Department Informatik
Lehrstuhl für Informatik 3 (Rechnerarchitektur)
Friedrich-Alexander-Universität Erlangen-Nürnberg
Martensstraße 3, 91058 Erlangen
10.02.2014
Klausur zu

Grundlagen der Rechnerarchitektur und -organisation

.....	.....	.....	.....
Matrikelnummer	Geburtsdatum	Vorname	Name

- Es sind *keine* Hilfsmittel erlaubt!
- Legen Sie den Ausweis (mit Lichtbild!) griffbereit auf den Platz!
- Dieses Aufgabenheft umfasst 22 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!
- Begründen Sie Ihre Antworten!

Durch meine Unterschrift bestätige ich

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

Erlangen, den 10.02.2014 .....
(Unterschrift)

Ich bin damit einverstanden, dass mein Prüfungsergebnis der Klausur unter Angabe der Matrikelnummer veröffentlicht wird.

Erlangen, den 10.02.2014 .....
(Unterschrift)

Aufgabe	1	2	3	4	5	6	7
max. Punktzahl	10	14	16	14	14	12	10
erreichte Punktzahl							

Summe	/90		
Bonus	/15		
Gesamt		Note	

Aufgabe 1: Umformung — 10 Punkte

Gegeben sei folgendes Hochsprachenprogramm:

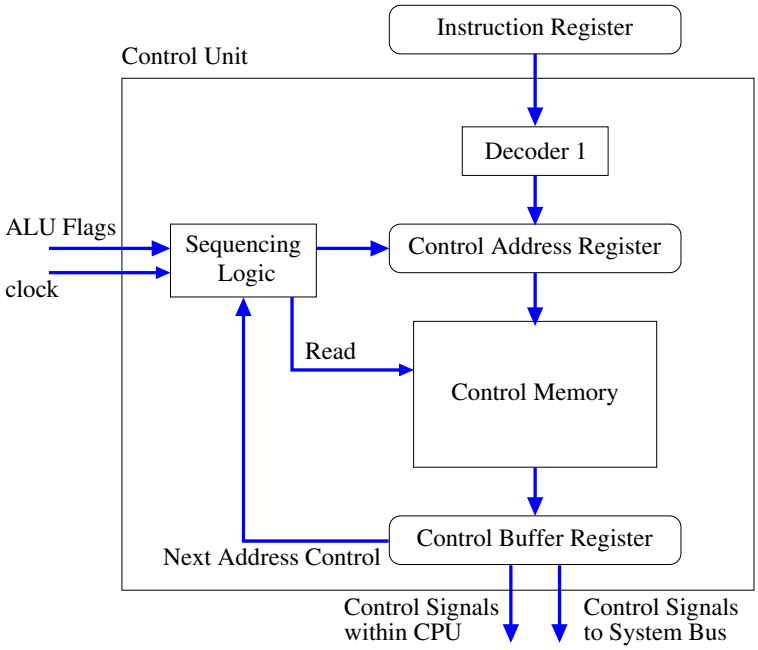
```
1  void bubbleSort(int *A, int length) {
2    for ( int n = length; n > 1; n-- ) {
3      for ( int i = 0; i < n-1; i++ ) {
4        if ( A[i] > A[i+1] ) {
5          int tmp = A[i];
6          A[i] = A[i+1];
7          A[i+1] = tmp;
8      }
9    }
10 }
```

1. Formen Sie die Funktion so in if-goto-Darstellung um, dass sie sich möglichst leicht in Assembler übersetzen lässt. Einfache Arrayzugriffe in der Form `A`[i] können direkt verwendet werden. Vergleichsoperanden und komplexere Adressierungen müssen vorher berechnet werden.

   Verändern Sie weder den Ablauf noch die ausgeführten Operationen. — 10 Punkte

Aufgabe 2: Mikroprogrammierung — 14 Punkte

Gegeben sei folgende Darstellung der Ablaufsteuerung einer mikroprogrammierten Architektur:



1. Nennen Sie jeweils die Aufgaben der folgenden Einheiten:

   - Control Address Register:

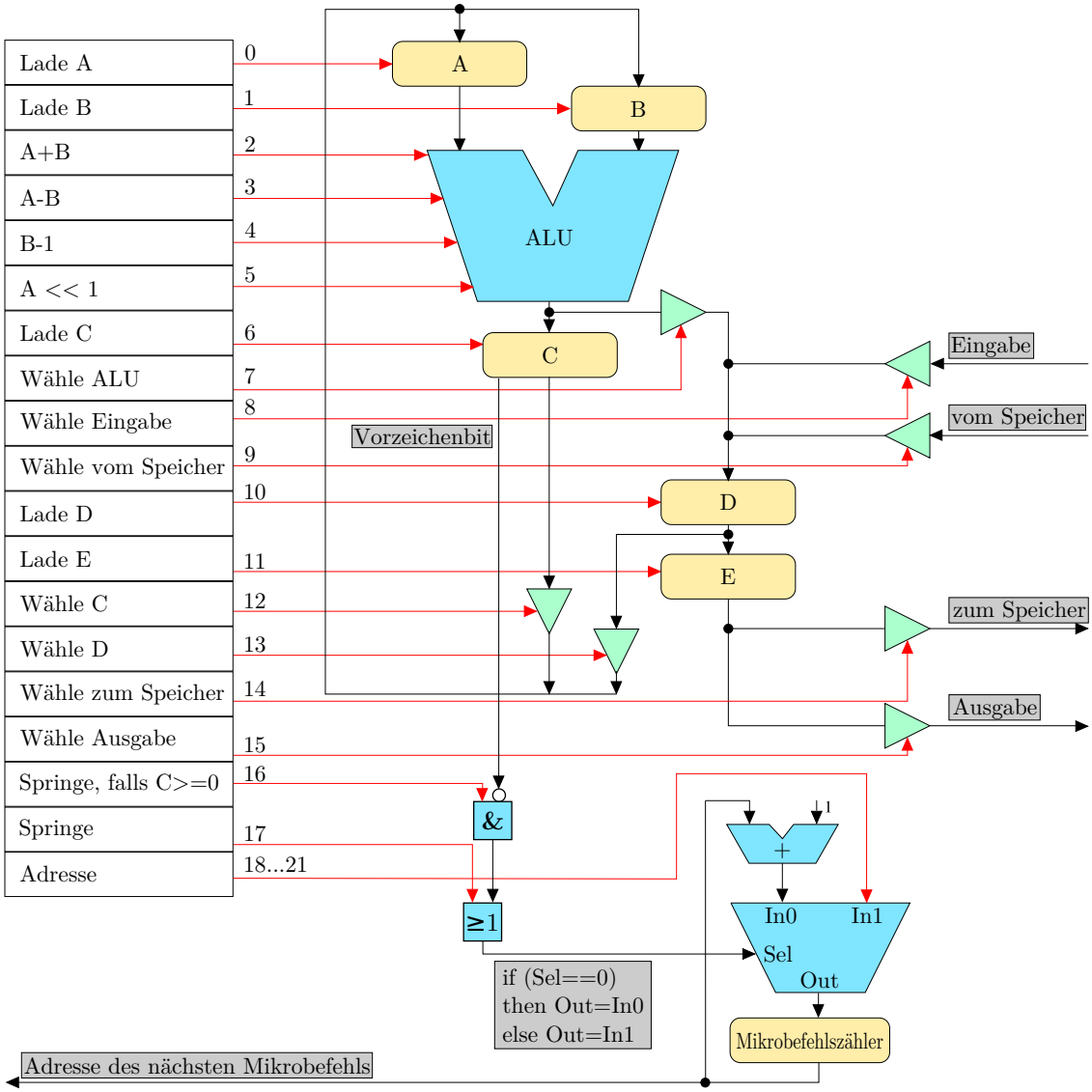
   - Decoder 1:

   - ALU-Flags:

   3 Punkte

2. Welche Form der Mikroprogrammierung wird hier verwendet und woran ist dies erkennbar?
   1 Punkt

Gegeben sei der folgende Teil einer CPU:



Die CPU soll um den Makrobefehl `Reg[A] mod Eingabe` → `Reg[A]` erweitert werden. `Reg[A]` und `Eingabe` sind natürliche Zahlen echt größer Null.

3. Überlegen Sie sich einen Programmablauf in Pseudocode. 2 Punkte

4. Implementieren Sie den Befehl als Mikroprogramm für obige CPU.

   Verwenden Sie die folgende Tabelle. Leer gelassene Steuerleitungen entsprechen dem Wert „0", bei Sprüngen muss die Zieladresse explizit angegeben werden! Das niederwertigste Bit des Sprungziels entspricht Steuerleitung 21.

   Eine Fehlerbehandlung ist nicht nötig.

   (Die Tabellenlänge entspricht nicht der erwarteten Mikroprogrammlänge!) 8 Punkte

Adr.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	Kommentar
0																							
1																							
2																							
3																							
4																							
5																							
6																							
7																							
8																							
9																							
10																							
11																							
12																							
13																							
14																							
15																							
16																							
17																							
18																							
19																							
20																							

Aufgabe 3: Paging — 16 Punkte

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

- zweistufige Adresstabellen
- je 1024 Einträge zu je 4 Bytes in den Tabellen
  - Bit 31-12: höherwertige Bits der physikalischen Adresse
  - Bit 11-4: unbenutzt
  - Bit 3: Cache-disable-Bit
  - Bit 2: Execute-enable-Bit
  - Bit 1: Write-enable-Bit
  - Bit 0: Present-Bit
- Pages zu je 4 KiB Größe

Folgendes ist für ein Programm erfüllt:

- im virtuellen Adressbereich `0x00 03 00 00` bis `0x00 03 ff ff` kann auf nicht ausführbare Daten zugegriffen werden,
- im virtuellen Adressbereich `0x00 40 00 00` bis `0x00 4f ff ff` ist der Speicher der Grafikkarte eingeblendet,
- im virtuellen Adressbereich `0x00 e0 00 00` bis `0x00 e0 0f ff` kann auf schreibgeschützten Programmcode zugegriffen werden,
- im virtuellen Adressbereich `0xc0 00 00 00` bis `0xcf ff ff ff` ist Betriebssystemcode eingeblendet, der mit anderen Prozessen geteilt wird,
- alle anderen Bereiche sollen Zugriffsfehler auslösen.

Die Seiten liegen ab der physikalischen Adresse `0x00 c0 00 00` im Arbeitsspeicher.

Hinweis:
Es ist nicht nötig, die Tabellenhierarchie zu zeichnen! Ergebnisse dürfen hexadezimal angegeben werden.

1. Welche Flags müssen für welche Bereiche in den Page-Tables (zweite Hierarchiestufe) gesetzt werden?

   Geben sie die Codierung an, wie sie in den Verwaltungstabellen verwendet wird. 2 Punkte

2. Berechnen Sie, wie viele Seiten für die Verwaltungsinformationen benötigt werden! 4 Punkte

3. Wieviele Seiten *im Arbeitsspeicher* werden für die Nutzdaten benötigt? 3 Punkte

4. Gegeben seien die folgenden Adressumrechnungen *virtuell* → *physikalisch*:

   `0x00 e0 00 04` → `0x00 c5 50 04`
   `0xc1 00 00 4c` → `0x01 90 00 4c`
   `0xc1 00 01 4c` → `0x01 90 01 4c`
   `0x00 03 b1 28` → `0x00 c5 01 28`
   `0x00 44 1c cc` → `0x00 0e 1c cc`

   Skizzieren Sie einen vollassoziativen TLB mit 4 Einträgen nach dem Zugriff auf die obigen virtuellen Adressen.

   Der TLB sei zu Beginn leer. Ignorieren Sie die Ersetzungsstrategie! 5 Punkte

5. Was muss beim Prozesswechsel auf einem System mit Paging und TLB gemacht werden, damit der Speicherschutz nicht gefährdet ist? 2 Punkte

Aufgabe 4: Assembler — 14 Punkte

1. Nennen Sie zwei Formen der Parameterübergabe bei Unterprogrammaufrufen und geben Sie jeweils einen Vorteil an. 2 Punkte

2. Ein Programm möchte einen verfügbaren Bereich des Arbeitsspeichers als Stack verwalten. Die verwendete 32-Bit-CPU unterstützt jedoch keine expliziten Stackoperationen.

   Bilden Sie die Operationen `push value` und `pop address` nach. Zur Verfügung stehen arithmetische Operationen sowie die in der Vorlesung behandelten Adressierungsarten.

   Der Stackpointer soll jeweils auf den Anfang des obersten Elements im Speicher zeigen, der Stack wachse Richtung der Adresse `0`. 4 Punkte

3. Gegeben sei das folgende Programm:

```
1  .text
2  a:
3    movl 4(%esp), %eax
4    incl %eax
5    pushl %eax
6    call b
7    addl $4, %esp
8    ret
9  b:
10   incl %eax
11   ret
12 main:
13   pushl $0x42
14   call a
15   addl $4, %esp
16   movl $0, %eax
17   ret
```

Ein Befehl ist wie folgt aufgebaut (AT&T-Syntax):

Assemblerbefehl	Operanden	Bedeutung
`add`	`$4 ,%esp`	`Reg[esp]=Reg[esp]+4`

Führen Sie die Funktion <main> aus.

Skizzieren Sie dabei den Aufbau des Stacks

- jeweils nach Ausführung einer `call`-Instruktion, also bevor die erste Instruktion der aufgerufenen Funktion ausgeführt wurde,
- sowie vor Auführung der Instruktion 17: `ret`.

Geben Sie jeweils den bekannten und gültigen Stackinhalt und den aktuellen Stackpointer an!

Der Stack wachse in Richtung der Adresse `0x00 00 00 00`. Der Stackpointer `%esp` zeige immer auf den Anfang des letzten Eintrags und hat zu Beginn der Funktion `main` den Wert `0xff ff d2 cc`. Verwenden Sie die Zeilennummern als Rücksprungadresse!

Alle Adressen und Daten auf dem Stack sind 32 Bit breit.

(Bearbeitung auf der nächsten Seite!) 8 Punkte

Fortsetzung von Aufgabe 4.3.

Aufgabe 5: Parallelverarbeitung — 14 Punkte

1. Was versteht man unter Pipelining von Befehlen in der CPU? 2 Punkte

Gegeben sei eine Architektur mit einer fünfstufigen Pipeline, bestehend aus *Befehl holen und decodieren* (BH-BD), *Operanden holen* (OH), *Befehl ausführen (BA1, BA2)*, *Ergebnis sichern* (ES):

BH-BD	OH	BA1	BA2	ES

2. Wie hoch ist der maximale Speedup obiger Pipeline?

   Warum wird er im Allgemeinen in der Realität nicht erreicht? Nennen Sie die entsprechenden Fachbegriffe! 4 Punkte

3. Führen Sie das folgende Programm auf obiger CPU aus:

```
1  Lstart:
2    movl $2, %eax
3    movl $1, %ecx
4  Lcond:
5    cmpl %eax, %ecx
6    je Lend
7    jg Lgreater
8  Lless:
9    subl %ecx, %eax
10   jmp Lcond
11 Lgreater:
12   subl %eax, %ecx
13   jmp Lcond
14 Lend:
```

Ein Befehl ist wie folgt aufgebaut (AT&T-Syntax):

Assemblerbefehl	Operanden	Bedeutung
`subl`	`%eax ,%ecx`	`Reg[ecx]=Reg[ecx]-Reg[eax]`

Die Pipeline verfüge über keine erweiterten Mechanismen wie Sprungvorhersage, spekulative Ausführung, Forwarding, etc. Um dennoch ein korrektes Ergebnis zu garantieren, soll statt dessen die Ausführung weiterer Instruktionen so lange verzögert werden, bis kein Konflikt mehr vorliegt, allerdings auch nicht länger. Felder mit `nop` können leer gelassen werden.

Bei Sprungbefehlen soll der Befehlszähler so früh wie möglich aktualisiert werden. Jeder Befehl muss die gesamte Pipeline durchlaufen.

Tragen Sie in der Tabelle auf der nächsten Seite in jeder Stufe *die Instruktion und die Zeile* des dort gerade ausgeführten Befehls ein (z.B. `subl12`). Geben Sie außerdem den aktuellen Registerinhalt von `eax` und `ecx` an, soweit bekannt.

Unbedingte Spünge werden bereits in `BH-BD` ausgewertet. Der Befehl `cmpl` aktualisiert das Statusregister zur Sprungauswertung erst in der Phase *ES*, bedingte Sprünge werten es in der *OH*-Phase aus.

Lassen Sie am Ende die Pipeline leer laufen.

Die Tabellenlänge entspricht nicht der erwarteten Ausführungszeit! 8 Punkte

Takt	BH-BD	OH	BA1	BA2	ES	`eax`	`ecx`
0	movl2					?	?
1	movl3	movl2				?	?
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25							
26							
27							
28							
29							
30							
31							
32							
33							
34							

Aufgabe 6: Cache — 12 Punkte

1. Was versteht man unter räumlicher und zeitlicher Lokalität typischer Programme? 2 Punkte

2. Eine CPU besitze einen Cache mit 256 KiB Nutzdaten mit einer Blockgröße von 64 Bytes. Wie viele Blöcke und Mengen besitzt der Cache, wenn er 8-fach assoziativ verwaltet wird? 2 Punkte

3. Wie verändert sich der Realisierungsaufwand bezüglich Anzahl und Breite der Adressvergleicher im Cache, wenn ein Assoziativitätsgrad von 256 statt 8 gewählt würde?

   Begründen Sie! 2 Punkte

4. Eine 32-Bit CPU verwende einen 4-fach assoziativen Cache mit insgesamt 8 Blöcken. Jeder Block umfasse 8 Bytes. Der Cache sei zu Beginn leer. Es wird jeweils der Eintrag verdrängt, der am längsten nicht verwendet wurde (LRU).

   Ein Programm liest nacheinander *jeweils 4 Bytes* von den folgenden Adressen:

   ```
   0x00 cd 10 50
   0x00 cd 10 58
   0x00 cd 10 60
   0x01 00 00 00
   0x01 00 00 10
   0x00 cd 10 78
   0x00 cd 10 74
   0x00 cd 10 70
   ```

   Der relevante Speicherbereich hat folgenden Inhalt:

   ```
   0x00 cd 10 50: ca 01 6b 7f ff 00 10 bc  bf 41 45 e8 00 a7 56 00
   0x00 cd 10 60: e8 00 a4 22 ff ff 50 bf  40 b0 c7 00 a3 05 21 21
   0x00 cd 10 70: 57 41 56 41 55 41 54 41  89 55 53 fd 89 48 48 f3
   ...
   0x01 00 00 00: a8 64 ff ff 89 48 e8 df  a9 1c ff ff 8b 48 ed 3d
   0x01 00 00 10: 32 4b e1 75 fb c1 11 09  7b 80 2e 01 94 0f ff 10
   ...
   ```

   Welchen Inhalt hat der Cache nach Ausführung des Programms?

   Es genügt, jeweils das erste und letzte Byte der Daten im Cache-Block einzutragen. 6 Punkte

Menge	Valid	Tag	Daten
0			
1			

Aufgabe 7: Arbeitsspeicher — 10 Punkte

1. Was unterscheidet dynamischen RAM (DRAM) von statischem RAM (SRAM)?

   Worin liegt jeweils der Nachteil? — 2 Punkte

2. Warum sollten aufeinanderfolgende Datenworte auf unterschiedlichen Speicherbänken abgelegt werden?

   Wie nennt sich dieses Vorgehen? — 2 Punkte

3. Beschreiben Sie in Stichpunkten, was jeweils zwischen zwei aufeinanderfolgenden Speichergenerationen von einfachem SDRAM bis DDR3-SDRAM geändert wurde und wie sich dadurch der Durchsatz geändert hat!

   Wie hat sich in dieser Zeit der Takt der eigentlichen Speicherzellen verändert? — 6 Punkte

Zusätzlicher Platz

Zusätzlicher Platz