Fakultät IV
Fachgebiet NI

Lernerfolgskontrolle (C)
PPR
Nur zur Übung. Es gibt keinerlei Garantien, dass die tatsächliche Klausur den gleichen Umfang und Schwierigkeitsgrad hat.

Probeklausur

Name: .....

Matr.-Nr. .....

Bearbeitungszeit: 75 Minuten

Bewertung

Aufgabe	Punkte	Erreichte Punkte
1	8	
2	4	
3	5	
4	8	
5	4	
6	4	
7	9	
8	5	
9	5	
10	8	
Summe	60	

Hinweise:

- Verwenden Sie für die Lösung der Aufgaben **nur** das mit diesem Deckblatt ausgeteilte Papier. Lösungen, die auf anderem Papier geschrieben werden, können **nicht** bewertet werden. Schreiben Sie ihre Lösung auch auf die Rückseiten der Blätter; benötigen Sie für eine Lösung mehr als ein Blatt, finden Sie am Ende der Klausur Leerblätter. Zusätzliches Papier können Sie von den Tutoren bekommen.
- Tragen Sie vor Beginn der eigentlichen Bearbeitungszeit auf **allen** Blättern ihren Namen ein.
- Schreiben Sie deutlich! Unleserliche oder mehrdeutige Lösungen können nicht gewertet werden.
- Schreiben Sie **nicht** mit Bleistift und **nicht** mit rotem oder grünem Stift (das sind die Farben für die Korrektur), verwenden Sie **kein** Tipp-Ex und **keinen** Tintenkiller
- Für Fragen mit wahr/falsch-Ankreuzmöglichkeiten (*Multiple Choice*) gilt: Für jede richtige Antwort gibt es einen halben Punkt, für jede falsche Antwort wird ein halber Punkt abgezogen – weniger als 0 Punkte in einer Aufgabe sind aber nicht möglich.

Wir wünschen Ihnen viel Erfolg!

Aufgabe 1 (8 Punkte) Zahlensystem.

1. (3 Punkte) Berechen Sie 15-47 unter Verwendung der 8 Bit Zweierkomplementdarstellung. Die Rechenschritte müssen erkennbar sein.

2. (2 Punkte) Die genetischen Informationen sind bei allen Lebewesen in den DNA-Molekülen gespeichert. Ein DNA-Molekül ist eine lineare Sequenz der vier Nukleobasen A,C,G und T. Das menschliche Genom enthält rund 2×10^9 Nukleobasen. Wie groß ist das gesamte Datenvolumen des menschlichen Genoms in Bytes?

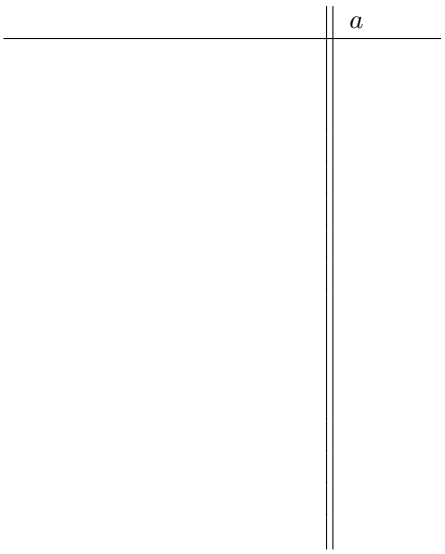
3. (3 Punkte) Stellen Sie 30,25 in der 2 Byte binären Gleitkommadarstellung dar.
   (1 Vorzeichenbit, 4 Bit Exponent: 7-Exzess-Darstellung, 11 Bit Mantisse).

Aufgabe 2 (4 Punkte) Logische Schaltungen.

1. (2 Punkte)

   Stellen Sie die Wahrheitstabelle für eine Funktion auf, die sowohl ausgeben kann, ob eine Zahl gerade ist, als auch, ob eine Zahl ungerade ist. Sie erhält als Eingabe eine Zahl von 0 bis 3 sowie eine Eingabe, die angibt, ob die Ausgabe anzeigt, ob die Zahl gerade oder ungerade ist. Diese Eingabe erfolgt natürlich binär.
   Beispiele (in Textform – eure Umsetzung dann in binär, und nur mit Zahlen von 0 bis 3!):
   f(123, gerade) = falsch
   f(514, gerade) = wahr
   f(865, ungerade) = wahr

   Wahrheitstabelle:

2. (2 Punkte) Zeichnen Sie eine Gatterschaltung, die obige Wahrheitstabelle umsetzt. Sämtliche bekannten Gatter dürfen verwendet werden.

Aufgabe 3 (5 Punkte) Rechnerarchitektur.

1. (3 Punkte)
   Was versteht man unter einem *Betriebsystem*? Nennen Sie *drei Aufgaben* eines Betriebsystems.

   (a)

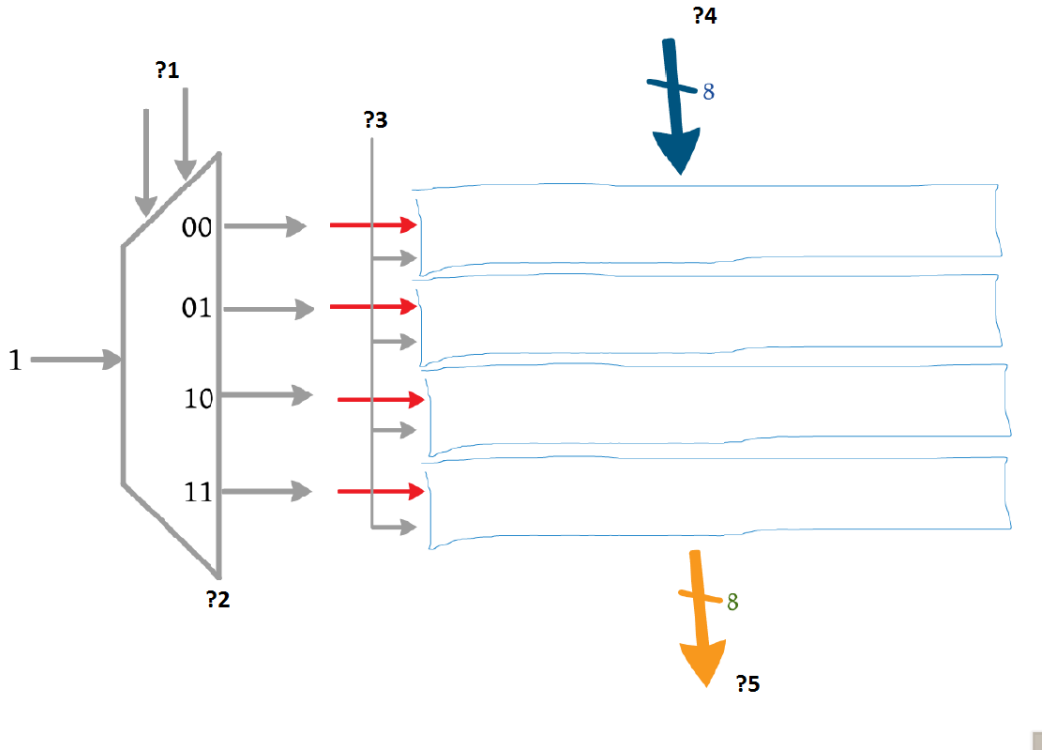
   (b)

   (c)

2. (2 Punkte)
   Erläutern Sie kurz die Begriffe *RAM* und *ROM*. Nennen Sie 2 Typen für RAM.

Aufgabe 4 (8 Punkte) Rechneraufbau.

1. (4 Punkte)
   Gegeben ist folgendes (aus der Vorlesung bekannte) Bild:



   (a) Was stellt dies als Gesamtheit dar? (Was für eine Funktionseinheit)

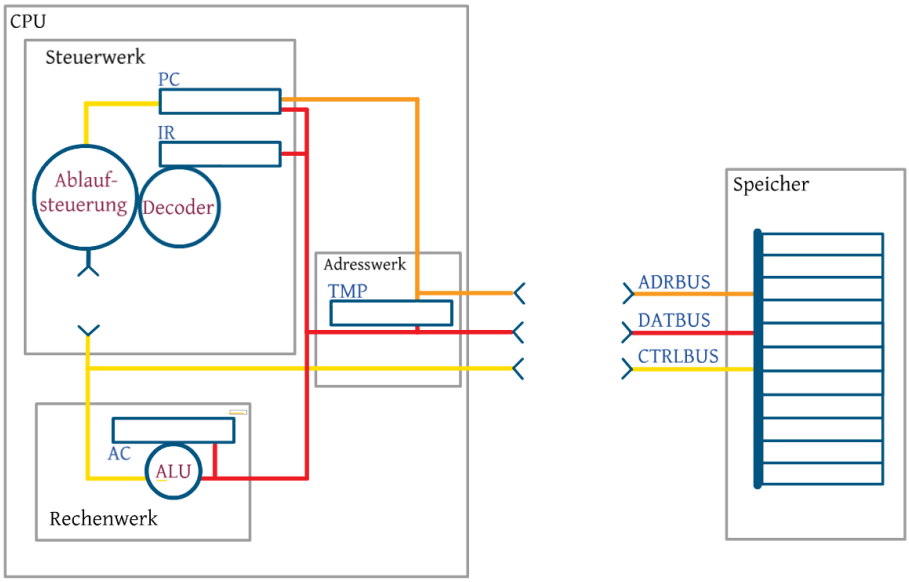
   (b) Geben Sie `?4` und `?5` vernünftige Namen und erläutern Sie die Funktion dieser Leitungen. Was bedeutet der Querstrich mit der „8“?

   (c) Was ist die Funktion von Leitung `?3`?

   (d) Wie heißt das mit `?2` gekennzeichnete Bauteil? Wozu wird es *allgemein* (nicht nur in Bezug auf diese Schaltung) verwendet?

   (e) Erläutern Sie die Funktion der Leitungen `?1`.

2. (4 Punkte)
   Gegeben ist wieder das folgende aus dem Tutorium bekannte vereinfachte Rechnerblockschaltbild:



   a) Erklären Sie kurz und knapp die Funktion der Bauteile PC, IR, ALU, AC, ADRBUS, CTRLBUS, DATBUS.

b) Die folgende Tabelle stellt einen Speicherausschnitt vor Beginn der darunter angegeben Befehlsausführung dar. Der Index $_2$ ist benutzt, um anzuzeigen, dass diese Zahl im Binärsystem angegeben ist.

	RAM
5003	*XOR*
5004	5114
...	
5114	00001111_2

PC	IR	AC	TMP	CTRL	CBUS	ABUS	DBUS
5003	XOR	01100110_2	5113	PC++, PC→ABUS,RAM→DBUS,DBUS→IR	read	5003	XOR
5004	XOR	01100110_2	5114	PC++, PC→ABUS,RAM→DBUS,DBUS→TMP	read	5004	5114
5004	XOR	01101001_2	5114	TMP→ABUS,RAM→DBUS,DBUS→ALU	read	5114	00001111_2

Erläutern Sie für jede Ablaufzeile die Programmausführung.

Aufgabe 5 (4 Punkte) Dateisystem.

1. (4 Punkte)

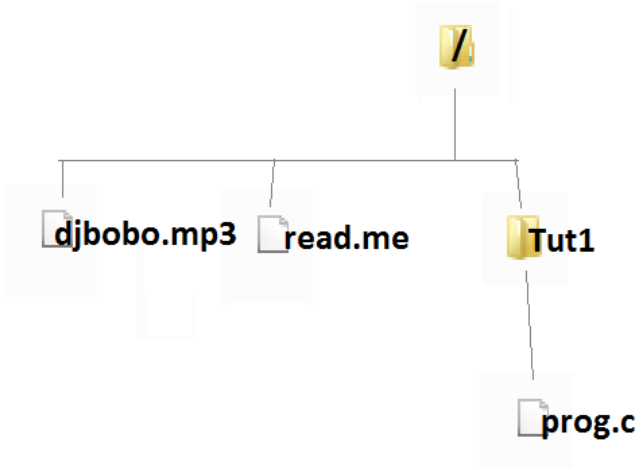
   Das Betriebssystem speichert alle Dateien in seinem internen Dateisystem, insbesondere in der Inode-Tabelle und im Datenblock.

   (a) Was wird im Datenblock abgelegt, was in der Inode-Tabelle?

   (b) Verzeichnisse sind auch Dateien – Verzeichnisdateien. Was liegt bei einer Verzeichnisdatei im Datenblock?

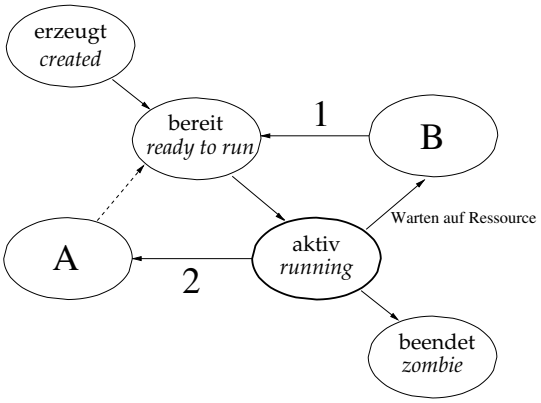
   (c) Wo findet das System den logischen Dateibaum, der dem Nutzer in Dateiexplorer-Tools angezeigt wird?

   (d) Wieviele Inode-Ids sind in folgendem Beispiel vergeben? (Nicht sichtbare Elemente werden nicht berücksichtigt)



Aufgabe 6 (4 Punkte) UNIX.

1. (2 Punkte)



Nennen Sie die in dem Diagramm fehlenden Prozesszustände A und B (englische oder deutsche Bezeichnung), sowie die Ursachen für die Übergänge 1 und 2:

Zustand A:

Zustand B:

Übergang 1:

Übergang 2:

2. (2 Punkte) Beantworten Sie die folgenden Fragen durch Ankreuzen von „wahr“ oder „falsch“. Jede richtige Antwort gibt 0,5 Punkte, für jede falsche Antwort werden 0,5 Punkte abgezogen. Insgesamt können aber nicht weniger als 0 Punkte erreicht werden.

	wahr	*falsch*
Das Anhängen von & an einen Befehl lässt diesen im Hintergrund laufen.	( )	( )
Systemprioritäten haben immer Vorrang vor Benutzerprioritäten.	( )	( )
Mit dem Kommando `nice` kann jeder Benutzer die Prioritäten seiner eigenen Prozesse verbessern.	( )	( )
Mehrere laufende Prozesse können sich eine Prozessnummer teilen	( )	( )

Aufgabe 7 (9 Punkte) C.

1. (3 Punkte)

   Schreiben Sie eine Funktion `copyData`, die zwei void-Pointer übergeben bekommt (einer für die Quelle, einer für das Ziel) und die Anzahl der zu kopierenden Bytes. Es kann davon ausgegangen werden, dass der benötigte Speicherbereich am Ziel bereits reserviert ist.

```
void copyData ( void* source, void* dest, int nrOfBytesToCopy )
{




}
```

2. (3 Punkte)

1.) Ergänzen Sie den Code unten (wo nötig mit beliebigen (aber halbwegs passenden) Werten). Hinweis: `fabs` ist eine Funktion der math-library, die den Betrag einer double-Zahl zurückgibt.

```
// Hier die Definitionen ergaenzen:
#define TRUE
#define FALSE

// Hier die fehlende define-Praeprozessor-Direktive ergaenzen:


// Hier die fehlende Typdefinition ergaenzen:
typedef

bool areEqual ( double d1, double d2 )
{
  if ( fabs( d1 - d2 ) > TOLERANCE ) return FALSE;
  if ( fabs( d1 - d2 ) <= TOLERANCE ) return TRUE;
}
```

2.) Erklären Sie die Funktion und Motivation für die Funktion `areEqual`

3.) Vereinfachen Sie die Funktion so weit wie möglich (eine Zeile reicht...).

```
// Hier die Funktion vereinfachen (die Praeprozessor- und Typdefinitionen von oben sind noch gueltig !):

bool areEqual ( double d1, double d2 )
{


}
```

3. (3 Punkte) Schreiben Sie einen Prototypen (nur der Funktionskopf!) für eine Funktion, die...
a) einen String „Hallo Welt“ erzeugt und diesen zurückgibt
b) von einem float-Array der Länge 100 eine Kopie anlegt und die Adresse dieser Kopie zurückgibt
c) einen String „Hallo Welt“ erzeugt und diesen ausgibt

Aufgabe 8 (5 Punkte) C.

1. (1 Punkt)

   Programmieren Sie eine Ausgabe an den Nutzer, dass er angeben möge, wieviele float-Zahlen in einem später anzulegenden Array gespeichert werden sollen, und lesen Sie daraufhin die folgende Nutzereingabe in die Variable memoryRequest ein.

```
int memoryRequest;
```

2. (2 Punkte)

   Reservieren Sie Speicher für den float-Array floatArray mit der Größe memoryRequest (siehe oben). Dabei sollen schon bei der Speicherreservierung alle Zellen auf 0 initialisiert werden. Überprüfen Sie anschließend, ob die Reservierung erfolgreich war, wenn nicht, geben Sie eine Fehlermeldung aus und beenden das Programm mit dem Fehlercode -1.

```
float* floatArray;
```

3. (2 Punkte)

   Analysieren Sie folgenden Code:

```
float* fa  = floatArray;
void* fav = (void*)floatArray;
double* fad = (double*)floatArray;
printf("%p", fa); // (Ausgabe)
fa  = fa + 4; // (1)
fav = fav + 4; // (2)
fad = fad + 4; // (3)
```

   Angenommen, an der Stelle (Ausgabe) würde „160“ ausgegeben werden – welchen Wert hat fa nach Ausführung der mit (1) markierten Zeile, welchen Wert hat fav nach Ausführung der mit (2) markierten Zeile, und welchen Wert hat fad nach Ausführung der mit (3) markierten Zeile?

Aufgabe 9 (5 Punkte) C.

1. (5 Punkte)

   Betrachten Sie folgendes Programm, und ergänzen Sie die darunter stehende Ablauftabelle.

   - Tragen Sie die Variablenbelegungen nach Ablauf der jeweiligen Zeile ein.
   - Schreiben Sie bei Funktionsaufrufen zusätzlich die aufrufende Zeile in Klammern hinter die momentane Zeilennummer. Bsp: Wir befinden uns in Zeile 14 in einer Funktion, die in Zeile 34 aufgerufen wurde. `Zeile: 14 (34)`
   - Die erste aufgeführte Zeile einer Funktion ist die öffnende Blockklammer ({).
   - Nach Aufruf eines `return`-Statements wäre die nächste ausgeführte Zeile der Funktionsabschluss (}).
   - Kennzeichnen Sie aktuell nicht im Speicher angelegte Variablen mit `-`, existierende Variablen mit undefinierten Werten mit `undef`.
   - Die Zahl der Zeilen in der Tabelle ist auch abgezählt, d.h., so viele Zeilen werden im Code auch durchlaufen und von euch bitte beschrieben.

```
1  double lcl = −3.0;
2
3  void sub(double z)
4  {
5     lcl  −= z;
6  }
7
8  int addCond(char x, double y)
9  {
10    if ( !x ) return (y + 0.49);
11    else if ( x >= 2 ) return (y + 4.9);
12    else return (y − 0.49);
13 }
14
15 int main()
16 {
17    int c;
18    c = addCond(2, lcl);
19    sub( lcl );
20     lcl  = lcl  − (c + 1);
21 }
```

Zeile	lcl	z	x	y	c
16					
21					

Aufgabe 10 (8 Punkte) C.

1. (2 Punkte)

   Definieren Sie eine Struktur mit dem Namen **pprStudent**, welche folgende Informationen erfasst: Den Namen (String, max. Länge 100), die Hausaufgabennote (als Gleitkommazahl), und die Klausurnote (auch Gleitkommazahl). Definieren Sie dabei auch einen neuen Typ **PprStudent** für diese Struktur.

2. (1 Punkt)

   Definieren Sie eine neue Struktur, die eine einfach verkettete Liste von **PprStudent**-Strukturen darstellen kann. Definieren Sie wieder auch einen neuen Typ **PprlistElem** für die Struktur.

3. (2 Punkte)

   Was ist der Vorteil einer verketteten Liste gegenüber der Speicherung in einem Array? Gibt es auch Nachteile? Fällt Ihnen eine weitere Datenstruktur ein, die besonders für die Suche in den Daten geeignet ist?

4. (3 Punkte)

   Schreiben Sie folgende Funktion, die die Durchschnittsnote aller Studenten berechnen soll (Note pro Student: 70% Klausur, 30% Hausaufgaben). Das übergebene Argument pprlist soll dabei das erste Element der Liste sein. Überprüfen Sie, ob überhaupt ein Element in der Liste vorhanden ist!

```
float averageMark( PprlistElem* pprlist )
{









}
```