

Архитектура рачунара



Садржај

- Скуп инструкција
- Формати инструкција
- Програмски доступни регистри
- Начини адресирања
- Типови података

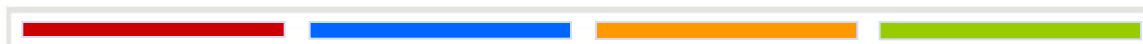


Дијаграм тока извршавања инструкције



Скуп инструкција

- Reduced Instruction Set Computers (RISCs)
 - Једноставне инструкције
 - Флексибилне
 - Већи проток
 - Брже извршавање
- Complex Instruction Set Computers (CISCs)
 - Хардверска подршка за програмске језике вишег нивоа
 - Компактни програми



RISC

- Само Load/Store инструкције приступају директно меморији
- Рад са подацима на нивоу трансфера регистар-регистар
- Једноставни начини адресирања
- Сви формати инструкција су исте дужине
- Инструкције раде једноставне операције
- Обично једна инструкција по циклусу (једноставна инструкција)



CISC

- Већина инструкција има приступ меморији
- Велики број начина адресирања
- Инструкције имају различите дужине
- Инструкције обављају и једноставне и комплексне операције
- Више циклуса по инструкцији (комплексне инструкције)



Скуп инструкција

- Скуп стандардних инструкција чине:
 - аритметичке инструкције,
 - логичке инструкције,
 - инструкције померања и ротирања,
 - инструкције скока,
 - инструкције преноса,
 - мешовите инструкције



Аритметичке инструкције

- Аритметичким инструкцијама се реализују стандардне аритметичке операције попут сабирања, одузимања, множења, дељења, инкрементирања, декрементира, и поређења.
- Неки мнемоници би били:
 - ADD
 - SUB
 - MUL
 - DIV
 - INC
 - DEC
 - CMP

Аритметичке инструкције - ADD

- Троадресним форматом инструкција:
- $ADD\ a,\ b,\ c$
- ADD је симболички означено поље кода операције, а са a , b и c спецификације два изворишна (b и c) и једног одредишног операнда (a).
- b и c могу да буду било која комбинација регистара процесора, меморијских локација и непосредних величина у инструкцији.
- a не може да буде непосредна величина.



Аритметичке инструкције - ADD

- Најчешће комбинације:
 - a , b и c су регистри процесора,
 - a је регистар процесора, b је регистар процесора и c је непосредна величина,
 - a , b и c су меморијске локације,
 - итд.

Аритметичке инструкције - ADD

- Двоадресним форматом инструкција:
- `ADD a, b`
- ADD је симболички означено поље кода операције, а са `a, b` спецификације два изворишна операнда од којих је један и одредишни операнд (`a` у примеру).
- `b` може да буду регистар процесора, меморијска локација и непосредна величина.
- `a` не може да буде непосредна величина.



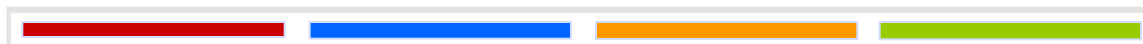
Аритметичке инструкције - ADD

- Најчешће комбинације:
 - a и b су регистри процесора,
 - a је регистар процесора, b је непосредна величина,
 - a и b су меморијске локације (ретко)
 - итд.



Аритметичке инструкције - ADD

- Једноадресним форматом инструкција:
- ADD *a*
- ADD је симболички означено поље кода операције, а са *a* спецификација другог изворишног операнда, при чему је акумулатор имплицитно извориште првог операнда и имплицитно одредиште резултат.



Аритметичке инструкције - ADD

- Најчешће комбинације:
 - a је регистар процесора
 - a је меморијска локација
 - a је непосредна величина
(код неких инструкција није дозвољено)



Аритметичке инструкције - ADD

- Нулаадресним форматом инструкција:
- ADD
- ADD је симболички означено поље кода операције, при чему је врх стека имплицитно извориште за оба изворишна операнда и имплицитно одредиште.
- Током извршавања ове инструкције са врха стека се имплицитно чита најпре први а потом и други изворишни операнд, затим се над њима реализује операција сабирања и на крају се добијени резултат смешта на врх стека као имплицитно одредиште.



Аритметичке инструкције - ADD

- Слична је ситуација и са форматима инструкција за операције одузимања, множења и дељења.

Аритметичке инструкције - INC

- Посебан случај операције сабирања код које је имплицитно одређено да је вредност која се сабира једнака 1.
- Двоадресни формат:
- INC a, b
- INC је симболички означено поље кода операције, а са a и b спецификације изворишног (b) и одредишног операнда (a).

*Нема троадресни формат. Односно поље намењено за други изворишни операнд се не користи.

Аритметичке инструкције - INC

- Најчешће комбинације:
 - a и b су регистри процесора,
 - a је регистар процесора, b је непосредна величина,
 - a и b су меморијске локације,
 - итд.

Аритметичке инструкције - INC

- Једноадресним форматом инструкција:
 - INC *a*
 - INC је симболички означено поље кода операције, а са *a* спецификација изворишног и одредишног операнда.
- * Некада је акумулатор имплицитно одредиште.



Аритметичке инструкције - INC

- Најчешће комбинације:
 - *a* је меморијска локација
 - *a* је регистар процесора,

Аритметичке инструкције - INC

- Нулаадресним форматом инструкција:
- INC
- INC је симболички означено поље кода операције, при чему је врх стека имплицитно извориште операнда и имплицитно одредиште.
- Током извршавања ове инструкције са врха стека се имплицитно чита најпре операнд, затим се над њим реализује операција увећавања за један и на крају се добијени резултат смешта на врх стека као имплицитно одредиште.



Аритметичке инструкције - INC

- **Безадресним** форматом инструкција:
- INC
- INC је симболички означено поље кода операције, при чему је акумулатор имплицитно извориште и имплицитно одредиште.
- Понекада се означава са INCA

Нулаадресни <> Безадресни



Аритметичке инструкције - INC

- Нулаадресни формат:
- INC; // скине са стека у акумулатор
// увећа акумулатор за један
// сними акумулатор на стек
- Безадресни формат:
- INC; // увећа акумулатор за један

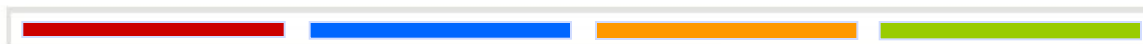


Аритметичке инструкције - CMP

- Посебан случај операције одузимања код које се резултат одузимања никуда не уписује.
- Постављање индикатора N, Z, C и V у програмској статусној речи PSW процесора.
- Ова инструкција се назива аритметичко упоређивање.
- Формати инструкције упоређивања за процесоре са двоадресним, једноадресним и нулаадресним форматима инструкција су:
 - CMP *a, b*
 - CMP *a*
 - CMP

Аритметичке инструкције - СМР

- Индикатор N се поставља на вредност 1 уколико је резултат извршене операције одузимања негативан, док се у супротном случају поставља на вредност 0. Усвојено правило је да се индикатор N поставља на вредност најстаријег бита резултата операције одузимања.
- Индикатор Z се поставља на вредност 1 уколико је резултат извршене операције одузимања нула, док се у супротном случају поставља на вредност 0.
- Ови индикатори се постављају за све типове података.



Аритметичке инструкције - СМР

- Индикатор C се поставља на 1 уколико је први операнд мањи од другог операнда и представља позајмицу, док се у супротном случају поставља на 0.
- Индикатор V се поставља на вредност 1 уколико је дошло до прекорачења (изласка из) опсега приликом операције одузимања.



Логичке инструкције

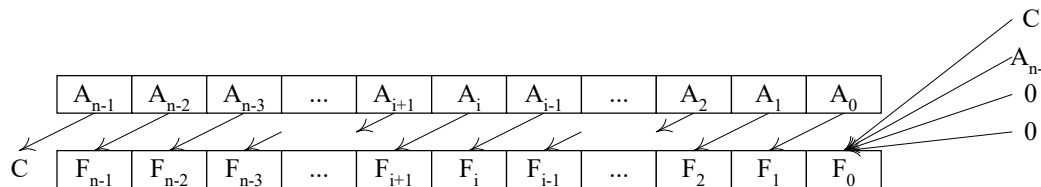
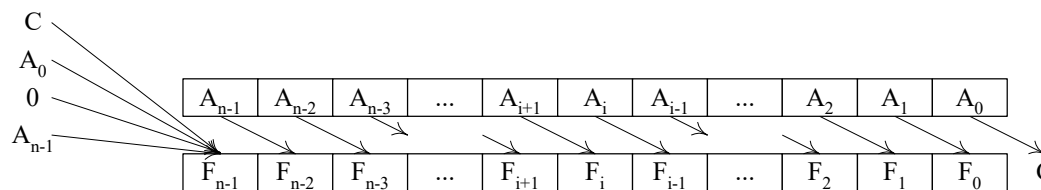
- Логичким инструкцијама се реализују стандардне логичке операције И, ИЛИ, ексклузивно ИЛИ и комплементирања.
- Мнемоници би били:
 - AND
 - OR
 - XOR
 - NOT
- Могу се јавити у свим форматима, осим инструкције NOT која нема троадресни формат.

Логичке инструкције - TST

- Посебан случај операције И код које се резултат никуда не уписује.
- Постављање индикатора N и Z у регистру PSW.
- Ова инструкција се назива логичко упоређивање.
- Формати инструкције упоређивања за процесоре са двоадресним, једноадресним и нулаадресним форматима инструкција су:
 - TST *a, b*
 - TST *a*
 - TST

Инструкције померања

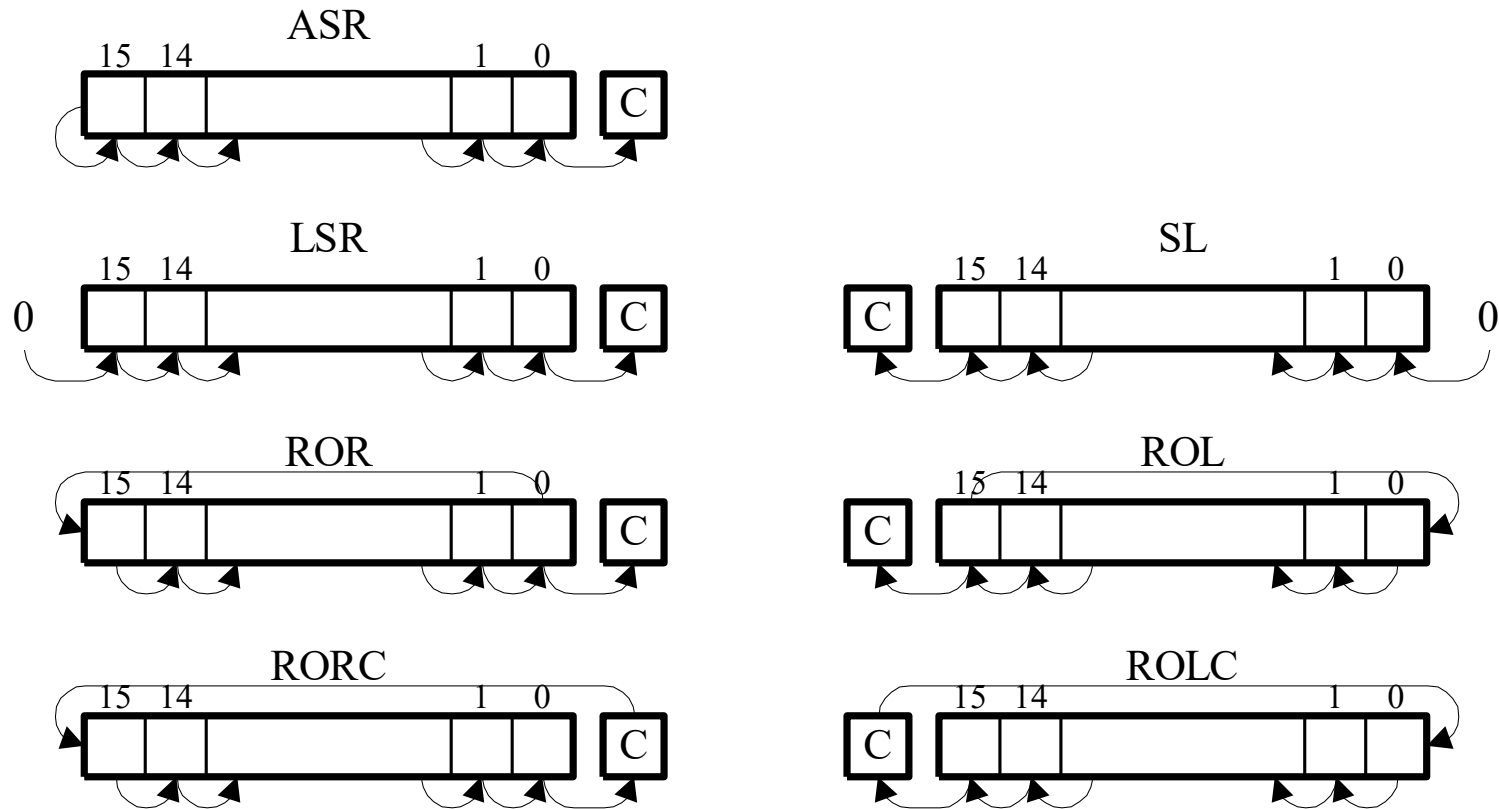
- Инструкцијама померања се реализују операције аритметичког и логичког померања удесно и улево, и ротација удесно и улево.
- Мнемоници би били:
 - ASR
 - ASL
 - LSR
 - LSL
 - ROR
 - ROL
 - (RORC)
 - (ROLC)



Инструкције померања - ASR

- Формати инструкције аритметичког померања удесно за процесоре са двоадресним, једноадресним и нулаадресним форматима инструкција су:
 - ASR a, b
 - ASR a
 - ASR
- Са ASR је симболички означено поље кода операције, а са a и b спецификације одредишних и изворишних операнада.

Померање за једно место



Инструкције преноса - MOV

- Двоадресни формат:
- `MOV a, b`
- MOV је симболички означено поље кода операције, а са *a* и *b* спецификације одредишног и изворишног операнда, респективно.

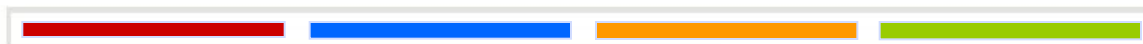
*Нема троадресни формат. Односно поље намењено за други изворишни операнд се не користи.

Инструкције преноса - MOV

- Најчешће комбинације:
 - a и b су меморијске локације
 - a и b су регистри процесора,
 - a је регистар процесора, b је непосредна величина,
 - итд.

Инструкције преноса - LOAD/STORE

- Једноадресни формат:
- LOAD *a*
- STORE *b*
- Са LOAD и STORE је симболички означено поље кода операције, а са *a* и *b* спецификације изворишног и одредишног операнда, респективно.
- Инструкцијом LOAD се операнд са изворишта *a* преноси у акумулатор као имплицитно одредиште,
- Инструкцијом STORE садржај акумулатора као имплицитно извориште преноси у одредиште *b*.



Инструкције преноса - LOAD/STORE

- Најчешће комбинације:
 - b није непосредна величина.

Инструкције преноса - IN/OUT

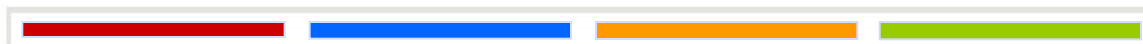
- Двоадресни формат:
- IN *regper, regproc*
- OUT *regproc, regper*
- Са IN је симболички означено поље кода операције за пренос из регистра контролера периферије у регистар процесора, при чему је са *regper* директно дата адреса регистра контролера периферије, док је са *regproc* директно дата адреса регистра процесора, са OUT је симболички означено поље кода операције за пренос из регистра процесора у регистар контролера периферије.

Инструкције преноса - IN/OUT

- Једноадресни формат:
- IN *regper*
- OUT *regper*
- Са IN је симболички означено поље кода операције за пренос из регистра контролера периферије у акумулатор, при чему је са *regper* директно дата адреса регистра контролера периферије, са OUT је симболички означено поље кода операције за пренос из акумулатора у регистар контролера периферије.

Инструкције преноса - PUSH/POP

- Нулаадресни формат:
- PUSH a
- POP b
- Са PUSH и POP је симболички означено поље кода операције, а са a и b спецификације изворишног и одредишног операнда, респективно.
- Инструкцијом PUSH се операнд са изворишта a преноси на стек као имплицитно одредиште,
- Инструкцијом POP садржај са стека као имплицитно извориште преноси у одредиште b .



Инструкције преноса - PUSH/POP

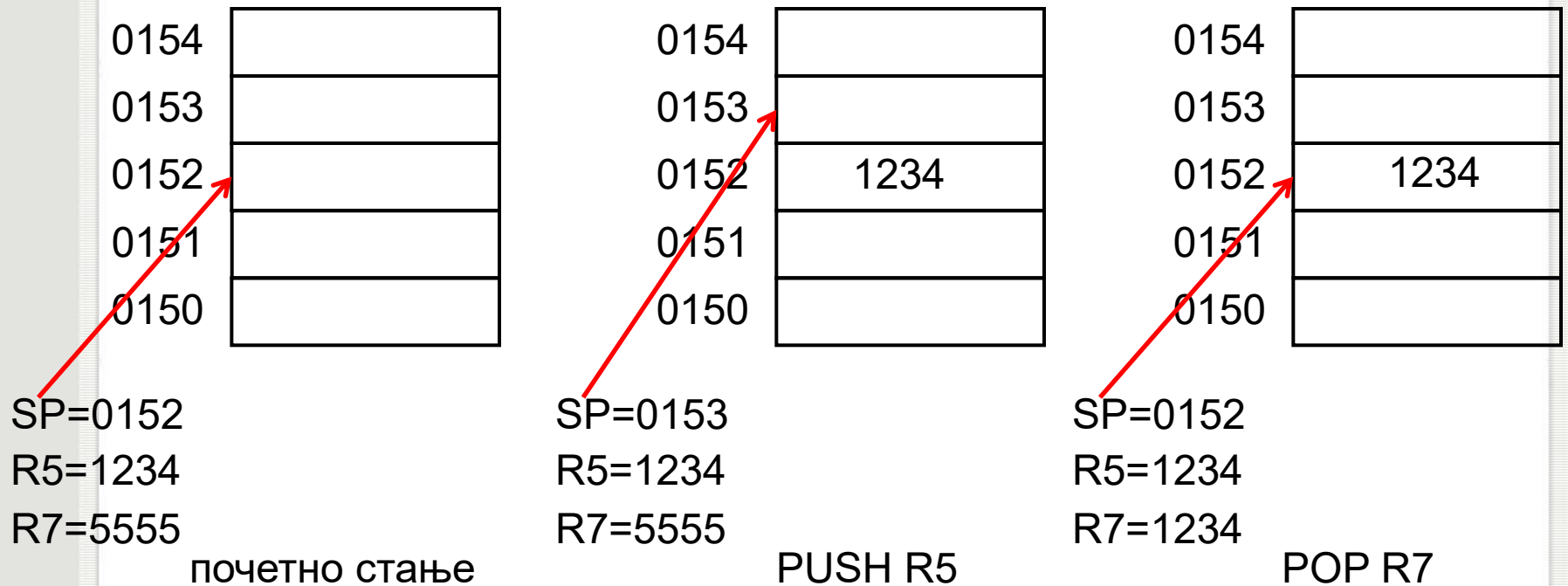
- Најчешће комбинације:
 - b није непосредна величина.

Варијанте раста стека

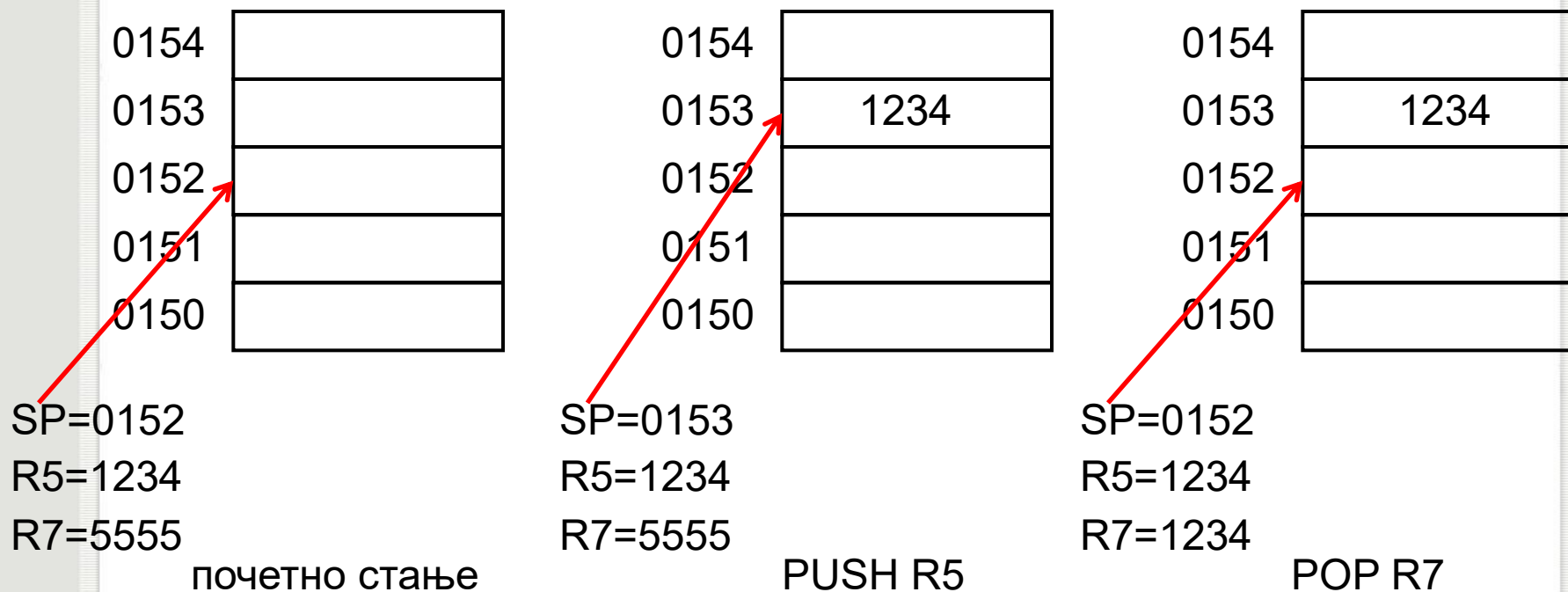
- Стек може да расте (да се попуњава) ка:
 - Нижим меморијским локацијама
 - Вишим меморијским локацијама
- Показивач на врх стека може да указује на:
 - Прву слободну локацију
 - Последњу попуњену локацију



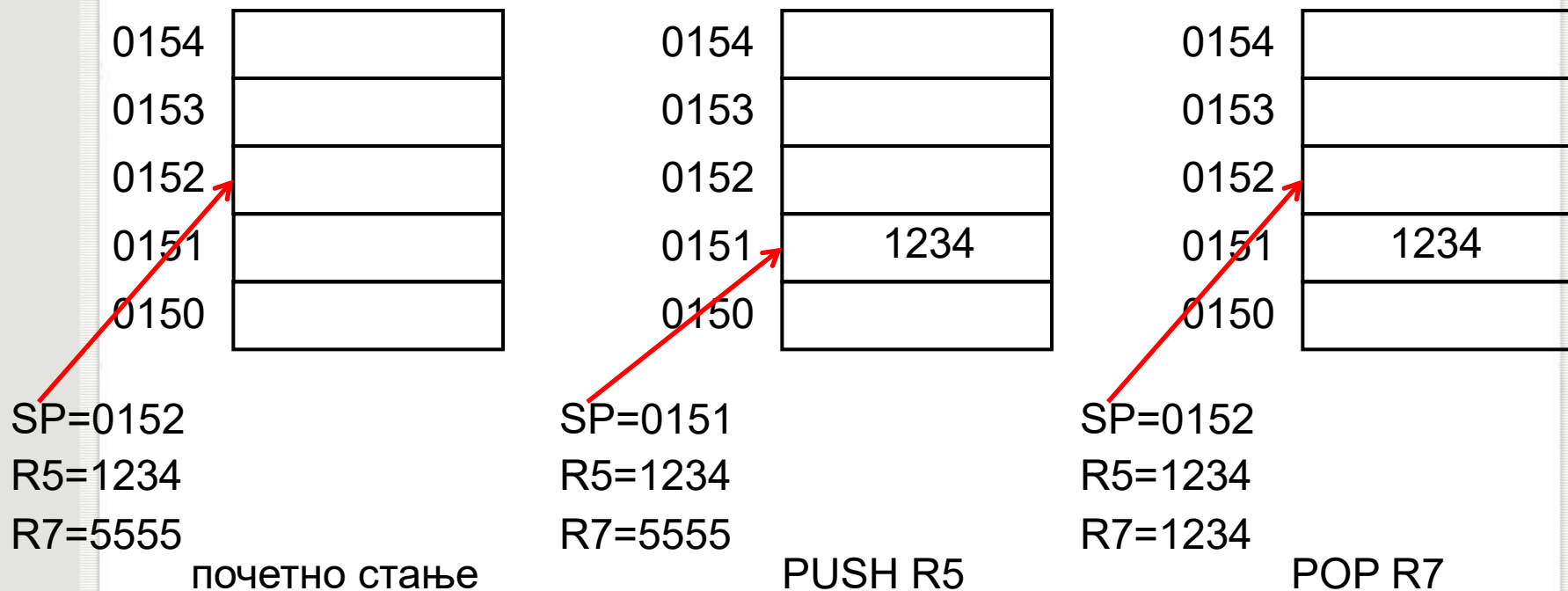
Варијанта: навише, на слободну



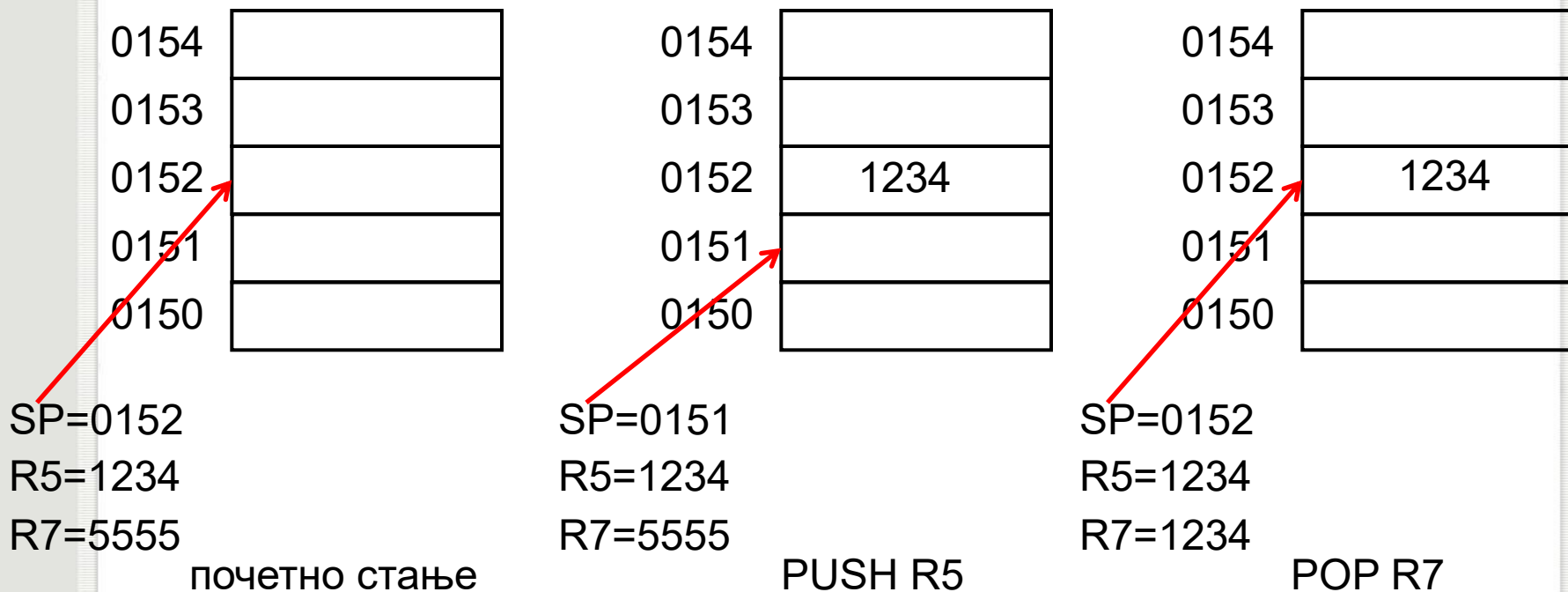
Варијанта: навише, на заузету



Варијанта: наниже, на заузету



Варијанта: наниже, на слободну



Инструкције скока

- Инструкције скока се сврставају у следеће групе:
 - инструкције безусловног скока,
 - инструкције условног скока,
 - инструкције скока на потпрограм,
 - повратка из потпрограма,
 - инструкција скока у прекидну рутину,
 - инструкција повратка из прекидне рутине.



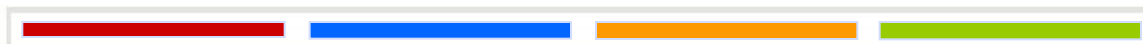
Инструкција безусловног скока

- Формат инструкције безусловног скока је
- `JMP adresa`
- при чему је са `JMP` симболички означено поље кода операције, а са *adresa* поље са вредношћу адресе меморијске локације на којој се налази инструкција на чије извршавање треба прећи. Извршавања инструкције `JMP` се састоји у уписивању вредност из поља *adresa* у програмски бројач РС.



Инструкција безусловног скока

- У случају када адреса инструкције на коју треба скочити није позната у време превођења, већ се израчунава током извршавања програма, користи се инструкција безусловног скока чији је формат
- `JMPIND a`
- при чему је са `JMPIND` симболички означено поље кода операције, а са `a` поље са спецификацијом адресе меморијске локације на којој се налази инструкција на чије извршавање треба прећи.



Инструкција условног скока

- Формат инструкције условног скока је
- `CODE disp`
- при чему је са `CODE` симболички означено поље кода једне од операције безусловног скока, а са *disp* поље са вредношћу помераја са којим треба направити релативан скок у односу на текућу вредност програмског бројача уколико је услов за скок испуњен.

Инструкције условног скока

инструкција	значење	услов
BEQL	скок на једнако	$Z = 1$
BNEQ	скок на неједнако	$Z = 0$
BGRTU	скок на веће него (без знака)	$C \vee Z = 0$
BGREU	скок на веће него или једнако (без знака)	$C = 0$
BLSSU	скок на мање него (без знака)	$C = 1$
BLEQU	скок на мање него или једнако (без знака)	$C \vee Z = 1$
BGRT	скок на веће него (са знаком)	$(N \oplus V) \vee Z = 0$
BGRE	скок на веће него или једнако (са знаком)	$N \oplus V = 0$
BLCC	скок на мање него (са знаком)	$(N \oplus V) = 1$
BLEQ	скок на мање него или једнако (са знаком)	$(N \oplus V) \vee Z = 1$
BNEG	скок на $N = 1$	$N = 1$
BNNG	скок на $N = 0$	$N = 0$
BOVF	скок на $V = 1$	$V = 1$
BNVF	скок на $V = 0$	$V = 0$

Инструкције условног скока - алтернатива

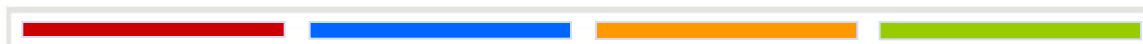
инструкција	значење	услов
BZ	скок на једнако	$Z = 1$
BNZ	скок на неједнако	$Z = 0$
BC	скок на мање него (без знака)	$C = 1$
BNC	скок на веће него или једнако (без знака)	$C = 0$
BN	скок на $N = 1$	$N = 1$
BNN	скок на $N = 0$	$N = 0$
BV	скок на $V = 1$	$V = 1$
BNV	скок на $V = 0$	$V = 0$

Инструкције скока на потпрограм

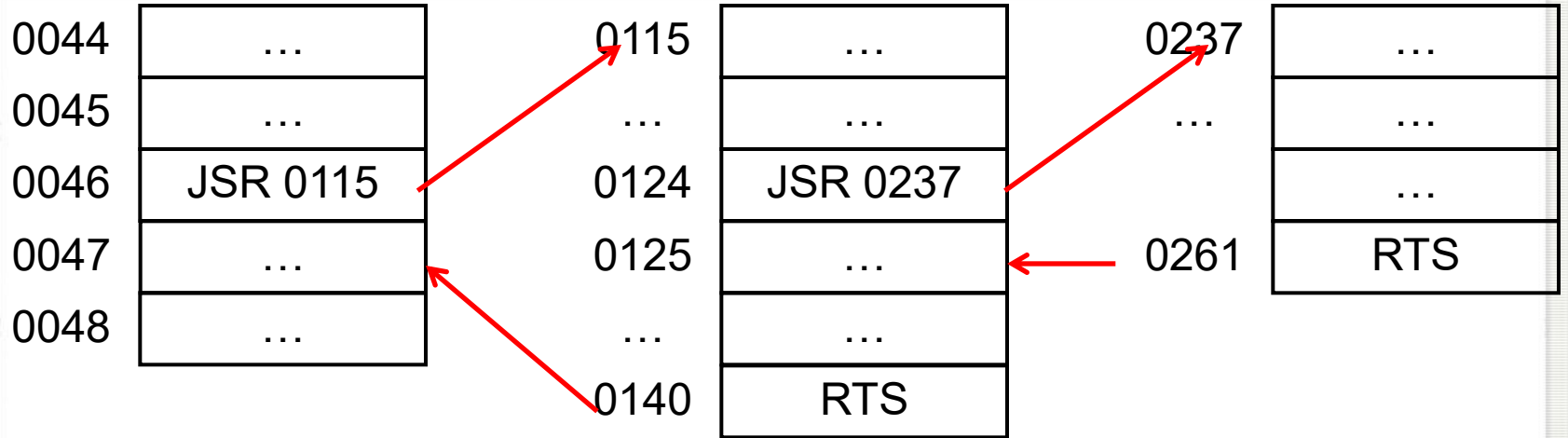
- Формат инструкције скока на потпрограм је
- JSR *adresa*
- при чему је са JSR симболички означено поље кода операције, а са *adresa* поље са вредношћу адресе меморијске локације на којој се налази прва инструкција потпрограма на чије извршавање треба прећи.
- Извршавање инструкције JSR се састоји у стављању на стек текуће вредности програмског бројача PC и уписивању вредности из поља *adresa* у програмски бројач PC.

Инструкције повратка из потпрограма

- Формат инструкције повратка из потпрограма је
- RTS
- при чему је са RTS симболички означено поље кода операције. Инструкција RTS мора да буде задња инструкција потпрограма.
- Извршавање инструкције RTS се састоји у скидању вредности са стека и уписивању у програмски бројач PC.

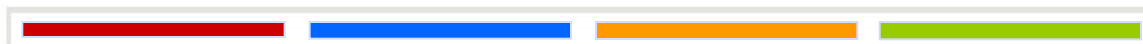


Скок на потпрограм



Инструкција скока на прекидну рутину

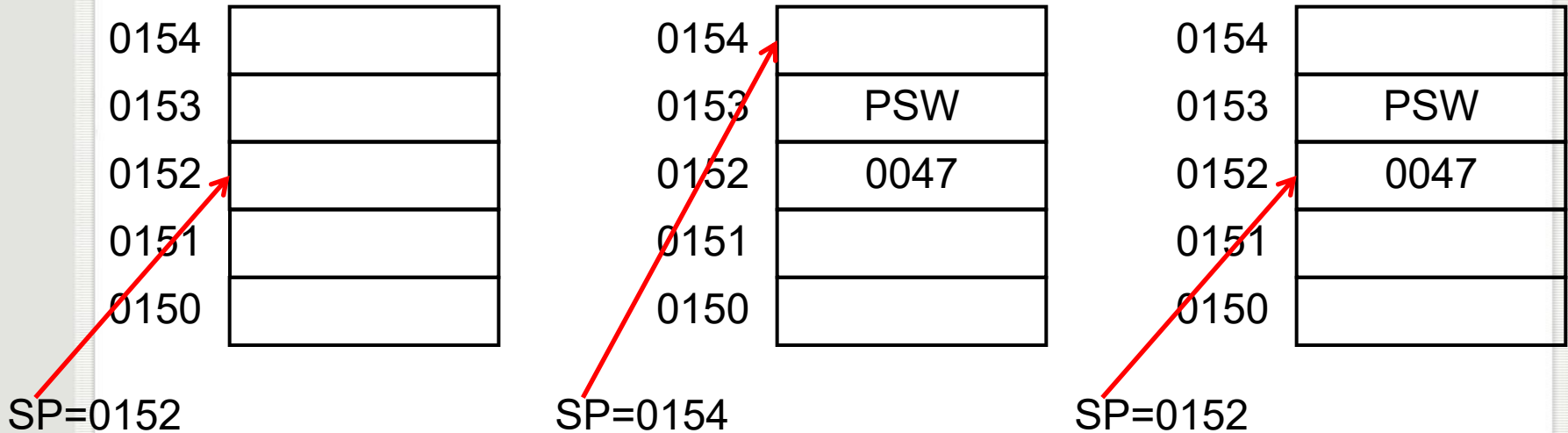
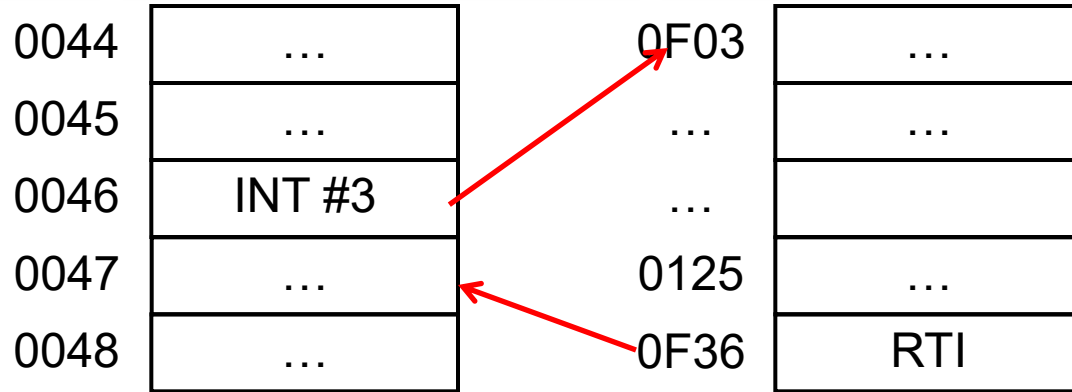
- Формат инструкције скока на прекидну рутину је
- *INT num*
- при чему је са *INT* симболички означено поље кода операције, а са *num* поље са вредношћу адресе меморијске локације на којој се налази прва инструкција прекидне рутине.
- Извршавање инструкције *INT* се састоји у стављању на стек текуће вредности програмског бројача *PC*, програмске статусне речи *PSW*, а у неким реализацијама процесора и свих програмски доступних регистара, и уписивању вредности из поља *adresa* у програмски бројач *PC*.



Инструкција повратка из прекидне рутине

- Формат инструкције повратка из прекидне рутине је
- RTI
- при чему је са RTI симболички означено поље кода операције. Инструкција RTI мора да буде задња инструкција прекидне рутине. Извршавање инструкције RTI се састоји у скидању вредности са стека и уписивању у програмски доступне регистре, уколико се ради о реализацијама процесора код којих се приликом скока на прекидну рутину ови регистри чувају на стеку, програмску статусну реч и програмски бројач PC.

Скок на прекид



Мешовите инструкции

- У скупу инструкција могу да се нађу и инструкции којима се омогућава задавање различитих режима рада рачунара, пружа подршка за тестирање програма итд.
- Пример:
 - INTE
 - INTD
 - TRPE
 - TRPD
 - NOP
 - итд.

Мешовите инструкции - INTE

- Формат инструкции задавања режима рада у коме се реагује на захтеве за прекид је
- INTE
- при чему је са INTE симболички означено поље кода операције. Извршавање инструкции INTE се састоји у уписивању вредности 1 у разред I регистра PSW.

Мешовите инструкции - INTD

- Формат инструкции задавања режима рада у коме се не реагује на захтеве за прекид је
- INTD
- при чему је са INTD симболички означено поље кода операције. Извршавање инструкции INTD се састоји у уписивању вредности 0 у разред I регистра PSW.

Мешовите инструкције - TRPE

- Формат инструкције задавања режима *прекид* после сваке инструкције је
- TRPE
- при чему је са TRPE симболички означено поље кода операције. Извршавање инструкције TRPE се састоји у уписивању вредности 1 у разред T регистра PSW.

Мешовите инструкције - TRPD

- Формат инструкције заустављање режима *прекид* после сваке инструкције је
- TRPD
- при чему је са TRPD симболички означено поље кода операције. Извршавање инструкције TRPD се састоји у уписивању вредности 0 у разред T регистра PSW.



Мешовите инструкции

- Формат инструкции без дејства је
- NOP
- при чему је са NOP симболички означено поље кода операције. Извршавање инструкции NOP се састоји само у читању инструкции и инкрементирању програмског бројача РС, после чега се прелази на извршавање следеће инструкции.

Вероватноћа појединих инструкција

Ранг	Инструкција	Процент
1	load	22%
2	conditional branch	20%
3	compare	16%
4	store	12%
5	add	8%
6	and	6%
7	sub	5%
8	move reg-reg	4%
9	call	1%
10	return	1%
Тотал		96%

8086 инструкције

Формати инструкција

- Форматом инструкције се специфицирају две врсте информација неопходне за извршавање инструкција програма и то:
 - операција и тип податка са којим операција треба да се реализује и
 - изворишни и одредишни операнди.



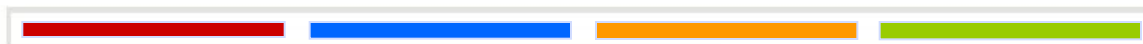
Наредна инструкција?

- По завршетку извршавања текуће инструкције одакле узети следећу инструкцију?
- Одређено је текућом вредношћу регистра РС.
- Код читања инструкције као адреса меморијске локације увек се користи текућа вредност регистра РС, која се том приликом инкрементира.
- Резултат => секвенцијално читати и извршавати.



Наредна инструкција?

- Постоји потреба за гранањем у програму.
- Не треба прећи на прву следећу инструкцију у секвенци, већ на неку инструкцију ван секвенце.
- То се постиже инструкцијама скокова, које треба да се убаце у делове програма када треба одступити од секвенцијалног извршавања програма.
- Ефекат инструкција скокова је уписивање нове вредности у програмски бројач РС.



Изворишни и одредишни операнди

- Овим пољем се експлицитно специфицирају операнди. Постоје више варијанти овог поља које настају као последица следећа два елемента:
 - број експлицитно специфицираних операнада и
 - могуће локације операнада.

Број експлицитно специфицираних операнада

- Адресни формати инструкција
 - Троадресни – три експлицитна операнда
 - Двоадресни – два експлицитна операнда
 - Једноадресни – један експлицитан операнд
 - Нулаадресни – нема експлицитних операнада (?)

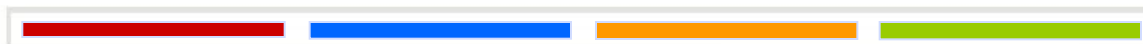


Троадресни формат инструкције

- Експлицитно су пољима А1, А2 и А3 дефинисана сва три операнда (локације?)



- **Напомена:** Код неких процесора поља А1 и А2 дефинишу изворишне операнде, а А3 одредишни операнд. Међутим, могу и А2 и А3 бити изворишни, а А1 одредишни операнд.



Троадресни формат инструкције

- Треба срачунати израз
- $A = B + C$;
- Асемблерски код:
- `ADD A, B, C`
- А, В и С представљају симболичке ознаке појединих операнда. Детаљу и секције која се односи на начине адресирања.

Двоадресни формат инструкције

- Експлицитно су пољима А1 и А2 дефинисана два изворишна операнда.
- Одредиште се не дефинише експлицитно већ се користи или А1 или А2.



Двоадресни формат инструкције

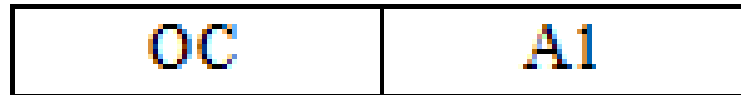
- Треба срачунати израз
- $C = A + B;$
- Асемблерски код:
- `MOV A, C`
- `ADD C, B`

Двоадресни формат инструкције

- Добра страна двоадресног формата у односу на троадресни формат је да је инструкција краћа.
- Лоша страна формата је потреба за већим бројем инструкција.

Једноадресни формат инструкције

- Експлицитно се пољем А1 дефинише један изворишни операнд.



- Код ових процесора постоји посебан регистар процесора који се обично назива акумулатор који представља једно имплицитно извориште и имплицитно одредиште.

Једноадресни формат инструкције

- Треба срачунати израз
- $C = A + B;$
- Асемблерски код:
- LOAD A
- ADD B
- STORE C



Једноадресни формат инструкције

- Добра страна једноадресног формата у односу на двоадресни формат је да је инструкција краћа.
- Лоша страна је да је потребно још више инструкција да се реализује срачунавање истог израза.

Нулаадресни формат инструкције

- Постоји само поље кода операције и не постоји ни једно поље којим би се експлицитно дефинисале изворишне и одредишне операнде.
- Врх стека је имплицитно извориште за оба операнда и имплицитно одредиште за резултат.

ОС

*Изузетак су две инструкције са једним операндом:

- PUSH којом се операнд смешта на врх стека
- POP којом се операнд са врха стека смешта на одредиште

ОС

А1

Нулаадресни формат инструкциије

- Треба срачунати израз
- $C = A + B;$
- Асемблерски код:
- PUSH A
- PUSH B
- ADD
- POP C



Нулаадресни формат инструкције

- Добра страна нулаадресног формата у односу на једноадресног формат је да је инструкција краћа.
- Лоша страна је да је потребно још више инструкција да се реализује срачунавање истог израза.
- (Сви коментари су исти?)



Број операнда

- Уколико је процесор са неким форматом инструкција то не значи да све инструкције користе све операнде.
- Код неких инструкција или начина адресирања могуће је да су неки операнди вишак и та поља се онда не користе за ту намену.



Могуће локације операнада

- Могуће локације операнада су
 - меморијске локације,
 - регистри процесора,
 - непосредне величине у инструкцији.
- Више варијанти реализације овог поља, а мотиви код његовог дефинисања су да:
 - овај део буде краћи да би се мање меморије заузимало за програме и брже читавале инструкције (адреса меморије, непосредна величина или адреса регистра),
 - се брже долази до операнада (меморије, непосредна величина или регистар),
 - се пружи подршка за могућу примену неких техника реализације процесора (pipeline)

Врсте архитектура

- На основу могућих локација операнада разликују се три врсте архитектура:
 - меморија – меморија, (тро, дво, једно, и нулаадресни)
 - меморија – регистар, (двоадресни)
 - регистар – регистар. (троадресни)

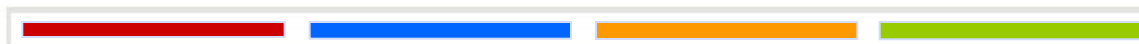
Програмски бројач РС

- Регистар чија се вредност имплицитно користи као адреса меморијске локације са које се чита инструкција.
- Вредност програмског бројача РС се мења:
 - имплицитно приликом сваког читања инструкције када се врши инкрементирање програмског бројача РС
 - експлицитно инструкцијама скока када се у програмски бројач РС уписује нова вредност.



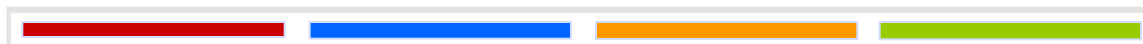
Регистри података DR

- Регистри података се уводе да би се убрзао приступ подацима, тако што би се током извршавања програма приступало подацима у регистрима података процесора уместо у меморијским локацијама, а резултат је чињенице да је приступ регистрима скоро за ред величине бржи од приступа меморијским локацијама.
- Регистрима података се приступа програмским путем инструкцијама преноса и аритметичким, логичким и померачким инструкцијама.



Адресни регистри AR

- Садржај адресног регистра користи се:
 - као адреса меморијске локације са које се чита или у коју се уписује податак;
 - инструкцијом скока JMPIND када се садржај адресног регистра користи као адреса скока коју треба уписати у програмски бројач РС.
- Адресни регистри се уводе да би се убрзао приступ адресама, тако што би се током извршавања програма приступало адресама у адресним регистрима процесора уместо у меморијским локацијама, а резултат је чињенице да је приступ регистрима скоро за ред величине бржи од приступа меморијским локацијама.



Базни регистри BR

- Збир садржаја специфицираног базног регистра и помераја код базног адресирања, односно базног регистра, индексног регистра и помераја код базно-индексног адресирања, представља адресу меморијске локације на којој се налази изворишни или одредишни операнд.
- Користи се код базног и базно-индексног адресирања

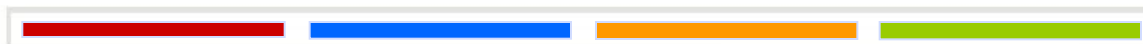
Индексни регистри XR

- Збир садржаја специфицираног индексног регистра и помераја код индексног адресирања, односно базног регистра, индексног регистра и помераја код базно-индексног адресирања, представља адресу меморијске локације на којој се налази изворишни или одредишни операнд.
- Користе се код индексног и базно-индексног адресирања.



Регистри опште намене GPR

- Користе се на исти начин као регистри података, адресни регистри, базни регистри и индексни регистри.
- Регистри опште намене GPR се јављају код оних процесора код којих не постоје посебни регистри података, адресни регистри, базни регистри и индексни регистри.
- Могу да користе било који од регистара уз било које адресирање.



Регистар PSW

- Стандардна програмска статусна реч процесора састављена је од одређеног броја битова, који се обично називају индикатори.
- Битови програмске статусне речи PSW се независно постављају и користе по правилима дефинисаним посебно за сваки бит.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I	-	-	-	-	-	-	-	-	-	-	-	V	C	Z	N

Пример: Структура регистра PSW

Битови регистра PSW

- N—бит који се поставља на 1 у случају да је резултат операције негативан,
- Z—бит који се поставља на 1 у случају да је резултат операције једнак 0,
- C—бит који се поставља на 1 у случају преноса/позајмице у аритметици целобројних величина без знака *
- V—бит који се поставља на 1 у случају прекорачења у аритметици целобројних величина са знаком *
- I—бит који је једнак 1 ако треба да буду дозвољени маскирајући прекиди.

Битови регистра PSW

*Ради једноставније и униформније реализације код неких процесора бити C и V се код аритметичких инструкција **увек** постављају и то на основу излаза ALU јединице.

- Бит C на највиши бит преноса C_n .
 $C = C_n$
- Бит V на разлику највиших бита преноса C_n и C_{n-1} .
 $V = C_n \text{ XOR } C_{n-1}$

Регистар SP

- Указивач на врх стека када је стек организован у оперативној меморији.
- Подршка за реализацију LIFO структуре података.
- Садржај регистра SP се користи као адреса меморијске локације у коју треба уписати податак или из које треба очитати податак.
- Ажурирање и то инкрементирање или декрементирање садржаја регистра SP.



Начини адресирања

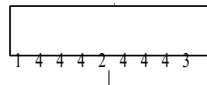
- Начини адресирања одређују да ли је операнд садржај неке меморијске локације, неког од регистара података или регистара опште намене процесора или непосредна величина у самој инструкцији.
- Начини адресирања специфицирају и како треба формирати адресу меморијске локације уколико је операнд садржај неке меморијске локације.

Меморијско директно адресирање

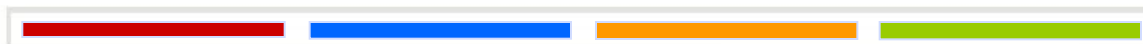
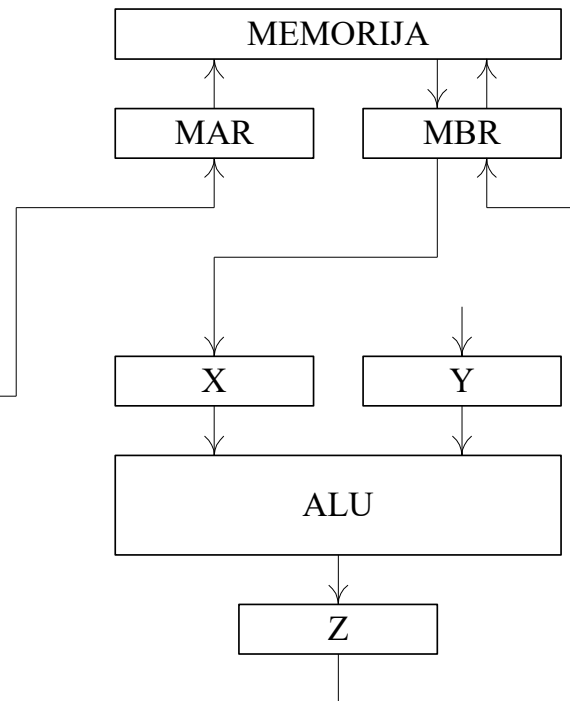
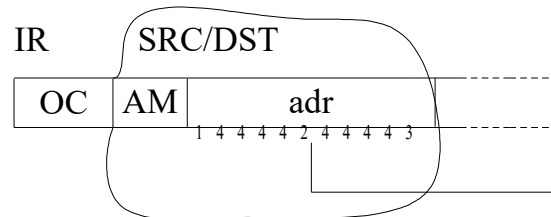
- Меморијско директно адресирање је адресирање код кога се операнд налази у меморијској локацији, а у адресном делу инструкције у коме се специфицира операнд, поред групе битова којима се задаје начин адресирања (AM), постоји и група битова којима се задаје адреса меморијске локације (adr).

Меморијско директно адресирање

PC



IR

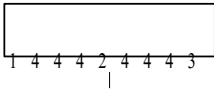


Меморијско индиректно адресирање

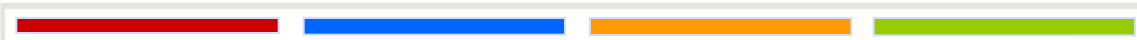
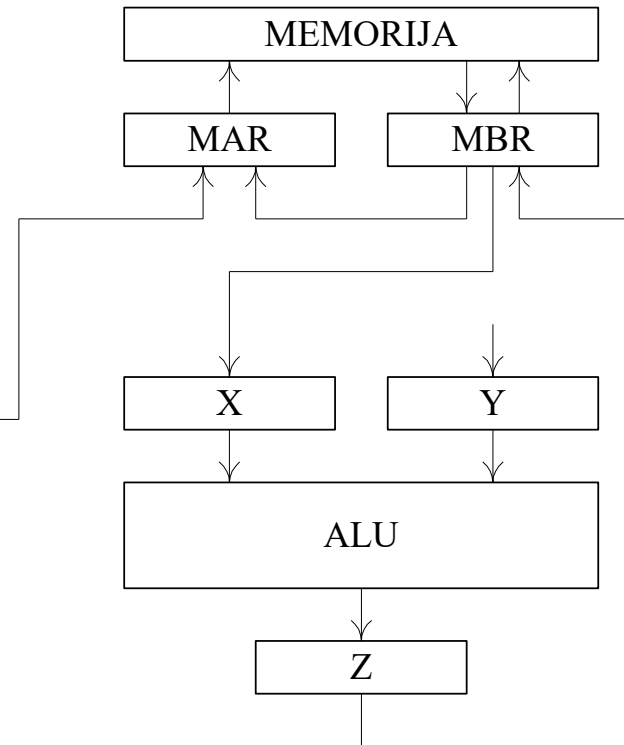
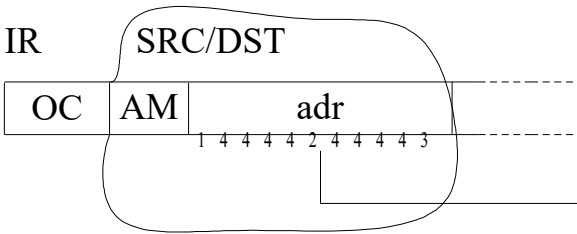
- Меморијско индиректно адресирање је адресирање код кога се операнд налази у меморији, а у адресном делу инструкције у коме се специфицира операнд, поред групе битова којима се задаје начин адресирања (АМ), постоји и група битова којима се задаје адреса меморијске локације на којој се налази адреса операнда.

Меморијско индиректно адресирање

PC



IR

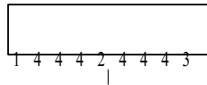


Непосредно адресирање

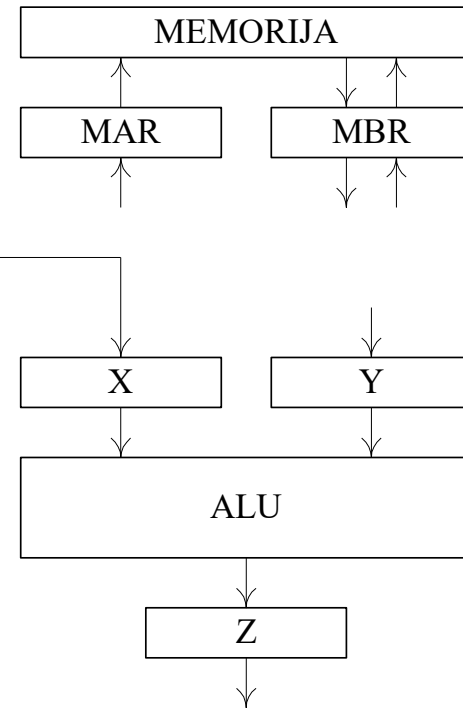
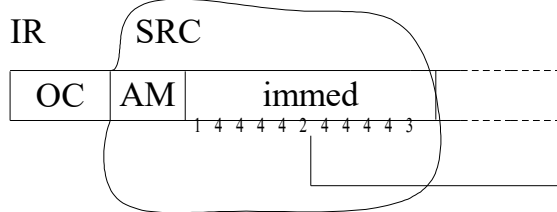
- Непосредно адресирање је адресирање код кога се операнд налази непосредно у самој инструкцији.

Непосредно адресирање

PC



IR

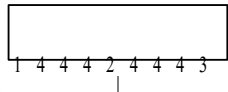


Регистарско директно адресирање

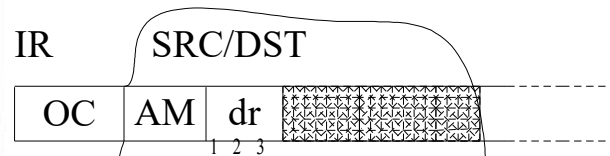
- Регистарско директно адресирање је адресирање код кога се операнд налази у једном од регистара података или регистара опште намене процесора.
- Код оних процесора код којих постоје посебно регистри података (DR), адресни регистри (AR), базни регистри (BR) и индексни регистри (XR), операнд се налази у једном од регистара података (DR), док код оних процесора код којих постоје само регистри опште намене (GPR), операнд се налази у једном од регистара опште намене.

Регистарско директно адресирање

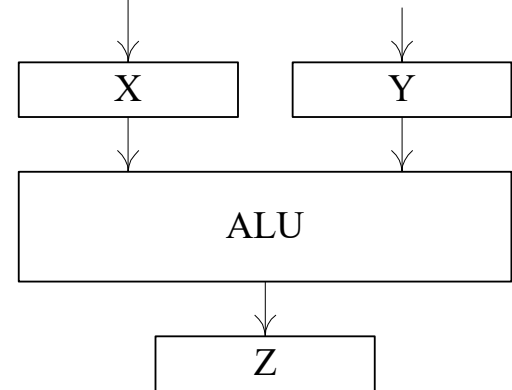
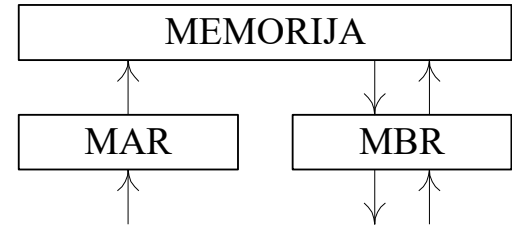
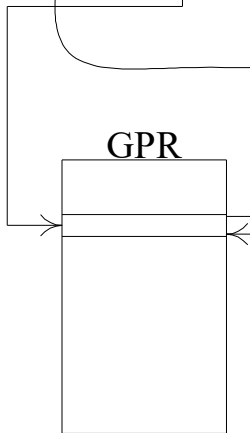
PC



IR

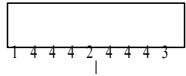


GPR

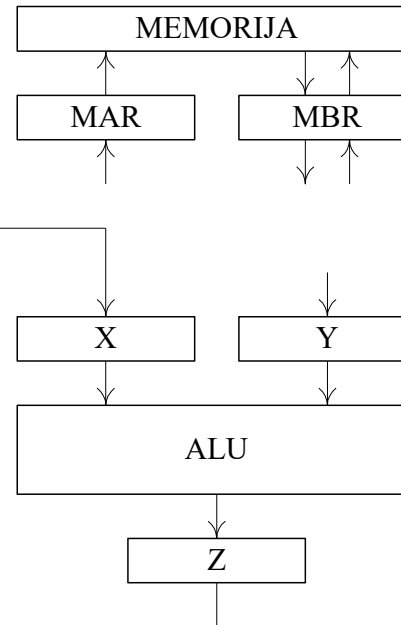
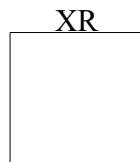
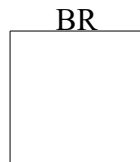
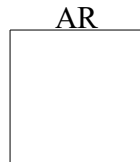
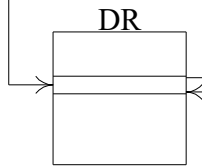
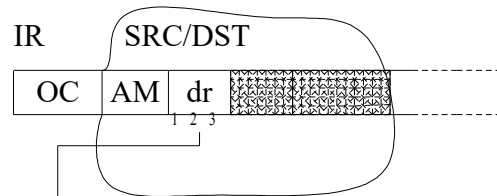


Регистарско директно адресирање

PC



IR

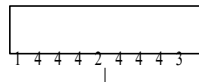


Регистарско индиректно адресирање

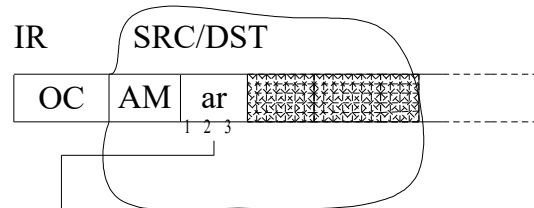
- Регистарско индиректно адресирање је адресирање код кога се операнд налази у меморијској локацији, а адреса меморијске локације се налази у једном од адресних регистара или регистара опште намене процесора.

Регистарско индиректно адресирање

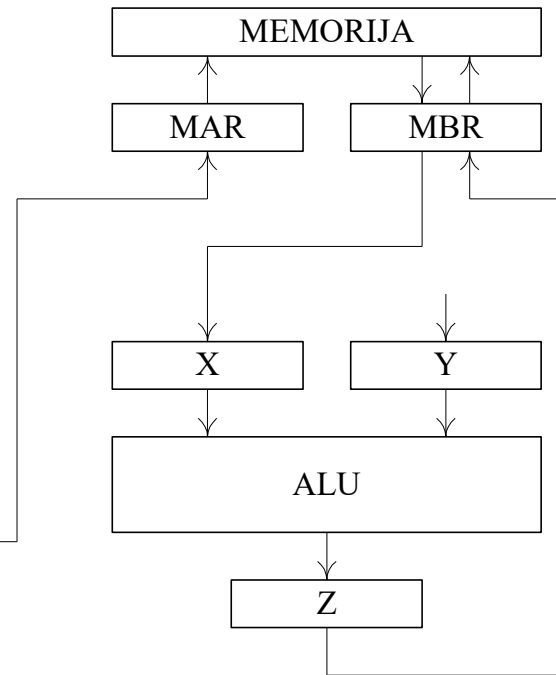
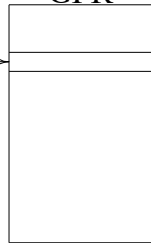
PC



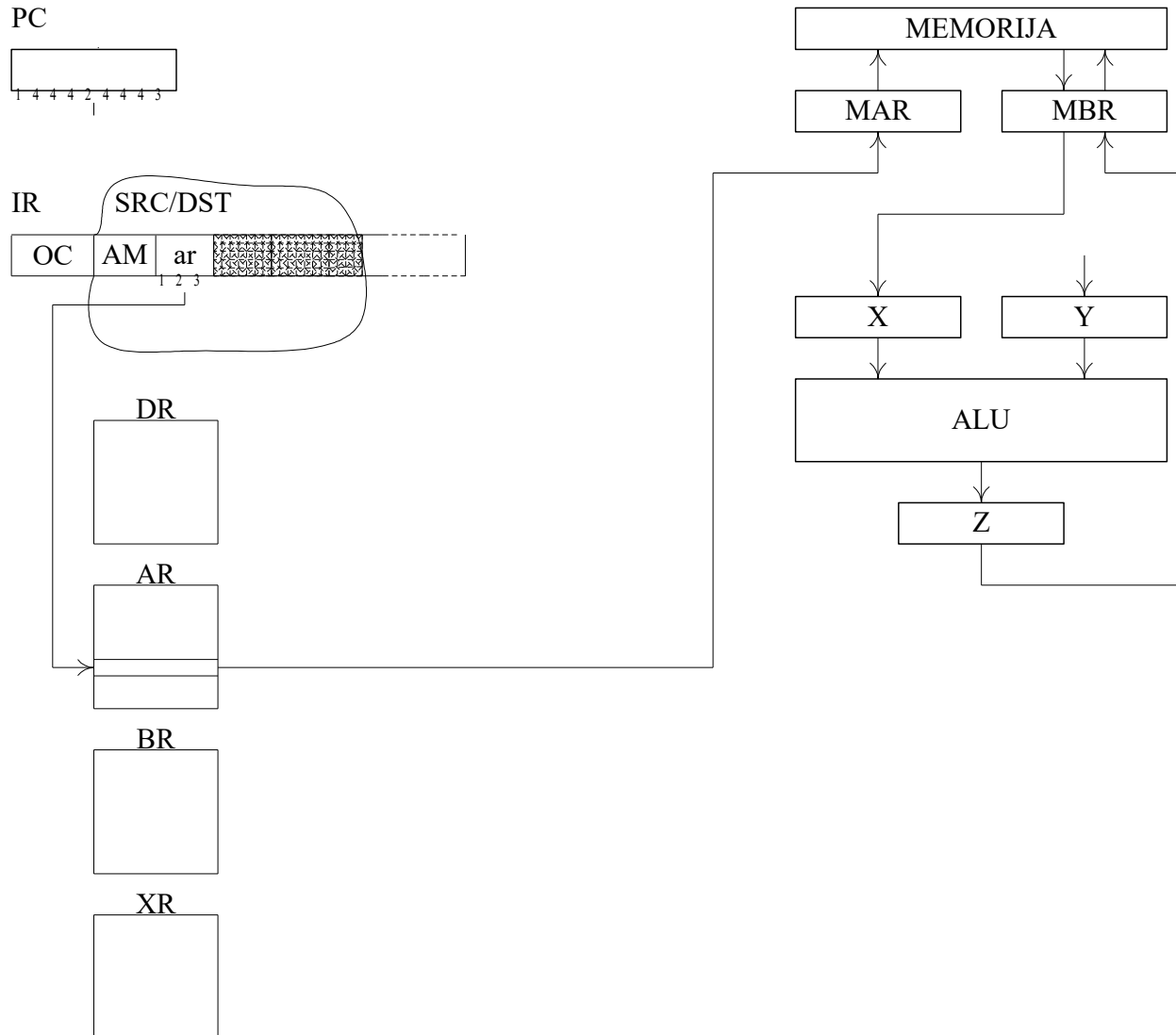
IR



GPR



Регистарско индиректно адресирање

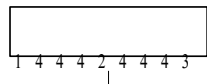


Регистарско индиректно са померајем

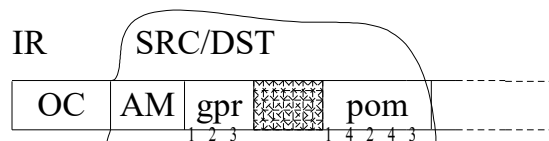
- Регистарско индиректно адресирање са померајем је адресирање код кога се операнд налази у меморији, а адреса меморијске локације се добија сабирањем садржаја једног од регистара опште намене и помераја.

Регистарско индиректно са померајем

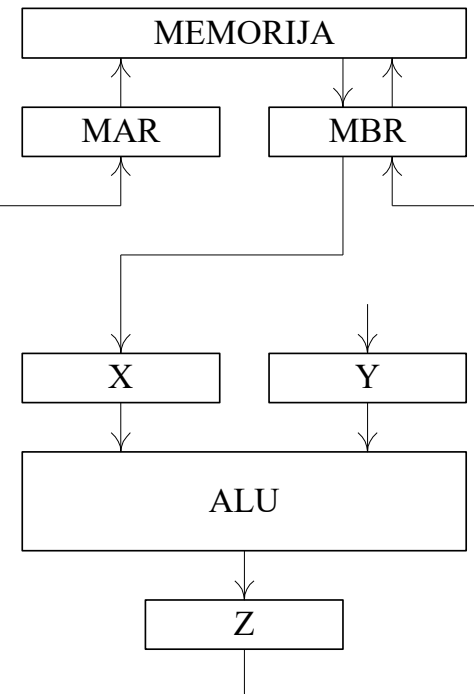
PC



IR



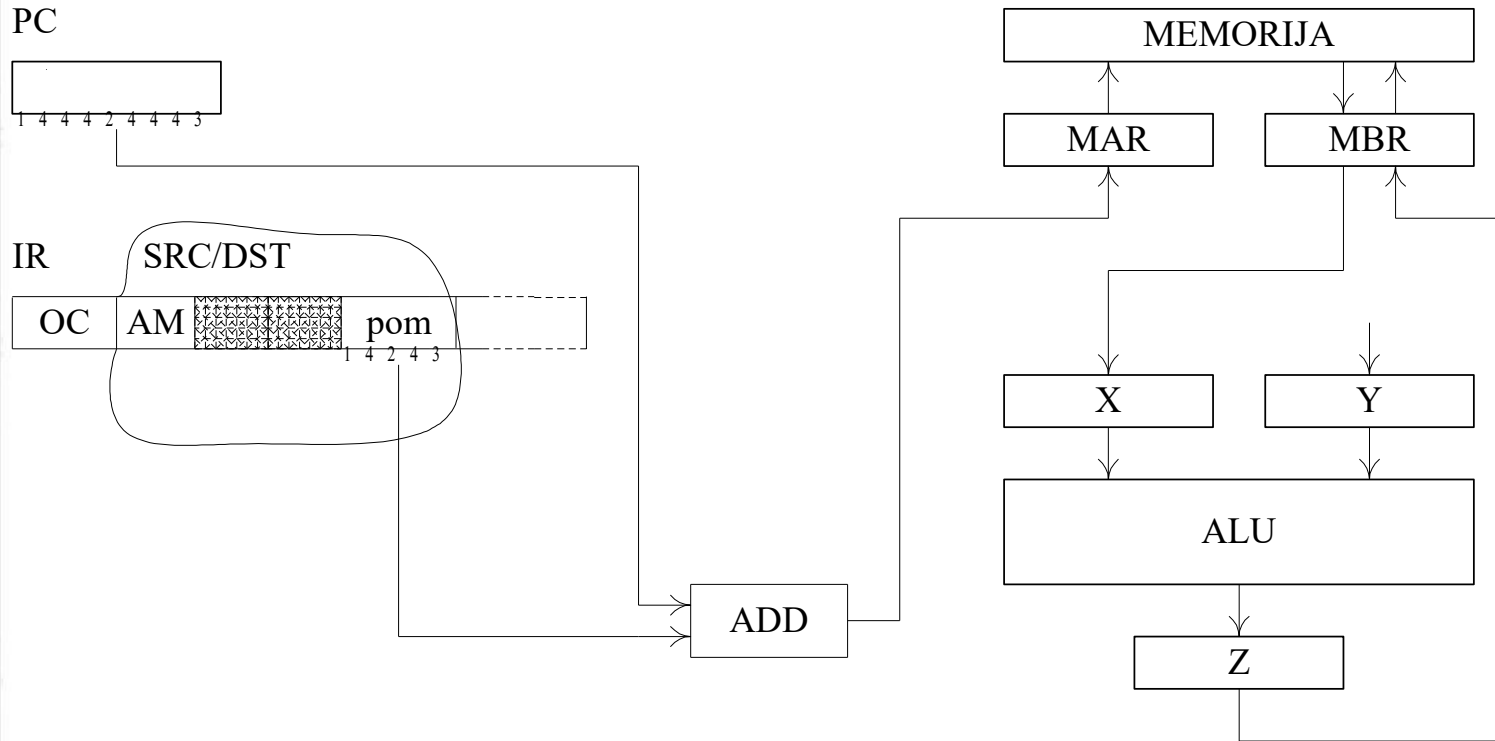
ADD



Релативно адресирање

- Релативно адресирање са померајем је адресирање код кога се операнд налази у једној од меморијских локација, а адреса меморијске локације се добија сабирањем садржаја програмског бројача РС и помераја.

Релативно адресирање са померајем

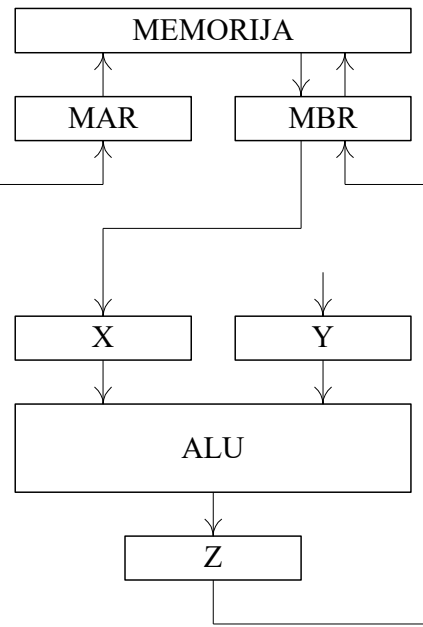
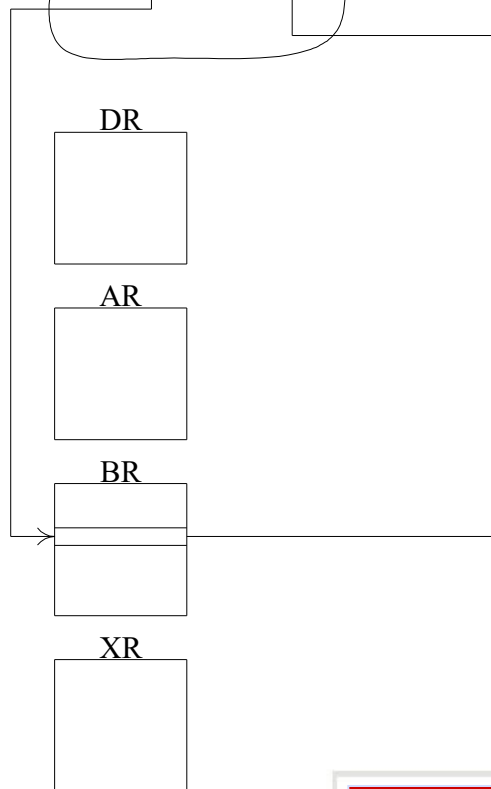
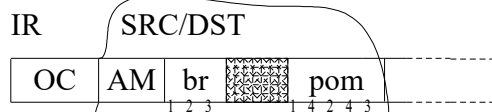
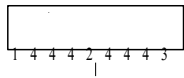


Базно адресирање са померајем

- Базно адресирање са померајем је адресирање код кога се операнд налази у меморији, а адреса меморијске локације се добија сабирањем садржаја једног од базних регистара и помераја.

Базно адресирање са померајем

- PC



Индексно адресирање са померајем

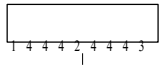
- Индексно адресирање са померајем је адресирање код кога се операнд налази у меморији, а адреса меморијске локације се добија сабирањем садржаја једног од индексних регистара и помераја.

*Уколико је величина операнда већа од онога што може у једном циклусу да се прочита из меморије индексни регистар може да се помножи величином операнда.

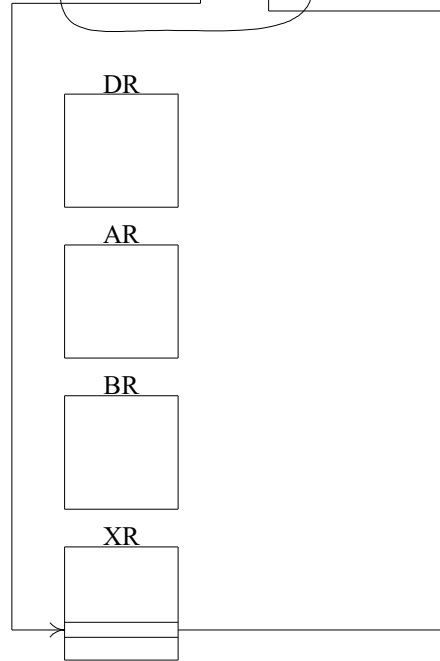
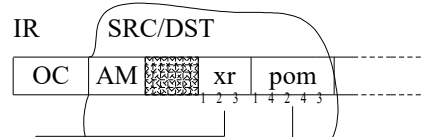
У том случају се адресирање назива скалирано.

Индексно адресирање са померајем

PC



IR



ADD

MEMORIJA

MAR MBR

X Y

ALU

Z

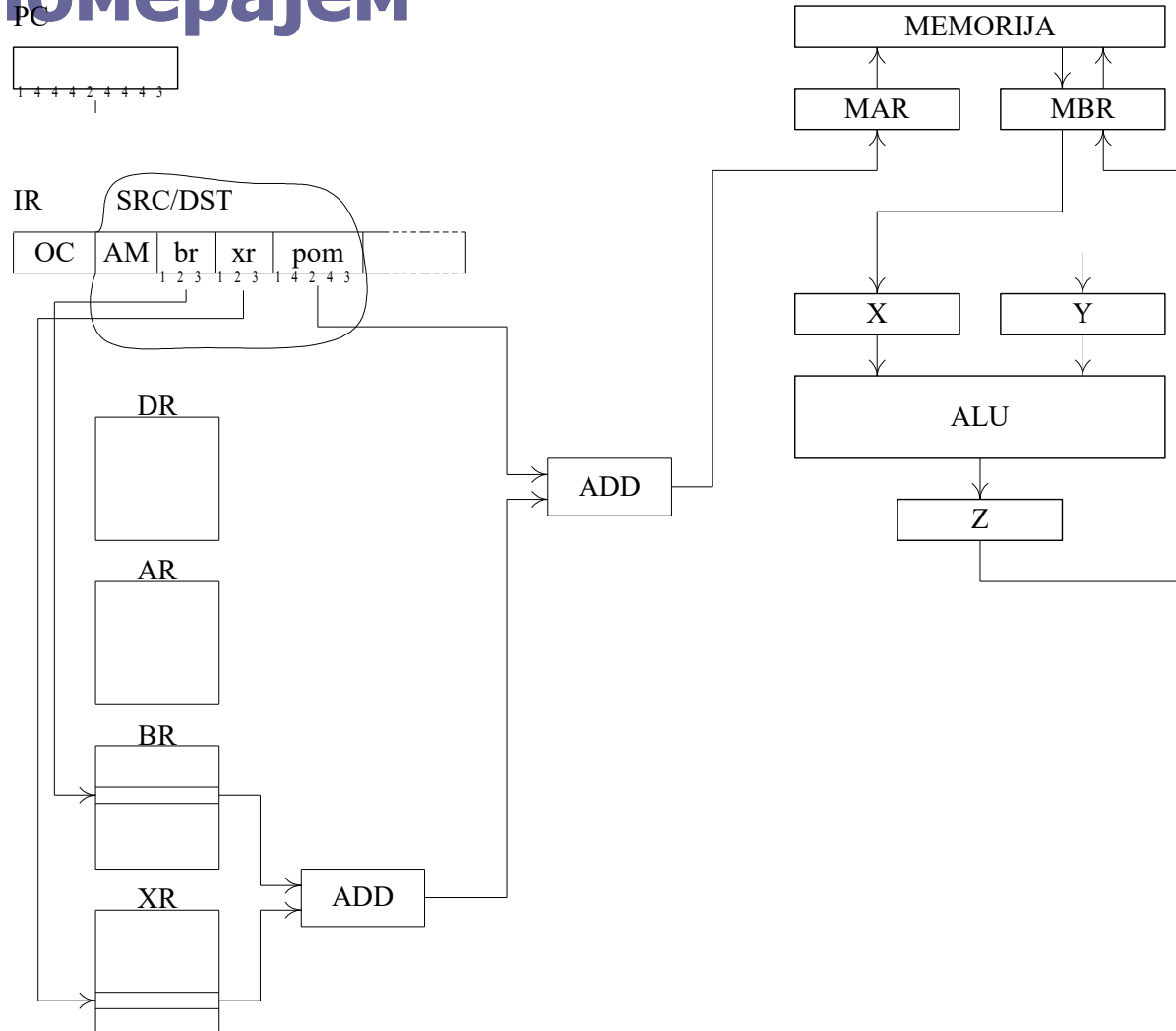
Базно-индексно адресирање са померајем

- Базно индексно адресирање са померајем је адресирање код кога се операнд налази у меморији, а адреса меморијске локације се добија сабирањем садржаја једног од базних регистара, једног од индексних регистара и помераја или сабирањем садржаја два регистра опште намене процесора и помераја.

*Уколико је величина операнда већа од онога што може у једном циклусу да се прочита из меморије индексни регистар може да се помножи величином операнда.

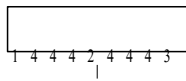
У том случају се адресирање назива скалирано.

Базно-индексно адресирање са померајем

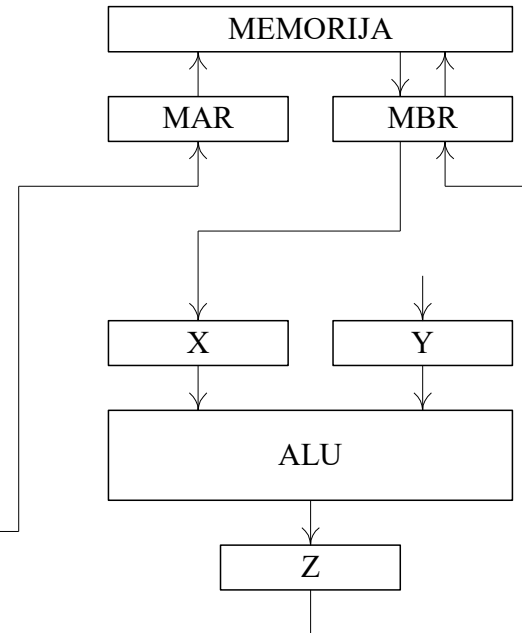
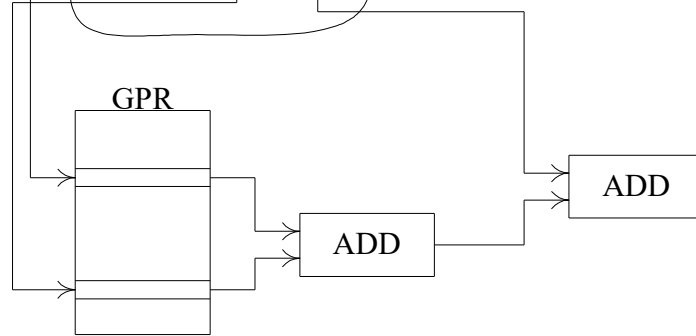
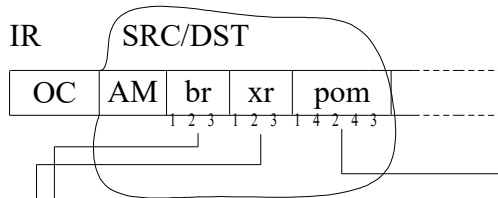


Базно-индексно адресирање са померајем

PC



IR



Регистарска индиректна са ауто inc/dec

- Регистарска индиректна адресирања са аутоинкрементирањем и аутодекрементирањем су веома слична регистарском индиректном адресирању, јер се и код ових адресирања као и код регистарског индиректног адресирања операнд налази у једној од меморијских локација, а адреса меморијске локације се налази у једном од регистара.

Регистарска индиректна са ауто inc/dec

- У зависности од тога да ли се прво инкрементира регистар па после тога његов садржај користи као адреса или се прво садржај регистра користи као адреса па се после тога инкрементира, регистарско индиректно адресирање са аутоинкрементирањем се јавља као преинкремент адресирање и постинкремент адресирање, респективно.

Адресирања

Начин адресирања	Пример	Значење	Типично коришћење
Меморијско директно (memdir)	Load 100h	ACC <= Mem[100h]	Приступа статичким подацима
Меморијско индиректно (memindir)	Load (100h)	ACC <= Mem[Mem[100h]]	Приступа коришћењем показивача
Непосредно (imm)	Load #100h	ACC <= 100h	За константе
Регистарско директно (regdir)	Load R1	ACC <= R1	Када је податак у регистру
Регистарско индиректно (regind)	Load (R1)	ACC <= Mem[R1]	Приступа помоћу показивача који је у регистру
Регистарско индиректно са померајем (reginddisp)	Load (R1)100h	ACC <= Mem[R1+100h]	За приступ локалним променљивама (Симулира меморијско директно (за R1=0) и регистарско индиректно (за померај једнак нули))
Релативно (rel)	Load (PC)100h	ACC <= Mem[PC+100h]	Приступ литералима (константама) и за релативне скокове

Адресирања

Начин адресирања	Пример	Значење	Типично коришћење
Базно са померајем (base)	Load (BP)100h	ACC <= Mem[BP+100h]	Приликом приступа локалним подацима у текућем блоку
Индексно са померајем (index)	Load (XR)100h	ACC <= Mem[XR+100h]	Приликом приступа елементима низа
Базно-индексно (baseindex)	Load (BP,XR)	ACC <= Mem[BP+XR]	Приликом приступа елементима низа који су у локалном блоку
Аутоинкрементирање (postinc)	Load (R1)+	ACC <= Mem[R1]; R1 <= R1 + d	За обилазак елемената низа који је почев од адресе на коју указује регистар како би се остварило померање на наредни елемент
Аутодекрементирање (predec)	Load -(R1)	R1 <= R1 - d; ACC <= Mem[R1]	За обилазак елемената низа са померањем на претходни елемент. auto inc/dec могу симулирати стек

Распоред бајтова (Byte ordering)

- Уколико се посматра целобројна величина дужине два бајта да ли се у меморију смешта прво млађи или старији бајт?
- Корите се два основна начина за смештање:
 - Уколико се млађи бајт смешта на нижу адресу ради се о ***little endian*** поретку.
 - Уколико се старији бајт смешта на нижу адресу ради се о ***big endian*** поретку.

Распоред бајтова

- Пример: у меморију почев од адресе 0x100 је потребно сместити целобројну величину дужине 4 бајта 0x12345678

- Уколико млађи бајт иде на нижу адресу:

Адреса	100	101	102	103
Little endian	78	56	34	12

- Уколико старији бајт иде на нижу адресу:

Адреса	100	101	102	103
Big endian	12	34	56	78

Типови података

- Најчешће коришћени типови података су:
 - целобројне величине
 - величине у покретном зарезу
 - алфанумерички низ
 - нумерички низ.

Интерпретација бинарне речи

- Ако се бинарна реч дужине n битова, у којој су битови означени са $a_{n-1}a_{n-2}\dots a_1a_0$, интерпретира као целобројна величина без знака, онда она представља податак A чија се вредност израчунава помоћу израза

$$A = \sum_{i=0}^{n-1} 2^i a_i$$

- Уз такав начин интерпретирања битова бинарне речи, представљају се целобројне величине без знака у опсегу 0 до 2^n-1 .

Интерпретација бинарне речи

- Међутим, ако се иста бинарна реч интерпретира као целобројна величина са знаком у другом елементу, онда она представља податак A чија се вредност израчунава помоћу израза

$$A = -2^{n-1} a_{n-1} + \sum_{i=0}^{n-2} 2^i a_i$$

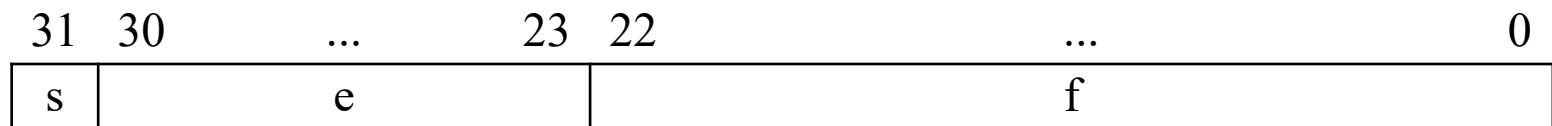
- Уз овакав начин интерпретирања битова бинарне речи, представљају се целобројне вредности са знаком у опсегу -2^{n-1} до $2^{n-1}-1$.

Интерпретација бинарне речи

- Целобројне величине фиксне дужине се у рачунарима представљају на фиксним дужинама од 8, 16, 32 и 64 бита и њихова дужина и начин интерпретације су одређени пољем кода операције инструкције.

Величине у покретном зарезу

- Величине у покретном зарезу имају:
 - поље знака (s),
 - поље експонента (e),
 - поље мантисе (f).
- Дужине су 32 или 64 бита.
- Пример:



Величине у покретном зарезу

31	30	...	23	22	...	0
s	e			f		

- Вредност се добија на следећи начин:
 - за $e=255$ и $f \neq 0$, v је Not a Number без обзира на s ,
 - за $e=255$ и $f=0$, $v=(-1)^s \infty$,
 - за $0 < e < 255$, $v=(-1)^s 2^{e-127} (1.f)$,
 - за $e=0$ и $f \neq 0$, $v=(-1)^s 2^{e-126} (0.f)$ – денормализовани број,
 - за $e=0$ и $f=0$, $v=(-1)^s 0$ (нула).

Питања?

Електротехнички Факултет
Универзитет у Београду

