Einführung in die Technische Informatik

Prüfungsordner zum 2. Test
22. Jänner 2005

Datum: 24. 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**	**17**	**18**	**19**
1	2	2	2	3	3	1	2	1	2	2	1	3	4	4	6	5	2	4

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

---

1. Was versteht man unter dem Scratchpad eines Prozessors?

Antwort: Das Scratchpad (oder Register File) eines Prozessors ist eine Konfiguration, bei der die ALU (Arithmetic Logic Unit) die Register A und B nur mehr als Zwischenspeicher verwendet.

---

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?

Antwort: Wenn möglichst viele Bauteile auf einem Chip untergebracht werden, so reduzieren sich die Wegstrecken zwischen diesen Bauteilen; damit erhöt sich die Prozessorgeschwindigkeit, da sich elektrischer Strom mit etwa der 0,7-fachen Lichtgeschwindigkeit ausbreitet.

---

3. Das Instruktionsformat der x86-Prozessorreihe von Intel (der auch die neuesten 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.

Antwort: Häufig verwendete Maschineninstruktionen können mit kürzeren Bitmustern codiert werden als weniger oft verwendete. Dadurch kann die Länge von Programmen reduziert werden. Dem gegenüber steht jedoch, dass ein Interpreter, der Maschineninstruktionen variabler Länge verarbeiten muss, naturgemäß langsamer arbeitet als ein Interpreter, der lediglich mit Instruktionen fester Länge zu tun hat.

---

4. Welche zwei Möglichkeiten gibt es, die Adressen der Ports für Input-/Output-Operationen zu vergeben?

Antwort:

- Independent I/O-System: Hauptspeicher und Ports haben völlig voneinander unabhängige Adressen.
- Memory-mapped-I/O-System: Ports werden so behandelt, als wären sie gewöhnliche Speicherstellen.

---

5. Was ist ein Trap? Geben Sie ein Beispiel.

Antwort: Ein Trap (auch Interrupt genannt) ist eine Unterbrechung der momentan vom Prozessor ausgeführten Programmsequenz, um auf einen im normalen Programmablauf nicht vorgesehenen Zustand zu reagieren.

---

6. Geben Sie zumindest drei verschiedene Segmente an, die sich in einem ausführbaren Programm befinden.

Antwort:

---

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

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

---

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 die Prozessoren eine sehr lange Pipeline (oft mehr als 20 Stufen) verwenden.

Antwort: Normalerweise werden der Pipeline die Daten sequentiell, wie sie kommen, zugeführt. Wird aber ein Sprung ausgeführt (dies wird erst später in der Pipeline erkannt) sind alle Daten, die nach diesem Sprung in die Pipeline geraten sind, wertlos und müssen verworfen werden. Ziel der *branch prediction* ist ein möglichst frühes Erkennen eines Sprungbefehls und Erkennen seiner Sprungzieladresse, damit gleich die Daten der Zieladresse dem Sprungbefehl in die Pipeline folgen können.

---

9. Was versteht man unter *logischer Parallelität* von Prozessen?

Antwort: Logische Parallelität von Prozessen liegt dann vor, wenn mehrere Prozesse scheinbar gleichzeitig auf einem einzelnen Prozessor ausgeführt werden.

---

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

Antwort: Der Network Layer stellt End-to-End-Verbindungen zwischen einzelnen Hosts her. Vorherrschend im Internet ist das IP-Protokoll.

---

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

Antwort: Ein Prozess hat zu jeder Zeit damit zu rechnen, auf Grund eines eingehenden Signals in der Ausführung unterbrochen zu werden.

---

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

Antwort: Der adressierbare Adressraum hat eine Größe von 64 KByte (65.536 Bytes).

---

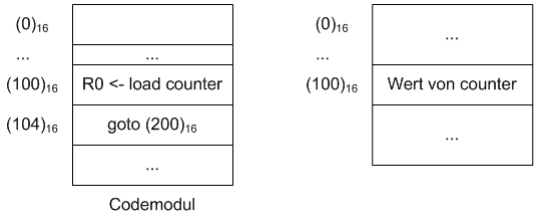
13. Wenn der Parent-Prozess in einem Unix-Betriebssystem auf das Terminieren seines Child-Prozesses wartet (mittels des *wait()* system calls), dann wird der Prozess so lange blockiert, bis der Child-Prozess einen *exit()* system call macht. Dabei wird das Ergebnis des Child-Prozesses (sein Rückgabewert) 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?

Antwort: Der Child-Prozess terminiert, sein Rückgabewert geht verloren.

---

14. Ein Linker soll ein Codemodul und ein Datenmodul zu einem ausführbaren Programm zusammenfügen. Sowohl das Codemodul als auch das Datenmodul liegen als Relocatable Object Code vor. Das Codemodul soll im ausführbaren Programm an der virtuellen Adresse 0x08048000 beginnen. Das Datenmodul soll an der virtuellen Adresse 0x4000000 liegen. Im Folgenden sind Ausschnitte aus dem Codemodul und dem Datenmodul gegeben.



Geben Sie an, nach welchen virtuellen Adressen (a) der *load* Befehl, (b) der *goto* Befehl und (c) der Wert der Variablen *counter* nach dem Linken (im ausführbaren Programm) gespeichert sind.
Geben Sie außerdem an, (d) zu welcher virtuellen Zieladresse der *goto* Befehl springt und (e) zu welcher Adresse die *unresolved external address* des *load* Befehls aufgelöst wird.

Antwort:

(a) 0x08048100

(b) 0x08048104

(c) 0x40000200

(d) 0x08048200

(e) 0x40000200

---

15. Der im Buch beschriebene Mikroprozessor 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 an 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 überschrieben werden können, weil sie die Konstanten 0, 1 bzw. -1 enthalten.

Antwort:

```
1. R3  <- lsh(1+1);     # 1+1=(10)2; lsh((10)2)=(100)2=(4)10
2. R3  <- lsh(R3);      # lsh((100)2)=(1000)2=(8)10
3. R3  <- lsh(R3);      # lsh((1000)2)=(10000)2=(16)10
4. R4  <- 1;
5. R11 <- R3+R4;        # 16+1=17
```

---

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 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:

- $(000078BF)_{16}$
- $(001DD1AB)_{16}$
- $(001DA472)_{16}$
- $(00004A02)_{16}$

Pagetable:

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)
$(EE)_{16}$	$(380)_{16}$
...	...

Antwort:

- Virtuelle Adresse: $(000078BF)_{16}$; Page Number: $(3)_{16}$; Offset: $(18BF)_{16}$; Frame Number: $(1B)_{16}$; Physikalische Adresse: $(0378BF)_{16}$
- Virtuelle Adresse: $(001DD1AB)_{16}$; Page Number: $(EE)_{16}$; Offset: $(11AB)_{16}$; Frame Number: $(380)_{16}$; Physikalische Adresse: $(7011AB)_{16}$
- Virtuelle Adresse: $(001DA472)_{16}$; Page Number: $(ED)_{16}$; Page fault tritt auf
- Virtuelle Adresse: $(00004A02)_{16}$; Page Number: $(2)_{16}$; Page fault tritt auf

---

17. Ein Betriebssystem verwendet den LFU (last frequently used) Algorithmus für das Page Replacement. Wenn zwischen Pages unterschieden 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 vier physikalische Frames zur Verfügung. Im Folgenden ist die Reihenfolge der Pagenummern angegeben, auf die der Prozess zugreift (Referenzstring). Geben Sie nach jedem Zugriff an, welche Pages sich im physikalischen Speicher befinden.
Referenzstring: 1 2 1 3 4 2 5 5 6 7 8 5 8 9 9 10

Antwort:

1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2
			3	3	3	5	5
				4	4	4	4

1	1	1	1	1	9	9	9
2	2	2	2	2	2	2	10
5	5	5	5	5	5	5	5
6	7	8	8	8	8	8	8

---

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?

Antwort: Der Stackpointer zeigt auf 0xFFA6.

---

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

Programm:

```
 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.

Antwort: Variable j hat in Thread A den Wert 11 und in Thread B den Wert 12.