I Manuali di Giobe2000

TUTORIAL ASSEMBLER

# Nuovo Ambiente Assembler

## Come si utilizza
## con l'editor WinAsm Studio 5.1.70 Full Package
## di *Antonis Kyprianou, Winasm.net Team*



Studio Tecnico ing. **Giorgio Ober** contatto@giobe2000.it

Questa ***Monografia*** può differire in parte dalla versione *on-line*
soggetta a probabili aggiornamenti e integrazioni.

Verifica sempre le eventuali novità direttamente sul Sito

# Come si utilizza
## con l'editor WinAsm Studio 5.1.70 Full Package
## di Antonis Kyprianou, Winasm.net Team

Per accedere alla creazione di un *progetto ASM* con il servizio di ***editor*** assicurato da **WinAsm Studio 5.1.70 Full Package** [Copyright **Antonis Kyprianou**, **Winasm.net Team**] basta un ***doppio-click*** con il *tasto sinistro del mouse* sulla seguente **iconcina**, scelta tra quelle disponibili nella cartella principale del **Nuovo Ambiente Assembler**, **C:\Arch-Lab**=%SystemDrive%**\Arch-Lab**:

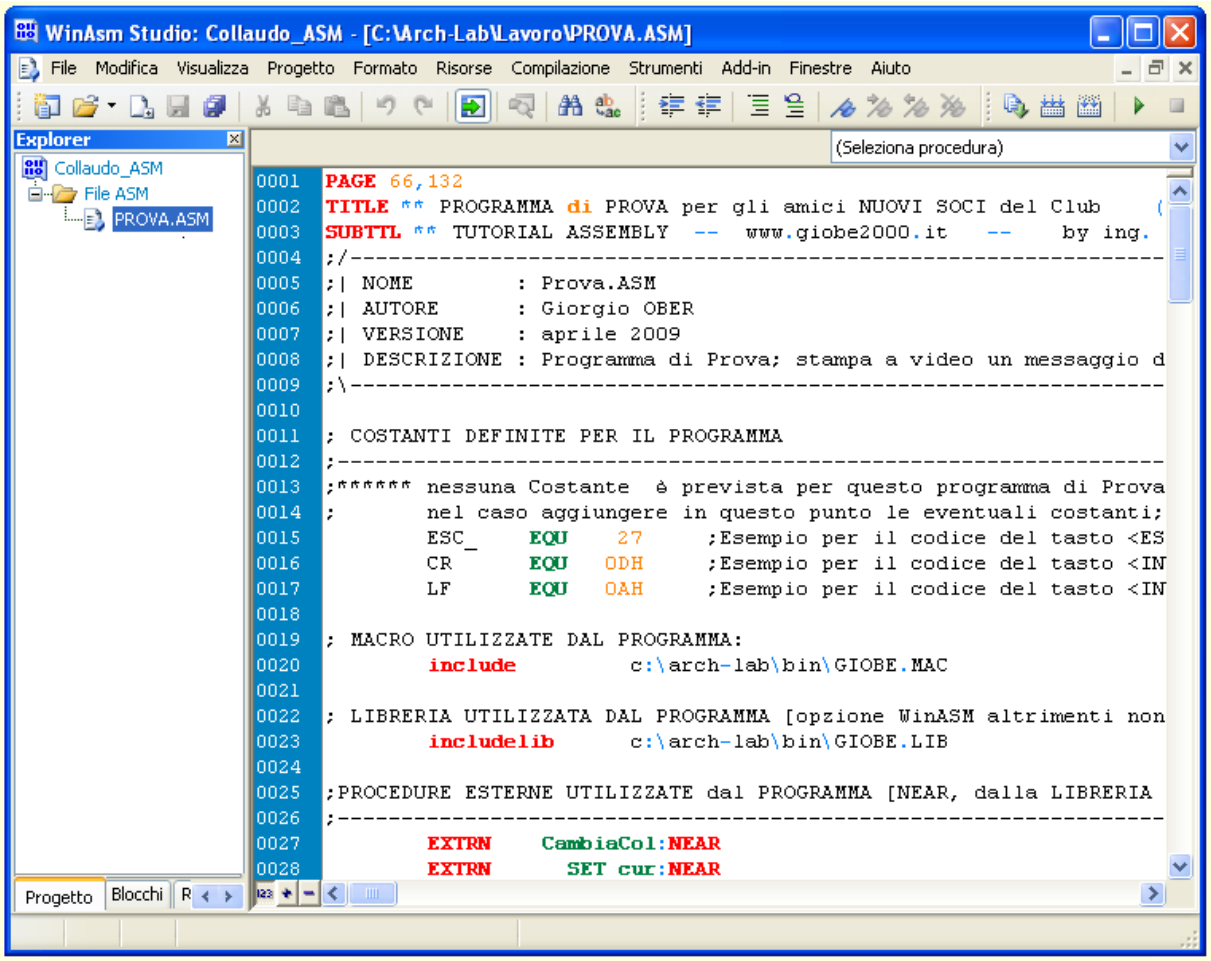

Anche l'***editor*** di **WinAsm Studio** è stato pazientemente configurato per rendere automatica la fase di sviluppo di un eseguibile a partire dal sorgente ASM; la scelta di includerlo tra le possibili modalità d'ingresso al mio **Ambiente** è legata al fatto di essere *intrinsecamente legato* alla programmazione del linguaggio Assembly, come traspare dal suo nome.

**WinAsm Studio** è, in realtà, di una ***struttura*** **freeware** pensata per organizzare un **IDE** (***I**ntegrated **D**evelopment **E**nvironment*); ***anche gli altri*** editor si prestano benissimo per creare un *ambiente integrato* per lo *sviluppo* di programmi, ma **WinAsm** è nato appositamente per il supporto alla ***programmazione Assembly***, sia quella **DOS a 16 bit** (da noi praticata in questo sito e *per la quale* [**NB!!**] *è stata configurata la versione in dotazione*) che quella **Windows a 32 bit** (per la quale esiste una notevole dotazione di strumenti, creati dagli utenti stessi, pensati specificatamente per la creazione di programmi basati sulle *API Win32*).

Si tratta quindi di una scelta ideale, nella prospettiva di passare, prima o poi, a livelli più evoluti.

Dotato di una *buona e chiara interfaccia grafica*, supportata da *menu in italiano*, è *configurabile* in molti dettagli (tra i quali capacità di interpretare le parole chiave dell'**assembly** con colori diversi) ma ***sembra eccellere*** proprio nella versatilità di integrare le strutture tipiche di un IDE, cioè assemblatore, linker, debugger e così via.

Non appena si clicca sull'iconcina l'***editor*** di **WinAsm Studio** si apre mostrando questa immagine:




Faccio notare che nella colonna di sinistra viene mostrato il *metodo di default* per accedere ai files da coinvolgere con questo programma: quella affidata ad un "***progetto***", una tecnica che rende disponibili *in un unico raccoglitore virtuale* tutte le risorse necessarie, anche se fisicamente localizzate nelle più svariate cartelle del nostro disco; data la filosofia di **WinAsm** (cioè il suo ***efficientissimo*** modo di gestire le cose, come detto, tipico di un **IDE**) è meglio rassegnarci all'idea e stare al gioco: se vogliamo utilizzare questo metodo *per ogni lavoro* che porteremo a termine *ci verrà chiesto* di definire un progetto.

Se la cosa non ti va bene puoi utilizzare l'**Ambiente Assembler** con **ConTEXT** (.. o con **PSPad**): è solo una questione di abitudine, probabilmente ricambiata dalla più volte ribadita potenza di **WinAsm Studio**; di certo se **apri un file sorgente ASM** (a partire dal menu in alto, ***File*** > ***Apri File..***) esso sarà correttamente visualizzato nella parte destra del piano di lavoro ma non sarà possibile sottoporlo alle azioni del IDE (compilazione, linker, debugger, ..), per cui in questo modo sarà trattato come un normale file di testo e la cosa non ci può minimamente interessare.

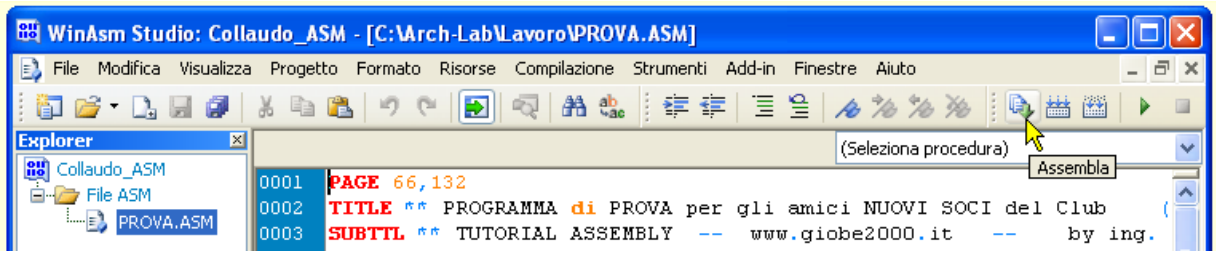
Per questa ragione ho comunque predisposto un *nostro* progetto, chiamato ***Collaudo_ASM***, contenente (per ora) un solo file (quello *di prova* classico, **PROVA.ASM**). La creazione e la gestione di questa *azione coordinata* è supportata da appositi files con estensione ***.wap***, memorizzati nell'**Ambiente** (nel nostro caso con il nome ***Collaudo_ASM.wap***) direttamente nella *nostra* tipica cartella di lavoro, **C:\Arch-Lab\Lavoro**=%SystemDrive%**\Arch-Lab\Lavoro**. [Per non perdere il filo del discorso vedremo *in fondo a questo documento* come fare per **creare un nuovo *progetto*** da zero..]

L'ambiente organizzato da **WinAsm Studio** è **straordinariamente attento** ad ogni dettaglio legato alla realizzazione di **progetti assembly**: la configurazione del cosiddetto ***progetto*** prevede numerose varianti, aperte anche alla programmazione **Win32**. Per quanto ci riguarda la scelta è quella di realizzare un **Programma DOS**, per il quale sono previste due varianti, una destinata a generare *eseguibili di tipo* **COM** e l'altra destinata a generare *eseguibili di tipo* **EXE**.

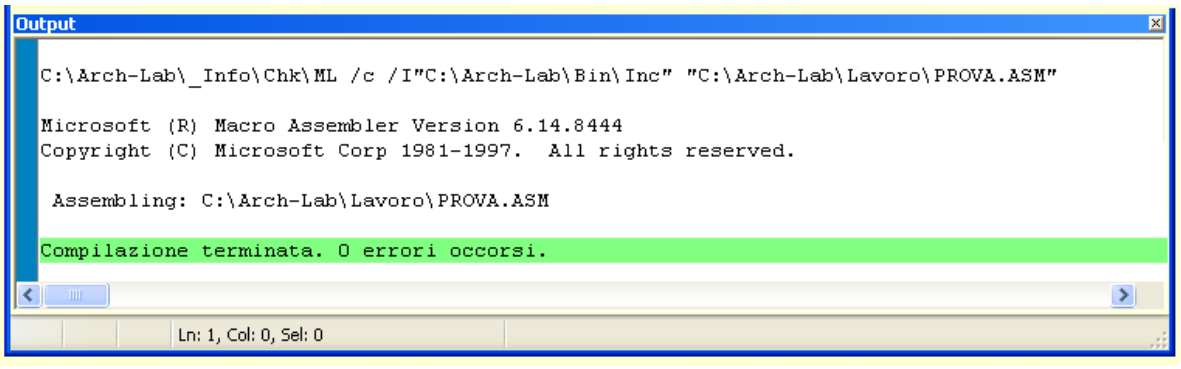
**Fai bene attenzione**: per una precisa scelta didattica l'ambiente in dotazione è stato configurato per generare *eseguibili di tipo* **COM**; naturalmente è facilissimo passare all'opzione destinata a generare *eseguibili di tipo* **EXE**, come descritto nel documento che si occupa della **messa a punto** di questo metodo.

**Dunque**: poichè tutto è già ***predisposto e funzionante*** puoi tentare immediatamente la tua prima compilazione; il *piano d'editazione* è pronto con un sorgente ASM affidabile (di nome **PROVA.ASM**); guardando l'immagine già puoi capire come verrà trattato il testo presente nel documento, evidenziandone ***a colori*** ogni *dettaglio sintattico e funzionale*; con la *barra di scorrimento* laterale (a destra) puoi scorrere tutto il sorgente e cercare di capire la sintassi necessaria per un progetto assembly: per aiutarti in questa analisi ognuno dei sorgenti offerti dal mio sito è ***sempre ampiamente commentato***.

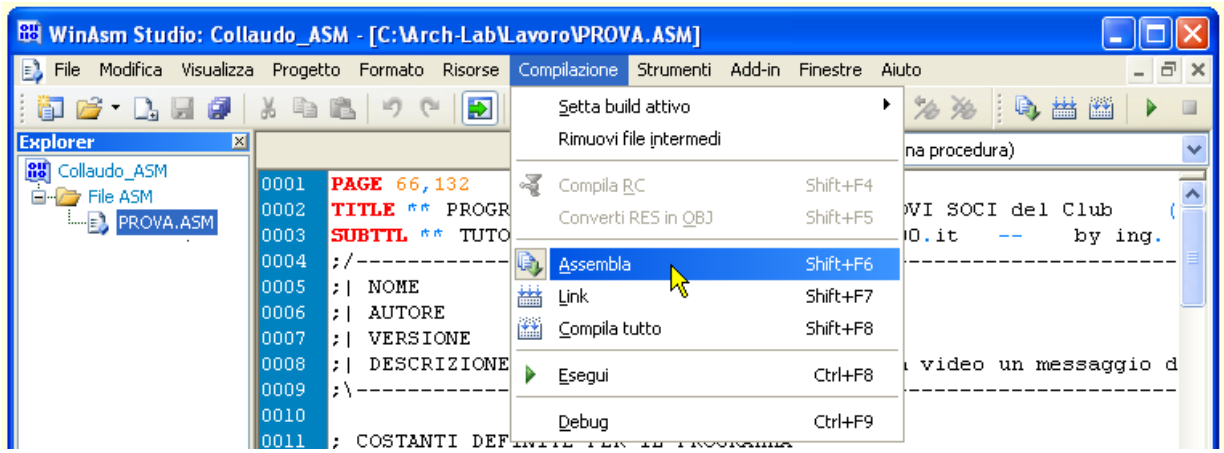
Se sei sufficientemente preparato puoi tentare di modificare qualche istruzione ma prima è conveniente imparare a **rendere eseguibile il sorgente**, così com'è; per provvedere a questa operazione il raffinato ambiente di **WinAsm Studio** prevede azioni separate, configurabili singolarmente, a cominciare dalla prima, fondamentale: per attivarla basta *cliccare* sulla **prima iconcina** del gruppo posto in alto a destra, sotto il menu dell'editor (attivabile anche con **Shift + F6**):



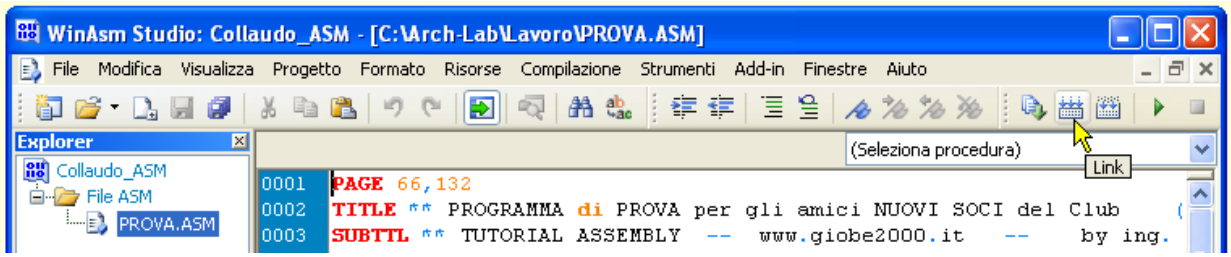

Immediatamente viene **automaticamente** attivato l'intervento dell'***Assembler***, operazione necessaria per generare il codice macchina relativo al nostro sorgente, in stretta coerenza con quanto decritto nel **capitolo**2 del **Tutorial Assembly**; l'area occupata dall'editor si divide in due parti, comprimendo le informazioni visualizzate in precedenza *nella metà superiore* e segnalando, *nella metà inferiore*, l'esito finale di questa fase; se tutto va bene avremo un messaggio simile a quello catturato qui sotto:



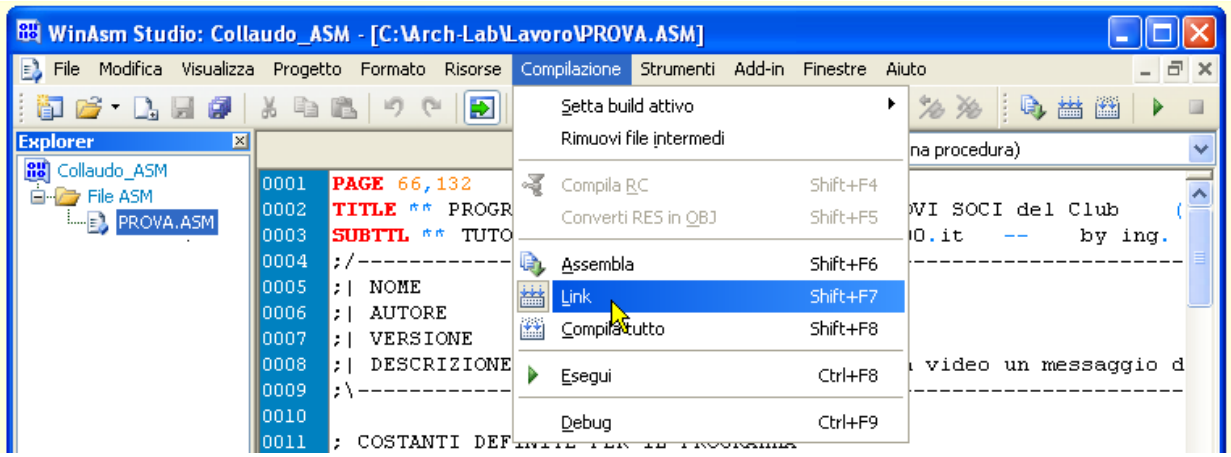
Gli stessi risultati possono ottenuti anche a partire dal menu in alto, via ***Compilazione*** > ***Assembla***:



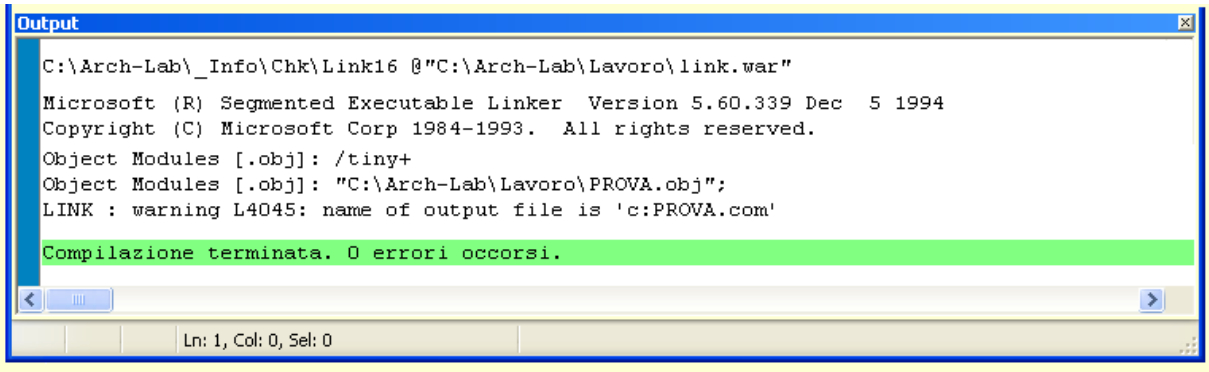

In entrambi i modi, dopo le operazioni di creazione dell'*oggetto* (***.OBJ***), è necessario renderlo ***eseguibile***; anche questa operazione è configurabile e lanciabile *separatamente* dal raffinato ambiente **WinAsm**: per attivarla basta *cliccare* sulla **seconda iconcina** del gruppo posto in alto a destra, sotto il menu dell'editor (attivabile anche con **Shift + F7**):



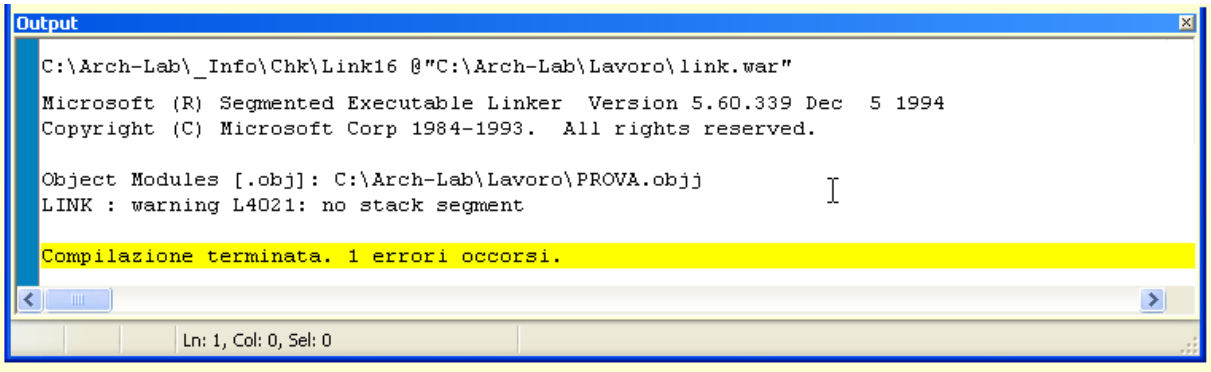

Gli stessi risultati possono ottenuti anche a partire dal menu in alto, via ***Compilazione*** > ***Link***:




Immediatamente viene **automaticamente** attivato l'intervento del ***Linker*** (vedi sempre **capitolo**2 del **Tutorial Assembly**); *nella metà inferiore* dell'area occupata dall'editor appare ora l'esito finale di questa fase; se tutto va bene avremo un messaggio simile a quello catturato qui sotto:

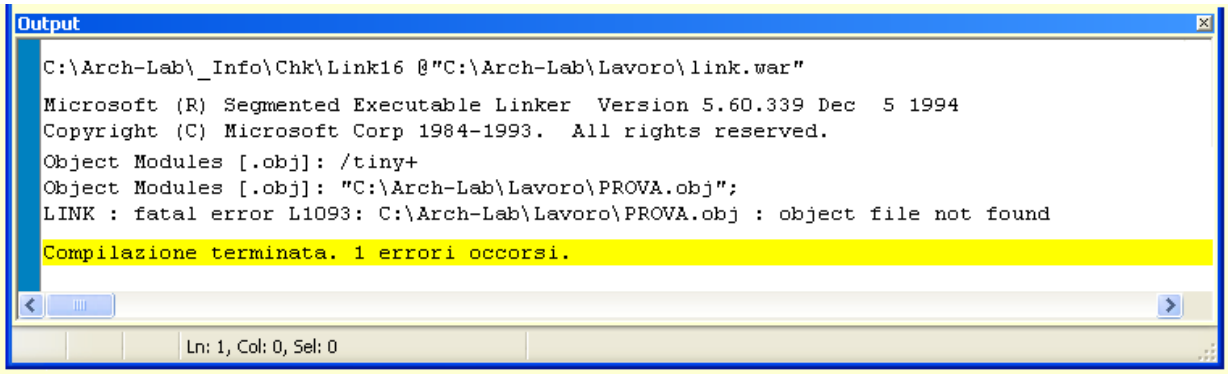



La ***fase di link*** ha restituito un messaggio verde perché il ***progetto*** (come annunciato all'inizio e come si nota dalla dichiarazione tiny tra le righe del messaggio) è stato volutamente da me configurato per generare **eseguibili di tipo COM**; se si decidesse (legittimamente) di modificare la configurazione per generare **eseguibili di tipo EXE**, la stessa fase, pur portata a termine correttamente, darebbe un messaggio di questo tipo:

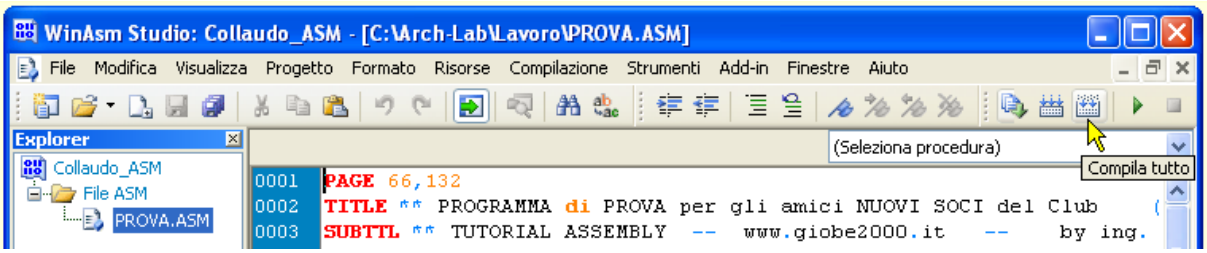



Si noti l'assenza della dichiarazione tiny tra le righe del messaggio. Questa *segnalazione di errore* non ti deve intimorire: si tratta di un errore ***warning*** (cioè più di un *avviso* che di un errore) generato perché il ***linker*** non ha trovato (nel sorgente ASM) la dichiarazione per lo ***Stack Segment***, cosa assolutamente legittima per un codice assembly destinato a diventare eseguibile **COM** (vedi **Tutorial Assembly**).

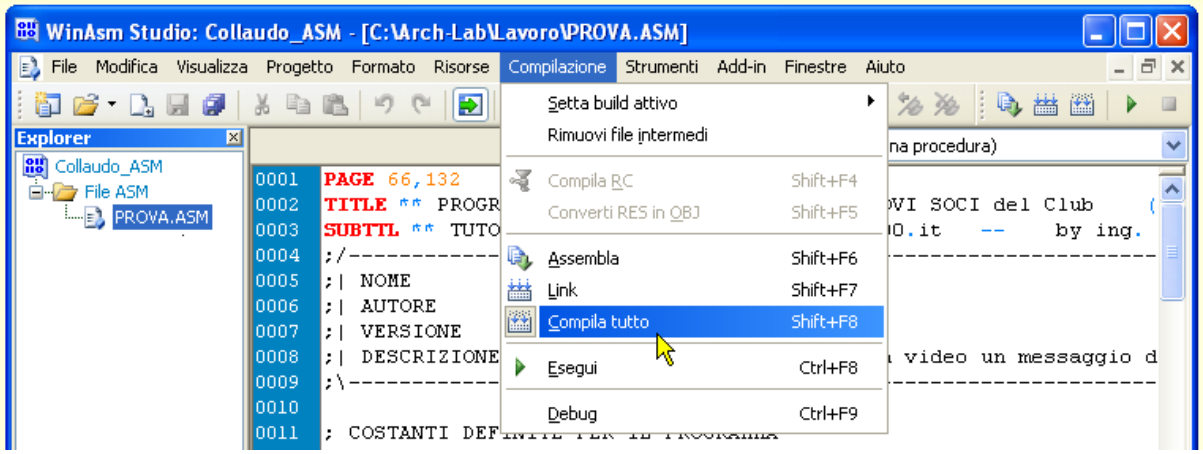
Naturalmente se attivassi questa fase senza prima aver provveduto ad ***assemblare*** il sorgente (con **Shift F6**) l'assenza dell'oggetto OBJ verrebbe subito rilevata e segnalata (si tratta ora di un errore ***fatal*** cioè tale da non poter essere tollerato, per cui l'eseguibile non sarà generato):



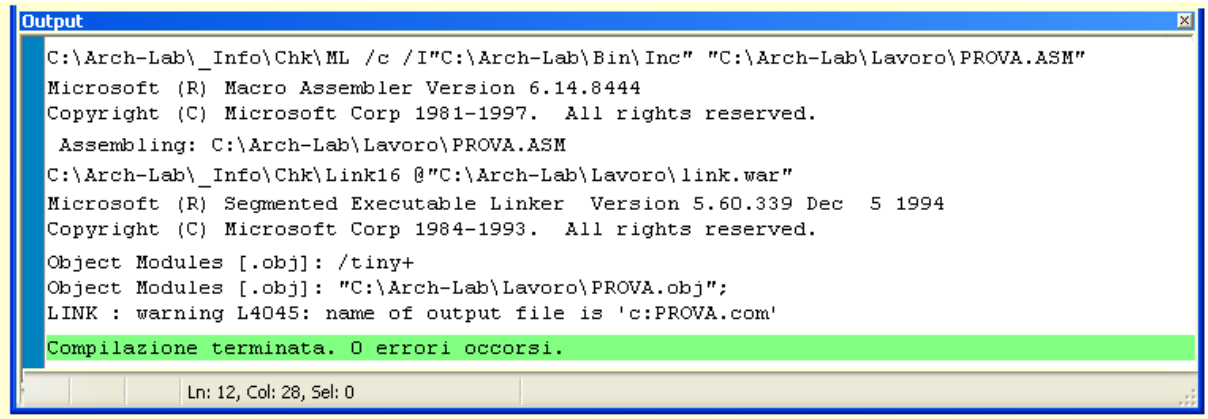
L'ambiente **WinAsm Studio** prevede anche la possibilità di attivare *in sequenza* **entrambe le fasi** di *creazione dell'eseguibile*, lanciando in sequenza prima l'***Assembler*** e il ***Linker***; per attivarla basta *cliccare* sulla **terza iconcina** del gruppo posto in alto a destra, sotto il menu dell'editor (attivabile anche con **Shift + F8**):



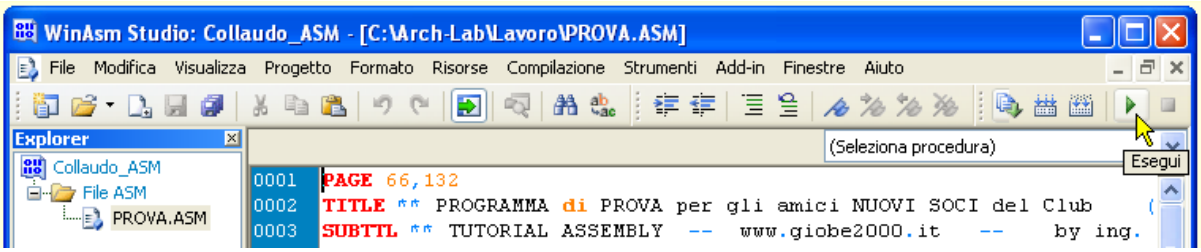

Gli stessi risultati possono ottenuti anche a partire dal menu in alto, via ***Compilazione*** > ***Compila tutto***:



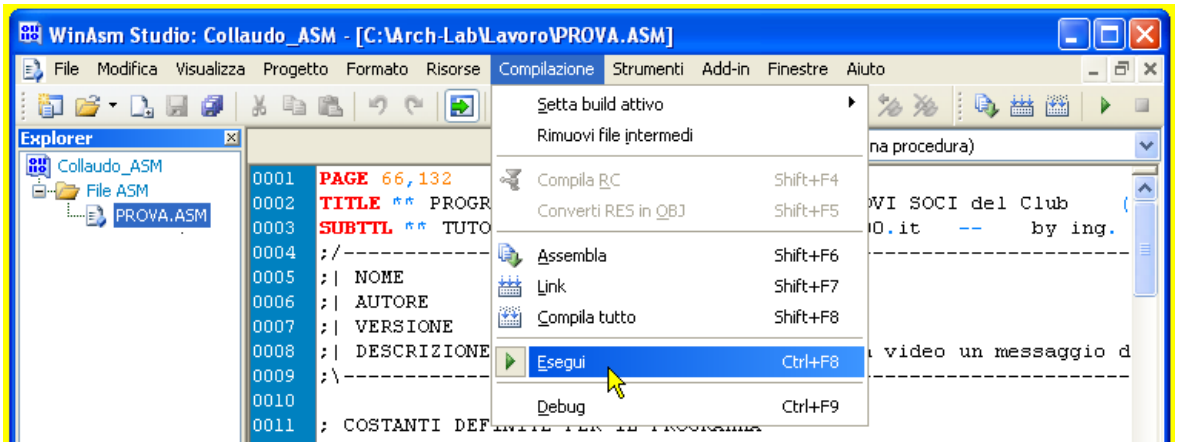

Coerentemente, *nella metà inferiore* dell'area occupata dall'editor appare l'esito finale con le informazioni relative ad entrambe le fasi; se tutto va bene avremo un messaggio simile a quello catturato qui sotto:



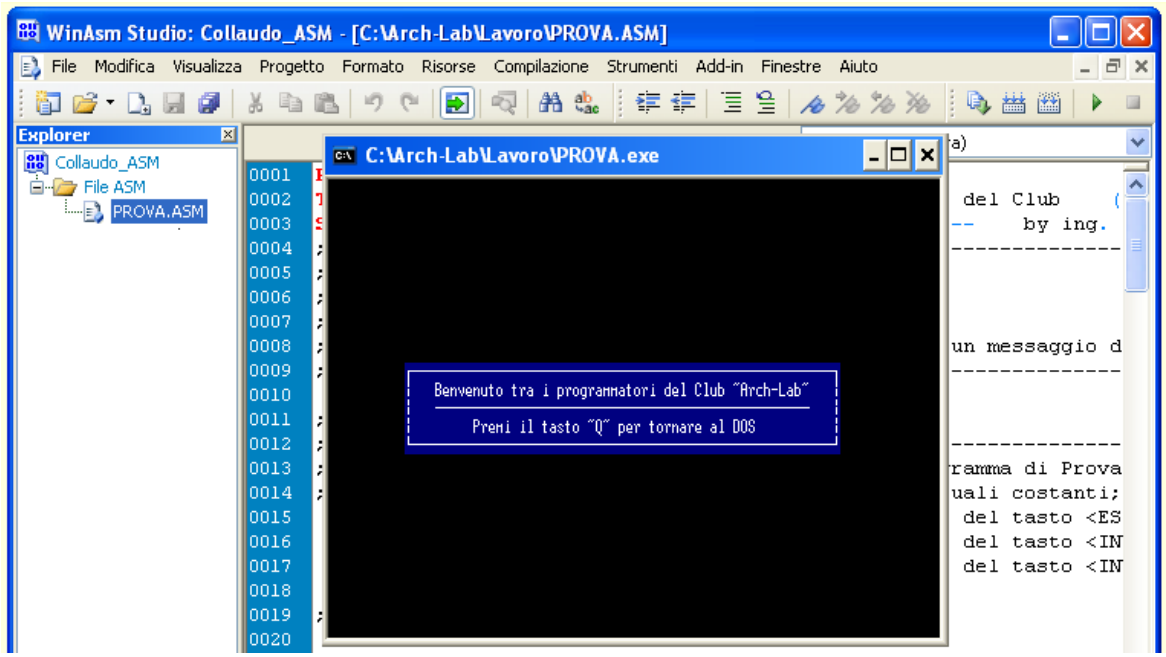

L'ultima **iconcina** del gruppo posto in alto a destra, sotto il menu dell'editor, ti permette di **vedere l'effetto del tuo lavoro** cioè mette ***in esecuzione*** l'*eseguibile* ***COM*** (o ***EXE*** se configurato diversamente) appena compilato e linkato: basta *cliccarci sopra* (oppure premere **Ctrl + F8**):




Gli stessi risultati possono ottenuti anche a partire dal menu, via ***Compilazione*** > ***Esegui***:

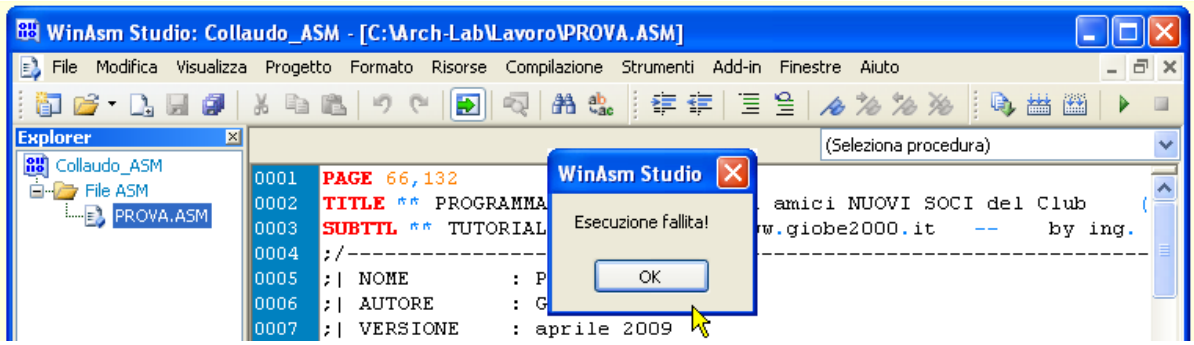



L'effetto prodotto è quello di mostrare l'*eseguibile* ***in esecuzione*** in una ***shell dos*** "***in finestra***"; per esempio con il nostro sorgente pilota, **PROVA.asm**, vedremo:



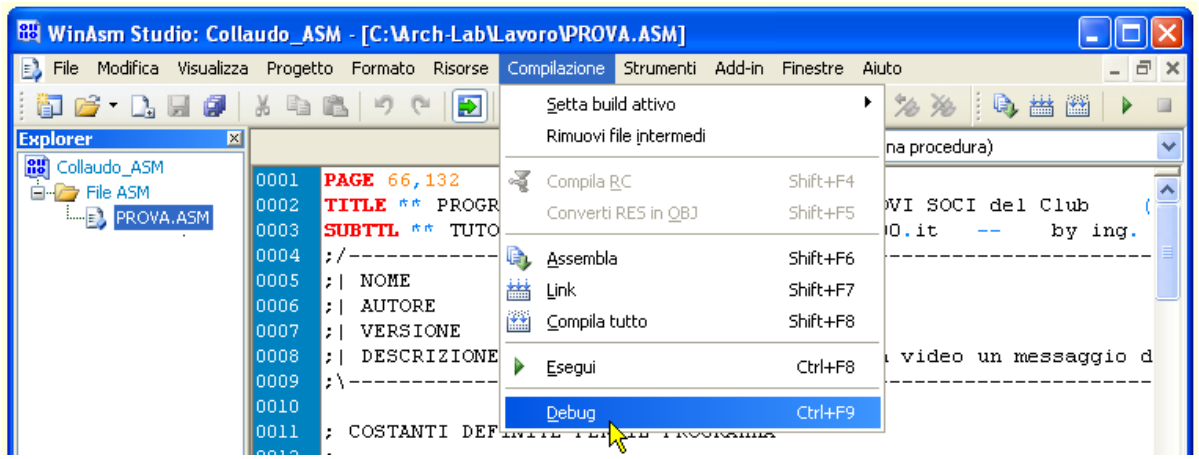

Ricordo che, per fruire di maggiore dettaglio è possibile aprire il progetto "***a pieno schermo***" semplicemente premendo **ALT** + **Invio** (questo con tutte le versioni di Windows **meno *Vista***!!); premendo di nuovo **ALT** + **Invio** si ritorna alla esecuzione della ***shell dos*** "***in finestra***".

Naturalmente se il sorgente ***non è stato ancora compilato*** l'operazione non sarà possibile e, in alternativa al servizio, verrà suggerito cosa fare:




**Riassumendo:** dopo aver **caricato** (o **creato**) un sorgente ASM dovrai prima **assemblarlo** con **Shift F6**, poi **linkarlo** con **Shift F7** e quindi **vederlo** con **Ctrl F8**

**WinAsm Studio** prevede intrinsecamente (nella sua ottica di costituire un **IDE**) la possibilità di disporre dal suo interno una **fase di debug** attivabile a partire dal menu in alto, via ***Compilazione*** > ***Debug***:




In fase di configurazione è possibile affidare questo compito a qualunque **debugger**: la cartella **C:\Arch-Lab_Debug**=%SystemDrive%**\Arch-Lab_Debug** del mio **Ambiente** ne raccoglie numerosi, come ***WinDbg*** o ***OllyDbg***, per citarne 2 che *vanno per la maggiore*); anche in quest'ambito ho però preferito attivare il classico **Debug DOS**, tipico riferimento trasversale in tutte le predisposizioni.

Nonostante diversi tentativi non sono però riuscito a rendere automatica l'apertura sul file eseguibile corrente, non essendo stato in grado di scoprire le ***variabili d'ambiente*** di **WinAsm Studio**, e in particolare quella utilizzata per localizzare il ***nome*** del file sotto

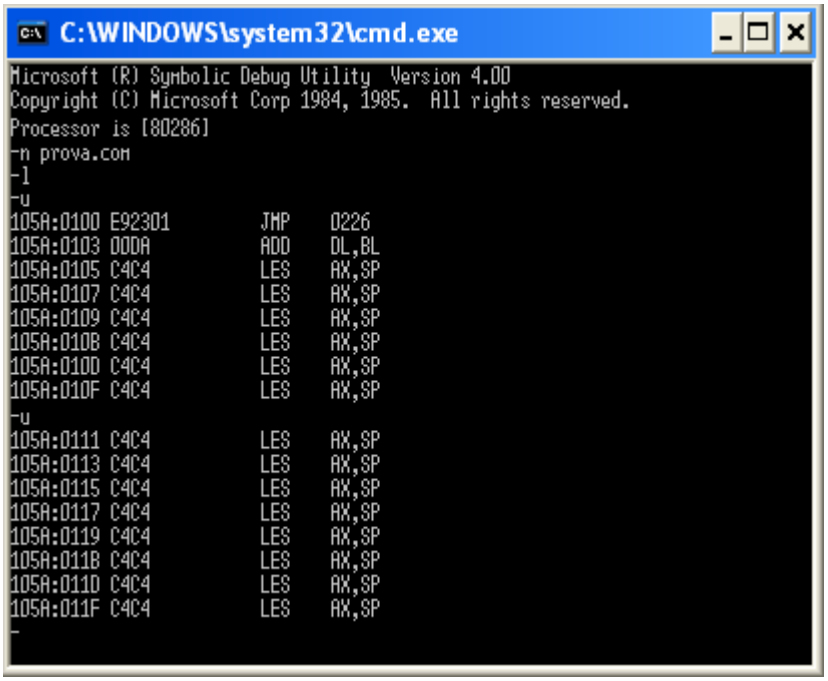
progetto: nell'ambito DOS esso si identifica con %1 e, in ambienti simili, con %Name% o %File%: nessuno di questi identificatori ha purtroppo funzionato..

Per questa ragione alla fine ho deciso di attivare comunque l'applicativo senza passargli il nome del file da aprire come parametro: **questo compito spetterà a te**, digitando come primi 2 comandi (al prompt di debug, cioè ***subito dopo*** la *lineetta lampeggiante*):

**- n** prova.com + <invio> (per specificare il ***nome del file da aprire*** con debug, nel nostro caso prova.com) e poi

**- l** <invio> (per dare a debug il comando di caricarlo, ***l****oad*, in memoria):


Subito dopo potrai operare come al solito, per esempio per disassemblare (***u****nassembly*):

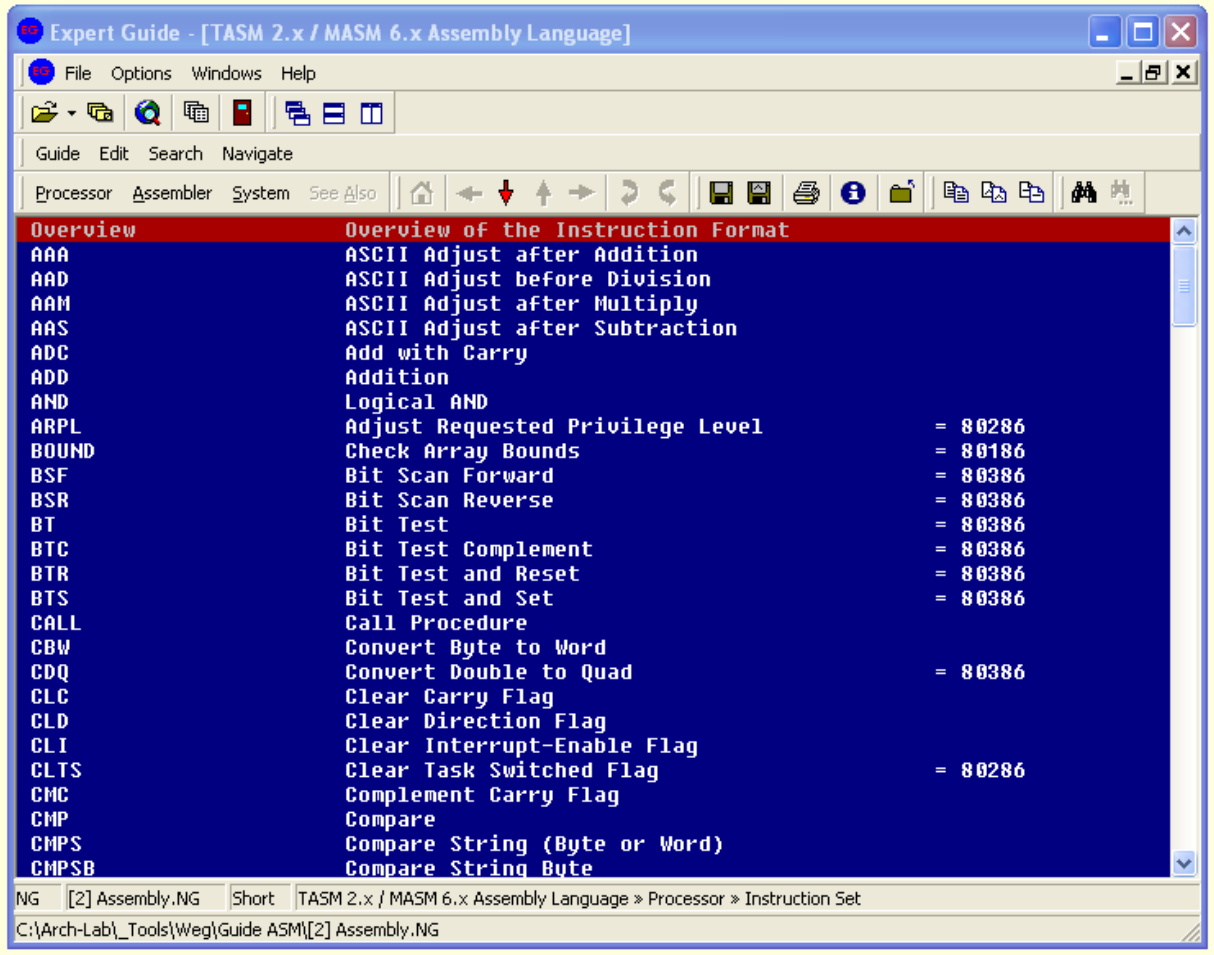



Gli altri editor alternativi previsti per il mio **Ambiente** (**ConTEXT** e **PSPad**) consentono la gestione di **comandi esterni**, ai quali affidare importanti azioni di gestione del sorgente ASM in lavorazione; come più volte sottolineato il sofisticato **WinAsm** prevede intrinsecamente la *messa a punto* delle principali azioni funzionali all'**IDE** che rappresenta; per questo non esistono *icone interne* per la gestione di questo tipo e quindi non è possibile affidare ad una

*icona interna* la gestione del prezioso programma **Weg_2.15** [Copyright **Dave Pearson**], lo straordinario **gestore di archivi di dati** (***Expert Guide Reader*** for ***Windows***) pensato per aiutarti a **scoprire molti gioielli** di questo nostro *tesoro.*

Naturalmente la cosa è fastidiosa ma sostanzialmente irrilevante, essendo sempre possibile attivare l'esecuzione per altra via.

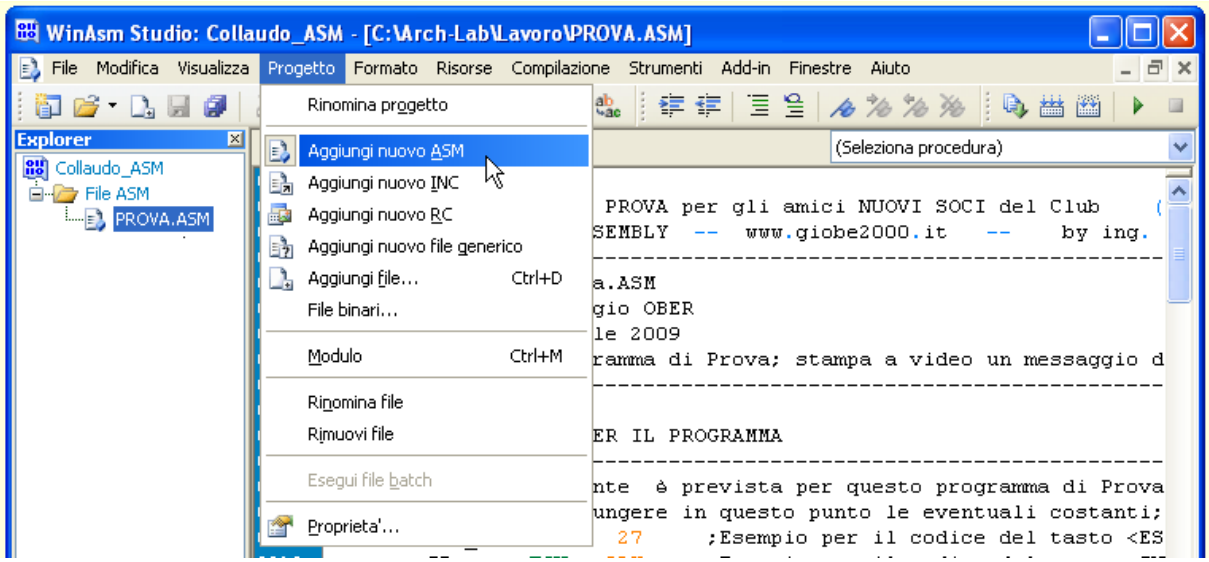
Nella cartella **C:\Arch-Lab_Tools**=%SystemDrive%**\Arch-Lab_Tools** è già pronto il collegamento ***WEG_2.15***, *cliccando* sul quale potrai accedere **anche** (mentre stai lavorando con **WinAsm**) al prezioso lettore, già configurato per vedere una ***ricchissima raccolta*** di informazioni tipiche dell'*assembly*.



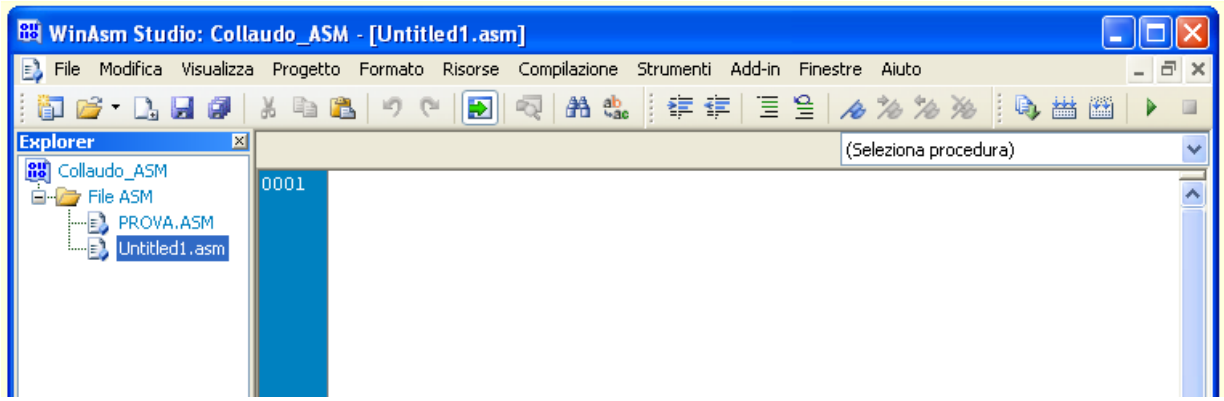

Vediamo ora come fare per ***scrivere*** un **nuovo sorgente Assembly**; **WinAsm Studio** è dotato di strutture adatte a gestire i files in tutte le loro necessità (a partire dal menu in alto, ***File***) ma, come detto all'inizio di questo documento, i files gestiti in questo modo ma non potranno essere sottoposti alle azioni del IDE (***assembler***, ***linker***, ***debugger***, ..).

La cosa più pratica è dunque quella di **aggiungerli al *progetto* attualmente aperto** (di nome ***Collaudo_ASM***) che d'ora in poi fungerà da *contenitore di riferimento* per ogni nostro futuro lavoro; ricordo che ogni variazione verrà comunque annotata nel file il nome ***Collaudo_ASM.wap***, presente nella cartella di lavoro, **C:\Arch-Lab\Lavoro**=%SystemDrive%**\Arch-Lab\Lavoro**.

Selezioniamo dunque ***Aggiungi nuovo ASM*** dal menu in alto, ***Progetto***):




Nell'elenco nella colonna di sinistra verrà aggiunta una nuova voce con il nome fittizio di un nuovo file, ***Untitled1.asm***, e il *piano di editazione* si presenterà **vuoto**, in attesa delle nostre digitazioni.



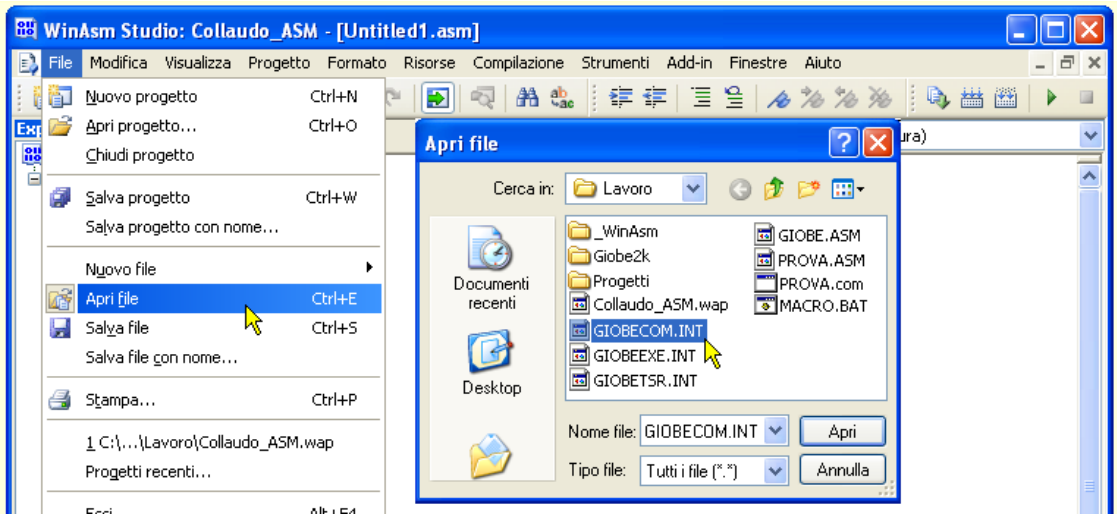

Da questo momento puoi cominciare a scrivere il tuo nuovo sorgente, se ne hai la competenza; al termine dovrai provvedere a **salvarlo** sul disco con un nome meno precario (via menu ***File*** > ***Salva con nome***..).

In alternativa ti ho preparato 3 **Matrici**, pronte per essere riadattate ad ogni tua esigenza di progetto: ciascuna di esse contiene la struttura di un sorgente ASM adatta alla creazione dei 3

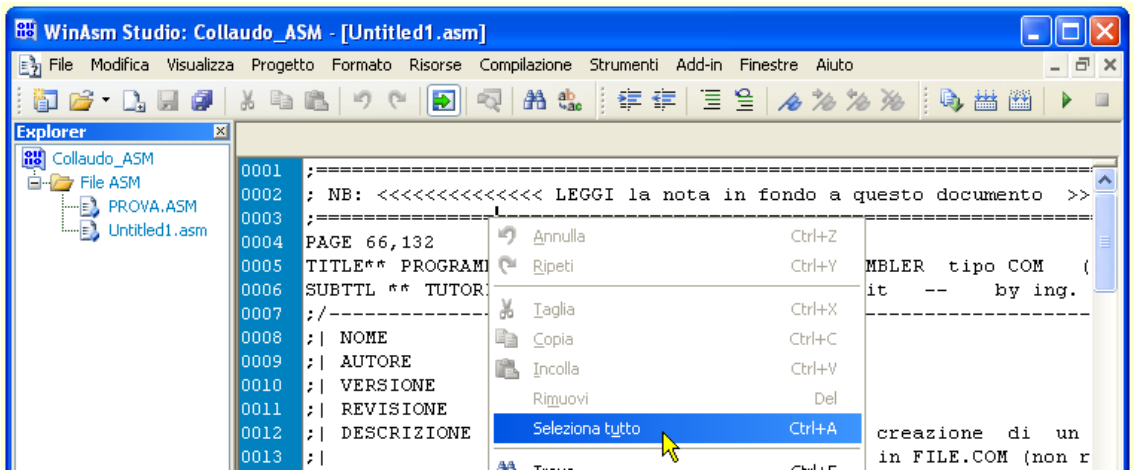
tipi principali di eseguibile: **COM**, **EXE** e **TSR** (per maggiori dettagli vedi il **capitolo**2 del **Tutorial Assembly**); per non appesantire troppo l'inizio del tuo studio ti consiglio di provare con la matrice **GIOBECOM.INT**, pensata proprio per produrre la versione di eseguibile più **COM**patta prevista dalla attuale configurazione.

Per trasferire la **matrice** desiderata nel *piano di editazione* ogni sistema è buono; la tua pratica nell'usare il gestore delle risorse di windows e la tecnica del copia-incolla ti può certamente aiutare.

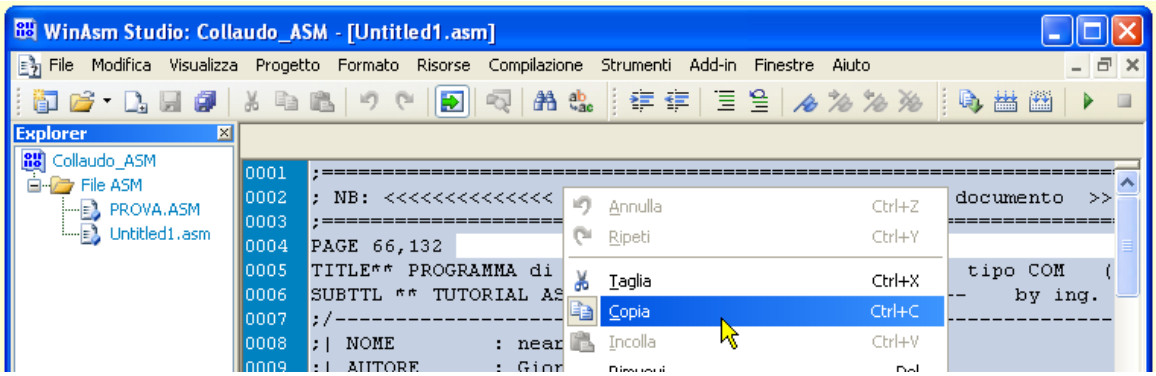
Volendo utilizzare le strutture di **WinAsm** è sufficiente selezionarla a partire dal menu in alto, via ***File***> ***Apri file***:



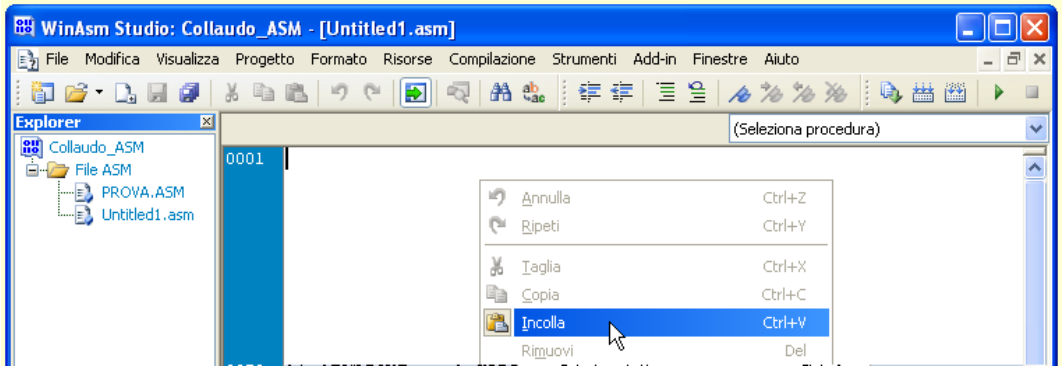

Il testo della matrice sarà ***immesso*** *nel piano di editazione* e basterà *selezionarlo tutto* ...



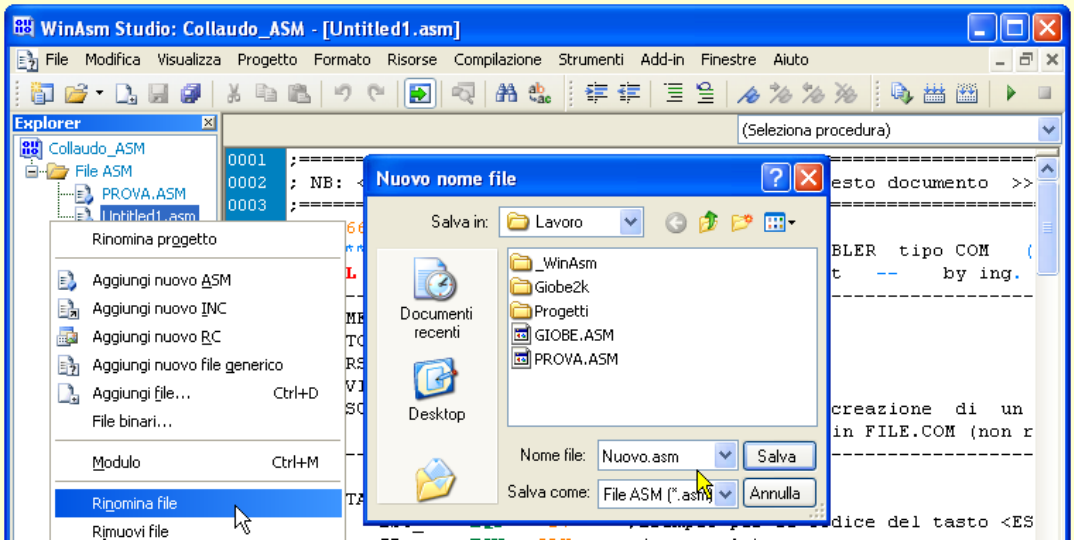

... copiarlo ...



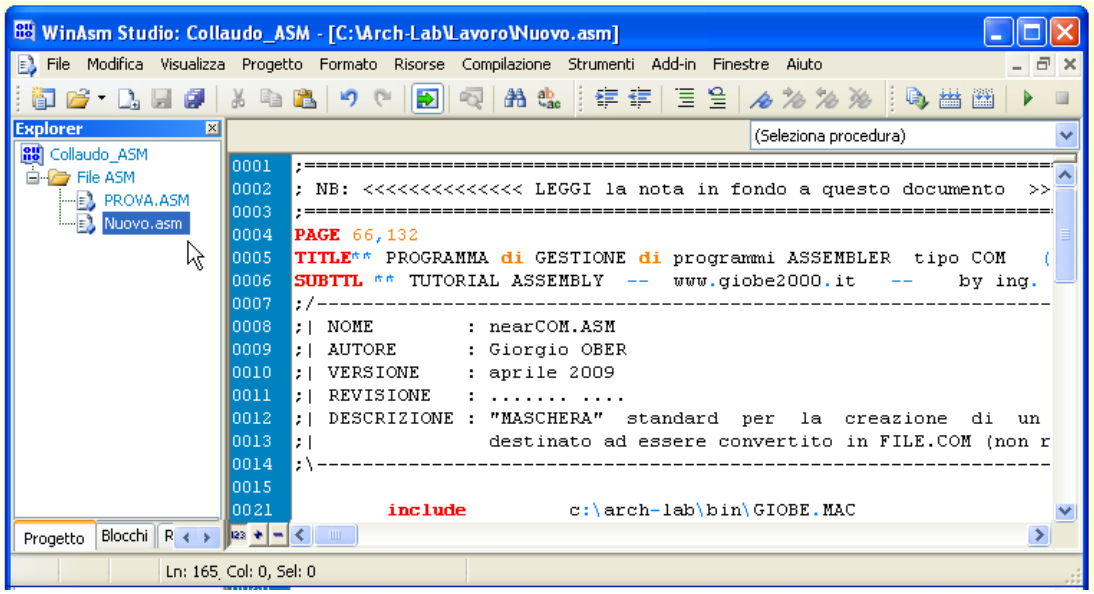

..e (dopo aver chiuso la sua finestra con la **X** in alto a destra), *incollarlo* nella finestra di ***Untitled1.asm***:




A questo punto basta rinominare il file fittizio ***Untitled1.asm*** (per esempio con il nome ***Nuovo.asm***) ...




.. e il gioco è fatto:




Osservando quello che è successo dopo aver assunto questo testo nel piano di lavoro (.. ma anche se decidi di scriverlo da te, completamente *ex-novo*) ti accorgerai che in esso **già si notano le tipiche colorazioni** dei *dettagli sintattici*: questo è un chiaro segno che la *configurazione dell'Ambiente* **è già attiva** e pronta a funzionare !!

---

Bene!! Non ti resta che ***smanettare*** e ***imparare*** la nobile arte .. Ma prima di lasciarti ti voglio proporre una situazione nella quale tuo malgrado incorrerai spesso, almeno le prime volte: **che succede se sbagli qualcosa**?

Per fortuna l'**Assemblatore** e il **linker** sono in grado di accorgersi se qualcosa va storto (come al solito, per ogni dettaglio ti rimando al **capitolo**2 del mio **Tutorial Assembly**); il grande vantaggio sta nel fatto che anche **WinAsmStudio** *provvede in proprio* a darti tutte le *informazioni* sull'eventuale **errore** commesso.

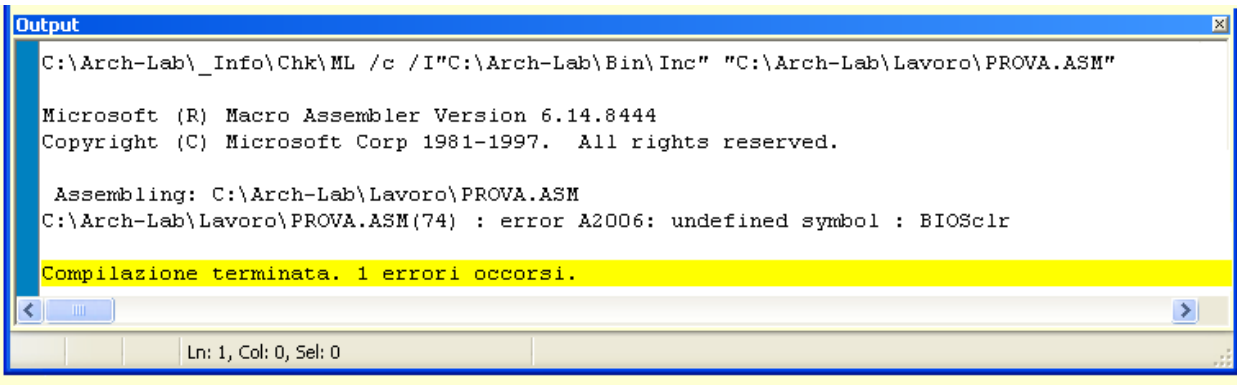
Supponiamo di aver digitato male il nome di una ***procedura***; per esempio alla **riga 74** del sorgente **PROVA.ASM** (ricaricalo e scorri il sorgente fino a localizzarle *questo numero*, sulla sinistra del *piano di editazione*) il sorgente prevede l'istruzione `CALL BIOScls`

```
0066 ;*****************************************************************************
0067                                ; ------------------------------------------------
0068 Prova    proc    NEAR          ;       AREA DEDICATA AL MAIN PROGRAM
0069                                ; ------------------------------------------------
0070 ;-----------------------------------------------------------------------------
0071                                ; ------------------------------------------------
0072                                ; Impostazione della pagina interattiva
0073                                ; ------------------------------------------------
0074          CALL     BIOScls      ;Provvede comunque alla cancellazione del video,
0075                                ;posizionando il cursore alla riga 0, colonna 0
0076          CALL     VIA_cur      ;Rende il  cursore invisibile
0077
0078         M_msgC TESTO1,10,10,1FH;
0079         M_msgC TESTO2,11,10,1FH;  Stampa il messaggio di benvenuto  dentro una
0080         M_msgC TESTO3,12,10,1FH;  cornice
0081         M_msgC TESTO4,13,10,1FH;
0082         M_msgC TESTO5,14,10,1FH;
```

Supponiamo di aver scritto, invece, `CALL BIOSclr`, per disattenzione ...

```
0074        CALL    BIOSclr         ;Provvede comunque alla cancellazione del video,
0075                                ;posizionando il cursore alla riga 0, colonna 0
0076        CALL    VIA_cur         ;Rende il  cursore invisibile
0077
0078      M_msgC TESTO1,10,10,1FH;
```

Non appena si provvede alla fase necessaria per **compilare il sorgente** (cioè *cliccando* sulla **prima iconcina** del gruppo posto in alto a destra, sotto il menu dell'editor (attivabile anche con **Shift + F6**) il gestore che lancia automaticamente l'***assembler*** si accorge subito dell'imprecisione, segnalandola nell'area posta nella parte in basso dell'editor:




In questa segnalazione c'è tutto quello che è necessario sapere: il messaggio è quello autentico di MASM, che indica **su quale riga** del testo sorgente ("PROVA.ASM(74)" si è manifestato l'errore, chiaramente alla riga 74 del sorgente PROVA.ASM, e **il tipo d'errore** ("undefined symbol: BIOSclr", chiaramente un simbolo non riconosciuto, essendo `BIOSclr` diverso da `BIOScls`).

Ti faccio notare che la mia personale predisposizione delle tabelle per la colorazione della sintassi dei principali comandi assembly (raccolte nel file ***MASM.vas*** e conservato nella sottocartella **C:\Arch-Lab_WinAsm\KeyFiles**) prevede una colorazione **verde** per tutte le strutture della preziosa libreria **Giobe.LIB**: già il fatto che l'etichetta sotto esame abbia ***perso il colore*** può insinuarti il dubbio che essa sia stata scritta in modo sbagliato!!

Certo, ora dovrai ***imparare ad interpretare*** questi tipi di informazione cercando di capire che tipo d'errore hai commesso; naturalmente ho già previsto sul mio **Tutorial Assembly** una serie di indicazioni utili, in questo senso, che ti consiglio di leggere qui.