Monaca

「情報Ⅰ」に向けたプログラミング研修会

～コンピュータの仕組み（JavaScript版）～

アシアル株式会社
アシアル情報教育研究所
岡本 雄樹

Copyright © Asial Corporation. All Right Reserved.

はじめに

Copyright © Asial Corporation. All Rights Reserved.

自己紹介

- 名前
  - 岡本雄樹(アシアル情報教育研究所 所長)
- 著書
  - イラストでよくわかるPHP
  - WordPressプロフェッショナル養成読本
  - Monacaで学ぶはじめてのプログラミング
- メッセージ
  - 「コンピューター」「インターネット」「プログラミング」私は高校生の時にそれらと出会うことで人生が拓けました。先生方とMonacaによるアプリ開発を通じて、情報技術の活用方法や作品作りの楽しさを広めてまいります。

Copyright © Asial Corporation. All Rights Reserved.

■ 教材サイトのご案内

- あんこエデュケーション
  - https://anko.education/
- 教科情報研修資料
  - https://anko.education/joho

Copyright © Asial Corporation. All Rights Reserved.

■ プロジェクトのインポート

- 教材サイトに各種モデルのサンプルプロジェクトがあります
- Monacaでインポートして進めます
- 予めMonacaのログインと空きプロジェクト数の確保をお願いします

Copyright © Asial Corporation. All Rights Reserved.

■ 学習11 コンピュータの仕組み

- コンピュータの構成要素
- 論理演算
- 論理演算を用いた二進数の足し算
- 計算誤差について

Copyright © Asial Corporation. All Rights Reserved.

コンピュータの仕組み（JavaScript版）

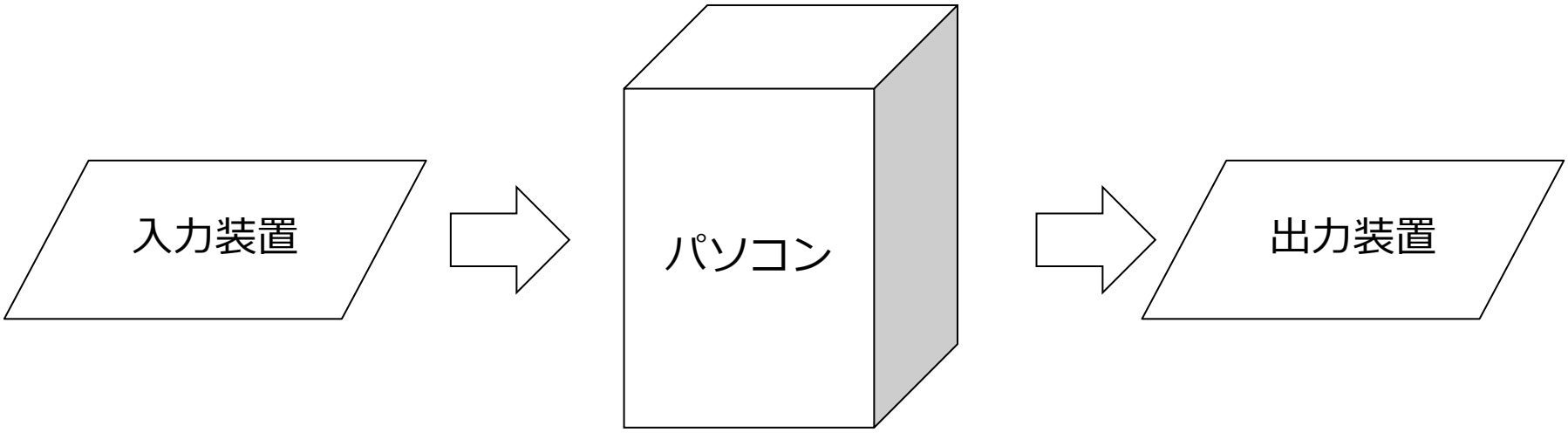
Copyright © Asial Corporation. All Rights Reserved.

コンピュータの構成要素

Copyright © Asial Corporation. All Rights Reserved.

■ パソコンと入力装置と出力装置

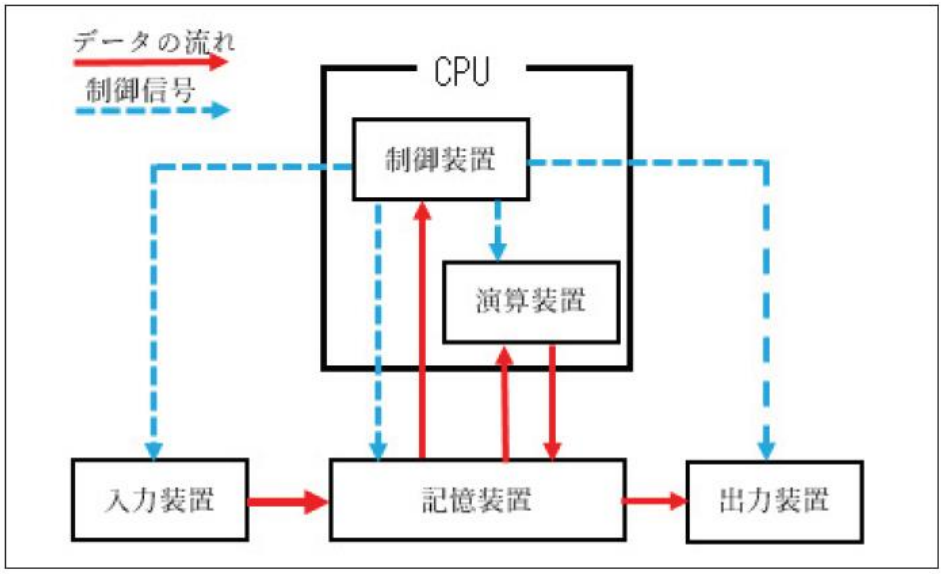
- 入力装置
  - マウス・キーボード・Webカメラなど
- 出力装置
  - プリンター・ディスプレイなど
- 補足
  - 最近では入出力装置との接続はUSB規格による接続が一般的です。また、映像の入出力はHDMIなども使われています。
    ネットワークの話は4章の範囲のため省略します。



Copyright © Asial Corporation. All Rights Reserved.

■ コンピュータの5大装置

- コンピュータの内部は入力装置と出力装置を制御する制御装置と、演算装置、そして記憶装置で構成されています。特に、演算と制御は主にCPU(中央演算処理装置)が担っています（厳密にいえば、チップセットが制御の一部を担ったり、GPUがグラフィックス処理を担当していたりします）。
  - 入力装置
  - 出力装置
  - 制御装置
  - 演算装置
  - 記憶装置

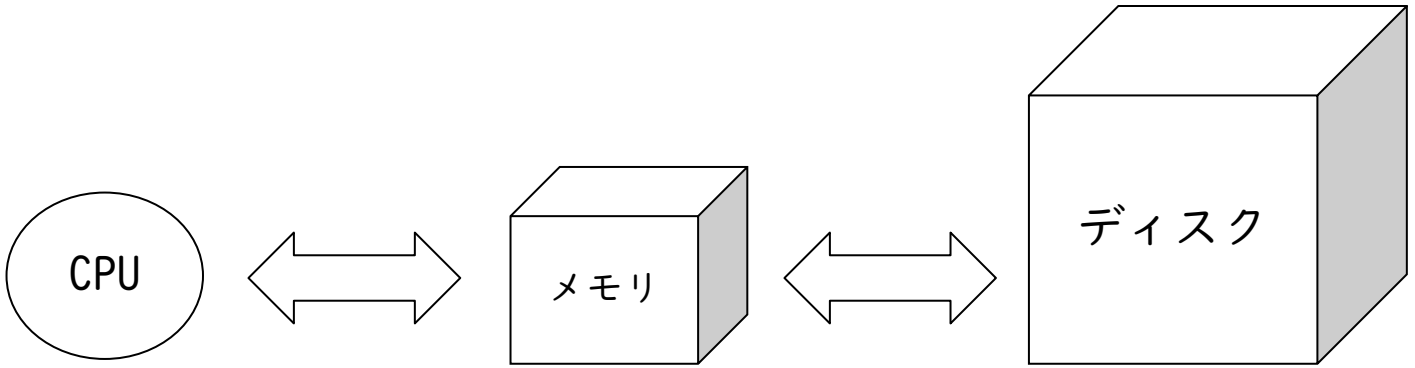


※ 高等学校情報科「情報 」教員研修用教材 JavaScript 版 p.4

Copyright © Asial Corporation. All Rights Reserved.

■ 記憶装置

- 記憶装置というとSDカードやハードディスクが思い浮かぶかもしれませんが、それらは二次記憶装置(補助記憶装置)に分類されます（ここではディスクと呼ぶことにします）。コンピュータの記憶装置には一次記憶装置（主記憶装置）であるメモリも欠かせません。
- 一次記憶装置の特性
  - 高速ですが容量は少なめです。また、データの保持に電気を必要とするものが大半です。
- 二次記憶装置の特性
  - メモリより低速ですが大容量です。また、電源がOFFでもデータを保持し続けられます。



Copyright © Asial Corporation. All Rights Reserved.

■ 補足：速いパソコンとは？

- パソコンは処理の最中にメモリが足りなくなった場合「強制終了して再起動」とはなりません。スワップという仕組みでメモリ不足を解決します。一時的にデータの一部をディスクに逃がして解決します。
- ディスクはメモリに比べると桁違いに低速なため、スワップが発生するとパソコンの動作が非常に重たくなります。そのため、速いパソコンを用意するためには、動かしたいOSや扱いたいソフトやデータに見合った量のメモリを搭載する必要があります。
- また、ソフトウェアの起動やファイルの読み書きではディスクを使いますので、ディスクにSSD(Solid State Drive)のような高速かつランダムアクセスに強いものを採用することで、高速化を実現できます。
- なお、プログラムを実際に演算処理するのはCPUの仕事のため、こちらも目的に見合った性能のCPUを用意する必要があります。
- 最近のCPUはグラフィックス処理を行うGPUを内蔵したものも増えていますが、高負荷なゲームを処理したい場合やGPUを活用した計算処理などを行いたい場合にはこちらも考慮する必要があります。

Copyright © Asial Corporation. All Rights Reserved.

論理演算

Copyright © Asial Corporation. All Rights Reserved.

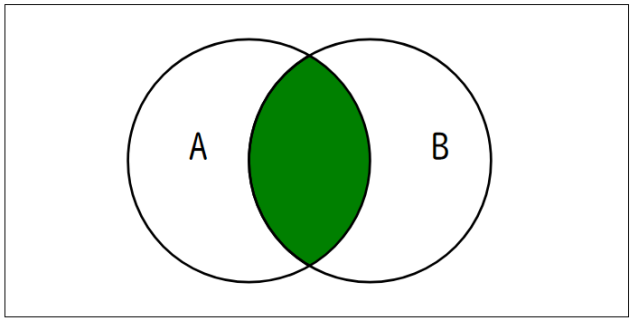
■ 論理回路と論理演算

- コンピュータの演算装置は2進数の値を扱う論理回路で設計されています。演算装置は論理演算を高速に処理できるため、それを応用して四則演算なども処理させています。
- 本項では、コンピュータの動作原理である論理演算の、特に基本となるANDとORとNOTの3つを学習します。
  - NANDやXORなどの他の論理演算は、AND・OR・NOTの組み合わせで実現できるため、この3つが最も基本となります。

Copyright © Asial Corporation. All Rights Reserved.

■ AND（論理積）

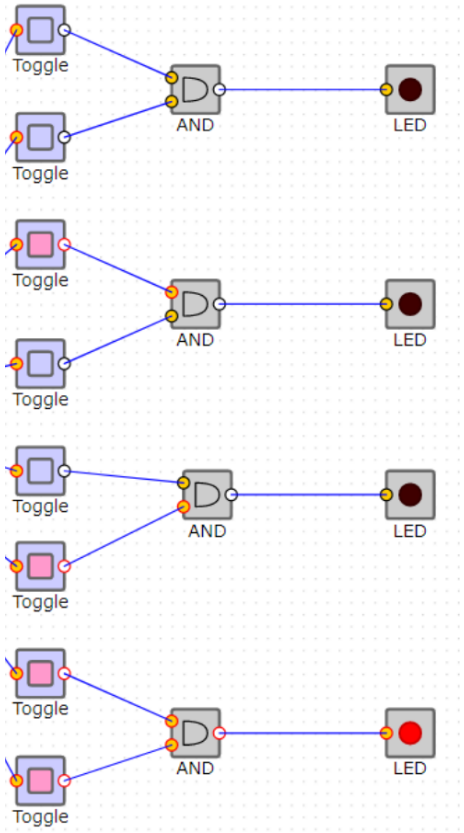
└ AとBの両方を含むときに結果が1(true)となる演算です。



ベン図

入力A	入力B	出力
0	0	0
0	1	0
1	0	0
1	1	1

真理値表

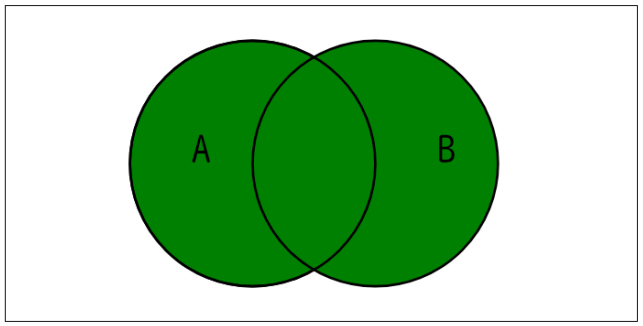


論理回路シミュレータの結果

Copyright © Asial Corporation. All Rights Reserved.

■ OR（論理和）

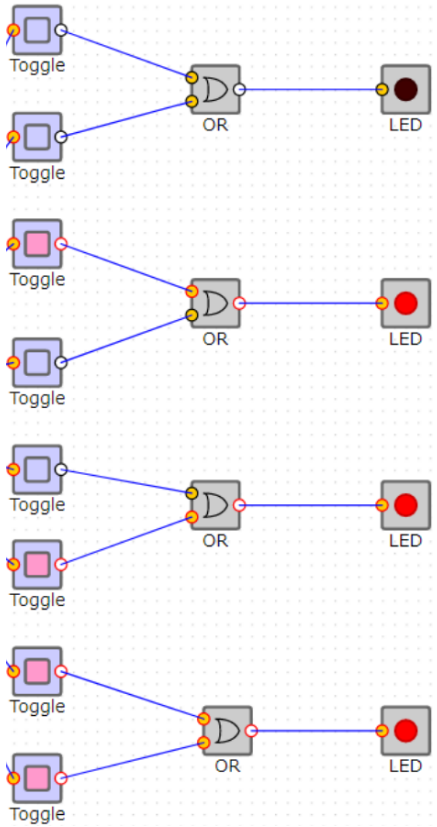
└ AとBのどちらかを含むときに結果が1(true)となる演算です。



ベン図

入力A	入力B	出力
0	0	0
0	1	1
1	0	1
1	1	1

真理値表

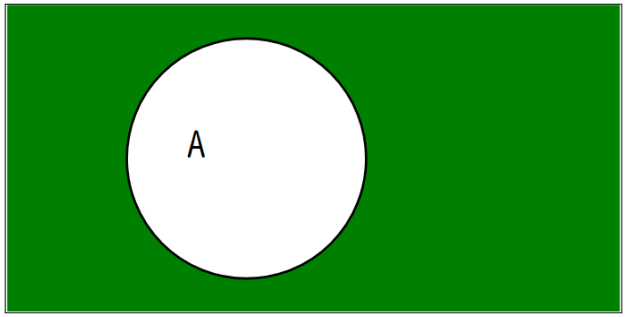


論理回路シミュレータの結果

Copyright © Asial Corporation. All Rights Reserved.

■ NOT（否定）

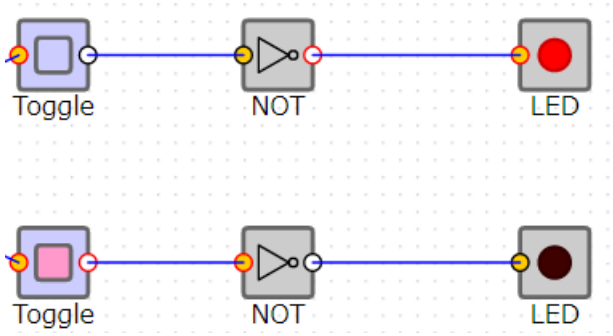
└ Aではないときに結果が1(true)となる演算です。



ベン図

入力A	出力
0	1
1	0

真理値表



論理回路シミュレータの結果

Copyright © Asial Corporation. All Rights Reserved.

論理演算を用いた二進数の足し算

Copyright © Asial Corporation. All Rights Reserved.

■ 論理演算を用いた二進数の足し算

└ 論理演算を応用することで四則演算も表すことができます、本項では論理演算で足し算に挑戦します。

■ 実現方法の検討

└ 1桁の二進数の値は0と1しかありませんので、1桁の二進数同士の足し算の結果は4通りに絞られます。表にすると以下の通りです。論理回路を組み合わせて以下を実現できれば二進数の足し算を表現できることとします。

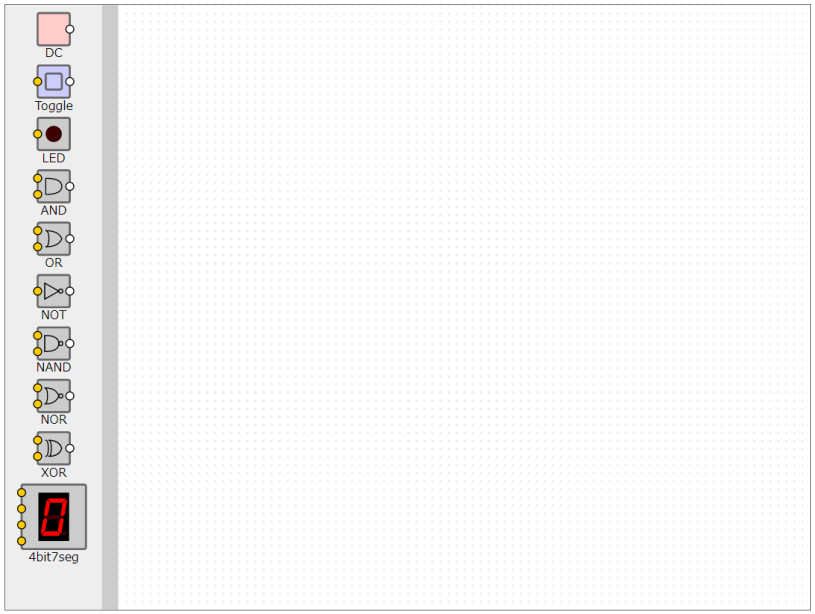
└ 入力が2つに対して出力も2つ必要なのがポイントです

入力A	入力B	出力C	出力F
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Copyright © Asial Corporation. All Rights Reserved.

■ 論理回路シミュレータによる実験

- ブラウザ上で論理回路の動きをシミュレーションできる「論理回路シミュレータ SimcirJS」で論理回路の動作を実験します
- オープンソース(MITライセンス)のため、改造や授業利用も可能です。
  - 情報Ⅰの学習に必要な部品を絞り込んだ改造版を以下に設置しております。

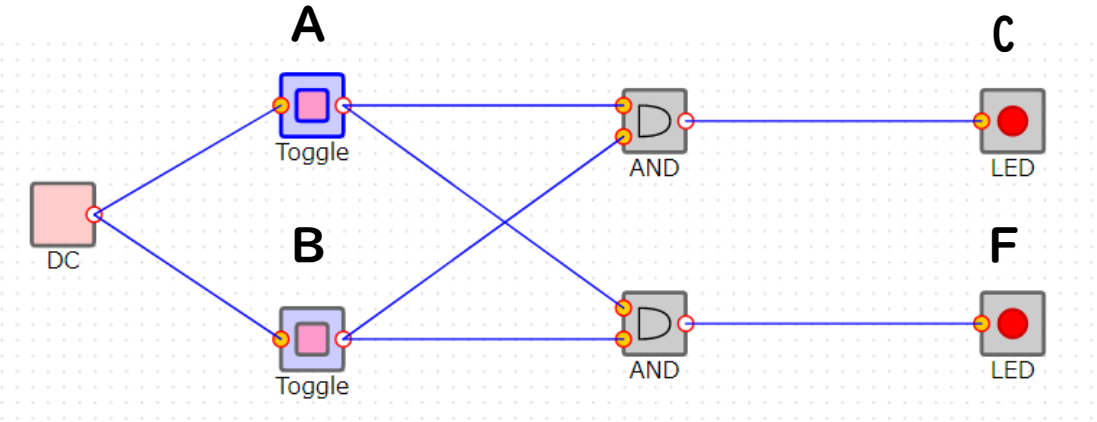


https://anko.education/tools/simcirjs/

Copyright © Asial Corporation. All Rights Reserved.

■ 2つの出力を用意する方法

└ 論理回路を以下のように接続すれば2つの入力に対して二つの出力を行うことが可能です。



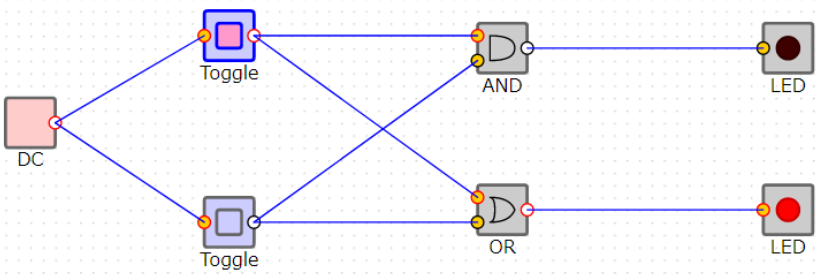
入力A	入力B	出力C	出力F
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

Copyright © Asial Corporation. All Rights Reserved.

■ ANDとORを組み合わせて挑戦

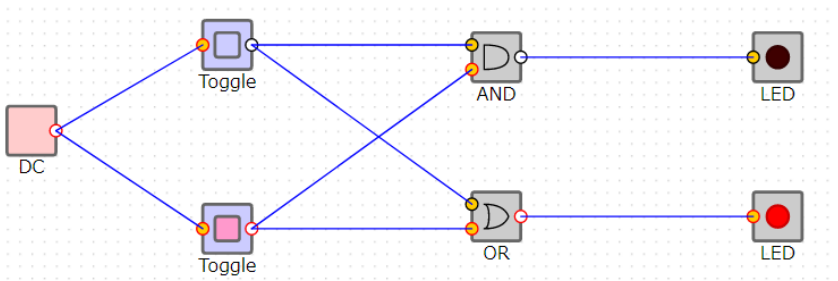
└ 片方をORに変えると惜しいところまでは実現できます。

└ 入力が11の時に出力を10にすることができません。



入力A	入力B	出力C	出力F
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

今回の真理値表



入力A	入力B	出力C	出力F
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

足し算の真理値表

Copyright © Asial Corporation. All Rights Reserved.

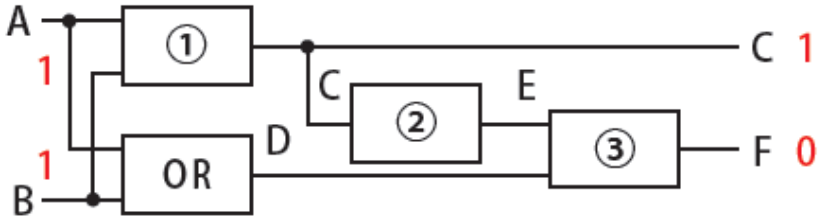
再検討

- 出力CはAND回路で求めるのが良さそう
- 出力FはOR回路だと01や10のときには出力が1となり期待通り、しかし11の時にも1が出てしまう
  - 出力FもAND回路が正解？
    - AND回路で01や10のときに01を出力するには？
- NOT回路の出番は？

Copyright © Asial Corporation. All Rights Reserved.

■ 研修資料における「二進数の足し算」の演習

- 以下の論理回路に論理演算を入れて二進数の足し算を成立させて下さい。
  - ①の回路はこれまでの内容で答えが出ています。
  - ①の出力結果(D)とOR回路の出力結果(E)を、さらに論理回路に入力しているのがポイントです。
  - ③に関しても、OR回路だとうまくいかないことは確認済みです。



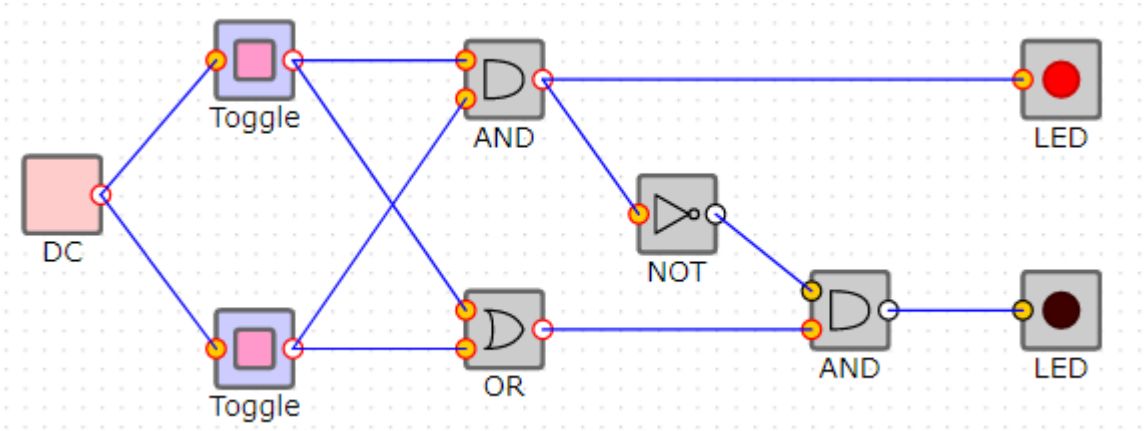
真理値表					
入力		途中経過		出力	
A	B	D	E	C	F
0	0	0	1	0	0
0	1	1	1	0	1
1	0	1	1	0	1
1	1	1	0	1	0

※ 高等学校情報科「情報 」教員研修用教材 JavaScript 版 p.6

Copyright © Asial Corporation. All Rights Reserved.

■ 「二進数の足し算」の答え

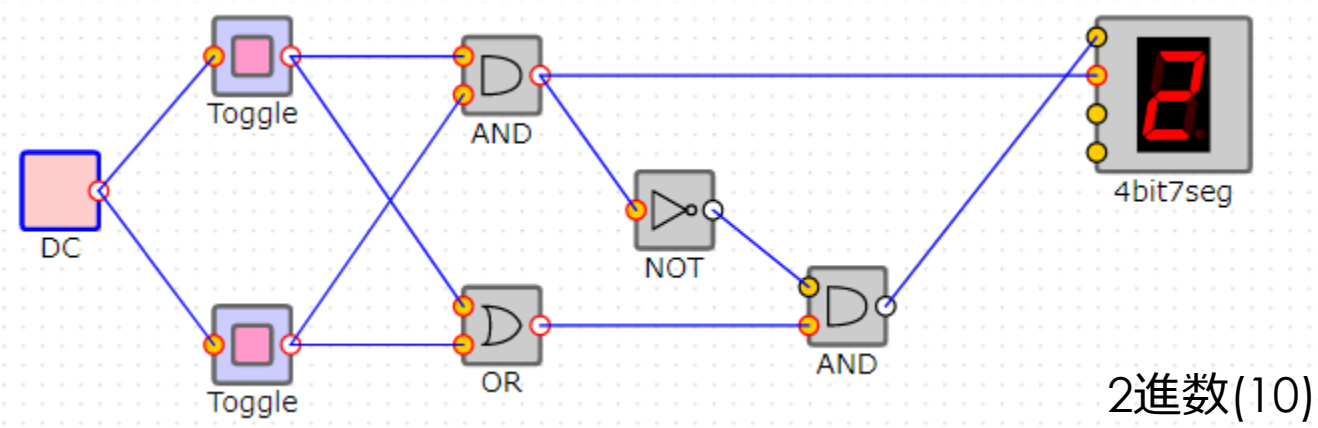
- ①の回路はANDです、入力が11のときだけ1を出力します
- ②の回路はNOTです、前段のANDの入力が11のときだけ0を出力します
- ③の回路はANDです
  - OR側は入力ABが10、01、11の時に1が入力されます
  - NOT側は入力ABが00、10、01のときに1が入力されます
  - つまり、このAND回路は入力ABが10と01のときに1を出力することとなります



Copyright © Asial Corporation. All Rights Reserved.

■ 「二進数の足し算」の結果を16進数で確認する

- 論理回路シミュレータの「7セグメントディスプレイ」を使うことで、2進数を16進数で表示できます
  - 0000～1111の範囲の2進数を入力して0～Fを出力できます



2進数(10)→16進数(2)

Copyright © Asial Corporation. All Rights Reserved.

計算誤差について

Copyright © Asial Corporation. All Rights Reserved.

■ 計算誤差について

- コンピュータは二進数を元に数値を扱っているため、小数点を含む数値を扱うと計算誤差が発生することがあります。
- 数値をCPUや処理系(プログラミング言語を処理するプログラム)で扱う際には、一定のビット幅のデータ型に格納して処理するため、それをあふれた値は失われます。

■ 数値を扱うデータ型

- JavaScriptでは整数も小数も全て「Number型」という64bitのデータ型が使用されます。他の言語におけるdouble型とほぼ同等の型です。

Copyright © Asial Corporation. All Rights Reserved.

■ Number型で表せる範囲

- 整数（安全に表せる範囲）
  - 9007199254740991（約9000兆）
    - Number.MAX_SAFE_INTEGER で確認可能
- 小数
  - 1.7976931348623157 x 10の308乗
    - 指数表記で「1.7976931348623157+e308」と表記します
  - ※ 誤差は発生しますが、整数よりも大きな桁数を表せます

Copyright © Asial Corporation. All Rights Reserved.

オーバーフロー・サンプル

```
// 17976931348623157がNumber型で表せる数値の桁的な上限
x = 1.7976931348623157e+308;
document.writeln(x+"<br>");

// e+308というのは10の308乗という意味
x = 1.7976931348623157 * 10 ** 308;
document.writeln(x+"<br>");

// 1.7976931348623157999は表せないので桁が切り捨てられる
x = 1.7976931348623157999e+308;
document.writeln(x+"<br>");

// 10の307乗に減らしても1.7976931348623157999は表せない
x = 1.7976931348623157999e+307;
document.writeln(x+"<br>");

// 10の309乗するとオ□バ□フロ□して無限になる
x = 1.7976931348623157e+309;
document.writeln(x+"<br>");

// 1.8 * 10の308乗もオ□バ□フロ□して無限になる
x = 1.8e+308;
document.writeln(x+"<br>");

// 1.8 * 10の307乗ならオ□バ□フロ□はしない
x = -1.8e+307;
document.writeln(x+"<br>");
```

実行結果

```
1.7976931348623157e+308

1.7976931348623157e+308

1.7976931348623157e+308

1.7976931348623158e+307

Infinity

Infinity

-1.8e+307
```

Copyright © Asial Corporation. All Rights Reserved.

計算誤差・サンプル

```
x = 28-27;
document.write(x+"<br>");

y = 0.28-0.27;
document.write(y+"<br>");

y = (28 - 27) / 100;
document.write(y+"<br>");
```

実行結果

```
1


0.010000000000000009


0.01
```

Copyright © Asial Corporation. All Rights Reserved.