CMPT 295

Unit - Data Representation

Lab 1 – Representing integral numbers in memory - unsigned and signed

Lab 1 - Objectives and Instructions

- In this lab, we shall practice the following encoding schemes:
  - Bits ⇔ Unsigned integral (positive numbers)
    - Encoding schemes: **B2U(x)** and **U2B(x**) where **B** stands for **B**inary (the bits in memory) and **U** stands for **U**nsigned integral numbers.
  - Bits ⇔ Signed integral (positive numbers)
    - Encoding schemes: **B2T(x)** and **T2B(x)** where **B** stands for **B**inary (the bits in memory) and **T** stands for Signed integral numbers expressed as **T**wo's complement.
  - Bits ⇔ Signed integral (negative numbers)
    - Encoding schemes: **B2T(x)** and **T2B(x)** where **B** stands for **B**inary (the bits in memory) and **T** stands for Signed integral numbers expressed as **T**wo's complement.

B2U(x) and **U2B(x)** produce the same results as **B2T(x)** and **T2B(x)** when the integral number **x** is positive.

- Instructions:
  - Read and complete each slide, answering the questions
  - Solution are located on the last slides of this lab

We will see in Lecture 3 that ...

Encoding Scheme give meaning to bits in memory

Bits in memory

01100010 01101001
01110100 01110011_2

Encoding Scheme

- ASCII character
- Unsigned integral
- Two's complement
- (signed) integral
- Floating point (IEEE 754)
- Memory Address
- Assembly language
- RGB
- MP3
- ...

Possible meanings

- Letters and symbols
- Positive numbers
- Negative numbers
- Real numbers
- C pointers
- Machine instructions
- Colour
- Audio/Sound
- ...

Bottom line: Which encoding scheme is used to give meaning (interpret) bits in memory depends on the application currently executing (the ***context***)

Unsigned integral numbers

Address	M[]
size-1	
⋮	⋮
0x0003	01000010
0x0002	01101001
0x0001	01110100
0x0000	01110011

- What if the byte at M[0x0002] represented an unsigned integral number, what would its value be?

Called “bit pattern”
Also called “bit vector”

- $X = 01101001_2$ $w = 8$

w =>width of the bit pattern

- Let’s apply the encoding scheme: $B2U(X) = \sum_{i=0}^{w-1} x_i \cdot 2^i$

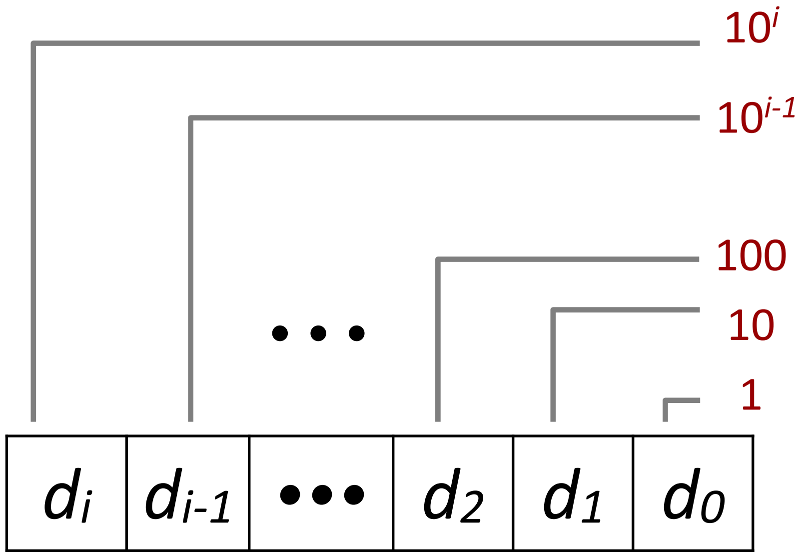
$0 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 =$

- For $w = 8$, range of possible *unsigned values*: [ ]
- For any w, range of possible *unsigned values*: [ ]

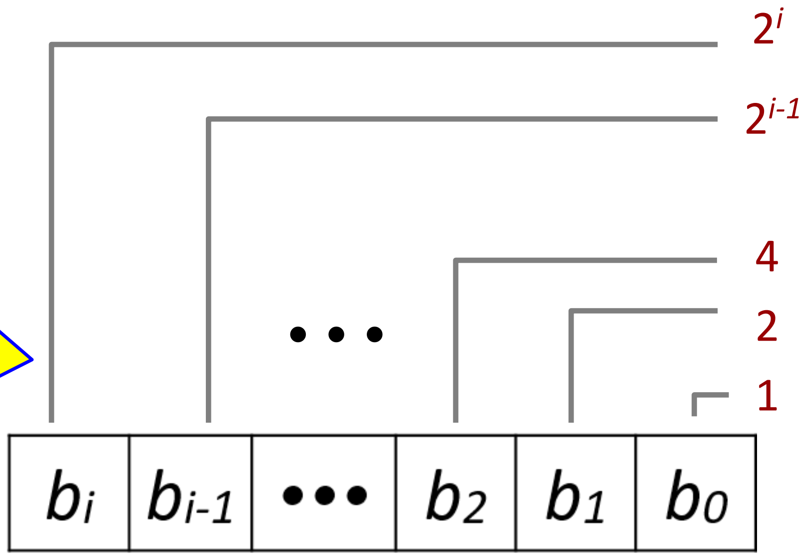
$B2U(X)$ Encoding Scheme (conversion)

- **Positional notation**: expand and sum all terms

Comparing the **decimal numeral system** with …



… the **binary numeral system** highlighting the similarity.



Example: $246_{10} = 2 \times 10^2 + 4 \times 10^1 + 6 \times 10^0$

1's = 10^0
10's = 10^1
100's = 10^2

$$B2U(X) = \sum_{i=0}^{w-1} x_i \cdot 2^i$$

Examples of “Show your work”

$U2B(X)$ Conversion (into 8-bit binary numbers => w = 8)

Method 1 - Using ***subtraction***: subtracting decreasing power of 2 until reach 0

246_{10} => 246 – 128 = 118 -> 128 = 1 x 2^7
118 – 64 = 54 -> 64 = 1 x 2^6
54 – 32 = 22 -> 32 = 1 x 2^5
22 – 16 = 6 -> 16 = 1 x 2^4
6 – 8 = nop! -> 8 = 0 x 2^3
6 – 4 = 2 -> 4 = 1 x 2^2
2 – 2 = 0 -> 2 = 1 x 2^1
0 – 1 = nop! -> 1 = 0 x 2^0

246 => 1 1 1 1 0 1 1 0_2

Method 2 - Using ***division***: dividing by 2 until reach 0

246_{10} => 246 / 2 = 123 -> R = 0
123 / 2 = 61 -> R = 1
61 / 2 = 30 -> R = 1
30 / 2 = 15 -> R = 0
15 / 2 = 7 -> R = 1
7 / 2 = 3 -> R = 1
3 / 2 = 1 -> R = 1
1 / 2 = 0 -> R = 1

246 => 1 1 1 1 0 1 1 0_2

Always check your answer! How?

$U2B(X)$ Conversion – A few tricks

- Decimal -> binary
  - **Trick**: When decimal number is $2^{\mathbf{n}}$, then its binary representation is 1 followed by **n** zeros
  - Let's try: if $X = 32 \Rightarrow X = 2^5$, then $n = 5 \Rightarrow 100000_2$ $(w = 6)$

    What if w = 8?
- Decimal -> hex
  - **Trick**: When decimal number is $2^{\mathbf{n}}$, then its hexadecimal representation is $2^{\mathbf{i}}$ followed by **j** zeros, where **n** = **i** + 4**j** and 0 <= **i** <=3
  - Let try: if $X = 8192 \Rightarrow X = 2^{13}$, then n = 13 and 13 = i + 4j => 1 + 4 x 3

    => 0x2000

    Two ways to check answer:

    1. Convert hex number into a binary number, then use conversion B2U(…)

    2. 1 x $2^{\mathbf{5}}$ = 32,   2 x $16^{\mathbf{3}}$ = 2 x 4096 = 8192

Signed integral numbers

Let's change the content of M[0x0001] to 11110100_2 ?

Address	M[ ]
size-1	
⋮	
0x0003	01000010
0x0002	01101001
0x0001	~~01110100~~ 11110100
0x0000	01110011

- What if the byte at M[0x0001] represented a signed integral number, what would be its value?
- $X = 11110100_2 \quad w = 8$
- Let's apply the encoding scheme:

T => **Two's Complement**

w =>width of the bit pattern

$$B2T(X) = \underbrace{-x_{w-1} \cdot 2^{w-1}}_{\text{Sign bit}} + \sum_{i=0}^{w-2} x_i \cdot 2^i$$

$-1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 =$

- What would be the bit pattern of the …
  - Most negative value (**TMin**):
  - Most positive value (**TMax**):
- For $w = 8$, range of possible signed values: [ ]
- For any w, range of possible signed values: [ ]

Examples of "Show your work"

$T2B(X)$ Conversion -> Two's Complement

$w = 8$

T2B(X) = U2B(T2U(X))

Method 1 If $X < 0$, **(~(U2B(|X|)))+1**

If X = -14

1. |X| => |-14| =
2. **U2B(**14**) =>**
3. **~(**00001110_2**) =>**
4. **(**11110001_2**)+1 =>**

Binary addition:

  11110001
+ 00000001

Method 2 If $X < 0$, **U2B(**X **+ 2^w)**

If X = -14

1. X **+ 2^w =>** -14 +
2. **U2B(**242**) =>**

Using subtraction:

242 – 128 = 114 -> 1 x 2^7
114 – 64 = 50 -> 1 x 2^6
50 – 32 = 18 -> 1 x 2^5
18 – 16 = 2 -> 1 x 2^4
2 – 8 -> nop! -> 0 x 2^3
2 – 4 -> nop! -> 0 x 2^2
2 – 2 = 0 -> 1 x 2^1
0 – 1 -> nop! -> 0 x 2^0

Always check your answer! How?

Solution

Unsigned integral numbers

Address	M[]
size-1	
⋮	
0x0003	01000010
0x0002	01101001
0x0001	01110100
0x0000	01110011

- What if the byte at M[0x0002] represented an unsigned integral number, what would its value be?

Called "bit pattern"
Also called "bit vector"

76543210 ← powers of 2

- X = 01101001_2 $w = 8$

x_7 x_0

w =>width of the bit pattern

- Let's apply the encoding scheme: $\boxed{B2U(X) = \sum_{i=0}^{w-1} x_i \cdot 2^i}$

$\cancel{0 \times 2^7} + 1 \times 2^6 + 1 \times 2^5 + \cancel{0 \times 2^4} + 1 \times 2^3 + \cancel{0 \times 2^2} + \cancel{0 \times 2^1} + 1 \times 2^0 = 64 + 32 + 8 + 1 = 105$

- For w = 8, range of possible *unsigned* values: [0.. 255]
- For any w, range of possible *unsigned* values: [0.. $2^w - 1$]

Examples of "Show your work"

$U2B(X)$ Conversion (into 8-bit binary numbers => $\boldsymbol{w} = 8$)

Method 1 - Using ***subtraction***:
subtracting decreasing
power of 2 until reach 0

MSb

$246_{10} \Rightarrow 246 - 128 = 118 \quad \rightarrow 128 = 1 \times 2^7$
$118 - 64 = 54 \quad \rightarrow 64 = 1 \times 2^6$
$54 - 32 = 22 \quad \rightarrow 32 = 1 \times 2^5$
$22 - 16 = 6 \quad \rightarrow 16 = 1 \times 2^4$
$6 - 8 =$ nop! $\rightarrow 8 = 0 \times 2^3$
$6 - 4 = 2 \quad \rightarrow 4 = 1 \times 2^2$
$2 - 2 = 0 \quad \rightarrow 2 = 1 \times 2^1$
$0 - 1 =$ nop! $\rightarrow 1 = 0 \times 2^0$

LSb

A ✓

MSb LSb
$246 \Rightarrow 1\,1\,1\,1\,0\,1\,1\,0_2$

Method 2 - Using ***division***:
dividing by 2
until reach 0

$246_{10} \Rightarrow 246 / 2 = 123 \quad \rightarrow R = 0$ LSb
$123 / 2 = 61 \quad \rightarrow R = 1$
$61 / 2 = 30 \quad \rightarrow R = 1$
$30 / 2 = 15 \quad \rightarrow R = 0$
$15 / 2 = 7 \quad \rightarrow R = 1$
$7 / 2 = 3 \quad \rightarrow R = 1$
$3 / 2 = 1 \quad \rightarrow R = 1$
$1 / 2 = 0 \quad \rightarrow R = 1$ MSb

Order of bits!

MSb LSb
$246 \Rightarrow 1\,1\,1\,1\,0\,1\,1\,0_2$

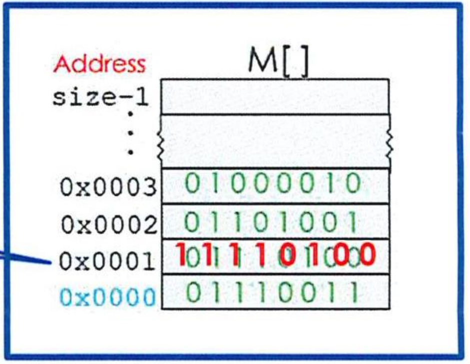
Always check your answer! How? By converting B back to U using B2U(x):

$1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + \cancel{0 \times 2^3} + 1 \times 2^2 + 1 \times 2^1 + \cancel{0 \times 2^0}$

$128 + 64 + 32 + 16 + 4 + 2 = 246$ ✓

Signed integral numbers

Let's change the content of M[0x0001] to 11110100_2 ?



- What if the byte at M[0x0001] represented a signed integral number, what would be its value?
- $\boldsymbol{X} = 11110100_2 \quad \boldsymbol{w} = 8$
- Let's apply the encoding scheme:

T => Two's Complement

w =>width of the bit pattern

$$B2T(X) = \underbrace{-x_{w-1} \cdot 2^{w-1}}_{\text{Sign bit}} + \sum_{i=0}^{w-2} x_i \cdot 2^i$$

$$\boxed{-1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + \cancel{0 \times 2^3} + 1 \times 2^2 + \cancel{0 \times 2^1} + \cancel{0 \times 2^0}} = -128 + 64 + 32 + 16 + 4 = -12$$

- What would be the bit pattern of the ...
  - Most negative value (**TMin**): $10000000_2 = -128$
  - Most positive value (**TMax**): $01111111_2 = 127$

$w=8$

- For $\boldsymbol{w} = 8$, range of possible signed values: [-128 .. 127]

-128 → TMin

127 → TMax

- For any $\boldsymbol{w}$, range of possible signed values: [-2^{w-1} .. $2^{w-1} - 1$]

Examples of "Show your work"

$T2B(X)$ Conversion -> Two's Complement

$w = 8$

$T2B(X) = U2B(T2U(X))$

Method 1 If $X < 0$, **(~(U2B(|X|)))+1**

If $X = -14$

1. $|X|$ => $|-14|$ = 14
2. **U2B**(14) => 00001110_2
3. ~(00001110_2) => 11110001_2
4. (11110001_2)+1 => 11110010_2

flip the bit! (bitwise operator)

Binary addition:

$$\begin{array}{r} 11110001_2 \\ +\ 00000001_2 \\ \hline 11110010_2 \end{array}$$

Method 2 If $X < 0$, **U2B($X + 2^w$)**

If $X = -14$

1. $X + 2^w$ => $-14 + 2^8 = -14 + 256 = 242$ ↓ U
2. **U2B**(242) => 11110010_2

Using subtraction:

242 – 128 = 114 -> 1×2^7
114 – 64 = 50 -> 1×2^6
50 – 32 = 18 -> 1×2^5
18 – 16 = 2 -> 1×2^4
2 – 8 -> nop! -> 0×2^3
2 – 4 -> nop! -> 0×2^2
2 – 2 = 0 -> 1×2^1
0 – 1 -> nop! -> 0×2^0

Always check your answer! How? -> will be shown in Lecture 3

11110010_2

$-128+64+32+16+2$

$= -14$