

MIL-STD-1750A

Inhaltsverzeichnis

Inhaltsverzeichnis	2
1 Introduction.....	3
2 Definitions	4
3 Data Formats.....	6
3.1 Integer (32 Bit)	6
3.2 Float (32 Bit).....	6
4 Instruction Formats	7
4.1 16 Bit Format : Register to Register Format	7
4.2 32 Bit Format : Long Instruction Format	7
5 Adressing Modes	9
5.1 Register Direct (R).....	9
5.2 Memory Direct (D).....	9
5.3 Memory Direct Indexed (DX)	9
5.4 Memory Indirect(I)	9
5.5 Memory Indirect with Pre-Indexing	9
5.6 Immediate Long (IM)	9
5.7 Immediate Short (IS)	9
5.8 Instruction Counter Relative (ICR)	9
5.9 Base Relativ (B).....	9
6 Registers and Support Features	10
6.1 General Registers.....	10
6.2 Special Registers.....	10
6.2.1 Instruction Counter (IC)	10
6.2.2 Status Word	10
6.2.3 Fault Register.....	11
6.2.4 Interrupt Mask	11
6.2.5 Pending Interrupt Register (PI).....	11
6.3 Optional Register	11
6.4 STACK.....	11
6.5 Prozessor Initialization	12
6.6 Intervall Timers (optional).....	12
7 Memory.....	13
7.1 Memory Addressing	13
7.2 Memory Addressing Arithmetic	13
7.3 Expanded Memory Addressing (optional).....	13
8 Interrupts.....	14
9 Input/Output.....	15
9.1 Input.....	15
9.2 Output	15
9.3 Input/Output Commands	15
9.4 Input/Output Command Partitioning	15
10 Instructions	16
10.1 Invalid Instructions	16
10.2 Mnemonic Convention	16
10.3 Instruction Set Notation.....	16
Instruction Matrix	17

1 Introduction

In den 1970ern zeigte sich das Verteidigungsministerium der USA besorgt über die wachsende Anzahl Prozessoren und Programmiersprachen welche in seinen Projekten verwendet wurden. Die Wartung, Ausbildung, Modularität und Wiederverwendung wurde dadurch schwer beeinträchtigt. Viele Entwicklungen waren zudem proprietär, so dass man sich von Anbietern Abhängig machte.

Aus diesen Gründen wurde am 2. Juli 1980 von der US Air Force der militärische Standard 1750A veröffentlicht. Dabei handelt es sich um die Normvorschrift eines 16 Bit Prozessors, der zusammen mit der validierte Programmiersprache Ada in möglichst vielen Flugzeugen und Waffensystemen eingesetzt werden sollte.

Der militärische Standard legt fest, was ein Prozessor können muss. Dazu zählt der Instruktionssatz, die Register und die Pins. Er steht aber nicht für ein bestimmtes Produkt, dessen Architektur man schützen kann. Die Umsetzung der Vorschrift zu einem Prozessor blieb den Herstellern überlassen.

Jede Ausschreibungen für ein Computersystem für die USAF zwischen 1979 und 1984 basiert auf dieser Normvorschrift. Prozessoren die dem MIL-STD-1750A (B) entsprachen, wurden auch für zivile Projekte wie Raumsonden oder Satelliten eingesetzt (die meisten Satelliten die bis 1993 in Planung gingen).

Anmerkung: Die Beschreibung zum MIL-STD-1750A umfasst komplett mehrere hundert Seiten und kann unter <http://www.xgc.com/manuals/m1750-ada/m1750/book1.html> eingesehen werden. Das vorliegende Dokument bietet einen Auszug aus dieser Beschreibung.

2 Definitions

Accumulator:

Ein Register in der Arithmetischen Logischen Einheit, das zur Zwischenspeicherung von Summen und anderen arithmetischen Ergebnissen und logischen Resultaten dient.

Address:

Eine Nummer, die eine Position im Speicher identifiziert an der sich die gewünschte Information befindet.

Arithmetic Logic-Unit (ALU):

Die ALU gehört zum Rechenwerk und verknüpft die Operanden mit dem Befehl und erzeugt ein Ergebnis.

Bit:

Binärer Wert entweder 0 oder 1. In der Informations-Theorie ist eine Binärziffer gleich einer binären Entscheidung oder die Benennung eines von zwei möglichen Werten oder Zuständen von etwas. Dieser dient zur Vermittlung und Speicherung von Informationen.

Byte:

Eine Gruppe von acht Binärziffern.

Central Processing Unit (CPU):

Der Teil eines Computers, welcher Instruktionen ausführt und kontrolliert.

Control Unit:

Der Teil der Hardware in der CPU, der Operationsfolgen abarbeitet. Er interpretiert und codierte Befehle und leitet die Befehle an andere Teile des Computers weiter.

General Purpose Register:

Ein Register, das für arithmetische und logische Operationen, Indexierung, Shifting, Input-, Output- und zur temporären Speicherung von Daten verwendet werden kann.

Index-Register:

Mit Indexregistern ist es möglich, sich von der statischen Adressierung zu lösen und Speicherzellen dynamisch anzusprechen.

Input / Output (I / O):

Schnittstellen zu der externen Welt.

Instruction:

Programmcode, der dem Computer sagt, was zu tun ist.

Instruction Counter (IC):

Ein Register in der CPU, das die Adresse der als nächstes auszuführenden Anweisung enthält.

Interrupt:

Ein spezielles Signal, das den normalen Operationsfluss des Prozessors unterbricht und es dem Prozessor ermöglicht, auf ein unabhängiges logisches oder unvorhersehbares Ereignis zu reagieren.

Memory:

Der Teil eines Computers, der Daten und Anweisungen speichert und auf die zu einem späteren Zeitpunkt zugegriffen werden kann.

Operation Code (OPCODE):

Ist eine Zahl, die Nummer eines Maschinenbefehls mit dem der Prozessor arbeitet.

Operand:

Der Teil der Instruktion, mit dem der Prozessor arbeitet. Dies kann die Adresse der Quelle, die Adresse des Ziels oder ein Datum sein (Einzahl von Daten).

Programmed Input / Output (PIO):

I/O-Kanal erlaubt dem Programm die Kontrolle des Informationstransfers zwischen dem Computer und einem externen Gerät.

Register:

Ein Gerät in der CPU zur temporären Speicherung von einem oder mehreren Wörtern. Dient zur Unterstützung arithmetischer, logischer oder Transfer Operationen.

Register Transfer Language (RTL):

Eine Sprache zur Beschreibung von Operationen (auf Registern), wird bei der Ausführung jeder Instruktion verwendet.

Reserved:

Reserviert, darf nicht verwendet werden.

Stack:

Eine Folge von Speicherstellen, die auf der Basis eines Last-in-first-out Speichers gespeichert und wieder gelesen werden können.

Stack-Pointer:

Ein Register, das auf den letzten Stackeintrag zeigt.

Status Word Register:

Ein Register, das bestimmte Zustände nach Ausführung eines Befehls angibt.

Word:

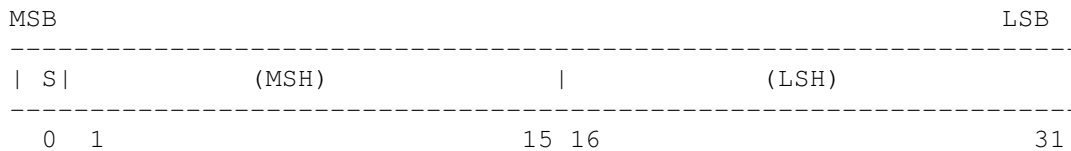
Sechzehn Bits.

3 Data Formats

Der Befehlssatz unterstützt 16 Bit und 32 Bit Ganzzahl-, sowie 32 Bit und 48 Bit Fließkommaberechnungen. Hierzu werden mehrere 16 Bit Register zusammengefasst.

MSB (Most Significant Bit) -> Bit mit höchster Wertigkeit
 LSB (Low Significant Bit) -> Bit mit niedrigster Wertigkeit

3.1 Integer (32 Bit)



Wert	Hexadecimal
2'147'483'647	7FFFFFFF
1	00000001
0	00000000
-1	FFFFFFFF
-2'147'483'647	80000001

3.2 Float (32 Bit)



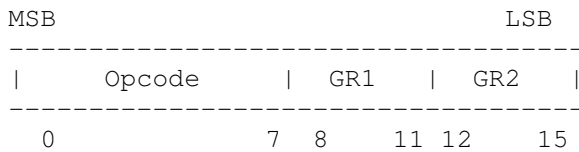
Wert	Hexadecimal	
	Mantisse	Exponent
0.9999998 x2 ¹²⁷	7FFFFFFF	7F
1.0 x2 ⁰		
0.0 x2 ⁰	000000	00
-1.0 x2 ⁰	800000	00
-1.5.x2 ¹²⁷	800000	7F

4 Instruction Formats

Im Instruktionssatz werden 6 Formate mit der Länge 16 oder 32 Bit unterstützt. Der Opcode (auszuführender Befehl →binär) befindet sich in den 8 Bits mit der höchsten Wertigkeit.

4.1 16 Bit Format : Register to Register Format

Die ersten 8 Bits werden für die durchzuführende Operation verwendet. GR1 bezeichnet das erste und GR2 das zweite an der Operation beteiligte general register field. In diesen Registern können Werte für Bit-Verschiebung, Bit-Zahlen, Opcode-Erweiterungen oder Verschiebungszähler hinterlegt werden.



Beispiel: Logische AND Verknüpfung

Addr								
Mode	Mnemonic		Format/Opcode					
			8	4	4			

R	ANDR	RA, RB	E3	RA	RB			

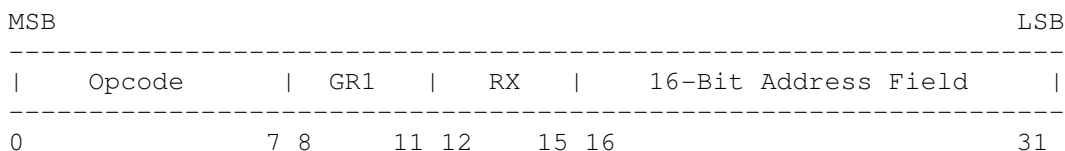
Beschreibung: Die beiden Register RA und RB werden Bitweise AND verknüpft. Das Resultat wird anschließend im Register RA gespeichert.

Berechnung:

```
OP1:      0011100011010101
OP2:      1110011101011100
-----
Resultat: 0010000001010100
```

4.2 32 Bit Format : Long Instruction Format

Ist eine 32 Bit Instruktion die ein 8 Bit Opcode, ein 4 Bit general register field, ein 4 Bit index register field und eine 16 Bit Adresse enthält.



Beispiel: Load Multiple Register

Addr		Mode		Mnemonic		Format/Opcode							
						8	4	4	16				
D	LM	N, ADDR			89		N		RX			ADDR	

Beschreibung: Der Inhalt der Adresse DA wird in das Register R0, der Inhalt von DA+1 in das Register R1bis der Inhalt von DA+N in das Register RN geladen wurde. Dieser Befehl erlaubt den Transfer von (N+1) Worten aus dem Speicher in die Register.

Register Transfer Beschreibung:

(R0) <-- [DA];
 (R1) <-- [DA + 1];
 (R2) <-- [DA + 2];
 (RN) <--[DA + N];

5 Adressing Modes

Nachfolgend werden die zu implementierenden Adressierungsarten aufgelistet. Das kleinste Adressierbare Wort ist 16 Bits lang. Der 16 Bit Adressbus erlaubt eine direkte Adressierung von 64 k (65536) Worten.

5.1 Register Direct (R)

In der Instruktion wird das Register des Operanden angegeben

5.2 Memory Direct (D)

In der Instruktion wird die Adresse des Operanden angegeben.

5.3 Memory Direct Indexed (DX)

Die in der Instruktion angegebene Adresse wird mit dem Inhalt eines Index Registers summiert und ergibt so die Adresse des Operanden.

5.4 Memory Indirect(I)

In der Instruktion wird die Speicherstelle angegeben, an der sich die Adresse des Operanden befindet.

5.5 Memory Indirect with Pre-Indexing

Die in der Instruktion angegebene Adresse wird mit dem Inhalt eines Index Registers summiert und ergibt so die Speicherstelle der Adresse des Operanden.

5.6 Immediate Long (IM)

Der Instruktion wird der Operand direkt übergeben (#daten). Der Operand ist 16 Bit und die Instruktion ist 32 Bit lang.

5.7 Immediate Short (IS)

Der Instruktion wird der Operand direkt übergeben (#daten). Der Operand ist 4 Bit, die Instruktion 16 Bit lang.

5.8 Instruction Counter Relative (ICR)

Diese Adressierungsart wird für 16 Bit Sprunganweisungen verwendet. Der Inhalt des Befehlszählers minus 1 wird zu dem 8 Bit Verschiebungsfeld der Instruktion addiert. Wenn die Sprungbedingung erfüllt ist, erfolgt der Sprung an diese Speicheradresse.

5.9 Base Relativ (B)

Der Inhalt des in der Instruktion angegebenen Basis Registers wird mit dem 8 Bit Verschiebungsfeld addiert (16 Bit Instruktion). Das Verschiebungsfeld muss dabei eine Nummer zwischen 0 und 255 aufweisen.

6 Registers and Support Features

6.1 General Registers

Der Befehlssatz muss ein Minimum von 16 Registern unterstützen (R0 bis R15). Die Register können als Akkumulatoren, Index Register, Basis Registers und temporärer Operanden Speicher und Stackpointer belegt werden. Folgende Restriktionen müssen dabei beachtet werden.

- Die Register R1, R2, ..., R15 können als Index Register verwendet werden.
- Die vier Register R12, R13, R14 und R15 können als Basis Register für Instruktionen mit Base Relativ adress mode verwendet werden.
- R15 muss als Stackpointer für push und pop Instruktionen verwendet werden.
- Die Allgemeinen Register sind nicht im logischen Speicher Adressbereich.
- Das Registerpaar R0, R1 ist der Akkumulator für 32 Bit ganzzahlige und fließkomma Operationen. Das Register R2 dient als Akkumulator für 16 Bit ganzzahlige Operationen.

Die Allgemeinen Register sind 16 Bit lang. Für Instruktionen die einen 32 Bit Betrieb erfordern, werden zwei Register miteinander verkettet. Für Instruktionen die einen 48 Bit Betrieb erfordern werden drei Register miteinander verkettet.

Bei Instruktionen mit verketteten Registern gilt das, in der Instruktion angegebene Register als most significant word (Wort mit höchster Wertigkeit). Die Register können überlaufend verkettet werden. Damit ist es für 32 Bit Operationen möglich R15 und R0 sowie für 48 Bit Operationen R15, R0 und R1 zu verketteten.

6.2 Special Registers

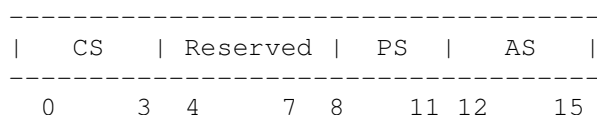
Nachfolgend werden die speziellen Register instruction counter, status word, fault register, interrupt mask, pending interrupt register und input/output interrupt code register erklärt.

6.2.1 Instruction Counter (IC)

Ein 16 Bit Register für die Programm-Sequenzierung (Befehlszähler). Es erlaubt Instruktionen innerhalb eines Bereiches von 65536 Wörtern. Wenn ein Interrupt auftritt, wird das Register im Speicher abgelegt.

6.2.2 Status Word

Ein 16 Bit Register in welchem wichtige, den Computer betreffende Ereignisse festgehalten werden. Dazu gehören Condition Status (CS), Reserved bits, Processor State (PS) und Address State (AS).



6.2.3 Fault Register

Ein 16 Bit Register zur Anzeige des Maschinen Fehlers. Eine logische ODER Verknüpfung der Fault Register Bits wird zur Erzeugung des Machine Error Interrupts verwendet. Über XIO Anweisungen kann das Register gelesen oder gelöscht werden. Bei einem Fehler wird das entsprechende Bit auf 1 gesetzt.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Memory		Parity		I/O		Spa		Illegal			Res		BITE		
Protect						re					rvd				

Beispiel:

Bit 5 → 1 Unerlaubtes I/O Kommando. Es wurde versucht ein noch nicht implementiertes oder unerlaubtes Kommando auszuführen

6.2.4 Interrupt Mask

Ein durch Interrupt Software Control überwachtes Register, das in der Maske für jeden Interrupt ein Bit besitzt. In der Maske wird festgehalten, ob ein Interrupt auftreten darf.

6.2.5 Pending Interrupt Register (PI)

Beim Auftreten eines Interrupts wird in diesem Register das entsprechende Interrupt Bit gesetzt. Durch AND Verknüpfung mit der Interrupt Maske wird festgestellt, ob der Interrupt ausgeführt werden soll.

6.3 Optional Register

Im Standard wurden folgende Register als Optional hinterlegt.

- Input/Output Interrupt Code Registers
- Page Register (Expanded Memory Addressing ist optional)
- Memory Fault Status Register

Bemerkung: Auf eine Beschreibung dieser optionalen Register wird verzichtet.

6.4 STACK

Der Befehlssatz unterstützt einen Stackmechanismus. Der Stapelungsmechanismus muss das Hinzufügen von Einträgen zum Stack als "last-in, first-out" Konzept realisieren. Das Stackregister (Stack Pointer) enthält immer die Speicheradresse des letzten auf dem Stack gespeicherten Eintrags. Zur Verschachtelung von Subroutinen wird das Register 15 verwendet. Der Stack selbst befindet sich in einem benutzerdefinierten Speicherbereich. Das Stackregister wird von den beiden Instruktionen Push Multiple Registers und Pop Multiple Registers verwendet.

6.5 Prozessor Initialization

Prozessorstatus nach einem Reset.

Register/Device/Function	Condition After Reset
Instruction Counter	All zeros
Status Word	All zeros
Fault Register	All zeros
Pending Interrupt Register	All zeros
Interrupt Mask Register	All zeros
General Registers	Indeterminate
Interrupts	Disabled
Timers A & B	Started and all zeros [a]
Page Registers	Group 0 enabled [a]
Page Registers AL Field	All zeros [a]
Page Registers W Field	Zero [a]
Page Registers E Field	Zero [a]
Page Registers PPA Field	Exact logical to physical [a]
Memory Protect RAM	Disabled and all zeros [a][b]
Start Up ROM	Enabled [a]
DMA Enable	Disabled [a]
Input Discretes	Indeterminate [a]
Trigger Go Indicator	Started [a]
Discrete Outputs	All zeros [a]
Bemerkungen: a. Falls implementiert (optional) b. Hauptspeicher protected	

6.6 Intervall Timers (optional)

Bemerkung: Auf eine Beschreibung der optionalen Timer wird verzichtet.

7 Memory

7.1 Memory Addressing

Der Befehlssatz verwendet 16 Bit Adressen um 65536 Worte zu adressieren. Wenn die Option mit dem erweiterten Speicherbereich (nicht implementiert wird, dann entsprechen die logischen Adressen den physikalischen Adressen.

7.2 Memory Addressing Arithmetic

Wird die letzte Speicheradresse 65536 (FFFF) um 1 hochgezählt, so entspricht dies der Speicheradresse 000000.

7.3 Expanded Memory Addressing (optional)

Bemerkung: Auf eine Beschreibung des erweiterten Speicherbereichs wird verzichtet (optional).

8 Interrupts

Der Instruktionssatz soll mindestens 16 Interrupts unterstützen. Eine Interrupt Request kann zu jeder Zeit auftreten. Der Interrupt-Prozess muss so lange warten, bis die Momentan von der CPU verarbeitete Instruktion komplett abgeschlossen ist. Einzige Ausnahme ist die Move Multiple Word Instruktion, die nach jedem Wort unterbrochen werden kann.

Interrupt Number	Interrupt Mask Bit Number	Interrupt Linkage Pointer Address (Hex)	Interrupt Service Pointer Address (Hex)	
0	0	20	21	Power Down (cannot be masked or disabled)
1	1	22	23	Machine Error (cannot be disabled)
2	2	24	25	Spare
3	3	26	27	Floating Point Overflow
4	4	28	29	Fixed Point Overflow
5	5	2A	2B	Executive Call (cannot be masked or disabled)
6	6	2C	2D	Floating Point Underflow
7	7	2E	2F	Timer A (if implemented)
8	8	30	31	Spare
9	9	32	33	Timer B (if implemented)
10	10	34	35	Spare
11	11	36	37	Spare
12	12	38	39	Input/Output Level 1 (if implemented)
13	13	3A	3B	Spare
14	14	3C	3D	Input/Output Level 2 (if implemented)
15	15	3E	3F	Spare

9 Input/Output

Nimmt man den Instruktionssatz, die I/O-Interrupts, und die I/O-Interrupt-Code-Register zusammen, so ergibt sich ein Framework mit dem der Benutzer seine eigene System-Schnittstelle implementieren kann. Alles ausserhalb dieses Frameworks ist kein Teil dieses Standards (z.B. Spezielle Speicherstellen, Page Register Access etc.)

9.1 Input

Die Input Instruktionen transferieren Daten von einem externen I/O Gerät oder einem speziellen internen Register zu einem CPU Allgemeinen Register. Das Kommando Input wird verwendet um Daten von externen Geräten zu lesen (Timer, Statusregister, etc.).

9.2 Output

Die Output Instruktionen transferieren Daten von einem CPU Allgemeinen Register zu einem externen I/O Gerät oder einem speziellen Register. Das Kommando Output wird verwendet um Daten auf ein externes Gerät zu schreiben (DMA ein- und auszuschalten, etc.).

9.3 Input/Output Commands

Input/Output Kommandos sind als verbindlich, optional oder reserviert klassifiziert. Verbindliche I/O Kommandos müssen so wie sie definiert sind auch implementiert werden. Werden optionale I/O Kommandos verwendet, so müssen diese der Definition entsprechend implementiert werden. Reservierte I/O Kommandos dürfen nicht implementiert werden. Fehlerhafte Aufrufe eines nicht implementierten optionalen oder reservierten I/O Kommando löst den illegal I/O command fault aus, womit im Fault Register das Bit 5 auf 1 gesetzt wird und damit der Machine Error Interrupt ausgelöst wird.

9.4 Input/Output Command Partitioning.

Der Input/Output Kommandobereich muss auf 128 Kanäle (I/O channel group) aufgeteilt werden. Mit jeder I/O Schnittstelle müssen bis zu 512 Kommandos ausgeführt werden Können (256 Input und 256 Output).

10 Instructions

10.1 Invalid Instructions

Wird eine Instruktion aufgerufen, die in den ersten 16 Bit keiner Instruktion aus dem Standard entspricht, so muss das Bit 9 des Fault Register gesetzt werden. Dies generiert einen Maschine Error Interrupt.

10.2 Mnemonic Convention

- Jede Instruktion hat eine Mnemonic Konvention. Normalerweise hat die Operation 1 oder 2 Buchstaben (L für Load oder ST für Store).
- Unterschiedlicher Prefix bei der Verwendung unterschiedlicher Datentypen → D, F, EF.
- Register zu Register Operationen haben den Suffix R.
- Direkte Adressierung Immediate Addressing hat den Suffix (IM).

10.3 Instruction Set Notation

Der folgende Auszug zeigt einen Teil der Instruktionsbeschreibung und dessen zugehöriges Symbol.

CPU Registers	
R0, R1, ..., R15	The 16, 16-bit general registers
IC	Instruction Counter
SW	Status Word
CS	Condition Status. A 4-bit quantity that is set according to the result of instruction executions.
LP	Linkage Pointer
SP	Stack Pointer; R15 for the Push and Pop Multiple instructions
SVP	Service Pointer
MK	Interrupt Mask Register
PI	Pending Interrupt Register
RA, RB	An unspecified general register
Addressing Modes	
R	Register Direct
D, DX	Memory Direct, Memory Direct-Indexed
I, IX	Memory Indirect, Memory Indirect with Pre-Indexing
IM, IMX	Immediate Long, Immediate Long with Indexing
ISP, ISN	Immediate Short with Positive Operand, Immediate Short with Negative Operand
ICR	IC-Relative

Instruction Matrix

Die Spaltenüberschrift der Operation Code Matrix weist den Spalten die hexadezimalen Werte von 0 bis F zu. Dieser Wert repräsentiert die vier höherwertigen Bits des Opcodes. Die hexadezimalen Werte am linken Rand repräsentieren die vier niederwertigen Bits des Opcodes.

	Load	Store	Add	Sub	Mult	Divide	Logical	Compare
	8	9	A	B	C	D	E	F
0	L	ST	A	S	MS	DV	OR	C
1	LR	STC	AR	SR	MSR	DVR	ORR	CR
2	LISP	STCI	AISP	SISP	MISP	DISP	AND	CISP
3	LISN	MOV	INCM	DECM	MISN	DISN	ANDR	CISM
4	LI	STI	ABS	NEG	M	D	XOR	CBL
5	LIM		DABS	DNEG	MR	DR	XORR	
6	DL	DST	DA	DS	DM	DD	N	DC
7	DLR	SRM	DAR	DSR	DMR	DDR	NR	DCR
8	DLI	DSTI	FA	FS	FM	FD	FLX	FC
9	LM	STM	FAR	FSR	FMR	FDR	FLT	FCR
A	EFL	EFST	EFA	EFS	EFM	EFD	EFLX	EFC
B	LUB	STUB	EFAR	EFSR	EFMR	EFDR	EFLT	EFCR
C	LLB	SLTB	FABS	FNEG			XBR	
D	LUBI	SUBI					XWR	
E	LLBI	SLBI						
F	POPM	PSHM						NOP

Der Standard weist 98 Befehle auf, die oftmals mit unterschiedlichen Parametern arbeiten.