

宮崎県の学校現場における情報処理の実際と問題点Ⅰ

－成績処理Ⅰ－予備調査において発見された問題点と解決方法

大 坪 勝 郎

The Realities and Problems of Information Processing in the Schools of Miyazaki Prefecture —Processing of Results— Problems and Solutions Identified in the Preliminary Survey

Katsuro OTSUBO

調査の目的

近年、小学校においてコンピュータを操作できる教員のニーズが高まってきた。宮崎県においても同様であるが現在小学校の学校現場においてどのような利用がなされているのか、どのような問題点をかかえているのか実際に調査をして可能な限りにおいて解決策を提案していきたい。

学校現場において考えられるのは、成績処理・児童の情報管理・教材等の作成が主であると考えられるが設備等を全学的に準備出来ればアンケート調査・インターネット端末の設置なども可能である。実際には設備（パソコンも含む）の問題で利用範囲も違ってくるし教員の能力やそれに応じたソフトウェアの確保や作成など学校による違いというのは様々であると考えられる。そこで数校に協力していただき利用範囲や問題点等の予備調査をおこなうこととなった。

調査の方法

今回の調査は宮崎県全体の調査をおこなうための準備段階であるので10校の小学校のデータを集めた。宮崎市の市街地の小学校10校を無作為抽出し直接訪問してシステム構成も同時に見せていただいたので調査用紙の回収率は100%である。実際に成績処理をおこなっている先生方の話を聞くことが出来た。

調査の内容

調査内容が広範囲にわたると詳細な結果が期待できないので今回は成績処理に的を絞った調査をおこなった。調査項目として考えられるのは

- ① 成績処理のパソコン利用
- ② 成績処理の範囲
- ③ 成績処理の時期
- ④ 成績処理の内容
- ⑤ 成績処理の利用
- ⑥ 成績処理の問題点

等がある。さらに

- ① 成績処理のパソコン利用

成績処理をパソコンを利用しておこなっているか、あるいは過去におこなっていたか、また、現在や過去においてどのような問題点を抱えているか。ハードウェア・ソフトウェア・人的資源の問題が考えられる。

- ② 成績処理の範囲

対象として個人担当部分の処理、学年対象、全学対象がある。

- ③ 成績処理の時期

対象によっても異なると思われるが頻繁におこなう場合とある程度まとまった段階での分析をおこなう場合が考えられる。

- ④ 成績処理の内容

単なる得点集計や偏差値の計算、グラフ化等が考えられる。

- ⑤ 成績処理の利用

利用については指導上の参考や様々な利用が考えられる。

- ⑥ 成績処理の問題点

先入観をなくすため特に質問項目を限定しない。

などの点が考えられる。以上の点に気をつけて調査項目を決定した。

調査の結果

調査の結果、システム構成としてはスタンド・アローンの構成でWindows95ベースの標準的なシステムが多かった。システム構成は図1のとおりである。

成績処理等の学校業務には十分なシステムであると思われる。ソフトウェア環境は、最近では統合環境のパッケージソフトウェアが多く利用されているようでワードプロセッサや表計算・データベース程度のものは普及しているようだ。

調査結果は表1のようになった。

今回の結果では成績処理をパソコン利用でおこなっている学校は多いものの個人所有のパソコンを利用した小規模なものが多く、指導上の参考程度の利用であった。内容的にも得点の集計のみにとどまるものが多かったが一部学年単位での集計を行っているところもあり参考となる問題点も浮かんできた。

組織的におこなう上で一番の問題点と思われるのは、漢字コードの相違であった。機種の違いにより一部の記号や文字のコードが異なっている。数値データ上は問題はないが今後問題となってく

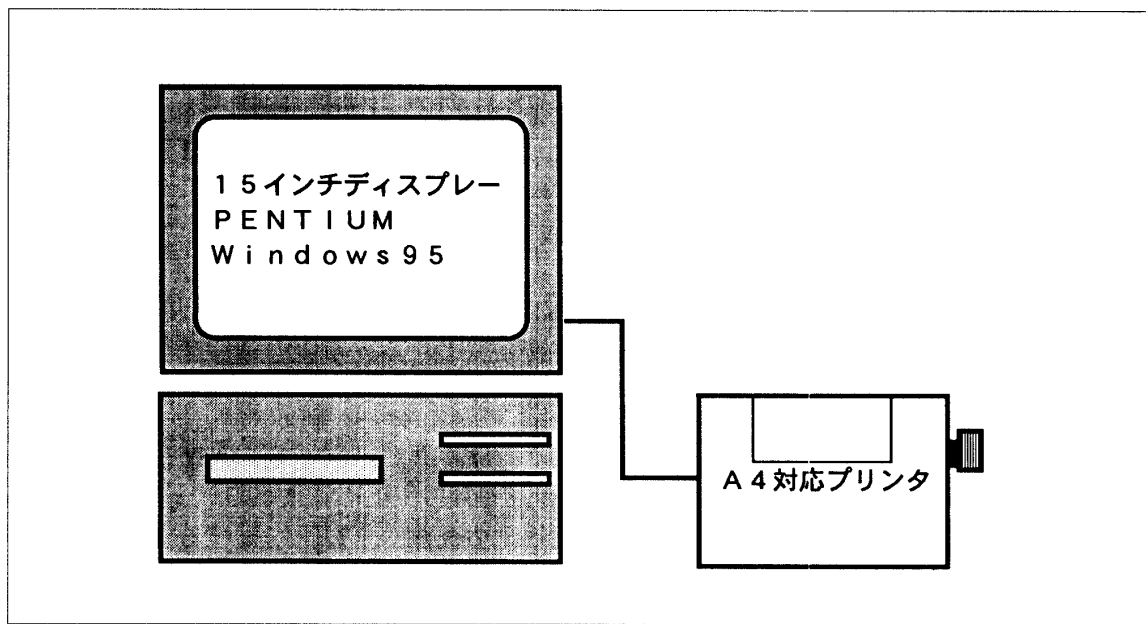


図1 標準的システムの構成

表1

質 問	記 号 ・ 番 号	回答数
1	a	8
	1	2
	2	
	3	
	4	
	5	
	6	
	1	
	2	
	3	
	4	
	5	
2	a	6
	b	
	c	2
	d	
3	a	8
	b	
	c	
	d	
	e	
4	a	
	b	8
	c	
	d	
	e	
5	a	7
	b	
	c	
	d	
	e	1

の一つであるといえる。予想されたソフトウェア間のデータ引き渡しや媒体の違いによるハードウェアに起因する障壁はみられなかった。

その他の問題としてフロッピーがよく壊れるというのがあった。ハードウェア（フロッピーディスクドライブユニット）の微妙な違いにより、たとえ同一メーカーの同一機種であってもフロッピーによるデータの受け渡しでフロッピーが壊れるというのは起こりうることである。おそらくフォーマット情報の上にデータを誤記してしまうという現象が起きていると思われる。

今回の調査内容は多くの学校に対しては焦点のずれたものとなってしまう。本格的な調査をおこなうにあたっては

- 成績処理はパソコンでおこなっている学校が多い
- 小規模な単位でおこなわれている

等の点を前提として成績処理の方法やソフトウェアに焦点を絞ったものにする必要があるだろう。また、様々な問題点に対しても解決策の提案できるものは可能な限り提案していきたい。今回の問題点に対しても解決案を検討してみた。

〔問題点1〕 漢字コードの相違

今回指摘のあったIBMとNECのパソコン間での漢字コードの相違について調べてみると以下の点で相違がみられた。

- FA55h～FA5Ah
- 849Fh～84BEh
- FA40h～FA49h

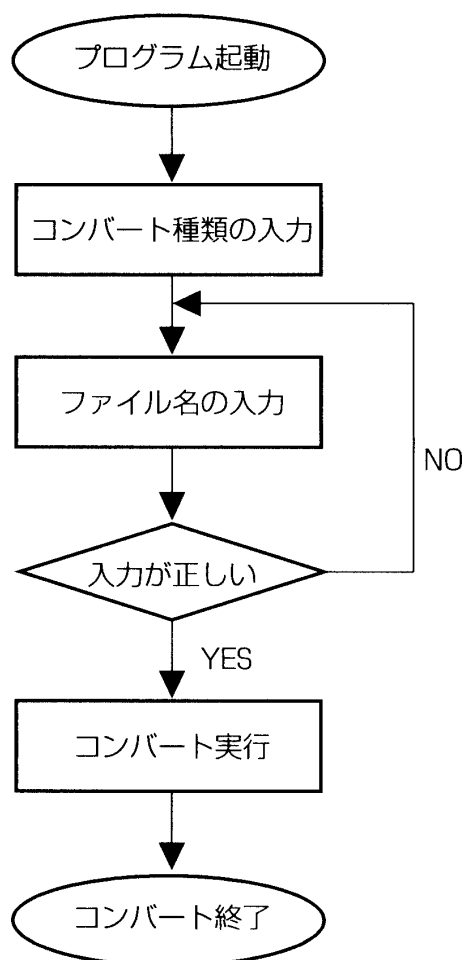
これは、Windows上であればフォントでも解決できることであるが操作上の知識が必要となるので適切ではない。理想的には双方向のコンバータが必要となる。市販の物を探してみたが見あたらないので、とりあえずプログラミングすることとした。

プログラムは単にコードの相違を双方向にコンバートするものであるがコードをよく分析してみると複雑に入り乱れていることがわかった。DOS上でのテキストレベル双方向コンバータとなった。プログラムの操作概要・ソースプログラム・使用上の注意点を以下に挙げる。

〔プログラム操作概要〕

操作は下記フローチャート手順による。

操作概念フローチャート



[プログラムソース]

MACRO ASSEMBLER

ソース・テキスト

```

dosint  MACRO  function ;; Call the DOS function
        mov    ah,function
                ;; Set function number in AH
        int     21h
        ENDM

error    MACRO  errnum ;; Display error and exit
        mov     dx,OFFSET err&errnum
        dosint  09h
        mov     al,errnum ;; return code of errnum
        dosint  4Ch      ;; Quit
        ENDM

col      MACRO  colnum ;; Display color and exit
        mov     dx,OFFSET col&colnum
        dosint  09h
        ENDM

data     SEGMENT word public 'DATA'
ibm_flg  DB      0
prompt   DB      0Dh,0Ah,0Ah,'ファイル名 : $'
msg      DB      '*IBM ←→ PC-9800          漢字コード変換'
        DB      ' Ver 1'
        DB      0Dh,0Ah,0Ah
        DB      '  1. IBMから98コードへ変換',
                0Dh,0Ah
        DB      '  2. 98からIBMコードへ変換'
        DB      0Dh,0Ah,0Ah
        DB      ' [s$'
        DB      ' '
msg3     DB
msg2     DB      ' [u [2K',0Dh,'  作業番号 [] [2D'
        DB      '$'
ad1      DW      0
ad2      DW      0
col1     DB      27,'[34m$'
col2     DB      27,'[31m$'
col3     DB      27,'[35m$'
col4     DB      27,'[32m$'
col5     DB      27,'[36m$'
col6     DB      27,'[33m$'
col7     DB      27,'[37m$'
err1     DB      'Can''t access file',0Dh,0Ah,'$'
err2     DB      'I/O error',0Dh,0Ah,'$'
err3     DB      'ファイルが大きすぎます',
                0Dh,0Ah,'$'

file_h   DW      1 DUP(?)
f_big    DW      1 DUP(?)
namebuf  DB      15h,?
                ; Maximum length of file name
fname    DB      15h DUP(?) ; is 15h (21d)
buffer   DB      10h DUP(?) ; Buffer size is 0FD00h (2048d)

data     ENDS

PUBLIC  prompt,namebuf,fname,buffer,err1,err2,err3
PUBLIC  get_file,open_file,ok,buff_read,done
PUBLIC  quit,word_c,file_h,ibm_flg,msg,msg2,msg3
PUBLIC  col1,col2,col3,col4,col5,col6,col7,ad1,ad2

stack    SEGMENT word stack 'STACK'
        DB      100h DUP (?)

stack    ENDS

code      SEGMENT byte public 'CODE'
        ASSUME cs:code,ds:data

start:    mov     ax,data
        mov     ds,ax ; Load data segment address

; *****
; type select
; 1.....ibm -> pc
; 2.....pc  -> ibm
; *****

        mov     dx,OFFSET msg
        dosint  9
        mov     dx,OFFSET msg2

re_ent:   dosint  9
        dosint  1
        mov     dx,OFFSET msg3
        cmp     al,32h
        ja      re_ent
        cmp     al,31h
        jb      re_ent
        je      flg_jump
        mov     bx,OFFSET ibm_flg
        mov     BYTE PTR[bx],1

flg_jump:

; ***** select end *****

        mov     dx,OFFSET prompt
        dosint  09h
get_file: mov     dx,OFFSET namebuf
        dosint  0Ah
        mov     si,dx
        mov     bl,BYTE PTR [si+1]
        mov     BYTE PTR [si+bx+2],0

        mov     dl,0Ah
        dosint  02h

open_file: mov     dx,OFFSET fname
        mov     al,2h
        dosint  3Dh
        jnc     ok

access:   error    1

ok:       mov     si,OFFSET file_h
        mov     DS:[si],ax
        mov     bx,ax
        mov     dx,OFFSET buffer
io_loop:  mov     cx,1
        dosint  3Fh

buff_read: jc      io_err
        cmp     ax,0
        je      done

```

```

        jmp      c_str
too_big: error    3
        jmp      done
io_err:  error    2      ; Error macro
        jmp      done
c_str:   mov      cx,0
        mov      bx,OFFSET ad1
        mov      [bx],cx
        mov      dx,1
        mov      bx,OFFSET ad2
        mov      [bx],dx
        mov      bx,OFFSET ibm_flg
        cmp      BYTE PTR[bx],0
        je       c_pc
        call     rword_c
        jmp      done
c_pc:    call     word_c
done:    dosint   3Eh
        ; Close (file handle already in BX)
        mov      dl,7
        dosint   2
quit:    xor      al,al      ; else set 0 for return code
        dosint   4Ch      ; Return to DOS function

; *****
; *****
; *****
;
;               sub area
;
; *****
; *****
; *****

;***** check *****

word_c:  nop
        mov      dl,7
        dosint   2
pre_c:   mov      bx,OFFSET buffer
        cmp      BYTE PTR[bx],0E0h
        jnb      c1
        cmp      BYTE PTR[bx],9Fh
        ja       c0
        cmp      BYTE PTR[bx],80h
        jnb      c1

; *****
        cmp      BYTE PTR[bx],1Ah
        jne      c0
        jmp      c_end
; *****

c0:      jmp      c_ext
c1:      mov      bx,OFFSET file_h
        mov      ax,[bx]
        mov      bx,ax
        mov      dx,OFFSET buffer
        inc      dx
        mov      cx,1
        dosint   3Fh
        mov      bx,OFFSET buffer
        cmp      BYTE PTR[bx],0FAh
        jne      c2
        cmp      BYTE PTR[bx+1],4Ah
        jnb      c2
        cmp      BYTE PTR[bx+1],53h
        ja       c2
        sub      BYTE PTR[bx],73h
        add      BYTE PTR[bx+1],0Ah
        jmp      col_chg
c2:      cmp      BYTE PTR[bx],0FAh
        jnb      c3
        cmp      BYTE PTR[bx],0FBh
        ja       c3
        cmp      BYTE PTR[bx+1],5Ch
        jnb      c3
        cmp      BYTE PTR[bx+1],0FCh
        ja       c3
        sub      BYTE PTR[bx],0Dh
        sub      BYTE PTR[bx+1],1Ch
        jmp      col_chg
c3:      cmp      BYTE PTR[bx],0FBh
        jne      c3_2
        cmp      BYTE PTR[bx+1],40h
        jnb      c3_2
        cmp      BYTE PTR[bx+1],5Bh
        jna       c3_3
c3_2:    cmp      BYTE PTR[bx],0FCh
        jne      c4
        cmp      BYTE PTR[bx+1],40h
        jnb      c4
        cmp      BYTE PTR[bx+1],4Bh
        ja       c3_4
c3_3:    sub      BYTE PTR[bx],0Eh
        add      BYTE PTR[bx+1],0A1h
c3_4:    jmp      col_chg
c4:      mov      si,OFFSET c_tbl
        push     cx
        mov      cx,CS:[si]
        inc      si
        inc      si
c4_str:  push     si
        add      si,1
        mov      al,[bx]
        cmp      al,CS:[si]
        jb       c4_nxt
        add      si,2
        mov      al,[bx]
        cmp      al,CS:[si]
        ja       c4_nxt
        sub      si,3
        mov      al,[bx+1]
        cmp      al,CS:[si]
        jnb      c4_nxt
        add      si,2
        mov      al,[bx+1]
        cmp      al,CS:[si]
        ja       c4_nxt

```

	add	si,6		dec	dx
	mov	al,CS:[si+1]		cmp	dx,0FFFFh
	cmp	al,1		jne	nxt2
	mov	al,CS:[si]		dec	cx
	je	c4_2			
	sub	[bx],al	nxt2:	mov	bx,OFFSET file_h
	jmp	c4_3		mov	ax,[bx]
c4_2:	add	[bx],al		mov	bx,ax
				mov	al,0
c4_3:	add	si,2		dosint	42h
	mov	al,CS:[si+1]		mov	bx,OFFSET file_h
	cmp	al,1		mov	ax,[bx]
	mov	al,CS:[si]		mov	bx,ax
	je	c4_4		mov	dx,OFFSET buffer
	sub	[bx+1],al		mov	cx,1
	jmp	c4_5		dosint	40h
c4_4:	add	[bx+1],al			
				mov	bx,OFFSET ad2
c4_5:	pop	si		mov	dx,[bx]
	pop	cx		sub	bx,2
	jmp	col_chg		mov	cx,[bx]
				inc	dx
c4_nxt:	pop	si		adc	cx,0
	add	si,12		clc	
	loop	c4_str		mov	[bx],cx
	pop	cx		add	bx,2
				mov	[bx],dx
c5:	mov	si,OFFSET spc_tbl		mov	bx,OFFSET buffer
	push	cx		mov	al,[bx+1]
	mov	cx,CS:[si]		mov	[bx],al
	inc	si			
	inc	si	c_ext:	mov	bx,OFFSET ad1
				mov	cx,[bx]
c5_str:	push	si		mov	bx,OFFSET ad2
	add	si,1		mov	dx,[bx]
	mov	al,[bx]		dec	dx
	cmp	al,CS:[si]		cmp	dx,0FFFFh
	jnz	c5_nxt		jne	ext2
	dec	si		dec	cx
	mov	al,[bx+1]	ext2:	mov	bx,OFFSET file_h
	cmp	al,CS:[si]		mov	ax,[bx]
	jnz	c5_nxt		mov	bx,ax
	add	si,2		mov	al,0
	mov	ax,CS:[si]		dosint	42h
	mov	[bx],ah			
	mov	[bx+1],al		mov	bx,OFFSET file_h
	pop	si		mov	ax,[bx]
	pop	cx		mov	bx,ax
	jmp	col_chg		mov	dx,OFFSET buffer
c5_nxt:	pop	si		mov	cx,1
	add	si,4		dosint	40h
	loop	c5_str			
	pop	cx		mov	bx,OFFSET file_h
				mov	ax,[bx]
c_user:	cmp	BYTE PTR[bx],0EFh		mov	bx,ax
	jb	no_c		mov	dx,OFFSET buffer
	cmp	BYTE PTR[bx],0F9h		mov	cx,1
	ja	no_c		dosint	3Fh
no_c:					
col_chg:	nop			mov	bx,OFFSET ad2
				mov	dx,[bx]
c_nxt:	mov	bx,OFFSET ad1		sub	bx,2
	mov	cx,[bx]		mov	cx,[bx]
	mov	bx,OFFSET ad2		inc	dx
	mov	dx,[bx]		adc	cx,0

	clc			cmp	BYTE PTR[bx+1],0FCh
	mov	[bx],cx		jna	rc3_3
	add	bx,2			
	mov	[bx],dx	rc3_2:	cmp	BYTE PTR[bx],0EEh
	jmp	pre_c		jne	rc4
				cmp	BYTE PTR[bx+1],0E1h
c_end:	cli			jb	rc4
	ret			cmp	BYTE PTR[bx+1],0ECh
				ja	rc4
; ***** check end *****					
; ***** rcheck *****					
rword_c:	nop		rc3_3:	add	BYTE PTR[bx],0Eh
	mov	di,7		sub	BYTE PTR[bx+1],0A1h
	dosint	2		jmp	rcol_chg
rpre_c:	mov	bx,OFFSET buffer	rc4:	mov	si,OFFSET c_tbl
	cmp	BYTE PTR[bx],0E0h		push	cx
	jae	rc1		mov	cx,CS:[si]
	cmp	BYTE PTR[bx],9Fh		inc	si
	ja	rc0	rc4_str:	inc	si
	cmp	BYTE PTR[bx],80h		push	si
	jae	rc1		add	si,5
; *****					
	cmp	BYTE PTR[bx],1Ah		add	si,2
	jne	rc0		mov	al,[bx]
	jmp	rc_end		cmp	al,CS:[si]
; *****					
rc0:	jmp	rc_ext		ja	rc4_nxt
rc1:	mov	bx,OFFSET file_h		sub	si,3
	mov	ax,[bx]		mov	al,[bx+1]
	mov	bx,ax		cmp	al,CS:[si]
	mov	dx,OFFSET buffer		jb	rc4_nxt
	inc	dx		add	si,2
	mov	cx,1		mov	al,[bx+1]
	dosint	3Fh		cmp	al,CS:[si]
	mov	bx,OFFSET buffer		ja	rc4_nxt
	cmp	BYTE PTR[bx],87h		add	si,2
	jne	rc2		mov	al,CS:[si+1]
	cmp	BYTE PTR[bx+1],54h		cmp	al,0
	jb	rc2		mov	al,CS:[si]
	cmp	BYTE PTR[bx+1],5Dh		je	rc4_2
	ja	rc2		sub	[bx],al
	add	BYTE PTR[bx],73h		jmp	rc4_3
	sub	BYTE PTR[bx+1],0Ah	rc4_2:	add	[bx],al
	jmp	rcol_chg			
rc2:	cmp	BYTE PTR[bx],0EDh	rc4_3:	add	si,2
	jb	rc3		mov	al,CS:[si+1]
	cmp	BYTE PTR[bx],0EEh		cmp	al,0
	ja	rc3		mov	al,CS:[si]
	cmp	BYTE PTR[bx+1],40h		je	rc4_4
	jb	rc3		sub	[bx+1],al
	cmp	BYTE PTR[bx+1],0E0h		jmp	rc4_5
	ja	rc3	rc4_4:	add	[bx+1],al
	add	BYTE PTR[bx],0Dh			
	add	BYTE PTR[bx+1],1Ch	rc4_5:	pop	si
	jmp	rcol_chg		pop	cx
rc3:	cmp	BYTE PTR[bx],0EDh		jmp	rcol_chg
	jne	rc3_2	rc4_nxt:	pop	si
	cmp	BYTE PTR[bx+1],0E1h		add	si,12
	jb	rc3_2			


```

        loop    rc4_str
        pop     cx

rc5:    mov     si,OFFSET spc_tbl
        push    cx
        mov     cx,CS:[si]
        inc     si
        inc     si

rc5_str: push    si
        add     si,3
        mov     al,[bx]
        cmp     al,CS:[si]
        jnz     rc5_nxt

        dec     si
        mov     al,[bx+1]
        cmp     al,CS:[si]
        jnz     rc5_nxt
        dec     si
        dec     si
        mov     ax,CS:[si]
        mov     [bx],ah
        mov     [bx+1],al

        pop     si
        pop     cx
        jmp     rcol_chg

rc5_nxt: pop     si
        add     si,4
        loop    rc5_str
        pop     cx

rno_c:  jmp     rc_nxt ; tmp-jmp-address

rcol_chg: nop

rc_nxt: mov     bx,OFFSET ad1
        mov     cx,[bx]
        mov     bx,OFFSET ad2
        mov     dx,[bx]
        dec     dx
        cmp     dx,0FFFFh
        jne     rnxt2
        dec     cx

rnxt2:  mov     bx,OFFSET file_h
        mov     ax,[bx]
        mov     bx,ax
        mov     al,0
        dosint  42h

        mov     bx,OFFSET file_h
        mov     ax,[bx]
        mov     bx,ax
        mov     dx,OFFSET buffer
        mov     cx,1
        dosint  40h

        mov     bx,OFFSET ad2
        mov     dx,[bx]
        sub     bx,2
        mov     cx,[bx]
        inc     dx
        adc     cx,0
        cld

        mov     [bx],cx
        add     bx,2
        mov     [bx],dx
        jmp     rpre_c

rc_end: cli
        ret

; ***** check end *****

; *****
; *
; *      下記のCHECK_TABLEの形式
; *
; *      テーブル内のデータ数
; *
; *      IBM下限, IBM上限, PC下限, PC上限,
; *      上位バイト, 下位バイト
; *
; *      上位・下位バイト ( 0 : sub/1:add ) +増減
; *
; *****

```

c_tbl	DW	1		DW	84A7h,86C6h	;+
	DW			DW	84A8h,86D6h	;+
			0FA40h,0FA49h,0EEEFh,0EEF8h,0Ch,1AFh			
spc_tbl	DW	47		DW	84A9h,86DEh	;+
				DW	84AAh,86A3h	;+
				DW	84ABh,86A5h	;+
	DW	0FA55h,0EEFAh	;・	DW	84ACh,86B1h	;+
	DW	0FA56h,0EEFBh	;・	DW	84ADh,86B5h	;+
	DW	0FA57h,0EEFCCh	;・			
	DW	0FA58h,878Ah	;株			
	DW	0FA59h,8782h	;No			
				DW	84AEh,86BDh	;+
				DW	84AFh,86B9h	;+
				DW	84B0h,86C5h	;+
	DW	0FA5Ah,8784h	;Tel	DW	84B1h,86D5h	;+
	DW	81BEh,879Ch	;U	DW	84B2h,86CDh	;+
	DW	81BFh,879Bh	;O			
	DW	81DAh,8797h	;△			
	DW	81DBh,8796h	;△			
; 10				DW	84B3h,86DDh	;+
	DW	81DFh,8791h	;△	DW	84B4h,86EDh	;+
	DW	81E0h,8790h	;+	DW	84B5h,86C2h	;+
	DW	81E3h,8795h	;√	DW	84B6h,86D1h	;+
	DW	0FA5Bh,879Ah	;+	DW	84B7h,86CAh	;+
	DW	81E7h,8792h	;f			
				DW	84B8h,86D9h	;+
				DW	84B9h,86E1h	;+
				DW	84BAh,86BFh	;+
	DW	849Fh,86A2h	;+	DW	84BBh,86D2h	;+
	DW	84A0h,86A4h	;+	DW	84BCh,86C7h	;+
	DW	84A1h,86AEh	;+			
	DW	84A2h,86B2h	;+			
	DW	84A3h,86BAh	;+	DW	84BDh,86DAh	;+
; 20				DW	84BEh,86E4h	;+
	DW	84A4h,86B6h	;+			
	DW	84A5h,86BEh	;+	code	ENDS	
	DW	84A6h,86CEh	;+	END	start	

[プログラム注意点]

- ・誤操作のプロテクションを掛けていないので確実に動作する。
- ・ファイル名はディスクリプタ、拡張子まで明示しなくてはならない。
- ・ファイル自身を変更するので事前にバックアップをしておく。
- ・テキストレベルを意識して作成したので、その他の対象には使用しない。
- ・拡張漢字の領域は機種依存性が強いので今回は除外した。
- ・ユーザ定義文字は利用者が意識的に管理しなくてはならない。
- ・旧機種でも対応できるよう 8 0 8 6 で記述しメモリもあまり必要としない。

[問題点 2] フロッピーがよく壊れる

実際に問題のフロッピーディスクを調べてみるとトラック 2 ～ 5 の間で異常が起きていた。これは記録密度の比較的低い領域であるので装置や技術的問題ではないことが判る。これはメーカー側の努力を望むしかないがいくつかの解決方法がある。一つは外部ドライブを使用することである。これならば共通のハードウェアを使用するので安定した電圧下では安定した動作をする。もう一つはイーサネット等の安価な簡易ネットを利用することである。これも直接的管理下にあるドライブのみの使用となるので安定した動作を得ることが出来る。

以上の 2 点に関する解決策は成績処理に限らず多方面で利用可能と思われるのでぜひ活用いただきたい。

今後はさらに学校現場の実態を把握し改善していくために調査を継続していきたい。学校現場の現状に合ったコンピュータシステムをソフトウェア面も含めて追求していくつもりである。最後に、たいへんお忙しい中調査にご協力していただいた先生方に感謝します。

参考文献

1984 ザ 8 0 8 6 ブック

広済堂産報出版

漢字コード一覧表 第4版 (N:GC18-2040-3)

日本IBM

[1998年11月30日受理]

参考資料：成績処理調査用紙

〔質問 1〕 パソコン等を利用した成績処理をおこなっていますか？

(a) している (b) していない (c) 以前はしていた

(b) していない とお答えの方、なぜですか？

- 1) 興味がない
- 2) 必要ない
- 3) 自由に使えるパソコンがない
- 4) ソフト（プログラム等）がない
- 5) 業者にまかせている
- 6) その他

(

)

(c) 以前はしていた とお答えの方、どうして中止したのですか

- 1) 担当者がいなくなった
- 2) パソコンが古くなった
- 3) あまり効果がない
- 4) プログラムに問題があった
- 5) ディスクに空きがなくなった
- 6) その他

(

)

〔質問 2〕 成績処理の対象についてお答えください

- (a) 自分のクラスのみ
- (b) 数クラス
- (c) 学年対象
- (d) 全校規模

〔質問 3〕 成績処理の時期についてお答えください

- (a) 時々、必要に応じて
- (b) 定期試験以外の小テスト
- (c) 定期試験の一部（期末等）
- (d) 全定期試験
- (e) その他

(

)

〔質問 4〕 成績処理の概要についてお答えください

- (a) 試験的におこなっている
- (b) 得点を集計している
- (c) 偏差値も計算する

(d)グラフ化等の工夫をしている

(e)その他

(

)

[質問 5] 成績処理の利用についてお答えください

(a)指導上の参考になっている

(b)実験段階でありあまり利用していない

(c)評価の参考になっている

(d)クラスごとの比較を行う

(e)その他

(

)

[質問 6] その他、成績処理の上でお困りの点がありましたらお書きください。