Assignment 2

CIS*2030: Structure and Application of Microcomputers

Assignment 2 is due **11:59pm, September 25, 2021**. The assignment is to be completed individually. If you have any questions, please see the teaching assistant or instructor during office hours.

Instructions:

(1) When answering questions, simply writing down a solution to a particular problem is not enough to obtain full marks. You must show how you arrived at the solution by showing the step-by step procedure that you performed. This procedure can be hand-written or typed – the choice is yours. However, solutions must be clear. What cannot be read cannot be marked.
(2) Make sure to neatly print/type your full name, student ID, and the name of your teaching assistant at the top of the first page, number the remaining pages consecutively at the bottom.
(3) Review your answers 1 or 2 days after completing the assignment. Taking a few minutes to do this can dramatically improve your understanding of key concepts and increase your retention. Also, it makes it much easier for you to catch any errors.

Submission:

(1) Once you are ready to submit your assignment for grading print or scan your individual assignment pages and convert them into a *single* PDF submission. (Do not upload individual pages or a zip file.) If you are working with paper and do not have a scanner, use a cellphone to take pictures of each page, then use a program like Adobe Acrobat Scan to create a single PDF. (Remember to use appropriate lighting, as dim or blurry pictures that cannot be read cannot be marked.) Upload your single PDF submission to the Dropbox labeled **Assignment 2** available in the **week 2** folder on CourseLink.
(2) After uploading your assignment to CourseLink, immediately download the assignment and verify that you have uploaded the correct document.

Best of success! 😊

Questions

This assignment covers general concepts related to **computer architecture** and specific details related to the **Motorola 68000 ISA**, as discussed in *week 2*.

1. A certain number of *data registers* are defined as part of the Motorola 68000's ISA. How many *bytes* of storage are available within the processor using all of these registers? How many *words* of storage are available? [1/2 point each]

2. The 68000's ISA defines the extensions .B, .W, and .L. What are these used for both with respect to registers and memory? [1 point]

3. Classify each of the decimal values -128 and 347689 as byte, word, or longword. Note: A value may belong to more than one classification. [1 point each]

4. What does the $ signify in the number $123? Is the number 123 the same as the number $123? Explain. [1 point]

5. Explain the difference between the following two 68000 instructions: MOVE.B 123,D4 and MOVE.B #123,D4. [1 point]

6. What, if anything, is wrong with the instructions MOVE.W #-32769,D1 and SUB D1,#$15? [1 point each]

7. A hypothetical computer employs a word size of 22 bits. How many unique memory addresses can the processor generate? If each cell in the memory has a size of 16 bits, how many bytes of memory can this processor address? [2 points]

8. Consider the following structure declaration:

```
struct S {
    int a;
   char b;
    int c;
};
```

   Assume that the previous structure is compiled on a hypothetical computer where integers are 4 bytes and characters are single bytes. Also, assume that the computer's ISA requires an **`int`** to be aligned (i.e., start) on an even address boundary. How many bytes would the compiler have to allocate to ensure that the structure satisfied the memory alignment requirement? Note: Help explain your answer by drawing the contents of memory and the offset of each object defined as part of the structure, as illustrated in lecture. [1 point]

9. This question follows on from the previous question. This time assume that the hypothetical computer's ISA requires an **`int`** to be aligned (i.e., start) at an address that is a multiple of 4 – something that is very common today. What size gap must the compiler insert in memory between **`b`** and **`c`** to ensure proper alignment? Again, help explain your answer by drawing the contents of memory and the offset of each object defined as part of the structure. [1 point]

10. What happens if the Motorola 68000 attempts to perform a word access or long word access to an odd memory address? Explain. [1 point]

11. A particular ISA employs Big Endian byte ordering. If the 32-bit value $12345678 is loaded into memory at location $1010, what is the memory address of the byte $34? What address would the byte $34 have if the ISA employed Little Endian byte ordering? [2 points]

12. Consider the C code below:

```
#include <stdio.h>

typedef unsigned char * pbyte;

void display_bytes(pbyte beginning, int numbytes) {

   for(int i=0; i<numbytes; i++)
        printf(" %.2x", beginning[i]);
   printf("\n");
}

void main(void){

   int hex_value = 0x12345678;
   pbyte ptr = (pbyte) &hex_value;

   display_bytes(ptr,4);
}
```

Once you understand what the code does, compile and run the code on the SoCS' servers available at *linux.socs.uoguelph.ca* (or *nomachine.socs.uoguelph.ca*). Then, examine the output produced by the program and use it to explain if the CPU that the code ran on employs Big Endian or Small Endian byte ordering. [1 point]

13. Consider the following sequential 68000 program segment:

    ```
    MOVE.B #$46,D0
    SUB.B  #$51,D0
    ```

    Give the values of the C, Z, N, and V flags in the Condition-Code Register (CCR) after the execution of the program segment. Remember to show your work, including how the processor performs the subtraction operation in binary. What do the values of the flags tell you about the result of the subtraction operation? [3 points]

14. The hexadecimal value $3803 is the machine-language code for a particular MOVE instruction. Using the datasheet for the MOVE instruction on pages 316 and 317 of your textbook, translate the instruction into its assembly-language form. [2 points]

15. The hexadecimal value $103C 0009 is the machine-language code for, yet another, MOVE instruction. Translate the instruction into its assembly-language form. [2 points]

Remember to show all of your work when answering the previous questions. This will ensure that you are eligible for partial marks.