Technische Universität Chemnitz
Fakultät für Informatik
Professur Technische Informatik
Dr.-Ing. B. Naumann

01.07.2002

LC1 - Beschreibung

Inhaltsverzeichnis

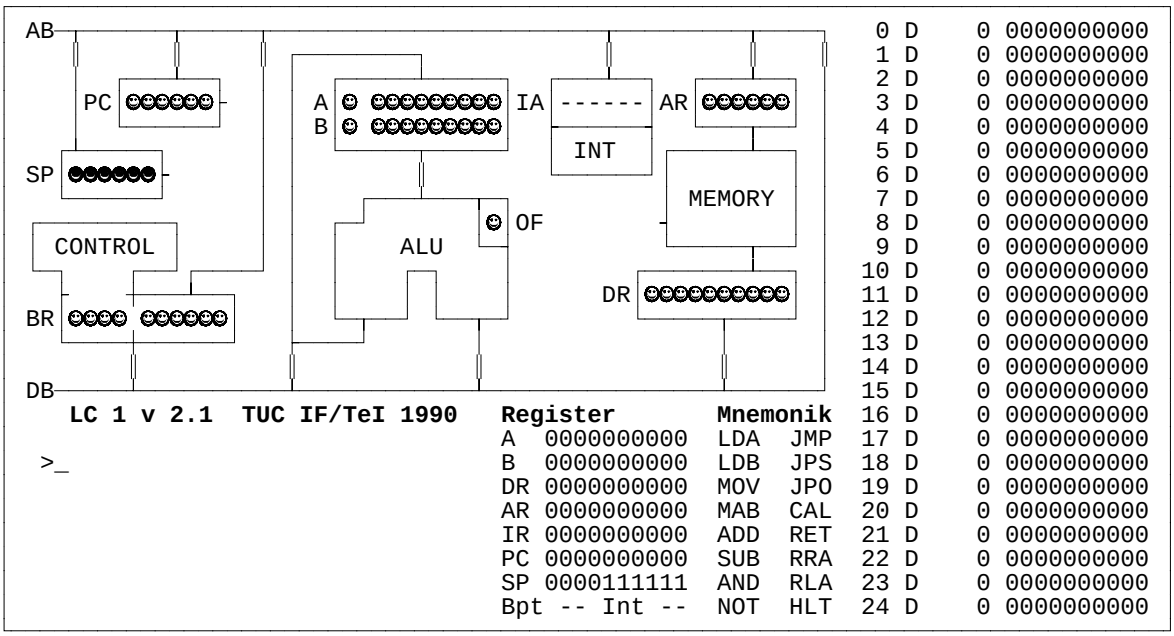
1. Bedienoberfläche 1
2. Architektur des LC1-Prozessors 3
2.1. Arbeitsweise des Steuerwerks 4
2.2. Arbeitsweise des Rechenwerks 5

Anhang - Maschinenbefehlsliste 6

1. Bedienoberfläche

Zum besseren Verständnis der nachfolgenden Ausführungen sollten die LC1-Bedienungsanleitung und die LC1-Befehlsbeschreibung zur Hand genommen werden.

Nach dem Aufruf des Simulators LC1 erscheint das Bild:

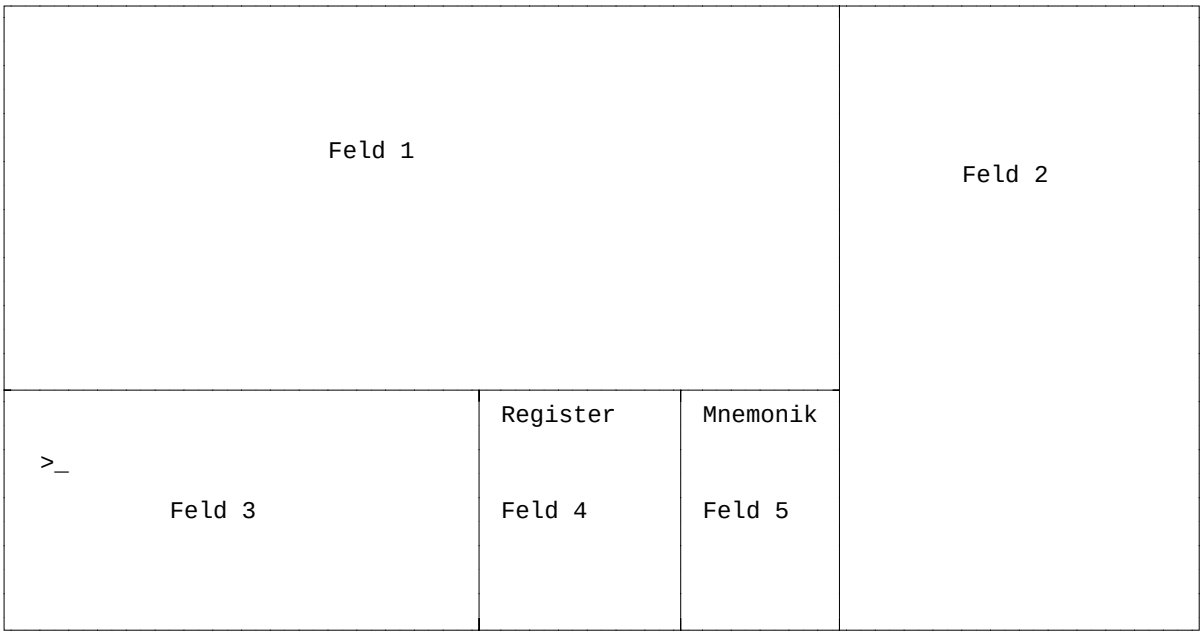


Bedeutung der verwendeten Abkürzungen:

A	register **A** (accumulator)
ADD	**add**
AB	**a**ddress **b**us
ALU	**a**rithmetic **l**ogic **u**nit
AND	logical **AND**
AR	**a**ddress **r**egister

B	register **B**
Bpt	**b**reak**p**oin**t** (dt. Haltepunkt) (-- = nicht gesetzt)
BR	instruction register (dt. **B**efehls**r**egister)
CAL	**cal**l procedure
CONTROL	**control** unit
D	**d**efine
DB	**d**ata **b**us
DR	**d**ata **r**egister
HLT	**H**a**lt** (dt.)
IA	**i**nterrupt **a**ddress register
Int	**i**nterrupt address register (-- = nicht gesetzt)
INT	**int**errupt unit
IR	**i**nstruction **r**egister (dt. Befehlsregister)
JMP	**jump**
JPS	**jump** if **s**ign
JPO	**j**ump if **o**verflow
LDA	**l**oa**d** register **A**
LDB	**l**oa**d** register **B**
MAB	**m**ove register **A** to register **B**
MEMORY	**memory** unit
MOV	**mov**e register A to memory
NOT	logical **NOT**
OF	**o**verflow **f**lag
PC	**p**rogram **c**ounter (instruction pointer)
RET	**ret**urn from procedure
RRA	**r**otate **r**ight register **A**
RLA	**r**otate **l**eft register **A**
SP	**s**tack **p**ointer
SUB	**sub**tract

Das Bild zerfällt in 5 Felder:



Feld 1 stellt die Schaltung des LC1-Prozessors dar. Der aktuelle Inhalt der Register A, B, BR, DR, IA, PC und SP sowie des Flags OF ist binär angegeben:

'☺' = '0'
'☻' = '1'
'-' = nicht gesetzt (nur bei Register IA).

Die Verbindung zwischen Adreßbus AB und Datenbus DB am rechten Rand des Feldes 1 ist eine sog. Bridge (Brücke), die z. B. für die Befehle CAL und RET benötigt wird.

Feld 2 zeigt den aktuellen Inhalt des Speichers in der Form

adr opc dec bin.

adr = absolute Adresse (adr = 0...63).

Nach Eingabe von **v 0** (Default) am LC1-Prompt wird der Inhalt der Speicherzellen 0...24, nach Eingabe von **v 25** der Inhalt der Speicherzellen 25 bis 49 und nach Eingabe von **v 50** der Inhalt der Speicherzellen 50 bis 63 angezeigt.

opc = mnemonischer Operationscode = LDA|LDB|...|HLT|D

Die Operationscodes LDA|LDB|...|HLT bezeichnen den entsprechenden LC1-Befehl und entsprechen damit den linken 4 Bits des Inhalts der Speicherzelle. Der Bezeichner D steht für den Pseudo-Operationscode DEF.

dec = Inhalt der Speicherzelle, dezimal

Bei opc = LDA|LDB|...|HLT wird in dec das Dezimaläquivalent der rechten 6 Bits des Inhalts der Speicherzelle angegeben, für opc = D das Dezimaläquivalent aller 10 Bits des Inhalts der Speicherzelle.

bin = Inhalt der Speicherzelle, binär

Feld 3 enthält nur den LC1-Prompt, hinter dem die LC1-Kommandos einzugeben sind.

Feld 4 zeigt den aktuellen, binären Inhalt der Register A, B, DR, AR, IR (= BR), PC und SP. In der letzten Zeile gibt

Bpt dec Int dec

an, ob ein Break point (= Haltepunkt) oder eine Interruptadresse gesetzt worden sind:

dec = -- , nicht gesetzt
dec = 0...63, auf diese Adresse gesetzt

Feld 5 faßt die Operationscodes der LC1-Befehle als Gedächtnisstütze zusammen (nur von Bedeutung für die Eingabe von LC1-Maschinenbefehlen direkt am LC1-Prompt.

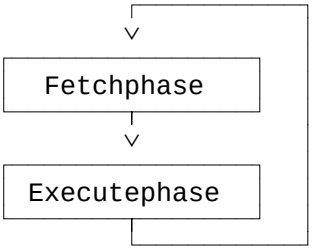
2. Architektur des LC1-Prozessors

Der LC1-Prozessor zerfällt grob in Steuerwerk und Rechenwerk. Zum Steuerwerk zählen wir die eigentliche Steuerung CONTROL und die Register BR, IA, PC und SP, zum Rechenwerk die ALU, die Register A und B und den Speicher MEMORY mit den Registern AR

und DR.

2.1. Arbeitsweise des Steuerwerks

Das Abarbeiten eines Maschinenbefehls erfolgt in zwei Phasen, der Fetchphase und der Executephase. In der Fetchphase wird der Befehl aus dem Speicher ausgelesen, in der Executephase wird er ausgeführt.

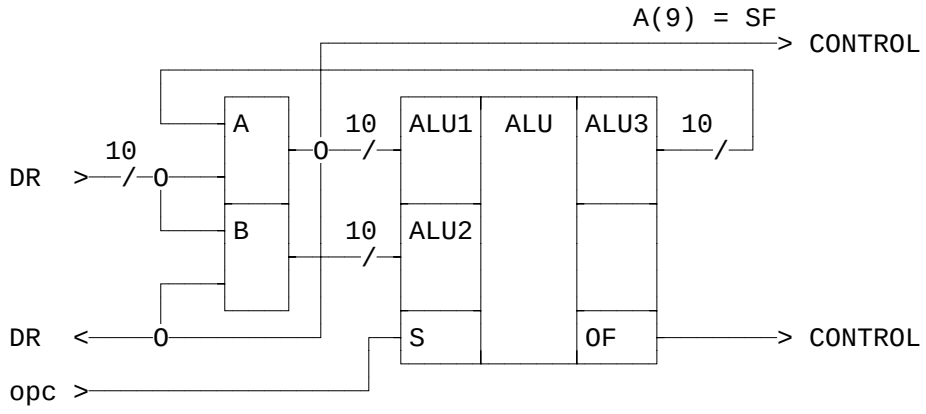


Die Arbeitsweise soll mit einer PASCAL-ähnlichen Notation des Ablaufs erläutert werden. Das Befehlsregister BR ist aufgeteilt in die linken, d. h. ersten 4 Bits, die den Operationscode opc des Maschinenbefehls aufnehmen (opc = $\alpha 4/BR$), und die rechten, d. h. letzten 6 Bits für die Operandenadresse opdadr (opdadr = $\omega 6/BR$).

Nach dem Einschalten (start) werden die Register PC und SP initialisiert. Danach tritt der Prozessor in eine unendliche Schleife aus Fetchphase und Executephase ein. In der Fetchphase wird aus der Speicherzelle, deren Adresse im Register PC steht, ein Befehl ausgelesen und ins Befehlsregister BR geladen. Anschließend wird das Register PC inkrementiert. In der Executephase wird der im Befehlsregister BR stehende Befehl ausgeführt. Abhängig vom aktuellen Operationscode wird eine Folge von "Mikrooperationen" abgearbeitet, die den Maschinenbefehl realisieren.

```
start:  PC := 0;                                  Initialisierung
        SP := 63;
;
loop:   AR := PC;
        DR := MEMORY(AR);                         Fetchphase
        BR := DR;
        PC := PC + 1;
;
        case opc of
           '0000': AR := opdadr;      {LDA}      Executephase
                   DR := MEMORY(AR);             s. Anlage
                   A  := DR;
           '0001': AR := opdadr;      {LDB}
                   DR := MEMORY(AR);
                   B  := DR;
           ......
           '0011': B  := A;           {MAB}
           ......
           '1000': PC := opdadr;      {JMP}
           ......
         end;
;
         goto loop;
```

2.2. Arbeitsweise des Rechenwerks



Im Mittelpunkt steht die ALU mit ihren 10-Bit-Eingangsports ALU1 und ALU2, dem 10-Bit-Ausgangsport ALU3, den Steuereingängen S und dem Flagausgang OF (overflow flag). Die Eingangsports sind mit den Datenausgängen der Register A bzw. B verbunden, das Ausgangsport mit den Dateneingängen des Registers A. Das Rechenwerk weist damit die für Ein-Adreß-Maschinen (Akkumulatormaschinen) typische Struktur auf. Das Register A ist der sog. Akkumulator. Die Funktion der ALU wird über die Steuereingänge S der ALU ausgewählt:

opc	Funktion	
ADD	Addition	ALU3 := ALU1 + ALU2
SUB	Subtraktion	ALU3 := ALU1 - ALU2
AND	Konjunktion	ALU3 := ALU1 & ALU2, bitweise
NOT	Negation	ALU3 := NOT ALU1, bitweise
RRA	Rechtsrotation	ALU3 := RR(ALU1,n)
RLA	Linksrotation	ALU3 := RL(ALU1,n)

Die in der schematischen Darstellung des Rechenwerks eingezeichneten Datenwege sind ungesteuert (A —> ALU1, B —> ALU2) oder abhängig vom Operationscode gesteuert (alle anderen):

opc	aktive Datenwege		
LDA	A := DR	-	-
LDB	-	B := DR	-
MOV	-	-	DR := A
MAB	-	B := A	-
ADD	A := ALU3	-	-
SUB	A := ALU3	-	-
AND	A := ALU3	-	-
NOT	A := ALU3	-	-
RRA	A := ALU3	-	-
RLA	A := ALU3	-	-

Die beiden Flags SF und OF werden auf unterschiedliche Weise gebildet. SF ist identisch mit dem höchstwertigen Bit des Registers A, d. h. SF = A(9). Jeder Maschinenbefehl, der den Inhalt

des Registers A verändern kann, beeinflußt auch das Flag SF (LDA, ADD, SUB, AND, NOT, RRA, RRL). Die Eigenschaft des LC1, daß der Ladebefehl LDA das Flag SF beeinflußt, ist eine Besonderheit des LC1, die bei INTEL-Prozessoren nicht zu finden ist! Sie führt zu sehr kompakten Programmen. OF wird nur durch die Befehle ADD und SUB beeinflußt. Die Tabelle stellt die möglichen Fälle zusammen:

opc	ALU1	ALU2	ALU3	OF
ADD	< 0	< 0	< 0	0
	< 0	< 0	≥ 0	1
	< 0	≥ 0		0
	≥ 0	< 0		0
	≥ 0	≥ 0	< 0	1
	≥ 0	≥ 0	≥ 0	0
SUB	< 0	< 0		0
	< 0	≥ 0	< 0	0
	< 0	≥ 0	≥ 0	1
	≥ 0	< 0	< 0	1
	≥ 0	< 0	≥ 0	0
	≥ 0	≥ 0		0

Anhang

Maschinenbefehlsliste

Format	Notation	Bemerkung
1	adr opc	opc = MAB|ADD|SUB|AND|NOT|RET|HLT
2	adr opc opdadr	opc = LDA|LDB|MOV|JMP|JPS|JPO|CAL
3	adr opc n	opc = RRA|RRL
4	adr opc const	opc = DEF

```
adr     = 0,1,...,63            - Befehlsadresse
opc     = LDA|LDB|...|HLT|DEF   - Operationscode
opdadr  = 0,1,...,63            - Operandenadresse     (opc = LDA|LDB|MOV)
                                - Sprungadresse        (opc = JMP|JPS|JPO)
                                - Unterprogrammadresse (opc = CAL)
n       = 0,1,...,63            - Rotationsweite
const   = -512,-511,...,511     - Konstante
```

Befehl	Mikrooperationen	Funktion	Flags
LDA opdadr	AR := opdadr DR := MEMORY(AR) A := DR	A := MEMORY(opdadr)	SF
LDB opdadr	AR := opdadr DR := MEMORY(AR) B := DR	B := MEMORY(opdadr)	-

Befehl	Mikrooperationen	Funktion	Flags
MOV opdadr	AR := opdadr DR := A MEMORY(AR) := DR	MEMORY(opdadr) := A	-
MAB	B := A	B := A	-
ADD	ALU1 := A ALU2 := B ALU3 := ALU1 + ALU 2 A := ALU3	A := A + B	SF,OF
SUB	ALU1 := A ALU2 := B ALU3 := ALU1 - ALU2 A := ALU3	A := A - B	SF,OF
AND	ALU1 := A ALU2 := B ALU3 := ALU1 & ALU2 A := ALU3	A := A & B, bitweise	SF
NOT	ALU1 := A ALU3 := NOT ALU1 A := ALU3	A := NOT A, bitweise	SF
JMP opdadr	PC := opdadr	PC := opdadr	-
JPS opdadr	if SF = 1 then PC := opdadr	if SF = 1 then PC := opdadr	-
JPO opdadr	if OF = 1 then PC := opdadr	if OF = 1 then PC := opdadr	-
CAL opdadr	DR := PC AR := SP MEMORY(AR) := DR SP := SP − 1 PC := opdadr	MEMORY(SP) := PC SP := SP - 1 PC := opdadr	-
RET	SP := SP + 1 AR := SP DR := MEMORY(AR) PC := DR	SP := SP + 1 PC := MEMORY(SP)	-
RRA n	ALU1 := A ALU3 := RR(ALU1,n) A := ALU3	A(9...0) := A(0),A(9...1) n-mal	SF
RLA n	ALU1 := A ALU3 := RL(ALU1,n) A := ALU3	A(9...0) := A(8...1),A(9) n-mal	SF
HLT	HALT	stoppt die Arbeit des Prozessors	-
DEF const	Definiere Konstante	ordnet die definierte Konstante der durch adr festgelegten Speicherzelle zu	-