シリーズ・ソースコードを読む

Linux ブートアップ ソースコードの解析

第5回

芳之内 弘

BOOT 第2部 カーネルのセットアップを調べる

●setup.Sの解読

前回は、LILOがカーネルイメージをメモリに読み込み、セットアップコードの先頭に制御を移すところまでを調べました。今回は、このセットアップコードの働きを調べます。ここからは、LILOから離れて、カーネルのコードが実行を開始します。

これからは、32bitコードや、プロテクトモードへの移行のため、LILOにはない命令が出てきます。BIOSの機能も、同じくプロテクトモードを利用したものが出てきます。少しずつLinuxカーネルの内部に近づいていくのが実感されると思います。このセットアップコードを終了すると、次はいよいよ、プロテクトモードでの実行です。その途中の経緯を充分に味わって、楽しんでください。

setup.Sが終了すると、今度はlinux/arch/i386/boot/head.Sによる圧縮されたカーネルイメージが解凍され、次の段階のシステム設定が始まります。この行程の解読は、setup.Sが終了した後になります。

今回調べるソースコードは、LinuxカーネルのVer.2.2.14(本誌付録CD-ROM Vol.1「Linux Kernel History」にも収録)の「linux/arch/i386/boot/setup.S」と「同video.S」です。

●setup.Sの概要

今回は、プロテクトモードへの移行や割り込みの設定など、ハードウェアがらみの設定が出てきますので、少し難しく感じるかもしれませんが、よく理解できない場合は、読み捨ててください。そんなときは、全体の流れをつかんだ後で読み直すと、よく理解できるものです。video.Sをインクルードしているsetup.Sは、次のシーケンスでシステムの設定を行います。

①ディスクシステムをリセットします。
②セットアップコードが正しく読み込まれているかをチェックし、読み込み位置の修正を行います.
③不適切なローダの場合はそれを検出して表示し、無限ループに入ります。
④キーボードのリピートレートを最大に設定します。
⑤ディスプレイをチェックし、パラメータを設定します(video.Sが実行します)。
⑥ハードデスクのディスクパラメータを設定します。
⑦MCA(Micro Channel Architecture)の検出を行い、実装されている場合はパラメータを設定します。
⑧PS/2マウスの検出を行い、実装されている場合はパラメータを設定します。
⑨APM(Advanced Power Management)が設定されていて、BIOSがサポートしている場合はパラメータを設定します。
⑩カーネルが1Mbytes以下にロードされている場合は、0x1000から始まる絶対番地へ移動させます。
⑪セットアップコードも0x90000から始まる番地へ移動させます。
⑫プロテクトモードへ移行する準備として、GDT(メモリをアクセスするためのテーブル)とIDT(割り込みを制御するためのテーブル)を設定します。
⑬1Mbytes以上のメモリ空間を有効にするためのアドレスビット「A20」を有効にします。
⑭HMA(High Memory Area)への書き込み/読み出しテストを実行します。
⑮数値演算コプロセッサのリセットを実行します。

⑯ハードウェア割り込みのベクタをすべて変更し、割り込みをマスクします。
⑰プロテクトモードに移行してカーネルの次の段階に制御を渡します。

●setup.Sとvideo.Sが設定するカーネルのパラメータ

setup.Sは、カーネル本体に渡すためにパラメータを設定します。そのオフセットアドレスと内容を表1にまとめました。ベースアドレスのセグメントは「INITSEG」です。これは、カーネルオリジナルのブートローダが読み込まれたセグメントです。

●x86 CPUの32bitコード

これまで、LILOのブートローダでは、16bitサイズのコードで書かれていました。リアルモードのx86は、デフォルトではオペランドサイズやアドレスサイズはすべて16bitで表されます。しかし、今回からは32bitのサイズが使われ始めますので、各サイズでの命令とそのマシン語を比較して、その違いを比べましょう。

たとえば、16bitの

```
mov  ax,#0x1234
```

という命令のマシンコードは、

```
0xB8 0x34 0x12
```

で、このとき「AX=0x1234」となります。一方、32bitの

```
mov  eax,0x1234
```

というマシンコードは、

```
0x66 0xB8 0x34 0x12
```

で、「EAX=0x00001234」となります。

この32bitサイズのマシンコードの前に付加された"0x66"は「オペランドサイズプリフィックス」と呼ばれ、16bitサイズを32bitサイズに変換する働きがあります。ただし、この0x66は、デフォルトが32bitコードの場合は、逆に32bitサイズを16bitサイズに変換します。setup.Sから、この32bitのコードは使われます。さらに詳しく知りたい方は、記事末に上げた参考文献を参照してください(RESOURCE[1])。

●BIOSの説明

PC互換機をハードウェアリセットすると、まず最初にCPUは「0xF000:FFF0」の命令を読みに行きます。この0xF000のセグメントは、ROMのシステムBIOSの領域です。従って、PCのブートアップの最初のシーケンスは、すべてBIOSが担当しています。このシステムBIOSは、まず最初にビデオアダプタのビデオBIOSを読みに行き、ビデオの初期設定を実行します。

その後、システムBIOSはPOST(Power On Self Test)を実行します。そして、それ以外のシステムの設定を実行して、最後に、0x7C00にブートセクタを読み込んで制御を引き渡します。ビデオBIOSは、システムBIOSとは全く別のもので、アダプタ固有のBIOSです。このビデオBIOSについては、video.Sのところで説明しますので、ここではシステムBIOSについてのみ説明します。

それでは、いつものように、Setup.Sで使用される新しいBIOSコールについて表2にまとめて説明します。今回取り上げるBIOSコールは、そのほとんどが新しいサービスで、古いBIOSの説明書には記載されていません。そこで、「PhoenixBIOS 4.0 Revision 6 User's Manual」をダウンロードして調べました。ただし、このドキュメントにはかなり間違いがありますので、注意して読んでください。

●ソースコードの解読

Linuxカーネルの2.0から2.2でのsetup.Sの大きな変更は、MCAの検出と設定を追加したことです。そして、2.2から2.3での大きな変更は、アセンブラを「as86」から「gas」に変更したことです。内容的には、2.2.14と2.3.51はほとんど変わりませんので、ここでは、2.2.14を使って解読していきます。

以下のリストの左側の数字は、Linuxのprコマンドで

```
$ pr -n setup.S > setup.S.n
```

のようにして、setup.Sのファイルに付けた行番号です。

表1　setup.Sが設定するカーネルパラメータのオフセットアドレスと内容

オフセット	データの型	内容
0x0000	short	ビデオパラメータの内のカーソル位置
0x0002	short	1Mbytes〜16Mbytesのメモリサイズ(1Kbytes単位)
0x0004	28bytes	ビデオパラメータ(include/linux/tty.hを参照)
0x0040	16bytes	APM BIOS情報
0x0080	16bytes	HDD0のディスクパラメータ
0x0090	16bytes	HDD1のディスクパラメータ
0x00A0	16bytes	PCのシステム情報(MCA情報も含む)
0x01E0	long	1Mbytes以上のメモリサイズ(1Kbytes単位)
0x01FF	byte	PS/2マウス情報

表2　Setup.Sで使用されているBIOSコール

APMサービス：INT 0x15
①実装のチェック
設定：AX=0x5300
BX=0000
結果：AH=APM **メジャーバージョン番号(BCD)**
AL=APM **マイナーバージョン番号(BCD)**
BH=ASCII "P"
BL=ASCII "M"
CX=APM**情報**
bit0=1：16bit**ポートモードをサポート**
bit1=1：32bit**ポートモードサポート**
bit2=1：CPU IDLE、CPU**速度スローダウン**
bit3=1：BIOS Power Management**無効**
bit4=1：APM **動作していない**
②インターフェイス接続
設定：AX=0x5301
BX=0000
③プロテクトモード32bit**インターフェイス接続**
設定：AX=0x5303
BX=0000
結果：AX：　APM 32bit**モードセグメント**
(リアルモードセグメントのベースアドレス)
EBX：**BIOSへのエントリのオフセット**
CX：　APM 16bit**モードセグメント**
(リアルモードのセグメントアドレス)
DX：　APM**データセグメント(リアルモードセグメントアドレス)**
SI：　**BIOSのコードセグメントの長さ**
DI：　**BIOSのデータセグメントの長さ**
④インターフェイスの切断
設定：AX=0x5304
BX=0000
※これは取説では別の機能になっていますが、setup.Sのソースコードの内容より推察しました。取説が間違っているようです。
ビッグメモリサービス：INT 0x15
①ビッグメモリのサイズチェック
設定：AH=0xE8
AL=0x01
結果：CY=0：**このサービスをサポートしている**
AX：　**1Mbytesから16Mbytesまでのメモリサイズ(1Kbytes単位)**
BX：　**16Mbytesを越えるメモリサイズ(64Kbytes単位)**
CX：　**1Mbytesから16Mbytesまでの設定済みメモリサイズ(1Kbytes単位)**
CX：　**16Mbytesを超える設定済みメモリサイズ(64Kbytes単位)**
MCA実装のチェック：INT 0x15
①システムパラメータを得る
設定：AH=0xC0
結果：CY=0：　**MCAの実装**
(このMCAサービスは、ドキュメントには記載がありませんが、setup.Sのソースコードより推察しました。)
ES:BX：　**システム構成データへのポインタ**
byte 1-2：**構成データのバイト単位の長さ**(8)
byte 3：　**モデル(0xFC=AT)**
byte 4：　**サブモデル(01=AT)**
byte 5：　BIOS**バージョンベル(0)**
byte 6：　**機能情報**
bit 0 0 = **予約済み**
bit 1 0 = ISA**形式 I/O チャンネル**I
bit 2 0 = EDBA**は許可されない**
bit 3 0 = **「**Wait for external event**」が**
サポートされている
bit 4 1 = **「**Keyboard intercept**」(**INT 0x154F)
INT 0x09**で要求される**
bit 5 1 = **リアルタイムクロックが実装されている**
bit 6 1 = Second PIC present
bit 7 0 = **固定ディスク**BIOS**はDMAチャンネル3を使わない**
byte 7：　**予約済み**
byte 8：　**予約済み**

キーボードのリピートレートの設定：INT 0x16
設定：AH=0x03
AL=0x05
BL=0x00-0x1F**(30文字/秒 - 2文字/秒)**
BH=Delay rate
00:250ms
01:500ms
02:750ms
03:1000ms
ハードディスクドライブのタイプのチェック：INT 0x13
設定：AH=0x15
DL=**ドライブ番号(**0x80、0x81**)**
結果：AH= 00：**ドライブなし**
01：**メディア交換検出不可能**
02：**メディア交換検出可能**
03：**固定ディスク**
コンソールへの文字の出力：INT 0x10
設定：AH=0x0E
AL=**出力する文字**
BL=**フォアグランドの色(グラフィックモードの場合)**
システム情報(実装されている機器の情報)を得る：INT 0x11
結果：AX：**装置の情報**
bit**定義**
0：　**未使用**
1：　**数値演算コプロセッサがある場合に“**1**”**
2：　PS/2**マウスがあるとき“**1**”**
3：　**未使用**
4, 5：　**初期ビデオモード**
00=EGA/VGA
01=40x25 CGA
10=80x25 CGA
11=**モノクロ**
6, 7：　**ディスケットドライブの数**
00=1**ドライブ**
01=2**ドライブ**
10=3**ドライブ**
11=4**ドライブ**
8：　**未使用**
9-11：　**シリアルポートの数**
12：　**ゲームポートがある場合は“**1**”**
13：　**未使用**
14, 15：**パラレルポートの数**
リアルタイムクロックの時刻の読み取り：INT 0x1A
設定：AH=0x02
結果：CH=**時(BCD)**
CL=**分(BCD)**
DH=**秒(BCD)**
DL：　**“**01**”なら夏時間制選択、“**00**”夏時間制なし**
CF=0：**成功**
CF=1：**失敗**

リスト

```
 61 start:                                  ! 最初のスタートアドレスのラベル
 62         jmp    start_of_setup           ! ヘッダ部分を飛ばします

■ヘッダ(9020:0002から始まります)
 67                .ascii "HdrS"            ! セットアップヘッダのサイン
 68                .word  0x0201            ! ヘッダフォーマットのバージョン
 69                                         ! これは0x105より大きいこと
 71 realmode_swtch:      .word  0,0
                                            ! callf  * readmode_swtch 命令を実行すると、このアドレスで示すサブルー
                                            !  チンをコールします。最初のワードデータはオフセットで、次のデータがセグ
                                            ! メントになります。通常はこの値はともに0になっています。
 72 start_sys_seg:       .word  SYSSEG      !  読み込んだカーネルからセットアップ部分を除いた部分の最初のセグメント値
 73                .word  kernel_version    ! カーネルバージョンを示す文字列のポインタ
 77 type_of_loader:      .byte  0           ! ローダのタイプ
 80                                         ! 0xTV：T=0はLILO
 81                                         !       T=1はLoadlin
 82                                         !       T=2はbootsect-loader
 83                                         !       T=3はSYSLX
 84                                         !       T=4はETHERBOOT
 85                                         !       V=バージョン
 86 loadflags:                              ! ロードフラグ(RFU)未使用ビットはゼロにする
 87 LOADED_HIGH   = 1                       ! 1Mbytes以上にカーネルをロードしたとき
 88                                         ! bit-0=1にセットされる
 89 CAN_USE_HEAP  = 0x80                    ! ローダがheap_end_ptrをセットした場合はbit-7=1になる
 90

 93 #ifndef __BIG_KERNEL__                  ! コンパイルオプションです
 94                .byte  0x00              ! BIG_KERNELでないときは00
 95 #else
 96                .byte  LOADED_HIGH       ! BIG_KERNELの場合は‘1”
 97 #endif
 98
 99 setup_move_size: .word  0x8000          ! ローダが読み込んだセットアップコードは、後で0x9000から始まるように
                                            ! 移動されます。この移動に必要なサイズはローダが知っています。初期値
                                            ! は0x8000ワードに設定します。
105 code32_start:                           ! 32bitコードのスタードアドレス
107 #ifndef __BIG_KERNEL__                  ! コンパイラオプション
108                .long  0x1000            ! BIG_KERNELでない場合は0x1,000
109 #else
110                .long  0x100000          ! BIG_KERNELの場合は0x100,000
111 #endif
112 ramdisk_image:       .long  0           ! 読み込まれるRAMディスクのスタートアドレス
116 ramdisk_size: .long  0                  ! RAMディスクのサイズ(バイト単位)
117 bootsect_kludge:
118 .word   bootsect_helper,SETUPSEG        ! ここにはカーネルオリジナルのローダであるbootsect.Sをフロッピーディ
                                            ! スクドライブから起動して、カーネルを1Mbytes以上にロードする場合に使
                                            ! 用するサブルーチンの先頭アドレスがストアされています。
119 heap_end_ptr: .word  modelist+1024      ! modelistはセットアップコードの最後のアドレスなので、そこから
                                            ! 1024bytesをローカルヒープ領域に確保します。

■setuup.Sの始まり
124 start_of_setup:
126         mov    ax,#0x01500              ! ディスクタイプチェック
127         mov    dl,#0x81                 ! 2台目のHDDをテスト
128         int    0x13
129
130 #ifdef SAFE_RESET_DISK_CONTROLLER

■ディスクシステムをリセットする
132         mov    ax,#0x0000               ! ディスクシステムのリセット機能
133         mov    dl,#0x80                 ! すべてのディスクシステムをリセットする
```

```
134         int     0x13
135 #endif
136
```

■DSとCSを同じ値にする(現在のCS=#SETUPSEG=0x9020)

```
138         mov     ax,cs                   ! AX=CS
139         mov     ds,ax                   ! DS=CS=0x9020
```

■セットアップコードがすべてSETUPSEG=0x9020に読み込まれているかをチェック

```
141                                         ! ローダが5セクタ以降をSYSSEGにロードすることもあるので
142         cmp     setup_sig1,#SIG1        ! setup_sig1=0xAA55か？(最後から1ワード手前)
143         jne     bad_sig                 ! Noでbad_sig(184)へ
144         cmp     setup_sig2,#SIG2        ! setup_sig2=0x5A5Aか？(最後のワード)
145         jne     bad_sig                 ! Noでbad_sig(184)へ
146         jmp     good_sig1               ! 2ワードとも正しいのでgood_sig(222)へ
147                                         ! goot_sig1(180)を経由してgootd_sig(222)へ
148
```

■DS：SIでポイントするASCII文字列をコンソールへ出力する

```
150 prtstr:     lodsb                   ! AL=(DS:SI)
151         and     al,al                   ! ZFの設定：AL=0でZF=1になる
152         jz      fin                     ! ZF=1でfin(155)へ
153         call    prtchr                  ! ALの文字を主力する
154         jmp     prtstr                  ! 繰り返す
155 fin:    ret
156
```

■スペースのコンソールへの出力

```
158
159 prtsp2:     call    prtspc              ! スペースを2つ出力する
160 prtspc:     mov     al,#0x20            ! AL=0x20=" "
161
```

■コンソールへ1文字出力する(これは上のルーチンの一部でもあります)

```
164 prtchr:     push    ax                  ! AXを待避させる(ALは出力する文字)
165         push    cx                      ! CXを待避させる
166         xor     bh,bh                   ! BH=0
167         mov     cx,#0x01                ! CX=1(意味不明？)
168         mov     ah,#0x0e                ! AH=0x0E(コンソールへALの文字を出力する)
169         int     0x10
170         pop     cx                      ! CXを戻す
171         pop     ax                      ! AXを戻す
172         ret
```

■BEEP音のためBELを出力する

```
174 beep:   mov     al,#0x07                ! AL=0x07=BEL
175         jmp     prtchr                  ! BELを出力
176
177 no_sig_mess:  .ascii "No setup signature found ..."
                                            ! メッセージの内容
178             db      0x00                ! メッセージデータのターミネータ
179
180 good_sig1:
181         jmp     good_sig                ! good_sig(222)へ
182
```

■セットアップコードの残りをSETUPSEGへコピーする

```
184 bad_sig:
185         mov     ax,cs                   ! AX= #SETUPSEG=0x9020
186         sub     ax,#DELTA_INITSEG       ! AX=AX - #DELTA_INITSEG(0x20)、AX=0x9000
187         mov     ds,ax                   ! DS=AX=0x9000
188         xor     bh,bh                   ! BH=0
```

```
189         mov     bl,[497]                ! BL=セットアップコードのセクタ数
                                            ! (0x9000：497)にはカーネルのコンパイル時に書き込まれる
190         sub     bx,#4                   ! BX=BX - 4(LILOが既に４セクタをSETUPSEGに読み込んでいるので、残りの
                                            ! セクタ数を計算する)
191         shl     bx,#8                   ! BX=4セクタ*256bytes=SYSSEGに残っているセットアップコードのワード数
192         mov     cx,bx                   ! CX=BX(残っているワード数)
193         shr     bx,#3                   ! BX=BX / 8(ワード数をセグメント数に換算)
194         add     bx,#SYSSEG              ! BX=BX+0x1000(セットアップコードの残りはSYSSEG:0から読み込まれている)
                                            ! 従ってカーネルのスタートセグメントはセットアップコードの残りを差し
引くとBX+0x1000になる
195         seg cs                          ! CS=0x9020
196         mov     start_sys_seg,bx        ! このスタートセグメント値をstart_sys_segに書き込む
```

■残りのセットアップコードをSETUPSEGへコピーする

```
198
199         mov     di,#2048                ! DI=2048(SETUPSEGへはすでに4セクタ分読み込まれているので、それ以降
                                            ! に移動する)
200         sub     si,si                   ! SI=0(SYSSEGの最初のバイトをポイント)
201         mov     ax,cs                   ! AX=#SETUPSEG=0x9020
202         mov     es,ax                   ! ES=CS=0x9020
203         mov     ax,#SYSSEG              ! AX=#SYSSEG=0x1000
204         mov     ds,ax                   ! DS=0x1000
205         rep                             ! CX初期値=残りのワード数(192)でセット
206         movsw                           ! (DS:SI)を(ES:DI)へ１ワードコピー
207                                         ! CXの初期値数くり返す(SI=SI+2,DI=DI+2)
```

■正しくSETUPSEGへコピーされたかをチェックする

```
208         mov     ax,cs                   ! AX=#SETUPSEG=0x9020
209         mov     ds,ax                   ! DS=AX=0x9020
210         cmp     setup_sig1,#SIG1        ! setup_sig1=0xAA55か？
211         jne     no_sig                  ! Noでno_sig(216)へ
212         cmp     setup_sig2,#SIG2        ! setup_sig2=0x5A5Aか？
213         jne     no_sig                  ! Noでno_sig(216)へ
214         jmp     good_sig                ! 2ワードとも正しいのでgood_sig(222)へ
```

■セットアップコードが正しくないのでループする

```
216 no_sig:
217         lea     si,no_sig_mess          ! SI=no_sig_messのオフセット値
218         call    prtstr                  ! ES：SIでポイントする文字列を出力
219 no_sig_loop:                            ! すなわち'No setup signature found ...'を出力する
220         jmp     no_sig_loop             ! 無限ループ
```

■セットアップコードは正しいので次に進む

```
222 good_sig:
223         mov     ax,cs                   ! AX=#SETUPSEG=0x9020
224         sub     ax,#DELTA_INITSEG       ! AX=#INITSEG=0x9000
225         mov     ds,ax                   ! DS=AX=0x9000
226

282         mov     [2],ax                  ! これを[0x9000:0002]にストアする
283
```

■キーボードのリピートレイトを最大にセットする

```
286         mov     ax,#0x0305              ! キーボードリピートレイトの設定機能
287         xor     bx,bx                   ! BX=0(リピートレイト最大数)
288         int     0x16
289
```

■ビデオアダプタとパラメータをチェックする

```
291                                         ! 選択した(対話モードも可能)ビデオモードのパラメータをセットする
292                                         ! DS=0x9000
```

```
293         call    video                   ! ビデオアダプタのチェックと設定ルーチン
294
```

■1台目のハードディスクのパラメータを16バイトコピーする

```
297         xor     ax,ax                   ! AX=0
298         mov     ds,ax                   ! DS=0
299         lds     si,[4*0x41]             ! SI=1台目のHDDのパラメータのアドレス(BIOSパラメータテーブル)
300         mov     ax,cs                   ! AX=CS=0x9020
301         sub     ax,#DELTA_INITSEG       ! AX=#INITSEG=0x9000
302         push    ax                      ! AXを待避する(0x9000)
303         mov     es,ax                   ! ES=AX=0x9000
304         mov     di,#0x0080              ! DI=0x0080
305         mov     cx,#0x10                ! CX=0x10(16バイトコピーする)
306         push    cx                      ! CXを待避(0x10)
307         cld                             ! SI,DIは増加する方向で繰り返し実行する
308         rep                             ! CXの初期値の回数(16)繰り返す
309         movsb                           ! [DS:SI] → [ES:DI]へ1byteコピーする
310                                         ! SI=SI+1、DI=DI+1
```

■2台目のハードディスクのパラメータを16bytesコピーする

```
312
313         xor     ax,ax                   ! AX=0
314         mov     ds,ax                   ! DS=AX=0
315         lds     si,[4*0x46]             ! SI=2台目のHDDのパラメータのアドレス
316         pop     cx                      ! 306行目でpushしたCXを戻す(CX=0x10)
317         pop     es                      ! 302行目でpushしたAX(0x9000)をESへセット
318         mov     di,#0x0090              ! DIは2台目HDDのパラメータアドレス
319         rep                             ! CXの初期値=0x10の回数繰り返す
320         movsb                           ! [DS:SI] → [ES:DI]へ1byteコピーする
321
```

■2台目のディスクドライブが実装されているかをチェックする

```
323
324         mov     ax,#0x01500             ! AX=0x1500(ドライブタイプのチェック)
325         mov     dl,#0x81                ! DL=0x81(HDDの2台目)
326         int     0x13                    ! BIOSコール実行
327         jc      no_disk1                ! CY=1でドライブは実装されていないのでno_disk1(330)へ
328         cmp     ah,#3                   ! 実装されているドライブはハードディスクか？
329         je      is_disk1                ! Yesでis_disk1(340)へ
330 no_disk1:
```

■HDD No2がないときパラメータテーブルをクリヤする

```
331         mov     ax,cs                   ! AX=CS=0x9020
332         sub     ax,#DELTA_INITSEG       ! AX=#INITSEG=0x9000
333         mov     es,ax                   ! ES=0x9000
334         mov     di,#0x0090              ! DI=0090(パラメータテーブルのアドレス)
335         mov     cx,#0x10                ! CX=0x10(繰り返し数は16)
336         xor     ax,ax                   ! AX=0
337         cld                             ! DI,SIは繰り返しで増加する方向にセット
338         rep                             ! CXの初期値(0x10)回数繰り返す
339         stosb                           ! [ES:DI]=0、DI=DI+1
340 is_disk1:
341
```

■MCA(Micro Channel)バスをチェックする
■MCAが検出されると、[0x9000:0xA0]へシステム情報をセットする

```
343         mov ax,cs                       ! AX=#SETUPSEG=0x9200
344         sub ax,#DELTA_INITSEG           ! AX=#INITSEG=0x9000
345         mov ds,ax                       ! DS=0x9000
346         mov ds,ax                       ! 意味不明？
347         xor ax,ax                       ! AX=0(初期値は0にする)
```

```
348       mov [0xa0], ax                ! [0x9000：0xA0](テーブル長)= 0
                                        ! 0x9000:0xA0：setup.Sのシステム記述テーブルの先頭アドレス
349       mov ah, #0xc0                 ! AH=0xC0：システムパラメータを得る
350       stc                           ! CY=1にセット(MCA検出でCY=0になるので)
351       int 0x15                      ! システム構成データのアドレス=[ES:BX]
                                        ! これは通常システムBIOSのアドレスになる
352       jc no_mca                     ! CY=1でMCA検出されないのでno_mcs(371)へ
353       push ds                       ! DS=0x9000を待避
354       mov ax,es                     ! AX=ES(351でセットしたセグメント値)
355       mov ds,ax                     ! DS=ES
356       mov ax,cs                     ! AX= #SETUPSEG=0X9020
357       sub ax, #DELTA_INITSEG        ! AX= #INITSEG=0x9000
358       mov es,ax                     ! ES=0x9000
359       mov si,bx                     ! SI=システム構成データのオフセット値
                                        ! これでアドレスDS:DIは行番号351のES:BXと同じになる
360       mov di,#0xa0                  ! DI=0xA0
361       mov cx,(si)                   ! CX=構成データの長さ(通常は8bytes)
362       add cx,#2                     ! CX=CX+2
363       cmp cx,#0x10                  ! CX < 0x10か？
364       jc sysdesc_ok                 ! CXが0x10より小さいときはsysdesc_ok(366)へ
365       mov cx,#0x10                  ! CX=0x10(最初の16bytesのみをコピー)
366 sysdesc_ok:
367       rep                           ! CXの初期値の回数次の命令を繰り返す
368       movsb                         ! [DS:SI] → [ES:DI]へ1byteコピー
369       pop ds                        ! DS=0x9000(353でpush)
370
371 no_mca:
```

■PS/2マウスを検出する
■[0x9000:0x1FF]はsetup.SのPS/2マウスパラメータ

```
375       mov   ax,cs                   ! AX=#SETUPSEG=0x9020
376       sub   ax,#DELTA_INITSEG       ! AX=#INITSEG=0x9000
377       mov   ds,ax                   ! DS=0x9000
378       mov   [0x1ff],#0              ! [0x9000:0x1FF]=0 (初期値は0にする)
379       int   0x11                    ! 装置を検出する
380       test  al,#0x04                ! bit 2=1？(PS/2マウスが検出？)
381       jz    no_psmouse              ! ZF=1は検出せずでno_psmouse(383)へ
382       mov   [0x1ff],#0xaa           ! パラメータ0xAAを書き込む
383 no_psmouse:
```

BOOT おわりに

次回は、いよいよプロテクトモードの切り替えを行なっている部分を取り上げますので、x86のハードウェアを少し詳しく説明します。ご期待ください。

訂正

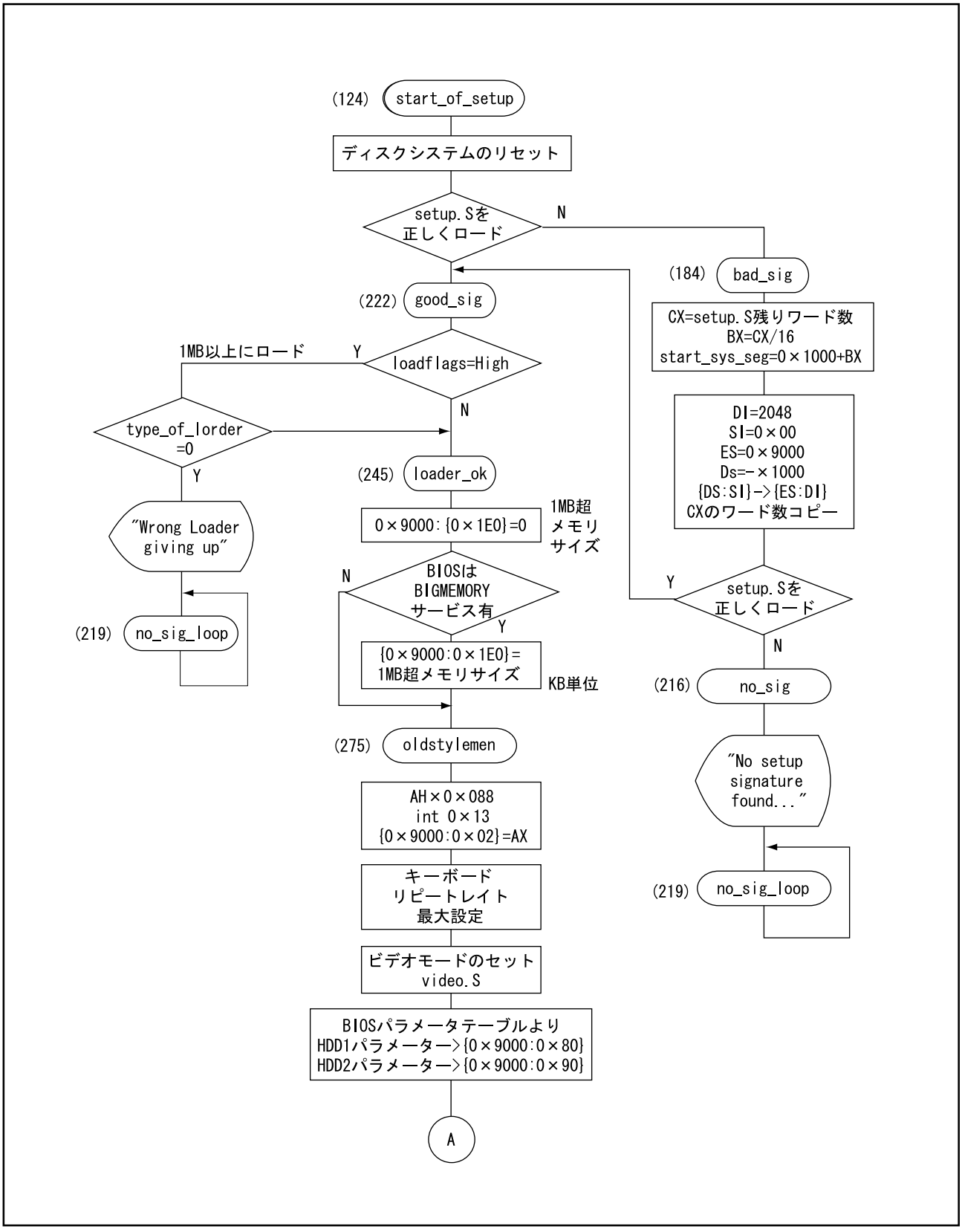
2000年7月号のフローチャート3で、最後の端子が「head.S」となっていますが、これは正しくは「setup.S」です。お詫びして訂正させていただきます。 （編集部）

RESOURCE

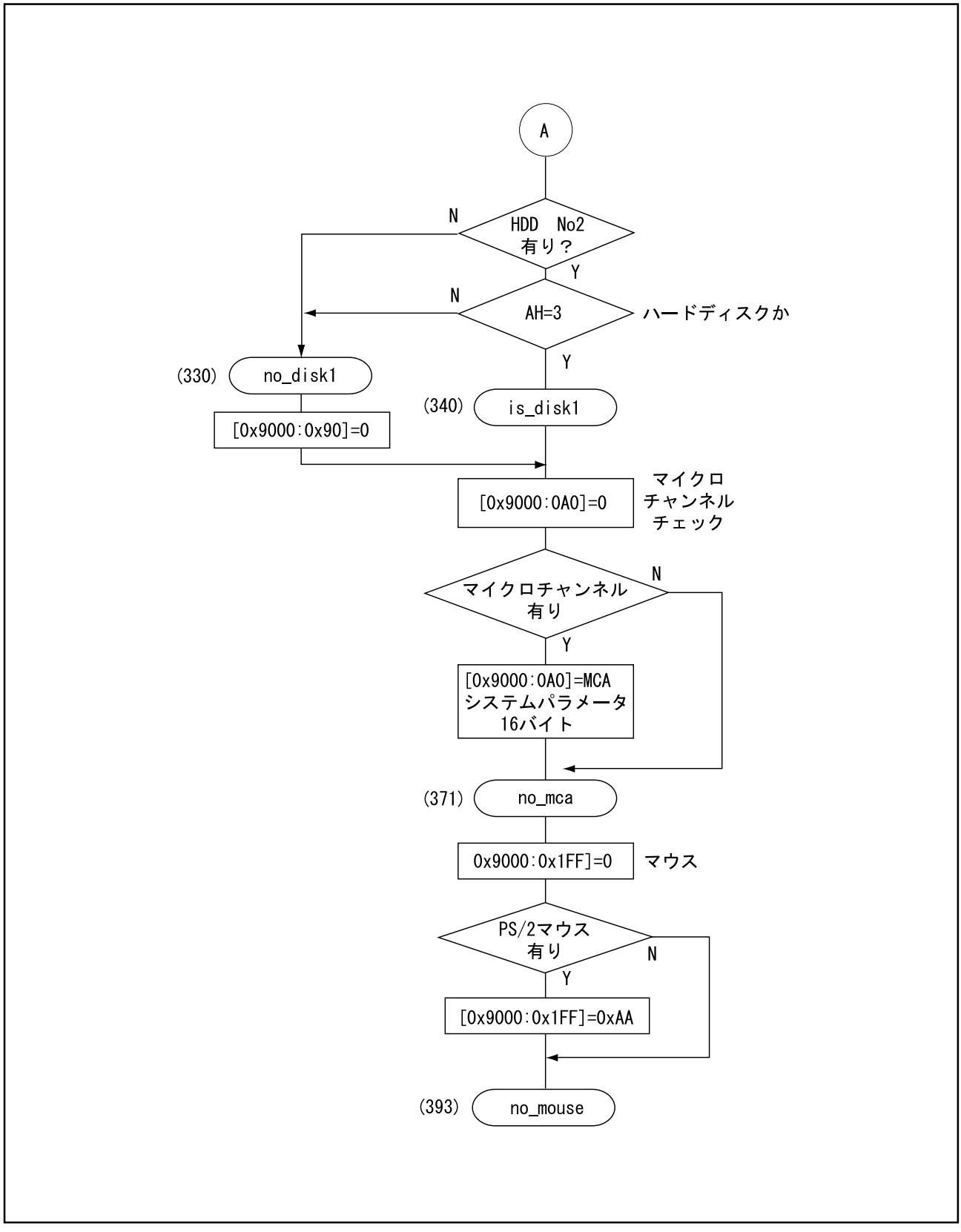
[1] 86 CPUのプロテクトモードに関する参考文献
「初めて読む486」
（蒲池輝尚著／アスキー出版）
「80486の使い方」
（W.B.スルヤント著／オーム社）
「Inter Architecture Software Developer's Manual Volume 3」
（Intel）

[2] PC互換機のアーキテキュチャに関する参考文献
「The Programmer's PC Sourcebook」
（Thom Hogan著、SE編集部訳、Microsoft Press）

[3] PC互換機のシステムBIOSに関する参考文献
「Phenix BIOS 4.0 Rivision 6 User's Manual」
（Phoenix Technologies LTD）
「BIOS Enhanced Disk Driver Specification version.3.0 Rev.0.8」
（Phoenix Technologines LTD）


(124) start_of_setup
ディスクシステムのリセット
setup.Sを
正しくロード
N
(184) bad_sig
CX=setup.S残りワード数
BX=CX/16
start_sys_seg=0×1000+BX
DI=2048
SI=0×00
ES=0×9000
Ds=-×1000
{DS:SI}->{ES:DI}
CXのワード数コピー
Y
setup.Sを
正しくロード
N
(216) no_sig
"No setup
signature
found..."
(219) no_sig_loop
(222) good_sig
1MB以上にロード
Y
loadflags=High
N
type_of_lorder
=0
Y
"Wrong Loader
giving up"
(219) no_sig_loop
(245) loader_ok
0×9000:{0×1E0}=0
1MB超
メモリ
サイズ
N
BIOSは
BIGMEMORY
サービス有
Y
{0×9000:0×1E0}=
1MB超メモリサイズ
KB単位
(275) oldstylemen
AH×0×088
int 0×13
{0×9000:0×02}=AX
キーボード
リピートレイト
最大設定
ビデオモードのセット
video.S
BIOSパラメータテーブルより
HDD1パラメーター>{0×9000:0×80}
HDD2パラメーター>{0×9000:0×90}
A

フローチャート1


A
HDD No2
有り？
N
Y
AH=3
ハードディスクか
N
Y
(330) no_disk1
[0x9000:0x90]=0
(340) is_disk1
[0x9000:0A0]=0
マイクロ
チャンネル
チェック
マイクロチャンネル
有り
N
Y
[0x9000:0A0]=MCA
システムパラメータ
16バイト
(371) no_mca
0x9000:0x1FF]=0
マウス
PS/2マウス
有り
N
Y
[0x9000:0x1FF]=0xAA
(393) no_mouse

フローチャート2