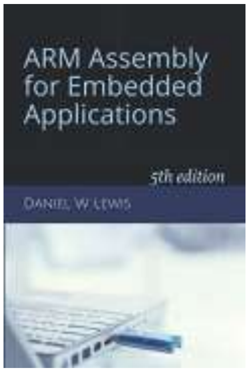


Programming Lab #6i

Simulating a Full Adder

Topics: Integer arithmetic, bitwise and shift instructions

Prerequisite Reading: Chapters 1-7

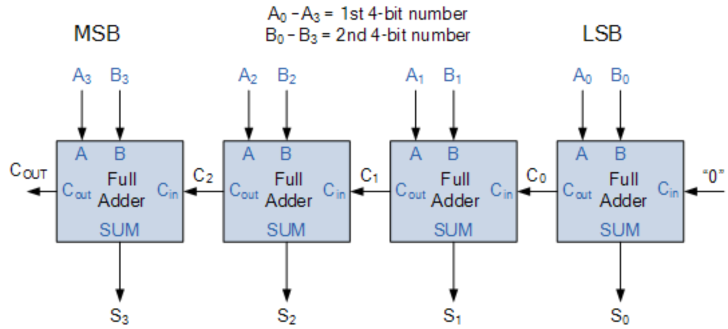
Revised: December 6, 2020

Background: A full adder is a logic circuit that adds two one-bit binary numbers (A_i and B_i) and a carry-in (C_i), producing a sum represented by a sum bit (S_i) and carry-out (C_{i+1}). A device to add two 4-bit numbers may be constructed by cascading four full adders, connecting the carry-out of one to the carry-in of the next.

The hardware implementation of the sum and carry outputs is usually defined in *Boolean Algebra* as:

$$S_i = A_i \oplus B_i \oplus C_i$$

and $$C_{i+1} = A_iB_i + A_iC_i + B_iC_i$$



The objective of this lab is to simulate these functions in software. Note that their functionality may be implemented in several different ways as illustrated by the following pseudocode examples using C operators. *(Hint: The arrays shown in Sum #4 and Cout #4 may be created with the assembler* `.byte` *directive.)*

Sum #1: $S_i = A_i \,\hat{}\, B_i \,\hat{}\, C_i$

Cout #1: $C_{i+1} = (A_i \& B_i) \mid (A_i \& C_i) \mid (B_i \& C_i)$

Sum #2: $S_i = (A_i + B_i + C_i) \,\&\, 1$

Cout #2: $C_{i+1} = (A_i + B_i + C_i) \gg 1$

Sum #3: $shift = (A_i \ll 2) \mid (B_i \ll 1) \mid (C_i)$

$S_i = (10010110_2 \gg shift) \,\&\, 1$

Cout #3: $shift = (A_i \ll 2) \mid (B_i \ll 1) \mid (C_i)$

$C_{i+1} = (11101000_2 \gg shift) \,\&\, 1$

Sum #4: $index = (A_i \ll 2) \mid (B_i \ll 1) \mid (C_i)$

$S_i = \{0,1,1,0,1,0,0,1\}[index]$

Cout #4: $index = (A_i \ll 2) \mid (B_i \ll 1) \mid (C_i)$

$C_{i+1} = \{0,0,0,1,0,1,1,1\}[index]$

Assignment: You are to implement each of these eight alternatives in ARM assembly as straight-line functions with no `IT` or conditional branch instructions, and defined by the following function prototypes. Although the parameters and return types are all declared as 32-bit signed integers, they values they hold are either decimal 0 or 1.

```
int32_t Sum1(int32_t Ai, int32_t Bi, int32_t Ci) ;
int32_t Sum2(int32_t Ai, int32_t Bi, int32_t Ci) ;
int32_t Sum3(int32_t Ai, int32_t Bi, int32_t Ci) ;
int32_t Sum4(int32_t Ai, int32_t Bi, int32_t Ci) ;

int32_t Cout1(int32_t Ai, int32_t Bi, int32_t Ci) ;
int32_t Cout2(int32_t Ai, int32_t Bi, int32_t Ci) ;
int32_t Cout3(int32_t Ai, int32_t Bi, int32_t Ci) ;
int32_t Cout4(int32_t Ai, int32_t Bi, int32_t Ci) ;
```

Test your functions using the main program downloaded from here. Your functions are used to implement six additions as shown on the right. The values of the two 4-bit integers *A* and *B* continuously cycle through all possible combinations unless an error is encountered. Correct values of the sum and carry bits produced by your functions are displayed in green; incorrect values are displayed in red.