CPS310 - COMPUTER ORGANIZATION II

LAB 3

ARC MEMORY MAPPED I/O

Submission instruction:

Labs will be done individually. Please complete and submit this lab on D2L by the submission deadline according to the details provided by your TA. Each student should submit a pdf file that includes all their written work and screenshots of the results of the simulations. Please note that Lab 3 will be graded based on attendance, and correct completion of the following two programs in addition to answer to TA's questions.

Doing I/O using ARC Memory Mapped I/O

ARC does not have any explicit I/O instructions. I/O is then done by reading from and writing to pre-defined memory locations in the memory which are assigned to I/O devices. This is known as memory mapped I/O. Each I/O device has a data port. Similarly, each device has a status port that is used to test the readiness status of the I/O device.

For the purpose of this lab, we are interested in using the ARC simulator to use:

- The console output port to display characters
- The keyboard input port to read characters

1) Please provide comments for all instructions.
2) What is the memory mapped address for the console output and keyboard input?
3) Please identify the specific line of code that prints to console and reads the keystroke.

PART A - Output: Printing character to the display

The memory addresses associated with the console output I/O device are:

- 0xffff0000 is the console (output) data port
- 0xffff0004 is the console (output) status port where bit 7 is the ready flag (0: not ready, 1: ready)

Printing a character is achieved by:

- Checking the ready flag to see if device is ready for printing, i.e., check if the bit 7 of the output status port is set to 1
- Storing the character to the output data port

```
! Prints "Hello, world!\n" in the message area.
                .begin
```

```
BASE        .equ    0x3fffc0        !Starting point of the memory mapped region
COUT        .equ    0x0             !0xffff0000 Console Data Port
COSTAT      .equ    0x4             !0xffff0004 Console Status Port


            .org    2048
            add     %r0, %r0, %r2
            add     %r0,  %r0,  %r4
            sethi  BASE, %r4
Loop:       ld      [%r2 + String], %r3 !Load next char into r3
            addcc %r3,%r0,%r3
            be     End                  ! stop if null
Wait:       ld      [%r4+COSTAT], %r1
            andcc %r1, 0x80, %r1
            be     Wait
            st      %r3, [%r4+COUT]   !Print to console
            add     %r2, 4, %r2       !increment String offset (r2)
            ba      Loop
End:        halt                !A non-standard instruction to stop the simulator

            .org    3000
! The "Hellow, world!" string
String:     0x48, 0x65, 0x6c, 0x6c, 0x6f
            0x2c, 0x20, 0x77, 0x6f, 0x72
            0x6c, 0x64, 0x21, 0x0a, 0
            .end
```

PART B - INPUT: Accepting character from the keyboard

The memory addresses associated with the keyboard input I/O device are:

- 0xffff0008 is keyboard (input) data port
- 0xffff000C is the keyboard (input) status port where bit 7 is the ready flag (0: not ready, 1: ready)

Reading a character is achieved by:

- checking the ready flag to see if the keyboard is ready, i.e., check if the bit 7 is set to 1
- loading the character from the keyboard data port

```
! Read a character from keyboard
            .begin
BASE        .equ    0x3fffc0        !Starting point of the memory mapped region
COUT        .equ    0x0             !0xffff0000 Console Data Port
COSTAT      .equ    0x4             !0xffff0004 Console Status Port.
CIN         .equ    0x8             !0xffff0008 Keyboard Data Port
CICTL       .equ    0xc             !0xffff000c Keyboard Control Port
```

```
                .org    2048
                add     %r0, %r0, %r4       !Clear r4
                sethi   BASE, %r4

InWait:         halt
                ld      [%r4 + CICTL], %r1
                andcc   %r1,  0x80,  %r1
                be      InWait

                ld      [%r4 + CIN], %r3
                subcc   %r3, 27, %r5        ! 27 is Escape
                be      End                 ! stop if it is.

Wait:           ld      [%r4 + COSTAT], %r1
                andcc %r1, 0x80, %r1
                be      Wait
                st      %r3, [%r4 + COUT]
                ba      InWait

End:            halt
                .end
```