Građa računala

Elementi CU

Von Neumann-ovo računalo

- Program koji se pokreće pohranjuje se u binarnom formatu u memorijskoj jedinici – kako bi rezultati izvršavanja mogli biti korišteni za neki “efekt” u programu – tzv. “Stored Program Concept”
- Ovaj concept kaže da su instrukcije pohranjene u memoriji skupa sa podacima u formatu koji je čitljiv CPU-u, a računalo može manipulirati sa tim podacima pošto instrukcije i podaci ovise i o drugim parametrima (npr. kontrolnim stanjima)
- Instrukcije se izvode serijski (sekvencijalno) prateći kontrolni tok programa kako je zapisan u memoriji

Arhitektura von Neumannovog računala

- Upravo kako bismo bili u mogućnosti ostvariti ovakvu metodu izvršavanja na von Neumannovom konceptu računala, trebamo uobičajene gradivne jedinice von Neumannovog računala:
  - ALU
  - Memory
  - Input I output
  - Kontrolna jedinica
- Isto tako, ALU mora imati registar koji se obično zove akumulator
- Kontrolna jedinica mora imati registar/brojač koji se zove PC (*Program Counter*, programsko brojilo) – prati koja je slijedeća instrukcija za izvršavanje u toku programa
- Ovi registri su obično ostvareni kao memorijske komponente u CPU-u koje pomažu ispravnom izvršavanju toka programa

Kako radi von Neumannovo računalo?

- Izvršava ili *emulira* Fetch-Decode-Execude sekvencu za izvršavanje programa
  - *Fetch* – dobavlja instrukciju iz memorije sa adrese koja je zapisana u PC, inkrementira se PC tako da se namjesti na adresu iduće instrukcije
  - *Decode* – dekodiranje instrukcije korištenjem kontrolne jedinice
  - *Execute* – izvršava se pgraom korištenjem podatkovnog toka, kontrolne jedinice i ALU
    - Dobavljaju se podaci iz memorije za izvršavanje instrukcije
    - Rezultati koje treba osvježiti u memoriji se osvježavaju na kraju izvršavanja instrukcije
- Nakon ovog ciklusa, kreće se od početka

Uloga ALU

- Uobičajene operacije koje obavlja ALU:
  - Aritmetičke operacije (npr. zbrajanje i oduzimanje)
  - Logičke operacije (npr. AND i OR)
  - Premještanje podataka (npr. Load i Store)
- ALU svoje ime duguje osnovnim operacijama koje izvršava
- ALU ostvarujemo korištenjem kombinacije različitih vrsta krugova/sklopova

Kako izgleda pokretanje računala?

- Resetiranje procesora I internih kompomenti na predefinirane vrijednosti
- Za vrijeme resetiranja procesora, učitava PC registar sa memorijskom lokacijom prve instrukcije – ovisi o arhitekturi procesora i BIOS programiranju
- PC je fundamentalno bitna jedinica kontrolne jedinice, bez koje nema modernog računala – u PCu se nalazi adresa slijedeće instrukcije koju treba izvršiti
- Na početku ciklusa izvršavanja, kontrolna jedinica čita podatke sa adrese koju daje PC I učitava ih u interni registar za dekodiranje I pokretanje – prva riječ instrukcije sadržava **opcode**
- Opcode je bitan jer na temelju te riječi kontrolna jedinica zna da li treba učitati još neke podatke sa dodatnih memorijskih lokacija, npr. adresu u memoriji ili neki operand

Uloga kontrolne jedinice

- Sinhroni, sekvencijalni digitalni sklop
- Koordinira sekvencijalno pristupanje podacima I instrukcijama u I iz CPUa I pripadajućih CPU registara
- Interpretira instrukcije procesora I upravlja njihovim izvršavanjem
- U sklopu procesa izvršavanja, surađuje sa drugim funkcionalnim jedinicama procesora i vanjskim jedinicama (memorija, ulazno-izlazni uređaji)
- Sudjeluje u dekodiranju instrukcija – na temelju dekodiranja instrukcije kontrolna jedinica identificira kategoriju instrukcije (npr. grananje/branching)

Što sve radi CU?

- Generira upravljačke signale
- Koordinira sve aktivnosti unutar mikroprocesora
- Sinkronizira prijenos podataka i komunikaciju modula
- Pribavlja, dekodira i omogućuje izvođenje instrukcija
- Komunicira s ostalim komponentama mikroračunala preko sabirnica (adresne, podatkovne, upr.)
- Upravlja odgovorima na vanjske signale (zahtjev za prekid, zaustavljanje itd)

Temeljne funkcije CU

1. Uspostavljanje određenog stanja tijekom svakog instrukcijskog ciklusa
2. Određivanje sljedećeg stanja na temelju trenutnog stanja, stanja zastavica u statusnom registru i stanja na ulaznim upravljačkim linijama procesora
3. Pohranjivanje informacije koja opisuje tekuće stanje u kojem se procesor nalazi
4. Generiranje upravljačkih signala za izmjenu podataka između procesora i drugih funkcijskih jedinica

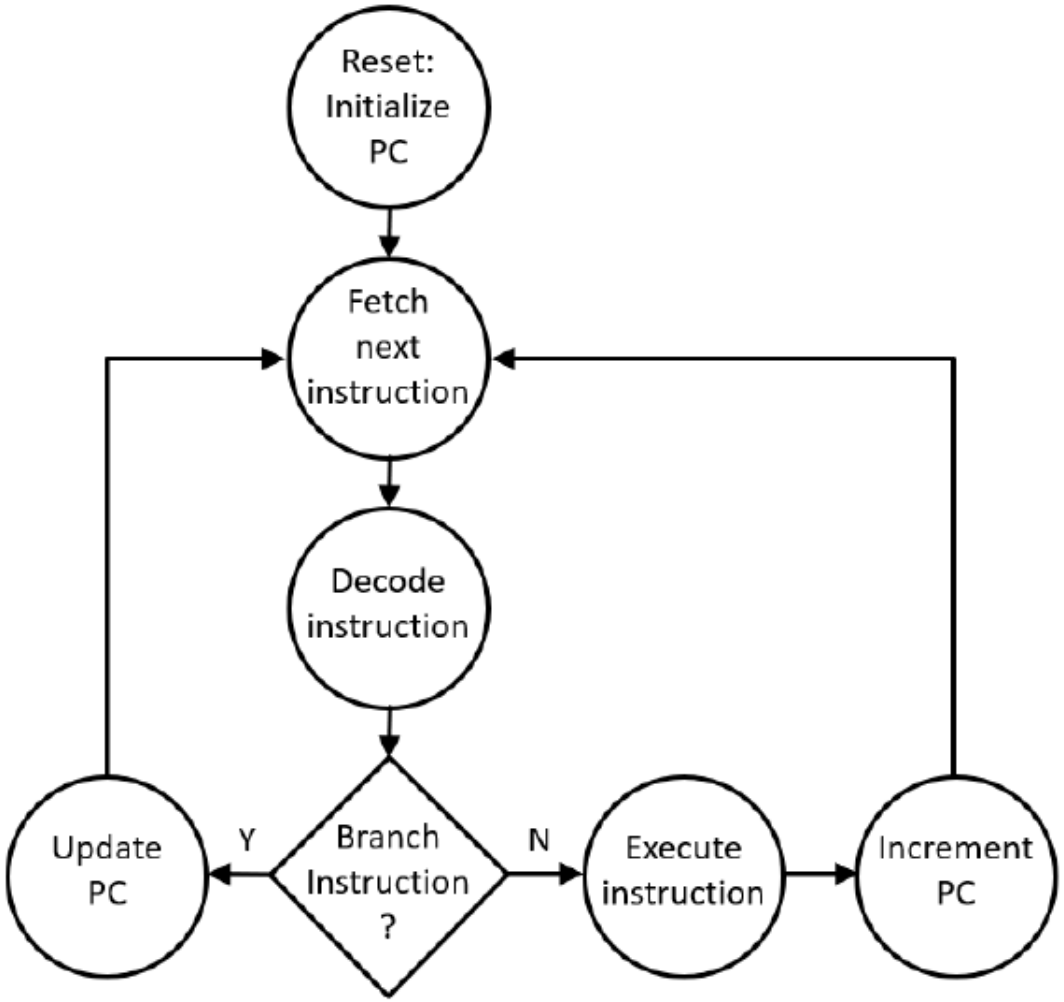
Prijenos upravljanja između različitih programa

- 1. Programske procedure su temelj strukturiranih programa
  - Pozivi procedura predstavljaju zapravo prijenos upravljanja s jednog programa na drugi
  - Kad se obavi zadatak definiran pozvanom procedurom (potprogramom), upravljanje se mora vratiti na instrukciju u pozivajućem programu, koja neposredno slijedi instrukciji koju je proceduru pozvala
- 2. Drugi važan slučaj prijenosa upravljanja između dva programa događa se tijekom prekida (engl. Interrupt)
  - U tom slučaju govorimo o:
    - prekinutom programu – programu čije je izvođenje prekinuto zahtjevom za prekid, i
    - prekidnom programu – programu na koje se prenosi upravljanje i koji obično poslužuje ulaznoizlaznu jedinicu koja je generirala zahtjev za prekid

Prijenos upravljanja između različitih programa

- 3. Prijenos upravljanja između programa može nastati uslijed iznimke (engl. exception)
  - Pod iznimkom podrazumijevamo posebne okolnosti kojima se narušava normalno stanje procesora i izvođenje tekućeg programa
  - Prekidi se mogu promatrati kao podskup iznimke
  - Iznimke mogu biti dvojake:
    - vanjske iznimke su posebne okolnosti izazvane događajem izvan procesora
    - unutarnje iznimke su posebne okolnosti kao posljedica događaja unutar procesora

Kako izgleda proces izvršavanja instrukcije?



Kako radi kontrolna jedinica?

- Prima ulazne informacije koje pretvara u kontrolne signale
- Kontrolni signali su potrebni da bi se instrukcije mogle izvršavati kako treba, za obradu prekida, internih pogrešaka I sl.
- Kontrolni signali se šalju procesoru
- Kontrolni signali se koriste I za komunikaciju prema sabirnicama, memoriji I ulazno-izlaznim sustavima
- Procesor govori dostupnim uređajima koje operacije treba izvršiti

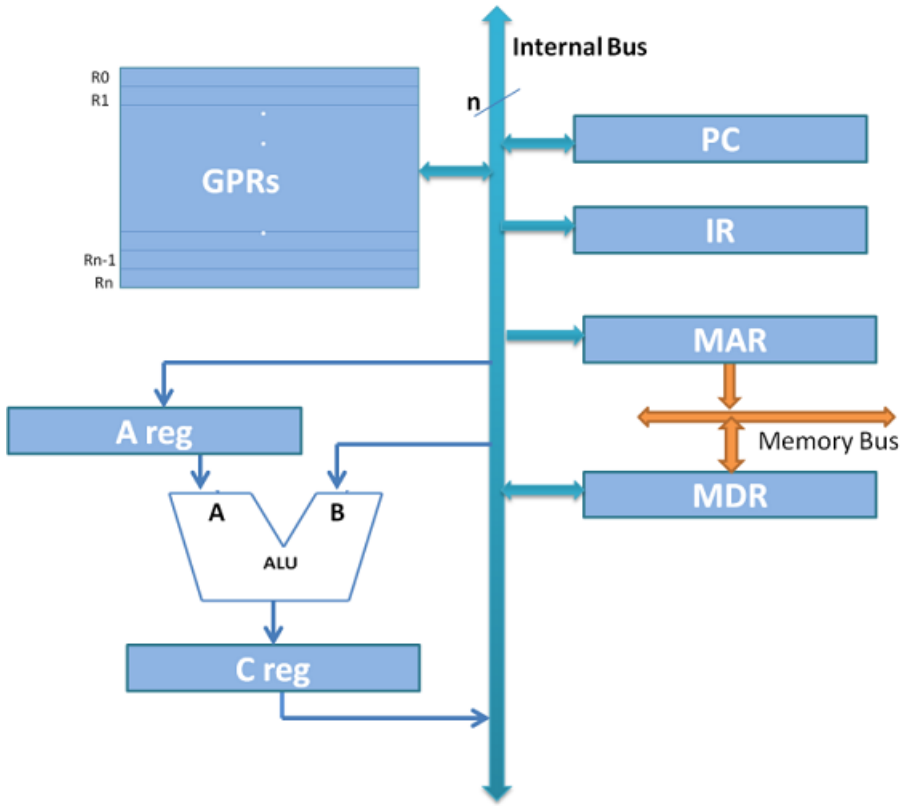
Jedinice koje sudjeluju u izvršavanju – I

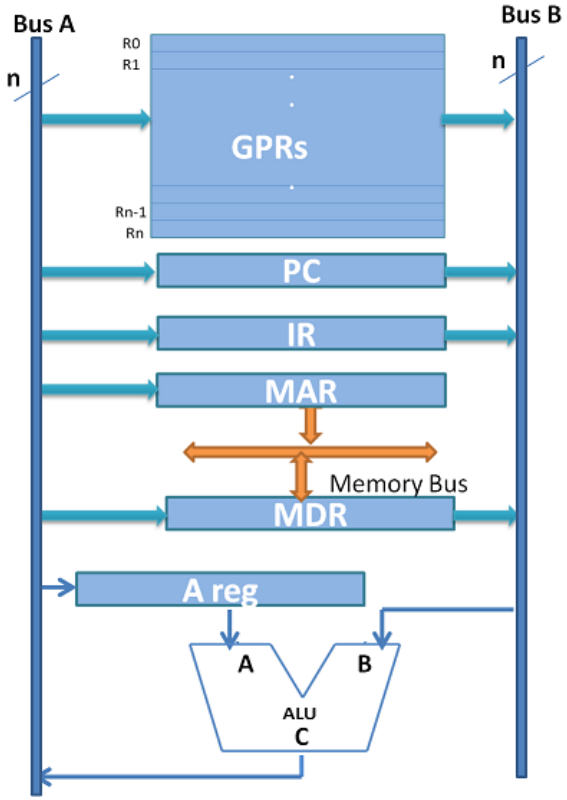
- ALU – rad sa podacima, aritmetika, kalkulacija adresa
- Instrukcijski registar I decoder – dekodiranje instrukcije koja se treba izvršiti
- Programski brojač – uvijek pokazuje na slijedeću instrukciju u nizu, upravljanje slijedom izvršavanja
- Memorija
  - Instrukcijska memorija u *fetch* fazi je skoro uvijek *read-only*
  - Podatkovna memorija – potrebna za dohvat operanada I pohranu rezultata
  - MAR (Memory Address Register) – sadrži memorijsku lokaciju kojoj treba pristupiti
  - MDR (Memory Data Register) – sadrži memorijsku lokaciju podataka koje treba dohvatiti

Jedinice koje sudjeluju u izvršavanju – II

- CPU registri – obično su izvedeni kao vrlo brze memorije – obavljaju operacije
- Interni registri, multiplekseri, …
- Interna sabirnica – povezuje sve ove elemente
- Kontrolna jedinica – The Boss koji upravlja svim elementima
- Ovi elementi moraju biti povezani i raditi sinhrono
- Ovisno o dizajnu, elementi mogu biti povezani preko jedne ili više sabirnica

Datapath sa 1 ili 2 sabirnice

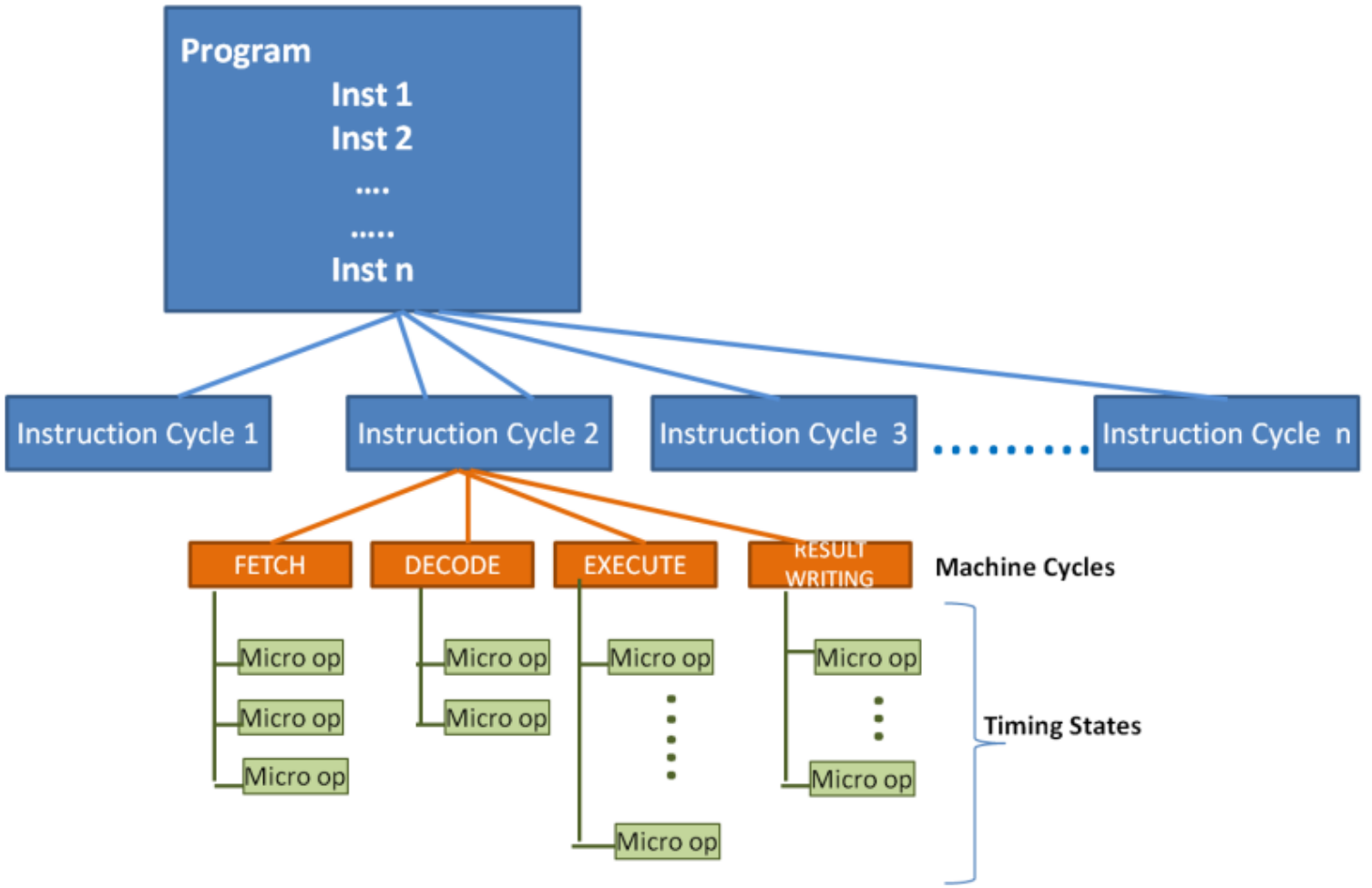




Što to sve skupa točno znači iz aspekta izvršavanja nekakvog programa?

- Program se sastoji od seta instrukcija
- Svaka instrukcija ima makro-cikluse, tipično FETCH, DECODE, EXECUTE I RESULT WRITING
- Broj koraka ovisi o dizajnu procesora – npr.neki procesori imaju integriranu detekciju prekida, neki ne
- Također, FETCH se obično promatra kroz dvije faze – dohvat instrukcije I dohvat operanada nad Kojima se instrukcija izvršava
- Svaki od ovih makro-koraka ima mikro-korake, tzv. *Atomic* operacije
- Brzina izvršavanja operacija ovisi o taktu procesora

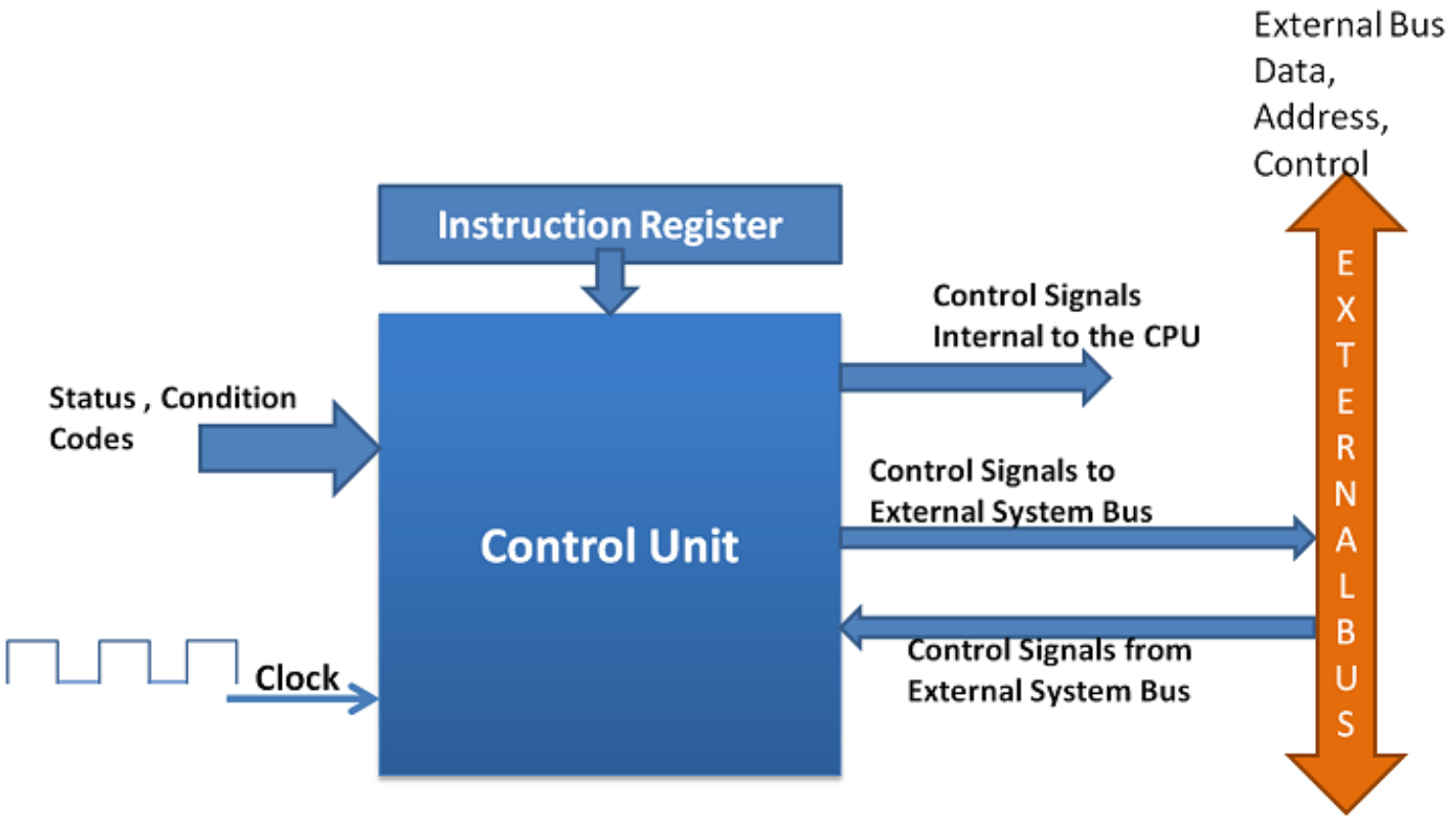
Tipični tok izvršavanja programa



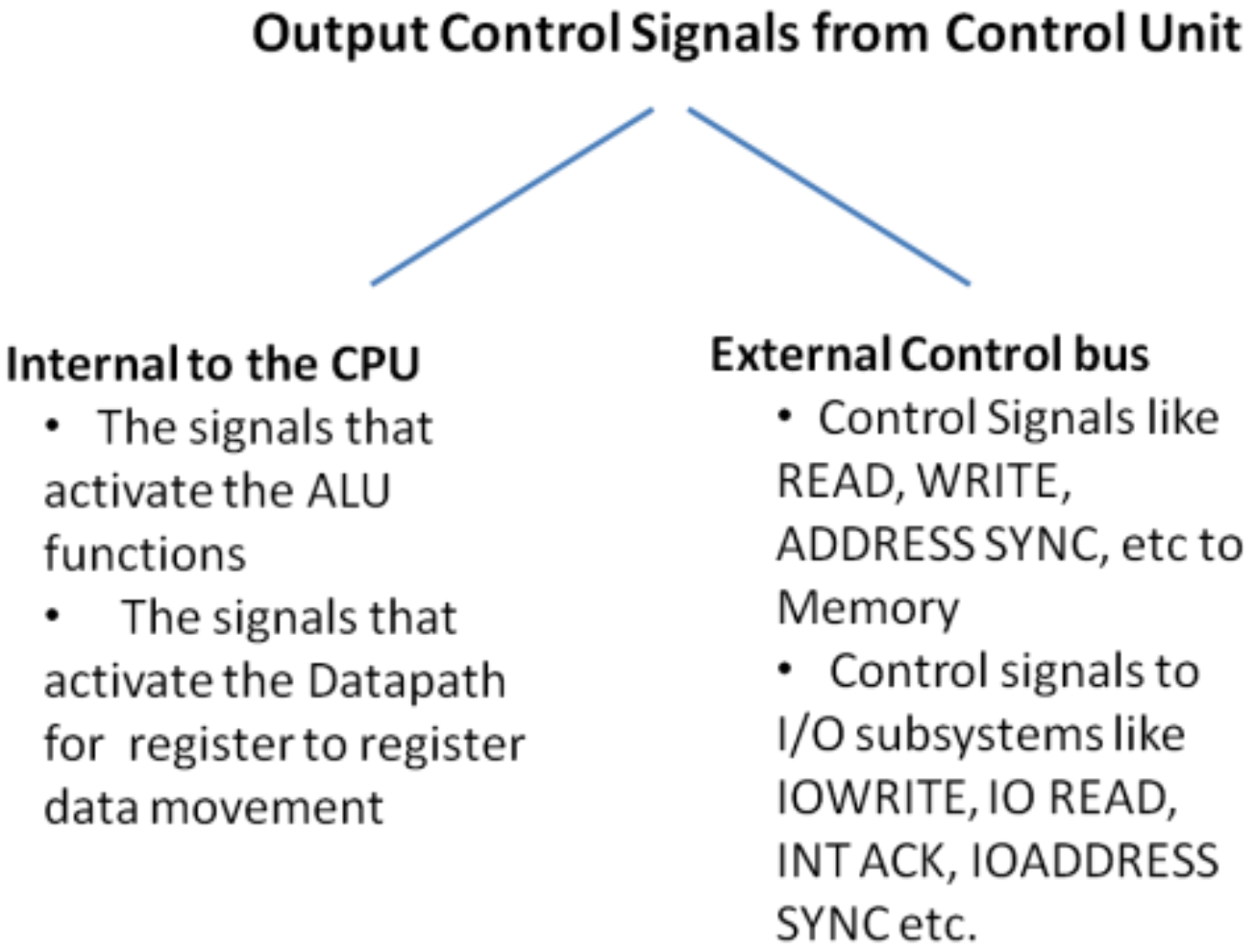
Što je onda, zapravo, kontrolna jedinica?

- Jedinica stanja (*state machine*) koja generira kontrolna stanja ovisno o ulazu, dok se izlaz generira s obzirom na uvjete I podatke na ulazu
- Bitni parametri kontrolne jedinice:
  - Processor clock – pogoni kontrolnu jedinicu I sinhronizira mikro-operacije, jedna ili više mikro-operacija po taktu – više operacija po taktu moguće je kod korištenja cjevovoda (*pipelining*), paralelizma, ili kod izvođenja operacija koje nisu u konfliktu
  - Instrukcijski registar (IR) – sadržava opcode za trenutnu instrukciju, direktno utječe na izdavanje mikro-operacija
  - Zastavice, kontrolni kodovi – zastavice tipa *Trap, Interrupt, Parity* I sl. – odnose se na trenutno stanje procesora I uvjete, kako bi se znao rezultat prethodno izvedenih operacija na procesoru
  - Eksterna kontrolna sabirnica – postoji zato da bi procesor mogao komunicirati I sa vanjskim uređajima – npr. prekid iz I/O sustava znači da je potrebna *pažnja* procesora

Model kontrolne jedinice



Signali kontrolne jedinice



Tipovi instrukcija iz aspekta kontrolne jedinice

- Generalne instrukcije – izvršavaju se direktno na procesoru, sekvencijalno kako dolaze na izvršavanje
  - Uloga kontrolne jedinice po von Neumannu – instrukcija po instrukcija, serijski, kontrola, *rinse*, *repeat*
- Instrukcije sa grananjem (*branching instructions*) – implementirane direktno od strane kontrolne jedinice
  - Kada se izvršavaju ovakve instrukcije, u PC se upisuje memorijska adresa gdje se kreće u grananje
  - Instrukcije tipa uvjetno izvršavanje (*conditional branching*), bezuvjetno izvršavanje (*jump*), poziv pod-rutina, povratak iz pod-rutina

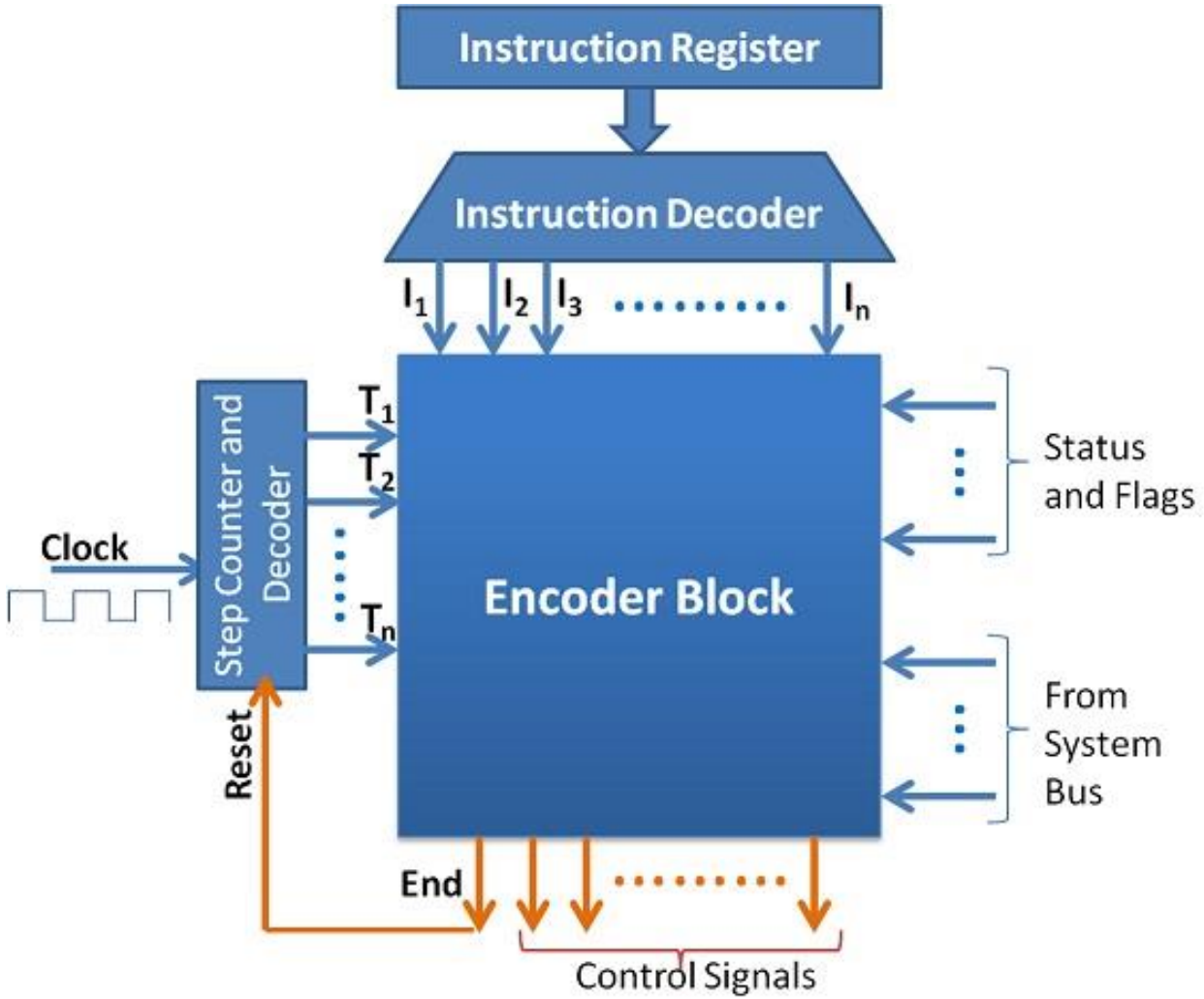
Dizajn I vrste kontrolne jedinice

- Radi se nakon što se analizira put podataka (*datapath*)
- Analitika datapatha nam zapravo govori koji kontrolni signali su nam potrebni, što direktno utječe na dizajn kontrolne jedinice
- CU zapravo radi sekvenciranje (operacije dohvata podataka) I izvršavanje na dostupnim izvršnim jedinicama
- Dva pristupa dizajnu logike kontrolne jedinice
  - *Hardwired*
  - *Micro-programmed*

Hardwired logika CU

- Redoslijed operacija ovisi o tome kako su logički elementi spojeni unutar sklopova (*hardwired*)
- Logika/sklopovi koriste gateove, dekodere, enkodere, brojače, direktno spojene u CU
- Mane:
  - Funkcionalno ograničavajuće
  - Komplicirano za izvedbu, dizajn, I testiranje
  - Teško dodati nove instrukcije
  - Kada su u pitanju CISC procesori, previše ograničavajuće s obzirom na količine instrukcija koje procesor podržava
- Prednosti:
  - Hardwired CU je puno brži u izvršavanju od mikroprogramirane CU
  - Vrlo popularna logika kod RISC-based procesora – RISC procesori su obično 1i/pc mašine pa im ovakva logika odgovara s obzirom na arhitekturu

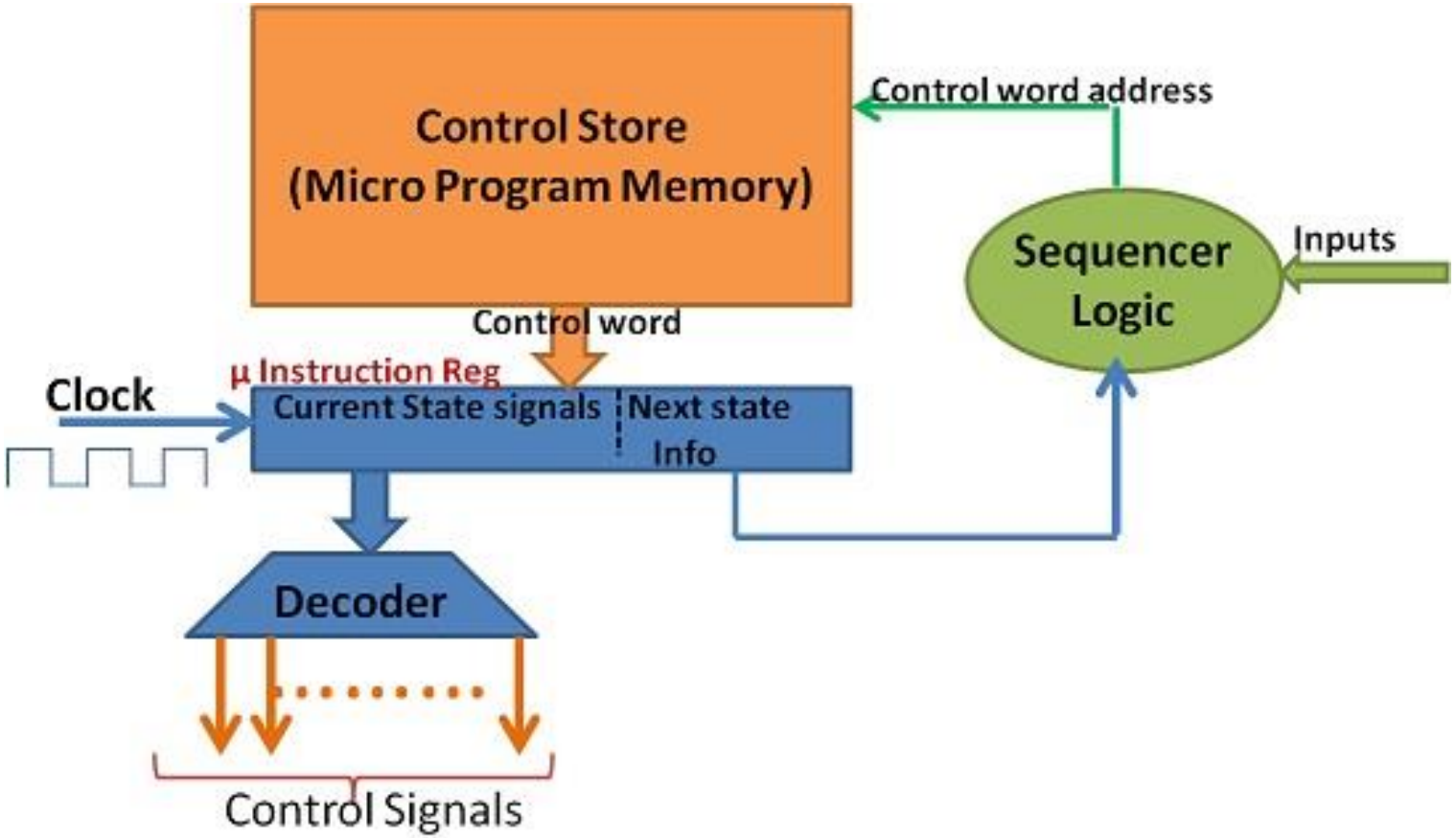
Tipična *hardwired* izvedba CU



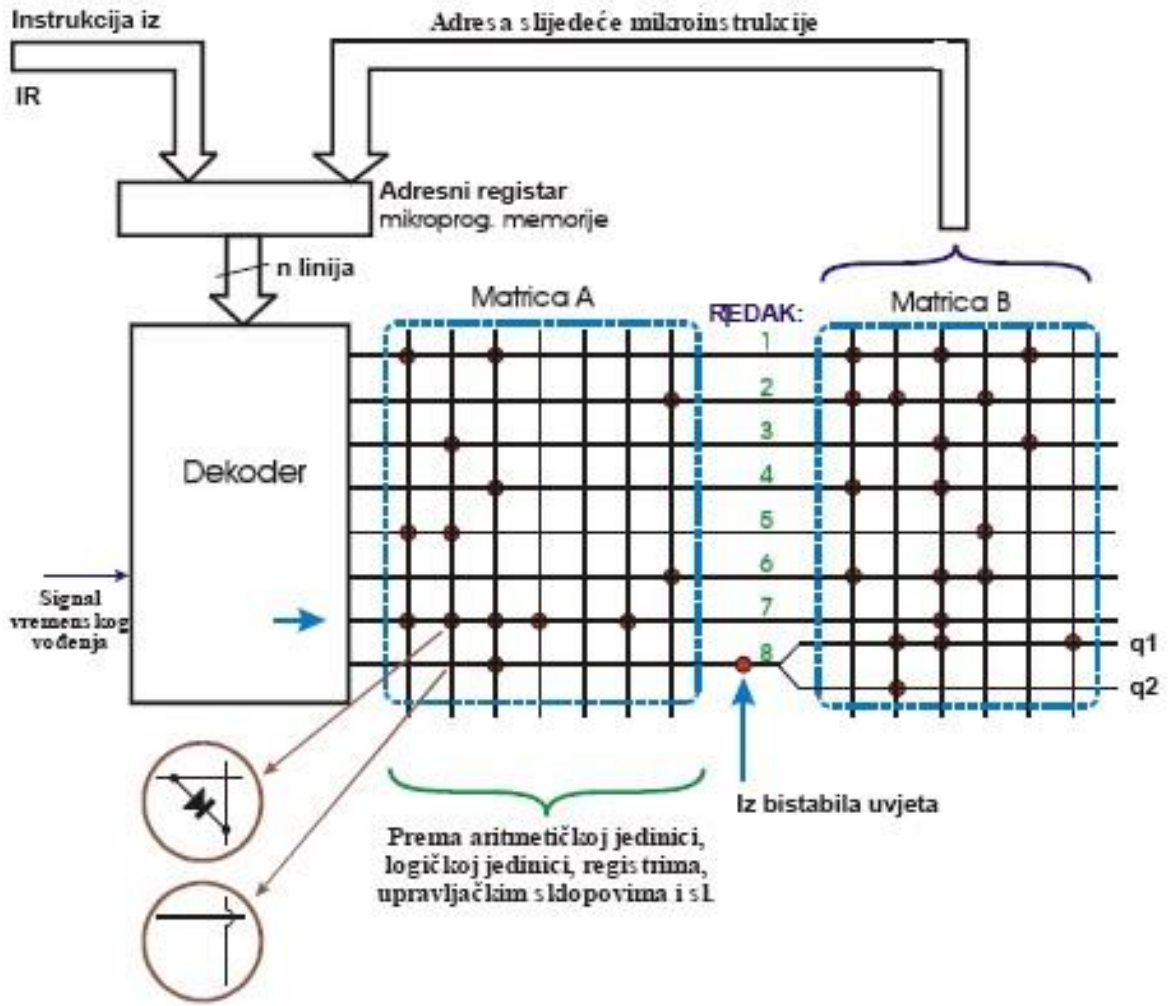
Micro-programmed logika CU

- Potpuno drugačiji dizajn, u kojem memorija igra esencijalnu ulogu
- Kaže se da su ovo memory-based kontrolne jedinice
- Sadržaj memorijske lokacije, nakon dekodiranja, zapravo postaje temelj za generiranje kontrolnih signala za trenutno stanje izvršavanja
- Informacija o slijedećem stanju je u memoriji, dohvaća se, I ciklus kreće ponovo
- "računalo u računalu" (engl. computer-withincomputer)
- Upravljačka jedinica tijekom izvođenja mikroprograma prolazi kroz:
  - mikrofazu – pribavi – kada se dohvaća mikroinstrukcija
  - mikrofazu – izvrši – kada se mikroinstrukcija izvršava

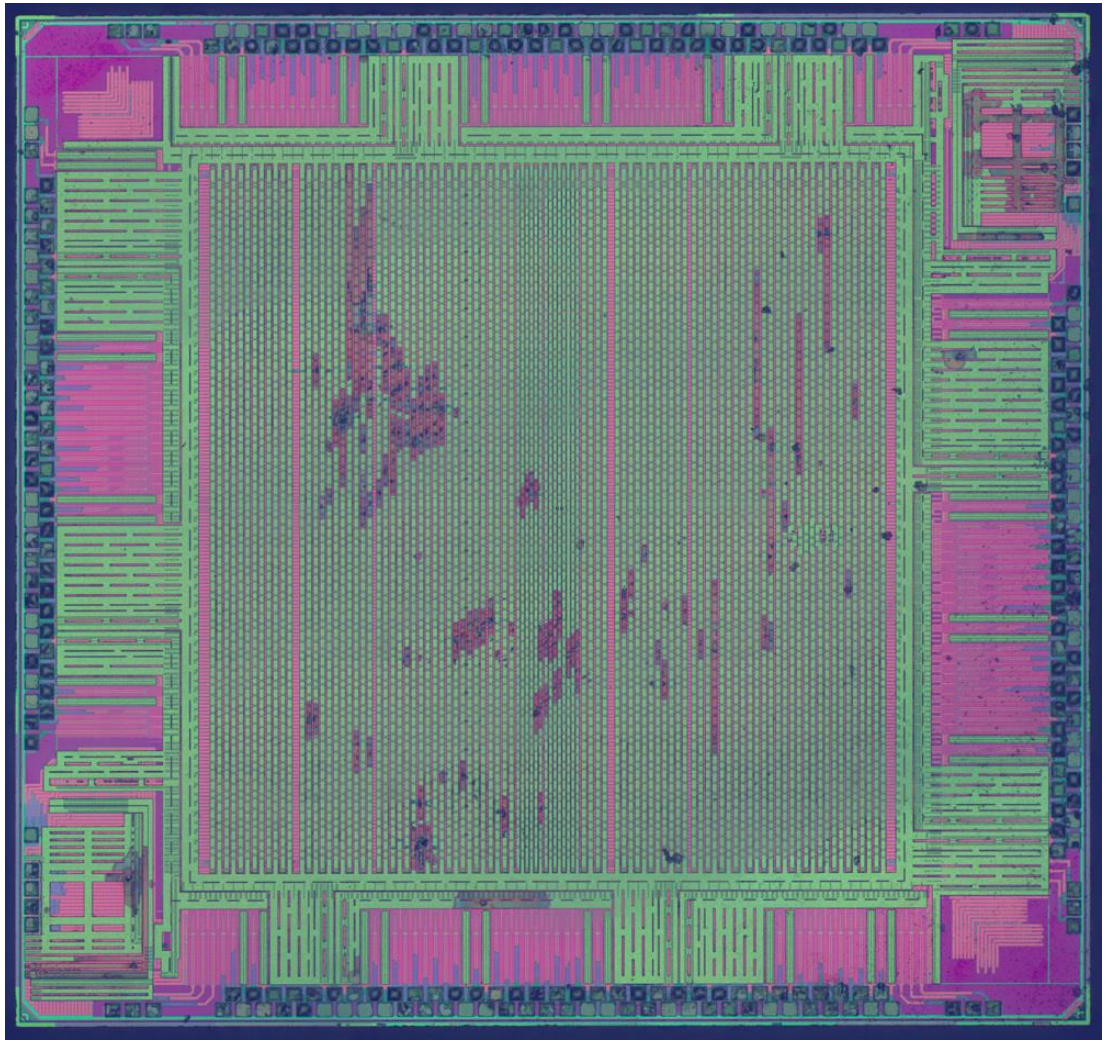
Tipična izvedba *micro-programmed* CU



Wilkesova shema mikroprogramirane CU



Altera
Cyclone V
FPGA



Vrste micro-programmed CU

- Sa pohranom u jednoj razini
- Sa pohranom u dvije razine – osim kontrolne memorije za mikroinstrukcije ima I memoriju za nanoinstrukcije
  - Kod ovakve CU nano-instrukcijska memorija zna sve kombinacije kontrolnih signala
  - Korištenje ovakve CU može značiti smanjenje duljine riječi u mikroinstrukcijama, što znači I manju potrebu za memorijom na razini CU

Prednosti I mane

- Prednosti:
  - Puno jednostavniji dizajn I modifikacija
  - Dizajniranje arhitekture I mikrokoda se može raditi paralelno
  - Problemi kontrolne logike se mogu rješiti nadogradnjom firmware-a
  - Ako je izvedena kako treba, obično bolje koristi interne registre
  - Puno bolje prilagođena procesorima sa kompliciranim instrukcijskim setom (CISC)
- Mane:
  - Obično sporija od hardwired
  - Dosta problematična kod paralelizma
  - Često traži dosta memorije, što može biti problem dizajna I cijene

Hvala na pažnji!