EE 2720

Handout #1.

Binary to octal conversion

- Separate the bits of the binary number into groups of three bits and replace each group with the corresponding octal digit. Put zeroes at the left of the integer part of the binary number and at the right of the fractional part of the binary number to make the length of both integer and fractional part multiple of 3.

Example: Convert in octal the following binary number 100011001110_2

Answer: The number is

$$\underbrace{100}\;\underbrace{011}\;\underbrace{001}\;\underbrace{110}_2 = 4316_8.$$

Example: Convert in octal the following binary number

11101101110101001_2

Answer: The length of the number is 17 bits which is not a multiple of 3 so we put a zero at the left to make the length 18 bits; (18 is multiple of 3). The number now becomes

$$\underbrace{011}\ \underbrace{101}\ \underbrace{101}\ \underbrace{110}\ \underbrace{101}\ \underbrace{001}_2 = 355651_8$$

Example: Convert in octal the following binary number 10.1011001011_2

Answer: The length of the integer part is 2 bits which is not a multiple of 3 so we put a zero at the left to make its length 3 bits. The length of the fractional part is 10 bits (not multiple of 3) so we put two zeroes at the right to make its length 12 bits; (12 is multiple of 3).

The number now becomes

$$\underbrace{010}.\ \underbrace{101}\ \underbrace{100}\ \underbrace{101}\ {\underbrace{100}}_2 = 2.5454_8$$

Octal to binary conversion

- Replace each octal digit with the corresponding 3-bit string.

Example: Convert into binary the following octal number 1357_8

Answer: $1357_8 = 001\ 011\ 101\ 111_2$

Example: Convert into binary the following octal number 2046.17_8

Answer: $2046.17_8 =$

$= 010\ 000\ 100\ 110.001\ 111_2$

Binary to hexadecimal conversion

- Separate the bits of the binary number into groups of four bits and replace each group with the corresponding hexadecimal digit.

Put zeroes at the left of the integer part of the binary number and at the right of the fractional part of the binary number to make the length of both integer and fractional part multiple of 4.

Example: Convert in hexadecimal the following binary number
100011001110_2

Answer: $100011001110_2 = \underbrace{1000}\ \underbrace{1100}\ \underbrace{1110}_2$
$= 8CE_{16}$.

Example: Convert in hexadecimal the following binary number
11101101110101001_2

Answer: The length of the number is 17 bits which is not a multiple of 4 so we put three zeroes at the left to make the length 20 bits (20 is multiple of 4). The number now becomes
$\underbrace{0001}\ \underbrace{1101}\ \underbrace{1011}\ \underbrace{1010}\ \underbrace{1001}_2 = 1DBA9_{16}$

<u>Example</u>: Convert in hexadecimal the following binary number
10.1011001011_2

<u>Answer</u>: The length of the integer part is 2 bits which is not multiple of 4 so we put two zeroes at the left to make its length 4 bits (4 is multiple of 4). The length of the fractional part is 10 bits (not multiple of 4) so we put two zeroes at the right to make its length 12 bits; (12 is multiple of 4).
The number now becomes
$\underbrace{0010}.\underbrace{1011}\ \underbrace{0010}\ \underbrace{1100}_2 = 2.B2C_{16}$

Hexadecimal to binary conversion

- Replace each hexadecimal digit with the corresponding 4-bit string.

Example: Convert into binary the following hexadecimal number $BEAD_{16}$

Answer:

$BEAD_{16} = 1011\ 1110\ 1010\ 1101_2$

Example: Convert into binary the following hexadecimal number $9F.46C_{16}$

Answer: $9F.46C_{16} =$

$= 1001\ 1111.0100\ 0110\ 1100_2$

Some Notations

- Decimal number

$X = 1753_{10} = 1753d = 1753$

- Binary number

$X = 100011_2 = 100011b$

- Octal number

$X = 45_8 = 45q = 45o$

- Hexadecimal number

$X = 1234A678_{16} = 1234A678h =$
$= 0x1234A678$

Some Definitions

- A 4-bit hexadecimal digit is also called nibble.
- A group of eight (8) bits is called a byte
- A group of 32 bits is called word.

Radix r to Decimal Conversion

$X = (x_{n-1}x_{n-2}\cdots x_1 x_0 . x_{-1}x_{-2}\cdots x_{-k})_r$

$\uparrow$ radix point

$$X_{value} = \sum_{i=-k}^{n-1} x_i \times r^i .$$

We will use the above formula for radix r to decimal conversions.

Some examples follow

Examples:

$101_2 = 1\times2^2 + 0\times2^1 + 1\times2^0 = 5_{10}$

$432_5 = 4\times5^2 + 3\times5^1 + 2\times5^0 = 117_{10}$

$75_8 = 7\times8^1 + 5\times8^0 = 61_{10}$

$1A_{16} = 1\times16^1 + 10\times16^0 = 26_{10}$

Decimal to Radix r Conversion

• Conversion of the integer decimal part to radix r number

• Divide the integer part of the decimal number by r. Keep the remainder and the quotient. This first remainder is the least significant digit (LSD) of the radix r number. Divide the obtained quotient by r. Keep the remainder and the quotient. Keep doing this until a zero quotient is reached. The obtained remainders are the digits of the radix r number.

The last remainder is the most significant digit (MSD) of the radix r number.

- <u>Conversion of the fractional decimal part to radix r number</u>
- Multiply the fractional part of the decimal number by r. This might give you a mixed number; (consisting of integer and fractional part). The first obtained integer part is the MSD of the [illegible] fractional part of the radix r number. Then multiply the obtained fractional part by r. The obtained integer part is the second fractional digit of the radix r number. Keep doing this until you get a zero fractional part. The integer parts obtained by this procedure are the fractional digits of the radix r number. The last one is the LSD.

This algorithm is not guaranteed to terminate since a finite fraction in one number system may correspond to an infinite one in another number system.

Example: Convert 26_{10} into binary

Answer:

	Quotient	Remainder	
26/2	13	0	LSB
13/2	6	1	
6/2	3	0	
3/2	1	1	
1/2	0	1	MSB

So $26_{10} = 11010_2$

Example: Convert 0.75_{10} into binary

Answer:

	Fractional part	Integer part	
$0.75 \times 2 = 1.5$	0.5	1	MSB
$0.5 \times 2 = 1.0$	0.0	1	LSB

So $0.75_{10} = 0.11_2$

__Example:__ Convert 46.375_{10} into binary.

__Answer:__

- Integer part or 46

	Quotient	Remainder	
46/2	23	0	LSB
23/2	11	1	
11/2	5	1	
5/2	2	1	
2/2	1	0	
1/2	0	1	MSB

- Fractional part or 0.375

	Fractional part	Integer part	
$0.375 \times 2 = 0.75$	0.75	0	MSB
$0.75 \times 2 = 1.5$	0.5	1	
$0.5 \times 2 = 1.0$	0.0	1	LSB

So $46.375_{10} = 101110.011_2$

__Example:__ Convert 465.5625_{10} into octal

__Answer:__ $\longrightarrow$ On next page

- Integer part or 465

	Quotient	Remainder
465/8	58	1 LSD
58/8	7	2
7/8	0	7 MSD

- Fractional part or 0.5625

	Fractional part	Integer part
0.5625 × 8 = 4.5	0.5	4 MSD
0.5 × 8 = 4.0	0.0	4 LSD

So $465.5625_{10} = 721.44_8$

* Double check if you want:

$721.44_8 = 7 \times 8^2 + 2 \times 8^1 + 1 \times 8^0 + 4 \times 8^{-1} + 4 \times 8^{-2}$
$= 448 + 16 + 1 + \frac{4}{8} + \frac{4}{64} = 465 + \frac{32}{64} +$
$\frac{4}{64} = 465 + \frac{36}{64} = 465 + 0.5625 =$
465.5625_{10}

Example: Convert 0.7_{10} into binary.

Answer: I leave this as a home-work problem. As you will see you will get a binary fraction with infinite number of bits. The process does not terminate.

Note: To convert between radix r_1 and r_2 first convert the radix r_1 number into decimal and then convert the obtained decimal number into radix r_2 number; (this is the easy way to do it).

Note: Some more conversion examples from decimal to other radices are given in the text on page 30. Please read them.

Addition of binary numbers

- Addition of binary numbers is similar to addition of decimal numbers; (you know decimal addition). It is only that the addition table is different.
- To add two binary numbers X and Y we add together their least significant

bits (LSBs) with an initial carry (C_{in}) of 0, producing carry-out (C_{out}) and sum (S) bits according to the addition table which will be shown later. We continue processing bits from right to left. For each column position we add the two bits from the two numbers together with a carry producing a sum and a carry-out which goes to the position at the left.

$$X = \overset{\substack{\text{MSB}\\ \swarrow}}{x_{n-1}}\, x_{n-2} \cdots x_1 \overset{\substack{\text{LSB}\\ \swarrow}}{x_0}{}_2$$

$$Y = \underset{\substack{\uparrow\\ \text{MSB}}}{y_{n-1}}\, y_{n-2} \cdots y_1 \underset{\substack{\uparrow\\ \text{LSB}}}{y_0}{}_2$$

The addition table for binary numbers is shown on the next page.

Addition table for binary numbers

x_i	y_i	carry in	carry out	sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Example: Add $X = 190_{10}$ with $Y = 141_{10}$.
Show your work in binary

Answer: I will first show work in decimal

```
   100     carry
   190
 + 141
   331     sum
```

Now I will show it in binary

$X = 190_{10} = 10111110_2$

$Y = 141_{10} = 10001101_2$

$$
\begin{array}{cccccccccl}
1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & \text{carry} \\
\hline
 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & \\
+ & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & \\
\hline
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & \text{sum}
\end{array}
$$

Here we have an overal carry-out of the most significant location with value 1. The result X+Y is 9-bit long.

Example: Add $X = 173_{10}$ with $Y = 44_{10}$. Show your work in binary

Answer: In decimal it looks like

$$
\begin{array}{rl}
100 & \text{carry} \\
\hline
173 & \\
+\ 44 & \\
\hline
217 & \text{sum}
\end{array}
$$

Now I will show it in binary.

$X = 173_{10} = 10101101_2$
$Y = 44_{10} = 00101100_2$

$$\begin{array}{rl} 0\;\;0\,1\,0\,1\,1\,0\,0\,0 & \text{carry} \\ \hline 1\,0\,1\,0\,1\,1\,0\,1 & \\ +\;\;0\,0\,1\,0\,1\,1\,0\,0 & \\ \hline 1\,1\,0\,1\,1\,0\,0\,1 & \text{sum} \end{array}$$

No overal carry out; (overal carry out is zero). Result $X+Y$ fits in 8 bits

<u>Example</u>: Add $X = 127_{10}$ with $Y = 63_{10}$
Show your work in <u>binary</u>
<u>Answer</u>: In decimal it looks like

$$\begin{array}{rl} 0\,1\,0 & \text{carry} \\ \hline 1\,2\,7 & \\ +\;\;6\,3 & \\ \hline 1\,9\,0 & \text{sum} \end{array}$$

Now I will show it in binary.

$X = 127_{10} = 01111111_2$
$Y = 63_{10} = 00111111_2$

$$\begin{array}{rl}
0\;\underline{1\,1\,1\,1\,1\,1\,1\,0} & \text{carry} \\
0\,1\,1\,1\,1\,1\,1\,1 & \\
+\;\underline{0\,0\,1\,1\,1\,1\,1\,1} & \\
\rightarrow 1\,0\,1\,1\,1\,1\,1\,0 & \text{sum}
\end{array}$$

No overal carry out; (overal carry out is zero). Result X+Y fits in 8 bits.

<u>Example</u>: Add $X = 170_{10}$ with $Y = 85_{10}$. Show all your work in <u>binary</u>.

<u>Answer</u>: In decimal it looks like

$$\begin{array}{rl}
\underline{1\,0\,0} & \text{carry} \\
170 & \\
+\;\underline{85} & \\
255 & \text{sum}
\end{array}$$

Now I will show it in binary.

$X = 170_{10} = 10101010_2$

$Y = 85_{10} = 01010101_2$

$$\begin{array}{rl}
0\;\underline{0\,0\,0\,0\,0\,0\,0\,0} & \text{carry} \\
1\,0\,1\,0\,1\,0\,1\,0 & \\
+\;\underline{0\,1\,0\,1\,0\,1\,0\,1} & \\
\rightarrow 1\,1\,1\,1\,1\,1\,1\,1 & \text{sum}
\end{array}$$

No overal carry out; (overal carry out = 0). Result X+Y fits in 8 bits.

Unsigned binary Fixed Point (FXP) Systems

So far we have only considered unsigned binary numbers. In an unsigned integer binary system an n-bit ~~unsigned number~~ integer unsigned number is of the form $X = (x_{n-1} x_{n-2} \cdots x_1 x_0)_2$. Here, all the bits of X represent magnitude; (no sign bit is included). The bit x_{n-1} is the most significant bit (MSB) while x_0 is the least significant bit (LSB).

- Dynamic Range (DR)

By Dynamic Range (DR) we mean the range of the numbers that can be represented by a system. The Dynamic Range (DR) of an n-bit integer binary unsigned system is $DR = [0 \quad 2^n - 1]$. In other words we can represent all numbers from 0 to 2^n-1 including 0 and including 2^n-1. The representation of 0 is $0 = \underbrace{00\ldots000}_{n \text{ zeroes}}$. The representation of 2^n-1 is $2^n-1 = \underbrace{11\ldots111}_{n \text{ ones}}$.

There are 2^n numbers that can be represented. These are $0, 1, 2, 3, \ldots, 2^n-1$.

- Overflow

In an unsigned system overflow is defined to be the situation where the result of the addition of two numbers is larger than the largest value of the Dynamic Range (DR). of the system.

• Overflow Detection in integer binary unsigned systems

Consider two n-bit integer binary unsigned numbers X and Y where

$$X = \underset{\substack{\uparrow \\ \text{MSB}}}{x_{n-1}} x_{n-2} \cdots x_1 \underset{\substack{\uparrow \\ \text{LSB}}}{x_0}$$

and

$$Y = \underset{\substack{\uparrow \\ \text{MSB}}}{y_{n-1}} y_{n-2} \cdots y_1 \underset{\substack{\uparrow \\ \text{LSB}}}{y_0}$$

Consider computing X+Y ; (by now you know how to add binary numbers). The addition X+Y is shown below

$$\begin{array}{rccccc} X+Y = & & x_{n-1} & x_{n-2} \cdots & x_1 & x_0 \\ + & & y_{n-1} & y_{n-2} \cdots & y_1 & y_0 \\ \hline & C & z_{n-1} & z_{n-2} \cdots & z_1 & z_0 \end{array}$$

C is overal carry out of addition

- If $c=0$ the correct sum $X+Y$ is $X+Y = z_{n-1} z_{n-2} \cdots z_1 z_0$. In this case an overflow did not occur and the result is within the Dynamic Range.
- If $c=1$ then an overflow occured and the result $X+Y$ is outside the Dynamic Range or $X+Y > 2^n - 1$; (remember that the Dynamic Range of an n-bit integer binary unsigned system is $DR = [0 \quad 2^n - 1]$). In this case the vector $z_{n-1} z_{n-2} \cdots z_1 z_0$ is not the correct sum $X+Y$.

Example: Compute $X+Y$ where X and Y are the following 8-bit binary unsigned numbers: $X = 173_{10} = 10101101_2$; $Y = 44_{10} = 00101100_2$.

Answer: The Dynamic Range of an 8-bit binary unsigned system is $DR = [0 \quad 2^8 - 1] = [0 \quad 255]$.

The addition follows

$$\begin{array}{rcccccccccl}
 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & \text{carry} \\ \hline
 & & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & \\
+ & & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & \\ \hline
 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & \text{sum} \\
 & \uparrow & & & & & & & & &
\end{array}$$

Here overal carry out is $c=0$.

So overflow did not occur. The correct result is $X+Y=11011001_2 = 217_{10}$.

This result is within the Dynamic Range; (remember DR = [0 255]).

In other words $217 < 255 \Rightarrow$ no overflow.

Example: Compute X+Y where X and Y are the following 8-bit binary unsigned numbers: $X = 190_{10} = 10111110_2$;
$Y = 141_{10} = 10001101_2$.

Answer: The Dynamic Range of

an 8-bit binary unsigned system is $DR = [0 \quad 2^8 - 1] = [0 \quad 255]$.

The addition follows

$$
\begin{array}{cccccccccl}
1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & \text{carry} \\
\hline
 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & \\
+ & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & \\
\hline
\underset{\uparrow}{1} & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & \text{sum}
\end{array}
$$

Here overal carry out is $c = 1$

So overflow occured. The obtained result is $X+Y = 101001011_2 = 331_{10}$. This result is outside the Dynamic Range; (remember $DR = [0 \quad 255]$).

In other words $331 > 255 \Rightarrow$ overflow.