hochschule mannheim

– Fakultät für Informatik –

HWP

– Bachelor – Informatik

PFLICHTÜBUNG

Testat bis zum: Bis zum Januar 2015

Jede Gruppe muss selbstständig und selbst kommentiert die folgenden Aufgaben in einer Sprache ihrer Wahl lösen. Ist ersichtlich, dass voneinander abgeschrieben wurde, so gilt für ALLE beteiligten Personen die Studienleistung als NICHT erbracht!

AUFGABE 1

Erstellen Sie ein Programm in einer ihrer favorisierten Sprachen (Java, C/C++, JavaScript o.ä.), welches eine Register basierte VM (mit 16 Registern, 16 Bit Befehle, 4096 Memory Adressen, Von Neumann Architektur), die folgenden Befehle implementiert:

Nr.	Befehl	Befehlsbits	Parameter	Bedeutung
1	NOP	0000	keine	keine Operation
2	LOAD	0001	#wert	R0=#wert
3	MOV	0010	Rx,Ry und ToMem/FromMem	Rx=Ry,(Rx)= Ry,Rx=(Ry)
4	ADD	0011	Rx,Ry	Rx=Rx+Ry
5	SUB	0100		Rx=Rx-Ry
6	MUL	0101		Rx=Rx*Ry
7	DIV	0110		Rx=Rx/Ry
8	PUSH	0111		Rx auf Stack
9	POP	1000		Rx von Stack
10	JMP	1001	#wert = adress	Jump
11	JIZ	1010	#wert = adress	JumpIf R0==0
12	JIH	1011	#wert = adress	JumpIf R0>0
13	JSR	1100	#wert = adress	Jump to Subroutine
14	RTS	1101		Return from Subroutine

Der Opcode ist folgendermassen aufgebaut:

/		To-Mem	From-Mem	Index von Ry				Index von Rx				Befehls Bits			
#wert												Befehls Bits			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Es gibt:

- 1 programCounter (zeigt auf Adresse der aktuellen Programmausführung)
- 16 Register (ganzzahlig, z.B. als Integer implementiert)
- Stack für Sprünge in Unterprogramme. Ein JSR Befehl legt die Rücksprungadresse auf den Stack, ein RTS Befehl holt diesen vom Stack und springt an die Adresse zurück. Wird RTS bei leerem Stack aufgerufen, so wird das Programm beendet.
- Stack für die Register, genutzt für PUSH und POP Befehle.

AUFGABE 2

Erweitern Sie nun die VM durch einen einfachen Assembler, der ein Textfile mit dem Programm einliest, in den Opcode umwandelt, in den Speicher schreibt und in der Virtuelle Maschine ausführt.

Schreiben Sie ein Beispielprogramm, welches die ersten 100 Fibonacci Folge ab Speicheradresse 1000 in den Speicher schreibt. Implementieren Sie den Algorithmus rekursiv (mit JSR Unterroutine)

AUFGABE 3

Schreiben Sie einen Profiler, der Ihnen nach einer vordefinierten Anzahl von Zyklen angibt, wieviel Rechenzeit jede Programmzeile benötigt hat. Dazu definieren Sie ein Profile-Array, welches bei jedem Programmschritt am Index=ProgramCounter um 1 hochgezählt wird. Speichern Sie schließlich den Programmquelltext in eine Profile.txt Datei und schreiben Sie vor jede Programmzeile die Rechenzeit in Prozent die in der Zeile Verbraucht worden ist.