

Милош Утвић

Информатички практикум 1

Процесор: рад и комуникација са примарном меморијом

Програм и програмске структуре

Степен аутоматизације обраде

- ▶ Ако упоредимо средства која се користе за извршавање неке радње (на пример, рачунање или прање веша), разликујемо **ручна, полуаутоматска и аутоматска** средства.
- ▶ Ручна средства помажу човеку да изврши радњу, али фактички све ради само човек (нпр. ручно прање чарапа, рачунање помоћу рачунаљке, тј. абакуса).
- ▶ Полуаутоматска средства могу сама да обаве појединачне кораке радње, али човек мора да зада сваки корак посебно: нпр. током рачунања помоћу калкулатора (дигитрона), човек задаје податке и операције, а дигитрон обавља рачун.

Програм

- ▶ Појам *програм* се среће на разним местима (програм културне манифестације, програм наставе, програм машине за прање веша, рачунарски програм, итд.).
- ▶ Средства за аутоматску обраду података омогућавају човеку да:
 - ▶ пре почетка обраде, поред података који треба да се обраде, зада и конкретне кораке обраде, њихов прецизан редослед и услове под којима се они извршавају.
 - ▶ током саме обраде човек не мора да интервенише, већ средство самостално изводи појединачне кораке обраде и провером услова доноси одлуке да ли неке кораке треба прескочити или поновити.
- ▶ Спецификација корака обраде које аутоматско средство треба да изврши одређеним редоследом и под одређеним условима назива се **програм**.

Терминологија

- У рачунарству се најмањи елементи (кораци) рачунарског програма који могу самостално да се изврше именују различито у зависности од тога на ком нивоу се посматра извршавање програма:

- На нивоу програмског језика (Pascal, C, C++, Java, C#, Python) се користи термин **наредба**, тј. програм је скуп наредби. Нпр. наредба доделе (за рачунање процентуалног износа):

`procenat = deo / celina;`

- На нивоу оперативног система (DOS, Windows, Linux, Macintosh) се користи термин **команда**, нпр. за приказ података о текстуалним датотекама у текућем каталогу (DOS, Windows Command Prompt)

`dir *.txt`

- На нивоу процесора се програм третира као скуп (машинских) **инструкција**

ADD X

Програмска структура



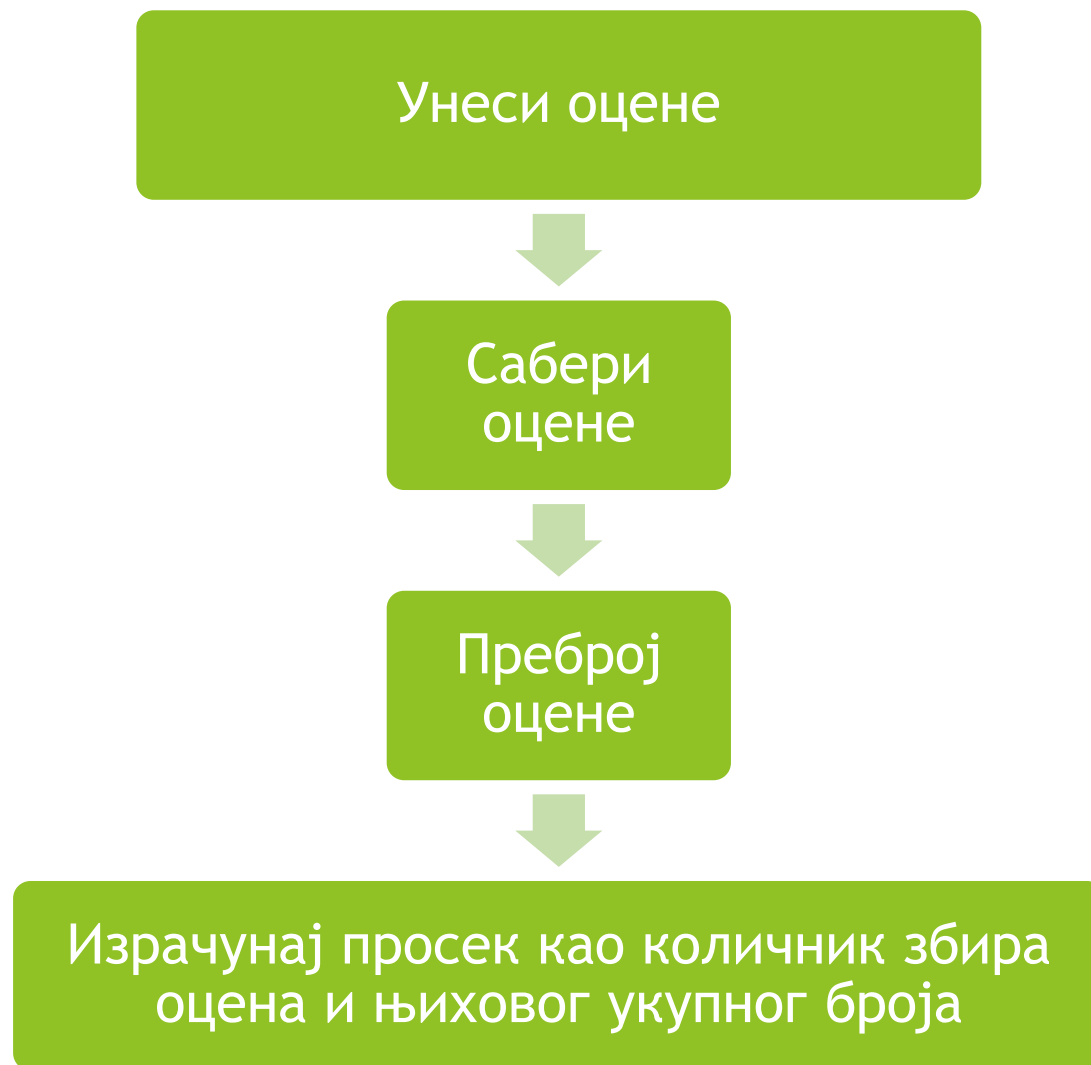
Пример:

- Улаз: унос оцена из свих предмета на крају студија за неког студента
- Обрада: израчунавање просечне оцене студента
- Излаз: исписивање просечне оцене студента на екран, у датотеку, штампање дипломе студента итд.

Редослед извршавања корака програма (наредби, команди, инструкција)

- ▶ У највећем броју случајева кораци програма се извршавају оним редоследом којим су записани у програму, тј. сваки корак се извршава после завршетка претходног корака, а онда се прелази на следећи корак.
- ▶ Таква структура програма се назива **линеарна структура**.
- ▶ Међутим, постоје и структуре програма као што су разграната и циклична, код којих редослед извршавања инструкција одступа од њиховог редоследа у запису програма.

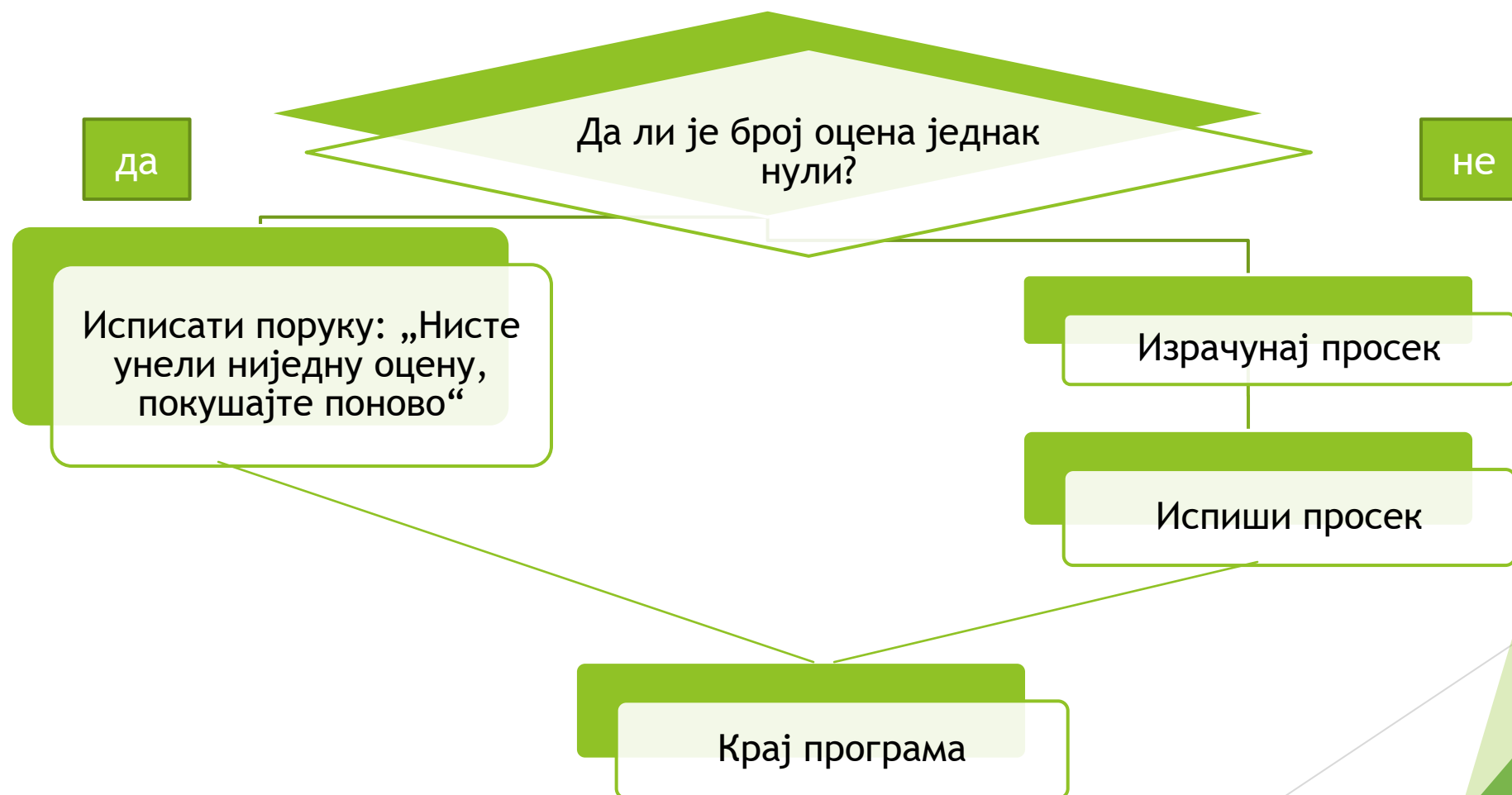
Линеарна структура (пример)



Разграната структура

- ▶ Понекад даље радње програма зависе од вредности добијене у неком кораку програма.
- ▶ Пример: када корисник програма уноси оцене, оне морају бити одређене вредности (5-10), тако да треба увести проверу да ли корисник уноси тражене вредности. Такође, ако корисник не унесе ниједну оцену, треба спречити програм да рачуна просек јер ће се у том случају просек свести на израз 0/0.

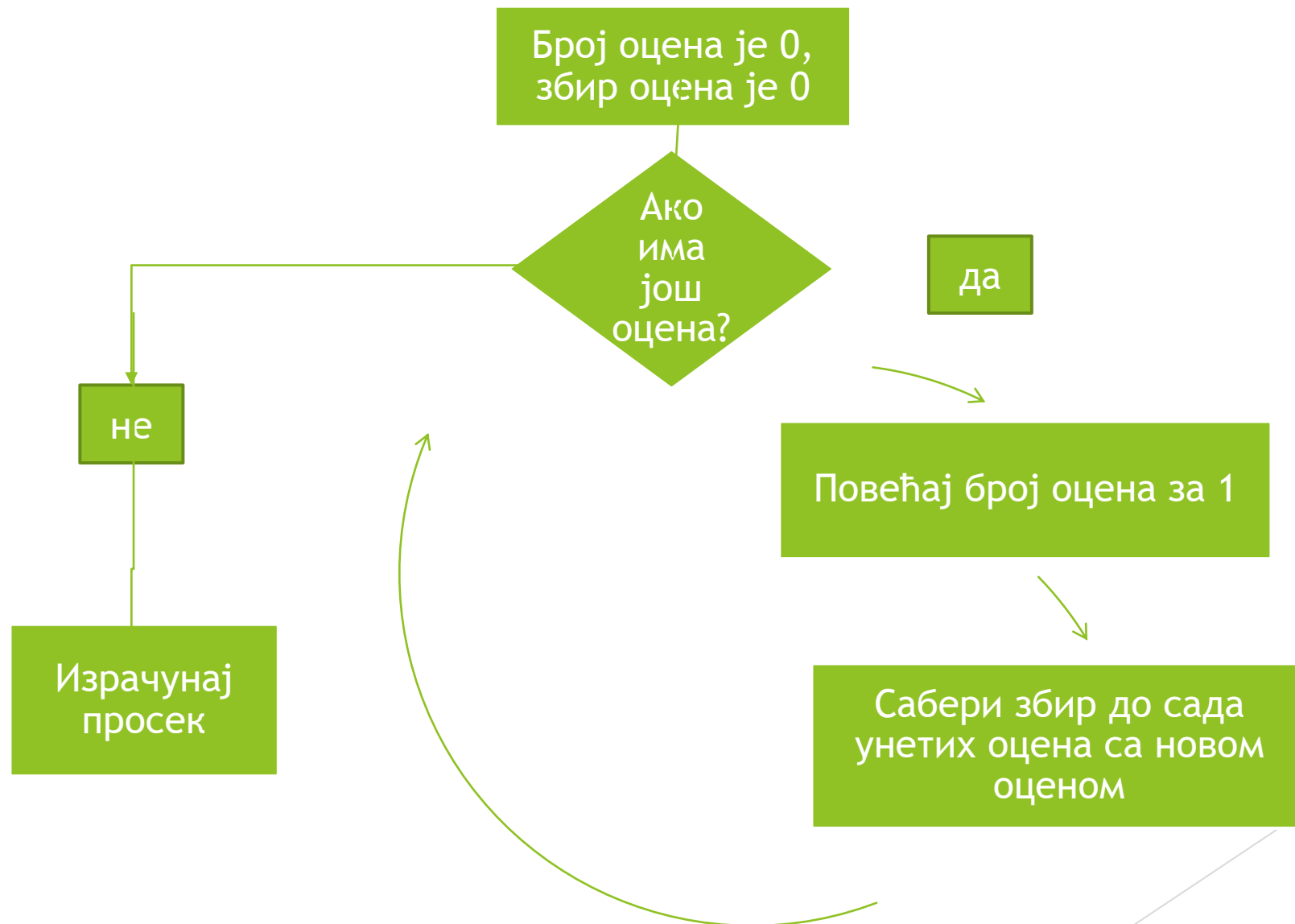
Разграната структура (пример)



Циклична структура (петља)

- ▶ Понекад је неопходно понављати одређени низ корака програма више пута, односно док је неки задати услов испуњен.
- ▶ Пример: ако се оцене студента уносе једна по једна, онда при сваком уносу треба урадити исти низ корака:
 - ▶ Повећати број оцена за један (пребројавање оцена)
 - ▶ Додати унету оцену на збир до тада унетих оцена.
- ▶ На почетку је збир оцена и број оцена једнак нули.
- ▶ Поступак се понавља све док има још оцена које треба унети.

Пример цикличне структуре (петље)



Упоредни преглед структуре делова програма

- ▶ Код линеарне структуре се сваки корак извршава тачно једном.
- ▶ Код разгранате структуре неки делови програма се извршавају највише једном, тј. једном или ниједном.
- ▶ Код цикличне структуре неки делови програма се могу извршавати једном, ниједном или више пута.
- ▶ Програм може садржати делове који користе сваку од наведене три структуре.

Инструкције и регистри процесора

Запис инструкције процесора

- ▶ Теоретски, највећи број инструкција процесора, поред ознаке радње која треба да се обави, захтева чак четири адресе:
- ▶ највише две адресе улазних података (аргумената),
- ▶ адресу резултата и
- ▶ адресу следеће инструкције која треба да се изврши
- ▶ На пример, могући запис инструкције за сабирање два броја би могао да изгледа овако:

ADD ADRESASABIRKA1 ADRESASABIRKA2 ADRESAZBIRA ADRESANAREDNEINSTRUKCIJE

- ▶ **ADD** је кôд (шифра) операције која треба да се обави (тзв. опкод);
- ▶ **ADRESASABIRKA1** и **ADRESASABIRKA2** су адресе првог и другог аргумента;
- ▶ **ADRESAZBIRA** је адреса резултата и
- ▶ **ADRESANAREDNEINSTRUKCIJE** је адреса следеће инструкције која треба да се изврши.

Четвороадресни процесор

- ▶ Процесор који користи инструкције са четири адресе назива се четвороадресни процесор.
- ▶ Добре особине четвороадресног процесора:
 - ▶ Омогућава да се приликом позива инструкције аргументи нађу на произвољним адресама.
 - ▶ Програми садрже најмањи број инструкција у односу на одговарајуће програме писане за процесоре са инструкцијама које користе мање од четири адресе.
- ▶ Лоше особине четвороадресног процесора:
 - ▶ Потребно је више меморије за запис инструкције.

Троадресни, двоадресни, једноадресни, нула-адресни процесори

- ▶ Уколико се одустане од тога да аргументи инструкције могу бити на произвољним адресама, могуће је смањити број аргумената инструкције, односно број адреса који се мора навести у инструкцији.
- ▶ Идеја је да се у сам процесор уграде меморијски елементи (регистри процесора) намењени за складиштење неких или свих аргумената инструкције. Тиме се постиже да у тренутку извршавања инструкције процесор унапред зна адресе појединих или свих аргумената, те инструкција може садржати мање од четири адресе.

Регистри процесора

- ▶ Регистри представљају „личну“ меморију процесора
- ▶ Представљају саставни део процесора, процесор им приступа директно, тако да је комуникација процесора са регистрима најбржа у односу на све остале меморије са којима процесор комуницира у рачунарском систему.
- ▶ Тренутна технологија израђује регистре као изузетно брзе електронске меморије ниског капацитета.
- ▶ Њихова величина се изражава у битовима, обично представља степен броја 2 и може бити од неколико битова (8) до неколико десетина битова (32, 64 итд.).

PC = бројач инструкција

- ▶ Посебан регистар процесора, који чува адресу следеће инструкције која треба да се изврши, назива се бројач инструкција (енгл. Program Counter, скр. PC).
- ▶ Регистар PC елиминише потребу за четвртом адресом у инструкцији.
- ▶ У случају линеарне структуре програма, тј. када се инструкције извршавају редоследом којим су записане у меморији, довољно је увећати PC за број бајтова којим се представља инструкција како би се задала адреса следеће инструкције која треба да се заврши. Нпр. ако се за запис једне инструкције користе два бајта и текућа инструкција је на адреси 50, следећа инструкција ће бити на адреси $50+2=52$ (ако се користи један бајт, следећа адреса је $50+1=51$, ако се користе четири бајта, следећа адреса је $50+4=54$ итд.).

АСС = Акумулатор

- ▶ Елиминација још две адресе се постиже увођењем регистра у процесору који се назива акумулатор. Акумулатор се користи како за смештање једног улазног податка пре извршења инструкције, тако и за складиштење резултата инструкције после њеног извршења.
- ▶ Ако се израчунавање које треба обавити у процесору састоји од више корака, неопходно је да се резултати појединих корака (тзв. међуреЗултати) привремено сачувају да би се искористили у наредним корацима.
- ▶ Осим што акумулатор поједностављује запис инструкције, уједно штеди време за смештање међуреЗултата и резултата израчунавања. Наиме, ако се међуреЗултати чувају у примарној меморији, одређено време се троши на упис међуреЗултата (њихово чување) и на читање међуреЗултата, тј. на њихов пренос из примарне меморије назад у процесор када поново затребају за следеће израчунавање.

Једноадресне инструкције

- ▶ Користећи регистре РС и АСС, број адреса које се користе у инструкцијама могуће је смањити на само једну адресу.
- ▶ РС елиминише потребу за четвртом адресом (адресом следеће инструкције).
- ▶ Ако се све операције које очекују два аргумента (бинарне операције попут сабирања) реализују тако да први улаз мора да дође из акумулатора и да резултат операције такође буде смештен у акумулатор, елиминишу се још две адресе.

Пример једноадресне инструкције

► ADD X

сабира вредност акумулатора са вредношћу која се налази у примарној меморији на адреси X и резултат (збир) уписује у акумулатор (тима се у акумулатору губи стара вредност која је послужила као први сабирак).

Додатне једноадресне инструкције за рад са акумулатором

- ▶ Међутим, коришћење акумулатора ствара потребу за два новим инструкцијама:
 1. **LOAD X**
учитава вредности из примарне меморије са адресе X у акумулатор („пуњење акумулатора“) и
 2. **STO X**
вредност акумулатора уписује на адресу X у примарној меморији (чување вредности акумулатора у примарној меморији).

Пример програма за процесор

Једноадресне инструкције усложњавају писање програма. Да би се обавило сабирање два броја који се налазе на меморијским локацијама $X=5$ (број 12) и $Y=6$ (број 8), уписао резултат (у овом случају 20) у меморијску локацију $Z=7$, а потом се прешло на следећу инструкцију (на адреси $W=4$), уместо једне четвороадресне инструкције

ADD X Y Z W

ADD 5 6 7 4

потребне су чак три једноадресне инструкције:

LOAD X

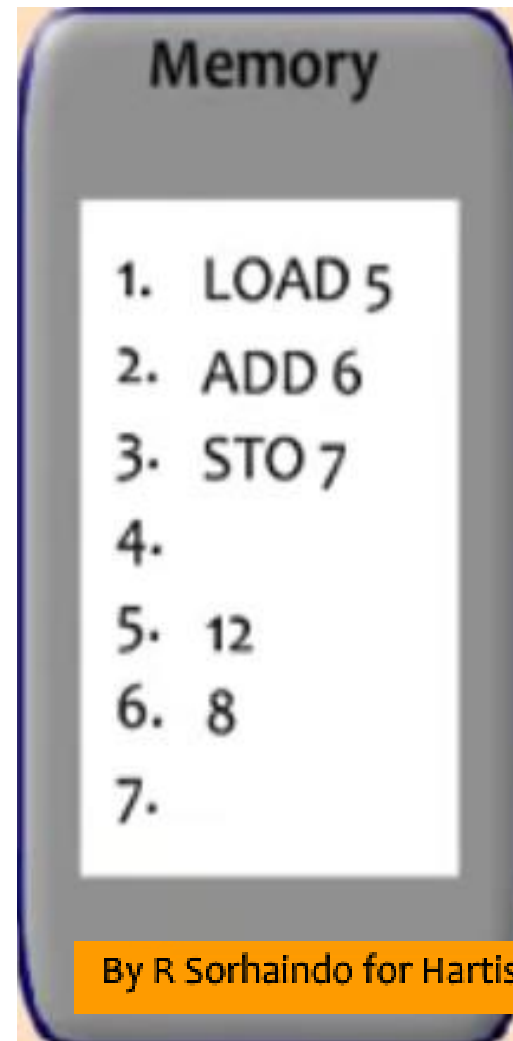
LOAD 5

ADD Y

ADD 6

STO Z

STO 7



Референце

- ▶ На претходном слајду је коришћен исечак слике снимљене као део анимације коју је направио R. Sorhaindo за државну средњу школу **Hartismere**. Анимација представља фазе у извршењу једне инструкције. Снимци анимације биће коришћени и у следећем одељку *Комуникација процесора и примарне меморије*. и на њих ће се реферисати помоћу

By R Sorhaindo for Hartismere

- ▶ Flash-верзија анимације:

<http://www.hartismere.com/staticvle/ictskills/FetchExecute.swf>

- ▶ Youtube-верзија анимације:

<https://www.youtube.com/watch?v=04UGopESS6A>

Комуникација процесора и примарне меморије

Магистрале (бусеви)

- ▶ Комуникација процесора и меморије се обавља преко три канала везе који се називају магистрале или бусеви (енгл. bus):
 - ▶ Адресне магистрале
 - ▶ Податковне магистрале или магистрале података и
 - ▶ Контролне магистрале

Адресна магистрала

- ▶ Комуникација се одвија само у једном смеру, од процесора ка меморији.
- ▶ Процесор шаље адресу меморијске локације у примарној меморију коју жели да чита или у коју жели да уписује податке.
- ▶ Пре слања адреса мора да се нађе у посебном регистру процесора који се зове регистар меморијске адресе (енгл. **Memory Address Register**, скр. **MAR**).

Податковна магистрала

- ▶ Комуникација може да се одвија у оба смера, од процесора ка примарној меморији или од примарне меморије ка процесору, али никако истовремено.
- ▶ Овом магистралом се шаљу подаци.
- ▶ Процесор шаље податак који жели да упише у примарну меморију. Пре слања податак мора да се нађе у посебном регистру процесора који се зове регистар меморијског податка (енгл. **Memory Data Register**, скр. **MDR**).
- ▶ Примарна меморија шаље податак који процесор жели да прочита. Када податак стигне до процесора, чува се у регистру **MDR**. Према томе, може да се каже да податковна магистрала заправо повезује **MDR** и примарну меморију.

Контролна магистрала

- ▶ Контролном магистралом се шаљу сигнали којима се управља адресном и податковном магистралом, односно којима процесор и примарна меморија размењују обавештења типа:
 - ▶ Да ли процесор жели да чита из примарне меморије или да у њу уписује.
 - ▶ Да ли је меморија спремна за читање или упис.
 - ▶ Да ли је послата адреса или податак стигао на одредиште.

Фазе извршења инструкције

Јединице процесора. ALU

- ▶ Процесор се састоји из две јединице:
 - ▶ аритметичко-логичка јединице (енгл. скр. ALU) и
 - ▶ контролно-комуникационе јединице.
- ▶ Аритметичко-логичка јединица садржи логичка кола (комбинаторна и секвенцијална) помоћу којих се симулирају операције са подацима (пре свега са целим бројевима и логичким вредностима), одлучивање испитивањем неког услова, као и промена адресе следеће инструкције која треба да се изврши (тзв. скокови у програму, условни и безусловни).

Контролна јединица процесора

- Контролна јединица процесором је добила име по томе што контролише процес припреме и извршења инструкције и усклађује рад процесора, примарне меморије и улазних и излазних уређаја. Између осталог, контролна јединица процесоре генерише и прима сигнале који путују контролном магистралом (на пример, да ли процесор жели да чита из меморије или да пише у меморију). Такође, када текућа инструкција стигне из меморије у процесор, контролна јединица препознаје о којој се инструкцији ради и генерише сигнале који омогућавају њено извршење (в. Декодер).

(C)IR = командни регистар (регистар инструкције)

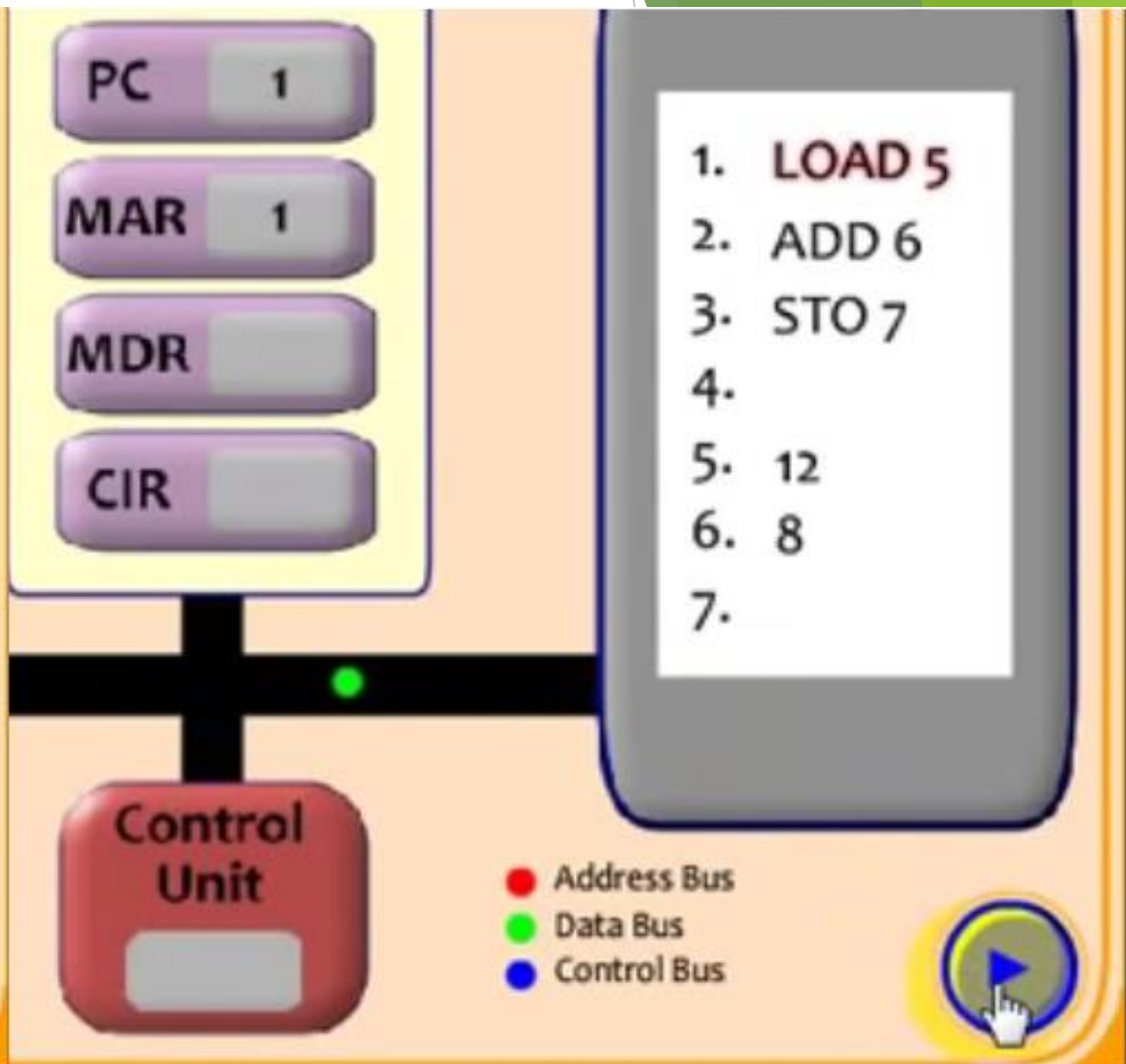
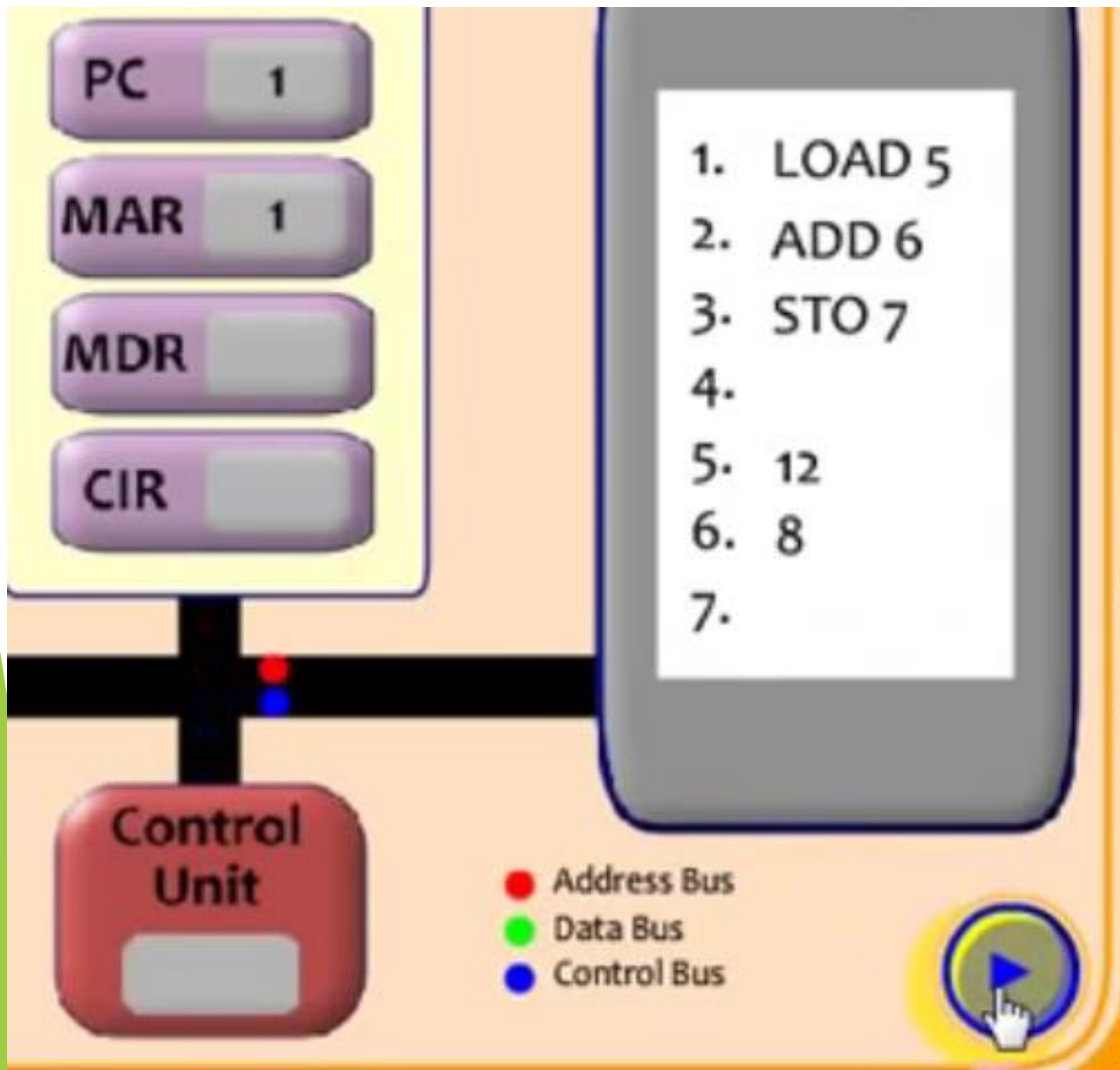
- ▶ Када процесор затражи од примарне меморије следећу по реду инструкцију коју треба да изврши (текућу инструкцију), смешта је у регистар који се назива **командни регистар** или **регистар инструкције** (енгл. **Instruction Register**, скр. **IR**, или **Current Instruction Register**, скр. **CIR**).
- ▶ Величина регистра (C)IR одговара величини јединице примарне меморије (RAM) која се користи за смештање једне инструкције (32 бита, 64 бита, итд.).

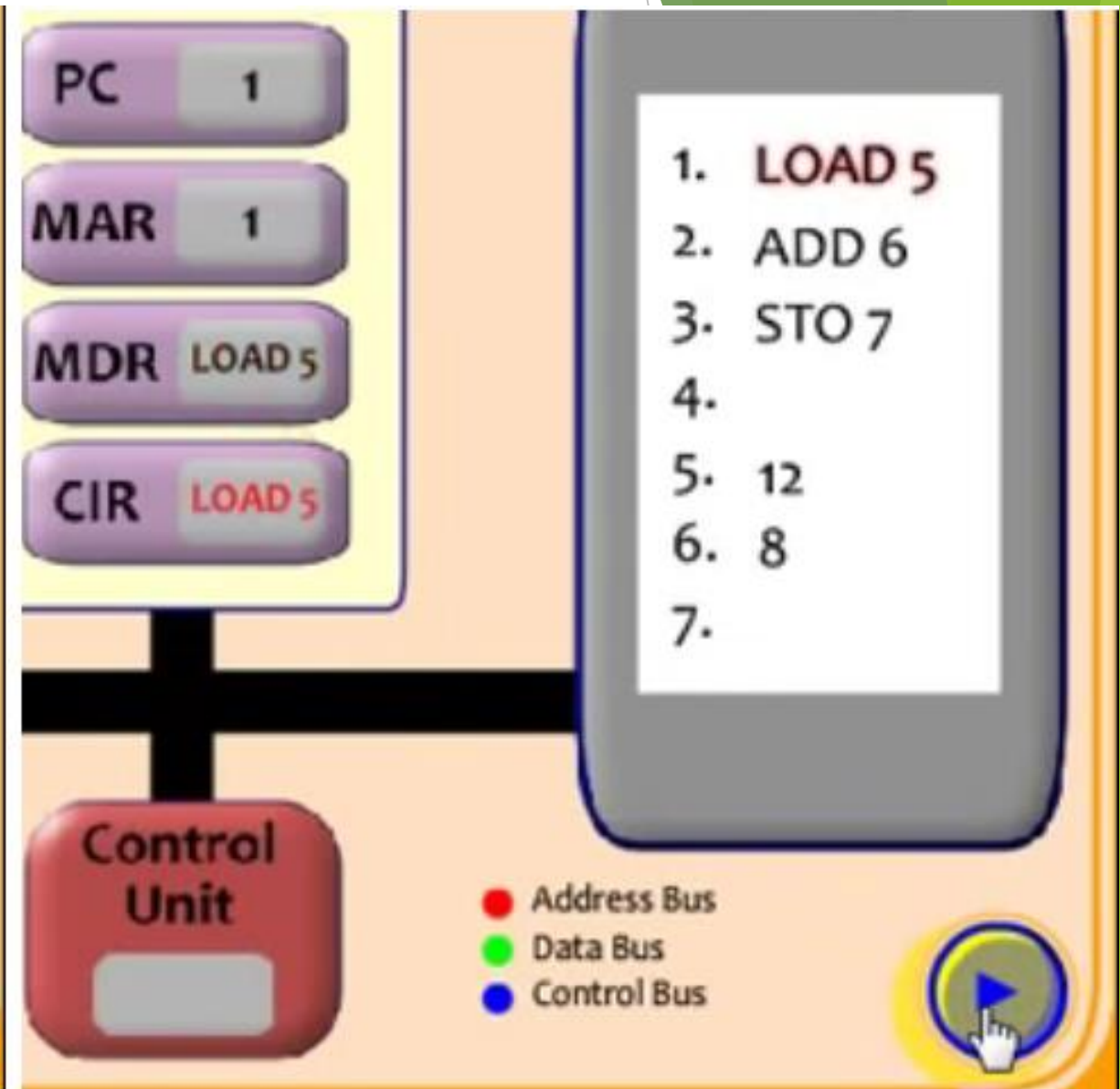
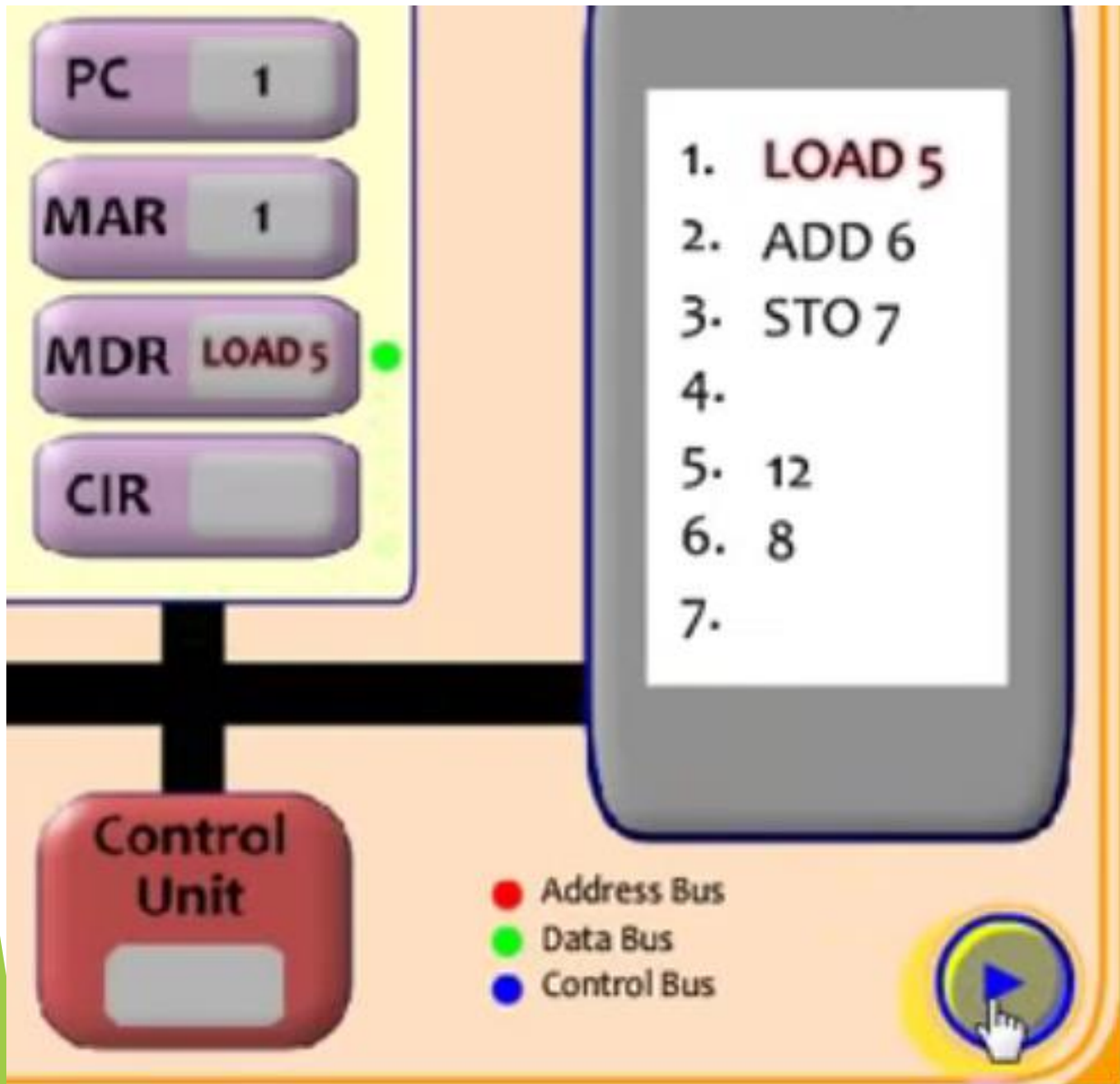
Фазе извршења текуће (једноадресне) инструкције

- ▶ Једноадресна инструкција је облика **опкод адресаАргумента**
- ▶ Извршење сваке инструкције процесора се обавља у две фазе:
 - ▶ Припремна фаза
 - ▶ Извршна фаза

Припремна фаза извршења инструкције

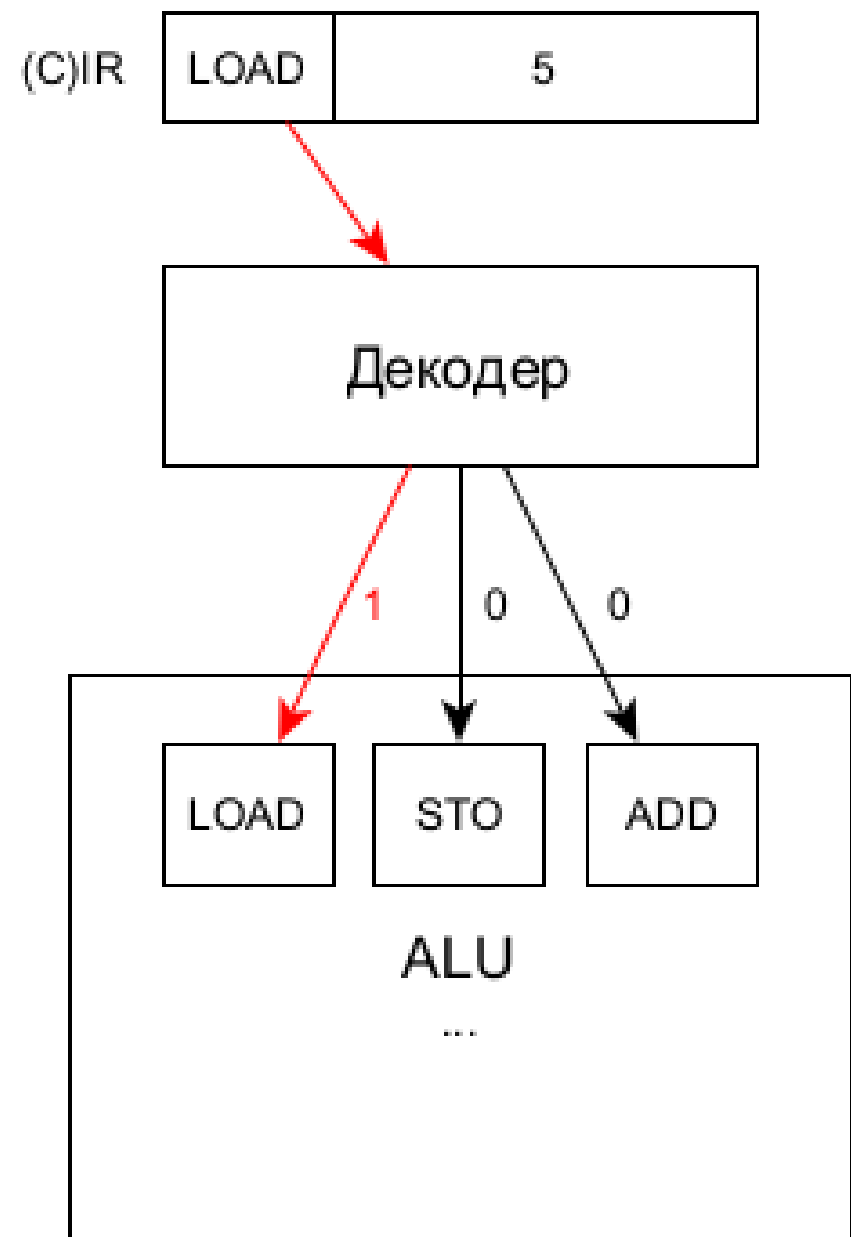
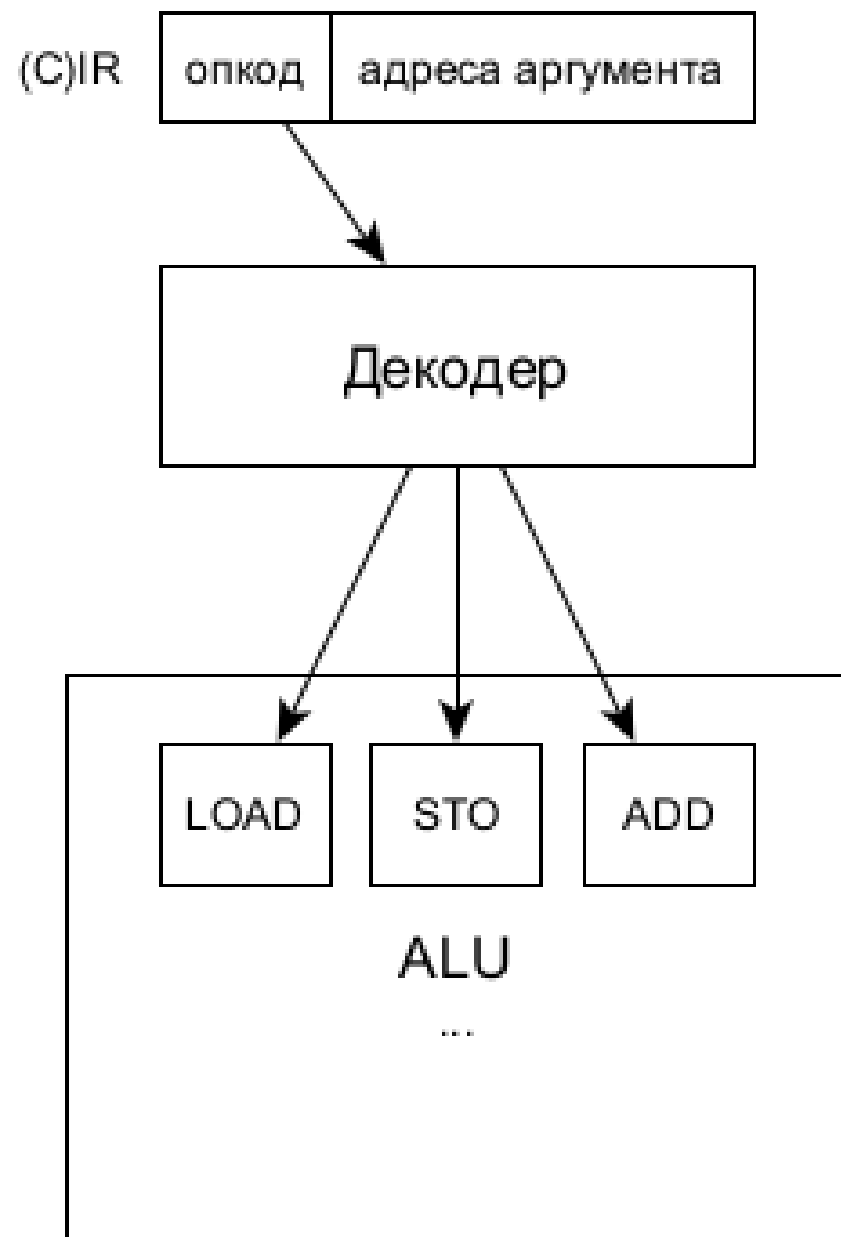
- ▶ Следећа два слајда (П1 и П2) илуструју припремну фазу.
- ▶ (П1, лево) Најпре се утврђује адреса (садржај регистра РС, тј. 1) следеће инструкције која треба да се изврши. Процесор затим захтева да се следећа инструкција копира из примарне меморије у командни регистар процесора. Према томе, садржај РС (1) се копира у MAR, а одатле шаље адресном магистралом до примарне меморије, док се контролном магистралом шаље захтев за читањем.
- ▶ (П1, десно) Примарна меморија одговара тако што проналази садржај (LOAD 5) на задатој адреси (1) и шаље га податковном магистралом до регистра MDR у процесору (П2, лево). Припремна фаза се завршава копирањем прочитане инструкције из регистра MDR у (C)IR (П2, десно).





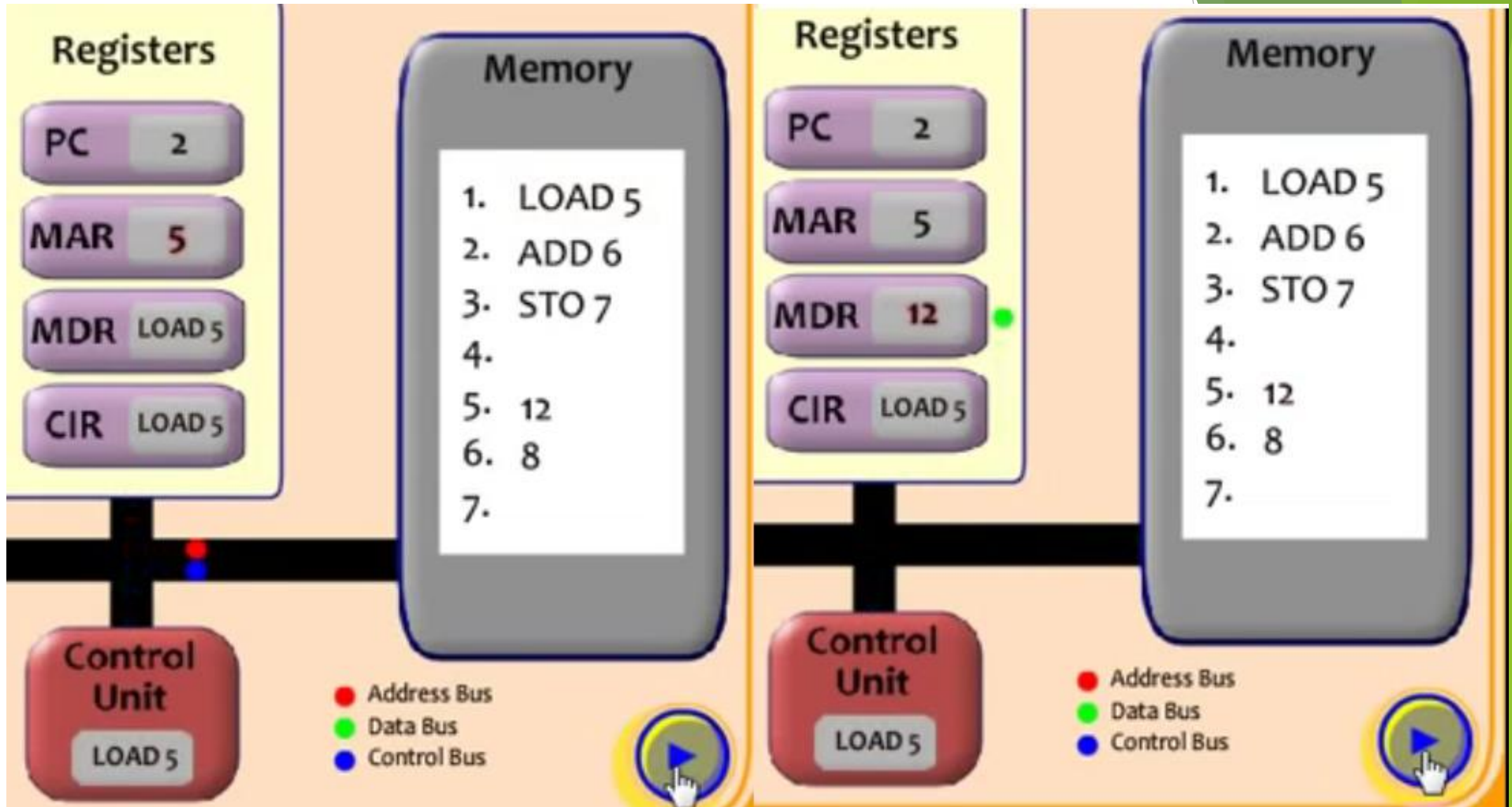
Извршна фаза извршења инструкције

- ▶ У извршној фази се опкод текуће инструкције, смештене у регистру (C)IR, шаље се контролну јединицу процесора, у посебно логичко коло: декодер. Назив кола потиче од тога што декодира опкод, тј. препознаје о којој се инструкцији ради.
- ▶ Сваком типу инструкције који процесор може да изврши одговара тачно један опкод и једно логичко коло у аритметичко-логичкој јединици (ALU). Логичко коло је заправо хардверска реализација инструкције, а опкод је ознака помоћу које се успоставља веза између инструкције и одговарајућег логичког кола.
- ▶ Излаз декодера је повезан са свим колима инструкција у ALU. Када декодер препозна опкод инструкције, сигнал на његовом излазу се шаље у ALU где активира само коло које одговара опкоду текуће инструкције, а блокира сва остала кола. Активирано коло у ALU извршава текућу инструкцију.



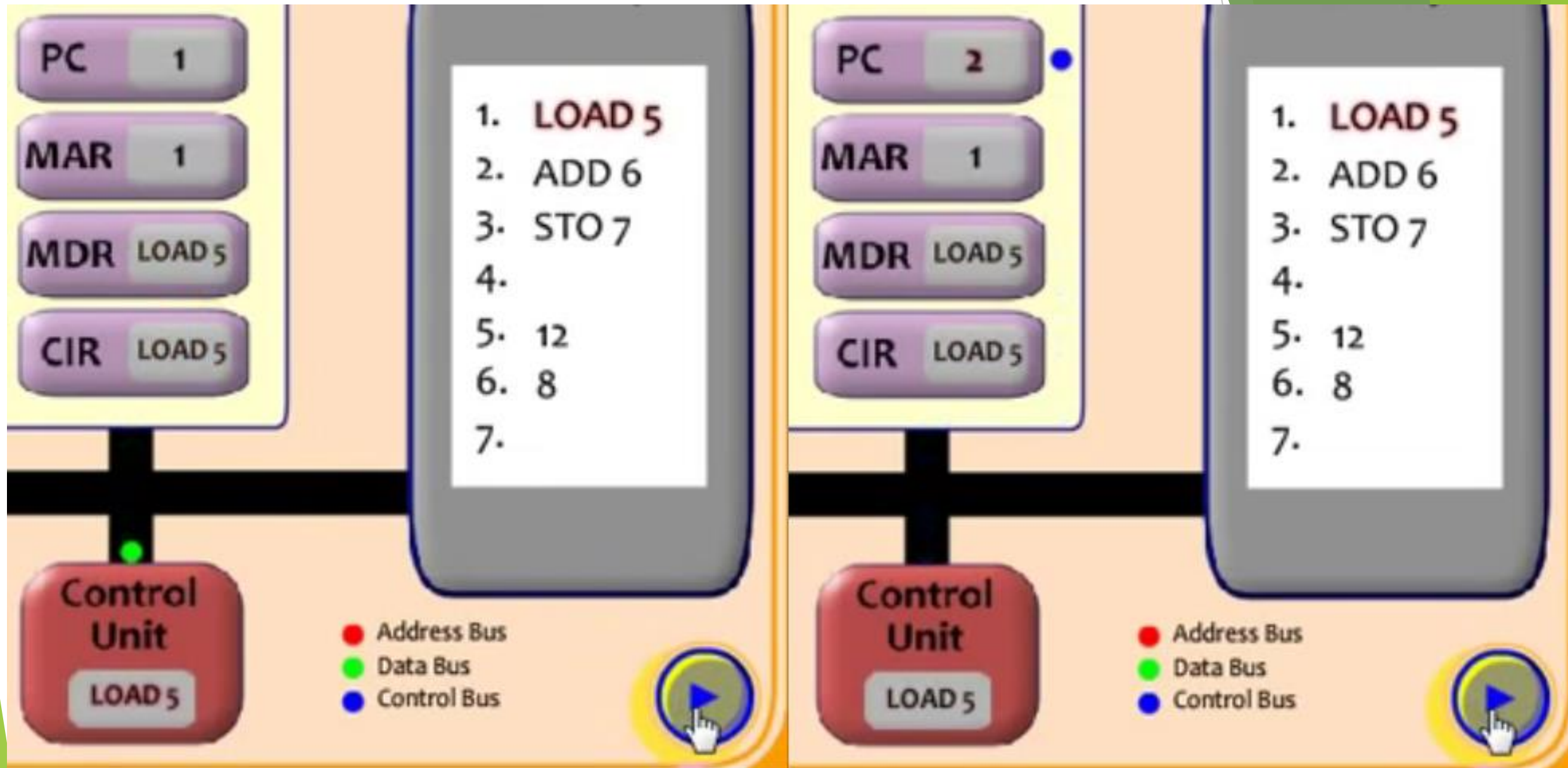
Извршавање инструкције LOAD

- ▶ Следећи слајд (Л1) илуструје комуникацију процесора и меморије током извршења прве инструкције једног машинског програма (LOAD 5) којом се чита садржај са меморијске адресе 5 (12).
- ▶ (Л1, лева слика) У MAR се уписује 5 (меморијска адреса са које се чита);
- ▶ (Л1, лева сл.) Адресном магистралом се шаље садржај регистра MAR (5);
- ▶ (Л1, лева сл.) Контролном магистралом се шаље сигнал да процесор жели да чита из меморије са адресе коју је послао адресном магистралом.
- ▶ (Л1, десна сл.) Примарна меморија проналази меморијску локацију 5 и њен садржај (12) шаље податковном магистралом, а о томе обавештава процесор контролном магистралом.
- ▶ (Л1, десна сл.) Прочитани садржај (12) стиже до процесора и смешта се у регистар MDR. Процесор контролном магистралом обавештава меморију да је податак успешно прочитан.
- ▶ Садржај регистра MDR (12) копира се у акумулатор.



Напомене

- ▶ У току неке од фаза извршења инструкција се ажурира вредност регистра РС и тиме одређује адреса следеће инструкције која треба да се изврши (видети следећи слајд).
- ▶ Припремна фаза, као и препознавање декодером за преостале две инструкције програма (ADD 6, STO 7) се реализују на потпуно исти начин као у случају прве инструкције програма (LOAD 5).
- ▶ Извршне фазе за ADD 6 и STO 7 су специфичне и њихово објашњење следи на наредним слајдовима.



Извршавање инструкције ADD

- ▶ Претпоставимо да процесор жели да изврши инструкцију ADD 6, тј. да сабере вредност акумулатора (12) и садржај са меморијске адресе 6 (8):
- ▶ У MAR се уписује 6 (меморијска адреса са које се чита);
- ▶ Адресном магистралом се шаље садржај регистра MAR (6);
- ▶ Контролном магистралом се шаље сигнал да процесор жели да чита из меморије са адресе коју је послао адресном магистралом.
- ▶ Примарна меморија проналази меморијску локацију 6 и њен садржај (8) шаље податковном магистралом, а о томе обавештава процесор контролном магистралом.
- ▶ Прочитани садржај (8) стиже до процесора и смешта се у регистар MDR. Процесор контролном магистралом обавештава меморију да је податак успешно прочитан.
- ▶ Процесор сабира вредност акумулатора (12) и регистра MDR (8) и резултат (20) се уписује у акумулатор као његова нова вредност.

Извршавање инструкције STO

- ▶ Претпоставимо да процесор жели да изврши инструкцију STO 7, тј. да упише садржај акумулатора (20) у меморијске адресу 7:
- ▶ У MAR се уписује 7 (меморијска адреса на коју се пише);
- ▶ Садржај акумулатора (20) копира се у регистар MDR.
- ▶ Адресном магистралом се шаље садржај регистра MAR (5);
- ▶ Процесор шаље податковном магистралом садржај регистра MDR.
- ▶ Контролном магистралом се шаље сигнал да процесор жели да пише у меморију на адресу коју је послао адресном магистралом.
- ▶ Примарна меморија проналази меморијску локацију 7 (послату адресном магистралом) и на ту адресу уписује 20 (послат податковном магистралом).
- ▶ Примарна меморија обавештава процесор контролном магистралом да је податак успешно уписан.