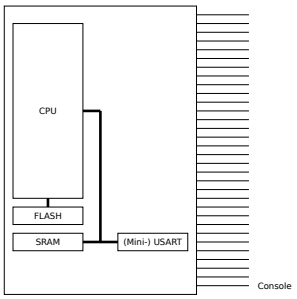
Übungen zu Virtuelle Maschinen, Aufgabe 1

Volkmar Sieh

WS2016/2017

1 Übersicht

In Aufgabe 1 soll eine Virtuelle Maschine entwickelt werden, welche eine begrenzte Untermenge der Befehle der AVR-Architektur ausführen kann. Neben dem CPU-Simulator sollen ein zusätzlicher Simulator für einen Seriellen Controller, sowie eine RAM- und einen FLASH-Komponente implementiert werden.



Die Aufgabe gilt als erfüllt, wenn die Test-Programme „simple“, „calc“ und „jump“ (siehe unten) korrekt ausgeführt werden.

2 Organisatorisches

Bitte senden Sie Ihre Lösung bis zum 29.11.2016 per Mail an i4vm@cs.fau.de. Die Projektdateien finden Sie unter
https://www4.cs.fau.de/Lehre/WS16/V_VM/Uebungen/.

3 Hardware-Simulatoren

Implementieren Sie Hardware-Simulatoren, die folgenden Modellbeschreibungen genügen. Der Einfachheit halber, soll sämtliche virtuelle Hardware direkt miteinander über Unterprogrammaufrufe verbunden werden.

3.1 SRAM

Die Virtuelle Maschine soll über eine Speicherkomponente verfügen. Das SRAM soll eine Größe von 2 KByte haben. Beim VM-Start sind die Bytes mit „0“ zu initialisieren.

3.1.1 FLASH

Der 32 KiB RAM-Baustein soll ab Adresse 0x000 angesteuert werden können. Initialisieren Sie den Inhalt des FLASH-Bausteins mit dem Inhalt eines Test-Programm-Images.

3.1.2 (Mini-) USART

Schreibt die CPU auf I/O-Adresse UDR (0x0c), soll der 8-Bit-Wert als ASCII-Zeichen auf die Console, in der die VM gestartet wurde, geschrieben werden.

3.2 CPU Simulator

Der CPU-Simulator soll – zumindest in kleinen Teilen – die *AVR*-ISA simulieren können. Zunächst sollen keine Exceptions oder Interrupts simuliert werden.

Da der Befehlssatz der *AVR*-Architektur sehr umfangreich ist, genügt es, nur die von den Testprogrammen benötigten Befehle zu unterstützen.

Beim Start der Emulation soll die CPU bei Adresse 0x000 beginnen, Instruktionen zu holen und auszuführen.

In den Projektdateien finden sich einige Test-Programme, mit denen Sie Ihren CPU-Simulator testen können. Wenn man die Test-Programme mit anderen gcc-Optimierungseinstellungen compiliert, ergeben sich ggf. weitere Test-Möglichkeiten.

4 Projektdateien

Im Folgenden finden Sie eine Kurzübersicht über die Projektdateien. Sämtliche Dateien dürfen Sie nach Belieben (sinnvoll) verändern.

`Makefile`: Makefile zum Bauen der VM und der Test-Programme

`docs`: AVR / ATmega32 Dokumentation (Kopien von `http://www.atmel.com/images/Atmel-0856-AVR-Instruction-Set-Manual.pdf` und `http://www.atmel.com/images/doc2503.pdf`)

`src`: Emulator-Sourcen

- `Makefile`: Makefile zum Bauen der VM
- `main.*`: Erzeugt/startet/beendet die VM
- `gui.*`: Grafische Benutzeroberfläche (gtk)
- `sig_std_logic.*`: Implementierung der Kabel
- `chip_atmel_atmega32.*`: ATmega32-Chip (Stub)
- `seg7.*`, `button.*`, **...**: Implementierung der I/O-Geräte

`test`: Test-Programm-Sourcen

- `Makefile`: Makefile zum Bauen aller Test-Programme
- `libspicboard`: Bibliothek mit einfachen Ein-/Ausgabefunktionen
- `simple`: „Hallo Welt"-Variante
- `calc`: Berechnet die Summe und die Fakultät verschiedener Werte und gibt diese auf der Console aus.
- `jump`: Testet viele bedingte Sprünge
- `boardtest`: Programm testet alle SPiC-Board-I/O-Komponenten
- **...**: ...