

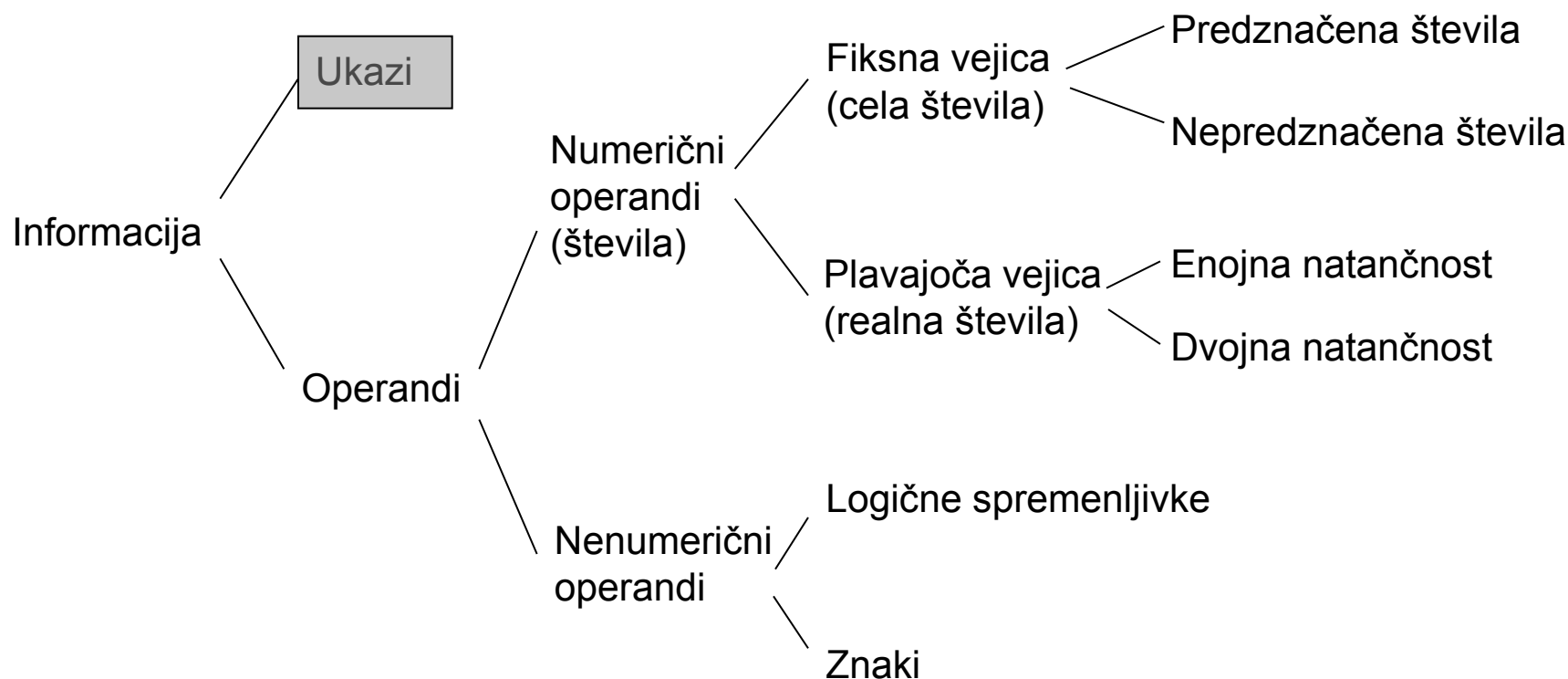


RAČUNALNIŠKA ARHITEKTURA

5 Ukazi



Osnovni tipi informacije v računalniku





Ukazi - vsebina:

- Splošno o ukazih
- Način shranjevanja operandov v CPE
 - Akumulator
 - Sklad
 - Množica registrov
- Število eksplicitnih operandov v ukazu
 - 3+1 - operandni računalniki
 - 3 - operandni računalniki
 - 2 - operandni računalniki
 - 1 - operandni računalniki
 - Brez - operandni računalniki
- Lokacija operandov in načini naslavljanja
 - Registrsko - registrski računalniki
 - Registrsko - pomnilniški računalniki
 - Pomnilniško - pomnilniški računalniki



- Takojšnje naslavljanje
- Neposredno naslavljanje
- Posredno naslavljanje
- Operacije (vrste ukazov)
 - Aritmetične in logične operacije (ALE operacije)
 - Prenosi podatkov
 - Kontrolne operacije
 - Operacije v plavajoči vejici
 - Sistemske operacije
 - Vhodno/izhodne operacije
- Vrsta in dolžina operandov
 - Sestavljeni pomnilniški operandi
 - Pravilo debelega konca
 - Pravilo tankega konca
 - Problem poravnosti
- Zgradba ukazov
- RISC – CISC računalniki



- Ukazi = Strojni ukazi (= ukazi običajnega strojnega jezika)
- Nabor ukazov \Rightarrow Arhitektura računalnika
ISA Instruction Set Architecture
Ukazna arhitektura
- Različni računalniki – različne arhitekture – različni strojni ukazi



- Delovanje von Neumannovega računalnika je popolnoma določeno z ukazi, ki jih CPE jemlje iz glavnega pomnilnika.
- Ti ukazi so strojni ukazi (oz. ukazi običajnega strojnega jezika).
- Z določitvijo množice strojnih ukazov dejansko v veliki meri izberemo arhitekturo računalnika.
- Zato govorimo o ukazni arhitekturi (angl. ISA – Instruction Set Architecture)



5.1 Splošno o ukazih

- Ukazi se izvršujejo v CPE, v uporabi sta dva načina za izvrševanje ukazov:
 - Uporaba trdo ožičene logike
 - Hitro izvajanje (logično vezje direktno izvaja strojne ukaze)
 - Težko spreminjanje in dodajanje novih ukazov (potrebno novo logično vezje v CPE \Rightarrow nov čip)
 - Mikroprogramiranje
 - Počasnejše izvajanje (potrebno interpretiranje na mikroprogramski nivo – logično vezje izvaja mikroukaze)
 - Lažje spreminjanje in dodajanje novih ukazov (samo sprememba v mikroprogramu)



- Vsak ukaz mora vsebovati informacijo dveh strogo ločenih vrst:
 - Informacijo o **operaciji**, ki naj se izvrši
 - Informacijo o **operandih**, nad katerimi se bo ta operacija izvršila

- Obe vrsti informacije sta opisani z biti v poljih, na katere je ukaz razdeljen - po dolžini (številu bitov) in številu teh polj so med ukazi različnih računalnikov velike razlike.



- **Operacijska koda** - je ime polja, ki vsebuje informacijo o operaciji.
- Polja, ki vsebujejo **informacijo o operandih**:
 - ☐ Lahko vsebujejo kar operand
 - ☐ Ali informacijo o naslovu, na katerem je operand shranjen
- Pri nekaterih ukazih je informacija o operandih vsebovana že v operacijski kodi.



- **Format ukaza** – določa razdelitev ukaza na polja z obema vrstama informacije, dolžine posameznih polj v bitih in pomen posameznih bitov v teh poljih.

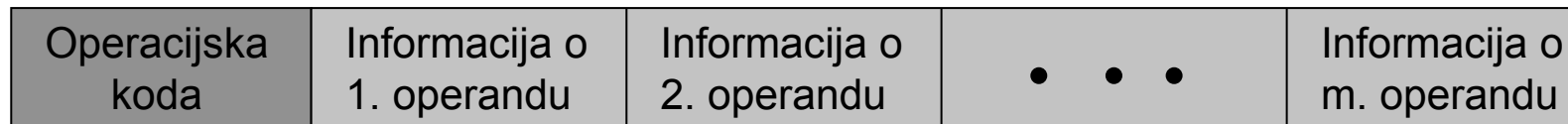
- Kakšen je format ukaza je odvisno od:
 - ☐ Števila operacij
 - ☐ Števila registrov v CPE
 - ☐ Dolžine pomnilniškega naslova
 - ☐ Dolžine pomnilniške besede
 - ☐ ...



- Format strojnega ukaza dolžine n – bitov, z m eksplicitno definiranimi operandi:

bit $n-1$

bit 0



Format strojnega ukaza dolžine n - bitov
z m - eksplicitno definiranimi operandi



- Razlikovanje med ukazi in operacijami - izvajanje iste operacije (npr. seštevanje) lahko dosežemo z več različnimi ukazi, ki imajo različne formate (običajno na različne načine podane informacije o operandih).
- Število ukazov je zato običajno precej večje kot število operacij.



- Osnovne lastnosti ukazov, po katerih se ukazi med seboj razlikujejo:
 - ☐ Način shranjevanja operandov v CPE
 - ☐ Število eksplicitnih operandov v ukazu
 - ☐ Lokacija operandov in načini naslavljanja
 - ☐ Operacije
 - ☐ Vrsta in dolžina operandov

- Odločitve pri vsaki od lastnosti vplivajo na zgradbo in delovanje računalnika.



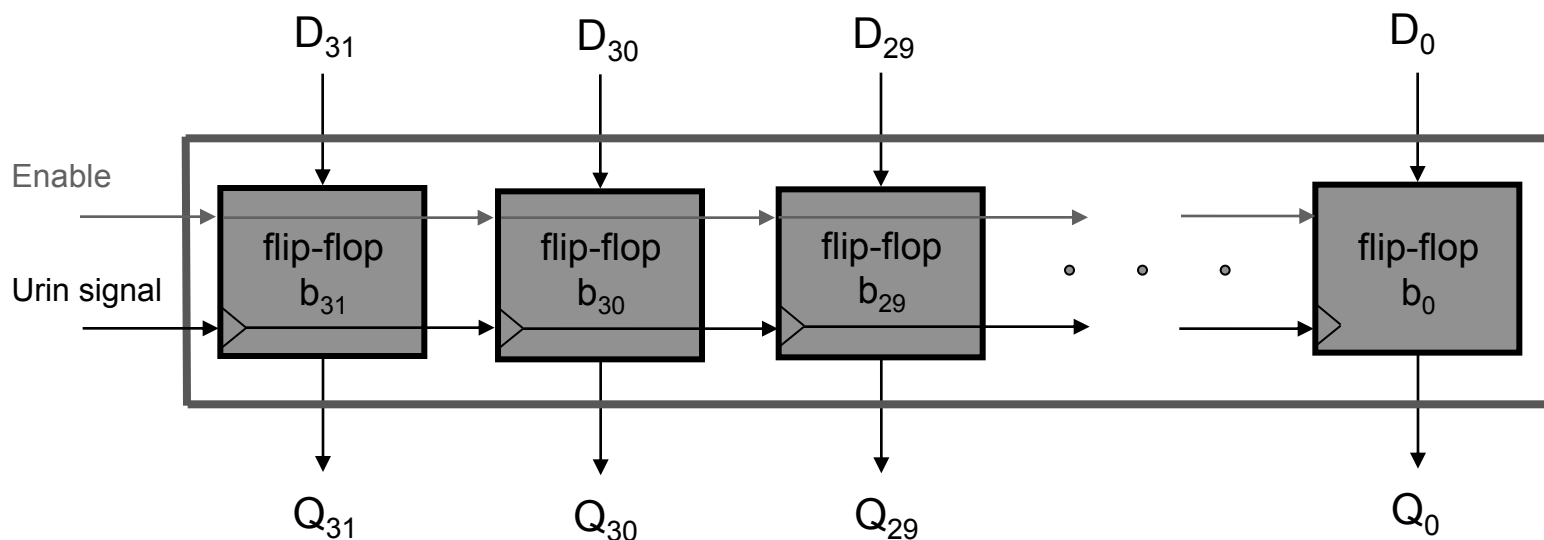
5.2 Načini shranjevanja operandov v CPE

- Lastnost, ki najbolj vpliva na to, kako uporabnik vidi računalnik.
- Pomembnejši so trije načini shranjevanja operandov v CPE:
 - Akumulator (en sam programsko dostopen register v CPE)
 - Sklad (v CPE)
 - Množica registrov (množica programsko dostopnih registrov v CPE)

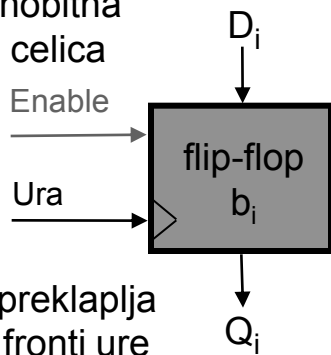


32-bitni register

b_{31}	b_{30}	b_{29}	\dots	b_1	b_0
----------	----------	----------	---------	-------	-------

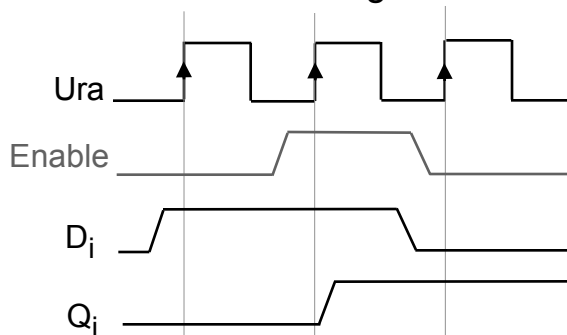


Flip-flop = enobitna pomnilniška celica



Flip-flop, ki preklaplja ob pozitivni fronti ure

Časovni diagram



Pravilnostna tabela

Ura	En	D_i	Q_i
↑	0	0	Q
↑	0	1	Q
↑	1	0	0
↑	1	1	1

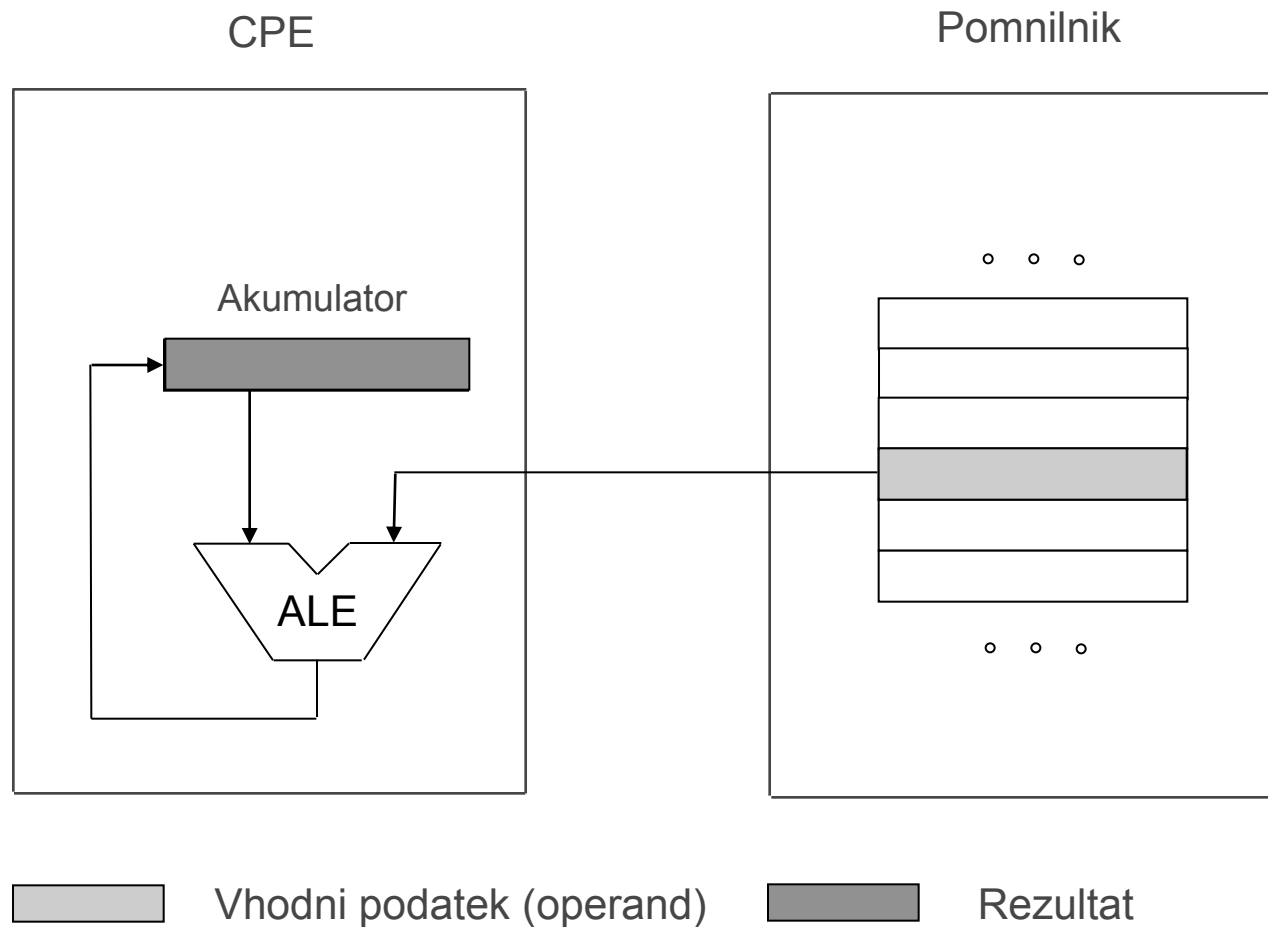


- **Akumulator:** Pomnilnik v CPE je en sam register, ki mu pravimo akumulator.
 - Vanj lahko shranimo en operand
 - Najstarejša rešitev, njena odlika je preprostost
 - Pri večini ukazov je eden od operandov v akumulatorju, tja se shrani tudi rezultat



Načini shranjevanja operandov v CPE - akumulator

Izvajanje ALE ukaza





Načini shranjevanja operandov v CPE - akumulator

- ☐ Ukaza za prenos operanda iz glavnega pomnilnika v akumulator ali obratno sta LOAD in STORE.
- ☐ Ker je akumulator en sam, v ukazih ni potrebno navajati njegovega naslova - krajši ukazi.
- ☐ Preprosti prevajalniki, ker ni potrebno odločanje med več možnostmi za shranjevanje operandov.
- ☐ Veliko prenosov med CPE in glavnim pomnilnikom (več kot pri drugih dveh rešitvah), ker je register en sam.

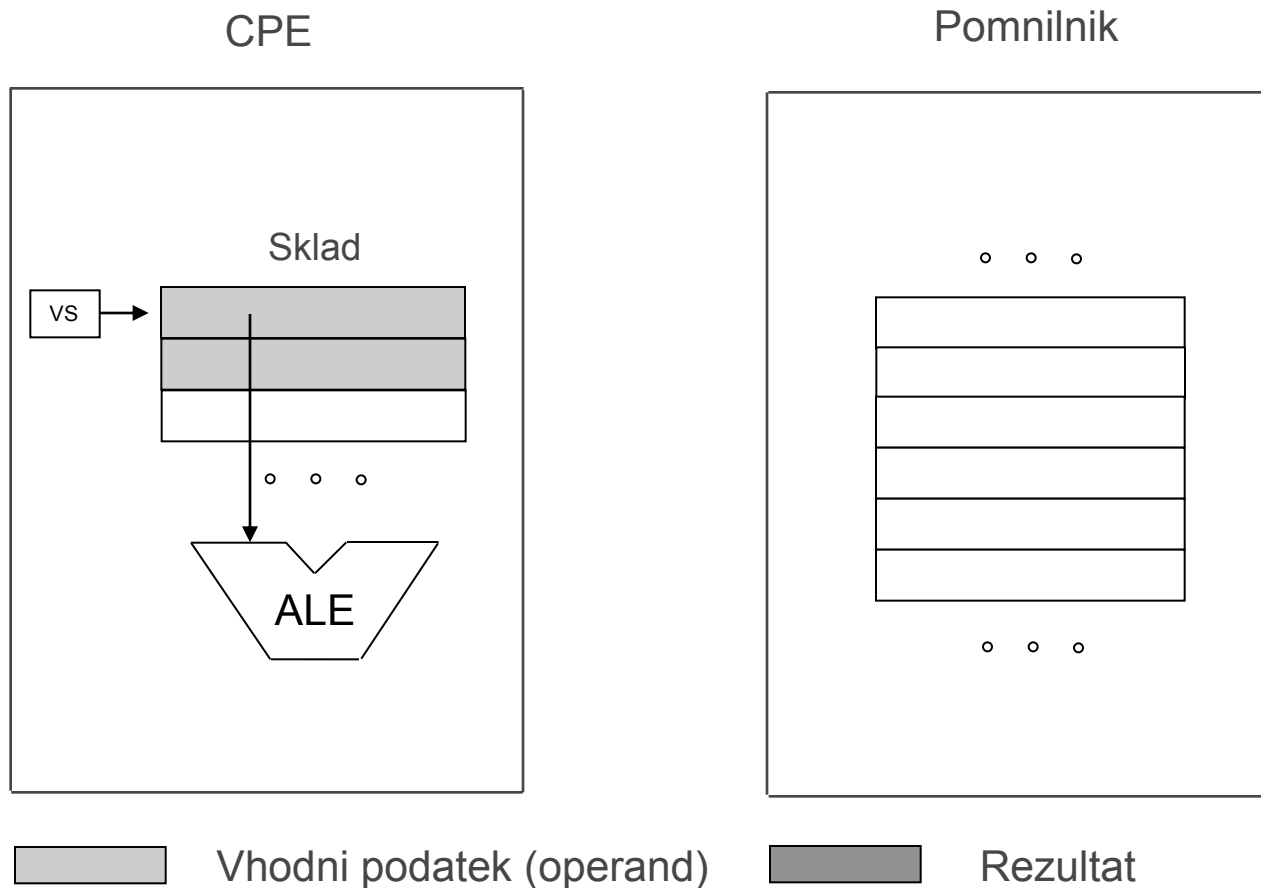


- **Sklad** (angl. stack): Enostaven način za povečanje pomnilnika v CPE je, da ga naredimo v obliki sklada.
 - Pri skladu je v vsakem trenutku dostopna samo najvišja lokacija - vrh sklada.
 - Način delovanja sklada označimo kot zadnji noter, prvi ven (angl. LIFO - Last In First Out).
 - Ukazi za prenos operanda iz glavnega pomnilnika v sklad in obratno so PUSH in PULL oziroma POP.
 - Velika podobnost z akumulatorjem. Vse prednosti akumulatorja veljajo tudi za sklad, lahko pa vanj shranimo več operandov.



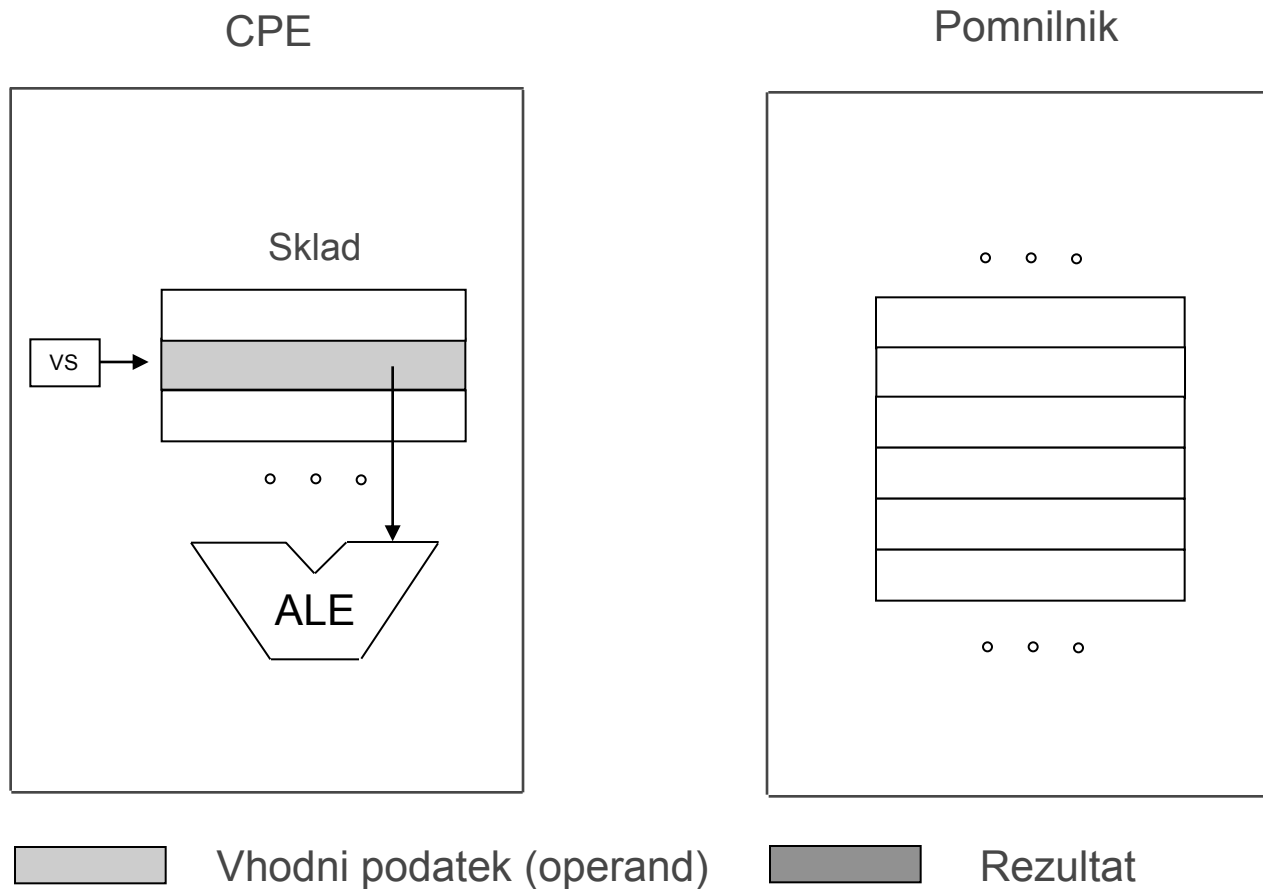
Načini shranjevanja operandov v CPE - sklad

VS – naslov vrha sklada v registru v CPE. V tem primeru kaže na zadnjo zasedeno lokacijo v skladu



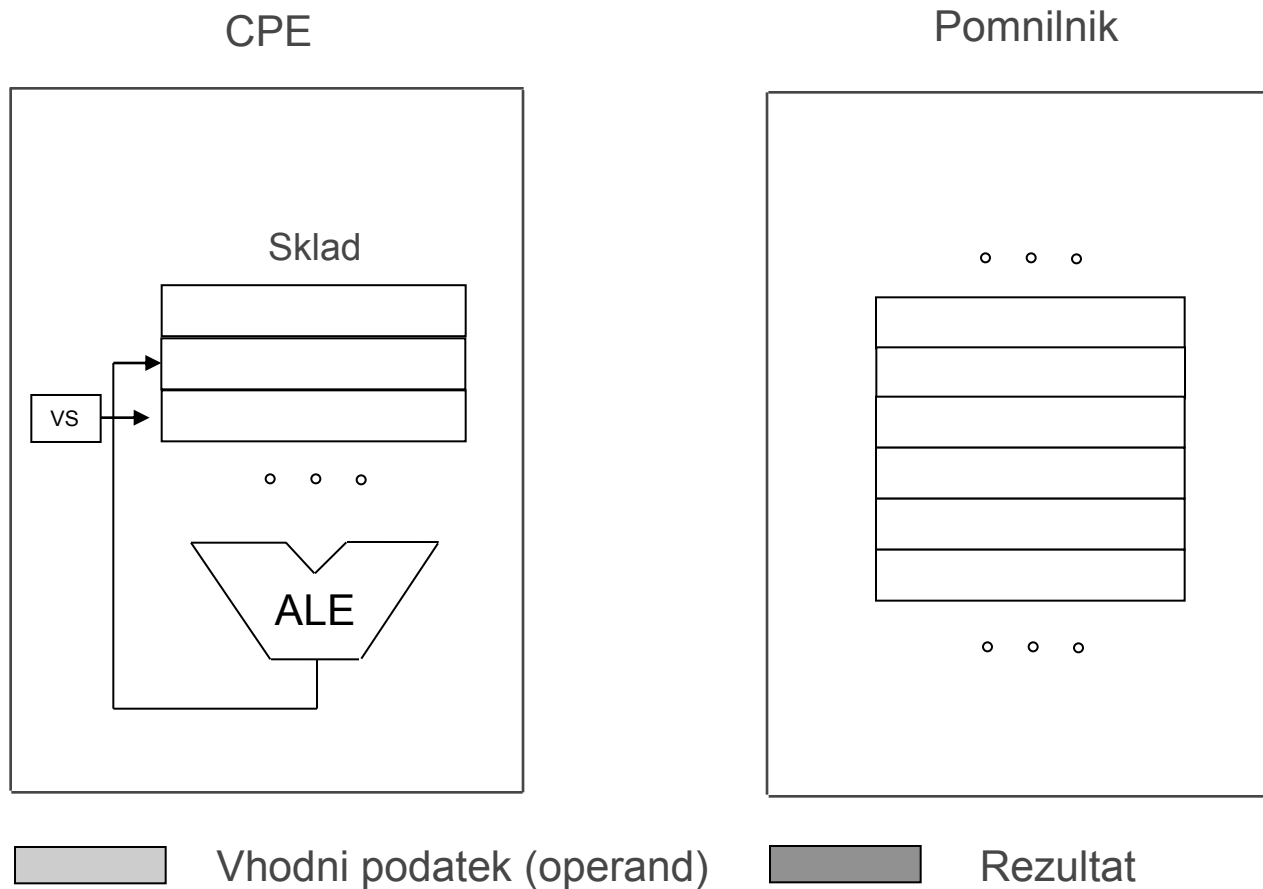


Načini shranjevanja operandov v CPE - sklad



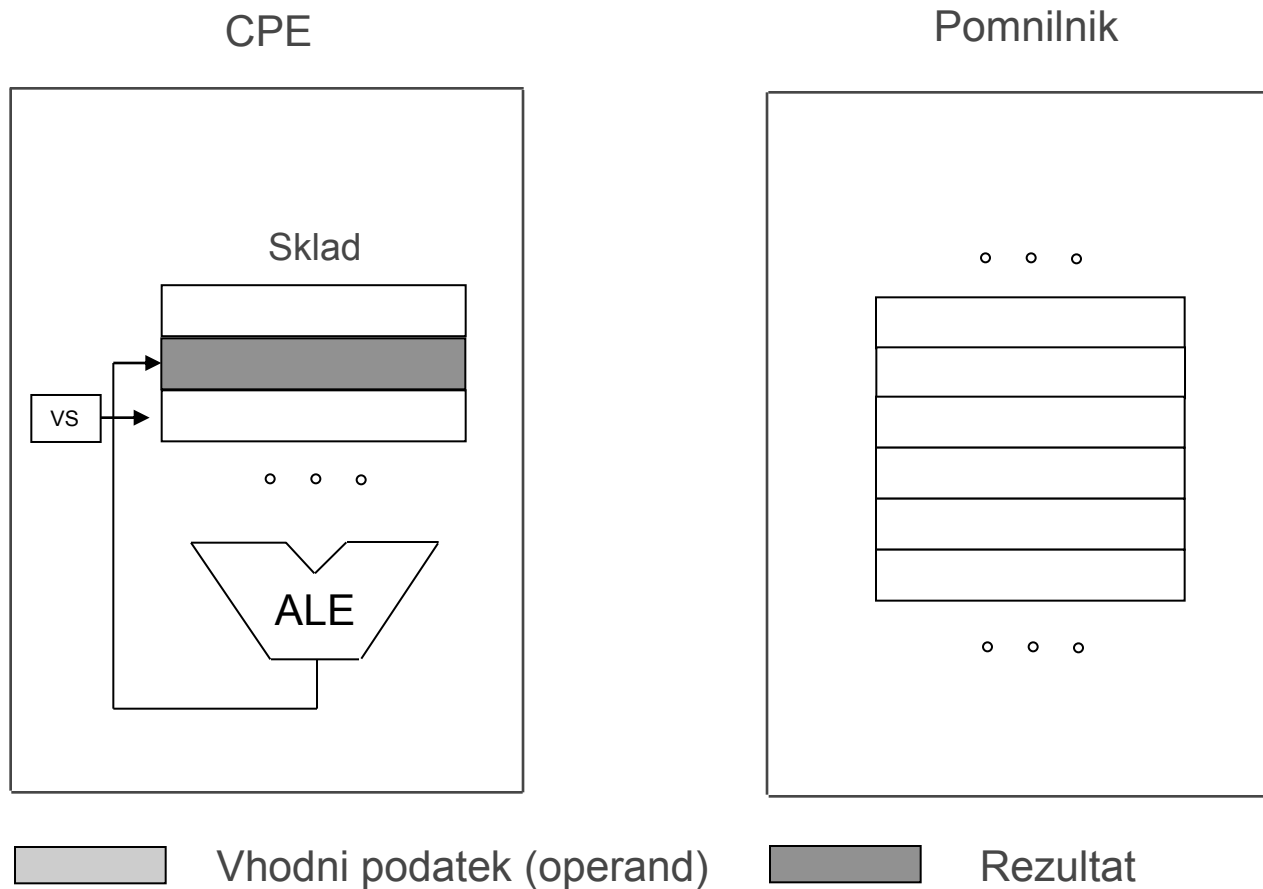


Načini shranjevanja operandov v CPE - sklad



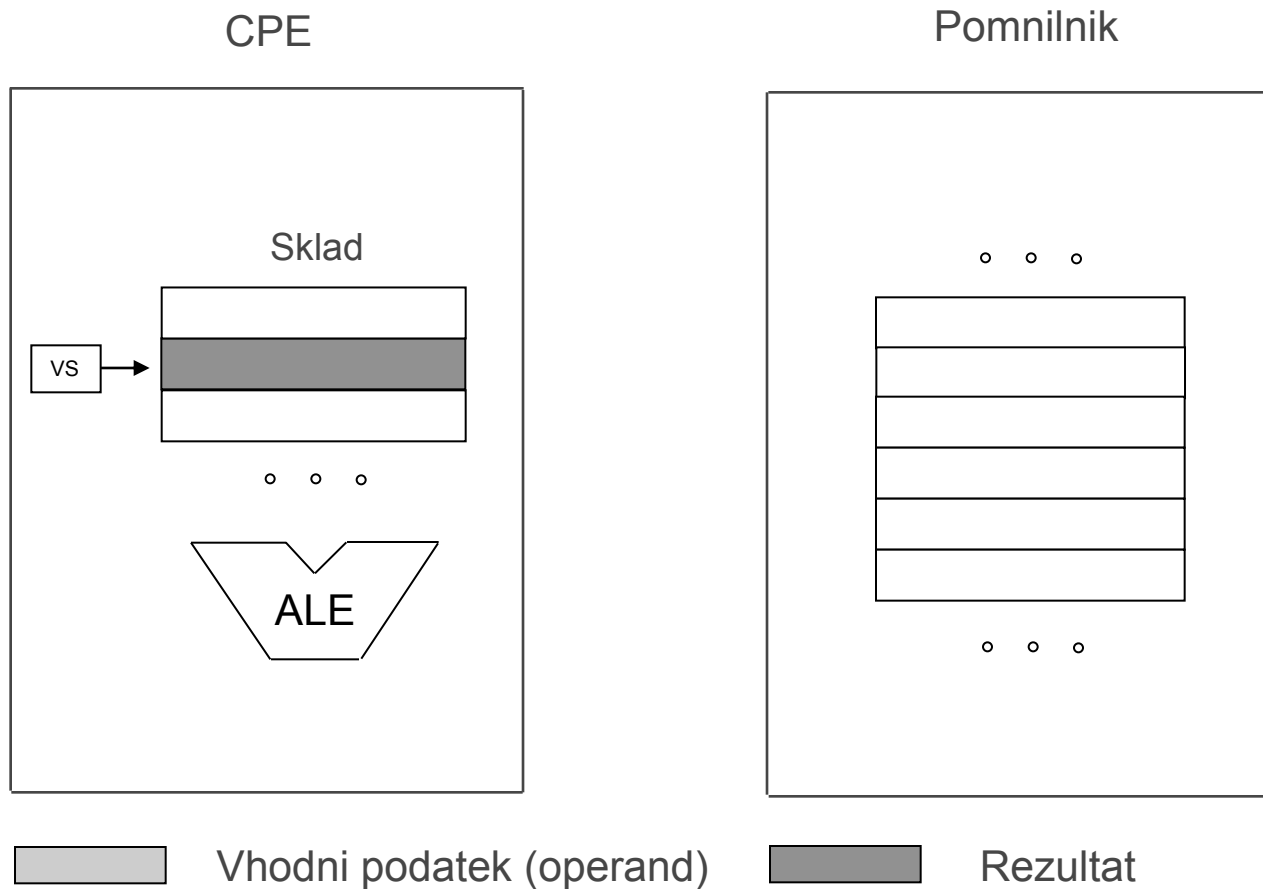


Načini shranjevanja operandov v CPE - sklad





Načini shranjevanja operandov v CPE - sklad





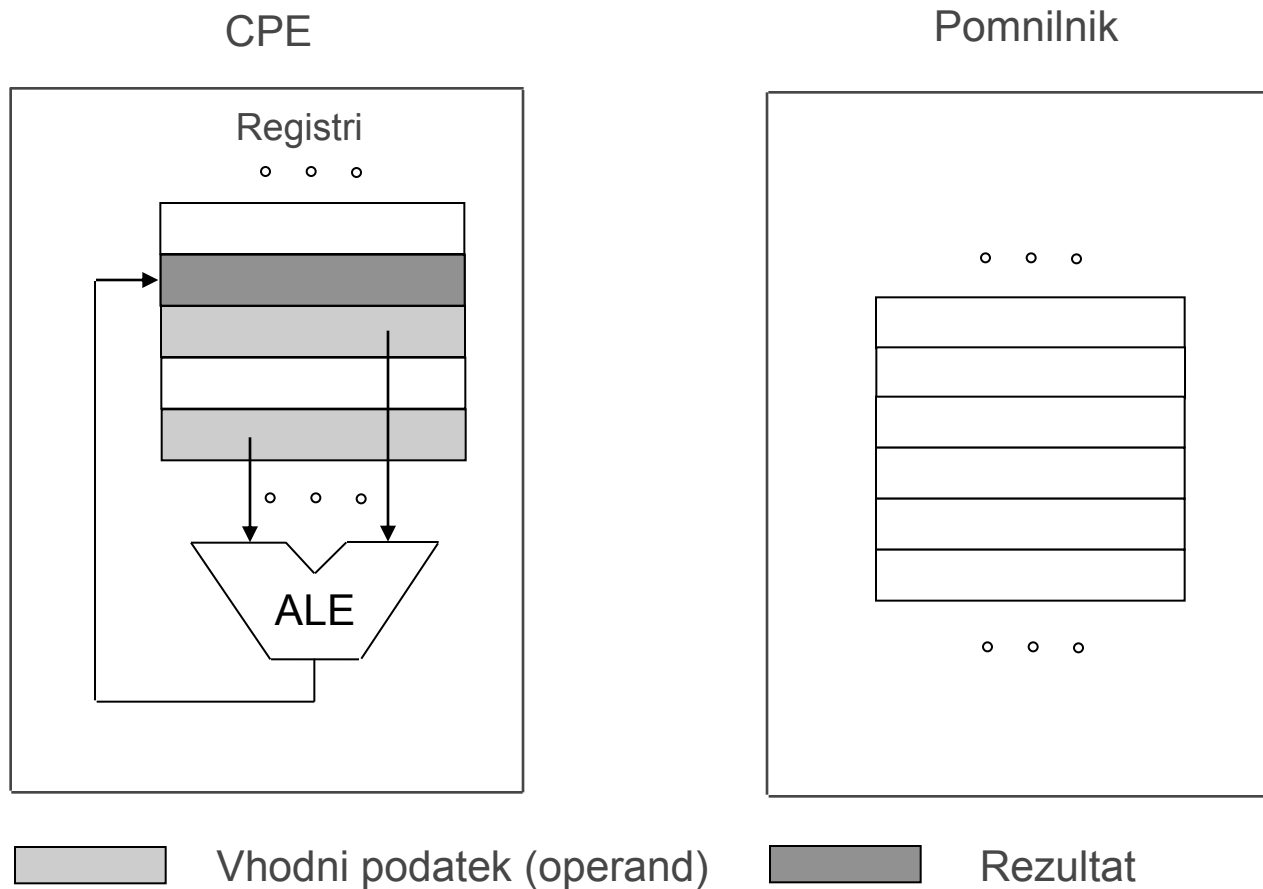
- ☐ Če beremo iz sklada, se vrh sklada pomakne za eno mesto navzdol in na vrhu sklada je nova vrednost.
- ☐ Pri pisanju v sklad se nova vrednost shrani v najvišjo lokacijo, ki postane vrh sklada.
- ☐ Pri dvo-operandnih operacijah (npr. seštevanje $A+B$), se operanda A in B prebereta iz sklada (A mora biti na vrhu sklada, B pa tik pod njim), rezultat pa se shrani na vrh sklada.
- ☐ Skladovni računalniki ali računalniki s skladovno arhitekturo so bili popularni okrog leta 1960.



- **Množica registrov** (angl. register set): Pomnilnik v CPE je narejen kot množica registrov, do katerih je možen dostop brez omejitev.
 - Število registrov v današnjih računalnikih je od 8 do 100 ali več.
 - Vsak register ima svoj naslov, podobno kot besede v glavnem pomnilniku. Za naslov registra je v ukazu potrebnih bistveno manj bitov, kot za pomnilniški naslov.
 - Razlikujemo dve rešitvi glede na svobodo uporabe registrov:
 - Vsi registri so ekvivalentni - splošnonamenski registri
 - Množica registrov je razdeljena na dve skupini. Ena se uporablja za aritmetično logične operande, druga za računanje z naslovi (bazni ali indeksni registri)



Načini shranjevanja operandov v CPE - množica registrov





- Prednosti programsko dostopnih registrov v CPE:
 - **Večja hitrost.** Ker je pomnilnik v CPE majhen, je lahko zgrajen v hitrejši tehnologiji kot glavni pomnilnik - krajši čas dostopa. Možno je dostopati tudi do več registrov hkrati.
 - **Krajši ukazi.** Ker je registrov v primerjavi z glavnim pomnilnikom malo, je za naslov registra v ukazu potrebnih manj bitov - krajša polja v ukazih za opis operandov.
 - **Zmanjša se promet** med CPE in glavnim pomnilnikom. Registri omogočajo shranjevanje vmesnih rezultatov (dokler seveda registrov ne zmanjka)



Načini shranjevanja operandov v CPE - množica registrov

- ☐ Za prevajalnike so bile razvite metode, ki omogočajo najboljšo možno uporabo registrov.
- ☐ Posledica: Vsi po letu 1980 razviti računalniki imajo v CPE pomnilnik v obliki množice registrov.
- Pri večini računalnikov je takoj vidno, katerega od treh načinov uporabljajo.

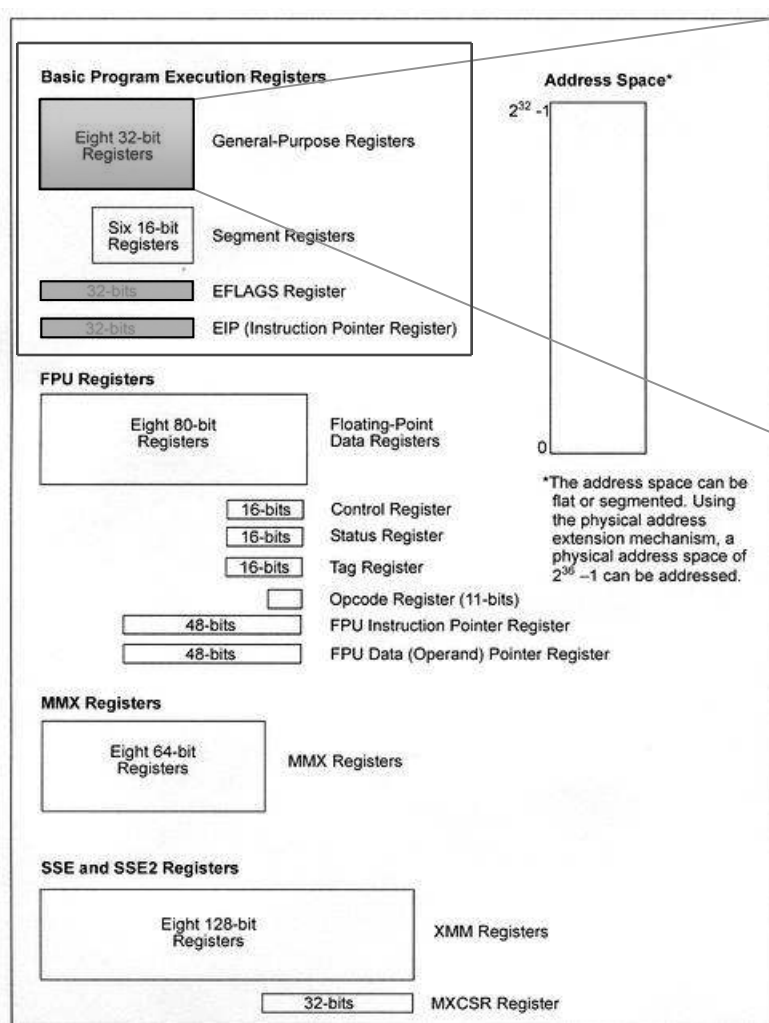


■ Nekateri računalniki so nekje vmes:

- ☐ Mikroprocesorji Intel 80x86 so v začetku (Intel 8086 - leta 1978) imeli en sam akumulator (splošnonamenski register) in nekaj dodatnih registrov za pomoč pri dostopu do operandov.
- ☐ Intel 80386 - leta 1985: 8 splošnonamenskih registrov.
- ☐ Pentium 4 - leta 2006: 16 splošnonamenskih registrov in 32 dodatnih registrov (FPU, MMX, XMM).



Registri procesorja Intel P4 v 32-bitnem načinu delovanja



General-Purpose Registers									
31	16	15	8	7	0	16-bit	32-bit		
			AH		AL	AX	EAX		
			BH		BL	BX	EBX		
			CH		CL	CX	ECX		
			DH		DL	DX	EDX		
			BP				EBP		
			SI				ESI		
			DI				EDI		
			SP				ESP		

EFLAGS

Register zastavic

EIP

Programski števec

EAX - akumulator

EBX - bazni register (kazalec na podatke v podatkovnem segmentu)

ECX - števeni reg. za operacije z nizi in zanke

EDX - V/I kazalec

EBP - bazni kazalec (kazalec na podatke v skladovnem segmentu)

ESI - indeksni register (kazalec na izvor)

EDI - indeksni register (kazalec na ponor)

ESP - kazalec na sklad



Programsko dostopni registri 8080 - 8-bitni procesor (leto 1974, 8085 leto 1977)

15	8	7	0	
A		PSW		Akumulator A in Register zastavic
B		C		Sekundarna akumulatorja/Števna registra
D		E		Sekundarna akumulatorja/Števna registra
H		L		Sekundarna akumulatorja/Števna registra
SP				Skladovni kazalec
PC				Programski števec (PC)



Programsko dostopni registri 8086 - 16-bitni procesor (leto 1978)

15	8	7	0	
AH		AL		Akumulator AX
BH		BL		Bazni register BX
CH		CL		Števni register (nizi, zanke) CX
DH		DL		Podatkovni register (množenje, deljenje) DX
BP				Bazni kazalec
SI				Indeksni register (izvor)
DI				Indeksni register (ponor)
SP				Skladovni kazalec
CS				Kazalec na ukazni segment
SS				Kazalec na skladovni segment
DS				Kazalec na podatkovni segment
ES				Kazalec na dod. podatkovni segment
IP				Programski števec (PC)
FLAGS				Pogojni biti



Programsko dostopni registri 80386 - 32-bitni procesor (leto 1985 \Rightarrow arhitektura x86)

31	16 15	8 7	0	
EAX	AX	AH	AL	Splošnonamenski register EAX
EBX	BX	BH	BL	Splošnonamenski register EBX
ECX	CX	CH	CL	Splošnonamenski register ECX
EDX	DX	DH	DL	Splošnonamenski register EDX
EBP	BP			Splošnonamenski register EBP
ESI	SI			Splošnonamenski register EDI
EDI	DI			Splošnonamenski register ESI
ESP	SP			Splošnonamenski register ESP
		CS		Kazalec na ukazni segment
		SS		Kazalec na skladovni segment
		DS		Kazalec na podatkovni segment
		ES		Kazalec na dod. podatkovni segment
		FS		Kazalec na podatkovni segment 2
		GS		Kazalec na podatkovni segment 3
EIP				Programski števec (PC)
EFLAGS				Pogojni biti

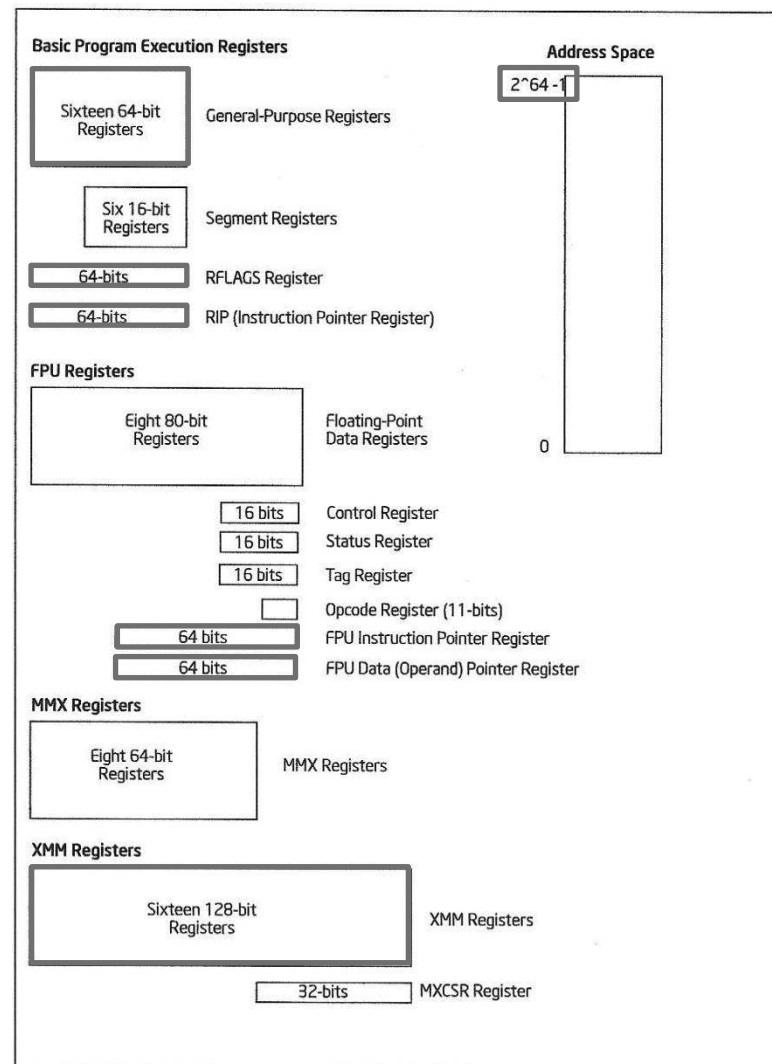
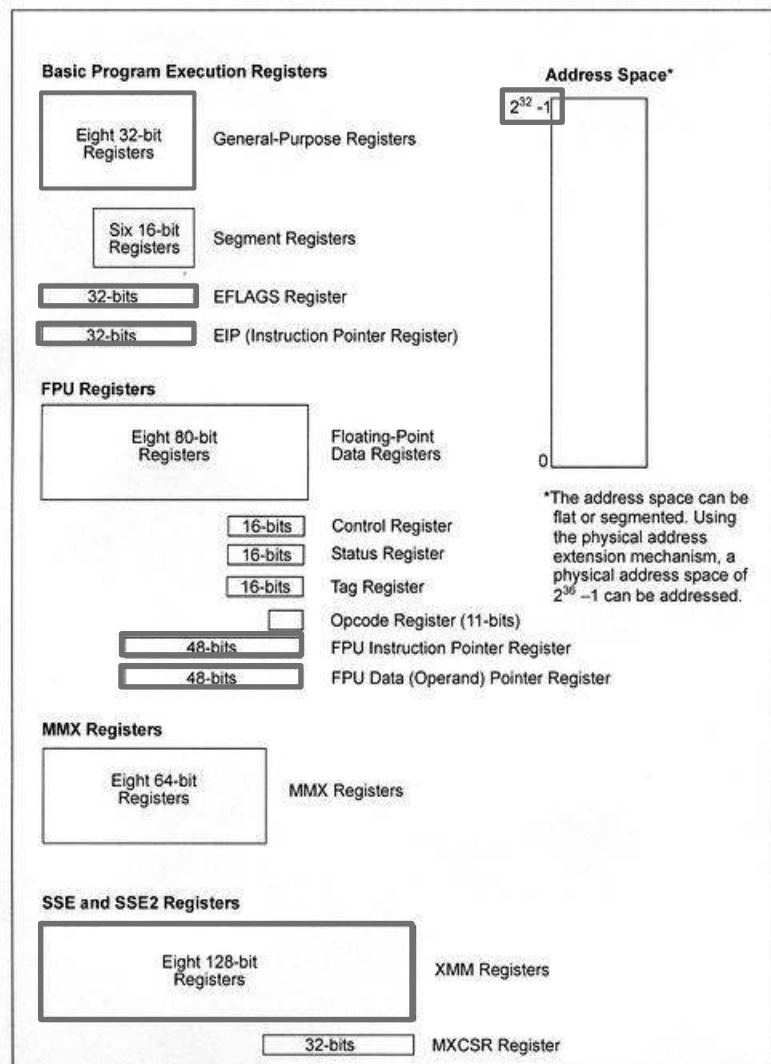


Programsko dostopni registri procesorjev Intel x86 arhitektura *Intel® 64*

32-bitni način IA-32

64-bitni način IA-32e

Razlike





5.3 Število eksplicitnih operandov v ukazu

- Druga najpomembnejša lastnost ukazov po vplivu na uporabnikovo videnje računalnika.
- Manjše število operandov v ukazu:
 - ☐ Krajši ukazi, ki zasedejo manj prostora v pomnilniku
 - ☐ Manjša moč ukazov
- Večje število operandov v ukazu:
 - ☐ Zmogljivejši ukazi
 - ☐ Bolj zapletena zgradba CPE, daljši ukazi



- Na število operandov v ukazu vpliva tudi vrsta operacije, ki jo ukaz izvaja.
- Elementarne operacije z več kot tremi operandi so redke (dva vhodna operanda in rezultat).
- Zato imajo današnji računalniki ukaze, ki vsebujejo največ tri eksplicitne operande.
- Eksplicitni operandi so v ukazu največkrat podani z neposredno ali posredno informacijo o naslovu, kjer je operand shranjen.



- Računalnik, ki uporablja ukaze z največ m -operandi, imenujemo **m-operandni** ali tudi **m-naslovni računalnik**.
- Glede na število eksplicitnih operandov v ukazih lahko računalnike razdelimo na pet skupin:
 - ☐ 3+1-operandni računalniki
 - ☐ 3-operandni računalniki
 - ☐ 2-operandni računalniki
 - ☐ 1-operandni računalniki
 - ☐ Brez-operandni ali skladovni računalniki



■ 3+1-operandni računalniki

- ☐ Predstavnik te vrste računalnikov je bil EDVAC
- ☐ Takih računalnikov danes ni več
- ☐ Oznaka +1 pomeni, da je v ukazu kot operand tudi naslov naslednjega ukaza

Operacijska koda	OP 1	OP 2	OP 3	OP 4
---------------------	------	------	------	------

- ☐ Simbolično lahko delovanje takega računalnika opišemo z:

$OP3 \leftarrow OP2 \oplus OP1$ (\oplus pomeni poljubno operacijo nad dvema operandoma)
 $PC \leftarrow OP4$



■ 3-operandni računalniki

- ☐ S pomnilniki z naključnim dostopom je odpadla potreba po naslovu naslednjega ukaza
- ☐ Implicitni vrstni red izvajanja ukazov določa pravilo $PC \leftarrow PC + 1$

Operacijska koda	OP 1	OP 2	OP 3
---------------------	------	------	------

- ☐ Simbolično lahko delovanje takega računalnika opišemo z:

$$OP3 \leftarrow OP2 \oplus OP1$$

$$PC \leftarrow PC + 1$$



- Osnovne aritmetične operacije imajo tri operande, zato so računalniki s tremi eksplicitnimi operandi najbližji navadam v matematiki.
- Pri računalnikih razvitih po letu 1980 je največ 3-operandnih, vendar običajno z omejitvijo, da so operandi v registrih v CPE (Load/Store računalniki).



■ 2-operandni računalniki

- ☐ Rezultat operacije lahko brez velike škode v večini primerov shranimo v prostor enega od dveh vhodnih operandov.
- ☐ Tako odpade potreba po tretjem operandu in dobimo iz 3-operandnega 2-operandni računalnik



- ☐ Simbolično lahko delovanje takega računalnik opišemo z:

$$\text{OP2} \leftarrow \text{OP2} \oplus \text{OP1}$$
$$\text{PC} \leftarrow \text{PC} + 1$$



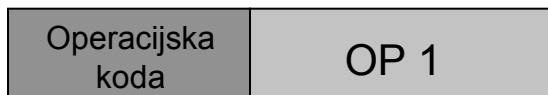
Število eksplicitnih operandov v ukazu – 2-operandni računalniki

- ☐ Operanda sta lahko v pomnilniku ali v enem od registrov v CPE
- ☐ Pri veliko 2-operandnih računalnikih sta lahko oba operanda v pomnilniku (pri 3-operandnih skoraj nikoli)
- ☐ Najpogosteje pa je eden od operandov v registru, drugi pa v pomnilniku
- ☐ Če pri kasnejših operacijah potrebujemo oba vhodna operanda, je treba enega prej shraniti na drugo mesto, v takem primeru je 2-operandni računalnik počasnejši od 3-operandnega
- ☐ Do 1990 so bili 2-operandni računalniki najpogostejši, danes prevladujejo 3-operandni



■ 1-operandni računalnik

- ☐ To so računalniki, ki imajo v CPE en sam akumulator (lahko tudi dva, npr. akumulatorja A in B - 68HC11)
- ☐ Eden od operandov se vedno nahaja v akumulatorju, vanj se shrani tudi rezultat, zato zadošča v ukazu en eksplicitni operand



- ☐ Simbolično lahko delovanje takega računalnika opišemo z:

$AC \leftarrow AC \oplus OP1$ (AC je oznaka za akumulator)
 $PC \leftarrow PC + 1$



Število eksplicitnih operandov v ukazu – 1-operandni računalniki

- Poleg akumulatorja (enega ali dveh) imajo 1-operandni računalniki običajno še vsaj nekaj registrov za posebne namene (npr. indeksni register za shranjevanje pomnilniških naslovov).
- V letih 1970 do 1980 so bili praktično vsi mikro-procesorji zaradi tehnoloških omejitev 1-operandni.



■ Brez-operandni (skladovni) računalniki

- V to skupino spadajo računalniki, ki imajo v CPE pomnilnik v obliki sklada

Operacijska
koda

- Simbolično lahko delovanje brez-operandnega računalnika opišemo z:

$$\text{Sklad}_{\text{VRH}} \leftarrow \text{Sklad}_{\text{VRH}} \oplus \text{Sklad}_{\text{VRH}-1}$$

$$\text{PC} \leftarrow \text{PC} + 1$$



Število eksplicitnih operandov v ukazu – brez-operandni računalniki

- ☐ Operacije se izvajajo nad operandi na vrhu sklada, zato v ukazu niso potrebni eksplicitni operandi
- ☐ Potrebna pa sta vsaj dva ukaza za prenos operanda iz pomnilnika na sklad (PUSH) in iz sklada v pomnilnik (POP ali PULL)
- ☐ Podobno kot pri 1-operandnih računalnikih je tudi pri brez - operandnih poleg sklada običajno v CPE še nekaj registrov za posebne namene
- ☐ Računalnik Atlas leta 1961
- ☐ Med računalniki, ki so bili razviti po letu 1980, ni bilo več skladovnih računalnikov



5.4 Lokacija operandov in načini naslavljanja

- Operandi so lahko shranjeni:
 - ☐ V programsko dostopnih registrih v CPE
 - ☐ V eni ali več sosednjih besedah glavnega pomnilnika (oziroma na kateremkoli nivoju pomnilniške hierarhije)
 - ☐ Operandi so lahko shranjeni tudi v enem od registrov krmilnika V/I naprave ali v V/I procesorju. Zaradi malo V/I ukazov te možnosti ne bomo posebej obravnavali



- **Registrski operandi** so operandi, ki so shranjeni v registrih v CPE
 - Registrski operandi so v ukazu skoraj vedno podani z naslovom registra, v katerem so shranjeni.
 - Za naslov registra je v ukazu posebno polje, lahko pa je naslov vsebovan v operacijski kodi.
 - Samo pri računalnikih z enim akumulatorjem (ali skladovnih) naslov ni potreben.
 - Ker je registrov malo, je za njihov naslov v ukazu potrebnih malo bitov.



- **Pomnilniški operandi** so tisti, ki so shranjeni v glavnem pomnilniku (oziroma na različnih nivojih pomnilniške hierarhije)
 - Naslavljanje je bistveno bolj komplicirano kot pri registrskih operandih
 - Prostor v glavnem pomnilniku je večji, to pa pomeni daljši naslov v ukazu (več bitov)
 - Do operandov v pomnilniku pogosto dostopamo po nekem pravilu, to pa običajno zahteva spreminjanje naslova



- Pri 2- in 3-operandnih računalnikih razlikujemo glede na lokacijo operandov tri vrste računalnikov:
 - ☐ Registrsko-registrski računalniki
 - ☐ Registrsko-pomnilniški računalniki
 - ☐ Pomnilniško-pomnilniški računalniki

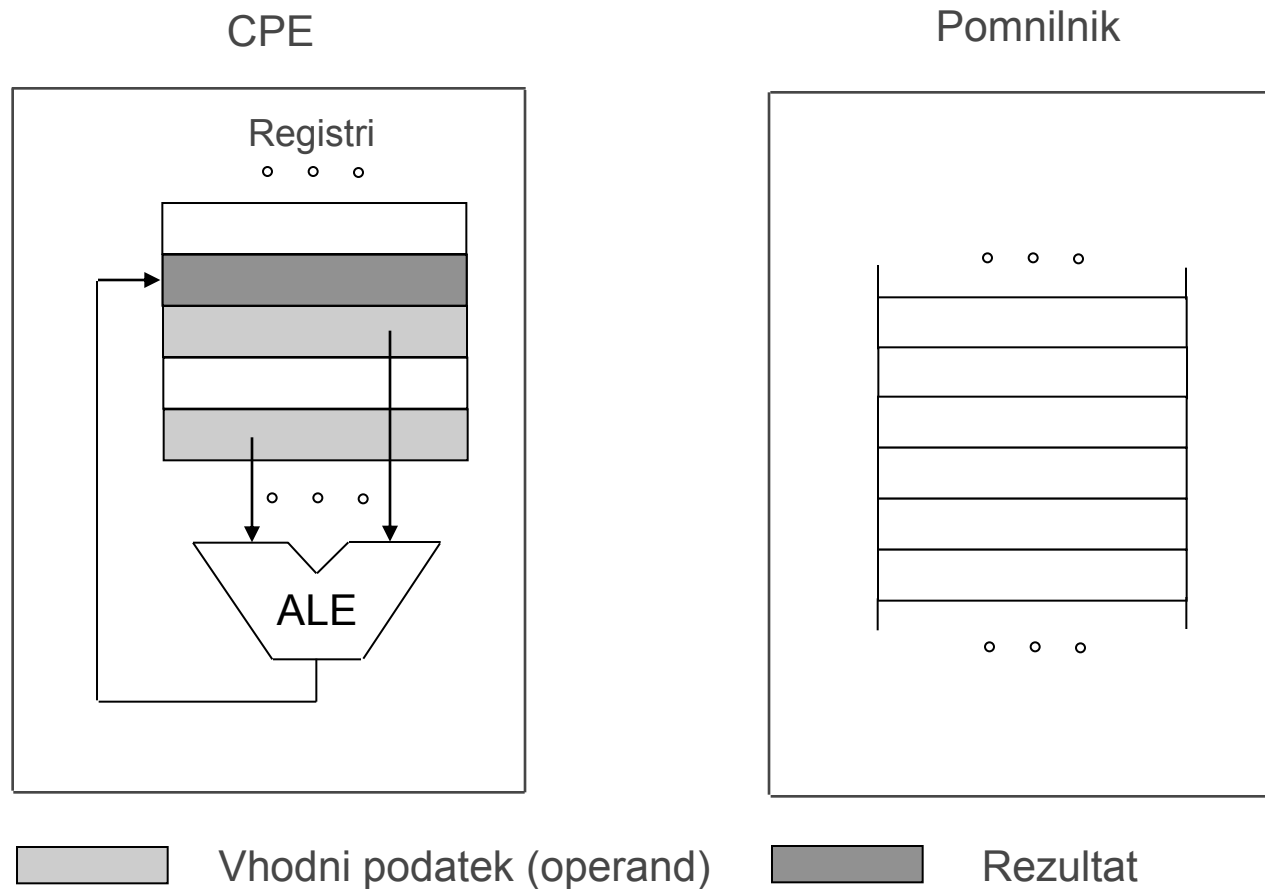


■ Registrsko-registrski računalniki

- ☐ Vsi operandi ALE ukazov so v registrih v CPE.
- ☐ Ukaza LOAD in STORE služita za prenos operandov iz pomnilnika v register in obratno, zato tudi ime **load/store računalniki**.
- ☐ Čas, v katerem se izvršijo ALE ukazi, je vedno enak.
- ☐ Za isti problem je potrebnih več ukazov kot pri računalnikih, ki imajo lahko operande ALE ukazov v pomnilniku.



Lokacija operandov in načini naslavljanja – registrsko-registrski računalniki



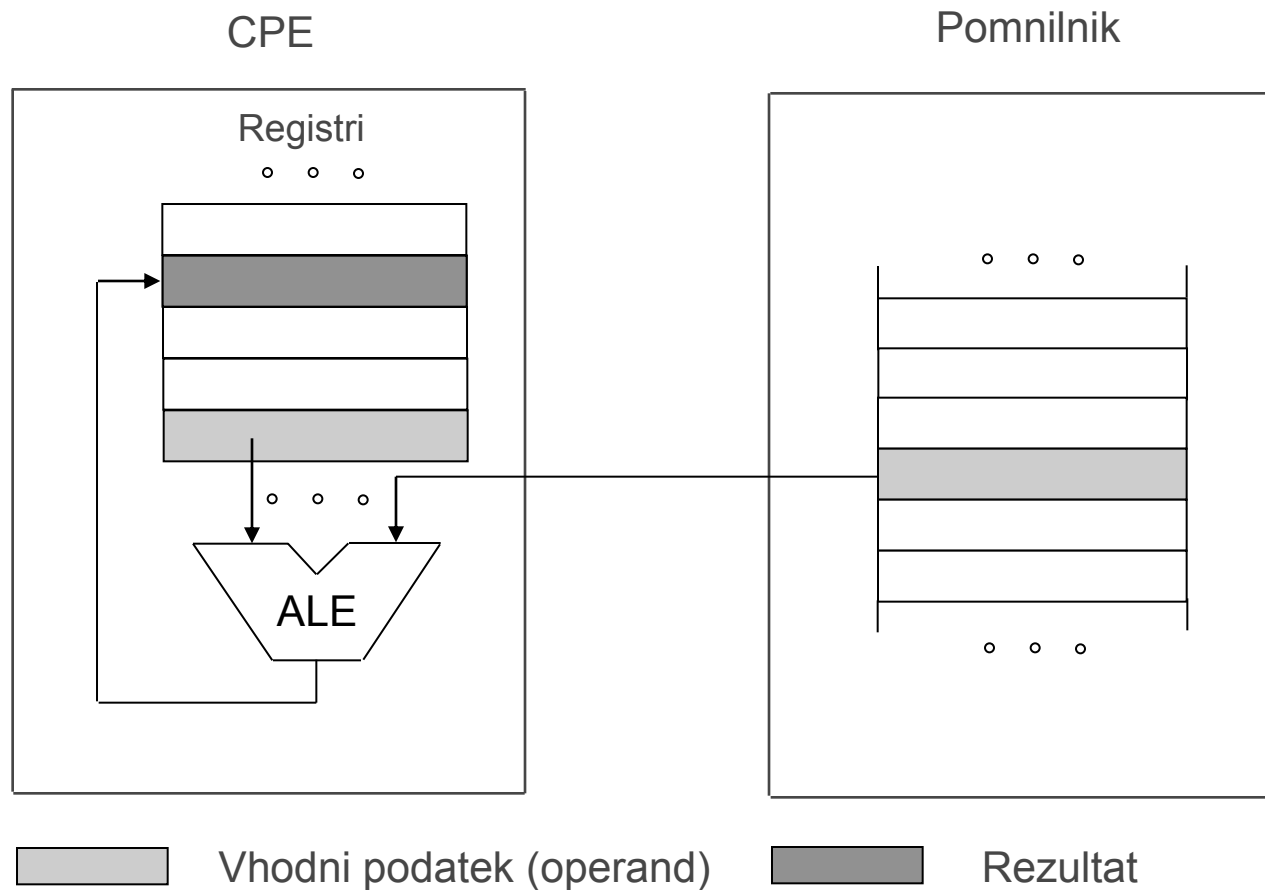


■ Registrsko-pomnilniški računalniki

- ☐ Eden od operandov je lahko v pomnilniku ali registru, drugi pa so vedno v registrih
- ☐ Kot podmnožico vsebujejo tudi ukaze značilne za registrsko-registrske računalnike
- ☐ V ALE ukazih je možno uporabljati pomnilniške operande, ne da bi jih morali pred tem prenesti z ukazom LOAD v register



Lokacija operandov in načini naslavljanja – registrsko-pomnilniški računalniki





- Ukazi so daljši in bolj zapleteni, vendar jih za isti problem potrebujemo manj.
- Čas izvajanja je odvisen od lokacije operandov:
 - Operand v registru – krajši čas izvajanja
 - Operand v pomnilniku – čas izvajanja je daljši in odvisen od tega na katerem nivoju v pomnilniški hierarhiji je shranjen operand

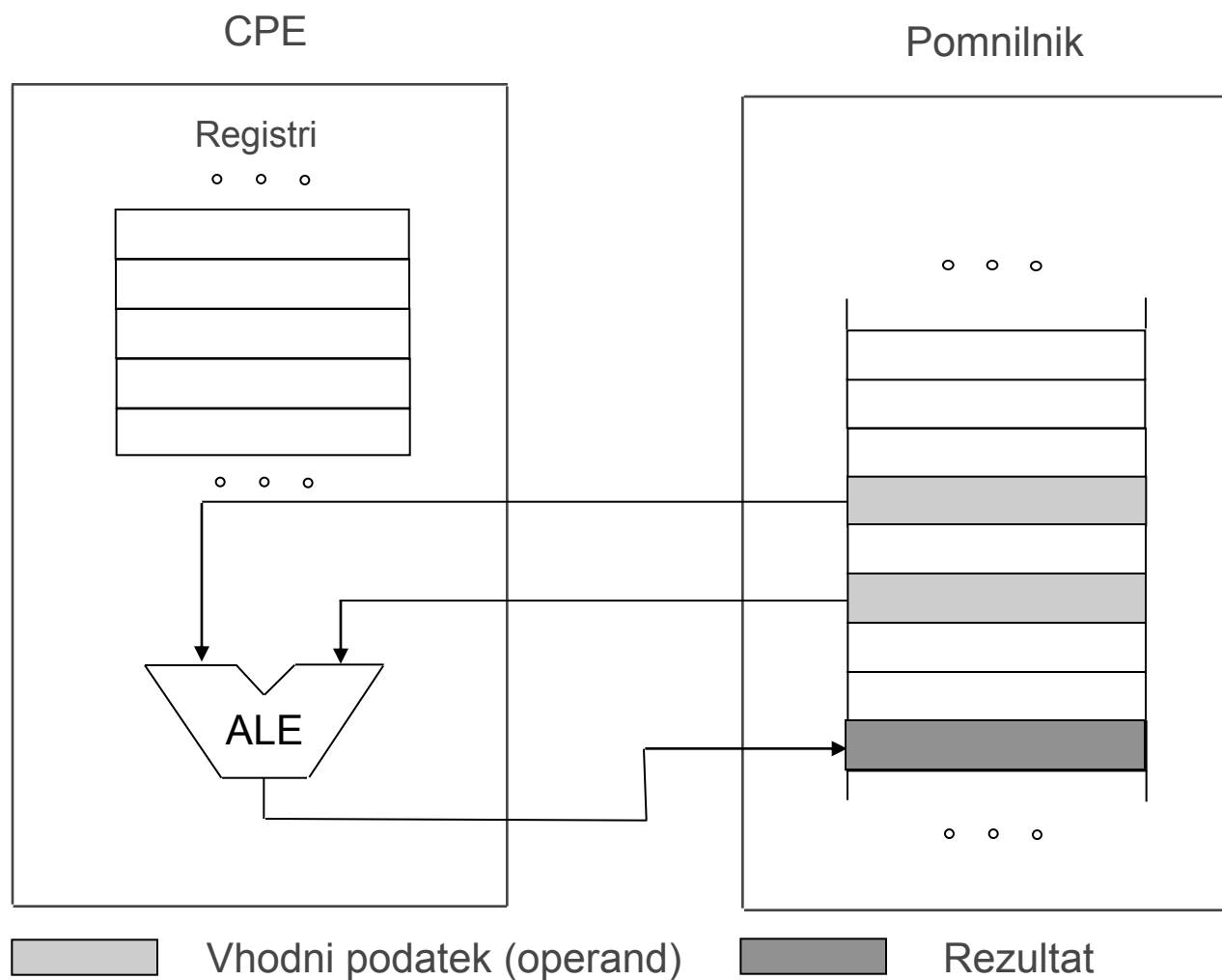


■ Pomnilniško-pomnilniški računalniki

- ☐ Vsak operand je lahko v pomnilniku ali v registru.
- ☐ Kot podmnožico vsebujejo ukaze značilne za registrsko-pomnilniške, kot tudi za registrsko-registrske računalnike.
- ☐ To so najbolj splošni računalniki in omogočajo veliko različnih rešitev pri programiranju istega problema.
- ☐ Ukazi so zapleteni in različno dolgi, velike so tudi razlike v času izvajanja.



Lokacija operandov in načini naslavljanja – pomnilniško-pomnilniški računalniki





Načini naslavljanja - kako so podani naslovi operandov

- Za rešitev teh problemov je bilo razvitih veliko načinov naslavljanja.
- Vse načine naslavljanja lahko razdelimo v tri osnovne skupine:
 - ☐ Takojšnje naslavljanje (angl. immediate addressing)
 - ☐ Neposredno naslavljanje (angl. direct addressing)
 - ☐ Posredno naslavljanje (angl. indirect addressing)



■ Takojšnje naslavljanje

- ☐ Operand je v ukazu podan z vrednostjo
- ☐ Operand je del ukaza in se v CPE prenese skupaj z njim, zato niso potrebni dodatni dostopi do pomnilnika
- ☐ Operand imenujemo **takojšnji operand** ali **literal**
- ☐ Računalniki se med seboj razlikujejo po številu ukazov, ki uporabljajo takojšnje naslavljanje in po dolžini takojšnjih operandov (8, 16 ali 32-bitni)
- ☐ Nekateri računalniki sploh nimajo takojšnjega naslavljanja.

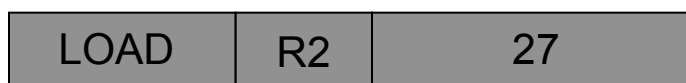


■ Primer ukaza s takojšnjim naslavljanjem:

- Z ukazom LOAD želimo npr. v register R2 prenesti vrednost 27_D , v zbirniku npr. LOAD R2, #27
- # je v zbirniku običajna oznaka za takojšnji operand

Ukaz: LOAD R2, #27 ; $R2 \leftarrow 27_D$

Registri v CPE



Oper. koda

Takojšnji operand



R0

R1

R2



R15



- ARM9: Primeri ukazov s takojšnjim naslavljanjem:
 - Takojšnji operand pri procesorju ARM je 8-biten in 4-biti za pomik
 - Vrednost $(0 \dots 255_D) * 2^{2*(0..12)}$

mov	r0, #0	@ R0 ← 0
add	r3, r3, #1	@ R3 ← R3 + 1
bic	r0, r0, #0x80	@ b7(R0) ← 0



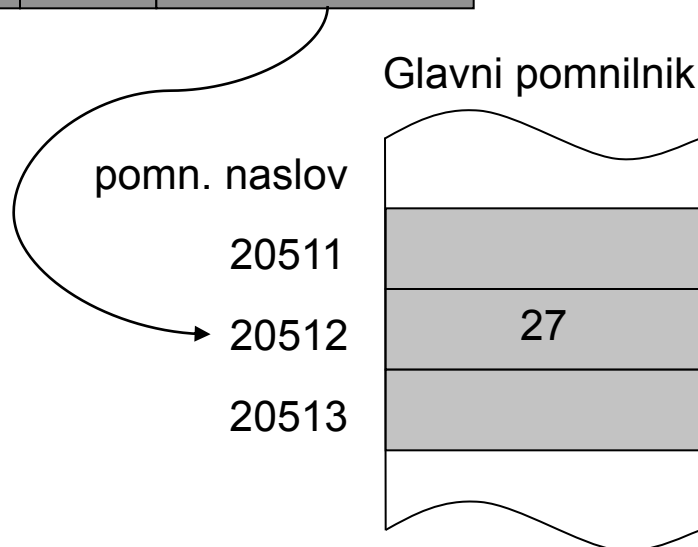
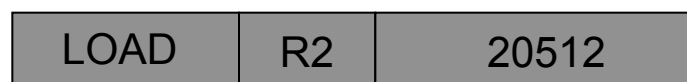
- **Neposredno naslavljanje** (tudi direktno ali absolutno naslavljanje)
 - V ukazu je podan naslov operanda
 - Ta način se uporablja tudi za registrske operande
 - Ukaz vsebuje naslov registra - registrsko naslavljanje
 - Pri pomnilniških operandih je v ukazu naslov lokacije v pomnilniku
 - Ker je naslov del ukaza, se ne spreminja, zato ga imenujemo tudi **absolutni naslov**



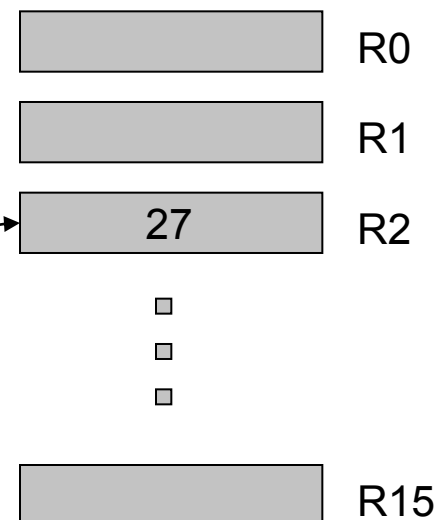
■ Primer ukaza z neposrednim naslavljanjem:

- Z ukazom LOAD želimo npr. v register R2 prenesti vsebino iz pomnilniške lokacije z naslovom 20512_D, v zbirniku npr.:
- LOAD R2, 20512

Ukaz: LOAD R2, 20512 ; R2 ← m[20512_D]



Registri v CPE





- ARM9 neposrednega pomnilniškega naslavljanja nima.
- Neposredno naslavljanje se pri ARM9 uporablja samo za registrske operande.

```
add r5, r0, r1    @ R5 ← R0 + R1
mov r2, r4        @ R2 ← R4
```



- **Posredno naslavljanje** (tudi indirektno ali angl. deferred addressing)
 - V ukazu je naslov pomnilniškega operanda podan posredno preko neke druge vrednosti
 - Ta druga vrednost je:
 - Ali v pomnilniku – **pomnilniško posredno naslavljanje**
 - Ali v registru v CPE – **registrsko posredno naslavljanje**



- Pri pomnilniškem posrednem naslavljanju je v ukazu pomnilniški naslov lokacije, kjer je shranjen pomnilniški naslov operanda.
- Pri registrskem posrednem naslavljanju je v ukazu naslov registra in odmik (displacement).
- Pomnilniški naslov operanda se izračuna iz vsebine registra in odmika.
- Posredno naslavljanje omogoča poljubno spreminjanje naslova operanda in tako odpravlja slabost neposrednega naslavljanja.



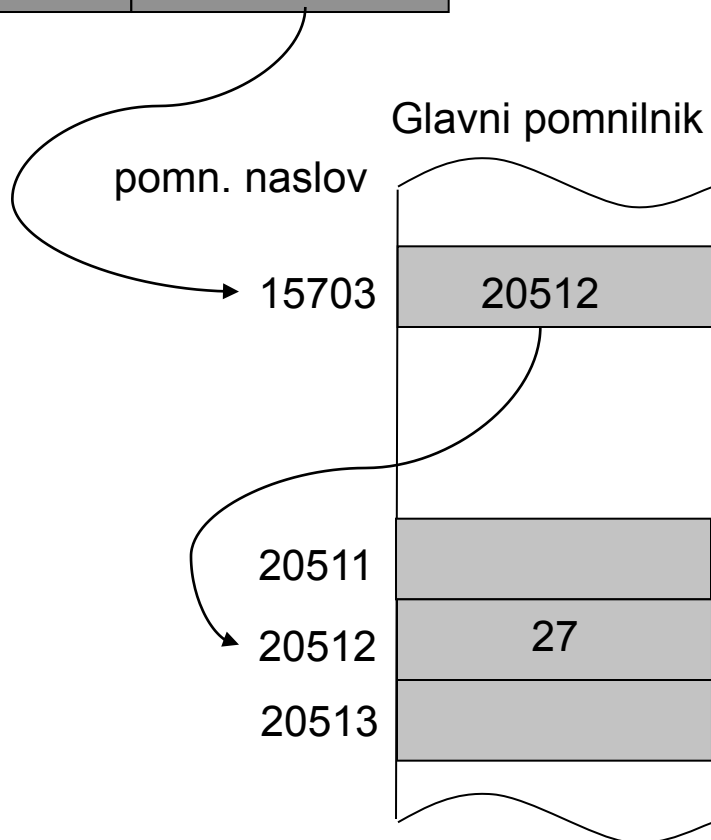
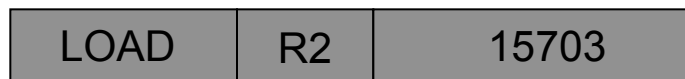
- Primer ukaza s pomnilniškim posrednim naslavljanjem:
 - Z ukazom LOAD želimo npr. v register R2 prenesti vsebino iz pomnilniške lokacije z naslovom 20512_D
 - Pomnilniški naslov 20512_D (naslov operanda) je shranjen v pomnilniku na naslovu npr. 15703_D (posredni naslov)
 - V zbirniku npr. `LOAD R2, @(15703)`
 - `@(....)` je v zbirnikih pogosta oznaka za posredni naslov



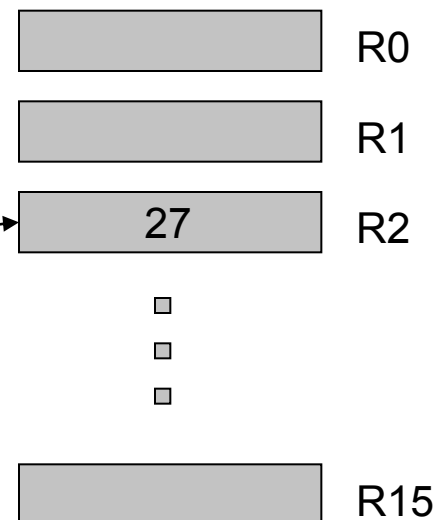
Lokacija operandov in načini naslavljanja – pomnilniško posredno naslavljanje

Ukaz: LOAD R2, @(15703) ; $R2 \leftarrow m[m(15703_D)]$

Posredni naslov



Registri v CPE





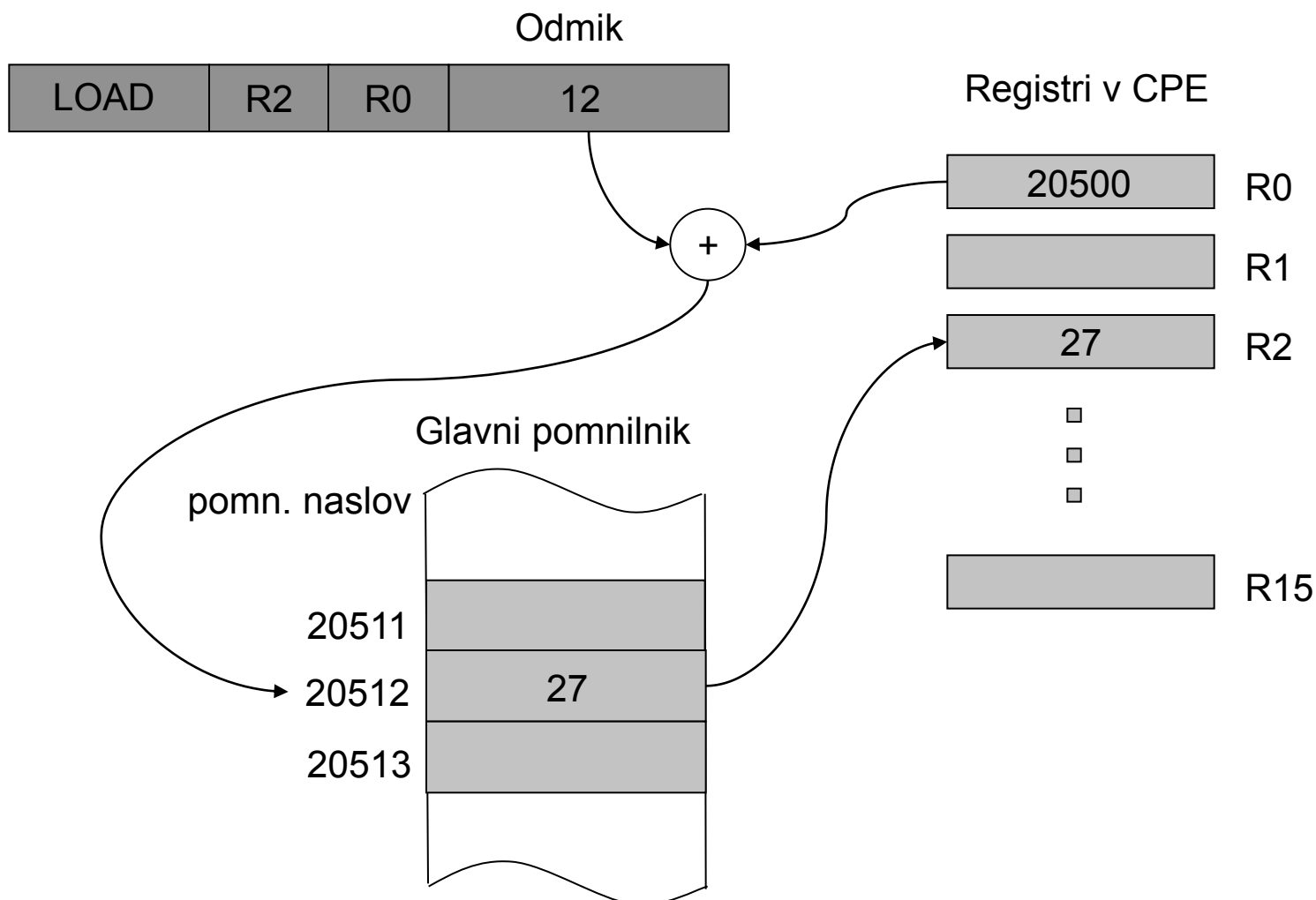
■ Primer ukaza z registrskim posrednim naslavljanjem:

- Z ukazom LOAD želimo npr. v register R2 prenesti vsebino iz pomnilniške lokacije z naslovom 20512_D
- V registru R0 imamo shranjen pomnilniški naslov npr. 20500_D (posredni naslov)
- V zbirniku npr. `LOAD R2, 12(R0)`
- V ukazu je 12_D odmik, ki se prišteje k naslovu v registru R0. Rezultat je pomnilniški naslov operanda ($20500_D + 12_D = 20512_D$)



Lokacija operandov in načini naslavljanja – registrsko posredno naslavljanje

Ukaz: **LOAD R2, 12(R0)** ; $R2 \leftarrow m[12+(R0)]$





- Največ različnih inačic je pri registrskem posrednem naslavljanju.
- Pri večini računalnikov je registrsko posredno naslavljanje najpogostejši način za dostop do pomnilniških operandov.
- Naslov operanda je vedno določen relativno glede na vsebino najmanj enega registra, drugo ime za to vrsto naslavljanja je zato **relativno naslavljanje**.



■ Vrste registrskega posrednega naslavljanja:

- ☐ Bazno naslavljanje (base addressing)
- ☐ Indeksno naslavljanje (indexed addressing)
- ☐ Avtomatsko indeksno naslavljanje (autoindexing)
- ☐ PC-relativno naslavljanje



■ Bazno naslavljanje (base addressing)

- Najpogostejša vrsta registrskega posrednega naslavljanja.
- V ukazu sta podana naslov registra R_b in odmik D .
- Pomnilniški naslov (npr. A) operanda dobimo tako, da seštejemo vsebino registra R_b in odmik D :
$$A = R_b + D$$
- Dolžina registra R_b je običajno enaka ali daljša od dolžine pomnilniškega naslova
- R_b imenujemo **bazni register** (base register), A pa **dejanski naslov** (effective address)



- ☐ Glede na dolžino odmika D obstaja veliko variant baznega naslavljanja
- ☐ Skrajna primera sta, da odmika sploh ni ($D = 0$) ali da je dolžina odmika (v bitih) enaka dolžini pomnilniškega naslova
- ☐ V drugem primeru bazno naslavljanje preide v **indeksno naslavljanje**



■ ARM9:

- Posredno naslavljanje s takojšnjim odmikom (= bazno naslavljanje z odmikom):

```
ldr r5, [r3, #12]    @ R5 ← m(R3 + 12)
```

[] v zbirniku za ARM9 oznaka za posredni naslov

- Posredno naslavljanje z odmikom v registru (= indeksno naslavljanje):

```
ldr r5, [r3, r1]     @ R5 ← m(R3 + R1)
```

32-bitni odmik je v registru R1



■ Indeksno naslavljanje (indexed addressing)

- Dolžina odmika D v bitih je enaka dolžini pomnilniškega naslova, kar pomeni, da lahko samo z odkikom naslovimo cel naslovni prostor
- Druga možnost je, da bazno naslavljanje spremenimo v indeksno tako, da dodamo še en register
- Dejanski naslov A se izračuna z vsoto vsebine indeksnega registra in odmika D1, odkim D1 pa je vsota vsebine baznega registra in odmika D:

$$A = Rx + D1 = Rx + \underbrace{Rb + D}$$

Odmik z dolžino
pomnilniškega naslova

Rx – indeksni register
Rb – bazni register



■ **Avtomatsko indeksiranje**

- ☐ Pred-dekrementno naslavljanje
- ☐ Po-inkrementno naslavljanje
- ☐ Velikostno indeksno naslavljanje

■ **Pri skočnih ukazih veliko računalnikov uporablja **PC - relativno naslavljanje****

- ☐ Kot bazni register se uporabi programski števec – PC
- ☐ Odmik je število s predznakom (v dvojiškem komplementu)
- ☐ Naslov se tako računa relativno glede na vrednost PC



5.5 Operacije (vrste ukazov)

- Po številu in vrsti operacij se računalniki med seboj zelo razlikujejo.
- Računalnik mora imeti toliko operacij, da je zagotovljena ekvivalenca s Turingovim strojem (z njimi mora biti mogoče izračunati vse, kar se da izračunati).
- Zato pa zadoščajo zelo primitivne operacije, ali v skrajnem primeru že ena sama dovolj zmogljiva operacija.



- Izhodišči za določitev vrste in števila operacij sta pri današnjih računalnikih dve:
 - **Množica operacij naj bo močna.** Za pogosto uporabljene funkcije naj zadošča ena operacija ali kratko zaporedje operacij
 - **Operacije naj bodo podobne že uveljavljenim vrstam operacij.** Večina proizvajalcev uporablja iste ali podobne operacije, kar poenostavi programiranje, izdelavo prevajalnikov in digitalne elektronike v CPE
- Vrsto operacije lahko razberemo iz imena ukaza
- Ime ukaza je **mnemonik**, s katerim je ukaz definiran v zbirnem jeziku



- Pri vseh računalnikih srečamo iste osnovne skupine operacij:
 - ☐ Aritmetične in logične operacije
 - ☐ Prenosi podatkov
 - ☐ Kontrolne operacije
 - ☐ Operacije v plavajoči vejici
 - ☐ Systemske operacije
 - ☐ Vhodno/izhodne operacije



■ Aritmetične in logične operacije (ALE)

- Te operacije se izvajajo v ALE, ukazi ki izvajajo te operacije se označujejo kot **ALE ukazi**

Aritmetične operacije: v tej skupini so operacije nad operandi v fiksni vejici (celimi števili)

- Tipične aritmetične operacije so:
 - Dvo-operandne - seštevanje, odštevanje, množenje in deljenje
 - Eno-operandne - negacija, absolutna vrednost, inkrement in dekrement



- Logične operacije so poleg Boolovih operacij še pomiki

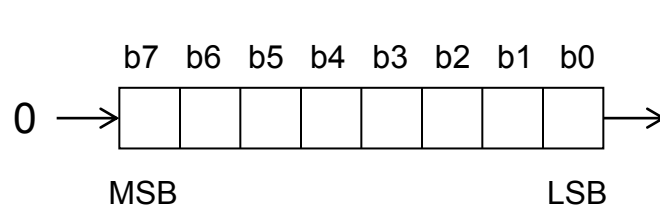
Boolove operacije:

- Čeprav je vse logične operacije mogoče realizirati samo z operacijo NAND ali NOR, ima večina računalnikov zaradi lažje uporabe ukaze za štiri Boolove operacije
 - Dvo-operandne - konjunkcija (AND), disjunkcija (OR) in ekskluzivna disjunkcija (XOR)
 - Enoperandna - logična negacija (NOT)

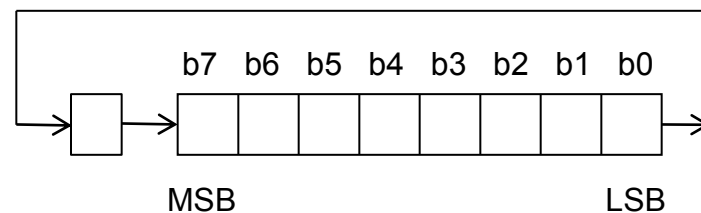


Pomiki:

- ☐ Navadni pomiki (angl. shift)
- ☐ Krožni pomiki - rotacije (angl. rotate)
- ☐ Pri obeh vrstah imamo lahko leve ali desne pomike



Desni navadni pomik



Desni krožni pomik



- Pri desnih navadnih pomikih razlikujemo:
 - Logični pomik - v izpraznjeno mesto se vstavljajo ničle
 - Aritmetični pomik - najbolj levi bit (predznak) ostaja nespremenjen in se širi na izpraznjena mesta
- Pri dvojiški aritmetiki je desni aritmetični pomik enak predznačenemu deljenju z 2
- Pri levih navadnih pomikih je aritmetični pomik enak logičnemu (množenje z 2)



- Logične operacije se ne uporabljajo samo pri računanju logičnih funkcij, temveč še pogosteje za npr.:
 - Izločanje posameznih bitov iz besede (operacija AND ali pomiki).
 - Vstavljanje posameznih bitov v besedo (operacija OR ali pomiki).
 - Pri programiranju V/I naprav.
 - Na splošno v primerih, ko so v eni besedi združeni biti z različnim pomenom.



■ Prenosi podatkov

- Operacija za prenos informacije iz enega dela računalnika v drugega je najelementarnejša operacija.
- Pri vsakem prenosu informacije imamo **izvor** informacije in **ponor** informacije.
- Po izvršitvi operacije za prenos imamo informacijo na obeh mestih, na ponoru in izvoru, zato bi bila bolj pravilna oznaka podvajanje ali kopiranje operandov.
- Ukazi za prenos so lahko za različne dolžine operandov (npr. 8, 16, 32 ali 64 bitov).
 - Pri ARM9: word (32-bitov), byte (8-bitov) ali half-word (16-bitov). Pri obeh prenosih, ki sta krajša kot 32-bitov, imamo lahko razširjanje ničle ali razširjanje predznaka.



- Pri večini računalnikov imamo več vrst ukazov za to operacijo.
- Razlog za to je, da so operandi lahko v registrih v CPE ali v glavnem pomnilniku.
- Običajno se pri operacijah za prenos podatkov uporabljajo naslednji mnemoniki:
 - LOAD pri prenosu iz pomnilnika v register (ARM9: **ldr**)
 - STORE pri prenosu iz registra v pomnilnik (ARM9: **str**)
 - MOVE pri prenosih iz registra v register in iz pomnilnika v pomnilnik (ARM9: **mov, mrs, ...**)
 - PUSH pri prenosu iz pomnilnika ali iz registra v sklad (ARM9: **stm**)
 - POP (PULL) pri prenosu iz sklada v register ali pomnilnik ARM9: **ldm**)



■ Kontrolne operacije

- ☐ To so operacije, s katerimi spreminjamo običajni vrstni red izvajanja ukazov.
- ☐ Vsi ukazi te vrste vsebujejo informacijo o naslovu (ukaza), na katerem naj se nadaljuje izvajanje programa.
- ☐ Temu naslovu pravimo **ciljni naslov** (angl. target address).



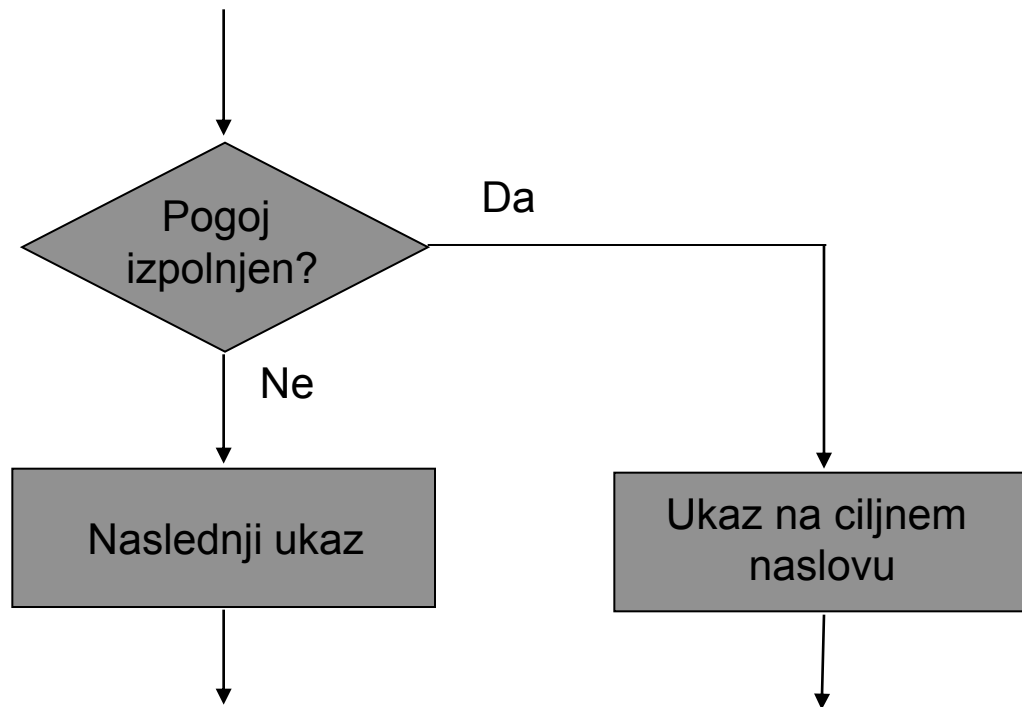
- ☐ Informacija o ciljnem naslovu je v ukazu običajno podana s PC-relativnim naslavljanjem, kjer je v ukazu podan odmik, ki se prišteje k trenutni vsebini PC.

- Kontrolne operacije razdelimo na tri vrste:

- ☐ **Pogojni skoki.** Skok na ciljni naslov se izvrši samo, če je pogoj v ukazu izpolnjen, če pogoj ni izpolnjen, se izvrši naslednji ukaz (običajni vrstni red izvajanja).



- Predstavitev ukaza za pogojni skok v diagramu poteka:





- **Brezpogojni skoki.** Skok na ciljni naslov se izvrši vedno.
 - Običajna terminologija proizvajalcev - brezpogojni skoki imajo ime (mnemonik) **jump**, pogojni pa **branch**.

- **Klici in vrnitve iz procedur oz. podprogramov.** Procedura ali podprogram je zaporedje ukazov, ki opravlja neko točno določeno delo in ga lahko pokličemo iz različnih mest v programu.
 - Ko procedura opravi svoje delo, se mora program nadaljevati z ukazom, ki sledi klicu procedure.



- Ukaz za klic mora zato shraniti naslov, na katerega se mora procedura vrniti - imenujemo ga **povratni naslov**.
- Tipična mnemonika za klic procedure sta CALL in JSR (angl. Jump to Subroutine).
- Povratni naslov se lahko shrani v nek register v CPE ali pa v sklad v glavnem pomnilniku.
- Tako shranjen povratni naslov uporabi ukaz za vrnitev iz procedure, ki je zadnji ukaz procedure.
- Mnemonik ukaza za vrnitev je običajno RET (Return) ali RTS (Return from Subroutine).



■ Operacije v plavajoči vejici

- ☐ Te operacije običajno obravnavamo posebej, čeprav spadajo med aritmetično-logične operacije.
- ☐ Vzrok je, da se običajno izvajajo v posebni enoti v CPE, ki jo imenujemo **enota za operacije v plavajoči vejici** (angl. FPU – Floating Point Unit).
- ☐ Ta enota ni del ALE in običajno lahko deluje paralelno z ALE.
- ☐ Intelovi procesorji s Core arhitekturo imajo 3 enote ALE in 2 enoti FPU.



- Operacije v plavajoči vejici vključujejo:

- Osnovne štiri aritmetične operacije

- in pogosto še operacije za računanje:

- Kvadratnega korena

- Logaritma

- Eksponencialne funkcije

- Trigonometričnih funkcij



■ **Sistemske operacije**

- Operacije s katerimi spreminjamo parametre delovanja računalnika in nadzorujemo njegovo delovanje.
- S temi operacijami lahko vplivamo na:
 - Prekinitve in pasti
 - Delovanje predpomnilnika
 - Delovanje navideznega pomnilnika
 - Nivo privilegiranosti
 - Ustavitev delovanja



Operacije - sistemske operacije

- ☐ Pri večini računalnikov spadajo ukazi za sistemske operacije med privilegirane ukaze.
- ☐ Večino ukazov za sistemske operacije uporabljajo programi operacijskega sistema in so za običajne programerje prepovedani.



■ Vhodno/izhodne operacije

- ☐ Ukazi za vhodno/izhodne operacije prenašajo ali sprožijo prenos informacije med:
 - Glavnim pomnilnikom in vhodno/izhodno napravo
 - CPE in vhodno/izhodno napravo
- ☐ Računalniki, ki uporabljajo pomnilniško preslikan vhod/izhod, nimajo posebnih ukazov za V/I operacije. Uporabljajo se kar ukazi za prenos podatkov.
- ☐ Tudi V/I ukazi so običajno privilegirani in jih uporabljajo samo programi operacijskega sistema.



- Pri aritmetičnih in logičnih operacijah in operacijah v plavajoči vejici lahko ukaze razdelimo še na **skalarne** in **vektorske**.
- Skalarni ukazi so običajni ukazi, ki jih ima večina računalnikov, operacija se izvrši nad operandi, ki jih podaja ukaz.
- Pri vektorskih ukazih pa se operacija izvrši na zaporedju N operandov.
- Pri vektorskih ukazih ni nujno, da se N operacij izvede paralelno, je pa za izvedbo potreben samo en ukaz.



5.6 Vrsta in dolžina operandov

- Pri operandih želimo, da imamo v ukazih vse tipe operandov, ki nastopajo v višjih programskih jezikih
- Najpogostejše vrste operandov v ukazih so:
 - ☐ Bit
 - ☐ Znak (angl. character)
 - ☐ Celo število (angl. integer)
 - ☐ Realno število (angl. real)
 - ☐ Desetiško število (angl. decimal)



- **Bit.** Enobitnih operandov v večini višjih programskih jezikov ni, so pa uporabni pri sistemskih in vhodno/izhodnih funkcijah.
- **Znak.** Dolžina je 8 bitov (tudi 16 bitov), predstavljeni so v ASCII, EBCDIC ali Unicode abecedi. Uporaba tudi v obliki nizov različnih dolžin.
- **Celo število.** Dolžina je 8, 16, 32 ali 64 bitov. Ti operandi so običajno predstavljeni kot predznačena števila v fiksni vejici (v dvojiškem komplementu).



- **Realno število.** Oznaka za število v plavajoči vejici (npr. po standardu IEEE 754). Dolžina operandov je 32, 64 ali 128 bitov

- **Desetiško število.** Niz 8-bitnih znakov v:
 - Nepakirani obliki (ena številka v ASCII ali EBCDIC v 8-bitnem znaku)

 - Pakirani obliki (dve BCD številki v 8-bitnem znaku)



■ Sestavljeni pomnilniški operandi

- ☐ Dolžina operandov v bitih je mnogokratnik števila 8, ker je običajna dolžina pomnilniške besede danes 8 bitov.
- ☐ Operandi, ki so daljši kot 8 bitov, zasedejo več pomnilniških besed, imenujemo jih **sestavljene pomnilniški operandi**.
- ☐ Na računalniku z 8-bitnimi pomnilniškimi besedami imamo lahko npr. 32-bitne operande, ki zasedejo 4 sosednje pomnilniške besede.



Vrsta in dolžina operandov - sestavljeni pomnilniški operandi

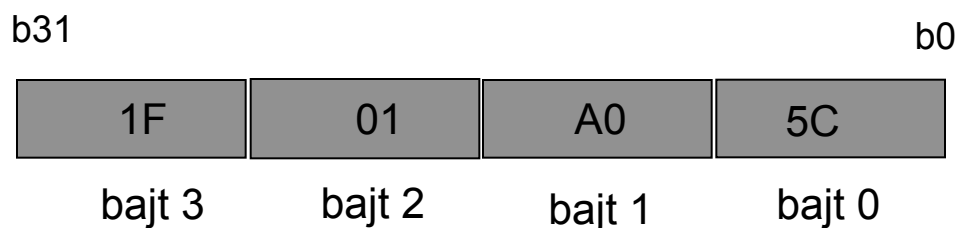
- ☐ Besede morajo biti sosednje, da lahko lokacijo operanda podamo z enim naslovom (z naslovom prve besede)
- ☐ Potreben je dogovor na katero od štirih besed kaže naslov in kakšen je vrstni red shranjevanja
- ☐ Za shranjevanje sestavljenih pomnilniških operandov v pomnilnik se danes uporabljata dva načina:
 - Pravilo debelega konca (angl. Big Endian Rule)
 - Pravilo tankega konca (angl. Little Endian Rule)



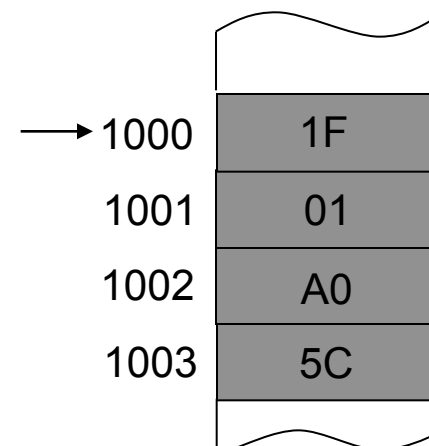
■ Pravilo debelega konca

- Naslov sestavljenega operanda je enak naslovu besede, ki vsebuje najtežji del (bajt) operanda
- Primer: 32-bitni sestavljeni pomnilniški operand *1F01A05C(hex)*, shranjen po pravilu debelega konca na naslov *1000(dec)*:

Sestavljen 32-bitni pomnilniški operand



Pomnilnik z 8-bitnimi besedami

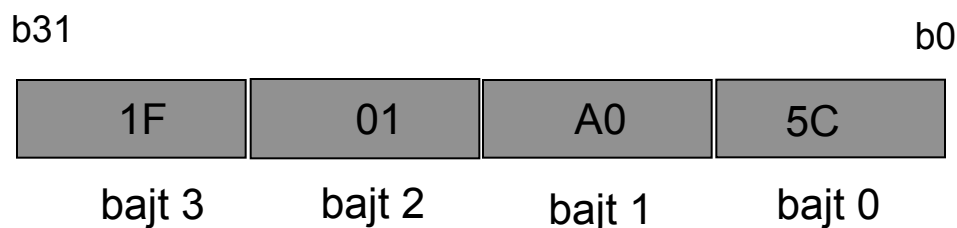




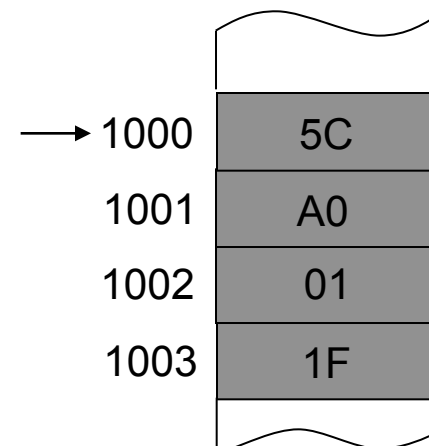
■ Pravilo tankega konca

- Naslov sestavljenega operanda je enak naslovu besede, ki vsebuje najlažji del (bajt) operanda
- Primer: 32-bitni sestavljeni pomnilniški operand *1F01A05C(hex)*, shranjen po pravilu tankega konca na naslov *1000(dec)*:

Sestavljen 32-bitni pomnilniški operand



Pomnilnik z 8-bitnimi besedami





- **Problem poravnosti** sestavljenih pomnilniških operandov
- Sestavljeni pomnilniški operand je **poravnan operand**, kadar je shranjen na naslovu A , za katerega velja:

$A \bmod s = 0$; kjer je s število besed sestavljenega operanda

- Naslov, A za katerega velja zgornja enačba, je **naravni naslov**.
- Če enačba ne velja, je operand neporavnan.

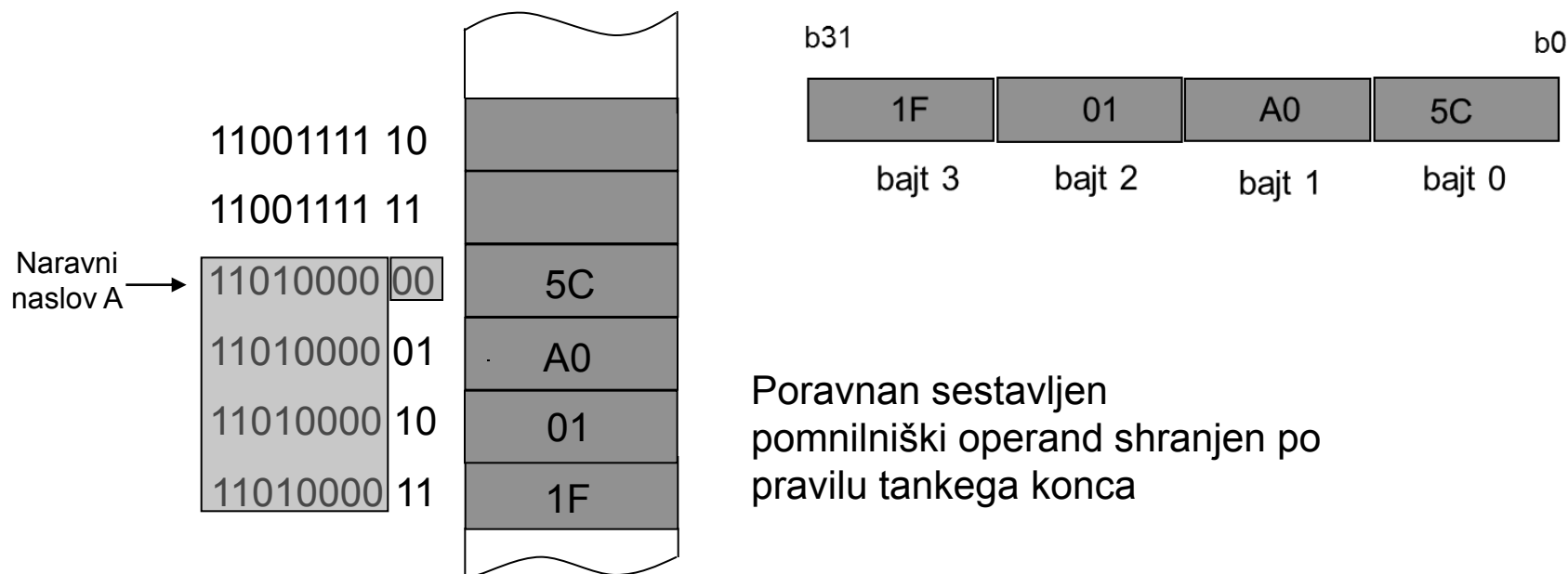


- To pomeni, da mora biti v našem primeru naslov 32-bitnega sestavljenega pomnilniškega operanda deljiv s 4 brez ostanka, da je operand poravnan (npr. $1000_D \bmod 4 = 0$)
 - Pomnilnik, ki omogoča dostop do 4 pomnilniških besed naenkrat, je lahko narejen kot 4 paralelno delujoči pomnilniki.
 - Če je 32-bitni operand poravnan, spodnja dva bita n -bitnega pomnilniškega naslova določata v katerem od 4 pomnilnikov je posamezen bajt sestavljenega operanda, preostalih $n-2$ bitov pomnilniškega naslova pa je za vse 4 bajte enakih.



Vrsta in dolžina operandov - sestavljeni pomnilniški operandi

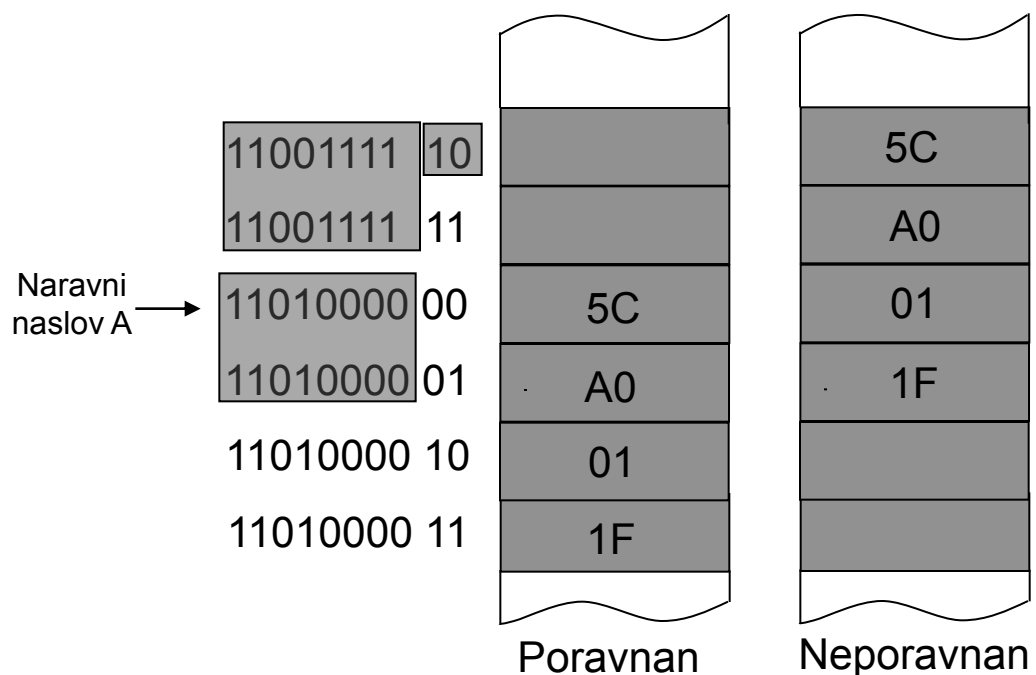
- Tako je možen dostop do vseh štirih pomnilnikov (bajtov) hkrati.
- Če pa 32-bitni operand ni poravnan, preostalih $n-2$ bitov pomnilniškega naslova ni enakih za vse 4 bajte in hkratni dostop do vseh štirih bajtov ni možen.





Vrsta in dolžina operandov - sestavljeni pomnilniški operandi

- Tako je možen dostop do vseh štirih pomnilnikov (bajtov) hkrati.
- Če pa 32-bitni operand ni poravnan, preostalih $n-2$ bitov pomnilniškega naslova ni enakih za vse 4 bajte in hkratni dostop do vseh štirih bajtov ni možen.



Pri naravnem naslovu sta najlažja bita enaka 0, torej je deljiv s 4 brez ostanka



Pomnilniški naslovi 8-bitnih pomnilniških besed, na katerih je sestavljen pomnilniški operand poravnan

Dolžina sestavljenega operanda v bitih	Naslov A, na katerem je sestavljeni operand poravnan
8 (ni sestavljen)	XXX...XXXXXXXXX
16	XXX...XXXXXXXX0
32	XXX...XXXXXXXX00
64	XXX...XXXXX000
128	XXX...XXXXX0000

X = 0 ali 1



5.7 Zgradba ukazov

- Lastnosti ukazov za katere se razvijalci odločijo, je potrebno vgraditi v ukaze
- Izbrane lastnosti določajo format ukaza, še vedno pa je mogoče zgraditi ukaz na več načinov
- Najpomembnejši faktorji, ki določajo zgradbo ukaza so:
 - **Dolžina pomnilniške besede.** Dolžina ukaza naj bo mnogokratnik dolžine pomnilniške besede



- **Število eksplicitnih operandov v ukazu.** Za vsak eksplicitni operand mora biti v ukazu določeno, kje je shranjen

- **Vrsta in število registrov v CPE.** Registri so lahko enakovredni in vsi enake dolžine, lahko pa imamo registre različnih dolžin za različne namene. Od števila registrov je odvisno, koliko bitov je v ukazu potrebnih za naslov registra, v katerem je eksplicitni operand

- **Dolžina pomnilniškega naslova.** Pri neposrednem naslavljanju je v ukazu pomnilniški naslov (dolžina), to je razlog, da se neposredno pomnilniško naslavljanje redko uporablja. Izjema so procesorji s sorazmerno kratkimi pomnilniškimi naslovi (npr. 68HC11 - 16-bitni naslov)



- Formati ukazov s krajšo dolžino se skušajo dati ukazom, ki se bolj pogosto uporabljajo
- V ukazu je informacija o operaciji in operandih lahko razporejena na različne načine
- Primer nesistematične zgradbe ukazov so Intelovi mikroprocesorji, ki so jim dodajali vedno nove ukaze, ki jih je bilo treba vključiti med že obstoječe
- Rezultat je veliko število nenavadnih in težko razumljivih formatov ukazov



- Tipični načini zgradbe ukazov pri današnjih računalnikih:
 - Spremenljiva dolžina ukazov
 - Število eksplicitnih operandov v ukazu se spreminja, veliko različnih formatov ukazov
 - Fiksna dolžina ukazov
 - Število eksplicitnih operandov v ukazu je vedno enako, malo formatov ukazov (PowerPC, SPARC)
 - Hibridni način
 - Različne dolžine ukazov (IBM 370, Intel 80x86)



- **Ortogonalnost ukazov** - zgradba ukazov za katero velja:
 - ☐ Informacija o operaciji je neodvisna od informacije o operandih
 - ☐ Informacija o enem operandu v ukazu je neodvisna od informacije o drugih operandih
 - ☐ Pri ortogonalnih ukazih lahko uporabimo za vsak operand vse načine naslavljanja (smiselne) in vse dolžine operandov



5.8 RISC - CISC računalniki

- Razprave o številu ukazov, ki naj jih imajo računalniki, so se pojavile po letu 1980
- **CISC računalniki** (angl. Complex Instruction Set Computer) - računalniki z velikim številom ukazov
- **RISC računalniki** (angl. Reduced Instruction Set Computer) - računalniki z majhnim številom ukazov



- Razvoj računalnikov kaže, da se je število ukazov nenehno povečevalo
- Meritve pogostosti izvajanja ukazov na CISC računalnikih pa kažejo, da se velik del ukazov uporablja zelo redko
- Najbolj pogosto se uporabljajo ravno najenostavnejši ukazi z enostavnimi načini naslavljanja



- Razlogi za povečevanje števila ukazov:
 - **Semantični prepad** - razlika med računalnikom kot ga vidi programer v višjem programskem jeziku in tistim, kar vidi programer v strojnem jeziku
 - **Mikroprogramiranje** - preprosto dodajanje novih ukazov
 - **Razmerje med hitrostjo glavnega pomnilnika in CPE** - v letih 1960 do 1980 je bila hitrost dostopa do informacije v CPE (do mikroukazov) več kot 10-krat višja kot hitrost dostopa do glavnega pomnilnika.



■ Razlogi za zmanjšanje števila ukazov:

- **Težave pri uporabi kompleksnih ukazov v prevajalnikih** - bolje je, da arhitektura ponuja preproste elemente za reševanje, kot pa rešitve, ki jih pogosto ni mogoče uporabiti
- **Spremenjeno razmerje med hitrostjo glavnega pomnilnika in CPE** - mikroprogramska realizacija je postala počasna v primerjavi s fiksno ožičeno, kompleksni ukazi pa težki za realizacijo v fiksno ožičeni logiki
- **Uvajanje paralelizma v CPE** - realizacija cevovodov je pri preprostih ukazih lažja kot pri kompleksnih



■ Ideja o RISC računalnikih

- Prvi RISC računalnik je bil IBM 801 iz leta 1975
- Pri določitvi ukazov sta bila uporabljena dva kriterija:
 1. Ukaz mora biti dovolj preprost, da je operacijo mogoče izvršiti v eni urini periodi
 2. Ukaz izvaja tako operacijo, da je ni mogoče realizirati hitreje z zaporedjem ukazov, ki jih tvori prevajalnik, ki razume visokonivojski pomen programa



- **Definicija pojma RISC arhitektura.** Računalnik ima RISC arhitekturo, če izpolnjuje naslednjih šest pogojev:
 1. Večina ukazov se izvrši v eni urini periodi (v enem CPE ciklu)
 2. Registrsko-registrska zasnova (load/store računalnik)
 3. Ukazi so realizirani s trdo ožičeno logiko in ne mikroprogramsko
 4. Malo ukazov in malo načinov naslavljanja
 5. Vsi ukazi imajo isto dolžino
 6. Dobri prevajalniki (upoštevajo zgradbo CPE)



- Pojem RISC arhitekture zajema več kot samo majhno število ukazov. Ta pogoj se pravzaprav danes še najmanj upošteva
- Praktično vsi po letu 1990 razviti računalniki so tipa RISC
- To pa ne pomeni, da CISC računalnikov danes ni več (Intel 80x86)



- Intelu je uspelo ideje RISC arhitekture vgraditi v CISC arhitekturo Pentiuma
- Od procesorja 80486 dalje vsebujejo Intelovi procesorji RISC jedro, ki izvaja enostavne (in najpogostejše uporabljane) ukaze
- Bolj komplicirani in zmogljivejši ukazi pa se izvajajo na običajni CISC način z interpretiranjem na mikroprogramskem nivoju