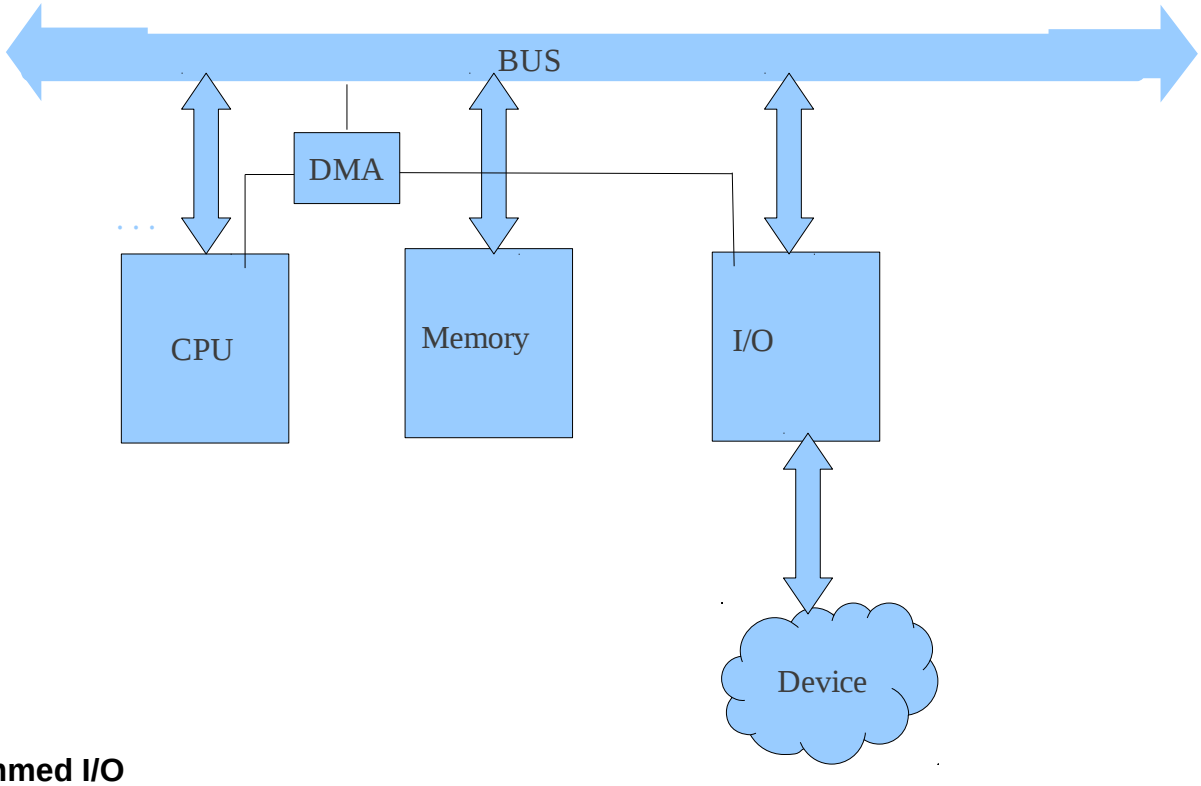
PROGRAMMED I/O

Ways to do INPUT/OUTPUT:

There are three fundamentally different ways to do I/O.

1. Programmed I/O
2. Interrupt-driven
3. Direct Memory access



Programmed I/O

The processor issues an I/O command, on behalf of a process, to an I/O module; that process then busy waits for the operation to be completed before proceeding.

When the processor is executing a program and encounters an instruction relating to input/output, it executes that instruction by issuing a command to the appropriate input/output module. With the programmed input/output, the input/output module will perform the required action and then set the appropriate bits in the input/output status register. The input/output module takes no further action to alert the processor. In particular it doesn't interrupt the processor. Thus, it is the responsibility of the

processor to check the status of the input/output module periodically, until it finds that the operation is complete.

It is simplest to illustrate programmed I/O by means of an example . Consider a process that wants to print the Eight character string ABCDEFGH.

1. It first assemble the string in a buffer in user space as shown in fig.
2. The user process then acquires the printer for writing by making system call to open it.

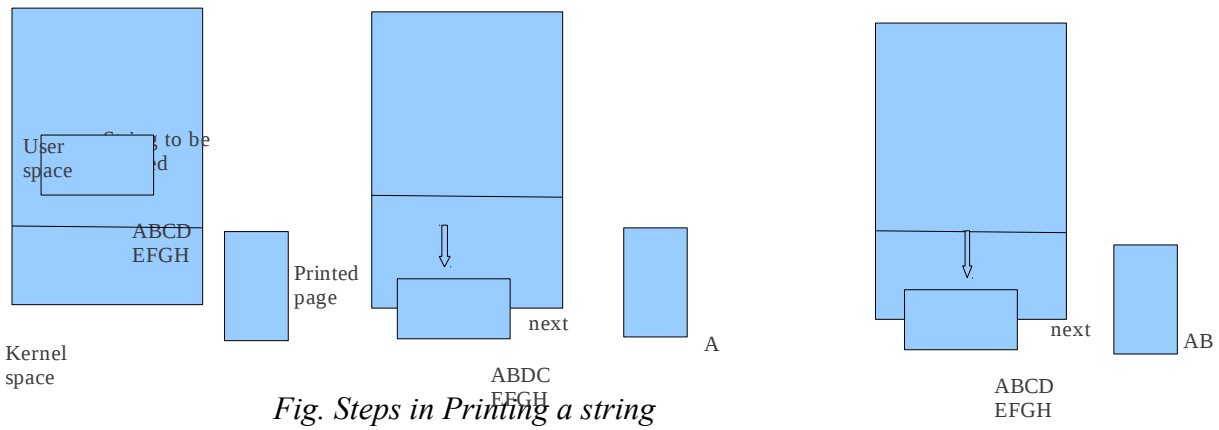


Fig. Steps in Printing a string

3. If printer is in use by other the call will fail and enter an error code or will block until printer is available, depending on OS and the parameters of the call.
4. Once it has printer the user process makes a system call to print it.
5. OS then usually copies the buffer with the string to an array, say P in the kernel space where it is more easily accessed since the kernel may have to change the memory map to get to user space.
6. As the printer is available the OS copies the first character to the printer data register, in this example using memory mapped I/O. This action activates the printer. The character may not appear yet because some printers buffer a line or a page before printing.
7. As soon as it has copied the first character to the printer the OS checks to see if the printer is ready to accept another one.
8. Generally printer has a second register which gives its status

The action followed by the OS are summarized in fig below. First data are copied to the kernel, then the OS enters a tight loop outputting the characters one at a time. The essentials aspects of programmed I/O is after outputting a character, the CPU continuously polls the device to see if it is ready to accept one. This behavior is often called polling or Busy waiting.

```
copy_from_user(buffer,p,count); /*P is the kernel buffer*/
for(i=0;i<count;i++) { /* loop on every characters*/
while(*printer_status_reg!=READY); /*loop until ready*/
printer_data_register=P[i]; /*output one character */
}
return_to_user();
```

Programmed I/O is simple but has disadvantages of tying up the CPU full time until all the I/O is done. In an embedded system where the CPU has nothing else to do, busy waiting is reasonable. However inmor

complex system where the cpu has to do other things, busy waiting is inefficient. A better I/O method is neede