technische universität
dortmund

Fakultät für
Informatik

Lehrstuhl Informatik 12
Eingebettete Systeme

Dr. Heiko Falk
Otto-Hahn-Straße 16
D-44227 Dortmund
Tel +49 231 755 6124
Fax +49 231 755 6116
Heiko.Falk@tu-dortmund.de

Diktatzeichen	Aktenzeichen	Ort	Datum	Dienstgebäude / Raum
		Dortmund	23. Juni 2009	Otto-Hahn-Straße 16 / E19

Compiler für Eingebettete Systeme – Übungsblatt Nr. 2

Es soll ein einfacher Code-Selektor realisiert werden, der eine Teilmenge von ANSI-C in Assemblercode für den Infineon TriCore übersetzt. In diesem Übungsblatt werden ausschließlich die C-Datentypen `int`, `short` und `char` betrachtet. Skalare Variablen werden ausschließlich lokale Variablen sein. Es wird angenommen, daß lokale skalare Variablen in virtuellen Registern abgelegt werden, von den beliebig viele zur Verfügung stehen. Felder werden ausschließlich globale Variablen sein, die im Hauptspeicher abgelegt werden.

1. Aufgabe: (Arithmetik) *(10 Punkte)*

(a) Erweitern Sie die vorgegebenen Regeln und fügen Sie neue Regeln in die Datei `1a.m4` ein, so daß für konstante Zahlen sowie für die C-Operatoren =, +, −, *, <, >, == und != auf lokalen skalaren Variablen Code generiert wird. *(5 Punkte)*
Hinweis: Lokale skalare Variablen werden in virtuellen Datenregistern abgelegt, die durch das Nichtterminal `reg` analog zur Vorlesung repräsentiert werden.
Die Terminale für o. g. C-Operatoren sind `tpm_IntConstExp`, `tpm_AssignExpASSIGN`, `tpm_BinaryExpPLUS`, `tpm_BinaryExpMINUS`, `tpm_BinaryExpMULT`, `tpm_BinaryExpLT`, `tpm_BinaryExpGT`, `tpm_BinaryExpEQ` und `tpm_BinaryExpNEQ`.

(b) Erweitern Sie die Grammatik in Datei `tc_rules.m4` (Zeile 350) um ein weiteres Nichtterminal `const9`, das Konstanten von 9 Bits Länge modelliert. Der Action-Teil von Regeln, die `const9` produzieren, soll einen `long`-Wert zurückgeben, der genau die Konstante enthält. *(5 Punkte)*
Fügen Sie in Datei `1b.m4` eine Regel ein, die `tpm_IntConstExp` nur dann nach `const9` ableitet, wenn die repräsentierte Konstante größer gleich (kleiner gleich) `minSignedConst9Value` (`maxSignedConst9Value`) ist. Diese Regel soll keinen Code erzeugen.
Fügen Sie für `tpm_BinaryExpPLUS`, `tpm_BinaryExpMULT`, `tpm_BinaryExpLT`, `tpm_BinaryExpGT`, `tpm_BinaryExpEQ` sowie `tpm_BinaryExpNEQ` Regeln hinzu, die Maschinencode generieren, in dem die 9-Bit-Konstante in die Befehlsworte eincodiert ist.

2. Aufgabe: (Kontrollfluß) *(10 Punkte)*

(a) Ergänzen Sie die Regeln in Datei `2a.m4`, so daß sie Code für if- und if-else-statements generieren. *(5 Punkte)*
Hinweise: if-else-statements enthalten einen sog. „impliziten Sprung“, mit dem unbedingt vom Ende des then-Zweigs direkt hinter das if-else-statement gesprungen wird. Daher muß die Regel für das Terminal `tpm_ImplicitJump` stets einen unbedingten Sprung erzeugen.
Die Regeln für `tpm_IfStmt` und `tpm_IfElseStmt` müssen einen bedingten Sprung in Abhängigkeit von der Bedingung, die in einem `reg` vorliegt, generieren.

(b) Ergänzen Sie die Regeln in Datei `2b.m4`, so daß sie Code für for-, while-do- und do-while-Schleifen generieren. *(5 Punkte)*

Konto Nr. 001 181 327
Sparkasse Dortmund
BLZ 440 501 99

Ust-Idnr DE 811258273
IBAN DE09 4405 0199 0001 1813 27
SWIFT DORTDE33

3. Aufgabe: (Felder) *(10 Punkte)*

Ergänzen Sie die Regeln in Datei `3.m4`, so daß sie Code für Zugriffe auf globale eindimensionale Felder generieren.

Hinweise: Hierzu ist das Terminal `areg` analog zu `reg` einzusetzen, das virtuelle Adreßregister modelliert. `areg` dient dazu, in einem Adreßregister die Adresse eines Feld-Elements vorzuhalten.

Um für Feld-Zugriffe Code zu erzeugen, ist es notwendig, für Feld-Symbole (`tpm_SymbolExp`) die Startadresse des Feldes in einem `areg` abzulegen, die Adresse eines Feld-Elements über das Terminal `tpm_IndexExp` in ein `areg` zu laden, um mit Hilfe einer solchen Adresse Zuweisungen an Felder über `tpm_AssignExpASSIGN` vorzunehmen, und schließlich ein Feld-Element über das Terminal `tpm_IndexExp` aus dem Speicher in ein `reg` zu laden,

Technische Hinweise:

- Loggen Sie sich mit den bereitgestellten Zugangsdaten an Ihrem Rechner ein. Loggen Sie sich von dort aus per SSH auf einer der folgenden LS12-Maschinen ein:

`ls12pc5.cs.tu-dortmund.de`	`ls12pc7.cs.tu-dortmund.de`
`ls12pc8.cs.tu-dortmund.de`	`ls12pc9.cs.tu-dortmund.de`

- Wechseln Sie ins Verzeichnis `WCC`; rufen Sie zum Bearbeiten der C++-Dateien des Code-Selektors den Editor `kate` auf. Bearbeiten Sie die Aufgaben dieses Blattes genau in obiger Reihenfolge, da die Aufgaben aufeinander aufbauen.
- Um Ihre bearbeiteten C++-Dateien zu übersetzen, rufen Sie im Verzeichnis `WCC` das Programm `make` auf.
- Zum Überprüfen, wie sich Ihr Compiler verhält, sind im Verzeichnis `WCC/tests` C-Test-Programme für die einzelnen Aufgaben abgelegt. Übersetzen Sie diese mit `wcc -O0 -Sv` *`file`*. Der produzierte Assemblercode ist in Datei *`file`*`.vreg.s` zu finden. Der Ablauf Ihrer Code-Selektion wird durch Debug-Ausgaben im aktuellen Fenster dargestellt, so daß Sie mitverfolgen können, welche Action-Teile jeweils für welche C-Konstrukte aufgerufen werden.

  Haben Sie alle drei Aufgaben erfolgreich bearbeitet, sollte Ihr Compiler in der Lage sein, den Bubblesort-Algorithmus aus Verzeichnis `WCC/tests/3` korrekt zu übersetzen.
- Rufen Sie nach jeder einzelnen Aufgabe Ihren Betreuer und demonstrieren Sie, wie sich Ihr Code-Selektor jeweils verhält.
- Zum Bearbeiten dieses Übungszettels benötigen Sie evtl. Hintergrundinformation zur verwendeten High-Level IR ICD-C und zur Klasse `CodeSelector`. Diese finden Sie (nur über Uni-Rechner erreichbar) unter

  `http://ls12-www.cs.tu-dortmund.de/ICD-C`

  bzw.

  `http://ls12-www.cs.tu-dortmund.de/codesel`

  Öffnen Sie im links erscheinenden Menü den Punkt `Data Structures`, und Sie erhalten detaillierte Informationen zu sämtlichen Klassen und Methoden.

  Der TriCore-Befehlssatz ist zu finden unter

  `http://ls12-www.cs.tu-dortmund.de/tricore.pdf`