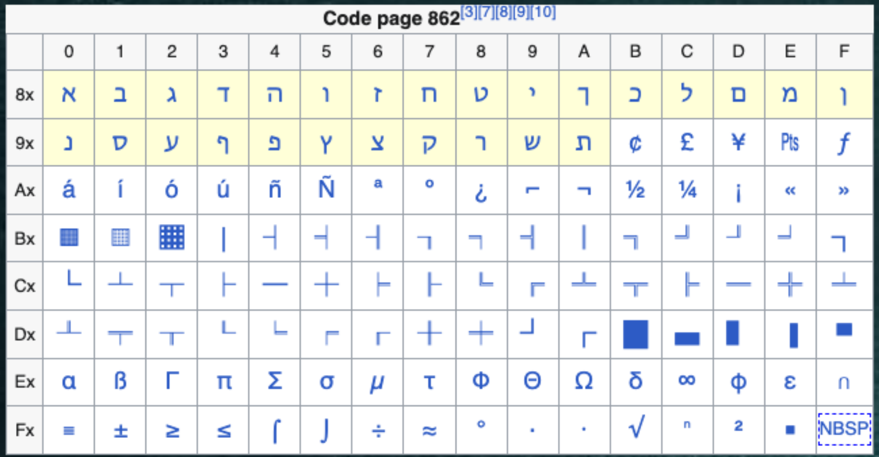
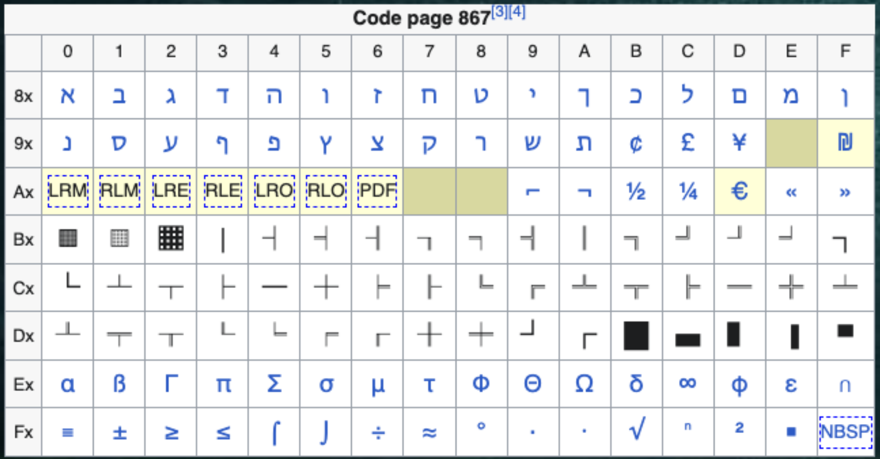
Decoding Hebrew

- Hebrew code pages of the era usually had the Hebrew alphabet start either at 0x80 or 0xE0.



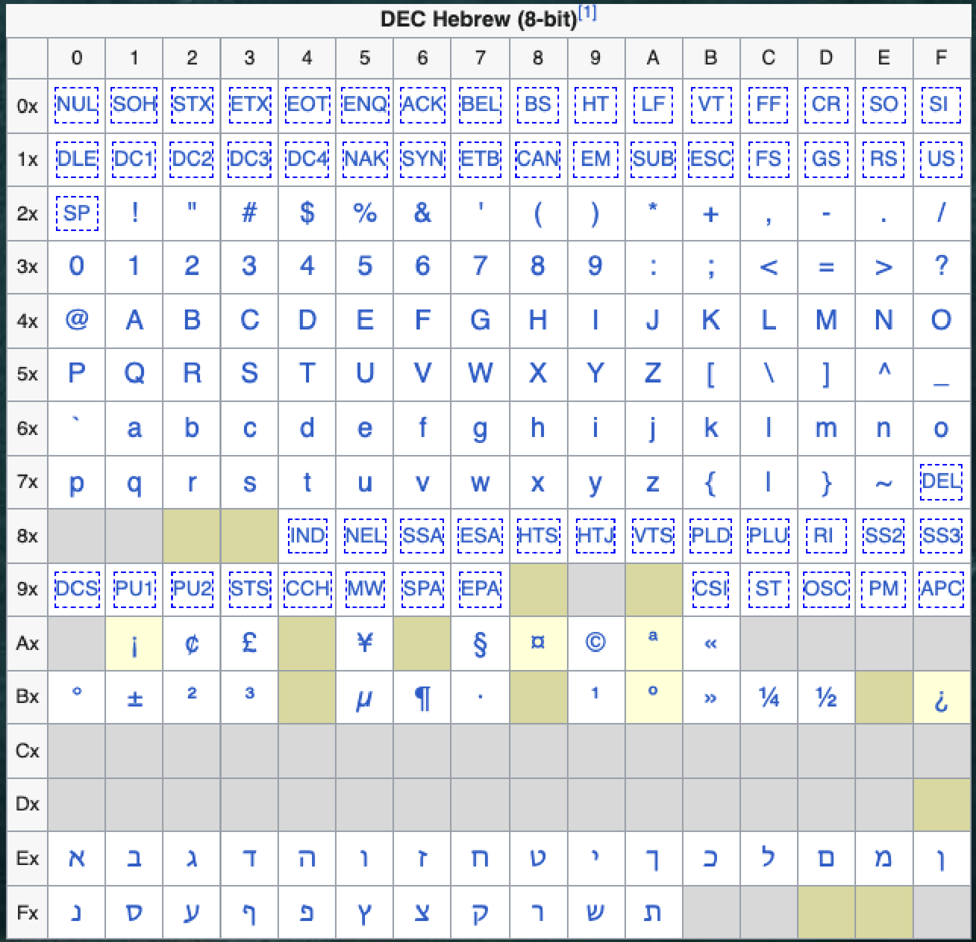
Code page 862[3][7][8][9][10]

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8x	א	ב	ג	ד	ה	ו	ז	ח	ט	י	ך	כ	ל	ם	מ	ן
9x	נ	ס	ע	ף	פ	ץ	צ	ק	ר	ש	ת	¢	£	¥	₧	ƒ
Ax	á	í	ó	ú	ñ	Ñ	ª	º	¿	⌐	¬	½	¼	¡	«	»
Bx	░	▒	▓	│	┤	╡	╢	╖	╕	╣	║	╗	╝	╜	╛	┐
Cx	└	┴	┬	├	─	┼	╞	╟	╚	╔	╩	╦	╠	═	╬	╧
Dx	╨	╤	╥	╙	╘	╒	╓	╫	╪	┘	┌	█	▄	▌	▐	▀
Ex	α	ß	Γ	π	Σ	σ	µ	τ	Φ	Θ	Ω	δ	∞	φ	ε	∩
Fx	≡	±	≥	≤	⌠	⌡	÷	≈	°	∙	·	√	ⁿ	²	■	NBSP



Code page 867[3][4]

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8x	א	ב	ג	ד	ה	ו	ז	ח	ט	י	ך	כ	ל	ם	מ	ן
9x	נ	ס	ע	ף	פ	ץ	צ	ק	ר	ש	ת	¢	£	¥		₪
Ax	LRM	RLM	LRE	RLE	LRO	RLO	PDF			⌐	¬	½	¼	€	«	»
Bx	░	▒	▓	│	┤	╡	╢	╖	╕	╣	║	╗	╝	╜	╛	┐
Cx	└	┴	┬	├	─	┼	╞	╟	╚	╔	╩	╦	╠	═	╬	╧
Dx	╨	╤	╥	╙	╘	╒	╓	╫	╪	┘	┌	█	▄	▌	▐	▀
Ex	α	ß	Γ	π	Σ	σ	µ	τ	Φ	Θ	Ω	δ	∞	φ	ε	∩
Fx	≡	±	≥	≤	⌠	⌡	÷	≈	°	∙	·	√	ⁿ	²	■	NBSP

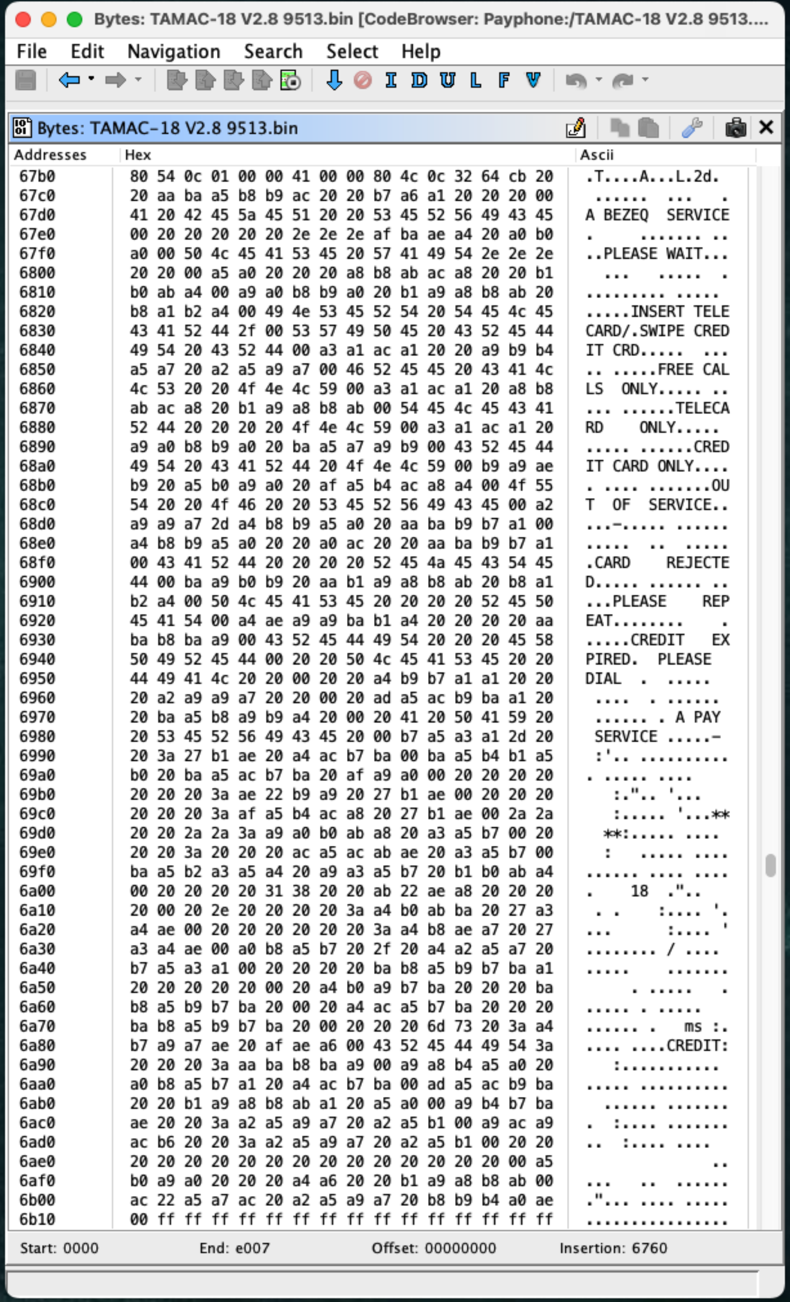


DEC Hebrew (8-bit)[1]

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1x	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
8x					IND	NEL	SSA	ESA	HTS	HTJ	VTS	PLD	PLU	RI	SS2	SS3
9x	DCS	PU1	PU2	STS	CCH	MW	SPA	EPA				CSI	ST	OSC	PM	APC
Ax		¡	¢	£		¥		§	¤	©	ª	«				
Bx	°	±	²	³		µ	¶	·		¹	º	»	¼	½		¿
Cx																
Dx																
Ex	א	ב	ג	ד	ה	ו	ז	ח	ט	י	ך	כ	ל	ם	מ	ן
Fx	נ	ס	ע	ף	פ	ץ	צ	ק	ר	ש	ת					

Decoding Hebrew

- But not in our case.
- It seems the Hebrew characters start at 0xA0.



Bytes: TAMAC-18 V2.8 9513.bin [CodeBrowser: Payphone:/TAMAC-18 V2.8 9513....

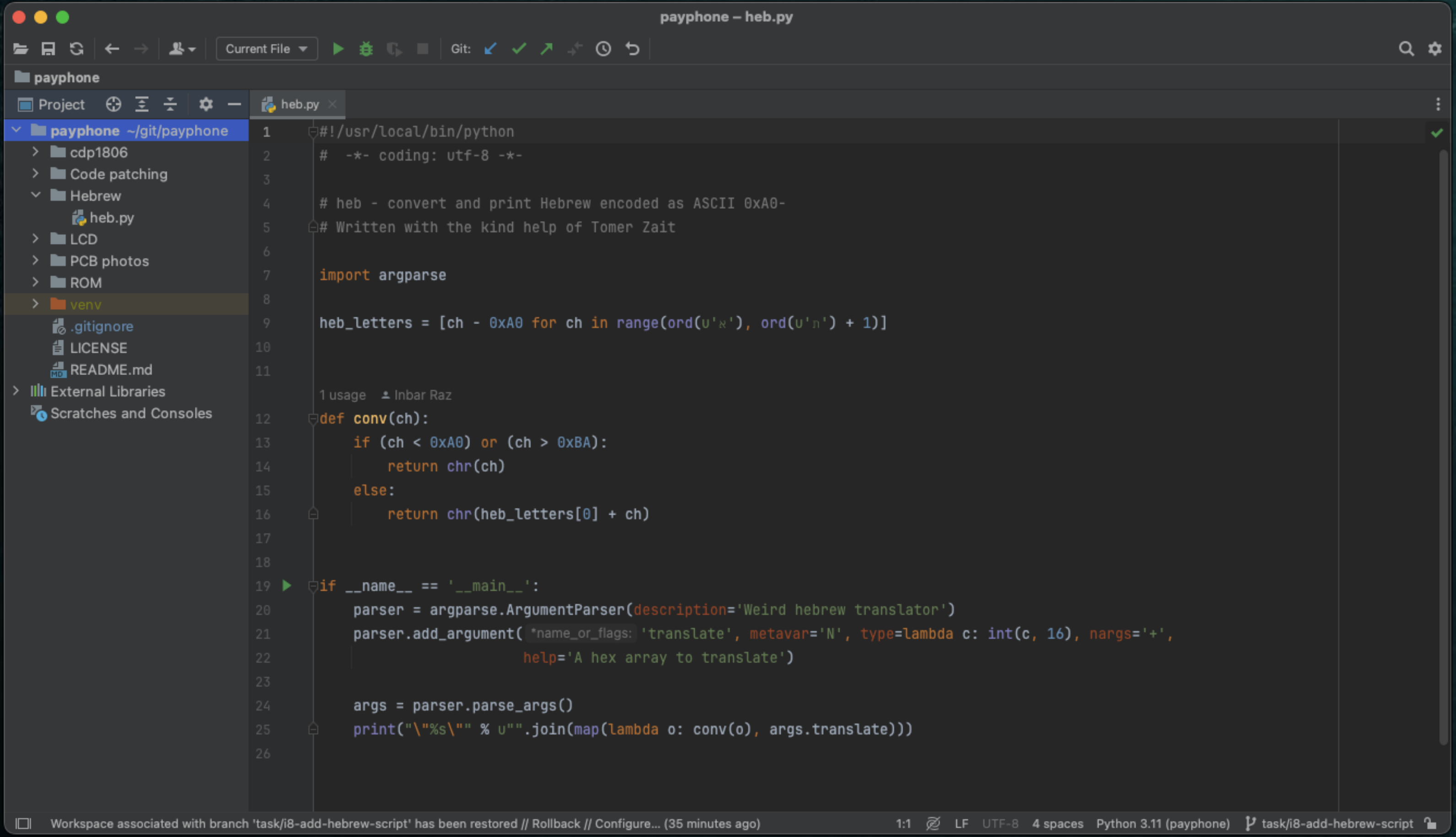
Bytes: TAMAC-18 V2.8 9513.bin

```
Addresses   Hex                                               Ascii
67b0        80 54 0c 01 00 00 41 00 00 80 4c 0c 32 64 cb 20   .T....A...L.2d. 
67c0        20 aa ba a5 b8 b9 ac 20 20 b7 a6 a1 20 20 20 00    ......  ...   .
67d0        41 20 42 45 5a 45 51 20 20 53 45 52 56 49 43 45   A BEZEQ  SERVICE
67e0        00 20 20 20 20 20 2e 2e 2e af ba ae a4 20 a0 b0   .     ....... ..
67f0        a0 00 50 4c 45 41 53 45 20 57 41 49 54 2e 2e 2e   ..PLEASE WAIT...
6800        20 20 00 a5 a0 20 20 20 a8 b8 ab ac a8 20 20 b1     ...   .....  .
6810        b0 ab a4 00 a9 a0 b8 b9 a0 20 b1 a9 a8 b8 ab 20   ......... ..... 
6820        b8 a1 b2 a4 00 49 4e 53 45 52 54 20 54 45 4c 45   .....INSERT TELE
6830        43 41 52 44 2f 00 53 57 49 50 45 20 43 52 45 44   CARD/.SWIPE CRED
6840        49 54 20 43 52 44 00 a3 a1 ac a1 20 20 a9 b9 b4   IT CRD.....  ...
6850        a5 a7 20 a2 a5 a9 a7 00 46 52 45 45 20 43 41 4c   .. .....FREE CAL
6860        4c 53 20 20 4f 4e 4c 59 00 a3 a1 ac a1 20 a8 b8   LS  ONLY..... ..
6870        ab ac a8 20 b1 a9 a8 b8 ab 00 54 45 4c 45 43 41   ... ......TELECA
6880        52 44 20 20 20 20 4f 4e 4c 59 00 a3 a1 ac a1 20   RD    ONLY..... 
6890        a9 a0 b8 b9 a0 20 ba a5 a7 a9 b9 00 43 52 45 44   ..... ......CRED
68a0        49 54 20 43 41 52 44 20 4f 4e 4c 59 00 b9 a9 ae   IT CARD ONLY....
68b0        b9 20 a5 b0 a9 a0 20 af a5 b4 ac a8 a4 00 4f 55   . .... .......OU
68c0        54 20 20 4f 46 20 20 53 45 52 56 49 43 45 00 a2   T  OF  SERVICE..
68d0        a9 a9 a7 2d a4 b8 b9 a5 a0 20 aa ba b9 b7 a1 00   ...-..... ......
68e0        a4 b8 b9 a5 a0 20 20 a0 ac 20 20 aa ba b9 b7 a1   .....  ..  .....
68f0        00 43 41 52 44 20 20 20 20 52 45 4a 45 43 54 45   .CARD    REJECTE
6900        44 00 ba a9 b0 b9 20 aa b1 a9 a8 b8 ab 20 b8 a1   D..... ...... ..
6910        b2 a4 00 50 4c 45 41 53 45 20 20 20 20 52 45 50   ...PLEASE    REP
6920        45 41 54 00 a4 ae a9 a9 ba b1 a4 20 20 20 20 aa   EAT........    .
6930        ba b8 ba a9 00 43 52 45 44 49 54 20 20 20 45 58   .....CREDIT   EX
6940        50 49 52 45 44 00 20 20 50 4c 45 41 53 45 20 20   PIRED.  PLEASE  
6950        44 49 41 4c 20 20 00 20 20 a4 b9 b7 a1 a1 20 20   DIAL  .  .....  
6960        20 a2 a9 a9 a7 20 20 00 20 ad a5 ac b9 ba a1 20    ....  . ...... 
6970        20 ba a5 b8 a9 b9 a4 20 00 20 41 20 50 41 59 20    ...... . A PAY 
6980        20 53 45 52 56 49 43 45 20 00 b7 a5 a3 a1 2d 20    SERVICE .....- 
6990        20 3a 27 b1 ae 20 a4 ac b7 ba 00 ba a5 b4 b1 a5    :'.. ..........
69a0        b0 20 ba a5 ac b7 ba 20 af a9 a0 00 20 20 20 20   . ..... ....    
69b0        20 20 20 3a ae 22 b9 a9 20 27 b1 ae 00 20 20 20      :.".. '...   
69c0        20 20 20 3a af a5 b4 ac a8 20 27 b1 ae 00 2a 2a      :..... '...**
69d0        20 20 2a 2a 3a a9 a0 b0 ab a8 20 a3 a5 b7 00 20     **:..... .... 
69e0        20 20 3a 20 20 20 ac a5 ac ab ae 20 a3 a5 b7 00     :   ..... ....
69f0        ba a5 b2 a3 a5 a4 20 a9 a3 a5 b7 20 b1 b0 ab a4   ...... .... ....
6a00        00 20 20 20 20 31 38 20 20 ab 22 ae a8 20 20 20   .    18  ."..   
6a10        20 00 20 2e 20 20 20 20 3a a4 b0 ab ba 20 27 a3    . .    :.... '.
6a20        a4 ae 00 20 20 20 20 20 20 3a a4 b8 ae a7 20 27   ...      :.... '
6a30        a3 a4 ae 00 a0 b8 a5 b7 20 2f 20 a4 a2 a5 a7 20   ........ / .... 
6a40        b7 a5 a3 a1 00 20 20 20 20 ba b8 a5 b9 b7 ba a1   .....    .......
6a50        20 20 20 20 20 00 20 a4 b0 a9 b7 ba 20 20 20 ba        . .....   .
6a60        b8 a5 b9 b7 ba 20 00 20 a4 ac a5 b7 ba 20 20 20   ..... . .....   
6a70        ba b8 a5 b9 b7 ba 20 00 20 20 20 6d 73 20 3a a4   ...... .   ms :.
6a80        b7 a9 a7 ae 20 af ae a6 00 43 52 45 44 49 54 3a   .... ....CREDIT:
6a90        20 20 20 3a aa ba b8 ba a9 00 a9 a8 b4 a5 a0 20      :.......... 
6aa0        a0 b8 a5 b7 a1 20 a4 ac b7 ba 00 ad a5 ac b9 ba   ..... ..........
6ab0        20 20 b1 a9 a8 b8 ab a1 20 a5 a0 00 a9 b4 b7 ba     ...... .......
6ac0        ae 20 20 3a a2 a5 a9 a7 20 a2 a5 b1 00 a9 ac a9   .  :.... .......
6ad0        ac b6 20 20 3a a2 a5 a9 a7 20 a2 a5 b1 00 20 20   ..  :.... ....  
6ae0        20 20 20 20 20 20 20 20 20 20 20 20 20 20 00 a5                 ..
6af0        b0 a9 a0 20 20 20 a4 a6 20 20 b1 a9 a8 b8 ab 00   ...   ..  ......
6b00        ac 22 a5 a7 ac 20 a2 a5 a9 a7 20 b8 b9 b4 a0 ae   ."... .... .....
6b10        00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff   ................
```

Start: 0000   End: e007   Offset: 00000000   Insertion: 6760

Decoding Hebrew

- Another dear fried comes to the rescue.



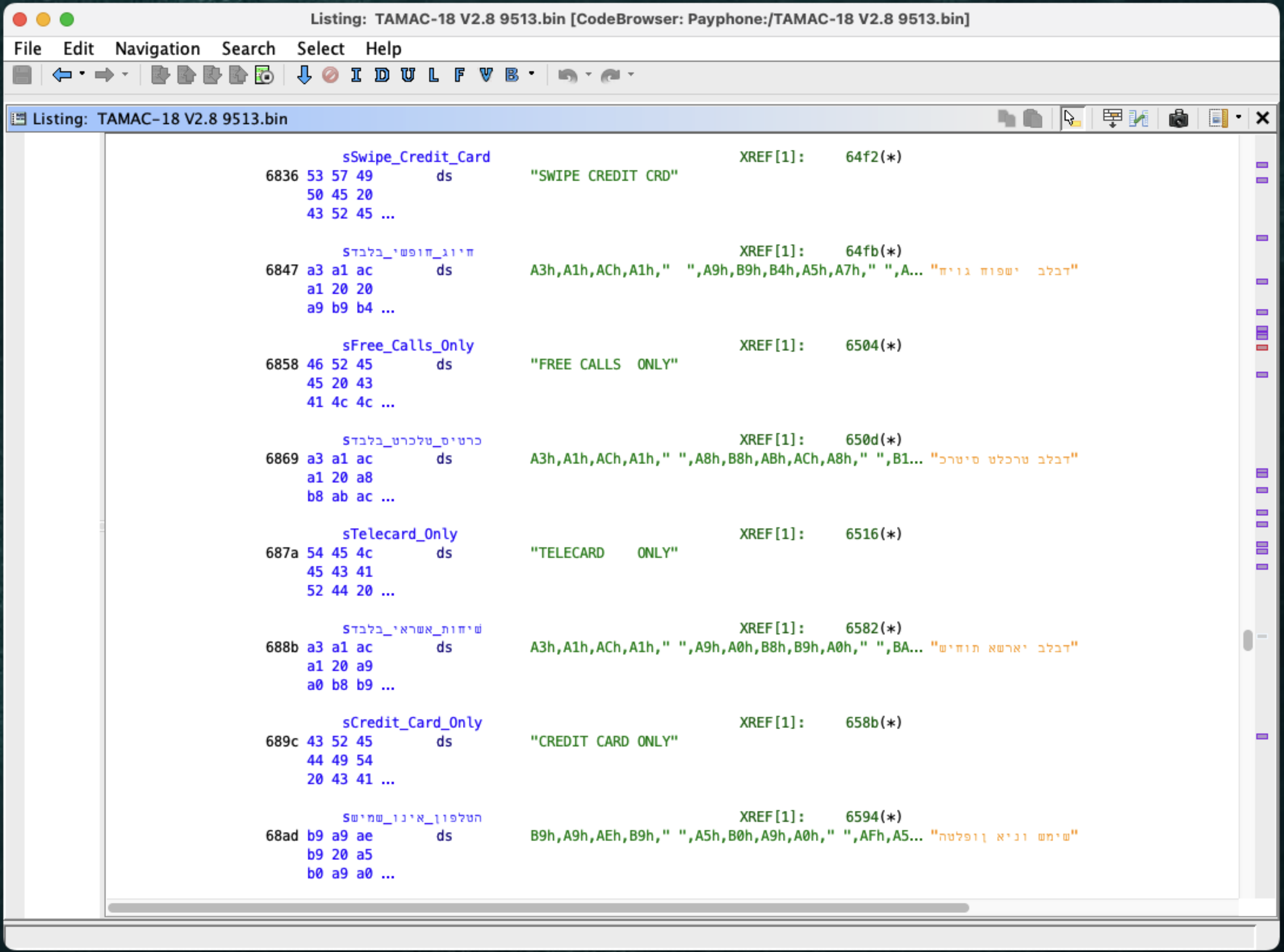
```python
#!/usr/local/bin/python
#  -*- coding: utf-8 -*-

# heb - convert and print Hebrew encoded as ASCII 0xA0-
# Written with the kind help of Tomer Zait

import argparse

heb_letters = [ch - 0xA0 for ch in range(ord(u'א'), ord(u'ת') + 1)]


1 usage  Inbar Raz
def conv(ch):
    if (ch < 0xA0) or (ch > 0xBA):
        return chr(ch)
    else:
        return chr(heb_letters[0] + ch)


if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Weird hebrew translator')
    parser.add_argument( *name_or_flags: 'translate', metavar='N', type=lambda c: int(c, 16), nargs='+',
                        help='A hex array to translate')

    args = parser.parse_args()
    print("\"%s\"" % u"".join(map(lambda o: conv(o), args.translate)))
```

Workspace associated with branch 'task/i8-add-hebrew-script' has been restored // Rollback // Configure... (35 minutes ago)

Tomer Zait

Decoding Hebrew

- Order is restored.



Listing: TAMAC-18 V2.8 9513.bin [CodeBrowser: Payphone:/TAMAC-18 V2.8 9513.bin]
File
Edit
Navigation
Search
Select
Help
Listing: TAMAC-18 V2.8 9513.bin

```
                 sSwipe_Credit_Card                           XREF[1]:     64f2(*)
6836 53 57 49        ds        "SWIPE CREDIT CRD"
     50 45 20
     43 52 45 ...

                 sחיוג_חופשי_בלבד                             XREF[1]:     64fb(*)
6847 a3 a1 ac        ds        A3h,A1h,ACh,A1h,"  ",A9h,B9h,B4h,A5h,A7h," ",A... "דבלב  ישפוח גויח"
     a1 20 20
     a9 b9 b4 ...

                 sFree_Calls_Only                             XREF[1]:     6504(*)
6858 46 52 45        ds        "FREE CALLS  ONLY"
     45 20 43
     41 4c 4c ...

                 sכרטיס_טלכרט_בלבד                            XREF[1]:     650d(*)
6869 a3 a1 ac        ds        A3h,A1h,ACh,A1h," ",A8h,B8h,ABh,ACh,A8h," ",B1... "דבלב טרכלט סיטרכ"
     a1 20 a8
     b8 ab ac ...

                 sTelecard_Only                               XREF[1]:     6516(*)
687a 54 45 4c        ds        "TELECARD    ONLY"
     45 43 41
     52 44 20 ...

                 sשיחות_אשראי_בלבד                            XREF[1]:     6582(*)
688b a3 a1 ac        ds        A3h,A1h,ACh,A1h," ",A9h,A0h,B8h,B9h,A0h," ",BA... "דבלב יארשא תוחיש"
     a1 20 a9
     a0 b8 b9 ...

                 sCredit_Card_Only                            XREF[1]:     658b(*)
689c 43 52 45        ds        "CREDIT CARD ONLY"
     44 49 54
     20 43 41 ...

                 sהטלפון_אינו_שמיש                            XREF[1]:     6594(*)
68ad b9 a9 ae        ds        B9h,A9h,AEh,B9h," ",A5h,B0h,A9h,A0h," ",AFh,A5... "שימש וניא ןופלטה"
     b9 20 a5
     b0 a9 a0 ...
```

TA0002: Execution

TA0011: Command & Control

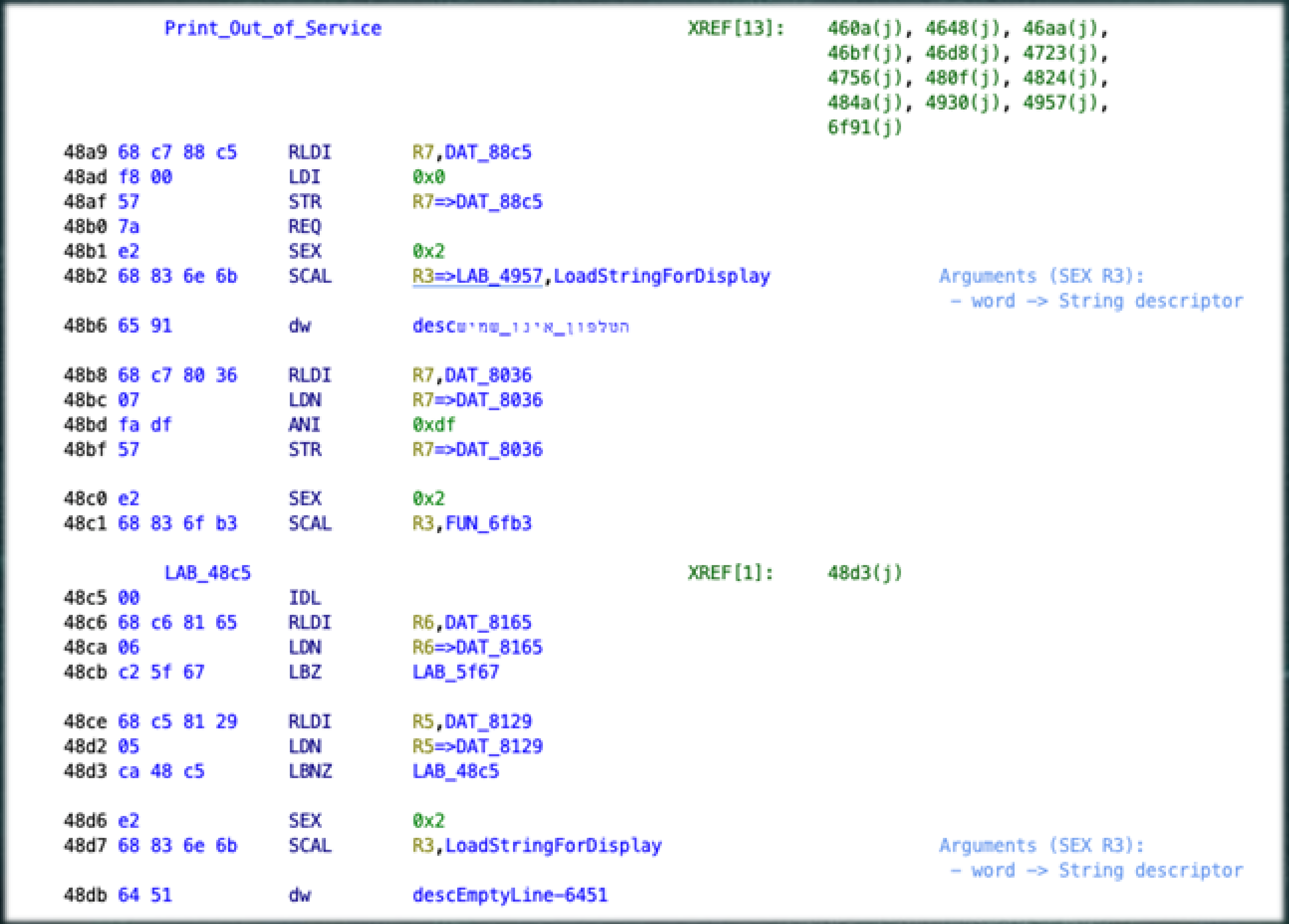
Booting the phone

- At some point, I decided to inject 5V into the backup battery terminal, and the phone woke up.
- Unfortunately, it was displaying an error message:

הטלפון אינו שמיש

Finding the error message

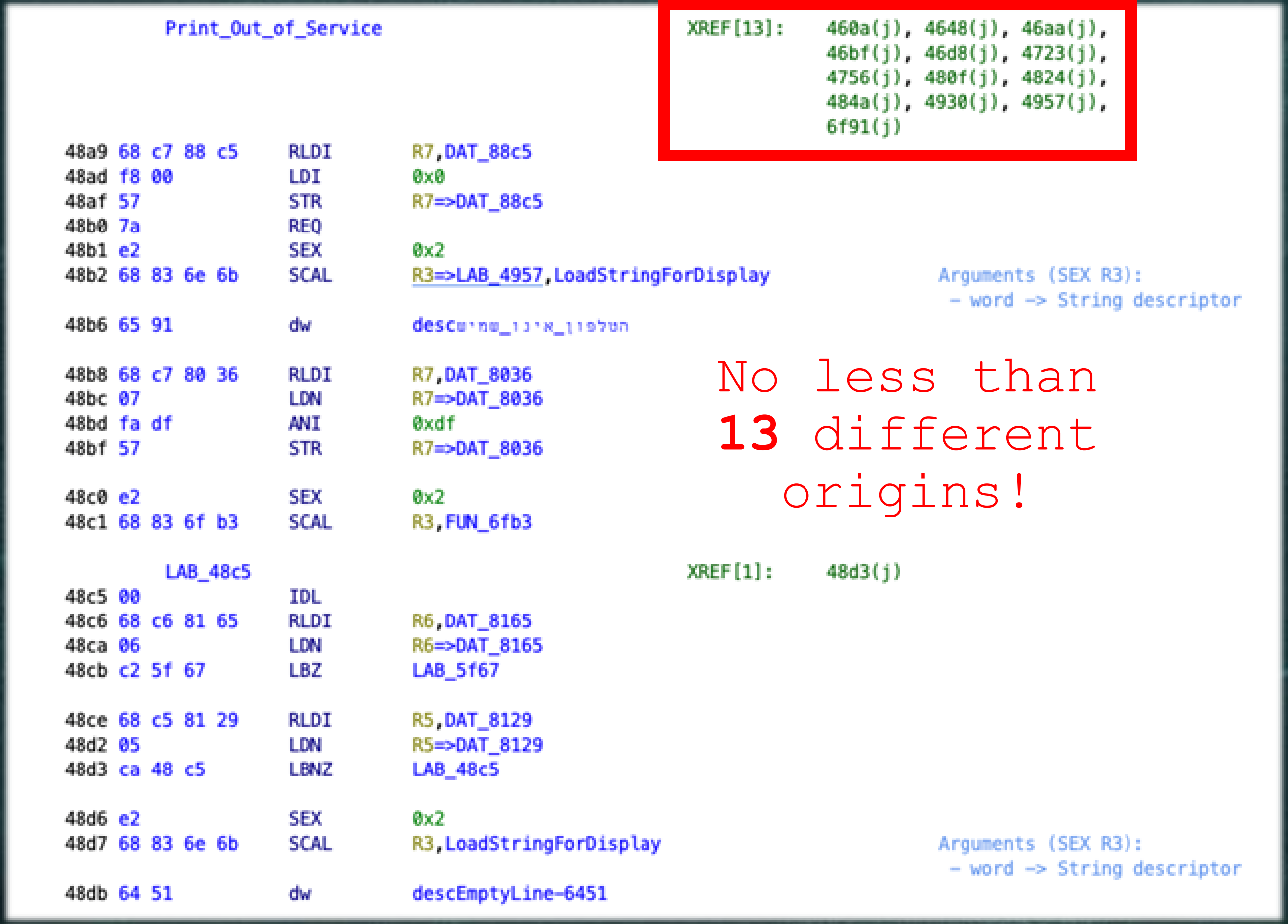
- Now that I had mapped all Hebrew strings and parts of the code, the search was easy.



```
             Print_Out_of_Service                       XREF[13]:    460a(j), 4648(j), 46aa(j),
                                                                     46bf(j), 46d8(j), 4723(j),
                                                                     4756(j), 480f(j), 4824(j),
                                                                     484a(j), 4930(j), 4957(j),
                                                                     6f91(j)
    48a9 68 c7 88 c5       RLDI       R7,DAT_88c5
    48ad f8 00             LDI        0x0
    48af 57                STR        R7=>DAT_88c5
    48b0 7a                REQ
    48b1 e2                SEX        0x2
    48b2 68 83 6e 6b       SCAL       R3=>LAB_4957,LoadStringForDisplay          Arguments (SEX R3):
                                                                                  - word -> String descriptor
    48b6 65 91             dw         descהטלפון_אינו_שמיש

    48b8 68 c7 80 36       RLDI       R7,DAT_8036
    48bc 07                LDN        R7=>DAT_8036
    48bd fa df             ANI        0xdf
    48bf 57                STR        R7=>DAT_8036

    48c0 e2                SEX        0x2
    48c1 68 83 6f b3       SCAL       R3,FUN_6fb3

             LAB_48c5                                   XREF[1]:     48d3(j)
    48c5 00                IDL
    48c6 68 c6 81 65       RLDI       R6,DAT_8165
    48ca 06                LDN        R6=>DAT_8165
    48cb c2 5f 67          LBZ        LAB_5f67

    48ce 68 c5 81 29       RLDI       R5,DAT_8129
    48d2 05                LDN        R5=>DAT_8129
    48d3 ca 48 c5          LBNZ       LAB_48c5

    48d6 e2                SEX        0x2
    48d7 68 83 6e 6b       SCAL       R3,LoadStringForDisplay                    Arguments (SEX R3):
                                                                                  - word -> String descriptor
    48db 64 51             dw         descEmptyLine-6451
```

Finding the error message

- Now that I had mapped all Hebrew strings and parts of the code, the search was easy.
- HOWEVER…



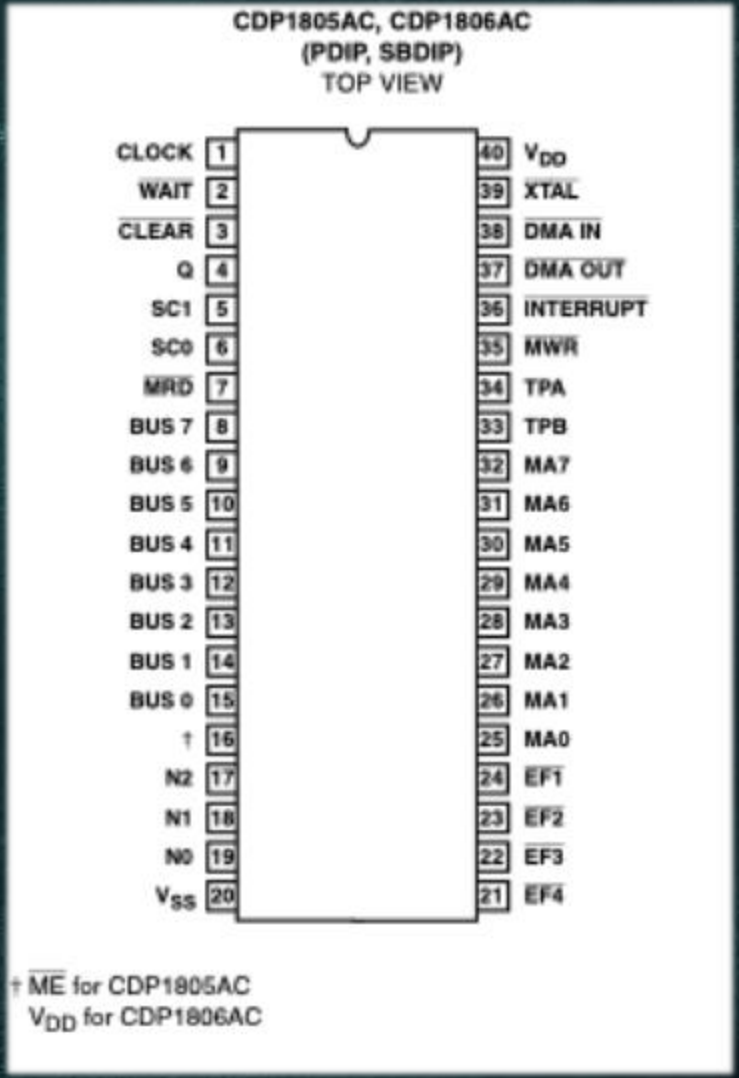
```
            Print_Out_of_Service                            XREF[13]:    460a(j), 4648(j), 46aa(j),
                                                                         46bf(j), 46d8(j), 4723(j),
                                                                         4756(j), 480f(j), 4824(j),
                                                                         484a(j), 4930(j), 4957(j),
                                                                         6f91(j)
48a9 68 c7 88 c5     RLDI      R7,DAT_88c5
48ad f8 00           LDI       0x0
48af 57              STR       R7=>DAT_88c5
48b0 7a              REQ
48b1 e2              SEX       0x2
48b2 68 83 6e 6b     SCAL      R3=>LAB_4957,LoadStringForDisplay            Arguments (SEX R3):
                                                                              - word -> String descriptor
48b6 65 91           dw        descהטלפון_אינו_שמיש

48b8 68 c7 80 36     RLDI      R7,DAT_8036
48bc 07              LDN       R7=>DAT_8036
48bd fa df           ANI       0xdf
48bf 57              STR       R7=>DAT_8036

48c0 e2              SEX       0x2
48c1 68 83 6f b3     SCAL      R3,FUN_6fb3

            LAB_48c5                                        XREF[1]:     48d3(j)
48c5 00              IDL
48c6 68 c6 81 65     RLDI      R6,DAT_8165
48ca 06              LDN       R6=>DAT_8165
48cb c2 5f 67        LBZ       LAB_5f67

48ce 68 c5 81 29     RLDI      R5,DAT_8129
48d2 05              LDN       R5=>DAT_8129
48d3 ca 48 c5        LBNZ      LAB_48c5

48d6 e2              SEX       0x2
48d7 68 83 6e 6b     SCAL      R3,LoadStringForDisplay                      Arguments (SEX R3):
                                                                              - word -> String descriptor
48db 64 51           dw        descEmptyLine-6451
```

No less than **13** different origins!

Finding the origin

- Just reading the code was not going to be helpful – I still hadn’t mapped the external peripherals (a mistake, in hind-sight).
- I decided to try to catch the address of the preceding opcodes by eavesdropping on the address bus.
    - Why? I have no idea. This project is not about the easy way out.

Finding the origin

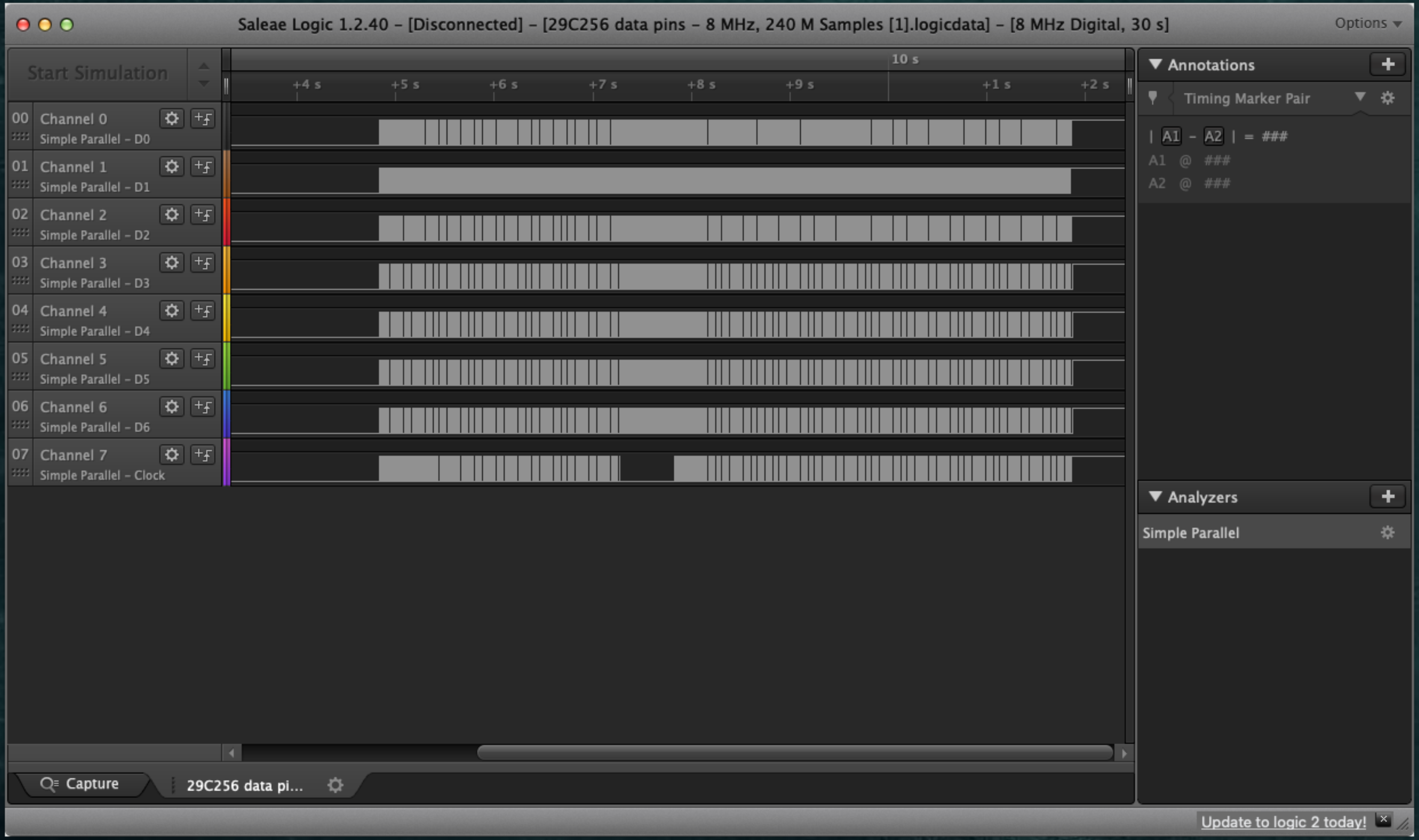


CDP1805AC, CDP1806AC
(PDIP, SBDIP)
TOP VIEW

† $\overline{\text{ME}}$ for CDP1805AC
V_{DD} for CDP1806AC

Finding the origin

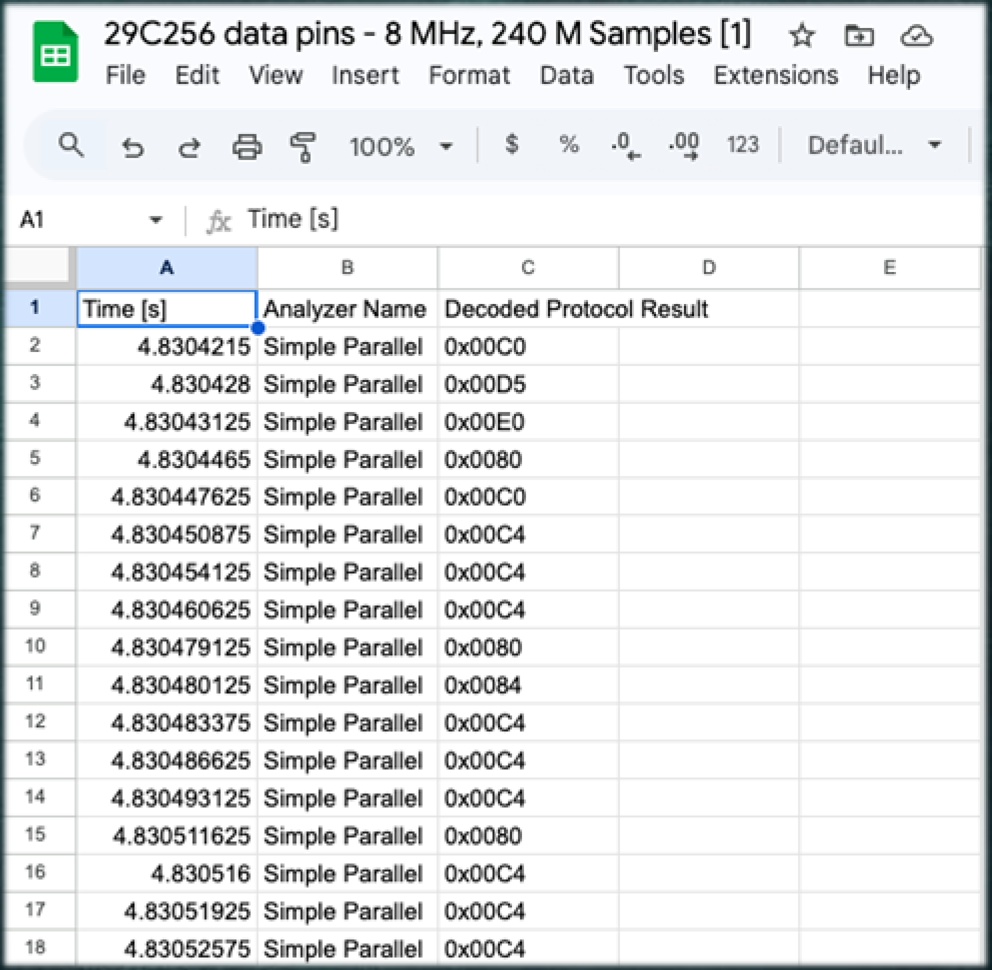
- I collected 7 seconds of data.



Saleae Logic 1.2.40 – [Disconnected] – [29C256 data pins – 8 MHz, 240 M Samples [1].logicdata] – [8 MHz Digital, 30 s]

Finding the origin

- I collected 7 seconds of data.
- Sadly, I couldn't find what I was looking for.
    - Maybe I tapped the wrong lines.
    - Maybe I captured the wrong time window.
    - Maybe something else.



29C256 data pins - 8 MHz, 240 M Samples [1]

A1 fx Time [s]

	A	B	C	D	E
1	Time [s]	Analyzer Name	Decoded Protocol Result		
2	4.8304215	Simple Parallel	0x00C0		
3	4.830428	Simple Parallel	0x00D5		
4	4.83043125	Simple Parallel	0x00E0		
5	4.8304465	Simple Parallel	0x0080		
6	4.830447625	Simple Parallel	0x00C0		
7	4.830450875	Simple Parallel	0x00C4		
8	4.830454125	Simple Parallel	0x00C4		
9	4.830460625	Simple Parallel	0x00C4		
10	4.830479125	Simple Parallel	0x0080		
11	4.830480125	Simple Parallel	0x0084		
12	4.830483375	Simple Parallel	0x00C4		
13	4.830486625	Simple Parallel	0x00C4		
14	4.830493125	Simple Parallel	0x00C4		
15	4.830511625	Simple Parallel	0x0080		
16	4.830516	Simple Parallel	0x00C4		
17	4.83051925	Simple Parallel	0x00C4		
18	4.83052575	Simple Parallel	0x00C4		

מה שלא הולך בכוח, הולך בעוד יותר כוח

Double or Nothing

- By now it has become personal.
- In my posession is the function that writes a string to the LCD.

➔ I could patch the code and send my own strings to the screen.

Double or Nothing

The plan:

1. Hook all locations that jump to 0x48A9 (Print_Out_of_Service).
2. Store current address in a memory location.
3. Resume original execution.
4. Overwrite 0x48A9 (Print_Out_of_Service) to print the stored address instead of the error message.

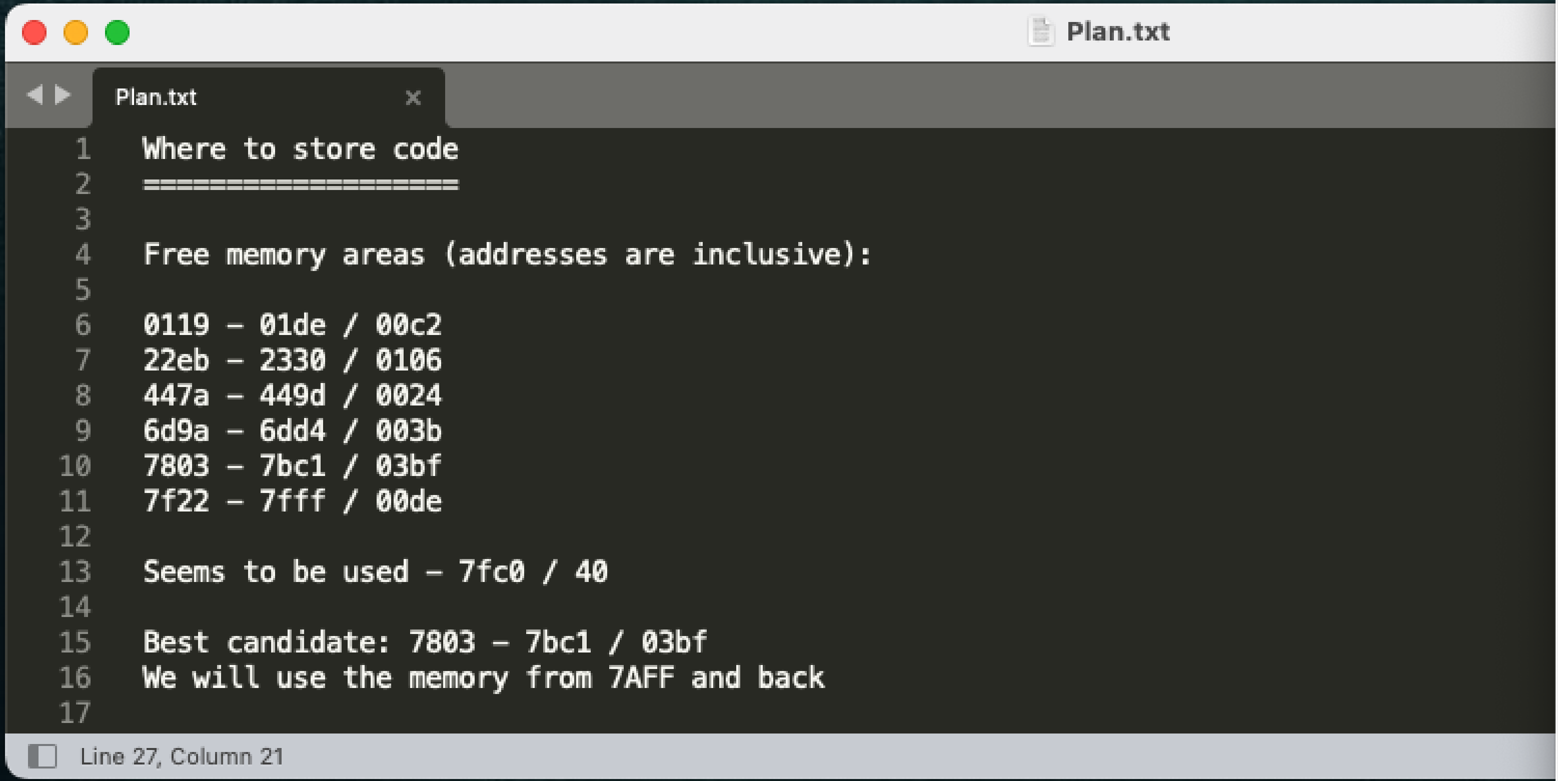
Double or Nothing

Requirements:

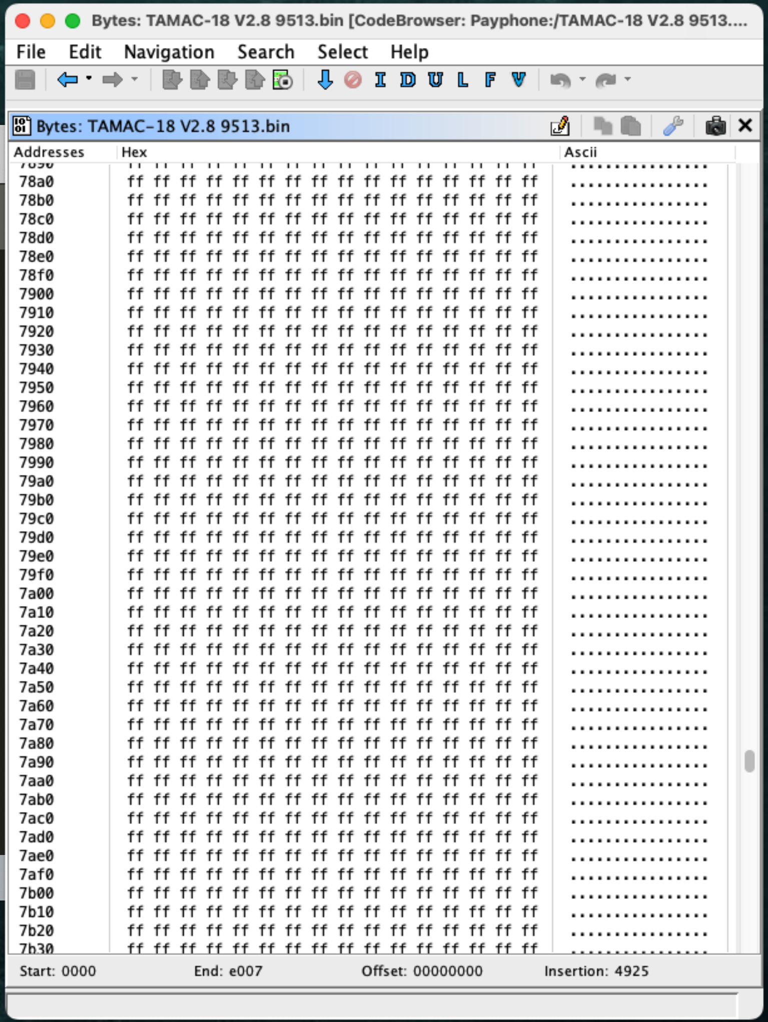
1. Must find unused memory areas for the hooking code.
2. Must determine which registers are available.
3. Must be able to convert an address to a string.
4. Better find an assembler or this will be a bitch.

Must find unused memory areas for the hooking code

- I mapped free and unused memory areas:



```
Where to store code
===================

Free memory areas (addresses are inclusive):

0119 - 01de / 00c2
22eb - 2330 / 0106
447a - 449d / 0024
6d9a - 6dd4 / 003b
7803 - 7bc1 / 03bf
7f22 - 7fff / 00de

Seems to be used - 7fc0 / 40

Best candidate: 7803 - 7bc1 / 03bf
We will use the memory from 7AFF and back

```

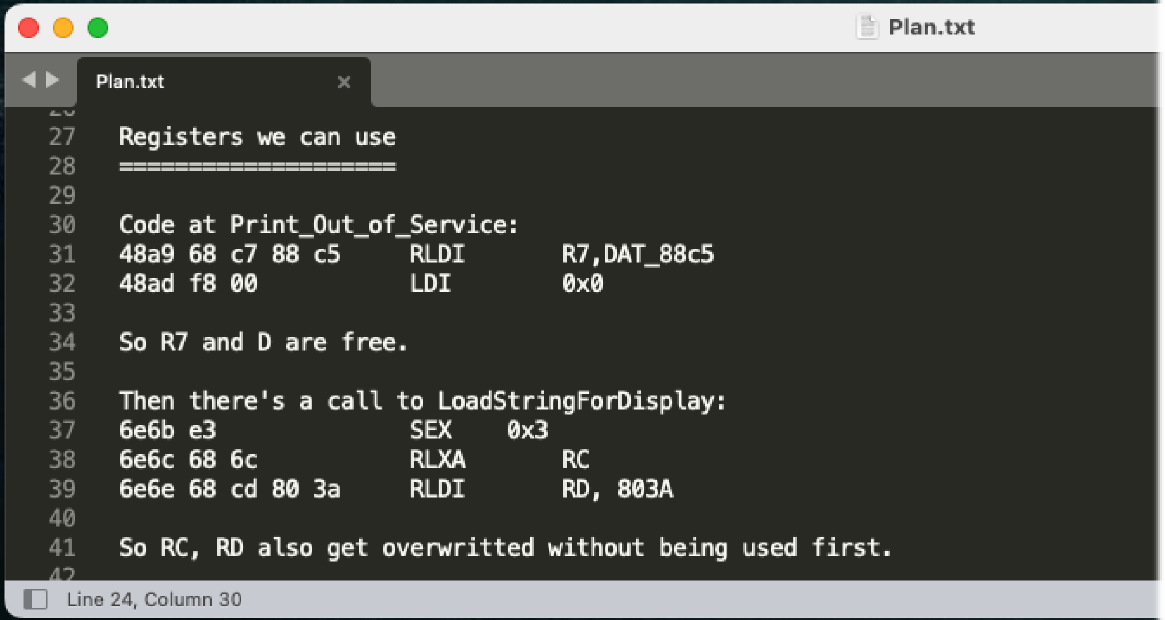

Bytes: TAMAC-18 V2.8 9513.bin [CodeBrowser: Payphone:/TAMAC-18 V2.8 9513....
File
Edit
Navigation
Search
Select
Help
I D U L F V
Bytes: TAMAC-18 V2.8 9513.bin

```
Addresses  Hex                                               Ascii
78a0       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
78b0       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
78c0       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
78d0       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
78e0       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
78f0       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7900       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7910       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7920       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7930       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7940       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7950       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7960       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7970       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7980       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7990       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
79a0       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
79b0       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
79c0       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
79d0       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
79e0       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
79f0       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7a00       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7a10       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7a20       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7a30       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7a40       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7a50       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7a60       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7a70       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7a80       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7a90       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7aa0       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7ab0       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7ac0       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7ad0       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7ae0       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7af0       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7b00       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7b10       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7b20       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
7b30       ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
```

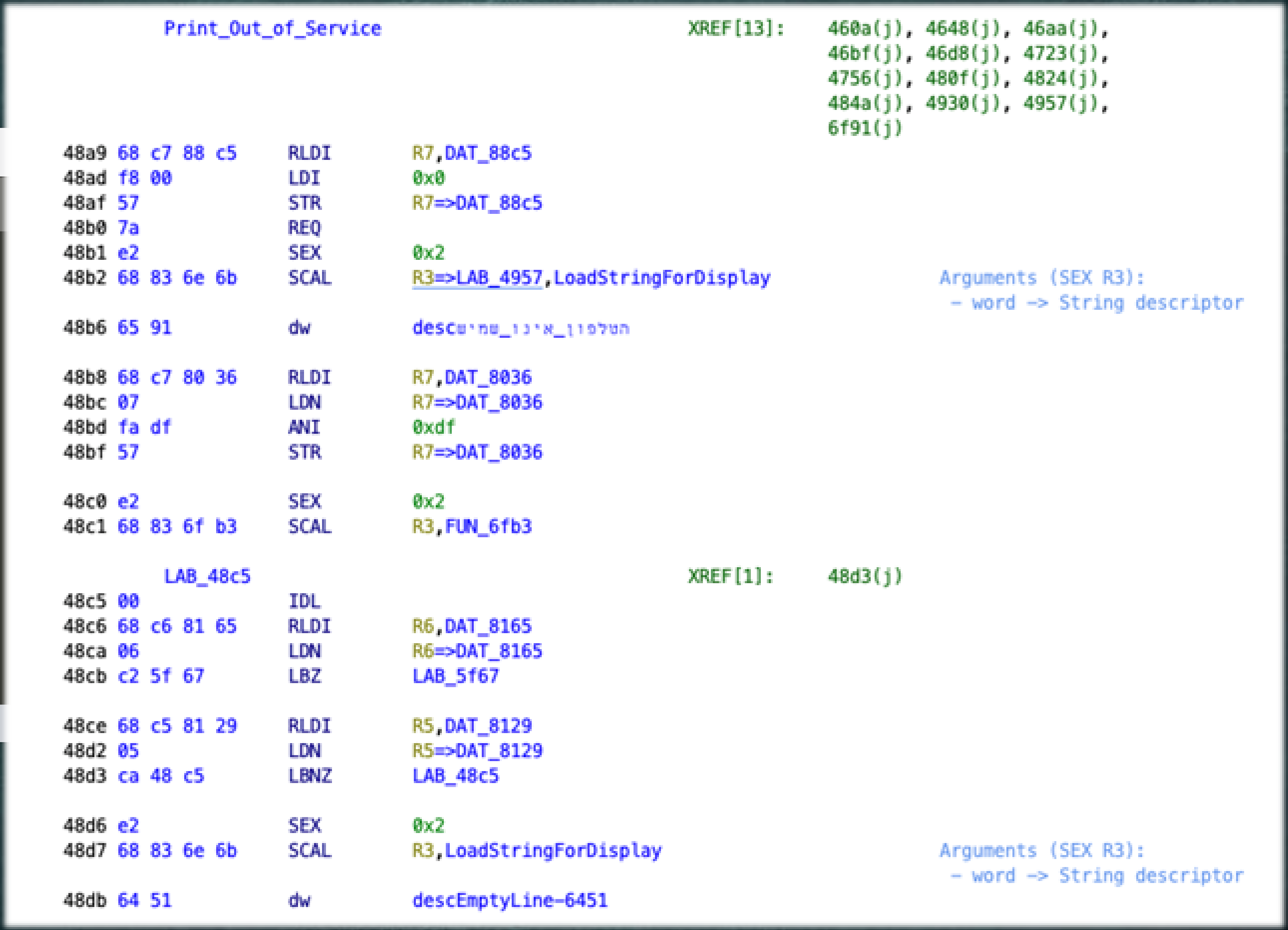
Start: 0000 End: e007 Offset: 00000000 Insertion: 4925

Must determine which registers are available

- I read the code and found the registers I can use:



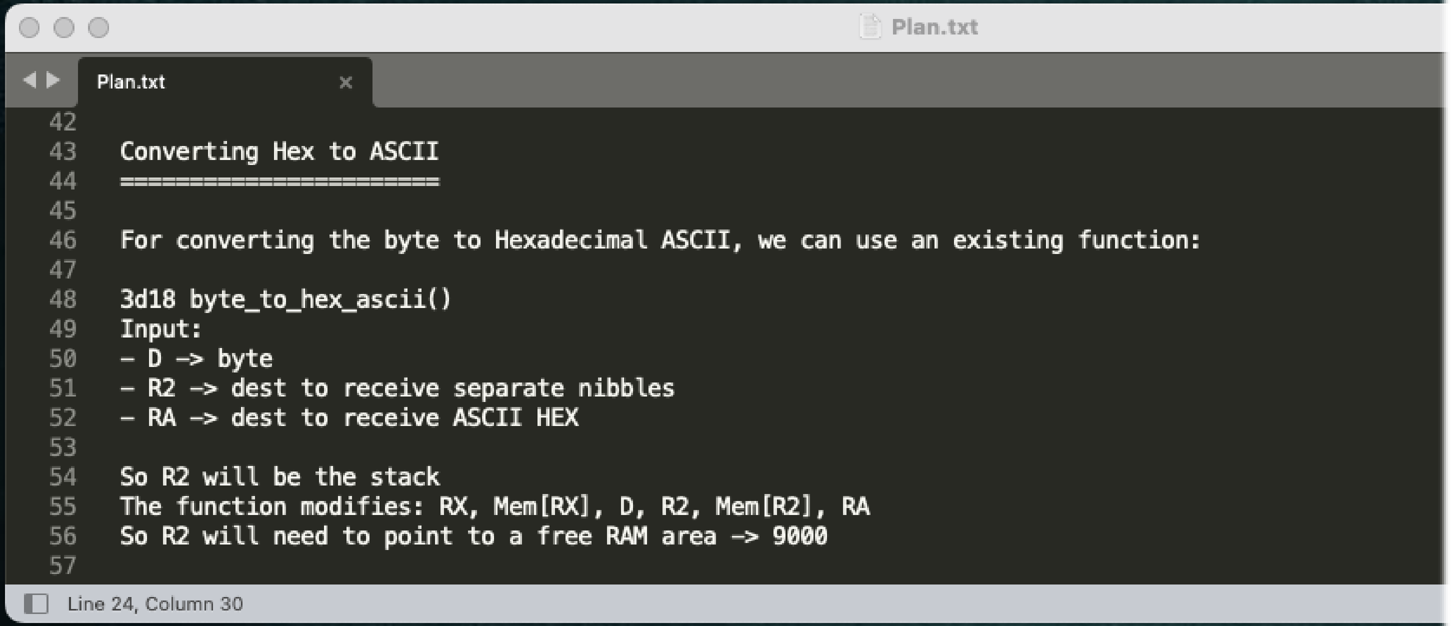
```
Registers we can use
====================

Code at Print_Out_of_Service:
48a9 68 c7 88 c5      RLDI       R7,DAT_88c5
48ad f8 00            LDI        0x0

So R7 and D are free.

Then there's a call to LoadStringForDisplay:
6e6b e3               SEX    0x3
6e6c 68 6c            RLXA       RC
6e6e 68 cd 80 3a      RLDI       RD, 803A

So RC, RD also get overwritted without being used first.
```



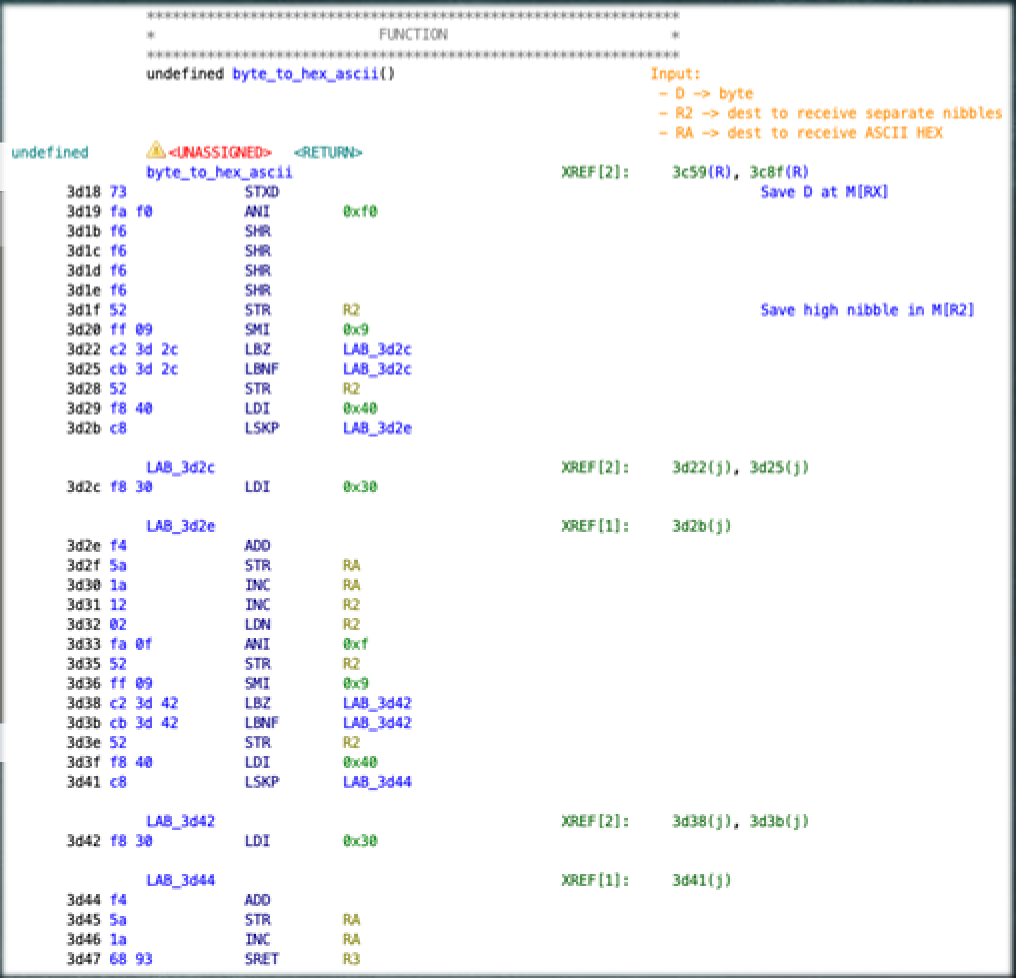
```
             Print_Out_of_Service                          XREF[13]:    460a(j), 4648(j), 46aa(j),
                                                                        46bf(j), 46d8(j), 4723(j),
                                                                        4756(j), 480f(j), 4824(j),
                                                                        484a(j), 4930(j), 4957(j),
                                                                        6f91(j)
    48a9 68 c7 88 c5     RLDI       R7,DAT_88c5
    48ad f8 00           LDI        0x0
    48af 57              STR        R7=>DAT_88c5
    48b0 7a              REQ
    48b1 e2              SEX        0x2
    48b2 68 83 6e 6b     SCAL       R3=>LAB_4957,LoadStringForDisplay              Arguments (SEX R3):
                                                                                     - word -> String descriptor
    48b6 65 91           dw         descהטלפון_אינו_שמיש

    48b8 68 c7 80 36     RLDI       R7,DAT_8036
    48bc 07              LDN        R7=>DAT_8036
    48bd fa df           ANI        0xdf
    48bf 57              STR        R7=>DAT_8036

    48c0 e2              SEX        0x2
    48c1 68 83 6f b3     SCAL       R3,FUN_6fb3

             LAB_48c5                                      XREF[1]:     48d3(j)
    48c5 00              IDL
    48c6 68 c6 81 65     RLDI       R6,DAT_8165
    48ca 06              LDN        R6=>DAT_8165
    48cb c2 5f 67        LBZ        LAB_5f67

    48ce 68 c5 81 29     RLDI       R5,DAT_8129
    48d2 05              LDN        R5=>DAT_8129
    48d3 ca 48 c5        LBNZ       LAB_48c5

    48d6 e2              SEX        0x2
    48d7 68 83 6e 6b     SCAL       R3,LoadStringForDisplay                        Arguments (SEX R3):
                                                                                     - word -> String descriptor
    48db 64 51           dw         descEmptyLine-6451
```

Must be able to convert an address to a string

- Not as cool as a ROP Gadget, but still Living Off the Land:

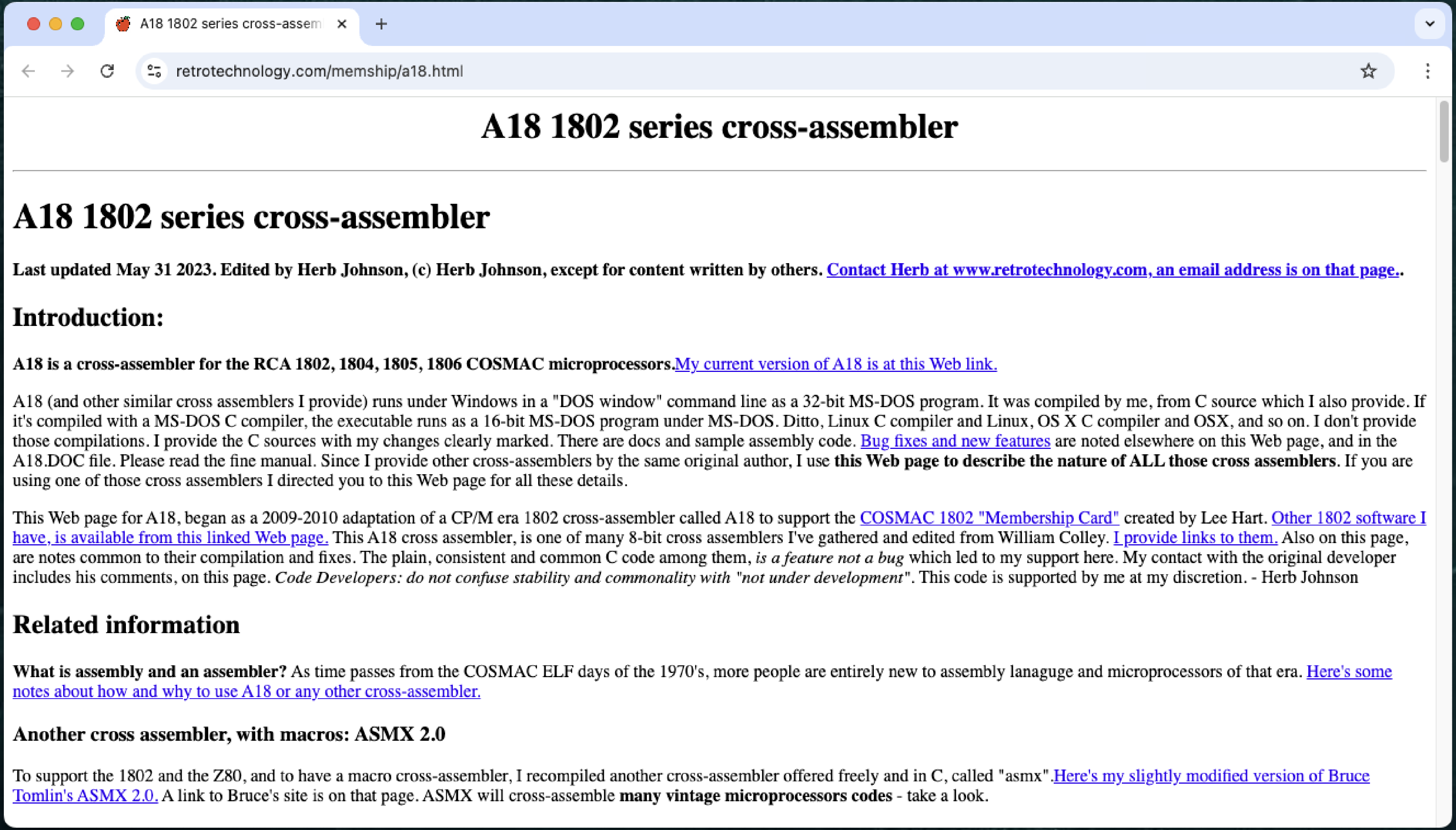


```
Converting Hex to ASCII
=======================

For converting the byte to Hexadecimal ASCII, we can use an existing function:

3d18 byte_to_hex_ascii()
Input:
- D -> byte
- R2 -> dest to receive separate nibbles
- RA -> dest to receive ASCII HEX

So R2 will be the stack
The function modifies: RX, Mem[RX], D, R2, Mem[R2], RA
So R2 will need to point to a free RAM area -> 9000
```



```
                 *******************************************************************
                 *                             FUNCTION                            *
                 *******************************************************************
                 undefined byte_to_hex_ascii()                            Input:
                                                                           - D -> byte
                                                                           - R2 -> dest to receive separate nibbles
                                                                           - RA -> dest to receive ASCII HEX
undefined        ⚠<UNASSIGNED>   <RETURN>
                 byte_to_hex_ascii                         XREF[2]:     3c59(R), 3c8f(R)
       3d18 73              STXD                                                 Save D at M[RX]
       3d19 fa f0           ANI        0xf0
       3d1b f6              SHR
       3d1c f6              SHR
       3d1d f6              SHR
       3d1e f6              SHR
       3d1f 52              STR        R2                                        Save high nibble in M[R2]
       3d20 ff 09           SMI        0x9
       3d22 c2 3d 2c        LBZ        LAB_3d2c
       3d25 cb 3d 2c        LBNF       LAB_3d2c
       3d28 52              STR        R2
       3d29 f8 40           LDI        0x40
       3d2b c8              LSKP       LAB_3d2e

                 LAB_3d2c                                  XREF[2]:     3d22(j), 3d25(j)
       3d2c f8 30           LDI        0x30

                 LAB_3d2e                                  XREF[1]:     3d2b(j)
       3d2e f4              ADD
       3d2f 5a              STR        RA
       3d30 1a              INC        RA
       3d31 12              INC        R2
       3d32 02              LDN        R2
       3d33 fa 0f           ANI        0xf
       3d35 52              STR        R2
       3d36 ff 09           SMI        0x9
       3d38 c2 3d 42        LBZ        LAB_3d42
       3d3b cb 3d 42        LBNF       LAB_3d42
       3d3e 52              STR        R2
       3d3f f8 40           LDI        0x40
       3d41 c8              LSKP       LAB_3d44

                 LAB_3d42                                  XREF[2]:     3d38(j), 3d3b(j)
       3d42 f8 30           LDI        0x30

                 LAB_3d44                                  XREF[1]:     3d41(j)
       3d44 f4              ADD
       3d45 5a              STR        RA
       3d46 1a              INC        RA
       3d47 68 93           SRET       R3
```

Better find an assembler or this will be a bitch


A18 1802 series cross-assem
retrotechnology.com/memship/a18.html

A18 1802 series cross-assembler

A18 1802 series cross-assembler

Last updated May 31 2023. Edited by Herb Johnson, (c) Herb Johnson, except for content written by others. **Contact Herb at www.retrotechnology.com, an email address is on that page.**.

Introduction:

A18 is a cross-assembler for the RCA 1802, 1804, 1805, 1806 COSMAC microprocessors. My current version of A18 is at this Web link.

A18 (and other similar cross assemblers I provide) runs under Windows in a "DOS window" command line as a 32-bit MS-DOS program. It was compiled by me, from C source which I also provide. If it's compiled with a MS-DOS C compiler, the executable runs as a 16-bit MS-DOS program under MS-DOS. Ditto, Linux C compiler and Linux, OS X C compiler and OSX, and so on. I don't provide those compilations. I provide the C sources with my changes clearly marked. There are docs and sample assembly code. Bug fixes and new features are noted elsewhere on this Web page, and in the A18.DOC file. Please read the fine manual. Since I provide other cross-assemblers by the same original author, I use **this Web page to describe the nature of ALL those cross assemblers**. If you are using one of those cross assemblers I directed you to this Web page for all these details.

This Web page for A18, began as a 2009-2010 adaptation of a CP/M era 1802 cross-assembler called A18 to support the COSMAC 1802 "Membership Card" created by Lee Hart. Other 1802 software I have, is available from this linked Web page. This A18 cross assembler, is one of many 8-bit cross assemblers I've gathered and edited from William Colley. I provide links to them. Also on this page, are notes common to their compilation and fixes. The plain, consistent and common C code among them, *is a feature not a bug* which led to my support here. My contact with the original developer includes his comments, on this page. *Code Developers: do not confuse stability and commonality with "not under development"*. This code is supported by me at my discretion. - Herb Johnson

Related information

What is assembly and an assembler? As time passes from the COSMAC ELF days of the 1970's, more people are entirely new to assembly lanaguge and microprocessors of that era. Here's some notes about how and why to use A18 or any other cross-assembler.

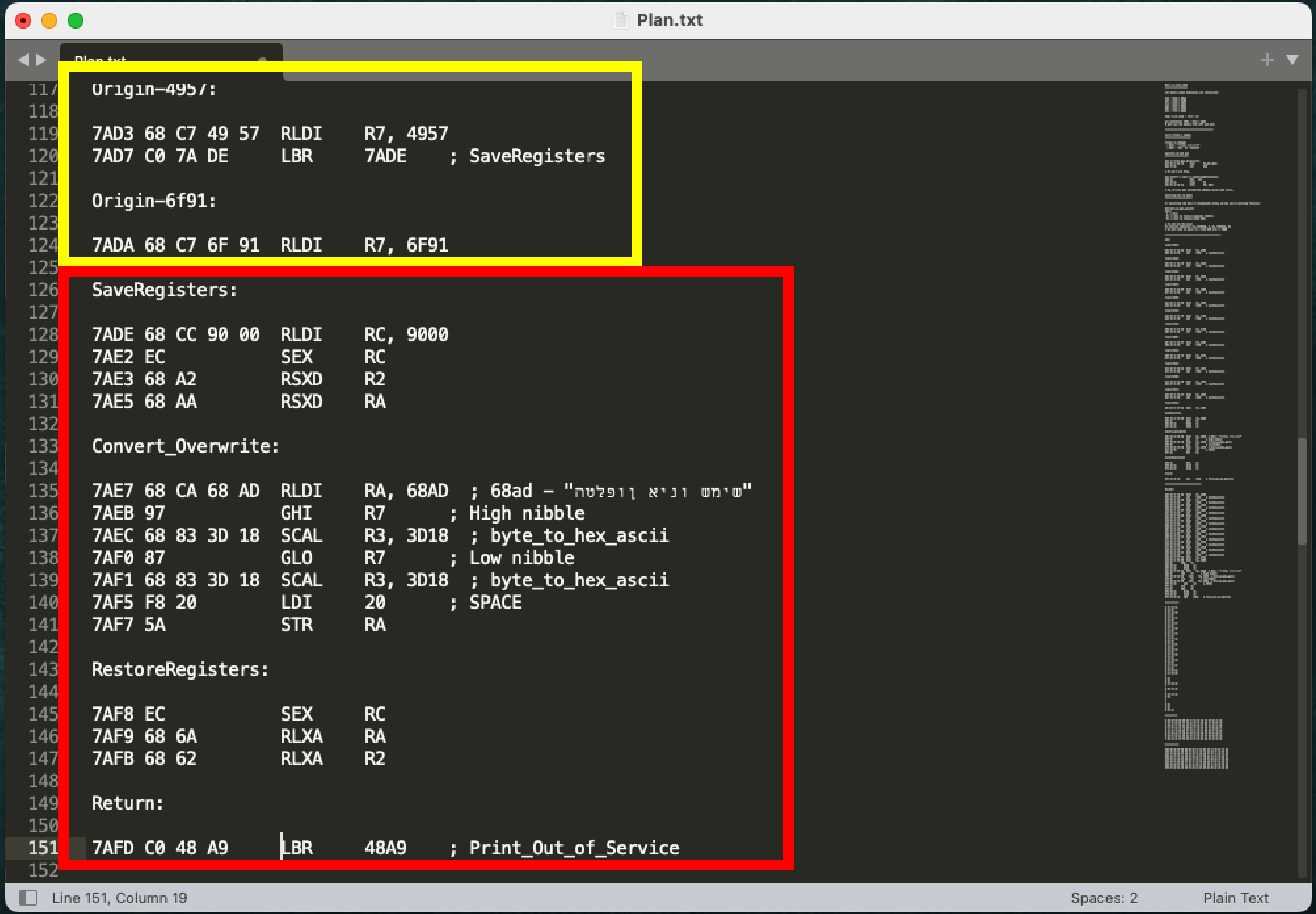
Another cross assembler, with macros: ASMX 2.0

To support the 1802 and the Z80, and to have a macro cross-assembler, I recompiled another cross-assembler offered freely and in C, called "asmx".Here's my slightly modified version of Bruce Tomlin's ASMX 2.0. A link to Bruce's site is on that page. ASMX will cross-assemble **many vintage microprocessors codes** - take a look.

Putting it all together

Hooks go here

Convert to Hex,
Store,
Resume execution



```
Origin-4957:

7AD3 68 C7 49 57  RLDI    R7, 4957
7AD7 C0 7A DE     LBR     7ADE     ; SaveRegisters

Origin-6f91:

7ADA 68 C7 6F 91  RLDI    R7, 6F91

SaveRegisters:

7ADE 68 CC 90 00  RLDI    RC, 9000
7AE2 EC           SEX     RC
7AE3 68 A2        RSXD    R2
7AE5 68 AA        RSXD    RA

Convert_Overwrite:

7AE7 68 CA 68 AD  RLDI    RA, 68AD  ; 68ad – "שימש וניא ןופלטה"
7AEB 97           GHI     R7       ; High nibble
7AEC 68 83 3D 18  SCAL    R3, 3D18  ; byte_to_hex_ascii
7AF0 87           GLO     R7       ; Low nibble
7AF1 68 83 3D 18  SCAL    R3, 3D18  ; byte_to_hex_ascii
7AF5 F8 20        LDI     20       ; SPACE
7AF7 5A           STR     RA

RestoreRegisters:

7AF8 EC           SEX     RC
7AF9 68 6A        RLXA    RA
7AFB 68 62        RLXA    R2

Return:

7AFD C0 48 A9     LBR     48A9     ; Print_Out_of_Service
```

Putting it all together

There were two problems with the code:

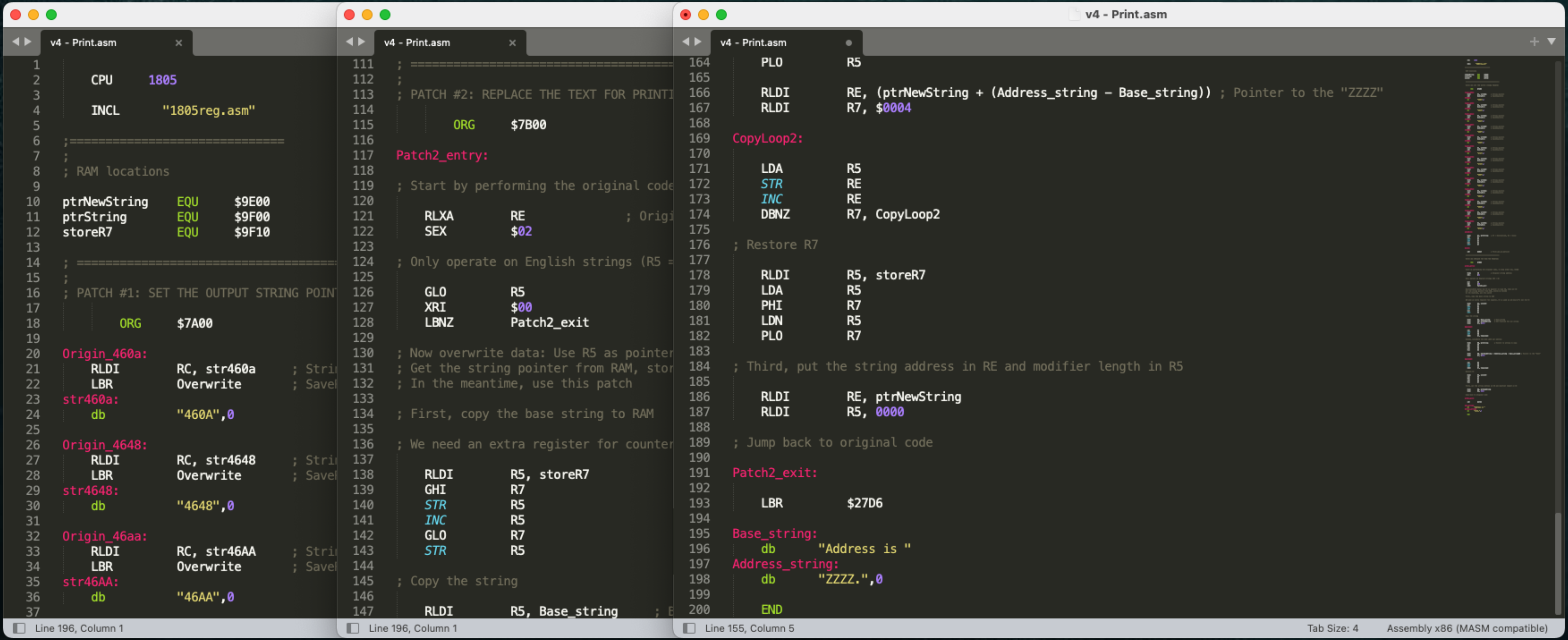
1. Location 0x68AD (הטלפון אינו שמיש) is ROM.
2. Just printing the address is not cool enough.

The solution:

1. Save the string somewhere else and skip the address assignment.
2. Print a nicer English text.

Putting it all together

- The final code:



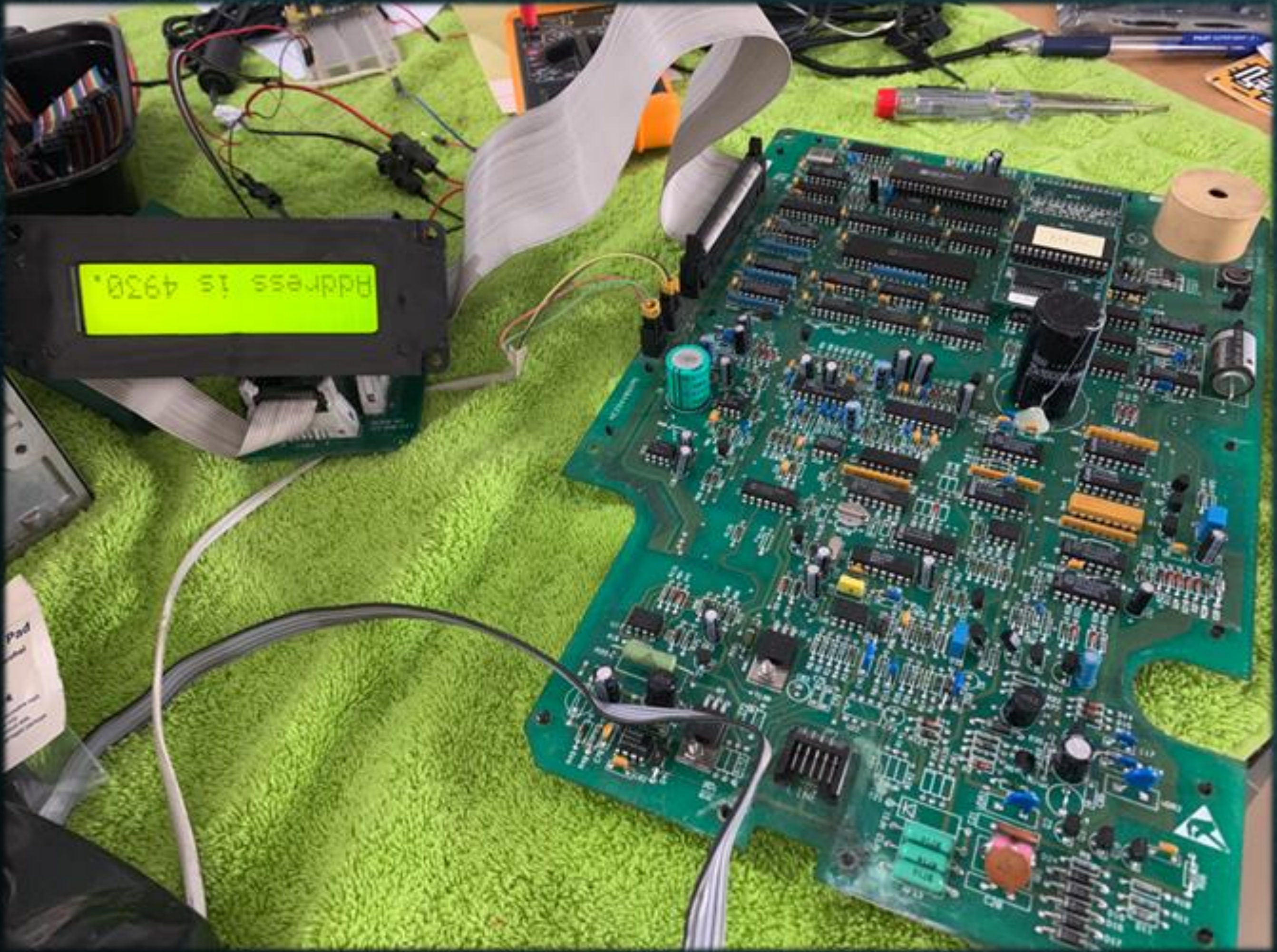
```

    CPU     1805

    INCL     "1805reg.asm"

;==============================
;
; RAM locations

ptrNewString    EQU     $9E00
ptrString       EQU     $9F00
storeR7         EQU     $9F10

; ===================================
;
; PATCH #1: SET THE OUTPUT STRING POIN

        ORG     $7A00

Origin_460a:
    RLDI        RC, str460a     ; Stri
    LBR         Overwrite       ; Save
str460a:
    db          "460A",0

Origin_4648:
    RLDI        RC, str4648     ; Stri
    LBR         Overwrite       ; Save
str4648:
    db          "4648",0

Origin_46aa:
    RLDI        RC, str46AA     ; Stri
    LBR         Overwrite       ; Save
str46AA:
    db          "46AA",0

```

```
; ====================================
;
; PATCH #2: REPLACE THE TEXT FOR PRINTI

        ORG     $7B00

Patch2_entry:

; Start by performing the original code

    RLXA        RE                  ; Origi
    SEX         $02

; Only operate on English strings (R5 =

    GLO         R5
    XRI         $00
    LBNZ        Patch2_exit

; Now overwrite data: Use R5 as pointer
; Get the string pointer from RAM, stor
; In the meantime, use this patch

; First, copy the base string to RAM

; We need an extra register for counter

    RLDI        R5, storeR7
    GHI         R7
    STR         R5
    INC         R5
    GLO         R7
    STR         R5

; Copy the string

    RLDI        R5, Base_string     ; E
```

```
    PLO         R5

    RLDI        RE, (ptrNewString + (Address_string - Base_string)) ; Pointer to the "ZZZZ"
    RLDI        R7, $0004

CopyLoop2:

    LDA         R5
    STR         RE
    INC         RE
    DBNZ        R7, CopyLoop2

; Restore R7

    RLDI        R5, storeR7
    LDA         R5
    PHI         R7
    LDN         R5
    PLO         R7

; Third, put the string address in RE and modifier length in R5

    RLDI        RE, ptrNewString
    RLDI        R5, 0000

; Jump back to original code

Patch2_exit:

    LBR         $27D6

Base_string:
    db      "Address is "
Address_string:
    db      "ZZZZ.",0

    END
```

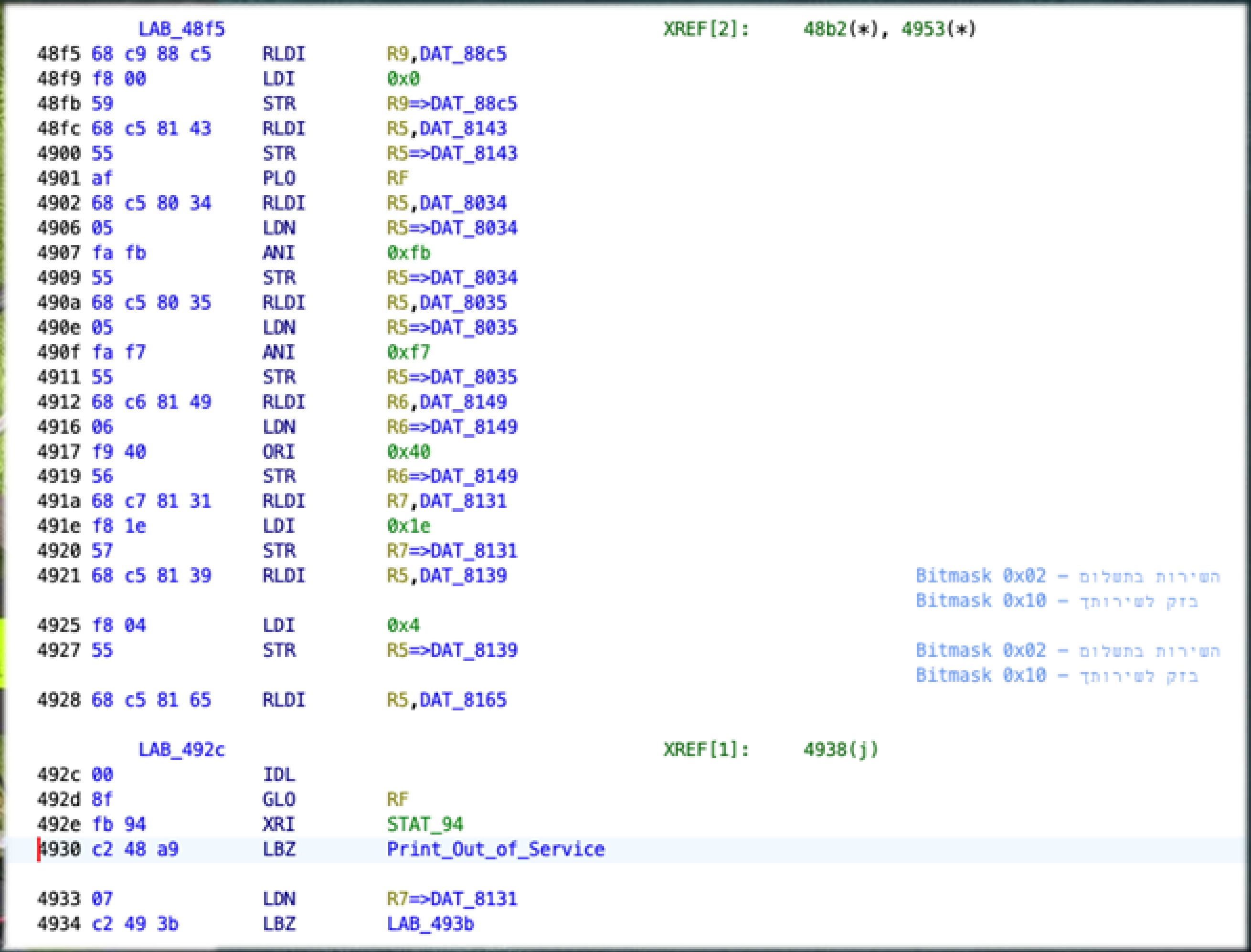
Putting it all together

- And the result:



Putting it all together

- And the result:



```
                     LAB_48f5                                    XREF[2]:     48b2(*), 4953(*)
    48f5 68 c9 88 c5      RLDI       R9,DAT_88c5
    48f9 f8 00            LDI        0x0
    48fb 59               STR        R9=>DAT_88c5
    48fc 68 c5 81 43      RLDI       R5,DAT_8143
    4900 55               STR        R5=>DAT_8143
    4901 af               PLO        RF
    4902 68 c5 80 34      RLDI       R5,DAT_8034
    4906 05               LDN        R5=>DAT_8034
    4907 fa fb            ANI        0xfb
    4909 55               STR        R5=>DAT_8034
    490a 68 c5 80 35      RLDI       R5,DAT_8035
    490e 05               LDN        R5=>DAT_8035
    490f fa f7            ANI        0xf7
    4911 55               STR        R5=>DAT_8035
    4912 68 c6 81 49      RLDI       R6,DAT_8149
    4916 06               LDN        R6=>DAT_8149
    4917 f9 40            ORI        0x40
    4919 56               STR        R6=>DAT_8149
    491a 68 c7 81 31      RLDI       R7,DAT_8131
    491e f8 1e            LDI        0x1e
    4920 57               STR        R7=>DAT_8131
    4921 68 c5 81 39      RLDI       R5,DAT_8139                                        Bitmask 0x02 - השירות בתשלום
                                                                                        Bitmask 0x10 - בזק לשירותך

    4925 f8 04            LDI        0x4
    4927 55               STR        R5=>DAT_8139                                       Bitmask 0x02 - השירות בתשלום
                                                                                        Bitmask 0x10 - בזק לשירותך

    4928 68 c5 81 65      RLDI       R5,DAT_8165

                     LAB_492c                                    XREF[1]:     4938(j)
    492c 00               IDL
    492d 8f               GLO        RF
    492e fb 94            XRI        STAT_94
    4930 c2 48 a9         LBZ        Print_Out_of_Service

    4933 07               LDN        R7=>DAT_8131
    4934 c2 49 3b         LBZ        LAB_493b
```

TA0003:

Persistence

Reuse the Hardware

- I started contemplating replacing the PCB.
- I needed to know which of the peripherals I can keep.
  - The keyboard is a simple matrix – no brainer.
  - The earpiece is also very simple – no logic there either.
  - The card reader is not relevant, will keep for appearance.
  - What about the LCD?

The LCD

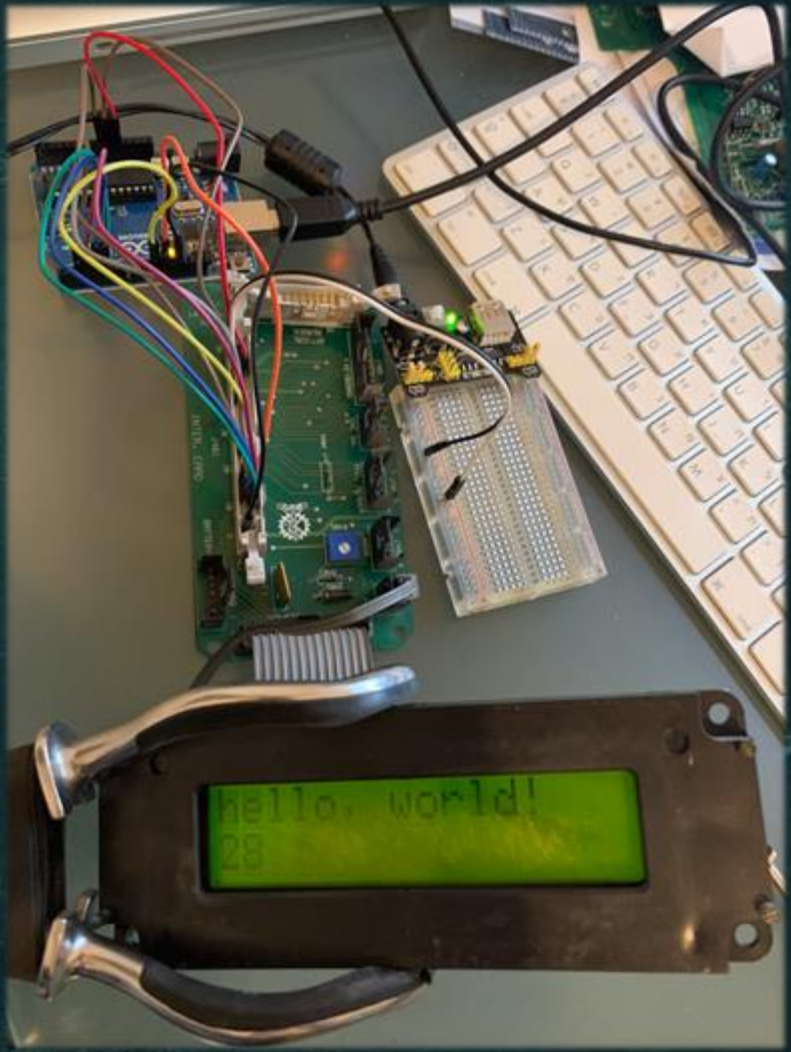
- The LCD provides 2 rows of 16 characters.
- It uses the standard HD44780 driver (IYKYK).

The LCD

- The LCD provides 2 rows of 16 characters.
- It uses the standard HD44780 driver (IYKYK).
- A quick test verified that.

The LCD

- There are standard libraries for Arduino which support the HD44780.
- I used the distribution board as a way to verify my pinout mapping.
    - With an external 5v supply for the backlight.

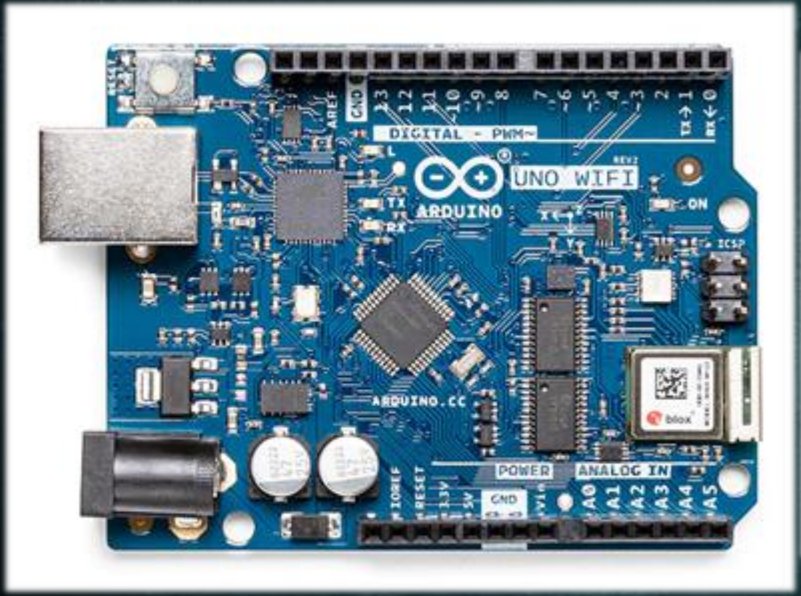


Options

- Replace the existing PCB with modern electronics.
  - A regular PSTN phone?

Options

- Replace the existing PCB with modern electronics.
  - A regular PSTN phone?
  - An Arduino/RasPi with a CallerID module?



And Then…

Life happened

- 2020 brought COVID.
- I got swamped with work (B787 has a LOT of manuals).
- The project was frozen.

Until…

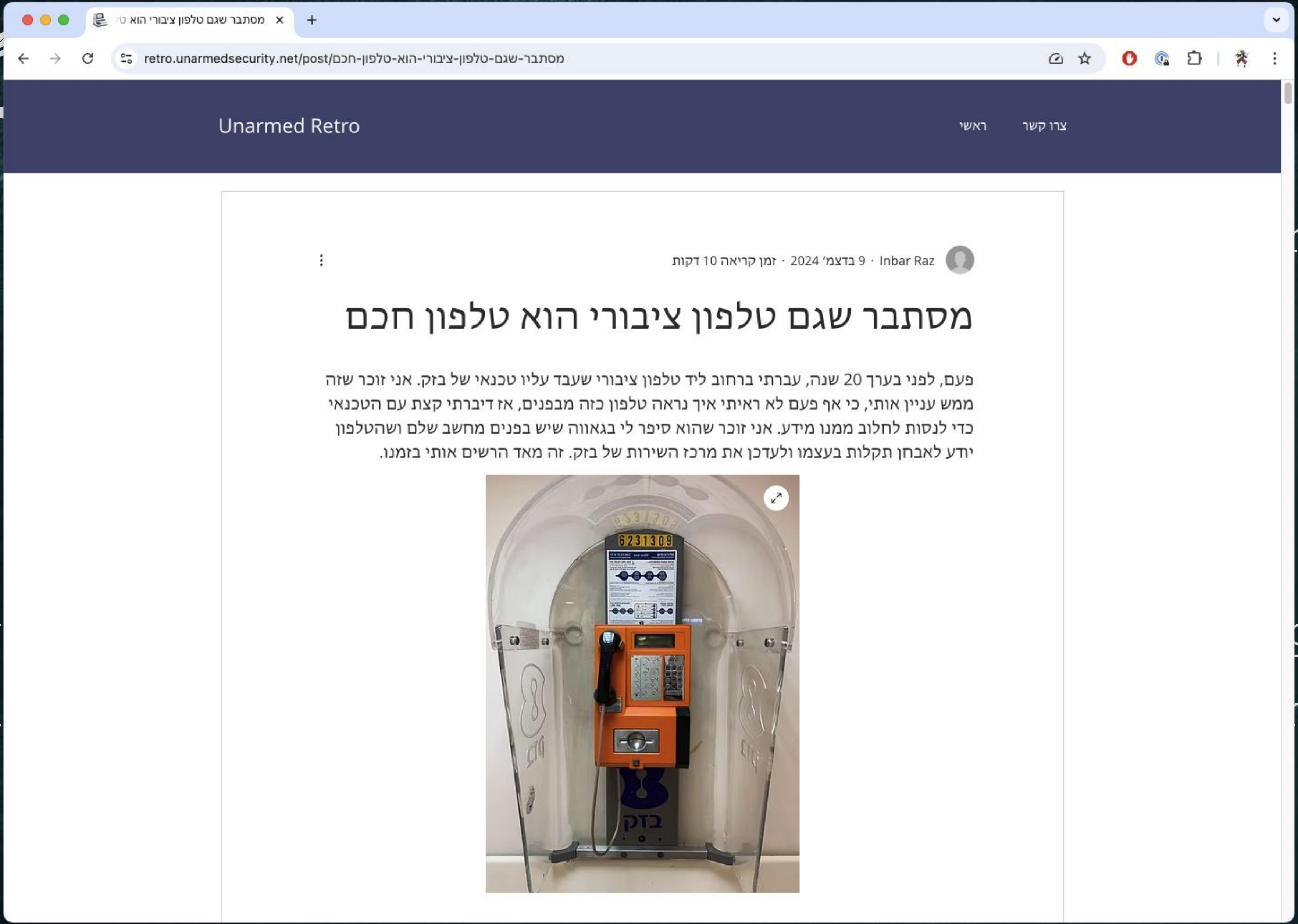
- In November 2024 I met Nicole Fishbein at ekoparty.
- I told her about the project, she asked for photos.
- I ended up writing a blog post instead.

Life happene

- 2020 b
- I got nuals)
- The pr

Until…

- In Nov party
- I told hotos
- I ende



מסתבר שגם טלפון ציבורי הוא ט
retro.unarmedsecurity.net/post/מסתבר-שגם-טלפון-ציבורי-הוא-טלפון-חכם
Unarmed Retro
ראשי
צרו קשר
6231309
בזק

Inbar Raz · 9 בדצמ׳ 2024 · זמן קריאה 10 דקות

מסתבר שגם טלפון ציבורי הוא טלפון חכם

פעם, לפני בערך 20 שנה, עברתי ברחוב ליד טלפון ציבורי שעבד עליו טכנאי של בזק. אני זוכר שזה
ממש עניין אותי, כי אף פעם לא ראיתי איך נראה טלפון כזה מבפנים, אז דיברתי קצת עם הטכנאי
כדי לנסות לחלוב ממנו מידע. אני זוכר שהוא סיפר לי בגאווה שיש בפנים מחשב שלם ושהטלפון
יודע לאבחן תקלות בעצמו ולעדכן את מרכז השירות של בזק. זה מאד הרשים אותי בזמנו.

Life happened

- The blog EXPLODED.
- And that’s all I remember, Doctor.

Looking Forward

What happens next?

- Continue reversing the firmware.
  - Collaborate with others?
  - Maybe end up with a followup talk?
- Communicate with the back office?
- Build a modern replacement hardware.
- Use for creating CTFs (BSidesTLV, anyone?)

HACK.LU 2025

BREAK. BUILD. SHARE.

Inbar Raz

Reversing a Payphone for Fun but no Profit

@inbarraz / @zenitysec – HIRING AI RESEARCHERS