[P220]

Увод у архитектуру рачунара 9

Саша Малков
Универзитет у Београду
Математички факултет
2013/2014

[P271]

Увод у архитектуру рачунара

Саша Малков

Тема 10

Виртуална меморија

Улога виртуалне меморије

- Концепт виртуалне меморије омогућава да програми у рачунарском систему употребљавају већи меморијски простор него што је стварна величина физички присутне меморије

Претходници

- Пре развоја технике виртуалне меморије
  - или је програм са подацима морао да стане цео у меморију
  - или је програмер морао експлицитно да управља тзв. механизмом преклапања (енгл. *overlay*)
- Техника виртуалне меморије аутоматизује старање о меморији и ослобађа програмера сувишног старања о физичким ресурсима
  - аутоматски се подаци и делови програма пребацују из физичке меморије на диск и обратно

Основне функције

- Две основне функције које остварује техника виртуалне меморије су:
  - Премештање
    - сваки програм користи свој виртуални адресни простор
    - током извршавања тај адресни простор се може пресликавати у различите физичке меморијске локације
    - детаљи управљања овим пресликавањем немају никакве везе са имплементацијом програма
    - сву бригу око пресликавања воде процесор и оперативни систем
  - Заштита
    - раздвојеност адресних простора програма пружа међусобну изолованост програма и заштиту података и кода

ВМ и кеш

- Виртуална меморија и кеш деле неке концепте и претпоставке
  - успешност примене виртуалне меморије и кеша почива на локалности простора и времена
  - као што је кеш мањи и бржи од главне меморије, тако је главна меморија бржа и мања од диска
- Разлике
  - различита мотивација и намена имају већи број последица
  - различит ниво перформанси омогућава да се део технике ВМ имплементира у софтверу, док се све у вези кеша имплементира искључиво у хардверу

Основи концепта ВМ

- Основи концепта виртуалне меморије су:
  - организација меморије по страницама
  - пресликавање виртуалних и физичких адреса
  - страничење помоћу диска

Странице меморије

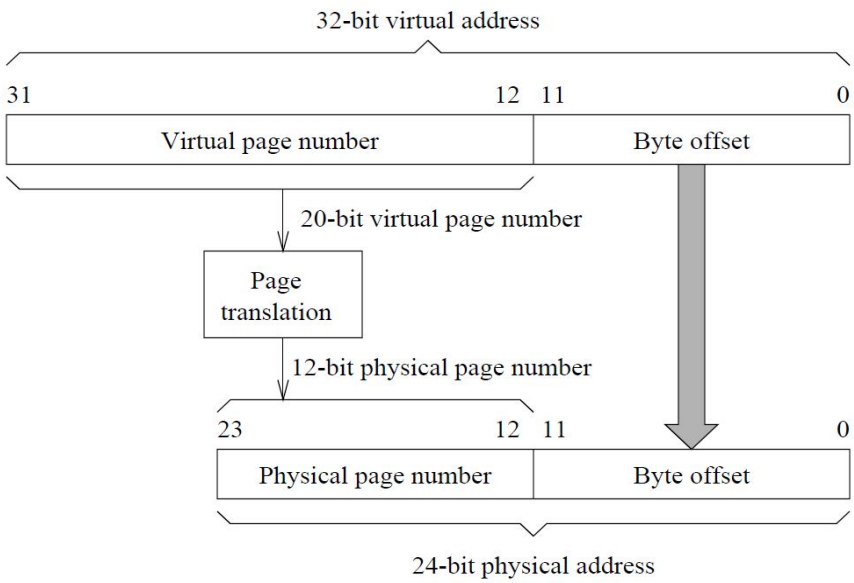
- Целокупан виртуалан адресни простор се дели на *виртуалне странице*
  - Битови виртуалне адресе се деле на *број виртуалне странице* и *адресу у страници*
- Слично томе, физичка меморија се дели на *физичке странице* (или *оквире за странице*)
  - Битови физичке адресе се деле на *број физичке странице* и *адресу у страници*

Пресликавање адреса

- При пресликавању виртуалних адреса у физичке
  - претпоставља се да су физичке и виртуалне странице исте величине
  - адреса у оквиру странице се не мења пресликавањем
  - број виртуалне странице се пресликава у број одговарајуће физичке странице
  - уобичајена величина странице је *4KiB*
- За пресликавање је задужена јединица за управљање меморијом (енгл. *memory management unit*)

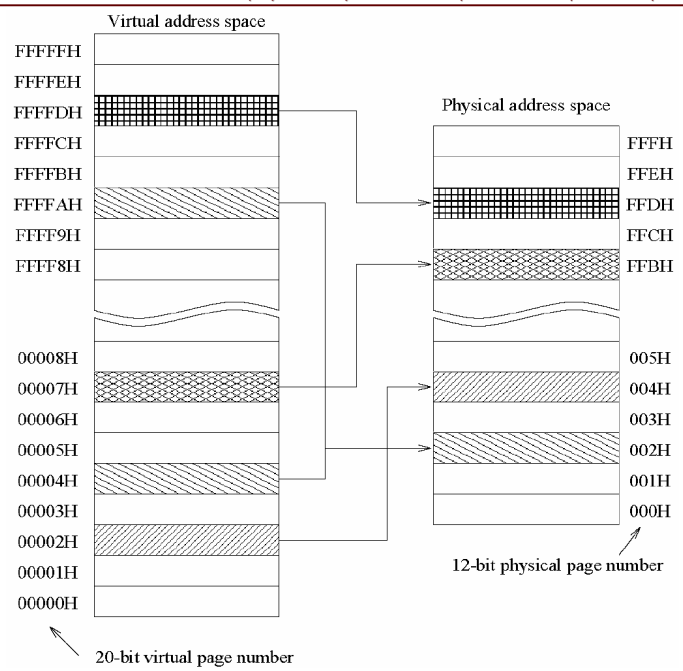
Универзитет у Београду - Математички факултет

Пример пресликавања 32-битне виртуалне у 24-битну физичку адресу, са страницом од *4KiB*



Универзитет у Београду - Математички факултет

Пример пресликавања 32-битног виртуалног у 24-битни физички адресни простор



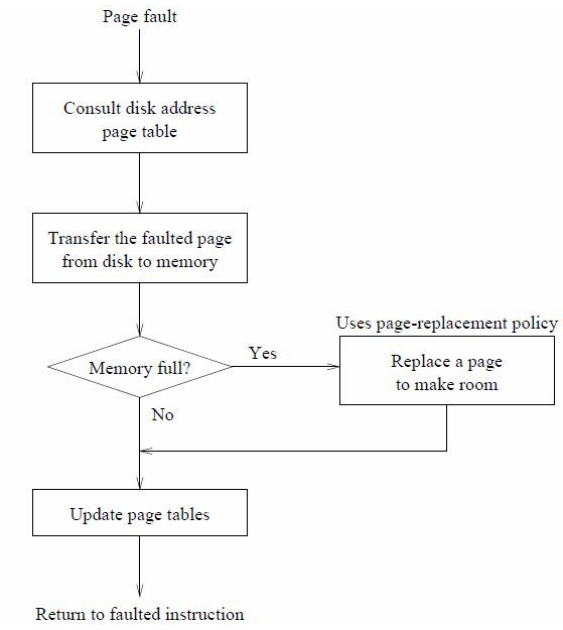
Употреба диска

- Из угла ОС-а свака страница ВМ је
  - у главној меморији или
  - на диску
- Штавише, и странице које су у главној меморији имају своју слику на диску

Страничење по захтеву

- Ако се покуша приступити садржају странице која није у главној меморији, тада настаје *грешка страничења* (*странична грешка*, енгл. *page fault*)
  - има сличности са промашајем кеша
  - у том случају ОС је задужен да одреагује и учита тражену страницу у главну меморију и ажурира податке који се односе на пресликавање страница
  - ово се назива *страничење по захтеву*, зато што се обавља када се захтева дата страница
- Ако је главна меморија пуна, примењује се *политика замене странице*

Страничење по захтеву



Имплицитно страничење

- ОС може да предузима страничење и имплицитно, тј. без експлицитно исказане потребе за страницом, уколико процени
  - да се страничењем неће значајно ослабити перформансе активних процеса
    - нпр. да се диск тренутно не употребљава
  - да се са релативно високом вероватноћом може претпоставити да ће у скорој будућности бити потребне неке странице које трнутно нису у физичкој меморији

Политике замењивања страница

- При разматрању политика замењивања узимају се у обзир разлике између ВМ и кеша
  - цена промашаја у случају ВМ је вишеструко већа него у случају кеша, па је већи значај добре политике
  - политике замењивања кеша се имплементирају у хардверу, а у случају ВМ у софтверу, што пружа већу слободу
- Због тога се у случају ВМ тежи постизању што квалитетнијег одабира страница, а по цену сложености алгоритма

Политике замењивања (2)

- Неке од политика замењивања су:
  - *FIFO*
  - политика друге шансе
  - политика ретко употребљаване странице
  - политика најдуже неупотребљаване странице

Политика *FIFO*

- Замењује се она страница која је најдуже у главној меморији
  - једноставна имплементација
  - мана је што не узима у обзир употребу странице
  - релативно слабе перформансе
  - ретко се примењује

Политика *друге шансе*

- Унапређена политика *FIFO*
- Замењује се она страница која је најдуже у главној меморији, а да није ниједанпут реферисана
  - релативно једноставна имплементација
    - додаје се по један додатни бит за сваку страницу, који означава да ли је поново употребљавана након иницијалног учитавања
  - умерено значајан допринос перформансама

Универзитет у Београду - Математички факултет

Политика *ретко употребљаване*

- Замењује се страница која је најмање пута реферисана
  - релативно једноставна имплементација
    - свакој страници се додаје бројач реферисања
    - оперативни систем повремено поништава бројаче
  - умерено значајан допринос перформансама

Универзитет у Београду - Математички факултет

Политика *најдуже неупотребљаване*

- Замењује се страница чији садржај најдуже није реферисан
  - иако се имплементира у софтверу а не у хардверу, ипак је непрактична пуна имплементација
  - најчешће се апроксимира
    - начелно слично као у случају кеша
    - али ипак се тежи бољој апроксимацији
  - ово је најчешће примењивана политика

Политике писања

- У случаја кеша разматрали смо писање са пропуштањем и писање са преписивањем
- У случају ВМ писање са пропуштањем није опција, због велике разлике у брзини диска и главне меморије
- Додатно, у случају ВМ постоји заштита на нивоу страница и процеса (попут закључавања) па је писање са преписивањем безбедније него у случају кеша
  - не постоји опасност да неко употребљава неажуриране податке из странице на диску

Улога величине странице (1)

- Мале странице су добре због:
  - интерне фрагментације
    - величина података, програма и стека није цео број страница
    - просечно је пола странице неупотребљено
    - што је страница мања, искоришћеност је већа
  - бољег погађања
    - што је већа страница, то ће при њеном учитавању у главну меморију заступљеност непотребних садржаја бити већа

Улога величине странице (2)

- Велике странице су добре због:
  - величине таблице страница
    - што су веће странице, биће их мање, па су и таблице мање
  - времена приступа диску
    - време приступа диску је много веће него време читања
      - реда 10ms по приступу, 100MiB/s читање
      - читање 100 страница од 4KiB траје
        - 100 x (10ms + 4/100ms) = 10.04s
      - читање 25 страница од 16KiB траје
        - 25 x (10ms + 16/100ms) = 2.54s
    - веће странице редукују број приступа диску и убрзавају рад

Примери величине странице

- *Intel x86*
  - странице су уобичајено 4*KiB*
  - подржане су и странице величине 2/4*MiB*
    - почев од процесора *Pentium*
    - документовано тек од процесора *PentiumPro*
    - у основи 4*MiB*, али 2*MiB* ако се користи *PAE*
- *Power PC*
  - странице су 4*KiB*
- *MIPS R4000*
  - подржава 7 величина страница, од 4*KiB* до 16*MiB*

Начин пресликавања

- Због високе цене промашаја потребно је да буде што мањи степен промашаја
- Због тога се у случају ВМ уобичајено примењује пуно асоцијативно пресликавање страница
- Имплементира се применом *таблица транслација*
  - краће се називају *таблице страница*

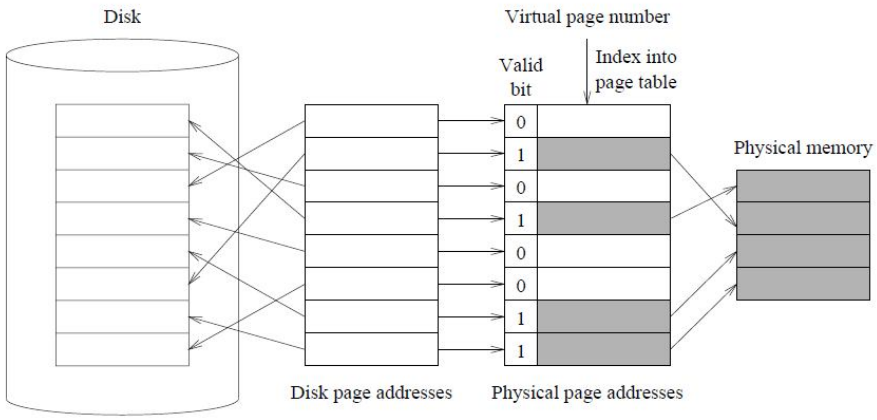
Таблице страница

- Примитивна имплементација
  - у основном облику таблица садржи скуп парова
    - (број виртуалне странице, број физичке странице) или
    - (број виртуалне странице, локација на диску)
    - ту су и додатни подаци о стању страница
  - по правилу се приступа по броју виртуалне странице
  - оваква имплементација захтева сувишна уређивања
- Поједностављена имплементација
  - таблица се имплементира као низ индексиран бројевима виртуалних страница

Таблице страница (2)

- Може се моделирати са две таблице
  - таблица физичких страница
    - низ индексиран бројевима виртуалних страница
      - (енгл. *virtual page number - VPN*)
    - садржи бројеве физичких страница
    - садржи и додатне контролне битове
  - таблица адреса на диску
    - низ индексиран бројевима виртуалних страница
    - садржи локације страница на диску
- Обично се имплементира као једна таблица која садржи обе врсте података

Илустрација таблице страница, за физичку меморију од 4, и ВМ од 8 страница



Универзитет у Београду - Математички факултет

Ставке табеле страница

- Свака *ставка табеле страница* (енгл. *page table entry - PTE*) садржи:
  - број физичке странице (енгл. *physical page number – PPN*)
    - локација дате странице у главној меморији
    - води се за странице које постоје у главној меморији
  - адреса странице на диску (енгл. *disk page address*)
    - локација дате странице на диску
    - води се за све странице
  - бит исправности (енгл. *valid bit*)
    - означава да ли је страница у меморији или не
  - ...

Универзитет у Београду - Математички факултет

Ставке табеле страница (2)

- ...
- бит измењености (енгл. *dirty bit*)
  - означава да ли је садржај странице мењан
  - ако је мењана, страница се при уклањању из главне меморије записује на диску
- бит референсирања (енгл. *referenced bit*)
  - користи се за имеплементацију алгоритма псеудо-*LRU*
  - ОС повремено поставља све битове референсирања на 0
  - при приступању страници бит се поставља на 1
- информација о власнику
  - ОС мора да зна ком процесу припада страница
- битови заштите
  - означавају тип приступа који власник остварује (само читање, читање и писање, извршавање,...)

Имплементација табеле страница

- Имплементација у софтверу је неефикасна
  - сваки приступ меморији би морао да се имплементира као бар два приступа меморији
    - приступање таблици пресликавања
    - приступање физичкој адреси податка
  - уобичајена употреба кеша може да омогући одређено убрзавање али то није довољно
- Имплементација у хардверу је скупа
  - савремени процесори имају велики адресни простор
  - таблице пресликавања би биле огромне и њихова имплементација у хардверу веома скупа
- Комбинована имплементација
  - у процесору се обезбеђује специјализован кеш за таблицу страница, тзв. *бафер таблице страница* (енгл. *translation lookaside buffer - TLB*)
  - имплементација свих алгоритама и политика је у софтверу – користи се само у случају промашаја

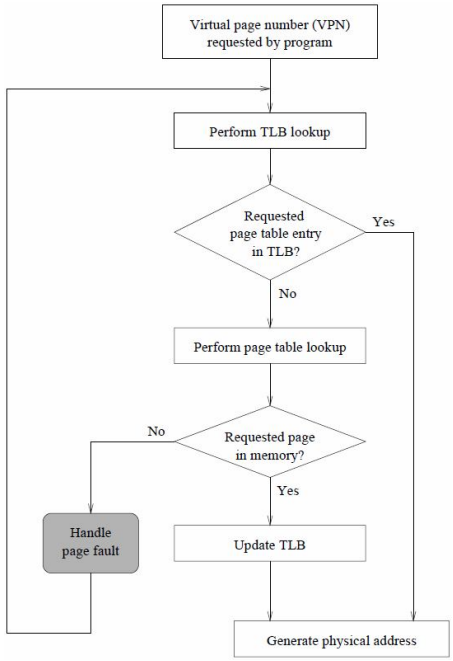
Бафер таблице страница

- *TLB* се имплементира у процесору
  - садржи податке о последњим коришћеним *PTE*
  - свака ставка *TLB* садржи:
    - број виртуалне странице (*VPN*)
    - одговарајући број физичке странице (*PPN*)
    - контролне битове
  - све ставке у *TLB* се налазе и у главној меморији у оквиру пуне таблице страница
    - допуњене додатним подацима, као што је локација на диску,...
  - као што кеш може бити подељен, тако и *TLB* може да се подели према намени – за инструкције и за податке
    - уобичајено је да уз подељен кеш иде подељен *TLB*
  - по правилу се имплементира са пуним асоцијативним пресликавањем

Употреба бафера таблице страница

- Алгоритам употребе:
  - најпре се подаци о страници траже у *TLB*
  - ако се пронађу, помоћу њих се рачуна физичка адреса
    - обично је правило да се страница реферисана из *TLB* не замењује, па је она већ у меморији
  - иначе се подаци траже у таблици страница
  - ако страница није у меморији, мора да се учита
  - рачуна се физичка адреса
  - приступа се садржају физичке меморије
- Политика замењивања ставки *TLB* је обично једноставна
  - политика случајног избора или
  - политика псеудо најдуже некоришћене

Транслација адреса помоћу бафера таблице страница



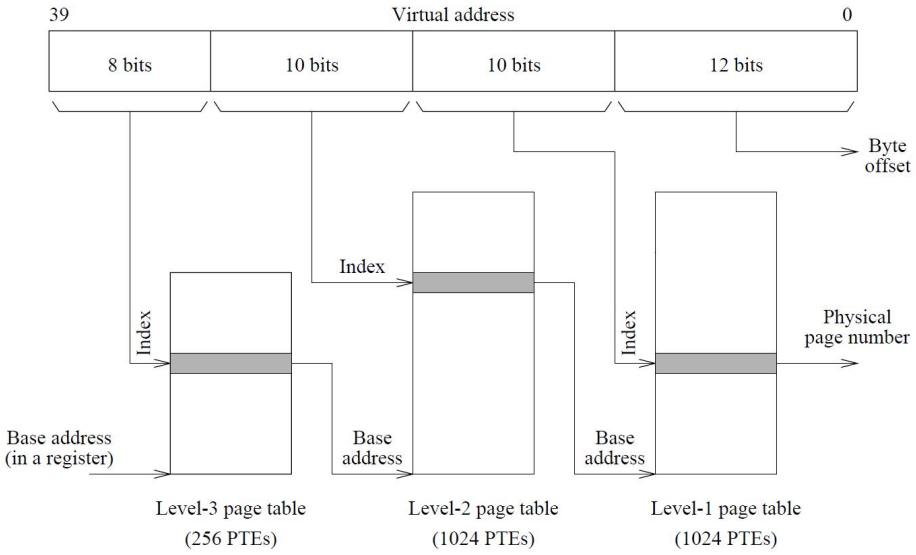
Локација таблице страница

- Таблица страница може бити врло велика
  - пример: нека су виртуалне адресе 40-битне, величина странице $4KiB$, а величина ставке 4 бајта
    - тада постоји 2^{28} страница
    - величина таблице је $1GiB$
- Зато није практично (а понекад ни остварив) да се лоцира у физичкој меморији, већ се записује у виртуалној меморији
  - и таблица се дели на странице и само се потребне странице чувају у физичкој меморији
  - за ове странице се прави додатна таблица *другог нивоа*
  - по потреби се може правити више нивоа, све док се не дође до таблице чија је величина прихватљива за стално лоцирање у физичкој меморији

Пример

- Нека су виртуалне адресе 40-битне, величина странице $4KiB$, а величина ставке 4 бајта
  - тада постоји 2^{28} страница
    - величина таблице првог нивоа је 2^{30}B = $1GiB$
    - таблица првог нивоа заузима $2^{30}/\ 2^{12} = 2^{18}$ страница
  - други ниво:
    - величина таблице другог нивоа је $2^{18} * 4$B = $1MiB$
    - таблица другог нивоа заузима $2^{20}/\ 2^{12} = 2^{8}$ страница
  - трећи ниво:
    - величина таблице трећег нивоа је $2^{8} * 4$B = $1KiB$
    - таблица трећег нивоа стаје у једну страницу

Илустрација примера хијерархијске таблице страница организоване у три нивоа



Инвертована организација таблице

- Проблем са описаном организацијом таблице страница је у величини таблице
  - за процесоре са 64-битним адресама, величина таблице може бити и 2^{54}B (!!!)
  - узрок проблема је употреба *VPN* као индекса таблице
- Алтернативни приступ је тзв. инвертована организација таблице
  - физичка адреса се користи као индекс
  - величина таблице је пропорционална физичкој меморији, а не виртуалној
  - штавише, постоји само једна таблица на нивоу система
    - у нормалној организацији постоји по једна за сваки процес
  - сложенија употреба

Универзитет у Београду - Математички факултет

Инвертована организација таблице (2)

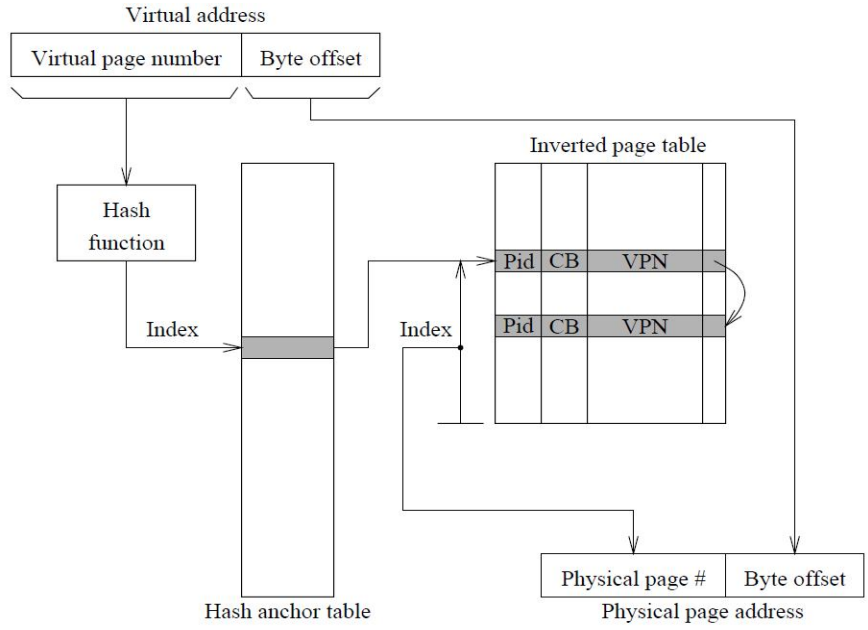
- Нама је потребно пресликавање *VPN* у *PPN*
  - ако су индекси *PPN*, ми заправо не можемо да их користимо
  - због тога се неком функцијом сецкања (енгл. *hash*) *VPN* пресликава у одговарајући индекс таблице
  - проблем са оваквим приступом је као и у случају функције непосредног пресликавања код кеша – више вредности се слика у исти индекс, тј. долази до *судара*

Универзитет у Београду - Математички факултет

Обрада судара

- Постоје две опште технике за обраду судара при употреби функција сецкања:
  - Отворено уланчавање:
    - при писању
      - у случају судара се пише у првој наредној слободној локацији
      - све ставке са истим индексом се уланчавају
    - при читању
      - добијен индекс се користи као почетна тачка за тражење одговарајуће ставке
      - тражење се наставља низ ланац све до проналажења одговарајуће ставке
  - Секундарно сецкање:
    - у случају судара се примењује додатна функција сецкања за разрешавање судара

Илустрација инвертоване организације таблице страница (примена уланчавања)



Сегментација

- Виртуализација представља линеарно пресликавање адресног простора
- Сегментација уводи додатну димензију пресликавања
- Основа концепта је да сваки процес добија *неколико независних* виртуалних адресних простора – *сегмената*
- Пуна адреса се састоји од две компоненте
  - броја (ознаке) сегмента и
  - адресе у оквиру сегмента

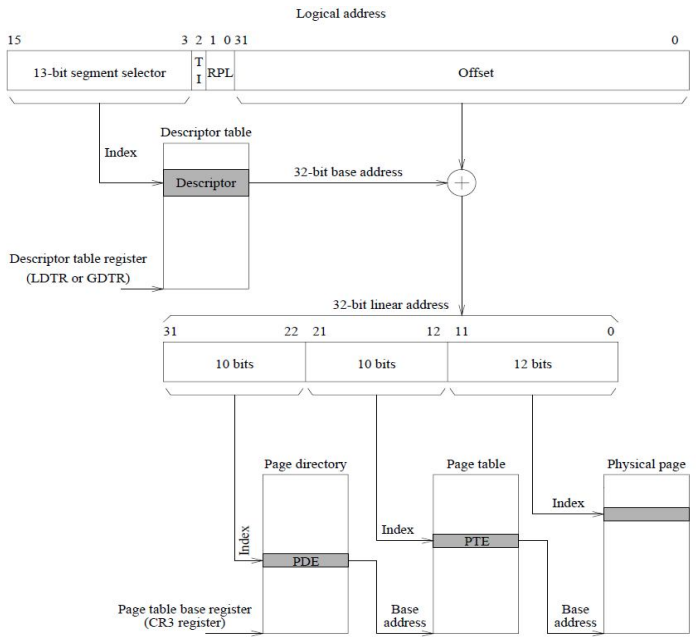
Сегментација (2)

- Сегментација је логичка организација која је видљива програмеру
- Пример: *Intel x86* процесори имају одвојене сегменте инструкција, података и стека
- Сегментација доноси
  - виши ниво заштите меморије
  - више адресних простора
  - могућност дељења сегмената међу процесима

Пример: *Intel Pentium*

- Подржава и виртуалну меморију и сегменте
  - свака од опција може да се искључи
  - *UNIX*, *Linux* не користе сегменте
- Сегментација пресликава 48-битне логичке адресе у 32-битне линеарне адресе, које се даље страничењем пресликавају у физичке адресе

Пример пресликавања адреса код процесора *Intel Pentium*



[P271]

Увод у архитектуру рачунара

Саша Малков

Тема 11

Улазно / излазни уређаји

Улазно/излазни уређаји

- Рачунарски систем обично има више различитих улазних и излазних уређаја
  - називају се и *периферни* уређаји, зато што се налазе на периферији рачунарског система
- Обезбеђују две основне функције
  - комуникацију рачунарског система са спољашњим светом и
  - чување података

Принципи рада

- Независно од врсте уређаја, основни принципи рада улазно/излазних уређаја су исти:
  - сви У/И уређаји се на системску магистралу повезују посредством одговарајућег У/И контролера
  - први разлог је у различитости уређаја
    - различити уређаји имају различите протоколе комуникације
    - уместо да процесор и системска магистрала "уче" како да комуницирају са различитм врстама уређаја, те специфичности се препуштају контролерима
    - контролери имају улогу *моста* између центра и периферије
  - други је у техничким ограничењима
    - магистрала ради на високим фрквенцијама
    - да се не би сувише грејала, ради под врло ниским напонима
    - низак напон и висока фреквенција могу да функционишу без сметњи само на врло кратким растојањима (неколико *cm*)
    - УИ уређаји захтевају дуже каблове, јачи напон и нижу фреквенцију

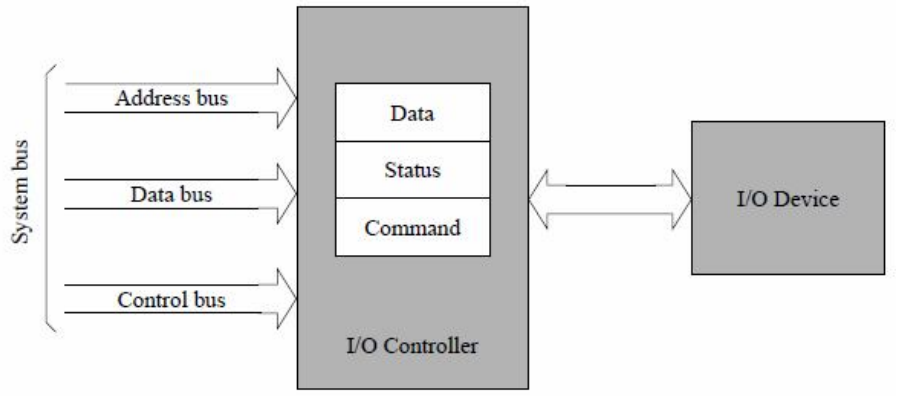
Универзитет у Београду - Математички факултет

У/И контролери

- Процесор се никада не обраћа непосредно уређајима, већ само одговарајућим контролерима
- У/И контролери имају улогу *моста* између центра (процесор, меморија и системска магистрала) и периферије (периферни уређаји рачунарског система)
- Контролери уобичајено имају три врсте интерних регистара:
  - регистар података
  - командни регистар
  - статусни регистар

Универзитет у Београду - Математички факултет

Блок-дијаграм уопштеног У/И контролера



Употреба У/И уређаја

- Два основна начина употребе У/И уређаја су
  - пресликавање портова у меморију
  - изоловани У/И

Пресликавање портова у меморију

- Сви У/И портови се пресликавају у меморијски адресни простор
- Није потребан никакав посебан интерфејс процесора
- Процесор користи уређај као меморију
- Сваки процесор може употребљавати уређаје путем пресликавања портова у меморију
- Неки процесори подржавају само овакав начин рада:
  - *PowerPC, MIPS*

Универзитет у Београду - Математички факултет

Изоловани У/И

- Изоловани У/И подразумева посебан У/И адресни простор, независан од меморијског адресног простора
- *Intel x86* процесори подржавају овакав начин рада
- Системи засновани на процесорима који подржавају изоловани У/И омогућавају избор метода
  - нпр: штампачи се користе путем изолованог У/И, а графички подсистем путем пресликавања у меморију

Универзитет у Београду - Математички факултет

Приступ портовима (*x86*) (1)

- Наредно излагање је на примеру процесора *Intel x86*
- Адресни простор за изоловани У/И је 64*KiB*
  - подржани су 8-битни, 16-битни и 32-битни портови али само док свеукупно не прелазе 64*KiB* адресног простора
    - 64Ki 8-битних портова
    - 32Ki 16-битних портова
    - 16Ki 32-битних портова
    - различите комбинације
  - изоловани У/И адресни простор је линеаран – не подлеже ни сегментацији ни страничењу

Приступ портовима (*x86*) (2)

- Постоје два типа инструкција
  - регистарске
    - за преношење појединачних података (8, 16 или 32 бита) између регистара и портова
    - *in* – чита податак са улазног порта
    - *out* – уписује податак на излазном порту
  - блоковске
    - за преношење блокова података између меморије и портова
    - *ins* – чита блок података са улазног порта
    - *outs* – уписује блок података на излазном порту

Пример – тастатура (1)

- Контролер тастатуре испитује тастатуру и извештава о притискању и отпуштању тастера у виду тзв. *кодова тастера* (енгл. *scan code*)
  - код тастера је идентификациони број који се додељује тастеру на основу његове локације
  - код тастера нема никакве везе са *ASCII* или неким другим кодовима карактера, већ улазни потпрограми преводе код тастера у одговарајуће кодове карактера

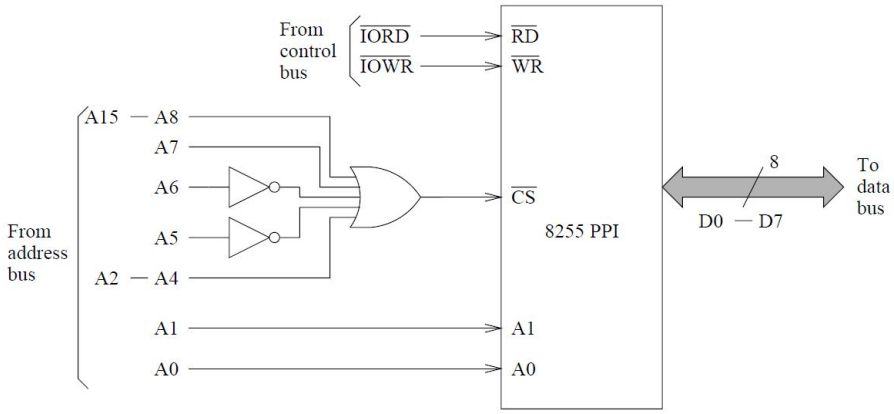
Пример – тастатура (2)

- Као контролер се (на пример) може користити чип 8255 који је развијен за употребу са 16-битним процесором 8086:
  - чип располаже са три 8-битна регистра (*PA*, *PB*, *PC*)
  - контролер испоручује код тастера путем регистра *PA*
    - битови *PA0-PA6* чине код тастера
    - бит *PA7* означава притисак (0) или отпуштање (1) тастера
  - нека су, нпр., ови регистри адресирани од адресе 60*H*:

8255 register	Port address
PA (input port)	60H
PB (output port)	61H
PC (input port)	62H
Command register	63H

Пример – тастатура (3)

- Пресликавање магистрала адресе и података је слично као у случају меморијских модула
- Основна разлика је у томе што се користе контролни сигнали за рад са У/И портовима *IORD* и *IOWR* уместо сигнала за рад са меморијом
  (нису приказане везе контролера и тастатуре)



Пренос података

- Пренос података између "система" и УИ уређаја подразумева размену података између меморије (или регистара) и УИ уређаја
- Пренос података чине две фазе:
  - фаза преноса података
  - фаза обавештавања о крају
- Неки аутори додају као посебну фазу
  - иницијализацију преноса података

Фаза преноса података

- Током ове фазе се преносе подаци између меморије и УИ уређаја
- Пренос се остварује путем
  - *програмираног У/И* или
  - *непосредног приступа меморији* (енгл. *direct memory access - DMA*)

Универзитет у Београду - Математички факултет

Фаза обавештавања о крају

- Током ове фазе процесор се информише да је пренос података довршен
- Информисање се остварује путем
  - *система прекида* или
  - *програмираног У/И*

Универзитет у Београду - Математички факултет

Пренос података (2)

- Три значајне технике:
  - програмирани У/И
  - непосредан приступ меморији (*DMA*)
  - система прекида
- Уобичајено:
  - ако се подаци преносе путем *DMA*, онда се обавештавање о крају одвија путем система прекида
  - ако се подаци преносе путем програмираног У/И, онда се и обавештавање о крају остварује истим путем

Универзитет у Београду - Математички факултет

Илустрација техника

- На примеру радника и шефа:
  - програмирани У/И
    - шеф задаје посао
    - “непрестано” проверава да ли је посао довршен
    - када је довршен, шеф узима резултат рада и ради даље
  - непосредан приступ меморији (*DMA*) / систем прекида
    - шеф задаје посао раднику и наставља са својим послом
    - радник самостално обавља посао
    - када заврши посао, радник обавештава шефа о завршетку
    - шеф реагује на обавештење и затим наставља са својим послом

Универзитет у Београду - Математички факултет

Програмирани У/И

- У основи програмираног У/И је петља у којој се чека да се доврши задати посао да би се могло наставити са радом
  - чекање се изводи кроз вишеструко проверавање (узорковање, тестирање) да ли је уређај достигао очекивано стање

Програмирани У/И – пример

- Пример тастатуре:
  - очитава се регистар *PA*
  - ако бит 7 има вредност 1, значи да није притиснут тастер, па се понавља претходни корак
  - ако бит 7 има вредност 0, значи да је притиснут тастер и прекида се петља
  - затим се прочитани код тастера може даље употребљавати, тј. преводити у код карактера

Програмирани У/И – пример (2)

- Исечак програмског кода:

```
        ...
key_up_loop:
        in      AL,60H
        test    AL,80H
        jnz     key_up_loop
        ...
```

Универзитет у Београду - Математички факултет

Литература

- *Sivarama Dandamudi*, ***Fundamentals of Computer Organization and Design***, *Springer, 2002.*
- *Margo Seltzer*, ***Operating Systems Course***, *http://www.eecs.harvard.edu/~margo/cs161/syllabus.html*
- *Ненад Митић*, ***Основи рачунарских система***, *Математички факултет, 2002.*

Универзитет у Београду - Математички факултет