第 3 刷･第 2 刷･第 1 刷訂正情報

本書（第 3 刷、第 2 刷および第 1 刷）の掲載内容に下記の誤りがございました。
ご迷惑をおかけしましたことをお詫び申し上げます。

（2020 年 6 月 25 日更新）

P.332　上部の表　3 箇所

誤：

INT 0x15(AH=0xE820)：システムメモリマップ

表 11-10 Int 0x15 AX=0xE820：システムメモリマップ

入力	AX	0xE820（固定）
	EBX	インデックス（初回のみ0を設定する）
	ES:DI	情報書き込み先
	ECX	書き込み先バイト数
	EDX	0x534D4150（"SMAP"固定）
出力	CF	0=成功、1=失敗
	EAX	0x534D4150（"SMAP"固定）
	ECX	書き込みバイト数
	EBX	インデックス（0のときは最終データ）

正：

EAX

P.336　一番下の段落　1 行目

誤：

ACPI テーブルの位置は、システムメモリマップ問い合わせ用 BIOS コール（INT15 0x8E20）で取得します。

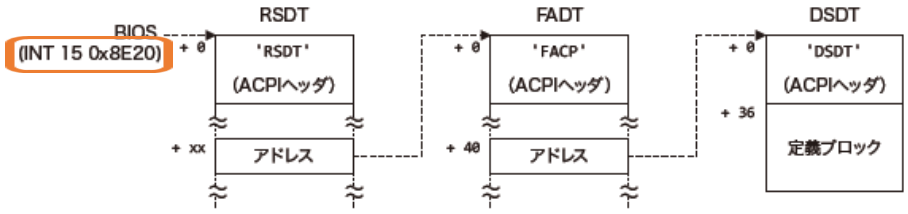
正：

ACPI テーブルの位置は、システムメモリマップ問い合わせ用 BIOS コール（INT15 0xE820）で取得します。

P.336　一番下の図

誤：

図 11-7 各ディスクリプタテーブルの関係



正：

（INT15 0xE820）

P.441　上から 2 番目のコード　下から 1 行目

誤：

prog/src/modules/real/get_mem_info.s

```
        cmp     ebx, 0                  ;
        jne     .10L                    ; }
.10E:                                   ; while (0 == EBX);
```

正：

while (0　!= EBX);

P.493　下から 1 番目のコード

誤：

prog/src/modules/protect/draw_char.s

```
;-------------------------------------
; 1文字分のフォントを出力
;-------------------------------------
movzx   ebx, word [ebp +16]             ; // 表示色

cdecl   vga_set_read_plane, 0x03        ; // 書き込みプレーン：輝度(I)
cdecl   vga_set_write_plane, 0x08       ; // 読み込みプレーン：輝度(I)
cdecl   vram_font_copy, esi, edi, 0x08, ebx
```

正：

prog/src/modules/protect/draw_char.s

```
;-------------------------------------
; 1文字分のフォントを出力
;-------------------------------------
movzx   ebx, word [ebp +16]             ; // 表示色

cdecl   vga_set_read_plane, 0x03        ; // 読み込みプレーン：輝度(I)
cdecl   vga_set_write_plane, 0x08       ; // 書き込みプレーン：輝度(I)
cdecl   vram_font_copy, esi, edi, 0x08, ebx
```

P.494　上から 1 番目のコード

誤：

```
cdecl   vga_set_read_plane, 0x02        ; // 書き込みプレーン：赤(R)
cdecl   vga_set_write_plane, 0x04       ; // 読み込みプレーン：赤(R)
cdecl   vram_font_copy, esi, edi, 0x04, ebx

cdecl   vga_set_read_plane, 0x01        ; // 書き込みプレーン：緑(G)
cdecl   vga_set_write_plane, 0x02       ; // 読み込みプレーン：緑(G)
cdecl   vram_font_copy, esi, edi, 0x02, ebx

cdecl   vga_set_read_plane, 0x00        ; // 書き込みプレーン：青(B)
cdecl   vga_set_write_plane, 0x01       ; // 読み込みプレーン：青(B)
cdecl   vram_font_copy, esi, edi, 0x01, ebx
```

正：

```
cdecl   vga_set_read_plane, 0x02        ; // 読み込みプレーン：赤(R)
cdecl   vga_set_write_plane, 0x04       ; // 書き込みプレーン：赤(R)
cdecl   vram_font_copy, esi, edi, 0x04, ebx

cdecl   vga_set_read_plane, 0x01        ; // 読み込みプレーン：緑(G)
cdecl   vga_set_write_plane, 0x02       ; // 書き込みプレーン：緑(G)
cdecl   vram_font_copy, esi, edi, 0x02, ebx

cdecl   vga_set_read_plane, 0x00        ; // 読み込みプレーン：青(B)
cdecl   vga_set_write_plane, 0x01       ; // 書き込みプレーン：青(B)
cdecl   vram_font_copy, esi, edi, 0x01, ebx
```

P.502　下から 1 番目のコード　上から 2 行目

誤：

prog/src/modules/protect/draw_color_bar.s

```
        mov     edx, ecx                        ;   EDX  = ECX;
        shl     edx, 1                          ;   EDX /= 2
        mov     edx, [.t0 + edx]                ;   EDX += Y;

        cdecl   draw_str, eax, ebx, edx, .s0    ;   draw_str();
```

正：

EDX *= 2;

P.511　一番上のコード

誤：

```
        push    dword 0                         ;     -12| dx    = 0; // X増分
        push    dword 0                         ;     -16| inc_x = 0; // X座標増分(1 or -1)
        push    dword 0                         ;     -20| x0    = 0; // Y座標
        push    dword 0                         ;     -24| dx    = 0; // Y増分
        push    dword 0                         ;     -28| inc_x = 0; // Y座標増分(1 or -1)
```

正：

y0

dy

inc_y

P.512　一番上のコード

誤：

prog/src/modules/protect/draw_line.s

```
        ;--------------------------------------
        ; 基準軸を決める
        ;--------------------------------------
        cmp     ebx, edx                        ; if (幅 <= 高さ)
        jg      .22F                            ; {
                                                ;
        lea     esi, [ebp -20]                  ;   // X軸が基準軸
        lea     edi, [ebp - 8]                  ;   // Y軸が相対軸
                                                ;
        jmp     .22E                            ; }
.22F:                                           ; else
                                                ; {
        lea     esi, [ebp - 8]                  ;   // Y軸が基準軸
        lea     edi, [ebp -20]                  ;   // X軸が相対軸
.22E:                                           ; }
```

正：

prog/src/modules/protect/draw_line.s

```
        ;--------------------------------------
        ; 基準軸を決める
        ;--------------------------------------
        cmp     ebx, edx                        ; if (幅 <= 高さ)
        jg      .22F                            ; {
                                                ;
        lea     esi, [ebp -20]                  ;   // Y軸が基準軸
        lea     edi, [ebp - 8]                  ;   // X軸が相対軸
                                                ;
        jmp     .22E                            ; }
.22F:                                           ; else
                                                ; {
        lea     esi, [ebp - 8]                  ;   // X軸が基準軸
        lea     edi, [ebp -20]                  ;   // Y軸が相対軸
.22E:                                           ; }
```

P.517　下 2 つのコードのフォルダ名

誤：

prog/src/23_draw_rect/kernel.s

正：

prog/src/24_draw_rect/kernel.s

P.540　下部のコードの 13 行目

誤：

prog/src/modules/protect/pic.s

```
init_pic:
        ;--------------------------------------
        ; 【レジスタの保存】
        ;--------------------------------------
        push    eax

        ;--------------------------------------
        ; マスタの設定
        ;--------------------------------------
        outp    0x20, 0x11                      ; // MASTER.ICW1 = 0x11;
        outp    0x21, 0x20                      ; // MASTER.ICW2 = 0x20;
        outp    0x21, 0x04                      ; // MASTER.ICW3 = 0x04;
        outp    0x21, 0x05                      ; // MASTER.ICW4 = 0x05;
        outp    0x21, 0xFF                      ; // マスタ割り込みマスク
```

正：

outp 0x21, 0x01 ; // MASTER.ICW4 = 0x01;

P.556　一番上のコードの 9 行目

誤：

prog/src/modules/protect/ring_buff.s

```
draw_key:
        ...
        ;--------------------------------------
        ; リングバッファの情報を取得
        ;--------------------------------------
        mov     ebx, [esi + ring_buff.rp]       ; EBX = wp;             // 書き込み位置
        lea     esi, [esi + ring_buff.item]     ; ESI = &KEY_BUFF[EBX];
        mov     ecx, RING_ITEM_SIZE             ; ECX = RING_ITEM_SIZE; // 要素数
                                                ;
.10L:                                           ; do
                                                ; {
        ...                                     ;   【バッファ内要素の表示】
        loop    .10L                            ;
.10E:                                           ; } while (ECX--);
```

正：

EBX = rp; // 読み込み位置

P.564　最後の一文

誤：

次の図は、セグメントセレクタの内部構成を再掲したものです。

正

次の図は、セグメントセレクタの内部構成を示したものです。

P.579　一番上の行（上から 1 番目のコード内）コメント部

誤：

// タスク 1 へのジャンプ

正：

// タスク 0 へのジャンプ

P.579　上から 2 番目のコード　ファイル名

誤：

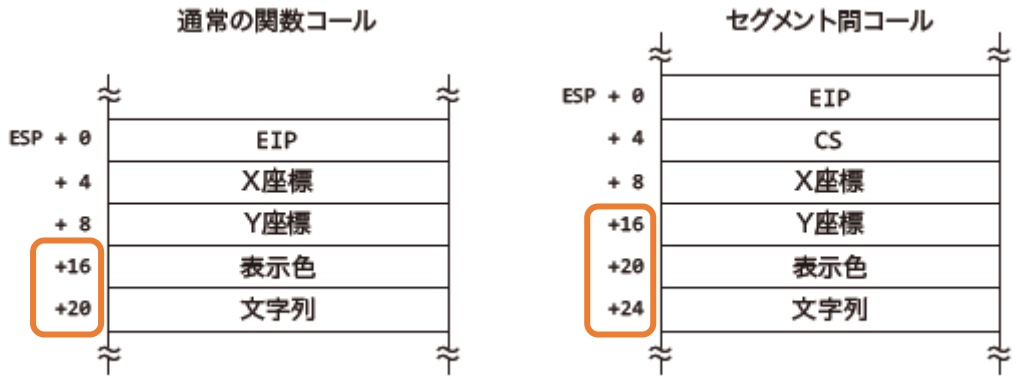
prog/src/32_task_non_pre/tasks/task_1.s

正：

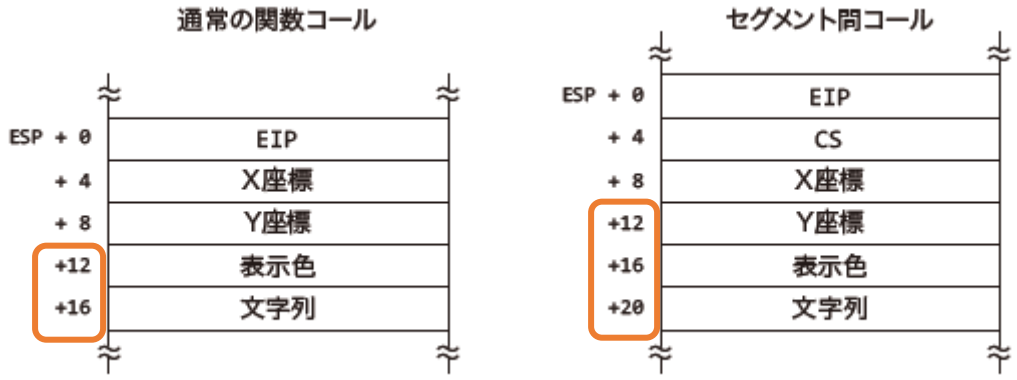
prog/src/32_task_non_pre/kernel.s

P.588　一番上の図（図 20-5）

誤：



正：



P.606　一番上のコード

誤：

prog/src/37_fpu/tasks/task_2.s

```
task_2:
        ...
                                ; --------+--------+--------|--------|--------|--------|
                                ;      ST0|     ST1|     ST2|     ST3|     ST4|     ST5|
                                ; --------+--------+--------|--------|--------|--------|
    fild    dword [.c1000]      ;    1000 |xxxxxxxx|xxxxxxxx|xxxxxxxx|xxxxxxxx|xxxxxxxx|
    fldpi                       ;      pi |    1000 |xxxxxxxx|xxxxxxxx|xxxxxxxx|xxxxxxxx|
    fidiv   dword [.c180]       ;  pi/180 |    1000 |xxxxxxxx|xxxxxxxx|xxxxxxxx|xxxxxxxx|
    fldpi                       ;      pi |  pi/180 |    1000 |xxxxxxxx|xxxxxxxx|xxxxxxxx|
    fadd    st0, st0            ;    2*pi |  pi/180 |    1000 |xxxxxxxx|xxxxxxxx|xxxxxxxx|
    fldz                        ;   θ= 0 |    2*pi |  pi/180 |    1000 |xxxxxxxx|xxxxxxxx|
                                ; --------+--------+--------|--------|--------|--------|
                                ;   θ= 0 |    2*pi |       d |    1000 |xxxxxxxx|xxxxxxxx|
                                ; --------+--------+--------|--------|--------|--------|
```

正：

d=pi/180

P.606　一番下のコード

誤：

prog/src/37_fpu/tasks/task_2.s

```
task_2:
        ...
                                ; --------+--------+--------|--------|--------|--------|
                                ;      ST0|     ST1|     ST2|     ST3|     ST4|     ST5|
                                ; --------+--------+--------|--------|--------|--------|
                                ;       θ |    2*pi |       d |    1000 |xxxxxxxx|xxxxxxxx|
                                ; --------+--------+--------|--------|--------|--------|
    fadd    st0, st2            ;   θ + d |    2*pi |       d |    1000 |xxxxxxxx|xxxxxxxx|
    fprem                       ;  MOD(θ) |       θ |    2*pi |       d |    1000 |xxxxxxxx|
    fld     st0                 ;       θ |       θ |    2*pi |       d |    1000 |xxxxxxxx|
    fsin                        ;  sin(θ) |       θ |    2*pi |       d |    1000 |xxxxxxxx|
                                ; --------+--------+--------|--------|--------|--------|
```

正：

prog/src/37_fpu/tasks/task_2.s

```
task_2:
        ...
                                ; --------+--------+--------|--------|--------|--------|
                                ;      ST0|     ST1|     ST2|     ST3|     ST4|     ST5|
                                ; --------+--------+--------|--------|--------|--------|
                                ;       θ |    2*pi |       d |    1000 |xxxxxxxx|xxxxxxxx|
                                ; --------+--------+--------|--------|--------|--------|
    fadd    st0, st2            ;   θ + d |    2*pi |       d |    1000 |xxxxxxxx|xxxxxxxx|
    fprem                       ;  MOD(θ) |    2*pi |       d |    1000 |xxxxxxxx|xxxxxxxx|
    fld     st0                 ;       θ |       θ |    2*pi |       d |    1000 |xxxxxxxx|
    fsin                        ;  sin(θ) |       θ |    2*pi |       d |    1000 |xxxxxxxx|
                                ; --------+--------+--------|--------|--------|--------|
```

P.607　上部のコード

誤：

prog/src/37_fpu/tasks/task_2.s

```
task_2:
        ...
                                ; ---------+---------+---------|---------|---------|---------|
                                ;       ST0|      ST1|      ST2|      ST3|      ST4|      ST5|
                                ; ---------+---------+---------|---------|---------|---------|
                                ;        θ |    2*pi |       d |    1000 |xxxxxxxxx|xxxxxxxxx|
                                ; ---------+---------+---------|---------|---------|---------|
    fadd    st0, st2            ;    θ + d |    2*pi |       d |    1000 |xxxxxxxxx|xxxxxxxxx|
    fprem                       ;   MOD(θ)|       θ |    2*pi |       d |    1000 |xxxxxxxxx|
    fld     st0                 ;        θ |       θ |    2*pi |       d |    1000 |xxxxxxxxx|
    fsin                        ;   sin(θ)|       θ |    2*pi |       d |    1000 |xxxxxxxxx|
    fmul    st0, st4            ; ST4sin(θ)|       θ |    2*pi |       d |    1000 |xxxxxxxxx|
    fbstp   [.bcd]              ;        θ |    2*pi |       d |    1000 |xxxxxxxxx|xxxxxxxxx|
                                ; ---------+---------+---------|---------|---------|---------|
        ...
.bcd:   times 10 db 0x00
```

正：

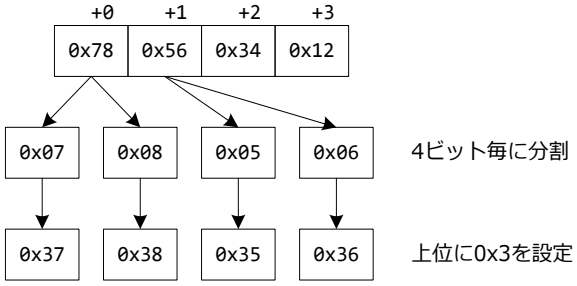
prog/src/37_fpu/tasks/task_2.s

```
task_2:
        ...
                                ; ---------+---------+---------|---------|---------|---------|
                                ;       ST0|      ST1|      ST2|      ST3|      ST4|      ST5|
                                ; ---------+---------+---------|---------|---------|---------|
                                ;        θ |    2*pi |       d |    1000 |xxxxxxxxx|xxxxxxxxx|
                                ; ---------+---------+---------|---------|---------|---------|
    fadd    st0, st2            ;    θ + d |    2*pi |       d |    1000 |xxxxxxxxx|xxxxxxxxx|
    fprem                       ;   MOD(θ)|    2*pi |       d |    1000 |xxxxxxxxx|xxxxxxxxx|
    fld     st0                 ;        θ |       θ |    2*pi |       d |    1000 |xxxxxxxxx|
    fsin                        ;   sin(θ)|       θ |    2*pi |       d |    1000 |xxxxxxxxx|
    fmul    st0, st4            ; ST4sin(θ)|       θ |    2*pi |       d |    1000 |xxxxxxxxx|
    fbstp   [.bcd]              ;        θ |    2*pi |       d |    1000 |xxxxxxxxx|xxxxxxxxx|
                                ; ---------+---------+---------|---------|---------|---------|
        ...
.bcd:   times 10 db 0x00
```

P.607　図 21-8 およびその上の一段落

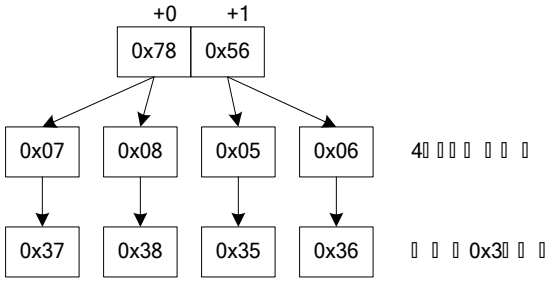
誤：

計算結果は小数点以下 3 桁までとしたので、バッファの先頭 4 バイトのみを ASCII コードに変換します。具体的には、先頭の 4 バイトを 4 ビットごとに分割し、文字コードに変換するために各桁の上位 4 ビットに 0x3 を設定します。



正：

計算結果は小数点以下 3 桁までとしたので、バッファの先頭 2 バイトのみを ASCII コードに変換します。具体的には、先頭の 2 バイトを 4 ビットごとに分割し、文字コードに変換するために各桁の上位 4 ビットに 0x3 を設定します。



P.610　上から 2 段落目

誤：

次に、タスク内で 100ms のウェイトを実行する例を示します。

正：

次に、タスク内で 200ms のウェイトを実行する例を示します。

P.610　上から 2 番目のコード　2 箇所

誤：

prog/src/37_fpu/tasks/task_2.s

```
task_2:
        ...
        ;-------------------------------------
        ; ウェイト
        ;-------------------------------------
        cdecl   wait_tick, 10                   ;   wait_tick(10);

        jmp     .10L                            ; }
```

正：

20

P.625　一番下のコード　囲み部分の一行を薄い文字で追記

正：

prog/src/39_rose/tasks/task_3.s

```
ALIGN 4, db 0
DRAW_PARAM:                                     ; 描画パラメータ
    istruc  rose
```

P.635　一番上のコード　囲み部分の一行を追記

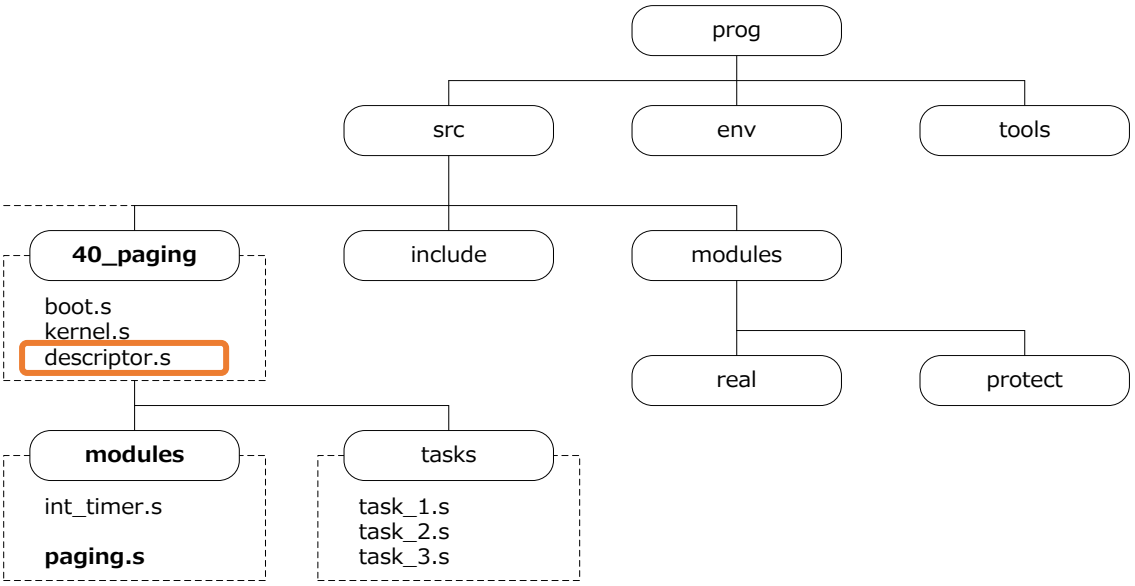
正：

prog/src/40_paging/modules/paging.s

```
page_set_4m:
    ...
        ;-------------------------------------
        ; ページ・ディレクトリの作成(P=0)
        ;-------------------------------------
        cld                                     ; // DFクリア（+方向）
        mov     edi, [ebp + 8]                  ; EDI = ページ・ディレクトリの先頭;
```

P.637　図 22-6　囲み部分を追記

正：



P.638　一番下のコードの上の一段落

誤：

最後に、CR0 レジスタの PE ビットを 1 に設定して

正：

最後に、CR0 レジスタの PG ビットを 1 に設定して

P.639　一番下のコード

誤：

prog/src/41_page_fault/modules/paging.s

```
init_page:
    ...
        cdecl   page_set_4m, 0x0010_5000        ; // ページ・テーブルの作成：タスク3用
        mov     [0x00106000 + 0x107 * 4], dword 0 ; // 0x0010_7000をページ不在に設定
```

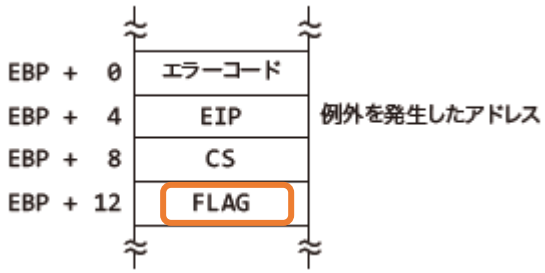
正：

prog/src/41_page_fault/modules/paging.s

```
init_page:
    ...
        cdecl   page_set_4m, CR3_BASE           ; // ページ・テーブルの作成：タスク3用
        mov     [0x00106000 + 0x107 * 4], dword 0 ; // 0x0010_7000をページ不在に設定
```

P.640　図 22-7

誤：

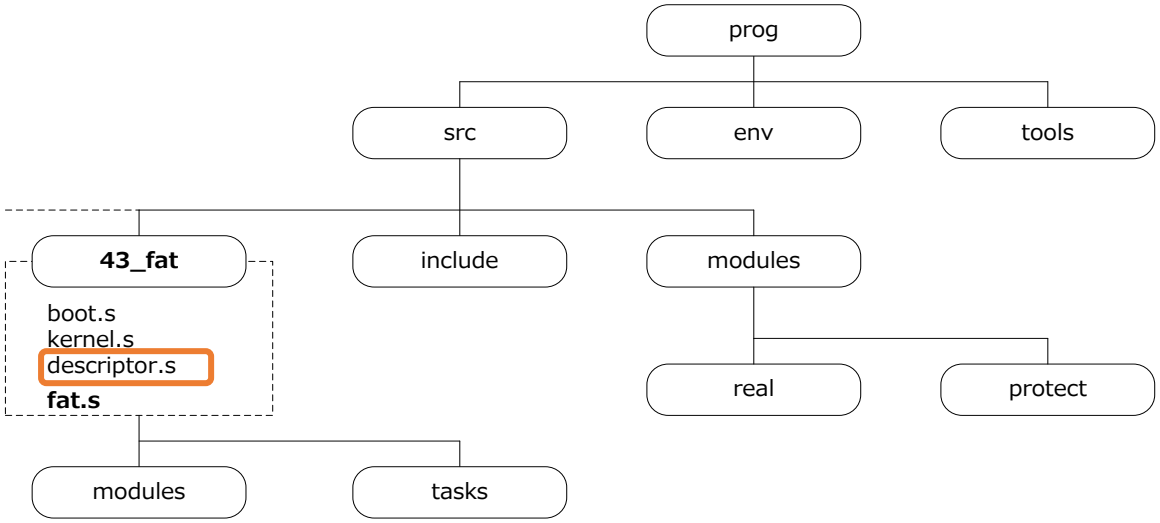


正：

EFLAGS

P.657　図 24-6　囲み部分を追記

正：



P.658　一番上のコード　3 箇所

誤：

prog/src/43_fat/fat.s

```
;-------------------------------------------------------------------------
;   FAT:ルートディレクトリ領域
;-------------------------------------------------------------------------
        times (ROOT_START) - ($ - $$)   db  0x00
;-------------------------------------------------------------------------
FAT_ROOT:
        db      'BOOTABLE', 'DSK'                               ; + 0:ボリュームラベル
        db      ATTR_ARCHIVE | ATTR_VOLUME_ID                   ; +11:属性
        db      0x00                                            ; +12:(予約)
        db      0x00                                            ; +13:TS
        dw      ( 0 << 11) | ( 0 << 5) | (0 / 2)                ; +14:作成時刻
        dw      ( 0 <<  9) | ( 0 << 5) | ( 1)                   ; +16:作成日
        dw      ( 0 <<  9) | ( 0 << 5) | ( 1)                   ; +18:アクセス日
        dw      0x0000                                          ; +20:(予約)
        dw      ( 0 << 11) | ( 0 << 5) | (0 / 2)                ; +22:更新時刻
        dw      ( 0 <<  9) | ( 0 << 5) | ( 1)                   ; +24:更新日
        dw      0                                               ; +26:先頭クラスタ
        dd      0                                               ; +28:ファイルサイズ
```

正：

1

P.658　一番下のコード

誤：

prog/src/43_fat/fat.s

```
        db      'SPECIAL ', 'TXT'                               ; + 0:ボリュームラベル
        db      ATTR_ARCHIVE                                    ; +11:属性
        db      0x00                                            ; +12:(予約)
        db      0                                               ; +13:TS
        dw      ( 0 << 11) | ( 0 << 5) | (0 / 2)                ; +14:作成時刻
        dw      ( 0 <<  9) | ( 1 << 5) | ( 1)                   ; +16:作成日
        dw      ( 0 <<  9) | ( 1 << 5) | ( 1)                   ; +18:アクセス日
        dw      0x0000                                          ; +20:(予約)
        dw      ( 0 << 11) | ( 0 << 5) | (0 / 2)                ; +22:更新時刻
        dw      ( 0 <<  9) | ( 1 << 5) | ( 1)                   ; +24:更新日
        dw      2                                               ; +26:先頭クラスタ
        dd      FILE.end - FILE                                 ; +28:ファイルサイズ
```

正：

0x00

P.665　一番下のコード　囲み部分の一行を挿入

正：

prog/src/44_to_real_mode/boot.s

```
        jmp     0:.real                 ; CS = 0x0000;
.real:  mov     ax, 0x0000              ;
        mov     ds, ax                  ; DS = 0x0000;
        mov     es, ax                  ; ES = 0x0000;
        mov     ss, ax                  ; SS = 0x0000;
        mov     sp, 0x7C00              ; SP = 0x7C00;
```

P.672 下部のコード

誤：

prog/src/45_fat_bios/**boot.s**

```
        ;-------------------------------------
        ; 割り込みマスクの設定（リアルモード用）
        ;-------------------------------------
        outp    0x20, 0x11                  ; out(0x20, 0x11); // MASTER.ICW1 = 0x11;
        outp    0x21, 0x08                  ; out(0x21, 0x20); // MASTER.ICW2 = 0x08;
        outp    0x21, 0x04                  ; out(0x21, 0x04); // MASTER.ICW3 = 0x04;
        outp    0x21, 0x01                  ; out(0x21, 0x05); // MASTER.ICW4 = 0x01;

        outp    0xA0, 0x11                  ; out(0xA0, 0x11); // SLAVE.ICW1  = 0x11;
        outp    0xA1, 0x10                  ; out(0xA1, 0x28); // SLAVE.ICW2  = 0x10;
        outp    0xA1, 0x02                  ; out(0xA1, 0x02); // SLAVE.ICW3  = 0x02;
        outp    0xA1, 0x01                  ; out(0xA1, 0x01); // SLAVE.ICW4  = 0x01;
```

正：

; out(0x21, 0x01); // MASTER.ICW4 = 0x01;

P.674　図 26-1 の下の一段落

誤：

ファイル名の検索は、ディレクトリエントリの先頭 11 バイトに記録されているファイル名と比較していけばよいのですが、1 つのセクタには 32 のディレクトリエントリしか含まれていません。このため、ルートディレクトリをセクタごとに読み込んで、読み込んだセクタ中にあるディレクトリエントリとファイル名との比較を繰り返さなくてはいけません。

正：

ファイル名の検索は、ディレクトリエントリの先頭 11 バイトに記録されているファイル名と比較していけばよいのですが、1 つのセクタには 16 のディレクトリエントリしか含まれていません。このため、ルートディレクトリをセクタごとに読み込んで、読み込んだセクタ中にあるディレクトリエントリとファイル名との比較を繰り返さなくてはいけません。

P.675　一番上のコード

誤：

prog/src/45_fat_bios/boot.s

```
        ;-------------------------------------
        ; ルートディレクトリのセクタを読み込む
        ;-------------------------------------
        mov     bx, 32 + 256 + 256              ; BX = ディレクトリエントリの先頭セクタ
        mov     cx, (512 * 32) / 512            ; CX = 512エントリ分のセクタ数
.10L:                                           ; do
                                                ; {
        ;-------------------------------------
        ; 1セクタ(16エントリ)分を読み込む
        ;-------------------------------------
        cdecl   read_lba, BOOT, bx, 1, 0x7600   ;   AX = read_lba();
        cmp     ax, 0                           ;   if (AX)
        je      .10E                            ;     break;
                                                ;
        ...                                     ;     // ファイルの検索処理
                                                ;
        loop    .10L                            ;
.10E:                                           ; } while (--CX);
```

正：

if (0 == AX)

P.679　一番上のコードおよびその上の一段落

誤：

この関数には、押下されたキーコードを引数として渡します。関数内では、キーコードの B7 を検査し、0 であれば BTS 命令で対応するビットをセット、1 であれば BTC 命令で対応するビットをクリアします。これらのビット操作命令は、第 1 オペランドにメモリを指定して、256 ビット（32 バイト）の配列としてアクセスすることが可能です。キーの押下状態は「.key_state」に保持します。

prog/src/modules/protect/ctrl_alt_end.s

```
ctrl_alt_end:
        ...
        ;-------------------------------------
        ; キー状態保存
        ;-------------------------------------
        mov     eax, [ebp + 8]                  ; EAX = key;
        btr     eax, 7                          ; CF  = EAX & 0x80;
        jc      .10F                            ; if (0 == CF)
        bts     [.key_state], eax               ; {
        jmp     .10E                            ;   // フラグセット
.10F:                                           ; } else {
        btc     [.key_state], eax               ;   // フラグクリア
.10E:                                           ; }
        ...

.key_state: times 32 db 0
```

正：

この関数には、押下されたキーコードを引数として渡します。関数内では、キーコードの B7 を検査し、0 であれば BTS 命令で対応するビットをセット、1 であれば BTR 命令で対応するビットをクリアします。これらのビット操作命令は、第 1 オペランドにメモリを指定して、256 ビット（32 バイト）の配列としてアクセスすることが可能です。キーの押下状態は「.key_state」に保持します。

prog/src/modules/protect/ctrl_alt_end.s

```
ctrl_alt_end:
        ...
        ;---------------------------------------
        ; キー状態保存
        ;---------------------------------------
        mov     eax, [ebp + 8]                  ; EAX = key;
        btr     eax, 7                          ; CY  = EAX & 0x80;
        jc      .10F                            ; if (0 == CY)
        bts     [.key_state], eax               ; {
        jmp     .10E                            ;   // フラグ・セット
.10F:                                           ; } else {
        btr     [.key_state], eax               ;   // フラグ・クリア
.10E:                                           ; }
        ...
.key_state: times 32 db 0
```