[P220]

Увод у архитектуру рачунара

14

Саша Малков
Универзитет у Београду
Математички факултет
2012/2013

[P271]

Увод у архитектуру рачунара

Саша Малков

Тема 16

Суперскаларни и векторски процесори

(наставак)

Унапређење перформанси

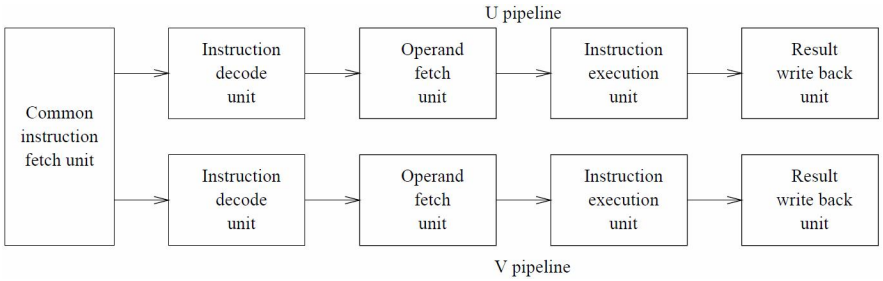
- Основни методи:
  - суперскаларни процесори
  - суперпреклапајући системи
  - архитектуре са веома дугим записом инструкције

Суперскаларни процесори

- Суперскаларни процесори су они који имају вишеструке јединице за извођење неке или свих фаза извршавања инструкција
- Може се поновити цео или скоро цео ток извршавања инструкција

Дуални ток извршавања

- Архитектура са два тока извршавања
  - јединица за читање инструкција чита по две инструкције у циклусу
  - ставља по једну у сваки ток извршавања
  - даље се извршавају свака на свом хардверу
  - наравно, мора се водити рачуна о међузависности



Универзитет у Београду - Математички факултет

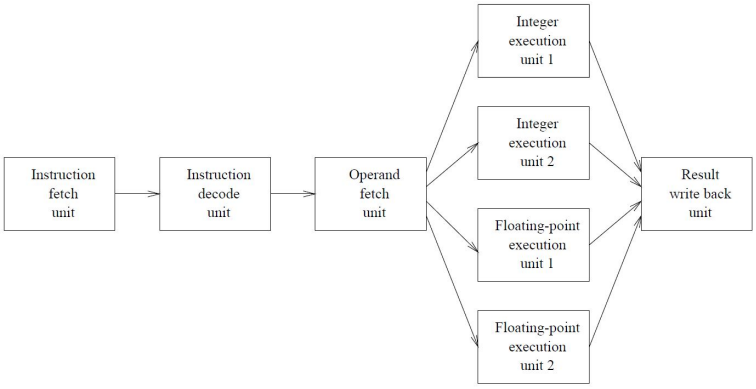
Понављање "спорих" јединица

- У пракси немају све фазе исто трајање
  - обично јединица за извршавање захтева више времена
- Пут за подизање перформанси је да се обезбеди више примерака оних јединица које имају дуже извршавање
  - обично су то јединице за извршавање инструкције

Универзитет у Београду - Математички факултет

Понављање "спорих" јединица

- Ток за извршавање са четири јединице за извршавање инструкције



Универзитет у Београду - Математички факултет

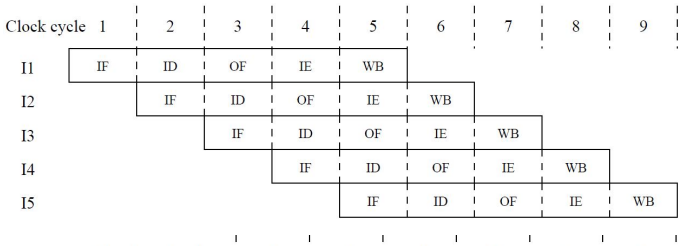
Суперпреклапајући процесори

- Процесор је *суперпреклапајући* ако се ток извршавања (преклапања) продужава тако што се сваки корак дели на поткораке
  - сваки циклус се дели на потциклусе
- Добија се на перформанса због тога што се постиже гушће паковање инструкција у току
- Суперпреклапајући процесор може да извршава више од једне инструкције по циклусу

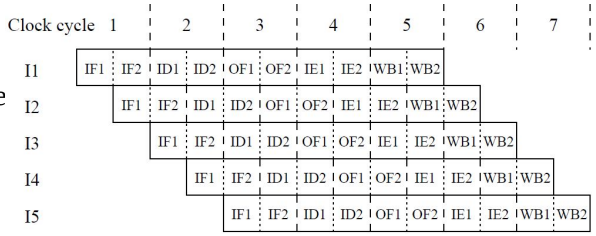
Универзитет у Београду - Математички факултет

Суперпреклапајући процесори

- уобичајено преклапање



- суперпреклапање



Универзитет у Београду - Математички факултет

Процесор *MIPS R4000*

- Процесор *MIPS R4000* има суперпреклапање са следећим фазама:
  - IF1 – читање инструкције, прва половина
  - IF2 – читање инструкције, друга половина
  - RF – декодирање инструкције и читање регистарских операнада
  - EX – извршавање инструкције
  - DF1 – читање података (*load*, *store*), прва половина
  - DF2 – читање података (*load*, *store*), друга половина
  - TC – провера читања података (*load*, *store*)
  - WB – писање резултата

Универзитет у Београду - Математички факултет

Процесор *MIPS R4000*

- читање инструкције и приступање меморији (читање података) су подељени у по два корака
- корак *EX* извршава инструкцију
  - у случају инструкција *load / store*, АЛЈ током корака *EX* рачуна адресу
  - у случају гранања, АЛЈ током корака *EX* одређује да ли се изводи гранање и рачуна циљну адресу
- корак *TC* проверава ознаке кеша

Универзитет у Београду - Математички факултет

Распоређивање инструкција

- Проблем *распоређивања инструкција* (енгл. *scheduling*) је у проналажењу најефикаснијег могућег редоследа извршавања инструкција на датом хардверу
- Није увек могуће оптимизовати изворни редослед
  - ако имамо једну јединицу за сабирање, не можемо извршавати две инструкције *add* истовремено
  - ако су неке инструкције међузависне, не можемо их прераспоређивати

Универзитет у Београду - Математички факултет

Пример

- Нека је дат код:

```
add     R1, R2, R3      ; R1 = R2 + R3
sub     R5, R6, R7      ; R5 = R6 – R7
and     R4, R1, R5      ; R4 = R1 AND R5
xor     R9, R9, R9      ; R9 = R9 XOR R9
```

- Дате инструкције се могу извршити у два корака:
  - 1. корак: add, sub, xor
  - 2. корак: and

Архитектуре *VLIW*

- Архитектуре са веома дугим записом инструкције
  - енгл. *Very Long Instruction Word Architectures – VLIW*
- Уместо да се распоређивање инструкција одвија у фази извршавања, идеја је да се одвија у фази превођења програма
  - избегава се одговарајућа хардервска имплементација
  - омогућава се груписање инструкција које се могу истовремено извршавати
  - више инструкција се спаја у једну дугачку инструкцију

Пример

- Систем *Multiflow TRACE*
  - инструкције су дужине 256 бита
  - свака се састоји од до 7 различитих операција
- напреднија верзија
  - инструкције су дужине 1024 бита
  - свака се састоји од до 28 различитих операција

Универзитет у Београду - Математички факултет

Архитектуре *VLIW* (2)

- Гранања неизоставно представљају проблем и код ових архитектура
- Варијанта решења је да процесор израчунава обе гране, до тренутка када се установи која је исправан исход
- Неки од концепата су примењени код процесора *Intel Itanium*
  - ширина магистрале је 128 бита
  - у једном циклусу се чита пакет од 3 инструкције кодиране са по 40 бита
  - преосталих 8 бита описује пакет

Универзитет у Београду - Математички факултет

Векторски процесори

- Векторски процесори су процесори који истовремено обрађују низове података
- За разлику од суперскаларних процесора, који повећавају ефикасност извршавањем више инструкција истовремено, векторски процесори извршавају исту инструкцију на више података
- У контексту векторксог процесирања, *вектор* је низ података фиксне дужуне
- Векторски процесори могу да раде и на матрицама, а не само на низовима

Поређење

- Нека су A и B два вектора исте дужине n (нпр. 64):
  $A = a_0, a_1, \ldots, a_{n-1}$
  $B = b_0, b_1, \ldots, b_{n-1}$
- Нека је потребно израчунати збир $C = A + B$
- Класичан приступ је употреба петље:

```
for(i=0; i<n; i++)
    C[i] = A[i] + B[i];
```

- Векторски приступ је да се то уради једном инструкцијом процесора

Архитектура

- Векторски процесори се састоје од
  - скаларне јединице
    - ради на скаларним подацима, слично скаларном процесору
  - векторске јединице
    - ради на векторским подацима
- Слично помаку са *CISC* на *RISC*, векторске архитектуре су прелазиле са архитектура меморија-меморија на вектор-меморија

Универзитет у Београду - Математички факултет

Архитектура (2)

- Архитектура меморија-меморија
  - прве векторске машине
  - све операције читају аргументе из меморије и записују резултат у меморији
- Архитектура вектор-меморија
  - већина новијих архитектура
  - све операције се одвијају над векторским регистрима
  - посебне инструкције су задужене за преписивање вектора података из меморије у регистре и обратно

Универзитет у Београду - Математички факултет

Приказ архитектуре

- Почива на процесору *Cray 1*
- Компоненте
  - векторски регистри
  - скаларни регистар
  - јединица за читање и писање (*load/store*)
  - векторска функционална јединица – ВФЈ (енгл. *VFU*)
  - меморијска јединица

Универзитет у Београду - Математички факултет

Регистри

- Векторски регистри
  - 8 векторских регистара
    - сваки регистар по 64 елемента од по 64 бита
    - сваки регистар има по два излазна и један улазни порт
  - Неки процесори имају програмибилне регистре
    - нпр. *Fujitsu VP 200:*
      - укупан регистарски простор од 8 K елемената
      - број регистара је био програмибилан, од 8 до 256
        - 8 регистара по 1024 64-битна елемента
        - 256 регистара по 32 64-битна елемента
- Скаларни регистар
  - садржи скаларни аргумент (нпр. множење скаларом)

Универзитет у Београду - Математички факултет

Јединица за читање и писање

- Служи за преписивање података између меморије и векторских регистара
  - операције (*load/store*)
- Ради са преклапањем
- Омогућава истовремено читање и писање
- Неки процесори имају више оваквих јединица

Универзитет у Београду - Математички факултет

Векторска функционална јединица

- (енгл. *Vector Functional Unit – VFU*)
- Векторски процесори имају више ВФЈ за операције са целим бројевима, бројевима у покретном зарезу и логичким подацима
  - *Cray* 1 је имао 6 јединица, као на дијаграму
  - *NEC SX/2* је имао 16 ВФЈ
    - 4 за целобројно сабирање и логичке операције
    - 4 за множење и дељење у покретном зарезу
    - 4 за сабирање и одузимање у покретном зарезу
    - 4 за померање
- За скаларне операције су коришћене посебне јединице, које овде не разматрамо

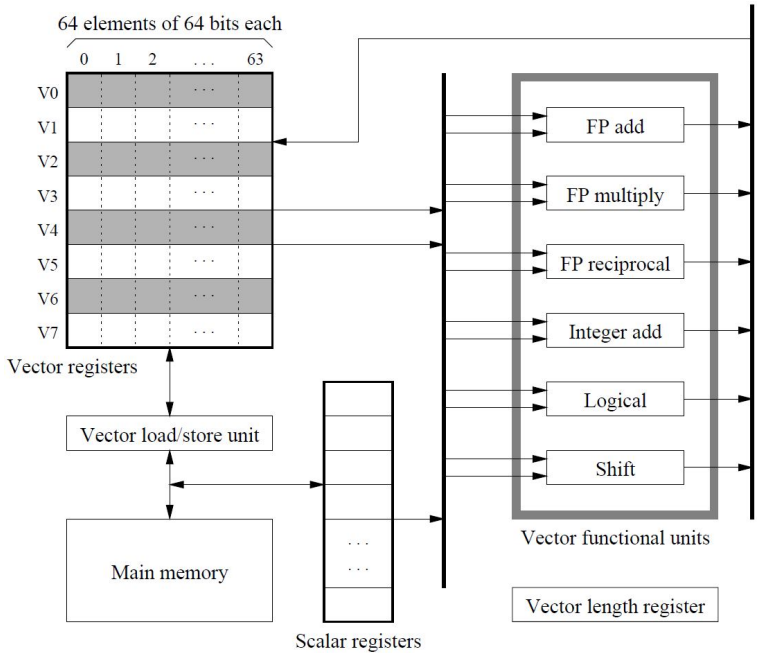
Универзитет у Београду - Математички факултет

Меморијска јединица

- Другачије организована него код скаларних процесора
- Уобичајено омогућава преклапања при преносу података у и из меморије
- Уобичајено се користи преплитање ради подизања перформанси

Универзитет у Београду - Математички факултет

Дијаграм архитектуре



Универзитет у Београду - Математички факултет

Особине векторских процесора

- Нису намењени за рачунаре опште намене
- Посебно прилагођени потребама научних и техничких израчунавања
- Предности
  - превазилажење *Флиновог уског грла*
  - елиминисање хазарда података
  - смањење кашњења меморије
  - хазарди контроле се смањују
  - преклапање се може применити у много већој мери

Универзитет у Београду - Математички факултет

Предности (1)

- Превазилажење *Флиновог уског грла*
  - *"све док се чита по једна инструкција у циклусу, не може се очекивати да се извршава више од једне инструкције по циклусу"*
  - једна векторска инструкција одговара већој количини посла него једна скаларна инструкција
- Елиминисање хазарда података
  - превазилазе се захваљујући употреби структурираних података (вектори, матрице) већ у фази превођења кода
- Смањење кашњења меморије
  - коришћењем преклапања рада са меморијом (*load/store*)
  - преплитањем се амортизује високо кашњење меморије

Универзитет у Београду - Математички факултет

Предности (2)

- Хазарди контроле се смањују
  - свака итерација векторског рачунара ради много више посла од инструкције скаларног рачунара
  - нпр. сабирање вектора захтева 64 пута мање понављања (гранања) него у случају скаларног приступа (ако је дужина вектора 64)
- Преклапање се може применити у много већој мери
  - услед мањег хазарда података и контроле има мање проблема
  - преклапање се користи не само као за операције са скаларним подацима него и за прослеђивање података од једне до друге функционалне јединице
    - тзв. *уланчавање*

Универзитет у Београду - Математички факултет

Перформансе преклапања

- Перформансе извршавања са преклапањем зависе од броја корака у току извршавања и броју извршених операције (дужине вектора)

$$фактор Убрзања = \frac{трајање Без Преклапања}{трајање Са Преклапањем}$$

Универзитет у Београду - Математички факултет

Перформансе преклапања (2)

- У идеалном случају, преклапање са n корака би требало да омогући фактор убрзавања n
- Постоје два фактора који то онемогућавају:
  - пуњење тока инструкција
    - на почетку израчунавања ток инструкција је празан
    - потребно је n циклуса да се напуни
    - до тада нису све јединице запослене
  - пражњење тока инструкција
    - на крају израчунавања више нема инструкција (нпр. повратак из процедуре)
    - ипак, морају се довршити све инструкције да би се добио резултат
    - током пражњења нису све јединице запослене

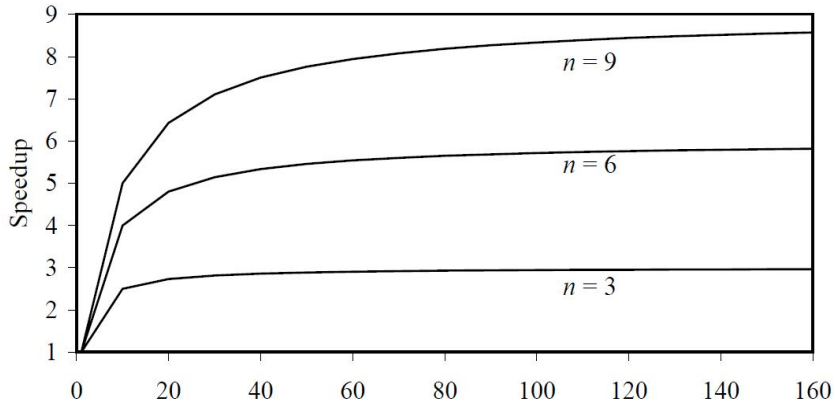
Универзитет у Београду - Математички факултет

Перформансе преклапања (3)

- Пример:
  - претпоставимо да морамо обавити N израчунавања
  - свако захтева $n*T$ јединица времена
  - без преклапања то је $N*n*T$ јединица времена
  - са преклапањем дужине n
    - први резултат се добија након $n*T$ јединица времена
    - затим се добија по један на сваких T јединица времена
    - укупно $n*T + (N-1)*T = (n+N-1) * T$ јединица времена
  - фактор убрзања је:
    - $n*N*T / (n+N-1) * T = n*N / (n+N-1)$
  - у случају великог N фактор убрзања се приближава n
    - губи се утицај пуњења и пражњења
  - у случају $N=1$ фактор убрзавања је 1
    - перформансе су исте као и без преклапања

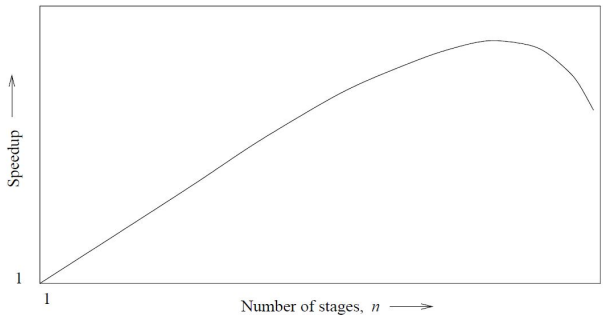
Универзитет у Београду - Математички факултет

Перформансе преклапања (4)



Перформансе преклапања (5)

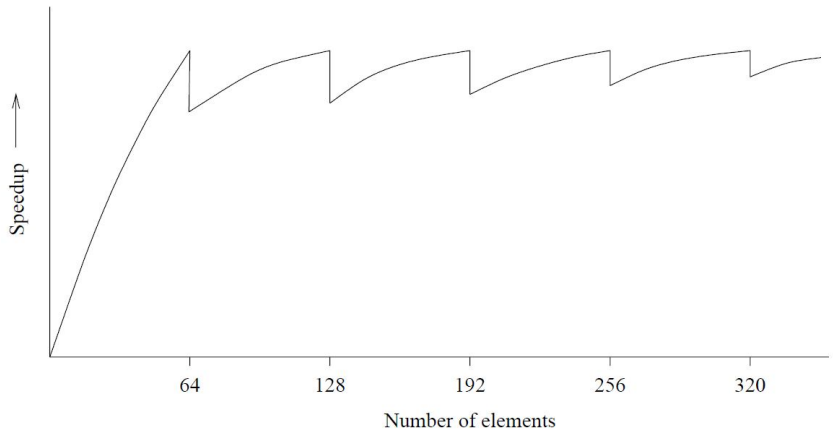
- Може се стећи утисак да се повећавањем броја корака при преклапању подижу перформансе
- То је само делимично тачно – када се пређе нека граница онда цена пуњења и пражњења тока инструкција постаје превисока
  - граница зависи од много фактора
  - на њу највише утичу хазарди података и контроле



Перформансе векторских проц.

- Перформансе у великом делу зависе од примењеног преклапања
- Други фактор је дужина векторских регистара
  - нека је, нпр., дужина регистара 64 податка
  - ако занемаримо све остале утицаје, добитак у перформансама расте када се количина података који се обрађују повећава од 1 до 64
  - међутим, када се пређе 64, добитак се смањује, да би се поново повећавао од 65 до 128
  - слично и касније...

Перформансе векторских проц.



[P271]

Увод у архитектуру рачунара

Саша Малков

Тема 17

Процесор *IBM Power 7*

Основни подаци

- представљени 2010.
- 1.2 милијарде тразистора
- 45nm
- 2.4-4.25 GHz

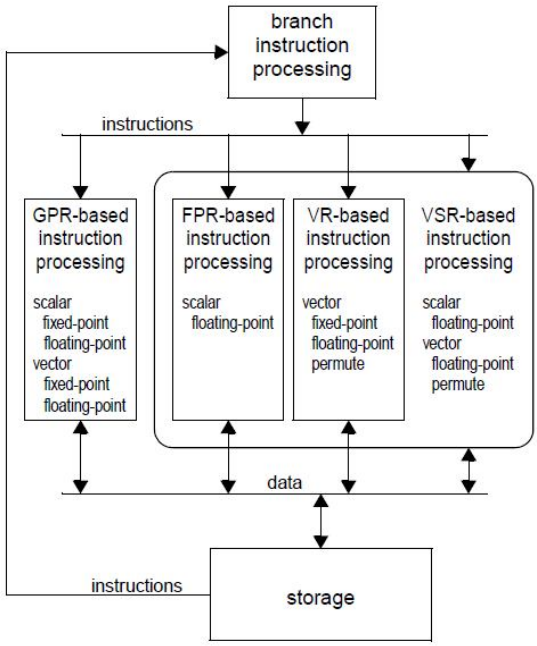
- 4, 6 или 8 језгара по чипу
- до 4 нити по језгру
- три нивоа кеша
  - ниво 1 – 32kB за инструкције и 32kB за податке, по језгру
  - ниво 2 – 256kB по језгру
  - ниво 3 – укупно 4MB по језгру, заједнички за сва језгра

Архитектура инструкција

- *POWER ISA*
- 12 извршних јединица по језгру
  - 2 целобројне / са фиксним зарезом
  - 2 јединице *load/store*
  - 4 јединице за рад у покретном зарезу двоструке тачности
  - 1 векторска јединица
  - 1 јединица за рад у децималном покретном зарезу
  - 1 јединица за гранања
  - 1 јединица условног регистра
- извршавање инструкција преко реда
- може да извршава до 6 инструкција по циклусу
  - под одређеним условима чак 8 инструкција у покретном зарезу по циклусу

Универзитет у Београду - Математички факултет

Логички дијаграм имп. инструкција



Универзитет у Београду - Математички факултет

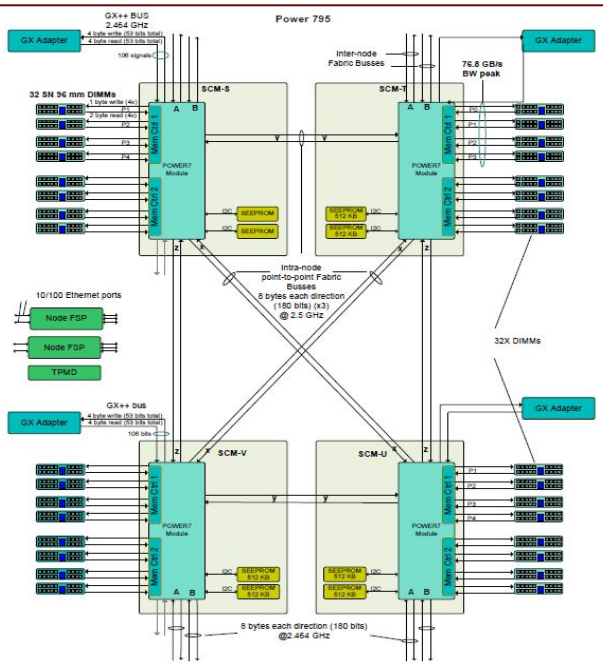
Регистри

- 32 регистра опште намене, 64-битни
- 32 регистра за рад у пкретном зарезу, 64-битни
- 32 векторска регистра, 128-битни
- 64 векторско-скаларна регистра, 128-битни
- ...

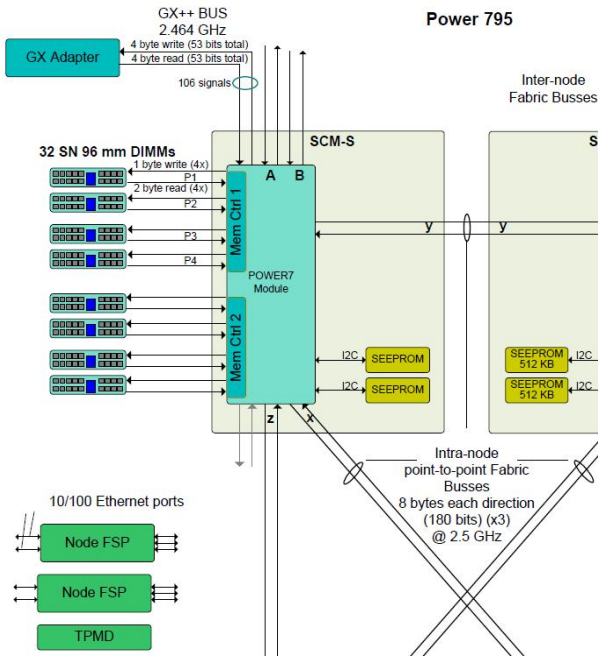
Повезивање

- Веома добро прилагођен повезивању у озбиљан вишепроцесорски систем
  - повезивање 4 чипа по *књизи* (*processor book*)
  - повезивање до 8 модула (књига) по рачунару
    - 8 x 4 x 8 = 256 језгара
  - интерне линије за повезивање чипова
    - 360GB/s
    - за међусобно повезивање до 32 чипа
- Капацитет према захтеву
  - (енгл. *capacity on demand – CoD*)
  - активирање чипова према лиценци, било стално, било повремено према потреби

Књига (*book*)



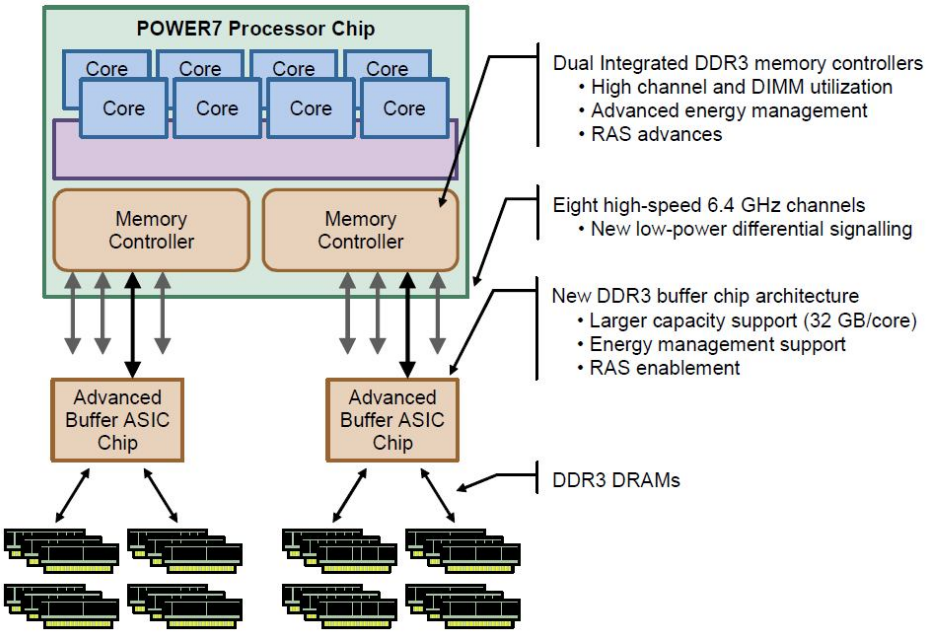
Књига (*book*)



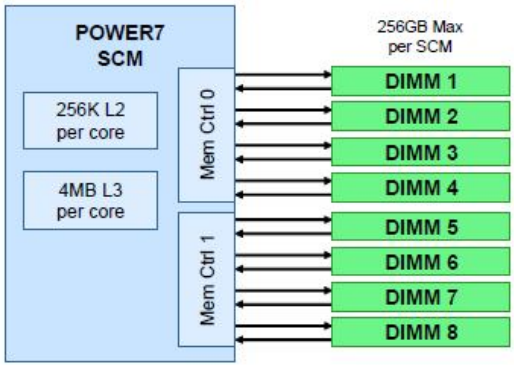
Меморија

- два четвороканална меморијска контролера *DDR3* по чипу
  - модули од по 8, 16, 32 GB
  - 8 модула по чипу
  - 32 модула по кћизи
  - укупно до 8TB
- пропусност (при 4GHz)
  - кеш нивоа 1 – 192 GB/s
  - кеш нивоа 2 – 192 GB/s
  - кеш нивоа 3 – 128 GB/s
  - системска меморија
    - макс. брзина читања 102.33 GB/s
    - макс. брзина писања 51.16 GB/s
    - комбинована макс. брзина 136.44 GB/s

Меморија



Меморија



Универзитет у Београду - Математички факултет

Виртуализација

- Претпоставља интензивну виртуализацију
- До 10 ВМ по језгру

Универзитет у Београду - Математички факултет

Процесор *IBM Power8*

- Представљен априла 2014.
- Око 2.4x просечно повећање перформанси по чипу
  - Унапређена језгра
  - Унапређен кеш
  - Унапређене магистрале
  - Велики број других технолошких унапређења

IBM Power8 – језгра

- Унапређења језгара у односу на *Power7*
  - 16 извршних јединица по језгру
    - нова јединица за преузимање инструкција (x2)
    - нова криптографска јединица
    - додата 1 векторска јединица (сада постоје укупно 2)
  - 2x више нити по језгру (до 8 нити по језгру)
  - 1.5x више језгара (12 језгара)
  - 1.6x перформансе језгра
  - очекивана радна фреквенција 2.5-5GHz (*Power7* до 4.25GHz)
  - 2.4x повећање перформанси по чипу на истој ФВ
  - 22nm (*Power7* 45nm)

IBM Power8 – кеш

- Унапређења кеша у односу на *Power7*
  - 2x већи кеш података нивоа 1
    - 32KiB кеш инструкција, 8-скуп-асоцијативан
    - 64 KiB кеш података, 8-скуп-асоцијативан
  - 2x већи кеш нивоа 2 (512 KiB)
  - 2x већи кеш нивоа 3 по језгру (8 MiB)
  - додат нови екстерни кеш нивоа 4 до 128 MiB
  - 4x већа таблица транслација
  - 2x шира магистрала између L1 и L2 кеша (R/W 256/64 B)

Универзитет у Београду - Математички факултет

IBM Power8 – магистрале

- Унапређења магистрала у односу на *Power7*
  - 2.3x меморијска магистрала (до 8 канала, укупно до 230GB/s)
  - до 32 DDR3 или DDR4 модула, укупно до 410 GB/s
  - до 1 TiB по процесору
  - 2.4x УИ магистрала (48GB/s)
  - 10x комуникација са другим процесорима (12 канала x 150 GB/s = 3.6 TB/s)

Универзитет у Београду - Математички факултет

IBM Power8 – друга унапређења

- Фузија инструкција
  - Поцесор повезује информације о две суседне инструкције и извршава их као једну, брже него да нису спојене
- Уграђене инструкције за енкрипцију
- Динамичка штедња енергије
- Унапређени надзорник ВМ
- ...

Универзитет у Београду - Математички факултет

[P271]

Увод у архитектуру рачунара

Саша Малков

Тема 18

Виртуалне машине

Контрола рачунарског система (1)

- Првобитна решења
  - Апликативни софтвер
    - непосредно користи хардвер

Контрола рачунарског система (2)

- Једноставни монопрограмски системи
  - БИОС (енгл. *Basic Input Output System*)
    - програми за непосредну контролу хардвера
  - Апликативни софтвер
    - користи хардвер посредством БИОС-а
    - самостално управља ресурсима рачунара

Контрола рачунарског система (3)

- Оперативни системи
  - БИОС (енгл. *Basic Input Output System*)
    - програми за непосредну контролу хардвера
  - Оперативни систем
    - програми за управљање ресурсима рачунарског система
  - Апликативни софтвер
    - користи све рачунарске ресурсе (и хардвер) посредством оперативног система

Универзитет у Београду - Математички факултет

Апстракција елемената хардвера

- Различити концепти / подсистеми ОС апстрахују различите елементе хардвера
  - Систем датотека
    - апстракција дискова
  - Процеси
    - апстракција адресног простора и изоловане меморије
  - Нити
    - апстракција процесора

Универзитет у Београду - Математички факултет

Раздвојеност ОС и хардвера

- Савремени ОС су веома сложени
- Обављају велики број различитих фунцкија
- Ниво апстракције ОС је далеко изнад хардвера
- Штавише, исти ОС често раде на потпуно различитом хардверу

Универзитет у Београду - Математички факултет

Слој апстракције хардвера

- Савремени ОС имају слој за апстракцију хардвера, који се непосредно ослања на (потенцијално специфичан) хардвер и остатку ОС-а пружа уједначену апстраховану слику
  - нпр. *Windows HAL* (*hardware abstraction layer*)
- Апликативни програми, као и већина компоненти ОС-а, раде користећи услуге слоја апстракције хардвера, практично потпуно независно од конкретних уређаја

Универзитет у Београду - Математички факултет

Виртуалне машине

- Концепт виртуалне машине представља наредни ниво апстракције:
  - слој апстракције хардвера се издваја из ОС-а и представља независну целину – Надзорник виртуалних машина (НВМ, енгл. *Virtual Machine Monitor – VMM*)
  - један *VMM* може истовремено да опслужује више оперативних система на истом рачунару
  - из угла сваког ОС-а изгледа као да је рачунар под "потпуном" контролом
  - из угла апликативног софтвера не постоји *значајна* разлика рада под ОС-ом који непосредно управља рачунаром и под ОС-ом који ради под НВМ-ом

Универзитет у Београду - Математички факултет

Виртуалне машине

- Концепт виртуалне машине представља наредни ниво апстракције:
  - слој апстракције хардвера се издваја из ОС-а и представља независну целину – Надзорник виртуалних машина (НВМ, енгл. *Virtual Machine Monitor – VMM*, или *hypervisor*)

Универзитет у Београду - Математички факултет

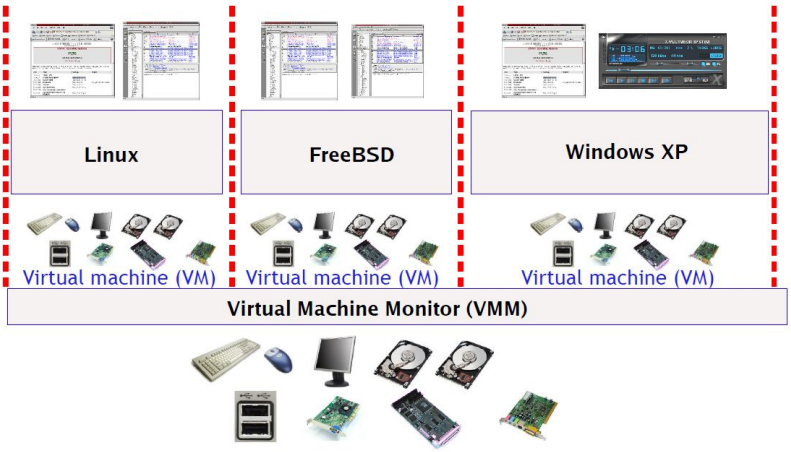
Виртуалне машине (2)

- НВМ се непосредно ослања на хардвер
- Један НВМ може истовремено да опслужује више оперативних система на истом рачунару
- Из угла сваког ОС-а изгледа као да је рачунар под његовом *потпуном* контролом
- Из угла апликативног софтвера *није видљиво* да ли је ОС непосредно над хардвером или под НВМ-ом

Терминологија

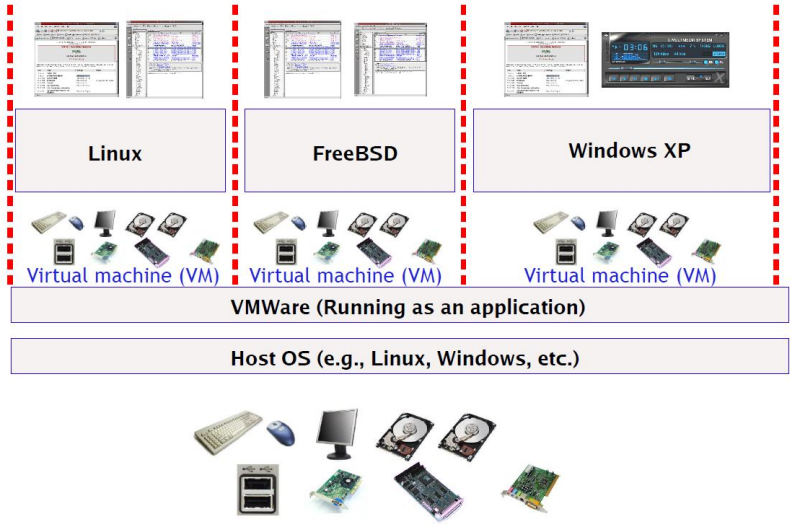
- Надзорник виртуалних машина
  - слој софтвера који се непосредно ослања на хардвер и подржава више ОС-а
- Виртуална машина
  - апстрахован рачунарски систем који ради под НВМ-ом, а не на физичком хардверу
- Виртуални уређај
  - виртуална машина са инсталираним и конфигурисаним софтвером, потпуно спремна за рад
  - довољно ју је ископирати на други рачунар и укључити
  - (енгл. *virtual appliance*)
- Машина домаћин (енгл. *host machine*)
  - физички рачунар и софтвер (ОС или НВМ) који ради на њему
- Машина гост (енгл. *guest machine*)
  - виртуални рачунар и софтвер који ради на њему, ВМ

Пример архитектуре ВМ



Универзитет у Београду - Математички факултет

Пример архитектуре ВМ (2)



Универзитет у Београду - Математички факултет

Мотивација

- Разликују се мотиви за виртуализацију
  - радних станица и
  - сервера

Мотивација – радне станице

- Повећана продуктивност
  - неки програми не постоје за све врсте ОС
  - различити ОС нису једнако погодни за све врсте послова
  - једноставније је имати више ВМ на једном рачунару него одржавати више физичких рачунара на једном радном месту
- Изоловање осетљивих система
  - заштита од вируса и других напасти
  - обука администратора, испробавање осетљивих поступака
  - обучавање у развоју системског софтвера
- Развојне радне станице
  - реално понашање мрежног окружења на једној радној станици
  - изградња на различитим развојним окружењима
  - тестирање
- Виртуални уређаји
  - поједностављена дистрибуција припремљених сложених окружења

Мотивација – сервери

- Консолидација сервера
  - више мањих сервера се виртуализује на једном физичком рачунару
  - раздвајање одговорности сервера, а тиме и олакшано одржавање
  - већа искоришћеност хардвера
  - уштеда на хардверу
- Напредна серверска окружења
  - олакшана репликација
  - једноставнија имплементација високе расположивости
  - пребацивање активних машина са једног на други рачунар

Универзитет у Београду - Математички факултет

Циљеви

- Стварање илузије *комплетних* физичких машина
  - процесори, меморија, нивои заштите, систем прекида, улазно излазни уређаји,...
- Потпуна компатибилност са постојећим физичким хардвером
  - ОС и апликације у ВМ морају радити у неизмењеном облику
- Потпуна међусобна изолованост виртуалних машина
  - софтвер једне ВМ не сме бити у стању да ступи у контакт са виртуалним хардвером друге ВМ
- Омогућавање истовременог рада више ОС-а
  - потребан је висок ниво скалабилности
- Високе перформансе
  - посебно процесор и меморија
  - динамичко прилагођавање оптерећењу
- Флексибилност
  - једноставно управљање расположивим ресурсима током рада ВМ
  - једноставан трансфер активне ВМ на други физички рачунар

Универзитет у Београду - Математички факултет

Врсте виртуалних машина

- Системске виртуалне машине
  - (називају се и *хардверске* виртуалне машине)
  - пружају апстракцију читавог рачунара
  - примери: *KVM, VMWare, VirtualBox, XEN,...*
  - само оне представљају данашњу тему
- Процесне виртуалне машине
  - пружају делимичну апстракцију рачунара, довољну за извршавање једног процеса
  - примери: *JVM, CLR,...*
- Виртуализација на нивоу ОС-а
  - комбинација – у оквиру једног ОС-а пружају парцијалне инстанце истог ОС-а
  - примери: *FreeBSD Jail, LXC, OpenVZ,...*

Универзитет у Београду - Математички факултет

Системске виртуалне машине

- Пуна виртуализација
- Хардверски подржана виртуализација
- Парцијална виртуализација
- Паравиртуализација

Универзитет у Београду - Математички факултет

Системске виртуалне машине

- **Пуна виртуализација**
  - ВМ симулира све елементе хардвера потребне за рад неизмењеног оперативног система ВМ
  - први представник: *IBM CP-40*
  - примери: *VMware* (*Workstation*, *Server*), *Oracle VirtualBox*,...
- Хардверски подржана виртуализација
- Парцијална виртуализација
- Паравиртуализација

Универзитет у Београду - Математички факултет

Системске виртуалне машине

- Пуна виртуализација
- **Хардверски подржана виртуализација**
  - хардвер пружа механизме за ефикасан рад надзорника, симулирање хардвера и изоловање ВМ
  - први представник: *IBM VM/370* (1972.)
  - примери: *Intel/AMD x86* (2005.), *IBM Power Architecture*, *VMware* (*Workstation*, *Fusion*), *Oracle VirtualBox*, *Xen*,...
- Парцијална виртуализација
- Паравиртуализација

Универзитет у Београду - Математички факултет

Системске виртуалне машине

- Пуна виртуализација
- Хардверски подржана виртуализација
- **Парцијална виртуализација**
  - ВМ симулира већину (али не све) елемената хардвера
  - ОС не може да ради у потпуности у оквиру ВМ
- Паравиртуализација

Универзитет у Београду - Математички факултет

Системске виртуалне машине

- Пуна виртуализација
- Хардверски подржана виртуализација
- Парцијална виртуализација
- **Паравиртуализација**
  - ВМ не симулира хардвер већ нуди *API* који замењује одређене компоненте ОС-а
  - већа ефикасност али мања флексибилност
  - примери: *KVM*, *Xen*,...

Универзитет у Београду - Математички факултет

Проблеми у имплементацији

- Гостујући ОС позива привилеговане инструкције
  - а то не би смео да чини, зато што то сме само надзорник (или домаћински ОС)
- Гостујући ОС жели да управља страницама виртуалне меморије
  - то производи потенцијалне проблеме зато што је тесно повезано са управљањем физичком меморијом
- Гостујући ОС мора да *верује* да ради на физичкој машини
  - ВМ мора да подржава све инструкције процесора, а неке нарушавају изолованост
  - ВМ мора да подржава све компоненте рада са виртуалном меморијом (страничење, сегментација,...)
  - ВМ мора да подржи улазно-излазне уређаје

Универзитет у Београду - Математички факултет

Режим рада

- ОС по правилу има компоненте које раде у повлашћеном режиму, што омогућава пуну контролу хардвера
- ВМ по правилу не би смела да ради у повлашћеном режиму зато што не сме непосредно да приступа физичком хардверу
- Последица: ОС у ВМ мора да ради у корисничком, а не у повлашћеном режиму

Универзитет у Београду - Математички факултет

Режим рада – интерпретација

- Једно решење је интерпретација
  - ВМ не извршава инструкције непосредно већ се рад процесора симулира програмом који интерпретира инструкције
  - може да се оствари пуна безбедност и изолованост
- Проблеми
  - сложеност – потребно је имплементирати читаву архитектуру процесора
  - спорост – интерпретирање инструкција процесора је за ред величине спорије од њиховод непосредног извршавања

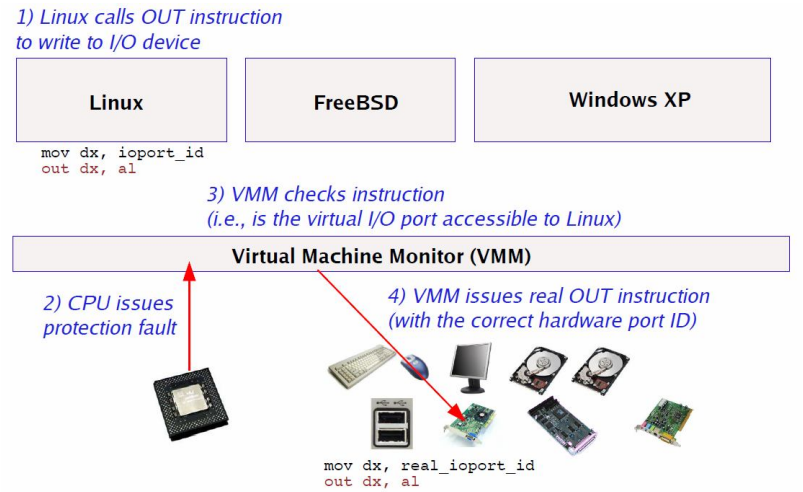
Универзитет у Београду - Математички факултет

Режим рада – извршавање

- Друго решење је непосредно извршавање
  - ВМ непосредно извршава инструкције процесора, али у корисничком режиму
  - овакав начин рада је високо ефикасан
  - покушаји извршавања заштићених инструкција изазивају *замке* и предају контролу надзорнику
    - надзорник проверава инструкцију и одлучује шта даље
      - може да безбедно емулира операцију и настави рад ВМ
      - може да забрани операцију и искључи ВМ
- Проблеми
  - умерена сложеност – потребно је имплементирати све заштићене инструкције процесора
  - умерено успорење – интерпретирају се само заштићене инструкције процесора
  - неке инструкције раде исправно само у заштићеном режиму, али не произвоге грешке иначе!!!

Универзитет у Београду - Математички факултет

Режим рада – извршавање (2)



Изоловање ОС

- Уобичајено је да ОС и апликације не раде на истом нивоу привилегија
- Ако раде на истом, може доћи до проблема услед безбедности или исправности рада
- Уобичајено је да се користе додатни нивои заштите процесора, који иначе нису искоришћени:
  - прстен 0: НВМ
  - прстен 1: ВМ ОС
  - прстен 3: ВМ апликације

Режим рада – превођење

- Треће решење је превођење кода при учитавању
  - непосредно пре извршавања програма (или чак током извршавања) надзорник протрчи кроз код и преведе све проблематичне инструкције
    - нпр. *x86* инструкција *POPF* би требало да постави ново стање заставица, али у корисничком режиму не поставља заставице прекида
  - праве се копије страница кода у меморији и одговарајуће таблице пресликавања
- Проблеми:
  - смањена ефикасност
- Паравиртуализација је вид оваквог рада, с тим да су сви делови ОС-а који користе привилеговане инструкције унапред преведени и прилагођени раду у ВМ
  - веома једноставно
  - врло ефикасно

Универзитет у Београду - Математички факултет

Рад са меморијом

- Не сме свака ВМ да имплементира сопствену ВМ, зато што постоји само једна физичка меморија
- Уводи се додатни ниво индирекције
  - ВМ смеју само да читају таблице страница
  - одржава их надзорник

Универзитет у Београду - Математички факултет

Литература

- *Sivarama Dandamudi*, ***Fundamentals of Computer Organization and Design***, *Springer, 2002.*
- *Andrew Tanenbaum*, ***Архитектура и организација рачунара***, *Микро књига, 2007.*
- *Ненад Митић*, ***Основи рачунарских система***, *Математички факултет, 2002.*
- *James E. Smith, Ravi Nair*, ***The Architecture of Virtual Machines***, *. Computer (IEEE) 38 (5): 32–38. 2005.*