

INSTITUT FÜR INFORMATIK
Lehrstuhl für Rechnerarchitektur (Informatik 3)
Universität Erlangen-Nürnberg
Martensstr. 3, 91058 Erlangen

14.07.2003

Klausur
 zu

"Organisation und Technologie von Rechensystemen 4"

.....
 Matrikelnummer Geb.-Datum Vorname Name

- Es sind keine elektronischen Hilfsmittel erlaubt.
- Legen Sie den Ausweis (mit Lichtbild!) griffbereit auf den Platz.
- Dieses Aufgabenheft umfaßt 10 Seiten. Überprüfen Sie die Vollständigkeit.
- Gesondert beigelegte Blätter werden nicht bewertet!
- Schreiben Sie deutlich! Unleserliches wird nicht bewertet!
- Es darf nicht mit der Farbe rot geschrieben werden!
- Bekanntgabe der Ergebnisse: Aushang in den Semesterferien
- Einsichtnahme: Siehe Aushang
- Bei Bestehen wird der Schein direkt an das Prüfungsamt geschickt!

Durch meine Unterschrift bestätige ich

- den Empfang der vollständigen Klausurunterlagen
- die Kenntnisnahme der obigen Informationen.

Erlangen, den 14.07.2003

(Unterschrift)

Ich bin damit einverstanden, daß mein Prüfungsergebnis der Klausur zu OTRS 4 unter Angabe der Matrikel-Nummer veröffentlicht wird.

Erlangen, den 14.07.2003

(Unterschrift)

Aufgabe	1	2	3	4	5	6	Summe
max. Punktzahl	8	8	12	12	8	12	60
erreichte Punktzahl							

Aufgabe 1: Speicher**(8 Punkte)**

- 1a) Wieviel Speicher kann ein Prozessor mit 16-Bit breitem Adressbus direkt ansprechen, wenn der Datenbus auch 16-Bit breit ist? (2 Punkte)
- 1b) Braucht ein Prozessor, der Daten im Big-Endian-Format speichert, mehr, genauso viel oder weniger Speicher für zu speichernde Daten als ein Prozessor mit Little-Endian-Format? (2 Punkte)
- 1c) Braucht ein Prozessor mit 32-Bit-Alignment für zu speichernde Daten immer mehr, manchmal mehr oder immer genauso viel Speicher wie ein Prozessor mit 8-Bit-Alignment? (2 Punkte)
- 1d) Wieviele der folgenden Strukturen passen in einen 32-KByte großen Hauptspeicher (char: 1 Byte, int: 4 Byte, Adressen: 4 Byte)? (2 Punkte)

```
struct data {
    struct data *prev; /* Adresse vorhergehender Datensatz */
    struct data *next; /* Adresse naechster Datensatz */

    unsigned int punkte;
    char name[20];
};
```

Aufgabe 2: Arithmetik**(8 Punkte)**

Implementiert werden soll eine Funktion, die das arithmetische Mittel zweier Zahlen berechnet:

$$\text{am}(x, y) = (x + y) / 2$$

2a) Welchen Wert wird die Hochsprachenfunktion

```
unsigned int
am(unsigned int x, unsigned int y)
{
    return (x + y) / 2;
}
```

vermutlich für die Werte $x = 0x7fffffff$ und $y = 0x80000001$ bei 32-Bit-Arithmetik liefern? Begründen Sie Ihre Antwort! (3 Punkte)

2b) Erklären Sie, warum bei folgender Assembler-Code-Implementierung dieser Funktion für alle Werte x, y des 32-Bit-Wertebereiches korrekte Ergebnisse geliefert werden! (5 Punkte)

```
am:
    movl    4(%esp), %eax
    addl   8(%esp), %eax
    rorl   $1, %eax
    ret
```

Aufgabe 3: Unterprogramm**(12 Punkte)**

Geschrieben werden soll ein Assembler-Unterprogramm, das den größten von N auf dem Stack übergebenen Werten ermittelt. Dem Unterprogramm wird neben den N Werten auch noch die Anzahl der Werte selbst (das N) übergeben.

Beispiel:

```
res = max(3, a, b, c);
```

soll als Resultat `res` das Maximum der drei (unsigned long) Zahlen `a`, `b` und `c` ermitteln.

Gehen Sie davon aus, dass der Compiler beim Aufruf Ihrer Funktion die übergebenen Werte von rechts nach links auf dem Stack ablegt (den Wert N damit als letztes). Das Resultat wird in `%eax` zurück erwartet.

Für das Beispiel oben würde der Compiler daher etwa folgenden Code erzeugen:

```
    pushl   c
    pushl   b
    pushl   a
    pushl   3
    call    max
x:   addl    16, %esp
    movl    %eax, res
```

- 3a) Welchen Inhalt haben die Speicherzellen 40, 44, 48, 52, 56 beim Start des Unterprogramms `max` im obigen Beispiel? Der Stack-Pointer enthalte vor Ausführung der ersten `pushl`-Anweisung den Wert 60. (3 Punkte)

3b) Schreiben Sie das Assembler-Unterprogramm max! Kommentieren Sie Ihren Code! (9 Punkte)

Aufgabe 4: Programm-Transformation**(12 Punkte)**

Schreiben Sie den unten stehenden Teil eines Hochsprachenprogramms so um, dass er sich (fast) direkt in Assembler-Code umwandeln läßt! Teilen Sie Ausdrücke in möglichst einfache Teil-Ausdrücke für eine Ein-Adress-Maschine auf! Wandeln Sie alle Kontrollfluss-Anweisungen in einfachere Goto- bzw. If-Then-Goto-Statements! (12 Punkte)

```
for (i = 0; i < 10; i++) {
    if (i < 5) {
        if (a[i] + 10 < i || ! (i < y)) {
            break;
        }
    } else {
        y += a[i] * (b + c) / d;
    }
}
```

Aufgabe 5: Stack**(8 Punkte)**

Welcher Wert steht nach der Ausführung des folgenden Assembler-Code-Fragments im Register `%eax`?
Begründen Sie Ihre Antwort! (8 Punkte)

```
func:
    ...
    call    x
x:
    popl   %eax
    subl   $x - func, %eax
    ...
```

Aufgabe 6: Hardware**(12 Punkte)**

Entwerfen Sie ein möglichst einfaches Schaltnetz, das eine 8-Bit-Zahl inkrementieren kann! Verwendet werden dürfen AND-, OR-, XOR- und NOT-Gatter. (12 Punkte)

Zusätzlicher Platz

Zusätzlicher Platz