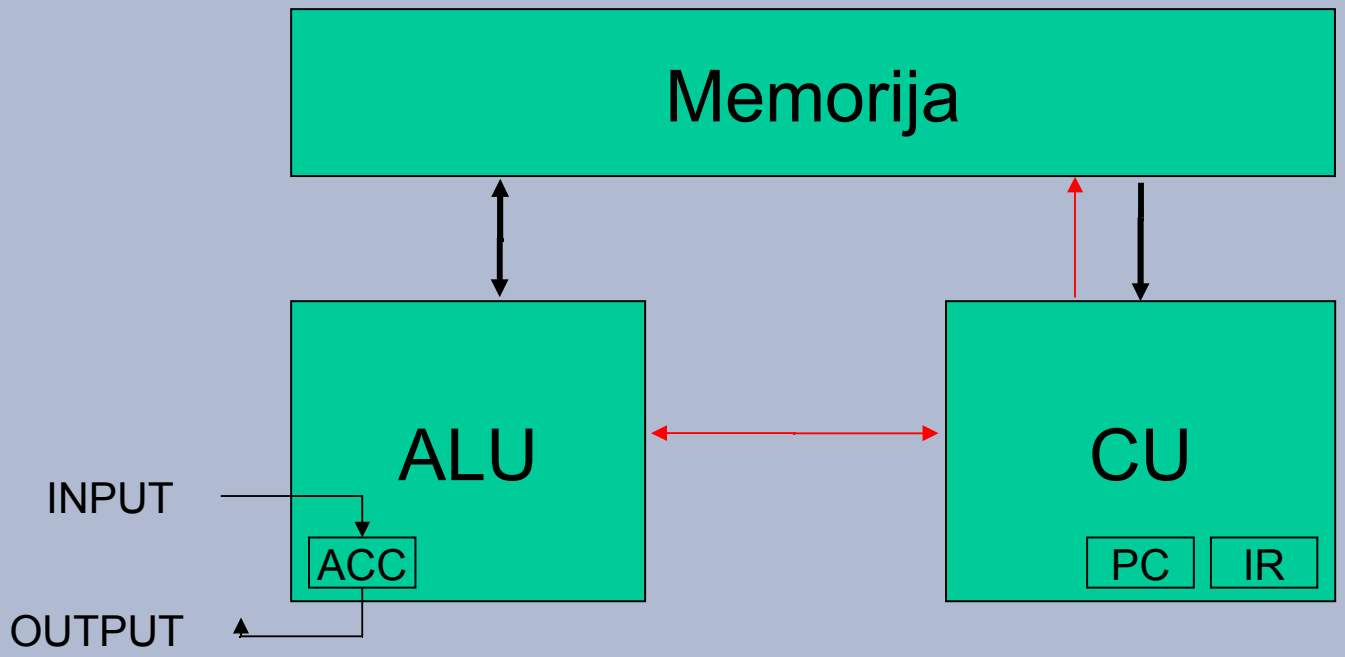
0b. Von Neumannov model računala

Razvoj **programirljivosti** računala:

- Univerzalni stroj [Turing36]
  - TS koji čita logičku funkciju s trake
- ENIAC (1943-1947): ručno prospajanje, prekidači (Mauchly, Eckert)
- EDVAC (1944-1949): **računalo s pohranjenim programom** (Mauchly, Eckert, von Neumann)
  - "First Draft of a Report on the EDVAC" [vonNeumann45]
- IAS (1952): program i podatci u zajedničkoj memoriji (von Neumann)
- IBM 704 (1954): Fortran (1954, 1957), LISP (1958)

Struktura von Neumannove arhitekture:

- **Memorija** pohranjuje program i podatke
- **Procesna jedinica**: izvodi aritmetičke i logičke operacije (→ **ALU**)
- **Upravljačka jedinica**: interpretiranje programa



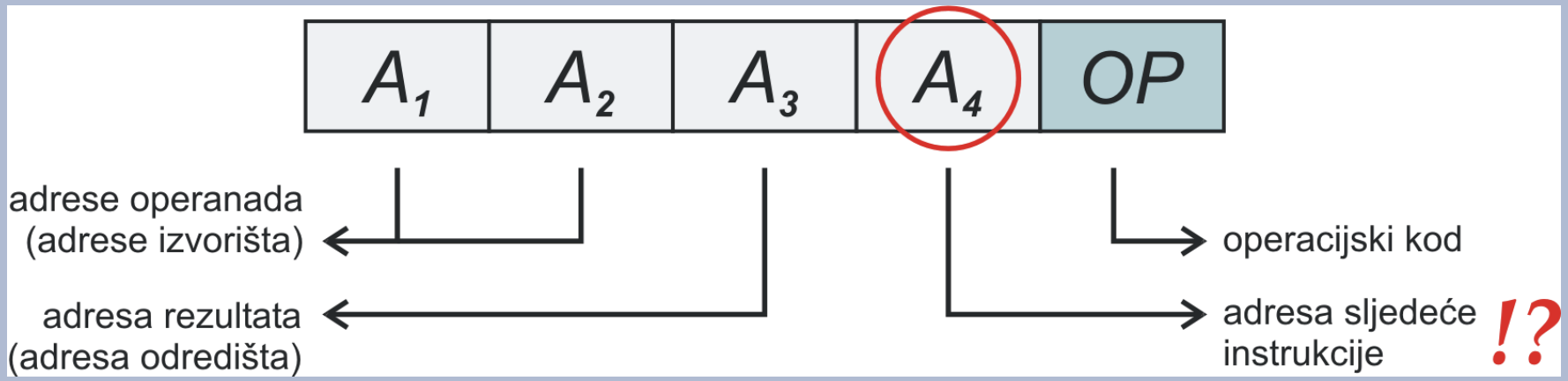
Pet funkcijskih jedinica von Neumannove arhitekture:

- Instrukcije svedene na *numerički kod* pohranjuju se kao i podatci, na jednak način i u istoj jedinici – memoriji
- Računalo – stroj za računanje mora imati jedinicu za izvršavanje osnovnih aritmetičkih operacija – aritmetičku jedinicu
- Jedinica koja "razumije" i tumači instrukcije te upravlja slijedom izvođenja operacija – upravljačka jedinica
- Računalo mora imati mogućnost komunikacije s vanjskim svijetom, to mu omogućava – ulazno-izlazna jedinica

[Ribarić]

Kako upravljati redosljedom instrukcija?

- EDVAC: svaka instrukcija eksplicitno određuje sljedeću instrukciju:

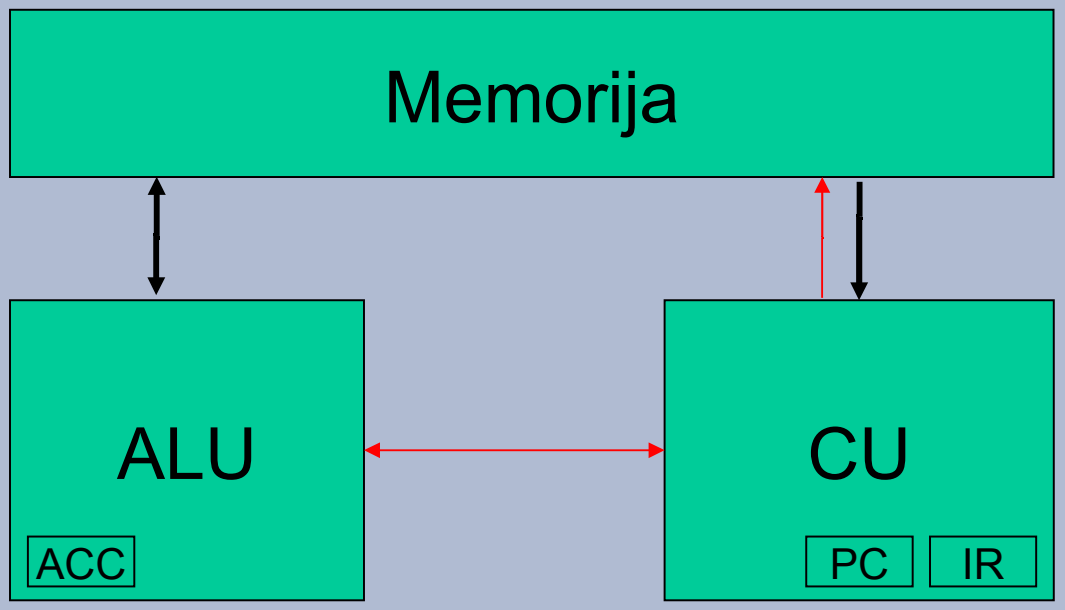


[Ribarić]

- IAS: adresa sljedeće instrukcije određena programskim brojilom PC
  - nakon većine instrukcija PC se implicitno uvećava za 1
  - značajna ušteda memorije u odnosu na EDVAC (programi su slijedni!)
  - prijelazi među slijednim odsječcima definirani specijalnim instrukcijama
    - uvjetno, bezuvjetno grananje
    - potprogrami su se pojavili tek sredinom 1960-tih
    - (IBM 360, PDP-8, HP 2100)

Tijek izvođenja programa:

- pribavljanje instrukcije:
  - CU adresira memoriju programskim brojilom
  - memorija šalje instrukciju
  - CU sprema instrukciju u IR, uvećava PC



- izvođenje instrukcije:
  - CU pribavlja memorijski operand i smješta ga u ALU; adresu operanda definira IR
  - ALU izvodi operaciju nad operandom i akumulatorom; operaciju definira IR
  - rezultat operacije smješta se natrag u akumulator

Značajke Von Neumannove arhitekture:

- numerički kodirane instrukcije spremljene zajedno s podatcima u **memoriji** sa slučajnim pristupom
- **upravljačka jedinica** prevodi instrukcije u slijed signala koji idu prema **ALU** i memoriji
- izmjenjuju se faze pribavi i izvrši, grananje i uvjetno grananje eksplicitnom promjenom programskog brojila
- paralelne operacije nad binarno kodiranim strojnim riječima, korištenje drugog komplementa
- dominantna arhitektura sve do početka 80-ih godina prošlog stoljeća
- glavni nedostatak: memorijsko usko grlo

Aritmetičko - logička jedinica

Elementi ALU:

- sklopovi za obavljanje aritmetičkih operacija
- registri (spremnici) za privremeno pohranjivanje operanada (operand – podatak koji sudjeluje u operaciji);

Prikaz podataka u **binarnom** sustavu:

- lakša tehnološka izvedba
- veća ekonomičnost predstavljanja brojeva
- laka implementacija logičkih operacija

[Ribarić]

Aritmetika računala IAS: zbrajalo i sklop za posmak

- Cjelobrojni operandi duljine 40 bita (12 decimala)
- Oduzimanje – pribrajanje dvojnog komplementa
- Množenje i dijeljenje - pod programskim upravljanjem ponavljanjem uzastopnih operacija zbrajanja, odnosno oduzimanja i posmaka

[Ribarić]

Upravljačka jedinica

- Generira upravljačke signale za ostale funkcijske jedinice

- Svaki korak algoritma predstavljen je jednom (strojnom) instrukcijom ili slijedom (strojnih) instrukcija.

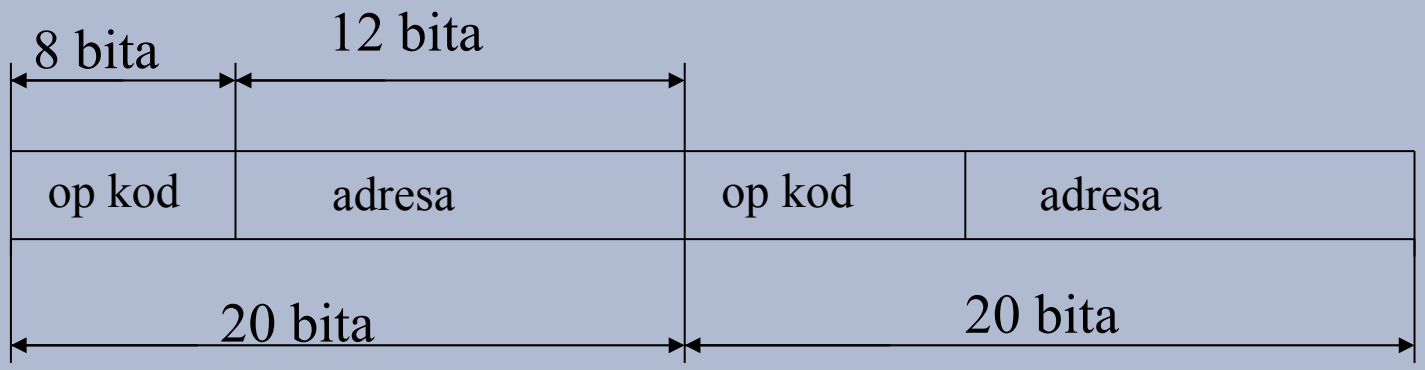
- Strojne instrukcije određuju elementarne operacije koje sklopovlje procesora može izvesti

[Ribarić]

Duljina riječi 40 bita (podaci predočeni 40-bitnim kodom).

Strojne instrukcije duljine 20 bita.

Dvije instrukcije smještene u jednoj riječi u memoriji (lijeva i desna instrukcija):



Lijeva instrukcija Desna instrukcija

Akumulatorski orijentirana arhitektura

- binarne operacije izvode se prema modelu A=f(A,M)
- inicijalan podatak u A je “izgubljen” nakon operacije

Instrukcije su jednoadresne:

- operacijski kod (8 bit): binarno kodirana instrukcija
  - npr: 11001100 – kod instrukcije add –(zbroji)
- adresa (12 bit): jednoznačna memorijska lokacija
  - npr: 1001 0011 0111 - $937
- 12 bitova omogućava izravno adresiranje 4096 memorijskih lokacija

operacijski kod	adresa memorijske lokacije

Što radi instrukcija $cc100?

[Ribarić]

Programsko brojilo **PC** (*Program Counter):*

- registar koji sadrži adresu sljedeće instrukcije
- IASov 13-bitovni PC: 12 adresnih bitova +1 bit za izbor lijeve ili desne instrukcije

Instrukcijski registar **IR**

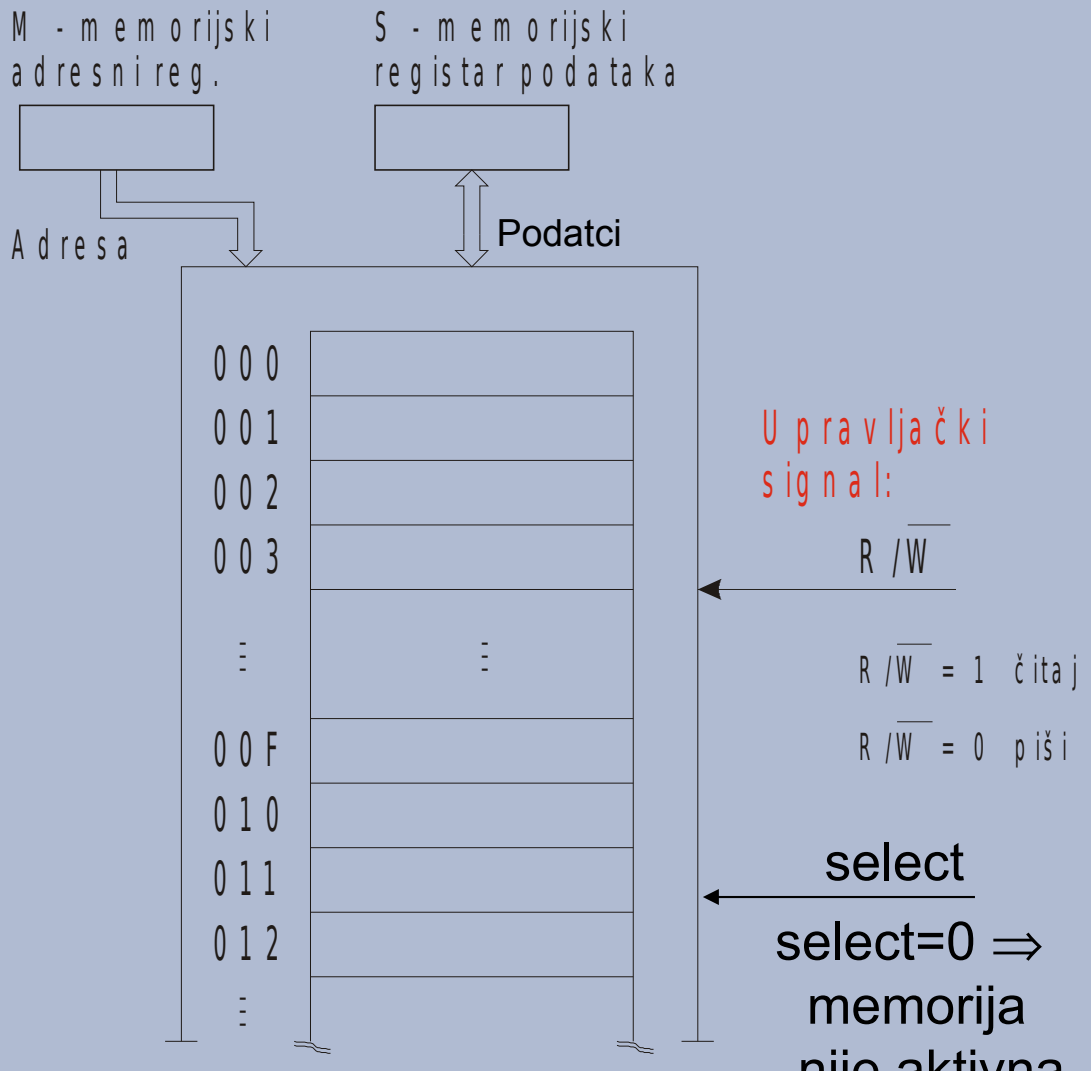
- registar koji sadrži instrukciju (operacijski kod)

Skup strojnih instrukcija (**ISA**)

- Aritmetičke i logičke instrukcije
- Instrukcije za prijenos podataka
- Instrukcije uvjetnog i bezuvjetnog grananja
- Ulazno-izlazne instrukcije

Memorija u von Neumannovom modelu

- Svakoj memorijskoj lokaciji jednoznačno je pridružena adresa
- Memorija nema procesnih sposobnosti
- Memorijska jedinica – prateći modul
- Procesor – vodeći modul
- Izvedba: katodne cijevi (!!!)
  - elementi: bijela točka na zaslonu
  - osvježavanje svakih cca 100ms
  - Vrijeme pristupa cca 50 us
  - 500 – 1000 bitova po cijevi
  - ukupno oko 2000 cijevi

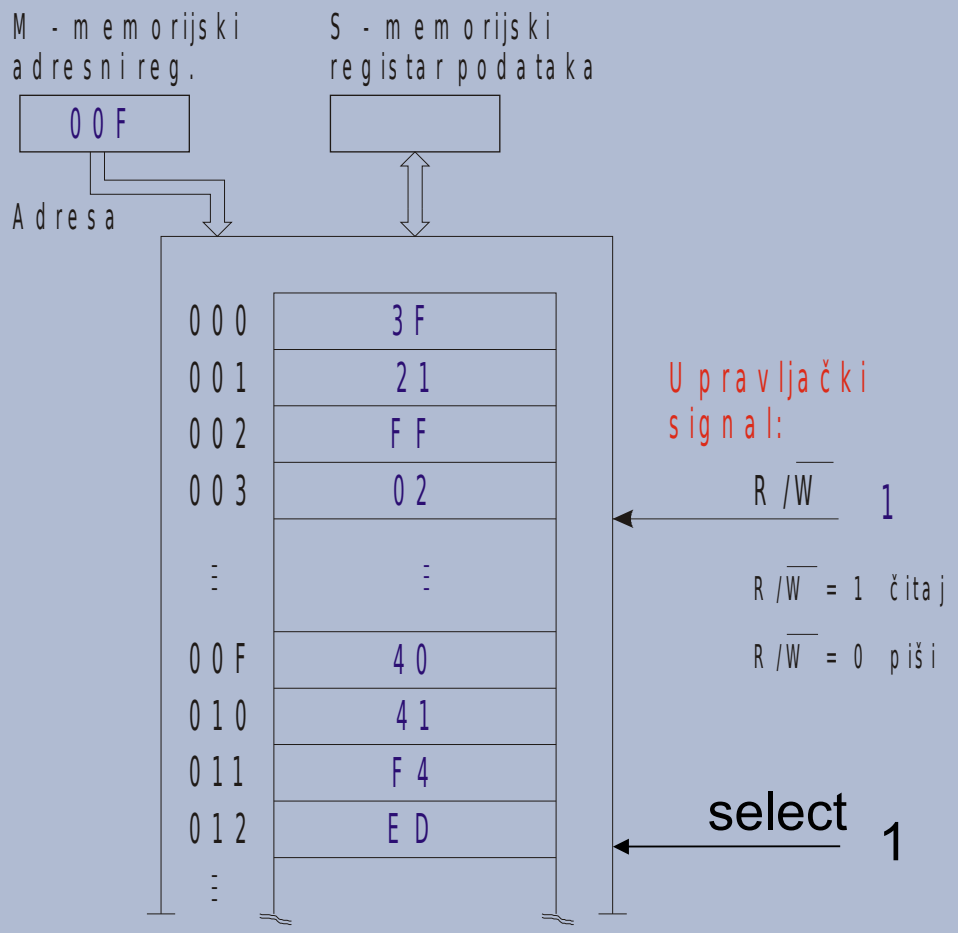


[Ribarić]

Operacija čitanja (select=1, R/W* = 1)

Adresa → M

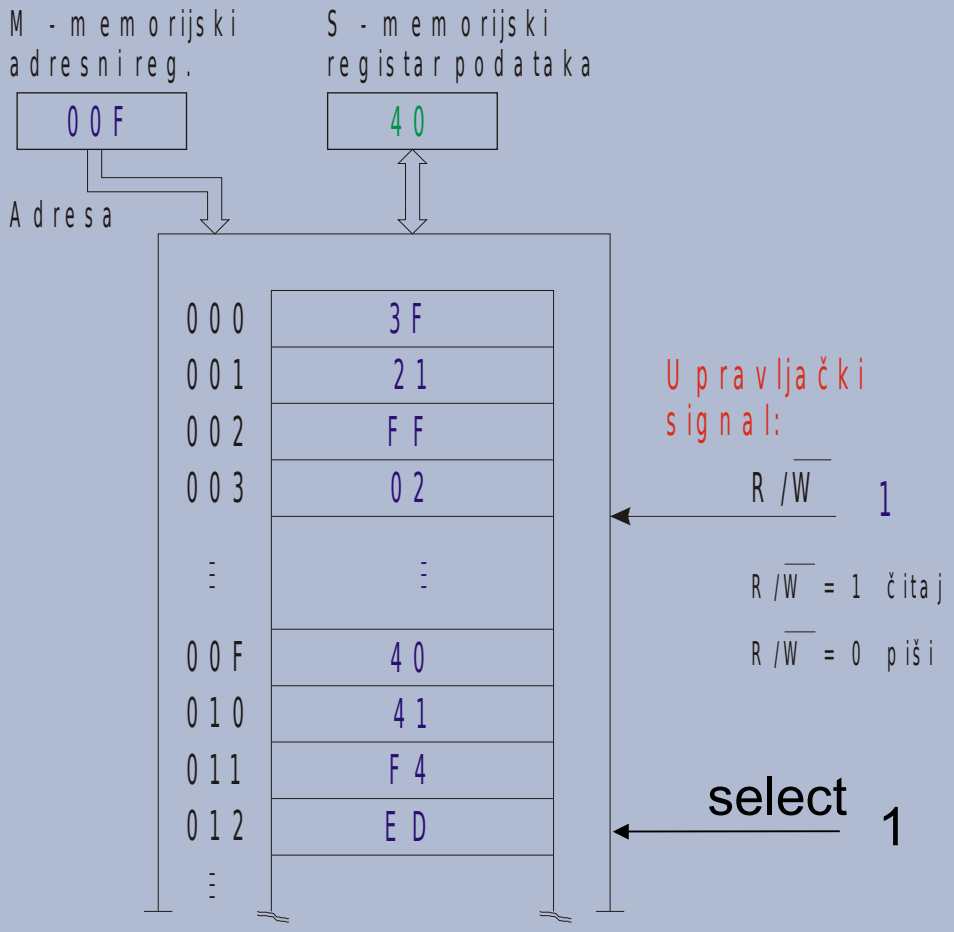
$R/\overline{W}$ → 1



[Ribarić]

Stanje nakon isteka vremena pristupa memoriji

- podatak s lokacije 00F “preslikan” u memorijski registar podataka **S**
- operacija čitanja je nedestruktivna operacija!

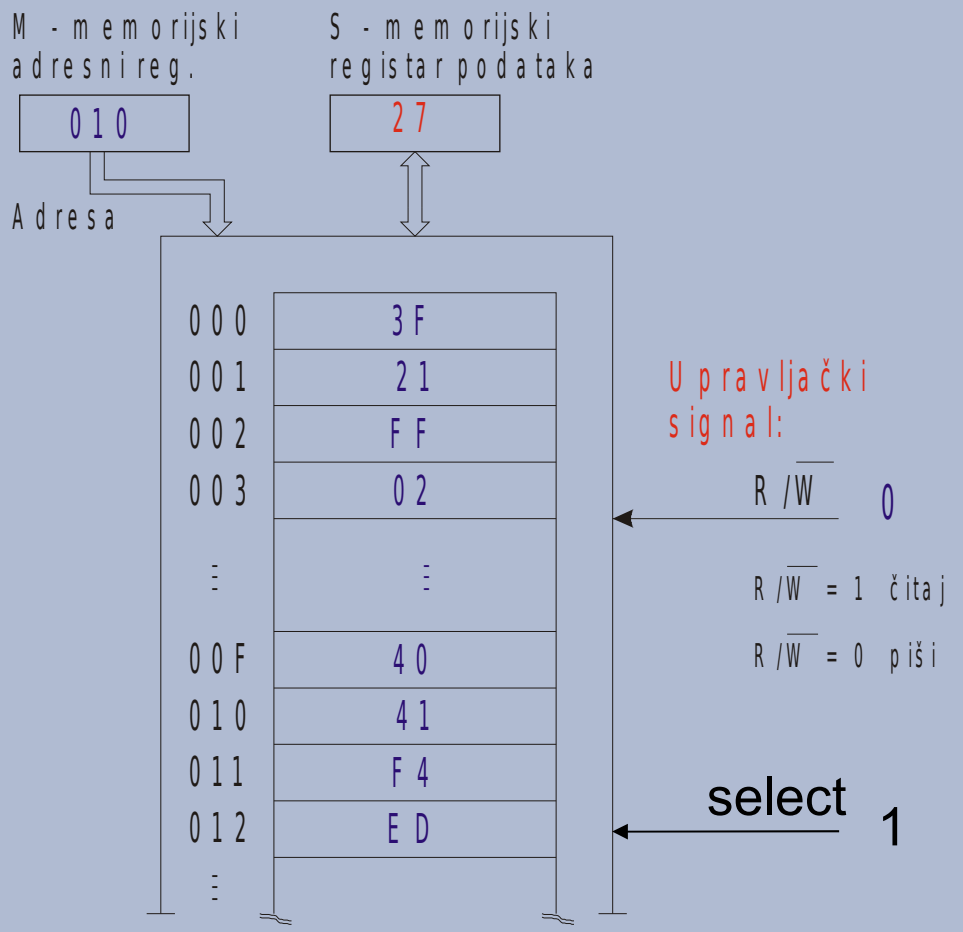


Operacija Pisanja (select=1, R/W* = 0)

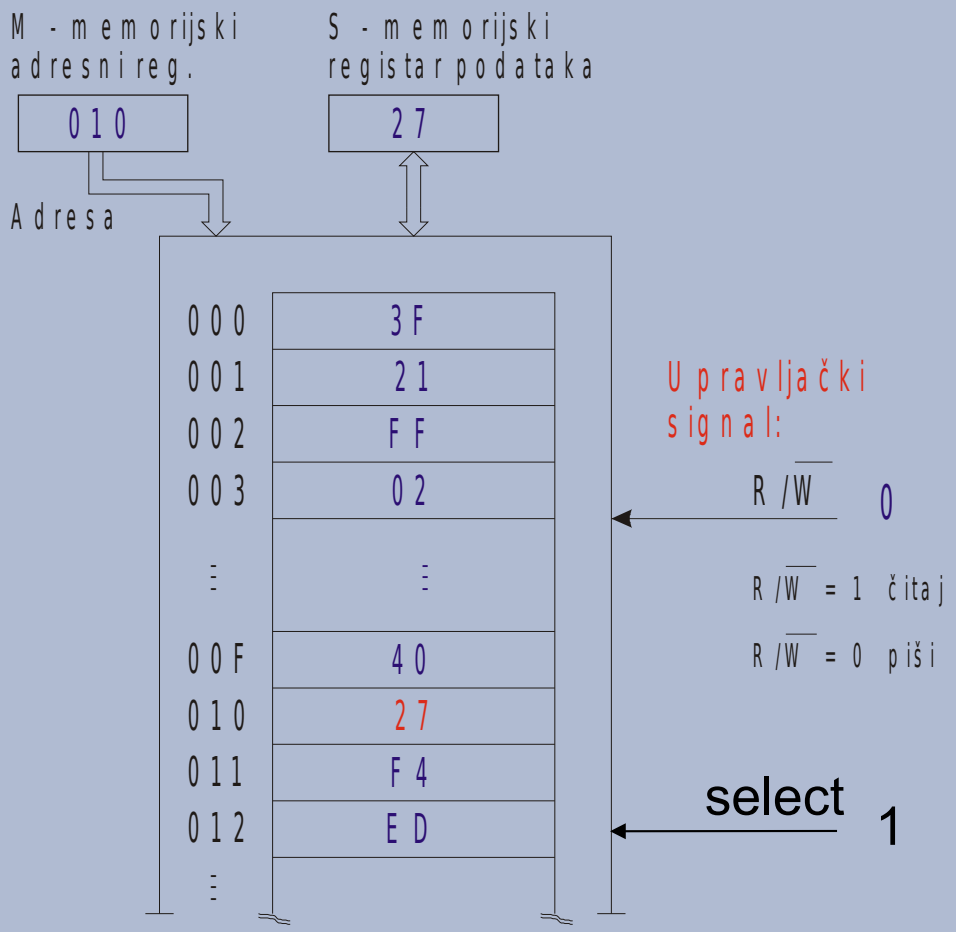
Adresa → M

Podatak → S

$R/\overline{W}$ → 0



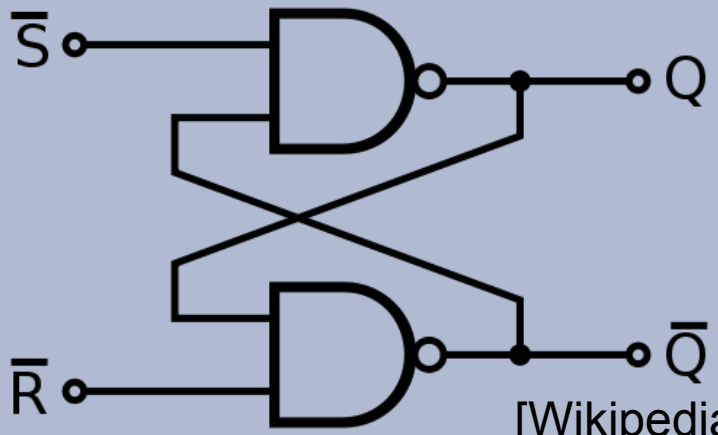
Stanje nakon isteka vremena pristupa memoriji



[Ribarić]

Građevni elementi konceptualne sheme memorije

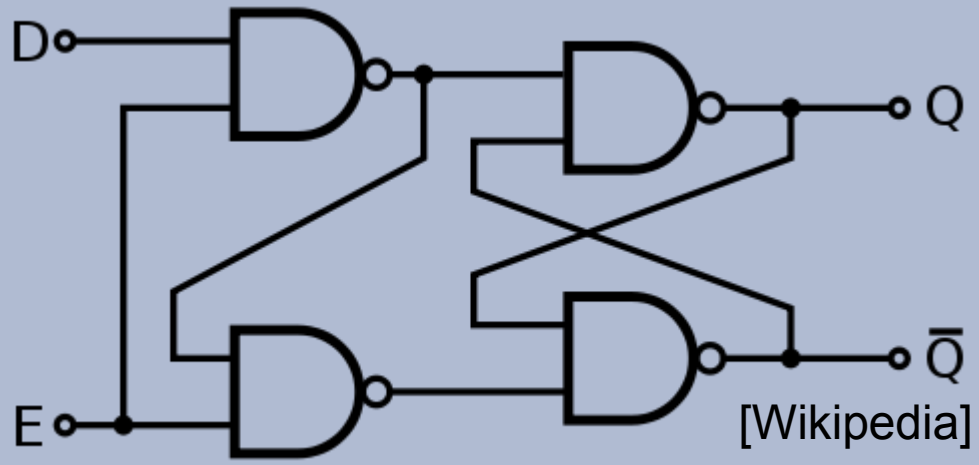
osnovni S*R*-Bistabil:



[Wikipedia]

		ulaz: S*,R*			
		00	01	10	11
stanje:	0	X	1	0	0
Q	1	X	1	0	1

D-Bistabil s ulazom za omogućavanje:

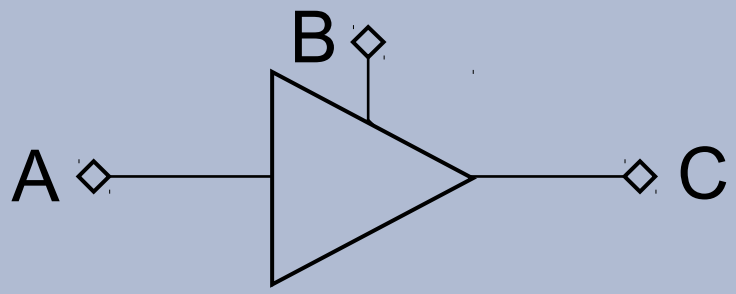


[Wikipedia]

		ulaz: E,D			
		00	01	10	11
stanje:	0	0	0	0	1
Q	1	1	1	0	1

Građevni elementi konceptualne sheme memorije (2):

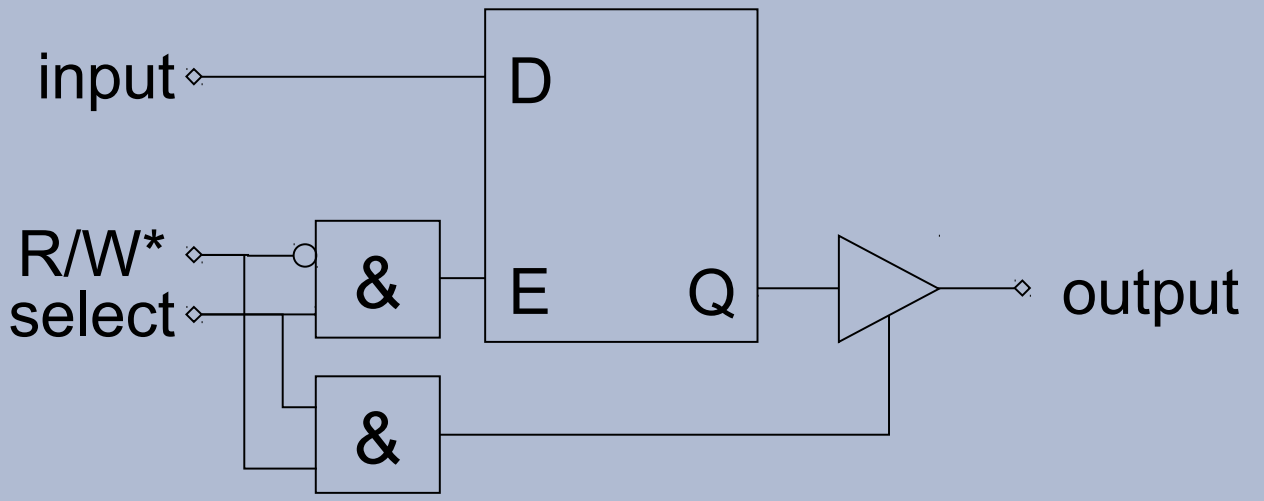
Pogonski sklop s tri stanja:

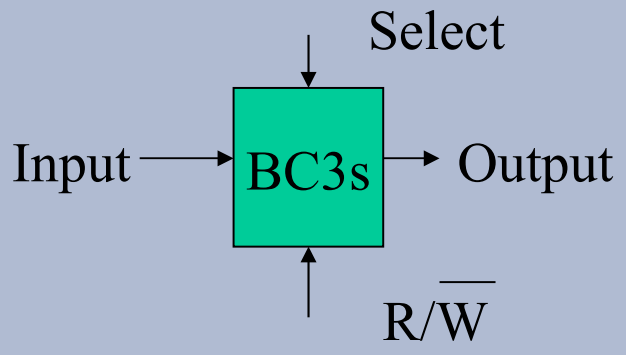


A	B	C
0	0	Z
0	1	0
1	0	Z
1	1	1

Z: stanje visoke impedancije (sklop je električki odspojen!)

Binarni memorijski element s izlazom s tri stanja:

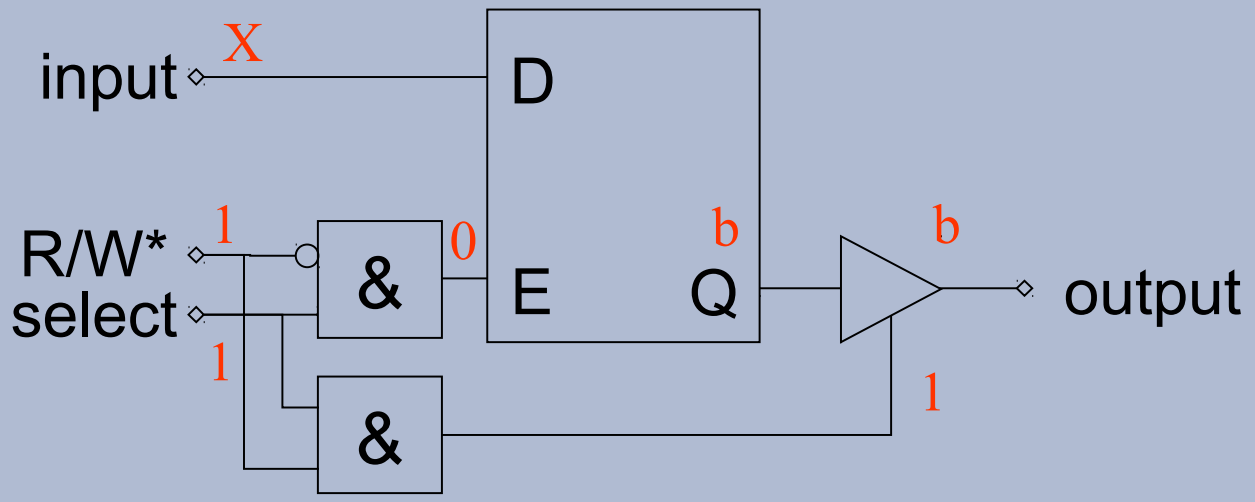




Binarni memorijski element s izlazom s tri stanja u akciji:

Operacija čitanja:

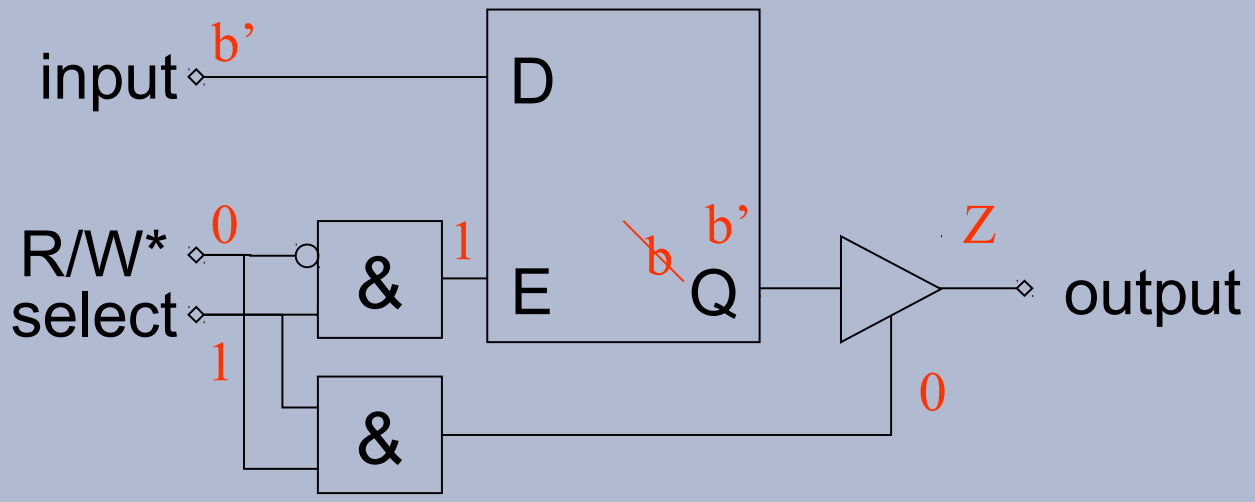
select=1, R/W*=1 ⇒ output=Q (0 ili 1)



Binarni memorijski element s izlazom s tri stanja u akciji (2):

Operacija pisanja:

select=1, R/W*=0 $\Rightarrow$ Q= input

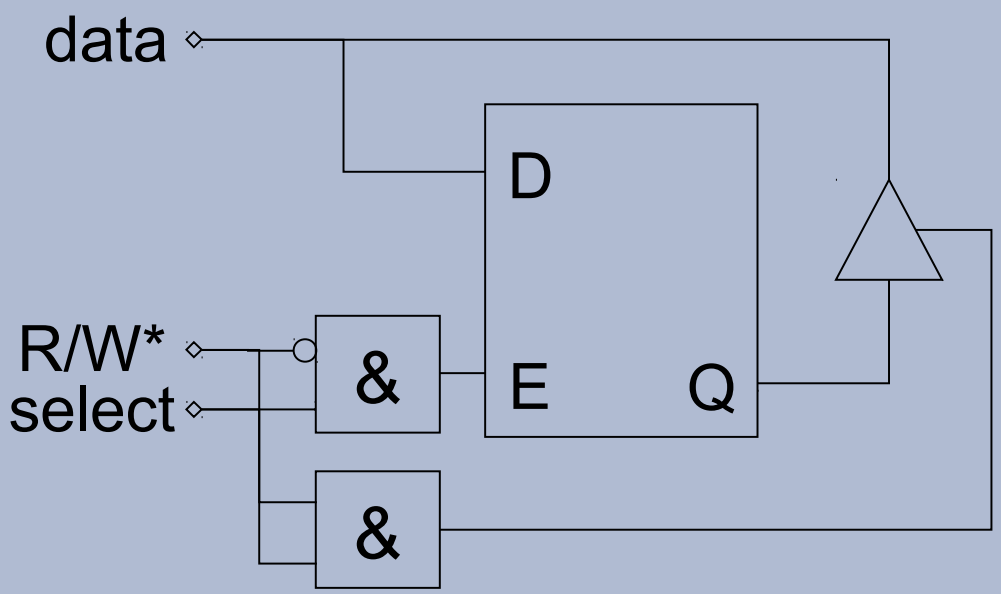


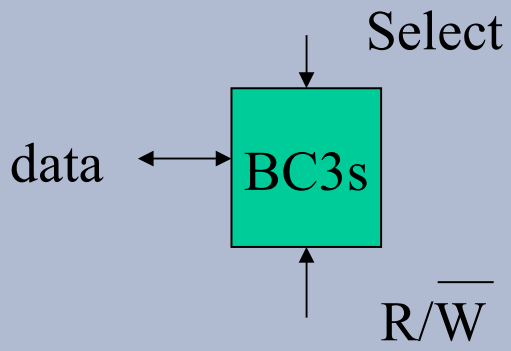
Binarni memorijski element: pojednostavljenje

Element čita podatkovni ulaz (`input`) akko: `select=1`, `R/W*=0`

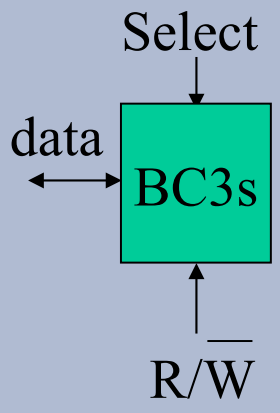
Element postavlja podatkovni izlaz (`output`) akko: `select=1`, `R/W*=1`

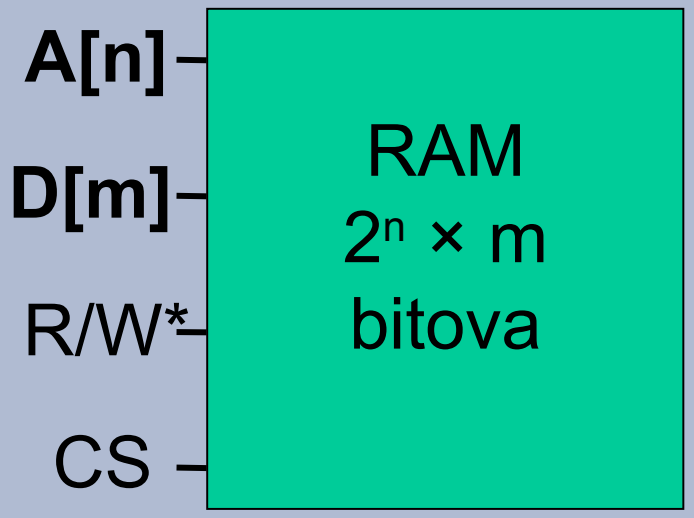
⇒ linije `input` i `output` možemo implementirati jednom linijom `data`!

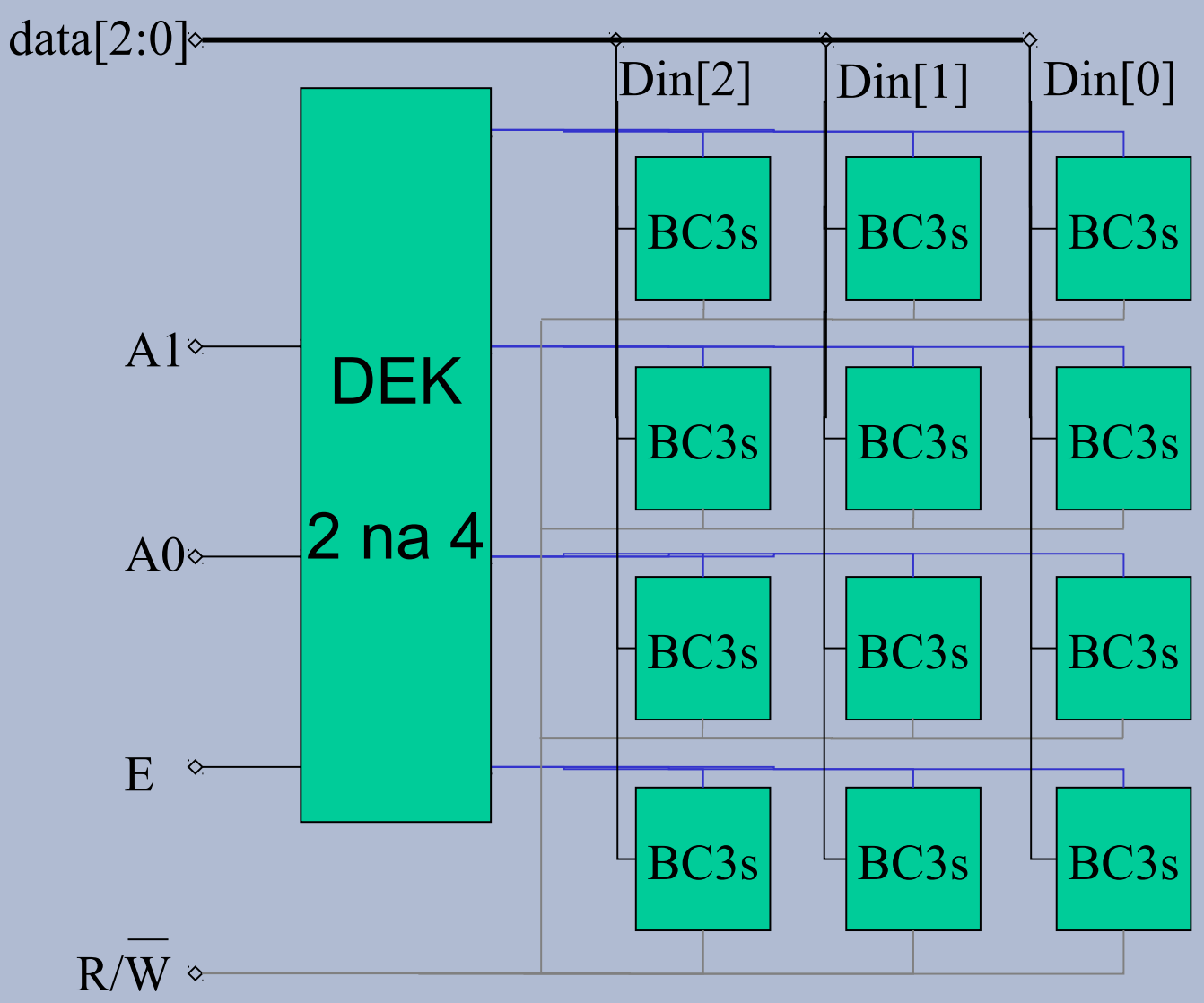




Konceptualna shema memorije 4 x 3 bita korištenjem binarnog memorijskog elementa s tri stanja:



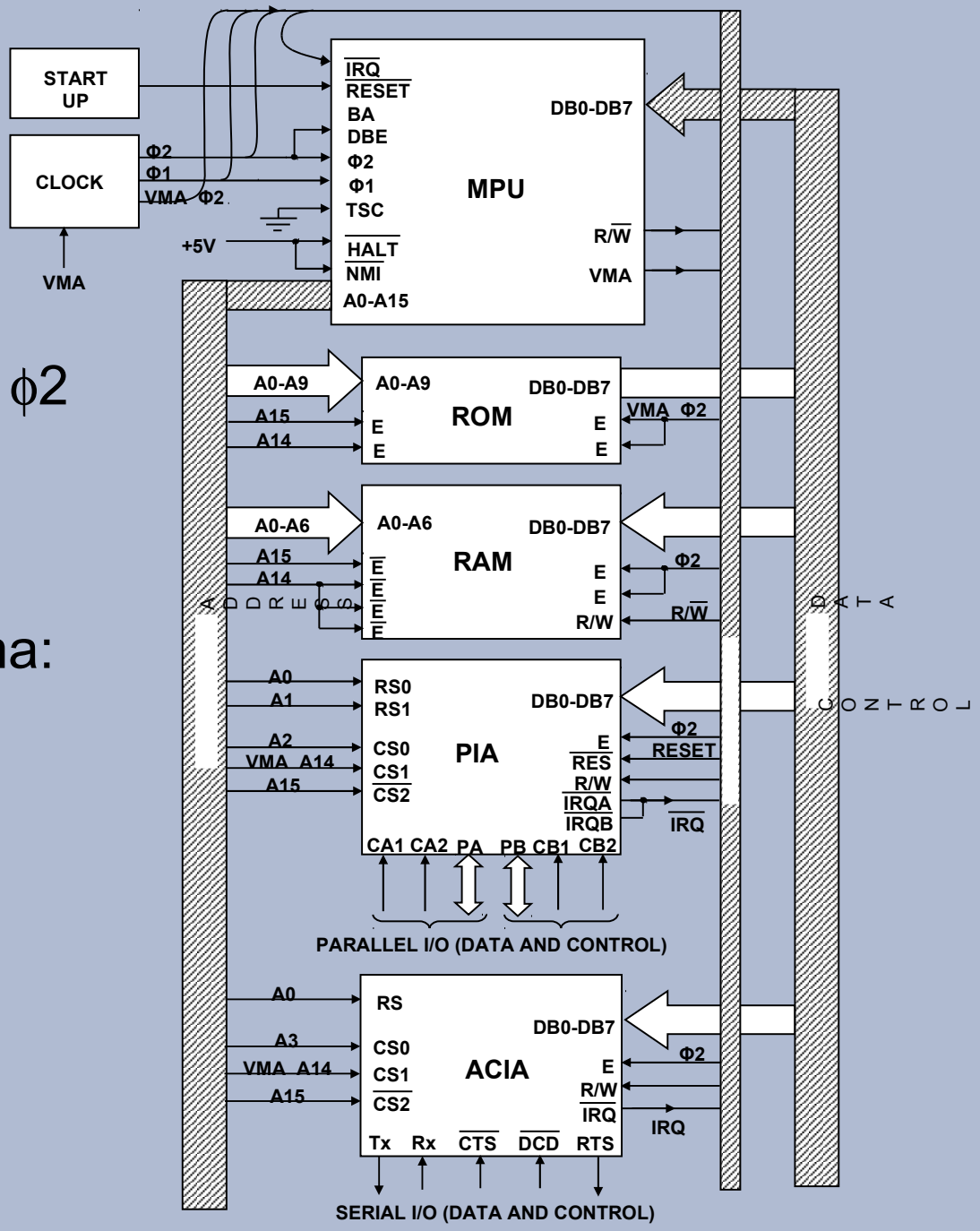




Zadatak za vježbu:

Nacrtajte izvedbu memorije 8 x 4 bita uporabom memorijskih elemenata BC3s.

Primjer: računalo temeljeno na MC6800 (1974)



- CPU (MPU) radi u taktovima $\phi 1$ i $\phi 2$
- Identificirati funkcijske jedinice
- Veze među funkcijskim jedinicama:
  - sabirnica podataka
  - upravljačka sabirnica
  - adresna sabirnica
- Zajednički adresni prostor
  - memorija
  - ulaz – izlaz (periferija)

[Ribarić]

Vježba:

- Ilustrirati načelo jednoznačnosti adresa određivanjem dijela adresnog prostora koji je dodijeljen memorijskim (RAM, ROM) i perifernim (PIA, ACIA) sklopovima
- Komentirati “rastrošno” raspolaganje memorijskim prostorom
- Objasniti razliku između potpunog i nepotpunog adresnog (de-)kodiranja

[Ribarić]

Zadatci:

1.Statički RAM sa sljedećim važnijim priključcima A0-A11, D0-D7, CS0, CS1,CS2*, CS3*, R/W* priključite na 16-bitnu adresnu i 8-bitnu podatkovnu sabirnicu tako da se sklop javlja na početnoj adresi $A000. Odrediti adresni prostor koji zauzima sklop te nacrtati shemu priključenja.
Kako bismo riješili zadatak ako bi memoriju trebalo spojiti na adresu $A800?

2. Programirljivi ulazno-izlazni međusklop ima četiri osmobitna naslovljiva registra (R0-R3). Uz pretpostavku da međusklop ima priključke RS0*, RS1*, E, E*, D0-D7, R/W* i ϕ, nacrtajte shemu priključenja na 16-bitnu adresnu sabirnicu te 8-bitnu sabirnicu podataka, tako da se međusklop javlja na početnoj adresi $8008. Koristiti potpuno dekodiranje adrese, te adresni dekoder izvesti diskretnim logičkim sklopovima ekvivalencije (EKS-NILI) i konjunkcije (I).

[Ribarić]

Prikaz veličine adresnog prostora za Intelovu porodicu procesora:

- Intel 8080 — Adresna sabirnica: A0 – A15
  64 K memorijskih lokacija
  256 I/O lokacija
- Intel 8086 — Adresna sabirnica: A0 – A19
  1M memorijskih lokacija
  64 K I/O lokacija
- Intel 286, 386 SX — A0 – A23
  16 M memorijskih lokacija
  64 K I/O lokacija

Prikaz veličine adresnog prostora za Intelovu porodicu procesora (nastavak):

- Intel 386 DX, 486, Pentium I:
  A0 – A31
  4G memorijskih lokacija
- Pentium II, III, ... A0 – A35 (PAE)
  64G memorijskih lokacija
- AMD Opteron, Intel Nehalem, Intel Itanium A0 – A63 (2^{64} bajtova)

Što se zbilo nakon ENIAC-a i IAS-a?

1. generacija računala (elektronske cijevi): 1945-1955
   ENIAC (1946), UNIVAC(1951), IBM 704 (1954)

2. generacija (diskretni tranzistori): 1950 - 1960
   CDC 1604 (1960)

3. generacija (miniračunala, LSI, TTL 7400): 1960-1970
   IBM 360 (1964), DEC PDP 8 (1965), VAX 11/780 (1977)

4. generacija (mikroračunala): 1970 –
   - integrirana upravljačka i aritmetička jedinica
   - u početku: jeftina manje moćna varijanta računala
   - vrlo brzo jedini izbor uslijed:
     - jeftinog proizvodnog ciklusa
     - velikih potencijala za povećanje performanse