Informatik 3 – Übung 04 – Georg Kuschk

4.1 a) Binärdarstellung

$\log_2 4096 = 12$

$\Rightarrow$ Es werden 13 Stellen zur Binärdarstellung benötigt.
(da inklusive der Null 4097 Zahlen dargestellt werden müssen)

$\log_2 2047 \approx 11.999$

$\Rightarrow$ Es werden 12 Stellen zur Binärdarstellung benötigt.

$\log_2 131578387 \approx 26.97$

$\Rightarrow$ Es werden 27 Stellen zur Binärdarstellung benötigt.

$\log_2 56238383874 \approx 35.71$

$\Rightarrow$ Es werden 36 Stellen zur Binärdarstellung benötigt.

$\log_2 8753784734747 \approx 42.99$

$\Rightarrow$ Es werden 43 Stellen zur Binärdarstellung benötigt.

4.1 b) Binärdarstellung - Verfahren

Die Dezimalzahl sei als n gegeben.
Umwandlung einer Dezimalzahl in eine Binärzahl mittels des Horner-Schemas :

1. setze $i = 0$
2. setze $x_i = n \bmod 2$ (Modulo-Division)
3. setze $n = n \; div \; 2$ (Ganzzahl-Division)
4. Falls $n > 0$, setze $i = i + 1$ und fahre mit Schritt 2 fort

Das Ergebnis ist $x = x_i x_{i-1} ... x_1 x_0$ und hat den Wert $\sum_{j=0}^{i} \left(2^j \cdot x_j\right)$ $, (= n)$.

4.2) MIPS

```
#Aufgabe 4.2)
#Georg Kuschk

#ACHTUNG :      Da laut Forum davon ausgegangen werden soll, dass der Eingabewert,
#               falls er denn kleiner gleich 10 Stellen besitzt,
#               mit maximal 32Bit darstellbar ist, wird nicht noch extra ueberprueft
#               ob der Eingabewert zwischen 4294967296 und 9999999999 liegt
#
#               Ebenso werden eventuell fuehrende Nullen nicht entfernt, sonder als
#               Dezimalstelle mitgezaehlt

.data
str_txt1:   .asciiz     "Bitte vorzeichenlose, maximal 10stellige Dezimalzahl eingeben : "
str_txt2:   .asciiz "\nFehler! - Eingabe ist keine vorzeichenlose, maximal 10stellige Dezimalzahl!"
str_txt3:   .asciiz "\nDezimalzahl akzeptiert"

str_input:  .space      64
```

```
.text

main:
        li      $v0,04          #print_string "Bitte vorzeichenlose, maximal 10stellige Dezimalzahl eingeben : "
        la      $a0,str_txt1
        syscall

        ##### Dezimalzahl einlesen #####
        li      $v0,08          #read_string - EingabeString in str_input einlesen
        la      $a0,str_input   #Adresse des EingabeStrings laden
        li      $a1,64          #maximale Laenge des EingabeStrings
        syscall                 #schliesslich die Funktion (read_string) aufrufen


        ##### Prüfen, ob die Eingabe eine gueltige, vorzeichenlose Dezimalzahl ist #####
        li      $s0,0           #$s0 zaehlt die Anzahl der Ziffern
input_evaluation:
        lb      $t1,str_input($s0)  #Zeichen des eingegebenen Strings einzeln in $t1 laden und auswerten

        beq     $t1,10,input_end    #Ende des Eingabestrings erreicht?
        blt     $t1,'0',input_error #wenn nicht : ist das Zeichen eine Ziffer ('0'-'9')?
        bgt     $t1,'9',input_error

        add     $s0,$s0,1       #Zeichen ist eine Ziffer -> Anzahl der Ziffern inkrementieren
                                #und den Integer-Wert der Ziffer auf den Stack schreiben

        addi    $sp,-1          #Stackpointer dekrementieren
        sub     $t1,$t1,'0'     #das gelesene Zeichen in entsprechende Ziffer umrechnen
        sb      $t1,1($sp)      #und die Ziffer auf den Stack schreiben

        j       input_evaluation    #da das Ende des Eingabestrings noch ni cht erreicht ist,
                                #das naechste Zeichen auswerten



input_end:
        beq     $s0,$zero,input_error   #hat die Zahl 0 Stellen ? -> dann Fehler
        bgt     $s0,10,input_error      #hat die Zahl > 10 Stellen ? -> dann Fehler

        li      $v0,04          #ansonsten print_string "\nDezimalzahl akzeptiert"
        la      $a0,str_txt3
        syscall

        ##### EingabeString wurde jetzt akzeptiert -> Binaerzahl berechnen
        ##### zur Erinnerung : Anzahl der Ziffern der Dezimalzahl steht in $s0 und muss an dieser Stelle >0 sein
        ##### und die Ziffern stehen in umgekehrter Reihenfolge auf dem Stack

        li      $t0,0           #das Gesamtergebnis wird zunaechst in $t0 zwischengespeichert
        li      $t1,1           #in $t1 stehen die gerade benoetigten 10er-Potenzen
dec2bin_loop:
        li      $t2,0           #$t2 fuer Aufnahme eines Bytes vorbereiten
        lb      $t2,1($sp)      #Ziffer vom Stack holen
        addi    $sp,1           #Stackpointer aktualisieren

        mul     $t2,$t2,$t1     #Ziffern-Wertigkeit entsprechend der Dezimal-Stelle berechnen
        add     $t0,$t0,$t2     #und zum Gesamtergebnis hinzuaddieren

        mul     $t1,$t1,10      #naechste 10er-Potenz berechnen
        sub     $s0,$ s0,1      #Anzahl der noch abzuarbeitenden Stellen dekrementieren
        bgt     $s0,$zero,dec2bin_loop  #und wenn diese >0 ist, mit dem Verfahren fortfahren

        j       work_finished   #wenn alle Ziffern abgearbeitet wurden, fertig


##### Hier ist das Sprungziel des Programms fuer eine ungueltige Eingabe
input_error:
        li      $v0,04          #print_string "\nFehler! - Eingabe ist keine vorzeichenlose, maximal 10stellige Dezimalzahl!"
        la      $a0,str_txt2
        syscall

        ##### Die bisher auf den Stack geschriebenen Ziffern wieder entfernen
        add     $sp,$sp,$s0     #Stackpointer wieder um die Anzahl der auf den Stack geschriebenen
                                #Bytes erhoehen

work_finished:
        add     $v0,$t0,0       #Das Ergebnis wie gefordert in $v0 schreiben
                                #(wurde in $t0 zwischengespeichert, da ueber $v0 alle Funktionsaufrufe erfolgen)



#Da das Ergebnis in $v0 gespeichert werden soll, kann das Programm nicht mit
#       li      $v0,10          #exit
#       syscall
#beendet werden.
```

4.2) ALU - Steuerleitungen

4.2 a)

Bei 16 verschiedenen Funktionen werden 4 Steuerleitungen zur Auswahl benötigt $(2^4 = 16)$.

4.2 b) & c)

Die gesuchte Schaltung ist ein Decoder.

Die 4 Steuerleitungen sind mit s1,s2,s3,s4 bezeichnet.
Ist die Enable-Leitung für eine Funktion gesetzt, so wird diese ausgeführt und liefert am Ausgang das Ergebnis.
Ist die Enable-Leitung nicht gesetzt (=0), so wird die Funktion nicht ausgeführt und am Ausgang liegt eine 0.
Die eventuell für die Funktion benötigten Datenleitungen (Operatoren) sind stark vereinfacht dargestellt.
Die 16 Funktionen sind mit f1,f2,...,f16 bezeichnet.

Der Schaltung liegt das Prinzip zugrunde

$$\begin{aligned} Output = [\quad & (\neg s1 \wedge \neg s2 \wedge \neg s3 \wedge \neg s4) \vee (\neg s1 \wedge \neg s2 \wedge \neg s3 \wedge s4) \vee \\ & (\neg s1 \wedge \neg s2 \wedge s3 \wedge \neg s4) \vee (\neg s1 \wedge \neg s2 \wedge s3 \wedge s4) \vee \\ & (\neg s1 \wedge s2 \wedge \neg s3 \wedge \neg s4) \vee (\neg s1 \wedge s2 \wedge \neg s3 \wedge s4) \vee \\ & (\neg s1 \wedge s2 \wedge s3 \wedge \neg s4) \vee (\neg s1 \wedge s2 \wedge s3 \wedge s4) \vee \\ & (s1 \wedge \neg s2 \wedge \neg s3 \wedge \neg s4) \vee (s1 \wedge \neg s2 \wedge \neg s3 \wedge s4) \vee \\ & (s1 \wedge \neg s2 \wedge s3 \wedge \neg s4) \vee (s1 \wedge \neg s2 \wedge s3 \wedge s4) \vee \\ & (s1 \wedge s2 \wedge \neg s3 \wedge \neg s4) \vee (s1 \wedge s2 \wedge \neg s3 \wedge s4) \vee \\ & (s1 \wedge s2 \wedge s3 \wedge \neg s4) \vee (s1 \wedge s2 \wedge s3 \wedge s4) \qquad] \end{aligned}$$