Oblikovanje programske potpore

2012./2013. grupa P01

Arhitektura protoka podataka

Prof.dr.sc. Vlado Sruk

Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva
Zavod za elektroniku, mikroel., računalne i inteligentne sustave

Literatura

- Jurij Šilc, Borut Robič, Theo Ungere: Processor Architecture, From Dataflow to Superscalar and Beyond, Springer 1999.

Arhitekture programske potpore

- Objektno usmjerena arhitektura
- Protok podataka
- Arhitektura zasnovana na događajima
- Arhitektura repozitorija podataka
- Slojevita arhitektura
- Virtualni strojevi
- Arhitektura programske potpore u upravljanju procesima

Opći modeli izračunavanja u programu

- Upravljački tok (*engl. control flow*)
- Programski jezici: VisualBasic, C, C++, JAVA, …
  - dominantno pitanje je kako se središte upravljanja pomiče kroz program ili sustav.
  - podaci mogu slijediti upravljački tok
    - taj slijed NE određuje arhitekturu sustava.
  - rasuđivanje u sustavu je primarno fokusirano na slijed izračunavanja.
- Protok podataka (*engl. data flow*)
  - dominatno pitanje je kako se podaci pomiču kroz kolekciju modula za izračunavanje.
  - s pomakom podataka aktivira se upravljanje (nije u fokusu interesa).

Arhitektura protoka podataka

- Arhitektura protoka podataka temelji se na mreži:
  - tj. skupu procesnih modula (automata - aktora) koji razmjenjuju podatke i paralelno (konkurentno) obavljaju obradu.

- Osnovna prednost ove arhitekture:
  - razdvajanje procesnih dijelova programa i
  - minimizacija dijelova programa koji se odnose na eksplicitno povezivanje varijabli, funkcija, modula, … .
  - 1970. i 1980. građene su i sklopovske arhitekture računala temeljene na protoku podataka.

Arhitektura protoka podataka

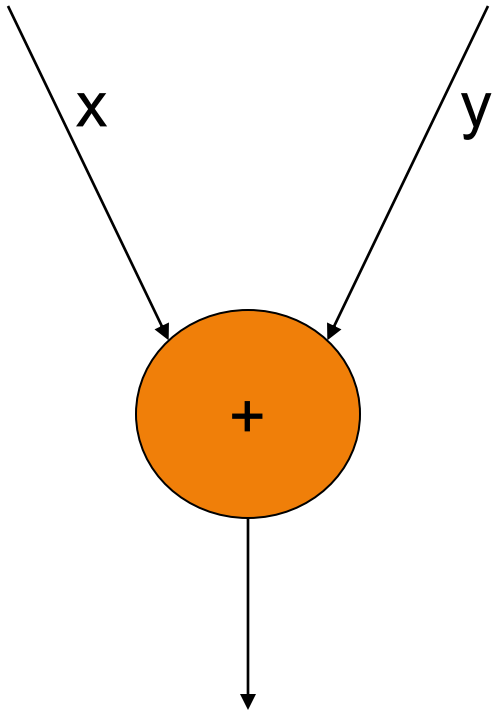
- *engl. data-flow*
- Komponenta
  - ima definirane ulaze i izlaze
  - čita podatke s ulaza i stvara niz podataka na izlazima
  - ulaze transformira lokalno i slijedno
  - radi neovisnosti često se naziva filter – ne poznaje ostale filtre
  - ako svi filtri u sustavu obrađuju sve podatke u jednom koraku - slijedno sekvencijski
- Konektor
  - medij za prijenos podataka – cjevovodi (*engl. pipes*)
  - specijalizacija:
    - pipeline (single sequence thru),
    - bounded (amount in pipe is restricted)
- Primjena: Unix, prvodioci, obradba signala, paralelni sustavi, distribuirani sustavi

Svojstva računala temeljenih na protoku podataka

- ***Programabilna*** *računala sa sklopovljem optimiziranim za podacima upravljanim izračunavanjem*

- ***Fina podjela*** (*engl. fine grain*)
  - na nivou instrukcija
- ***Upravljani podacima*** (*engl. data-driven*)
  - paralelizam uvjetovan samo raspoloživošću podataka
  - programi predstavljeni grafovima

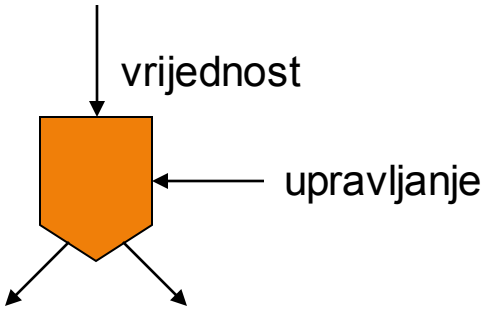
Terminologija



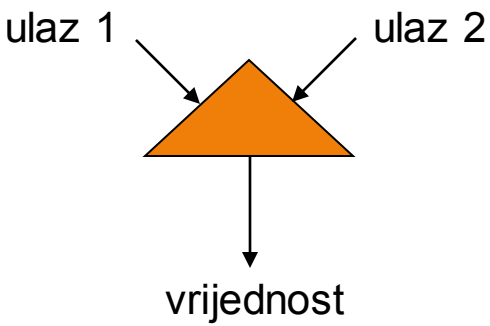
- Aktori (FUs)
- Podaci (engl. tokens/data)
- Lukovi (engl. dependences)
- Ulazi
- Omogućavanje (engl. enabling)
- Okidanje (engl. firing)
- Izlaz podataka (prisutni svi ulazni podaci + okidanje)

Tipovi aktora

- Funkcijski:
  - . +, -, *,…
- Uvjetni:

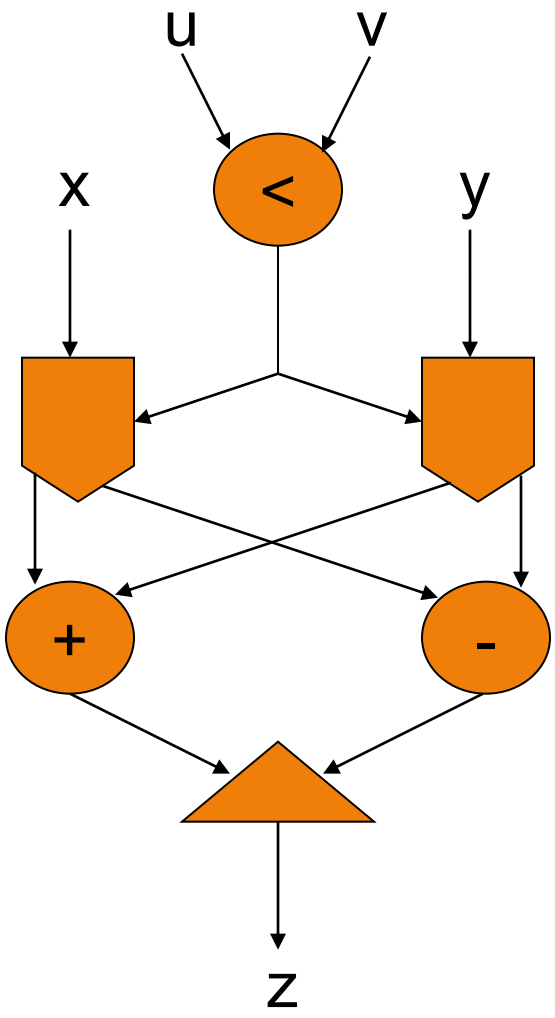


- Spajanje:



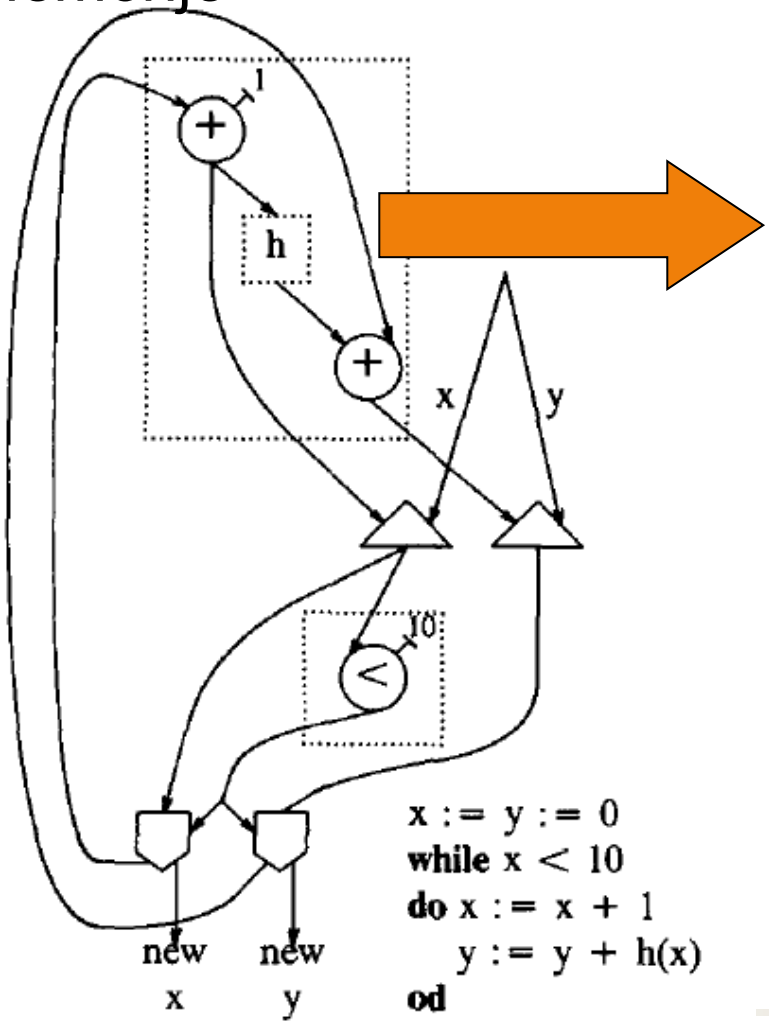
If ... then ... else ...

```
if(u < v)
  z = x + y;
else
  z = x - y;
```



Iteracije, rekurzije, ...

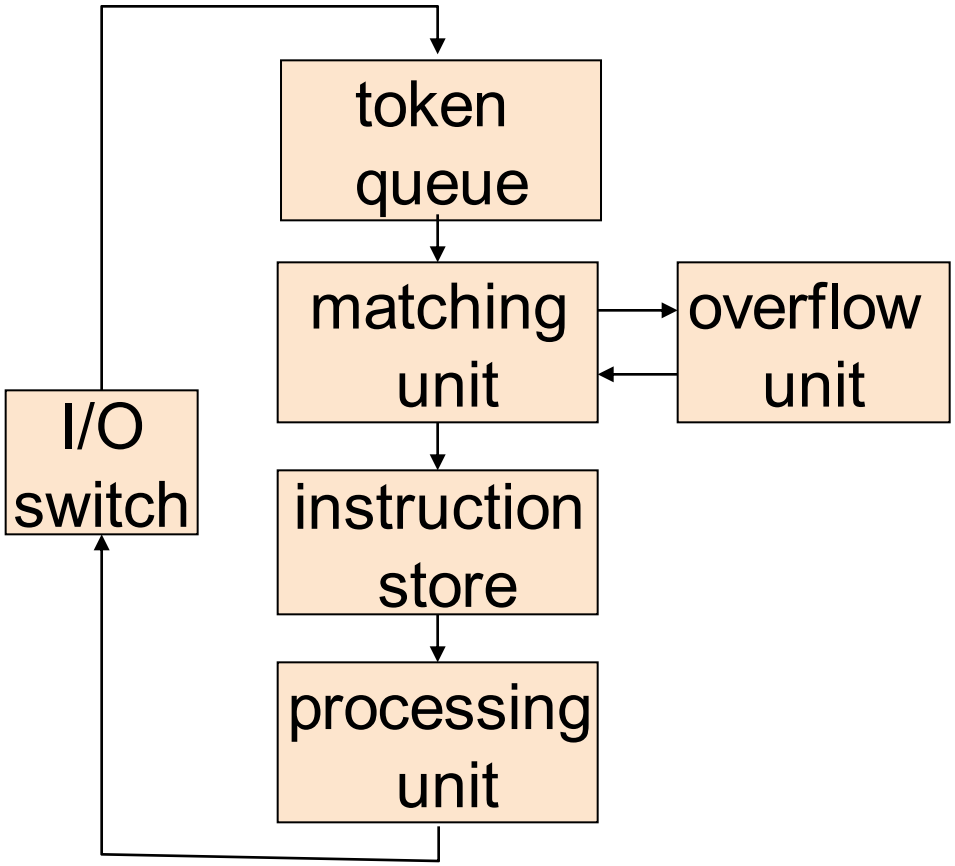
- ***Jedan graf*** – različiti podaci
- Nema memorije



t	x	y
0	***m***	***m***
1	***<,c***	***c***
2	***c,c***	***c***
3	***+1***	***+***
4	***h,m***	***+***
5	<,c	+
6	c,c	+
7	+1	+
8	h,***m***	+

Manchester Dataflow Machine

- 1976.-198. – J. R. **GURD**, I. **WATSON**



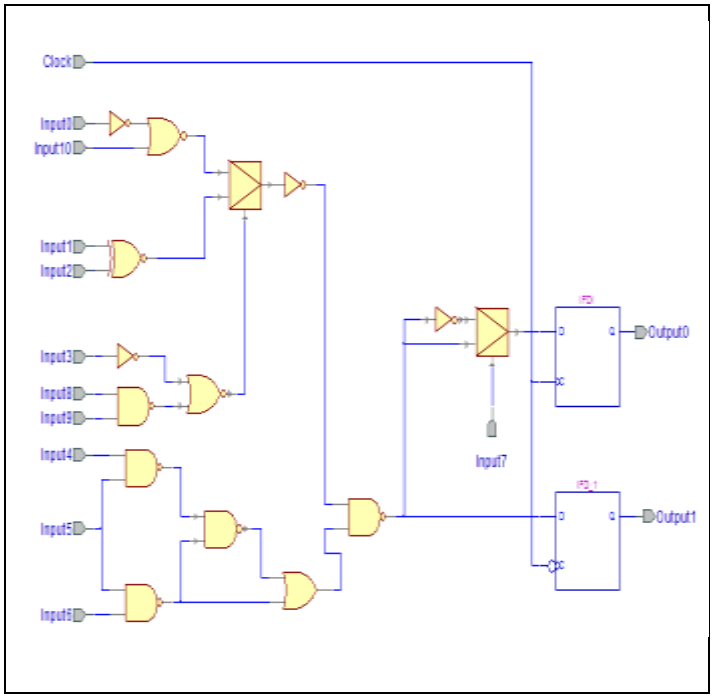
Trend

- Prednosti:
  - izražavanje paralelizma
  - tolerancija kašnjenja
  - mehanizmi fine sinkronizacije
- Nedostaci:
  - gubitak lokalnosti (ispreplitanje instrukcija)
  - rasipanje resursa
  - zauzeće prostora
- Konvergencija podacima upravljanim arhitekturama s Von-Neumann arhitekturom

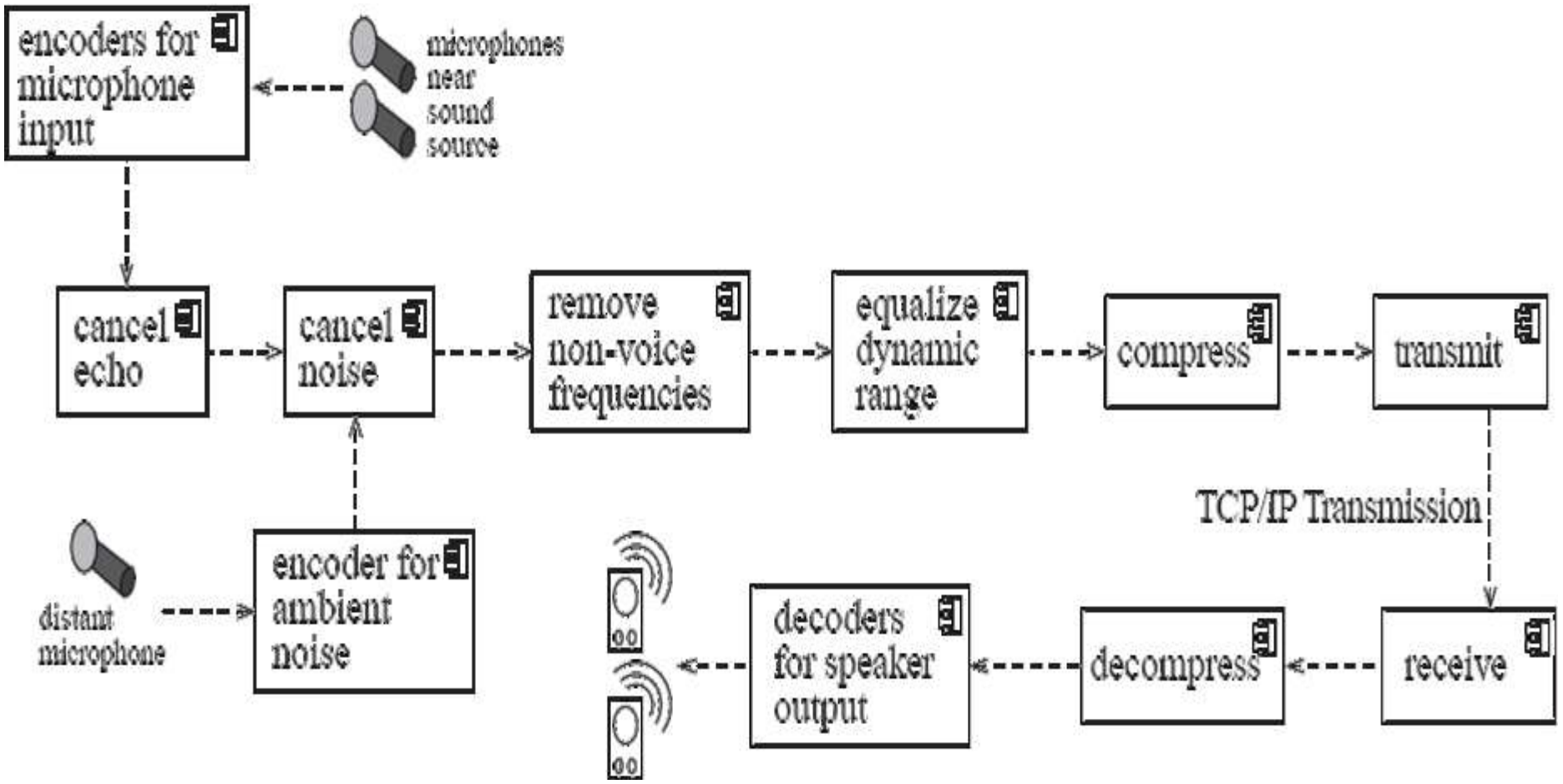
Primjer: VHDL

- VHDL

```
architecture MLU_DATAFLOW  of MLU is

signal A1:STD_LOGIC;
signal B1:STD_LOGIC;
signal Y1:STD_LOGIC;
signal MUX_0,  MUX_1,  MUX_2,  MUX_3:  STD_LOGIC;

begin
                A1<=A when  (NEG_A='0')  else
                                not A;
                B1<=B when  (NEG_B='0')  else
                                not B;
                Y<=Y1 when  (NEG_Y='0')  else
                                not Y1;

                MUX_0<=A1  and B1;
                MUX_1<=A1  or B1;
                MUX_2<=A1  xor B1;
                MUX_3<=A1  xnor B1;

                with (L1 & L0) select
                                Y1<=MUX_0  when "00",
                                                MUX_1  when "01",
                                                MUX_2  when "10",
                                                MUX_3  when others;

end MLU_DATAFLOW;
```



Primjer arhitekture protoka podataka



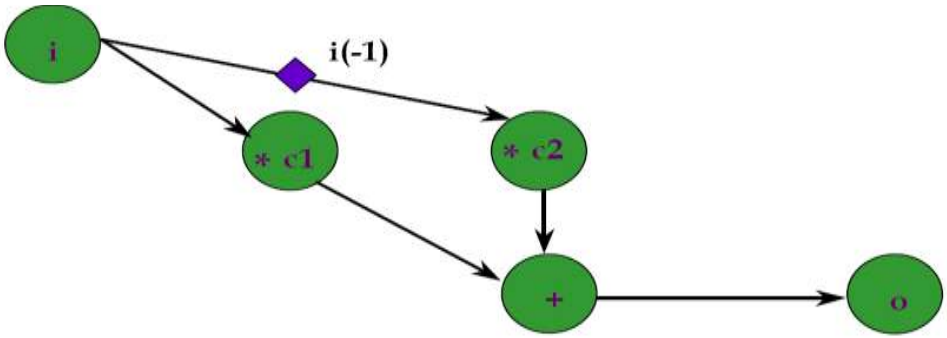
Primjer arhitekture protoka podataka

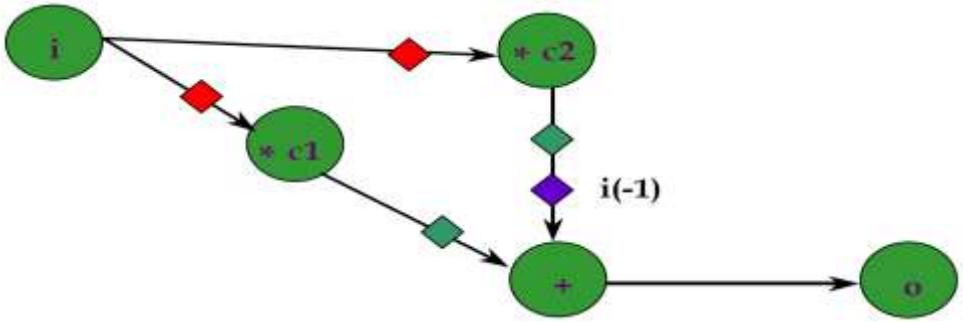
- sustav s konačnim impulsnim odzivom (engl. Finite Impulse Response - FIR)

▲single input sequence i(n)

▲single output sequence o(n)

▲o(n) = c1 i(n) + c2 i(n-1)





PROTOK PODATAKA

- Kahn: Procesne mreže (1958).
- Karp: Izračunski grafovi (1966).
- Denis: Mreže protoka podataka (1975).

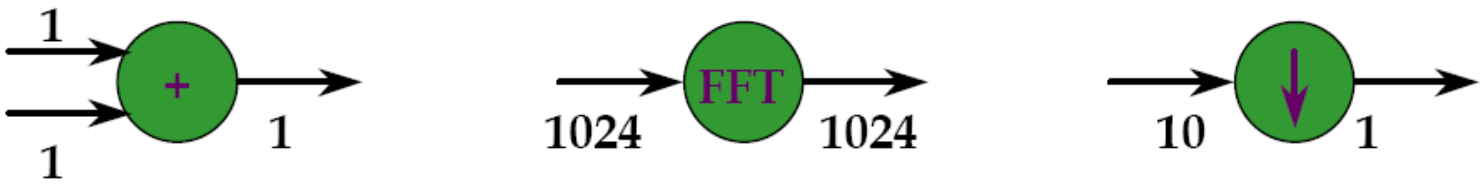
- Neke moderne implementacije:
- Grafičke:
  - LabView (Nat. Instr.), Ptolomy (UCB), SPW (Cadence), …
- Tekstualne:
  - Silage (UCB), Lucid, **Haskell**, …

Semantika

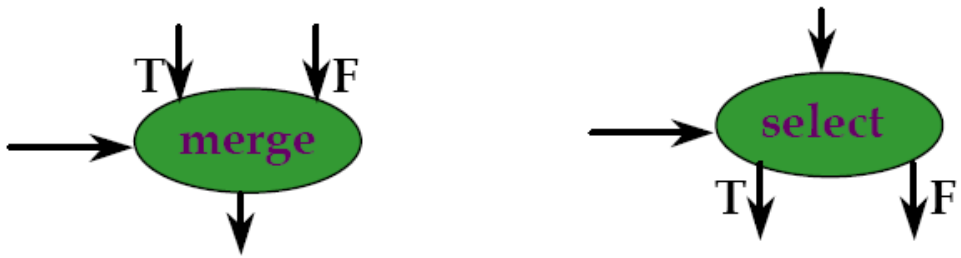
- Aktori - procesni elementi
  - izvode izračunavanje
  - ***često bez stanja***
- Neograničeni FIFO (first-in-first-out) međuspremnik sudjeluje u komunikaciji preko sekvencija znački (*engl, tokens*) koje mogu predstavljati:
    - cijele brojeve
    - realne brojeve
    - matrice
    - skupove slikovnih elemenata (piksela)
    - . . .
- Određenost:
  - na jedinstvene ulazne sekvence generiraju se jedinstvene izlazne sekvence.

Primjeri aktora

- SDF: Synchronous (or, better, Static) Data Flow
  - fixed input and output tokens



- BDF: Boolean Data Flow
  - control token determines consumed and produced tokens



Obilježja sustava

- Temeljenog na protoku podataka:
  - raspoloživost podataka upravlja izračunavanjem.
  - struktura arhitekture je određena redoslijedom pomicanja podataka od komponente do komponente (tj. od aktora do aktora).
  - oblik strukture je eksplicitan.
  - prijenos podataka je jedini način komunikacije između komponenata u sustavu.
- Varijacije glavne ideje (implementacijske razlike) :
  - izvedba upravljanja aktorima (npr. “pull” ili “push”).
  - stupanj paralelizma između procesa.
  - topologija mreže.

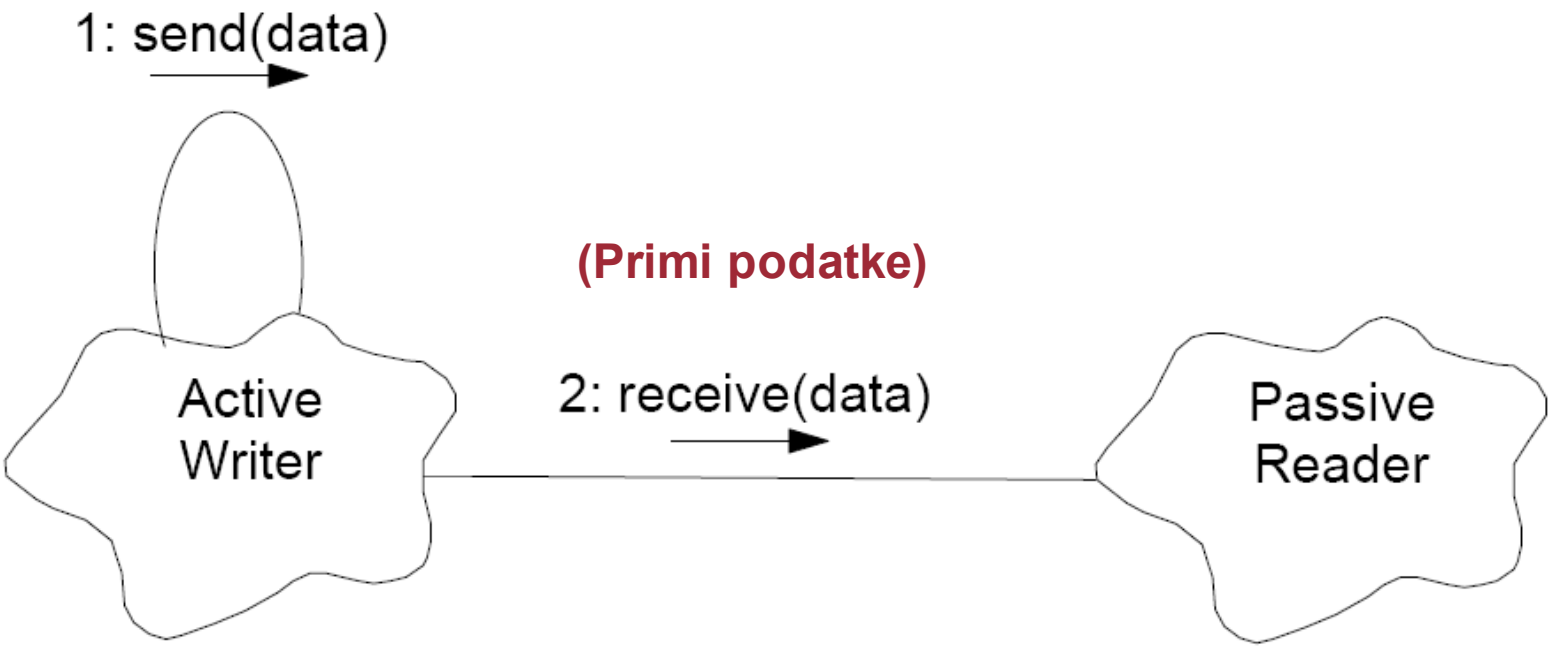
Mehanizmi upravljanja protokom podataka

- Guranje (*engl. push*)
  - izvor podataka gura podatke od sebe.
- Povlačenje (*engl. pull*)
  - ponor podataka uvlači podatke.
- Guranje/povlačenje
  - filtar aktivno povlači podatke iz niza, obavlja obradu i gura podatke dalje.
- Pasivni mehanizam
  - ne čini ništa, podaci tonu.

- Kombinacije mogu biti složene.
  - ako više filtara gura/povlači, potrebna je sinkronizacija.

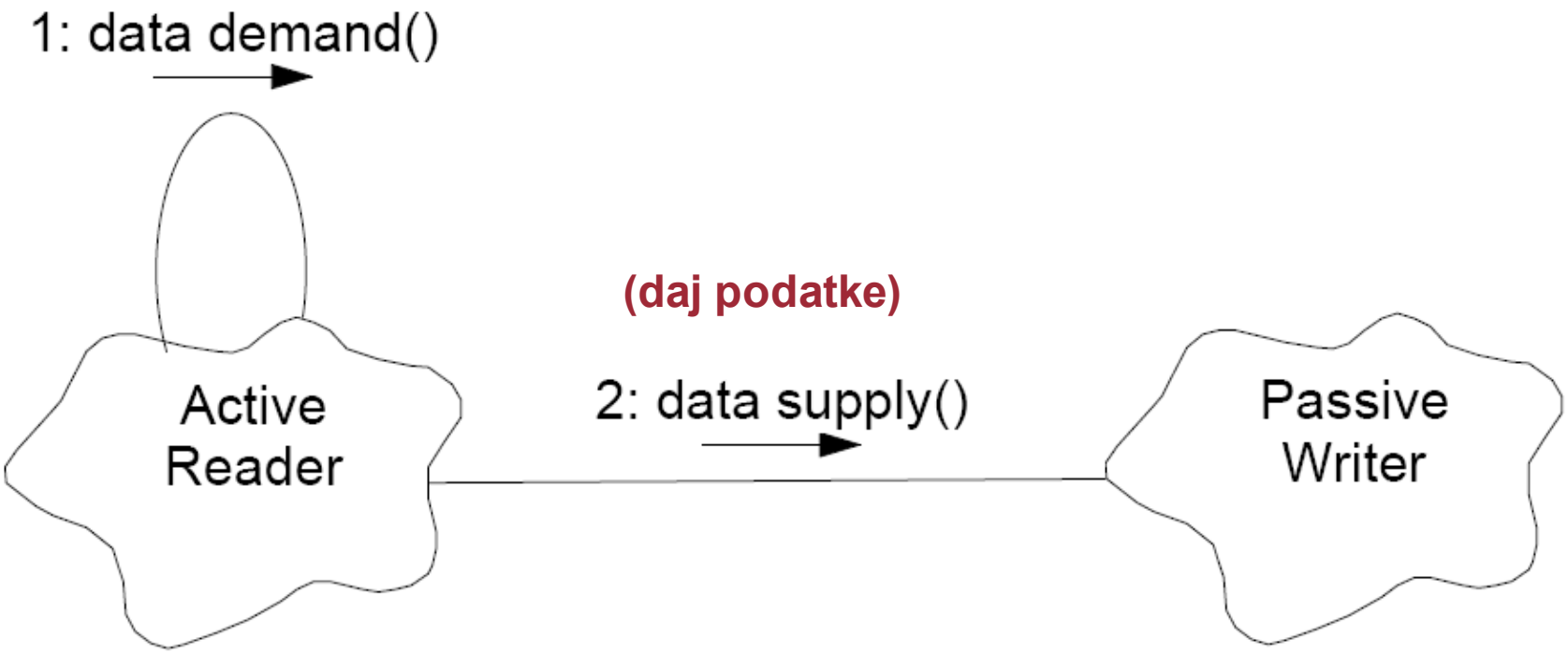
Guranje (*engl. push flow*)

- Aktivni pisatelj (*engl. writer*) inicira prijenos podataka pozivom primajuće procedure receive(data) pasivnog čitatelja.
  - Podaci se prenose kao parametar procedure.
  - Upravljački tok i tok podataka u postupku guranja su u istom smjeru.



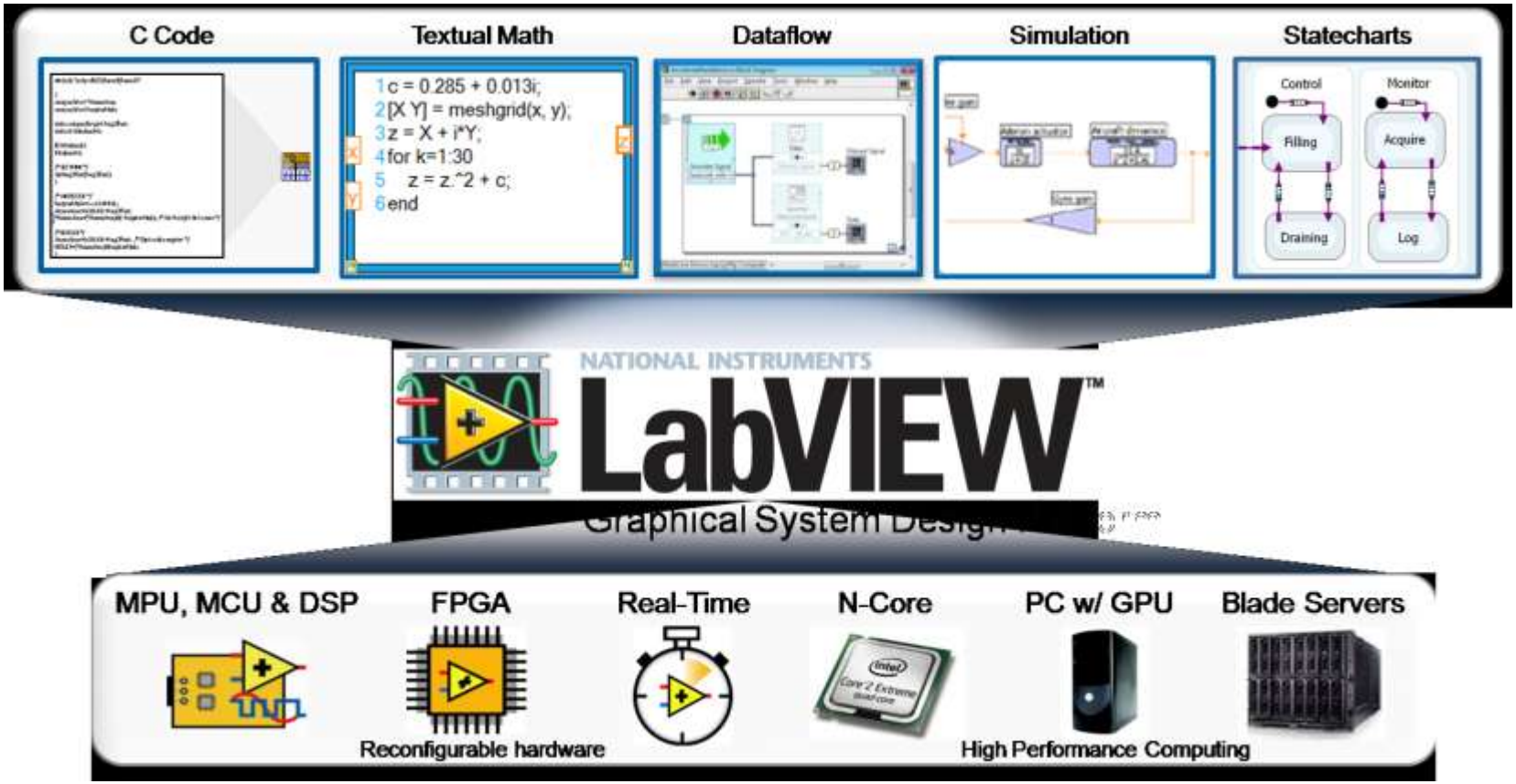
Povlačenje (*engl. pull flow*)

- Aktivni čitatelj (*engl. reader*) inicira prijenos pozivom procedure data_supply() pasivnom pisatelju.
  - Podaci se prenose kao povratna vrijednost procedure (metode).



Primjer

- Primjer arhitekture sustava temeljenog na protoku podataka
- LabVIEW http://www.ni.com/labview/

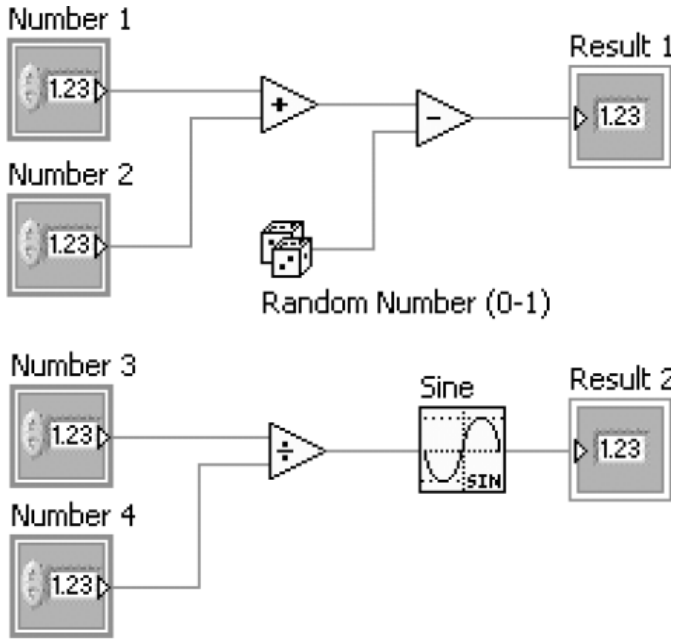


Koncept LabVIEW-a

- protok podataka – svaka komponenta izvršava zadatak kada su svi ulazni uvjeti zadovoljeni.
  - To implicira paralelizam ili pseudo-paralelizam.

dependent on the flow of data; block diagram does NOT execute left to right

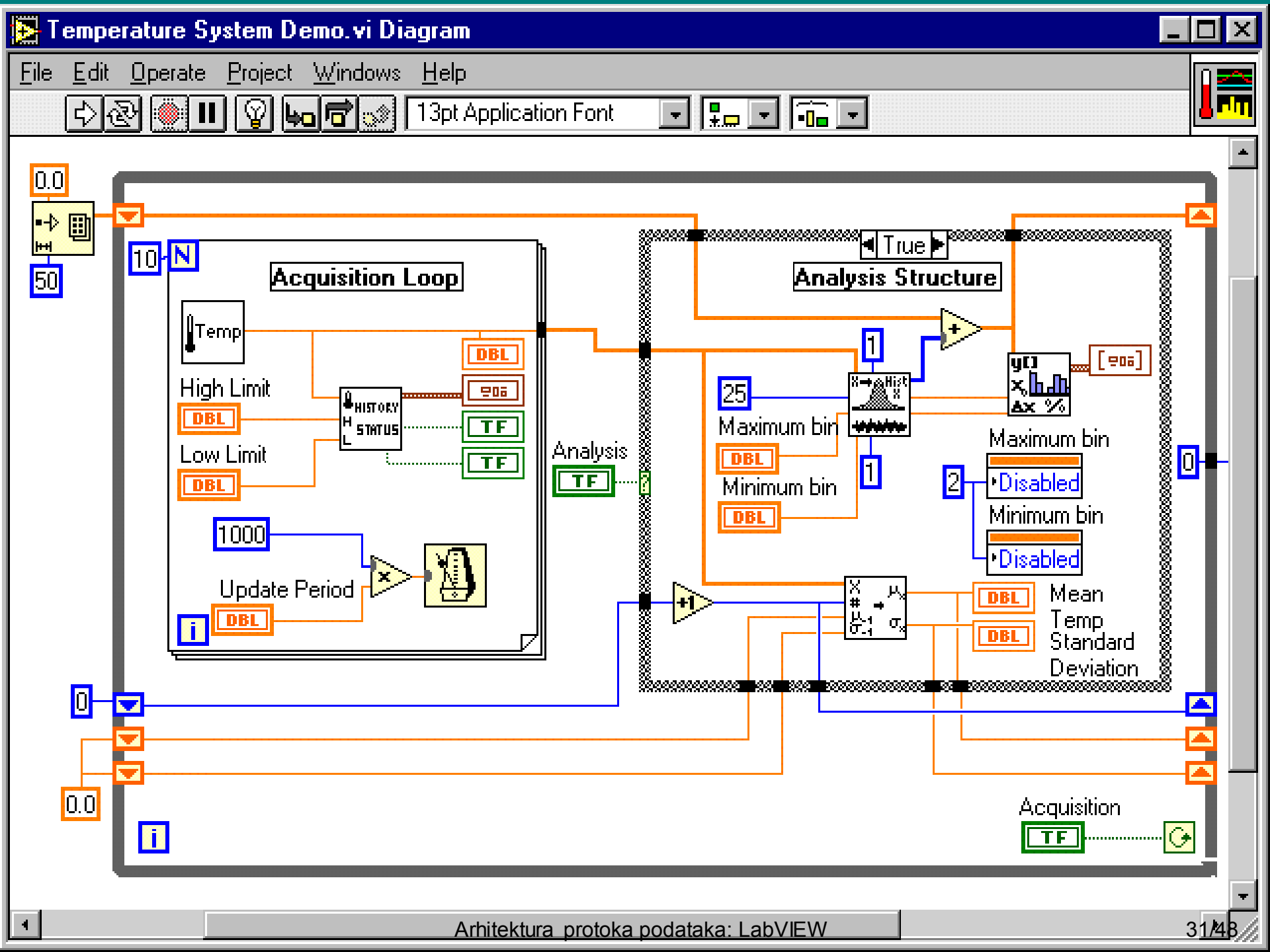
- Node executes when data is available to ALL input terminals
- Nodes supply data to all output terminals when done

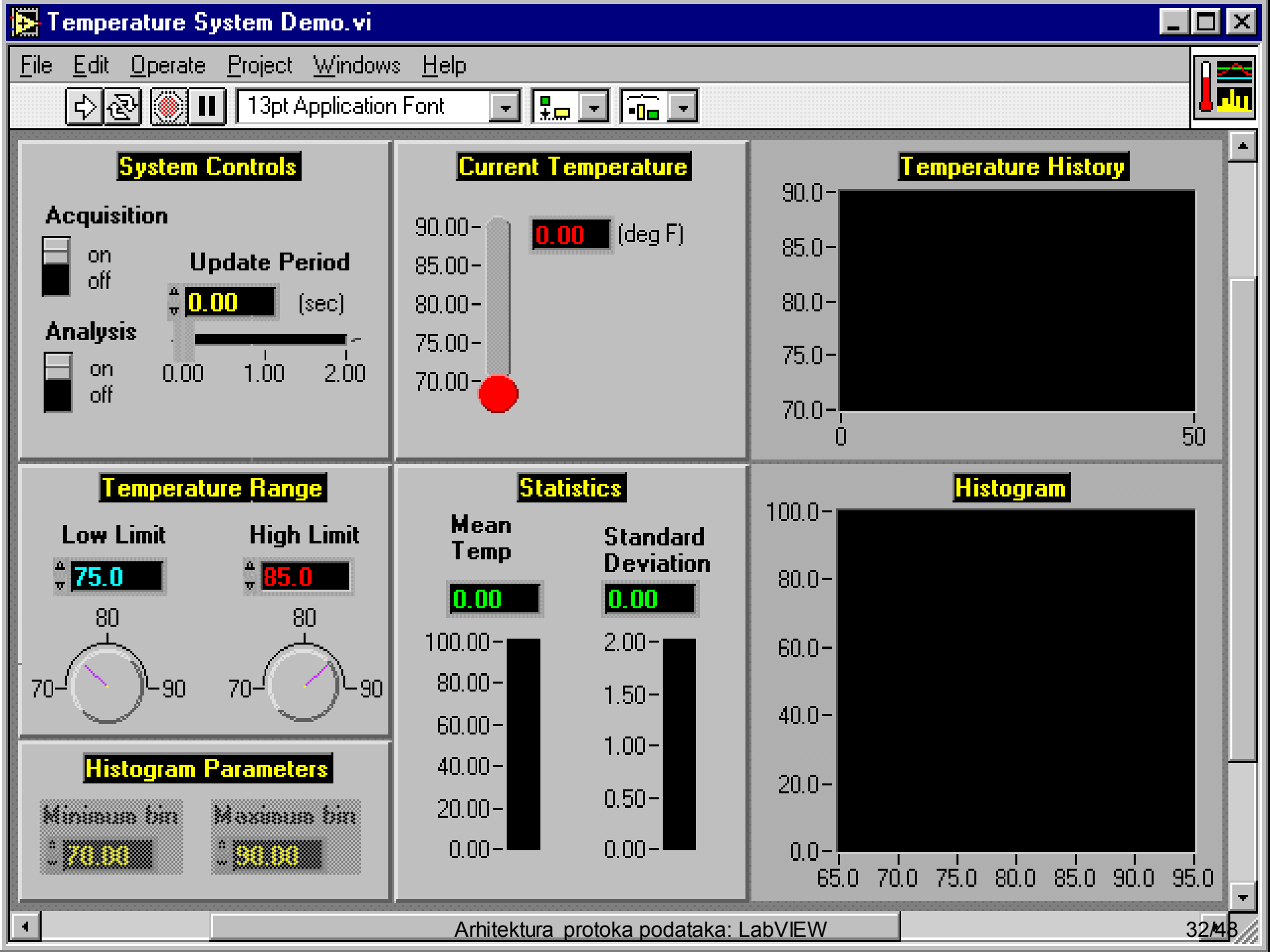


Programski jezik G

- Temeljna komponenta LabView sustava je prevoditelj za G dataflow programming language.
  - zasnovan na paradigmi protoka podataka, dobro prihvaćen zbog intuitivne grafičke reprezentacije i definirane sintakse.
  - G je homogen, dinamičan i višedimenzijski:

- Homogen - G aktori proizvode i konzumiraju pojedinačne značke na svakom luku grafa.
- Dinamičan - G sadrži konstrukcije koje dopuštaju da se dijelovi grafa izvode uvjetno, ovisno o ulaznim podacima.
- Višedimenzijski - G podržava višedimenzijske podatke (nizove, polja i sl.). Konstruktori petlje u G jeziku mogu kombinirati pojedinačne značke u nizove znački, ili razdvajati nizove u pojedinačne značke.


Temperature System Demo.vi Diagram
File Edit Operate Project Windows Help
13pt Application Font
0.0
50
10
N
Acquisition Loop
Temp
High Limit
DBL
Low Limit
DBL
HISTORY
H
L
STATUS
DBL
TF
TF
1000
Update Period
DBL
i
Analysis
TF
True
Analysis Structure
1
25
Maximum bin
DBL
Minimum bin
DBL
1
+1
y[]
Maximum bin
2
Disabled
Minimum bin
Disabled
0
DBL
DBL
Mean
Temp
Standard
Deviation
0
0.0
i
Acquisition
TF

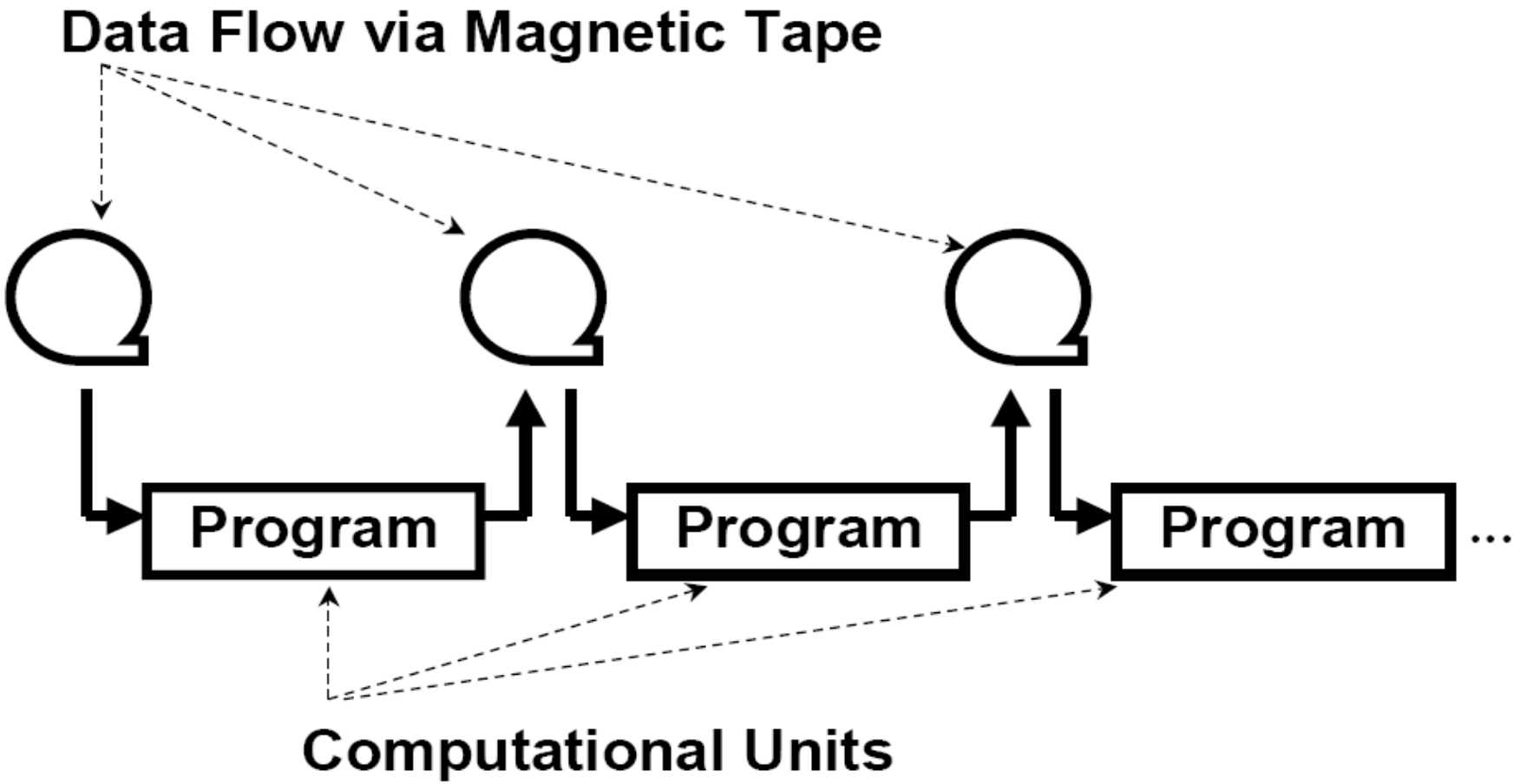

Temperature System Demo.vi
File Edit Operate Project Windows Help
13pt Application Font
System Controls
Acquisition
on
off
Update Period
0.00
(sec)
Analysis
on
off
0.00
1.00
2.00
Current Temperature
90.00
85.00
80.00
75.00
70.00
0.00
(deg F)
Temperature History
90.0
85.0
80.0
75.0
70.0
0
50
Temperature Range
Low Limit
75.0
High Limit
85.0
80
70
90
80
70
90
Statistics
Mean Temp
0.00
Standard Deviation
0.00
100.00
80.00
60.00
40.00
20.00
0.00
2.00
1.50
1.00
0.50
0.00
Histogram
100.0
80.0
60.0
40.0
20.0
0.0
65.0 70.0 75.0 80.0 85.0 90.0 95.0
Histogram Parameters
Minimum bin
70.00
Maximum bin
90.00

Stilovi arhitekture protoka podataka

- ***Skupno-sekvencijski*** (*engl. batch – sequential*)
- ***Cjevovodi i filtri*** (*engl. pipes and filters*)
  - …

Skupno sekvencijski stil

- engl. Batch sequential pattern



Obilježja skupnog sekvencijskog sustava:

- Svaki procesni korak je nezavisan program.
- Svaki procesni korak (program) izvodi se do potpunog završetka, prije prijelaza na slijedeći korak.
- Podaci se između koraka prenose u cijelosti.

- Tipične primjene:
  - poslovne: Diskretne transakcije unaprijed određenog tipa i koje se pojavljuju u periodičnim intervalima, periodični ispisi i dopune, obrada koja nije pod strogim vremenskim ograničenjima, skupna obrada (npr: obračun plaća).
  - transformacijska analiza podataka: Sakupljanje i analiza sirovih podataka u koračnom i skupnom modu (npr. crna kutija u avionu).

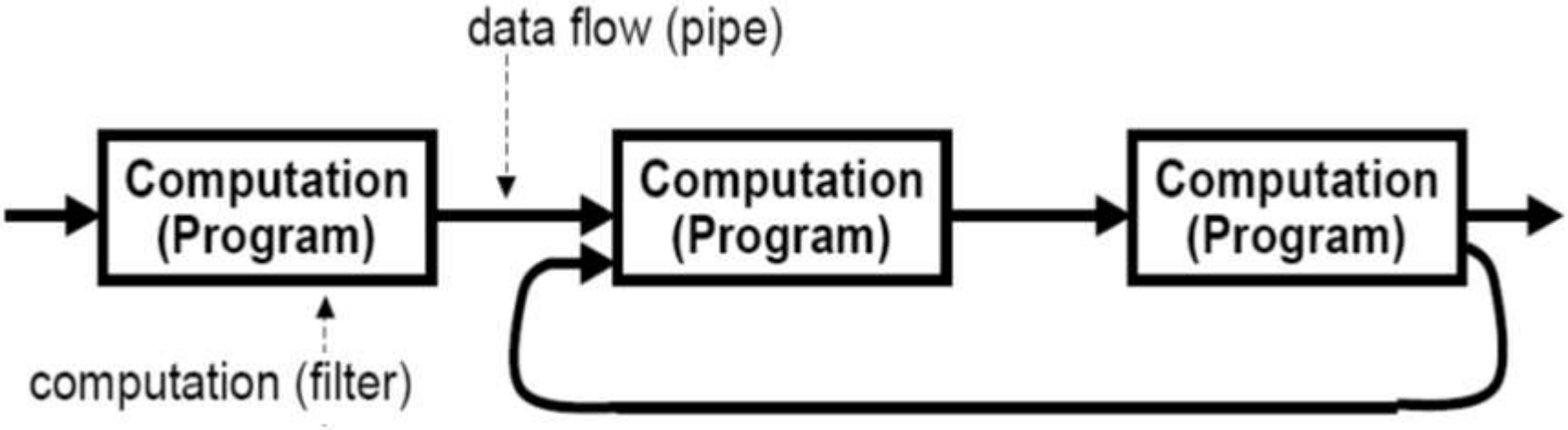
Primjeri skupne sekvencijske arhitekture

- Kompilatori (prevoditelji)
  - početno mehanizam preslikavanja izvornog koda više razine u objektni kod.
  - rani kompilatori su nastali kao skupni sekvencijski sustavi (leksička analiza, parsiranje, semantička analiza, generiranje koda).

- Computer Aided Software Engineering (CASE)
  - početno okolina za pisanje i kompiliranje.
  - rani alati su bili nezavisni programi.
  - kasniji alati međusobno dijele samo datoteke.
  - moderni alati uključuju oblikovanje, dokumentaciju, upravljanje konfiguracijama i inačicama te generiranje koda.

Cjevovodi i filtri

- *engl. pipes-and-filters*
- alternativa skupno sekvecijskom sustavu kod potrebe kontinuirane obrade
- Magnetska vrpca u skupnom sekvencijskom sustavu poprima oblik jezika i konstruktora u operacijskom sustavu.



Obilježja i uloge (čistih) filtara

- Čita niz podataka (*engl. stream*) na ulazu i generira niz podataka na izlazu (tradicijski byte usmjereno).
- Transformira niz u niz
  - obogaćuje podatke dodajući nove informacije.
  - rafinira podatke izbacujući nerelevantne informacije.
  - transformira podatke promjenom njihovog značenja.
- Inkrementalno transformira podatke
  - podaci se obrađuju po dolasku (ne prvo sakupe pa obrade).
- Filtri su nezavisni entiteti
  - nema konteksta u obradi niza.
  - nema čuvanja stanja između pojedinih instanciranja sustava.
  - nema znanja o neposrednim susjedima (prethodnicima i sljedbenicima).
  - ispravnost izlaza filtra ne ovisi o redoslijedu povezivanja u mreži.

Obilježja (čistih) cjevovoda

- Prenose podatke od izlaza filtra do ulaza filtra (ulazno/izlazne naprave ili datoteke).
- Podaci se prenose u jednom smjeru. Dozvoljava se samo eventualno upravljanje u obrnutom smjeru.
- Cjevovodi formiraju graf prijenosa podataka.

- Rad sustava cjevovoda i filtera
  - akcija sustava je određena dobavom podataka.
  - cjevovodi i filtri obavljaju prijenos i obradu podataka nedeterministički dok ne nestane podataka ili dok nije više moguće izračunavanje (obrada).

Primjer cjevovoda i filtara: UNIX

- UNIX philosophy (original statements)
  - a program should ***do one thing well***.
  - complex problems should be solved by ***combining simple programs*** whenever possible.
  - write as little new code as ***possible leveraging*** existing code and utilities in order to solve a problem.
  - the internal software architecture is ***well documented*** and adheres to standards.

Primjer cjevovoda i filtara: UNIX

- UNIX procesi koji preslikavaju stdin u stdout (ili drugi izvori i ponori) nazivaju se filtrima.
- Često konzumiraju sve ulazne podatke prije generiranja izlaza. Time narušavaju osnovnu pretpostavku o filtrima (npr. sort, grep, awk).
- UNIX cjevovodi tretiraju datoteke (i druge entitete) kao filtre te kao izvore i ponore podataka. Datoteke (i mnogi drugi entiteti) su pasivni te nije moguće načiniti povezivanje po volji.
- UNIX pretpostavlja da cjevovodi prenose ASCII znakove.
  - povoljno: jer je moguće povezivanje bez ograničenja.
  - nedostatak: jer se sve mora kodirati u ASCII i dekodirati.

```
cat /etc/passwd | grep "joe" | sort > junk
```

Usporedba

- skupno sekvencijske arhitekture s cjevovodima i filtrima

Skupno sekvencijska	***Cjevovodi i filtri***
Gruba zrnatost	Fina zrnatost
Visoka latencija	Rezultati s početkom obrade
Vanjski pristup ulazima	Lokalizirani ulazi
Nema paralelizma	Moguć paralelizam
Nema interaktivnosti	Interaktivnost moguća (ali nespretno)

Razlozi izbora arhitekture protoka podataka:

- Zadatkom dominira dobavljivost podataka.
- Podaci se mogu prediktivno prenositi od procesa do procesa.
- Cjevovodi i filtri su dobar izbor u mnogim primjenama protoka podataka jer:
  - Dozvoljavaju ponovnu uporabu i rekonfiguracija filtara.
  - Rasuđivanje o cijelom sustavu je olakšano.
  - Reducira se ispitivanje (testiranje) sustava.
  - Može se ostvariti inkrementalna i paralelna obrada.
- Arhitektura protoka podataka može reducirati performanse sustava (npr. svođenjem na jedan tip podataka u cjevovodima - UNIX).

Evaluacija arhitekture protoka podataka

1. *Podijeli i vladaj (Divide and conquer)*:
   - odvojeni procesi mogu se oblikovati nezavisno.
2. Povećaj koheziju (*Increase cohesion)*:
   - procesi posjeduju funkcionalnu koheziju.
3. Smanji međuovisnost (*Reduce coupling)*:
   - procesi imaju mali broj ulaza i izlaza.
4. Povećaj apstrakciju (*Increase abstraction)*:
   - cjevovodne komponente su dobra apstrakcija jer skrivaju unutarnje detalje.
5. Oblikuj da se omogući ponovna uporabivost dijelova (*Increase reusability)*:
   - procesi se često koriste u mnogim različitim kontekstima.
6. Uporabi postojeće komponente:
   - često se mogu pronaći gotove komponente za uključenje u sustav.

Evaluacija arhitekture protoka podataka

7. Oblikuj za fleksibilnost (Design for flexibility):
   - postoji više različitih ostvarenja fleksibilnosti sustava. Komponente se mogu jednostavno izbacivati i nadomještati. Nekim komponentama se može mijenjati redoslijed u sustavu.
8. Anticipiraj zastaru (Anticipate obsolescence).
9. Oblikuj za prenosivost (Design for portability).
10. Oblikuj za jednostavno ispitivanje (Design for testability): Uobičajeno je jednostavnije ispitivanje pojedinačnih komponenti.
11. Oblikuj konzervativno (Design defensively): Rigorozno se provjeravaju ulazi svake komponente ili se mogu postaviti jasne preduvjeti i post-kondicije svake komponente.

Diskusija