東北大学 工学部 機械知能・航空工学科
2017年度 5セメスター・クラスC3 D1 D2 D3

計算機工学

3. 数の表現 — 符号つき整数

(教科書1.2節)

大学院情報科学研究科

鏡 慎吾

http://www.ic.is.tohoku.ac.jp/~swk/lecture/

負の数の表現

素朴な方法 (符号と絶対値法)

通常我々は，10進数の絶対値の前に － をつけて負の数を表す．
同様に，最上位ビットで符号を表し，残りで絶対値を表せばよい

10111011

符号ビット:　　絶対値
0: 正
1: 負

問題点:
- ゼロの表現が2通り存在する
- 加減算が煩雑になる

→ 通常は，2の補数表示と呼ばれる方式が用いられる

「2の補数」表現による符号つき数

3ビットの場合:

2進のビット列	符号なし数	2の補数表現による符号つき数
111	7	-1
110	6	-2
101	5	-3
100	4	-4
011	3	3
010	2	2
001	1	1
000	0	0

000から1ずつ減らして行ったときの表現を素直に考えるとよい

- 一般にnビットの場合，
  -2^{n-1} ～ $(2^{n-1}-1)$
  の範囲の数を表せる
- MSBが1であれば，負の数である

C言語では，各種の整数型に符号なし，符号つきの種類がある.

signed int, unsigned int
signed short, unsigned short
signed char, unsigned char

2の補数の定義

n ビットの 2 進数において，ある数 x の
「2の補数 (2's complement)」とは，

$2^n - x$

である
(8 ビットなら，256 – x になる)

2の補数表示による n ビットの符号つき数とは，

- 非負の数 x ($0 \leqq x \leqq 2^{n-1} - 1$) を x の符号なし2進表示で，
- 負の数 –x ($0 < x \leqq 2^{n-1}$) を x の「2の補数」，すなわち ($2^n - x$) の符号なし2進表示で

表したものである

2の補数が使われる理由

符号を気にせず加算・減算を実行することができる

$$
\begin{array}{rll}
 & 01111010 & = 122_{(10)} \\
+) & 11111111 & = -1_{(10)} \\
\hline
 & 101111001 & = 121_{(10)}
\end{array}
$$

はみ出したビットは捨てる

なぜこのようにうまく行くのか？ → **循環している**のがミソ

8ビットの場合，2^8を足すと一巡して元の数に戻る

$2^8 - 1$ を足すと，元の数より 1 少ない数に落ち着く

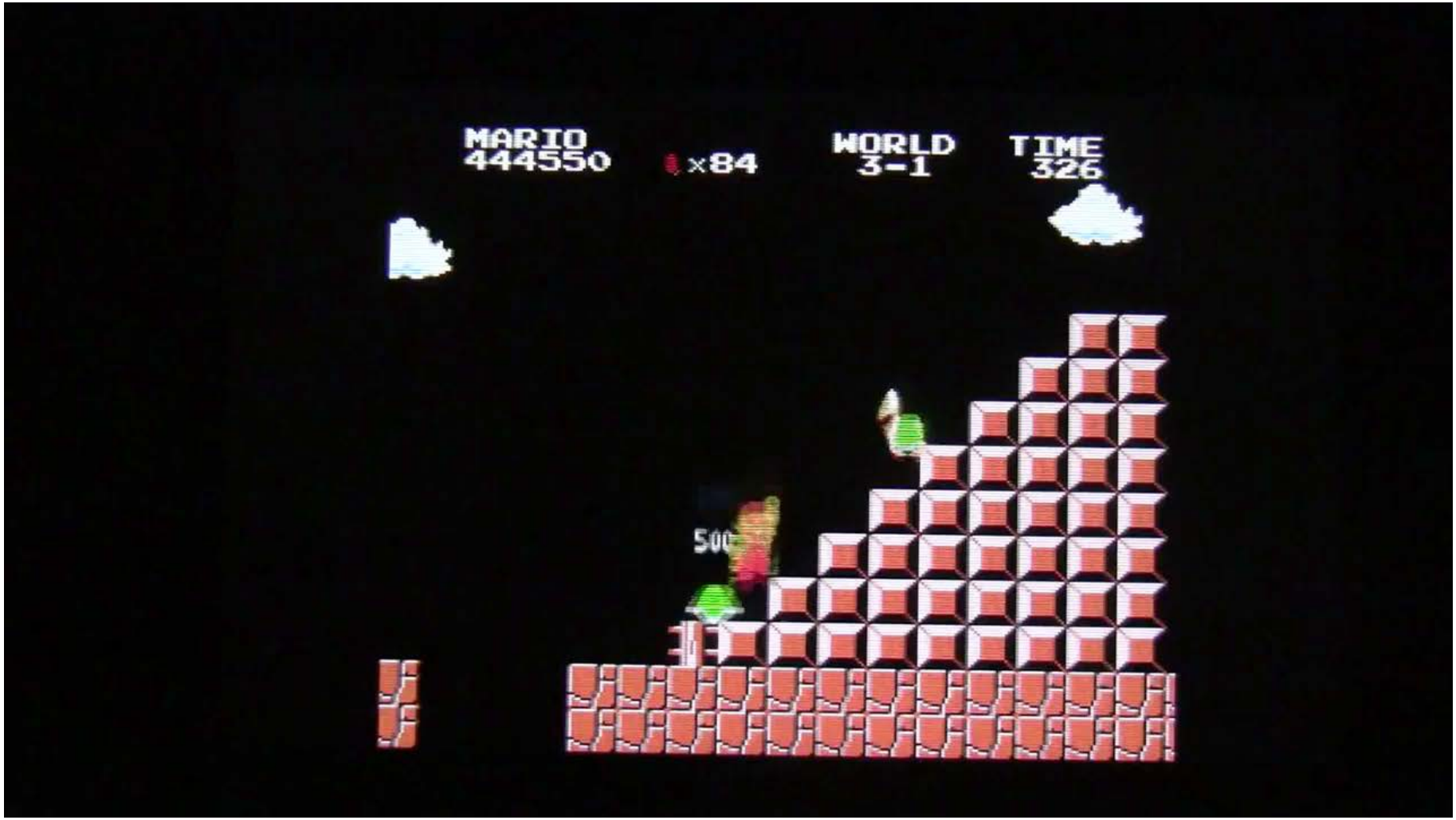
$2^8 - x$ を足すと，元の数より x 少ない数に落ち着く（= 減算）

（符号と絶対値法だと，数の並ぶ順序が変わってしまうのでこうはいかない:
0, 1, 2, …, 126, 127, -0, -1, -2, …, -126, -127)

例題

ファミリーコンピュータ用ゲーム「スーパーマリオブラザーズ」(任天堂(株), 1985年) には，「無限1up」「無限増殖」などと呼ばれるテクニックが存在したことが知られている．すなわち，ある操作を行うことで，プレイヤストック数 (ゲームオーバになるまでに許容されるミスの回数，残機数) を際限なく増加させ続けることが可能であった．

一方，プレイヤストックを一定数以上に増やしすぎると，その後たった一度のミスでゲームオーバになってしまうという現象も生じた．内部でどのような処理が行われていたか推測して述べよ．



https://www.youtube.com/watch?v=FvbWaPfscGg

例: アリアン5型ロケット打ち上げ失敗

1996年6月4日，欧州宇宙機関が70億ユーロをかけて開発したアリアン5型ロケットは，打ち上げ37秒後に爆発した

- 慣性基準装置が16ビット符号つき数からのオーバフローでエラーを起こした
- そのエラー診断番号がフライトデータだとみなされてブースターおよびエンジンのノズル方向が制御された
- 迎角が20度以上になり，分解，爆発した
- バックアップの慣性基準装置も同じエラーを起こしていた

J. L. Lions et al.: ARIANE 5, Flight 501 Failure, Ariane 501 Inquiry Board Report, 1996.
http://www.bloomberg.com/news/photo-essays/2012-08-07/when-software-catastrophe-strikes

2の補数表現の符号つき数の操作

- 加減算
  - 加算はそのまま計算するだけだということが分かった
  - 減算は，「引く数」に負号をつけて2の補数で表し，加算を実行すればよい: 100 – 30 = 100 + (-30)
  - では，符号反転を行うには?

- 符号反転（2の補数変換）
  - 「ビットを反転して1を足す」

- 2の補数表現 と 普通の10進正負の数の相互変換

符号反転

$x = 30_{(10)} \rightarrow 00011110_{(2)}$

$-x = -30_{(10)} \rightarrow ?_{(2)}$

x の2の補数 $= 2^n - x = (2^n - 1) - x + 1$

111…111（1がn個並んだもの）

```
   11111111   ←  繰り下がりが起きない!
-) 00011110
-------------
   11100001   ←  結局 1 と 0 を反転させるだけ
                 (「1の補数」と呼ばれる)
   11100001
+) 00000001   ←  それに1を足すと，符号反転結
-------------    果が得られる
   11100010
```

結論: 符号反転をするには，各ビットを反転し，1を加えればよい

正→負，負→正 のどちらでもOK （∵$2^n - (2^n - x) = x$）

10進の正負の数との変換

10進 → 2進:

- 正の数であれば，そのまま2進数で表示
- 負の数であれば，

  方法1) 絶対値を2進表示し，符号反転

  方法2) 絶対値を 2^n から引いたものを2進表示

2進 → 10進:

- MSBが0であれば非負なので，そのまま10進数へ変換
- MSBが1であれば負なので，

  方法1) 符号反転処理をしてから10進数へ変換し，負号をつける

  方法2) 10進数へ変換したものを 2^n から引いて，負号をつける

例題

1. 10010110 (2進8ビット，2の補数表現の符号つき数) を10進数に変換せよ

2. –50 (10進数) を 8 ビットの2の補数表現の符号つき2進数で表せ.

例題 解答例

1. MSB = 1 なので負の数である.

方法1) まず符号反転処理してから10進に変換し，負号をつける

$$
\begin{array}{ll}
 & 10010110 \\
\rightarrow & 01101001 \\
+) & \qquad\quad 1 \\
\hline
 & 01101010 \rightarrow 106 \rightarrow -106
\end{array}
$$

方法2) まず10進数に変換して，それを256から引いて負号をつける

10010110（正数だと思って変換）

→ 150

→ −(256−150)= −106

2. **方法1)** 50 を符号反転する. $50 = 32 + 16 + 2 = 00110010_{(2)}$

なので，$11001101_{(2)} + 1 = 11001110_{(2)}$

方法2) $256 - 50 = 206 = 128 + 64 + 8 + 4 + 2 = 11001110_{(2)}$

符号拡張とゼロ拡張

あるビット長の2進数を，より長いビット長の2進数に変換するとき，MSB 側を符号ビットで埋める方法を**符号拡張**と呼ぶ．2の補数表示の符号つき数の拡張の際に用いられる．

16ビットから32ビットへの符号拡張の例:

00000000 00001000 = 8 (10)

→ 00000000 00000000 00000000 00001000 = 8 (10)

11111111 11111000 = –8(10)

→ 11111111 11111111 11111111 11111000 = –8(10)

(–8 :「あと8 増やせば 2^n になる数」)

これに対して，常にゼロで埋める方法を**ゼロ拡張**と呼ぶ．符号なし数の拡張に用いられる．

練習問題

8ビットの符号つき数

(a) 00101100
(b) 11101101
(c) 10100101

をそれぞれ，

1) 10 進数，16進数で表せ
2) 符号を反転した数を，2の補数表示の符号つき数で表せ
3) 16ビットに符号拡張せよ
4) (a) を 3ビット右シフトし，元の数の8分の1になっているかどうか調べよ.

練習問題 解答例

1) 10進数: (a) 44 (b) – 19 (c) – 91
   16進数: (a) 2c (b) ed (c) a5
2) (a) 11010100 (b) 00010011 (c) 01011011
3) (a) 00000000 00101100 (b) 11111111 11101101
   (c) 11111111 10100101
4) 000000101 (44/8 の小数点以下切捨てになっている)

1) 正の数は普通に変換する. 負の数は, 10進にしてから 2^8 から引くか, 符号反転してから10進にするか. どうせ(2)で符号反転するんだから, 後者の方が楽かも.
2) ビット反転して 1 を足す.
3) MSBで拡張する.
4) 定義どおり計算.