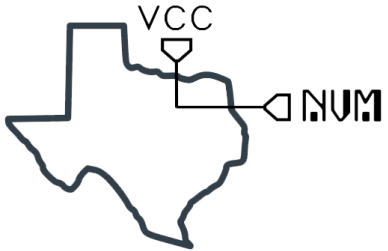
VCF Southwest

Presented by the National Videogame Museum

Industry Speaker and Technical Training Series

NATIONAL VIDEOGAME MUSEUM

Exploring the IBM 5100



From the SCAMP prototype to the PALM instruction set

Steve Lewis

June 25th, 2023

Topics

20min

PART 1: Review of IBM 5100-series systems

PART 2: Q&A with IBM 5100 pioneers

- Examples of IBM 5110 users
- CORE International
- Aspects about the IBM 5110 BASIC implementation

20min

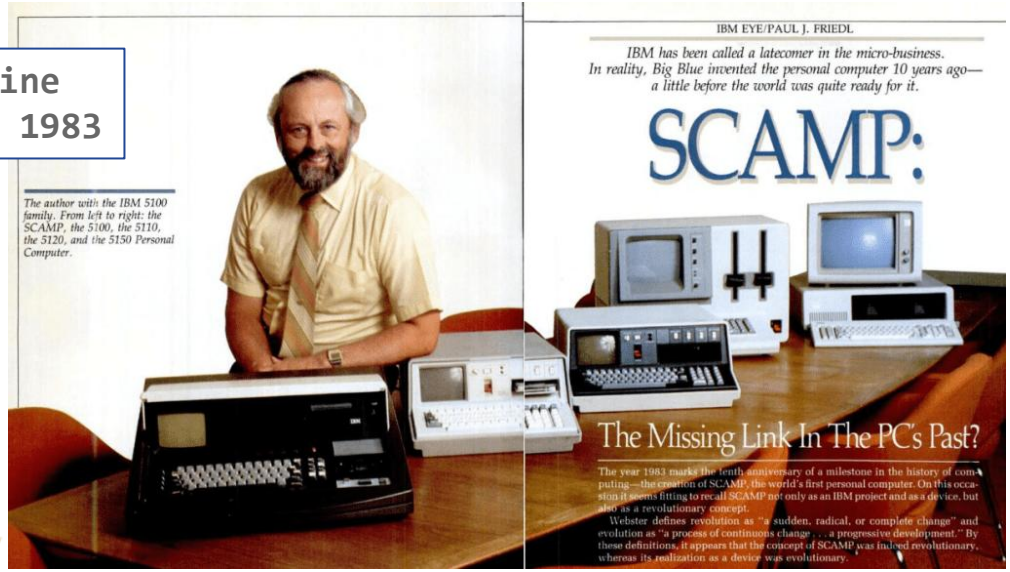
PART 3: Evidence of the S/360 and S/3 mainframe relationship

- How the 5100/5110 Executable ROS was extracted
- How the PALM machine code was reverse engineered
- Arrangement of Non-Executable segments
- Exploring EBCDIC, Jump Table Vectors, PALM

5min

PART 4: IBM 5100 News!

PC Magazine
November 1983



IBM EYE/PAUL J. FRIEDL

IBM has been called a latecomer in the micro-business. In reality, Big Blue invented the personal computer 10 years ago—a little before the world was quite ready for it.

SCAMP:

The Missing Link In The PC's Past?

The author with the IBM 5100 family. From left to right: the SCAMP, the 5100, the 5110, the 5120, and the 5150 Personal Computer.

The year 1983 marks the tenth anniversary of a milestone in the history of computing—the creation of SCAMP, the world's first personal computer. On this occasion it seems fitting to recall SCAMP not only as an IBM project and as a device, but also as a revolutionary concept.

Webster defines revolution as "a sudden, radical, or complete change" and evolution as "a process of continuous change . . . a progressive development." By these definitions, it appears that the concept of SCAMP was indeed revolutionary, whereas its realization as a device was evolutionary.

The necessarily short development time had a similar effect upon our software plans; we knew we could not afford to develop a new APL language processor from scratch. The solution was, therefore, to emulate a processor for which a suitable APL system already existed. With this as a strategy, we chose to write an IBM 1130 emulator in PALM microcode so we could "plug in" almost all of version of APL that was available for the 1130. In this way, we were able to replace a programming effort of several person-years with one of several person-months.

SCAMP (1973)

IBM 5100 (Sept. 1975)

IBM 5110 (Jan. 1978)

IBM 5120 (Feb. 1980)

IBM 5322 (July 1981)
Intel 8085
Model F Keyboard
ISA-bus

IBM PC 5150 (August 1981)
Intel 8088
Model F Keyboard
ISA-bus

(Special Computer, APL Machine Portable)

Datamaster/23

P.A.L.M.

8085

8088
("x86")

IBM 1130

The IBM 5100 introduced the **"big red power switch"** also used in the IBM PC.

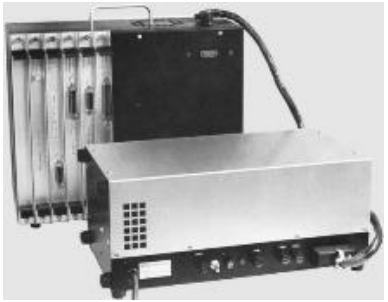
S.C.A.M.P. (Special Computer, APL Machine Portable)

YT: SCAMP 2013 by Friedl

https://www.wang2200.org/systems.html

In 2013, **Alvin Ginsburg** relayed stories about the development of SCAMP.

SCAMP is currently on display on the first floor of the National Museum of American History.



c. 1973

Stuttgart Computer Museum, Germany

P.A.L.M. Put All Logic in Microcode

INSTRUCTIONS FOR
THE USE OF

- ASSEMBLER GENERATOR
- GENERAL ASSEMBLER
- GENERAL LOADER

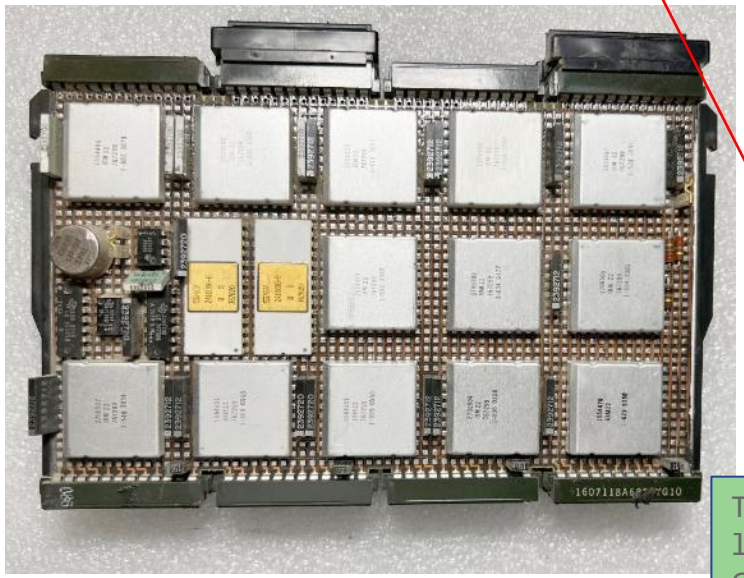
H. J. MYERS
8-624-3155

2/1/77

Internal IBM document.
“GENASM” (General Assembler)

GENASM PALM

IT TAKES ABOUT 45 MINUTES ON A 5100 TO COMPLETELY PROCESS ALL OF "PALM" TO PRODUCE THE PALM ASSEMBLER WHICH IS IN "6 PALM".



TAPE #3 "ASSEMBLER GENERATOR"

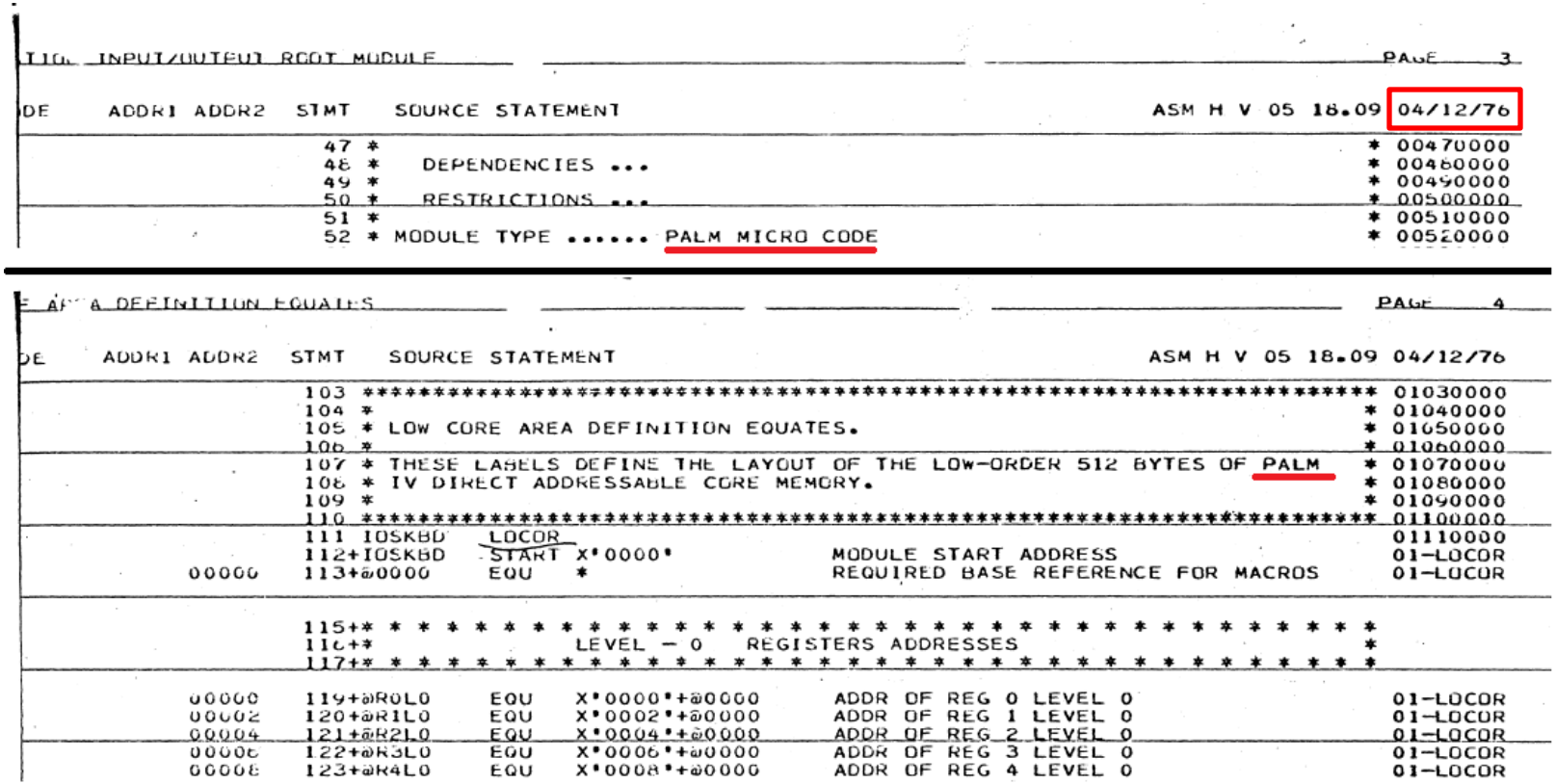
)LIB

001	TEMP	06	030,27
002	ASMO	06	021,001
003	GENASM	06	029,001
004	LOADO	06	014,001
005	SAMPLE	07	009,001
006	PALM	06	036,001

Not the 1990s PALM Pilot!

The two “microprocessor” looking chips are each 64-byte Register Files.

PALM Reference – 1976 (equates.pdf)



```
...TIO  INPUT/OUTPUT ROOT MODULE                                                PAGE     3

...DE    ADDR1 ADDR2  STMT    SOURCE STATEMENT                  ASM H V 05 18.09 04/12/76
                       47 *                                                          * 00470000
                       48 *    DEPENDENCIES ...                                      * 00480000
                       49 *                                                          * 00490000
                       50 *    RESTRICTIONS ...                                      * 00500000
                       51 *                                                          * 00510000
                       52 * MODULE TYPE ..... PALM MICRO CODE                        * 00520000
```

```
...E AP...A DEFINITION EQUATES                                                  PAGE     4

...DE    ADDR1 ADDR2  STMT    SOURCE STATEMENT                  ASM H V 05 18.09 04/12/76
                      103 ********************************************************************** 01030000
                      104 *                                                                      * 01040000
                      105 * LOW CORE AREA DEFINITION EQUATES.                                    * 01050000
                      106 *                                                                      * 01060000
                      107 * THESE LABELS DEFINE THE LAYOUT OF THE LOW-ORDER 512 BYTES OF PALM   * 01070000
                      108 * IV DIRECT ADDRESSABLE CORE MEMORY.                                  * 01080000
                      109 *                                                                      * 01090000
                      110 ********************************************************************** 01100000
                      111 IOSKBD   LOCOR                                                        01110000
                      112+IOSKBD   START X'0000'          MODULE START ADDRESS                  01-LOCOR
              00000   113+@0000    EQU   *                 REQUIRED BASE REFERENCE FOR MACROS    01-LOCOR

                      115+* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                      116+*                LEVEL - 0   REGISTERS ADDRESSES                      *
                      117+* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

              00000   119+@R0L0    EQU   X'0000'+@0000     ADDR OF REG 0 LEVEL 0                 01-LOCOR
              00002   120+@R1L0    EQU   X'0002'+@0000     ADDR OF REG 1 LEVEL 0                 01-LOCOR
              00004   121+@R2L0    EQU   X'0004'+@0000     ADDR OF REG 2 LEVEL 0                 01-LOCOR
              00006   122+@R3L0    EQU   X'0006'+@0000     ADDR OF REG 3 LEVEL 0                 01-LOCOR
              00008   123+@R4L0    EQU   X'0008'+@0000     ADDR OF REG 4 LEVEL 0                 01-LOCOR
```

IBM Assembler Generator

```
      GENASM PALM
DO YOU WANT ERROR CHECKING? Y
MACHINE NUMBER BASE: 2
DIGITS/WORD: 16
ADDRESS UNITS/WORD: 2
DIGITS IN MAXIMUM ADDRESS FIELD: 16
DISPLACEMENT FIELD (SIGN AND DIGITS, IF ANY): 8
```

```
    ∇ IOR OP
[1]   →PASS2/A1
[2]   LCX[''⍴LV↑⎕LC;]←2
[3]   →0
[4]  A1:→((3≥⍴ND)∧2≤⍴ND←,ND)/A2
[5]   ND←LER ERR( 3 2 ⌈2>⍴ND]↑ND),1
[6]  A2:ND←3↑ND
[7]   EMIT POK PACK OP,(1⌈ND[0]),ND[1],MMEX ND[2]
    ∇

    ∇ JMP OP
[1]   →PASS2/A1
[2]   LCX[''⍴LV↑⎕LC;]←2
[3]   →0
[4]  A1:→(2=⍴ND←,ND)/A2
[5]   ND←LER ERR(2↑ND),1
[6]  A2:EMIT POK PACK 12,ND[ 0 1 ],OP
    ∇
```

```
    ∇ ALU OP
[1]   →PASS2/A1
[2]   LCX[''⍴LV↑⎕LC;]←2
[3]   →0
[4]  A1:→(2=⍴ND←,ND)/A2
[5]   ND←LER ERR(2↑ND),1
[6]  A2:EMIT POK PACK 0,ND[ 0 1 ],OP
    ∇

    ∇ BRCN OP;LC
[1]   LC←''⍴LV↑⎕LC
[2]   →PASS2/A2
[3]   →(0≠⍴⍴PASS2)/A1
[4]   LCX[LC;]← 4 8
[5]   →~SL[LC]←1
[6]  A1:255 CKREACH 2
[7]   →0
[8]  A2:→(3=⍴ND←,ND)/A3
[9]   ND←LER ERR(3↑ND),1
[10] A3:→(4≠LCX[LC])/A4
[11]  EMIT(POK PACK 12,ND[ 0 1 ],OP[1]),P2K PACK( 10 15 FWB ND[2]
      ),0,|DISP ND[2]-1
[12]  →0
[13] A4:EMIT(POK PACK 12,ND[ 0 1 ],OP[0]),40964,53256,3 RELOC ND[
      2]
```

IBM Assembler Generator

```
      GENASM PALM
DO YOU WANT ERROR CHECKING? Y
MACHINE NUMBER BASE: 2
DIGITS/WORD: 16
ADDRESS UNITS/WORD: 2
DIGITS IN MAXIMUM ADDRESS FIELD: 16
DISPLACEMENT FIELD (SIGN AND DIGITS, IF ANY): 8
```

```
    ∇ IOR OP
[1]   →PASS2/A1
[2]   LCX[''⍴LV⍳⌈/LC;]←2
[3]   →0
[4]  A1:→((3≥⍴ND)∧2≤⍴ND←,ND)/A2
[5]   ND←LER ERR( 3 2 ⌈2>⍴ND]⍳ND),1
[6]  A2:ND←3⍳ND
[7]   EMIT POK PACK OP,(1⌈/ND[0]),ND[1],MMEX ND[2]
    ∇

    ∇ JMP OP
[1]   →PASS2/A1
[2]   LCX[''⍴LV⍳⌈/LC;]←2
[3]   →0
[4]  A1:→(2=⍴ND←,ND)/A2
[5]   ND←LER ERR(2⍳ND),1
[6]  A2:EMIT POK PACK 12,ND[ 0 1 ],OP
    ∇
```

```
    ∇ ALU OP
[1]   →PASS2/A1
[2]   LCX[''⍴LV⍳⌈/LC;]←2
[3]   →0
[4]  A1:→(2=⍴ND←,ND)/A2
[5]   ND←LER ERR(2⍳ND),1
[6]  A2:EMIT POK PACK 0,ND[ 0 1 ],OP
    ∇

    ∇ BRCN OP;LC
[1]   LC←''⍴LV⍳⌈/LC
[2]   →PASS2/A2
[3]   →(0≠⍴⍴PASS2)/A1
[4]   LCX[LC;]← 4 8
[5]   →⍬⍴SL[LC]←1
[6]  A1:255 CKREACH 2
[7]   →0
[8]  A2:→(3=⍴ND←,ND)/A3
[9]   ND←LER ERR(3⍳ND),1
[10] A3:→(4≠LCX[LC])/A4
[11]  EMIT(POK PACK 12,ND[ 0 1
      ),0,|DISP ND[2]-1
[12]  →0
[13] A4:EMIT(POK PACK 12,ND[ 0 1 ],OP[0]),40964,53256,3 RELOC ND[
      2]
```

12 APL functions defined and used to provide ~60 PALM assembler mnemonics

```
∇ ADDB ND
  ALU 8
∇

∇ AND ND
  ALU 5
∇

∇ BREQ ND
  BRCN 2 10
∇

∇ JLE ND
  JMP 0
∇

∇ ROTR ND
  SHF 13
∇

∇ PUTB ND
  IOR 4
∇
```

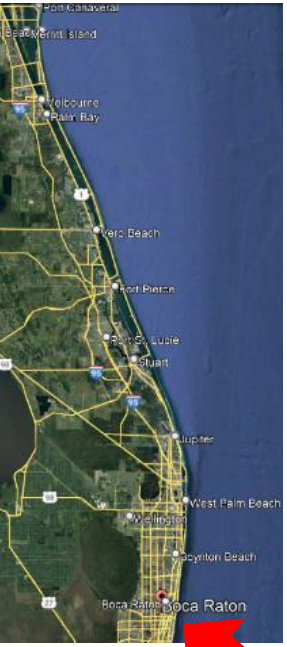
PALM

Some Q&A w/ Industry Pioneers

- **Dennis Roberson**
  - Lead engineer for the IBM 5100

- **Hal Prewitt**
  - Founder of CORE International

Dennis Roberson Q&A 1/3



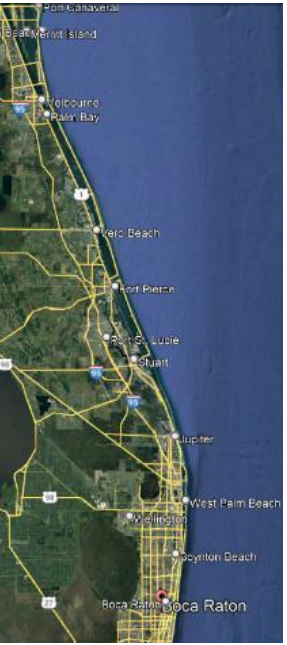
Q: What do you recall on the development of the PALM processor?

A: PALM was developed by IBM's General Systems Division (GSD) Laboratory in Boca Raton, Florida (S/7, S/1, IBM PC)

- "Development was under the leadership of **Roger Abernathy** with **Bob Tuttle** and **Virgil Wyatt**"
- "it was the processor for the early prototype system **that is now in the Smithsonian.**"
- "was intended for use in **industrial application**"

Dennis Roberson (M.S. E.E. Stanford)
Also highly involved in development of the SCAMP prototype.
"A Microprocessor-based portable computer: The IBM 5100"
(IEEE Volume 64 issue 6, June 1976)

Dennis Roberson Q&A 1/3



Q: What do you recall on the development of the PALM processor?

A: …continued…

Paul Friedl asked **Roy Harper** (Los Gatos, CA) via **Joe George** on the "best" processor for a "portable" system, they recommended PALM.

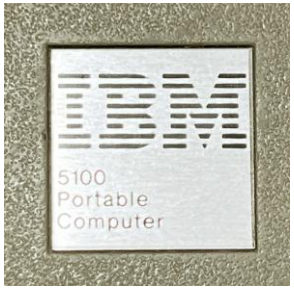
- "**Joe George** was the primary developer for the SCAMP electronics"
- "**Roy Harper** developed the power supply"
  - w/ George Marenin, Ed Finnegan, Warren Christopherson
- "**Patrick Smith** provided the software"

Dennis Roberson Q&A 2/3

- **Q: It has been suggested that PALM had some relationship to the Apollo program. Aside from scientists and universities, who else used the IBM 5100?**

- A: “While the NASA Apollo connection is possible, I don’t think the idea was ever consummated.” (the Boca Raton division was intended for industrial computers)
- A: “The 5100 was indeed used by the **DoD** and in **banking** as well... They also had and have many needs aligned with problem solving and often in environments where ‘portability’ or more correctly size, weight and power consumption were important characteristics. **It was used by other government agencies as well.**”

IRS… CIA…



Dennis Roberson Q&A 3/3

- **Q: “Was the Norelco Carry-Corder in SCAMP functional?”**
  - Known as **Philips** EL3300 in Europe (c.1963), available under **Norelco** brand in the US a year later (World’s First portable cassette recorder)
- A: “The Norelco carry-coder was very much an operational part of SCAMP and did store the software that was regularly update. It was however **very unreliable and every time we did a demo we usually loaded the software the night before and kept the system running over night for fear that the tape would not work at the wrong time.”**

Dennis Roberson Q&A 3/3

- **Q: "Was the Norelco Carry-Corder in SCAMP functional?"** (cont.)
  - Known as **Philips** EL3300 in Europe (c.1963), available under **Norelco** brand in the US a year later (World's First portable cassette recorder)
- "The **3M Tape cassette** (DC300) was everything the Norelco was not, i.e., it was very reliable, was much faster and had much higher capacity, yet it was still small enough to fit in a transportable box."
- **"There was a strong push in Rochester, MN where the 5100 was developed to switch to a floppy disk as the storage media.** Unfortunately, floppy disks had an 8" format at the time and would have eliminated the notion of portability we were trying to get to with the 5100."

Hal Prewitt Q&A (CORE)

- **Q: Do you know of any CoreNET hardware that is still available?**

- A: "…there are copies of PC-51 but **no CoreNet LAN or hardware remains available.** The hardware was for 5110, 5120 and the original IBM PCs."

NOTE: CORE was sold to SONY (Aiwa) in c. 1993.

The amount I came across was that it was sold for ~$5 million.

Hal Prewitt Q&A (CORE)

- A: “CoreNET was our storage & network for the 5110/5120 systems. We wrote **PALM assembler,** linker and IO drivers. We engineered an interface card that connected to the 5110/20 cabling design. Build cables and assorted storage systems. **Did not use Async, Comm or parallel card features.”**

- A: “The HDD interface inside our drive boxes used SASI (predecessor to SCSI). On the 5110/20 we loaded a driver from the floppy which provided access to the CoreNet and our HD drives. The 5110/20 could directly access its storage devices (tape & floppies) and our HD systems just by using new references for our storage devices.”
  - e.g. D08 instead of **D80**

542 S.E. 5th Avenue • Delray Beach, Florida 33444 • 305/276-3929

FOR IMMEDIATE RELEASE
Attn: News
Features
Products

CORE INTERNATIONAL

RELEASE: NO. 21985/1723
Contact: Bennett Greene
V.P. Marketing

PC-51 SOFTWARE REPLICATES IBM 5110/5120 OPERATING ENVIRONMENT; BASIC PROGRAMS RUN WITHOUT MODIFICATION

In the face of incredible advances in the Personal/Business field, many users like the power of the advanced Basic Language on their 5110/20s -- it can run business oriented programs that heretofore simply were too much for even the "best" of the PC Basics. That's why there are over 30,000 of them still in use today. And CORE has grown to be the largest independent service and support organization for IBM 5110/20 systems, as well as the sole source for IBM 5110/20 expansion hardware and software.

(c. 1982)

Since no new IBM 5110s or 5120s were in production and few pre-owned were available, CORE immediately began working on software that would allow users to move their custom and commercial 5110/20 Basic programs and data from the 5110s and 5120s to reliable yet relatively inexpensive hardware. After nearly two years of development, CORE's PC-51(TM) operating system and advanced business Basic was recently introduced.

PC-51

GET ABSOLUTE PROOF FOR $50.

THE CORE LIFETIME GUARANTEE.

Our PC-51 Demo Diskettes deliver proof that every statement of every ad we've made for PC-51 is factual. Beyond that comforting fact, our Software Guarantee says that PC-51 will perform exactly as represented by CORE or we'll fix it, replace it, or refund your hard earned money. And that's not just for 30 days or 90 days or a year. That's forever.

THE PLIGHT OF THE PIONEERS.

When you put over three years of R&D into an advanced software project, and create a product that revolutionizes the IBM 5110/20 market, you're going to ruffle more than a few feathers. Software pioneers leave themselves open for a lot of scoffing, humbug, and good old-fashioned skepticism. And some folks just outright won't believe you. But if you can SHOW the doubters that the software does EXACTLY what the ads say it does, then you can win them over based on objective observation, rational thought, and superior support.

YOU CAN TRUST US. BUT WE'D RATHER YOU TRUST YOURSELF.

You have three options. First, you can ignore this ad and all terrific things you have heard about PC-51 and do nothing, or worse, buy a System/36 and suffer the slings and arrows of overspending, rewriting all your custom software, and retraining your operators.

Second, you can believe every word of our ads for PC-51 because: a) they're the honest-to-goodness truth and b) because we back this up with our Life time Software Guarantee.

Finally, if you're from Missouri (where the most skeptical skeptics are from), you can send us a paltry $50 for the enlightening opportunity to see PC-51 in action.

WHAT YOU GET FOR 50 BUCKS:

You get two diskettes with identical contents. One 5¼" with a DEMO copy of PC-51 for the IBM PC and one 8" diskette for the IBM 5110/20. Both diskettes contain the same BASIC programs so you can verify that the PC does in fact run identical programs.

Included on the diskettes are an Amortization (loan) program, a sample General Ledger Account Enter/Edit program and a Chart of Accounts Listing program. We have also included a special program which will demonstrate calculations, display screen functions, key file access, and improved printer performance.

First, run the programs on the IBM 5110/20 and measure the actual time required for each function. Next, find an IBM PC with a 360KB diskette drive, PC-DOS 2.0 and at least 256KB of main memory. Run the same programs and measure the performance improvement. We expect you'll become a believer once you see IDENTICAL 5110/20 software running on a PC.

SUPPOSE I LIKE IT...

IF the demo does what it is supposed to, that is, show you the true light about PC-51, THEN you'll soon be sending us an order with $2995 for your registered copy of PC-51. And, if you return all demo materials to us, we'll give you a $25 credit toward future purchases.

CORE INTERNATIONAL

542 S.E. 5th Avenue • Delray Beach, Florida 33444 • 305/276-3929

Interesting BASIC Features: FORM

The following is a sample program that writes data to and reads data from the display screen:

"screen"

```
0010 OPEN FILE FL1,'002',ALL
0020 WRITEFILE USING 30,FL1,'NAME','ADDR','CITY'
0030 FORM POS84,C,POS148,C,POS212,C,POS91
0040 READFILE USING 50,FL1,N$
0050 FORM POS91,C18
0060 REWRITEFILE USING 70,FL1,N$
0070 FORM POS91,C,POS155
0080 READFILE USING 90,FL1,A$
0090 FORM POS155,C18
0100 REWRITEFILE USING 110,FL1,A$
0110 FORM POS155,C,POS219
0120 READFILE USING 130,FL1,C$
0130 FORM POS219,C18
0140 REWRITEFILE USING 150,FL1,N$,A$,C$
0150 FORM POS577,'YOU ENTERED',POS641,C,X2,C,X2,C
```

The following are examples of insertion characters. Assume a data item value of 112233 is to be printed:

PIC Specification	Printed Output
PIC(###B##B####)	000 11 2233
PIC(ZZZBZZBZ###)	11 2233
PIC(ZZZ,ZZZ.###)	112,233
PIC(ZZZZZ/Z#/##)	11/22/33
PIC(******#.##)	*112233.00
PIC($$$$$$###+)	$112233+
PIC($$$,$$$,$$$.##)	$112,233.00

```
NAME   12345678901234567A90
ADDR   12345678901234567B90
CITY   12345678901234567C90

YOU ENTERED
12345678901234567A  12345678901234567B  12345678901234567C

RUN                                  60464 001
```

Concept of re-directing from screen to file to printer ("unix"-ish style)

Interesting BASIC Features: FLS

- Access special system state using the “FLS” internal file

WRITE FILE FLS Statement

You can set or use certain system functions by using the WRITE FILE FLS statement to place the appropriate code in file FLS, as follows:

System Function	*File FLS Position*	*Code*
Turn the display screen off	1	F
Turn the display screen on	1	N
Turn the audible alarm on	1	S
Turn the audible alarm off	1	Q
Pulse the audible alarm	1	A
Set keyboard input to lowercase character mode	2	L
Set keyboard input to standard BASIC character mode	2	U
Turn the display trace on	3	N
Turn the display trace off	3	F
Turn the printer trace on	4	N
Turn the printer trace off	4	F
Set rounding precision	5-6	0 to 15
File FLx	7-9	FL0-9
Set number of print lines per inch (2.54 centimeters) For example, the statements	10-11	8-99

```
0280 WRITEFILE USING 290,FLS,'L'
0290 FORM POS2,C
```

place the character L in position 2 of file FLS. This causes the be in lowercase character mode. That is, lowercase alphabetic characters are entered from the keyboard unless the shift key The 5110 remains in lowercase character mode until the 5110 changed back to standard BASIC character mode or the work cleared.

FILE FLS

File FLS is a 35-byte, system-oriented file that allows you to indicate (with the WRITE FILE FLS statement) such information as console control, national character set selection, and rounding of numeric data. Also, you can use the READ FILE FLS statement to access such data as total user work area in the system, total work area available for variables and buffers, and total number of lines printed.

Early “Windows Registry” file concept?

Interesting BASIC Features: FNEND

- Multi-line function definitions

```
0020 DEF FNF(N)
0030 IF N≠0&N≠1 GOTO 50
0040 RETURN 1
0050 K=1
0060 FOR J=2 TO N
0070 K=K*J
0080 NEXT J
0090 RETURN K
0100 FNEND
```

```
0120 FOR I=0 TO 10
0130 PRINT I,FNF(I)
0140 NEXT I
```

```
RUN
 0             1
 1             1
 2             2
 3             6
 4             24
 5             120
 6             720
 7             5040
 8             40320
 9             362880
 10            3628800
```

Float Point
Exponentials (**)
Trig. functions
MAT functions (no DOT)

CHAIN

AUTO/RENUM

“Function Keys” macros (LOAD0, KEY6)

MERGE

RUN ,P=D

constants (&PI)

Other Lacking BASIC Features

- Strings use single quote (‘) instead of double (“)

- BASIC **MAT** (matrix) has many features but is lacking DOT-PRODUCT
  - (lack of DOT-PRODUCT can make conversion from APL to BASIC difficult)

- **ELSE** of an **IF** is not supported (S/370 BASIC had all these features)

- No single key exclamation (!), must use an “overstrike” sequence
  - (“overstrike” is using a multi-character input to enter in a normally non-printable single character; same approach used by Asian languages on English keyboards)

Lacking BASIC Features

x

←

- No **PEEK/POKE (or SYS/CALL)**
  - This prevents making an INKEY like function to poll for keystrokes

- No **CLS** (clear screen)
  - The IBM 5100-series is emulating systems that used line printers that didn’t have a screen (CRT) to clear.
  - If you were sitting at a line printer terminal, the idea of “clearing the screen” wouldn’t mean anything

BYTE Magazine “letters to editor”

c.July 1977 in response to a prior article

CORRECT CREDIT FOR PEEK, POKE & CALL

Dear Jim Warren, Received: 77-Jul-21

Jef Raskin's review of MITS, Poly, & NIBL BASIC needs some clarification. The PEEK & POKE functions were not invented by Micro-Soft. DEC (Digital Equip. Corp.) has PEEK & POKE in its BASIC PLUS as well as CALL when implemented with their RT-11 system & has had it since before MITS ever dreamed up their Altair 8800. PEEK & POKE were first put in a hobbyist BASIC by Micro-Soft, but the young people reading your excellent mag should be aware that the world of software did not begin with the advent of MITS Altair. Jef should be aware of *all* the facts before he uses such words as “invented” or “useful innovations”.

Ray Atnip 7331 Harwin, Suite 117
Owner, The Bit Barn Houston, TX 77036

P.S. You will probably get a lot of responses in relation to Jef's article.

Time to get technical…

Image: Reuters

Deep Dive!

MICROPROGRAMMING

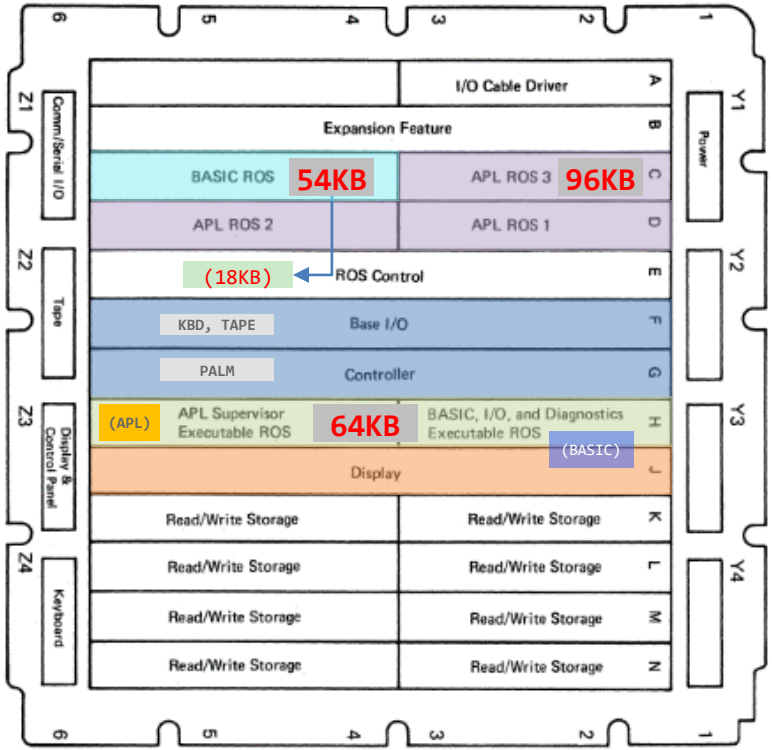
An internal machine program controls the 5100. This internal machine program consists of several microprograms, which the 5100 controller uses to accomplish given tasks. Because the 5100 controller cannot process an APL or BASIC user program directly, it must emulate the APL or BASIC program to the internal machine microcode by means of the APL (H4) or BASIC (H2) emulators on the executable ROS card.

When an APL or BASIC language statement is decoded by the 5100 computer, the machine microcode performs a series of microinstructions that accomplish the required emulation of the user language.

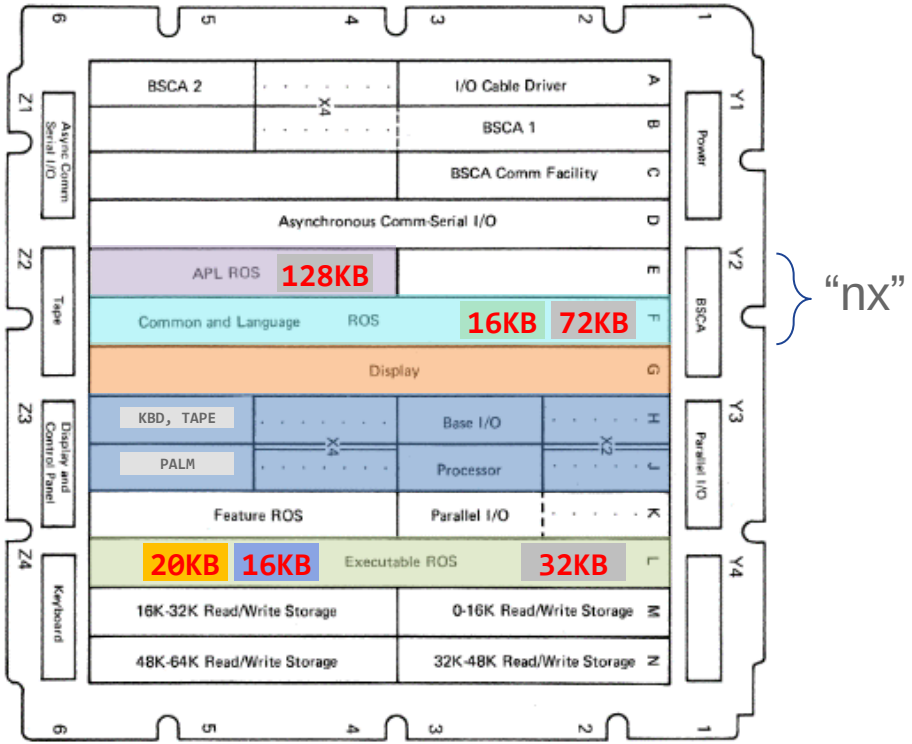
(from the IBM 5100 Maintenance manual)

A1 Board Comparison

IBM 5100



IBM 5110



Comparison of Instruction Sets

IBM System/360

List of Instructions by Set and Feature

Standard Instruction Set

NAME	MNEMONIC	TYPE		CODE
Add	AR	RR	C	1A
Add	A	RX	C	5A
Add Halfword	AH	RX	C	4A
Add Logical	ALR	RR	C	1E
Add Logical	AL	RX	C	5E
AND	NR	RR	C	14
AND	N	RX	C	54
AND	NI	SI	C	94
AND	NC	SS	C	D4
Branch and Link	BALR	RR		05
Branch and Link	BAL	RX		45
Branch on Condition	BCR	RR		07
Branch on Condition	BC	RX		47
Branch on Count	BCTR	RR		06
Branch on Count	BCT	RX		46
Branch on Index High	BXH	RS		86
Branch on Index Low or Equal	BXLE	RS		87
Compare	CR	RR	C	19
Compare	C	RX	C	59
Compare Halfword	CH	RX	C	49
Compare Logical	CLR	RR	C	15
Compare Logical	CL	RX	C	55
Compare Logical	CLC	SS	C	D5
Compare Logical	CLI	SI	C	95
Convert to Binary	CVB	RX		4F
Convert to Decimal	CVD	RX		4E

IBM 1130

Instruction	Mnemonic	Binary OP Code
Load and Store		
Load ACC	LD	11000
Load Double	LDD	11001
Store ACC	STO	11010
Store Double	STD	11011
Load Index	LDX	01100
Store Index	STX	01101
Load Status*	LDS ⑦	00100
Store Status	STS	00101
Arithmetic		
Add	A	10000
Add Double	AD	10001
Subtract	S	10010
Subtract Double	SD	10011
Multiply	M	10100
Divide	D	10101
And	AND	11100
Or	OR	11101
Exclusive Or	EOR	11110
Shift Left* Modifier Bits 8 & 9:		
Shift Left ACC 00	SLA ⑦	00010
Shift Left ACC and EXT 10	SLT ⑦	00010
Shift Left and Count ACC 01	⑧ SLCA ⑦	00010
Shift Left and Count ACC and EXT 11	⑧ SLC ⑦	00010
Shift Right* Modifier Bits 8 & 9:		
Shift Right ACC 00 or 01	SRA ⑦	00011
Shift Right ACC and EXT 10	SRT ⑦	00011
Rotate Right 11	RTE ⑦	00011

IBM System/3

MNEMONIC OPERATION CODES (MACHINE

*Instruction**	*Mnemonic Operation*
Zero and Add Zoned Decimal	ZAZ
Add Zoned Decimal	AZ
Subtract Zoned Decimal	SZ
Move Hex Character	MVX
Move Characters	MVC
Compare Logical Characters	CLC
Add Logical Characters	ALC
Subtract Logical Characters	SLC
Insert and Test Characters	ITC
Edit	ED
Move Logical Immediate	MVI
Compare Logical Immediate	CLI
Set Bits On Masked	SBN
Set Bits Off Masked	SBF
Test Bits On Masked	TBN
Test Bits Off Masked	TBF
Store Register	ST
Load Register	L
Add to Register	A
Branch On Condition	BC
Test I/O and Branch	TIO
Sense I/O	SNS
Load I/O	LIO
Load Address	LA
Load CPU***	LCP
Store CPU***	SCP
Advance Program Level	APL
Halt Program Level	HPL

IBM 5100 (PALM)

Microinstructions **Appendix C**

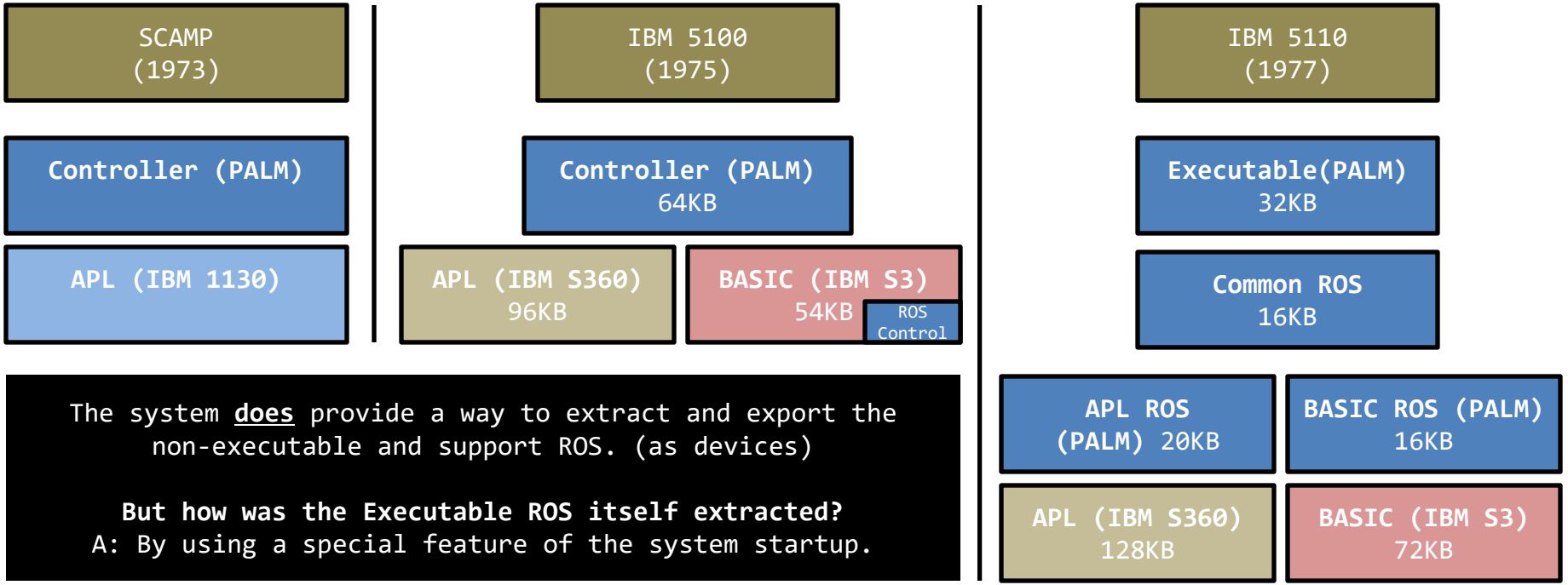
I/O control and the high level languages (APL and BASIC) are implemented with microinstructions in read/write storage and executable ROS. All 5100 Portable Computer microinstructions are a halfword (2 bytes). The first 4 bits of the halfword is the op code. The meaning of the remaining 12 bits depends on the op code (refer to *Formats*). Some op codes have a modifier (bits 12-15) that expands the number of microinstructions beyond 16.

Op cod
classifie
I/O
Fetc
Regi
Arit
Logi
Jum

Op Code	Second Hex Digit	Third Hex Digit	Fourth Hex Digit	Instruction Mnemonic
0	Rx	Ry	AM	ADD, ADDS1, ADDS2, AND, MVM1, MVM2, MVP1, MVP2,
1	DA	Command		CTL
2	Rx	Address		LDHD
3	Rx	Address		STHD
4	DA	Ry	M	PUTB
5	Rx	Ry	M	STHI
6	Rx	Ry	M	LDBI
7	Rx	Ry	M	STBI
8	Rx	Data		EMIT
9	Rx	Mask		CLRI
A	Rx	Data		ADDI
B	Rx	Mask		SETI
C	Rx	Ry	JM	ALL JUMPS
D	Rx	Ry	M	LDHI
E	DA	Ry	SM	ROTR, SHFTR, GETB, GETR
F	Rx	Data		SUBI

Appendix C

Prototype to Production



The system **does** provide a way to extract and export the non-executable and support ROS. (as devices)

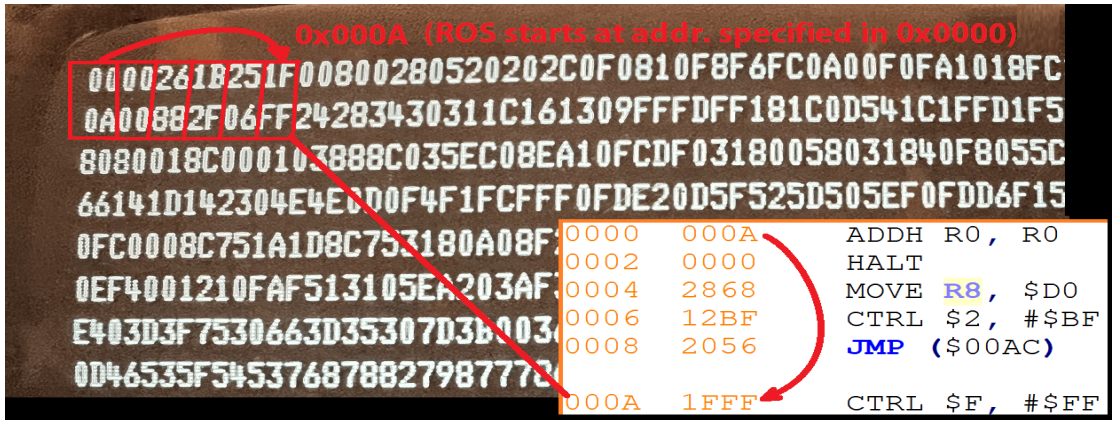
But how was the Executable ROS itself extracted?
A: By using a special feature of the system startup.

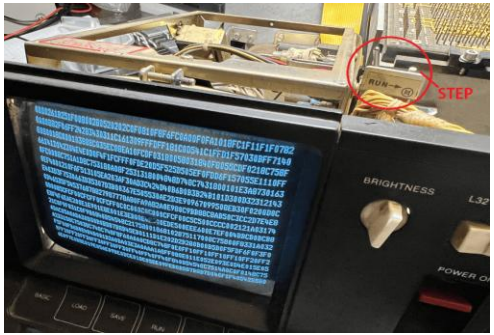
The above was discussed by Joey Tuttle (IBM 5100 developer) at VCF in 2003.

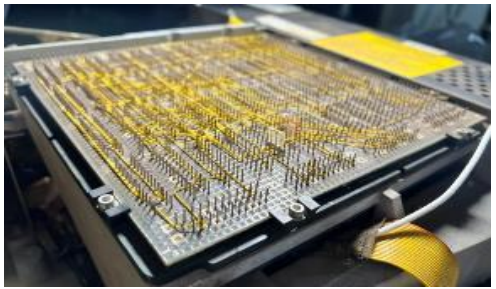
Executable ROS Extraction

- **Christian Corti** determined an approach on the IBM 5110 in c. 2005
- **Tom Stepleton** applied this approach on the IBM 5100 in c. 2019

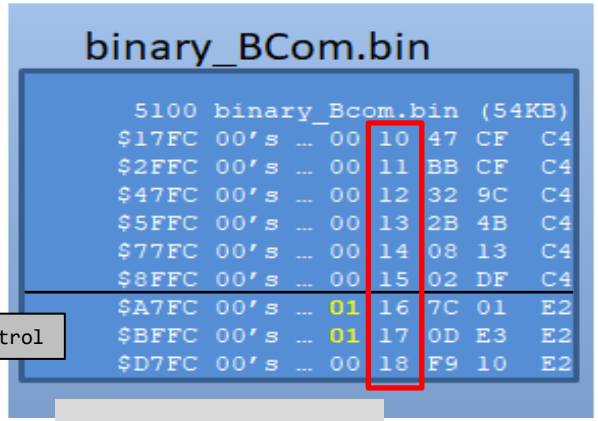
  - Pull the address lines to the desired 512 byte address
    - This is done 128 times (address lines in Logic Diagrams)
    - 128 * 512 = 64KB
  - Place the system into STEP mode by flipping the RUN switch
  - Power on the system and take an image of the startup screen while in “Registers Mode” (with hexadecimal)
  - OCR these images (or type all the values manually!)

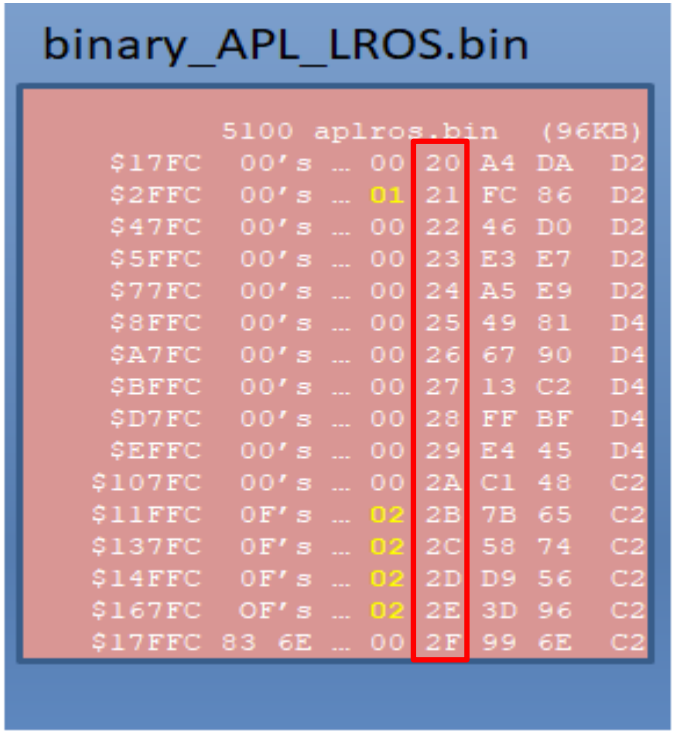






6KB ROS Segments (IBM 5100)



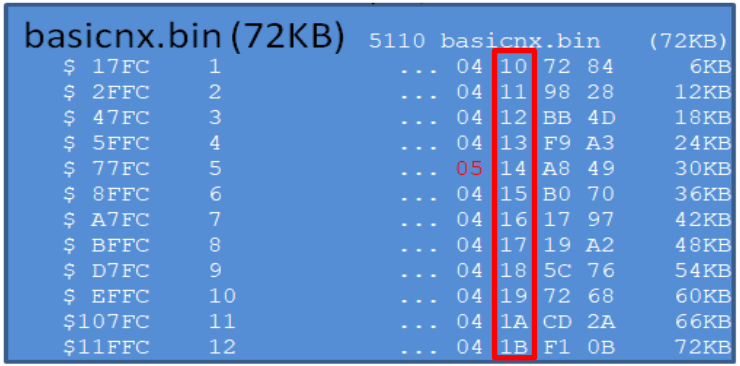


[1]ROS content:

Sequence Number	ROS Card
10, 11, 12, 13, 14, and 15	C4 (BASIC ROS)
16, 17 and 18	E2 (ROS adapter)
20, 21, 22, 23, and 24	D2 (APL ROS 1)
25, 26, 27, 28, and 29	D4 (APL ROS 2)
2A, 2B, 2C, 2D, 2E, and 2F	C2 (APL ROS 3)

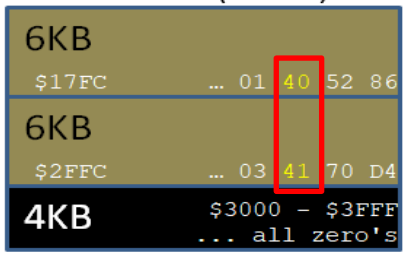
"I" and "O" omitted since they are often confused with 1 and 0.

6KB ROS Segments (IBM 5110)

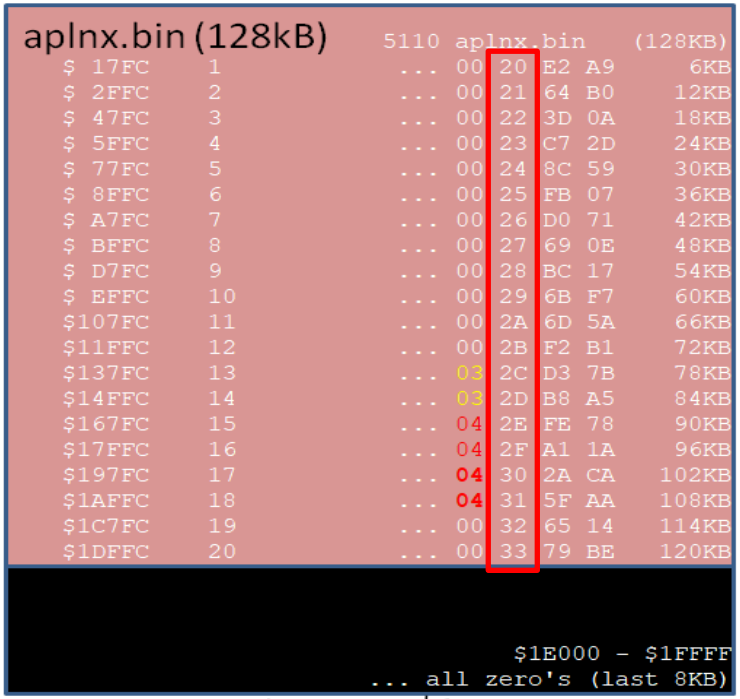


CTRL 1, #$08

comros.bin (16KB)



CTRL 1, #$02



CTRL 1, #$04

The Mysterious “Chapter 2”

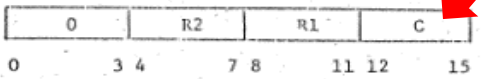
Found by C. Corti ~2006 in an IBM Museum archive
(greater detail than Appendix C)

2.0 MICRO-INSTRUCTION SET

The micro-instructions are designed to support a wide range of applications including Input/Output control, Desk Calculator Functions, Remote Work Stations, Interactive Terminal Operations, or a completely selfcontained General Purpose Computer. The micro-instructions selected for PALM are composed of sixteen bit control words, grouped into six basic op code classifications. In general, they support instruction/ operand fetch and store, arithmetic and logical operations, test under mask and jump, bit manipulation, input/output control, and data transfer. Figure 11 is a summary of the microinstruction set.

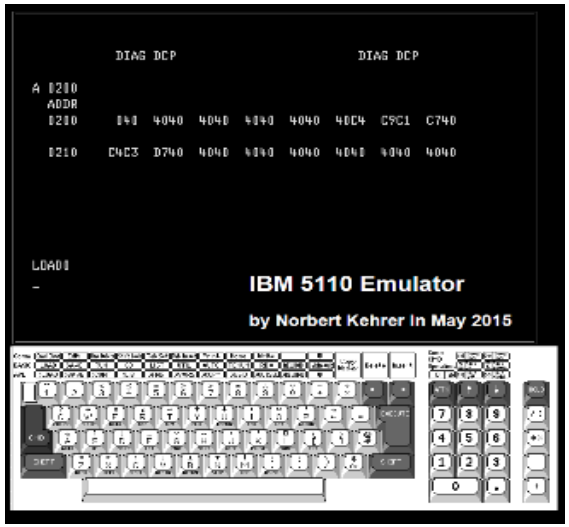
A detailed description of each group follows.

2.2.9 HI TO LO (HTL R2,R1)



The high order byte of the source register (R1) replaces the low order byte of the destination register (R2). The high order bytes of the source and destination registers are not altered as a result of instruction execution.

important!

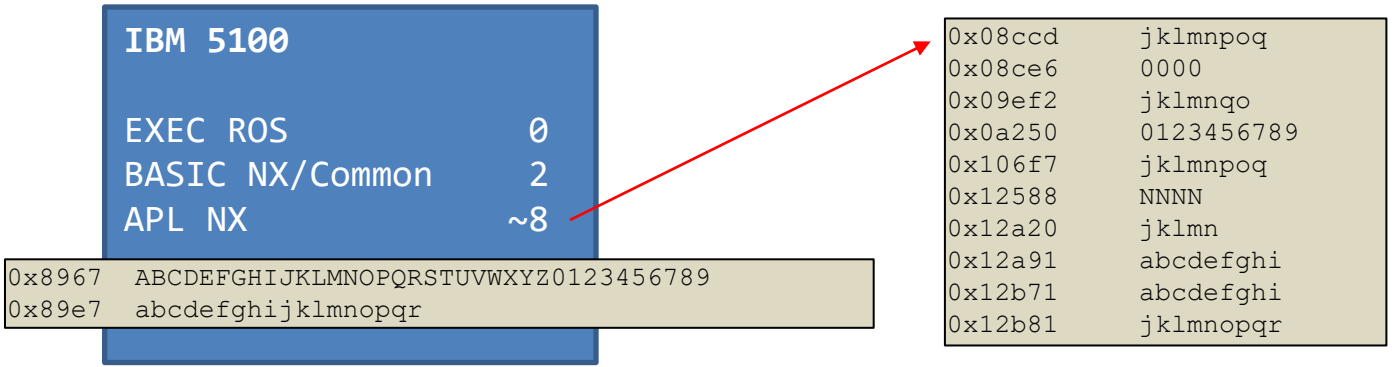


Compare: IBM 5100 MIM Appendix C

HTL Rx, Ry (op code 0): The high order byte of register Ry is moved to the low order byte of register Rx. Register Ry is not changed unless Ry and Rx are designated as the same register.

The IBM System/360 Relationship:
Search for EBCDIC
(Extended Binary Coded Decimal Interchange Code)

- Search the binary ROS and assume the given values are EBCDIC
- Convert the given value to the ASCII equivalent
  - e.g. EBCDIC index 129 == ASCII symbol ‘a’ (97)
  - E.g. EBCDIC index 240 == ASCII symbol ‘0’ (48)
- Do this only for printable characters and queue up such conversion longer than 3 character (4 or more)



The IBM System/360 Relationship:
Search for EBCDIC
(Extended Binary Coded Decimal Interchange Code)

- Search the binary ROS and assume the given values are EBCDIC
- Convert the given value to the ASCII equivalent
  - e.g. EBCDIC index 129 == ASCII symbol ‘a’ (97)
  - E.g. EBCDIC index 240 == ASCII symbol ‘0’ (48)
- Do this only for printable characters and queue up such conversion longer than 3 character (4 or more)

IBM 5100

EXEC ROS	0
BASIC NX/Common	2
APL NX	~8

IBM 5110

EXEC ROS	0
BASIC ROS	0
APL ROS	0
Common ROS	319
BASIC NX	60
APL NX	174

PALM

Sample of EBCDIC Strings found in IBM 5110

Hex given values are the corresponding address offset.

Common ROS

0x00c0 HOLD
0x00e0 **IBM5100**
0x0302 AABBCCDDEEFFGGHHII
0x0c06 DIAG
0x0c10 ADDR
0x0c22 BRANCH
0x0cb4 FILE
0x0d27 ERROR
0x0df4 SORT
0x0e5b COLLATING
0x0fa6 FIELD
0x100f COMPLETED
0x1030 BASIC
0x1150 LOCATION
0x11ce DISK
0x11ee SYSAREA
0x1200 ERRORSET
0x13db AUXILIARY
0x1579 ATTN
0x2e29 ASYNCHRONOUS
0x2e62 BSCA
0x2e7a PARALLEL
0x2e95 **MICROPROCESSOR**
0x2edb CHART
0x2f39 STATUS

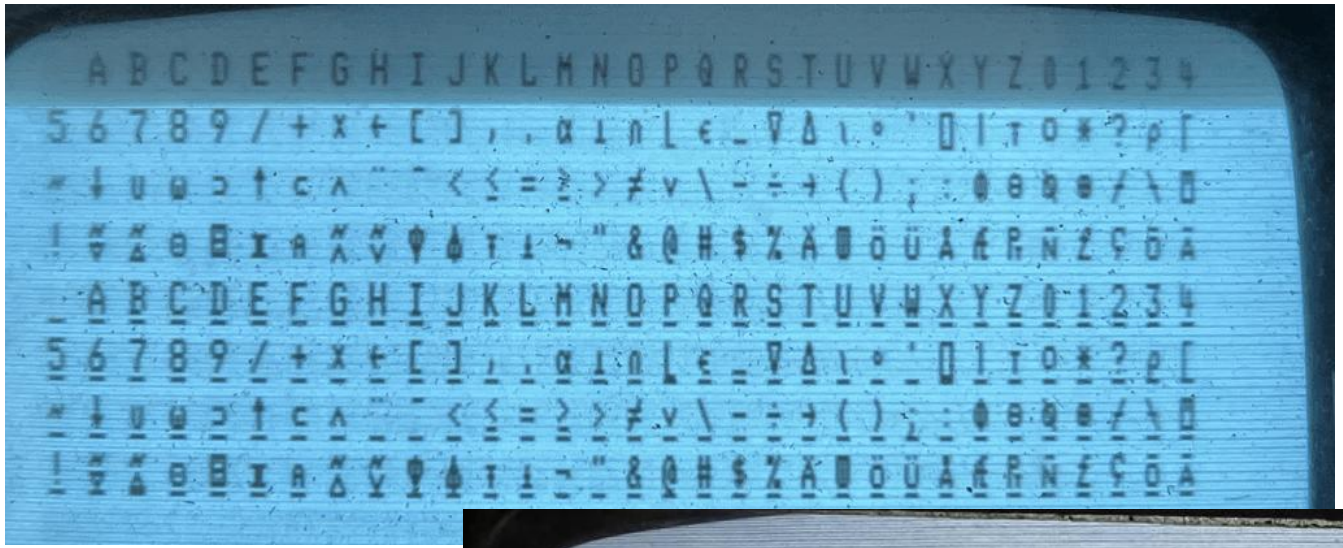
BASIC NX ROS

0x039a RUNS PROC GO
0x03c6 ALERT
0x03cc RENUM
0x03d2 AUTO MERGE
0x03dc UTIL REWIND LINK
0x042e FORM
0x0512 CHAIN
0x0536 AIDX (?)
0x053c DIDX (?)
0x0542 DBCRPOS (?)
0x0557 EOF ATTN READY
0x058b NOREC
0x0591 NOKEY
0x0597 DUPKEY
0x05c5 SQR2
0x05df INCM (?)
0x060b RECL (?)
0x0611 NOBLOCK
0x0620 PROTECT
0x0628 FILEID VOLID
0x0634 DROP FREE SORT OFF
0x072e 00112233445566778899

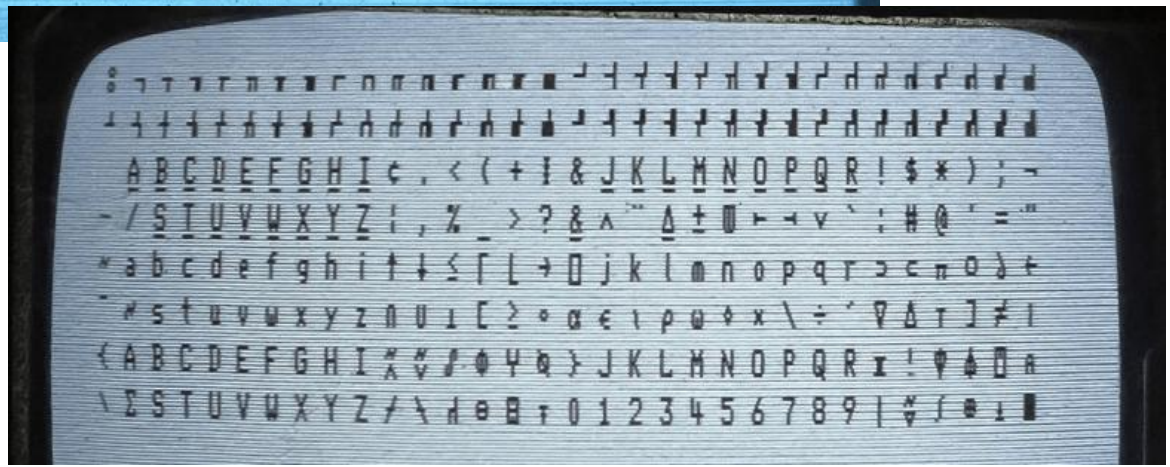
APL NX ROS

0x0e056 ABCDEFGHIJKLMNOPQRSTUVWXYZ
0x0e08c 0123456789
0x0e0ba abcdefghijklmnopqr
0x10f0a **COPYRIGHT**
0x10f18 **1976**
0x12044 ERROREXCEEDED
0x12052 MAXIMUM
0x1205a RECORD
0x13ee0 WS LOADED RESUMED NOT
0x13f1b INCORRECT
0x13f25 COMMAND
0x13f44 LOCKED
0x13f79 SYMBOL
0x13fa1 SUSPENDED
0x13fab FUNCTIONS
0x15af1 CHECKPOINTED CONTINUED
0x15b5f THIS
0x15b6e LOCKED
0x15b7c FOUND
0x15b8d INVALID
0x15b95 DEVICE
0x15bf9 PROTECTED
0x176ea WS FULL
0x17997 VARS
0x17a00 SYSAREA

< 5100



< 5110



IBM System/360 Assembly Example

```
  LOC   OBJECT CODE   STMT   SOURCE STATEMENT

                         1          PRINT NOGEN
000000                   2 FORMAT   START 0
000000 05B0              3 BEGIN    BALR  11,0
000002                   4          USING *,11
000002 41A0 0007         5          LA    10,7              REG 10 I
000006 1B99              6          SR    9,9               CLEAR RE
000008 439A B075         7 LOOP     IC    9,NUMBER-1(10)    INSERT I
00000C 5490 B08E         8          N     9,MASK1           STRIP OF
000010 5990 B0A2         9          C     9,TEN             IS NUMBE
000014 4740 B018        10          BL    OK                BRANCH A
                        11          EOJ                     NOT A DI
00001A 46A0 B006        14 OK      BCT   10,LOOP           REDUCE C
00001E 4380 B07C        15          IC    8,NUMBER+6        IF HERE,
000022 5480 B092        16          N     8,MASK2           STRIP OF
000026 5980 B09A        17          C     8,PLUS            COMPARE
00002A 4780 B02E        18          BE    OK2               BRANCH IF OK
                        19          EOJ                     NOT AN EBCDIC SIGN
000030 1B88             22 OK2     SR    8,8               CLEAR REG 8
000032 1898             23          LR    9,8               CLEAR REG 9 BY LOADING FROM REG 8
000034 18A8             24          LR    10,8              CLEAR REG 10 BY LOADING FROM REG 8
000036 8B90 0001        25 LOOP2   SLA   9,1               SHIFT REG 9 LEFT 1 BIT
00003A 438A B07D        26          IC    8,COMB(10)        INSERT 1 BYTE IN REG 8--INDEXED
00003E 5480 B096        27          N     8,MASK3           STRIP OFF DIGIT PART
000042 5980 B09A        28          C     8,PLUS            COMPARE WITH CODING FOR PLUS
000046 4780 B052        29          BE    YES               BRANCH IF PLUS
00004A 5980 B09E        30          C     8,MINUS           COMPARE WITH CODING FOR MINUS
00004E 4780 B056        31          BE    NO                BRANCH IF MINUS
                        32          EOJ                     NEITHER PLUS NOR MINUS
000054 5A90 B0A6        35 YES     A     9,ONE             IF PLUS ADD 1 TO CONTENTS OF REG 9
000058 5AA0 B0A6        36 NO      A     10,ONE            ADD 1 TO REG 10 FOR LOOP TEST
00005C 59A0 B0AA        37          C     10,TEST           COMPARE
```

From IBM 5110 APL "NX" ROS

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
000001E0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001F0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000200   50 C0 D1 DC 05 C0 50 F0 D1 D8 41 EE 01 A8 41 60  PÀÑÜ.ÀPðÑØAî.¨A`
00000210   00 01 58 10 D0 48 12 11 0F 20 01 13 50 60 D0 4C  ..X.ÐH... ..P`ÐL
00000220   D2 00 D0 23 D0 6F 58 10 D0 68 12 11 0F 20 01 1D  Ò.Ð#ÐoX.Ðh... ..
00000230   50 60 D0 6C D2 00 D0 1F D0 4F 58 10 D0 6C 50 10  P`ÐlÒ.Ð.ÐOX.ÐlP.
00000240   D0 24 59 10 D0 4C 0F 20 01 28 D2 03 D0 24 D0 4C  Ð$Y.ÐL. .(Ò.Ð$ÐL
00000250   41 F0 00 04 58 60 D0 44 58 50 D0 64 50 60 D0 98  Að..X`ÐDXPÐdP`Ð˜
00000260   18 45 19 56 0F B0 01 35 18 46 41 A0 D1 EC 41 8A  .E.V.°.5.FA ÑìAŠ
00000270   40 08 41 78 40 08 41 37 40 08 19 3E 41 10 00 10  @.Ax@.A7@..>A...
00000280   0F B0 03 E4 12 55 0F 80 01 4B 58 30 D0 60 41 33  .°.ä.U.€.KX0Ð`A3
```

```
50 C0 D1 DC   Store Source: Reg. C  0D1DC = Target
              Offset = 0, Base Register = D, Displacement = $1DC
05 C0         BALR (Branch and Link) Registers C,0
50 F0 D1 D8   Store Source: Reg. F  0D1D8 = Target
              Offset = 0, Base Reg. = D, Displacement = $1D8
41 EE 01 A8   Load Address  Target: Reg. E
              Offset = E, Base 0, Displacement = $1A8
41 60 00 01   Load Address, Target: Reg. 6
58 10 D0 48   Load Reg. 1 0:D:048 = Source address
12            LTR Load and Test (Reg)
11            LNR Load Negative (Reg)
```

NAME	MNEMONIC	TYPE	EXCEPTIONS	CODE
Store	ST	RX	P,A,S	50
Branch and Link	BALR	RR		05
Store	ST	RX	P,A,S	50
Load Address	LA	RX		41
Load Address	LA	RX		41
Load	L	RX	A,S	58
Load and Test	LTR	RR	C	12
Load Negative	LNR	RR	C	11

From IBM 5110 APL “NX” ROS

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000200	50	C0	D1	DC	05	C0	50	F0	D1	D8	41	EE	01	A8	41	60	PÀÑÜ.ÀPðÑØAî.¨A`
00000210	00	01	58	10	D0	48	12	11	0F	20	01	13	50	60	D0	4C	..X.ÐH... ..P`ÐL
00000220	D2	00	D0	23	D0	6F	58	10	D0	68	12	11	0F	20	01	1D	Ò.Ð#ÐoX.Ðh... ..
00000230	50	60	D0	6C	D2	00	D0	1F	D0	4F	58	10	D0	6C	50	10	P`ÐlÒ.Ð.ÐOX.ÐlP.
00000240	D0	24	59	10	D0	4C	0F	20	01	28	D2	03	D0	24	D0	4C	Ð$Y.ÐL. .(Ò.Ð$ÐL
00000250	41	F0	00	04	58	60	D0	44	58	50	D0	64	50	60	D0	98	Að..X`ÐDXPÐdP`Ð˜
00000260	18	45	19	56	0F	B0	01	35	18	46	41	A0	D1	EC	41	8A	.E.V.°.5.FA ÑìAŠ
00000270	40	08	41	78	40	08	41	37	40	08	19	3E	41	10	00	10	@.Ax@.A7@..>A...
00000280	0F	B0	03	E4	12	55	0F	80	01	4B	58	30	D0	60	41	33	.°.ä.U.€.KX0Ð`A3

Similar code found in 5100 APL “NX” ROS @ addr. 0

5100_binary_APL_LROS.bin

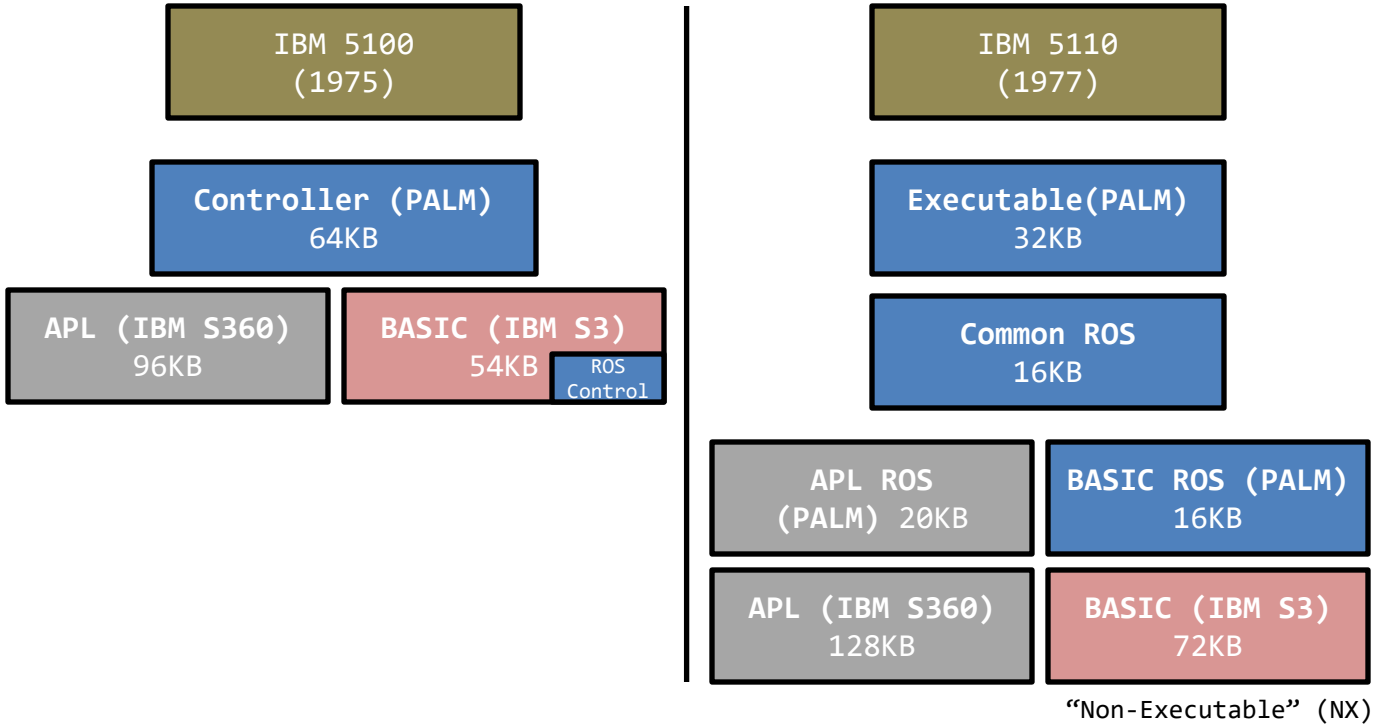
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	50	C0	D1	DC	05	C0	50	F0	D1	D8	41	EE	01	A8	41	60
00000010	00	01	58	10	D0	48	12	11	47	20	C0	20	50	60	D0	4C
00000020	D2	00	D0	23	D0	6F	58	10	D0	68	12	11	47	20	C0	34
00000030	50	60	D0	6C	D2	00	D0	1F	D0	4F	58	10	D0	6C	50	10
00000040	D0	24	59	10	D0	4C	47	20	C0	4A	D2	03	D0	24	D0	4C
00000050	41	F0	00	04	58	60	D0	44	58	50	D0	64	50	60	D0	98
00000060	18	45	19	56	47	B0	C0	64	18	46	41	A0	D1	EC	41	8A
00000070	40	08	41	78	40	08	41	37	40	08	19	3E	41	10	00	10
00000080	47	B0	C5	7E	12	55	47	80	C0	90	58	30	D0	60	41	33

IBM 5110 System/360 OpCode Vector Table

Found in APL "NX" ROS address 0x1C800 (these addresses refer back to the APL ROS)
(for the 5100, this table is at 0x17E00 in its APL NX; the addresses refer back to the APL Supervisor portion of its Executable ROS) [curious: why does OpCode 0xFF have an address?]

ADDR. 0x1C800	INSTRUCTION	OP	addr.	INSTRUCTION	OP	addr.	INSTRUCTION	OP	addr.	INSTR.	OP	addr.
0x03D4		00	0x0A3E	LPDR	20	0x0CD0	STH	40	0x0CE0	STD	60	0x03
0x03D4		01	0x0A54	LNDR	21	0x0B78	LA	41	0x03D4		61	0x03
0x03D4		02	0x0A80	LTDR	22	0x0D1E	STC	42	0x03D4		62	0x04
0x03D4		03	0x0A6A	LCDR	23	0x0B86	IC	43	0x03D4		63	0x03
0x047E	SPM	04	0x1F08	HDR	24	0x058E	EX	44	0x03D4		64	0x03
0x0DB6	BALR	05	0x03D4		25	0x0DAC	BAL	45	0x03D4		65	0x03
0x0D5C	BCTR	06	0x03D4		26	0x0D56	BCT	46	0x03D4		66	0x0E
0x0E20	BCR	07	0x03D4		27	0x0E5C	BC	47	0x03D4		67	0x0E
0x03D4		08	0x09C4	LDR	28	0x09A6	LH	48	0x09D2	LD	68	0x11
0x03D4		09	0x258C	CDR	29	0x086E	CH	49	0x259A	CD	69	0x11
0x03E4	SVC	0A	0x270A	ADR	2A	0x06D2	AH	4A	0x2718	AD	6A	0x11
0x03D4		0B	0x2732	SDR	2B	0x07CE	SH	4B	0x2740	SD	6B	0x11
0x03D4		0C	0x19B4	MDR	2C	0x1872	MH	4C	0x19C2	MD	6C	0x11
0x30E4	Model 67 (BASR)	0D	0x2224	DDR	2D	0x03D4		4D	0x2232	DD	6D	0x11
0x03D4		0E	0x270A	ADR	2E	0x170C	CVD	4E	0x2718	AD	6E	0x11
0x0F5A	Test PSW/CC	0F	0x2732	SDR	2F	0x158C	CVB	4F	0x2740	SD	6F	0x11
0x0AD0	LPR	10	0x0A3E	LPDR	30	0x0D0C	ST	50	0x0CFA	STE	70	0x0D
0x0AE6	LNR	11	0x0A54	LNDR	31	0x03D4		51	0x03D4		71	0x10

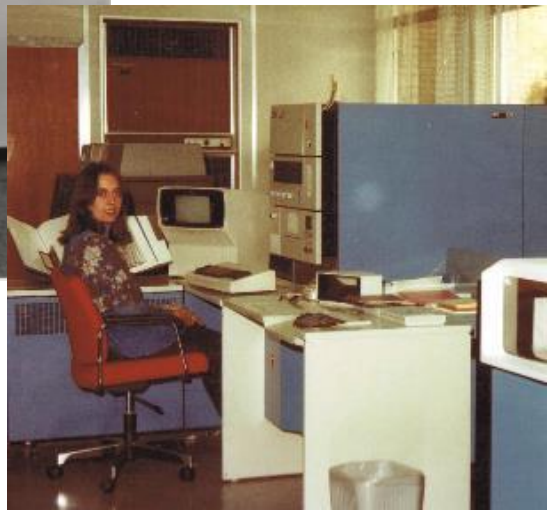
What about the BASIC side of things?



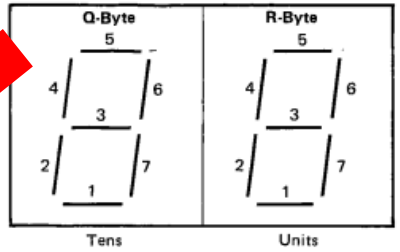
BASIC: Beginners' All-purpose Symbolic Instruction Code

System/3 Model 6 (c. 1970) aka IBM 5406

IBM 2265 Display Station
64x15 CRT



http://www.ibmsystem3.nl/stories/Jenny_C.html

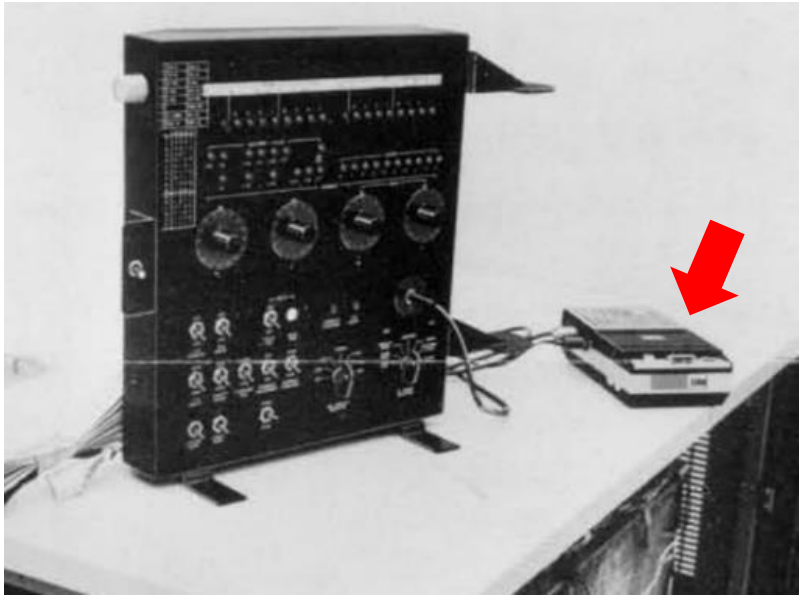


Bits 1-7 turn on bar lights 1-7, respectively.

LED accessible directly by using machine code instruction **F0**.

Spotted in an IBM System/3 Model 6 “Product Announcement” Brochure

C.E. Equipment



https://ed-thelen.org/comp-hist/IBM-ProdAnn/sys3m6.pdf

An audio tape recorder utilizing cassette cartridges will be used as an Alternate Program Load Device. The APLD will provide an alternate method of loading disk diagnostics into the system. It will also furnish an inexpensive means of updating the CE disk pack.

Earliest use of an audio tape cassette for digital data storage?

This is the same Carry Corder used on SCAMP!

On the IBM System/3 Model 6…
from Glenn Henry (IBMer involved in BASIC dev. on S/3)

- "System/3 Model 6 was designed for interactive usage…" (BASIC OS, RPG II, CRT)
- "This was probably the worst architecture ever imagined for engineering and scientific computing"
- "this system could do what the Apple II did, only much better (internal fixed disk, 64KB virtual memory, etc.), and **9 years earlier.**"
  - Except: "it weighed 1,300 lbs, it used 220V power, and it rented for about $1,000 a month."
- "In spite of my complaints about the System/3 instruction set, the **BASIC system actually performed well.**"

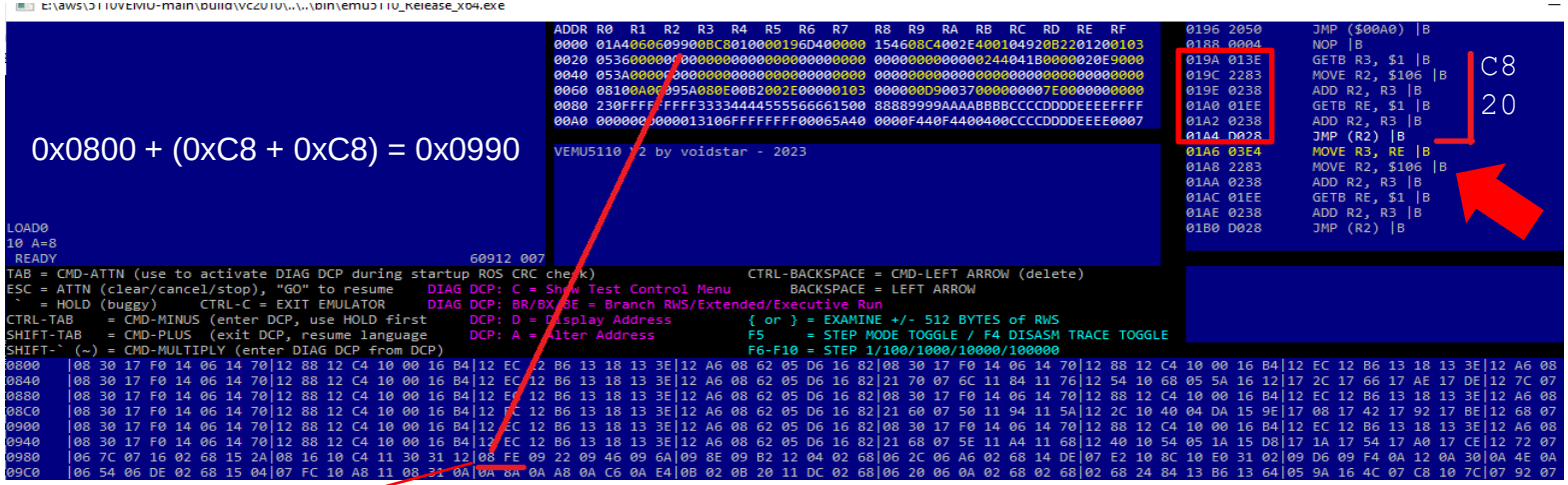
IBM System/3 Assembly Example

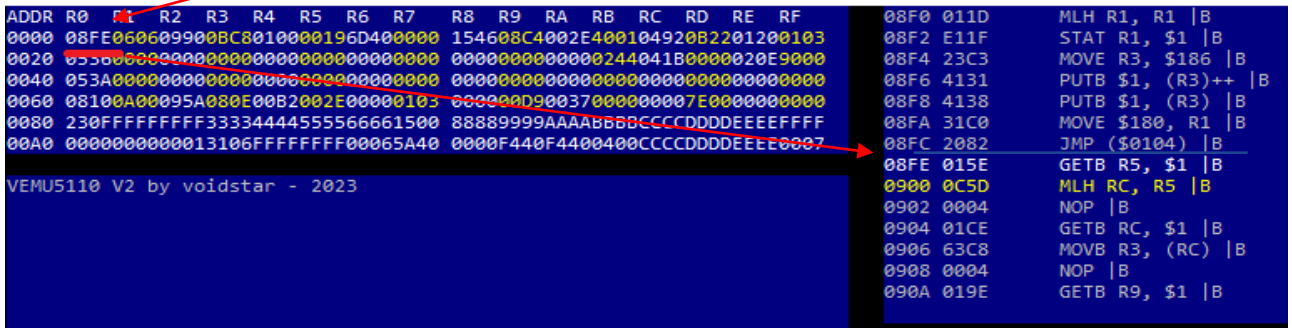
```
ADDR  OBJ. CODE

0017                                111          ORG    X'0017'                          REQUIRED FOR EACH CON
                              0078  112 FLG      EQU    START+X'78'                      AREA FROM X'78' TO X'
                                    113 *                                                  USABLE FOR WORKING
                                    114 *                                                  THIS BYTE USED TO F
                                    115 *                                                  ROUTINE IS REFERENC
                                    116 *                                                  SPECIFICATIONS
                              0000  117          USING  START,XR1                        VALID AT ENTRY TO ANY
0017  7C 00 78                      118          MVI    FLG(,XR1),X'00'                  INITIALIZE FLAG FOR N
                                    119 *                                                  ON FILE DESCRIPTION
001A  4E 01 43 030D                 120          ALC    #ENTRY(2,XR1),RELOCF             CALCULATE TRUE ENTRY
001F  C0 87 0338                    121          B      F1EAE1                           INITIATE SCAN OF 'F'
                              0000  122          USING  FCFG,XR2                         VALID UPON RETURN FRO
0023  6D 01 45 0C                   123 SPCA1    CLC    #IDENT(2,XR1),FCIDNT(,XR2)       IS THE IDENT THE RIGH
0027  B8 0A 0F                      124          TBN    FCDVA(,XR2),B'00001010'            AND IS DEVICE CODE
002A  B9 85 0F                      125          TBF    FCDVA(,XR2),B'10000101'            'SPECIAL'
002D  F2 96 07                      126          JC     SPCA2,X'96'                      IF THIS IS NOT THE RI
```

Humans wrote this? Yikes.

http://www.bitsavers.org/pdf/ibm/system3/plm/LY34-0001-1_System3_Model_6_BASIC_Logic_Manual_Jan72.pdf
https://bitsavers.org/pdf/ibm/system3/assembler/SC21-7509-7_System3_Basic_Assembler_Reference_Manual.pdf
http://www.bitsavers.org/pdf/ibm/system3/
GA21-9236-1_IBM_System-3_Model_8_10_12_15_Components_Reference_Manual_Nov77.pdf (page 33 for F0 LED notes)

Using 5110VEMU (PALM emulator)





Used internal breakpoints in emulator to determine that 0x019A is the “instruction fetch” routine for the BASIC emulation. (on 5110)

0B

2A

05

The BASIC NX OpCode Vector Table

- This is for IBM 5110 @ address 0x0000 (in BASIC NX ROS)
  - During startup, this table is copied into RWS at 0x0800
  - The 5100 has a similar convention (just different addresses)

	B	C	E	F	H	I	K	L	N	O	Q	R	T	U	W	X	Z	AA	AC	AD	AF	AG	AI	AJ	AL	AM	AO	AP	AR	AS	AU	AV
1	HEX		HEX		HEX		HEX		HEX		HEX		HEX		HEX		HEX		HEX		HEX		HEX		HEX		HEX		HEX		HEX	
2	00	0830	01	17F0	02	1406	03	1470	04	1288	05	12C4	06	1000	07	16B4	08	12EC	09	12B6	0A	1318	0B	133E	0C	12A6	0D	0862	0E	05D6	0F	1682
3	10	0830	11	17F0	12	1406	13	1470	14	1288	15	12C4	16	1000	17	16B4	18	12EC	19	12B6	1A	1318	1B	133E	1C	12A6	1D	0862	1E	05D6	1F	1682
4	20	0830	21	17F0	22	1406	23	'1470	24	1288	25	12C4	26	1000	27	'16B4	28	12EC	29	12B6	2A	1318	2B	133E	2C	12A6	2D	0862	2E	05D6	2F	1682
5	30	2170	31	076C	32	1184	33	1176	34	1254	35	1068	36	055A	37	1612	38	172C	39	1766	3A	17AE	3B	17DE	3C	127C	3D	07B8	3E	04BA	3F	1584
6	40	0830	41	17F0	42	1406	43	1470	44	1288	45	12C4	46	1000	47	16B4	48	12EC	49	12B6	4A	1318	4B	133E	4C	12A6	4D	0862	4E	05D6	4F	1682
7	50	0830	51	17F0	52	1406	53	1470	54	1288	55	12C4	56	1000	57	16B4	58	12EC	59	12B6	5A	1318	5B	133E	5C	12A6	5D	0862	5E	05D6	5F	1682
8	60	0830	61	17F0	62	1406	63	1470	64	1288	65	12C4	66	1000	67	16B4	68	12EC	69	12B6	6A	1318	6B	133E	6C	12A6	6D	0862	6E	05D6	6F	1682
9	70	2160	71	0750	72	1194	73	115A	74	122C	75	1040	76	04DA	77	159E	78	1708	79	1742	7A	1792	7B	17BE	7C	1268	7D	07A0	7E	047E	7F	1554
10	80	0830	81	17F0	82	1406	83	1470	84	1288	85	12C4	86	1000	87	16B4	88	12EC	89	12B6	8A	1318	8B	133E	8C	12A6	8D	0862	8E	05D6	8F	1682
11	90	0830	91	17F0	92	1406	93	1470	94	1288	95	12C4	96	1000	97	16B4	98	12EC	99	12B6	9A	1318	9B	133E	9C	12A6	9D	0862	9E	05D6	9F	1682
12	A0	0830	A1	17F0	A2	1406	A3	1470	A4	1288	A5	12C4	A6	1000	A7	16B4	A8	12EC	A9	12B6	AA	1318	AB	133E	AC	12A6	AD	0862	AE	05D6	AF	1682
13	B0	2168	B1	075E	B2	11A4	B3	1168	B4	1240	B5	1054	B6	051A	B7	15D8	B8	171A	B9	1754	BA	17A0	BB	17CE	BC	1272	BD	07AC	BE	049C	BF	156C
14	C0	067C	C1	0716	C2	0268	C3	152A	C4	0816	C5	10C4	C6	1130	C7	3112	C8	08FE	C9	0922	CA	0946	CB	096A	CC	098E	CD	09B2	CE	1204	CF	0268
15	D0	062C	D1	06A6	D2	0268	D3	14DE	D4	07E2	D5	108C	D6	10E0	D7	3102	D8	09D6	D9	09F4	DA	0A12	DB	0A30	DC	0A4E	DD	0A6C	DE	11B4	DF	0268
16	E0	0654	E1	06DE	E2	0268	E3	1504	E4	07FC	E5	10A8	E6	1108	E7	310A	E8	0A8A	E9	0AA8	EA	0AC6	EB	0AE4	EC	0B02	ED	0B20	EE	11DC	EF	0268
17	F0	0620	F1	060A	F2	0268	F3	0268	F4	0268	F5	2484	F6	13B6	F7	1364	F8	059A	F9	164C	FA	07C8	FB	107C	FC	0792	FD	077C	FE	01A6	FF	0150

(start of **5110 BASIC NX ROS** in support of **"10 A=8"** BASIC statement)

```
                                    CardCode   Mnem.
0924: C8 20 0B 2A 05 C4               BA8
      CA 0F 0B 2A 23 EE             D BA8 2
      3B 10 0B 6C                    C 8 21    SBF
      38 80 0B A1 04 9F             DC 8       TBN
        31 00 0B A8    (skipped)
      3C 40 05 C0                    C 84      MVI
      3C 00 0B 57                    C 84      MVI
      09 01 0B 58 00 00             DCBA8 1
      3B 10 0B 6D                    C 8 21    SBF
      3B 08 0B A4                    C 8 21    SBF
      33 40 01 3F                   DC   21
      3C FF 10 CE                    C 84      MVI   (set  RWS[10CE] == FF)
      04 40 10 CF 10 CE             DCBA 4     ZAZ   (copy RWS[10CE] into 10CF++, 320 times)
      3B 04 0B 6D                    C 8 21    SBF
      35 40 0B 63                   DC  4 1    L
      33 50 00 03                   DC   21
      F8 40 50                         8
      C0 A0 04 D5                   D          BC
…
09DE: 33 40 00 05                   DC   21
      33 10 05 80                   DC   21
      33 30 05 BF                   DC   21
      F1 05 F4                         1       APL
…
0BE8: 34 80 0F D9                   DC  4      ST
      33 30 05 C0                   DC   21
      DD 40 00 06 01                D B 84 1
```

For System/3, not all OpCodes had an assembler mnemonic.

CODE CONVERSIONS

Dec Val	Hex Val	Card Code DCBA8421	Mnem	IPL (Note 1) T1T3	IPL (Note 1) T2T3	EBCDIC Code 01234567
000	00	C		4	1	00000000
001	01	DCBA 1		A @	A 3	00000001
002	02	DCBA 2		B @	B 3	00000010
003	03	DCBA 21		C @	C 3	00000011
004	04	DCBA 4	ZAZ	D @	D 3	00000100
005	05	DCBA 4 1		E @	E 3	00000101
006	06	DCBA 42	AZ	F @	F 3	00000110
007	07	DCBA 421	SZ	G @	G 3	00000111
008	08	DCBA8	MVX	H @	H 3	00001000

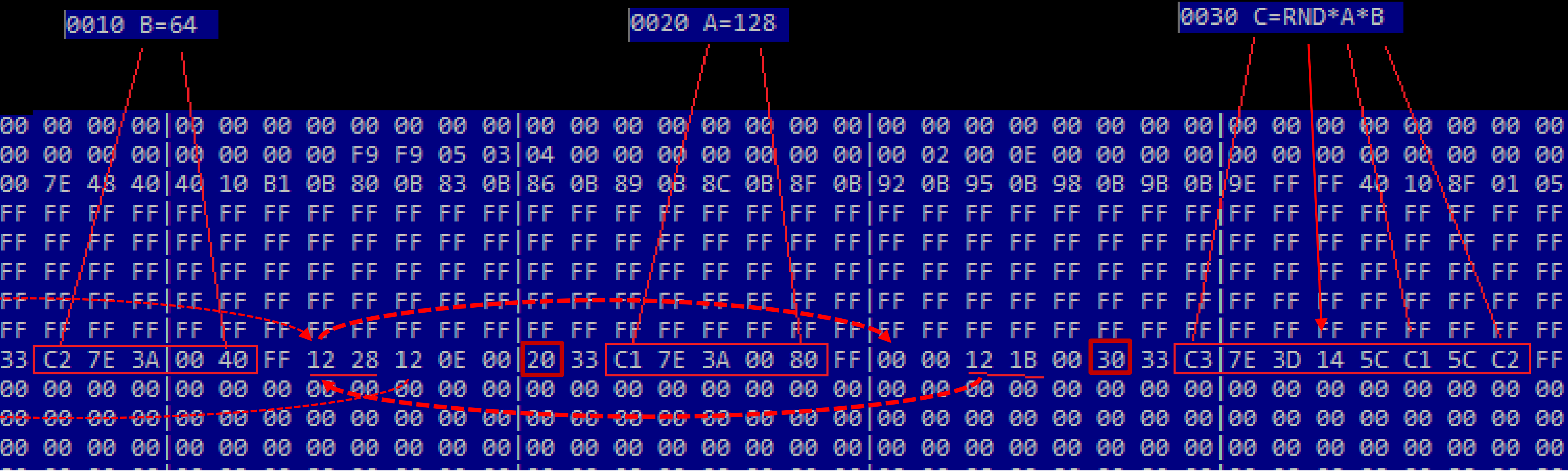
IBM System/3 Component Reference Manual
GA34-0001-3 (Model 6 @ archive.org)
GA21-9236-1(other Models)

Confirmed similar for the IBM 5100, using **0x06B8** as starting point:

```
06B8: 3B 10 0B 69
      3D 0F 0B 1C 10 48 39
      38 80 0B 97 06 CD
      31 00 0B 9A
      3C 00 05 C0
      3B 10 0B 6A
      09 FF 0F D4 00 FF
      09 49 10 D4 00 FF
      33 40 00 05
      33 10 05 80
      33 30 05 BF
      F1 08 9C
      F0 07 33          ;  "7h"
      33 20 0F D4
      7D 1B 00 18 07 0C
```

BASIC Tokens @ 120Eh

- As with more modern implementations of BASIC, each line of BASIC is “tokenized” into a byte sequence
- A “linked list” (next/prev. pointers) of tokens starts at RWS address **0x120E** (just after a long marker of 0xFF’s)
- NOTE: The “FF” markers change to “40” after a **RUN** (garbage collection), then revert back to “FF” after a **LIST**



P.A.L.M. Assembly Example

```
ADDR OBJ.      ASM.
2000 D601 ABCD LWI R6, #$ABCD
2004 0203      INC2 R2, R0
2006 D021      JMP (R2)+
2008 2016      JMP ($002C)
200A 8C03      LBI RC, #$03
200C 5B21      MOVE (R2)+, RB
200E 0CC1      DEC RC, RC
2010 CC03      SZ RC
2012 F007      BRA $200C  ; -> -
2014 0000      HALT
2016 DE01 0200 LWI RE, #$0200
201A D901 4040 LWI R9, #$4040
201E 8306      LBI R3, #$06
2020 59E1      MOVE (RE)+, R9
2022 CE37      SBSH RE, R3
2024 F005      BRA $2020  ; -> -
2026 0024      RET R2
2028 0000      HALT
202A 0000      HALT
```

```
1         cpu         IBM5110
2         include     "ebcdic_5110.inc"  ; or use ebcdic.inc
3         intsyntax   +$hex,-x'hex'      ; support $-style hex (not IBM 0x style)
4         codepage    cp037              ; activate a string mapping of chars
5  ADDR_SCREEN        equ $0200          ; Address of CRT display
6         ORG   $2000
7         LWI   R6, #$ABCD
8         CALL  ClrScr, R2   ; Call Clear Screen (using R2 as return address)
```

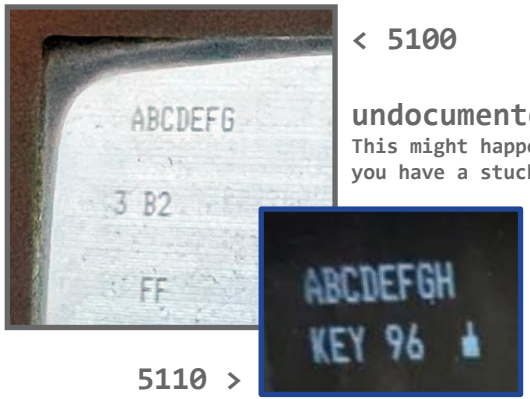
```
9  ; Loop Demonstration --------
10        LBI   R12, #3      ; R12 = $ xx03 (prepare to loop 3 times)
11 DEMO1:
12        ; Copy full word value of R11 to three consecutive RWS/RAM addresses.
13        MOVE  (R2)+, R11   ; RWS[R2] = R11 ($40CD) --> R2 = R2 + 1 = 0B02
14        DEC   R12          ; LO(R12) = LO(R12) - 1 (decrement loop counter)
15        SZ    R12          ; SKIP IF LO(R12) IS ZERO  ("while R12 not zero")
16        BRA   DEMO1        ; not skipped: repeat loop
17 ; --------------------------
18        HALT
```

```
19 ClrScr:
20        LWI   R14, #ADDR_SCREEN   ; R14 used as the screen offset
21        LWI   R9,  #"  "          ; R9 full word is used as screen code that is
22                                  ;   to be written (e.g. double space: "  ")
23        LBI   R3,  #$06           ; LO[R3] = $06 (checking to see if our address
24                                  ;   pointer has reached $0600 yet)
25 cs_2:  MOVE  (R14)+, R9          ; RWS[R14] = R9, then R14 = R14 + 2 (next addr)
26                                  ;   (^ this does the actual drawing)
27        SBSH  R14, R3             ; Have we reached $0600 yet?
28                                  ;   (one past end of screen)
29                                  ; Skip if all set bits in R3 are also set
30                                  ;   in HI(R14)
31        BRA   cs_2
32        RET   R2
```

Working on preparing a better explanation of PALM assembler (will be in github), but here is a few simple examples.

DCP: Diagnostics Control Program

- All IBM 5100's have a built-in DCP that can be used to modify any memory address

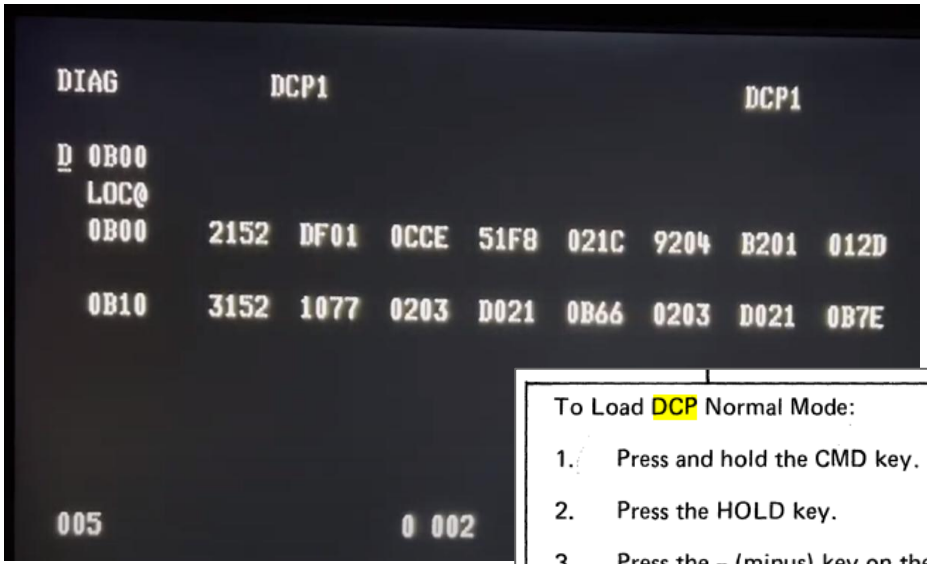


< 5100

undocumented!
This might happen if you have a stuck key.

5110 >

During ROS CRC checks, you can enter DCP by pressing: **CMD-ATTN.** However, **if you press a key too early, the system shows you the scan code of the key pressed!**



To Load DCP Normal Mode:

1. Press and hold the CMD key.
2. Press the HOLD key.
3. Press the – (minus) key on the numeric keyboard.
4. Release the CMD key. (DCP is now loaded)

IBM 5100/5110 NEWS

- **Christmas Star 2022 Contest**
  - Sponsored by Logiker (Dec 2, 2022 – Dec 26, 2022)
    - https://logiker.com/Vintage-Computing-Christmas-Challenge-2022
    - Output a **17x17** symmetric star in as few bytes as possible
  - All results at **demozoo.org**
    - IBM 5110 APL 3rd place (Peter De Watcher, 30 bytes)
      - https://demozoo.org/productions/316775/

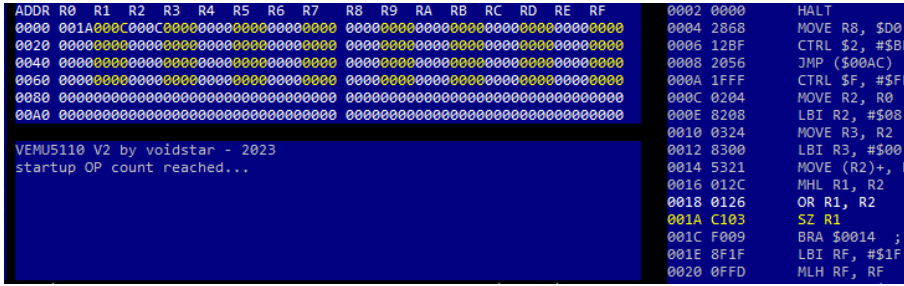
```
CLEAR WS
      ' *'[1+V∘.≤-V←B,0,⌽B←A,-A←⌽⍳4]_
```

    - IBM 5100 BASIC 3rd place **WILD** (voidstar)
      - https://demozoo.org/productions/317067/

IBM 5100/5110 NEWS

- **IBM 5110 SLM (System Logic Manuals) digitized on github (JPG)**
  - https://github.com/voidstar78/IBM5110_SLM
  - For IBM 5100, these are part of MIM Section 5
  - **JUST IN:** Now have complete SLM for 5120 also!

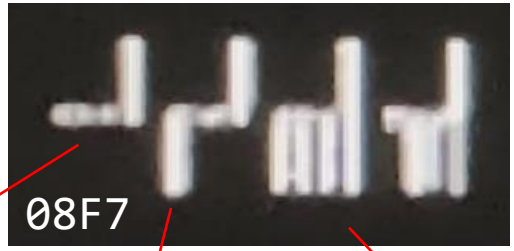
- **VEMU5110 now has an integrated PALM disassembler**
  - V2 branch in github
  - Still struggling to fully emulate 5100-system
  - Still investigating access to external pins

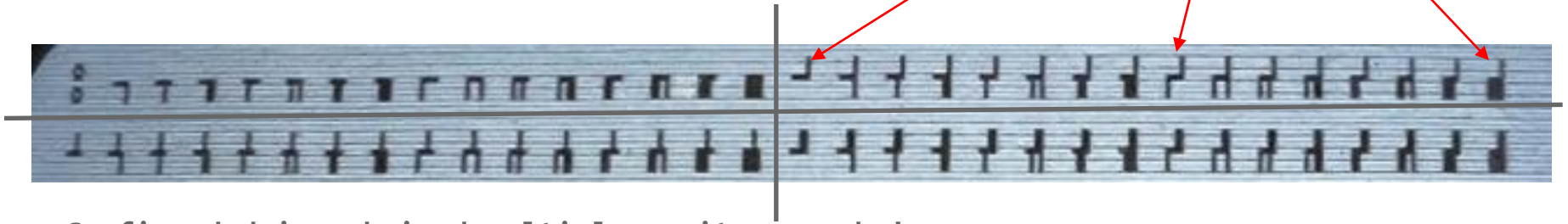


- **Alfred Arnold Macro AS Assembler now supports PALM**
  - Supports both IBM and Corti-style mnemonics
  - http://john.ccac.rwth-aachen.de:8000/as/

IBM 5100/5110 NEWS

- **“Discovered” use of the IBM 5110 character set**
  - Linear binary sequencing
  - Unique to the 5110 (?)





- **Confirmed daisy-chained multiple monitors works!**
  - Superbat 3G/HD SDI Cable BNC Cable
  - JVC 9in to 14in BNC monitor **(plus BNC to VGA adapters)**

IBM 5100/5110 NEWS

- Reminders…
  - Have a serial adapter based on Arduino or ESP32 to stream into the IBM 5100-series keyboard [requires 330ohm resistors]
    - Notes on github “KBD5110” project
  - Have an approach to replacing the IBM 5100-series power supply (based on a modern Mean Well PSU)

See notes at:

https://voidstar.blog/ibm-5100-personal-computer/

END.

- Thank-you to bitsavers.org & archive.org
- Thank-you to Christian Corti

- “ASCII” BadApple demo on 5100 when? ;)

  64KB in memory-only (few seconds); use 5110 for speaker

Key Takeaways

Key Takeaways

- Per Hal Prewitt, IBM 5100 to 5150 data exchange is possible using only PALM instructions and the external I/O pins
  Could a WiModem232 compatible terminal be developed to get the IBM 5100 onto the internet? Could BadApple be streamed to an IBM 5110 display?

- The FORM, USING, PIC keywords gave the IBM 5110 BASIC some interesting features for data entry
- PEEK/POKE predate Microsoft BASIC by several years

- Hardware signals can be manipulated to override the intent of software (re: IBM 5100 Executable ROS extraction process)

- For the IBM 5100, the IBM System/360 OpCode Vector table is at 0x17E00 of the APL NX ROS (0x1C800 for the 5110)
- The use of OpCode 0D suggest the **IBM System/360 Model 67** was used

- The Norelco Carry-Corder used in the SCAMP prototype was also used in the IBM System/3 CE equipment
- The IBM 5110 ROS contains an interesting set of plain-English words when converted from EBCDIC to ASCII

- For the IBM 5100 and 5110, the IBM System/3 OpCode Vector table is at 0x0000 of the BASIC NX ROS
- This table is copied to 0x0800 of the RWS during startup
- For the IBM 5110, BASIC tokens are stored at address 0x120E of the RWS

- 5110VEMU is a new front-end to the Christian Corti IBM PALM processor emulator
- An MCU can be used as a keyboard substitute for the IBM 5100/5110/5120
- A replacement to the IBM 5100/5110 power supply is available using modern parts

- Alfred Arnold Macro AS assembler now supports PALM instruction set (which can be input using the built in DCP)

Key Takeaways

Looking for a copy of the CORE PC-51 program for IBM PC-DOS

Looking for any info about the CoreNET hard drive available for IBM 5110/5120

Looking for copies of IBM GB30-xxxx articles

Looking for anyone willing to repair IBM 5110 Power Supply

Looking for any stories from those who used an IBM 5100 or involved in its development

Looking to contact Harry Katzan, Jr (author of IBM 5100 Portable Computer and still actively writing fiction stories as recent as 2019)

Additional Resources

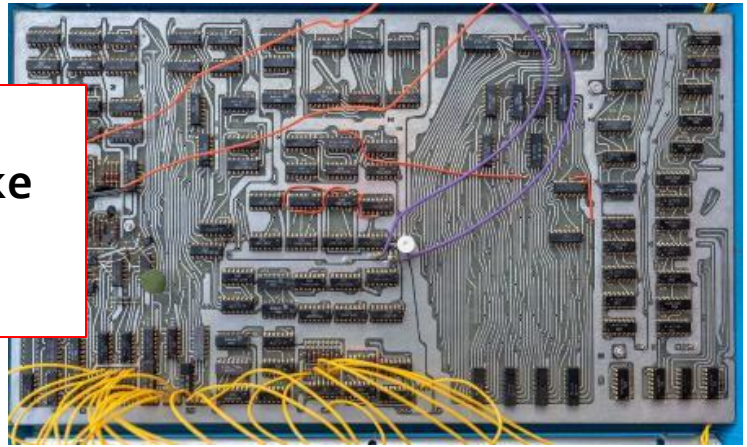
Additional Resources

- **IBM 5100/5110 Reference Info**
  - https://voidstar.blog/ibm-5100-personal-computer/
  - https://voidstar.blog/ibm-5110-notes/
  - https://github.com/voidstar78/IBM_5100_DOCS
  - https://github.com/voidstar78/IBM5110_SLM
  - http://computermuseum.informatik.uni-stuttgart.de/dev/ibm_5110/technik/en/
- **5110VEMU**
  - https://github.com/voidstar78/5110VEMU
- **KBD5110**
  - https://github.com/voidstar78/KBD5110
- **Macroassembler AS (Alfred Arnold) [now supports PALM]**
  - http://john.ccac.rwth-aachen.de:8000/as/
- **IBM 5110 PALM Christmas Tree Demo**
  - https://www.youtube.com/watch?v=2pKpCWdYujA
- **IBM 5100 Tape Usage Demo**
  - https://www.youtube.com/watch?v=e2GYWyZyfpE
- **“A Microprocessor-based portable computer: The IBM 5100”** (IEEE Volume 64 issue 6, June 1976)
- SCAMP and J.D.George related documentation at Smithsonian Institute
  - https://americanhistory.si.edu/collections/search?return_all=1&edan_local=1&edan_q=1988.3146&

What is Microcode?

- “Introduction to Microcoded Implementation of a CPU Architecture”
  - N.S. Matloff, Jan. 1999
- “How the 8086 processor's microcode engine works”
  - Ken Shirriff, Dec. 2022
- “The IBM System/360 and the Use of Microcode” – SciHi Blog, April 2022
- “DOWN THE INTEL MICOCODE RABBITHOLE” – HACKADAY, JULY 2022

"Microcode allows a processor with a small instruction set to function like one with a larger instruction set."
- smoore, 2010 (StackExchange)

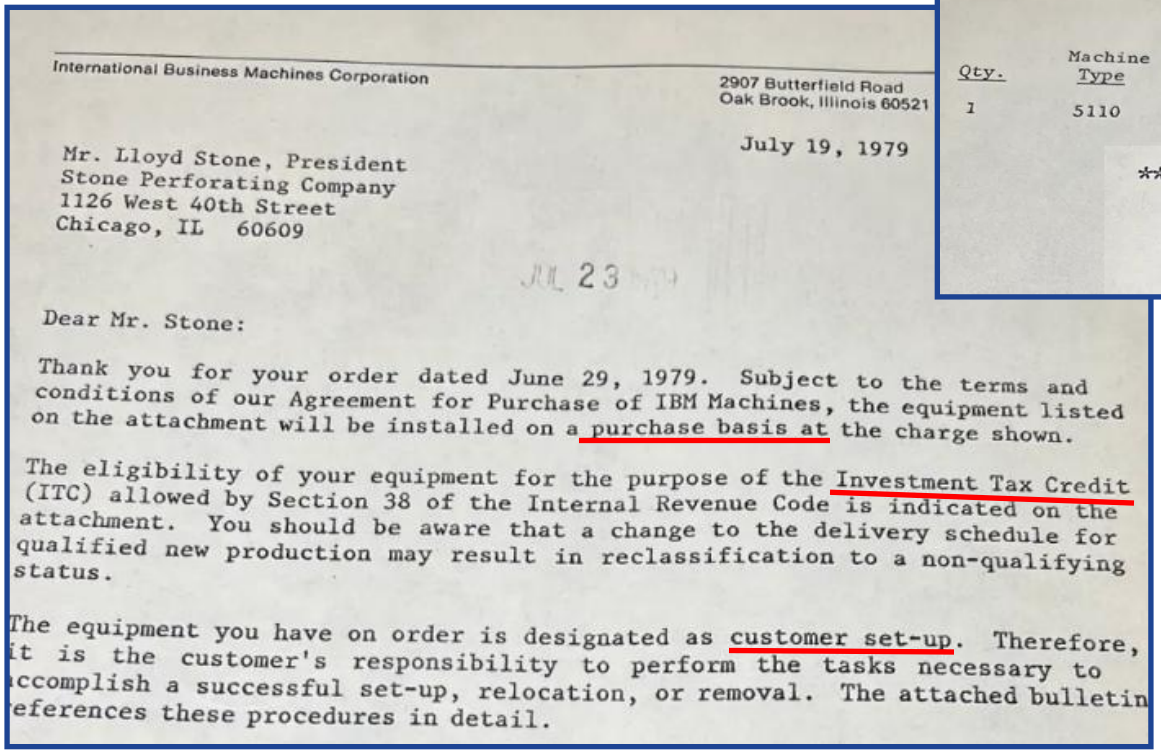


Example of IBM 5110 Usage

Publications referenced in…
IBM 5110/5120 Computing Systems Bibliography (GH30-0232-1)

- GB30-1038 IBM 5110 **Bowling League** Scoring System
- GB30-1180 IBM 5110 Linear Programming **Meat Blending System**
- GB30-1263 IBM 5110 Computing System For Insurance Agents and Brokers
- GB30-1126 IBM 5110 Computing System Construction Payroll/Labor Costing
- GB30-1268 IBM 5110 **Doctors Office** Management System
- GB30-1034 IBM 5110 Computing System **Air Freight Rating Optimization System**
- GB30-1113 IBM 5110 Computing System General Ledger System
- GB30-1242 IBM 5110 Computing System Fixed Asset Accounting and Control System
- G360-0638 Viewpoint IBM 5110 **A furniture manufacturer, a bank, and a hospital**

I have not yet found hard-copies of these articles (GB30-xxxx).
But this is evidence to the variety of uses of these systems.



International Business Machines Corporation

2907 Butterfield Road
Oak Brook, Illinois 60521

July 19, 1979

Mr. Lloyd Stone, President
Stone Perforating Company
1126 West 40th Street
Chicago, IL 60609

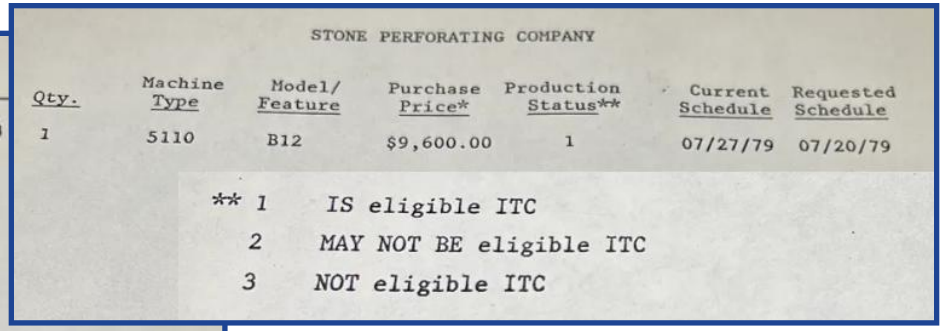
JUL 23

Dear Mr. Stone:

Thank you for your order dated June 29, 1979. Subject to the terms and conditions of our Agreement for Purchase of IBM Machines, the equipment listed on the attachment will be installed on a purchase basis at the charge shown.

The eligibility of your equipment for the purpose of the Investment Tax Credit (ITC) allowed by Section 38 of the Internal Revenue Code is indicated on the attachment. You should be aware that a change to the delivery schedule for qualified new production may result in reclassification to a non-qualifying status.

The equipment you have on order is designated as customer set-up. Therefore, it is the customer's responsibility to perform the tasks necessary to accomplish a successful set-up, relocation, or removal. The attached bulletin references these procedures in detail.



STONE PERFORATING COMPANY

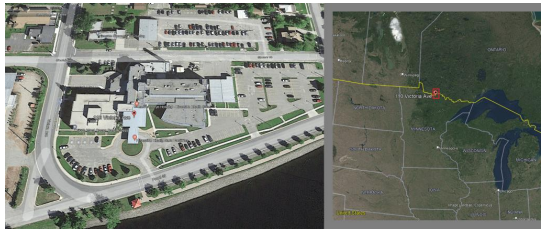
Qty.	Machine Type	Model/ Feature	Purchase Price*	Production Status**	Current Schedule	Requested Schedule
1	5110	B12	$9,600.00	1	07/27/79	07/20/79

** 1 IS eligible ITC
2 MAY NOT BE eligible ITC
3 NOT eligible ITC

Example of IBM 5110 Users

Example of IBM 5110 Users

- A) “The first **5110** was shipped from the GSD's Rochester, Minn., plant on February 2, 1978, to **Punxsutawney Electric Repair Company**, a small electrical products distributor in **Pennsylvania**. The customer used the **5110** for billing, inventory control, accounts receivable and sales analysis.”
  - Contacted **Jeff Grube**, vice president of Punxsutawney Electric Repair (c. 2021)
  - Confirmed: “I programmed three of them that our businesses used until pc’s replaced them.”
- B) Insurance company in Goldthwaite, TX (5110 Type-2) for General Ledger, Payroll and Accounts Payable for local farms and businesses
- C) Electronics components (filters) business in New Jersey (accounting and payroll) [krfilters.com]
- D) The **IBM 5100** I acquired has matching labels on the main unit and the IBM 5106 tape unit:
  - La Verendrye General Hospital, Fort Frances, Ontario 0262



CORE

Association of Users for
Small IBM Computer Systems

542 South East 5th Avenue
Delray Beach, Florida 33444
(305) 276-3929

General Specifications

Storage Capacity — Formatted

CORE 5115 Removable	10.6 MB
CORE 5116 Fixed	42.4 MB
CORE 5117 Fix/Remov	53.0 MB
Per sector:	256 Bytes
Sector/track:	48
Max Files/Disk:	2,000
Max Records/File:	65,535
Max File Size:	9,999 KB

Run Times

Time to SORT 3,000 records:	1.34 min
Average Access:	40 MS
To Drive Ready:	45 sec
Average Increase in Computer Performance:	30%
Average Time to Copy 42 MB to removable:	15 min

Error Rates — Data Transfer

Recoverable:	1 in 10^{10}
Non-Recoverable:	1 in 10^{12}
Seek errors:	1 in 10^{9}

Reliability & General Data

Mean time between failures:	8,000 Hours
Voltages:	110 or 220 AC 6 or 3 Amps

Rotational Speed: 3600 RPM
Data Transfer: 875 K Byte/s
Max Drives/System: 4
Max Users/System: 8
Temperature: 50 to 104 F
Humidity: 20 to 80%
Memory Required: 1.5 KB
Installation Time: 3 min
IBM Service Required: No

SAVE IBM 5110/20's

from the junk yards of the world!

NATIONAL ASSOCIATION OF OWNERS &
OPERATORS OF SMALL IBM COMPUTERS

CORE

OUR 5th YEAR - VOL 4 -- No. 3 SEPTEMBER 1982 NEWSLETTER

IBM 5110/20 RUN FASTER

NEW EQUIPMENT FOR IBM 5110 & 5120 USERS
September 1, 1982 - LARGE STORAGE, HIGH SPEED DISK. It is now possible to increase the speed and storage capabilities of all IBM 5110 and 5120 Computer Systems. The new disks are available in two models. The first is called the CORE 5115 which stores 10.6 MB of data on a removable cartridge. (That's 10 times the

WHO IS CORE ?
Each week we receive many calls from new computer owners asking the function and services of CORE. We are the first, the largest and exclusively an IBM 5100, 5110, 5120 and System 23 Datamaster user's group. We are a national association of computer owners and operators. There are other groups which just sell programs and print newsletters, but we do much more. Our function is simple, we provide services to allow our members to make their computer more useful. We do this by offering major discounts on IBM computer supplies, develop low cost computer programs, provide newsletters and design new products to solve the computer problems of our members. CORE lowers the cost of computing. We offer a 20% discount on the IBM Maintenance Agreement, 20% and more off on IBM computer supplies and wholesale prices on many

CONNECT MULTIPLE COMPUTERS

Multiple 5110/5120's can now be attached and share files on any of the hard disks up to a system limit of 8 computers. In addition, new expansion cables are available to increase the distance between computers, disks and printers up to 30 feet.

Some PALM Processor Technical's

Oscillators/Clocks: The processor uses a 15.1 MHz oscillator to generate 66.2 nanosecond clock pulses. These multiclock cycle (MCC) pulses are used to control data throughout the computing system. MCC pulses make up the I phase (instruction) and E phase (execute) machine cycles.

Machine Cycles

Every machine cycle consists of an I-phase (instruction) and an E-phase (execute). Each I-phase consists of three I cycles. I cycles 1 and 3 consist of 3 MCC pulses. I-cycle 2 consists of 8 MCC pulses. Every I-phase is followed by an E-phase. Each E-phase consists of one through five E cycles. E cycles can be either 3 MCC or 8 MCC pulses, depending on the instruction being executed.

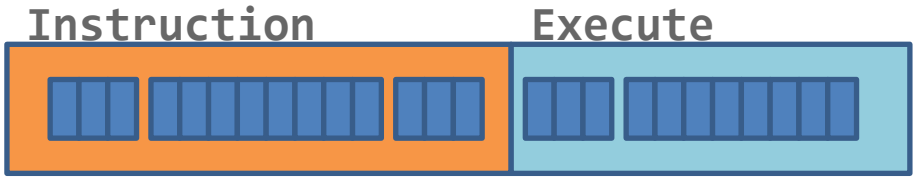
I-Phase: Each I-phase consists of three I cycles for each microinstruction.

I cycle 1 — The contents of program level register 0 are loaded into SAR.

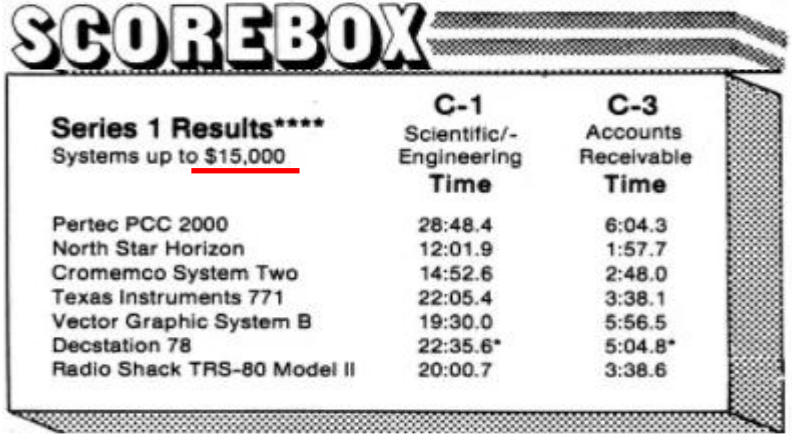
I cycle 2 — SAR addresses either R/W storage or executable ROS and transfers the microinstruction located at that address into the operation register.

I cycle 3 — The address in the SAR is incremented by two and read back into register 0 of the current program level.

E Phase: During the E phase, the processor performs the operation specified in the Op register. E cycles continue until the operation is completed. Only R/W storage can be addressed during execute cycles.



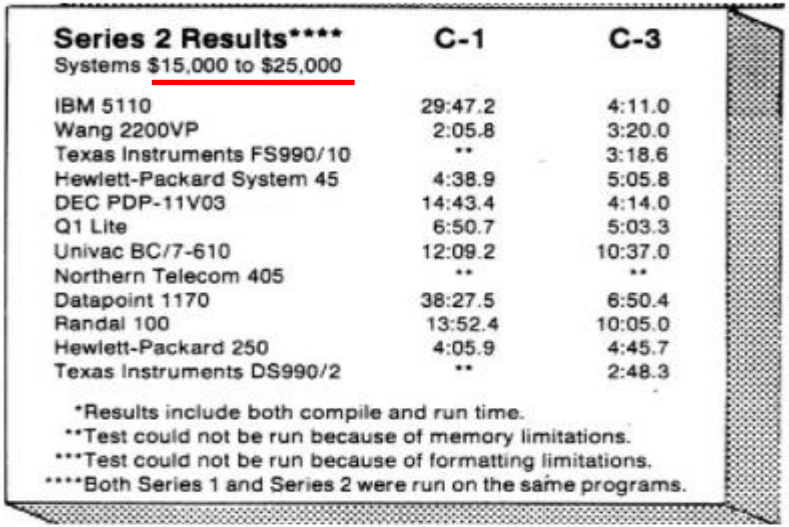
Computerworld Aug. 1980 Benchmark



SCOREBOX

Series 1 Results**** Systems up to $15,000	C-1 Scientific/-Engineering Time	C-3 Accounts Receivable Time
Pertec PCC 2000	28:48.4	6:04.3
North Star Horizon	12:01.9	1:57.7
Cromemco System Two	14:52.6	2:48.0
Texas Instruments 771	22:05.4	3:38.1
Vector Graphic System B	19:30.0	5:56.5
Decstation 78	22:35.6*	5:04.8*
Radio Shack TRS-80 Model II	20:00.7	3:38.6

This is the 21st in a series of articles giving the highlights of benchmark tests conducted on popular small computer systems. The full reports were originally published by the Association of Computer Users, a 4,000-member nonprofit organization.



Series 2 Results**** Systems $15,000 to $25,000	C-1	C-3
IBM 5110	29:47.2	4:11.0
Wang 2200VP	2:05.8	3:20.0
Texas Instruments FS990/10	**	3:18.6
Hewlett-Packard System 45	4:38.9	5:05.8
DEC PDP-11V03	14:43.4	4:14.0
Q1 Lite	6:50.7	5:03.3
Univac BC/7-610	12:09.2	10:37.0
Northern Telecom 405	**	**
Datapoint 1170	38:27.5	6:50.4
Randal 100	13:52.4	10:05.0
Hewlett-Packard 250	4:05.9	4:45.7
Texas Instruments DS990/2	**	2:48.3

*Results include both compile and run time.
**Test could not be run because of memory limitations.
***Test could not be run because of formatting limitations.
****Both Series 1 and Series 2 were run on the same programs.

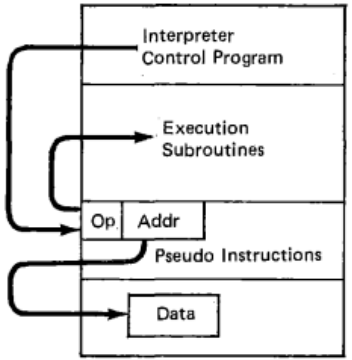
Not stellar performance by 1980, but…
It was **Portable** and Well supported.
Unsure if the test disabled CRT (cycle-stealing overhead).
A lot of existing software was available.
Doesn't reflect the raw processor performance. (PALM)

PML/PMC (Pseudo Machine Language/Code)

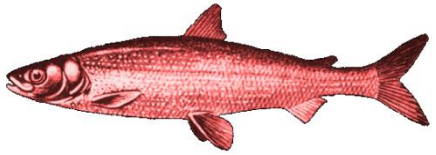
PSEUDO MACHINE LANGUAGE CONCEPT (PSEUDO OBJECT PROGRAM)

A pseudo machine language (object program) concept speeds the compilation time of a user program. It reduces the quantity of instruction output by the compiler and eliminates the necessity for an assembly pass or passes over the output instructions.

The pseudo instructions that make up the pseudo machine language (object program) invoke the execution of preassembled machine-language execution subroutines to perform the functions indicated by the pseudo instructions. This concept (Figure 2-5) is similar to the emulation of an instruction set foreign to the object machine, or the execution of machine instructions by hardware microprogramming.



No evidence found that this is used on the IBM 5100-series.



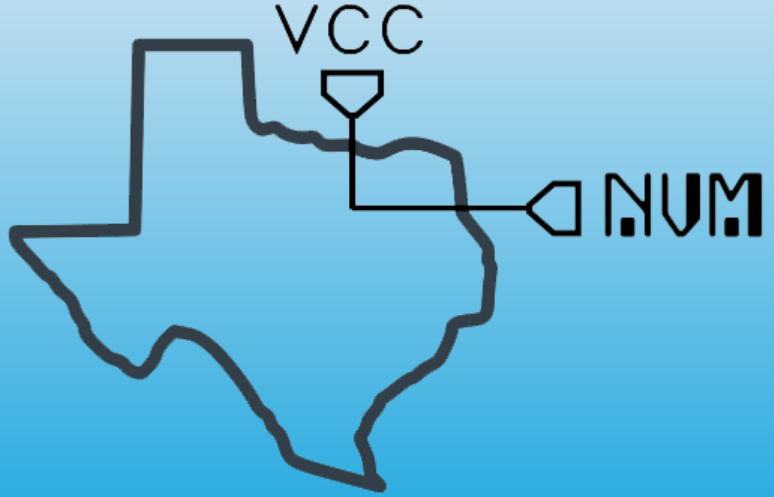
(3-227, page 263)

Mnemonic	Length (bytes)	Operand	Hexadecimal Op Code	Name
ADD	1	*	06	Add
ADF	2	XX	58	Activate external data file
BNX	3	VADR	4A	Branch and suppress execution
BRA	3	VADR	46	Branch unconditionally
BRC	4	VADR CC	44	Branch on condition
BRD	3	VADR	48	Branch and delete function entry
BRS	1	*	4C	Branch to stacked address
CLS	1	*	5E	Close external data file
CMC	1	*	42	Compare character elements
CMF	1	*	40	Compare floating point values
CSA	2	NN	3E	Compute stacked address
DCA	3	VADR	6A	Define constant address
DDL	3	VADR	6C	Define data linkage
DIV	1	*	0C	Divide
DWA	2	NN	6E	Define work area
EOF	1	*	70	End of program
EOP	1	*	68	End of page

System/3 Model 6 BASIC Logic Manual (Jan '72, page 23)

NVM

NATIONAL
VIDEOGAME
MUSEUM



VCF Southwest
June 23rd-25th 2023