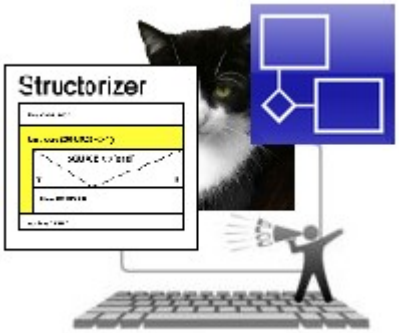
SALa macchine

Software per Attività di Laboratorio

2014.06



Indice

Di cosa si parla

Questi appunti coprono la descrizione dell'uso di tre programmi che possono essere di aiuto in fasi dell'apprendimento di aspetti diversi legati alla programmazione.

- **Structorizer**: (`http://structorizer.fisch.lu`) programma scritto in Java (e quindi eseguibile in qualsiasi piattaforma dotata di una JVM) che permette, in un ambiente visuale, di generare diagrammi di Nassi-Schneiderman ma, soprattutto permette di vederne l'esecuzione, anche passo-passo, in un ambiente tipo debugger. Si possono vedere i cambiamenti, frutto dell'esecuzione delle singole istruzioni, dei valori contenuti nelle variabili. L'ambiente permette anche di generare il codice, corrispondente al diagramma, in diversi linguaggi di programmazione.
- **Johnny simulator**: (`http://sourceforge.net/projects/johnnysimulator`) ovvero Simulation of Semplified Von Neumann Computer. Programma in sorgente FreePascal: nel sito è disponibile una versione compilata per ambiente Win eseguibile anche in ambiente Linux utilizzando Wine. Simulazione grafica del lavoro effettuato da una CPU durante l'esecuzione di un programma in un linguaggio Assembly semplificato. È possibile ampliare il set di istruzioni riconosciute dalla CPU del simulatore.
- **JDER**: (`http://www.ballini.it/Software/ProgER`) ovvero Java Diagrammi E-R è un software scritto, appunto, in Java (e quindi eseguibile in qualsiasi piattaforma dotata di una JVM) che consente di costruire, operando su oggetti grafici, un diagramma E-R specificandone tutte le caratteristiche e consentendone l'esportazione in formato grafico.

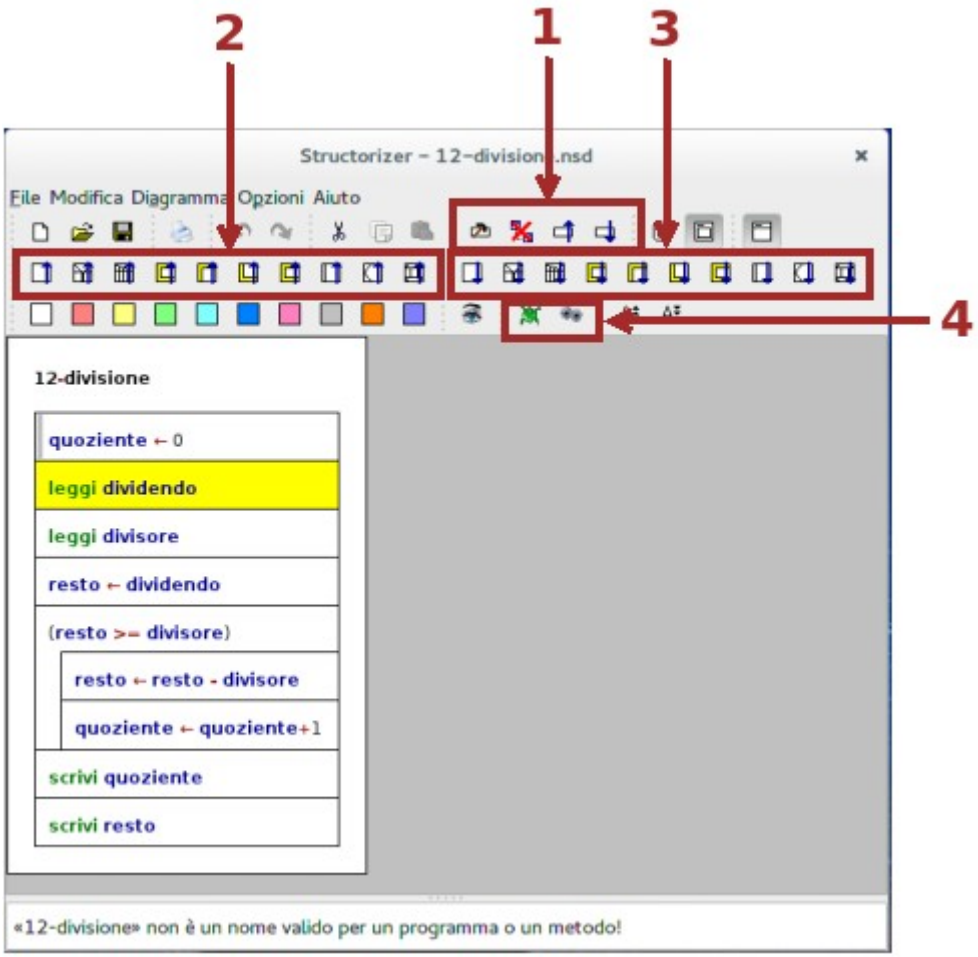
Ognuno dei software descritti fa riferimento a parti teoriche inerenti attività di programmazione di cui non si parla in questi appunti che trattano solo l'attività pratica di laboratorio. I materiali teorici sono presenti in dispense pubblicate nel sito dell'autore.

Per i diagrammi N-S si può fare riferimento agli appunti su C++, programmazione ed oggetti.

Per quanto riguarda il simulatore Johnny i riferimenti sono gli appunti sul funzionamento di un PC (PC inside ultima parte, propedeutica rispetto agli argomenti trattati in questi appunti) e sui linguaggi di programmazione (LinProg, parte iniziale sul linguaggio Assembly). Questi appunti, tuttavia, trattano dettagli non presenti nelle dispense citate e quindi possono essere intesi come integrazione di quelle.

I diagrammi E-R sono trattati negli appunti su DB&SQL.

Structorizer: interfaccia grafica, impostazioni



La prima fila in alto dei pulsanti, subito dopo quelli standard comuni a tutte le applicazioni, ne rende disponibili (**1**) quattro per operare sull'elemento selezionato (proprietà, elimina, sposta in alto o in basso). Le operazioni possono essere scelte anche dal menù contestuale disponibile in seguito alla selezione, con il tasto destro del mouse, dell'elemento.

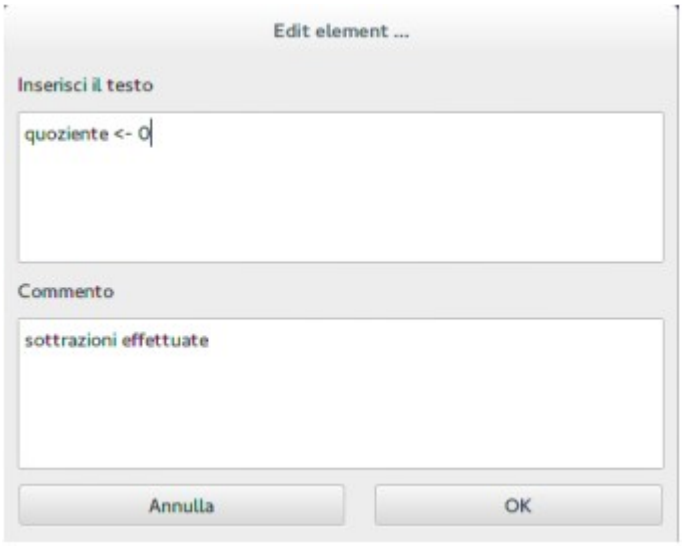
I blocchi **2** e **3** permettono di scegliere gli elementi da inserire, rispettivamente, prima o dopo quello selezionato.

Il pulsante a destra del blocco **4** permette di avviare l'esecuzione delle istruzioni comprese nel diagramma. Il pulsante a sinistra visualizza l'ambiente Turtle graphic (ambiente didattico elementare per la programmazione degli spostamenti di una tartaruga che disegna sullo schermo). Le istruzioni consentite sono elencate nel manuale del programma. I pulsanti sulla sinistra della stessa riga permettono di impostare colori diversi per gli elementi selezionati.

Il programma, quando si inseriscono gli elementi, è in grado di controllare il corretto utilizzo delle strutture sintattiche come anche dei nomi delle variabili utilizzate, visualizzandoli con opportuni colori: per esempio, nel diagramma riportato, le istruzioni *leggi* e *scrivi* sono in verde, gli operatori in rosso e le variabili sono in blu. In questo modo si possono evitare gli errori comuni di errata trascrizione. Affinché ciò sia possibile è necessario effettuare alcune impostazioni iniziali, indispensabili fra l'altro per l'esecuzione delle istruzioni. Le impostazioni accessibili dal menù *Opzioni* sono: *Strutture* e *Parser*.

Le prime opzioni permettono di impostare gli identificativi associati alle strutture fondamentali della programmazione. Le seconde consentono il **parsing** (*processo atto ad analizzare uno stream continuo in input ... in modo da determinare la sua struttura grammaticale* Wikipedia) del programma per permetterne la sua esecuzione. Nella seconda finestra si impostano anche i verbi per richiamare le operazioni di I/O.

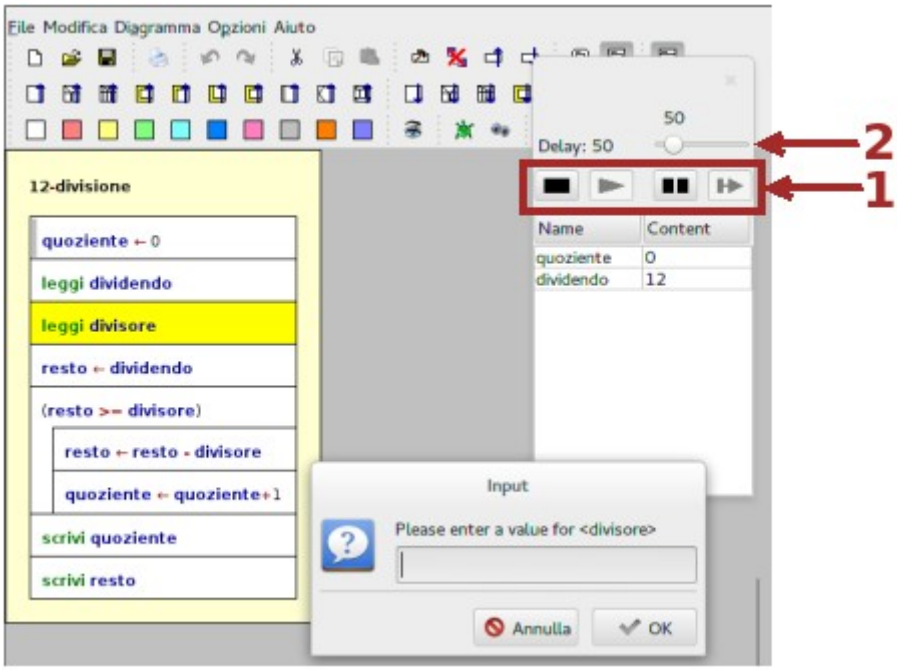
Dopo aver regolato le impostazioni si possono inserire gli elementi:



Quando si inserisce un elemento viene mostrata la finestra per l'impostazione delle proprietà. La finestra può essere visualizzata in qualsiasi momento selezionando con il doppio clic sinistro del mouse l'elemento, scegliendo dal menù contestuale o selezionando il pulsante dal gruppo **1**. Nella parte superiore bisognerà inserire il testo visualizzato dentro l'elemento, in quella inferiore, se si vuole, un commento. Quando presente il commento, l'elemento visualizza una striscia grigia sulla sinistra e il commento viene visualizzato non appena il mouse si porta sull'elemento. Per ogni inserimento di un nuovo elemento il diagramma viene ridisegnato, se necessario, per contenere gli altri elementi.

Structorizer, oltre a salvare utilizzando un formato proprio, può esportare il diagramma in formati grafici (PNG, PDF ...) o anche tradurre gli elementi in uno dei linguaggi di programmazione previsti. La scelta si può effettuare dal menù *File*, *Esporta Immagine* o *Codice*.

Structorizer: ambiente di esecuzione

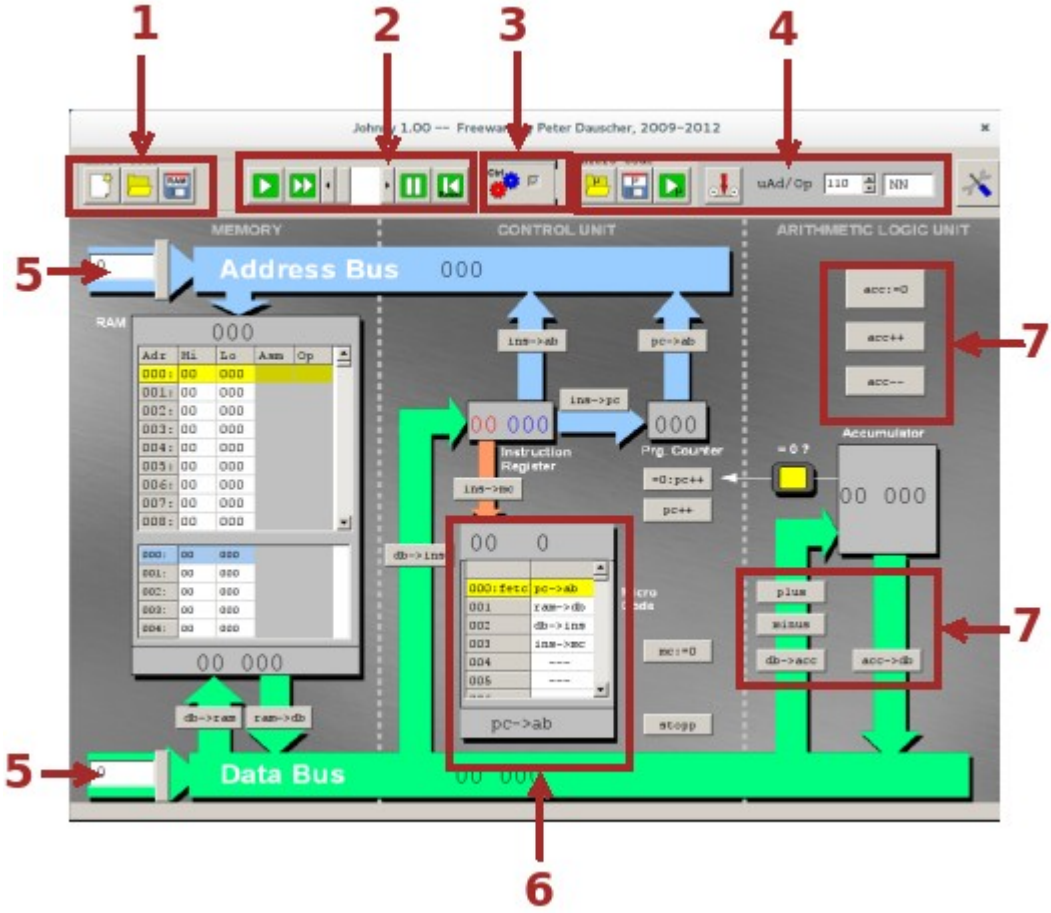


Avviata l'esecuzione viene visualizzata una finestra da cui se ne possono controllare le modalità. Nel gruppo di pulsanti **1** il secondo da sinistra avvia l'esecuzione: gli elementi vengono evidenziati mano a mano che l'esecuzione procede. L'esecuzione si può interrompere (primo pulsante a sinistra), avviare, gestirne la velocità (agendo sul cursore **2**), interrompere temporaneamente (terzo pulsante da sinistra del gruppo **1**) o eseguire un passo per volta (ultimo pulsante a destra del gruppo **1**).

Nella finestra di esecuzione sono visualizzati i cambiamenti nei valori delle variabili in seguito all'effetto prodotto dall'esecuzione dell'istruzione in questo momento evidenziata.

Johnny simulator: interfaccia grafica

La finestra dentro cui gira il programma è divisa in tre parti in ciascuna delle quali è visualizzata una componente che interviene durante l'esecuzione di un programma. Da sinistra verso destra: la Memoria Centrale, l'Unità di Controllo e l'Unità Logico-Aritmetica.



Il gruppo di pulsanti **1** serve per azzerare la memoria preparandola per un nuovo programma, portarne in memoria uno già salvato in precedenza o salvare su disco l'immagine del contenuto attuale della RAM.

Il gruppo di pulsanti **2** regola l'esecuzione del ***macro codice***, nel linguaggio Assembly del simulatore, di cui è composto il programma in RAM. In particolare, da sinistra, si gestisce: l'avvio di ogni singola macro istruzione, l'avvio in sequenza di tutte le istruzioni (la velocità di esecuzione può essere regolata agendo tramite il cursore successivo), la sospensione temporanea dell'esecuzione e, infine, il reset che porta il contenuto dei registri dell'Unità di Controllo e l'accumulatore dell'Unità Logico-Aritmetica allo stato iniziale.

Il pulsante **3** visualizza/nasconde i dettagli dell'Unità di Controllo: se si vuole osservare solo l'esecuzione del macro codice non si ha interesse a cosa succede *dentro* l'unità, cosa importante se si vuole osservare come viene tradotto (***micro codice***), nel senso di operazioni effettuate nei registri interni, il macro codice di cui è composto il programma.

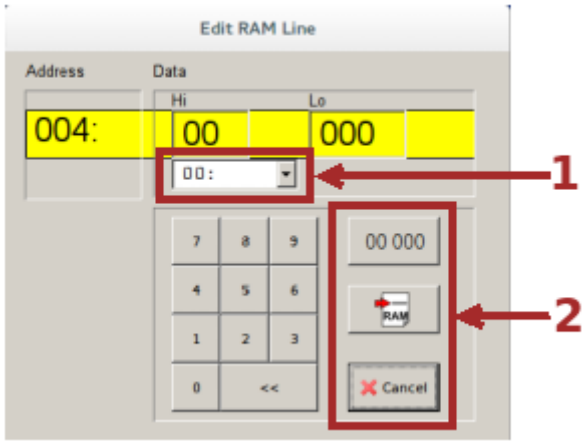
Se l'unità di controllo è visualizzata lo è pure il gruppo di pulsanti **4** che regola il micro codice: caricamento da disco di un set di micro codice, salvataggio, esecuzione passo-passo di una

istruzione del micro codice corrispondente alla macro istruzione di cui si richiede l'esecuzione. Il pulsante successivo permette di registrare le azioni che faranno parte della traduzione di istruzioni in macro codice personalizzato di cui si specifica il numero identifico e lo ***mnemonico*** (sequenza di caratteri che riportano alla memoria l'operazione descritta). Caratteristica, questa ultima, interessante in quanto consente di ampliare il linguaggio macchina del simulatore in modo da poter permettere la costruzione di programmi sempre più complessi.

Le caselle di controllo **5**, con i pulsanti successivi, permettono di forzare determinati valori, rispettivamente, nell'Address Bus o nel Data Bus.

La Memoria è divisa in due parti: in quella superiore si possono inserire i dati nelle locazioni prescelte. La parte inferiore mostra, durante l'esecuzione, la zona interessata dai dati e si può monitorarne il contenuto in tempo reale in sede di esecuzione.

Il clic sinistro del mouse su una locazione di memoria permette di inserire i valori:



Nella parte alta (**Hi**gh) della word individuata dall'indirizzo selezionato si può inserire uno dei codici operativi previsti dal linguaggio riconosciuto dal simulatore e che è possibile scegliere dal menù di selezione **1**. La parte bassa (**Lo**w) è destinata a contenere l'indirizzo di memoria oggetto dell'istruzione o un dato. I dati possono essere inseriti selezionando la parte dove si vuole inserire (Hi o Lo) e scrivendo direttamente o per mezzo del tastierino disponibile nella finestra. Il pulsante di mezzo di **2** scrive il dato nella locazione e chiude la finestra. Il primo pulsante in alto del gruppo azzera il contenuto della locazione, la funzione dell'ultimo pulsante in basso è intuitiva.

La zona dell'Unità di Controllo, oltre ai consueti registri, mostra il micro codice che descrive *come* le istruzioni vengono eseguite e quali parti sono interessate. Per esempio il codice corrispondente alla fase di *fetch* di ogni istruzione (codice 0) prevede:

```
pc->ab    ; il contenuto del Program Counter viene copiato nell'Address Bus
ram->db   ; il contenuto della locazione di RAM selezionata copiato nel Data Bus
db->ins   ; il dato dal Bus copiato nell'Instruction Register
ins->mc   ; dall'Instruction Register il codice seleziona, per l'interpretazione,
          ; la parte del micro codice corrispondente
```

Nella zona del'Unità Logico-Aritmetica oltre all'accumulatore sono contenuti alcuni pulsanti (**7**) che assieme a quelli contenuti nelle altre parti, permettono la micro programmazione della CPU.

Johnny simulator: programmazione

Rispetto al funzionamento di una CPU reale, il simulatore effettua alcune semplificazioni che in ogni caso non inficiano la comprensione del funzionamento di una CPU e, anzi, evitano complicazioni permettendo di concentrare l'attenzione sugli aspetti fondamentali:

- Normalmente una micro istruzione attiva una serie di segnali nel Control Bus. In Johnny invece corrisponde ad un singolo pulsante che può essere anche azionato manualmente dall'utente.
- Il micro codice è editabile ed è possibile espanderlo.
- L'Unità Logico-Aritmetica è provvista del solo accumulatore e manca il Registro di Stato.
- Tutti i codici si introducono in decimale. Gli indirizzi di memoria sono compresi nel range 0...999 e i dati che possono essere inseriti sono compresi nel range 0...19999. Non vengono gestiti overflow e underflow per cui: 3-5 = 0 e 19999+1 = 19999
- Il set di istruzioni è composto da 10 istruzioni e ogni istruzione permette soltanto indirizzamento assoluto nell'unico operando.

Lo pseudo-Assembly riconosciuto dal simulatore prevede, nella sua forma standard, le istruzioni:

Istruzione	*effetto*
`TAKE`	Il valore contenuto nella locazione di memoria specificata nell'operando, viene ricopiato nell'accumulatore.
`SAVE`	Il valore contenuto nell'accumulatore è ricopiato nella locazione di memoria il cui indirizzo è specificato nell'operando
`ADD`	Il valore dell'accumulatore cambia aggiungendo al contenuto precedente il valore contenuto nella locazione specificata nell'operando.
`SUB`	Come il precedente ma, questa volta, il valore nella memoria viene sottratto dall'accumulatore.
`INC`	Il valore contenuto nella locazione di memoria con indirizzo specificato dall'operando viene incrementato.
`DEC`	Come il precedente ma, questa volta, il valore viene decrementato.
`NULL`	La locazione di memoria il cui indirizzo è specificato nell'operando viene resettata (viene inserito il valore 0).
`TST`	(TeST) Se nella locazione specificata c'è un valore nullo, la prossima istruzione viene saltata.
`JMP`	(JuMP) Il programma continua dalla locazione di indirizzo specificato nell'operando.
`HLT`	(HaLT) Il simulatore visualizza un messaggio e il programma termina.

Come esempio di funzionamento dell'esecuzione di un programma si può considerare il programma che effettua la somma di due numeri già disponibile con l'installazione del simulatore e caricabile in memoria utilizzando l'apposito pulsante. Il programma prevede di sommare due numeri, che devono essere inseriti agli indirizzi 10 e 11, e fornisce il risultato dell'operazione all'indirizzo 12.

Di seguito, con l'aggiunta del commento per spiegare il senso delle istruzioni, è riportato tale codice:

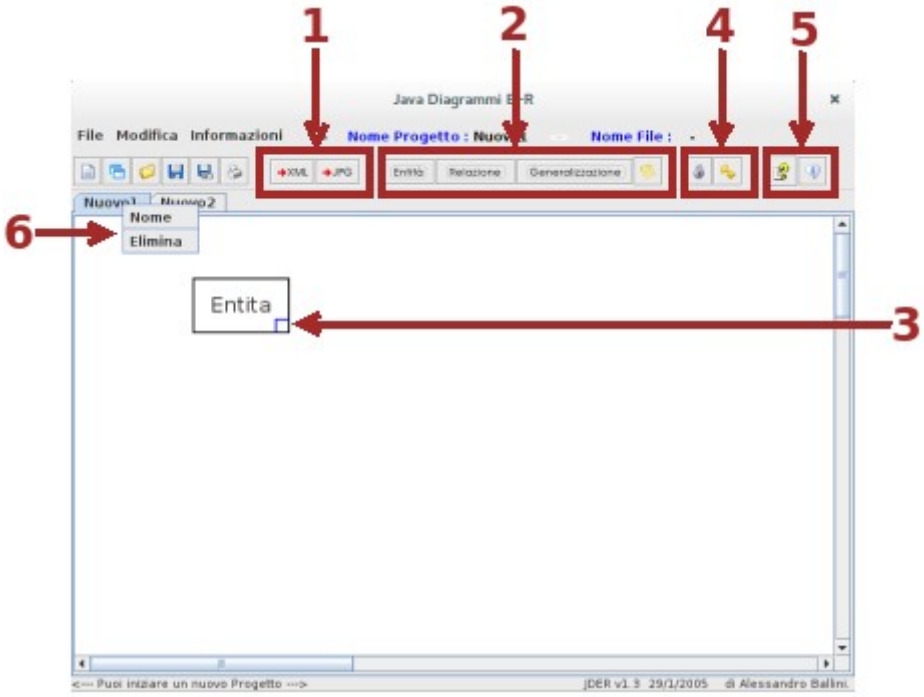
```
TAKE 10    ; mette nell'accumulatore il valore contenuto nell'indirizzo 10
ADD  11    ; somma al valore dell'accumulatore il contenuto della locazione 11
SAVE 12    ; conserva nella locazione 12 il valore dell'accumulatore
HLT   0    ; termina il programma
```

Si può osservare l'esecuzione del programma con le operazioni descritte nei commenti: il simulatore evidenzia con un lampeggio il cammino dei dati in funzione delle istruzioni. È interessante inoltre osservare l'interpretazione dell'istruzione seguendo l'esecuzione del micro codice associato. A titolo di esempio si riporta di seguito, commentato, il micro codice associato all'istruzione `ADD` (codice 20):

```
ins->ab    ; il contenuto dell'Instruction Register va nell'Address Bus
           ; (l'istruzione contiene nell'operando l'indirizzo del dato
           ; che si vuole sommare)
ram->db    ; il dato è disponibile nel Data Bus
plus       ; viene sommato al contenuto attuale dell'accumuletore
pc++       ; il Program Counter si aggiorna alla prossima istruzione da
           ; eseguire
mc:=0      ; si lancia il fetch della prossima istruzione. Il codice 0 è
           ; infatti il codice, come già esaminato, associato a fetch
```

JDER: interfaccia grafica e uso

JDER è un programma che consente facilmente di generare diagrammi E-R e di esportare il grafico, per esempio, in formato JPG.



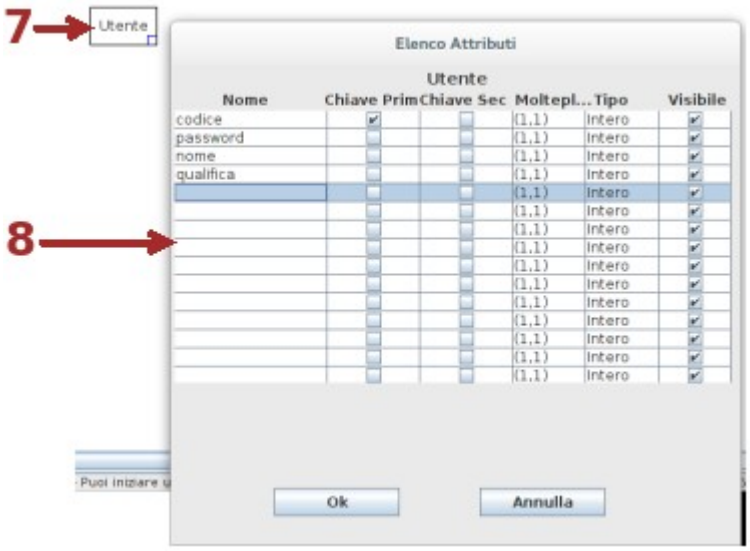
A parte il gruppo, a sinistra, di pulsanti standard presenti in tutti i programmi con interfaccia grafica e con funzionalità ben definite, i pulsanti **1** permettono la conservazione del diagramma in formati diversi.

Il gruppo **2** permette di inserire nel diagramma gli elementi fondamentali che presentano un indicatore (**3**) ad indicare la possibilità di spostare/ridimensionare l'elemento. I pulsanti **4** consentono di fissare/sbloccare la modifica degli elementi (l'indicatore **3** scompare quando il grafico è bloccato).

I pulsanti **5** permettono la visualizzazione del manuale del programma o di informazioni sull'autore del programma.

Con il menù contestuale (clic destro del mouse) sul tab (**6**) si può cambiare il nome.

Per mostrare l'uso del programma si supponga di voler disegnare il diagramma relativo al database degli accessi ai laboratori da parte di utenti: saranno definite due entità (Utente e Laboratorio) e una relazione Accesso. Gli attributi per ogni elemento potranno essere: Utente (codice, password, nome e qualifica), Laboratorio (codice, tipologia, postazioni), Accesso (data, ora inizio, ora fine).

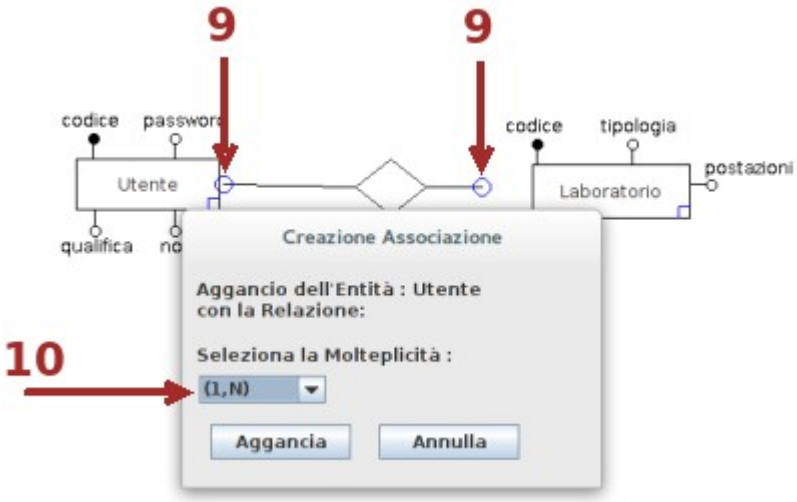


Selezionato il pulsante dell'entità dal gruppo **2**, si può attivare il menù contestuale per l'oggetto (clic destro) e scegliere *Nome* per cambiare il nome generico *Entità* in *Utente* (**7**) e poi selezionare, sempre dal menù contestuale, *Attributi...* (**8**) per specificare gli attributi interessati. Il *codice* sarà la chiave primaria.

Quando si esce dalla finestra di definizione degli attributi, tutte le definizioni risultano sovrapposte (sono tutte definite nella stessa posizione): bisogna trascinarle col mouse in modo che siano visibili chiaramente.

Allo stesso modo si può generare l'entità *Laboratorio* con definiti i relativi attributi.

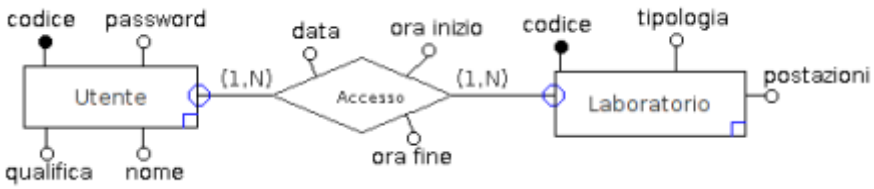
Ora bisogna creare una associazione fra le due entità.



Selezionato il relativo pulsante, dal simbolo occorre trascinare, con il pulsante destro del mouse, i rami della relazione (**9**) in modo che finiscano dentro l'entità con cui ci si vuole collegare e, dalla finestra visualizzata (**10**), scegliere il tipo di cardinalità e confermare. Trascinando con il pulsante destro del mouse si possono anche modificare le dimensioni del simbolo della relazione. Se la relazione dovesse prevedere più di due rami occorre visualizzare il menù contestuale (clic destro sull'elemento) e scegliere la voce più opportuna da *Rami...*

Anche nella relazione si possono scegliere il nome e gli eventuali attributi da inserire (possono essere posizionati in punti diversi scegliendoli dall'ultima colonna degli attributi).

Il risultato finale dovrebbe essere:



L'ultima cosa da fare è fissare tutto in modo da fare scomparire i punti che congiungono i rami della relazione con le entità e gli indicatori di ridimensionamento nell'angolo in basso a destra delle entità e il diagramma finale può essere esportato come immagine.