令和に取り残された

PC-9800 と機械語プログラミング

上羽 未栞（a.k.a. KusaReMKN）

2026-07-14

https://KusaReMKN.com/
mkn@kusaremkn.com
Twitter: @KusaReMKN

令和に取り残された
PC-9800 と機械語プログラミング

上羽 未栞（a.k.a. KusaReMKN）
2026-07-14
https://KusaReMKN.com/
mkn@kusaremkn.com
Twitter: @KusaReMKN

はじまるよ〜。
「令和に取り残された PC-9800 と機械語プログラミング」と題して、上羽未栞が発表するよ。

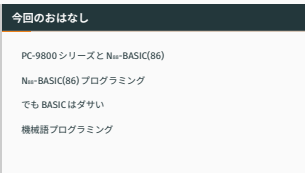
今回のおはなし

PC-9800 シリーズと N88-BASIC(86)

N88-BASIC(86) プログラミング

でも BASIC はダサい

機械語プログラミング



今回の発表の流れはこんな感じだよ。おおよそ 15 分くらいで進められたらいいな。

みかんちゃんについて

みかんちゃんについて

まずは自己紹介するよ〜。

自称・大天才美少女プログラミング初心者

「上羽 未栞（うわば みかん）」あるいは「KusaReMKN（くされみかん）」
みかんちゃんって呼んでね！

17(18) 歳の JK（超重要）
実はプログラマでもエンジニアでもない
普段はホラを吹いて生活している
古い計算機っぽいものが大好き

Twitter で思想を垂れ流すことが得意
https://kusaremkn.com/ も見てね

2026-07-14
令和に取り残された PC-9800 と機械語プログラミング
└みかんちゃんについて
└自称・大天才美少女プログラミング初心者

自称大天才美少女プログラミング初心者の上羽未栞だよ。みかんちゃんって呼ばれると大変喜ぶよ。
ｲﾁﾅﾅ歳の JK だよ。重要だよ。大天才とか偉ぶっているけれど、実際のところプログラマでもエンジニアでもないよ。普段は人々にホラを吹いて生活しているよ。古い計算機っぽいものが大好きで、その流れで黒電話とか電話網を作ったりとかして遊んでいるよ。

Twitter や自分のウェブサイトで思想を垂れ流すのが得意だよ。暇な人は眺めてみてね。

PC-9800 シリーズと N88-BASIC(86)

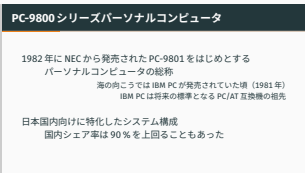
まずは今回の登場人物についてお話するよ〜

PC-9800 シリーズパーソナルコンピュータ

1982 年に NEC から発売された PC-9801 をはじめとする
パーソナルコンピュータの総称

海の向こうでは IBM PC が発売されていた頃（1981 年）
IBM PC は将来の標準となる PC/AT 互換機の祖先

日本国内向けに特化したシステム構成
国内シェア率は 90 % を上回ることもあった



今回使う PC-9800 についてお話するよ。PC-9800 シリーズパーソナルコンピュータは、1982 年に NEC から発売された PC-9801 をはじめとするコンピュータの総称だよ。これが出た一年前には、海の向こうで IBM PC が発売されていて、これは将来の標準となる PC/AT 互換機の祖先にあたるものだよ。

PC-9800 シリーズは、日本国内向けに特化したシステム構成をとり、世界標準とは別の道に振り切ったことで、国内シェア率は一時 90 % を上回ることもあったよ。

標準プログラミング言語 N88-BASIC(86)

PC-9800 シリーズで利用できる標準のプログラミング言語

旧来の機種に搭載された BASIC と高い互換性をもつ
- PC-8000 シリーズの N-BASIC
- PC-8800 シリーズの N88-BASIC

標準プログラミング言語 N88-BASIC(86)
PC-9800 シリーズで利用できる標準のプログラミング言語
旧来の機種に搭載された BASIC と高い互換性をもつ
PC-8000 シリーズの N-BASIC
PC-8800 シリーズの N88-BASIC

そんな PC-9800 シリーズのコンピュータで利用できる標準のプログラミング言語として N88-BASIC(86) があるよ。

これは、過去に NEC から発売されたコンピュータに搭載されていた BASIC と高い互換性をもっているよ。コンピュータのユーザは、過去のプログラミング資産の大部分を再利用できたよ。

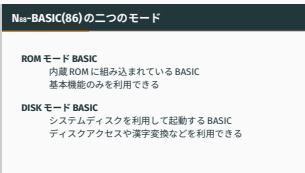
N88-BASIC(86)の二つのモード

ROM モード BASIC

内蔵 ROM に組み込まれている BASIC
基本機能のみを利用できる

DISK モード BASIC

システムディスクを利用して起動する BASIC
ディスクアクセスや漢字変換などを利用できる



N88-BASIC(86) には、ROM モード BASIC と DISK モード BASIC との二つのモードがあるよ。
ROM モード BASIC は、内蔵 ROM に組み込まれている BASIC で、システムディスクを必要とせずに起動するものだよ。これは、96 kB の ROM に格納されているもので、基本機能のみを利用できるものだよ。
DISK モード BASIC は、システムディスクを使って起動する BASIC だよ。これは、ROM モード BASIC の機能に加えて、ディスクアクセスや漢字変換などの機能を利用できるものだよ。
今回のお話は、ROM モード BASIC を前提に進めていくけれど、いづれのモードでも使えるものだよ。

N88-BASIC(86)プログラミング

ではね、N88-BASIC(86) をつかってプログラミングすることについて説明していくよ。

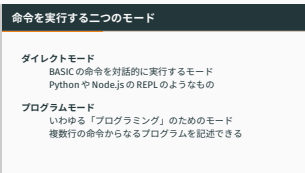
命令を実行する二つのモード

ダイレクトモード

BASIC の命令を対話的に実行するモード
Python や Node.js の REPL のようなもの

プログラムモード

いわゆる「プログラミング」のためのモード
複数行の命令からなるプログラムを記述できる



N88-BASIC(86) には、命令を実行する二つのモードがあるよ。
まずは、ダイレクトモードだよ。これは、BASIC の命令を対話的に実行するモードだよ。Python とか Node.js とかにある REPL 環境のようなものだよ。
もう一つのモードは、プログラムモードだよ。これは、いわゆる「プログラミング」をするためのモードだよ。複数行の命令からなるプログラムを記述するには、このモードを使うよ。

ダイレクトモードの利用

```
プロンプト①　Ok
入力するプログラム①　PRINT "hello, world"
上の命令の結果①　hello, world
プロンプト②　Ok
入力するプログラム②　PRINT 57*29
上の命令の結果②　 1653
プロンプト③　Ok
```


まずは、簡単なダイレクトモードの対話の例を見てみるよ。下線のついている部分は、ユーザの入力する部分だよ。
N88-BASIC(86)では、プロンプトとしてOkの文字が表示されるよ。これは、プログラムの入力を受ける準備ができていることを表しているよ。ここでプログラムを入力するよ。ここでは、表示するためにPRINT命令をつかって、順に二つの内容を表示しているよ。PRINT命令によってメッセージが表示されるよ。上の例では、指定した文字列"hello, world"が、下の例では、二つの数のかけ算の結果が表示されているよ。

こんな感じで、ダイレクトモードは、その場限りの計算だったり、覚えておく必要のない命令だったりを実行するために使われるよ。

プログラムモードの利用

```
プロンプト    Ok
              10 A = 57
プログラム    20 B = 29
              30 PRINT A*B
プログラムの実行    RUN
プログラムの実行結果     1653
プロンプト    Ok
```


プログラムモードの利用

```
プロンプト    Ok
              10 A = 57
プログラム    20 B = 29
              30 PRINT A*B
プログラムの実行    RUN
プログラムの実行結果     1653
プロンプト    Ok
```

続いて、プログラムモードの例を見てみるよ。例によって下線のついている部分は、ユーザの入力する部分だよ。
プログラムモードでは、命令に先立って行番号と呼ばれる数を入力するよ。入力された行番号と命令との組がメモリに格納されるよ。
入力されたプログラムを実行するには、RUN 命令を利用するよ。これによって、入力されたプログラムが、行番号の小さいものから順に実行されるよ。この例では、まず変数 A に 57 を代入し、変数 B に 29 を代入し、それらの積を表示しているよ。

変数の操作とか、分岐や反復を含む処理とかは、プログラムモードでプログラムを記述した上で実行することが多いよ。

ちょっと複雑な例: Fizz Buzz

```
10 FOR I = 1 TO 100
20 IF I MOD 3 > 0 AND I MOD 5 > 0 THEN PRINT I: GOTO 60
30 IF I MOD 3 = 0 THEN PRINT "FIZZ";
40 IF I MOD 5 = 0 THEN PRINT "BUZZ";
50 PRINT
60 NEXT
```


ちょっと複雑な例: Fizz Buzz

```
10 FOR I = 1 TO 100
20 IF I MOD 3 > 0 AND I MOD 5 > 0 THEN PRINT I: GOTO 60
30 IF I MOD 3 = 0 THEN PRINT "FIZZ";
40 IF I MOD 5 = 0 THEN PRINT "BUZZ";
50 PRINT
60 NEXT
```

1 から 100 までの Fizz Buzz をシミュレートするプログラムを紹介するよ。このプログラムは、(10) FOR 命令によって 1 から 100 までの繰り返しを行う中で、(20) 3 でも 5 でも割り切れない数についてはそのまま表示して 60 行目にジャンプし、(30) そうでない数については、3 で割り切れたら Fizz を表示し、(40) 5 で割り切れたら Buzz を表示し、(50) 改行し、(60) 次の数へ進みます。

こんな感じで複雑なプログラムを記述することもできるよ。

でも BASIC はダサい

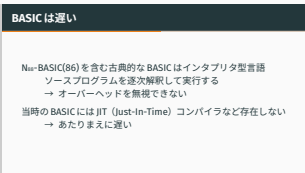
でも BASIC はダサい

とはいえ、BASIC はなんだかダサい気がするよ。

BASIC は遅い

N88-BASIC(86) を含む古典的な BASIC はインタプリタ型言語
　　ソースプログラムを逐次解釈して実行する
　　→ オーバーヘッドを無視できない

当時の BASIC には JIT（Just-In-Time）コンパイラなど存在しない
　　→ あたりまえに遅い



まず、BASIC は遅い点がダサいよ。
N88-BASIC(86) を含む古典的な BASIC は、ソースプログラムを逐次的に解釈して実行するインタプリタ型言語だよ。これは、プログラムを実行するにあたってのオーバーヘッドを無視できないので、遅くなりがちだよ。

この問題はインタプリタ型言語の宿命であるから、現代においては JIT コンパイラなどを利用して高速化が図られているよ。例えば、Python なんかは典型的な例だね。でも、当時の BASIC に JIT コンパイラなど存在しないので、あたりまえに遅いよ。うん、遅いよ。

BASIC は名前もダサい

BASIC: Beginners’ All-purpose Symbolic Instruction Code
　　和訳すると「初心者向け汎用記号命令コード」

「初心者向けの BASIC なんて使ってるわけないでしょ」
　　とか言われかねない

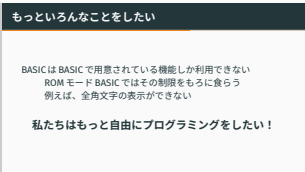
BASIC は名前もダサい
BASIC: Beginners’ All-purpose Symbolic Instruction Code
和訳すると「初心者向け汎用記号命令コード」
「初心者向けの BASIC なんて使ってるわけないでしょ」
とか言われかねない

さらに、BASIC は名前もダサいよ。BASIC は Beginners’ All-purpose Symbolic Instruction Code のバクロニムになっていて、これは「初心者向け汎用記号命令コード」と和訳されるよ。みかんちゃんはプログラミング初心者なので、まぁこれくらいでもちょうど良いんだけど、それにしたってちょっとダサいよ。

もっといろんなことをしたい

BASIC は BASIC で用意されている機能しか利用できない
　　ROM モード BASIC ではその制限をもろに食らう
　　例えば、全角文字の表示ができない

私たちはもっと自由にプログラミングをしたい！



そして、なにより、BASIC は BASIC で用意されている機能しか利用できないよ。この制約は、基本機能しか利用できない ROM モード BASIC では特に大きなものになるよ。ROM モード BASIC では、全角文字（漢字やかな）の入力を行う方法が存在しないので、工夫なしにはこれらを表示できないよ。もっと自由なプログラミングをする方法が求められているよ。

機械語プログラミング

2026-07-14

令和に取り残された PC-9800 と機械語プログラミング
└機械語プログラミング

というわけで、自由なプログラミングを求めて機械語を喋れる身体になってみよう。

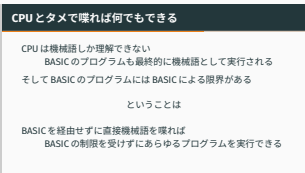
CPU とタメで喋れば何でもできる

CPU は機械語しか理解できない
　　BASIC のプログラムも最終的に機械語として実行される

そして BASIC のプログラムには BASIC による限界がある

ということは

BASIC を経由せずに直接機械語を喋れば
　　BASIC の制限を受けずにあらゆるプログラムを実行できる



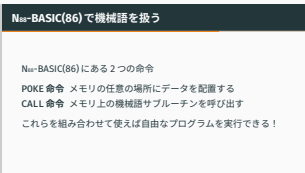
どうして？ と思う人もいるかもしれないので、雑に説明しておくよ。コンピュータの頭脳である CPU は機械語しか理解できないよ。ので、BASIC のプログラムも最終的には機械語として実行されるよ。
いま、BASIC のプログラムは BASIC による限界があるので、これが自由なプログラミングを邪魔しているよ。
ということは、BASIC を経由せずに直接機械語を喋れれば、BASIC の制限を受けずにあらゆるプログラムを実行できるといえそうだよ。

N88-BASIC(86)で機械語を扱う

N88-BASIC(86)にある2つの命令

POKE 命令 メモリの任意の場所にデータを配置する
CALL 命令 メモリ上の機械語サブルーチンを呼び出す

これらを組み合わせて使えば自由なプログラムを実行できる！



というわけで、N88-BASIC(86)の上で機械語を扱う方法について考えてみるよ。N88-BASIC(86)には、POKEとCALLと二つの命令があるよ。POKEはメモリの任意の場所にデータを配置する命令、CALLはメモリ上の機械語サブルーチンを呼び出す命令だよ。これらを使って、メモリ上に機械語プログラムを配置して、それを実行することで、自由なプログラムを実行できるよ。

え、でも待って！

Q: 機械語プログラムはどうやって用意するの？

よくある質問だよ。

え、でも待って！

Q: 機械語プログラムはどうやって用意するの？

A: DATA 命令と READ 命令とを使うとデータの列を簡単に扱える
機械語の列を DATA 命令で記述すれば OK

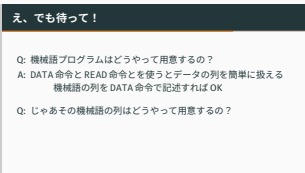
よくある質問だよ。

え、でも待って！

Q: 機械語プログラムはどうやって用意するの？

A: DATA 命令と READ 命令とを使うとデータの列を簡単に扱える
機械語の列を DATA 命令で記述すれば OK

Q: じゃあその機械語の列はどうやって用意するの？



よくある質問だよ。

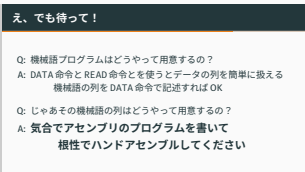
え、でも待って！

Q: 機械語プログラムはどうやって用意するの？

A: DATA 命令と READ 命令とを使うとデータの列を簡単に扱える
機械語の列を DATA 命令で記述すれば OK

Q: じゃあその機械語の列はどうやって用意するの？

A: **気合でアセンブリのプログラムを書いて**
根性でハンドアセンブルしてください



よくある質問だよ。

そして出来上がったものがこちらです

```
10 CLEAR,&H8000
20 DEF SEG=&H8000
30 FOR P=0 TO 49
40 READ C
50 POKE P,C
60 NEXT P
70 F%=0
80 CALL F%
```

```
100 DATA &H60, &H06, &H1E, &H0E
105 DATA &H1F, &HC4, &H3E, &H16
110 DATA &H00, &HBE, &H1A, &H00
115 DATA &HB9, &H0C, &H00, &HFC
120 DATA &HF3, &HA5, &H1F, &H07
125 DATA &H61, &HCF, &H00, &H00
130 DATA &H00, &HA0, &H68, &H00
135 DATA &H65, &H00, &H6C, &H00
140 DATA &H6C, &H00, &H6F, &H00
145 DATA &H20, &H00, &H20, &H24
150 DATA &HA0, &H24, &H13, &H26
155 DATA &H93, &H26, &H01, &H2A
160 DATA &H81, &H2A
```


そして出来上がったものがこちらです
10 CLEAR,&H8000
20 DEF SEG=&H8000
30 FOR P=0 TO 49
40 READ C
50 POKE P,C
60 NEXT P
70 F%=0
80 CALL F%
100 DATA &H60, &H06, &H1E, &H0E
105 DATA &H1F, &HC4, &H3E, &H16
110 DATA &H00, &HBE, &H1A, &H00
115 DATA &HB9, &H0C, &H00, &HFC
120 DATA &HF3, &HA5, &H1F, &H07
125 DATA &H61, &HCF, &H00, &H00
130 DATA &H00, &HA0, &H68, &H00
135 DATA &H65, &H00, &H6C, &H00
140 DATA &H6C, &H00, &H6F, &H00
145 DATA &H20, &H00, &H20, &H24
150 DATA &HA0, &H24, &H13, &H26
155 DATA &H93, &H26, &H01, &H2A
160 DATA &H81, &H2A

で、できあがったものがこちらになります。左側の BASIC らしい BASIC の部分は、メモリに機械語サブルーチンを展開してそれを呼び出すよ。READ 命令によって後続の DATA 命令のデータ（機械語サブルーチン）をひとつひとつ読み出して、POKE 命令によってそれをメモリに配置しているよ。最後に CALL 命令によってその機械語サブルーチンを呼び出すよ。右側のアホの DATA 命令の羅列は、機械語で記述されたプログラムだよ。

機械語プログラムの内容

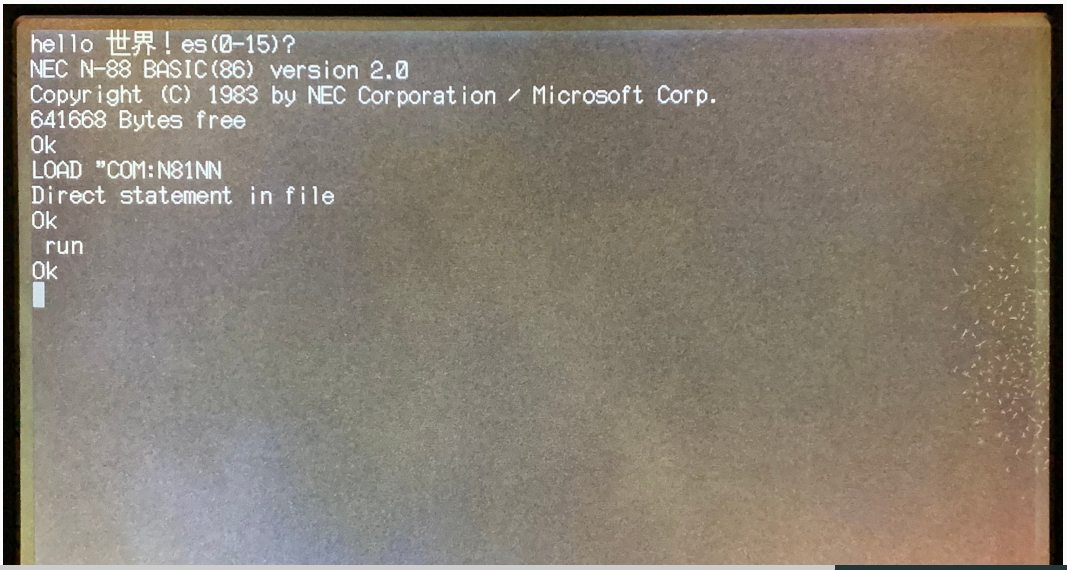
機械語プログラムでは、

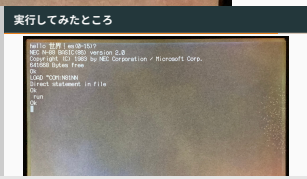
- レジスタの内容を退避し、
- VRAM 領域にメッセージ「hello 世界！」を転送し、
- レジスタの内容を復帰し、
- BASIC に処理を戻している

つまりちょっと大袈裟に「hello, world」しているだけ

機械語プログラムの内容が気になるところだと思うんだけど、これは、レジスタの内容をスタックに退避してから、VRAM 領域にメッセージを転送して、そしてレジスタの内容を復帰して BASIC に処理を戻しているだけだよ。つまり、ちょっと大袈裟に hello, world しているだけなんだけど、全角文字である「世界！」を表示できているところがウマみだよ。

実行してみたところ





実行してみたところがこれだよ。画面ではちょっとインチキをしていて（LOAD 命令の部分）、プログラムを打ち込む代わりにシリアルポートから流し込んでいるよ。きちんと画面の上の部分に「hello 世界！」を表示できていることがわかるよ。

おわりに

PC-9800 と機械語プログラミング

PC-9800 シリーズの標準プログラミング言語 N88-BASIC(86)
　　システムディスクがなくてもプログラミングできる

ROM モード BASIC は機能に制限がある
　　プログラミングの自由度が制限されてしまう
　　→ 機械語プログラムでこの問題を克服

機械語プログラムで記述された「hello 世界！」プログラム
　　ROM モード BASIC 単体では難しい漢字の表示を実現

PC-9800 と機械語プログラミング
PC-9800 シリーズの標準プログラミング言語 N88-BASIC(86)
システムディスクがなくてもプログラミングできる
ROM モード BASIC は機能に制限がある
プログラミングの自由度が制限されてしまう
→ 機械語プログラムでこの問題を克服
機械語プログラムで記述された「hello 世界！」プログラム
ROM モード BASIC 単体では難しい漢字の表示を実現

ソースコードをじっくり見たいなら

KusaReMKN/
hellosekai.bas

N-88 BASIC(86) と機械語とを組み合わせて世界に挨拶する

1 Contributor　0 Issues　1 Star　0 Forks

おわりです

おわりだよ〜。

このスライドについて

Written in July 2026.

Permanent ID of this document: c7e71f36986189b7.

このスライドについて

Written in July 2026.
Permanent ID of this document: c7e71f36986189b7.
Copyright © 2026 KusaReMKN.
特記無き場合、プログラムやソースコードは MIT License で、
それ以外のコンテンツは CC-BY 4.0 で利用可能です。
一部の画像には別のライセンスが適用されるかもしれません。