2. Test aus „Einführung in die Technische Informatik“ 22.01.05	
Name	
Vorname	
Matrikelnummer	
Kennzahl	

- Bitte deutlich schreiben, unleserliche Antworten werden nicht gewertet !
- Namen und Vornamen in die richtigen Felder in Druckschrift eintragen !

Arbeitzeit 90 Minuten

Bsp	Pnkt.	
1	1	
2	2	
3	2	
4	2	
5	3	
6	3	
7	1	
8	2	
9	1	
10	2	

Bsp	Pnkt.	
11	2	
12	1	
13	3	
14	4	
15	4	
16	6	
17	5	
18	2	
19	4	
Σ	**50**	

1. Was versteht man unter dem Scratchpad eines Prozessors? (1 Punkt)

2. Das in der Informatik bekannte „Moore's Law“ (benannt nach Gordon Moore, einem der Gründer von Intel) sagt voraus, dass die Dichte der Transistoren auf einem integrierten Schaltkreis exponentiell wächst. Obwohl diese Vorhersage schon 1965 getätigt worden ist, trifft sie immer noch zu. Welchen Vorteil hat eine Erhöhung der Bauteildichte bei Mikroprozessoren? (2 Punkte)

3. Das Instruktionsformat der x86 Prozessorreihe von Intel (der auch die neusten Pentium Prozessoren angehören) verwendet Maschineninstruktionen mit variabler Länge. Eine x86 Maschineninstruktion kann zwischen einem und 15 Bytes lang sein. Nennen Sie einen Vorteil und einen Nachteil von Instruktionen mit variabler Länge im Vergleich zu Instruktionen mit fixer Länge. (2 Punkte)

4. Welche zwei Methoden gibt es, die Adressen der Ports für Input/Output-Operationen zu vergeben? (2 Punkte)

5. Was ist ein Trap? Geben Sie ein Beispiel. (3 Punkte)

6. Geben Sie zumindest 3 verschiedene Segmente an, die sich in einem ausführbaren Programm befinden? (3 Punkte)

7. Wo werden die Return Adressen bei Unterprozeduraufrufen gespeichert? (1 Punkt)

8. Warum ist eine hohe Trefferquote der *branch prediction* Einheit (verantwortlich für die Vorhersage des Ziels von bedingten Sprüngen – predicted branches) bei modernen Prozessoren so wichtig? Denken Sie daran, dass diese Prozessoren eine sehr lange Pipeline (oft mehr als 20 Stufen) verwenden. (2 Punkte)

9. Was versteht man unter *logischer Parallelität* bei Prozessen? (1 Punkt)

10. Nennen Sie die Hauptaufgabe des Layer 3 (Network Layer) im OSI Referenzmodell? Geben Sie das vorherrschende Network Layer Protokoll im Internet an. (2 Punkte)

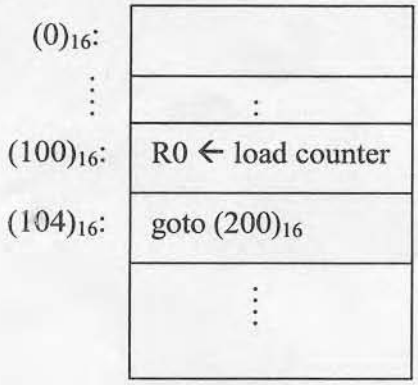
11. Nennen Sie einen Nachteil von asynchronen Methoden zur Interprozess-Kommunikation (IPC) im Vergleich zu synchronen Methoden? Geben Sie ein Beispiel für asynchrone IPC Methoden, die unter Unix und auch in MS Windows realisiert sind. (2 Punkte)

12. Wie groß ist der Adressraum, der mit 16-bit Adressen adressiert werden kann? (1 Punkt)

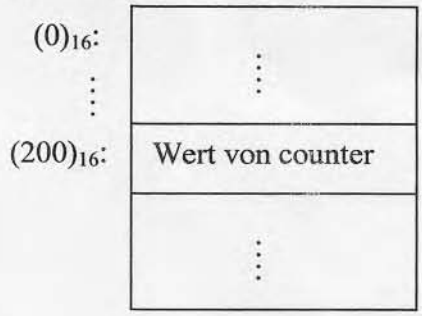
13. Wenn der Parent-Prozess in einem Unix Betriebssystem auf das Terminieren seines Child-Prozesses wartet (mittels des `wait()` system calls), dann wird der Prozess solange blockiert, bis der Child-Prozess einen `exit()` system call macht. Dabei wird das Ergebnis des Child-Prozesses (sein Rückgabewert) an den wartenden Parent-Prozess übermittelt. Was passiert nun mit einem Child-Prozess, der terminiert und sein Ergebnis an den Parent-Prozess übermitteln will, wenn der Parent-Prozess nicht vorher ein `wait()` aufgerufen hat? (3 Punkte)

14. Ein Linker soll ein Codemodul und ein Datenmodul zu einem ausführbaren Program zusammenfügen. Sowohl das Codemodul als auch das Datenmodul liegen als Relocatable Object Code vor. Das Codemodul soll im ausführen Programm an der virtuellen Adresse 0x08048000 beginnen. Das Datenmodul soll an der virtuellen Adresse 0x40000000 liegen.

Im Folgenden sind Ausschnitte aus dem Codemodul und dem Datenmodul gegeben.



Codemodul



Datenmodul

Geben Sie an, an welchen virtuellen Adressen (a) der **load** Befehl, (b) der **goto** Befehl und (c) der Wert der Variable **counter** nach dem Linken (im ausführbaren Programm) gespeichert sind.

Geben Sie außerdem an, (e) zu welcher virtuellen Zieladresse der **goto** Befehl springt und (f) zu welcher Adresse die *unresolved external address* des **load** Befehl aufgelöst wird. (4 Punkte)

(a)
(b)
(c)
(d)
(e)

15. Der im Buch beschriebene Prozessor Mikro16 enthält leider keinen eigenen Befehl, um einen beliebigen numerischen Wert (das heisst, ein immediate value) in ein Register zu laden. Geben Sie eine Sequenz von Instruktionen für den Mikro16 an, der den Wert 17 in das Register R11 lädt. Sie können dazu beliebige, andere Register überschreiben. Bedenken Sie jedoch, dass die Register R0, R1 und R2 nicht beschrieben werden können, weil sie die Konstanten 0, 1 beziehungsweise -1 beinhalten. (4 Punkte)

16. Ein Speicherverwaltungssystem mit Paging verwendet 32-bit virtuelle Adressen und 24-bit physikalische Adressen. Die Größe einer Page beträgt 8 KB (8192 Bytes). Gegeben ist eine Pagetable. Wandeln Sie mit Hilfe dieser Table die folgenden 4 virtuellen 32-bit Adressen in die entsprechenden physikalischen Adressen um. Geben Sie auch an, ob ein *page fault* auftritt. (6 Punkte)

Virtuelle Adressen

1: $(000078BF)_{16}$ 2:$(001DD1AB)_{16}$

3: $(001DA472)_{16}$ 4:$(00004A02)_{16}$

Page-Nummer	Frame-Nummer
$(1)_{16}$	(not present)
$(2)_{16}$	(not present)
$(3)_{16}$	$(1B)_{16}$
...	
$(E9)_{16}$	(not present)
$(EA)_{16}$	$(207)_{16}$
$(EB)_{16}$	(not present)
$(EC)_{16}$	$(7C)_{16}$
$(ED)_{16}$	(not present)
$(EE)_{16}$	$(380)_{16}$

1:

2:

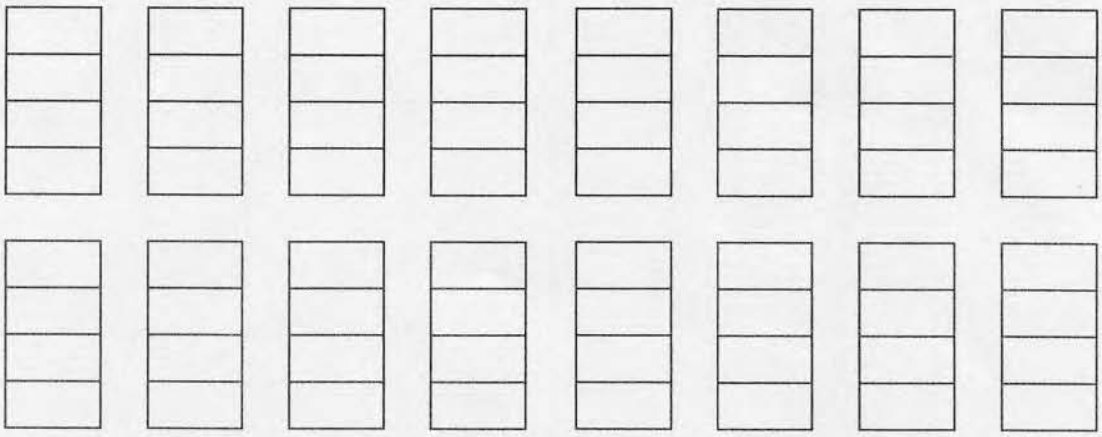
3:

4:

17. Ein Betriebssystem verwendet den LFU (least frequently used) Algorithmus für das Page Replacement. Wenn zwischen Pages entschieden werden muss, die gleich oft verwendet worden sind, wird die älteste Page aus dem Speicher entfernt (jene Page, die zuerst geladen worden ist).

Einem Prozess stehen 4 physikalische Frames zur Verfügung. Im Folgenden ist die Reihenfolge der Page Nummern angegeben, auf die dieser Prozess zugreift (Referenzstring). Geben Sie nach jedem Zugriff an, welche Pages sich im physikalischen Speicher befinden. (5 Punkte)

Referenzstring: 1 2 1 3 4 2 5 5 6 7 8 5 8 9 9 10



18. Der Stackpointer (SP) zeigt bei der Ausführung eines Programms auf die Adresse 0xFFA8. Das Programm führt daraufhin zwei push Befehle durch, die jeweils ein 16-bit Wort auf den Stack legen. Auf welche Adresse zeigt der Stackpointer nach diesen Operationen? (2 Punkte)

19. Gegeben ist ein Programm P, welches von zwei Threads A und B ausgeführt wird. Das Programm ist in der linken Box angegeben.

Programm P:

```
 1: int j;
 2:
 3: void f(int i)
 4: {
 5:    int k;
 6:    k = i + j;
 7:    j = k;
 8:    return;
 9: {
10:
11: int main()
12: {
13:    int i;
14:
15:    i = read_int();
16:    j = read_int();
17:    f(i);
18:    return 0;
19: }
```

Ausführungsreihenfolge:

```
Thread A:
15 (read_int() liefert 1)
16 (read_int() liefert 42)

Thread B:
15 (read_int() liefert 2)
16 (read_int() liefert 10)
17
6

Thread A:
17
6

Thread B:
7
8
18
ENDE (Wert von j angeben)

Thread A:
7
8
18
ENDE (Wert von j angeben)
```

Die Ausführungsreihenfolge der Threads ist in der rechten Box angegeben. Es sind dabei jeweils die Zeilennummern der Instruktionen angegeben, die ein Thread ausführt. Es beginnt Thread A mit der Instruktion auf Zeile 15. Dabei ist für jeden Aufruf der Funktion `read_int()` angegeben, welchen Wert die Funktion zurückliefert. Nachdem Thread A zwei Instruktionen ausgeführt hat, schaltet der Scheduler zu Thread B um, der dann vier Instruktionen ausführt, usw.

Geben Sie an, welchen Wert die Variable `j` hat, (a) nachdem Thread B die Ausführung beendet hat und (b) nachdem auch Thread A die Ausführung beendet hat. (4 Punkte)

(a)
(b)