DEPARTMENT INFORMATIK
Lehrstuhl für Informatik 3 (Rechnerarchitektur)
Friedrich-Alexander-Universität Erlangen-Nürnberg
Martensstraße 3, 91058 Erlangen
15.02.2010

Klausur zu

Grundlagen der Rechnerarchitektur und -organisation

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

- Es sind *keine* Hilfsmittel erlaubt (Ausnahme: 1 DIN-A4 Blatt mit Notizen)!
- Legen Sie den Ausweis (mit Lichtbild!) griffbereit auf den Platz!
- Dieses Aufgabenheft umfasst 18 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!
- Offensichtlich falsche oder überflüssige Antworten können zu Punktabzug führen!

Durch meine Unterschrift bestätige ich

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

Erlangen, den 15.02.2010 .....
(Unterschrift)

Ich bin damit einverstanden, dass mein Prüfungsergebnis der Klausur unter Angabe der Matrikel-Nummer veröffentlicht wird.

Erlangen, den 15.02.2010 .....
(Unterschrift)

Aufgabe	1	2	3	4	5	6	7	8	9	Summe
max. Punktzahl	8	8	12	6	5	15	10	12	14	90
erreichte Punktzahl										

Aufgabe 1: Der Universalrechenautomat (8 Punkte)

- Nennen Sie die Phasen des Maschinenbefehlszyklus des klassischen von-Neumann-Rechners und erklären Sie, was in jeder Phase jeweils passiert.
  (4 Punkte)

- Wie fügen sich Interrupts und Exceptions in dieses Bild ein?
  (4 Punkte)

Aufgabe 2: Befehlssatz-Architekturen (8 Punkte)

Zur Berechnung von

```
x = y - z;
```

soll auf den folgenden Befehlssatz-Architekturen jeweils ein kleines Assembler-Programm programmiert werden. Gehen Sie davon aus, dass y und z zu Programmstart im Speicher liegen. Das Ergebnis x soll das Programm in den Speicher zurückschreiben.

- Stack (2 Punkte)

- Accumulator (2 Punkte)

- Register-Memory (2 Punkte)

- Register-Register (2 Punkte)

Aufgabe 3: Speicher (12 Punkte)

- Definieren Sie die Begriffe ,,Big-Endian" und ,,Little-Endian"!
  (2 Punkte)

- Schreiben Sie die Zahlen 1 und 256 jeweils als 2-Byte-Zahl einmal im Big-Endian- und einmal im Little-Endian-Format!
  (2 Punkte)

- Definieren Sie den Begriff „Alignment"!
  (2 Punkte)

- Wieviel Speicherplatz braucht ein Datensatz mit korrektem Alignment auf einem 32-Bit-Rechner, der folgende Daten (in dieser Reihenfolge) enthält:

  – drei 16-Bit-Integer-Zahlen (a, b, c)
  – zwei 32-Bit-Integer-Zahlen (d, e)
  – ein 8-Bit-Zeichen (f)

  Wieviel Speicherplatz benötigt ein Array bestehend aus 10 solcher Datensätze? Die erste Zahl des Datensatzes liege jeweils an einer durch 4 teilbaren Adresse.
  Begründen Sie Ihre Antworten!
  (6 Punkte)

Aufgabe 4: CPU-Architekturen (6 Punkte)

- Was versteht man unter dem Begriff „Pipelining"?
  (2 Punkte)

- Was versteht man unter dem Begriff „Multi-Core-CPU"?
  (2 Punkte)

- Was versteht man unter dem Begriff „Super-Skalar-CPU"?
  (2 Punkte)

Aufgabe 5: Busse (5 Punkte)

1. Beschreiben Sie die Aufgaben der folgenden Leitungen!

   - Adress-Leitungen
   - Daten-Leitungen
   - Kontroll-Leitungen

   (3 Punkte)

2. Wieviel Speicher kann man maximal direkt an einen Bus anschließen, der 30 Adress- und 32 Daten-Leitungen enthält?
   (2 Punkte)

Aufgabe 6: Ausdrücke (15 Punkte)

Schreiben Sie eine Assembler-Funktion, die zwei Fließkomma-Zahlen miteinander multipliziert. (Überläufe und Unterläufe dürfen ignoriert werden, beide Operanden nicht 0.) Gehen Sie davon aus, dass der verwendete Prozessor Integer-Additions-, -Subtraktions-, -Multiplikations-, und -Divisions-Operationen sowie Shift- und Logik-Operationen für 32-Bit-Arithmetik bietet.

Die Fließkomma-Zahlen seien wie folgt kodiert:

- Bit 31: Vorzeichen
- Bit 30-16: Charakteristik
- Bit 15-0: Mantisse (einschließlich der führenden „1")

Die zwei zu multiplizierenden Zahlen werden in den Speicherzellen `a` und `b` übergeben und das Ergebnis wird in einem Register zurückerwartet.

Aufgabe 7: Zwei-Adress-Code (10 Punkte)

Vereinfachen Sie das folgende Code-Stück für eine Zwei-Adress-CPU!

Benutzen Sie nur

- `if-then-goto`-Anweisungen mit einfachen Bedingungen
- `goto`-Anweisungen
- Funktionsaufrufe der Form `result = function(param1, ..., paramN);`
- Zuweisungen der Form
  - `var = operand;`
  - `var = var <binop> operand;`

  (`binop` ist Element von $\{+, -, *, /, <, >, ...\}$, `var` ist eine Variable, `operand` kann eine Variable oder eine Konstante sein).

```
val = 0;
c = fgetc(fp);
while ('0' <= c && c <= '9') {
    val = val * 10 + c - '0';
    c = fgetc(fp);
}
```

Aufgabe 7: Zwei-Adress-Code (Fortsetzung)

Aufgabe 8: Protection (12 Punkte)

Eine CPU biete eine Memory-Management-Unit mit folgenden Eigenschaften:

- zweistufige Adresstabellen
- je 1024 Einträge zu je 4 Byte in den Tabellen
  - Bit 31-12: höherwertige Bits der physikalischen Adresse
  - Bit 11-2: unbenutzt
  - Bit 1: Present-Bit
  - Bit 0: Read-Only-Bit
- Pages zu je 4 kByte Größe

Folgendes soll für ein Programm erfüllt sein:

- im virtuellen Adressbereich `0x00001000` bis `0x00002fff` soll auf einen ROM-Baustein zugegriffen werden (Code),
- im virtuellen Adressbereich `0x003f0000` bis `0x003fefff` soll auf Hauptspeicher zugegriffen werden (Daten),
- im virtuellen Adressbereich `0x003ff000` bis `0x003fffff` soll auf Hauptspeicher zugegriffen werden (Stack),
- alle anderen Bereiche sollen Zugriffsfehler auslösen.

Physikalisch sind der ROM-Baustein an Adresse `0x0fffe000` und der Hauptspeicher von Adresse `0x00000000` bis `0x08000000` am Bus.

Beschreiben Sie eine mögliche Page-Tabelle, die obige Bedingungen erfüllt!

Aufgabe 8: Protection (Fortsetzung)

Aufgabe 9: Caching (14 Punkte)

Gegeben sei folgendes Assemblerprogramm:

```
    movl $0, %eax

    movl $1, 2000

    movl $1, 2004

    movl $2000, %ebx

loop:
    movl (%ebx), %ecx

    addl 4(%ebx), %ecx

    movl %ecx, 8(%ebx)

    addl $4, %ebx

    addl $1, %eax

    cmpl $11, %eax

    jne loop
```

Ein Befehl soll 1 Nanosekunde zur Bearbeitung benötigen. Ein Hauptspeicherzugriff dauert 100 Nanosekunden. Jeder Befehl soll mit einem Speicherzugriff geladen werden können.

1. Kommentieren Sie die einzelnen Assembler-Befehle!
   (2 Punkte)

2. Auf welche Daten-Bytes im Speicher greift das Programm zur Laufzeit zu?
   (2 Punkte)

Aufgabe 9: Caching (Fortsetzung)

3. Wie lange benötigt das Programm zur Abarbeitung, wenn keine Caches verwendet werden?
   (3 Punkte)

4. Der Prozessor soll nun über einen ausreichend großen Primär-Cache verfügen, um sämtliche Befehle und Daten des Programms zwischenzuspeichern. Ein Zugriff auf diesen Cache benötigt 5 Nanosekunden. Ursprünglich sei der Cache leer. Die Schreib-Strategie sei „Write-Through". Wie lange benötigt das Programm nun?
   (4 Punkte)

5. Ein „Direct-Mapped"-Cache habe die Größe 2^{20} Bytes. Die Adressen im Rechner seien 32 Bit groß. Wie groß müssen die Tag-Felder im Cache sein? Begründen Sie ihre Antwort!
   (3 Punkte)

Zusätzlicher Platz

Zusätzlicher Platz