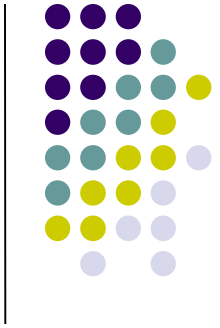


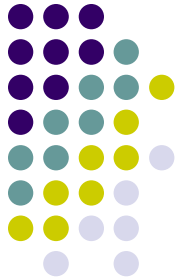
Увод

Рачунарски систем

Рачунарски систем

NIT

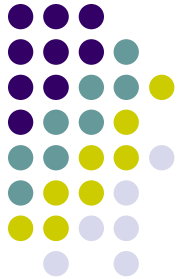
Рачунарски систем



- Хардвер - софтвер
- Физичка архитектура - програмска подршка
- Изучавају се често одвојено, али у рачунарској техници су тесно повезани и док се бави једним увек се мора мислити на друго.
- Пошто развој физичке архитектуре често претходи развоју програмске подршке, развој програмске подршке више зависи од физичке архитектуре него обрнуто.

NIT

Рачунарски систем

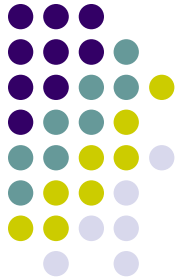


- Кад расте величина рачунара, расте и цена
- Кад расте величина рачунара, расте и потрошња

- Рачунарски систем треба ради **што боље**, да буде што јефтинији и да што мање троши.

NIT

Рачунарски систем



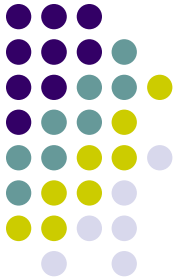
- Кад расте величина рачунара, расте и цена
- Кад расте величина рачунара, расте и потрошња

- Рачунарски систем треба ради **довољно добро**, да буде што јефтинији и да што мање троши.

- **Шта** треба да ради довољно добро?
  - Да довољно добро (брзо) извршава очекиване програме

NIT

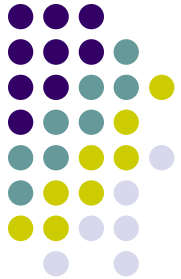
Рачунарски системи



- Системи за рад у реалном времену
- Уграђени системи
- Системи са ограниченим ресурсима -> Процесори са ограниченим ресурсима
- **Наменски рачунарски системи** -> Наменски процесор

NIT

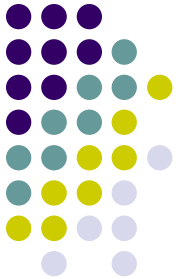
Системи у реалном времену



- Системи (за рад) у реалном времену.

- Приликом пројектовања готово је немогуће направити оштру разлику између „шта“ и „како“.

NIT

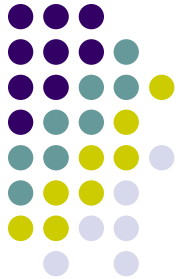


Уграђени системи

- Нема спреге човека и машине (или је сведена на минимум)
  - Дакле, нема: тастатуре, екрана итд.

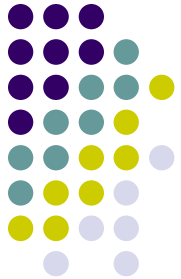
- 99% процесора је у уграђеним системима
- Животни циклус система: развој, производња, одржавање.
- Код уграђених система цена развоја је обично испод 5% укупне цене целог циклуса.
- Економски аспекти имају далеко већи значај.

Системи са ограниченим ресурсима



- Ресурси:
  - Меморија, регистри
  - **Брзина**
  - Енергија
  - ...
- Ограниченост је релативна
- али радни такт уграђених система је обично до неколико стотина мега-херца

NIT

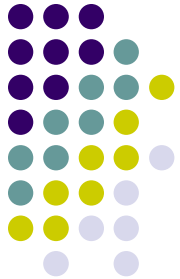


Наменски системи

- Системи посебне намене

- Наменски процесори
  - Сваки процесор сабира бројеве... ?

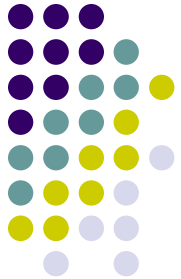
Груписање процесора по намени



- Процесори опште намене
- Наменски процесори (Процесори посебне намене)
  - ДСП-ови
  - Микроконтролери
  - Графички процесори...

NIT

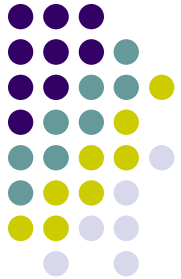
Процесори опште намене



- Најбоље извршавају све могуће врсте програма

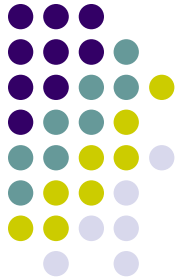
NIT

Процесори опште намене



- Најбоље извршавају све могуће врсте програма
- Подједнако добро извршавају све могуће врсте програма

NIT

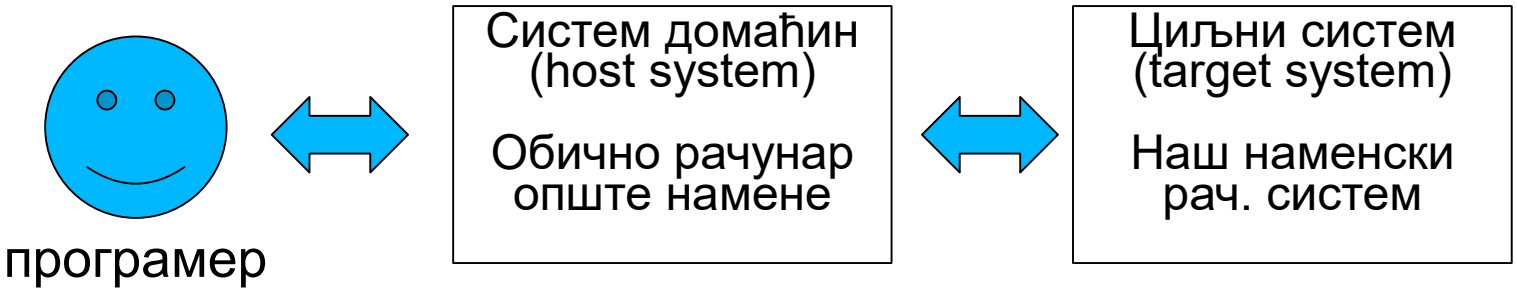


Процесори опште намене

- Најбоље извршавају све могуће врсте програма
- Подједнако добро извршавају све могуће врсте програма
- Неке врсте програма извршавају боље, неке лошије, али је разлика обична мања од једног реда величине (10x)

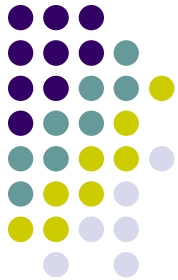
Чиме се програмирају наменски рачунарски системи

- Рад са HPC најчешће није директан.



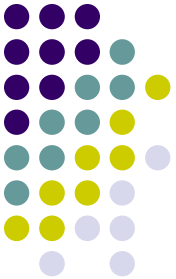
- Део системског софтвера се извршава на домаћину.
- Домаћин је систем са више ресурса и већом брзином, а пре свега комотнији за рад.
- Многи елементи системског софтвера ни не би могли бити извршавани на циљном систему.

Чиме се програмирају наменски рачунарски системи



- Алати:
  - Асемблер
  - Компајлер (за виши програмски језик)
  - Дебагер (компонента за контролисано извршавање програма)
  - Симулатор (омогућава бољу контролу и бољи увид у извршавање кода)
  - Интегрисано развојно окружење
- Оперативни систем и библиотеке

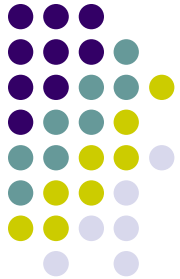
Када се програмирају наменски рачунарски системи



- Пре него што је физичка архитектура готова
  - Да би се убрзао развој система
  - Део кода је већ раније развијен
- Када је физичка архитектура фиксирана, али још увек није произведена
  - Развојне плоче са прототипом
  - Симулатори
- Када је физичка архитектура готова, али системски софтвер још није (алати)
  - Развој системског софтвера обично креће пре него што је физичка архитектура готова
- Када је физичка архитектура спремна и зрела
  - Код неких система ово је већ крај животног циклуса софтвера
  - Код неких дугих - нови почетак

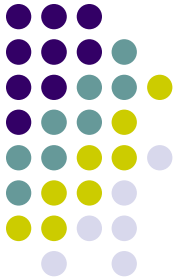
NIT

Како се програмирају наменски рачунарски системи



- Зависи од много фактора:
  - Фазе у развоју - спремност физичке архитектуре и алата
  - Уложеног развојног напора - у развој системског софтвера, алата пре свега
  - Обима конкретног програма - за мали програм може и бинарно да се програмира
  - Захтевности конкретног програма - на нижем нивоу апстракције се може више исцедити из архитектуре

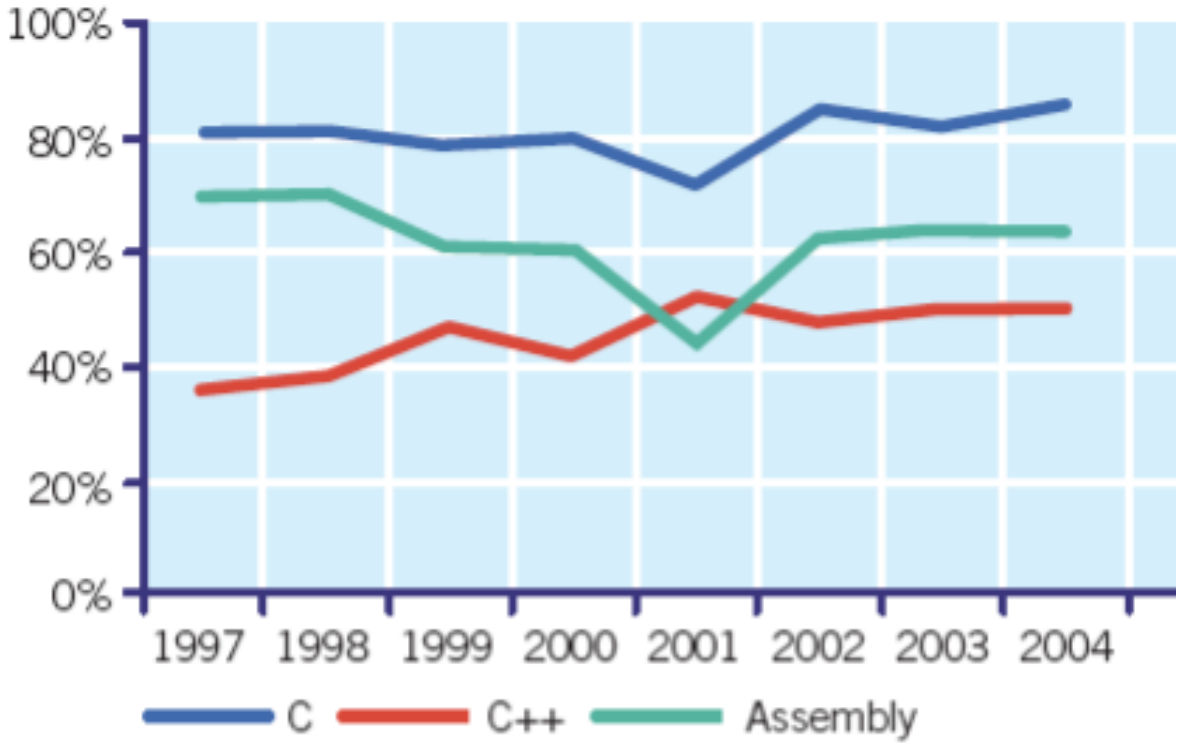
Како се програмирају наменски рачунарски системи



- Асемблер
  - Ближи архитектури
  - Преводилац лакши за прављење
  - Компликованији за успешно програмирање
- Виши програмски језик
  - Даљи од архитектуре
  - Преводилац компликованији
  - Програмирање лакше

Прогамски језик Це

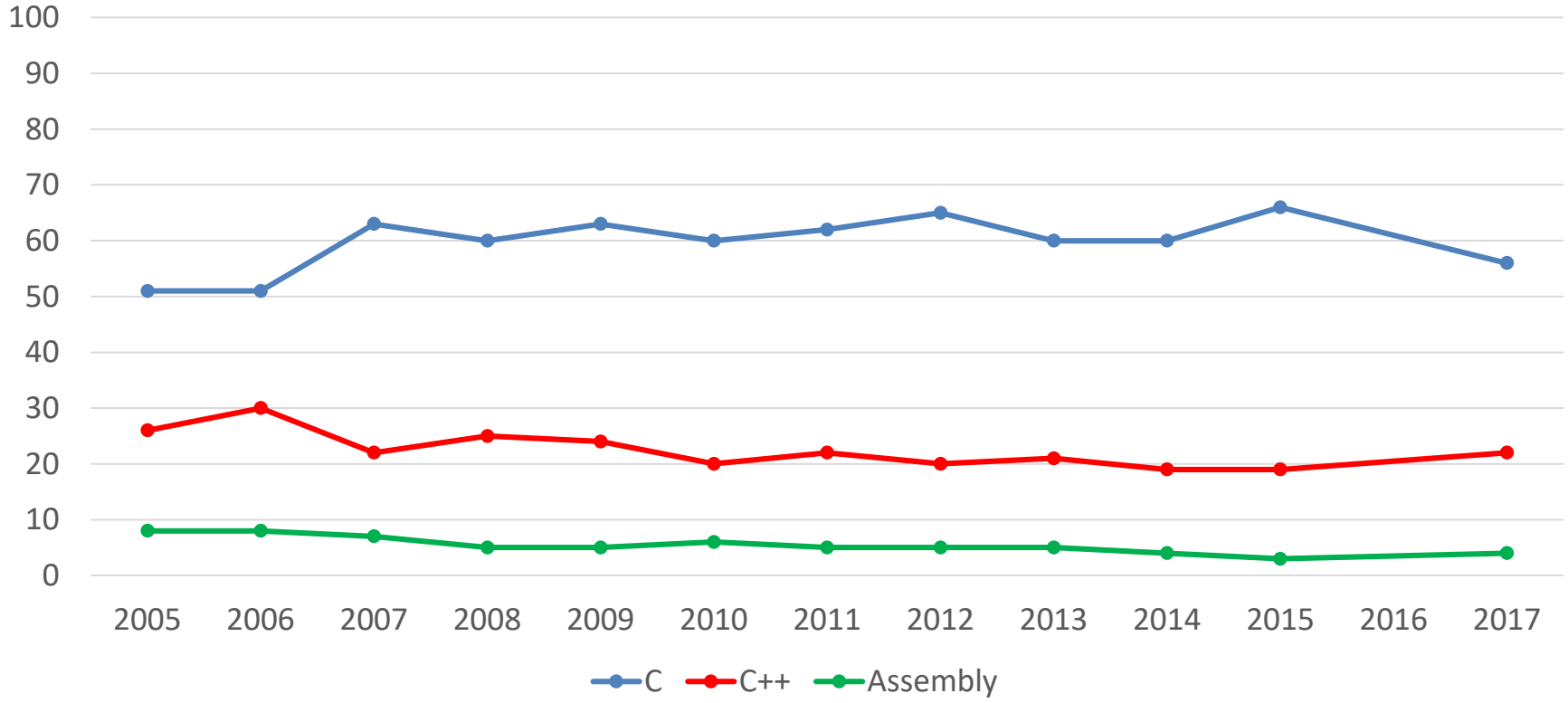
- Виши програмски језик који се најчешће користи за програмирање HPC је Це.



Проценат пројеката у којима се користи дати програмски језик

Прогамски језик Це

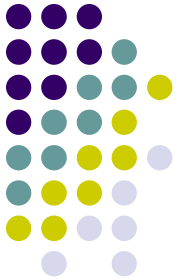
- Виши програмски језик који се најчешће користи за програмирање HPC је Це.



Проценат пројеката у којима се **већински** користи дати програмски језик

NIT

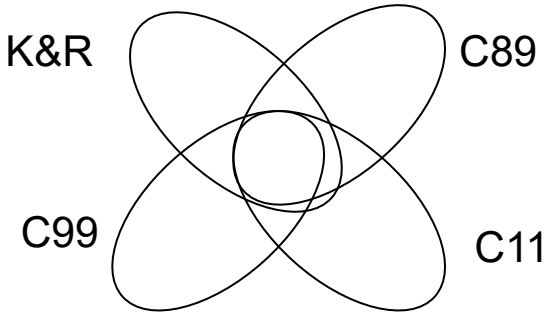
Прогамски језик Це

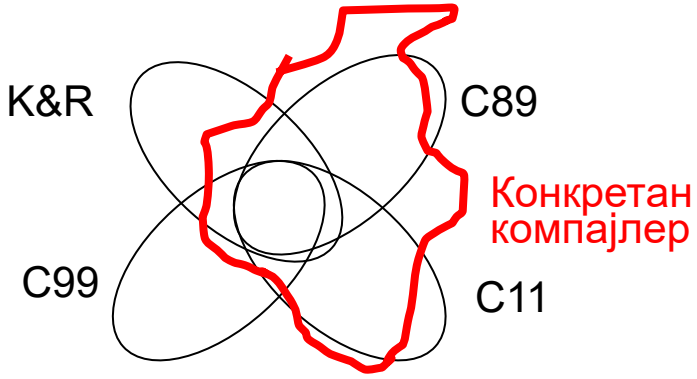


- Зашто програмски језик Це?
  - Није превише висок
    - Омогућује боље цеђење перформанси
    - Релативно добро пресликавање његових операција на операције физичке архитектуре
  - Историјски разлози
    - Нагомилан већ постојећи код
    - Велики број његових познаваоца
    - Конструкција компајлера релативно једноставнија

Прогамски језик Це

- Зашто **НЕ** програмски језик Це?
  - Није превише висок
    - Виши ново апстракције би олакшавао програмирање
    - Релативно **лоше** пресликавање његових операција на операције физичке архитектуре
  - Историјски разлози
    - Ваљда је до сада могло бити смишљено нешто паметније
    - Велики број његових „познаваоца”
    - Који стандард тачно подржава конкретан систем?



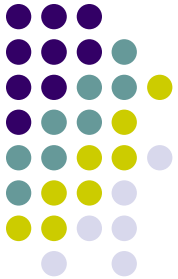


NIT

Прогамски језик Це++

- Цеов млађи брат
  - Нису потпуно у релацији подскуп/надскуп
    - Али 99% Цеа је исто и у Це++-у.
  - Хардверски модел је идентичан
  - Скоро све што будемо учили на овом курсу важи и у Це++-у.
    - Скренућемо пажњу на места где постоје разлике
  - Зашто Це++ није потиснуо Це?
    - За већину проблема из овог домена изгледа да је Це сасвим довољан.
    - Инерција.
    - Више могућности => више шанси да се погреши.
    - Више могућности => више мора да се учи!

NIT



Циљ курса

Оспособити полазнике да могу:

- Писати нови Це код за HPC
  - Научити делове језика који нису у фокусу када се програмирају системи опште намене и када се ад хок програмира, али су важни при програмирању HPC
  - Учврстити добру праксу, зарад писања лепог, јасног и употребљивог кода
  - Препознати сиве зоне језика да их не бисте користили
- Разумети постојећи Це код за HPC
  - Омогућити дубинско разумевање значења кода
  - Препознати сиве зоне језика да бисте их разумели у коду са којим се будете сусретали