HOCHSCHULE ESSLINGEN FAKULTÄT INFORMATIONSTECHNIK

Sample Exam A		Page #:	**1 of 11**
Studiengang:	**Kommunikationstechnik KTB** **Softwaretechnik SWB** **Technische Informatik TIB** **Ingenieurpädagogik IEP**	Semester:	IT4
Prüfungsfach:	**Computerarchitektur 3**	Exam ID:	**4022, 1054003**
Ressources:	**Lecture and Lab Manuscript** **Books, Pocket Calculator**	Duration: Prof:	**90 min** Zimmermann

Please enter your name here: **Total: 90 Points**

Given Name:	Family Name:	Student ID:

Solution hints (no guarantee for correctness)

Please use the free space on these sheets for your solution. Solutions may be in English or German. If space is not sufficient, please use the backside or additional sheets.

(Note: This is a translated version, the original exam was in German).

Problem 1: Miscellaneous (20 Points)

1.1 List the various memory areas in the HCS12's memory map (name and purpose of the memory):

On-Chip-Peripherials
EEPROM for persistent variables
RAM for stack and variables
Flash-ROM for program code and constants

1.2 What is the purpose of the 5 lower bits in HCS12's Condition Code Register.

Interrupt mask: Disable on-chip interrupts
Negative: Indicates negative instruction result
Zero: Indicates zero instruction result
Overflow: Indicates carry for instructions with signed values
Carry: Dito for unsigned values

Sample Exam A		Page #:	**2 of 11**
Studiengang:	**Kommunikationstechnik KTB** **Softwaretechnik SWB** **Technische Informatik TIB** **Ingenieurpädagogik IEP**	Semester:	IT4
Prüfungsfach:	**Computerarchitektur 3**	Exam ID:	**4022, 1054003**
Ressources:	**Lecture and Lab Manuscript** **Books, Pocket Calculator**	Duration: Prof:	**90 min** Zimmermann

1.3 HCS12 microcontrollers provide various addressing modes for operands. The following instructions use variables `var1: ds.w` and `const: dc.w 1`. For each operand specify the associated addressing mode. One of the combinations (instruction plus addressing mode) is invalid. Mark it with a cross X.

	1st Operand	2nd Operand
`TFR D,X`	explicit reg.addr.	explicit reg.addr.
`LDD const`	implicit reg.addr.	direct addressing
`STD #const`	XXX: immediate addr. not allowed as destination	
`MOVB 1,X,0,X`	reg.indirect w/ index	reg.indirect w/ index
`LDD [var1,X]`	implicit reg.addr.	mem.indirect w/ index
`LDAA 1,Y+`	implicit reg.addr.	reg.indirect post-ink.
`LDX const,Y`	implicit reg.addr.	reg.indirect w/ index

1.4 What is the purpose of HCS12 registers DDRH, PPSH und PERH?

DDRH:
Data Direction Register
Defines whether a port pin acts as input or output

PPSH:
Port Polarity Select
Defines, whether a pull-up or pull-down resitor shall be used for a digital input pin (plus trigger edge direction for port H)

PERH:
Port Enable pull up/down Resistor
Turns pull up/down resistors on

Sample Exam A		Page #:	**3 of 11**
Studiengang:	**Kommunikationstechnik KTB** **Softwaretechnik SWB** **Technische Informatik TIB** **Ingenieurpädagogik IEP**	Semester:	IT4
Prüfungsfach:	**Computerarchitektur 3**	Exam ID:	**4022, 1054003**
Ressources:	**Lecture and Lab Manuscript** **Books, Pocket Calculator**	Duration: Prof:	**90 min** Zimmermann

1.5 Which conditions are required for an interrupt service routine, e.g. for port H, to be called periodically?

I-Bit in CCR must be 0 (interrupt mask cleared)
ISR address entry in Interrupt-Vector-Table
Port H pin(s) configured for interrupts (PIEHx = 1)
Reset Interrupt Flag (PIFH) within ISR
Periodical hardware trigger signal on port H pin(s)

1.6 Assume, that the HCS12 uses a crystal clock frequency of 8 MHz. Which value do you have to configure in RTICTL for the Real Time Interrupt Modul (RTI) to generate interrupts every 100 ms?
If an alarm clock is driven by this interrupt, which time error (in minutes) would this clock have after 24 hours?

RTICTL (binär oder hex): _____

Time error after 24 h: _____

Required f_{RTI} = 10 Hz = f_{oscclk} / (2^{9+x} (Y+1)) (see e.g. pg. 3.12)

Select x=7 → Y = 11

RTICTL = 0111 1011 (7B)

Actual clock frequency f_{RTI} = 10,17253 Hz, i.e. 1,7% too fast.

After 24 hours : 0.017*24*60 min = 24,5 minutes time offset

Sample Exam A		Page #:	**4 of 11**
Studiengang:	**Kommunikationstechnik KTB** **Softwaretechnik SWB** **Technische Informatik TIB** **Ingenieurpädagogik IEP**	Semester:	IT4
Prüfungsfach:	**Computerarchitektur 3**	Exam ID:	**4022, 1054003**
Ressources:	**Lecture and Lab Manuscript** **Books, Pocket Calculator**	Duration: Prof:	**90 min** Zimmermann

Problem 2: Program Analysis (30 Points)

The following HCS12 assembler listing shows two functions which can be called from a C program. The associated C prototypes are

```
int f1(char *a1, char *a2);              strcpy a2→a1
void f2(void *a1, void *a2, int a3);     memcpy a1→a2
```

```
1     f1:  TFR   D,  X              ; X = a2
2          LDY   +2, SP             ; Y = a1
3          CLRA                     ; D = 0
4          CLRB
5
6     la:  ADDD  #1                 ; D++
7          MOVB  1, X+, 1, Y+       ; *Y++ = *X++       (*X → *Y)
8          TST   -1, X              ; Nullbyte?
9          BNE   la                 ; if no, proceed with
10                                  ; ... next byte
11         RTS                      ; return byte count

12    ;-----------------------------------------------------

13    f2:  LDY   +2, SP             ; Y = a2
14         LDX   +4, SP             ; X = a1
15
16         CPD   #0                 ; a3==0 ?
17         BEQ   lc                 ; ... if yes, return
18
19    lb:  MOVB  1, X+, 1, Y+       ; *Y++ = *X++
20         SUBD  #1                 ; a3--
21         BNE   lb                 ; if a3>0 proceed with
22                                  ; ... next byte
23    lc:  RTS                      ; return
```

2.1 The first function will be called as: `e = f1(0x1234, 0x2345)`. What is the contents of register X after execution of line 1?

X = a2 = $2345

Sample Exam A		Page #:	**5 of 11**
Studiengang:	**Kommunikationstechnik KTB** **Softwaretechnik SWB** **Technische Informatik TIB** **Ingenieurpädagogik IEP**	Semester:	IT4
Prüfungsfach:	**Computerarchitektur 3**	Exam ID:	**4022, 1054003**
Ressources:	**Lecture and Lab Manuscript** **Books, Pocket Calculator**	Duration: Prof:	**90 min** Zimmermann

2.2 Function f1 now will be called as: `e = f1(0x1000, 0x2000).` The following table shows the memory content, before the function is called. Please fill out the column showing the values, when the function returns.

Memory address		Values before call	Values after call
1000h	a1[0]	1	20 (a2[0])
1001h	. . .	2	30 (a2[1])
1002h	. . .	3	0 (a2[2])
1003h	a1[3]	4	4 (no change)
…			
2000h	a2[0]	20	20 (no change)
2001h	. . .	30	30 (no change)
2002h	. . .	0	0 (no change)
2003h	. . .	-4	-4 (no change)
2004h	a2[4]	20	20 (no change)

2.3 What is the contents of register D before line 11 is executed, if the function is called as in question 2.2?

D = 3

2.4 The second function will be called as follows: `f2(0x1000, 0x2000, 4).` What is the contents of register X after line 14 was executed?

X = a1 = $1000

Sample Exam A		Page #:	**6 of 11**
Studiengang:	**Kommunikationstechnik KTB** **Softwaretechnik SWB** **Technische Informatik TIB** **Ingenieurpädagogik IEP**	Semester:	IT4
Prüfungsfach:	**Computerarchitektur 3**	Exam ID:	**4022, 1054003**
Ressources:	**Lecture and Lab Manuscript** **Books, Pocket Calculator**	Duration: Prof:	**90 min** Zimmermann

2.5 Please fill out the following table, when function f2 returns after being called as in question 2.4.

Memory address		Values before call	Values after call
1000h	a1[0]	1	1 (no change)
1001h	. . .	2	2 (no change)
1002h	. . .	3	3 (no change)
1003h	a1[3]	4	4 (no change)
. . .			
2000h	a2[0]	20	1 (a1[0])
2001h	. . .	30	2 (a1[1])
2002h	. . .	0	3 (a1[2])
2003h	. . .	-4	4 (a1[3])
2004h	a2[0]	20	20 (no change)

2.6 Does function f2 work correctly, if a3 is null hat or negative? If not, what should be changed?

null is okay (line 17), negative values won't work correctly
Should also check for negative values in line 17.

Sample Exam A		Page #:	**7 of 11**
Studiengang:	**Kommunikationstechnik KTB** **Softwaretechnik SWB** **Technische Informatik TIB** **Ingenieurpädagogik IEP**	Semester:	IT4
Prüfungsfach:	**Computerarchitektur 3**	Exam ID:	**4022, 1054003**
Ressources:	**Lecture and Lab Manuscript** **Books, Pocket Calculator**	Duration: Prof:	**90 min** Zimmermann

Problem 3: Addressing Modes and Stack (25 Points):

3.1
An HCS12 assembler program defines the following global variables:

```
.const:     SECTION
            ORG   $D000
tabelle1:   DC.B  $11, $22, $33, $44, $55, $66, $77, $88
tabelle2:   DC.W  $D002, $D004
```

Please specify the contents of CPU registers D, X and Y after execution of the instruction for each row in the following table. Leave fields empty, if the register contents does not change.

Assemblerbefehle	**D**	**X**	**Y**
	$0000	$0000	$0000
LDX #2		$2	
LDD tabelle1, X	$3344		
LDX tabelle1		$1122	
LDY #tabelle1			$D000
LDAA 1, Y+	$1144		$D001
LDAA 2, +Y	$4444		$D003
LDAA 1, -Y	$3344		$D002
LDX 3, Y		$6677	
LDX -1, Y		$2233	
LEAY 2, +Y			$D004
LDX #tabelle2		$D008	
LDD [2, X]	$5566		

Sample Exam A		Page #:	**8 of 11**
Studiengang:	**Kommunikationstechnik KTB** **Softwaretechnik SWB** **Technische Informatik TIB** **Ingenieurpädagogik IEP**	Semester:	IT4
Prüfungsfach:	**Computerarchitektur 3**	Exam ID:	**4022, 1054003**
Ressources:	**Lecture and Lab Manuscript** **Books, Pocket Calculator**	Duration: Prof:	**90 min** Zimmermann

3.2

A C-program defines the following global variables:

```
int valA, valB, valC;
int m;
```

These variables are used in the following C code, which you have to compile “manually” into the associated HCS12 assembler instructions.

Note: Assembler directives XDEF, XREF, INCLUDE, SECTION etc. may be omitted.

a) Compile the C program into HCS12 assembler instructions

C-Program

```
//***** Main Program *****
void main(void)
{    . . .
     m = add3(valA, valB, valC);
     . . .
}
```

```
//***** Subroutine *****
int add3(int a, int b, int c)
{
     return a + b + c;
}
```

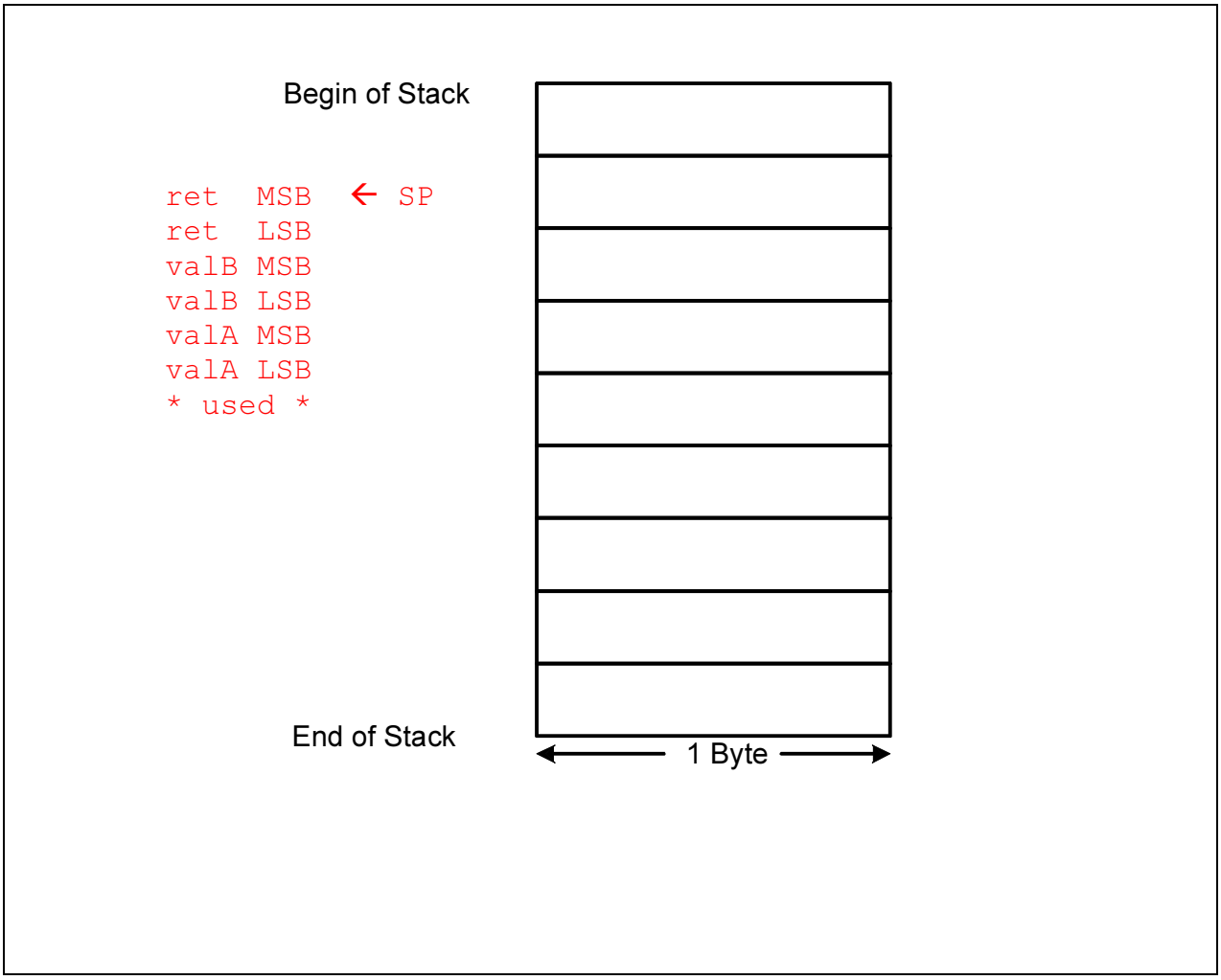
HCS12-Assembler-Program

```
      LDD   valC
      LDX   valB
      LDY   valA
      PSHY
      PSHX
      JSR   add3
      LEAS  4,SP
      STD   m
```

```
add3:
      ADDD  4,SP
      ADDD  2,SP
      RTS
```

Sample Exam A		Page #:	**9 of 11**
Studiengang:	**Kommunikationstechnik KTB** **Softwaretechnik SWB** **Technische Informatik TIB** **Ingenieurpädagogik IEP**	Semester:	IT4
Prüfungsfach:	**Computerarchitektur 3**	Exam ID:	**4022, 1054003**
Ressources:	**Lecture and Lab Manuscript** **Books, Pocket Calculator**	Duration: Prof:	**90 min** Zimmermann

b) Specify the state of the stack at the begin of subroutine **`add3()`**:



Sample Exam A		Page #:	**10 of 11**
Studiengang:	**Kommunikationstechnik KTB** **Softwaretechnik SWB** **Technische Informatik TIB** **Ingenieurpädagogik IEP**	Semester:	IT4
Prüfungsfach:	**Computerarchitektur 3**	Exam ID:	**4022, 1054003**
Ressources:	**Lecture and Lab Manuscript** **Books, Pocket Calculator**	Duration: Prof:	**90 min** Zimmermann

Problem 4: HCS12 Signal Generator (25 Points):

4.1 Write an HCS12 assembler subroutine **`initSCI1`**, which shall configure the HCS12's serial interface SCI1 in such a way, that periodically sending hex character "0x55" generates a square wave signal with 440 Hz. The serial interface shall use transmit interrupts. The function does not have parameters or return values.

Note: A square wave signal consists of sequence of 1-0- and a 0-1 transitions. For each signal period the serial interface has to output two bits (bits, not bytes!).

```
2 bit shall take 1/440 s → T_Bit = 1/880 s (bit length = 1/bit rate).
=> SCI1BD = 24 MHz / 16 * T_Bit = 1705

initSCI1:
     MOVW #1705, SCI1BD          ; Initialize bit rate
     MOVB #0, SCI1CR1            ; 8bit, no parity, 1 stop bit
     MOVB #%10001000, SCI1CR2    ; Transmit enable
                                 ; Transmit Interrupt enable
     RTS
```

4.2 Write the HCS12 assembler interrupt service routine **`isrSCI1`** to be used with the code in question 4.1. On each transmit interrupt event, the ISR shall send hex character "0x55".

```
isrSCI1:
     LDAA SCI1SR1          ; Read status register to reset interrupt
                                flag

     MOVB #$55, SCI1DRL    ; send next character
     RTI
```

Sample Exam A		Page #:	**11 of 11**
Studiengang:	**Kommunikationstechnik KTB** **Softwaretechnik SWB** **Technische Informatik TIB** **Ingenieurpädagogik IEP**	Semester:	IT4
Prüfungsfach:	**Computerarchitektur 3**	Exam ID:	**4022, 1054003**
Ressources:	**Lecture and Lab Manuscript** **Books, Pocket Calculator**	Duration: Prof:	**90 min** Zimmermann

4.3 Write the HCS12 assembler main program, which shall initialize the serial interface and start generating the 440 Hz square wave signal. Don't forget the required assembler directives like SECTIONS etc including the entry of the ISR in the interrupt vector table.

```
.vect: SECTION
       ORG $FFD4
int21: DC.W isrSCI1
.init  SECTION
main:
     LDS #SEG_END_SSTACK   ; Initialize stack
     CLI                   ; Global interrupt enable
     JSR  initSCI1         ; Initialize serial interface

(    LDAA SCI1SR1          ; Optional: Read status register )

     MOVB #$55, SCI1DRL    ; Send first character

loop:                      ; Infinite loop
     BRA loop
```