Name: Matrikelnummer:

Aufgabe 1: Assembler (16 Punkte)

o (a) **(1 Punkt)** Was wird in den so genannten "*flags*" (im Statusregister) gespeichert? Nennen Sie eine Gruppe von MC68000-Befehlen, die diese Informationen auswertet.

Bedingungscodes (Flag), enthalten Aussagen über das Ergebnis vorgangengener Operationen (arith/log.), z.B. Überlauf oder Null
- Bedingte Sprungbefehle
Bedingte Sprungbefehle werten diese aus.

- Z: wird gesetzt, wenn das Erg.s der Operation Null liefert.
- N: wird gesetzt, wenn das Erg.s der Operation einen wert liefert, der als negative Zahl interpretiert werden kann
- C: wird gesetzt, wenn die Zahl groß/nicht passt und positiv
- V: wird gesetzt, " " " nicht passt und negativ.

o (b) **(1 Punkt)** Was ist indirekte Adressierung?
Register Register als Zeiger. MOVE.W (A1),D1 Sprungbefehl BMI, BPL, BNE, BEQ

wenn die in Befehl angegebene Adresse (oder das Register) nicht selbst Ziel der Operation ist, sondern die Adresse des Operanden im Speicher enthält

LDA @A0 MOVE.L D0,(A0)

Der wert steht im Speicher an der Adr., die im Register steht

mens: (Das angegebene Ziel enthält die Adr. des Ziels).
wenn die im Bef angegebene Adr. (oder das Register) nicht selbst Ziel der Operation ist, sondern die Adr. des Operanden im Speicher enthält.

o (c) **(1 Punkt)** Worin besteht der Unterschied zwischen den Befehlen BRA und JSR des 68000er?

Sprung BRA: ein einfacher Sprungbefehl.
JSR: Sprung zum Unterprogramm.

BRA: unbedingter, relativer Sprung

JSR: Unterprogrammaufruf rettet PC auf Stack, absolute Adressierung

BRA: unbedingte Sprung, relative Adressierung.

JSR: Sprung zu einem Unterprogramm, wobei der PC auf den Stack GELEGT wird. Absolute Adres-g.

JSR - Sprung zum Unterprogramm, wird im Stack RSA abgelegt.

BRA - Unbedingte Sprung.

Punkte	

Name: *Matrikelnummer:*

Befehlsfolgen

(d) **(2 Punkte)** Die folgenden anfänglichen Register- und Speicherinhalte sind bekannt:

A0= 1012
A1= 1004
A2= 1008
D1= -8

Speicherstelle (Langwort) 1000= 1
Speicherstelle (Langwort) 1004= 2
Speicherstelle (Langwort) 1008= 3

Welchen Inhalt besitzen die Register NACH den folgenden Befehlen? Hinweis: Vor dem ersten Befehl enthalten die Register- und Speicherstellen die oben genannten Werte. Der zweite Befehl wird nach dem ersten ausgeführt, mit den Werten in den Registern und dem Speicher, die der erste Befehl hinterlassen hat.

```
1. LEA 4(A2,D1.L),A0
2. MOVE.L -(A0),D1
```

	A0	A1	A2	D1	1000	1004	1008
1	1004	1004	1008	-8	1	2	3
2	1000	1004	1008	1	1	2	3

D1.L – 4 Byte
D1.W – 2 Byte
D1.B – 1 Byte

~~-(2) = 1 → D1~~
· [A0] ← [A0] − 4
[D1] ← [M([A0])]

1. 4 + (1008 + (−8)) = 1004 → A0
2. (1004 − 4) = (1000) → ...
1 → D1

(e) **(1 Punkt)** Würde der Sprungbefehl verzweigen? Begründen Sie Ihre Antwort!

```
MOVE.L #256,D0
TST.B D0
BEQ Irgendwohin
```

1 Bei

0000, 0001, 0000,0000 – unterste 8 Bit = 0 werden ausgewertet ⇒ verzweigt.

31 [| | 1 | 0000 0000] 0

TST.B D0
BEQ D0
BEQ Sprung
Befehl wird ausgeführt

ja, Sprungbefehl verzweigen

$256_{10} \rightarrow 0000\ 0001\ 0000\ 0000_2$

da TST.B → 1 Byte

und BEQ ...

Если было бы 255, то 0000 0000 1111 1111 ⇒ (1 Byte)

тогда BEQ ... была бы пропущена.

?? Hexadezimal

z.B. $7b diese Zahl ist Hexadezimal

Kak $7b = 111 1011₂ = 2⁰+2¹+2³+2⁴+2⁵+2⁶ = 123 in Dezimal

7 = 111, b = 1011

Punkte	

Name: *Matrikelnummer:*

Unterprogramme

Ein über Deutschland befindliches Sportflugzeug muß notlanden. Aus den zur Verfügung stehenden Flughäfen soll ein freier mit der kürzesten Entfernung zum Flugzeug ausgewählt werden, sofern er sich innerhalb der möglichen Reichweite befindet.

Entwickeln Sie hierzu eine Assembler-Funktion, die das Array der Flughäfen in Deutschland durchsucht und die Nummer des (für das Flugzeug) nächsten freien Flughafens zurückliefert. Das erste Element des Arrays hat hierbei die Nummer 0.
Befindet sich kein freier Flughafen in Reichweite, so soll –1 zurückgeliefert werden.

Die Assembler-Funktion soll folgenden Prototypen in C-Notation besitzen:

```
int SucheFlughafen(EinFlughafen *FH, int AnzFH, EinFlugzeug *FZ);
```

Hierbei ist `FH` die Adresse des ersten Flughafens im Flughafen-Array, `AnzFH` die Anzahl der Flughäfen und `FZ` ein Zeiger auf die Struktur `EinFlugzeug` im Speicher (dort befindet sich die Variable `Flugzeug`).
Der Rückgabewert soll als Langwort in D0 zurückgeliefert werden und die Nummer des nächsten Flughafens beinhalten oder –1, falls sich keiner in Reichweite befindet.

Die Strukturen sind folgendermaßen definiert:

```
struct EinFlughafen
    {
        int Laenge, Breite;   // Koordinaten des Flughafens
                              // (Längen- und Breitengrad)
        int frei;             // 0= besetzt, 1= frei
    };


struct EinFlugzeug
    {
        int Laenge, Breite;   // ungefähre Position des Flugzeuges
        int Reichweite;       // maximale Reichweite in Kilometern
    };


#define AnzFlughaefen 120     // Anzahl der Flughäfen in Deutschland

EinFlughafen AlleFlughaefen[AnzFlughaefen];
                              // Array aller Flughäfen

EinFlugzeug Flugzeug;         // das in Not befindliche Flugzeug
```

Der Funktion wird demnach `AlleFlughaefen`, `AnzFlughaefen` und die Adresse von `Flugzeug` übergeben.

Punkte	

Name: *Matrikelnummer:*

(f) **(1 Punkt)** Geben Sie den Kellerinhalt und den Kellerzeiger nach dem Aufruf der Funktion `SucheFlughafen` unmittelbar nach Sichern der Register an. Sie können die vorgegebenen Kästchen als Zeichenhilfe verwenden. Ein Kästchen sei dabei 4 Byte groß.

? Kann man D1–D7 und A0–A7 reservieren

	Distanz zum SP	Inhalt
32	36	*FZ = # Flugzeug
28	32	AnzFH
24	28	*FH = Alle Flughäfe
20	24	RSA Rücksprungadresse
16	20	D4
12	16	D3
8	12	D2
4	8	D1
0 →	4	A1
SP →	0	A0

*FZ

AnzFH

*FH

RSA

A0

A1

D1

D2 D0 für Ergebnis reserviert

D3

A1, A0 – benutzen für Zeiger (Adr)

D1 – D4 – benutzen für Daten

Punkte	

Name: *Matrikelnummer:*

(g) **(9 Punkte)** Schreiben Sie die Funktion in Motorola 68000-Assembler. Berücksichtigen Sie die oben angegebenen Anforderungen.

Prozeduren ohne Dokumentation werden mit Null Punkten bewertet!

Hinweis nur für Wiederholer: Die Prozedur kann sinngemäß auch in Modula-2-Inline-Assembler implementiert werden. Geben Sie dazu den Prozedurkopf an!

```
Init                                   حل خلف الصفحة
A0   akt FN                            يجب حل جيد.
A1   Fz

FN frei
Distanz berechnen. Reichweite < Distanz → Next.FN
Destanz > min Distanz → Next FN
Update

NextFN
    ENDE → Do-Erg
```

```
.GLOBAL  _SucheFN
-SucheFN: MOVEM.L  A0-A1/D1-D4,-(SP)   | Reg sichern
          MOVEA.L  28(SP),A0           | A0 = Alle FH
          MOVEA.L  36(SP),A1           | A1 = Fz
          MOVE.L   32(SP),D1           | D1 = Anzahl FH
          MOVE.L   #FFFFFF,D2          | D2 = min Dist
          MOVE.L   #-1,D3              | D3 = letzte FH
          MOVE.L   #0,D4               | D4 = aktu.FH
          MOVE.L   4(A1),-(SP)         | Breite (Fz) → Stack
          MOVE.L   (A1),-(SP)          | Länge (Fz) → Stack
```

Fortsetzung auf Seite 7

Punkte	

Name: Matrikelnummer:

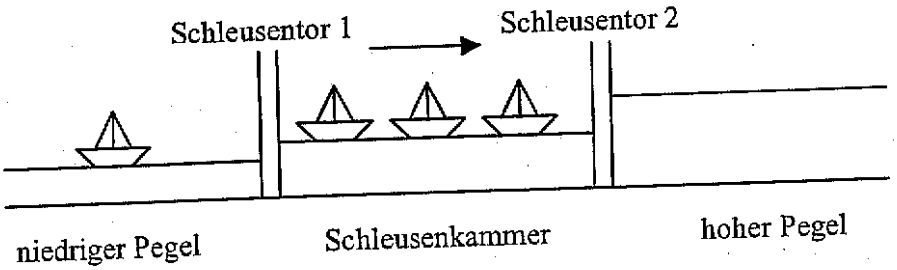
Aufgabe 2: Semaphore **(10 Punkte)**

(a) **(1 Punkt)** Beschreiben Sie **kurz** (z.B. in Stichpunkten) den Unterschied zwischen dem ersten und dem zweiten Leser-Schreiber-Problem.

1 LSP: schreiber haben priorität gegenüber leser
1 LSP: schreiber haben keine " " "
1 LSP – Keines der beiden Prozeßtypen ist gegenüber andern priorisiert.
2. LSP – Einer Prozesstyp hat Priorisierung gegenüber anderen.

Modellierung einer Schiffs-Schleuse

Eine Schiffs-Schleuse dient zur Überbrückung von Höhenunterschieden zwischen verschiedenen Gewässern. Dabei fahren die Schiffe durch das Schleusentor in eine Schleusenkammer, deren Wasserspiegel nach dem Schließen dem jeweils anderen Wasserpegel angepasst wird.



Die Schleuse sowie die Schiffe sollen durch Prozesse modelliert und mit Hilfe von Semaphoren synchronisiert werden. Dabei sollen folgende Bedingungen gelten:

1. In der Schleusenkammer ist Platz für genau drei Schiffe.

2. Aus Kostengründen wird der Wasserspiegel der Schleusenkammer erst angepasst, wenn sich genau drei Schiffe in der Schleusenkammer befinden.

3. Schiffe fahren nur von links nach rechts.

Im folgenden sehen Sie, wie die oben geschilderten Synchronisationsbedingungen mit Semaphoren implementiert sind. Leider sind dabei ein paar **Fehler** unterlaufen.

Globale Deklarationen:

```
Tor1              :  Semaphor(0)
Tor2              :  Semaphor(0)
Schleuse          :  Semaphor(0)
Zähler_Schutz     :  Semaphor(1)
alle_Schiffe_da   :  Semaphor(0)
Schiffs_Zähler    :  INTEGER = 0
```

Semaphore (3)

Punkte	

Name: _____ Matrikelnummer: _____

Prozeß Schleuse:

```
200 REPEAT
210   IF Pegel_unten THEN
220     -- öffne Tor 1   END IF
230   // Pegel_unten ist wahr, wenn der
240   // Pegel der Kammer dem des
250   // niedrigen Gewässers entspricht
260     Tor1.V
270     -- warten bis drei Schiffe
280     -- eingefahren sind
290     Schleuse.P
300     -- Tor 1 schließen und pumpen
310   IF Pegel_oben THEN
320     -- öffne Tor 2   END IF
330   // Öffne Tor2, wenn der Pegel der
340   // Kammer gleich dem Pegel des hohen
350   // Gewässers ist
360     Tor2.V
370     -- warten bis die drei Schiffe
380     -- rausgefahren sind
390     Schleuse.V
400     -- Tor 2 schließen und pumpen
410   END REPEAT
```

Punkte

Prozeß Boot:

```
500 Zähler_Schutz.P
510   INC(Schiffs_Zähler)
520   IF Schiffs_Zähler = 1 THEN
530     Tor1.P
540   END IF
550     -- einfahren
560   IF Schiffs_Zähler = 3 THEN
570     Schleuse.V
580     Tor2.P
590     FOR i:=1 TO Schiffs_Zähler-1 DO
600       alle_Schiffe_da.V
610     END FOR
620     Zähler_Schutz.V
630   ELSE alle_Schiffe_da.P
640   END IF
650   -- rausfahren
660   Zähler_Schutz.P
670   DEC(Schiffs_Zähler)
680   IF Schiffs_Zähler = 0 THEN
690     Schleuse.V
700   END IF
710   Zähler_Schutz.V
```

Informatik 4 (Oktober 2000)

Name: *Matrikelnummer:*

(b) **(3 Punkte)** Simulieren Sie von Hand soweit wie möglich die in der Tabelle angegebene Abfolge von neu eintreffenden sowie die Schleuse verlassenden Schiffs-Prozesse sowie den Schleusen-Prozeß für die gegebene (**nicht korrekte**) Implementierung. Schiffe verlassen die Schleuse nur dann, wenn das als Ereignis in der Tabelle angegeben ist. Geben Sie pro Zeile jeweils den Endzustand an, bei dem jeder Prozeß entweder blockiert oder komplett abgearbeitet ist. Werte, die mit der Vorgängerzeile übereinstimmen, brauchen nicht aufgeschrieben zu werden. Die Notation ".zähler" steht für die Semaphorzähler. Im Anfangszustand entspricht der Pegel der Schleusenkammer dem Pegel des niedrig gelegenen Gewässers.

Hinweis: Da die gegebene Implementierung fehlerhaft ist, kann es passieren, daß die links angegebenen Ereignisse nicht stattfinden, da die entsprechenden Prozesse noch blockiert sind.

Ereignis	**Schiffs_zähler**	**Zähler_Schutz.zähler**	**alle_Schiffe_da.zähler**	**Schleuse.zähler**	**Tor1.zähler**	**Tor2.zähler**	
Anfangszustand	0 0	1 1	0 0	0 0	0 0	0 0	
Schleusenprozeß wird gestartet 290	? 1	? 0		-1 -1	1 1		290
Schiff 1 kommt an 630	1	0	-1 -1		0		630
Schiff 2 kommt an 630	2	-1 -1	-2	-			500
Schiff 3 kommt an 580	3	2 1	0	0		-1	500
Die Schleuse pumpt				1		0	(Wie schon korrigiert.)
Die Schiffe verlassen die Schleuse							(1P)

Punkte	

Name: *Matrikelnummer:*

(c) **(6 Punkte)** Die gegebene Implementierung hat 3 logische Fehler (die jedoch mehr als 3 Stellen im Programmtext betreffen und nicht alle in Teil a aufgetreten zu sein brauchen). Geben Sie in der folgenden Tabelle die Auswirkungen zu jedem dieser Fehler an. Beschreiben Sie die erforderlichen Korrekturmaßnahmen in Stichpunkten.

Fehlverhalten	**Korrekturmaßnahme**
Read-lock, wenn das erst Schiff eingefahren ist können keine weiteren einfahren. Zwischen Zeilen 540 und 550 soll Zähler_schutz.V stehen. Bei ersten Schiff wird alles blockieren. und Zeil 620 muss weg. Deadlock DR Kein Schiff mehr den KA betreten kann	545 : Zähler_schutz.V 545 : Zähler_schutz.V 545 Zähler schutz.V
Schleuse wartet mit runter pumpen nicht, bis alle Schiffe raus sind. Es kann mehr als 3 Schiffe einfahren und Zähler wird nie Null sein. Es können mehr als 3 Boot einfahren	390 : Schleuse.P 160 Boot : Semaphor(3) 490 Boot.P 720 Boot.V 160 Boot : Semaphore(3) 499 : Boot.P 711 Boot.V
Es können mehr als 3 Boote einfahren Schleuse nicht wartet bis die 3 Schiffe rausfahren. Schleuse wartet nicht, bis alle Schiffe ausgefahren sind	160: Boot : Semaphore(3) 490 : Boot.P 711 Boot.V. 390 Schleuse.P 390 schleuse.

Punkte	

Aufgabe 2 Semaphoren

a) L-L-S problem:
- Viele Leser
- ein Schreiber
- sofern gelesen wird, wartet der Schreiber
- Schreiber sind nicht priorisiert
- Leser sind priorisiert, überh. warten sie

2 L-S-problem
- Viele Leser
- ein Schreiber
- sofern gelesen wird, wartet der Sch.
- Schreiber sind priorisiert, d. h. die überholen evtl. wartende Leser

	Zähler dyn.	Zähler Schuss	alle S. da	Schleuse	Tor 1	Tor 2	Z. erreicht	Soll
...d	0	1	0	0	0	0	/	/
...chleuse				−1 Schleuse wird nach Schuss blockiert	1 ?		250	250
...schiff ... an	1	0	−1 Sl-b.		0		630	630
...9 an		−1 b.					500 ✓	630
...3 an		−2 b.					500 ✓	580

Informatik 4 (Oktober 2000)

Name: *Matrikelnummer:*

(8 Punkte)

Aufgabe 3: Nachrichten

(a) (1 Punkt) Worin unterscheidet sich die Synchronisation über Nachrichten von der mit Semaphoren?

N

Nachrichten: implizite, einseitige, Synchronisation (Empfänger verzögert

Semaphoren: - ein- und mehrseitige Synchronisation
- Mechanismus wird im BS implementiert.

Semaphore: Mehr- und Einseitige Synch-ion
Implementierung im BS

(b) (1 Punkt) Wann ist ein Nachrichtenaustausch synchron und wann asynchron?

Ein Nachrichtaustausch ist asynchron, wenn in der Lage ein ~~und~~ Puffer ist, und wenn keinen, dann spricht man ~~dann~~ von synchron Nachrichtaustausch.

blockierende Operationen

Synchron: sender und Empfänger verwenden bzw. Empfangsoperation

Asynchron: sonst.

Synchron: Senden und empfangen
verwenden Blockierenden
Senden und empfangen

Punkte	

Klausur 2000

Aufg 4

a) Flag ist ein Bedingt, der eine Aussage über die Verg. Op's liefert. z.B. ob bei der Addition ein Überlauf (C - Carry bit) aufgetreten ist.
Gruppe von Befehlen: bedingte Sprungbefehle.
z.B.: $b_3 \in Q$ wurde Zero-bit aus.

b) das Reg enthält eine Adresse (Zeiger) des Elementes, der im Speicher liegt. Bei Op's wird aus den Operanden indirekt über seine Adresse im Reg zugegriffen.

c) BRA - Sprung an bel. Stelle des Programms
der Wert von PC + wird + gerechnet

JSR - Sprung ins Unterprogramm
der PC - Stand wird in Stackpointer gespeichert
Anschließend wird im Speichenstack gespeichert (gerettet).

d)

	A0	A1	A2	b1	1000	1004	1008
Vor	1012	1004	1008	-8	1	2	3
1.	1004	1004	1008	-8	1	2	3
2.	1000	1004	1008	1	1	2	3

1008 - 8 = 1000

↑ ↑ ↑

A2 b1

e)

2^8

1 0 0 0 0 0 0 0 0

9-stelliges Bit.

Für die Zahl 256 braucht man ein 9-stelliges Bit. Die ersten 8 Bits sind = 0.

⇒ TST.B wird ausgeführt (byte) die ersten 8 Bits testen und feststellen, dass die Zahl = 0 ist ⇒ Wird verworfen.

Name: *Matrikelnummer:*

Meßwertverarbeitung mit Nachrichten

Mit Hilfe von Nachrichten soll der Mittelwert von zwei Messwerten bestimmt werden. Zur Steuerung dienen die Prozesse `Sensor1`, `Sensor2` und `ZBE` (zentrale Berechnungseinheit). Die Prozesse `Sensor1` und `Sensor2` messen in regelmäßigen Zeitabständen die Eingangsgrößen $x_1(t)$ und $x_2(t)$ zum Zeitpunkt t. Der Prozess `ZBE` berechnet das arithmetische Mittel $x(t) = (x_1(t) + x_2(t))/2$ von $x_1(t)$ und $x_2(t)$. Der Mittelwert x(t) kann nur berechnet werden, wenn beide Messwerte $x_1(t)$ und $x_2(t)$ vorliegen.

Zur Nachrichtenübertragung steht ein Kanal für den Nachrichtenaustausch zwischen `Sensor1` und `ZBE` und ein weiterer Kanal für die Kommunikation zwischen `Sensor2` und `ZBE` zur Verfügung. Nachrichten werden in jedem Kanal in einem Puffer mit N Plätzen zwischengespeichert. Das Senden einer Nachricht in einen Puffer und das Empfangen einer Nachricht daraus geschieht mit den Operationen:

```
PROCEDURE send(wert:REAL, kanal:TKanal);
PROCEDURE receive(VAR wert:REAL, kanal:TKanal);
```

Diese vorgegebenen Prozeduren (nicht selbst zu programmieren!) erfüllen folgende Eigenschaften:

1. Beim Empfangen wird der Wert aus dem Puffer automatisch entfernt.
2. Ist beim Senden bzw. beim Empfangen einer Nachricht der Puffer voll bzw. leer, wird der ausführende Prozess blockiert.

(c) **(1 Punkt)** Sind die Sendeoperationen der Prozesse Sensor1 und Sensor2 nicht-blockierend, blockierend oder pufferblockierend?

Puffer - blockierend

Puffer - blockierend

(d) **(5 Punkte)** Vervollständigen Sie bitte die Vorgaben auf der folgenden Seite für die drei Prozesse Sensor1, Sensor2 und ZBE. Zwischen verschiedenen Empfangsoperationen soll stets nichtdeterministisch ausgewählt werden (Stichwort: selektives Empfangen). Deklarieren Sie an den entsprechenden Stellen die Variablen. Vergessen Sie nicht die Mittelwertberechnung.

Es ist nicht notwendig, die genaue Syntax einer bestimmten Programmiersprache zu verwenden. Die Lösung muß aber klar erkennbar sein

Punkte	

Name: _____

2 * zgl. Kanäle buffer-blockierend

```
-- Globale Variablen
VAR
  K1, K2 : T_Kanal        K1, K2 : TKanal;
```

Sensor1: process

```
VAR
  x1 : Real               int x1;
                          A1 : Nachricht.
BEGIN
  REPEAT
    -- Wert x1 messen
    Send (x1, K1);
    wait ( xy );          → Zeit anfang
    Send (x1, K1)
  end REPEAT
END process;
```

Sensor2: process

```
VAR
  x2 : Real               int x2;
                          A2 : Nachricht.
BEGIN
  REPEAT
    -- Wert x2 messen
    Send (x2, K2);
    wait ( xy )           → Zeit ende.
    Send (x2, K2)
  end REPEAT
END process;
```

ZBE: process ~~nicht Determ.~~ / Auswahl.

```
VAR
  x1, x2, x : Real
  D1, D2 : Nachricht
  int ~~erg~~ erg
BEGIN
  REPEAT
    Select                          select
    . receive (x1, K1);             receive (x1, K1);
                                    receive (x2, K2);
    receive ( x2, K2);              erg = (D1+D2)/2;
  □ receive (x2, K2);             □ receive (x1, K2);
                                    receive (x1, K1);
    receive ( x1 , K1 );            erg = (D1+D2)/2;
    end select                      end select
                                    end REPEAT
    x := (x1+x2)/2
```

Matrikelnummer

Unterprogramme

Solange es noch FlugHöhen gibt:

nehme einen (die 1.) FH.

prüfe ob frei = 1.

(1)

ja - frei → berechne die Distanz

nein - bereits → inc Zähler

Vgl mit d. Rw des Fahrzeugs

(2)

OK → Vgl mit der Rw davon

not ok → inc Zähler

(3)

kleiner / größer

inc Zähler

Каждая Abzweigung

это относительно R (признак)

Концу каждого Abzweigung

относительно признаку

на следующий элемент

von Array.

признак

Если заполнен

*F8	72
AnzFH	68
*FH	64
RSH	60
b0	56
b1	52
b2	48
b3	44
b4	40
b5	36
b6	32
b7	28
A0	24
A1	20
A2	16
A3	12
A4	8
A5	4
A6	0

```
.Global  - Suche Flughafen
.EVEN
- Suche-Flughafen:

        MOVEM.L  D0-D7/A0-A6, (-SP)  | retten d. Reg.
        MOVEA.L  72(SP), A0          | Adr. d Flugzeuges
        MOVEA.L  68(SP), A1          | Adr. der Anz. der
                                     | FH.
        MOVEA.L  64(SP), A2          | Adr des Anf. des
                                     | FH-Arrays.

| Initialisierung.
        MOVE.L   #0, D2              | Zähler des akt. se
                                     | Feldelement FH.
        MOVE.L   8(A0), D3           | möchte keine
                                     | FH Res.
        MOVE.L   #(-1), D4           | Nr d. FH mit
                                     | der bisher kurz.
                                     | Entfernung

LOOP    CMP      D2, (A1)
        BEQ      Ende.
        CMP      #0, 8(A2)           | Prüfe, ob berechs.
        BEQ      nächster-FH
        CMP      #1, 8(A2)           | Prüfe ob frei
        BEQ      Berechne-RW         | Prozedur, in der
                                     | RW berechnet wird
```

```
berechneRW    -> bistanz( (A1), 4(A1), (A0), 4(A0))
                 | berechnet die Kosten zum FH.
              CMP   D0, 8(A0)      | vgl. mit der RW
                                   | des Flugzeuges
              BLT   Vergleich      | Falls die RW des
                                   | Flugzeuges ausreicht
                                   | wird FH evtl. notiert
              BGT   nächsterFH     | Falls nicht nehme
                                   | den n. FH.

Vergleich     CMP.L D0, D3         | prüfe ob neue Entf.
                                   | kleiner ist als alte
                                   | d.h. besser FH
                                   | gefunden
              BLT   NotiereRW      | Speicher die Nr des
                                   | neu gef. FH/1.
              BGT   nächsterFH     | es nicht kürzer
                                   | als davor gef.-e.

NotiereRW     MOVE.L D0, D3        | notiere neue RW
              MOVE.L D2, D4        | den neg. FH
              BRA nächsterFH.      | merken.

nächsterFH    ADDA.L #12, (A2)     | nehme den n.
                                   | FH
              ADD.L  #1, D2        | ist Zähler der
                                   | FH's
              BRA LOOP.

Ende          MOVE.L D4, D0        | Ers.
              MOVEM.W (SP+), A6-A0 / D7-D0
              | wiederherstellen der Register.
```