LEHRSTUHL FÜR RECHNERTECHNIK UND RECHNERORGANISATION

Rechnerarchitekturpraktikum

Projektaufgabe – Aufgabenbereich Mikroprogrammierung (2.501)

1 Organisatorisches

Auf den folgenden Seiten finden Sie die Aufgabenstellung zu einer Ihrer Projektaufgabe für das Praktikum. Die Rahmenbedingungen für die Bearbeitung werden in der Praktikumsordnung festgesetzt, die Sie auch über die Praktikumshomepage aufrufen können.

Bei Fragen/Unklarheiten in Bezug auf den Ablauf und die Aufgabenstellung wenden Sie sich bitte an Ihren Tutor.

Mit freundlichen Grüßen,
Ihre Übungsleitung

PS: Vergessen Sie nicht, sich rechtzeitig in TUMonline zur Prüfung anzumelden. Dies ist Voraussetzung für eine erfolgreiche Teilnahme am Praktikum im laufenden Semester.

2 Aufgabenumfeld

Viele Probleme der Analysis lassen sich nur im Raum der komplexen Zahlen realisieren. Die komplexen Zahlen erweitern den Zahlenbereich der reellen Zahlen derart, dass die Gleichung $x^2+1=0$ lösbar wird. Dies gelingt durch Einführung einer neuen imaginären Zahl i mit der Eigenschaft $i^2=-1$. Diese Zahl i wird als imaginäre Einheit bezeichnet. In der Technik wird stattdessen der Buchstabe j verwendet, um einer Verwechslung mit einer (durch i oder i(t) bezeichneten) von der Zeit t abhängigen Stromstärke vorzubeugen. [1]

Eine komplexe Zahl z kann durch zwei reelle Zahlen ausgedrückt werden: $a, b \in \mathbb{R}, z = a + i \cdot b$. Da reelle Zahlen nicht ohne weiteres binär dargestellt werden können wird auf die folgende Zahlendarstellung ausgewichen:
Die reellen Zahlen a und b werden jeweils im vozeichenbehafteten (8.8) Fixpunktformat dargestellt, d.h. mit acht Vor- und acht Nachkommastellen. Die gesamte komplexe Zahl nimmt also 32 Bit (2 Speicherzellen) in Anspruch. Tabelle **??** stellt diesen Zusammenhang nochmals dar.

Tabelle 1: Fixpunktformat für komplexe Zahlen

Byte	3		2		1		0
z =	a			+ $i\cdot$	b		
	Vorkomma	.	Nachkomma		Vorkomma	.	Nachkomma

Um die Befehle, die komplexe Zahlen verarbeiten, zu vereinfachen, wird folgende Akkumulator-Architektur festgelegt:

- Die Register r6 und r7 bilden zusammen das komplexe Akkumulationsregister ($accu = r6 + i \cdot r7$)
- Arithmetische Befehle:
  - Der Akkumulator ist immer die erste beteiligte komplexe Zahl
  - Die zweite beteiligte komplexe Zahl kommt immer aus dem Speicher
  - Jede Berechnung speichert das Ergebnis wieder im Akkumulator
  - Das Maschinenstatusregister soll sich auf den Realteil des Ergebnisses beziehen.
- Der Akkumulator kann mit speziellen Transportbefehlen geladen oder kopiert werden

Befehlssyntax:

- $[addr]$: $z = [addr] + i \cdot [addr + 1]$ (addr ist ein Immediate)
- $[RA]$ (oder $[RB]$): $z = [RA] + i \cdot [RA + 1]$

[1] https://de.wikipedia.org/wiki/Komplexe_Zahl

3 Aufgabe

3.1 Arbeitsaufträge

Bitte bearbeiten Sie nur die folgenden Arbeitsaufträge:

- Informieren Sie sich über komplexe Zahlen und die damit verbundenen Rechenregeln.
- Welcher Zahlenbereich lässt sich mit dem vorgestellten Zahlenformat darstellen?
- Welche Vor- und Nachteile ergeben sich aus dem Fixpunktformat im Vergleich zu einer Fließkommazahl?
- Implementieren Sie die folgenden Befehle:
  - **cLoad [RA]**: Lädt eine komplexe Zahl aus den Speicher in den Akkumulator
  - **cStore [RB]**: Kopiert den Akkumulator in den Speicher
  - **cAdd [RA]**: Addiert die komplexe Zahl ([RA]) zum Akkumulator
  - **cSub [addr]**: Subtrahiert die komplexe Zahl ([addr]) vom Akkumulator
- Stellen Sie Test (als Maschinenprogramm in JMic) zur Verfügung, mit denen die richtige Funktionsweise Ihrer Befehle nachvollzogen werden kann.