CHAPTER 4 : LOGIC DESIGN / CIRCUITS

UP TIL NOW WE HAVE CONSIDERED THE COMPUTING AGENT WHICH EXECUTES ALGORITHMS TO BE AN ABSTRACT ENTITY.

NOW WE WILL BE CONCERNED WITH TECHNIQUES FOR DESIGNING REAL COMPUTING MACHINES.

WE WILL DISCUSS FUNDAMENTAL BUILDING BLOCKS CALLED <u>LOGIC GATES</u> WHICH CAN BE COMBINED TO FORM <u>CIRCUITS</u> THAT PERFORM BASIC OPERATIONS SUCH AS ADDITION, MULTIPLICATION, COMPARISON, AND FETCHING INSTRUCTIONS.

ULTIMATELY, REAL COMPUTING MACHINES STORE AND PROCESS ALL INFORMATION IN THE FORM OF <u>BITS</u> (<u>BINARY DIGITS</u>) i.e. 0's AND 1's. THEREFORE WE BEGIN BY REVIEWING THE BINARY NUMBERING SYSTEM.

WHAT DOES 12,526 MEAN?

$$12526 = 1\cdot10^4 + 2\cdot10^3 + 5\cdot10^2 + 2\cdot10^1 + 6\cdot10^0$$

THIS IS THE BASE 10 NUMBER SYSTEM WITH WHICH WE ARE FAMILIAR.

GIVEN AN INTEGER $b > 1$, THE <u>BASE b POSITIONAL NUMBERING SYSTEM</u> IS OBTAINED BY ASSIGNING b SYMBOLS (CALLED <u>DIGITS</u>) TO THE INTEGERS $0, 1, 2, \ldots, (b-1)$. A STRING $a_{n-1}a_{n-2}\cdots a_1 a_0$ OF n DIGITS IS THEN INTERPRETED AS THE INTEGER:

$$[a_{n-1}\cdots a_0]_b = a_{n-1}b^{n-1} + a_{n-2}b^{n-2} + \cdots + a_1 b^1 + a_0 b^0$$

MORE GENERALLY A FRACTION IS REPRESENTED

$$[a_{n-1}\cdots a_0 . a_{-1}\cdots a_{-k}]_b = a_{n-1}b^{n-1} + \cdots + a_0 b^0 + a_{-1}b^{-1} + \cdots + a_{-k}b^{-k}$$

EACH a_i IS OF COURSE A DIGIT $\{0, 1, \ldots, b-1\}$.

IN COMPUTER SCIENCE WE ARE MOST CONCERNED WITH THE BASES

$b = 2$: BINARY $\{0, 1\}$

$b = 8$: OCTAL $\{0, 1, 2, 3, 4, 5, 6, 7\}$

$b = 16$: HEXIDECIMAL
$\{0, 1, 2, \ldots, 9, A, B, C, D, E, F\}$

$b = 10$: DECIMAL $\{0, 1, \ldots, 9\}$

EX. COUNT IN BASE 2

0 1 10 11 100 101 110 111 1000

NOTE: 1 FOLLOWED BY n ZEROS IS 2^n.

EX. $[1011010111001]_2 = 5,817$

EX. $[237]_8 = 159$

EX. $[A17D]_{16} = 41,341$

EX. $[357]_{10} = [\ ?\ ]_2$

K	0	1	2	3	4	5	6	7	8	9
2^K	1	2	4	8	16	32	64	128	256	512

$$\begin{aligned} 357 &= 256 + 101 \\ &= 256 + 64 + 37 \\ &= 256 + 64 + 32 + 5 \\ &= 256 + 64 + 32 + 4 + 1 \\ &= 2^8 + 2^6 + 2^5 + 2^2 + 2^0 \end{aligned}$$

$\therefore\ [357]_{10} = [101100101]_2$

EX. $[\underline{101}\ \underline{100}\ \underline{101}]_2 = [585]_8$

EX. $[\underline{0001}\ \underline{0110}\ \underline{0101}]_2 = [165]_{16}$

COMPUTING MACHINES REPRESENT POSITIVE INTEGERS AT THE MOST ELEMENTARY LEVEL AS 0's AND 1's, i.e. IN BINARY.

IN ANY SUCH MACHINE THERE IS A MAXIMUM NUMBER OF BINARY DIGITS (BITS) WHICH CAN BE USED TO STORE AN INTEGER, TYPICALLY 16, 24, 32, OR 64 BITS.

EX.

IF 16 BITS ARE AVAILABLE FOR INTEGER REPRESENTATION, THEN THE LARGEST SUCH INTEGER IS

$$\begin{aligned} 1111\ 1111\ 1111\ 1111 &= 1000000000000000 - 1 \\ &= 2^{16} - 1 \\ &= 65,535 \end{aligned}$$

IF 32 BITS ARE AVAILABLE THEN THE LARGEST REPRESENTABLE INTEGER IS

$$2^{32} - 1 = 4,294,967,295.$$

ANY OPERATION WHICH RESULTS IN AN INTEGER LAGER THAN THE MACHINE MAXIMUM CAUSES AN ERROR CONDITION CALLED ARITHMETIC OVERFLOW

THUS FAR WE'VE TALKED ABOUT THE INTERNAL REPRESENTATION OF UNSIGNED (I.E. POSITIVE) INTEGERS. WHAT ABOUT SIGNED INTEGERS?

ONE APPROACH IS TO RESERVE THE LEADING BIT FOR THE SIGN: 1 FOR −, AND 0 FOR +. THIS IS CALLED THE SIGN/MAGNITUDE REPRESENTATION.

EX. IF 8 BITS ARE AVAILABLE FOR SIGNED INTEGER REPRESENTATION:

−27 REPRESENTED AS 1 0011011

+27 " " 0 0011011

SIGN MAGNITUDE

HOW DOES THE COMPUTER KNOW THAT THE BIT STRING 10011011 REPRESENTS THE SIGNED INTEGER −27, AND NOT THE UNSIGNED INTEGER 155?

IT DOESN'T. ANY BIT STRING IS LIKE ANY OTHER TO THE COMPUTER: MEANINGLESS. THE DIFFERENCE IS IN HOW A BIT STRING IS PROCESSED, I.E. WHAT ALGORITHM IS APPLIED.

GIVEN 8 BITS WE CAN REPRESENT SIGNED INTEGERS IN THE RANGE

$$-(2^7-1) \quad \text{TO} \quad +(2^7-1)$$

i.e. $\quad -127 \quad \text{TO} \quad 127$

(NOTE 0 HAS TWO REPRESENTATIONS: +0, −0, WHICH CAN CAUSE PROBLEMS.)

FLOATING POINT NUMBERS

WE CAN REPRESENT SIGNED AND UNSIGNED INTEGERS, BUT WHAT ABOUT FRACTIONS.

e.g. 12.75 OR −0.2109375

WE FIRST CONVERT TO SCIENTIFIC NOTATION

$$\pm M \cdot 2^{\pm E}$$

WHERE ±M AND ±E ARE SIGNED INTEGERS CALLED THE MANTISSA AND THE EXPONENT RESPECTIVELY. (BOTH BINARY.)

ASSUME WE HAVE 16 BITS AVAILABLE: 10 BITS FOR THE MANTISSA, 6 BITS FOR THE EXPONENT.

EX $12.75 = 8 + 4 + \frac{1}{2} + \frac{1}{4}$

$= [1100.11]_2$

$= [.110011]_2 \cdot 2^4$

$= [.110011]_2 \cdot 2^{[100]_2}$

THUS 12,75 IS REPRESENTED AS

0	110011000	0	00100	
+ .	MAGNITUDE	+	MAGNITUDE	FLOATING POINT REPRESENTATION
	MANTISSA		EXPONENT	

TO FIND THE MANTISSA, NORMALIZE THE NUMBER SO THAT THE FIRST SIGNIFICANT BIT IS IMMEDIATELY TO THE RIGHT OF THE BINARY POINT.

EX $-.2109375 = -\left(\frac{1}{8} + \frac{1}{16} + \frac{1}{64} + \frac{1}{128}\right)$

$= -[.0011011]_2 = -[.11011]_2 \cdot 2^{-2}$

$= -[.11011]_2 \cdot 2^{-[10]_2}$

$\therefore$

1	11011 0000	1	00010	
−	MAGNITUDE	−	MAGNITUDE	
	MANTISSA		EXPONENT.	

HOW DO WE REPRESENT TEXT INFORMATION?

EACH CHARACTER OR SYMBOL IS ASSIGNED AN INTEGER CALLED ITS CHARACTER CODE, WHICH IS THEN STORED AND PROCESSED AS A BIT STRING.

THERE ARE SEVERAL STANDARD WAYS OF ASSIGNING CHARACTER CODES. BY FAR THE MOST COMMON IS ASCII (AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE.)

ASCII USES 8 BITS AND HENCE IS ABLE TO ENCODE $2^8 = 256$ CHARACTERS WITH THE CODES 0 - 255.

THE CODES 32 - 126 ARE ASSIGNED TO PRINTABLE CHARACTERS, WHILE THE REMAINDER ARE EITHER UNASSIGNED, OR ARE ASSIGNED TO NON-PRINTABLE CHARACTERS (NEW LINE, NEW PAGE, BELL, ETC..)

SEE FIGURE 4.3 P.128 FOR A TABLE OF ASCII CODES.