Christian-Albrechts-Universität zu Kiel
Institut für Informatik

Multimedia Information Processing Group
Prof. Dr. Reinhard Koch
Dr.-Ing. Christoph Starke, M.Sc. Vasco Grossmann,
M.Sc. Sascha Clausen

Computersysteme

Wintersemester 2018/2019

Serie 4

Ausgabetermin: Freitag, 09.11.2018
Abgabetermin: Freitag, 23.11.2018, 08:00 Uhr im Schrein

Bitte klammern oder heften Sie Ihre Abgabeblätter geeignet zusammen und notieren Sie sowohl Ihre Namen als auch Ihre Gruppennummer auf der Abgabe!

Präsenzaufgaben

Aufgabe 1

Vereinfachen Sie den Ausdruck für die Boolesche Algebra $(A, +, \cdot)$ mit $A = \{0, 1\}$ – wobei 0 als *falsch* und 1 als *wahr* interpretiert werden kann – so weit wie möglich:

$$(x_0 + x_1) \cdot (x_0 + \overline{x_1}) \quad x_0, x_1 \in A$$

Stellen Sie dafür zunächst eine Wahrheitstabelle für alle Kombinationen von $x_0, x_1 \in A$ auf.

Aufgabe 2

Gegeben ist ein sehr einfacher Rechner, der nur folgende arithmetisch-logische Operationen kann:

- Zuweisung: `= (y=x)`
- Increment: `x-- (x=x-1), x++ (x=x+1)`
- Shiftoperatoren: `x << y` ($x = x \cdot 2^y$), `x >> y` ($x = x \cdot 2^{-y}$)
- Test auf Null: `(x==0)`
- Bitweise logische Operationen:

`~` (bitweise Komplement)	`|` (bitweise OR)
`&` (bitweise AND)	`^` (bitweise XOR)

Es dürfen nur diese Operationen benutzt werden.

Berechnen Sie für eine Integerzahl die Anzahl der signifikanten Stellen der Binärdarstellung des Betrags der Zahl. Sie können davon ausgehen, dass diese Zahl von Null verschieden ist und größer ist als die vom Betrage nach größtmögliche negative Zahl. Bei negativen Zahlen soll daher erst die positive Zahl berechnet werden. Geben Sie die Anzahl der signifikanten Stellen des Betrags aus, d.h. führende Nullen sollen nicht betrachtet werden.
Es liegt bereits ein Codefragment vor, mit dem die Zahl eingelesen und ausgegeben werden soll. Entwerfen Sie zunächst jeweils einen PAP für die beiden Teilprobleme (a) und (b). Versuchen Sie, den PAP so einfach wie möglich zu gestalten.

(a) Testen Sie, ob die eingegebene Zahl `eingabe` positiv oder negativ ist. Falls negativ, wandeln Sie die Zahl in ihren positiven Betrag um. Verwenden Sie in ihrem Code ausschließlich die obigen Operationen.

(b) Berechnen Sie, wie viele signifikante Stellen die Zahl hat (führende Nullen sollen weggelassen werden). Geben Sie die Stellenzahl mit `stellen` aus.

```c
#include <stdio.h>
int main()
{
  // Gib die Anzahl der signifikanten Stellen des Betrags einer ganzen Zahl aus

  // eingabe: zu analysierende Zahl, betrag: positiver Wert von eingabe,
  // stellen: Anzahl der signifikanten Stellen von betrag als Ausgabe,
  // maxStellen: maximal moegliche Stellenanzahl von betrag als Integerzahl
  int eingabe, betrag, stellen, maxStellen;
  printf("Gib ganze Zahl ein: ");
  scanf("%d", &eingabe);
  maxStellen= sizeof(int) << 3; // Berechne Bitanzahl aus Byteanzahl * 8
  if (eingabe==0)
  {
    printf("Sonderfall Eingabe=0, keine Auswertung\n");
    return 0;
  }

  // Hier bitte eigenen Code fuer Aufgaben a) und b) einbringen

  printf("Zahl hat %d signifikante Stellen\n",stellen);
  return 0;
}
```

Hausaufgaben

Aufgabe 1

Vereinfachen Sie folgende Ausdrücke für die Boolesche Algebra $(A, +, \cdot)$ so weit wie möglich:

(a) $x_0, x_1 \in A : \overline{(\overline{x_0} + ((x_0 \cdot (x_1 + x_0)) \cdot x_0)) + \overline{x_0}}$

(b) $x_0, x_1 \in A : (\overline{x_0} + \overline{x_1}) + (x_0 \cdot x_1)$

Benutzen Sie dazu die Axiome zu Booleschen Algebren aus dem Vorlesungsskript sowie eine geeignete Auswahl der Sätze 1 bis 10.
Beachten Sie dazu den Hinweis in der Beispiellösung zur Präsenzaufgabe.

12, 12 Punkte

Aufgabe 2

Es soll ein Programm zur Analyse der IEEE-Floating-Point Zahlen (32 Bit) geschrieben werden, mit dem Vorzeichen, Mantisse, Exponent und Exponent ohne Bias separat ausgegeben werden sollen. Dazu sollen die Werte je einmal in hexadezimal, dezimal und binär dargestellt werden. Da kein Binärformat bei `printf()` existiert, soll hierzu eine Funktion `printBinary()` geschrieben werden. Weiterhin sollen die fünf Fälle des IEEE-Formats (Normalisiert, sehr kleine Zahlen, Null, Infinity, NAN) erkannt werden und entsprechende Ausgaben dazu erfolgen. Für die Zerlegung des IEEE-Formats soll eine `union` eingesetzt werden.

insgesamt 76 Punkte

(a) Entwerfen Sie eine Funktion void `printBinary(int zahl, int stellen)`. Die Funktion soll den Integer `zahl` bitweise als Zeichen $\{0, 1\}$ ausgeben, wobei der Wert `stellen` angibt, wieviele Bits von `zahl` ausgegeben werden sollen, in der Reihenfolge der letzten Stellen {`stellen-1, stellen-2, ... ,0`}. Geben Sie einen PAP an und schreiben Sie Ihren Code in den Funktionsrumpf.

25 Punkte

(b) Interpretieren Sie die 32-Bit-Darstellung der `union` (`ieee.i`) als Floating Point-Repräsentation und zerlegen Sie diese in die Komponenten Vorzeichen, Exponent und Mantisse. Maskieren Sie die jeweils relevanten Bits und verschieben Sie diese derart, dass die Zahlen immer mit niederwertigsten Bit (LSB) beginnen. Beispiel: Welche Bits belegt der Exponent, wie maskiere ich diese, und wie müssen sie verschoben werden, damit diese Bits ganz rechts stehen? Setzen Sie Ihren Code in das Rumpfprogramm ein, das die Werte anzeigt.

20 Punkte

(c) Prüfen Sie die fünf Fälle des IEEE-Formates und schreiben Sie für die vier Sonderfälle (sehr kleine Zahl, Null, Infinity, NAN) jeweils eine Ausgabe in `printf()` mit der Mitteilung, um welchen Sonderfall es sich handelt. Beispiel für Null: `printf("\nNull! Eingabe = %f\n",eingabe);`

10 Punkte

(d) Wenn der normalisierte Standardfall ausgewählt wird, setzen Sie die IEEE-Zahl wieder aus Vorzeichen, Mantisse, Exponent korrekt zusammen und geben Sie diese Zahl als Floatzahl mit `printf()` aus (15 Punkte). Testen Sie das Programm ausgiebig und geben Sie wie üblich das Programm und einen Screenshot für die Zahlen 1.625, 0.0, -3.3 mit ab (jeweils 2 Punkte).

21 Punkte

```c
#include <stdio.h>
void printBinary(int zahl, int stellen)
{

   // Schreiben Sie hier Ihren Code fuer Aufgabe a) hinein

}

int main()
{
    // Interpretation IEEE Floatformat
    unsigned int vorzeichen, mantisse, exponent ; //Natuerliche Zahlen Vorzeichen, Exponent, Mantisse. Nutzen Sie diese Variablen zur Speicherung der Komponenten der IEEE-Zahl
    char expUnbiased; // Fuer den Exponenten ohne Bias
    float eingabe; // zu interpretierende Zahl

    union {  // dient zur Adressierung der Bitfolge der floatzahl ieee.f mittels der Integerzahl ieee.i
        float f;
        unsigned int i;
    } ieee;

    printf("Gib die zu analysierende Zahl ein: ");
    scanf("%f",&eingabe); // lies die Zahl ein

    ieee.f = eingabe; // union floatwert ieee.f setzen,
    // Binearwerte aus ieee.i rauslesen
    // In ieee.i steht die 32 Bit Bitfolge der Floatzahl

    // Geben Sie hier Ihr Codefragment fuer Aufgabe b) ein

    // Ausgabe der Zerlegung, nutzen Sie diesen Code fuer die Ausgabe:
    printf("Float %f als Hex %X ist zerlegt in\nVorzeichen %d \nMantisse %X, \nExponent %d, \nExpUnbiased %d\n", ieee.f, ieee.i, vorzeichen, mantisse, exponent, expUnbiased);

    printf("\nVorzeichen ist \t");
    printBinary(vorzeichen,1);

    printf("\nMantisse binaer ist \t");
    printBinary(mantisse,23);

    printf("\nExponent binaer ist \t");
    printBinary(exponent,8);

    printf("\nExpUnbiased binaer ist \t");
    printBinary(expUnbiased,8);

    // Pruefe die Sonderfaelle

    // Fuegen Sie Ihren Code fuer Aufgabe c) ein (die vier Sonderfaelle)

    // Fuer den Standardfall (normalisiert), bauen Sie die IEEE-Zahl wieder zusammen,
    // so dass sie ueber ieee.f ausgegeben werden kann:

    // Fuegen Sie Ihren Code fuer Aufgabe d) hier ein. Geben Sie die Zahl mit
    printf("Zahl ist: %f\n", ieee.f);
    //   aus.

    return 0;
}
```