CS-228: Discrete Structures II: REPRESENTATION of NUMBERS in DIGITAL COMPUTERS

Charles Abzug, Ph.D.
Department of Computer Science
James Madison University
Harrisonburg, VA 22807

Voice Phone: *540-568-8746;* Cell Phone: *443-956-9424*
E-mail: *abzugcx@JMU.edu* OR *CharlesAbzug@ACM.org*
Home Page: *http://www.cs.jmu.edu/users/abzugcx*

Outline

1. Number Systems: Basic Concepts
2. Interconversion
3. Binary Numbers
4. Arithmetic
5. Broadening Our Horizons:
   - Different Forms of Integer Number Representation
   - Performance of Arithmetic in Each Representation Scheme

Number Systems

a. Positional Number Notation
   Example: the integer 603,550

   6<u>0</u>3,55<u>0</u>

   Question 1: Are the two zeroes equivalent?

   603,<u>55</u>0

   Question 2: Are the two fives equivalent?

Number Systems

a. Positional Number Notation
   Example: the integer 603,550

   6<u>0</u>3,55<u>0</u>

   Question 1: Are the two zeroes equivalent?
   Answer 1: Yes AND no: on the one hand 0 = 0; nevertheless, the right-hand 0 signifies no units, the other 0 signifies no ten-thousands.

   603,<u>55</u>0

   Question 2: Are the two fives equivalent?
   Answer 2: Similarly, 5 = 5; however, the right-hand 5 signifies five tens, the left-hand five signifies five hundreds.

Number Systems

a. Positional Number Notation

b. Radix or Base

Example_1: $603{,}550_{10} = 0 \times 10^0 + 5 \times 10^1 + 5 \times 10^2 + 3 \times 10^3 + 0 \times 10^4 + 6 \times 10^5$

Example_2: $603{,}550_{16} = 0 \times 16^0 + 5 \times 16^1 + 5 \times 16^2 + 3 \times 16^3 + 0 \times 16^4 + 6 \times 16^5 = 3{,}905{,}404_{10}$

Number Systems

a. Positional Number Notation

b. Radix or Base
   In general, $603{,}550_{\text{Radix } x} \neq 603{,}550_{\text{Radix } y}$

Number Systems

a. Positional Number Notation

b. Radix or Base

$603{,}550_{\text{Radix } x}$:

QUESTION: Without knowing the value of '*x*', what conclusion can we draw about '*x*' just by looking at the number?

Number Systems

a. Positional Number Notation

b. Radix or Base

Default Radix vs. Specified Radix

Number Systems

a. Positional Number Notation

b. Radix or Base

c. Why decimal?

Number Systems

a. Positional Number Notation

b. Radix or Base

c. Why decimal? Number of Fingers

Number Systems

a. Positional Number Notation

b. Radix or Base

c. Why decimal? Number of Fingers

Other radices in use in various civilizations:

Duodecimal (R = 12)

Number Systems

a. Positional Number Notation

b. Radix or Base

c. Why decimal? Number of Fingers

Other radices in use in various civilizations:

Duodecimal (R = 12): Finger-Joints excluding Thumb

Number Systems

a. Positional Number Notation

b. Radix or Base

c. Why decimal? Number of Fingers
   Other radices in use in various civilizations:
   Duodecimal (R = 12): Finger-Joints excluding Thumb
   Quinary (R = 5)

Number Systems

a. Positional Number Notation

b. Radix or Base

c. Why decimal? Number of Fingers
   Other radices in use in various civilizations:
   Duodecimal (R = 12): Finger-Joints excluding Thumb
   Quinary (R = 5): Fingers excluding Thumb

Number Systems

a. Positional Number Notation

b. Radix or Base

c. Why decimal? Number of Fingers
   Other radices in use in various civilizations:
   Duodecimal (R = 12): Finger-Joints excluding Thumb
   Quinary (R = 5): Fingers excluding Thumb
   Vigesimal (R = 20)

Number Systems

a. Positional Number Notation

b. Radix or Base

c. Why decimal? Number of Fingers
   Other radices in use in various civilizations:
   Duodecimal (R = 12): Finger-Joints excluding Thumb
   Quinary (R = 5): Fingers excluding Thumb
   Vigesimal (R = 20): Total of Fingers and Toes

Number Systems

a. Positional Number Notation

b. Radix or Base

c. Why decimal? Number of Fingers
   Other radices in use in various civilizations:
   Duodecimal (R = 12): Finger-Joints excluding Thumb
   Quinary (R = 5): Fingers excluding Thumb
   Vigesimal (R = 20): Total of Fingers and Toes

d. Other important radices:
   Sexagesimal (R = 60)

Number Systems

a. Positional Number Notation

b. Radix or Base

c. Why decimal? Number of Fingers
   Other radices in use in various civilizations:
   Duodecimal (R = 12): Finger-Joints excluding Thumb
   Quinary (R = 5): Fingers excluding Thumb
   Vigesimal (R = 20): Total of Fingers and Toes

d. Other important radices:
   Sexagesimal (R = 60)
   Binary (R = 2)

Number Systems

a. Positional Number Notation

b. Radix or Base

c. Why decimal? Number of Fingers
   Other radices in use in various civilizations:
   Duodecimal (R = 12): Finger-Joints excluding Thumb
   Quinary (R = 5): Fingers excluding Thumb
   Vigesimal (R = 20): Total of Fingers and Toes

d. Other important radices:
   Sexagesimal (R = 60)
   Binary (R = 2): physical representation of a bit

Number Systems

a. Positional Number Notation

b. Radix or Base

c. Why decimal? Number of Fingers
   Other radices in use in various civilizations:
   Duodecimal (R = 12): Finger-Joints excluding Thumb
   Quinary (R = 5): Fingers excluding Thumb
   Vigesimal (R = 20): Total of Fingers and Toes

d. Other important radices:
   Sexagesimal (R = 60)
   Binary (R = 2): physical representation of a bit
   Octal (R = 8)

Number Systems

a. Positional Number Notation

b. Radix or Base

c. Why decimal? Number of Fingers
   Other radices in use in various civilizations:
   Duodecimal (R = 12): Finger-Joints excluding Thumb
   Quinary (R = 5): Fingers excluding Thumb
   Vigesimal (R = 20): Total of Fingers and Toes

d. Other important radices:
   Sexagesimal (R = 60)
   Binary (R = 2): physical representation of a bit
   Octal (R = 8): groups of three bits

Number Systems

a. Positional Number Notation

b. Radix or Base

c. Why decimal? Number of Fingers
   Other radices in use in various civilizations:
   Duodecimal (R = 12): Finger-Joints excluding Thumb
   Quinary (R = 5): Fingers excluding Thumb
   Vigesimal (R = 20): Total of Fingers and Toes

d. Other important radices:
   Sexagesimal (R = 60)
   Binary (R = 2): physical representation of a bit
   Octal (R = 8): groups of three bits
   Hexadecimal (R = 16)

Number Systems

a. Positional Number Notation

b. Radix or Base

c. Why decimal? Number of Fingers
   Other radices in use in various civilizations:
   Duodecimal (R = 12): Finger-Joints excluding Thumb
   Quinary (R = 5): Fingers excluding Thumb
   Vigesimal (R = 20): Total of Fingers and Toes

d. Other important radices:
   Sexagesimal (R = 60)
   Binary (R = 2): physical representation of a bit
   Octal (R = 8): groups of three bits
   Hexadecimal (R = 16): groups of four bits

Number Systems

a. Positional Number Notation

b. Radix or Base

c. Why decimal?

d. Other important radices

<u>We are NOT limited to integers</u>:

e. Rational Numbers (mathematical terminology) ≡
   "Real" Numbers (computer science terminology)

   Fractional Numbers: e.g., 1,941/10,000 = 0.1941
   "Mixed" Numbers: e.g., 603,550.1941

Fractional Numbers

DECIMAL NUMBERS:

Decimal Point 603,550.1941

NUMBERS OF VARIOUS RADICES:

Radix Point $603{,}550.1941_{11}$ $603{,}550.1941_{13}$
$603{,}550.1941_{26}$ $603{,}550.1941_{18}$

SPECIFIC RADICES:

Binary Point: $1011\ 1101\ 0101.1100\ 1001\ 0001_2$

Octal Point: 7214.36502_8

Hexadecimal Point: $FEED37.09BADE_{16}$

Outline

1. Number Systems: Basic Concepts

Outline

1. Number Systems: Basic Concepts
2. Interconversion

Outline

1. Number Systems: Basic Concepts
2. Interconversion
3. Binary Numbers

Outline

1. Number Systems: Basic Concepts
2. Interconversion
3. Binary Numbers
4. Arithmetic

Outline

1. Number Systems: Basic Concepts
2. Interconversion
3. Binary Numbers
4. Arithmetic
5. Broadening Our Horizons:

Outline

1. Number Systems: Basic Concepts
2. Interconversion
3. Binary Numbers
4. Arithmetic
5. Broadening Our Horizons:
   Different Forms of Integer Number Representation

Outline

1. Number Systems: Basic Concepts
2. Interconversion
3. Binary Numbers
4. Arithmetic
5. Broadening Our Horizons:
   Different Forms of Integer Number Representation
   Performance of Arithmetic in Each Representation Scheme

Notation of Numbers in Computer Science: Integers

Multiplication Factor in non-CS Scientific Notation			Computer Science	
	Name	Sym-bol	Multipli-cation Factor	Magnitude
10^3	kilo	k	2^{10}	1,024
10^6	Mega	M	2^{20}	1,048,576
10^9	Giga	G	2^{30}	1,073,741,824
10^{12}	Tera	T	2^{40}	1,099,511,627,776
10^{15}	Peta	P	2^{50}	1,125,899,906,842,624
10^{18}	Exa	E	2^{60}	1,152,921,504,606,846,976
10^{21}	Zetta	Z	2^{70}	1,180,591,620,717,411,303,424
10^{24}	Yotta	Y	2^{80}	1,208,925,819,614,629,174,706,176

Notation of Numbers in Computer Science: Fractions

Multiplication Factor in non-CS Scientific Notation			Computer Science	
	Name	Sym-bol	Multipli-cation Factor	Magnitude
10^{-3}	milli	m	2^{-10}	1.0/1,024
10^{-6}	micro	μ	2^{-20}	1.0/1,048,576
10^{-9}	nano	n	2^{-30}	1.0/1,073,741,824
10^{-12}	pico	p	2^{-40}	1.0/1,099,511,627,776
10^{-15}	femto	f	2^{-50}	1.0/1,125,899,906,842,624
10^{-18}	atto	a	2^{-60}	1.0/1,152,921,504,606,846,976
10^{-21}	zepto	z	2^{-70}	1.0/1,180,591,620,717,411,303,424
10^{-24}	yocto	y	2^{-80}	1.0/1,208,925,819,614,629,174,706,176

Dubious Metrics

Base Quantity		*Equivalent*
10^12 dactyls	=	1 teradactyl
10^21 piccolos	=	1 gigolo
10^6 airs	=	1 millionair
10^-12 dillies	=	1 picodilly
10^12 bulls	=	1 terabull
2x10^3 mockingbirds	=	2 kilo mockingbird

Representation		Decimal Value Represented			
Hex	Binary	Unsigned (Non-Explicitly-Signed)	Ones' Complement	Two's Complement	Signed-Magnitude
0	0000	0	+0	0 ("+"0)	+0
1	0001	1	+1	+1	+1
2	0010	2	+2	+2	+2
3	0011	3	+3	+3	+3
4	0100	4	+4	+4	+4
5	0101	5	+5	+5	+5
6	0110	6	+6	+6	+6
7	0111	7	+7	+7	+7
8	1000	8	-7	-8	-0
9	1001	9	-6	-7	-1
A	1010	10	-5	-6	-2
B	1011	11	-4	-5	-3
C	1100	12	-3	-4	-4
D	1101	13	-2	-3	-5
E	1110	14	-1	-2	-6
F	1111	15	-0	-1	-7

Representation		Decimal Value Represented		
Hex	Binary	Unsigned (Non-Explicitly-Signed)	Excess-7	Excess-8
0	0000	0	-7	-8
1	0001	1	-6	-7
2	0010	2	-5	-6
3	0011	3	-4	-5
4	0100	4	-3	-4
5	0101	5	-2	-3
6	0110	6	-1	-2
7	0111	7	0 ("+"0)	-1
8	1000	8	+1	0 ("+"0)
9	1001	9	+2	+1
A	1010	10	+3	+2
B	1011	11	+4	+3
C	1100	12	+5	+4
D	1101	13	+6	+5
E	1110	14	+7	+6
F	1111	15	+8	+7

Addition of Fixed-Point Numbers (*e.g.*, Integers)

1. Terminology for three or four registers:
   a) Addition: Augend, Addend, Sum
   b) Subtraction: Minuend, Subtrahend, Difference
   c) Multiplication: Multiplicand, Multiplier, Product
   d) Division: Dividend, Divisor, Quotient, Remainder

Addition of Fixed-Point Numbers (*e.g.*, Integers)

1. Terminology for three or four registers:
   a) Addition: Augend, Addend, Sum
   b) Subtraction: Minuend, Subtrahend, Difference
   c) Multiplication: Multiplicand, Multiplier, Product
   d) Division: Dividend, Divisor, Quotient, Remainder

2. Rules for single-bit addition
   2. 0 + 0 + 0 = 0 (sum), plus 0 (carry in to next bit)
   3. 0 + 0 + 1 = 0 + 1 + 0 = 1 + 0 + 0 = 1 (sum), plus 0 (carry OUT from the current bit, which becomes the carry in to the next bit)
   4. 0 + 1 + 1 = 1 + 1 + 0 = 1 + 0 + 1 = 0 (sum), plus 1 (carry OUT from the current bit, which becomes the carry in to the next bit)
   5. 1 + 1 + 1 = 1 (sum), plus 1 (carry OUT from the current bit, which becomes the carry in to the next bit)

Addition of Fixed-Point Numbers (*e.g.*, Integers)

1. Terminology for three or four registers:
   a) Addition: Augend, Addend, Sum
   b) Subtraction: Minuend, Subtrahend, Difference
   c) Multiplication: Multiplicand, Multiplier, Product
   d) Division: Dividend, Divisor, Quotient, Remainder

2. Rules for single-bit addition
   a) 0 + 0 + 0 = 0 (sum), plus 0 (carry in to next bit)
   b) 0 + 0 + 1 = 0 + 1 + 0 = 1 + 0 + 0 = 1 (sum), plus 0 (carry OUT from the current bit, which becomes the carry in to the next bit)
   c) 0 + 1 + 1 = 1 + 1 + 0 = 1 + 0 + 1 = 0 (sum), plus 1 (carry OUT from the current bit, which becomes the carry in to the next bit)
   d) 1 + 1 + 1 = 1 (sum), plus 1 (carry OUT from the current bit, which becomes the carry in to the next bit)

3. Four *status bits* to assist in programming:
   a) C: Carry out from *M*ost *S*ignificant *B*it) of the Sum.
   b) Z: Sum has a value of Zero.
   c) N: Content of the Sum Register is a Negative number.
   d) V: Result of the arithmetic operation is an oVerflow.

Utility of Status Bits in Programming

Assembly Language (ASL): Symbolic representation of Individual Machine Instructions

ASL Mnemonic	Meaning of the Mnemonic	Function
JC	Jump on Carry	If the 'C' bit is a 1, then skip to a specified location instead of executing the next instruction
JNC	Jump on *NOT* Carry	If the 'C' bit is a 0, then skip to a specified location instead of executing the next instruction
JZ	Jump on Zero	If the 'Z' bit is a 1, then skip to a specified location instead of executing the next instruction
JNZ	Jump on *NOT* Zero	If the 'Z' bit is a 0, then skip to a specified location instead of executing the next instruction
JN	Jump on Negative	If the 'N' bit is a 1, then skip to a specified location instead of executing the next instruction
JNN	Jump on *NOT* Negative	If the 'N' bit is a 0, then skip to a specified location instead of executing the next instruction
JV	Jump on oVerflow	If the 'V' bit is a 1, then skip to a specified location instead of executing the next instruction
JNV	Jump on *NOT* oVerflow	If the 'V' bit is a 0, then skip to a specified location instead of executing the next instruction

ORGANIZATION of the SLIDES

Starting at this point and continuing for a substantial number of slides, we shall be examining several different pairs of 8-bit Augends and 8-bit Addends. We shall be adding each Augend-Addend pair in accordance with the rules of five different types of arithmetic:

- Unsigned-Number (Non-Explicitly-Signed-Number) Arithmetic
- Ones'-Complement Arithmetic
- Two's-Complement Arithmetic
- Signed-Magnitude Arithmetic
- Saturation Arithmetic

Please note that the same principles demonstrated here with 8-bit numbers are equally applicable to words of any width, from 2 bits up to any arbitrarily large number of bits, although today almost all processors have a word width that is a power of two (4-bit, 8-bit, 16-bit, 32-bit, 64-bit, or 128-bit).

FIRST Augend-Addend Pair

Augend Register	0	0	1	0	1	1	1	1	
Addend Register	0	1	0	0	1	1	1	0	
Sum Register									

1. Set up the problem.

"UNSIGNED" (Non-Explicitly-Signed) INTEGER											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits										C	
Augend Register	0	0	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register										V	

2. Carry out the operation.

"UNSIGNED" (Non-Explicitly-Signed) INTEGER											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	0		C	
Augend Register	0	0	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register	0	1	1	1	1	1	0	1		V	

3. Determine the values of the status bits.

"UNSIGNED" (Non-Explicitly-Signed) INTEGER											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	0		C	0
Augend Register	0	0	1	0	1	1	1	1		Z	0
Addend Register	0	1	0	0	1	1	1	0		N	0
Sum Register	0	1	1	1	1	1	0	1		V	0

4. Determine decimal values of register contents.

"UNSIGNED" (Non-Explicitly-Signed) INTEGER											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	0		C	0
Augend Register	0	0	1	0	1	1	1	1	(+)47	Z	0
Addend Register	0	1	0	0	1	1	1	0	(+)78	N	0
Sum Register	0	1	1	1	1	1	0	1	(+)125	V	0

1. Set up the problem.

ONES'-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits										C	
Augend Register	0	0	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register										V	

2. Carry out the operation.

ONES'-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	0		C	
Augend Register	0	0	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register	0	1	1	1	1	1	0	1		V	

3. Determine the values of the status bits.

ONES'-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	0		C	0
Augend Register	0	0	1	0	1	1	1	1		Z	0
Addend Register	0	1	0	0	1	1	1	0		N	0
Sum Register	0	1	1	1	1	1	0	1		V	0

NOTE that ***BOTH*** the 'C' bit ***AND*** the 'V' bit are 0s. Therefore, the content of the Sum Register is the *correct* sum of the Augend and the Addend.

4. Determine decimal values of register contents.

ONES'-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	0		C	0
Augend Register	0	0	1	0	1	1	1	1	+47	Z	0
Addend Register	0	1	0	0	1	1	1	0	+78	N	0
Sum Register	0	1	1	1	1	1	0	1	+125	V	0

1. Set up the problem.

TWO'S-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits										C	
Augend Register	0	0	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register										V	

2. Carry out the operation.

TWO'S-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	0		C	
Augend Register	0	0	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register	0	1	1	1	1	1	0	1		V	

3. Determine the values of the status bits.

TWO'S-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	0		C	0
Augend Register	0	0	1	0	1	1	1	1		Z	0
Addend Register	0	1	0	0	1	1	1	0		N	0
Sum Register	0	1	1	1	1	1	0	1		V	0

4. Determine decimal values of register contents.

TWO'S-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	0		C	0
Augend Register	0	0	1	0	1	1	1	1	+47	Z	0
Addend Register	0	1	0	0	1	1	1	0	+78	N	0
Sum Register	0	1	1	1	1	1	0	1	+125	V	0

1. Set up the problem.

SIGNED-MAGNITUDE											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits										C	
Augend Register	0	0	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register										V	

2. Carry out the operation.

SIGNED-MAGNITUDE											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	X	0	0	1	1	1	0	0		C	
Augend Register	0	0	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register	0	1	1	1	1	1	0	1		V	

3. Determine the values of the status bits.

SIGNED-MAGNITUDE											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	X	0	0	1	1	1	0	0		C	0
Augend Register	0	0	1	0	1	1	1	1		Z	0
Addend Register	0	1	0	0	1	1	1	0		N	0
Sum Register	0	1	1	1	1	1	0	1		V	0

4. Determine decimal values of register contents.

SIGNED-MAGNITUDE											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	X	0	0	1	1	1	0	0		C	0
Augend Register	0	0	1	0	1	1	1	1	+47	Z	0
Addend Register	0	1	0	0	1	1	1	0	+78	N	0
Sum Register	0	1	1	1	1	1	0	1	+125	V	0

1. Set up the problem.

SATURATION ARITHMETIC											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits										C	
Augend Register	0	0	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register										V	

2. Carry out the operation.

SATURATION ARITHMETIC											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	0		C	
Augend Register	0	0	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register	0	1	1	1	1	1	0	1		V	

3. Saturation Arithmetic: Does *NOT* use Status bits.

SATURATION ARITHMETIC											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	0		C	
Augend Register	0	0	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register	0	1	1	1	1	1	0	1		V	

4. Determine decimal values of register contents.

SATURATION ARITHMETIC											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	0		C	
Augend Register	0	0	1	0	1	1	1	1	(+)47	Z	
Addend Register	0	1	0	0	1	1	1	0	(+)78	N	
Sum Register	0	1	1	1	1	1	0	1	(+)125	V	

SECOND Augend-Addend Pair

Augend Register	0	1	1	0	1	1	1	1	
Addend Register	0	1	0	0	1	1	1	0	
Sum Register									

1. Set up the problem.

"UNSIGNED" (Non-Explicitly-Signed) INTEGER											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits										C	
Augend Register	0	1	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register										V	

2. Carry out the operation.

"UNSIGNED" (Non-Explicitly-Signed) INTEGER											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	
Augend Register	0	1	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register	1	0	1	1	1	1	0	1		V	

3. Determine the values of the status bits.

"UNSIGNED" (Non-Explicitly-Signed) INTEGER											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	0
Augend Register	0	1	1	0	1	1	1	1		Z	0
Addend Register	0	1	0	0	1	1	1	0		N	1
Sum Register	1	0	1	1	1	1	0	1		V	0

4. Determine decimal values of register contents.

"UNSIGNED" (Non-Explicitly-Signed) INTEGER											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	0
Augend Register	0	1	1	0	1	1	1	1	(+)111	Z	0
Addend Register	0	1	0	0	1	1	1	0	(+)78	N	1
Sum Register	1	0	1	1	1	1	0	1	(+)189	V	0

1. Set up the problem.

ONES'-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits										C	
Augend Register	0	1	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register										V	

2. Carry out the operation.

ONES'COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	
Augend Register	0	1	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register	1	0	1	1	1	1	0	1		V	

3. Determine the values of the status bits.

ONES'-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	0
Augend Register	0	1	1	0	1	1	1	1		Z	0
Addend Register	0	1	0	0	1	1	1	0		N	1
Sum Register	1	0	1	1	1	1	0	1		V	1

Determination of Overflow in Ones'-Complement:
Criteria for Setting the *V* Bit

3. Determine the values of the status bits.

ONES'-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	0
Augend Register	0	1	1	0	1	1	1	1		Z	0
Addend Register	0	1	0	0	1	1	1	0		N	1
Sum Register	1	0	1	1	1	1	0	1		V	1

Determination of Overflow in Ones'-Complement:
Criteria for Setting the *V* Bit

- Human Criterion: Augend & Addend have identical sign, Sum has opposite sign.

3. Determine the values of the status bits.

ONES'-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	0
Augend Register	0	1	1	0	1	1	1	1		Z	0
Addend Register	0	1	0	0	1	1	1	0		N	1
Sum Register	1	0	1	1	1	1	0	1		V	1

Determination of Overflow in Ones'-Complement:
Criteria for Setting the *V* Bit

- Human Criterion: Augend & Addend have identical sign, Sum has opposite sign.
- Computer Criterion: Carry-IN to *Most Significant Bit* differs from the Carry-OUT.

3. Determine the values of the status bits.

ONES'-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	0
Augend Register	0	1	1	0	1	1	1	1		Z	0
Addend Register	0	1	0	0	1	1	1	0		N	1
Sum Register	1	0	1	1	1	1	0	1		V	1

NOTE, since the 'V' bit is a 1, that the content of the Sum Register is the *NOT* the *correct* sum of the Augend and the Addend.

4. Determine decimal values of register contents.

ONES'-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	0
Augend Register	0	1	1	0	1	1	1	1	+111	Z	0
Addend Register	0	1	0	0	1	1	1	0	+78	N	1
Sum Register	1	0	1	1	1	1	0	1	-66	V	1

1. Set up the problem.

TWO'S-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits										C	
Augend Register	0	1	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register										V	

2. Carry out the operation.

TWO'S-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	
Augend Register	0	1	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register	1	0	1	1	1	1	0	1		V	

3. Determine the values of the status bits.

TWO'S-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	0
Augend Register	0	1	1	0	1	1	1	1		Z	0
Addend Register	0	1	0	0	1	1	1	0		N	1
Sum Register	1	0	1	1	1	1	0	1		V	1

Determination of Overflow in Two's-Complement:
Criteria for Setting the *V* Bit

3. Determine the values of the status bits.

TWO'S-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	0
Augend Register	0	1	1	0	1	1	1	1		Z	0
Addend Register	0	1	0	0	1	1	1	0		N	1
Sum Register	1	0	1	1	1	1	0	1		V	1

Determination of Overflow in Two's-Complement:
Criteria for Setting the *V* Bit

- Human Criterion: Augend & Addend have identical sign, Sum has opposite sign.

3. Determine the values of the status bits.

TWO'S-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	0
Augend Register	0	1	1	0	1	1	1	1		Z	0
Addend Register	0	1	0	0	1	1	1	0		N	1
Sum Register	1	0	1	1	1	1	0	1		V	1

Determination of Overflow in Ones'-Complement:
Criteria for Setting the *V* Bit

- Human Criterion: Augend & Addend have identical sign, Sum has opposite sign.
- Computer Criterion: Carry-IN to *Most Significant Bit* differs from the Carry-OUT.

3. Determine the values of the status bits.

TWO'S-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	0
Augend Register	0	1	1	0	1	1	1	1		Z	0
Addend Register	0	1	0	0	1	1	1	0		N	1
Sum Register	1	0	1	1	1	1	0	1		V	1

NOTE, since the 'V' bit is a 1, that the content of the Sum Register is the *NOT* the *correct* sum of the Augend and the Addend.

4. Determine decimal values of register contents.

TWO'S-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	0
Augend Register	0	1	1	0	1	1	1	1	+111	Z	0
Addend Register	0	1	0	0	1	1	1	0	+78	N	1
Sum Register	1	0	1	1	1	1	0	1	-67	V	1

1. Set up the problem.

SIGNED-MAGNITUDE INTEGERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits										C	
Augend Register	0	1	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register										V	

2. Carry out the operation.

SIGNED-MAGNITUDE INTEGERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	X	0	0	1	1	1	0	0		C	
Augend Register	0	1	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register	0	0	1	1	1	1	0	1		V	

3. Determine the values of the status bits.

SIGNED-MAGNITUDE INTEGERS										Status Bits	
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value		
Carry-In Bits	X	0	0	1	1	1	0	0		C	1
Augend Register	0	1	1	0	1	1	1	1		Z	0
Addend Register	0	1	0	0	1	1	1	0		N	0
Sum Register	0	0	1	1	1	1	0	1		V	1

3. Determine the values of the status bits.

SIGNED-MAGNITUDE INTEGERS										Status Bits	
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value		
Carry-In Bits	X	0	0	1	1	1	0	0		C	1
Augend Register	0	1	1	0	1	1	1	1		Z	0
Addend Register	0	1	0	0	1	1	1	0		N	0
Sum Register	0	0	1	1	1	1	0	1		V	1

Determination of Overflow in Signed-Magnitude:
Criterion for Setting the *V* Bit

3. Determine the values of the status bits.

SIGNED-MAGNITUDE INTEGERS										Status Bits	
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value		
Carry-In Bits	X	0	0	1	1	1	0	0		C	1
Augend Register	0	1	1	0	1	1	1	1		Z	0
Addend Register	0	1	0	0	1	1	1	0		N	0
Sum Register	0	0	1	1	1	1	0	1		V	1

Determination of Overflow in Signed-Magnitude:
Criterion for Setting the *V* Bit

- Carry-OUT from the Most-Significant ***MAGNITUDE*** Bit (MSB), i.e., from Bit **7** in this particular example.

4. Determine decimal values of register contents.

SIGNED-MAGNITUDE INTEGERS										Status Bits	
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value		
Carry-In Bits	X	0	0	1	1	1	0	0		C	1
Augend Register	0	1	1	0	1	1	1	1	+111	Z	0
Addend Register	0	1	0	0	1	1	1	0	+78	N	0
Sum Register	0	0	1	1	1	1	0	1	**+61**	V	1

1. Set up the problem.

SATURATION ARITHMETIC										Status Bits	
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value		
Carry-In Bits										C	
Augend Register	0	1	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register										V	

2. Carry out the operation.

SATURATION ARITHMETIC										Status Bits	
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value		
Carry-In Bits	1	0	0	1	1	1	0	0		C	
Augend Register	0	1	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register	1	0	1	1	1	1	0	1		V	

3. Saturation Arithmetic: Does *NOT* use Status bits.

SATURATION ARITHMETIC											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	
Augend Register	0	1	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register	1	0	1	1	1	1	0	1		V	

4. Determine decimal values of register contents.

SATURATION ARITHMETIC											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	
Augend Register	0	1	1	0	1	1	1	1	+111	Z	
Addend Register	0	1	0	0	1	1	1	0	+78	N	
Sum Register	1	0	1	1	1	1	0	1	+189	V	

THIRD Augend-Addend Pair

Augend Register	1	0	1	0	1	1	1	1	
Addend Register	0	1	0	0	1	1	1	0	
Sum Register									

1. Set up the problem.

UNSIGNED (Non-Explicitly-Signed) Numbers											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits										C	
Augend Register	1	0	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register										V	

2. Carry out the operation.

UNSIGNED (Non-Explicitly-Signed) Numbers											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	1		C	
Augend Register	1	0	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register	1	1	1	1	1	1	0	1		V	

3. Determine the values of the status bits.

UNSIGNED (Non-Explicitly-Signed) Numbers											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	1		C	0
Augend Register	1	0	1	0	1	1	1	1		Z	0
Addend Register	0	1	0	0	1	1	1	0		N	0
Sum Register	1	1	1	1	1	1	0	1		V	0

4. Determine decimal values of register contents.

UNSIGNED (Non-Explicitly-Signed) Numbers											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	1		C	0
Augend Register	1	0	1	0	1	1	1	1	(+)175	Z	0
Addend Register	0	1	0	0	1	1	1	0	(+)78	N	0
Sum Register	1	1	1	1	1	1	0	1	(+)253	V	0

1. Set up the problem.

ONES'-COMPLEMENT INTEGERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits										C	
Augend Register	1	0	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register										V	

2. Carry out the operation.

ONES'-COMPLEMENT INTEGERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	1		C	
Augend Register	1	0	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register	1	1	1	1	1	1	0	1		V	

3. Determine the values of the status bits.

ONES'-COMPLEMENT INTEGERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	1		C	0
Augend Register	1	0	1	0	1	1	1	1		Z	0
Addend Register	0	1	0	0	1	1	1	0		N	1
Sum Register	1	1	1	1	1	1	0	1		V	0

NOTE that ***BOTH*** the 'C' bit ***AND*** the 'V' bit are 0s. Therefore, the content of the Sum Register is the *correct* sum of the Augend and the Addend.

4. Determine decimal values of register contents.

ONES'-COMPLEMENT INTEGERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	1		C	0
Augend Register	1	0	1	0	1	1	1	1	-80	Z	0
Addend Register	0	1	0	0	1	1	1	0	+78	N	1
Sum Register	1	1	1	1	1	1	0	1	-2	V	0

1. Set up the problem.

TWO'S-COMPLEMENT INTEGERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits										C	
Augend Register	1	0	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register										V	

2. Carry out the operation.

TWO'S-COMPLEMENT INTEGERS										Status Bits	
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value		
Carry-In Bits	0	0	0	1	1	1	0	1		C	
Augend Register	1	0	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register	1	1	1	1	1	1	0	1		V	

3. Determine the values of the status bits.

TWO'S-COMPLEMENT INTEGERS										Status Bits	
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value		
Carry-In Bits	0	0	0	1	1	1	0	1		C	0
Augend Register	1	0	1	0	1	1	1	1		Z	0
Addend Register	0	1	0	0	1	1	1	0		N	1
Sum Register	1	1	1	1	1	1	0	1		V	0

4. Determine decimal values of register contents.

TWO'S-COMPLEMENT INTEGERS										Status Bits	
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value		
Carry-In Bits	0	0	0	1	1	1	0	1		C	0
Augend Register	1	0	1	0	1	1	1	1	-81	Z	0
Addend Register	0	1	0	0	1	1	1	0	+78	N	1
Sum Register	1	1	1	1	1	1	0	1	-3	V	0

1. Set up the problem.

SIGNED-MAGNITUDE INTEGERS										Status Bits	
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value		
Carry-In Bits										C	
Augend Register	1	0	1	0	1	1	1	1		Z	
Addend Register	0	1	0	0	1	1	1	0		N	
Sum Register										V	

2. Determine decimal values of register contents.

SIGNED-MAGNITUDE INTEGERS										Status Bits	
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value		
Carry-In Bits										C	
Augend Register	1	0	1	0	1	1	1	1	-47	Z	
Addend Register	0	1	0	0	1	1	1	0	+78	N	
Sum Register									+31	V	

Reason for change in the order of the steps:

The problem requires the addition of numbers of opposite sign.

ALGORITHM for ADDITION of SIGNED-MAGNITUDE NUMBERS of OPPOSITE SIGN:

1. Determine which number (Augend or Addend) has the LARGER MAGNITUDE.
2. Replicate its sign bit in the SUM REGISTER.
3. Subtract the MAGNITUDE of the SMALLER-MAGNITUDE number from the magnitude of the LARGER-MAGNITUDE number.
4. Fill in the MAGNITUDE FIELD of the SUM REGISTER appropriately.

3. Fill in the contents of the Sum Register.

SIGNED-MAGNITUDE INTEGERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits										C	
Augend Register	1	0	1	0	1	1	1	1	-47	Z	
Addend Register	0	1	0	0	1	1	1	0	+78	N	
Sum Register	0	0	0	1	1	1	1	1	+31	V	

4. Determine the values of the status bits.

SIGNED-MAGNITUDE INTEGERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	1		C	0
Augend Register	1	0	1	0	1	1	1	1	-80	Z	0
Addend Register	0	1	0	0	1	1	1	0	+78	N	1
Sum Register	1	1	1	1	1	1	0	1	-3	V	0

1. Set up the problem.

SATURATION ARITHMETIC										
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits
Carry-In Bits										C
Augend Register	1	0	1	0	1	1	1	1		Z
Addend Register	0	1	0	0	1	1	1	0		N
Sum Register										V

2. Carry out the operation.

SATURATION ARITHMETIC										
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits
Carry-In Bits	0	0	0	1	1	1	0	1		C
Augend Register	1	0	1	0	1	1	1	1		Z
Addend Register	0	1	0	0	1	1	1	0		N
Sum Register	1	1	1	1	1	1	0	1		V

3. Saturation Arithmetic: Does *NOT* use Status bits.

SATURATION ARITHMETIC										
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits
Carry-In Bits	0	0	0	1	1	1	0	1		C
Augend Register	1	0	1	0	1	1	1	1		Z
Addend Register	0	1	0	0	1	1	1	0		N
Sum Register	1	1	1	1	1	1	0	1		V

4. Determine decimal values of register contents.

SATURATION ARITHMETIC										
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits
Carry-In Bits	0	0	0	1	1	1	0	1		C
Augend Register	1	0	1	0	1	1	1	1	(+)175	Z
Addend Register	0	1	0	0	1	1	1	0	(+)78	N
Sum Register	1	1	1	1	1	1	0	1	(+)253	V

FOURTH Augend-Addend Pair

Augend Register	1	0	1	0	1	1	1	1	
Addend Register	1	1	0	0	1	1	1	0	
Sum Register									

1. Set up the problem.

UNSIGNED ("Non-Explicitly-Signed) NUMBERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits										C	
Augend Register	1	0	1	0	1	1	1	1		Z	
Addend Register	1	1	0	0	1	1	1	0		N	
Sum Register										V	

2. Carry out the operation.

UNSIGNED ("Non-Explicitly-Signed) NUMBERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	0		C	
Augend Register	1	0	1	0	1	1	1	1		Z	
Addend Register	1	1	0	0	1	1	1	0		N	
Sum Register	0	1	1	1	1	1	0	1		V	

3. Determine the values of the status bits.

UNSIGNED ("Non-Explicitly-Signed) NUMBERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	0		C	1
Augend Register	1	0	1	0	1	1	1	1		Z	0
Addend Register	1	1	0	0	1	1	1	0		N	0
Sum Register	0	1	1	1	1	1	0	1		V	1

4. Determine decimal values of register contents.

UNSIGNED ("Non-Explicitly-Signed) NUMBERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	0		C	1
Augend Register	1	0	1	0	1	1	1	1	(+)175	Z	0
Addend Register	1	1	0	0	1	1	1	0	(+)206	N	0
Sum Register	0	1	1	1	1	1	0	1	(+)125	V	1

1. Set up the problem.

ONES'-COMPLEMENT INTEGERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits										C	
Augend Register	1	0	1	0	1	1	1	1		Z	
Addend Register	1	1	0	0	1	1	1	0		N	
Sum Register										V	

2. Carry out the operation.

ONES'-COMPLEMENT INTEGERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	0		C	
Augend Register	1	0	1	0	1	1	1	1		Z	
Addend Register	1	1	0	0	1	1	1	0		N	
Sum Register	0	1	1	1	1	1	0	1		V	

3. Determine the values of the status bits.

ONES'-COMPLEMENT INTEGERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	0		C	1
Augend Register	1	0	1	0	1	1	1	1		Z	0
Addend Register	1	1	0	0	1	1	1	0		N	0
Sum Register	0	1	1	1	1	1	0	1		V	1

NOTE that ***BOTH*** the 'C' bit ***AND*** the 'V' bit are **1**s. The 1 in the 'V' bit is critical. Even though the 'C' bit value of 1 would indicate that an End-Around Carry needs to be performed, nevertheless since the 'V' bit is also a 1, we conclude that the sum can***NOT*** be "fixed" by implementing the "end-around carry".

4. Determine decimal values of register contents.

ONES'-COMPLEMENT INTEGERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	0		C	1
Augend Register	1	0	1	0	1	1	1	1	-80	Z	0
Addend Register	1	1	0	0	1	1	1	0	-49	N	0
Sum Register	0	1	1	1	1	1	0	1	+125	V	1

1. Set up the problem.

TWO'S-COMPLEMENT INTEGERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits										C	
Augend Register	1	0	1	0	1	1	1	1		Z	
Addend Register	1	1	0	0	1	1	1	0		N	
Sum Register										V	

2. Carry out the operation.

TWO'S-COMPLEMENT INTEGERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	0		C	
Augend Register	1	0	1	0	1	1	1	1		Z	
Addend Register	1	1	0	0	1	1	1	0		N	
Sum Register	0	1	1	1	1	1	0	1		V	

3. Determine the values of the status bits.

TWO'S-COMPLEMENT INTEGERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	0		C	1
Augend Register	1	0	1	0	1	1	1	1		Z	0
Addend Register	1	1	0	0	1	1	1	0		N	0
Sum Register	0	1	1	1	1	1	0	1		V	1

4. Determine decimal values of register contents.

TWO'S-COMPLEMENT INTEGERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	0		C	1
Augend Register	1	0	1	0	1	1	1	1	-81	Z	0
Addend Register	1	1	0	0	1	1	1	0	-50	N	0
Sum Register	0	1	1	1	1	1	0	1	+125	V	1

1. Set up the problem.

SIGNED-MAGNITUDE INTEGERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits										C	
Augend Register	1	0	1	0	1	1	1	1		Z	
Addend Register	1	1	0	0	1	1	1	0		N	
Sum Register										V	

2. Carry out the operation.

SIGNED-MAGNITUDE INTEGERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	X	0	0	1	1	1	0	0		C	
Augend Register	1	0	1	0	1	1	1	1		Z	
Addend Register	1	1	0	0	1	1	1	0		N	
Sum Register	1	1	1	1	1	1	0	1		V	

3. Determine the values of the status bits.

SIGNED-MAGNITUDE INTEGERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	X	0	0	1	1	1	0	0		C	0
Augend Register	1	0	1	0	1	1	1	1		Z	0
Addend Register	1	1	0	0	1	1	1	0		N	1
Sum Register	1	1	1	1	1	1	0	1		V	0

4. Determine decimal values of register contents.

SIGNED-MAGNITUDE INTEGERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	X	0	0	1	1	1	0	0		C	0
Augend Register	1	0	1	0	1	1	1	1	-47	Z	0
Addend Register	1	1	0	0	1	1	1	0	-78	N	1
Sum Register	1	1	1	1	1	1	0	1	-125	V	0

1. Set up the problem.

SATURATION ARITHMETIC											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits										C	
Augend Register	1	0	1	0	1	1	1	1		Z	
Addend Register	1	1	0	0	1	1	1	0		N	
Sum Register										V	

2. Carry out the operation.

SATURATION ARITHMETIC											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	0		C	
Augend Register	1	0	1	0	1	1	1	1		Z	
Addend Register	1	1	0	0	1	1	1	0		N	
Sum Register	0	1	1	1	1	1	0	1		V	

3. Saturation Arithmetic: Does *NOT* use Status bits.

SATURATION ARITHMETIC											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	0		C	
Augend Register	1	0	1	0	1	1	1	1		Z	
Addend Register	1	1	0	0	1	1	1	0		N	
Sum Register	1	1	1	1	1	1	1	1		V	

3. Saturation Arithmetic: Does *NOT* use Status bits.

SATURATION ARITHMETIC											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	0		C	
Augend Register	1	0	1	0	1	1	1	1		Z	
Addend Register	1	1	0	0	1	1	1	0		N	
Sum Register	1	1	1	1	1	1	1	1		V	

HOWEVER, in place of overflow, which would have occurred in ordinary "Unsigned-Number" arithmetic, the content of the Sum Register is set via hardware to the SATURATION level: the largest "Unsigned" (Non-Explicitly-Signed) number that the register can hold (all '1's).

4. Determine decimal values of register contents.

SATURATION ARITHMETIC											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	0	0	0	1	1	1	0	0		C	
Augend Register	1	0	1	0	1	1	1	1	(+)175	Z	
Addend Register	1	1	0	0	1	1	1	0	(+)206	N	
Sum Register	1	1	1	1	1	1	1	1	(+)255	V	

NOTE that according to the rules of **8-bit Saturation Arithmetic**, +255 is the ***CORRECT*** sum of +175 and +206.

SUBTRACTION in Saturation Arithmetic

1. Set up the problem.

SATURATION ARITHMETIC											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
										C	
Minuend Register	1	0	1	0	1	1	1	1		Z	
Subtrahend Register	1	1	0	0	1	1	1	0		N	
Difference Register										V	

SUBTRACTION in Saturation Arithmetic

2. Carry out the operation.

SATURATION ARITHMETIC											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
										C	
Minuend Register	1	0	1	0	1	1	1	1		Z	
Subtrahend Register	1	1	0	0	1	1	1	0		N	
Difference Register	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*		V	

SUBTRACTION in Saturation Arithmetic

3. Saturation Arithmetic: Does *NOT* use Status bits.

SATURATION ARITHMETIC											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
										C	
Minuend Register	1	0	1	0	1	1	1	1		Z	
Subtrahend Register	1	1	0	0	1	1	1	0		N	
Difference Register	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*		V	

SUBTRACTION in Saturation Arithmetic

3. Saturation Arithmetic: Does *NOT* use Status bits.

SATURATION ARITHMETIC											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
										C	
Minuend Register	1	0	1	0	1	1	1	1		Z	
Subtrahend Register	1	1	0	0	1	1	1	0		N	
Difference Register	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*		V	

In ordinary "Unsigned-Number" arithmetic, when a larger number is subtracted from another, smaller number, an *underflow* occurs, since "Unsigned Numbers" are necessarily non-negative (i.e., either zero or positive), and therefore the true difference cannot be represented. *HOWEVER,* under the rules of Saturation Arithmetic, *instead of the underflow,* the content of the Difference Register is automatically adjusted via the processor's hardware to the SATURATION level: the smallest "Unsigned Number" that the register can hold, namely, *ZERO.*

SUBTRACTION in Saturation Arithmetic

4. Determine decimal values of register contents.

SATURATION ARITHMETIC											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
										C	
Minuend Register	1	0	1	0	1	1	1	1	(+)175	Z	
Subtrahend Register	1	1	0	0	1	1	1	0	(+)206	N	
Difference Register	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	0	V	

NOTE that **by the rules of Saturation Arithmetic,** 175 - 206 = 0; **this is the correct value** for the difference.

FIFTH Augend-Addend Pair

Augend Register	1	1	1	0	1	1	1	1	
Addend Register	1	1	0	0	1	1	1	0	
Sum Register									

1. Set up the problem.

UNSIGNED ("Non-Explicitly-Signed) NUMBERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits										*C*	
Augend Register	1	1	1	0	1	1	1	1		Z	
Addend Register	1	1	0	0	1	1	1	0		N	
Sum Register										V	

2. Carry out the operation.

UNSIGNED ("Non-Explicitly-Signed) NUMBERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		*C*	
Augend Register	1	1	1	0	1	1	1	1		Z	
Addend Register	1	1	0	0	1	1	1	0		N	
Sum Register	1	0	1	1	1	1	0	1		V	

3. Determine the values of the status bits.

UNSIGNED ("Non-Explicitly-Signed) NUMBERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	1
Augend Register	1	1	1	0	1	1	1	1		Z	0
Addend Register	1	1	0	0	1	1	1	0		N	0
Sum Register	1	0	1	1	1	1	0	1		V	1

4. Determine decimal values of register contents.

UNSIGNED ("Non-Explicitly-Signed) NUMBERS											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	1
Augend Register	1	1	1	0	1	1	1	1	(+)239	Z	0
Addend Register	1	1	0	0	1	1	1	0	(+)206	N	0
Sum Register	1	0	1	1	1	1	0	1	(+)189	V	1

1. Set up the problem.

ONES'-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits										C	
Augend Register	1	1	1	0	1	1	1	1		Z	
Addend Register	1	1	0	0	1	1	1	0		N	
Sum Register										V	

2. Carry out the operation.

ONES'-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	
Augend Register	1	1	1	0	1	1	1	1		Z	
Addend Register	1	1	0	0	1	1	1	0		N	
Sum Register	1	0	1	1	1	1	0	1		V	

3. Determine the values of the status bits.

ONES'-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	1
Augend Register	1	1	1	0	1	1	1	1		Z	0
Addend Register	1	1	0	0	1	1	1	0		N	1
Sum Register	1	0	1	1	1	1	0	1		V	0

NOTE that the 'C' bit, ***BUT NOT*** the 'V' bit, is a 1. Therefore, it is necessary to implement an "*END-AROUND CARRY*" operation in the Sum Register in order to obtain the *correct* sum of the Augend and the Addend.

4. Increment the Sum Register ("End-Around Carry").

ONES'-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*1*		C	0
Prior Content of Sum Register:										Z	0
	1	0	1	1	1	1	0	1		N	1
New Sum Register	1	0	1	1	1	1	1	0		V	0

NOTE that the 'C' bit, ***BUT NOT*** the 'V' bit, is a 1. Therefore, it is necessary to implement an "*END-AROUND CARRY*" operation in the Sum Register in order to obtain the *correct* sum of the Augend and the Addend.

5. Determine decimal values of register contents.

ONES'-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*1*		*C*	0
Augend Register	1	1	1	0	1	1	1	1	-16	Z	0
Addend Register	1	1	0	0	1	1	1	0	-49	N	1
Sum Register	1	0	1	1	1	1	1	0	-65	V	0

1. Set up the problem.

TWO'S-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits										C	
Augend Register	1	1	1	0	1	1	1	1		Z	
Addend Register	1	1	0	0	1	1	1	0		N	
Sum Register										V	

2. Carry out the operation.

TWO'S-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	
Augend Register	1	1	1	0	1	1	1	1		Z	
Addend Register	1	1	0	0	1	1	1	0		N	
Sum Register	1	0	1	1	1	1	0	1		V	

3. Determine the values of the status bits.

TWO'S-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	1
Augend Register	1	1	1	0	1	1	1	1		Z	0
Addend Register	1	1	0	0	1	1	1	0		N	0
Sum Register	1	0	1	1	1	1	0	1		V	0

In TWO'S COMPLEMENT, when we are performing single-precision addition we ignore the 'C' bit. It is useful only when we perform double- or multiple-precision addition.

4. Determine decimal values of register contents.

TWO'S-COMPLEMENT											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	1
Augend Register	1	1	1	0	1	1	1	1	-17	Z	0
Addend Register	1	1	0	0	1	1	1	0	-50	N	1
Sum Register	1	0	1	1	1	1	0	1	-67	V	0

1. Set up the problem.

SIGNED-MAGNITUDE											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits										C	
Augend Register	1	1	1	0	1	1	1	1		Z	
Addend Register	1	1	0	0	1	1	1	0		N	
Sum Register										V	

2. Carry out the operation.

SIGNED-MAGNITUDE											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	X	0	0	1	1	1	0	0		C	
Augend Register	1	1	1	0	1	1	1	1		Z	
Addend Register	1	1	0	0	1	1	1	0		N	
Sum Register	1	0	1	1	1	1	0	1		V	

3. Determine the values of the status bits.

SIGNED-MAGNITUDE											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	X	0	0	1	1	1	0	0		C	1
Augend Register	1	1	1	0	1	1	1	1		Z	0
Addend Register	1	1	0	0	1	1	1	0		N	1
Sum Register	1	0	1	1	1	1	0	1		V	1

4. Determine decimal values of register contents.

SIGNED-MAGNITUDE											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	X	0	0	1	1	1	0	0		C	1
Augend Register	1	1	1	0	1	1	1	1	-111	Z	0
Addend Register	1	1	0	0	1	1	1	0	-78	N	1
Sum Register	1	0	1	1	1	1	0	1	-61	V	1

1. Set up the problem.

SATURATION ARITHMETIC											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits										C	
Augend Register	1	1	1	0	1	1	1	1		Z	
Addend Register	1	1	0	0	1	1	1	0		N	
Sum Register										V	

2. Carry out the operation.

SATURATION ARITHMETIC											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	
Augend Register	1	1	1	0	1	1	1	1		Z	
Addend Register	1	1	0	0	1	1	1	0		N	
Sum Register	1	0	1	1	1	1	0	1		V	

3. Saturation Arithmetic: Does *NOT* use Status bits.

SATURATION ARITHMETIC											
	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	
Augend Register	1	1	1	0	1	1	1	1		Z	
Addend Register	1	1	0	0	1	1	1	0		N	
Sum Register	1	1	1	1	1	1	1	1		V	

HOWEVER, in place of overflow, which would have occurred in ordinary "Unsigned-Number" arithmetic, the content of the Sum Register is set via hardware to the SATURATION level: the largest non-explicitly-signed number that the register can hold (all '1's).

4. Determine decimal values of register contents.

SATURATION ARITHMETIC	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Decimal Value	Status Bits	
Carry-In Bits	1	0	0	1	1	1	0	0		C	
Augend Register	1	1	1	0	1	1	1	1	+239	Z	
Addend Register	1	1	0	0	1	1	1	0	+206	N	
Sum Register	1	1	1	1	1	1	1	1	+255	V	

NOTE that according to the rules of **8-bit Saturation Arithmetic**, +255 is the ***CORRECT*** sum of +239 and +206.

Representation of Floating-Point Numbers

Modern physics, chemistry, biology, astronomy, meteorology, and also various disciplines of engineering require the recording of both very large numbers and very small numbers, as well as the performance of mathematical operations upon them.

For such numbers, integer or fixed-point notation is extremely inefficient, since a substantial part of such a representation would have to be taken up with large numbers of leading or trailing zeroes.

For example, the star system Alpha Centauri, the nearest star system from our sun, is *approximately* 41,153,298,540,000,000 meters away from Earth. The wavelength of Copper K-alpha radiation (a particular type of X-ray used by chemists to measure the dimensions of molecular crystals) is .0000000001541 meters.

(continued)

Representation of Floating-Point Numbers

(continued)

Scientists developed a notational system for the purpose of writing very large and very small numbers without having to use a large number of zeroes. This is called *Scientific Notation*.

Computer science makes use of several varieties of notation, all based upon scientific notation, but especially adapted to use in digital computers. These consist of several varieties of a scheme known as *Floating-Point Number Representation*.

Scientific Notation is informal, flexible, and highly adaptable.

Floating-Point Number Representation must necessarily be very carefully defined and specified in each place where it is used, so that programs making use of it will generate results that are correct.

Scientific Notation

Format: *Mantissa * Radix^{Exponent}*

Example 1: $+6.02214199 * 10^{23}$ (Avogadro's Number, in Physics/Chemistry: the number of molecules in a gram-mole of a substance; named after Amadeo Avogadro, 1776-1856)

Example 2: $+9.10939 * 10^{-31}$ Kilograms (Mass of the electron, according to Robert A. Millikan, 1868-1953, winner of the Nobel Prize in physics in 1923 for making the measurement)

Example 3: $-1.60217733 * 10^{-19}$ Coulombs (first measured accurately by Millikan)

Simple notation is used *both* for very large *and* for very small numbers.

Scientific Notation (continued)

$5.00 * 10^{0} = 0.5 * 10^{1} = 0.05 * 10^{2} = 5,000 * 10^{-3}$

All four notations are legal, but the *de facto* standard *usually* has ONE significant decimal digit (i.e., a digit other than zero) to the left of the mantissa's decimal point.

Components of a number represented in Scientific Notation:

(1) Mantissa
(2) Sign of the Mantissa
(3) Radix (almost always 10 in Scientific Notation)
(4) Sign of the Exponent
(5) Magnitude of the Exponent

Floating-Point Representation: IEEE Standard 754

FOUR components of a number in Floating-Point Representation:

(1) Sign of the Mantissa: EXPLICIT
(2) Magnitude of the Mantissa: EXPLICIT
(3) Radix: ***IMPLICIT***
(4) Exponent: EXPLICIT

TWO different levels of precision: SINGLE (32-bit) & DOUBLE (64-bit)

(continued)

Floating-Point Representation: IEEE Standard 754

Single-Precision:

	8-bit Exponent	23-bit Normalized Fractional Significand
	Excess-127	Integer bit elided

Sign Bit (Matissa)

Double-Precision: *High*-Order Word

	11-bit Exponent	Upper 20 bits of Normalized Fractional Significand
	Excess-1023	Integer bit elided

Sign Bit (Matissa)

Double-Precision: *Low*-Order Word

Lower 32 bits of 52-bit Normalized Fractional Significand
(Integer bit elided)

Floating-Point Representation: IEEE Standard 754

1. Sign bit *S*: Sign of the Mantissa = $(-1)^S$

2. Radix = 2 (NOTE, however, that non-IEEE formats for Floating-Point numbers have been implemented on various computers with radices of 2, 8, and 16.)

3. With limited exceptions, Floating Point numbers are stored as *normalized* numbers, that is, the exponent is adjusted so that the mantissa has a value greater than or equal to 1.0_{10} and less than 2.0_{10}. All normalized numbers therefore have a mantissa composed of a single integer bit of 1, followed by several fractional bits consisting of 1s and 0s as appropriate. The general form of such a mantissa is: $1.f$, where f stands for the fractional component.

(continued)

Floating-Point Representation: IEEE Standard 754

(continued)

4. Since every normalized mantissa contains an integer bit of 1, it is not necessary to waste one of the precious available bits to represent it explicitly. Instead, the integer bit of 1 is ***elided*** (that is, its presence is implied and therefore it is not explicitly represented anywhere in the memory storage location or in the floating-point register. Only the bits of the fractional portion of the mantissa are represented explicitly, and therefore these bits are called the significand and not the mantissa. of the register or memory storage location. Thus, either 24 bits or 53 bits of precision are obtained from only 23 bits or 52 bits of storage space.

(continued)

Floating-Point Representation: IEEE Standard 754

(continued)

5. Special cases: Several representations are specially defined as having values that deviate from the normal pattern. These are:

   a) Zero: Since the standard Floating-Point representation incorporates an elided integer component of 1, a value of zero would not be exactly representable; it would be necessary to approximate zero by using the value of the smallest non-zero value. This would be $1.0 * 2^{-127}$ for single-precision, or $1.0 * 2^{-1023}$ for double-precision.

(continued)

Floating-Point Representation: IEEE Standard 754

(continued)

In fact, there are two representations that are reserved for zero: +0 (sign bit = 0) and -0 (sign bit = 1). For both of these, all exponent bits and all significand bits are zeroes.

b) *Denormalized* numbers: If only normalized numbers were represented, then single-precision Floating-Point numbers would be limited to a minimum non-zero value of $\pm 1.0 * 2^{-127}$, and double-precision to $\pm 1.0 * 2^{-1023}$. The representation in the exponent field of all zeroes is also used to enable the extension of floating-point representation to much smaller non-zero values.

(continued)

Floating-Point Representation: IEEE Standard 754

(continued)

When the exponent field contains all zeroes, then the significand field represents a non-normalized fractional mantissa. Thus, the minimum non-zero values that can be represented are $\pm 2^{-23} * 2^{-126} = \pm 2^{-149}$ (single-precision) and $\pm 2^{-1074}$ (double-precision).

6. *±Infinity*: denoted by BOTH all 1s in the exponent field AND all 0s in the significand field.

(continued)

Floating-Point Representation: IEEE Standard 754

(continued)

7. *"Not a Number (NaN)"* : In certain cases, the result of a specific operation is defined in mathematics as being indeterminate: for example, an attempt to divide ±zero by ±zero, or ±infinity by ±infinity, or an attempt to subtract infinity from infinity, or an attempt to multiply ±infinity by ±zero. These cases are all represented by an exponent field consisting of all 1s, together with a significand field that contains *at least* one 1. These representations are collectively referred to as "Not a Number (NaN)".

END