Institut für Informatik
und Praktische Mathematik
der Universität Kiel

Kiel, 05.06.2002

Übungen zu Organisation und Architektur von Rechnern
SS 2002
Serie 6

Aufgabe 16 (4 Punkte)

Der Simulator `bsvc` verfügt über einen seriellen I/O-Baustein (M68681 Dual UART Device) der dazu verwendet werden kann, Zeichenketten in einem Terminalfenster auszugeben. Die Setup-Datei `serial.setup` sorgt dafür, daß über die Adresse `$effc01` auf einen solchen DUART-Baustein zugegriffen werden kann und ein Terminalfenster geöffnet wird, in dem die an den DUART übermittelten Daten ausgegeben werden.

Bevor der DUART verwendet werden kann, muß er erst einmal geeignet initialisiert werden. Schreiben Sie für diesen Zweck eine Funktion `INIT_IO`, die gemäß der unter
`http://www.informatik.uni-kiel.de/~sim68k/bsvc/doc/Manual/html/chap3.htm`
abrufbaren Dokumentation die folgenden Schritte (in genau der angegebenen Reihenfolge!) durchführt:

1) Einen Reset auf den MR-Pointer durchführen (`%00010000` nach CRA schreiben).
2) 8-Bit Modus wählen (MR1A).
3) Normal-Modus wählen (MR2A).
4) 9600 Baud wählen (CSRA).
5) Transmitter aktivieren (CRA).

Hinweis: Alle Register des DUART sind 8 Bit lang, liegen jedoch auf Wortgrenzen, d.h. MR1A und MR2A liegen auf `$effc01`, CSRA liegt auf `$effc01+2`, CRA liegt auf `$effc01+4`, etc.

Aufgabe 17 (3 Punkte)

Schreiben Sie eine Funktion `PUT_CHAR`, die ein einzelnes Zeichen (ein Byte) an einen bereits initialisierten DUART-Baustein sendet, indem es das Byte-Datum in den Puffer TBA schreibt. Zuvor muß allerdings durch eine Überprüfung des TxRDY-Bit im SRA sichergestellt werden, daß der UART-Baustein empfangsbereit ist. Gegebenenfalls muß mit dem Senden solange gewartet werden, bis dieses Bit auf 1 steht.

Hinweis: Die Ausgabe des gesendeten Zeichens in dem Terminalfenster erfolgt im ASCII-Code. Wird zum Beispiel der Wert `$21` gesendet, wird das Zeichen `'!'` ausgegeben. Siehe auch:
`http://www.asciitable.com` .

Aufgabe 18 (3 Punkte)

Schreiben Sie eine Funktion `PUT_STR`, die eine Zeichenkette (eine Sequenz von Bytes) an einen bereits initialisierten DUART-Baustein sendet. Die Funktion `PUT_STR` erhält als einzigen Parameter die Startadresse der Zeichenkette (d. h. die Adresse des ersten Bytes). Das Ende der Zeichenkette muß durch den speziellen Wert `$00` gekennzeichnet sein.

Beispiel: Die Sequenz `$48 $61 $6c $6c $6f $00` repräsentiert die Zeichenkette "`Hallo`".

Aufgabe 19 (10 Punkte)

Schreiben Sie eine Funktion `PUT_HEX`, die eine Hex-Zahl (unsigned long) auf einem bereits initialisiertem DUART-Baustein ausgibt. Definieren Sie dazu eine Hilfsfunktion `HTOA`, die eine Hex-Zahl (unsigned long) in eine 0-terminierte Zeichenkette (siehe `PUT_STR`, Aufgabe 16) konvertiert. Dokumentieren Sie ausführlich(!), wie Sie den für die Zeichenkette benötigten Speicherbereich verwalten.

Hinweis: Mit Hilfe der Assembler-Anweisung `DS` läßt sich ein Speicherbereich fester Größe reservieren.

Kommentieren Sie Ihre Assemblerprogramme ausführlich. Geben Sie insbesondere im Kopf jeder Funktion an, welche Argumente und Rückgabewerte wie und wo erwartet bzw. geliefert werden. Dokumentieren Sie auch alle evtl. vorhandenen Seiteneffekte der Funktionen wie z.B. Ausgaben oder Veränderungen von Speicherbereichen außerhalb des Stacks. Etablieren Sie bei allen Funktionen die für Aufgabe 14 vorgegebene Aufrufkonvention.

Abgabe: Di., 18.06.2002 in den Übungen.