Reviews

- Was due March 21

- Was due Saturday March 26

- Due Wednesday before class
Readings: Data Flow

- Review papers

- Recommended papers
Data Flow
Data Flow

- The models we have examined in 740 all assumed
  - Instructions are fetched and retired in sequential, control flow order

- This is part of the Von-Neumann model of computation
  - Single program counter
  - Sequential execution
  - Control flow determines fetch, execution, commit order

- What about out-of-order execution?
  - Architecture level: Obeys the control-flow model
  - Uarch level: A window of instructions executed in data-flow order → execute an instruction when its operands become available
Data Flow

- In a data flow machine, a program consists of data flow nodes
- A data flow node fires (fetched and executed) when all its inputs are ready
  - i.e. when all inputs have tokens

- Data flow node and its ISA representation
Data Flow Nodes

*Conditional

*Relational

*Barrier Synch
A small set of dataflow operators can be used to define a general programming language.
Dataflow Graphs

\{x = a + b; \\
y = b \times 7 \\
in \\
(x-y) \times (x+y)\}\}

- Values in dataflow graphs are represented as tokens

  token \( \langle \text{ip}, \text{p}, \text{v} \rangle \)

  instruction ptr  port  data

- An operator executes when all its input tokens are present; copies of the result token are distributed to the destination operators

  no separate control flow
Example Data Flow Program
Control Flow vs. Data Flow

\[ a := x + y \]
\[ b := a \times a \]
\[ c := 4 - a \]

**Figure 2.** A comparison of control flow and dataflow programs. On the left a control flow program for a computer with memory-to-memory instructions. The arcs point to the locations of data that are to be used or created. Control flow arcs are indicated with dashed arrows; usually most of them are implicit. In the equivalent dataflow program on the right only one memory is involved. Each instruction contains pointers to all instructions that consume its results.
Static Dataflow

- Allows only one instance of a node to be enabled for firing

- A dataflow node is fired only when all of the tokens are available on its input arcs and no tokens exist on any of its output arcs

Static Dataflow Machine: Instruction Templates

Each arc in the graph has an operand slot in the program

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Destination 1</th>
<th>Destination 2</th>
<th>Operand 1</th>
<th>Operand 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+</td>
<td>3L</td>
<td>4L</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>*</td>
<td>3R</td>
<td>4R</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>5L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>+</td>
<td>5R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>*</td>
<td>out</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Many such processors can be connected together

Programs can be statically divided among the processor
Static versus Dynamic Dataflow Machines

Figure 1. The basic organization of the static dataflow model.

Figure 2. An instruction template for the static dataflow model.

Figure 3. The general organization of the dynamic dataflow model.

Figure 4. An instruction format for the dynamic dataflow model.
Static Data Flow Machines

- Mismatch between the model and the implementation
  - The model requires *unbounded FIFO token queues* per arc but the architecture provides storage for one token per arc
  - The architecture *does not ensure FIFO* order in the reuse of an operand slot

- The static model *does not support*
  - Reentrant code
    - Function calls
    - Loops
  - Data Structures
Problems with Re-entrancy

- Assume this was in a loop
- Or in a function
- And operations took variable time to execute
- How do you ensure the tokens that match are of the same invocation?

Fig. 3. A firing sequence for “s + A[i] * B[i].”
Dynamic Dataflow Architectures

- Allocate instruction templates, i.e., a frame, dynamically to support each loop iteration and procedure call
  - termination detection needed to deallocate frames

- The code can be shared if we separate the code and the operand storage

![Diagram](attachment:image.png)
A Frame in Dynamic Dataflow

Need to provide storage for only one operand/operator
Monsoon Processor (ISCA 1990)
Concept of Tagging

- Each invocation receives a separate tag

Fig. 6. Dataflow graph for function call and return linkage.
Procedure Linkage Operators

Like standard call/return but caller & callee can be active simultaneously

token in frame 0

- token in frame 1

Graph for f

1:  change Tag 0

Fork

n:  change Tag n

change Tag n

change Tag 1

change Tag 1

...
Function Calls

- Need extra mechanism to direct the output token of the function to the proper calling site

- Usually done by sending special token containing the return node address
Loops and Function Calls Summary

Figure 10. An implementation of a loop using tagged tokens. At the start of the loop a new tag area is allocated. Tokens belonging to consecutive iterations receive consecutive tags within this area. The tag from before the loop is restored on tokens that exit from the loop.

Figure 11. Interface for a procedure call. On the left a call of procedure $P$ whose graph is on the right. $P$ has one parameter and one return value. The actual parameter receives a new tag and is sent to the input node of $P$ and concurrently a token containing address $A$ is sent to the output node. This SEND-TO-DESTINATION node transmits the other input token to a node of which the address is contained in the first token. The effect is that, when the return value of the procedure becomes available, the output node sends the result to node $A$, which restores the tag belonging to the calling expression.
Control of Parallelism

- Problem: Many loop iterations can be present in the machine at any given time
  - 100K iterations on a 256 processor machine can swamp the machine (thrashing in token matching units)
  - Not enough bits to represent frame id

- Solution: Throttle loops. Control how many loop iterations can be in the machine at the same time.
  - Requires changes to loop dataflow graph to inhibit token generation when number of iterations is greater than N
Data Structures in Dataflow

- Data structures reside in a structure store
  - ⇒ tokens carry pointers

- I-structures: Write-once, Read multiple times or
  - allocate, write, read, ..., read, deallocate
    - ⇒ No problem if a reader arrives before the writer at the memory location