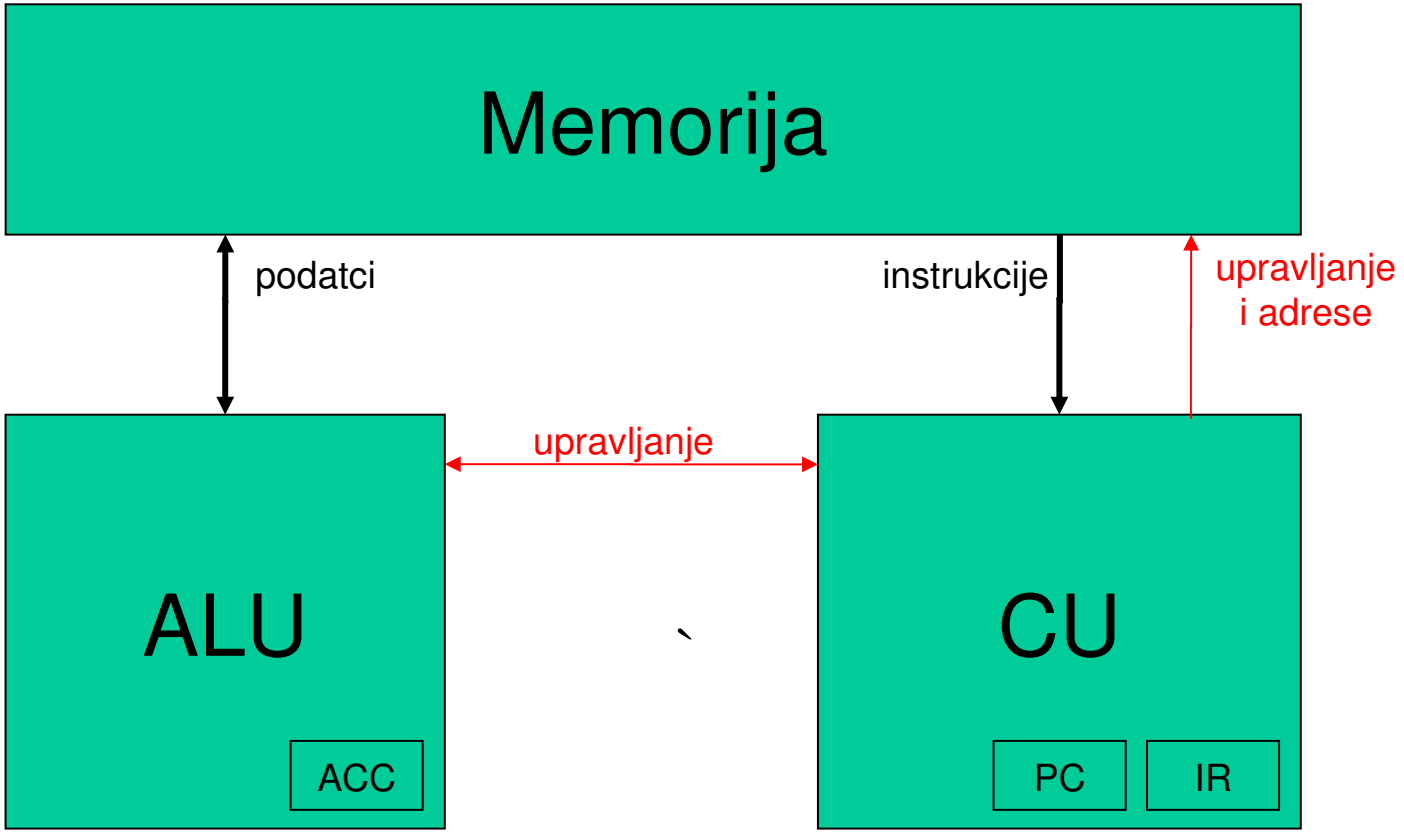
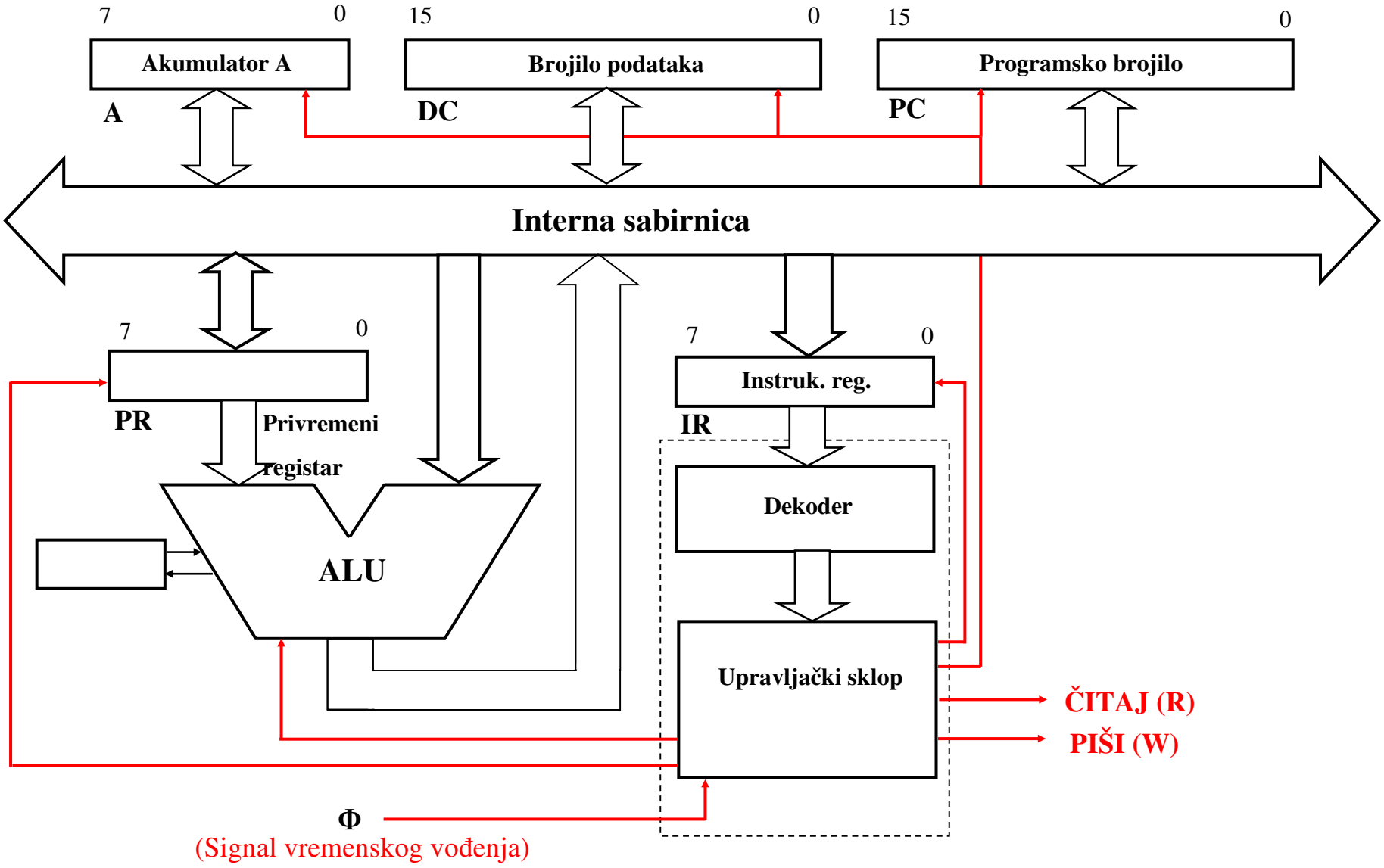
SKLOPOVSKA IZVEDBA UPRAVLJAČKE JEDINICE RAČUNALA S AKUMULATORSKOM ARHITEKTUROM

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

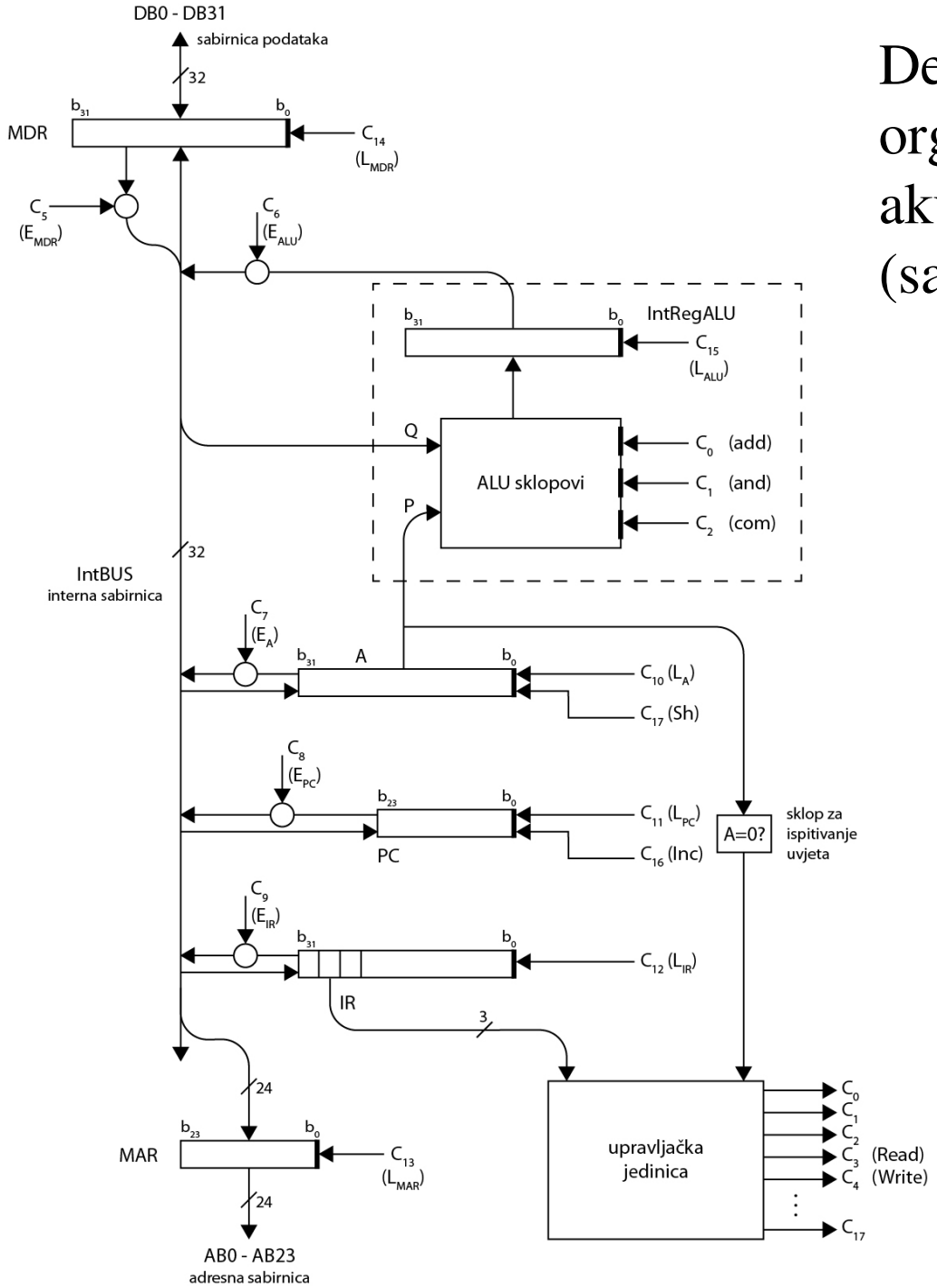


Von Neumannov model: gruba arhitektura računala s pohranjenim programom



Pojednostavnjeni model računala: organizacija procesora
(CU, ALU, put podataka)

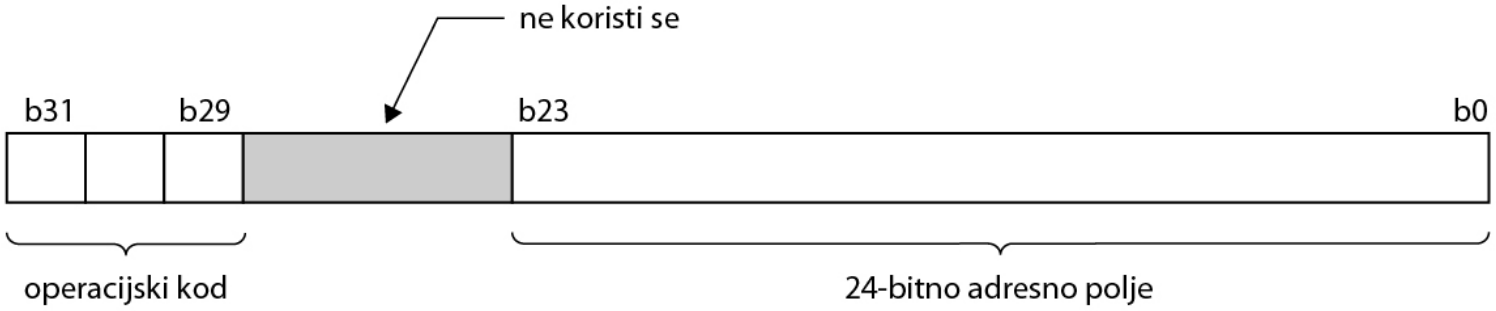
Detaljni model organizacije računala akumulatorske arhitekture (sada razmatramo)



Instrukcijska arhitektura (ISA)

Mnemonik	Opis
`lda X`	`A ← M(X)`
`sta X`	`M(X) ← A`
`adda X`	`A ← A + M(X)`
`anda X`	`A ← A ∧ M(X)`
`coma`	`A ← ¬A` (prvi komplement)
`jmp X`	`PC ← X`
`jmpz X`	`if AC==0: PC ← X`
`shra`	`A[30:0] ← A[31:1]`

Fiksni instrukcijski format (za operacijski kod dovoljna tri bita)



Instrukcijski skup oblikovan za što veću jednostavnost :

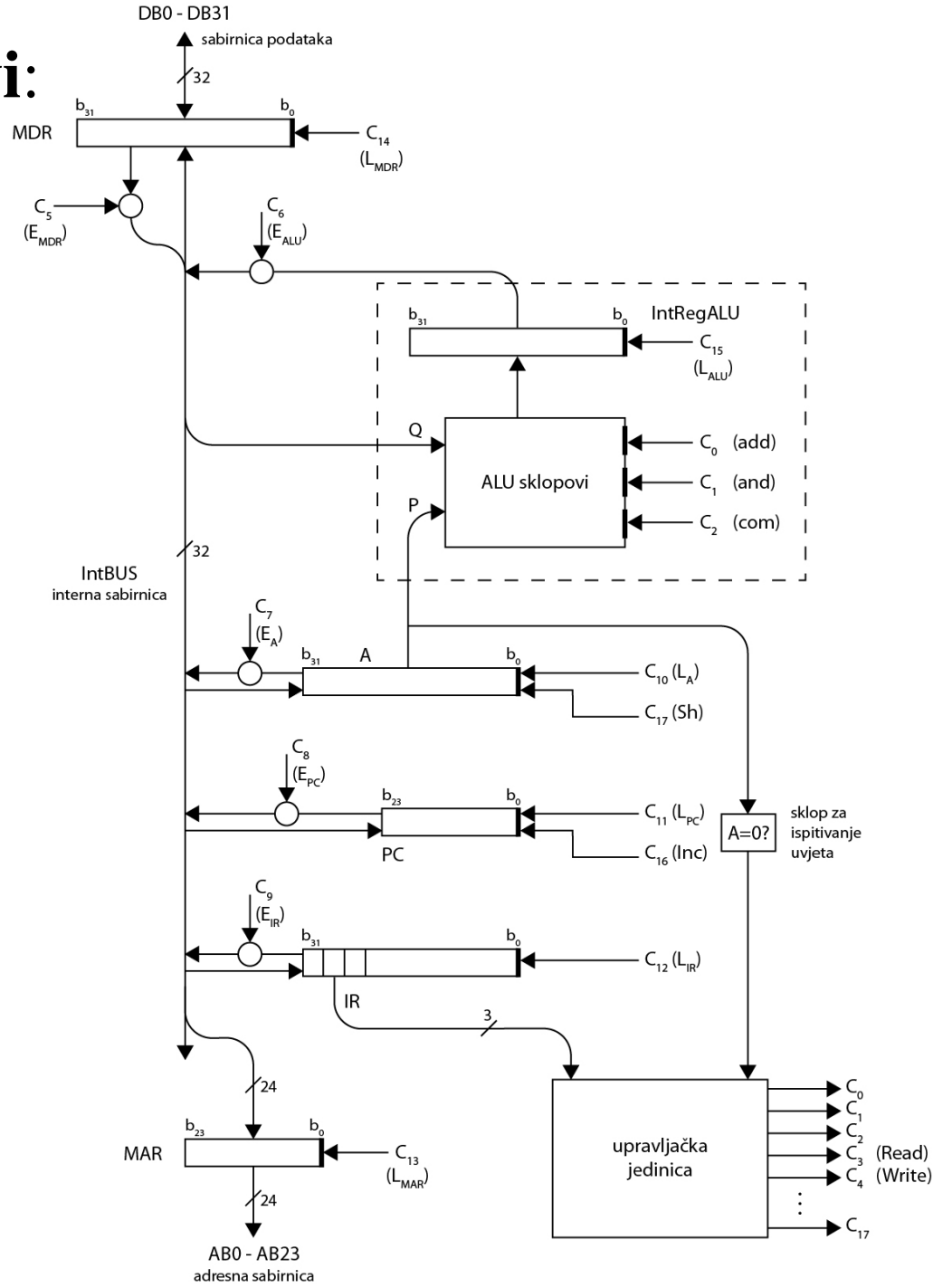
- sve instrukcije zauzimaju jednu riječ memorije (32 bit)
  - operacijski kod: 3 bita
  - adresa operanda (izravno adresiranje): 24 bita
- adresna sabirnica procesora ima 24 bita
- podatkovna sabirnica procesora ima 32 bita
- =32-bitno proširenje tipičnog osmobitnog procesora
  - jednostavna akumulatorski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

Koraci provedbe faze **pribavi**:

1. MAR ← PC
2. MDR ← M[MAR]
3. PC ← PC+1, IR ← MDR
4. dekodiranje operacijskog kôda

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

1. korak: MAR ← MDR[23:0]
2. korak: MDR ← M[MAR]
3. korak: A ← A + MDR



Detalji izvedbe koraka faza pribavi i izvrši

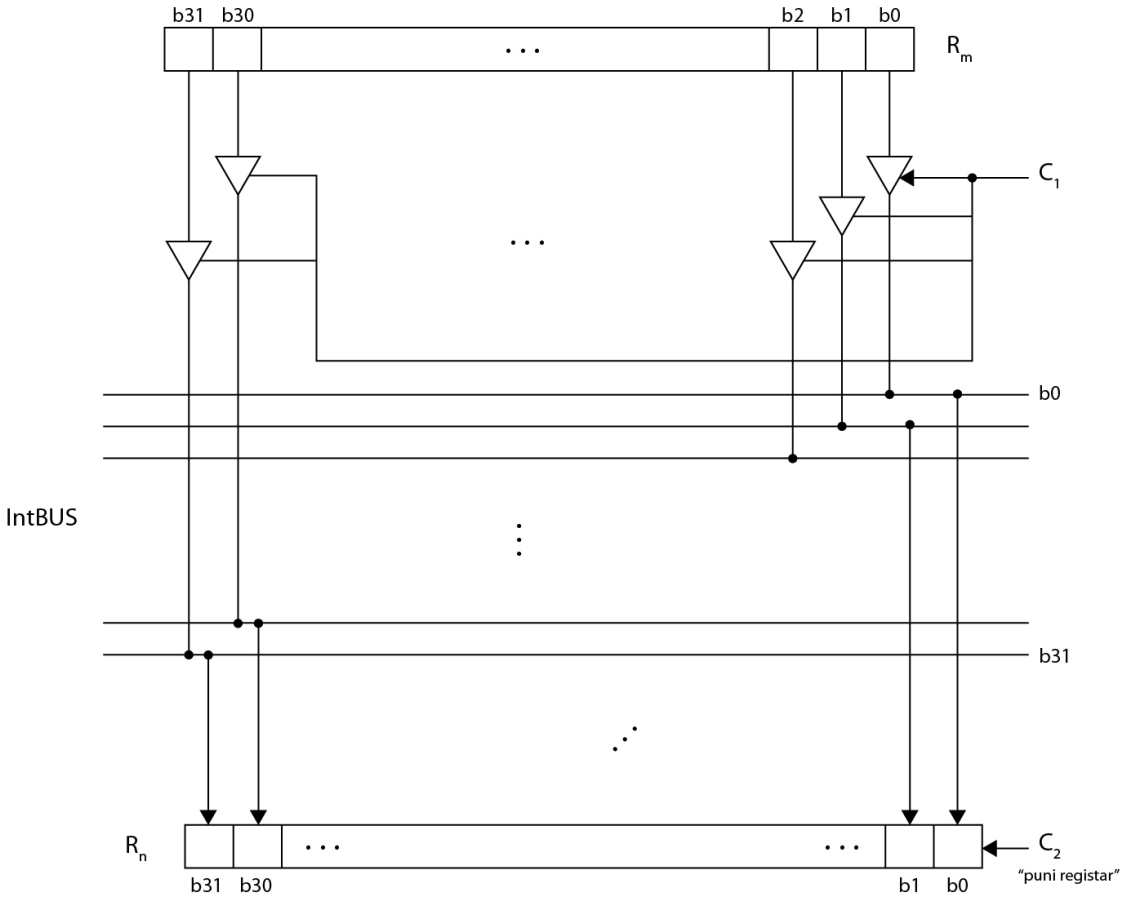
- svaki korak faza pribavi i izvrši izvodimo potrebnim brojem **mikrooperacija**
- mikrooperacija: nedjeljiva, primitivna, izravno sklopovski podržana operacija
- tipične mikrooperacije:
    1. prijenos podatka između dva registra
    2. pristup glavnoj memoriji
    3. elementarne aritmetičke operacije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

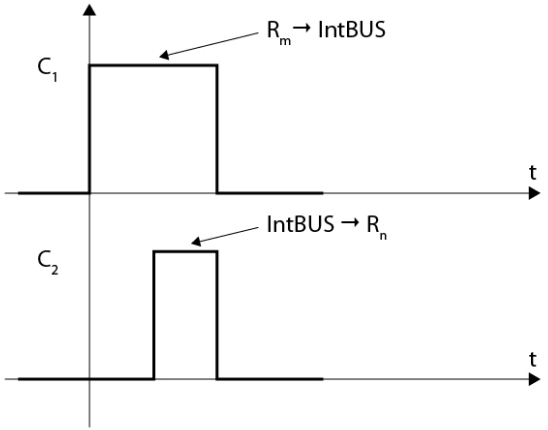
Potrebna sklopovska podrška

- 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!

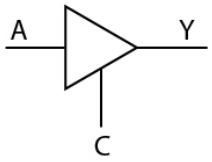
Prijenos podataka među registrima na sabirnici

- 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 postižemo $R_n \leftarrow R_m$





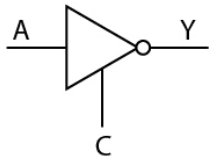
Pogonski sklopovi s tri stanja (ponavljanje)



Y=A ako je C=1

Y=veliki izlazni otpor (treće stanje) ako je C=0

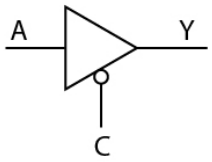
a1)



Y=$\overline{A}$ ako je C=1

Y=veliki izlazni otpor (treće stanje) ako je C=0

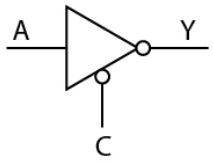
a2)



Y=A ako je C=0

Y=veliki izlazni otpor (treće stanje) ako je C=1

a3)

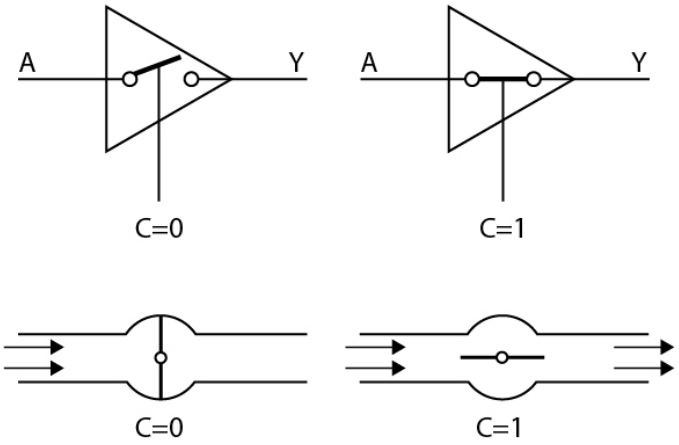


Y=$\overline{A}$ ako je C=0

Y=veliki izlazni otpor (treće stanje) ako je C=1

a4)

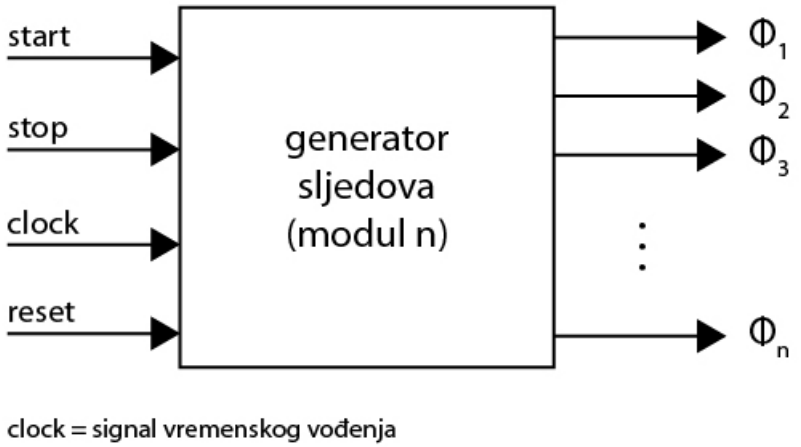
a)



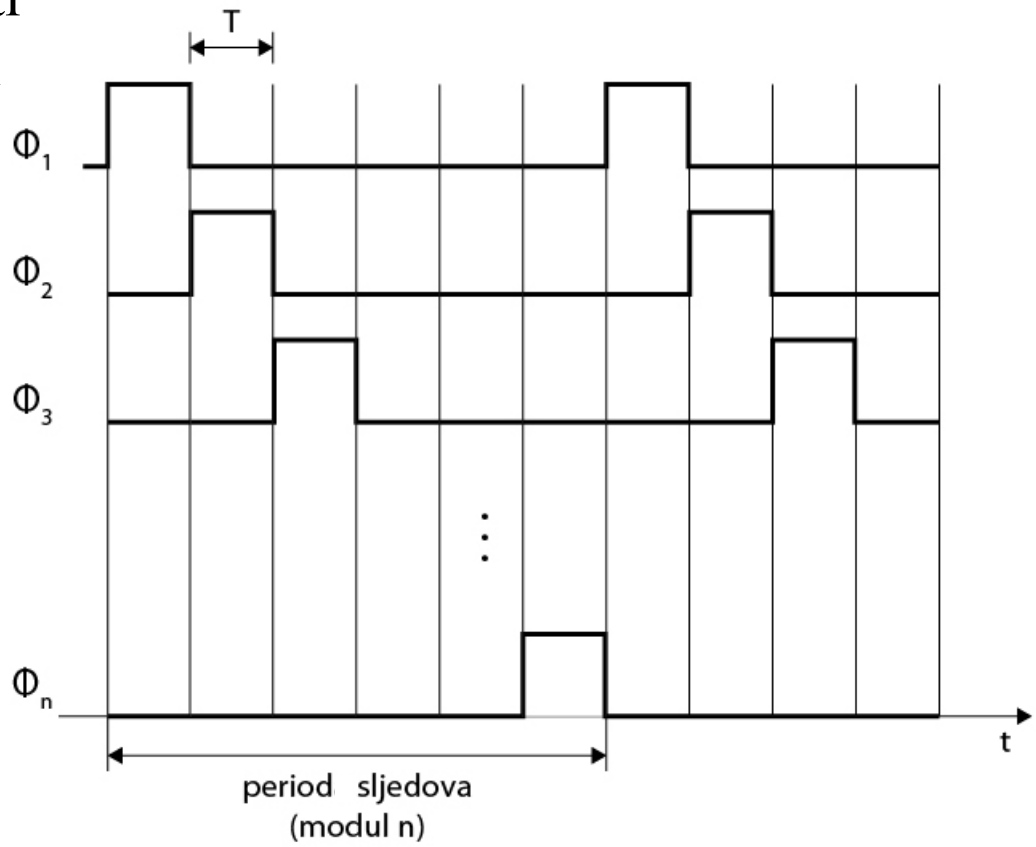
b)

Mehanizam za slijeđenje mikrooperacija

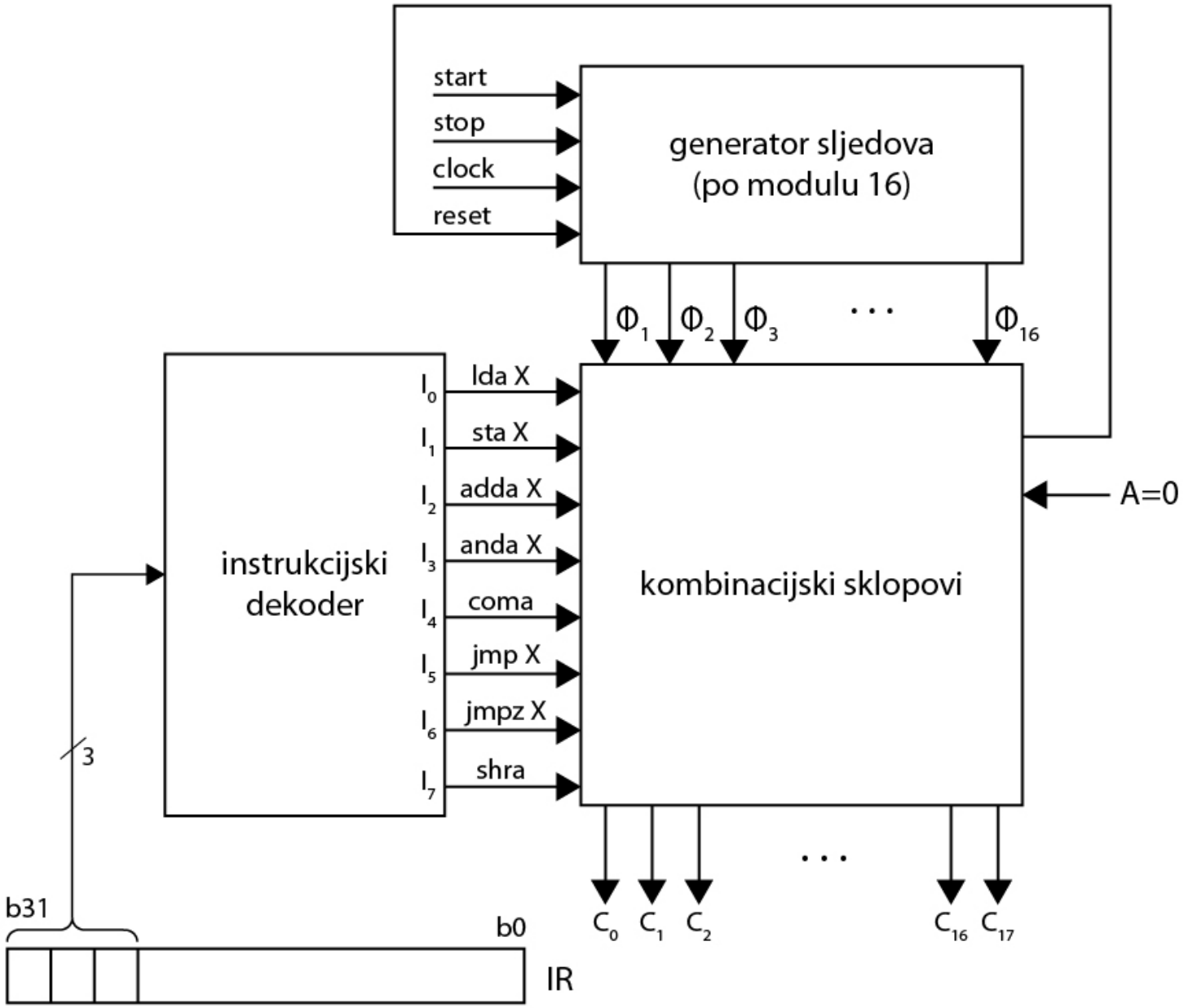
- osnovni sklop: generator slijedova po modulu n
- **pretp:** mikrooperacija $R_n \leftarrow R_m$ ($2\Delta T$) se treba dogoditi u i-tom taktu ciklusa instrukcije I_x
- **tada:** $C_{m\,r}$ i $C_{n\,w}$ izvodimo s:
  - $C_{m\,r} = (\Phi_i + \Phi_{i+1}) \cdot I_x$
  - $C_{n\,w} = \Phi_{i+1} \cdot I_x$
- kako dobiti signal I_x? (vidi sljedeću stranicu)



clock = signal vremenskog vođenja

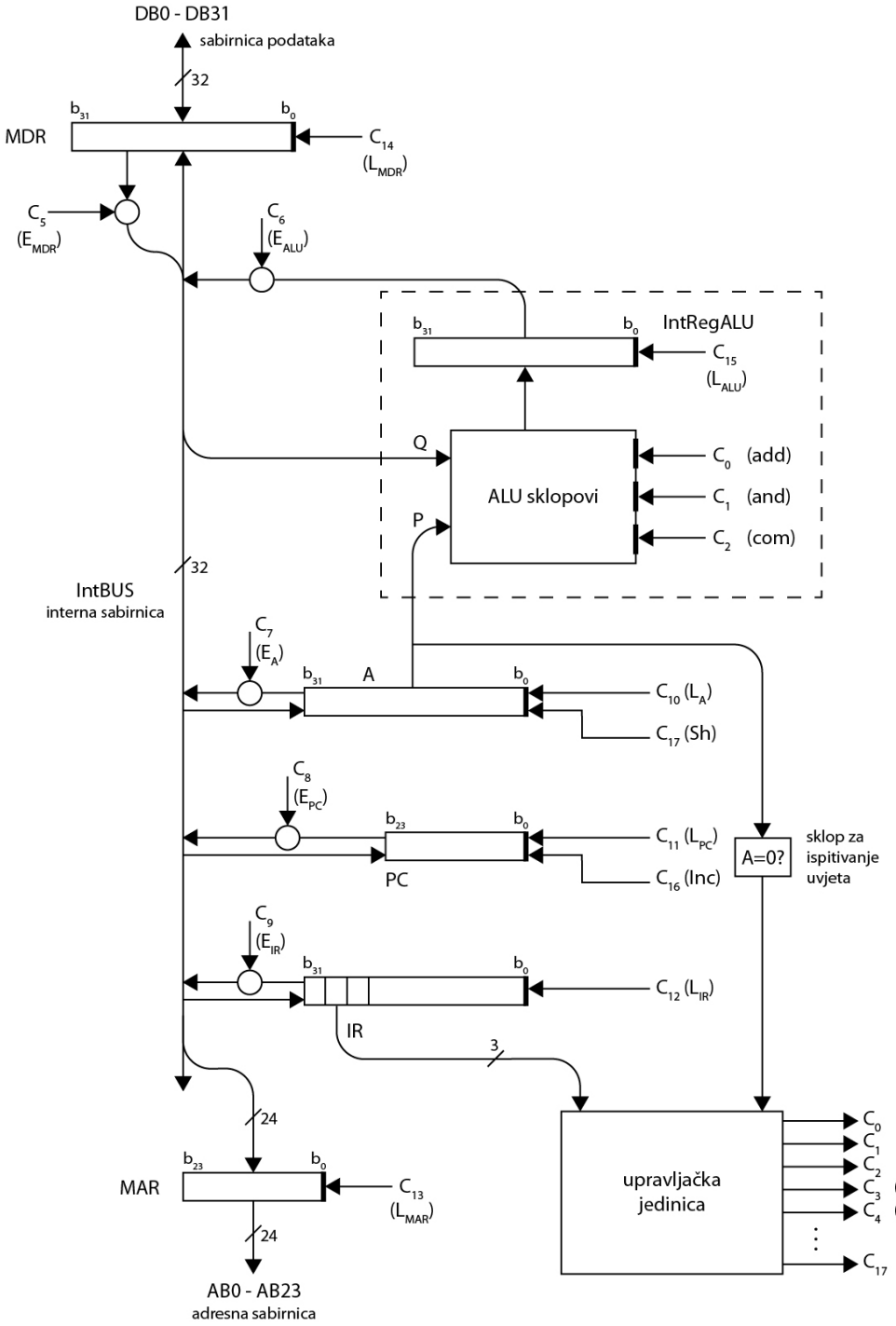


Dekodiranje instrukcija, generiranje upravljačkih signala



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
- ad hoc signali C_{16} (++PC) i C_{17} (A>>1)



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

Detaljna izvedba faze pribavi

1. MAR ← PC
   - EPC: $C_8 = \Phi_1 + \Phi_2$
   - LMAR: $C_{13} = \Phi_2$
2. MDR ← M[MAR]
   - READ: $C_3 = \Phi_3 + \Phi_4$
   - LMDR: $C_{14} = \Phi_4$
3. PC ← PC+1, IR ← 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 ← MDR[23:0]
   - EMDR: $C_5 = I_2 \cdot (\Phi_8 + \Phi_9)$
   - LMAR: $C_{13} = I_2 \cdot \Phi_9$
2. MDR ← M[MAR]
   - READ: $C_3 = I_2 \cdot (\Phi_{10} + \Phi_{11})$
   - LMDR: $C_{14} = I_2 \cdot \Phi_{11}$
3. A ← 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 ← MDR[23:0]
   - EMDR: $C_5 = I_1 \cdot (\Phi_8 + \Phi_9)$
   - LMAR: $C_{13} = I_1 \cdot \Phi_9$
2. MDR ← A
   - EA: $C_7 = I_1 \cdot (\Phi_{10} + \Phi_{11})$
   - LMDR: $C_{14} = I_1 \cdot \Phi_{11}$
3. M[MAR] ← MDR
   - WRITE: $C_4 = I_1 \cdot (\Phi_{12} + \Phi_{13})$

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

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

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

1. PC ← 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!

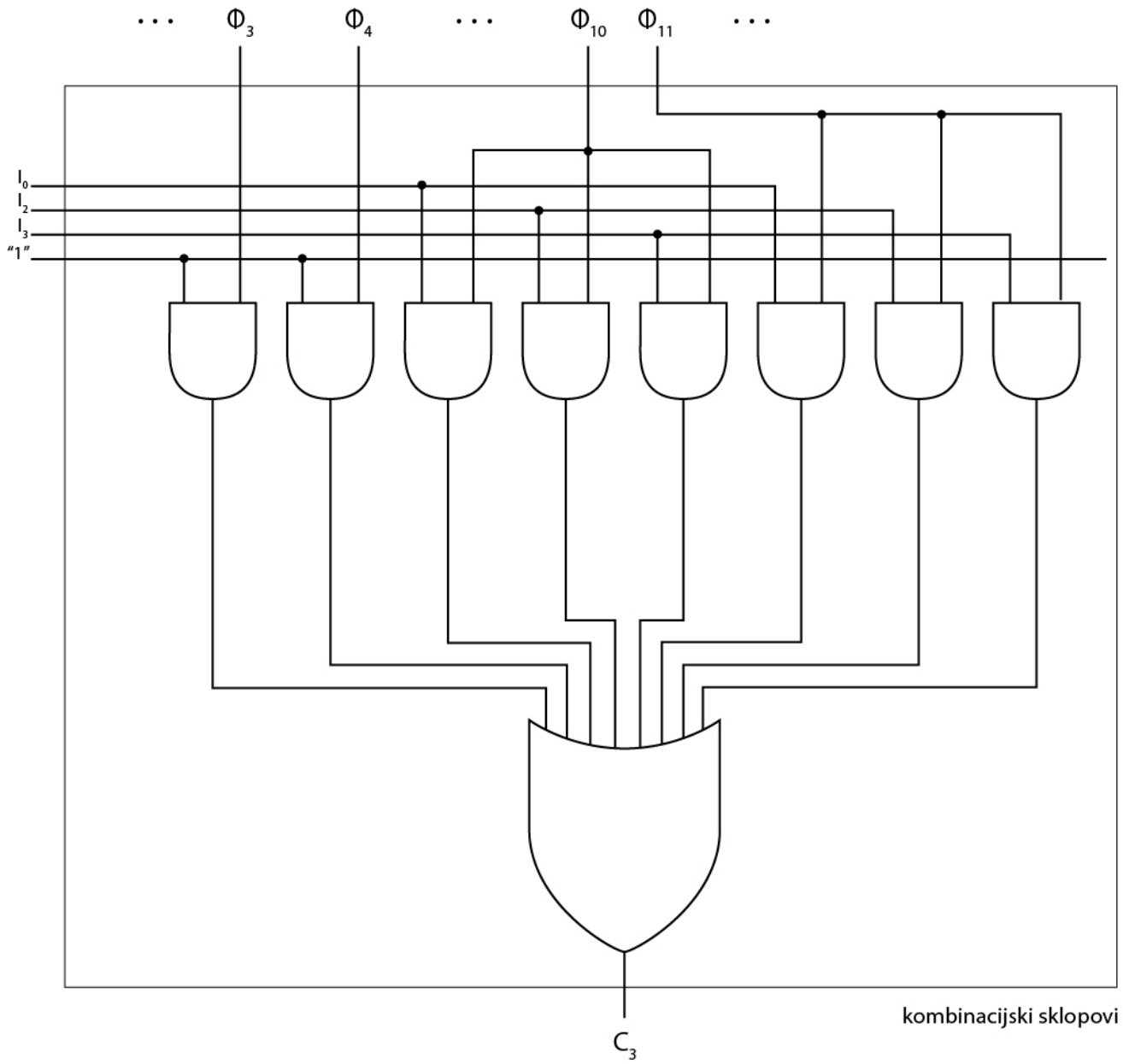
Detaljna izvedba faze izvrši za instrukciju **coma** (I_5)

1. A ← ⌐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: $C_{10} = I_5 \cdot \Phi_{10}$

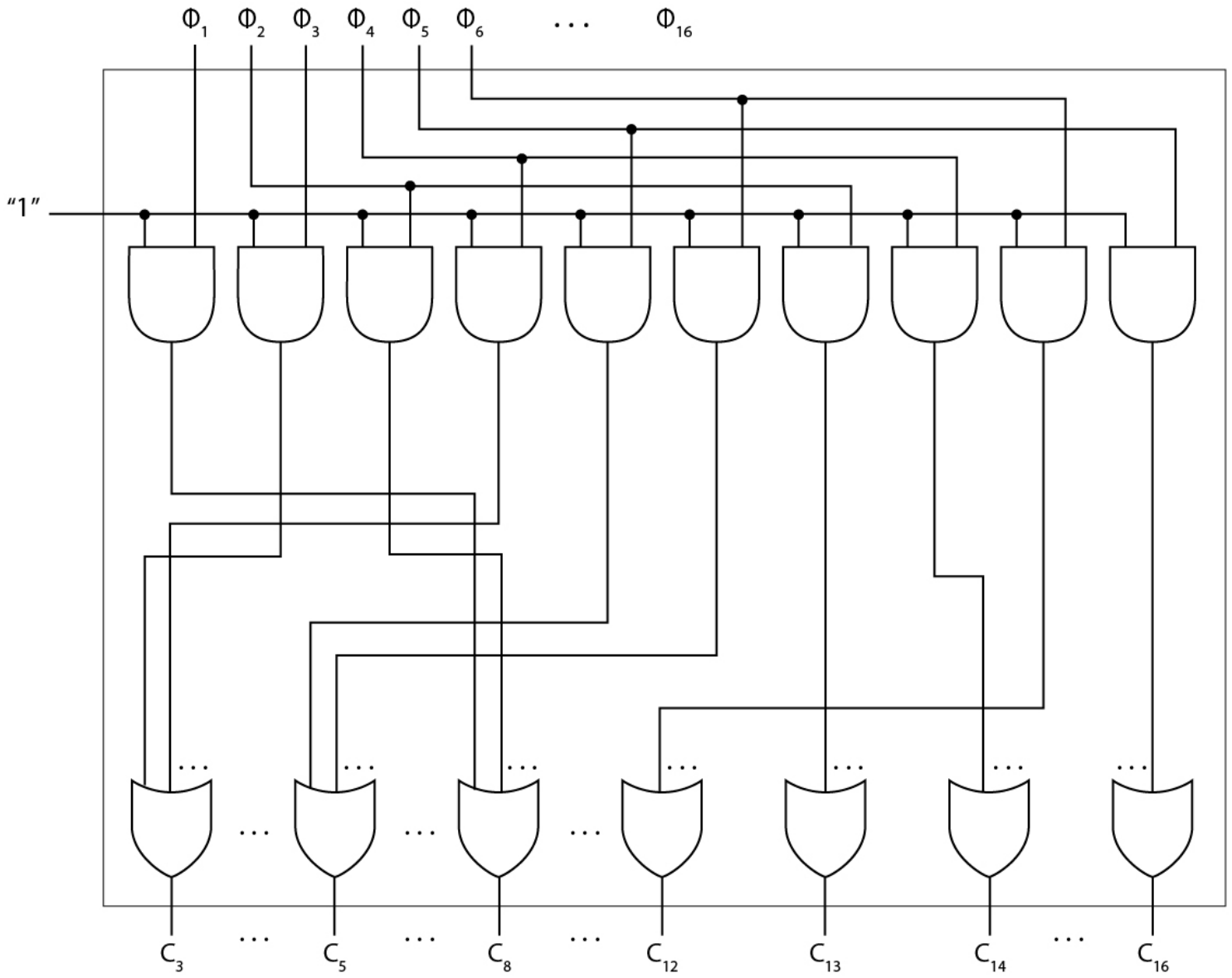
Instrukcijski ciklus (sažetak)

- 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 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 ($i = 0, 1, 2, \ldots, 12$) može se opisati logičkom jednadžbom:

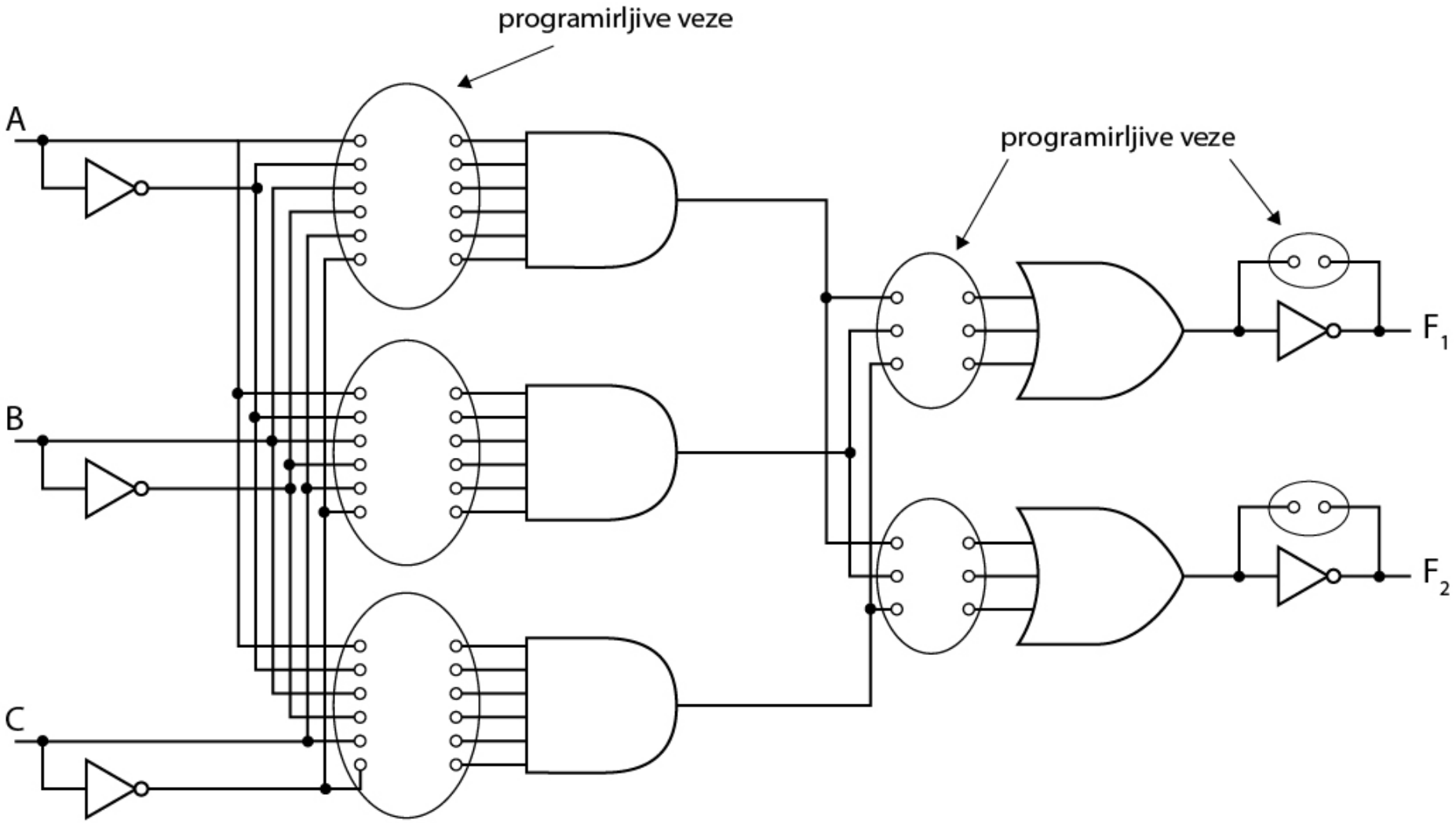
$$C_i = \sum_j \left(\Phi_j \cdot \sum_{m \in M_{ij}} I_m \right)$$

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

$M_{ij} \in \{1, 2, \ldots, 8\}$ - skup instrukcija za koje je C_i aktivan tijekom ϕ_j

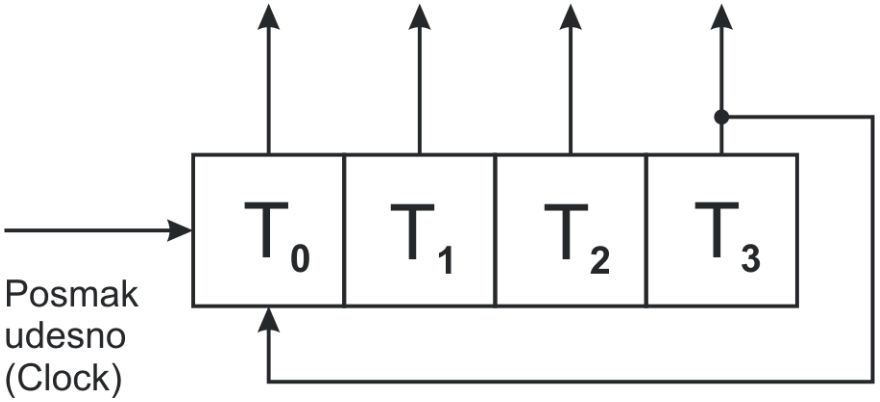
I_m – m-ti izlaz iz dekodera instrukcija

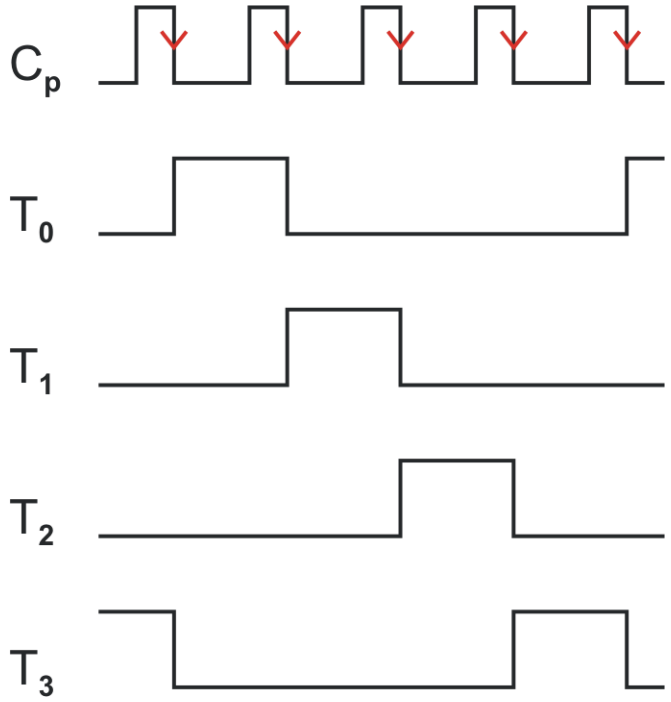
Programirljivi sklop PLA



Izvedba generatora slijedova (ideja 1):

- posmačni registar spojen u prstenasto brojilo (engl. Ring-counter)
- pri pokretanju (reset) paralelno upisujemo podatak **100...0**



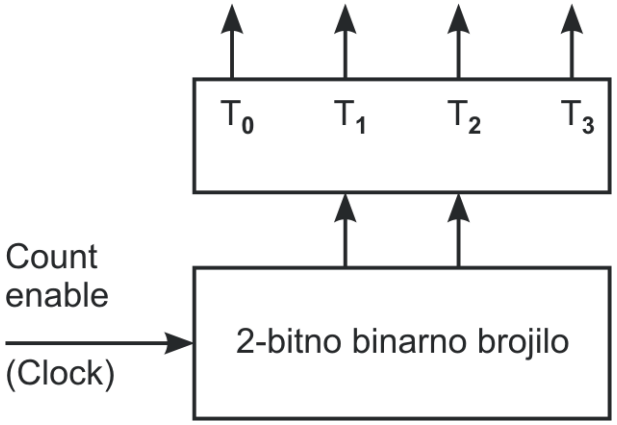


Izvedba generatora slijedova (ideja 2):

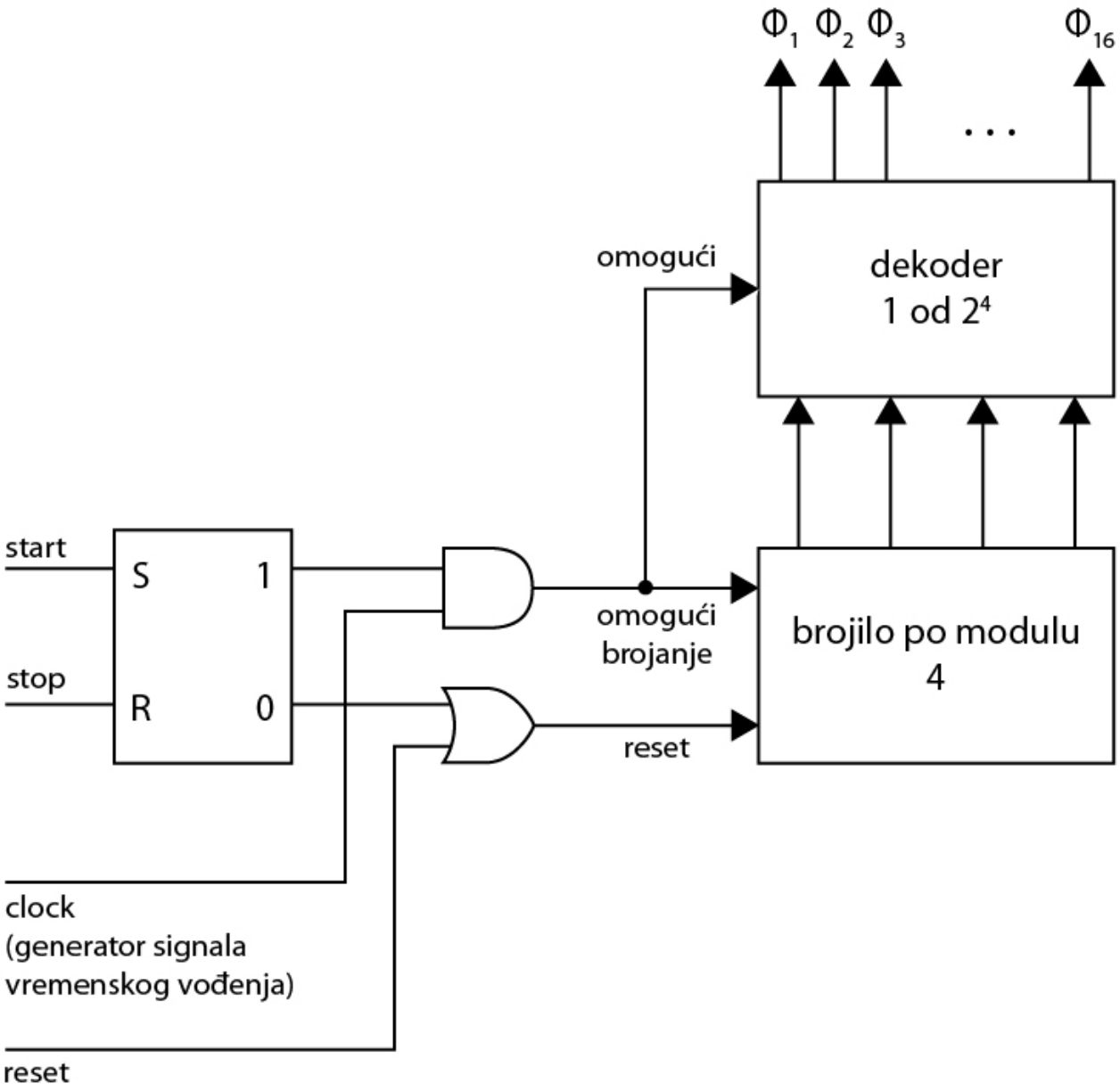
- binarno brojilo spojeno na dekoder
- pri pokretanju (reset) brojilo postavimo na početnu vrijednost

Dekoder:

Brojilo:



generator slijedova s mogućnošću zaustavljanja:



ZADATAK

U model 8 instrukcijskog procesora dodajte instrukciju sa sljedećom asemblerskom sintaksom:

```
lda konst(A)
```

Prikazana instrukcija učitava podatak iz memorije sa adrese A+konst i sprema ga u A. Nacrtajte kombinacijski dio upravljačke jedinice za tu instrukciju.

Riješite zadatak pod pretpostavkom da bilo koja od komponenata efektivne adrese (konstanta, akumulator) može biti negativna.