1 Avvertenze e legenda (significa DA LEGGERE!)

1.1 Aspetto grafico del testo e contenuto

I termini sono scritti un grassetto la prima volta che vengono usati oppure vicino alla loro definizione.
Essendo presente un indice analitico se non si ricorda il significato di un termine si può andare all'indice analitico, trovare il riferimento alla pagina ove si trova citato quel termine, in quella pagina la definizione si trova subito perché il termine è in grassetto.

I vari paragrafi di questo libro hanno aspetto grafico leggermente diverso. Ciò per far individuare a prima vista il livello del contenuto. Infatti in alcuni punti del testo vengono presentati argomenti un po' più avanzati, che sono distinti con diverse caratteristiche tipografiche.

I paragrafi normali del testo sono scritti così:
Normale (fondamentale): argomenti più semplici e fondamentali per la comprensione del testo. Non si deve rinunciare a capirli. Insistere fino a che non sono chiari, aiutandosi utilizzando l'insegnante e/o i compagni.

Evidenziato: alcune parti del testo normale, che si vuole indicare come più importanti, sono scritte così.

Sviluppato: argomenti un po' più sofisticati, che danno una visone più "intima" della materia, importanti per l'approfondimento ma non indispensabili per la comprensione dei fondamenti.

Avanzato: argomenti accessori di interesse non immediato, sconsigliati durante la prima lettura, almeno a quelli che, come lo scrivente, non hanno il dono di capire tutto e subito. Quando si studiano questi argomenti e non si capiscono, è meglio fare un segnetto sul libro e proseguire. In seguito, quando lo studio degli argomenti successivi avrà illuminato la mente e chiarito alcuni punti, si potrà tornare al segnetto per finire bene il lavoro.
Spesso le frasi scritte in questo modo faranno riferimento a concetti spiegati più avanti nel libro, ma sono scritte il quel punto per aumentare la completezza e la coerenza del testo.

```
Applicativo: riferimenti a strumenti di lavoro specifici, esempi pratici, proposte o soluzioni di problemi.
```

```
Sintassi: descrizione della sintassi di un'istruzione di un linguaggio di programmazione.
Le convenzioni adottate per queste descrizioni sono descritte al successivo paragrafo 1.3
```

Il codice dei programmi viene presentato in questo modo:

```
    MOV AX, 0    ; (questa è un'istruzione Assembly)
```

A proposito della scrittura del codice c'è da dire che molto spesso verranno presentati degli spezzoni di programma di poche righe. Ho usato molto spesso la notazione **..** (punto punto) per significare che mancano dei pezzi di programma e che quelli inseriti sono solo la parte che mi interessa nell'esempio in questione. Voglio rimarcare che la notazione **..**, che mi piace e viene dal Pascal, NON E' PARTE DEL LINGUAGGIO ASSEMBLY, per cui se si ricopiano molti esempi così come sono scritti nel testo, essi non funzioneranno dove ci sono i due punti!

Nel testo le citazioni hanno l'aspetto che segue:
Citazione:
TOP TEN THINGS ENGINEERING SCHOOL DIDN'T TEACH YOU:
..
7. Anything practical you learn will be obsolete before you use it, except the complex math, which you will never use.
..
5. Engineering is like having an 8 a.m. class and a late afternoon lab every day for the rest of your life.
..
10. If you like junk food, caffeine and all-nighters, go into software.

Da "The Embedded Muse" occasional newsletter sent via email by Jack Ganssle, September 3, 1997.
(naturalmente non tutte le citazioni sono in Inglese!)[1]

1Note per la comprensione di questa citazione: in America e in Inghilterra "engineer" è un qualsiasi tecnico, non necessariamente un "ingegnere" laureato, come in Italia. Dunque un bravo perito industriale si può ben dire un "engineer"!
"Class" significa "lezione", "lab" esercitazione di laboratorio, "junk" spazzatura ("junk food": hamburger, merendine e schifezze del genere ..)

oppure:
".. sempre l'ignoranza fa paura ed il silenzio è uguale a morte"
(F. Guccini, "Canzone per Silvia")

Purtroppo alcune citazioni potranno essere anonime:

Frustration is a part of learning so practice patience
(non ricordo da dove l'ho presa)

1.2 Termini, lingua inglese

Molte definizioni sono presenti solamente come introduzione al gergo informatico, spesso si potrebbero usare parole italiane più belle, ma siccome nella "vita reale" non si usano quelle, ma i termini di gergo, spesso ho preferito adottare questi ultimi, nella speranza di scrivere un testo più utile.
La prima volta che ho introdotto un nuovo termine tecnico ho messo sempre la versione in Inglese e quella in Italiano, questo perché nella vita reale è più frequente avere a che fare direttamente con la terminologia in Inglese che con quella in Italiano.
In questo libro ci sono molti simboli strani, quasi tutti sono delle sigle (**acronym**) o parti di parole troncate. A dispetto del fatto che si chiamino "mnemonici" sono abbastanza difficili da ricordare, a meno che non si tenga presente il loro significato in Inglese. Per questo ho sempre messo, vicino al simbolo mnemonico, anche ciò che esso significa. Nelle parole che illustrano il significato ho evidenziato in grassetto le lettere che sono state usate per comporre la sigla (p.es. RAM = **R**andom **A**ccess **M**emory o XCHG = e**xch**an**g**e).
Da notare che in America la lettera X, che somiglia ad una croce, spesso viene usata come "cross" (p.es. cross assembler viene spesso abbreviato in XASM; cross reference è XREF).

1.3 Convenzioni sulla descrizione della sintassi dei linguaggi

La notazione usata in questo testo per la descrizione della sintassi degli elementi di linguaggio è ispirata ai descrittori BNF (**B**ackus – **N**aur **F**ormalization). Non si tratta di BNF rigorosi, come quelli (bellissimi!) che ci sono nella definizione del Pascal (Wirth, Jensen: "Pascal, Manuale e standard del linguaggio"). La notazione è leggermente diversa ed è stata semplificata. Quando la notazione si complicava, ho preso qualche scorciatoia, assegnando dei metacomandi che ritengo noti, senza specificarli fino in fondo, fidando un po' sull'intuito di chi legge.

Notazione utilizzata:
Le parentesi angolari (< >) includono un'entità della sintassi del linguaggio, un "**metacomando**"[2], la cui definizione è data in un altro punto. La definizione di un metacomando è data a destra del simbolo :=.
Come già detto può però succedere che la definizione di un metacomando non sia ulteriormente dettagliata, per non appesantire troppo il testo. In quei casi si considererà che il solo nome del metacomando spieghi tutto.

Esempi di metacomandi che non saranno dettagliati:
`<Espressione>`
la formalizzazione di una sequenza di operazioni che si deve risolvere in un valore univoco.
Il risultato di una espressione ha tipo: per esempio <EspressioneLogica>, <EspressioneNumerica>, <EspressioneStringa>.

`<Variabile>`
una entità che può memorizzare uno o più valori, modificabili da programma

`:=`
il segno di uguale preceduto dal duepunti è una **definizione**. Il metacomando il cui nome sta a sinistra fra parentesi angolari è definito nel modo specificato a destra. Nella definizione di istruzioni complesse il metacomando che sta a sinistra va sostituito con ciò che sta a destra dell'uguale.

`caratteri in grassetto`
le cose scritte in grassetto sono lettere che fanno **parte dell'istruzione**; pertanto devono essere effettivamente scritte nel codice sorgente del programma

`..`
la presenza di un segno **..** in grassetto significa che si può **ripetere** quello che segue per tutte le volte che si vuole

2 Il nome è molto pomposo, ma è quello che viene usato in questi casi :-(

`[ ]`
Le parentesi quadre comprendono un insieme di metacomandi **opzionale**, cioè che si può anche omettere.

`|`
il segno | sta per "oppure" (**or**)

`;`
commento: ciò che segue il punto e virgola è un **commento** a ciò che precede, non fa parte della descrizione della sintassi

`( )`
La parentesi tonda è usata quando ci potrebbero essere ambiguità nel capire l'**ordine** con cui vanno valutate le "espressioni" che definiscono i metacomandi.