HOCHSCHULE ESSLINGEN FAKULTÄT INFORMATIONSTECHNIK

Sommersemester 2009		Blatt Nr.:	**1 von 12**
Studiengang:	**Kommunikationstechnik Softwaretechnik Technische Informatik**	Semester:	SWB4, TIB4, KTB4
Prüfungsfach:	**Computerarchitektur 3**	Fachnummer:	**4021**
Hilfsmittel:	**Vorlesungs- und Labormanuskript, Fachliteratur, Taschenrechner**	Dauer:	**90 min**

Aufgabe 1: Verständnisfragen (20 Punkte)

1.1 Erläutern Sie kurz die drei CPU-Sicherheitsmechanismen zur Unterstützung von Betriebssystemen, die ein moderner PC-Prozessor zur Verfügung stellt.

Lösung zu Aufgabe 1.1:

- Privilegstufen (Kernel Mode, User Mode)
- Privilegierte Befehle (z.B. Sperren von Interrupts)
- Speicherverwaltung und Zugriffsschutz (s. 5.7)

1.2 Was unterscheidet eine Load-Store-Architektur von der Architektur des Mikrocontrollers MC9S12DP256 von Freescale?

Lösung zu Aufgabe 1.2:

Operanden von Befehlen außer Load/Store können beim HCS12 auch Speicherzellen sein, während sie sich bei einer reinen Load/Store-Architektur auf Register beschränken.

Bitte geben Sie alle Aufgabenblätter wieder ab!

Sommersemester 2009		Blatt Nr.:	**2 von 12**
Studiengang:	**Kommunikationstechnik Softwaretechnik Technische Informatik**	Semester:	SWB4, TIB4, KTB4
Prüfungsfach:	**Computerarchitektur 3**	Fachnummer:	**4021**
Hilfsmittel:	**Vorlesungs- und Labormanuskript, Fachliteratur, Taschenrechner**	Dauer:	**90 min**

1.3 Erläutern Sie die Funktion des folgenden Befehls für einen Mikrocontroller mit ARM-Architektur: ADDNE R4, R6, #21.

Lösung zu Aufgabe 1.3:

R4 = R6 + 21, aber nur, wenn die vorhergehende Operation das Flag für NE gesetzt hat (oder Zero-Flag ist zurück gesetzt)

1.4 Erläutern Sie die Funktion des folgenden Befehls für einen Mikrocontroller mit ARM-Architektur: STR R4,[SP, #-4]!

Lösung zu Aufgabe 1.4:

```
SP <- SP – 4 (Register Write Back)
R4 -> Stack
```

1.5 Wo wird bei einem Mikrocontroller mit ARM-Architektur die Rückkehradresse bei einem Unterprogrammaufruf gespeichert?

Lösung zu Aufgabe 1.5:

In R14 = LR

Bitte geben Sie alle Aufgabenblätter wieder ab!

Sommersemester 2009		Blatt Nr.:	**3 von 12**
Studiengang:	**Kommunikationstechnik Softwaretechnik Technische Informatik**	Semester:	SWB4, TIB4, KTB4
Prüfungsfach:	**Computerarchitektur 3**	Fachnummer:	**4021**
Hilfsmittel:	**Vorlesungs- und Labormanuskript, Fachliteratur, Taschenrechner**	Dauer:	**90 min**

1.6 Das Auf- bzw. Entladen eines Kondensators über einen Widerstand lässt sich über die Gleichung

$$u(t) = (U_E - U_A) \cdot (1 - e^{-t/RC}) + U_A$$

beschreiben, wobei U_E die Endspannung für t gegen Unendlich und U_A die Ausgangsspannung ist (siehe Vorlesungen Elektronik und Elektrotechnik).

Der Pulsweitenmodulatorausgang des Mikrocontrollers sei nun mit einem RC-Glied als Tiefpassfilter verbunden, die Ausgangsspannung wird am Kondensator gemessen. Bei welchem Tastverhältnis des pulsweitenmodulierten Signals tritt am Kondensator die maximale Welligkeit auf, d.h. Differenz zwischen minimaler und maximaler Ausgangsspannung im eingeschwungenen Zustand? Bitte begründen.

Lösung zu Aufgabe 1.6:

Differenz maximal, wenn Aufladezeit = Entladezeit,
d.h. Duty Cycle = 50%

1.7 Berechnen Sie näherungsweise die maximale Welligkeit der Ausgangsspannung gemäß Aufgabe 1.6 in Abhängigkeit von der Zeitkonstanten RC sowie der Periodendauer T_P des pulsweitenmodulierten Signals. Nehmen Sie dabei an, dass die Welligkeit klein gegenüber der maximalen Ausgangsspannung ist. Wie viel mal größer als T_P muss RC sein, damit die Welligkeit 1% der maximalen Ausgangsspannung nicht überschreitet?

Lösung zu Aufgabe 1.7:

RC = 25 T_P

Bitte geben Sie alle Aufgabenblätter wieder ab!

Sommersemester 2009		Blatt Nr.:	**4 von 12**
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: Programmanalyse (30 Punkte)

Das folgende Assemblerlisting stellt eine Funktion dar, die von einem C-Programm aufgerufen werden kann. Das Programm wurde für den Codewarrior-Compiler/Assembler geschrieben. Die C-Prototyp-Definitionen sehen so aus:

```
int f1(unsigned char array[], unsigned char size);
```

Listing zu Aufgabe 2:

```
1          STAB  3,-SP
2          TSTB
3          BHI   L1
4          LDD   #-1
5          BRA   Ende
6    L1:   CMPB  #10
7          BLS   L2
8          LDD   #-2
9          BRA   Ende
11   L2:   CLRB
12         CLRA
13         STD   1,SP
14         BRA   LoopStart
15 LoopBody:
16         CLRA
17         PSHB
18         ADDD  6,SP
19         TFR   D,X
20         LDAB  0,X
21         CLRA
22         ADDD  2,SP
23         STD   2,SP
24         PULB
25         INCB
26 LoopStart:
27         CMPB  0,SP
28         BCS   LoopBody
29         LDD   1,SP
30 Ende:   LEAS  3,SP
31         RTS
```

Bitte geben Sie alle Aufgabenblätter wieder ab!

Sommersemester 2009		Blatt Nr.:	**5 von 12**
Studiengang:	**Kommunikationstechnik Softwaretechnik Technische Informatik**	Semester:	SWB4, TIB4, KTB4
Prüfungsfach:	**Computerarchitektur 3**	Fachnummer:	**4021**
Hilfsmittel:	**Vorlesungs- und Labormanuskript, Fachliteratur, Taschenrechner**	Dauer:	**90 min**

2.1 Zeichnen Sie den Stack mit Stackzeiger und Werten nachdem die erste Zeile der Funktion ausgeführt wurde. Markieren Sie die Position des Stackzeigers mit einem X in der Adressenspalte:

Adresse		Wert	Bedeutung
$2FF4			
$2FF5			
$2FF6		**B (saved)**	**Schleifenindex**
$2FF7	X	**Size**	
$2FF8		**Summe MSB**	
$2FF9		**Summe LSB**	
$2FFA		Return-Adress MSB	Rücksprungadresse MSB
$2FFB		Return-Adress LSB	Rücksprungadresse LSB
$2FFC		**Zeiger auf array[0], MSB**	
$2FFD		**Zeiger auf array[0], LSB**	
$2FFE			
$2FFF			
$3000			

2.2 Welche Bedeutung hat der erste Befehl in Zeile 1?

Lösung zu Aufgabe 2.2:

Er schafft Platz für das Ergebnis und speichert den zweiten Parameter auf dem Stack.

Bitte geben Sie alle Aufgabenblätter wieder ab!

Sommersemester 2009		Blatt Nr.:	**6 von 12**
Studiengang:	**Kommunikationstechnik Softwaretechnik Technische Informatik**	Semester:	SWB4, TIB4, KTB4
Prüfungsfach:	**Computerarchitektur 3**	Fachnummer:	**4021**
Hilfsmittel:	**Vorlesungs- und Labormanuskript, Fachliteratur, Taschenrechner**	Dauer:	**90 min**

2.3 Beim Aufruf der Funktion wird ein Zeiger auf ein Array übergeben. Das Array ist in der aufrufenden Funktion so definiert:

unsigned char array[10] = {1,2,3,4,5,6,7,8,9,10}

Die Funktion wird so aufgerufen: retWert = f1(array,10);

Wird in diesem Fall die Zeile 8 ausgeführt? Bitte begründen.

Lösung zu Aufgabe 2.3:

Nein, nur falls der zweite Parameter > 10

2.4 Welche Funktion(en) übernimmt das B-Register zwischen Zeile 14 und Zeile 28?

Lösung zu Aufgabe 2.4:

Schleifenzähler und Summand

2.5 Schreiben Sie in der Programmiersprache C in Form einer einzigen Befehlszeile auf, was der Maschinencode zwischen Zeile 18 und Zeile 23 macht.

Werte und Lösung zu Aufgabe 2.5:

```
result = result + array[i]
```

Bitte geben Sie alle Aufgabenblätter wieder ab!

Sommersemester 2009		Blatt Nr.:	**7 von 12**
Studiengang:	**Kommunikationstechnik Softwaretechnik Technische Informatik**	Semester:	SWB4, TIB4, KTB4
Prüfungsfach:	**Computerarchitektur 3**	Fachnummer:	**4021**
Hilfsmittel:	**Vorlesungs- und Labormanuskript, Fachliteratur, Taschenrechner**	Dauer:	**90 min**

2.6 Die Funktion wird nun so aufgerufen:

retWert = f1(array,-8);

Was steht im D-Register vor Ausführen von Zeile 31?

Lösung zu Aufgabe 2.6:

-1

2.7 Die Funktion wird nun so aufgerufen:

retWert = f1(array,12);

Was steht im D-Register vor Ausführen von Zeile 31?

Lösung zu Aufgabe 2.7:

-2

2.8 Die Funktion wird nun so aufgerufen (array ist wie in Aufgabe 2.3 definiert):

retWert = f1(array, sizeof(array)/sizeof(char));

Was steht im D-Register vor Ausführen von Zeile 31?

Lösung zu Aufgabe 2.8:

55

Sommersemester 2009		Blatt Nr.:	**8 von 12**
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: Adressierungsarten und Stack (25 Punkte):

3.1 In einem HCS12-Assemblerprogramm sind folgende globalen Variablen definiert:

```
.data       SECTION
            ORG $1000
var1 :      DS.W 3
.const:     SECTION
            ORG   $D000
const1:     DC.B  $02, $01, $00, $03
tabelle1:   DC.W  $D004, $D006
tabelle2:   DC.B  $D0, $02, $33, $44, $55, $66, $77, $88
```

Geben Sie den Inhalt der CPU-Register D, X und Y nach jedem Assemblerbefehl an, wenn das folgende Programm ausgeführt wird. Es reicht aus, wenn Sie bei jedem Befehl diejenigen Registerwerte eintragen, die sich jeweils ändern.

Assemblerbefehle	D	X	Y
	$AA55	**$0000**	**$0000**
LDD #tabelle2	$D008		
LDX #var1		**$1000**	
LDY tabelle1			**$D004**
MOVW 2,Y+,2,X+		**$1002**	**$D006**
MOVW 2,Y+,2,-X		**$1000**	**$D008**
LDD var1	**$D006**		
LDX const1+2		**$0003**	
DEX		**$0002**	
ADDD const1,X	**$D009**		
LDD tabelle1	**$D004**		
LDY const1,X			**$0003**
INY			**$0004**
LDD [D, Y]	**$0003**		

Bitte geben Sie alle Aufgabenblätter wieder ab!

Sommersemester 2009		Blatt Nr.:	**9 von 12**
Studiengang:	**Kommunikationstechnik Softwaretechnik Technische Informatik**	Semester:	SWB4, TIB4, KTB4
Prüfungsfach:	**Computerarchitektur 3**	Fachnummer:	**4021**
Hilfsmittel:	**Vorlesungs- und Labormanuskript, Fachliteratur, Taschenrechner**	Dauer:	**90 min**

3.2
In einem C-Programm seien die folgenden globalen Variablen definiert:

```
int n, max, m;
```

Diese Variablen werden im folgenden Ausschnitt des C-Programms verwendet, das Sie „von Hand“ in die entsprechenden HCS12-Assemblerbefehle übersetzen sollen. Die Definition der globalen Variablen muss nicht übersetzt werden. Assemblerdirektiven wie XDEF, XREF, INCLUDE, SECTION usw. dürfen weggelassen werden.

a) Geben Sie den Assembler-Programmcode an:

Lösung zu Aufgabe 3.2 a:

C-Programm

```
//***** Hauptprogramm *****
void main(void)
{   . . .
    n = 2;
    max = 5;
    m = fac(n, max);
    . . .
}



//***** Unterprogramm *****
int fac(int n, int max)
{
    if (n > 1) {
        return n*fac(n-1, max);
    }
    else {
       return 1;
    };
}
```

HCS12-Assembler-Programm

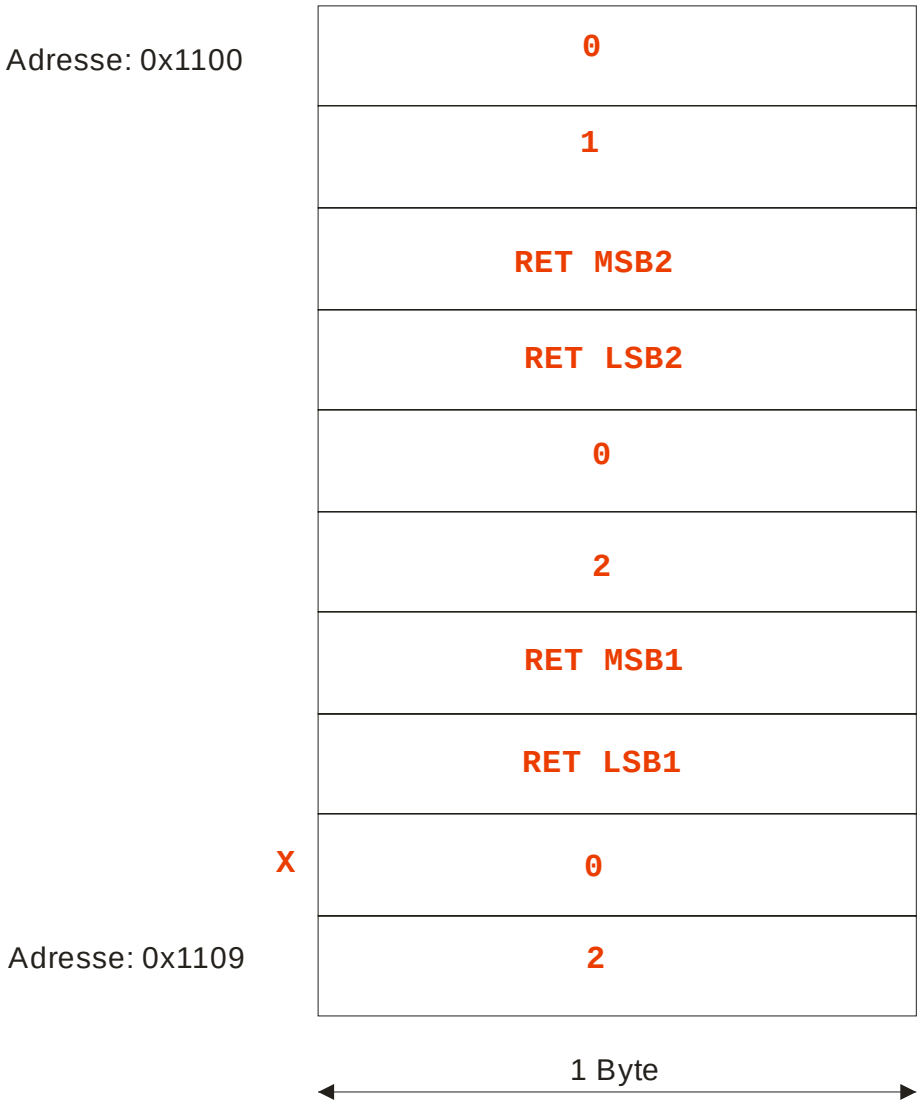
```
  LDAB #2
  CLRA
  STD  n
  LDAB #5
  STD max
  ASRB (oder LDAB #2)
  PSHD
  LDAB #5
  JSR fac
  STD m


fac:
  LDX 2,SP
  CPX #1
  BLE N1
  LEAY -1,X (oder Subtr.)
  PSHY
  BSR fac ; Rekursion!
  PULY
  LDY 2,SP
  EMUL
  RTS

N1:
  LDAB #1
  CLRA
  RTS
```

Bitte geben Sie alle Aufgabenblätter wieder ab!

Sommersemester 2009		Blatt Nr.:	**10 von 12**
Studiengang:	**Kommunikationstechnik Softwaretechnik Technische Informatik**	Semester:	SWB4, TIB4, KTB4
Prüfungsfach:	**Computerarchitektur 3**	Fachnummer:	**4021**
Hilfsmittel:	**Vorlesungs- und Labormanuskript, Fachliteratur, Taschenrechner**	Dauer:	**90 min**

b) Tragen Sie in die folgende Tabelle den Zustand des Stacks direkt vor der Rückkehr ins Hauptprogramm ein. Zeigen Sie alle durch die Funktionsaufrufe von fac auf den Stack gelegten Daten. Geben Sie an, auf welche Speicherzelle der Stack Pointer zu diesem Zeitpunkt zeigt.

Lösung zu Aufgabe 3.2b:



Bitte geben Sie alle Aufgabenblätter wieder ab!

Sommersemester 2009		Blatt Nr.:	**11 von 12**
Studiengang:	**Kommunikationstechnik Softwaretechnik Technische Informatik**	Semester:	SWB4, TIB4, KTB4
Prüfungsfach:	**Computerarchitektur 3**	Fachnummer:	**4021**
Hilfsmittel:	**Vorlesungs- und Labormanuskript, Fachliteratur, Taschenrechner**	Dauer:	**90 min**

Aufgabe 4: HCS12-Peripheriebausteine (25 Punkte):

4.1 Schreiben Sie ein Assemblerprogramm „beep“, das auf dem im Labor verwendeten Board mit 24 MHz Busfrequenz den Beeper zum Klingen bringt. Der Beeper soll permanent abwechselnd für jeweils eine Sekunde einen Ton mit 440 Hz und 660 Hz erzeugen. Sie dürfen nur genau einen Timer verwenden, die Sekunde soll genau eingehalten werden.

Schreiben Sie in HCS12-Assemblersprache eine Routine initBeep(), die die Hardware so initialisiert, dass der erste Ton mit 440 Hz erzeugt wird.

Lösung zu Frage 4.1: **(siehe 3.17)**

```
…
F440: EQU (24000000/128/880);   =213
F660: EQU (24000000/128/1320);  = 142

.data   SECTION
Delay:  ds.w
OneSec: ds.w

.init SECTION

initBeep:
   BSET TSCR1, #80
   MOVB #$07, TSCR2
   BSET TIOS, #$20
   MOVB #$04,TCTL1
   LDD TCNT

   LDX #F440
   STX Delay
   LDX #880
   STX OneSec

   ADDD Delay
   STD TC5
   BSET TIE, #20
   RET
```

Bitte geben Sie alle Aufgabenblätter wieder ab!

Sommersemester 2009		Blatt Nr.:	**12 von 12**
Studiengang:	**Kommunikationstechnik Softwaretechnik Technische Informatik**	Semester:	SWB4, TIB4, KTB4
Prüfungsfach:	**Computerarchitektur 3**	Fachnummer:	**4021**
Hilfsmittel:	**Vorlesungs- und Labormanuskript, Fachliteratur, Taschenrechner**	Dauer:	**90 min**

4.2 Schreiben Sie in HCS12-Assemblersprache eine Interruptservice-Routine „isrBeep()“ für die oben initialisierte Hardware, die die unter 4.1 beschriebene Funktion realisiert.

Lösung zu Frage 4.2:

```
isrBeep:
   LDD TC5
   ADDD Delay
   STD TC5
   BSET TFLG1, #20
   LDX OneSec
   DBNE X, retisr
switchFreq:
   CPD #F440
   BEQ Set660
Set440:
   LDD #F440
   STD Delay
   LDD #880
   STD OneSec
   BRA retisr
Set660:
   LDD #F660
   STD Delay
   LDD #1320
   STD OneSec
retisr:
   RTI
```

4.3 Schreiben Sie das Hauptprogramm, dass mit Hilfe der obigen Routinen die Hardware initialisiert und die Erzeugung des Signals startet. Tragen Sie die Adresse der Interruptservice-Routine mit Hilfe von Pseudo-Assemblerbefehlen in die Interruptvektortabelle ein.

Lösung zu Frage 4.3:

```
.vect: SECTION
       ORG $FFE4
int13:  dc.w isrBeep
.init  SECTION ; steht schon weiter oben, hier nur wiederholt
    JSR initBeep;
loop:  BRA loop
```

Bitte geben Sie alle Aufgabenblätter wieder ab!