ulm university universität
uulm

Prof. Dr. Heiko Falk | Universität Ulm | 89069 Ulm

Fakultät für Ingenieurwissenschaften und Informatik
Institut für Eingebettete Systeme / Echtzeitsysteme

Prof. Dr. Heiko Falk

James-Franck-Ring
D-89081 Ulm

Tel.: +49 731 50-24183
Fax: +49 731 50-24182
Heiko.Falk@uni-ulm.de
http://www.uni-ulm.de/in/esys

Ulm, 3. Juni 2013

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 denen 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 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_BinaryExpPLUS`, `tpm_BinaryExpMINUS`, `tpm_BinaryExpMULT`, `tpm_BinaryExpLT`, `tpm_BinaryExpGT`, `tpm_BinaryExpEQ` und `tpm_BinaryExpNEQ`.

(b) Die Grammatik enthält in Zeile 351 von Datei `tc_rules.m4` ein weiteres Nichtterminal `const9`, das Konstanten von 9 Bits Länge modelliert. Der Action-Teil von Regeln, die `const9` produzieren, gibt einen `long`-Wert zurück, der genau die Konstante enthält. *(5 Punkte)*
Ergänzen Sie in Datei `1b.m4` die Regel, die `tpm_IntConstExp` nur dann nach `const9` ableitet, wenn die repräsentierte Konstante größer gleich (kleiner gleich) `minSignedConst9Value` (`maxSignedConst9Value`) ist. Diese Regel soll keinen aktiven 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)*

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 unter Linux ein.
- 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.
- 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

  `https://wcc-web.informatik.uni-ulm.de/lehre/CfES/ICD-C`

  bzw.

  `https://wcc-web.informatik.uni-ulm.de/lehre/CfES/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

  `https://wcc-web.informatik.uni-ulm.de/lehre/CfES/tricore.pdf`