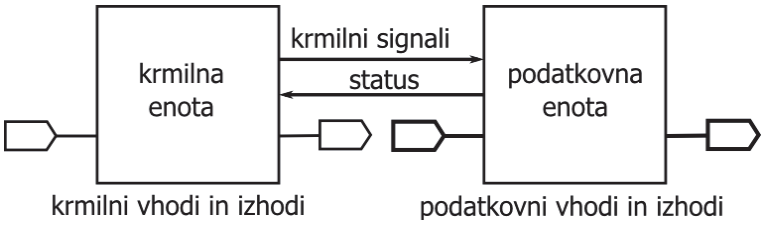
8
Digitalni sistemi

Digitalni sistem je kompleksno digitalno elektronsko vezje, narejeno za izvedbo ene ali več nalog. Sestavljen je iz podsistemov ali enot za prenašanje, obdelovanje podatkov in nadzor oz. krmiljenje. Slika 8.1 prikazuje delitev digitalnega sistema na podatkovno in krmilno enoto. Krmilna enota sprejema krmilne (kontrolne) vhode in oddaja izhode za interakcijo z drugimi deli sistema. Krmilne signale pošilja v podatkovno enoto in iz nje dobiva informacijo o stanju (status). Tako aktivira različne operacije obdelave podatkov. Delovanje krmilne enote določa diagram stanj ali pa program v primeru *programirane* krmilne enote.



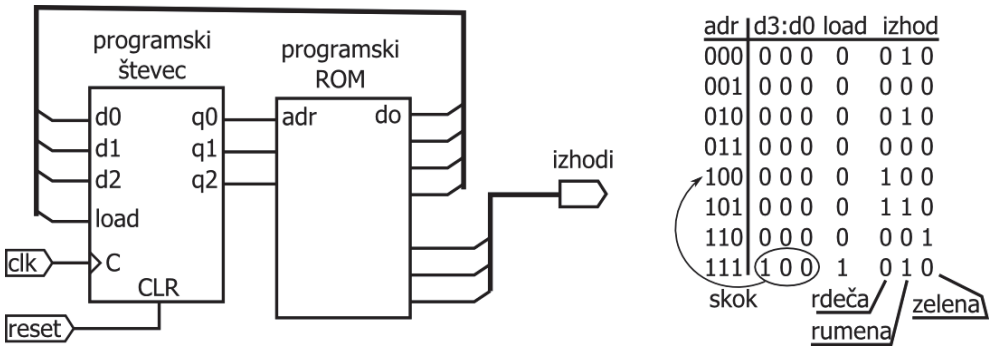
Slika 8.1: Delitev digitalnega sistema na podsisteme.

Podatkovna enota prenaša in obdeluje podatke z vhodov in jih pošilja na izhode. Delovanje enote določajo registri in operacije nad podatki, ki jih hranijo registri. Obravnavali bomo registrske *mikrooperacije*: prenos, seštevanje, odštevanje, logične operacije in pomikanje podatka. Predstavili bomo mikroprocesorske sisteme in delovanje centralne procesne enote.

8.1 Krmilna enota

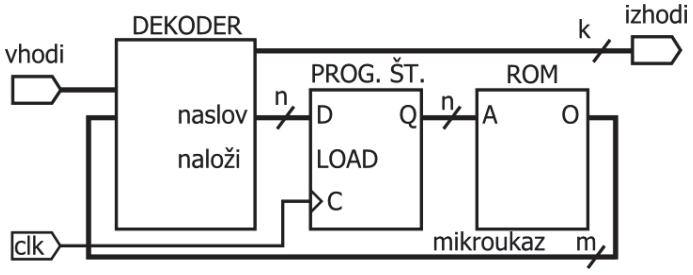
Krmilna enota je sekvenčno vezje, ki nadzira delovanje digitalnega sistema. Enoto z vnaprej definiranimi krmilnimi operacijami naredimo v obliki sekvenčnega stroja. Kadar potrebujemo večjo prilagodljivost in možnost spreminjanja krmilnega postopa, naredimo *mikroprogramirano* krmilno enoto. *Mikroprogramirana* krmilna enota ali mikrosekvenčnik ima krmilni postopek zapisan v programskem pomnilniku. Najpreprostejša izvedba vsebuje pomnilnik vrste ROM in programski števec v povratni vezavi.

Za ilustracijo si bomo ogledali delovanje enostavnega mikrosekvenčnika za krmiljenje semaforskih luči, ki je sestavljen iz 3-bitnega programskega števca in pomnilnika. V pomnilniku so shranjeni *mikroukazi*, predstavljeni v tabeli slike 8.2. Programski števec naslavlja pomnilnik in s tem poskrbi, da se ob uri izvaja zaporedje mikroukazov: ko je signal load na 0, se izvajajo mikroukazi po vrsti, pri 1 pa se izvede skok na poljuben naslov.



Slika 8.2: Mikrosekvenčnik s 3-bitnim izhodom in tabela za izvedbo semaforja.

Ob resetu se začnejo po vrsti izvajati mikroukazi, ki najprej povzročijo utripanje rumene luči, potem pa vklapljanje luči od rdeče do zelene. Zadnji mikroukaz povzroči, da se izvajanje nadaljuje na ukazu z binarnim naslovom 100, kar sproži ponoven cikel prižiganja luči.



Slika 8.3: Mikroprogramirana krmilna enota z ukaznim dekodirnikom.

Kompleksnejše enote vsebujejo še kombinacijski dekodirnik. Dekodirnik krmili programski števec glede na vrednosti trenutnega mikroukaza in krmilnih vhodov. Mikroukazi so v tem primeru zapisani bolj kompaktno, z manj biti, saj ni treba da bi posamezen bit neposredno krmilil programski števec ali izhode. Tako so narejene tudi krmilne enote mikroprocesorjev.

8.2 Registrske mikrooperacije

Prenos podatkov med registri in obdelavo podatkov imenujemo *registrski prenos* ali z angleško kratico RT (Register Transfer). Gradniki RT-vezij so registri, ki izvajajo eno ali več elementarnih operacij. Elementarne operacije ali *mikrooperacije* so npr. nalaganje podatka v register ali prištevanje k vrednosti v registru. Mikrooperacije se izvedejo v enem ciklu na vseh podatkovnih bitih hkrati. Rezultat operacije se lahko shrani nazaj v register ali pa prenese v naslednji register. Spoznali bomo štiri vrste mikrooperacij: prenos, aritmetične, logične in mikrooperacije pomika.

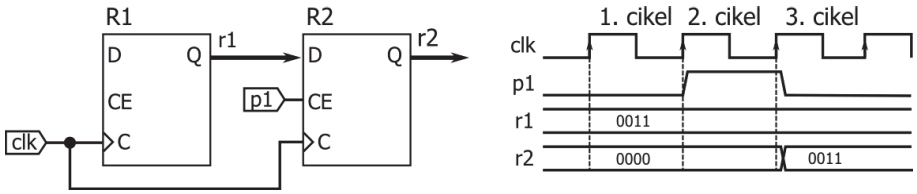
Registrski prenos

Vzemimo dva registra $R1$ in $R2$, ki imata izhodna signala $r1$ in $r2$ z enakim številom podatkovnih bitov. Prenos podatka iz enega v drug register zapišemo z izrazom:

$$r2 \leftarrow r1$$

Register $R1$ predstavlja izvor podatkov, register $R2$ pa ponor ali ciljni register. Vsebina registra, ki je izvor podatkov, se pri prenosu ne spremeni, spremeni se le podatek v ciljnem registru. Podatki se prenesejo ob fronti ure, na katero sta vezana oba registra. Če ne želimo, da se prenos izvede ob vsakem urnem ciklu, uporabimo pogojni registrski prenos. Pogoj je logični izraz, ki je vezan na kontrolni vhod CE drugega registra. Slika 8.4 prikazuje vezje iz dveh registrov, ki prenese podatek v drug register, kadar je vrednost kontrolnega signala $p1$ enaka 1.

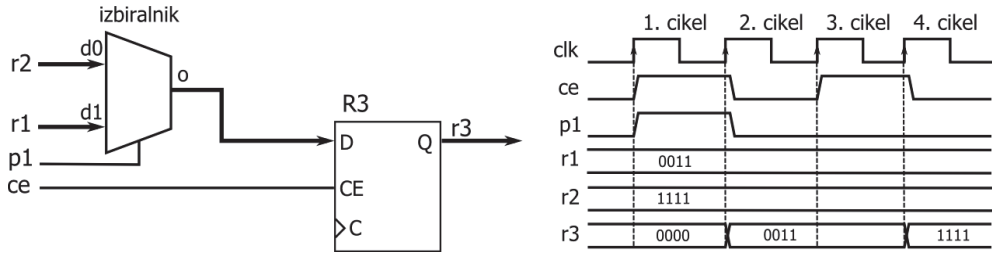
$$r2 \leftarrow r1 \mid p1 = 1$$



Slika 8.4: Vezje za pogojni prenos podatkov med dvema registroma.

V vezju so običajno vsi signali sinhronizirani z uro, zato tudi pričakujemo, da se pogoj $p1$ aktivira ob naraščajoči fronti ure. V časovnem diagramu na sliki 8.4 se $p1$ postavi na 1 v drugem urnem ciklu. Prenos podatka pa se naredi ob naraščajoči fronti tretjega cikla. V sinhronih vezjih vedno velja, da se registrski prenos izvrši en cikel kasneje, kot je izpolnjen pogoj. Razlog za takšno delovanje so zakasnitve, ki jih ima vsako realno vezje. Če se pogoj aktivira ob nekem urnem ciklu, bo zaradi zakasnitev vrednost kontrolnega signala postavljena na 1 za fronto ure, drug register pa bo sprejel novo vrednost šele ob naslednji naraščajoči fronti.

Nadgradnja osnovnega prenosa je izbirni prenos, kjer imamo več izvorov, iz katerih se podatek shrani v register. Osnovno vezje je sestavljeno iz registra in izbiralnika.



Slika 8.5: Vezje za registrski prenos z izbirnim signalom in časovni diagram.

V ciljni register $R3$ se ob aktivnem signalu $ce = 1$ shrani podatek iz vodila $r1$, kadar je $p1 = 1$, sicer pa iz vodila $r2$. Na časovnem diagramu vidimo, da se dejanski prenos podatka izvede en cikel za tem, ko sta postavljena kontrolna signala ce in $p1$. Mikrooperacijo za izbirni prenos s pogojem $p1$ zapišemo:

$$r3 \leftarrow r1 \mid p1 = 1 \text{ AND } ce = 1$$

$$r3 \leftarrow r2 \mid p1 = 0 \text{ AND } ce = 1$$

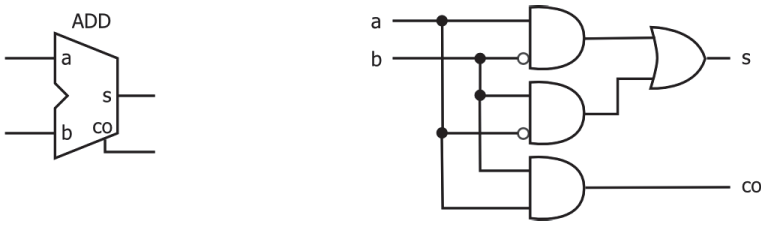
Aritmetične mikrooperacije

Osnovna aritmetična operacija je seštevanje. Naredimo izračun vsote dveh enobitnih dvojiških vrednosti pri vseh možnih kombinacijah:

$$\begin{array}{r} 0 \\ +\,0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0 \\ +\,1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +\,0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +\,1 \\ \hline 10 \end{array}$$

prenos

Pri zadnji kombinaciji smo dobili prenos, ki ga upoštevamo kot dodatno števko. Logično vezje za izračun vsote se imenuje seštevalnik. Slika 8.6 prikazuje simbol in logično shemo enobitnega seštevalnika. Na vhodu sta operanda $a0$ in $b0$, na izhodu pa je vsota $s0$ (angl. sum) in prenos $c0$ (angl. carry).

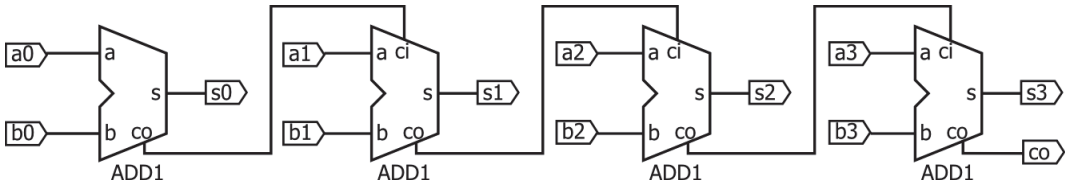


Slika 8.6: Simbol in logična shema enobitnega seštevalnika.

Postopek seštevanja večmestnih števil poznamo že iz ročnega seštevanja v desetiškem sistemu: števili poravnamo in seštevamo števke od desne proti levi. Če pride pri vsoti števk do prenosa, ga upoštevamo pri naslednji vsoti:

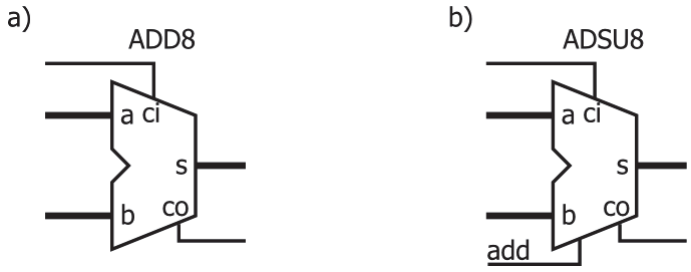
$$\begin{array}{r} 0\,0\,1\,1 \\ +\,0\,0\,0\,1 \\ \hline 0 \end{array} \quad \begin{array}{r} 0\,0\,1\,1 \\ +\,0\,0\,0\,1 \\ \hline 0\,0 \end{array} \quad \begin{array}{r} 0\,0\,1\,1 \\ +\,0\,0\,0\,1 \\ \hline 1\,0\,0 \end{array} \quad \begin{array}{r} 0\,0\,1\,1 \\ +\,0\,0\,0\,1 \\ \hline 0\,1\,0\,0 \end{array}$$

Vezje večbitnega seštevalnika naredimo s povezavo enobitnih seštevalnikov z vhodnim in izhodnim prenosom. Seštevalniki ADD1 na sliki 8.7 računajo vsoto operandov a, b in vhodnega prenosa ci. Vsak izmed seštevalnikov izračuna en bit vsote. Po seštevanju nižjih bitov se izhodni prenos upošteva kot vhod v naslednji seštevalnik. Vsoto sestavljajo biti od $s0$ do $s3$ in izhodni prenos co.



Slika 8.7: Shema 4-bitnega seštevalnika.

Seštevalniki so pogosto uporabljeni elementi, zato dobimo v knjižnicah osnovnih gradnikov že pripravljene večbitne seštevalnike. Slika 8.8 a) prikazuje simbol seštevalnika ADD8 z 8-bitnimi vhodi a in b ter izhodom s. Seštevalnik ima tudi vhod in izhod za prenos, tako da lahko z zaporedno vezavo zgradimo še večje seštevalnike.

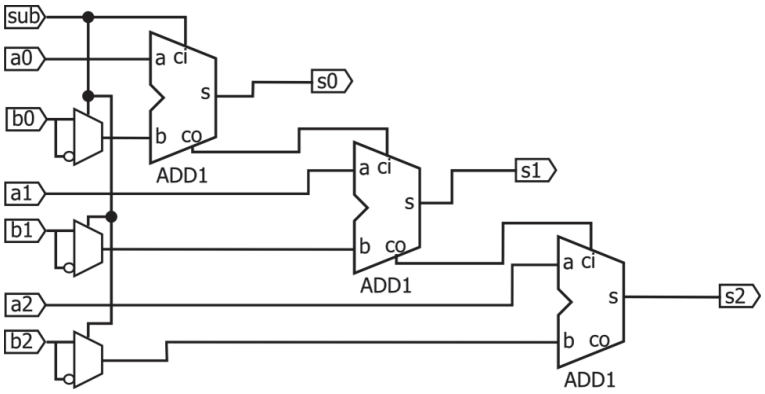


Slika 8.8: Simboli: a) 8-bitni seštevalnik in b) 8-bitni seštevalnik/odštevalnik.

Enak simbol se uporablja tudi za splošno aritmetično enoto, ki izvaja različne računske operacije nad večbitnimi signali. Slika 8.8 b) prikazuje simbol 8-bitne seštevalno/odštevalne enote ADSU8. Dodatni vhod add določa, ali enota izvaja operacijo seštevanja ($add = 1$) ali odštevanja ($add = 0$). Odštevanje binarnih števil naredimo s pomočjo dvojiškega komplementa, tako da odštevanec negiramo in naredimo vsoto z vhodnim prenosom:

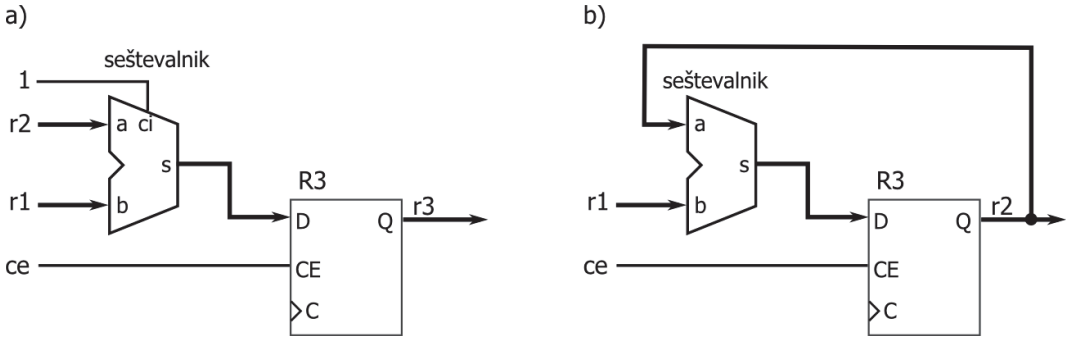
$$\begin{array}{r} 0\,0\,1\,1 \\ -\,0\,0\,0\,1 \\ \hline 0 \end{array} \quad \begin{array}{r} 0\,0\,1\,1 \\ +\,1\,1\,1\,0 \\ \hline 1 \\ 0 \end{array} \begin{array}{l} \leftarrow \text{negacija} \\ \leftarrow \text{prenos} \end{array} \quad \begin{array}{r} 0\,0\,1\,1 \\ +\,1\,1\,1\,0 \\ \hline 1\,1\,1\,1 \\ 0\,0\,1\,0 \end{array}$$

Slika 8.9 prikazuje izvedbo 3-bitnega gradnika za izračun vsote ali razlike. Kadar je signal sub na 0, bo vezje izračunalo vsoto vhodov a in b; kadar je sub enak 1, pa bodo prišle na vhod seštevalnikov negirane vrednosti b, na vhod prvega pa še prenos in rezultat bo razlika vrednosti.



Slika 8.9: Izvedba 3-bitnega seštevalno-odštevalnega gradnika.

Če povežemo na izhod večbitnega seštevalnika register, dobimo aritmetično mikrooperacijo. Seštejemo lahko izhodne vrednosti dveh registrov in vsoto shranimo v tretjega ($r3 \leftarrow r1 + r2$) ali pa enemu registru prištejemo vsoto drugega ($r2 \leftarrow r2 + r1$). Vezje se v drugem primeru imenuje akumulator in vsebuje povratno zanko.



Slika 8.10: Vezje za mikrooperacijo seštevanja: a) vsota $r1 + r2 + 1$ in b) akumulator.

Akumulator v praktični izvedbi potrebuje začetno vrednost v registru, ki jo naložimo prek dodatnega vhodnega izbiralnika ali pa s signalom reset:

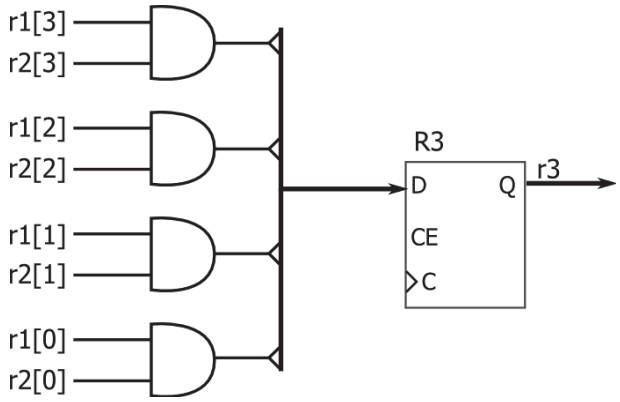
$$r2 \leftarrow r2 + r1 \mid reset = 0$$
$$r2 \leftarrow 0000 \mid reset = 1$$

Odštevanje binarnih vrednosti naredimo z uporabo dvojiškega komplementa. Vrednost, ki jo želimo odšteti, invertiramo in ji prištejemo 1 (vhodni prenos ci), nato pa seštejemo z drugo vrednostjo: $r3 \leftarrow (\text{ NOT } r1 + 1) + r2$.

Povečevanje (angl. increment) in zmanjševanje (angl. decrement) sta aritmetični mikrooperaciji, pri katerih je eden izmed vhodov konstantna vrednost, najpogosteje kar 1. Kadar prištevamo ali odštevamo vrednost k istemu signalu, dobimo dvojiški števec. Množenje in deljenje ne spadata med osnovne mikrooperacije, ker jih lahko naredimo z zaporedjem osnovnih operacij; množenje je narejeno kot zaporedno seštevanje in pomikanje vrednosti. Kadar potrebujemo v vezju hiter množilnik ali delilnik, ga naredimo v obliki paralelnega kombinacijskega vezja. Ko se rezultat pojavi na izhodu kombinacijskega množilnika ali delilnika, ga ob fronti ure shranimo v register in v tem primeru vezje predstavlja mikrooperacijo.

Logične mikrooperacije

Logične mikrooperacije so uporabne za manipulacijo s posameznimi biti v registru, saj se logična operacija izvaja nad vsakim bitom posebej. Osnovne so invertiranje ($r2 \leftarrow$ NOT $r1$), logična *in* ($r3 \leftarrow r1$ AND $r2$) ter logična *ali* ($r3 \leftarrow r1$ OR $r2$).



Slika 8.11: Izvedba logične mikrooperacije.

Vezje logičnih mikrooperacij vsebuje paralelno vezana logična vrata in register. Invertiranje je npr. uporabno pri izvedbi mikrooperacije odštevanja z uporabo dvojiškega komplementa. Drugi dve logični mikrooperaciji pa sta uporabni za maskiranje bitov. Če imamo npr. 8-bitno vrednost $r1$ in bi želeli dobiti rezultat, pri katerem so zgornji 4 biti postavljeni na 0, spodnji pa enaki spodnjim bitom signala $r1$, naredimo operacijo:

$$r3 \leftarrow r1 \text{ AND } 00001111$$

Operacija se izvrši nad istoležnimi biti:

$r1$ (podatek)	10100101
$r2$ (maska)	00001111
$r3$ (rezultat)	00000101

Kadar izvajamo maskiranje z operacijo *in*, se pobrišejo vsi biti, ki imajo v maski vrednost 0. Obratno je pri maskiranju z operacijo *ali*, kjer se vsi biti, ki so v maski postavljeni na 1, tudi na rezultatu postavijo na 1.

Mikrooperacije pomika

Z mikrooperacijami pomika spreminjamo mesta posameznih bitov v registru. Osnovni mikrooperaciji sta pomik vseh bitov za eno mesto v desno in pomik vseh bitov za eno mesto v levo. Pomik za eno mesto v desno predstavlja deljenje številske vrednosti z 2, pomik v levo pa množenje vrednosti z 2.

Poglejmo primer pomika 4-bitne vrednosti $r1$ za eno mesto v desno:

$r1$ (podatek)	1010
$r2$ (pomaknjena vrednost)	0101

Pomaknjena vrednost ima najvišji bit vedno postavljen na 0, nato pa sledijo zgornji trije biti vhodne vrednosti, najnižji bit pa pri pomikanju izpade. Enačba mikrooperacije za pomik 4-bitne vrednosti r1 za eno mesto v desno je:

$$r2 \leftarrow \{0, r1[3..1]\}$$

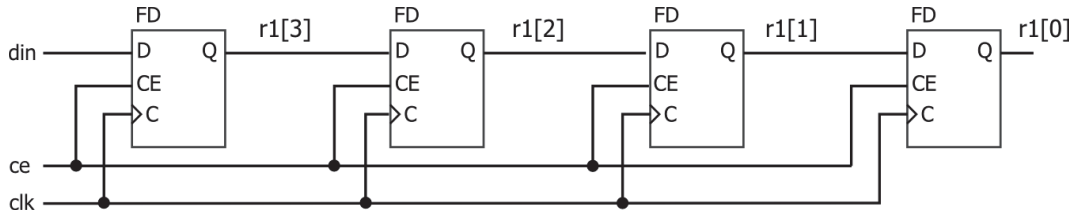
Rezultat je sestavljen iz dveh delov, ki smo ju zapisali v zavitem oklepaju. Prvi del je konstanta 0, drugi del pa sestavljajo biti vektorja $r1$ z indeksi od 3 do 1. Podobno velja pri pomiku za eno mesto v levo, le da tokrat izpade najvišji bit in dodajamo ničlo na desni strani:

$$r2 \leftarrow \{r1[2..0], 0\}$$

Vezje za osnovni mikrooperaciji pomika je zelo preprosto, saj vsebuje le register, ki ima na vhodu ustrezno povezane bite in konstanto 0. Vezje za pomikanje, pri katerem sta izvor in cilj v istem registru, se imenuje pomikalni register. Pri pomikalnem registru moramo poskrbeti, da nanj naložimo podatek. Glede na način nalaganja ločimo paralelne in serijske pomikalne registre.

Slika 8.12 prikazuje 4-bitni serijski pomikalni register, ki izvaja naslednjo operacijo:

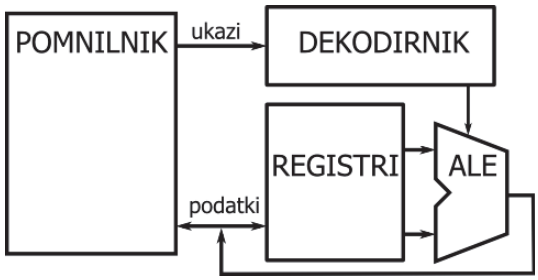
$$r1 \leftarrow \{din, r1[3..1]\} \mid ce = 1$$



Slika 8.12: Serijski pomikalni register.

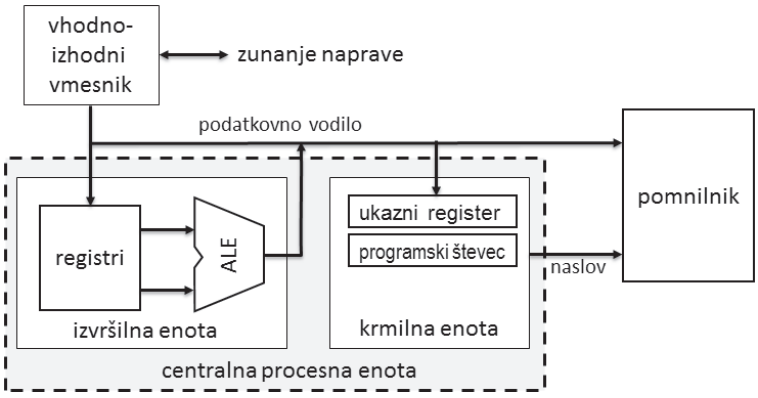
8.3 Mikroprocesorski sistemi

Mikroprocesorski sistemi so večnamenski digitalni sistemi, ki izračunajo in obdelajo podatke po ukazih, shranjenih v pomnilniku. Izraz *mikroprocesor* označuje integrirano vezje, v katerem je računalniška procesna enota. Mikroprocesor prenaša iz pomnilnika podatke in ukaze, ki jih dekodira, in izvede ustrezne mikrooperacije z registri in računskimi enotami. Rezultate začasno shrani v registre ali prenese v pomnilnik.



Slika 8.13: Prenos ukazov in podatkov v mikroprocesorju.

Jedro mikroprocesorja je *centralna procesna enota* (CPE), ki je sestavljena iz krmilne enote in podatkovne poti. Krmilna enota vsebuje ukazni dekodirnik in sekvenčno vezje s programskim števcem, ki določa korake izvajanja ukazov. Izvršilna enota se nahaja na podatkovni poti procesorja in vsebuje registre ter aritmetično-logično enoto (ALE) za obdelavo podatkov. Podatki se prenašajo prek vhodnih in izhodnih vmesnikov.



Slika 8.14: Zgradba preprostega mikroprocesorskega sistema.

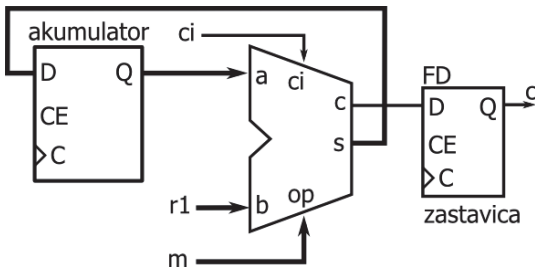
Mikroprocesor izvaja računalniški *program* v obliki zaporedja strojnih ukazov. Procesorji vrste RISC (angl. Reduced Instruction Set Computer) imajo majhen nabor strojnih ukazov z enostavnimi operacijami, ki se izvedejo v enem ali dveh korakih. Procesorji CISC (angl. Complex Instruction Set Computer) pa poznajo veliko strojnih ukazov, med katerimi so kompleksni ukazi, ki potrebujejo več korakov za izvajanje. Zmogljivi mikroprocesorji imajo lahko več jeder in računskih enot, tako da zmorejo izvajati več ukazov hkrati.

Aritmetično-logična enota

Aritmetično-logična enota (ALE) je srce mikroprocesorja, v katerem se odvija obdelava podatkov. Osembitni mikroprocesor ima ALE z 8-bitnim podatkovnim vhodom in izhodom, 32-bitni mikroprocesor pa ima 32-bitno ALE. Tipična ALE ima dva podatkovna vhoda a in b, podatkovni izhod s, kontrolni vhod za izbiro operacij op, vhodni ci in izhodni prenos c. Enota izvaja različne mikrooperacije, kot so na primer:

$s \leftarrow a$	prenos podatka
$s \leftarrow a + b$	seštevanje
$s \leftarrow a - b$	odštevanje
$s \leftarrow$ NOT a	logična negacija
$s \leftarrow a$ AND b	logični *in*
$s \leftarrow a$ OR b	logični *ali*
$s \leftarrow \{0, a[7..1]\}$	pomik v desno
$s \leftarrow \{a[6..0], 0\}$	pomik v levo

Rezultat operacije ALE shranimo v registru, izhodni prenos pa v flip-flopu, imenovanem prenosna zastavica. V praksi ima ALE še druge zastavice: ničelna zastavica se postavi na 1, kadar je rezultat operacije enak 0, negativnostna zastavica pa označuje, da je rezultat negativno število. Izhodni prenos uporabimo pri kombiniranju operacij, npr. naredimo seštevanje s prenosom iz prejšnje vsote. Prenos dobimo tudi pri operacijah pomikanja, pri katerih shranimo v prenos tisti bit, ki pri pomikanju vrednosti izpade.

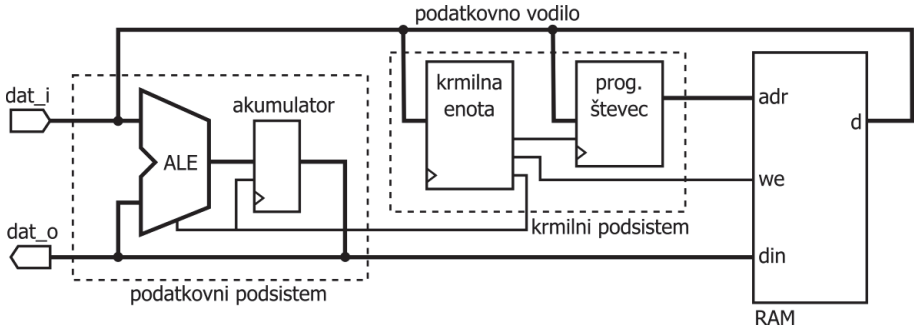


Slika 8.15: Aritmetično-logična enota z akumulatorjem.

Aritmetično-logična enota je lahko v vezavi s povratno zanko, pri kateri se rezultat vedno shrani v register z imenom akumulator, kot prikazuje slika 8.15. Vezava ALE z akumulatorjem je najpreprostejša, ni pa najbolj optimalna, ker mora mikroprocesor ob izvajanju algoritma pogosto prenašati podatke med akumulatorjem in pomnilnikom. Sodobni mikroprocesorji imajo v podatkovnem podsistemu večje število registrov, ki začasno shranjujejo podatke med izvajanjem operacij z ALE.

8.4 Centralna procesna enota

Delovanje centralne procesne enote si bomo podrobneje ogledali na primeru enostavne učne enote CPE4. Ukazi učne enote so sestavljeni iz 4-bitne mikrooperacije in naslova operanda, ki ga prenaša med pomnilnikom in podatkovnim podsistemom. Podatkovni del CPE4 sestavlja aritmetično-logična enota z akumulatorjem in n-bitno (n=8,12,16) podatkovno vodilo. V pomnilniku vrste RAM so shranjeni ukazi in podatki. Delovanje CPE4 časovno usklajuje krmilna enota, programski števec pa določa naslov trenutnega ukaza.



Slika 8.16: Blokovna shema centralne procesne enote CPE4.

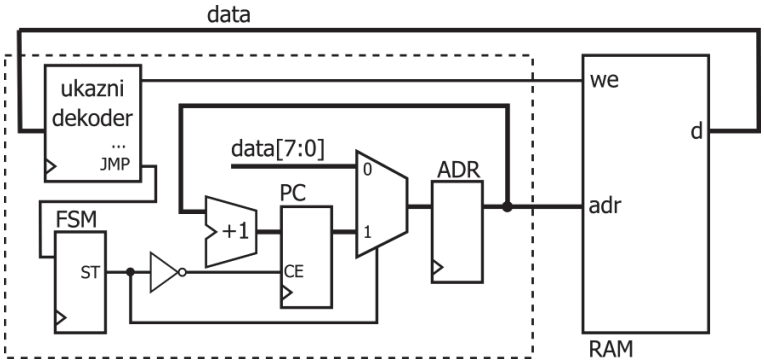
V centralni procesni enoti se izvajajo binarni strojni ukazi, katerih dolžina je odvisna od širine podatkovnega vodila. Ker je strojna koda težko berljiva, jo zapišemo raje v obliki programa v *zbirniku* (angl. assembler), ki ga prevajalnik prevede v strojno kodo.

Ukazi za izvedbo računskih operacij dobijo en operand iz akumulatorja, drugega pa prenesejo iz pomnilnika na določenem naslovu *m(Adr)*. Rezultat operacij se shrani v akumulator. Poglejmo nekaj osnovnih ukazov v zbirniku:

mikrooperacija	**zbirnik**	**pomen**
$a \leftarrow m(Adr)$	LDA Adr	naloži vrednost v akumulator
$m(Adr) \leftarrow a$	STA Adr	shrani akumulator v pomnilnik
$a \leftarrow a + m(Adr)$	ADD Adr	prištej vrednost k akumulatorju
$a \leftarrow a - m(Adr)$	SBT Adr	odštej vrednost od akumulatorja
$a \leftarrow$ NOT a	NOTA	naredi negacijo
$a \leftarrow a$ OR $m(Adr)$	ORA Adr	naredi logično *ali* operacijo
$a \leftarrow a$ AND $m(Adr)$	ANDA Adr	naredi logično *in* operacijo
$a \leftarrow \{a[n-2..0], 0\}$	SHL	pomik akumulatorja v levo
$a \leftarrow \{0, a[n-1..1]\}$	SHR	pomik akumulatorja v desno

V programih potrebujemo poleg računskih tudi skočne ukaze, s katerimi naredimo vejitve programske kode. Procesor CPE4 pozna brezpogojni skok (JMP Adr) in dva pogojna skočna ukaza. Ukaz JZE Adr izvede skok, kadar je v akumulatorju vrednost 0, ukaz JCS Adr pa, kadar je postavljena zastavica prenosa.

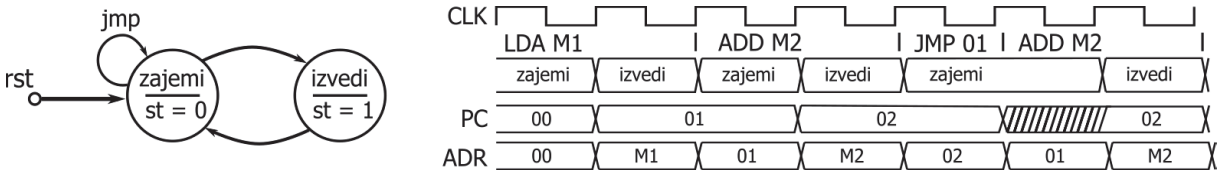
Krmilni podsistem procesorja CPE4 vsebuje ukazni dekoder, sekvenčni stroj (FSM), programski števec (PC) in pomnilniški naslovni register (ADR). Programski števec se po vsakem ukazu poveča za 1, razen pri skočnem ukazu, ko iz podatkovnega vodila naloži nov naslov.



Slika 8.17: Shema krmilnega podsistema procesorja CPE12.

Sekvenčni stroj skrbi za časovne cikle oz. korake izvajanja ukazov. V stanju zajemi se iz pomnilnika preneseta in dekodirata ukaz in naslov, na katerem se nahaja podatek. V stanju izvedi pa se iz pomnilnika prenese podatek in izvede ustrezna operacija. Slika 8.18 prikazuje diagram prehajanja stanj in primer izvajanja kratke kode, ki ima tri ukaze:

```
        LDA M1
zanka:  ADD M2
        JMP zanka
```



Slika 8.18: Diagram stanj krmilne enote in časovni potek izvajanja ukazov.

Za komunikacijo z okolico ima procesor vhodne in izhodne enote. Vhodni vmesnik procesorja prenaša podatke s priključka dat_i v akumulator, izhodni vmesnik pa iz akumulatorja na priključek dat_o z ukazi:

mikrooperacija	**zbirnik**	**pomen**
$a \leftarrow dat\_i$	INP ID	prenos iz vhodne enote ID
$dat\_o \leftarrow a$	OUTP ID	prenos iz registra na izhodno enoto ID

Izhodne enote krmilijo vmesnike, npr. analogno-digitalni pretvornik, serijski vmesnik za prenos podatkov po eni povezavi, pulzno-širinski modulator za pulzno krmiljene enote. Do različnih vmesnikov dostopa CPE z izbiro naslova (ID), ki je del strojnega ukaza INP ali OUTP.

8.5 Obdelava podatkov

Digitalni sistemi izvajajo obdelavo podatkov v binarni obliki: obdelavo signalov, digitaliziranega zvoka, slike ipd. Mikroprocesorski sistemi izvajajo obdelavo podatkov z zaporednimi ukazi v centralni procesni enoti.

Centralna obdelava podatkov

Poglejmo primer algoritma, ki iz vhodnega vmesnika prebere štiri vrednosti, izračuna njihovo povprečje in ga pošlje na izhod. Program prebere prva dva podatka, naredi njuno vsoto, nato pa prebere in prišteje še tretji in četrti podatek. Povprečje dobimo tako, da skupno vsoto delimo s 4, kar storimo z dvema zaporednima pomikoma vrednosti za eno mesto v desno.

Listing 8.1: Program v zbirniku za izračun povprečja štirih vrednosti.

```
start:  INP VHD ; beri 1. podatek
        STA SUM ;   shrani
        INP VHD ; beri 2. podatek
        ADD SUM ; dodaj k trenutni vsoti
        STA SUM ;   shrani vsoto
        INP VHD ; beri 3. podatek
        ADD SUM ; dodaj k trenutni vsoti
        STA SUM ;   shrani vsoto
        INP VHD ; beri 4. podatek
        ADD SUM ; dodaj k trenutni vsoti
        SHR     ; deli z 2
        SHR     ; deli z 2
        STA SUM ;   shrani rezultat
        OUTP IZH ; na izhod
```

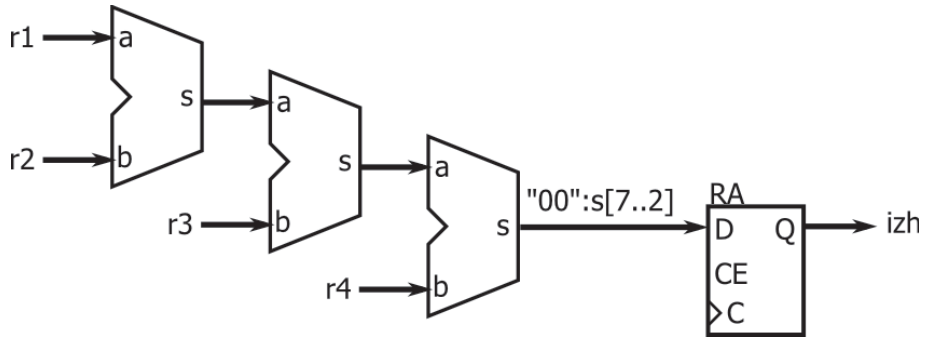
Centralna procesna enota iz pomnilnika nalaga ukaze, jih dekodira in pripravi kontrolne signale, ki izvršijo mikrooperacijo v podatkovni podatkovni poti ter shrani rezultat v izbrani register. Izvajanje ukazov poteka v več korakih, ki jih lahko opazujemo na simulatorju procesorja. Celoten program se izvede v osemindvajsetih urnih ciklih.

Mikroprocesorski sistem s centralno obdelavo podatkov je zelo fleksibilen, saj lahko njegovo delovanje hitro spremenimo s spremembo programa, ki ga naložimo v pomnilnik. Takšni sistemi se zato vgrajujejo v veliko različnih naprav in predstavljajo pomemben del digitalnih elektronskih vezij. Narejeni so v obliki mikroprocesorjev na integriranih vezjih, ki jih programiramo tako, da naložimo nov program v pomnilnik.

Glavna slabost sistemov s centralno obdelavo podatkov je v tem, da potrebujejo veliko urnih ciklov za izvajanje algoritma, ki ga razstavimo na osnovne mikrooperacije. Frekvence ure vgrajenih mikroprocesorjev so nekaj 10 ali nekaj 100 MHz, kar povsem zadostuje za veliko aplikacij. Novejša tehnologija integriranih vezij omogoča doseganje višjih frekvenc delovanja, vendar pa z višjo frekvenco narašča tudi poraba energije. V prenosnih baterijsko napajanih napravah zaradi tega ne moremo uporabljati najhitrejših in najzmogljivejših procesorjev.

Porazdeljena obdelava signalov

Kadar imamo v vezju zelo hitre signale, kot sta npr. hitri podatkovni tok in video signal, moramo uporabiti vezja s porazdeljeno oz. vzporedno obdelavo signalov, saj zaporedna obdelava na CPE ni dovolj hitra. Vzemimo primer izračuna povprečja iz prejšnjega poglavja, ki ga lahko najdemo v aplikaciji za digitalno obdelavo video slike. Vezje naredimo iz treh seštevalnikov in registra, kot prikazuje slika 8.19.



Slika 8.19: Vezje za vzporedni izračun povprečja štirih vrednosti.

Če imamo hkrati na voljo vse štiri vrednosti vhodnih podatkov (signali $r1$, $r2$, $r3$ in $r4$), dobimo rezultat v enem urnem ciklu. Tudi če potrebujemo več ciklov, da na vhode pripeljemo vse vrednosti, je vezje hitrejše od mikroprocesorja, saj ne potrebujemo ciklov za shranjevanje vmesnih rezultatov, pa tudi pomikanje je narejeno le z ustreznimi povezavami. Takšno vezje lahko pri manjši frekvenci ure obdela hiter tok video podatkov v realnem času.

Slabost porazdeljene obdelave signalov je, da potrebujemo več računskih enot in s tem večje vezje, ter da moramo za vsako spremembo algoritma razviti novo vezje. Kadar uporabljamo programirljive digitalne sisteme z vezji CPLD ali FPGA, je tudi takšna rešitev dovolj prilagodljiva, ker omogoča poljubno spreminjanje in nalaganje novega vezja. Glavna slabost je višja cena vezja v primerjavi z mikroprocesorji in omejitev velikosti porazdeljenega vezja, ki pa jo nekoliko kompenzira hiter razvoj tehnologije integriranih vezij.