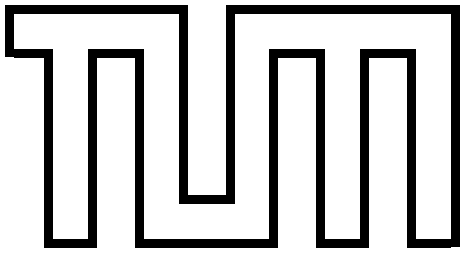


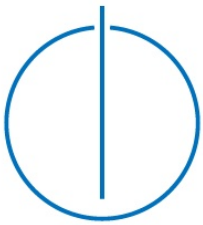
Technische Universität München

Fakultät für Informatik

Seminararbeit in Informatik

Die ersten Mikroprozessoren: ein Rechner, ein Chip

Michael Kratzer, Michael Kiener



Contents

Chapter 1

Einleitung

Test Einleitung

Chapter 2

Intel 4004

Test Intel

Chapter 3

TMS-1000

3.1 Geschichte

Während Intel es erstmals schaffte alle Bausteine eines Prozessors auf einem Mikrochip zu vereinigen, implementierte Texas Instruments parallel zusätzlich Peripherie auf dem Chip und schuf so den ersten Mikrocontroller, welcher auch System-on-a-Chip genannt wurde.
Texas Instruments wurde 1951 von Cecil Howard Green, Jon Erik Jonsson, Eugene McDermott und Henry Bates Peacock gegründet. Das Unternehmen war unter anderem für den ersten integrierten Schaltkreis bekannt. Die Konstruktion eines System-on-a-Chip gelang erstmal den Ingeneuren Gary Boone und Michael Cochran mit dem Controller TMS-1000 im Jahre 1971. Texas Instruments stellte verschiedene Versionen des TMS-1000 her, die in Größe des ROM und RAM Speichers variierten. Anfangs verwendete die Firma die Controller nur in eigenen Produkten, vor allem Taschenrechner, wie der SR-16, welcher 1972 auf den Markt kam. Die Firma ist aus diesem Grund und aufgrund der zahlreichen Nachfolger für ihre Taschenrechner bekannt.
Erst 3 Jahre nach der Erfindung, also 1974, war der TMS-1000 auf dem freien Markt erhältlich. Dies hatte jedoch zur Folge, dass der Markt durch Intel mit ihren Mikroprozessoren schon zum großen Teil eingenommen war. Trotzdem hatte der Controller aufgrund seines extrem niedrigen Preises, 2$ pro Stück, großen Erfolg auf dem Markt. Der TMS-1000 half dadurch die moderne Elektronik für jedermann zugängig zu machen. Der Controller fand nicht nur in Taschenrechnern Verwendung, sondern auch in ersten Handheld-Geräten, Jukeboxen, Türklingeln, Uhren und vielen mehr. Bis zum heutigen Tag wurden ca. 100 Millionen Controller verkauft.

3.2 Mikrocontroller

3.2.1 Aufbau

Mikrocontroller werden nicht zu Unrecht System-on-a-chip genannt und der Aufbau eines Controllers kann daher sehr gut mit den Bestandsteilen eines Computers verglichen werden. Im folgenden werden die Komponenten eines Computers und die Bestandteile eines Mikrocontrollers vergliechen. Außerdem wird die Aufgabe der einzelnen Bauteile des Controllers beschrieben.

PC	Mikrocontroller	Aufgabe
Prozessor	CPU	Der Prozessor führt die aritmethische und logische Operationen aus
Arbeitsspeicher	RAM	RAM ist ein temporärer Speicher für Variablen. Verliert den Speicherinhalt nach dem Entfernen der Betriebsspannung
Festspeicher	ROM	Enthält das Programm
Takt	Takt	Gibt die Geschwindigkeit der Befehlsfolge an
Peripherie	I/O-Ports	Der Controller enthält einfach Ein- und Ausgänge für beispielsweise LEDs, Displays und Schalter

3.2.2 Abgrenzung zu Mikroprozessoren

Oftmals ist es schwierig eine klare Grenze zwischen Mikroprozessoren und Mikrocontrollern zu ziehen. Das liegt vor allem daran, dass nach einiger Zeit meiß Mikrocontroller-Varianten von neuen Mikroprozessor-Architekturen erschienen sind. Die haupstächliche Abgrenzung erfolgt durch die Austattung der Zusatzmodule des Chips. Während ein Mikrocontroller einen Prozessor inklusive Bausteine wie Speicher, In- und Outputs, Timer, usw., konzentriert sich ein Mikroprozessor auf seine eigentliche Hauptaufgabe, die Rechengeschwindigkeit. Durch den gesparten Platz auf dem Chip wird diese wesentlich erhöht.

3.2.3 Architekturen

Viele der verbauten Mikrocontrollern verwendet 8-Bit-Prozessoren, deren Architektur auf die 1970er Jahre zurückzuführen ist. Allerdings gibt es auch 4-, 16- und 32-Bit Mikrocontroller. Die ersten Controller die gebaut wurden waren dabei 4-Bit-Controller und sind deswegen heutzutage noch stark vertreten. Den größten Marktanteil haben mittlerweile die 32-Bit-Controller da die meisten heutigen Mikrocontroller auf Prozessorkernen basieren, von denen 8- und 16-Bit Prozessoren fast nicht mehr hergestellt werden.

3.3 TMS-1000

3.3.1 Allgemeine Daten

Der TMS-1000 gehört zur Familie der 4-Bit Mikrocontroller. Die Größe des ROMs des Controllers beträgt 8.192 Bits, während die des RAMs 256 Bits beträgt. Die maximale Spannung, welche an der Clock und sowohl an den Eingang- und Ausgang-Pins angelegt werden darf, bemisst sich auf 20 Volt. Durch die intern verbauten Oszilliatoren kann der Controller einen Takt von bis zu 0,4 MHz erreichen. Jede Instruktion benötigt 6 Oszilliatoren-Zyklen um vollständig ausgeführt zu werden. Insgesamt verfügt der TMS-1000 über 43 Basisinstruktionen, 12 fixierte und 31 programmierbare.

3.3.2 Pins

Der TMS-1000 verfügt über insgesamt 28 Pins.

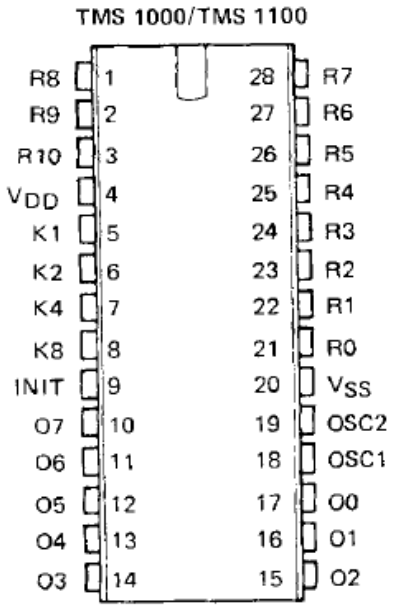


Figure 3.1: Skizze der Pins

Bezeichnung	Anzahl	Aufgabe
R-Output	11	Für die Ausgabe von Kontroll-Daten zuständig
O-Output	8	Gibt den Inhalt des Akkumulators und das Status-Logic-Bit aus
Oszillatoren	2	Zuständig für den Takt des Controllers
Power-Supply	2	Versorgt den Controller mit Spannung
K-Input	4	Eingang für 4-Bits zur Verwendung im Programm
INIT	1	Initialisiert oder setzt die Hardware zurü

3.3.3 Aufbau & Funktionsweise

Der folgende Abschnitt befasst sich mit dem Aufbau und der Funktionsweise der Hardware des Mikrocontrollers TMS-1000.

3.3.3.1 ROM - Read Only Memory

Der Festspeicher besteht aus 16 Seiten, welche je 64 Wörter enthalten. Jedes Wort ist dabei 8 Bits groß. Es werden 4 verschiedene Register benutzt um den Speicher zu adressieren.

Das Page Adress Register(PA): Dieses Register enthält die aktuelle Seitenanzahl, an dem sich das Programm zur Zeit befindet. Das Register ist 4-Bits groß, um alle Seiten von 0 bis 15 adressieren zu können.
Das Page Buffer Register(PB): Das PB-Register wird mit einer neuen Seitenadresse geladen und nach einer erfolgreichen Branch oder Subroutinen-Operation wird die Adresse in das PA-Register geladen. Aus diesem Grund entspricht die Größe des Registers der des PA-Registers
Das Program Counter Register(PC): Dieses Register enthält das aktuelle Wort der Seite, in dem sich das Programm zur Zeit befindet. Um alle 64 Wörter adressieren zu können enthält das Register 6 Bits.
Das Subroutine Return Register(SR): In dem SR-Register wird bei einer Call Operation das aktuelle Wort des Porgramms gespeichert, um bei einer Return Instruktion zum ursprünglichen Wort zurückkehren zu können.

3.3.3.2 Branching & Subroutinen

Branching und Subroutinen sind ein essentieller Teil der im Ablauf eines Mikrocontroller-Programms. Ein Branch ist mit einer herkömmlichen if-Abfrage aus einer beliebigen Programmiersprache gleichzusetzen. Der Branch wird erfolgreich ausgeführt, wenn das Status-Logic-Bit gesetzt ist. Das Bit ist zwar standardmäßig gleich 1, kann durch Rechenoperationen, oder Vergleiche durch die Recheneinheit gleich 0 gesetzt werden. Nach einem Instruktionszyklus wird der Zustand wieder zurück gesetzt. Bei erfolgreicher Ausführung eines Branches wird das PB-Register in das PA-Register geladen.

Subroutinen sind sehr ähnlich zu Branch Instruktionen. Bei einer erfolgreichen Call-Instruktion springt das Programm, an eine andere Adresse im ROM. Bei einer Return-Instruktion kehrt das Programm zu dem urprünglichen Call-Statement zurück. Die Subroutine wird genau wie bei einem Branch nur ausgeführt, wenn das Status-Logic-Bit gesetzt ist. Zusätzlich muss jedoch das so genannte Call-Latch gesetzt sein. In diesem Fall wird der Inhalt des PA-Registers mit dem des PB-Registers vertauscht. Die Adresse des PC-Registers wird in dem SR-Register zwischengespeichert. Bei der ReturnInstruktion werden die temporär gespeicherten Adressen wieder in das PA- und PC-Register geladen. Es ist nicht möglich eine Call Instruktion innerhalb einer Call Instruktion aufzurufen, ohne dass die korrekte Return-Adresse verlorgen geht.

3.3.3.3 RAM - Random Access Memory

Der RAM-Speicher besteht aus 4 Dateien, welche jede 16 Wörter enthalten. Jedes Wort ist dabei 4 Bit groß. Der Speicher wird durch 2 Register X und Y adressiert. Das X-Register gibt dabei die Datei an, das Y-Register das Wort.
Der Input erfolt durch den Write-Multiplexer. Sowohl das Akkumulator Register, als auch die Constant and K Input Logic (CKI) können in den Speicher schreiben. Der Read-Bus übernimmt den Output des RAMs. Der Bus schreibt die das Wort entweder in den P-Multiplexer oder den N-Multiplexer, welche beide in die Recheneinheit führen. Die Recheneinheit leitet die Daten entweder in das Y oder Akkumulator Register.
Dem Programmierer des Controllers steht es frei, wie er sich den Speicher einteilt. Typischerweise werden die ersten 7 Wörter jeder Seite für als Register verwendet. Der Rest wird durch beispielsweise Pointer, Event Counter oder Flags befüllt. Diese sollten sich wenn möglich in der gleichen Datei gespeichert sein, in der auch die Register liegen mit dem sie interagieren.

3.3.3.4 Constant & K Input Logic

Die CKI Logik selektiert entweder die 4 K-Input Bits oder eine 4-Bit Konstante aus dem ROM und legt diese auf den CKI Daten Bus. Die Konstanten, welche aus dem ROM geliefert werden, werden für Befehle verwendet, welche Konstanten in ihrem Opcode enthalten. Zum Beispiel die Instruktion A8AAC. Diese Instruktion addiert die Zahl 8 zu dem Inhalt des Akkumulators und speichert das Ergebnis wiederum im Akkumulator. Der CKI Daten Bus kann die das 4-Bit Wort entweder in einen der beiden Adder leiten, die zur Recheneinheit führen, oder über den Write-Multiplexer in den RAM-Speicher.

3.3.3.5 Y-Register

Das Y-Register addressiert zusammen mit dem X-Register den RAM-Speicher. Außerdem addressiert das Register das R-Output Register, welches die individuellen Latches setzt und resettet. Das Y-Register wird zusätzlich als Arbeitsregister bezeichnet, da temporär ROM Wörter gespeichert werden können und sich das Register sehr gut als Counter eignet.

3.3.3.6 R- und O-Output

Der TMS-1000 hat zwei verschiede Arten an Output, R und O. Die R-Outputs werden für Kontroll Daten verwendet. Diese Kontroll Daten werden für externe Geräte verwendet oder Status-Logic-Outputs, wie zum Beispiel Overflow.
Das O-Output Register enthält den Inhalt des Akkumulators, sowie das Status-Logic-Bit. Durch ein PLA[1] werden die 5 Bits zu den 8 Output-Bits kodiert. Der Programmierer kann dabei selbst bestimmen wie er die Ein- und Ausgänge verbindet. Das O-Output Register wird generell für die Ausgabe von Daten benutzt im Gegensatz zum R-Output Register.

3.3.3.7 Akkumulator & ALU - Aritmethic Logic Unit

Das Akkumulator Register ist sehr wichtig, da es mit der ALU dem RAM-Speicher und dem O-Output Register interagiert. Hauptsächlich wird das Register für Additionen und Substraktionen verwendet. Außerdem werden im Akkumulator Konstanten zwischengespeichert und alle Variablendaten, die in den RAM-Speicher geschrieben werden sollen, müssen den Akkumulator durchlaufen.
Die ALU ist ein 4-Bit Adder und Komparator. Er kann addieren, subtrahieren und

[1] Programmable Logic Array

logische Vergleiche. Seine Inputs bekommt die Einheit durch 2 4-Bit Multiplexer, die ihren Inhalt durch das Y-Register, die CKI, den RAM-Speicher oder den Akkumulator bekommen. Das Ergebnis wird je nach Instruktion entweder im Y-Register oder Akkumulator gespeichert. Die Substraktion wird durch die two's complement arithmetic ausgeführt. Bei dieser Arithmetik wird der Subtrahend invertiert und auf den Minuend addiert.
Die ALU setzt außerdem das Status-Logic-Bit. Das Bit kann nicht nur bei logischen Vergleichen gesetzt werden, sonder kann auch bei arithmetischen Operationen als Carry gesetzt werden für beispielsweise Überträge.

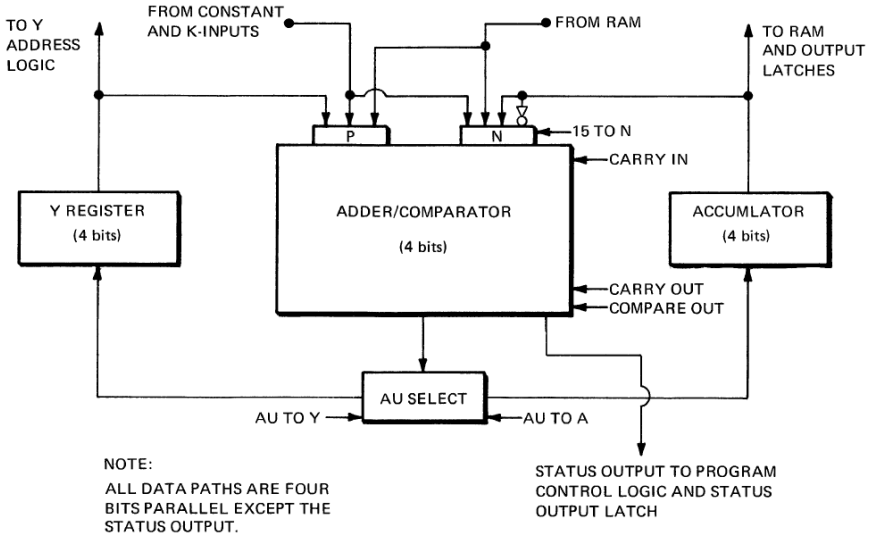


Figure 3.2: Eine Skizze der arithmetisch logischen Einheit

3.3.3.8 Instruction-Decoders

Der TMS-1000 enthält zwei logische Blöcke, die jeweils die 8-Bit Instruktionen in die verschiedenen Mikroinstruktionen dekodiert. Der erste Decokodierer, der "'Fixed instruction decoder"', kann nicht modifiziert werden und aktiviert 12 fixierte Mikroinstruktionen.
Die restlichen 31 Basis Instruktionen sind die programmierbaren Instruktionen. Diese werden durch den zweiten Dekodierer, den "'Programmable instruction PLA"', definiert. Standardmäßig sind diese für den Programmierer vordefiniert. Sie können aber auch durch das umprogrammieren des PLAs umdefiniert werden. Dabei stehen dem Dekodierer 16 Mikroinstruktionen zur Verfügung, die beliebig kombiniert werden können.

Chapter 4

Schluss

Test Schluss

List of Figures