Florian von Zabiensky

FB MNI
TH Mittelhessen

Probeklausur – Konzepte systemnaher Programmierung – 08.07.2016

Nachname	
Vorname	
Matrikelnummer	
Klausurzulassung	WS / SS _ _
Dozent	

	max. Punktzahl	erreicht
Aufgabe 1	14	
Aufgabe 2	12	
Aufgabe 3	10	
Aufgabe 4	6	
Aufgabe 5	10	
Gesamt	52	

Aufgabe 1 ((1 + 1 + 1 + 1 + 1 + 1) Punkte)
Gegeben sind folgende Deklarationen:

```
int a[25];
unsigned int b;
char* c;
void (*func)(int x, int y);
char* argv[];
```

a) Ergänzen Sie die umgangssprachlichen Erklärungen zu folgenden Deklarationen (benutzen Sie dabei deutsche Begriffe):

a ist …

… ein Feld mit 25 vorzeichenbehafteten Ganzzahlen.

b ist …

… eine vorzeichenlose Ganzzahl.

c ist …

… ein Zeiger auf ein Zeichen.

func ist …

… ein Zeiger auf eine Funktion mit zwei Parametrn und keinem Rückgabewert. Die Parameter sind beide vorzeichenbehaftete Ganzzahlen.

argv ist ...

… ein Feld, unbestimmter Größe, welches Zeiger auf Zeichen beinhaltet.

Aufgabe 2 (4 + 4 + 2 + 2 Punkte)

a) Wie definieren Sie ein Makro für den Präprozessor und worauf müssen Sie dabei achten?

Bsp.:

```
#define KONSTANTE              25
#define MAKRONAME(param)       ((param) + 1)
```

Es gibt die Möglichkeit konstante Werte für die Textersetzung zu definieren (1. Beispiel). Dabei wird der symbolische Name hinter einem #define angegeben und darauf folgend der Wert dieser Konstanten.

Alternativ können Makros Inline Funktionen wiederspiegeln. Dabei wird zusätzlich zum Namen eine Parameterliste angegeben, die in dem darauf folgenden Ausdruck verwendet werden kann. Hierbei ist auf die Klammerung der Parameter, sowie die Klammerung des Gesamtausdrucks zu achten, da durch den Präprozessor eine reine Textersetzung stattfindet.

b) Basierend auf der Instruktionscodierung (8 Bit Opcode || 24 Bit Immediate), was bewirkt folgendes Makro? Erklären Sie und wählen Sie einen geeigneteren Namen!

```
#define        A(x)    ((x) & 0x00800000 ? (x) | 0xFF000000 : (x))
```

Die Instruktion x wird mit einer Bitmaske bitweise verunded. Dabei ist der Ausdruck ungleich 0, sobald das MSB des 24-Bit Immediate-Werts gesetzt ist. Dieses gesetzte Bit Lässt darauf schließen, dass es sich um einen negativen Wert handelt.
Um diese vorzeichenbehaftete Zahl in einem 32 Bit Wert darzustellen, muss das höchstwertige Byte des 32-Bit Werts ebenfalls mit einsen gefüllt sein. Dies wird für Den Fall ungleich 0 getan: (x) | 0xFF000000

Falls das MSB des 24-Bit Werts nicht gesetzt ist, muss das höchstwertigste Byte des entstehenden 32-Bit Werts auf 0 gesetzt werden. Demnach wird keine Änderung vorgenommen: (x)

Ein geeigneterer Name wäre z.B. VORZEICHEN_ERWEITERUNG

c) Was kommt in eine .c Datei und was kommt in eine .h Datei?

In eine .c Datei kommen Implementierungen von Funktionen, Deklarationen von globalen Variablen.

In eine .h Datei kommen Funktionsprototypen, Typdefinitionen, Makros und die Zusicherungen, dass bestimmte globale Variablen existieren (extern)

d) Wozu dient in C der tag bei einem Struct oder einer Union?

Er dient zur rekursiven Nutzung. Innerhalb der Struktur gibt es ohne ihn z.B. keine Möglichkeit einen Zeiger auf diese Struktur als Element der Struktur zu wählen.

Aufgabe 3 (5 + 5 Punkte):

a) Schreiben Sie eine Datenstruktur, um einen binärbaum in C darzustellen. Hierbei sollen Knoten ein Zeichen und zwei weitere Knoten enthalten. Blätter sollen vorzeichenbehaftete Ganzzahlen enthalten.

```
typedef struct node {
	unsigned char isInnerLeaf;
	union {
		struct {
			char c;
			struct node *left;
			struct node *right;
		} innerNode;
		int c;
	} u;
} Node_t;
```

b) Schreiben Sie eine Funktion in C, die zwei Knoten und ein Zeichen entgegennimmt, einen neuen Elternknoten für die beiden Knoten erstellt und einen Verweis auf diesen Elternknoten zurück gibt. Beachten Sie mögliche auftretende Fehler!

```
Node_t* newNode(char c, Node_t *left, Node_t *right) {
	Node_t* n = malloc(sizeof(Node_t));
	if(n == NULL) {
		error("No memory");
	}
	n->isInnerLead = 1;
	n->u.innerNode.c = c;
	n->u.innerNode.left = left;
	n->u.innerNode.right = right;
	return n;
}
```

Aufgabe 4 (5 + 2 + 2 Punkte)

a) Beschreiben Sie die drei Phasen des Stop & Copy Garbage Collection Verfahrens.

1. Flip:
   Ziel- und Quellspeicher werden getauscht und der Freizeiger auf 0 gesetzt.
2. Kopieren der Root-Objekte:
   Alle bekannten Objekte (in Stack, sda, bip, rv) werden vom Quellspeicher in den Neuen Zielspeicher kopiert. Dabei wird im Quellspeicher in dem Objekt das Broken Heart Flag gesetzt und ein Forwardpointer zum neuen Objekt hinterlegt. Die aktuell behandelte Referenz wird dabei auf das neue Objekt gelegt.
   Ist ein Objekt bereits kopiert (Broken Heart Flag), wird die Referent mittels Des Forwardpointers aktualisiert.
3. Scan-Phase:
   Der Zielspeicher wird vom Start bis zum Freizeiger durchgegangen. Dabei wird jedes Objekt einzelnd Betrachtet. Ist es kein primitives Rechenobjekt, werden alle Referenzen innerhalb des Objekts aktualisiert. Falls eine Referenz in den Quellspeicher zeigt, wird dieses Objekt in den Zielspeicher kopiert (inkl. Setzen von broken heat flag und forwardpointer) und die Referenz wird angepasst.

b) Welche Aufgabe hat der Stack Pointer?

Der Stack Pointer dient der Organisation des Stacks. Er zeigt immer auf das nächste freie Element im Stack. Dieses befindet sich direkt oberhalb des zuletzt hinzugefügten und noch nicht entfernten Elements auf dem Stack.

c) Aus welchem Grund gibt es eine Static Data Area in der Ninja VM?

Um Variablen zu hinterlegen, deren Lebensdauer länger als ein Funktionsaufruf ist. Diese dürfen demnach nicht im Stackframe sein.

Aufgabe 5 (10 Punkte)

Schreiben Sie in Ninja Assembler ein Programm, welches die Fakultät von 10 mit einer Schleife berechnet.

```
  pushc 10
  popg 0
  pushc 1
  popg 1
  jmp bedingung

schleife:
  pushg 0
  pushg 1
  mul
  popg 1
  pushg 0
  pushc 1
  sub
  popg 0

bedingung:
  pushg 0
  pushc 0
  ne
  brt schleife

end:
  pushg 1
  wrint
  halt
```