Entwicklung eines Laborversuches zum Thema Mikroarchitekturebene anhand des Mic-1 MMV Simulators

Niklas Lampe – 5017616
Thade Struckhoff – 5016730

22. Juli 2021

Inhaltsverzeichnis

1 Thema

Der Laborversuch mit dem Titel: Verstehen und Anwenden des IJVM-Befehlssatzes auf einer MAL Mikroarchitektur anhand ausgewählter Algorithmen mithilfe des Mic-1 MMV Simulators soll praxisorientiert dazu beitragen, ein grundlegendes Verständnis für die Mikroarchitekturebene und IJVM-Entwicklung zu erhalten. Dazu sollen sich die Studierenden zunächst mit der Umgebung des 'Mic-1 MMV' Simulators vertraut machen und anschließend mithilfe dessen eine Fehleranalyse durchführen. Dabei sollen Fehlerzustände in einem gegebenen IJVM-Programmbeispiel gefunden und behoben werden. Nachdem so der Befehlssatz kennengelernt wurden, sollen die neu erworbenen Kenntnissen angewendet werden, um ein eigenes IJVM-Programm anhand einer vorgegebenen Problemstellung umzusetzen.

2 Zielsetzung und Lernziele

Die Studierenden sollen in Zweier oder Dreier-Gruppen (vorzugsweise Zweier-Gruppen) ein tiefergehendes Verständnis über das Thema 'Mikroarchitekturebene' und IJVM-Entwicklung erlangen. Dazu soll ein durch den IJVM-Befehlssatz abgebildeter Algorithmus analysiert und auf vorhandene Fehlerzustände hin überprüft werden. Durch diverse Prüfungsverfahren können außerdem in einem IJVM-Programm Fehlerzustände gefunden und behoben werden.

Zudem soll ein in Java geschriebener Algorithmus erkannt und durch den IJVM-Befehlssatz der Beispiel-ISA implementiert werden.

Als Hilfe wird der 'Mic-1 MMV' Simulator eingesetzt, der neben dem MAL Mikroassembler auch die Beispiel-ISA IJVM implementiert. Diese umfasst die ganzzahligen Befehle der Java Virtual Machine. Der Simulator stellt eine grafische Oberfläche mit visueller Darstellung des Datenpfads bereit, unterstützt das Debugging eines IJVM-Programms u.a. mithilfe von Breakpoints und bietet darüber hinaus noch weitere Unterstützungsmöglichkeiten.

Nach der erfolgreichen Bearbeitung des Versuchs haben die Studierenden ein grundlegendes Verständnis über die MAL Mikroarchitektur und den IJVM-Befehlssatz der Beispiel-ISA erlangt. Sie sind in der Lage, Fehler in einem IJVM-Programm mithilfe eines Debuggers zu erkennen und zu beheben. Zudem können die Studierenden problembezogen ein IJVM-Programm implementieren.

3 Konzeption des Versuchs

Dieser Laborversuch soll als praktische Übung während des laufenden Semesters durchgeführt werden. Er soll den Studierenden eine Möglichkeit bieten, ihre Fähigkeiten in der ISA-Programmierung zu festigen und das Thema 'Mikroarchitektur' zu vertiefen. Dies sind notwendige Kenntnisse, die zum Einen bei der finalen Laborprüfung und zum Anderen bei der Klausur benötigt werden. Dieser Versuch untergliedert sich in drei aufeinander aufbauende Teile, die nachfolgend erläutert werden.

In der erste Aufgabe wird der Simulator 'Mic-1 MMV' eingeführt. Die Einführung orientiert sich in einer etwas abgewandelten Form an dem Tutorial in der Dokumentation des Simulators. Dabei werden die wichtigsten Funktionen beschrieben und Studierende in deren Ausführung angeleitet. Dabei erlernen sie, wie Mikroprogramme geladen sowie IJVM-Programme assembliert, geladen und ausgeführt werden können. Außerdem werden Möglichkeiten zur strukturierten Programmausführung aufgezeigt, wodurch bspw. das Speicherverhalten in Programmabläufen oder die Programmabläufe selbst nachvollzogen werden können. Diese Aufgabe wird zu Beginn des Laborversuchs angesetzt, da die weitere Bearbeitung ein weitreichendes Verständnis über die zielgerichtete Nutzung des 'Mic-1 MMV' Simulators erfordert.

Die zweite Aufgabe befasst sich mit einer Fehleranalyse und vertieft die Debugging-Unterstützung des Simulators. Die Studierenden müssen dabei insgesamt fünf Fehlerzustände im Mikroprogramm sowie im IJVM-Programm identifizieren und beheben. Die Fehleranalyse des Mikroprogramms soll den Studierenden einen Anreiz geben, sich mit der Mikroarchitektur auseinander zusetzen sowie die Befehlsstrukturen und -sequenzen zu verstehen. Dadurch erlangen sie ein grundlegendes Verständnis über die MAL Mikroarchitektur. Im Anschluss sollen die Studierenden eine Fehleranalyse für ein bereitgestelltes IJVM-Programm durchführen. Dies soll den Ansporn geben sich mit dem IJVM-Befehlssatz näher auseinander zusetzen, um Aufbau und mögliche Parameter zu verinnerlichen. Zusätzlich werden dabei Fähigkeiten erlangt, um qualitätssichernde Maßnahmen zusammen mit dem Simulator auf eigenständig implementierte IJVM-Programm anzuwenden. Insgesamt bereitet diese Aufgabe die Grundlage für die nachfolgende Aufgabe, da für ein eigenständig implementiertes IJVM-Programm einerseits das interpretierende Mikroprogramm verstanden und andererseits der zur Verfügung stehende Befehlssatz bekannt sein muss. Außerdem müssen auf etwaige Fehlerwirkungen entsprechend reagiert werden, um Fehlerzustände lokalisieren und beheben zu können.

Durch die dritte und letzte Aufgabe des Laborversuchs sollen Studierende zeigen, dass Befehlssatz und Interpreter verstanden sind und zielgerichtet zur Problemlösung eingesetzt

werden können. Das zu lösende Problem ist dabei ein allgemein bekannter Algorithmus aus der Reihe der Sortieralgorithmen. Er soll benannt und als ein IJVM-Programm implementiert werden. Der Algorithmus wird in Form einer Java-Methode vorgegeben. Die Bearbeitung dieser Aufgabe erfüllt abschließend das letzte Lernziel, nämlich die problembezogene Umsetzung eines IJVM-Programms, und vertieft nochmals den IJVM-Befehlssatz.

4 Beschreibung der Teilaufgaben

Aufgabe 1: Einführung

In dieser Aufgabe soll der Aufbau von IJVM-Mikroprogrammen anhand ausgewählter Beispiele verstanden werden. Zudem soll den Studierenden die Verwendung des Simulators anhand von Beispielen näher dargelegt werden, um so das Ausführen eines IJVM- Mikroprogramms kennenlernen. Hierbei werden die verschiedenen Ausführungsmöglichkeiten des Simulators gezeigt, wodurch die Studierenden einen guten Überblick über die Funktionsweise des Simulators erhalten. Dadurch kann in den weiteren Aufgaben vertiefend auf diese Funktionen zurückgegriffen werden, um Probleme lösen zu können. Im Anschluss sollen die Studierenden anhand einer vorgegebenen Rechenoperation die Entwicklung eines Stacks durchführen, um schließlich mit einem Beispielprogramm die Entwicklung im Simulator nachzuvollziehen. Dadurch lernen die Studierenden die visuelle Darstellung des Kellerbereichs im Simulator kennen. Bei der Rechenoperation sind Zwischenschritte eingebaut, um die Auswirkungen der Speicher- und Laderoutinen bei der Stackentwicklung mit zu berücksichtigen.

Aufgabe 2: Fehleranalyse

An dieser Stelle des Versuchs sollen sich die Studierenden näher mit der Nutzung des Simulators befassen. Sie sollen mit den entsprechenden Möglichkeiten zur Fehleranalyse experimentieren und IJVM-Programmabläufe nachvollziehen können. Damit werden später benötigte Debugging-Fähigkeiten erlangt und das Wissen über die ISA-Ebene und IJVM-Programmierung durch einen praktischen Bezug vertieft werden. Zu beginn sollen Sie den vorgegebenen Programmcode sichten. Dadurch erhalten Sie einen ersten Eindruck über die Implementierung der mathematischen Problemstellung. Dabei wird geprüft, ob die Studierenden IJVM-Programme lesen und verstehen können.

Durch die anschließende Fehleranalyse üben die Studierenden erneut das Codeverständnis und algorithmisches Denken. Außerdem erkennen Sie die Vorteile und Funktionsweisen der verschiedenen Sichten des Simulators und das Setzen gezielter Breakpoints wird geübt. Zusätzlich müssen sie zeigen, dass sie IJVM-Befehle nicht nur lesen, sondern sie auch

selbstständig anwenden und Befehlssequenzen anpassen bzw. erweitern können, um ein Problem zu lösen.

Aufgabe 3: Implementierung

Diese Aufgabe stellt den Kern dieses Laborversuchs dar. Die Studierenden besitzen nun alle notwendigen Voraussetzungen, um selbstständig ein IJVM-Programm zu implementieren, was sie in dieser Aufgabe auch unter Beweis stellen müssen. Es wird eine Java Implementierung des *Selection Sort*-Algorithmus zur Verfügung gestellt, der zunächst gesichtet und erkannt werden muss. Dadurch wird zusätzlich das algorithmische Denken trainiert. Danach sollen die Studierenden eben jenen Algorithmus in ein IJVM-Programm überführen, wobei ein Grundgerüst zur Verfügung gestellt wird, sodass die Implementierung der Sortierung an sich fokussiert wird.

Nach der Bearbeitung dieser Aufgabe sollen die Studierenden sowohl die Mikroarchitekturebene verstanden, den Mic-1 IJVM-Befehlssatz kennengelernt und die Verwendung des 'Mic-1 MMV' Simulators verstanden haben. Sie haben gezeigt, dass sie eine IJVM-Implementierung umsetzen können, um dadurch ein bestehendes Problem zu lösen. Dadurch sind schlussendlich alle Lernziele des Versuchs erfüllt.

5 Hilfsmittel

Für diesen Versuch dient der Mic-1 MMV-Simulator als wichtigstes Hilfsmittel. Die Studierenden finden die Funktionalität dieses Simulators innerhalb des Laborversuches selbstständig heraus. Die Laboraufgabe ist dementsprechend formuliert, so dass die Studierenden dieser folgen können. Hinweis: Ein bekannter Fehler des Simulators ist, dass bei einer Änderung der Ausführungsgeschwindigkeit im laufenden Programm das eingelesene Programm abstürzen kann. Es kann sein, dass daher das Programm neu eingelesen werden muss. Sollten die Studierenden nicht selbstständig auf die Lösung des Neueinlesens kommen, kann darauf hingewiesen werden. Der IJVM-Befehlssatz ist auf dem Labor-Aufgabenzettel zur Verfügung gestellt und muss daher nicht zusätzlich bereitgestellt werden.

Für den Laborversuch sind drei weitere Programmdateien bereit zustellen. Dazu gehören die Dateien 'rechenOperation.jas', 'ggT.jas' und 'sortieralgorithmus.jas'. Die drei Dateien können dafür in den examples-Ordner des Simulators eingefügt werden, in der sich auch die 'echo.ijvm'-Datei des Simulators befindet. Mit den drei Dateien sowie der 'echo.ijvm'-Datei des Simulators können die drei Laboraufgaben erfolgreich durchgeführt werden.

6 Erwartetes Vorgehen

Vor Beginn des Laborversuchs wird von den Studierenden erwartet, dass das Kapitel 3 'Mikroarchitekturebene' im Skript der Vorlesung vollständig erarbeitet ist. Der Versuch wird dieses Wissen festigen und weiter vertiefen. Weiterführende Vorbereitungen sind nicht notwendig. Zu Beginn des Versuchs wir von den Studierenden erwartet, sich mit den Funktionen des Simulators auseinander zu setzen. Eine Beschreibung, wie Programme assembliert und geladen werden können, ist auf dem Aufgabenblatt enthalten. Hieran können sich die Studierenden orientieren. Es wird erwartet, dass die verschiedenen Ausführungsgeschwindigkeiten kennen gelernt und unterschieden werden können. Über den visuellen Datenpfad und den verschiedenen Speicheransichten sollen Programmabläufe nachvollzogen werden. Durch die Ein- und Ausgabekonsole soll mit IJVM-Programmen interagiert werden. Für die Entwicklung des Stacks in der Teilaufgabe A1.2 soll eine einfache Formatierung in das Textdokument des Protokolls aufgenommen werden. Es wird kein einzelnes Blatt benötigt. Es wird erwartet, dass die Studierenden den zur Verfügung stehenden IJVM-Befehlssatz für sämtliche Aufgaben heranziehen, um so die vorgegebenen Programme zu verstehen und eigene entwicklen zu können. Kommen Fragen an den Dozenten zu dem Simulator, so dürfen diese beantwortet werden. Bei inhaltliche Fragen sollte auf das Skript verwiesen werden. Für die Ausführung ist den Studierenden freigestellt, ob diese den Laborrechner oder einen eigenen Rechner verwenden. Als Voraussetzung ist lediglich eine Java Version ab der 1.4 notwendig, ansonsten stehen keine weiteren Voraussetzungen an.

Treten Fragen bei der Fehleranalyse auf, kann der Dozent Hinweise geben, sofern die Studierenden festhängen und die Zeit vorangeschritten ist. Als Beispielhinweis ist aufzuführen, dass das große Ganze der Methoden 'ggT' und 'mod' analysiert werden sollte. Zur Erklärung: sobald der Funktionsgraph klar geworden ist, fällt die Dauerschleife innerhalb dieser Methoden auf. Danach sollten die nicht auf den Stack gelegten Eingangsparameter sowie das nicht getätigte Ablegen des berechneten ggTs in der Methode 'ggT' ersichtlich sein. Für die dritte Aufgabe ist keine Hilfestellung vorgesehen. Hier sollen die Studierenden ihr aufgebautes Wissen unter Beweis stellen.

Eine Nachbereitung für den Versuch ist nicht vorgesehen. Sollte der Versuch jedoch nicht beendet werden können, so muss dies in Eigenarbeit nachgeholt und zum nächsten Labortermin nachgereicht werden.

Die Studierenden sollen die Aufgaben nacheinander durcharbeiten. So können sie Erlerntes aus vorherigen Aufgaben jeweils in den darauffolgenden Aufgaben in Verbindung setzen und darauf aufbauen. Durch die Relation der Teilaufgaben sollen die Studierenden ein

umfangreiches Wissen über die Mikroarchitekturebene und die IJVM-Programmierung erlangen.

7 Aufwandsschätzung

Da die Vorbereitung für diesen Versuch auf der Vorlesung Mikroarchitekturebene beruht und sich die Studierenden in diesem Zuge mit dem IJVM-Befehlssatz auseinander gesetzt haben sollten, ist kein zusätzlicher Vorbereitungsaufwand vorgesehen. Eine Nachbereitung ist im Normalfall nicht notwendig. Sollten die Studierenden jedoch in der regulären Laborzeit nicht fertig geworden sein, da bspw. Themen zu oberflächlich verstanden worden sind, müssen diese entsprechend nachbereitet werden, um die fehlenden Lücken aufzuarbeiten. Dieser sollte jedoch nicht allzu groß sein. Der Aufwand, der für die Präsenzzeit vorgesehen ist, wird im Folgenden für die einzelnen Aufgaben abgeschätzt.

Für die erste Aufgabe werden 40 Minuten eingeplant. Die Studierenden sollen genug Zeit bekommen, um sich mit dem Mic-1 Simulator und dessen Funktionalitäten auseinander zu setzen. Anschließend soll auch der Aufbau von IJVM-Programmen verstanden und eine Stackentwicklung eines Programms erarbeitet und im Simulator nachverfolgt werden. Für die händische Entwicklung des Stacks sowie die Überprüfung mit dem Simulator sollte die vorgegebene Zeit ausreichen.

Die Fehleranalyse als zweite Aufgabe wird mit 45 Minuten kalkuliert. Dabei vertiefen die Studierenden Ihre Fähigkeiten in der und das Verständnis über die IJVM-Programmierung. Bei der Zeitplanung mit inbegriffen ist eine initiale Sichtung des gegebenen Programms und zwei durch den Assemblierer angezeigte Fehler können schnell behoben werden. Durch durch das Auffinden der verbleibenden drei Fehler bewirkt eine Umfassende Programmanalyse und die Verwendung des Simulators ausreichend Grundwissen, um nachfolgend eigenständig ein IJVM-Programm zu implementieren.

Die Aufgabe drei, als Kern des Labors, benötigt mit seinen vorgesehenen 95 Minuten in der Präsenzzeit anteilsmäßig den größten Aufwand. Dabei sollen die Studierenden zeigen, dass Sie die Mikroarchitekturebene verstanden und eigenständig IJVM-Programme implementieren können. Dabei wenden Sie auch die Möglichkeiten des Mic-1 Simulators an, um Ihrer Implementierungen einer Qualitätssicherung zu unterziehen.

Da es sich hierbei um einen Versuch des laufenden Semesters handeln soll, kann nach diesen 180 Minuten ein Wechsel der Laborgruppen stattfinden. Sollte der Zeitplan des Semesters die Möglichkeit bereitstellen, können die Gruppen auch an verschiedenen Tagen ihren Versuch durchführen. Um sicherzustellen, dass das Labor zeitlich nicht überschritten wird, kann der Dozent nach der Hälfte der Zeit darauf hinweisen, dass die Studierenden

spätestens dann mit der dritten Aufgabe anfangen haben sollten. Idealerweise sind Sie schon dabei und wissen womöglich bereits, um welchen Sortieralgorithmus es sich handelt.

8 Erwartete Ergebnisdokumentation

Die Studierenden sollen alle erarbeiteten Ergebnisse schriftlich protokollieren. Dabei ist ein Schreiben in digitaler Form (z.B. in einem Editor) wünschenswert, welches im Anschluss über den Labordrucker ausgedruckt und abgegeben werden kann. In den einzelnen Aufgabenteilen ist jeweils beschrieben, was zu dokumentieren oder auch abzugeben ist. Die bearbeiteten Programmdateien sind von den Studierenden als Zip-Ordner zu verpacken und in eine entsprechend zu erstellenden Übungen in Aulis hochzuladen. Aus dem Protokoll bzw. der digitalen Abgabe sollte die Vorgehensweise, also die Reihenfolge, der bearbeiteten Aufgaben sowie die entsprechenden Ergebnisse ersichtlich sein.

Falls bei der Bewertung gravierende Unstimmigkeiten auftreten, sollten die betreffenden Studierenden damit konfrontiert werden. In einem gemeinsamen Dialog soll der Grund dessen erörtert werden. Dazu sollen die Studierenden zunächst eine Vermutung anstellen, die den Unterschied begründet. Dabei ist zu bewerten, ob die Abweichung an der falschen Durchführung seitens der Studierenden oder an einer falschen Erwartungshaltung liegt. Dabei unterstützt den Dozenten die Musterlösung (Kap. 11). Dabei ist drauf zu achten, dass den Studierenden lediglich Anregungen gegeben werden und noch nicht die vollständige Lösung. Dadurch machen Sie sich selbst noch einmal Gedanken, wodurch sichergestellt wird, dass sie aufgetretene Problempunkte verstehen und dadurch dennoch das Lernziel des Labors erreichen.

Über den Umstand sollten die Studierenden vorweg informiert werden. Außerdem sollte daran erinnert werden, dass ein zu erstellendes Protokoll nach dem Labor an den Dozenten abzugeben ist und veränderte Programmdateien in Aulis hochzuladen sind. Diese Ergebnisse sollten mit der Musterlösung aus Kapitel 11 verglichen und nach der in Kapitel 9 beschriebenen Bewertung an die Studierenden zurückgegeben werden.

9 Bewertungsmodus und -richtlinien

Da der Versuch im laufenden Semester durchgeführt wird, findet keine direkt Benotung statt. Der Versuch ist lediglich ein Teil der Gesamt-Labornote und dient als Vorbereitung für die abschließende Laborprüfung. Diese wird nicht bestanden, sollte aus der Bewertung ersichtlich werden, dass das Thema Mikroarchitektur nicht verstanden wurde. Als Richtlinie dafür gilt die erreichte Punktzahl 50 von 100 Punkten, um den Versuch als bestanden zu werten. Die Punkteverteilung ist im Folgenden aufgeschlüsselt. Eine genaue Ausarbei-

tung der Anforderungen ist in der Musterlösung im Kapitel 11 zu finden. Allgemein gilt: Wird eine Teilaufgabe von dem Studierenden nicht bearbeitet, so wird dieser Abschnitt mit 0 Punkten bewertet.

9.1 Aufgabe 1 - 20 Punkte

In der ersten Teilaufgabe werden lediglich die Funktionalitäten des Simulators getestet. Hierbei werden keine Punkte vergeben. Die zweite Aufgabe befasst sich mit der händischen Entwicklung des Stacks. Dazu werden für jeden richtigen Schritt 0.25 Punkte vergeben, bei 16 Schritten ergibt dies in Summe 4 Punkte.

Die nächste Teilaufgabe beinhaltet die stichpunktartige Beschreibung des Programms 'rechenOperation.jas'. Dafür werden insgesamt 10 Punkte vergeben, die wie folgt zusammengesetzt werden:

- P1 - P4 geben jeweils 0.5 Punkte.
- P5 gibt aufgrund einer größeren Funktionalität mit Methodenaufruf 2 Punkte
- Die Punkte P6 bis P11 geben jeweils 1 Punkt

Die letzte Teilaufgabe befasst sich mit dem Vergleich der Stackentwicklung im Simulator. Für das Wiederfinden der Rechenoperation im Stackverlauf werden 2 Punkte vergeben. Für das Erkennen der Initialisierung gibt es ebenfalls 2 Punkte. Auch für das Erkennen der Ausgabe werden 2 Punkte vergeben, womit insgesamt 6 Punkte bei dieser Teilaufgabe zu erlangen sind.

9.2 Aufgabe 2 - 30 Punkte

Bei der zweiten Aufgabe müssen fünf Fehlerzustände durch eine Analyse gefunden und behoben werden. Dies wird in der Summer mit 30 Punkten dotiert, die sich wie im Folgenden beschreiben aufteilen:

- Erkennen und Beheben der ersten beiden Fehler (jeweils 3 Punkte)
- Erkennen und Beheben der Endlosschleife (10 Punkte)
- Erkennen und Beheben der Fehler 4 und 5 (jeweils 7 Punkte)

9.3 Aufgabe 3 - 50 Punkte

Die erste Teilaufgabe erfordert das Sichten eines vorgegebenen Sortieralgorithmus sowie seiner Benennung. Dafür sind insgesamt 5 Punkte vorgesehen. Wenn der Algorithmus nicht erkannt wurde, sind diese Punkte nicht zu vergeben.

Die nächste Teilaufgabe befasst sich mit der Implementierung des Sortieralgorithmus in IJVM. Dafür sind in der Summe 36 Punkte zu erreichen. Diese große Anzahl setzt sich wie nachfolgend beschrieben zusammen:

- Implementieren der äußeren Schleife (8 Punkte)
- Implementieren der inneren Schleife (8 Punkte)
- Implementieren der if-Abfrage (5 Punkte)
- Idee und Umsetzung des Array-Zugriffs bzw. den Zugriff auf die jeweils benötigte der fünf Variablen (15 Punkte). Dabei ist es egal, ob der in der Musterlösung enthaltene Weg oder ein anderer eingeschlagen wird. Das Ziel muss lediglich sinngemäß passen.

Für die letzte Teilaufgabe können insgesamt 9 Punkte erreicht werden. Hierbei soll der Implementierung durch geeignete Testfolgen eine hinreichende Qualität nachgewiesen werden. Geeignete Testfolgen sind bspw. in der Musterlösung aufgezeigt und spiegeln das korrekte Verhalten bei nur gleichen, streng monoton steigenden, streng monoton fallenden und gemischten Eingabewerten wieder. Es sollten mindestens drei derer oder auch sinnvoll gewählte andere Testziele ersichtlich sein, sodass jedes entsprechend mit drei Punkten bewertet werden kann.

10 Aufgabentext

Der Aufgabentext des Laborversuchs befindet sich in der Datei 'Versuch_Mic-1_Simulator.pdf'. Dabei handelt es sich um das Aufgabenblatt, welches den Studierenden ausgehändigt wird. Bei der Aushändigung nicht zu Vergessen sind die Dateien und Programme:

- rechenOperation.jas
- ggT.jas
- sortieralgorithmus.jas
- Mic-1 Simulator, mit der darin enthaltenen 'echo.ijvm'-Datei

11 Musterlösung

Die Musterlösung ist in der Datei 'Musterlösung_Mic-1_Simulator.pdf' enthalten. Diese dient dem Dozenten als Hilfe und kann, sofern gewünscht, nach Abgabe der Ergebnisse aller Laborgruppen den Studierenden zur Verfügung gestellt werden. Die PDF bezweckt ebenfalls ein Nachschlagewerk. Darüber hinaus sind für die Aufgaben 2 und 3 nachfolgende IJVM-Programme als Musterlösungen beigefügt.

- ggT-musterloesung.jas
- selectionsort-musterloesung.jas