HOCHSCHULE ESSLINGEN FAKULTÄT INFORMATIONSTECHNIK

MUSTERPRÜFUNG C		Blatt Nr.:	**1 von 7**
Studiengang:	**Kommunikationstechnik Softwaretechnik Technische Informatik**	Semester:	SWB4, TIB4, KTB4
Prüfungsfach:	**Computerarchitektur 3**	Fachnummer:	**4021**
Hilfsmittel:	**Vorlesungs- und Labormanuskript, Fachliteratur, Taschenrechner**	Dauer:	**90 min**

Tragen Sie hier bitte Ihren Namen ein:

Vorname: Nachname:

Lösungsvorschlag (ohne Gewähr)

Aufgabe 1 (30 Punkte):

a) Beschreiben Sie die fünf wesentlichen Komponenten eines jeden Computersystems:

Lösung zu Aufgabe 1a)

- CPU mit
  - Datenpfad (Rechenwerk, ALU)
  - Steuerung (Control, Steuerwerk)
- Speicher (Memory)
- Eingabe und Ausgabe (Input/Output, Peripherie)
- Adress-, Daten- und Steuerbussystem

b) Beschreiben Sie fünf mögliche Adressierungsarten eines 68HCS12-Rechners und geben Sie für jede Variante ein Beispiel:

Lösung zu Aufgabe 1b)

Bezeichnung	Beispiel
• Unmittelbare Adressierung (Immediate)	LDAA #0
• Implizite Adressierung	CLRA
• Registeradressierung	TFR B, A
• Direkte Adressierung	STAA variable
• Register-Indirekte Adressierung	
Mit Index (mit Offset)	STAA 0, X
Mit Prä/Post-Inkrement/Dekrement	STAA 1, X+
• Speicher-Indirekte-Adressierung	
Mit Index	STAA [# $1000, X]
• Relative Adressierung	BRA start
(sieht für Programmierer aus wie direkte Adressierung)	

MUSTERPRÜFUNG C		Blatt Nr.:	**2 von 7**
Studiengang:	**Kommunikationstechnik Softwaretechnik Technische Informatik**	Semester:	SWB4, TIB4, KTB4
Prüfungsfach:	**Computerarchitektur 3**	Fachnummer:	**4021**
Hilfsmittel:	**Vorlesungs- und Labormanuskript, Fachliteratur, Taschenrechner**	Dauer:	**90 min**

c) Welche Funktion hat ein Compiler? Bitte ausführlich erklären.

Lösung zu Aufgabe 1c)

Übersetzen des Programmcodes von Hochsprache (z.B. C) in hexadezimal codierte Maschinensprache (Objektcode) bzw. der Daten in Speicherplatzreservierungen und Initialisierung des Speicherplatzes.

d) Welche Funktion hat ein Linker? Bitte ausführlich erklären.

Lösung zu Aufgabe 1d)

Zusammenbinden des Objektcodes verschiedener Programmmodule sowie Bibliotheken (dabei Auflösung von Referenzen) zu relokierbarem Objektcode.

Konvertierung des relokierbaren Objektcodes in absoluten Objectcode, dh. Zuweisen von absoluten Adressen, erfolgt durch den Locator (bei Embedded Systems, Locator ist bei manchen Toolketten Teil des Linkers) oder durch den Lader des Betriebssystems (bei Computern).

MUSTERPRÜFUNG C		Blatt Nr.:	**3 von 7**
Studiengang:	**Kommunikationstechnik Softwaretechnik Technische Informatik**	Semester:	SWB4, TIB4, KTB4
Prüfungsfach:	**Computerarchitektur 3**	Fachnummer:	**4021**
Hilfsmittel:	**Vorlesungs- und Labormanuskript, Fachliteratur, Taschenrechner**	Dauer:	**90 min**

e) Erläutern Sie den Unterschied zwischen einer von-Neumann-Architektur und einer Harvard-Architektur.

Lösung zu Aufgabe 1e)

Von-Neumann:
Programmcode und Daten im selben Speicher.
Zugriff über ein gemeinsames Bussystem.

Harvard:
Programmcode und Daten in getrennten Speichern.
Zugriff über getrennte Bussysteme.

f) Erläutern Sie die Funktion der I, N-, Z- und V-Flags im CCR des 68HCS12.

Lösung zu Aufgabe 1f)

I=1 Interrupts gesperrt (maskiert)
N=1 Ergebnis (als 2er-Komplement-Zahl interpretiert) einer Operation war negativ
Z=1 Ergebnis einer Operation war Null
V=1 Überlauf bei einer Operation (Werte als 2er-Komplement-Zahl interpretiert)

MUSTERPRÜFUNG C		Blatt Nr.:	**4 von 7**
Studiengang:	**Kommunikationstechnik Softwaretechnik Technische Informatik**	Semester:	SWB4, TIB4, KTB4
Prüfungsfach:	**Computerarchitektur 3**	Fachnummer:	**4021**
Hilfsmittel:	**Vorlesungs- und Labormanuskript, Fachliteratur, Taschenrechner**	Dauer:	**90 min**

Aufgabe 2 (35 Punkte):

Im folgenden Ausschnitt aus einem Assemblerprogrammlisting für einen Freescale 68HCS12-Rechner (Codewarrior-Entwicklungsumgebung und Aufrufkonventionen) sehen Sie ein Unterprogramm mit einem 16-Bit Integer-Wert als Parameter und einem 16-Bit Integer-Wert als Rückgabewert. Sie sollen herausfinden, was es macht.

```
MYFUNCTION:             ; unsigned int MYFUNCTION(unsigned int n)
    STD   4,-SP         ; Lokale Variable: unsigned int k =  n    (A)
    LDD   #-1           ; Lokale Variable: unsigned int N = -1
    STD   2,SP          ;                                         (B)
    LDX   0,SP          ; im Unterprogramm: k → SP+0, N → SP+2
L1: BNE   else1         ; if (k==0)
    NEGB                ; { k = +1;
    CLRA
    STD   0,SP
else1:                  ; }
    LDX   0,SP          ; if (k<=8)
    DEX
    CPX   #7
    BHI   return1
    LDAB  #1            ; {   N = 1;                  Reg. D = 1
    CLRA
    LDY   0,SP                                        Reg. Y = k
dowhile1:               ;     do
    TFR   Y,X           ;     {                       Reg X = Y
    EMUL                ;           N = N * k;        (Y,D) = D x Y
    DEX                 ;           k = k - 1;        Reg X = X - 1
    CPX   #1
    TFR   X,Y           ;                             Reg. Y = X
    BGT   dowhile1      ;     } while (k > 1) ;       Reg. X > 1
    STD   2,SP          ;                             N = Reg. D
return1:                ; }
    LDD   2,SP          ; return N;
L2: LEAS  4,SP          ; Lokale Variable vom Stack entfernen
    RTS
```

a) An das Unterprogramm wird ein Parameter übergeben. Wie wird dieser Parameter übergeben?

Übergabe n im Register D, wird bei (A) in lokale Variable auf Stack kopiert

b) Wie wird das Ergebnis des Unterprogramms an den Aufrufer zurückgegeben?

Rückgabewert N in Register D (siehe Zeile nach Label return1)

c) Welcher Wert steht im X-Register, wenn der Programmzeiger auf Label L1: zeigt und der Parameterwert n=6 übergeben wurde?

X = k = n = 6

MUSTERPRÜFUNG C		Blatt Nr.:	**5 von 7**
Studiengang:	**Kommunikationstechnik Softwaretechnik Technische Informatik**	Semester:	SWB4, TIB4, KTB4
Prüfungsfach:	**Computerarchitektur 3**	Fachnummer:	**4021**
Hilfsmittel:	**Vorlesungs- und Labormanuskript, Fachliteratur, Taschenrechner**	Dauer:	**90 min**

d) Die Funktion soll mit dem Parameterwert 3 aufgerufen werden. Zeichnen Sie den Stack mit allen Werten (numerisch wenn möglich, ein Byte pro zeile), wenn der Programm-zeiger auf **Label L2**: zeigt.

Adresse	Wert	Bedeutung
$2FFA	**$00 (MSB)**	**Lokale Variable** (im Unterprogramm nach A → SP+0) **unsigned int k = n (Kopie des Parameters n),** wird nach Zeile A nicht mehr verändert
$2FFB	**$03 (LSB)**	
$2FFC	**$00 (MSB)**	**Lokale Variable** (im Unterprogramm nach A → SP+2) **unsigned int N = n!** Wurde in Zeile B zunächst mit N=-1 initialisiert.
$2FFD	**$06 (LSB)**	
$2FFE	PC MSB	Return-Adresse vom Aufruf von MYFUNCTION
$2FFF	PC LSB	

e) Schreiben Sie auf ein separates Blatt die C-Routine, die zum oben gezeigten Assembler-Programm gehört.

f) Geben Sie hier für die folgenden Parameterwerte die Ergebnisse des Unterprogramms an:

Parameterwert n:		Ergebnis
0	……	**N = 0! = 1**
-6	……	**N = (unsigned int) –1 = 65535** [*1]
6	……	**N = 6! = 720**
10	……	**N = (unsigned int) –1 = 65535**

[*1] Beachte: Die Abfrage BHI im Zweig else1 behandelt negative Zahlen wie große positive Werte!

g) Geben Sie den gültigen Wertebereich für den Parameter an, innerhalb dessen das Ergebnis korrekt dargestellt wird. Was passiert außerhalb dieses Bereiches?
Antwort:

-1 < n < **9** ; außerhalb: **N= (unsigned int) –1 = 65535**

d.h. n = 0 … 8 → N = n! = 1 … 40320 (9!=362 880 nicht mehr mit 16bit darstellbar)

MUSTERPRÜFUNG C		Blatt Nr.:	**6 von 7**
Studiengang:	**Kommunikationstechnik Softwaretechnik Technische Informatik**	Semester:	SWB4, TIB4, KTB4
Prüfungsfach:	**Computerarchitektur 3**	Fachnummer:	**4021**
Hilfsmittel:	**Vorlesungs- und Labormanuskript, Fachliteratur, Taschenrechner**	Dauer:	**90 min**

Aufgabe 3 (35 Punkte):

Der Beeper auf unserem Lautsprecherboard hängt am Ausgang des Timers Kanal 5. In dieser Aufgabe sollen Sie in der Assemblersprache die notwendige Software erstellen, so dass beim Drücken auf den Taster SW1 an Port H.0 der Beeper ertönt. Dazu gehen Sie wie folgt vor:

a) Sie schreiben eine Prozedur `initBeeper`, die den Timer intialisiert. Dazu müssen Sie die Register TSCR1, TIOS und TIE geeignet beschreiben. Alle anderen Register spielen zunächst keine Rolle. Der Timer wird im Output Compare Mode betrieben. Die Taktfrequenz des Timers soll so klein sein wie möglich.

Lösung zu Aufgabe 3a)

```
initBeeper:
      BSET TSCR1, #$80      ; Enable timer module

      MOVB #$07, TSCR2      ; Timer clock period T_TCNT,max = 5,3µs
                            ; (nicht gefordert, Default-Wert 42ns)

      BSET TIOS, #$20       ; Set timer channel 5 to output
                              compare mode

      BSET TIE, #$20        ; Enable interrupt for timer ch. 5

      RTS
```

b) Schreiben Sie zwei Routinen `beeperOn` und `beeperOff`, die den Ausgang des Timer-Kanals 5 aktivieren bzw. stilllegen. Dazu verwenden Sie lediglich das Register TCTL1. Alle anderen Timer-Ausgänge dürfen nicht verändert werden! Setzen Sie den Ausgang so, dass er toggelt, d.h. zwischen 0 und 1 wechselt.

Lösung zu Aufgabe 3b)

```
beeperOn:
      BCLR TCTL1, #%00001000 ; optional, da Default-Wert Bit 3 = 0
      BSET TCTL1, #%00000100 ; On next timer 5 event toggle
      RTS

beeperOff:
      BCLR TCTL1, #%00001100 ; On timer 5 event no output
      RTS
```

MUSTERPRÜFUNG C		Blatt Nr.:	**7 von 7**
Studiengang:	**Kommunikationstechnik Softwaretechnik Technische Informatik**	Semester:	SWB4, TIB4, KTB4
Prüfungsfach:	**Computerarchitektur 3**	Fachnummer:	**4021**
Hilfsmittel:	**Vorlesungs- und Labormanuskript, Fachliteratur, Taschenrechner**	Dauer:	**90 min**

c) Jetzt erstellen Sie die Interrupt-Routine. Diese inkrementiert TC5 immer um einen bestimmten Wert, der die Frequenz festlegt. Wird der Wert vom Zeitgeber erreicht, wird ein Interrupt erzeugt. Die Interrupt-Routine muss also nur den Vergleichswert in TC5 um ein Delta erhöhen und das Interrupt-Flag zurücksetzen. Delta sollte so gewählt werden, dass der Beeper mit ca. 440 Hz pfeift.

Lösung zu Aufgabe 3c)

```
.vect: SECTION
       ORG $FFE4
int13:  DC.W timer5Isr      ; Interrupt vector for timer channel 5

period: EQU  214            ; Signal period 2 * 214 · 5,3µs = 2,27ms
                            ; (ca.440Hz)
.init: SECTION
timer5Isr:
     LDD  #period           ; Add period
     ADDD TC5               ; Set timer for next interrupt
     STD  TC5

     BSET TFLG1, #$20       ; Reset interrupt flag for timer ch. 5
     RTI
```

d) Jetzt schreiben Sie das Hauptprogramm. Zunächst schalten Sie alle Interrupts aus. Sie setzen Port H als Eingang und schalten seine Interrupts aus.

Lösung zu Aufgabe 3d)

```
Main:
Entry:
     LDS  #__SEG_END_SSTACK; Initialize stack pointer
     SEI                   ; Disable interrupts
     MOVB #$00, DDRH       ; Port H as inputs (optional, da Default)
     MOVB #$00, PIEH       ; Port H interrupts off ( - " - )
```

e) Nun initialisieren Sie den Beeper-Timer und schalten die Interrupts wieder ein. Danach gehen Sie in eine Endlosschleife, in der Sie beim Drücken der Taste SW1 den Beeper ertönen lassen.

Lösung zu Aufgabe 3e)

```
     JSR initBeeper
     CLI
Loop:
     BRSET PTH, $01, off   ; Button not pressed = 1
     JSR  beeperOn         ; Turn beeper on, if button pressed
     BRA  Loop
Off:
     JSR  beeperOff        ; Turn beeper off, if button not pressed
     BRA  Loop
```