Digitalna vezja
Primeri nalog za izpit

Učbenik: Mira Trebar, Osnove logičnih vezij, 2005

Naloge z rešitvami:
Poglavje 3: Booleova algebra
Poglavje 4: Logične funkcije
Poglavje 5: Minimizacija logičnih funkcij
Poglavje 6: Dvo- in več-nivojske logične funkcije (brez Reed-Mullerjeve oblike)
Poglavje 7: Aritmetična vezja
Poglavje 8: Strukturalni gradniki
Poglavja 9: Programabilni logični gradniki
Poglavje 10: Sekvenčna vezja
Poglavje 11: Končni stroj stanj - avtomat

Druge naloge (brez rešitev)

Booleova algebra in logične funkcije

Primer 1:
Za podan izrek zapišite dokaz z uporabo postulatov Booleove algebre.

$$x.x = x$$

Primer 2:
Podana je logična funkcija

$$f(x,y,z) = (x \equiv y) \uparrow (y \oplus z) \vee (\bar{x} \downarrow y.\bar{z})$$

a) Zapišite spodje funkcije z operatorji NOT, AND, OR:

x≡y = _____

x⊕y = _____

x↑y = _____ = _____ (dva zapisa)

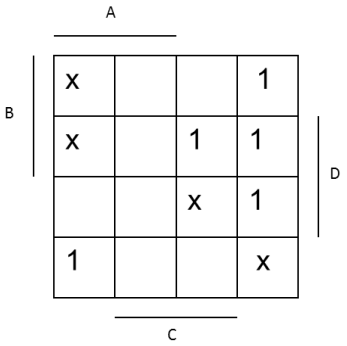
x↓y = _____ = _____ (dva zapisa)

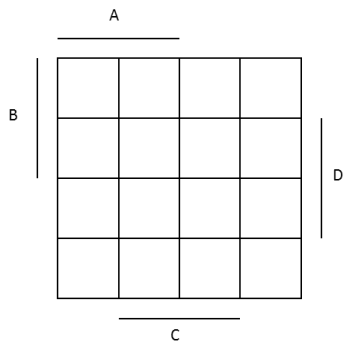
b) Zapišite funkcijo f(x,y,z) v minimalni konjunktivni normalni obliki.

Primer 3:

Podana je logična funkcija F(A,B,C,D) z redundancami. Zapišite:

- MDNO (Minimalno disjunktivno normalno obliko)
- MKNO (Minimalno konjunktivno normalno obliko)
- MNO (določite število operatorjev in število vhodov) za MDNO in MKNO.





Primer 4:

Realizirajte podano logično funkcijo f(x_1,x_2,x_3) v dvonivojski obliki z minimalnim številom NOR operatorjev.

x_1	x_2	x_3	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Primer 5:

Podana je logična funkcija

$$f(A,B,C,D) = A.B.\overline{D} \vee A.B.D \vee \overline{A}.\overline{B}.C \vee \overline{A}.\overline{B}.\overline{C}$$

Naoge:

- Zapišite funkcijo v Veitchev diagram
- Pokažite, da je funkcija linearna
- Zapišite jo z operatorji XOR

Primer 6:

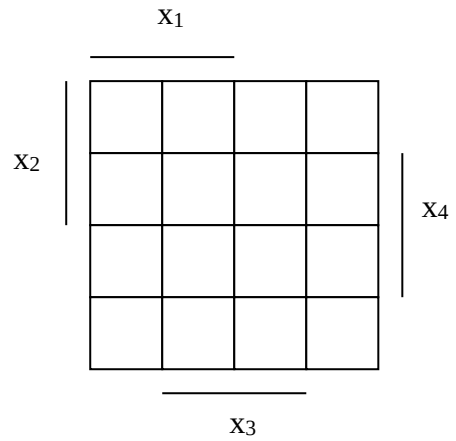
Naloge:

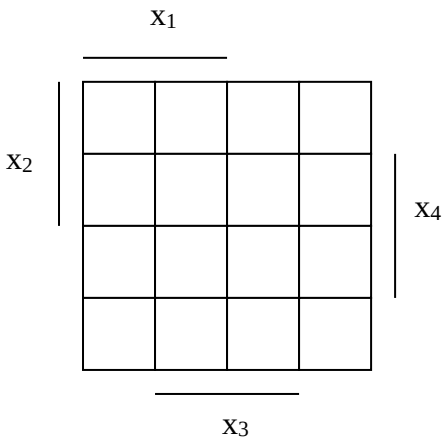
a) Narišite dekoder s tremi krmilnimi vhodi in 8 izhodi (3/8 DEK) in označite vhode in izhode.
b) Zapišite logično funkcijo **f(x,y,z) = x.y V z** v pravilnostno tabelo.
c) Realizirajte zgoraj definirano logično funkcijo z enim dekoderjem (3/8 DEK) in ustreznimi logičnimi vrati (OR, AND, NOT).

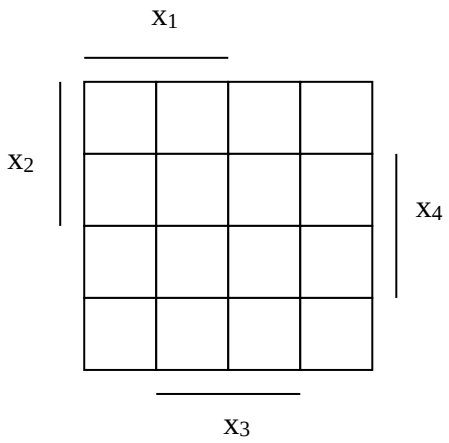
Primer 7:
Realizirajte logično vezje za pretvorbo 4 - bitne binarne kode X= (x_1,x_2,x_3,x_4) v 4 - bitno Grayevo kodo G= (g_1,g_2,g_3,g_4).

1. Zapišite pravilnostno tabelo izhodnih funkcij.
2. Poiščite MDNO (minimalno disjunktivno normalno obliko) izhodnih funkcij g_1, g_2, g_3, g_4.
3. Realizirajte logično vezje z MUX-i (8/1,4/1, 2/1)

x_1	x_2	x_3	x_4	g_1	g_2	g_3	g_4
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	1				
1	0	1	0				
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1				





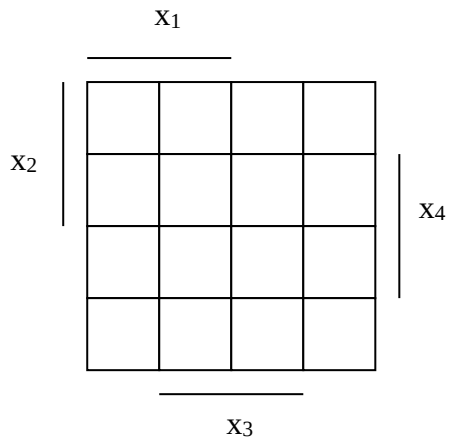


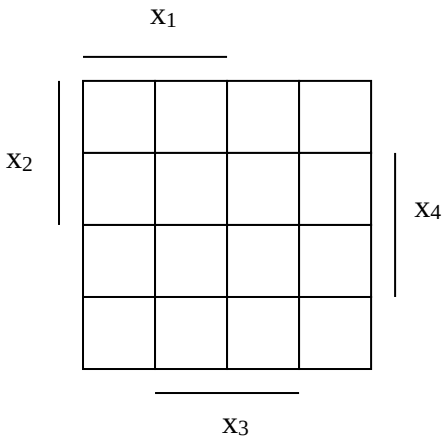
Primer 8:

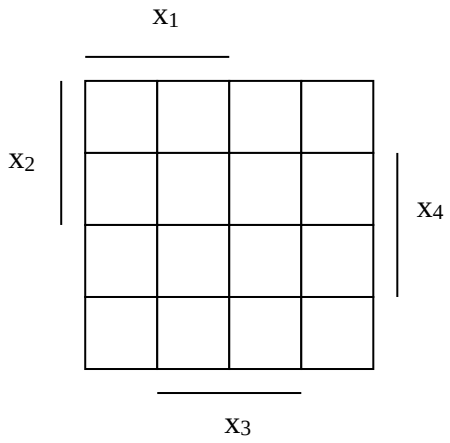
Realizirajte logično vezje za pretvorbo 4 - bitne binarne kode X= (x_1,x_2,x_3,x_4) v 4 - bitno Grayevo kodo G= (g_1,g_2,g_3,g_4).

1. Zapišite pravilnostno tabelo izhodnih funkcij.
2. Poiščite MDNO (minimalno disjunktivno normalno obliko) izhodnih funkcij g_1, g_2, g_3, g_4.
3. Realizirajte logično vezje z MUX-i (8/1,4/1, 2/1)

x_1	x_2	x_3	x_4	g_1	g_2	g_3	g_4
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	1				
1	0	1	0				
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1				

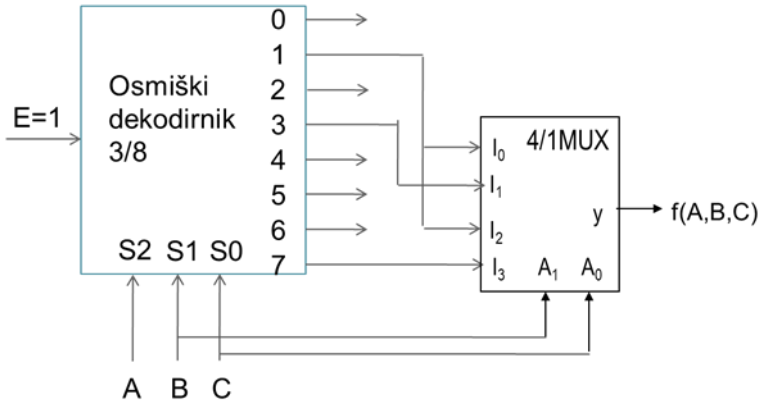






Strukturalni gradniki

Primer 1:
Podano je logično vezje, ki ga sestavljata dekodirnik in multiplekser. Zapišite funkcijo f(A,B,C) v disjunktivni normalni obliki.



Pomnilne celice

Primer 1:

- Narišite povratno vezavo NAND operatorjev in definirajte vhoda R, S in izhoda Q in negirani Q.
- Zapišite binarno aplikacijsko tabelo prehajanja stanj za izhod Q(t+1).
- Zapišite pomnilno enačbo Q(t+1) = ? v minimalni obliki, kjer je prepovedana kombinacija določena z redundancami (x).
- Z uporabo povratne vezave NAND realizirajte sinhronsko T pomnilno celico tako, da vključite dodaten nivo operatorjev NAND z urinim signalom CLK.

Primer 2:
Realizirajte sinhronsko D pomnilno celico z uporabo sinhronske JK pomnilne celice.
Naloge:

- Zapišite binarno aplikacijsko tabelo za D pomnilno celico – Q(t+1)=f(D, Q(t)).
- Določite vzbujevalni funkciji J in K.
- Za realizacijo uporabite 2/MUX.
- Narišite vezje sinhronske D pomnilne celice.

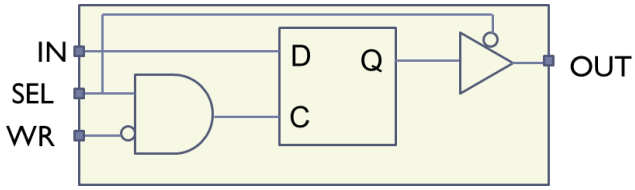
Programabilni gradniki in pomnilnik

Primer 1:
Zapišite v pravilnostno tabelo delovanje dekodirnika 8/3. Zapišite njegovo realizacijo z uporabo programabilnih gradnikov:

- PLA
- PAL
- ROM

Za vsakega od njih zapišite ustrezno obliko logičnih funkcij in narišite njegovo izvedbo s programabilnim poljem AND-OR.

Primer 2:
V statičnem RAMu je uporabljena pomnilna celica D, kjer se izvede vpis podatka na vhodu IN v pomnilno celico, ko je signal C=1 in se vsebina pojavi na izhodu OUT, ko je SEL = 0.



Zapišite pravilnostno tabelo delovanja celice D in preverite ali vezje ustreza podanim zahtevam: a) pri pisanju v celico je izhod OUT neaktiven; b) pri pisanju v celico je izhod OUT=Q . Če vezje pomnilne celice ne ustreza zgornjim zahtevam, zapišite razlog in definirajte novo shemo.

Rešitev:

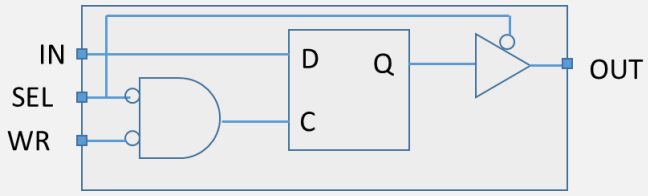
SEL	WR	C
0	0	0
0	1	0
1	0	1
1	1	0

$C = SEL.\overline{WR}$

a) Vezje ustreza zahtevam: SEL=1, se izvede vpis (C=1), izhod OUT je neaktiven
b) Konflikt: SEL =1 -> C=1, se izvede vpis, podatek Q ni na voljo na OUT

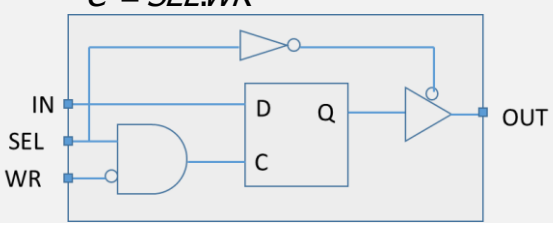
Popravek:
1-Vhod SEL=0 -> negiramo SEL na vhodu v konjunkcijo, dobimo C=1

$C = \overline{SEL}.\overline{WR}$



2- Vhod SEL=0 -> dodamo negator signalu SEL za določanje izhoda OUT

$C = SEL.\overline{WR}$

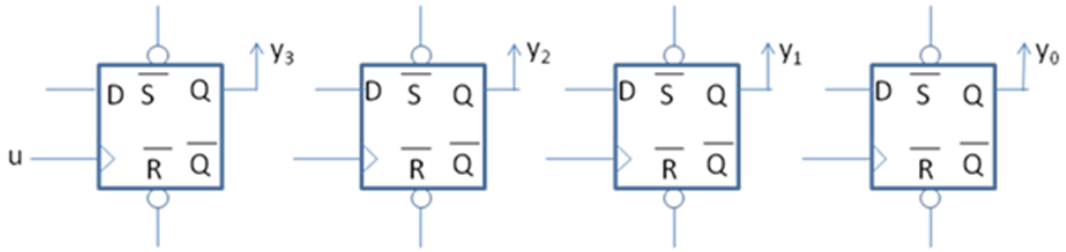


Registri in števci

Primer 1:

Realizirajte 4-bitni sinhronski pomikalni register Y=(y_3 y_2 y_1 y_0) za pomik vsebine za eno mesto desno. Na voljo imate logična vrata NEG. Naloge:

- Z vhodnim signalom Set=0 na asinhronskih vhodih (~R, ~S) nastavite začetno vrednost registra Y=(1111). Asinhronski vhod vpliva na spremembo stanja celice, če je po vrednosti 0.
- V najbolj pomemben bit se pri pomiku vpisuje spremenljivka a=(0,1).
- Definirajte krmilne signale in ustrezno povežite vse vhode pomnilnih celic.



Primer 2:

Realizirajte 2-bitni sinhronski univerzalni register Y=(y_1,y_0), ki opravlja naslednje naloge:

- Vpis: V pomnilne celice se vpiše poljubna vrednost Y(t+1) = X(x_1,x_0).
- Hold: Vrednost pomnilnih celic se ohrani Y(t+1) = Y(t).
- CPL: ciklični pomik levo za eno mesto: $y_1(t+1) = y_0(t)$, $y_0(t+1) = y_1(t)$.
- PD: pomik desno za eno mesto, tako da se v izpraznjeno celico y_1 zapiše konstanta 1.

Za realizacijo imate na voljo: dve sinhronski D pomnilni celici in 2-naslovne MUX-je (označeni morajo biti podatkovni in naslovni vhodi).

Primer 3:

Realizirajte 3- bitni sinhronski števec Q=(q_2,q_1,q_0), ki opravlja dve različni nalogi:

- A=0 - Šteje po modul M=6, s funkcijo odštevanja (DEC) in korakom k=1.
- A=1 - V pomnilne celice se vpiše konstantna vrednost X=(1,0,1).

Za realizacijo imate na voljo sinhronske D pomnilne celice, ki jih krmilimo z:

- Multiplekserji, kjer za vsako pomnilno celico izberite MUX 4/1 ali MUX 8/1
- Logičnimi vrati NOT, AND, OR

Napišite celovito tabelo stanj števca in narišite logično shemo s pomnilnimi celicami D, MUX-i (označenimi podatkovnimi in naslovnimi vhodi) in z vsemi povratnimi vezavami izhodov Q in negirani Q.

Avtomati

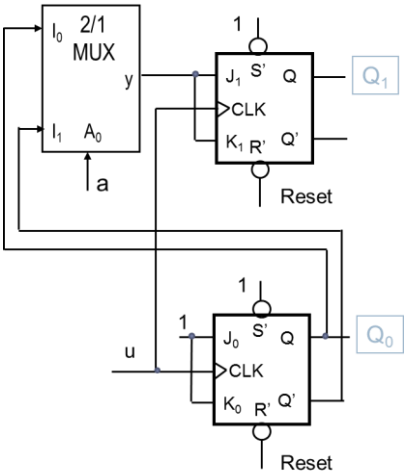
Primer 1:

Definirajte in narišite diagram prehajanja stanj za Mooreov avtomat, ki omogoča razpoznavanje zaporedja X=01010 v poljubnem serijskem nizu ničel (0) in enic (1). Avtomat naj vrne izhod:

- y_1, ko razpozna celotno zaporedje X=01010.
- y_2, če je v tem zaporedju razpoznano delno zaporedje iz X, to je 101.
- y_3 v ostalih primerih.

Primer 2:

Podana ja logična shema Mooreovega avtomata z izhodoma Q_1 in Q_0. Zapišite tabelo prehajanja stanj in diagram prehajanja stanj za podani avtomat



Rešitev:

Enačbi za vhode J in K:

$$J_1 = K_1 = \bar{a}.Q_0 \vee a.\overline{Q_0}$$

$$J_0 = K_0 = 1$$

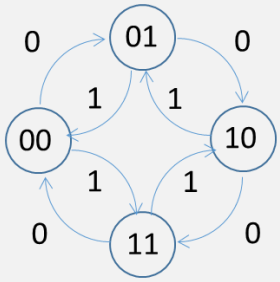
Zapišemo tabelo za J in K za obe celici v odvisnosti od a, Q_1 (t), Q_0(t). Za poznane vhodne vrednosti določimo izhode pomnilnih celic Q_1(t+1) in Q_0(t+1).

a	Q_1(t)	Q_0(t)	J_1=K_1	J_0=K_0	Q_1(t+1)	Q_0 (t+1)
0	0	0	0	1	0	1
0	0	1	1	1	1	0
0	1	0	0	1	1	1
0	1	1	1	1	0	0
1	0	0	1	1	1	1
1	0	1	0	1	0	0
1	1	0	1	1	0	1
1	1	1	0	1	1	0

Napišemo še sekvenco delovanja ali narišemo diagram delovanja sekvenčnega:

a=0: 00→01 →10 →11 →00 →… ali 0→1 →2 →3 →0 → …

a=1: 00→11 →10 →01 →00 →… ali 0→3 →2 →1 →0 →



Procesor MIPS

Primer 1:

Za ukaz MIPS, ki je podan kot:

1 0 0 1 0 0 s s s s s t t t t t i i i i i i i i i i i i i i i i

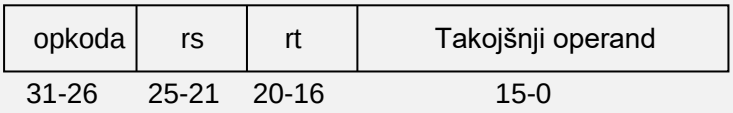
določite:

- tip ukaza, shemo delitve v polja z imeni in številčenjem bitov
- strojno kodo ukaza (heksadecimalni zapis), če je takojšnji operand (i) enak 28 in je uporabljen register R1 za izvorni operand rs (s) in register R2 za izvorni operand rt (t).
- narišite podatkovne poti za izvedbo ukaza,
- zapišite izvedbo ukaza v RTL jeziku.

Rešitev:

- tip ukaza in njegovo razdelitev v polja z oštevilčenjem bitov

Ukaz `lbu` (I tip) : R[rt] = {24'b0, M[R[rs]+SignExtImm](7-0)}

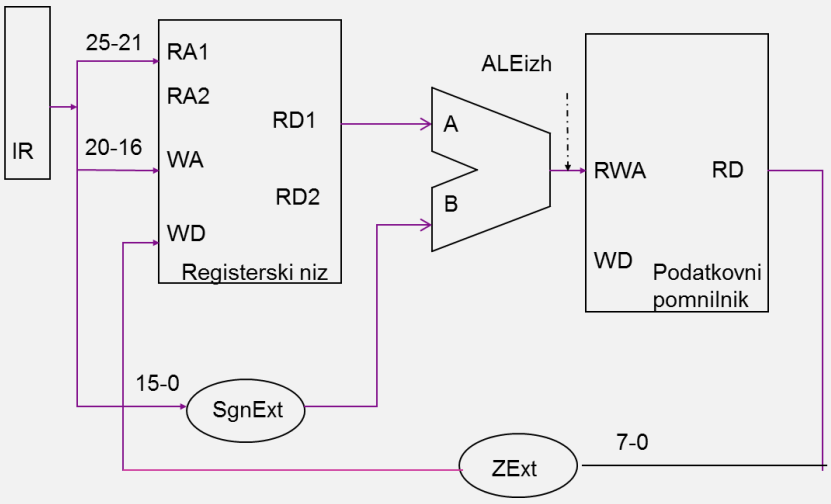


- strojno kodo ukaza (binarni in heksadecimalni zapis), če je takojšnji operand (i) enak 28 in je uporabljen R1 za izvorni operand rs (s) in R2 za izvorni operand rt (t).

100100 sssss ttttt iiiiiiiiiiiiiiii
Bin: 100100 00001 00010 0000000000011100
Hex: 1001 0000 0010 0010 0000 0000 0001 1100 -- > Hex: 9022001c

- narišite podatkovne poti in zapišite operacije izvajanja ukaza v RTL jeziku.

Ukaz `lbu` – unsigned bajt se vpiše v register – raztegnitev ničle v zgornjih 24 bitov.



A(RD1) <= Reg[IR(25-21)]
ALEizh <= A + B(SgnExt(IR(15-0)))
Reg[IR(20-16)] <= ZE(Pom[ALEizh](7-0))

Primer 2:

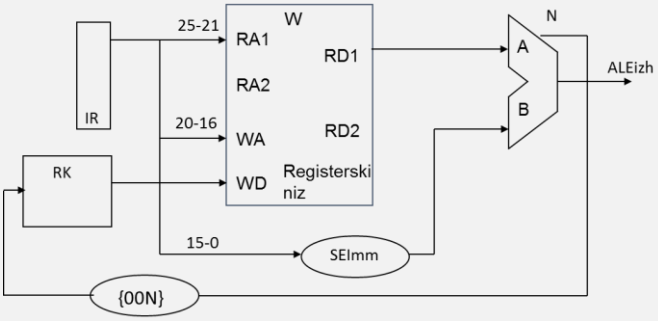
MIPS procesor ima ukaz SLTI -- *Set on less than immediate (signed)*

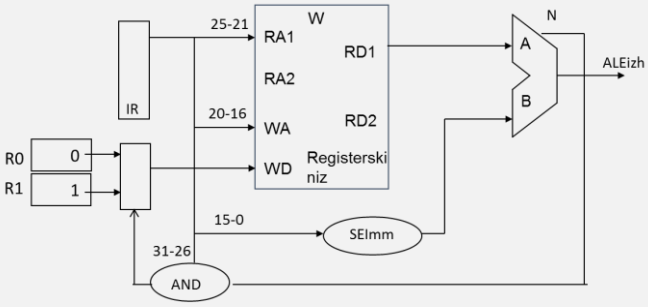
Description:	If $s is less than immediate, $t is set to one. It gets zero otherwise.
Operation:	if $s < imm $t = 1; advance_pc (4); else $t = 0; advance_pc (4);
Syntax:	slti $t, $s, imm
Encoding:	`0010 10ss ssst tttt iiii iiii iiii iiii`

Naloge:

1. Zapišite v kateri tip ukazov sodi SLTI.
2. Narišite podatkovne poti tako, da vključite v izvedbo niz registrov, ki imajo shranjene konstante (0,1,4,8,16,32,64,128).
3. Definirajte zapis v RTL jeziku.
4. Predlagajte rešitev, ki bi omogočala naslavljanje registrov konstant z uporabo OP kode in zastavice N.

Rešitev:

1. I tip ukaza
2. 

3. A(RD1) <= Reg[IR(25-21)]
B <= SignExt[IR(15-0)]
ALEizh <= A - B(SgnExt(IR(15-0)))
Reg[IR(20-16)] <= RK[00N]
4. 

A(RD1) <= Reg[IR(25-21)]
B <= SignExt[IR(15-0)]
ALEizh <= A - B(SgnExt(IR(15-0)))
IF IR[31-26]&N=0: Reg[IR(20-16)] <= [R0]
IF IR[31-26]&N=1: Reg[IR(20-16)] <= [R1]

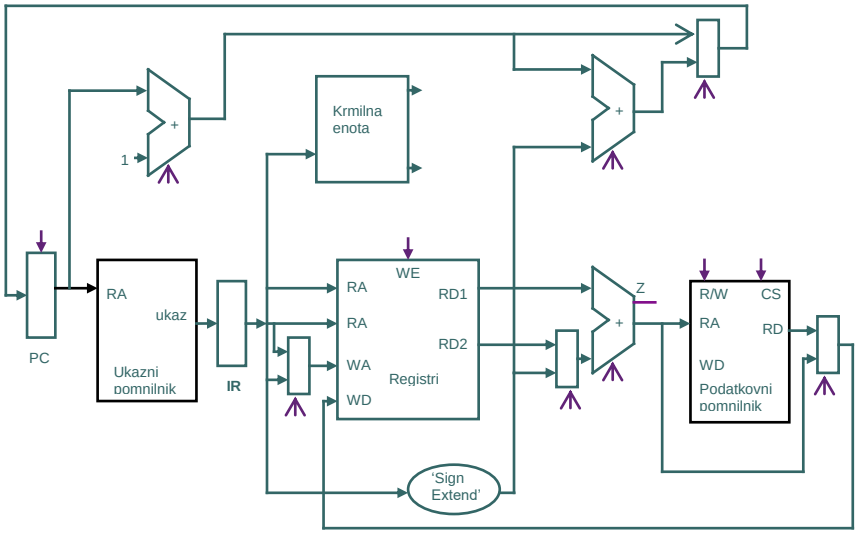
Primer 3:

Za izvajanje naslednjih dveh ukazov:

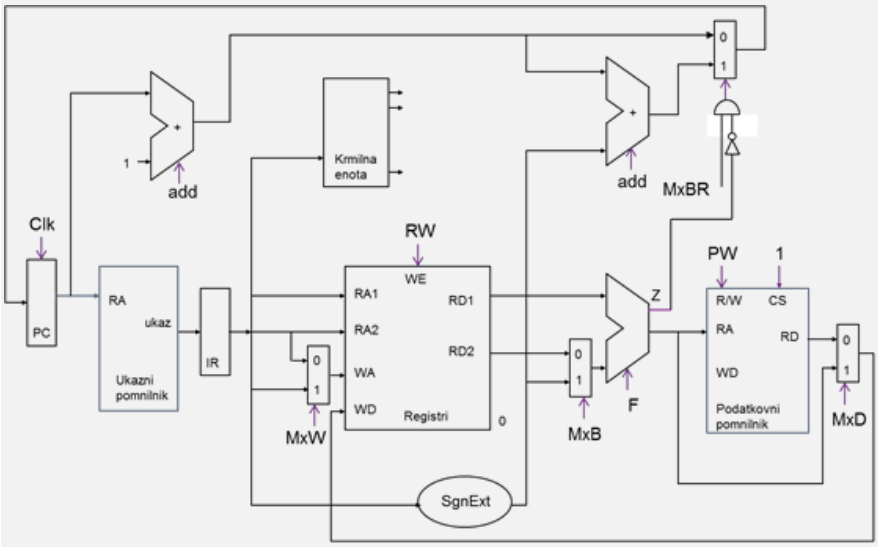
```
lw, bne
```

Določite:

- imena signalov, ki določajo krmiljenje posameznih enot in jih vpišite v spodnjo shemo
- definirajte funkcije ob določenih vrednostih signalov (tako kot smo opisali signale pri procesorju MIPS, npr. MUX1: 0 – RD2, 1- 'SignExtend'(IR[15-0]))
- v tabeli zapišite krmilne signale za oba ukaza (če signal ni uporabljen za krmiljenje, ga postavite kot karkoli (x))
- v shemo dodajte vezje, ki je potrebno za izvedbo ukaza `bne`.



Rešitev:



Krmilni signali:

MxW: (1 signal)	PW: (1signal)
0: IR(20-16)	1 - PomRead
1: IR(15-11)	0 – PomWrite
MxB: (1 signal)	RW: (1signal)
0: RD2	0 – NOP

1: SgnExt(IR(15-16))

1 –RegWrite

MxD: (1 signal)
0: RD
1: ALEIzh

F : ALE funkcija (4 signali)
0: A+B
1: A-B

BR: (1 signal)
0: PC+4
1: BranchAddr

add: seštevalnik za izračun naslova

Ukaz	MxW	MxB	MxD	MxBR	PW	RW	F
lw	0	1	0	0	1	1	0
bne	x	0	x	1	1	0	1