In order to multiply two unsigned numbers without using mul instruction it is necessary to perform two unsigned numbers addition as many times as the value of the multiplier is. For example if we are going to multiply 5 by 3 we will need to add 5 + 5 + 5, in first step we add 5 to 0 then 5 to the current value of the sum which is 5, and finaly 5 to 10 which is current value of the sum. There are 3 additions in total necessary to perform this operation.

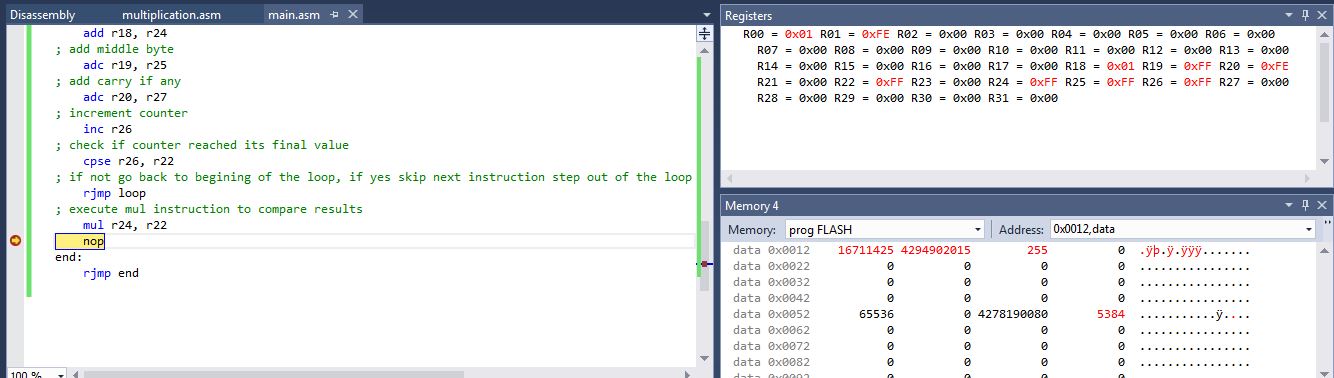
This is done in the loop block of the code, registers R18, R19 and R20 are initialized with value 0, register R26 is used as counter, in every loop iteration register R26 is incremented and its value is compared with the value of the multiplier, if the values are equal multiplication process is finished and program will break from the loop.

Inside the loop addition is performed in the following way: lower byte of the multiplicand is added to register R18 value with add instruction, this addition could produce carry bit, ie C bit of the status register is set, this means that next addition (higher byte of the multiplicand and R19) must be done using adc instruction, adc instruction will add higher byte of the multiplicand (R25) with R19 and carry bit. This additon also could produce carry bit, so next addition of R20 (which holds the most significant byte of the result) and R27 (which is set to 0 and it is used only to allow adc instruction to be executed on R27) must be performed again using adc instruction. This way result number is assembled in registers R18, R19 and R20.

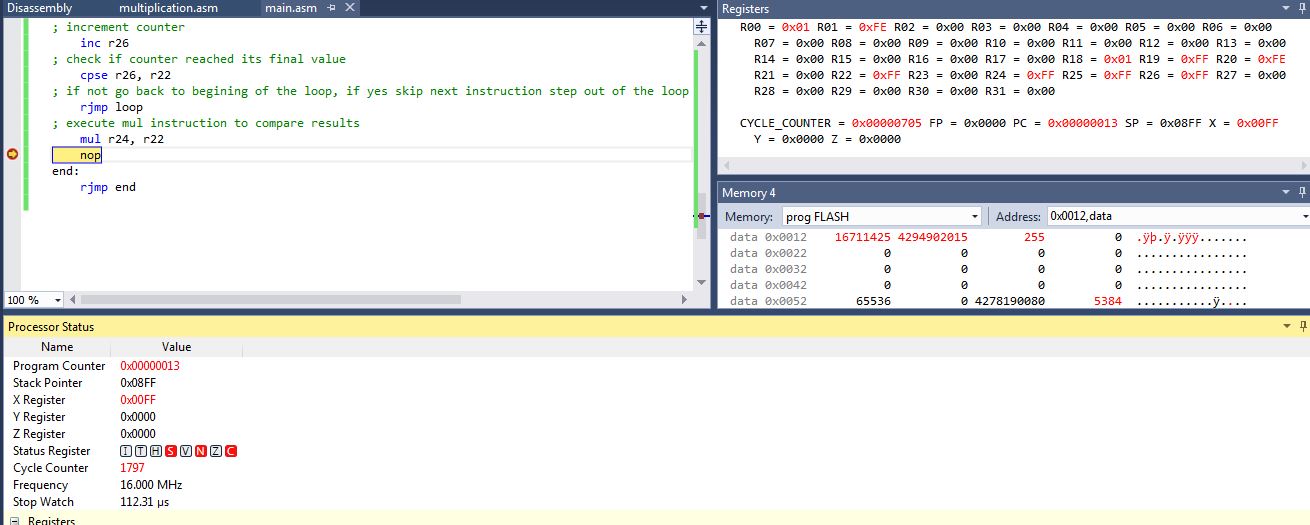
While debuging Atmel studio memory window can be set to show register values in form of unsigned 4-byte integers, right click inside the window and then mark 4-byte integer and Unsigned display. It is easier this way to read assembled value in registers R18 – R20. Picture below shows results of the program execution. Register R22 has value of 225 (multiplier), register R24 and R25 have value of 255 in each (multiplicant is 16 bit that in this example all bits are 1 (65535) to prove the result).

The result is stored at adress 0x0012 which is address of R18 register, as shown below

65535\*225 =16711425.



The following screenshot shows execution time (112.31 us) and number of cycles (1797) at frequency of 16MHz.



Complete code with all comments is shown below:

;

; multiplication.asm

;

; Created: 08.02.2019. 23:08:51

; Author : Ali Asadi

;

.macro multiplicand\_l

cbr @0, $FF

sbr @0, low(@2)

cbr @1, $FF

sbr @1, high(@2)

.endm

start:

multiplicand\_l r24, r25, 65535 ; THIS IS WHERE MULTIPLICAND VALUE CAN BE CHANGED

; Clear R22 register and assign 8-bit multiplier value

cbr r22, $FF

sbr r22, 255

; Clear registers R18, R19 and R20, this registers are going to hold multiplication result

cbr r18, $FF

cbr r19, $FF

cbr r20, $FF

; Clear register R26, R26 is used as a counter, in every loop iteration R26 is incremented

; then its value is compared with value of multiplier

cbr r26, $FF

sbr r26, 0

; Clear R27 register, R27 is added to R20 which holds the most significant byte of the result.

; R27 is set to 0 and it is used in order to perform adc instruction with R20 which means if carry (C) flag

; is set as a result of previous adc instruction (adc r19, r25), instruction adc r20, r27 will push carry bit into

; the R20 register, in every next itteration carry bit (if any) will be added to R20 register

cbr r27, $FF

; loop block, itterate addition as many times as multiplier value is

; example: if multiplier value is 5 and multiplicand value is 3

; we need to add 3 five times in order to multiply numbers.

; R18, R19 and R20 are cleared, first we add least significant byte

; of the multiplicand to R18 without carry (add) because this is first addition

; next we add middle byte of the multiplicand to R19 together with carry bit

; if instruction add r18, r24 set carry flag

; and finaly we add 0 to R20 together with carry bit if previous addition produced carry bit

; at the end of the loop block R26 which is used as a counter is incremented and compared to

; multiplier value, if they are equal programs steps out of the loop.

loop:

; add lowest byte

add r18, r24

; add middle byte

adc r19, r25

; add carry if any

adc r20, r27

; increment counter

inc r26

; check if counter reached its final value

cpse r26, r22

; if not go back to begining of the loop, if yes skip next instruction step out of the loop

rjmp loop

; execute mul instruction to compare results

mul r24, r22

nop

end:

rjmp end

To verify the result, I have written the following C code:

/\*

\* MUL.c

\*

\* Created: 10.02.2019. 21:28:39

\* Author : Ali Asadi

\*/

#include <avr/io.h>

int main(void)

{

volatile unsigned int multiplier = 255;

volatile unsigned int multiplicand = 65535;

volatile unsigned long result;

result = (long int)multiplicand\*multiplier;

result = result;

}

The following screenshot verifies the result of mutipilication.

