Izvedba Aritmetičko logističke jedinice u simulacijskom softveru Logisim

Kirin, Renato

Sveučilište Jurja Dobrile u Puli

Tehnički Fakultet u Puli

Renato Kirin

IZVEDBA ARITMETIČKO LOGIČKE JEDINICE U SIMULACIJSKOM SOFTVERU LOGISIM

Završni rad

Pula, srpanj, 2023. godine

Sveučilište Jurja Dobrile u Puli

Tehnički Fakultet u Puli

Renato Kirin

IZVEDBA ARITMETIČKO LOGIČKE JEDINICE U SIMULACIJSKOM SOFTVERU LOGISIM

Završni rad

JMBAG: 0303098843, redovni student

Studijski smjer: Računarstvo

Predmet: Digitalna elektronika

Znanstveno područje: Tehničke znanosti

Znanstveno polje: Računarstvo

Znanstvena grana: Arhitektura računalnih sustava

Mentor: izv. prof. dr. sc. Nicoletta Saulig

Komentor: izv. prof. dr. sc. Željka Tomasović

Pula, srpanj, 2023. godine

Sadržaj

1. Uvod

Računala su osmišljena za izvršavanje izračuna i naredbi. Središnja komponenta koja omogućuje izvršavanje tih naredbi je centralna procesorska jedinica, odnosno procesor. Dio procesora koji vrši aritmetičke i logičke operacije je aritmetičko logička jedinica. Prije pokušaja izrade ALU-a i jednostavnog računala potrebno je razumjeti binarni brojevni sustav, sustav u kojemu komponente digitalne elektronike rade. Sljedeći korak je istražiti komponente digitalne elektronike poput logičkih sklopova od kojih se mogu izraditi složeniji sklopovi. Kako bi se ostvarila aritmetička funkcionalnost aritmetičko logičke jedinice, potrebno je istražiti zbrajala. Kada se usvoje potrebne osnove u softveru Logisim moguće je izraditi jednostavan ALU te ga integrirati u jednostavan procesor i napisati jednostavne programe koji demonstriraju njegovu sposobnost. Procesor opisan u ovom radu temelji se na modificiranoj verziji SAP-1 (engl. Simple As Possible 1) arhitekturi. Izvorna SAP-1 arhitektura sadrži aritmetičko logičku jedinicu koja podržava isključivo operacije zbrajanja i oduzimanja. U ovom radu implementirana je aritmetičko logička jedinica proširenih mogućnosti koja dodatno uključuje i logičke operacije.

2. Brojevni sustavi

Za izradu digitalnog računala potreban je brojevni sustav pogodan za rad s elektroničkim komponentama. Dekadski brojevni sustav intuitivan je ljudima, no njegova implementacija u elektroničkim sklopovima nije jednostavna. S druge strane, binarni brojevni sustav se sastoji od samo dvije znamenke, ‘0’ i ‘1’, koje se mogu predstaviti kao prisutnost ili odsutnost napona ili struje. Stoga je binarni brojevni sustav prikladniji za elektroničke sklopove. Heksadekadski brojevni sustav koji se sastoji od 16 znamenaka pruža kompaktniji prikaz velikih binarnih brojeva, čime se ubrzava i pojednostavnjuje njihovo čitanje i manipulacija. BCD (engl. Binary-coded decimal) kod koristi se za prikaz binarnih brojeva u dekadskom obliku na elektroničkim zaslonima.

Broj N u određenom brojevnom sustavu može se prikazati u općem obliku na sljedeći način:

$$N = a_n r^n + a_{n-1} r^{n-1} + \ldots + a_0 r^0$$

Gdje su:

r – baza, broj simbola koje brojevni sustav koristi

a – znamenka s vrijednostima od 0 do r-1

2.1 Dekadski brojevni sustav

Dekadski brojevni sustav je brojevni sustav s bazom 10, što znači da se sustav temelji na 10 znamenki od 0 do 9. Svaki broj moguće je izraziti kao zbroj potencija broja 10 pomnoženih odgovarajućim znamenkama. Primjerice, broj 1234 može se zapisati kao:

$$1234 = 1 \cdot 10^3 + 2 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0$$

2.2 Binarni brojevni sustav

Binarni brojevni sustav je brojevni sustav s bazom 2 čije znamenke mogu poprimiti dvije vrijednosti: 0 i 1. Jedna znamenka binarnog sustava naziva se bit i predstavlja najmanju jedinicu podataka kojom se može manipulirati i pohranjivati. Na primjer, binarni broj 1101 može se zapisati kao:

$1101 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$

$= 8 + 4 + 0 + 1 = 13$

Jedan od načina pretvorbe dekadskog broja u binarni je uzastopno dijeljenje dekadskog broja s 2. Kod svakog koraka dijeljenja, ako se pojavi ostatak, bilježi se znamenka 1, u suprotnom bilježi se 0. Postupak se nastavlja sve dok se u rezultatu dijeljenja ne dobije 0, a dobiveni niz znamenki čini traženi binarni broj. Početni korak dijeljenja daje znamenku najnižeg brojnog mjesta.

Primjer pretvorbe dekadskog broja 25 u binarni:

25 / 2 = 12 i ostatak 1

12 / 2 = 6 i ostatak 0

6 / 2 = 3 i ostatak 0

3 / 2 = 1 i ostatak 1

1 / 2 = 0 i ostatak 1

$25_{10} = 11001_2$

Zbrajanje u binarnom sustavu analogno je dekadskom zbrajanju. Prilikom zbrajanja znamenki istog položaja, ako je rezultat veći od znamenke koja se može prikazati, vrši se prijenos koji se dodaje sljedećoj znamenci ulijevo.

Primjer zbrajanja dva binarna broja:

$$\begin{array}{rl} & 1011_2 \ \ (11_{10}) \\ + & 1001_2 \ \ (9_{10}) \\ \hline & 10100_2 \ \ (20_{10}) \end{array}$$

Oduzimanje u binarnom sustavu također je analogno dekadskom oduzimanju. Prilikom oduzimanja znamenki istog položaja, ako je rezultat manji od znamenke koja se može prikazati, vrši se posudba koja se oduzima od sljedeće znamenke ulijevo.

Primjer oduzimanja dva binarna broja:

$$\begin{array}{rl} & 1010_2 \ \ (10_{10}) \\ - & 101_2 \ \ (5_{10}) \\ \hline & 101_2 \ \ (5_{10}) \end{array}$$

U slučajevima kada su potrebne operacije koje uključuju negativne brojeve, tada je potrebno koristiti drugi komplement. Drugi komplement predstavlja način prikazivanja negativnih brojeva u binarnom sustavu i dobiva se inverzijom svih bitova te dodavanjem jednog bita. U tablici su prikazani dekadski ekvivalenti binarnih vrijednosti u drugom komplementu.

7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8
0111	0110	0101	0100	0011	0010	0001	0000	1111	1110	1101	1100	1011	1010	1001	1000

Tablica 2.1. Dekadski ekvivalenti binarnih vrijednosti u drugom komplementu

Primjerice, pretvaranje 4-bitnog broja 6 u drugi komplement glasi:

$6_{10} = 110_2$

Prvi komplement od 0110 iznosi 1001.

Nakon dodavanja jednog bita dobije se:

$-6_{10} = 1010_2$

Nakon što se željeni broj pretvori u drugi komplement, oduzimanje se provodi binarnim zbrajanjem:

0010_2 (2_{10})

$+ \quad 1010_2$ (-6_{10})

1100_2 (-4_{10})

2.3 Heksadekadski brojevni sustav

Heksadekadski brojevni sustav je brojevni sustav sa 16 znamenaka od kojih su 10 numerički znakovi a preostalih 6 abecedni znakovi. Heksadekadski sustav omogućuje kompaktno prikazivanje velikih binarnih brojeva budući da je konverzija između binarnog i heksadekadskog sustava jednostavna. Četiri bita u binarnom sustavu mogu se prikazati jednom znamenkom u heksadekadskom sustavu, što znači da se jedan bajt može zapisati dvjema heksadekadskim znamenkama. U tablici su prikazane sve znamenke heksadekadskog brojevnog sustava s binarnim ekvivalentima.

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Tablica 2.2. Heksadekadski ekvivalenti binarnih vrijednosti

2.4 BCD kod

Binarno kodirani decimalni broj (engl. Binary-coded decimal, BCD) je metoda prikaza jedne dekadske znamenke u binarnom sustavu. U sljedećoj tablici je prikazan 8421 kod gdje je svaka dekadska znamenka predstavljena odgovarajućom četverobitnom binarnom vrijednošću.

Dekadska znamenka	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Tablica 2.3. Dekadski ekvivalenti BCD vrijednosti

Prikaz dekadskog broja 12 u BCD kodu:

0001 0010

1 2

Prikaz dekadskog broja 935 u BCD kodu:

1001 0011 0101

9 3 5

3. Komponente digitalne logike

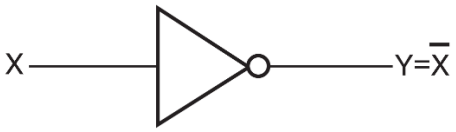
Za izradu digitalnog računala potrebne su komponente koje omogućuju izvođenje logičkih operacija, upravljanje tokom informacija i pohranjivanje podataka. Komponente kao što su logički sklopovi, multipleksori, registri i brojila su neophodni za rad digitalnog računala.

3.1 Logički sklopovi

Logički sklopovi su najosnovniji sklopovi digitalne logike i pomoću njih se mogu izraditi složeniji sklopovi. Sastoje se od najmanje jednog ulaza, te ovisno o kombinaciji ulaza vraćaju izlaz. Za prikaz svih mogućih kombinacija ulaznih vrijednost i odgovarajućih izlaznih vrijednosti koristi se tablica stanja.

3.1.1 Logički sklop NE

Logički sklop NE (engl. NOT gate) također poznat kao logička negacija sadrži jedan ulaz i jedan izlaz. Izlaz je uvijek suprotan vrijednosti ulaza. Simbol i tablica stanja prikazani su na sljedećoj slici:

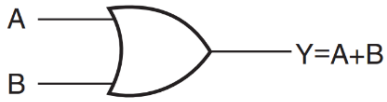


X	Y
0	1
1	0

Slika 3.1. Logički sklop NE: a) Simbol, b) Tablica stanja

3.1.2 Logički sklop ILI

Logički sklop ILI (engl. OR gate) također poznat kao logički zbroj. Operacija ILI se može izraziti kao Y = A + B. Y će biti istinit ako je barem jedan ulaz istinit, odnosno ako su A ili B ili oboje istiniti. Logički sklop ILI može imati više od dva ulaza. Na primjer, izlaz ILI sklopa s tri ulaza A, B, C može se napisati kao Y = A + B + C, što znači da je Y istinit ako je bilo koji od ulaza A, B ili C istinit. Simbol i tablica stanja prikazani su na sljedećoj slici:

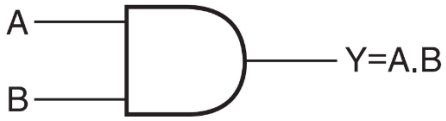


A	B	Y
0	0	0
1	0	1
0	1	1
1	1	1

Slika 3.2. Logički sklop ILI: a) Simbol, b) Tablica stanja

3.1.3 Logički sklop I

Logički sklop I (engl. AND gate) također poznat kao logički umnožak. Operaciju I se može izraziti kao Y = A • B. Y će biti istinit ako su svi ulazi istiniti, odnosno ako su A i B istiniti. Logički sklop I može imati više od dva ulaza. Na primjer, izlaz I sklopa s tri ulaza A, B i C može se napisati kao Y = A • B • C, što znači da je Y istinit ako su svi ulazi A, B i C istiniti. Simbol i tablica stanja prikazani su na sljedećoj slici:

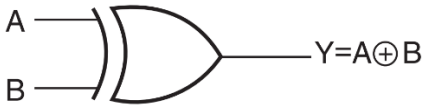


A	B	Y
0	0	0
1	0	0
0	1	0
1	1	1

Slika 3.3. Logički sklop I: a) Simbol, b) Tablica stanja

3.1.4 Logički sklop Isključivo-ILI

Logički sklop Isključivo-ILI (engl. Exclusive OR, XOR) se može dobiti kombiniranjem I, ILI i NE sklopovima. Izlaz XOR sklopa je istinit samo ako je točno jedan od ulaza istinit. Operacija XOR može se izraziti kao Y = A $\oplus$ B. XOR sklop može imati više od dva ulaza. Na primjer, izlaz XOR sklopa s tri ulaza A, B i C može se napisati kao Y = A $\oplus$ B $\oplus$ C, što znači da je Y istinit ako je jedan i samo jedan od ulaza A, B i C istinit. Ako su svi ulazi neistiniti ili je više od jednog ulaza istinito, tada će Y biti neistinit. Jedna od primjena XOR sklopa je binarno zbrajanje ili usporedba dvije binarne vrijednosti radi utvrđivanja nejednakosti. Simbol i tablica stanja prikazani su na sljedećoj slici:

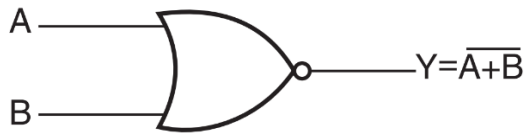


A	B	Y
0	0	0
1	0	1
0	1	1
1	1	0

Slika 3.4. Logički sklop Isključivo-ILI: a) Simbol, b) Tablica stanja

3.1.5 Logički sklop NILI

Logički sklop NILI (engl. NOR) može se dobiti korištenjem ILI sklopa te NE sklopa na njegovom izlazu. Izlaz NILI sklopa bit će istinit samo kada su svi ulazi neistiniti, u suprotnom izlaz će biti neistinit. Jedna od značajki NILI sklopa je da se bilo koji logički izraz može implementirati samo s kombinacijom NILI sklopova. Simbol i tablica stanja prikazani su na sljedećoj slici:

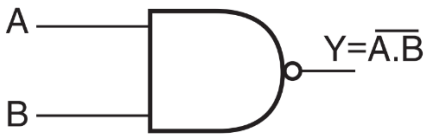


A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Slika 3.5. Logički sklop NILI: a) Simbol, b) Tablica stanja

3.1.6 Logički sklop NI

Logički sklop NI (engl. NAND) može se dobiti korištenjem I sklopa te NE sklopa na njegovom izlazu. Izlaz NI sklopa bit će neistinit samo kada su svi ulazi istiniti, u suprotnom izlaz će biti istinit. NI sklop kao NILI sklop također ima mogućnost implementacije bilo kojeg logičkog izraza s kombinacijom NI sklopova. Simbol i tablica stanja prikazani su na sljedećoj slici:

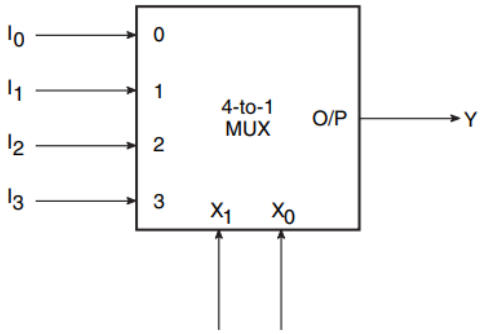


A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Slika 3.6. Logički sklop NI: a) Simbol, b) Tablica stanja

3.2 Multipleksor i demultipleksor

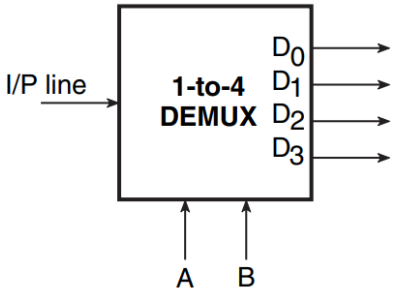
Multipleksor je komponenta digitalne logike koja od nekoliko ulaznih signala odabire jedan ulazni signal i šalje ga na izlaz. Odabir se vrši pomoću kontrolnih signala. Ako postoji n linija za odabir, tada je maksimalni broj mogućih ulaznih linija 2^n. Takav multipleksor naziva se 2^n-na-1 multipleksor. Simbol i tablica stanja 4-na-1 multipleksora prikazani su na sljedećoj slici:



X_1	X_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Slika 3.7. Multipleksor: a) Simbol, b) Tablica stanja

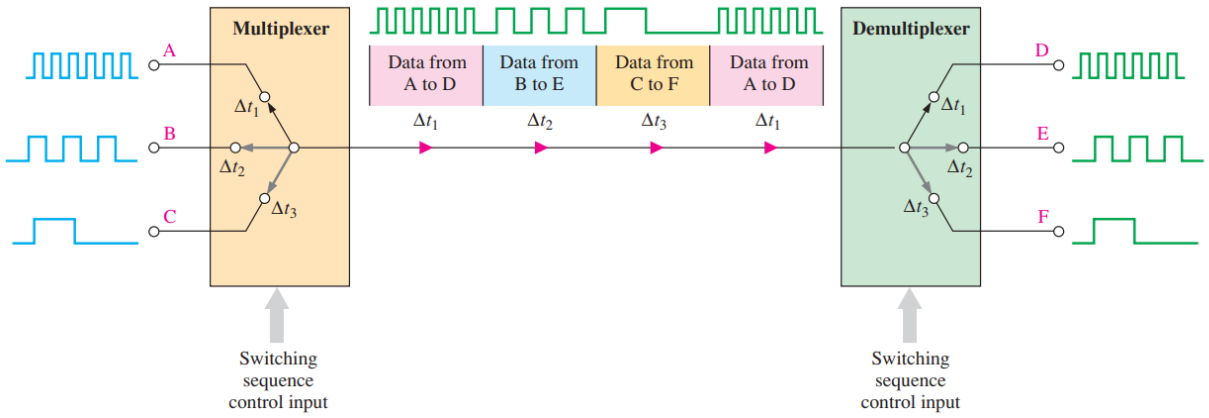
Demultipleksor je komponenta digitalne logike koja sadrži jedan ulaz te s odabirom dobivenim pomoću kontrolnih signala vraća izlaz na jedan od nekoliko izlaza. Simbol i tablica stanja 1-na-4 demultipleksora prikazani su na sljedećoj slici:



I/P	Select		O/P			
	A	B	D_0	D_1	D_2	D_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

Slika 3.8. Demultipleksor: a) Simbol, b) Tablica stanja

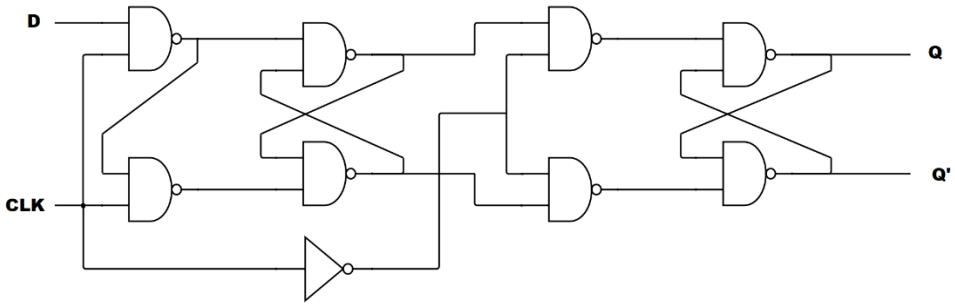
Multipleksor i demultipleksor mogu se zajedno koristiti za multipleksiranje s vremenskom podjelom, prijenos podataka iz nekoliko izvora jednim kanalom te redistribuciju na nekoliko odredišta. Vrijeme je podijeljeno između nekoliko izvora i odredišta pri čemu svaki ima svoj red za slanje i primanje podataka. Kako je na sljedećoj slici prikazano, tijekom prvog vremenskog intervala, podaci s ulaza A prenose se na izlaz D. Tijekom drugog vremenskog intervala, podaci s ulaza B prenose se na izlaz E. Tijekom trećeg vremenskog intervala, podaci s ulaza C prenose se na izlaz F. Nakon toga se niz ponavlja.



Slika 3.9. Multipleksiranje s vremenskom podjelom

3.3 Registri

Registri su sklopovi koji služe za privremeno pohranjivanje binarnih podataka. Registar sadrži četiri glavna ulaza: podatke (D), takt (CLK), load i reset te jedan izlaz na kojem je spremljena vrijednost. Registri se sastoje od više bridom okidanih bistabila od kojih se jedan može ostvariti na sljedeći način prikazan na slici:



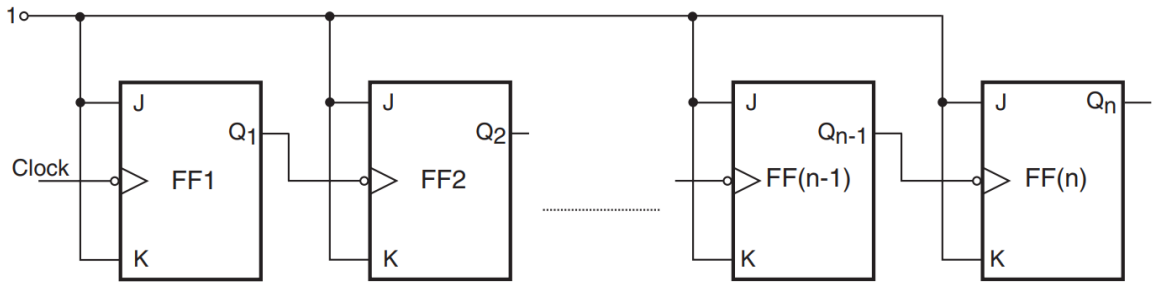
D	CLK	Q
0	↑	0
1	↑	1

Slika 3.10. Bridom okidani D bistabil: a) Simbol, b) Tablica stanja

Sastoji se od D bistabila koji služi kao glavni sklop i SR bistabila koji služi kao sporedni sklop. Glavni sklop sadrži ulaz za podatak, te kada je signal takta u visokom stanju podatak se spremi dok je sporedni sklop isključen. Kada signal takta prelazi u nisko stanje, sporedni sklop se aktivira, a podatak je dostupan na izlazu. Izlaz bistabila javlja se tijekom završetka signala takta što omogućuje da se izlaz koristi za komponentu čiji vlastiti izlaz vodi u ulaz bridom okidanog bistabila.

3.4 Brojilo

Binarno brojilo predstavlja posebnu vrstu registra koji povećava ili smanjuje binarnu vrijednost na signal takta. Najjednostavniji oblik binarnog brojila je ripple-carry brojilo, poznato i pod nazivom asinkrono brojilo. U ripple-carry brojilu izlaz svakog bistabila spojen je na ulaz takta sljedećeg bistabila. Kada prvi bistabil promjeni stanje, on pokreće drugi bistabil da promjeni stanje, i tako dalje. Na slici je prikazano n-bitno asinkrono brojilo.



Slika 3.11. Asinkrono brojilo

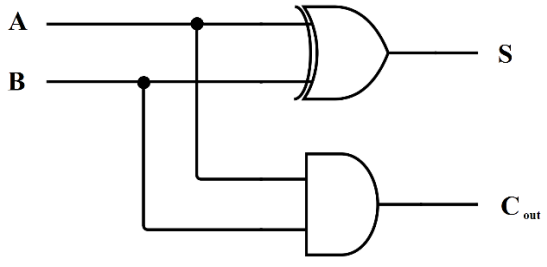
4. Zbrajala

Zbrajala su neophodna komponenta aritmetičko logičke jedinice. Zbrajala su nužna za provođenje operacija poput zbrajanja, brojanja, uspoređivanja i adresiranja. Postoje različite inačice zbrajala koje se razlikuju po broju komponenti potrebnih za njihov rad. Jednostavnija zbrajala s manjim brojem komponenti sporija su u izvođenju računskih operacija u odnosu na složenija zbrajala s većim brojem komponenti.

4.1 Ripple-carry zbrajalo

Ripple Carry zbrajalo sastoji se od niza povezanih potpunih zbrajala, gdje je izlaz za prijenos svakog potpunog zbrajala povezan s prijenosnim ulazom sljedećeg potpunog zbrajala.

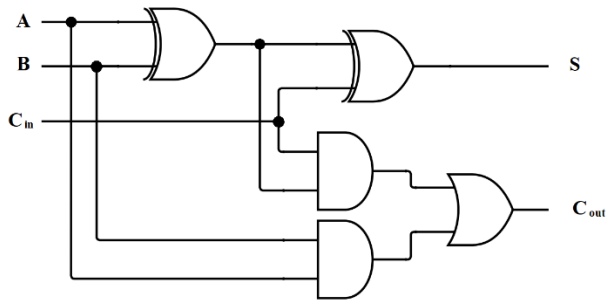
Poluzbrajalo je sklop koji služi za izvršavanje operacije binarnog zbrajanja između dva bita. Ovaj sklop sadrži dva ulaza koji predstavljaju dva bita čije se vrijednosti zbrajaju te dva izlaza – jedan predstavlja zbroj ulaznih bitova, dok drugi predstavlja prijenosni bit. Sastoji se od jednog isključivo-ILI logičkog sklopa koji vrši operaciju zbrajanja te jednog I logičkog sklopa koji vrši operaciju prijenosnog bita. Izvedba sklopa i tablica stanja prikazani su na sljedećoj slici:



A	B	Cout	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Slika 4.1. Poluzbrajalo: a) Izvedba, b) Tablica stanja

Potpuno zbrajalo sadrži dodatan ulaz za prijenos. Sastoji se od dva isključivo-ILI logička sklopa koja vrše operaciju zbrajanja nad 3 ulaza te dva I logička sklopa i jednog ILI logičkog sklopa koji vraćaju izlaz prijenosa. Izvedba sklopa i tablica stanja prikazani su na sljedećoj slici:



A	B	Cin	Cout	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Slika 4.2. Potpuno zbrajalo: a) Izvedba, b) Tablica stanja

Kašnjenje prijenosa je primarni čimbenik koji ograničava brzinu ripple-carry zbrajala, do ovog kašnjenja dolazi jer se prijenosni bit mora propagirati kroz svako potpuno zbrajalo sekvencijalno, s time se ukupno kašnjenje povećava linearno s brojem bitova u zbrajalu.

4.2 Carry look-ahead zbrajalo

Budući da je kašnjenje propagacije prijenosa glavni ograničavajući faktor brzine ripple-carry zbrajala, carry look-ahead zbrajala koristi sklop koji paralelno unaprijed izračunava prijenos za sve ulaze prijenosa.

Kako bi se koncept unaprijednog izračuna prijenosa objasnio, potrebno je definirati dvije nove varijable, P_i (engl. carry propagate) i G_i (engl. carry generate). [Maini, 2007]:

$G_i = A_i \bullet B_i$

$P_i = A_i \oplus B_i$

$S_i = P_i \oplus C_i$

$C_{i+1} = P_i C_i + G_i$

Za četvernobitno binarno zbrajalo dobiju se sljedeći izrazi [Maini, 2007]:

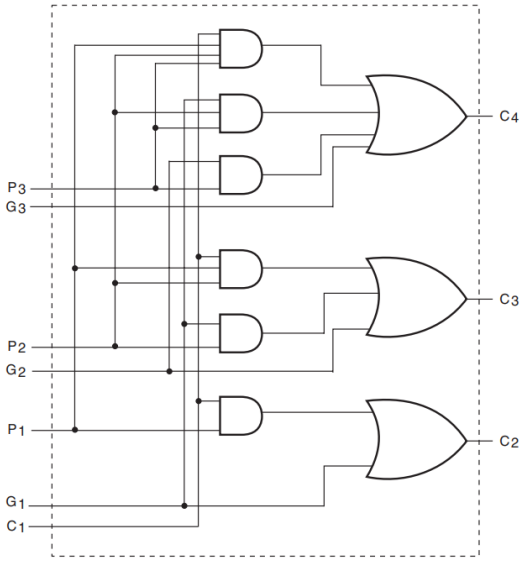
$C_2 = G_1 + P_1C_1$

$C_3 = G_2 + P_2G_1 + P_1P_2C_1$

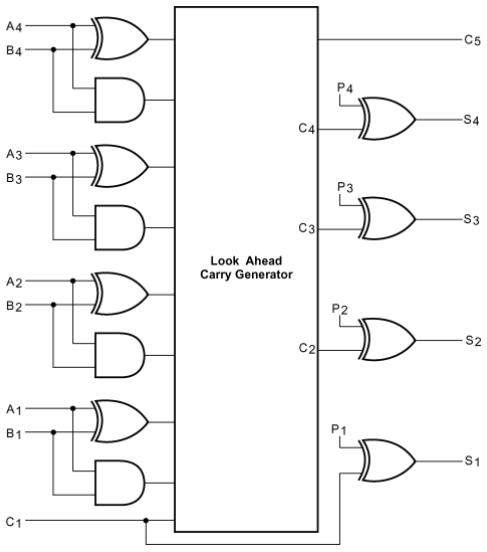
$C_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_1P_2P_3C_1$

Iz izraza za C_2, C_3 i C_4 očito je kako C_4 ne mora čekati širenje C_3 i C_2.

Sukladno tome, C_3 ne treba čekati širenje C_2. Hardverska implementacija navedenih izraza daje svojevrsni look-ahead carry generator. Izvedba sklopa generatora prijenosa i carry look-ahead zbrajala prikazani su na sljedećim slikama:



Slika 4.3. Generator prijenosa

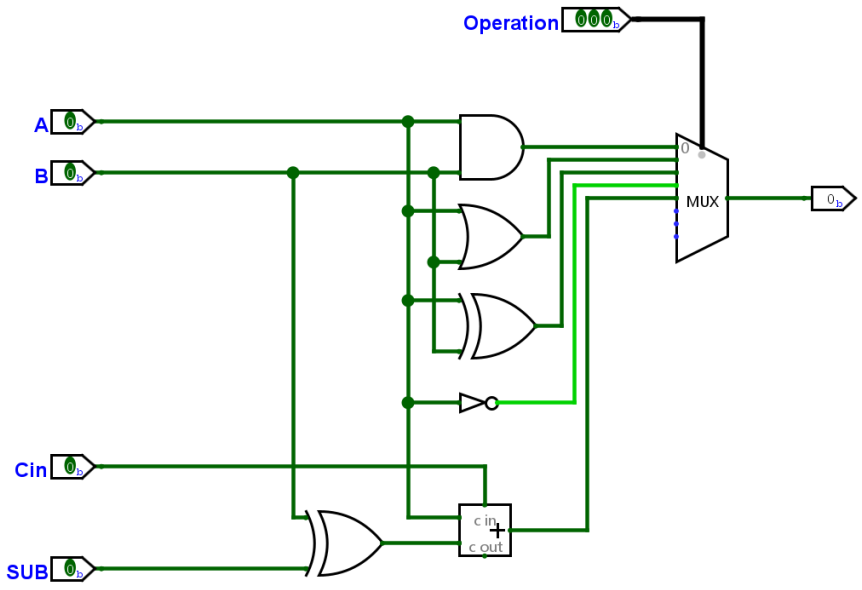


Slika 4.4. Carry look-ahead zbrajalo

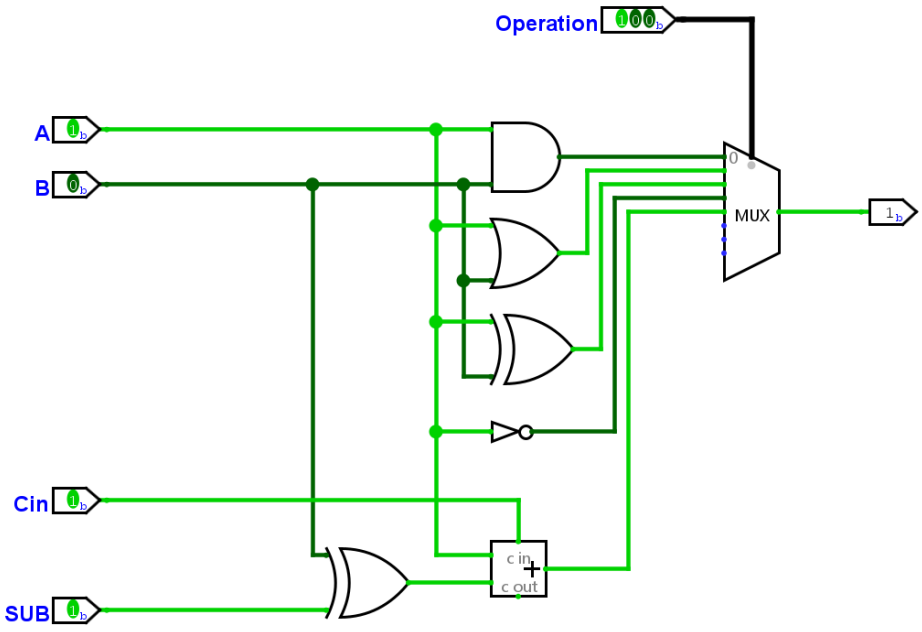
5. Izvedba jednostavne Aritmetičko logičke jedinice

Aritmetičko logička jedinica (engl. ALU) sastavni je dio centralne procesorske jedinice (engl. CPU). Zadužena je za izvođenje svih aritmetičkih i logičkih operacija nad podacima. Ostale komponente CPU-a isključivo služe za pružanje ulaznih podataka ALU-u. ALU izvodi logičke i aritmetičke operacije na dva ulazna operandna podatka, te na temelju odabrane operacije generira izlazni rezultat.

Korištenjem simulacijskog softvera Logisim u ovom primjeru je izveden 1 bit ALU-a koji podržava logičke operacije I, ILI, ekskluzivno ILI i NE te aritmetičke operacije zbrajanja i oduzimanja. Za provedbu operacije oduzimanja potrebno je izvršiti inverziju svih bitova te dodavanje broja 1. Ovaj postupak se ostvaruje ekskluzivno ILI logičkim sklopom koji invertira bitove jednog ulaza, ako je drugi ulaz istinit. Dodavanje broja jedan se ostvaruje C_{in} (engl. Carry-in) ulaza. Izlazi logičkih sklopova i zbrajala su spojeni na multipleksor te se unošenjem podatka u kontrolni ulaz vraća odgovarajući izlaz. Izvedba sklopa 1-bitnog ALU-a u Logisimu i odabir operacije oduzimanja prikazani su na sljedećim slikama:



Slika 5.1. 1-bitni ALU



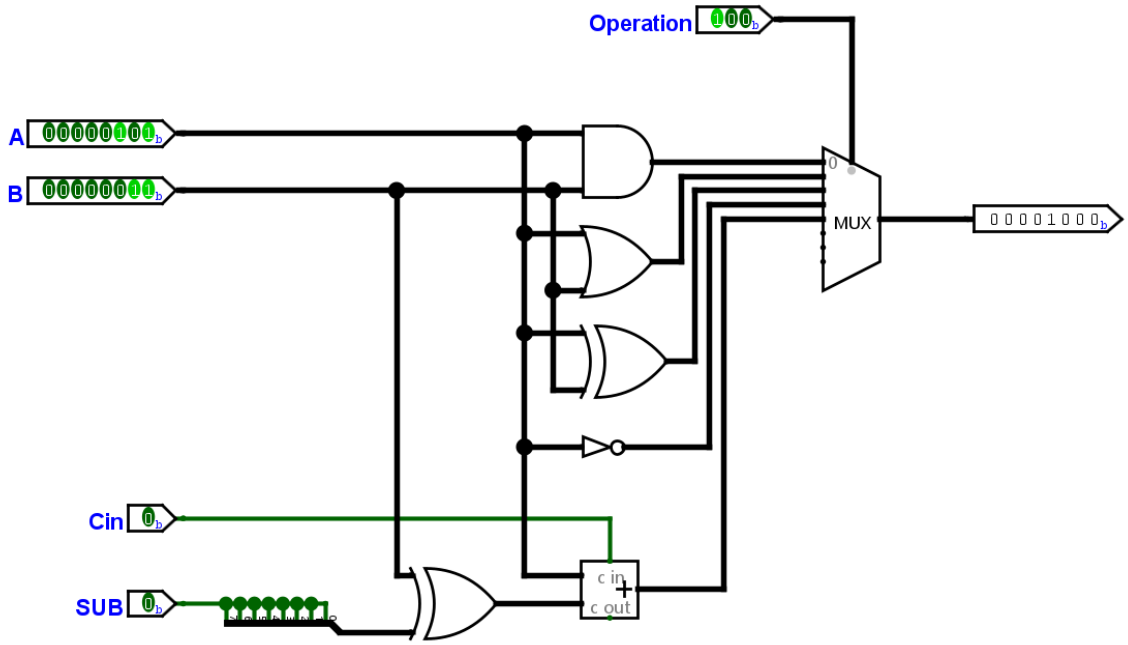
Slika 5.2. Operacija binarnog oduzimanja u 1-bitnom ALU

U tablici su prikazane operacije koje je navedeni ALU u mogućnosti izvoditi:

Kod	Operacija
000	I
001	ILI
010	Isključivo ILI
011	NE
100	Zbrajanje/oduzimanje

Tablica 5.1. Operacije ALU-a

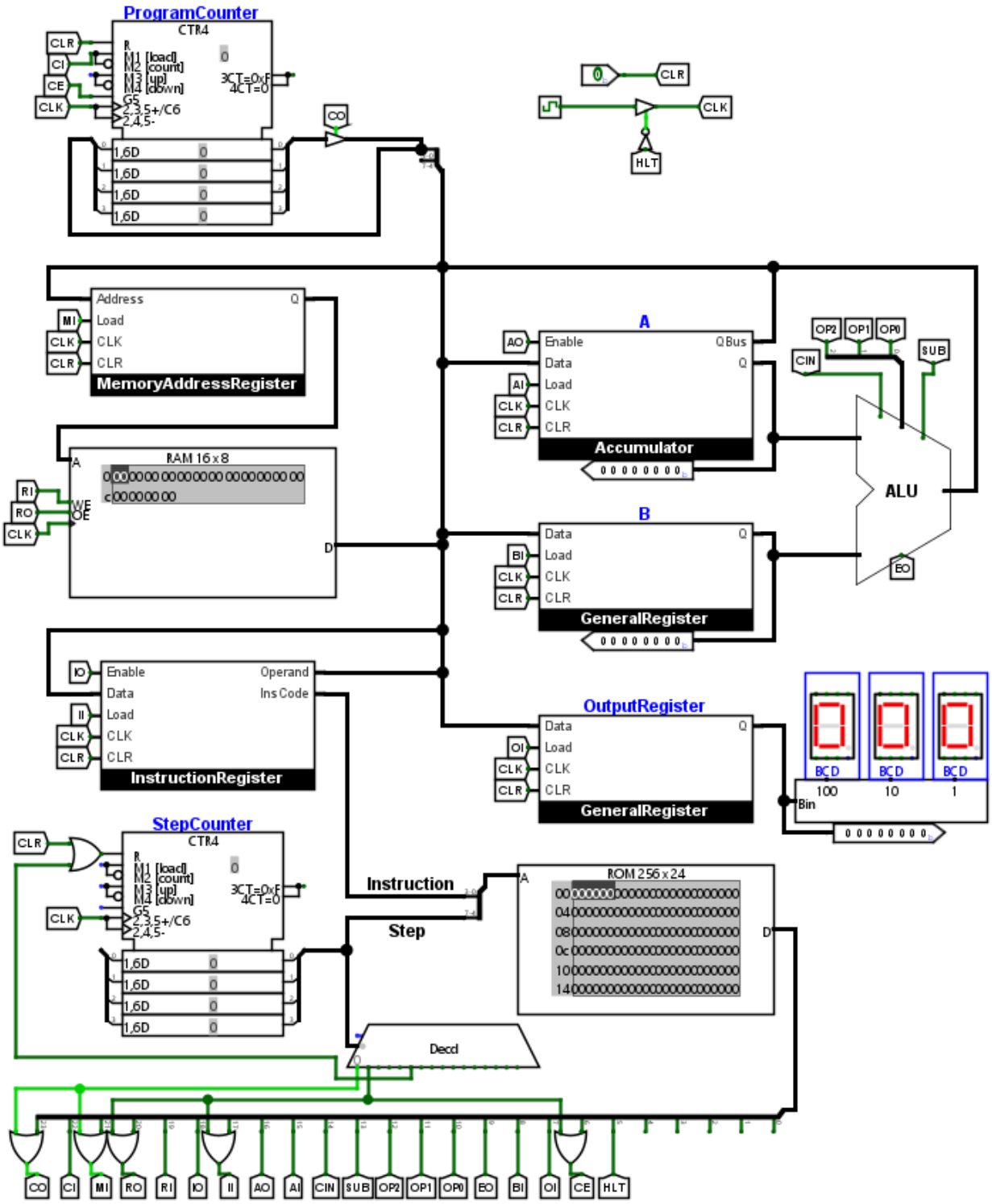
Logisim omogućuje konfiguraciju širine ulaza i izlaza logičkih sklopova, izraženu u bitovima. U ovom primjeru na slici je implementiran ALU s osam ulaznih bitova.



Slika 5.3. 8-bitna izvedba ALU-a

6. Izvedba i programiranje jednostavnog procesora

Procesor predstavljen u ovom radu temelji se na modificiranoj verziji SAP-1 (engl. Simple As Possible 1) procesora koji pomaže pri usvajanju osnovnih koncepata funkcioniranja računalnog sustava. Procesor se temelji na von Neumannovoj arhitekturi koja podrazumijeva korištenje jedinstvene glavne memorije za pohranu programskih instrukcija i podataka. Također, procesor koristi sabirničku arhitekturu 8-bitne širine koja služi za prijenos podataka i instrukcija između svih komponenata. Sadrži ključne komponente poput radne memorije, izlaza, registara, programskog brojača te kontrolne jedinice. Procesor posjeduje relativno ograničen skup instrukcija od svega 11 naredbi, što je dovoljno za izvršavanje jednostavnih programa aritmetičko logičkih operacija te prijenosa podataka između memorijskih lokacija. Dodatno, implementiran je dekadski displej kako bi se olakšao rad s aritmetičkim rezultatima. Na slici je prikazan navedeni procesor ostvaren u Logisimu.



Slika 6.1. Procesor temeljen na SAP-1

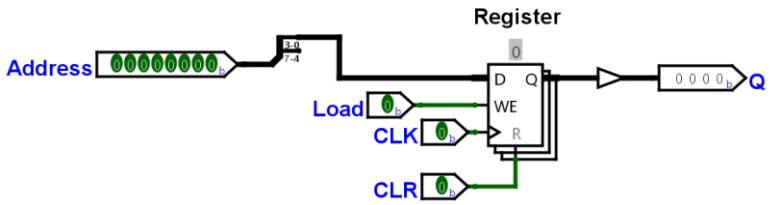
6.1 Izvedba i princip rada

Programski brojač

Programski brojač sadrži memorijsku adresu sljedeće instrukcije predviđene za izvršavanje. Programer ga može postaviti na proizvoljnu memorijsku adresu, što omogućuje JUMP naredbu. Spojen je na sabirnicu te na odgovarajuće kontrolne signale poveća vrijednost za jedan, šalje izlaz na sabirnicu ili učita željenu vrijednost sa sabirnice.

Registar memorijske adrese

Registar memorijske adrese (engl. Memory Address Register, MAR) pohranjuje memorijsku adresu instrukcije koja će se pročitati ili zapisati u memoriju. Prema prikazanoj slici, za izvedbu koristi se 4-bitni registar koji prihvaća ulaz s 8-bitne sabirnice.

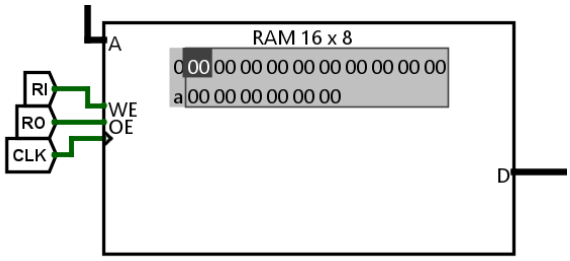


Slika 6.2. Izvedba registra memorijske adrese u Logisim-u

Glavna memorija (RAM)

RAM (engl. Random Access Memory) je vrsta memorije koja se koristi za pohranjivanje instrukcija i podataka. Prema prikazanoj slici, RAM se u ovom računalu sastoji od 16 memorijskih lokacija od 8 bitova, što omogućuje pohranjivanje do 16 bajtova podataka.

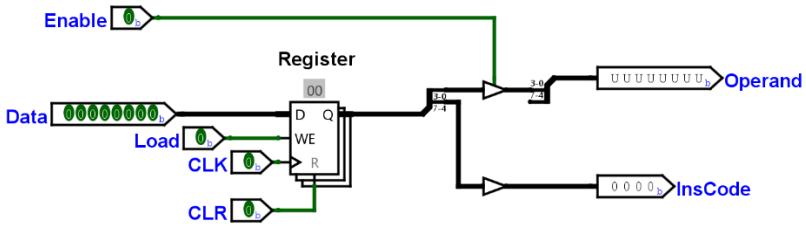
Budući da se RAM sastoji od 16 memorijskih lokacija, te se lokacije mogu adresirati jedinstvenom 4-bitnom adresom.



Slika 6.3. RAM u Logisim-u

Registar instrukcija

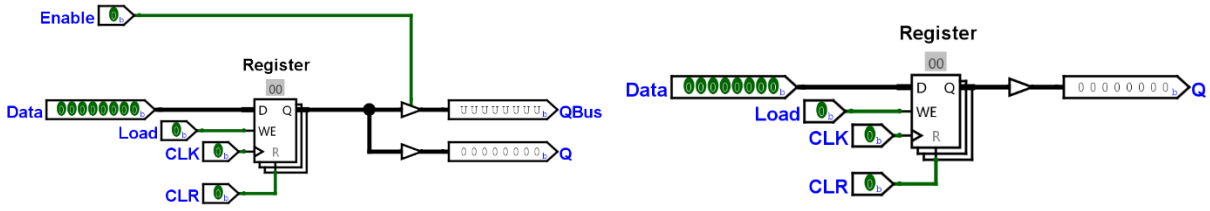
Registar instrukcija (engl. Instruction Register, IR) sadrži trenutačnu instrukciju koju izvršava središnja procesorska jedinica. Tijekom ciklusa dohvata instrukcije, središnja procesorska jedinica čita instrukcije iz memorije i pohranjuje ih u registar instrukcija. Instrukcija iz registra instrukcija dekodira se u upravljačkoj jedinici, koja određuje potrebne mikroinstrukcije. Prema prikazanoj slici, za izvedbu koristi se 8-bitni registar koji sadrži dva izlaza. Jedan izlaz za operand spojen je na sabirnicu te postaje dostupan nakon primitka kontrolnog signala. Drugi izlaz je za instrukcijski kod koji je povezan s upravljačkom jedinicom.



Slika 6.4. Izvedba registra instrukcije u Logisim-u

Akumulator i B registar

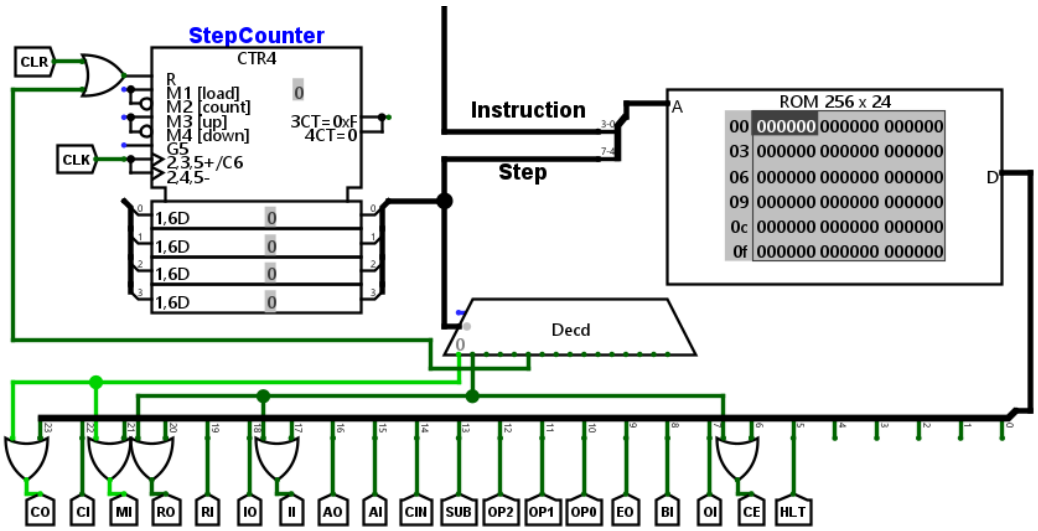
Akumulator je registar koji se koristi za pohranu prvog operanda te za pohranu rezultata aritmetičkih i logičkih operacija aritmetičko logičke jedinice. Akumulator je povezan sa sabirnicom i aritmetičko logičkom jedinicom, podaci su stalno dostupni aritmetičko logičkoj jedinici, dok su sabirnici dostupni samo pri aktiviranju odgovarajućeg upravljačkog signala. B registar koristi se za pohranu drugog operanda tijekom aritmetičkih i logičkih operacija, za razliku od akumulatora izlaz B registra nije spojen na sabirnicu. Na slici je prikazana izvedba akumulatora i B registra.



Slika 6.5. Izvedba registra operanda u Logisim-u: a) Akumulator, b) B registar

Upravljačka jedinica

Upravljačka jedinica odgovorna je za koordinaciju i kontrolu rada procesora. U ovom procesoru se koristi ROM (engl. Read Only Memory) za pohranu mikroinstrukcija. Adresiranje tih mikroinstrukcija uključuje kombinaciju rednog broja koraka i koda instrukcije. Svaka instrukcija ograničena je na maksimalno četiri koraka. Prva dva koraka instrukcije uvijek su ista, točnije dohvaćanje (engl. fetch), što omogućuje hardversku implementaciju tih koraka bez upotrebe ROM-a. Na slici je prikazana skupina komponenata koje čine upravljačku jedinicu.



Slika 6.6. Izvedba upravljačke jedinice u Logisim-u

Dohvaćanje i izvršavanje instrukcije

Proces izvršavanja instrukcije sastoji se od dva koraka:

1. Dohvaćanje instrukcije (engl. fetch) – procesor dohvati sljedeću instrukciju iz memorije.
2. Izvršavanje instrukcije (engl. execute) – procesor dekodira instrukciju i šalje potrebne kontrolne signale za izvršavanje instrukcije.

Ciklus dohvaćanja i izvršavanja instrukcije ovog procesora može se objasniti u sljedećim koracima:

- Registar memorijske adrese sprema memorijsku adresu koju mu dodjeljuje programski brojač. Upravljački signali: CO (Counter Out), MI (MAR In).
- Programski brojač se nakon toga povećava za jedan kako bi pokazivao na sljedeću instrukciju u memoriji. Upravljački signali: CE (Counter Enable).
- Kada registar memorijske adrese omogući pristup toj adresi u radnom memoriji, radna memorija na izlazu daje instrukciju koja se prenosi u registar instrukcija. Upravljački signali: RO (RAM Out), II (IR In).

- Završni korak je dekodiranje instrukcije u kojem kontrolna jedinica dekodira instrukciju iz registra instrukcija u odgovarajuće kontrolne signale za komponente poput registra, aritmetičko logičke jedinice i radne memorije. Primjer upravljačkih signala za obavljanje instrukcije ADD:
    1. Učitavanje adrese na kojoj se nalazi drugi operand iz instrukcijskog registra u registar memorijske adrese. Upravljački signali: IO (IR Out), MI (MAR In).
    2. Učitavanje drugog operanda iz RAM-a u B registar. Upravljački signali: RO (RAM Out), BI (B In).
    3. Izvođenje operacija zbrajanja koja koristi kontrolni kod ALU-a 100 te spremanje rezultata u akumulator. Upravljački signali: OP2=1, OP1=0, OP0=0, EO (ALU Out), AI (A In).

6.2 Programiranje procesora

Računalo razumije strojni jezik, odnosno binarne kodove koje izravno upravljaju središnjom procesorskom jedinicom. Međutim, strojni jezik nije praktičan za programiranje, stoga se koristi asemblerski jezik koji koristi mnemonike, kratke nizove slova koji predstavljaju pojedine instrukcije. Instrukcije se sastoje od operacijskog koda i operanda, te se svaka instrukcija asemblerskog jezika prevodi u jednu instrukciju strojnog jezika. Operacijski kod šalje se kontrolnoj jedinici koja generira upravljačke signale za izvršenje instrukcija. Operand se šalje u memorijski adresni registar i koristi se za dohvaćanje lokacije željene vrijednosti iz memorije. Na tablici je prikazana lista mnemonika s odgovarajućim operacijskim kodom, te prikaz asemblerskih instrukcija prevedenih u strojni jezik.

Mnemonik	Operacijski kod	Asembler	Strojni jezik
LDA	0001	LDA 5	0001 0101
STA	0010	STA 5	0010 0101
ADD	0011	ADD 5	0011 0101
SUB	0100	SUB 5	0100 0101
AND	0101	AND 5	0101 0101
OR	0110	OR 5	0110 0101
XOR	0111	XOR 5	0111 0101
NOT	1000	NOT 5	1000 0101
JMP	1001	JMP 5	1001 0101
OUT	1010	OUT	1010 0000
HLT	1011	HLT	1011 0000

Tablica 6.1. Instrukcijski set procesora obrađenog u radu

Primjer 1: Zbrajanje dva broja

```
0: LDA 4
1: ADD 5
2: OUT
3: HLT
4: 2
5: 3
```

Primjer 2: Fibonaccijev niz

```
0: 1
1: STA D
2: LDA 0
3: STA E
4: ADD D
5: OUT
6: STA F
7: LDA E
8: STA D
9: LDA F
A: STA E
B: JMP 4
```

Primjer 3: Aritmetički niz

```
0: 1
1: 3
2: LDA 0
3: OUT
4: ADD 1
5: STA 0
6: JMP 2
```

7. Zaključak

Najprije su prezentirani brojevni sustavi s naglaskom na binarni brojevni sustav i operacije nad binarnim brojevima. Proučeni su osnovni logički sklopovi koji se koriste za izradu složenijih digitalnih sklopova poput multipleksora, demultipleksora, registra i brojila. Zatim su pomoću logičkih sklopova ostvarene operacije binarnog zbrajanja, pri čemu su uzete u obzir dva tipa zbrajala. Nakon razumijevanja osnovnih koncepata, u softveru Logisim implementirana je aritmetička logička jedinica koja podržava šest operacija. Potom je konstruirana jednostavna centralna procesorska jedinica bazirana na SAP-1 arhitekturi u kojoj je implementirana aritmetička logička jedinica. Kako bi se demonstrirale funkcionalnosti CPU-a, napisana su tri jednostavna programa. Ovaj rad je koristan za shvaćanje načina funkcioniranja računala. Procesor predstavljen u radu pomaže u razumijevanju toka podataka između različitih komponenti procesora. Procesor također olakšava razumijevanje načina na koji se programski jezik više razine, poput asemblera, prevodi u strojni kod. Znanje stečeno ovim radom može se primijeniti na složenije vrste dizajna aritmetičko logičke jedinice i središnje procesorske jedinice.

8. Literatura

[1] Stanko Paunović, Digitalna elektronika, Školska knjiga, 1995

[2] Anil K. Maini, Digital Electronics, 2007

[3] Albert P. Malvino, Jerald A. Brown, Digital Computer Electronics, 1999

[4] Thomas L. Floyd, Digital Fundamentals, 2015

[5] Slobodan Ribarić, Građa računala, 2011

[6] https://www.youtube.com/watch?v=HyznrdDSSGM&list=PLowKtXNTBypGqImE405J2565dvjafglHU

9. Popis slika

10. Popis tablica

11. Sažetak

Zadatak ovog završnog rada je izvedba aritmetičko logičke jedinice u simulacijskom softveru Logisim. Opisani su brojevni sustavi i komponente digitalne logike potrebne za izvedbu aritmetičko logičke jedinice i procesora. Obrađena su dva tipa zbrajala, ripple-carry i carry look-ahead zbrajalo. U softveru je izvedena aritmetičko logička jedinica sa šest operacija, te procesor temeljen na SAP-1 arhitekturi. Objašnjen je rad procesora te su napisana tri programa za procesor.

Ključne riječi: ALU, Logisim, digitalna logika, zbrajala, procesor

12. Abstract

The task of this final work is the implementation of the arithmetic logic unit in the simulation software Logisim. This involves describing the number systems and digital logic components necessary for the arithmetic logic unit and processor implementation. Two types of adders are described, ripple-carry and carry look-ahead adder. An arithmetic logic unit with six operations and a processor based on the SAP-1 architecture are implemented in the Logisim software. The functioning of the processor is explained, and three programs are created for the processor.

Keywords: ALU, Logisim, digital logic, adders, processor