

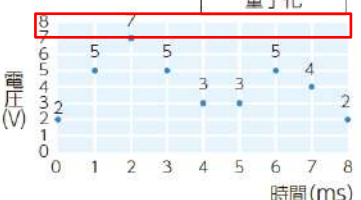
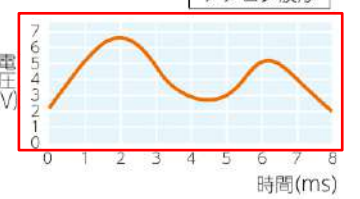
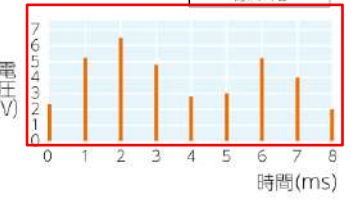

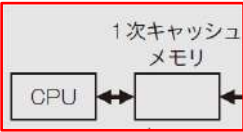
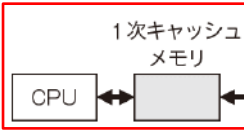
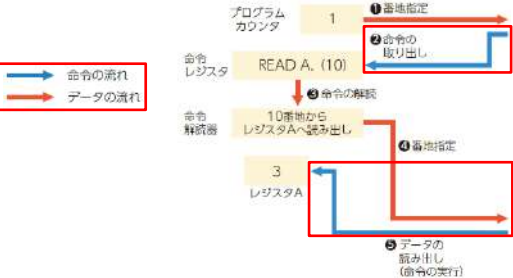
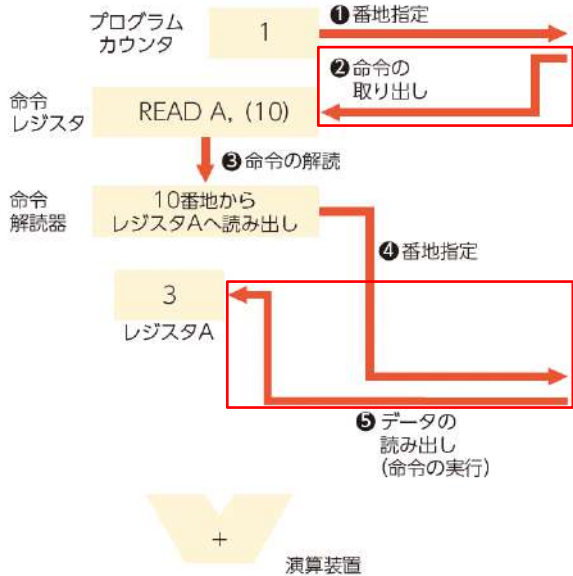


番号	訂正箇所		原 文	訂 正 文
	ページ	行		
1	19	例題1 問(1)	<u>同級生にあげ</u>	<u>多数の同級生</u> に配布し
2	47	19行	⑦00000001 ₍₂₎ (→補数) ⑧01010011 ₍₂₎ (→補数)	⑦00000001 ₍₂₎ → 補数 ⑧01010011 ₍₂₎ → 補数
3	51	1-4行	<p>次の計算結果を指定された形式で表現しなさい。ただし数値は4ビットの整数とし、2進数の負の値は補数で表現すること。</p> <p>① $10_{(2)} + A_{(16)} \rightarrow 10$進数 ② $100_{(2)} - C_{(16)} - 5_{(10)} \rightarrow 2$進数 ③ $B_{(16)} - 11_{(10)} - 11_{(2)} \rightarrow 16$進数 ④ $7_{(10)} + 3_{(16)} - 1100_{(2)} \rightarrow 2$進数</p>	<p>次の計算をしなさい。ただし数値は4ビットの整数とし、<u>減算は補数を使って計算すること。</u></p> <p>① <u>$0100_{(2)} - 0011_{(2)}$</u> ② <u>$1110_{(2)} - 0010_{(2)}$</u> ③ <u>$1101_{(2)} - 0101_{(2)}$</u> ④ <u>$1111_{(2)} - 1100_{(2)}$</u></p>

番号	訂正箇所		原文	訂正文
	ページ	行		
4	54	図2	<p style="text-align: center;">アナログ波形</p>  <p style="text-align: center;">標本化</p>  <p style="text-align: center;">量子化</p> 	<p style="text-align: center;">アナログ波形</p>  <p style="text-align: center;">標本化</p>  <p style="text-align: center;">量子化</p> 







番号	訂正箇所		原文	訂正文
	ページ	行		
5	61	図3		
6	61	図3 キャプション	<u>キャッシュメモリの仕組み</u>	<u>CPU, 1次/2次キャッシュメモリと主記憶装置の例</u>
7	62	手順1 図		

番号	訂正箇所		原文	訂正文
	ページ	行		
8	63	手順2 図	<p>Original diagram description: The diagram shows the execution of an ADD instruction. <ul style="list-style-type: none"> Step 1: Program Counter (2) specifies address 11. Step 2: Command 'ADD A, (11)' is retrieved from memory. Step 3: Command is decoded to 'sum of register A and address 11'. Step 4: Address 11 is specified for the ALU. Step 5: Addition is performed: Register A (8) + 11 = 19. </p>	<p>Revised diagram description: This diagram is identical to the original one, showing the execution of an ADD instruction with the same steps and components.</p>

番号	訂正箇所		原文	訂正文
	ページ	行		
9	63	手順3 図	<p>Original diagram details: - Program Counter: 3 - Command Register: WRITE (12), A - Command Decoder: レジスタAから12番地へ書き込み - Register A: 8 - ALU: + - Steps: ①番地指定, ②命令の取り出し, ③命令の解読, ④番地指定, ⑤データの書き込み(命令の実行)</p>	<p>Revised diagram details: - Program Counter: 3 - Command Register: WRITE (12), A - Command Decoder: レジスタAから12番地へ書き込み - Register A: 8 - ALU: + - Steps: ①番地指定, ②命令の取り出し, ③命令の解読, ④番地指定, ⑤データの書き込み(命令の実行)</p>

番号	訂正箇所		原文	訂正文
	ページ	行		
10	68	3-4行	<p>値を量子化してから2ビットに符号化して</p>	<p>値を2ビットで量子化してから符号化し</p>
11	91	11行	<p>パリティビット</p>	<p>チェックディジット</p>
12	91	13行	<p>パリティビットを 削除</p>	<p>(削除)</p>
13	91	17行	<p>削除</p>	<p>(削除)</p>
14	91	17行	<p>ただし</p>	<p>この1ビットをパリティビットという。ただし</p>
15	93	6-7行	<p>公開鍵暗号方式で暗号化・復号を行う二重の暗号化方式を利用している。</p>	<p>公開鍵暗号方式または鍵共有方式で共有する方法をとっている。</p>



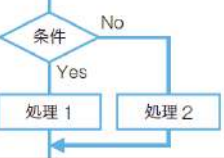
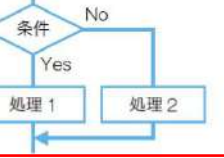
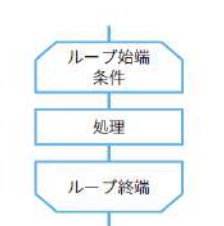
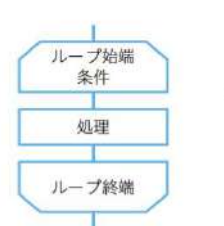
番号	訂正箇所		原文	訂正文
	ページ	行		
16	93	側注⑤ および ⑥	<div style="border: 1px solid red; height: 100px; width: 100%;"></div> <p>⑤もともと利用されていたSSLは、セキュリティ強化のために改良されTLSに移行したが、SSLという語が普及しているためその発展系という意味でSSL/TLSと表記したり、単にSSLと記される場合もある。</p> <div style="border: 1px solid red; height: 100px; width: 100%;"></div> <p>⑥デジタル署名だけでは、な</p>	<p>⑤共通鍵暗号方式で利用する共通鍵を安全に共有する方法の一つである。大まかな手順は次の通りである。</p> <ol style="list-style-type: none"> 1. 送信側、受信側でそれぞれ秘密鍵を準備する。 2. 送信側、受信側でそれぞれ秘密鍵を利用して公開鍵を計算し、相手に送る。 3. 送信側、受信側はそれぞれ受け取った公開鍵と自分もつ秘密鍵を使用し、共通鍵を算出するための数値を求める。 4. 3で共有された数値を使い、送信側も受信側も等しい値の共通鍵を算出する(鍵共有完了)。 <p>⑥もともと利用されていたSSLは、セキュリティ強化のために改良されTLSに移行したが、SSLという語が普及しているためその発展系という意味でSSL/TLSと表記したり、単にSSLと記したりする場合がある。</p> <div style="border: 1px solid red; height: 15px; width: 100%;"></div> <p>⑦デジタル署名だけでは、な</p>
17	93	図4 キャプション	SSL/TLS_	SSL/TLS(<u>共通鍵を公開鍵暗号方式で共有する場合</u>)

番号	訂正箇所		原 文	訂 正 文
	ページ	行		
18	93	11行		
19	93	13行		
20	93	側注⑦		
21	128	9行	データの <u>処理</u>	データの <u>扱い</u>
22	162	表5 「値の取 り出し」	<pre>for_ (変数 in 連想配列) { 処理 }</pre>	<pre>for_ (変数 in 連想配列) { 処理 }</pre>
23	134	表1	別添1	別添1

番号	訂正箇所		原文	訂正文																																																																																																										
	ページ	行																																																																																																												
24	135	表2	<table border="1"> <thead> <tr> <th colspan="2">代入演算子</th> <th colspan="2">比較演算子</th> <th colspan="2">論理演算子</th> </tr> </thead> <tbody> <tr> <td>=</td> <td>代入^①</td> <td>==</td> <td>等しい^①</td> <td>&&</td> <td>かつ</td> </tr> <tr> <td colspan="2">算術・結合演算子</td> <td>!=</td> <td>等しくない</td> <td> </td> <td>または</td> </tr> <tr> <td rowspan="2">+</td> <td>数値の足し算^①</td> <td><</td> <td>未満</td> <td>!</td> <td>ではない</td> </tr> <tr> <td>文字列の結合^①</td> <td>></td> <td>超過</td> <td colspan="2">増減演算子</td> </tr> <tr> <td>-</td> <td>引き算</td> <td><=</td> <td>以下</td> <td>++</td> <td>1増やす</td> </tr> <tr> <td>*</td> <td>掛け算</td> <td>>=</td> <td>以上</td> <td>--</td> <td>1減らす</td> </tr> <tr> <td>/</td> <td>割り算</td> <td colspan="2">カンマ演算子</td> <td colspan="2">ドット演算子</td> </tr> <tr> <td>%</td> <td>割り算の余り</td> <td>,</td> <td>式の区切り</td> <td>.</td> <td>～の</td> </tr> </tbody> </table>	代入演算子		比較演算子		論理演算子		=	代入 ^①	==	等しい ^①	&&	かつ	算術・結合演算子		!=	等しくない		または	+	数値の足し算 ^①	<	未満	!	ではない	文字列の結合 ^①	>	超過	増減演算子		-	引き算	<=	以下	++	1増やす	*	掛け算	>=	以上	--	1減らす	/	割り算	カンマ演算子		ドット演算子		%	割り算の余り	,	式の区切り	.	～の	<table border="1"> <thead> <tr> <th colspan="2">代入演算子</th> <th colspan="2">比較演算子</th> <th colspan="2">論理演算子</th> </tr> </thead> <tbody> <tr> <td>=</td> <td>代入^①</td> <td>==</td> <td>等しい^①</td> <td>&&</td> <td>かつ</td> </tr> <tr> <td colspan="2">算術・結合演算子</td> <td>!=</td> <td>等しくない</td> <td> </td> <td>または</td> </tr> <tr> <td rowspan="2">+</td> <td>数値の足し算^①</td> <td><</td> <td>未満</td> <td>!</td> <td>ではない</td> </tr> <tr> <td>文字列の結合^①</td> <td>></td> <td>超過</td> <td colspan="2">増減演算子</td> </tr> <tr> <td>-</td> <td>引き算</td> <td><=</td> <td>以下</td> <td>++</td> <td>1増やす</td> </tr> <tr> <td>*</td> <td>掛け算</td> <td>>=</td> <td>以上</td> <td>--</td> <td>1減らす</td> </tr> <tr> <td>/</td> <td>割り算</td> <td colspan="2">カンマ演算子</td> <td colspan="2">ドット演算子</td> </tr> <tr> <td>%</td> <td>割り算の余り</td> <td>,</td> <td>式の区切り</td> <td>.</td> <td>～の</td> </tr> </tbody> </table>	代入演算子		比較演算子		論理演算子		=	代入 ^①	==	等しい ^①	&&	かつ	算術・結合演算子		!=	等しくない		または	+	数値の足し算 ^①	<	未満	!	ではない	文字列の結合 ^①	>	超過	増減演算子		-	引き算	<=	以下	++	1増やす	*	掛け算	>=	以上	--	1減らす	/	割り算	カンマ演算子		ドット演算子		%	割り算の余り	,	式の区切り	.	～の
代入演算子		比較演算子		論理演算子																																																																																																										
=	代入 ^①	==	等しい ^①	&&	かつ																																																																																																									
算術・結合演算子		!=	等しくない		または																																																																																																									
+	数値の足し算 ^①	<	未満	!	ではない																																																																																																									
	文字列の結合 ^①	>	超過	増減演算子																																																																																																										
-	引き算	<=	以下	++	1増やす																																																																																																									
*	掛け算	>=	以上	--	1減らす																																																																																																									
/	割り算	カンマ演算子		ドット演算子																																																																																																										
%	割り算の余り	,	式の区切り	.	～の																																																																																																									
代入演算子		比較演算子		論理演算子																																																																																																										
=	代入 ^①	==	等しい ^①	&&	かつ																																																																																																									
算術・結合演算子		!=	等しくない		または																																																																																																									
+	数値の足し算 ^①	<	未満	!	ではない																																																																																																									
	文字列の結合 ^①	>	超過	増減演算子																																																																																																										
-	引き算	<=	以下	++	1増やす																																																																																																									
*	掛け算	>=	以上	--	1減らす																																																																																																									
/	割り算	カンマ演算子		ドット演算子																																																																																																										
%	割り算の余り	,	式の区切り	.	～の																																																																																																									
25	171	8行	込み <u>関数</u> などを利用する。これらの <u>関数</u>	込み <u>メソッド</u> などを利用する。これらの <u>メソッド</u>																																																																																																										
26	171	図2 右表 図3 右表 図4 右表 図5 右表	<u>関数</u> <u>関数</u> <u>関数</u> <u>関数</u>	<u>メソッド</u> <u>メソッド</u> <u>メソッド</u> <u>メソッド</u>																																																																																																										

番号	訂正箇所		原 文	訂 正 文		
	ページ	行				
27	171	図2キャ プション 図3キャ プション 図4キャ プション 図5キャ プション	<u>関数の例</u> <u>関数の例</u> <u>関数の例</u> <u>関数の例</u>	<u>メソッドの例</u> <u>メソッドの例</u> <u>メソッドの例</u> <u>メソッドの例</u>		
28	172	13行	<u>関数</u>	<u>メソッド</u>		
29	179	3行	<u>60kg重</u>	<u>588N</u>		
30	179	4行 6行	<u>kg重</u> <u>kg重</u>	<u>N</u> <u>N</u>		
31	72	表1	<table border="1" data-bbox="385 1230 638 1289"> <tr> <td>1層</td> <td><u>リンク層</u></td> </tr> </table>	1層	<u>リンク層</u>	<u>ネットワークインタフェース</u>
1層	<u>リンク層</u>					

番号	訂正箇所		原文	訂正文
	ページ	行		
32	73	上の図		<u>ネットワークインタフェース</u>
33	73	側注④	<p>④ アプリケーション層のヘッダは、全パケットには付加されない。また、<u>リンク層</u>では、ヘッダだけでなくデータの最後にエラーをチェックするためのトレーラという情報も付く。</p>	<u>ネットワーク</u> <u>インタフェース</u>
34	190	左から4段目 43-44行	<p><u>ネットワーク</u>..... 70 <u>ネットワーク層</u>..... 72</p>	<u>ネットワークインタフェース層..... 72</u>
35	191	左から3段目 33行	<p><u>リンク</u>..... 37 <u>リンク層</u>..... 72 類似色..... 26 削除</p>	<p><u>リンク</u>..... 37 <u>類似色</u>..... 26</p>

別添 No.	原 文		訂 正 文						
1									
	順次		処理1; 処理2;	上から下へ記述された順に処理を実行する(ここでは処理1の後に処理2)。		順次		処理1; 処理2;	上から下へ記述された順に処理を実行する(ここでは処理1の後に処理2)。
	選択		if (条件){ 処理1; }else{ 処理2; }	条件が真(Yes)の時(条件が成立する時)は処理1を実行し、偽(No)の時(条件が成立しない時)は処理2を実行する。		選択		if (条件){ 処理1; }else{ 処理2; }	条件が真(Yes)の時(条件が成立する時)は処理1を実行し、偽(No)の時(条件が成立しない時)は処理2を実行する。
	反復		① for (初期設定; 条件; 再設定){ 処理; } ② 初期設定; while (条件){ 処理; 再設定; }	条件が真の間、ループの始端と終端の間にある処理を繰り返し実行し、偽になるとループから抜ける。 初期設定でループ変数に初期値を設定し、再設定でループ変数の値を増減する。 途中で繰り返しを中断するにはループ内にbreak;を入れる。		反復		① for (初期設定; 条件; 再設定){ 処理; } ② 初期設定; while (条件){ 処理; 再設定; }	条件が真の間、ループの始端と終端の間にある処理を繰り返し実行し、偽になるとループから抜ける。 初期設定でループ変数に初期値を設定し、再設定でループ変数の値を増減する。 途中で繰り返しを中断するにはループ内にbreak;を入れる。