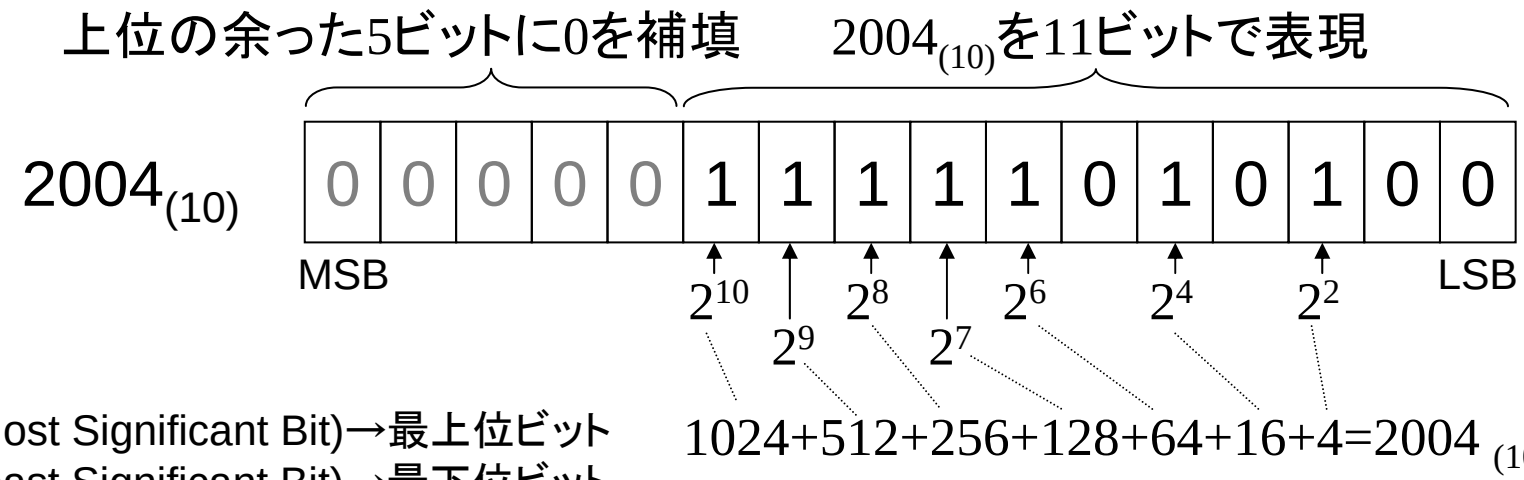
整数の表現

整数の表現

- コンピュータは決まったビット幅を単位にデータを扱う．一般に8ビット機なら8ビット単位で，16ビット機なら16ビット単位でデータを扱う．
- nビットの整数表現では 2^n 種類の整数しか表現できない．
  - 整数は**可算無限**である．
  - コンピュータ上では限られた範囲の整数しか扱えない．
- 整数の表現の例:
  - **符号なし表現**
  - **符号つき絶対値表現**
  - **2の補数表現**

符号なし整数表現

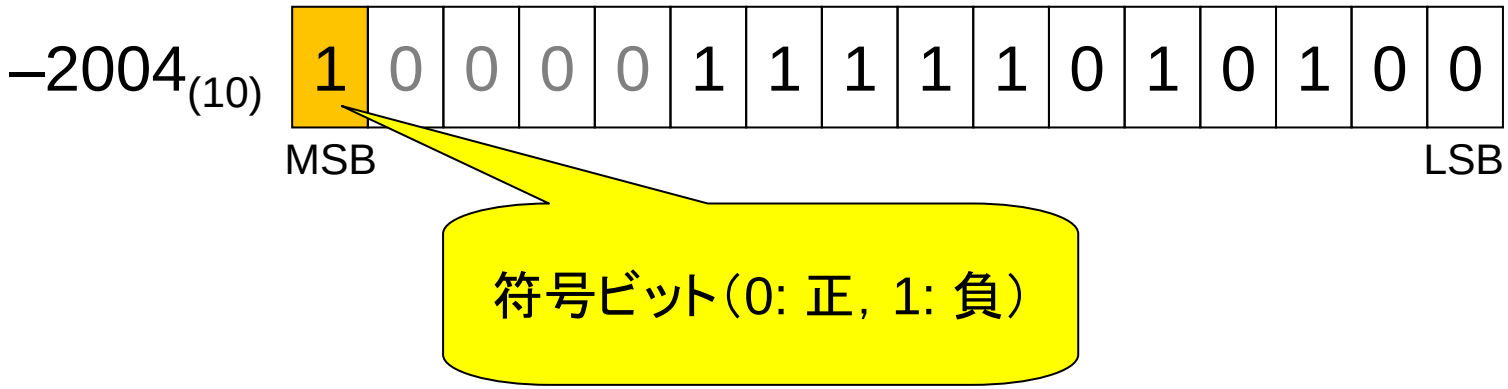
- 決められたビット幅 n に，整数 m を2進数に表現したものをそのまま入れる.
- 上位の余ったビットには 0 を補填する.
- 0 以上 2^n 未満の整数が表現できる.
- 負の数は表現できない.
- C言語の unsigned int/short/long 型は「符号なし整数表現」.



MSB (Most Significant Bit)→最上位ビット
LSB (Least Significant Bit)→最下位ビット

符号つき絶対値表現

- 決められたビット幅 n に，整数 m の絶対値を2進数に表現したものをそのまま入れる.
- このとき上位の余ったビットには 0 を補填する.
- 最上位ビットは符号を表す**符号ビット**として使用する. m が正ならば符号ビットは 0，負ならば符号ビットは 1 にする.
- -2^{n-1} より大きく，2^{n-1} 未満の整数が表せる.
- 0 を表す表現が 000…0 または 100…0 の 2 種類ある.



2の補数表現(1)

- 0000…00 から 0111…11 まで(前半)は 0 と正の数に, 1000…00 から 1111…11 まで(後半)は負の数に使用する.
- 最上位ビットは符号を表す**符号ビット**として機能する. m が非負ならば符号ビットは 0, 負ならば符号ビットは 1 になる.
- 正の数 m は, 決められたビット幅 n に, 整数 m を2進数に表現したものをそのまま入れて表現する. このとき上位の余ったビットには0を補填する.
- 負の数 m は, 整数 $2^n + m$ を2進数で表現したものをそのまま入れて表現する. 最上位ビットは常に 1 になる. (たとえば, 16ビット表現では, –1 は「65535」として表される. 逆に言えば, 65535 を –1 に読み替える約束とする.)
- 最上位ビットには -2^{n-1} の重みがあると考えてもよい.
- -2^{n-1} 以上 2^{n-1} 未満の整数が表現できる.
- C言語の (signed) int/short/long 型は「2の補数表現」.

2の補数表現(2)

16ビット表現の場合:

前半			後半		
ビット表現	符号なし	補数表現	ビット表現	符号なし	補数表現
0000 0000 0000 0000	0	0	1111 1111 1111 1111	65535	–1
0000 0000 0000 0001	1	1	1111 1111 1111 1110	65534	–2
0000 0000 0000 0010	2	2	1111 1111 1111 1101	65533	–3
0000 0000 0000 0011	3	3	1111 1111 1111 1100	65532	–4
0000 0000 0000 0100	4	4	1111 1111 1111 1011	65531	–5
0000 0000 0000 0101	5	5	1111 1111 1111 1010	65530	–6
0000 0000 0000 0110	6	6	1111 1111 1111 1001	65529	–7
…	…	…	…	…	…
0111 1111 1111 1110	32766	32766	1000 0000 0000 0001	32769	–32767
0111 1111 1111 1111	32767	32767	1000 0000 0000 0000	32768	–32768

2の補数表現(3)

2の補数表現による負数のビット表現の簡単な求め方:

① 表現したい負数の絶対値を2進数で表現する.

–5 のビット表現(16ビット)を求める場合, 5 の 2 進数は 101.

② 上位ビットに 0 を補填する.

0000 0000 0000 0101

③ ②で得られた2進数の 0 と 1 を反転する.

1111 1111 1111 1010

④ ③で得られた2進数をひとつカウントアップする.

1111 1111 1111 1011

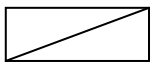
応用問題①: なぜ上記の計算で2の補数表現が正しく得られるのか証明を示せ.

2進表現の比較

3ビット表現の場合:

10進表現	2進表現		
	符号なし整数表現	符号つき絶対値表現	2の補数表現
+7	111		
+6	110		
+5	101		
+4	100		
+3	011	011	011
+2	010	010	010
+1	001	001	001
+0	000	000	000

10進表現	2進表現		
	符号なし整数表現	符号つき絶対値表現	2の補数表現
-0		100	000
-1		101	111
-2		110	110
-3		111	101
-4			100
-5			
-6			
-7			


3ビットでは表現できない

符号なし整数の加減算

10進数の場合と同様に筆算で加減算できる.

キャリー		1	1	1		1			
		0	1	1	1	0	1	1	0	$118_{(10)}$
	+)	0	0	1	1	0	1	0	1	$53_{(10)}$
		1	0	1	0	1	0	1	1	$171_{(10)}$

ボロー			1	1			1		
		0	1	1	0	0	1	1	0	$102_{(10)}$
	–)	0	0	0	1	1	1	0	1	$29_{(10)}$
		0	1	0	0	1	0	0	1	$73_{(10)}$

2の補数表現の加減算

符号を気にすることなく，符号なし整数の加減算とまったく同じ方法で加減算できる．

→同じ回路で加減算できる!（つまり、符号なし整数用と2の補数用に別々の回路を設ける必要はない）

キャリー	~~1~~	1	1			1	1		
		0	1	1	0	0	1	1	0	$102_{(10)}$
+)		1	1	1	0	0	0	1	1	$-29_{(10)}$
		0	1	0	0	1	0	0	1	$73_{(10)}$
ボロー	~~1~~	1			1		1	1		
		0	0	1	1	0	1	1	0	$54_{(10)}$
–)		1	1	0	0	1	0	1	1	$-53_{(10)}$
		0	1	1	0	1	0	1	1	$107_{(10)}$

符号拡張(1)

ビット幅の異なる2進数を加減算する場合は，短いほうの2進数のビットを長いほうにあわせなければならない.

符号なし整数の場合: そのまま0を補填する.

	0	1	1	1	0	1	1	0	$118_{(10)}$
+)	0	0	0	0	0	1	0	1	$5_{(10)}$

2の補数表現の場合はそのまま0を補填すると問題発生!

	0	1	1	1	0	1	1	0	$118_{(10)}$
+)	0	0	0	0	1	0	1	1	$-5_{(10)}$

$11_{(10)}$

符号拡張(2)

符号拡張: 2の補数表現をビット幅を増やすときは，最上位ビットの値を補填する.

正の数の場合:

	0	1	1	1	0	1	1	0	$118_{(10)}$
+)	0	0	0	0	0	1	0	1	$5_{(10)}$

負の数の場合:

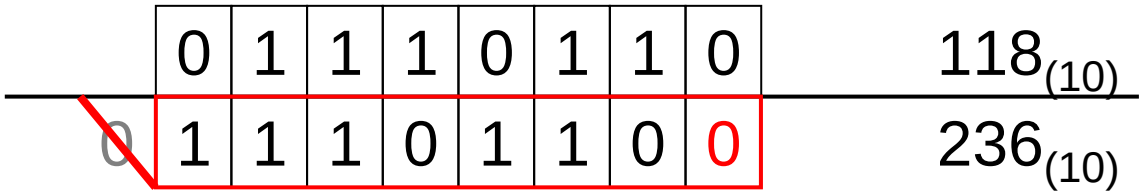
	0	1	1	1	0	1	1	0	$118_{(10)}$
+)	1	1	1	1	1	0	1	1	$-5_{(10)}$

応用問題②: なぜ上記の計算で正しく符号拡張できるのか証明せよ.

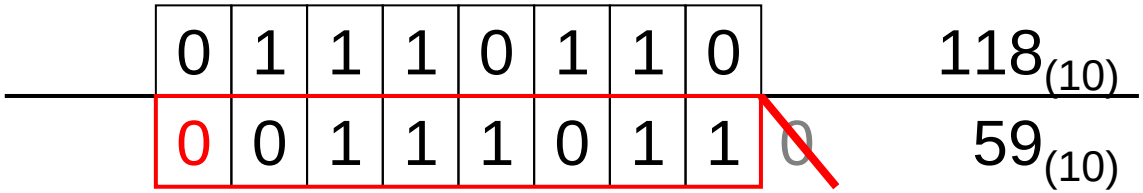
論理シフト演算

論理シフト演算: ビットをひとつ左右にずらし，はみ出したビットを捨てて，空いたビットに 0 を補填する.

左シフト:



右シフト:

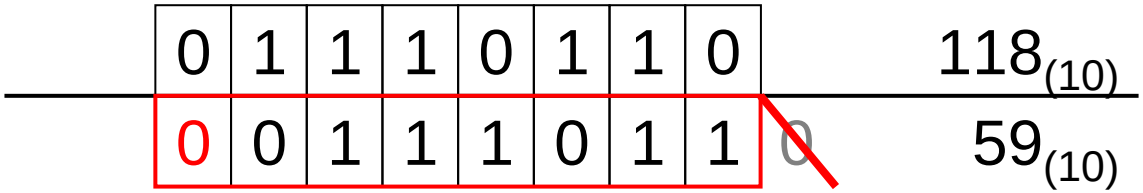


左シフトは2倍することに，右シフトは2で割ることに相当する.

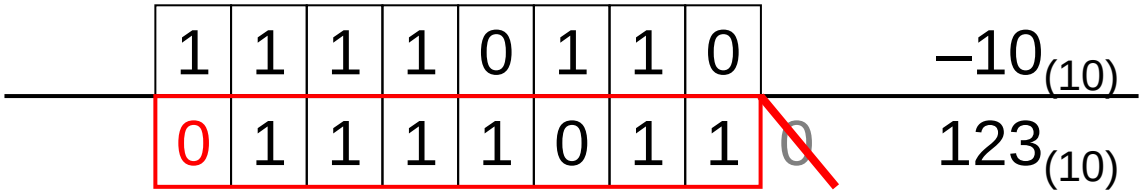
算術右シフト演算(1)

2の補数表現の場合，右シフトが常に2で割ることにはならない．

正の数の場合:



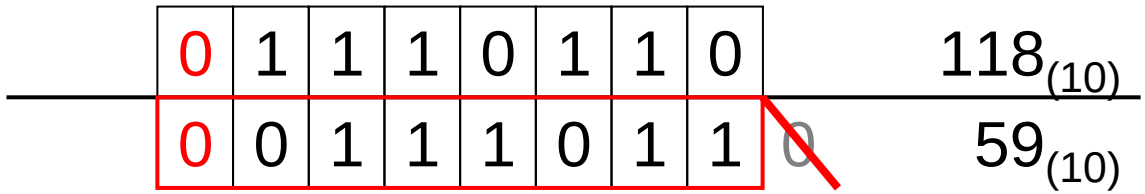
負の数の場合:



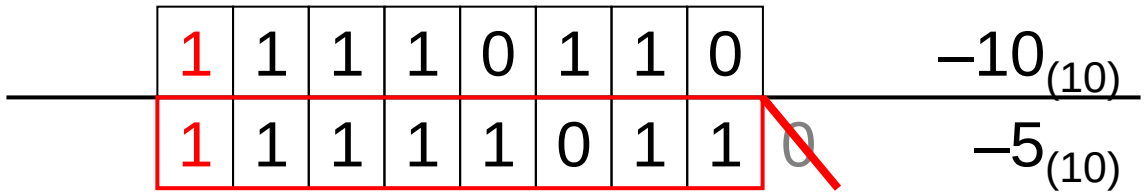
算術右シフト演算(2)

算術右シフト演算: ビットをひとつ右にずらし，はみ出したビットを捨てて，空いたビットに元の数の最上位ビットの値を補填する.

正の数の場合:



負の数の場合:



応用問題①解答例

n bit の2の補数表現された任意の正の整数を m とし，そのビット表現を $a_{n-1}a_{n-2}\ldots a_0$ とする．m は次式で表現される．

$$m = -2^{n-1}a_{n-1} + \sum_{k=0}^{n-2} 2^k a_k$$

ビット a_k を反転したものは $(1 - a_k)$ と表現される．m の全ビットを反転し，1 を加えて得られる値 m' は次式の通り表現され，その値は明らかに $-m$ である．

$$\begin{aligned}
m' &= -2^{n-1}(1-a_{n-1}) + \sum_{k=0}^{n-2} 2^k (1-a_k) + 1 \\
&= -2^{n-1} + 2^{n-1}a_{n-1} + \sum_{k=0}^{n-2} 2^k - \sum_{k=0}^{n-2} 2^k a_k + 1 \\
&= -2^{n-1} + 2^{n-1}a_{n-1} + (2^{n-1}-1) - \sum_{k=0}^{n-2} 2^k a_k + 1 \\
&= -(-2^{n-1} + \sum_{k=0}^{n-2} 2^k a_k) \\
&= -m
\end{aligned}$$

応用問題②解答例

n bit の2の補数表現された任意の整数を m とし，そのビット表現を $a_{n-1}a_{n-2}\ldots a_0$ とする．m は次式で表現される．

$$m = -2^{n-1}a_{n-1} + \sum_{k=0}^{n-2} 2^k a_k$$

この整数を n' bit（$n' > n$）に符号拡張して得られる整数 m' は次式で表現される．

$$m' = -2^{n'-1}a_{n-1} + \sum_{k=n-1}^{n'-2} 2^k a_{n-1} + \sum_{k=0}^{n-2} 2^k a_k$$

$m' - m$ は以下の通り 0 であり，符号拡張により値が変わらないことが証明される．

$$\begin{aligned} m' - m &= a_{n-1}(-2^{n'-1} + \sum_{k=n-1}^{n'-2} 2^k + 2^{n-1}) \\ &= a_{n-1}\{-2^{n'-1} + 2^{n-1}(2^{n'-n} - 1) + 2^{n-1}\} \\ &= 0 \end{aligned}$$

演習問題

10進数「+19」を

1）8ビット符号なし整数表現

2）8ビット符号つき絶対値表現

3）8ビット2の補数表現

の2進数に変換せよ！

Step1：2進数へ変換

基数 → 2) 19 ・・・ 1 ← 余り

2) 9 ・・・ 1 ← 商

2) 4 ・・・ 0

2) 2 ・・・ 0

2) 1 ・・・ 1

0

$19_{(10)} = 10011_{(2)}$

Step2：各表現へ変換

1）符号なし整数

0	0	0	1	0	0	1	1

2）符号つき絶対値

0	0	0	1	0	0	1	1

3）2の補数

0	0	0	1	0	0	1	1

演習問題

10進数「－19」を

1）8ビット符号つき絶対値表現

2）8ビット2の補数表現

の2進数に変換せよ！

Step1：「+19」を2進数へ変換

基数 → 2) 19 ・・・ 1 ← 余り

2) 9 ・・・ 1 ← 商

2) 4 ・・・ 0

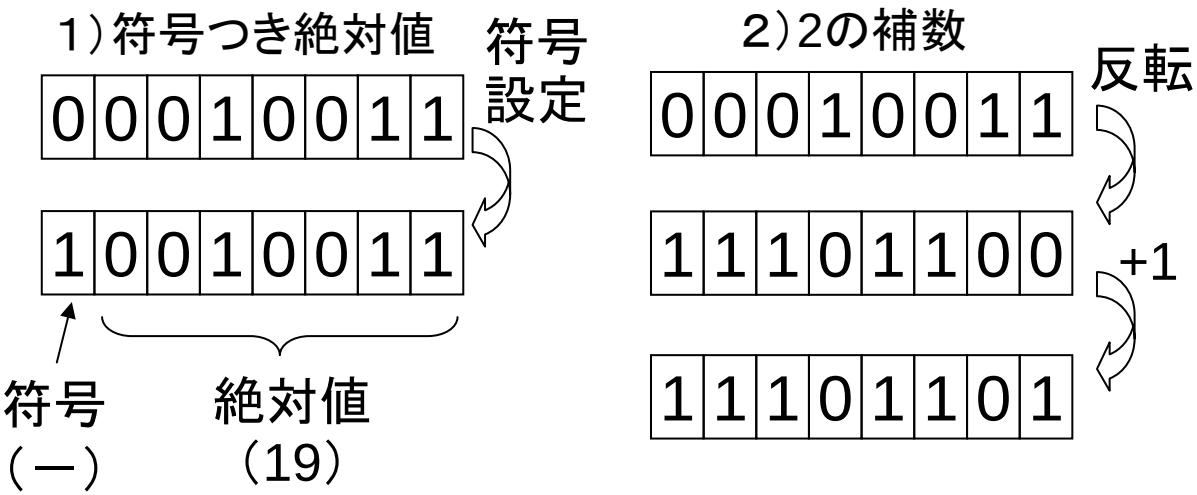
2) 2 ・・・ 0

2) 1 ・・・ 1

0

$19_{(10)} = 10011_{(2)}$

Step2：各表現へ変換



演習問題

10進数「+42」を

1）16ビット符号なし整数表現

2）16ビット符号つき絶対値表現

3）16ビット2の補数表現

の2進数に変換せよ！

Step1：2進数へ変換

基数 → 2) 42 ・・・ 0 ← 余り

2) 21 ・・・ 1 ← 商

2) 10 ・・・ 0

2) 5 ・・・ 1

2) 2 ・・・ 0

2) 1 ・・・ 1

0

$42_{(10)} = 101010_{(2)}$

Step2：各表現へ変換

1）符号なし整数

0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0

2）符号つき絶対値

0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0

3）2の補数

0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0

演習問題

10進数「－42」を

1）16ビット符号つき絶対値表現

2）16ビット2の補数表現

の2進数に変換せよ！

Step1：2進数へ変換

基数 → 2) 42 ・・・ 0 ← 余り

2) 21 ・・・ 1 ← 商

2) 10 ・・・ 0

2) 5 ・・・ 1

2) 2 ・・・ 0

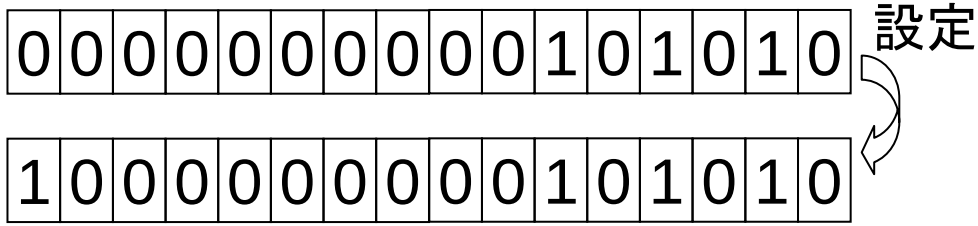
2) 1 ・・・ 1

0

$42_{(10)} = 101010_{(2)}$

Step2：各表現へ変換

1）符号つき絶対値



2）2の補数