



# McGill

December 2009  
Sample Final Examination

Introduction to Computer Systems

COMP-273

December 15, 2009 at 9:00 – 12:00

Examiner: Joseph Vybihal

Assoc Examiner: Michael Langer

|               |  |            |  |  |  |  |  |  |  |  |  |  |  |
|---------------|--|------------|--|--|--|--|--|--|--|--|--|--|--|
| Student Name: |  | McGill ID: |  |  |  |  |  |  |  |  |  |  |  |
|---------------|--|------------|--|--|--|--|--|--|--|--|--|--|--|

## INSTRUCTIONS:

- This is a **CLOSED BOOK** examination.
- You are permitted
- **STANDARD CALCULATORS**
- This examination is **PRINTED ON BOTH SIDES** of the paper.
- This examination paper **MUST BE RETURNED**
- You are permitted to write your answers in either **English or French**
- Write your answers in the **exam booklet** provided
- Attempt all questions, **part marks** will be assigned, show your work.

## Grading

| Section                                 | Grade | Your Mark                            |
|---|-------|--------------------------------------|
| Question 1: Definitions                 | 25    | <input type="text"/>                 |
| Question 2: MIPS Program                | 25    | <input type="text"/>                 |
| Question 3: MIPS Peripheral Programming | 25    | <input type="text"/>                 |
| Question 4: Calculations                | 25    | <input type="text"/>                 |
| <b>Total</b>                            | 100 % | <input type="text"/> Bonus: ____ / 2 |

**Question 1: Definitions (25 points)****(A) Two-Pass Assemblers**

Answer the following questions: What happens during pass one? What happens during pass two? What is the symbol table and what is its structure? What's the address of the 5<sup>th</sup> instruction in an assembler program? What does BEQ \$T1, \$T2, END assemble into, assuming END is 3 instructions below? What happens to the symbol table at the end of the assembly process?

**(B) Stack, Heap, Memory organization**

Diagrammatically contrast the standard memory organization of RAM with the MIPS VM memory organization of RAM. Who controls the stack and heap and show MIPS examples of how you can use it? Using the standard model describe and draw how the peripherals are connected to RAM. What hardware exists to help the programmer communicate with the peripherals?

**(C) Virtual Memory**

Contrast a Page with a Frame. What is a TLB and how does it work? What is a page table and how does it work? Explain how the VM address is converted into a Real address. Identify some of the registers and hardware structures that exist in a CPU to support VM. What is the swap file? What happens when you run a ctim and how does it select one?

<https://eduassistpro.github.io/>

How it will be graded:

8 points – (A), 1 point for each question, except for BEQ

8 points – (B), 2 points for each question

8 points – (C), 1 point for each question

1 point – for answering all the questions

**Question 2: MIPS Program (25 points)**

Using any MIPS techniques you like, create a complete MIPS program that asks the user for 10 numbers and then performs a Bubble Sort (or a Selection Sort) on those numbers. The program should then display the sorted numbers to the screen. The swap operation must be its own function.

How it will be graded:

5 points – swap

5 points – output sorted numbers

5 points – input 10 numbers

10 points – Bubble Sort

### Question 3: MIPS Peripheral Programming (25 points)

Operating System designers often have to write hardware drivers in assembler. For this problem, I am asking you to write portions of a printer's software driver. A driver is a set of functions that the OS or a user's program can call to interface with a device. It is just a regular assembler function. What is special about it is that it is specifically designed to manage and interface a particular device. It is critical that a driver function protect the registers once it is invoked.

Generally speaking, printers are very slow devices compared to CPUs. In class we compared hard disk access with memory access and saw that it took 20 million ns for data to move to and from a hard disk. To compare this to a printer, hard disk access is lightning fast. This complicates printing a little.

This question asks you to write three device functions in MIPS:

- `int Aquire_Device ()`
- `int Send_Data (int id, char c)`
- `void Release_Device (int id)`

We will imagine that this is a multiprocessing environment and any process may ask for the device at any time. Now, we know that only one device can use the printer at any time. `Aquire_Device` is the function that determines if the device is available for use by the requesting process. If a process does not have control over the device, it must call `Send_Data` function.

The OS must manage the device interface with the device directly. Instead, this manages the state of the device using a word of memory. This ID is used to store an ID number. The label for this memory is incremented each time a process calls `Aquire_Device`. This ID is returned by the `Aquire_Device` function. The process uses the returned number as a ticket when it is its turn to use the printer.

A process can call `Send_Data` at any time but the function will only send the character to the printer if the process is permitted to use the device. This function knows who is permitted to use the device because it has a word in memory called `ActiveID`. This memory location stores the ticket ID number of the process who is permitted to use the device. The process who calls this function and does not match this ID is kicked out with a return code of -1. If the processes ID number matches `ActiveID` then the data is sent to the printer.

Once a process is finished printing it calls the `Release_Device` function. This function verifies if the id number matches `ActiveID`. If it does match it then increments `ActiveID` to permit the next process with the next ticket number in. This function then returns. If the id did not match `ActiveID` then the function just terminates as well.

To send data to a printer occurs in this manner:

The printer for this questions has two registers: a combined status/command register and a data register. The Statis/Command register is at address 1000 Hex while the data register is at address 1004 Hex. The data register is very simple. Any data placed in their will be sent to the printer for display. You

can put in there a Unicode character or an integer number. If it is a Unicode character then it will use only the first 16 bits, 0 to 15. If it is a number then you can use any of the 32 bits.

The status register is a 16-bit register. It contains information on how busy the printer is, what type of data is in the data register, if there were any errors, and the command bits. The register is formatted as follows:

- Bit 0 = True if busy (I.E. currently printing something)
- Bit 1 = Data type (0 = character, 1 = integer)
- Bit 2 = Existence of data (1 = data in data register, 0 = no data or not valid data)
- Bit 3-5 = Printer error code (0-no error, 1-paper, 2-ink, 3-data, 4-time out, space for future errors)
- Bit 6 = Print (0 = don't print, 1 = print contents of data register)
- Bit 7 = Form feed (1 = perform form feed)
- Bit 8 = Reset (1 = reset/reboot printer)
- Bit 8-F = Not used

To send data, you must first wait for the availability of the printer. Then you must put information into the data register. Then you must tell the printer to print. The program should wait for the printer to finish and return its error code. Then the function returns the error code.

When the printer finishes printing it does the following: sets busy to zero, sets existence of data to zero only if there was an error, set the error code, and clears all the command bits (bits 6, 7 and 8).

#### Question 4: Calculati

All the following questions assume a 1GHz MIPS CPU. Each instruction takes 4 cycles to run, calculate the following:

1. Given the data and the code you wrote for Question 4, calculate the overhead in cycles your polling and interrupt algorithms would have, individually. (The actual answer is not important here but how you calculated it)
2. Then compute the impact on the CPU by percentage of load.
  - Assume you want to poll 100 times per second
  - Assume you want the interrupt to transfer 100 numbers to the array
    - Suggest an optimization to your original interrupt code (show code)
    - How would the interrupt calculation change if we had DMA (show calculations)? What about the algorithm (show code)?

Note: You do not need to rewrite ALL the code. Just enough to help us understand.

#### How it will be graded:

- |                           |   |
|---------------------------|---|
| 5 points – Question 1     | 5 points – Polling load                     |
| 5 points – Interrupt load | 5 points – Suggested interrupt optimization |
| 5 points – DMA question   |   |

### Bonus Question

First bonus question: Who is Tom West?

Second bonus question: What's a Microkid?

**Assignment Project Exam Help**

**<https://eduassistpro.github.io/>**

**Add WeChat edu\_assist\_pro**

**Assignment Project Exam Help**

**<https://eduassistpro.github.io/>**

**Add WeChat edu\_assist\_pro**

**Assignment Project Exam Help**

**<https://eduassistpro.github.io/>**

**Add WeChat edu\_assist\_pro**

**Assignment Project Exam Help**

**<https://eduassistpro.github.io/>**

**Add WeChat edu\_assist\_pro**