Universität Mannheim
Lehrstuhl für Praktische Informatik IV
Prof. Dr. W. Effelsberg
Christoph Kuhmünch, Gerald Kühne

Praktische Informatik II
SS 2000

Übungsblatt 8 **Ausgabe: Mi, 28.06.00** **Abgabe: Di, 04.07.00, 18 Uhr**

Aufgabe 1: Assembler Programmierung [8 Punkte]

Erstellen Sie ein Assembler-Programm, welches den ggT zweier ganzer Zahlen $m, n \in \{1, \ldots, 32\,767\}$ nach dem Euklid'schen Algorithmus berechnet.

```
 *------   Berechnung des ggT zweier positiver Zahlen M,N aus [1, 65535]

 1           ORG     $0
 2           DC.L    $8000           Stack pointer value after a reset
 3           DC.L    START           Program counter value after a reset
 4
 5           ORG     $2000           Start at location 2000 Hex
 6
 7 START     MOVE.W q M,D5
 8           BEQ.S   ENDE            M auf ungleich 0 pruefen
 9           MOVE.W  N,D6
10           BEQ.S   ENDE            N auf ungleich 0 pruefen
11           CMP.W   D6,D5           D5 (min) < D6 (max) ?
12           BCS.S   GGT             ja   -> zum Anfang der Berechnung
13           EXG     D6,D5           nein -> Register tauschen
14
15 GGT       TST.W   D5              D5 = 0?
16           BEQ.S   ENDE            ja -> Berechnung abgeschlossen
17           DIVU.W  D5,D6           D6.L = D6.L / D5.W   (D6 = Rest|Ergebnis)
18           CLR.W   D6              D6 = Rest|0000
19           SWAP    D6              D6 = 0000|Rest
20           EXG     D5,D6           D5 und D6 vertauschen
21           BRA.S   GGT
22 ENDE      MOVE.W  D6,ERGEBNIS
23
24 M         DC.W    10              Konstantendefinition 1
25 N         DC.W    5               Konstantendefinition 2
26 ERGEBNIS  DS.W    1               Konstantendefinition 3
```

Anmerkungen:

- Die Zeilennummern sind zur besseren Übersicht hinzugefügt worden.
- Zeilen 8,12,16,21: `Bxx.S` – die Verwendung des Suffixes `.S` (= short) im Zusammenhang mit dem Branch-Befehl führt zur Generierung von kompakterem Code, da der Sprung dann auf einen kleineren Wertebereich beschränkt wird.
- Zeile 12: `BCS.S` – ist D6 größer als D5 so entsteht ein Unterlauf und das Carry-Flag wird gesetzt. Man könnte auf die Idee kommen, stattdessen `BLE` (branch on less or equal) zu verwenden. Dieser condition code bezieht sich aber auf vorzeichenbehaftete Zahlen. Wir rechnen hier aber mit positiven Zahlen.
- Zeilen 8,10: Das Programm wird beendet, wenn eine der Zahlen M, N gleich null ist. Statt zum Label `ENDE` zu verzweigen, wäre es sinnvoller, eine entsprechende Fehlerbehandlung einzusetzen.

Aufgabe 2: Matrizenmultiplikation in Assembler [12 Punkte]

Entwerfen Sie ein 68000-Programm zur Multiplikation zweier $n \times n$ Matrizen zu folgender Spezifikation [...].

C-Implementierung zum besseren Verständnis:

```
/* Matrixmultiplikation der NxN Matrizen A und B
   Das Programm ist auf die spaetere 68000-Implementierung ausgerichtet */

#include <stdio.h>

#define N 2

short int matrix1[N * N] = { 1, 2, 3, 4 }; /* Quellmatrix 1 */
short int matrix2[N * N] = { 5, 6, 7, 8};  /* Quellmatrix 2 */
short int matrix3[N * N];                  /* Ergebnismatrix */

int main()
{
    short int* A = matrix1; /* A0-Register */
    short int* B = matrix2; /* A1-Register */
    short int* C = matrix3; /* A2-Register */

    int row, column, sum, comp;
    short int* temp;

    for (row = 0; row < N; row++) {              /* alle Zeilen durchlaufen (D3) */
        for (column = 0; column < N; column++) { /* alle Spalten durchlaufen (D2) */
            temp = B;                            /* Startadresse von B retten */
            sum  = 0;                            /* Summe aus Zeile * Spalte ist 0 */
            for (comp = 0; comp < N; comp++) {   /* alle Komponenten einer
                                                    Zeile durchlaufen (D1)*/
                sum += A[comp] * B[column];      /* Aij * Bji */
                B   += N; /* Zeiger auf B auf die naechste Zeile setzen */
            }
            *C = sum;    /* Summe abspeichern */
            C++;         /* Zeiger auf C auf naechstes Element setzen */
            B = temp;    /* Zeiger auf B zurueckholen */
        }
        A += N;          /* Zeiger auf A auf die naechste Zeile setzen */
    }

    return 0;
}
```

Assembler-Implementierung:

```
                ORG     $0
                DC.L    $8000           Stack pointer value after a reset
                DC.L    START           Program counter value after a reset

                ORG     $2000           Start at location 2000 Hex

START           LEA     MATRIX1(PC),A0
                LEA     MATRIX2(PC),A1
                MOVE.W  GROESSE(PC),D0
                LEA     MATRIX3(PC),A2
                BSR.S   MATMULT         Multiplikation
                BREAK


*------ MATMULT
*  multipliziert die NxN Matrizen A, B und legt das Ergebnis
*  in der Matrix C ab
*
*  Parameter: A0 = Zeiger auf Matrix A
*             A1 = Zeiger auf Matrix B
*             A2 = Zeiger auf Matrix C
*             D0 = N

MATMULT         MOVEM.L D1-D5,-(SP)     lokal verwendete Register retten
                ASL.L   #1,D0           D0 * 2, da Wortlaenge der Matrixelemente
                MOVEQ   #0,D3           Zeilenzaehler = 0
MATZEILEN       MOVEQ   #0,D2           Spaltenzaehler = 0
MATSPALTEN      MOVEQ   #0,D1           Komponentenzaehler = 0
                MOVE.L  A1,-(SP)        Zeiger auf B retten
                MOVEQ   #0,D5           Ergebnis = 0
MATKOMP         MOVE.W  (A0,D1.W),D4    Zeilenelement von Matrix A
                MULS    (A1,D2.W),D4    D4 = Aij * Bji
                ADD.W   D4,D5           Produkte aufsummieren
                ADDA.L  D0,A1           naechstes Spaltenelement von B
                ADDQ.L  #2,D1           naechstes Zeilenelement von A
                CMP.L   D0,D1           alle Komponenten ?
                BNE.S   MATKOMP         nein -> naechste Komponente
                MOVE.W  D5,(A2)+        Ergebnis nach C
                MOVEA.L (SP)+,A1        Zeiger auf B zurueckholen
                ADDQ.L  #2,D2           Spaltenzaehler + 1
                CMP.L   D0,D2           alle Spalten ?
                BNE.S   MATSPALTEN      nein -> naechste Spalte
                ADDA.L  D0,A0           naechste Zeile
                ADDQ.L  #2,D3           Zeilenzaehler + 1
                CMP.L   D0,D3           alle Zeilen ?
                BNE.S   MATZEILEN       nein -> naechste Zeile
                MOVEM.L (SP)+,D1-D5     Register zurueckholen
                RTS

GROESSE   DC.W    2             Konstantendefinition 1
MATRIX1   DC.W    1,2,3,4       Konstantendefinition 2
MATRIX2   DC.W    5,6,7,8       Konstantendefinition 3
MATRIX3   DS.W    4             Konstantendefinition 4
```