Compiler 1: Übung 3

Gruppe:

1. Michael Scholz (matr.: 1576630, rbg: mi48azih)
2. Ulf Gebhardt (matr.: 1574373, rbg: hu56nifa)

Aufgabe 3.1:

a)
Stack: Lokale Variablen existieren nur während der Laufzeit des aufgerufenen Programms. Hierfür wird bei jedem Prozeduraufruf ein Stack Frame angelegt, der nach dem Prozedurende wieder abgebaut wird. Somit ist die Lebenszeit der Variablen an den Geltungsbereich gebunden. Die globalen Variablen existieren über die gesamte Programmlaufzeit.
Heap: Beim Heap ist die Lebenszeit der Variablen unabhängig von den Geltungsbereichen. Somit muss jedoch eine explizite Verwaltung der Variablen erfolgen. Dies ist beispielsweise in C der Fall. Java nutzt mit dem Carbage Collector eine automatisierte Lösung. Unabhängige Lebenszeiten der Variablen sind beispielsweise bei Datenstrukturen wie Listen und Bäumen nötig.

b)
In jedem Stack Frame liegen vor den lokalen Variablen der static link, der dynamic link und die return address. Der static link verweist auf den Frame der im Programmtext umschliessenden Prozedur und dient somit dem Zugriff auf nicht-lokale Variablen. Der dynamic link ist ein Verweis auf den Frame der aufrufenden Prozedur und somit ein Verweis auf den vorherigen Stack Frame. Genauer auf den alten Wert von Local Base (LB), welcher nach dem Prozedurende wiederhergestellt werden muss.

c)
Das Routinenprotokoll definiert die Calling Convention. Diese legt fest wie Argumente an Funktionen übergeben werden und wie der Aufrufer die Ergebnisse erhält. Zudem verwaltet es die statische Verkettung. Das Routinenprotokoll ist ein wichtiger Bestandteil der Runtime-Spezifikation, da durch die Calling Convention jede Prozedur weiss, wie sie eine andere aufrufen kann.

Aufgabe 3.2:

a)

```
LOADL 2
LOAD a
MUL
LOAD b
MUL
LOADL 2
LOAD a
MUL
LOAD c
MUL
ADD
LOADL 2
LOAD b
MUL
LOAD c
MUL
ADD
STORE X
```

a) optimiert

```
LOAD a
LOAD b
MUL
LOAD a
LOAD c
MUL
ADD
LOAD b
LOAD C
MUL
ADD
LOADL 2
MUL
STORE X
```

b) → nichts zu optimieren!

```
LOAD a
LOADL 2
MUL
LOADL 4
SUB
LOAD a
MUL
LOADL 5
SUB
LOAD a
MUL
LOADL 7
ADD
LOAD a
MUL
LOADL 11
ADD
STORE Y
```

Aufgabe 3.3:

Die Speicherorganisation der Variable state ist in folgender Tabelle verdeutlicht:

Adresse	**Datentyp**				
0	Boolean	empty	state.board[0]	state.board	state
1	Char	occupant.symbol			
2	Integer	owner.number			
3	Boolean	empty	state.board[1]		
4	Char	occupant.symbol			
5	Integer	owner.number			
… … …			state.board[2] – state.board[62]		
189	Boolean	empty	state.board[63]		
190	Char	occupant.symbol			
191	Integer	owner.number			
192	Integer	next.number		state.next	
193	Integer			state.moves	
...					

Aufgabe 3.4:

BC(Before Call) steht für VOR dem Aufruf
BR(Before Return) steht für VOR Austritt aus dem Aufruf
AC(After Return) steht für NACH dem Aufruf

SL = Static Link
DL = Dynamic Link
Return = Return address

a) power(2,1)

BC power(2,1)	BC even(1)	BR even(1)	AC even(1)	BC power(2,0)
2	2	2	2	2
1	1	1	1	1
	SL	SL	SL	SL
	DL	DL	DL	DL
	Return	Return	Return	Return
	1	1	false	false
		SL		2
		DL		0
		Return		
		false		
----------	----------	----------	----------	----------
BR power(2,0)	AC power(2,0)	BC sqr(1)	BR sqr(1)	AC sqr(1)
2	2	2	2	2
1	1	1	1	1
SL	SL	SL	SL	SL
DL	DL	DL	DL	DL
Return	Return	Return	Return	Return
false	false	false	false	false
2	1	1	1	1
0			SL	
SL			DL	
DL			Return	
Return			1	
1				
----------	----------	----------	----------	----------
BR power(2,1)	AC power(2,1)			
2	2			
1				
SL				
DL				
Return				
false				

2				

b) power(3,3)

BC power(3,3)	BC even(3)	BR even(3)	AC even(3)	BC power(3,1)
3	3	3	3	3
3	3	3	3	3
	SL	SL	SL	SL
	DL	DL	DL	DL
	Return	Return	Return	Return
	3	3	false	false
		SL		3
		DL		1
		Return		
		false		
----------	----------	----------	----------	----------
BC even(1)	BR even(1)	AC even(1)	BC power(3,0)	BR power(3,0)
3	3	3	3	3
3	3	3	3	3
SL	SL	SL	SL	SL
DL	DL	DL	DL	DL
Return	Return	Return	Return	Return
false	false	false	false	false
3	3	3	3	3
1	1	1	1	1
SL	SL	SL	SL	SL
DL	DL	DL	DL	DL
Return	Return	Return	Return	Return
1	1	false	false	false
	SL		3	3
	DL		0	0
	Return			SL
	false			DL
				Return
				1

BC sqr(1)	BR sqr(1)	AC sqr(1)	BR power(3,1)	BC sqr(3)
3	3	3	3	3
3	3	3	3	3
SL	SL	SL	SL	SL
DL	DL	DL	DL	DL
Return	Return	Return	Return	Return
false	false	false	false	false
3	3	3	3	3
1	1	1	1	
SL	SL	SL	SL	
DL	DL	DL	DL	
Return	Return	Return	Return	
false	false	false	false	
1	1	1	3	
	SL			
	DL			
	Return			
	1			
----------	----------	----------	----------	----------
BR sqr(3)	AC sqr(3)	BR power(3,3)	AC power(3,3)	
3	3	3	27	
3	3	3		
SL	SL	SL		
DL	DL	DL		
Return	Return	Return		
false	false	false		
3	9	27		
SL				
DL				
Return				
9				