DEPARTMENT INFORMATIK
Lehrstuhl für Informatik 3 (Rechnerarchitektur)
Friedrich-Alexander-Universität Erlangen-Nürnberg
Martensstraße 3, 91058 Erlangen
29.09.2010

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 20 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 29.09.2010 .....
(Unterschrift)

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

Erlangen, den 29.09.2010 .....
(Unterschrift)

| Aufgabe | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Bonus | Summe |
|---|---|---|---|---|---|---|---|---|---|
| max. Punktzahl | 7 | 13 | 8 | 10 | 28 | 18 | 16 | 15 | 115 |
| erreichte Punktzahl | | | | | | | | | |

## Aufgabe 1: Instruktionssatzarchitektur (7 Punkte)

- Vier Klassen von Befehlssatz-Architekturen wurden in der Vorlesung vorgestellt. Zu welcher Klasse gehört eine CPU, die nachfolgenden Assembler-Code ausführen kann? Begründen Sie Ihre Antwort!
  (2 Punkte)

```
load a
add b
store tmp
load a
sub b
mul tmp
store c
```

- Schreiben Sie obigen Assembler-Code um, sodass er für eine CPU der Klasse „Register-Register/Load-Store" geeignet ist! Die CPU kenne die Register `%r0` bis `%r15`.
  (2 Punkte)

- Was unterscheidet eine RISC- und eine CISC-CPU? Nennen Sie sechs Unterscheidungsmerkmale!
  (3 Punkte)

## Aufgabe 2: Speicher (13 Punkte)

Gegeben sei folgendes Assembler-Unterprogramm:

```
func:
    jmp next            % springe nach "next"
loop:
    movb (%ebx), %dl    % kopiere Byte von Adr. %ebx ins Reg. %dl
    movb %dl, (%ecx)    % kopiere Byte von Reg. %dl nach Adr. %ecx
    addl $1, %ebx       % addiere 1 zum Inhalt von Reg. %ebx
    addl $1, %ecx       % addiere 1 zum Inhalt von Reg. %ecx
    subl $1, %eax       % subtrahiere 1 vom Inhalt des Reg. %eax
next:
    cmpl $0, %eax       % vergleiche Inhalt von Reg. %eax mit 0
    jne loop            % springe nach "loop" wenn nicht gleich
    ret
```

Das Unterprogramm habe die Startadresse 0xf000. Jeder Befehl sei in 32 Bits kodiert.

- Welche Bytes im Speicher werden gelesen, welche beschrieben, wenn das Unterprogramm mit `%eax`=10, `%ebx`=0x1000 und `%ecx`=0x2000 aufgerufen wird?
  (4 Punkte)

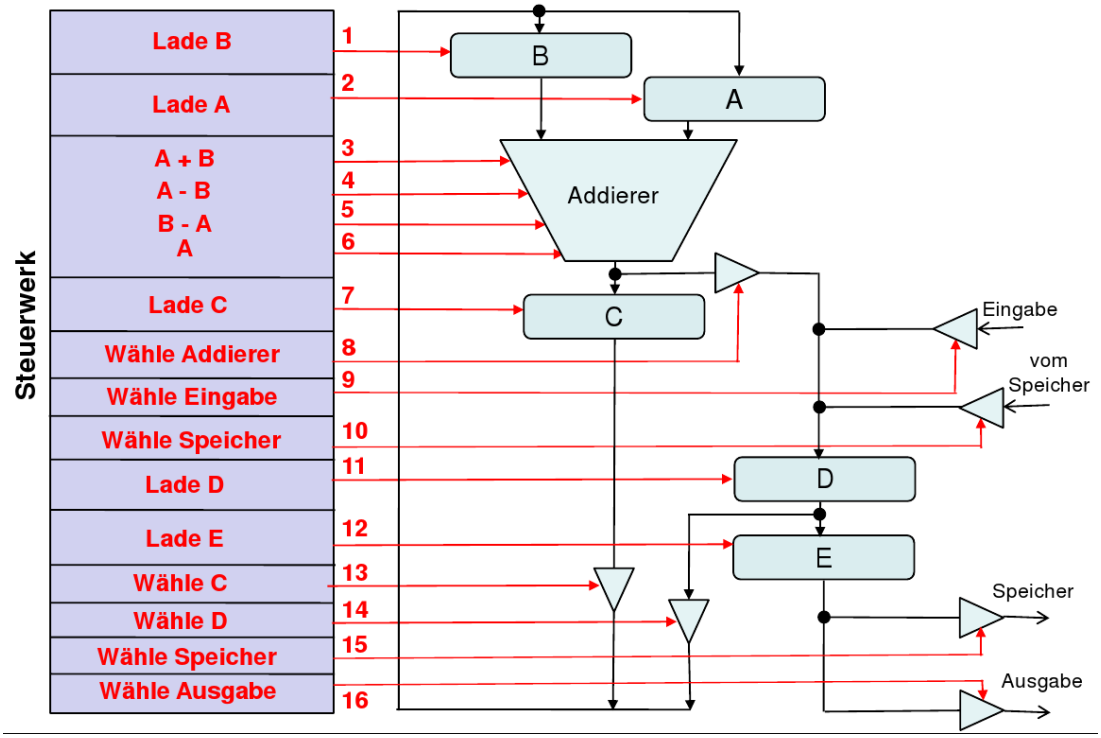
- Was wird als ,,räumliche Lokalität" und was als ,,zeitliche Lokalität" bezeichnet? Wie zeigt sich die räumliche und zeitliche Lokalität am obigen Beispiel?
  (4 Punkte)

- Welche Bytes aus dem Hauptspeicher müssten von einem Cache sinnvollerweise mindestens aufgenommen werden können, um das Programm zu beschleunigen? Begründen Sie Ihre Antwort!
  (2 Punkte)

- Es gibt drei Gründe, warum ein Speicherwort möglicherweise nicht im Cache liegt. Beschreiben Sie diese!
  Welcher dieser drei Gründe kann in einem voll-assoziativen Cache *nicht* auftreten?
  (3 Punkte)

## Aufgabe 3: Mikroprogrammierung (8 Punkte)

Gegeben sei der aus der Vorlesung/Übung bekannte Teil einer CPU:




- Schreiben Sie ein Mikroprogramm, das nacheinander zwei Werte von der Eingabe liest, diese zusammenaddiert und danach zur Ausgabe leitet!
  (8 Punkte)

## Aufgabe 4: I/O (10 Punkte)

- Beschreiben Sie die Idee, wie mittels Polling Bytes von einer seriellen Schnittstelle von der CPU abgeholt werden können.
  (2 Punkte)

- Was ändert sich an der obigen Beschreibung, wenn Interrupts verwendet werden sollen?
  (2 Punkte)

- Warum kann mittels DMA ein Rechner beschleunigt werden? (2 Punkte)

- Ein Bus enthalte 32 gemultiplexte Daten- und Adressleitungen. Er wird mit 133 MHz betrieben. Wieviele Bytes können pro Sekunde mittels Burst-Modus maximal übertragen werden? Eine Abschätzung ist ausreichend. Begründen Sie Ihre Antwort!
  (4 Punkte)

## Aufgabe 5: Pipelining (28 Punkte)

- Nennen Sie die Phasen einer Befehlsabarbeitung!
  (3 Punkte)

- Können Phasen zusammengefasst werden? Geben Sie eine eine kurze Begründung an!
  (2 Punkte)

- Kann es sinnvoll sein, Phasen weiter zu unterteilen? Begründen Sie Ihre Antwort!
  (2 Punkt)

- Um wieviel ist eine CPU mit 5-Stufen-Pipelining maximal schneller als eine CPU ohne Pipelining? Begründen Sie Ihre Antwort!
  (3 Punkt)

- Bei der Abarbeitung von Sprüngen sind bei CPUs mit Pipelining Besonderheiten zu beachten. Erklären Sie dies!
  (4 Punkte)

- Gegeben sei folgendes Code-Fragment:

```
load x, %r0
load y, %r1
add %r0, %r1, %r2
sub $1, %r2, %r2
store %r2, z
```

  Warum wird bei Pipelining ohne weitere Vorkehrungen nach der Abarbeitung dieses Code-Fragments nicht das erwartete Ergebnis in der Speicherzelle `z` stehen?
  (4 Punkte)

- Gegeben sei eine 4-stufige Pipeline, die folgende Phasen beinhaltet:

  1. „Befehl holen",
  2. „Operanden holen",
  3. „Befehl ausführen" und
  4. „Ergebnis zurückschreiben".

  Die CPU führe unten stehendes Programmstück aus. Welche Werte stehen nach fünf Takten in den Registern `%r1`-`%r3`? Tragen Sie auch die Werte in den Registern zwischen den einzelnen Pipeline-Stufen in unten stehende Tabelle ein! Alle Register enthalten bei Programmstart den Wert 0.

```
0: add $1, %r1, %r1
1: sub $2, %r2, %r3
2: nop
3: mul %r1, %r2, %r2
4: add $1, %r1, %r1
```

  (10 Punkte)

| | 1↔2 | 2↔3 | | | | 3↔4 | | | Register | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Takt | Inst | Inst | Op0 | Op1 | Ziel | Inst | Res | Ziel | %r1 | %r2 | %r3 |
| 1 | | ? | ? | ? | ? | ? | ? | ? | | | |
| 2 | | | | | | ? | ? | ? | | | |
| 3 | | | | | | | | | | | |
| 4 | | | | | | | | | | | |
| 5 | | | | | | | | | | | |

## Aufgabe 6: Segmentierung/MMU (18 Punkte)

- Beschreiben Sie, wie sich die physikalische Adresse bei Segmentierung aus einer gegebenen Segment-Nummer und einem gegebenen Offset berechnet und wie der Offset überprüft wird!
  (4 Punkt)

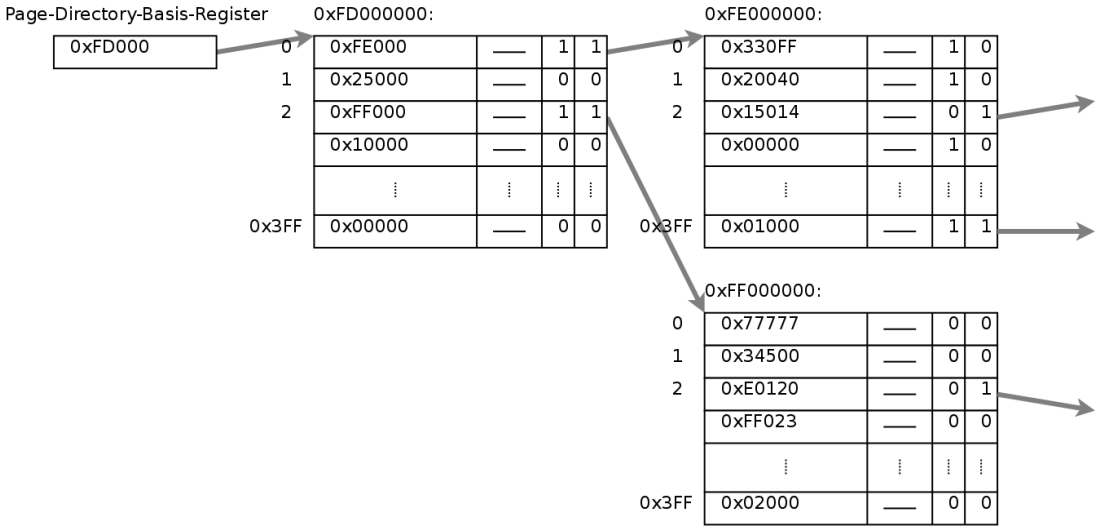
- Wo sind die Page-Tables der MMU i.A. gespeichert?
  (1 Punkt)

- Warum werden Page-Tables i.A. mehrstufig organisiert?
  (2 Punkt)

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: Write-Enable-Bit
  - Bit 0: Present-Bit
- Pages zu je 4 KByte Größe

Gegeben seien folgende Page-Tables:




- Welche Einträge stehen in einem voll-assoziativen TLB mit vier Einträgen, nachdem die CPU nacheinander auf die virtuellen Adressen 0x00802111, 0x003FF124, 0x00002346 und 0x003FF834 lesend zugegriffen hat?
  (10 Punkte)

- Was passiert, wenn die CPU auf die virtuelle Adresse 0x00801234 zugreift? (1 Punkt)

## Aufgabe 7: Assembler-Programmierung (16 Punkte)

- Vereinfachen Sie den folgenden bedingten Ausdruck und stellen Sie ihn nur mit Hilfe von if-then-goto-/goto-Anweisungen dar!
  (6 Punkte)

```
if (((a > 0) && (b > 0)) || ((a < 0) && (b < 0)))
    x();
else
    y();
```

- Ein Compiler habe folgenden Assembler-Code generiert:

```
func:
    cmpl $0, 4(%esp)
    jle error
    movl $0, %eax
    movl $1, %edx
loop:
    addl %edx, %eax
    addl $1, %edx
    cmpl %edx, 4(%esp)
    jge loop
    jmp end
error:
    movl $-1, %eax
end:
    ret
```

  Wie könnte der dazugehörige Hochsprachen-Code ausgesehen haben? (10 Punkte)

## Zusätzlicher Platz

## Zusätzlicher Platz