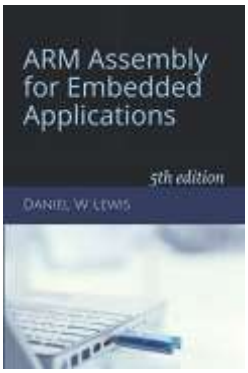


Programming Lab 7J

Signed Magnitude

Topics: Bitwise and bitfield operations, arithmetic with signed magnitude integers.

Click to download Lab7J-Main.c

Prerequisite Reading: Chapters 1-7

Revised: June 26, 2021

***Background*[1]:** Signed magnitude represents the sign and magnitude parts of an integer separately, similar to how we write integers by hand by placing a minus sign or optional plus sign in front of the magnitude. Although simpler for humans, signed magnitude complicates the hardware implementation of arithmetic operations:

$C \leftarrow Add(A,B)$: $if\ sign(A)=sign(B)$ $mag(C) \leftarrow mag(A)+mag(B),\ sign(C) \leftarrow sign(A)$
$else\ if\ mag(A) > mag(B)$ $mag(C) \leftarrow mag(A)-mag(B),\ sign(C) \leftarrow sign(A)$
$else$ $mag(C) \leftarrow mag(B)-mag(A),\ sign(C) \leftarrow sign(B)$

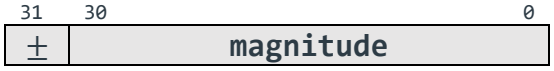
$C \leftarrow Sub(A,B)$: $mag(Temp) \leftarrow B,\ sign(Temp) \leftarrow \sim sign(B),\ C \leftarrow Add(A,Temp)$

$C \leftarrow Mul(A,B)$: $mag(C) \leftarrow mag(A) \times mag(B),\ sign(C) \leftarrow sign(A) \oplus sign(B)$

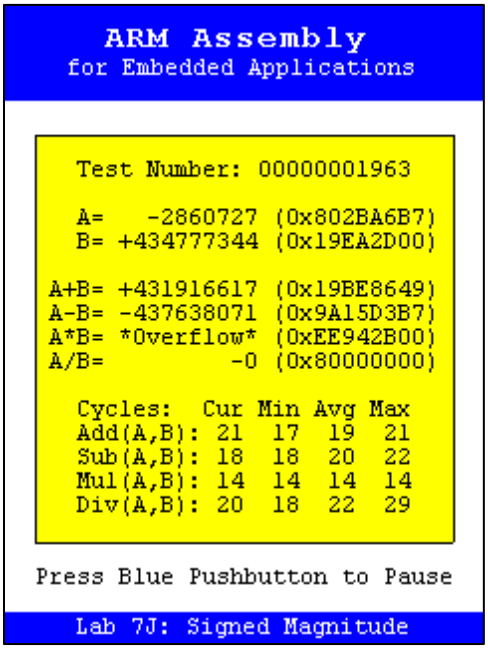
$C \leftarrow Div(A,B)$: $mag(C) \leftarrow mag(A) \div mag(B),\ sign(C) \leftarrow sign(A) \oplus sign(B)$

Assignment: The main program implements these operations as the following C functions:

```
typedef uint32_t smag32_t ; // 32-bit container:
```




```
smag32_t Add(smag32_t a, smag32_t b) ; // returns a+b
smag32_t Sub(smag32_t a, smag32_t b) ; // returns a-b
smag32_t Mul(smag32_t a, smag32_t b) ; // returns a*b
smag32_t Div(smag32_t a, smag32_t b) ; // returns a/b
```



The main program will compile and run without writing any assembly. However, your task is to create equivalent assembly language replacements for these functions. Since computing uses arithmetic operations extensively, your code should require as few clock cycles as possible. The original C versions have been defined as "weak" and will be automatically replaced in the executable image by those you create in assembly; you do not need to remove the C versions. This feature allows you to write and test one assembly language function at a time.

Test your functions using the C main program. If your code is correct, the program will continuously test each function with randomly selected operands. The execution time of each of your functions is displayed in clock cycles. An incorrect result will be displayed as **white text on a red background**, pause the program, and wait for you to press the pushbutton to proceed.

[1] https://en.wikipedia.org/wiki/Signed_number_representations#Signed_magnitude_representation