Увод у организацију и архитектуру рачунара 2

Александар Картељ

kartelj@matf.bg.ac.rs

Напомена: садржај ових слајдова је преузет од проф. Саше Малкова

Кеш меморија

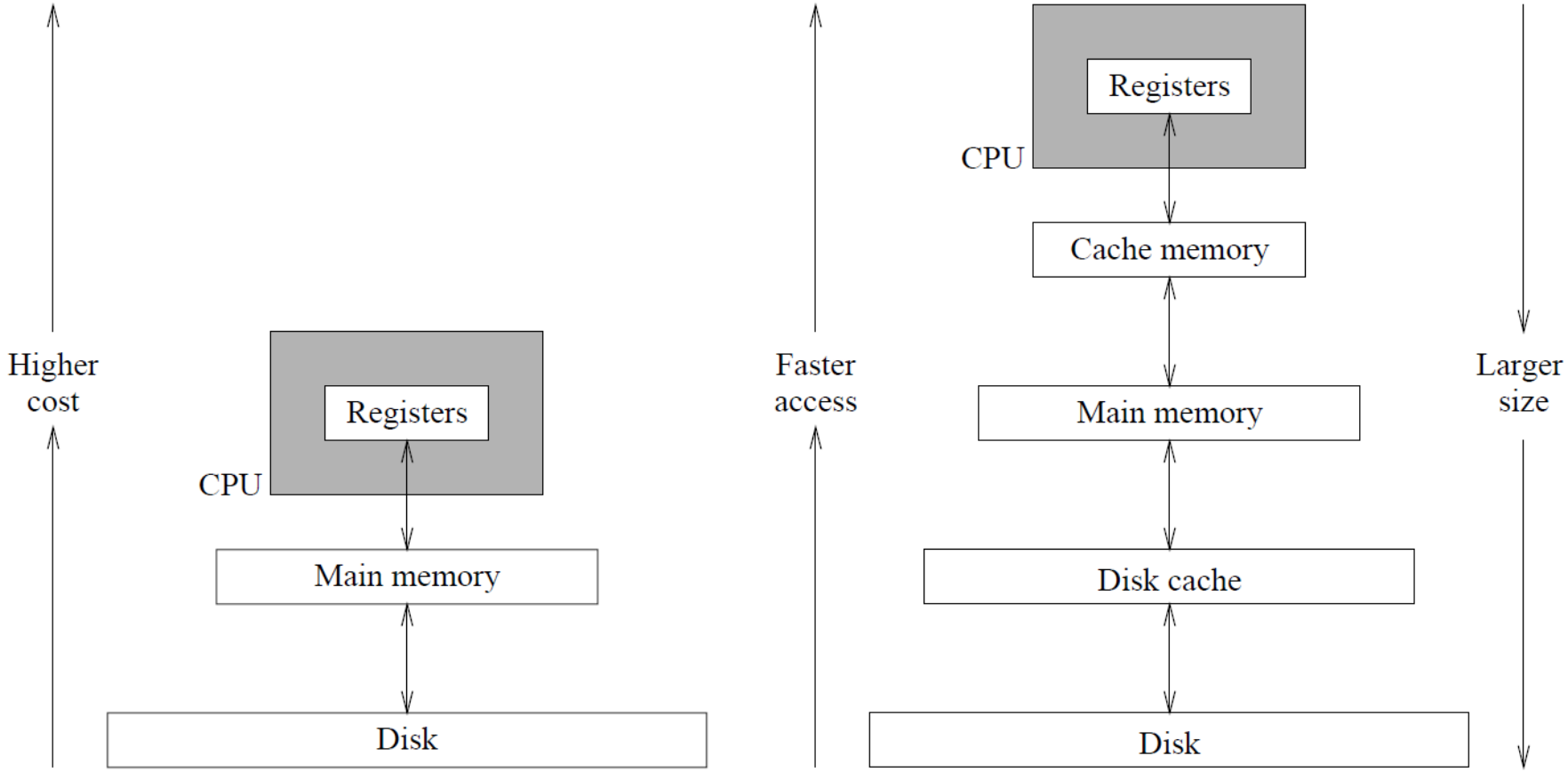
Увод

Намена кеша

- Процесор садржи регистре, као најефикаснију меморију у систему
- Радна меморија рачунара је релативно велика и спора
- Разлика у брзини ова два слоја је довољно велика
  да може да значајно ослаби перформансе система

- Кеш се додаје као међуслој између регистара (тј. процесора) и радне меморије
  - брзина између регистара и радне меморије
  - величина између регистара и радне меморије
- Ако је разлика превелика, додаје се више слојева кеш меморије

Хијерархија меморија



Примери брзина и величина кеш меморија

CPU Type	Pentium	Pentium Pro	Pentium II	Pentium III	Pentium 4
CPU speed	233MHz	200MHz	450MHz	1.4GHz	3.6GHz
L1 cache speed	4.3ns (233MHz)	5.0ns (200MHz)	2.2ns (450MHz)	0.71ns (1.4GHz)	0.28ns (3.6GHz)
L1 cache size	16KiB	32KiB	32KiB	32KiB	20KiB
L2 cache type	onboard	on-chip	on-chip	on-die	on-die
CPU/L2 speed ratio		1/1	1/2	1/1	1/1
L2 cache speed	15ns (66MHz)	5ns (200MHz)	4.4ns (225MHz)	0.71ns (1.4GHz)	0.28ns (3.6GHz)
L2 cache size	varies	256K	512K	512K	1M
CPU bus speed	66MHz	66MHz	100MHz	133MHz	800MHz
Memory bus speed	60ns (16MHz)	60ns (16MHz)	10ns (100MHz)	7.5ns (133MHz)	1.25ns (800MHz)

Принцип рада кеша

- Основни принцип рада је захватање података из радне меморије унапред (*prefetch*), пре него што заиста затребају процесору
  - ако је успешно предвиђено који ће подаци бити потребни процесору у блиској будућности, њих ће процесор читати из кеша а не из меморије
  - тиме се значајно подижу перформансе система

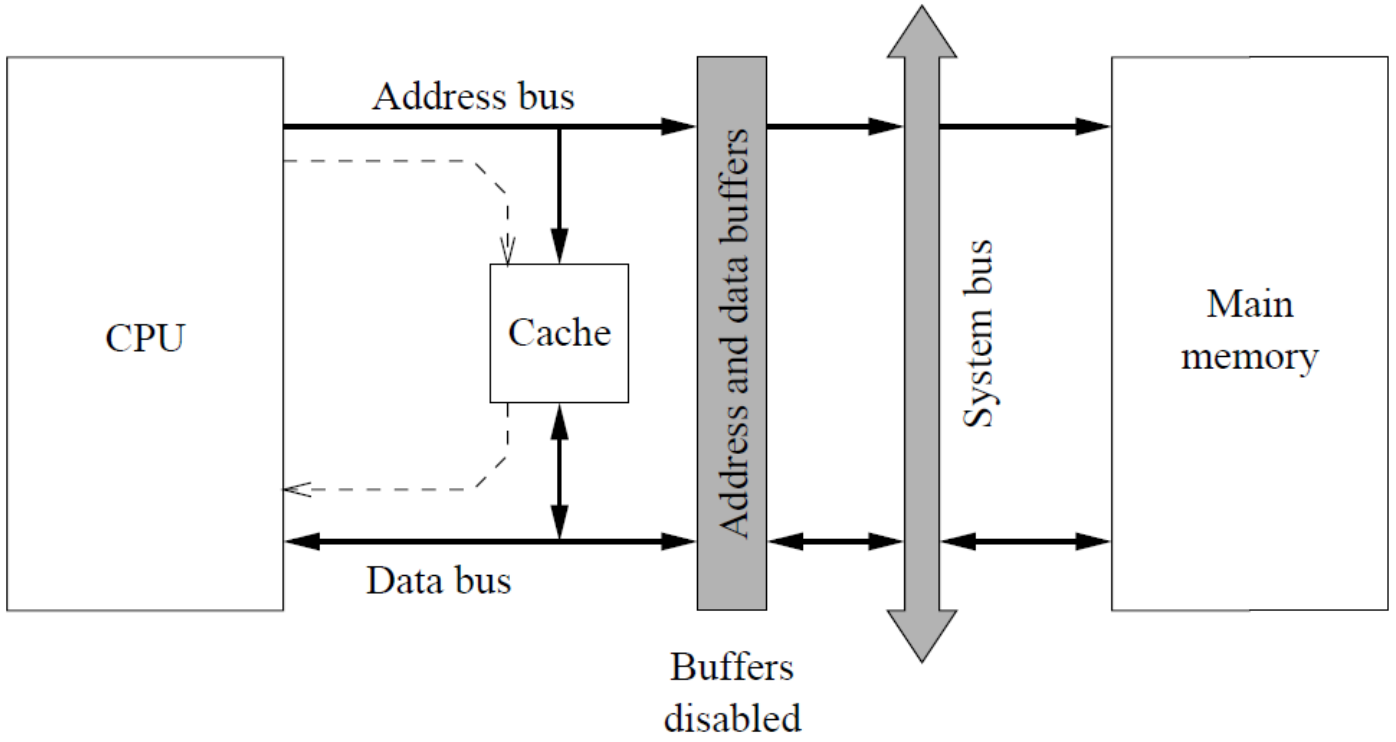
Основне операције кеша

- Кеш се користи у случају две основне меморијске операције
  - читање и
  - писање

- У оба случаја постоје по две варијанте
  - када су подаци присутни у кешу
    - тзв. *погодак* (*hit*)
  - када подаци нису присутни у кешу
    - тзв. *промашај* (*miss*)

Читање у случају поготка

- Ако се потребни подаци налазе у кешу, онда се одатле и читају



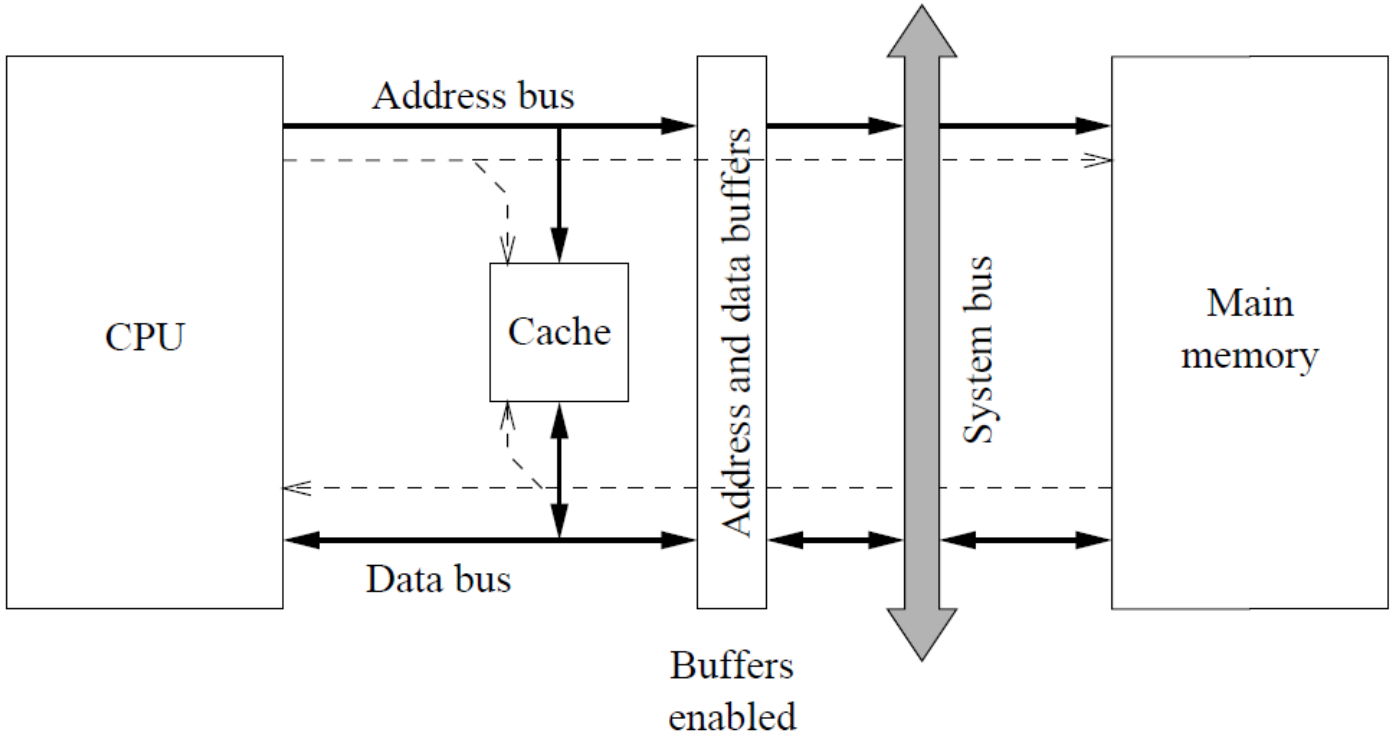
Читање у случају поготка (2)

- У случају поготка, линије адресе и података према меморији се блокирају
- Размена информација се одвија искључиво са кешом

- Читање са поготком је значајно брже од читања из меморије без примене кеша

Читање у случају промашаја

- Ако се потребни подаци не налазе у кешу, онда се читају из меморије и истовремено уписују у кеш



Читање у случају промашаја (2)

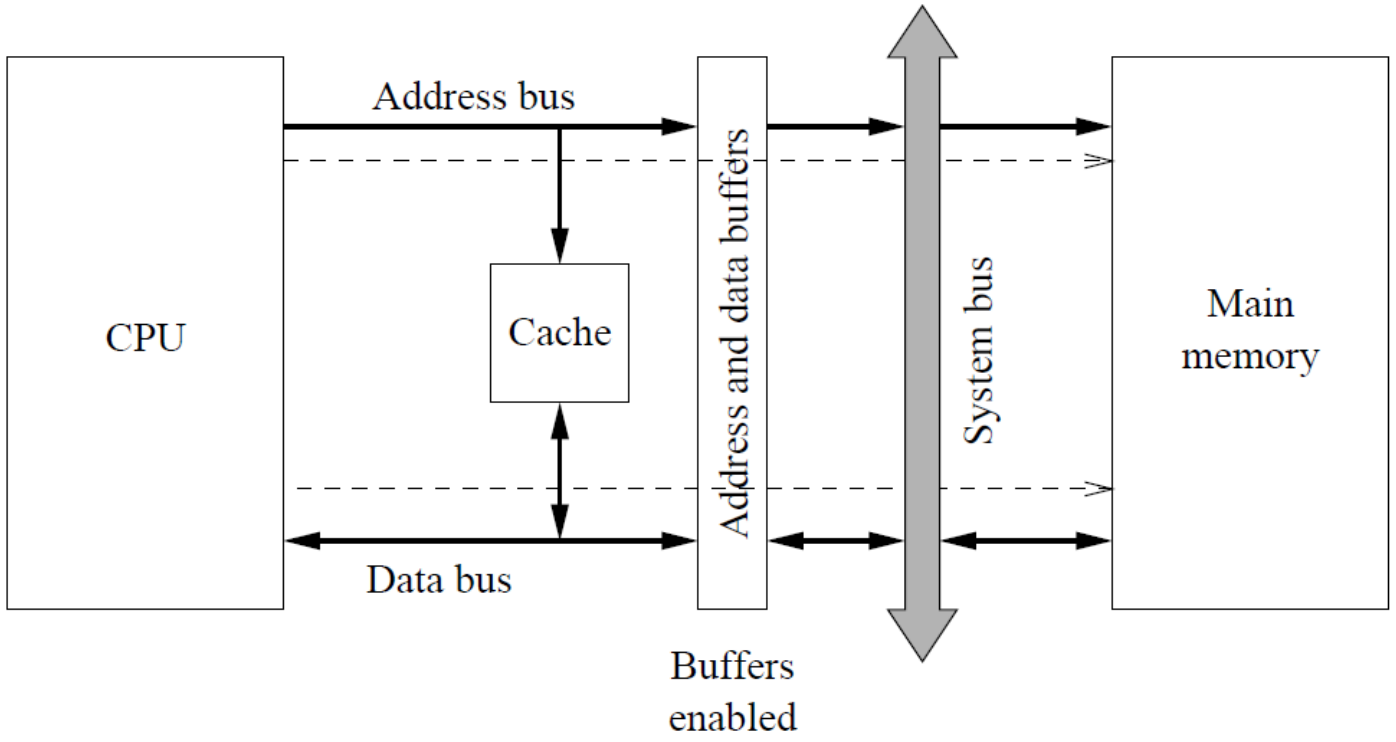
- У случају промашаја, линије адресе и података према меморији су активне
- Одвија се уобичајено (као да нема кеша) читање података из меморије
  - додатно се прочитани подаци уписују и у кеш
- Читање са промашајем је нешто спорије од читања из меморије без примене кеша
  - због неопходног проверавања да ли податак постоји у кешу или не

Перформансе кеша

- Поред брзине рада кеш меморије постоје и додатне мере перформанси
  - ***степен погодака*** (*hit rate*, *hit ratio*) је коефицијент који показује колико често се тражени подаци проналазе у кешу
  - ***степен промашаја*** (*miss rate, miss ratio*) показује колико често се тражени подаци не проналазе у кешу
    - (степен погодака + степен промашаја) = 1
  - ***време поготка*** (*hit time*) је време потребно да се податак прочита ако је у кешу
    - обухвата и време потребно да се провери да ли је податак у кешу
  - ***цена промашаја*** (*miss penalty*) је време потребно да се установи да податак није у кешу и да се читање преусмери на меморију

Писање у случају промашаја

- У случају промашаја подаци се уписују само у меморију, зато што не постоје у кешу

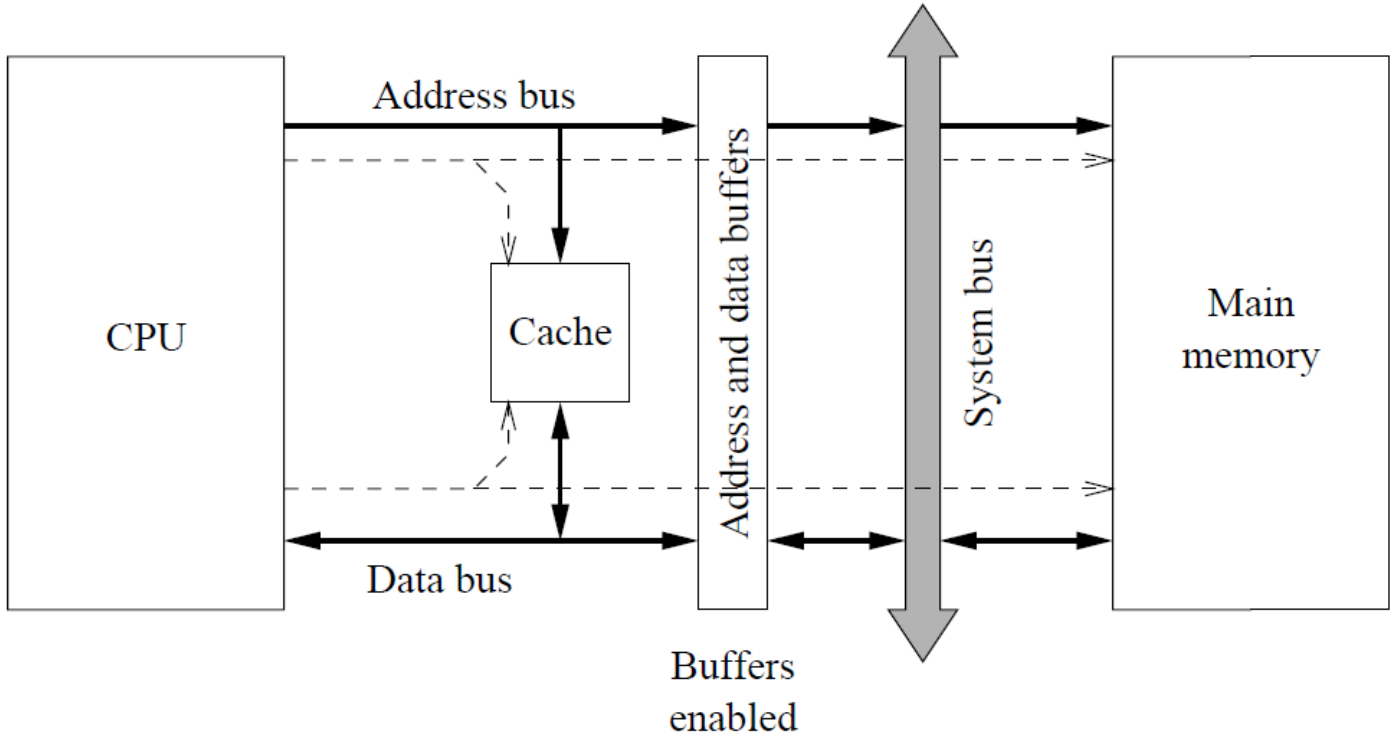


Писање у случају поготка

- У случају поготка постоје две основне могућности:
  - писање се обавља само у кеш или
  - писање се обавља и у кеш и у меморију

Писање у случају поготка (2)

- Случај писања и у меморију и у кеш



Значајна питања

- Како се установљава да ли је података у кешу или није?
- Ако податак није у кешу, како се у њега уписује?
- Колико података се уписује у једној операцији?
- Шта се дешава ако нема више места?
- Како се претпоставља чему ће процесор приступати у блиској будућности?

Зашто кеш ради?

- Практичан пример: увећавање свих елемената матрице (нпр. *double*) за *K*:

```
for(int i=0; i<M; i++)
   for(int j=0; j<N; j++)
      X[i][j] = X[i][j] + K;
```

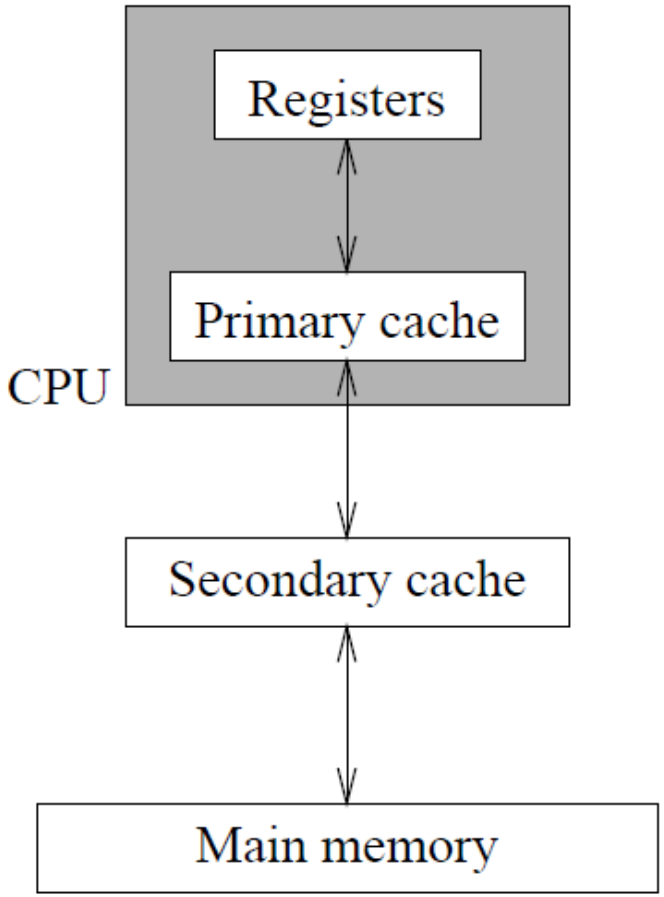
Зашто кеш ради? (2)

- Први фактор за успешну примену кеша је *поновљена употреба* истих података или делова кода
  - Наредба у петљи се понавља *M*x*N* пута
  - Ако је наредба записана у кешу, њено извршавање је значајно убрзано зато што се чита из брзог кеша а не из споре меморије

Зашто кеш ради? (3)

- Други фактор је планско пуњење кеша подацима и инструкцијама пре него што их процесор затражи
  - Планско пуњење подиже перформансе из два разлога:
    - маскира кашњење спорије главне меморије
    - пуњење се одвија у блоковима, а пренос блокова података је вишеструко бржи него пренос појединачних података

Више нивоа кеша



Више нивоа кеша (2)

- Увођењем више нивоа кеша (*вишестепени кеш*) омогућава се
  - физичко лоцирање кеша на различитим местима
    - L1 кеш (кеш нивоа 1) на процесору, ближе регистрима
    - L2 кеш (кеш нивоа 2) ближе меморији
  - обезбеђивањем L2 кеша који је већи и са већим линијама смањује се цена промашаја кеша L1

Перформансе вишестепеног кеша

- Намена кеша L2 је да “ухвати” промашаје кеша L1
  - Ако је степен погодака кеша L1 90% и степен погодака кеша L2 такође 90%, онда је укупан степен промашаја (заступљеност приступа који морају да се обрате главној меморији) свега 1%

    (инклузиван случај)
    (идеализовано, за случај различитих величина линија)

Пример савремене архитектуре

- Процесор *Intel Core i7 (Ivy Bridge)*
  - Свако језгро процесора има
    - кеш нивоа L1, одвојено за инструкције и податке, сваки по:
      - 32 *KiB*, 512 линија по 64B
      - 8-скуп-асоцијативан
      - 4 циклуса по инструкцији
    - кеш нивоа L2
      - 256*KiB*, 512 линија по 512B
      - 8-скуп-асоцијативан
      - 10 циклуса по инструкцији
  - Процесор садржи и дељени кеш L3
    - 2-15*MiB*, величина линије 512B
    - 16-скуп-асоцијативан
    - инклузиван
    - брзина зависи од варијанте процесора

Пример савремене архитектуре
Процесор *Intel Core i7*