



FACHHOCHSCHULE DARMSTADT

FACHBEREICH INFORMATIK

MIKROPROZESSORPRAKTIKUM 1

WS2005/06

Termin 3

Load/Store, Stacks, Branch, Conditional Befehle

Arbeitsverzeichnis:

Sie arbeiten in dem Verzeichnis `/home/milabuser/mi1/termin3`. Dort stehen die Datei *Rahmen_Liste.S* als Programmgerüst und eine einfache Makedatei *makefile* zur Verfügung. Kopieren Sie die Datei *Rahmen_Liste.S*

```
cp Rahmen_Liste.S liste.S
```

Mit dem Aufruf *make* wird ein ausführbares Programm *liste.elf* erzeugt.

Lernziele:

Load/Store-Architektur, Stack-Handling, Branch-Befehl, Conditional-Befehle.

Aufgabenstellung:

Es ist ein Programm (*kopieren*) zu entwickeln, das eine Liste TAB1 in eine Liste TAB2 kopiert.

Nach dem Kopiervorgang sollen in einem weiteren Programm (*vergleichen*) die beiden Listen verglichen werden. Ist ein Wert der beiden Tabellen ungleich, so soll in eine Routine (*korrigiere*) verzweigt werden, die den Fehler in der Liste TAB2 korrigiert, danach wird mit dem Vergleich weitergemacht.

Zusätzlich soll ein Zähler die Anzahl der ungleichen Speicherstellen mitzählen und an das aufrufende Programm zurückgeben.

// Rahmen für die Aufgabe zu Termin3

// Name:_____ Matrikel-Nr.:_____

// Name:_____ Matrikel-Nr.:_____

```

        .file      "liste.S"
        .text
        .align     2          @ legt eine Textsection für ProgrammCode + Konstanten an
                                @ sorgt dafür, dass nachfolgende Anweisungen auf einer durch 4
                                @ teilbaren Adresse liegen unteren 2 Bit sind 0
        .global    main      @ nimmt das Symbol main in die globale Sysmboltabelle auf
        .type      main,function

main:
        stmfd      sp!, {lr}   @ Rücksprungadresse sichern und sp (r13) um 4 verkleinern
// ..      Hier bitte Ihren ProgrammCode einfügen

        bl        kopieren    @ Aufruf der Kopierroutine
// ..
        bl        vergleichen  @ Aufruf der Vergleichsroutine
// ..
        ldmdfd     sp!, {pc}    @ zurück zum aufrufenden System und sp (r13) um 4 erhöhen

// In Register r0 und Register r1 sollten die Zeiger auf die beiden Listen übergeben werden
kopieren:
        stmfd      sp!, {lr}
// ..
        ldmdfd     sp!, {pc}

// In Register r0 und Register r1 sollten die Zeiger auf die beiden Listen übergeben werden
// In Register r0 wird der returnwert zurückgegeben z.B.Fehlerzahl
vergleichen:
        stmfd      sp!, {r4, r5, lr} @ Rücksprungadresse und Register (nicht Scratch) sichern
                                @ zu rettende Register müssen noch angepaßt werden
                                @ es handelt sich hier nur um ein Beispiel
// ..
        bl        korrigieren
// ..
        ldmdfd     sp!, {r4, r5, pc}
// In Register r0 und Register r1 sollten die Zeiger auf die beiden zu korrigierenden Werte übergeben werden
// kein Returnwert
korrigieren:
        stmfd      sp!, {lr}
// ..
        ldmdfd     sp!, {pc}

// Tabellen
// erster Wert der Tabellen steht für die Länge der Tabelle
TAB1:    .word      ((TAB1ende-TAB1)/4), 1, 2, 3, 4, 5, 6, 7, 8, 9, 0
TAB1ende:

TAB2:    .word      Tabelle2

.Lfe1:
        .size      main,.Lfe1-main          @ Information über die Größe von main
        .comm      Tabelle2, TAB1ende-TAB1  @ Speicherbereich mit der Größe von
TAB1 reservieren
// End of File

```