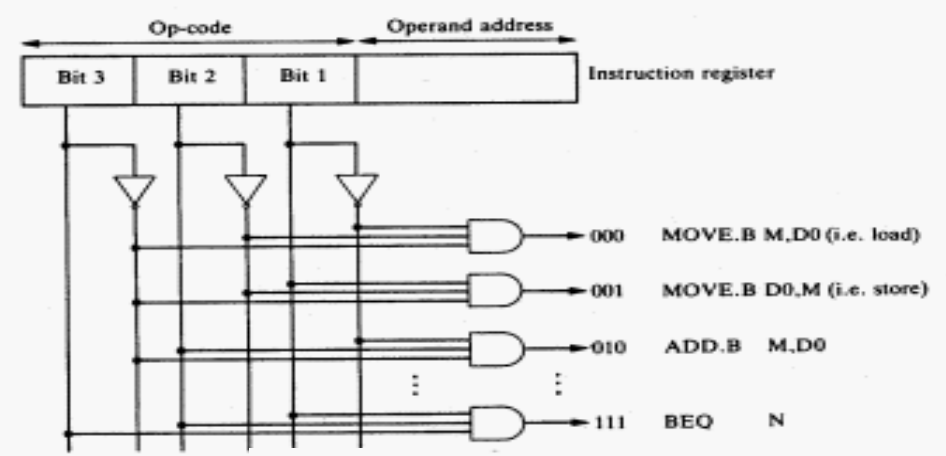
Struktur der CPU (11)

Interpretation des Befehlssatzes:

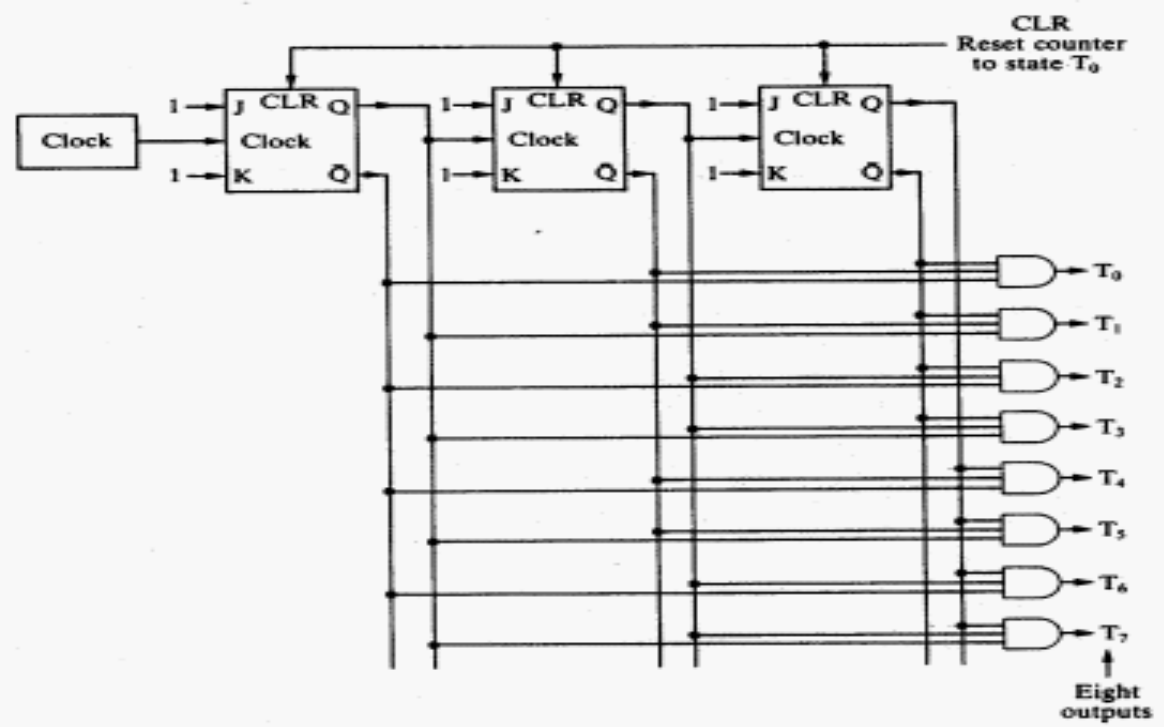
Instruction	Op-code	Operations (RTL)	Control actions
Fetch	—	[MAR]←[PC] [IR]←[MS([MAR])] ALU←[PC] [PC]←ALU	$E_{PC}=1$, C_{MAR} $R=1$, C_{IR} $E_{PC}=1$, $F_1,F_0=1,0$ $E_{ALU}=1$, C_{PC}
LOAD	000	[MAR]←[IR] [D0]←[MS([MAR])]	$E_{IR}=1$, C_{MAR} $R=1$, C_{D0}
STORE	001	[MAR]←[IR] [MS([MAR])]←[D0]	$E_{IR}=1$, C_{MAR} $E_{D0}=1$, $W=1$
ADD	010	[MAR]←[IR] [MBR]←[MS([MAR])] ALU←[MBR] [D0]←ALU	$E_{IR}=1$, C_{MAR} $R=1$, C_{MBR} $E_{MBR}=1$, $F_1,F_0=0,0$ $E_{ALU}=1$, C_{D0}
SUB	011	[MAR]←[IR] [MBR]←[MS([MAR])] ALU←[MBR] [D0]←ALU	$E_{IR}=1$, C_{MAR} $R=1$, C_{MBR} $E_{MBR}=1$, $F_1,F_0=0,1$ $E_{ALU}=1$, C_{D0}
INC	100	[MAR]←[IR] [MBR]←[MS([MAR])] [ALU]←[MBR] [MBR]←ALU [MS([MAR])]←[MBR]	$E_{IR}=1$, C_{MAR} $R=1$, C_{MBR} $E_{MBR}=1$, $F_1,F_0=1,0$ $E_{ALU}=1$, C_{MBR} $E_{MBR}=1$, $W=1$
DEC	101	[MAR]←[IR] [MBR]←[MS([MAR])] ALU←[MBR] [MBR]←ALU [MS([MAR])]←[MBR]	$E_{IR}=1$, C_{MAR} $R=1$, C_{MBR} $E_{MBR}=1$, $F_1,F_0=1,1$ $E_{ALU}=1$, C_{MBR} $E_{MBR}=1$, $W=1$
BRA	110	[PC]←[IR]	$E_{IR}=1$, C_{PC}
BEQ	111	IF Z=1 THEN [PC]←[IR]	IF Z=1 THEN $E_{IR}=1$, C_{PC}

Struktur der CPU (12)

Der Befehlsdekodierer:

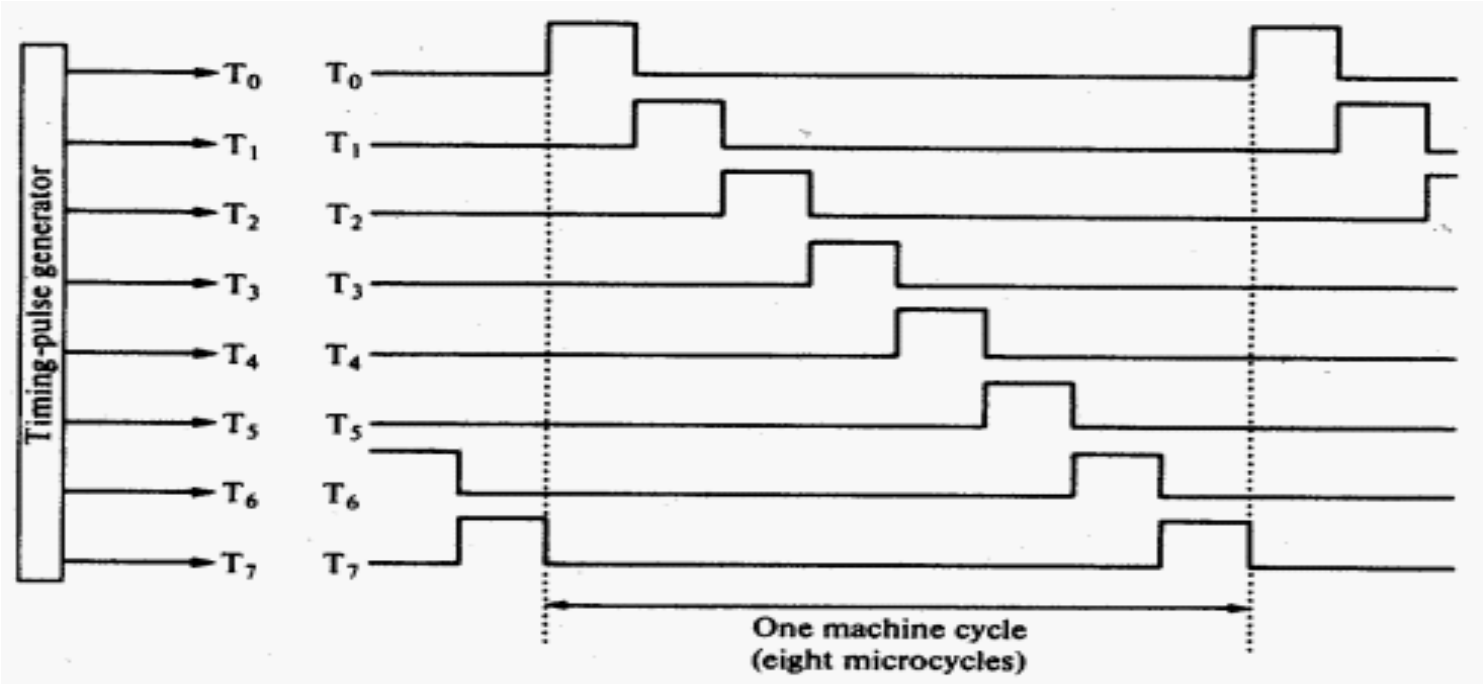


Der Sequenzierer:



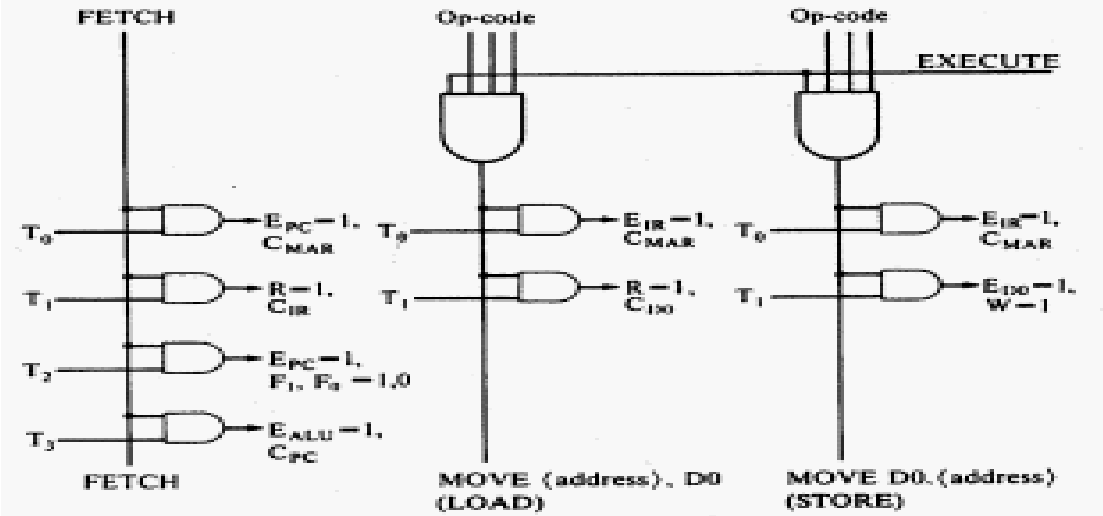
Struktur der CPU (13)

Der Output des Sequenzierers (Impulsgenerator) als Zeitimpulsfolge:

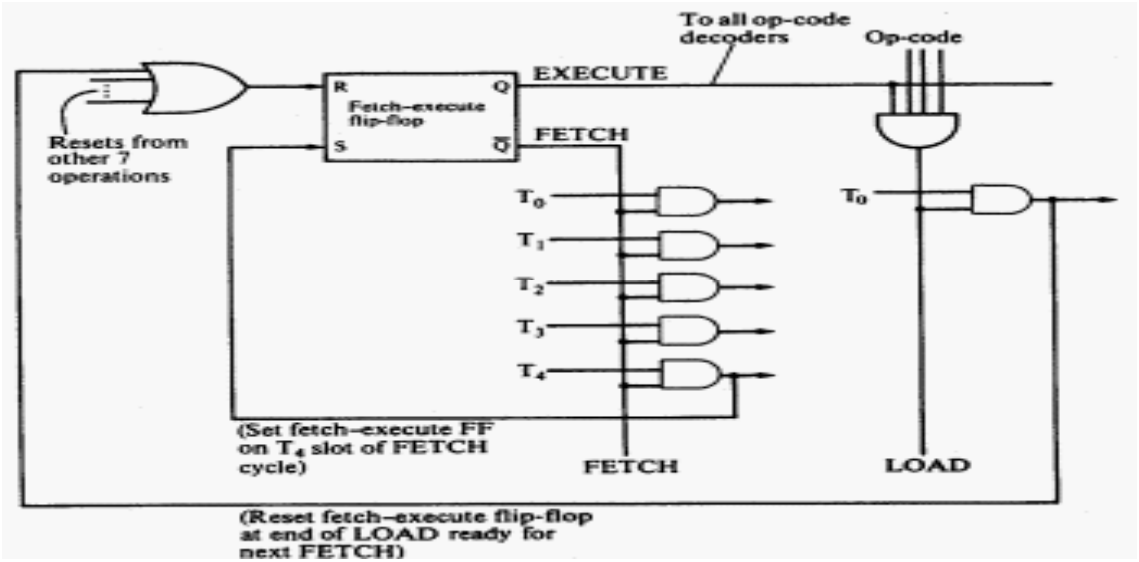


Struktur der CPU (14)

Die verdrahtete (random logic) Kontrolleinheit:



Hinzufügung des „fetch/execute Flip-Flops:



Struktur der CPU (15)

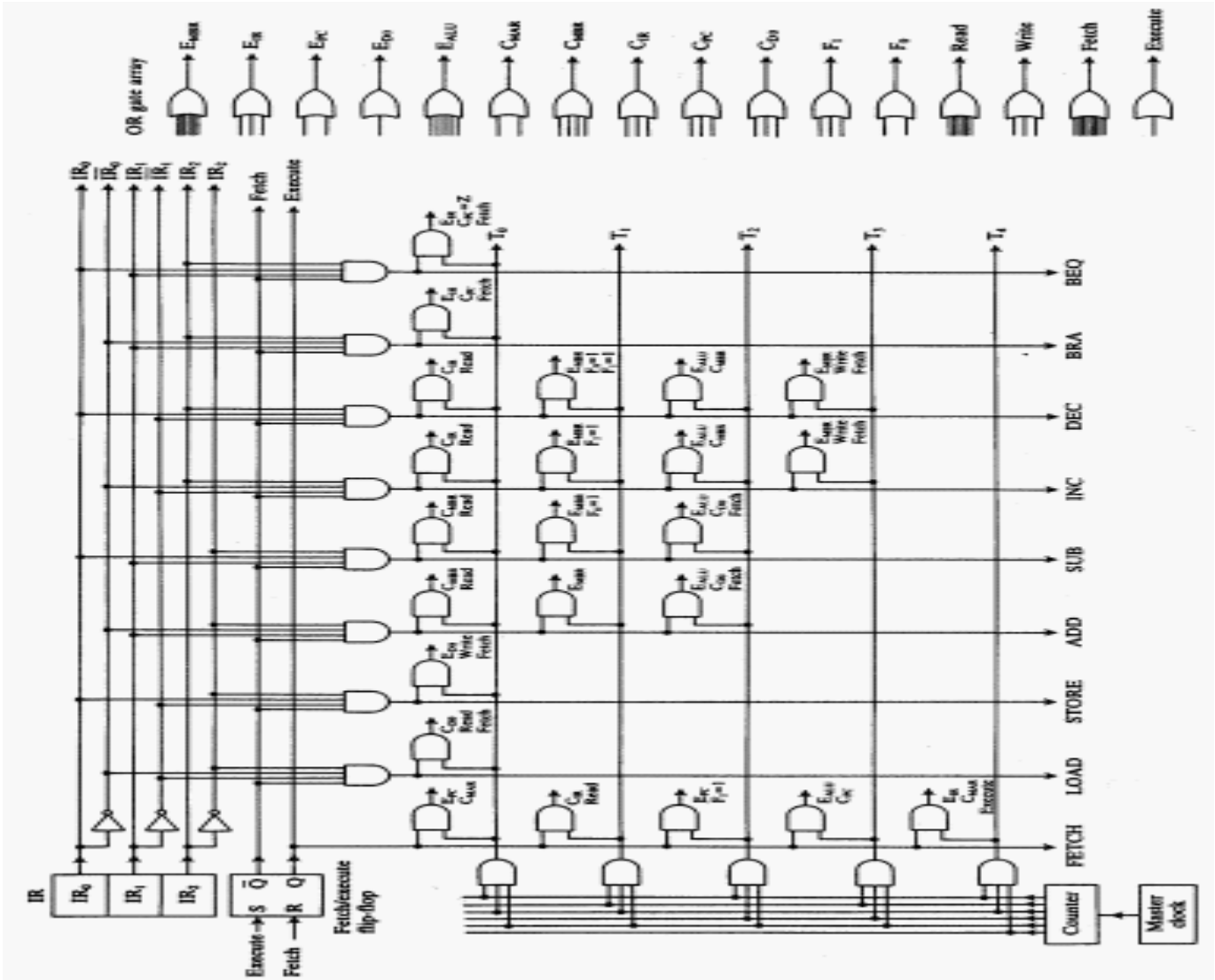
Gesamtdarstellung des beispielhaften Maschinenbefehlsatzes :

Instruction	Time	Enables					Clocks					ALU		MS		FF	
		MBR	IR	PC	DO	ALU	MAR	MBR	IR	PC	DO	F_1	F_0	R	W	R	S
Fetch	T_0	0	0	1	0	0	1	0	0	0	0	X	X	0	0	0	0
	T_1	0	0	0	0	0	0	0	1	0	0	X	X	1	0	0	0
	T_2	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
	T_3	0	0	0	0	1	0	0	0	1	0	X	X	0	0	0	0
	T_4	0	1	0	0	0	1	0	0	0	0	X	X	0	0	0	1
LOAD	T_0	0	0	0	0	0	0	0	0	0	1	X	X	1	0	1	0
STORE	T_0	0	0	0	1	0	0	0	0	0	0	X	X	0	1	1	0
ADD	T_0	0	0	0	0	0	0	1	0	0	0	X	X	1	0	0	0
	T_1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	T_2	0	0	0	0	1	0	0	0	0	1	X	X	0	0	1	0
SUB	T_0	0	0	0	0	0	0	1	0	0	0	X	X	1	0	0	0
	T_1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	T_2	0	0	0	0	1	0	0	0	0	1	X	X	0	0	1	0
INC	T_0	0	0	0	0	0	0	1	0	0	0	X	X	1	0	0	0
	T_1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
	T_2	0	0	0	0	1	0	1	0	0	0	X	X	0	0	0	0
	T_3	1	0	0	0	0	0	0	0	0	0	X	X	0	1	1	0
DEC	T_0	0	0	0	0	0	0	1	0	0	0	X	X	1	0	0	0
	T_1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
	T_2	0	0	0	0	1	0	1	0	0	0	X	X	0	0	0	0
	T_3	1	0	0	0	0	0	0	0	0	0	X	X	0	1	1	0
BRA	T_0	0	1	0	0	0	0	0	0	1	0	X	X	0	0	1	0
BEQ	T_0	0	1	0	0	0	0	0	0	Z	0	X	X	0	0	1	0

Note: Z=zero flag from condition code register.

Struktur der CPU (16)

Gesamtschaltbild der verdrahteten Kontrolleinheit:



Struktur der CPU (17)

verdrahtete gegenüber mikroprogrammierbarer Kontrolleinheit:

• verdrahtet ist schneller als mikroprogrammierbar

- optimiert für einen bestimmte Maschinensprache (Befehlssatz)
- keine Notwendigkeit, Mikrobefehle aus Speicher zu lesen
- Ausführung Boolescher Funktionen ist schneller als Speicherzugriffe

• mikroprogrammiert ermöglicht den flexibleren Entwurf

- leicht modifizierbar
- Behebung von Entwurfsfehlern einfacher
- skalierbar (Hinzufügung neuer und komplexer Maschinenbefehle)

Allgemein gilt, dass die Vorteile der Mikroprogrammierung abhängig sind von

• (hardware-)technologischem Fortschritt
• prinzipieller Sichtweise, die Hauptaufgabe der Rechnerarchitektur betreffend
- Unterstützung des Compilerentwurfs (Top-Down, CISC)
- Performanz und Durchsatz (RISC)

--->Hochzeit der Mikroprogrammierung bis Ende der 80er Jahre