Z80 Single Board Computer: Note e Diario di Lavoro

SAM Bellinzona 2016/2017
REF: Daniele Kamm
PIF: Naoki Pross

- 30.01.2017: PERCHÈ UNO Z80? -

Originariamente questo progetto era un esperimento per costruire una cartridge per il GameBoy Classic (DMZ-01) che conteneva dell'hardware aggiuntivo che avrebbe potuto interfacciare dell'hardware esterno con la CPU del GameBoy. Successivamente però il progetto si è rivelato più complicato del previsto a causa della complessa struttura del GB (GameBoy) e la difficoltà per ritrovare l'hardware stesso. Quindi sotto consiglio del docente ho cambiato il progetto in un Single Board Computer dato che sono interessato in informatica di basso livello e la CPU del GB era basata sul processore Z80 con un instruction set e assembly simile.

- 09.02.2017: HARDWARE -

Dopo una ricerca abbastanza intensiva dal magazzino della scuola abbiamo trovato i seguenti componenti principali del che utilizzerò per costruire il computer.

Z8400AB1 (Z80ACPU)	Zilog	x1	CPU
Z8420AB1 (Z80APIO)	Zilog	x1	Port Interface
Z8430AB1 (Z80ACTC)	Zilog	x1	Timer Clock
M28C64	ST	x2	EEPROM
HM62256B	HITACHI	x1	SRAM
TL16C550C	TI	x1	Seriale (UART / RS232)

Tutti gli altri componenti secondati come porte logiche e circuiti combinatori integrati saranno indicati in una lista finale nella documentazione riassuntiva.

- 13.02.2017: ADDRESS SPACE -

Come prima cosa dopo aver deciso il processore (Z80) è necessario definire l'address space per decidere come collegare l'hardware. Si vede chiaramente che la RAM usa la maggior parte dell'address space mentre la rom è solamente di 16KB, ma questo non è un problema perchè ho intenzione di aggiungere delle interfacce esterne per poter collegare dispositivi di memoria come per esempio le uSD. Dunque questa EEPROM vicina al processore sarà utilizzata unicamente per il bootloader e per un sistema operativo molto basilare.

- 06.03.2017: SCHEDE NORMATE E CLOCK SECONDARI -

In seguito ad una discussione ho deciso di implementare il circuito su un PCB di dimensioni standard, come per esempio l'Eurocard (IEEE 1101.10) in maniera da poter montare tutte le schede su un rack. Così facendo si avrebbe dei connettori standard sul retro che si potrebbero utilizzare per le periferiche esterne. Alternativamente si potrebbe utilizzare una struttura simile al PC/104 permettendo di interfacciare delle periferiche specifiche per computer ancora diponibili sul mercato. Il connettore standard per PC/104 è un header 32x2, quindi con 64 pin mappato come indicato sotto, con un opzionale estensione che può aumentare il connettore a 146 pins[1].

[1] http://pinouts.ru/Slots/Pc104_pinout.shtml

Pin	J1/P1	J1/P1
Number	Row A	Row B
0	–	–
1	IOCHCHK*	0V
2	SD7	RESETDRV
3	SD6	+5V
4	SD5	IRQ9
5	SD4	-5V
6	SD3	DRQ2
7	SD2	-12V
8	SD1	ENDXFR*
9	SD0	+12V
10	IOCHRDY	(KEY)2
11	AEN	SMEMW*
12	SA19	SMEMR*
13	SA18	IOW*
14	SA17	IOR*
15	SA16	DACK3*
16	SA15	DRQ3
17	SA14	DACK1*
18	SA13	DRQ1
19	SA12	REFRESH*
20	SA11	SYSCLK
21	SA10	IRQ7
22	SA9	IRQ6
23	SA8	IRQ5
24	SA7	IRQ4
25	SA6	IRQ3
26	SA5	DACK2*
27	SA4	TC
28	SA3	BALE
29	SA2	+5V
30	SA1	OSC
31	SA0	0V
32	0V	0V

- 07.03.2017: TASTIERA MISTERIOSA -

Nello stesso luogo in cui avevo trovato lo Z80 stesso ho trovato anche una tastiera con un connettore mai visto. Il connettore era composto da 19 pin, di cui 16 erano collegati all'interno della tastiera. Inizialmente pensavo che la tastiera contenesse già un driver che generava degli interrupt, come una qualsiasi tastiera moderna, con l'unica differenza che il bus di comunicazione era parallelo. Ma

dopo una ricerca rapida senza risultati ho deciso di aprire la tastiera per analizzare il layout. Inaspettatamente ho scoperto che l'intero PCB non è altro che una griglia 9x9 con i 18 fili che portano al connettore. Questo hardware probabilmente era per un C64 Commodore 64 che aveva il resto della logica programmata nella ROM o sulla scheda madre.

Per utilizzare l'hardware in questo stato è dunque necessario attivare i primi 9 bit del connettore e successivamente leggere i seguenti 9 mascherando il tasto interessato. Qui sotto ho preso un esempio che ho trovato online per dimostrare il concetto in assembly.

```
; This program waits until the key "S" was pushed.
; Start with SYS 49152

*=$c000                     ; startaddress

PRA  =  $dc00               ; CIA#1 (Port Register A)
DDRA =  $dc02               ; CIA#1 (Data Direction Register A)

PRB  =  $dc01               ; CIA#1 (Port Register B)
DDRB =  $dc03               ; CIA#1 (Data Direction Register B)


start    sei                ; interrupts deactivated

         lda #%11111111     ; CIA#1 port A = outputs
         sta DDRA

         lda #%00000000     ; CIA#1 port B = inputs
         sta DDRB

         lda #%11111101     ; testing column 1 (COL1) of the matrix
         sta PRA

loop     lda PRB
         and #%00100000     ; masking row 5 (ROW5)
         bne loop           ; wait until key "S"

         cli                ; interrupts activated

ende     rts                ; back to BASIC
```

- 16.03.2017: VISUALIZZARE E I DATI -

Avendo messo un “clock-stepper” probabilmente è utile poter anche visualizzare i valori esadecimali sul bus di dati e di indirizzi, dunque ho pensato di aggiungere dei display a 7 segmenti collegati alle 8 e 16 linee dei due bus,