Assignment 3

CIS*2030: Structure and Application of Microcomputers

Due: **11:59pm, November 1, 2024**

Neatly print your name, student identification number, and lab information below:

Name:

ID: **Section:**

Instructions:

(1) You must work alone and not discuss solutions with others (artificial or human).
(2) 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.
(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 3** available 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

1. Download the sample program called **factorial.X68** from the course website. The program computes *n!* recursively, where *n* is assumed to be a non-negative integer. The parameter *n* is passed to the subroutine on the stack, while the result (a 16-bit integer) is returned in data register D1. The subroutine uses one working register, D0, which is saved and restored upon entry and exit, respectively.

   Before proceeding, first load the program into the Easy68K environment and then run it. When prompted to enter the value of *n*, enter the value 5. This will cause the program to compute factorial 5. Make sure that the program correctly outputs the value 120.

   Reload the program. Set a breakpoint at on line 24, where the instruction is labeled `callFactorial`. Run the program. When prompted to input the value of *n*, enter the value 4. When the simulator stops at the breakpoint, fill in the value of the stack pointer in the figure on the following page.

   Continue to trace through the program one instruction at a time until the program terminates. Every time that an instruction affects the contents of the stack, the stack pointer, or the frame pointer report the changes by filling in the blanks in the memory map on the following page.

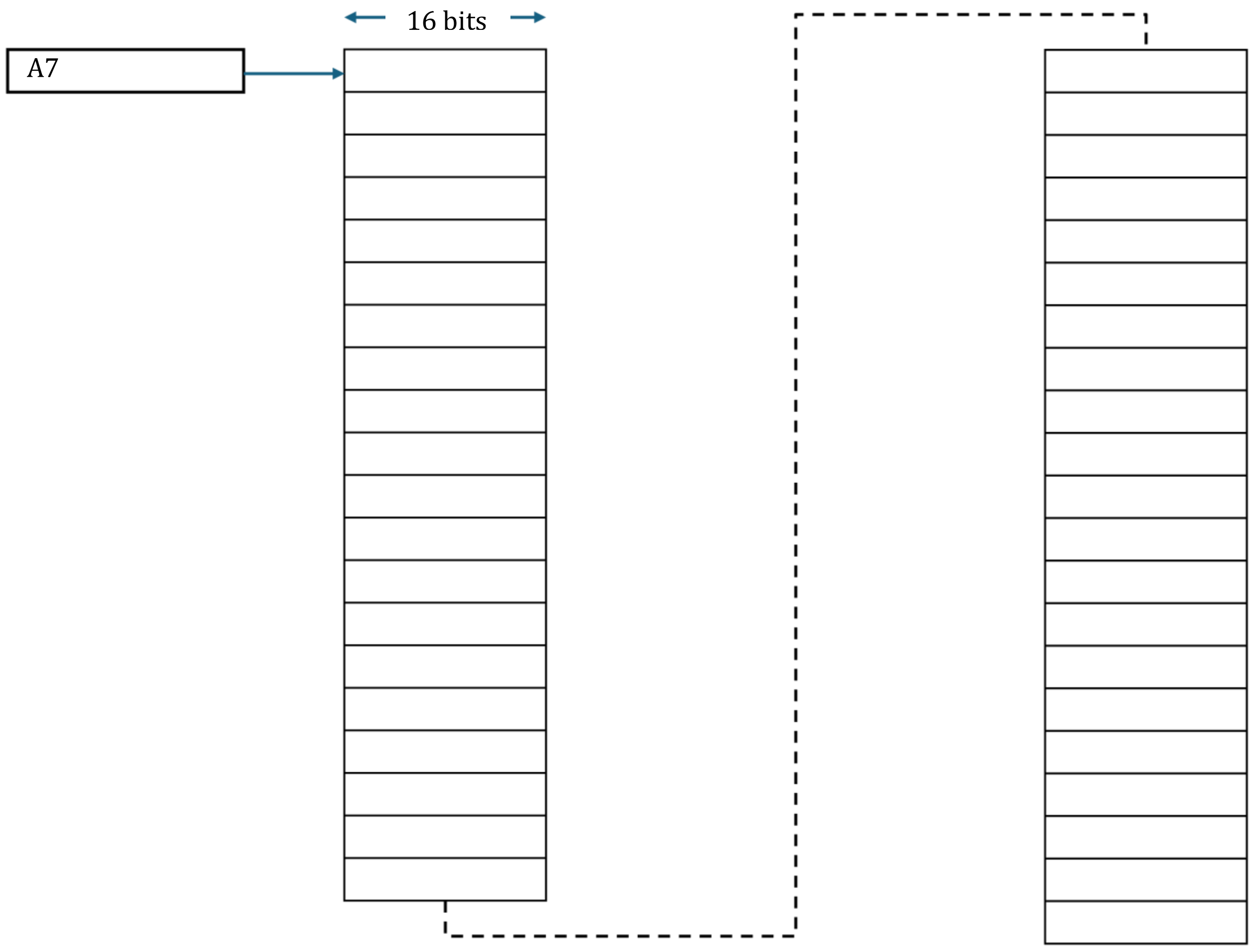
2. How many recursive calls were made in the program?

3. Each time a recursive call is performed the runtime stack increases in size. By how many bytes does it increase per call?

4. Reload and run the program a few more times with values of n > 5 until the result is incorrect. What is the maximum integer you may enter such that the subroutine still returns the correct value?

5. Given that the stack grows with each recursive call, it is possible for the stack to become exhausted. Assuming the stack size is 512K bytes, what is the minimum input value, n, that would exhaust the stack?

6. What factors do you think limit a factorial subroutine from calculating the factorial of very large numbers? Explain.


16 bits
A7