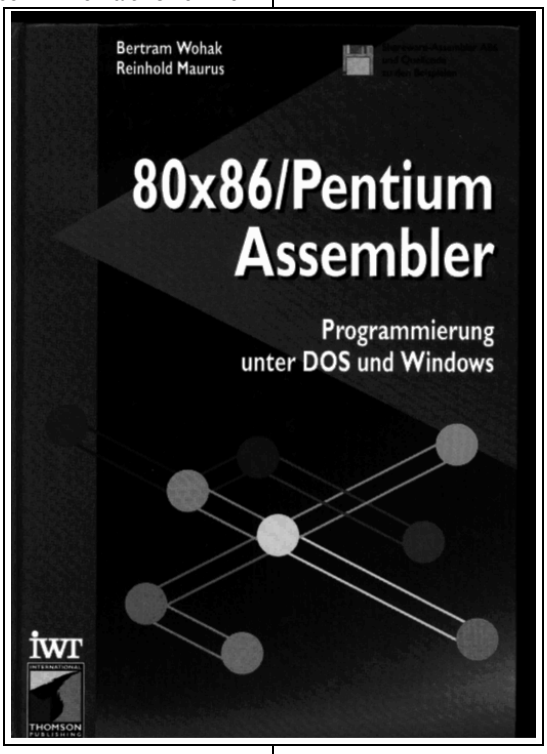
80x86/Pentium Assembler

Bertram Wohak und Reinhold Markus, IWT Verlag GmbH, 1. Auflage 1995, ISBN 3-8266-2601-X, öS 694,-, Shareware-Assembler/Debugger A86/D86 auf Diskette

Walter Riemer

Im Unterricht werde ich gelegentlich vom einen oder anderen Schüler gefragt, welches Assembler-Buch ich ihm, etwa um versäumten Stoff nachzuholen, empfehle. Die Frage ist für mich nicht zu beantworten, weil sämtliche 86-Assemblerbücher auf dem Markt den gleichen Mangel haben: sie sind viel zu umfangreich, erfordern also beim Studium unverhältnismäßig viel Zeitaufwand, geben aber im Verhältnis zu ihrem Umfang zu wenig Anleitung zum praktischen Programmieren.



Auch das vorliegende Buch hinterläßt einen zwiespältigen Eindruck. Bei einem Gesamtumfang von fast 800 Seiten sind 362 einer detaillierten Beschreibung des Befehlssatzes gewidmet, wobei bei vielen Befehlen sogar der binäre Machinencode angegeben ist. Gerade in letzterem manifestiert sich deutlich das Mißverhältnis in der Konzeption des Buches: einerseits richtet es sich (wie mehrfach betont wird) nicht an Systemprogrammierer; deswegen wird zum Beispiel auf das hardwareunterstützte Speichermanagement der Prozessoren nicht eingegangen. Andererseits interessiert aber doch wohl den Assembler- Anwendungsprogrammierer (und erst recht den Lernenden) der binäre Object-Code nicht, hat er doch mit Assembler und Debugger Mittel zur Hand, die ihn gerade davon entlasten, sich um den Binärcode kümmern zu müssen. Diesbezüglich hätte eine kurze Prinzipdarstellung des Codeaufbaus völlig genügt; wie sonst soll der unerfahrene Lernende in der Lage sein, für seine praktische Arbeit Wesentliches von Unwesentlichem zu unterscheiden?

Auf der anderen Seite werden Grundlagen (des Befehlssatzes, der Adressierung usw.) für Lernende zu knapp dargeboten; insbesondere sind die einzeiligen Bespiele von Befehlen unkommentiert und daher in ihrer tieferen Bedeutung für den Lernenden schwer durchschaubar.

Man wird den Eindruck nicht los, daß die Autoren im wesentlichen die Referenzhandbücher in eine andere Form gegossen haben, ohne wirklich eine eigenständige Darstellung anzustreben.

Auch Ankündigungen auf dem Umschlag wird nicht wirklich in die Tiefe gehend entsprochen: die angekündigte "Einführung in die Programmierung von Peripheriebausteinen, Interruptroutinen und TSRs" entbehrt völlig eines systematischen Unterbaus; bestenfalls kann das eine oder andere Prinzip einem kommentierten Beispiel entnommen werden. Besonders deutlich wird dies im Falle der TSRs: weder Inhaltsübersicht noch Sachverzeichnis helfen beim Suchen nach der TSR-Programmierung, das Beispiel ist sicher nicht problemlos allgemein umsetzbar und verrät überdies nichts über die zugrundliegenden Gedankengänge; auf das nicht unaktuelle Entladen einer TSR wird überhaupt nicht eingegangen.

Das Buch entspricht zwar dem durchaus positiv zu bewertenden Trend, mithilfe des MASM 6.11 einem strukturierten Programmieren und einer hochsprachenorientierten Speichermodellfestlegung noch näher zu kommen, schweigt sich aber über die Grundidee strukturierter Progammentwicklung aus. Es wird auch nicht sehr deutlich, daß die neuen Punkt-Direktiven (.IF-Gruppe, .WHILE und .REPEAT usw.) eigentlich etwas ähnliches sind wie Makros. Allerdings erscheint ein zu weit gehendes Ausnützen derartiger Fähigkeiten eines Assemblers aus Sicht eines Elektronik-Schülers, der auch noch andere µP-Assembler kennenlernen soll, auch wieder etwas fragwürdig; jedoch ist dies nicht ein das vorliegende Buch betreffender Aspekt.

Aus didaktischer Sicht empfinde ich es als sehr störend, daß bezüglich Schreibweise nicht auf Einheitlichkeit geachtet wurde: So sind die Mnemonics in den Erklärungsbeispielen in Großbuchstaben, in den Programmbeispielen teilweise in Kleinbuchstaben; gleiches gilt für die Kommentare. Man weiß doch heute (und die Entwicklung der Schreibeweise in den Hochsprachen, insbesondere in der Windows-Programmierung bestätigt dies), daß Kleinschreibung und insbesondere gemischte Groß-/Kleinschreibung viel besser lesbar ist und außerdem bei systematischem Einsatz Denkanker bietet. Das Buch trägt also nicht zum Vermitteln einer sauberen Programmierweise bei. Auch die Ausrichtung im Quellcode wird teilweise vernachlässigt: Das Minium wäre zumindest, daß symbolische Adressen ganz links stehen und mnemonische OpCodes jedenfalls entsprechend eingerückt sind; auch dies ist nicht konsequent durchgeführt (so zum Beispiel auf Seiten 613, 649 und 650).

Den geläufigen Schülerfehler von einem im Speicher "resistenten" Segment auf Seite 748 ganz unten lesen zu müssen, überrascht dann auch nicht mehr (obwohl der Besprecher durchaus Verständnis für das Bestehenbleiben des einen oder anderen Druckfehlers hat, das aus eigener leidvoller Erfahrung resultiert).

Warum das Zeichen '&' (auf Seite 619) nur "kaufmännisches Und" und nicht fachsprachlich einwandfrei "Ampersand" genannt wird, ist auch rätselhaft, lohnte es sich doch durchaus, diesen Fachausdruck bekannt zu machen, umso mehr als er oft von Personen, die ihn wenigstens im Prinzip kennen, fälschlich französisch gefärbt ungefähr wie "en passant" ausgesprochen wird.

Wenn wir schon beim Sprachlichen sind: auch der "physikalische" Speicher (und ähnliches) feiert fröhliche Urständ: sollte nicht ein Buch, dessen Ziel es ist, zu einem besser fundierten Wissen des Lesers beizutragen, auch auf den Unterschied zwischen "physisch" und "physikalisch" (auf Englisch unglückseligerweise beides "physical") aufmerksam machen?

Stellvertretend für viele kleine Ungenauigkeiten, die leider einem schlampigen Denken, wie wir es als Lehrer immer wieder bekämpfen müssen, Vorschub leisten, sei noch auf Seite 77 zitiert: "... wurde der maximal adressierbare Speicher auf 2 hoch 32 - 1 erhöht." 2 hoch 32 - 1 ist jedoch die höchste Adresse in diesem Adreßraum; der Speicher umfaßt 2 hoch 32 adressierbare Speicherstellen.

Kurz und gut: um das Buch zu verstehen, muß man schon recht viel wissen. Meinen Schülern kann ich es daher im Regelfall auch nicht empfehlen; es wird allerdings einen Platz als Nachschlagewerk in meinem Rechenzentrum finden. □