Mem	Opcode 7:4	3:0	Data 7..0 (hex)	Eng	De
RET	0000	0000			
SAT	0000	0001		Store Accu(2:0) to T – Register	die unteren 3 Bits des Akkus werden in das Register „T“ geladen
SIX	0000	0010		Store Accu to RAM/IO (X,Z)	X ist modul addresse (addr = Bit 5..0 aus X-reg)/ Z aus dem Module Addresse 7..0 ?
SST	0000	0011		Store Accu(5:3) -> T (2:0)->S	die unteren 6 Bits des Akkus werden in die Register „S“ und „T“ geladen
LAL I	0000	0100	KK	Load Accu with I	Lade Literal (das folgende Byte) in Akku (2 Byte Befehl)
ANL I / KOL I	0000	0101	KK	AND Accu with I	Konjunktion (UND-Verknüpfung) Akku mit Literal
LIX	0000	0110		Load Accu from RAM/IO (X,Z)	
LIY	0000	0111		Load Accu from RAM/IO (Y,Z)	
LAV	0000	1000		LOAD V to Accu	
LAW	0000	1001		LOAD W to Accu	
LAX	0000	1010		LOAD X to Accu	
LAY	0000	1011		LOAD Y to Accu	
EOL I	0000	1100	KK	Exor	Exklusiv-Oder Akku mit Literal
ORL I	0000	1101	KK	Or	Oder Akku mit Literal
ADL I /ABL I	0000	1110	KK	Add	addiere binär Literal: das folgende Byte „Literal“ wird zum Akku addiert
CML I / VGL I	0000	1111	KK	Compare	Vergleich Akku mit Literal, gesetzt werden nur Flags
	0001	0000			
	0001	0001			
SZX	0001	0010		Store Accu to one Z-reg (X-Reg)	
SZY	0001	0011		Store Accu to one Z-reg (Y-Reg)	
	0001	0100			
	0001	0101			
SQX	0001	0110		Store Accu to one Q-reg (X-Reg)	Xreg (2..0) ->Q(11..8) – Accu -> Q(7..0) ROM-Module X(5..3)
SQY	0001	0111		Store Accu to one Q-reg (Y-Reg)	
SAV	0001	1000		Store Accu to V	
SAW	0001	1001		Store Accu to W	
SAX	0001	1010		Store Accu to X	
SAY	0001	1011		Store Accu to Y	
ALS/VLL	0001	1100			Verschiebe Akku logisch 1 Bit nach links
ARS/VTL	0001	1101			Verschiebe Akku logisch 1 Bit nach rechts
ALF/VLR	0001	1110			Verschiebe Akku um 1 Tetrade (4 Bit) nach links
ARF/VTR	0001	1111			Verschiebe Akku um 1 Tetrade (4 Bit) nach rechts
INP N/LAM	0010	0NNN		Load Input from IO-Addr N to Accu	die unteren 3 Bit des Befehls enthalten die Adresse des I/O-Ports -> Accu
LSS I/LTK	0010	1NNN		Lower 3 Bit NNN to S	die unteren 3 Bit des Befehls werden in das Register „S“ geladen
OUT N/SAM	0011	0NNN		Store Accu to IO-Addr N	Accu -> die unteren 3 Bit des Befehls enthalten die Adresse des I/O-Ports
LTS I	0011	1NNN		Lower 3 Bit NNN to T	
JMP/SPR	0100	0MMM	KK		absoluter Sprung nach M,KK in 2K über 2k
JAZ/SGN	0100	1MMM	KK		springe wenn Akku Null
JAN/SUN	0101	0MMM	KK		springe wenn Akku ungleich Null
JAP/SPB	0101	1MMM	KK		springe wenn Akku positiv
JSD/SSP	0110	0MMM	KK		springe wenn Register „S“ ungleich 7 (Schleifenzähler
JCN/SEU	0110	1MMM	KK		springe wenn Übertrag gesetzt ist
JCZ/SKU	0111	0MMM	KK		springe wenn kein Übertrag gesetzt ist
JSB/SUP	0111	1MMM	KK	PUSH PC->RA->RB->RZ	Sprung in Unterprogramm
LAR R	1000	RRRR		Load Accu from Reg R	Lade Akku mit Registerinhalt, Adressierung wie bei Befehl ABR / **Reg Addr** R
SAR R	1001	RRRR		Store ACcu to Reg R	Speichere Accu to Reg R / **Reg Addr** R
ADR R/ABR	1010	RRRR		ADD Accu and Reg R -> Accu	addiere Registerinhalt zum Akku, **Reg Addr** R: 0..B Register addresse,B ist via T+S, C wie B autodec,E wie B autoinrc
ANR R/KOR	1011	RRRR		AND Accu an Reg R-> Accu	Konjunktion Akku mit Register, Adressierung wie bei ABR / **Reg Addr** R
EOR R	1100	RRRR			Exklusiv-Oder Akku mit Register, Adressierung wie bei ABR / **Reg Addr** R
DER R	1101	RRRR		Decrement Reg R	**Reg Addr** R
DAR R/ADR	1110	RRRR			Dezimaladdition / **Reg Addr** R
LAS I/LAK	1111	KKKK			Lade Akku „kurz“ mit Wert x des Befehlscodes (1 Byte Befehl)

NOTE RRRR = 15 (dez) ist verboten!

Note: NNN -> ADDR = 111NNN ? QRgister die unteren 8 Bit ???
NNN=/ CPU IO

RRRR Register Addr		AddrType	Register Addr	indexing
Bin	Decimal			
0000	0	Direct	0	No
0001	1		1	
	2		2	
	3		3	
	4		4	
	5		5	
	6		6	
	7		7	
	8		8	
	9		9	
1010	10		10	
1011	11		11	
1100	12	Indirect	BY(T,S)	No
1101	13	Indirect	BY(T,S)	S-1 -> S
1110	14	Indirect	BY(T,S)	S+1 -> S
1111	15		forbidden	

Note S<>7 -> Flag

Spezial Register	
V	Gerneral purpose
W	Gerneral purpose
X	indexing modules
Y	= PMC prog module index

Flags:		
	C	Carry Flag
	Z	Zero
	Sbit	1=Sbit<>0
	Sign	Vorzeichen = MSB of Akku

inn CPU		Periph/ROM		
Accu	8-Bit			
S	3-Bit			
T	3-Bit			
		Q	11-Bit	Programmcouneter
		RA	11-Bit	Stack A
		RB	11-Bit	Stack B
		Z	11-Bit	3th Stack/Datapointer
		Periph RAM		
		Z	8-Bit/könnte aber auch 12 Bit sein?	

```
                                Sprung über 2k Grenze



LAL         NEWM        Load new mnodule code into Accu
SAX                     And store it to X.reg
LAL         STADR       Load new start address to accu
SQX                     store it into new q-reg
LAX                     load x-reg to accu
SAY                     now to y.reg = PMC  prog module index
```