Brown University
School of Engineering
EN1640 Design of Computing Systems
Professor Sherief Reda
LAB 03 (100 points).
Due in lab to TA by Friday, Mar 7.

Please download and use the MARS simulator for the first four exercises. You should start on question 5 ASAP.

1. [10 points] Write a MIPS program to reverse (not inverse!) the bits in a register (e.g., 10111→11101). Use as few instructions as possible. Assume the register of interest is `$t3`. Use the MIPS simulator to test your code.

2. [10 points] Write a MIPS assembly code to test whether overflow occurs when $t2 and $t3 are added. Use as fewest instructions as possible. Use the MIPS simulator to test your code.

3. [15 points] Write a MIPS procedure that tests whether a given input string is a palindrome. The procedure should return 1 if the string is palindrome; otherwise, 0. (Recall that a palindrome is a word that is the same forward and backward. For example, the words "wow" and "racecar" are palindromes). Write an assembly program that test the procedure by calling it three times using the strings "wow", "racecar", and "sunshine" as inputs. Use the MIPS simulator to test your code.

4. [15 points] Each number in the Fibonacci series is the sum of the previous two numbers:

```
n:    1 2 3 4 5 6  7 ...
f(n): 1 1 2 3 5 8 13 ...
```

By definition the first two numbers in the series are equal to 1. Write a ***recursive*** procedure called `fib` in MIPS assembly language using the fewest instructions. Make sure your procedure respect the preserved register. Use the MIPS simulator to test your code. **Two points of this question will be allocated based on the number of instructions you use.**

5. [50 points] Your objective is to design a programmable stack-based calculator, which is also known as a postfix calculator, that can do 8-bit integer arithmetic. You have to include five instructions:

Instruction	Meaning
push *value*	push the immediate *value* into the top of the stack.
add	pops the two top elements in the stack and push the result of the addition to the stack.
sub	pops the two top elements in the stack and push the result of the subtraction to the stack.
mult	pops the two top elements in the stack and push the result of the multiplication to the stack.
halt	Stops execution and displays the top of the stack on the 7 segment display in decimal.

Here are two examples that show the operation of the calculator

Example A: 8-5*3	**Example B: 5 + ((1 + 2) * 4) – 3**
push 8 push 5 push 3 mult sub halt	push 5 push 1 push 2 add push 4 mult add push 3 sub halt

A. Design an instruction set format of this machine.
B. Translate the two example assembly programs into machine code using the format you created in part (A)
C. Write the Verilog code (+ schematic if applicable) for this machine and implement it on the DE2 board. Assume the following:
    - The machine has a stack of exactly 10 8-bit entries (users will never create programs that require more than 10 items in the stack).
    - One LED will turn on if there is an arithmetic overflow.
    - The machine code for programs is stored in a ROM. You will need to instantiate a ROM from the megafunction wizard and initialize its content from a .mif file. The .mif file can be created and edited by choosing File → New → Memory Files → Memory Initialization File.
    - When the machine boots, it start executing instructions from address 0 until a halt is encountered.
    - To save your effort, leverage the ALU design you created in Lab 1.
D. Please report (i) the total logic and routing resources using by your circuit; (ii) RTL circuit view; (iii) the actual critical path of your circuit and the propagation delay of your circuit critical path.
E. 10 points of this question will be determined upon your final design size in terms of LEs. You should not use any RAM elements to substitute for LEs.