Homework 8 Computer Organization Due: 12/9/11 (F) at 5PM

(You have your choice of doing either Booth's algorithm OR the set-of-letters assignment)

The goal of this assignment is to provide you with experience in using the "Logical and Shift Instructions" (bit-wise operations: and, andi, or, etc. & shifting instructions: sra, srav, etc.) and to help you understand Booth's multiplication algorithm. For this assignment, you are to implement Booth's multiplication algorithm on two 32-bit signed integers and get a 64-bit integer result.

I want you to write a function called "Multiply" that is passed two signed integers (in $a0 and $a1) and returns their 64-bit product across registers $v0 and $v1. Register $v0 should return the most-significant 32-bits of the product and $v1 should contain the least-significant 32-bits of the product. Be sure that you DO NOT use any form of the multiply (e.g., mul, mult, multu, etc.) assembly language instruction, and be sure to follow the MIPS register convensions when implementing "Multiply."

Your "main" program should:

1) interactively prompt for and read two integer values from the user by using the PCSpim system calls (pages 632 - 634 of text). Enter the value -30 for the multiplicand and 983 for the multiplier.
2) call your Multiply function with these values
3) use the PCSpim system call "print_int" to print the value of $v0 returned by Multiply
4) use the PCSpim system call "print_int" to print the value of $v1 returned by Multiply

The resulting "Console" window showing the interaction of the complete program should look something like:

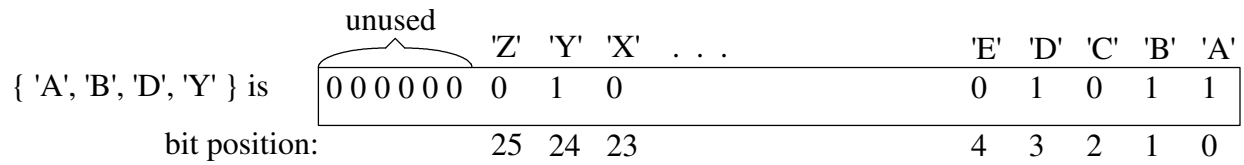
```
Enter the integer multiplicand: -30
Enter the integer multiplier: 983
The value returned in $v0 = -1
The value returned in $v1 = -29490
```

Notice that the PCSpim system call printed $v0's value as "-1" which is represented in two's-complement as all 1's. This is expected since the product (-29490) fits in the least-significant 32-bit value in $v1 with only the sign-extended 1's of a negative number carrying over into the most-significant 32-bits of $v0.

Turn in the following:

1) The MIPS assembly language programs (main and Multiply) for performing Booth's algorithm.
2) A snapshot of PCSpim containing the register values IMMEDIATELY after returning from your Multiply subprogram which was passed the integer value -30 for the multiplicand and 983 for the multiplier.
3) A snapshot of the Console window showing the complete interaction of your program using the value -30 for the multiplicand and 983 for the multiplier.

You are to build an abstract-data type (ADT) for the set of letters using a bit string. The bit string representation for the set of letters can use a 32-bit word with the least-significant bit associated with the letter 'A', etc.



The set of letters ADT should have the following operations (subprograms):

Subprogram Name	**Parameters**	**Description**
bitString	▪pass in a pointer to the an .ASCIIZ string ▪returns a word containing the set of letters as a bitString	Returns a bit string corresponding to the set of letters in the .ASCIIZ string. Non-letter characters are ignored, and both upper and lower-case letters should be represented as the upper-case letter.
union	▪passed two set bitStrings ▪returns the set union of the two sets	The resulting set should contain the elements that are in one or both of the input sets.
intersection	▪passed two set bitStrings ▪returns the set intersection of the two sets	The resulting set should contain the elements that are in both of the input sets.
difference	▪passed two set bitStrings ▪returns the set difference of the first set - second set	The resulting set should contain the elements that are in the first set, but not also in the second set.
contains	▪passed an .ASCII character and a set bitString ▪returns a Boolean (0 for false or 1 for true)	Returns 1 (true) if the .ASCII character is in the bitString set; otherwise return 0 (false).
print	▪passed an set bitString	Prints the bitString to the console using the print_string system call. The set should be printed in the conventional format, i.e., "{ E, G, T, Y }"

Additionally, you should have a main program that

1) allows a user to interactively enter two strings (use the PCSpim I/O),
2) constructs two bitString sets from these strings,
3) prints the set of letters contained in each string,
4) determines and prints the union, intersection, and difference of two bitString sets,
5) checks to see if the first bitString set contains the letters: 'A', 'Q', 'Y', and 'Z'. The results of each of these checks should be printed to the console.

Turn in the following:

1) The MIPS assembly language program. Use the MIPS register conventions correctly and include comments describing which registers are being used for parameters and local variables.
2) The "output" of the MIPS program. I want a picture of console window in the MIPS simulator **after** the program has run.