3.0 8051 Assembler und Hochsprachen

Eine kurze Übersicht zum Ablauf einer Programmierung eines 8051 Mikrocontrollers.

3.1 Der 8051 Maschinencode

Grundsätzlich akzeptiert ein 8051 Mikrocontroller als Befehle nur binären Maschinencode. Ein Programmsprung oder das Speichern von Werten wird im Maschinencode durch verschiedene Binärzahlen angegeben. Ein Maschinencode wird auch Operation Code genannt oder auch einfach mit OpCode abgekürzt.
Die Opcodes kann man nach deren Funktionen unterscheiden, z.B. arithmetische Funktionen wie addieren (ADD) und subtrahieren (SUBB), logische Funktionen wie UND, ODER und Schiebeoperationen, Datentransferbefehle wie MOV, PUSH, POP oder z.B. Sprungbefehle in Abhängigkeit von bestimmten Zuständen im Flagregister.
Die Ausführung eines Maschinenbefehls benötigt eine bestimmte Zeit, die in Datenblättern mit der Anzahl der Taktzyklen angegeben ist.

Beim 8051 – Mikrocontroller sind u.a. die folgenden Register (benannte Speicherstellen) wichtig, die einen Zustand beschreiben oder aktuelle Werte für eine Verarbeitung speichern:

A	Der Akkumulator, ein 8 Bit Arbeitsregister
B	Das Hilfsregister bei Multiplikation und Division
R0-R7	Allgemeine Register im RAM (Registerbänke 0-3)
PSW	Das Flagregister zeigt aktuelle Zustände an
SP	Der Stackpointer ist ein Zeiger auf den Stapelspeicher

Weitere SFR Register (Special Function Register) dienen zur Steuerung der Hardware. Mit ihnen können z.B. die I/O-Ports (P1 bzw. P3), die Zähler/Timer oder die serielle Kommunikation gesteuert werden.

Alle SFR Register werden über ihre Adresse als RAM angesprochen (Adressen über 128 Byte).

Der Maschinencode, z.B. 74 0F F5 90 80 FE (hier angegeben im HEX - Format) steht im Flash oder ROM eines 8051und wird einfach von

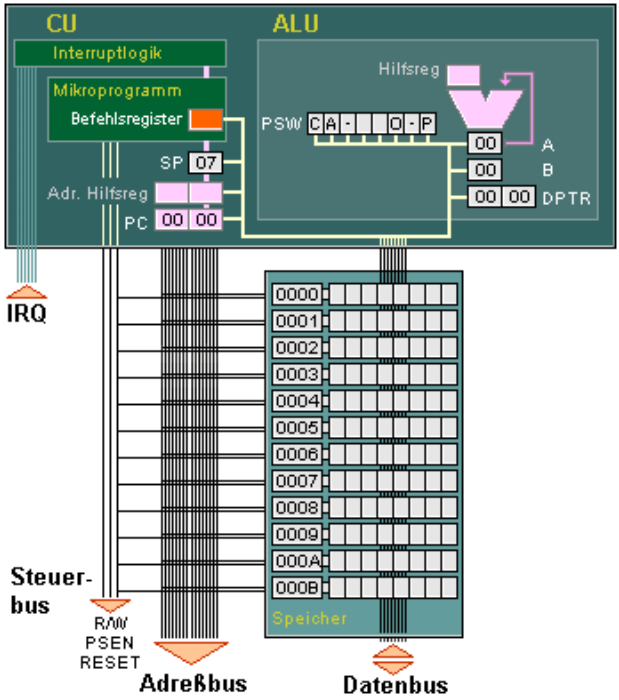
Anfang bis zum Ende hintereinander gesetzt. Dort wird es vom Controller gelesen und dem Programm entsprechend abgearbeitet.

Bei Atmel AT89LPx052 sucht der Mikrocontroller, wie die meisten 8051er Controller, nach dem Einschalten den ersten Befehl an der ersten Adresse 0000H im Speicher.

Der Datenbus wird gelesen und die Zahl in ein Befehlsregister geladen. Der Befehlszähler wird um eins erhöht. Dieser könnte nun die nächste Speicherzelle adressieren, doch die CPU gibt diesen Wert noch nicht zum Adressbus aus, sie muss erst den Befehl untersuchen.

Im Internet findet man zu internen 8051 Abläufen z.B. folgendes Bild und einige detaillierte Erläuterungen unter:

http://www.goblack.de/desy/mc8051chip/theorie/cpu.html



3.2 Hochsprachen und Assembler zur Erzeugung von Maschinencode

Dieser Zahlenwirrwarr ist für den Menschen unleserlich und wird einem Programmierer durch sog. Hochsprachen vereinfacht. Als Hochsprachen, die Programme wiederum in Maschinencode umsetzen können, kommen z.B. C, Basic oder seltener auch Pascal in Frage. Das Übersetzungsprogramm zur Umsetzung des Quelltextes in Maschinencode wird in diesem Fall Compiler genannt (to compile = zusammenstellen). Die Programme werden nach der Erstellung kompiliert und dadurch die HEX oder BIN Dateien für eine Übertragung zum Mikrocontroller erzeugt.

Assembler ist die maschinennaheste Sprache und besitzt nur einen kleinen und überschaubaren Befehlsatz. Die jeweiligen Befehle werden im Quelltext in einfachen Kürzeln (Mnemoniks) angegeben. Jeder Befehl ist genau einem Maschinencode zugeordnet.
Ein Assembler-Quelltext wird in einem Texteditor als Textdatei angefertigt und anschließend wird dieser von einem Assemblerprogramm in Maschinencode umgewandelt.

TASM ist zum Beispiel als Shareware Assemblerprogramm im Internet erhältlich. Die Ergebnisse einer reinen Assemblerprogrammierung sind im Vergleich zu den Compilerausgaben einer Hochsprache in den meisten Fällen erheblich effektiver. Manchmal ist die Verwendung von Assembler sogar unumgänglich wenn man Ausführungszeiten im µs oder ms Bereich berücksichtigen muss.

3.2 Download –eine Hex oder Bin Datei zum 8051 übermitteln

Die erzeugte Datei, mit dem Opcode im Hex Format oder auch im binären (BIN) Format, wird in den Speicher oder Flash eines 8051 übertragen und nach einem Reset vom Mikrocontroller ausgeführt. Das Übertragen einer HEX oder BIN Datei wird dabei oft auch als *Download* oder einfach auch als *Flash* bezeichnet.

Für die Signalübertragung vom PC zum Atmel AT89LPx052 Mikrocontroller kann die SPI Schnittstelle des Controllers verwendet werden. Da ein Standard PC keine SPI Schnittstelle besitzt ist ein Adapter notwendig, der die Signale einer parallelen, seriellen oder USB Schnittstelle des PC für die SPI Schnittstelle umsetzt.

Adapter (und dazugehörige Flash Software) für die parallele PC Schnittstelle sind z.B. von Atmel, GMS Bentheimer Softwarehaus oder anderen Unternehmen erhältlich.

Alternativ kann die HEX oder BIN Datei auch mit einem externen Programmiergerät verarbeitet werden. Der Nachteil dabei ist, dass der Baustein aus der Fassung genommen und in das Programmiergerät gesteckt werden muss. Dieser Vorgang ist nach jeder neuen Programmierung erneut durchzuführen und der Mikrocontroller ist nach dem Programmiervorgang wieder zurück in die Schaltung zu stecken. Mit den oben angegebenen Adaptern wird der Atmel Mikrocontroller direkt in der Schaltung programmiert.

Ob Sie sich für die eine oder eine andere der o.g. Hochsprachen entscheiden, in den meisten Fällen sind eine Vielzahl von Assembler- und Compilerprogrammen im Internet kostenlos verfügbar. Sie finden zudem genügend Internetseiten, die sich mit dieser Thematik beschäftigen. Einige davon sind im Literaturverzeichnis im Anhang aufgeführt.

3.3 TASM eine kurze Übersicht

Mit TASM können Sie Ihre Assembler Quelltext in den Maschinencode umwandeln. Der TASM Assembler entstand ursprünglich bei Speech Technology Incorporated auf der Grundlage einiger C Bibliotheken von Borland International und ist für das Betriebssystem DOS konzipiert. Der TASM unterstützt, neben dem 8051, noch weitere Prozessoren. Für jeden Prozessortyp gibt es eine Tabelle, z.B. TASM51.TAB für den 8051.

Nach einem Download aus dem Internet z.B.
http://www.bookcase.com/library/software/msdos.devel.cross-asm.html
entzippen Sie alle Dateien z.B. in das Verzeichnis c:\tasm\. Ihre Quelltexte sollten Sie ebenfalls im gleichen Verzeichnis ablegen.

Den Assembler Quelltext können Sie mit einem einfachen Texteditor erstellen und danach mit der Dateiendung .asm speichern, z.B. Test.asm.

Das DOS Programm TASM.EXE kann wie folgt aufgerufen werden um einen Maschinencode im Binärformat zu erzeugen:

c:\tasm\tasm -51 –b test.asm test.bin

Unter Windows können sie sich z.B. dazu eine kleine Batch Datei erstellen, die diese Befehlszeile beinhaltet.

-51 steht für die Verwendung der TASM51.Tab Datei
-b für die Ausgabe im Binärformat

TASM erzeugt neben der Test.bin auch eine Test.lst Datei. Diese Datei listet neben dem Quelltext und den Maschinencode auch die aufgetretenen Fehler auf.

```
                          .org 0000H
0000   75 C2 00    main    mov    0c2H,#00H
0003   75 C3 00            mov    0c3H,#00H
0006   75 C6 00            mov    0c6H,#00H
0009   75 C7 00            mov    0c7H,#00H
000C   74 00               mov    a,#00
000E   F5 90       next    mov    P1,a
0010   7A FF               mov    r2,#255
0012   7B 14       loop1   mov    r3,#20
0014   DB FE       loop2   djnz   r3,loop2
0016   DA FA               djnz   r2,loop1
0018   04                  inc    a
0019   80 F3               sjmp   next
001B                       .end
```

Die Angaben auf der linken Seite zeigen die Ergebnisse nach der Übersetzung mit dem Assembler TASM in der vom TASM erzeugten *.lst Datei. Am Anfang jeder Zeile ist die Speicheradresse angegeben.

Die Zahlen in Kursivschrift zeigen die Maschinencodes des jeweiligen Befehls.

Im Internet http://www.hjberndt.de/mc/mc5.htm steht für auch eine Automatisierung eine Windowsoberfläche namens TASMEDIT zur Verfügung, die von H.J.Bernd bzw. B.Kainka in verschiedenen Büchern beschrieben wird.