7 Digitalni sistemi s procesorjem

Digitalni sistemi sprejemajo podatke preko vhodnih signalov in posredujejo rezultate v obliki izhodnih signalov. Glede na način obdelave podatkov ločimo sisteme s centralno enoto za obdelavo podatkov (npr. mikroprocesor) in sisteme s porazdeljeno obdelavo podatkov.

7.1 Centralna obdelava podatkov

Koncept centralne obdelave podatkov temelji na digitalnem gradniku za izvedbo različnih mikrooperacij. Obdelavo podatkov razdelimo na zaporedje mikrooperacij, ki jih izvaja centralno procesna enota (CPE) imenovana **mikroprocesor**. V računalništvu so razvili teorijo in orodja s katerimi še tako zahteven algoritem razstavimo na osnovne aritmetične in logične operacije, ki se izvršijo v digitalnih vezjih.

7.1.1 Aritmetično logična enota

Aritmetično logična enota (ALE) je srce mikroprocesorja v katerem se odvija obdelava podatkov. Osem bitni mikroprocesor ima ALE z 8-bitnim podatkovnim vhodom in izhodom, 32-bitni mikroprocesor pa ima 32-bitno ALE.

Na nivoju registrskih prenosov lahko naredimo univerzalne gradnike, ki jim s parametri določimo katero registrsko operacijo naj izvedejo. Osnovni element teh gradnikov so **registrske celice** sestavljene iz kombinacijske logike in flip-flopa D. Če med seboj povežemo n registrskih celic, dobimo n-bitni register z logiko za izvedbo ene ali več mikrooperacij.

Primer 4.4
Registrska celica za izvedbo dveh logičnih mikrooperacij:

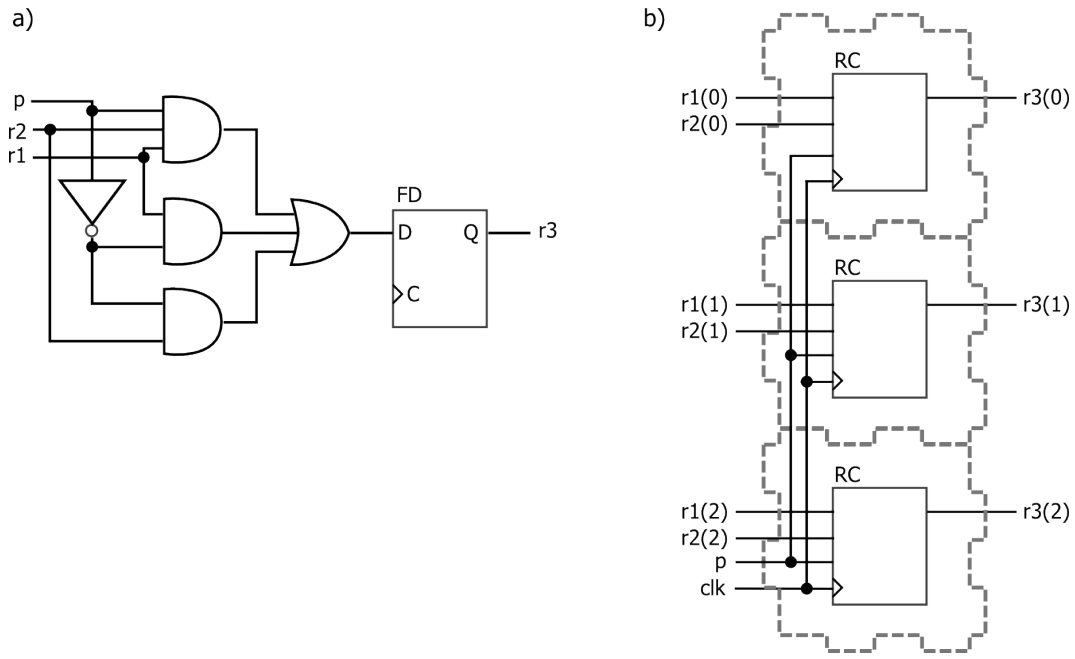
$$\begin{array}{ll} r3 <= r1 \text{ AND } r2 & | \, p = 1 \\ r3 <= r1 \text{ OR } r2 & | \, p = 0 \end{array}$$

Parameter p določa vrsto mikrooperacije; kadar je p=1 se izvede logična AND, pri $p = 0$ pa OR. Za načrtovanje registrske celice privzemimo, da so $r1$, $r2$ in $r3$ enobitni signali. Signal $r3$ ima vrednost 1 v primeru ko so $r1$ in $r2$ in p enaki 1 ali pa ko sta $r1$ ali $r2$ enaka 1 pri $p = 0$, kar zapišemo z enačbo:

$$r3 <= (r1 \text{ AND } r2 \text{ AND } p) \text{ OR } ((r1 \text{ OR } r2) \text{ AND NOT}(p))$$

Enačbo nekoliko preuredimo, da bo primerna za AND - OR izvedbo, kot prikazuje slika 7.1 Z **vzporedno** vezavo registrskih celic dobimo univerzalni gradnik za dve logični mikrooperaciji.

$$r3 <= (r1 \text{ AND } r2 \text{ AND } p) \text{ OR } (r1 \text{ AND NOT}(p)) \text{ OR } (r2 \text{ AND NOT}(p))$$



Slika 7.1: Shema a) registrske celice in b) gradnika za izvedbo operacij AND ali OR.

Primer 4.5
Registrska celica za izvedbo aritmetičnih mikrooperacij z dvema parametroma:

$$\begin{array}{ll} r3 <= r1 & \mid p1 = 0,\ p0 = 0 \text{ (prenesi } r1) \\ r3 <= r1 + r2 & \mid p1 = 0,\ p0 = 1 \text{ (seštej)} \\ r3 <= r1 + (\text{ NOT } r2) & \mid p1 = 1,\ p0 = 0 \\ r3 <= r1 - 1 & \mid p1 = 1,\ p0 = 1 \text{ (zmanjšaj)} \end{array}$$

Osnova aritmetične registrske celice je seštevalnik z dodatno logiko na vhodu. Seštevalnik ima tudi vhodni prenos ci, s katerim lahko naredimo dodatne operacije:

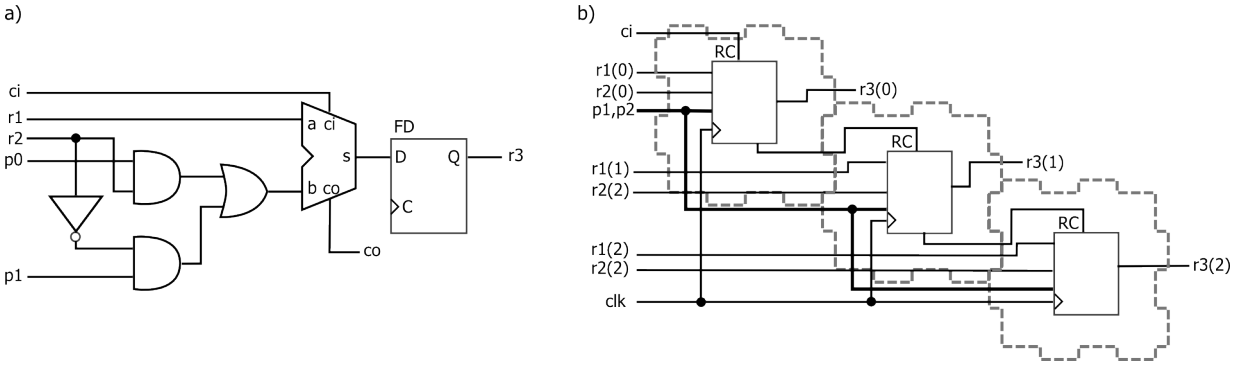
$$\begin{array}{ll} r3 <= r1 + ci & \mid p1 = 0,\ p0 = 0,\ ci = 1 \text{ (povečaj)} \\ r3 <= r1 + r2 + ci & \mid p1 = 0,\ p0 = 1,\ ci = 1 \text{ (seštej s prenosom)} \\ r3 <= r1 + (\text{ NOT } r2) + ci & \mid p1 = 1,\ p0 = 0,\ ci = 1 \text{ (odštej)} \\ r3 <= r1 - 1 + ci & \mid p1 = 1,\ p0 = 1,\ ci = 1 \text{ (prenesi } r1) \end{array}$$

Logično funkcijo za registrsko celico zapišemo v obliki:

$$r3 <= r1 + y + ci$$

$$\begin{array}{ll} y = 0 & \mid p1 = 0,\ p0 = 0 \\ y = r2 & \mid p1 = 0,\ p0 = 1 \\ y = (\text{ NOT } r2) & \mid p1 = 1,\ p0 = 0 \\ y = 1 & \mid p1 = 1,\ p0 = 1 \end{array}$$

Uvedli smo nov signal y, ki predstavlja kombinacijsko logiko z izhodom odvisnim od parametrov.
Z razvijanjem logičnih enačb pridemo do sheme registrske celice, ki jo vidimo na sliki 7.2a. Aritmetične celice povezujemo v večbitno enoto na podoben način kot gradimo večbitni seštevalnik; izhodne prenose (co) nižjih bitov **zaporedno** povežemo z vhodnimi prenosi (ci) višjih bitov, kot vidimo na primeru 7.2b.

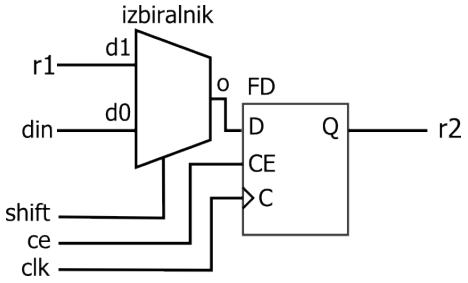


Slika 7.2: Shema a) registrske celice in b) gradnika za izvedbo aritmetičnih operacij.

Primer 4.6
Registrska celica za izvedbo mikrooperacije pomika in prenosa:

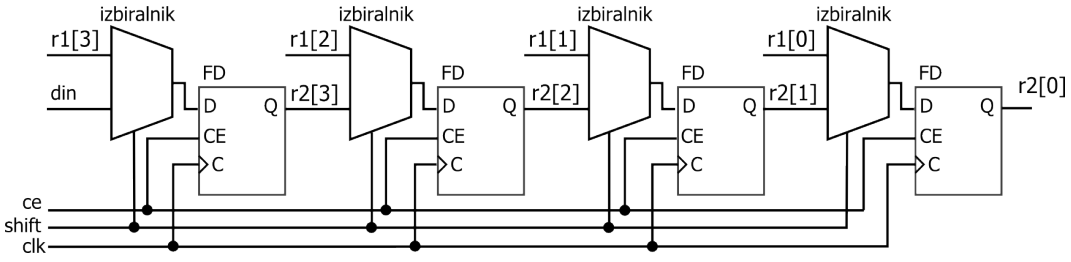
$$\begin{array}{ll} r2 <= din \ : \ r1[3..1] & \mid shift = 0 \text{ and } ce = 1 \\ r2 <= r1 & \mid shift = 1 \text{ and } ce = 1 \end{array}$$

Signal $shift$ določa, ali naj se ob pogoju za nalaganje ($ce = 1$) v register $R2$ naloži pomaknjena vrednost ali pa vrednost iz registra $R1$. Registrska celica je sestavljena iz izbiralnika in flip-flopa D, kot prikazuje slika 7.3.



Slika 7.3: Shema registrske celice izvedbo pomika in prenosa.

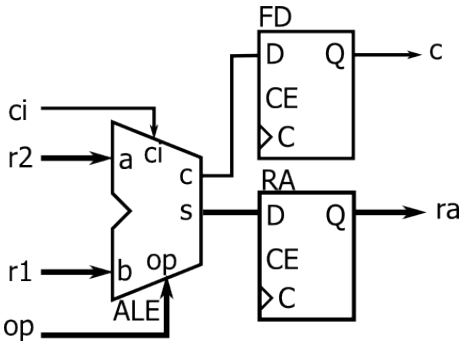
Sestavljen gradnik, ki je prikazan na sliki 7.4 se imenuje paralelno serijski pomikalni register, ker ima vhod za paralelno nalaganje podatkov ($r1$) in zaporedno oz. serijsko pomikanje bitov proti izhodu. Takšen register je osnova za izdelavo digitalnih oddajnikov, ki omogočajo zaporedni prenos podatkov po eni povezovalni liniji.



Slika 7.4: Paralelno serijski pomikalni register.

Tipična ALE ima dva podatkovna vhoda a in b, podatkovni izhod s, kontrolni vhod za izbiro operacij op in izhod za prenos c, kot prikazuje slika 7.5. Enota izvaja različne mikrooperacije, kot so naprimer:

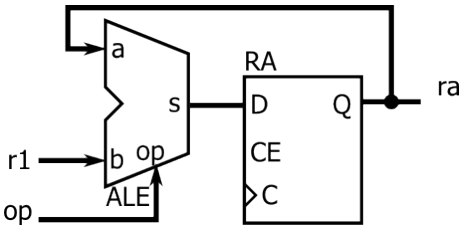
$s <= a$	$c <= 0$	(prenos podatka)
$s <= a + b$	$c <= co$	(seštevanje)
$s <= a + b + ci$	$c <= co$	(seštevanje s prenosom)
$s <= a - b$	$c <= co$	(odštevanje)
$s <= a - (b + ci)$	$c <= co$	(odštevanje z izposojo)
$s <= a$ AND b	$c <= 0$	(logični in)
$s <= a$ OR b	$c <= 0$	(logični ali)
$s <= 0 \ : \ a[3..1]$	$c <= a[0]$	(pomik v desno)
$s <= a[2..0] \ : \ 0$	$c <= a[3]$	(pomik v levo)



Slika 7.5: Aritmetično logična enota z registrom RA in flip-flopom FD za zastavico.

Rezultat operacije ALE shranimo v registru RA, izhodni prenos aritmetičnih operacij (co) pa v flip-flopu FD. Tudi operacije pomika shranijo shranijo bit, ki izpade iz registra v prenos. Izhod za prenos uporabimo pri kombiniranju operacij (npr. naredimo seštevanje s prenosom iz prejšnje vsote) ali pa kot kontrolno zastavico, ki pove da je prišlo do prekoračitve območja vrednosti. V praksi ima ALE še druge zastavice, ki označujejo ali je rezultat negativen, ali je enak 0 ipd.

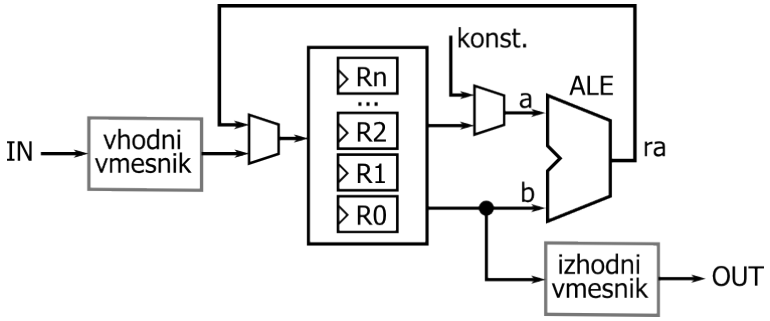
Aritmetično logična enota je lahko v vezavi s povratno zanko, pri kateri se rezultat vedno shrani v enega izmed vhodnih registrov, ki se v tem primeru imenuje akumulator. Vezava z akumulatorjem na sliki 7.6 je preprosta za uporabo, saj zahteva le en podatkovni vhod. Žal pa ni najbolj optimalna, ker moramo pogosto prenašati vrednost iz in v akumulator ter enak algoritem potrebujemo več korakov.



Slika 7.6: Aritmetično logična enota z akumulatorjem.

7.1.2 Centralno procesna enota

Podatkovna pot v centralno procesni enoti vsebuje običajno večje število registrov in aritmetično logično enoto, kot prikazuje slika 7.7. Podatkovna vhoda prideta iz dveh izbranih registrov ali pa enega registra in konstante, rezultat pa se shrani v enega izmed vhodnih registrov. Krmilni vhodi v ALE, registre, izbiralnike in vezave zastavic ALE na sliki zaradi poenostavitve niso prikazani.



Slika 7.7: Podatkovna pot z registri v centralno procesni enoti.

Aritmetično logična enota v tej vezavi izvaja prenašanje podatkov ali konstant v registre ter računanje s podatki in konstantnimi vrednostmi. V centralno procesni enoti se izvajajo ukazi v obliki **strojne kode**, ki določajo vrsto mikrooperacije. Ker je strojna koda za nas težko berljiva jo zapišemo raje v obliki programa v zbirniku (angl. assembler), ki ga prevajalnik prevede v strojno kodo. Naštejmo nekaj primerov operacij z zapisom v zbirniku:

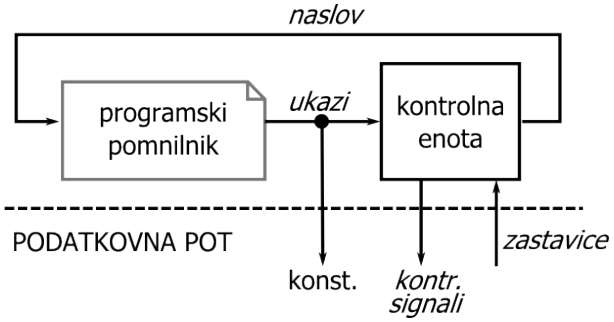
mikrooperacija	**zbirnik**	**pomen**
$ra <= rb$	LOAD ra, rb	prenos podatka med registri
$ra <= k$	LOAD ra, K	prenos konstante K iz pomnilnika
$ra <= ra + rb$	ADD ra, rb	vsota dveh registrov
$ra <= ra + k$	ADD ra, K	vsota registra in konstante
$ra <= ra$ OR rb	OR ra, rb	logični ali med registri
$ra <= 0 \ : \ ra[3..1]$	SR0 ra	pomik registra v desno

Za vgrajene digitalne sisteme so primerne centralno procesne enote z vmesniki. Vmesniki omogočajo prenos zunanjih signalov v CPE in prenos rezultatov algoritma iz CPE. Na sliki 7.7 je prikazana vezava preprostega vhodnega vmesnika, ki iz vhodnih vrat IN prenese podatek v izbrani register ter izhodnega vmesnika, ki prenese vsebino izbranega registra na izhodna vrata OUT.

Naloga vmesnikov je tudi pretvarjanje signalov - npr. analogno-digitalni pretvornik pretvarja analogni signal v digitalne vrednosti, serijski vmesnik pretvarja digitalne besede v zaporedje bitov za prenos po eni povezavi, pulzno-širinski modulator pa pretvarja digitalne vrednosti v impulze različnih širin. Do različnih vmesnikov dostopa CPE tako, da krmili ustrezne identifikacijske signale (ID):

mikrooperacija	**zbirnik**	**pomen**
$ra <= IN$	IN ra, ID	prenos iz vhodne enote ID
$OUT <= rb$	OUT rb, ID	prenos iz registra na izhodno enoto ID

Kontrolni del centralno procesne enote je sestavljen iz programskega pomnilnika in kontrolne enote, kot prikazuje slika 7.8. Kontrolna enota skrbi za branje ukazov oz. strojne kode iz programskega pomnilnika, dekodiranje ukazov in nastavljanje kontrolnih signalov za podatkovno pot mikroprocesorja.



Slika 7.8: Kontrolni del centralno procesne enote.

Programski pomnilnik vgrajenih mikroprocesorjev je narejen tako, da vanj pri programiranju procesorja vpišemo strojno kodo, ki se ohrani tudi po izklopu napajanja. Danes je to običajno pomnilnik vrste FLASH v katerem so programski ukazi in konstante. Za shranjevanje večjih količin podatkov pa imajo mikroprocesorji poleg registrov tudi pomnilnik vrste RAM, ki je priključen podobno kot vhodni in izhodni vmesnik. Miniaturni vgrajeni mikroprocesorji imajo v integriranem vezju obe vrsti pomnilnika, zmogljivejši pa imajo zunanje pomnilnike v različnih izvedbah.

7.1.3 Obdelava podatkov

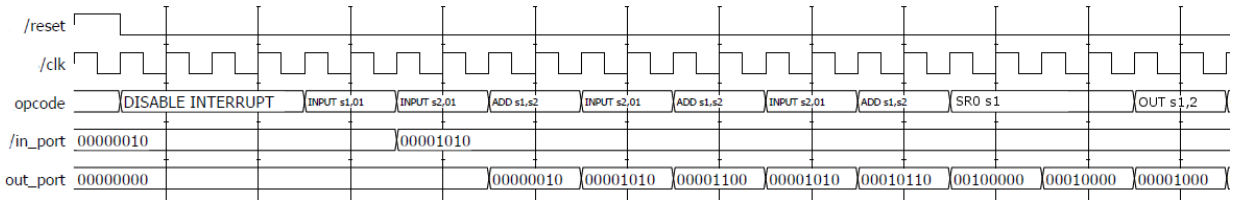
Poglejmo si primer algoritma, ki iz vhodnega vmesnika prebere 4 vrednosti, izračuna njihovo povprečje in ga pošlje na izhod. Program prebere prva dva podatka, naredi njuno vsoto, nato pa prebere in prišteje še tretji in četrti podatek. Povprečje dobimo tako, da skupno vsoto delimo s 4, kar storimo z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

Centralno 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. Takšno izvrševanje ukazov poteka v več korakih, ki jih lahko opazujemo na simulatorju procesorja.

Listing 7.1: Program v zbirniku za izračun povprečja štirih vrednosti

```
start:  IN      s1, VHD   ; beri 1. podatek
        IN      s2, VHD   ; 2. podatek
        ADD     s1, s2    ; vsota
        IN      s2, VHD   ; 3. podatek
        ADD     s1, s2    ; vsota
        IN      s2, VHD   ; 4. podatek
        ADD     s1, s2    ; vsota
        SR0     s1        ; deli z 2
        SR0     s1        ; deli z 2
        OUT     s1, IZH   ; na izhod
```

Slika 7.9 prikazuje simulacijo izvrševanja programa, ki smo ga prevedli za 8-bitni procesor Picoblaze. Iz simulacije lahko razberemo, da je med prvim ukazom za branje podatka in zadnjim ukazom za izpis rezultata 20 urinih ciklov. Procesor Picoblaze potrebuje za vsak ukaz dva urina cikla.



Slika 7.9: Simulacija izvrševanja kode na 8-bitnem procesorju Picoblaze.

Digitalni sistem s centralno obdelavo podatkov je zelo fleksibilen, saj lahko njegovo delovanje hitro spremenimo s spremembo programa, ki ga naložimo v pomnilnik. Sistemi s centralno obdelavo podatkov se zato nahajajo v veliko vgrajenih napravah in predstavljajo pomemben del digitalnih elektronskih sistemov. 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 vedno več urinih ciklov za izvrševanje 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.