In this assembly program we’re are going to be multiplying a 16 bit by an 8 bit without using the mul instruction. Using iterative addition to perform the above multiplication.

The multiplicand is stored in 2 registers: r24(lower byte) and r25(high byte). The multiplier is stored in r22. The result(product) will be stored in three registers: r18, r19, and r20. The order I stored the result is r18 holds the lower byte of the results and r19 and r20 will hold the high bytes.

My logic to approach this program is that I will add r24 and r25 by the amount of times by the multiplier. The lower byte (r24) will be stored in r18 and if there’s a carry bit it will be stored in r19. The high byte of the multiplicand will be stored in r19 and if there’s a carry bit it will be stored in r20. I created a label named “addlp” which is where the program will continue to add the values in r24 and r25 as many times the multiplier is set. Every time the multiplicand is added it will decrement the value of the multiplier and will return back to the beginning of the label “addlp” until the multiplier value is zero. If the multiplier in the beginning was zero, then before the programs enters the “addlp” label it will be compared to another register that holds a zero value. If it is zero, then it will skip the “addlp” label and will go to a jump instruction to go to the end of the program.

In my program I multiplied 0x0A0D by 0x05 which equals to 0x3441. So my expected value should be stored in r18 and r19. In r18 should hold 0x41 and in r19 should hold 0x34. My clock cycle was 42 and my stop watch was 2.63µs at 16MHz frequency.

Assembly Code:

.org 0x0000

start:

ldi r16, 0 ; holds a zero

ldi r17, 0 ; will hold a copy of the multiplier

ldi r18, 0 ; register for the product

ldi r19, 0 ; register for the product

ldi r20, 0 ; register for the product

ldi r22, 0x05 ; multiplier

ldi r24, 0x0D ; low byte of the multiplicand

ldi r25, 0x0A ; high byte of the multiplicand

mov r17, r22 ; makes a copy of r22

cpse r17, r16 ; if r17 equals zero then it will skip the next instruction

jmp addlp ; if r17 doesn't equal to zero then it will go to this instruction

; condition for this loop is that it will keep looping the amount of times the multipier is set.

jmp done ; will jump to the done label

; loop to add both the low and high bytes

addlp:

add r18, r24 ; adds the low byte of the multiplicand into the register that holds the product

adc r19, r25 ; will add (if there is a) carry to the next product register

adc r20, r16 ; will add (if there is a) carry to the next product register

dec r17 ; decrements the multiplier every time it loops

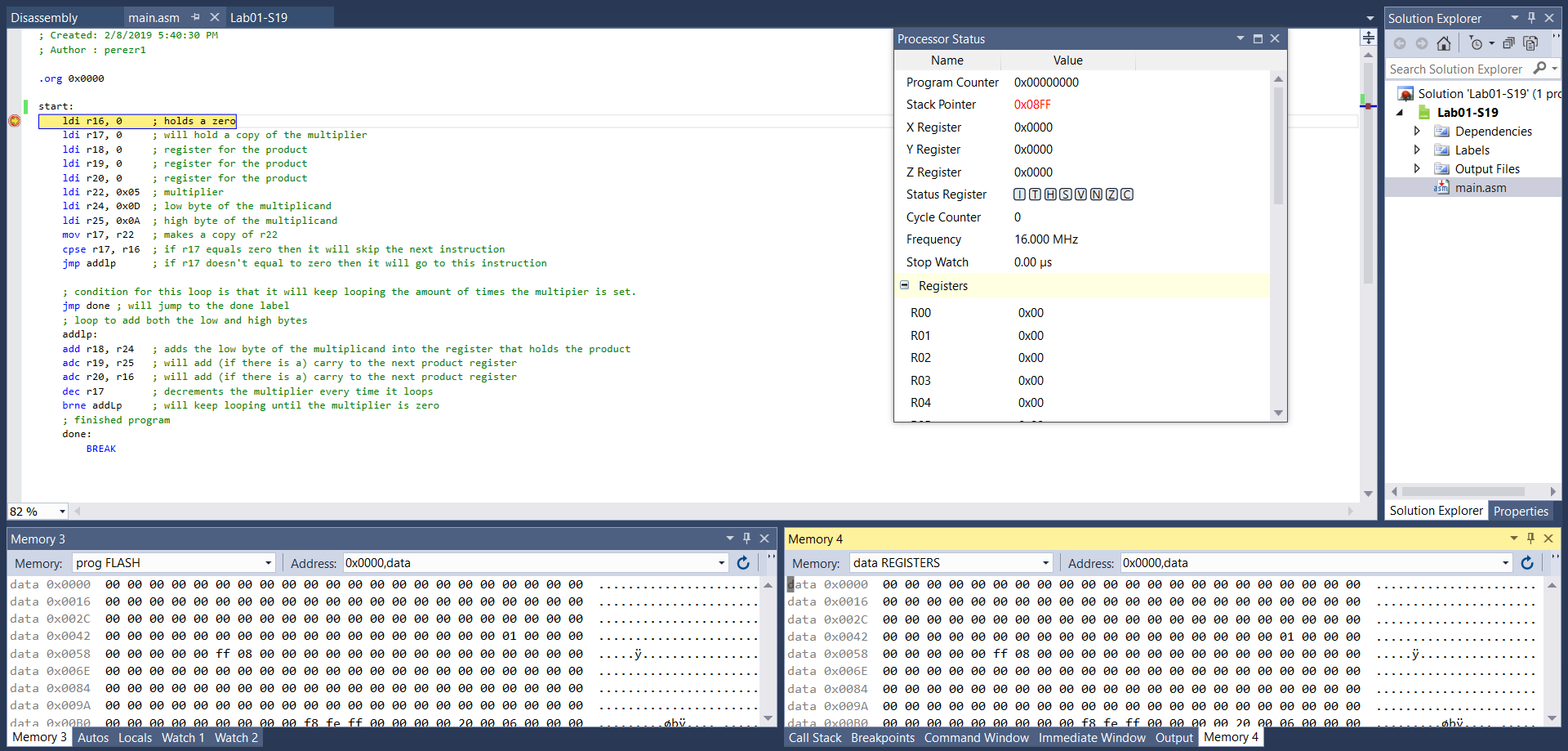
brne addLp ; will keep looping until the multiplier is zero

; finished program

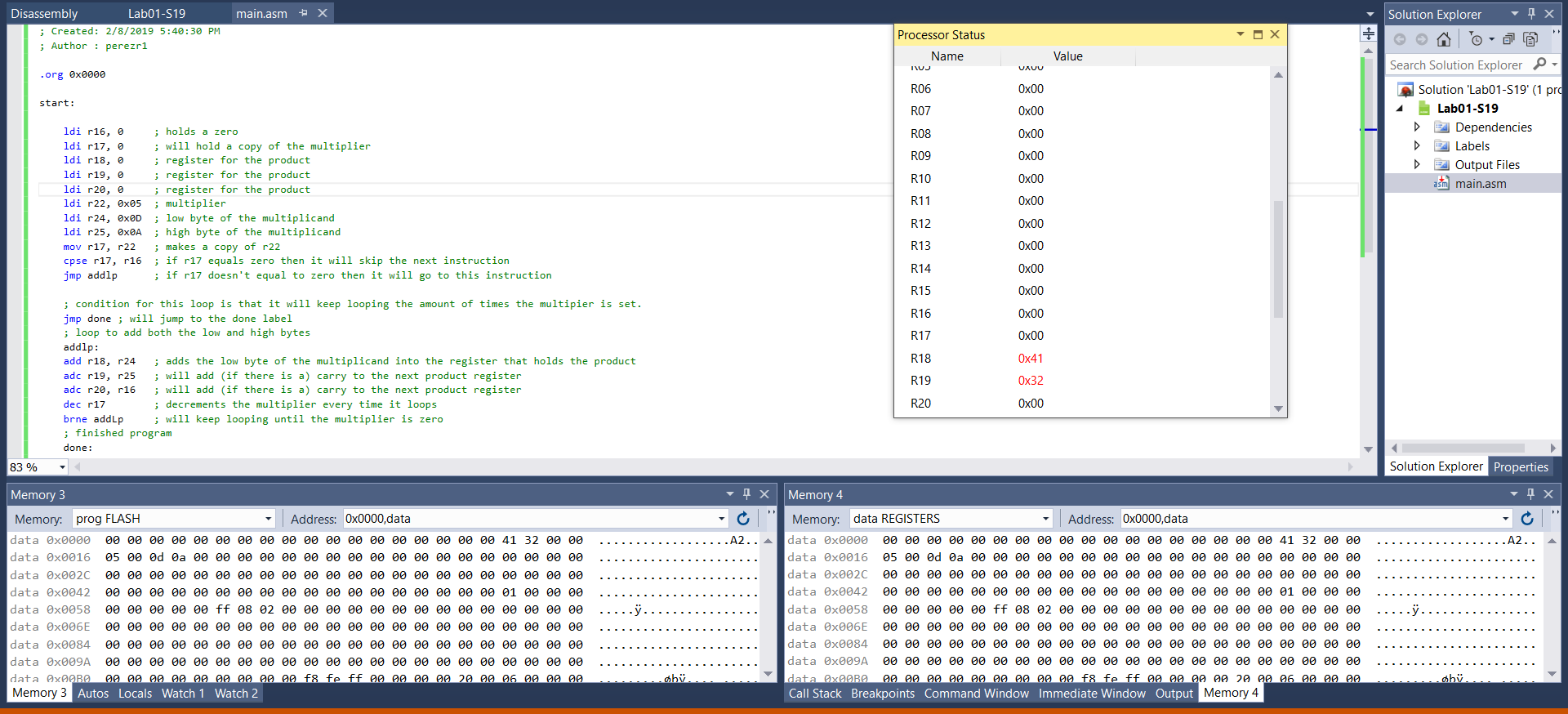
done:

BREAK

Beginning of the program @16MHz:



End Of program@16MHz:



Processor Status at the end of the program:

|  |  |
| --- | --- |
|  |  |