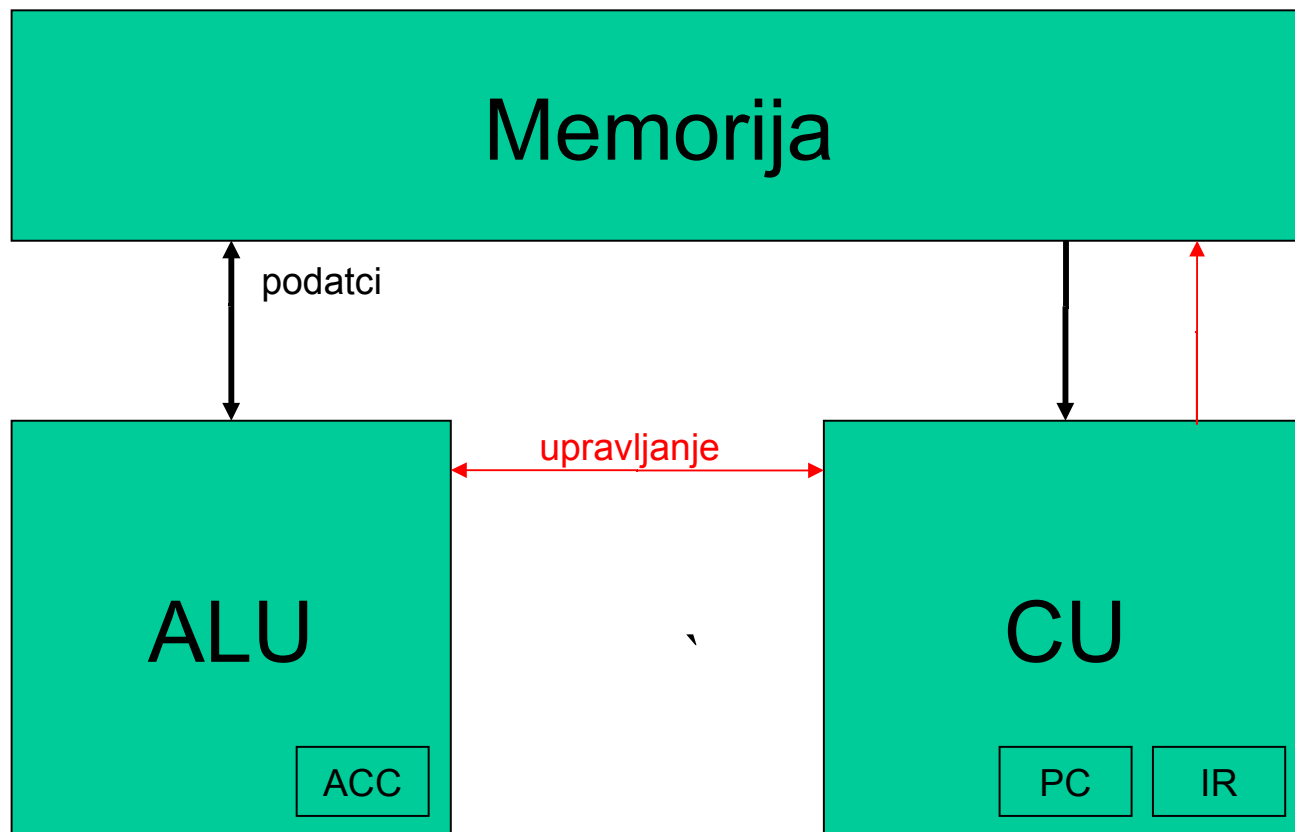
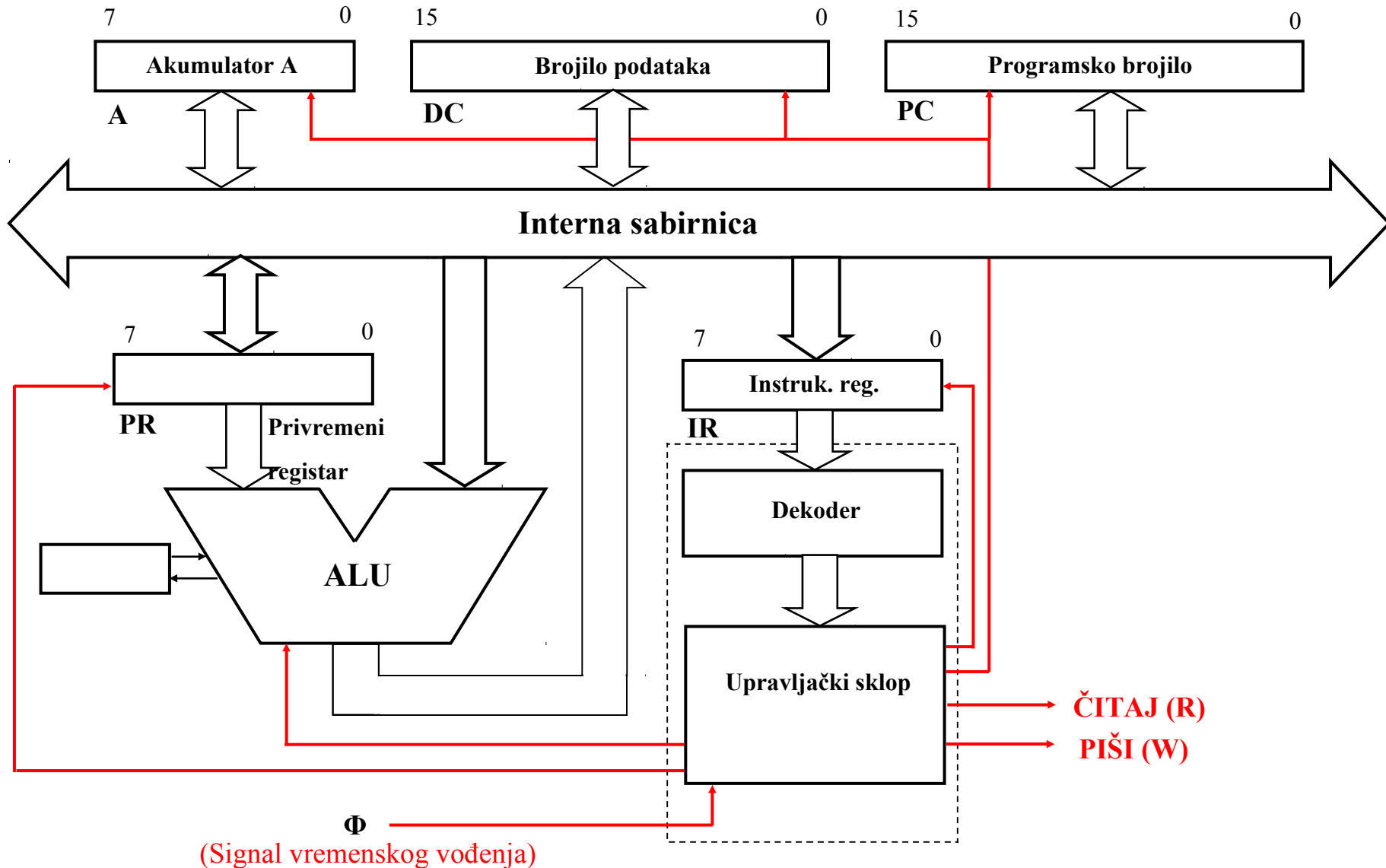


3. SKLOPOVSKA IZVEDBA UPRAVLJAČKE JEDINICE RAČUNALA S AKUMULATORSKOM ARHITEKTUROM

- Skup instrukcija
- Organizacija računala
- Struktura upravljačke jedinice
- Komponente upravljačke jedinice:
 - dekodeer
 - PLA
 - generator taktnog signala
 - generator slijedova

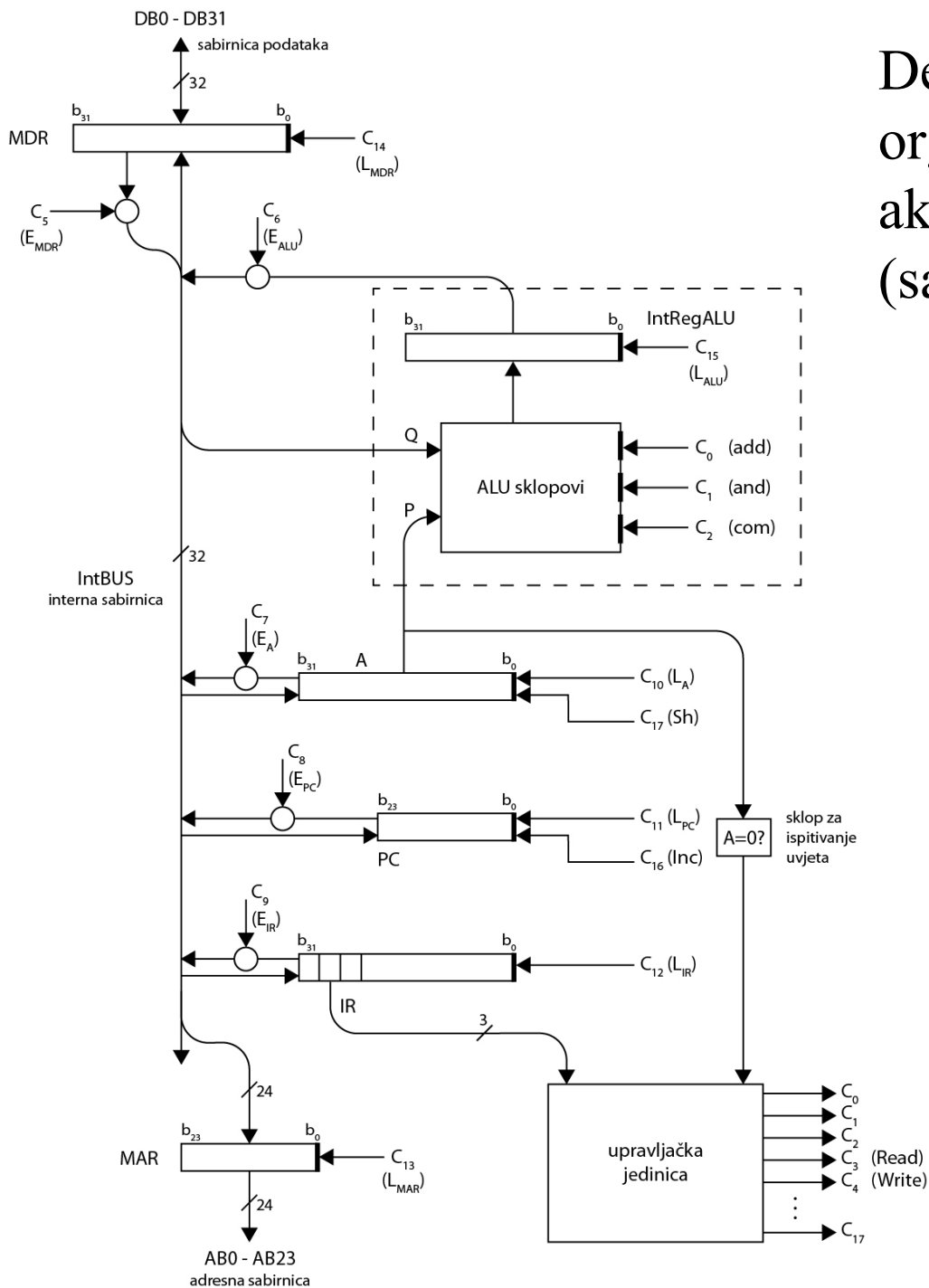


Von Neumannov model računala (drugo predavanje)



Pojednostavljeni model računala (treće predavanje)

S. Ribarić, AIOR

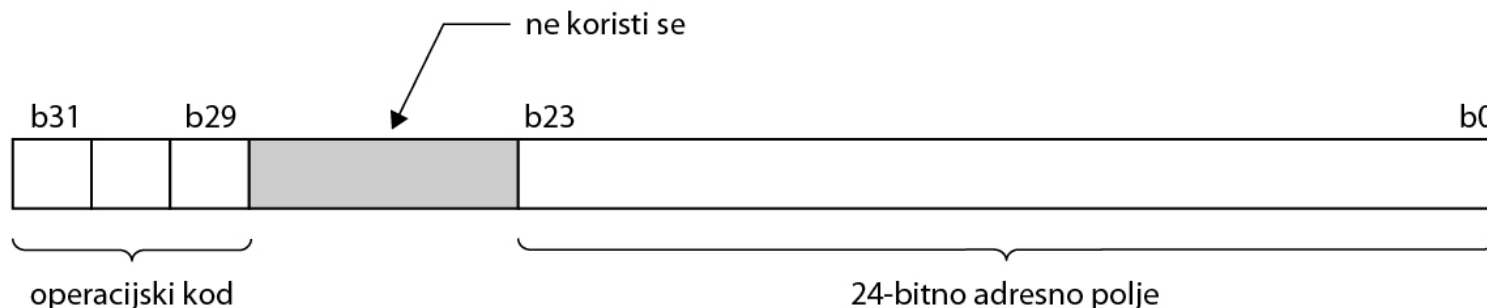


Detaljni model
organizacije računala
akumulatorske arhitekture
(sada razmatramo)

Instrukcijska arhitektura (ISA)

Mnemonik	Opis
lda X	$A \leftarrow M(X)$
sta X	$M(X) \leftarrow A$
adda X	$A \leftarrow A + M(X)$
anda X	$A \leftarrow A \wedge M(X)$
coma	$A \leftarrow \neg A$ (prvi komplement)
jmp X	$PC \leftarrow X$
jmpz X	if $AC == 0$: $PC \leftarrow X$
shra	$A[30:0] \leftarrow A[31:1]$

Fiksni instrukcijski format (za operacijski kod dovoljna tri bita)



Instrukcijski skup oblikovan za što veću jednostavnost :

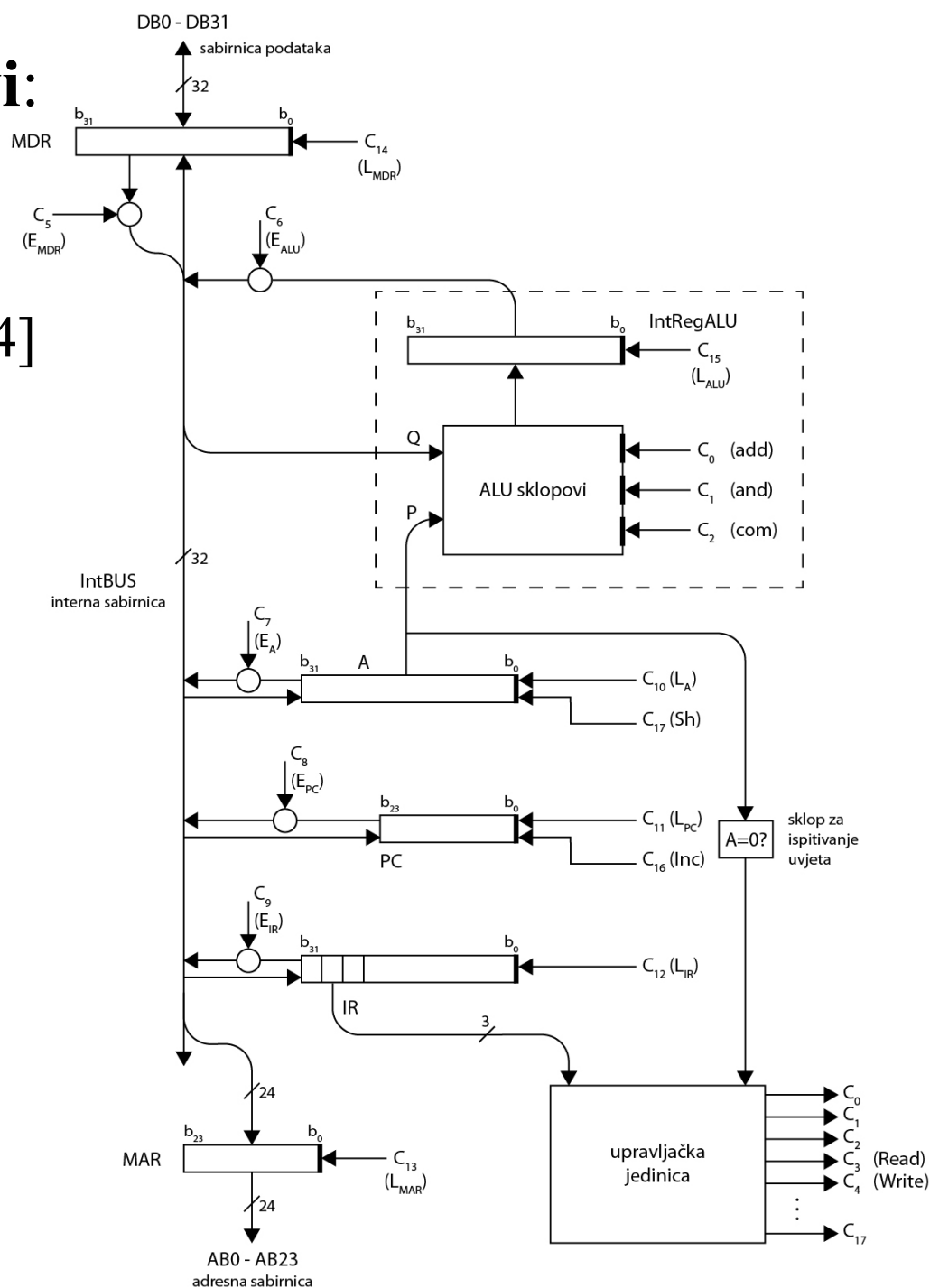
- sve instrukcije zauzimaju jednu riječ memorije (32 bit)
 - operacijski kod: 3 bita
 - memorijski operand (izravno adresiranje): 24 bita
- adresna sabirnica procesora ima 24 bita
- podatkovna sabirnica procesora ima 32 bita
- =32-bitno proširenje tipičnog osmobitnog procesora
 - jednostavna akumulatorski orijentirana arhitektura
 - faza pribavi je sada ista za sve instrukcije

Koraci provedbe faze **pribavi**:

1. $MAR \leftarrow PC$
2. $MDR \leftarrow M[MAR]$
3. $PC \leftarrow PC+1$, $IR \leftarrow MDR[31:24]$
4. dekodiranje operacijskog kôda

Koraci provedbe faze **izvrši** za instrukciju **adda**:

1. korak: $MAR \leftarrow MDR[23:0]$
2. korak: $MDR \leftarrow M[MAR]$
3. korak: $A \leftarrow A + MDR$



Detalji izvedbe koraka faza pribavi i izvrši

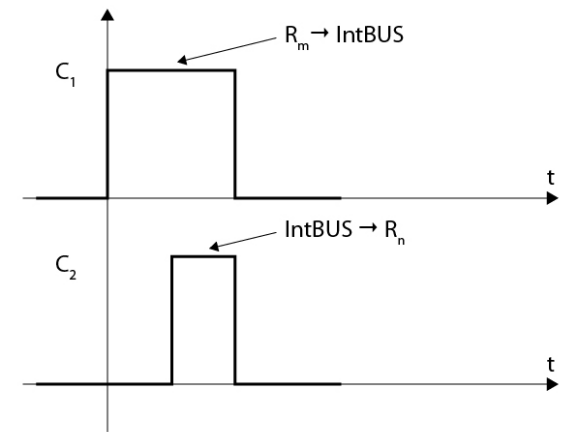
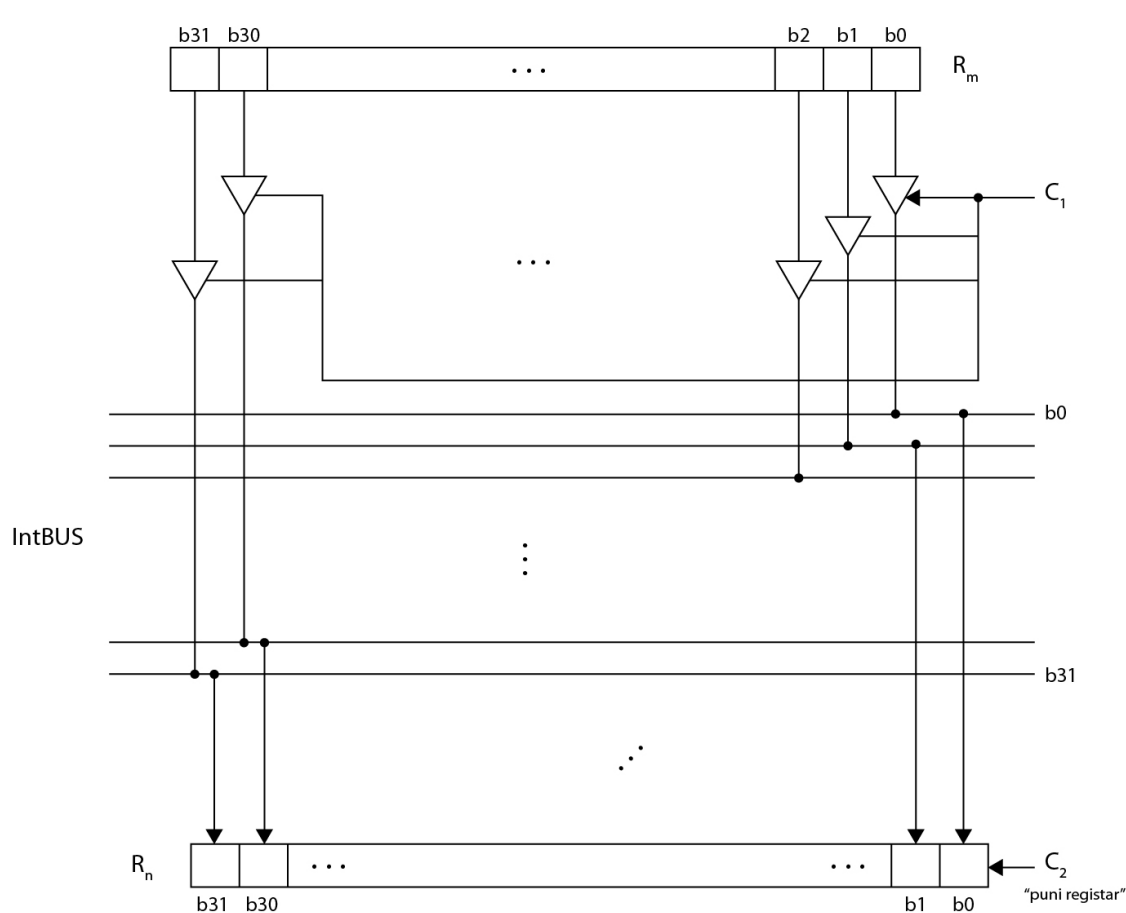
- svaki korak faza pribavi i izvrši izvodimo potrebnim brojem **mikrooperacija**
- mikrooperacija: nedjeljiva, primitivna, izravno sklopovski podržana operacija
- tipične mikrooperacije:
 1. prijenos podatka između dva registra
 2. pristup glavnoj memoriji
 3. elementarne aritmetičke operacije
- pretpostavljeni vremenski odnosi mikrooperacija:
 1. memorijske mikrooperacije zahtijevaju dvije vremenske jedinice ΔT
 2. prijenos podatka preko interne sabirnice također zahtijeva $2\Delta T$
 3. sve ostale mikrooperacije izvode se tijekom $1\Delta T$

Potrebna sklopovska podrška

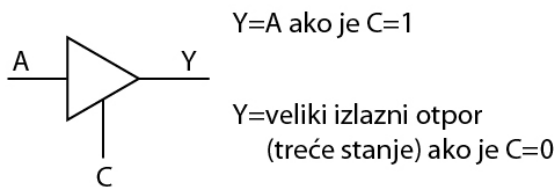
- prijenos podataka među registrima na zajedničkoj sabirnici
- mehanizam za slijeđenje mikrooperacija
- dekodiranje instrukcijskog registra
- aritmetičko-logička jedinica
 - to znamo iz Digitalne logike!
- izvedba glavne memorije
 - s detaljima ćemo se upoznati pri kraju semestra!
- izvedba vanjskog sabirničkog sustava i UI jedinica
 - to ćemo ostaviti za Projektiranje digitalnih sustava!

Prijenos podataka među registrima na sabirnici

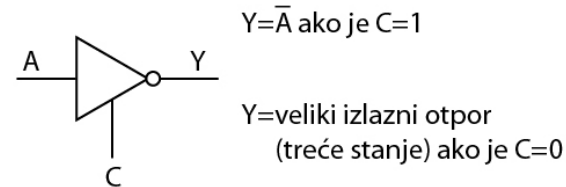
- signal C_1 propušta R_m na sabirnicu
- signal C_2 upisuje podatak sa sabirnice u R_n
- sinkroniziranom primjenom C_1 i C_2 postićemo $R_n \leftarrow R_m$



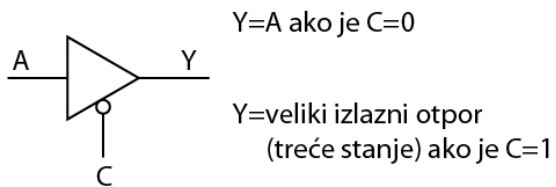
Pogonski sklopovi s tri stanja (ponavljanje)



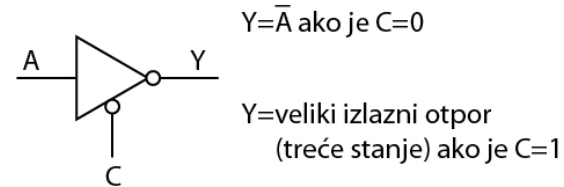
a1)



a2)

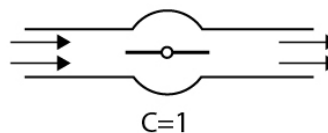
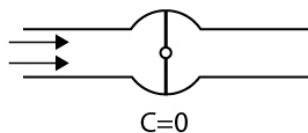
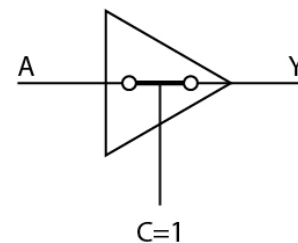
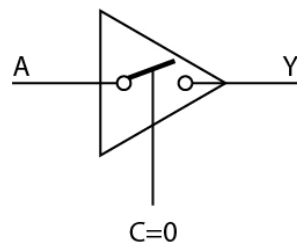


a3)



a4)

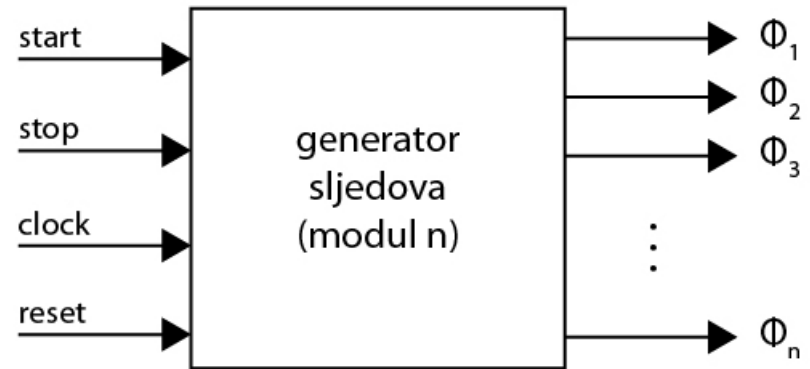
a)



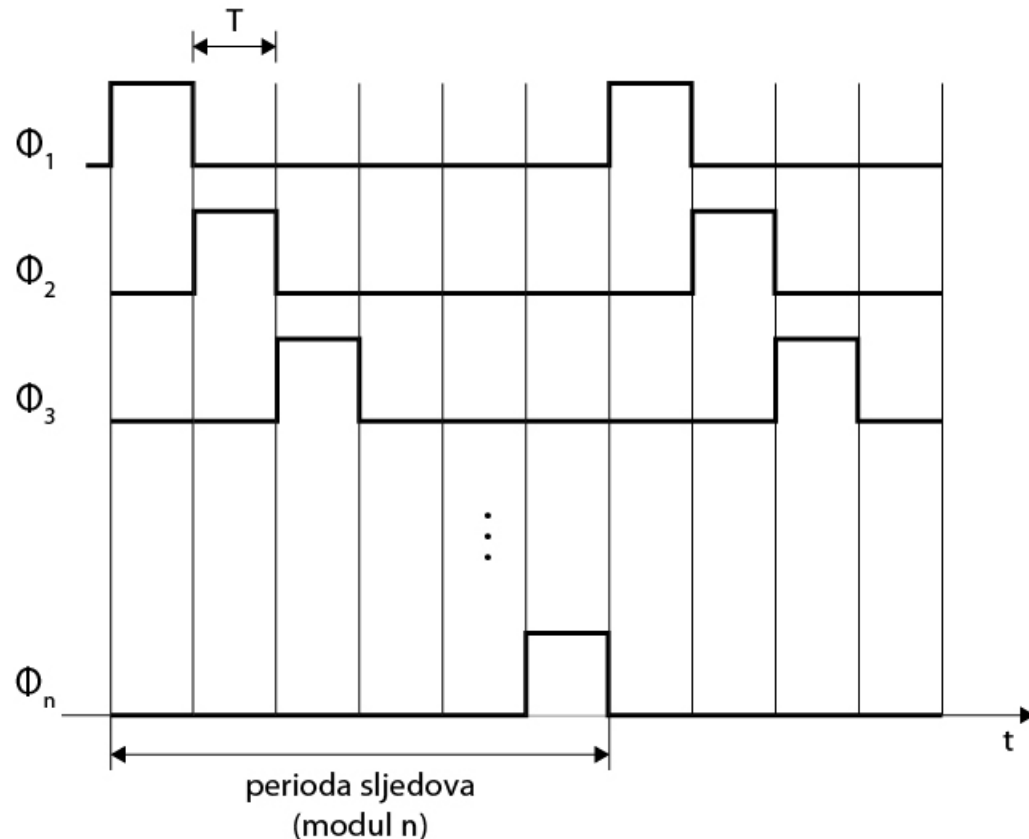
b)

Mehanizam za sljeđenje mikrooperacija

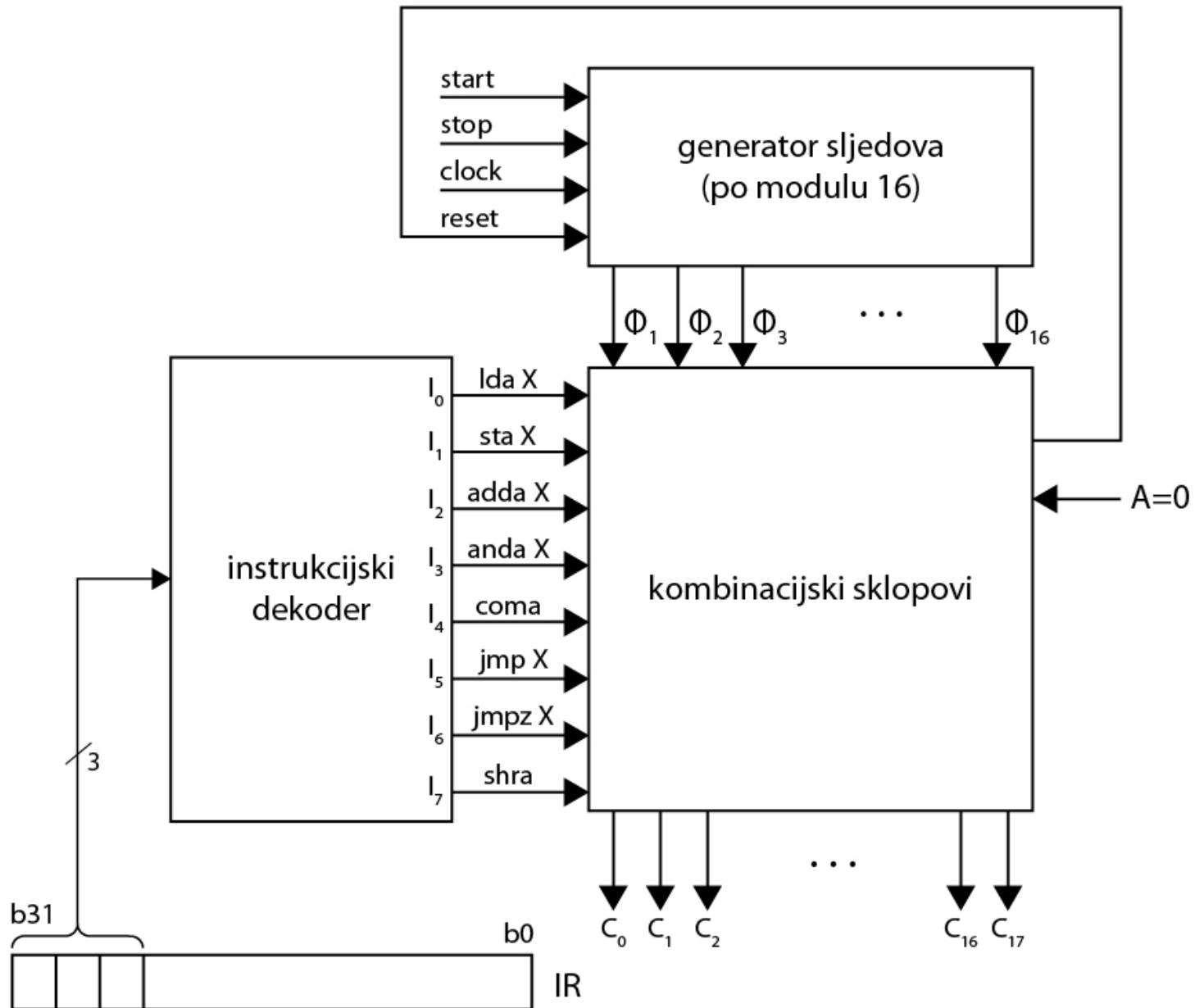
- osnovni sklop: generator sljedova po modulu n
- **pretp:** $R_n \leftarrow R_m$ je prva mikrooperacija instrukcije I_x
- **tada:** C_1 i C_2 izvodimo s:
 - $C_1 = (\Phi_1 + \Phi_2) \cdot I_x$
 - $C_2 = \Phi_2 \cdot I_x$
- kako dobiti signal I_x ?
(vidi sljedeću stranicu)



clock = signal vremenskog vođenja

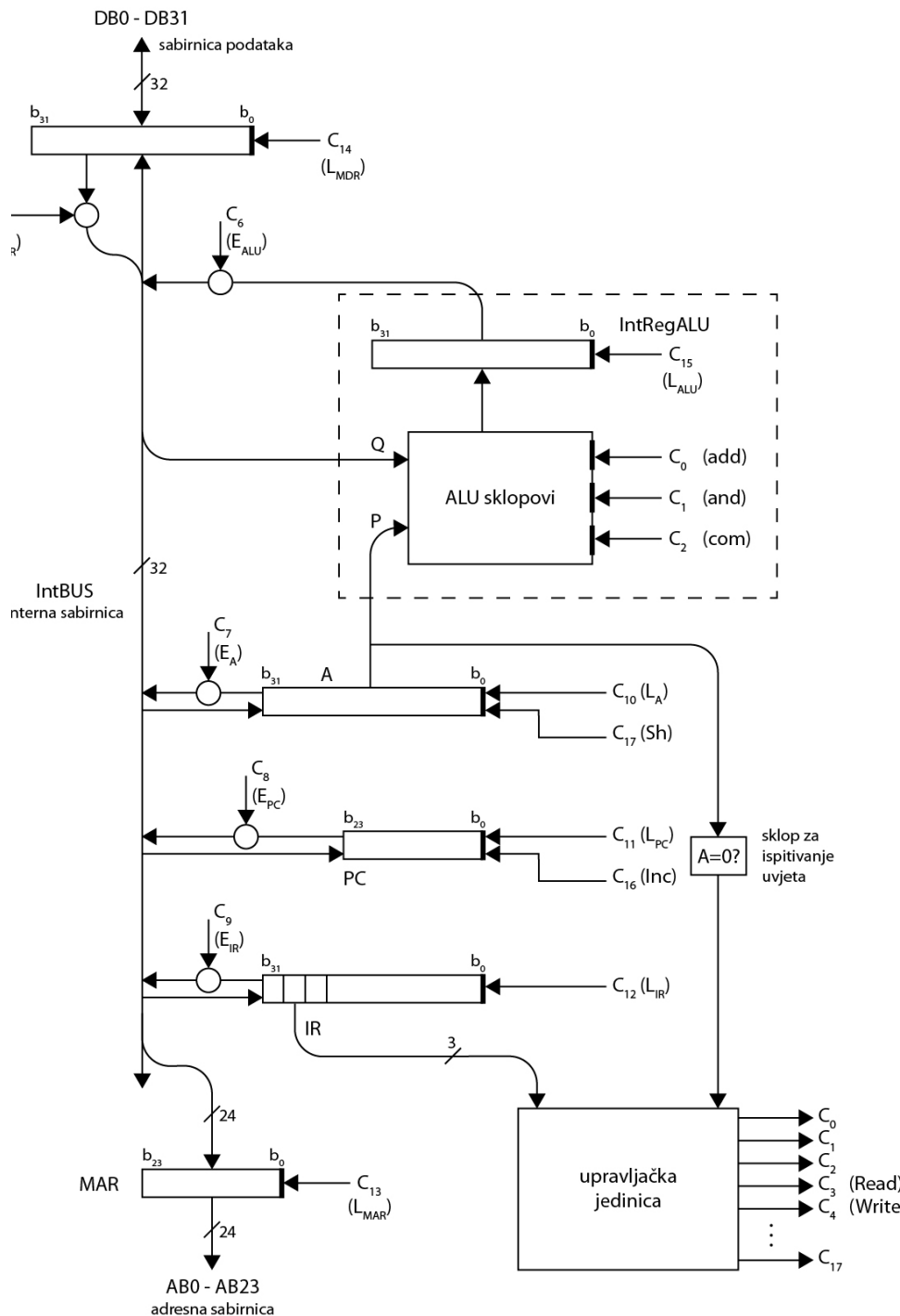


Dekodiranje instrukcija, generiranje upravljačkih signala



Novi pogled na put podataka

- signali C_{14} , C_{15} itd. upravljaju upisivanjem podataka u registre (sinkronizacijski ulazi registara spojeni su na signal takta)
- signali C_5 , C_6 , C_7 itd. definiraju izlazak registara na sabirnicu ($2\Delta T$)
- signali C_3 , C_4 reguliraju pristup glavnoj memoriji ($2\Delta T$)
- signali C_0 , C_1 , C_2 definiraju funkciju ALU



Signal	Simbol	Vrsta	Operacija
C0	add	ALU	$P + Q$; zbroji
C1	and	ALU	$P \cdot Q$
C2	com	ALU	$\neg P$
C3	Read	memorija	čitanje (Read)
C4	Write	memorija	pisanje (Write)
C5	EMDR	sabirnica	$MDR \rightarrow \text{IntBUS}$
C6	EALU	sabirnica	$\text{IntRegALU} \rightarrow \text{IntBUS}$
C7	EA	sabirnica	$A \rightarrow \text{IntBUS}$
C8	EPC	sabirnica	$PC \rightarrow \text{IntBUS}$
C9	EIR	sabirnica	$IR \rightarrow \text{IntBUS}$
C10	LA	registar	$\text{IntBUS} \rightarrow A$
C11	LPC	registar	$\text{IntBUS} \rightarrow PC$
C12	LIR	registar	$\text{IntBUS} \rightarrow IR$
C13	LMAR	registar	$\text{IntBUS} \rightarrow MAR$
C14	LMDR	registar	$\text{IntBUS} \rightarrow MDR$ ili $(DB0-DB31) \rightarrow MDR$
C15	LALU	registar	$ALU \rightarrow \text{IntRegALU}$
C16	Inc	registar	$PC + 1 \rightarrow PC$
C17	Shr	registar	posmak akumulatora

Detaljna izvedba faze pribavi

1. $\text{MAR} \leftarrow \text{PC}$

– EPC: $C_8 = \Phi_1 + \Phi_2$

– LMAR: $C_{13} = \Phi_2$

2. $\text{MDR} \leftarrow \text{M}[\text{MAR}]$

– READ: $C_3 = \Phi_3 + \Phi_4$

– LMDR: $C_{14} = \Phi_4$

3. $\text{PC} \leftarrow \text{PC}+1$, $\text{IR} \leftarrow \text{MDR}[31:24]$

– INC: $C_{16} = \Phi_5$

– EMDR: $C_5 = \Phi_5 + \Phi_6$

– LIR: $C_{12} = \Phi_6$

4. dekodiranje operacijskog kôda

– (Φ_7)

Detaljna izvedba faze izvrši za instrukciju **adda** (I_2)

1. $MAR \leftarrow MDR[23:0]$

– EMDR: $C_5 = I_2 \cdot (\Phi_8 + \Phi_9)$

– LMAR: $C_{13} = I_2 \cdot \Phi_9$

2. $MDR \leftarrow M[MAR]$

– READ: $C_3 = I_2 \cdot (\Phi_{10} + \Phi_{11})$

– LMDR: $C_{14} = I_2 \cdot \Phi_{11}$

3. $A \leftarrow A + MDR$

– EMDR: $C_5 = I_2 \cdot (\Phi_{12} + \Phi_{13})$

– ADD: $C_0 = I_2 \cdot (\Phi_{12} + \Phi_{13})$

– LALU: $C_{15} = I_2 \cdot \Phi_{13}$

– EALU: $C_6 = I_2 \cdot (\Phi_{14} + \Phi_{15})$

– LA: $C_{10} = I_2 \cdot \Phi_{15}$

izvedbe instrukcija

lda i **anda**

vrlo su slične

pa ih ostavljamo

za vježbu!

Detaljna izvedba faze izvrši za instrukciju **sta** (I_1)

1. $MAR \leftarrow MDR[23:0]$
 - EMDR: $C_5 = I_1 \cdot (\Phi_8 + \Phi_9)$
 - LMAR: $C_{13} = I_1 \cdot \Phi_9$
2. $MDR \leftarrow A$
 - EA: $C_7 = I_1 \cdot (\Phi_{10} + \Phi_{11})$
 - LMDR: $C_{14} = I_1 \cdot \Phi_{11}$
3. $M[MAR] \leftarrow MDR$
 - WRITE: $C_4 = I_1 \cdot (\Phi_{12} + \Phi_{13})$

Detaljna izvedba faze izvrši za instrukciju **shra** (I_7)

1. $A \leftarrow shr(A)$
 - SHR: $C_{17} = I_7 \cdot \Phi_8$

Detaljna izvedba faze izvrši za instrukciju **jmpz** (I_6)

1. $PC \leftarrow MDR[23:0]$, ali samo ako $A=0$

- EMDR: $C_5 = I_6 \cdot (\Phi_8 + \Phi_9)$
- LPC: $C_{11} = I_6 \cdot \Phi_9 \cdot (A=0)$

izvedba instrukcije **jmp**
vrlo je slična, pa je
ostavljamo za vježbu!

Detaljna izvedba faze izvrši za instrukciju **coma** (I_5)

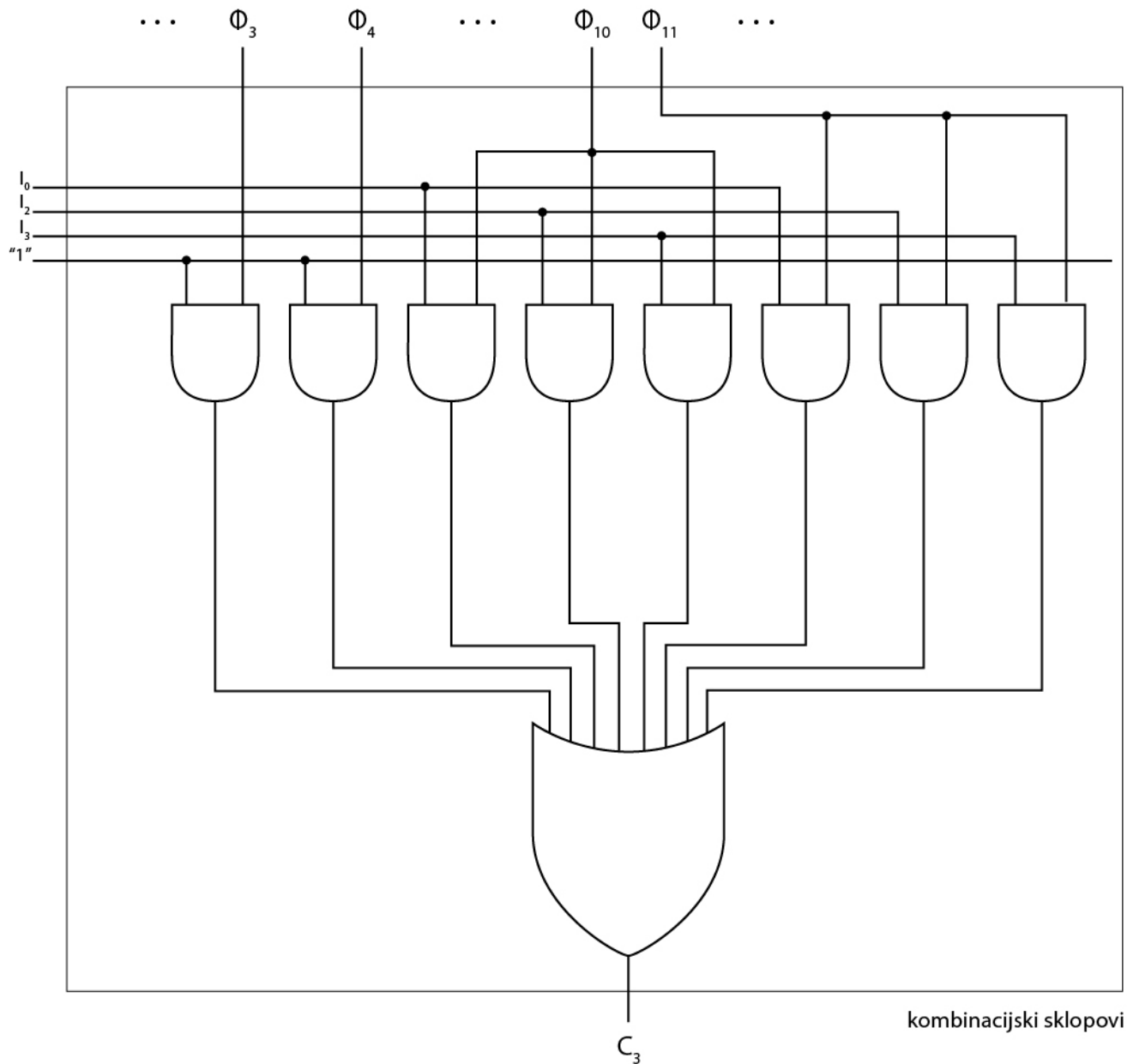
1. $A \leftarrow \neg A$

- COM: $C_2 = I_5 \cdot \Phi_8$
- LALU: $C_{15} = I_5 \cdot \Phi_8$
- EALU: $C_6 = I_5 \cdot (\Phi_9 + \Phi_{10})$
- LA: $C_{10} = I_5 \cdot \Phi_{10}$

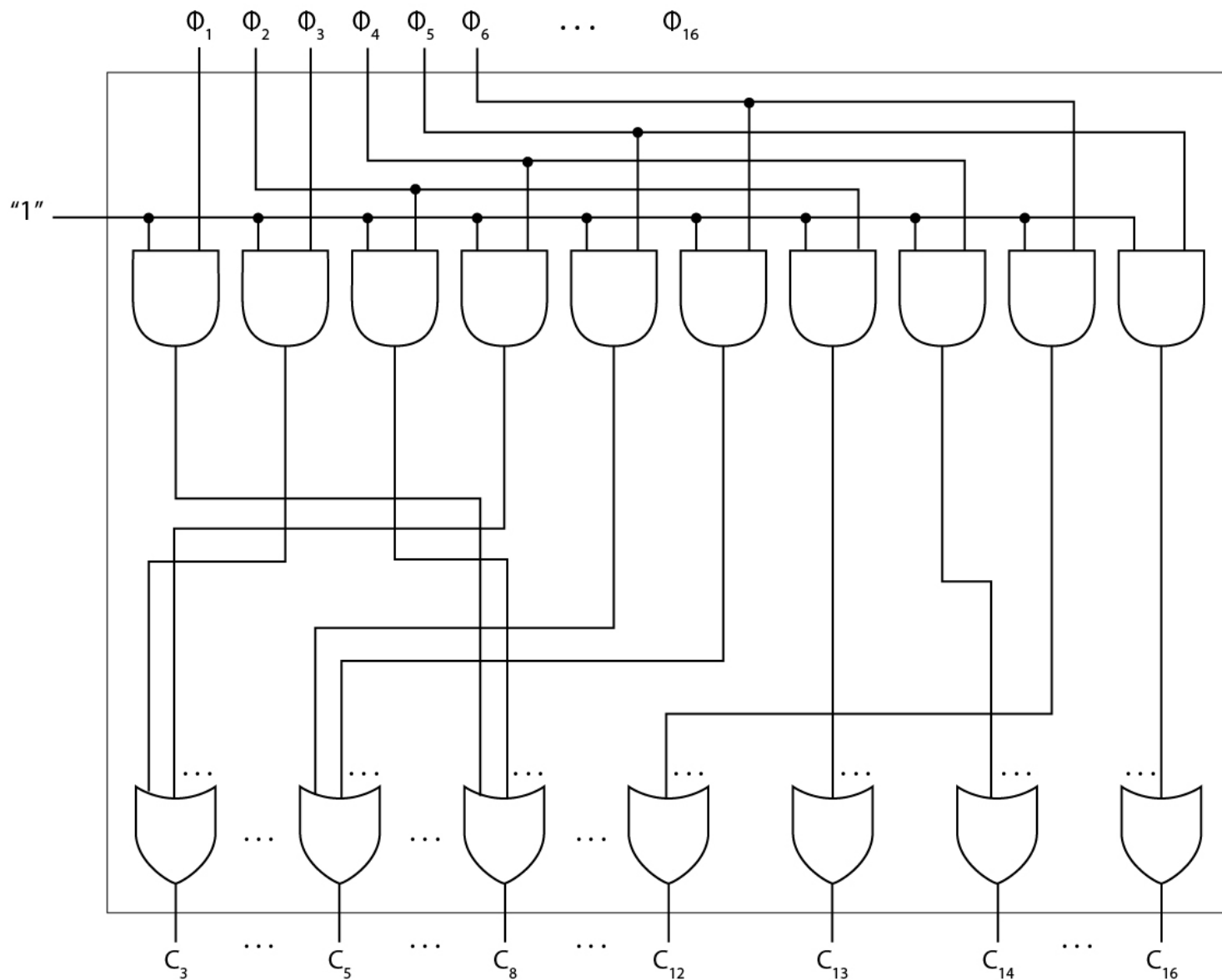
Instrukcijski ciklus (sažetak)

- periodima instrukcijskog ciklusa dodijeljeni su zasebni signali Φ_1 - Φ_{15} (generira ih **generator slijedova**)
- instrukcijski ciklus sastoji se od ukupno 8-15 perioda takta ΔT (nakon kratkih instrukcija generator slijedova postavljamo na Φ_1)
- mikrooperacije traju po jedan period takta, osim pristupa memoriji (C_3 , C_4) te izlaska na sabirnicu (C_5 , C_6 , ...) koji traju dvostruko više
- instrukciju definira razdioba mikrooperacija po periodima instrukcijskog ciklusa (više mikrooperacija se mogu izvoditi usporedno)
- instrukcije s koje pribavljaju operand iz memorije tipično traju duže od ostalih instrukcija

Potpuna implementacija upravljačkog signala C_3



Implementacija upravljačkih signala za fazu pribavi



Upravljački signal C_i ($i = 0, 1, 2, \dots, 12$) može se opisati logičkom jednažbom:

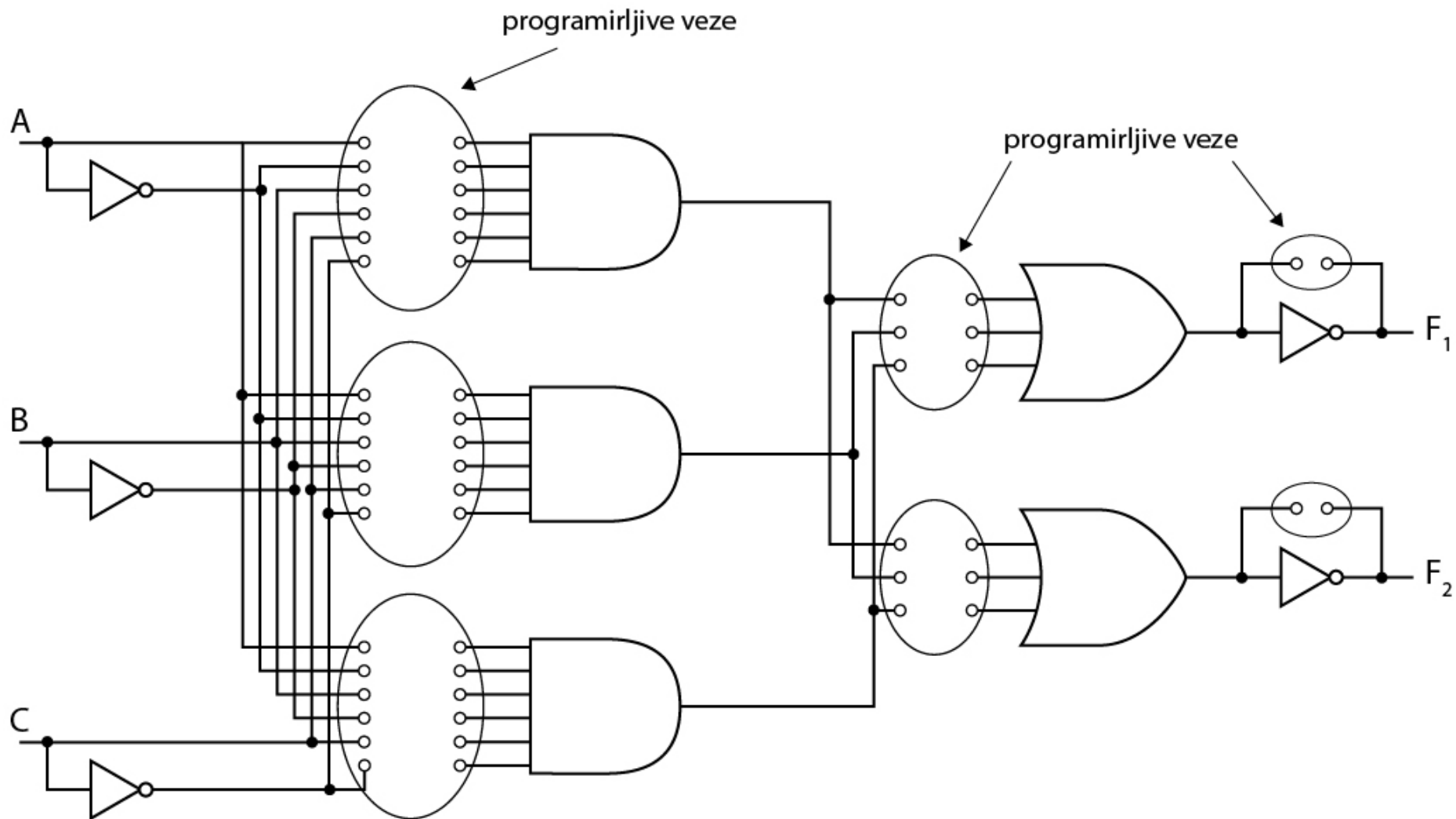
$$C_i = \sum_j \left(\Phi_j \cdot \sum_m I_m \right)$$

$j = 1, 2, \dots, 15$

$m = 1, 2, \dots, 8$

I_m – izlaz iz dekodera instrukcija

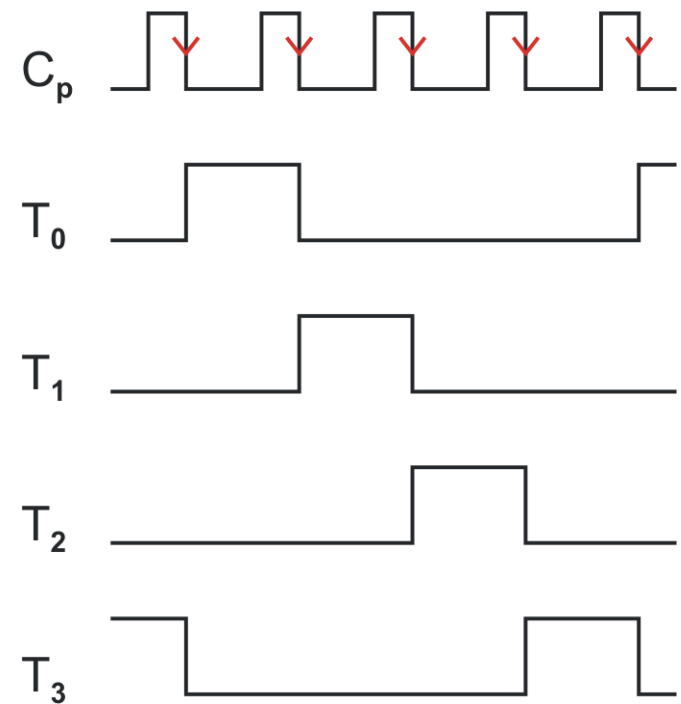
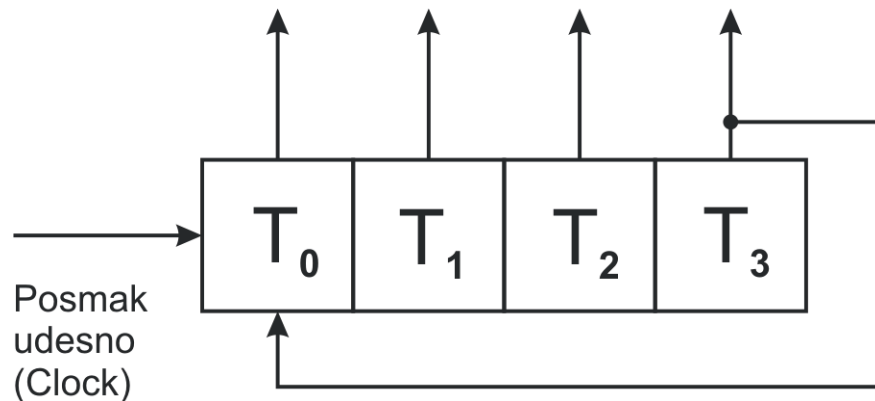
Programirljivi sklop PLA



S. Ribarić, AIOR

Izvedba brojila sekvenci (ideja 1):

- posmačni registar spojen u prstenasto brojilo (engl. Ring-counter)
- pri pokretanju paralelno upisujemo podatak **10...0**

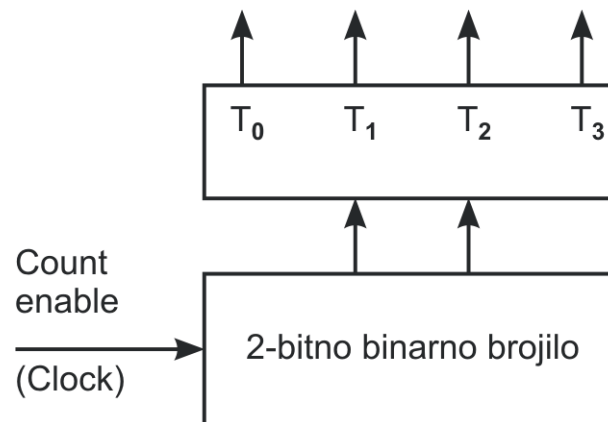


Izvedba brojila sekvenci (ideja 2):

- binarno brojilo spojeno na dekodeer
- pri pokretanju brojilo resetiramo

Dekoder:

Brojilo:



Brojilo sekvenci s mogućnošću zaustavljanja:

