HOCHSCHULE ESSLINGEN FAKULTÄT INFORMATIONSTECHNIK

Sample Exam F		Page #:	**1 of 9**
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 (Σ 10 Points)

1.1 (3 Points)

Name the 3 most important aspects of a microcontroller's programming model.

Register set
Instruction set
Addressing modes

1.2 (4 Points)

Subroutine parameters can be passed via CPU-registers or via the stack. What are the main adavantages and disadvantages?

Advantage:

Very flexible, as the stack space >> register space

Disadvantage:

Parameter passing via the stack is much slower. Accessing parameters on the stack is error prone (for human programmers).

1.3

Why is it difficult for C compilers and assembler programmers, if a CPU's address word size is bigger than its data word size? (3 Points)

Using pointers, especially pointer calculation gets complicated, if pointers don't fit into normal data registers.
(E.g. Intel's first PC CPU generation 8086 was a bad example. The 8086 is a 16 bit CPU, i.e. 16 bit ALU and registers, but had 20 bit addresses. Its 16bit successor 80286 had 24 bit addresses. Thus, pointer arithmetic was done in two steps with the lower 16bit first and then with the upper 4 / 8 bits)

Sample Exam F		Page #:	**2 of 9**
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: Addressing Modes (Σ 30 Points)

2.1 (8 Points)

Name the addressing mode for each of the following HCS12 assembler instructions.

Instruction	1st Operand	2nd Operand
`LDX #$D0F0`	Implicit Register Addressing for X	Immediate Addressing
`STD 2, X`	Implicit Register Addressing for D	Register-Indirect Addressierung with Index (Offset)
`LDY $C000`	Implicit Register Addressing for Y	Direct Addressing
`TST 1, -X`	Implicit Constant 0	Register-Indirect Addressing with Pre-Decrement
`LDD [1, Y]`	Implicit Register Addressing for D	Memory-Indirect with Index

2.2

What is wrong with the following HCS12 instructions?

(4 Points)

`LDAB #$DCAB`	An 8 bit Register cannot store a 16bit value
`STD #$55AA`	A constant must not be the destination of an instruction

2.3

A HCS12 assembler program uses the following data definitions:

```
.const:     SECTION
            ORG   $D000
value1:     DC.B  $A0, $B0, $C0, $D0, $E0, $F0, $F1, $F2
value2:     DC.W  $A1B2, $C3D4
value3:     DC.L  $87654321

.data:      SECTION
            ORG   $2000
p:          DS.W 1
```

Sample Exam F		Page #:	**3 of 9**
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

For each row in the following table specify the contents of CPU registers D, X and Y, after the instructions in the left field of the row have been executed. Fields, which do not change values, may be left empty. Mark unknown values as "???" if necessary. (18 Points)

Assembler Instructions	**D**	**X**	**Y**
`Initial values`	`$1234`	`$5678`	`$ABCD`

`LDD value1`
`LDX value2`
`LDY value3`
	`$A0B0`	`$A1B2`	`$8765`

`LDAB value1`
`LDX value1+3`
`LDY #value1`
	`$A0A0`	`$D0E0`	`$D000`

`LDD 1, Y` `LDX 4, +Y`	`$B0C0`	`$E0F0`	`$D004`

`LDY #value1`
`LEAX 2, Y`
`LDD 0, Y`
	`$A0B0`	`$D002`	`$D000`

`MOVW #value3, p`
`LDX p`
`LDY #p`
`LDD [0,Y]`
	`$8765`	`$D00C`	`$2000`

`MOVW #$3355, p`
`LDX #p`
`LDAA +1, X`
`LDAB 1, X+`
`TFR A, Y`
	`$5533`	`$2001`	`$0055`

`LDD #$1122`
`LDX #$3344`
`LDY #$5566`
`PSHD`
`PSHX`
`PSHY`
`PULB`
`PULA`
`PULX`
`PULY`
	`$6655`	`$3344`	`$1122`

`MOVW #$1234, value2`
`LDD value2`
`LDX #$D008`
`LDY $D000`
	`$A1B2`
	`(value2`
	`is in`
	`ROM !!!)`
		`$D008`	`$A0B0`

Sample Exam F		Page #:	**4 of 9**
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: Program Analysis (Σ 30 Points)

The following code listing shows the C-code of a HCS12 program:

```
void subA(char *p);
unsigned char subB(char *p, unsigned char val);

char array1[80] = "Hallo-Welt!!!";
char array2[7]  = "abcdef";
unsigned char i;

void main(void)
{     . . .
      subA(array1);
      . . .
      i = subB(array2, 39);
      . . .
}
```

The respective subroutinges are in HCS12 assembler code:

```
subA: PSHX
      TFR  D,X
L2:   LDAA 0,X
      TSTA
      BEQ  L0
      CMPA #'a'
      BLO  L1
      CMPA #'z'
      BHI  L1
      ADDA #-$20
L1:   STAA 1,X+
      BRA  L2
L0:   PULX
RTS
```

```
tab:  DC.B "0123456789ABCDEFGHIJK"

subB:PSHD
      PSHX
      PSHY
M0:   LDY  8, SP
      CLRA
      STAA 2, Y
      LDX  #10
      IDIV
M1:   PSHD
      LDAA tab, X
      STAA 0, Y
      PULX
      LDAA tab, X
      STAA 1, Y
M2:   PULY
      PULX
      PULD
      RTS
```

3.1

What is the contents of registers A and X, when the program has executed the instruction at `L2` in subroutine `subA()` for the first time?

(4 Points)

A = `$48 = 'H'` **X =** `&array1[0]`

Sample Exam F		Page #:	**5 of 9**
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
What is the contents of `array1` an, when `subA()` has been executed completely.
What is the purpose of subroutine `subA()`?

(4 Points)

`array1 =` **`"HALLO-WELT!!!"`**

P `subA()`: **Convert characters in a string from lower case to upper case**

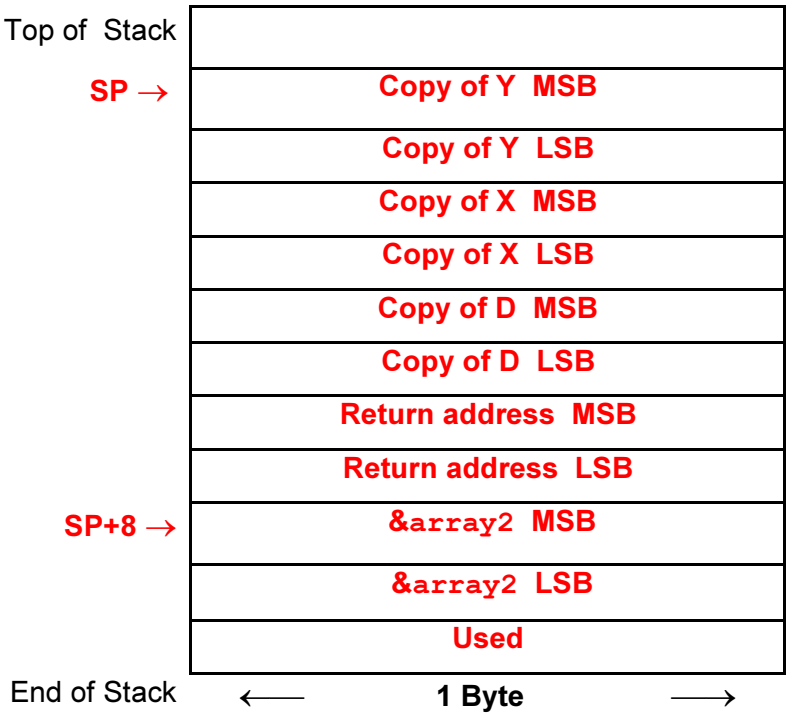
3.3
Into which HCS12-assembler instructions does the HCS12-C-compiler translate the following C-code: `i = subB(array2, 39);`

(4 Points)

```
LDD   #array2
PSHD
LDAB  #39
JSR   subB
STAB  i
LEAS  2, SP
```

3.4
Specify the state of the stack including SP, when the program reaches label `M0` in subroutine `subB()`:

(6 Points)



Sample Exam F		Page #:	**6 of 9**
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

3.5
What is the contents of register D and X,when the program reaches label `M1` in `subB()`?

(4 Points)

`D =` **9**

`X =` **3**

3.6
Specify the contents of `array2` and `i`, when subroutine `subB()` has been executed.
What is the purpose of subroutine `subB()`?

(6 Points)

`array2 =` `"39" (ASCIIZ String with 0-Byte as last byte)`

`i =` `39 (Number as unsigned 8bit integer)`

Zweck von `subB()`: **Convert 8 bit variable val into an ASCII-String representing its decimal value → Similar to decToASCII in lab 1)**

3.7
What is the purpose of instructions `PSH...`, und `PUL...` at the begin and at the end of subroutine `subB()`?

(2 Points)

`Registers D, X and Y are save and restored for the calling program, because they are used within the subroutine.`

Sample Exam F		Page #:	**7 of 9**
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: Program for Dragon12-Board (Σ 20 Points)

The following HCS12-program for the Dragon12-board shall be analyzed. The crystal clock frequency of our boards is f_{OSCCLK} = 4 MHz.

```
//Variable ////////////////////////////////////////////////////
#define DIVIDER 20
char msg[32] = "Aufgabe 4";
int  flag    = DIVIDER,
int  adc     = 0;

//Functions known from lab 1 //////////////////////
void initLCD(void);        //Initialize LCD display
void writeLine(void);      // ASCII-string output on LCD display
void decToASCII(void);     //Convert decimal number into ASCII string

//Wrapper functions for functions used in lab 1 //
void initLCDWrapper(void)
{   asm {  JSR initLCD
        };
}
void lcdWriteLineWrapper(char *text, char lineNr);
{     . . .
}
void decToAsciiWrapper(char *text, int val)
{     . . .
}
//Miscellaneous functions//////////////////////////////////////
interrupt 7 void rtiISR(void)
{
    flag--;
    ATD0CTL5 = 0x87;
    while ( ATD0STAT0 & 0x80 == 0 ) { };        //← ??? ← ??? ← ???
    adc = ATD0DR0;
    PWMDTY0 = adc >> 2;
    CRGFLG = CRGFLG | 0x80;
}
void rtiInit(void)
{   RTICTL = 0x6B;
    CRGINT = CRGINT | 0x80;
}
void adcInit(void)
{   . . .
}
```

Sample Exam F		Page #:	**8 of 9**
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

```
void pwmInit(void)
{   PWMCLK   = 0x00;
    PWMPRCLK = 0x77;
    PWMPOL   = 0x01;
    PWMPER0  = 0xFF;
    PWMDTY0  = PWMPER0/2;
    PWME     = 0x01;
}

void main(void)
{   EnableInterrupts;

    initLCDWraper();
    lcdWriteLineWrapper(msg, 0);

    pwmInit();
    adcInit();
    rtiInit();

    for (;;)
    {       if (flag <= 0)
            {     decToAsciiWrapper(msg, adc);
                  lcdWriteLineWrapper(msg, 1);
                  flag = DIVIDER;
            }
    }
}
```

The program periodically measures an analog signal to control the duty cycle of a PWM output. Additionally, the analog value is shown on the LCD display.

4.1

With which frequency is the RTI-Interrupt-Service-Routine (ISR) triggered?

(4 Points)

$2^{9+6}(11+1)/4MHz$ = 98,3 ms → 1/98,3ms = 10,2 Hz

4.2

Which PWM channel is used in the program? Does this channel use the slow or the fast PWM clock?

(4 Points)

Channel 0

Fast Taktsignal T_A

4.3

The C-program shall use the LCD driver function **`writeLine`**, which you analyzed in lab 1. In the same lab you developed function **`decToASCII`** to convert a 16 bit value into an ASCII string. Both functions were written in HCS12 assembler, passing all parameters via registers. Unfortunately, this is not compatible with the HCS12 C-compiler. Thus, so-called *wrapper func-*

tions in C are required, to adapt the parameter passing between the C-program and the original HCS12 assembler functions from lab 1.

Assembler function **`decToASCII`** uses the following registers for parameters

```
decToASCII:     Pointer to ASCII-string in X, 16 bit value in D
```

Write wrapper function **`decToAsciiWrapper`** (C with HCS12 inline assembler):

(4 Points)

```
void decToAsciiWrapper(char *text, int val)
{     asm
      {     LDX  text
            LDD  val
            JSR  decToASCII
      }

Note:
LDD val may be ommitted, when you rely on the fact, that the C compil-
er uses the same register.


}
```

4.4

- Write a C-function **`adcInit()`** o initialize the analog to digital converter ADC for channel 7. Start of the measurement and reading the result are already implemented in function **`rtiISR()`**.
- Does the ADC use polling or interrupt mode?
- What is the purpose of the line marked with „← ??? ← ???“ in **`rtiISR()`**?

(8 Points)

```
ADC operating mode:              Polling

Purpose of line „← ??? ← ???“:   Wait for end of ADC conversion



//Initialize ADC
void adcInit(void)
{
      ATD0CTL2 = 0xC0;
      ATD0CTL3 = 0x08;
      ATD0CTL4 = 0x05;



}
```