MONASH University

Information Technology

Binary and other Value Placed systems

Prepared by:
Ingrid Zuckerman based on CSE1303
Revised by Fabian Bohnert, Graham Farr and Maria Garcia de la Banda

Objectives

- To understand binary and what it can represent
- To understand how value placed systems work
- To be able to convert between (non-negative) integers encoded in binary, octal, decimal and hexadecimal

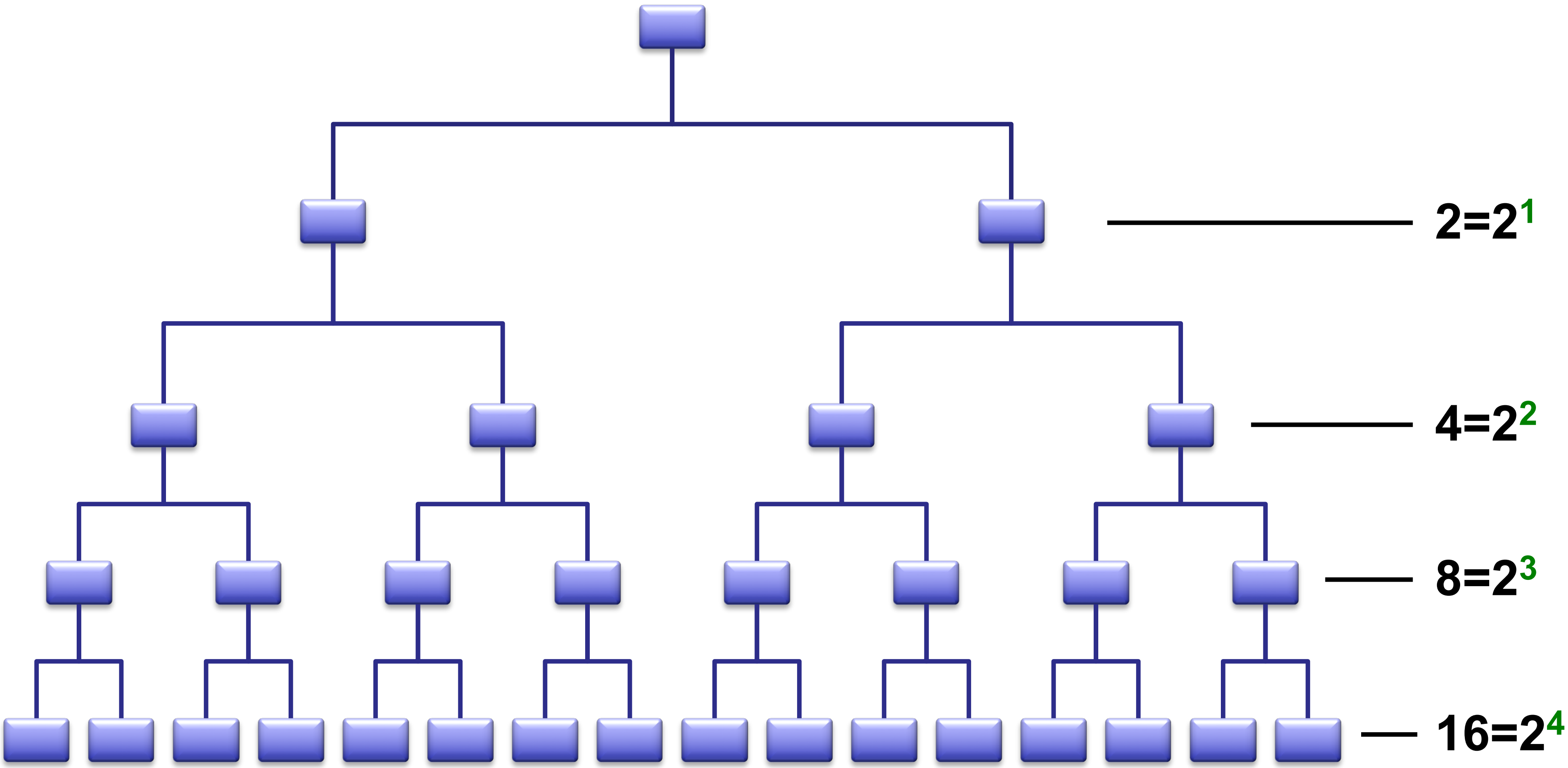
Place Value Systems

Bits, Bytes, Kilo-, Mega-, Giga-, …

- **Bit**: 0 or 1
- **Byte** (B) = 8 bits
- **Word**: chunk of bits (8, 16, 32 or 64) used as basic unit of data in a computer (to store, operate on, move, etc)
  - depends on the computer
  - not always fixed size
- **Kilobyte** (KB) = 1024 bytes = 2^{10} bytes ≈ 10^{3} bytes
- **Megabyte** (MB) = 1024 KB = 2^{20} bytes ≈ 10^{6} bytes
- **Gigabyte** (GB) = 1024 MB = 2^{30} bytes ≈ 10^{9} bytes
- **Terabyte** (TB) = 1024 GB = 2^{40} bytes ≈ 10^{12} bytes
- **Petabyte** (PB) = 1024 TB = 2^{50} bytes ≈ 10^{15} bytes

How many values in N bits?

2^n: think about it in tree form



Thinking binary

- **Since we have 2^N values in N bits:**

2^{10} = about 1000 (“k”)

$2^1 = 2$
$2^2 = 4$
$2^3 = 8$
$2^4 = 16$
$2^5 = 32$
$2^6 = 64$
$2^7 = 128$
$2^8 = 256$
$2^9 = 512$
$2^{10} = 1024$
$2^{11} = 2048$

$2^{12} = 4096$
$2^{13} = 8192$
$2^{14} = 16384$
$2^{15} = 32768$
$2^{16} = 65536$

$2^{20} = 1\,048\,576$

$2^{24} = 16\,777\,216$

$2^{30} = 1\,073\,741\,824$
$2^{31} = 2\,147\,483\,648$
$2^{32} = 4\,294\,967\,296$

2^{20} = about 1000000 (“M”)

Place value systems

- **Each digit worth a value depending on its position in a number**
  - 600 > 060 > 006
- **Ratio between adjacent columns' value is constant**
  - 600 is ten times 060
  - 060 is ten times 006
- **Ratio is called base of number system**
  - Most human cultures now use base 10 (decimal)
  - Other base values are possible

Place value systems

- **What does "123" (decimal) mean?**

place value (scale)

base (radix) is constant

$$10^2 \times 1 + 10^1 \times 2 + 10^0 \times 3$$

exponent (power) increases leftward from 0 at right

digit (multiplier) ranges from 0 to 9 (base−1)

Place value systems

base	
2	10100111100
3	1211122
4	110330
5	20330
6	10112
7	3623
8	2474
9	1748
10	1340
11	1009
12	938
13	7C1
14	6BA
15	5E5
16	53C

Smaller bases: less compact, simpler range of digits

These are all representations of the number 1340 (decimal) in bases from 2 to 16

Higher bases: more compact, wider range of digits

Place value systems

base	
2	10100111100
3	1211122
4	110330
5	20330
6	10112
7	3623
8	2474
9	1748
10	1340
11	1009
12	938
13	7C1
14	6BA
15	5E5
16	53C

Digits with value greater than 9 use letters from the alphabet:

A = ten (10)

B = eleven (11)

C = twelve (12)

D = thirteen (13)

E = fourteen (14)

F = fifteen (15)

etc.

Place value systems

base		
2	10100111100	binary
3	1211122	
4	110330	
5	20330	
6	10112	
7	3623	
8	2474	octal
9	1748	
10	1340	decimal
11	1009	
12	938	
13	7C1	
14	6BA	
15	5E5	
16	53C	hexadecimal (“hex”)

bases common in computing

Represented Unsigned Integers

Representing Unsigned Integers

- **Humans often represent unsigned integers using a base-10 positional notation**
  - So the number 90210 means
    $(90210)_{10} = 9\times\mathbf{10^4} + 0\times\mathbf{10^3} + 2\times\mathbf{10^2} + 1\times\mathbf{10^1} + 0\times\mathbf{10^0}$
- **Computers represent unsigned integers using a base-2 positional notation**
  - So the number 101011 means
    $(101011)_2 = 1\times\mathbf{2^5} + 0\times\mathbf{2^4} + 1\times\mathbf{2^3} + 0\times\mathbf{2^2} + 1\times\mathbf{2^1} + 1\times\mathbf{2^0}$
    $= 1\times\mathbf{32} + 0\times\mathbf{16} + 1\times\mathbf{8} + 0\times\mathbf{4} + 1\times\mathbf{2} + 1\times\mathbf{1} = (43)_{10}$
- **The range of unsigned integers we can represent in N bits is**

$$0, 1, \ldots, 2^N-1$$

Representing Unsigned Integers

- **The first few binary numbers (4-bit, unsigned) are:**

```
0000........ 0
0001........ 1
0010........ 2
0011........ 3
0100........ 4
0101........ 5
0110........ 6
0111........ 7
1000........ 8
1001........ 9
1010........ 10
1011........ 11
1100........ 12
1101........ 13
1110........ 14
1111........ 15
```

Converting Decimal to/from Binary

Bit position

7	6	5	4	3	2	1	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Place value

Converting Decimal to/from Binary

Bit position

7	6	5	4	3	2	1	0
128	**64**	**32**	**16**	**8**	**4**	**2**	**1**

Place value

Converting Binary to Decimal

Example:

Convert the unsigned binary number `10011010` to decimal

1	0	0	1	1	0	1	0

Converting Binary to Decimal

7	6	5	4	3	2	1	0
1	0	0	1	1	0	1	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Converting Binary to Decimal

7	6	5	4	3	2	1	0
1	0	0	1	1	0	1	0
128	64	32	16	8	4	2	1

Converting Binary to Decimal

7	6	5	4	3	2	1	0
1	0	0	1	1	0	1	0
128	64	32	**16**	**8**	4	**2**	1

$$128 + 16 + 8 + 2 = \boldsymbol{\mathit{154}}$$

So, **10011010** in unsigned binary is **154** in decimal

Converting Binary to Decimal

Example:

Convert the decimal number **105** to unsigned binary

Converting Binary to Decimal

Q. Does 128 fit into **105**?

A. **No**

7	6	5	4	3	2	1	0
0							
128	64	32	16	8	4	2	1

Next, consider what's left: still 105

Converting Binary to Decimal

Q. Does 64 fit into **105**?

A. **Yes**

7	6	5	4	3	2	1	0
0	1						
128	64	32	16	8	4	2	1

Next, consider what's left: 105- 64 = 41

Converting Binary to Decimal

Q. Does 32 fit into **41**?

A. **Yes**

7	6	5	4	3	2	1	0
0	1	1					
128	64	32	16	8	4	2	1

Next, consider what's left: 41- 32 = 9

Converting Binary to Decimal

Q. Does 16 fit into **9**?

A. **No**

7	6	5	4	3	2	1	0
0	1	1	0				
128	64	32	16	8	4	2	1

Next, consider what's left: still 9

Converting Binary to Decimal

Q. Does 8 fit into **9**?

A. **Yes**

7	6	5	4	3	2	1	0
0	1	1	0	1			
128	64	32	16	8	4	2	1

Next, consider what's left: 9 – 8 = 1

Converting Binary to Decimal

Q. Does 4 fit into **1**?

A. **No**

7	6	5	4	3	2	1	0
0	1	1	0	1	0		
128	64	32	16	8	4	2	1

Next, consider what's left: still 1

MONASH University

Converting Binary to Decimal

Q. Does 2 fit into **1**?

A. **No**

7	6	5	4	3	2	1	0
0	1	1	0	1	0	0	
128	64	32	16	8	4	2	1

Next, consider what's left: still 1

Converting Binary to Decimal

Q. Does 1 fit into **1**?

A. **Yes**

7	6	5	4	3	2	1	0
0	1	1	0	1	0	0	1
128	64	32	16	8	4	2	1

For integer decimal-to-binary, we STOP here.

Thinking binary

binary value

decimal value		binary value		hexadecimal digit
0	=	0000	=	0
1	=	0001	=	1
2	=	0010	=	2
3	=	0011	=	3
4	=	0100	=	4
5	=	0101	=	5
6	=	0110	=	6
7	=	0111	=	7
8	=	1000	=	8
9	=	1001	=	9
10	=	1010	=	A
11	=	1011	=	B
12	=	1100	=	C
13	=	1101	=	D
14	=	1110	=	E
15	=	1111	=	F

hexadecimal digit

decimal value

Hexadecimal

- **Base 16**
  - Digits 0 to 9, A to F
  - Often called “hex” for short
- **Convenient shorthand for writing binary values**
  - Easier for humans to read
  - Still shows underlying binary representation
  - $16 = 2^4$ (power of two)
- **Computers don’t use hex internally**
  - All memory is bits (binary)

Binary to hex

fill with zeroes at front to make digit count a multiple of four

$$\underline{0101}\,\underline{0011}\,\underline{1100}_2$$

collect binary digits in groups of four digits (“quartets”)

write hex equivalent of each binary quartet

$$53C_{16}$$

Hex to binary

$5FE1_{16}$

$\overline{0101}\ \overline{1111}\ \overline{1110}\ \overline{0001}_2$

write out the binary equivalent of each hex digit

Representing Signed Integers

Representing Signed Integers

- **To handle negative integers, we need to use one of the bits to store the sign**
- **The largest signed integer we can represent in N bits is roughly half the largest unsigned integer**
- **Three representation methods:**
  - Signed magnitude
  - Two's complement
  - Excess-k

Used for exponents for float representation, not studied in FIT1008/FIT2085

Signed Magnitude Representation

- **The most significant bit (MSB) stores the sign**
  - MSB: the one with the highest place value (leftmost)
- **The rest store the absolute value of the number**
  - i.e., its magnitude
- **That is why it is known as "signed magnitude"**
- **The range of signed integers we can represent in N bits is**

$$-(2^{N-1}-1), \ldots, 2^{N-1}-1$$

- **Why? Recall, for unsigned the range with N bits**

$$0, \ldots, 2^{N}-1$$

- **We need one bit for the sign (so $2^{N-1}-1$) and can be negative so… lets see an example**

Signed Magnitude -- Example with 4 bits

- **Representable numbers:**

0000…………………+0
0001…………………+1
0010…………………+2
0011…………………+3
0100…………………+4
0101…………………+5
0110…………………+6
0111…………………+7
1000…………………–0
1001…………………–1
1010…………………–2
1011…………………–3
1100…………………–4
1101…………………–5
1110…………………–6
1111…………………–7

We have +0 and –0

-7, …, 7

$-(2^3-1), \ldots, 2^3-1$

$-(2^{4-1}-1), \ldots, 2^{4-1}-1$

$-(2^{N-1}-1), \ldots, 2^{N-1}-1$

Adding in Signed Magnitude

- **Rule1**
  - If the signs are the same, **add** the magnitudes and use same sign for result
- **Rule2**
  - If the signs differ, determine which integer has the largest magnitude
  - Sign of result: same as sign of integer with the largest magnitude
  - Magnitude: **subtract** smaller magnitude from larger one
- **Note: might get *overflow*, if the sum is too large**
  - That is, if the resulting number is outside the range
    $-(2^{N-1}-1), \ldots, 2^{N-1}-1$
  - Try to encode -7 + (-3) with 4 bits…

Two's Complement Representation

- **Aim: makes arithmetic operations easy, regardless of the signs of the operands**
- **For N bits, -M is represented as $2^N – M$**
- **For example, for N= 4:**

  -1 is represented as $2^4 – 1 = 16-1 = 15$ which in binary is 1111
  -8 is represented as $2^4 – 8 = 16-8 = 8$ which in binary is 1000
- **But careful, M must be within range:**
  - Half of the range is positive, half negative. For example, for 4 bits:
    the numbers from 0000 to 0111 are positive and
    the numbers from 1000 to 1111 are negative
- **In general, the range of signed integers we can represent in N bits is $-2^{N-1}, ..., 2^{N-1}-1$**

It was $–(2^{N-1}-1), ..., 2^{N-1}-1$ for signed magnitude. Why?

Two's Complement -- Example with 4 bits

- **Representable numbers:**

0000.....................+0
0001.....................+1
0010.....................+2
0011.....................+3
0100.....................+4
0101.....................+5
0110.....................+6
0111.....................+7
1000.....................–8
1001.....................–7
1010.....................–6
1011.....................–5
1100.....................–4
1101.....................–3
1110.....................–2
1111.....................–1

We have only +0

$-8, \ldots, 7$

$-2^3, \ldots, 2^3-1$

$-2^{4-1}, \ldots, 2^{4-1}-1$

$-2^{N-1}, \ldots, 2^{N-1}-1$

Two's Complement to Decimal

- **For N bits, you can:**
  - Give the MSB a negative weight
    - E.g., if using 8 bits, then the MSB has positional value $\mathbf{-2^7}$, instead of $+2^7$
- **OR:**
  - Convert to decimal as usual
  - If result is $\geq 2^{N-1}$ (means the MSB was 1):
    - Subtract the decimal value from 2^N and negate

Two's Complement – Example

What integer does 101011 represent in a 6-bit computer?

- **Negative weight for MSB**

$$1\times(-2^5) + 0\times 2^4 + 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 1\times 2^0$$
$$= 1\times(-32) + 0\times 16 + 1\times 8 + 0\times 4 + 1\times 2 + 1\times 1$$
$$= (\mathbf{-21})_{10}$$

- **Usual conversion and subtracting 2^N**

$$1\times 2^5 + 0\times 2^4 + 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 1\times 2^0$$
$$= 1\times 32 + 0\times 16 + 1\times 8 + 0\times 4 + 1\times 2 + 1\times 1$$
$$= (\mathbf{43})_{10}$$

Since 43 >= 32 = 2^{6-1}, it represents a negative number.

So compute 2^N- M and negate:

$$-(2^6 - 43) = -(64 - 43) = -21$$

Negating Two's Complement is easy!

0000.....................+0
0001.....................+1
0010.....................+2
0011.....................+3
0100.....................+4
0101.....................+5
0110.....................+6
0111.....................+7
1000.....................–8
1001.....................–7
1010.....................–6
1011.....................–5
1100.....................–4
1101.....................–3
1110.....................–2
1111.....................–1

Flip 0011 into 1100 and add 1 to get 1101

Flip 1101 into 0010 and add 1 to get 0011

Negation with Two's Complement

- **To negate a two's complement number:**
  Flip *all* its bits and then add 1
  - Example (positive number)

00101010	(-0+0+32+0+8+0+2+0)	+42
11010101	(-128+64+0+16+0+4+1)	-43
11010110	(-43 + 1)	-42

  - Example (negative number)

11010110	(-128+64+ 0+16+0+4+2+0)	-42
00101001	(-0 +0+32 +0+8+0+0+1)	+41
00101010	(+41 +1)	+42

Two's Complement to Decimal Alternative

- **If MSB = 0, as usual for unsigned**
- **Otherwise, we know it is a negative number, so:**
  - Negate the binary:
    - Flip 1s by 0s and viceversa,
    - Add 1,
  - Convert as usual for unsigned
  - Negate the resulting decimal

Summary (of the lesson)

- **Value Places Systems**
- **Unsigned integers in binary, octal, decimal and hexadecimal**
- **Signed integers in**
  - Signed Magnitude
  - Two’s complement