

Imperial College London  
Department of Computing

**Computer Systems (113) / Architecture (110)**  
Exercises – *Input/Output*

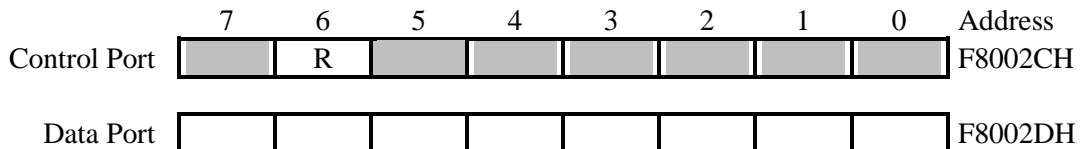
- 1) Why is a control port needed?
- 2) What might happen if the following instruction sequence is executed? Assume that copying a byte to the PrinterDataPort causes the byte to be printed on the printer.

```

mov byte [PrinterDataPort], 'A'
mov byte [PrinterDataPort], 'B'
mov byte [PrinterDataPort], 'C'
mov byte [PrinterDataPort], 'D'

```

- 3) Given a device with an 8-bit dataport that can transmit data at a maximum rate of 2000 bytes per second, and an interrupt processing time of 10 **microseconds**, calculate the maximum time that the CPU might spend servicing the device in each **second**.
- 4) If the time to process a DMA interrupt is also 10 **microseconds** and given a DMA controller that (i) can transfer blocks upto 1000 bytes at time (ii) has a 16-bit dataport for memory transfers and (iii) controls a device that can transmit data at a maximum rate of 2000 bytes per second, calculate:
  - a) how much time the CPU might devote to servicing the device in each **second**.
  - b) how much time the DMA controller might spend on memory accesses if a 16-bit memory access takes 100 **nanoseconds**.
- 5) Using memory-mapped programmed I/O, write an Intel IA-32 function called *readch* to return the next character read from the keyboard (return in register AL). Your function should loop (poll) waiting for the R bit to be set and then return the character in the dataport.



R=0	Keyboard is idle and waiting for a character to be typed.
R=1	Character has been read and is available in the Data Port. <b>Assume that on reading the Data Port, bit R is set to 0 and the device awaits another char.</b>

- 6) Write an interrupt handler called *kbhandler*, that will place the next ASCII character read from the keyboard into a buffer. Use the I/O port addresses in question 5. You can use the following routine to buffer characters:

```

void putchar (char ch) /* Adds character ch to buffer */

```

- 7) Write an input character function called *nextch*, which returns the next ASCII character read from the buffer, if no characters are available the function should loop until a character is available. You can use the following routine:

```

int getchar ( ) /* Returns next character in buffer or -1 if buffer is empty */

```

You should assume that *getchar* will disable interrupts if modifying data structures shared with the device-driver's interrupt handler. Assume an **int** is 32-bits.

Imperial College London  
Department of Computing

**Computer Systems (113) / Architecture (110)**  
Solutions – *Input/Output*

- 1) Without a control port the CPU cannot synchronise correctly with the I/O controller, e.g. it cannot know when data is ready for input or when data has been output. The control port is also used to issue device commands and indicate I/O errors.
- 2) There are a several possibilities. A “nice” I/O controller would indicate an overrun error (i.e. data is being copied into the dataport too quickly) in one of the status bits of the control port. A more “primitive” I/O controller might output the character A or perhaps the character D or perhaps even some other character. Remember that the printer outputs characters relatively slowly compared with how fast the CPU executes consecutive instructions.
- 3) CPU Time Spent = Time for 2000 interrupts =  $2000 \times 10 \times 10^{-6} = 20,000 \times 10^{-6} = 20 \times 10^{-3} = 20$  milliseconds
- 4) (a) CPU Time Spent = Time for 2 interrupts =  $2 \times 10 \times 10^{-6} = 20$  microseconds.  
(b) DMA Time Spent = Time for 1000 memory accesses =  $1000 \times 10^{-7} = 100$  microseconds.
- 5)

```
controlport equ    F8002CH
dataport  equ     F8002DH

readch:   test    byte[controlport], 40H    ; Test if R bit is 1
          jz     readch                    ; If R=0 then repeat
          mov    al, [dataport]           ; Copy ASCII char to output register
          ret
```
- 6)

```
kbhandler: sti                                ; Re-enable interrupts
            push  ax                          ; Save register ax
            mov  al, [dataport]              ; Read ASCII char
            push  ax                          ; Parameter in AL part
            call putchar
            add  esp, 2
            pop  ax                          ; Restore register ax
            iret
```
- 7)

```
nextch:   call  getchar
            cmp  eax, 0
            jl   nextch                      ; Repeat if negative.
            ret
```