► ***Question 1.*** If we changed the keyboard interface so that the R bit occupies position 5 in the control/status register, which instruction would you change and how?

► **Question 2.** Modify *wait.asm* by adding an instruction that reads the data register and leaves the read value in *$t2*, as shown in Figure 6. Note that the address of this register is expressed in Figure 4 as "DB+4".

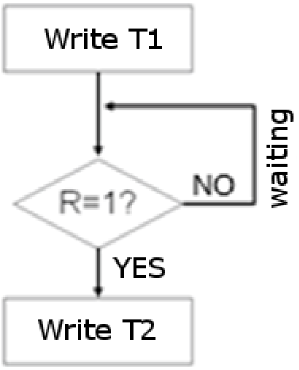


Figure 1. Diagram of modified *wait.asm*

► Try to run the modified program, several times without closing the simulator. You will notice that the program always waits for a key to be typed because the ready bit R is 0 at the end of the execution

► Write a program *ascii.asm* that repeatedly reads the keys that are typed on the keyboard and then writes their corresponding ASCII codes on the console. The program will finish when a chosen key is typed (return, point, etc.) The processing will be:

1. Read the data register from the keyboard interface.
2. Write on the console the value read by using the `print_int` system call.

The program pseudocode is:

```
Repeat
    wait until keyboard read (bit R = 1)
    Processing:
    read the keyboard data register
    write on the console the value read
until read_caracter == chosen_character
```

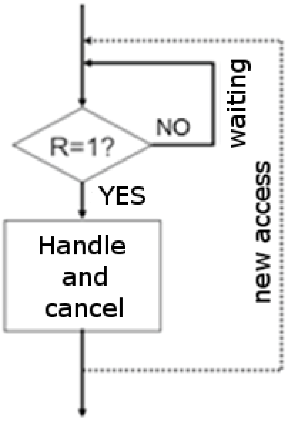
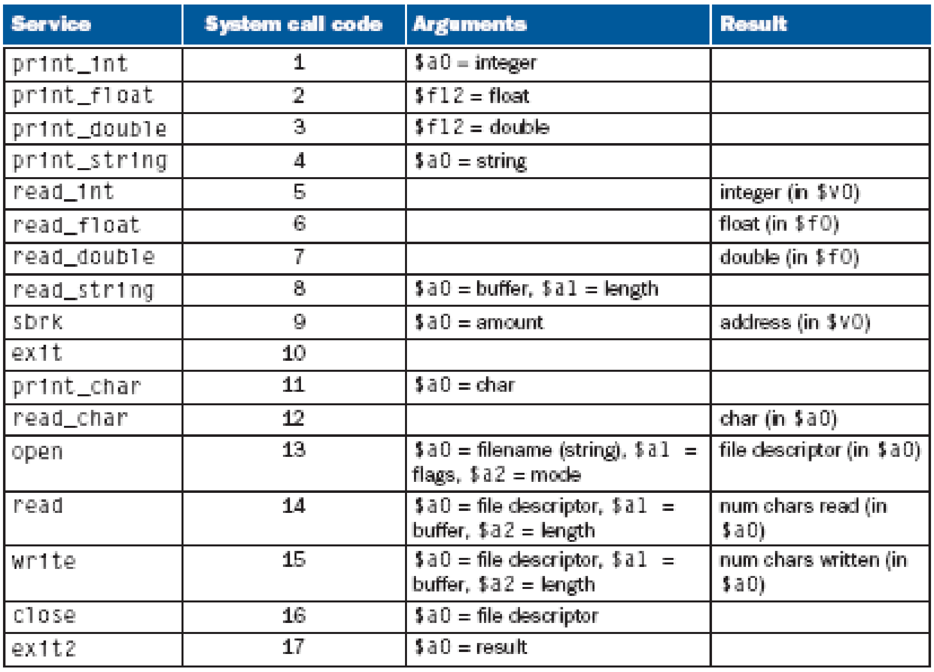


Figure 2. Common diagram for peripheral management by polling

► ***Question 3.*** Copy here the lines of code in charge of synchronization and reading the data register.

► ***Question 4.*** Write here the code for getchar and putchar

► ***Question 5.*** Can you explain why the program has stopped at that point?



Service	System call code	Arguments	Result
print_int	1	$a0 = integer	
print_float	2	$f12 = float	
print_double	3	$f12 = double	
print_string	4	$a0 = string	
read_int	5		integer (in $v0)
read_float	6		float (in $f0)
read_double	7		double (in $f0)
read_string	8	$a0 = buffer, $a1 = length	
sbrk	9	$a0 = amount	address (in $v0)
exit	10		
print_char	11	$a0 = char	
read_char	12		char (in $a0)
open	13	$a0 = filename (string), $a1 = flags, $a2 = mode	file descriptor (in $a0)
read	14	$a0 = file descriptor, $a1 = buffer, $a2 = length	num chars read (in $a0)
write	15	$a0 = file descriptor, $a1 = buffer, $a2 = length	num chars written (in $a0)
close	16	$a0 = file descriptor	
exit2	17	$a0 = result	