2.1.3 Принципи за дизајн кај современите компјутери

- Се нарекуваат уште **RISC принципи за дизајн**
- RISC = Reduced Instruction Set Computer (Компјутер со редуцирано инструкциско множество)
  - Директно хардверско извршување на инструкциите
    - Не се интерпретираат со микроинструкции
    - Наспроти нив, кај CISC компјутерите (CISC = Complex Instruction Set Computer – Компјутер со комплексно инструкциско множество), покомплексните инструкции треба да се разбиваат на делови, кои потоа ќе бидат извршувани како низи од микроинструкции
      - Тоа доведува до успорување на машината и може да биде прифатливо само за инструкции кои поретко се појавуваат!

2.1.3 Принципи за дизајн кај современите компјутери

- Максимизирање на интензитетот на **издавање (issue)** на инструкциите
  - Не е толку важно времетраењето на извршувањето на инструкциите, колку што е важен **бројот на инструкции кои можат да започнат со извршување (да бидат издадени) во една секунда**
  - Паралелизмот игра главна улога во подобрувањето на перформансите, бидејќи издавање на голем број бавни инструкции во кус временски интервал е можно само ако **истовремено** се извршуваат повеќе инструкции

2.1.3 Принципи за дизајн кај современите компјутери

- Едноставно декодирање на инструкциите
  - Декодирањето на одделните инструкции (со цел да се утврдат потребните ресурси) претставува сериозно ограничување за интензитетот на издавање
  - Се настојува инструкциите да бидат со фиксна должина и со помал број полиња – колку е помал бројот на различни формати на инструкции, толку подобро
  - Една инструкција се состои од:
    - Код на операцијата (opcode)
    - Адреси (од каде доаѓаат операндите и каде треба да се запише резултатот)

2.1.3 Принципи за дизајн кај современите компјутери

- Обраќање до меморијата со посебни инструкции
  - Еден од наједноставните начини за разбивање на операциите во одделни чекори е настојувањето операндите за повеќето инструкции да доаѓаат **од-** и да се враќаат **во-** регистри
  - Пренесувањето на операнди помеѓу регистрите и меморијата може да биде изведено со посебни инструкции
    - LOAD – читање од меморија
    - STORE – запишување во меморија

2.1.3 Принципи за дизајн кај современите компјутери

- Обезбедување на голем број регистри
  - Меѓу другото, бидејќи пристапот до меморијата е релативно бавен, се настојува да се обезбедат голем број регистри (најмалку 32) – тогаш, зборовите кои се преземени од меморијата можат да се чуваат во регистрите се’ до моментот кога веќе нема да бидат потребни

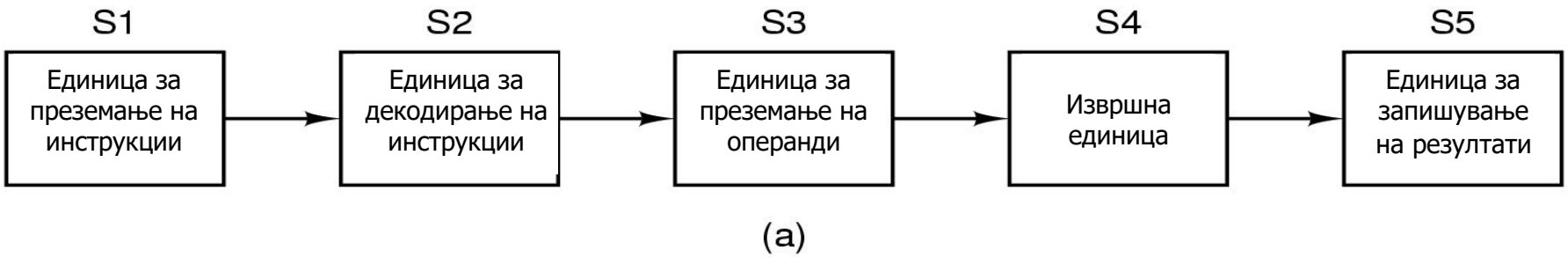
2.1.4 Паралелизам на инструкциско ниво

- Подобрување на перформансите
  - Зголемување на брзината на системскиот часовник (clock speed)
  - Паралелизација (извршување на две или повеќе работи одеднаш)
    - **Паралелизам на инструкциско ниво** – извлекување на колку што е можно повеќе паралелизам од секвенцијалната низа од инструкции
    - **Паралелизам на процесорско ниво** – повеќе процесори работат заедно (на ист проблем)

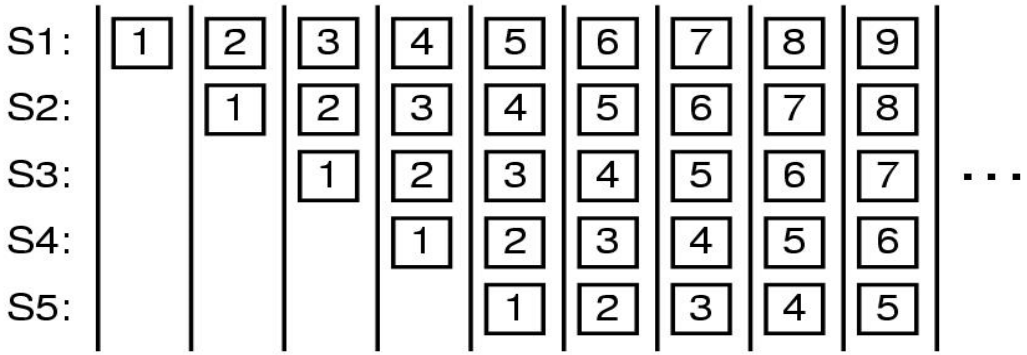
2.1.4.1 Протечна обработка (pipelining)

- Обработката на инструкциите се одвива во неколку фази, опслужувани од различни делови од хардверот кои можат да работат паралелно
- Пример:
  - Фаза 1 – **преземање** на инструкцијата **(fetch)** од меморијата и сместување во **бафер (buffer)**
  - Фаза 2 – **декодирање** на инструкцијата **(decode)**– определување на типот на инструкцијата и на операндите што и се потребни
  - Фаза 3 – лоцирање и **преземање на операндите (operand fetch)**, или од регистрите, или од меморијата
  - Фаза 4 – **извршување** на инструкцијата **(execute)**– спроведување на операндите низ податочната патека
  - Фаза 5 – **запишување на резултатот (write back)** во соодветниот регистер

2.1.4.1 Протечна обработка (pipelining)



(a)



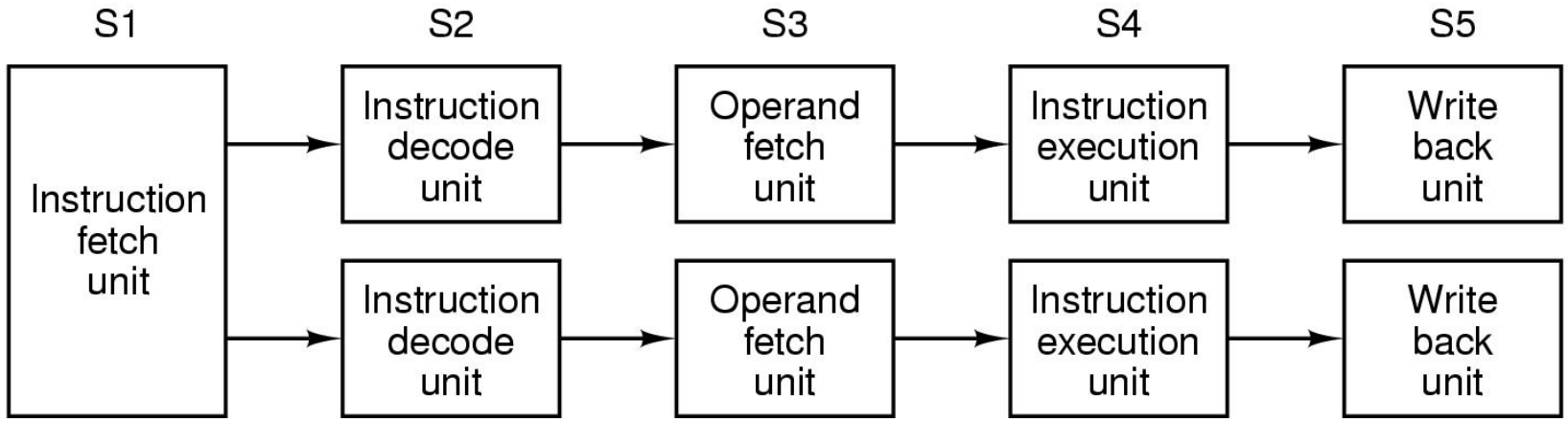
(b)

2.1.4.1 Протечна обработка (pipelining)

- Должина на циклусот на машината – **T [ns]**
- Број на фази во протечната обработка – **n**
- Време на извршување на една инструкција (**latency**) – **n·T [ns]**
- Пропусна моќ на процесорот (**processor bandwidth**) – **1000/T [MIPS]**
  - MIPS – Millions of Instructions Per Second
- Пример:
  - T = 2 ns (= 1 / 500 MHz)
  - n = 5
  - Latency = 10 ns
  - Processor bandwidth = 500 MIPS

2.1.4.2 Суперскаларни архитектури

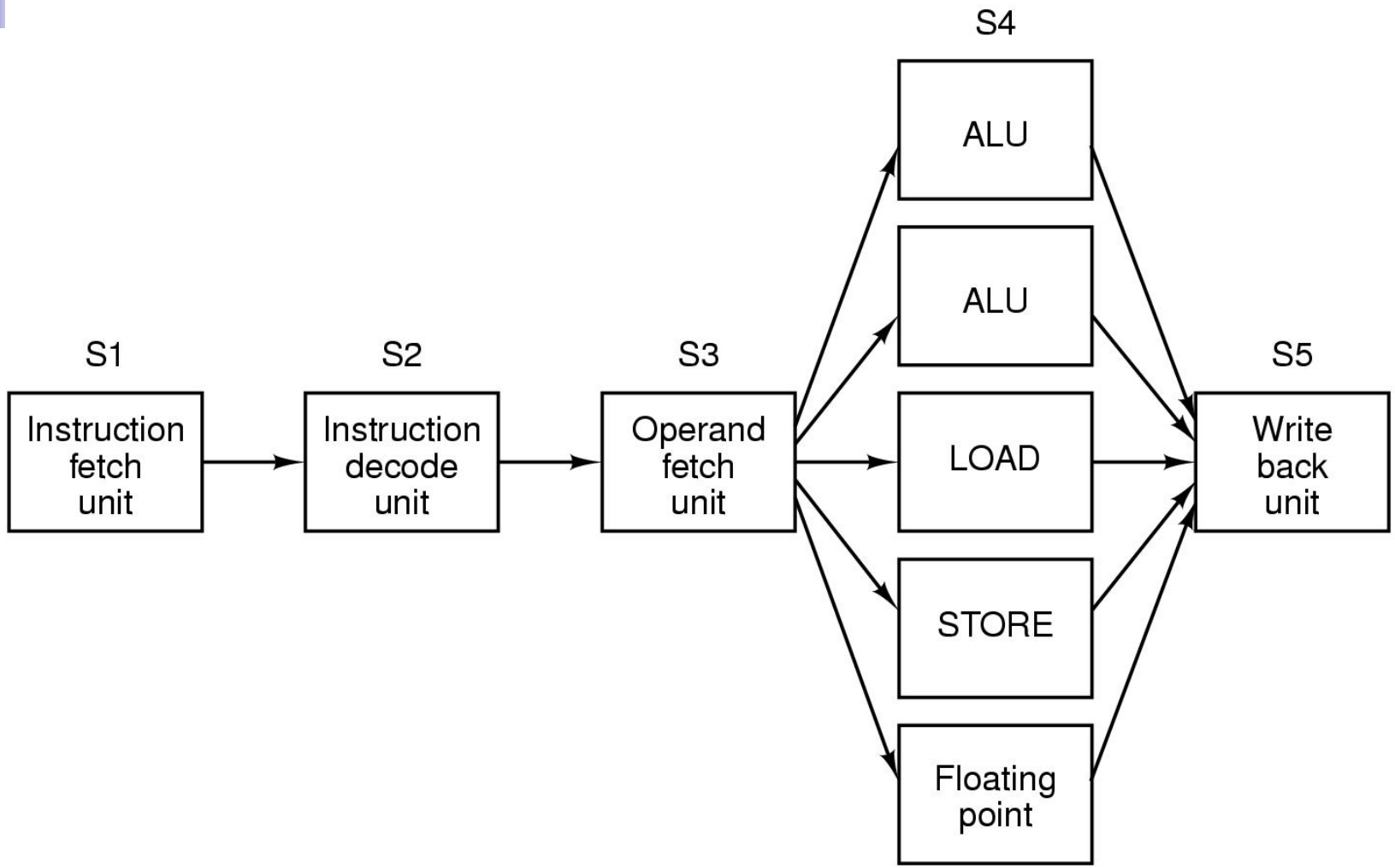
- Една единствена единица за преземање на инструкции може да презема неколку (на пр., две) инструкции одеднаш и секоја од нив да ја упатува кон посебни единици за декодирање и извршување (посебни „цевководи“)
- За да можат да се извршуваат паралелно, инструкциите не смеат да бидат во конфликт околу употребата на ресурси (на пр., регистри), ниту пак некоја од нив смее да зависи од резултатот на друга инструкција



2.1.4.2 Суперскаларни архитектури

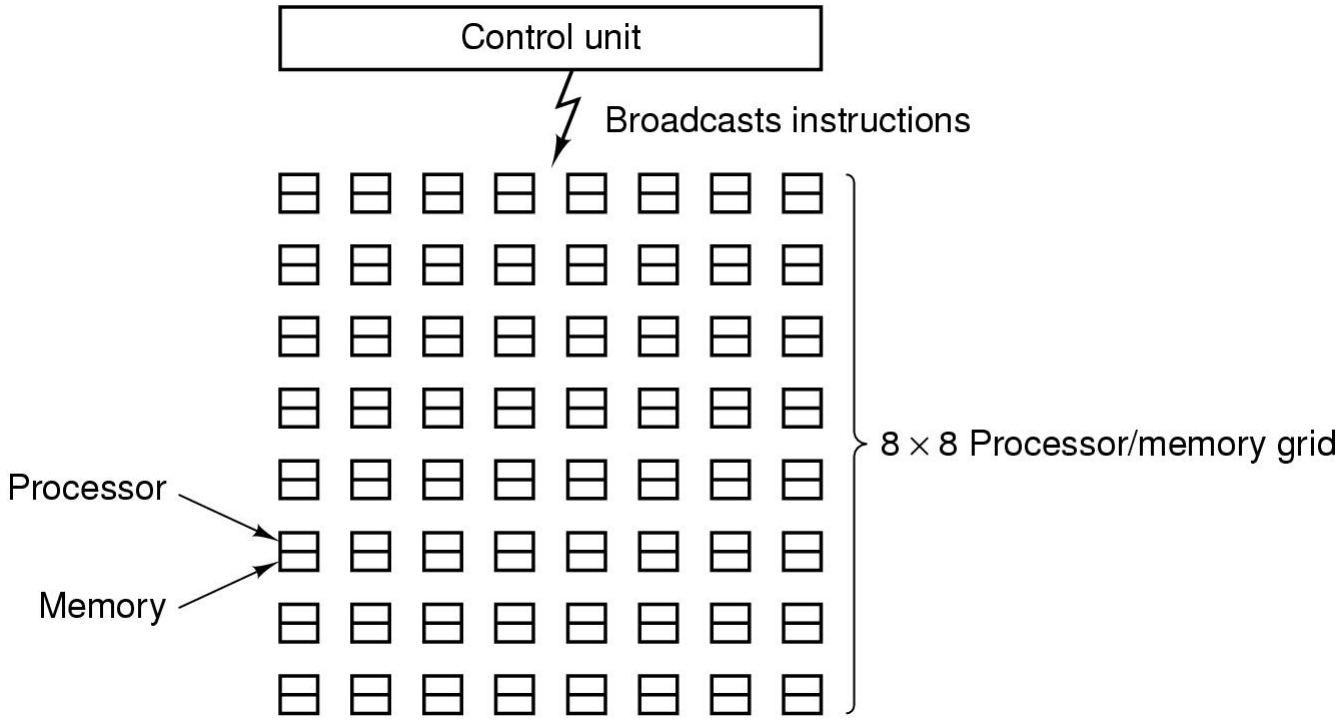
- Заради намалување на комплексноста на хардверот, кај современите процесори се применува малку поинаков пристап – еден единствен „цевковод“, но со повеќе извршни единици
- Се претпоставува дека на извршните единици можат да им се доставуваат инструкции побрзо отколку што тие се во можност да ги извршуваат
- Терминот **суперскаларна архитектура** датира од 1987 година (Agerwala & Cocke), иако влече корени уште од пред 40-тина години (од појавата на CDC 6600, со 10 извршни единици)

2.1.4.2 Суперскаларни архитектури



2.1.5 Паралелизам на процесорско ниво

- **Процесорско поле (Array Processor)** – голем број идентични процесори изведуваат **иста** секвенца од инструкции над **различни** множества податоци (ILLIAC IV, University of Illinois, 1972)
  - **SIMD – Single Instruction-stream Multiple Data-stream**



2.1.5 Паралелизам на процесорско ниво

- **Векторски процесор** – за програмерот, наликува на процесорско поле (Cray-1, Cray Research, 1974)
  - Основна разлика:
    - **Процесорско поле** – за реализација на операциите на собирање, постојат онолку единици за собирање колку што има парови од податоци
    - **Векторски процесор** – сите операции на собирање на паровите од податоци ги изведува една единствена високо-проточна (deeply-pipelined) единица за собирање
      - Техниките на векторска обработка се среќаваат кај конзолите за видео-игри и кај графичките акцелератори

2.1.5 Паралелизам на процесорско ниво

Собирање на две групи од по 10 броеви

...кај векторски процесор

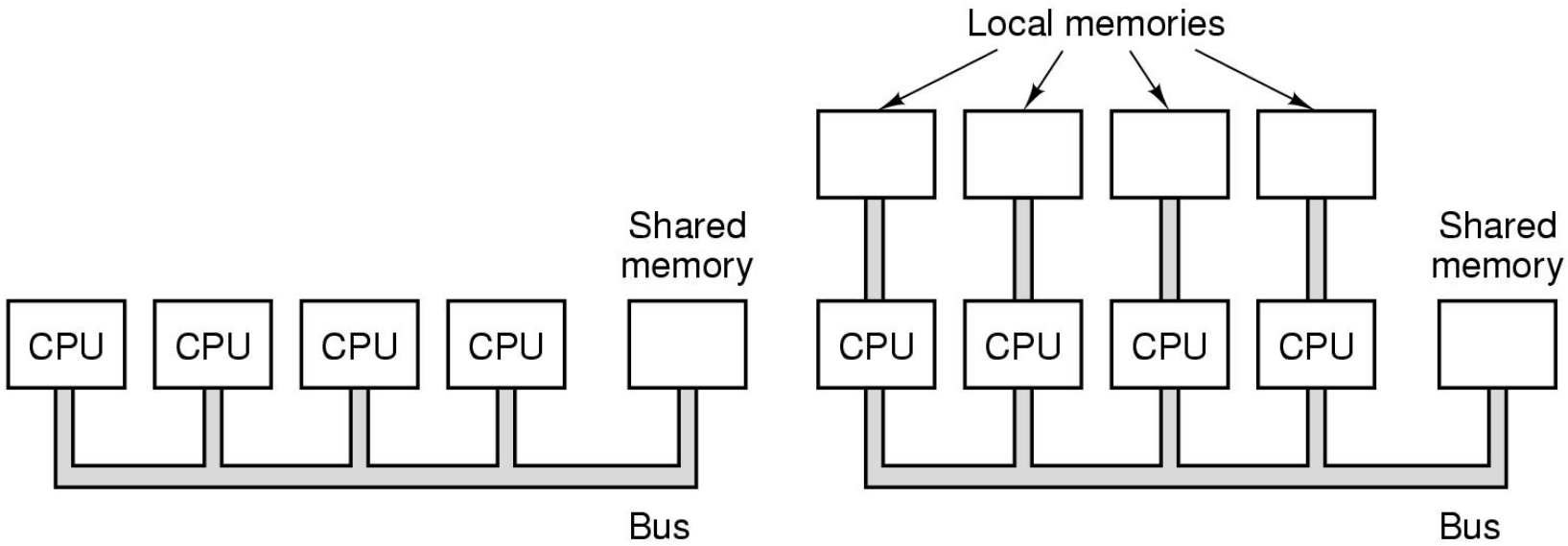
execute this loop 10 times
- read the next instruction and decode it
- fetch this number
- fetch that number
- add them
- put the result here

end loop

- read instruction and decode it
- fetch these 10 numbers
- fetch those 10 numbers
- add them
- put the results here

2.1.5 Паралелизам на процесорско ниво

- **Мултипроцесор** – систем од повеќе процесори (секој со своја контролна единица) со заедничка меморија (shared memory)



2.1.5 Паралелизам на процесорско ниво

- **Мултикомпјутер** – систем од голем број меѓусебно поврзани компјутери, секој со своја сопствена меморија
- Не постои заедничка меморија, со цел да се избегне потребата од поврзување на сите процесори со заедничката меморија, што е мошне тешко кога бројот на процесори е релативно голем (>64)
- Процесорите комуницираат со меѓусебно **проследување на пораки (message passing)** (нешто како e-mail, но многу побрзо!)
- Познати се примери на мултикомпјутери со приближно 10,000 CPU