06 The microprocessor core

06.05 Micro-benchmarks

- Incremental micro-benchmarks
- CPI characterization
- Latency characterization
- Repetition time characterization
- Branch characterization
- Dealing with OOO execution

Problem statement

- Objective:
  - Empirical characterization of the performance model of a CPU (CPI, LAT and RT of each instruction)
- Issues:
  - Single contributions not directly accessible
  - The overall CPUT of a benchmark can be measured with a limited resolution
- Approach
  - Isolation of the contribution under characterization by means of differential measures of the CPUT of incremental synthetic benchmarks
  - Amplification of the contribution under characterization by means of loops that repeat the execution of the same code segment a large number of times

Incremental benchmarks

- CUT:
  - code under test
- Incremental benchmark
  - Benchmark B0 doesn't contain CUT
  - Benchamrk B1 contains CUT
- Differential measure:

$$CPUT(CUT) = \frac{CPUT(B1) - CPUT(B0)}{N} \quad (1)$$

- Implementation issues:
  - CUT must be independent from code segments 0 and 1
- Property:

$$err(CPUT(CUT)) = \frac{err(CPUT(B))}{N} \quad (2)$$

Loop control

Decoupling

CUT

```
code segment A;
for (i = 0; i < N; i++) {
      code segment 0;
      CUT;
      code segment 1;
}
code segment B;
```


Characterization of CPI

- CUT:
  - The CUT contains only the instruction under measurement (*instr*)
- Differential measurement
  - Two measures are sufficient to compute the CPI from equation (1):

    CPI(*intsr*) = *CPUT*(*CUT*)
- Implementation issue:
  - Instruction *instr* must be independent from code segments 0 and 1

Characterization of LAT (1)

- CUT:
  - To measure the LAT between instr1 and instr2
  - Use a first CUT (called CUT0) that contains only instr1
  - Use a second CUT (called CUT1) that contains both instr1 and instr2
- Differential measurement
  - Use equation (1) twice to compute CPUT(CUT0) and CPUT(CUT1)
  - Compute LAT by means of:

$$LAT(instr1, instr2) = CPUT(CUT1) - CPUT(CUT0) \quad (3)$$

Characterization of LAT (2)

- CUT:
  - To measure the LAT between instr1 and instr2
  - Use a parametric CUT (called CUT<n>) that contains instruction 1, followed by n NOPs, followed by instr2
- Differential measurement
  - Use equation (1) twice to compute CPUT(CUT<n>) for all benchmarks
  - If CPUT(CUT<n>)=CPUT(CUT0), than the n NOPs are masked by latency (i.e., they replace the CPU stalls caused by the latency in CUT0)
  - If N is the maximum number of masked NOPs

$$LAT(instr1, instr2) = CPI(instr1) + N \cdot CPI(NOP) \quad (4)$$

Characterization of RT

- Objective:
  - Characterization of the repetition time between two instructions instr1 and instr2
- Approach:
  - The repetition time can be measured by using the same techniques outlined for characterizing the latency
- Implementation issue:
  - Instructions instr1 and instr2 must be data-independent

Characterization of branches

- Untaken branches:
  - The CUT contains only a branch conditioned to a never-true condition
- Taken branches:
  - The CUT contains a conditional branch followed by one or more dummy instructions
  - The conditional branch has an always-true condition and leads to the first instruction of code segment 1

Dealing with OOO execution (1)

- Issues:
  - Dynamic scheduling makes it more difficult to build effective benchmarks to characterize instruction-specific performance parameters
  - The programmer has no control on the actual issuing order
- Implementation requirements:
  - The instructions in the CUT are executed in the order they appear in the code if and only if they are in the most convenient position (otherwise dynamic reordering is performed, possibly interleaving the CUT with code segments 1 and 2)

Dealing with OOO execution (2)

- Micro-benchmarks with repeated CUT:
  - B<m> contains m independent replicas of the CUT
  - The CPUT of the benchmarks is plotted as a function of m
  - For m larger than m', the CPUT grows linearly with m
  - The CPUT(CUT) is computed as

$$CPUT(CUT) = \frac{CPUT(B_{m+1}) - CPUT(B_m)}{N} \quad (5)$$

    with m > m'
  - Code segments 0 and 1 are not strictly required

Loop control

Decoupling

CUT

```
code segment A;
for (i = 0; i < N; i++) {
    code segment 0;
    CUT;        1
    CUT;        2
    ...
    CUT;        m
    code segment 1;
}
code segment B;
```