Very Long Instruction Word Architectures

- Compiler extracts parallelism
  - Static scheduling
  - Trace scheduling
- Multiple operations to various execution units grouped together in one very long instruction
- Compiler schedule based on pre-knowledge of hardware execution units and latencies
- Simple hardware
  - No dependence checks in hardware
Trace Scheduling

- Profiling to identify most frequent traces
- Compiler schedules as if trace is one big basic block
  - Moves instructions across branches
  - Moves loads above earlier stores
  - Compensation code for early exits from trace
  - NOPs inserted when slots cannot be filled with parallel operations
  - Compiler schedules for a specific hardware implementation
- Repeat scheduling the next most frequent trace, including compensation code
VLIW Issues

- Backward compatibility
  - Requires recompilation for new hardware
- Low code density
  - NOPs and compensation code
- Compiler cannot anticipate dynamic events or account for variable execution latencies
  - Cache misses
  - Memory disambiguation
  - Branch outcomes
EPIC and the Itanium

- Solves some of the issues in VLIW
  - Compiler provides register dependence information and hardware schedules accordingly
    - Eliminates NOPs
    - Provides backward compatibility
  - Provides hardware mechanisms for events that cannot be predicted by the compiler
    - Predication for branches
    - Control speculation
    - Data speculation
Paper Reading Assignment

- Dynamic Instruction Reuse
- The Predictability of Data Values
- Limits of Control Flow on Parallelism
- MultiScalar