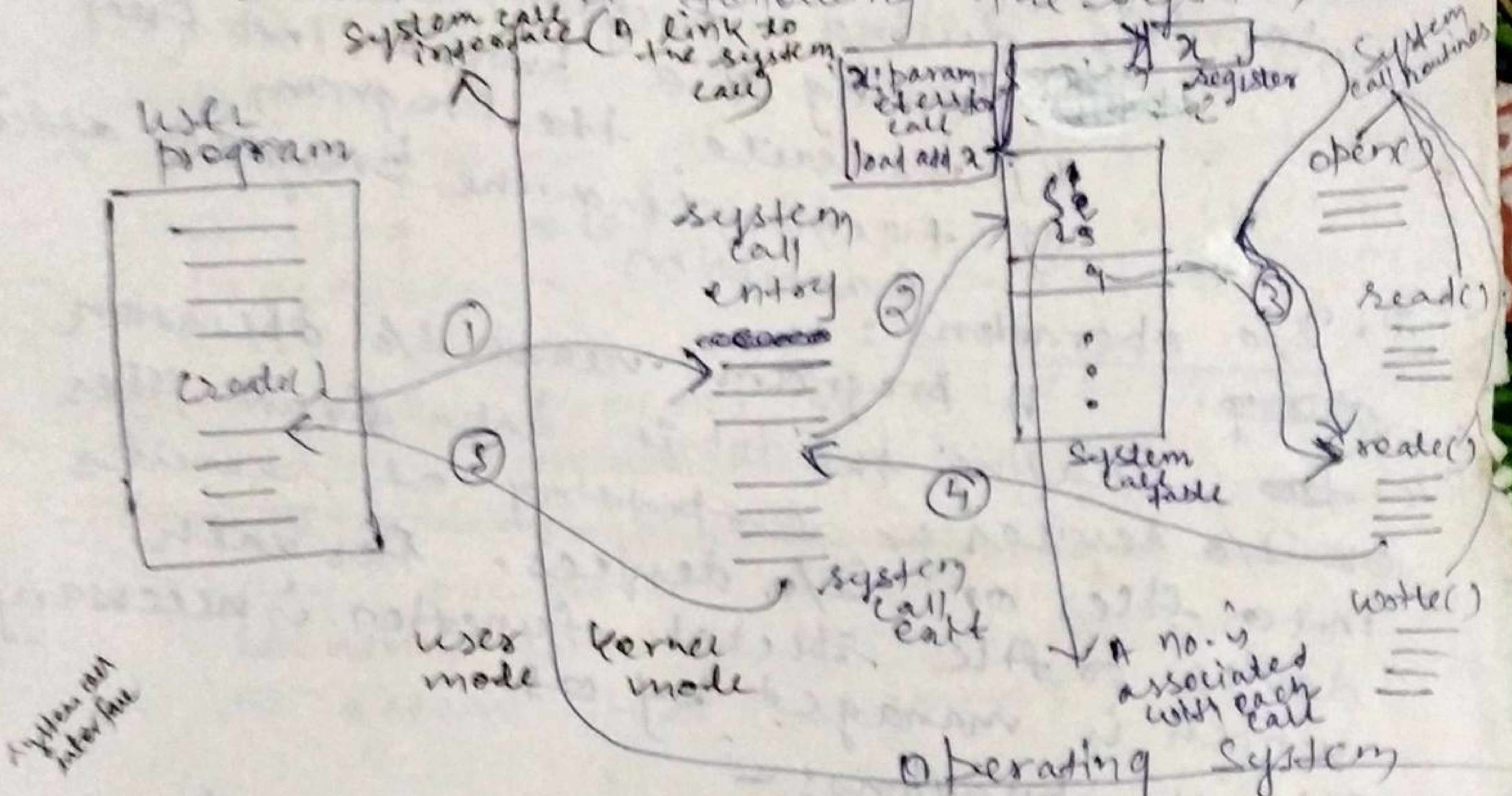
(ii) If there are more parameters than registers, then they are stored in a block or table in memory & the address of the block is passed as a parameter in a register.

(iii) Parameters can also be ~~placed or~~ pushed onto the stack or popped off from the stack.

3. Then the system call is carried out by branching to the service function

4. It returns from the service function

5. Finally the control is returned to the next statement following the system call.



1. system service is requested.
2. after switch mode it verify arguments & service
3. branch to the service function
4. return from service function
5. return from system call

<u>Note</u>

→ The OS is the most fundamental piece of s/w that runs in kernel mode (supervisor mode). In this mode it has complete access to all the h/w & can execute any instruction.

→ The rest of the s/w runs in user mode where only a subset of m/c instrⁿs are available. So they donot have complete access to all instrⁿs & comp. h/w.