

Informationstechnik

Klausur Sommersemester 2015

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



Informationstechnik

Datum: 01.10.2015
Name: «Vorname» «Nachname»
Matrikelnummer: «Matrikelnummer»
ID #: «LaufNr»

Hörsaal: «Hörsaal»

Platznummer: «PlatzNr»

Hinweise zur Klausur

Hilfsmittel

- Erlaubte Hilfsmittel sind Lineal und eine Notizen-/Formelsammlung, 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 **30** Seiten Aufgabenblättern (dieses Titelblatt, **8** Aufgabenblöcke). 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, 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. [%]	13	13	12	10	13	13	12	14	100
Ergebnis [P]									



Aufgabe 1 Allgemeine Fragen

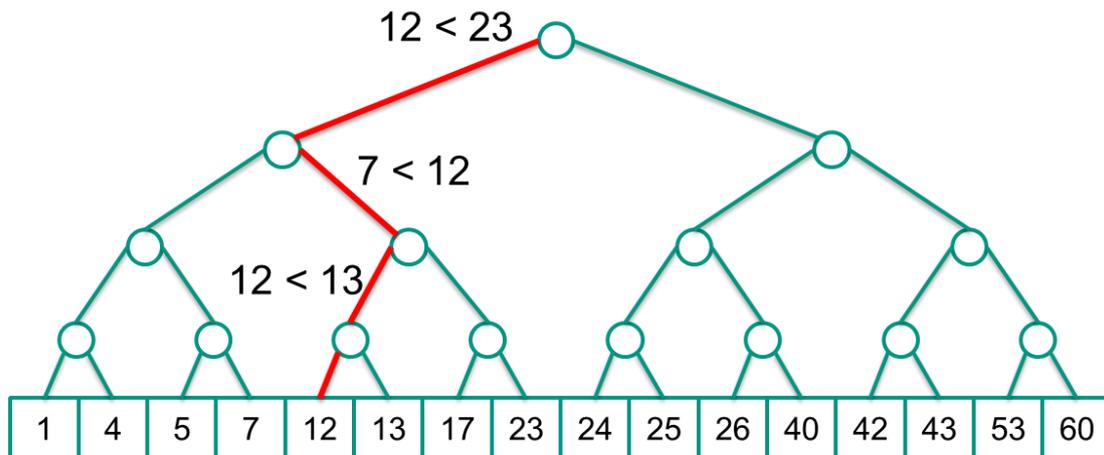


Abbildung 1-1: Suchalgorithmus

- A) Mit welchem Suchalgorithmus wurde hier in Abbildung 1-1 nach dem Zielelement 12 gesucht? Geben Sie zudem die worst case Komplexität des Algorithmus an.

- B) Nennen Sie einen In-Place Sortieralgorithmus und erklären Sie die Eigenschaft In-Place.

- C) Nennen Sie zuerst den Zweck von Polling und Interrupts an Hand eines Beispiels. Erklären Sie zudem den Unterschied anhand des Vorgehens und listen Sie je einen Vor- und Nachteil auf.

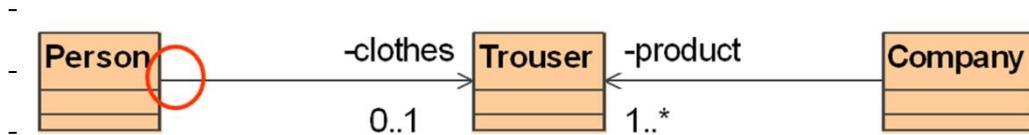
D) Nennen Sie zuerst die vier Gruppen von elementaren Datentypen und geben Sie je ein Beispiel an:

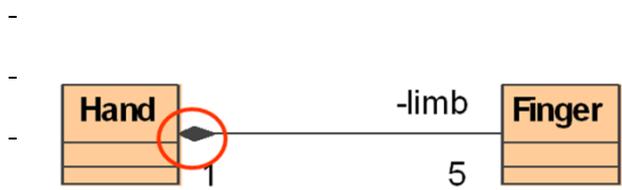
E) Beschreiben Sie kurz das iterative Vorgehen von Simulated Annealing mit Bedingung zum Tauschen von Knoten aus unterschiedlichen Partitionen.

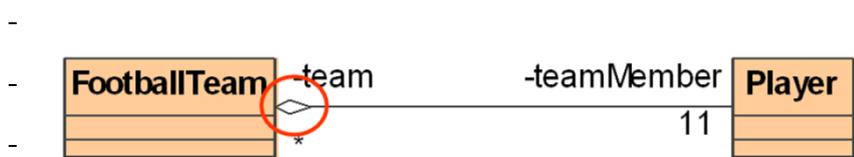
F) Nennen Sie die geeignetste Datenstruktur zum Zwischenspeichern von zeitabhängigen Sensordaten die in zeitlich korrekter Abfolge weiterverarbeitet werden sollen. Begründen Sie Ihre Antwort!

G) Was versteht man unter Effizienz und Effektivität eines Algorithmus?

H) Geben Sie die durch das jeweilige Bild dargestellte Klassen-Beziehung an (die durch einen Kreis markierte Stelle) und erklären Sie deren Bedeutung.







Aufgabe 2 C++ Verständnisfragen



A) Welchen Wert speichert der String `s` nach Ausführung des folgenden Codes:

```
std::string s("inline static int get_value()");  
int n = s.find('t');  
s.erase(4, n - 2);  
s = s.substr(3, 3) + s[16];
```

B) Entscheiden Sie sich im Folgenden jeweils für einen sinnvollen C++ Datentypen für die gegebenen Ausdrücke/Variablennamen:

	<code>array_index = 280;</code>
	<code>var0 = &array_index;</code>
	<code>var1 = 0xFC3FB0A1 & 0xFD9A1002;</code>
	<code>var2 = (var0) && (array_index < 0xF);</code>
	<code>var3 = "Hallo Welt!";</code>
	<code>C_0 = 2.99e8; //nicht veränderlich</code>

C) Gegeben ist der folgende fehlerbehaftete C++ Quellcode:

```

01 #ifndef PREPROCESSOR_MACRO
02
03 #include <cstdlib>
04
05 class Fahrzeug {
06 public:
07     float gewicht
08 };
09
10 class PKW: public Fahrzeug {
11     void set_gewicht(float g) {
12         this->gewicht = g;
13         return g;
14     }
15 public:
16     void PKW(float g) {
17         gewicht = g;
18     }
19 }
20
21 PKW* init_pkw(void) {
22     PKW* pkw = new PKW((rand() - 40.0) / "4.0");
23     Fahrzeug* fz = pkw;
24
25     if (fz->gewicht >= 2.0) {
26         return fz;
27     } else if (pkw.gewicht < 0) {
28         return 0;;
29     } else {
30         pkw->set_gewicht( 1e1 );
31     }
32 }
33 }
34
    
```

Finden Sie mindestens 8 Stellen im Quellcode, an denen der Compiler einen Fehler ausgeben würde. Geben Sie in der folgenden Tabelle jeweils die entsprechende Zeilennummer sowie die Ursache des Fehlers an:

Zeile	Fehler

Matrikelnr: «Matrikelnummer»
«LaufNr»

Name: «Vorname» «Nachname»

ID-Nr.:

--	--

D) Selbst wenn alle Compiler-Fehler im Code der vorigen Teilaufgabe 0 behoben werden, verursacht die Funktion `init_pkw` zur Laufzeit u.U. ein weiteres Problem. Um welches Problem handelt es sich hierbei?

Hinweis: Das Problem tritt z.B. beim Erreichen von Zeile 28 auf, wo 0 statt dem neu erstellten PKW-Objekt zurückgegeben wird.

E) Gegeben sei die folgende C++ Funktion, welche zur Vereinfachung der Fehlersuche um diverse Konsolenausgaben ergänzt wurde:

```
string do_something(unsigned int input) {
    unsigned int &i = input;

    const char* vect = "0123456789ABCDEF";
    string temp = "";

    cout << "input=" << input << ", i=" << i << endl;

    while (i > 0) {
        short index = input % 0x10;
        temp = *(vect + index) + temp;
        input = input / 0x10;
        cout << "input=" << input << ", index=" << index << endl;
    }
    cout << "input=" << input << ", i=" << i << endl;

    return temp;
}
```

Welche Bildschirmausgabe erzeugt die Funktion bei dem Aufruf `do_something(92)`?
Welcher Rückgabewert ergibt sich in diesem Fall?

F) Welchen Zweck erfüllt die Funktion aus der vorherigen Teilaufgabe E)?



Aufgabe 3 Objektorientierung in C++

Aufgabe 3.1 C++ Code Programmausgabe

```
#include <iostream>
using namespace std;
class A {
    public:
    int f() { return 1; }
    virtual int g() { return 2; }
};
class B : public A {
    public:
    int f() { return 3; }
    int g() { return 4; }
};
int main()
{
    A a;
    B b;
    A* p1 = &a;
    A* p2 = &b;
    B* p3 = &b;
    cout << "p1: " << p1->f() << ' ' << p1->g() << endl;
    cout << "p2: " << p2->f() << ' ' << p2->g() << endl;
    cout << "p3: " << p3->f() << ' ' << p3->g() << endl;
    return 0;
}
```

A) Geben Sie die Ausgabe des obenstehenden Programms an:



Aufgabe 3.2 Fehlersuche

A) Benennen Sie drei Fehler im folgenden C++-Code:



```
class A {
    private:
        A() { }

    protected:
        int BerechneWert() { return -1; }
};

class B : public A {
    private:
        const int value;

    public:
        B() { }
        int BerechneWert() { return -2; }
        void set_value( int x ) {
            value = x;
        }
};

int main() {
    B b;
    A* pA = &b;
    return pA->BerechneWert();
}
```

Aufgabe 3.3 C++ Ausdrücke

A) Erklären Sie kurz die Bedeutung der folgenden Ausdrücke von C++.



i. Konstruktoren und Destruktoren in Klassen in C++?

ii. `protected` als Schlüsselwort einer Methode in Bezug auf Vererbung in C++?

iii. Erklären sie den Begriff „Polymorphie“ in Bezug auf C++ und geben Sie dazu ein einfaches Beispiel in C++, welches die relevanten Konstrukte enthält und das Prinzip verdeutlicht (eine main() Funktion ist nicht nötig).

Aufgabe 3.4 Vererbung

A) Schreiben Sie den C++ Code, der die Klassen entsprechend der folgenden Beschreibung enthält:



Eine Klasse `Person` mit zwei Attributen `Nachname` und `Vorname`. Sowie eine Klasse `StudentIn` und eine Klasse `ProfessorIn`, die beide von der Klasse `Person` erben. Die Klasse `StudentIn` soll ein Attribut `Matrikelnummer` haben, die Klasse `ProfessorIn` ein Attribut `Lehrgebiet`. Alle Attribute sind vom Typ `string` und sollen von Außen nicht sichtbar sein. Zum Zugriff von Außen sollen Methoden zum Lesen der Attribute in jeder Klasse hinzugefügt werden.

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



Aufgabe 4 STL & Datenstrukturen

```
#include <string>
#include <map>

using namespace std;

class KaffeeMaschine{
private:
    map< string, double > konten;

public:
    KaffeeMaschine ();
    int      konto_hinzufuegen(string name);
    int      konto_aufladen( string name, double betrag);
    double   gesamtguthaben();
};
```

Zum Testen der Klasse KaffeeMaschine wurde ein Programm geschrieben, dessen Code wie folgt ist:

```
#include "kaffee.h"
#include <iostream>
#include <vector>

int main(void)
{
    KaffeeMaschine *Kaffee = new KaffeeMaschine();
    string name = "";
    double betrag = 0;

    unsigned int i;
    std::vector<pair<string, double>> buchungen;
    buchungen.push_back({ "Lipp", 10.00 });
    buchungen.push_back({ "Steinbuch", 45.00 });
    buchungen.push_back({ "Lipp", -5.00 });
    buchungen.push_back({ "Steinbuch", -50.00 });

    for (i = 0; i != buchungen.size(); i++)
    {
        name = buchungen[i].first;
        betrag = buchungen[i].second;
        if (Kaffee->konto_hinzufuegen(name))
            cout << "Konto " << name
                << " bereits vorhanden\n\n";

        if (Kaffee->konto_aufladen(name,betrag))
            cout << "Aufladen Konto " << name
                << " nicht moeglich\n\n";
    }
    int kontostand = Kaffee->gesamtguthaben();
    std::cout << "Der Gesamtbetrag ist " << kontostand << "\t\n";
    return 0;
}
```

Aufgabe 4.1 Implementierung

Implementieren Sie die folgenden Methoden der Klasse `KaffeeMaschine` in C++. Die Methoden sollen so implementiert werden, dass die Ausführung des angegebenen Testprogramms folgende Ausgabe erzeugt:

```
Konto Lipp bereits vorhanden
```

```
Aufladen Konto Steinbuch nicht moeglich
```

```
Der Gesamtbetrag ist 50
```

A) Die Methode `Kaffemaschine::konto_hinzufuegen` soll implementiert werden. Falls zu dem String `name` ein Eintrag in der map `konten` existiert, soll die Methode den **Wert -1** zurückgeben und keine Änderung durchführen. Ansonsten soll in `konten` ein Eintrag mit dem Schlüssel `name`, dem **Wert 0** angelegt, und als Rückgabewert **0** zurückgegeben werden.



```
int KaffeeMaschine::konto_hinzufuegen(string name)
```

```
{
```

```
}
```

B) Die Methode **Kaffemaschine::konto_aufladen** soll eine Buchung durchführen. Dabei wird der Kontostand eines Nutzers erhöht oder verringert werden. Falls der Kontostand, bei Verrechnung des Betrags, negativ werden würde, soll der **Wert -1** zurückgegeben und der Kontostand nicht verändert werden. Dasselbe gilt auch für den Fall, dass der Nutzer in **konten** nicht gefunden wurde. Bleibt der resultierende Betrag positiv, so wird der **Wert 0** zurückgegeben.

Der Zugriff auf die Elemente der map **konten**, der Klasse **KaffeMaschine**, ist nur über **Iteratoren** erlaubt.

```
int Kaffeemaschine::konto_aufladen (string name, double betrag)
{
```

```
}
```

C) Die Methode **Kaffeemaschine::gesamtguthaben** ist zu implementieren. Sie soll die Summe der Guthaben aller Konten als Rückgabewert zurückgeben.

Der Zugriff auf die Elemente der map **konten**, der Klasse **KaffeMaschine**, ist nur über **Iteratoren** erlaubt.

```
double Kaffeemaschine::gesamtguthaben()
{
```

```
}
```

Aufgabe 4.2 STL Verständnisfragen

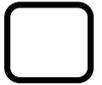
A) Nennen Sie zwei Gründe wieso zum Zugriff auf Container Iteratoren, anstatt eines direkten Zugriffs, verwendet werden.

B) Gibt es einen Unterschied in der Schnittstelle zwischen einer `queue`, die eine `deque` und einer `queue`, die eine `list` verwendet? Falls ja, welchen? Begründen Sie Ihre Antwort!

C) Nennen Sie einen Vorteil in der Verwendung von Algorithmen aus der STL für die Verarbeitung von Datenelementen eines Containers.

D) Nennen Sie den Unterschied zwischen einer `Map` und einer `Multimap` der STL.

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



Die Caesar-Verschlüsselung (auch als Cäsar-Chiffre, Cäsar-Algorithmus, Caesar-Verschiebung, Verschiebechiffre oder als Einfacher Caesar bezeichnet) ist ein einfaches symmetrisches Verschlüsselungsverfahren.

Bei der Verschlüsselung wird jeder Buchstabe des Klartexts auf einen Geheimtextbuchstaben abgebildet. Diese Abbildung ergibt sich, indem man die Zeichen eines geordneten Alphabets um eine bestimmte Anzahl zyklisch nach **links** verschiebt (rotiert). Die Anzahl der verschobenen Zeichen bildet den Schlüssel, der für die gesamte Verschlüsselung unverändert bleibt. (Quelle: Wikipedia)

Geschrieben werden soll ein Programm, das einen Text aus einer Datei ausliest und dann Caesar-verschlüsselt.

Dazu soll die Datei zeilenweise ausgelesen werden, mit einer eigens erstellten Methode verschlüsselt und in eine Zielfeile geschrieben werden.

Zur Vereinfachung werde ein Alphabet verwendet, das nur aus den Kleinbuchstaben **a – z** und den Großbuchstaben **A – Z** besteht.

Beispiel:

```
ich bin ein Beispieltext
```

wird mit einer Verschiebung um 3 Zeichen zu:

```
lfk elq hlq EhlvslhowhAw
```

Aufgabe 5.1 Grundlagen

A) Was ist die maximal sinnvolle Verschiebung beim hier genutzten Alphabet? Begründen Sie!

B) Wie muss die Verschiebung gewählt werden, um den verschlüsselten Text durch das Verschlüsselungsprogramm wieder lesbar zu machen?

- C) Schreiben Sie den Code, der die Datei `quelle.txt` zum Lesen als `infile` und die Datei `ziel.txt` zum Schreiben als `outfile` öffnet. Außerdem soll jeweils überprüft werden, ob das Öffnen erfolgreich war.

```
int main()
```

```
{
```

```
/* Hier folgt der Code aus Aufgabe 5.2B) */
```

```
}
```

Aufgabe 5.2 Verschlüsselung

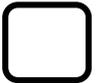
Erstellen Sie nun im Folgenden die Methode `verschluesselel(...)` zur Verschlüsselung. Beachten Sie folgende Anforderungen:

- Die Methode soll eine ganze Zeile als Parameter annehmen und verschlüsselt zurückgeben
- Die Methode soll einen weiteren Parameter enthalten, der den Betrag der Verschiebung als Ganzzahl angibt
- Die Methode soll das verwendete Alphabet (Kleinbuchstaben a – z und Großbuchstaben A - Z) in einem String ablegen
- Die Methode soll das um den per Parameter übergebenen Betrag verschobene Alphabet in einem String speichern
- Falls der übergebene Parameter für die Verschiebung außerhalb des gültigen Bereichs liegt, soll „fehler“ zurückgegeben werden

A) Wie sieht der Prototyp der Methode `verschluesselel(...)` in der Header-Datei aus?

B) Geben Sie das C++ Code-Fragment an, das die bereits als `infile` geöffnete Datei ausliest. Anschließend soll die durch den Prototypen in **Aufgabe 5.2A)** gegebene Methode genutzt werden, den Inhalt zu verschlüsseln. Dieser verschlüsselte Inhalt soll schließlich in die als `outfile` geöffnete Datei geschrieben werden. Setzen Sie eine Zeichenverschiebung von **13** ein.

Beachten Sie, dass die gesamte Datei zeilenweise ausgelesen werden soll!



Aufgabe 6 Graphentheorie

Aufgabe 6.1 Breitensuche

Die Adjazenzmatrix in Tabelle 6-1 beschreibt einen Graphen.

Quelle \ Ziel	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	1	0	1	0	0	0	1	0	0	0
2	1	0	1	0	0	0	0	0	0	0	0	0
3	0	0	0	1	0	0	1	0	0	0	0	0
4	0	0	0	0	0	0	0	1	0	0	0	0
5	0	0	0	0	0	1	0	0	1	0	0	0
6	0	1	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	1	0	1	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	1	0	0	0	0	1	1
11	0	0	0	0	0	0	1	0	0	0	0	0
12	0	0	0	0	0	0	0	1	0	0	0	0

Tabelle 6-1 Adjazenzmatrix

A) Vervollständigen Sie den Graphen in Abbildung 6-1 anhand der Adjazenzmatrix.

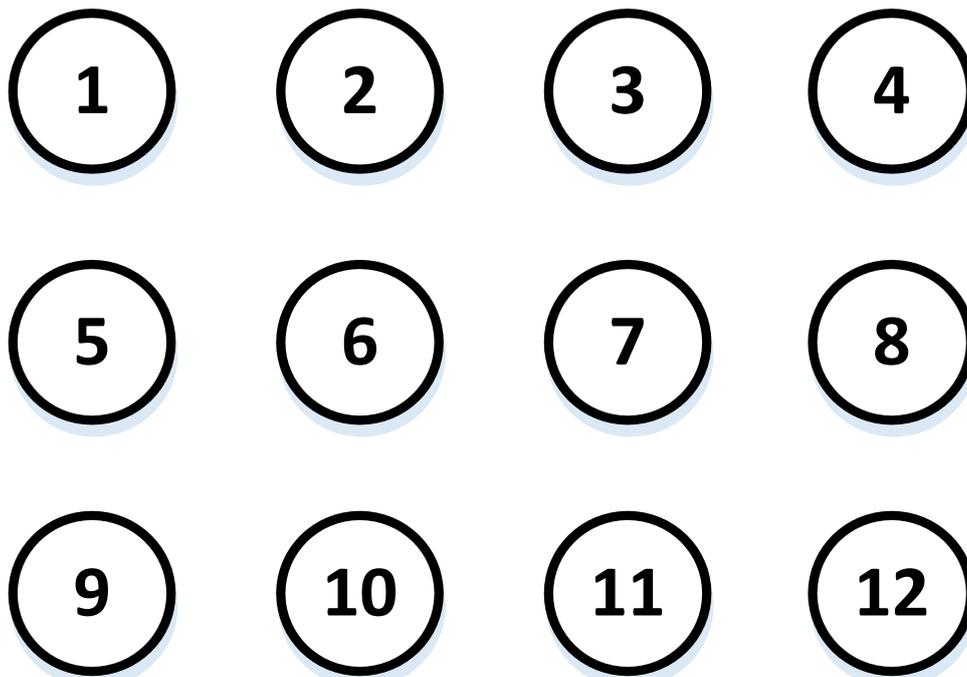


Abbildung 6-1 Graph

Aufgabe 6.2 Critical Path Method

In Abbildung 6-2 ist ein CPM (Critical Path Method) Graph dargestellt. Hierbei sind die Vorgänger als Pfeile mit der Vorgangsdauer t in Stunden und Ereignisse als Knoten (A-I) dargestellt (siehe Legende).

A) Setzen Sie in die leeren Kästchen jeweils die richtigen Werte für die Vorgangsdauer t , den frühesten Termin x , spätesten Termin y und den Puffer z mit Hilfe der Critical Path Method ein. Anschließend geben Sie den „kritischen Pfad“ unter dem Graph an.

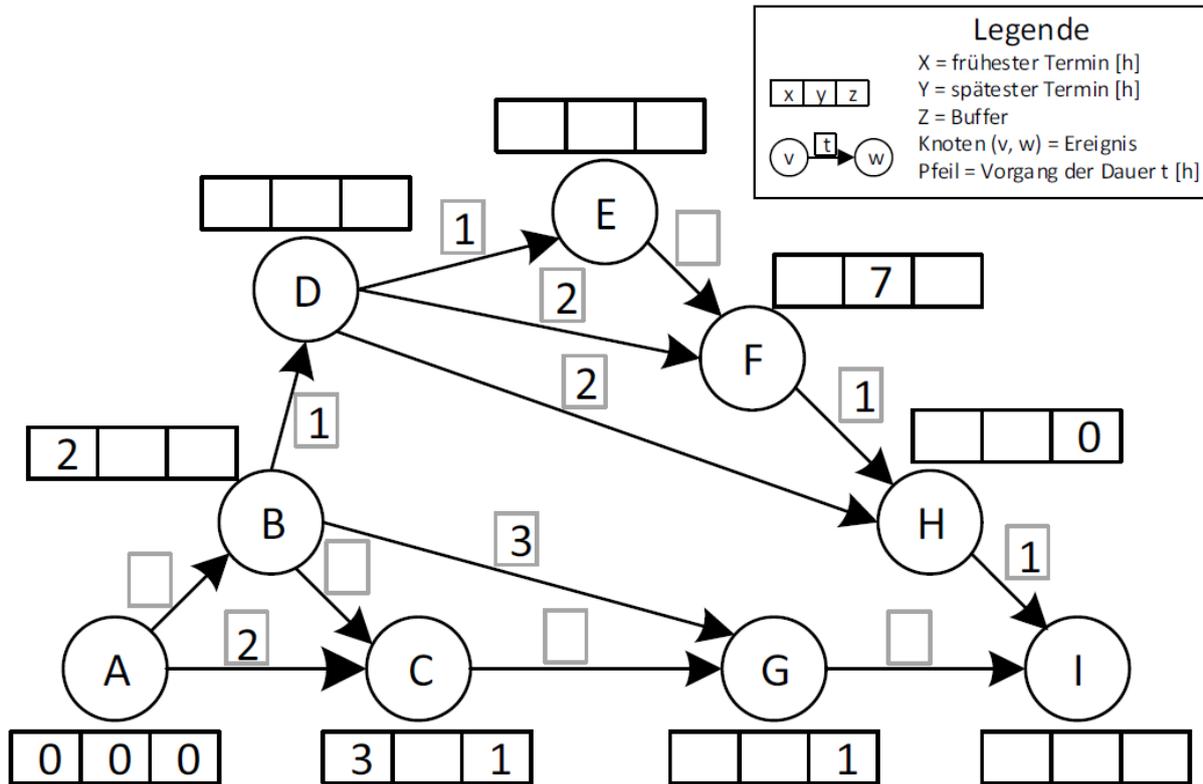


Abbildung 6-2 Critical Path Method Graph

Kritischer Pfad:



Aufgabe 7 Algorithmenanalyse

Gegeben ist der C++ Code eines angepassten Swap-Sort Algorithmus. Dieser sortiert ein Array in aufsteigender Reihenfolge. Der gezeigten Funktion wird ein Pointer auf ein Array `sortMe` und die Anzahl der Elemente `length` übergeben.

```
01 void swapSort (int* sortMe, int length) {
02     int sortval = 0;
03     bool* markerarr = new bool[length];
04     for (int i = 0; i < length; i++) {
05         markerarr[i] = false;
06     }
07
08     while (sortval < length) {
09         int counter = 0;
10         for (int i = 0; i < length; i++) {
11             if (sortMe[i] < sortMe[sortval]) {
12                 counter++;
13             }
14         }
15
16         int tmp = sortMe[sortval];
17         sortMe[sortval] = sortMe[counter];
18         sortMe[counter] = tmp;
19         markerarr[counter] = true;
20
21         sortval = 0;
22         while (markerarr[sortval] == true) {
23             sortval++;
24         }
25     }
26 }
```

Aufgabe 7.1 Algorithmenverständnis

A) Erklären Sie die Aufgabe der Zeilen 09-14 innerhalb des Algorithmus. Was ist der Zweck der Variablen `counter` in Bezug auf das Array?



B) Welche Aufgabe hat innerhalb des Algorithmus das Array `markerarr`?

Aufgabe 7.2 Algorithmenanalyse

A) Gegeben sei der folgende Code zur Multiplikation zweier quadratischer Matrizen. Führen Sie eine Laufzeitanalyse durch und füllen Sie dabei für die Zeilen 01-09 die folgende Tabelle aus. Geben Sie jeweils die Anzahl der Ausführungen in Abhängigkeit von n , welches der Anzahl der Elemente in den Arrays entspricht, an.

Zeile		Anzahl der Ausführungen
01	<code>int i, j, k, n, z;</code>	
02	<code>for(i = 0; i < n; i++) {</code>	
03	<code> for(j = 0; j < n; j++) {</code>	
04	<code> z = 0;</code>	
05	<code> for (k = 0 ; k < n ; k++)</code>	
06	<code> z += input1[i][k] * input2[k][j];</code>	
07	<code> output[i][j] = z;</code>	
08	<code> }</code>	
09	<code>}</code>	

Gesamtkomplexität des Algorithmus:

B) Wie verhält sich die Gesamtkomplexität des obigen Algorithmus für beliebige Eingabematrizen? Begründen Sie Ihre Antwort.

Aufgabe 7.3 Algorithmenentwurf

Der Algorithmus aus Aufgabe 7.2 soll für die Multiplikation von oberen Dreiecksmatrizen verwendet und optimiert werden.

Hinweis: Bei einer oberen Dreiecksmatrix sind alle Elemente unterhalb der Hauptdiagonale gleich Null. Eine obere Dreiecksmatrix hat allg. folgende Form:

$$M = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ 0 & a_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{n-1,n} \\ 0 & \dots & 0 & a_{n,n} \end{pmatrix}, \text{ Beispiel: } M_{4 \times 4} = \begin{pmatrix} 3 & 2 & 3 & 4 \\ 0 & 5 & 5 & 6 \\ 0 & 0 & 6 & 7 \\ 0 & 0 & 0 & 9 \end{pmatrix}$$

A) Welche Optimierung ist in Bezug auf den obigen Hinweis nötig, damit der Algorithmus unter Beibehaltung der Funktionalität schneller arbeitet? Begründen Sie Ihre Antwort.

B) Wie muss das Programm aus Aufgabe 7.2 verändert bzw. ergänzt werden, damit der Algorithmus schneller arbeitet? Sie dürfen das Programm nur in Zeile 3 oder Zeile 5 verändern bzw. ergänzen. Sie können die Lösung entweder textuell beschreiben oder direkt Programmcode angeben.

Hinweis: Geben Sie für Veränderungen bzw. Ergänzungen immer die entsprechende Zeilennummer des Quellcodes an.



Aufgabe 8 Rechnerarchitekturen

Aufgabe 8.1 Verständnisfragen

In Abbildung 8-1 sei das Blockbild einer Prozessorarchitektur gegeben.

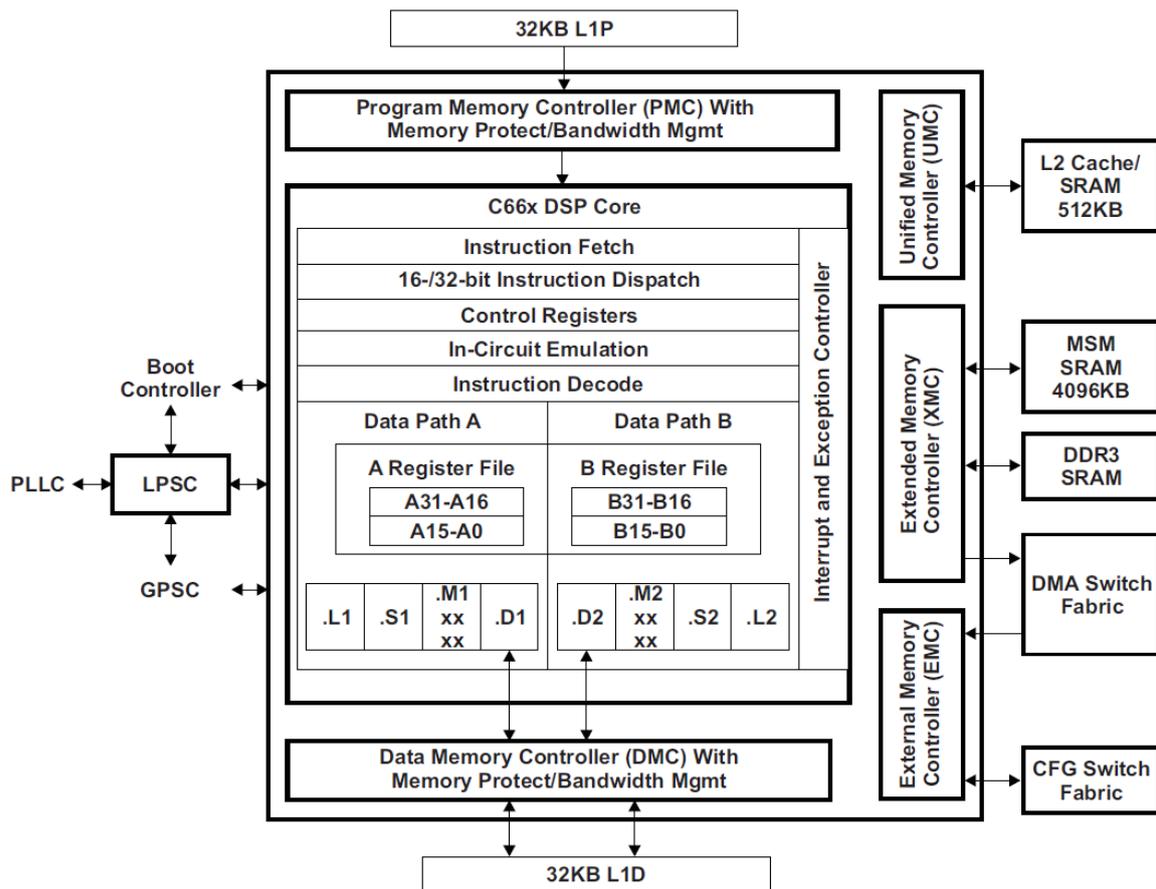


Abbildung 8-1: Blockbild eines Prozessorkerns

- A) Handelt es sich in Abbildung 8-1 um eine Von-Neumann oder um eine Harvard Architektur? Begründen Sie Ihre Antwort!



Aufgabe 8.2 Cache

Ein Prozessor habe einen Hauptspeicher mit einer Größe von 4 GByte, wobei jedem einzelnen Byte im Speicher eine 32 Bit lange Adresse zugeordnet sei.

A) Der Prozessor verfügt über einen Cache-Baustein der Technologie SRAM, der 128 KByte (nur Daten) vom Hauptspeicher aufnehmen kann, unabhängig vom notwendigen Speicher zur Speicherung der weiteren Bits, wie z.B. Tag. Der Cache sei 4-fach assoziativ und habe eine Blockgröße von 256 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.

B) Unabhängig von dem vorherigen Aufgabenteil A) sei nun ein 2-fach assoziativer Cache mit einer Blockgröße von 64 Bit, einem Offset von 3 Bit, einem Index von 14 Bit und einem Tag von 15 Bit gegeben. Ergänzen Sie in die folgende Abbildung 1-2 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.

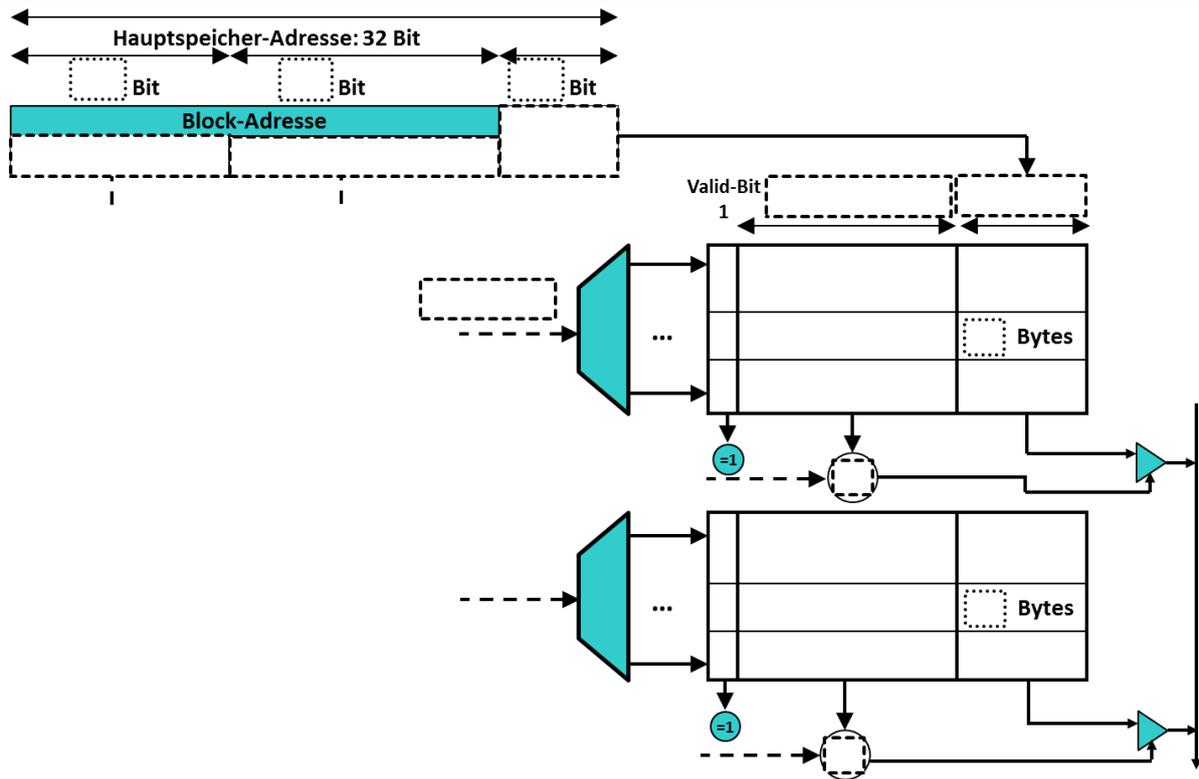


Abbildung 8-2: Speicherorganisation des Caches