Lehrstuhl für Informatik IV
Rhein.-Westf. Techn. Hochschule Aachen
Prof. Dr. Otto Spaniol

Diplom-Vorprüfung „Systemprogrammierung"

(WS 1990/91)

Aufgabe 1: Central-Server-Model (3+1 Punkte)

In einer Firma ist die FAX-Verschickung auf folgende Weise geregelt :

Man wählt einen FAX-Server an und schickt das FAX dorthin. Der Server hat 6 FAX-Leitungen zur Verfügung um FAX-Mitteilungen zu verschicken. Der Server hat eine Kapazität von 60 abfertigbare Jobs pro Sekunde. Das System verteilt die FAX-Jobs wie folgt auf die 6 FAX-Leitungen :

25 % der FAX-Jobs gehen jeweils an die FAX-Geräte Nr. 1 und Nr. 2,
15 % der FAX-Jobs gehen jeweils an die FAX-Geräte Nr. 3 und Nr. 4,
10 % der FAX-Jobs gehen jeweils an die FAX-Geräte Nr. 5 und Nr. 6.

Zwischen den Zeitpunkten [0:T] wird die Kapazität des Servers zu 1/3 ausgenutzt.

Berechnen Sie für das Zeitintervall [0:T] :

a) Die Busyzeit des Servers und die Ankunftsraten an den FAX-Geräten.

b) Gesamtdurchsatz des Systems, wenn jedes FAX-Gerät 10 Jobs/Sek. fehlerfrei bedienen kann, d.h. daß FAX-Jobs.das System nach dem Absenden direkt verlassen.

Aufgabe 2: Operationelle Analyse (4+2+2 Punkte)

Ein *saturiertes* und *stationäres* System mit einer Bedieneinheit wird 10 Sekunden lang beobachtet. Die mittlere Servicezeit und die mittlere Responsezeit pro Job werden gemessen und betragen 0,5 Sekunden bzw. 2 Sekunden.

Hinweis: Geben Sie bei allen zu berechnenden Werten die Einheiten an!

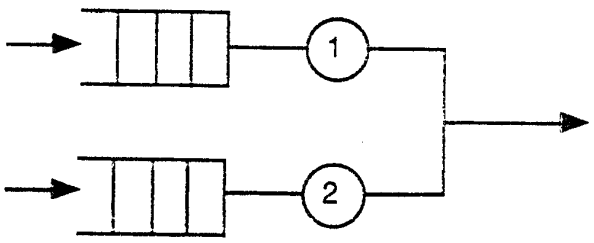
a) Ermitteln Sie für das oben beschriebene System die Basisgrößen

A := # Ankünfte in [0:10]
B := Busyzeiten des Servers in [0:10]
C_{job} := # beendeter Jobs in [0:10]
Z := # von Zeiteinheiten aller Jobs in [0:10]

b) Leiten Sie daraus folgende Leistungsgrößen ab :

λ := Ankunftsrate
X := Abgangsrate
U := Auslastung des Servers
$\bar{n}$:= mittlere Zahl Jobs im System

c) Zwei der oben beschriebenen Single-Server-Systeme werden wie folgt zu einem Gesamtsystem verbunden:



Berechnen Sie für dieses System die in Aufgabenteil b) angegebenen Leistungsgrößen.

Aufgabe 3 : **Buddy-Systeme** (3+5 Punkte)

Gegeben sei ein 16 KByte (K=1000) Speicher. Folgende Speicheranforderungen seien der Reihe nach zu bedienen: 3 , 3 , 4, 1, 2, 2 (Kbyte-Angaben). Geben Sie die Endbelegung des Speichers in Baumstruktur an, die sich bei Benutzung folgender Buddy-Systeme ergeben:

a) *"normales"* Buddy-System

b) *gewichtetes* Buddy-System der Vorlesung

Geben Sie die Belegungsreihenfolge und den auftretenden Verschnitt an!

Aufgabe 4 : **Endliche Automaten und Seitenersetzungsstrategien** (5+5+2 Punkte)

Gegeben sei das Seitenersetzungsverfahren CLIMB (d.h. wie FIFO, mit der Erweiterung, daß aufgerufene Seiten, die bereits im Speicher vorhanden sind, eine Position im Sinne von FIFO nach oben aufrücken und sich somit von der Auslagerungsposition entfernen). Der Speicher umfaßt drei Speicherplätze.

a) Geben Sie einen endlichen Automaten für CLIMB an (graphische Darstellung), der nur den gefüllten Speicher mit Startzustand r_1, r_2, r_3 (verkürzte Schreibweise 1, 2, 3) berücksichtigt, wobei die letzte rechte Stelle, die nach FIFO als nächste auszulagernde Seite enthält. Insgesamt befinden sich nur die drei Seiten r_1, r_2 und r_3 im System, d.h. es wird keine andere Seite angefordert.

b) Geben Sie die Funktion $g_{CLIMB}(r_t, S_{t-1}, q_{t-1})$ für $|S_{t-1}| < 3$ und $|S_{t-1}| = 3$ für einen Speicher mit drei Speicherplätzen und jetzt aber beliebigen Anzahl an Seiten explizit an ($r_t \in \{1, ..., n\}$).

c) Gegeben sei ein gefüllter Speicher mit m Plätzen. Das System unterscheidet n verschiedene Seiten. Wieviele Zustände sind für einen endlichen Automaten vorzusehen, der die Seitenersetzungsstrategie **LIFO** auf diesem (fest vorgegebenen) gefülltem Speicher mit n verschiedenen Seiten beschreibt. Begründen Sie ihre Antwort!

Aufgabe 5: **Der schlafende Barbier** (10 Punkte)

Ein Frisör-Salon besteht aus einem Warteraum mit n Stühlen und einem Arbeitsraum mit dem Frisör-Stuhl. Sobald der Barbier 'frei' ist (d.h. bei Arbeitsbeginn oder nach der Bedienung eines Kunden) geht er in den Warteraum, um den nächsten Kunden in den Arbeitsraum zu holen. Wenn kein Kunde im Laden ist, setzt sich der Barbier auf einen Stuhl und schläft. Trifft ein Kunde den schlafenden Barbier an, so weckt er ihn und wird sofort bedient. Wenn ein Kunde den Salon betritt, während der Barbier arbeitet, belegt er (wenn möglich) einen der n Warteplätze. Sind allerdings alle n Stühle besetzt, geht der Kunde ohne Bedienung wieder nach Hause.

Entwerfen Sie unter Verwendung von *Semaphoren*, die nur die Werte (-1, 0, 1) annehmen können, einen Algorithmus, der den Kundenprozeß mit dem Barbierprozeß koordiniert. Benutzen Sie dabei die informelle Notation aus der Vorlesung und gehen Sie von einem *leeren* System aus. Die Art der Bedienstrategie des Barbiers (FIFO, LIFO, ...) braucht nicht betrachtet zu werden.

Hinweis: Erläutern Sie zuerst die verwendeten Semaphoren und Variablen, und geben Sie deren Initialisierung an.

Aufgabe 6 : **Deadlock Prevention and Avoidance** (5+2 Punkte)

Ein Verfahren der Deadlock-Vermeidung durch eine Betriebsmittelbeschränkung für die einzelnen Prozesse wurde in Übungsaufgabe 22 vorgestellt. Deadlocks können aber auch durch eine Beschränkung der Summe des maximalen Betriebsmittelbedarfs aller Prozesse vermieden werden:

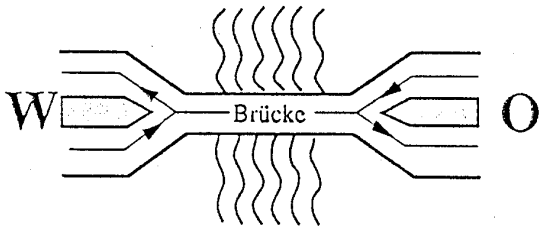
n Prozesse benutzen **m** (gleichartige) Betriebsmittel gemeinsam, von denen zu einem Zeitpunkt jeweils nur eines reserviert bzw. freigegeben werden kann. Der maximale Bedarf jedes einzelnen Prozesses liegt zwischen einem und m Betriebsmitteln.

a) Zeigen Sie, daß eine Verklemmung dieses Systems vermieden werden kann, wenn die Summe der von allen Prozessen maximal benötigten Betriebsmittel auf kleiner als **m+n** beschränkt wird.

b) Zeigen Sie an einem Beispiel, daß eine Begrenzung der Summe der maximal benötigten Betriebsmittel aller Prozesse mit kleiner-gleich eine Verklemmung nicht vollständig ausschließt.

Aufgabe 7: **Das Brückenproblem** (4+4 Punkte)

Wagen, die aus Osten und Westen kommen, müssen eine Brücke über einen Fluß passieren (siehe nachfolgende Abbildung). Unglücklicherweise hat die Brücke nur eine Fahrspur. Sie kann also zu jedem Zeitpunkt nur von einem oder mehreren Wagen aus derselben Richtung benutzt werden.



a) Schreiben Sie mit Hilfe kritischer Regionen jeweils einen Algorithmus für einen Wagen aus dem Osten und dem Westen, wie er an der Brücke ankommt, den Fluß überquert und auf der anderen Seite wegfährt, ohne mit einem entgegenkommenden Wagen zu kollidieren.

b) Verfeinern Sie die Lösung so, daß sich die Fahrtrichtung über die Brücke jedesmal dann ändert, wenn sie 10 Wagen aus einer Richtung passiert haben, während einer oder mehrere Wagen aus der anderen Richtung warten.

Aufgabe 8: **Binder** (5+3 Punkte)

Gegeben seien 2 Unterprogramme A und B, die erst übersetzt und anschließend zu einem ladefähigen Modul zusammengebunden werden sollen. Die für Compiler und Binder relevanten Informationen enthält die folgende Aufstellung:

	Unterprogramm A	Unterprogramm B
Länge (Bytes)	400	350
externe Referenzen *Define Constant* an Adresse	B 350	Paul 230	George 170
externe Namen liegen auf Adresse	George 150	Paul 300
lokale Referenzen *Define Constant* an Adresse	keine	Ringo 174
lokale Namen liegen auf Adresse	keine	Ringo 100

a) Erstellen Sie aus den gegebenen Angaben die Objektmodule für die beiden Unterprogramme A und B,wie sie nach dem Übersetzen und vor dem Binden aussehen.

b) Bestimmen Sie das Output-Load-Modul eines Binders, der das Unterprogramm A vor das Unterprogramm B ablegt.

Lehrstuhl für Informatik IV
Rhein.-Westf. Techn. Hochschule Aachen
Prof. Dr. Otto Spaniol

Diplom-Vorprüfung „Systemprogrammierung"

(WS 1991/92)

Aufgabe 1: **Buddy-Systeme** (3+1+3 Punkte)

Gegeben ist ein Speicherbereich der Größe 2^4 K (K=1024). Folgende Speicheranforderungen sollen der Reihe nach bearbeitet werden:

1) 1 K 2) 1,5 K 3) 3 K 4) 6 K 5) 1 K 6) 2 K

a) Verteilen Sie die Speicheranforderungen mit Hilfe des ***normalen Buddy-Systems*** auf den verfügbaren Speicherbereich. Falls mehrere gleichgroße Buddies zur Verfügung stehen, wird immer der im Baum am weitesten links stehende Buddy zur Aufspaltung bzw. Speicherung gewählt. Geben Sie graphisch nur die Endbelegung des Speichers nach Bearbeitung der letzten Anforderung an. Welcher **interne Verschnitt** ist dabei entstanden?

b) Geben Sie die Anfangsadressen der Buddies, in denen die einzelnen Anforderungen abgelegt sind, im Dualsystem an!

c) Bearbeiten Sie die gleichen Speicheranforderungen mit ***gewichteten Buddies***! Geben Sie graphisch wiederum nur die Endbelegung nach Abarbeitung der letzten Speicheranforderung an.

Aufgabe 2: **Seitenersetzungsstrategien** (4+3 Punkte)

a) Ein Programmsystem verfügt über vier logische Seiten (n = 0,1,2,3), aber nur über drei physikalische Seiten. Bearbeiten Sie den Referenzstring r = 012310201 jeweils mit den Seitenersetzungsstrategien 'Second Chance', 'Climb' und 'OPT' (Für OPT gilt: Der gesamte Referenzstring sei schon zu Beginn bekannt!). Geben Sie dabei in jedem Schritt den sich ergebenden Speicherzustand an. Zu Beginn wird der Speicher jeweils gemäß der FIFO-Strategie gefüllt, bis alle physikalischen Seiten belegt sind.

b) Bei einer anderen Seitenersetzungsstrategie wird in einem Programmsystem (wie oben: vier logische Seiten (n = 0,1,2,3), drei physikalische Seiten) bei einem Seitenfehler (beim Zugriff auf Seite n) jeweils - außer der referierten Seite n - auch die Seite [(n+1) mod 4] und die Seite [(n-1) mod 4] nachgeladen. Zu Beginn seien immer die Seiten 0, 1 und 2 geladen. Geben Sie die Menge der Zustände und der Zustandsübergänge an. Stellen Sie das Verhalten des Systems durch einen endlichen Automaten graphisch dar.

Aufgabe 3: **Crossbar Switch** (3+3 Punkte)

a) Durch einen Fehler zeigen die Crossbar Switches in einem 3-stufigen Delta-Netzwerk ein irreguläres Verhalten. Signale am Eingang A des Crossbar Switches werden normal weitergeleitet. Signale am Eingang B werden jedoch immer an den jeweils falschen Ausgang weitergeleitet. (Also an den unteren Ausgang falls $C_B = 0$ und an den oberen Ausgang falls $C_B = 1$). Wohin gelangt ein Signal am
   i) Eingang 3 des Netzwerks
   ii) Eingang 6 des Netzwerks
   wenn die Kontrollbits auf 111 gesetzt sind?

b) Realisieren Sie mithilfe von 7 defekten Crossbar Switches (vgl. Aufgabenteil a) ein 3-stufiges Netzwerk mit binärbaumartiger Struktur, bei dem zwei Eingänge auf 8 Ausgänge gemultiplext werden können.

Aufgabe 4: **Der Bäckerei-Algorithmus** (8+4 Punkte)

Der Bäckerei-Algorithmus wurde zur Lösung des Wechselseitigen Ausschluss Problems in verteilten Systemen entwickelt. Um das Verhalten des Algorithmus besser studieren zu können, sind in der folgenden Version idealisierte Bearbeitungszeiten für die Einzelschritte angegeben. Die mit einem Label (A - J) gekennzeichneten Zeilen benötigen jeweils genau einen CPU-Zyklus zur Ausführung, während alle anderen Zeilen keine CPU-Zeit verbrauchen:

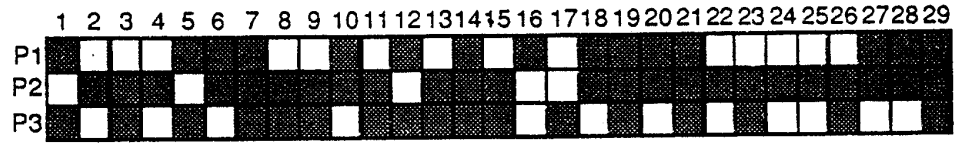
```
var c: array [1 ... n] of boolean  := false
var p, m: array [1 ... n] of integer  := 0

begin
A     c [i] := true;
B     m [i] := max (p [1], p [2], ... , p [n]);
C     p [i] := m [i] + 1;
D     c [i] := false;

      for j := 1 to n do
      begin
E           if c [j] then goto E
F           elseif (p [j] ≠ 0) and ((p [j] < p [i]) or (p [j] = p [i] and (j < i))) then goto F;
            endif;
      end;

G     kritischer Bereich;
H     p [i] := 0;
I     Rest des Programms;
J     goto A;
end
```

a) 3 Prozesse (P1, P2 und P3) bewerben sich um ein exklusiv nutzbares Betriebsmittel, dessen Zugang nach dem Bäckerei-Algorithmus koordiniert wird. Den Prozessen stehen auf beliebigen Prozessoren in einem verteilten System CPU-Zyklen nach dem folgenden Schema zur Verfügung.



■=: Pi steht ein CPU-Zyklus zur Verfügung □=:Pi steht kein CPU-Zyklus zur Verfügung

Bestimmen Sie anhand dieses Schemas die Reihenfolge, in der die Prozesse das Betriebsmittel nutzen dürfen; geben Sie dabei für jeden Zeitpunkt und jeden Prozeß das Label der Anweisung an, die von dem Prozeß aktuell ausgeführt wird. Zu einem Zeitpunkt geänderte Inhalte von Variablen stehen allen Prozessen zum nächsten Zeitpunkt zur Verfügung. Zum Startzeitpunkt stehen alle Prozesse vor Label A.

b) Wie oft kann ein Prozeß, der die Anweisung A ausgeführt hat, in einem System mit n Prozessen maximal überholt werden? Begründen Sie Ihre Antwort!

Aufgabe 5: **Petrinetz** (3+2+2 Punkte)

Gegeben sei ein Petrinetz mit :

S = {Z1, Z2, Z3, run, stop}, T={A, B, C, off, on},
F = {(Z1,A), (run,A), (Z2,B), (run,B), (Z3,C), (run,C), (run,off), (stop,on), (on,run), (off,stop), (C,Z1), (C,run), (A,Z2), (A,run), (B,Z3), (B,run)},
M_0={(Z1,1), (Z2,0), (Z3,0), (run,1), (stop,0)}

Die Kapazität für alle Stellen s ∈ S ist unendlich. Das Kantengewicht ist für alle Kanten f ∈ F gleich eins.

a) Stellen Sie das Netz graphisch dar, und geben Sie alle Folgemarkierungen für M_0 an.
b) Stellen Sie den Erreichbarkeitsgraphen des Netzes graphisch dar.
c) Liefern Sie eine anschauliche Interpretation des Netzes.

Aufgabe 6: **Erzeuger-Verbraucher-Problem** (10 Punkte)

Lösen Sie das *Erzeuger-Verbraucher-Problem* (vgl. Kapitel 1.8.1 im Skript) mit Hilfe von ***binären Semaphoren***. Binäre Semaphore können nur die Werte 0 oder 1 annehmen, und die zugehörigen wait-Aufrufe werden in einer FIFO-Warteschlange verwaltet. Berücksichtigen Sie bei ihrer Lösung folgende zusätzliche Randbedingungen:

- ***n Erzeuger*** und ***ein Verbraucher***.
- Unbeschränkter Puffer zwischen Erzeugern und Verbraucher.
- Es dürfen maximal zwei binäre Semaphore verwendet werden; der Zählsemaphore aus der Vorlesung entfällt.
- Gehen Sie anfangs von einem leeren System aus.

Erläutern Sie zuerst die verwendeten Semaphoren und Variablen, und geben Sie deren Initialisierung an. Geben Sie dann jeweils die Programmsequenz für einen Erzeuger und den Verbraucher zur Lösung des Problems an und kommentieren Sie Ihre ausführlich!

Aufgabe 7: **Bewertung des Paging-Verfahrens** (3+5+4 Punkte)

Insgesamt geht beim Paging-Verfahren neben dem internen Verschnitt auch für die Verwaltung der Seitentabelle Speicherplatz verloren. Wir betrachten nun ein System, in dem mit **p** die verwendete Seitengröße und mit **s** die mittlere Programmgröße - jeweils mit der Anzahl an Speicherworten - angegeben ist. Dabei gilt $p << s$; weiterhin soll jeder Eintrag in die Seitentabelle mit einem Speicherwort berücksichtigt werden.

a) Geben Sie eine Funktion an, mit der der **mittlere Wert** f des beschriebenen Gesamtverschnitts (in Speicherworten) bei der Verwaltung typischer Programme in Abhängigkeit von p und s berechnet werden kann. Auf die Ganzzahligkeit für f kann hierbei verzichtet werden.

b) Stellen Sie das **prozentuale Verhältnis** F des in a) berechneten Gesamtverschnitts f zur mittleren Programmgröße s für Seitengrößen p = 32 bzw. 1024 Worte und für s = 1024, 2*1024, 8*1024, 16* 1024 und 32*1024 Worte graphisch dar. Interpretieren Sie kurz den Kurvenverlauf.

c) Zeigen Sie, daß für das **Minimum** F_0 des in b) dargestellten Verhältniss des Speicherverlustes f zur mittleren Programmgröße s bei festem s gilt: $F_0 \approx 100 * \sqrt{\frac{2}{s}}$.

Aufgabe 8: **Realisation eines Monitors in der Programmiersprache Ada** (5+7 Punkte) Ge.

Das Monitorkonzept zur Koordination des Zugriffs paralleler Prozesse auf ein gemeinsam genutztes Betriebsmittel läßt sich in der Programmiersprache Ada sehr leicht mit einer Monitor-Task realisieren.

a) In einem Prozeßsystem sei der Zugriff auf ein gemeinsam genutztes Betriebsmittel mit drei Prozeduren A, B und C möglich. Geben Sie den **grundsätzlichen Aufbau** des task bodys einer Monitor-Task in der Sprache Ada an, die eine Zugriffskontrolle für das gemeinsame Betriebsmittel so realisiert, daß der Zugriff jeweils exklusiv erfolgt.

b) In einem Prozeßsystem greifen beliebig viele parallele Prozesse auf eine gemeinsame Integervariable zu. Neben Inkrementieren und Dekrementieren der Variablen soll die Ausgabe des aktuellen Wertes möglich sein. Liegt der ausgelesene Wert der Variablen außerhalb des Intervalls [-n,n] soll dem Prozeß, der sich den Wert der Variablen aktuell geholt hat, innerhalb von 5 sec. ein Zurücksetzen der Variablen auf Null möglich sein, ohne das ein anderer Prozeß auf die Variable zugreifen kann. Realisieren Sie unter Verwendung des in a) entworfenen Schemas einen task body einer Monitor-Task, die die möglichen Funktionsaufrufe der verschiedenen Prozesse so koordiniert, daß der Zugriff auf die Variable exklusiv erfolgt, und deren Wert immer konsistent bleibt. Vor dem ersten Funktionsaufruf soll die Variable mit Null initialisiert werden.

Lehrstuhl für Informatik IV
Rhein.-Westf. Techn. Hochschule Aachen
Prof. Dr. Otto Spaniol

Diplom-Vorprüfung „Systemprogrammierung"

(SS 1992)

Aufgabe 1: **Bankers-Algorithmus** (5+2 Punkte)

Gegeben seien drei Prozesse P_1, P_2 und P_3, die sich die Betriebsmittel BM_1, BM_2 und BM_3 teilen. Es stehen 10 BM vom Typ1 und jeweils 2 BM von Typ 2 und 3 zur Verfügung, die nur exklusiv nutzbar sind. Von folgender Habenbelegung ist zum Zeitpunkt k für die Anfragen i) - iii) auszugehen:

$$H_1(k) := (2, 0, 1),\ H_2(k) := (3, 1, 1),\ H_3(k) := (1, 0, 0)$$

a) Führen Sie den Bankers-Algorithmus für folgende Anfragen durch:

i) $Q_1(k) := (3, 1, 0),\ Q_2(k) := (2, 1, 0),\ Q_3(k) := (0, 1, 1)$
ii) $Q_1(k) := (3, 1, 0),\ Q_2(k) := (7, 1, 1),\ Q_3(k) := (0, 2, 1)$
iii) $Q_1(k) := (3, 2, 0),\ Q_2(k) := (2, 1, 0),\ Q_3(k) := (0, 1, 1)$

Geben Sie v(k) und jeweils die schrittweisen Änderungen von i, j, W und δ_i an.

b) Falls sich in den Anfragen i) - iii) eine Teilmenge D von Prozessen in einem Deadlock befindet, prüfen Sie dieses formal nach.

Aufgabe 2 : **Adressenberechnung beim Paging-Verfahren** (3+5 Punkte)

Ein virtueller Speicher soll mittels Paging-Technik realisiert werden. Der Hauptspeicher ist dabei in 8 Frames aufgeteilt, die jeweils eine Page von 1024 Speicherworten aufnehmen können. Nun soll ein Programm mit einem logischen Adressraum von 10000 Worten zur Ausführung gebracht werden. Während der Ausführung des Programms ist der Programmcode im logischen Speicher ab der ersten Position von Page 0 gespeichert. Diese Page ist im 3. Frame des Hauptspeichers abgelegt. Zum Startzeitpunkt belegt das Programm außerdem noch zwei weitere Frames: Page 3 ist in Frame 7 abgelegt und Page 5 in Frame 0. Die restlichen Seiten des Programms sind aktuell nicht geladen.

a) Skizzieren Sie für diese Situation den Aufbau des Page Tables, und geben Sie dessen Einträge an. Berechnen Sie dann mit diesem Page Table für folgende logische Adressen (p,d) die physikalischen Adressen :

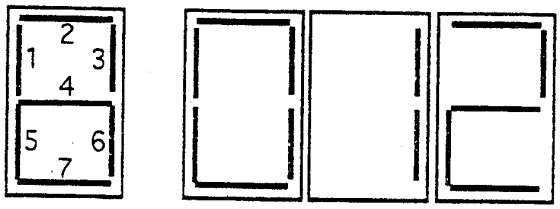
(3,936) (5,24) (3,1420) (2,30)

b) Auch während seiner Ausführung darf das Programm nur die drei oben angegebenen Frames 0, 3 und 7 belegen, womit bei Zugriffen auf nicht geladene Seiten ein Seitenaustausch erforderlich wird. Bestimmen Sie die Einträge in der Page Table, wie sie während der Abarbeitung der unten angegebenen Befehlsequenz jeweils nach einer Befehlsausführung entstehen, wenn zur Seitenersetzung die LRU Strategie angewendet wird. Berücksichtigen Sie dabei sowohl Daten- als auch Befehlszugriffe, wobei zur Vereinfachung angenommen werden kann, daß ein Befehl jeweils aus einem Speicherwort besteht, und so mit nur einem Zugriff geladen werden kann.

0: Hole von log. Adresse 3263
1: Lege in log. Adresse 6144 ab
2: Lege in log. Adresse 6145 ab
3: Lese von der I/O-Einheit
4: Verzweige nach 4 wenn I/O-Einheit belegt
5: Hole von log. Adresse 3263
6: Lege in log. Adresse 6144 ab
7: Halt.

Aufgabe 3: **Petrinetze** (4+3 Punkte)

Eine 7-Segmentanzeige stellt mit sieben stabförmigen Lichtquellen die Ziffern 0 bis , wie folgt dar:



Das Verhalten eines Modulo-3 Zählers, der seinen Zustand über diese 7-Segmentanzeige ausgibt, soll nun durch ein Stellen/Transitionsnetz Y = (S, T, F, K, W, M_0) dargestellt werden (s. Anhang 1 der Hilfsblätter). Die jeder Stelle zugeordnete Kapazität sei unendlich und das jeder Kante zugeordnete Gewicht sei immer eins.

Eine Lichtquelle x soll dabei immer durch zwei Stellen s_x und $s_{/x/}$ dargestellt werden. Eine Marke in einer Stelle s_x besagt, daß die entsprechende Lichtquelle x leuchtet. Eine Marke in einer Stelle $s_{/x/}$ besagt, daß die entsprechende Lichtquelle x nicht leuchtet. Die Menge S ist also gegeben durch :

$$S = \{ s_1, s_{/1/}, s_2, s_{/2/}, s_3, s_{/3/}, s_4, s_{/4/}, s_5, s_{/5/}, s_6, s_{/6/}, s_7, s_{/7/} \}$$

a) Geben Sie M_0 so an, daß damit der Zählerzustand 0 repräsentiert wird, und geben Sie die Mengen T und F so an, daß damit das Verhalten des Zählers nachgebildet wird.

b) Geben Sie alle aus dem Startzustand erreichbaren Folgemarkierungen und den Erreichbarkeitsgraphen des Netzes an (Hinweis : Folgemarkierungen müssen nicht formal vollständig angegeben werden. Andeuten reicht !).

Aufgabe 4: **Seitenersetzungstrategie Climb** (2+2+4 Punkte)

Die Seitenersetzungsstrategie Climb in einem Rechnersystem mit zwei physikalischen und vier logischen Seiten soll durch einen endlichen Automaten A=(I, Z, T, z_0) dargestellt werden.

a) Geben Sie das Eingangsalphabet I und die Zustandsmenge Z an.

b) Wie lautet die Menge der Zustandsübergänge T ?

c) Geben Sie einen möglichen Anfangszustand z_0 und einen Referenzstring ω so an, daß jede Seite zweimal nachgeladen werden muß. Wieviele Elemente aus I muß der Referenzstring dazu mindestens enthalten ?

Aufgabe 5: **Hash Codierung** (3+3+3 Punkte)

In einem zweidimensionalen Feld mit 10 Speicherplätzen (0 - 9) sollen Tupel der Art (*Name einer Person, Telefonnummer*) mit dem Hash-Verfahren abgespeichert werden. Als Schlüssel für die Abbildung auf eine Speicherposition wird die letzte Ziffer der Telefonnummer verwendet. Im Falle eines Overflows wird der Schlüssel solange um eins erhöht, bis eine freie Position gefunden wird.

a) Speichern Sie die folgenden Datensätze in der Reihenfolge von oben nach unten in dem Feld ab, und geben Sie die endgültige Belegung des Feldes an.

Waigel	7411
Spaniol	8733
Brandt	3631
Schmidt	6492
Rupprecht	6184
Kohl	0000
von Sinnen	4325
Meuser	7597
Walkes	5725

b) Wie oft muß in dieser Belegung jeweils auf den Speicher zugegriffen werden, um zu einer Telefonnummer die zugehörige Person zu ermitteln? Wieviele Zugriffe werden im Mittel benötigt?

c) Wie oft müßte jeweils zugegrifen werden, wenn die Daten sequentiell in der Reihenfolge ihrer Eingabe abgespeichert worden wären, und dementsprechend linear gesucht werden müßten ?

Aufgabe 6: **Prozeß-Synchronisation mit Semaphoren** (7 Punkte)

Gegeben sei ein Kommunikationssystem mit einem Sender und einem Empfänger. Sobald der Empfänger seine Bereitschaft signalisiert, sendet der Sender eine Nachricht. Bevor der Sender dann die nächste Nachricht senden darf, muß der Empfänger erst eine Quittung für die vorherige Nachricht schicken. Nachdem dem der Erhalt dieser Quittung wiederum vom Sender bestätigt worden ist, kann dieser die nächste Übertragung starten.

Schreiben Sie mit Hilfe von Semaphoren sowohl für den Sende- als auch für den Empfangsprozeß einen Algorithmus, der beide Prozesse synchronisiert und dabei auch mögliche Verklemmungen verhindert. Wie müssen die Semaphore initialisiert werden, damit vermieden wird, daß der Sender Nachrichten sendet, bevor der Empfänger dazu bereit ist?

Aufgabe 7: **Stackalgorithmen** (4+6 Punkte)

A sei ein Demand-Paging Algorithmus, $\omega = (r_1, \ldots, r_T)$ ein abzuarbeitender Referenz-String und **m** die Größe des Speichers $S(m) = \{s_1, \ldots, s_m\}$. Dann bezeichnet $R(r_t, S(m,\omega), q_{t-1})$ die Menge der Seiten, die bei gegebenen Speicherzustand q_{t-1} und Referenz auf die Seite r_t ausgelagert werden. Das Austauschverhalten eines Stack-Algorithmus kann damit wie folgt beschrieben werden:

Ist A ein Stack-Algorithmus ⇒

$$\left[\text{falls } r_t \notin S(m,\omega) = \{s_1(\omega), \ldots, s_m(\omega)\}, \text{ dann } R(r_t, S(m,\omega), q_{t-1}) = \left\{ \begin{array}{c} R(r_t, S(m-1,\omega), q_{t-1}) \\ \text{oder} \\ s_m(\omega) \end{array} \right\}\right].$$

a) Zeigen Sie mit dieser Beziehung, daß FIFO kein Stack-Algorithmus ist. Benutzen Sie dabei eine Speichergröße m=4 und führen Sie den geforderten Nachweis nach der Abarbeitung des Referenz-Strings $\omega = (1, 2, 3, 4, 1, 2)$, also zum Zeitpunkt $t = 7$.

b) Beweisen Sie formal die oben angegebene Formel zur Beschreibung des Austauschverhaltens von Stack-Algorithmen.