Građa računala

izv. prof. dr. sc. Tomislav Hrkać
Fakultet elektrotehnike i računarstva
email: tomislav.hrkac@fer.hr

ak. godina 2022/23

Građa računala – sadržaj predmeta

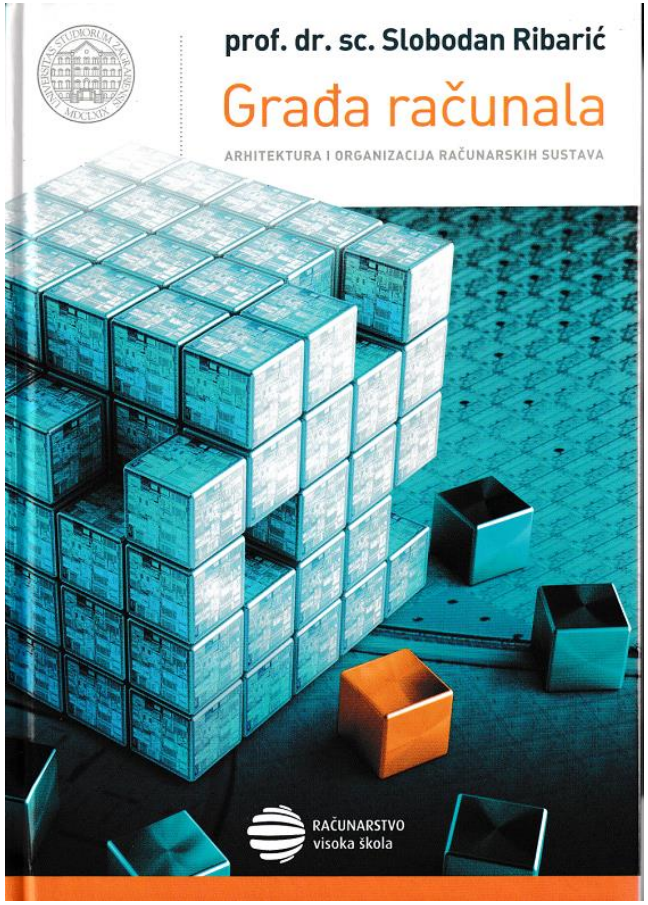
1. Uvodno predavanje (osnovni računski modeli, upravljački tok, tok podataka, upravljanje utemeljeno zahtjevom)
2. Turingov stroj
3. Arhitektura, organizacija i građa računarskog sustava
4. Flynnova klasifikacija arhitekture
5. Von Neumanov model računala
6. Pojednostavljeni modeli CISC i RISC procesora

Građa računala – sadržaj predmeta

7. Performansa računala
8. Zbirni jezik i prevodioci za zbirni jezik
9. Upravljačka jedinica (mikroprogramska i sklopovska izvedba)
10. Aritmetičko-logička jedinica
11. Memorijski sustav
12. Priručna (cache) memorija
13. Virtualna memorija
14. U/I podsustav
15. Protočnost
16. Višeprocesorski sustavi, višejezgreni i grafički procesori

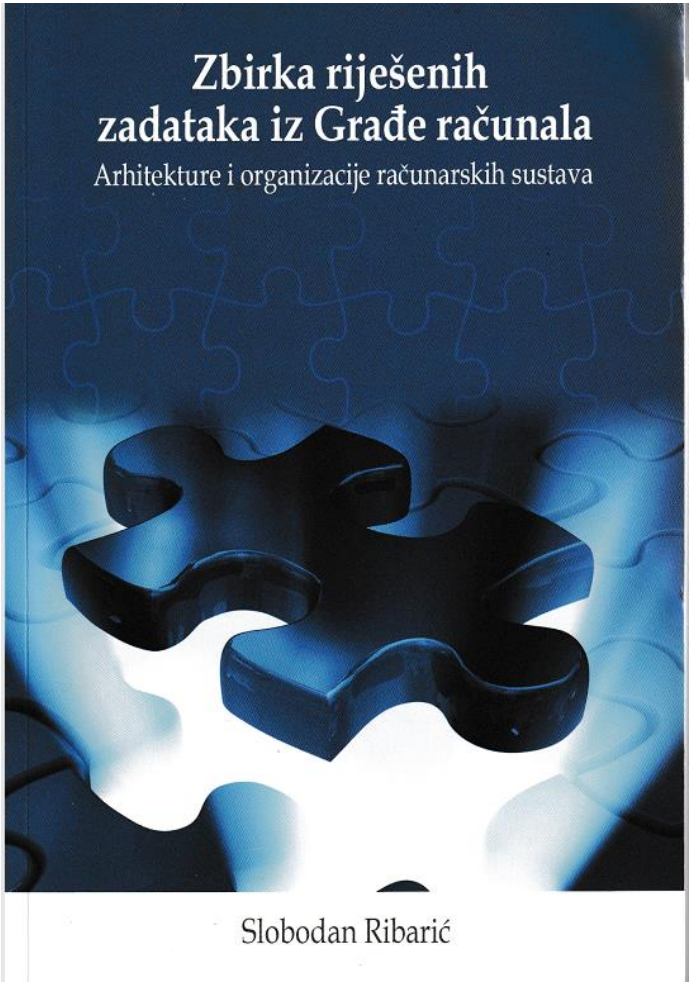
Osnovna literatura (1)

1. S. Ribarić, *Građa računala*, *Arhitektura i organizacija računarskih sustava,* Algebra, Zagreb, 2011.(542 stranice)



Osnovna literatura (2)

2. S. Ribarić, *Zbirka riješenih zadataka iz Građe računala, Arhitekture i organizacije računarskih sustava,* Merkur A.B.D. 2017. (399 stranica)



Dodatna literatura

3. D. A. Patterson, J. L. Hennessy, *Computer Organization & Design, The Hardware/Software Interface,* Morgan Kaufmann Pub., Second edition, 2014. https://ict.iitk.ac.in/wp-content/uploads/CS422-Computer-Architecture-ComputerOrganizationAndDesign5thEdition2014.pdf

4. J. L. Hennessy, D. A. Patterson, *Computer Architecture, A Quantitative Approach,* Morgan Kaufmann Pub., Fifth edition, 2011. http://acs.pub.ro/~cpop/SMPA/Computer%20Architecture%20A%20Quantitative%20Approach%20(5th%20edition).pdf

5. A. S. Tanenbaum, *Structured Computer Organization*, Prentice-Hall, 2006. http://fuuu.be/polytech/INFOF201/Bouquin%20syst%C3%A8mes%20d_exploitations.pdf

6. S. Ribarić, *Naprednije arhitekture mikroprocesora*, Element, Zagreb, 2.izdanje, 1997. (231 stranica)

7. S. Ribarić, *Arhitektura RISC i CISC računala,* Školska knjiga, Zagreb, 1996.

(380 stranica)

Uvodno predavanje

(Građa računala, Arhitektura i organizacija računarskih sustava, str. 1 - 18)

• Vrlo pojednostavljeno, *obradu podataka* (engl. *data processing*) možemo opisati kao svrsishodnu aktivnost koja ima za cilj dobivanje tražene informacije iz raspoloživih podataka.

• Aktivnosti: *pohrana* i *obrada* velikih količina podataka, *slanje*, odnosno upućivanje tražene informacije prema odredištu te njezino ponovno pohranjivanje.

• Tri glavne komponente obrade:
  - podaci,
  - algoritam i
  - izvršitelj.

• *Podaci* su objekti u obradi i moraju biti predočeni u obliku koji je prilagođen izvršitelju;
• *Algoritam* predstavlja preciznu uputu ili “recept” izvršitelju kojom se opisuje transformacija početnih ili ulaznih podataka u procesu obrade u traženu informaciju;
Postupak transformacije grupiran je u korake algoritma.
Svaki algoritam ima svojstva *određenosti*, *konačnosti* i *širinu primjene*, odnosno definirano područje uporabe.

• *Izvršitelj* može biti čovjek ili stroj

Primjer 1.1.
Ilustrirajmo algoritam i njegova svojstva na primjeru Euklidovog algoritma (Euklid, 400. p.n.e) kojim se rješava sljedeći problem: za zadana dva prirodna broja *a* i *b* treba naći najveću zajedničku mjeru, odnosno najveći cijeli broj kojim su oba prirodna broja djeljiva bez ostatka.

1. korak: Promotri dva broja: *a* i *b*.
Prijeđi na sljedeći korak.
2. korak: Usporedi brojeve.
Prijeđi na sljedeći korak.
3. korak: Ako su promatrani brojevi jednaki, svaki daje traženi rezultat – obustavi postupak računanja. Ako brojevi nisu jednaki, prijeđi na sljedeći korak.
4. korak: Ako je prvi promatrani broj manji od drugog, zamijeni im mjesta.
Prijeđi na sljedeći korak.
5. korak: Oduzmi drugi broj od prvog i promatraj taj drugi broj i ostatak.
Prijeđi na korak 2.

Određenost se Euklidovog algoritma ogleda u tome što je u svakom koraku precizno naznačeno što se mora poduzeti i na koji se sljedeći korak u postupku treba prijeći;

Algoritam je tako sročen da jamči da će se za svaka dva prirodna broja *a* i *b* u *konačnom* broju koraka naći najveća zajednička mjera.

Područje uporabe Euklidovog algoritma su *prirodni brojevi;*

Možemo uočiti da je algoritam namijenjen i prilagođen čovjeku kao izvršitelju;

Oblik algoritma u velikoj mjeri ovisi o sposobnosti izvršitelja.

Jedan od uobičajenih pristupa obradi podataka je i onaj koji se temelji na transformaciji ulaznih podataka u traženi rezultat na temelju *računanja*.

Računanje je proces određivanja izlazne supstitucije, za zadanu određenu ulaznu supstituciju, koja se pokorava svim specifičnim svojstvima, odnosno ograničenjima problema.

- izlazna supstitucija?
- ulazna supstitucija?
- ograničenje problema?

Problem naprtnjače:

Primjer 1. 2.
Pretpostavimo da imamo naprtnjaču nosivosti (ili kapaciteta) C kilograma te da pred sobom imamo N predmeta koje bismo željeli ponijeti u naprtnjači. Svaki od predmeta ima svoju vrijednost v_i i svoju težinu w_i, i = 1, 2, ..., N. Budući da sve predmete ne možemo ponijeti zbog ograničenog kapaciteta naprtnjače, potrebno je naći predmete, odnosno udjele predmeta koje ćemo ponijeti tako da u naprtnjači koju nosimo imamo najveću vrijednost, a pritom ne prekoračimo nosivost naprtnjače. (Opaska: predmeti se mogu rastaviti na dijelove čija je vrijednost proporcionalna njihovoj težini.)
Parametri i varijable problema su:
C – nosivost (kapacitet) naprtnjače [kg],
N – ukupan broj predmeta,
v_i; i = 1, 2, ..., N – vrijednost svakog predmeta [$],
w_j; i = 1, 2, ..., N – težina svakog od predmeta [kg],
F_i; i = 1, 2, ..., N – udio svakog od predmeta koji će se ponijeti u naprtnjači,
P_{maks} – maksimalna vrijednost predmeta (ili udjela predmeta) u naprtnjači [$].

Ulaznu supstituciju dobivamo tako da za zadane parametre problema unesemo njihove vrijednosti. Pretpostavimo da je naprtnjača nosivosti 14 kg te da imamo tri predmeta za koje znamo težinu i vrijednost.

Na primjer, ulazna supstitucija je sljedeća:
C = 14 kg, N = 3, v_1 = 30 $, w_1 = 4 kg, v_2 = 48 $, w_2 = 6 kg, v_3 = 50 $, w_3 = 7 kg.

Rješenje se mora pokoravati *specifičnim ograničenjima* – ne smije se prekoračiti kapacitet naprtnjače (C = 14 kg) i mora zadovoljavati zahtjev da je vrijednost predmeta u naprtnjači maksimalna (P_{maks}).

Rješenje problema jest *izlazna supstitucija*.

U naprtnjaču stavljamo: Cijeli predmet 1 i cijeli predmet 2 te 4/7 predmeta 3. Ukupna vrijednost u naprtnjači je 106,57 $, što je ujedno i maksimalna vrijednost koju možemo ponijeti (P_{maks}).

Algoritam:

1. korak: Razvrstaj objekte na temelju omjera vrijednosti i težine:
predmet 2 = 48/6 = 8 $/kg;
predmet 1= 30/4 =7,5 $/kg;
predmet 3 = 50/7 = 7,14 $/kg
2. korak: Ponavljaj sve dok se naprtnjača ne prenatrpa:
Iz skupa predmeta uzmi predmet s najvećim omjerom vrijednost
težina i smjesti ga cijelog u naprtnjaču.
3. korak: Izvadi iz naprtnjače posljednji predmet kojim je prekoračen kapacitet naprtnjače i razdijeli ga tako da upravo njegovi dijelovi popune naprtnjaču.

Osnovni računski modeli

Računski model predstavlja višu razinu apstrakcije od arhitekture računala i programskog jezika. On se opisuje skupom triju apstrakcija:

1. temeljnim elementima koji sudjeluju u računanju;
2. modelom kojim se opisuje problem i
3. izvršnim modelom.

Prva apstrakcija opisuje elemente i operacije koje se na elementima mogu izvoditi. Na primjer, u von Neumannovom modelu temeljni elementi koji sudjeluju u računanju jesu *podaci* koji su predstavljeni imenima da bi se omogućilo razlikovanje brojnih različitih podataka u postupku računanja. Tako imenovani podaci obično se u programskim jezicima nazivaju *varijablama* i implementirani su u arhitekturi računala pomoću adresa memorijskih lokacija i adresa (imena) registara. Operacije se izvode nad podacima – svaki definirani tip podataka ima i definirani skup operacija.

Model kojim se opisuje problem odnosi se na stil i metode opisa problema. Stilovi su *proceduralni* i *deklarativni*.

Proceduralnim stilom opisuje se postupak računanja u obliku algoritma.

Deklarativnim stilom opisuju se sve činjenice i odnosi koji su relevantni za zadani problem.

Dvije se metode koriste za izražavanje odnosa i činjenica. Prva upotrebljava funkcije, a druga primjenjuje opis odnosa i činjenica u obliku predikata.

Primjer 1.3.
Prikažimo jednostavan primjer proceduralnog i deklarativnog stila opisivanja problema računanja ‘n’ faktorijela uporabom jezika sličnom Pascalu.

Proceduralni stil

```
if n := 0 then
fac := 1
else fac :=1;
for i := 1 to n do
fac :=i * fac;
end;
```

Deklarativni stil

```
fac (0) := 1
fac (n >0) := n * fac(n-1);
```

Izvršni model definiran je trima sastavnicama:

- tumačenjem kako se izvodi računanje,
- izvršnom semantikom i
- upravljanjem slijedom izvršavanja

Na primjer, u von Neumanovom modelu računanja problem je opisan kao slijed instrukcija (uputa) koje rukuju imenovanim podacima i određuju slijed upravljanja. Računanje se u von Neumannovom modelu tumači kao izvršavanje zadanog slijeda instrukcija.

Izvršna semantika može se promatrati kao *pravilo* koje propisuje kako se izvodi pojedini korak računanja.

Von Neumannov model temelji se na *semantici prijelaza stanja*. Ona određuje kako će se za svaku instrukciju promijeniti trenutno stanje stroja.

Osim semantike prijelaza stanja, koja se rabi i u Turingovom modelu računanja, koriste se *semantika toka podataka* (engl. *dataflow semantic*) i *redukcijska semantika* (engl. *reduction semantic*) koje su namijenjene računskom modelu temeljenom na toku podataka (engl. *dataflow*) i redukcijskom računskom modelu.

Upravljanje slijedom izvršavanja može se temeljiti na *upravljačkom toku* (engl*. control-driven execution*), *toku podataka* (engl. *data-driven*) i *upravljanju zahtjevom* (engl. *demand-driven*).

Upravljački tok, tok podataka i upravljanje slijedom izvršavanja na temelju zahtjeva

Poslužit ćemo se jednostavnim modelom instrukcije koji je predočen *n*-torkom argumenata:

(*arg 1 arg 2 ... arg n*),

gdje su argumenti operatori (određuju operaciju na podacima), vrijednosti podataka (operandi i rezultati), adrese operanada, adresa rezultata te adrese koje upućuju na sljedeću instrukciju.
Ovakav oblik zapisa instrukcije dopušta nam prikaz izvršavanja slijeda instrukcija i u računskim modelima koji se korjenito razlikuju od von Neumannovog modela.

Za tri različita načina upravljanja slijedom izvršavanja prikazat ćemo primjer izvođenja programa koji se može u višem programskom jeziku zapisati samo jednom naredbom:

$$z = (a * 5) + (c * d).$$

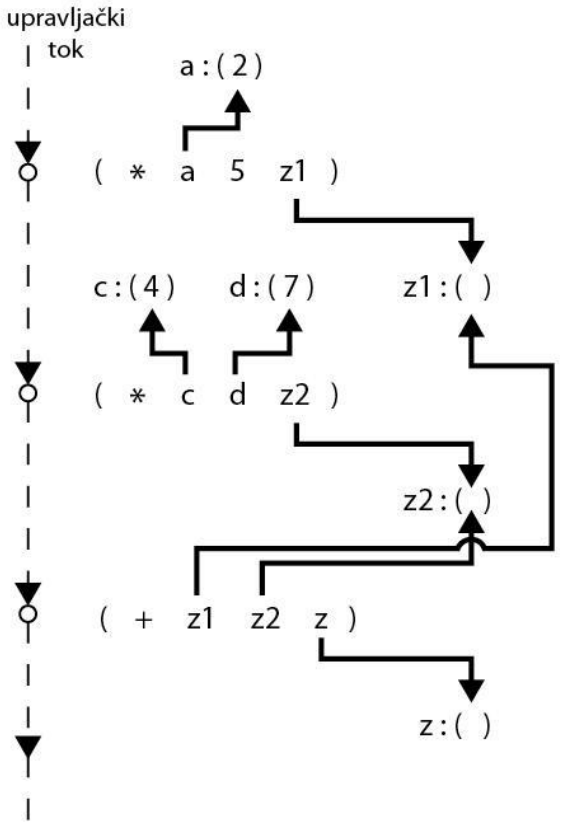
Upravljački tok

Program za *z = (a * 5) + (c * d)* u obliku slijeda instrukcija za model računanja koji se temelji na upravljačkom toku sastoji se od slijeda triju instrukcija:

(* a 5 z1) (* c d z2) (+ z1 z2 z)

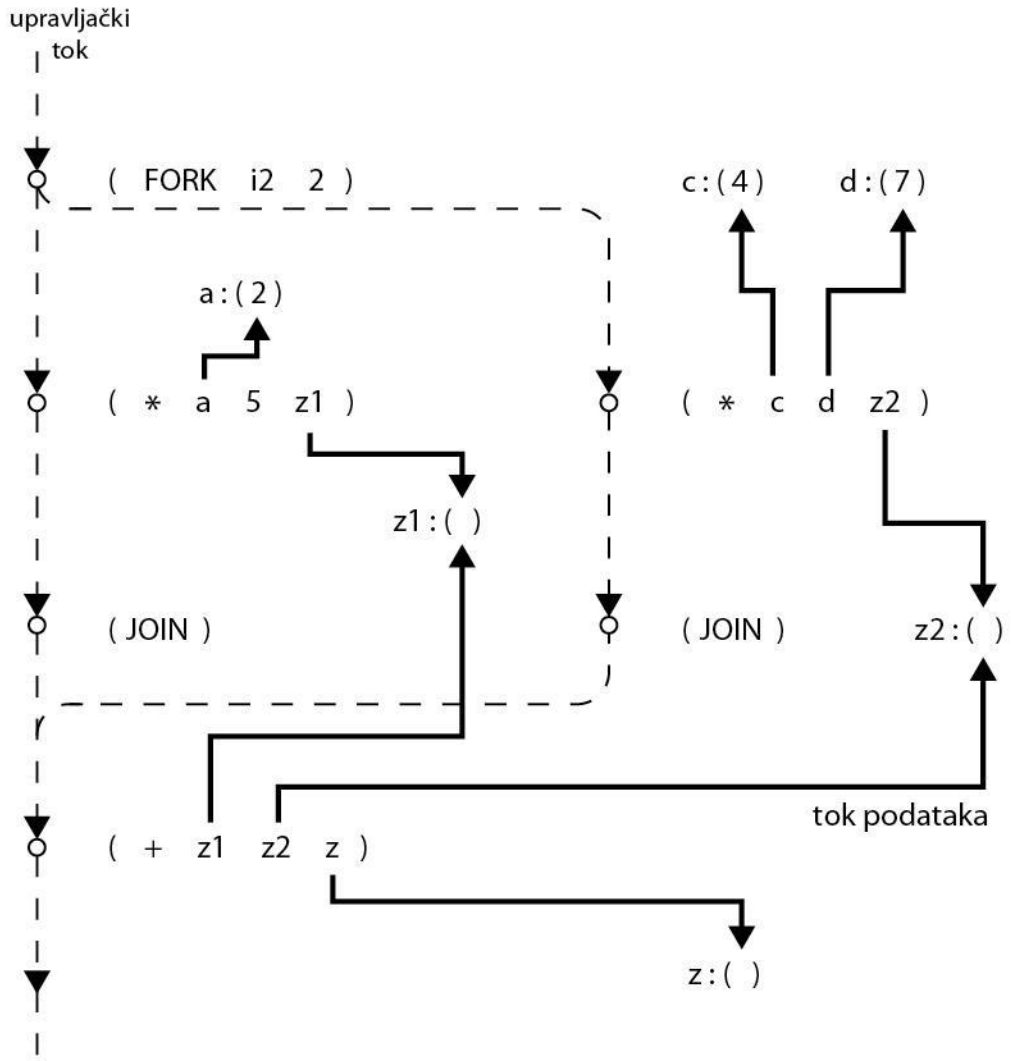
gdje su * i + operatori, 5 vrijednost operanada, *a, c* i *d* adrese operanada, *z1* i *z2* adrese međurezultata i *z* adresa memorijske lokacije na koju se pohranjuje rezultat. U ovom ćemo primjeru pretpostaviti da su na memorijskim lokacijama *a, c* i *d* pohranjeni podaci 2, 4 i 7.
Argument koji upućuje izvođenje na sljedeću instrukciju u ovom modelu računanja nije element instrukcije.

Izvršavanje programa u računskom modelu temeljenom na upravljačkom toku



Slijed izvršavanja implicitno je određen. Implicitni slijed izvršavanja može se promijeniti uporabom eksplicitnih upravljačkih instrukcija koje imaju operatore tipa *GO TO* i *JUMP*

Izvršavanje programa u paralelnom računskom modelu temeljenom na upravljačkom toku s eksplicitno označenim paralelno izvodljivim instrukcijama



Instrukcija *(FORK i2 2)* formira upravljački tok u dvije grane i omogućuje istodobno izvršavanje instrukcija *(* a 5 z1)* i *(* c d z2).*

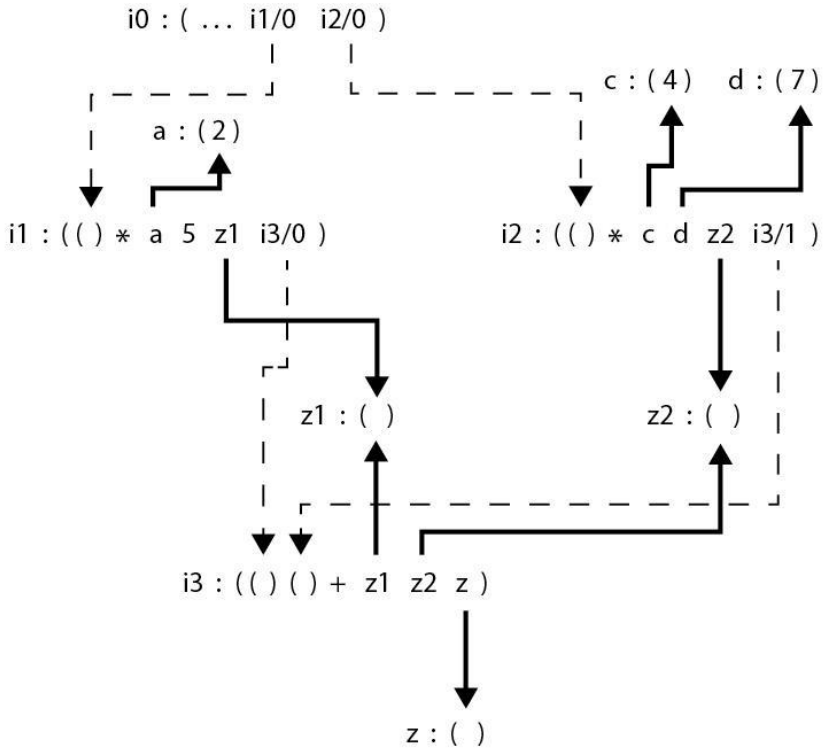
Argument *i2* u instrukciji *FORK i2 2* označava mjesto nastanka novih, dodatnih upravljačkih tokova, a argument *2* predstavlja vrijednost sinkronizacijskog brojila pomoću kojeg se sinkroniziraju upravljački tokovi.

Instrukcija s operatorom *JOIN* sinkronizira upravljačke tokove i nakon izvođenja instrukcija

(a 5 z1)* i *(* c d z2)*

stapa ih u jedinstven upravljački tok. Sinkronizacija se ostvaruje tako da svaka instrukcija s *JOIN* operatorom umanjuje vrijednost sinkronizacijskog brojila za 1. U trenutku kada vrijednost sinkronizacijskog brojila dostigne vrijednost 0, upravljački tokovi se stapaju.

Izvršavanje programa u paralelnom računskom modelu temeljenom na upravljačkom toku s upravljačkim značkama



- u ovom se slučaju koriste instrukcije u kojima se posebnim argumentom određuju sljedeću ili sljedeće instrukcije. Upravljanje s instrukcije na instrukciju prenosi se upravljačkim značkama (engl. control token).

Npr. i3/1 određuje sljedeću (treću) instrukciju a 1 označava da se u i3 i to u prvom argumentu ( ) očekuje upravljačka značka.

Iz prethodnih triju jednostavnih primjera izvođenja programa mogu se izdvojiti neke osnovne značajke računskih modela temeljenih na upravljačkom toku:

• tok upravljanja je slijedan. Uporabom posebnih operatora može se upravljati tokom izvođenja programa,

• za paralelni se računski model tok upravljanja uvišestručuje uporabom specijalnih operatora ili primjenom upravljačkih znački,

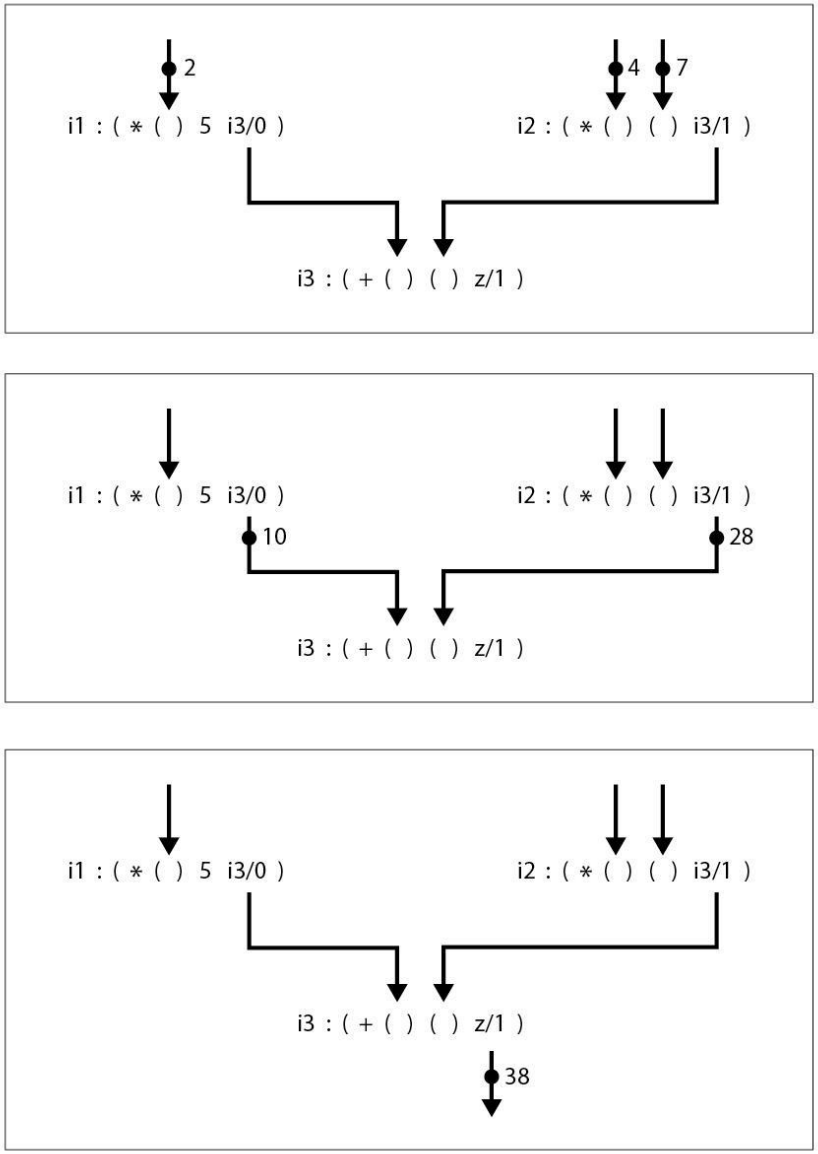
• tok ili tokovi upravljanja i tokovi podataka odvojeni su,

• podaci (operandi) se prosljeđuju između instrukcija posredno – putem memorijskih lokacija što ih instrukcije međusobno dijele (pohranjuju i čitaju sadržaje memorijskih lokacija),

• argumenti instrukcija mogu biti izravno i vrijednosti operanada, tako da se smanjuje broj referenciranja memorije.

Tok podataka

Instrukcija se u tom modelu izvršava u trenutku kada su joj raspoloživi svi operandi, odnosno kada su u instrukciji popunjeni značkama podataka svi kalupi tj. argumenti oblika ( ).



Primjer izvršavanja programa za taj model pokazuje neke njegove osnove značajke:

• upravljački tok stopljen je s tokom podataka,

• tok podataka *uvjetuje* izvršavanje instrukcija: instrukcije se izvršavaju u trenutku kada su im raspoloživi svi potrebni operandi, odnosno kada su popunjeni svi kalupi instrukcije,

• međurezultati se izravno prosljeđuju među instrukcijama,

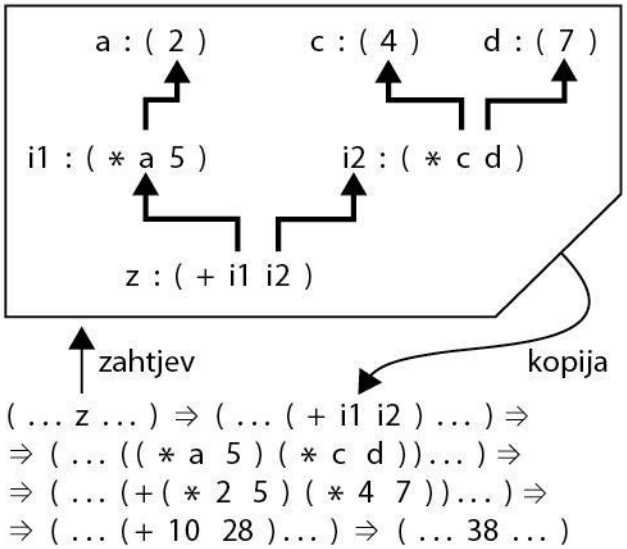
• nakon izvršavanja instrukcije, ulazni podaci za instrukciju (oni koji su bili smješteni u kalupima) nisu više raspoloživi za tu niti za bilo koju drugu instrukciju,

• vrijednost operanada mogu biti argumenti instrukcije.

Upravljanje zahtjevom

Osnovna značajka računskog modela upravljanog zahtjevom je ta da zahtjev za rezultatom pobuđuje operacije koje će taj rezultat proizvesti. Na primjer, za naš se program

*z = (a * 5) + (c * d)* zahtijeva vrijednost za *z*



Vrijednost za z dobiva se redukcijom definicije (slika):

(2* 5) +(4*7) → 10 + 28 → 38

Osnovne značajke računskog modela upravljanog zahtjevom su:

• nema dodatnih ograničenja u pogledu upravljanja slijedom izvršavanja instrukcija osim onih koja su postavljena zahtjevima za rezultatom,

• ne rabi se koncept pohranjivanja i obnavljanja vrijednosti (nema varijabli),

• dopušteno je gniježđenje zahtjeva za rezultatima.

Tri su osnovna računska modela koji se temelje na konceptu podataka:

1. Turingov model,
2. von Neumannov model,
3. model toka podataka.

Spomenimo neke računske modele koji se *ne* temelje na konceptu podataka – to su *objektno temeljeni modeli* (engl. *object-based model*) u kojima se manipulira objektima slijedom poruka, *aplikativni modeli* u kojima su temeljni elementi koji sudjeluju u računanju argumenti a problem je opisan skupom definicija funkcija koje se vrednuju, i *predikatni računski modeli* koji koriste elemente skupove predikata (engl. *predicate-logic-based*) i u kojima se opisi problema predočavaju Hornovim klauzulama kojima se izražavaju pravila i činjenice.

(Hornove klauzule HK
konstante, varijable i funkcije -> termi; predikati -> atomi; atomi ili njegove negacije -> literali;
HK -> klauzula (disjunkcija literala) s najviše jednim pozitivnim literalom:

$$\neg p \vee \neg q \vee \ldots \vee \neg t \vee u \;)$$