Disassembling an UFO

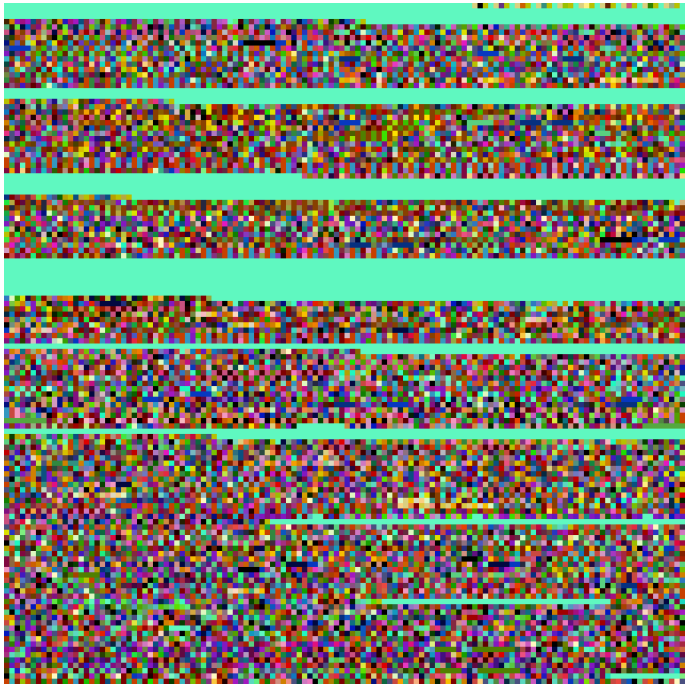
Hacking an unknown instruction set

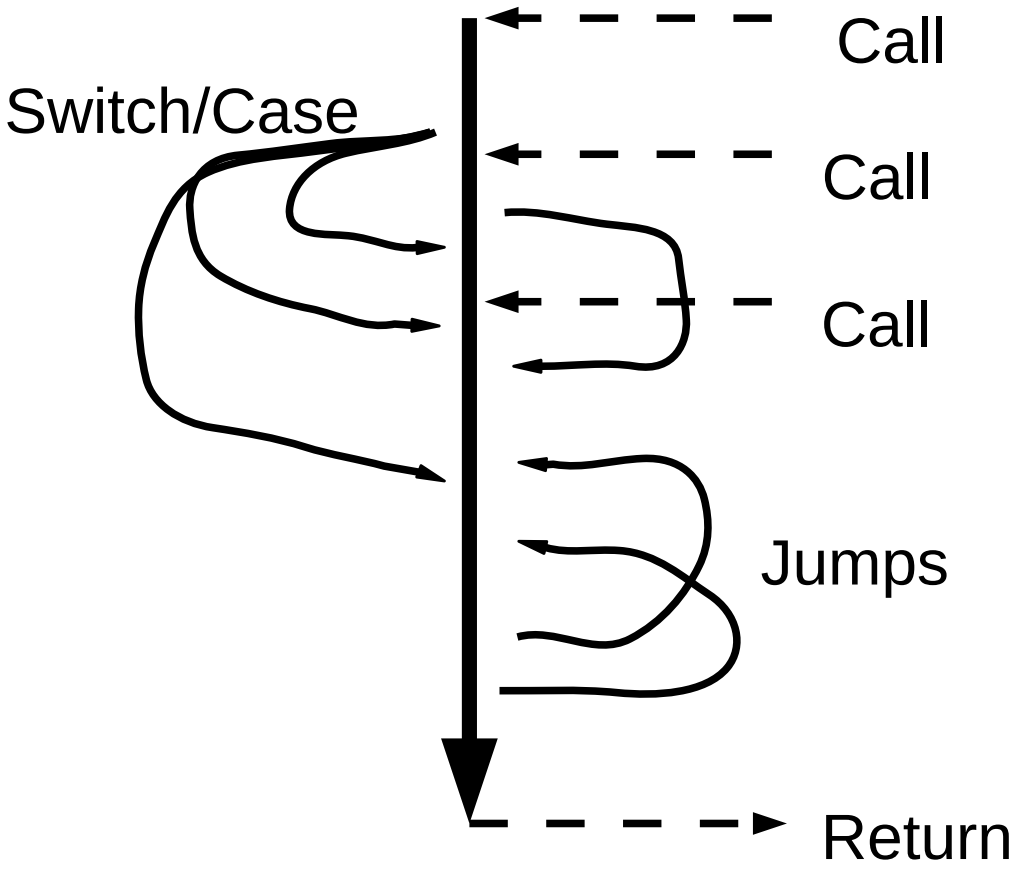

Fry's
ELECTRONICS
STORE HOURS
MON - FRI
9AM - 9PM
SATURDAY
9AM - 9PM
SUNDAY
9AM - 7PM
EXIT
ENTRANCE

```
00000000  c8 37 84 98 c8 31 80 10  c8 32 80 14 c8 33 80 18
00000010  c8 34 80 0a c8 35 80 16  c8 36 80 1c c8 37 80 08
00000020  c8 37 85 78 c8 31 86 f9  c8 20 c9 a3 4a a3 5b 5e
00000030  c8 10 c9 d2 d0 f2 4a 09  cf ef 00 5a c8 b0 30 80
00000040  50 31 80 56 d0 02 c9 0d  4a c9 f4 36 c9 f6 80 5a
00000050  d0 04 c9 f8 80 5a d0 01  c9 fa 4a 5b 51 cb 05 52
00000060  4a 5b 51 cb 06 52 60 5b  51 cb 07 52 83 7c ca 0a
00000070  c8 30 c9 e9 4a 0b 00 5a  c9 ec 4a 00 5a c9 ff 4a
00000080  00 5a c9 ea 4a 17 80 b3  d0 ff 0e 80 9e 5b 36 80
00000090  a3 35 80 98 cf ff 80 ac  07 27 a6 20 80 ac cf 7f
000000a0  5b 80 ac 30 80 aa cf ff  80 ac 06 27 5a 51 cb 04
000000b0  52 81 1b c9 eb 4a 0f 80  bd 01 5a 81 1b c9 e5 4a
000000c0  15 81 1b 14 80 da a4 5a  c9 df 4a 5b 51 cb 01 52
000000d0  c9 ea ca be c9 eb ca 02  81 1b 16 80 e5 a6 5a c9
000000e0  eb ca 07 80 e8 23 a5 5a  c9 e5 4a 5b 51 cb 02 52
000000f0  81 1b 5f 5f 5f 5f 5f 5f  5f 5f 5f 5f 5f 5f 5f 7e
00000100  48 3f 23 5f 7d 49 c8 b0  c9 ed 5a 4b 5a 60 5a 61
00000110  5a 62 5a 63 5a 4a cb 00  30 54 5f 40 30 87 b5 32
00000120  84 5e 33 80 28 31 81 45  c8 70 c9 f3 4a 30 5b 54
00000130  4a 73 4a 72 4a 71 4a 70  4a 5b 4a 59 6d b5 33 b4
00000140  a3 58 6e 5c b1 c8 30 c9  43 4a 71 28 50 40 11 81
```

```
RJ[Q
R`[Q
_____________~H?#_}I
ZKZ`ZaZbZcZJ
0T_@0
J0[TJsJrJqJpJ[JYm
CJq(P@
PPPPC
sPPPPPPPPC
ua q()
Pc%s
BJ>&
ZYJra
AZYZ
BJ'Z
AZYbZ
Ba#qc
[SPPPPb[SPPPPPPPP)
qJpJrF
cZa!q`r
QK$[
Qa #qV
aZ`ZbZ
Rb"r
['sQ
____J[`
```

Altarian
Standard
Code for
Information
Interchange




Call
Switch/Case
Call
Call
Jumps
Return

Bytes right before ”obviously non-code”

```
2427  80 71 80 95 5f*985
00f2  02 52 81 1b 5f*13          Unconditional
1f43  5a 2f 81 8c 5f*189         Jumps ?

07c7  5b 5e 5c b8 5f*43
0f32  0b 87 24 b8 5f*78          Unconditional
1728  d0 31 4a b8 5f*216         Returns ?
2d98  b4 b8 b5 b8 5f*544
36a0  e6 6b 5a b8 5f*352
3e44  0b 86 3e b8 5f*404
```

```
cpu.it.load_string('''
RET     -       |1 0 1 1|1 0 0 0|                       |
J80     bx      |1 0 0 0|0 0 0 0| bx                    |
J81     bx      |1 0 0 0|0 0 0 1| bx                    |
''')
```

Disassembler for strange instruction sets:

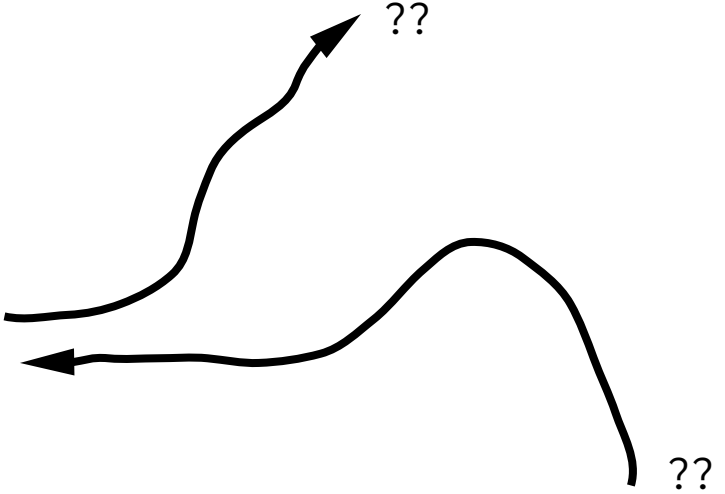
https://github.com/bsdphk/PyReveng3

For data-archeology on old/rare instruction sets

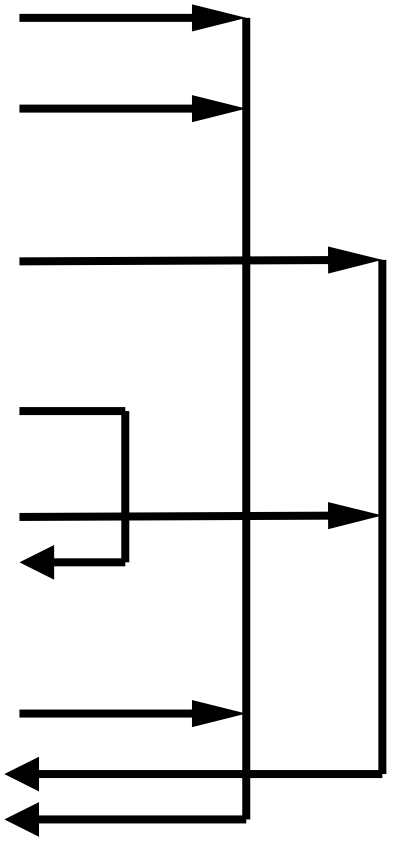
```
000008c7  cd 4a 11    |
000008ca  80 f3       |          J80     #0xf3
000008cc  32          |          .XXX
000008cd  80 f3       |          J80     #0xf3
000008cf  c8 11 c9 cf |          .XXX
000008d3  4a d0 f3 0a |
000008d7  80 ef       |          J80     #0xef
000008d9  c8 21 c9 cf |          .XXX
000008dd  4a 34       |
000008df  80 e4       |          J80     #0xe4
000008e1  15          |          .XXX
000008e2  80 ef       |          J80     #0xef
000008e4  c8 11 c9 cd |          .XXX
000008e8  af 4a a1 5a |
000008ec  2f          |
000008ed  80 f3       |          J80     #0xf3
000008ef  dc 20 89 7e |          .XXX
000008f3  69 01 79 0b |
000008f7  81 41       |          J81     #0x41
000008f9  d9 a8 89 f4 |          .XXX
000008fd  89 b6 c8 11 |
00000901  c9 d9 4a 71 |
00000905  ca 00 44 32 |
```

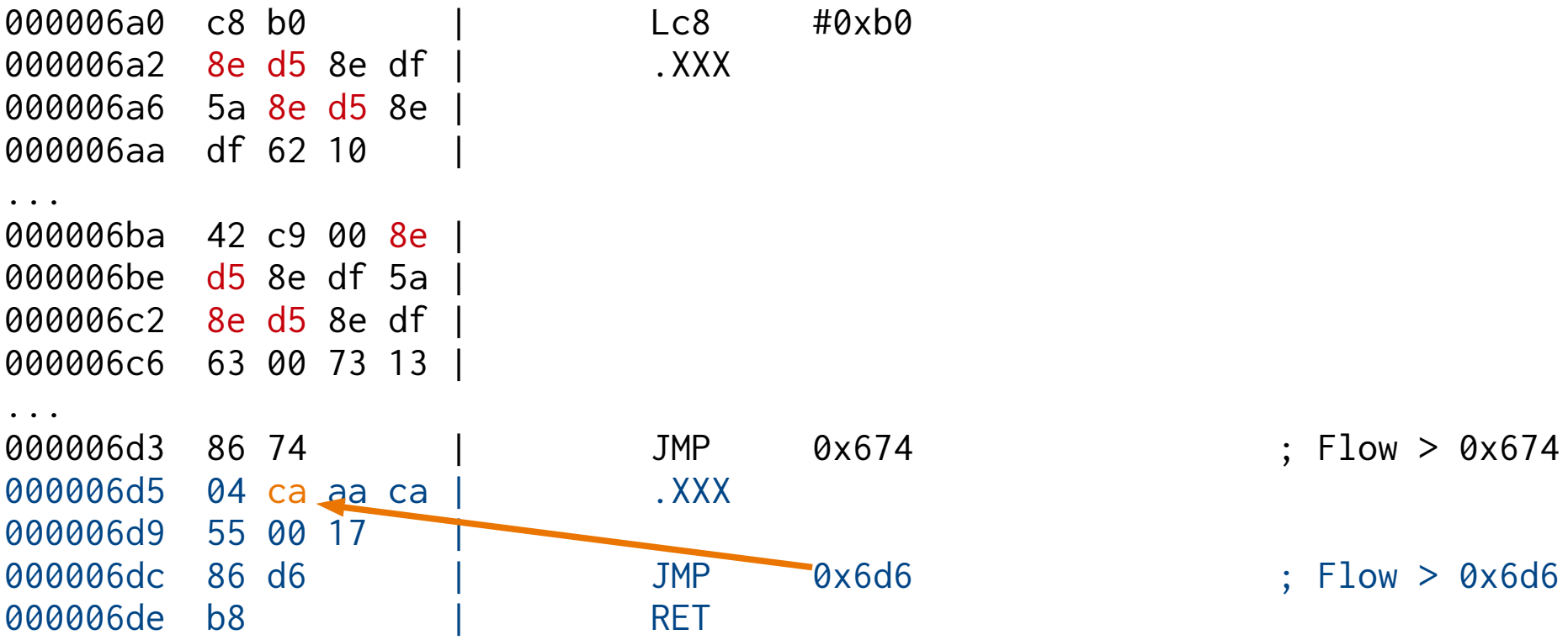
```
000008c7  cd 4a 11    |
000008ca  80 f3       |          J80     #0xf3
000008cc  32          |          .XXX                ← Single byte instruction
000008cd  80 f3       |          J80     #0xf3
000008cf  c8 11 c9 cf |          .XXX
000008d3  4a d0 f3 0a |
000008d7  80 ef       |          J80     #0xef
000008d9  c8 21 c9 cf |          .XXX
000008dd  4a 34       |
000008df  80 e4       |          J80     #0xe4
000008e1  15          |          .XXX                ← Single byte instruction
000008e2  80 ef       |          J80     #0xef
000008e4  c8 11 c9 cd |          .XXX
000008e8  af 4a a1 5a |
000008ec  2f          |
000008ed  80 f3       |          J80     #0xf3
000008ef  dc 20 89 7e |          .XXX
000008f3  69 01 79 0b |
000008f7  81 41       |          J81     #0x41
000008f9  d9 a8 89 f4 |          .XXX
000008fd  89 b6 c8 11 |
00000901  c9 d9 4a 71 |
00000905  ca 00 44 32 |
```

```
000008c7  cd 4a 11     |
000008ca  80 f3        |          J80     #0xf3
000008cc  32           |          .XXX
000008cd  80 f3        |          J80     #0xf3
000008cf  c8 11 c9 cf  |          .XXX
000008d3  4a d0 f3 0a  |
000008d7  80 ef        |          J80     #0xef
000008d9  c8 21 c9 cf  |          .XXX
000008dd  4a 34        |
000008df  80 e4        |          J80     #0xe4
000008e1  15           |          .XXX
000008e2  80 ef        |          J80     #0xef
000008e4  c8 11 c9 cd  |          .XXX
000008e8  af 4a a1 5a  |
000008ec  2f           |
000008ed  80 f3        |          J80     #0xf3
000008ef  dc 20 89 7e  |          .XXX
000008f3  69 01 79 0b  |
000008f7  81 41        |          J81     #0x41
000008f9  d9 a8 89 f4  |          .XXX
000008fd  89 b6 c8 11  |
00000901  c9 d9 4a 71  |
00000905  ca 00 44 32  |
```



```
000008c7  cd 4a 11    |
000008ca  80 f3       |          J80     #0xf3
000008cc  32          |          .XXX
000008cd  80 f3       |          J80     #0xf3
000008cf  c8 11 c9 cf |          .XXX
000008d3  4a d0 f3 0a |
000008d7  80 ef       |          J80     #0xef
000008d9  c8 21 c9 cf |          .XXX
000008dd  4a 34       |
000008df  80 e4       |          J80     #0xe4
000008e1  15          |          .XXX
000008e2  80 ef       |          J80     #0xef
000008e4  c8 11 c9 cd |          .XXX
000008e8  af 4a a1 5a |
000008ec  2f          |
000008ed  80 f3       |          J80     #0xf3
000008ef  dc 20 89 7e |          .XXX
000008f3  69 01 79 0b |
000008f7  81 41       |          J81     #0x41
000008f9  d9 a8 89 f4 |          .XXX
000008fd  89 b6 c8 11 |
00000901  c9 d9 4a 71 |
00000905  ca 00 44 32 |
```




000006a0 c8 b0 | Lc8 #0xb0
000006a2 8e d5 8e df | .XXX
000006a6 5a 8e d5 8e |
000006aa df 62 10 |
...
000006ba 42 c9 00 8e |
000006be d5 8e df 5a |
000006c2 8e d5 8e df |
000006c6 63 00 73 13 |
...
000006d3 86 74 | JMP 0x674 ; Flow > 0x674
000006d5 04 ca aa ca | .XXX
000006d9 55 00 17 |
000006dc 86 d6 | JMP 0x6d6 ; Flow > 0x6d6
000006de b8 | RET

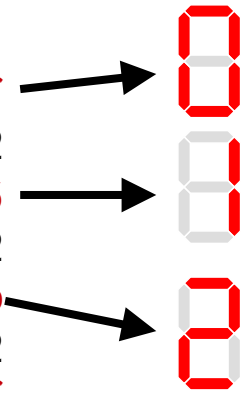
```
cpu.it.load_string('''
    JSB     adrl,>C         |1 0 0 0 1| ahi | alo           |
''')

000006a0  c8 b0       |         Lc8     #0xb0
000006a2  8e d5       |         JSB     0x6d5                   ; Flow C 0x6d5
000006a4  8e df       |         JSB     0x6df                   ; Flow C 0x6df
000006a6  5a          |         .XXX
000006a7  8e d5       |         JSB     0x6d5                   ; Flow C 0x6d5
000006a9  8e df       |         JSB     0x6df                   ; Flow C 0x6df
000006ab  62 10       |         .XXX
000006ad  86 b4       |         JMP     0x6b4                   ; Flow > 0x6b4
000006af  31          |         .XXX
000006b0  86 74       |         JMP     0x674                   ; Flow > 0x674
000006b2  86 b0       |         JMP     0x6b0                   ; Flow > 0x6b0
000006b4  c8 a0       |         Lc8     #0xa0
000006b6  d3 04 63 5b |         .XXX
000006ba  42 c9 00    |
000006bd  8e d5       |         JSB     0x6d5                   ; Flow C 0x6d5
000006bf  8e df       |         JSB     0x6df                   ; Flow C 0x6df
000006c1  5a          |         .XXX
000006c2  8e d5       |         JSB     0x6d5                   ; Flow C 0x6d5
000006c4  8e df       |         JSB     0x6df                   ; Flow C 0x6df
```

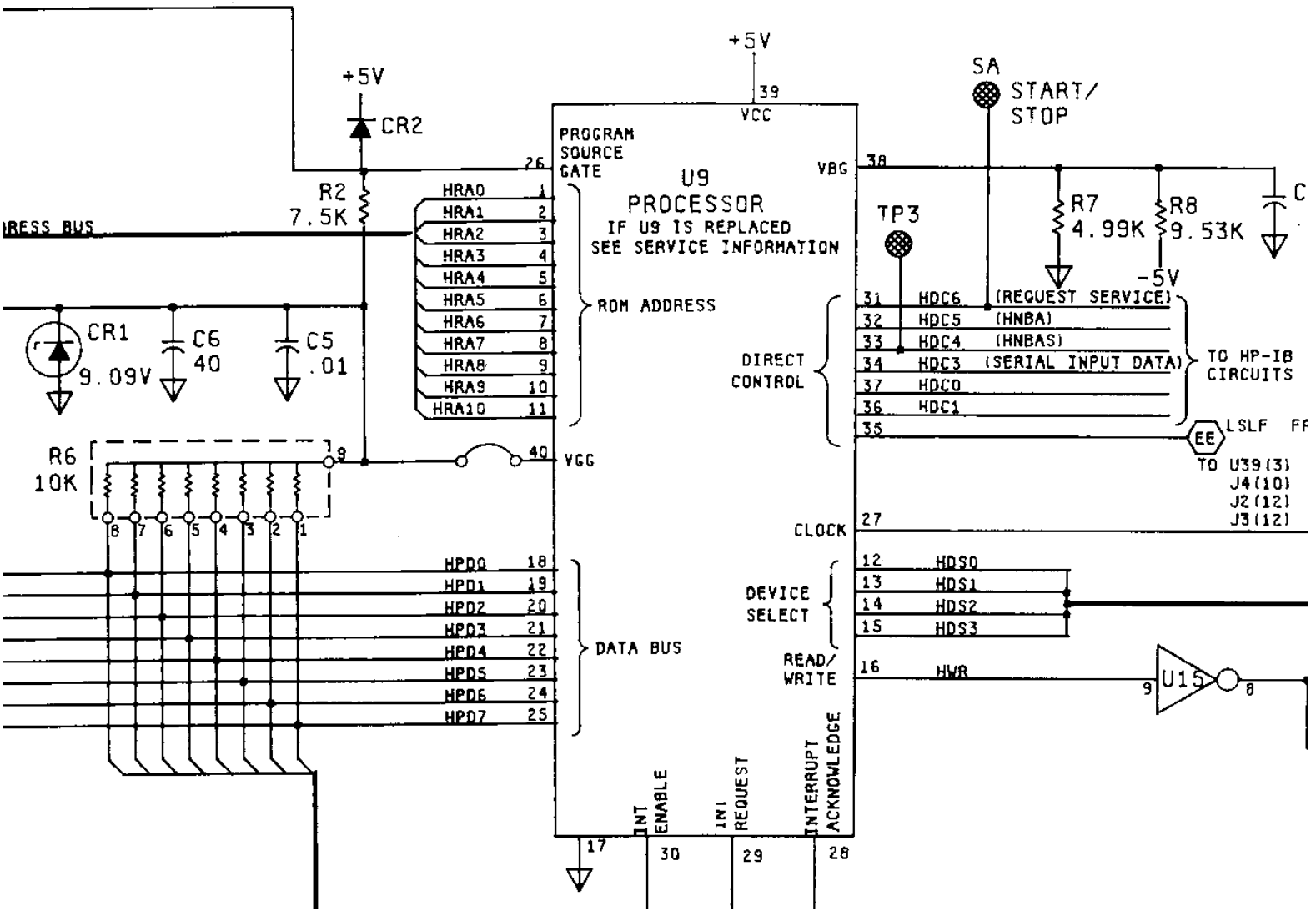
```
cpu.it.load_string('''
    Lca    bx    |1 1 0 0 1 0 1 0| bx            |
''')

00000f82  85 02       |          JMP     0xd02                    ; Flow > 0xd02
00000f84  ca 06       |          .XXX
00000f86  85 02       |          JMP     0xd02                    ; Flow > 0xd02
00000f88  ca 5b       |          .XXX
00000f8a  85 02       |          JMP     0xd02                    ; Flow > 0xd02
00000f8c  ca 4f       |          .XXX
00000f8e  85 02       |          JMP     0xd02                    ; Flow > 0xd02
00000f90  ca 66       |          .XXX
00000f92  85 02       |          JMP     0xd02                    ; Flow > 0xd02
00000f94  ca 6d       |          .XXX
00000f96  85 02       |          JMP     0xd02                    ; Flow > 0xd02
00000f98  ca 7d       |          .XXX
00000f9a  85 02       |          JMP     0xd02                    ; Flow > 0xd02
00000f9c  ca 07       |          .XXX
00000f9e  85 02       |          JMP     0xd02                    ; Flow > 0xd02
00000fa0  ca 7f       |          .XXX
00000fa2  85 02       |          JMP     0xd02                    ; Flow > 0xd02
00000fa4  ca 6f       |          .XXX
00000fa6  85 02       |          JMP     0xd02                    ; Flow > 0xd02
```

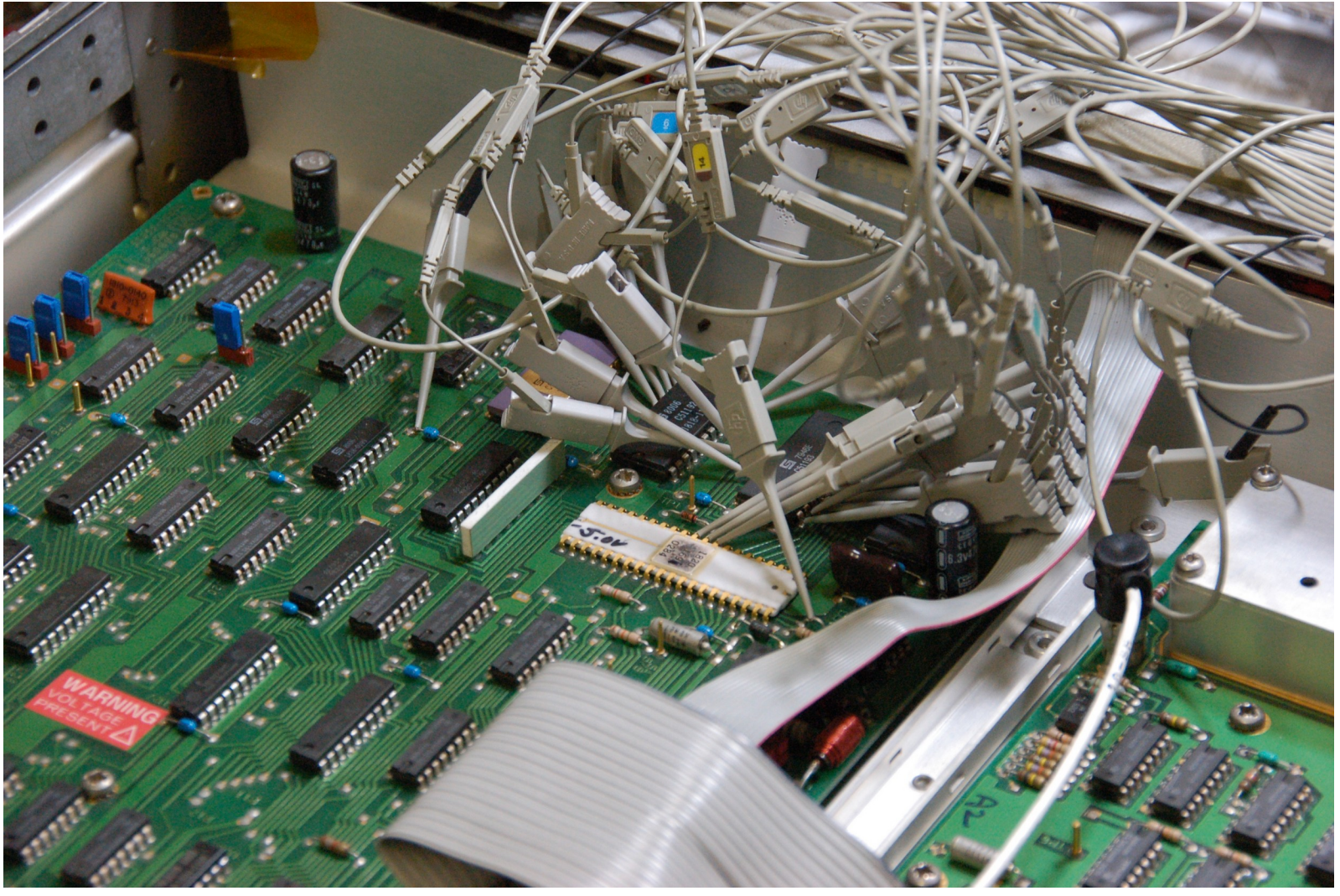
```
00001909  80 d8       |        JMP     0x18d8                  ; Flow > 0x18d8
0000190b  c9          |        Lc9
0000190c  70          |        .XXX
0000190d  81 1f       |        JMP     0x191f                  ; Flow > 0x191f
0000190f  c9          |        Lc9
00001910  80 81       |        JMP     0x1881                  ; Flow > 0x1881
00001912  1f d1 10    |        .XXX
00001915  81 19       |        JMP     0x1919                  ; Flow > 0x1919
00001917  d1 08       |        .XXX
00001919  c9          |        Lc9
0000191a  60          |        .XXX
0000191b  81 1f       |        JMP     0x191f                  ; Flow > 0x191f
0000191d  c9          |        Lc9
0000191e  50 d1 10 cb |        .XXX
----------------------+
00001909  80 d8       |        JMP     0x18d8                  ; Flow > 0x18d8
0000190b  c9 70       |        Lc9     #0x70
0000190d  81 1f       |        JMP     0x191f                  ; Flow > 0x191f
0000190f  c9 80       |        Lc9     #0x80
00001911  81 1f       |        JMP     0x191f                  ; Flow > 0x191f
00001913  d1 10       |        .XXX
00001915  81 19       |        JMP     0x1919                  ; Flow > 0x1919
00001917  d1 08       |        .XXX
00001919  c9 60       |        Lc9     #0x60
0000191b  81 1f       |        JMP     0x191f                  ; Flow > 0x191f
0000191d  c9 50       |        Lc9     #0x50
```

```
00000f80  ca 3f       |          Lca     #0x3f
00000f82  85 02       |          JMP     0xd02                    ; Flow > 0xd02
00000f84  ca 06       |          Lca     #0x06
00000f86  85 02       |          JMP     0xd02                    ; Flow > 0xd02
00000f88  ca 5b       |          Lca     #0x5b
00000f8a  85 02       |          JMP     0xd02                    ; Flow > 0xd02
00000f8c  ca 4f       |          Lca     #0x4f
00000f8e  85 02       |          JMP     0xd02                    ; Flow > 0xd02
00000f90  ca 66       |          Lca     #0x66
00000f92  85 02       |          JMP     0xd02                    ; Flow > 0xd02
00000f94  ca 6d       |          Lca     #0x6d
00000f96  85 02       |          JMP     0xd02                    ; Flow > 0xd02
00000f98  ca 7d       |          Lca     #0x7d
00000f9a  85 02       |          JMP     0xd02                    ; Flow > 0xd02
00000f9c  ca 07       |          Lca     #0x07
00000f9e  85 02       |          JMP     0xd02                    ; Flow > 0xd02
00000fa0  ca 7f       |          Lca     #0x7f
00000fa2  85 02       |          JMP     0xd02                    ; Flow > 0xd02
00000fa4  ca 6f       |          Lca     #0x6f
00000fa6  85 02       |          JMP     0xd02                    ; Flow > 0xd02
```

```
00000f80  ca 3f     |        Lca    #0x3f
00000f82  85 02     |        JMP    0xd02          ; Flow > 0xd02
00000f84  ca 06     |        Lca    #0x06
00000f86  85 02     |        JMP    0xd02          ; Flow > 0xd02
00000f88  ca 5b     |        Lca    #0x5b
00000f8a  85 02     |        JMP    0xd02          ; Flow > 0xd02
00000f8c  ca 4f     |        Lca    #0x4f
00000f8e  85 02     |        JMP    0xd02          ; Flow > 0xd02
00000f90  ca 66     |        Lca    #0x66
00000f92  85 02     |        JMP    0xd02          ; Flow > 0xd02
00000f94  ca 6d     |        Lca    #0x6d
00000f96  85 02     |        JMP    0xd02          ; Flow > 0xd02
00000f98  ca 7d     |        Lca    #0x7d
00000f9a  85 02     |        JMP    0xd02          ; Flow > 0xd02
00000f9c  ca 07     |        Lca    #0x07
00000f9e  85 02     |        JMP    0xd02          ; Flow > 0xd02
00000fa0  ca 7f     |        Lca    #0x7f
00000fa2  85 02     |        JMP    0xd02          ; Flow > 0xd02
00000fa4  ca 6f     |        Lca    #0x6f
00000fa6  85 02     |        JMP    0xd02          ; Flow > 0xd02
```




+5V
CR2
R2
7.5K
RESS BUS
CR1
9.09V
C6
40
C5
.01
R6
10K
+5V
39
VCC
PROGRAM
SOURCE
GATE
26
U9
PROCESSOR
IF U9 IS REPLACED
SEE SERVICE INFORMATION
HRA0 1
HRA1 2
HRA2 3
HRA3 4
HRA4 5
HRA5 6
HRA6 7
HRA7 8
HRA8 9
HRA9 10
HRA10 11
ROM ADDRESS
9
40
VGG
8 7 6 5 4 3 2 1
HPD0 18
HPD1 19
HPD2 20
HPD3 21
HPD4 22
HPD5 23
HPD6 24
HPD7 25
DATA BUS
17
INT ENABLE 30
IN1 REQUEST 29
INTERRUPT ACKNOWLEDGE 28
VBG 38
SA
START/
STOP
TP3
R7
4.99K
R8
9.53K
-5V
C
31 HDC6 (REQUEST SERVICE)
32 HDC5 (HNBA)
33 HDC4 (HNBAS)
34 HDC3 (SERIAL INPUT DATA)
37 HDC0
36 HDC1
DIRECT
CONTROL
TO HP-IB
CIRCUITS
35
EE
LSLF FF
TO U39(3)
J4(10)
J2(12)
J3(12)
CLOCK 27
12 HDS0
13 HDS1
14 HDS2
15 HDS3
DEVICE
SELECT
READ/
WRITE
16 HWR
U15
9
8


WARNING
VOLTAGE
PRESENT

MAY 1980

HEWLETT-PACKARD JOURNAL

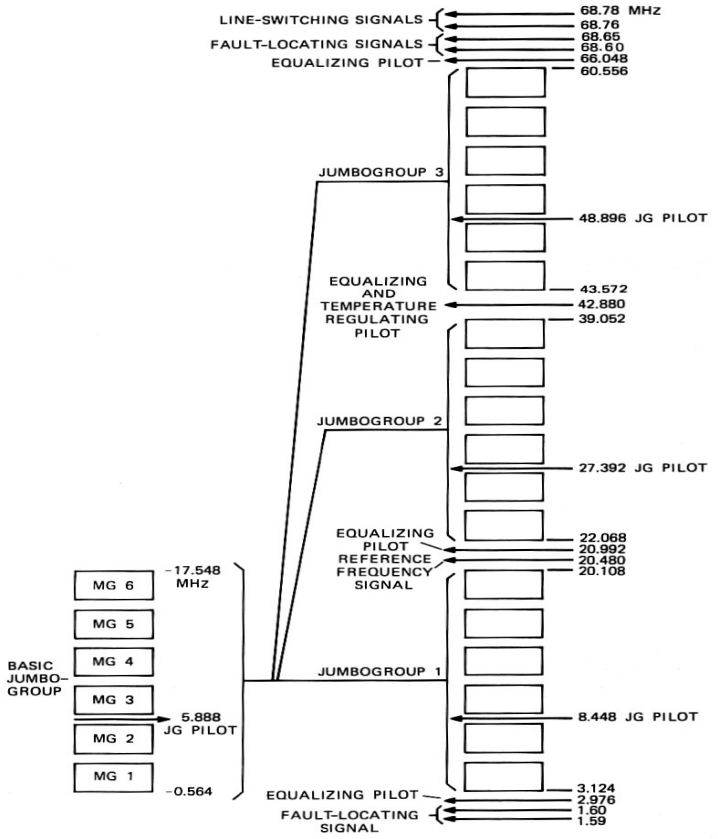


Fig. 1—Three-jumbogroup line frequency spectrum.

(JMX) frequency-division multiplex equipment[2] to form the 10,800-channel line signal.

In addition to the message band, Fig. 1 also shows the placement of

1 coax cable @ 1…69MHz

13200 phone circuits

4000 repeaters (1/mile)

4000 * 0.01dB = 40dB

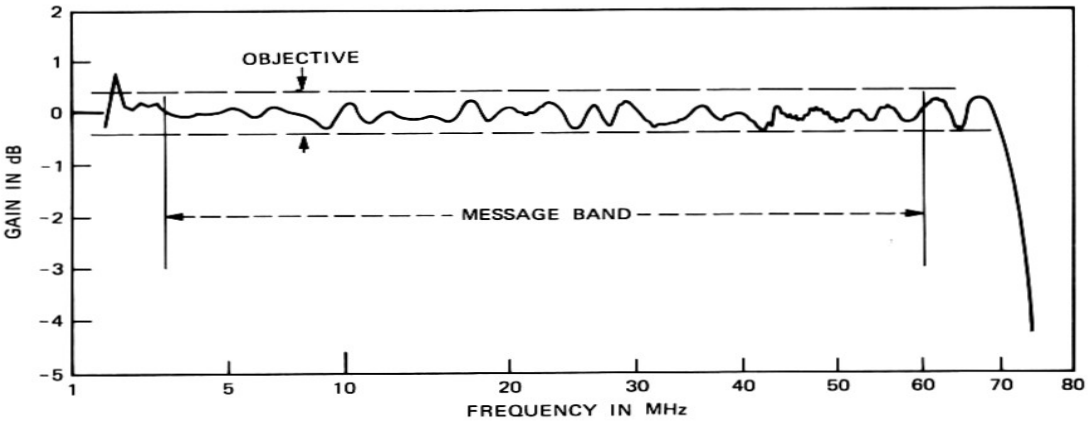
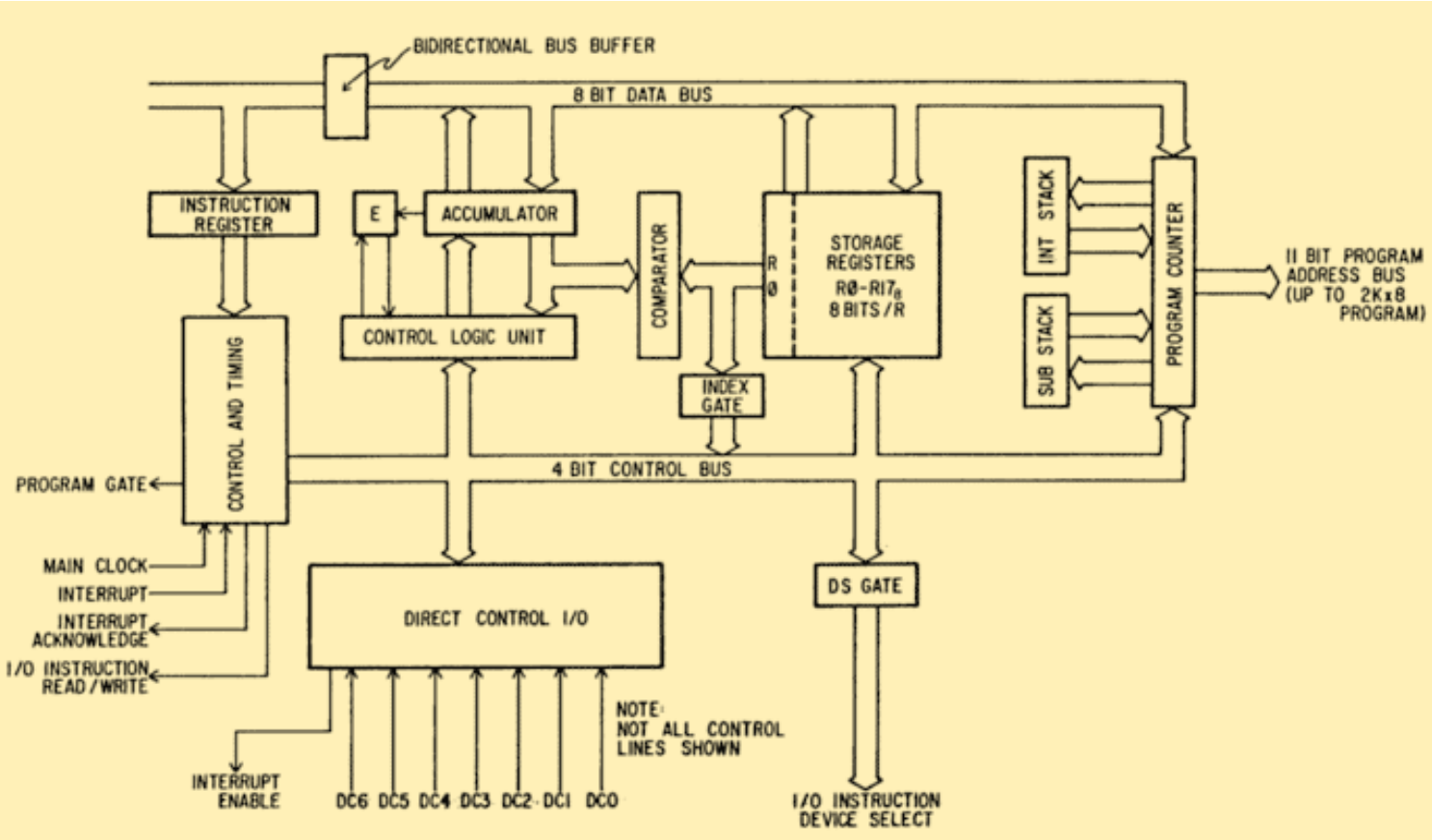
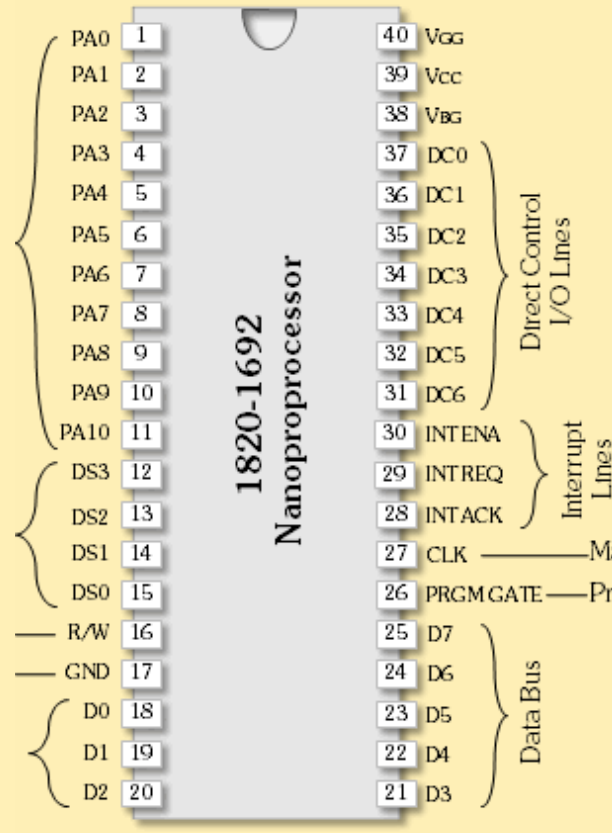


Fig. 35—Switching section equalized line response.

```
hp_nanoproc_instructions='''
INB     -               |0 0 0 0|0 0 0 0|             Increment A
DEB     -               |0 0 0 0|0 0 0 1|             Decrement A
IND     -               |0 0 0 0|0 0 1 0|             Increment BCD
DED     -               |0 0 0 0|0 0 1 1|             Decrement BCD
CLA     -               |0 0 0 0|0 1 0 0|             Clear A
CMA     -               |0 0 0 0|0 1 0 1|             Complement A
RSA     -               |0 0 0 0|0 1 1 0|             Right Shift A
LSA     -               |0 0 0 0|0 1 1 1|             Left Shift A
SGT     S,>JC           |0 0 0 0|1 0 0 0|             Skip Greater Than
SLT     S,>JC           |0 0 0 0|1 0 0 1|             Skip Less Than
SEQ     S,>JC           |0 0 0 0|1 0 1 0|             Skip Equal
SAZ     S,>JC           |0 0 0 0|1 0 1 1|             Skip A Zero
SLE     S,>JC           |0 0 0 0|1 1 0 0|             Skip Less or Equal
SGE     S,>JC           |0 0 0 0|1 1 0 1|             Skip Greater or Equal
SNE     S,>JC           |0 0 0 0|1 1 1 0|             Skip Not Equal
SAN     S,>JC           |0 0 0 0|1 1 1 1|             Skip A non-zero
SBS     S,bno,>JC       |0 0 0 1|0| bno |             Skip Bit Set
SFS     S,dctl,>JC      |0 0 0 1|1| dctl|
SES     S,>JC           |0 0 0 1 1 1 1 1|             Skip E Set
SBN     bno             |0 0 1 0|0| bno |             Set A.bit
STC     dctl            |0 0 1 0|1| dctl|
ENI     -               |0 0 1 0|1 1 1 1|             Enable Irq
SBZ     S,bno,>JC       |0 0 1 1|0| bno |             Skip Bit Zero
SFZ     S,dctl,>JC      |0 0 1 1|1| dctl|
SEZ     S,>JC           |0 0 1 1|1 1 1 1|             Skip E Zero
INA     dev             |0 1 0 0| dev   |
OTA     dev             |0 1 0 1| dev   |
NOP     -               |0 1 0 1|1 1 1 1|
LDA     reg             |0 1 1 0| reg   |
```

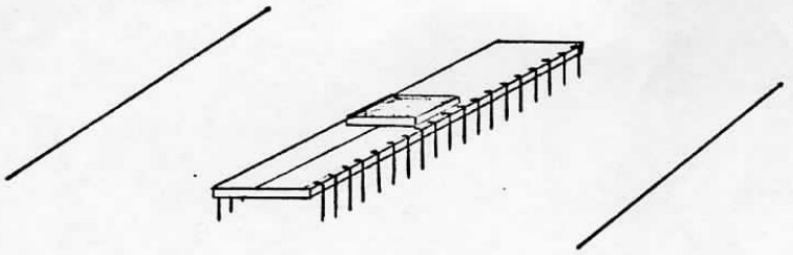

BIDIRECTIONAL BUS BUFFER
8 BIT DATA BUS
INSTRUCTION REGISTER
E
ACCUMULATOR
COMPARATOR
STORAGE REGISTERS
RØ-RI7₈
8 BITS/R
INT STACK
SUB STACK
PROGRAM COUNTER
11 BIT PROGRAM ADDRESS BUS (UP TO 2K x 8 PROGRAM)
CONTROL LOGIC UNIT
INDEX GATE
CONTROL AND TIMING
4 BIT CONTROL BUS
PROGRAM GATE
MAIN CLOCK
INTERRUPT
INTERRUPT ACKNOWLEDGE
I/O INSTRUCTION READ/WRITE
DIRECT CONTROL I/O
DS GATE
NOTE: NOT ALL CONTROL LINES SHOWN
INTERRUPT ENABLE
DC6 DC5 DC4 DC3 DC2 DC1 DC0
I/O INSTRUCTION DEVICE SELECT


1820-1692
Nanoprocessor
PA0 1
PA1 2
PA2 3
PA3 4
PA4 5
PA5 6
PA6 7
PA7 8
PA8 9
PA9 10
PA10 11
DS3 12
DS2 13
DS1 14
DS0 15
R/W 16
GND 17
D0 18
D1 19
D2 20
40 VGG
39 VCC
38 VBG
37 DC0
36 DC1
35 DC2
34 DC3
33 DC4
32 DC5
31 DC6
30 INTENA
29 INTREQ
28 INTACK
27 CLK
26 PRGM GATE
25 D7
24 D6
23 D5
22 D4
21 D3
Direct Control I/O Lines
Interrupt Lines
Data Bus

```
0003 *--------------------------------------------------------------
0004  *      HP 9411A SWITCH CONTROLLER MICROCODE
0005  *
0006  *      SOURCE       09411-18001
0007  *
0008  *      ABSOLUTE (FOR ALL 9411 FIRMWARE)     09411-14001
0009  *
0010  *      S. JOSEPH     5/2/77     REV. A
0011  *      S. JOSEPH     5/20/78    REV. B
0012  *
0013  *      NANOPROCESSOR MICROCODE IS THE PROPRIETARY
0014  *      MATERIAL OF THE HEWLETT-PACKARD COMPANY.  USE AND
0015  *      DISCLOSURE THEREOF IS COMPLETELY RESTRICTED!!!
0016  *
0017  *      (C) COPYRIGHT HEWLETT-PACKARD COMPANY 1976.
0018  *      ALL RIGHTS RESERVED.
0019  *
0020  *--------------------------------------------------------------
0021  *
0022  * MEMORY BLOCK 0
0023  *
0024  * LISTENER/TALKER IDLE STATE (LIDS/TIDS)
0025  *
0026  0-000  0000  137  START NOP          POWER UP SEQUENCE
0027  0-001  0001  137        NOP
0028  0-002  0002  004        CLA          INDICATE UNADDRESSED
0029  0-003  0003  175        STA LTREG
0030  0-004  0004  056        STC ATN      SETUP TO WATCH ATN FLAG
0031  0-005  0005  055        STC LED      TURN OFF ADDRESSED LIGHT
0032  0-006  0006  004  IDLE  CLA          RESET HPIB OUTPUT
```

Preliminary Larry Bower
(for corrections)

LID
NANOPROCESSOR



Users
Guide

LID

Nano Processor

User's Guide

Drawing Number
A-5955-0331-1

ACM ”unprogramming challenge”

I used this as a challenge in my ACM Queue Column*

* ”The Bikeshed” - what did you expect :-)

ACM ”unprogramming challenge”

I used this as a challenge in my ACM Queue Column

A well known hacking-group solved it…

ACM ”unprogramming challenge”

I used this as a challenge in my ACM Queue Column

A well known hacking-group solved it

… in 48 hours