Anforderungen an den Shifter eines ARM-Prozessors

Handout zum 7. Aufgabenblatt

Hardwarepraktikum SoSe 20

Wie Sie bereits in der Übersicht über die ARM-Befehlssatzarchitektur erfahren haben, kennt der ARM-Befehlssatz keine dedizierten Instruktionen für Shifts oder Rotationen eines Operanden. Stattdessen kann der zweite Operand zahlreicher Instruktionen in einem Shifter manipuliert werden, bevor er einer weiteren Verarbeitungseinheit zugeführt wird. In Verbindung mit dem MOV-Opcode der ALU sind damit auch reine Schiebeoperationen auf Operand 2 einer arithmetisch-logischen Instruktion möglich. Der Shifter verwendet und verändert ggf. das Carry-Bit des Prozessors, die anderen CC-Bits werden nicht beeinflusst.

Um die Anforderungen an den Shifter bestimmen zu können, ist eine Analyse der Instruktionen nötig, die den Shifter verwenden. Dies sind:

- alle arithmetisch-logischen Instruktionen
- die Load/Store-Instruktionen `STR(T)`, `LDR(T)`, `STRB(T)` und `LDRB(T)`
- `MSR`
- die Sprungbefehle `B`, `BL`

Für die Sprungbefehle ist die Verwendung des Shifters implizit, der Offset wird immer um zwei Stellen nach links geschoben[1]. In allen anderen Instruktionen wird die gewünschte Schiebe- oder Rotationsoperation explizit im Instruktionswort angegeben. Die größte Bandbreite aller Shift- und Rotationsmöglichkeiten tritt in den arithmetisch-logischen Instruktionen auf, es handelt sich um die in Abschnitt 1.8.3 der ISA-Übersicht vorgestellten Formen von Operand 2. Alle anderen Instruktionen verwenden nur Teilmengen.

[1] Die Verwendung des Shifters für `B` und `BL` ist nicht zwingend notwendig, der statische Shift um zwei Stellen kann im Prozessor im Prinzip auch anders erreicht werden.

Vorbereitenden Vereinfachungen

Um die Komplexität der Shifter-Implementierung für das Praktikum zu begrenzen, sind diverse Vereinfachungen notwendig. Die erste Vereinfachung besteht in der Angleichung des zu schiebenden Datums (Registerinhalt oder Direktoperand). Direktoperanden werden noch im Kontrollpfad des Prozessors auf 32 Bit erweitert und in den Datenpfad eingespeist, sodass aus Sicht des Shifters nur ein Operandenformat existiert - der Eingangsoperand ist immer ein 32-Bit-Vektor und seine Herkunft irrelevant.

Die in Instruktionsworten angegebenen Shift- bzw. Rotationsinformationen lassen sich leider nicht direkt zur Steuerung des Shifters heranziehen, da sie verschiedene Spezialfälle in Abhängigkeit vom Operandenformat codieren (`RRX`-Operation, Shift um 32 Stellen codiert als Schiebeweite 00000). Die zweite Vereinfachung besteht deshalb darin, alle möglichen Operationstypen und -weiten in allen Formaten von Operand 2 auf den allgemeinsten Fall abzubilden. Dies geschieht noch im Kontrollpfad, der Shifter erhält vereinfachte Steuerinformationen ohne Spezialfälle. Sie sind nicht für die Erzeugung dieser verallgemeinerten Steuersignale verantwortlich!

Abbildung der Steuerinformationen auf den allgemeinen Fall

Eine vorgegebene Komponente im Kontrollpfad des Prozessors verringert die Menge der im Shifter zu unterscheidenden Operationen, indem alle auftretenden Fälle auf `LSL`, `LSR`, `ASR` und `ROR` mit einer in einem Byte angegebenen Operationsweite abgebildet werden. Aus Sicht des Shifters existieren also nur noch 4 verschiedene reguläre Operationstypen, die um 0 bis 255 Stellen auf einen 32-Bit-Eingangsoperanden angewendet werden können. Hinzu kommt ein Steuersignal, welches die Anwendung der RRX-Operation verursacht. Damit entspricht der Shifter dem Block aus Abbildung 1.

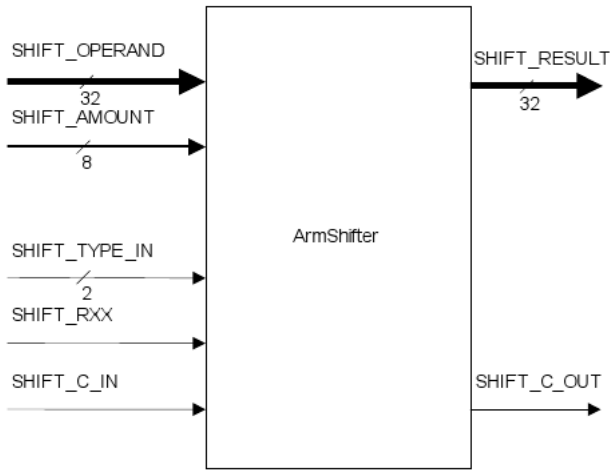

Abbildung 1: Blockschaltbild des ARM-Shifters

Der Shifter nimmt das C-Bit aus dem CPSR entgegen. Bei Shifts und Rotation um 0 Stellen wird das C-Bit unverändert wieder ausgegeben, in allen übrigen Fällen neu erzeugt. Es entspricht dann dem

letzten aus dem Operanden geschobenen bzw. rotierten Bit. Das neue (oder eben alte) C-Bit wird gemeinsam mit dem Shifter-Ergebnis an die ALU weitergeleitet und dort verarbeitet.

Shifter-Struktur im Hardwarepraktikum

Der ARM-Shifter im Praktikum besteht im Kern aus einem Barrelshifter für reguläre Shifts und Rotationen, umgeben von einer Abstraktionsschicht für Spezialfälle. Sämtliche Operationen mit einer Operationsweite von 0 bis 31 Stellen (außer `RRX`) können durch den Barrelshifter realisiert werden. Er erzeugt sowohl das Shifter-Resultat als auch das neue Carry-Bit.

`RRX` und Shifts/Rotationen um 32 bis 255 Stellen sind Sonderfälle, die in der Abstraktionsschicht behandelt werden müssen. Die Möglichkeit, einen 32-Bit-Operanden um mehr als 31 Stellen schieben zu können, erscheint widersinnig. Für die Implementierung einer geeigneten Schaltung ist zu analysieren, welche Konsequenzen aus diesen Operationen erwachsen.

- Logische Shifts um mehr als 32 Stellen erzeugen 0 am Ergebnis- und Carry-Ausgang.
- Der arithmetische Rechtsshift um mehr als 32 Stellen schreibt das MSB des Operanden in alle Stellen des Ergebnisses und den Carry-Ausgang.
- Eine Rotation um n Stellen mit $n > 32$ und $n \neq k \cdot 32, k \in \mathbb{N}$ erzeugt dasselbe Ergebnis und Carry wie eine Rotation um (n modulo 32) Stellen, die stattdessen durchgeführt werden kann.
- Das Ergebnis einer Rotation um $n \cdot 32$ Stellen, $n \neq 0$, entspricht wieder dem ursprünglichen Operanden, das Carry-Bit wird jedoch mit dem MSB des Operanden überschrieben.
- Bei Shifts um genau 32 Stellen entspricht das resultierende Carry-Bit dem MSB (Rechtsshift) bzw. LSB (Linksshift) des ursprünglichen Operanden, der Ergebnisausgang wird wie bei Shifts um mehr als 32 Stellen gebildet.