Computer Organisation 300096

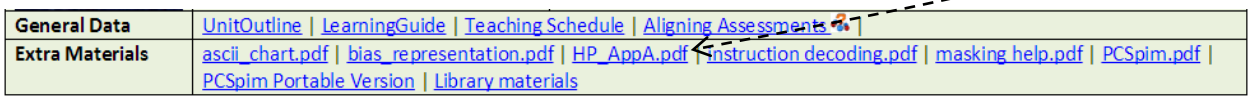
Lab Sheet 9 (starts session week 11, due in week 12)

Student Name and Number	
Date, Grade and Tutor signature, **max mark 5**	

Keep this cover sheet marked and signed by the tutor.

1. Preparation [Total max. mark: 0.5]

The main goal of today's lab is to understand the memory mapped I/O, and the assembly language techniques for manipulating strings of characters. Study section A.8 from the **HP_AppA.pdf** (on the website), the lecture notes, and the appropriate sections from the textbook.



Self study task [0.5 marks]: Describe below in your own words how memory mapped I/O is implemented in PCSpim. Illustrate with hand drawing where appropriate.

2. Workshop Task I [Total max. mark: 0.5]

Please answer the lab Questions listed below in writing - print or neatly write your answers.

Get the assembly language files *repeat.s* and *stringlength.s* provided for this lab. Invoke the PCSpim simulator. Make sure that it has memory-mapped I/O **enabled** (see "Settings").

1. Open and run *repeat.s*, analyse what it does. It helps to set breakpoints in appropriate places and observe the changes in receiver and transmitter registers (**note:** this program will not single step).

   Questions (max. mark 0.3): draw a flow chart diagram to how the program implements reading character from the keyboard and writing to the console. To reinforce your understanding, insert breakpoints at **`0x0040004c`** and **`0x00400068`** (if you have a different memory layout, refer to the memory image below to set breakpoints in appropriate places), run the program, then **record** status/changes observed in **relevant registers** (for handing memory-mapped I/O) at both breakpoints (you need to type in characters to let go; if QtSpim is not working, try PCSpim).

```
[0x00400048]    0x8d100004  lw $16, 4($8)                    ; 30: lw   $s0, 4($t0)
[0x0040004c]    0x3c011001  lui $1, 4097                     ; 32: lbu  $s1, terminator
[0x00400050]    0x90310020  lbu $17, 32($1)
[0x00400054]    0x12110006  beq $16, $17, 24 [exit-0x00400054]; 33: beq  $s0, $s1, exit
[0x00400058]    0x8d090008  lw $9, 8($8)                     ; 37: lw  $t1, 8($t0)
[0x0040005c]    0x31290001  andi $9, $9, 1                   ; 38: andi $t1, $t1,0x0001
[0x00400060]    0x1120fffe  beq $9, $0, -8 [writeloop-0x00400060]; 39: beq  $t1, $zero,wr
[0x00400064]    0xad10000c  sw $16, 12($8)                   ; 40: sw   $s0, 12($t0)
[0x00400068]    0x0810000f  j 0x0040003c [readloop]          ; 41: j  readloop
[0x0040006c]    0x34020004  ori $2, $0, 4                    ; 50: li  $v0, 4
```

2. Open and run the program *stringlength.s* Experiment with it to understand how it works, specifically how the procedures are called from the main program.

   Question (max. mark 0.2): describe in writing (or drawing) how *stringlength.s* interfaces with its internal procedure *strlen*, and how *strlen* calculates the length of a string.

3. Workshop Task II [Total max. mark: 1.5]

Write a program which reads in characters from the keyboard using memory mapped I/O, and stores them in a buffer in memory (array of spaces; define any size of the buffer you like, say 6). When the buffer is full, the program displays appropriate message, prints all entered characters in the buffer using memory mapped I/O, and terminates. You may use the program *repeat.s* as a starting point, rename it, modify it, add your own code, etc.

Note: Characters should be read from the keyboard, and displayed at the console window using memory mapped I/O. **Do not use syscalls for this task**, using syscalls would make this task trivial and would receive NO marks for this question. **You are allowed to use syscalls only** to display the counts of characters and all text messages.

4. Workshop Task III [Total max. mark: 2.5]

Expand the program which you wrote above to perform the following additional tasks:

1. When the buffer is full, it prints the accumulated string in the buffer on to the console. Don't terminate the program, but reuse the buffer for keyboard input (start filling in the buffer from the beginning);
2. The program maintains count of the total characters entered;
3. When the terminating character is entered (define any character you like, for instance, 'Enter' or capital letter 'A'), the program displays the leftover characters accumulated in the buffer (to the current position), reports counted total number of characters entered, and terminates.

   Note 1: You are allowed to use syscalls only for displaying the counts of characters. Characters should be read from the keyboard, and displayed on the console using **memory mapped I/O**.

   Note 2: Task II and Task III should be demonstrated separately; Task III doesn't automatically cover Task II.

Be prepared to explain the code. For the marking purpose.

Assessment notice

When you ready, present to the tutor a printed copy of your program source code, with your name and student number included in the comments (#…), and typed or neatly written answers to questions, if there are any listed in the lab sheet. Your tutor may decide to keep the source code printout, but you should keep marked and signed cover sheet.

Warning: Any source code duplicated amongst students will result in a zero mark, and possible further action according to the WSU policy on plagiarism.