GAME65

パーソナルユースに，システムソフトにその威力を発揮する最速GAMEインタプリタ

GAME65を紹介します．一応3月には出来ていたのですがDebugのため発表が遅れました．PETを対象にしていますが，APPLE等他の65系マイコンへの移植法も解説します．

PETでのオペレーション

打ち込んだプログラムは走らせる前にセーブします．出来上がったテープは　LOAD RUNですぐGAMEインタープリタにとぶ様になっています．これは，

```
10  SYS1037
```

というBASICのプログラムも同時にロードされるためです．

プログラム1を入れてみて下さい．0[CR]でリストを見て，間違えた所はカーソルエディトで直します．＃＝1[CR]で実行させて下さい．画面の白黒が次々と反転すれば成功です．暇な人は同様のプログラムをBASICで書いて，スピードの比較をしてみて下さい．GAMEの速さが良く分かるでしょう．

ブレークは[STOP]と[RVS]キーを二つ同時に押して下さい．このブレーク判定はPETのインターラプト・ルーチンで行なっていますので，機械語にとんだ時も有効に動作します．PETにはリセットキーがないためこの機能をつけてあります．（これは隅井洋氏のアイデアによるものです．）

プログラムのセーブは，

```
>=$C08
```

で実行します．ラベル，Device＃の設定等はできません．ロードは，

```
>=$C06
```

です．このルーチンではプログラム末ポインタを設定しませんから　==（Search End命令）を実行した方が良いでしょう．

通常プログラム先頭アドレスは$C5Aに設定されていますが，GAMEでは複数のプログラムの共存は可能です．例えば，$D00から別のプログラムを書く時は，

```
=$D00[CR]
&=0[CR]
```

とします．（マルチステートメントにはできません．）

すでに別の場所に書かれているプログラムへ移る時は単に　=Address　だけで良く，プ

```
PROG. 1

10 V=$8000
20 K=0,499
30 V(K)=V(K)!$8080
40 @=K+1
50 #=20
```

6502 Programing Technic 1

まずは6502の特徴をつかめ

8080系のテクニックについては様々な所で紹介されており情報に不自由しませんし，6800系もほぼ同様です．さて6502はというと，あまりにPET，APPLE等のパーコンのイメージが強すぎ，機械語に関する情報はほとんどありません．

6502は命令数も少く単純ですぐ覚えられるものばかりですが，アドレシングでとまどう人も多い様です．

そこで何かの助けになればと思い，アドレシングを始めとし思いつくままに書いてみます．ななめ読みで構いませんので目を通してみて下さい．（基本的な事は3月号の特集に出ていますので全く知らない人はそちらを見て下さい．）

アドレシングについて

イミーディエイト，アブソリュート，ゼロページでつまづく人はいないでしょうから，ここではインデックス・アドレシングに話をしぼります．

6502は内部に16bitのレジスタを持っていませんから実際のアドレスはオペランドか，オペランドで示されるゼロページの内容で指定されます．Indexレジスタはそれを8 bitの範囲で修飾します．従って6502においてはデータの長さ，データ間の距離が256 byteを越えるか否かがプログラミングの際の決定的要因となります．ですからプログラムを作る前にデータの構造等を良く検討して，努めて256 byteでおさまる様にします．

さてABS，Indexアドレシングは2 byteのオペランドを1 byteのレジスタで修飾するもので，68やZ80とは本質的に違います．このアドレシングの特長はループが簡単に作れる事で，例えば80ではポインタの他にカウンタを用意しなければなりませんが，65ではレジスター本で済みます．

その反面256byte以上のデータが扱えない，汎用サブルーチンが使いにくいことは大きな欠点です．256byte以上のためには，後で述べるIndirectアドレシングを使うとして，汎用サブルーチンについてはいっそ考えを改め，必要な個所に専用のルーチンを組むように転向すべきでしょう．勿論Indirectなら汎用サブルーチンを作る事は可能ですが，メインルーチンでのポインタの設定に手間がかかりすぎて不便です．

例として「ある番地から，ある個数だけ文字をプリントする」というルーチンを考えて見ましょう．68や80では文句なしにサブルーチンにする所ですが，65では左の様にした方が有利な場合が多いのです．

```
    LDX  #LE          LDA  #CL
A:  LDA  C,X          STA  PL
    JSR  OUT          LDA  #CH
    DEX               STA  PH
    BNE  A            LDX  #LE
                      JSR  STR
```

また上の例ではスピードはあまり問題になりませんが，特にスピードが要求される場合には更に左のプログラムは有利になり，65の高速性が強調されます．

次にIndirectアドレシングですが，これにはIND，XとIND，Yの2種がありますが実際に使われるのはほとんど後者です．IND，Xが有効となるのは，ポインタを多く使う場合以外にはあまりなく，ちょっとしたtrickの際使われる程度です．

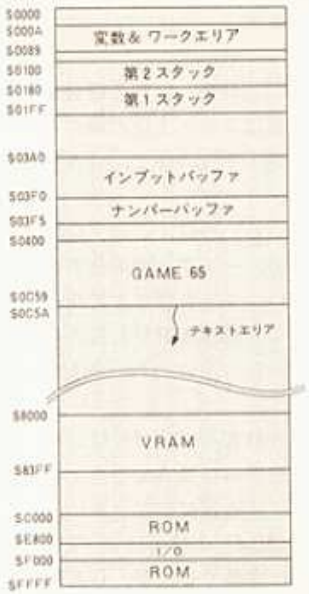
このアドレシングはゼロページとレジスタ間の複雑な修飾関係を理解すれば問題ないでしょう．他のアドレシングについては説明を省略します．

〈追加された演算子〉 AND OR Exclusive OR	 & ・ !	 論理積をとる. 論理和をとる. 排他的論理和をとる.
LOAD SAVE	＞＝$C06 ＞＝$C08	ＬＯＡＤ後サーチエンド命令＝＝を行うこと．32Kも同じアドレス 32Kも同じ.
プログラムの中断 (BREAK)	(RVS STOP)	左の2つのキーを同時に押す. 機械語ルーチンでも有効.
INPUT CHR	$	ＰＥＴのＧＥＴ文と同じ．ＣＲまで待たず，nullなら0をかえす.

GAME65 拡張機能

OUTPUT	($0BA2)	Accの内容をASCIIコードで出力．A，X，Yreg.は保持されなければならない.
INPUT	($0BAE)	インプットバッファに，入力された文字列をストアする．区切りに00を入れる．A，Xは保持しなくて良い．Yは00にクリア．インプットバッファの先頭アドレスを$40，$41にストアする必要がある.
GET	($0BDF)	Accに一文字ASCIIコードで入力．Xはクリア．Yは保持されなければならない.

入出力ルーチンの仕様



メモリマップ

ログラム末ポインタも正しく設定されます.

最後に注意を一つ．このＧＡＭＥ65はゼロページをかなり使います．それで一旦ＧＡＭＥを起動するとＢＡＳＩＣのワークエリアを壊してしまい，もうＢＡＳＩＣは正常に動作しません．テープのセーブ，ロード中にブレークをかけるとＢＡＳＩＣに戻ってしまいますから気をつけて下さい．誤ってブレークをかけてしまったら，一応　ＳＹＳ1055 CR として下さい．運が良ければ(！)，ＧＡＭＥのホットスタートに戻れます．16，32KのＰＥＴではモニタに入るかも知れませんが，あわてず　Ｇ　041F CR とします.

ＢＡＳＩＣに戻りたい時はＰＥＴの最強力インタラプトである電源スイッチを使うことにしましょう.

ＧＡＭＥ65の文法

ＧＡＭＥの文法については「ＧＡＭＥとは」を参照して下さい．ＧＡＭＥ65ではビット操作を容易にする為に次の3つの二項演算子を

6502 Programing Technic 2

頭を使って エレガントに

Trick　その1

「レジスタA，X，Yの内容を変えずに，メモリの内容が0か否かをチェックするプログラム」を書く必要ができたらどうしますか？「どうもこうも，スタックに内容を退避させれば良いんだろう」と次の様に書く人が大部分でしょう.

```
        PHA
        LDA  MEM
        BNE  NEQ
        PLA
      (処理A)
      (処理B)
NEQ   PLA
      (処理C)
```

間違いという訳では無いのですが，感心できません．データの一時的退避にスタックを使用したらすぐにもとに戻すべきなのに，間にブランチを入れるなどもってのほかと言えるでしょう．この場合，ブランチ先でのＰＬＡを忘れるケースが実に多く，またBからCへプログラムが続いている場合にはやはりスタックのレベルが狂います．もしこれがサブルーチンだと即暴走につながります.

慣れた人なら次の様なプログラムにするでしょう.

```
        DEC  MEM
        INC  MEM
        BNE  NEQ
          ⋮
```

こちらの方がエレガントでしょう．このテクニックは80系にも応用できます.

Trick　その2

「サブルーチンの Entry　Point によってレジスタに入れる数値を変える」という場合は以外と多いものです．さて，どうしますか.

```
        LDA  #$00
        JMP  COM
        LDA  #$0A
        JMP  COM
        LDA  #$0D
        JMP  COM
        LDA  #$20
COM     ⎫   (処理)
```

上の様なプログラムではまだまだです．65に無条件相対ブランチが無いからといって，何でも無条件ジャンプではメモリの無駄です.

状態があらかじめ分っているフラグは利用しましょう.

```
        LDA  #$00
        BEQ  COM
        LDA  #$0A
        BNE  COM
        LDA  #$0D
        BNE  COM
        LDA  #$20
COM     ⎫   (処理)
```

の様にするのがベターです．もし更にメモリを節約したい場合，少し遅くなっても構わないなら下の様にするのがベストでしょう.

```
        LDA  #$00
        FDB  $2C
        LDA  #$0A
        FDB  $2C
        LDA  #$0D
        FDB  $2C
        LDA  #$20
COM     ⎫   (処理)
```

ＦＤＢというのは疑似命令で，右にある数値を単にメモリに書込む機能があること，及び$2CはＢＩＴ命令のアブソリュート・アドレシングのオペコードであること，この2つだけを参考に読んで理解して下さい.

導入しています。

&　　.　　!

（AND）（OR）（EXOR）

2 byte変数は下位・上位の順に並んでおり，文番号は上位・下位の順で，GAME80と同じ形式です。

注意して欲しいのは，このプログラムはPET用のため，一文字入力の仕方が従来のGAMEと違い，キーを押すまで待ってくれません。言ってみればAPPLEのGET文とPETのそれとの違いです。（何も押してない時は0を返す）

従来のGAMEとコンパチにしたい方は，$FFE4を呼んだ後Accをチェックして，0ならループする様にして下さい。

なお%（Mod）の使用についてですが，VTLでは%は変数でしたが，GAMEでは単項演算子なのです。ですから以前に行なった割算の剰余を知りたい時，%だけでは駄目で%A とか %1 の様にダミーの項を付けて下さい。A／Bの余りは勿論 %（A／B）で構いません。

移植の実際

68のプログラムを65へ移すにあたって最大の問題はIndexアドレシングでしょう。機械的にやっていたのではすべてIndirectアドレシングになり，INXはJSR INCPNTに化けてしまいます。

そこで今回は，「一行の長さは決して256バイトを越えない」ことに注目し，一行の先頭アドレスを示すポインタをゼロページに作り行の内容を（IND），Yで取出すことにします。これによりスタックの内容も変わり，GOSUBの場合は先頭アドレスとYレジスタの3バイト，FORやDOの場合はそれに終値パラメータを加えた5バイトが1レベルとなります。

またGAME68では式の値がAccAとAccBに入りましたが，GAME65ではXreg，Accに入ります。（Xregが上位）

それからオリジナルと考えを異にしているのは，加減乗除の演算ルーチンです。オリジナルはAccA・BとIndexレジスタで示されるメモリの間で演算していますが，GAME65は演算の相手を固定しています。これはレジスタとメモリとスピードの節約のつもりですが成功しているのかは分りません。

ADR	8K → 16・32K
$0429	20 → 40（16K） 80（32K）
$0BD6	19 → 90
$0BD7	02 → 00
$0BDB	1A → 91
$0BDC	02 → 00
$0BF1	85 → 2E
$0BFB	85 → 2E
$0BFD	19 → 90
$0BFE	02 → 00
$0C02	1A → 91
$0C03	02 → 00
$0C06 ～はリストIIに変える。	

16K・32K PETへの変更一覧

他のシステムへの移植

$0400～$040CはPET以外のシステムでは不要です。$040D～$0B9Fまでを JMP，JSRに注意して移して下さい。頭のところでテキストの上下限を決めていますからメモリに合せて書き換えます。

$0BA0～$0BE8はシステムに合せ書き換る必要があります。$0BE9以降は不要です。

またバッファとして，数字出力のワークエリア（$03F0～$03F5），インプットバッファ（$03A0～$03EF）を使用しています。

$068Aからの3byteのNOPはブレークチェックへのJSRにして下さい。なお$0219～AはIRQベクトルですので無視して下さい。大変な仕事でしょうが，健闘を祈ります。

PETへのロード

8KPETへは後ろページのモニタを使ってそのまま打込んで下さい。32K，16KのPETはモニタをROMに持っていますのでそれを使用し（チェックサムは使わない）変更一覧を見ながら変更したデータを打ち込んで下さい。必ずRUNさせる前にセーブしておく事を忘れない様にして下さい。

GAMEで書いた$400からのダンプをするプログラムを載せておきます。ライン10の$400を変えればダンプ開始アドレスを変えられます。右端はチェックサムです。一画面分表示しキーを押すと次の画面を表示します。GAMEはこの種のプログラムは非常に簡単に出来るためシステムソフトには最適です。

```
0C00 E6 8D 91 00 58 60 18 24
0C08 38 20 F9 0B A2 00 86 9D
0C10 86 D1 E8 86 D4 B0 0D 20
0C18 22 F3 20 E6 F8 A5 96 29
0C20 10 D0 1A 60 A5 84 85 FB
0C28 A5 85 85 FC A6 56 A4 57
0C30 E8 D0 01 C8 86 C9 84 CA
0C38 A2 00 4C A4 F6 A9 3F 4C
0C40 A2 0B 00 00 00 00 00 00
0C48 00 00 00 00 00 00 00 00
0C50 00 00 00 00 00 00 00 00
```

ダンプリストII （変更点）

```
 10 A=$400
 20  J=1,25 /
 25 " "
 30 ??=A "   " C=0
 40 I=0,7
 50 ?$=A;0) " " C=C+A:0)
 60 A=A+1
 70 @=I+1
 80 "   " ?$=C @=J+1
 90 Q=$ ;=Q=0 #=90
100 #=20
```

ダンプ用プログラム