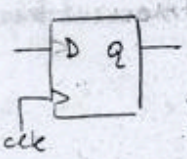
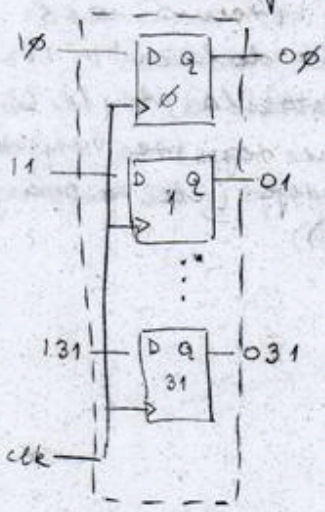
(1) Prijenos podataka između registara

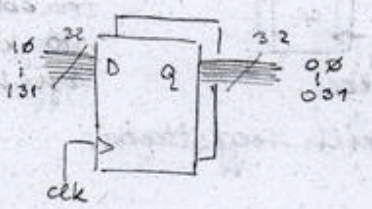
Najbolje je odabrati D bistabil za realizaciju registara.



1° Realizacija registra

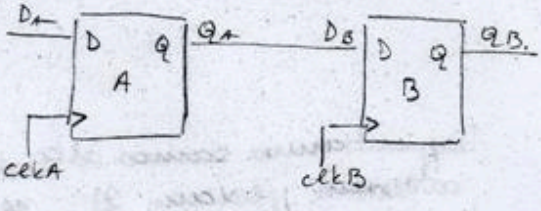


ili



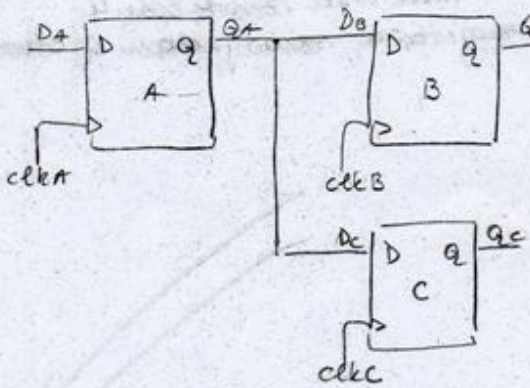
2° Povezivanje registara za prebacivanje podataka

1:1



⇒ sa clkB se upisuje iz A u B

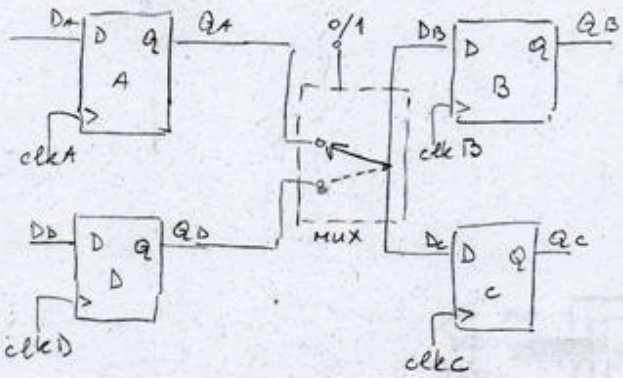
1:N



⇒ iz A u C → clkC
iz A u B → clkB
iz A u B i C → clkB + clkC

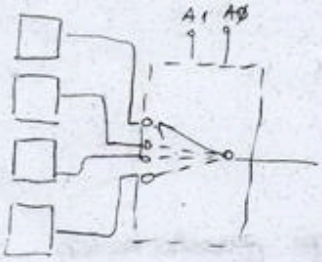
M : N

Kada prebacujemo podatke iz više mogućih registara na više mogućih odredišta (također registara), podaci mogu dolaziti samo iz jednog od registara u trenutku. Prvi način realizacije toga je MUX-om.

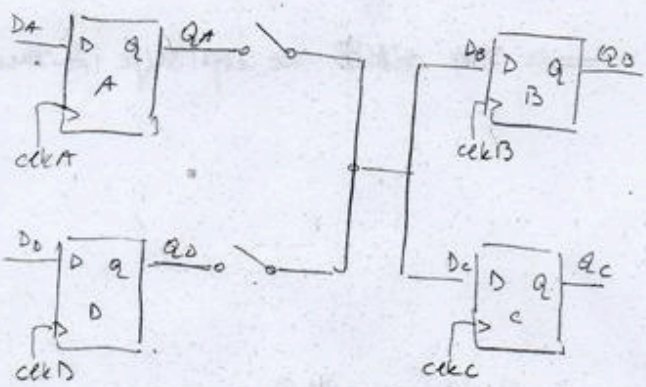


Postavljanjem MUX-a u 0 ili 1 biramo hoćemo li podatak dolaziti iz registra (bistabila) A ili D. Za odredišne registre vrijedi isto kao i prije (clk za onaj koji hoćemo)

Ako ima više izvorišnih registara:

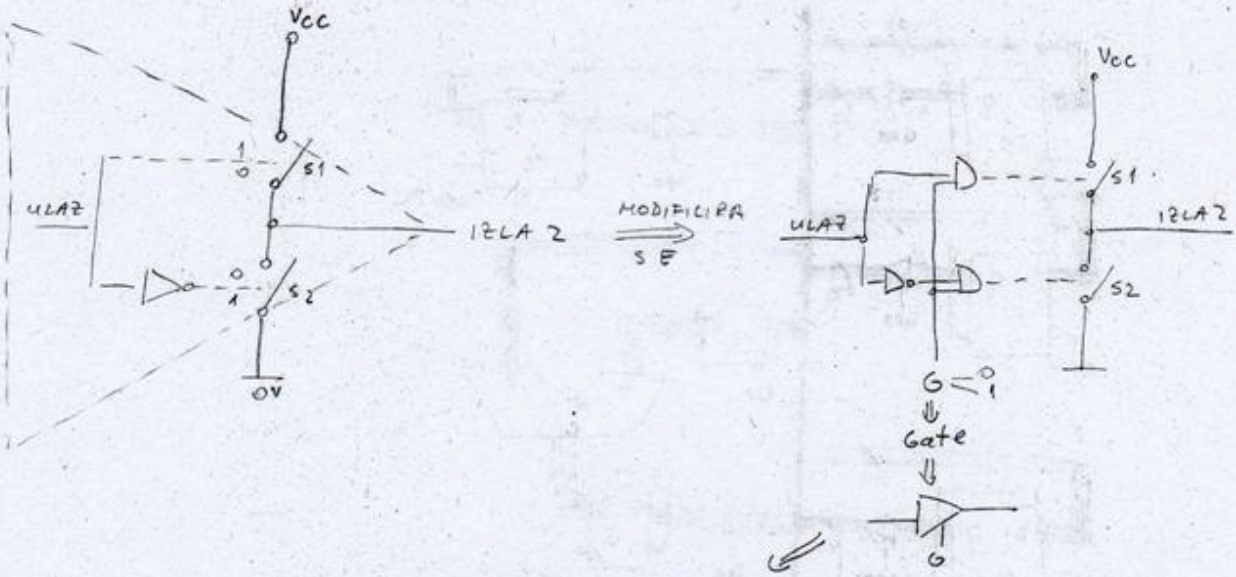


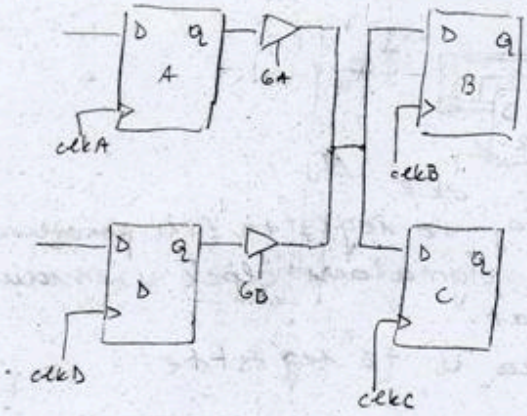
Kontroliramo izvorišne registre pomoću A0 i A1.



Dopuštamo samo da je otvoren jedan Q, na QA ili QD. Isto se primjenjuje i na više izvorišnih registara, samo jedan Q otvoren

2° Logika s tri stanja

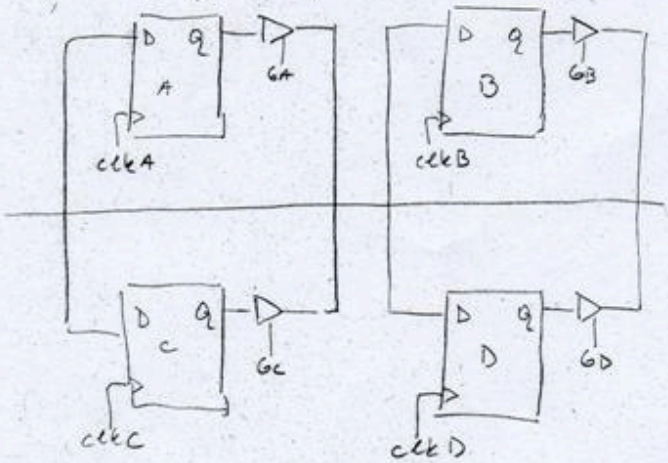




⇒ Sa G-ovima upravljamo:
- G = 0 → odspojeno
- G = 1 → spojeno

Samo jedan G smije biti 1 jedinici

⇓ → svaki registar može slati i primati podatke

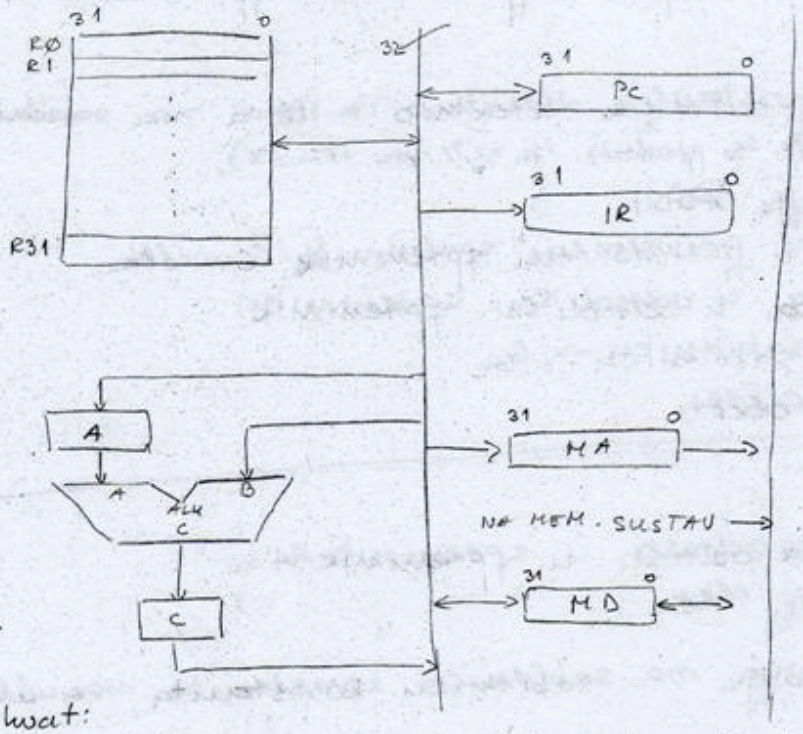


iz C upisati u A i D

$G_C = 1$ ($G_A, G_B, G_D = 0$)

clkA, clkD

③ Izvođenje naredbi na jednosabirničkom procesoru



Dohvat:

① Sadržaj PC-a na sabirnicu i upisati u MA
G_{PC} clk_{MA}

Iskoristiti priliku kada je sadržaj PC-a na sabirnici (na ulazu B u ALU), inkrementirati ga i spremiti u C
INC4 clkC

② Na adresnoj sabirnici je adresa lokacije na kojoj je naredba, postavi se signal READ, i učita naredba u MD. Sabirnica je slobodna, pa se preko nje inkrementirani sadržaj PC-a prebaci iz C u PC.
G_C, clk_{PC}

③ Naredbu iz MD prebaciti u IR
G_{MD}, clk_{IR}

Izvođenje:
ADD R2, R0, R1

④ Dohvat prvog operanda i upis u privremeni spremnik A
G_{R0}, clk_A

⑤ Dohvat drugog operanda i postavljanje na ulaz B
G_{R1}, Nakon toga se ALU zbrajanje ADD, rezultat se upiše u privremeni spremnik C → clk_C

⑥ Pohrana rezultata u odredišni spremnik
→ sadržaj C na sabirnicu → G_C
→ upis u R2 → clk_{R2}

ADDI R31, R1, 10

(4.) Dohvat 1. operanda i upis u privremeni spremnik A
G_{R1}, clk_A

(5.) Drugi operand se postavlja direktno iz IR-a na sabirnicu na ulaz B (konstanta se proširi 16 bit na 17...31)
Naredi se zbrajanje ADDI
Rezultat se upiše u privremeni spremnik C - clk_C

(6.) Pohrana rezultata u odredišni spremnik
→ sadržaj C na sabirnicu → G_C
→ upis u R_{31} → clk_{R31}

LOAD R31, R1, 10

(4.) Upisuje se bazna adresa u spremnik A
G_{R1}, clk_A

(5.) Iz IR-a se postavlja na sabirnicu konstanta, na ulaz B (16 → 17..31)
Naredi se zbrajanje.
Rezultat se upiše u privremeni spremnik C - clk_C

(6.) Upisujemo sadržaj spremnika C u MA, naredimo READ
G_C, clk_{MA}

(7.) Upisujemo podatak iz memorije u MD

(8.) Podatak iz MD-a upisujemo u R31
G_{MD}, clk_{R31}

STORE R31, R1, 10

(4.) Upisuje se bazna adresa u spremnik A
G_{R1}, clk_A

(5.) Iz IR-a se postavlja na sabirnicu konstanta, na ulaz B (16 → 17..31)
Naredi se zbrajanje, rezultat se upiše u privremeni spremnik C. clk_C

(6.) Upisujemo sadržaj iz C-a u MA. G_C, clk_{MA}

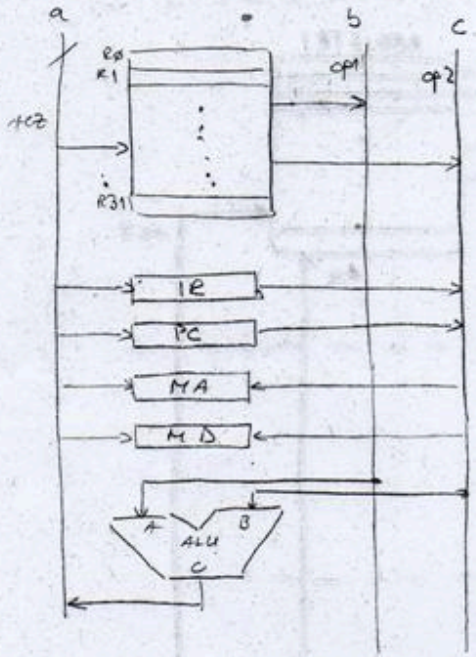
(7.) Upisujemo iz R31 u MD. G_{R31}, clk_{MD}

(8.) WRITE

(4) Trosabirnički procesor

Cilj nam je učiniti procesor brži.
a) brži sklopovima
b) manje koraka

Kako imamo samo jednu sabirnicu, možemo izvoditi samo po jedan korak. Zato koristimo 3 sabirnice, za op1, op2, rezultat. Time dohvat op1, dohvat op2+AL oper + i spremanje rezultata radimo u jednom koraku.



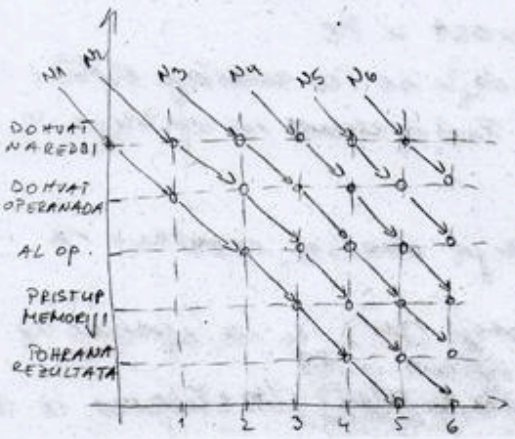
Ovom arhitekturom smanjen je ukupan broj koraka na svega tri, te su eliminirani spremnici A i C.

Prednost: moguće je istovremeno prenositi više podataka

Nedostatak: cijena

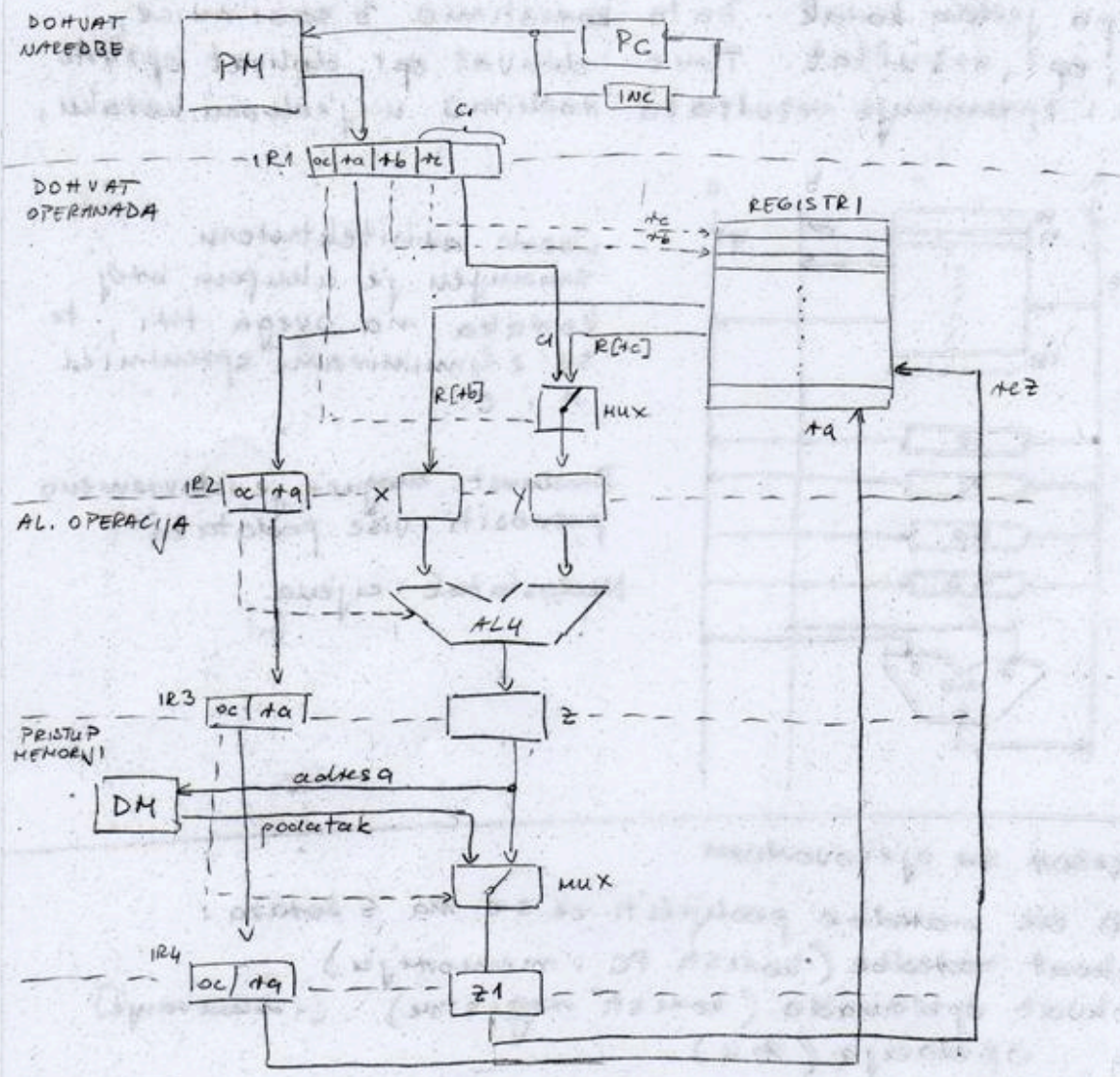
(5) Procesor sa cjevovodom

Općenito sve naredbe podijeliti će se na 5 koraka:
1.) Dohvat naredbe (koristi PC i memoriju)
2.) Dohvat operanada (koristi registre) (+ dekodiranje)
3.) AL operacija (ALU)
4.) pristup memoriji (koristi memoriju)
5.) Pohrana rezultata (koristi registre)



Uvijek je jedna ili više naredbi u fazi obrade. Cjevovod podrazumijeva dohvat i početak izvođenja sljedeće naredbe prije nego što je završila prethodna naredba → paralelizam
U jednom taktu izbacuje jednu naredbu (nakon što se napuni)

Korak 1 i korak 4 se ne mogu raditi istovremeno sa memorijom, stoga se memorija dijeli na programsku memoriju (dohvat naredbe) i podatkovnu memoriju (pristup memoriji). Arhitektura sa takvom podjelom memorije se zove Harvardska arhitektura.



Korak 1. Naredba na koju pokazuje PC se dohvaća iz programske memorije, a sadržaj PC-a se inkrementira. Na kraju se upisuje naredba u IR1 i nova vrijednost u PC

Korak 2. Naredba se čita iz IR1. Dohvaćaju se ili sadržaji R[tb] i R[tc] ili R[tb] i c1 (ovisno o MUX-u). Prvi operand se upisuje u spremnik X, a drugi u Y.

Korak 3. Iz IR2 vidimo koja će se operacija obaviti, rezultat se pohranjuje u spremnik Z

Korak 4. Ovisno da li se pristupa memoriji (DM) ili ne, sprema se u Z1. (preko MUX-a, ide adresa) → to vidimo u IR3

Korak 5. Rezultat iz Z1 se sprema u spremnik R[ta] (što očitamo iz IR4)