FACH Hochschule Aachen

BSK/ARBK SS-04 Oß

(Fachbereich 5, Elektrotechnik und Informationstechnik)

Zuname:	Vorname:	Matr.-Nr.:

Klausur "Betriebssystemkonzepte" sowie "Architektur von Rechnersystemen und" "Betriebssystemkonzepte"

Fach-Nummer:	5552(BSK) / 5310(ARBK) Fachprüfung
Termin:	Donnerstag, 15. Juli 2004, 13.30-16.30 Uhr
Prüfer:	Prof. Dr.-Ing. Martin Oßmann

Hinweise:

Bearbeitungszeit:	180 Minuten (3 Zeitstunden)
Hilfsmittel:	Taschenrechner mit einzeiligem Display ohne Datenbankfunktion
	Ein zweiseitig selbstbeschriebenes A-4 Blatt
Vollständige Klausur:	6 Aufgaben

Die Aufgaben direkt nur auf den Aufgabenblättern oder Leerbögen bearbeiten. Lösungen ohne erkennbaren und leserlichen Lösungsweg sind wertlos. Jedes Blatt mit Name und Matrikelnummer versehen ! Deckblatt unterschreiben !

Zur Kenntnis genommen:

_____

Unterschrift

Korrekur:

1	2	3	4	5	6	Summe	Note

(Max. 100 Punkte)

Aufg. 1 (20 Punkte)

Eine Festplatte verfügt über 20 Oberflächen und 4096 Zylinder. Die Clustergrösse ist 64k Byte. Es wird FAT32 (d.h. 32-Bit große FAT-Einträge) verwendet und eine FAT belegt genau 50 Cluster.

a) Was ist die genaue Größe der Speicherkapazität der Festplatte in k-Bytes (1k Byte=1024 Byte)? Wieviele Cluster hat die Festplatte genau ?

b) Im Hauptverzeichnis werden 64 Bytes pro Eintrag reserviert. Es können maximal 500000 Dateien im Hauptverzeichnis stehen. Wieviele Cluster belegt das Hauptverzeichnis.

c) Wieviele Cluster bleiben zur Datenspeicherung frei, wenn 16 Boot-Cluster, 2 FATs und ein Hauptverzeichnis angelegt werden ? Wieviel Prozent der Festplattenkapazität sind also Overhead ?

d) Es wird nun auf die vorher leere Platte eine 200000 Byte große Datei mit Namen aaa.dat und danach eine 10 Byte große Datei mit Namen bbb.dat geschrieben. Freie Cluster werden jeweils vom Beginn der FAT aus gesucht. Skizzieren Sie, wie nach Schreiben dieser zwei Dateien die FAT aussieht:

Adresse	Inhalt
0	reserviert
1	reserviert
2	
3	
4	
5	
6	
7	
8	
9	
10	

FAT zu d

Adresse	Inhalt
0	reserviert
1	reserviert
2	
3	
4	
5	
6	
7	
8	
9	
10	

FAT zu e

Adresse	Inhalt
0	reserviert
1	reserviert
2	
3	
4	
5	
6	
7	
8	
9	
10	

Fat zu f

e) Danach wird die Datei aaa.dat gelöscht und eine 80000 Bytes große Datei ccc.dat geschrieben. Wie sieht nun die FAT aus ?

f) Danach wird eine 200000 Bytes große Datei ddd.dat geschrieben. Wie sieht nun die FAT aus ?

Zuname:	Vorname:	Matr.-Nr.:

Es liest jeweils nur ein Kopf auf einer Oberfläche. Die Festplatte arbeite mit einer Umdrehungsrate von 3000 rpm. Ein Spurwechsel dauert 30ms.

g) Wieviele Bits hat eine Spur ? Welche Datenrate (Bit pro Sekunde) liefert der Lesekopf maximal ?

h) Wie lange dauert es im ungünstigsten Fall, eine 50k grosse Datei zu lesen ? Geben Sie genau an, wie sie zu Ihrem Ergebnis gelangen !

i) Vom Betriebssystem kann man sich von einer Festplatte den belegten Platz (=Summe der Bytes in Dateien) und den freien Platz (=Summe der Bytes in freien Clustern) anzeigen lassen. Auf der Festplatte sind 400000 Dateien gespeichert, die jeweils 10k Gross sind. Wieviele Bytes sind noch zur Speicherung anderer Daten auf der Platte verfügbar ? Wie gross ist die Summe aus freiem und belegtem Platz ?

j) Ein Betriebssystem bekommt den Namen einer Datei auf einer Festplatte. Wie ermittelt das Betriebssystem, wo der erste Sektor der Datei steht ?

Aufg. 2 (20 Punkte)

Ein Rechner verwaltet seinen Speicher mittels einstufigem Demand Paging. Er besitzt eine Memory Management Unit MMU. welche den Speicherzugriff durchführt. Alle dazu benötigten Hilfsmittel werden auf der MMU implementiert. Der Speicher ist in Bytes organisiert. Eine Seite ist 64k-Bytes groß und eine Adresse des virtuellen Adressraums ist 32 Bit groß. Der physikalische Speicher ist 512k-Bytes groß.

a) Wieviele Einträge hat die Seitentabelle ? Wieviele Seitenrahmen passen in den Speicher ?

b) Geben Sie genau an, was in einem Eintrag der Seitentabelle gespeichert wird. Geben sie (z.B. in Form einer Skizze) an, wie die phys. Adresse aus einer log. Adresse bestimmt wird !

Am Anfang sind keine Seiten im physikalischen Seicher eingelagert, und die MMU ist entsprechend initialisiert. Es erfolgen dann lesende Speicherzugriffe auf die Speicherstellen mit folgenden Adressen:

Adresse
00035673H
00027777H
00027785H
00036789H
0002789AH

c) Skizzieren Sie nun den Inhalt der Seitentabelle und die Belegung des physikalischen Speichers nach diesen Zugriffen. Wieviele gültige Seitenrahmen befinden sich im Hauptspeicher ? Geben Sie zu jeder der oben angegebenen Adresse die zugehörige physikalische Adresse an !

d) Wieviele Seiten umfasst normalerweise (geschätzt) das "Working Set" eines Prozesses, der im M32 Assembler-Praktikum programmiert wurde. Geben Sie an, wie Sie zu der Grösse gelangen.

e) Woran merkt ein Betriebssystem, dass es einem Prozess zu wenig Seiten allokiert hat ?

f) Warum verwendet man heutzutage häufig mehrstufiges Paging ?

g) Ein 256M-Byte grosser Speicher wird mit dem Buddy-System verwaltet. Die kleinste verwaltbare Segmentgröße ist 2M-Byte. Für belegte und freie Segmente wird jeweils eine getrennte Liste verwendet.

g-i) Wieviele Listen sind zu verwalten ?

g-ii) Wieviele Elemente sind maximal in einer einzelnen Liste ?

g-iii) Beschreiben Sie im Detail, wie eine Anforderung nach einem 17M-Byte Segment vom Betriebssystem bearbeitet wird !

g-iv) Beschreiben Sie im Deteil, was passiert, wenn ein Segment von einem Prozess ans Betriebssystem zurückgegeben wird !

Aufg. 3 (15 Punkte)

Ein Betriebssystem vergibt die Rechenzeit an Prozesse mittels Warteschlangen. Das Scheduling wird prioritätsbasiert durchgeführt. Immer wenn ein neuer Prozess eintrifft, oder ein Prozess fertig wird, wird der aktuell rechnende Prozess unterbrochen und er kommt wieder in die Warteschlange. Dann wird der Prozess aus der Warteschlange gescheduled, der die höchste Priorität hat. Haben mehrere Prozesse die höchste Priorität, wird derjenige gescheduled, der am längsten nicht gerechnet hat. Die niedrigste Priorität ist Priorität=1.

Es sind die folgenden Prozesse zu bearbeiten !

Prozess	Ankunft/s	Rechenzeit/s	Prorität
A	0	3	1
B	1	2	3
C	2	1	2
D	5	4	3
E	6	1	2
F	12	1	1

a) Zeichen Sie das Gantt-Chart für die Prozessorvergabe. Geben Sie zu den Zeitpunkten, in welchen eine Scheduling-Entscheidung getroffen wird, jeweils den Inhalt der Warteschlangen an, und welcher Prozess jeweils als nächster den Prozessor bekommt. Der vor der Entscheidung rechnende Prozess soll in der Warteschlange mit dargestellt werden.

b) Geben Sie die Wartezeit und die Turn-Around Zeit für die Prozesse A,B und C an.

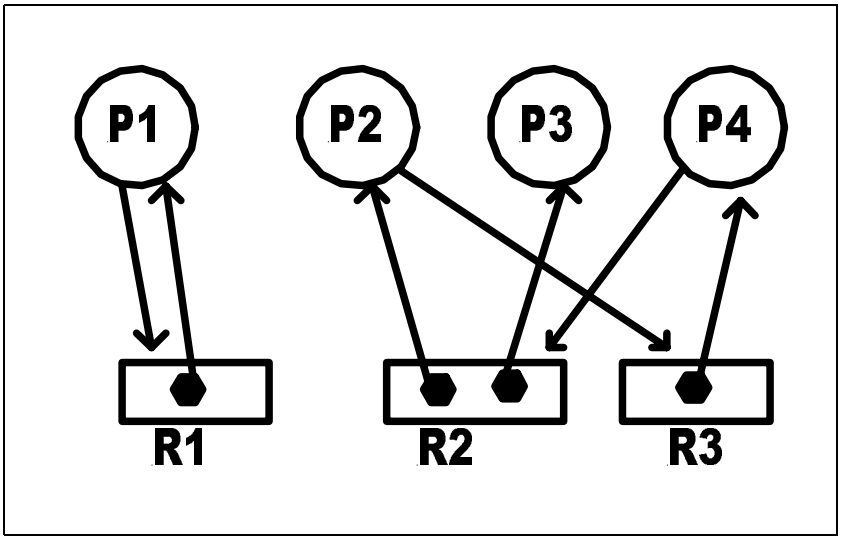
c) Wird bei diesem Scheduling jeder bedient (Begründung) ?

d) Wie hoch ist die Wartezeit eines Prozesses bei reinem Round-Robin Scheduling maximal und minimal ?

e) Warum wird reines Round-Robin Scheduling in der Praxis selten angewandt ?

Aufg. 4 (15 Punkte)

Gegeben ist der folgende Betriebsmittelgraph:



a) Man stelle die zu dieser Situation gehörige Anforderungsmatrix $R = (r_{ij})$ und Belegungsmatix $C = (c_{ij})$ sowie den Betriebsmittelrestvektor A und den Betriebsmittelvektor E auf !

b) Liegt ein Deadlock vor ? Begründen Sie Ihre Antwort mit Hilfe des Algorithmus zur Deadlockerkennung ! Nach jedem Schritt des Algorithmus sind die neuen Matrizen und Vektoren anzugeben ! Welche Prozesse sind am Deadlock beteiligt ?

c) Wieviele und welche Zyklen enthält der Betriebsmittelgraph aus a) ? Geben Sie für jeden Zyklus die Prozesse/Betriebsmittelklassen in der Durchlaufungsreihenfolge an.

d) Auf welche Weisen kann man den Deadlock aus b) beheben ? Bei Ihrer Lösung müssen irgendwann alle Prozesse erfolgreich enden !

e) Nennen Sie Beispiele für Betriebsmittel, die man eventuell temporär entziehen kann. Wie wird jeweils sichergestellt, dass der Prozess, dem ein Betriebsmittel entzogen wird, keinen schweren Schaden erleidet ?

f) Wann kann die Zyklensuche zur Deadlockerkennung angewandt werden ?

Aufg. 5 (15 Punkte)

Ein Betriebssystem wird in einer C-ähnlichen Programmiersprache programmiert. Bereits implementiert sind die Routinen $DISABLE()$ und $ENABLE()$ um Interrupts zu sperren bzw. freizugeben.

a) Erklären Sie im Detail, was eine TSL-Funktion macht.

b) Programmieren Sie in einer C-ähnlichen Sprache die TSL Funktion !

c) Der Eintritt in folgenden Block soll durch wechselseitigen Ausschluss unter Verwendung der TSL Funktion aus b) mit Busy-Waiting geschützt werden. Programmieren Sie das in der C-ähnlichen Programmiersprache einschließlich Initialisierung !

```
 /* Beginn kritischer Bereich */
A=A+B ;
C=C-B ;
 /* Ende kritischer Bereich */
```

d) Warum wird Busy-Waiting in der Praxis kaum verwendet ? Was ist die Alternative ? Warum ist diese besser ?

e) In einem Betriebssystem können sich Prozesse in einem der folgenden Zustände befinden: "READY", "WAITING", "RUNNING". Das Scheduling wird interruptgesteuert (z.B. mit Round-Robin) durchgeführt, die zugehörigen Ereignisse werde mit "INTERRUPT" und "SCHEDULED" bezeichnet. Prozesse können die Semaphor-Operationen "UP" und "DOWN" ausführen. Diese sind auch als Ereignisse aufzufassen. Stellen Sie die möglichen Zustände mit den Übergängen in einem Diagramm dar !

Zuname:	Vorname:	Matr.-Nr.:

Aufg. 6 (15 Punkte)

Gegeben ist das folgende M32 Assemblerprogramm:

```
LINE   RAM        CODE                              TEXT

 01 00000000H                                        ORG  0

 02 00000000H  21002002H 00000002H                   MOV  R2,2

 03 00000002H  21004203H ________H                   MOV R3,5(R2)

 04 00000004H  21002222H 0000000FH ________H         MOV @HERE,@STOP

 05 00000007H  31000020H ________H                   JMP  LOOP

 06 00000009H  21002000H ________H                   MOV  R0,5

 07 0000000BH  11002000H 0000000FH           LOOP ADD  R0,0FH

 08 ________H  12002002H 00000001H                   SUB  R2,1

 09 ________H  01000000H                     HERE NOP

 10 ________H  32000020H ________H                   JNZ  LOOP

 11 ________H  F2000000H                     STOP HALT

 12 ________H                                        DS   3

 13 ________H  00000041H ________H ________H XYZ  DW   "ABC"

 14 ________H                                ABC  END


Leerstellen (jeweils ein 32 Bit-Wort) mit ________H markiert !!
```

a) In der Spalte RAM steht normalerweise die Adresse, unter welcher der Assembler Code und Daten abspeichert. Ergänzen Sie alle Leerstellen dieser Spalte.

b) Ergänzen Sie ebenfalls die in den Spalten CODE vorhandenen Leerstellen !

c) Geben Sie den Inhalt der Symboltabelle nach dem Assemblerlauf an !

d) Geben Sie den Inhalt der Register R0 bis R4 und des Program-Counters PC nach dem Programmlauf (gestartet durch RESET) an.

e) Das Programm wird assembliert und dann (nach RESET) gestartet. Geben Sie für die ersten 10 Speicherzugriffe jeweils die Adresse und die gelesenen bzw. geschriebenen Daten (hexadezimal) an !

f) Welche Register ändern sich bei Ausführung des Befehls:

```
CALL 100H
```