Informationstechnik
Klausur WS 2015/2016

Institut für Technik der Informationsverarbeitung – ITIV
Prof. Dr.-Ing. E. Sax

Informationstechnik
Datum: 02.03.2016
Name:
Matrikelnummer:
ID #:

Hörsaal:

Platznummer:

Hinweise zur Klausur

Hilfsmittel

- Erlaubte Hilfsmittel sind Lineal und eine Formel-/Notizensammlung, handgeschrieben DIN A4 Blatt zweiseitig beschrieben.
- Verwenden Sie zum Bearbeiten der Aufgaben nur dokumentenechte Schreibgeräte – keinen Bleistift, keine Rotstifte!
- Alle nicht genannten Hilfsmittel sind untersagt. Dies beinhaltet auch jegliche Kommunikation mit Anderen.

Prüfungsdauer

Die Prüfungsdauer beträgt 120 Minuten.

Prüfungsunterlagen

Die Prüfungsunterlagen bestehen aus insgesamt **34** Seiten Aufgabenblättern (dieses Titelblatt, **8** Aufgabenblöcke, 1 zusätzliches Lösungsblatt). Geben Sie immer volle und nachvollziehbare Lösungs- und Rechenwege an.

Bitte kontrollieren Sie vor der Bearbeitung der Aufgaben auf jeder Seite oben Ihren Namen und Ihre Matrikelnummer!

Falls Sie zusätzliche Blätter zur Lösung der Aufgaben benötigen, verwenden Sie das zusätzliche Lösungsblatt bzw. fragen Sie nach zusätzlichem Papier. Auf jedes zusätzliche Lösungsblatt ist neben dem Namen auch die Aufgabennummer mit einzutragen. Vermeiden Sie das Beschreiben der Rückseiten. Die Verwendung von eigenen Blättern ist nicht erlaubt.
In den letzten 30 Minuten der Klausur ist eine vorzeitige Abgabe der Klausur nicht möglich. Am Ende der Prüfung bleiben Sie bitte sitzen. Alle Aufgaben- und Lösungsblätter sowie alle zusätzlichen Lösungsblätter sind in die ausgehändigten Umschläge zu geben. Diese werden von der Aufsicht eingesammelt.

Aufgabe	1	2	3	4	5	6	7	8	Σ
≈ Gewichtung ca. [%]	15	12	14	12	12	12	12	12	100
Ergebnis [P]									

Aufgabe 1 Allgemeine Fragen

A) Was versteht man unter der Abkürzung ALU im Gebiet der Rechnerarchitektur?

B) Geben Sie für folgende Aussagen an, ob diese wahr oder falsch sind.

Bei falscher Antwort gibt es Punktabzug. Die Aufgabe wird minimal mit 0 Punkten bewertet.

	Wahr	**Falsch**
Die Von Neumann-Architektur hat getrennte Speicher für Programm-Daten und Daten-Daten. Der gleichzeitige Zugriff auf Speicher ist unmöglich.		
Die Verwendung von Polymorphie erlaubt die Definition von Funktionen die für dieselben Eingabevariablen unterschiedliche Ausgabevariablen zurückliefern.		
Der Sortieralgorithmus Quick-Sort beruht auf dem „Teile und Herrsche“ Prinzip.		
Der Optimierungsalgorithmus „Simulated Annealing“ ist nicht deterministisch.		
Eine Referenz ist eine veränderbare Variable, deren Inhalt die Adresse einer anderen Variablen im Speicher enthalten kann.		
Der Operator `||` verknüpft zwei Operanden und gibt `true` zurück, wenn beide Operanden den Wert `true` haben, sonst `false`.		
Wenn Breiten- und Tiefensuche auf denselben Graphen angewendet werden, liefern sie immer denselben Ergebnisbaum.		
In einer verketteten Liste kann über einen Index direkt auf die einzelnen Elemente zugegriffen werden.		

Tabelle 1-1: Allgemeines

C) Hängt der Speicherbedarf einer Zeigervariablen vom Datentyp ab, auf den sie verweist? Begründen Sie Ihre Antwort.

D) Welche Vor- und Nachteile hat die Datenstruktur verkettete Liste gegebenüber einem Array? Erläutern Sie je einen Vor- und Nachteil.

E) Kann es vorkommen, dass Sie beim mehrmaligen Ausführen des Optimierungsalgorithmus Simulated Annealing verschiedene Ergebnisse erhalten? Begründen Sie Ihre Antwort.

F) Nennen Sie vier typische Eigenschaften von objektorientierten Sprachen.

G) Beschreiben Sie kurz jeweils die Aufgabe der Compiler, Linker und Debugger.

H) Definieren Sie den Begriff "Write through" in Bezug auf Caches und nennen Sie einen Vor- und einen Nachteil.

Definition:

Vorteil:

Nachteil:

Aufgabe 2 C++ Verständnisfragen

A) Welchen Wert hat die Variable x nach dem Ausführen der folgenden Codesequenz?

```
int i = 5, x = 0;
do{
    x += --i;
} while (i > 0);
```

// x =

B) Entscheiden Sie sich im Folgenden für den jeweils sinnvollen C++ Datentyp und achten Sie dabei auf Speichereffizienz (Datentyp int: 4 Byte):

	`messwert1 = -1000; // -1000 bis +1000`
	`zeichen = 'a'; // 'a' - 'z'`
	`PI = 3.141; // Konstante`
	`name = 'Mustermann';`
	`messwert2 = 1.234; // -2.0 bis +2.0`
	`laengeDatenarray = 50000; // 0 bis 65000`

Tabelle 2-1 C++ Datentypen

C) Geben Sie den Wert der Variablen `n` und des Strings `s` nach der Ausführung der jeweiligen Anweisung an.

```
string s("Treffen heute");
int n = s.find(" ");
```

// n =

```
s.replace(n - 1, 6, "punktwolke", 0, 5);
```

// s =

```
s.insert(6, "Treffen", 1, 1);
```

// s =

D) Gegeben ist das folgende fehlerhafte C++ Programm. Markieren Sie **fünf** Fehler.

```
#include <iostream>
using namespace std;

int main()
{
    double u; i; r;
    cout << "Berechnung eines Ohmschen Widerstandes\n";
    cout << "aus der Spannung U und dem Strom I\n";
    cout << "Spannung U in Volt = ";
    cin >> U;
    cout << "Strom I in Ampere = ";
    cin >> I;

    if (i = 0)
    {
     cout << "Unendlicher Widerstand (durch 0 teilen ist nicht
möglich).";
    }
    else if
    {
        r = u / i;
        cout << "Ohmscher Widerstand = " << r << " Ohm";
    }
    return 0;
}
```

E) Was ist das Ergebnis des folgenden C++ Ausdrucks? Dabei sind die Variablen Z1 und Z2 vom Typ `bool`.

a) `Z1 = ((1 & 2) | (1 ^ 2));` ☐ true ☐ false

b) `Z2 = !(3 | 0) && ~(5 & 4);` ☐ true ☐ false

F) Was ist der Inhalt des Arrays `arr` nach dem Ausführen des folgenden Codes und welche Funktion erfüllt dieser?

```
int arr[4] = { 5, -6, 100, 0 };
for (int i = 0; i <= 2; i++)
{
    for (int j = (i + 1); j <= 3; j++)
    {
        if (*(arr + i) > arr[j])
        {
            int temp = *(arr + i);
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
}
```

// arr = ☐☐☐☐

Aufgabe 3 Objektorientierung in C++

A) Benennen und erklären Sie kurz **vier** Fehler im folgenden C++-Code:

```
#include <iostream>

using namespace std;

class Base {
private:
    int value = 10;
protected:
    void set_value(x) {
        value = x;
    }
    int get_value() const {
        return value;
    }
};

class Derived : public Base {
public:
    Derived(int x) {
        set_value(x * 10);
    }
    int get_value() const {
        return value * 10;
    }
};

int main() {
    Base b;
    cout << b.get_value();

    Derived d;
    cout << d.get_value();
    getchar();
    return 0;
}
```

B) Schreiben Sie eine Zeile C++-Code, …

1) um ein neues Objekt der Klasse `Regal` dynamisch anzulegen und weisen Sie dieses einer Variablen `neuesObjekt` der Oberklasse `Moebel` zu.

2) um den dynamisch allokierten Speicher aus Teilaufgabe 1) wieder freizugeben.

C) Erklären Sie den Unterschied zwischen **`private`** und **`protected`** in Bezug auf Zugriffsrechte in C++.

D) Gegeben sei folgender C++-Code:

```
#include <iostream>
#include <string>

using namespace std;

class Fahrzeug {
protected:
    int gesamt_masse; // kg
    int top_speed; // km/h
public:
    /*
    Hier fehlt der Code der Methoden get_masse(), get_speed() und
    get_typ() von Teilaufgabe i. und ii.
    */
    void print() {
        cout << "Gesamtmasse: " << get_masse() << " kg." << endl;
        cout << "Topspeed: " << get_speed() << " km/h." << endl;
        cout << "Typ: " << get_typ() << endl;
    }
};

int main() {
    Fahrzeug f;
    PKW pkw;
    LKW lkw;
    cout << f.get_typ() << endl;
    cout << pkw.get_typ() << endl;
    cout << lkw.get_typ() << endl;
    Fahrzeug& p = pkw;
    Fahrzeug& l = lkw;
    p.print();
    l.print();
    return 0;
}
```

i. Implementieren Sie in der Klasse `Fahrzeug` die Methoden `get_masse()` und `get_speed()`. Die Methoden sollen keinen Schreibzugriff auf Klassenattribute besitzen.

ii. Implementieren Sie in der Klasse `Fahrzeug` die Methode `get_typ()`, die den String „Fahrzeug“ zurückgibt. Leiten Sie außerdem von der Klasse `Fahrzeug` die zwei Klassen `PKW` und `LKW` ab. Setzen Sie dabei sinnvolle Werte für die Attribute `gesamt_masse` und `top_speed` bei der Initialisierung der Objekte. Für `PKW` soll die Methode `get_typ()` den String „PKW“ zurückgeben, für `LKW` analog „LKW“. Implementieren Sie in den beiden Kindklassen (`PKW` und `LKW`) die nötigen Methoden, so dass die Aufrufe der oben angegebenen `main()`-Funktion funktionieren.

iii. Geben Sie die Ausgabe des Programms an:

Aufgabe 4 Datenstrukturen & STL

Aufgabe 4.1 Implementierung

Programmieren Sie unter Zuhilfenahme der STL-Klasse `map` ein Telefonbuch. Dabei wird der Name als Schlüssel verwendet, während die Telefonnummer den Wert darstellt.

Das Header-File `phonebook.h` der Klasse `Phonebook` ist im Folgenden gegeben:

```
#ifndef PHONEBOOK_H
#define PHONEBOOK_H

#include <map>
#include <string>
#include <iostream>

using namespace std;

class Phonebook {
private:
 map<string, string> directory;

public:
 Phonebook();
 string searchNr( string name );
 void add( string name, string nr );
 int list_all();
};

#endif // !PHONEBOOK_H
```

A) Implementieren Sie die Klassen-Methode `add` zum Speichern eines neuen Schlüssel-Wert-Paares in der `map` der Klasse. Dabei stellt der übergebene Name den Schlüssel und die übergebene Nummer den Wert dar. Eine Überprüfung, ob der Eintrag bereits existiert soll nicht vorgenommen werden.

```
void Phonebook::add( string name, string nr ) {




}
```

B) Implementieren Sie die Klassen-Methode `searchNr`. Die Methode sucht in der `map` der Klasse nach dem übergebenen Namen und gibt bei Erfolg die Telefonnummer als `string` zurück. Wird der Name nicht gefunden gibt die Methode den String „`nicht gefunden`" zurück.

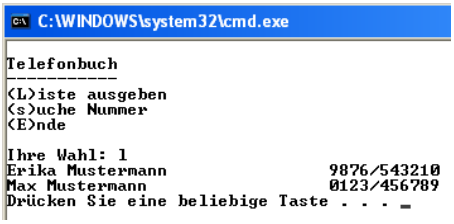
```
string Phonebook::searchNr( string name ) {




}
```

C) Implementieren Sie die Methode `list_all()` zum Auflisten des Inhalts des Telefonbuchs. Dabei soll die `map` mit einem **Iterator** vollständig durchlaufen werden. Die Einträge des Telefonbuchs sollen wie im Screenshot gezeigt in zwei Spalten erscheinen. Die Methode gibt als Rückgabewert die Anzahl der Einträge in der `map` zurück.

Hinweis: Die im Screenshot gezeigte Linksbündigkeit der ersten Spalte wird nicht verlangt!




```
int Phonebook::list_all() {

}
```

Aufgabe 4.2 Verständnisfragen zu STL

A) Das Telefonbuch der Klasse `Phonebook` aus Aufgabe Aufgabe 4.1 soll erweitert werden. Anstatt nur eine Telefonnummer zu einem Namen speichern zu können, sollen nun mehrere Nummern zu einem Schlüssel hinterlegt werden können.
Welcher Container-Typ der STL ist dafür besonders geeignet und wieso? Geben Sie zusätzlich die Definition des neuen `directory` an.

B) Erklären Sie den Unterschied des Funktionsprinzips bzgl. des Daten-/Elementzugriffs bei den beiden STL Containern `stack` und `queue`.

C) Die STL Container lassen sich in drei Typen klassifizieren. Zu welchem Typ zählt die `map`?

D) Nennen Sie zwei sequenzielle STL Container.

Aufgabe 5 Datei-/Stream-/Stringverarbeitung in C++

Es soll eine Auswertung über die Einwohnerzahl von bis zu 1000 Städten implementiert werden. Dazu wurde für jede Stadt eine Datei erzeugt (siehe Abbildung 5-1 links). Diese Dateien haben die durchgehenden Dateinamen „File001.txt“ bis maximal „File999.txt“ und enthalten jeweils nur eine Zeile. Diese Zeile beschreibt zuerst den Namen der Stadt, anschließend ob die Stadt eine Großstadt ist (0 = keine Großstadt, 1 = Großstadt) und als dritten Parameter die Einwohnerzahl. Die einzelnen Parameter sind dabei durch Semikolon getrennt. Um die Auswertung zu vereinfachen, sollen alle einzelnen Dateien ausgewertet und die Großstädte in eine separate Datei geschrieben werden (siehe Abbildung 5-1 rechts).

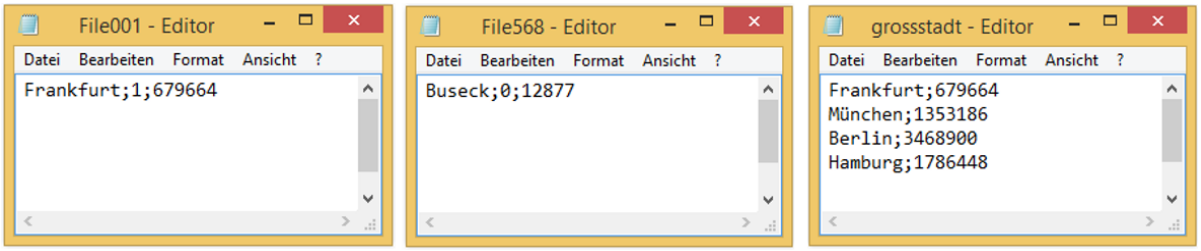


Abbildung 5-1: Beispiel-Dateien

Vorgegeben ist hierzu die folgende Klassenstruktur. Dabei sollen Sie die einzelnen Methoden entsprechend der folgenden Teilaufgaben implementieren.

Bitte beachten Sie, dass verwendete Variablen immer vorher deklariert werden müssen.

```
class CityFileHandling {
private:
        int numberOfCities;
        string intToStr( int zahl );
public:
        bool filePresent( string filename );
        void filesRead( string praefix, int start, int end );
        void writeFile( string zeile );
   CityFileHandling() { numberOfCities = 0; };
};
```

Hinweis:

Die Methode

```
string intToStr(int zahl);
```

ist schon vorhanden. Sie können diese bei der Lösung der Teilaufgaben verwenden.
Die Methode konvertiert eine übergebene Integer-Zahl in einen String und gibt diesen zurück.

Beispiel:

```
intToStr(1); //Rückgabe String "1"
intToStr(21); //Rückgabe String "21"
```

A) Implementieren Sie die C++ Methode `filePresent()`. Diese soll überprüfen, ob eine Datei (im aktuellen Verzeichnis) vorhanden ist oder nicht. Der zu überprüfende Dateiname wird dabei als `string filename` an die Methode übergeben. Die Methode soll `true` zurückgeben, wenn die Datei vorhanden ist, ansonsten `false`.

```
bool CityFileHandling::filePresent( string filename ) {



}
```

B) Schreiben Sie die C++ Methode `filesRead()`, welche die Dateien der Städte einliest. Die Methode bekommt dabei den Präfix der Dateien als String, sowie die Anfangszahl (=Nummer der ersten Datei) und die Endzahl (=Nummer der letzten Datei) als Integer übergeben. Die Methode soll die einzelnen Dateien nacheinander öffnen und jeweils die erste Zeile einlesen und diese an die Funktion `writeFile()` (siehe Teilaufgabe C) übergeben. Die Dateien sind dabei durchgängig mit dem Präfix, einer **immer dreistelligen Zahl** und .txt codiert (siehe Abbildung 5-1). Alle bereits vorhandenen Methoden dürfen verwendet werden.

Hinweise:

1. Diese Methode soll die eingelesene Zeile nicht verändern bzw. auswerten, sondern nur immer an die Funktion `writeFile()` übergeben!
2. Überprüfen Sie vor dem Öffnen der jeweiligen Datei, ob diese existiert. Verwenden Sie dazu die in Aufgabenteil A) implementierte Methode `filePresent()`

```
void CityFileHandling::filesRead( string praefix, int start, int end )
{




}
```

C) In der C++ Methode `writeFile()` soll eine **einzelne** Zeile, welche aus einer Datei (siehe Abbildung 5.1 links) ausgelesen wurde, ausgewertet werden. Die Zeile wird der Methode dabei als `string` übergeben.

1. Die Methode soll im 1. Schritt die Zeile mit Hilfe von Stringmanipulationsfunktionen in ihre Bestandteile (Name, Großstadt, Einwohnerzahl) zerlegen und die Daten in die entsprechenden gegebenen string-Variablen speichern.
2. Anschließend soll, wenn es sich um eine Großstadt handelt, der Name der Stadt und ihre Einwohner in die Datei „grossstadt.txt“ geschrieben werden. Die Daten sollen dabei durch ein Semikolon getrennt werden (siehe Abbildung 5-1).
3. Zusätzlich soll die Anzahl der Großstädte im Attribut `numberOfCities` der Klasse gespeichert werden.

Hinweise:

- Bitte beachten Sie, dass die Methode für jede Datei einzeln aufgerufen wird, daher müssen Sie z.B. die Datei immer zum Anhängen öffnen, wenn Sie Daten hinzufügen.
- Weiterhin gilt, dass der String immer vorhanden ist und im korrekten Format vorliegt. Es müssen diesbezüglich keine Fehlerüberprüfungen stattfinden.

```
void CityFileHandling::writeFile( string zeile ) {
string name;
string bigCity;

string einwohner;









}
```

Aufgabe 6 Graphentheorie und Algorithmen

Aufgabe 6.1 Graphentheorie

A) Zeichnen Sie den zur folgenden Adjazenzmatrix gehörigen gerichteten Graphen:

	A	B	C	D	E
A	**1**	**1**	0	0	0
B	0	0	0	**1**	0
C	0	**1**	**1**	0	0
D	0	0	0	**1**	**1**
E	**1**	**1**	0	0	0

Tabelle 6-1 Adjazenzmatrix für den gerichteten Graphen

B) Ändern Sie die Adjazenzmatrix aus A) so ab, dass aus dem gerichteten Graphen ein ungerichteter Graph wird. Entfernen Sie dabei keine bereits vorhandenen Kanten!

	A	B	C	D	E
A					
B					
C					
D					
E					

Tabelle 6-2 Adjazenzmatrix für den ungerichteten Graphen

C) Begründen Sie Ihre Änderungen an der Adjazenzmatrix in B).

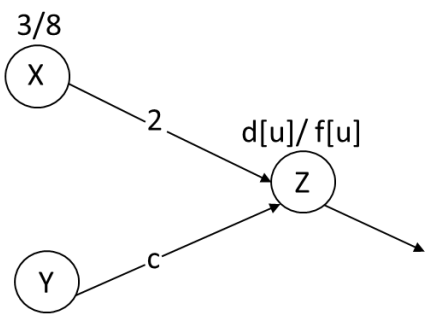
D) Zeichnen Sie einen ungerichteten Graphen, welcher kein Baum ist und aus 4 Kanten und 5 Knoten besteht.

Aufgabe 6.2 Tiefensuchalgorithmus

A) Wenden Sie auf den Graphen in Abbildung 6-1 den Tiefensuchalgorithmus (DFS) an, um festzustellen, ob ausgehend vom Knoten A der Knoten G erreichbar ist. Arbeiten Sie den DFS-Algorithmus **vollständig** ab.
Zeichnen Sie den entstehenden Tiefensuchbaum ausgehend von Startknoten A. Bestimmen Sie für jeden Knoten die Entdeckungszeit d[u] und die Endzeit f[u]

Hinweis:
Die Entdeckungsreihenfolge ist dabei bestimmt durch die Kantengewichte c in aufsteigender Reihenfolge.

Zeichenvorgabe:



Gegeben ist zusätzlich der Pseudocode des Tiefensuchalgorithmus:

```
DepthFirstSearch( G )
  for alle Knoten u in G
  do farbe[u] = weiss
     vater[u] = NIL
  zeit = 0
  for alle Knoten u in G
  do if farbe[u] == weiss
     then DFS-VISIT( u )

DFS-VISIT( u )
  farbe[u] = grau; zeit = zeit + 1
  startTime[u] = zeit
  for alle Knoten v aus Adj[u]
  do if farbe[v] == weiss
     then vater[v] = u
          DFS-VISIT( v )
  farbe[u] = schwarz; zeit = zeit + 1
  endTime[u] = zeit
```

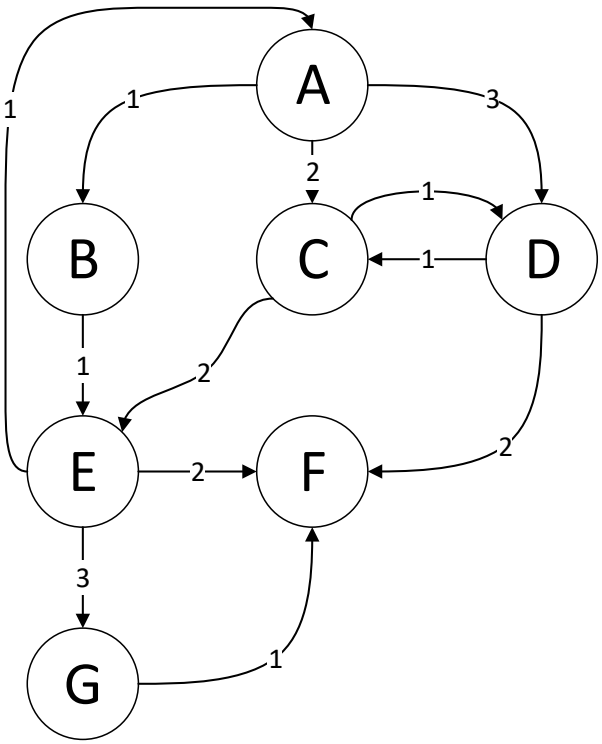


Abbildung 6-1 Graph auf den der Tiefensuchalgorithmus vollständig angewandt werden soll

B) Ergänzen Sie in Ihrem gezeichneten Tiefensuchbaum aus A) alle restlichen Kanten und klassifizieren Sie die Kanten. Markieren Sie:

- *Baumkanten*: keine Markierung
- *Vorwärtskanten*: mit einem F (Forward)
- *Rückwärtskanten*: mit einem B (Backward)

Aufgabe 7 Algorithmenanalyse

Gegeben ist das folgende Programm zur Berechnung einer mathematischen Funktion.

```
01  #include <iostream>
02  using namespace std;
03
04  int doSomething(int zahl) {
05        if (zahl == 1) {
06              return 1;
07        }
08        else {
09              return zahl * do(zahl-1);
10        }
11  }
12
13  int main () {
14        int zahl;
15
16        cout << "Bitte geben Sie eine Zahl ein: ";
17        cin >> zahl;
18
19        cout << "Das Ergebnis betraegt: " << doSomething(zahl) << endl;
20
21        return 0;
22  }
```

Aufgabe 7.1 Algorithmenverständnis

A) Welche mathematische Funktion erfüllt der oben gegebene Algorithmus `doSomething ()`?

B) Ist der Algorithmus ein rekursiver oder iterativer Algorithmus? Begründen Sie Ihre Antwort.

Nehmen Sie an, dass das obige Programm pro Aufruf der Funktion `doSomething()` 192 Byte auf dem 2048 Byte großen Stack belegt.

C) Berechnen Sie die größte Zahl `zahl`, für die die mathematische Funktion berechnet werden kann, bevor es zu einem Stack-Overflow kommt.
Wie müsste der Algorithmus umgeformt werden, um diese Zahl zu vergrößern, ohne die Größe des Stacks anzupassen?

Hinweis: Gehen Sie davon aus, dass die `main()` Funktion selbst und die darin befindlichen Variablen etc. keinen Speicher auf dem Stack belegen.

Aufgabe 7.2 Laufzeitanalyse

A) Führen Sie zum unten gezeigten Code eine Laufzeitanalyse durch und füllen Sie dabei für die Zeilen 01-09 die folgende Tabelle für den Worst-Case Fall aus. Geben Sie jeweils die Anzahl der Ausführungen in Abhängigkeit von *n*, welches der Anzahl der Elemente im Array entspricht, an.

Hinweis: Es gilt allgemein: $\sum_{j=1}^{n}(j)=\frac{n\cdot(n+1)}{2}$

Zeile		Anzahl der Ausführungen

01
02
03
	`/* Conversion of matrix to upper triangular */`
	`int i, j;`
	`float ratio;`	

04	`for(i = 0; i < n; i++){`	
05	`  for(j = 0; j < n; j++){`	
06	`  if(j>i){`	
07	`  ratio = matrix[j][i]/matrix[i][i];`	
08	`  for(k = 0; k < n; k++){`	
09	`  matrix[j][k] -= ratio * matrix[i][k];`	

10
11
12
13
	`  }`
	`  }`
	`  }`
	`}`	

Gesamtkomplexität des Algorithmus:

B) Der Algorithmus könnte durch das Entfernen einer Zeile beschleunigt werden. Welche Zeile müsste entfernt werden? Welche weitere Zeile müsste dann wie modifiziert werden, damit der Algorithmus noch korrekt arbeitet?

Aufgabe 8 Rechnerarchitektur

Aufgabe 8.1 Allgemeine Fragen zur Rechnerarchitektur

A) Nennen Sie die Phasen eines von-Neuman-Zyklus und ordnen Sie diese der Reihenfolge nach.

B) Was versteht man in Bezug auf Caches unter Verdrängungsstrategie? Nennen Sie ein Beispiel für eine Strategie und erklären Sie dessen Funktionsweise.

C) Gegeben ist das Schema eines einfachen von-Neumann Rechners. Tragen Sie die Namen der vier Komponenten, aus denen sich das Schema zusammensetzt, in die leeren Felder ein.

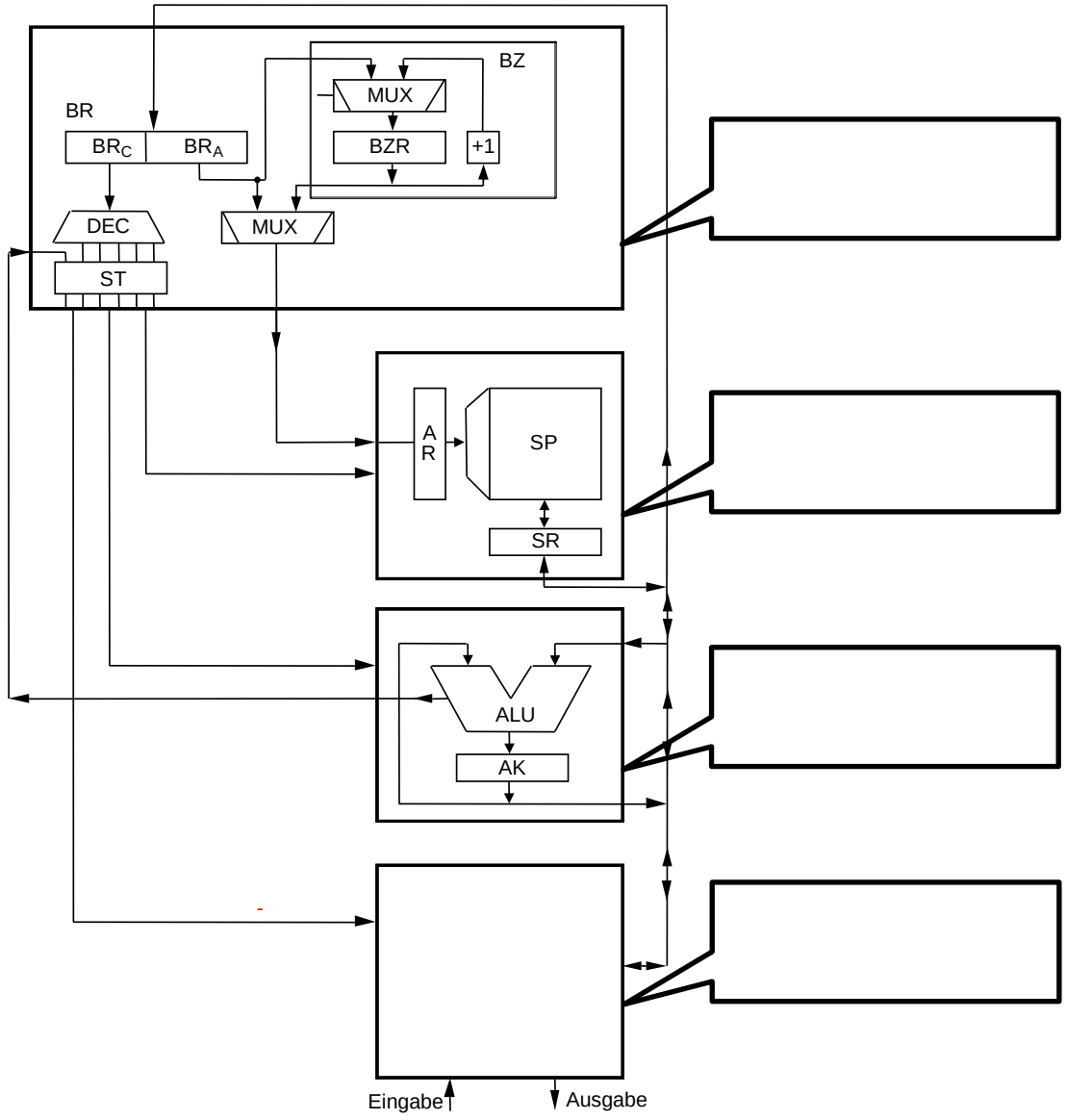


Abbildung 8-1: Aufbau eines von-Neumann-Rechners

Aufgabe 8.2 Cache

Ein Prozessor habe einen Hauptspeicher mit einer Größe von 8 GByte, wobei jedem einzelnen Wort (=2 Byte) im Speicher eine 32 Bit lange Adresse zugeordnet sei.
Der Prozessor verfügt über einen Cache-Baustein der Technologie SRAM, der 1024 KByte (nur Daten) vom Hauptspeicher aufnehmen kann, unabhängig vom notwendigen Speicher zur Speicherung der weiteren Bits, wie z.B. Tag.

Annahmen:

- Setzen Sie das Cache-Modell, wie es in der Vorlesung eingeführt wurde, voraus
- Pro Cache-Block (=Zeile) werden *k* Worte adressiert

A) Der Cache sei 4-fach assoziativ und habe eine Blockgröße von 128 Bit. Zusätzlich sei für jeden Cacheblock ein Valid-Bit vorgesehen.

Berechnen Sie die Größe des Offsets, des Index und des Tags in Bit. Geben Sie dabei den Rechenweg an.

Offset =

Index =

Tag =

B) Unabhängig von dem vorherigen Aufgabenteil A) sei nun ein 2-fach assoziativer Cache mit einer Blockgröße von 64 Bit, einem Offset von 2 Bit, einem Index von 16 Bit und einem Tag von 14 Bit gegeben. Ergänzen Sie in die folgende Abbildung 8-2 die 4 offenen gestrichelten Verbindungen und füllen Sie die 13 offenen Felder aus, indem Sie in die gepunkteten Rechtecke die passenden Zahlen und in die gestrichelten Rechtecke die Passenden Begriffe bzw. Symbole eintragen.

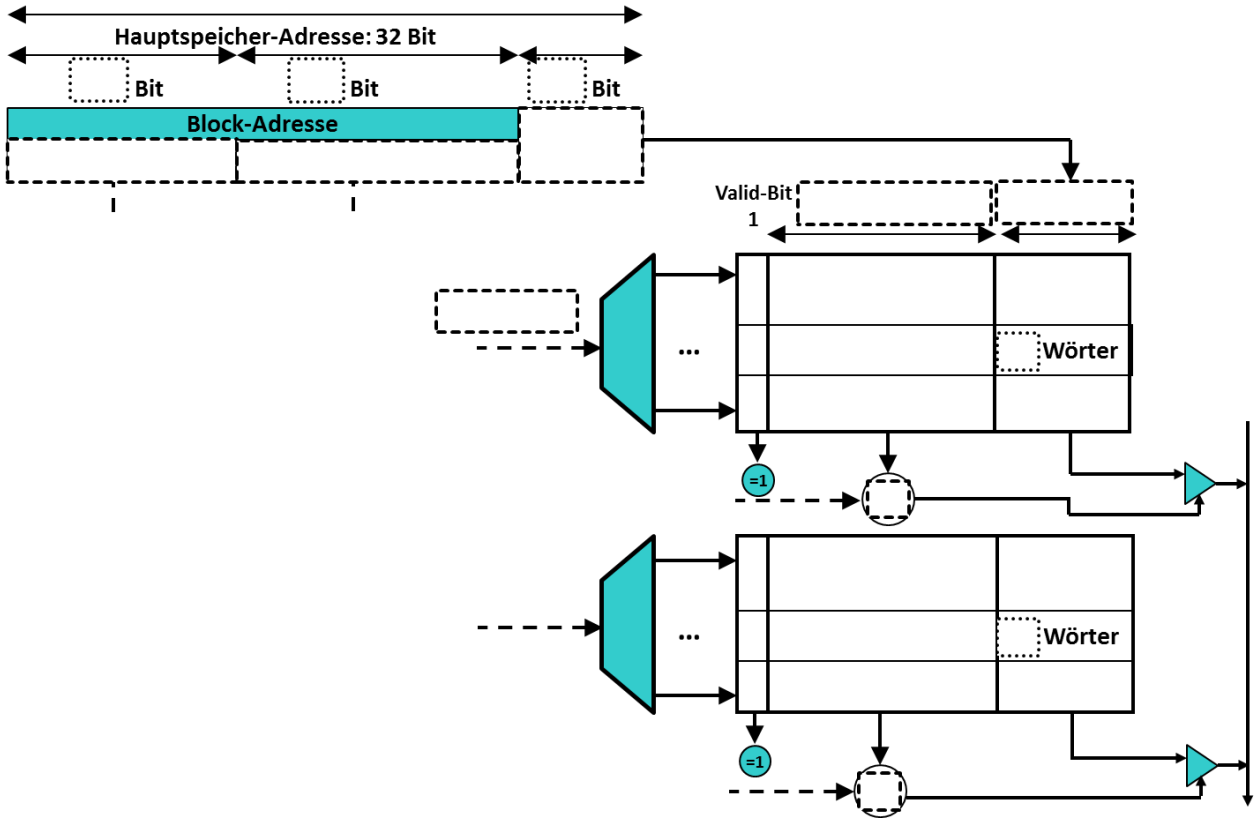


Abbildung 8-2: Speicherorganisation des Caches

Zusätzliches Lösungsblatt

Aufgabe _____