Tallinna Tehnikaülikool

Arvutisüsteemide instituut

Praktikumi nr. 3 juhendmaterjal

lühendatud versioon

IAX0043

Lembitu Valdmets

Protsessori üldstruktuur:

Kasutatav seade kuulub 1 aadressiga protsessorite hulka, mis tähendab, et käsule antakse kaasa 1 operand. Protsessoril on 5-bitine operand, mille sees on praegu kasutuses 28 käsku (5-bitti mahutavad maksimaalselt 32 käsku). Protsessori juhtautomaat on mikroprogramne, mis võimaldab protsessorit mikroprogrammeerida, optimeerida selle tööd ning lisada uusi käske.

Järgnevalt on välja toodud protsessorimudeli põhiosad ja nende tööpõhimõtete kirjeldused:

Protsessori põhisiin – on seade, mis ühendab enesega pea kõiki komponente, mis suhtlevad 8-bitises vormingus. Käesoleva protsessori põhisiinil toimub nii andmete kui käskude kopeerimine. Kuna põhisiini on seadmel ainult 1, toimub andmete kopeerimine ükshaaval, mida reguleerib juhtautomaat. Siinil on kasutuses elementide väljundite ja siini vahel väljundpuhvrid, et takistada andmete leket põhisiinile.

Käsuloendur(program Counter) – joonisel PC on register, mis hoiab eneses järgmisena töötlusesse mineva käsu aadressi. Registrit saab suurendada 1 võrra või kirjutada selle väärtust üle suvalise väärtusega.

Mälu aadressiregister (lühendina MAR) on register, mille väärtusega pöördutakse mälu poole sealt andmete lugemise ja sinna andmete kirjutamise korral.

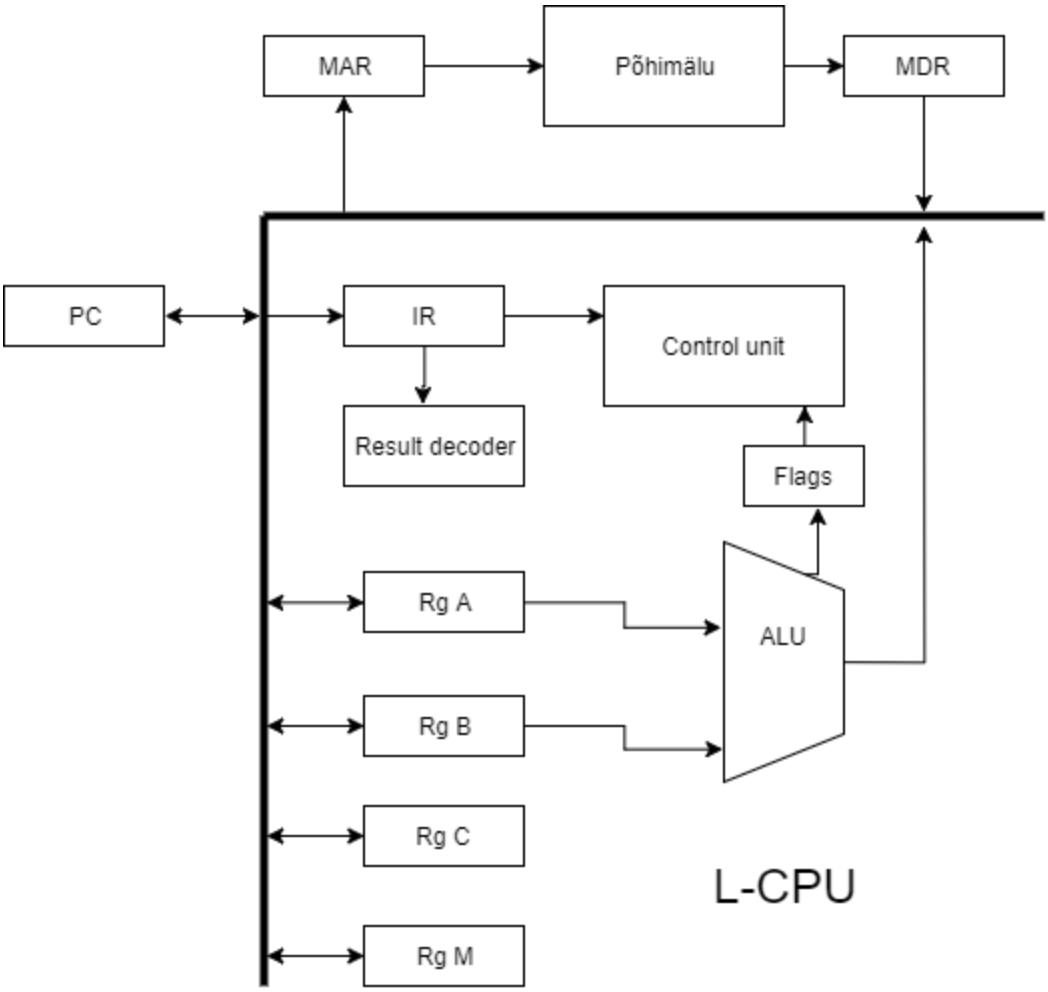
Mälu andmeregister (lühendina MDR) on register, kuhu salvestatakse mälust lugemise ajal andmebaidi väärtus enne, kui antud andmebait saadetakse edasi soovitud protsessoriossa. Salvestuskäskude korral mälu andmeregistrit ei kasutata ja andmed kopeeritakse põhimällu otse põhisiinilt.

Aritmeetika-Loogikaplokk (lühendina ALU) on seade, mis tegeleb erinevate aritmeetika ja loogikaoperatsioonidega. Käesolev Aritmeetika- ja loogikaplokk koosneb 4-bitisest operatsioonisisendist, mistõttu on võimalik sooritada kuni 16 erinevat operatsiooni. Operantide sisendid on jäigalt seotud registrite A ja B väljunditega.

Käsuregister (lühendina IR) on register, mis hoiab eneses töötluses oleva käsu väärtust. Registri väljund on ühendatud juhtseadmega, mis kasutab selle väärtusi programmi töös.

Tulemregistri dekooder (Result Decoder) – on seade, mis loeb töödeldava käsu viimast 3 bitti. Lihtsa joonise peal on seadme väljundid visualiseerimata, sest ühendused teiste komponentidega on ainult 1 bitise laiusega.

Registrid A, B C M – on registrid, kuhu saab salvestada nii ajutisi muutujaid kui ka operande, tulemusi ja mäluaadresse.


MAR
Põhimälu
MDR
PC
IR
Control unit
Result decoder
Flags
Rg A
ALU
Rg B
Rg C
Rg M

L-CPU

Protsessori kasutamine:

Protsessor on konstrueeritud loogikasimulaatoris Logisim ja seetõttu kasutamine toimub samas keskkonnas. Hierarhilise koostamise tõttu on alamkomponendid salvestatud omaette failidesse, kust neid saab eraldi avada ja redigeerida. Skeemi avamiseks tuleb avada peamine fail, mis on märgitud nimega „pohiskeem.circ“ Kui kõik alamskeemide failid on samas kaustas, siis avaneb skeem probleemideta. Protsessori juhtloogika põhineb mikroprogrammil, mille sisu on saadaval failis „protsessorROMsisu.txt“.

Programmeerimine toimub kas põhimällu käsitsi väärtuste sisestamist kuueteistkümnendsüsteemis või laetakse põhimällu tekstifail, kuhu on programm kirjutatud. Protsessori enese tööd saab simuleerida kas manuaalselt taktsignaali vajutades või automaatse taktigeneraatoriga.

Protsessor on Logisimis tehtud, kui 1 suur digitaalskeem, jälgides sama mudelit, mis oli ka lihtsa skeemi peal. Logisimis oleval skeemil eksisteerivad kõik ühendused seadmete vahel, mida digitaalskeemi simuleerimiseks vaja on. Põhikomponendid on siiski samad na nende tegevust jälgides on võimalik aru saada, mida protsessor mingil ajahetkel teeb.

Protsessori mudeli korrektseks käivitamiseks ning tõrgeteta kasutamiseks on soovitatav käituda järgmiselt:

*Käivitada programm Logisim

*Avada fail pohiskeem.circ

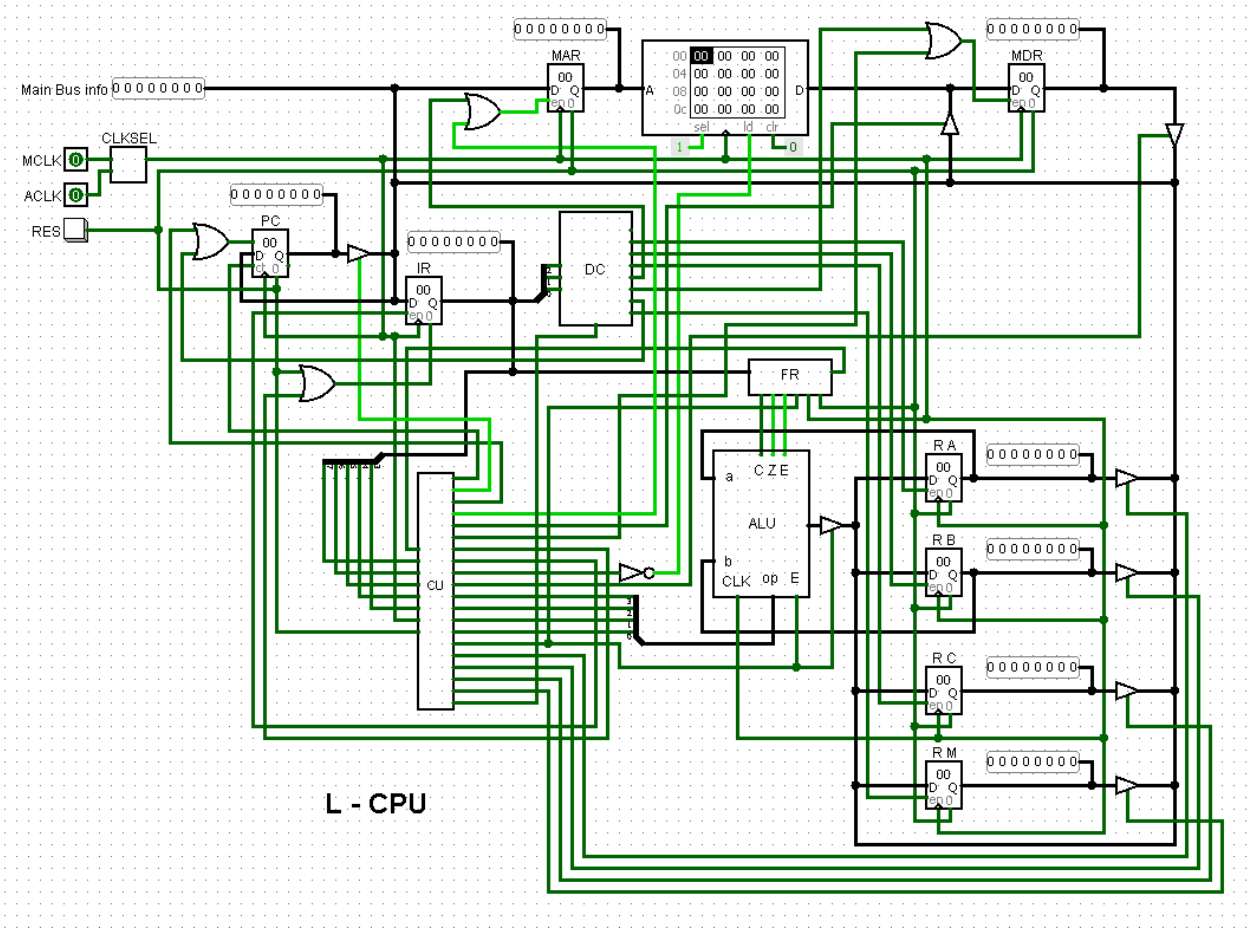
*Avada protsessori skeemil topeltklikiga juhtautomaadi alaskeem

*Paremklikiga ROM-elemendi peal laadida “Load image...” ning valida fail nimega protsessoriROMsisu.txt

*Protsessori kasutamise alustamiseks soovitatav vajutada “RES” nuppu

*Protsessor on valmis kasutamiseks

Juhul, kui programmerimise või käskude täitmise ajal tekib olukordi, kus protsessor käitub kummaliselt või programm on jälgitavast kohast mööda liikunud, on soovitatav vajutada nuppu RESET, mis asünkroonselt lähtestab kõik registrid väärtusega 0 ning seade alustab oma tööd uuesti kohalt 0.


Main Bus info
MCLK
ACLK
RES
CLKSEL
MAR
MDR
PC
IR
DC
FR
CU
ALU
RA
RB
RC
RM
L - CPU

Ülesanne:

Peamiseks ülesandeks jääb antud protsessorimudelil programmi koostamine, selle töö simuleerimine ja programmi töövoolu kirjeldamine.

Näiteülesanne ja lahendus:

Ülesanne: Lugeda andmebait väärtusega 23h andmepesast 0f ja kopeerida selle väärtus registrisse B.

Selgitus: Tegemist on ülesandega, kus peamiseks tegevuseks on andmete laadimine mälust. Kuna käesoleval protsessoril on 2 adresseerimise viisi, on meil võimalik valida nende vahel. Näitelahenduses on välja toodud mõlema adresseerimisega andmete laadimine:

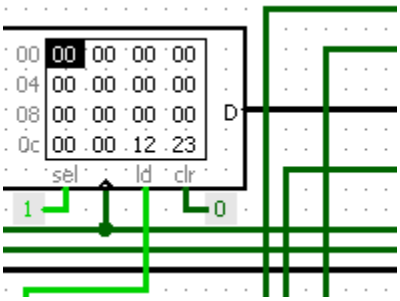
Laadekäsud leiab praktikumi juhendmaterjalist protsessori käsustiku peatükist ning need on järgnevad:
00001 – otsene adresseerimine

00010 – vahetu adresseerimine

Vahetu adresseerimine:

Vahetu adresseerimise korral on ülesannet mõnevõrra raskem lahendada, sest loetav andmebait asub kindla väärtusega andmepesas põhimälus. On võimalik kirjutada laadekäsk enne loetavat andmebaiti ning sellisel kujul lugeda sisse andmeväärtus.

Vahetu adresseerimisega laadekäsk koos andmebaidiga näeb Logisimi keskkonnas välja järgnevalt:



Põhiliseks miinuseks sellisel lahendusel on asjaolu, et programm alustab oma tegevust andmerealt 00h ning käib läbi kogu põhimälu kuni jõuab soovitud väärtuseni. Seda protsessi saab kiirendada, kasutades siirdekäske.

Otsene adresseerimine:

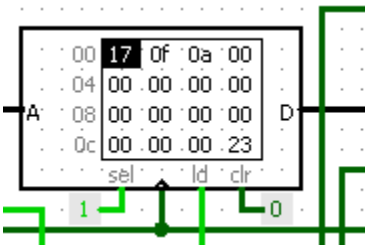
Antud ülesannet on efektiivsem lahendada otsese adresseerimisega. Selle võtta kasutamine eeldab, et registris M on ka laaditava andmebaidi väärtus. Seega esmalt tuleb registrisse M laadida vahetu adresseerimisega andmebaidi väärtus 0Fh ja seejärel kasutada otsest adresseerimist otsitava andmebaidini jõudmiseks.

Kirjutame esmalt käsu aadressi lisamiseks registrisse M:

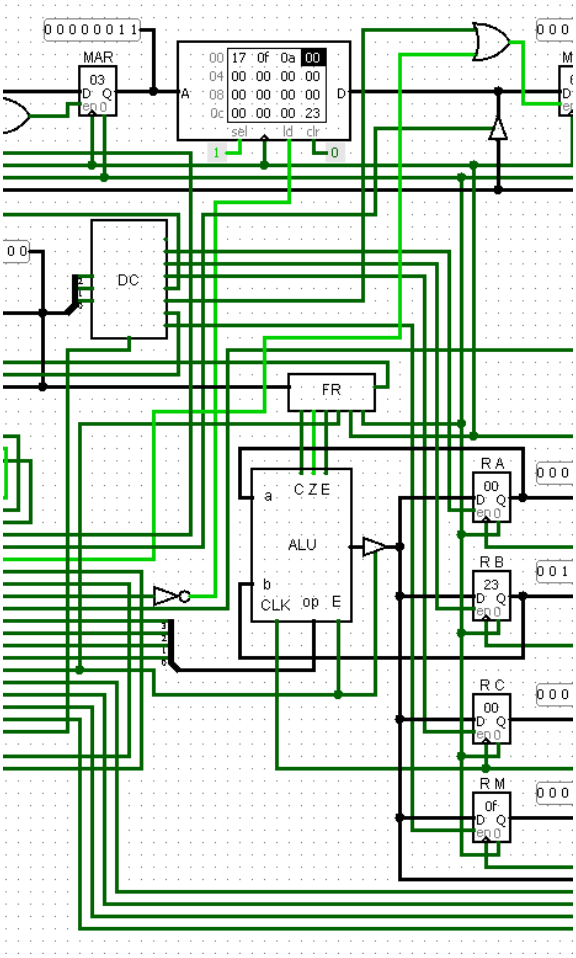
Seejärel kasutame teist laadekäsku, et kätte saada soovitud väärtus:

0Ah

Otsese adresseerimisega laadimine koos andmebaidiga näeb Logisimi keskkonnas välja järgnevalt:



Kui programm on põhimällu kirjutatud ja programmi käsud läbi jooksutatud, siis 4. käsu töötlemise ajal on põhiregistreid saavutanud oma soovitud oleku:



Kus Registrisse B on kopeeritud väärtus 23h.

Ülesanded:

Ülesanded on antud edu.pld.ttu.ee keskkonnas. Ülesandeks jääb peamiselt protsessori käsustikuga tutvumine, programmi koostamine protsessorimudelile ja programmi tõrgeteta demonstreerimine praktikumi juhendajale. Praktikum loetakse sooritatuks, kui üliõpilane on endatehtud kodutöö ette näidanud ja selgitanud programmi tööd.

Protsessori mikrokäsud

Protsessori juhtautomaadi disainimine põhineb klassikalise automaadi koostamisele, sellega seoses on käsu täitmisel protsessoril eri olekud, mis mikroprogramse juhtimise korral nimetatakse mikrokäskudeks. Järgnevalt on välja toodud protsessori juhtautomaadi mikrokäskude nimed ja nende kirjeldused. Mikrokäskude enese nimed on praktikumi juhendaja välja mõeldud, illustreerimaks nende üldist tegevust töö ajal. Selgituseks tuleb lisada, et probe'i all peetakse silmas skeemil seadet, mis näitab põhisiinil asuvat väärtust.

"PC to MAR" - Mikrokäsu eesmärgiks on kopeerida käsuloenduri sisu mälu aadressiregistrisse. Selle jaoks avatakse käsuloenduri väljundventiil siinile ning mälu aadressiregistri lubamissignaal omistab väärtuse "1". Taktsignaali positiivse signaali korral on protsessori põhisiinil võimalik läbi probe'i näha käsuloenduri väärtust.

"RAM to MDR" - Mikrokäsu eesmärgiks on põhimälus olevate andmete kopeerimine mälu andmeregistrisse. Aadress, kust andmed võetakse, on eelnevalt määratud Mälu aadressiregistri määramisega mikrokäsu

"PC to MAR" poolt. Paljude käskude täitmisel toimub just selle mikrokäsu ajal käsuloenduri suurendamine 1 võrra. Taktsignaali positiivse signaali korral on põhisiinil näha väärtust 0.

"MDR to IR" - Selle tegevuse juures kopeeritakse mälu aadressiregistrist andmed käsuregistrisse. Selle mikrokäsu lõpus on käsuregistrisse saadud käsk, mida hakatakse töötlema. Taktsignaali positiivse väärtuse korral on põhisiinil näha käsu väärtust.

"Decode IR" - mikrokäsk on vajalik, sest selle oleku juures juhtautomaat analüüsib sisendite ehk opkoodi järgi, millisesse järgnevasse olekusse juhtautomaat läheb. Ideaalis oleks saanud selle mikrokoodi vahele von Neumanni tsüklis jätta ning kohe peale

"MDR to IR" suunduda käsu täitmise poole, kuid arusaadavuse paremaks tagamiseks ning traditsioonilise tsükli simuleerimiseks jäeti sisse. Põhisiinil mikrokäsu ajal on väärtus 0. "MDR to reg X" - selle mikrokäsu juures kopeeritakse mälu andmeregistrist andmed registrisse, millele tulemregistri dekooder on lubava signaali väljastanud. Korraga on lubatud olekus vaid 1 register.

“ALU A & B” - Selle mikrooperatsiooni jooksul suhtleb juhtautomaat peamiselt aritmeetika- ja loogikaseadmega (ALU). ALU’le saadetakse opkoodi järgselt 4-bitine käsukood ning lubamissignaal. Tulemregistri dekooder määrab mikrokäsu ajal registri, 19 kuhu ALU poolt arvutatud tulemus saadetakse ning sinna registrisse saadetakse lubamissignaal. Põhisiini peal on näha ALU poolt arvutatud tulemust.

“JMP w flags” - Eesmärgiks on järgmiseks mikrokäsuks tuvastada, kas siidrekäsk läheb täitmisesse või lõpeb käsk ilma siiret sooritamata. Seda kontrollib juhtautomaat “flag input” - nimelise sisendi abil, oleku enda ajal juhtautomaadi väljundid on madala nivoo peal.. Antud mikrokäsk iseenesest ei ole hädavajalik ning siirdekäsku saaks korraldada ka ilma mikrokäsuta, sest käesolev mikroprogramne juhtautomaat on suuteline määrama järgnevat olekut nii sisendite kui ka konkreetse oleku põhjal. Parema arusaadavuse huvides on see mikrokäsk sisse jäetud riistvaralisest lahendusest, sest nii on hõlpsam aru saada protsessori töötsüklist. Taktsignaali ajal on käsuloenduril väärtus 0.

“M-reg to PC” - Antud mikrokäsu korral kopeeritakse M-registri sisu käsuloendurisse. Selle jaoks avatakse puhver M- registri väljundis ning lubav signaal käsuloenduri sisendis. Põhisiini peal on näha M-registris asuvat väärtust.

“X-reg to reg-Y” - Eesmärgiks on kopeerida opkoodi poolt antud registrist väärtus registrisse, millele viitab tulemregistri dekooder. Põhisiinil on näha kopeeritava andmebaidi väärtust.

“M-reg to MAR” - Selle mikrokäsu korral kopeeritakse M-registri väärtus mälu aadressiregistrisse. Selline tegevus on vajalik peamiselt salvestuskäskude sooritamiseks. Põhisiinil on näha kopeeritava andmebaidi väärtust.

“Reg C to RAM” - Tegemist on ainsa mikrokäsuga, mille korral toimub põhimällu andmete kopeerimine. Põhisiinil on näha kopeeritava andmebaidi väärtust.

“MDR to PC” - Antud mikrokäsk kopeerib mälu andmeregistri sisu käsuloendurisse. See tegevus on oluline eestkätt siirdekäskude realiseerimisel. Põhisiinil on näha kopeeritava andmebaidi väärtust.

“END Command” - See mikrokäsk asetseb peaaegu iga käsu täitmise lõpus, mille ainsaks eesmärgiks on kustutada hetkel täitmisel oleva mikrokäsu sisemus ning asendada see väärtusega 0. Käsu lõpetamine on vajalik, sest käimasoleva käsu järgi saab juhtautomaat aru mitmete teiste mikrooperatsioonide juures, kas tegemist on käsu laadimise või juba täitmisega. Käsu laadimise puhul käsuregistris on väärtus 0, aga käsu täitmise ajal on käsuregistris täidetava käsu opkood.

Protsessori käsustik

Käsustiku kirjeldus:

Käesoleva protsessori käsustik kuulub 2-aadressiga arvutite käsustiku alla sellepärast, et käsustikus lisaks opkoodile on võimalik vastavalt käsule kaasata kuni 2 aadressi.

Käsud jaotuvad peamiselt 5-bitiseks opkoodiks ja 3-bitiseks operandiks. Opkood defineerib täidetava käsu ning 3-bitine operandiga määratakse sihtregister, kuhu salvestatakse käsu tulemus. Käsu tulemust ei salvestata kõikide käskude puhul, näiteks salvestuskäskude puhul on operand kasutu ning väärtuseks võib sisestada 0, sellest pikemalt juba vastavates alapeatükkides.

Järgnevalt on välja toodud kolm näidet erineva pikkusega käsust:

Tühikäsk

Opkood	...

Registri kopeerimiskäsk

Opkood	Operand(sihtregistri 3-bitine kood)

Laadekäsk vahetu adresseerimisega

Opkood	Operand(sihtregistri 3-bitine kood)
Operand (8-bitine väärtus)	

Joonis 1. Erinevate käsipikkuste kujutamine.

Käskude kirjeldused käsutüüpide kaudu

Järgnevalt on välja toodud käskude kirjeldused käsutüüpide kaudu. Kõige efektiivsem on käske liigitada nende tüüpide kaudu, siis sarnaste käskude kirjeldamine on tõhusam.

Erandlikuks käsuks saab nimetada tühikäsku, mis ei tee protsessori juures midagi muud, kui hakkab pärast dekodeerimist järgmist käsku laadima. Tühikäsu tsükkel on välja toodud järgnevalt ning taktide täitumist tuleb lugeda vasakult paremale.

PC to MAR	RAM to MDR	MDR to IR	Decode IR

Joonis 2. Tühikäsu taktideks jagamine.

Opkood: 00000

Takte: 4

Operandi kasutamine tühikäsu puhul ei ole vajalik.

Laadekäsud

Laadekäske on kaks ja need erinevad teineteisest adresseerimise viiside poolest. Opkoodi järgne operand on oluline, sest see määrab, kuhu laetakse väärtus.

00001 - Laadida mälust, kasutades M registrit

Selle käsu puhul on tegemist otsese adresseerimisega. Andmed loetakse aadressist, mida talletab eneses M-register.

Takte: 8

00010 – Laadida mälust, kasutades aadressi, mida käsuloendur hoiab

Siin on tegemist vahetu adresseerimisega. Käsku täites laetakse justkui uut käsku, kuid käsuregistrisse kopeerimise asemel kopeeritakse andmed soovitud registrisse.

Takte: 8

Käsu 00001 taktideks jagamine

PC to MAR	RAM to MDR	MDR to IR	Decode IR	M -reg to MAR	RAM to MDR	MDR to Reg X	END Command

Käsu 00010 taktideks jagamine

PC to MAR	RAM to MDR	MDR to IR	Decode IR	PC to MAR	RAM to MDR	MDR to Reg X	END Command

Joonis 3. Laadekäskude taktideks jagamine.

Salvestuskäsud

Salvestuskäsud omavad samuti vahetut ja otsest adresseerimise võimalust, kuid 3-bitine operandi väärtus ei mõjuta käsu täitmist.

00100 – salvestada mällu registrist C, kasutades M-registrit aadressi jaoks

Takte: 7

00101 – salvestada põhimällu registrist C, kasutades aadressi, mida käsuloendur hoiab

Käesoleva käsu kasutamisel tuleb ettevaatlik olla, kuna käsust järgnev mälupesa kirjutatakse üle uue väärtusega, mis asetseb registris C.

Takte: 7

Käsu 00100 taktideks jagamine

PC to MAR	RAM to MDR	MDR to IR	Decode IR	M -reg to MAR	Reg C to RAM	END Command

Käsu 00101 taktideks jagamine

PC to MAR	RAM to MDR	MDR to IR	Decode IR	PC to MAR	Reg C to RAM	END Command

Joonis 4. Salvestuskäskude taktideks jagamine.

Aritmeetika- loogikakäsud

Aritmeetika- ja loogikakäsud on kõik samasuguste taktidega, peamine erinevus on opkoodis, mis määrab, missugust ALU funktsiooni kasutatakse. Siinsete käskude puhul on oluline kasutada 3-bitist opkoodi, sest see määrab, kuhu kirjutatakse tulemi väärtus. Kui opkoodiks sisestada 0, siis tulemuse väärtust ei kirjutata kuhugi, ainult lippude väärtused muutuvad.

Aritmeetika- ja loogikakäskude taktideks jagamine

PC to MAR	RAM to MDR	MDR to IR	Decode IR	ALU A & B	END Command

Joonis 5. Aritmeetika- ja loogikakäskude taktideks jagamine.

10000 – Registrite A ja B loogiline AND, tulemus kirjutatakse sihtregistrisse

Takte. 6

10001 – Registrite A ja B loogiline OR, tulemus kirjutatakse sihtregistrisse

Takte. 6

10010 – Registrite A ja B loogiline NOR, tulemus kirjutatakse sihtregistrisse

Takte. 6

10011 – Registrite A ja B loogiline XOR, tulemus kirjutatakse sihtregistrisse

Takte. 6

10100 – Registrite A ja B liitmine ülekandeta, tulemus kirjutatakse sihtregistrisse

Takte. 6

10101 – Registrite A ja B lahutamine, tulemus kirjutatakse sihtregistrisse

Takte. 6

10110 – Registrite A ja B võrdlus, tulemus kirjutatakse sihtregistrisse

Takte. 6

10111 – Registrite A ja B liitmine ülekandega, tulemus kirjutatakse sihtregistrisse

Takte. 6

11000 – Registri A ringnihe paremale, tulemus kirjutatakse sihtregistrisse

Takte. 6

11001 – Registri B ringnihe paremale, tulemus kirjutatakse sihtregistrisse

Takte. 6

11010 – Registri A ringnihe vasakule, tulemus kirjutatakse sihtregistrisse

Takte. 6

11011 – Registri B ringnihe vasakule, tulemus kirjutatakse sihtregistrisse

Takte. 6

11100 – Registri A inverteerimine, tulemus kirjutatakse sihtregistrisse

Registrit inverteeritakse pöördkoodi alusel, seega kõikide bittidele omistatakse vastandväärtused.

Takte. 6

11101 – Registri B inverteerimine, tulemus kirjutatakse sihtregistrisse

Registrit inverteeritakse pöördkoodi alusel, seega kõikide bittidele omistatakse vastandväärtused.

Takte. 6

11110 – Registri B dekrementeerimine, tulemus kirjutatakse sihtregistrisse

Takte. 6

11111 – Registri A inkrementeerimine, tulemus kirjutatakse sihtregistrisse

Takte. 6

Registrite-vahelised kopeerimiskäsud

Registrite omavahelised kopeerimiskäsud on olemuselt sarnased aritmeetika- ja loogikakäskude täitmisele taktide arvu poolest. Lühikese pikkusega sihtaadress on oluline, sest sellega defineeritakse register, kuhu andmed kopeeritakse. Tuleb märkida, et käsuloendurit sihtregistrina kasutades on võimalik teostada programmi siirdeid ehk hüppekäske.

Registritevaheliste kopeerimiskäskude taktideks jagamine

PC to MAR	RAM to MDR	MDR to IR	Decode IR	X-reg to reg-Y	END Command

Joonis 6. Registritevaheliste kopeerimiskäskude taktideks jagamine.

00110 – registri A sisu kopeerimine sihtregistrisse

Takte: 6

00111 – registri B sisu kopeerimine sihtregistrisse

Takte: 6

01000 – registri C sisu kopeerimine sihtregistrisse

Takte: 6

01001 – registri M sisu kopeerimine sihtregistrisse

Takte: 6

Siirdekäsud

Siirdekäsud on protsessori juures väga olulised, sest need võimaldavad teostada tingimustega siirdeid ühest programmi osast teise. Sellega on võimalik teostada nii tingimusi kui ka tsükleid programmides, andes laiad võimalused protsessori programmeerimiseks. Siirdekäskudel ei ole olulised lühikesed operandi väärtused, sestap on soovitatav need asendada väärtusega 0.

Järgnevalt on välja toodud kõik kolm tingimuskäsu opkoodi ja tingimust. Käsu täitmise pikkus taktides ei sõltu tingimusest endast, vaid tingimuse täituvusest.

01011 – Hüpata aadressile, mida hoiab käsuloendur, kui ALU tulemus on 0.

01100 – Hüpata aadressile, mida hoiab käsuloendur, kui ALU tulemus on ületäitumine.

01101 – Hüpata aadressile, mida hoiab käsuloendur, kui ALU tulemus on võrdne

Tingimuse mittetäitumise korral

PC to MAR	RAM to MDR	MDR to IR	Decode IR	JMP w flags	END Command

Tingimuse täitumise korral

PC to MAR	RAM to MDR	MDR to IR	Decode IR	JMP w flags	PC to MAR	RAM to MDR	MDR to PC	END Command

Tingimusteta siirdekäsu täitmine

PC to MAR	RAM to MDR	MDR to IR	Decode IR	PC to MAR	RAM to MDR	MDR to PC	END Command

Joonis 7. Siirdekäskude taktideks jagamine.

Lisaks tingimuslikele siirdekäskudele eksisteerib veel tingimusteta siirdekäsk:

01110 – Hüpata aadressile, mida hoiab käsuloendur

Takte: 8

Käsustiku kirjelduse lõppu tuleb kokkuvõttev tabel, kuhu on kirjutatud käskude opkoodid ja nende väärtused kuueteistkümnendsüsteemis, lisaks ka lihtne käsu seletus. Kuueteistkümnendsüsteemis on näidatud vasakpoolne väärtus, sest parempoolsel väärtusel on teada vaid esimene bitt enamik käskudel. Kui käsul sihtregister ei ole oluline, on võimalik kohe määrata terve käsusõna, muul juhul käsusõna moodustub käsu opkoodi ja operandi liitmisel.

Bittide väärtused	Väärtus 16’nd koodis	selgitus
00000	00	tühikäsk
00001	0-	Laadida mälust, kasutades M-registrit
00010	1-	Laadida mälust, kasutades aadressi, mida käsuloendur hoiab
00011	1-	tühikäsk
00100	20	salvestada mällu registrist c, kasutades M- registrit aadressi jaoks
00101	28	salvestada mällu registrist c, kasutades aadressi, mida käsuloendur hoiab
00110	3-	A registri sisu kopeerimine sihtregistrisse
00111	3-	B registri sisu kopeerimine sihtregistrisse
01000	4-	C registri sisu kopeerimine sihtregistrisse
01001	4-	M registri sisu kopeerimine sihtregistrisse
01010	5-	tühikäsk
01011	58	Hüpata aadressile, mida käsuloendur hoiab, kui ALU tulemus on 0
01100	60	Hüpata aadressile, mida käsuloendur hoiab, kui ALU tulemus on ületäitumine
01101	68	Hüpata aadressile, mida käsuloendur hoiab, kui ALU tulemus on võrdne
01110	70	Hüpata aadressile, mida käsuloendur hoiab
01111	7-	tühikäsk
10000	8-	Registrite A ja B loogiline AND, tulemus kirjutatakse sihtregistrisse
10001	8-	Registrite A ja B loogiline OR, tulemus kirjutatakse sihtregistrisse
10010	9-	Registrite A ja B loogiline NOR, tulemus kirjutatakse sihtregistrisse
10011	9-	Registrite A ja B loogiline XOR, tulemus kirjutatakse sihtregistrisse
10100	A-	Registrite A ja B liitmine ülekandeta, tulemus kirjutatakse sihtregistrisse
10101	A-	Registrite A ja B lahutamine ülekandeta, tulemus kirjutatakse sihtregistrisse
10110	B-	Registrite A ja B võrdlus
10111	B-	Liitmine ülekandega
11000	C-	registri A ringnihe paremale
11001	C-	registri B ringnihe paremale
11010	D-	registri A ringnihe vasakule
11011	D-	registri B ringnihe vasakule
11100	E-	registri A inverteerimine
11101	E-	registri B inverteerimine
11110	F-	registri B dekrementeerimine
11111	F-	Registri A inkrementeerimine

Kindlasti tuleb mainida ka lühikese operandi olemus ja selle kasutamine:

Lühike operand on 3-bitine operand, mis käib käsusõna lõppu ja määrab vastavalt käsust tulemuse aadressi, kuhu see salvestatakse.

Lisaks on välja toodud tabelisse veerud, kuhu on kirjutatud käsusõna väärtused kuueteistkümnendsüsteemis vastavalt sellele, kas opkood lõppeb väärtusega 0 või 1.

Bittide väärtused:	Kui opkood lõpeb 0'ga:	Kui opkood lõpeb 1'ga:	Register, kuhu salvestatakse:
000	0	8	Mitte kuhugi
001	1	9	Register A
010	2	A	Register B
011	3	B	Register C
100	4	C	Mälu aadressiregister
101	5	D	Mälu andmeregister
110	6	E	Käsuloendur
111	7	F	Register M