Einführung in die Rechnerarchitektur Praktikum

ENTWICKLERDOKUMENTATION

Projekt:

MI: Komplexe Zahlen

Projektleiter	Sandra Grujovic
Dokumentation	Ivan Chimeno
Formaler Vortrag	Martin Zinnecker

1 Einleitung

Dieses Mikroprogramm ermöglicht die Verarbeitung komplexer Zahlen in der mikroprogrammierbaren Maschine. Es beschränkt sich nur auf die Addition und Subtraktion zweier vorzeichenbehafteter Zahlen.

2 Beschreibung des Programms

Das Programm bietet die Möglichkeit, komplexe Zahlen auf eine einfachere Art und Weise zu addieren und zu subtrahieren. Dabei steht eine Akkumulator-Architektur zur Verfügung,welche aus den Registern R6 und R7 besteht. In den Akkumulator kann eine 32-Bit Zahl geladen werden, um darauf Rechenoperationen durchzuführen. Es ist möglich, das Ergebnis wieder in dem Hauptspeicher zu speichern oder in dem Akkumulator zu behalten, um darauf weitere Operationen auszuführen.

3 Tabelle der verfügbaren Befehle

Die folgende Tabelle beschreibt die von dem Mikroprogramm gegebenen Befehle:

Opcode	Länge	Befehl	Beschreibung
01	16-Bit	cLoad [RA]	Lädt eine komplexe Zahl aus dem Speicher in den Akkumulator.
02	16-Bit	cStore [RB]	Kopiert den Wert des Akkumulators in den Speicher.
03	16-Bit	cAdd [RA]	Addiert die komplexe Zahl [RA] zum Akkumulator.
04	32-Bit	cSub [*addr*]	Subtrahiert die komplexe Zahl [*addr*] vom Akkumulator. *addr* ist eine Immediate.

Tabelle 1: Verfügbare Befehle

Tipp: Da alle Befehle Adressen annehmen, bedeutet [X] die Hauptspeicherzelle mit der Adresse X. Das heißt, dass bei [X], also im Register X, die gewünschte Hauptspeicherzelleadresse eingetragen werden muss. **cSub** verwendet als Parameter eine Immediate und verbraucht dementsprechend zwei 16-Bit lange Speicherzellen.

3 Belegung der Register und was zu beachten ist

Die *Tabelle 2* beschreibt die Belegungen der Register vom Mikroprogramm. Die Register R0 bis R7 stehen für den Anwender zur Verfügung. Die Register ab R8 können von dem Entwickler verwendet werden.

Register	Verwendet Von	Kommentar
R6	Akkumulator	Beschreibt den ersten Teil von der komplexen Zahl.
R7	Akkumulator	Beschreibt den zweiten Teil von der komplexen Zahl.
Q	cSub	Wird zum Zwischenspeichern verwendet.

Tabelle 2: Belegte Register vom Programm

- Das Programm arbeitet mit 32-Bit vorzeichenbehafteten Zahlen, d.h. eine Zahl benötigt im Hauptspeicher zwei Speicherzellen, und nicht nur eine! (Siehe Beispiele in der Anwenderdokumentation, ab *Abschnitt 3.3)*
- Die Zahlen sind durch das sogenannte Fixpunktformat dargestellt. Dieses Format wird durchgehend verwendet und sollte von dem Entwickler bzw. Anwender verstanden werden. Für detaillierte Informationen über das Fixpunktformat, siehe Pflichtenheft und Spezifikation.

4 Implementierung der Befehle

In den folgenden Abschnitten werden die Implementierungen der einzelne Befehle genauer beschrieben.

4.1 cLoad [RA]

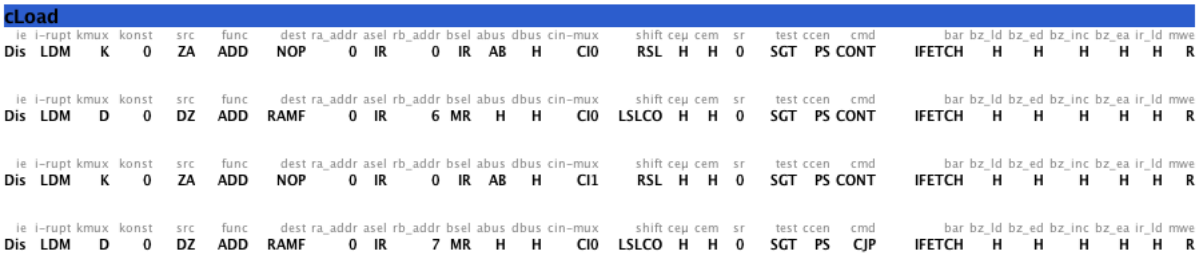

Abbildung 1: cLoad [RA] Implementierung

Folgendes beschreibt die Implementierung für das Laden einer komplexen Zahl aus dem Speicher in den Akkumulator:

1. Den Inhalt des Registers RA auf den Adressbus legen, um den in dieser Adresse liegenden Wert, im nächsten Takt zu erhalten (erster Teil der Zahl).
2. Der Wert liegt jetzt auf dem Datenbus. Diesen in das Register 6 (reeller Teil des Akkumulators) schieben.
3. Die Adresse, die sich im Register RA befindet, anhand des Carry-Flags inkrementieren und auf den Adressbus legen. Der Effekt ist der Gleiche wie in Schritt 1, mit dem Unterschied, dass RA jetzt auf den imaginären Teil b der Zahl zeigt.
4. Der zweite Schritt wird nochmal ausgeführt mit dem Unterschied, dass der Wert in das Register 7 (imaginärer Teil des Akkumulators) geschoben werden muss.

4.2 cStore [RB]

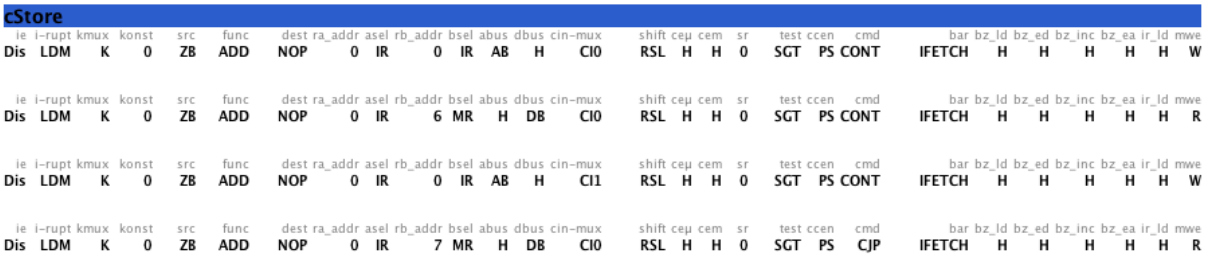

Abbildung 2: cStore [RB] Implementierung

Folgendes beschreibt die Implementierung für das Kopieren des Akkumulators in den Speicher:

1. Den Inhalt des Registers RB auf den Adressbus legen und MWE auf „Write“ setzen, da ein Schreibzugriff an die Zieladresse, die im Register RB enthalten ist, gestartet werden soll. Im nächsten Takt werden die Daten, die im Datenbus liegen, in die durch RB angegebenen Speicheradresse, geschrieben.

2. Den Inhalt des Registers 6 in den Datenbus schieben und MWE wieder auf „Read“ setzen. Im nächsten Takt steht der Inhalt im Speicher.

3. Die Adresse, welche sich im Register RB befindet, anhand des Carry-Flags inkrementieren und auf den Adressbus legen. Der Effekt ist der Gleiche wie in Schritt 1.

4. Schritt 2 wiederholen, mit dem Unterschied, dass nicht Register 6 verwendet wird, sondern Register 7 (zweiter Teil des Akkumulators). Nach diesem Takt wurde der Akkumulator erfolgreich in den Speicher kopiert.

4.3 cAdd [RA]

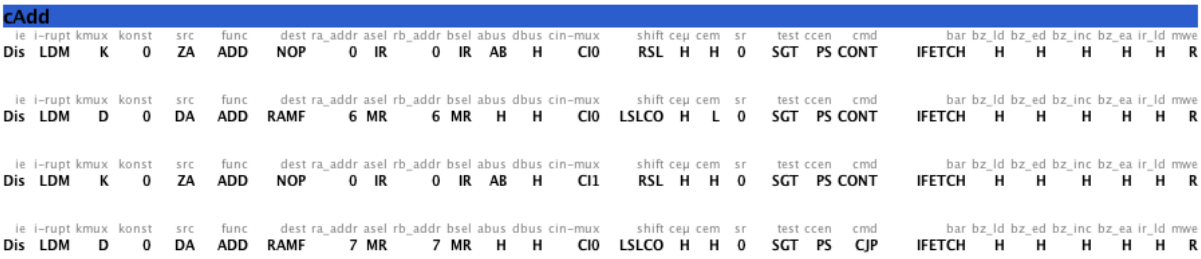

Abbildung 3: cAdd [RA] Implementierung

Folgendes ist die Implementierung für das Addieren einer komplexen Zahl zum Akkumulator, unter Beachtung, dass RA die Adresse ist, die auf den Wert der Zahl zeigt:

1. Den Inhalt des Registers RA auf den Adressbus legen und den lesenden Speicherzugriff starten.

2. Daten, die im Datenbus liegen mit dem Wert, die sich im Register 6 befinden zusammenaddieren und in das Register 6 zurückschreiben. Das Maschinenstatusregister auf „Low“ setzen.

3. Adresse im Register RA anhand des Setzens des Carry-Flags um Eins inkrementieren und auf den Adressbus legen.
4. Daten, die im Datenbus liegen mit dem Wert, der sich im Register 7 befindet zusammenaddieren und in das Register 7 zurückschreiben. Hier wird das Maschinenstatusregister auf „High“ gesetzt!

4.4 cSub [*addr*]

cSub [addr]

ie	i-rupt	kmux	konst	src	func	dest	ra_addr	asel	rb_addr	bsel	abus	dbus	cin-mux	shift	ceμ	cem	sr	test	ccen	cmd	bar	bz_ld	bz_ed	bz_inc	bz_ea	ir_ld	mwe
Dis	LDM	K	0	ZA	ADD	NOP	0	IR	0	IR	H	H	CI0	RSL	H	H	0	SGT	PS	CONT	IFETCH	H	H	H	E	H	R
Dis	LDM	D	0	DZ	ADD	QREG	0	IR	0	IR	AB	H	CI0	RSL	H	H	0	SGT	PS	CONT	IFETCH	H	H	H	H	H	R
Dis	LDM	D	0	DA	SUBR	RAMF	6	MR	6	MR	H	H	CI1	LSLCO	H	L	0	SGT	PS	CONT	IFETCH	H	H	H	H	H	R
Dis	LDM	K	0	ZQ	ADD	NOP	0	IR	0	IR	AB	H	CI1	RSL	H	H	0	SGT	PS	CONT	IFETCH	H	H	H	H	H	R

Abbildung 4: cSub [addr] Implementierung

Folgende Beschreibung stellt die Implementierung für das Subtrahieren einer komplexen Zahl vom Akkumulator dar:

1. Befehlszähler auf den Adressbus laden, um im nächsten Takt die Adresse der komplexen Zahl zu erhalten.
2. Die Adresse liegt auf dem Datenbus. Diese im Q Register speichern, um zu vermeiden, dass die Adresse nicht verloren geht. Gleichzeitig die Adresse auf den Adressbus legen. Im nächsten Takt liegt der erste Teil der Zahl auf dem Datenbus.
3. Die Zahl, die im Register 6 liegt, von der Zahl, die im Datenbus liegt subtrahieren. Das Maschinenstatusregister auf „Low“ setzen und das Ergebnis zurück in das Register 6 schreiben.
4. Die Adresse, die im Q Register zwischengespeichert ist um Eins inkrementieren und auf den Adressbus legen.
5. Der zweite Teil der Zahl liegt jetzt auf dem Datenbus. Register 7 mit diesem Wert subtrahieren und in das gleiche Register zurückschreiben.

5 Ausführung der Befehle

Zum Ausführen der Befehle können Sie im JMic die Befehle und Werte direkt in den Hauptspeicher eingeben. Zu beachten ist die Reihenfolge der Ausführung: JMic startet per Default bei der Adresse *0x0* und arbeitet bis nach unten. Für detaillierte Beispiele, kann man sich an die Anwenderdokumentation wenden.

6 Probleme

Probleme können auftreten, wenn der Wertebereich verlassen wird oder wenn die Belegung der Register des Programms nicht beachtet wird. Durch die Verwendung des

Fixpunktformats, können viele Probleme vermieden werden, wie zum Beispiel, der Überfluss.

7 Programm erweitern

Auf der Basis des implementierten Programms können Erweiterungen durchgeführt werden. Die Befehle bieten beispielsweise eine Grundlage für die Multiplikation, Division und Negation von 32-Bit langen Zahlen. Das Programm kann auch so modifiziert werden, dass sie nur mit vorzeichenlosen Zahlen arbeitet. Wenn der Entwickler mit der MI-Maschine vertraut ist, kann dieser die Befehle natürlich auch optimieren!