DEPARTMENT INFORMATIK
Lehrstuhl für Informatik 3 (Rechnerarchitektur)
Friedrich-Alexander-Universität Erlangen-Nürnberg
Martensstraße 3, 91058 Erlangen
13.04.2012

Klausur zu

Grundlagen der Rechnerarchitektur und -organisation

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

- Es sind *keine* Hilfsmittel erlaubt!
- 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!
- Begründen Sie Ihre Antworten!

Durch meine Unterschrift bestätige ich

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

Erlangen, den 13.04.2012 .....
(Unterschrift)

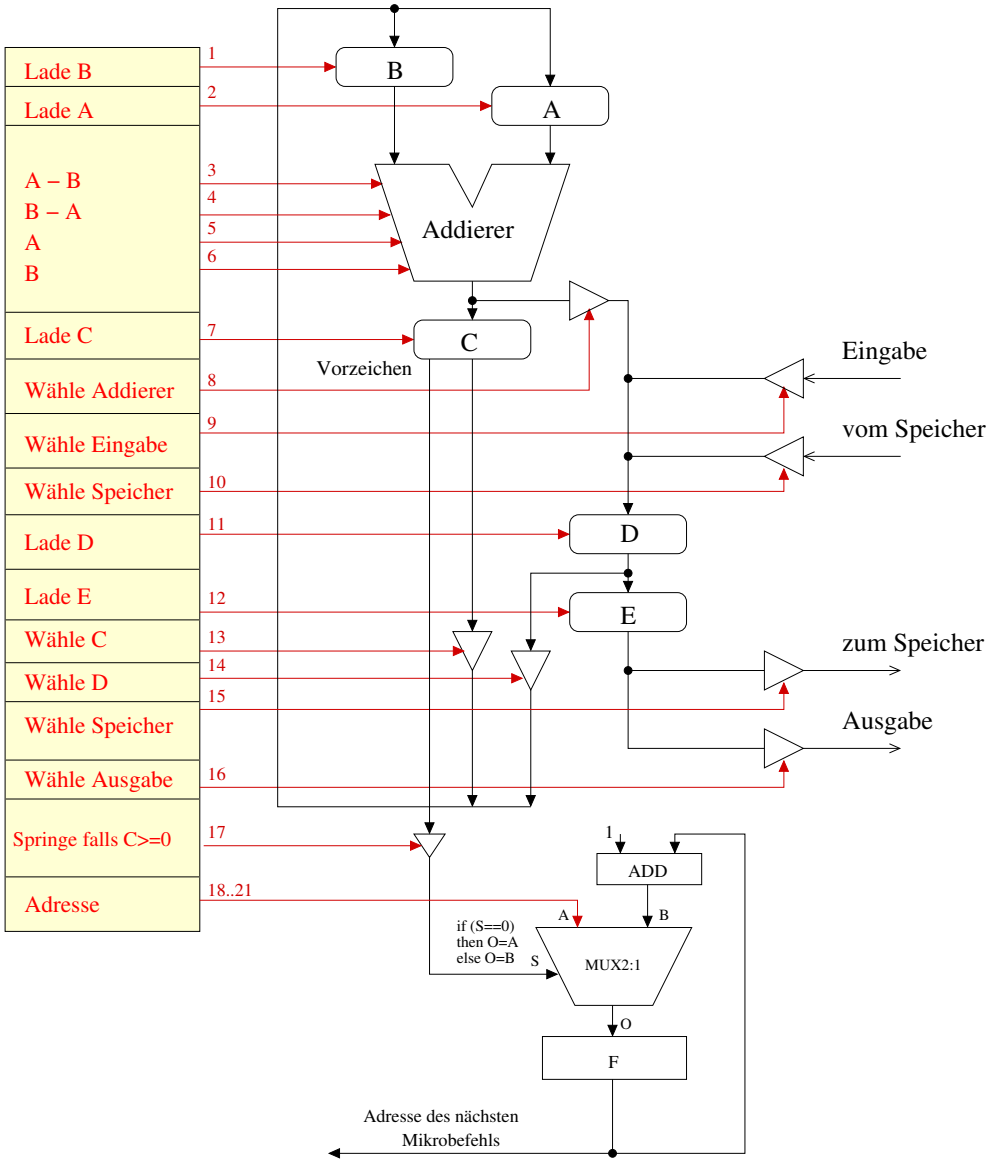
Ich bin damit einverstanden, dass mein Prüfungsergebnis der Klausur unter Angabe der Matrikel-Nummer veröffentlicht wird.

Erlangen, den 13.04.2012 .....
(Unterschrift)

Aufgabe	1	2	3	4	5	6	Bonus	Summe
max. Punktzahl	12	14	19	10	16	19	15	105
erreichte Punktzahl								

Aufgabe 1: Mikroprogrammierung (12 Punkte)

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



Hinweis: Am Ende der Klausur finden Sie eine Kopie der Angabe, die Sie zur Bearbeitung heraustrennen dürfen.

Wie üblich haben die Register eine Verzögerung von einem Takt und sind zu Beginn leer. Das Register F entspricht dem Mikrobefehlszähler und zeigt zunächst auf die erste Ihrer Instruktionen.

1. Beschreiben Sie grob, aus welchen Teilen ein Mikrobefehl für die obige Architektur besteht. Inwiefern kann vom sequentiellen Programmablauf abgewichen werden? (2 Punkte)

2. Schreiben Sie ein Mikroprogramm, das jeweils zwei Zahlen von der Eingabe liest und der Größe nach absteigend sortiert wieder an die Ausgabe zurückgibt.

   Erstellen Sie die Funktion zunächst in Pseudocode, der sich möglichst leicht als Mikroprogramm umsetzen lässt.

   Nutzen Sie für das Mikroprogramm die Tabelle auf der nächsten Seite. Die letzte Spalte dient lediglich der Orientierung. Alle nicht explizit gesetzten Steuerleitungen gelten als deaktiviert.

   Hinweis: Sortieren Sie zunächst die Eingabe. (6 Punkte)

Befehls-adresse	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18–21 Adresse	Erklärung
0																			
1																			
2																			
3																			
4																			
5																			
6																			
7																			
8																			
9																			
10																			
11																			
12																			
13																			
14																			
15																			

3. Wie hoch ist der Speicherbedarf Ihrer Lösung (in Bit)? (1 Punkt)

4. Wodurch kann unter Beibehaltung der Funktionalität der Speicherbedarf für das Mikroprogramm verringert werden? Wie nennt sich diese Art der Mikroprogrammierung? Was ist ihr Nachteil? (3 Punkte)

Aufgabe 2: Adressierungsarten (14 Punkte)

Folgendes Assemblerprogramm soll auf einer geeigneten CPU ausgeführt werden.

```
1  fib:     file format elf32-i386
2
3  Disassembly of section .text:
4
5  0x08048080 <_start>:
6   0x08048080:    mov    eax,0x80490d0 <array>
7   0x08048085:    push   eax
8   0x08048086:    mov    eax,0x3
9   0x0804808b:    push   eax
10  0x0804808c:    call   0x8048098 <fib>
11  0x08048091:    pop    eax
12  0x08048092:    pop    eax
13  0x08048093:    jmp    0x80480c1 <Lend>
14
15 0x08048098 <fib>:
16  0x08048098:    mov    ebx,[esp+0x8 ]
17  0x0804809c:    mov    ecx,0x2
18  0x080480a1:    xor    edx,edx
19  0x080480a3:    mov    [ebx],edx
20  0x080480a5:    mov    eax,0x1
21  0x080480aa:    mov    [ebx+0x4],eax
22  0x080480ad:    jmp    0x080480ba <Lloop_cond>
23
24 0x080480b2 <Lloop>:
25  0x080480b2:    push   eax
26  0x080480b3:    add    eax,edx
27  0x080480b5:    mov    [ebx+ecx*4],eax
28  0x080480b8:    pop    edx
29  0x080480b9:    inc    ecx
30
31 0x080480ba <Lloop_cond>:
32  0x080480ba:    cmp    ecx,[esp+0x4]
33  0x080480be:    jle    0x080480b2 <Lloop>
34  0x080480c0:    ret
35
36 0x080480c1 <Lend>:
37  0x080480c1:    mov    ebx,0x0  ; Exit code 0 = success
38  0x080480c6:    mov    eax,0x1  ; Select Systemcall exit
39  0x080480cb:    int    0x80     ; Systemcall
40
41 Disassembly of section .bss:
42
43 0x080490d0 <array>:
44         ...
```

Hinweis: Am Ende der Klausur finden Sie eine Kopie der Angabe, die Sie zur Bearbeitung heraustrennen dürfen.

Die erste Spalte enthält jeweils die Adresse im Speicher. Das Label `array` zeigt auf einen ausreichend großen Speicherbereich zur Ablage von Daten. Das Segment `.text` beinhaltet den Programmcode, `.bss` mit `0` vorinitialisierte Daten.

1. Analysieren Sie obiges Programm.

   Beschreiben Sie kurz, was das Programm bewirkt (*nicht* die einzelnen Instruktionen). (2 Punkte)

2. Vervollständigen Sie die Tabelle.

   Tragen Sie jeweils die Anzahl der Speicherzugriffe während der Ausführung des bereits geladenen Befehls in der entsprechenden Zeile ein. Beschreiben Sie außerdem jeweils kurz die Aufgabe des Befehls im Programm (nicht lediglich seine Bedeutung!). (6 Punkte)

Zeile	Speicherzugriffe	Bedeutung
6		
7		
10		
16		
27		
34		

3. Welcher Klasse von Befehlssatzarchitekturen ist eine CPU zuzuordnen, die obiges Assemblerprogramm ausführen kann?

   Handelt es sich eher um eine CISC- oder RISC-Architektur?

   Belegen Sie Ihre Antworten am gegebenen Assemblerprogramm. (3 Punkte)

4. Skizzieren Sie den Aufbau des Stack genau zu Beginn der Ausführung des Unterprogramms `fib` in Zeile 16. Der Stackpointer habe zu diesem Zeitpunkt den Wert `0xffffdb94`.

   Geben Sie zu jeden Eintrag die Adresse (entspricht dem Wert des ehemaligen Stackpointers), den Speicherinhalt und seine Bedeutung an. (3 Punkte)

Aufgabe 3: Cache (19 Punkte)

1. Warum ist der Einsatz von Caches generell sinnvoll? Welche Eigenschaften typischer Programme liegen der Geschwindigkeitssteigerung zugrunde? Was ist deren Ursache im Programm? (3 Punkte)

2. Welche Organisationsformen für Caches wurden in der Vorlesung behandelt? Inwiefern unterschieden sie sich hinsichtlich der ,,3 Cs"? (3 Punkte)

3. Ein ,,Direct-Mapped"-Cache habe die Größe 2^{20} Bytes. Die Adressen im Rechner seien 32 Bit groß. Pro Eintrag werden 64 Bytes Nutzdaten gespeichert. Wie teilt sich die Adresse auf Index, Tag und Byte-Adresse auf? (3 Punkte)

Gegeben sei folgendes Assemblerprogramm:

```
1     mov eax, 0
2     mov [2000], 1
3     mov [2004], 1
4     mov ebx, 2000
5  loop:
6     mov ecx, [ebx]
7     add ecx, [ebx+4]
8     mov [ebx+8], ecx
9     add ebx, 4
10    add eax, 1
11    cmp eax, 10
12    jne loop
```

Hinweis: Am Ende der Klausur finden Sie eine Kopie der Angabe, die Sie zur Bearbeitung heraustrennen dürfen.

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

4. Wie lange benötigt das Programm zur Abarbeitung, wenn keine Caches verwendet werden? (4 Punkte)

5. Der Prozessor verfüge nun über einen gemeinsamen Vollassoziativ-Cache mit insgesamt 9 Einträgen (B0–B8) für Code und Daten. Pro Block kann ein Befehl bzw. Datum (4 Byte) abgelegt werden. Der Cache sei zu Beginn leer. Die Schreibstrategie sei „Write-Back". Bei Verdrängung wird das älteste Datum im Cache überschrieben.

   - Tragen Sie den Inhalt des Caches nach Ausführung der Instruktionen in Zeile 6, 8, 10 und 12 im initialen Schleifendurchlauf in die Tabelle ein. Falls nötig wird der Eintrag verdrängt, auf den am längsten nicht zugegriffen wurde (LRU).

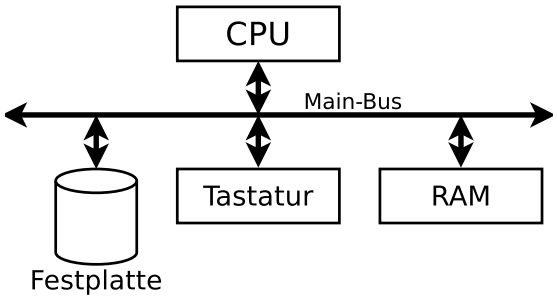
     Es ist ausreichend, jeweils die Zeile des Befehls bzw. die Anfangsadresse des Datums im Cache einzutragen.

     (5 Punkte)

	6–Iteration 0	8–Iteration 0	10–Iteration 0	12–Iteration 0
B0				
B1				
B2				
B3				
B4				
B5				
B6				
B7				
B8				

   - Wie beurteilen Sie die Effizienz des Caches? (1 Punkt)

Aufgabe 4: I/O (10 Punkte)



Gegeben sei eine CPU, die über einen gemeinsamen Bus mit dem Arbeitsspeicher und angeschlossenen Geräten kommunizieren kann.

1. Welche drei verschiedenen Möglichkeiten für die CPU gibt es im Allgemeinen, Ein-/Ausgabe von Daten durchzuführen? (1,5 Punkte)

2. Es sollen Daten von einem I/O-Gerät gelesen und in den Speicher geladen werden.

   Wie läuft bei den oben genannten Möglichkeiten die Kommunikation zwischen den beteiligten Komponenten jeweils im Detail ab? (4,5 Punkte)

3. Welche der genannten Methoden ist am besten geeignet, wenn

   - die CPU ein Zeichen von der Tastatur einlesen will, und

   - ein 512 Byte großer Block auf die Festplatte geschrieben werden soll?

   Geben Sie jeweils eine kurze Begründung an. (2 Punkte)

4. Welche grundlegenden Adressierungsmodi für angeschlossene Peripherie gibt es? Erläutern Sie jeweils Vor- und Nachteile.

   Inwiefern spielen hier Caches eine Rolle? (2 Punkte)

Aufgabe 5: MMU (16 Punkte)

Auf einer gegebenen Architektur werde Paging zur Speicherverwaltung verwendet. Ein auszuführendes Programm brauche folgende virtuellen Speicherbereiche:

```
0x08000000 - 0x080ff0ea Textsegment
0x20000000 - 0x20100123 Datensegment
0x40000000 - 0x400ff9af Library
0xbffff120 - 0xbfffffff Stack
```

Der Rechner biete 4 KiB große Pages.

Hinweis: $4096 = 2^{12}$, $1024 = 2^{10}$

1. Wieviele Pages benötigt das Programm jeweils für das Text-, Daten-, Library- und Stack-Segment? Begründen Sie Ihre Antwort! (5 Punkte)

   Hinweis: Es ist nicht nötig, die Tabellenhierarchie zu zeichnen.

2. Wieviele Pages benötigt das Programm für die Page-Tabellen? Begründen Sie Ihre Antwort! (5 Punkte)

   Hinweis: Die Page-Tabellen seien zweistufig. Sowohl die Page-Group-Tabelle als auch die Page-Tabelle enthalten jeweils 1024 Einträge und sind je 4 KiB groß.

3. Wie tief muss die Seitentabellenhierarchie auf einem 64-Bit System mit 48 Bit breiten physikalischen Adressen sein, wenn die Page-Größe von 4 KiB beibehalten wird?

   Geben Sie die genaue Aufteilung der virtuellen Adresse an. (2 Punkte)

   Hinweis: Ein Eintrag in den Verwaltungstabellen ist jeweils 64 Bit breit.

4. Welche Informationen müssen in einem Translation Lookaside Buffer (TLB) gespeichert werden, um den Zugriff auf die Tabellenhierarchie vermeiden zu können?

   Bedenken Sie, dass Seiten ggf. vor Ausführung, Überschreiben, ... geschützt sein könnten. (2 Punkte)

5. Welche Konstellation aus TLB- und Cache-Hit bzw. Miss ist möglich? Der Cache befinde sich aus Sicht der CPU nach der MMU. (2 Punkte)

TLB	Cache	möglich	Begründung
Hit	Hit		
Hit	Miss		
Miss	Hit		
Miss	Miss		

Aufgabe 6: Parallelverarbeitung (19 Punkte)

In der Vorlesung wurden im Rahmen der „Evolution“ moderner Architekturen verschiedene Ebenen der parallelen Verarbeitung vorgestellt.

1. Nennen Sie drei Formen der Parallelität in aktuellen CPUs und erklären Sie jeweils, was parallel ausgeführt wird und wie sie sich dem Programmierer präsentiert. (6 Punkte)

2. Welche Veränderungen hinsichtlich Modifikation, bzw. Vervielfältigung von Teilwerken des von Neumann'schen Universalrechenautomaten werden zur Realisierung dieser Formen jeweils durchgeführt? (6 Punkte)

3. Vergleichen Sie die Formen hinsichtlich des Realisierungsaufwandes. (1 Punkt)

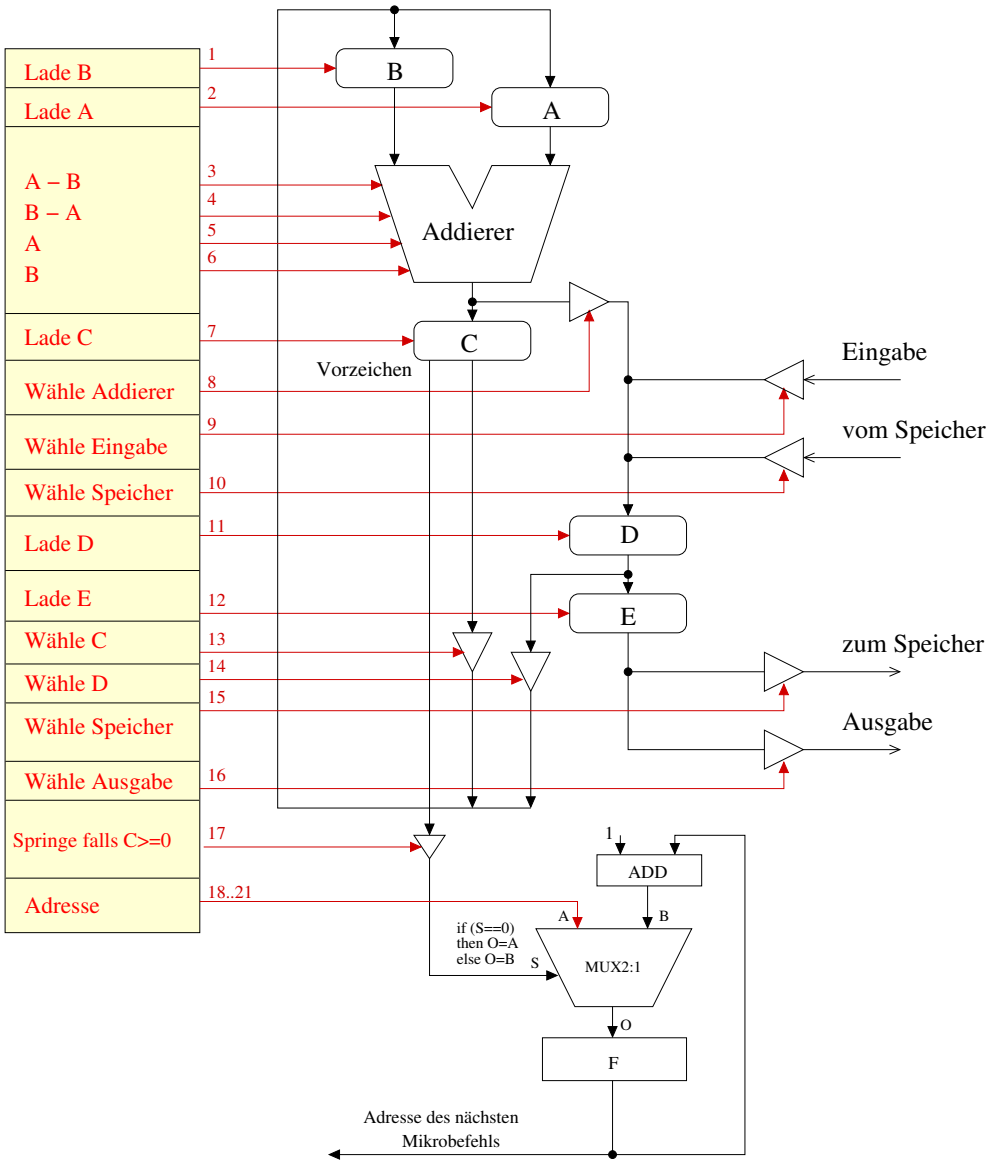
4. Leiten Sie den asymptotischen Speedup von Pipelining aus der Anzahl auszuführender Befehle n und Pipelinestufen p her. (3 Punkte)

5. Was besagt die Regel von Pollack?

   Wie lässt sich daraus ein Geschwindigkeitsvorteil einer Mehrkernarchitektur gegenüber eines Einzelkern-Prozessors mit gleichem Ressourcenverbrauch ableiten? (3 Punkte)

Zusätzlicher Platz

Zusätzlicher Platz


Lade B
Lade A
A – B
B – A
A
B
Lade C
Wähle Addierer
Wähle Eingabe
Wähle Speicher
Lade D
Lade E
Wähle C
Wähle D
Wähle Speicher
Wähle Ausgabe
Springe falls C>=0
Adresse
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18..21
B
A
Addierer
C
Vorzeichen
Eingabe
vom Speicher
D
E
zum Speicher
Ausgabe
1
ADD
A
B
if (S==0)
then O=A
else O=B
S
MUX2:1
O
F
Adresse des nächsten
Mikrobefehls

```
1  fib:     file format elf32-i386
2
3  Disassembly of section .text:
4
5  0x08048080 <_start>:
6   0x08048080:    mov    eax,0x80490d0 <array>
7   0x08048085:    push   eax
8   0x08048086:    mov    eax,0x3
9   0x0804808b:    push   eax
10  0x0804808c:    call   0x8048098 <fib>
11  0x08048091:    pop    eax
12  0x08048092:    pop    eax
13  0x08048093:    jmp    0x80480c1 <Lend>
14
15 0x08048098 <fib>:
16  0x08048098:    mov    ebx,[esp+0x8 ]
17  0x0804809c:    mov    ecx,0x2
18  0x080480a1:    xor    edx,edx
19  0x080480a3:    mov    [ebx],edx
20  0x080480a5:    mov    eax,0x1
21  0x080480aa:    mov    [ebx+0x4],eax
22  0x080480ad:    jmp    0x080480ba <Lloop_cond>
23
24 0x080480b2 <Lloop>:
25  0x080480b2:    push   eax
26  0x080480b3:    add    eax,edx
27  0x080480b5:    mov    [ebx+ecx*4],eax
28  0x080480b8:    pop    edx
29  0x080480b9:    inc    ecx
30
31 0x080480ba <Lloop_cond>:
32  0x080480ba:    cmp    ecx,[esp+0x4]
33  0x080480be:    jle    0x080480b2 <Lloop>
34  0x080480c0:    ret
35
36 0x080480c1 <Lend>:
37  0x080480c1:    mov    ebx,0x0  ; Exit code 0 = success
38  0x080480c6:    mov    eax,0x1  ; Select Systemcall exit
39  0x080480cb:    int    0x80     ; Systemcall
40
41 Disassembly of section .bss:
42
43 0x080490d0 <array>:
44         ...
```

```
1      mov eax, 0
2      mov [2000], 1
3      mov [2004], 1
4      mov ebx, 2000
5  loop:
6      mov ecx, [ebx]
7      add ecx, [ebx+4]
8      mov [ebx+8], ecx
9      add ebx, 4
10     add eax, 1
11     cmp eax, 10
12     jne loop
```

Anhang zu Aufgabe 3 — Diese Seite dürfen Sie heraustrennen und behalten!