Emulatore **emu8086**

guida introduttiva

Installazione

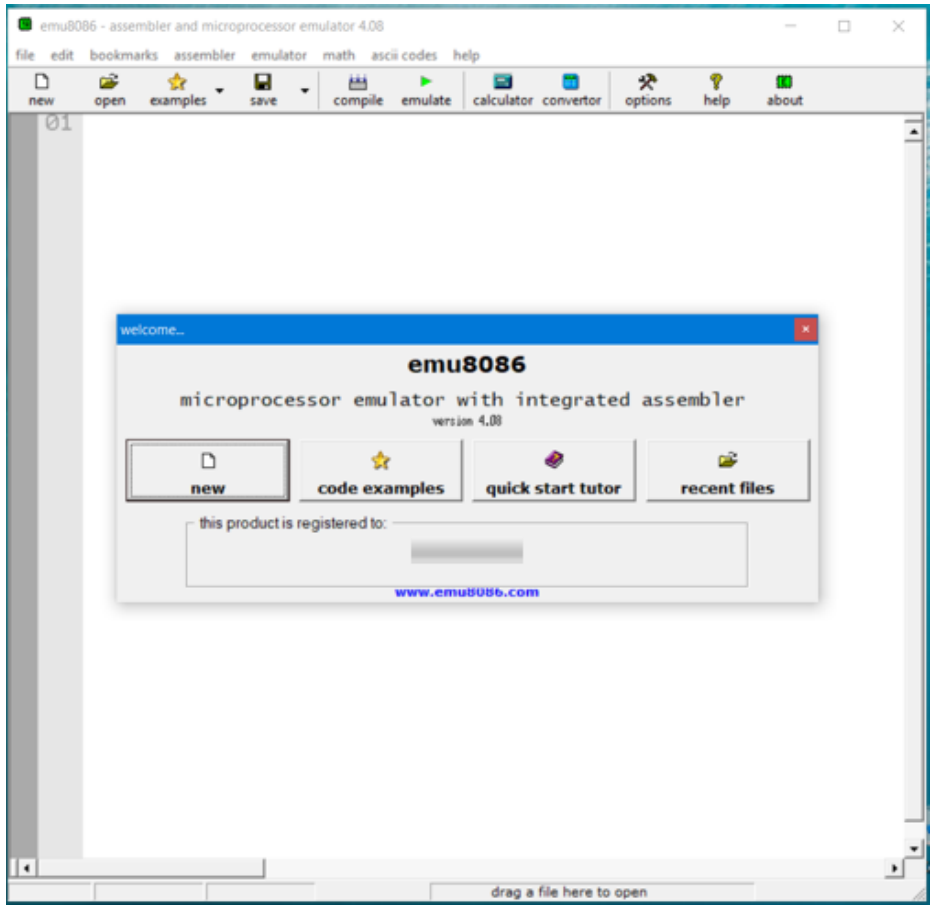
emu8086 può essere scaricato, gratuitamente per 40 giorni, dal sito https://download.cnet.com/Emu8086-Microprocessor-Emulator/3000-2069_4-10392690.html; s'installa come un qualsiasi software.
Dopo l'installazione compare sul desktop l'icona del programma.

Introduzione

Emu8086, è un emulatore del microprocessore 8086; è un programma IDE (Integrated Development Environment), ovvero ambiente di sviluppo integrato, che mette a disposizione dei programmatori:

- un **editor**, molto simile a "Blocco Note", tramite il quale si scrivono i file sorgenti (source code)
- un **assembler** per tradurre il source code in codice macchina (è compatibile con gli assembler Masm di Microsoft e Tasm della Borland)
- un **debugger** che consente di provare il codice macchina, step-by-step, fornendo controllo completo sui registri interni e con la possibilità di inserire un break-point
- **utility** come il converitore numerico, in grado di tradurre un valore numerico nei sistemi di numerazione nelle varie basi (2, 8, 10, 16), oppure una calcolatrice in grado valutare espressione con operandi rappresentabili nella varie basi
- interfaccia GUI (Graphical User Interface)

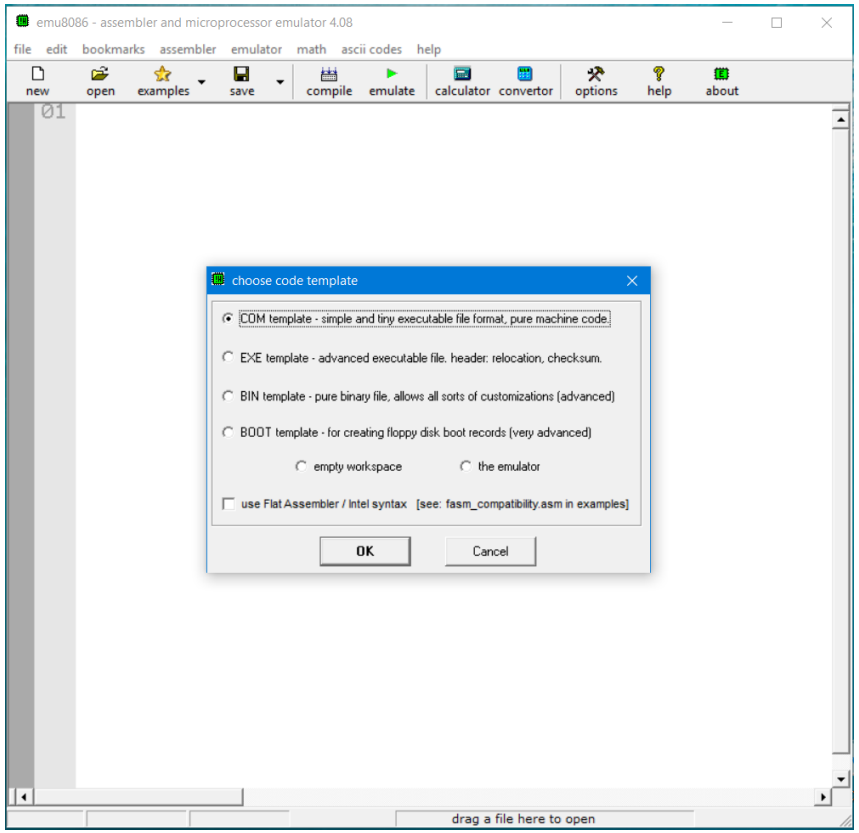
Avviamo l'emulatore



All'avvio comparirà la seguente schermata una finestra dove compaiono dei bottoni; con un click possiamo scegliere tra:

- **new** : quando si vuole scrivere un nuovo programma in assembly
- **code examples :** quando si vogliono considerare esempi (programmi) già pronti, da utilizzare come riferimento, o anche solo per provare l'emulatore
- **quick start tutor** : per consultare il manuale
- **recent files** : quando si vuole lavorare con un file chiuso di recente

Selezioniamo il modello (template)



Facendo click sul bottone **new** compare una schermata che consente la selezione del tipo di file che si desidera produrre.

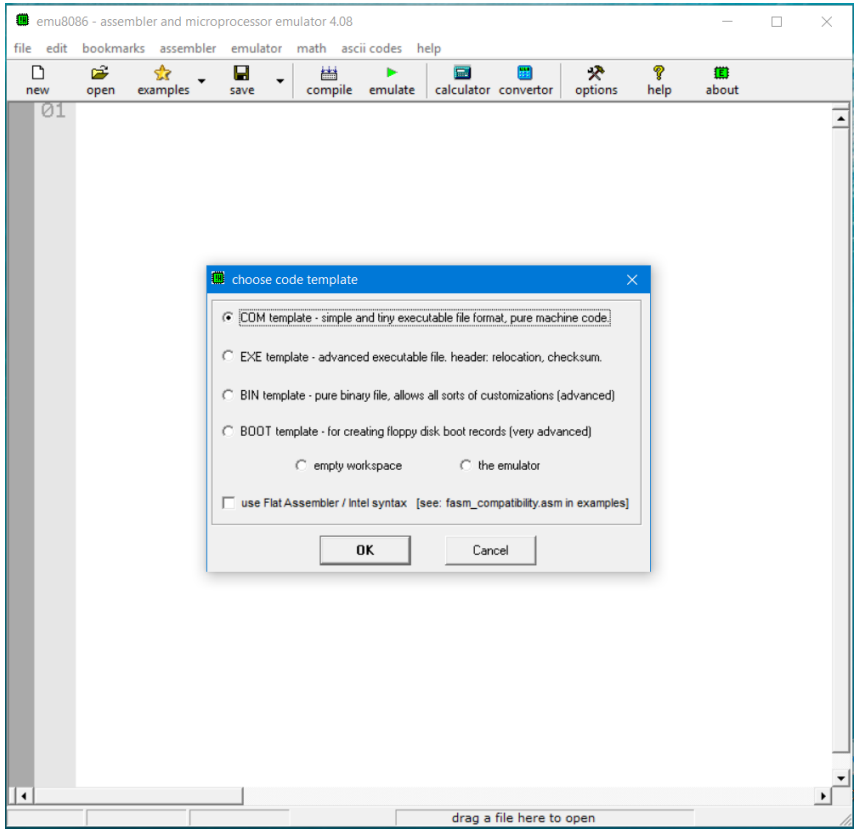
Selezionando **COM** compare nell'editor la struttura tipica (modello/template) pre-compilata di un file .COM, che completeremo e personalizzeremo con il nostro codice

Selezionando **EXE**, nell'editor compare la struttura tipica di un file .EXE

Selezionando **BIN** nell'editor comparirà una struttura che consente all'utente di decidere anche i segmenti di memoria fisica dove dovranno essere caricati i segmenti definiti dall'utente.

~~Selezionando **BOOT** si genererà un file da caricare nei primi 512 byte di un floppy, in modo da rendere il floppy stesso auto avviante (bootstrappabile).~~

Selezioniamo il modello (template)



Selezionando **empty workspace** si entra direttamente nella finestra di editor senza utilizzare alcun modello predefinito.

~~Scegliendo **use FlatAssembler/Intel syntax** si vuole produrre un file con l'assemblatore "Flat Assembler", prodotto da Grysztar, scaricabile e utilizzabile gratuitamente dal sito http://flatassembler.net/~~

Selezionando **the emulator** si entra direttamente nell'ambiente di emulazione per testare ed eseguire file eseguibili memorizzati su disco.

I file *modello (template)* che compaiono nella finestra di editor si trovano nella sottocartella **inc** della cartella d'installazione di emu8086. Questi file sono di tipo testo e quindi possono essere modificati per essere personalizzati.

empty workspace

Dopo aver selezionato **empty workspace**, la finestra dell'editor apparirà come nella figura 1. Dopodiché si passa alla scrittura del programma in assembly, figura 2.

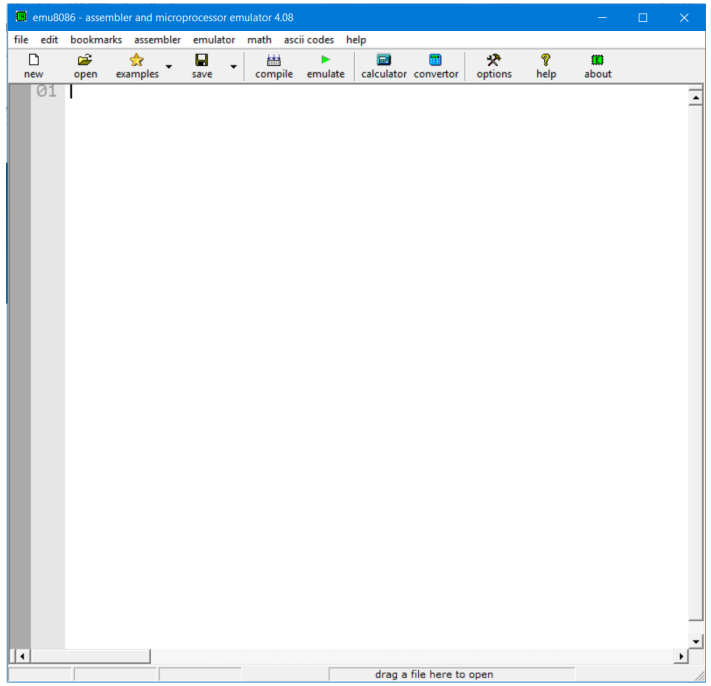


Figura 1

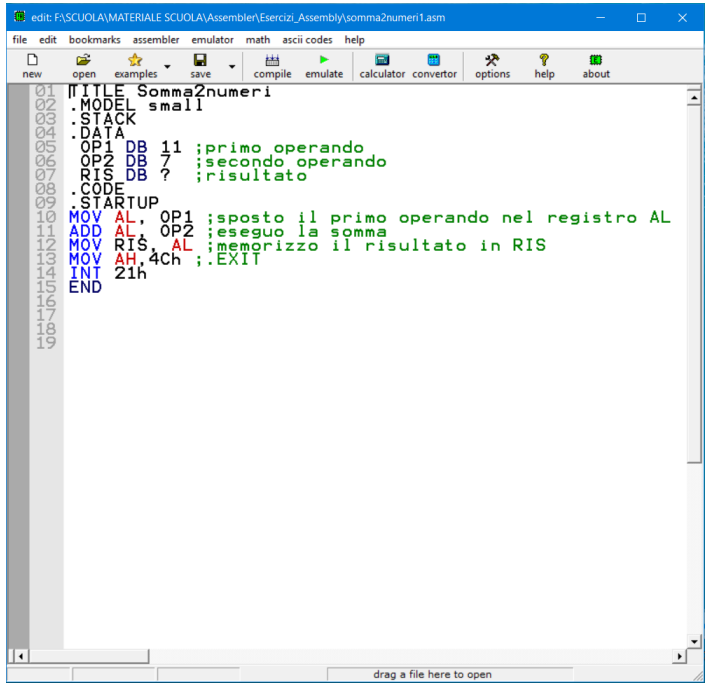
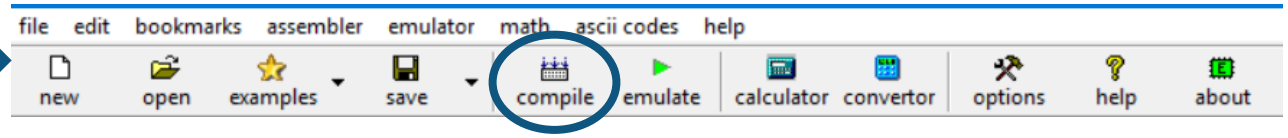
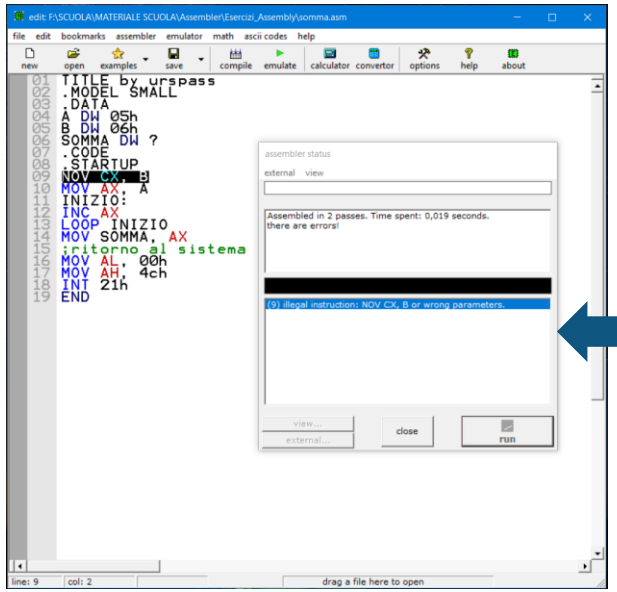


Figura 2

Compiliamo (assembliamo) il sorgente

Terminata la scrittura del sorgente si passa alla fase di compilazione, facendo click sull'icona **compile** della barra degli strumenti.

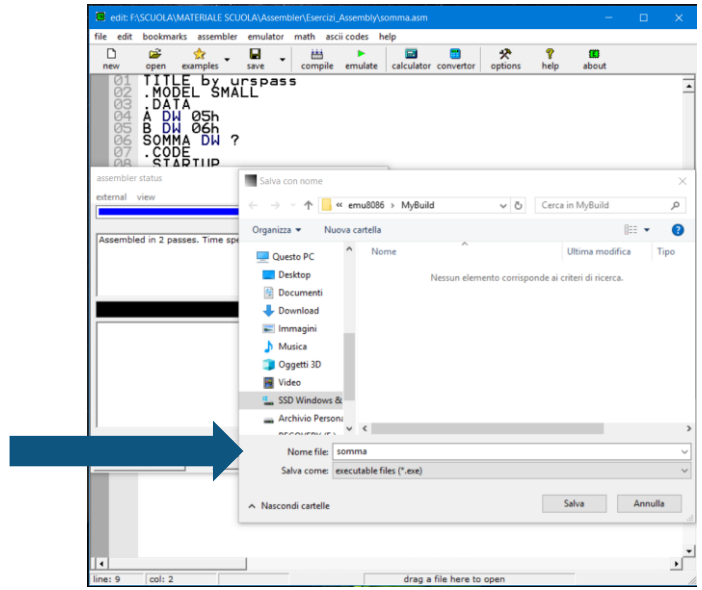




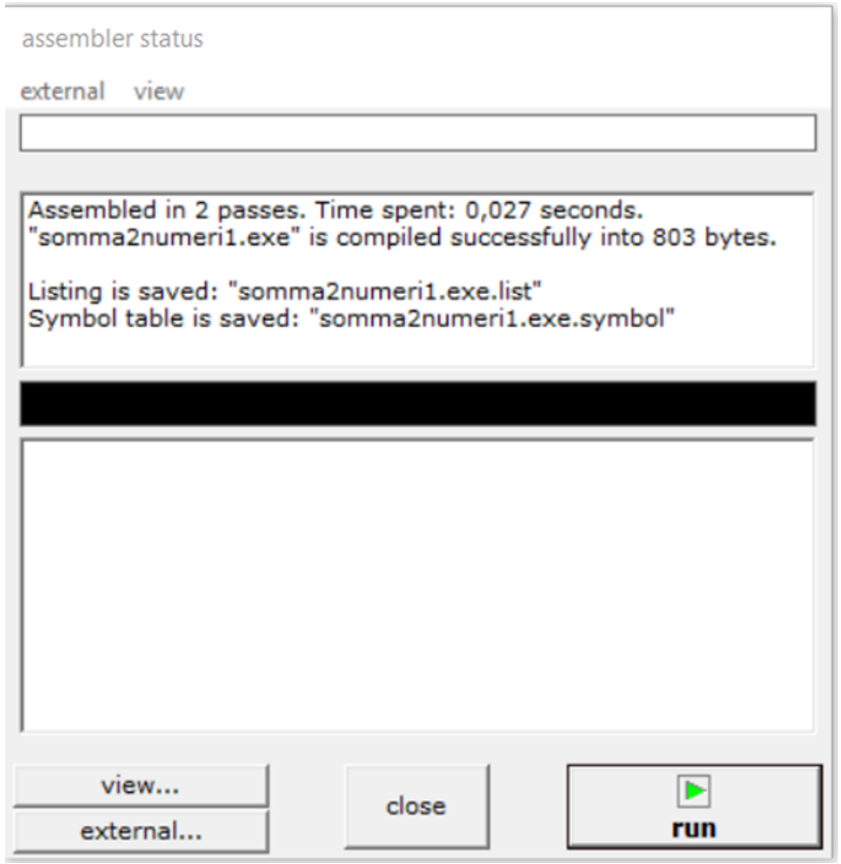
Compare una finestra che comunica il risultato della fase di assemblaggio nella finestra "assembler status".

Se avete fatto errori di sintassi, vengono segnalati nella finestra "assembler status" con indicazione del tipo d'errore e del numero di riga in cui è presente.

Se non ci sono errori di sintassi verrà richiesto dove e con quale nome memorizzare il file.



Compiliamo (assembliamo) il sorgente



Dopo il salvataggio, nella finestra assembler status si attivano alcuni bottoni:

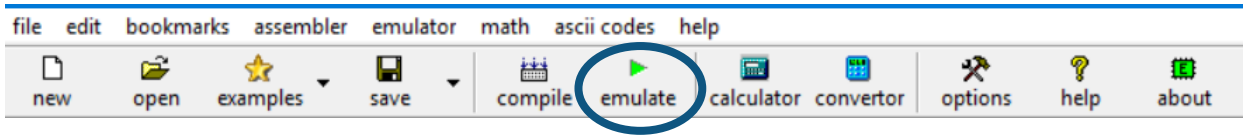
Facendo click sul bottone **view…** o **external…** si aprono dei menù a tendina con opzioni varie (listing, symbol table ecc).

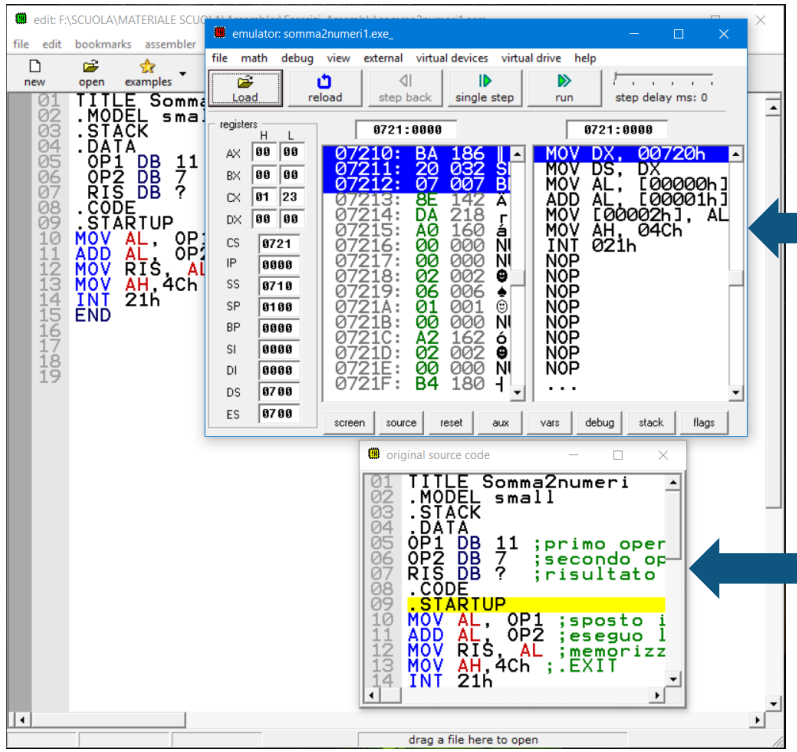
Se si fa click sul pulsante **run** si lancia il programma. Se tutto va bene alla fine dell'esecuzione appare una finestra con il messaggio che il processo ha ritornato il controllo al Sistema Operativo.

La scelta tipica, soprattutto quando il programma deve essere provato per la prima volta, è fare click sul bottone **close** in modo da tornare alla finestra principale.

Uso dell'Emulatore

Per caricare il codice eseguibile all'interno dell'emulatore basta fare click sul pulsante **emulate** nella barra degli strumenti.



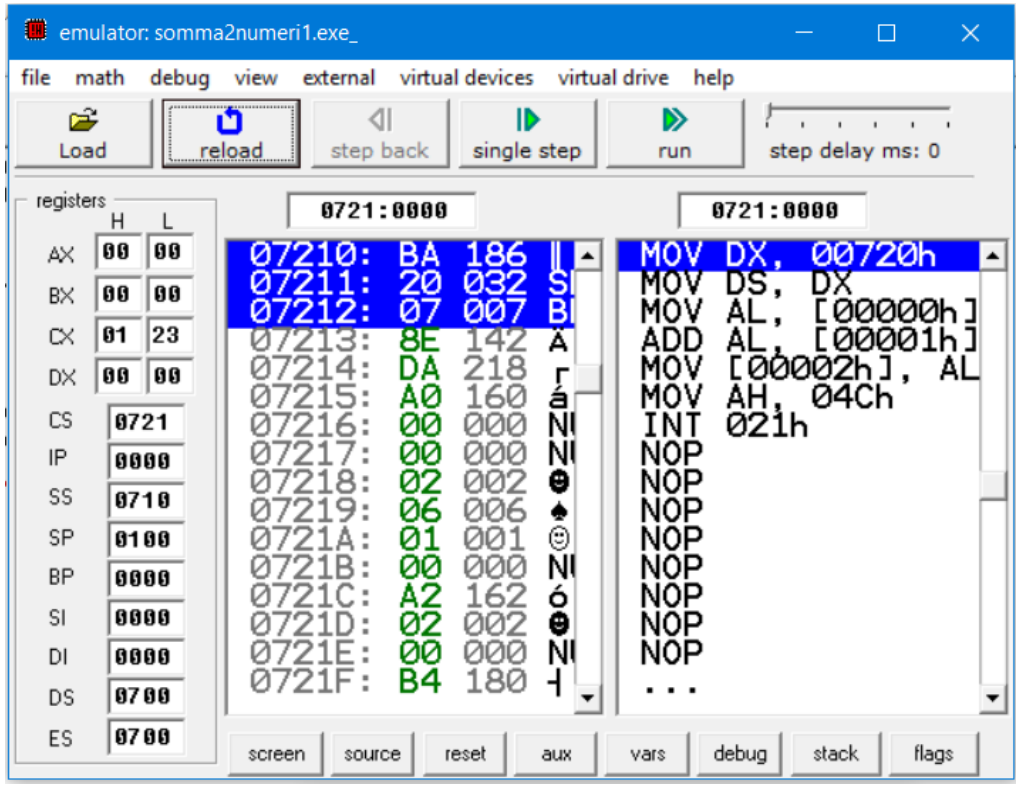


Cliccando si aprono 2 finestre:

- finestra dell'emulatore, divisa in 3 parti da sinistra verso destra: (1) il contenuto dei registri (2) il codice macchina (3) il codice disassemblato, cioè il codice sorgente, dove però le variabili sono sostituite con i relativi indirizzi di memoria, tutto espresso in esadecimale.
- finestra del codice sorgente: evidenzia la prossima istruzione che sarà eseguita, e consente all'utente di verificare se l'esecuzione delle istruzioni precedenti ha prodotto i risultati attesi. In questo modo è possibile individuare le istruzioni che non si comportano secondo le previsioni.

La finestra dell'emulatore

(pulsanti della barra superiore)



Load : carica un file eseguibile presente su disco

Reload : ricarica il file, per riportare tutto all'inizio

step back : torna indietro all'istruzione precedente

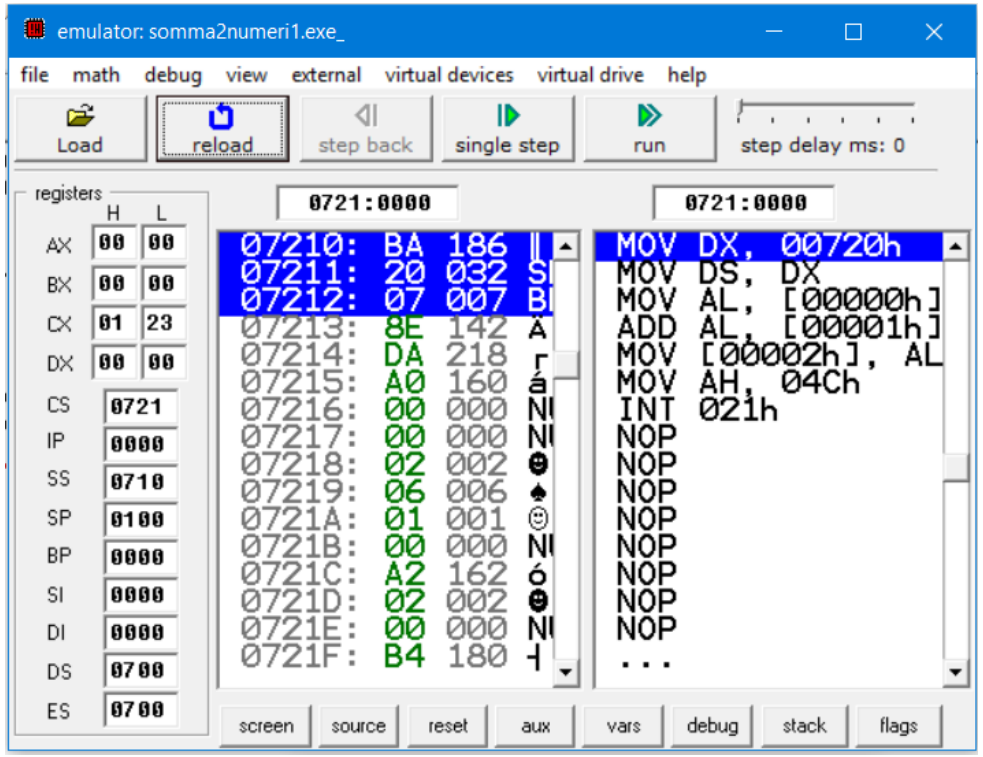
single step : esegue il programma passo-passo (un'istruzione alla volta)

run : esegue le istruzioni fino alla fine (o fino al break point). Dopo aver fatto click sul pulsante" run" esso assume la funzione (e l'immagine) di "stop"; **stop**: interrompe l'esecuzione del programma, utile in caso di loop infinito; se il programma attende un input da tastiera al posto di "run" compare la scritta "waiting for input", in giallo

step delay ms : imposta l'intervallo di tempo, in running, tra un'istruzione e la successiva; per default è impostato a 0 ms (free running)

La finestra dell'emulatore

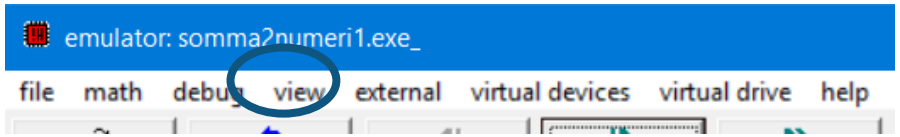
(pulsanti della barra inferiore)

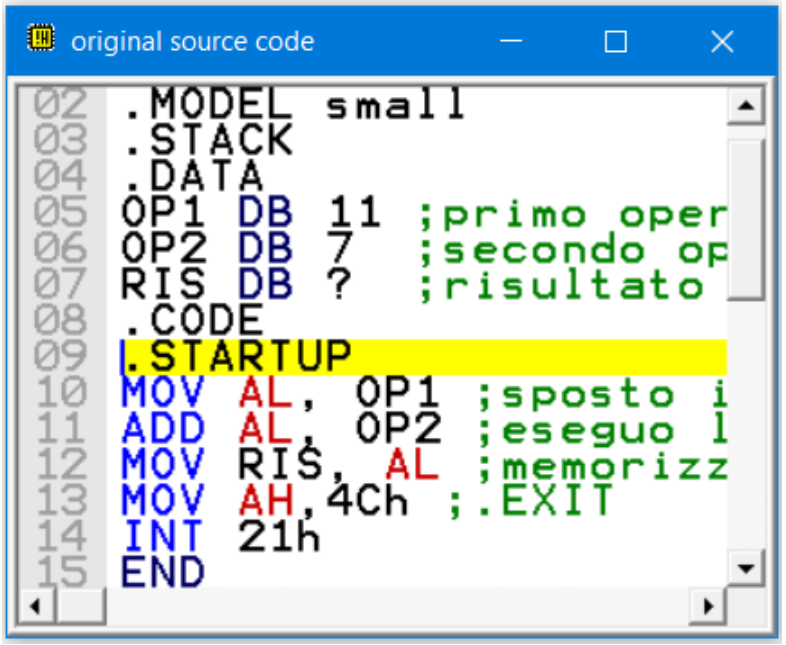


screen : mostra la finestra di input/output, dove l'utente può scrivere l'input e vedere l'output
source : mostra la finestra del codice sorgente
reset : un po' come reload
aux : cose "ausiliarie", come: la "symbol table", il listing, il contenuto della RAM, i registri di ALU e FPU, la condizione dinamica del breakpoint (stop on condition)
vars : visualizza e imposta le variabili del programma
debug : visualizza e consente di memorizzare in un file di log i valori dei registri per ogni istruzione eseguita, da quando si apre la finestra di debug a quando viene chiusa
stack : mostra l'area di stack
flags : mostra i flags e consente di modificarli

Breakpoint

Selezionando un'istruzione nella finestra *original source code* si può impostare un break point; basta selezionare, nel menu **debug** , la voce **set break point**.

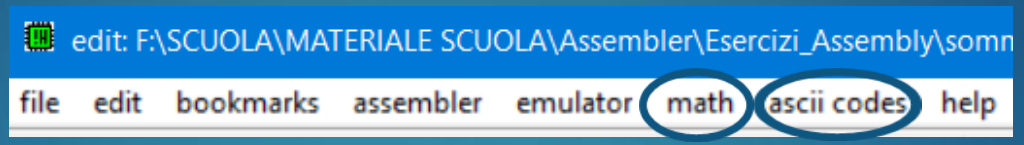




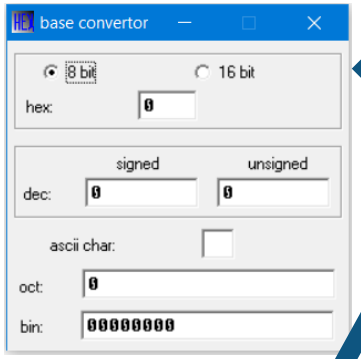
Un break point è un punto (di fatto un indirizzo) scelto dal programmatore in cui bloccare l'esecuzione del programma, a scopo di debugging/test. In questo modo il programmatore può verificare se l'esecuzione delle istruzioni precedenti ha prodotto i risultati attesi, consultando registri e memoria; è una procedura che si adotta in presenza di errori logici nel programma; gli errori sintattici sono scoperti dall'assembler.

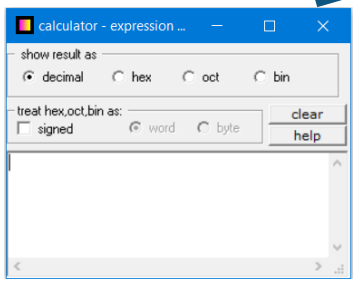
I break point così impostati sono permanent,i cioè se ricarichiamo il programma con reload ritroviamo i precedenti break point, agli stessi indirizzi. Per eliminare i break point selezionare dal menu **debug** la voce **clear break point**, mentre **show break point** serve per vedere i breakpoint impostati, e **run until** permette di impostare un break point temporaneo, valido solo una volta.

utility a disposizione



Durante la scrittura di un programma Assembly potrebbe servire convertire un numero in uno dei sistemi di numerazione, o conoscere il carattere ASCII associato ad un numero e viceversa.
Per questo l'emulatore mette a disposizione:





- Convertitore numerico: menu **math → base converter**
- Calcolatrice: menu **math → multi base calculator**.
- Tabella dei codici ASCII in decimale o esadecimale (fare click sull'area)