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	16. Juli 2009	Otto-Hahn-Straße 16 / E19

Compiler für Eingebettete Systeme – Übungsblatt Nr. 3

Es sollen einfache ILP-basierte Scratchpad-Allokationen für Programm-Code realisiert werden, die auf WCET-Daten beruhen.

1. Aufgabe: (SPM-Allokation von Funktionen) *(10 Punkte)*

(a) Realisieren Sie eine einfache Scratchpad-Allokation analog zu Kapitel 6, Folie 61 der Vorlesung, die komplette Funktionen (keine Daten) auf den Scratchpad verschiebt. Der Gewinn einer einzelnen Funktion bei Verschiebung auf den Scratchpad ergibt sich aus der Differenz der WCETs der Funktion im Hauptspeicher bzw. Scratchpad, multipliziert mit der Anzahl der Ausführungen der Funktion im worst case. *(8 Punkte)*
Ergänzen Sie hierzu die Methoden `computeWCETSpmMem()`, `computeWCETMainMem()`, `generateILPVariables()`, `generateILPConstraints()`, `generateILPObjective()`, `solveILP()` und `moveCode()` aus Datei `spm_function.cc`.

(b) Wenden Sie Ihre Optimierung auf den Benchmark `adpcm_g721_board_test.c` für die Scratchpad-Größen 0, 795, 1590, 2385, 3180, 3975, 4770, 5565, 6360, 7155 und 7954 Bytes an. Erzeugen Sie mit OpenOffice ein Diagramm, auf dessen X-Achse obige Scratchpad-Größen abgetragen sind, und das in einer Kurve die durch Ihre Optimierung erzielten WCETs zeigt. *(2 Punkte)*

2. Aufgabe: (SPM-Allokation von Funktionen und Basisblöcken) *(10 Punkte)*

(a) Realisieren Sie eine einfache Scratchpad-Allokation analog zu Kapitel 6, Folie 72 der Vorlesung, die komplette Funktionen und / oder einzelne Basisblöcke (keine Multi-Basisblöcke) auf den Scratchpad verschiebt. *(3 Punkte)*
Ergänzen Sie hierzu Datei `spm_basicblock.cc` analog zu Aufgabe 1.

(b) Wenden Sie Ihre Optimierung auf den Benchmark `adpcm_g721_board_test.c` für die gleichen Scratchpad-Größen wie in Aufgabe 1(b) an. Fügen Sie eine neue Kurve zum Diagramm von Aufgabe 1(b) hinzu. *(2 Punkte)*

(c) Vergleichen Sie die beiden Kurven des Diagramms. Was fällt bei einer Scratchpad-Größe von 7954 Bytes auf? *(5 Punkte)*
Wieso wird nicht das gesamte Programm auf den Scratchpad verschoben? Analysieren Sie zum Beantworten der Fragen das von Ihnen generierte ILP und dessen Lösung, indem Sie die Ausgabe des Compilers in eine Datei umleiten und diese öffnen. Betrachten Sie z. B. die Entscheidungsvariable, die zum Basisblock `_L0_0` gehört (Zeile 183 in Datei `adpcm_g721_board_test.c`). Sehen Sie sich diesen Basisblock auch im Assemblercode an (`wcc -O0 -S adpcm_g721_board_test.c`). Weshalb wird z. B. dieser Basisblock trotz ausreichend freier Scratchpad-Kapazität nicht auf den Scratchpad verschoben?

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: (Einfache Sprung-Strafe) *(10 Punkte)*

(a) Erweitern Sie Ihr ILP zur Allokationen von Funktionen und Basisblöcken, so daß Sprünge zwischen aufeinanderfolgenden Basisblöcken, die in unterschiedlichen Speichern plaziert wurden, bestraft werden. *(8 Punkte)*
Realisieren Sie dazu die Strafe jp^i_{impl} analog zu Kapitel 8, Folie 74. Ergänzen Sie zunächst die Methode `LLIR_BasicBlockSPM::generateXOR`, die zwei Entscheidungsvariablen x_i und x_j für Basisblöcke b_i und b_j als Parameter erhält. Diese Methode soll Nebenbedingungen in Ihr ILP einbauen und eine neue binäre Entscheidungsvariable zurückgeben, die das Boolesche XOR von x_i und x_j enthält.
Rufen Sie danach `generateXOR` für alle zwei aufeinanderfolgenden Basisblöcke b_i und b_j in Methode `generateILPConstraints` auf. Multiplizieren Sie die von `generateXOR` zurückgegebene Entscheidungsvariable in Methode `generateILPObjective` mit der Straf-Konstanten 16 und fügen Sie diesen Straf-Term der Zielfunktion hinzu.
Hinweis: Stellen Sie XOR mit Hilfe des Booleschen AND, OR und NOT dar. Überlegen Sie, wie Sie Ungleichungen formulieren können, um das Boolesche AND und OR im ILP zu modellieren. Komponieren Sie XOR mit Hilfe dieser Ungleichungen.

(b) Wenden Sie Ihre Optimierung auf den Benchmark `adpcm_g721_board_test.c` für die gleichen Scratchpad-Größen wie in Aufgabe 1(b) an. Fügen Sie eine neue Kurve zum Diagramm von Aufgabe 2(b) hinzu. *(2 Punkte)*

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, ist im Verzeichnis `WCC` das Programm `adpcm_g721_board_test.c` hinterlegt. Übersetzen Sie dieses für Aufgabe 1 mit `wcc -O0 -S adpcm_g721_board_test.c -fpspm-fct --param pspm-size=`*`SPM-Größe`*. Für Aufgaben 2 und 3 verwenden Sie den Schalter `-fpspm-fctbb` anstatt `-fpspm-fct`.
- Rufen Sie nach jeder einzelnen Aufgabe Ihren Betreuer und demonstrieren Sie, wie sich Ihr ILP jeweils verhält.
- Zum Bearbeiten dieses Übungszettels benötigen Sie evtl. Hintergrundinformation zur verwendeten Low-Level IR ICD-LLIR und zur ILP-Bibliothek. Diese finden Sie (nur über Uni-Rechner erreichbar) unter

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

  bzw.

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

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