

Die ersten Mikroprozessoren: ein Rechner, ein Chip

Michael Kiener, Michael Kratzer

13. Juni 2016

Agenda

1 Der Intel 4004

2 The TMS-1000

Der Intel 4004

Agenda

① Der Intel 4004

- Geschichte des MCS-4
- MCS-4
 - Der 4001
 - Der 4002
 - Der 4003
 - Der 4004
- Funktionsweise des 4004
- Auswirkungen auf die Prozessorentwicklung

Geschichte des MCS-4

1968 - Gründung von Intel

- Gegründet von Gordon E. Moore und Robert Noyce
- Hersteller von Speicherchips auf Halbleiterbasis
- Problem: Kunden zögerten von magnetischen Speichern zu wechseln



Geschichte des MCS-4

1969 - Auftrag von Busicom

- Spezialauftrag für Busicoms Rechenmaschine
- Erstes Design:
 - 7 - 12 Chips mit geschätzt 3000 - 5000 Transistoren und 40 Pins
 - Schieberegister Speicher
 - komplexes Makroinstruktionsset
- Intel hatte keine Erfahrung mit Design von Logikchips

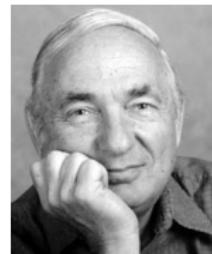


Busicom Unicom 141P

Geschichte des MCS-4

Neues Design von Intel

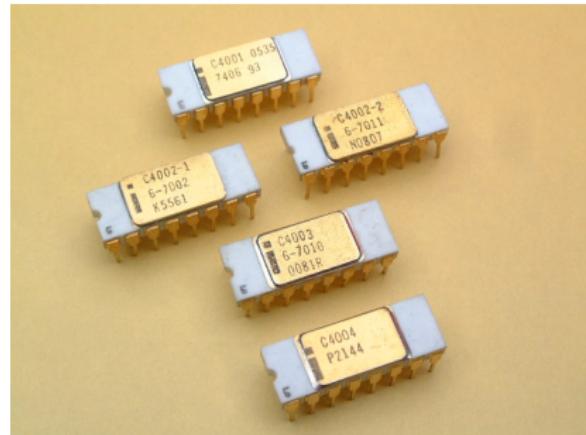
- Reduzierung der Komplexität mit Hilfe von größerem Speicher
- Mikroinstruktionen
- 4-bit Binary Coded Decimal Arithmetik
- 3 Chip Design bestehend aus CPU, RAM und ROM
- 1900 Transistoren pro Chip
- Vergleich zweites Busicom Design: 12 Chips mit 2000 Transistoren und 40 Pins



Geschichte des MCS-4

1970 - Entscheidung und Produktion

- Busicom entscheidet sich Intels Designvorschlag anzunehmen
- Intel stellt Federico Faggin als Logik- und Chipdesigner ein
- Masatoshi Shima arbeitet an den ersten Programmen für den neuen Prozessor
- erster Prototyp nach 9 Monaten fertig

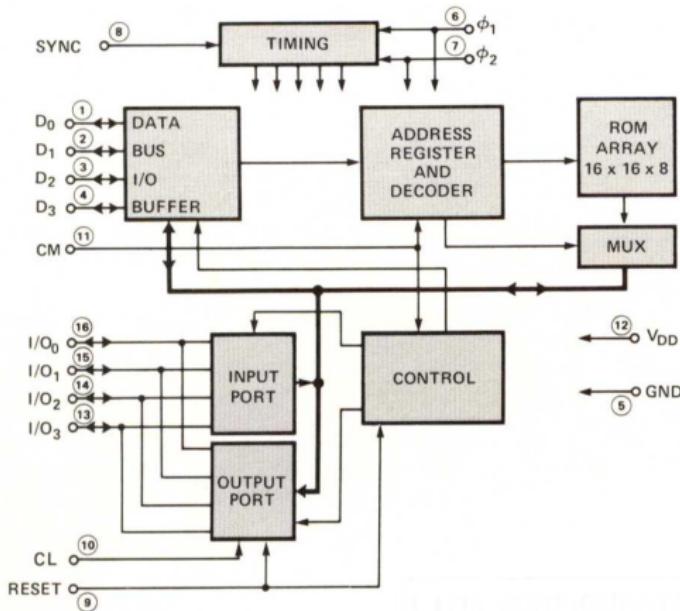


Intel MCS-4

Der 4001

Programmierbarer ROM

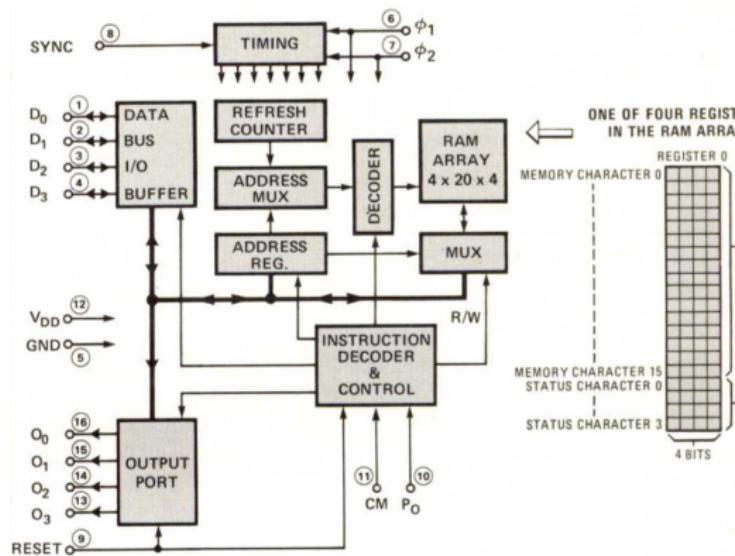
- Programmspeicher
- 2048 bits als 256 8-bit Wörter
- maximal 16 ROMs hintereinander schaltbar
- 2. Operationsmodus: I/O Befehle



Der 4002

Random-Access Memory

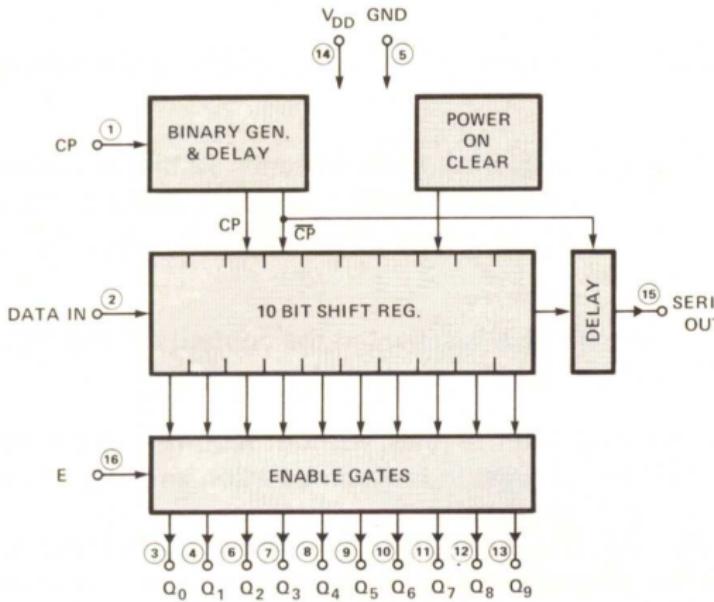
- 2 Versionen: 4002-1 und 4002-2
- Datenspeicher für Berechnungen
- 320 bits als 4 Register von 20 4-bit character
 - 16 Daten character
 - 4 Status character



Der 4003

I/O Schieberegister

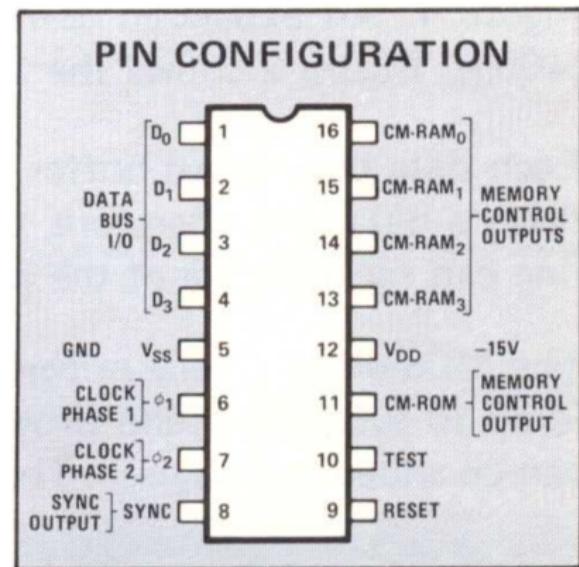
- 10-bit statisches Schieberegister
- Eingabe: seriell
- Ausgabe: seriell, parallel
- Erweitert ROM und RAM I/O Ports
- Können hintereinander geschaltet werden
- Ermöglicht das Anschließen von Tastaturen, Displays, Druckern, etc



Der 4004

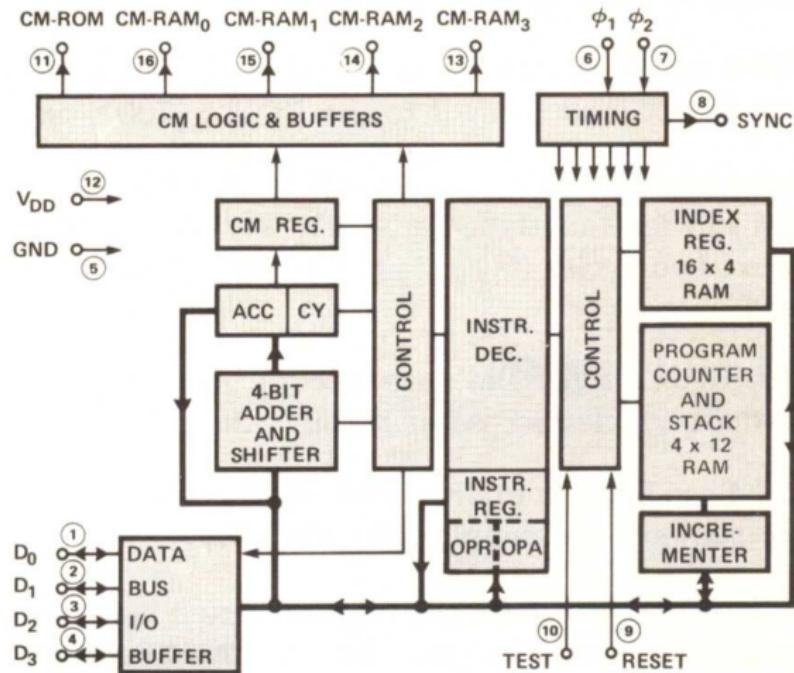
Die CPU

- 4 funktionale Blöcke
 - Adressregister
 - Indexregister
 - 4-bit Addierer
 - Instruktionsregister



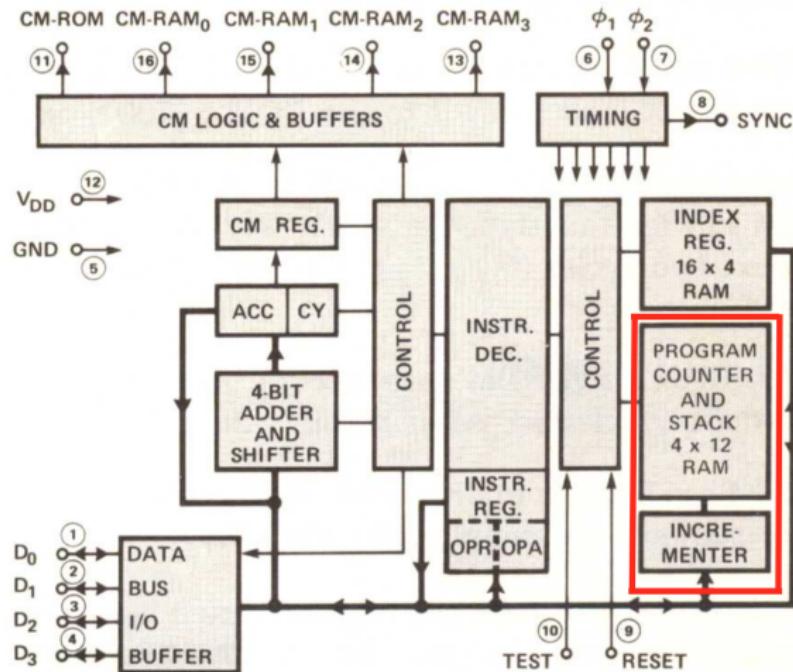
Der 4004

Detailansicht des 4004



Der 4004

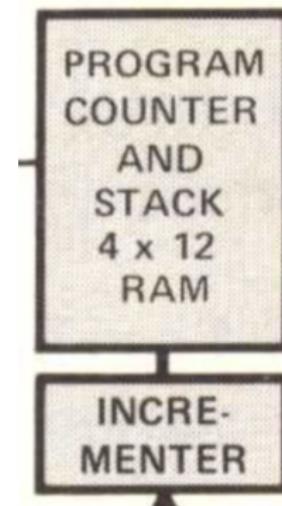
Detailansicht des 4004



Der 4004

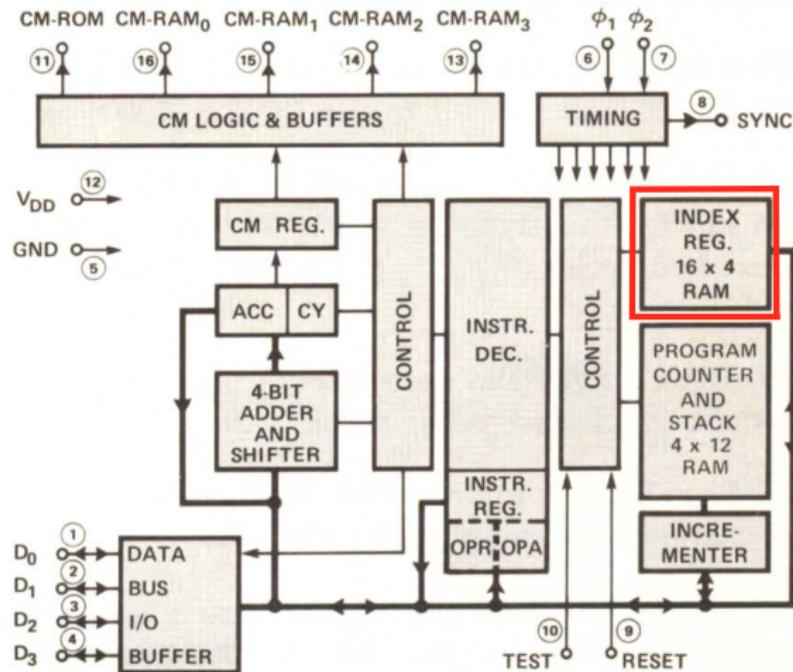
Adressregister

- 4 12bit Register
- Organisiert als Stack
- Speichert die aktuelle Programmadresse
- Zusätzliche 3 Level für Subroutinen



Der 4004

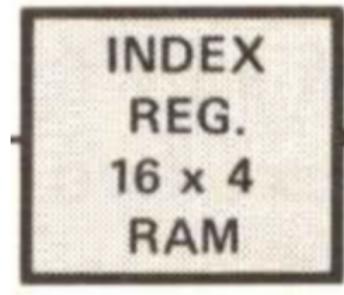
Detailansicht des 4004



Der 4004

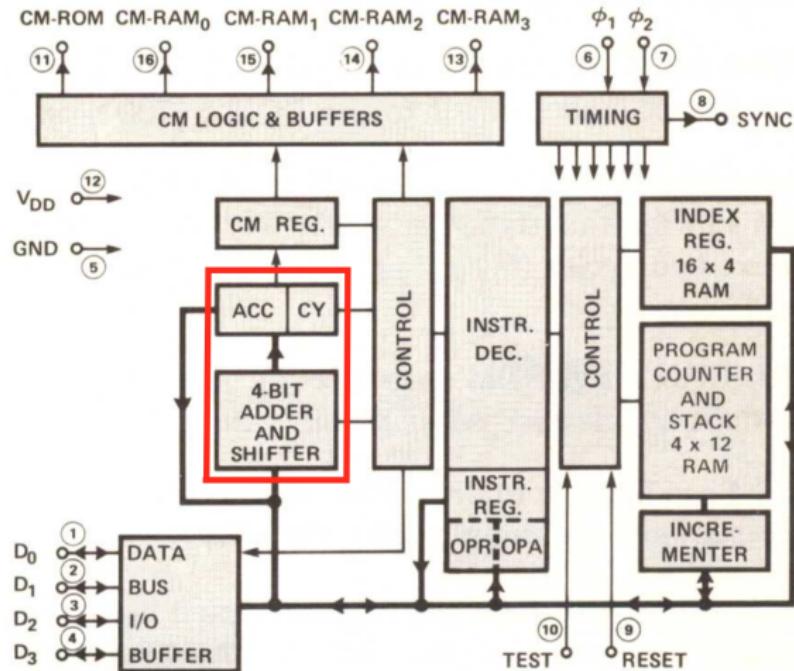
Indexregister

- Organisiert in 8 Reihen mit jeweils 8 bit
- Möglichkeit eine Reihe von 8bit oder jedes Wort einzeln auszulesen
- Cache für Zwischenergebnisse oder Instruktionen



Der 4004

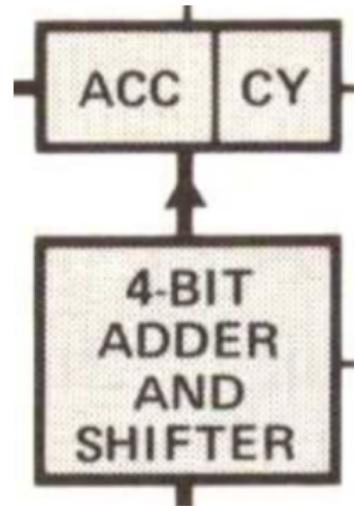
Detailansicht des 4004



Der 4004

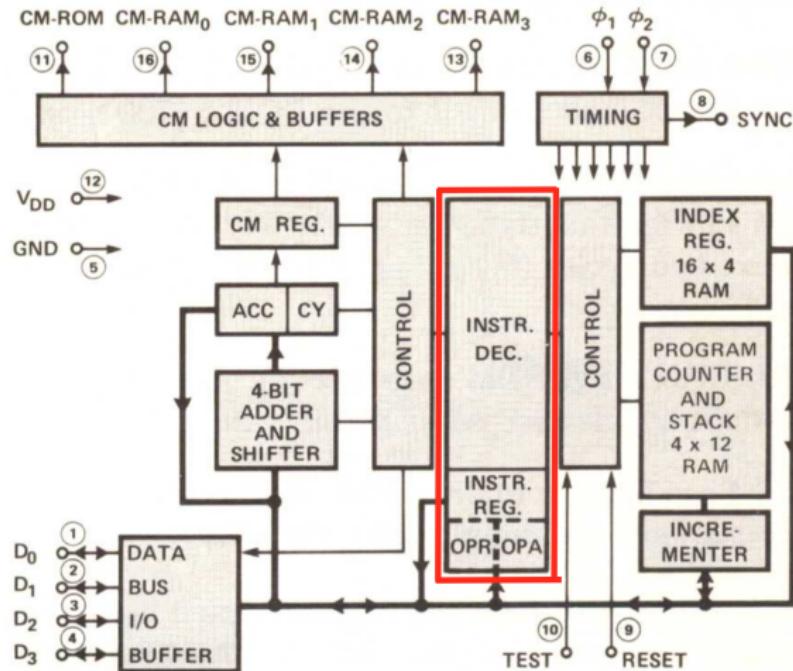
4-bit Addierer

- Kann sowohl in binär als auch in BCD Arithmetik rechnen
- Erster Term aus einem internen Bufferregister
- Carry und 2. Term aus dem Akkumulator
- Ergebnis wird im Akkumulator gespeichert



Der 4004

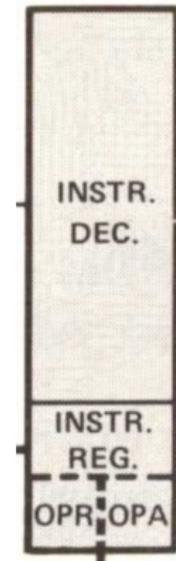
Detailansicht des 4004



Der 4004

Instruktionsregister

- OPR und OPA
- Enthält die aktuelle Instruktion
- Enthält zusätzliche Einheit zur Dekodierung von Instruktionen



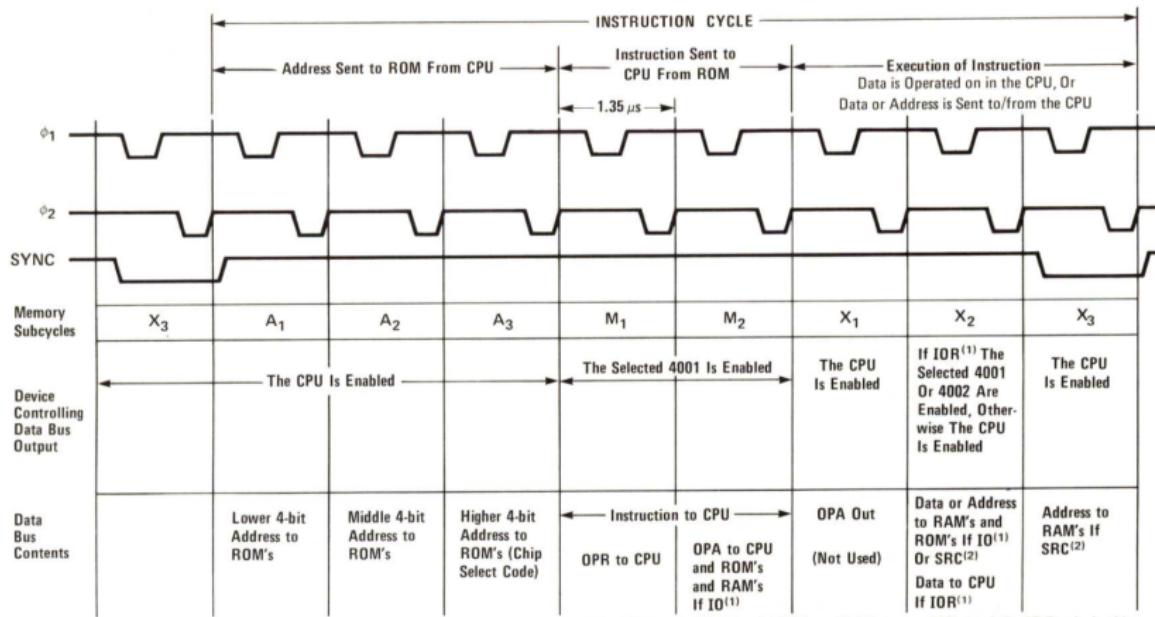
Funktionsweise des 4004

Architektur

- Mischung aus Harvard und von Neumann Architektur
- Besteht aus 1 Prozessor, 1-16 ROMs und 0-16 RAMs
- 4-bit breiter Datenbus zwischen den Chips
- Kontrollleitungen zur Adressierung oder Reset

Funktionsweise des 4004

Instruktionszyklus



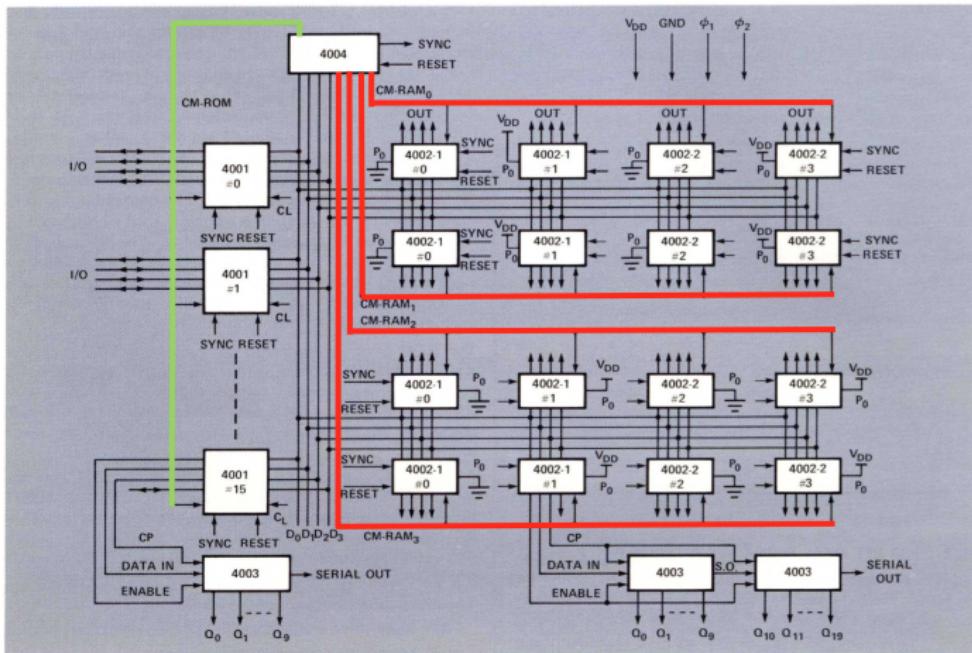
Funktionsweise des 4004

Addressierung von ROM und RAM

- ROM
 - In A1 und A2 verschicken der 8bit Adresse
 - A3 Verschicken der Chipnummer
- RAM
 - RAM Bank wird ausgewählt durch CM-Signal
 - In X2 wird die Chipnummer und die Registernummer verschickt
 - Erinnerung: Ein 4002 ist in 4 Register aufgeteilt
 - In X3 wird die Adresse innerhalb des Registers ausgewählt

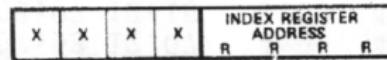
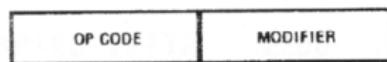
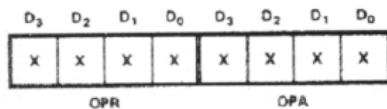
Funktionsweise des 4004

MCS-4 Gesamtübersicht

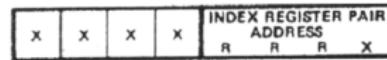


Funktionsweise des 4004

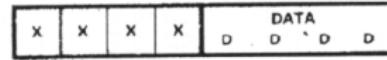
Instruktionen



OR



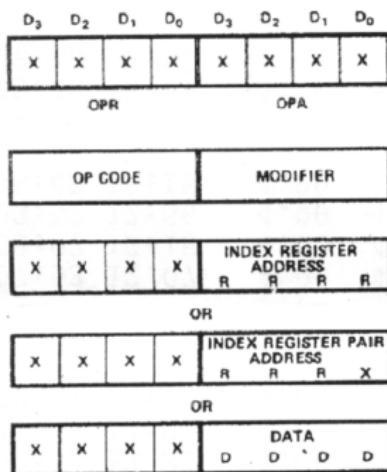
OR



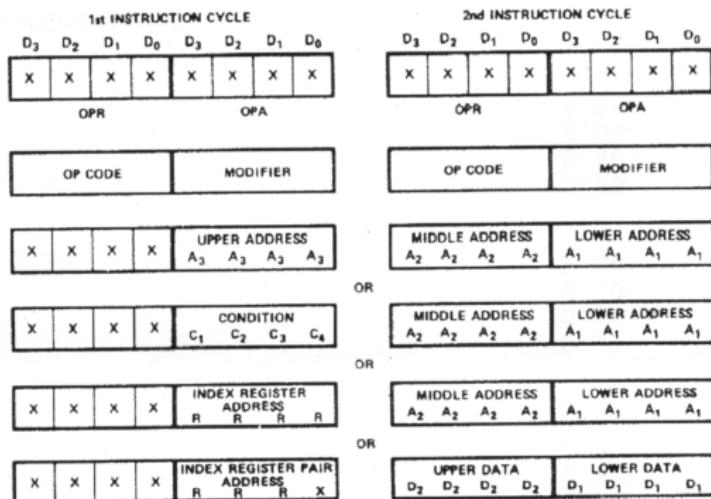
1 Wort Befehle

Funktionsweise des 4004

Instruktionen



1 Wort Befehle



2 Wort Befehle

Funktionsweise des 4004

Befehlssatz - Maschienbefehle

MNEMONIC	OPR D ₃ D ₂ D ₁ D ₀	OPA D ₃ D ₂ D ₁ D ₀	DESCRIPTION OF OPERATION
NOP	0 0 0 0	0 0 0 0	No operation.
*JCN	0 0 0 1 A ₂ A ₂ A ₂ A ₂	C ₁ C ₂ C ₃ C ₄ A ₁ A ₁ A ₁ A ₁	Jump to ROM address A ₂ A ₂ A ₂ A ₂ , A ₁ A ₁ A ₁ A ₁ (within the same ROM that contains this JCN instruction) if condition C ₁ C ₂ C ₃ C ₄ ⁽¹⁾ is true, otherwise skip (go to the next instruction in sequence).
*FIM	0 0 1 0 D ₂ D ₂ D ₂ D ₂	R R R 0 D ₁ D ₁ D ₁ D ₁	Fetch immediate (direct) from ROM Data D ₂ , D ₁ to index register pair location RRR. ⁽²⁾
SRC	0 0 1 0	R R R 1	Send the address (contents of index register pair RRR) to ROM and RAM at X ₂ and X ₃ time in the Instruction Cycle.
FIN	0 0 1 1	R R R 0	Fetch indirect from ROM, Send contents of index register pair location 0 out as an address. Data fetched is placed into register pair location RRR at A ₁ and A ₂ time in the Instruction Cycle.
JIN	0 0 1 1	R R R 1	Jump indirect, Send contents of register pair RRR out as an address at A ₁ and A ₂ time in the Instruction Cycle.
*JUN	0 1 0 0 A ₂ A ₂ A ₂ A ₂	A ₃ A ₃ A ₃ A ₃ A ₁ A ₁ A ₁ A ₁	Jump unconditional to ROM address A ₃ , A ₂ , A ₁ .
*JMS	0 1 0 1 A ₂ A ₂ A ₂ A ₂	A ₃ A ₃ A ₃ A ₃ A ₁ A ₁ A ₁ A ₁	Jump to subroutine ROM address A ₃ , A ₂ , A ₁ , save old address. (Up 1 level in stack.)
INC	0 1 1 0	R R R R	Increment contents of register RRRR. (3)
*ISZ	0 1 1 1 A ₂ A ₂ A ₂ A ₂	R R R R A ₁ A ₁ A ₁ A ₁	Increment contents of register RRRR. Go to ROM address A ₂ , A ₁ (within the same ROM that contains this ISZ instruction) if result ≠ 0, otherwise skip (go to the next instruction in sequence).
ADD	1 0 0 0	R R R R	Add contents of register RRRR to accumulator with carry.
SUB	1 0 0 1	R R R R	Subtract contents of register RRRR to accumulator with borrow.
LD	1 0 1 1	R R R R	Load contents of register RRRR to accumulator.
XCH	1 0 1 1	R R R R	Exchange contents of index register RRRR and accumulator.
BBL	1 1 0 0	D D D D	Branch back (down 1 level in stack) and load data DDDD to accumulator.
LDM	1 1 0 1	D D D D	Load data DDDD to accumulator.

Intel 4004 Spezifikation

- Technik: PMOS Transistoren
- Strukturbreite: $10\mu\text{m}$
- Transistorzahl: 2300
- Taktfrequenz: 740 KHz
- Dauer eines Befehlszyklus: $10.8 \mu\text{s}$
- 92000 Befehle pro Sekunde
- Addition 2er 8 stelligen Zahlen: $850 \mu\text{s}$

Auswirkungen auf die Prozessorentwicklung

- Intel zögert Rechte von Busicom zurückzukaufen, da Prozessormarkt zu klein
- “Revolution“ erst nachdem Intel weitere Developer Tools zur Verfügung stellt
- Weiterproduziert bis 1986
- Basis für den Intel 8080
- Intel hat heute einen Marktanteil von 80% bei PC-Mikroprozessoren

Geschichte des MCS-4

1971 - MCS-4 Werbung

Announcing a new era of integrated electronics

Intel introduces an integrated CPU compatible with a 4-bit parallel input-output system. It's a micro-programmable computer on a chip. It's one of a family of four new ICs which comprise the MCS-4 micro-computer system. The 4004 offers the same power and flexibility of a dedicated general-purpose computer at low cost in as few as five dual-in-line packages.

MCS-4 systems provide complete computing and control functions for applications such as printing machines, measuring systems, numeric control systems and process control. The heart of any MCS-4 system is a Type 4004 CPU, which includes a powerful set of 45 instructions. Adding the 4004 to a memory system and data tables gives you a fully functioning micro-programmable computer. To this you may add Type 4004 ROMs to expand the output ports.

Using any one of the four ICs from this family of four, you can create a system with 4096 8-bit bytes of ROM storage and \$1.00 lire of RAM storage. When you add a monitor, keyboard, printer or other free options, Intel's erasable and re-programmable ROM, Type 1750, can be customized for the Type 4004 micro-programmed ROM.

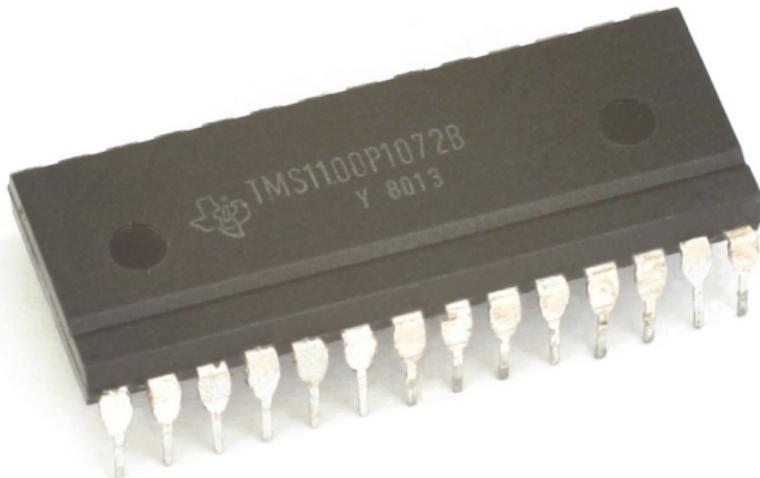
MCS-4 systems interface easily with serial printers, tape drives, disk drives, monitors, readers, A/D converters and other popular peripherals.

The 4004 is available from Intel's Santa Clara headquarters and at our marketing headquarters in Europe and Japan. In the U.S., contact your local Intel representative or write to Intel Corporation, Marketing Department, One Market Place, Suite 1000, San Jose, California 95111. In Europe, contact Intel at Avenue Louise 216, B-1050 Brussels, Belgium. In Japan, contact Intel Japan, Inc., Parciale First Bldg, No. 4-2, Minami-Aoyama 1-chome, Minato-ku, Tokyo 107-0047. Intel Corporation now produces micro-computers, memory devices and memory systems at 3005 Bowers Avenue, Santa Clara, Calif. 95051. Phone (408) 248-7501.

A micro-programmable computer on a chip!

intel delivers.

Der erste Mikrocontroller - System-on-a-Chip



Inhalt

1 Geschichte

Inhalt

- ① Geschichte
- ② Mikrokontroller allgemein

Inhalt

- ① Geschichte
- ② Mikrokontroller allgemein
- ③ TMS1000

Inhalt

- ① Geschichte
- ② Mikrokontroller allgemein
- ③ TMS1000
 - Allgemeine Daten

Inhalt

- ① Geschichte
- ② Mikrokontroller allgemein
- ③ TMS1000
 - Allgemeine Daten
 - Aufbau & Funktionsweise

Inhalt

- ① Geschichte
- ② Mikrokontroller allgemein
- ③ TMS1000
 - Allgemeine Daten
 - Aufbau & Funktionsweise
 - Befehlssatz

Inhalt

- ① Geschichte
- ② Mikrokontroller allgemein
- ③ TMS1000
 - Allgemeine Daten
 - Aufbau & Funktionsweise
 - Befehlssatz
- ④ Verwendung des TMS1000

Entstehung

- Entwickelt von Gary Boone & Michael Cochran 1971

Entstehung

- Entwickelt von Gary Boone & Michael Cochran 1971
- Zuerst von Texas Instrument verwendet

Entstehung

- Entwickelt von Gary Boone & Michael Cochran 1971
- Zuerst von Texas Instrument verwendet
- Erst 1974 auf dem freien Markt erhältlich

Entstehung

- Entwickelt von Gary Boone & Michael Cochran 1971
- Zuerst von Texas Instrument verwendet
- Erst 1974 auf dem freien Markt erhältlich
- Bis heute ca. 100 Millionen verkaufte Exemplare

Mikrokontroller allgemein

Aufbau:

- CPU - Prozessor

Mikrokontroller allgemein

Aufbau:

- CPU - Prozessor
- RAM - Arbeitsspeicher

Mikrokontroller allgemein

Aufbau:

- CPU - Prozessor
- RAM - Arbeitsspeicher
- ROM - Festspeicher

Mikrokontroller allgemein

Aufbau:

- CPU - Prozessor
- RAM - Arbeitsspeicher
- ROM - Festspeicher
- Takt

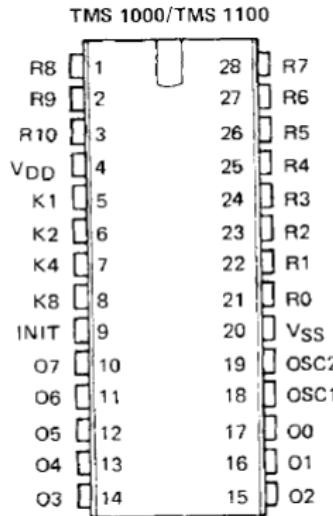
Mikrokontroller allgemein

Aufbau:

- CPU - Prozessor
- RAM - Arbeitsspeicher
- ROM - Festspeicher
- Takt
- Peripherie - I/O-Ports

Allgemeine Daten - TMS1000

Abbildung der Pins



Allgemeine Daten - TMS1000

- ROM: 1024x8 Bits

Allgemeine Daten - TMS1000

- ROM: 1024x8 Bits
- RAM: 64x4 Bits

Allgemeine Daten - TMS1000

- ROM: 1024x8 Bits
- RAM: 64x4 Bits
- 43 Basis Instruktionen

Allgemeine Daten - TMS1000

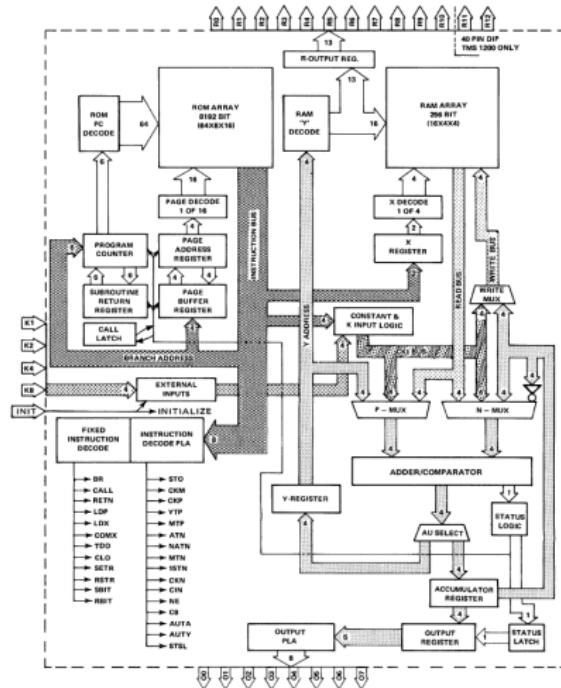
- ROM: 1024x8 Bits
- RAM: 64x4 Bits
- 43 Basis Instruktionen
- Max. Spannung 20V

Allgemeine Daten - TMS1000

- ROM: 1024x8 Bits
- RAM: 64x4 Bits
- 43 Basis Instruktionen
- Max. Spannung 20V
- Höchste erreichbare Frequenz: 0,4MHz

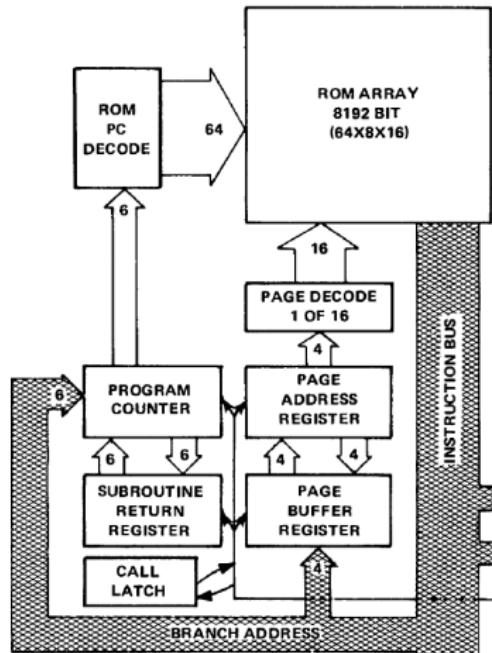
Aufbau & Funktionsweise

Schaltbild des TMS-1000



Aufbau & Funktionsweise

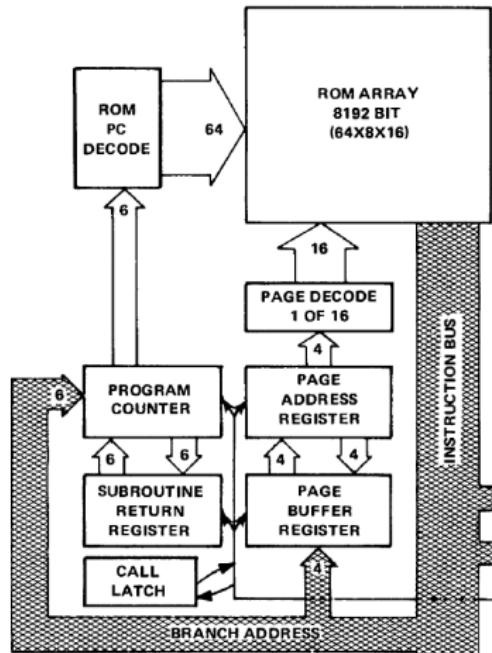
ROM



- 16 Seiten je 64 Wörter

Aufbau & Funktionsweise

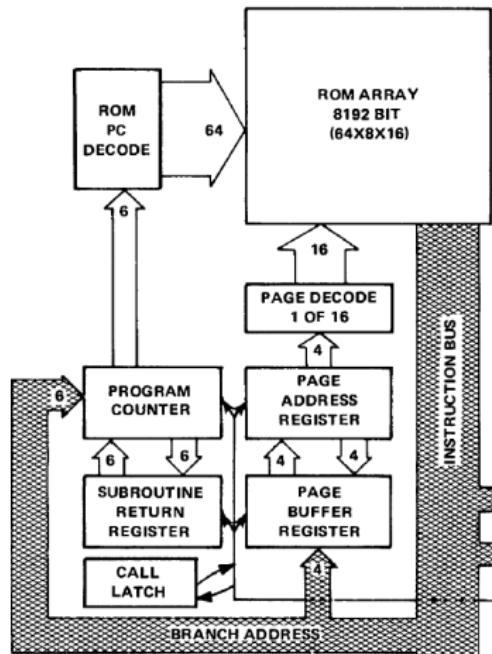
ROM



- 16 Seiten je 64 Wörter
- 3 Register die den ROM adressieren

Aufbau & Funktionsweise

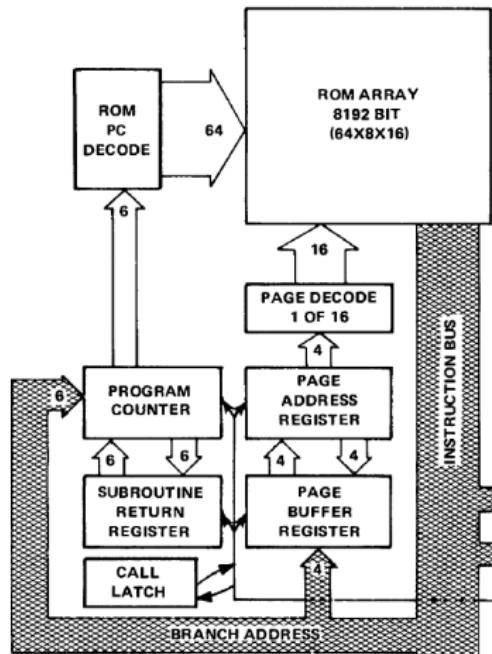
ROM



- 16 Seiten je 64 Wörter
- 3 Register die den ROM adressieren
- Page Adress (PA)

Aufbau & Funktionsweise

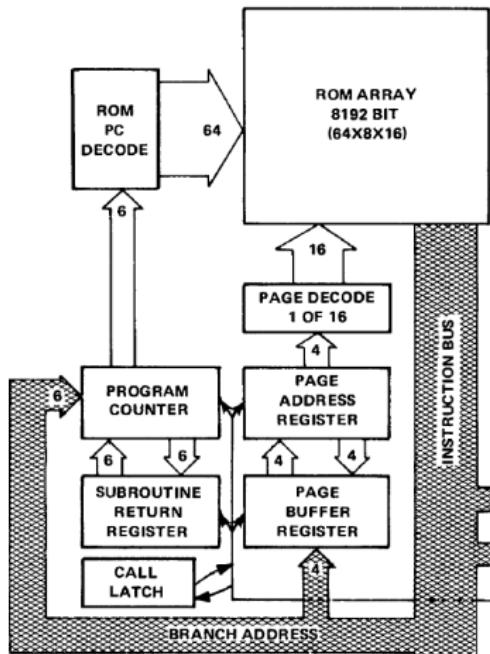
ROM



- 16 Seiten je 64 Wörter
- 3 Register die den ROM adressieren
- Page Adress (PA)
- Page Buffer (PB)

Aufbau & Funktionsweise

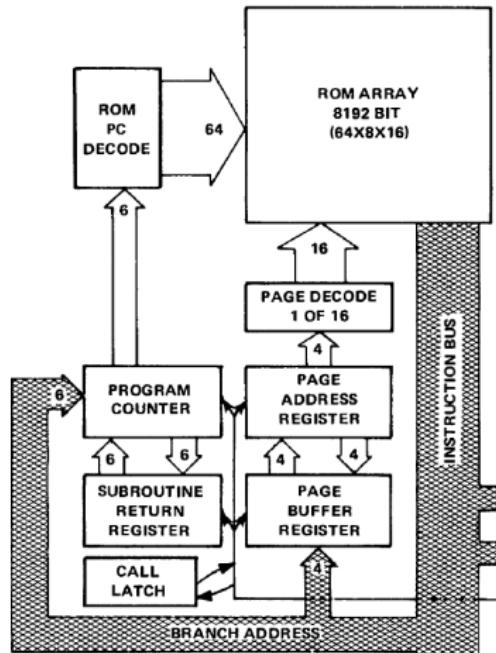
ROM



- 16 Seiten je 64 Wörter
- 3 Register die den ROM adressieren
- Page Adress (PA)
- Page Buffer (PB)
- Program Counter (PC)

Aufbau & Funktionsweise

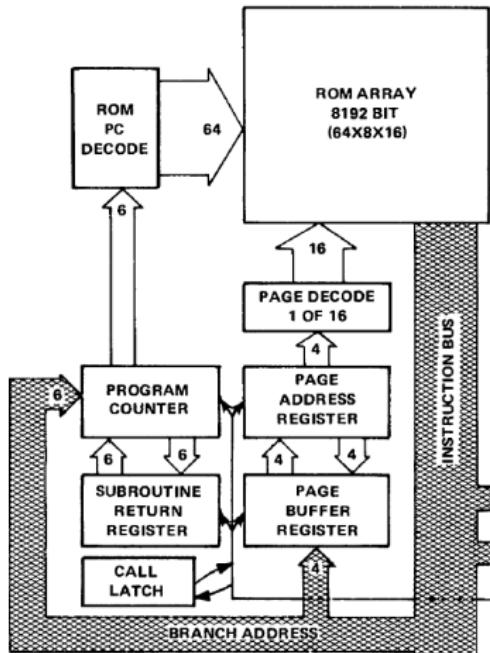
Branching & Subroutines



- Conditional

Aufbau & Funktionsweise

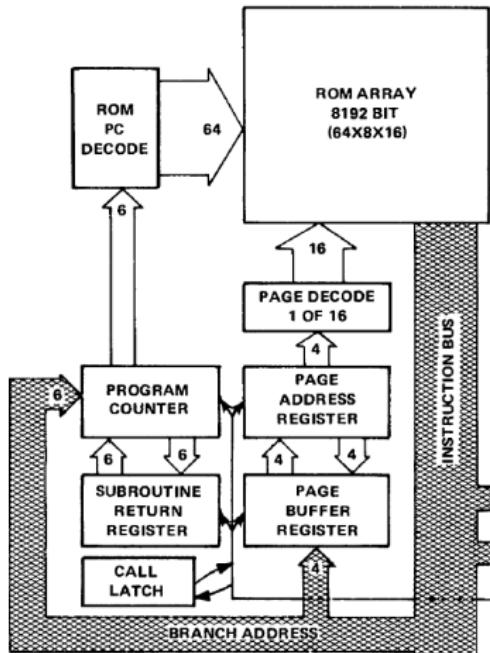
Branching & Subroutines



- Conditional
- ALU setzt Status-Logic Bit

Aufbau & Funktionsweise

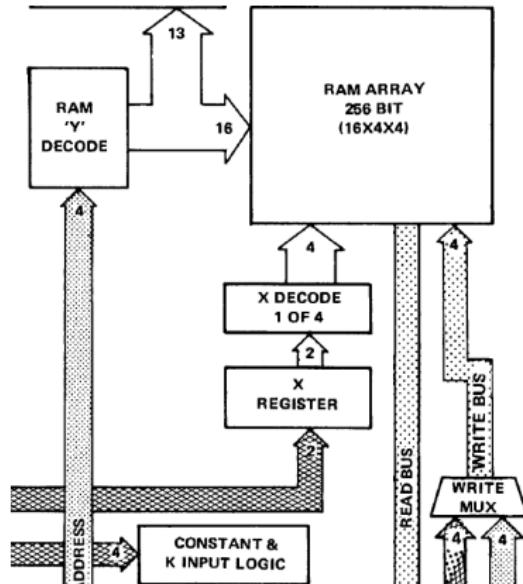
Branching & Subroutines



- Conditional
- ALU setzt Status-Logic Bit
- Standardmäßig auf 1

Aufbau & Funktionsweise

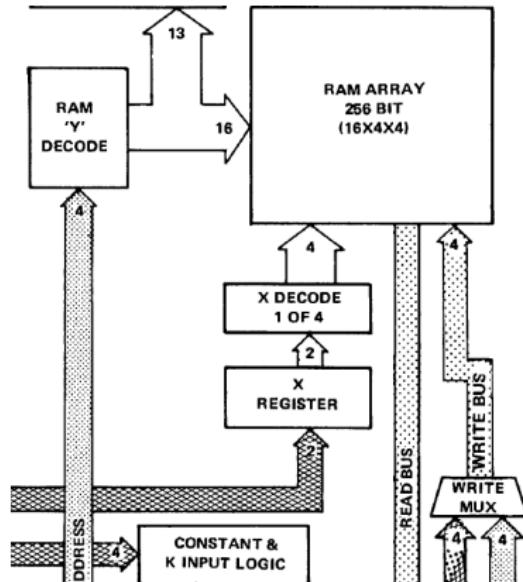
RAM



- 4 Dateien mit je 16 Wörtern

Aufbau & Funktionsweise

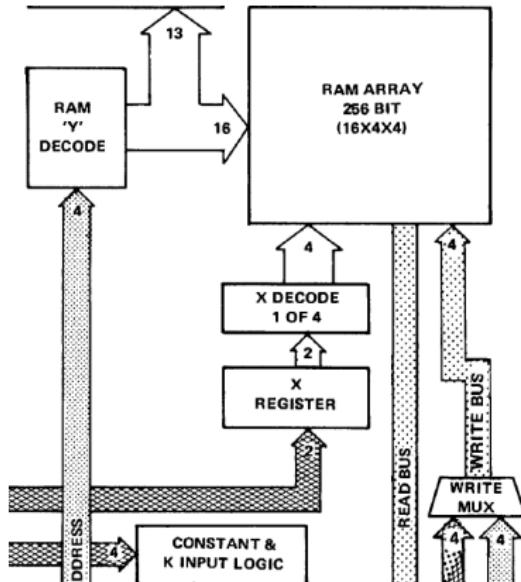
RAM



- 4 Dateien mit je 16 Wörtern
- Wird durch X und Y Register adressiert

Aufbau & Funktionsweise

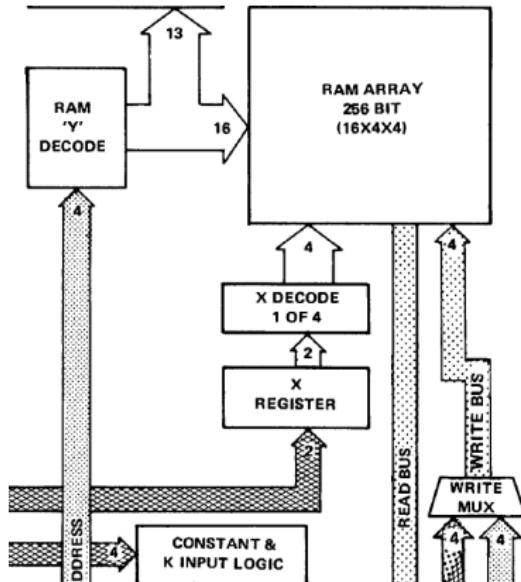
RAM



- 4 Dateien mit je 16 Wörtern
- Wird durch X und Y Register adressiert
- X adressiert „Seite“, Y das Wort

Aufbau & Funktionsweise

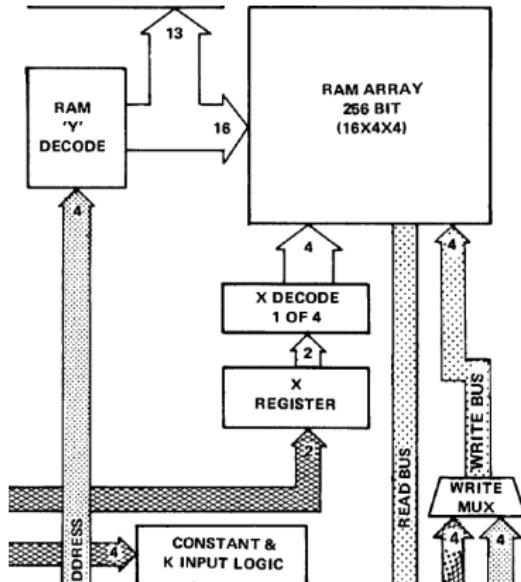
RAM



- 4 Dateien mit je 16 Wörtern
- Wird durch X und Y Register adressiert
- X adressiert „Seite“, Y das Wort
- Input durch Write-Multiplexer

Aufbau & Funktionsweise

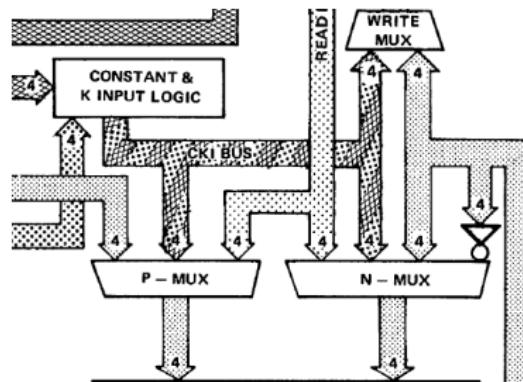
RAM



- 4 Dateien mit je 16 Wörtern
- Wird durch X und Y Register adressiert
- X adressiert „Seite“, Y das Wort
- Input durch Write-Multiplexer
- Output durch Read-Bus

Aufbau & Funktionsweise

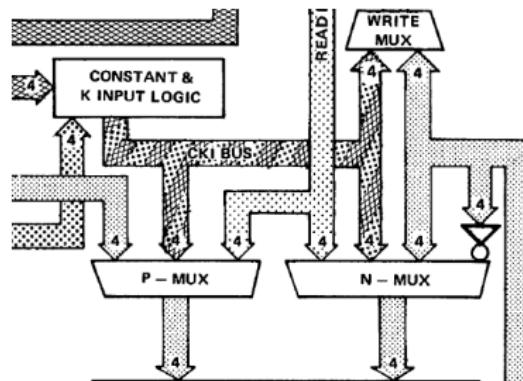
Constant and K Input (CKI) Logic



- Verwendet entweder K-Inputs oder Konstanten aus dem ROM

Aufbau & Funktionsweise

Constant and K Input (CKI) Logic



- Verwendet entweder K-Inputs oder Konstanten aus dem ROM
- Kann entweder in P-MUX, N-MUX oder RAM schreiben

Aufbau & Funktionsweise

Y-Register

- Adressiert RAM

Aufbau & Funktionsweise

Y-Register

- Adressiert RAM
- Setzt R-Output Latches

R-Output

- Externe Geräte

R-Output

- Externe Geräte
- Display Scans

R-Output

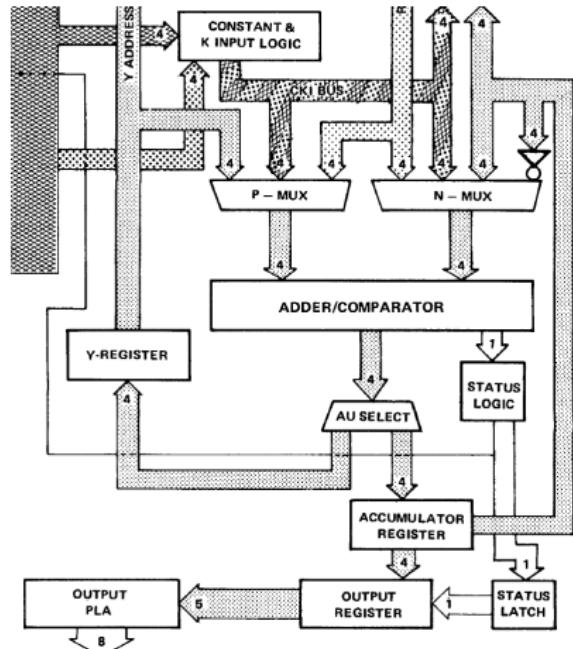
- Externe Geräte
- Display Scans
- Input Encoding

R-Output

- Externe Geräte
- Display Scans
- Input Encoding
- Status Logic Outputs

Aufbau & Funktionsweise

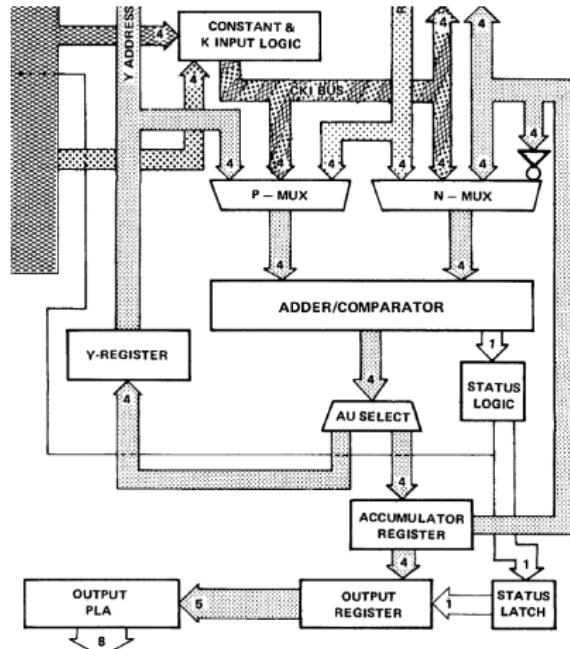
Akkumulator & ALU



- 4-Bit Adder/Comperator

Aufbau & Funktionsweise

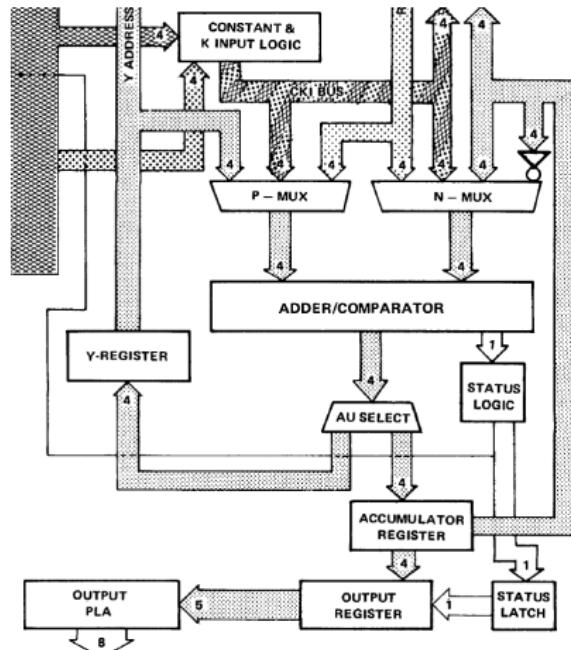
Akkumulator & ALU



- 4-Bit Adder/Comperator
- Kann addieren, subtrahieren und logische Vergleiche

Aufbau & Funktionsweise

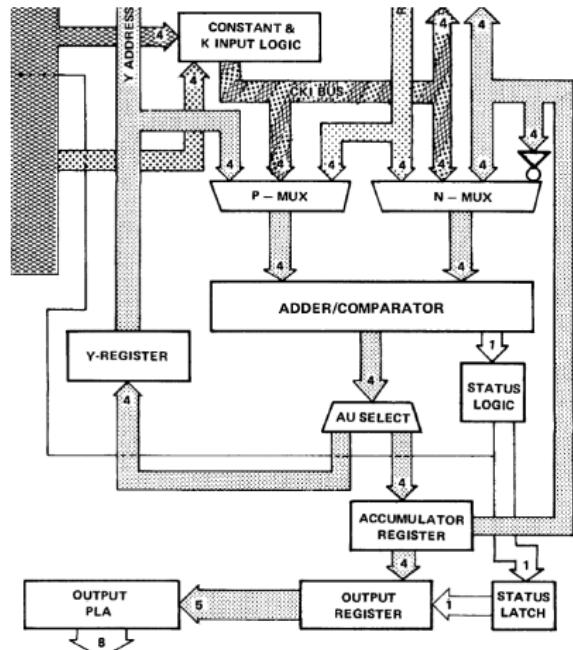
Akkumulator & ALU



- 4-Bit Adder/Comperator
- Kann addieren, subtrahieren und logische Vergleiche
- Addition/Substraktion entweder im Y-Register oder Akkumulator gespeichert

Aufbau & Funktionsweise

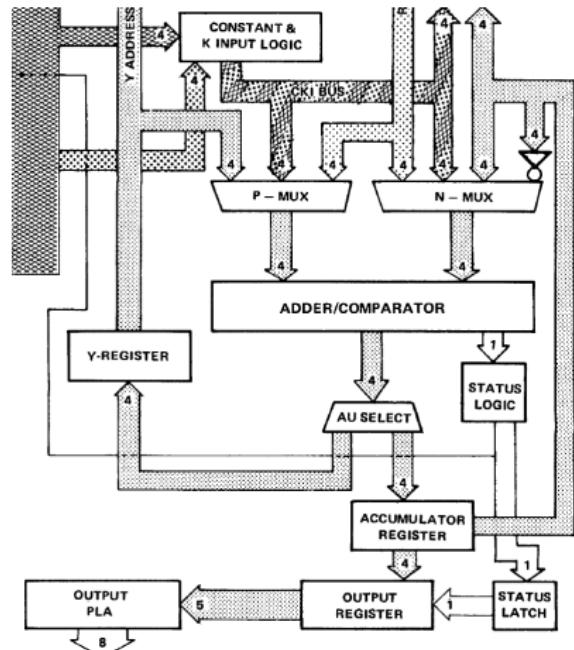
Akkumulator & ALU



- 4-Bit Adder/Comperator
- Kann addieren, subtrahieren und logische Vergleiche
- Addition/Substraktion entweder im Y-Register oder Akkumulator gespeichert
- Ergebnis Vergleiche im Status-Logic-Bit

Aufbau & Funktionsweise

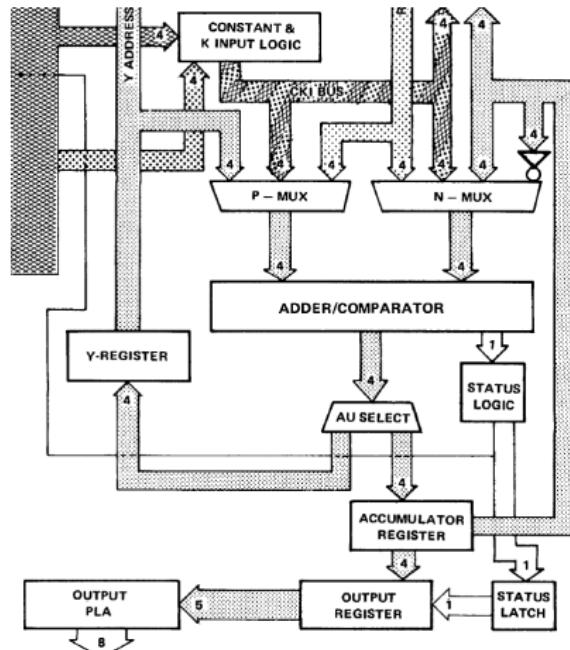
O-Output



- Gibt Akkumulator und Status-Bit aus

Aufbau & Funktionsweise

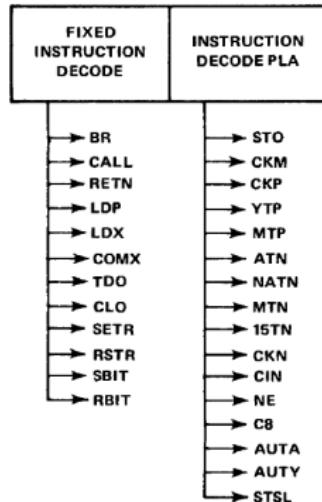
O-Output



- Gibt Akkumulator und Status-Bit aus
- PLA kodiert 5 Input-Bits in 8 Ausgänge

Aufbau & Funktionsweise

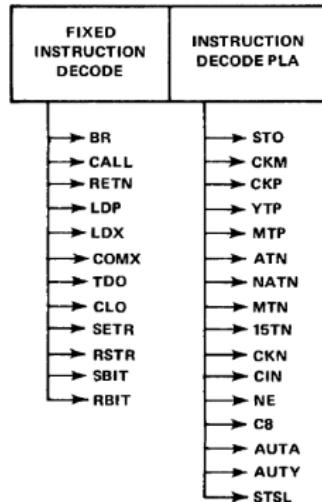
Instruction-Decoder



- 12 feste Instruktionen

Aufbau & Funktionsweise

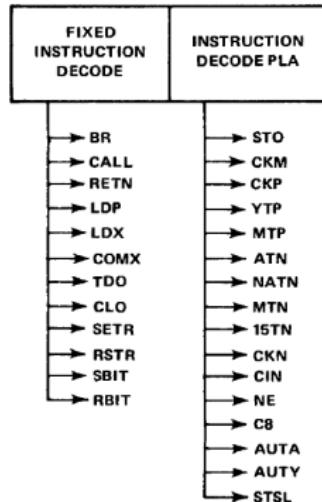
Instruction-Decoder



- 12 feste Instruktionen
- 31 programmierbare Instruktionen

Aufbau & Funktionsweise

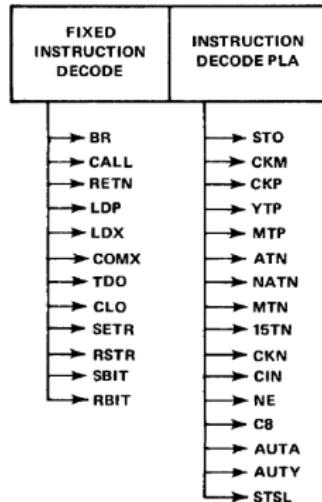
Instruction-Decoder



- 12 feste Instruktionen
- 31 programmierbare Instruktionen
- Sind standardmäßig definiert, können aber abgeändert werden

Aufbau & Funktionsweise

Instruction-Decoder



- 12 feste Instruktionen
- 31 programmierbare Instruktionen
- Sind standardmäßig definiert, können aber abgeändert werden
- PLA kombiniert die 16 Mikroinstruktionen zu einem Ausdruck

Aufbau & Funktionsweise

INIT-Pin

- Initialisiert die Hardware

Aufbau & Funktionsweise

INIT-Pin

- Initialisiert die Hardware
- Resettet PA, PC und die R- & O-Output Register

Aufbau & Funktionsweise

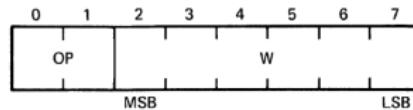
Befehlssatz

Mnemonic	Opcodes	Microinstructions		Reference Paragraph
		Fixed	Programmable	
ALEC	0 1 1 1 C	CKP, NATN, CIN, CR		4.5.2
ALEM	0 0 1 0 1 0 0 0 1	MTP, NATN, CIN, CR		4.5.1
AAEAC	0 0 1 0 0 0 1 0 1	MTP, ATN, CR, AUTA		4.4.1
AGAAC	0 0 0 0 0 0 0 1 0	CKP, ATN, CEAUTA		4.4.11
ABAAC	0 0 0 0 0 0 0 0 1	CKP, ATN, CR, AUTA		4.4.9
A10AAC	0 0 0 0 0 0 1 0 1	CKP, ATN, CR, AUTA		4.4.10
BR	1 0 W			4.12.1
CALL	1 1 W	CALL		4.12.2
GLA+	0 0 1 0 1 1 1 1 1		AUTA	4.2.3
GLO-	0 0 0 0 1 0 1 1 1	GLO		4.10.4
COMX	0 0 0 0 0 0 0 0 0	COMX		4.11.2
CPAIZ	0 0 1 0 1 1 1 0 1		NATN,CIN,CS,AUTA	4.4.12
DAN	0 0 0 0 0 0 1 1 1	CKP, ATN, CIN, CR, AUTA		4.4.7
DMAN*	0 0 1 0 1 0 1 0 0	MTP, 1STN, CR, AUTA		4.4.6
DYN	0 0 1 0 1 1 0 0 0	YTP, 1STN, CR, AUTY		4.4.8
IM	0 0 0 0 1 1 1 1 0	ATN, CIN, AUTA		4.4.5
INAC+	0 0 1 0 1 0 0 0 0	MTP, CIN, CR, AUTA		4.4.3
IVC	0 0 1 0 1 0 0 1 1	YTP, CIN, CB, AUTY		4.4.6
KNEZ	0 0 0 0 0 1 0 0 1	CKP, NE		4.9.1
LDP	0 0 0 0 1 C	LDP		4.12.4
LDX	0 0 1 1 1 1 1 B	LDX		4.11.1
MNEZ	0 0 1 0 0 1 1 0		MTP, NE	4.6.1
RBIT	0 0 1 1 0 1 1 B	RBIT		4.7.2
RETN	0 0 0 0 0 1 1 1 1	RETN		4.12.3
RSTK	0 0 0 0 0 1 1 0 0	RSTR		4.10.2
SAMAN	0 0 1 0 0 1 1 1 1		MTP, NATN, CIN, CR, AUTA	4.4.2
SEBT	0 0 1 1 0 0 0 B	SEBT		4.7.1
SETR	0 0 0 0 0 1 1 0 1		STO	4.10.1
TAM	0 0 0 0 0 0 0 1 1		STO, YTP, CIN, AUTY	4.3.1
TAMY	0 0 1 0 0 0 0 0 0		STO, AUTA	4.3.2
TAMZA	0 0 0 0 0 0 1 0 0		ATN, AUTY	4.3.3
TAY	0 0 1 0 0 1 0 0 0		CKP, CKN, MTP, NE	4.7.3
TBYT 1	0 0 1 1 1 0 B	TDO		4.6.1
TCY	0 1 0 0 0 C		CKP, AUTY	4.6.2
TMVY	0 1 1 0 0 C		CKM, YTP, CIN, AUTY	4.10.3
TDO	0 0 0 0 1 0 1 0 B		CKP, AUTA	4.9.2
TKA	0 0 0 0 1 0 0 0 0		MTP, AUTA	4.3.5
TMA	0 0 1 0 0 0 0 0 1		MTP, AUTY	4.3.4
TMY	0 0 1 0 0 0 0 1 0		YTP, AUTA	4.2.2
TYA	0 0 1 0 0 0 1 1 1		MTP, STO, AUTA	4.3.6
XMA	0 0 1 0 1 1 1 0		YTP, ATN, NE, STSL	4.6.2
YNEA	0 0 0 0 0 0 1 0 0		YTP, CKN, NE	4.4.3
YNES	0 1 0 1 C			

Aufbau & Funktionsweise

Instruktions-Formate

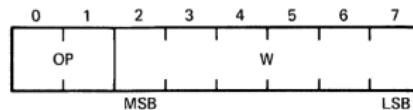
Format 1:



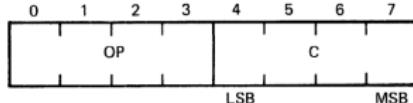
Aufbau & Funktionsweise

Instruktions-Formate

Format 1:



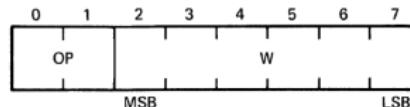
Format 2:



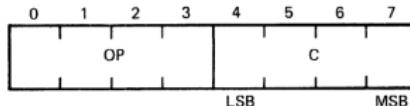
Aufbau & Funktionsweise

Instruktions-Formate

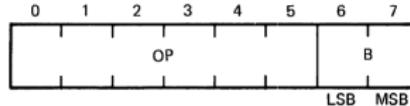
Format 1:



Format 2:



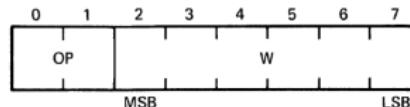
Format 3:



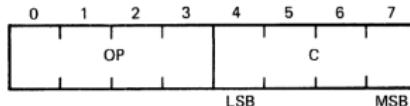
Aufbau & Funktionsweise

Instruktions-Formate

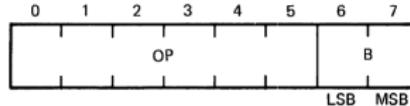
Format 1:



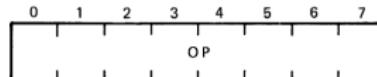
Format 2:



Format 3:



Format 4:



Aufbau & Funktionsweise

Beispiel: AMAAC

- Add Memory to Accumulator, Result to Accumulator

Aufbau & Funktionsweise

Beispiel: AMAAC

- Add Memory to Accumulator, Result to Accumulator
- Format: 4

Aufbau & Funktionsweise

Beispiel: AMAAC

- Add Memory to Accumulator, Result to Accumulator
- Format: 4
- Mikroinstruktionen: MTP, ATN, C8, AUTA

Aufbau & Funktionsweise

Beispiel: AMAAC

- Add Memory to Accumulator, Result to Accumulator
- Format: 4
- Mikroinstruktionen: MTP, ATN, C8, AUTA
- $M(X,Y) + A \text{ in } A$

Aufbau & Funktionsweise

Beispiel: AMAAC

- Add Memory to Accumulator, Result to Accumulator
- Format: 4
- Mikroinstruktionen: MTP, ATN, C8, AUTA
- $M(X,Y) + A$ in A
- Carry = 1 falls Summe größer 15

Aufbau & Funktionsweise

Beispiel: SAMAN

- Subtract Accumulator from Memory, Result to Accumulator

Aufbau & Funktionsweise

Beispiel: SAMAN

- Subtract Accumulator from Memory, Result to Accumulator
- Format: 4

Aufbau & Funktionsweise

Beispiel: SAMAN

- Subtract Accumulator from Memory, Result to Accumulator
- Format: 4
- Mikroinstruktionen: MTP, NATN, CIN, C8, AUTA

Aufbau & Funktionsweise

Beispiel: SAMAN

- Subtract Accumulator from Memory, Result to Accumulator
- Format: 4
- Mikroinstruktionen: MTP, NATN, CIN, C8, AUTA
- $M(X,Y) - A \rightarrow A$

Aufbau & Funktionsweise

Beispiel: SAMAN

- Subtract Accumulator from Memory, Result to Accumulator
- Format: 4
- Mikroinstruktionen: MTP, NATN, CIN, C8, AUTA
- $M(X,Y) - A$ in A
- Carry = 0 falls A größer $M(X,Y)$

Aufbau & Funktionsweise

Beispielprogramm

LABEL	OP CODE	OPERAND	COMMENT
	AMAAC		ADD CURRENT DIGIT TO A
	BR	FIXUP	BRANCH IF CARRY (SUM > 15)
	TAM		TRANSFER A TO MEMORY
	A6AAC		ADD 6, TEST FOR DIGIT 10 TO 15
	BR	CORRECT	BRANCH IF CARRY
	CLA		CLEAR ACCUMULATOR
CONTU			EXIT
	.		
	.		
	.		
FIXUP	A6AAC		ADD 6 TO CORRECT TO BCD
CORRECT	TAMZA		TRANSFER A TO MEMORY, CLEAR A
	IA		INCREMENT ACCUMULATOR
	BR	CONTU	EXIT
	.		
	.		
	.		

Verwendung

- In eingebetteten Systemen

Verwendung

- In eingebetteten Systemen
- In Leistung und Ausstattung meist auf eine Anwendung angepasst

Verwendung

- In eingebetteten Systemen
- In Leistung und Ausstattung meist auf eine Anwendung angepasst
- Beispiele: Unterhaltungselektronik, Waschmaschinenen, Steuergeräte, uvm.

Verwendung

- In eingebetteten Systemen
- In Leistung und Ausstattung meist auf eine Anwendung angepasst
- Beispiele: Unterhaltungselektronik, Waschmaschinenen, Steuergeräte, uvm.
- TMS1000: Taschenrecher, wie z.B SR-16

Verwendung

Der SR-16

