

基礎コンピュータ工学 第1章 はじめに

<https://github.com/tctsigemura/TecTextBook>

本スライドの入手：

基礎コンピュータ工学第1章 はじめに 1 / 4

この科目で学ぶこと

この科目ではコンピュータの動作原理を学ぶ。

- 1946年にフォン・ノイマンが（Von Neumann）が発明
- ノイマン型コンピュータと呼ばれる。
- スーパーコンピュータからマイコンまで全てノイマン型。
(マイコン=マイクロコンピュータ：超小型コンピュータ)
- ノイマン型は発明されて70年以上が経過
- ノイマン型の時代は、まだ、しばらく続く

ノイマン型コンピュータの動作原理を学ぶことは、寿命の長いエンジニアになるために大切なステップ！

基礎コンピュータ工学第1章 はじめに

2 / 4

教材用コンピュータ

TeC (Tokuyama Educational Computer)：徳山高専教育用コンピュータ



- PCやスマホは巨大システム
- PCやスマホは動作原理を勉強するには難しそう
- TeCは動作原理を学ぶために特化し単純・小規模
- 学生が所有し、家でも演習ができる。

基礎コンピュータ工学第1章 はじめに 3 / 4

資料の電子データ入手

- 教科書のPDF（PCの場合）
<https://github.com/tctsigemura/TecTextBook> → [tec.pdf] 
- 教科書のPDF（スマホの場合） 
- スライドのPDF（PCの場合）
<https://github.com/tctsigemura/TecTextBook> → [Sld] → [chap1_Sld.pdf]
- スライドのPDF（スマホの場合） 
- TeCのホームページ（PCの場合）
<https://github.com/tctsigemura/TeC> 
- TeCのホームページ（スマホの場合） 

基礎コンピュータ工学第1章 はじめに

4 / 4

基礎コンピュータ工学 第2章 情報の表現 (パート1：ビット)

<https://github.com/tctsigemura/TecTextBook>

本スライドの入手：

基礎コンピュータ工学第2章 情報の表現 (パート1) / 12

情報の表現

コンピュータの内部で情報が表現されるか。
どのような回路で扱うことができるか。

コンピュータは電気で動くので情報も電気で表現する必要がある。

$$\text{情報の表現} = \begin{cases} \text{人：音声，文字，絵,...} \\ \text{コンピュータ：電圧，電流} \end{cases}$$

電気を用いた情報の表現（おおかみ情報）

電気の「ON/OFF」を用いて情報を表現する。

ランプ	意味
OFF	おおかみは来ていない
ON	おおかみが来た！！

基礎コンピュータ工学第2章 情報の表現 (パート1) / 3 / 12

おおかみ情報表示装置

牧場に次のような表示装置（カンパン）を設置する。

おおかみ情報表示装置

ランプ	意味	
OFF	来ていない	
ON	来た！！	

基礎コンピュータ工学第2章 情報の表現 (パート1) / 4 / 12

ビット

前例のような「二つのどちらか」を表す情報が「情報の最小単位」になる。情報の最小単位のことを「ビット (bit)」と呼ぶ。

on/off のどちらか → 情報の最小単位 (ビット)

ビットの値は「ON/OFF」ではなく、「1/0」で書く。

$$\begin{pmatrix} \text{ON} & : & 1 \\ \text{OFF} & : & 0 \end{pmatrix}$$

「おおかみが来た情報」をビットで表現する。

ビット値	意味
0 (off)	おおかみは来ていない
1 (on)	おおかみが来た！！

基礎コンピュータ工学第2章 情報の表現 (パート1) / 5 / 12

より複雑な情報の表現（拡張おおかみ情報）

複雑な情報は複数のランプ（ビット）の組み合わせで表現する。

ビット値	意味
00	おおかみはきていない（平気）
01	おおかみが1頭来た（戦う）
10	おおかみが2頭来た（？）
11	おおかみがたくさん来た（逃げる）

基礎コンピュータ工学第2章 情報の表現 (パート1) / 6 / 12

拡張おおかみ情報表示装置

牧場に次のような表示装置（カンパン）を設置すればよい！！



ビットの組合せと表現できる情報

拡張おおかみ情報は2ビットで4種類の情報を表現した。一般にはnビットで 2^n 種類の情報を表現できる。

ビット数	ビットの組合せ	組合せ数
1	0 1	$2^1 (= 2)$
2	00 01 10 11	$2^2 (= 4)$
3	000 001 010 011 100 101 110 111	$2^3 (= 8)$
...	...	2^n

「拡張おおかみ情報」のように、ビットの組合せに意味を持たせることで様々な情報を表現できる。

ビットの組合せの意味を表にして定義する。

ビット、ニブル、バイト

「ビット」は情報の最小単位

「ビット」は小さすぎるので「4ビット」、「8ビット」まとめたものもある。

名前	ビット数	組合せの数
ビット (bit)	1	$2^1 = 2$
ニブル (nibble)	4	$2^4 = 16$
バイト (byte)	8	$2^8 = 256$

スマホの容量：32GB, 64GB, 128GB（「B」はバイトの意味）

USBメモリの容量：32GB, 64GB, 128GB（「B」はバイトの意味）

通信速度制限：7GBを超えると制限される（「B」はバイトの意味）

通信速度：通常は100Mbps（「b」はビットの意味）

通信速度：制限されると128Mbps（「b」はビットの意味）

参考：bps：【bits per second / ビット毎秒】

数値の表現

これまで、ビットの組合せの意味決める。（表などにする）

ビットの組合せの意味をルールで決める場合もある。

コンピュータの内部では数値は2進数で表現する。

10進数

- 0～9の10種類の数字だけを使用する数値の表現方法。
- 一桁毎に10倍の重みを持つ

2進数

- 0, 1の2種類の数字だけを使用する数値の表現方法。
- 一桁毎に2倍の重みを持つ
- 0, 1の2種類の数字をビットの0, 1と対応付けやすい。
- nビット（桁）の2進数で0～ $2^n - 1$ までの値を表現できる。

4ビットの2進数

b_3	b_2	b_1	b_0	意味
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

宿題

宿題

1) 言葉の確認（ビット、ニブル、バイト）

2) nビットの組合せの数

3ビットで表現できる情報の種類は何種類か？

3種類の情報表現するためには何ビット必要か？

3) 0_{10} (0000_2) ~ 15_{10} (1111_2) の範囲を

2進数で数を数える練習をしなさい。

（小学校の1年生が10まで数える練習をするように）

**基礎コンピュータ工学
第2章 情報の表現
(パート2：2進数)**

<https://github.com/tctsigemura/TecTextBook>

本スライドの入手：

基礎コンピュータ工学第2章 情報の表現 (バー) 1 / 14

2進数から10進数への変換

2進数の桁ごとの重みは、桁の番号を n とすると 2^n になる。

$$\begin{array}{cccc} b_3 & b_2 & b_1 & b_0 \\ 2^3 = 8 & 2^2 = 4 & 2^1 = 2 & 2^0 = 1 \end{array}$$

2進数の数値は、その桁の重みと桁の値を掛け合わせたものの合計。例えば2進数の 1010_2 は、 2^3 の桁が1, 2^2 の桁が0, 2^1 の桁が1, 2^0 の桁が0ですから、次のように計算できる。

$$\begin{aligned} 1010_2 &= 2^3 \times 1 + 2^2 \times 0 + 2^1 \times 1 + 2^0 \times 0 \\ &= 8 \times 1 + 4 \times 0 + 2 \times 1 + 1 \times 0 \\ &= 8 + 0 + 2 + 0 \\ &= 10_{10} \end{aligned}$$

2進数から10進数への変換（問題）

問題1：次の2進数を10進数に変換しなさい。

- 1) 00011100_2
- 2) 00111000_2
- 3) 11100000_2

基礎コンピュータ工学第2章 情報の表現 (バー) 3 / 14

10進数から2進数への変換

2進数を2で割ると右に1桁移動する。

(10進数は10で割ると右に1桁移動した。)
その時の余りは最下位の桁からはみ出した数になる。
同じ値の10進数を2で割っても余りは同じ。

$$\begin{array}{rcl} 10_{10} &=& 1010_2 \\ \div 2 \downarrow && \\ 5_{10}^{..0} &=& 0101_2^{..0} \\ \div 2 \downarrow && \\ 2_{10}^{..1} &=& 0010_2^{..1} \\ \div 2 \downarrow && \\ 1_{10}^{..0} &=& 0001_2^{..0} \\ \div 2 \downarrow && \\ 0_{10}^{..1} &=& 0000_2^{..1} \end{array}$$

10進数から2進数への変換

2で割る操作を繰り返しながらはみ出して来た数を記録する。
右から並べると2進数で表したときの0/1の並びが分かる。

$$\begin{array}{r} 2) \overline{10} \\ 2) \overline{5 \cdots 0} \\ 2) \overline{2 \cdots 1} \quad \text{余りを右から順に並べると} \\ 2) \overline{1 \cdots 0} \quad 1010_2 \\ 0 \cdots 1 \end{array}$$

基礎コンピュータ工学第2章 情報の表現 (バー) 5 / 14

10進数から2進数への変換（問題）

問題2：次の10進数を8桁の2進数に変換しなさい。

- 1) 16_{10}
- 2) 50_{10}
- 3) 100_{10}
- 4) 127_{10}
- 5) 130_{10}

16進数

- 2進数4桁を16進数1桁で書く。
- 16種類の数字が必要、
- AからFを数字の代用にする。
- 2進数の書き方
 - 01100100₂
 - 01100100_b
- 16進数の書き方
 - 64₁₆
 - 64H
- 右の表は暗記すること。

2進数	16進数	10進数
0000 ₂	0 ₁₆	0 ₁₀
0001 ₂	1 ₁₆	1 ₁₀
0010 ₂	2 ₁₆	2 ₁₀
0011 ₂	3 ₁₆	3 ₁₀
0100 ₂	4 ₁₆	4 ₁₀
0101 ₂	5 ₁₆	5 ₁₀
0110 ₂	6 ₁₆	6 ₁₀
0111 ₂	7 ₁₆	7 ₁₀
1000 ₂	8 ₁₆	8 ₁₀
1001 ₂	9 ₁₆	9 ₁₀
1010 ₂	A ₁₆	10 ₁₀
1011 ₂	B ₁₆	11 ₁₀
1100 ₂	C ₁₆	12 ₁₀
1101 ₂	D ₁₆	13 ₁₀
1110 ₂	E ₁₆	14 ₁₀
1111 ₂	F ₁₆	15 ₁₀

16進数のFFまで数えてみよう

00 10 20 30 40 50 60 70 80 90 A0 B0 C0 D0 E0 F0
 01 11 21 31 41 51 61 71 81 91 A1 B1 C1 D1 E1 F1
 02 12 22 32 42 52 62 72 82 92 A2 B2 C2 D2 E2 F2
 03 13 23 33 43 53 63 73 83 93 A3 B3 C3 D3 E3 F3
 04 14 24 34 44 54 64 74 84 94 A4 B4 C4 D4 E4 F4
 05 15 25 35 45 55 65 75 85 95 A5 B5 C5 D5 E5 F5
 06 16 26 36 46 56 66 76 86 96 A6 B6 C6 D6 E6 F6
 07 17 27 37 47 57 67 77 87 97 A7 B7 C7 D7 E7 F7
 08 18 28 38 48 58 68 78 88 98 A8 B8 C8 D8 E8 F8
 09 19 29 39 49 59 69 79 89 99 A9 B9 C9 D9 E9 F9
 0A 1A 2A 3A 4A 5A 6A 7A 8A 9A AA BA CA DA EA FA
 0B 1B 2B 3B 4B 5B 6B 7B 8B 9B AB BB CB DB EB FB
 0C 1C 2C 3C 4C 5C 6C 7C 8C 9C AC BC CC DC EC FC
 0D 1D 2D 3D 4D 5D 6D 7D 8D 9D AD BD CD DD ED FD
 0E 1E 2E 3E 4E 5E 6E 7E 8E 9E AE BE CE DE EE FE
 0F 1F 2F 3F 4F 5F 6F 7F 8F 9F AF BF CF DF EF FF

16進数との変換

- 2進数 <=> 16進数
4桁の2進数と1桁の16進数の対応は暗記する。
- 10進数 <=> 16進数
 - 10進数 <=> 2進数 <=> 16進数
一度、2進数に変換してから変換する。
 $100_{10} = 01100100_2 = 64_{16}$
 - 直接計算する
桁の重みは16倍になっていく。

$$\begin{array}{ccccccc} h_3 & & h_2 & & h_1 & & h_0 \\ 16^3 = 4096 & \quad 16^2 = 256 & \quad 16^1 = 16 & \quad 16^0 = 1 \\ 2^{12} = 4096 & \quad 2^8 = 256 & \quad 2^4 = 16 & \quad 2^0 = 1 \end{array}$$

16進数との変換

10進数 => 16進数

$$\begin{array}{r} 16) \quad 100 \\ \hline 16) \quad 6 \cdots 4 \quad \text{余りを右から順に並べると } 64_{16} \\ \hline 0 \cdots 6 \end{array}$$

16進数 => 10進数

16進数の数値は、その桁の重みと桁の値を掛け合わせたものの合計。

$$\begin{aligned} 64_{16} &= 16^1 \times 6 + 16^0 \times 4 \\ &= 16 \times 6 + 1 \times 4 \\ &= 96 + 4 \\ &= 100_{10} \end{aligned}$$

16進数（問題1/4）

問題3 : 0016 から FF₁₆ まで、声に出して数えなさい。
 問題4 : 次の2進数を2桁の16進数に変換しなさい。

- 1) 00011100₂
- 2) 00111000₂
- 3) 11100000₂
- 4) 01110101₂

16進数（問題2/4）

問題5 : 次の16進数を8桁の2進数に変換しなさい。

- 1) 11₁₆
- 2) 56₁₆
- 3) AB₁₆
- 4) CD₁₆
- 5) 3C₁₆

16進数（問題3/4）

問題6：次の16進数を10進数に変換しなさい。

1) 11_{16}

2) 56_{16}

3) AB_{16}

4) CD_{16}

5) $3C_{16}$

16進数（問題4/4）

問題7：次の10進数を2桁の16進数に変換しなさい。

1) 16_{10}

2) 50_{10}

3) 100_{10}

4) 127_{10}

5) 130_{10}

**基礎コンピュータ工学
第2章 情報の表現
(パート3：2進数の計算と2の補数)**

<https://github.com/tctsigemura/TecTextBook>

本スライドの入手：

基礎コンピュータ工学第2章 情報の表現 (パート3)
1 / 16

2進数の和差の計算

10進数の場合を思い出してみる。

- 9より大きくなる時に桁上げが発生する。

$$\begin{array}{r} 103 \\ + 104 \\ \hline 207 \end{array} \quad \begin{array}{r} 105 \\ + 107 \\ \hline 212 \end{array} \quad \begin{array}{r} 135 \\ + 127 \\ \hline 262 \end{array} \quad \begin{array}{r} 155 \\ + 167 \\ \hline 322 \end{array} \quad \begin{array}{r} 099 \\ + 001 \\ \hline 100 \end{array}$$

- 桁借りでは10借りてくる。

$$\begin{array}{r} 207 \\ - 104 \\ \hline 103 \end{array} \quad \begin{array}{r} 212 \\ - 107 \\ \hline 105 \end{array} \quad \begin{array}{r} 262 \\ - 127 \\ \hline 135 \end{array} \quad \begin{array}{r} 322 \\ - 167 \\ \hline 155 \end{array} \quad \begin{array}{r} 100 \\ - 001 \\ \hline 099 \end{array}$$

基礎コンピュータ工学第2章 情報の表現 (パート3)
2 / 16

2進数の和差の計算

2進数の場合は以下のようになる。

- 1より大きくなる時に桁上げが発生する。

$$\begin{array}{r} 010 \\ + 001 \\ \hline 011 \end{array} \quad \begin{array}{r} 001 \\ + 011 \\ \hline 010 \end{array} \quad \begin{array}{r} 010 \\ + 011 \\ \hline 101 \end{array} \quad \begin{array}{r} 011 \\ + 001 \\ \hline 100 \end{array} \quad \begin{array}{r} 011 \\ + 011 \\ \hline 110 \end{array}$$

- 桁借りでは2借りてくる。

$$\begin{array}{r} 011 \\ - 001 \\ \hline 010 \end{array} \quad \begin{array}{r} 010 \\ - 001 \\ \hline 001 \end{array} \quad \begin{array}{r} 101 \\ - 011 \\ \hline 010 \end{array} \quad \begin{array}{r} 100 \\ - 001 \\ \hline 011 \end{array} \quad \begin{array}{r} 110 \\ - 011 \\ \hline 011 \end{array}$$

基礎コンピュータ工学第2章 情報の表現 (パート3)
3 / 16

2進数の和差の計算 (問題)

問題8：10進数の計算と2進数の計算をしなさい。

$\begin{array}{r} 3+8 \\ \hline 10\text{進} & 2\text{進} \\ 3 & 0011 \\ + 8 & + 1000 \\ \hline \end{array}$	$\begin{array}{r} 5+7 \\ \hline 10\text{進} & 2\text{進} \\ 5 & 0101 \\ + 7 & + 0111 \\ \hline \end{array}$
$\begin{array}{r} 11-8 \\ \hline 10\text{進} & 2\text{進} \\ 11 & 1011 \\ - 8 & - 1000 \\ \hline \end{array}$	$\begin{array}{r} 12-7 \\ \hline 10\text{進} & 2\text{進} \\ 12 & 1100 \\ - 7 & - 0111 \\ \hline \end{array}$

基礎コンピュータ工学第2章 情報の表現 (パート3)
4 / 16

負数の表現

負の数を2進数でどのようにビットで表現するか約束する。

(1) 符号付き絶対値表現
左端のビットを符号 (+ / -) として使用する。

4ビット符号付き絶対値表現の例

負数	2進数	正数	2進数
-7	1111 ₂	+7	0111 ₂
-6	1110 ₂	+6	0110 ₂
-5	1101 ₂	+5	0101 ₂
...
-1	1001 ₂	+1	0001 ₂
-0	1000 ₂	+0	0000 ₂

• 4ビットで-7から+7の範囲を表現できる。
• 0の表現が二つある(-0と+0)。

基礎コンピュータ工学第2章 情報の表現 (パート3)
5 / 16

負数の表現

補数表現

- n 桁の**b**進数において
 b^n から x を引いた数 y を x に対する「 b の補数」と呼ぶ。
 $y = b^n - x$ (y は x に対する b の補数)
- n 桁の**b**進数において
 $b^n - 1$ から x を引いた数 z を x に対する「 $(b-1)$ の補数」と呼ぶ。
 $z = b^n - 1 - x$ (z は x に対する $(b-1)$ の補数)

基礎コンピュータ工学第2章 情報の表現 (パート3)
6 / 16

負数の表現

2桁の10進数における補数の例

$$\begin{array}{r} b = 10\text{進数} \\ n = 2\text{桁} \\ b^n = 100 \\ x = 25 \end{array} \quad \begin{array}{r} 100 \\ -25 \\ \hline 75 \end{array} \quad 75 \text{は } 25 \text{に対する } 10 \text{の補数}$$

$$\begin{array}{r} b = 10\text{進数} \\ n = 2\text{桁} \\ b^n - 1 = 99 \\ x = 25 \end{array} \quad \begin{array}{r} 99 \\ -25 \\ \hline 74 \end{array} \quad 74 \text{は } 25 \text{に対する } 9 \text{の補数}$$

負数の表現

4桁の2進数における補数の例

$$\begin{array}{r} b = 2\text{進数} \\ n = 4\text{桁} \\ b^n = 10000_2 \\ x = 1010_2 \end{array} \quad \begin{array}{r} 10000_2 \\ -1010_2 \\ \hline 0110_2 \end{array} \quad 0110_2 \text{は } 1010_2 \text{に対する } 2 \text{の補数}$$

$$\begin{array}{r} b = 2\text{進数} \\ n = 4\text{桁} \\ b^n - 1 = 1111_2 \\ x = 1010_2 \end{array} \quad \begin{array}{r} 1111_2 \\ -1010_2 \\ \hline 0101_2 \end{array} \quad 0101_2 \text{は } 1010_2 \text{に対する } 1 \text{の補数}$$

負数の表現

(2) 1の補数による負数の表現

1の補数を負数の表現に使用する。

4ビット2進数の1の補数 ($2^4 - 1 - x = z$)

もとの数 (x)	補数へ変換	補数 (z)
0	$1111_2 - 0000_2 =$	1111_2
1	$1111_2 - 0001_2 =$	1110_2
2	$1111_2 - 0010_2 =$	1101_2
3	$1111_2 - 0011_2 =$	1100_2
4	$1111_2 - 0100_2 =$	1011_2
5	$1111_2 - 0101_2 =$	1010_2
6	$1111_2 - 0110_2 =$	1001_2
7	$1111_2 - 0111_2 =$	1000_2

負数の表現

1の補数を用いた符号付き数値

-7	1000_2	-	-	-	-	-	-	-	+
-6	1001_2	-	-	-	-	-	-	-	+
-5	1010_2	-	-	-	-	-	-	+	
-4	1011_2	-	-	-	-	-	+		
-3	1100_2	-	-	-	-	+			
-2	1101_2	-	-	-	+				
-1	1110_2	-	+						
-0	1111_2	+							
+0	0000_2	+							
+1	0001_2	-	+						
+2	0010_2	-	-	+					
+3	0011_2	-	-	-	+				
+4	0100_2	-	-	-	-	+			
+5	0101_2	-	-	-	-	-	+		
+6	0110_2	-	-	-	-	-	-	+	
+7	0111_2	-	-	-	-	-	-	-	+

負数の表現

1の補数の求め方

ビット反転

$$x = +3_{10} = 0011_2 \text{ (もとの数)}$$

$$y = -3_{10} = 1100_2 \text{ (1の補数)}$$

表現できる数値の範囲

$$4\text{ビット}: -7 \sim +7 \quad (-(2^3 - 1) \sim +(2^3 - 1))$$

$$n\text{ビット}: -(2^{n-1} - 1) \sim +(2^{n-1} - 1)$$

正負の判定

最上位ビットが

- 0 : 正の値を表現している。
- 1 : 負の値を表現している。

負数の表現

(3) 2の補数による負数の表現

2の補数 ($2^n - x$) を負数の表現に使用する。

4ビット2進数の2の補数 ($2^4 - x = y$)

もとの数 (x)	補数へ変換	補数 (y)
0	$10000_2 - 0000_2 =$	10000_2
1	$10000_2 - 0001_2 =$	1111_2
2	$10000_2 - 0010_2 =$	1110_2
3	$10000_2 - 0011_2 =$	1101_2
4	$10000_2 - 0100_2 =$	1100_2
5	$10000_2 - 0101_2 =$	1011_2
6	$10000_2 - 0110_2 =$	1010_2
7	$10000_2 - 0111_2 =$	1001_2
8	$10000_2 - 1000_2 =$	1000_2

負数の表現

2の補数を用いた符号付き数値

-8	1000 ₂	-	-	-	-	-	-	-	+
-7	1001 ₂	-	-	-	-	-	-	-	
-6	1010 ₂	-	-	-	-	-	-	+	
-5	1011 ₂	-	-	-	-	-	+		
-4	1100 ₂	-	-	-	-	+			
-3	1101 ₂	-	-	-	+				
-2	1110 ₂	-	-	+					
-1	1111 ₂	-	+						
0	0000 ₂	+	-	-	-	-	-	-	
1	0001 ₂	-	+	-	-	-	-	-	
2	0010 ₂	-	-	+	-	-	-	-	
3	0011 ₂	-	-	-	+	-	-	-	
4	0100 ₂	-	-	-	-	+	-	-	
5	0101 ₂	-	-	-	-	-	+	-	
6	0110 ₂	-	-	-	-	-	-	+	
7	0111 ₂	-	-	-	-	-	-	-	+

負数の表現

- 2の補数の求め方

ビット反転+1

$$x = +3_{10} = 0011_2 \text{ (との数)}$$

$$y = -3_{10} = 1100_2 + 1 = 1101_2 \text{ (2の補数)}$$

元に戻すのもビット反転+1

$$y = -3_{10} = 1101_2 \text{ (2の補数)}$$

$$y = +3_{10} = 0010_2 + 1 = 0011_2 \text{ (との数)}$$

- 表現できる数値の範囲

$$4ビット : -8 \sim +7 (-2^3 \sim + (2^3 - 1))$$

$$nビット : -2^{n-1} \sim + (2^{n-1} - 1)$$

- 正負の判定

最上位ビットが

0 : 正の値を表現している。

1 : 負の値を表現している。

負数の表現（問題1/2）

問題9：次の10進数を2の補数表現形式の4桁の2進数に変換しなさい。

1) 4_{10}

2) -4_{10}

3) 5_{10}

4) -5_{10}

5) 6_{10}

6) -6_{10}

負数の表現（問題2/2）

問題10：次の2の補数表現形式の4桁の2進数を10進数に変換しなさい。

1) 1001_2

2) 0111_2

3) 1101_2

4) 0011_2

5) 1011_2

6) 1100_2

**基礎コンピュータ工学
第2章 情報の表現
(パート4：2の補数の和差)**

<https://github.com/tctsigemura/TecTextBook>

本スライドの入手：

基礎コンピュータ工学第2章 情報の表現 (バー) 1 / 9

2進数の和差の計算 (復習)

2進数の場合は以下のようになる。

- 1より大きくなる時に桁上げが発生する。

$$\begin{array}{r} 010 \\ + 001 \\ \hline 011 \end{array} \quad \begin{array}{r} 001 \\ + 001 \\ \hline 010 \end{array} \quad \begin{array}{r} 010 \\ + 011 \\ \hline 101 \end{array} \quad \begin{array}{r} 011 \\ + 001 \\ \hline 100 \end{array} \quad \begin{array}{r} 011 \\ + 011 \\ \hline 110 \end{array}$$

- 桁借りでは2借りてくる。

$$\begin{array}{r} 011 \\ - 001 \\ \hline 010 \end{array} \quad \begin{array}{r} 010 \\ - 001 \\ \hline 001 \end{array} \quad \begin{array}{r} 101 \\ - 011 \\ \hline 010 \end{array} \quad \begin{array}{r} 100 \\ - 001 \\ \hline 011 \end{array} \quad \begin{array}{r} 110 \\ - 011 \\ \hline 011 \end{array}$$

2進数の和差の計算 (復習)

10進数の計算と2進数の計算をしなさい。

$\begin{array}{r} 3+8 \\ \hline \begin{array}{l} 10\text{進} \\ 3 \\ + 8 \\ \hline \end{array} \quad \begin{array}{l} 2\text{進} \\ 0011 \\ + 1000 \\ \hline \end{array} \end{array}$	$\begin{array}{r} 5+7 \\ \hline \begin{array}{l} 10\text{進} \\ 5 \\ + 7 \\ \hline \end{array} \quad \begin{array}{l} 2\text{進} \\ 0101 \\ + 0111 \\ \hline \end{array} \end{array}$
$\begin{array}{r} 11-8 \\ \hline \begin{array}{l} 10\text{進} \\ 11 \\ - 8 \\ \hline \end{array} \quad \begin{array}{l} 2\text{進} \\ 1011 \\ - 1000 \\ \hline \end{array} \end{array}$	$\begin{array}{r} 12-7 \\ \hline \begin{array}{l} 10\text{進} \\ 12 \\ - 7 \\ \hline \end{array} \quad \begin{array}{l} 2\text{進} \\ 1100 \\ - 0111 \\ \hline \end{array} \end{array}$

基礎コンピュータ工学第2章 情報の表現 (バー) 3 / 9

負数の表現 (復習)

- 2の補数による負数の表現

2の補数 ($2^n - x$) を負数の表現に使用する。

4ビット2進数の2の補数 ($2^4 - x = y$)

もとの数(x)	補数へ変換	補数(y)
0	$10000_2 - 0000_2 = 10000_2$	10000_2
1	$10000_2 - 0001_2 = 1111_2$	1111_2
2	$10000_2 - 0010_2 = 1110_2$	1110_2
3	$10000_2 - 0011_2 = 1101_2$	1101_2
4	$10000_2 - 0100_2 = 1100_2$	1100_2
5	$10000_2 - 0101_2 = 1011_2$	1011_2
6	$10000_2 - 0110_2 = 1010_2$	1010_2
7	$10000_2 - 0111_2 = 1001_2$	1001_2
8	$10000_2 - 1000_2 = 1000_2$	1000_2

基礎コンピュータ工学第2章 情報の表現 (バー) 4 / 9

負数の表現 (復習)

- 2の補数の求め方

ビット反転+1

$$x = +3_{10} = 0011_2 \text{ (もとの数)}$$

$$y = -3_{10} = 1100_2 + 1 = 1101_2 \text{ (2の補数)}$$

元に戻すのもビット反転+1

$$y = -3_{10} = 1101_2 \text{ (2の補数)}$$

$$y = +3_{10} = 0010_2 + 1 = 0011_2 \text{ (もとの数)}$$

- 表現できる数値の範囲

$$4\text{ビット} : -8 \sim +7 (-2^3 \sim + (2^3 - 1))$$

$$n\text{ビット} : -2^{n-1} \sim + (2^{n-1} - 1)$$

- 正負の判定

最上位ビットが

0 : 正の値を表現している。

1 : 負の値を表現している。

基礎コンピュータ工学第2章 情報の表現 (バー) 5 / 9

負の数を含む計算

2の補数表現の負数は符号無し2進数と同じ手順で計算できる！！

- 最上位ビットからの桁上げは無視する。

$$\begin{array}{r} 0010 \quad (+2) \quad 1011 \quad (-5) \quad 1101 \quad (-3) \\ + 1111 \quad (-1) \quad + 0101 \quad (+5) \quad + 1101 \quad (-3) \\ \hline 0001 \quad (+1) \quad 0000 \quad (+0) \quad 1010 \quad (-6) \end{array}$$

- 仕組み

正の数と負の数の和 (-bを2の補数($2^n - b$)と表現する)

正の値aと負の値-bの和を計算し 2^n (最上位の桁上げ)を無視する

$$a + (-b) = a + (2^n - b) = 2^n + a - b = a - b$$

負の数と負の数の和 (-a, -bを2の補数で表現する)

$$2^n \text{ (最上位からの桁上げ)} を一つ無視すると$$

$$(-a) + (-b) = (2^n - a) + (2^n - b) = 2^n - (a + b)$$

負の数を含む計算

2の補数表現の負数は符号無し2進数と同じ手順で計算できる！！

- 最上位ビットの桁借りは制限なしとする。

$$\begin{array}{r} 0010 \quad (+2) & 0000 \quad (+0) & 1101 \quad (-3) \\ - 1111 \quad (-1) & - 0101 \quad (+5) & - 1010 \quad (-6) \\ \hline 0011 \quad (+3) & 1011 \quad (-5) & 0011 \quad (+3) \end{array}$$

- 仕組み

- 正の数と負の数の差 ($-b$ を2の補数 $(2^n - b)$ と表現する)
正の値 a と負の値 $-b$ の差を計算し -2^n (最上位の桁借り) を許す

$$a - (-b) = a - (2^n - b) = -2^n + a + b = a + b$$
- 負の数と負の数の差 ($-a$, $-b$ を2の補数で表現する)
 2^n (最上位からの桁上げ) を一つ無視すると

$$(-a) - (-b) = (2^n - a) - (2^n - b) = (-a) + b$$

負数を含む計算 (問題1/2)

問題1 1：次の計算を2進数と10進数でしなさい。
(ただし、2進数は2の補数表現形式になっている)

1) $0011\ 0010_2$
 $+ 0011\ 0010_2$ → $\begin{array}{r} \boxed{} \\ \boxed{} \\ \hline \boxed{b} \end{array}$

2) $1111\ 1111_2$
 $+ 1111\ 1111_2$ → $\begin{array}{r} \boxed{} \\ \boxed{} \\ \hline \boxed{b} \end{array}$

負数を含む計算 (問題2/2)

3) $0110\ 0100_2$
 $+ 1001\ 1100_2$ → $\begin{array}{r} \boxed{} \\ \boxed{} \\ \hline \boxed{b} \end{array}$

4) $1111\ 0000_2$
 $+ 1110\ 1111_2$ → $\begin{array}{r} \boxed{} \\ \boxed{} \\ \hline \boxed{b} \end{array}$

5) $0001\ 0000_2$
 $- 1110\ 1111_2$ → $\begin{array}{r} \boxed{} \\ \boxed{} \\ \hline \boxed{b} \end{array}$

**基礎コンピュータ工学
第2章 情報の表現
(パート5 : 小数や文字の表現と補助単位)**

<https://github.com/tctsigemura/TecTextBook>

本スライドの入手：

基礎コンピュータ工学第2章 情報の表現 /パート5 : 小数や文字の表現と補助単位

2進数による小数の表現

固定小数点方式（小数点の位置を約束する。）

00.00 ₂ = 0.0 ₁₀	桁の重み
00.01 ₂ = 0.25 ₁₀	● 小数点から左に進むと 2倍
00.10 ₂ = 0.5 ₁₀	001.0000 ₂ = 1.0 ₁₀
00.11 ₂ = 0.75 ₁₀	010.0000 ₂ = 2.0 ₁₀
01.00 ₂ = 1.0 ₁₀	100.0000 ₂ = 4.0 ₁₀
01.01 ₂ = 1.25 ₁₀	
01.10 ₂ = 1.5 ₁₀	● 小数点から右に進むと 1/2倍
01.11 ₂ = 1.75 ₁₀	000.1000 ₂ = 0.5 ₁₀
10.00 ₂ = 2.0 ₁₀	000.0100 ₂ = 0.25 ₁₀
...	000.0010 ₂ = 0.125 ₁₀
11.11 ₂ = 3.75 ₁₀	000.0001 ₂ = 0.0625 ₁₀

基礎コンピュータ工学第2章 情報の表現 /パート5 : 小数や文字の表現と補助単位

2 / 10

固定小数点方式 2進数 → 10進数

桁の重みを合計する。

$$\begin{aligned} 10.01_2 &= 1 \times 2 + 0 \times 1 + 0 \times 1/2 + 1 \times 1/4 \\ &= 2 + 0 + 0 + 1/4 \\ &= 2 + 0.25 \\ &= 2.25 \end{aligned}$$

問題1 2 : 2進数を10進数に変換しなさい。

- 1) 0101.1010₂
- 2) 0011.0011₂
- 3) 0100.0101₂
- 4) 1010.1111₂

基礎コンピュータ工学第2章 情報の表現 /パート5 : 小数や文字の表現と補助単位

3 / 10

10進数 → 固定小数点方式 2進数

2進数	$\times 2$ は	10進数
0.101 ₂	左シフトと同じ	0.625
\swarrow		$\times \frac{2}{2}$
1.010 ₂		1.250

2進数	$\times 2$ は	10進数
0.010 ₂	左シフトと同じ	0.250
\swarrow		$\times \frac{2}{2}$
0.100 ₂		0.500

2進数	$\times 2$ は	10進数
0.100 ₂	左シフトと同じ	0.500
\swarrow		$\times \frac{2}{2}$
1.000 ₂		1.000

10進数で計算したとき、小数点を横切って整数部に出てきた数を小数点の右に順番に並べると 0.101₂ になる。

基礎コンピュータ工学第2章 情報の表現 /パート5 : 小数や文字の表現と補助単位

4 / 10

固定小数点方式 2進数 → 10進数

問題1 3 : 10進数を2進数に変換しなさい。
但し2進数は、小数点以下4桁、全体で6桁とする。

- 1) 0.75₁₀
- 2) 0.5625₁₀
- 3) 2.5₁₀
- 4) 1.1875₁₀

基礎コンピュータ工学第2章 情報の表現 /パート5 : 小数や文字の表現と補助単位

5 / 10

文字の表現 (ASCII コード)

ASCII コード (American Standard Code for Information Interchange)
1963年にアメリカ規格協会(ANSI)が定めた情報交換用の文字コード。

(上位3ビット)

	0	1	2	3	4	5	6	7
O	NUL	DLE (sp)	O	Q	P	~	p	
1	SOH	DCL !	1	A	Q	a	q	
2	STX	DCL "	2	B	R	b	r	
3	ETX	DCL #	3	C	S	c	s	
4	EDT	DCL \$	4	D	T	d	t	
5	ENQ	NAB %	5	E	U	e	u	
6	ACK	SYN &	6	F	V	f	v	
7	BEL	ETB '	7	G	W	g	w	
8	BS	CAN (8	H	X	h	x	
9	HT	EM)	9	I	Y	i	y	
A	LF	SUB *	:	J	Z	j	z	
B	VT	ESC +	;	K	[k	[
C	FF	FS ,	<	L	\	l	\	
D	CR	GS -	=	M]	m]	
E	SO	RS .	>	N	^	n	~	
F	SI	US /	?	O	_	o	DEL	

(下位4ビット)

基礎コンピュータ工学第2章 情報の表現 /パート5 : 小数や文字の表現と補助単位

6 / 10

文字の表現 (JIS 8 ビットコード)

JIS (Japan Industrial Standard : 日本工業規格) 8ビットコード
JIS 8ビットコードは、ASCII コードに半角カタカナを追加したもの。記号、数字、英字の部分は、ほぼ同じ並びになっている。

	(上位4ビット)																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0 NUL	DLE	0	@	P	-	p					※	※	※	※	※	※	
1 SOH	DC1	!	1	A	Q	a	q				※	※	※	※	※	※	
2 STX	DC2	"	2	B	R	b	r				※	※	※	※	※	※	
3 ETX	DC3	#	3	C	S	c	s				※	※	※	※	※	※	
4 EOT	DC4	\$	4	D	T	d	t				※	※	※	※	※	※	
ENQ	NAK	%	5	E	U	e	u				※	※	※	※	※	※	
6 ACK	SYN	&	6	F	V	f	v				※	※	※	※	※	※	
7 BEL	ETB	*	7	G	W	g	w				※	※	※	※	※	※	
8 BS	CAN	(8	H	X	h	x				※	※	※	※	※	※	
9 HT	EM)	9	I	Y	i	y				※	※	※	※	※	※	
A LF	SUB	*	:	J	Z	j	z				※	※	※	※	※	※	
B VT	ESC	=	K	L	k	{					※	※	※	※	※	※	
C FF	FS	_	L	M	l						※	※	※	※	※	※	
D CR	GS	-	=	M	m)					※	※	※	※	※	※	
E SO	RS	>	N	^	n						※	※	※	※	※	※	
F ST	US	/	2	O	c	o	ng				※	※	※	※	※	※	

はASCIIコード表と異なる部分

文字の表現 (Unicode)

Unicode：最も用いられている業界標準規格
世界中で使用される全ての文字を収録しようと現在も拡張が続いている。

- 全ての文字に 21 ビットのコードを割り振る。
最大で約 111 万文字にコードを割り振ることができる。
現在、約 15 万文字が登録されている。
 - 000000H～00007FH の範囲は ASCII コードと同じ
 - 000370H～0003FFH の範囲にギリシャ文字
 - その後に、キリル文字、アルメニア文字、ヘブライ文字...
 - 002E80H～009FFFH の範囲に漢字・かな・カナ・その他
 - 00F900H～00FAFFH の範囲は漢字
 - 01F600H～01F64FH の範囲に顔文字
 - 020000H～0323AEH の範囲も漢字

参考：<https://ja.wikipedia.org/wiki/Unicode>

文字の表現 (UTF-8)

Unicode のエンコーディング（符号化）方式の一つ

21ビットのUnicodeをそのままファイル等に書き込むと、00Hのバイトだらけになり容量がもったいないし、通信すると時間がかかる。

よく使う文字(ASCII)を短く符号化する。(8ビット可変長)

- 000000H～00007FH の範囲は 1 バイトに変換
(00H～7FH, 第 1 バイトが 0 で始まる)
 - 000080H～0007FFH の範囲は 2 バイトに変換
(C2H, 80H～DFH,BFH, 第 1 バイトが 110 で始まる)
 - 000800H～00FFFFH の範囲は 3 バイトに変換
(E0H,A0H,80H～EFH,BFH,BFH, 第 1 バイトが 1110 で始まる)
 - 010000H～10FFFFH の範囲は 4 バイトに変換
(F0H,90H,80H,80H～F7H,BFH,BFH,BFH, 第 1 バイトが 11110 で始まる)

第2バイト以降は、10で始まっている。

参考：<https://ja.wikipedia.org/wiki/UTF-8>

辅助单位

1,000m を 1km, 1,000g を 1kg, 0.001 l を 1ml, 0.001m を 1mm ここで, k や m は補助単位と呼ばれる.

一般的に			記憶容量		
値	記号	読み方	値	記号	読み方
10^3	k	キロ	2^{10}	Ki	キビ
10^6	M	メガ	2^{20}	Mi	メビ
10^9	G	ギガ	2^{30}	Gi	ギビ
10^{12}	T	テラ	2^{40}	Ti	ティ

- 通常は 10^3 毎に補助単位がある。
 - コンピュータの記憶容量では 2^{10} 毎に補助単位がある。
 $2^{10} = 1,024 = 1Ki$
 $10^3 = 1,000 = 1k$

**基礎コンピュータ工学
第2章 情報の表現
(パート6：論理演算と基本回路)**

<https://github.com/tctsigemura/TecTextBook>

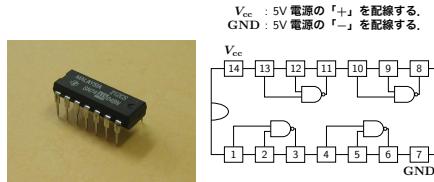
本スライドの入手：

基礎コンピュータ工学第2章 情報の表現 (パート6)
1 / 17

コンピュータの基本回路

論理回路を組合せてコンピュータは製作される。

- 電気の ON と OFF だけ用いて情報を表現する。
- ON/OFF をビット 1/0 に対応付ける。
- これは論理値 (True/False, 真/偽, Yes/No) と同じ。
- 論理値を対象とする演算を論理演算と言う。
- 論理演算を計算する回路を論理回路と言う。

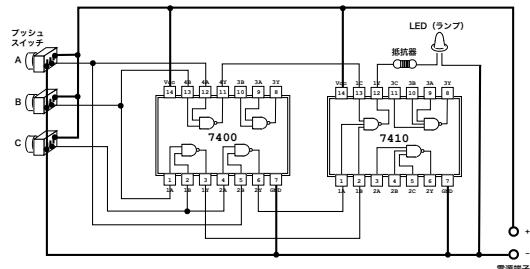
基礎コンピュータ工学第2章 情報の表現 (パート6)
2 / 17

論理 IC で作った手作りコンピュータ

徳山高専卒業研究で約40年前に製作された手作りコンピュータ

基礎コンピュータ工学第2章 情報の表現 (パート6)
3 / 17

論理 IC で回路を作る

基礎コンピュータ工学第2章 情報の表現 (パート6)
4 / 17

現代の手作りコンピュータ

FPGAの中に論理 IC 数万個に相当する回路が書き込める。

基礎コンピュータ工学第2章 情報の表現 (パート6)
5 / 17

基本的な論理回路 (1)

論理積 (AND) — 「かつ」

2ビット入力し、両方が1のときだけ1を出力する。

入力	出力	
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

AND の真理値表

AかつBが1なら1
A AND Bが1なら1

$$X = A \cdot B$$

AND の論理式



AND の回路記号

基礎コンピュータ工学第2章 情報の表現 (パート6)
6 / 17

基本的な論理回路（2）

論理和（OR）—「または」

2ビット入力し、どちらかが1のとき1を出力する。

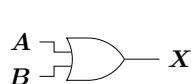
入力	出力	
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

OR の真理値表

A または B が1なら1
A OR B が1なら1

$$X = A + B$$

OR の論理式



OR の回路記号

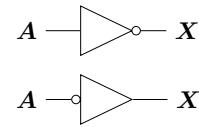
基本的な論理回路（3）

否定（NOT）—「ではない」

1ビット入力し、入力とは逆の論理値を出力する。

$$X = \overline{A}$$

NOT の論理式



NOT の回路記号

入力	出力
A	X
0	1
1	0

NOT の真理値表

A が1ではないなら1

基本的な論理回路（4）

排他的論理和（XOR）—「異なる」

2ビット入力し、二つが異なるなら1を出力する。

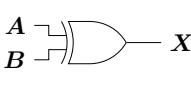
入力	出力	
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

XOR の真理値表

A と B が異なるなら1

$$X = A \oplus B$$

XOR の論理式

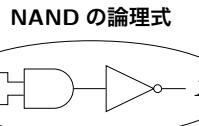


XOR の回路記号

基本的な論理回路（5）

NOT と AND の組合せ（NAND）

$$X = \overline{A \cdot B}$$



NAND の回路記号

入力	出力	
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

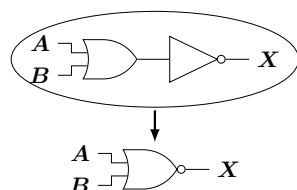
NAND の真理値表

基本的な論理回路（6）

NOT と OR の組合せ（NOR）

$$X = \overline{A + B}$$

NOR の論理式



NOR の回路記号

入力	出力	
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

NOR の真理値表

演算回路（1）

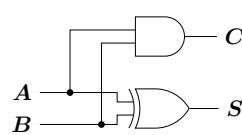
半加算器

1桁の2進数を二つ入力し、0, 1, 2のどれかを出力する。

$$\begin{array}{r}
 A \quad 0 \\
 + B \quad + 0 \\
 \hline
 C \quad S \quad 0 \quad 0
 \end{array}
 \quad
 \begin{array}{r}
 A \quad 0 \\
 + B \quad + 1 \\
 \hline
 C \quad S \quad 0 \quad 1
 \end{array}
 \quad
 \begin{array}{r}
 A \quad 1 \\
 + B \quad + 0 \\
 \hline
 C \quad S \quad 0 \quad 1
 \end{array}
 \quad
 \begin{array}{r}
 A \quad 1 \\
 + B \quad + 1 \\
 \hline
 C \quad S \quad 1 \quad 0
 \end{array}$$

入力	出力		
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

半加算器の真理値表



半加算器の回路図

演算回路（2）

全加算器

1桁の2進数を三つ入力し、0, 1, 2, 3のどれかを出力する。

入力	出力			
A	B	I	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

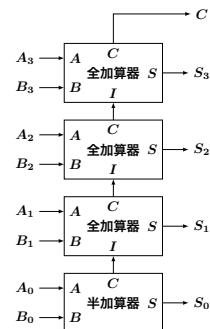
A : 計算の入力
B : 計算の入力
I : 桁上がりの入力

全加算器の真理値表

演算回路（3）

4ビット加算器

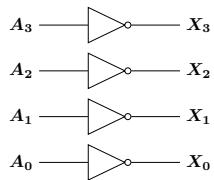
4桁の2進数を二つ入力し、和を計算する



演算回路（4）

4ビット 1の補数器

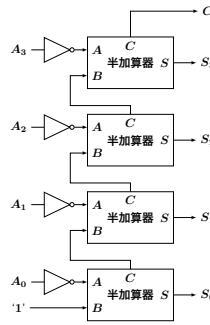
4桁の2進数を入力し、1の補数を計算する



演算回路（5）

4ビット 2の補数器

4桁の2進数を入力し、2の補数を計算する

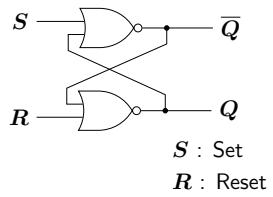


記憶回路

RS フリップフロップ

直前の状態を記憶する回路

入力	出力
S R	Q
0 0	記憶
0 1	0
1 0	1
1 1	禁止



RS-FF の動作

RS-FF の回路

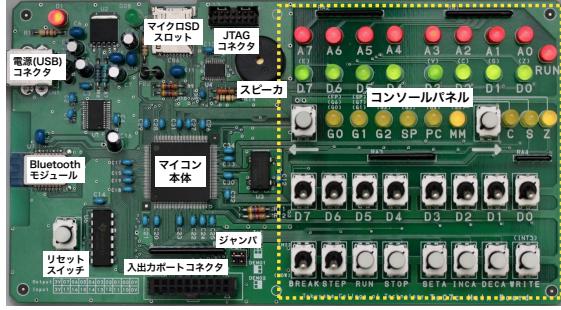
基礎コンピュータ工学 第4章 マイコンの構成と操作

<https://github.com/tctsigemura/TecTextBook>

本スライドの入手：

基礎コンピュータ工学第4章 マイコンの構成と操作 1 / 10

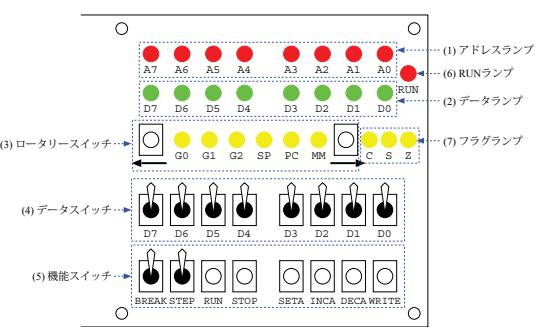
各部の名称



- ビデオ（各部の名称、前半）

基礎コンピュータ工学第4章 マイコンの構成と操作 2 / 10

コンソールパネル

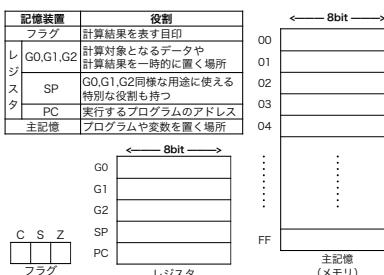


- ビデオ（各部の名称、後半）

基礎コンピュータ工学第4章 マイコンの構成と操作 3 / 10

TeC 内部の記憶装置

記憶装置	役割
フラグ	計算結果を表す目印
G0,G1,G2	計算対象となるデータや計算結果を一時的に置く場所
SP	G0,G1,G2同様の用途に使える特別な役割を持つ
PC	実行するプログラムのアドレス
主記憶	プログラムや変数を置く場所



- ビデオ（TeC 内部の記憶装置）

基礎コンピュータ工学第4章 マイコンの構成と操作 4 / 10

記憶装置の内容を表示／書き込み

- ビデオ（表示から書き込みまで）
- フラグ
- レジスタ
- 主記憶（メモリ）

基礎コンピュータ工学第4章 マイコンの構成と操作 5 / 10

プログラムの実行

- 主記憶にプログラムを書き込む。
- PC（プログラムカウンタ）
- STEP, BREAK スイッチ
- RESET, RUN スイッチ

基礎コンピュータ工学第4章 マイコンの構成と操作 6 / 10

練習問題1

次のプログラムを実行しなさい。（プログラムは16進数で書く）

- 主記憶にプログラムを書き込む。

番地	データ	コメント
00	13	
01	0A	
02	17	
03	0F	
04	1B	
05	A0	
06	1F	
07	F0	
08	FF	

- 番地から実行する。
- 実行後の各レジスタの値は？
- STEP 実行を用いて各命令の意味を推定する。→コメントに書く

練習問題2（前半）

どのような命令が含まれているか推定しなさい。

- プログラム1

番地	データ	コメント
00	13	
01	01	
02	33	
03	01	
04	FF	

- プログラム2

番地	データ	コメント
00	13	
01	01	
02	33	
03	01	
04	A0	
05	02	

練習問題2（後半）

- プログラム3

番地	データ	コメント
00	13	
01	01	
02	20	メモリの 10_{16} 番地に何か起こる
03	10	
04	FF	

期末試験について

1. 試験範囲

- 中間試験の範囲に加えて第4章
- 第4章の内容は、操作方法、プログラムの実行と実行結果の確認

2. 持ち込み物品

TeC本体、ケース、電源ケーブル

3. 試験の準備

- 中間試験の範囲を良く復習する。
- 教科書の第4章を良く読む。
- ビデオ（教科書21ページのQRコード）を良く見る。
- 実際にTeCで試してみる。

4. 参考（過去問）

練習問題として活用してください。

<https://github.com/tctsigemura/Exam/tree/master/FCE>

基礎コンピュータ工学 第5章 機械語プログラミング (パート1：プログラムの実行)

<https://github.com/tctsigemura/TecTextBook>

本スライドの入手：

基礎コンピュータ工学第5章 機械語プログラミング

1 / 10

本科目の目的を再確認

「ノイマン型コンピュータ」の基本原理を学ぶ。
(99%以上のコンピュータはノイマン型だから。)

これまでに学んだこと。

(1) 情報の表現 (2進数(ON/OFF)で情報表現できる。)

おおかみ情報、数値(計算、負数、小数)、文字

(2) コンピュータの基本回路 (2進数の計算や記憶ができる。)

NOT, AND, OR, XOR, 加算器, RS-FF

(3) マイコンの組み立てと操作

ハンダ、コンソールパネル、レジスタ、フラグ、メモリ

基礎コンピュータ工学第5章 機械語プログラミ

2 / 10

コンピュータとは

- コンピュータって何？
Compute(計算する) + er(もの) = Computer(計算機)
もともとは、数値計算をするための機械
- 計算機？(電卓と何が違うの？)
計算手順を記憶することができる。(平均点を計算する例)

電卓：

コンピュータ：

ノイマン型コンピュータは計算手順を記憶できる。

基礎コンピュータ工学第5章 機械語プログラミ

3 / 10

ノイマン型コンピュータの特徴

● プログラム内蔵方式(ストアード・プログラム方式)
データだけでなく、プログラムもメモリに記憶する。

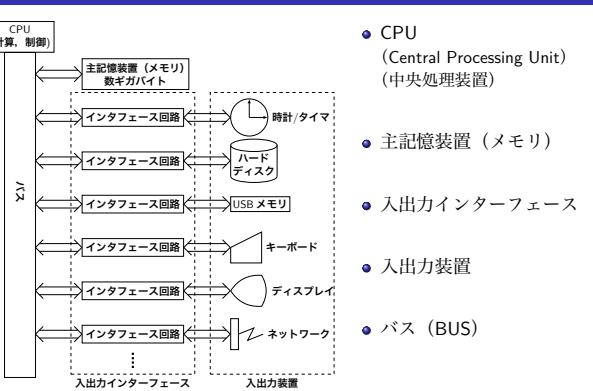
● 逐次実行方式
メモリに記憶したプログラムの命令を、
一つ一つ順番に(自動的に)実行する。

● 2進法
コンピュータ内部の情報表現は、
ハードウェアで扱いやすい2進数を用いる。

基礎コンピュータ工学第5章 機械語プログラミ

4 / 10

コンピュータの構成(一般的)

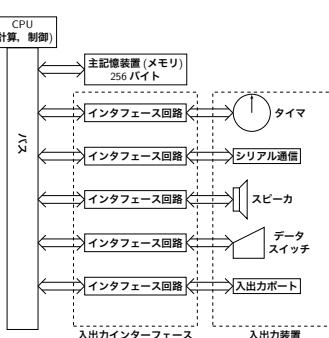


一般的なコンピュータ構成図を示す。CPU(計算、制御)は主記憶装置(メモリ)256バイトとインターフェース回路を介して接続される。インターフェース回路は、ディスク、USBメモリ、キーボード、ディスプレイ、ネットワークなどの外部装置と接続している。また、CPUはバス(BUS)を介して入出力インターフェースと接続されている。

基礎コンピュータ工学第5章 機械語プログラミ

5 / 10

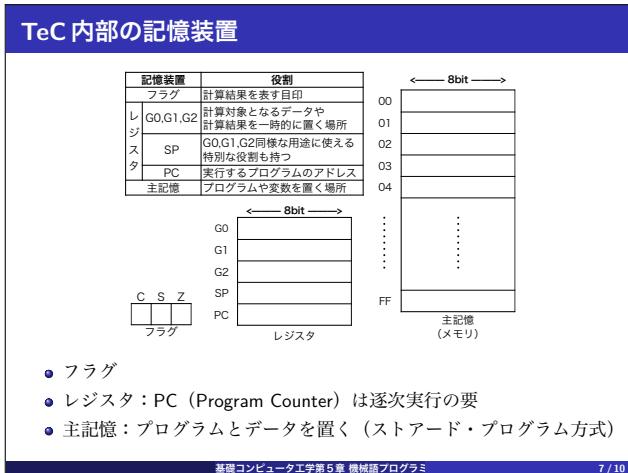
コンピュータの構成(TeCの場合)



TeCの場合のコンピュータ構成図を示す。CPU(計算、制御)は主記憶装置(メモリ)256バイト、入出力インターフェース、入出力装置、バス(BUS)を介して接続される。インターフェース回路は、シリアル通信、スピーカー、データスイッチ、入出力ポートなどの外部装置と接続している。

基礎コンピュータ工学第5章 機械語プログラミ

6 / 10



機械語プログラミングと機械語命令

「機械語 (Machine Language)」=機械 (CPU) の言語

「機械語プログラミング」=機械語プログラムを作る作業のこと

「機械語プログラム」=機械語命令で記述したプログラムのこと

「機械語命令」=機械 (CPU) が理解できる命令のこと
(機械語命令は2進数で表現する。)

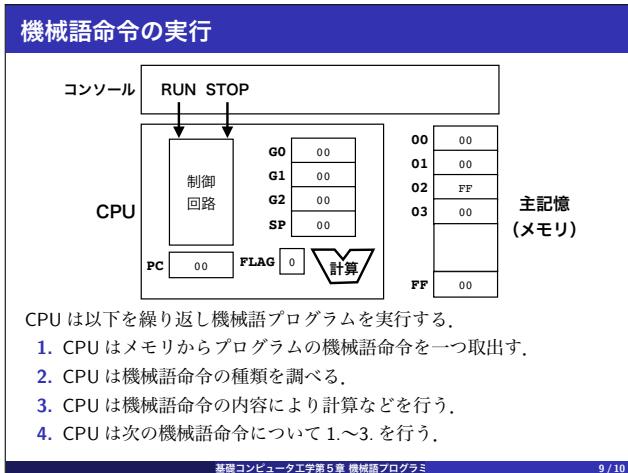
機械語プログラムの例

機械語命令	ニーモニック	意味
0000 0000 ₂	NO	No Operation
1111 1111 ₂	HALT	Halt

「ニーモニック」=命令の意味の英語を簡略化した綴

基礎コンピュータ工学第5章 機械語プログラミング

8 / 10



演習

逐次実行とPC (Program Counter) の働きを確認する。
以下のプログラムを実行した後のPCの値はいくつになるか？

番地	命令	番地	命令	番地	命令
00 ₁₆	00 ₁₆	00 ₁₆	NO	00 ₁₆	NO
01 ₁₆	FF ₁₆	01 ₁₆	HALT	01 ₁₆	NO
		02 ₁₆	00 ₁₆	02 ₁₆	00 ₁₆
		03 ₁₆	FF ₁₆	03 ₁₆	NO
				04 ₁₆	00 ₁₆
				05 ₁₆	00 ₁₆
				06 ₁₆	FF ₁₆

次の言葉の意味を確認しなさい。

- プログラム内蔵方式
- 逐次実行方式
- 2進法
- CPU, メモリ
- PC
- 機械語
- ニーモニック
- NO, HALT

基礎コンピュータ工学第5章 機械語プログラミング

10 / 10

**基礎コンピュータ工学
第5章 機械語プログラミング
(パート2：転送命令)**

<https://github.com/tctsigemura/TecTextBook>

本スライドの入手：

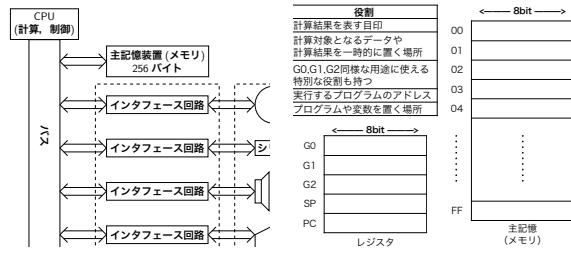
基礎コンピュータ工学第5章 機械語プログラミング

1 / 9

データ転送命令

CPUとメモリの間でデータを転送する機械語命令（2種類）

- LD (Load) 命令：CPUのレジスタ ← メモリ
- ST (Store) 命令：メモリ ← CPUのレジスタ



基礎コンピュータ工学第5章 機械語プログラミング

2 / 9

LD (Load) 命令（ニーモニックと命令フォーマット）

メモリ (EA) から CPU のレジスタ (GR) へデータを転送 (コピー) する。
ニーモニック： LD GR, EA

命令フォーマット：2 バイトの長さを持つ。

第1 バイト	第2 バイト
OP	GR XR
0001 ₂	GR XR aaaa aaaa

フィールド： OP, GR, XR, A

GR フィールドの意味と値： GR の 2 ビットで CPU レジスタを指定する。

GR	意味
00 ₂	G0
01 ₂	G1
10 ₂	G2
11 ₂	SP

基礎コンピュータ工学第5章 機械語プログラミング

3 / 9

LD (Load) 命令（具体的な命令の例）

メモリの 3 番地からから G1 レジスタへデータを転送 (コピー) する。
ニーモニック： LD G1,03H

命令フォーマット： G1 と 03H を反映する。

第1 バイト	第2 バイト
OP	GR XR
0001 ₂	01 ₂ 00 ₂ 0000 0011 ₂

メモリに格納した状態： HALT 命令やデータも格納している。

番地	命令
0016	14 ₁₆ LD G1,03H
0116	03 ₁₆ HALT
0216	FF ₁₆
0316	12 ₁₆ 何かデータ

基礎コンピュータ工学第5章 機械語プログラミング

4 / 9

LD (Load) 命令（少し長い例）

プログラムの例： データを G0, G1 にロードする。

番地	機械語	ラベル	ニーモニック
00 ₁₆	10 ₁₆ 05 ₁₆		LD G0,05H
02 ₁₆	14 ₁₆ 06 ₁₆		LD G1,06H
04 ₁₆	FF ₁₆		HALT

メモリに格納した状態： 何かデータも準備する必要がある。

番地	機械語	意味
00 ₁₆	10 ₁₆	LD G0,05H
01 ₁₆	05 ₁₆	
02 ₁₆	14 ₁₆	LD G1,06H
03 ₁₆	06 ₁₆	
04 ₁₆	FF ₁₆	HALT
05 ₁₆	12 ₁₆	データ！！
06 ₁₆	34 ₁₆	データ！！

基礎コンピュータ工学第5章 機械語プログラミング

5 / 9

LD (Load) 命令（フローチャートの描き方）

LD 命令のフローチャート： [] を忘れないように！

```

    graph TD
        START([START]) --> G0["G0 ← [05H]"]
        G0 --> G1["G1 ← [06H]"]
        G1 --> END([END])
    
```

LD 命令のフローチャート例： START と END を追加

```

    graph TD
        START([START]) --> G0["G0 ← [05H]"]
        G0 --> G1["G1 ← [06H]"]
        G1 --> END([END])
    
```

基礎コンピュータ工学第5章 機械語プログラミング

6 / 9

ST (Store) 命令 (ニーモニックと命令フォーマット)

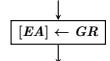
CPUのレジスタからメモリへデータを転送（コピー）する。

ニーモニック： ST GR,EA

命令フォーマット： 2バイトの長さを持つ。

第1バイト		第2バイト
OP	GR XR	
0010 ₂	GR XR	aaaa aaaa

ST 命令のフローチャート： [と] を忘れないように！

**ST (Store) 命令 (プログラム例)**

プログラムの例： 05H番地のデータを06H番地にコピーする。

番地	機械語	ラベル	ニーモニック
00	10 05		LD G0,05H
02	20 06		ST G0,06H
04	FF		HALT

番地と機械語はいつも16進数で書く（小さく16と書く必要なし）。

フローチャート： 上のプログラムのフローチャートを描いてみる。

演習

次の手順を守って演習を行う。

1. フローチャートを描いて考えをまとめる。
2. ニーモニック（オペレーション、オペランド）に変換する。
3. 番地（アドレス）を決める。
4. 機械語を決める。
5. TeCに打ち込み実行して結果を確認する。

Title	基礎計算機工学 演習課題	No. _____	氏名 _____	Date	_____	No. 6
(1) 11H番地のデータを12H番地に、10H番地のデータを11H番地にコピーするプログラム						
フローチャート	アドレス	機械語	ラベル	オペレーション	オペランド	

**基礎コンピュータ工学
第5章 機械語プログラミング
(パート3：計算命令)**

<https://github.com/tctsigemura/TecTextBook>

本スライドの入手：

基礎コンピュータ工学第5章 機械語プログラミング

1 / 10

算術演算命令（2種類）

算術計算（普通の計算）をする命令

- ADD (Add) 命令：加算命令（足し算をする）
CPU のレジスタ \leftarrow CPU のレジスタ + メモリ
- SUB (Subtract) 命令：減算命令（引き算をする）
CPU のレジスタ \leftarrow CPU のレジスタ - メモリ

CPU のレジスタは一つだけ使用できる。

- ADD 命令の例：
 $G0 \leftarrow G0 + [EA]$
- SUB 命令の例：
 $G1 \leftarrow G1 - [EA]$

ADD (Add) 命令（ニーモニックと命令フォーマット）

メモリ (EA) のデータを CPU のレジスタ (GR) 足し込む。
ニーモニック： ADD GR,EA ($GR \leftarrow GR + [EA]$)

命令フォーマット：2バイトの長さを持つ。

第1バイト	第2バイト
OP	GR XR
0011 ₂	GR XR
aaaa aaaa	

例：メモリの3番地のデータを G2 レジスタへ足し込む。
ニーモニック： ADD G2,03H 動作：
命令フォーマット： G2 と 03H を反映する。

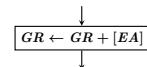
第1バイト	第2バイト
OP	GR XR
0011 ₂	10 ₂ 00 ₂
0000 0011 ₂	

基礎コンピュータ工学第5章 機械語プログラミング

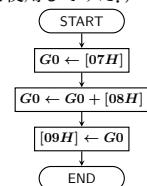
3 / 10

ADD (Add) 命令（フローチャートの描き方）

ADD 命令のフローチャート：[と]を忘れないように！



例：7番地のデータと8番地のデータの和を9番地に求める。
(今回は G0 を使用してみた。)



ADD (Add) 命令（プログラム例）

プログラムの例：7番地と8番地のデータの和を9番地に求める。

番地	機械語	ラベル	ニーモニック
00	10 07		LD G0,07H
02	30 08		ADD G0,08H
04	20 09		ST G0,09H
06	FF		HALT

メモリに格納した状態：何かデータも準備する必要がある。

番地	機械語	意味
00	10	LD G0,07H
01	07	
02	30	ADD G0,08H
03	08	
04	20	ST G0,09H
05	09	
06	FF	HALT
07	12	データ!!
08	34	データ!!
09	00	データ!!

基礎コンピュータ工学第5章 機械語プログラミング

5 / 10

SUB (Subtract) 命令（ニーモニックとフォーマット）

メモリ (EA) のデータを CPU のレジスタ (GR) から引く。

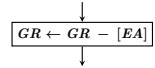
ニーモニック： SUB GR,EA ($GR \leftarrow GR - [EA]$)**命令フォーマット：** 2バイトの長さを持つ。

例：メモリの3番地のデータを G1 レジスタから引く。

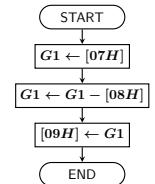
ニーモニック： SUB G1,03H 動作：**命令フォーマット：** G1 と 03H を反映する。

SUB (Subtract) 命令 (フローチャートの描き方)

SUB 命令のフローチャート： [と] を忘れないように！



例：7番地のデータと8番地のデータの差を9番地に求める。
(今回はG1を使用してみた。)



SUB (Subtract) 命令 (プログラム例)

プログラムの例：7番地と8番地のデータの差を9番地に求める。

番地	機械語	ラベル	ニーモニック
00	14 07		LD G1,07H
02	44 08		SUB G1,08H
04	24 09		ST G1,09H
06	FF		HALT

メモリに格納した状態：何かデータも準備する必要がある。

番地	機械語	意味
00	14 07	LD G1,07H
01	07	
02	44	SUB G1,08H
03	08	
04	24	ST G1,09H
05	09	
06	FF	HALT
07	0A	データ！！
08	03	データ！！
09	00	データ！！

演習 (1)

足し算プログラム、引き算プログラムのデータを変更して計算結果を確認しなさい。

$$\begin{array}{r}
 0000\ 1010\ (10) \\
 +\ 0000\ 1010\ (10) \\
 \hline
 0001\ 0100\ (20)
 \end{array}
 \quad
 \begin{array}{r}
 0000\ 1010\ (-10) \\
 +\ 1111\ 0110\ (-10) \\
 \hline
 (\)
 \end{array}
 \quad
 \begin{array}{r}
 1100\ 0000\ (-64) \\
 +\ 0111\ 0000\ (112) \\
 \hline
 (\)
 \end{array}$$

$$\begin{array}{r}
 0000\ 1010\ () \\
 -\ 0000\ 0101\ () \\
 \hline
 (\)
 \end{array}
 \quad
 \begin{array}{r}
 0000\ 1100\ (-12) \\
 -\ 1111\ 0011\ (-13) \\
 \hline
 (\)
 \end{array}
 \quad
 \begin{array}{r}
 0000\ 0000\ (0) \\
 -\ 0000\ 0001\ (1) \\
 \hline
 (\)
 \end{array}$$

演習 (2)

次の手順を守って演習を行う。

1. フローチャートを描いて考えをまとめる。
2. ニーモニック（オペレーション、オペランド）に変換する。
3. 番地（アドレス）を決める。
4. 機械語を決める。
5. TeCに打ち込み実行して結果を確認する。

Title. 基礎計算機工学 演習課題 No. _____ 氏名 _____	Date _____	No. 6
(1) 11H番地のデータを12H番地に、10H番地のデータを11H番地にコピーするプログラム。		
フローチャート	アドレス	機械語
	ラベル	オペレーション
		オペランド

**基礎コンピュータ工学
第5章 機械語プログラミング
(パート4：分岐命令)**

<https://github.com/tctsigemura/TecTextBook>

本スライドの入手：

プログラムの流れ

- プログラムは番地順に実行される（逐次実行）。
- 実行が進んでいく流れを「プログラムの流れ」と呼ぶ。
- 「プログラムの流れ」はPCによって管理されている。
- 通常、PCは増加していく。
- 「プログラムの流れ」を別のアドレスに変えることも必要。
 - 条件によって処理内容を変更したい場合。
 - 同じ処理内容を繰り返したい場合。
- 「プログラムの流れ」を変える命令をジャンプ命令と呼ぶ。
「プログラムの流れ」を飛ばす = PCにアドレスをロードする

基礎コンピュータ工学第5章 機械語プログラミ

2 / 10

ジャンプ命令（7種類）

無条件ジャンプ命令：プログラムの流れを指定のアドレスに飛ばす。

条件ジャンプ命令：条件が成立したときだけジャンプする。

無条件ジャンプ命令（JMP命令）の役割イメージ

番地	機械語	ラベル	ニーモニック
00	10 08		LD G0,08H
02	30 09		ADD G0,09H
04	20 0A		ST G0,0AH
06	A0 0B		JMP 0BH
08	12	データ	
09	34	データ	
0A	00	データ	
0B	30 09		ADD G0,09H
0D	...		

JMP (Jump) 命令（ニーモニックと命令フォーマット）

無条件ジャンプ命令： JMP (Jump) 命令

ニーモニック： JMP EA (PC ← EA)

命令フォーマット： 2バイトの長さを持つ。

第1バイト	第2バイト
OP	GR XR
1010 ₂	00 ₂ XR aaaa aaaa

例：メモリの10₁₆番地へ飛ぶ（ジャンプする）。

ニーモニック： JMP 10H **動作：**PCに10₁₆をロードする。

命令フォーマット： 10Hを反映する。

第1バイト	第2バイト
OP	GR XR
1010 ₂	00 ₂ 00 ₂ 0001 0000 ₂

基礎コンピュータ工学第5章 機械語プログラミ

4 / 10

JMP (Jump) 命令（フローチャートとプログラム例）

JMP命令のフローチャート： ←, →, ↑, ↓など

フローチャートの例： ADD命令を永遠に繰り返す。（無限ループ）



```

graph TD
    START([START]) --> ADD[G0 ← G0 + [04H]]
    ADD --> START
  
```

プログラムの例： 0番地のADD命令を永遠に繰り返す。（無限ループ）

番地	機械語	ラベル	ニーモニック
00	30 04		ADD G0,04H
02	A0 00		JMP 00H

演習(1)： 上のプログラムを4番地に1を格納した状態で実行する。
STOPボタンでプログラムを停止し G0 の値を確認する。

ラベル

ニーモニックだけでプログラムを完結させるために使用する。

- JMP命令のプログラム例では、ジャンプ先のアドレスをニーモニックに中に数値で書いた。
- 機械語の番地が決まらないとニーモニックが完成しない。一方で、ニーモニックを書かないと機械語が完成しない。
- ニーモニックだけでプログラムを完結させる必要がある。
→ 場所（アドレス）に名前（ラベル）を付ける。

前のプログラムをラベルを使って書き直したもの。

番地	機械語	ラベル	ニーモニック
00	30 04	LOOP	ADD G0,04H
02	A0 00		JMP LOOP

LOOP = 「輪」 … 意味を持った名前を付けるとより良い。

基礎コンピュータ工学第5章 機械語プログラミ

6 / 10

DC (Define Constant) 命令

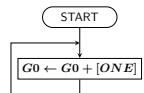
まだ、データ部分がニーモニックで表現できていない。

- データもニーモニックで表現できる必要がある。
- DC命令はデータを記述するための疑似命令 (\neq 機械語命令)
- ニーモニック : DC データの値
- 前のプログラムを DC命令を使って書き直したもの。

番地	機械語	ラベル	ニーモニック
00	30 04	LOOP	ADD G0, ONE
02	A0 00		JMP LOOP
04	01	ONE	DC 1

データの番地 (04H) もラベル (ONE) で参照できる。

- フローチャートの例



DS (Define Storage) 命令

結果を格納する領域を作るための疑似命令。

ニーモニック : DS 領域の大きさ (バイト数)

プログラムの例 : X番地と Y番地のデータの和を Z番地に求める。
(7番地と8番地のデータの和を9番地に求めると同じ)

番地	機械語	ラベル	ニーモニック
00	10 07		LD G0, X
02	30 08		ADD G0, Y
04	20 09		ST G0, Z
06	FF		HALT
07	12	X	DC 12H
08	34	Y	DC 34H
09	00	Z	DS 1

DC命令とDS命令の使い分け

入力となるデータを色々変化させたい場合。

プログラムの例 : X番地のデータに1を加えたものをZ番地に求める。

番地	機械語	ラベル	ニーモニック
00	10 08		LD G0, X
02	30 07		ADD G0, ONE
04	20 09		ST G0, Z
06	FF		HALT
07	01	ONE	DC 1
08	00	X	DS 1
09	00	Z	DS 1

DCとDSの区別 : 値が変化しないものをDCで準備する。

入力になるものは、典型的な値をDCで準備する。

入力になるものは、後で決めるのでDSで場所を確保する。

出力は、DSで場所を確保する。

まとめ

学んだこと

- 無条件ジャンプ命令 (JMP命令)
- ラベル
- データを表現する命令 (DC命令)
- データ領域を予約する命令 (DS命令)

演習 (2) (以下の目的で演習を行う)

- PCの役割を再確認する。
- PCとJMP命令の関係を調べる。
- 計算結果とフラグの関係を調べる。※
- ステップモード実行の練習をする。
- ブレークモード実行の練習をする。

※次回はフラグの値を条件にするジャンプ命令を学ぶ。

**基礎コンピュータ工学
第5章 機械語プログラミング
(パート5：フラグと条件分岐)**

<https://github.com/tctsigemura/TecTextBook>

本スライドの入手：

基礎コンピュータ工学第5章 機械語プログラミング

1/14

フラグ (1)

フラグ (C, S, Z) は計算結果の特徴を表す。
フラグ変化ありの命令を実行する度に値が変化する。
(教科書の本文、命令表を再度確認する。)

- **Z (Zero) フラグ**

- Zero は「ゼロ」の意味。

- 計算の結果がゼロにならなかった。

$$\begin{array}{r} 0011\ 0010_2 \\ + \quad 0011\ 0010_2 \\ \hline Z \boxed{0} \quad 0110\ 0100_2 \end{array} \rightarrow \begin{array}{r} 50_{10} \\ + \quad 50_{10} \\ \hline 100_{10} \end{array}$$

- 計算の結果がゼロになった。

$$\begin{array}{r} 0011\ 0010_2 \\ - \quad 0011\ 0010_2 \\ \hline Z \boxed{1} \quad 0000\ 0000_2 \end{array} \rightarrow \begin{array}{r} 50_{10} \\ - \quad 50_{10} \\ \hline 0_{10} \end{array}$$

基礎コンピュータ工学第5章 機械語プログラミング

2/14

フラグ (2)

- **S (Sign) フラグ**

- Sign はプラス・マイナスの「符号」の意味

- 計算の結果を符号付き2進数と解釈すると正の値になった。

$$\begin{array}{r} 0011\ 0010_2 \qquad \qquad 50_{10} \\ + \quad 0011\ 0011_2 \qquad \rightarrow \quad + \quad 51_{10} \\ \hline S \boxed{0} \quad 0110\ 0101_2 \qquad \qquad 101_{10} \end{array}$$

- 計算の結果を符号付き2進数と解釈すると負の値になった。

$$\begin{array}{r} 0011\ 0010_2 \qquad \qquad 50_{10} \\ - \quad 0011\ 0011_2 \qquad \rightarrow \quad - \quad 51_{10} \\ \hline S \boxed{1} \quad 1111\ 1111_2 \qquad \qquad -1_{10} \end{array}$$

- S フラグは符号付き2進数と考えたときの「負」の意味

- 計算結果の最上位ビットと同じ値になる。→ゼロは「正」とみなす。

基礎コンピュータ工学第5章 機械語プログラミング

3/14

フラグ (3)

- **C (Carry) フラグ**

- Carry は「桁を繰り上げる」の意味

- 足し算 (ADD) で桁上げが起きる。

- 足し算で最上位桁からの桁上げがない場合

$$\begin{array}{r} 0111\ 1111_2 \qquad \qquad 127_{10} \\ + \quad 0000\ 0001_2 \qquad \rightarrow \quad + \quad 1_{10} \\ \hline C \boxed{0} \quad 1000\ 0000_2 \qquad \qquad 128_{10} \end{array}$$

- 足し算で最上位桁からの桁上げがあった場合 (オーバーフロー)

$$\begin{array}{r} 1111\ 1111_2 \qquad \qquad 255_{10} \\ + \quad 0000\ 0001_2 \qquad \rightarrow \quad + \quad 1_{10} \\ \hline C \boxed{1} \quad 0000\ 0000_2 \qquad \qquad 0_{10} \end{array} ?$$

基礎コンピュータ工学第5章 機械語プログラミング

4/14

フラグ (4)

- **C (Carry) フラグ (Borrow の意味を代用)**

- Borrow は「桁を借りる」の意味

- 引き算 (SUB) で桁借りが起こる

- 引き算で最上位桁で桁借りがない場合

$$\begin{array}{r} 1111\ 1111_2 \qquad \qquad 255_{10} \\ - \quad 0000\ 0001_2 \qquad \rightarrow \quad - \quad 1_{10} \\ \hline C \boxed{0} \quad 1111\ 1110_2 \qquad \qquad 254_{10} \end{array}$$

- 引き算で最上位桁で桁借りがあった場合 (負にオーバーフロー)

$$\begin{array}{r} 0000\ 0000_2 \qquad \qquad 0_{10} \\ - \quad 0000\ 0001_2 \qquad \rightarrow \quad - \quad 1_{10} \\ \hline C \boxed{1} \quad 1111\ 1111_2 \qquad \qquad 255_{10} \end{array} ?$$

- C フラグは、符号なし2進数と考えたときのオーバーフローの意味

基礎コンピュータ工学第5章 機械語プログラミング

5/14

ジャンプ命令 (7種類)

無条件ジャンプ命令：プログラムの流れを指定のアドレスに飛ばす。

- **JMP (Jump) 命令：**いつもジャンプする。

条件ジャンプ命令：ある条件のときだけジャンプする。

- **JZ (Jump on Zero) 命令：**Z = 1 ならジャンプ

- **JC (Jump on Carry) 命令：**C = 1 ならジャンプ

- **JM (Jump on Minus) 命令：**S = 1 ならジャンプ

- **JNZ (Jump on Not Zero) 命令：**Z = 0 ならジャンプ

- **JNC (Jump on Not Carry) 命令：**C = 0 ならジャンプ

- **JNM (Jump on Not Minus) 命令：**S = 0 ならジャンプ

基礎コンピュータ工学第5章 機械語プログラミング

6/14

JZ (Jump on Zero) 命令

Z フラグが 1 なら（計算結果が 0 なら）ジャンプする。

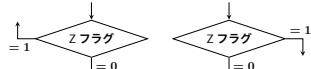
フラグ：変化しない。

ニーモニック： JZ EA (if(Z=1) PC ← EA)

命令フォーマット： 2 バイトの長さを持つ。

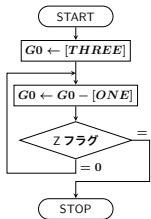
第1バイト		第2バイト	
OP	GR XR		
1010 ₂	01 ₂ XR	aaaa	aaaa

フローチャート： ある程度、自由にアレンジしてよい。



JZ 命令の使用例

ループを 3 回、繰り返すプログラム



番地	機械語	ラベル	ニーモニック
00	10 09		LD GO, THREE
02	40 0A	LOOP	SUB GO, ONE
04	A4 08		JZ STOP
06	A0 02		JMP LOOP
08	FF	STOP	HALT
09	03	THREE	DC 3
0A	01	ONE	DC 1

- 演習 (1)：ステップモードで実行をトレースしてみる。

JC (Jump on Carry) 命令

C フラグが 1 なら（オーバーフローなら）ジャンプする。

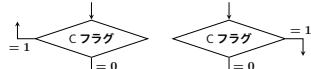
フラグ：変化しない。

ニーモニック： JC EA (if(C=1) PC ← EA)

命令フォーマット： 2 バイトの長さを持つ。

第1バイト		第2バイト	
OP	GR XR		
1010 ₂	10 ₂ XR	aaaa	aaaa

フローチャート： ある程度、自由にアレンジしてよい。



JM (Jump on Minus) 命令

S フラグが 1 なら（負なら）ジャンプする。

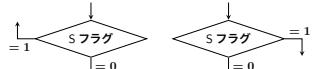
フラグ：変化しない。

ニーモニック： JM EA (if(S=1) PC ← EA)

命令フォーマット： 2 バイトの長さを持つ。

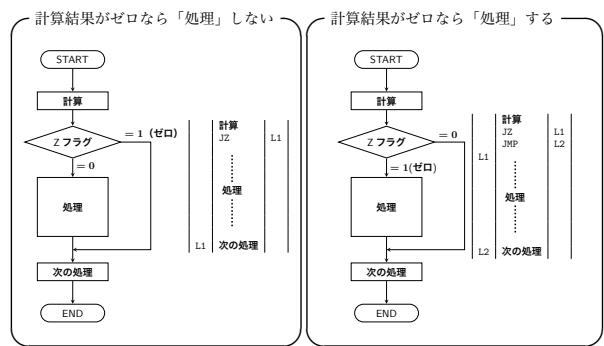
第1バイト		第2バイト	
OP	GR XR		
1010 ₂	11 ₂ XR	aaaa	aaaa

フローチャート： ある程度、自由にアレンジしてよい。



条件判断 1

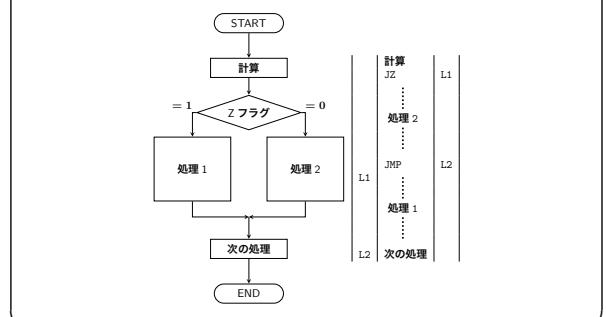
計算結果により処理をするかしないか変化する例



条件判断 2

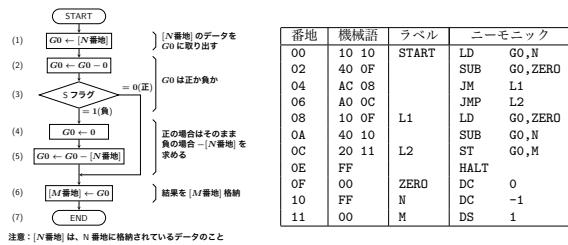
計算結果によりどちらかの処理をする例

計算結果がゼロなら「処理 1」、そうでなければ「処理 2」をする



条件判断の例

絶対値を求めるプログラム（例題5-1）



- 演習（2）：ステップモードで実行をトレースしてみる。

まとめ

学んだこと

- フラグ (Carry, Zero, Sign)
- 条件ジャンプ命令 (JZ, JC, JM)
- 条件判断

演習（宿題）

- 飽和演算：計算結果が最大値または最小値を超えそうになった時、計算結果を最大値または最小値に留める演算方式
- TeC の符号なし 2 進数を用いて表現できる最大値は 255 である。
- 足し算結果が 255 を超える（オーバーフローする）かもしれない。
- オーバーフローが発生したら計算結果を 255 に訂正するようにする。
- 以上のような足し算プログラムを作る。

**基礎コンピュータ工学
第5章 機械語プログラミング
(パート6：残りの条件分岐命令と条件判断の演習)**

<https://github.com/tctsigemura/TecTextBook>

本スライドの入手：

基礎コンピュータ工学第5章 機械語プログラミング 1 / 8

ジャンプ命令（7種類）の残り3種類**無条件ジャンプ命令：**プログラムの流れを指定のアドレスに飛ばす。

- **JMP (Jump)** 命令：いつもジャンプする。

条件ジャンプ命令：ある条件のときだけジャンプする。

- **JZ (Jump on Zero)** 命令： $Z = 1$ ならジャンプ
- **JC (Jump on Carry)** 命令： $C = 1$ ならジャンプ
- **JM (Jump on Minus)** 命令： $S = 1$ ならジャンプ
- **JNZ (Jump on Not Zero)** 命令： $Z = 0$ ならジャンプ
- **JNC (Jump on Not Carry)** 命令： $C = 0$ ならジャンプ
- **JNM (Jump on Not Minus)** 命令： $S = 0$ ならジャンプ

JNZ (Jump on Not Zero) 命令

Z フラグが 0 なら（計算結果が 0 でないなら）ジャンプする。

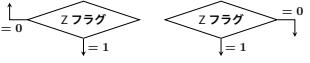
フラグ：変化しない。

ニーモニック： JNZ EA (if($Z=0$) PC ← EA)

命令フォーマット：2バイトの長さを持つ。

第1バイト		第2バイト	
OP	GR XR		
1011 ₂	01 ₂ XR	aaaa	aaaa

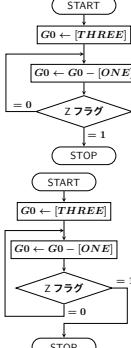
フローチャート：ある程度、自由にアレンジしてよい。



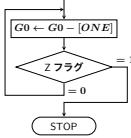
基礎コンピュータ工学第5章 機械語プログラミング 3 / 8

JNZ 命令の使用例 (JZ 命令との比較)

ループを3回、繰り返すプログラム



番地	機械語	ラベル	ニーモニック
00	10 07		LD GO, THREE
02	40 08	LOOP	SUB GO, ONE
04	B4 02		JNZ LOOP
06	FF		HALT
07	03	THREE	DC 3
08	01	ONE	DC 1



番地	機械語	ラベル	ニーモニック
00	10 09		LD GO, THREE
02	40 0A	LOOP	SUB GO, ONE
04	A4 08		JZ STOP
06	A0 02		JMP LOOP
08	FF		HALT
09	03	THREE	DC 3
0A	01	ONE	DC 1

基礎コンピュータ工学第5章 機械語プログラミング 4 / 8

JNC (Jump on Not Carry) 命令

C フラグが 0 なら（オーバーフローしていないなら）ジャンプする。

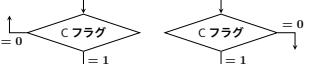
フラグ：変化しない。

ニーモニック： JNC EA (if($C=0$) PC ← EA)

命令フォーマット：2バイトの長さを持つ。

第1バイト		第2バイト	
OP	GR XR		
1011 ₂	10 ₂ XR	aaaa	aaaa

フローチャート：ある程度、自由にアレンジしてよい。



基礎コンピュータ工学第5章 機械語プログラミング 5 / 8

JNM (Jump on Not Minus) 命令

S フラグが 0 なら（正かゼロなら）ジャンプする。

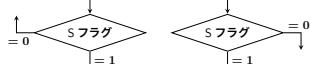
フラグ：変化しない。

ニーモニック： JNM EA (if($S=0$) PC ← EA)

命令フォーマット：2バイトの長さを持つ。

第1バイト		第2バイト	
OP	GR XR		
1011 ₂	11 ₂ XR	aaaa	aaaa

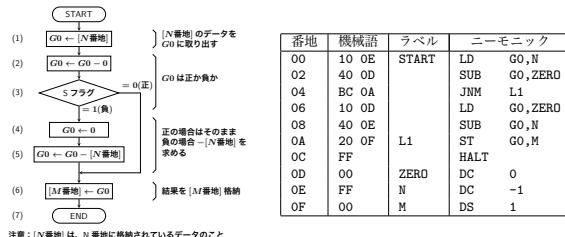
フローチャート：ある程度、自由にアレンジしてよい。



基礎コンピュータ工学第5章 機械語プログラミング 6 / 8

条件判断の例 (JNM 使用)

絶対値を求めるプログラム (例題5-1の改良)



- JNM を使用したほうが短くなる。

まとめ

●学んだこと

必須ではないがあると便利な JNZ, JNC, JNM 命令

●条件判断の演習

1. プログラムの作成手順を再度確認

- (1) フローチャートを描く。
- (2) フローチャートを基にニードルニックを書く。
- (3) アドレスを決める。
- (4) 機械語を作る。

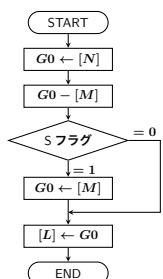
2. 演習 (宿題)

- (1) N番地の値がゼロなら M番地にゼロを、そうでなければ M番地に1を格納するプログラム

- LD 命令はフラグを変化させないので...
 - 前回の「条件判断2」のパターンを利用
- (2) N番地の値と M番地の値で、大きい方を L番地に格納するプログラム
- 値は符号付きの数値とする。
 - 比較は引き算ができる。

大小比較 (CMP, JNM 命令を使用して改良)

M と N 大きい方を選択して L に格納する。



	LD	GO, N
	CMP	GO, M
	JNM	L1
L1	LD	GO, M
N	ST	GO, L
M	HALT	
L	DS	1

まとめ

学んだこと

- ループの作り方
 - Java の do-while に似たタイプ
 - Java の while に似たタイプ
- 比較命令 (CMP)
 - 比較だけするときには役立つ。
 - 引き算をしてみてフラグだけ変化する。

演習 (do-while タイプのループ)

- 掛け算プログラム : N 番地のデータと、 M 番地のデータのかけ算を計算し L 番地に格納するプログラム
- データはどれも符号なし整数とする。
- 繰り返し処理 1 のフローチャートを参考に作る。

まとめ

演習 (while タイプのループ)

- 掛け算プログラム : N 番地のデータと、 M 番地のデータのかけ算を計算し L 番地に格納するプログラム
- データはどれも符号なし整数とする。
- 繰り返し処理 2 のフローチャートを参考に作る。

演習 (CMP 命令)

- 割り算プログラム : M 番地のデータを N 番地のデータで割り、 商を K 番地、 余りを L 番地に格納するプログラム
- データはどれも符号なし整数とする。
- 割り算は引き算の繰り返しでできる。

**基礎コンピュータ工学
第5章 機械語プログラミング
(パート8：シフト命令)**

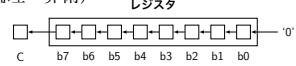
<https://github.com/tctsigemura/TecTextBook>

本スライドの入手：

基礎コンピュータ工学第5章 機械語プログラミング 1 / 11

シフト（桁ずらし）命令

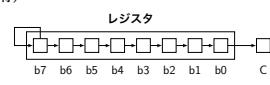
- データの2進数を左右に桁移動する命令のこと。
- TeCは4種類（実質は3種類）の命令を持っている。
- 左シフト（論理・算術）



- 右シフト（論理）



- 右シフト（算術）

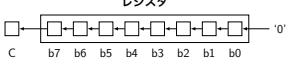


基礎コンピュータ工学第5章 機械語プログラミング

2 / 11

SHLA (Shift Left Arithmetic) 命令

左算術（算術=Arithmetic）シフト命令。
レジスタの値を左に1ビットずらす。（シフトする）

レジスタ


C フラグ 上の図のように変化する。
S フラグ 結果が負なら1, それ以外は0になる。
Z フラグ 結果がゼロなら1, それ以外は0になる。

フローチャート：Javaのシフト演算子を流用する。

```

    graph TD
        A[GR ← GR << 1] --> B[ ]
    
```

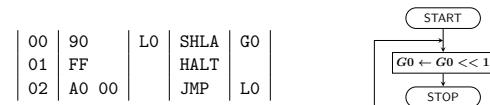
基礎コンピュータ工学第5章 機械語プログラミング 3 / 11

ニーモニック：SHLA GR

命令フォーマット：1バイトの長さを持つ。

第1バイト	
OP	GR XR
1001 ₂	GR 00 ₂

例題5-4：SHLA命令を実行して確かめる。（イルミネーション？）
(次のプログラムをG0に何かデータをセットしたあとでG0を表示したまま実行する。)



注：左シフトは×2を計算している。

基礎コンピュータ工学第5章 機械語プログラミング

4 / 11

SHLL (Shift Left Logical) 命令

左論理（論理=Logical）シフト命令。
レジスタの値を左に1ビットずらす。（シフトする）
(SHLL命令とSHLA命令の動作は全く同じ。)

フラグ SHLAと同じ

フローチャート：SHLAと同じ

ニーモニック：SHLL GR

命令フォーマット：1バイトの長さを持つ。

第1バイト	
OP	GR XR
1001 ₂	GR 01 ₂

基礎コンピュータ工学第5章 機械語プログラミング 5 / 11

左シフトを用いた×2計算

符号なし数の×2	符号付き数の×2
0000 0001 (1)	1111 1111 (-1)
0000 0010 (2)	1111 1110 (-2)
0000 0100 (4)	1111 1100 (-4)
0000 1000 (8)	1111 1000 (-8)
0001 0000 (16)	1111 0000 (-16)
0010 0000 (32)	1110 0000 (-32)
0100 0000 (64)	1100 0000 (-64)
1000 0000 (128)	1000 0000 (-128)
0000 0000 (ERR)	0000 0000 (ERR)

SHLL命令はこちら用

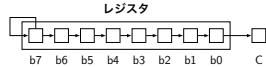
SHLA命令はこちら用

基礎コンピュータ工学第5章 機械語プログラミング

6 / 11

SHRA (Shift Right Arithmetic) 命令

右算術（算術 = Arithmetic）シフト命令。
レジスタの値を右に 1 ビットずらす。（シフトする）



フラグ SHLA と同じ

フローチャート：Java のシフト演算子を流用する。



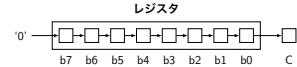
命令フォーマット：1 バイトの長さを持つ。

第 1 バイト	
OP	GR XR
1001 ₂	GR 10 ₂

注：SHRA は符号付き数の $\div 2$ を計算している。

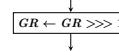
SHRL (Shift Right Logical) 命令

右論理（論理 = Logical）シフト命令。
レジスタの値を右に 1 ビットずらす。（シフトする）



フラグ SHLA と同じ

フローチャート：Java のシフト演算子を流用する。



命令フォーマット：1 バイトの長さを持つ。

第 1 バイト	
OP	GR XR
1001 ₂	GR 11 ₂

注：SHRL は符号なし数の $\div 2$ を計算している。

右シフトを用いた $\div 2$ 計算（1）

符号なし数の $\div 2$	符号付き数の $\div 2$
1100 0000 (192)	1100 0000 (-64)
0110 0000 (96)	1110 0000 (-32)
0011 0000 (48)	1111 0000 (-16)
0001 1000 (24)	1111 1000 (-8)
0000 1100 (12)	1111 1100 (-4)
0000 0110 (6)	1111 1110 (-2)
0000 0011 (3)	1111 1111 (-1)
0000 0001 (1)	1111 1111 (-1)
0000 0000 (0)	1111 1111 (-1)

SHRL 命令を使用する

SHRA 命令を使用する

右シフトを用いた $\div 2$ 計算（2）

符号付き正数の $\div 2$	符号付き負数の $\div 2$
0100 0000 (64)	1100 0000 (-64)
0010 0000 (32)	1110 0000 (-32)
0001 0000 (16)	1111 0000 (-16)
0000 1000 (8)	1111 1000 (-8)
0000 0100 (4)	1111 1100 (-4)
0000 0010 (2)	1111 1110 (-2)
0000 0001 (1)	1111 1111 (-1)
0000 0000 (0)	1111 1111 (-1)
0000 0000 (0)	1111 1111 (-1)

SHRA 命令使用

SHRL 命令使用

まとめ

学んだこと

- TeC のシフト命令は 1 ビットシフトする。
- TeC は 4 種類（実質は 3 種類）のシフト命令を持っている。
- シフト命令はイルミネーション（？）に使用できる。
- 左シフト（論理・算術）は、
符号付き・なし兼用の $\times 2$ 計算に使用できる。
- 右シフト（論理）は、符号なし数の $\div 2$ 計算に使用できる。
- 右シフト（算術）は、符号付き数の $\div 2$ 計算に使用できる。

演習

- ビットの右回転（例題 5-5 を参考に）
- シフト命令を使用した「 $\times 7$ の計算」（例題 5-6 を参考に）
- シフト命令を使用した「 $\div 4$ の計算」
- シフト命令を使用した「 $\times 1, 5$ の計算」

基礎コンピュータ工学 第5章 機械語プログラミング (パート9：論理演算命令)

<https://github.com/tctsigemura/TecTextBook>

本スライドの入手：

基礎コンピュータ工学第5章 機械語プログラミング 1 / 12

論理演算

「2.8 コンピュータの基本回路」で学んだ論理演算を再確認。

論理積 (AND)	論理和 (OR)																								
$X = A \cdot B$ AND の論理式	$X = A + B$ OR の論理式																								
<table border="1" style="width: 100px; text-align: center;"><tr><th>入力</th><th>出力</th></tr><tr><td>A B</td><td>X</td></tr><tr><td>0 0</td><td>0</td></tr><tr><td>0 1</td><td>0</td></tr><tr><td>1 0</td><td>0</td></tr><tr><td>1 1</td><td>1</td></tr></table>	入力	出力	A B	X	0 0	0	0 1	0	1 0	0	1 1	1	<table border="1" style="width: 100px; text-align: center;"><tr><th>入力</th><th>出力</th></tr><tr><td>A B</td><td>X</td></tr><tr><td>0 0</td><td>0</td></tr><tr><td>0 1</td><td>1</td></tr><tr><td>1 0</td><td>1</td></tr><tr><td>1 1</td><td>1</td></tr></table>	入力	出力	A B	X	0 0	0	0 1	1	1 0	1	1 1	1
入力	出力																								
A B	X																								
0 0	0																								
0 1	0																								
1 0	0																								
1 1	1																								
入力	出力																								
A B	X																								
0 0	0																								
0 1	1																								
1 0	1																								
1 1	1																								
AND の真理値表	OR の真理値表																								

排他的論理和 (XOR)	否定 (NOT)																				
$X = A \oplus B$ XOR の論理式	$X = \bar{A}$ NOT の論理式																				
<table border="1" style="width: 100px; text-align: center;"><tr><th>入力</th><th>出力</th></tr><tr><td>A B</td><td>X</td></tr><tr><td>0 0</td><td>0</td></tr><tr><td>0 1</td><td>1</td></tr><tr><td>1 0</td><td>1</td></tr><tr><td>1 1</td><td>0</td></tr></table>	入力	出力	A B	X	0 0	0	0 1	1	1 0	1	1 1	0	<table border="1" style="width: 100px; text-align: center;"><tr><th>入力</th><th>出力</th></tr><tr><td>A</td><td>X</td></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	入力	出力	A	X	0	1	1	0
入力	出力																				
A B	X																				
0 0	0																				
0 1	1																				
1 0	1																				
1 1	0																				
入力	出力																				
A	X																				
0	1																				
1	0																				
XOR の真理値表	NOT の真理値表																				

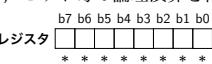
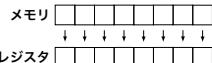
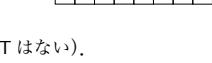
論理回路記号	論理回路記号
 A  B → X	 A  B → X

論理回路記号	論理回路記号
 A  B → X	 A → X

基礎コンピュータ工学第5章 機械語プログラミング 2 / 12

論理演算命令

論理演算を行う TeC の命令
8 ビットデータを単位に、ビット毎の論理演算を行う。

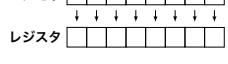
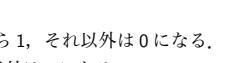
b7 b6 b5 b4 b3 b2 b1 b0							
レジスタ							
*	*	*	*	*	*	*	*
メモリ							
↓	↓	↓	↓	↓	↓	↓	↓
レジスタ							

次の 3 種類がある (NOT はない)。

- 論理積 (AND)
- 論理和 (OR)
- 排他的論理和 (XOR)

基礎コンピュータ工学第5章 機械語プログラミング 3 / 12

AND (Logical AND) 命令 (論理積)

b7 b6 b5 b4 b3 b2 b1 b0
レジスタ 
メモリ 
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
レジスタ 

C フラグ 常に 0 になる。
S フラグ 結果が負 (MSB が 1) なら 1、それ以外は 0 になる。
Z フラグ 結果がゼロなら 1、それ以外は 0 になる。

ニーモニック : AND GR,EA (GR ← GR & [EA])
命令フォーマット : 2 バイトの長さを持つ。

第1バイト	第2バイト
OP	GR XR
0110 ₂	aaaa aaaa

基礎コンピュータ工学第5章 機械語プログラミング 4 / 12

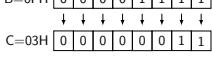
フローチャート : Java の演算子を流用する。

```

    ↓
    [GR ← GR & [EA]]
    ↓
  
```

使用例 : A 番地のデータと B 番地のデータのビット毎の論理積を計算し、C 番地に格納するプログラムの例を示す。

番地	機械語	ラベル	ニーモニック
00	10 07		LD G0, A
02	60 08		AND G0, B
04	20 09		ST G0, C
06	FF		HALT
07	63	A	DC 63H
08	0F	B	DC 0FH
09	00	C	DS 1

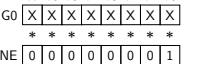
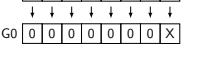
b7 b6 b5 b4 b3 b2 b1 b0
 A=63H 
 * * * * * * * *
 B=0FH 
 ↓ ↓ ↓ ↓ ↓ ↓
 C=03H 

基礎コンピュータ工学第5章 機械語プログラミング 5 / 12

AND 命令の応用 (1)

特定のビットがゼロか判定する。
AND の結果が 00₁₆ かどうかで判断できる。
次は最下位ビット (LSB) を調べる例。

ラベル	ニーモニック
...	...
AND G0, ONE	AND G0, ONE
JZ L1	JZ L1
...	...
L1	...
...	...
ONE	ONE DC 01H

b7 b6 b5 b4 b3 b2 b1 b0
G0 
* * * * * * * *
ONE 
↓ ↓ ↓ ↓ ↓ ↓
G0 

LSB : Least Significant Bit (最下位ビットのこと、P.10 参照)

基礎コンピュータ工学第5章 機械語プログラミング 6 / 12

AND 命令の応用 (2)

特定のビットを右詰めで取り出す。
(b_3, b_2 を右詰めで取り出す。)

ラベル	ニーモニック
...	
AND	GO,MSK
SHRL	GO
SHRL	GO
...	
MSK	DC OCH



OR (Logical OR) 命令 (論理和)

レジスタ値とメモリ値のビット毎の論理和をレジスタに格納する。

C フラグ 常に 0 になる。

S フラグ 結果が負 (MSB が 1) なら 1, それ以外は 0 になる。

Z フラグ 結果がゼロなら 1, それ以外は 0 になる。

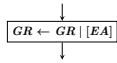
ニーモニック : OR GR,EA (GR ← GR | [EA])

命令フォーマット : 2 バイトの長さを持つ。

第1バイト	第2バイト
OP 0111 ₂	GR XR GR XR aaaa aaaa

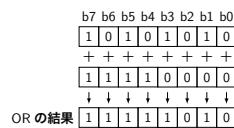
MSB : Most Significant Bit (最上位ビットのこと, P.10 参照)

フローチャート : Java の演算子を流用する。



応用 : G0 の上位 4 ビットを全部 1 にする。

ラベル	ニーモニック
...	
LD	GO,DATA
OR	GO,MSK
...	
DATA	DC OAAH
MSK	DC OFOH



16進数の表記 : ラベルと区別が付くように注意 ! (P.44 参照)

XOR (Logical XOR) 命令 (排他的論理和)

レジスタ値とメモリ値のビット毎の排他的論理和をレジスタに格納する。

C フラグ 常に 0 になる。

S フラグ 結果が負 (MSB が 1) なら 1, それ以外は 0 になる。

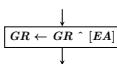
Z フラグ 結果がゼロなら 1, それ以外は 0 になる。

ニーモニック : XOR GR,EA (GR ← GR ^ [EA])

命令フォーマット : 2 バイトの長さを持つ。

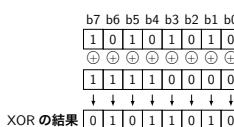
第1バイト	第2バイト
OP 1000 ₂	GR XR GR XR aaaa aaaa

フローチャート : Java の演算子を流用する。



応用 : G0 上位 4 ビットの 1/0 を入れ替える (ビット反転する)。

ラベル	ニーモニック
...	
LD	GO,DATA
XOR	GO,MSK
...	
DATA	DC OAAH
MSK	DC OFOH



まとめ

学んだこと

- ビット毎の論理演算命令
- TeC は次の演算命令を持っている。
 - 論理積 (AND) 命令
 - 論理和 (OR) 命令
 - 排他的論理和 (XOR) 命令

演習

- TeC には NOT 命令が無い。
NOT 命令があったとすると、どんな計算をする命令になるか？
- NOT 命令の代用となる命令を考えなさい。
- 値が奇数か偶数か判定する方法を考えなさい。
- 8 で割った余りを計算する方法を考えなさい。

**基礎コンピュータ工学
第5章 機械語プログラミング
(パート10 : アドレッシングモード)**

<https://github.com/tctsigemura/TecTextBook>

本スライドの入手：

基礎コンピュータ工学第5章 機械語プログラミング

アドレッシングモード

LD, ST, ADD, SUB, CMP, AND, OR, XOR, JMP, JZ, JC, JM, JNZ, JNC, JNM の命令フォーマットは同じだった。

第1バイト		第2バイト	
OP	GR XR	OP	GR XR
0001	00 01	aaaa	aaaa

これまで、XR フィールドは 00_2 にしてきた。
XR フィールドは、メモリデータのアドレス計算方法を決めるアドレッシングモードを指定する。

XR	意味
00_2	ダイレクトモード (直接モード)
01_2	G1 インデクスドモード (G1 指標モード)
10_2	G2 インデクスドモード (G2 指標モード)
11_2	イミディエイトモード (即値モード)

基礎コンピュータ工学第5章 機械語プログラミ

2 / 11

ダイレクト（直接）モード

これまで使用してきたアドレッシングモードはダイレクトモード

- 実効アドレス (EA : Effective Address)
実効アドレス = 第2バイトの内容
- XR フィールド = 00_2
- ニーモニック例

LD	G0, A
ST	G0, B
- フローチャート例
 

```

      G0 ← [A]
      ↓
      [B] ← G0
    
```

実効アドレス = 命令の操作対象となるメモリアドレスのこと。

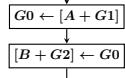
基礎コンピュータ工学第5章 機械語プログラミ

3 / 11

インデクスド（指標）モード

G1, G2 が配列データをアクセスするために使用できる。
(G0, SP は使用できないので注意！！)

- 実効アドレス (EA : Effective Address)
実効アドレス = 第2バイトの内容 + G1 の内容
実効アドレス = 第2バイトの内容 + G2 の内容
(この時、G1, G2 はインデクスレジスタと呼ばれる。)
- XR フィールド ($G1=01_2, G2=10_2$)
- ニーモニック例

LD	G0, A, G1
ST	G0, B, G2
- フローチャート例
 

```

      G0 ← [A + G1]
      ↓
      [B + G2] ← G0
    
```

基礎コンピュータ工学第5章 機械語プログラミ

4 / 11

- 機械語の例 (LD 命令) LD G0, A, G1

第1バイト		第2バイト	
OP	GR XR	OP	GR XR
0001	00 01	aaaa	aaaa
- 機械語の例 (ST 命令) ST G0, A, G2

第1バイト		第2バイト	
OP	GR XR	OP	GR XR
0010	00 10	aaaa	aaaa
- 機械語の例 (レジスタ) LD G2, A, G1

第1バイト		第2バイト	
OP	GR XR	OP	GR XR
0001	10 01	aaaa	aaaa

基礎コンピュータ工学第5章 機械語プログラミ

5 / 11

インデクスマードの使用例

配列 A の I 番目のデータ (A[I]) を X にコピーする。

第1バイト		第2バイト	
OP	GR XR	OP	GR XR
0001	00 01	0000	1000

番地	機械語	ラベル	ニーモニック
00	14 07		LD G1, I
02	11 08		LD G0, A, G1
04	20 0B		ST G0, X
06	FF		HALT
07	01	I	DC 1
08	08	A	DC 8
09	02		DC 2
0A	0A		DC 10
0B	00	X	DS 1

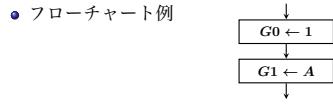
基礎コンピュータ工学第5章 機械語プログラミ

6 / 11

イミディエイト（即値）モード

命令の第2バイトがデータそのものになる。
ZERO, ONE 等のデータを準備しなくても即値を使用できる。
(ST命令やジャンプ命令では使用できない。)

- 実効アドレス (EA : Effective Address)
実効アドレス = 第2バイト
- XR フィールド = 11_2
- ニーモニック例
LD G0,#1
LD G0,#A
 $\#A$ は、 A の内容ではなく、 A のアドレスの意味！！



- 機械語の例 (データの1) LD G0,#1

第1バイト		第2バイト	
OP	GR	XR	
0001	00	11	0000 0001

- 機械語の例 (アドレス A) LD G1,#A

第1バイト		第2バイト	
OP	GR	XR	
0001	01	11	aaaa aaaa

- イミディエイトなし・ありの比較

```

    ...
    LD G0,ZERO
    ADD G0,ONE
    ...
    ZERO DC 0
    ONE DC 1
  
```

```

    ...
    LD G0,#0
    ADD G0,#1
    ...
  
```

イミディエイトモードの使用例

A番地のデータに1を加えB番地に格納する。

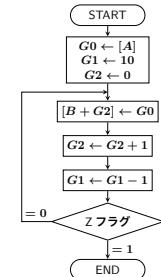
第1バイト		第2バイト	
OP	GR	XR	
0011	00	11	0000 0001

番地	機械語	ラベル	ニーモニック
00	10 07		LD G0,A
02	33 01		ADD G0,#1
04	20 08		ST G0,B
06	FF		HALT
07	05	A	DC 5
08	00	B	DS 1

アドレッシングモードの使用例

A番地のデータでB番地からの10バイトの配列を初期化する。

番地	機械語	ラベル	ニーモニック
00	10 11		LD G0,A
02	17 0A		LD G1,#10
04	1B 00		LD G2,#0
06	22 12	LOOP	ST G0,B,G2
08	3B 01		ADD G2,#1
0A	47 01		SUB G1,#1
0C	A4 10		JZ STOP
0E	A0 06		JMP LOOP
10	FF	STOP	HALT
11	AA	A	DC OAAH
12	00 00	B	DS 10
14	00 00		
16	00 00		
18	00 00		
1A	00 00		



まとめ

学んだこと

- 「実効アドレス (EA)」 = 「データのメモリアドレス」
- 「アドレッシングモード」 = 「実効アドレスの計算方法」
- TeC では次のアドレッシングモードが使用できる。
 - ダイレクト (直接) モード
「命令の第2バイトの内容」が実効アドレス
 - インデクスド (指標) モード
「命令の第2バイトの内容 + レジスタの内容」が実効アドレス
(アドレス計算には、G1, G2 レジスタだけが使用できる。)
 - イミディエイト (即値) モード
「命令の第2バイト」が実効アドレス

演習

- イミディエイトモードの ST 命令を TeC で実行してみる。
- A 番地からの 5 バイトのデータの和を B 番地に求める。
- A 番地からの 5 バイトのデータを B 番地から 5 バイトにコピーする。

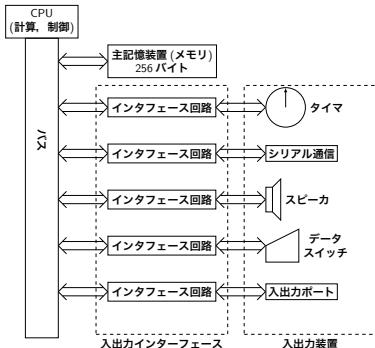
**基礎コンピュータ工学
第5章 機械語プログラミング
(パート11：入出力命令)**

<https://github.com/tctsigemura/TecTextBook>

本スライドの入手：

基礎コンピュータ工学第5章 機械語プログラミング 1 / 10

TeCの構成を思い出してみる



入出力装置を操作するにはどうしたら良いのか？

基礎コンピュータ工学第5章 機械語プログラミング 2 / 10

メモリ領域とI/O領域

メモリ領域とは別に、
入出力インターフェースを配置したI/O領域がある。

メモリ領域	
番地	内容
00	
01	RAM
...	自由に使用可能
DA	
DB	
DC	
...	システム領域
FF	

LD, ST, ADD... 命令で使用
(プログラムもここに置いた)

I/O領域	
番地	内容
0	Data-Sw/b0:Beep
1	Data-Sw/b0:Spk
2	SIO-Data
3	SIO-Clt/Stat
...	...
F	空き/空き

IN, OUT命令で使用

基礎コンピュータ工学第5章 機械語プログラミング 3 / 10

I/Oマップ (I/O領域内の配置を書いたもの)

I/O領域の内容を表すI/Oマップがある。

番地	Read	Write
0	データスイッチ	ブザー
1	データスイッチ	スピーカ
2	SIO 受信データ	SIO 送信データ
3	SIO ステータス	SIO コントロール
4	タイマ現在値	タイマ周期
5	タイマステータス	タイマコントロール
6	空き	INT3 コントロール
7	PIO 入力ポート	PIO 出力ポート
8	ADC CH0	空き
9	ADC CH1	空き
A	ADC CH2	空き
B	ADC CH3	空き
C	空き	PIO コントロール
D	空き	空き
E	空き	空き
F	空き	空き

基礎コンピュータ工学第5章 機械語プログラミング 4 / 10

IN (Input) 命令 (入力命令)

I/O領域からデータを入力(Read)し結果をレジスタに格納する。

フラグ：変化しない。

ニーモニック： IN GR,P (GR ← IO[P])

命令フォーマット：2バイトの長さを持つ。

第1バイト	OP	GR	XR	第2バイト
1100 ₂	GR	00 ₂		pppp

フローチャート：平行四辺形の中に説明を書く。

 ↓
 動作の説明

基礎コンピュータ工学第5章 機械語プログラミング 5 / 10

IN (Input) 命令の使用例

データスイッチの値をデータランプに表示する。

番地	機械語	ラベル	ニーモニック
00	CO 00	START	IN GO,0
02	AO 00		JMP START

- 無限ループになっているので停止しない。
- IN命令はI/Oの0番地(データスイッチ)を読む。
- IN命令は即座に次の命令に進む(入力待はしない)。
- プログラムは全速力でループをまわる。
- GOを表示した状態でプログラムを実行すると、
データスイッチの値がデータランプに表示され続ける。

基礎コンピュータ工学第5章 機械語プログラミング 6 / 10

IN (Input) 命令の応用

入力したデータの合計を G0 に求める。

ラベル	ニーモニック
START	LD G0, #0
LOOP	IN G1, OOH
	ST G1, TMP
	ADD G0, TMP
	HALT
	JMP LOOP
TMP	DS 1

1. プログラムを入力
2. PC に実行開始番地をセット
(0 番地なら RESET でも良い)
3. G0 を表示した状態にする
4. データスイッチにデータをセット
5. RUN ボタンを押す
6. データ分, 4, 5 を繰り返す
7. データランプに合計表示中

OUT (Output) 命令 (出力命令)

I/O 領域ヘレジスタのデータを出力 (Write) する。

フラグ：変化しない。

ニーモニック： OUT GR,P (IO[P] ← GR)

命令フォーマット： 2 バイトの長さを持つ。

第1バイト		第2バイト
OP	GR XR	
1100 ₂	GR 11 ₂	0000 ₂ pppp

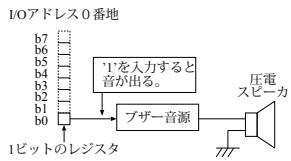
フローチャート： 平行四辺形の中に説明を書く (IN と同じ)。



OUT (Output) 命令の応用

データスイッチ (D0) の操作でブザーを鳴らしたり止めたりする。

- TeC のブザーの仕組み



- ブザーを鳴らすプログラム

番地	機械語	ラベル	ニーモニック
00	C0 00	START	IN G0, 0
02	C3 00		OUT G0, 0
04	A0 00		JMP START

ブザーが鳴り続けて困ったら RESET を押す。

まとめ

学んだこと

- 「入出力命令」 = 「IN 命令と OUT 命令」
- I/O 領域と I/O マップ
- IN 命令と応用
 - データスイッチの値をデータランプに表示する。
 - データスイッチから入力した値の合計を求める。
- OUT 命令と応用
 - ブザーを鳴らす。

演習

- 0 で終わるデータ列を入力し合計を X 番地に求める。
- データスイッチのビット 7 (D7) でブザーを制御する。

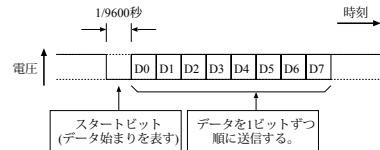
基礎コンピュータ工学 第5章 機械語プログラミング (パート15：シリアル入出力)

<https://github.com/tctsigemura/TecTextBook>

シリアル入出力 (SIO)

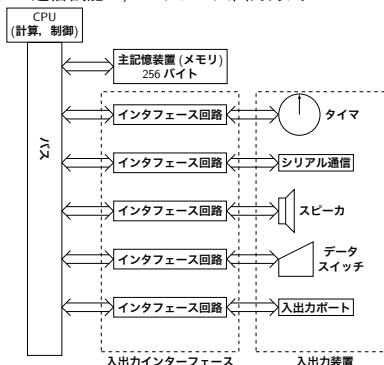
シリアル入出力 (Serial Input Output : SIO) (Serial = 直列)
PCでは使用されなくなってきたが組込み用のマイコン等では現役

- データを1ビットずつ送受信する方式
- 1本の信号線でデータを送信できる
- TeCの場合は送信用・受信用に2本使用している
- TeCの場合は1/9600秒を単位にしている
- この通信速度を9600baud(ボード)と言う



TeCのシリアル通信

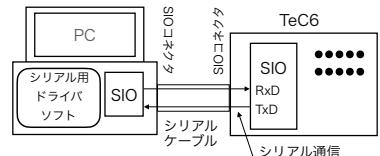
TeCのシリアル通信機能は、シリアル入出力方式



PCとシリアルケーブルで接続する

シリアル通信用のケーブルで接続する。

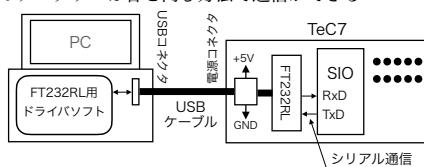
- 古いTeC (TeC6)で使用していた方式
- PCのSIO通信ハードウェアとTeCのSIOインターフェースを直接接続
- シリアル用ドライバがPC上のソフトにSIO通信機能を提供
- シリアル用ドライバはOSに標準で装備されていた
- 最近はPCがSIOコネクタを装備しなくなった



PCとUSBで接続する

PCとUSBケーブルで接続するとシリアル通信が可能になる。

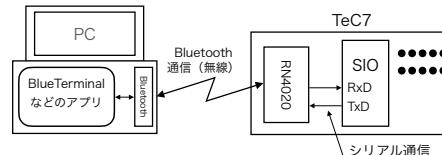
- USBケーブルで電源も供給される
- TeCが「USB-シリアル変換LSI(FT232RL)」を内蔵
- FT232RLとSIOインターフェース間だけがシリアル通信
- FT232RL用ドライバがPC上のソフトにSIO通信機能を提供
- FT232RL用ドライバはmacOSに標準で装備されている
- TeCのプログラムは昔と同じ方法で通信ができる



PCとBluetoothで接続する

PCとBluetooth(無線)で接続しシリアル通信を行う。

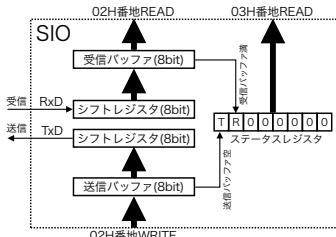
- TeCが「Bluetooth-シリアル変換LSI(RN4020)」を内蔵
- RN4020とSIOインターフェース間だけがシリアル通信
- Macには専用のアプリ(BlueTerminal)が必要
- USB接続とBluetooth接続は同時に使用できる
- TeCのプログラムは昔と同じ方法で通信ができる



シリアル入出力回路の仕組み

TeC の SIO インタフェース回路の仕組み

- 受信したシリアルデータはシフトレジスタで 8bit まとめる
- 8bit まとまつたら受信バッファに移し、次の受信に備える
- シフトレジスタが空になったら送信バッファからデータが移される
- シフトレジスタで 1bit ずつに分解して出力する



基礎コンピュータ工学第5章 機械語プログラミ

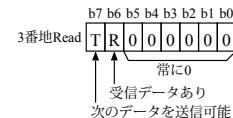
7 / 12

I/O ポート

TeC のプログラムが SIO インタフェース回路をアクセスする I/O 番地

番地	read(IN 命令)	write(OUT 命令)
02H	受信バッファ	送信バッファ
03H	ステータス	コントロール

- 02H 番地から受信・送信バッファの読み書きができる
- 03H 番地で受信・送信バッファの状態を知ることができる
- ステータスの意味は次の通り



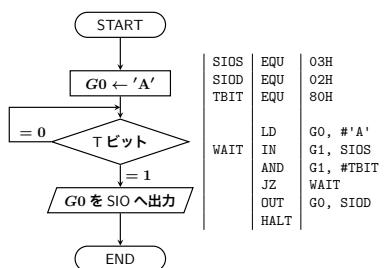
基礎コンピュータ工学第5章 機械語プログラミ

8 / 12

シリアル出力プログラム

SIO へ 1 文字出力するプログラムの例

- ステータスの T ビットが 1 になるまで待つ
- 送信バッファに送信データを書き込む



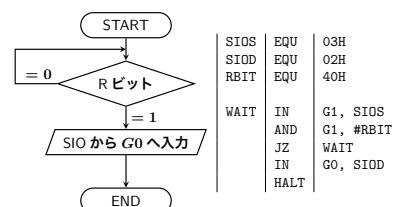
基礎コンピュータ工学第5章 機械語プログラミ

9 / 12

シリアル入力プログラム

SIO から 1 文字入力するプログラムの例

- ステータスの R ビットが 1 になるまで待つ
- 受信バッファから受信データを読み取る



基礎コンピュータ工学第5章 機械語プログラミ

10 / 12

シリアル通信アプリ

Mac 側で SIO 通信に使用する通信プログラム (screen コマンド)

- SIO から受信したデータを文字コードとみなし表示
- 押されたキーに対応する文字コードを SIO へ送信

```

1. TeC と Mac を USB ケーブルで接続する
2. ターミナルで次のように操作する

% screen /dev/cu.usb[TAB] [Enter] # TAB, Enter を順に入力する
... 通信画面になる ...
[^A][^\$]y                         # Ctrl-A, Ctrl-\$, y で終了
%
  
```

基礎コンピュータ工学第5章 機械語プログラミ

11 / 12

まとめ

学んだこと

- シリアル入出力の仕組み
- TeC のシリアル入出力回路の仕組み
- シリアル入出力プログラムの作り方
- Mac の通信プログラム (screen)

演習

- 「例題 5-9 'A'～'Z' の文字を出力」を試しなさい。
- 「例題 5-10 echo プログラム」を試しなさい。
- 受信した文字を「シーザー暗号」にして送り返すプログラムを作りなさい。（文字は A-Z の 26 種類だけ、受信した文字を辞書順で 3 文字シフトする）

基礎コンピュータ工学第5章 機械語プログラミ

12 / 12