Einführung in die Technische Informatik

Prüfungsordner zum 2. Test
26. Juni 2004
© Paul Staroch
Datum: 25. Juni 2005
Erstellt mit LaTeX

Punkteverteilung (insgesamt 50 Punkte):

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
3	3	1	1	1	2	2	2	2	1	6	4	2	8	6	6

Für Korrektheit und Vollständigkeit der Daten kann wie immer keine Garantie abgegeben werden.

---

1. Nennen Sie zumindest drei Bestandteile des Prozessdeskriptors!

Antwort:

- Prozessidentifikation
  - Prozess-ID (PID)
- Process State Information
  - Prozesszustand (RUNNING, READY, ...)
  - Priorität
  - Registerinhalte (Register Save Area)
- Process Control Information
  - Besitzer (User ID)
  - Zugriffsrechte (effektive User ID)
  - Liste offener Dateien (File Handles)
  - Verweise auf Programmcode und Daten
  - Accounting Information

---

2. Durch welche drei im Buch erläuterten Arten der Parallelverarbeitung kann die Performance eines Prozessors gesteigert werden?

Antwort:

- Vektorverarbeitung
- Superskalare Verarbeitung
- Instruction-Pipelining

---

3. Was ist die Aufgabe eines *Linkers*?

Antwort: Der Linker fügt die als *Relocatable Object Code* vorliegenden Programmmodule, welche aus entsprechendem Sourcecode kompiliert worden sind, zu einem ausführbaren Programm zusammen.

---

4. Was versteht man unter dem Program Counter (PC) bei einem Mikroprozessor?

Antwort: Der Program Counter ist ein Prozessorregister, welches die Speicheradresse speichert, an welcher der auszuführende Maschinencode gespeichert ist.

---

5. Wo werden die Return-Adressen bei Unterprozeduraufrufen gespeichert?

Antwort: Die Speicherung der Return-Adressen erfolgt auf einem Stack.

---

6. Welche zwei wichtigen Schritte werden beim Aufruf einer Unterprozedur durch einen Subroutine Call verrichtet?

Antwort:

1. Retten des PSW (Program Status Word), der Registerinhalte und des PC (Program Counter)
2. Laden des PC mit der Prozedur-Startadresse

---

7. Welche zwei Möglichkeiten gibt es, die Adresse einer Interrupt Service Routine (ISR) zu bestimmen?

Antwort:

1. Fixe Zuordnung
2. Interruptvektor

---

8. Welche zwei Komponenten benötigt jeder Thread für sich alleine?

1. Registersatz des Prozessors
2. Thread-spezifische Daten (lokale Variablen, Rücksprungadressen etc.)

Antwort:

---

9. Nennen Sie mindestens zwei Probleme bei der Verwendung von Threads!

Antwort:

- Mehrere Threads in einem Prozess verwenden denselben Adressraum; ein Prozess kann daher durch Speichermanipulationen andere Threads in demselben Prozess oder auch den Prozess selbst zum Absturz bringen.
- Um das Prozessverhalten mit den Threads in Einklang zu bringen, ist meistens eine gesonderte bzw. zusätzliche Behandlung notwendig.
- Bei der Verwendung von Libraries, die nicht für den Einsatz in einem mehrfach genutzten Adressraum ausgelegt sind, können Probleme auftreten.

---

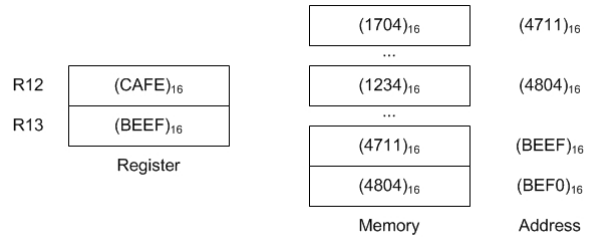
10. Was versteht man unter einem *system call*?

Antwort: System calls sind Betriebssystemaufrufe, mit Hilfe derer ein Prozess in der Lage ist, die verschiedensten Funktionen des Betriebssystems in Anspruch zu nehmen.

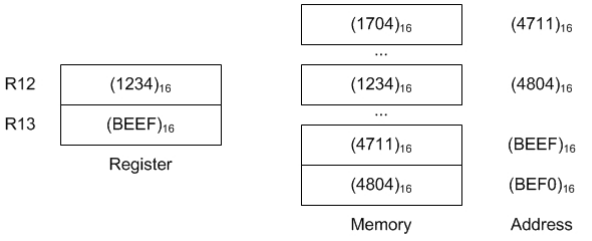
---

11. Geben Sie die Werte in allen **gezeigten Registern und Speicherzellen im Endzustand an, die sich aus dem Ausgangszustand nach dem Abarbeiten der folgenden Instruktion I ergeben.**

Instruktion I (indirect addressing): $R12 \leftarrow memory[memory[R13] + (F3)_{16}]$



Antwort:



---

12. Ein Speicherverwaltungssystem mit Paging verwendet 32-bit virtuelle Adressen und 24-bit physikalische Adressen. Die Größe einer Page beträgt 2 KB (2048 Bytes). Gegeben ist eine Pagetable. Wandeln Sie mit Hilfe dieser Tabelle die folgenden vier virtuellen 32-bit-Adressen in die entsprechende physikalische Adresse um. Geben Sie dabei jeweils an, ob ein *page fault* auftritt.
Virtuelle Adressen:

- $(000008BF)_{16}$
- $(0006E1F2)_{16}$
- $(0006E821)_{16}$
- $(00001047)_{16}$

Pagetable:

Page-Nummer	**Frame-Nummer**
$(0)_{16}$	(Not present)
$(1)_{16}$	(Not present)
$(2)_{16}$	$(AF)_{16}$
...	...
$(D9)_{16}$	(Not present)
$(DA)_{16}$	$(C04)_{16}$
$(DB)_{16}$	(Not present)
$(DC)_{16}$	$(7C)_{16}$
$(DD)_{16}$	(Not present)
$(DE)_{16}$	$(B48)_{16}$
...	...

Antwort:

- Virtuelle Adresse: $(000008BF)_{16}$; Page Number: $(1)_{16}$; Page fault tritt auf
- Virtuelle Adresse: $(0006E1F2)_{16}$; Page Number: $(DC)_{16}$; Offset: $(1F2)_{16}$; Frame Number: $(7C)_{16}$; Physikalische Adresse: $(03E1F2)_{16}$
- Virtuelle Adresse: $(0006E821)_{16}$; Page Number: $(DD)_{16}$; Page fault tritt auf
- Virtuelle Adresse: $(00001047)_{16}$; Page Number: $(2)_{16}$; Offset: $(47)_{16}$; Frame Number: $(AF)_{16}$; Physikalische Adresse: $(057847)_{16}$

---

13. Ein 256 MB-Speicherbauteil hat eine durchschnittliche Zugriffszeit von 12 ns. Um den Zugriff zu beschleunigen, wird ein Cache von 4 MB verwendet, der eine Zugriffszeit von durchschnittlich nur 4 ns hat. Ein Programm zur Bestimmung von Primzahlen weist eine exzellente Lokalität der Speicherzugriffe auf, sodass eine *hit-rate* von 97 % erzielt werden kann. Wie groß ist die effektive Speicherzugriffszeit bei der Abarbeitung dieses Programms?

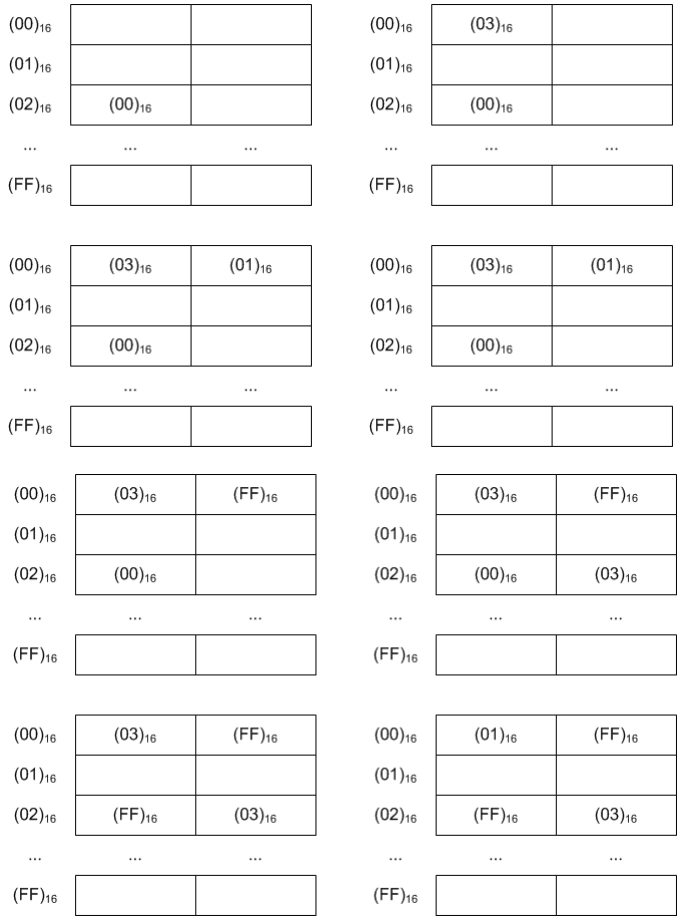
Antwort: $t_{eff} = 4,24ns$

---

14. Gegeben ist ein assoziativer Zweiweg-Cache mit LRU-Verfahren. Geben Sie an, welche Werte sich im 16-Bit-Tag-RAM befinden, wenn der Cache anfangs leer ist und die folgende Reihe von (32-bit) Speicherzugriffen erfolgt. Beachten Sie, dass uns der Datenbereich des Caches nicht interessiert, und er daher auch nicht eingezeichnet werden soll.
Speicherzugriffe (in folgender Reihenfolge):

- $(0002)_{16}$
- $(0300)_{16}$
- $(0100)_{16}$
- $(0300)_{16}$
- $(FF00)_{16}$
- $(0302)_{16}$
- $(FF02)_{16}$
- $(0100)_{16}$

Antwort:



15. Ein Betriebssystem verwendet den *shortest remaining time* (SRT)-Algorithmus für das Prozess-Scheduling. Die folgende Tabelle zeigt den Zeitpunkt, zu dem die jeweiligen Prozesse in den Zustand READY wechseln und deren verbleibende Restlaufzeit. Geben Sie an, in welcher Reihenfolge die Prozesse ausgeführt werden. Dazu tragen Sie in die Tabelle ein, zu welchen Zeitpunkten ein Prozess in den Zustand RUNNING übergeht (den Zeitpunkt *und* den dazugehörigen Prozess in die Tabellen eintragen).
Dabei nehmen Sie einmal an, dass das Betriebssystem nicht-preemptives Scheduling verwendet, und danach, dass preemptives Scheduling unterstützt wird.

Prozess	READY-Zeitpunkt	Restlaufzeit
A	0	2
B	3	7
C	4	3
D	5	1
E	8	5
F	12	4

Antwort: Nicht-preemptives Scheduling:

A	B	D	C	F	E
0	3	10	11	14	18

Preemptives Scheduling:

A	B	C	D	C	E	F	B
0	3	4	5	6	8	13	17

16. Gegeben ist ein Semaphore S mit einem Counter, der auf den Wert 1 initialisiert ist. Dabei ist zu beachten, dass die Ordnung der Prozessqueue dieses Semaphores prioritätsgesteuert funktioniert.

Die folgende Tabelle listet eine Menge von Prozessen auf, die diesen Semaphore verwenden wollen (d.h., die *S_P(S)* aufrufen). Dabei ist der Zeitpunkt gegeben, zu dem die jeweiligen Prozesse den Aufruf *S_P(S)* tätigen, sowie die Zeitspanne, die jeder Prozess den Semaphore hält, bevor er mittels *S_V(S)* diesen wieder freigibt. Außerdem ist die Priorität jedes Prozesses angegeben. Dabei bedeutet ein höherer Wert eine höhrere Priorität.

Prozess	Priorität	Zeitpunkt von *S_P(S)*	Haltedauer
A	1	1	5
B	2	2	3
C	4	4	1
D	0	12	2
E	2	13	1

textbfGeben Sie in einem entsprechenden Diagramm den Wert des Counters von Semaphore S als Funktion der Zeit an, d.h. zeichnen Sie ein, wann und wie sich der Counter auf Grund der Benutzung durch die Prozesse ändert.

Antwort: