A journey documenting the Sanco 8003 computer

FOSDEM 2024-02-04

Giovan Battista - giomba

- MS Computer Engineering @ University of Pisa
- Embedded software developer
- Retrocomputer Enthusiast

Find me on...

- www.giomba.it





Giulio - giuliof

- MS Electronic Engineering @ University of Pisa
- Firmware engineer in machine vision field
- Too many hobbies to fit in three-line bio

Find me on...

Github: <u>giuliof</u> Mastodon: <u>@giuliof@mastodon.uno</u> Twitter (?): <u>@GiulioFie</u> Personal: <u>me.giuliof.it/en/</u>



A computer murder was prevented that day...





However... we owe a special thank you to *"Associazione Porte Aperte"* for this gift!

What is this?

- Computer from the 80s
- Runs CP/M
- Possible italian rebrand
- Very little information from the internet

C'E' SANCO IBEX E SANCO IBEX.

Perché c'è azienda e azienda. E ognuna ha problemi diversi di gestione e diverse prospettive future. E per ognuna SANCO iBEX ha previsto un minisistema gestionale adeguato. Per questo, cercando fra SANCO iBEX e SANCO iBEX, troverai ciò che ti occorre. Per esempio: la tua azienda è piccola, ma ha bisogno di una gestione moderna e razionale, con un investimento contenuto? Ecco SANCO iBEX mod, 8001, un calcolatore professionale al prezzo di un personal. E se ti occorre una memoria di massa piú estesa, c'è SANCO iBEX mod. 8103. O desideri per caso una gestione integrata con tutti gli archivi in linea e un eventuale secondo posto di lavoro? La soluzione ai tuoi problemi è SÁNCO iBEX mod. 8150. Ma se la tua azienda è già cresciuta, se hai esigenze di

O A https://en.m.wikipedia.org/wiki/CP/M

(comparatively) low-cost microcomputers running CP/M, as independent programmers and hackers bought them and shared their creations in user groups.^[11] CP/M was eventually displaced by DOS following the 1981 introduction of the IBM PC.

~ History

^ Hardware model

.



Sanco 8001 computer, running under CP/M 2.2 (1982)

http://www.vintads.it/file.php?cod=802





System ROM



Standard 2732 (4KiB) Z80 instructions Mapped at... 0xC000? "BIOS"

jp ;[c000] \$c030 ld a,\$89 ;[c030] out (\$83),a ;[c032] di ;[c034] ld a,\$10 ;[c035] out (\$81),a ;[c037]

0001



00028983

....

https://z88dk.org/ https://sigrok.org/wiki/PulseView https://github.com/GLGPrograms/ceda-rom-disassembly

Bus inspection setup



.text and .data not found

	ld	hl,\$c134	;[c108]	
labe	; loop a	around \$c134 table, writing to \$	b1 (chanı	nel A)
	ld inc cp jr out ld out inc jr	a,(hl) hl \$ff z,label_c119 (\$b1),a a,(hl) (\$b1),a hl label_c10b	<pre>;[c10b] ;[c10c] ;[c10d] ;[c10f] ;[c111] ;[c113] ;[c114] ;[c116] ;[c117]</pre>	load index of internal SIO register fetch next data check table end if table end, configure channel B index internal SIO register load desired internal SIO register value write desidered internal SIO register value fetch next data loop
sio	; Confi	guration table for SIO ChA?		
5.00	BYTE \$0 BYTE \$1 BYTE \$0 BYTE \$0 BYTE \$1	 0 0 0	;[c134] ;[c135] ;[c136] ;[c137]	register 0 reset peripheral interrupts
	BYTE \$0 BYTE \$4 BYTE \$0 BYTE \$0	4 4 1 0	;[c138] ;[c139] ;[c13a] ;[c13b]	register 4 1 stop bit, /16 clock divider register 1 disable all interrupts
	BYTE \$0 BYTE \$c BYTE \$0 BYTE \$e BYTE \$e	3 1 5 a f	;[c13c] ;[c13d] ;[c13e] ;[c13f] ;[c140]	register 3 rx 8 bit word, synch character load inhibit register 5 enable tx, RTS, DTR, tx 8 bit word end of table

Add Github CI to automatically check for disassembly errors.

🎮 giomba authored and giuliof committed 10 months ago 🧹 1/1

Characters generation



Standard 2732 (4 KiB) 8 pixel x 16 rows

SHS*E*ELEOH* ₽4 > T + \$ + @E Q, Q, Q, D, D, Q, H, S, E, X, E, X, E, S, E, F, G, K, U, !"#\$%&'()*+,-./ 0123456789:;<=>? @ABCDEFGHIJKLMNO PQRSTUVWXYZEN3^_ `abcdefghijklmno pqrstuvwxyz{!}~/ \$8800%%22\$60#f*«» **₩4**1⁄21⁄4 i – α βΥπΣόμζΦθΩδ∞ø∈∩ ≡±≥≤∫J÷≈∘∘+√η²∎ çβéèàùäëïöüâêîôû ÄÖÜÄ&-++H+++J--L

256 bytes ROM

old 28L22 narrow PDIP dump with custom hardware glue logic?



First "hacking" attempt



00000da0:	c622	daff	3e02	32d9	ff06	460e	20cd	15c7	.">.2E
00000db0:	7123	1019		c947	3c32		2ada		q#G<2*
00000dc0:		713a		cd95	c777		c723	22 da	. q:
00000dd0:	ff78	fe47		c9 3a			3c32		.x.G:
00000de0:					cd15	c772		c773	
00000df0:							c756		
00000e00:	c75e			0000	0000	0000	0000	0000	· · · · · · · · · · · · · · · · · · ·
00000ff0:	0000	0000	0000	0000	0000	0000	312e	3031	







From the board to the schematics



Lots of manuals and datasheets

Checking connections

Drafts and connection ideas

X-raying a multi layer board





- Four layer board
- Inner layers: maybe supply rails
- Tracks are all on the outer layers...
- ... but some of them hide under ICs!

Follow that wire!

How do we know if a path was already hit?

Just take notes on an actual photo using GIMP :)



What's inside a Sanco?

- Schematic is now on KiCad
- About 90% of the board is mapped and documented
- FDC circuitry is pretty much at early stage (help required!)





We'll say something about these four in a second

https://github.com/GLGPrograms/ceda-schematics

Memory map

- **64k** DRAM fully addressable with **bank switching** system
- Dedicated 4k video SRAM
- Further **4k SRAM**





Boot circuit

- Z80 boots from **0x0000**
- ROM isn't mapped there...
- A latch forces ROM to be mapped everywhere, just at boot!



0x0000	System ROM	Dynamie
	System ROM 	RAM
0xB000	System ROM	Alternate RAM
0xC000	System ROM	System ROM
0xD000	System ROM	Video RAM
0xE000	System ROM	Dynamic
0xFFFF	System ROM	RAM
	at boot	at runtime

Video Generation



Combinatorial network



	A7 /	A6	A5	A4	A3	A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1
2	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
3	0	0	0	0	0	0	1	1	0	0	1	0	0	0	1	1
4	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0
5	0	0	0	0	0	1	0	1	0	0	1	0	0	1	0	1
6	0	0	0	0	0	1	1	0	0	0	1	0	0	1	1	0
7	0	0	0	0	0	1	1	1	0	0	1	0	0	1	1	1
8	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0
9	0	0	0	0	1	0	0	1	0	0	1	0	1	0	0	1
A	0	0	0	0	1	0	1	0	0	0	1	0	1	0	1	0
в	0	0	0	0	1	0	1	1	0	0	1	0	1	0	1	1
С	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0
D	0	0	0	0	1	1	0	1	0	0	1	0	1	1	0	1
E	0	0	0	0	1	1	1	0	0	0	1	0	1	1	1	0
F	0	0	0	0	1	1	1	1	0	0	1	0	1	1	1	1
10	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0
11	0	0	0	1	0	0	0	1	0	0	1	1	0	0	0	1
12	0	0	0	1	0	0	1	0	0	0	1	1	0	0	1	0
13	0	0	0	1	0	0	1	1	0	0	1	1	0	0	1	1
14	0	0	0	1	0	1	0	0	0	0	1	1	0	1	0	0
15	0	0	0	1	0	1	0	1	0	0	1	1	0	1	0	1
16	0	0	0	1	0	1	1	0	0	0	1	1	0	1	1	0
17	0	0	0	1	0	1	1	1	0	0	1	1	0	1	1	1
18	0	0	0	1	1	0	0	0	0	0	1	1	1	0	0	0
19	0	0	0	1	1	0	0	1	0	0	1	1	1	0	0	1
1A	0	0	0	1	1	0	1	0	0	0	1	1	1	0	1	0
1B	0	0	0	1	1	0	1	1	0	0	1	1	1	0	1	1
1C	0	0	0	1	1	1	0	0	0	0	1	1	1	1	0	0
1D	0	0	0	1	1	1	0	1	0	0	1	1	1	1	0	1
1E	0	0	0	1	1	1	1	0	0	0	1	1	1	1	1	0
1F	0	0	0	1	1	1	1	1	0	0	1	1	1	1	1	1

this is it now, feel old yet?



256 bytes ROM

We could have surprised you with special effects... invert horizontal stretch vertical stretch



We could have surprised you with special effects...



horizontal stretch

vertical stretch



invert



FOSDEM 2024 FOSDEM 2024 FOSDEM 2024

OSDEM 2024

ceda2vga

Sanco 8003 a desktop computer







Boot from floppy

- We don't have any floppy for Sanco...
- We know a little about how a floppy is supposed to work...
- ... but we have the boot code
- ... and a CP/M disk image (thanks to bayo7 and pconseil)

; Boot trampoline executed when BOOT key is pressed bios_bootkey:

ld	de,\$0000	track = 0; sector = 0
ld	bc,\$4000	cmd = read (\$40); drive = 0
ld	hl,\$0080	load in \$0080
ld	a,\$01	
call	fdc_rwfs	invoke bios routine
ср	\$ff	check for error
jr	nz,bios_bootdisk	if ok, go on
out	(\$da),a	else, beep and retry
jr	bios_bootkey	

; if disk has been correctly copied into RAM, execute it bios_bootdisk:

ld	a,\$06
out	(\$b2),a
out	(\$da), <mark>a</mark>
јр	\$0080

; sound speaker beep ; run loader

https://archive.org/details/sanco-8003-cpm-2.2fr.dsqd https://github.com/GLGPrograms/ceda-cpm

Sanco CP/M floppy format

- ROM loads the "loader" from the boot track
- the "loader" loads the CP/M BIOS, BDOS and the CCP in RAM

tracks	80
double side	2
sectors per track	5
bytes per sector	1024

800k

• control is given to the CP/M, which finishes loading



Floppy-o-burner do it yourself

- Custom Z80 assembly code, small serial parser with read, write and format commands
- Python script that feeds the whole disk image, track per track, through serial





Keyboard





https://github.com/GLGPrograms/ceda-ps2-keyboard/



Emulator

CEDA-CEMU a Sanco emulator by RETROFFICINA GLGPROGRAMS

http://retrofficina.glgprograms.it/

Work in Progress

https://github.com/GLGPrograms/ceda-cemu

Working:

- Z80 (lib)
- Mode2 IRQ
- CRTC
- SIO/2 peripherals
- keyboard
- integrated monitor, breakpoints

Work in progress:

- PIO
- Floppy

Future:

- More tests
- More tests
- Did I already say more tests?

