Quiz 4 Review

Microcoded and VLIW processors & Vector processors

Guowei Zhang

Lab 4

- Request a deadline extension till Wednesday 13 midnight if needed

Quiz 4 logistics

- Time: 1pm on Tuesday May 12
- Style: same as Quiz 3
- Zoom link: same as recitations

Topics

- Microcoded and VLIW processors
- Vector processing
- GPUs
- Transactional memory

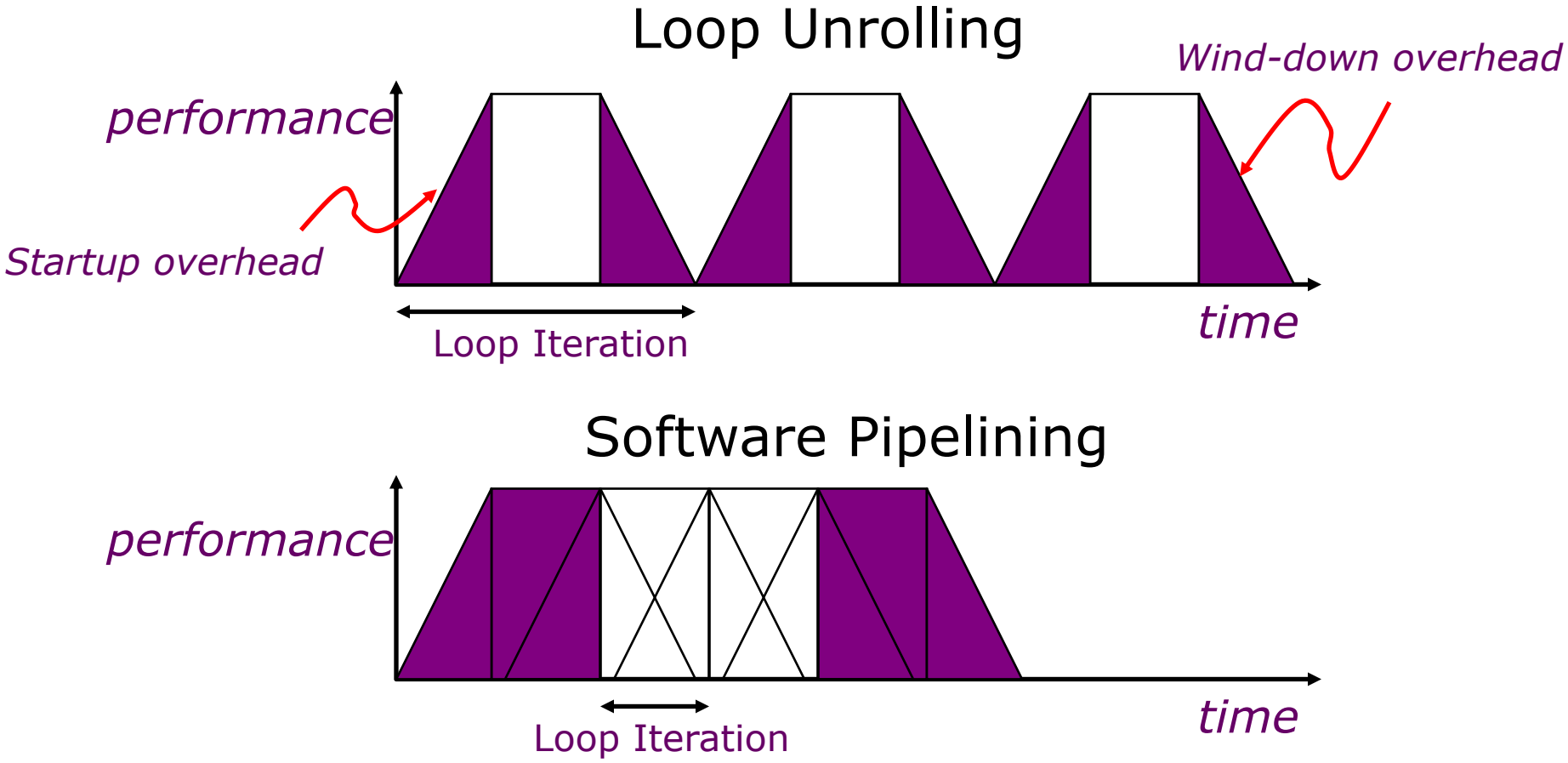
Microcoded processors

- Introduces a layer of interpretation
  - Each ISA instruction is executed as a sequence of simpler microinstructions
- Pros:
  - Enables simpler hardware
  - Enables more flexible ISA
- Cons:
  - Sacrifices performance

VLIW: Very Long Instruction Word

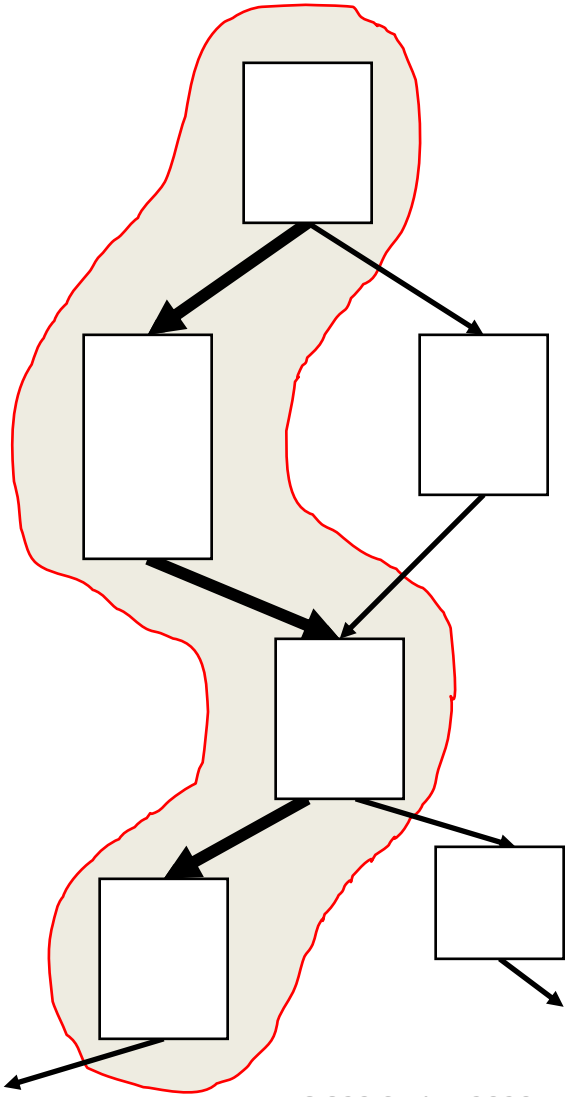
- The compiler:
  - Guarantees intra-instruction parallelism
  - Schedules (reorders) to maximize parallel execution
- The architecture:
  - Allows operation parallelism within an instruction
    - No cross-operation RAW check
  - Provides deterministic latency for all operations
- Enables simple hardware but leaves hard tasks to software

Software pipelining vs. Unrolling



Software pipelining pays startup/wind-down costs only once per loop, not once per iteration

Trace scheduling



VLIW issues

- Limited by static information
  - Unpredictable branches
    - Possible solution: predicated execution
  - Unpredictable memory operations
    - Possible solution: Memory Latency Register (MLR)
- Code size explosion
  - Wasted slots
  - Replicated code
- Portability
- Compiler complexity

Vector processing

- Supercomputers in 70s – 80s
- Multimedia/SIMD extensions in current ISAs

- Single-Instruction Multiple-Data (SIMD)

- Typical hardware implications
  - Simpler instruction fetch due to fewer instructions
  - Banked register files/memory due to simple access patterns

Vector processing

- Vector chaining
- Vector stripmining
- Vector scatter/gatter
- Masked vector instructions

Example: Masks

Problem: Want to vectorize loops with conditional code:

```
for (i = 0; i < N; i++)
    if (A[i] > 0) then
        A[i] = B[i];
```

Solution: Add vector *mask* (or *flag*) registers

- vector version of predicate registers, 1 bit per element

…and *maskable* vector instructions

- vector operation becomes NOP at elements where mask bit is clear

Code example:

```
CVM              # Turn on all elements
LV vA, rA        # Load entire A vector
SGTVS.D vA, F0   # Set bits in mask register where A>0
LV vA, rB        # Load B vector into A under mask
SV vA, rA        # Store A back to memory under mask
```

Wish you all the best!