Example Tagged Token and Instruction Formats 8

example token format:

tag				data
frame pointer	**statement pointer**	**port**	**type**	**value**

where **frame pointer** = address of the start of the activation frame
statement pointer = address of the target statement
port = indicates left or right operand
type = integer, floating point etc.
value = typically 64 bits of data

example instruction format:

op-code	**(r)**	**dest1**	**(dest2)**

where **( )** indicates optional parameters
op-code is the instruction identifier
r is the activation frame offset number for dyadic operations
dest1 and **dest2** are the destination offsets (**dest2** being optional)

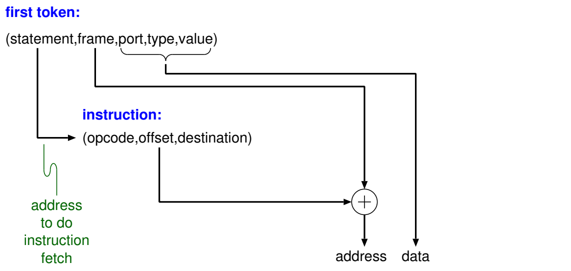
Matching Algorithm for Dyadic Operations 9

- incoming token's *statement pointer* is used to look up the *instruction*
- the instruction's *activation frame offset* is added to the token's *activation frame* number to give an *effective address*
- the *effective address* is then used to look up the the *presence* bit in the activation frame
- if the *presence* = empty then the token's value and port are written to the location
- if the *presence* = full then the stored value and token value should make up the two operands for the dyadic instruction (assuming their ports are different)
- the operation, its operands and the destination(s) are executed

note:

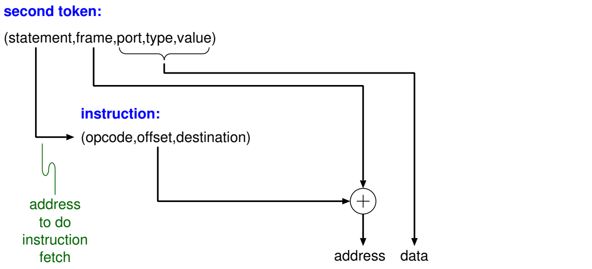
- these stages correspond to the stages in the pipeline

Matching Dyadic Operations cont... 10



use address to access activation frame to see if empty
the first time it will be empty so the data will be written

Matching Dyadic Operations cont... 11



use address to access activation frame to see if empty
this time it will be full so the data pair will be sent for execution

Example Tagged-token Data-flow Program 12

address	**instruction**		
(e.g.)	op-code	offset	destinations
0x30	`mul`	0,	0x31ℓ,nil
0x31	`add`	2,	0x33ℓ,nil
0x32	`div`	1,	0x31r,nil
0x33	`ret`	0,	(dest)ℓ,nil

note:

- `ret` accepts a $\langle \mathit{destination\ instruction}, \mathit{port}, \mathit{frame} \rangle$ triplet as its left parameter

advantages:

- simple matching algorithm which may be implemented using a pipeline
- garbage collecting unmatched tokens is easy

problems:

- pipeline bubble every time the first operand of an instruction is matched
- token explosion problem can still occur (careful code generation required)

Evaluation of Data-flow 13

advantages:

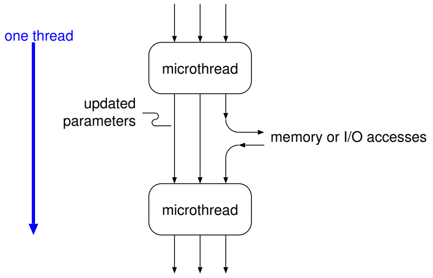
- inherently concurrent and latency tolerant (no need for caches)
- multiprocessor applications are easy to write

disadvantages:

- assignment a problem because there is too much concurrency, thus functional languages tend to be used. Furthermore, this makes I/O difficult
- ineffective use of very local storage (a register file or stack)
- scheduling policies have to be simple because of the instruction level concurrency

Multithreaded Processors — Combining Control-flow and Data-flow 14

example machine: Anaconda (Cambridge)



- unit of execution is larger so matching time does not dominate
- concurrency allows memory latency to be tolerated