Musterlösung - Übungsblatt 9

Aufgabe 1

a) *Was leistet dieses Programm ?*
Das Programm liest eine Long-Zahl ein und gibt ihre Ziffern sortiert aus.

b) *Wie wird Speicherplatz für die lokalen Variablen der Prozeduren geschaffen ? Wie sind die Variablen laenge, eingabe und sort auf dem Stack in main angeordnet ?*
Es wird immer zunächst der Befehl "link a6,#xx" ausgeführt. Dadurch wird auf dem Stack Platz für lokale Variablen geschaffen, die über den FramePointer (A6) angesprochen werden können. Der Offset "#xx" gibt dabei die Größe des reservierten Speicherplatzes (in Byte) an.
Die Variablen sind in main wie folgt abgelegt:
laenge: -4(a6)
eingabe: -8(a6)
sort: -48(a6) (10 mal 4, wobei sort[0] bei -48(a6) und sort[10] bei -12(a6) liegen)

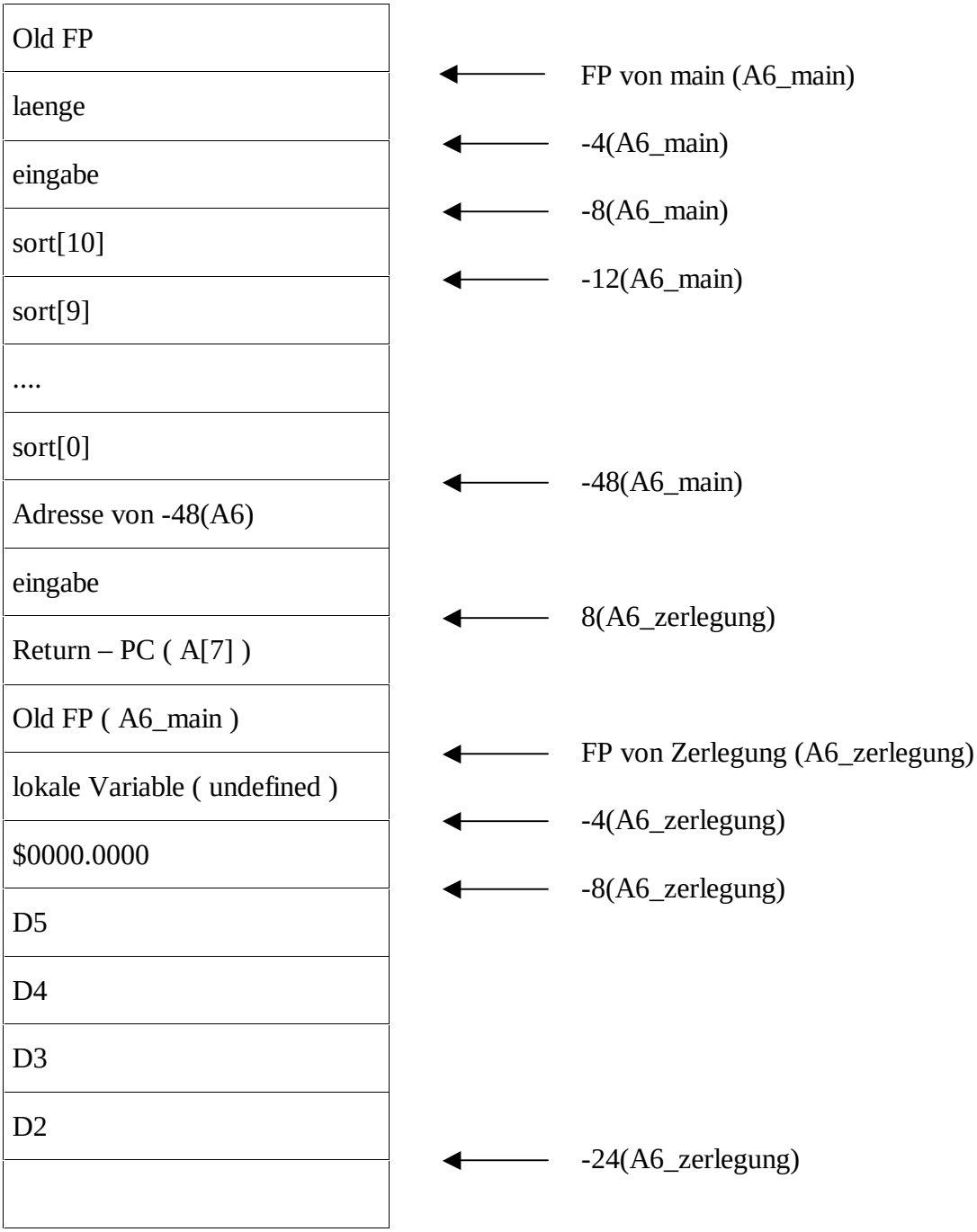
c) *Wie werden Variablen-Werte an Unterprogramme übergeben ?*
Die Werte der Variablen werden mittels "move n,-(sp)" auf den Stack gelegt. Ausnahme ist das Feld der sortierten Werte. Von diesen wird die Anfangsadresse auf den Stack gelegt.

d) *Wie werden Ergebnisse an das Hauptprogramm zurückgeliefert ?*
Die Werte werden, solange es einzelne Werte sind, in dem Register D0 übergeben. Die Werte des Sortierfeldes werden direkt im Speicher (auf dem Stack) geändert.

e) *Skizzieren Sie den Stack an der mit →***** gekennzeichneten Stelle. Gehen Sie dabei von einem eingelesenen Wert von 7428 aus.*
siehe Zeichnung.

f) *Wie wird der Unterprogrammaufruf durchgeführt ? Wie wird die Rücksprungadresse gespeichert ?*
Durch den Befehl "jsr label". Dabei wird die Rücksprungadresse auf den Stack gelegt.

Stackabbild bis zu markierten Stelle:



Aufgabe 2 – Version 1 (Parameterübergabe über den Stack)

```
*------ Berechnung des Binomialkoeffizienten
*          (N ueber K) = (N-1 ueber K) + (N-1 ueber K-1)
*
* C-Funktion (angepasst an das Assemblerporgramm):
*
* short int binom (short int N, short int K)
* {
*     short int temp = 1;
*     if (K == 0)
*        return temp;
*     if (N == K)
*        return temp;
*     temp  = binom(N - 1, K);
*     temp += binom(N - 1, K - 1);
*     return temp;
* }
*
* Anmerkungen:
* Die Parameteruebergabe erfolgt ueber den Stack,
* des weiteren wird zur Abspeicherung des Zwischenergebnisses
* nach der ersten Rekursion eine lokale Variable verwendet.
* Dadurch wird der Code an einigen Stellen recht umstaendlich, aber
* der Sinn und Zweck von LINK wird verdeutlicht.

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

PAR_N       EQU     4+8             Position von N auf dem Stack (relativ)
PAR_K       EQU     2+8             Position von K
RET_VAL     EQU     0+8             Position des Ergebnisses
                                    * Anmerkung: +8 resultiert aus der
                                    * Tatsache, dass die Ruecksprungadresse
                                    * und der Framepointer zunaechst
                                    * abgelegt werden

            ORG $2000               Start at location 2000 Hex

START       MOVE.W  N,-(SP)         Parameter N auf den Stack
            MOVE.W  K,-(SP)         Parameter K auf den Stack
            SUBQ.L  #2,SP           Platz fuer das Ergebnis schaffen
            BSR.S   BINOM           los geht's
            MOVE.W  (SP)+,ERGEBNIS      Ergebnis vom Stack holen
            ADDQ.L  #4,SP           Parameter vom Stack loeschen (2+2)
            BREAK

TEMP        EQU     -2              Position der lokalen Variable TEMP

BINOM       LINK    A6,#-2          Frame anlegen, Platz fuer die
                                    * lokale Variable schaffen
            MOVEM.L D0-D2,-(SP)     benoetigte Register retten
            MOVE.W  PAR_N(A6),D0    Parameter einlesen
            MOVE.W  PAR_K(A6),D1
            MOVE.W  #1,TEMP(A6)     Default-Ergebnis nach TEMP
            TST.W   D1              K = 0 ?
            BEQ.S   BINOMENDE       ja -> Rekursionsende
            CMP.W   D0,D1           N = K ?
            BEQ.S   BINOMENDE       ja -> Rekursionsende
```

```
            SUBQ.W   #1,D0          N = N - 1
            MOVE.W   D0,-(SP)       Parameter auf den Stack
            MOVE.W   D1,-(SP)
            SUBQ.L  #2,SP           Platz fuer Ergebnis schaffen
            BSR.S    BINOM          (N-1 ueber K)
            MOVE.W   (SP)+,D2       Ergebnis vom Stack holen
            MOVE.W   D2,TEMP(A6)    und zwischenspeichern
            ADDQ.L   #4,SP          Parameter N-1 und K vom Stack
            SUBQ     #1,D1          K = K - 1
            MOVE.W   D0,-(SP)       Parameter auf den Stack
            MOVE.W   D1,-(SP)
            SUBQ.L   #2,SP          Platz fuer Ergebnis schaffen
            BSR.S    BINOM          (N-1 ueber K-1)
            MOVE.W   TEMP(A6),D2    Zwischenergebnis laden
            ADD.W    (SP)+,D2       neues Ergebnis dazu addieren
            MOVE.W   D2,TEMP(A6)    und zwischenspeichern
            ADDQ.L   #4,SP          Parameter vom N-1 und K-1 loeschen
BINOMENDE   MOVE.W   TEMP(A6),D2    Zwischenergebnis laden
            MOVE.W   D2,RET_VAL(A6)      und als Endergebnis speichern
            MOVEM.L  (SP)+,D0-D2    Register restaurieren
            UNLK     A6             Frame loeschen
            RTS

N           DC.W 25                 Konstantendefinition 1
K           DC.W 5                  Konstantendefinition 2
ERGEBNIS    DS.L 1                  Konstantendefinition 3
```

Aufgabe 2 – Version 2 (Parameterübergabe in Registern)

```
*------ Berechnung des Binomialkoeffizienten
*           (N ueber K) = (N-1 ueber K) + (N-1 ueber K-1)
* Anmerkungen:
* Parameteruebergabe per Register

            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       MOVE.W   N,D0           Parameter N nach D0
            MOVE.W   K,D1           Parameter K nach D1
            BSR.S    BINOM          los geht’s
            MOVE.W   D6,ERGEBNIS    Ergebnis speichern
            BREAK

BINOM       MOVEM.L  D0-D2,-(SP)    benoetigte Register retten
            MOVE.W   #1,D6          Default-Ergebnis setzen
            TST.W    D1             K = 0 ?
            BEQ.S    BINOMENDE      ja -> Rekursionsende
            CMP.W    D0,D1          N = K ?
            BEQ.S    BINOMENDE      ja -> Rekursionsende
            SUBQ.W   #1,D0          N = N - 1
            BSR.S    BINOM          (N-1 ueber K)
            MOVE.W   D6,D2          Zwischenergebnis speichern
            SUBQ     #1,D1          K = K - 1
            BSR.S    BINOM          (N-1 ueber K-1)
            ADD.W    D2,D6          neues Ergebnis dazu addieren
BINOMENDE   MOVEM.L  (SP)+,D0-D2    Register restaurieren
            RTS
```

```
N           DC.W 25    Konstantendefinition 1
K           DC.W 5     Konstantendefinition 2
ERGEBNIS    DS.L 1     Konstantendefinition 3
```

Aufgabe 2 – Version 3 (Parameterübergabe über Stack, Ergebnis in Register)

```
* Das Programm berechnet rekursiv den Binomialkoeffizienten ( n , k )
* ( n , k ) = ( n-1 , k ) + ( n-1 , k-1 )
* Register: D0 => n, D1 => k, D6 => Ergebnis


            ORG         $0
            DC.L        $8000
            DC.L        START
            ORG         $2000

START       MOVE.W      #14,D0      * Initialisierung: n = 14
            MOVE.W      #5,D1       * k = 5
            MOVE.L      #0,D6       * Ergebnis = 0

            MOVE.W      D0,-(SP)    * n & k werden auf den Stack geschrieben,
            MOVE.W      D1,-(SP)    * d.h. sie werden an die Routine BINOM übergeben

            JSR         BINOM       * die Unterroutine BINOM wird aufgerufen

            MOVE.W      (SP)+,D1    * nach dem Ende der Routine sind die Originale
            MOVE.W      (SP)+,D0    * von D0 und D1 noch auf dem Stack
                                    * das Ergebnis steht im Register D6
            BREAK


* rekursive Unterroutine BINOM
* diese Routine schreibt solange die Werte von n-1 und k(k-1) auf den Stack,
* bis k==0 oder k==n ist. Nur falls diese Bedingung erfüllt ist, wird das
* Ergebnis verändert. Es wird dann eine 1 addiert.


BINOM       MOVE.W      6(SP),D0    * aktueller Wert von n wird vom Stack gelesen
            MOVE.W      4(SP),D1    * aktueller Wert von k wird vom Stack gelesen
            CMP.W       #0,D1       * ist k == 0 ?
            BEQ         BASIS       * falls ja, ist die Basis der Rekursion erreicht
            CMP.W       D0,D1       * ist k == n ?
            BEQ         BASIS       * falls ja, ist ebenfalls die Basis erreicht

            SUBQ.W      #1,D0       * n' = n - 1, s. Formel

            MOVE.W      D0,-(SP)    * ein rekursiver Aufruf mit n' = n - 1 und
            MOVE.W      D1,-(SP)    * k' = k wird vorbereitet

            JSR         BINOM       * rekursiver Aufruf, es wird (n-1,k) berechnet
```

```
        MOVE.W    (SP)+,D1      * die vorherigen Werte
        MOVE.W    (SP)+,D0      * für n und k werden vom Stack gelesen

        SUBQ.W    #1,D1         * k' = k - 1, s. Formel

        MOVE.W    D0,-(SP)      * ein rekursiver Aufruf mit n' = n - 1 und
        MOVE.W    D1,-(SP)      * k' = k - 1 wird vorbereitet

        JSR       BINOM         * rekursiver Aufruf, es wird (n-1,k-1) berechnet

        MOVE.W    (SP)+,D1
        MOVE.W    (SP)+,D0

        RTS


BASIS   ADD.L     #1,D6         * falls die Basis der Rekursion erreicht wurde,
        RTS                     * wird eine 1 zum Ergebnis dazuaddiert,
                                * denn ( n , 0 ) = ( n , k ) = 1
```