計算機工学

第３章 数値データの表現法

教科書 コンピュータアーキテクチャの基礎, 柴山潔先生著(京都工芸繊維大学)

参考書 コンピュータの構成と設計, パターソン&ヘネシー

基数(radix)

10進数 10数える毎に, 桁が上がっていく. 人間の手が5本指だから?

2進数 2数える毎に桁が上がっていく. $(10)_2=(2)_{10}$

- コンピュータ内部では, 1=電源電圧(5V, 3.3V, 1.5V. etc.), 0=0V
- 2進数の1桁は, "ビット". 8ビット=1バイト

8進数 8数える毎に桁が上がっていく.

- 2進数では, 3ビット分. $(011)_2=(3)_8$, $(101)_2=(5)_8$, $(110101)_2=(65)_8$
- C言語では, 0を数字の頭につけると8進数表現.
- 例: $(0710)_8=7*64+1*8=(456)_{10}$

16進数 16数える毎に桁が上がっていく．2進数では，4ビット．

- $(a)_{16}=(10)_{10}$, $(b)_{16}=(11)_{10}$, $(c)_{16}=(12)_{10}$, $(d)_{16}=(13)_{10}$, $(e)_{16}=(14)_{10}$, $(f)_{16}=(15)_{10}$,
- 16進数2桁で，8ビット(1バイト)．$(0101)_2=(5)_{16}$, $(1110)_2=(e)_{16}$
- C言語では，0xをつけると，16進数．xは，hexa(6)のx

2進	0000	0001	0010	0011	0100	0101	0110	0111
16進	0	1	2	3	4	5	6	7
2進	1000	1001	1010	1011	1100	1101	1110	1111
16進	8	9	a	b	c	d	e	f

基数変換

- 2, 8, 16進数間は, **非常に**簡単. 単なる置き換え
- 10進数と他の基数の間は, 少々ややこしい.
- 数学としては, 中学生レベル.

$$
\begin{aligned}
(0101)_2 &= (0,101)_2 = (05)_8 \\
(11000101)_2 &= (1100,0101)_2 \\
&= (c5)_{16} \\
(b0c5)_{16} &= (1011,0000,1100,0101)_2 \\
&= (1,011,000,011,000,101)_2 \\
&= (1,3,0,3,0,5)_8 \\
(0101)_2 &= 1*2^2+1*2^0 \\
&= 1*4+1 = (5)_{10} \\
(e13b)_{16} &= (14,1,3,11)_{10\text{進化した}16\text{進}} \\
&= 14*16^3+1*16^2+3*16^1+11*16^0 = 57659
\end{aligned}
$$

10進数から他の基数への変換

- 2の倍数 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, ..., 65536
- 10進数を, 2の倍数で割り算 (=引き算) していく.
- 一旦2進数に変換すると, 8, 16進数は簡単.

$(3950)_{10}/2048=$ 1 余り 1902
$(1902)_{10}/1024=$ 1 余り 878
$(878)_{10}/512=$ 1 余り 366
$(366)_{10}/256=$ 1 余り 110
$(110)_{10}/128=$ 0 余り 110
$(110)_{10}/64=$ 1 余り 46
$(46)_{10}/32=$ 1 余り 14
$(14)_{10}/16=$ 0 余り 14
$(14)_{10}/8=$ 1 余り 6
$(6)_{10}/4=$ 1 余り 2
$(2)_{10}/2=$ 1 余り 0
$(0)_{10}/1=$ 0 余り 0

商を, 上から順に並べて, $(3950)_{10} = (1111, 0110, 1110)_2 = (\mathrm{f6e})_{16}$

小数の２進数変換

1. ２倍する.
2. １の位が１になったら, その桁は1
3. １の位が１なら, １を引く.
4. 最初に戻る.

停止する例(誤差無し)

$(0.5625)_{10}$

$0.5625 \times 2 =$ 1.125
$0.125 \times 2 =$ 0.25
$0.25 \times 2 =$ 0.5
$0.5 \times 2 =$ 1.0
0.0

停止しない例(誤差あり)

$(0.3)_{10}$

$0.3 \times 2 =$ 0.6
$0.6 \times 2 =$ 1.2
$0.2 \times 2 =$ 0.4
$0.4 \times 2 =$ 0.8
$0.8 \times 2 =$ 1.6
$0.6 \times 2 =$ 1.2
$0.2 \times 4 =$ 0.4
無限に続く

近似値と誤差

- 小数が２で割りきれる場合，誤差は出る
- 割りきれない場合，誤差が出る

$(0.3)_{10}$を２進数で表現するときの誤差.

循環小数	$(0.0\dot{1}00\dot{1})_2 = (0.3)_{10}$
２桁	$(0.01)_2 = (0.25)_{10}$
５桁	$(0.01001)_2 = (0.2815)_{10}$
９桁	$(0.010011001)_2 = (0.298828125)_{10}$

補数表現による加減算

加算 数を増やすこと.

減算 数を減らすこと.

補数表現による演算 ２進数を循環させて, 減算 (数を減らす) を加算 (数を増やす) 演算と見なす.

計算機上の演算 補数表現を使用して, 加減算を同じ回路で実現する.

２進数の１の補数と２の補数

１の補数 ビット反転．足すと全ビット1(＝0: １の補数表現で)

２の補数 ビット反転＋1．足すと全ビット0(＝0)

最上位ビットは，ともに符号ビット

元の２進数	１の補数	２の補数
001100	110011	110100
110100	001011	001100
000000	111111	000000

元の２進数	符号	２の補数	符号つきの値
0011	正の数	1101	＋3
1011	負の数	0101	−5

2進数の循環輪(詳細は6章で)

10進(符号無視)	2進	2の補数	1の補数	
3(11)	1011	3	3	
2(10)	1010	2	2	
1(9)	1001	1	1	
0(8)	1000	0	0	桁あふれは無視
7	111	−1	−0	
6	110	−2	−1	
5	101	−3	−2	
4	100	−4	−3	
3	011	3	3	
2	010	2	2	
1	001	1	1	
0	000	0	0	

- 2進数では, 2減ずる(表で2段下に下がる)ことは, 6加算する(表で6段上に上がる)ことと同じ.
- 6を−2と定義すれば, 加算=減算となる.
- ただし, 桁あふれに注意.

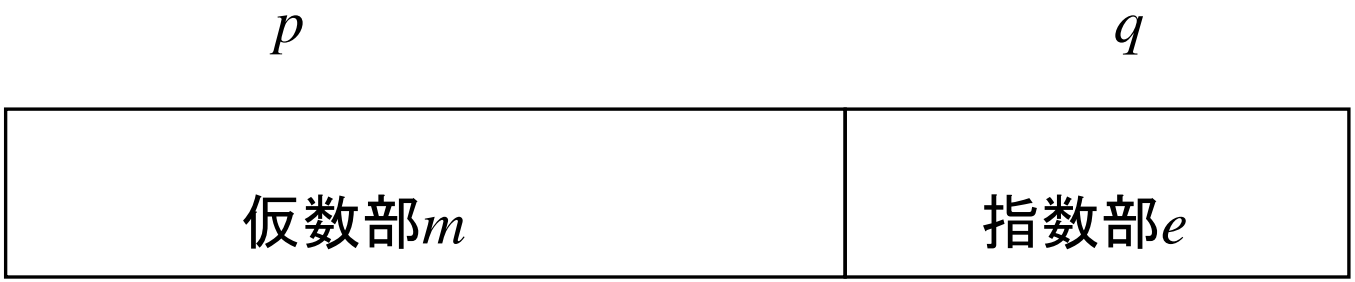
固定小数点表現と浮動小数点表現

固定小数点 (Fixed Point Number) kビットの2進数をnビットの整数部と, mビットの小数部に分けて表現する ($k = n + m$). 整数表現は, 固定小数点表現で$m = 0$の場合. 表せる数の範囲が狭い. nとmのビット幅は可変

浮動小数点 (Floating Point Number) 数を仮数部(pビット)と指数部(qビット)に分けて表現. $k = p + q$

$$m \times 2^e$$

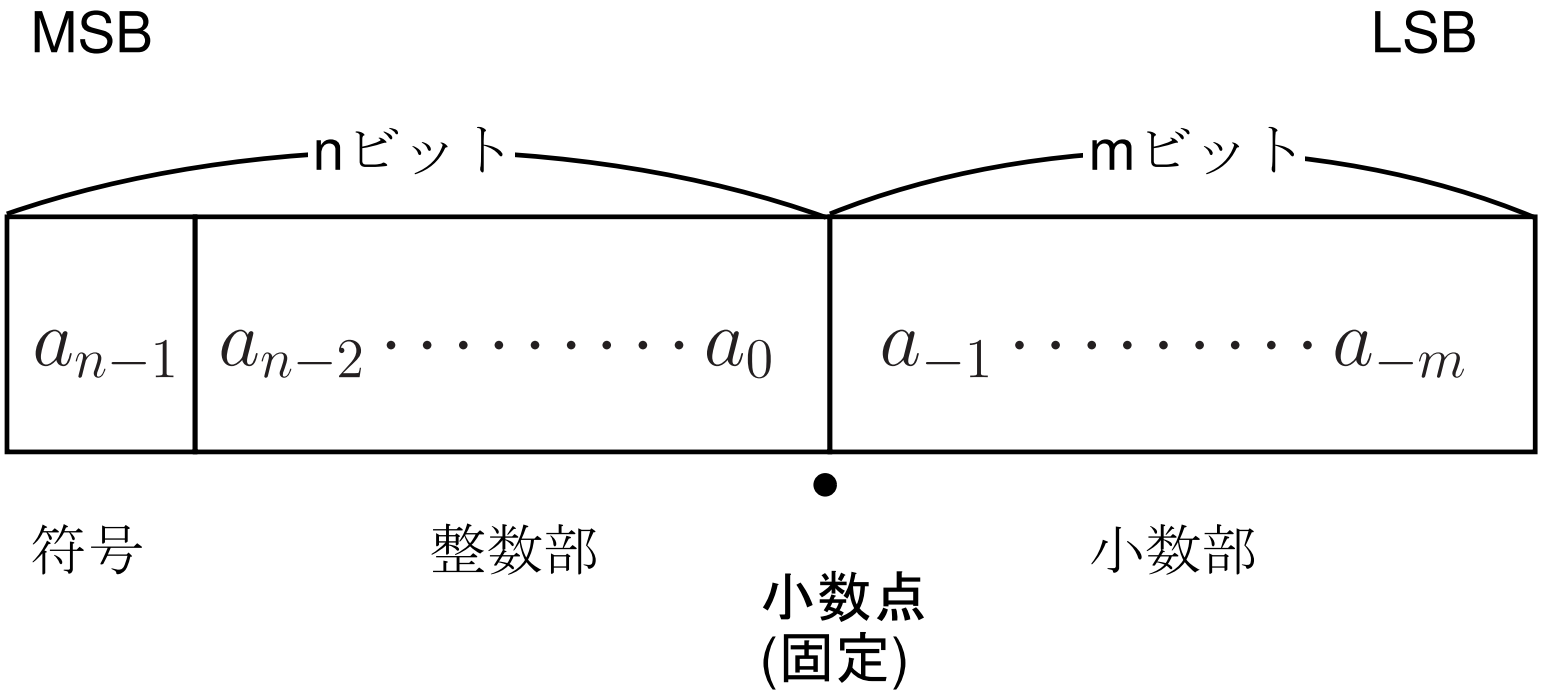
pとqのビット幅の振り分け方には, 標準がある.



教科書P87 図3.7より

２進数の固定小数点(Fixed Point Number)表現

- 小数点位置を**固定**して，数を表現.
- 小数点位置が右端だと整数.
- 小数点の位置は，適宜決める.
- 見ただけでは，どんな数を表現しているか判別不能.
- MSB: Most Significant Bit(Byte), LSB: Least Significant Bit(Byte)



教科書P80 図3.4より

浮動小数点(Floating Point Number)表現

表現方法 仮数部と指数部で表現 (1.12×2^{15})

科学的記数法 (Scientific Notation) 3.2×10^{-2} (小数点の左側が1桁),

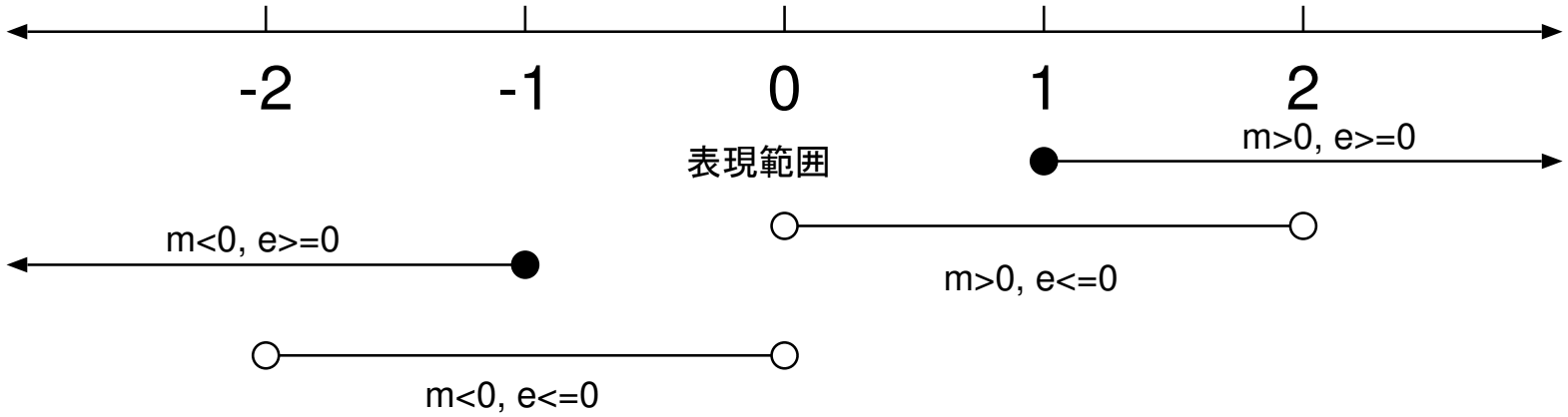
正規化 教科書の定義は誤り．科学的記数法で整数部が非0のもの.

非正規化		正規化後
$(314)_{10}$	$=$	3.14×10^{2}
$(0.0314)_{10}$	$=$	3.14×10^{-2}
$(11.0001)_{2}$	$=$	1.10001×2^{1}
$(0.0110001)_{2}$	$=$	1.10001×2^{-2}

正規表現時の浮動小数点の表現範囲

2進数 $R = m \times 2^e$ **の表現範囲**

$$
\begin{aligned}
1 &\leq m < 2 && (m > 0) \\
-2 &< m \leq -1 && (m < 0) \\
1 &\leq R < 2^{e+1} && (m > 0, e \geq 0) \\
0 < 1 \times 2^e &\leq R < 2 && (m > 0, e \leq 0) \\
-2^{e+1} &< R \leq -1 && (m < 0, e \geq 0) \\
-2 &< R \leq -1 \times 2^e < 0 && (m < 0, e \leq 0)
\end{aligned}
$$



正の整数の32ビット固定小数点表現(unsigned int)では,

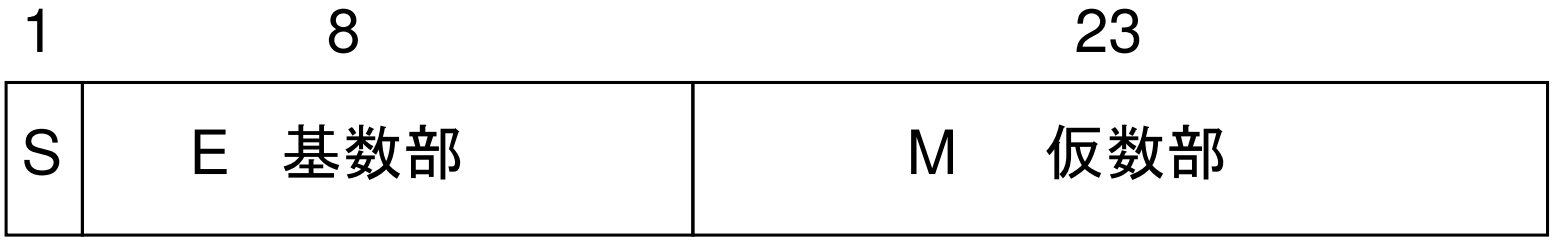
$$0 \leq N < 2^{32}$$

単精度，倍精度 Cでは，float(1ワード，32ビット)，double(2ワード，64ビット).

浮動小数点を格納する形式

$m = S(\text{符号})M, e = E$

IEEE/ANSI形式



教科書P92 図3.13より

E 基数に127を加えて，正数化(バイアス表現)

M 正規化後の，整数部は常に$(1)_2$であるので，整数部を省略して絶対値を格納(けち表現，隠しビット).

S 仮数部の符号

ソートしやすいように定められている．Sの値で条件分岐して，あとは，E,Mをまとめて整数と見て大小比較すれば良い.

C言語による浮動小数点表現の表示

```
// float.c
#include <stdio.h>
#include <math.h>
main()
{
  float a=1.5;
  float b=1.5*pow(2,-15); // 1.5*2^-15
  float c=1.75*pow(2,-20);// 1.75*2^-20
  unsigned char *e; //char型のポインタ(1バイト毎にアクセス可能)
  e=(unsigned char *)&a; // aのアドレスをeに代入.
  printf("a=%02x,%02x,%02x,%02x\n",*(e+3),*(e+2),*(e+1),*(e+0));
  e=(unsigned char *)&b;
  printf("b=%02x,%02x,%02x,%02x\n",*(e+3),*(e+2),*(e+1),*(e+0));
  e=(unsigned char *)&c;
  printf("c=%02x,%02x,%02x,%02x\n",*(e+3),*(e+2),*(e+1),*(e+0));
}
```

eを, float型のポインタとすると, 4バイト毎にしかアクセスできない.

実行結果

```
a=3f,c0,00,00
b=38,40,00,00
c=35,e0,00,00
```

確認してみよう

	S	E	M	S,E,M
1.5×2^{-15}	0	$(-15 + 127 = 112)_{10}$	$(.5)_{10}$	0,112,0.5
1.75×2^{-20}	0	$(-20 + 127 = 107)_{10}$	$(.75)_{10}$	0,107,0.75

	S,E,M	2進化	16進化
1.5×2^{-15}	0, 112, 0.5	0,{0111,0000},{100,0_{20}}	38,40,00,00
1.75×2^{-20}	0, 107, 0.75	0,{0110,1011},{110,0_{20}}	35,e0,00,00

(0_n=0がn個連なる)

浮動小数点演算(詳細は6章)

固定小数点の演算と比べて，浮動小数点の演算は，加減乗除のどれが難しい?

```
1500+252023087=
1500-252023087=
1500*252023087=
1500/252023087=
1.5e-12+2.5202e-8=
1.5e-12-2.5202e-8=
1.5e-12*2.5202e-8=
1.5e-12/2.5202e-8=
```

浮動小数点での計算精度

- float型では，仮数部が24ビットしかないので，24ビット分しか精度がない．

$$
\begin{aligned}
2^{-23} &= .0000,0000,0000,0000,0000,0001)_2 \\
&= .00000011920928955078125 \\
&\approx 1.19 \times 10^{-7} \\
2^{-24} &= (.0000,0000,0000,0000,0000,0000,1)_2 \\
&= 0.000000059604644775390625 \\
&\approx 5.96 \times 10^{-8}
\end{aligned}
$$

- 通常，浮動小数点の計算は，float型で行なわれるので，小数点7桁以下の値を議論しても意味がない．
- 異なる基数部の加減算を行なうとさらに精度が落ちる．

コード化による数/文字の表現

２進コード化10進数 10進数を２進数で表現．10進数の１桁毎に，４ビット使用してコード化.

$$(9\_7\_1\_8)_{10} = (1001\_0111\_0001\_1000)_{\mathrm{BCD}}$$

ASCIIコード 制御コード(改行等)，英数字，記号を，７ビットで表現する．８ビット目は未使用．いわゆる半角カナは，８ビット目が１に割り当て.

日本語文字コード 漢字は８ビットでは表現不可能．16ビット(２バイトで表現).

JIS 区点コード．94×94の配列で表現．７ビットコードを２個使って表現．漢字の始まりと終わりに特別な制御コード(エスケープシーケンス)を付加して．１バイト２バイト文字を区別.

SJIS シフトJIS．１バイト目を(ASCII+半角カナ)が使っていないところに割り当てる．２バイト目は7ビットもしくは８ビット．WINDOWSはSJIS

EUC １バイト目も２バイト目も８ビット．(JISの８ビット目を１に)．UNIXはEUC

Unicode 世界規格の２バイト文字表現法．中国，日本，韓国で異形同義字を同じコードに割り当てている．MacはUNICODE

各漢字コード

- 此れはJISコードです.

```
0000000 1b 24 42 3a 21 24 6c 24 2c 1b 28 42 4a 49 53 1b
    esc  $  B  :  !  $  l  $  , esc  (  B  J  I  S esc
0000020 24 42 25 33 21 3c 25 49 24 47 24 39 1b 28 42 2e
     $  B  %  3  !  <  %  I  $  G  $  9 esc  (  B  .
0000040 20 00
     sp nul
```

- 此れはSJISコードです.

```
0000000 8d 9f 82 ea 82 aa 45 55 43 83 52 81 5b 83 68 82
     cr us stx  j stx  *  E  U  C etx  R soh  [ etx  h stx
0000020 c5 82 b7 2e 20 00
     E stx  7  . sp nul
```

- 此れはEUCコードです.

```
0000000 ba a1 a4 ec a4 ac 45 55 43 a5 b3 a1 bc a5 c9 a4
     :  !  $  l  $  ,  E  U  C  %  3  !  <  %  I  $
0000020 c7 a4 b9 2e 20 00
     G  $  9  . sp nul
```

各漢字コードの使われ方

SJIS Microsoft Windowsの標準漢字コード.

JIS 電子メールの送受信に使われる.

EUC UNIX系のOSで主に利用される.

UTF-8(Uni Code) Apple社のOSの標準漢字コード. Red Hat Linuxでも採用されている.

- 欧米のプログラムの場合, コメントなどは, EUCコードにしておくと, 比較的問題が少ない.
- ただし, 8ビット目を無視する場合があるので, 日本語を通すと, 問題となる場合がある.

```
int main(){
    int i,j;
    printf("←メイン関数です. \n");
....
```

EUCコードで保存

```
int main(){
  int i,j;
  printf("",%a%$4X?t$G$9.\n");
}
```

8ビット目を落とすと, ←の1バイト目が" となりエラーとなる.

身近な?16進数

- ８ビットの時代のゲーム機．(「255個」でgoole検索)

  「ＦＦ３ではアイテム変化技により全てのアイテム（イベントアイテムや没アイテムまでも）を９９個入手する事が可能です。しかし、さらにそれを凌駕しアイテムの個数を２５５にする事が可能なのです。」

IPアドレスとMACアドレス

MACアドレス 機器(ネットワークの端子)固有にもつ，48ビットの物理アドレス．

- 世界中で唯一無二．2^{48}=約$2^{16}*2^{32}$台=65536*40億台まで可能．
  例: 00:0D:60:13:2B:55, 00:80:e0:22:50:40

IPアドレス 機器を区別するために，論理的に付けられた32ビットのアドレス．

例 130.54.28.31 (通常は8ビット毎に10進数で表示する．)

インターネットと2進数

アドレスを手動設定するときのパラメータ

	10進表記	16進表記
IPアドレス	192.168.0.5	c0.a8.00.05
ネットマスク	255.255.0.0 (16ビット)	ff.ff.00.00
ブロードキャストアドレス	192.168.255.255	c0.a8.ff.ff
ゲートウェイ	192.168.0.1	c0.a8.00.01

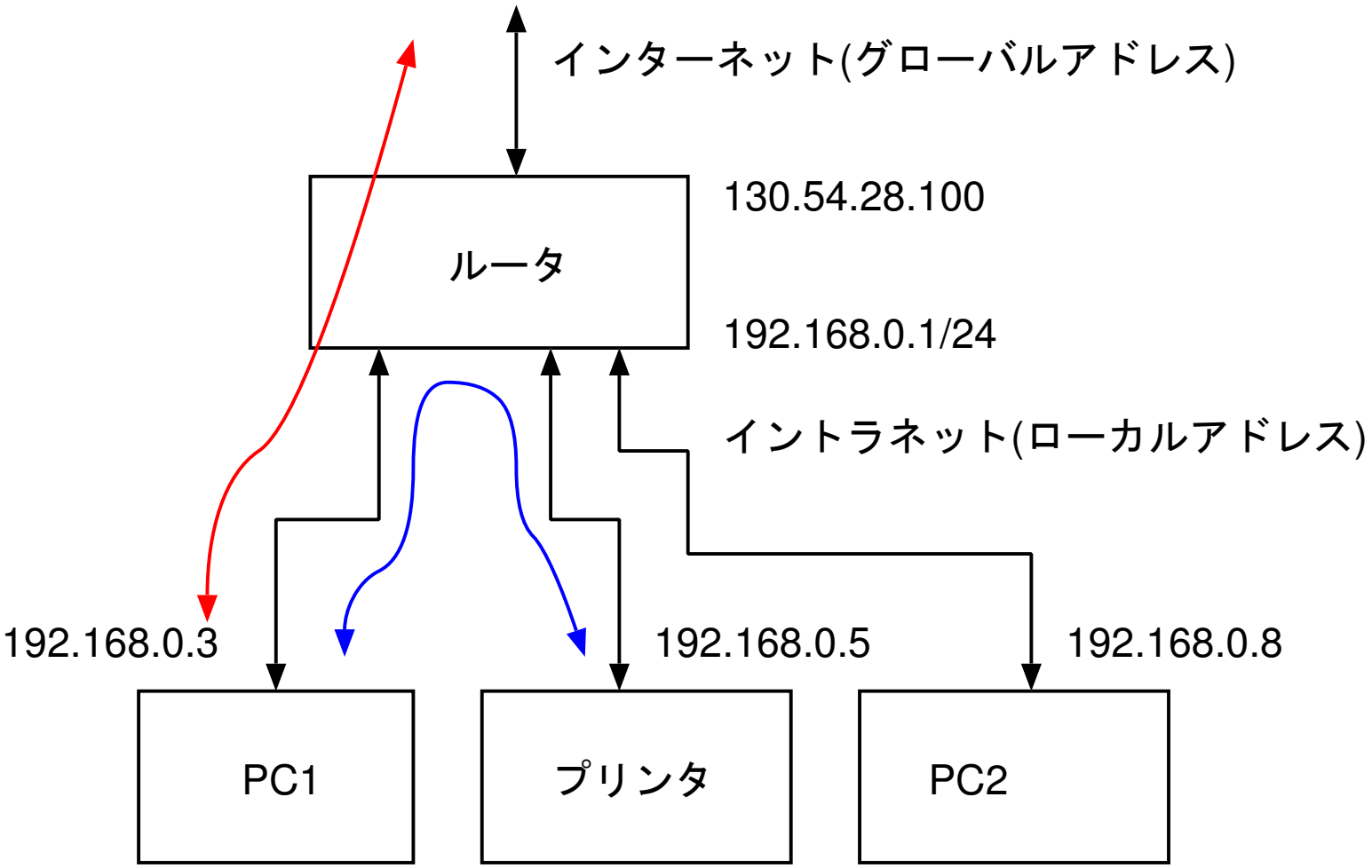
	2進表記
IPアドレス	1100,0000.1010,1000.0000,0000.0000,0101
ネットマスク	1111,1111.1111,1111.0000,0000.0000,0000
ブロードキャストアドレス	1100,0000.1010,1000.1111,1111.1111,1111
ゲートウェイ	1100,0000.1010,1000.0000,0000.0000,0001

同一ネットワークの範囲 IPアドレス AND ネットマスク ＝ 192.168.0.0 /16 (192.168.
から16ビット(192.168.255.255まで)

ブロードキャストアドレス 同一ネットワークの全機器に問い合わせをするためのアドレス. (全ビット1)

ゲートウェイ 異なるネットワークと通信するための出口

ネットワーク図


インターネット(グローバルアドレス)
130.54.28.100
ルータ
192.168.0.1/24
イントラネット(ローカルアドレス)
192.168.0.3
192.168.0.5
192.168.0.8
PC1
プリンタ
PC2