

コンピュータ・サイエンス1

第9回 コンピュータでの情報の扱い方(5)

人間科学科コミュニケーション専攻
白銀 純子

第9回の内容

- コンピュータでの情報の扱い方(5)

前回の出席問題の解答(設問1)

- 設問1: 以下の「(ア)」と「(イ)」に入る数を答えなさい。
 - 1バイト = $\boxed{(ア)}$ ビット
 - 1Kbyte = $\boxed{(イ)}$ バイト

解答
(ア) 8
(イ) 1024

前回の出席問題の解答(設問2)

- 設問3: 「やってみよう!」の10進数 $\Leftrightarrow$ 2の補数の変換問題の1. と3. の計算結果を報告すること
 - 1. の問題: -25を2の補数10桁で表現

- $(-25)_{10} \rightarrow (25)_{10}$ (マイナス記号を取る)
- $(25)_{10} = (0000011001)_2$ (10進数を10桁の2進数に直す)
- $(0000011001)_2 \rightarrow (1111100110)_2$ (1と0を反転する)
- $(1111100110 + 1)_2 = (1111100111)_2$ (1を足す)

解答: $(1111100111)_2$

前回の出席問題の解答(設問2)

- 設問3: 「やってみよう!」の10進数 $\Leftrightarrow$ 2の補数の変換問題の1. と3. の計算結果を報告すること
 - 3. の問題: 2の補数10000を10進数で表現

- $(10000 - 1)_2 = (01111)_2$ (2の補数から1を引く)
- $(01111)_2 \rightarrow (10000)_2$ (1と0を反転する)
- $(10000)_2 = (16)_{10}$ (2進数を10進数に直す)
- $(16)_{10} \rightarrow (-16)_{10}$ (マイナス記号をつける)

解答: $(-16)_{10}$

設問2前回の出席問題の解答(設問3)

- 「やってみよう!」の2の補数の足し算の1. の結果を報告すること
 - 1. の問題: $(+10) + (+8)$ を5桁の2の補数として計算し、10進数として表現

$$\begin{array}{r} (+10)_{10} = (01010)_2 \\ (+8)_{10} = (01000)_2 \end{array}$$

$$\begin{array}{r} 01010 \\ + 01000 \\ \hline 10010 \end{array}$$

負の数と判断

- 2の補数(マイナスの数)として10進数に直す
- $(10010 - 1)_2 = (10001)_2$ (2の補数から1を引く)
 - $(10001)_2 \rightarrow (01110)_2$ (1と0を反転する)
 - $(01110)_2 = (14)_{10}$ (2進数を10進数に直す)
 - $(14)_{10} \rightarrow (-14)_{10}$ (マイナス記号をつける)

解答: -14

● 前回の質問の回答

● 情報処理技術者試験

- 国家資格で情報処理技術全般について問われる試験
 - 一番の入門編: ITパスポート
 - 専門や職業にかかわらず、身に着けておくべき情報処理の知識
 - 中級程度の試験: 基本情報技術者試験
 - 情報科学系を専門とする人にとって、身につけておくべき基本的な知識
 - 情報科学系を専攻する大学生が3・4年生くらいで挑戦する試験
 - 上級程度の試験: 応用情報技術者試験
 - 情報科学系を専門とする人が目指すべき高度な知識
 - 情報科学系を専攻する大学院生が2年生くらいで挑戦する試験
- その他にも、各種分野に特化した試験
 - セキュリティ、データベース、ネットワーク、etc.

WebClassに
自習教材あり

● $(-10)_{10}$ の2進数表記

- 前回授業資料のp. 15

Ex.

$(-10)_{10}$

$= (-01010)_2$

$\rightarrow (10101 + 1)_2 = (10110)_2$

2進数での負数は、最も大きな桁が1になるので、
そうなるように適当に5桁に桁あわせしただけ

● 2の補数に関するオーバーフロー

- 試験問題の計算では...

1. 足し算やシフト算などの計算をする
2. 計算結果が指定の桁数を超えていれば、超えた分の桁を削除する
(桁数あわせ)
3. 計算結果の先頭の桁が0か1かで、正か負を判断する
 - 正の数(先頭の桁が0)であれば、割り算だけで10進数に直す
 - 負の数(先頭の桁が1)であれば、2の補数の方法で10進数に直す

問題文に、「2の補数を考慮する」と

➤ 書かれていなければ、1. の結果を解答とする(必要に応じて10進数に直す)

➤ 書かれていれば、3. の結果を解答とする

→必ずしも、計算結果そのものが解答とは限らない!

● やってみよう[6]の2. の問題

- $(-10) + (+8)$ を5桁の2の補数として計算し、10進数として表現
 - $(-10)_{10} + (+8)_{10} = (10110)_2 + (01000)_2 = (11110)_2$
 - 最も大きな桁が「1」なので、2の補数(つまり負数)として10進数に直す
 - $(11110)_2 \rightarrow (11110 - 1)_2 = (11101)_2 \rightarrow (-00010)_2 = (-2)_{10}$

● 32ビットの2進数

- 最小値: $(10000000000000000000000000000000)_2$
 $= (-2147483648)_{10}$
 - 最大値: $(01111111111111111111111111111111)_2$
 $= (2147483647)_{10}$
 - 最大値(2147483647)に1を足すと、最小値(-2147483648)になる
= オーバーフロー
 - 最大値(2147483647)より大きい数は32ビットでは扱えない
 - ✓ 日本の国家予算(約100兆円)
 - ✓ 世界の人口(2017年時点で約76億人), etc.
 - 普段の生活で使っている数くらいでは、オーバーフローにならないことが多い
- ちなみに...64ビット: -9223372036854775808 ~ 9223372036854775807

教科書の演習問題

- 教科書の演習問題の解答は、出版社のところで見たらなかった
 - 授業では、教科書の演習問題はほとんど使わない

Question!

お知らせ

- 授業のページの下の方に、計算問題の追加練習問題を掲載
 - 自習として取り組んで、計算にしっかり慣れておこう!

前回の復習

2の補数表現[2](p. 9)

- 2の補数 = 負の数を2進数で表現したもの(コンピュータの世界では)
- 計算方法(例: -20を10桁の2進数に直す)
 - 2の補数に直したい10進数のマイナスを取り除く
 - $(-20)_{10} \rightarrow (20)_{10}$
 1. の結果を2進数に直す
(この時点で桁数あわせ!! 後で桁数あわせをすると、合わせ方を間違えやすい)
 - $(20)_{10} = (0000010100)_2$
 2. の結果の0と1を逆にする(0の桁を1、1の桁を0にする)

$$\begin{array}{cccccccccc} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ \downarrow & & & & & & & & & \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \end{array}$$

2の補数表現[2](p. 9)

- 2の補数 = 負の数を2進数で表現したもの(コンピュータの世界では)
- 計算方法(例: -20を10桁の2進数に直す)
 3. の結果に1を足し算する

$$\begin{array}{r} 1111101011 \\ + \quad \quad \quad 1 \\ \hline 1111101100 \end{array}$$

-20を2進数に直した結果
(2の補数 = 2進数での負の数の表現)

2進数での負の数の表現では、
「-」の記号はつけない

2の補数を10進数に変換[2]

- 計算方法(例: 1111101100を10進数に直す)

- 2の補数から1を引き算する

$$\begin{array}{r} 1111101100 \\ -1 \\ \hline 1111101011 \end{array}$$

1. 結果の0と1を逆にする(0の桁を1、1の桁を0にする)

$$\begin{array}{r} 1111101011 \\ \downarrow \\ 0000010100 \end{array}$$

※2の補数→10進数の方法は、10進数→2の補数の逆

2の補数を10進数に変換[2]

- 計算方法(例: 1111101100を10進数に直す)

2. の結果を10進数に直す

$$(0000010100)_2 = (20)_{10}$$

- 2.3. の結果に-(マイナス)をつける

$$(20)_{10} \rightarrow (-20)_{10}$$

1111101100を
10進数に直した数

※2の補数→10進数の方法は、10進数→2の補数の逆

2進数の引き算[2]

- 2進数の引き算だと...

- ある桁の引かれる数が引く数より小さければ、1つ大きな桁から $(10)_2$ (10進数で2)を借りる

- 2を借りる: 貸した桁から1を引き、借りた桁に2を足す

$$\begin{array}{r} 100 \\ -001 \\ \hline \end{array} \rightarrow \begin{array}{r} 020 \\ -001 \\ \hline \end{array} \rightarrow \begin{array}{r} 012 \\ -001 \\ \hline \end{array}$$

引き算の答え: 011

※コンピュータ的には引き算はしないので、人間が2の補数→10進数の計算をするための引き算

正の数と負の数の見分け方[1]

- 大前提: 数を表す2進数の桁数は決まっている

- 普通のコンピュータで32桁(or 64桁)

ということは...例えば $(10)_{10}$ は、コンピュータ的には...

$$\underbrace{0000...00001010}_{28\text{個の「0」}} \text{と考えている}$$

※授業のスライド中では32桁分も書けないので、そのときどきで適当なところで割愛

正の数と負の数の見分け方[2]

- 負の数(2の補数)の計算方法: 負の数Nを、正の数N(2進数)の0と1を反転させて1を加える

コンピュータ的には32桁で数を表すので...

$$\begin{aligned} (-10)_{10} &\rightarrow (10)_{10} \\ &= (0000...00001010)_2 \\ &\rightarrow ((1111...11110101 + 1)_2 = (1111...11110110)_2 \end{aligned}$$

28個の「0」も全て「1」に反転される

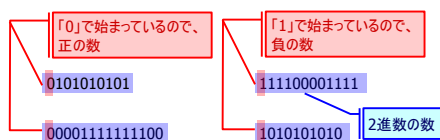
- 負の数は結果的に一番大きな桁(一番左の桁)が「1」になる

- 一番大きな桁(一番左の桁)が「0」であれば正の数、「1」であれば負の数として扱う

正の数と負の数の見分け方[3]

- 2進数を見たときに...(2の補数を考える場合)

- 「2の補数を考える」という場合は、先頭の桁を見て、正の数か負の数かを判断
- 「2の補数を考える」と書かれていない場合は、負の数を考えなくてOK



桁あふれ(オーバーフロー)(p. 11)

- 2の補数に関連した桁あふれ(オーバーフロー)が起こりうる

Ex. 2進数5桁の計算(10進数で14+5の計算)

$$\begin{array}{r} 01110 \\ + 00101 \\ \hline 10011 \end{array}$$

先頭の桁が1になってしまった

先頭の桁が1の場合は、2進数で負の数として扱う

10011
負の数を表す

計算結果: $(-13)_{10}$ (負の数)

2の補数に関連した桁あふれ(オーバーフロー)

負数込みの計算の考え方

- 計算結果を見て...(オーバーフローがどのと考えるのではなく!)

- 計算結果が指定の桁数を超えていれば、超えた分の桁を削除(桁数あわせ)
 - 4桁の2進数の計算結果: $(100110)_2$ → 桁数を超えた部分 = 削除 → 計算結果: $(0110)_2$
- 計算結果の先頭の桁が0か1かで、正か負を判断
 - 正の数(先頭の桁が0)であれば、割り算だけで10進数に直す
 - 負の数(先頭の桁が1)であれば、2の補数の方法で10進数に直す

4桁の2進数の計算結果: $(0101)_2$ → 正の数と判断 → 計算結果: $(5)_{10}$

4桁の2進数の計算結果: $(1010)_2$ → 負の数と判断 → 計算結果: $(-6)_{10}$

小数の表現方法

整数以外の数を表現するには?

- コンピュータが表現できる数: 整数, 小数
- 整数以外の数
 - 小数
 - 分数
 - n乗根(平方根, 立方根, etc)
 - n(円周率)
 - etc.

全て小数として表現

10進数の小数を2進数に直す[1]

- 10進数の整数部分は、通常の方法で2進数に直す
- 10進数の小数部分に2をかけ算する
- 2の結果、整数部分を小数点第1桁にする
- 3の結果、小数部分をまた10進数の数とする
 - 小数部分が0になるまで繰り返す
 - 無限小数になることも(10進数でけりい数も2進数では無限小数になりえる)
 - 2の整数部分を小数点第2桁、第3桁、...と置いていく
 - 1の整数部分と2の整数部分を並べたものが2進数での小数になる

10進数の小数を2進数に直す[2]

- 10進数の小数を2進数に直すには?(例1)
- 10進数の0.8125を2進数に直す

0.8125

$$\begin{array}{r} \times 2 \\ \hline 1.625 \end{array}$$

0.625

$$\begin{array}{r} \times 2 \\ \hline 1.25 \end{array}$$

0.25

$$\begin{array}{r} \times 2 \\ \hline 0.5 \end{array}$$

0.5

$$\begin{array}{r} \times 2 \\ \hline 1.0 \end{array}$$

0.1

0.1101

0になったので計算終了

$(0.8125)_{10} = (0.1101)_2$

10進数の小数を2進数に直す[3]

- 10進数の小数部分を2進数に直すには?(例2)
 - 10進数の3.8を2進数に直す(整数部分の3は2進数で11)

$$\begin{array}{r} 0.8 \\ \times) 2 \\ \hline 1.6 \\ \times) 2 \\ \hline 1.2 \\ \times) 2 \\ \hline 0.4 \\ \times) 2 \\ \hline 0.8 \end{array}$$

$\rightarrow$

$$\begin{array}{r} 0.6 \\ \times) 2 \\ \hline 1.2 \\ \times) 2 \\ \hline 0.4 \\ \times) 2 \\ \hline 0.8 \end{array}$$

$\rightarrow$

$$\begin{array}{r} 0.2 \\ \times) 2 \\ \hline 0.4 \\ \times) 2 \\ \hline 0.8 \end{array}$$

$\rightarrow$

$$\begin{array}{r} 0.4 \\ \times) 2 \\ \hline 0.8 \end{array}$$

0.8になって最初に戻ったので、無限小数

$(3.8)_{10} = (11.110011001100\dots)_2$

2進数の小数を10進数に直す[1]

- 例: 101.1101 (整数部分101は10進数で5)
 - 2進数の整数部分は、通常の方法で10進数に直す
 - 10進数の小数部分各桁の上に「1/2」を書く
 2. で書いた「1/2」の「2」の右肩に左から1, 2, 3, ...と書いていく
 - $1/2^0, 1/2^1, 1/2^2, \dots$ ができていく

$$\begin{array}{r} \frac{1}{2} \frac{1}{2} \frac{1}{2} \frac{1}{2} \\ 0.1101 \end{array}$$

$\rightarrow$

$$\begin{array}{r} \frac{1}{2^0} \frac{1}{2^1} \frac{1}{2^2} \frac{1}{2^3} \\ 0.1101 \end{array}$$

左から右に、1, 2, 3, ...と番号をつける

2進数の小数を10進数に直す[2]

- 各桁の上の「1/2ⁿ」と、それぞれの桁の数をかけあわせる
4. の結果を足し合わせ、1. の整数部分とあわせる

$$\begin{array}{r} \frac{1}{2^1} \frac{1}{2^2} \frac{1}{2^3} \frac{1}{2^4} \\ 0.1101 \end{array}$$

$\rightarrow$

$$\begin{array}{r} \frac{1}{2^1} \frac{1}{2^2} \frac{1}{2^3} \frac{1}{2^4} \\ 0.1101 \\ \times \quad \times \quad \times \quad \times \\ \hline \frac{1}{2^1} \frac{1}{2^2} 0 \frac{1}{2^4} \end{array}$$

足し合わせる

$0.5 + 0.25 + 0.0625 = 0.8125$

1. の整数部分とあわせる

5.8125

小数の表現方式

小数を表現する方法(p. 10)

- 固定小数点方式
- 浮動小数点方式

固定小数点方式[1](p. 10)

- 小数部分の桁数がnの場合: 2進数では、小数部分が $1/2^n$ 刻みで表現

$\downarrow$

nを大きくすると、それだけ小数部分を細かく表現可能

※ただし、実際コンピュータは小数も2進数で考えているが、人間が考えるときの便宜上、10進数で考えることが多い

Question!

Copyright (C) Junichi Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

文字の表現

Copyright (C) Junichi Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

文字の符号化(p. 13)

- 文字: コンピュータは整数に置き換えて扱う(番号をつけて扱う)
 - 文字を2進数で表現する(「**符号化**」と呼ぶ)
- 2進数で表現される文字集合
 - 半角英数文字
 - 図形文字
 - 制御文字
 - 多バイト文字
 - 図形文字

※文字集合: 文字の集まり

Copyright (C) Junichi Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

図形文字と制御文字(p. 13)

- 図形文字**: 通常、画面に表示される文字
 - 人間が明示的に書いたり読んだりする文字
 - アルファベット, 数字, ひらがな, 漢字, 記号, etc.
- 制御文字**: 通常、画面に表示されない文字
 - コンピュータに何らかの制御をするための文字
 - 改行, TAB, ESC, etc.

Copyright (C) Junichi Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

ASCII文字

Copyright (C) Junichi Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

ASCII文字(p. 13)

- ASCII**: American Standard Code for Information Interchange
- 半角文字を表す文字集合
 - アルファベット大文字(26文字)
 - アルファベット小文字(26文字)
 - 数字(10文字)
 - 記号(スペース, 「,」, 「.」, etc.)

1文字を表すために、最低限7ビット必要
(6ビット: 64種類の情報, 7ビット: 128種類の情報)

※1文字を表す2進数の桁数(ビット数)は、どの文字でも同じ(つまり7ビット)

Copyright (C) Junichi Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

● 図形文字(p. 13)

- ASCII: 情報量が7ビットで収まるように、扱う文字を取り決めた文字集合
 - 図形文字: 95文字
 - アルファベット(大文字・小文字): 52文字
 - 数字: 10文字
 - 記号(スペースを含む): 33文字
 - 制御文字: 33文字
 - BackSpace, Delete, Tab, 改行(CRとLF), etc.

● 番号例(p. 14)

番号	文字	番号	文字	番号	文字
47	0	65	A	97	a
48	1	67	B	98	b
49	2	68	C	99	c
50	3	69	D	100	d
51	4	70	E	101	e
52	5	71	F	102	f
53	6	72	G	103	g
54	7	73	H	104	h
55	8	74	I	105	i
56	9	75	J	106	j

「数」としての0～9ではなく、「文字」としての0～9

● ビット数[1](p. 14)

- コンピュータでは8ビットを1つの単位として扱うことが多い
 - ASCII文字も8ビットで表現すると扱いやすい
- 8ビットのうち、7ビット分(2進数で7桁目まで)で文字を表現する
- 残りビット(2進数で8桁目)に常に0を入れておく
 - ASCII文字としては無駄なビット
 - 日本語を表現するときに利用

例えば...

A: 65番(10進数)

= 1000001番(2進数)

= 01000001番(2進数, コンピュータ内での表現)

→ ASCII的には無駄な(何も利用していない・1にはならない)ビット

● ビット数[2](p. 14)

- 8ビットで1文字を表現 = 1バイトで1文字を表現

「1バイト文字」と呼ばれる

Ex. 「Hello, my name is John.」

➢ アルファベット: 17文字

➢ 記号: 2文字

➢ スペース: 4文字

23文字 = 23バイト

● ちなみに...

- アスキーアートも文字コードのASCIIから(ASCII art)
 - アスキーアート: 文字だけで作った絵
 - 感情を表す「(^_^)」のような単純なものから、人や動物に見えるものまで様々
- アスキーアートの例
 - <http://ja.wikipedia.org/wiki/%E3%82%A2%E3%82%B9%E3%82%AD%E3%83%BC%E3%82%A2%E3%83%BC%E3%83%88>
 - <http://bdaa.sakura.ne.jp/zukan/>

● Question!

多バイト文字

Copyright (C) Tokyo Shoin, Inc. Tokyo Woman's Christian University 2016. All rights reserved.

背景[2](p. 15)

- 様々な言語圏の文字: 英語圏の文字と同様に2進数で表現する必要性
 - 英語圏の文字: 128文字で表現可能
 - 1バイト分(256文字分)のうち、128文字分は英語圏の文字
 - 英語圏以外の文字: 128文字以上必要な場合も
 - 日本語
 - 中国語
 - 韓国語
 - etc.

⇒ 128文字では収まらない

1文字を複数のバイト(多バイト)で表現

Copyright (C) Tokyo Shoin, Inc. Tokyo Woman's Christian University 2016. All rights reserved.

文字化け(p. 15)

- 多バイト文字の出現により、文字化けが発生
- 文字化けの原因
 - フォントの問題
 - 文字集合の符号化方式の問題

「文字コード」と呼ぶ

詳細な理由は何であれ...
要は文字を表す2進数(0と1の並び)を、コンピュータが理解していないために発生
➢ その2進数をどのような形でディスプレイに表示して良いかをコンピュータが理解していないため

Copyright (C) Tokyo Shoin, Inc. Tokyo Woman's Christian University 2016. All rights reserved.

フォントの問題[3](p. 15)

- 機種依存文字: コンピュータによって表現のしかたが違う文字
 - それぞれの文字を表現するビット列が、コンピュータによって異なる
 - 1文字1文字を表現するビット列は、JIS(日本の国家規格)などで決まっている
→ コンピュータの環境に依存しない
 - 規格で決められた文字に含まれていない文字もある
機種依存文字
 - Ex. 丸付き数字(①, ②, ...), ローマ数字(I, II, ...), etc.
- 外字: 登録されていない文字を、利用者が作ったもの
 - 人名漢字などを作ることが多い
 - 作ったコンピュータでしか使えない

Copyright (C) Tokyo Shoin, Inc. Tokyo Woman's Christian University 2016. All rights reserved.

符号化方式の問題[1](p. 15)

- 符号化: 1つの文字を2進数(ビット列)として表現すること
- ある1つの文字を表現するビット列が複数通り存在する場合
 - 半角英数の文字はASCIIの1通りだけ
 - 他にも存在するが、ASCIIが世界標準
 - 大部分のコンピュータはASCIIを利用
 - 日本語は複数通り存在

Copyright (C) Tokyo Shoin, Inc. Tokyo Woman's Christian University 2016. All rights reserved.

Copyright (C) Tokyo Shoin, Inc. Tokyo Woman's Christian University 2016. All rights reserved.

次回

- 6月19日は出張のため休講

Copyright (C) Soken Shoin Inc. Tokyo Women's Christian University 2018. All rights reserved.