SimCPU v1.7

Note di Copyright

Il linguaggio Assembler SimCPU, il linguaggio macchina, il compilatore e tutto il materiale relativo sono stati realizzati persso l'Università di Udine dal Prof. Pier Luca Montessoro.
Il materiale aggiornato è disponibile al seguente indirizzo:

http://web.diegm.uniud.it/pierluca/public_html/teaching/free_software_project_italian.html

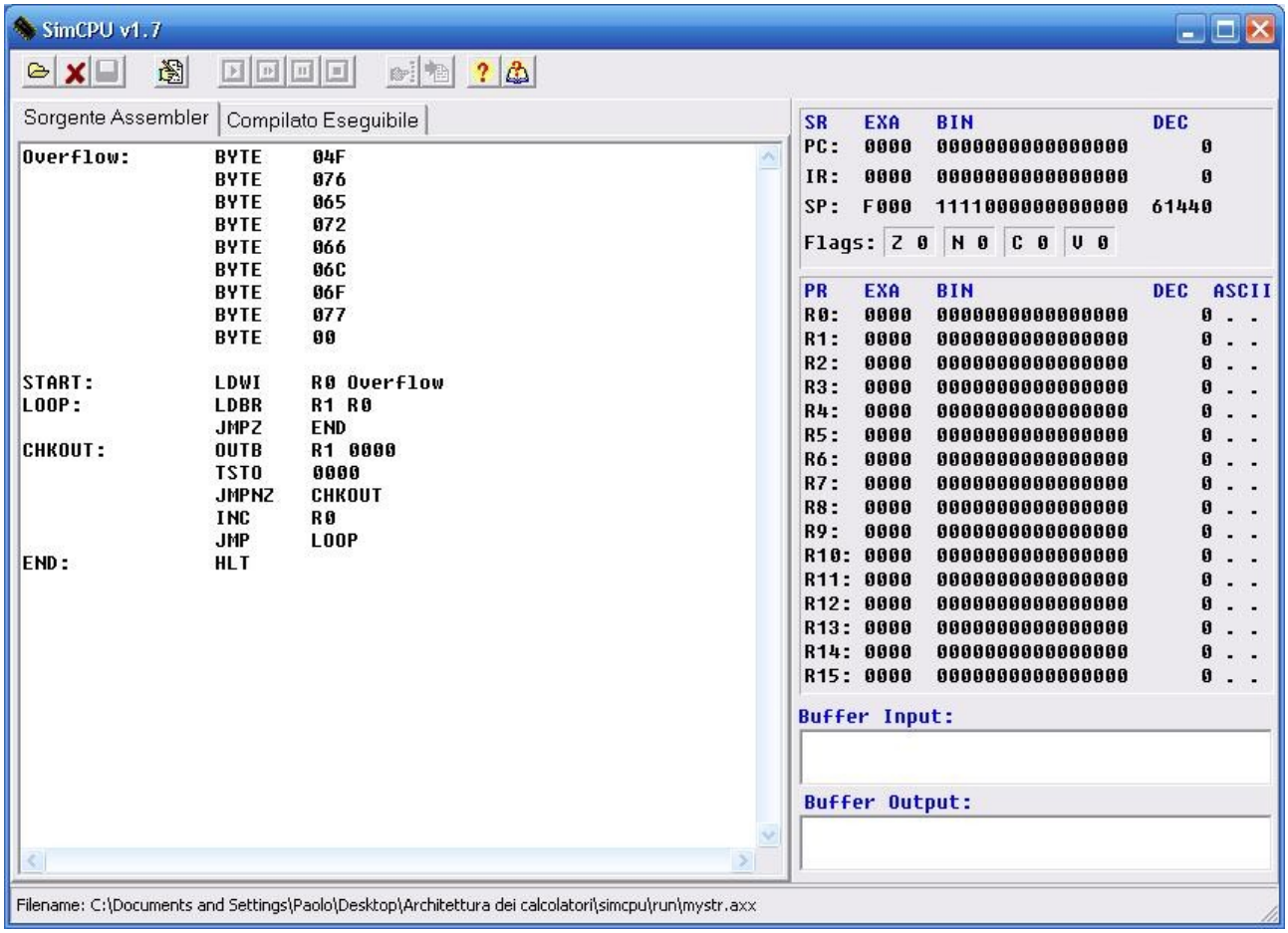
University of Udine
ITALY

montessoro@uniud.it
www.montessoro.it

L'interfaccia grafica ed il simulatore basato sul linguaggio SimCPU del Porf. Montessoro sono stati realizzati da Paolo Ferrara presso l'Università di Palermo.
paoloferrara@soluzioni-digitali.it
www.soluzioni-digitali.it

Uso dell'ambiente integrato

Inizialmente la schermata che si presenta all'apertura del programma è come quella riportata di seguito, ma vuota. Possiamo caricare un codice sorgente utilizzando la prima icona in alto a sinistra o possiamo scrivere direttamente il codice sorgente nell'editor.



Possiamo provare a scrivere il nostro codice sorgente iniziando a scrivere la label **Overflow:** Posizionando il cursore immediatamente dopo i due punti e premendo il tasto destro del mouse comparirà un menù popup con l'unica voce disponibile *"Inserisci stringa come byte"*. Selezionando questa voce comparirà la maschera di immisione della stringa.

Nella mashcera è possibile indicare se l'ultimo byte deve essere il codice terminatore di stringa che per convenzione è il codice ASCII 0.
Inserendo la stringa che si intende convertire e premendo il pulsante di conferma verrà inserita la sequenza di linee *byte / valore esadecimale* così come nell'esempio visto prima.

Completiamo l'esempio scrivendo il resto del codice.

Comandi:
In alto è presente la barra dei bottoni. Vediamo a cosa servono.

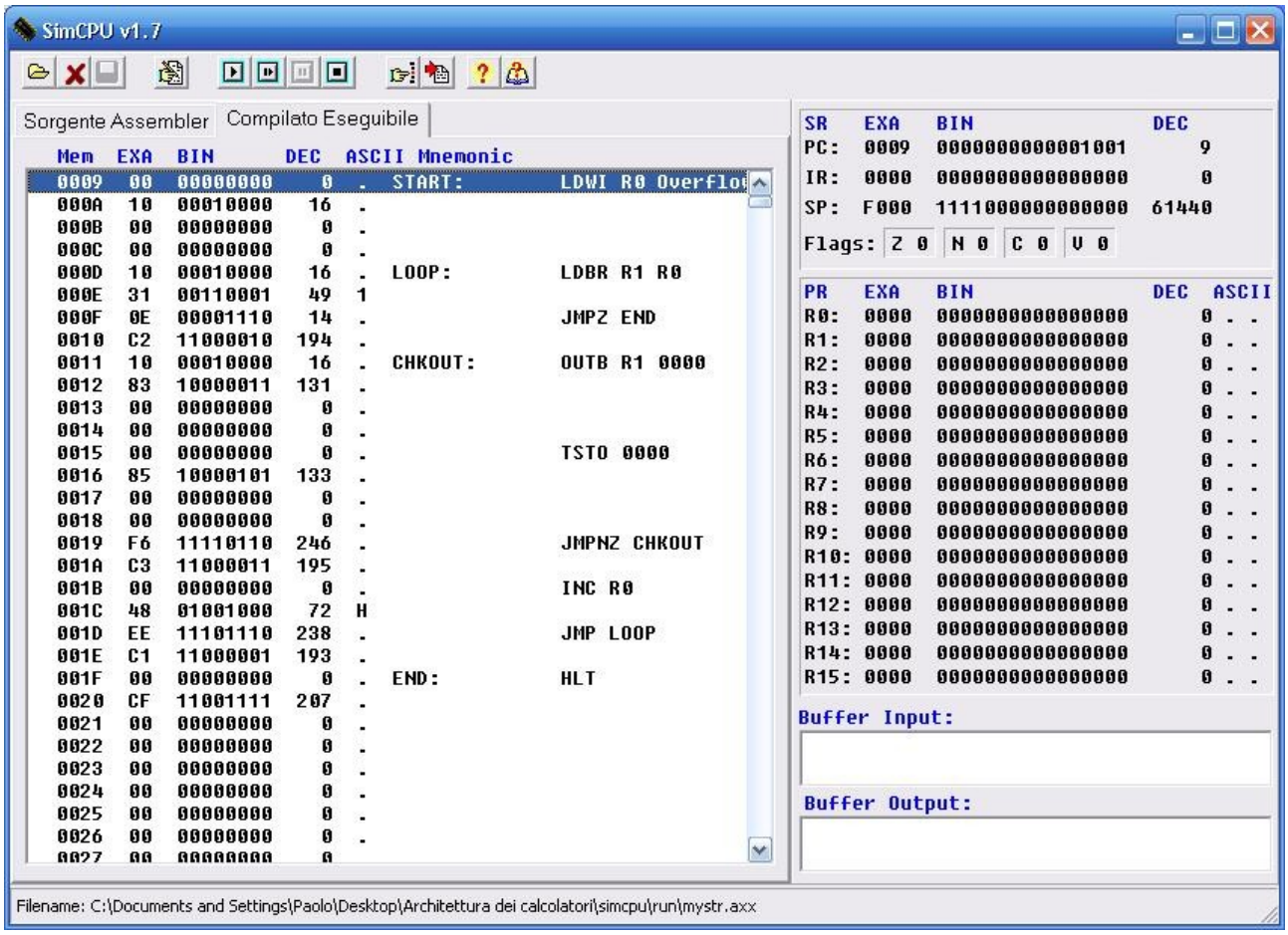
1) **Apri file:** Apre un file sorgente (.axx) precedentemente salvato. Se è presente un file sorgente nell'editor e vi sono state apportate modifiche viene chiesto se si intende salvare il file.
2) **Chiudi file:** cancella il file corrente. Se sono state apportate delle modifiche viene chiesta la conferma prima di eliminarlo.
3) **Salva file:** salva il file corrente sovrascrivendo l'originale oppure, se è un file appena creato, viene chiesto di dargli un nome prima di salvarlo.
4) **Compila:** avvia la compilazione del sorgente richiamando il programma di compilazione *assembler.exe* che assieme alla *dll* deve essere presente nella stessa cartella del simulatore SimCPU.
5) **Esegui singolo passo:** esegue una singola istruzione del programma compilato.
6) **Esegui programma:** immediatamente dopo la barra dei comandi comparirà un menù a tendina, inizialmente impostato su un tempo di 2 secondi, dal quale è possibile selezionare la velocità di esecuzione dei singoli passi. Le istruzioni verranno eseguite alla velocità impostata.
7) **Pausa:** attivo solo se in modalià "Esegui programma". Sospende l'esecuzione del programma. E' possibile ricominciare dallo stesso punto prenendo nuovamente "Esegui programma".
8) **Sposta su PC:** se scorriamo il listato e vogliamo tornare alla prossima istruzione da eseguire, premendo questo pulsante torneremo all'istruzione puntata dal Program Counter.
9) **Sposta su SP:** porta il cursore direttamente alla testa dello Stack in relazione al contenuto del registro Stack Pointer.

Compilazione:
Se durante la compilazione vengono riscontrati errori (proviamo a commetterne uno volutamente scrivendo LDWF al posto di LDWI) verrà mostrato un messaggio di errore (quello del programma assembler.exe) e, per quanto possibile, verrà selezionata la linea dove presumibilmente è presente l'errore.


```
START:      LDWF    R0 Overflow
LOOP:       LDBR    R1 R0
            JMPZ    END
CHKOUT:     OUTB    R1 0000
```

Purtroppo il programma assembler.exe in alcuni casi indica come errore l'ultima linea del programma (ad esempio se scordiamo di usare la label START:) per cui è consigliabile leggere il messaggio di errore per capire di cosa si tratta.

Se la compilazione va a buon fine allora verrà visualizzato il codice compilato e a questo punto saranno attivi i pulsanti per l'esecuzione.



Durante l'esecuzione del programma i registri cambieranno colore.
Un registro color rosso indica che il contenuto del registro è stato alterato dall'istruzione appena eseguita.

Un registro color blue indica invece che il registro non è stato alterato dall'istruzione appena eseguita, ma che l'istruzione ha sovrascritto il contenuto del registro con lo stesso valore.

Menù popup in fase di esecuzione:
posizionandosi su una linea e premendo il tasto destro del mouse si accede ad un popup menu che presenta le seguenti voci:

1) **Aggiungi Segnalibro:** inserisce la linea corrente nell'elenco dei segnalibri
2) **Vai al segnalibro:** contiene l'elenco dei segnalibri inseriti. Posizionandosi su uno di essi il listato verrà spostato fino a visualizzare la linea relativa al segnalibro. Questa voce contiene a sua volta una sottovoce (Delete) che permette la cancellazione del segalibro dall'elenco.
3) **Posiziona all'indirizzo...:** viene richiesto un indirizzo al quale spostare la finestra di visualizzazione.
4) **Breakpoint Toggle:** inserisce o toglie un breakpoint alla riga corrente.

I Breakpoint sono utili in fase di avanzamento veloce. Quando il Program Counter raggiunge una linea dove è stato inserito un Breakpoint, lesecuzione viene interrotta.