Paralelni Računarski sistemi

Zoran Jovanović

Koliko traje operacija?

- **Jedinstveni takt sinhronizuje faze izvršenja Operacije.**
- **Određuje se trajanje celokupne operacije u ciklusima.**
- **Minimalno trajanje operacije je veće ili jednako minimalnom kašnjenju od trenutka kada su argumenti na raspolaganju, do trenutka kada se može prihvatiti rezultat.**

Polazno ograničenje

- **Kod bez programskih petlji i uslovnih grananja - sekvenca ili bazični blok.**
- **Sekvencijalno napisan kod bazičnog bloka određuje jedan od semantički ispravnih redosleda izvršavanja operacija.**
- **Između dve operacije se definiše relacija prethođenja koja određuje redosled u sekvencijalno napisanom kôdu.**
- **Binarna relacija prethođenja određuje redosled javljanja u sekvencijalnom kodu bazičnog bloka**

KADA SE MOŽE IZMENITI REDOSLED DVE OPERACIJE ?

- **Direktna prava zavisnost po podacima je relacija koja znači: rezultat Opi je argument Opj.**
- **Opj sme da započne dohvatanje argumenata posle početka Opi, i to najranije onoliko ciklusa kasnije koliko traje Opi. (Smatra se da Opi u zadnjem ciklusu generiše rezultat)**
- **Ako je i Opk direktno zavisan po podacima od Opj, Opk tada mora biti zavisna po podacima i od Opi (ali ne mora i najčešće nije direktno).**
- **Zavisnost po podacima je relacija koja pruža informaciju da li se može izmeniti redosled dve operacije.**

TRANZIVNOST ZAVISNOSTI PO PODACIMA

- **Činjenica da je Opk zavisan po podacima od Opi, ne sadrži nikakvu novu informaciju o mogućnosti izmene redosleda izvršavanja, jer se može izvesti iz direktnih zavisnosti, jer je relacija tranzitivna**
- **Može postojati i tranzitivna direktna zavisnost po podacima.**
- **Da li samo prava zavisnost po podacima sprečava izmenu redosleda operacija?**

Antizavisnost

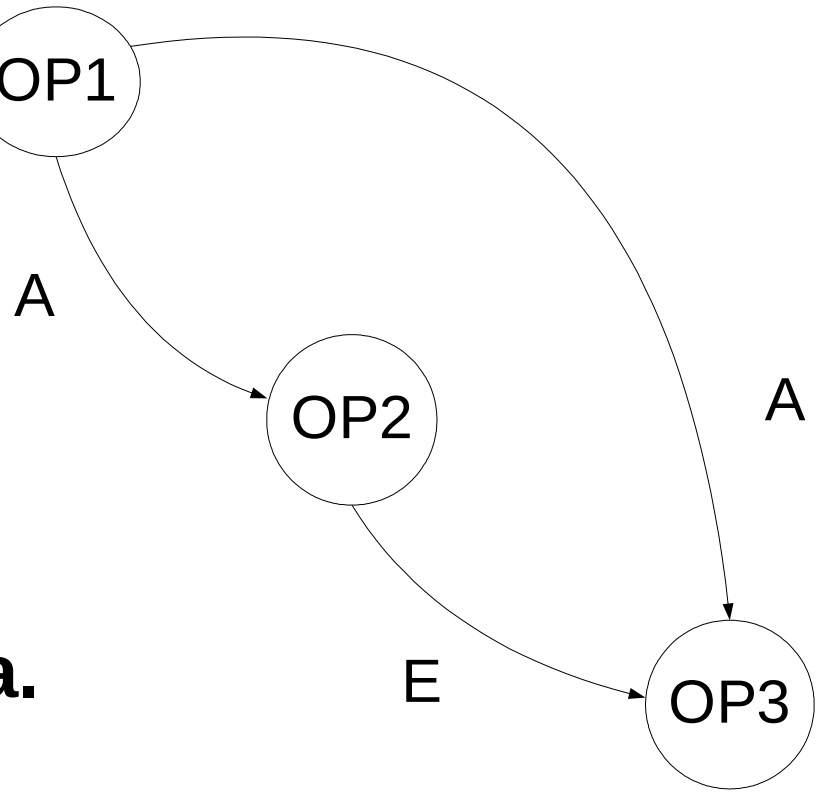
- **Pretpostavka: rezultat operacije se pamti u memorijskoj lokaciji, pa se iz lokacije prihvata kao argument drugih operacija.**
- **Opi prethodi Opj. Opj upisuje u lokaciju iz koje Opi uzima argument. Izmenom redosleda, Opi bi koristio pogrešan argument. To se naziva antizavisnost po podacima (upis pre čitanja).**
- **Kako prevazići ograničenje koje proizilazi iz antizavisnosti? - U jednoj lokaciji čuvati vrednost argumenta za Opi (staru), a rezultat Opj čuvati u drugoj lokaciji. Za argumente operacija posle Opj u sekvencijalnom kôdu koristiti samo vrednost iz druge lokacije.**
- **Funkcionalno programiranje - u svaku lokaciju se samo jednom upisuje sadržaj.**

GRAFOVI ZAVISNOSTI PO PODACIMA

- Binarna relacija direktne prave zavisnosti po podacima između operacija se predstavlja granom grafa, a operacije se predstavljaju čvorovima.
- Sve grane su orijentisane, jer je relacija zavisnosti jednosmerna (antisimetrična).
- Orijentisani grafovi – sve grane moraju biti orijentisane. Polaze od čvora koji generiše rezultat i završavaju u čvoru koji rezultat koristi kao argument

Tranzitivnost

Odmah se eliminišu tranzitivne grane, jer ne nose nikakvu novu informaciju o mogućnosti izmene redosleda operacija.



TRANZITIVNA GRANA OP1 OP3:

OP1: A:=B+C;

OP2: E:=A+D;

OP3: F:=E*A;

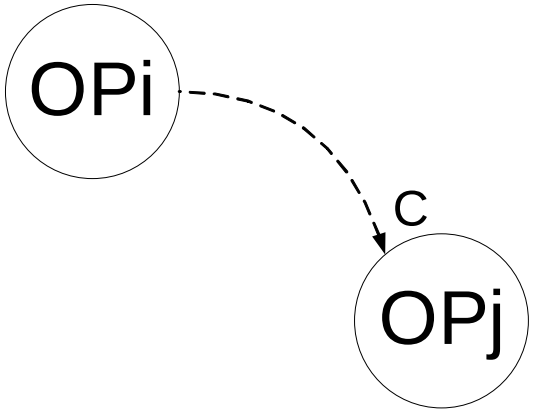
PRIMER ELIMINACIJE ANTIZAVISNOSTI

PRIMER:

. . .

Opi: E:=C*D;

. . .



Opj: C:=A+B;

. . .

Eliminacija antizavisnosti – C1:=A+B; i u kodu koji sledi iza Opj se C zamenjuje sa C1.

Jezici za funkcionalno programiranje obezbeđuju eliminaciju antizavisnosti na nivou jezika - ne može se upisati u istu lokaciju 2 puta. Tipično - eliminacija antizavisnosti obavlja u prevodiocu ili u toku izvršavanja na paralelnoj mašini. Eliminacija antizavisnosti može da se uradi pre formiranja Grafova zavisnosti po podacima.

IZLAZNA ZAVISNOST

- **Pretpostavimo: Opj upisuje sadržaj u lokaciju u koju i Opi upisuje.**
- **Izmenom redosleda Opi i Opj, sve operacije koje koriste sadržaj te lokacije kao argument, a bile su iza Opj bi dobijale pogrešnu vrednost.**
- **Izlazna zavisnost po podacima (upis pre upisa).**
- **Kako prevazići ograničenje koje proizilazi iz izlazne zavisnosti?**
- **U jednoj lokaciji sačuvati staru vrednost rezultata za opi, a rezultat opj sačuvati u nekoj drugoj lokaciji, i posle koristiti samo vrednost iz druge lokacije.**

PRIMER ELIMINACIJE IZLAZNE ZAVISNOSTI

PRIMER:

. . .

Opi: E:=C*D;

. . .

Opj: E:=A+B;

. . .

- **Eliminacija izlazne zavisnosti – E1:=A+B;**
- **U sekvencijalnom kodu koji sledi iza Opj se E zamenjuje sa E1.**

POSTUPAK GRAFA ZAVISNOSTI PO PODACIMA BEZ REDUNDANSE:

1. Pronaći sve antizavisnosti i izlazne zavisnosti.
2. Eliminisati antizavisnost i izlazne zavisnosti pomoću dodatnih promenljivih.
3. Eliminisati direktne prave zavisnosti po podacima koje su tranzitivne.

Dakle - formirali smo graf zavisnosti po podacima na osnovu pravih direktnih zavisnosti po podacima koje nisu tranzitivne.

ACIKLIČKI GRAFOVI ZAVISNOSTI PO PODACIMA

PRIMER ZA SEKVENCU – BAZIČNI BLOK

OP1: A:=B+G;

OP2: E:=A*D;

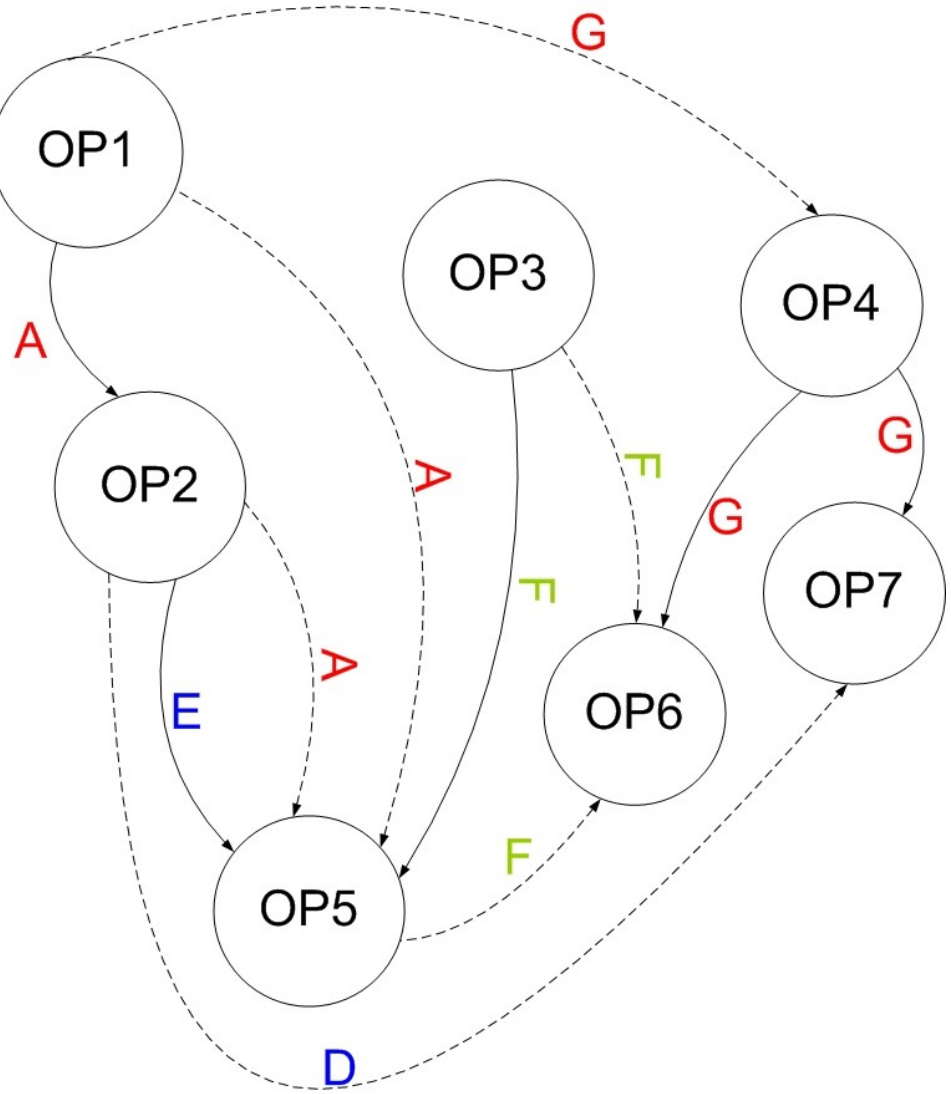
OP3: F:=B+C;

OP4: G:=M*M;

OP5: A:=E/F;

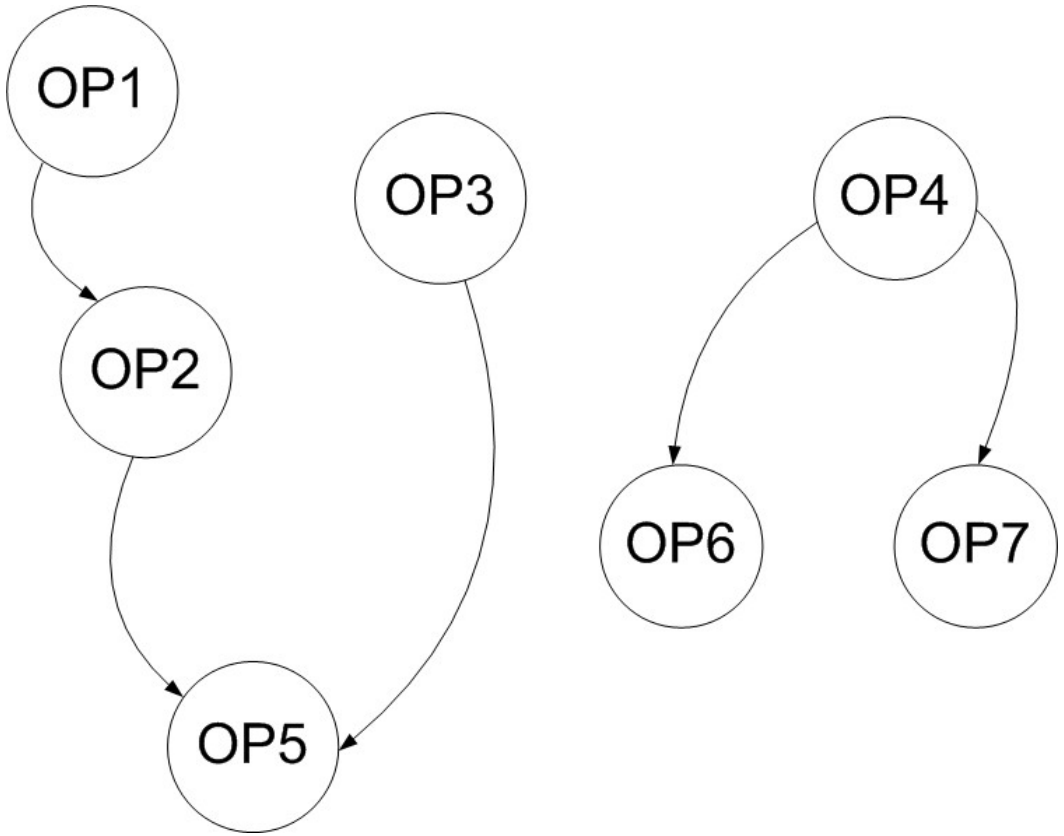
OP6: F:=5+G;

OP7: D:=G*H;



POSLE ELIMINACIJE ANTIZAVISNOSTI I IZLAZNIH ZAVISNOSTI

OP1: **A:=B+G;**
OP2: **E:=A+D;**
OP3: **F:=B+C;**
OP4: **G1:=M*M;**
OP5: **A1:=E/F;**
OP6: **F1:=5+G1;**
OP7: **D1:=G1*H;**



IDEALNA MAŠINA

- **Mašina sa beskonačno mnogo procesora; izlaz bilo kog procesora se bez kašnjenja dovodi do ulaza u bilo koji drugi procesor; trajanje svake operacije je 1 ciklus (pojednostavljenje).**
- **1. Ciklus: sve operacije koje nisu bile zavisne po podacima ni od jedne operacije se mogu odmah izvršiti. Taj skup se naziva *slobodan skup operacija*.**
- **''Skraćivanje'' grafa sa gornje strane eliminacijom operacija iz slobodnog skupa.**
- **2. Ciklus: eliminacijom operacija iz prethodnog slobodnog skupa nastaje novi slobodan skup operacija. Izvrše se sve operacije iz tog novog slobodnog skupa. ...**

PRIMER IZVRŠAVANJA NA IDEALNOJ MAŠINI

OP1: A:=B+G;

OP2: E:=A*D;

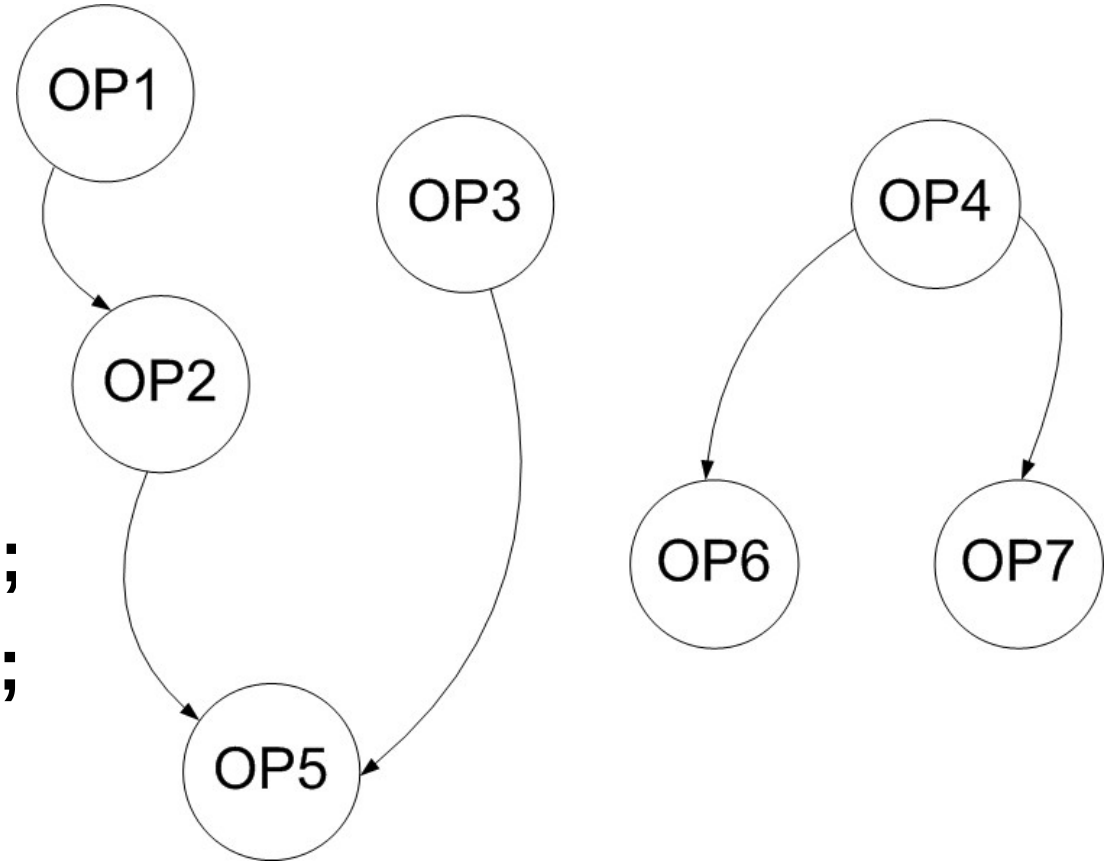
OP3: F:=B+C;

OP4: G1:=M*M;

OP5: A1:=E/F;

OP6: F1:=5+G1;

OP7: D1:=G1*Z;



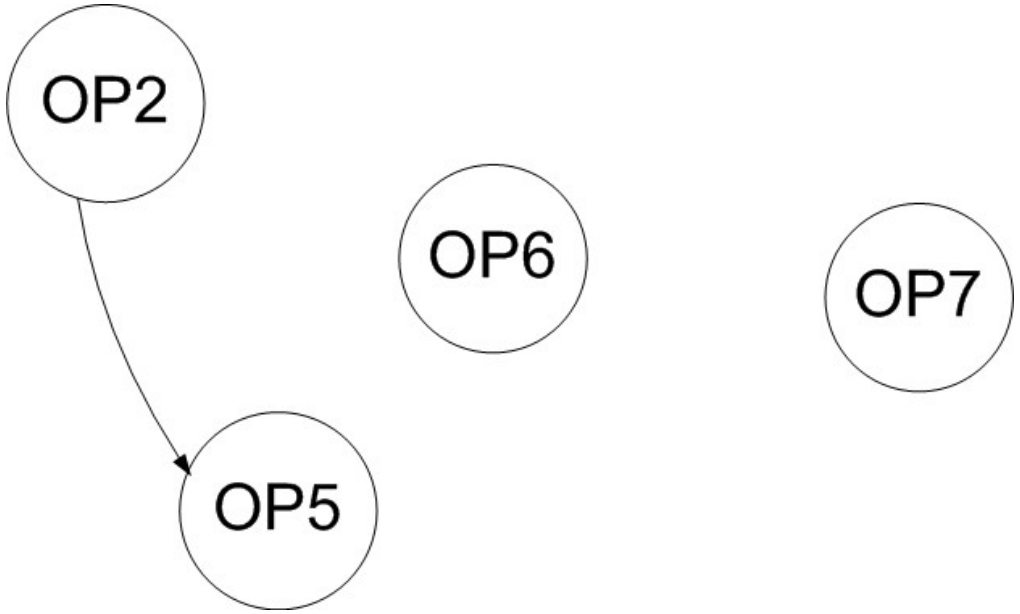
2. CIKLUS

OP2: **E:=A+D;**

OP5: **A1:=E/F;**

OP6: **F1:=5+G1;**

OP7: **D1:=G1*Z;**



3. CIKLUS

- **OP5: A1:=E/F;**
- **NAJDUŽI PUT (KRITIČAN PUT) JE NAJKRAĆE VREME IZVRŠAVANJA NA IDEALNOJ MAŠINI.**