

8bit * 8bit = 16bit multiply

```
;-----  
; 8bit * 8bit = 16bit multiply  
; Multiplies "num1" by "num2" and stores result in .A (low byte, also in .X)  
and .Y (high byte)  
; uses extra zp var "num1Hi"  
  
; .X and .Y get clobbered. Change the tax/txa and tay/tya to stack or zp  
storage if this is an issue.  
; idea to store 16-bit accumulator in .X and .Y instead of zp from bogax  
  
; In this version, both inputs must be unsigned  
; Remove the noted line to turn this into a 16bit(either) * 8bit(unsigned) =  
16bit multiply.  
  
lda #$00  
tay  
sty num1Hi ; remove this line for 16*8=16bit multiply  
beq enterLoop  
  
doAdd:  
clc  
adc num1  
tax  
  
tya  
adc num1Hi  
tay  
txa  
  
loop:  
asl num1  
rol num1Hi  
enterLoop: ; accumulating multiply entry point (enter with .A=lo, .Y=hi)  
lsr num2  
bcs doAdd  
bne loop  
  
; 26 bytes  
  
--  
White Flame (aka David Holz)  
http://www.white-flame.com/
```

Last update: 2015-04-17 04:30 base:8bit_multiplication_16bit_product https://codebase64.org/doku.php?id=base:8bit_multiplication_16bit_product&rev=1429237809

From:
<https://codebase64.org/> - **Codebase 64 wiki**

Permanent link:
https://codebase64.org/doku.php?id=base:8bit_multiplication_16bit_product&rev=1429237809

Last update: **2015-04-17 04:30**

