Unsigned number representations

- Digital circuits and systems are a means to perform computation and logical operations via machines.
- We've considered logic equations, but we also need to understand how to represent values (numbers). Need to consider how to represent numbers so that we can build circuits to do things like addition, multiplication, etc.
- Let's first consider how to represent unsigned integers.

Values, representations, bases and digits

- Numbers have a *value*. Values have *representations* in different *bases* using *digits*.
- In base-r, we have digits from $0, 1, \cdots, r-1$.
- Popular bases and digits:

Base	**Digits**	
Base-10	$0, 1, \cdots, 9$	$\leftarrow$ *decimal*
Base-2	$0, 1$	$\leftarrow$ *binary*
Base-8	$0, 1, \cdots, 7$	$\leftarrow$ *octal*
Base-16	$0, 1, \cdots, 9, A, B, C, D, E, F$	$\leftarrow$ *hexadecimal*

- Hexadecimal uses $A, \cdots, F$ for digits 10 through 15.
- In binary, the digits are often called *bits*.

Positional number representation

- Consider any unsigned integer with value V...
- In base-r (r sometimes called the radix) using n digits (sometimes called coefficients), V is represented by $D = (d_{n-1}d_{n-2}\cdots d_1 d_0)_r$.
- Digit d_0 is the least significant digit and d_{n-1} is the most significant digit.
- This is called positional number representation.
- Note that the value of a unsigned integer V "looks the same" as its representation in base-10.

Positional number representation

- Given a representation D of some value V in base r using n digits, we can compute the value V using

$$V = d_{n-1} \times r^{n-1} + d_{n-2} \times r^{n-2} + \cdots d_1 \times r^1 + d_0 \times r^0$$

- Examples...
  - A number represented in base-10 using 3 digits:
$$\begin{aligned}(219)_{10} &= 2 \times 10^2 + 1 \times 10^1 + 9 \times 10^0 \\ &= 200 + 10 + 9 \\ &= 219\end{aligned}$$
  - A number represented in base-2 using 8 bits:
$$\begin{aligned}(11011011)_2 &= 1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 \\ &\quad +1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\ &= 128 + 64 + 16 + 8 + 2 + 1 \\ &= 219.\end{aligned}$$

Positional number representation

- A value's representation in base r can be computed using recursive division by r:

$$V = d_{n-1} \times r^{n-1} + d_{n-2} \times r^{n-2} + \cdots d_1 \times r^1 + d_0 \times r^0$$

- Division by r gives

$$\frac{V}{r} = \underbrace{d_{n-1} \times r^{n-2} + d_{n-2} \times r^{n-3} + \cdots d_1 \times r^0}_{quotient} + \underbrace{\frac{d_0}{r}}_{remainder}$$

  - One of the digits "pops out" as the remainder of the division.
  - Repeat the divison using the quotient to get the next digit.
  - Stop once the quotient is 0... (All further "leading digits" will be zero).

Positional number representation

- Example... Represent the value 53 in base-2.

		Quotient	**Remainder**	
$\frac{53}{2}$	$=$	26	1	$\rightarrow d_0 = 1$
$\frac{26}{2}$	$=$	13	0	$\rightarrow d_1 = 0$
$\frac{13}{2}$	$=$	6	1	$\rightarrow d_2 = 1$
$\frac{6}{2}$	$=$	3	0	$\rightarrow d_3 = 0$
$\frac{3}{2}$	$=$	1	1	$\rightarrow d_4 = 1$
$\frac{1}{2}$	$=$	0	1	$\rightarrow d_5 = 1$

  - Stop here since further division would simply indicate that $d_6 = 0$, $d_7 = 0$, etc.
  - Therefore 53 is represented in base-2 as 110101_2.

Positional number representation

- Worthwhile to consider the representations in base-2 of unsigned values as we count (we can relate this to the rows of a truth table).

Value/Decimal Representation	**Binary Representation**
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
...	...

- We can also figure out the *range* of values we can represent in base-2 using n bits; We can represent the values $0 \cdots 2^n - 1$ in n bits. For example, with 4 bits we can represent the values $0 \cdots 15$. With 8 bits we can represent the values $0 \cdots 255$.

Conversion between bases

- Conversion from base-a to base-b is easy:
  - Find the value of the base-a representation (the result is also the base-10 representation);
  - Use successive division to convert the value to base-b.
- Example... Convert $(110101)_2$ to base-6.
  - **Step 1:** $110101_2 = 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^2 + 1 \times 2^0 = 32 + 16 + 4 + 1 = 53 = (53)_{10}$.
  - **Step 2:** Successive division...

		Quotient	**Remainder**	
$\frac{53}{6}$	$=$	8	5	$\rightarrow d_0 = 5$
$\frac{8}{6}$	$=$	1	2	$\rightarrow d_1 = 2$
$\frac{1}{6}$	$=$	0	1	$\rightarrow d_2 = 1$

    and $53 = (125)_6$.
  - **Step 3:** Check...
    $125_6 = 1 \times 6^2 + 2 \times 6^1 + 5 \times 6^0 = 36 + 12 + 5 = 53$.

Fast conversions between bases which are powers of 2

- Conversion to/from base-2 to base-8 and base-16 is easy since they are all powers of two.
  - Binary to/from octal... Convert groups of 3-bits.
  - Binary to/from hexadecimal... Convert groups of 4-bits.
- Add leading zeros if required.
- Example...
  - $1100111100101_2 = \underbrace{0001}_{4bits;leadingzeros} \underbrace{1001}_{4bits}\underbrace{1110}_{4bits}\underbrace{0101}_{4bits} = 19E5_{16}$.
  - $1100111100101_2 = \underbrace{001}_{3bitsleadingzeros} \underbrace{100}_{3bits}\underbrace{111}_{3bits}\underbrace{100}_{3bits}\underbrace{101}_{3bits} = 14745_8$.

Unsigned addition

- Addition of unsigned numbers in any base is done just like we would in base-10 — We add digits (for base-2 we add bits) to get sums and to generate carries.
- Since this course is on digital circuits, we care mostly about numbers represented in binary.

Unsigned addition

- Example...

$$
\begin{array}{cccccccccc}
 & \curvearrowleft^{0} & \curvearrowleft^{0} & \curvearrowleft^{1} & \curvearrowleft^{0} & \curvearrowleft^{0} & \curvearrowleft^{0} & \curvearrowleft^{0} & \curvearrowleft^{0} & \\
 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 170 \\
+ & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & +48 \\
\hline
0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 218
\end{array}
$$

- Example...

$$
\begin{array}{cccccccccc}
 & \curvearrowleft^{1} & \curvearrowleft^{0} & \curvearrowleft^{1} & \curvearrowleft^{0} & \curvearrowleft^{0} & \curvearrowleft^{0} & \curvearrowleft^{0} & \curvearrowleft^{0} & \\
 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 170 \\
+ & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & +176 \\
\hline
1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 346
\end{array}
$$

- **NOTE** that we can get a non-zero carry out which is non-zero from the addition of the most significant bits. If we are limited to n digits, then a carry out of 1 signifies *unsigned overflow*. The result of the addition is too large to represent in the available number of bits.

Unsigned subtraction

- Subtraction of unsigned numbers in any base is done just like we would in base-10 — We subtract digits (in base-2 bits) to get differences and use borrows as required.
- Since this course is on digital circuits, we care mostly about numbers represented in binary.

Unsigned subtraction

- Example...

$$
\begin{array}{ccccccccccr}
 & \curvearrowright^{0} & \curvearrowright^{2} & \curvearrowright^{2} & \curvearrowright^{2} & \curvearrowright^{0} & \curvearrowright^{0} & \curvearrowright^{0} & \curvearrowright^{0} & & \\
 & \cancel{1}0 & \cancel{0}\,\cancel{2}\,1 & \cancel{1}\,0 & 0 & 1 & 0 & 1 & 0 & & 170 \\
- & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & & -48 \\
\hline
 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & & 122
\end{array}
$$

- Example...

$$
\begin{array}{ccccccccccr}
 & \curvearrowright^{2} & \curvearrowright^{2} & \curvearrowright^{2} & \curvearrowright^{2} & \curvearrowright^{0} & \curvearrowright^{0} & \curvearrowright^{0} & \curvearrowright^{0} & & \\
 & \cancel{1}\,\cancel{0}\,2 & \cancel{0}\,\cancel{2}\,1 & \cancel{1}0 & 0 & 1 & 0 & 1 & 0 & & 170 \\
- & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & & -176 \\
\hline
 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & & -6
\end{array}
$$

- **NOTE** that in the second case, we get garbage. For unsigned arithmetic, we cannot subtract a larger number from a smaller number otherwise we get *unsigned underflow*. This is indicated by the need to generate a borrow at the most-significant digit.