To-Be エンジニア試験公式テキスト

情報 I

（情報の基礎・プロセッサ・プログラミング）

To-Be エンジニア試験企画委員会　編著

コガク

目　次

1章 情報の基礎 ..... 1～52

1 情報表現（1と0で何を表すか？） 2

1.1　情報量（ビット，ワード）..... 2
　1.1.1　情報量の考え方 2
　1.1.2　2 進表示と 16 進表示 3
　1.1.3　n ビットの情報量 3
　1.1.4　情報量の単位 4
1.2　基数変換（2 進 / 10 進など）..... 4
　1.2.1　基数とは（n 進数）..... 5
　1.2.2　2 進→ n 進変換 6
　1.2.3　n 進→ 2 進変換 6
　1.2.4　BCD コード（2 進化 10 進数）..... 7
1.3　文字の表現（文字コード）..... 8
　1.3.1　文字コードとは 8
　1.3.2　アルファベットと数字 9
　1.3.3　日本語の符号 11
　1.3.4　文字コードの標準化 11
1.4　ディジタル化（標本化，量子化，符号化など）..... 12
　1.4.1　A/D 変換（標本化，量子化）..... 12
　1.4.2　標本化定理 12
　1.4.3　音の符号化 13
　1.4.4　映像の符号化 14

2 論理演算（1と0を自由に操る。ワードとしての処理）..... 15

2.1　論理和と論理積（特定のビットの ON/OFF）..... 15
　2.1.1　2 値論理の基本 15
　2.1.2　論理積 16
　2.1.3　論理和 17

2.1.4　真理値表の実現 17
2.2　排他的論理和（特定のビットの反転）..... 19
2.2.1　やや複雑な論理演算 19
2.2.2　排他的論理和の特徴 21
2.2.3　排他的論理和の実現 22
2.2.4　MIL 記号 23
2.3　フラグ（ゼロフラグ，キャリーフラグ）..... 24
2.3.1　フラグとは 24
2.3.2　情報の記憶原理 25
2.3.3　代表的なフラグ 26
2.3.4　やや特殊なフラグ 26
2.4　ビットシフト（論理シフトと算術シフト）..... 27
2.4.1　2 進数シフトの意味 27
2.4.2　シフトとローテイト 27
2.4.3　論理シフト 28
2.4.4　算術シフト 28

3 数値演算（ディジタル演算特有の基礎知識） 30

3.1　表現可能な数値範囲（整数型変数）..... 30
3.1.1　符号なし整数型変数の範囲 30
3.1.2　2 進数の加算 31
3.1.3　2 進数の乗算 31
3.1.4　型が混在する場合 32
3.2　負数表現（2 の補数）..... 32
3.2.1　負の数の表し方 32
3.2.2　反対符号数値の求め方 33
3.2.3　符号付き整数型変数の範囲 34
3.2.4　整数の演算 34
3.3　小数表現（固定小数点，浮動小数点）..... 35
3.3.1　固定小数 35
3.3.2　浮動小数 35
3.3.3　正規表現 36
3.3.4　非正規表現 37
3.4　演算精度（誤差の種類と対策）..... 37
3.4.1　桁落ち誤差 37
3.4.2　丸め誤差 38
3.4.3　オーバーフロー 38
3.4.4　情報落ち誤差 39

4 符号の応用（1と0による情報のさまざま利用に関する用語） 40

4.1　誤り検出（パリティ，チェックサム，CRC など）..... 40
4.1.1　パリティ 40

4.1.2　チェックサム 42
4.1.3　CRC 42
4.1.4　ハッシュ 43
4.2　誤り訂正（リードソロモン符号など） 44
4.2.1　誤り訂正の原理 44
4.2.2　ハミング符号 44
4.2.3　リードソロモン 45
4.2.4　畳み込み 45
4.3　データ圧縮（ランレングス，ハフマン符号など） 45
4.3.1　ランレングス符号 45
4.3.2　ハフマン符号 46
4.3.3　時間 / 空間と周波数 46
4.3.4　DCT（離散コサイン変換） 47
4.4　身近な符号（バーコード，QR コードなど） 49
4.4.1　符号とは 49
4.4.2　PCM 49
4.4.3　バーコード 49
4.4.4　QR コード 51

2章　プロセッサ ..... 53～104

1　コンピュータの原理
（ストアドプログラム方式の基本を理解） 54

1.1　5 大機能（入力 / 出力 / 記憶 / 演算 / 制御） 54
1.1.1　ハードウェアシステム（ゲート／フリップフロップ） 54
1.1.2　入出力ポート（論理の分離） 55
1.1.3　演算機能の集約（レジスタ，ALU） 55
1.1.4　ノイマン型 CPU（機械語による制御） 56
1.2　アーキテクチャ（CISC/RISC など） 57
1.2.1　CPU の種類（用途） 57
1.2.2　レジスタ構成 57
1.2.3　演算とフラグ 58
1.2.4　CPU の例（自作） 58
1.3　機械語（命令語の構成，アドレシングなど） 59
1.3.1　機械語の定義（1 と 0 による定義） 59
1.3.2　命令の分類（移動 / 演算） 60
1.3.3　直接 / 間接アドレシング 60
1.3.4　絶対 / 相対アドレシング 62
1.4　割込み（内部 / 外部） 63
1.4.1　割込みと例外 63
1.4.2　外部割込みと内部割込み 64

1.4.3　割込みの仕組み 65
1.4.4　割込みの有効性 66

2 高速化技術（代表的な高速化技術） 67

2.1　キャッシュ（キャッシュのしくみ） 67
2.1.1　命令のフェッチと実行 67
2.1.2　プログラム実行の特徴 68
2.1.3　キャッシュの原理 68
2.1.4　キャッシュの構成 69
2.2　パイプライン（パイプラインのしくみ） 71
2.2.1　命令の解読から実行まで 71
2.2.2　パイプラインとは 71
2.2.3　命令形式の統一 72
2.2.4　RISC の特徴 73
2.3　ハーバードアーキテクチャ（バスの分離） 73
2.3.1　命令実行とデータ演算 73
2.3.2　同時処理による高速化 74
2.3.3　空間の分離 74
2.3.4　キャッシュ / パイプラインの分離 75
2.4　マルチプロセッサ（マルチコアなど） 75
2.4.1　並列処理による高速化 76
2.4.2　マルチプロセッサ 76
2.4.3　マルチコア 76
2.4.4　タスクとスレッド 77

3 周辺回路（CPUを支える主な周辺回路） 78

3.1　タイマ（インタバルタイマ） 78
3.1.1　タイマモジュール 78
3.1.2　クロック 79
3.1.3　カウンタ 80
3.1.4　PWM 81
3.2　シリアル通信（UART，I^2C，SPI） 82
3.2.1　シリアルインターフェース 82
3.2.2　RS232C と UART 83
3.2.3　I^2C 84
3.2.4　SPI 85
3.3　I/O ポート（GPIO，双方向制御） 85
3.3.1　汎用 I/O ポート 86
3.3.2　双方向のしくみ 86
3.3.3　電気的特性 87
3.3.4　特殊なポート 88
3.4　DMA（DMA の用途） 89

3.4.1　DMA とは 89
3.4.2　DMA の仕組み 89
3.4.3　高速データ転送 90
3.4.4　割込みとの連動 91

4 周辺装置（パソコン等に接続される周辺装置） 92

4.1　入力装置（タッチスクリーン，マウスなど） 92
4.1.1　マウスとキーボード 92
4.1.2　スキャナ 93
4.1.3　タッチパネル 94
4.1.4　マイク 94
4.2　出力装置（ディスプレイ，プリンタなど） 95
4.2.1　ディスプレイ 95
4.2.2　プロジェクタ 95
4.2.3　プリンタ 96
4.2.4　スピーカ 97
4.3　補助記憶装置（ハードディスク，DVD など） 98
4.3.1　ハードディスク 98
4.3.2　SSD 98
4.3.3　CD/DVD 99
4.3.4　カード 100
4.4　主なインターフェース（USB など） 101
4.4.1　PC インターフェース 101
4.4.2　USB 101
4.4.3　IEEE1394 102
4.4.4　Bluetooth 103

3章 プログラミング ..... 105～150

1 データ構造とアルゴリズム（データ構造と考え方） 106

1.1　代表的なデータ構造（配列 / リスト / 木構造） 106
1.1.1　流れ図（フローチャート） 106
1.1.2　配列とリスト 107
1.1.3　ハッシュ 108
1.1.4　木構造 109
1.2　スタックとキュー（FIFO/LIFO の基本操作） 110
1.2.1　スタックとは 110
1.2.2　スタックの応用 111

1.2.3　キューとは 111
1.2.4　キューの応用 112
1.3　代表的なアルゴリズム（ソート / 探索 / 再帰など） 113
1.3.1　アルゴリズムとは 113
1.3.2　ソート 113
1.3.3　探索 116
1.3.4　再帰 118
1.4　流れ図（フローチャート） 118
1.4.1　フローチャートとは 118
1.4.2　フローチャートの書き方 119
1.4.3　よいフローチャート 119
1.4.4　構造化フローチャート 120

2 プログラミング（信頼性や移植性/保守性を考えたプログラミング） 121

2.1　プログラム構造（サブルーチン，モジュールの独立性） 121
2.1.1　ソフトウェアの製造 121
2.1.2　モジュールの独立性 121
2.1.3　移植性と保守性 122
2.1.4　プログラム構造表記 122
2.2　ソースコードの品質（プログラミング作法 / コーディング規約） 123
2.2.1　ソースコード品質とは 123
2.2.2　コーディング規約 124
2.2.3　プログラミング作法 124
2.2.4　品質向上のために 125
2.3　Web プログラミング（クライアント / サーバシステム，クラウド） 125
2.3.1　Web プログラミングとは 125
2.3.2　クライアントサイド 126
2.3.3　サーバサイド 126
2.3.4　新しい流れ 127
2.4　ソフトウェア工学（構造化など） 128
2.4.1　ソフトウェア開発の問題点 128
2.4.2　ソフトウェア工学の考え方 128
2.4.3　構造化の重要性 128
2.4.4　新しい流れ 129

3 主なプログラミング言語（各種言語に関する基礎知識） 131

3.1　言語の種類（手続き型 / オブジェクト指向 / スクリプト言語） 131
3.1.1　プログラミング言語とは 131
3.1.2　高水準言語と低水準言語 131
3.1.3　インタプリタ型 132
3.1.4　スクリプト言語 132
3.2　アセンブリ言語（擬似命令 / アドレシング / レジスタ / スタッフ） 133

3.2.1　アセンブリ言語とは 133
3.2.2　疑似命令 133
3.2.3　レジスタとスタック 134
3.2.4　アドレシング 135
3.3　C 言語（組込み系の主流言語，構造化，関数型）..... 135
3.3.1　C 言語の特徴 136
3.3.2　構造化 136
3.3.3　プリプロセッサ 137
3.3.4　C 言語の応用 137
3.4　Java（オブジェクト指向の代表言語，クラス，仮想マシン）..... 138
3.4.1　オブジェクト指向言語 138
3.4.2　クラスライブラリ 139
3.4.3　仮想マシン 139
3.4.4　バイトコード 139

4　その他の言語（新しい概念に基づく言語） 141

4.1　マークアップ言語（HTML/XML など）..... 141
4.1.1　マークアップ言語とは 141
4.1.2　マークアップ言語と Web 142
4.1.3　HTML 142
4.1.4　XML 143
4.2　ハードウェア記述言語（HDL）..... 143
4.2.1　ハードウェア記述言語とは 144
4.2.2　シミュレーション 144
4.2.3　VHDL 145
4.2.4　VerilogHDL 145
4.3　システム記述言語（SystemC など）..... 146
4.3.1　システム記述言語とは 146
4.3.2　抽象度 146
4.3.3　SystemC 147
4.3.4　論理合成と動作合成 148
4.4　モデリング言語（UML など）..... 148
4.4.1　モデリング言語とは 148
4.4.2　モデリング言語の使い方 149
4.4.3　UML 149
4.4.4　モデル駆動開発 150

索　引 151

1章 情報の基礎

1 情報表現
（1と0で何を表すか？）

1.1　情報量（ビット，ワード）

ディジタルシステムでは情報はすべて 1 と 0 で表わされます。

さまざまな情報が最終的に 1 と 0 でどのように表されるのか，イメージできるようにしておきましょう。

ディジタル情報で 1 か 0 かの基本（最小）単位は「**ビット**」です。

複数ビットをまとめて「**ワード**」と呼びますが,「ワード」は定義により異なり，16 ビットのこともあれば，32 ビットのこともあります。

何ビット構成で何を表すのかを明確にして議論する必要があります。

1.1.1　情報量の考え方

情報量について考える場合，基本的に重要なのは「状態の区別」です。

ビット数を増やすと，より多くの状態が区別でき，情報量が多くなります。

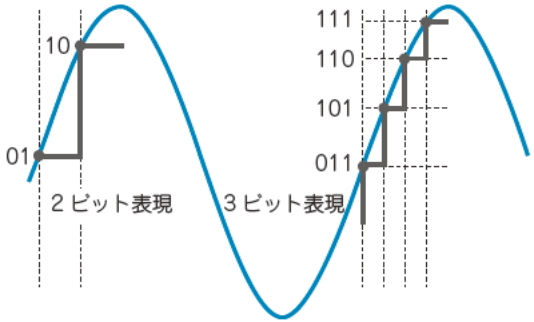

図 1.1　波形のディジタル化情報量

1.1.2　2 進表示と 16 進表示

ディジタル情報は 1 と 0 で表す「2 進表示」で表現されますが，ビット数が多くなると扱いにくいので，4 ビットを 1 文字で表す「16 進表示」がよく使われます。

0 ～ 9 の 10 文字では不足するので A ～ F（または a ～ f）のアルファベットを利用します。

16 進		4 ビット	16 進		4 ビット
0	=	0000	8	=	1000
1	=	0001	9	=	1001
2	=	0010	A	=	1010
3	=	0011	B	=	1011
4	=	0100	C	=	1100
5	=	0101	D	=	1101
6	=	0110	E	=	1110
7	=	0111	F	=	1111

図 1.2　16 進表示は 4 ビットを 1 文字で表現

16 進表示であることを示すため，「12H」や「0x12」のように H や 0x を付けて表記します。

1.1.3　n ビットの情報量

1 ビットでは **1 か 0 か**の 2 つの状態を区別でき，2 ビットでは，00，01，10，11 の 4 つの状態が区別できます。

1 ビット増えるごとに 2 倍の状態を区別できるので，一般に **n ビットでは 2^n 通りの状態を表せる**ことになります。

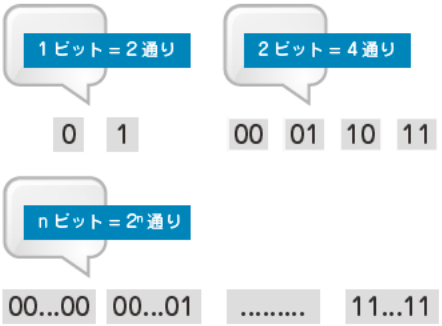

図 1.3　n ビットの情報量

1.1.4　情報量の単位

コンピュータなどで情報量を表すために下記のような単位が使われます。

ビット（bit，b）	
バイト（Byte，B）	8bit
キロバイト（KB）	10^3Byte
メガバイト（MB）	10^6Byte
ギガバイト（GB）	10^9Byte
テラバイト（TB）	10^{12}Byte

以降，10^3 ごとにペタ(PB)，エクサ(EB)，ゼタ(ZB)，ヨタ(YB) と続きます。

1 と 0 の世界では 2 進数が基本なので，慣習的に「$2^{10} = 1024$」をキロ（K）と呼び，これを基準に，以降，メガ，ギガ，テラ…と上記の 10 進数と同様に使われますが，実際には誤差を含むので厳密な議論では注意を要します。

例えば，2 進数で 64KB（キロバイト）は 2^{16} なので，10 進数の 64,000 に対し 65,536 です。

1.2　基数変換（2進/10進など）

ディジタルの世界では，数値は 1 と 0 のみ，つまり数学的には 2 進数で表されます。

人間が数値を扱う場合通常 10 進数を使いますが，さまざまな理由でそれ以外の表現方法を利用することもあります。

人間は文字を使って情報を表現するので，その文字（数字）が何を意味するのかを明確にしておかないと混乱や誤解を招きます。

例えば，「101」個のリンゴは 10 進数表記なら 101 個ですが，「101」が 2 進数表記なら 5 個，16 進数表記なら 257 個を意味します。

1.2.1　基数とは（n 進数）

数値を表すときに，各桁の重み付けの基本となる数を**基数**（radix）といいます。

10 進数では，1 の桁，10 の桁，100 の桁のように重さが 10 倍になっていきますが，2 進数では，1 の桁，2 の桁，4 の桁，8 の桁のように重さが 2 倍になっていきます。

n 進数では，1 の桁，n の桁，n^2 の桁，n^3 の桁のように重さが n 倍になりますが，整理すると「下から **i 番目の桁の重さが $n^{(i-1)}$ である**」ことになります（$n^1 = n$，$n^0 = 1$ です）。

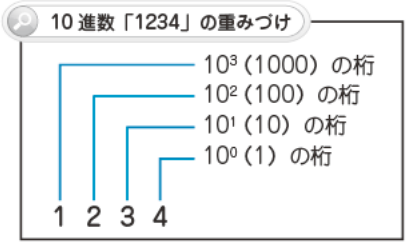

図 1.4　n 進数（1）

そこで $abcd$ が n 進表記であることを $(abcd)_n$ と表す場合，数式で正確に示せば，

$$(abcd)_n = a \times n^3 + b \times n^2 + c \times n^1 + d \times n^0$$

です。

図 1.5 に 10 進数と 7 進数の例を示します。

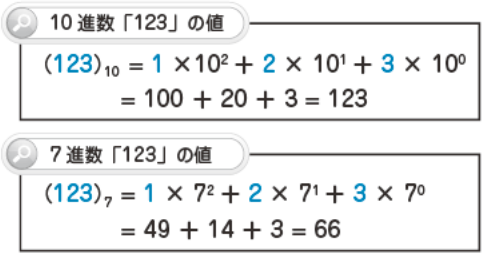

図 1.5　n 進数（2）