# Data Flow Computer
# Module 4
# **Non von Neumann architectures**

Ref:Hwang Chapter 2 and

Hwang and Briggs Chapter 10 (page734/softcopy page 753) to page743/softcopy762

# Sequential or Parallel Control Flow

Characteristic Features:

- Data is passed between instructions via references to shared memory cells.
- Flow of control is implicitly sequential, but special control operators can be used explicitly for parallelism. (fort ,join statements used to create parallel process)
- Program counters are used to sequence the execution of instruction in a centralized control environment.

# Data Flow Model

Characteristic Features:

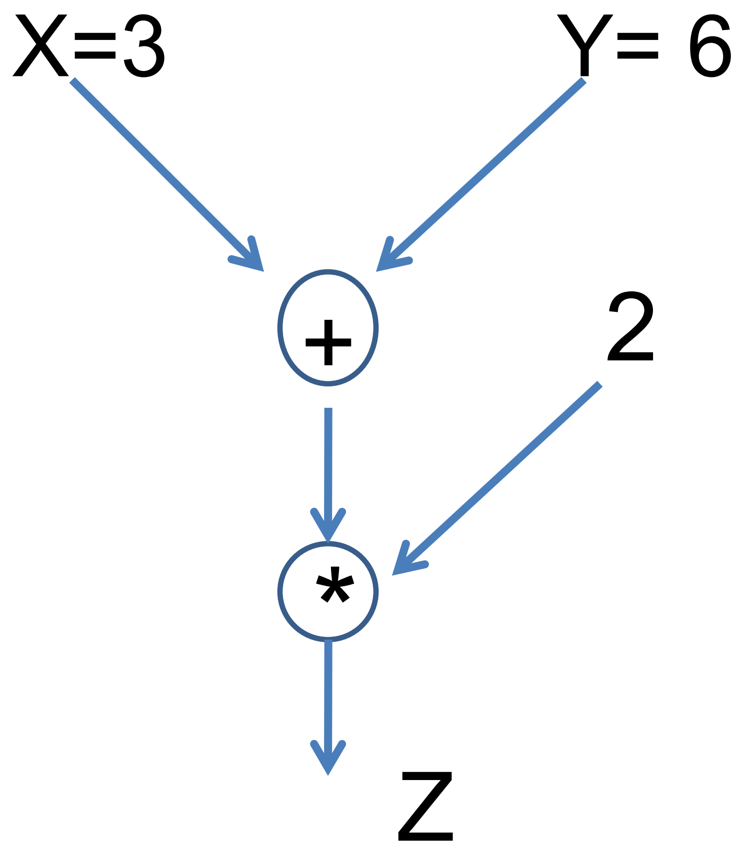
- Intermediate or final results are passed directly as data token between instructions.
- There is no concept of shared data storage as embodied in the traditional notion of a variable.
- Program sequencing is constrained only by data dependency among instructions.

•Note:

- **No Program Counter**
- **In data flow computers the machine level program is represented by data flow graphs**
- **Firing rule of instructions is based on data availability**

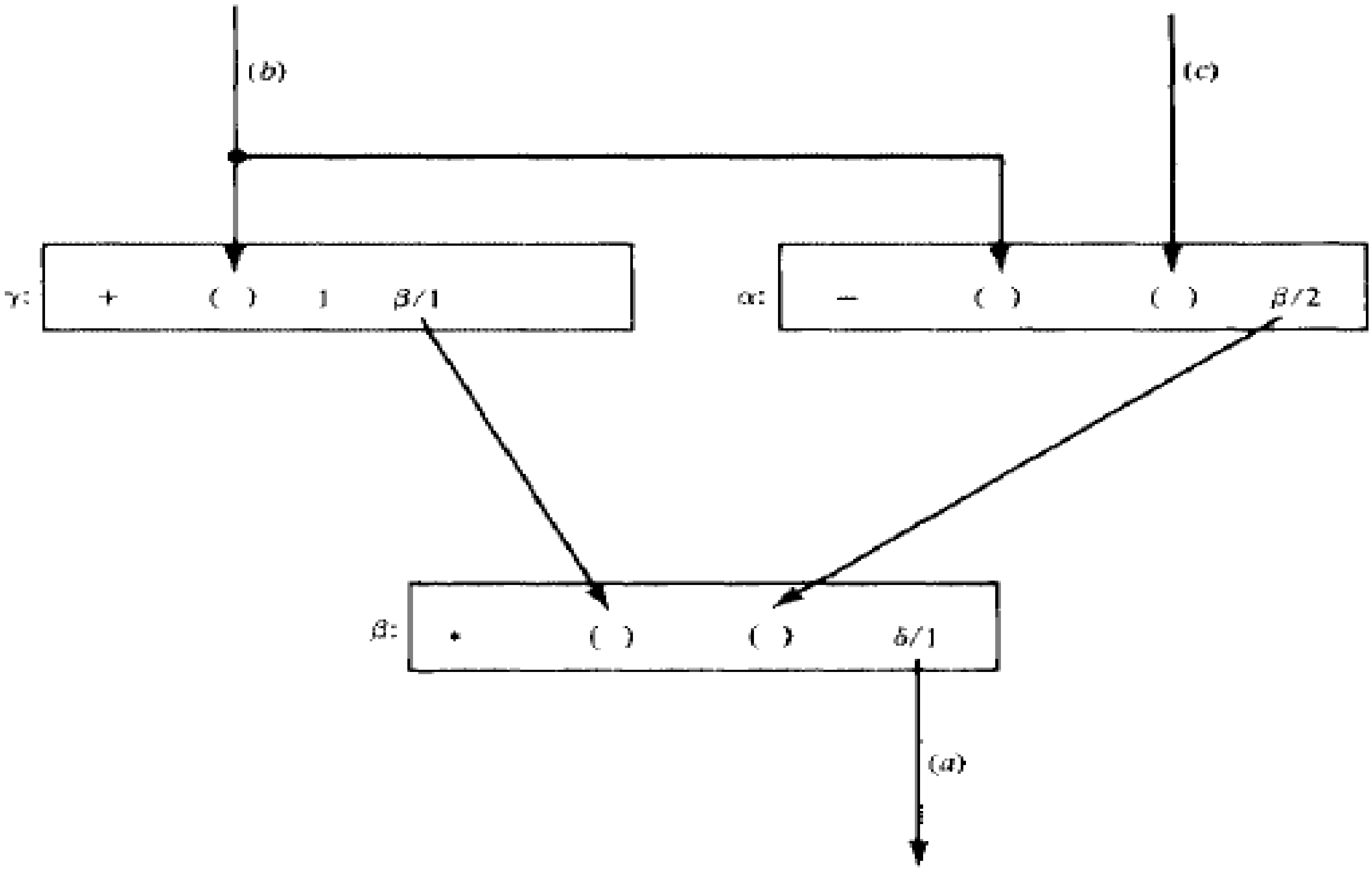
# Data Flow Execution Sequence shown by Data Flow Graph

Z= (x+y) * 2



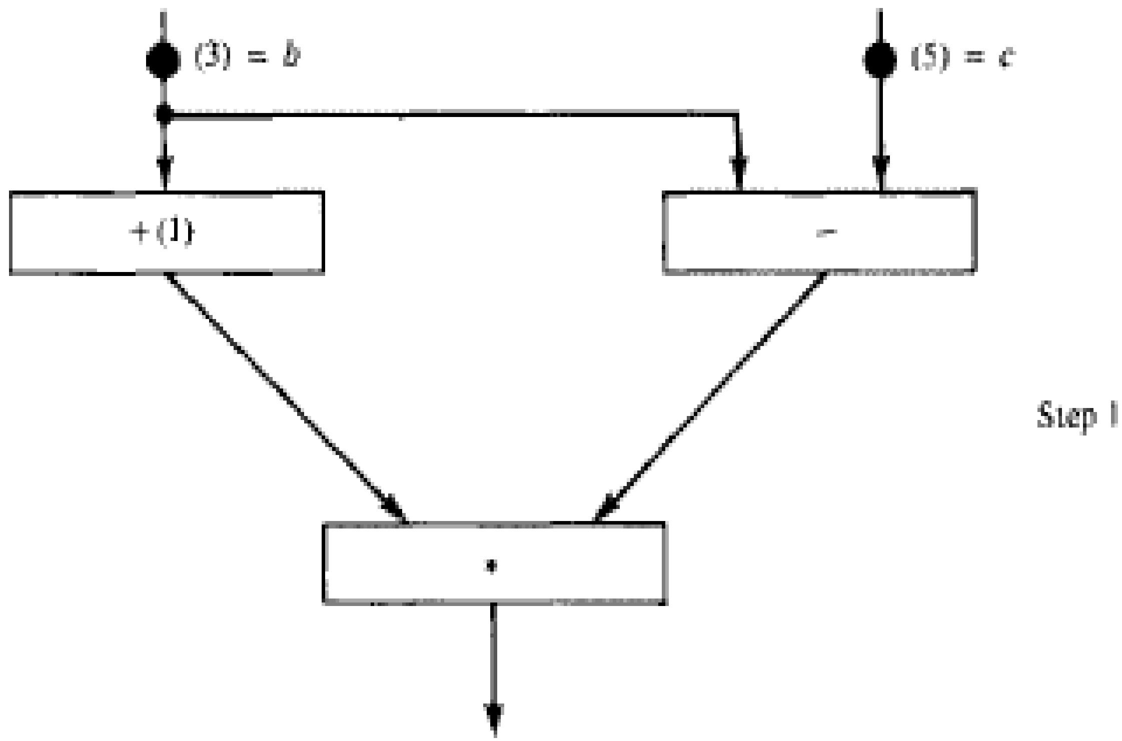
# Data Flow Computers

$$a = (b + 1) * (b - c).$$

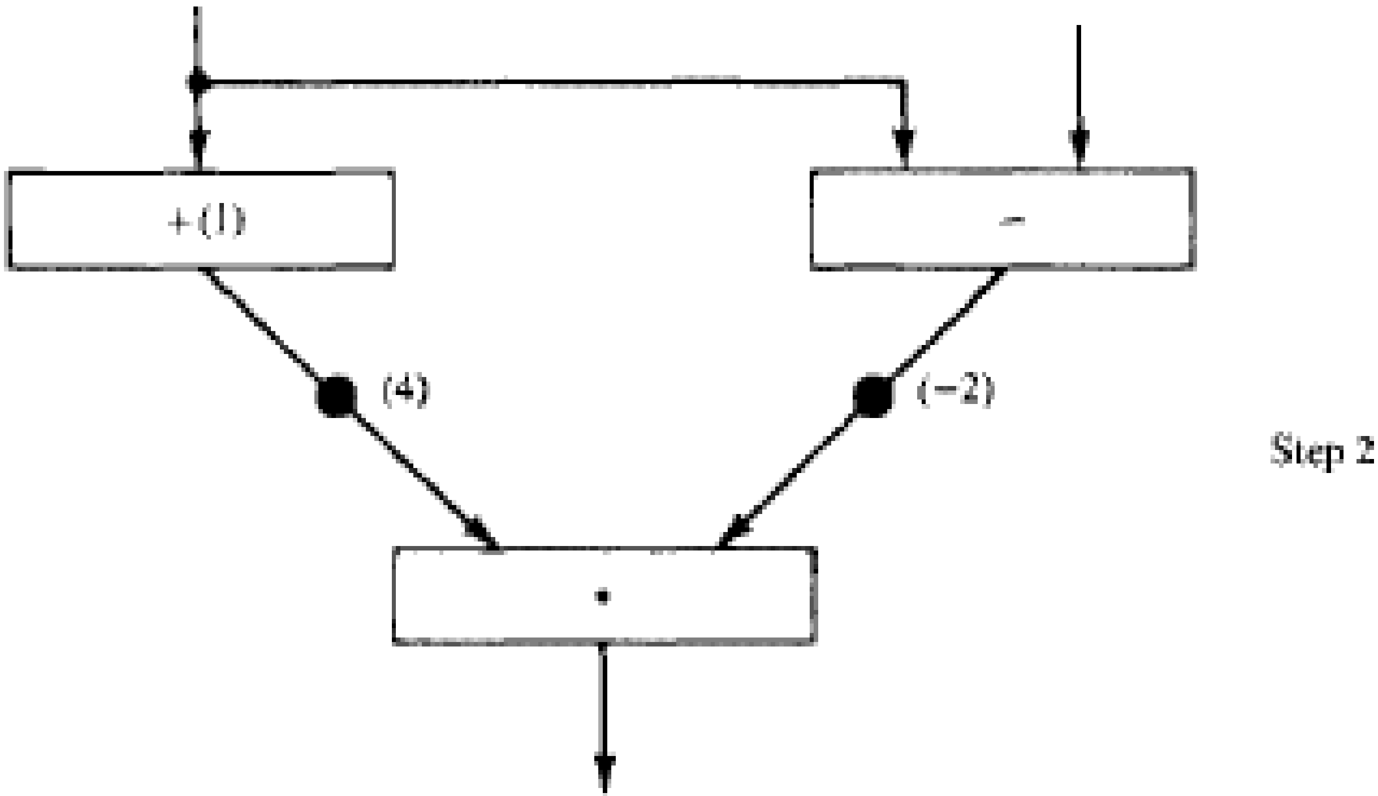


# Data Flow Computation

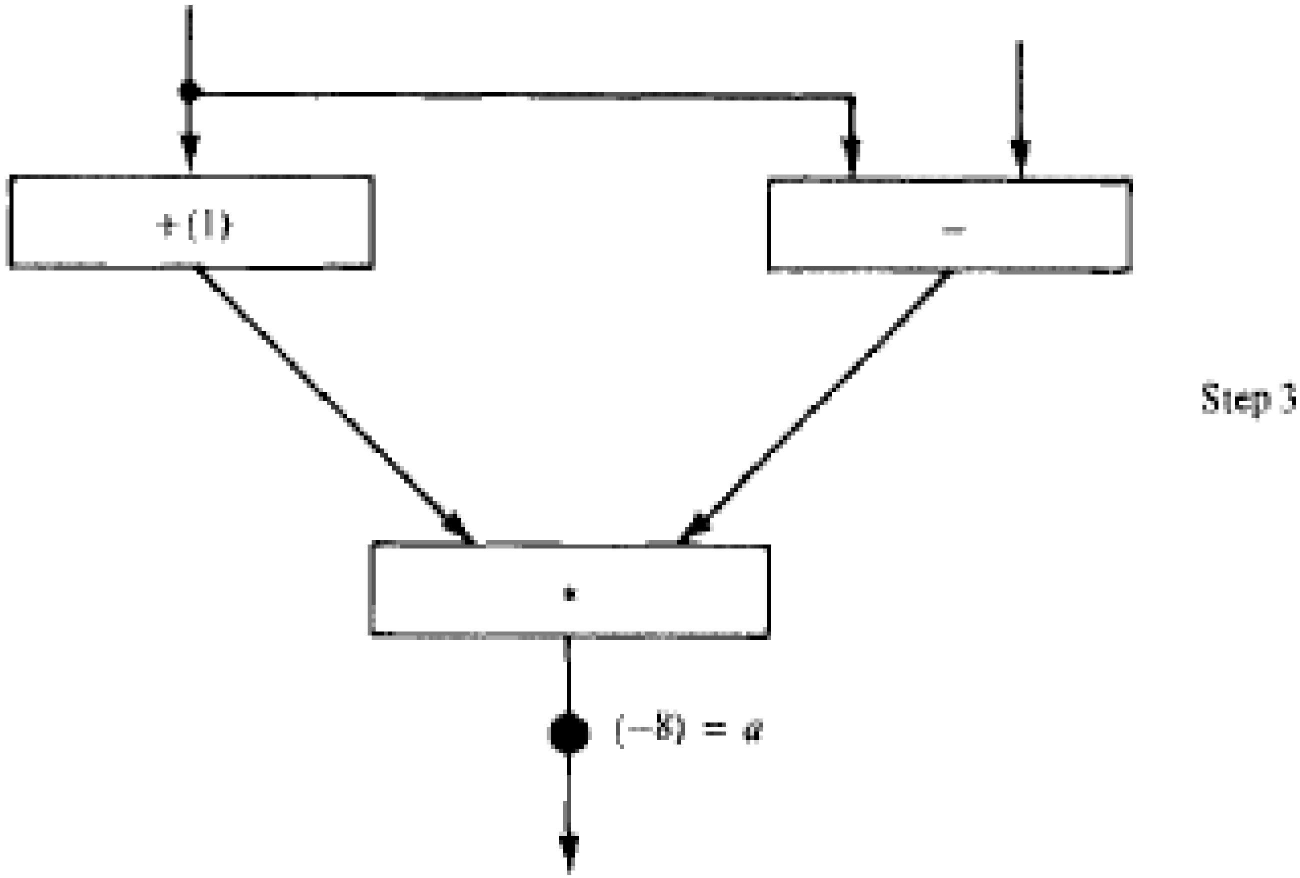
$$a = (b + 1) * (b - c).$$



# Data Flow Computation



# Data Flow Computation



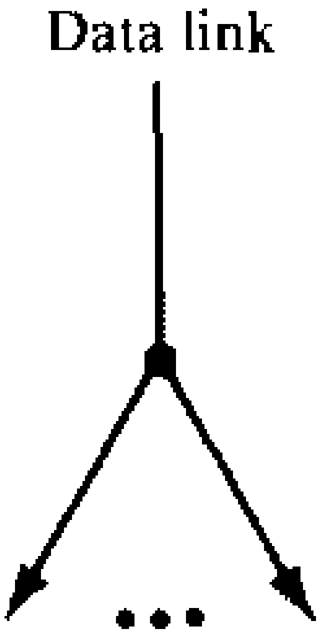
# To Execute a Program in Data Flow Computer

- Program is analyzed to find the dependencies among data in the program
- Program analysis can be represented by data flow graphs

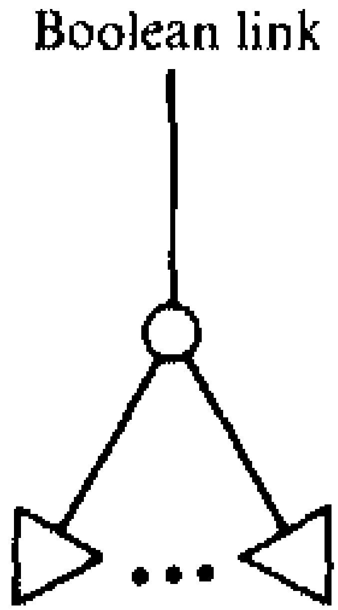
# What is Data Flow Graph?

- A directed graph whose nodes correspond to operators and arcs are pointers for forwarding data tokens
- The graph demonstrates sequencing constraints (data dependencies) among instructions

# Data Flow Graph Symbols

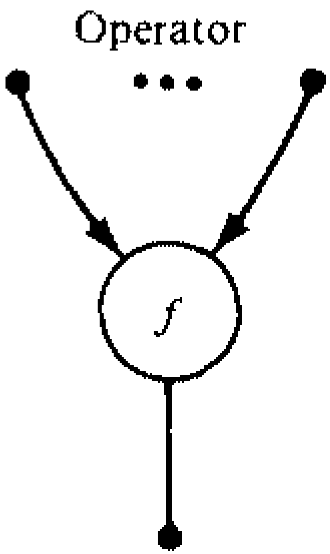



Transmit integer , real , complex numbers

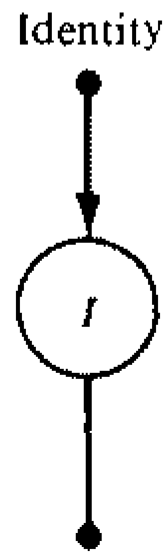



Carries only Boolean value for control purposes

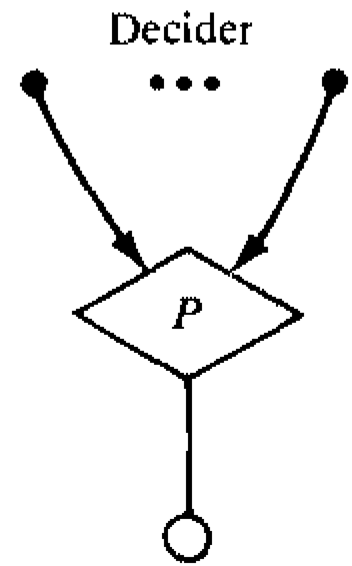
# Data Flow Graph Symbols




Operation

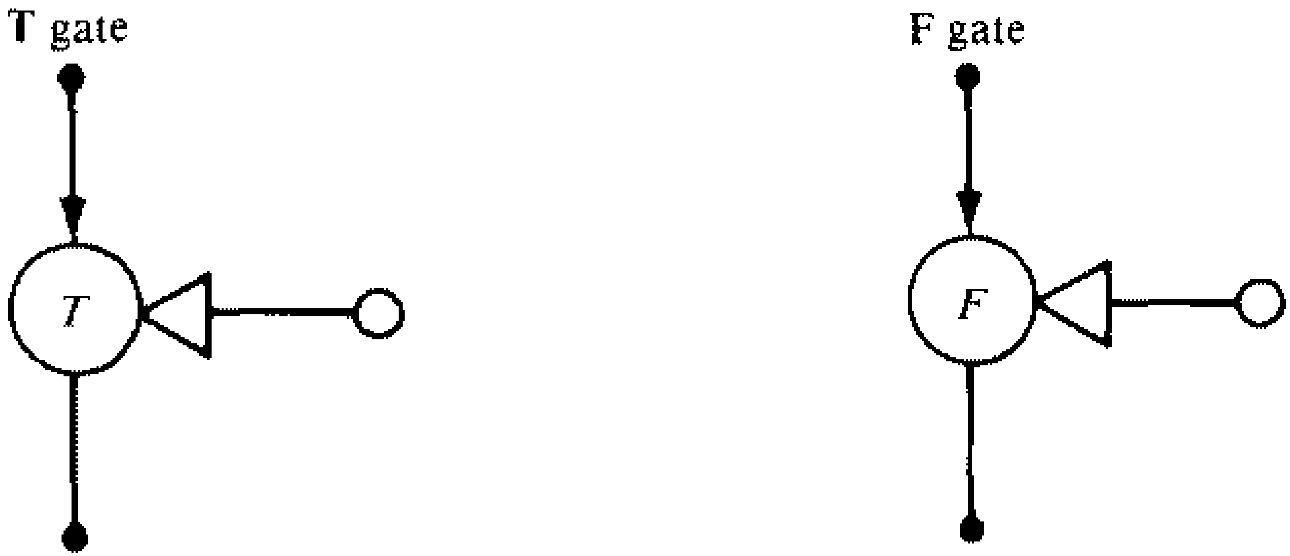



Value goes through




A value for each input arc produces the truth value applying the predicate P to values received

# Data Flow Graph Symbols

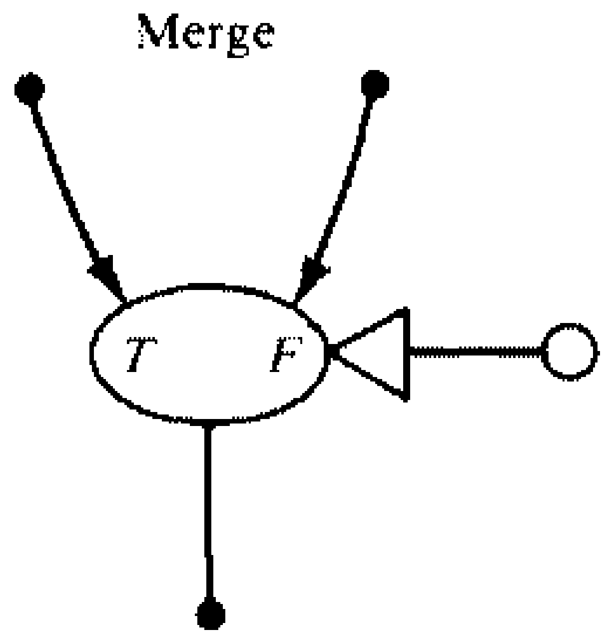



Used for conditional computation or iterative computation

T gate passes a data token from its input arc to its output arc if it receives a Control token with “true” Boolean value

F gate passes data token on if “false” control token is received

# Data Flow Graph Symbols

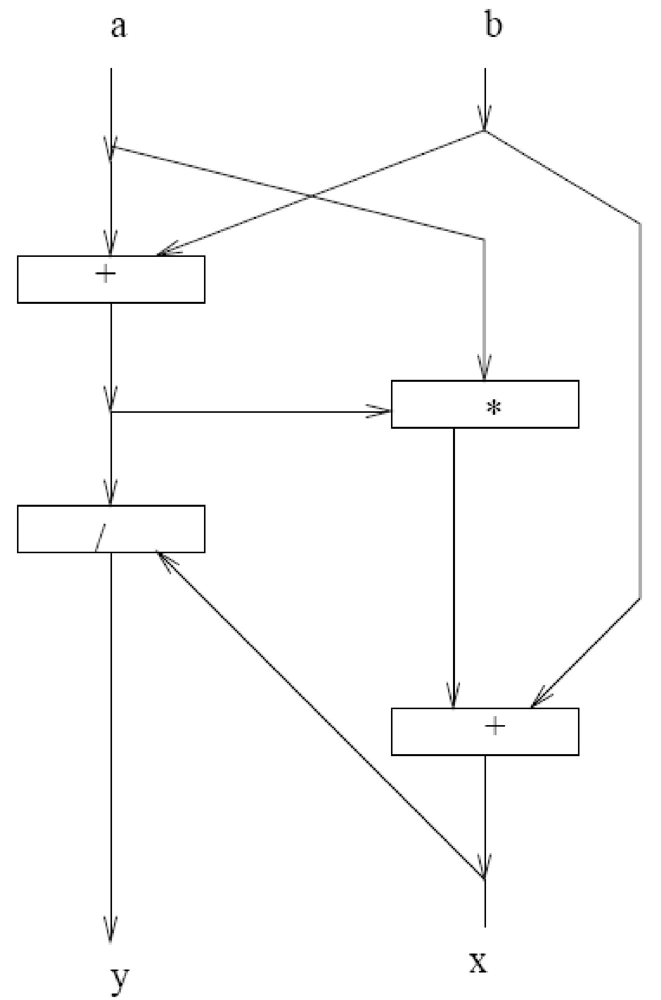



- When control value is true the input token of the true arc is transmitted
- When control value is false the input token of the false arc is transmitted

# Example:

- ```
  input a,b
      y := (a+b)/x
      x := (a*(a+b))+b
    output y,x
  ```

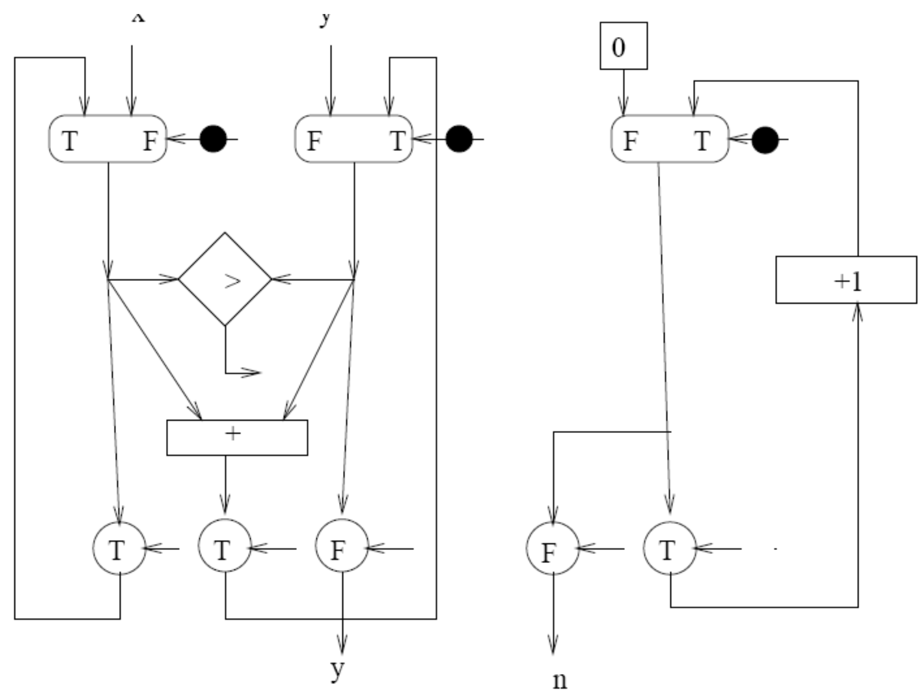
note ordering of statements in program is irrelevant


  ```

# Loop Example

```
input y,x
 n := 0
 while y<x do
   y :=  y + x
   n :=  n + 1
 end
 output y,n
```

***(Using arrays was intentionally avoided)***



# Assignment

- Represent by data flow graph using the symbols described in previous slides the sequence of following computations:

```
1) if  (x > 3)
           y=x + 2
   else
           y = x – 1
           y= y * 4
2) while (x > 0) do
           x=x-3
3) fact  = 1
   while ( n > 0){ fact = fact *n;
                   n= n – 1; } o/p fact
```

# Assignment contd…

Represent by data flow graph the sequence of following computations:
(Hwang and Briggs Softcopy page 759-760)

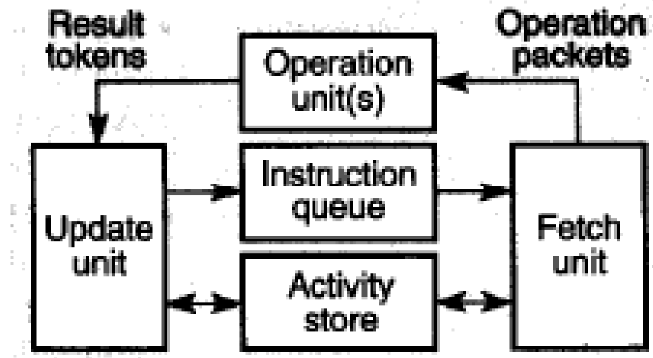
4)

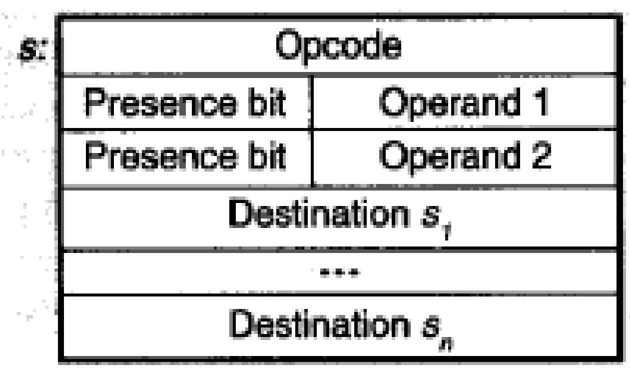
**Example 10.1**

1. $P = X + Y$ must wait for inputs X and Y
2. $Q = P \div Y$ must wait for instruction 1 to complete
3. $R = X \times P$ must wait for instruction 1 to complete
4. $S = R - Q$ must wait for instructions 2 and 3 to complete
5. $T = R \times P$ must wait for instruction 3 to complete
6. $U = S \div T$ must wait for instruction 4 and 5 to complete

# Static Dataflow

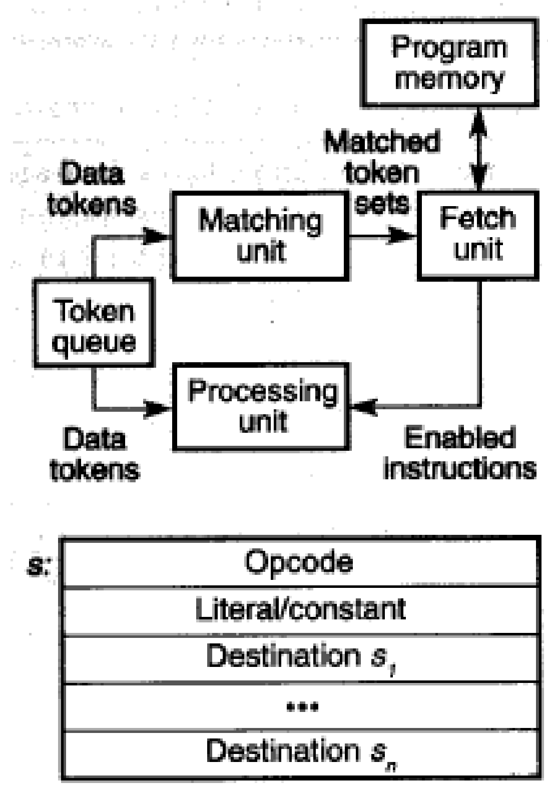
- Combine control and data into a template
  - like a reservation station
  - except they are held in memory
  - can inhibit parallelism among loop iterations
  - re-use of template ⇒ acks



# Dynamic Dataflow

- Separate data tokens and control
  - Token: labeled packet of information
- Allows multiple iterations to be simultaneously active
  - shared control (instruction)
  - separate data tokens
  - A data token can carry a loop iteration number
- Match tokens' tags in matching store via assoc. search
  - if match not found, make entry, wait for partner
- When there is a match, fetch corresponding instruction from program memory
- Requires large associative search
  - to match tags
- Adds "structure storage"
  - access via select function – index and structure descriptor as inputs



# Dataflow: Advantages/Disadvantages

- Advantages:
  - no program counter
    - data-driven
    - execution inhibited only by true data dependences
  - stateless / side-effect free
    - further enhances parallelism
- Disadvantages
  - no program counter
    - leads to very long fetch/execute latency
    - spatial locality in i-fetch hard to exploit
    - requires matching (e.g., via associative compares)
  - stateless / side-effect free
    - no shared data structures
    - no pointers into data structures (implies state)
    - In theory take entire data structure as input "token" and emit a new version
  - I/O difficult – depends on state
    - Virtual memory??

- A global version of Tomasulo's algorithm
  (Tomasulo's alg came first)