Rechnerorganisation

Sommersemester 2023

Prof. Stefan Roth, Ph.D.

TECHNISCHE UNIVERSITÄT DARMSTADT

8. Aufgabenblatt

19.6.2023
Mikroarchitekturen

Aufgabe 1: Theoriefragen

a) Was ist der Unterschied zwischen einem Eintakt-Prozessor, einem Mehrtakt-Prozessor und einem Pipelined-Prozessor?

b) Warum benötigt das Registerfeld in dem vorgestellten Prozessor ein Write Enable Signal?

c) Erläutern Sie anhand des Datenpfads wie bei einem ldr/str Befehl die Adresse berechnet wird.

d) Worin unterscheidet sich eine Von-Neumann-Architektur von einer Harvard-Architektur?

Aufgabe 2: Opcodes und Bitfeldbelegung von Maschinenbefehlen

Befehle werden durch sogenannte Opcodes (Operationcodes) definiert. Der ARM®-Prozessor kennt außerdem einige Besonderheiten, wie z. B. die konditionale Befehlsausführung. Bei der Entwicklung des Eintakt-Prozessors wurde bereits die Bitfeldbelegung verschiedener Befehle vorgestellt. Diese Bitfeldbelegung erlaubt neben der Konstruktion des Datenpfads auch ein weiteres vertieftes Verständnis der Befehlskodierung, welches beim Disassemblieren hilfreich ist.

Betrachtet wird das folgende, einfache Assemblerprogramm.

```
/* -- analysis.s */
/* Kommentar  */
.global main /* Einsprungpunkt Hauptprogramm */

main:           /* Hauptprogramm */
    mov r1, #42 /* Schreibe eine 42 in das Register r1 */
    mov r2, #5 /* Schreibe eine 5 in das Register r2 */
    add r0,r1,r2 /* Addiere die Register r1 und r2 */
    bx lr       /* Springe zurueck zum aufrufenden Programm */
```

Assemblieren und Linken Sie das Programm. Schauen Sie sich nun den Object Dump an. Suchen Sie den Additionsbefehl und analysieren Sie die Belegung des Bitfeldes. Nutzen Sie dazu das ARM® Instruction Set[1] Handbuch. Insbesondere das Studium von Abschnitt 4.2 und 4.5 sollte hilfreich sein.

Aufgabe 3: Ägyptisches Multiplizieren

Das Produkt m zweier ganzer, vorzeichenbehafteter Zahlen a und b lässt sich leicht durch das sogenannte ägyptische Multiplikationsverfahren berechnen.

Der Multiplikand wird ständig verdoppelt, der Multiplikator (unter Wegwerfen des Restes) ständig halbiert; aufaddiert werden sogleich oder schlussendlich diejenigen Vielfachen, bei denen in der Multiplikatorhalbierung ein Rest weggeworfen wurde. Bedenken Sie zusätzlich Sonderfälle wie $b < 0$ und $a == 0$.

Beispiel: a = 11, b = 5

11	5
(22)	2
44	1
55	

Die mit „(..)“gekennzeichnete Zahl wird für die Produktbildung nicht aufaddiert.

1. Implementieren Sie den Algorithmus in C. Die Benutzung der Multiplikation und Division (*, / und %) ist dabei nicht zulässig. Benutzen Sie stattdessen die Shiftoperationen.
2. Implementieren Sie den Algorithmus in ARM-Assembler. Dabei sollen 32-Bit Integer, also vorzeichenbehaftete Zahlen in 2K-Darstellung, verwendet werden. Beachten Sie dabei:
   - Die Multiplikationsbefehle sowie die Divisionsbefehle dürfen nicht verwendet werden.
   - Die Werte a und b können Sie fest im `.data`-Bereich unterbringen sowie andere Werte die gegebenenfalls benötigt werden.
   - Kommentieren Sie Ihre Lösung.
3. Testen Sie Ihr Programm mit den Werten

a	b
11	5
-99	99
13	-50
-72	-32
0	56

[1] `https://moodle.tu-darmstadt.de/pluginfile.php/1809291/mod_folder/content/0/Material/arm-instructionset.pdf`