54. Analysewerkzeuge für Worst-Case Execution Time (Attributanalyse)

Prof. Dr. rer. nat. Uwe Aßmann

Institut für Software- und Multimediatechnik

Lehrstuhl Softwaretechnologie

Fakultät für Informatik

TU Dresden

http://st.inf.tu-dresden.de

Version 12-1.0, 12.01.13

1) WCETA mit AiT

Obligatorische Literatur

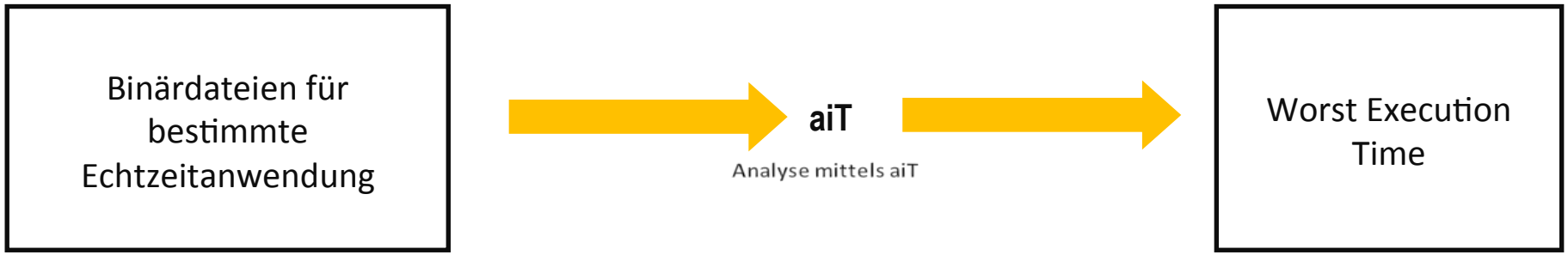
- Reinhold Heckmann, Christian Ferdinand. Worst-Case Execution Time Prediction by Static Program Analysis. AbsInt Angewandte Informatik GmbH. Science Park 1, D-66123 Saarbrücken, Germany, http://www.absint.com/aiT_WCET.pdf
- www.sureal-projekt.de

54.1 AiT - Worst-Case Execution Time Analyzer

http://www.absint.com/ait/

Was ist AiT?

- AiT ist ein Analysator für “worst-case execution time” (WCET)
  - berechnet die längste Ausführungszeit eines Programmes, z.B. für Echtzeitsysteme
  - berechnet dazu für jedes Statement ein Attribut (Attributanalyse), wie lange es höchstens dauert
  - bezieht eine Cache-Analyse und eine Pipeline-Analyse mit ein
- AiT benutzt PAG und abstrakte Interpretation



Warum braucht man WCET?

- Sicherstellung der Ausführungszeiten der Tasks innerhalb bestimmter Schranken des Echtzeitsystems:

Beispielanwendungen
fly by wire Flugzeuge
Airbagsysteme
Motorsteuerung
ABS-Steuerung
brake by wire

Probleme
Komplexe Laufzeitbestimmung durch Caches und Pipelines
Analysemethoden, die Caches und Pipelines ignorieren, liefern zu hohe Zeitschranken → Ressourcenverschwendung
Test- und Messverfahren unsicher, keine Garantie der oberen Schranken möglich

- aiT berechnet automatisch obere Zeitschranken Worst-Case Execution Time (WCET)

Schritte

1) **Kontrollfluss-Analyse:** Rekonstruktion des Kontrollflusses aus den ausführbaren Binärdateien des zu analysierenden Systems
2) **Value-Analyse:** Bestimmung von Schleifengrenzen und Berechnung von Adreßbereichen für Instruktionen, die auf den Speicher zugreifen
3) **Cache-Analyse:** Klassifizierung der Speicherzugriffe als Cache-Treffer oder Cache-Verfehlen
4) **Pipeline-Analyse:** Vorausberechnung des Verhaltens der Prozessor-Pipeline
5) **Pfadanalyse:** Bestimmung des Worst-Case-Ausführungspfades

http://www.absint.com/ait/analyse.htm

Überblick

Sourcecode

```
void do_something()
{
sleeping();
eating();
while(drinking())
{
drink_more();
}
}
```

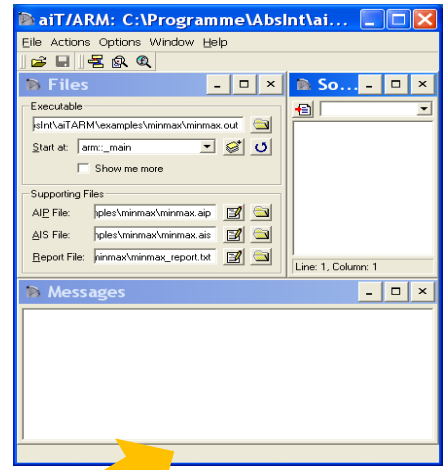
Compiler

Binärdatei

```
à  =€ @€  aŒ† | @€ ,@€ ;ÞKÿÿô;ÿ
```

Spezifikationen zu Prozessor

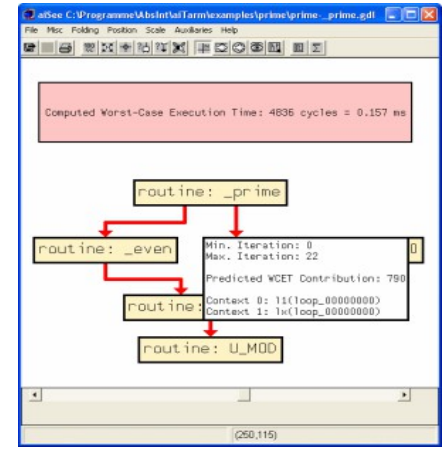
aiT

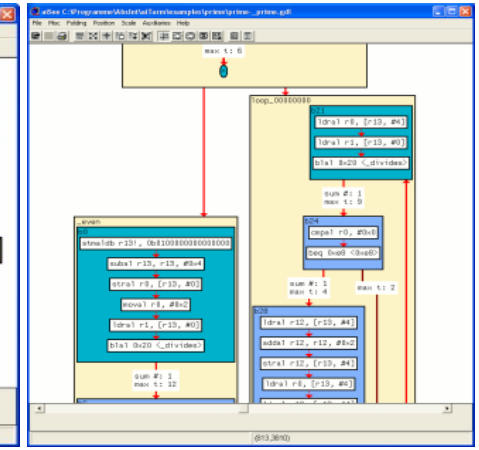


Einsprungpunkte im Programm

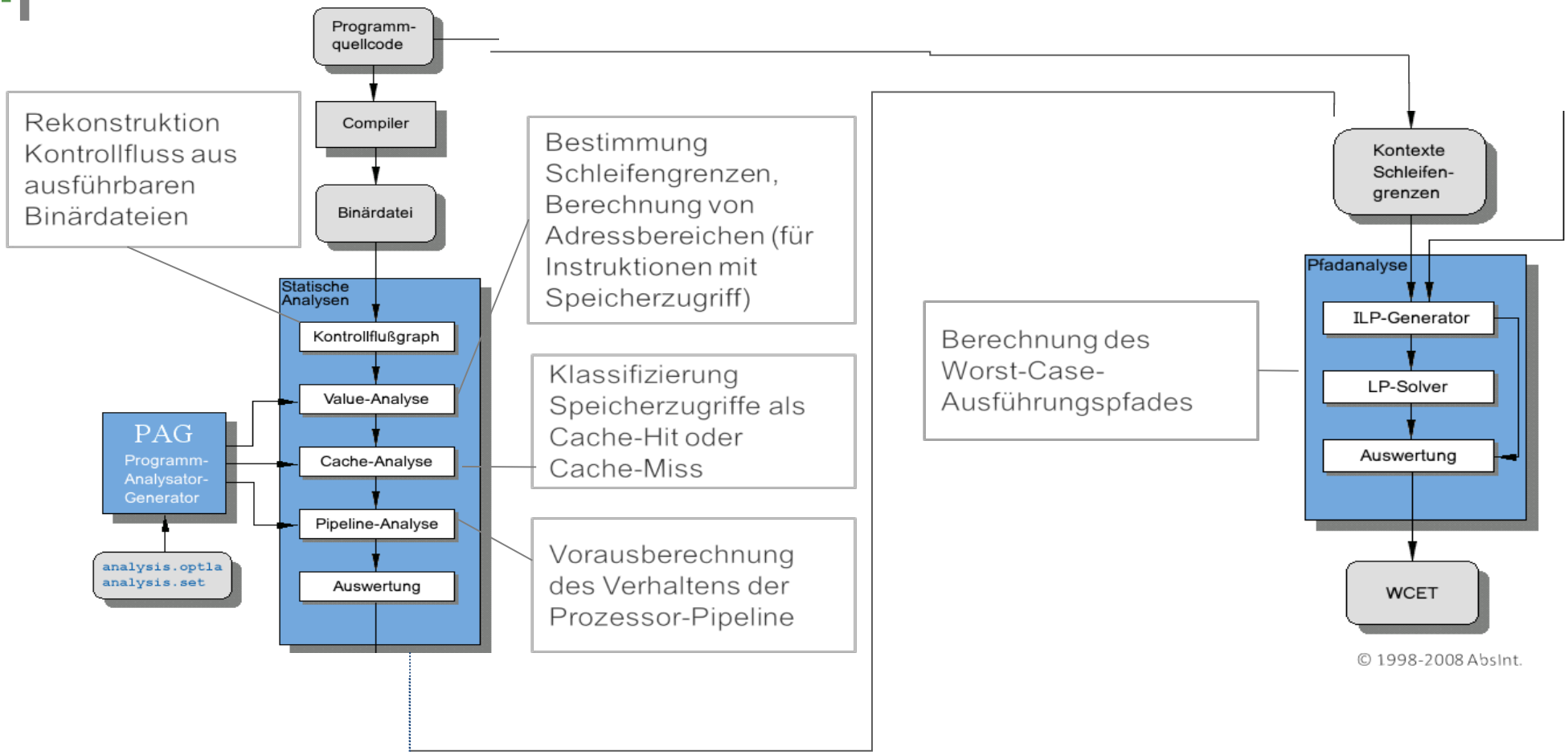
Worst-Case-Execution-Time

Visuelle Darstellungen des Tools





Phasen-Struktur von AiT



Behandlung von Schleifen

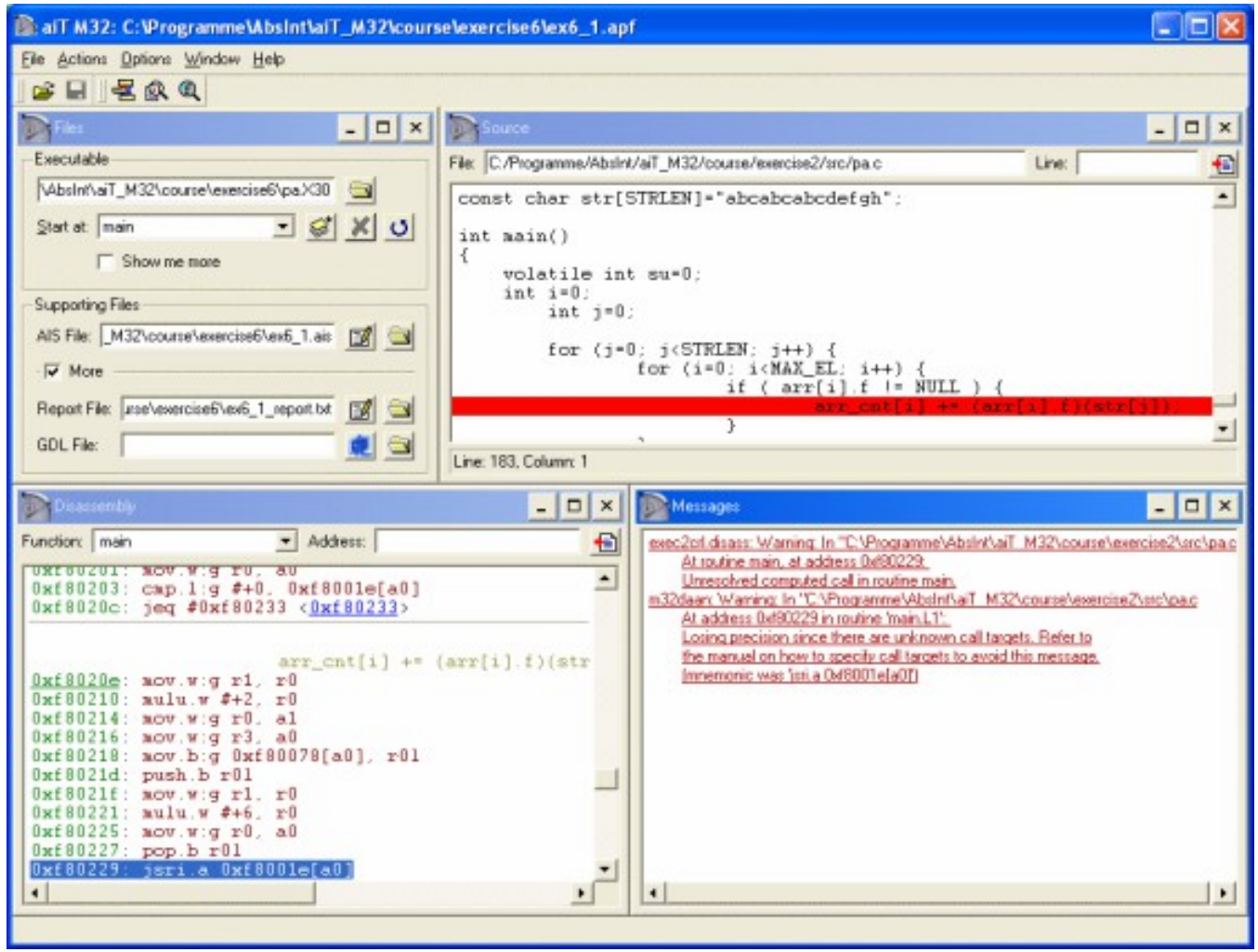
- Schleifen und Rekursionen verursachen großen Teil der Ausführungszeit
  → gesonderte Behandlung notwendig
- meistens im 1. Durchlauf der Schleife Daten in den Cache geladen, danach meist nur noch Cachezugriffe:

Naive Methode:
→Zusammenfassung des Zustandes des Caches vor Schleife mit Zustand nach Schleife
→Informationen über Schleifenverhalten geht verloren

aiT:
Programm-Analysator-Generator PAG nutzt implementierte Methoden zum virtuellen Abrollen der Schleifen
→ Analyse der Speicherzugriffe einzelner Schleifendurchläufe möglich

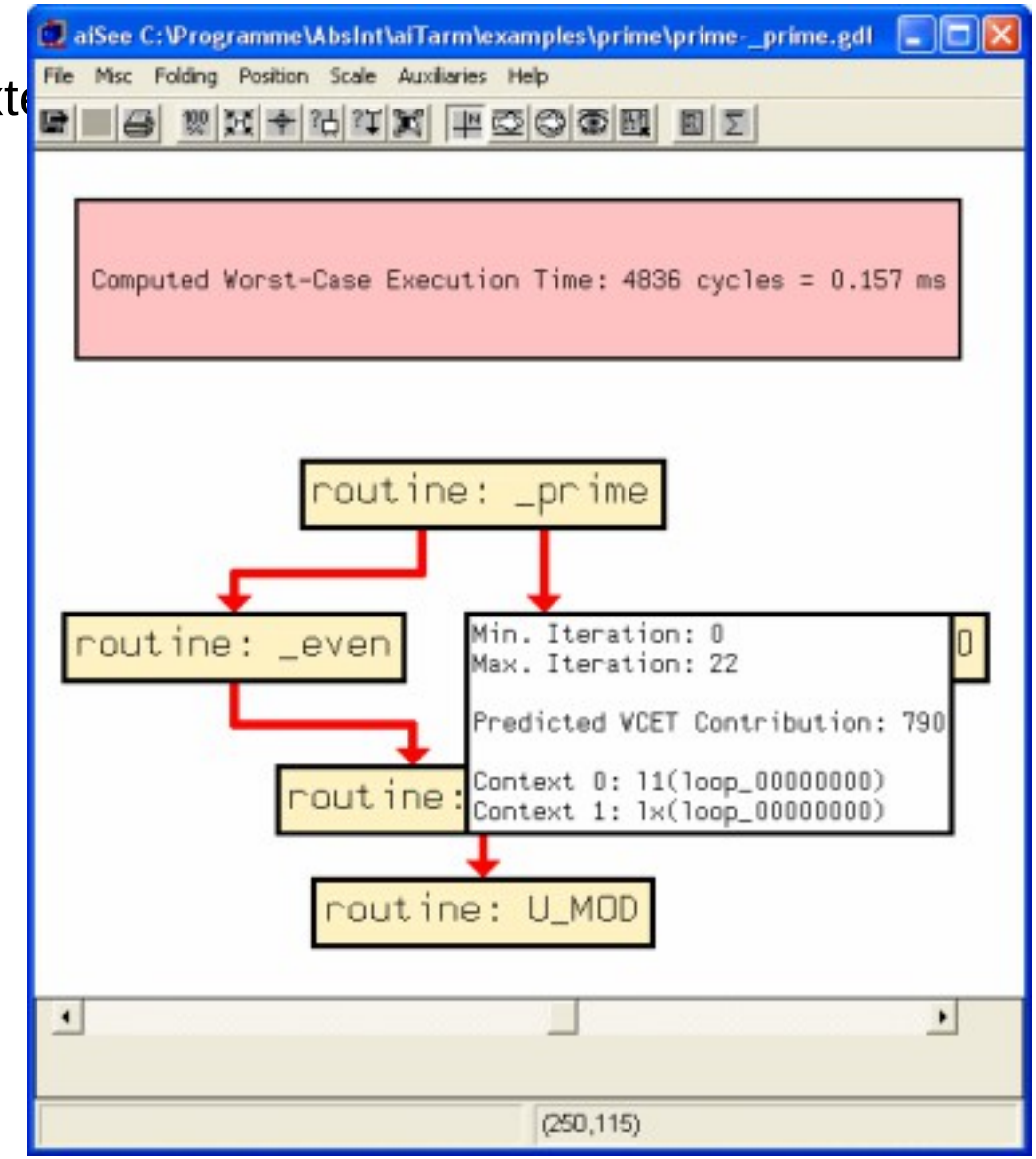
AiT Screenshot

- Die folgenden Bilder sind aus http://www.absint.com/ait/galerie.htm

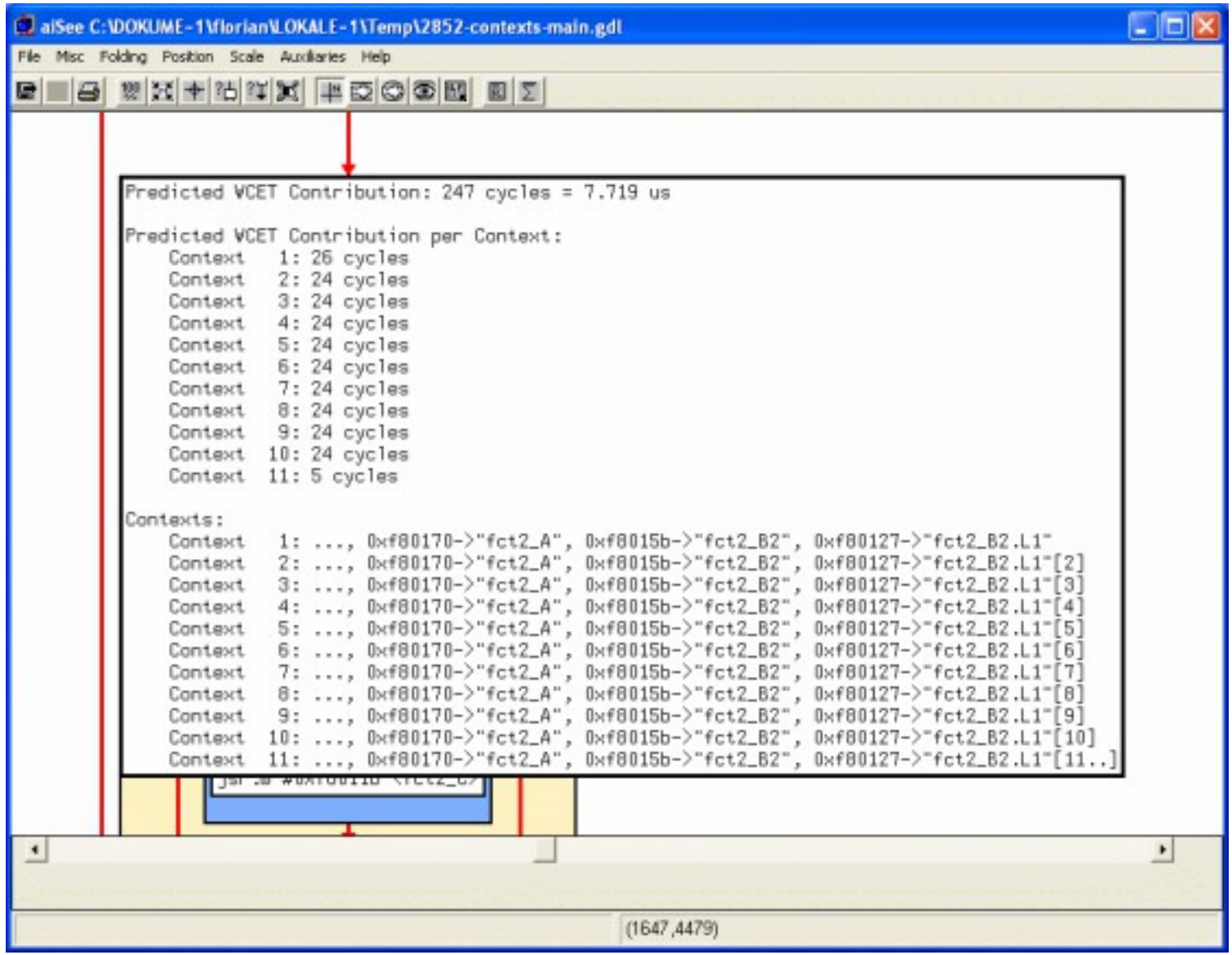


Präsentation einer WCET

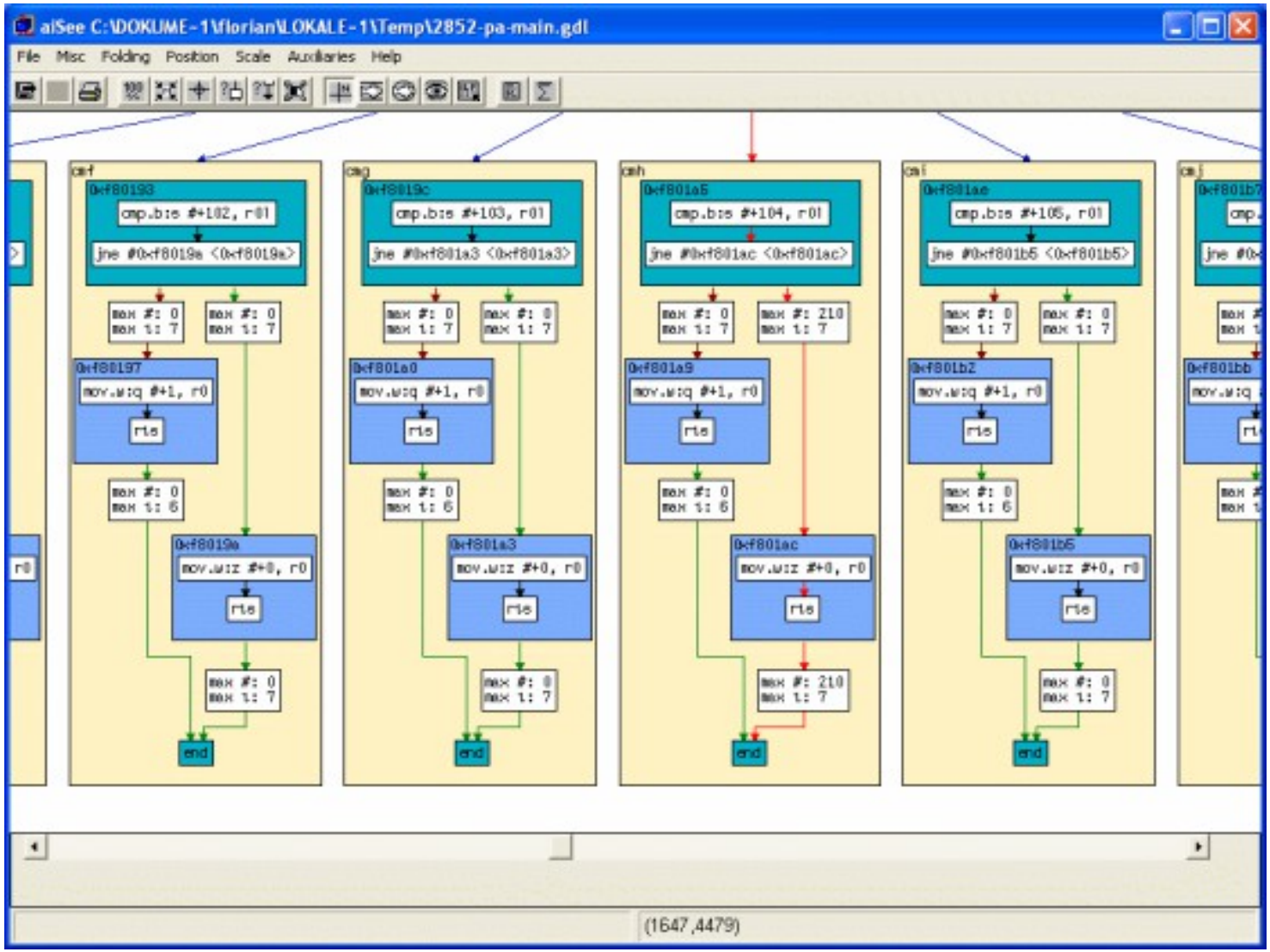
- Interprozedurale Analyse
- Unterscheidung von Aufrufkontexte



Berechnete Attribute



Schlechtester Pfad in Rot

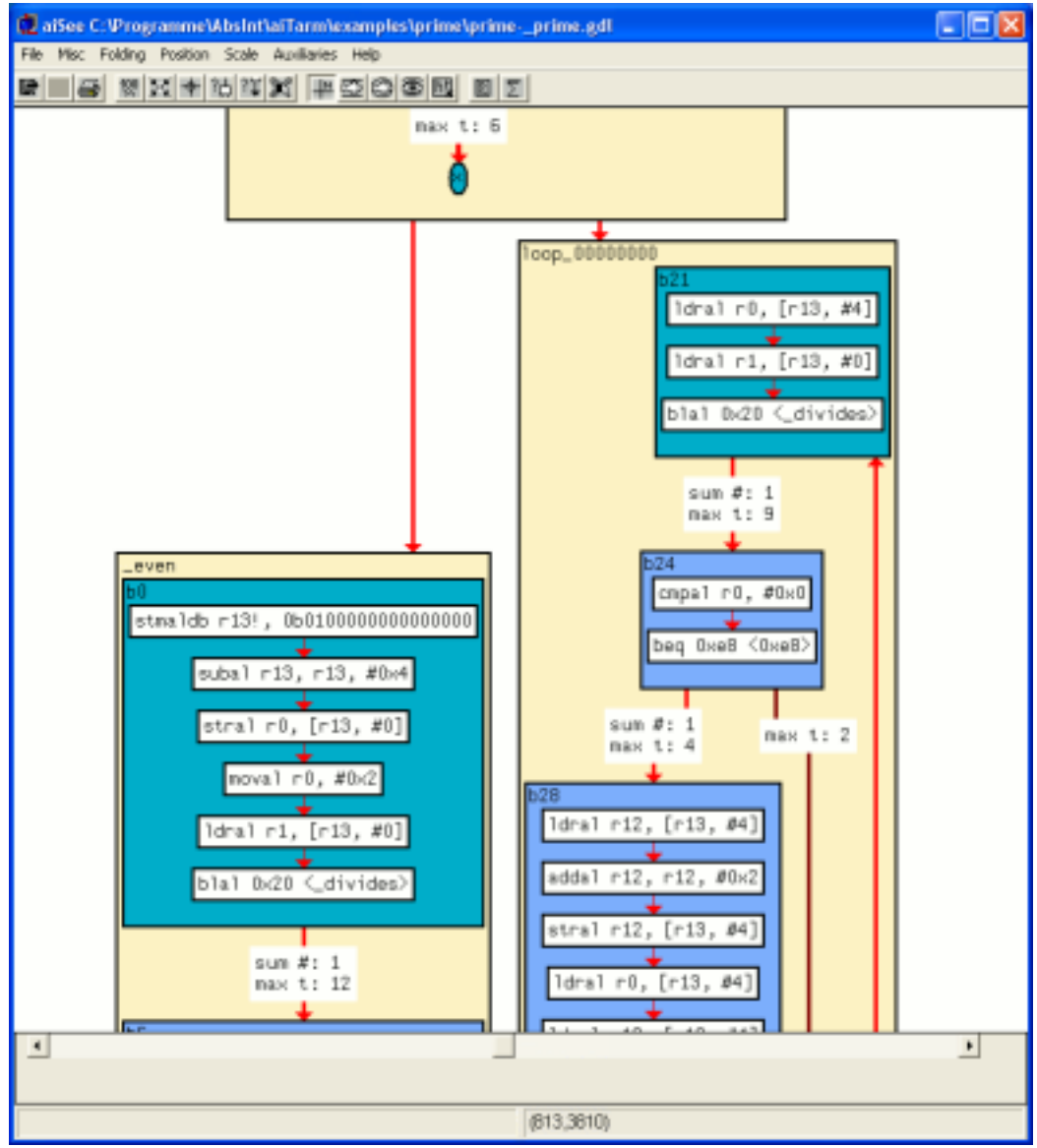


aiT für ARM7 TDMI - Basisblock-Ansicht

sum #: Anzahl Durchläufe im schlimmsten Fall

max t: die WCET des Basisblocks, von dem die Kante ausgeht.

Für jede Instruktion kann die Menge aller möglichen Pipeline-Zustände visualisiert werden



aiT für MPC5xx: Visualisierung der Ergebnisse der Pipeline-Analyse

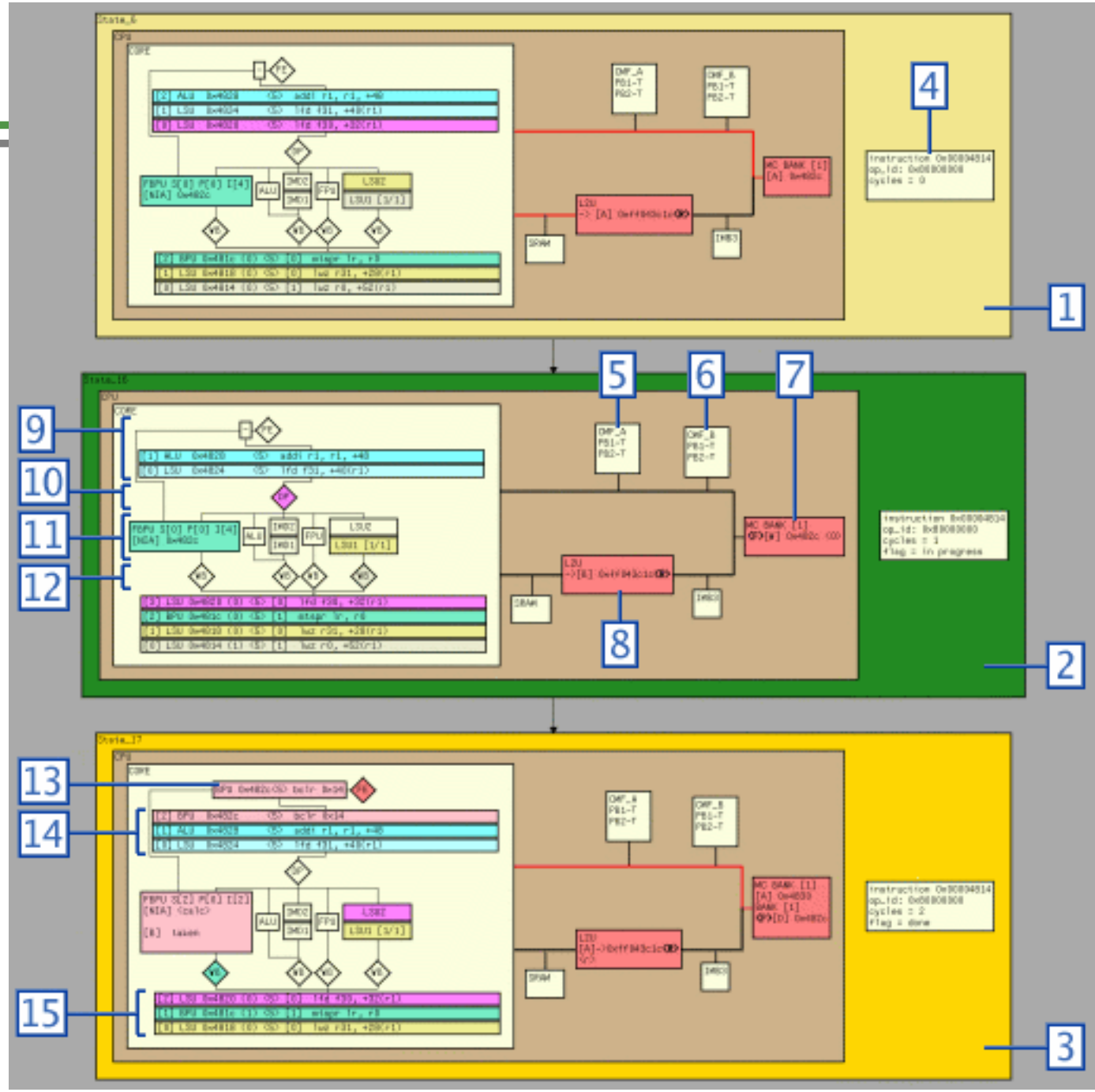
Für eine einzige Instruktion: Jeder gelbe bzw. grüne Untergraph entspricht einem Pipeline-Zustand.

Der in aiT integrierte Graph-Browser aiSee ermöglicht interaktives Erkunden von CPU- und CPU-Core-Zuständen an beliebigen Stellen im Graphen.

aiT für MPC5xx

Visualisierung der Pipeline-Analyseergebnisse

1. Startzustand*
2. Zwischenzustand*
3. Endzustand*
4. Zustandsbeschreibung
5. Flash A
6. Flash B
7. Memory-Controller
8. L2U
9. Fetch
10. Dispatch
11. Execute
12. Write-back
13. Decode-Buffer
14. Prefetch-Queue
15. History-Queue



The End

- Für fly-by-wire und andere sicherheitskritische Anwendungen ist WCETA notwendig
- Die Ergebnisse können zur Zertifizierung der Anwendungen eingesetzt werden (z.B. gegenüber dem TÜV)