Data Flow Summary

- Availability of data determines order of execution
- A data flow node fires when its sources are ready
- Programs represented as data flow graphs (of nodes)

- Data Flow at the ISA level has not been (as) successful

- Data Flow implementations under the hood (while preserving sequential ISA semantics) have been successful
  - Out of order execution
  - Hwu and Patt, “HPSm, a high performance restricted data flow architecture having minimal functionality,” ISCA 1986.
Data Flow Characteristics

- Data-driven execution of instruction-level graphical code
  - Nodes are operators
  - Arcs are data (I/O)
  - As opposed to control-driven execution
- Only real dependencies constrain processing
- No sequential I-stream
  - No program counter
- Operations execute asynchronously
- Execution triggered by the presence of data
- Single assignment languages and functional programming
  - E.g., SISAL in Manchester Data Flow Computer
  - No mutable state
Data Flow Advantages/Disadvantages

- Advantages
  - Very good at exploiting irregular parallelism
  - Only real dependencies constrain processing

- Disadvantages
  - High bookkeeping overhead (tag matching, data storage)
  - Debugging difficult (no precise state)
    - Interrupt/exception handling is difficult (what is precise state semantics?)
  - Too much parallelism? (Parallelism control needed)
  - Implementing dynamic data structures difficult in pure data flow models
  - Instruction cycle is inefficient (delay between dependent instructions), memory locality is not exploited
Combining Data Flow and Control Flow

- Can we get the best of both worlds?

- Two possibilities
  - Model 1: Keep control flow at the ISA level, do dataflow underneath, preserving sequential semantics
  - Model 2: Keep dataflow model, but incorporate control flow at the ISA level to improve efficiency, exploit locality, and ease resource management
    - Incorporate threads into dataflow: statically ordered instructions; when the first instruction is fired, the remaining instructions execute without interruption
Model 2 Example: Macro Dataflow

- Data flow execution of large blocks, control flow within a block

Figure 1 An Example of a Strongly Connected Block.

Macro Dataflow Program Example

(a) A Normal Dataflow Program

(b) A Strongly Connected Dataflow Program

Figure 5  FIBONACCI Program.
Macro Dataflow Machine Example

Figure 6 Block Diagram of the EMC-R.

Figure 8 Execution Unit Organization.

IR: Instruction Register  OPi: Operand Register i
Macro Dataflow Pipeline Organization

Figure 9 Pipeline Organization of the EMC-R.
Model 1 Example: Restricted Data Flow

- Data flow execution under sequential semantics and precise exceptions

Restricted Data Flow DFG Formation

Figure 3.