# Organizacija i arhitektura računala

**Poglavlje 12**

***Reduced Instruction Set Computers***

***RISC***

# Značajniji napredci u računalnim sustavima (1)

- koncept "obitelji" računala
  - IBM System/360  1964
  - DEC PDP-8
  - odvajaju se arhitektura od načina implementacije
- Mikroprogramirana upravljačka jedinica - mikrokod
  - to je ideja Wilkes-a 1951
  - a napravio IBM S/360 1964
- Cache memorija
  - IBM S/360 model 85  1969

# Značajniji napredci u računalnim sustavima (2)

- Solid State RAM – poluvodički RAM
  - o ovome je više rečeno pod dijelom koji se bavi memorijom
- Mikroprocesori
  - Intel 4004  1971
- Pipelining
  - Uvođenje postupaka paralelizacije u fetch-execute/zahvat-izvrši intrukcijskom ciklusu
- Višeprocesorski sustavi

# Slijedeći korak - RISC

- Reduced Instruction Set Computer

- Glavne značajke
  - Veliki broj registara za opću namjenu - general purpose registers
  - ili koristiti unaprjeđenu compiler tehnologiju za optimizaciju uporabe registara opće namjene
  - Limitirani/ograničeni i jednostavan skup instrukcija
  - Naglasak je dan na optimizaciju intrukcijskog pipeline-a

# Usporedbe procesora

| **CISC** | | | **RISC** | | **Superscalar** | |
|---|---|---|---|---|---|---|
| IBM | DEC VAX | Intel | Motorola | MIPS | IBM | Intel |
| 370/168 | 11/780 | 486 | 88000 | R4000 | RS/6000 | 80960 |
| 1973 | 1978 | 1989 | 1988 | 1991 | 1990 | 1989 |
| No. of instruction | | | | | | |
| 208 | 303 | 235 | 51 | 94 | 184 | 62 |
| Instruction size (octets) | | | | | | |
| 2-6 | 2-57 | 1-11 | 4 | 32 | 4 | 4 or 8 |
| Addressing modes | | | | | | |
| 4 | 22 | 11 | 3 | 1 | 2 | 11 |
| GP Registers | | | | | | |
| 16 | 16 | 8 | 32 | 32 | 32 | 23-256 |
| Control memory (k bytes) (microprogramming) | | | | | | |
| 420 | 480 | 246 | 0 | 0 | 0 | 0 |

# Zašto onda CISC?

- Troškovi izrade programa danas daleko nadilaze troškove razvoje sklopovlja/hardwarea
- Povećavanje kompleksnosti jezika visokog nivoa
- Semantička rupa!?!
- To sve vodi do:
  - Rasta/povećanja instrukcijskih skupova
  - Sve više i više adresnih modova/načina dresiranja
  - Hardwerska implementacija HLL (high level language) stanja
    - npr. CASE (prekidač/selektor) naredba

# Namjera i namjena CISC procesora

- Jednostavno pisanje/razvijanje kompilera
- Poboljšanje efikasnosti izvršavanja programa
  - Kompleksne operacije u mikrokôdu
- Podrša za još kompleksnije HLL

# Izvršne/obradbene karakteristike

- Obavljanje operacija
- Korištenje operanada
- Sekvenciranje obrade
- Analize se rade na programima pisanim u HLL-u
- Analize dinamike izvršavanja rade se tijekom izvršavanja ovih programa

# Operacije

- Dodjele
  - Prebacivanje/premještanje podataka
- Uvjetna stanja (IF, LOOP)
  - Kontrola sekvencijalnog izvršavanja
- Procedura “call-return” vrlo je zahtjevna u pogledu vremena potrebnog da se izvrši
- Pojedine HLL instrukcije vode do pojave nužnosti uporabe puno operacija strojnog kôda

# Relativna dinamička frekvencija
# Relative Dynamic Frequency

| | Dinamička pojavnost | | Strojna instr. (Weighted) | | Uporaba mem. (Weighted) | |
|---|---|---|---|---|---|---|
| | Pascal | C | Pascal | C | Pascal | C |
| Assign | 45 | 38 | 13 | 13 | 14 | 15 |
| Loop | 5 | 3 | 42 | 32 | 33 | 26 |
| Call | 15 | 12 | 31 | 33 | 44 | 45 |
| If | 29 | 43 | 11 | 21 | 7 | 13 |
| GoTo | - | 3 | - | - | - | - |
| Other | 6 | 1 | 3 | 1 | 2 | 1 |

# Operandi

- Većinom su to lokalne skalarne varijable
- Postupci optimizacije trebaju se koncentrirati na pristup lokalnim varijablama

| | Pascal | C | Average |
|---|---|---|---|
| Integer constant | 16 | 23 | 20 |
| Scalar variable | 58 | 53 | 55 |
| Array/structure | 26 | 24 | 25 |

# Procedure Calls

- Vrlo zahtjevne u pogledu vremena potrebnog za izvršavanje
- Ovise o broju proslijeđenih im parametara
- Ovise i o nivou ugniježenja/nesting
- Većina programa ne radu “puno” poziva procedura bez njihovog povratka
- Većina varijabli je lokalna

# Implikacije

- Najbolja podrška dana je kroz optimizaciju najviše korištenih i vremenski najzahtjevnijih značakji ovakvog sustava
- Velika količina/broj registara
  - Adresno referenciranje na operande
- Pažljivo osmišljavanje cjevovoda/pipeline-a
  - Predviđanje grananja, itd.
- Pojednostavljeni (reducirani) instrukcijski set

# Kako koristiti puno registara?

- Programsko rješenje:
  - Optimizacijom kompilera u svrhu alociranja većeg broja registara
  - Alociranje se temelji na dodjeli registara varijablama koje su najviše u uporabi
  - Ovo zahtijeva sofisticiranu analizu programa da bi se pravilno optimiziralo za uporabu velikog broja registara u takvom procesoru
- Sklopovsko/hardware rješenje:
  - paaa.... imamo puno registara, zar ne?
  - stoga... možemo više varijabli koristiti i jednostavno i spremati u registre a ne u gl. memoriju

# Registri za lokalne varijable

- Pohrana lokalnih skalarnih varijabli u registre
- Smanjena potreba za pristupom gl. memoriji -> brzina
- Svaki poziv procedure/funkcije mijenja pojam "lokalnosti"
- Moraju se proslijediti parametri
- ali se i moraju vratiti rezultati u glavnu petlju
- Mora se uspostaviti i stanje varijabli prozvanih /pozvanih potprograma u prethodno stanje nakon povratka

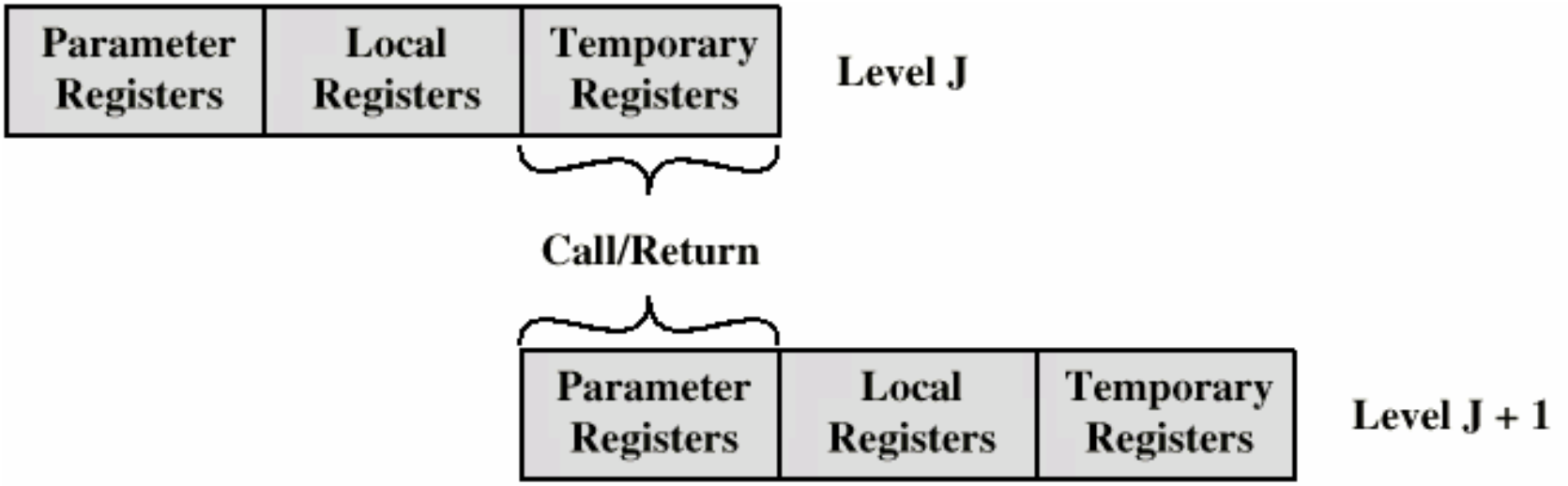
# Registarski prozori/skupovi (1)

- U uporabi je samo nekoliko parametara
- Limitirana dubina poziva procedura
- Uporaba više skupina malih skupova registara
- Poziv procedure prebacuje uporabu na drugi skup registara
- A povratak iz poziva procedure vraća na uporabu prethodnog skupa registara

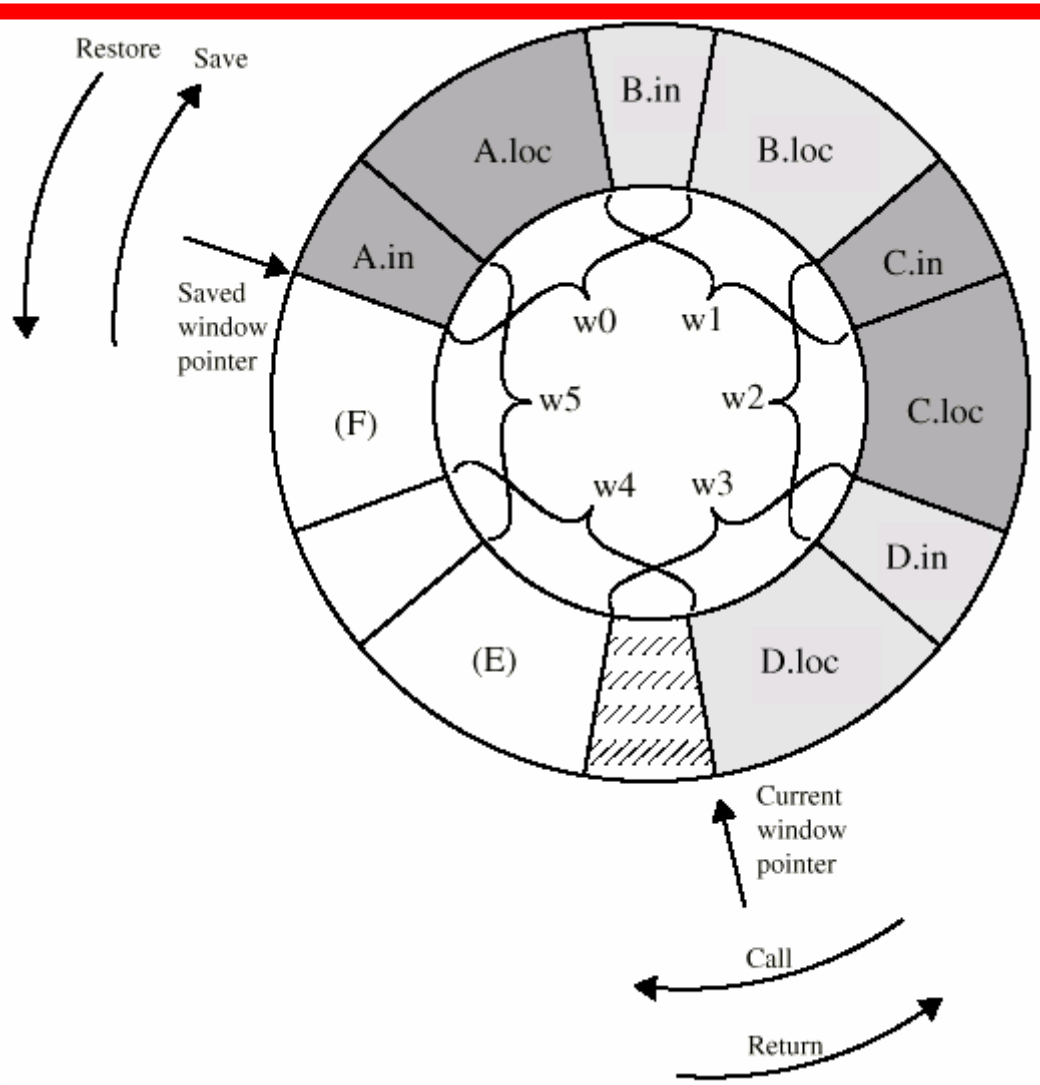
# Registarski prozori/skupovi (2)

- Postoje tri područja/tipa unutar skup registara
  - Registri za parametre
  - Registri za lokalne varijable
  - Registri za privremene parametre/varijable
  - Registri za privremenu uporabu jednog skupa registara preklapaju se sa istim takvim registrima ali iz nekog drugog skupa registara
  - Ovo omogućuje prosljeđivanje parametara bez stvarnog prebacivanja podataka

# Preklapajući registri raznih skupova registara



# Dijagram kružnog međuspremnika/buffer-a



# Način rada kružnog međuspremnika

- Kada se napravi poziv potprograma, trenutni pokazivač skupa registara pomakne se na trenutno korišteni skup registara
- Ako su svi skupovi iskorišteni ili u uporabi, generira se interrupt i "najstariji" skup registara se sprema u memoriju
- Pokazivač spremljenog skupa registara pokazuje gdje se treba ponovo obnoviti/restorirati slijedeći snimljeni skup registara

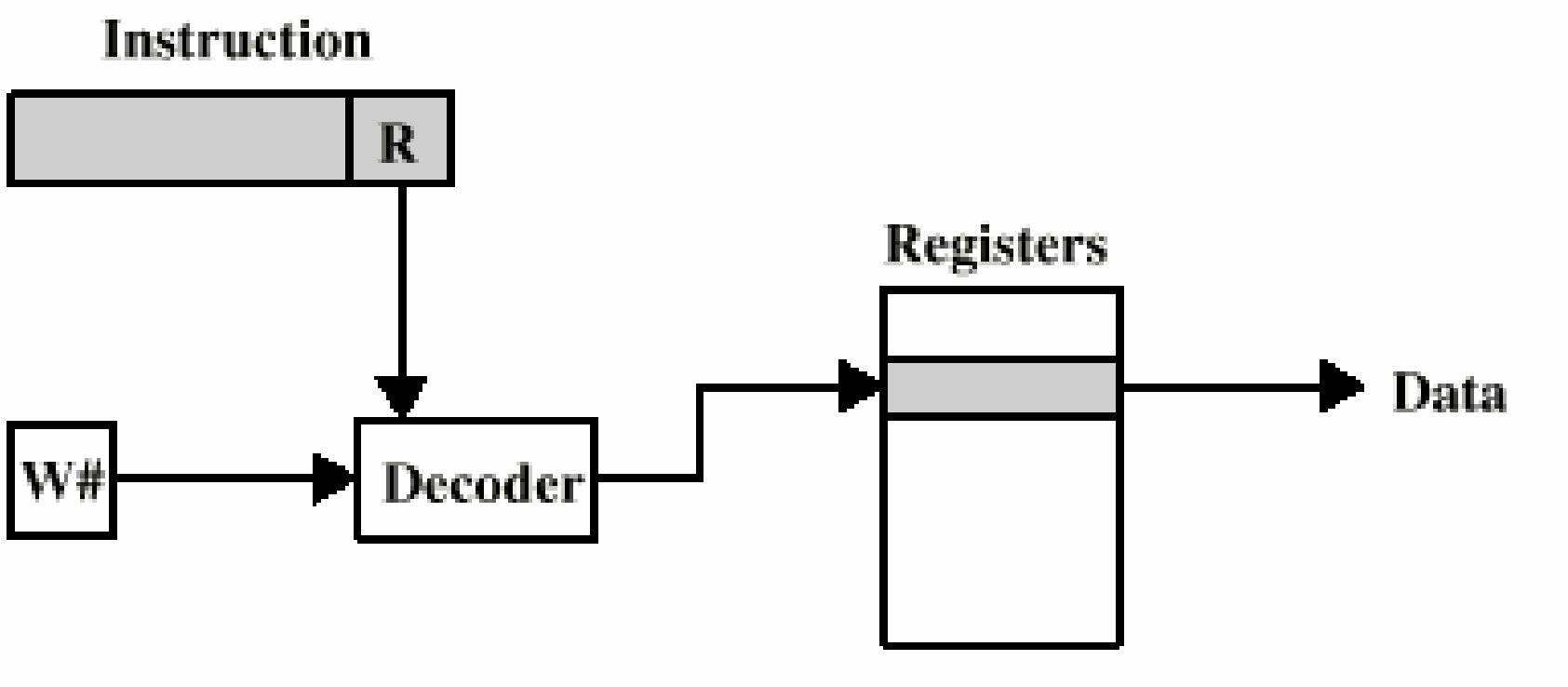
# Globalne varijable

- Alocira ih kompiler u glavnoj memoriji
  - Vrlo neefikasno za učestalo pristupanje takvim varijablama
- Postoje registri rezervirani samo za globalne varijable

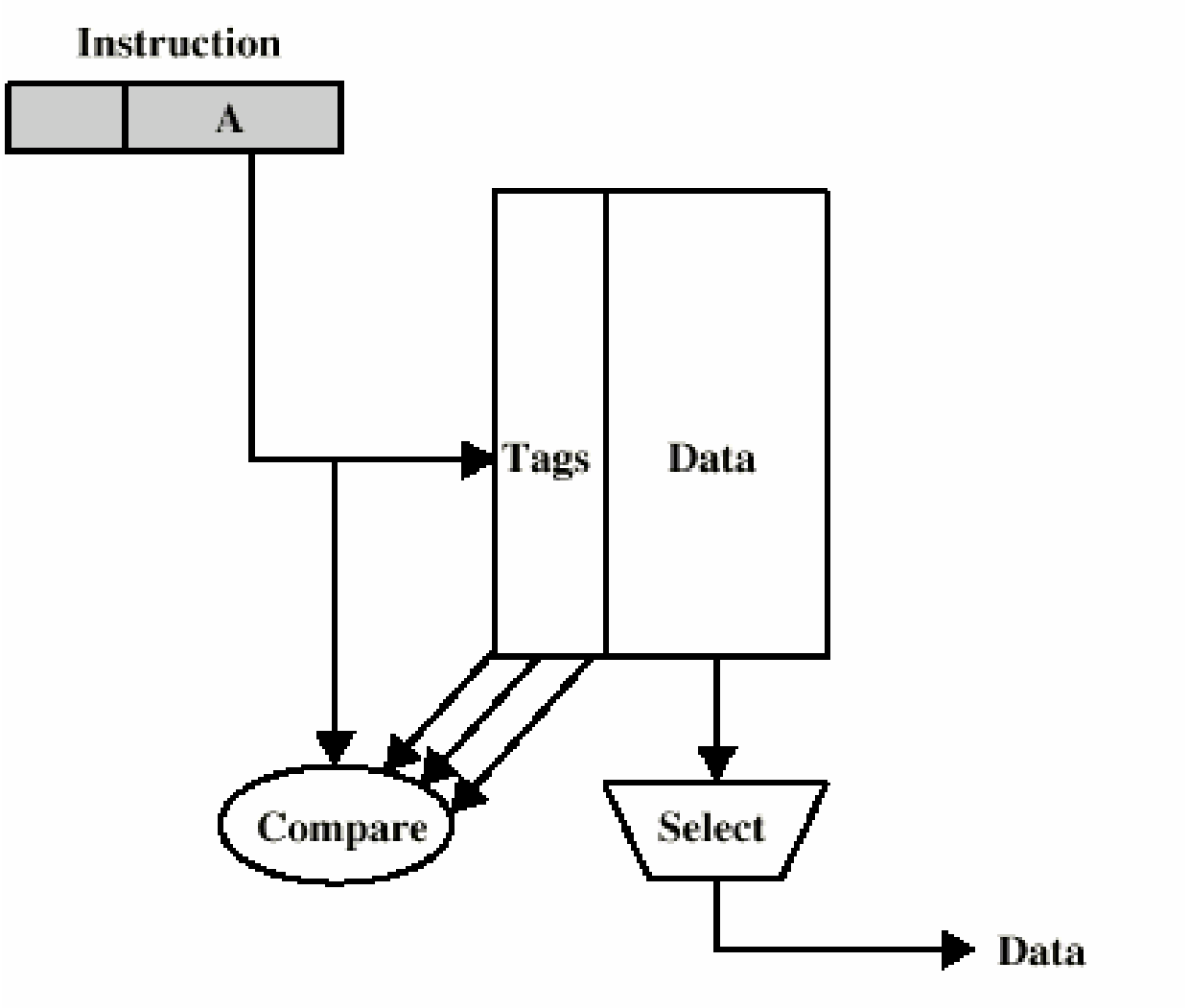
# Registeri vs. Cache

| | |
|---|---|
| ⌘ Large Register File | Cache |
| ⌘ Sve su lokalni skalari | Samo zadnje korišteni lokalni skalari |
| ⌘ Individualne varijable | Blok memorije |
| ⌘ Compiler dodjeljuje globalne varijable | Zadnje korištene globalne varijable |
| ⌘ Save/restore temeljene na proceduri | Save/restore temeljeni na ugniježđenju caching algorithm |
| ⌘ Adresiranje registara | Adresiranje memorije |

# Referenciranje/adresiranje skalara - korištenje skupa registara



# Referenciranje/adresiranje skalara - Cache



# Kompilersko optimiziranje uporabe registara

- Pretpostavimo mali broj registara (16-32)
- Posao postupka optimizacije je na kompileru
- HLL programi nemaju eksplicitne načine da se adresiraju registri – to rade kompileri ili asemblerske rutine
- Dodjeli se simbolički ili virtualni registar svakoj potencijalnoj varijabli
- Mapiraj simboličkim registrima prave/postojeće registre
- Simbolički registri koji se inače ne preklapaju mogu čak dijeliti prave registre
- Ako ponestane pravih registara pojedine varijable mogu koristiti i memoriju umjesto njih

# Zašto onda CISC (1)?

- Pojednostavljenost kompilera?
  - o ovome se može raspravljati...
  - Kompleksne instrukcije teže je i pravovaljano iskoristiti u različite svrhe
  - Optimizacija je u ovom slučaju prilično teška
- Da li su programi manji?
  - Program zauzima manje memorije, ali…
  - Memorija je (sada) relativno jeftina
  - ali možda ne zauzima manje bita nego samo izgleda kraće ali u svojoj simboličkoj formi
    - Složenije instrukcije zahtijevaju i operacijske kodove veće duljine – više memorije
    - Referenciranje registara zahtijeva doista manje memorije

# Zašto onda CISC (2)?

- Brži programi?
  - Naginje se uporabi jednostavnijih instrukcija
  - Jako složena upravljačka jedinica/dekoder
  - Veći mikrokod  više mikrokod memorije
  - ali jednostavne instrukcije ne traže više vremena u svom izvršavanju nego složene

- Kako stvari stoje... CISC se ne smatra baš adekvatnim rješenjem za izgradnju računala

# RISC karakteristike

- Jedna instrukcija po ciklusu izvršavanja
- Registar-registar operacije
- malo, jednostavnih načina adresiranja
- malo, jednostavnih instrukcijskih formata
- Ožićeni dizajn – nema mikrokoda
- FInstrukcijski format fiksne veličine
- ali kompiler tu ima više posla

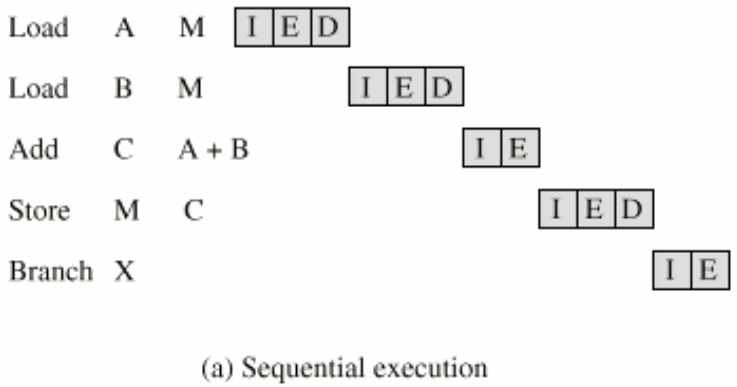
# RISC vs. CISC

- Nema tu jasne granice što, kako i kada
- Mnogi dizajni procesora posuđuju rješenja za izgradnju pojedinih podsustava kako iz CICS-a tako i iz RISC-a
- npr. PowerPC i Pentium II,... to su ustvari mješavine i CISC-a i RISC-a

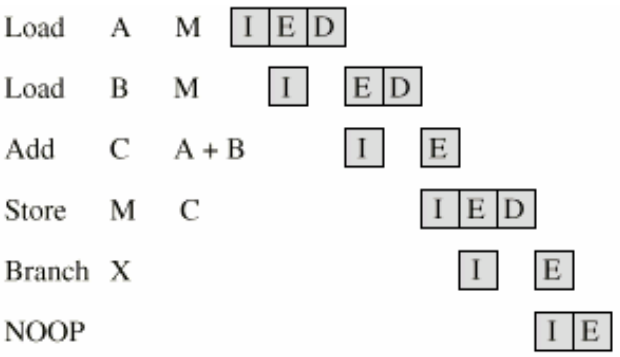
# RISC Cjevovod/Pipelining

- Većina instrukcija je naravi register-register
- Dvije faze instrukcijskog ciklusa
  - I: Instruction fetch – zahvat instrukcije
  - E: Execute - izvršenje
- Te za operacije učitavanj/load i pohrana/store
  - I: Instruction fetch
  - E: Execute
    - Izračun memorijske adrese
  - D: Memory
    - Register→memorija ili memorija→register operacije

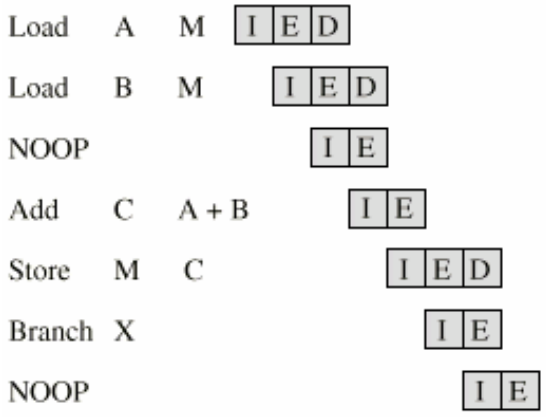
# Efekat cjevovoda/Pipelininga



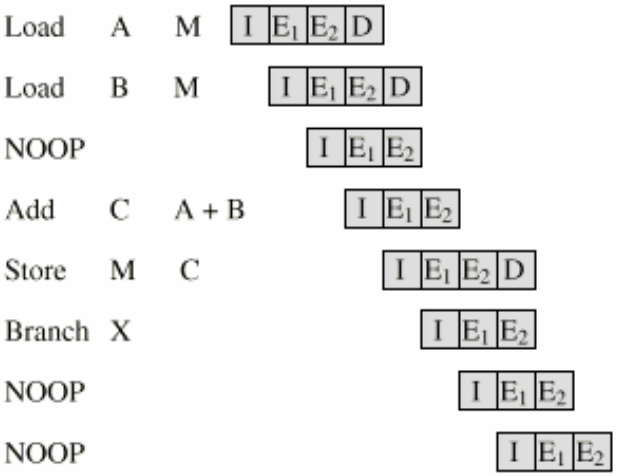

(a) Sequential execution




(b) Two-way pipelined timing




(c) Three-way pipelined timing




(d) Four-way pipelined timing

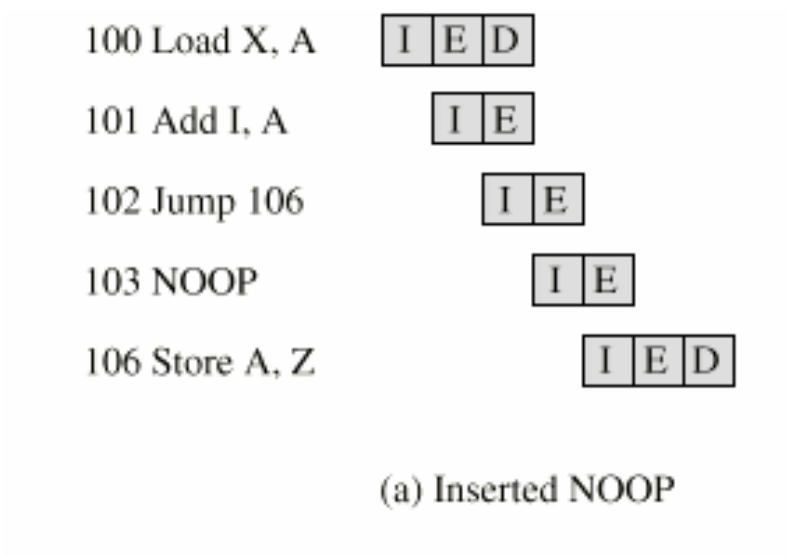
# Optimizacija cjevovoda

- Zakašnjelo grananje
  - Nema efekta sve dok se slijedeća instrukcija u cjevovodu ne izvrši
  - Ova slijedeća instrukcija uvodi i uzrokuje vremensko čekanje

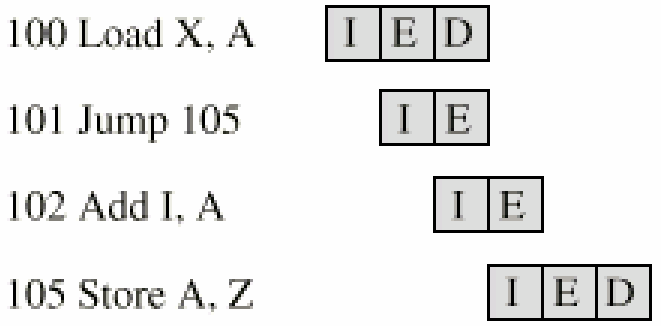
# Normalno i zakašnjelo grananje

| Address | Normal | Delayed | Optimized |
|---|---|---|---|
| 100 | LOAD X,A | LOAD X,A | LOAD X,A |
| 101 | ADD 1,A | ADD 1,A | JUMP 105 |
| 102 | JUMP 105 | JUMP 105 | ADD 1,A |
| 103 | ADD A,B | NOOP | ADD A,B |
| 104 | SUB C,B | ADD A,B | SUB C,B |
| 105 | STORE A,Z | SUB C,B | STORE A,Z |
| 106 | | STORE A,Z | |

# Uporaba zakašnjelog grananja




(a) Inserted NOOP




(b) Reversed instructions

# Kontroverze uporabe RISC-a

- Kvantitativne
  - usporedbu po veličini programa i brzini izvršavanja
- Kvalitativne
  - problematična uporaba HLL-a – optimizacija vrlo naporna
- Problemi
  - Nema dva RISC i CISC procesora koja su izravno usporediva
  - Nema skupa programa za testiranje obje vrste procesora
  - Teško je razlučiti efekte hardwarea od efekata kompilera
  - Većina komercijalnih procesora je mješavina RISC-a i CISC-a tako da je teško pronaći arhitekturni čitsti ili CISC pa tako i RISC