1. Use Booth's algorithm to calculate the 8-bit product of \(0110_2 \times 1011_2\).

   \[
   \begin{array}{c|c|c|c|c}
   \text{Multiplicand} & \text{Product register} & \text{Multiplier register} & \text{Previous bit} \\
   \hline
   0110 & 0000 & 1011 & 0 \\
   \hline
   \end{array}
   \]

2. You are to implement a function called "Multiply" that is passed two 32-bit signed integers and returns their 64-bit product. This function should use Booth’s algorithm only, i.e., sure that you DO NOT use any form of the multiply (e.g., mul, mult, multu, etc.) assembly language instruction.
   a) Since you want to follow the MIPS register conventions, what registers should be used to pass the two integers into Multiply?

   b) What registers should be used to return the 64-bit product?

   c) If Multiply does not call any subprograms(/procedures/functions/methods), what type of registers should we try to use?

   d) How many times will you need to loop if we are checking one bit of the multiplier (and the previous bit) each iteration of the loop?
$a0$, Multiplicand

| 0 | 1 | 1 | 0 |

Product register

$\text{ Previous bit}$

| 0 | 0 | 0 | 0 |

Multiplier register

$\text{ For (- Multiplicand), we'll subtract } a0$

| 0 | 0 | 0 | x |

| 0 | 0 | 0 | y |

$\text{ v1}$

$\text{ v0}$

$\text{ t4}$

$\text{ t6}$

$\text{ t5}$

$\text{ t6}$

\[\begin{align*}
\text{ e)} & \text{ How would you set the previous bit in } t4 \text{ to zero?} \\
\text{ f)} & \text{ How would you get the LSB of the multiplier into } t5? \\
\text{ g)} & \text{ How would you implement the following if-statements?} \\
& \text{ if } \text{ LSB of multiplier (in } t5) \neq \text{ previous bit (in } t4) \text{ then} \\
& \quad \text{ if previous bit (in } t4) == 1 \text{ then} \\
& \quad \quad \text{ } v1 = v1 + a0 \\
& \quad \quad \text{ else} \\
& \quad \quad \quad v1 = v1 - a0 \\
& \quad \text{ end if} \\
& \text{ end if} \\
\text{ h)} & \text{ How would you get the LSB of the } v1 \text{ into } t6? \\
\text{ i)} & \text{ How would you shift the LSB of } t6 \text{ to be the MSB of } t6? \\
\text{ j)} & \text{ Now, how would you set the MSB of the } v0 \text{ (multiplier reg.) to the MSB of } t6? \\
\text{ k)} & \text{ How would you shift } v1 \text{ to the right one bit position?}
\end{align*}\]
2) Complete the following Multiply code:

```
# Programmer: Mark Fienup
# Booth's algorithm implementation using shifting & logic instructions
.data
multiplicand: .word 103
multiplier: .word -25
result_upper: .word 0
result_lower: .word 0

.text
.globl main
main:
    lw $a0, multiplicand # load multiplicand into $a0
    lw $a1, multiplier # load multiplier into $a1
    jal multiply # call multiply subprogram
    sw $v0, result_lower # store 64-bit result
    sw $v1, result_upper
    li $v0, 10 # exit system call
    syscall
endMain:

multiply:
```

3/30/06