1. Praktikum

Regelung, WCET-Analyse, RT Scheduling

13.10.2010

Aufgabe 1: Regelungsverfahren

Implementieren Sie einen Rahmen für ein Steuerungs- und Regelungsverfahren mit einem Mittel Ihrer Wahl – also eine Umsetzung des folgenden Pseudo-Codes:

```
void main (void){
  int state1, state2,…, staten;

  initializeClock();

  while(1){
    readSensors(…);

    if (CalculateNewSetValues(…))
        ActuateProcess(…);
    else
        Error();
    WaitForNextCycle(…);
  }
}
```

Aufgabe 2: WCET-Analyse

Bestimmen Sie den maximalen Aufwand des folgenden Code-Fragments:

```
for (int y = 0; y < 20; y++){
  if(test(y) == 1){
    for (x = 0; x < 64; x++){
      if(image[x][y] == 1){
        int weight;
        weight = calc_weight(image, x, y);
        x_sum += x * weight;
        y_sum += y * weight;
        squares += weight;
      }
    }
  }
}
```

Die Kosten der Operationen/Funktion sind wie folgt (Pipelining und Caching können ignoriert werden):

- STORE, ACCESS: 1
- ADD, SUB, COMPARE: 1; MUL, DIV: 2
- JUMP: 1
- CALL: 10; WCET(`test()`): 180; WCET(`calc_weight()`): 350

Tipp: Benutzen Sie ein Tabellen-Kalkulationsprogramm

Aufgabe 3: RT Scheduling

Implementieren Sie eine Simulationsumgebung für die folgenden Scheduling-Verfahren:

- **Rate Monotonic (RMS)**
- **Deadline Monotonic (DMS)**

Dabei werden die **vorliegenden Tasks** in einer ASCII-Datei der folgenden Form beschrieben:

```
A 1 3  3
B 1 6  6
C 1 5  5
D 2 10 9
```

Jede Zeile drückt dabei aus:

```
Name ComputationTime Period Deadline
```

Fehlt `D`, so gilt `D=P`.

Die **Ausgabe** des Programms ist die Ausführungsreihenfolge über das kleinste gemeinsame Vielfache der Periodenlängen.

Beispiel: `ACBADCABDA...`

Achten Sie auf eine möglichst gut zu erweiternde d.h. modulare Lösung, damit später weitere Algorithmen integriert werden können.