ЈОВАН ЂОРЂЕВИЋ

АРХИТЕКТУРА И ОРГАНИЗАЦИЈА РАЧУНАРА

ЗБИРКА РЕШЕНИХ ЗАДАТАКА

Београд , 2011.

ПРЕДГОВОР

Књига садржи решене задатке из предмета Архитектура и организација рачунара. Задаци су груписани по областима и то извршавање инструкција, ...

Књига је написана у веома кратком временском периоду, па су, и поред извршених провера, могуће одређене грешке. Аутор ће бити захвалан свима онима који буду указили на откривене грешке.

Аутор

Београд

01.11.2011.

САДРЖАЈ

1 ЗАДАТАК 1

Посматра се део рачунара који чине меморија и процесор.

Меморија је капацитета 2^{16} бајтова. Ширина меморијске речи је 1 бајт.

Процесор је са једноадресним форматом инструкција. Подаци су целобројне величине без знака дужине два бајта. Подаци у меморији заузимају две суседне меморијске локације, при чему се старији бајт налази на нижој локацији а млађи бајт на вишој локацији.

У процесору постоји регистар програмског бројача PC дужине два бајта, адресни регистар меморије MAR дужине два бајта, прихватни регистар податка меморије MBR дужине један бајт, прихватни регистар инструкције IR дужине четири бајта, регистар акумулатора A дужине два бајта, помоћни регистар податка B дужине два бајта, регистри опште намене R[0] до R[31] дужине два бајта, програмска статусна реч PSW дужине један бајт, указивач на врх стека SP дужине 2 бајта, регистар броја улаза у табелу са адресама прекидних рутина BRU дужине 2 бита и указивач на табелу са адресама прекидних рутина IVTP дужине 2 бајта. Инструкције су дужине један, два, три или четири бајта.

Бит 7 првог бајта инструкције има вредност 0 за инструкције скока, при чему бит 6 првог бајта инструкције има вредност 0 за инструкција условног скока и 1 за инструкције безусловног скока. Инструкција условног скока је инструкција условног скока уколико је резултат нула (BZ). Битовима 5 до 0 првог бајта инструкције специфицира код операције за инструкције условног скока. На основу тога је за инструкцију BZ усвојен код операције 00000000. Инструкција BZ се реализује као релативни скок у односу на текућу вредност програмског бројача PC, а померај је 8 битна целобројна величина са знаком дата другим бајтом инструкције. Дужина инструкције је два бајта. Инструкције безусловног скока су инструкција безусловног скока (JMP) и инструкција скока на потпрограм (JSR). Битовима 5 до 0 првог бајта инструкције се специфицира код операције за инструкције безусловног скока. На основу тога су за инструкције JMP и JSR усвојени кодови операција 01000000 и 01000001, респективно. Инструкције JMP и JSR се реализују као апсолутни скокови, а адреса скока је дата другим и трећим бајтом инструкције, при чему је старији бајт адресе скока дат другим бајтом инструкције а млађи бајт адресе скока трећим бајтом инструкције. Дужина инструкција је три бајта.

Бит 7 првог бајта инструкције има вредност 1 за безадресне и адресне инструкције, при чему бит 6 првог бајта инструкције има вредност 0 за безадресне инструкције и 1 за адресне инструкције. Безадресне инструкције су инструкција стављања садржаја акумулатора на стек (PUSH), инструкција скидања садржаја са стека у акумулатор (POP), инструкција повратка из потпрограма (RTS) и инструкција повратка из прекидне рутине (RTI). Битовима 5 до 0 првог бајта инструкције специфицира се код операције за безадресне инструкције. На основу тога су за инструкције PUSH, POP, RTS и RTI усвојени кодови операција 10000000, 10000001, 10000010 и 1000000011, респективно. Дужина инструкција је један бајт. Адресне инструкције су инструкција преноса у акумулатор (LD), инструкција преноса из акумулатора (ST), аритметичка инструкција сабирања (ADD), логичка инструкција логички производ (AND), инструкција аритметичког померања удесно за једно место (ASR) и инструкција безусловног скока на срачунату адресу (JADR). У инструкцији ST није дозвољено непосредно адресирање,

у инструкцији JADR није дозвољено директно регистарско и непосредно адресирање, па уколико се јаве ова адресирања у овим инструкцијама, инструкције треба да буду без дејства, а инструкција ASR резултат померања смешта у регистар A. Битовима 5 до 0 првог бајта инструкција специфицира се код операције за адресне инструкције. На основу тога су за инструкције LD, ST, ADD, AND, ASR и JADR усвојени кодови операција 11000000, 11000001, 11000010, 11000011, 11000100 и 11000101, респективно. Дужина инструкција је два, три или четири бајта и зависи од специфицираног начина адресирања.

За адресне инструкције се битовима 7, 6 и 5 другог бајта инструкције специфицира начина адресирања и то на следећи начин: 000-регистарско директно адресирање (regdir), 001-регистарско индиректно адресирање (regind), 010-регистарско индиректно са постдекрементирањем адресирање (postdecr), 011-регистарско индиректно са преинкрементирањем адресирање (preincr), 100-меморијско директно адресирање (memdir), 101-меморијско индиректно адресирање (memind), 110-регистарско индиректно са померајем адресирање (regindpom) и 111-непосредно адресирање (immed). Адресирања код којих бит 7 има вредност 0 користе неки од регистара опште намене R[0] и R[31] специфициран битовима 4 до 0 другог бајта инструкције. Дужина инструкција је два бајта. Адресирања код којих бит 7 има вредност 1 имају трећи или трећи и четврти бајт инструкције. Код меморијског директног и меморијског индиректног адресирања трећи и четврти бајт инструкције садрже адресу меморијске локације, при чему је старији бајт адресе дат трећим бајтом а млађи бајт адресе четвртим бајтом. Код меморијског индиректног адресирања адреса дужине 16 бита заузима две суседне меморијске локације, при чему је старији бајт адресе налази на нижој а млађи бајт адресе на вишој локацији. Битови 4 до 0 другог бајта инструкције се не користе. Дужина инструкција је четири бајта. Код регистарског индиректног са померајем адресирања трећи бајт инструкције садржи 8 битни померај који је дат као целобројна величина са знаком. Један од регистара опште намене R[0] до R[31] који се користи специфициран је битовима 4 до 0 другог бајта инструкције. Дужина инструкција је три бајта. Код непосредног адресирања трећи и четврти бајт инструкције садрже 16 битни податак, при чему је старији бајт податка дат трећим бајтом а млађи бајт податка четвртим бајтом. Битови 4 до 0 другог бајта инструкције се не користе. Дужина инструкција је четири бајта.

Стек расте према нижим меморијским локацијама, а регистар SP указује на прву слободну меморијску локацију.

Захтеви за прекид долазе од четири улазно/излазна уређаја по линијама означеним од 0 до 3. По линији 0 стиже захтев за прекид најнижег, а по линији 3 највишег приоритета. Број линије највишег приоритета по којој је стигао захтев за прекид налази се у бинарном облику у регистру BRU. Адресе прекидних рутина 4 улазно/излазна уређаја који по линијама означеним од 0 до 3 шаљу захтеве за прекид налазе се у улазима 0 до 3 табеле са адресама прекидних рутина. Адресе дужине 16 бита заузимају по две суседне меморијске локације, при чему се старији бајт адресе налази на нижој локацији а млађи бајт адресе на вишој локацији. Садржај регистра BRU представља број улаза у табелу са адресама прекидних рутина. Почетна адреса табеле са адресама прекидних рутина се налази у регистру IVTP. У оквиру хардверског дела опслуживања захтева за прекид на стек са стављају само регистри PC и PSW.

Нацртати и објаснити дијаграм тока фаза извршавања инструкције и то: фазе читање инструкције, фазе формирање адресе и читања операнда, фазе извршавања операција LD, ST, PUSH, POP, ADD, AND, ASR, BZ, JMP, JADR, JSR, RTS и RTI и фазе опслуживање захтева за прекид.

РЕШЕЊЕ

Дијаграм тока извршавања инструкције је дат на сликама 1.а, 1.б, 1.в и 1.г. Извршавање инструкције се састоји из четири фазе: читање инструкције (слика 1.а), формирање адресе и читање операнда (слика 1.б), извршавање операција (слика 1.в) и опслуживање прекида (слика 1.г). Дијаграм тока је дат у сагласности са форматима инструкција (табела 1.а), при чему први, други, трећи и четврти бајт инструкције се смештају у разреде 31 до 24, 23 до 16, 15 до 8 и 7 до 0 прихватног регистра инструкције IR. У њему се користе сигнали логичких услова операција (табела 1.б), начина адресирања (табела 1.в) и дужина инструкција (табела 1.г) који се формирају на основу разреда 31 до 24 и 23 до 21 прихватног регистра инструкције. У дијаграму тока се користе и сигнали логичких услова **Z** и **PREKID**. Сигнал **Z** представља један од индикатора програмске статусне речи PSW који вредностима 1 и 0 указује да ли је резултат задње извршене инструкције нула или различит од нуле, респективно, док сигнал **PREKID** вредностима 1 и 0 указује да ли је током извршавања инструкције дошло или није дошло до генерисања прекида, респективно. Генерисање сигнала **Z** и **PREKID** је ван оквира овог задатка и није приказано.

Табела 1.а Формати инструкција

Instrukcija relativnog skoka BZ

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	OC						disp							

Instrukcije apsolutnog skoka JMP i JSR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
0	1	OC						high								low							

Bezadresne instrukcije PUSH, POP, RTS i RTI

31	30	29	28	27	26	25	24
1	0	OC					

Adresne instrukcije LD, ST, ADD, AND, ASR i JADR sa registarskim direktnim, registarskim indirektnim, registarskim indirektnim sa postdekrementiranjem i registarskim indirektnim sa preinkrementiranjem adresiranjima*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	1	OC						0	x	x	reg				

Adresne instrukcije LD, ST, ADD, AND, ASR i JADR sa registarskim indirektnim sa pomerajem adresiranjem

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
1	1	OC						1	0	0	reg					disp							

Adresne instrukcije LD, ST, ADD, AND, ASR i JADR sa memorijskim direktnim, memorijskim indirektnim i neposrednim adresiranjima

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	OC						1	0	1	reg					high								low							
1	1	OC						1	1	0	reg					high								low							
1	1	OC						1	1	1	reg					high								low							

* x je 0 ili 1

Табела 1.б Сигнали операција

$BZ=\overline{IR_{31}}\cdot\overline{IR_{30}}\cdot\overline{IR_{29}}\cdot\overline{IR_{28}}\cdot\overline{IR_{27}}\cdot\overline{IR_{26}}\cdot\overline{IR_{25}}\cdot\overline{IR_{24}}$

$JMP=\overline{IR_{31}}\cdot IR_{30}\cdot\overline{IR_{29}}\cdot\overline{IR_{28}}\cdot\overline{IR_{27}}\cdot\overline{IR_{26}}\cdot\overline{IR_{25}}\cdot\overline{IR_{23}}$

$JSR=\overline{IR_{31}}\cdot IR_{30}\cdot\overline{IR_{29}}\cdot\overline{IR_{28}}\cdot\overline{IR_{27}}\cdot\overline{IR_{26}}\cdot\overline{IR_{25}}\cdot IR_{24}$

$PUSH=IR_{31}\cdot\overline{IR_{30}}\cdot\overline{IR_{29}}\cdot\overline{IR_{28}}\cdot\overline{IR_{27}}\cdot\overline{IR_{26}}\cdot\overline{IR_{25}}\cdot\overline{IR_{24}}$

$POP=IR_{31}\cdot\overline{IR_{30}}\cdot\overline{IR_{29}}\cdot\overline{IR_{28}}\cdot\overline{IR_{27}}\cdot\overline{IR_{26}}\cdot\overline{IR_{25}}\cdot IR_{24}$

$RTS=IR_{31}\cdot\overline{IR_{30}}\cdot\overline{IR_{29}}\cdot\overline{IR_{28}}\cdot\overline{IR_{27}}\cdot\overline{IR_{26}}\cdot IR_{25}\cdot\overline{IR_{24}}$

$RTI=IR_{31}\cdot IR_{30}\cdot IR_{29}\cdot IR_{28}\cdot IR_{27}\cdot IR_{26}\cdot IR_{25}\cdot IR_{24}$

$LD=IR_{31}\cdot IR_{30}\cdot\overline{IR_{29}}\cdot\overline{IR_{28}}\cdot\overline{IR_{27}}\cdot\overline{IR_{26}}\cdot\overline{IR_{25}}\cdot\overline{IR_{24}}$

$ST=IR_{31}\cdot IR_{30}\cdot\overline{IR_{29}}\cdot\overline{IR_{28}}\cdot\overline{IR_{27}}\cdot\overline{IR_{26}}\cdot\overline{IR_{25}}\cdot IR_{24}$

$ADD=IR_{31}\cdot IR_{30}\cdot\overline{IR_{29}}\cdot\overline{IR_{28}}\cdot\overline{IR_{27}}\cdot\overline{IR_{26}}\cdot IR_{25}\cdot\overline{IR_{24}}$

$AND=IR_{31}\cdot IR_{30}\cdot\overline{IR_{29}}\cdot\overline{IR_{28}}\cdot\overline{IR_{27}}\cdot\overline{IR_{26}}\cdot IR_{25}\cdot IR_{24}$

$ASR=IR_{31}\cdot IR_{30}\cdot\overline{IR_{29}}\cdot\overline{IR_{28}}\cdot\overline{IR_{27}}\cdot IR_{26}\cdot\overline{IR_{25}}\cdot\overline{IR_{24}}$

$JADR=IR_{31}\cdot IR_{30}\cdot\overline{IR_{29}}\cdot\overline{IR_{28}}\cdot\overline{IR_{27}}\cdot IR_{26}\cdot\overline{IR_{25}}\cdot IR_{24}$

Табела 1.в Сигнали начина адресирања

$regdir=\overline{IR_{23}}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}$

$regind=\overline{IR_{23}}\cdot\overline{IR_{22}}\cdot IR_{21}$

$postdecr=\overline{IR_{23}}\cdot IR_{22}\cdot\overline{IR_{21}}$

$preincr=\overline{IR_{23}}\cdot IR_{22}\cdot IR_{21}$

$regindpom=IR_{23}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}$

$memdir=IR_{23}\cdot\overline{IR_{22}}\cdot IR_{21}$

$memind=IR_{23}\cdot IR_{22}\cdot\overline{IR_{21}}$

$immed=IR_{23}\cdot IR_{22}\cdot IR_{21}$

Табела 1.г Сигнали дужина инструкција

bezadr1=PUSH+POP+RTS+RTI

branch2=BZ

adr2=(LD+ST+ADD+AND+ASR+JADR)·(regdir+regind+postdect+preincr)

jmp3=JMP+JSR

adr3=(LD+ST+ADD+AND+ASR+JADR)·regindpom

Читање инструкције (слика 1.а)

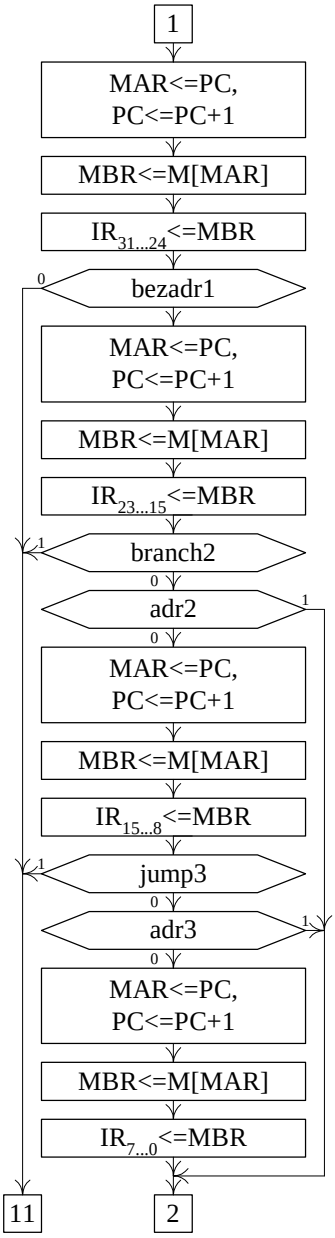
Инструкција се чита из меморије почев од адресе на коју указују тренутка вредност регистра програмског бројача $PC_{15...0}$. То се реализује тако што се чита бајт по бајт и после сваког прочитаног бајта садржај регистра $PC_{15...0}$ се инкрементира. Прочитани бајтови инструкције се смештају у прихватни регистар инструкције $IR_{31...0}$. У зависности од типа инструкције читају се један, два, три или четири бајта и смештају у разреде $IR_{31...24}$, $IR_{23...16}$, $IR_{15...8}$ и $IR_{7...0}$. Обавезно се чита први бајт инструкције. У оквиру тога се, најпре, садржај регистра $PC_{15...0}$ пребацује у регистар $MAR_{15...0}$ и инкрементира садржај регистра $PC_{15...0}$. Затим се из меморије M са адресе која се налази у регистру $MAR_{15...0}$ чита бајт и уписује у регистар $MBR_{7...0}$. Садржај регистра $MBR_{7...0}$ се, на крају, пребацује у разреде $IR_{31...24}$.

Сада се врши провера сигнала логичког услова дужине инструкције **bezadr1**. Уколико је његова вредност 1, ради се о безадресној инструкцији чија је дужина један бајт. Тиме је завршена фаза читање инструкције и прелази се на корак 11 и фазу

извршавање операција (слика 1.в). Уколико је његова вредност 0, ради се о некој од инструкција чија дужина може да буде два, три или четири бајта. Зато се у овом случају, најпре, чита други бајт инструкције и уписује у разреде $IR_{23...16}$. Читање другог бајта инструкције се реализује на сличан начин као и читање првог бајта инструкције.

Затим се врши провера сигнала логичког услова дужине инструкције **branch2**. Уколико је његова вредност 1, ради се о инструкцији условног скока чија је дужина два бајта. Тиме је завршена фаза читање инструкције и прелази се на корак 11 и фазу извршавање операција (слика 1.в). Уколико је његова вредност 0, врши се провера сигнала логичког услова дужине инструкције **adr2**. Уколико је његова вредност 1, ради се о адресној инструкцији чија је дужина два бајта. Тиме је завршена фаза читање инструкције и прелази се на корак 2 и фазу формирање адресе и читање операнда (слика 1.б). Уколико је његова вредност 0, ради се о некој од инструкција чија дужина може да буде три или четири бајта. Зато се у овом случају, најпре, чита трећи бајт инструкције и уписује у разреде $IR_{15...8}$. Читање трећег бајта инструкције се реализује на сличан начин као и читање првог бајта инструкције.

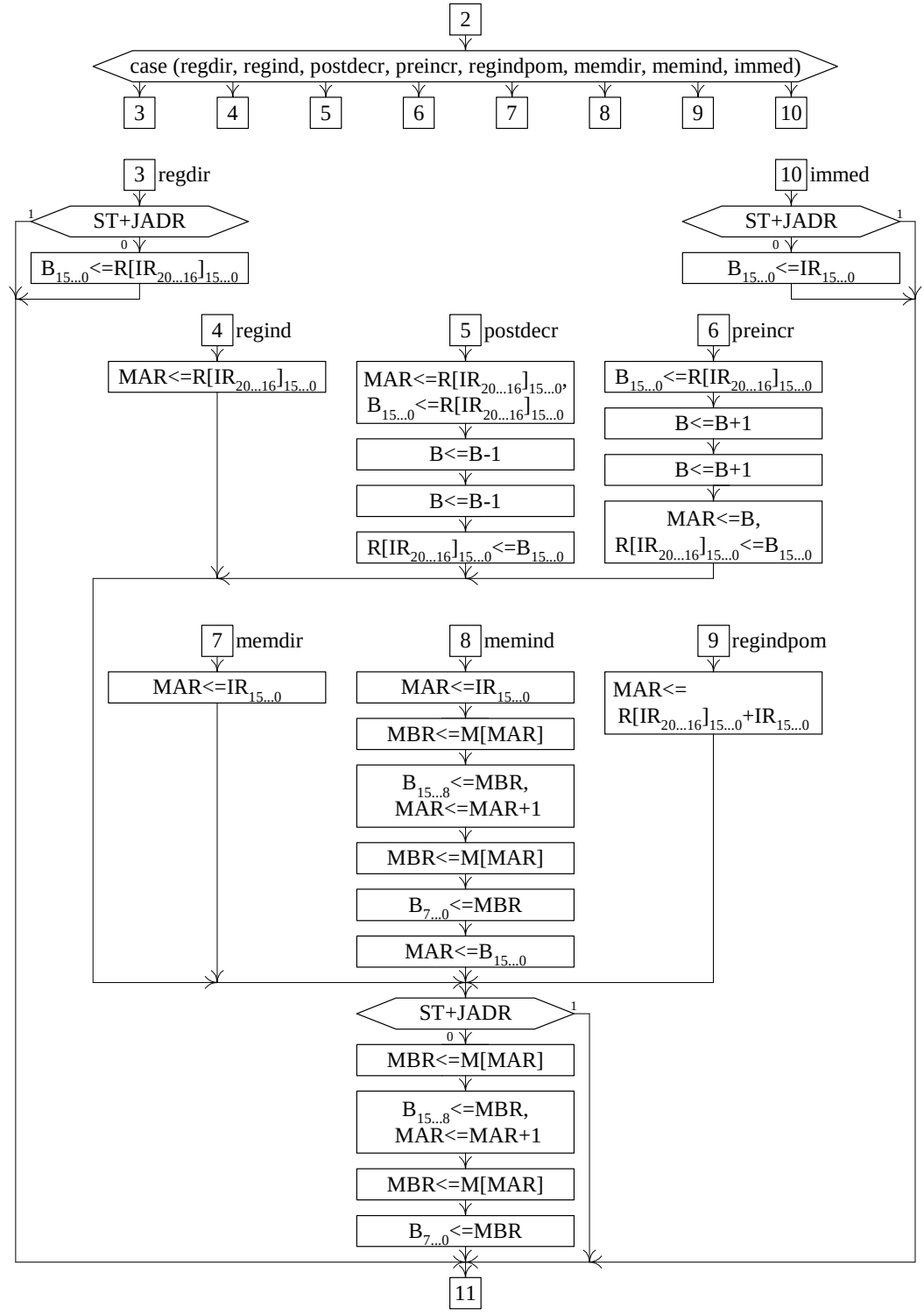
Сада се врши провера сигнала логичког услова дужине инструкције **jump3**. Уколико је његова вредност 1, ради се о инструкцији безусловног скока чија је дужина три бајта. Тиме је завршена фаза читање инструкције и прелази се на корак 11 и фазу извршавање операција (слика 1.в). Уколико је његова вредност 0, врши се провера сигнала логичког услова дужине инструкције **adr3**. Уколико је његова вредност 1, ради се о адресној инструкцији чија је дужина три бајта. Тиме је завршена фаза читање инструкције и прелази се на корак 2 и фазу формирање адресе и читање операнда (слика 1.б). Уколико је његова вредност 0, ради се о некој од адресних инструкција чија је дужина четири бајта. Зато се у овом случају чита четврти бајт инструкције и уписује у разреде $IR_{7...0}$. Читање четвртог бајта инструкције се реализује на сличан начин као и читање првог бајта инструкције. Тиме је завршена фаза читање инструкције и прелази се на корак 2 и фазу формирање адресе и читање операнда (слика 1.б).



Слика 1.а Дијаграм тока – фаза читање инструкције

Формирање адресе и читање операнда (слика 1.б)

Формирање адресе и читање операнда се реализује само за адресне инструкције и то од корака 2. У овом кораку се реализује вишеструки условни скок на један од корака 3 до 10 у зависности од тога који од сигнала логичких услова начина адресирања **regdir** до **immed** има вредност 1. Сигнали логичких начина адресирања услова **regdir** до **immed** (табела 1.в) су бинарно кодирани разредима $IR_{23.22.21}$ прихватног регистра инструкције и само један од њих може да има вредност 1.



Слика 1.б Дијаграм тока – фаза формирање адресе и читање операнда

Уколико вредност 1 има сигнал логичког услова **regdir**, ради се о регистарском директном адресирању и прелазу са корака 2 на корак 3. Операнд је тада у регистру опште намене R_i на адреси која се налази у разредима $IR_{20...16}$. Садржај регистра R_i се пребацује у регистар $B_{15...0}$ за све операције сем операција ST и JADR. Због тога се овде врши провера да ли један од сигнала логичких услова операција **ST** или **JADR** има вредност 1. Уколико оба сигнала **ST** и **JADR** имају вредност 0, регистар опште намене $R_и$ се пребацује у регистар $B_{15...0}$. Тиме је завршена фаза формирање адресе и читање

операнда и прелази се на корак 11 и фазу извршавање операција (слика 1.в). Уколико један од сигнала **ST** и **JADR** има вредност 1, одмах се завршава фаза формирање адресе и читање операнда и прелази се на корак 11 и фазу извршавање операција (слика 2.в).

Уколико вредност 1 има сигнал логичког услова **regind**, ради се о регистарском индиректном адресирању и прелазу са корака 2 на корак 4. Операнд је тада у меморији на адреси која се налази у регистру опште намене R_i, а регистар R_i је на адреси која се налази у разредима $IR_{20...16}$. Селектовани регистар R_i се пребацује у регистар $MAR_{15...0}$ и прелази на корак почев од кога се за све операције, сем операција ST и JADR, чита операнд и смешта у регистар $B_{15...0}$. Због тога се овде врши провера да ли један од сигнала логичких услова операција **ST** или **JADR** има вредност 1. Уколико оба сигнала **ST** и **JADR** имају вредност 0, из меморије M се са адресе која се налази у регистру $MAR_{15...0}$, најпре, чита бајт и уписује у регистар $MBR_{7...0}$, а затим се садржај регистра $MBR_{7...0}$ пребацује у разреде $B_{15...8}$ и врши инкрементирање садржаја регистра $MAR_{15...0}$. Како је операнд ширине два бајта а ширина меморијске речи један бајт, потребно је из меморије прочитати још један бајт. С тога се из меморије M са адресе која се налази у регистру $MAR_{15...0}$, најпре, чита бајт и уписује у регистар $MBR_{7...0}$, а затим се садржај регистра $MBR_{7...0}$ пребацује у разреде $B_{7...0}$. Треба уочити да је први бајт операнда прочитан из ниже меморијске локације старији бајт операнда и да је стога уписан у старијих осам разреда регистра $B_{15...8}$ и да је други бајт операнда прочитан из више меморијске локације млађи бајт операнда и да је стога уписан у млађих осам разреда регистра $B_{7...0}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 11 и фазу извршавање операција (слика 1.в). Уколико један од сигнала **ST** и **JADR** има вредност 1, прелази се на корак 11 и фазу извршавање операција (слика 1.в).

Уколико вредност 1 има сигнал логичког услова **postdecr** или **preincr**, ради се о регистарском индиректном са постдекрементирањем адресирању или регистарском индиректном са преинкрементирањем адресирању и прелазу са корака 2 на корак 5 или 6, респективно. У оба случаја је, као и код регистарског индиректног адресирања, операнд у меморији на адреси која се налази у регистру опште намене R_i, а регистар R_i је на адреси која се налази у разредима $IR_{20...16}$. Селектовани регистар R_i се пребацује у регистар $MAR_{15...0}$ и прелази се на корак почев од кога се, на већ описани начин, за све операције, сем операција ST и JADR, чита операнд и смешта у регистар $B_{15...0}$. Разлика у односу на регистарско индиректно адресирање је само у томе да се код регистарског индиректног са постдекрементирањем адресирања садржај регистра R_i најпре пребаци у регистар $MAR_{15...0}$ а затим се декрементирањем умањи за 2, док се код регистарског индиректног са преинкрементирањем адресирања садржај регистра R_i најпре инкрементирањем увећа за 2 а затим пребаци у регистар $MAR_{15...0}$. Садржај регистра R_i се умањује или увећава за 2, јер се то чини за дужину операнда, чија је дужина 2 бајта, изражену у броју меморијских речи, чија је дужина 1 бајт. Треба уочити да је узето да су регистри опште намене реализовани као регистарски фајл и да не постоји могућност да се у оквиру регистарског фајла врши декрементирање и инкрементирање појединачних регистара. С тога је узето да се регистар R_i пребацује у регистар $B_{15...0}$, да се садржај регистра $B_{15...0}$ декрементира или инкрементира и да се добијена вредност уписује у регистар R_i.

Уколико вредност 1 има сигнал логичког услова **memdir**, ради се о меморијском директном адресирању и прелазу са корака 2 на корак 7. Операнд је тада у меморији на адреси која се налази у разредима $IR_{15...0}$. Стога се садржај разреда $IR_{15...0}$ пребацује у регистар $MAR_{15...0}$ и прелази на корак почев од кога се, на већ описани начин, за све операције, сем операција ST и JADR, чита операнд и смешта у регистар $B_{15...0}$. Тиме је

завршена фаза формирање адресе и читање операнда и прелази се на корак 11 и фазу извршавање операција (слика 1.в).

Уколико вредност 1 има сигнал логичког услова **memind**, ради се о меморијском индиректном адресирању и прелазу са корака 2 на корак 8. Операнд је тада у меморији, а адреса меморијске локације је у другој меморијској локацији чија се адреса налази у разредима $IR_{15...0}$. Стога се садржај разреда $IR_{15...0}$ пребацује у регистар $MAR_{15...0}$. Из меморије M се са адресе која се налази у регистру $MAR_{15...0}$, најпре, чита бајт и уписује у регистар $MBR_{7...0}$, а затим се садржај регистра $MBR_{7...0}$ пребацује у разреде $B_{15...8}$ и врши инкрементирање садржаја регистра $MAR_{15...0}$. Како је адреса ширине два бајта а ширина меморијске речи један бајт, потребно је из меморије прочитати још један бајт. С тога се из меморије M са адресе која се налази у регистру $MAR_{15...0}$, најпре, чита бајт и уписује у регистар $MBR_{7...0}$, а затим се садржај регистра $MBR_{7...0}$ пребацује у разреде $B_{7...0}$. Треба уочити да је први бајт прочитан из ниже меморијске локације старији бајт адресе и да је стога уписан у старијих осам разреда регистра $B_{15...8}$ и да је други бајт прочитан из више меморијске локације млађи бајт адресе и да је стога уписан у млађих осам разреда регистра $B_{7...0}$. Операнд је у меморији на адреси која се налази у разредима $B_{15...0}$. Стога се садржај регистра $B_{15...0}$ пребацује у регистар $MAR_{15...0}$ и прелази на корак почев од кога се, на већ описани начин, за све операције, сем операције ST и JADR, чита операнд и смешта у регистар $B_{15...0}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 11 и фазу извршавање операција (слика 1.в).

Уколико вредност 1 има сигнал логичког услова **regindpom**, ради се о регистарском индиректном адресирању са померајем. Операнд је тада у меморији на адреси која се добија сабирањем садржаја регистра опште немене R_i и помераја. Регистар опште намене R_i је на адреси која се налази у разредима $IR_{20...16}$, док се померај добија проширивањем знаком на 16 битну вредност разреда $IR_{15...8}$. Добијена адреса се пребацује у регистар $MAR_{15...0}$ и прелази на корак почев од кога се, на већ описани начин, за све операције, сем операције ST и JADR, чита операнд и смешта у регистар $B_{15...0}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 11 и фазу извршавање операција (слика 1.в).

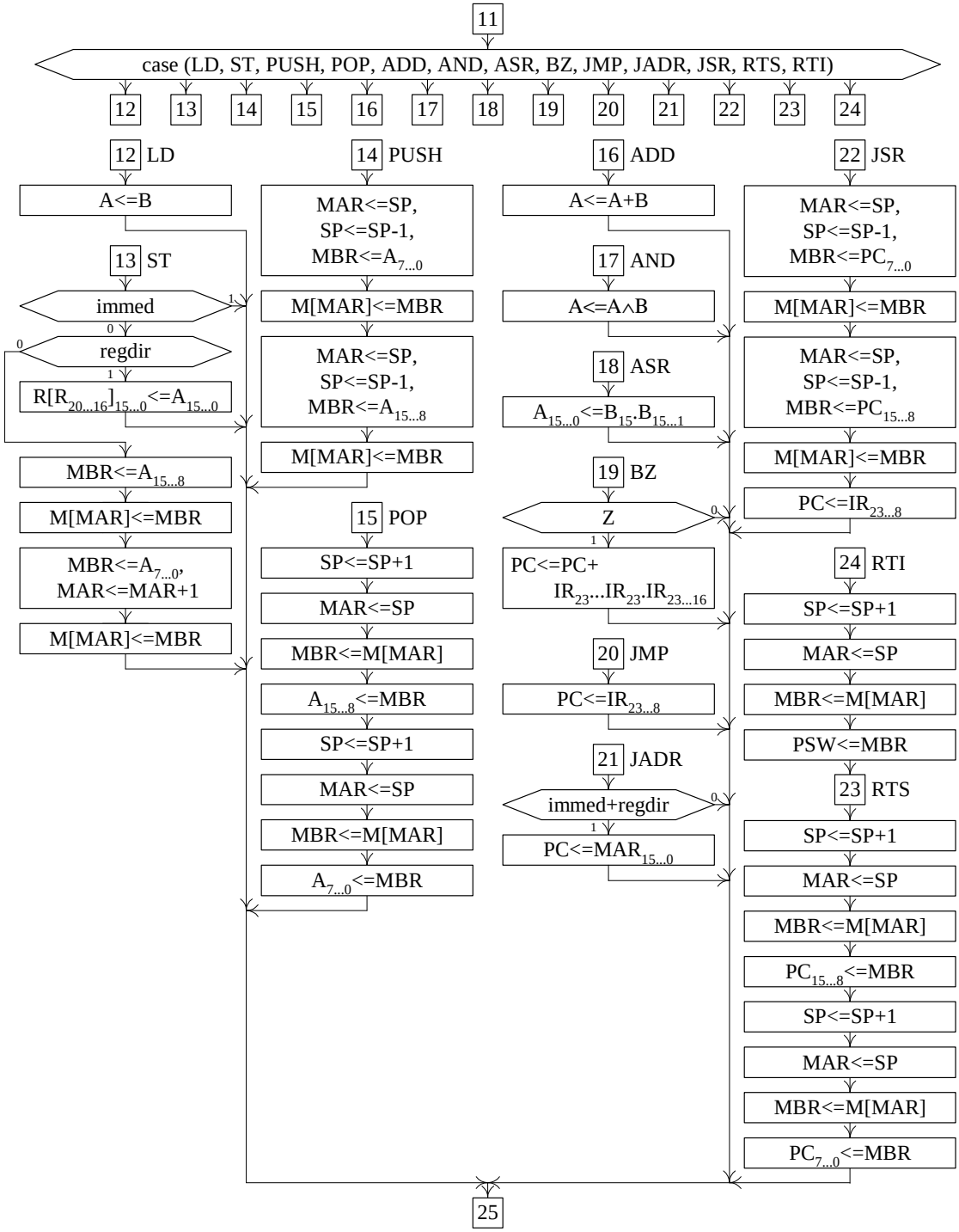
Уколико вредност 1 има сигнал логичког услова **immed**, ради се о непосредном адресирању и прелазу са корака 2 на корак 10. Операнд се тада налази у разредима $IR_{15...0}$ чији се садржај пребацује у регистар $B_{15...0}$ за све операције сем операција ST и JADR. Због тога се овде врши провера да ли један од сигнала логичких услова операција **ST** или **JADR** има вредност 1. Уколико оба сигнала **ST** и **JADR** имају вредност 0, садржај разреда $IR_{15...0}$ се пребацује у регистар $B_{15...0}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 11 и фазу извршавање операција (слика 1.в). Уколико један од сигнала **ST** и **JADR** има вредност 1, завршава се фаза формирање адресе и читање операнда и прелази се на корак 11 и фазу извршавање операција (слика 1.в).

Извршавање операција (слика 1.в)

Извршавање операција се реализује за све инструкције и то од корака 11. У овом кораку се реализује вишеструки условни скок на један од корака 12 до 24 у зависности од тога који од сигнала логичких услова операција **LD** до **RTI** има вредност 1. Сигнали логичких услова операција **LD** до **RTI** (табела 1.б) су бинарно кодирани разредима $IR_{31...24}$ прихватног регистра инструкције и само један од њих може да има вредност 1.

Уколико вредност 1 има сигнал логичког услова **LD**, садржај регистра $B_{15...0}$ се пребацује у регистар акумулатора $A_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 1.г).

Уколико вредност 1 има сигнал логичког услова **ST**, садржај регистра акумулатора $A_{15...0}$ се пребацује у регистар опште намене R_i уколико је специфицирано директно регистарско адресирање или у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$ уколико је специфицирано неко од меморијских адресирања. Непосредно адресирање није дозвољено за одредишни операнд и зато се узима да се у случају да се у инструкцији ST појави непосредно адресирање, фаза извршавања ове операције завршава, чиме се ова инструкција претвара у инструкцију без дејства. С тога се проверава вредност сигнала **immed**. Уколико сигнал **immed** има вредност 1, специфицирано је непосредно адресирање, па се фаза извршавања операције завршава и прелази на корак 25 и фазу опслуживање прекида (слика 1.г). Уколико сигнал **immed** има вредност 0, проверава се вредност сигнала **regdir**. Уколико сигнал **regdir** има вредност 1, регистарско директно адресирање је специфицирано, па се садржај регистра $A_{15...0}$ уписује у регистар опште намене R_i чија се адреса налази у разредима $IR_{20...16}$ и прелази на корак 25 и фазу опслуживање прекида (слика 1.г). Уколико сигнал **regdir** има вредност 0, неко од меморијских адресирања је специфицирано, па се садржај регистра $A_{15...0}$ уписује у меморијску локацију чија се адреса налази у регистру $MAR_{15...0}$. У оквиру тога се, најпре, старији бајт регистра $A_{15...8}$ пребацује у регистар $MBR_{7...0}$ и уписује у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$. Затим се, пошто се млађи бајт регистра $A_{7...0}$ пребаци у регистар $MBR_{7...0}$ и садржај регистра $MAR_{15...0}$ инкрементира, садржај регистра $MBR_{7...0}$ уписује у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 1.г).



Слика 1.в Дијаграм тока – фаза извршавање операција

Ukoliko вредност 1 има сигнал логичког услова **PUSH**, садржај регистра акумулатора $A_{15...0}$ се ставља на стек. Најпре се пребацује садржаја регистар $SP_{15...0}$ у регистар $MAR_{15...0}$, декрементира садржај регистра $SP_{15...0}$ и пребацује млађи бајт регистра $A_{7...0}$ у регистар $MBR_{7...0}$. Затим се садржај регистра $MBR_{7...0}$ уписује у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$. После тога се поново пребацује садржаја регистар $SP_{15...0}$ у регистар $MAR_{15...0}$, декрементира садржај регистра $SP_{15...0}$ и пребацује старији бајт регистра $A_{15...8}$ у регистар $MBR_{7...0}$. На крају се садржај регистра $MBR_{7...0}$ уписује у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$. Треба уочити да стек расте према нижим локацијама и да регистар $SP_{15...0}$

указује на прву слободну локацију. С тога се садржај регистра $SP_{15...0}$ прво пребацује у регистар $MAR_{15...0}$ и после тога декрементира. Такође треба уочити да се на стек прво ставља млађи а потом старији бајт регистра $A_{15...0}$. То је последица начина смештања дво бајтовских величина у меморију, који предвиђа да старији бајт буде на нижој, а млађи бајт на вишој адреси. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 1.г).

Уколико вредност 1 има сигнал логичког услова **POP**, садржајем са стека се рестаурира регистар акумулатора $A_{15...0}$. Најпре се садржај регистра $SP_{15...0}$ инкрементира и пребацује у регистар $MAR_{15...0}$. Затим се из меморијске локације са адресе која се налази у регистру $MAR_{15...0}$ чита садржај и уписује у регистар $MBR_{7...0}$. На крају се садржај регистра $MBR_{7...0}$ уписује у старији бајт регистра $A_{15...8}$. После тога се поново садржај регистра $SP_{15...0}$ инкрементира и пребацује у регистар $MAR_{15...0}$. Затим се из меморијске локације са адресе која се налази у регистру $MAR_{15...0}$ чита садржај и уписује у регистар $MBR_{7...0}$. На крају се садржај регистра $MBR_{7...0}$ уписује у млађи бајт регистра $A_{7...0}$. Треба уочити да стек расте према нижим локацијама и да регистар $SP_{15...0}$ указује на прву слободну локацију. С тога се приликом читања садржаја са стека, прво инкрементира садржај регистра $SP_{15...0}$ и после тога пребацује у регистар $MAR_{15...0}$. Такође треба уочити да се са стека прво скида старији а потом млађи бајт регистра $A_{15...0}$. То је последица начина смештања дво бајтовских величина у меморију, који предвиђа да старији бајт буде на нижој, а млађи бајт на вишој адреси. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 1.г).

Уколико вредност 1 има сигнал логичког услова **ADD**, сабирају се садржаји регистара $A_{15...0}$ и $B_{15...0}$ и резултат уписује у регистар $A_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 1.г).

Уколико вредност 1 има сигнал логичког услова **AND**, логичка операција логички производ се реализује над садржајима регистара $A_{15...0}$ и $B_{15...0}$ и резултат уписује у регистар $A_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 1.г).

Уколико вредност 1 има сигнал логичког услова **ASR**, садржај регистра $B_{15...0}$ се аритметички помера удесно за једно место и уписује у регистар $A_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 1.г).

Уколико вредност 1 има сигнал логичког услова **BZ**, условни скок на основу вредности сигнала логичког услова **Z** се реализује. Сигнал **Z** има вредност 1 уколико је резултат задње извршене инструкције 0 и вредност 0 уколико резултат задње извршене инструкције није 0. Уколико сигнал **Z** има вредност 0, услов за скок није испуњен. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 1.г). Уколико сигнал **Z** има вредност 1, услов за скок је испуњен, па се сабирају садржај регистра $PC_{15...0}$, који представља текућу вредност регистра програмског бројача $PC_{15...0}$, и садржај разреда $IR_{23...16}$ проширен знаком на 16 бита, који представља померај, и добијена вредност, која представља адресу скока, уписује у регистар $PC_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 1.г).

Уколико вредност 1 има сигнал логичког услова **JMP**, безусловни скок се реализује. Садржај разреда $IR_{15...0}$, који представља адресу скока, уписује се у регистар $PC_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 1.г).

Уколико вредност 1 има сигнал логичког услова **JADR**, безусловни скок на срачунату адресу се реализује, под условом да је задато неко од меморијских адресирања. У оквиру тога се садржај разреда $MAR_{15...0}$, који представља срачунату адресу скока задату неким од меморијских адресирања, уписује у регистар $PC_{15...0}$. Непосредно адресирање и регистарско директно адресирање нису дозвољени јер не дају адресу меморијске локације и зато се узима да се у случају да се у инструкцији JADR појави непосредно адресирање или регистарско директно адресирање, фаза извршавања ове операције завршава, чиме се ова инструкција претвара у инструкцију без дејства. С тога се проверава да ли вредност 1 имају сигнал **immed** или сигнал **regdir**. Уколико да, специфицирано је непосредно адресирање или регистарско директно адресирање, па се фаза извршавања операције завршава и прелази на корак 25 и фазу опслуживање прекида (слика 1.г). Уколико не, специфицирано је неко од меморијских адресирања, па се садржај разреда $MAR_{15...0}$ уписује у регистар $PC_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 1.г).

Уколико вредност 1 има сигнал логичког услова **JSR**, скок на потпрограм се реализује. У оквиру тога се садржај регистра $PC_{15...0}$ ставља на стек. Најпре се пребацује садржаја регистра $SP_{15...0}$ у регистар $MAR_{15...0}$, декрементира садржај регистра $SP_{15...0}$ и пребацује млађи бајт регистра $PC_{7...0}$ у регистар $MBR_{7...0}$. Затим се садржај регистра $MBR_{7...0}$ уписује у меморијску локацију чија се адреса налази у регистру $MAR_{15...0}$. После тога се поново пребацује садржај регистра $SP_{15...0}$ у регистар $MAR_{15...0}$, декрементира садржај регистра $SP_{15...0}$ и пребацује старији бајт регистра $PC_{15...8}$ у регистар $MBR_{7...0}$. На крају се садржај регистра $MBR_{7...0}$ уписује у меморијску локацију чија се адреса налази у регистру $MAR_{15...0}$. Треба уочити да стек расте према нижим локацијама и да регистар $SP_{15...0}$ указује на прву слободну локацију. С тога се садржај регистра $SP_{15...0}$ прво пребацује у регистар $MAR_{15...0}$ и после тога декрементира. Такође треба уочити да се на стек прво ставља млађи а потом старији бајт регистра $PC_{15...0}$. То је последица начина смештања дво бајтовских величина у меморију, који предвиђа да старији бајт буде на нижој, а млађи бајт на вишој адреси. Затим се садржај разреда $IR_{15...0}$, који представља адресу потпрограма, уписује у регистар $PC_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 1.г).

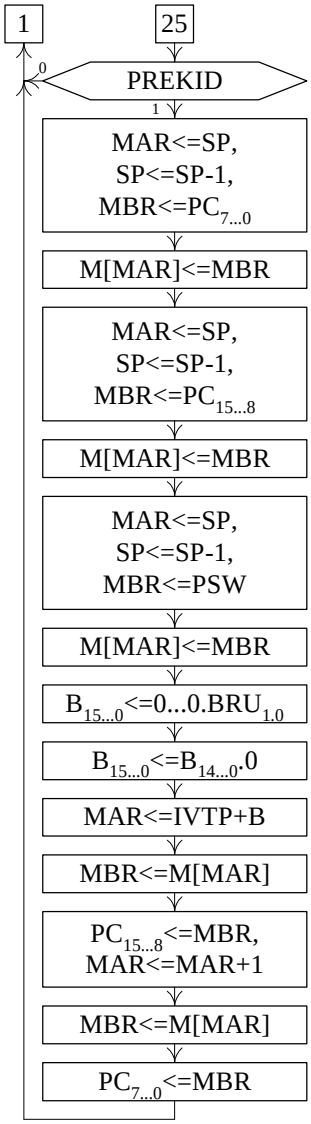
Уколико вредност 1 има сигнал логичког услова **RTI**, повратак из прекидне рутине се реализује. У оквиру тога се садржајима са стека рестаурирају садржаји регистара $PSW_{7...0}$ и $PC_{15...0}$. Најпре се садржај регистра $SP_{15...0}$ инкрементира и пребацује у регистар $MAR_{15...0}$. Затим се из меморијске локације са адресе која се налази у регистру $MAR_{15...0}$ чита садржај и уписује у регистар $MBR_{7...0}$. На крају се садржај регистра $MBR_{7...0}$ уписује у регистар $PSW_{7...0}$. На исти начин се са стека читају још два бајта и уписују најпре у старији а потом у млађи бајт регистра $PC_{15...0}$. Треба уочити да стек расте према нижим локацијама и да регистар $SP_{15...0}$ указује на прву слободну локацију. С тога се приликом читања садржаја са стека, прво инкрементира садржај регистра $SP_{15...0}$ и после тога пребацује у регистар $MAR_{15...0}$. Такође треба уочити да се са стека прво скида старији а потом млађи бајт регистра $PC_{15...0}$. То је последица начина смештања дво бајтовских величина у меморији, који предвиђа да старији бајт буде на нижој, а млађи бајт на вишој адреси. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 1.г).

Уколико вредност 1 има сигнал логичког услова **RTS**, повратак из потпрограма се реализује. У оквиру тога се садржајем са стека рестаурира садржај регистра $PC_{15...0}$. Ово се реализује на идентичан као и у случају инструкције **RTI**. Тиме је завршена фаза

извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 1.г).

Опслуживање прекида (слика 1.г)

Опслуживање прекида се реализује почев од корака 25. Уколико сигнал логичког услова **PREKID** има вредност 0, у току извршавања претходних фаза није дошло до генерисања сигнала прекида, па се фаза опслуживање прекида завршава и прелази се на корак 1 и фазу читање инструкције (слика 1.а). Уколико сигнал **PREKID** има вредност 1, у току извршавања претходних фаза дошло је до генерисања сигнала прекида, па се прелази на кораке у оквиру којих се на стек најпре смештају садржаји регистара $PC_{15...0}$ и $PSW_{7...0}$ и потом утврђује адреса прекидне рутине и уписује у регистар $PC_{15...0}$. Чување садржаја регистра $PC_{15...0}$ на стеку се реализује на идентичан начин као и у случају инструкције JSR. На исти начин се на стек ставља и садржај регистра $PSW_{7...0}$. Адресе прекидних рутина се налазе у улазима табеле са адресама прекидних рутина. Број улаза у табелу је дат садржајем регистра $BRU_{1.0}$, а почетна адреса табеле садржајем регистра $IVTP_{15...0}$. Најпре се садржај регистра $BRU_{1.0}$ проширен нулама до дужине 16 бита пребацује у регистар $B_{15...0}$, па се садржај регистра $B_{15...0}$ померањем улево за једно место множи са два. Тиме се број улаза претвара у померај. Потом се сабирањем садржаја регистара $IVTP_{15...0}$ и $B_{15...0}$ и смештањем њихове суме у регистар $MAR_{15...0}$ добија адреса на којој се налази адреса прекидне рутине. Са те и следеће адресе из меморије се читају два бајта и уписују у старијих и млађих осам разреда регистра $PC_{15...0}$. Фаза опслуживање прекида је тиме завршена и прелази се на корак 1 и фазу читање инструкције (слика 1.а).



Слика 1.г Дијаграм тока – фаза опслуживање прекида

Треба уочити да се јављају две ситуације везане за вредност регистра $PC_{15...0}$ по завршетку фазе опслуживање прекида и преласка на корак 1 и фазу читање инструкције (слика 1.а). Уколико је сигнал PREKID имао вредност 0, у регистру $PC_{15...0}$ је адреса прве следеће инструкције после инструкције која је извршена. Уколико је сигнал PREKID имао вредност 1, у регистру $PC_{15...0}$ је адреса прве инструкције прекидне рутине.

2 ЗАДАТАК 2

Посматра се део рачунара који чине меморија и процесор.

Меморија је капацитета 2^{16} бајтова. Ширина меморијске речи је 1 бајт.

Процесор је са једноадресним форматом инструкција. Подаци су целобројне величине без знака дужине 2 бајта. Подаци у меморији заузимају две суседне меморијске локације, при чему се млађи бајт налази на нижој а старији бајт на вишој адреси.

У процесору постоји регистар програмског бројача PC дужине 2 бајта, адресни регистар меморије MAR дужине 2 бајта, прихватни регистар податка меморије MBR дужине 1 бајт, прихватни регистар инструкције IR дужине 3 бајта, регистар акумулатора A дужине 2 бајта, прихватни регистар податка B дужине 2 бајта, регистри опште наме R0 и R3 дужине 2 бајта, програмска статусна реч PSW дужине 1 бајт, указивач на врх стека SP дужине 2 бајта, регистар броја улаза у табелу са адресам прекидних рутина BRU дужине 2 бита и указивач на табелу са адресама прекидних рутина IVTP дужине 2 бајта. Инструкције су дужине 1, 2 или 3 бајта.

Битови 7, 6, 5 и 4 првог бајта инструкције су 0000 за све инструкције скока, при чему бит 3 првог бајта инструкције има вредност 0 за инструкција условног скока и вредност 1 за инструкције безусловног скока. Инструкције скока су инструкција условног скока уколико резултат није нула (BNZ). Битовима 2 до 0 првог бајта инструкције специфицира се код операције за инструкције условног скока. На основу тога је за инструкцију BNZ усвојен код операције 00000000. Инструкција BNZ се реализује као релативни скок у односу на текућу вредност програмског бројача PC, а померај је 8 битна целобројна величина са знаком дата 2. бајтом инструкције. Дужина инструкција је 2 бајта. Инструкције безусловног скока су инструкција безусловног скока (JMP) и скока на потпрограм (JSR). Битовима 2 до 0 првог бајта инструкције се специфицира код операције за инструкције безусловног скока. На основу тога су за инструкције JMP и JSR усвојени кодови операција 00001000 и 00001001, респективно. Инструкције JMP и JSR се реализују као апсолутни скокови, а адреса скока је дата 2. и 3. бајтом инструкције, при чему је млађи бајт адресе скока дат другим бајтом инструкције а старији бајт адресе скока трећим бајтом инструкције. Дужина инструкција је 3 бајта.

Битови 7, 6, 5 и 4 првог бајта инструкције су 1111 за безадресне инструкције. Безадресне инструкције су инструкције стављања садржаја акумулатора на стек (PUSH), пуњења акумулатора садржајем са стека (POP), повратка из потпрограма (RTS) и повратка из прекидне рутине (RTI). Битовима 3 до 0 првог бајта инструкције специфицира се код операције за безадресне инструкције. На основу тога су за инструкције PUSH, POP, RTS и RTI усвојени кодови операција 11110000, 11110001, 11110010 и 11110011, респективно. Дужина инструкција је 1 бајт.

Битови 7, 6, 5 и 4 првог бајта инструкције у опсегу вредности 0001 до 1110 специфицирају код операције за адресне инструкције. Адресне инструкције су инструкција преноса у акумулатор (LD), инструкција преноса из акумулатора (ST), аритметичка инструкција одузимања (SUB), логичка инструкција логичка сума (OR), инструкција аритметичког померања улево за једно место (ASL) и инструкција безусловног скока на срачунату адресу (JADR). У инструкцији ST није дозвољено непосредно адресирање, у инструкцији JADR није дозвољено регистарско директно и непосредно адресирање, па уколико се јаве ова адресирања у овим инструкцијама,

инструкције треба да буду без дејства, а инструкција ASL резултат померања смешта у регистар A. На основу тога су за инструкције LD, ST, SUB, OR, ASL и JADR усвојени кодови операција 0001, 0010, 0011, 0100, 0101 и 0110, респективно. Дужина инструкција је 1, 2, или 3 бајта и зависи од специфицираног начина адресирања.

Начини адресирања су специфицирани битовима 3 и 2 првог бајта инструкције и то на следећи начин: 00-непосредно адресирање (immed), 01-меморијско директно адресирање (memdir), 10-регистарско индиректно са померајем адресирање (regindpom) и 11-регистарско директно адресирање (regdir). Код непосредног адресирања 16 битни операнд је дат другим и трећим бајтом инструкције, при чему је млађи бајт операнда дат другим а старији бајт трећим бајтом. Битови 1 и 0 првог бајта инструкције се не користе. Дужина инструкција је 3 бајта. Код меморијског директног адресирања 16 битна адреса меморијске локације је дата другим и трећим бајтом инструкције, при чему је млађи бајт адресе дат другим а старији бајт трећим бајтом. Битови 1 и 0 првог бајта инструкције се не користе. Дужина инструкција је 3 бајта. Код регистарског индиректног адресирања са померајем 8 битни померај је целобројна величина са знаком у другом комплементу дата другим бајтом инструкције. Битови 1 и 0 првог бајта инструкције се користе за адресирање једног од регистара опште намене R0 до R3. Дужина инструкција је 2 бајта. Код регистарског директног адресирања битови 1 и 0 првог бајта инструкције се користе за адресирање једног од регистара опште намене R0 до R3. Дужина инструкција је 1 бајт.

Стек расте према нижим меморијским локацијама, а регистар SP указује на задњу заузету меморијску локацију.

Захтеви за прекид долазе од 4 улазно/излазна уређаја по линијама означеним од 0 до 3. По линији 0 стиже захтев за прекид најнижег, а по линији 3 највишег приоритета. Број линије највишег приоритета по којој је стигао захтев за прекид налази се у бинарном облику у регистру BRU дужине 2 разреда. Адресе прекидних рутина 4 улазно/излазна уређаја који по линијама означеним од 0 до 3 шаљу захтеве за прекид налазе се у улазима 0 до 3 табеле са адресама прекидних рутина. Адресе дужине 16 бита заузимају по две суседне меморијске локације, при чему се млађи бајт налази на нижој а старији бајт на вишој адреси Садржај регистра BRU представља број улаза у табелу са адресом прекидних рутина. Почетна адреса табеле са адресама прекидних рутина се налази у регистру IVTP дужине 2 бајта. У оквиру хардверског дела опслуживања захтева за прекид на стек са стављају само регистри PC и PSW.

Нацртати и објаснити дијаграм тока фаза извршавања инструкције и то: фазе читање инструкције, фазе формирање адресе и читања операнда, фазе извршавање операција LD, ST, PUSH, POP, SUB, OR, ASL, BNZ, JMP, JADR, JSR, RTS и RTI и фазе опслуживање захтева за прекид.

РЕШЕЊЕ

Дијаграм тока извршавања инструкције је дат на сликама 2.а, 2.б, 2.в и 2.г. Извршавање инструкције се састоји из четири фазе: читање инструкције (слика 2.а), формирање адресе и читање операнда (слика 2.б), извршавање операција (слика 2.в) и опслуживање прекида (слика 2.г). Дијаграм тока је дат у сагласности са форматима инструкција (табела 2.а), при чему први, други и трећи бајт инструкције се смештају у разреде 23 до 16, 15 до 8 и 7 до 0 прихватног регистра инструкције IR. У њему се користе сигнали логичких услова операција (табела 2.б), начина адресирања (табела 2.в) и дужина инструкција (табела 2.г) који се формирају на основу разреда 23 до 16 прихватног регистра инструкције. У дијаграму тока се користе и сигнали логичких услова **Z** и **PREKID**. Сигнал **Z** представља један од индикатора програмске статусне речи PSW који вредностима 1 и 0 указује да ли је резултат задње извршене инструкције

нула или различит од нуле, респективно, док сигнал **PREKID** вредностима 1 и 0 указује да ли је током извршавања инструкције дошло или није дошло до генерисања прекида, респективно. Генерисање сигнала **Z** и **PREKID** је ван оквира овог задатка и није приказано.

Табела 2.а Формати инструкција

Instrukcija uslovnog skoka BNZ

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
0	0	0	**0**	0	OC			disp							

Instrukcije bezuslovnog skoka JMP i JSR

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	OC			low								high							

Bezadresne instrukcije PUSH, POP, RTS i RTI

23	22	21	20	19	18	17	16
1	1	1	1	OC			

Adresne instrukcije LD, ST, SUB, OR, ASL i JADR sa neposrednim i memorijskim direktnim adresiranjima*

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC				0	x	x	x	low								high							

Adresne instrukcije LD, ST, SUB, OR, ASL i JADR sa registarskim indirektnim sa pomerajem adresiranjem*

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
OC				1	0	reg		disp							

Adresne instrukcije LD, ST, SUB, OR, ASL i JADR sa registarskim direktnim adresiranjem*

23	22	21	20	19	18	17	16
OC				1	1	reg	

*U polju OC nisu dozvoljene vrednosti 0000 i 1111 koje se koriste kao kodovi izuzetka za instrukcije skoka i bezadresne instrukcije

Табела 2.б Сигнали операција

$BNZ=\overline{IR_{23}}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot\overline{IR_{20}}\cdot\overline{IR_{19}}\cdot\overline{IR_{18}}\cdot\overline{IR_{17}}\cdot\overline{IR_{16}}$

$JMP=\overline{IR_{23}}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot\overline{IR_{20}}\cdot IR_{19}\cdot\overline{IR_{18}}\cdot\overline{IR_{17}}\cdot\overline{IR_{16}}$

$JSR=\overline{IR_{23}}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot\overline{IR_{20}}\cdot IR_{19}\cdot\overline{IR_{18}}\cdot\overline{IR_{17}}\cdot IR_{16}$

$PUSH=IR_{23}\cdot IR_{22}\cdot IR_{21}\cdot IR_{20}\cdot\overline{IR_{19}}\cdot\overline{IR_{18}}\cdot\overline{IR_{17}}\cdot\overline{IR_{16}}$

$POP=IR_{23}\cdot IR_{22}\cdot IR_{21}\cdot IR_{20}\cdot\overline{IR_{19}}\cdot\overline{IR_{18}}\cdot\overline{IR_{17}}\cdot IR_{16}$

$RTS=IR_{23}\cdot IR_{22}\cdot IR_{21}\cdot IR_{20}\cdot\overline{IR_{19}}\cdot\overline{IR_{18}}\cdot IR_{17}\cdot\overline{IR_{16}}$

$RTI=IR_{23}\cdot IR_{22}\cdot IR_{21}\cdot IR_{20}\cdot\overline{IR_{19}}\cdot\overline{IR_{18}}\cdot IR_{17}\cdot IR_{16}$

$LD=\overline{IR_{23}}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot IR_{20}$

$ST=\overline{IR_{23}}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot IR_{20}$

$SUB=\overline{IR_{23}}\cdot\overline{IR_{22}}\cdot IR_{21}\cdot IR_{20}$

$OR=\overline{IR_{23}}\cdot IR_{22}\cdot\overline{IR_{21}}\cdot\overline{IR_{20}}$

$ASL=\overline{IR_{23}}\cdot IR_{22}\cdot\overline{IR_{21}}\cdot IR_{20}$

$JADR=\overline{IR_{23}}\cdot IR_{22}\cdot IR_{21}\cdot\overline{IR_{20}}$

Табела 2.в Сигнали начина адресирања

$immed=\overline{IR_{19}}\cdot\overline{IR_{18}}$

$memdir=\overline{IR_{19}}\cdot IR_{18}$

$regindpom=IR_{19}\cdot\overline{IR_{18}}$

$regdir=IR_{19}\cdot IR_{18}$

Табела 2.г Сигнали дужина инструкција

bezadr1=PUSH+POP+RTS+RTI

adr1=(LD+ST+SUB+OR+ASL+JADR)·regdir

branch2=BNZ

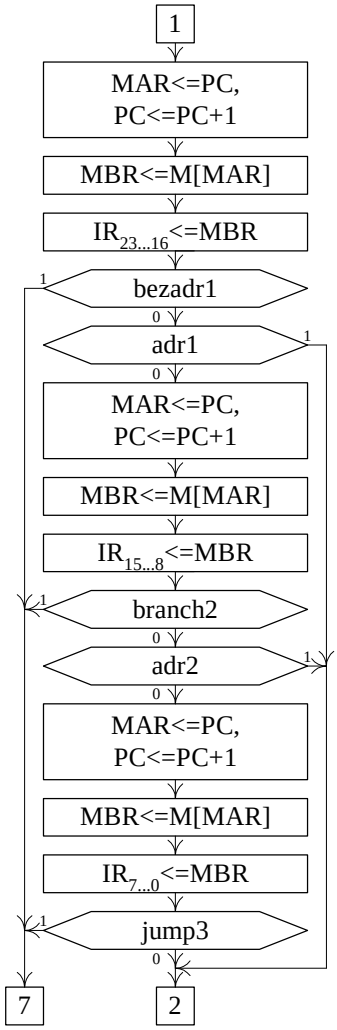
adr2=(LD+ST+SUB+OR+ASL+JADR)·regindpom

jmp3=JMP+JSR

adr3=(LD+ST+SUB+OR+ASL+JADR)·(immed+memdir)

Читање инструкције (слика 2.а)

Инструкција се чита из меморије почев од адресе на коју указују тренутка вредност регистра програмског бројача $PC_{15...0}$. То се реализује тако што се чита бајт по бајт и после сваког прочитаног бајта садржај регистра $PC_{15...0}$ се инкрементира. Прочитани бајтови инструкције се смештају у прихватни регистар инструкције $IR_{23...0}$. У зависности од типа инструкције читају се 1, 2 или 3 бајта и смештају у разреде $IR_{23...16}$, $IR_{15...8}$ и $IR_{7...0}$. Обавезно се чита први бајт инструкције. У оквиру тога се, најпре, садржај регистра $PC_{15...0}$ пребацује у регистар $MAR_{15...0}$ и инкрементира садржај регистра $PC_{15...0}$. Затим се из меморије M са адресе која се налази у регистру $MAR_{15...0}$ чита бајт и уписује у регистар $MBR_{7...0}$. Садржај регистра $MBR_{7...0}$ се, на крају, пребацује у разреде $IR_{23...16}$.



Слика 2.а Дијаграм тока – фаза читање инструкције

Сада се врши провера сигнала логичког услова дужине инструкције **bezadr1**. Уколико је његова вредност 1, ради се о безадресној инструкцији чија је дужина 1 бајт. Тиме је завршена фаза читање инструкције и прелази се на корак 7 и фазу извршавање операција (слика 2.в). Уколико је његова вредност 0, врши се провера сигнала логичког услова дужине инструкције **adr1**. Уколико је његова вредност 1, ради се о адресној инструкцији чија је дужина 1 бајт. Тиме је завршена фаза читање инструкције и прелази се на корак 2 и фазу формирање адресе и читање операнда (слика 2.б). Уколико је његова вредност 0, ради се о некој од инструкција чија дужина може да буде 2 или 3 бајта. Зато се у овом случају, најпре, чита 2. бајт инструкције и уписује у разреде $IR_{15...8}$. Читање 2. бајта инструкције се реализује на сличан начин као и читање 1. бајта инструкције.

Затим се врши провера сигнала логичког услова дужине инструкције **branch2**. Уколико је његова вредност 1, ради се о инструкцији условног скока чија је дужина 2 бајта. Тиме је завршена фаза читање инструкције и прелази се на корак 7 и фазу извршавање операција (слика 2.в). Уколико је његова вредност 0, врши се провера сигнала логичког услова дужине инструкције **adr2**. Уколико је његова вредност 1, ради се о адресној инструкцији чија је дужина 2 бајта. Тиме је завршена фаза читање инструкције и прелази се на корак 2 и фазу формирање адресе и читање операнда (слика 2.б). Уколико је његова вредност 0, ради се о некој од инструкција чија је дужина може да буде 3 бајта. Зато се у овом случају, најпре, чита 3. бајт инструкције и уписује у

разреде $IR_{7...0}$. Читање 3. бајта инструкције се реализује на сличан начин као и читање 1. бајта инструкције.

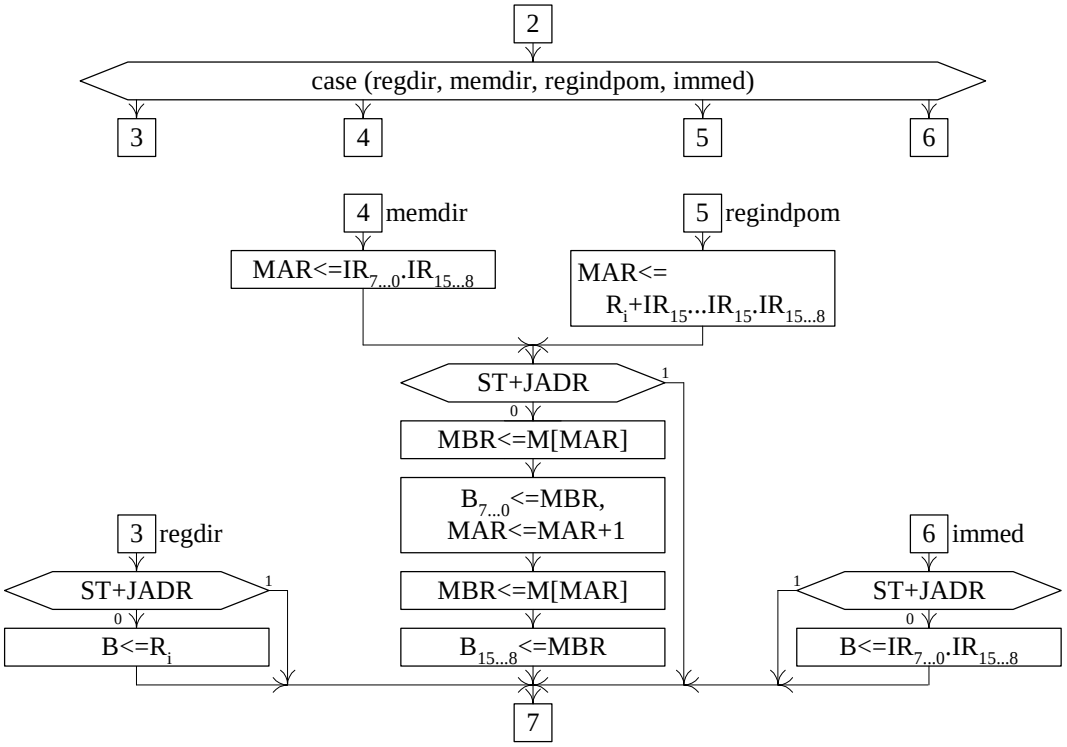
Сада се врши провера сигнала логичког услова дужине инструкције **jump3**. Уколико је његова вредност 1, ради се о инструкцији безусловног скока чија је дужина 3 бајта. Тиме је завршена фаза читање инструкције и прелази се на корак 7 и фазу извршавање операција (слика 2.в). Уколико је његова вредност 0, ради се о адресној инструкцији чија је дужина 3 бајта. Тиме је завршена фаза читање инструкције и прелази се на корак 2 и фазу формирање адресе и читање операнда (слика 2.б).

Формирање адресе и читање операнда (слика 2.б)

Формирање адресе и читање операнда се реализује само за адресне инструкције и то од корака 2. У овом кораку се реализује вишеструки условни скок на један од корака 3 до 6 у зависности од тога који од сигнала логичких услова начина адресирања **regdir** до **immed** има вредност 1. Сигнали логичких услова начина адресирања **regdir** до **immed** (табела 2.в) су бинарно кодирани разредима $IR_{19.18}$ прихватног регистра инструкције и само један од њих може да има вредност 1.

Уколико вредност 1 има сигнал логичког услова **regdir**, ради се о регистарском директном адресирању и прелазу са корака 2 на корак 3. Операнд је тада у регистру опште намене R_i на адреси која се налази у разредима $IR_{17...16}$. Садржај регистра R_i пребацује у регистар $B_{15...0}$ за све операције сем операција ST и JADR. Због тога се овде врши провера да ли један од сигнала логичких услова операција **ST** или **JADR** има вредност 1. Уколико оба сигнала **ST** и **JADR** имају вредност 0, регистар опште намене R_i се пребацује у регистар $B_{15...0}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 7 и фазу извршавање операција (слика 2.в). Уколико један од сигнала **ST** и **JADR** има вредност 1, одмах се завршава фаза формирање адресе и читање операнда и прелази се на корак 7 и фазу извршавање операција (слика 2.в).

Уколико вредност 1 има сигнал логичког услова **memdir**, ради се о меморијском директном адресирању и прелазу са корака 2 на корак 4. Операнд је тада у меморији на адреси која се налази у разредима $IR_{7...0}.IR_{15...8}$. Садржај $IR_{7...0}.IR_{15...8}$ се пребацује у регистар $MAR_{15...0}$ и прелази на корак почев од кога се за све операције, сем операција ST и JADR, чита операнд и смешта у регистар $B_{15...0}$. Због тога се овде врши провера да ли један од сигнала логичких услова операција **ST** или **JADR** има вредност 1. Уколико оба сигнала **ST** и **JADR** имају вредност 0, из меморије M се са адресе која се налази у регистру $MAR_{15...0}$, најпре, чита бајт и уписује у регистар $MBR_{7...0}$, а затим се садржај регистра $MBR_{7...0}$ пребацује у разреде $B_{7...0}$ и врши инкрементирање садржаја регистра $MAR_{15...0}$. Како је операнд ширине два бајта а ширина меморијске речи један бајт, потребно је из меморије прочитати још један бајт. С тога се из меморије M са адресе која се налази у регистру $MAR_{15...0}$, најпре, чита бајт и уписује у регистар $MBR_{7...0}$, а затим се садржај регистра $MBR_{7...0}$ пребацује у разреде $B_{15...8}$. Треба уочити да је први бајт податка прочитан из ниже меморијске локације млађи бајт операнда и да је стога уписан у млађих осам разреда регистра $B_{7...0}$ и да је други бајт податка прочитан из више меморијске локације старији бајт операнда и да је стога уписан у старијих осам разреда регистра $B_{15...8}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 7 и фазу извршавање операција (слика 2.в). Уколико један од сигнала **ST** и **JADR** има вредност 1, одмах се завршава фаза формирање адресе и читање операнда и прелази се на корак 7 и фазу извршавање операција (слика 2.в).



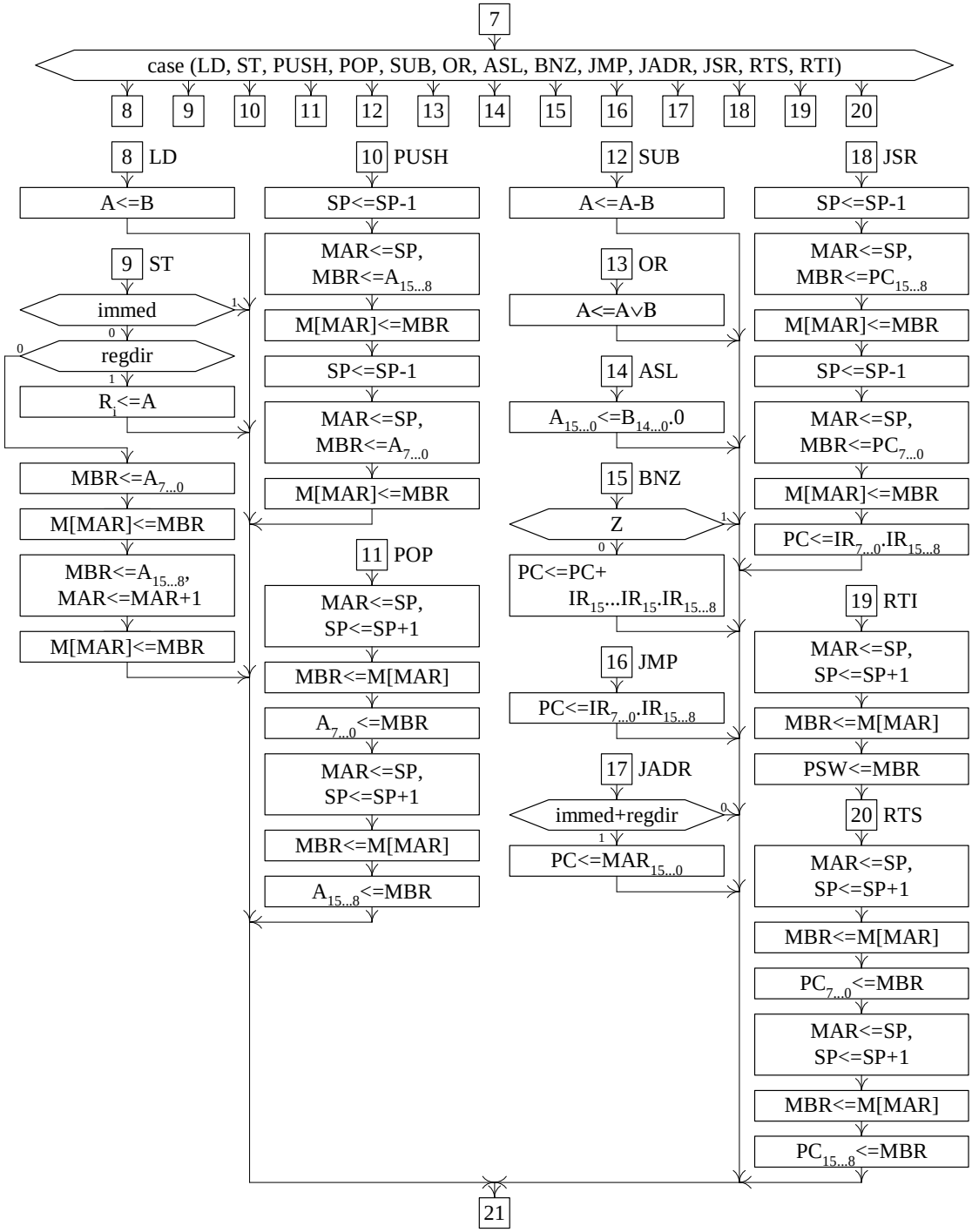
Слика 2.б Дијаграм тока – фаза формирање адресе и читање операнда

Уколико вредност 1 има сигнал логичког услова **regindpom**, ради се о регистарском индиректном адресирању са померајем и прелазу са корака 2 на корак 5. Операнд је тада у меморији на адреси која се добија сабирањем садржаја регистра опште намене R_i и помераја. Регистар опште намене R_i је на адреси која се налази у разредима $IR_{17.16}$, док се померај добија проширивањем знаком на 16 битну вредност разреда $IR_{15...8}$. Добијена адреса се пребацује у регистар $MAR_{15...0}$ и прелази на корак почев од кога се, на већ описани начин, за све операције, сем операција ST и JADR, чита операнд и смешта у регистар $B_{15...0}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 7 и фазу извршавање операција (слика 2.в).

Уколико вредност 1 има сигнал логичког услова **immed**, ради се о непосредном адресирању и прелазу са корака 2 на корак 6. Операнд се тада налази у разредима $IR_{7...0}.IR_{15...8}$ чији се садржај пребацује у регистар $B_{15...0}$ за све операције сем операција ST и JADR. Због тога се овде врши провера да ли један од сигнала логичких услова операција **ST** или **JADR** има вредност 1. Уколико оба сигнала **ST** и **JADR** имају вредност 0, разреди $IR_{7...0}.IR_{15...8}$ се пребацује у регистар $B_{15...0}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 7 и фазу извршавање операција (слика 2.в). Уколико један од сигнала **ST** и **JADR** има вредност 1, завршава се фаза формирање адресе и читање операнда и прелази се на корак 7 и фазу извршавање операција (слика 2.в).

Извршавање операција (слика 2.в)

Извршавање операција се реализује за све инструкције и то од корака 7. У овом кораку се реализује вишеструки условни скок на један од корака 8 до 20 у зависности од тога који од сигнала логичких услова операција **LD** до **RTI** има вредност 1. Сигнали логичких услова операција **LD** до **RTI** (табела 2.б) су бинарно кодирани разредима $IR_{23...16}$ прихватног регистра инструкције и само један од њих може да има вредност 1.



Слика 2.в Дијаграм тока – фаза извршавање операција

Уколико вредност 1 има сигнал логичког услова **LD**, садржај регистра $B_{15...0}$ се пребацује у регистар акумулатора $A_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 21 и фазу опслуживање прекида (слика 2.г).

Уколико вредност 1 има сигнал логичког услова **ST**, садржај регистра акумулатора $A_{15...0}$ се пребацује у регистар опште намене R_i уколико је специфицирано директно регистарско адресирање или у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$ уколико је специфицирано неко од меморијских адресирања. Непосредно адресирање није дозвољено за одредишни операнд и зато се узима да се у случају да се у инструкцији ST појави непосредно адресирање, фаза извршавања ове операције завршава, чиме се ова инструкција претвара у инструкцију без дејства. С тога се

проверава вредност сигнала **immed** 1. Уколико сигнал **immed** има вредност 1, специфицирано је непосредно адресирање, па се фаза извршавања операције завршава и прелази на корак 21 и фазу опслуживање прекида (слика 2.г). Уколико сигнал **immed** има вредност 0, проверава се вредност сигнала **regdir**. Уколико сигнал **regdir** има вредност 1, регистарско директно адресирање је специфицирано, па се садржај регистра $A_{15...0}$ уписује у регистар опште намене R_i на адреси специфицираној разредима $IR_{19...18}$ и прелази на корак 21 и фазу опслуживање прекида (слика 2.г). Уколико сигнал **regdir** има вредност 1, неко од меморијских адресирања је специфицирано, па се садржај регистра $A_{15...0}$ уписује у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$. У оквиру тога се, најпре, старији бајт регистра $A_{15...8}$ пребацује у регистар $MBR_{7...0}$ и уписује у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$. Затим се, пошто се млађи бајт регистра $A_{7...0}$ пребаци у регистар $MBR_{7...0}$ и садржај регистра $MAR_{15...0}$ инкрементира, садржај регистра $MBR_{7...0}$ уписује у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$. Треба уочити да се у меморију прво уписује млађи а потом старији бајт регистра $A_{15...0}$. То је последица начина смештања дво бајтовских величина у меморију, који предвиђа да старији бајт буде на вишој, а млађи бајт на нижој адреси. Тиме је завршена фаза извршавање операције и прелази се на корак 21 и фазу опслуживање прекида (слика 2.г).

Уколико вредност 1 има сигнал логичког услова **PUSH**, садржај регистра акумулатора $A_{15...0}$ се ставља на стек. То се реализује тако што се најпре декрементира садржај регистра $SP_{15...0}$. Потом се пребацују садржај регистар $SP_{15...0}$ у регистар $MAR_{15...0}$ и старији бајт регистра $A_{15...8}$ у регистар $MBR_{7...0}$. Затим се садржај регистра $MBR_{7...0}$ уписује у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$. После тога се поново најпре декрементира садржај регистра $SP_{15...0}$ и затим пребацују садржај регистра $SP_{15...0}$ у регистар $MAR_{15...0}$ и млађи бајт регистра $A_{7...0}$ у регистар $MBR_{7...0}$. На крају се садржај регистра $MBR_{7...0}$ уписује у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$. Треба уочити да стек расте према нижим локацијама и да стек поинтер указује на задњу заузету локацију. С тога се садржај регистра $SP_{15...0}$ прво декрементира и после тога пребацује у регистар $MAR_{15...0}$. Такође треба уочити да се на стек прво ставља старији а потом млађи бајт регистра $A_{15...0}$. То је последица начина смештања дво бајтовских величина у меморију, који предвиђа да старији бајт буде на вишој, а млађи бајт на нижој адреси. Тиме је завршена фаза извршавање операција и прелази се на корак 21 и фазу опслуживање прекида (слика 2.г).

Уколико вредност 1 има сигнал логичког услова **POP**, садржајем са стека се рестаурира садржај регистра акумулатора $A_{15...0}$. То се реализује тако што се најпре врши пребацивање садржаја регистра $SP_{15...0}$ у регистар $MAR_{15...0}$ и инкрементирање садржаја регистра $SP_{15...0}$. Затим се из меморијске локације са адресе која се налази у регистру $MAR_{15...0}$ чита садржај и уписује у регистар $MBR_{7...0}$. На крају се садржај регистра $MBR_{7...0}$ уписује у млађи бајт регистра $A_{7...0}$. После тога се поново врши пребацивање садржаја регистра $SP_{15...0}$ у регистар $MAR_{15...0}$ и инкрементирање садржаја регистра $SP_{15...0}$. Затим се из меморијске локације са адресе која се налази у регистру $MAR_{15...0}$ чита садржај и уписује у регистар $MBR_{7...0}$. На крају се садржај регистра $MBR_{7...0}$ уписује у старији бајт регистра $A_{15...8}$. Треба уочити да стек расте према нижим локацијама и да стек поинтер указује на задњу заузету локацију. С тога се приликом читања садржаја са стека, садржај регистра $SP_{15...0}$ прво пребацује у регистар $MAR_{15...0}$ и после тога инкрементира. Такође треба уочити да се са стека прво скида млађи а потом старији бајт регистра $A_{15...0}$. То је последица начина смештања дво бајтовских величина у меморију, који предвиђа да старији бајт буде на вишој, а млађи бајт на нижој адреси.

Тиме је завршена фаза извршавање операција и прелази се на корак 21 и фазу опслуживање прекида (слика 2.г).

Уколико вредност 1 има сигнал логичког услова **SUB**, садржај регистра $A_{15...0}$ се умањује за садржај регистра $B_{15...0}$ и резултат уписује у регистар $A_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 21 и фазу опслуживање прекида (слика 2.г).

Уколико вредност 1 има сигнал логичког услова **OR**, логичко сабирање се реализује над садржајима регистара $A_{15...0}$ и $B_{15...0}$ и резултат уписује у регистар $A_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 21 и фазу опслуживање прекида (слика 2.г).

Уколико вредност 1 има сигнал логичког услова **ASL**, садржај регистра $B_{15...0}$ се логички помера улево за једно место и уписује у регистар $A_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 21 и фазу опслуживање прекида (слика 2.г).

Уколико вредност 1 има сигнал логичког услова **BNZ**, условни скок на основу вредности сигнала логичког услова **Z** се реализује. Сигнал **Z** има вредност 1 уколико је резултат задње извршене инструкције 0 и вредност 0 уколико резултат задње извршене инструкције није 0. Уколико сигнал **Z** има вредност 1, услов за скок није испуњен. Тиме је завршена фаза извршавање операција и прелази се на корак 21 и фазу опслуживање прекида (слика 2.г). Уколико сигнал **Z** има вредност 0, услов за скок је испуњен, па се сабирају садржај регистра $PC_{15...0}$, који представља текућу вредност регистра програмског бројача $PC_{15...0}$, и садржај разреда $IR_{15...8}$ проширен знаком на 16 бита, који представља померај, и добијена вредност, која представља адресу скока, уписује у регистар $PC_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 21 и фазу опслуживање прекида (слика 2.г).

Уколико вредност 1 има сигнал логичког услова **JMP**, безусловни скок се реализује. Садржај разреда $IR_{7...0}.IR_{15...8}$, који представља адресу скока, уписује се у регистар $PC_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 21 и фазу опслуживање прекида (слика 2.г).

Уколико вредност 1 има сигнал логичког услова **JADR** активан, безусловни скок на срачунату адресу се реализује, под условом да је задато неко од меморијских адресирања. У оквиру тога се садржај разреда $MAR_{15...0}$, који представља срачунату адресу скока задату неким од меморијских адресирања, уписује у регистар $PC_{15...0}$. Непосредно адресирање и регистарско директно адресирање нису дозвољени јер не дају адресу меморијске локације и зато се узима да се у случају да се у инструкцији JADR појави непосредно адресирање или регистарско директно адресирање, фаза извршавања ове операције завршава, чиме се ова инструкција претвара у инструкцију без дејства. С тога се проверава да ли вредност 1 имају сигнал **immed** или сигнал **regdir**. Уколико да, специфицирано је непосредно адресирање или регистарско директно адресирање, па се фаза извршавања операције завршава и прелази на корак 21 и фазу опслуживање прекида (слика 2.г). Уколико не, специфицирано је неко од меморијских адресирања, па се садржај разреда $MAR_{15...0}$ уписује у регистар $PC_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 21 и фазу опслуживање прекида (слика 1.г).

Уколико вредност 1 има сигнал логичког услова **JSR**, скок на потпрограм се реализује. У оквиру тога се садржај регистра $PC_{15...0}$ ставља на стек. Најпре се декрементира садржај регистра $SP_{15...0}$. Затим се пребацују садржај регистра $SP_{15...0}$ у регистар $MAR_{15...0}$ и старији бајт регистра $PC_{15...8}$ у регистар $MBR_{7...0}$. Затим се садржај регистра $MBR_{7...0}$ уписује у меморијску локацију на адреси која се налази у регистру

$MAR_{15...0}$. После тога се поново декрементира садржај регистра $SP_{15...0}$ и пребацују садржај регистар $SP_{15...0}$ у регистар $MAR_{15...0}$ и млађи бајт регистра $PC_{7...0}$ у регистар $MBR_{7...0}$. На крају се садржај регистра $MBR_{7...0}$ уписује у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$. Треба уочити да стек расте према нижим локацијама и да стек поинтер указује на задњу заузету локацију. С тога се садржај регистра $SP_{15...0}$ прво декрементира и после тога пребацује у регистар $MAR_{15...0}$. Такође треба уочити да се на стек прво ставља старији а потом млађи бајт регистра $PC_{15...0}$. То је последица начина смештања дво бајтовских величина у меморију, који предвиђа да старији бајт буде на вишој, а млађи бајт на нижој адреси. Затим се садржај разреда $IR_{7...0}.IR_{15...8}$, који представља адресу потпрограма, уписује у регистар $PC_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 21 и фазу опслуживање прекида (слика 2.г).

Уколико вредност 1 има сигнал логичког услова **RTI**, повратак из прекидне рутине се реализује. У оквиру тога се садржајима са стека рестаурирају садржаји регистара $PSW_{7...0}$ и $PC_{15...0}$. Најпре се врши пребацивање садржаја регистра $SP_{15...0}$ у регистар $MAR_{15...0}$ и инкрементирање садржаја регистра $SP_{15...0}$. Затим се из меморијске локације са адресе која се налази у регистру $MAR_{15...0}$ чита садржај и уписује у регистар $MBR_{7...0}$. На крају се садржај регистра $MBR_{7...0}$ уписује у регистар $PSW_{7...0}$. На исти начин се са стека читају још два бајта и уписују најпре у млађи а потом у старији бајт регистра $PC_{15...0}$. Треба уочити да стек расте према нижим локацијама и да стек поинтер указује на задњу заузету локацију. С тога се приликом читања садржаја са стека, садржај регистра $SP_{15...0}$ прво пребацује у регистар $MAR_{15...0}$ и после тога инкрементира. Такође треба уочити да се са стека прво скида млађи а потом старији бајт регистра $PC_{15...0}$. То је последица начина смештања дво бајтовских величина у меморију, који предвиђа да старији бајт буде на вишој, а млађи бајт на нижој адреси. Тиме је завршена фаза извршавање операција и прелази се на корак 21 и фазу опслуживање прекида (слика 2.г).

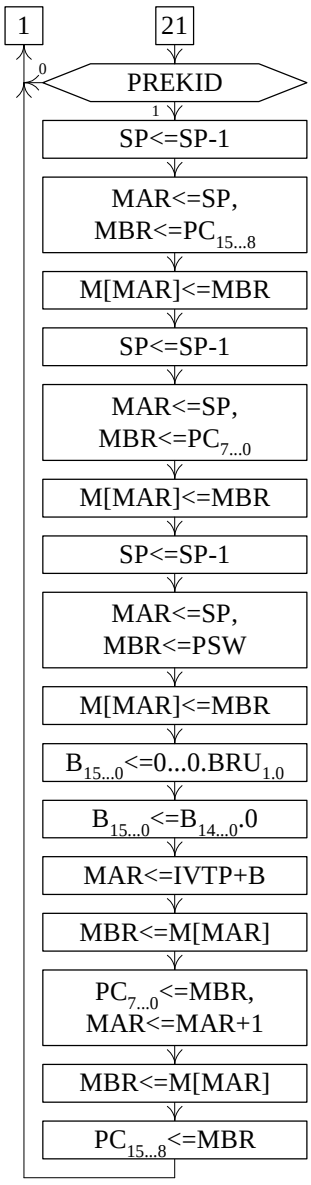
Уколико вредност 1 има сигнал логичког услова **RTS**, повратак из потпрограма се реализује. У оквиру тога се садржајем са стека рестаурира садржај регистра $PC_{15...0}$. Ово се реализује на идентичан као и у случају инструкције RTI. Тиме је завршена фаза извршавање операција и прелази се на корак 21 и фазу опслуживање прекида (слика 2.г).

Опслуживање прекида (слика 2.г)

Опслуживање прекида се реализује за све инструкције и то почев од корака 21. Уколико је сигнал логичког услова **PREKID** има вредност 0, у току извршавања претходних фаза није дошло до генерисања сигнала прекида, па се фаза опслуживање прекида завршава и прелази се на корак 1 и фазу читање инструкције (слика 2.а). Уколико сигнал **PREKID** има вредност 1, у току извршавања претходних фаза дошло је до генерисања сигнала прекида, па се прелази на кораке у оквиру којих се на стеку стављају садржаји регистара $PC_{15...0}$ и $PSW_{7...0}$ и потом утврђује адреса прекидне рутине и уписује у регистар $PC_{15...0}$. Стављање садржаја регистра $PC_{15...0}$ на стек се реализује на идентичан начин као и у случају инструкције JSR. На исти начин се на стек ставља и садржај регистра $PSW_{7...0}$. Адресе прекидних рутина се налазе у улазима табеле са адресама прекидних рутина. Број улаза у табелу је дат садржајем регистра $BRU_{1.0}$, а почетна адреса табеле садржајем регистра $IVTP_{15...0}$. Најпре се садржај регистра $BRU_{1.0}$ проширен нулама до дужине 16 бита пребацује у регистар $B_{15...0}$, па се садржај регистра $B_{15...0}$ померањем улево за једно место множи са два. Тиме се број улаза претвара у померај. Потом се сабирањем садржаја регистара $IVTP_{15...0}$ и $B_{15...0}$ и смештањем њихове

суме у регистар $MAR_{15...0}$ добија адреса на којој се налази адреса прекидне рутине. Са те и следеће адресе из меморије се читају млађи и старији бајт адресе прекидне рутине и уписују у млађих и старијих осам разреда регистра $PC_{15...0}$. Фаза опслуживање прекида је тиме завршена и прелази се на корак 1 и фазу читање инструкције (слика 2.а).

Треба уочити да се јављају две ситуације везане за вредност регистра $PC_{15...0}$ по завршетку фазе опслуживање прекида и преласка на корак 1 и фазу читање инструкције (слика 2.а). Уколико је сигнал **PREKID** имао вредност 0, у регистру $PC_{15...0}$ је адреса прве следеће инструкције после инструкције која је извршена. Уколико је сигнал **PREKID** имао вредност 1, у регистру $PC_{15...0}$ је адреса прве инструкције прекидне рутине.



Слика 2.г Дијаграм тока – фаза опслуживање прекида

3 ЗАДАТАК 3

Посматра се део рачунара који чине меморија и процесор.

Меморија је капацитета 2^{16} бајтова. Ширина меморијске речи је 1 бајт.

Процесор је са једноадресним форматом инструкција. Подаци су целобројне величине без знака дужине 1 бајт.

У процесору постоји програмски бројач PC дужине 2 бајта, адресни регистар меморије MAR дужине 2 бајта, прихватни регистар податка меморије MBR дужине 1 бајт, прихватни регистар инструкције IR дужине 3 бајта, акумулатор A дужине 1 бајт, прихватни регистар податка B дужине 2 бајта, регистри опште намене R[0] и R[1] дужине 2 бајта, програмска статусна реч PSW дужине 1 бајт, указивач на врх стека SP дужине 2 бајта, регистар броја улаза у табелу са адресама прекидних рутина BRU дужине 2 бита и указивач на табелу са адресама прекидних рутина IVTP дужине 2 бајта. Инструкције су дужине један, два или три бајта.

Бит 7 првог бајта инструкције има вредност 0 за безадресне инструкције и инструкције скока, док бит 6 првог бајта инструкције има вредност 0 за безадресне инструкције и вредност 1 за инструкције скока. Безадресне инструкције су инструкције повратка из потпрограма (RTS), повратка из прекидне рутине (RTI), стављања садржаја акумулатора на стек (PUSH) и скидања садржаја са стека и пуњење акумулатора (POP). Битовима 5 до 0 првог бајта инструкција специфицира се код операције за безадресне инструкције. На основу тога су за инструкције RTS, RTI, PUSH и POP усвојени кодови операција 000000, 000001, 000010 и 000011, респективно. Дужина инструкција је 1 бајт. Бит 5 првог бајта инструкције има вредност 0 за инструкције условног скока и 1 за инструкције безусловног скока. Инструкције скока је инструкција условног скока уколико је резултат нула (BZ). Битовима 4 до 0 првог бајта инструкција специфицира се код операције за инструкција условног скока. На основу тога је за инструкцију BZ усвојен код операције 00000. Инструкција BZ се реализује као релативни скок у односу на текућу вредност програмског бројача PC, а померај је 8 битна целобројна величина са знаком дата 2. бајтом инструкције. Дужина инструкција је 2 бајта. Инструкције безусловног скока су инструкција безусловног скока (JMP) и инструкција скока на потпрограм (JSR). Битовима 4 до 0 првог бајта инструкција специфицира се код операције за инструкција безусловног скока. На основу тога је за инструкције JMP и JSR усвојени кодови операција 00000 и 00001, респективно. Инструкције JMP и JSR се реализују као апсолутни скокови, а адреса скока је дата 2. и 3. бајтом инструкције, при чему је старији бајт адресе скока дат другим бајтом инструкције а млађи бајт адресе скока трећим бајтом инструкције. Дужина инструкција је 3 бајта.

Бит 7 првог бајта инструкције има вредност 1 за адресне инструкције. Адресне инструкције су инструкције преноса у акумулатор (LD), инструкција преноса из акумулатора (ST), аритметичка инструкција сабирања (ADD), логичка инструкција ексклузивно ИЛИ (XOR), инструкција логичког померања удесно за једно место (LSR) и инструкција безусловног скока на срачунату адресу (JADR). У инструкцији ST није дозвољено непосредно адресирање, у инструкцији JADR није дозвољено регистарско директно и непосредно адресирање, па уколико се јаве ова адресирања у овим инструкцијама, инструкције треба да буду без дејства, а инструкција LSR резултат померања смешта у регистар A. Битовима 6 до 3 првог бајта инструкција специфицира се код операције. На основу тога су за инструкције LD, ST, ADD, XOR, LSR и JADR

усвојени кодови операција 0000, 0001, 0010, 0011, 0100 и 0101, респективно. Дужина инструкција је 1, 2 или 3 бајта и зависи од специфицираног начина адресирања.

За адресне инструкције се битовима 2, 1 и 0 првог бајта инструкције специфицира начин адресирања и регистар опште намене уколико се користи у задатом начину адресирања. Бит 2 има вредност 0 за регистарско директно и регистарско индиректно адресирање, а вредност 1 за непосредно, PC релативно са померајем, меморијско директно и меморијско индиректно адресирање. Уколико бит 2 има вредност 0, тада се вредностима 0 и 1 бита 1 одређује регистарско директно и регистарско индиректно адресирање, респективно, а битом 0 специфицира регистар опште намене R0 или R1 који се користе за ова два адресирања. Дужина инструкције је 1 бајт. Уколико бит 2 има вредност 1, тада са вредностима 00, 01, 10 и 11 битова 1 и 0 одређује непосредно, PC релативно са померајем, меморијско директно и меморијско индиректно адресирање, респективно. Код непосредног адресирања други бајт инструкције садржи 8 битни податак. Дужина инструкције је 2 бајта. Код PC релативног са померајем адресирања други бајт инструкције садржи 8 битни померај који је дат као целобројна величина са знаком. Дужина инструкције је 2 бајта. Код меморијског директног и меморијског индиректног адресирања други и трећи бајт инструкције садрже адресу меморијске локације, при чему је старији бајт адресе меморијске локације дат другим а млађи бајт трећим бајтом. Дужина инструкције је 3 бајта.

Стек расте према вишим меморијским локацијама, а регистар SP указује на задњу заузету меморијску локацију.

Захтеви за прекид долазе од 4 улазно/излазна уређаја по линијама означеним од 0 до 3. По линији 0 стиже захтев за прекид најнижег, а по линији 3 највишег приоритета. Број линије највишег приоритета по којој је стигао захтев за прекид налази се у бинарном облику у регистру BRU дужине 2 разреда. Адресе прекидних рутина 4 улазно/излазна уређаја који по линијама означеним од 0 до 3 шаљу захтеве за прекид налазе се у улазима 0 до 3 табеле са адресама прекидних рутина. Адресе дужине 16 бита заузимају по две суседне меморијске локације, при чему се старији бајт налази на нижој а млађи бајт на вишој адреси Садржај регистра BRU представља број улаза у табелу са адресама прекидних рутина. Почетна адреса табеле са адресама прекидних рутина се налази у регистру IVTP дужине 2 бајта. У оквиру хардверског дела опслуживања захтева за прекид на стек са стављају само регистри PC и PSW.

Нацртати и објаснити дијаграм тока фаза извршавања инструкције и то: фазе читања инструкције, фазе формирања адресе и читања операнда, фаза извршавања операција LD, ST, PUSH, POP, ADD, XOR, LSR, BZ, JMP, JADR, JSR, RTS и RTI и фазе опслуживања захтева за прекид.

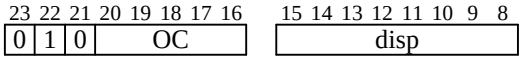
РЕШЕЊЕ

Дијаграм тока извршавања инструкције је дат на сликама 3.а, 3.б, 3.в и 3.г. Извршавање инструкције се састоји из четири фазе: читање инструкције (слика 3.а), формирање адресе и читање операнда (слика 3.б), извршавање операција (слика 3.в) и опслуживање прекида (слика 3.г). Дијаграм тока је дат у сагласности са форматима инструкција (табела 3.а), при чему први, други и трећи бајт инструкције се смештају у разреде 23 до 16, 15 до 8 и 7 до 0 прихватног регистра инструкције IR. У њему се користе сигнали логичких услова операција (табела 3.б), начина адресирања (табела 3.в) и дужина инструкција (табела 3.г) који се формирају на основу разреда 23 до 16 прихватног регистра инструкције. У дијаграму тока се користе и сигнали логичких услова **Z** и **PREKID**. Сигнал **Z** представља један од индикатора програмске статусне речи PSW који вредностима 1 и 0 указује да ли је резултат задње извршене инструкције

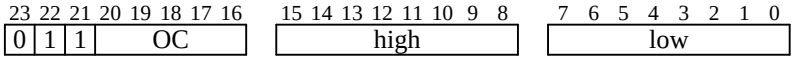
нула или различит од нуле, респективно, док сигнал **PREKID** вредностима 1 и 0 указује да ли је током извршавања инструкције дошло или није дошло до генерисања прекида, респективно. Генерисање сигнала **Z** и **PREKID** је ван оквира овог задатка и није приказано.

Табела 3.а Формати инструкција

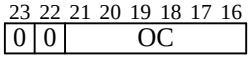
Instrukcija uslovnog skoka BZ



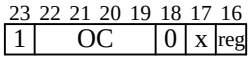
Instrukcije bezuslovnog skoka JMP i JSR



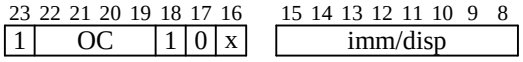
Bezadresne instrukcije PUSH, POP, RTS i RTI



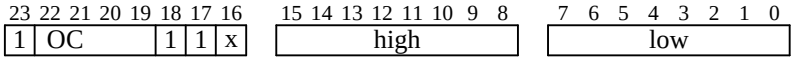
Adresne instrukcije LD, ST, ADD, XOR, LSL i JADR sa registarskim direktnim i registarskim indirektnim adresiranjima



Adresne instrukcije LD, ST, ADD, XOR, LSL i JADR sa neposrednim i PC relativnim sa pomerajem adresiranjima



Adresne instrukcije LD, ST, ADD, XOR, LSL i JADR sa memorijskim direktnim i memorijskim indirektnim adresiranjima



Табела 3.б Сигнали операција

$BZ=\overline{IR_{23}}\cdot IR_{22}\cdot\overline{IR_{21}}\cdot\overline{IR_{20}}\cdot\overline{IR_{19}}\cdot\overline{IR_{18}}\cdot\overline{IR_{17}}\cdot\overline{IR_{16}}$

$JMP=\overline{IR_{23}}\cdot IR_{22}\cdot IR_{21}\cdot\overline{IR_{20}}\cdot\overline{IR_{19}}\cdot\overline{IR_{18}}\cdot\overline{IR_{17}}\cdot\overline{IR_{16}}$

$JSR=\overline{IR_{23}}\cdot IR_{22}\cdot IR_{21}\cdot\overline{IR_{20}}\cdot\overline{IR_{19}}\cdot\overline{IR_{18}}\cdot\overline{IR_{17}}\cdot IR_{16}$

$PUSH=\overline{IR_{23}}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot\overline{IR_{20}}\cdot\overline{IR_{19}}\cdot\overline{IR_{18}}\cdot\overline{IR_{17}}\cdot\overline{IR_{16}}$

$POP=\overline{IR_{23}}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot\overline{IR_{20}}\cdot\overline{IR_{19}}\cdot\overline{IR_{18}}\cdot\overline{IR_{17}}\cdot IR_{16}$

$RTS=\overline{IR_{23}}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot\overline{IR_{20}}\cdot\overline{IR_{19}}\cdot\overline{IR_{18}}\cdot IR_{17}\cdot\overline{IR_{16}}$

$RTI=\overline{IR_{23}}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot\overline{IR_{20}}\cdot\overline{IR_{19}}\cdot\overline{IR_{18}}\cdot IR_{17}\cdot IR_{16}$

$LD=IR_{23}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot\overline{IR_{20}}\cdot\overline{IR_{19}}$

$ST=IR_{23}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot\overline{IR_{20}}\cdot IR_{19}$

$ADD=IR_{23}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot IR_{20}\cdot\overline{IR_{19}}$

$XOR=IR_{23}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot IR_{20}\cdot IR_{19}$

$JADR=IR_{23}\cdot\overline{IR_{22}}\cdot IR_{21}\cdot\overline{IR_{20}}\cdot\overline{IR_{19}}$

Табела 3.в Сигнали начина адресирања

$$regdir=\overline{IR_{18}}\cdot\overline{IR_{17}}$$
$$regind=\overline{IR_{18}}\cdot IR_{17}$$
$$immed=IR_{18}\cdot\overline{IR_{17}}\cdot\overline{IR_{16}}$$
$$pcrelpom=IR_{18}\cdot\overline{IR_{17}}\cdot IR_{16}$$
$$memdir=IR_{18}\cdot IR_{17}\cdot\overline{IR_{16}}$$
$$memind=IR_{18}\cdot IR_{17}\cdot IR_{16}$$

Табела 3.г Сигнали дужина инструкција

bezadr1=PUSH+POP+RTS+RTI

adr1=(LD+ST+ADD+XOR+LSR)·(regdir+regind)+JADR·regind

branch2=BNZ

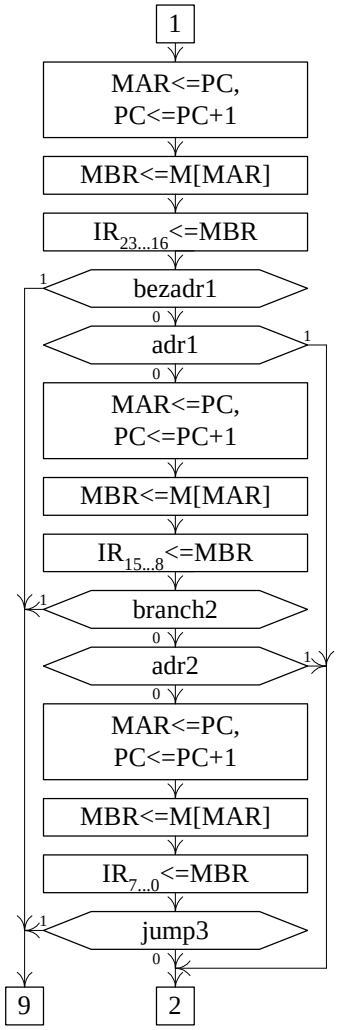
adr2=(LD+ST+ADD+XOR+LSR)·(immed+pcrelpom)+JADR·pcrelpom

jmp3=JMP+JSR

adr3=(LD+ST+ADD+XOR+LSR+JADR)·(memdir+memind)

Читање инструкције (слика 3.а)

Инструкција се чита из меморије почев од адресе на коју указују тренутка вредност регистра програмског бројача $PC_{15...0}$. То се реализује тако што се чита бајт по бајт и после сваког прочитаног бајта садржај регистра $PC_{15...0}$ се инкрементира. Прочитани бајтови инструкције се смештају у прихватни регистар инструкције $IR_{23...0}$. У зависности од типа инструкције читају се 1, 2 или 3 бајта и смештају у разреде $IR_{23...16}$, $IR_{15...8}$ и $IR_{7...0}$. Обавезно се чита први бајт инструкције. У оквиру тога се, најпре, садржај регистра $PC_{15...0}$ пребацује у регистар $MAR_{15...0}$ и инкрементира садржај регистра $PC_{15...0}$. Затим се из меморије M са адресе која се налази у регистру $MAR_{15...0}$ чита бајт и уписује у регистар $MBR_{7...0}$. Садржај регистра $MBR_{7...0}$ се, на крају, пребацује у разреде $IR_{23...16}$.



Слика 3.а Дијаграм тока – фаза читање инструкције

Сада се врши провера сигнала логичког услова дужине инструкције **bezadr1**. Уколико је његова вредност 1, ради се о безадресној инструкцији чија је дужина 1 бајт. Тиме је завршена фаза читање инструкције и прелази се на корак 9 и фазу извршавање операција (слика 3.в). Уколико је његова вредност 0, врши се провера сигнала логичког услова дужине инструкције **adr1**. Уколико је његова вредност 1, ради се о адресној инструкцији чија је дужина 1 бајт. Тиме је завршена фаза читање инструкције и прелази се на корак 2 и фазу формирање адресе и читање операнда (слика 3.б). Уколико је његова вредност 0, ради се о некој од инструкција чија дужина може да буде 2 или 3 бајта. Зато се у овом случају, најпре, чита 2. бајт инструкције и уписује у разреде $IR_{15...8}$. Читање 2. бајта инструкције се реализује на сличан начин као и читање 1. бајта инструкције.

Затим се врши провера сигнала логичког услова дужине инструкције **branch2**. Уколико је његова вредност 1, ради се о инструкцији условног скока чија је дужина 2 бајта. Тиме је завршена фаза читање инструкције и прелази се на корак 9 и фазу извршавање операција (слика 3.в). Уколико је његова вредност 0, врши се провера сигнала логичког услова дужине инструкције **adr2**. Уколико је његова вредност 1, ради се о адресној инструкцији чија је дужина 2 бајта. Тиме је завршена фаза читање инструкције и прелази се на корак 2 и фазу формирање адресе и читање операнда (слика 3.б). Уколико је његова вредност 0, ради се о некој од инструкција чија је дужина може

да буде 3 бајта. Зато се у овом случају, најпре, чита 3. бајт инструкције и уписује у разреде $IR_{7...0}$. Читање 3. бајта инструкције се реализује на сличан начин као и читање 1. бајта инструкције.

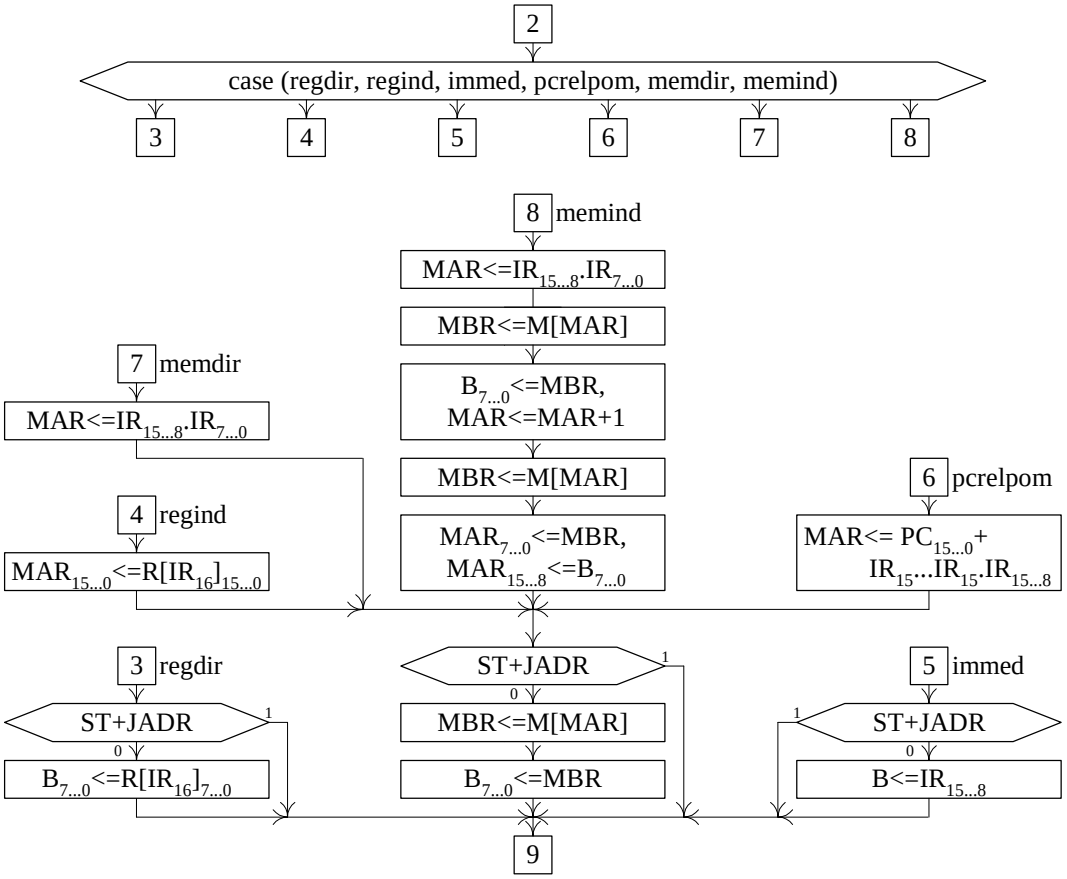
Сада се врши провера сигнала логичког услова дужине инструкције **jump3**. Уколико је његова вредност 1, ради се о инструкцији безусловног скока чија је дужина 3 бајта. Тиме је завршена фаза читање инструкције и прелази се на корак 9 и фазу извршавање операција (слика 3.в). Уколико је његова вредност 0, ради се о адресној инструкцији чија је дужина 3 бајта. Тиме је завршена фаза читање инструкције и прелази се на корак 2 и фазу формирање адресе и читање операнда (слика 3.б).

Формирање адресе и читање операнда (слика 3.б)

Формирање адресе и читање операнда се реализује само за адресне инструкције и то од корака 2. У овом кораку се реализује вишеструки условни скок на један од корака 3 до 8 у зависности од тога који од сигнала логичких услова начина адресирања **regdir** до **memind** има вредност 1. Сигнали логичких услова начина адресирања **regdir** до **memind** (табела 3.в) су бинарно кодирани разредима $IR_{18.17}$ или $IR_{18...16}$ прихватног регистра инструкције и само један од њих може да има вредност 1.

Уколико вредност 1 има сигнал логичког услова **regdir**, ради се о регистарском директном адресирању и прелазу са корака 2 на корак 3. Операнд је тада у доњих 8 разреда регистра опште намене R[0] или R[1] адресираног садржајем разреда IR_{16}. Адресирани регистар R[0] или R[1] се пребацује у регистар $B_{7...0}$ за све операције сем операција ST и JADR. Због тога се овде врши провера да ли један од сигнала логичких услова операција **ST** или **JADR** има вредност 1. Уколико оба сигнала **ST** и **JADR** имају вредност 0, адресирани регистар R[0] или R[1] се пребацује у регистар $B_{7...0}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 9 и фазу извршавање операција (слика 3.в). Уколико један од сигнала **ST** и **JADR** има вредност 1, одмах се завршава фаза формирање адресе и читање операнда и прелази се на корак 9 и фазу извршавање операција (слика 3.в).

Уколико вредност 1 има сигнал логичког услова **regind**, ради се о регистарском индиректном адресирању и прелазу са корака 2 на корак 4. Операнд је тада у меморији на адреси која се налази у регистру опште намене R[0] или R[1] адресираном садржајем разреда IR_{16}. Адресирани регистар R[0] или R[1] се пребацује у регистар $MAR_{15...0}$ и прелази на корак почев од кога се за све операције, сем операција ST и JADR, чита операнд и смешта у регистар $B_{7...0}$. Због тога се овде врши провера да ли један од сигнала логичких услова операција **ST** или **JADR** има вредност 1. Уколико оба сигнала **ST** и **JADR** имају вредност 0, из меморије M се са адресе која се налази у регистру $MAR_{15...0}$, најпре, чита бајт и уписује у регистар $MBR_{7...0}$, а затим се садржај регистра $MBR_{7...0}$ пребацује у разреде $B_{7...0}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 9 и фазу извршавање операција (слика 3.в). Уколико један од сигнала **ST** и **JADR** има вредност 1, одмах се завршава фаза формирање адресе и читање операнда и прелази се на корак 9 и фазу извршавање операција (слика 3.в).



Слика 3.б Дијаграм тока – фаза формирање адресе и читање операнда

Уколико вредност 1 има сигнал логичког услова **immed**, ради се о непосредном адресирању и прелазу са корака 2 на корак 5. Операнд се тада налази у разредима $IR_{15...8}$ чији се садржај пребацује у регистар $B_{7...0}$ за све операције сем операција ST и JADR. Због тога се овде врши провера да ли један од сигнала логичких услова операција **ST** или **JADR** има вредност 1. Уколико оба сигнала **ST** и **JADR** имају вредност 0, разреди $IR_{15...8}$ се пребацује у регистар $B_{7...0}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 9 и фазу извршавање операција (слика 3.в). Уколико један од сигнала **ST** и **JADR** има вредност 1, завршава се фаза формирање адресе и читање операнда и прелази се на корак 9 и фазу извршавање операција (слика 3.в).

Уколико вредност 1 има сигнал логичког услова **pcrelpom**, ради се о PC релативном са померајем адресирању и прелазу са корака 2 на корак 6. Операнд је тада у меморији на адреси која се добија сабирањем садржаја програмског бројача $PC_{15...0}$ и помераја који се добија проширивањем разреда $IR_{15...8}$ знаком на 16 битну вредност. Добијена адреса се пребацује у регистар $MAR_{15...0}$ и прелази на корак почев од кога се, на већ описани начин, за све операције, сем операција ST и JADR, чита операнд и смешта у регистар $B_{7...0}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 9 и фазу извршавање операција (слика 3.в).

Уколико вредност 1 има сигнал логичког услова **memdir**, ради се о меморијском директном адресирању и прелазу са корака 2 на корак 7. Операнд је тада у меморији на адреси која се налази у разредима $IR_{15...8}.IR_{7...0}$. Садржај $IR_{15...8}.IR_{7...0}$ се пребацује у регистар $MAR_{15...0}$ и прелази на корак почев од кога се, на већ описани начин, за све операције, сем операција ST и JADR, чита операнд и смешта у регистар $B_{7...0}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 9 и фазу извршавање операција (слика 3.в).

Уколико вредност 1 има сигнал логичког услова **memind**, ради се о меморијском индиректном адресирању и прелазу са корака 2 на корак 8. Операнд је тада у меморији, а адреса меморијске локације је у другој меморијској локацији чија се адреса налази у разредима $IR_{15...0}$. Стога се садржај разреда $IR_{15...0}$ пребацује у регистар $MAR_{15...0}$. Из меморије M се са адресе која се налази у регистру $MAR_{15...0}$, најпре, чита старији бајт адресе операнда и уписује у регистар $MBR_{7...0}$, а затим се садржај регистра $MBR_{7...0}$ привремено пребацује у разреде $B_{15...8}$ и врши инкрементирање садржаја регистра $MAR_{15...0}$. Како је адреса ширине два бајта а ширина меморијске речи један бајт, потребно је из меморије прочитати још један бајт. С тога се из меморије M са адресе која се налази у регистру $MAR_{15...0}$, затим, чита бајт млађи бајт адресе и уписује у регистар $MBR_{7...0}$. Треба уочити да је први бајт прочитан из ниже меморијске локације старији бајт адресе операнда и да је привремено уписан у осам виших разреда регистра $B_{15...8}$ и да је други бајт прочитан из више меморијске локације млађи бајт адресе операнда и да је уписан у осам разреда регистра $MBR_{7...0}$. Стога се у задњем кораку читања адресе операнда из меморије врши истовремено пребацивање садржаја регистра $MBR_{7...0}$ у разреде $MAR_{7...0}$ и садржаја регистра $B_{15...8}$ у разреде $MAR_{15...8}$. Операнд је у меморији на адреси која се налази у разредима $MAR_{15...0}$. Стога се прелази на корак почев од кога се, на већ описани начин, за све операције, сем операције ST и JADR, чита операнд и смешта у регистар $B_{7...0}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 9 и фазу извршавање операција (слика 3.в).

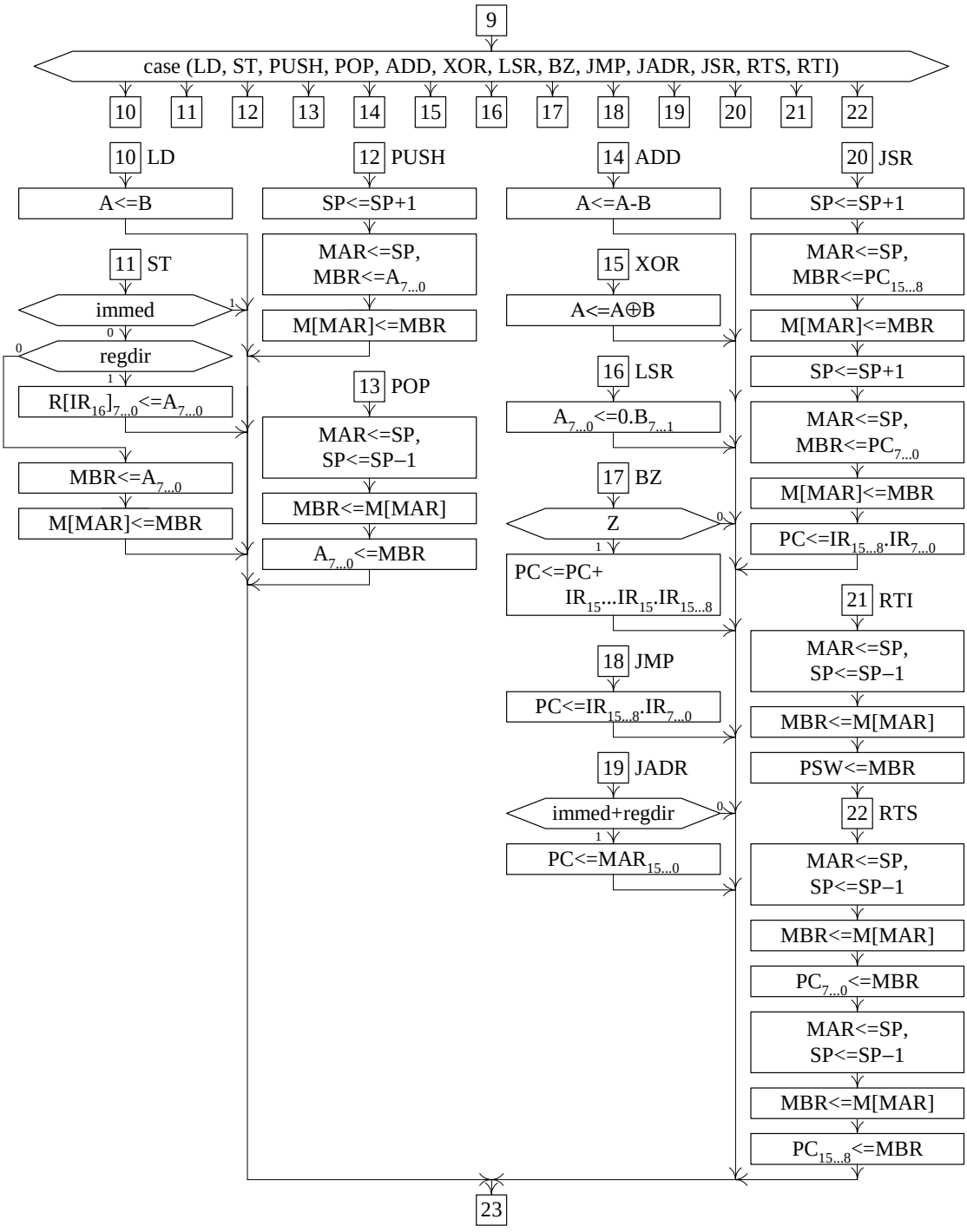
Извршавање операција (слика 3.в)

Извршавање операција се реализује за све инструкције и то од корака 9. У овом кораку се реализује вишеструки условни скок на један од корака 8 до 20 у зависности од тога који од сигнала логичких услова операција **LD** до **RTI** има вредност 1. Сигнали логичких услова операција **LD** до **RTI** (табела 3.б) су бинарно кодирани разредима $IR_{23...16}$ или $IR_{23...19}$ прихватног регистра инструкције и само један од њих може да има вредност 1.

Уколико вредност 1 има сигнал логичког услова **LD**, садржај регистра $B_{7...0}$ се пребацује у регистар акумулатора $A_{7...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 23 и фазу опслуживање прекида (слика 3.г).

Уколико вредност 1 има сигнал логичког услова **ST**, садржај регистра акумулатора $A_{7...0}$ се пребацује у регистар опште намене R[0] или R[1] адресиран садржајем разреда IR_{16} уколико је специфицирано директно регистарско адресирање или у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$ уколико је специфицирано неко од меморијских адресирања. Непосредно адресирање није дозвољено за одредишни операнд и зато се узима да се у случају да се у инструкцији ST појави непосредно адресирање, фаза извршавања ове операције завршава, чиме се ова инструкција претвара у инструкцију без дејства. С тога се проверава вредност сигнала **immed** 1. Уколико сигнал **immed** има вредност 1, специфицирано је непосредно адресирање, па се фаза извршавања операције завршава и прелази на корак 23 и фазу опслуживање прекида (слика 3.г). Уколико сигнал **immed** има вредност 0, проверава се вредност сигнала **regdir**. Уколико сигнал **regdir** има вредност 1, регистарско директно адресирање је специфицирано, па се садржај регистра $A_{7...0}$ уписује у регистар опште намене R[0] или R[1] адресиран садржајем разреда IR_{16} и прелази на корак 23 и фазу опслуживање прекида (слика 3.г). Уколико сигнал **regdir** има вредност 1, неко од меморијских адресирања је специфицирано, па се садржај регистра $A_{7...0}$ уписује у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$. У оквиру тога се садржај регистра $A_{7...0}$ пребацује у регистар $MBR_{7...0}$ и уписује у меморијску локацију на

адреси која се налази у регистру $MAR_{15...0}$. Тиме је завршена фаза извршавање операције и прелази се на корак 21 и фазу опслуживање прекида (слика 3.г).



Слика 3.в Дијаграм тока – фаза извршавање операција

Уколико вредност 1 има сигнал логичког услова **PUSH**, садржај регистра акумулатора $A_{7...0}$ се ставља на стек. То се реализује тако што се најпре инкрементира садржај регистра $SP_{15...0}$. Потом се пребацују садржај регистар $SP_{15...0}$ у регистар $MAR_{15...0}$ и садржај регистра $A_{7...0}$ у регистар $MBR_{7...0}$. Затим се садржај регистра $MBR_{7...0}$ уписује у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$. Треба уочити да стек расте према вишим локацијама и да стек поинтер указује на последњу заузету локацију. С тога се садржај регистра $SP_{15...0}$ прво инкрементира и

после тога пребацује у регистар $MAR_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 23 и фазу опслуживање прекида (слика 3.г).

Уколико вредност 1 има сигнал логичког услова **POP**, садржајем са стека се рестаурира садржај регистра акумулатора $A_{7...0}$. То се реализује тако што се најпре врши пребацивање садржаја регистра $SP_{15...0}$ у регистар $MAR_{15...0}$ и декрементирање садржаја регистра $SP_{15...0}$. Затим се из меморијске локације са адресе која се налази у регистру $MAR_{15...0}$ чита садржај и уписује у регистар $MBR_{7...0}$. На крају се садржај регистра $MBR_{7...0}$ уписује у регистар $A_{7...0}$. Треба уочити да стек расте према нижим локацијама и да стек поинтер указује на задњу заузету локацију. С тога се приликом читања садржаја са стека, садржај регистра $SP_{15...0}$ прво пребацује у регистар $MAR_{15...0}$ и после тога декрементира. Тиме је завршена фаза извршавање операција и прелази се на корак 23 и фазу опслуживање прекида (слика 3.г).

Уколико вредност 1 има сигнал логичког услова **ADD**, садржаји регистара $A_{7...0}$ и $B_{7...0}$ се сабирају и резултат уписује у регистар $A_{7...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 23 и фазу опслуживање прекида (слика 3.г).

Уколико вредност 1 има сигнал логичког услова **XOR**, логичка ексклузивно ИЛИ операција се реализује над садржајима регистара $A_{7...0}$ и $B_{7...0}$ и резултат уписује у регистар $A_{7...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 23 и фазу опслуживање прекида (слика 3.г).

Уколико вредност 1 има сигнал логичког услова **LSR**, садржај регистра $B_{7...0}$ се логички помера удесно за једно место и уписује у регистар $A_{7...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 23 и фазу опслуживање прекида (слика 3.г).

Уколико вредност 1 има сигнал логичког услова **BZ**, условни скок на основу вредности сигнала логичког услова **Z** се реализује. Сигнал **Z** има вредност 1 уколико је резултат задње извршене инструкције 0 и вредност 0 уколико резултат задње извршене инструкције није 0. Уколико сигнал **Z** има вредност 0, услов за скок није испуњен. Тиме је завршена фаза извршавање операција и прелази се на корак 23 и фазу опслуживање прекида (слика 3.г). Уколико сигнал **Z** има вредност 1, услов за скок је испуњен, па се сабирају садржај регистра $PC_{15...0}$, који представља текућу вредност регистра програмског бројача $PC_{15...0}$, и садржај разреда $IR_{15...8}$ проширен знаком на 16 бита, који представља померај, и добијена вредност, која представља адресу скока, уписује у регистар $PC_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 23 и фазу опслуживање прекида (слика 3.г).

Уколико вредност 1 има сигнал логичког услова **JMP**, безусловни скок се реализује. Садржај разреда $IR_{15...8}$.$IR_{7...0}$, који представља адресу скока, уписује се у регистар $PC_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 23 и фазу опслуживање прекида (слика 3.г).

Уколико вредност 1 има сигнал логичког услова **JADR** активан, безусловни скок на срачунату адресу се реализује, под условом да је задато неко од меморијских адресирања. У оквиру тога се садржај разреда $MAR_{15...0}$, који представља срачунату адресу скока задату неким од меморијских адресирања, уписује у регистар $PC_{15...0}$. Непосредно адресирање и регистарско директно адресирање нису дозвољени јер не дају адресу меморијске локације и зато се узима да се у случају да се у инструкцији JADR појави непосредно адресирање или регистарско директно адресирање, фаза извршавања ове операције завршава, чиме се ова инструкција претвара у инструкцију без дејства. С тога се проверава да ли вредност 1 имају сигнал **immed** или сигнал **regdir**. Уколико да, специфицирано је непосредно адресирање или регистарско директно адресирање, па се фаза извршавања операције завршава и прелази на корак 23 и фазу опслуживање

прекида (слика 3.г). Уколико не, специфицирано је неко од меморијских адресирања, па се садржај разреда $MAR_{15...0}$ уписује у регистар $PC_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 23 и фазу опслуживање прекида (слика 3.г).

Уколико вредност 1 има сигнал логичког услова **JSR**, скок на потпрограм се реализује. У оквиру тога се садржај регистра $PC_{15...0}$ ставља на стек. Најпре се инкрементира садржај регистра $SP_{15...0}$. Затим се пребацују садржај регистра $SP_{15...0}$ у регистар $MAR_{15...0}$ и старији бајт регистра $PC_{15...8}$ у регистар $MBR_{7...0}$. Затим се садржај регистра $MBR_{7...0}$ уписује у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$. После тога се поново инкрементира садржај регистра $SP_{15...0}$ и пребацују садржај регистар $SP_{15...0}$ у регистар $MAR_{15...0}$ и млађи бајт регистра $PC_{7...0}$ у регистар $MBR_{7...0}$. На крају се садржај регистра $MBR_{7...0}$ уписује у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$. Треба уочити да стек расте према вишим локацијама и да стек поинтер указује на задњу заузету локацију. С тога се садржај регистра $SP_{15...0}$ прво инкрементира и после тога пребацује у регистар $MAR_{15...0}$. Такође треба уочити да се на стек прво ставља старији а потом млађи бајт регистра $PC_{15...0}$. То је последица начина смештања дво бајтовских величина у меморију, који предвиђа да старији бајт буде на нижој, а млађи бајт на вишој адреси. Затим се садржај разреда $IR_{15...8}.IR_{7...0}$, који представља адресу потпрограма, уписује у регистар $PC_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 23 и фазу опслуживање прекида (слика 3.г).

Уколико вредност 1 има сигнал логичког услова **RTI**, повратак из прекидне рутине се реализује. У оквиру тога се садржајима са стека рестаурирају садржаји регистара $PSW_{7...0}$ и $PC_{15...0}$. Најпре се врши пребацивање садржаја регистра $SP_{15...0}$ у регистар $MAR_{15...0}$ и декрементирање садржаја регистра $SP_{15...0}$. Затим се из меморијске локације са адресе која се налази у регистру $MAR_{15...0}$ чита садржај и уписује у регистар $MBR_{7...0}$. На крају се садржај регистра $MBR_{7...0}$ уписује у регистар $PSW_{7...0}$. На исти начин се са стека читају још два бајта и уписују најпре у млађи а потом у старији бајт регистра $PC_{15...0}$. Треба уочити да стек расте према вишим локацијама и да стек поинтер указује на задњу заузету локацију. С тога се приликом читања садржаја са стека, садржај регистра $SP_{15...0}$ прво пребацује у регистар $MAR_{15...0}$ и после тога декрементира. Такође треба уочити да се са стека прво скида старији а потом млађи бајт регистра $PC_{15...0}$. То је последица начина смештања дво бајтовских величина у меморију, који предвиђа да старији бајт буде на нижој, а млађи бајт на вишој адреси. Тиме је завршена фаза извршавање операција и прелази се на корак 23 и фазу опслуживање прекида (слика 3.г).

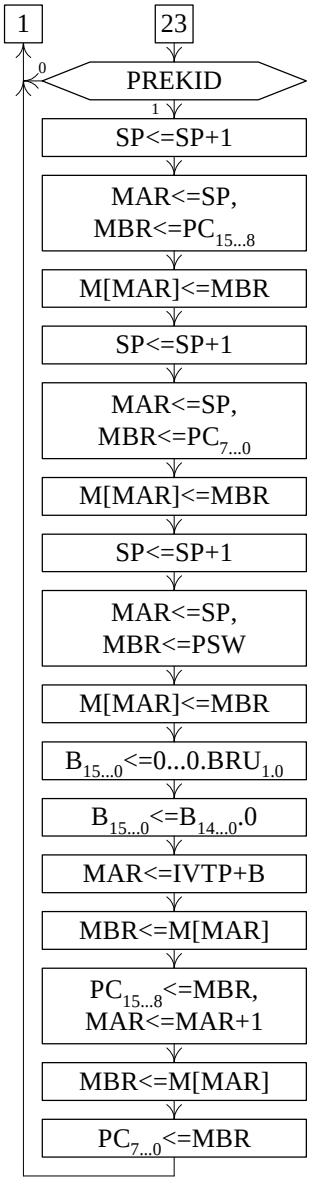
Уколико вредност 1 има сигнал логичког услова **RTS**, повратак из потпрограма се реализује. У оквиру тога се садржајем са стека рестаурира садржај регистра $PC_{15...0}$. Ово се реализује на идентичан као и у случају инструкције RTI. Тиме је завршена фаза извршавање операција и прелази се на корак 23 и фазу опслуживање прекида (слика 3.г).

Опслуживање прекида (слика 3.г)

Опслуживање прекида се реализује за све инструкције и то почев од корака 23. Уколико је сигнал логичког услова **PREKID** има вредност 0, у току извршавања претходних фаза није дошло до генерисања сигнала прекида, па се фаза опслуживање прекида завршава и прелази се на корак 1 и фазу читање инструкције (слика 3.а). Уколико сигнал **PREKID** има вредност 1, у току извршавања претходних фаза дошло је до генерисања сигнала прекида, па се прелази на кораке у оквиру којих се на стеку

стављају садржаји регистара $PC_{15...0}$ и $PSW_{7...0}$ и потом утврђује адреса прекидне рутине и уписује у регистар $PC_{15...0}$. Стављање садржаја регистра $PC_{15...0}$ на стек се реализује на идентичан начин као и у случају инструкције JSR. На исти начин се на стек ставља и садржај регистра $PSW_{7...0}$. Адресе прекидних рутина се налазе у улазима табеле са адресама прекидних рутина. Број улаза у табелу је дат садржајем регистра $BRU_{1.0}$, а почетна адреса табеле садржајем регистра $IVTP_{15...0}$. Најпре се садржај регистра $BRU_{1.0}$ проширен нулама до дужине 16 бита пребацује у регистар $B_{15...0}$, па се садржај регистра $B_{15...0}$ померањем улево за једно место множи са два. Тиме се број улаза претвара у померај. Потом се сабирањем садржаја регистара $IVTP_{15...0}$ и $B_{15...0}$ и смештањем њихове суме у регистар $MAR_{15...0}$ добија адреса на којој се налази адреса прекидне рутине. Са те и следеће адресе из меморије се читају старији и млађи бајт адресе прекидне рутине и уписују у старијих и млађих осам разреда регистра $PC_{15...0}$. Фаза опслуживање прекида је тиме завршена и прелази се на корак 1 и фазу читање инструкције (слика 3.а).

Треба уочити да се јављају две ситуације везане за вредност регистра $PC_{15...0}$ по завршетку фазе опслуживање прекида и преласка на корак 1 и фазу читање инструкције (слика 3.а). Уколико је сигнал **PREKID** имао вредност 0, у регистру $PC_{15...0}$ је адреса прве следеће инструкције после инструкције која је извршена. Уколико је сигнал **PREKID** имао вредност 1, у регистру $PC_{15...0}$ је адреса прве инструкције прекидне рутине.



Слика 3.г Дијаграм тока – фаза опслуживање прекида

4 ЗАДАТАК 4

Посматра се део рачунара који чине меморија и процесор.

Меморија је капацитета 2^{16} бајтова. Ширина меморијске речи је 1 бајт.

Процесор је са једноадресним форматом инструкција. Подаци су целобројне величине без знака дужине 1 бајт.

У процесору постоји програмски бројач PC дужине 2 бајта, адресни регистар меморије MAR дужине 2 бајта, прихватни регистар податка меморије MBR дужине 1 бајт, прихватни регистар инструкције IR дужине 3 бајта, акумулатор A дужине 1 бајт, помоћни регистар B дужине 2 бајта, регистар податка DR дужине 1 бајт, адресни регистар AR дужине 2 бајта, базни регистар BR дужине 2 бајта, индексни регистар XR дужине 2 бајта, програмска статусна реч PSW дужине 1 бајт, указивач на врх стека SP дужине 2 бајта, регистар броја улаза у табелу са адресама прекидних рутина BRU дужине 2 бита и указивач на табелу са адресама прекидних рутина IVTP дужине 2 бајта. Инструкције су дужине 1 или 3 бајта.

Битови 7 до 3 првог бајта инструкције су 00000 за све инструкције скока. Бит 2 првог бајта инструкције има вредност 0 за инструкције условног скока и 1 за инструкције безусловног скока. Инструкција скока је инструкција условног скока уколико је резултат није нула (BNZ). Битовима 1 и 0 првог бајта инструкција специфицира се код операције за инструкције условног скока. На основу тога је за инструкцију BNZ усвојен код операције 00. Инструкција BNZ се реализује као релативни скок у односу на текућу вредност програмског бројача PC, а померај је 8 битна целобројна величина са знаком дата 2. бајтом инструкције. Дужина инструкција је 2 бајта. Инструкције безусловног скока су инструкција безусловног скока (JMP) и инструкција скока на потпрограм (JSR). Битовима 1 и 0 првог бајта инструкција специфицира се код операције за инструкција безусловног скока. На основу тога је за инструкције JMP и JSR усвојени кодови операција 00 и 01, респективно. Инструкције JMP и JSR се реализују као апсолутни скокови, а адреса скока је дата 2. и 3. бајтом инструкције, при чему је млађи бајт адресе скока дат другим бајтом инструкције а старији бајт адресе скока трећим бајтом инструкције. Дужина инструкција је 3 бајта.

Битови 7 до 3 првог бајта инструкције су 11111 за безадресне инструкције. Безадресне инструкције су инструкције повратка из потпрограма (RTS), повратка из прекидне рутине (RTI), стављања садржаја акумулатора на стек (PUSH) и скидања садржаја са стека у акумулатор (POP). Битовима 2 до 0 првог бајта инструкција специфицира се код операције за безадресне инструкције. На основу тога су за инструкције RTS, RTI, PUSH и POP усвојени кодови операција 000, 001, 010 и 011, респективно. Дужина инструкција је 1 бајт.

Битови 7 до 3 првог бајта инструкције у опсегу вредности 00001 до 11110 специфицирају код операције за адресне инструкције. Адресне инструкције су инструкције преноса у акумулатор (LD), инструкција преноса из акумулатора (ST), аритметичка инструкција одузимања (SUB), логичка инструкција комплементирања (NOT), инструкција логичког померања улево за једно место (LSL) и инструкција безусловног скока на срачунату адресу (JADR). У инструкцији ST није дозвољено непосредно адресирање, у инструкцији JADR није дозвољено регистарско директно и непосредно адресирање, па уколико се јаве ова адресирања у овим инструкцијама, инструкције треба да буду без дејства, а инструкција LSL резултат померања смешта у

регистар A. Битовима 6 до 3 првог бајта инструкција специфицира се код операције. На основу тога су за инструкције LD, ST, SUB, NOT, LSL и JADR усвојени кодови операција 00001, 00010, 00011, 00100 и 00101, респективно. Дужина инструкција је 1, 2 или 3 бајта и зависи од специфицираног начина адресирања.

За адресне инструкције се битовима 2, 1 и 0 првог бајта инструкције специфицира начин адресирања и то на следећи начин: 000-регистарско директно адресирање, 001-регистарско индиректно адресирање, 010-PC релативно са померајем адресирање, 011-непосредно адресирање, 100-меморијско директно адресирање, 101-базно адресирање са померајем, 110-индексно адресирање са померајем и 111-базно индексно адресирање са померајем. Код регистарског директног адресирања имплицитно се користи регистар података $DR_{7...0}$. Дужина инструкције је један бајт. Код регистарског индиректног адресирања имплицитно се користи адресни регистар $AR_{15...0}$. Дужина инструкције је један бајт. Код PC релативног са померајем адресирања други бајт инструкције садржи 8 битни померај који је дат као целобројна величина са знаком. Дужина инструкције је два бајта. Код непосредног адресирања други бајт инструкције садржи 8 битни податак. Дужина инструкције је два бајта. Код меморијског директног адресирања други и трећи бајт инструкције садрже адресу меморијске локације, при чему је млађи бајт адресе дат другим а старији бајт адресе трећим бајтом инструкције. Дужина инструкције је три бајта. Код базног адресирања са померајем имплицитно се користи базни регистар $BR_{15...0}$, док други бајт инструкције садржи 8 битни померај који је дат као целобројна величина са знаком. Дужина инструкције је два бајта. Код индексног адресирања са померајем имплицитно се користи индексни регистар $XR_{15...0}$, док други бајт инструкције садржи 8 битни померај који је дат као целобројна величина са знаком. Дужина инструкције је два бајта. Код базно индексног адресирања са померајем имплицитно се користе регистри $BR_{15...0}$ и $XR_{15...0}$, а други бајт инструкције садржи 8 битни померај који је дат као целобројна величина са знаком. Дужина инструкција је 2 бајта.

Стек расте према вишим меморијским локацијама, а регистар SP указује на прву слободну меморијску локацију.

Захтеви за прекид долазе од 4 улазно/излазна уређаја по линијама означеним од 0 до 3. По линији 0 стиже захтев за прекид најнижег, а по линији 3 највишег приоритета. Број линије највишег приоритета по којој је стигао захтев за прекид налази се у бинарном облику у регистру BRU дужине 2 разреда. Адресе прекидних рутина 4 улазно/излазна уређаја који по линијама означеним од 0 до 3 шаљу захтеве за прекид налазе се у улазима 0 до 3 табеле са адресама прекидних рутина. Адресе дужине 16 бита заузимају по две суседне меморијске локације, при чему се млађи бајт налази на нижој а старији бајт на вишој адреси Садржај регистра BRU представља број улаза у табелу са адресам пекидних рутина. Почетна адреса табеле са адресама прекидних рутина се налази у регистру IVTP дужине 2 бајта. У оквиру хардверског дела опслуживања захтева за прекид на стек са стављају само регистри PC и PSW.

Нацртати и објаснити дијаграм тока фаза извршавања инструкције и то: фазе читања инструкције, фазе формирања адресе и читања операнда, фаза извршавања операција LO, ST, PUSH, POP, SUB, NOT, LSL, BNZ, JMP, JADR, JSR, RTS и RTI и фазе опслуживања захтева за прекид.

РЕШЕЊЕ

Дијаграм тока извршавања инструкције је дат на сликама 4.а, 4.б, 4.в и 4.г. Извршавање инструкције се састоји из четири фазе: читање инструкције (слика 4.а), формирање адресе и читање операнда (слика 4.б), извршавање операција (слика 4.в) и опслуживање прекида (слика 4.г). Дијаграм тока је дат у сагласности са форматима

инструкција (табела 4.а), при чему први, други и трећи бајт инструкције се смештају у разреде 23 до 16, 15 до 8 и 7 до 0 прихватног регистра инструкције IR. У њему се користе сигнали логичких услова операција (табела 4.б), начина адресирања (табела 4.в) и дужина инструкција (табела 4.г) који се формирају на основу разреда 23 до 16 прихватног регистра инструкције. У дијаграму тока се користе и сигнали логичких услова **Z** и **PREKID**. Сигнал **Z** представља један од индикатора програмске статусне речи PSW који вредностима 1 и 0 указује да ли је резултат задње извршене инструкције нула или различит од нуле, респективно, док сигнал **PREKID** вредностима 1 и 0 указује да ли је током извршавања инструкције дошло или није дошло до генерисања прекида, респективно. Генерисање сигнала **Z** и **PREKID** је ван оквира овог задатка и није приказано.

Табела 4.а Формати инструкција

Instrukcija uslovnog skoka BNZ

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
0	0	0	0	0	0	OC		disp							

Instrukcije bezuslovnog skoka JMP i JSR

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	OC		low								high							

Bezadresne instrukcije PUSH, POP, RTS i RTI

23	22	21	20	19	18	17	16
1	1	1	1	1	OC		

Adresne instrukcije LD, ST, SUB, NOT, LSL i JADR sa registarskim direktnim i registarskim indirektnim adresiranjima*

23	22	21	20	19	18	17	16
OC					0	0	x

Adresne instrukcije LD, ST, SUB, NOT, LSL i JADR sa PC relativnim sa pomerajem i neposrednim adresiranjima *

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
OC					0	1	x	immed/disp							

Adresne instrukcije LD, ST, SUB, NOT, LSL i JADR sa memorijskim direktnim adresiranjem*

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC					1	0	0	low								high							

Adresne instrukcije LD, ST, SUB, NOT, LSL i JADR sa baznim adresiranjem sa pomerajem*

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
OC					1	0	1	disp							

Adresne instrukcije LD, ST, SUB, NOT, LSL i JADR sa indeksnim i bazno indeksnim adresiranjima*

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
OC					1	1	x	disp							

*U polju OC nisu dozvoljene vrednosti 00000 i 11111 koje se koriste kao kodovi izuzetka za instrukcije skoka i bezadresne instrukcije

Читање инструкције (слика 4.а)

Инструкција се чита из меморије почев од адресе на коју указују тренутка вредност регистра програмског бројача $PC_{15...0}$. То се реализује тако што се чита бајт по бајт и после сваког прочитаног бајта садржај регистра $PC_{15...0}$ се инкрементира. Прочитани бајтови инструкције се смештају у прихватни регистар инструкције $IR_{23...0}$. У зависности од типа инструкције читају се 1, 2 или 3 бајта и смештају у разреде $IR_{23...16}$, $IR_{15...8}$ и $IR_{7...0}$. Обавезно се чита први бајт инструкције. У оквиру тога се, најпре,

садржај регистра $PC_{15...0}$ пребацује у регистар $MAR_{15...0}$ и инкрементира садржај регистра $PC_{15...0}$. Затим се из меморије M са адресе која се налази у регистру $MAR_{15...0}$ чита бајт и уписује у регистар $MBR_{7...0}$. Садржај регистра $MBR_{7...0}$ се, на крају, пребацује у разреде $IR_{23...16}$.

Табела 4.б Сигнали операција

$BNZ=\overline{IR_{23}}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot\overline{IR_{20}}\cdot\overline{IR_{19}}\cdot\overline{IR_{18}}\cdot\overline{IR_{17}}\cdot\overline{IR_{16}}$

$JMP=\overline{IR_{23}}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot\overline{IR_{20}}\cdot\overline{IR_{19}}\cdot IR_{18}\cdot\overline{IR_{17}}\cdot\overline{IR_{16}}$

$JSR=\overline{IR_{23}}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot\overline{IR_{20}}\cdot\overline{IR_{19}}\cdot IR_{18}\cdot\overline{IR_{17}}\cdot IR_{16}$

$PUSH=IR_{23}\cdot IR_{22}\cdot IR_{21}\cdot IR_{20}\cdot IR_{19}\cdot\overline{IR_{18}}\cdot\overline{IR_{17}}\cdot\overline{IR_{16}}$

$POP=IR_{23}\cdot IR_{22}\cdot IR_{21}\cdot IR_{20}\cdot IR_{19}\cdot\overline{IR_{18}}\cdot\overline{IR_{17}}\cdot IR_{16}$

$RTS=IR_{23}\cdot IR_{22}\cdot IR_{21}\cdot IR_{20}\cdot IR_{19}\cdot\overline{IR_{18}}\cdot IR_{17}\cdot\overline{IR_{16}}$

$RTI=IR_{23}\cdot IR_{22}\cdot IR_{21}\cdot IR_{20}\cdot IR_{19}\cdot\overline{IR_{18}}\cdot IR_{17}\cdot IR_{16}$

$LD=\overline{IR_{23}}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot\overline{IR_{20}}\cdot IR_{19}$

$ST=\overline{IR_{23}}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot IR_{20}\cdot\overline{IR_{19}}$

$SUB=\overline{IR_{23}}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot IR_{20}\cdot IR_{19}$

$NOT=\overline{IR_{23}}\cdot\overline{IR_{22}}\cdot IR_{21}\cdot\overline{IR_{20}}\cdot\overline{IR_{19}}$

$LSL=\overline{IR_{23}}\cdot\overline{IR_{22}}\cdot IR_{21}\cdot\overline{IR_{20}}\cdot IR_{19}$

$JADR=\overline{IR_{23}}\cdot\overline{IR_{22}}\cdot IR_{21}\cdot IR_{20}\cdot\overline{IR_{19}}$

Табела 4.в Сигнали начина адресирања

$regdir=\overline{IR_{18}}\cdot\overline{IR_{17}}\cdot\overline{IR_{16}}$

$regind=\overline{IR_{18}}\cdot\overline{IR_{17}}\cdot IR_{16}$

$pcrelpom=\overline{IR_{18}}\cdot IR_{17}\cdot\overline{IR_{16}}$

$immed=\overline{IR_{18}}\cdot IR_{17}\cdot IR_{16}$

$memdir=IR_{18}\cdot\overline{IR_{17}}\cdot\overline{IR_{16}}$

$brpom=IR_{18}\cdot\overline{IR_{17}}\cdot IR_{16}$

$xrpom=IR_{18}\cdot IR_{17}\cdot\overline{IR_{16}}$

$brxrpom=IR_{18}\cdot IR_{17}\cdot IR_{16}$

Табела 4.г Сигнали дужина инструкција

$bezadr1=PUSH+POP+RTS+RTI$

$adr1=(LD+ST+SUB+NOT+LSL+JADR)\cdot(regdir+regind)$

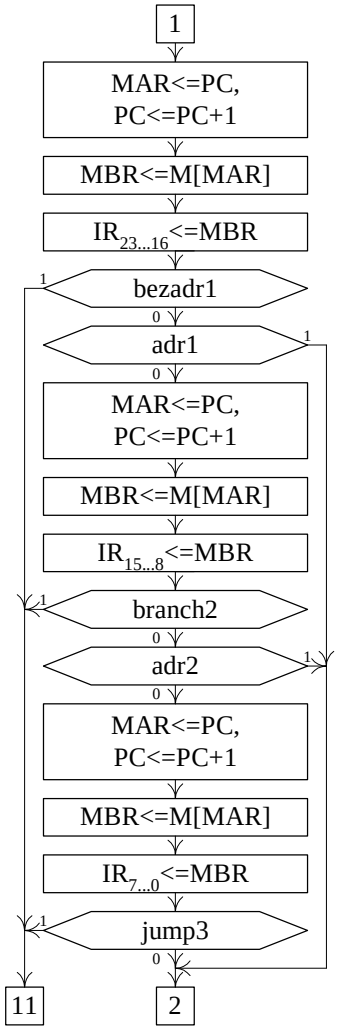
$branch2=BNZ$

$adr2=(LD+ST+SUB+NOT+LSL+JADR)\cdot(pcrelpom+immed+brpom+xrpom+brxrpom)$

$jmp3=JMP+JSR$

$adr3=(LD+ST+SUB+NOT+LSL+JADR)\cdot memdir$

Сада се врши провера сигнала логичког услова дужине инструкције **bezadr1**. Уколико је његова вредност 1, ради се о безадресној инструкцији чија је дужина 1 бајт. Тиме је завршена фаза читање инструкције и прелази се на корак 11 и фазу извршавање операција (слика 4.в). Уколико је његова вредност 0, врши се провера сигнала логичког услова дужине инструкције **adr1**. Уколико је његова вредност 1, ради се о адресној инструкцији чија је дужина 1 бајт. Тиме је завршена фаза читање инструкције и прелази се на корак 2 и фазу формирање адресе и читање операнда (слика 4.б). Уколико је његова вредност 0, ради се о некој од инструкција чија дужина може да буде 2 или 3 бајта. Зато се у овом случају, најпре, чита 2. бајт инструкције и уписује у разреде $IR_{15...8}$. Читање 2. бајта инструкције се реализује на сличан начин као и читање 1. бајта инструкције.



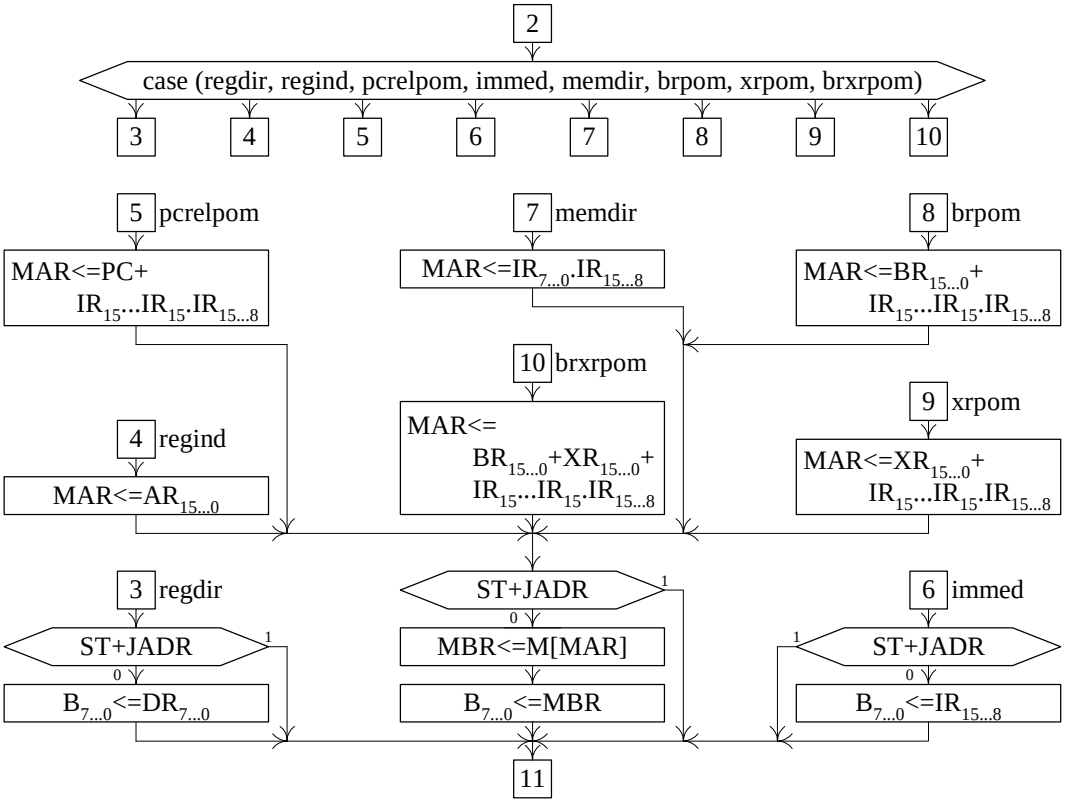
Слика 4.а Дијаграм тока – фаза читање инструкције

Затим се врши провера сигнала логичког услова дужине инструкције **branch2**. Уколико је његова вредност 1, ради се о инструкцији условног скока чија је дужина 2 бајта. Тиме је завршена фаза читање инструкције и прелази се на корак 11 и фазу извршавање операција (слика 4.в). Уколико је његова вредност 0, врши се провера сигнала логичког услова дужине инструкције **adr2**. Уколико је његова вредност 1, ради се о адресној инструкцији чија је дужина 2 бајта. Тиме је завршена фаза читање инструкције и прелази се на корак 2 и фазу формирање адресе и читање операнда (слика 4.б). Уколико је његова вредност 0, ради се о некој од инструкција чија је дужина може да буде 3 бајта. Зато се у овом случају, најпре, чита 3. бајт инструкције и уписује у разреде $IR_{7...0}$. Читање 3. бајта инструкције се реализује на сличан начин као и читање 1. бајта инструкције.

Сада се врши провера сигнала логичког услова дужине инструкције **jump3**. Уколико је његова вредност 1, ради се о инструкцији безусловног скока чија је дужина 3 бајта. Тиме је завршена фаза читање инструкције и прелази се на корак 11 и фазу извршавање операција (слика 4.в). Уколико је његова вредност 0, ради се о адресној инструкцији чија је дужина 3 бајта. Тиме је завршена фаза читање инструкције и прелази се на корак 2 и фазу формирање адресе и читање операнда (слика 4.б).

Формирање адресе и читање операнда (слика 4.б)

Формирање адресе и читање операнда се реализује само за адресне инструкције и то од корака 2. У овом кораку се реализује вишеструки условни скок на један од корака 3 до 10 у зависности од тога који од сигнала логичких услова начина адресирања **regdir** до **brxrpom** има вредност 1. Сигнали логичких услова начина адресирања **regdir** до **brxrpom** (табела 4.в) су бинарно кодирани разредима $IR_{18...16}$ прихватног регистра инструкције и само један од њих може да има вредност 1.



Слика 4.б Дијаграм тока – фаза формирање адресе и читање операнда

Уколико вредност 1 има сигнал логичког услова **regdir**, ради се о регистарском директном адресирању и прелазу са корака 2 на корак 3. Операнд је тада у регистру података $DR_{7...0}$. Садржај регистра $DR_{7...0}$ се пребацује у нижих 8 разреда помоћног регистра $B_{7...0}$ за све операције сем операција ST и JADR. Због тога се овде врши провера да ли један од сигнала логичких услова операција **ST** или **JADR** има вредност 1. Уколико оба сигнала **ST** и **JADR** имају вредност 0, регистар $DR_{7...0}$ се пребацује у нижих 8 разреда помоћног регистра $B_{7...0}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 11 и фазу извршавање операција (слика 4.в). Уколико један од сигнала **ST** и **JADR** има вредност 1, одмах се завршава фаза формирање адресе и читање операнда и прелази се на корак 11 и фазу извршавање операција (слика 4.в).

Уколико вредност 1 има сигнал логичког услова **regind**, ради се о регистарском индиректном адресирању и прелазу са корака 2 на корак 4. Операнд је тада у меморији на адреси која се налази у адресном регистру $AR_{15...0}$. Садржај регистра $AR_{15...0}$ се пребацује у регистар $MAR_{15...0}$ и прелази на корак почев од кога се за све операције, сем операција ST и JADR, чита операнд и смешта у нижих 8 разреда помоћног регистра $B_{7...0}$. Због тога се овде врши провера да ли један од сигнала логичких услова операција **ST** или **JADR** има вредност 1. Уколико оба сигнала **ST** и **JADR** имају вредност 0, из меморије M се са адресе која се налази у регистру $MAR_{15...0}$, најпре, чита бајт и уписује у регистар $MBR_{7...0}$, а затим се садржај регистра $MBR_{7...0}$ пребацује у нижих 8 разреда

помоћног регистра $B_{7...0}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 11 и фазу извршавање операција (слика 4.в). Уколико један од сигнала **ST** и **JADR** има вредност 1, одмах се завршава фаза формирање адресе и читање операнда и прелази се на корак 11 и фазу извршавање операција (слика 4.в).

Уколико вредност 1 има сигнал логичког услова **pcrelpom**, ради се о PC релативном са померајем адресирању и прелазу са корака 2 на корак 5. Операнд је тада у меморији на адреси која се добија сабирањем садржаја програмског бројача $PC_{15...0}$ и помераја који се добија проширивањем разреда $IR_{15...8}$ знаком на 16 битну вредност. Добијена адреса се пребацује у регистар $MAR_{15...0}$ и прелази на корак почев од кога се, на већ описани начин, за све операције, сем операција ST и JADR, чита операнд и смешта у нижих 8 разреда помоћног регистра $B_{7...0}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 11 и фазу извршавање операција (слика 4.в).

Уколико вредност 1 има сигнал логичког услова **immed**, ради се о непосредном адресирању и прелазу са корака 2 на корак 6. Операнд се тада налази у разредима $IR_{15...8}$ чији се садржај пребацује у нижих 8 разреда помоћног регистра $B_{7...0}$ за све операције сем операција ST и JADR. Због тога се овде врши провера да ли један од сигнала логичких услова операција **ST** или **JADR** има вредност 1. Уколико оба сигнала **ST** и **JADR** имају вредност 0, разреди $IR_{15...8}$ се пребацује у регистар $B_{7...0}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 11 и фазу извршавање операција (слика 4.в). Уколико један од сигнала **ST** и **JADR** има вредност 1, одмах се завршава фаза формирање адресе и читање операнда и прелази се на корак 11 и фазу извршавање операција (слика 4.в).

Уколико вредност 1 има сигнал логичког услова **memdir**, ради се о меморијском директном адресирању и прелазу са корака 2 на корак 7. Операнд је тада у меморији на адреси која се налази у разредима $IR_{7...0}.IR_{15...8}$. Садржај регистра $IR_{7...0}.IR_{15...8}$ се пребацује у регистар $MAR_{15...0}$ и прелази на корак почев од кога се, на већ описани начин, за све операције, сем операција ST и JADR, чита операнд и смешта у нижих 8 разреда помоћног регистра $B_{7...0}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 11 и фазу извршавање операција (слика 4.в).

Уколико вредност 1 има сигнал логичког услова **brpom**, ради се о базном адресирању са померајем и прелазу са корака 2 на корак 8. Операнд је тада у меморији на адреси која се добија сабирањем садржаја базног регистра $BR_{15...0}$ и помераја који се добија проширивањем разреда $IR_{15...8}$ знаком на 16 битну вредност. Добијена адреса се пребацује у регистар $MAR_{15...0}$ и прелази на корак почев од кога се, на већ описани начин, за све операције, сем операција ST и JADR, чита операнд и смешта у нижих 8 разреда помоћног регистра $B_{7...0}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 11 и фазу извршавање операција (слика 4.в).

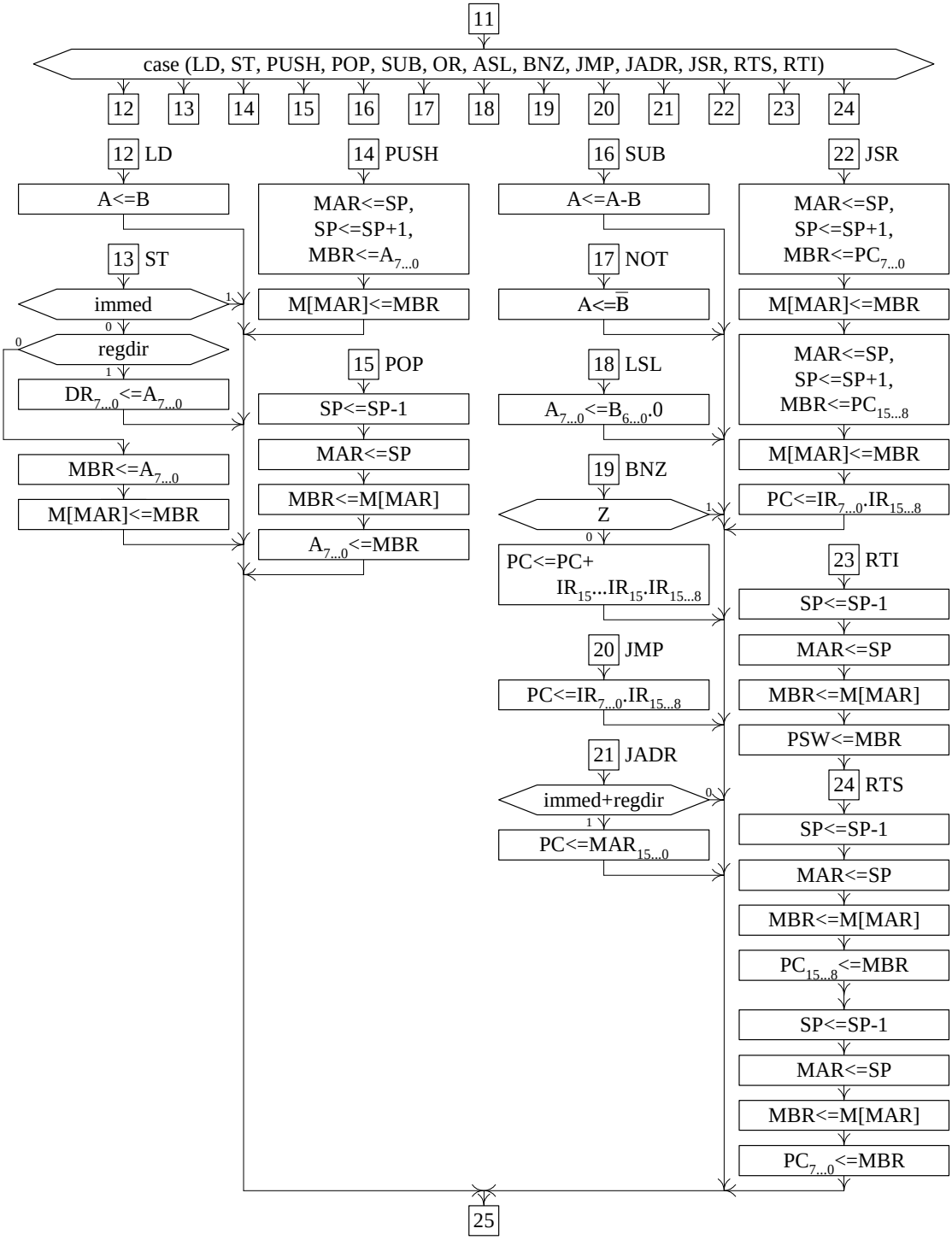
Уколико вредност 1 има сигнал логичког услова **xrpom**, ради се о индексном адресирању са померајем и прелазу са корака 2 на корак 9. Операнд је тада у меморији на адреси која се добија сабирањем садржаја индексног регистра $XR_{15...0}$ и помераја који се добија проширивањем разреда $IR_{15...8}$ знаком на 16 битну вредност. Добијена адреса се пребацује у регистар $MAR_{15...0}$ и прелази на корак почев од кога се, на већ описани начин, за све операције, сем операција ST и JADR, чита операнд и смешта у нижих 8 разреда помоћног регистра $B_{7...0}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 11 и фазу извршавање операција (слика 4.в).

Уколико вредност 1 има сигнал логичког услова **brxrpom**, ради се о базно индексном адресирању са померајем и прелазу са корака 2 на корак 10. Операнд је тада у меморији на адреси која се добија сабирањем садржаја базног регистра $BR_{15...0}$, индексног регистра $XR_{15...0}$ и помераја који се добија проширивањем разреда $IR_{15...8}$ знаком на 16

битну вредност. Добијена адреса се пребацује у регистар $MAR_{15...0}$ и прелази на корак почев од кога се, на већ описани начин, за све операције, сем операција ST и JADR, чита операнд и смешта у нижих 8 разреда помоћног регистра $B_{7...0}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 11 и фазу извршавање операција (слика 4.в).

Извршавање операција (слика 4.в)

Извршавање операција се реализује за све инструкције и то од корака 9. У овом кораку се реализује вишеструки условни скок на један од корака 12 до 24 у зависности од тога који од сигнала логичких услова операција **LD** до **RTI** има вредност 1. Сигнали логичких услова операција **LD** до **RTI** (tabela 4.b) су бинарно кодирани разредима $IR_{23...16}$ или $IR_{23...19}$ прихватног регистра инструкције и само један од њих може да има вредност 1.



Слика 4.в Дијаграм тока – фаза извршавање операција

Уколико вредност 1 има сигнал логичког услова **LD**, садржај регистра $B_{7...0}$ се пребацује у регистар акумулатора $A_{7...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 4.г).

Уколико вредност 1 има сигнал логичког услова **ST**, садржај регистра акумулатора $A_{7...0}$ се пребацује у регистар података $DR_{7...0}$ уколико је специфицирано директно регистарско адресирање или у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$ уколико је специфицирано неко од меморијских адресирања. Непосредно адресирање није дозвољено за одредишни операнд и зато се узима да се, у случају да се у инструкцији ST појави непосредно адресирање, фаза извршавања ове операције

завршава, чиме се ова инструкција претвара у инструкцију без дејства. С тога се проверава вредност сигнала **immed** 1. Уколико сигнал **immed** има вредност 1, специфицирано је непосредно адресирање, па се фаза извршавања операције завршава и прелази на корак 25 и фазу опслуживање прекида (слика 4.г). Уколико сигнал **immed** има вредност 0, проверава се вредност сигнала **regdir**. Уколико сигнал **regdir** има вредност 1, регистарско директно адресирање је специфицирано, па се садржај регистра $A_{7...0}$ пребацује у регистар података $DR_{7...0}$ и прелази на корак 25 и фазу опслуживање прекида (слика 4.г). Уколико сигнал **regdir** има вредност 1, неко од меморијских адресирања је специфицирано, па се садржај регистра $A_{7...0}$ уписује у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$. У оквиру тога се садржај регистра $A_{7...0}$ пребацује у регистар $MBR_{7...0}$ и уписује у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$. Тиме је завршена фаза извршавање операције и прелази се на корак 25 и фазу опслуживање прекида (слика 4.г).

Уколико вредност 1 има сигнал логичког услова **PUSH**, садржај регистра акумулатора $A_{7...0}$ се ставља на стек. То се реализује тако што се најпре пребацују садржај регистра $SP_{15...0}$ у регистар $MAR_{15...0}$ и садржај регистра $A_{7...0}$ у регистар $MBR_{7...0}$ и инкрементира садржај регистра $SP_{15...0}$. Затим се садржај регистра $MBR_{7...0}$ уписује у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$. Треба уочити да стек расте према вишим локацијама и да стек поинтер указује на прву слободну локацију. С тога се садржај регистра $SP_{15...0}$ прво пребацује у регистар $MAR_{15...0}$ и после тога инкрементира. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 4.г).

Уколико вредност 1 има сигнал логичког услова **POP**, садржај са стека се скида у регистар акумулатора $A_{7...0}$. То се реализује тако што се најпре врши декрементирање садржаја регистра $SP_{15...0}$ и потом пребацивање садржаја регистра $SP_{15...0}$ у регистар $MAR_{15...0}$. Затим се из меморијске локације са адресе која се налази у регистру $MAR_{15...0}$ чита садржај и уписује у регистар $MBR_{7...0}$. На крају се садржај регистра $MBR_{7...0}$ уписује у регистар $A_{7...0}$. Треба уочити да стек расте према вишим локацијама и да стек поинтер указује на прву слободну локацију. С тога се приликом читања садржаја са стека, садржај регистра $SP_{15...0}$ прво декрементира, па после пребацује у регистар $MAR_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 4.г).

Уколико вредност 1 има сигнал логичког услова **CUB**, садржај регистра $B_{7...0}$ се одузима од садржаја регистра $A_{7...0}$ и резултат уписује у регистар $A_{7...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 4.г).

Уколико вредност 1 има сигнал логичког услова **NOT**, логичка операција комплементирања садржаја регистра $B_{7...0}$ се реализује и резултат уписује у регистар $A_{7...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 4.г).

Уколико вредност 1 има сигнал логичког услова **LSL**, садржај регистра $B_{7...0}$ се логички помера улево за једно место и уписује у регистар $A_{7...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 4.г).

Уколико вредност 1 има сигнал логичког услова **BNZ**, условни скок на основу вредности сигнала логичког услова **Z** се реализује. Сигнал **Z** има вредност 1 уколико је резултат задње извршене инструкције 0 и вредност 0 уколико резултат задње извршене инструкције није 0. Уколико сигнал **Z** има вредност 1, услов за скок није испуњен. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање

прекида (слика 4.г). Уколико сигнал **Z** има вредност 0, услов за скок је испуњен, па се сабирају садржај регистра $PC_{15...0}$, који представља текућу вредност регистра програмског бројача $PC_{15...0}$, и садржај разреда $IR_{15...8}$ проширен знаком на 16 бита, који представља померај, и добијена вредност, која представља адресу скока, уписује у регистар $PC_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 4.г).

Уколико вредност 1 има сигнал логичког услова **JMP**, безусловни скок се реализује. Садржај разреда $IR_{7...0}.IR_{15...8}$, који представља адресу скока, уписује се у регистар $PC_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 4.г).

Уколико вредност 1 има сигнал логичког услова **JADR** активан, безусловни скок на срачунату адресу се реализује, под условом да је задато неко од меморијских адресирања. У оквиру тога се садржај разреда $MAR_{15...0}$, који представља срачунату адресу скока задату неким од меморијских адресирања, уписује у регистар $PC_{15...0}$. Непосредно адресирање и регистарско директно адресирање нису дозвољени јер не дају адресу меморијске локације и зато се узима да се у случају да се у инструкцији JADR појави непосредно адресирање или регистарско директно адресирање, фаза извршавања ове операције завршава, чиме се ова инструкција претвара у инструкцију без дејства. С тога се проверава да ли вредност 1 имају сигнал **immed** или сигнал **regdir**. Уколико да, специфицирано је непосредно адресирање или регистарско директно адресирање, па се фаза извршавања операције завршава и прелази на корак 25 и фазу опслуживање прекида (слика 4.г). Уколико не, специфицирано је неко од меморијских адресирања, па се садржај разреда $MAR_{15...0}$ уписује у регистар $PC_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 4.г).

Уколико вредност 1 има сигнал логичког услова **JSR**, скок на потпрограм се реализује. У оквиру тога се садржај регистра $PC_{15...0}$ ставља на стек. Најпре се пребацују садржај регистра $SP_{15...0}$ у регистар $MAR_{15...0}$ и млађи бајт регистра $PC_{7...0}$ у регистар $MBR_{7...0}$ и инкрементира садржај регистра $SP_{15...0}$. Затим се садржај регистра $MBR_{7...0}$ уписује у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$. После тога се поново пребацују садржај регистра $SP_{15...0}$ у регистар $MAR_{15...0}$ и старији бајт регистра $PC_{15...8}$ у регистар $MBR_{7...0}$ и инкрементира садржај регистра $SP_{15...0}$. Затим се поново садржај регистра $MBR_{7...0}$ уписује у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$. Треба уочити да стек расте према вишим локацијама и да стек поинтер указује на прву слободну локацију. С тога се садржај регистра $SP_{15...0}$ прво пребацује у регистар $MAR_{15...0}$ и после тога инкрементира. Такође треба уочити да се на стек прво ставља млађи а потом старији бајт регистра $PC_{15...0}$. То је последица начина смештања дво бајтовских величина у меморију, који предвиђа да млађи бајт буде на нижој, а старији бајт на вишој адреси. На крају се садржај разреда $IR_{7...0}.IR_{15...8}$, који представља адресу потпрограма, уписује у регистар $PC_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 4.г).

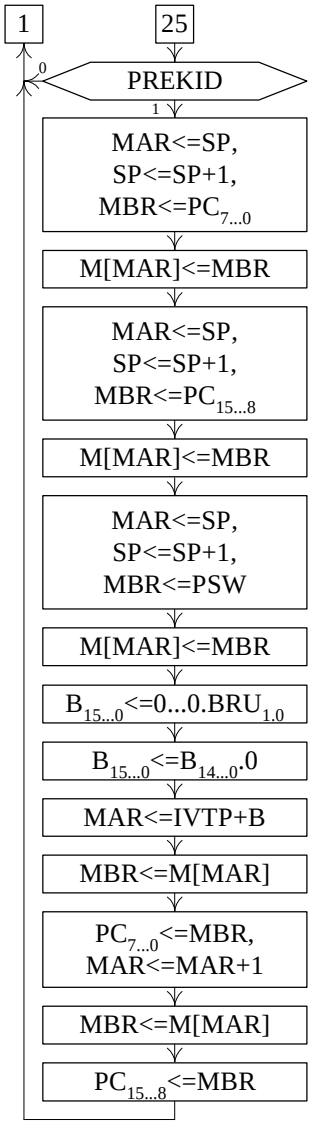
Уколико вредност 1 има сигнал логичког услова **RTI**, повратак из прекидне рутине се реализује. У оквиру тога се садржајима са стека рестаурирају садржаји регистара $PSW_{7...0}$ и $PC_{15...0}$. Најпре се врши декрементирање садржаја регистра $SP_{15...0}$ и потом пребацивање садржаја регистра $SP_{15...0}$ у регистар $MAR_{15...0}$. Затим се из меморијске локације са адресе која се налази у регистру $MAR_{15...0}$ чита садржај и уписује у регистар $MBR_{7...0}$. На крају се садржај регистра $MBR_{7...0}$ уписује у регистар $PSW_{7...0}$. На исти начин се са стека читају још два бајта и уписују најпре у старији а потом у млађи бајт

регистра $PC_{15...0}$. Треба уочити да стек расте према вишим локацијама и да стек поинтер указује на задњу заузету локацију. С тога се приликом читања садржаја са стека, садржај регистра $SP_{15...0}$ прво декрементира и после тога пребацује у регистар $MAR_{15...0}$. Такође треба уочити да се са стека прво скида старији а потом млађи бајт регистра $PC_{15...0}$. То је последица начина смештања дво бајтовских величина у меморију, који предвиђа да млађи бајт буде на нижој, а старији бајт на вишој адреси. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 4.г).

Уколико вредност 1 има сигнал логичког услова **RTS**, повратак из потпрограма се реализује. У оквиру тога се садржајем са стека рестаурира садржај регистра $PC_{15...0}$. Ово се реализује на идентичан као и у случају инструкције RTI. Тиме је завршена фаза извршавање операција и прелази се на корак 25 и фазу опслуживање прекида (слика 4.г).

Опслуживање прекида (слика 4.г)

Опслуживање прекида се реализује за све инструкције и то почев од корака 25.



Слика 4.г Дијаграм тока – фаза опслуживање прекида

Уколико је сигнал логичког услова **PREKID** има вредност 0, у току извршавања претходних фаза није дошло до генерисања сигнала прекида, па се фаза опслуживање прекида завршава и прелази се на корак 1 и фазу читање инструкције (слика 4.а).

Уколико сигнал **PREKID** има вредност 1, у току извршавања претходних фаза дошло је до генерисања сигнала прекида, па се прелази на кораке у оквиру којих се на стеку стављају садржаји регистара $PC_{15...0}$ и $PSW_{7...0}$ и потом утврђује адреса прекидне рутине и уписује у регистар $PC_{15...0}$. Стављање садржаја регистра $PC_{15...0}$ на стек се реализује на идентичан начин као и у случају инструкције JSR. На исти начин се на стек ставља и садржај регистра $PSW_{7...0}$. Адресе прекидних рутина се налазе у улазима табеле са адресама прекидних рутина. Број улаза у табелу је дат садржајем регистра $BRU_{1.0}$, а почетна адреса табеле садржајем регистра $IVTP_{15...0}$. Najpre se sadržaj registra $BRU_{1.0}$ проширен нулама до дужине 16 бита пребацује у регистар $B_{15...0}$, па се садржај регистра $B_{15...0}$ померањем улево за једно место множи са два. Тиме се број улаза претвара у померај. Потом се сабирањем садржаја регистара $IVTP_{15...0}$ и $B_{15...0}$ и смештањем њихове суме у регистар $MAR_{15...0}$ добија адреса на којој се налази адреса прекидне рутине. Са те и следеће адресе из меморије се читају млађи и старији бајт адресе прекидне рутине и уписују у млађих и старијих осам разреда регистра $PC_{15...0}$. Фаза опслуживање прекида је тиме завршена и прелази се на корак 1 и фазу читање инструкције (слика 4.а).

Треба уочити да се јављају две ситуације везане за вредност регистра $PC_{15...0}$ по завршетку фазе опслуживање прекида и преласка на корак 1 и фазу читање инструкције (слика 4.а). Уколико је сигнал **PREKID** имао вредност 0, у регистру $PC_{15...0}$ је адреса прве следеће инструкције после инструкције која је извршена. Уколико је сигнал **PREKID** имао вредност 1, у регистру $PC_{15...0}$ је адреса прве инструкције прекидне рутине.

5 ЗАДАТАК 5

Посматра се део рачунара који чине меморија и процесор.

Меморија је капацитета 2^{16} бајтова. Ширина меморијске речи је један бајт.

Процесор је са једноадресним форматом инструкција. Подаци су целобројне величине без знака дужине два бајта.

У процесору постоји програмски бројач PC дужине два бајта, адресни регистар меморије MAR дужине два бајта, прихватни регистар података меморије MBR дужине један бајт, прихватни регистар инструкције IR дужине три бајта, акумулатор A дужине два бајта, помоћни регистар B дужине два бајта, регистри опште намене R[0], R[1], R[2] и R[3] дужине два бајта, програмска статусна реч PSW дужине један бајт, указивач на врх стека SP дужине два бајта, регистар броја улаза у табелу са адресама прекидних рутина BRU дужине 2 бита и указивач на табелу са адресама прекидних рутина IVTP дужине два бајта. Инструкције су дужине један, два или три бајта.

Бит 7 првог бајта инструкције има вредност 0 за безадресне инструкције и инструкције скока, док бит 6 првог бајта инструкције има вредност 0 за безадресне инструкције и вредност 1 за инструкције скока. Безадресне инструкције су инструкција повратка из потпрограма (RTS), инструкција повратка из прекидне рутине (RTI), инструкција стављања садржаја акумулатора на стек (PUSH) и инструкција скидања садржаја са стека у акумулатор (POP). Битовима 5 до 0 првог бајта инструкција специфицира се код операције за безадресне инструкције. На основу тога су за инструкције RTS, RTI, PUSH и POP усвојени кодови операција 000000, 000001, 000010 и 000011, респективно. Дужина инструкција је један бајт. Бит 5 првог бајта инструкције има вредност 0 за инструкције условног скока и 1 за инструкције безусловног скока. Инструкција условног скока је инструкција условног скока уколико је резултат нула (BZ). Битовима 4 до 0 првог бајта инструкција специфицира се код операције за инструкције условног скока. На основу тога је за инструкцију BZ усвојен код операције 00000. Инструкција BZ се реализује као релативни скок у односу на текућу вредност програмског бројача PC, а померај је 8 битна целобројна величина са знаком дата другим бајтом инструкције. Дужина инструкције је два бајта. Инструкције безусловног скока су инструкција безусловног скока (JMP) и инструкција скока на потпрограм (JSR). Битовима 4 до 0 првог бајта инструкција специфицира се код операције за инструкције безусловног скока. На основу тога су за инструкције JMP и JSR усвојени кодови операција 00000 и 00001, респективно. Инструкције JMP и JSR се реализују као апсолутни скокови, а адреса скока је дата другим и трећим бајтом инструкције, при чему је млађи бајт адресе скока дат другим бајтом инструкције а старији бајт адресе скока трећим бајтом инструкције. Дужина инструкција је три бајта.

Бит 7 првог бајта инструкције има вредност 1 за адресне инструкције. Адресне инструкције су инструкција преноса у акумулатор (LD), инструкција преноса из акумулатора (ST), аритметичка инструкција сабирања (ADD), логичка инструкција ексклузивно ИЛИ (XOR), инструкција логичког померања удесно за једно место (LSR) и инструкција безусловног скока на срачунату адресу (JADR). У инструкцији ST није дозвољено непосредно адресирање, у инструкцији JADR није дозвољено регистарско директно и непосредно адресирање, па уколико се јаве ова адресирања у овим инструкцијама, инструкције треба да буду без дејства, а инструкција LSR резултат померања смешта у регистар A. Битовима 6 до 3 првог бајта инструкција специфицира

се код операције. На основу тога су за инструкције LD, ST, ADD, XOR, LSR и JADR усвојени кодови операција 0000, 0001, 0010, 0011, 0100 и 0101, респективно. Дужина инструкција је један, два или три бајта и зависи од специфицираног начина адресирања.

За адресне инструкције се битовима 2, 1 и 0 првог бајта инструкције специфицира начин адресирања и регистар опште намене уколико се користи у задатом начину адресирања. Уколико бит 2 има вредност 0, ради се о регистарском директном адресирању код кога се користи један од регистара опште намене R[0], R[1], R[2] и R[3] специфициран вредностима 00, 01, 10 и 11 битова 1 и 0. Дужина инструкције је један бајт. Уколико бит 2 има вредност 1, ради се о непосредном, PC релативном са померајем, меморијском директном или меморијском индиректном адресирању специфицираном вредностима 00, 01, 10 и 11 битова 1 и 0. Код непосредног адресирања други и трећи бајт инструкције садрже 16 битни податак, при чему је млађи бајт податка дат другим а старији бајт податка трећим бајтом инструкције. Дужина инструкције је три бајта. Код PC релативног са померајем адресирања други бајт инструкције садржи 8 битни померај који је дат као целобројна величина са знаком. Дужина инструкције је два бајта. Код меморијског директног и меморијског индиректног адресирања други и трећи бајт инструкције садрже адресу меморијске локације, при чему је млађи бајт адресе дат другим а старији бајт адресе трећим бајтом инструкције. Дужина инструкције је три бајта.

Стек расте према нижим меморијским локацијама, а регистар SP указује на задњу заузету меморијску локацију.

Захтеви за прекид долазе од 4 улазно/излазна уређаја по линијама означеним од 0 до 3. По линији 0 стиже захтев за прекид најнижег, а по линији 3 највишег приоритета. Број линије највишег приоритета по којој је стигао захтев за прекид налази се у бинарном облику у регистру BRU дужине два разреда. Адресе прекидних рутина четири улазно/излазна уређаја који по линијама означеним од 0 до 3 шаљу захтеве за прекид налазе се у улазима 0 до 3 табеле са адресама прекидних рутина. Адресе дужине 16 бита заузимају по две суседне меморијске локације, при чему се млађи бајт адресе налази на нижој а старији бајт адресе на вишој адреси Садржај регистра BRU представља број улаза у табелу са адресама прекидних рутина. Почетна адреса табеле са адресама прекидних рутина се налази у регистру IVTP. У оквиру хардверског дела опслуживања захтева за прекид на стек са стављају само регистри PC и PSW.

Нацртати и објаснити дијаграм тока фаза извршавања инструкције и то: фазе читања инструкције, фазе формирања адресе и читања операнда, фаза извршавања операција LD, ST, PUSH, POP, ADD, XOR, LSR, BZ, JMP, JADR, JSR, RTS и RTI и фазе опслуживања захтева за прекид.

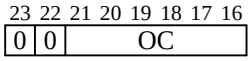
РЕШЕЊЕ

Дијаграм тока извршавања инструкције је дат на сликама 5.а, 5.б, 5.в и 5.г. Извршавање инструкције се састоји из четири фазе: читање инструкције (слика 5.а), формирање адресе и читање операнда (слика 5.б), извршавање операција (слика 5.в) и опслуживање прекида (слика 5.г). Дијаграм тока је дат у сагласности са форматима инструкција (табела 5.а), при чему први, други и трећи бајт инструкције се смештају у разреде 23 до 16, 15 до 8 и 7 до 0 прихватног регистра инструкције IR. У њему се користе сигнали логичких услова операција (табела 5.б), начина адресирања (табела 5.в) и дужина инструкција (табела 5.г) који се формирају на основу разреда 23 до 16 прихватног регистра инструкције. У дијаграму тока се користе и сигнали логичких услова **Z** и **PREKID**. Сигнал **Z** представља један од индикатора програмске статусне речи ПСW који вредностима 1 и 0 указује да ли је резултат задње извршене инструкције

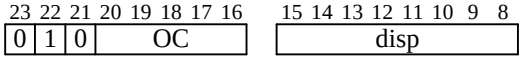
нула или различит од нуле, респективно, док сигнал **PREKID** вредностима 1 и 0 указује да ли је током извршавања инструкције дошло или није дошло до генерисања прекида, респективно. Генерисање сигнала **Z** и **PREKID** је ван оквира овог задатка и није приказано.

Табела 5.а Формати инструкција

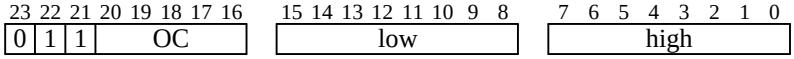
Bezadresne instrukcije PUSH, POP, RTS i RTI



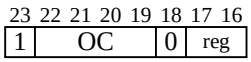
Instrukcija uslovnog skoka BZ



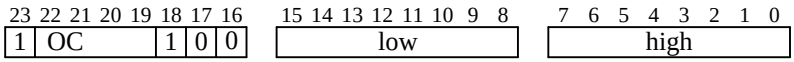
Instrukcije bezuslovnog skoka JMP i JSR



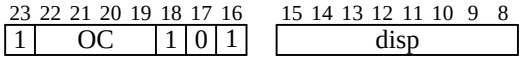
Adresne instrukcije LD, ST, ADD, XOR, LSR i JADR sa registarskim direktnim adresiranjem



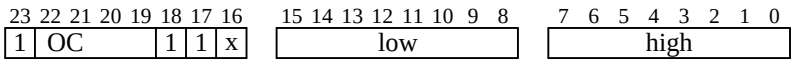
Adresne instrukcije LD, ST, ADD, XOR, LSR i JADR sa neposrednim adresiranjem



Adresne instrukcije LD, ST, ADD, XOR, LSR i JADR sa PC relativnim sa pomerajem adresiranjem



Adresne instrukcije LD, ST, ADD, XOR, LSR i JADR sa memorijskim direktnim i memorijskim indirektnim adresiranjima*



* x je 0 ili 1

Табела 5.б Сигнали операција

$PUSH=\overline{IR_{23}}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot\overline{IR_{20}}\cdot\overline{IR_{19}}\cdot\overline{IR_{18}}\cdot\overline{IR_{17}}\cdot\overline{IR_{16}}$

$POP=\overline{IR_{23}}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot\overline{IR_{20}}\cdot\overline{IR_{19}}\cdot\overline{IR_{18}}\cdot\overline{IR_{17}}\cdot IR_{16}$

$RTS=\overline{IR_{23}}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot\overline{IR_{20}}\cdot\overline{IR_{19}}\cdot\overline{IR_{18}}\cdot IR_{17}\cdot\overline{IR_{16}}$

$RTI=\overline{IR_{23}}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot\overline{IR_{20}}\cdot\overline{IR_{19}}\cdot\overline{IR_{18}}\cdot IR_{17}\cdot IR_{16}$

$BZ=\overline{IR_{23}}\cdot IR_{22}\cdot\overline{IR_{21}}\cdot\overline{IR_{20}}\cdot\overline{IR_{19}}\cdot\overline{IR_{18}}\cdot\overline{IR_{17}}\cdot\overline{IR_{16}}$

$JMP=\overline{IR_{23}}\cdot IR_{22}\cdot IR_{21}\cdot\overline{IR_{20}}\cdot\overline{IR_{19}}\cdot\overline{IR_{18}}\cdot\overline{IR_{17}}\cdot\overline{IR_{16}}$

$JSR=\overline{IR_{23}}\cdot IR_{22}\cdot IR_{21}\cdot\overline{IR_{20}}\cdot\overline{IR_{19}}\cdot\overline{IR_{18}}\cdot\overline{IR_{17}}\cdot IR_{16}$

$LD=IR_{23}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot\overline{IR_{20}}\cdot\overline{IR_{19}}$

$ST=IR_{23}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot\overline{IR_{20}}\cdot IR_{19}$

$ADD=IR_{23}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot IR_{20}\cdot\overline{IR_{19}}$

$XOR=IR_{23}\cdot\overline{IR_{22}}\cdot\overline{IR_{21}}\cdot IR_{20}\cdot IR_{19}$

$LSR=IR_{23}\cdot\overline{IR_{22}}\cdot IR_{21}\cdot\overline{IR_{20}}\cdot\overline{IR_{19}}$

$JADR=IR_{23}\cdot\overline{IR_{22}}\cdot IR_{21}\cdot\overline{IR_{20}}\cdot IR_{19}$

Табела 5.в Сигнали начина адресирања

regdir=$\overline{IR_{18}}$

immed=$IR_{18}\cdot\overline{IR_{17}}\cdot\overline{IR_{16}}$

pcrelpom=$IR_{18}\cdot\overline{IR_{17}}\cdot IR_{16}$

memdir=$IR_{18}\cdot IR_{17}\cdot\overline{IR_{16}}$

memind=$IR_{18}\cdot IR_{17}\cdot IR_{16}$

Табела 5.г Сигнали дужина инструкција

bezadr1=PUSH+POP+RTS+RTI

adr1=(LD+ST+ADD+XOR+LSR+JADR)·regdir

branch2=BZ

adr2=(LD+ST+ADD+XOR+LSR+JADR)·pcrelpom

jmp3=JMP+JSR

adr3=(LD+ST+ADD+XOR+LSR+JADR)·(immed+memdir+memind)

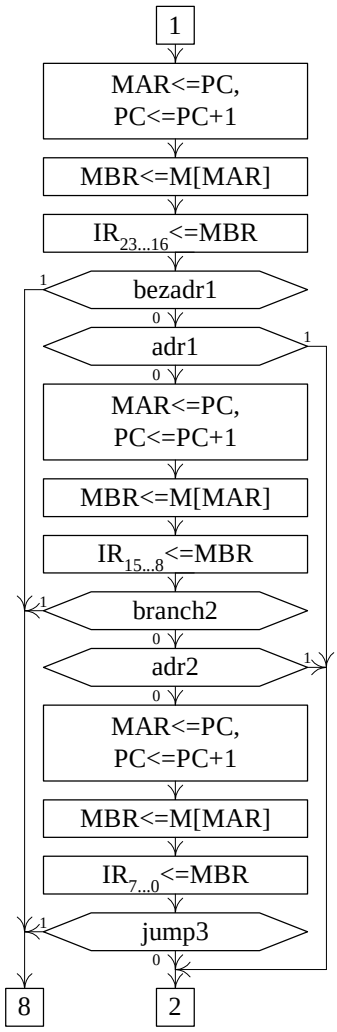
Читање инструкције (слика 5.а)

Инструкција се чита из меморије почев од адресе на коју указују тренутка вредност регистра програмског бројача $PC_{15...0}$. То се реализује тако што се чита бајт по бајт и после сваког прочитаног бајта садржај регистра $PC_{15...0}$ се инкрементира. Прочитани бајтови инструкције се смештају у прихватни регистар инструкције $IR_{23...0}$. У зависности од типа инструкције читају се један, два или три бајта и смештају у разреде $IR_{23...16}$, $IR_{15...8}$ и $IR_{7...0}$. Обавезно се чита први бајт инструкције. У оквиру тога се, најпре, садржај регистра $PC_{15...0}$ пребацује у регистар $MAR_{15...0}$ и инкрементира садржај регистра $PC_{15...0}$. Затим се из меморије M са адресе која се налази у регистру $MAR_{15...0}$ чита бајт и уписује у регистар $MBR_{7...0}$. Садржај регистра $MBR_{7...0}$ се, на крају, пребацује у разреде $IR_{23...16}$.

Сада се врши провера сигнала логичког услова дужине инструкције **bezadr1**. Уколико је његова вредност 1, ради се о безадресној инструкцији чија је дужина један бајт. Тиме је завршена фаза читање инструкције и прелази се на корак 8 и фазу извршавање операција (слика 5.в). Уколико је његова вредност 0, врши се провера сигнала логичког услова дужине инструкције **adr1**. Уколико је његова вредност 1, ради се о адресној инструкцији чија је дужина 1 бајт. Тиме је завршена фаза читање инструкције и прелази се на корак 2 и фазу формирање адресе и читање операнда (слика 5.б). Уколико је његова вредност 0, ради се о некој од инструкција чија дужина може да буде 2 или 3 бајта. Зато се у овом случају, најпре, чита други бајт инструкције и уписује у разреде $IR_{15...8}$. Читање другог бајта инструкције се реализује на сличан начин као и читање првог бајта инструкције.

Затим се врши провера сигнала логичког услова дужине инструкције **branch2**. Уколико је његова вредност 1, ради се о инструкцији условног скока чија је дужина два бајта. Тиме је завршена фаза читање инструкције и прелази се на корак 8 и фазу извршавање операција (слика 5.в). Уколико је његова вредност 0, врши се провера сигнала логичког услова дужине инструкције **adr2**. Уколико је његова вредност 1, ради се о адресној инструкцији чија је дужина два бајта. Тиме је завршена фаза читање инструкције и прелази се на корак 2 и фазу формирање адресе и читање операнда (слика 5.б). Уколико је његова вредност 0, ради се о некој од инструкција чија је дужина три бајта. Зато се у овом случају, најпре, чита трећи бајт инструкције и уписује у разреде

$IR_{7...0}$. Читање трећег бајта инструкције се реализује на сличан начин као и читање првог бајта инструкције.



Слика 5.а Дијаграм тока – фаза читање инструкције

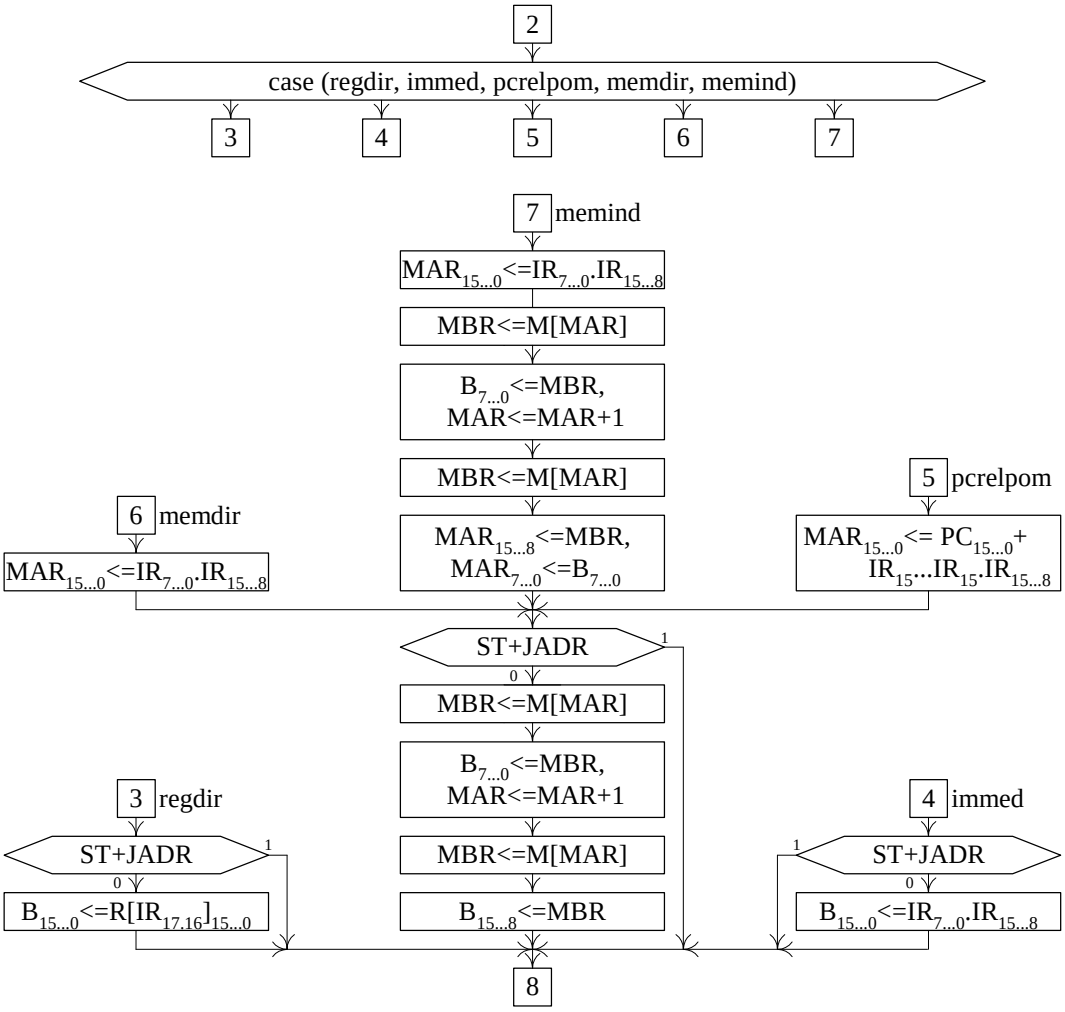
Сада се врши провера сигнала логичког услова дужине инструкције **jump3**. Уколико је његова вредност 1, ради се о инструкцији безусловног скока чија је дужина три бајта. Тиме је завршена фаза читање инструкције и прелази се на корак 8 и фазу извршавање операција (слика 5.в). Уколико је његова вредност 0, ради се о адресној инструкцији чија је дужина три бајта. Тиме је завршена фаза читање инструкције и прелази се на корак 2 и фазу формирање адресе и читање операнда (слика 5.б).

Формирање адресе и читање операнда (слика 5.б)

Формирање адресе и читање операнда се реализује само за адресне инструкције и то од корака 2. У овом кораку се реализује вишеструки условни скок на један од корака 3 до 7 у зависности од тога који од сигнала логичких услова начина адресирања **regdir** до **memind** има вредност 1. Сигнали логичких услова начина адресирања **regdir** до **memind** (табела 5.в) су бинарно кодирани разредом IR_{18} или разредима $IR_{18...16}$ прихватног регистра инструкције и само један од њих може да има вредност 1.

Уколико вредност 1 има сигнал логичког услова **regdir**, ради се о регистарском директном адресирању и прелазу са корака 2 на корак 3. Операнд је тада у једном од регистара опште намене R[0], R[1], R[2] или R[3] адресираном садржајем разреда

$IR_{17.16}$. Адресирани регистар R[0], R[1], R[2] или R[3] се пребацује у регистар $B_{15...0}$ за све операције сем операција ST и JADR. Због тога се овде врши провера да ли један од сигнала логичких услова операција **ST** или **JADR** ima vrednost 1. Ukoliko oba signala **ST** и **JADR** имају вредност 0, адресирани регистар R[0], R[1], R[2] или R[3] се пребацује у регистар $B_{15...0}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 8 и фазу извршавање операција (слика 5.в). Уколико један од сигнала **ST** и **JADR** има вредност 1, одмах се завршава фаза формирање адресе и читање операнда и прелази се на корак 8 и фазу извршавање операција (слика 5.в).



Слика 5.б Дијаграм тока – фаза формирање адресе и читање операнда

Уколико вредност 1 има сигнал логичког услова **immed**, ради се о непосредном адресирању и прелазу са корака 2 на корак 4. Операнд се тада налази у разредима $IR_{15...8}$ и $IR_{7...0}$ чији се садржај пребацује у регистар $B_{7...0}$ и $B_{15...8}$ за све операције сем операција ST и JADR. Због тога се овде врши провера да ли један од сигнала логичких услова операција **ST** или **JADR** има вредност 1. Уколико оба сигнала **ST** и **JADR** имају вредност 0, разреди $IR_{15...8}$ се пребацују у разреде регистар $B_{7...0}$ и разреди $IR_{7...0}$ у разреде $B_{15...8}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 8 и фазу извршавање операција (слика 5.в). Уколико један од сигнала **ST** и **JADR** има вредност 1, завршава се фаза формирање адресе и читање операнда и прелази се на корак 8 и фазу извршавање операција (слика 5.в).

Уколико вредност 1 има сигнал логичког услова **pcrelpom**, ради се о PC релативном са померајем адресирању и прелазу са корака 2 на корак 5. Операнд је тада у меморији

на адреси која се добија сабирањем садржаја програмског бројача $PC_{15...0}$ и помераја који се добија проширивањем разреда $IR_{15...8}$ знаком на 16 битну вредност. Добијена адреса се пребацује у регистар $MAR_{15...0}$ и прелази на корак почев од кога се за све операције, сем операција ST и JADR, чита операнд и смешта у регистар $B_{15...0}$. Због тога се овде врши провера да ли један од сигнала логичких услова операција **ST** или **JADR** има вредност 1. Уколико оба сигнала **ST** и **JADR** имају вредност 0, из меморије M се са адресе која се налази у регистру $MAR_{15...0}$, најпре, чита бајт и уписује у регистар $MBR_{7...0}$, а затим се садржај регистра $MBR_{7...0}$ пребацује у разреде $B_{7...0}$ и врши инкрементирање садржаја регистра $MAR_{15...0}$. Како је операнд ширине два бајта а ширина меморијске речи један бајт, потребно је из меморије прочитати још један бајт. С тога се из меморије M са адресе која се налази у регистру $MAR_{15...0}$, најпре, чита бајт и уписује у регистар $MBR_{7...0}$, а затим се садржај регистра $MBR_{7...0}$ пребацује у разреде $B_{15...8}$. Треба уочити да је први бајт операнда прочитан из ниже меморијске локације млађи бајт операнда и да је стога уписан у млађих осам разреда регистра $B_{7...0}$ и да је други бајт операнда прочитан из више меморијске локације старији бајт операнда и да је стога уписан у старијих осам разреда регистра $B_{15...8}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 8 и фазу извршавање операција (слика 5.в). Уколико један од сигнала **ST** и **JADR** има вредност 1, прелази се одмах на корак 8 и фазу извршавање операција (слика 5.в).

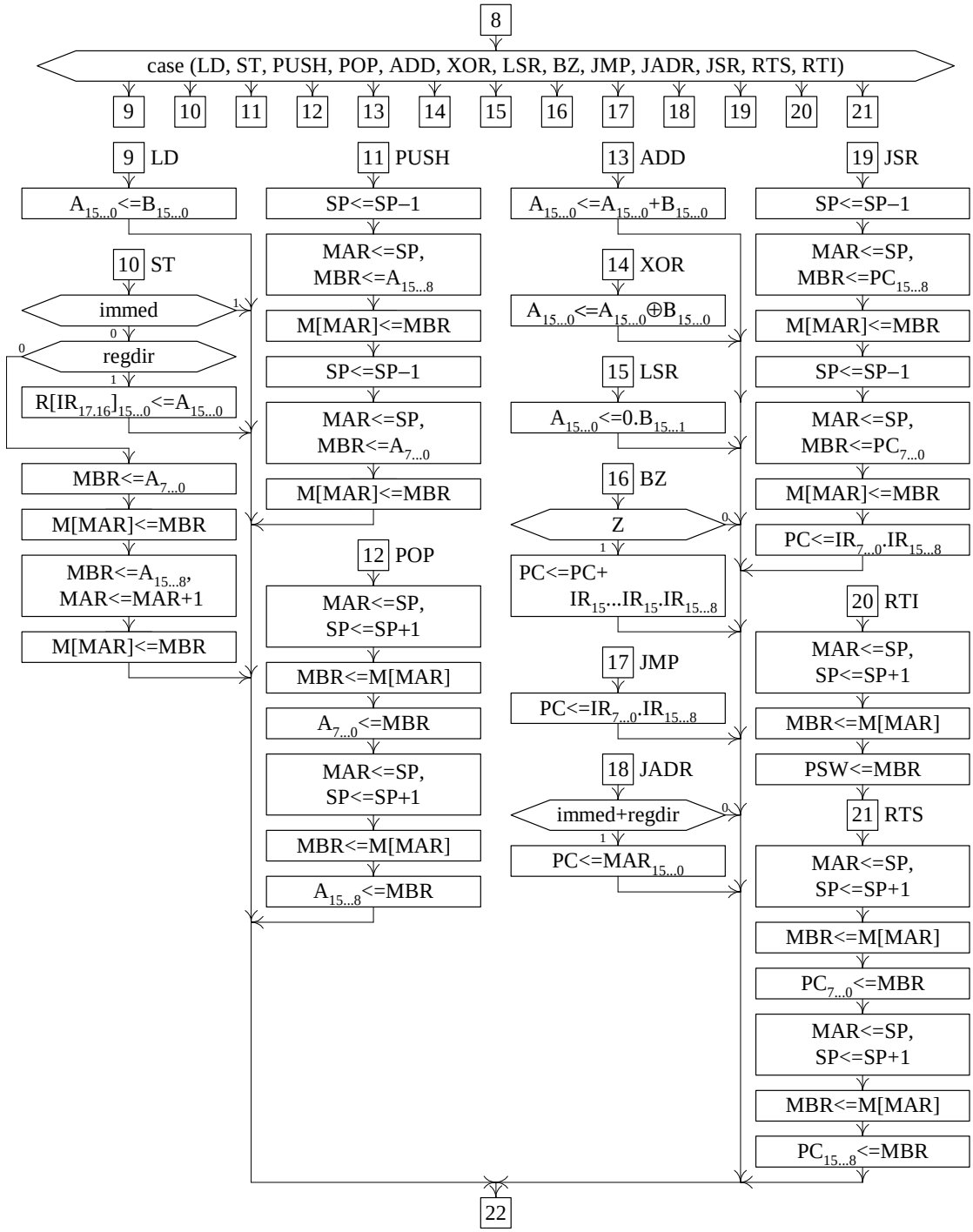
Уколико вредност 1 има сигнал логичког услова **memdir**, ради се о меморијском директном адресирању и прелазу са корака 2 на корак 6. Операнд је тада у меморији на адреси која се налази у разредима $IR_{7...0}.IR_{15...8}$. Садржај $IR_{7...0}.IR_{15...8}$ се пребацује у регистар $MAR_{15...0}$ и прелази на корак почев од кога се, на већ описани начин, за све операције, сем операција ST и JADR, чита операнд и смешта у регистар $B_{15...0}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 8 и фазу извршавање операција (слика 5.в).

Уколико вредност 1 има сигнал логичког услова **memind**, ради се о меморијском индиректном адресирању и прелазу са корака 2 на корак 7. Операнд је тада у меморији, а адреса меморијске локације је у другој меморијској локацији чија се адреса налази у разредима $IR_{7...0}.IR_{15...8}$. Стога се садржај разреда $IR_{7...0}.IR_{15...8}$ пребацује у регистар $MAR_{15...0}$. Из меморије M се са адресе која се налази у регистру $MAR_{15...0}$, најпре, чита млађи бајт адресе операнда и уписује у регистар $MBR_{7...0}$, а затим се садржај регистра $MBR_{7...0}$ привремено пребацује у разреде $B_{7...0}$ и врши инкрементирање садржаја регистра $MAR_{15...0}$. Како је адреса ширине два бајта а ширина меморијске речи један бајт, потребно је из меморије прочитати још један бајт. С тога се из меморије M са адресе која се налази у регистру $MAR_{15...0}$, затим, чита старији бајт адресе и уписује у регистар $MBR_{7...0}$. Треба уочити да је први бајт прочитан из ниже меморијске локације млађи бајт адресе операнда и да је привремено уписан у осам нижих разреда регистра $B_{7...0}$ и да је други бајт прочитан из више меморијске локације старији бајт адресе операнда и да је уписан у осам разреда регистра $MBR_{7...0}$. Стога се у задњем кораку читања адресе операнда из меморије врши истовремено пребацивање садржаја регистра $MBR_{7...0}$ у разреде $MAR_{15...8}$ и садржаја регистра $B_{7...0}$ у разреде $MAR_{7...0}$. Операнд је у меморији на адреси која се налази у разредима $MAR_{15...0}$. Стога се прелази на корак почев од кога се, на већ описани начин, за све операције, сем операција ST и JADR, чита операнд и смешта у регистар $B_{15...0}$. Тиме је завршена фаза формирање адресе и читање операнда и прелази се на корак 8 и фазу извршавање операција (слика 5.в).

Извршавање операција (слика 5.в)

Извршавање операција се реализује за све инструкције и то од корака 8. У овом кораку се реализује вишеструки условни скок на један од корака 9 до 21 у зависности од тога који од сигнала логичких услова операција **LD** до **RTI** има вредност 1. Сигнали логичких услова операција **LD** до **RTI** (табела 5.б) су бинарно кодирани разредима $IR_{23...16}$ или $IR_{23...19}$ прихватног регистра инструкције и само један од њих може да има вредност 1.

Уколико вредност 1 има сигнал логичког услова **LD**, садржај регистра $B_{15...0}$ се пребацује у регистар акумулатора $A_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 22 и фазу опслуживање прекида (слика 5.г).



Слика 5.в Дијаграм тока – фаза извршавање операција

Уколико вредност 1 има сигнал логичког услова **ST**, садржај регистра акумулатора $A_{15...0}$ се пребацује у један од регистара опште намене R[0], R[1], R[2] или R[3] адресиран садржајем разреда $IR_{17.16}$ уколико је специфицирано директно регистарско адресирање или у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$ уколико је специфицирано неко од меморијских адресирања. Непосредно адресирање није дозвољено за одредишни операнд и зато се узима да се у случају да се у инструкцији ST појави непосредно адресирање, фаза извршавања ове операције завршава, чиме се ова инструкција претвара у инструкцију без дејства. С тога се проверава вредност сигнала **immed** 1. Уколико сигнал **immed** има вредност 1, специфицирано је непосредно адресирање, па се фаза извршавања операције завршава и прелази на корак 22 и фазу опслуживање прекида (слика 5.г). Уколико сигнал **immed** има вредност 0, проверава се вредност сигнала **regdir**. Уколико сигнал **regdir** има вредност 1, регистарско директно адресирање је специфицирано, па се садржај регистра $A_{15...0}$ уписује у један од регистара опште намене R[0], R[1], R[2] или R[3] адресиран садржајем разреда $IR_{17.16}$ и прелази на корак 22 и фазу опслуживање прекида (слика 5.г). Уколико сигнал **regdir** има вредност 0, неко од меморијских адресирања је специфицирано, па се садржај регистра $A_{15...0}$ уписује у две суседне меморијске локације почев од меморијске локације чија се адреса налази у регистру $MAR_{15...0}$. У оквиру тога се, најпре, млађи бајт регистра $A_{7...0}$ пребацује у регистар $MBR_{7...0}$ и уписује у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$. Затим се, пошто се старији бајт регистра $A_{15...8}$ пребаци у регистар $MBR_{7...0}$ и садржај регистра $MAR_{15...0}$ инкрементира, садржај регистра $MBR_{7...0}$ уписује у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 22 и фазу опслуживање прекида (слика 5.г).

Уколико вредност 1 има сигнал логичког услова **PUSH**, садржај регистра акумулатора $A_{15...0}$ се ставља на стек. Најпре се декрементира садржај регистра $SP_{15...0}$, па се пребацују садржај регистра $SP_{15...0}$ у регистар $MAR_{15...0}$ и старији бајт регистра $A_{15...8}$ у регистар $MBR_{7...0}$. Затим се садржај регистра $MBR_{7...0}$ уписује у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$. После тога се поново најпре декрементира садржај регистра $SP_{15...0}$, па се пребацују садржај регистар $SP_{15...0}$ у регистар $MAR_{15...0}$ и млађи бајт регистра $A_{7...0}$ у регистар $MBR_{7...0}$. На крају се садржај регистра $MBR_{7...0}$ уписује у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$. Треба уочити да стек расте према нижим локацијама и да регистар $SP_{15...0}$ указује на задњу заузету локацију. С тога се садржај регистра $SP_{15...0}$ прво декрементира па после тога пребацује у регистар $MAR_{15...0}$. Такође треба уочити да се на стек прво ставља старији а потом млађи бајт регистра $A_{15...0}$. То је последица начина смештања дво бајтовских величина у меморији, који предвиђа да млађи бајт буде на нижој, а старији бајт на вишој адреси. Тиме је завршена фаза извршавање операција и прелази се на корак 22 и фазу опслуживање прекида (слика 5.г).

Уколико вредност 1 има сигнал логичког услова **POP**, садржајем са стека се рестаурира регистар акумулатора $A_{15...0}$. Најпре се садржај регистра $SP_{15...0}$ пребацује у регистар $MAR_{15...0}$ и инкрементира. Затим се из меморијске локације са адресе која се налази у регистру $MAR_{15...0}$ чита садржај и уписује у регистар $MBR_{7...0}$. На крају се садржај регистра $MBR_{7...0}$ уписује у млађи бајт регистра $A_{7...0}$. После тога се поново садржај регистра $SP_{15...0}$ пребацује у регистар $MAR_{15...0}$ и инкрементира. Затим се из меморијске локације са адресе која се налази у регистру $MAR_{15...0}$ чита садржај и уписује у регистар $MBR_{7...0}$. На крају се садржај регистра $MBR_{7...0}$ уписује у старији бајт регистра $A_{15...8}$. Треба уочити да стек расте према нижим локацијама и да регистар $SP_{15...0}$ указује на задњу заузету локацију. С тога се приликом читања садржаја са стека

садржај регистра $SP_{15...0}$ прво пребацује у регистар $MAR_{15...0}$ и после инкрементира. Такође треба уочити да се са стека прво скида млађи а потом старији бајт регистра $A_{15...0}$. То је последица начина смештања дво бајтовских величина у меморији, који предвиђа да млађи бајт буде на нижој, а старији бајт на вишој адреси. Тиме је завршена фаза извршавање операција и прелази се на корак 22 и фазу опслуживање прекида (слика 5.г).

Уколико вредност 1 има сигнал логичког услова **ADD**, садржаји регистара $A_{15...0}$ и $B_{15...0}$ се сабирају и резултат уписује у регистар $A_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 22 и фазу опслуживање прекида (слика 5.г).

Уколико вредност 1 има сигнал логичког услова **XOR**, логичка ексклузивно ИЛИ операција се реализује над садржајима регистара $A_{15...0}$ и $B_{15...0}$ и резултат уписује у регистар $A_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 22 и фазу опслуживање прекида (слика 5.г).

Уколико вредност 1 има сигнал логичког услова **LSR**, садржај регистра $B_{15...0}$ се логички помера удесно за једно место и уписује у регистар $A_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 22 и фазу опслуживање прекида (слика 5.г).

Уколико вредност 1 има сигнал логичког услова **BZ**, условни скок на основу вредности сигнала логичког услова **Z** се реализује. Сигнал **Z** има вредност 1 уколико је резултат задње извршене инструкције 0 и вредност 0 уколико резултат задње извршене инструкције није 0. Уколико сигнал **Z** има вредност 0, услов за скок није испуњен. Тиме је завршена фаза извршавање операција и прелази се на корак 22 и фазу опслуживање прекида (слика 5.г). Уколико сигнал **Z** има вредност 1, услов за скок је испуњен, па се сабирају садржај регистра $PC_{15...0}$, који представља текућу вредност регистра програмског бројача $PC_{15...0}$, и садржај разреда $IR_{15...8}$ проширен знаком на 16 бита, који представља померај, и добијена вредност, која представља адресу скока, уписује у регистар $PC_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 22 и фазу опслуживање прекида (слика 5.г).

Уколико вредност 1 има сигнал логичког услова **JMP**, безусловни скок се реализује. Садржај разреда $IR_{7...0}.IR_{15...8}$, који представља адресу скока, уписује се у регистар $PC_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 22 и фазу опслуживање прекида (слика 5.г).

Уколико вредност 1 има сигнал логичког услова **JADR** активан, безусловни скок на срачунату адресу се реализује, под условом да је задато неко од меморијских адресирања. У оквиру тога се садржај разреда $MAR_{15...0}$, који представља срачунату адресу скока задату неким од меморијских адресирања, уписује у регистар $PC_{15...0}$. Непосредно адресирање и регистарско директно адресирање нису дозвољени јер не дају адресу меморијске локације и зато се узима да се у случају да се у инструкцији JADR појави непосредно адресирање или регистарско директно адресирање, фаза извршавања ове операције завршава, чиме се ова инструкција претвара у инструкцију без дејства. С тога се проверава да ли вредност 1 имају сигнал **immed** или сигнал **regdir**. Уколико да, специфицирано је непосредно адресирање или регистарско директно адресирање, па се фаза извршавања операције завршава и прелази на корак 22 и фазу опслуживање прекида (слика 5.г). Уколико не, специфицирано је неко од меморијских адресирања, па се садржај разреда $MAR_{15...0}$ уписује у регистар $PC_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 22 и фазу опслуживање прекида (слика 5.г).

Уколико вредност 1 има сигнал логичког услова **JSR**, скок на потпрограм се реализује. У оквиру тога се садржај регистра $PC_{15...0}$ ставља на стек. Најпре се декрементира садржај регистра $SP_{15...0}$. Затим се пребацују садржај регистра $SP_{15...0}$ у регистар $MAR_{15...0}$ и старији бајт регистра $PC_{15...8}$ у регистар $MBR_{7...0}$. Затим се садржај регистра $MBR_{7...0}$ уписује у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$. После тога се поново декрементира садржај регистра $SP_{15...0}$. Затим се пребацују садржај регистра $SP_{15...0}$ у регистар $MAR_{15...0}$ и млађи бајт регистра $PC_{7...0}$ у регистар $MBR_{7...0}$. На крају се садржај регистра $MBR_{7...0}$ уписује у меморијску локацију на адреси која се налази у регистру $MAR_{15...0}$. Треба уочити да стек расте према нижим локацијама и да стек поинтер указује на задњу заузету локацију. С тога се садржај регистра $SP_{15...0}$ прво декрементира и после тога пребацује у регистар $MAR_{15...0}$. Такође треба уочити да се на стек прво ставља старији а потом млађи бајт регистра $PC_{15...0}$. То је последица начина смештања дво бајтовских величина у меморији, који предвиђа да старији бајт буде на вишој, а млађи бајт на нижој адреси. Затим се садржај разреда $IR_{7...0}$.$IR_{15...8}$, који представља адресу потпрограма, уписује у регистар $PC_{15...0}$. Тиме је завршена фаза извршавање операција и прелази се на корак 22 и фазу опслуживање прекида (слика 5.г).

Уколико вредност 1 има сигнал логичког услова **RTI**, повратак из прекидне рутине се реализује. У оквиру тога се садржајима са стека рестаурирају садржаји регистара $PSW_{7...0}$ и $PC_{15...0}$. Најпре се врши пребацивање садржаја регистра $SP_{15...0}$ у регистар $MAR_{15...0}$ и инкрементирање садржаја регистра $SP_{15...0}$. Затим се из меморијске локације са адресе која се налази у регистру $MAR_{15...0}$ чита садржај и уписује у регистар $MBR_{7...0}$. На крају се садржај регистра $MBR_{7...0}$ уписује у регистар $PSW_{7...0}$. На исти начин се са стека читају још два бајта и уписују најпре у млађи а потом у старији бајт регистра $PC_{15...0}$. Треба уочити да стек расте према нижим локацијама и да стек поинтер указује на задњу заузету локацију. С тога се приликом читања садржаја са стека, садржај регистра $SP_{15...0}$ прво пребацује у регистар $MAR_{15...0}$ и после тога инкрементира. Такође треба уочити да се са стека прво скида млађи а потом старији бајт регистра $PC_{15...0}$. То је последица начина смештања дво бајтовских величина у меморији, који предвиђа да старији бајт буде на вишој, а млађи бајт на нижој адреси. Тиме је завршена фаза извршавање операција и прелази се на корак 22 и фазу опслуживање прекида (слика 5.г).

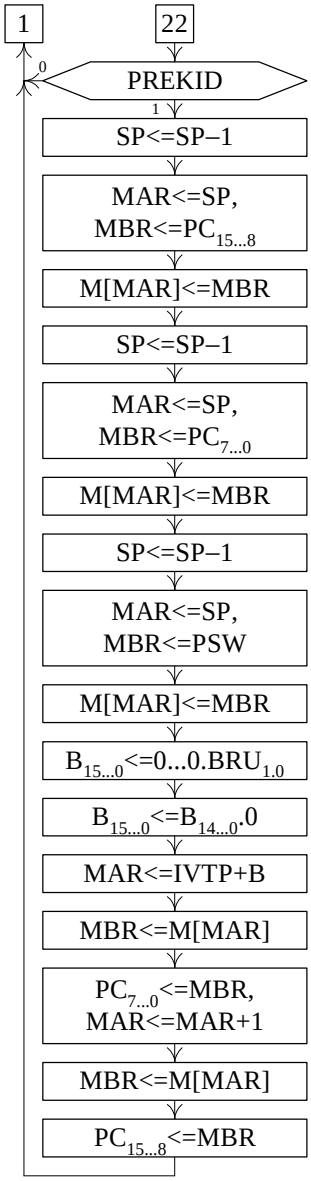
Уколико вредност 1 има сигнал логичког услова **RTS**, повратак из потпрограма се реализује. У оквиру тога се садржајем са стека рестаурира садржај регистра $PC_{15...0}$. Ово се реализује на идентичан као и у случају инструкције RTI. Тиме је завршена фаза извршавање операција и прелази се на корак 22 и фазу опслуживање прекида (слика 5.г).

Опслуживање прекида (слика 5.*г)*

Опслуживање прекида се реализује за све инструкције и то почев од корака 22. Уколико је сигнал логичког услова **PREKID** има вредност 0, у току извршавања претходних фаза није дошло до генерисања сигнала прекида, па се фаза опслуживање прекида завршава и прелази се на корак 1 и фазу читање инструкције (слика 5.а). Уколико сигнал **PREKID** има вредност 1, у току извршавања претходних фаза дошло је до генерисања сигнала прекида, па се прелази на кораке у оквиру којих се на стеку стављају садржаји регистара $PC_{15...0}$ и $PSW_{7...0}$ и потом утврђује адреса прекидне рутине и уписује у регистар $PC_{15...0}$. Стављање садржаја регистра $PC_{15...0}$ на стек се реализује на идентичан начин као и у случају инструкције JSR. На исти начин се на стек ставља и садржај регистра $PSW_{7...0}$. Адресе прекидних рутина се налазе у улазима табеле са

адресама прекидних рутина. Број улаза у табелу је дат садржајем регистра $BRU_{1.0}$, а почетна адреса табеле садржајем регистра $IVTP_{15...0}$. Најпре се садржај регистра $BRU_{1.0}$ проширен нулама до дужине 16 бита пребацује у регистар $B_{15...0}$, па се садржај регистра $B_{15...0}$ померањем улево за једно место множи са два. Тиме се број улаза претвара у померај. Потом се сабирањем садржаја регистара $IVTP_{15...0}$ и $B_{15...0}$ и смештањем њихове суме у регистар $MAR_{15...0}$ добија адреса на којој се налази адреса прекидне рутине. Са те и следеће адресе из меморије се читају млађи и старији бајт адресе прекидне рутине и уписују у млађих и старијих осам разреда регистра $PC_{15...0}$. Фаза опслуживање прекида је тиме завршена и прелази се на корак 1 и фазу читање инструкције (слика 5.а).

Треба уочити да се јављају две ситуације везане за вредност регистра $PC_{15...0}$ по завршетку фазе опслуживање прекида и преласка на корак 1 и фазу читање инструкције (слика 5.а). Уколико је сигнал **PREKID** имао вредност 0, у регистру $PC_{15...0}$ је адреса прве следеће инструкције после инструкције која је извршена. Уколико је сигнал **PREKID** имао вредност 1, у регистру $PC_{15...0}$ је адреса прве инструкције прекидне рутине.



Слика 5.г Дијаграм тока – фаза опслуживање прекида