So far in MIPS, we have seen how to multiply by a power of 2: simply do a left shift by the appropriate number of bits. Thus,

\[
x: \text{.word 34}
\]

\[
\ldots
\]
\[
\text{lw} \quad \$t0,x \quad \# \text{load the value } x=34 \text{ into register } t0
\]
\[
\text{sll} \quad \$t0,$t0,3 \quad \# \text{shift left logical by 3 = multiply by 8}
\]
\[
\text{sw} \quad \$t0,x \quad \# \text{save the value } 272 = 34\times8 \text{ into } x
\]

We have also seen how to divide by 2: do an arithmetic shift right (the sign bit is “extended”). Thus,

\[
x: \text{.word 34}
\]

\[
\ldots
\]
\[
\text{lw} \quad \$t0,x \quad \# \text{load the value } x=34 \text{ into register } t0
\]
\[
\text{sra} \quad \$t0,$t0,3 \quad \# \text{shift right arithmetic by 3 = divide by 8}
\]
\[
\text{sw} \quad \$t0,x \quad \# \text{save the value } 4 = 34/8 \text{ into } x
\]

This is all we need to do multiplication of any pair of nonnegative numbers! (Actually, negative, too, but the algorithm below needs one small fix to work for that case.)

Here’s a C function to multiply any pair of nonnegative integers using only addition and multiplication and division by 2:

\[
\text{int mult(int a, int b)}
\]
\[
\{
\text{  int p; \quad /* temporary for holding product */}
\text{  if (a == 0 || b == 0) \quad /* base case */}
\text{      return 0;}
\text{  }
\text{  int q = a; \quad /* temporary for holding } a/2 \text{ */}
\text{  while (b > 0)}
\text{      if (b & 1)}
\text{         p += q; \quad /* add } a/2 \text{ if } b \text{ is odd */}
\text{      b >>= 1; \quad /* divide } b \text{ by 2 */}
\text{      q <<= 1; \quad /* double } a/2 \text{ */}
\text{  }
\text{  return p;}
\}
\]
\[ p = 2\text{\texttt{mult}}(a, b/2); /* recursive call */ \]

\[
\text{if } (b\%2 != 0)
\]
\[
p = p + a; /* add one more copy of a */
\]

\[
\text{return } p;
\]

It is recursive. (NOTE: we can do a \text{"\%\"} operation in MIPS by checking to see if \( b/2 \times 2 == b \) or by doing a logical \text{"and\"} between \( b \) and 1 to see if the result is non-zero.)

Write a fully-commented MIPS program that can multiply together two nonnegative integers using the recursive algorithm given above.

Your program should load the two values directly into the argument registers $a0$ and $a1$ and then call a function at location \text{"mult:\"}. The product should be stored in register v0 just prior to returning from the function. Upon return to the main program, the result should be saved in a memory location labeled \text{result}. Your program should then print the value of the result, with an appropriate label.

Here’s a model of the program (but you have to supply comments!). In this program, we are computing the product 8271 \times 3621:

```
.data
result: .word 0

.text
li $a0,8271
li $a1,3621
jal mult
sw $v0,result
    ... print the result ...
li $v0,10
syscall
```

Be sure to preserve any needed variables or registers on the stack prior to making the recursive call. Be sure to restore the stack to its original state prior to returning.

Run your program with at least five pairs of values (you’ll need to change the \text{li instructions} each time and re-assemble the program). If you are ambitious, you can input these values from the user at run-time.

Paste the outputs into a document and print it out. Hand in your program and outputs (Courier font, please!). Upload your program to Sakai.

NOTE: this is not the method used by the hardware to perform multiplication!