Hochschule RheinMain
Fachbereich Design Informatik Medien
Marcus Thoss, M.Sc.

Mikroprozessortechnik
SS 2020
LV 2522

Übungsblatt 4

Aufgabe 4.1 (Serielle Ausgabe auf QEMU/ARM):

Die Emulation des TI LM3S6965 mit `qemu` enthält u.a. auch eine virtuelle serielle UART-Schnittstelle. Hiermit kann im einfachsten Fall zeichenweise Ein- und Ausgabe auf der Konsole des Emulators erfolgen.

Für den folgenden Code darf bei den Startoptionen von `qemu` anders als für Blatt 2 die Option `--serial null` *nicht* angegeben werden (sonst wird die serielle UAR-Schnittstelle deaktiviert). Dadurch startet `qemu` aber auch nicht mehr mit seiner eigenen Eingabeaufforderung für Monitor-Kommandos, sondern zeigt direkt die Ausgabe der UART an.

Um zwischen diesen Modi hin- und herzuschalten oder `qemu` zu verlassen, gibt es in `qemu` besondere Tastenkombinationen, die Sie sich mit `Ctrl-a h` (bzw. `Strg-a h`) anzeigen lassen können.

a) Erweitern Sie Ihr ARM-Testprogramm um eine Ausgabe eines einzelnen 'A'. Das UART-Datenregister liegt an der Adresse `0x4000c000`. Durch Schreiben eines Wortes in dieses Register wird auch die Ausgabe angestoßen.

   Schreiben Sie in Ihrem Assemblerprogramm ein Datenwort, das in den unteren 8 Bit den ASCII-Wert für 'A' enthält, in dieses Register, und beobachten Sie beim Test die Ausgabe auf der Konsole.

b) Programmieren Sie eine Schleife, in der zehnmal der Buchstabe 'A' ausgegeben wird.

c) Geben Sie "Hello, World!" aus: definieren Sie dazu mit der `.string`-Direktive den Ausgabestring und implementieren Sie eine Subroutine, die die angegebene Anzahl ASCII-Bytes ab einer angegeben Adresse in einer Schleife ausgibt. Bedenken Sie, dass die Standard-Wortbreite bei ARM 32 Bit beträgt! Sehen Sie sich für eine alternative Lösung auch `.string32` an.

Aufgabe 4.2 (Test 1):

Die Ergebnisse zu den folgenden Testaufgaben müssen für die Berücksichtigung der Leistung als PDF-Dokument (mit darin eingebettetem Quellcode) mit dem Dateinamen `<Nachname>_<Vorname>_mpts20_t01.pdf` am 25.05.2020 bis zum *Tagesende* per Email an `marcus.thoss@hs-rm.de` gesendet werden. Sofern nicht anders angegeben, wird für eine vollständig gelöste Teilaufgabe ein Punkt vergeben.

a) Eine in AVR-Assembler realisierte und an der Adresse `myfunc` beginnende Subroutine wird mit den Werten `0x13` in Register `r24` und `0x2e` in Register `r22` aufgerufen und soll mit diesen Werten eine Rechenoperation durchführen, die `0x3d` ergibt. Das Ergebnis soll in Register `r24` zurückgegeben werden. Geben Sie AVR-Assemblercode für die Subroutine (beginnend mit dem Sprunglabel) an. Geben Sie auch Beispielcode für einen Aufruf von `myfunc` mit vorherigem Laden der Parameter in die Register an. (**2P**)

b) Geben Sie analog zur vorigen Frage **ARM**-Assemblercode für die Subroutine `myfunc` und ihren Aufruf mit Laden der Parameter an. Hier sollen die Parameter `0x13` in `r0` und `0x2e` in `r1` übergeben werden und die Rückgabe in `r0` erfolgen. (**2P**)

c) Bringen Sie die folgenden Stufen einer Befehlspipeline in die richtige Reihenfolge:

(a) Operand Fetch (b) Write Back (c) Execute (d) Instruction Fetch (e) Instruction Decode

- für LOAD-/STORE-Archtitekturen
- für Architekturen ohne LOAD-/STORE-Beschränkung

(2 Antworten) (**2P**)

d) Erläutern Sie: Was ist in der AVR-Architektur der Unterschied zwischen dem S- und dem N-Flag?

e) Was passiert, wenn während der Programmausführung das PC-Register mit der Adresse der Subroutine `myfunc` geladen wird?

f) Welchen Wert haben auf einem ARM Cortex-M3-Prozessor die Bits `N`,`Z`,`C` und `V` nach der Ausführung der Instruktion `ADDS r0,r0,r1` für die Summanden `0x80000000` in `r0` und `0x80000000` in `r1`? Welcher Wert liegt als Ergebnis in `r0`? Interpretieren und ***erläutern*** Sie anhand der Flags, was bei der Rechnung passiert ist. (**2P**)