RAČUNARI SA USKLADIŠTENIM PROGRAMOM FON-NOJMANOVI RAČUNARI

Računari sa fiksiranim programom

- ENIAC (1940 - ih)
- Sastojao se iz:
  - ALU jedinice
  - kontrolne jedinice
  - memorije
  - nekoliko pomoćnih registrara
- Problem: program bio fiksiran, tj. bio je implementiran u hardveru

RAČUNARI SA USKLADIŠTENIM PROGRAMOM

- engl. *stored program computers*
- Program, kao i podaci, bude kodiran na binarnom jeziku (koji nazivamo mašinski jezik) i da kao takav bude smešten u memoriju.
- Kontrolna jedinica je konfigurisana tako da implementira INTERPRETATOR mašinskog jezika.
- Povećana fleksibilnost.

RAČUNARI SA USKLADIŠTENIM PROGRAMOM

- Postoje dva tipa:
  - Harvardski:
    - dve odvojene memorije u dva adresna prostora, jedna za čuvanje podataka i jedna za čuvanje programa
    - **Harvard Mark II**
  - Fon - Nojmanovi računari:
    - podaci i mašinski program se čuvaju u istoj memoriji, samo na različitim adresama.

Komunikacija sa memorijom

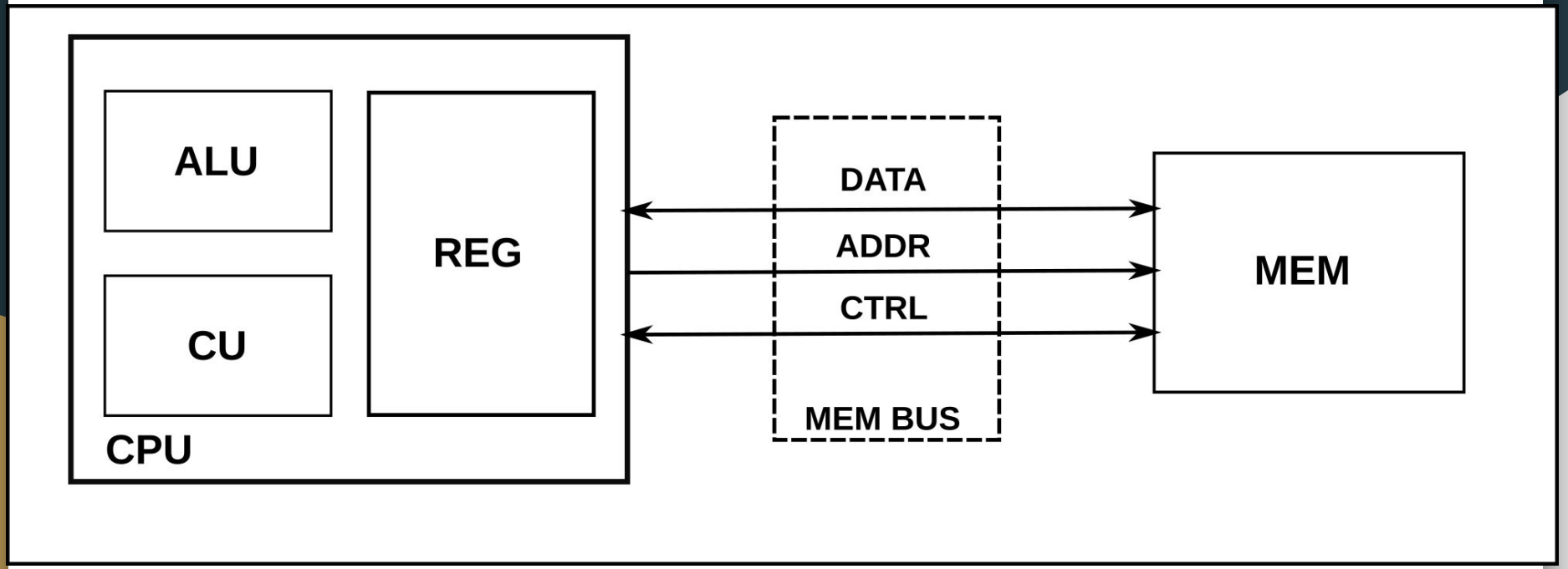
- Prvi računari: čitanje iz memorije i upis u memoriju može se obaviti u jednom ciklusu časovnika.
- Ovo je moguće za veoma male momorije, merene u bajtovima.
- Veće memorije su po pravilu sporije i potrebno je čekati nekoliko ciklusa časovnika.

HIJERARHIJA MEMORIJA

- I dalje je potrebno da imamo na raspolaganju određeni memorijski prostor kome možemo brzo pristupiti.
- Mali broj registrar blizu ALU jedinice.
- Čuvaju podatke sa kojima se trenutno radi i međurezultate.
- U današnjim računarima imamo čitavu hijerarhiju memorija.

Struktura računarskog sistema

- Komponente koje izvršavaju program: kontrolna jedinica i ALU jedinica
- Centralna procesorka jedinica (CPU): komponente koje izvršavaju program + registri
- Operativna memorija (RAM)
- Memorijska magistrala:
    - magistrale podataka
    - adresna magistrala
    - kontrolna magistrala

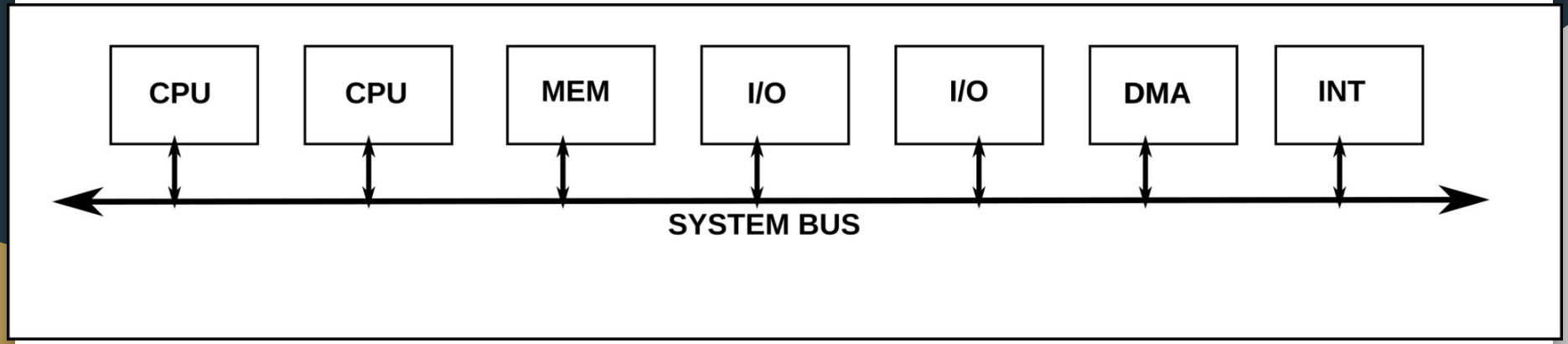

ALU
CU
REG
CPU
DATA
ADDR
CTRL
MEM BUS
MEM

Registri

- Brze komponente
- U početku većina registara su bili specijalne namene:
  - programski brojač, instrukcioni registar, itd..
- Danas je uglavnom veći broj registara opšte namene
- Njihov broj nije veliki:
  - prvi računari su imali samo akumulator
  - savremeni računari: uglavnom 16, 32

Ulazno izlazni (U/I) uređaji

- Savremeni računari imaju ulazno–izlazne uređaje
- Povezuju se putem U/I kontrolera (u okviru kojih se nalaze registri) i oni se putem magistrale povezuju sa CPU
- Najednostavniji model: SISTEMSKA MAGISTRALA
- U jednom trenutku = Jedan transfer


CPU
CPU
MEM
I/O
I/O
DMA
INT
SYSTEM BUS

ADRESNI PROSTOR

- Adresni prostor predstavlja celokupan opseg adresa koje procesor može da koristi (tj. koje može da postavi na adresnu magistralu).
- Npr. ako su adrese 32-bitne, tada procesor ima na raspolaganju opseg adresa od 0 do 2^32 - 1 (4GB prostora)
- Kako bi procesor mogao da razlikuje U/I kontrolere od memorije, registrima U/I kontrolera se moraju dodeliti posebne adrese koje su disjunktne sa opsegom adresa koje su dodeljene memorijskim lokacijama.
- Procesor sada zadaje adresu, a uređaj u čijem je opsegu ta adresa se prepoznaje i odgovara na zahtev procesora.

ARHITEKTURA RAČUNARA

- Arhitekturu skupa instrukcija (engl. *instruction set architecture (ISA)*)
  - skup registara, skup instrukcija, formati instrukcija, načini adresiranja operanada, veličina memorijskog prostora (virtuelnog i stvarnog), režimi rada, sistem prekida itd.
- Određuje iz kojih funkcionalnih jedinica se računar sastoji:
  - procesor, memorija, magistrale, DMA kontroler, U/I uredjaji, i sl.

ORGANIZACIJA RACUNARA

- MIKROARHITEKTURA
- Odnosi na aspekte implementacije zadate arhitekture
- Dizajn kontrolne ili ALU jedinice spada
- Način implementacije memorije, U/I kontrolera ili magistrale

MAŠINSKE INSTRUKCIJE

- Svaka mašinska instrukcija je niz bitova koji se strukturno može podeliti na OPERACIONI KOD (kodira koju operaciju instrukcija kodira) i OPERANDE (kodira na koje podatke se primenjuje operacija).
- Instrukcija moze imati nula ili više operanada, ali retko više od 3.

| OP KOD | OPR1 | OPR2 | OPR3 |

- FORMAT INSTRUKCIJE određuje način kodiranja gornjih komponenti u binarnom obliku u mašinskoj instrukciji.

MAŠINSKE INSTRUKCIJE

- Proces tumačenja binarnog koda mašinske instrukcije naziva se DEKODIRANJE INSTRUKCIJE.
- Prema vrsti, operandi mogu bit:
  - REGISTARSKI (podatak se nalazi u registru procesora)
  - MEMORIJSKI (podatak se nalazi u memoriji računara)
  - NEPOSREDNI (podatak se kodira kao deo same instrukcije)

VRSTE ARHITEKTURA (PREMA BROJU OPERANADA)

- TROADRESNI – odvojeni izvorišni i odredišni operandi.
- Savremene RISC arhitekture su tipično troadresne.

PRIMER: x = a*b + c*d

```
mul t, a, b

mul s, c, d

add x, t, s
```

VRSTE ARHITEKTURA (PREMA BROJU OPERANADA)

- DVOADRESNI – prvi operand je i odredišni.
- Tipičan predstavnik dvoadresne arhitekture je Intelova arhitektura.

PRIMER: x = a*b + c*d

```
mov t, a

mul t, b

mov s, c

mul s, d

mov x, t

add x, s
```

VRSTE ARHITEKTURA (PREMA BROJU OPERANADA)

- JEDNOADRESNI – postoji registar koji se zove akumulator.
- Jednoadresni računari su bili karakteristični za neke ranije generacije računara, dok su savremene arhitekture obično dvoadresne ili troadresne.

PRIMER: x = a*b + c*d

```
load a
mul b
store t
load c
mul d
add t
store x
```

VRSTE ARHITEKTURA (PREMA BROJU OPERANADA)

- NULOADRESNI – postoji stek, a operandi su implicitni.
- Nuloadresne arhitekture se zovu i STEK MAŠINE. Primer stek mašine je matematicki koprocesor Intel-ove arhitekture (poznat i kao x87).

PRIMER: x = a*b + c*d

```
push a
push b
mul
push c
push d
mul
add
pop x
```

MAŠINSKE INSTRUKCIJE

- PROBLEM: Kod dvoadresnih i troadresnih računara instrukcije postaju duže, pa je za učitavanje instrukcija potrebno više memorijskih pristupa.
- RESENJE1: Ograničavanje broja memorijskih operanada (Intelova arhitektura dozvoljava najviše jedan memorijski operand u instrukciji).
- RESENJE2: Postoje posebne instrukcije kojima se vrsi transfer podataka izmedju memorije i procesora i ove instrukcije su jedine instrukcije koje mogu imati memorijski operand. Ostale instrukcije (npr. aritmeticko-logicke) imaju iskljucivo registarske operande. Ovakve arhitekture zovemo Load-Store arhitekture.

NAČINI ADRESIRANJA OPERANADA

- IMPLICITNO ADRESIRANJE
  - Instrukcija ne sadrži u svom kodu nikakvu informaciju o lokaciji operanda, već se operand nalazi na lokaciji koja je implicitno određena (npr. uvek u istom registru).
- REGISTARSKO ADRESIRANJE
  - Instrukcija sadrži binarni kod registra čija se vrednost koristi kao operand.
- NEPOSREDNO ADRESIRANJE
  - Instrukcija sadrži binarni kod konstantne vrednosti koja se koristi kao operand.
  - Ne moraju sve konstante biti dozvoljene, kako bi se uštedelo na bitovima.

NAČINI ADRESIRANJA OPERANADA

- DIREKTNO ADRESIRANJE
  - Instrukcija sadrži apsolutnu adresu memorijskog operanda.
- RELATIVNO ADRESIRANJE
  - Instrukcija sadrži relativnu adresu operanda u odnosu na adresu tekuće instrukcije (tj. u odnosu na vrednost PC registra).
- BAZA + POMERAJ
  - Instrukcija sadrži binarni kod registra procesora (BAZNOG REGISTRA) i binarni kod konstante (POMERAJ). Vrednost kodiranog registra se sabira sa konstantom i tako se dobija adresa memorijskog operanda.
  - Korisno za pristup podacima na steku (pomeraj u odnosu na vrh steka).

NAČINI ADRESIRANJA OPERANADA

- INDIREKTNO ADRESIRANJE:
  - Instrukcija sadrži adresu na kojoj se nalazi adresa memorijskog operanda.
  - Često, adresa ne mora biti u memoriji, već u nekom od registara procesora.
  - U tom slucaju, instrukcija sadrži binarni kod registra procesora čija se vrednost koristi kao adresa memorijskog operanda.
  - Ovakva varijanta indirektnog adresiranja se naziva i REGISTARSKO INDIREKTNO ADRESIRANJE. Ono je u savremenim računarima mnogo češće od klasicnog indirektnog adresiranja, jer je efikasnije (ne moramo ići u memoriju po adresu).
  - Indirektno adresiranje je korisno kada adresa operanda nije fiksirana i poznata u fazi prevođenja (npr, kod dereferenciranja pokazivača, iteracije kroz niz, i sl.)

NAČINI ADRESIRANJA OPERANADA

- INDEKSNO ADRESIRANJE (BAZA + INDEKS)
  - Instrukcija sadrži binarne kodove dva registra čije se vrednosti sabiraju i tako dobijamo adresu memorijskog operanda.
  - Obično je vrednost jednog registra fiksirana, a drugi predstavlja indeks koji se pomera.
  - Korisno za pristup elementima niza.
  - Vrednost indeksnog registra se može množiti konstantom (npr. 2, 4 ili 8) pre sabiranja sa baznim registrom. Ovo je SKALIRANO INDEKSNO ADRESIRANJE.
  - Bazna adresa ne mora biti u registru, već može biti zadata kao apsolutna adresa na koju se dodaje vrednost indeksnog registra (uz eventualno prethodno skaliranje). Ovo je APSOLUTNO INDEKSNO ADRESIRANJE.

NAČINI ADRESIRANJA OPERANADA

- BAZA + FAKTOR*INDEKS + POMERAJ
  - Instrukcija sadrži binarne kodove dva registra (baznog i indeksnog), kao i binarni kod konstante koja predstavlja pomeraj. Indeksni registar može biti i skaliran. Adresa se dobija tako što se saberu vrednosti baznog i indeksnog registra (uz eventualno skaliranje indeksnog registra) i vrednosti pomeraja.
  - Ovaj način adresiranje uopštava indeksno, baza + pomeraj, direktno i indirektno adresiranje.

TIPOVI INSTRUKCIJA

- INSTRUKCIJE TRANSFERA
  - Kopiraju podatke sa jedne na drugu lokaciju (u procesoru ili memoriji)
  - Ponekad postoje odvojene instrukcije za transfer između registara, i za transfer između procesora i memorije (tipicno za Load/Store arhitekture)
  - Instrukcije za rad sa stekom su poseban vid ovih instrukcija
- ARITMETICKO LOGICKE INSTRUKCIJE
  - Sabiranje i oduzimanje
  - Množenje (rezultat može imati duplo više bitova)
  - Deljenje (daje količnik i ostatak)
  - Poređenje
  - Bitovske instrukcije

TIPOVI INSTRUKCIJA

- INSTRUKCIJE KONTROLE TOKA
  - Bezuslovni skok
  - Uslovni skokovi
  - Pozivi procedura i vraćanje iz njih
- SISTEMSKE INSTRUKCIJE
  - Instrukcije za promenu režima rada
  - Instrukcije za promenu nivoa privilegija
  - Instrukcije za izazivanje softverskih prekida
  - ...
- ULAZNO/IZLAZNE INSTRUKCIJE

O POZIVANJU PROCEDURA I SISTEMSKOM STEKU

```
function h() {
        …
}

function g() {
        …
        h()
        …
}


function f() {
        ….
        g()
        …
}
```

- Funkcija koja je poslednja pozvana prva će se zavrsiti.
- Dealocirati podatke funkcije koja je poslednja pozvana.
- Prirodno se nameće stek.
- Poziv funkcije je sličan običnom bezuslovnom skoku, s tom razlikom da se pre skoka mora zapamtiti adresa povratka (tj. adresa instrukcije koja sledi neposredno nakon instrukcije skoka).
- Povratne adrese prirodno da se čuvaju na steku.

O POZIVANJU PROCEDURA I SISTEMSKOM STEKU

- Stek se po pravilu nalazi u operativnoj memoriji.
- Na Intel-ovim procesorima podrazumeva se da se stek nalazi na visokim adresama u RAM-u, kao i da raste ka nižim adresama.
- Za rad sa stekom neophodno je znati gde se u memoriji u svakom trenutku nalazi vrh steka.
- U ovu svrhu se može koristiti neki od registara procesora.
- Većina modernih procesora ima poseban registar koji služi specijalno za tu namenu, kao i posebne instrukcije poput PUSH i POP koje rade sa ovim registrom.

O POZIVANJU PROCEDURA I SISTEMSKOM STEKU

- Alternativa korišćenju steka prilikom prenosa argumenata i čuvanja povratne adrese je da se u tu svrhu koriste registri procesora.
- Prednost ovog pristupa je u brzini.
- Ovakav pristup zahteva da procesor ima dovoljan broj registara.
- Problem kada imamo lanac poziva.
- U praksi su veoma česte funkcije koje sasvim jednostavne i koje ne pozivaju ni jednu drugu funkciju.

O RISC I CISC ARHITEKTURAMA

- RISC (reduced instruction set computer) je naziv koji označava arhitekture čiji se skup instrukcija sastoji iz relativno malog broja krajnje jednostavnih instrukcija koje obavljaju samo osnovne jednostavne operacije.
- Složenije aritmetičke operacije se moraju implementirati softverski, svođenjem na ove jednostavnije.
- Karakteristično da imaju uniformisani format instrukcija, kao i veoma jednostavne načine adresiranja operanada.
- Tipično se realizuju kao Load-Store arhitekture.
- RISC arhitekture takođe imaju veći broj registara opšte namene.

O RISC I CISC ARHITEKTURAMA

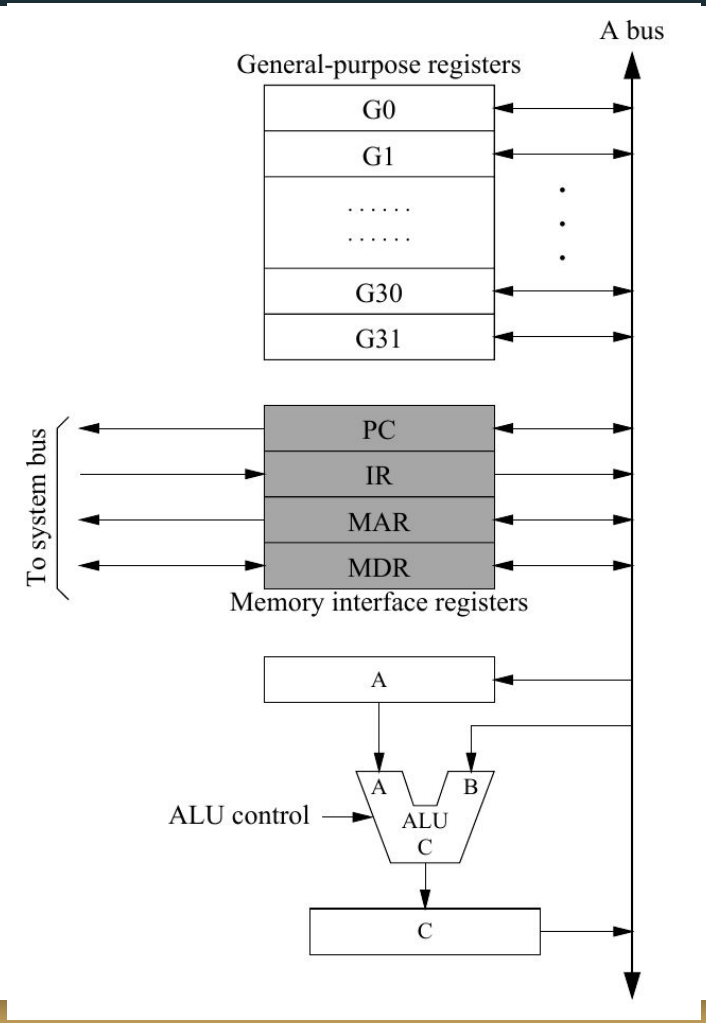
- CISC (complex instruction set computer) je naziv koji označava arhitekture čiji skup instrukcija može sadržati i instrukcije koje obavljaju prilično složene operacije (poput korena, sinusa, logaritma, i sl.).
- Zahtevaju izračunavanje u velikom broju ciklusa unutar samog procesora, svođenjem na jednostavnije koje direktno podržava ALU jedinica.
- Otuda je organizacija ovakvih procesora po pravilu mnogo složenija.
- CISC procesori takođe podržavaju veliki broj različitih formata instrukcija kao i veći broj složenih načina adresiranja.
- Ista instrukcija može imati više varijanti, u zavisnosti od tipa operanada.
- Mogućnost aritmetickih instrukcija da koriste podatke direktno iz memorije je tipična karakteristika CISC arhitektura.
- CISC arhitekture su imale relativno mali broj registara opšte namene.

O RISC I CISC ARHITEKTURAMA

- RISC zahteva duže programe.
- CISC arhitekture su bile veoma popularne 60-ih i 70-ih godina.
- Moderni procesori imaju sve veći broj registara što je neophodno kod load-store arhitekture.
- RISC arhitektura:
  - jednostavna implementacija procesora, → izvršavanje instrukcija veoma brzo
  - olakšava realizaciju nekih naprednih tehnika paralelizacije (poput tehnike preklapanja, engl. *pipelining*)
- CISC arhitekture:
  - glomaznije → sporije izvršavanje instrukcija
- Tipičan prestavnik CISC arhitekture je Intel
- Tipičan prestavnik RISC arhitekture je AMD

PUTANJA PODATAKA (engl. *datapath*)

- Podaci se kroz procesor krecu izmedju registara i ALU jedinice
- Izršavanje elementarnih operacija koje hardver direktno podržava
  - transfer podataka između registara
  - aritmetičko – logičke operacije koje se izvršavaju u ALU
- Kontrolu toka podataka na putanji podataka vrsi KONTROLNA JEDINICA (engl. *control unit, CU*)

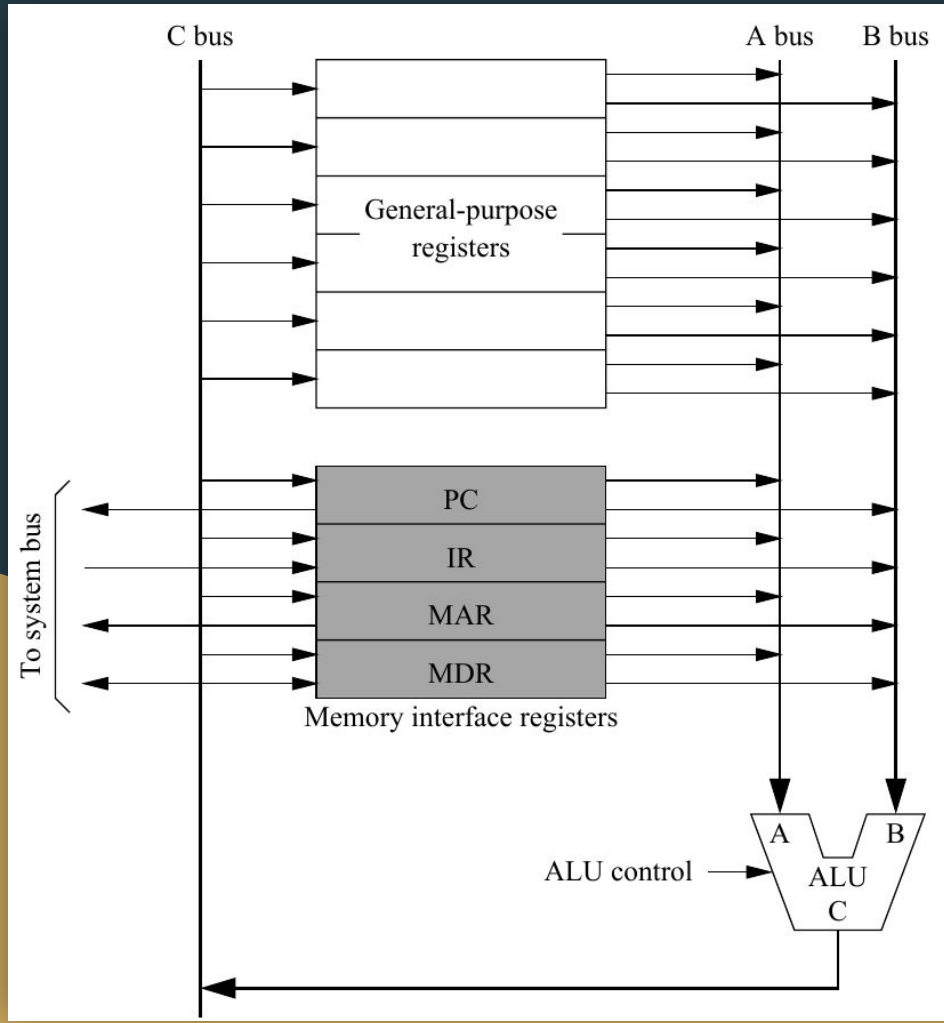

A bus
General-purpose registers
G0
G1
G30
G31
PC
IR
MAR
MDR
To system bus
Memory interface registers
A
ALU control
A
B
ALU
C
C

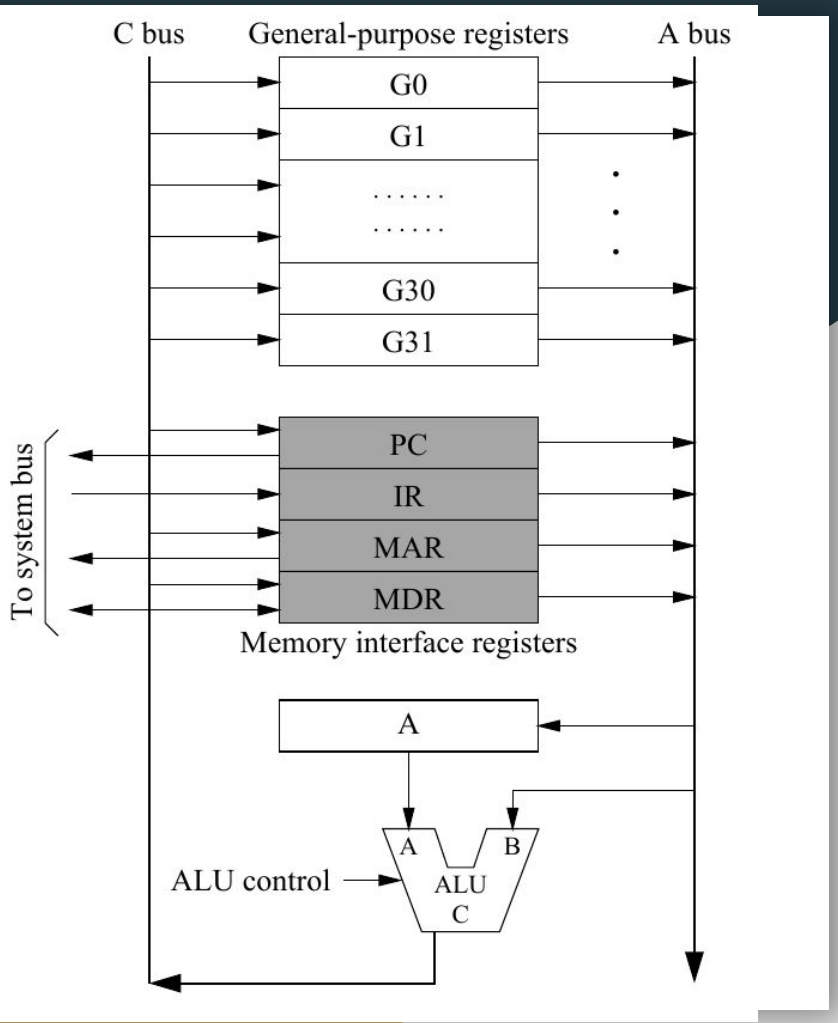
PUTANJA PODATAKA

- Transfer podataka između registara procesora i lokacija van procesora se vrši pomoću magistrala.
- Te magistrale funkcionišu na složeniji način u odnosu na magistrale procesora.
- Unutar procesora, transfer se obično može obaviti u toku jednog ciklusa časovnika.
- To nije slučaj sa komponentama van procesora.
- PROTOKOL MAGISTRALE: skup pravila po kojima magistrala funkcioniše
- Putanja podataka magistralama koje se nalaze unutar procesora mora biti usaglašena sa spoljašnjim magistralama.

PUTANJA PODATAKA

- Veći broj internih magistrala → više transfera u svakom ciklusu
- Npr. u slučaju sabiranja
  - tri magistrale u jednom ciklusu je moguće dovesti podatke do ALU jedinice i prebaciti rezultat do odredišta.
  - dve magistrale: dva ciklusa
  - jedna magistrala: tri ciklusa
- ALI, više magistrala čini i više kontrolnih signala od strane CU
- Današnji računari imaju više magistrala i prilično složene putanje podataka.


C bus
A bus
B bus
General-purpose registers
PC
IR
MAR
MDR
To system bus
Memory interface registers
A
B
ALU
C
ALU control


C bus
General-purpose registers
A bus
G0
G1
G30
G31
PC
IR
MAR
MDR
To system bus
Memory interface registers
A
B
ALU
C
ALU control

INTERNI REGISTRI PROCESORA

- Procesori obično sadrže i neke registre koji služe za implementaciju internih funkcionalnosti procesora i nisu deo arhitekture, tj. nisu vidljivi programeru, ne može ih koristiti.
- PC: programski brojac
  - sadrži memorijsku adresu instrukcije koja se trenutno izvršava
  - nakon završetka tekuće instrukcije automatski uvećava za odgovarajući broj bajtova i pokazuje na sledeću instrukciju
- IR: instrukcioni registar
  - sadrži operacioni kod i ostale komponente instrukcije dobijene nakon dekodiranja
  - vrednost registra se sprovodi do CU da bi CU znao šta dalje da radi

INTERNI REGISTRI PROCESORA

- MAR: adresni registar
  - njegov izlaz je povezan na adresnu magistralu
  - sadrši adresu sa koje se čita ili gde se upisuje podatak
- MDR: prihvatni registar memorije
  - povezan na magistralu podataka
- PSW: statusni registar (engl. process status word)
  - sadrži FLEGOVE koji opisuju stanje rezultata prethodne instrukcije
  - stanje ovog registra se izračunava u ALU jedinici, a prosleđuje se CU jedinici

INTERNI REGISTRI PROCESORA

- CW: kontrolni registar
  - sadrži FLEGOVE i isto kao i PSW utiče na buduće ponašanje procesora
  - često su ova dva registra spojena u jedan

FAZE IZVRŠAVANJA INSTRUKCIJE

- Izvršavanje svake instrukcije je u fazama i u okviru svake faze se generišu kontrolni signali.
- DOHVATANJE INSTRUKCIJE (FETCH)
  - Instrukcija se učitava u registar sa adrese na koju pokazuje PC registar i smešta se u MDR registar.
  - Vrednost iz PC registra se prvo prebaci u MAR registar i onda se izda signal za čitanje.
  - Ovaj postupak može zahtevati i više ciklusa časovnika.
- DEKODIRANJE INSTRUKCIJE (DECODE)
  - Instrukcija se prebaciju u IR registar
  - Instrukcija se prosleđuje CU

FAZE IZVRŠAVANJA INSTRUKCIJE

- IZVRSAVANJE INSTRUKCIJE (EXECUTE)
  - ova faza može zahtevati više ciklusa
  - ako instrukcija zahteva i memorijske operande, vrši se i dohvatanje operanada iz memorije (adresa operanda ide u MAR registar, a onda se sadržaj prvo prebaciju u MDR registar)
  - u slučaju transfera između registara, vrednost izvorišnog registra se propušta kroz ALU, ali se ne vrši nikakva operacija
  - u slučaju skoka: ako flegovi ukazuju da je uslov ispunjen, adresa skoka se upisuje u PC registar, inače se ne vrši ništa

FAZE IZVRŠAVANJA INSTRUKCIJE

- UPIS REZULTATA (WRITE)
  - odredište rezultata može biti registar ili memorija.
  - ako je memorija - tada se ponovo obavlja transfer preko magistrale (MAR i MDR)
  - memoriji se šalje signal za upis.
- OBRADA PREKIDA (INTERRUPT)
  - provera da li je došao neki spoljašnji signal za prekid
  - sačuvati stanje procesora i izvršiti obradu prekida

NAČINI IMPLEMENTACIJE KONTROLNE JEDINICE

- TVRDO OZICENA IMPLEMENTACIJA (engl. *hardwired control unit*)
- "hardverska implementacija"
- Dizajnira se kao sekvencijalno kolo.
- Na ulazu se nalazi vrednosti IR i PSW registara, a na izlazu skup kontrolnih signala.
- Stanja odgovaraju fazama izvršavanja.
- Mogu postojati i međustanja, kod komplikovanijih instrukcija (npr. množenje ili transfer).
- Pogodna za RISC arhitekture.
- Nema fleksibilnosti → nova instrukcija, celo kolo se redizajnira.

NAČINI IMPLEMENTACIJE KONTROLNE JEDINICE

- MIKROPROGRAMIRANA IMPLEMENTACIJA (engl. *microprogrammed control unit*)
- Kontrolna jedinica sadrži memoriju koja se zove kontrolna memorija (engl. *control store*).
- Svaka lokacija — jedna mikroinstrukcija.
- Obično je ROM memorija (mada neki njeni delovi mogu biti i izmenjivi, sto je slučaj sa modernim Intel-ovim i AMD-ovim procesorima).

MIKROPROGRAMIRANA IMPLEMENTACIJA

- Svaka mikroinstrukcija se izvršava u jednom ciklusu.
- Adresni bitovi iz tekuće mikroinstrukcije se kombinuju na odgovarajući način sa vrednošću IR i PSW registra čime se dobija adresa sledeće mikroinstrukcije u kontrolnoj memoriji.
- MIR registar (*Micro Instruction Register*) sadrži tekuću mikroinstrukciju.
- Kontrolna memorija, kao i kolo za računanje adrese su kombinatorna kola.
- MIR je registar: sekvencijalno kolo.

MIKROPROGRAMIRANA IMPLEMENTACIJA

- MIKROPROGRAM – niz mikroinstrukcija u kontrolnoj memoriji.
- Svakoj instrukciji odgovara poseban mikroprogram koji izvršava tu instrukciju.
- PSW registar takodje utice na adresu, pojedine instrukcije se izvršavaju drugačije, u zavisnosti da li je uslov ispunjen ili ne (npr. instrukcije skoka).
- Pogodno za CISC arhitekture.
- Fleksibilnija, može se dodati nova instrukcija mikroprogramiranjem postojećih.
- Sporija od tvrdo ožičene.

MIKROPROGRAMIRANA IMPLEMENTACIJA

- Mikroinstrukcije mogu imati HORIZONTALNU i VERTIKALNU strukturu.
- Horizontalna struktura:
  - svi kontolni signali postoje kao pojedinačni bitovi mikroinstrukcije, i mogu se direktno usmeriti ka odgovarajućim komponentama procesora
- Vertikalna struktura:
  - ideja je da se dužina mikroinstrukcije smanji, da bi se smanjila memorija koju zauzima i učinila jefinijom
  - mikroinstrukcije su kompaktinije, ali je duži proces dekodiranja pa je zato sporije