KSP - Tutorium

Test 2
Africa‘s Geek

KSP

Donchi Fofack Donald

Test für die Klausurvorbereitung – KSP

Nachname:	
Vorname:	
Betreuer:	Donchi Fofack Donald

	Max. Punktzahl	erreicht
Aufgabe 1	21	
Aufgabe 2	9	
Aufgabe 3	22	
Aufgabe 4	8	
Aufgabe 5	25	
Wissensfragen	15	
Gesamt	100	

Aufgabe 1: (21 Punkte)

Gegeben ist folgender Code:

```c
int main(int argc, char const *argv[])
{
    long z[] = {2, 19, 40, 12, 0};
    int a[5] = {5, 4, 3, 2, 1};
    int *b = a + 2;
    signed int c=0;

    char str[] = "Hallo Welt";
    char q = str[b[-1]];
    printf("z[z[z[0]] = %ld\n",z[z[z[4]]] );
    printf("*b = %d\n", *b);
    printf("*(str+3) = %c\n", *(str + 3));
    printf("*str+3 = %c\n", *str +3);
    printf("q = %c\n", q);
    printf("sizeof(str) = %ld\n", sizeof(str));
    printf("sizeof(z) = %ld\n", sizeof(z));
    return 0;
}
```

a) Ergänzen Sie die umgangssprachlichen Erklärungen zu obenstehenden Deklarationen. Zu Vorbelegungen brauchen Sie nichts schreiben! (benutzen Sie dabei deutsche Begriffe): (7 Punkte)

z ist ein Feld unbestimmter Größe von vorzeichenbehafteten Ganzzahlen

a ist ein Feld mit der Größe 5 von vorzeichenbehafteten Ganzzahlen.

b ist ein Zeiger auf eine vorzeichenbehaftete Ganzzahl

c ist eine vorzeichenbehaftete Ganzzahl

str ist ein Feld unbestimmter Größe von Zeichen.

q ist ein Zeichen.

main ist eine Function, die eine vorzeichenbehaftete Ganzzahl zurückgibt und eine vorzeichenbehaftete Ganzzahl und Feld unbestimmter Größe von Zeiger auf Zeichen erwartet.

b) welche Ausgabe erscheint bei der Ausführung des Programms?
NB: Der Architektur des Zielsystems X86_64 (14 punkte)

```
z[z[z[0]] = 40
*b = 3
*(str+3) = l
*str+3 = K
q = o
sizeof(str) = 11
sizeof(z) = 40
```

2 Punkte pro Zeile!

Aufgabe 2: (9 Punkte)

Wir haben zwei c-datei: main.c und helper.c:
Welcher Befehl benötigt man, damit ein ausführbares Programm (main) aus den beiden C-Quellcode erstellt wird?
Dabei soll alle Warnungen ausgegeben werden, das Debug-symbol soll im Programm vorhanden sein, das Programm soll nach c99 standart kompiliert werden, das Programm soll pedantisch an Standart halten.

gcc -g -pedantic -std=c99 -Wall -o main main.c helper.c

NB: wenn bei dem Befehl gcc, main.c und helper.c nicht da ist, dann 0 Punkt. Die 3 müssen unbedingt da sein.

Aufgabe 3: (22 Punkte)

I- Definieren Sie folgende Makros: (6 Punktes)

a) FUNCTION1: a/b
#define FUNCTION1(a,b) ((a)/(b))

b) FUNCTION2: (x*y)%5
#define FUNCTION2(x, y) (((x)*(y)) % 5)

c) FUNCTION3: x^5+x^4+5
#define FUNCTION3(x) (((x)*(x)*(x)*(x)*(x))+((x)*(x)*(x)*(x))+5)

NB: 2 Punkte pro richtige Antwort.

II- (16 Punktes)
a) Definieren Sie ein Makro: Set(x,n), welches das n-te Bit in der Zahl x setzt.
#define set(x,n) ((1<<(n))|(x))

b) Definieren Sie ein Makro, neg(x), welches alle Bits in Zahl x negiert.
#define neg(x) ~(x)

c) Definieren Sie ein Makro: del(x,n), welches das n-te Bit in der Zahl x löscht(delete).
#define del(x,n) (~(1<<(n)) & (x))

d) Definieren Sie ein Makro: tog(x,n), welches das n-te Bit in der Zahl x toggelt(toggeln).
#define tog(x,n) ((1<<(n)) ^ (x))

e) Angenommen, wir haben foldende Zahl in Hexadeximal: x = 0x0F2E,
x in Binäre ist: 0b 0000 1111 0010 1110

*negieren Sie bitte alle Bits in Zahl x
0b1111 0000 1101 0001 = 0xF0D1
*setzen Sie bitte das 5-te und 6-te Bit in der Zahl x.
0b0000 1111 0110 1110 = 0x0F6E
*Löschen Sie bitte das 4-te und 10-te Bit in der Zahl x.
0b0000 1011 0010 1110 = 0x0B2E
*Toggeln Sie bitte das 1-te und 2-te Bit in der Zahl x.
0b0000 1111 0010 1000 = 0x0F28

NB: 2 Punkte pro richtige Antworten!!

Aufgabe 4: (8 Punktes)

Gegeben Seien die 4 folgenden Header add.h, mul.h, sub.h, mod.h und die Datei main.c

add.h	mul.h
#ifndef ADD_H #define ADD_H void add(int a, int b); #endif	#ifndef MUL_H #define MUL_H void mul(int a, int b); #endif
main.c	**sub.h**

#include <stdio.h>
#include <stdlib.h>
#include “add.h”
#include “mul.h”
#include “sub.h”
#include “mod.h”
 	
int main(int argc, char *argv[]){
  add(6,9);
  mul(10,7);
  sub(20, 2);
  mod(10, 5);
 	
}
	#ifndef SUB_H
	#define SUB_H
	
	void sub(int a, int b);
	
	#endif

mod.h	**Frage:** Verhinden Sie bitte die Mehrfachinclusion!!
#define MOD_H #define MOD_H void mod(int a, int b); #endif	

NB: 0.5 Punkte pro richtige Antwort!

Aufgabe 5: (25 Punktes)

Wir haben folgende Berechnungen:
a) x=40*x + 2*y + z
b) y = x + y +z
c) z = x-y+6*z
d) x = 5*y – z
e) y = 6*x + 2*y + 19

Initiale Belegung der SDA:

sda[2]	z	5
sda[1]	y	1
sda[0]	x	-2

1- Erstellen Sie für jede Berechnung ein Programm in Ninja-Assembler. (10 Punktes)

2- zeichnen Sie noch bitte für jeden Assembler-code den Stack-Inhalt mit der größten Ausdehnung, dabei geben Sie den sp(stackpointer), pc(programmcounter) und welche Werte nun in der SDA stehen. (15 Punktes)

a) x=40*x + 2*y + z						
Ninja-Assember	Stack-Ausdehnung					
pushc 40 pushg 0 mul pushc 2 pushg 1 mul add pushg 2 add popg 0 halt		mul			Ende	
		1				
	-2	2	2			
	40	-80	-80	-160		

b) y = x + y +z						
Ninja-Assember	Stack					
pushg 0 pushg 1 add pushg 2 add popg 1 halt			Ende			

c) z = x-y+6*z						
Ninja-Assember	Stack					
Pushg 0 pushg 1 sub pushc 6 pushg 2 mul add popg 2 halt			Ende			

d) x = 5*y – z						
Ninja-Assember	Stack					
Pushc 5 pushg 1 mul pushg 2 sub popg 0 halt			Ende			

y = 6*x + 2*y + 19						
Ninja-Assember	Stack					
Pushc 6 pushg 0 mul pushc 2 pushg 1 mul add pushc 19 add popg 1 halt			Ende			