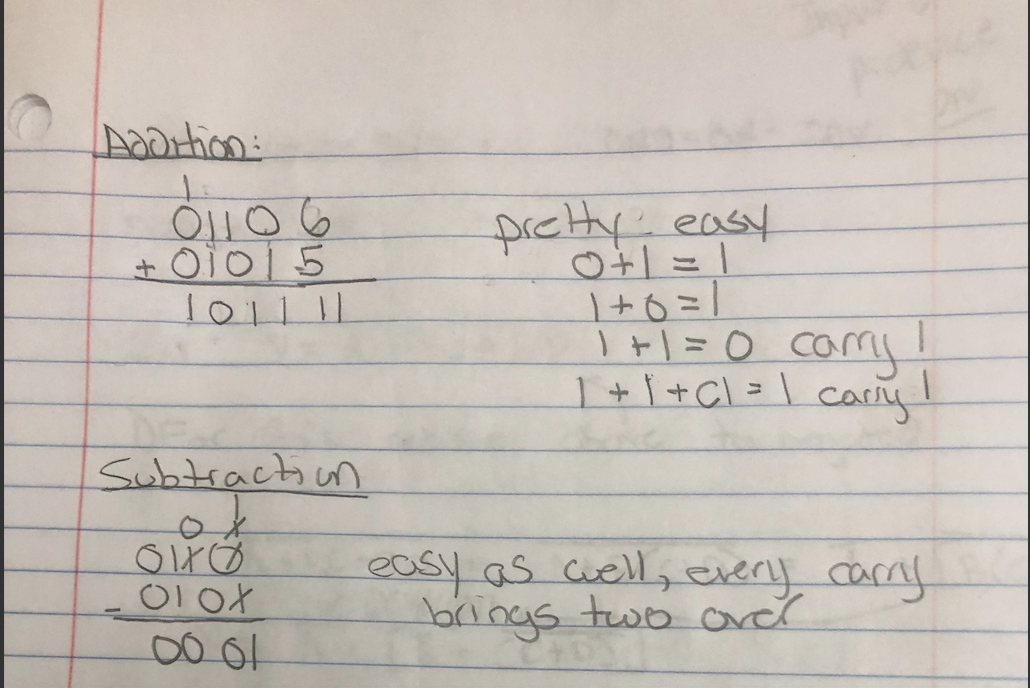
**Chapter 3: Arithmetic For Computers:**

**Addition and Subtraction:**

****

**OVERFLOW:**

* Adding two numbers of **different** signs **does not** yield overflow
* Subtracting two numbers of **same** sign **does not** yield overflow

Overflow - when the number of available bits is not sufficient

In addition - when addition causes the carry out into a sign bit

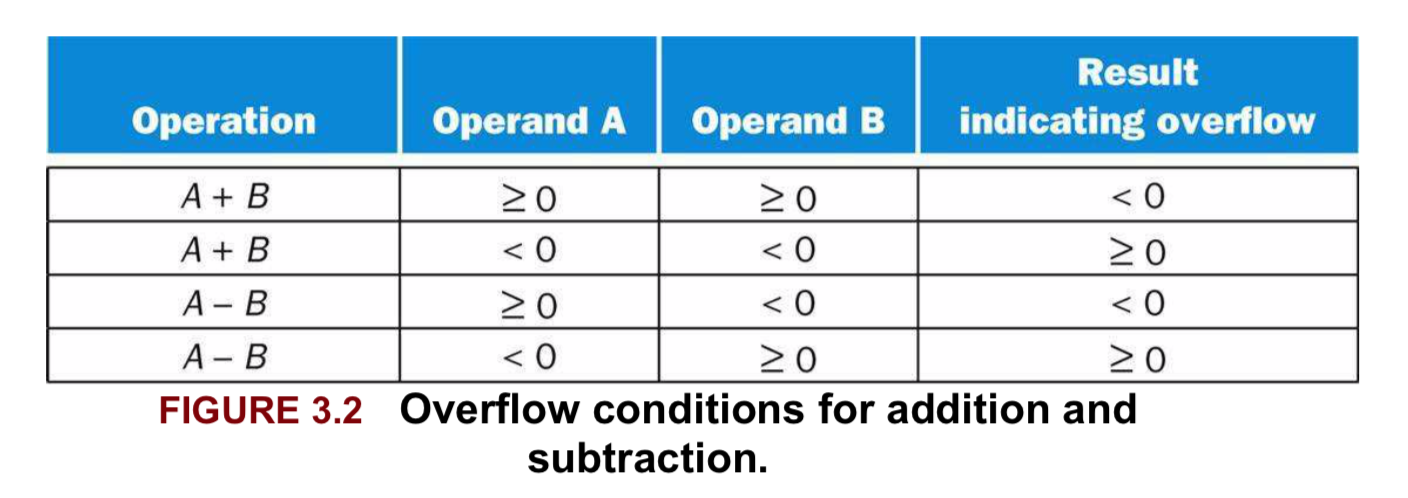
In subtraction - when subtraction causes borrow into MSB

TextBook:

Clearly, adding or subtracting two 64-bit numbers can yield a result that needs 65 bits to be fully expressed.

The lack of a 65th bit means that when an overflow occurs, the sign bit is set with the *value* of the result instead of the proper sign of the result. Since we need just one extra bit, only the sign bit can be wrong. Hence, overflow occurs when adding two positive numbers and the sum is negative, or vice versa. This spurious sum means a carry out occurred into the sign bit.

Overflow occurs in subtraction when we subtract a negative number from a positive number and get a negative result, or when we subtract a positive number from a negative number and get a positive result. Such a ridiculous result means a borrow occurred from the sign bit. Figure 3.2 shows the combination of operations, operands, and results that indicate an overflow.



Detection:

* Instruction like “add” cause an interruption (exception)
* When overflow occurs, it interrupts, and control jumps to predefined memory address for exceptions
* The address of instruction causing the overflow for possible resumption
* The offending address is stored in exception program counter
  + **Sepc** - which is not part of the register file
  + **Scause** - stores the cause of the interrupt
* To debug we can move the content of of the SEPC into the general purpose register

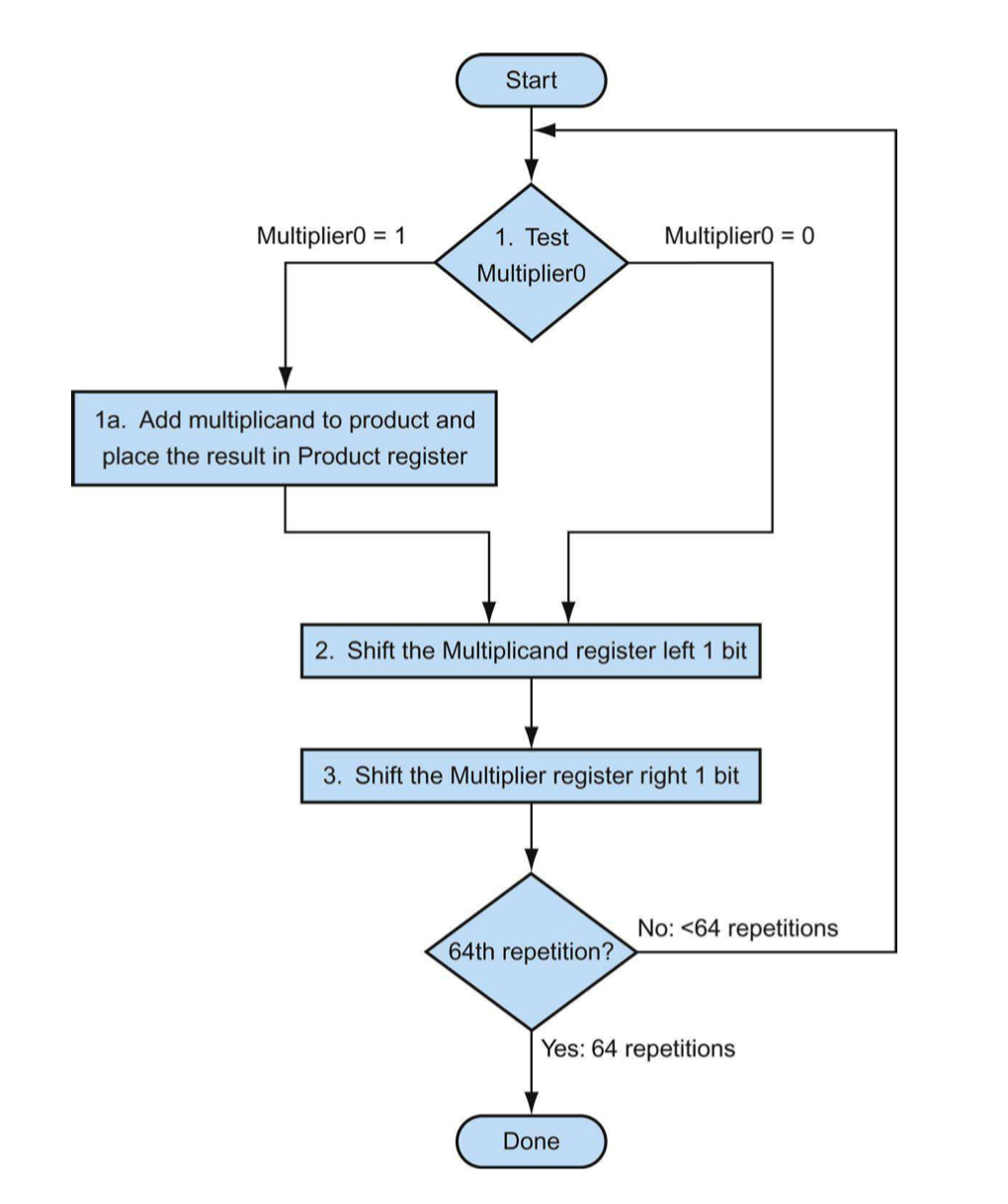
**Multiplication:**

The first operand is called the *multiplicand* and the second the *multiplier*. The final result is called the *product*.

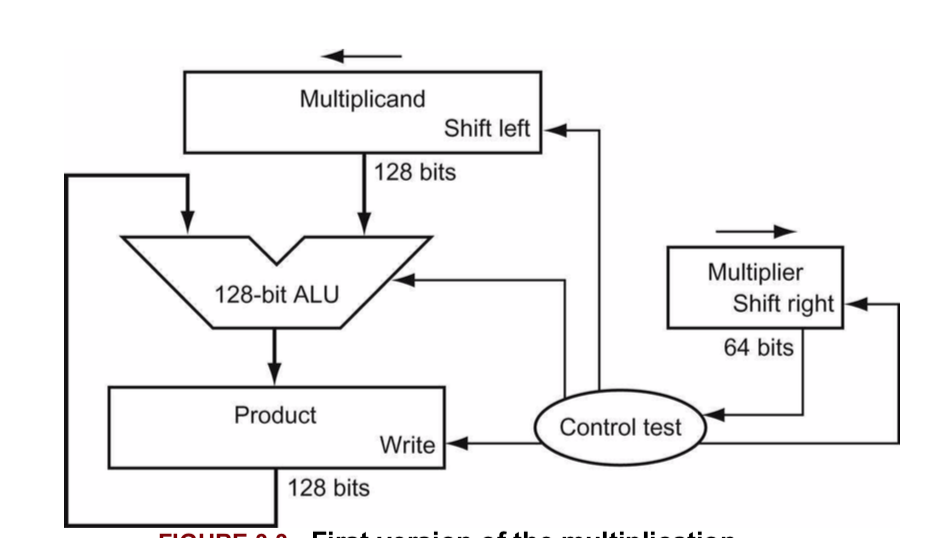
Binary Multiplication:

* A Bunch of right shifts and adds
* When multiplying two 64 bit numbers you get a 128 bit number
* Overflow is checked in software

Unoptimized Binary Multiplication:



HardWare:



Example:

