**КОНТРОЛНО 2**

Сбирщина от въпроси/отговори през годините:

**1. Какво е характерно за регистрите с общо предназначение при 64-битов режим?**

Регистрите за общо предназначение на IA-32 и Intel-64 архитектурата са EAX, EBX, ECX, EDX, EBP, ESP, ESI и EDI. Тези регистри се използват за съхранението на операндите на логическите и аритметични операции. Те също така могат да бъдат използвани за операнди на изчисленията на адрес (освен че ESP не може да бъде използван като индексен операнд).

**2.Каква е ролята на завършващата част?**Завършващата част се грижи резултатът от изчисленията да съответства на реда на постъпване на инструкции и че състоянието на системата се обновява правилно.

**3. Защитеният режим (protected mode).** Той се характеризира с апаратно предотвратяване на достъпа на дадена програма извън границите на заделената и памет. Първата версия на защитения режим се появи в 16-битовия процесор 80286. В съвременните операционни системи се използва защитеният режим на работа. Освен това сегментирането се използва много ограничено, като защитата и управлението на паметта стават основно с помощта на механизма на страницирането. Именно поради тази причина, в 64-битовото разширение на x86 архитектурата AMD64(x64), сегментирането е "орязано" до минимум.

**4. Обяснете механизма за обновяване на страница от диска**

Мениджърът на паметта използва информацията от таблиците за да намери неизползвана страница физическа памет (такава, към която скоро не е имало обръщения) и я записва в резервирана област на диска, наречена swap файл. После уведомява централния процесор за необходимостта да прочете изискваната страница от по-бавната памет (файл на диска или swap файл). Накрая MMU установява съответствието между виртуалната и физическата страница и обновява таблицата с тази информация.

5. **Опишете последователността от стъпки, която се изпълнява при прекъсване при IA32**

Аварийният цикъл се състои в следното:

1. Запазва се векторното състояние на текущата програма;   
2. Определя се видът на аварийното събитие или чрез сигнал от линиите за прекъсване, или като код за линиите за данни към процесора;   
3. По кода на вида прекъсване се извлича от оперативната памет нов вектор на състоянието, т.е. се прави зареждане с ново съдържание на централния процесор, включително и program counter-а. Старите регистри отиват в друга област – старо състояние. Новото и старото състояние са области в операционната памет;   
4. Преход към нормален цикъл. В резултат на новото съдържание процесорът тръгва от нова команда.

6. **Обяснете групите, на които се делят изключенията**

1) Грешки (*faults*). Грешките се откриват и обслужват непосредствено преди следващата команда. Най-често се явяват в системата на виртуалната памет, когато е заявен адрес към несъществуваща страница или сегмент. Съобщението подготвя процесора за рестарт на командата, генерирала грешката, т.е. адресът за връщане от обслужване на прекъсването представлява старото съдържание на програмния брояч. В същото време операционната система търси нужната страница или сегмент.

2)Капани (*traps*). Съобщението за капан, в който е попаднал хода на изчислителния процес, се генерира непосредствено в края на текущата и преди следващата команда. Капаните това са контролни точки, които могат да бъдат залагани умишлено от потребителя, с цел контрол на изчислителния процес, обикновено при тестване на софтуера, чрез вмъкване в кода на програмата на еднобайтова команда INT 3.

3) Повреди (аварийни прекъсвания) (*aborts*). При такива ситуации не винаги се разполага с адреса на командата, по време на която е настъпило изключението. По тази причина не винаги е възможен рестарт на командата и следователно на хода на изчислителния процес. Типични изключения от този вид са различни апаратни грешки, откривани от контролиращи логически схеми, или при противоречиви (недопустими) стойности в системните таблици.

**7. GDTR (****Global Descriptor Table Register)** – този регистър сочи към началния адрес на глобална дескрипторна таблица (Global Description Table), която се използва при сегментирания модел на адресация.

**8. Какво е предназначението на сегментните селектори**

Сегментните селектори са 16 битови идентификатори на сегмента. Старшите 13 бита се използват за определяне на отместването в глобалната или локалната дескрипторни таблици. Трите младши бита имат по-специфично предназначение.При сегментацията адресът се получава от 16-битов сегментен регистър и 32-битово отместване.

9. **Как работи изпълняващата част?**Изпълняващата част пренарежда микрооперациите по такъв начин, че тези чиито операнди са готови (и има налични изчислителни ресурси) да се изпълнят възможно по - скоро. Изпълняващата част може да обработи (issue) няколко микрооперации за един цикъл.

10. **Прилики и разлики между прекъсване и изключение**

Иключението възниква в тялото на текущата програма, където работи процесорът, и се обработва в нея. Обикновено това събитие е породено от невъзможността на процесора да извърши текущата команда. Например в резултат на прехода program counter (програмният брояч) да сочи някъде извън реалната памет или да се окаже, че командата е наред, но има адрес на една от данните, която сочи извън паметта. Други такива команди са: опит за делене на 0; умножение на две големи числа, при което битовете не достигат; опит на програмата да пише в област, в която няма право и др. Прекъсванията възникват винаги извън тялото на текущия процес, т.е. възникват от входно-изходните устройства, схемите на централния процесор или от други схеми. Тези събития се обработват от операционната система, извън тялото на текущия процес. И в двата случая обработката на аварийната ситуация се извършва първо от апаратурата, като централният процесор преминава в авариен цикъл на изпълнение, и след това от софтуерен модул, който обработва данните с цел да намали щетите от възникналото събитие.

11. **Как се осъществява превключване между задачите**

Операционната система има модул, наречен loader (зареждащ модул), който, получавайки съответна команда, извършва зареждане на задача за предстоящо изпълнение, като при това зареждане в оперативната памет за тази задача се създава споменатия вече (TSS). Освен този модул, при управлението на задачите участва и още един – планиращ модул (модул за планиране), който решава коя от задачите, заредени в оперативната памет, да активира. Той работи на следния принцип:  
1) Издава команда към процесора за задействането му след даден квант време.  
2) Обхожда таблицата на задачите, взима първата, която е в режим на готовност и я стартира, след което продължава по следния начин – записва състоянието на предишната и започва да работи по новата задача.  
3) След като изтече квантът време, процесорът е бил задействан от таймер и този вид прекъсване води до активиране на планиращия модул, който отново издава команда за задействане на процесора и отново сканира таблицата. Има статистическа информация, която пренарежда таблицата, така че да не се позволи една и съща задача да се изпълнява многократно постоянно.

12. **Какво представляват и какви са предимствата на пакетираните SIMD типове данни**

SIMD типовете данни са MMX и XMM. ММХ типът е разработен, за да ускори мултимедийните програми и програмите за комуникация като включват нови инструкции и типове данни които позволяват на програмите да достигат ново ниво на производителност. ММХ типът е разработен като множество от integer инструкции които могат да бъдат приложени към нуждите на голямо разнообразие от мултимедия.

13. **Каква информация се съдържа в IDT?**

В Protected mode за всяка програмна част, която се записва някъде, се създава един дескриптор, който представлява таблица на прекъсванията IDT (Interrupt Descriptor Table) и за нея има един регистър, който сочи началото на тази таблица. Когато дойде код на прекъсване, той се разглежда като код на отместване в таблицата, откъдето се определя началото на съответния програмен модул.

14. **По какво се различават изключенията от тип faults и abort**

За faults се знае коя инструкция ги е причинила и програмата или процесът могат да бъдат рестартирани без загуба, докато при

Aborts това може да не е ясно и следва прекратява на програмата или процесът.

Изключенията от тип грешки (fault)като цяло могат да бъдат коригирани и за това след като се случат и бъде изпълнена обработващата програма, се стартира отново същата програма (която е генерирала грешката), като се счита, че грешката вече е коригирана (от обработващата програма).

15. **Каква е функцията на челната част (front-end)**

Челната част (Front End) на вътрешната архитектура на IA-32 е един от основните компоненти на процесорите Intel, както може да се види на фигурата. Тя съдържа кеша за проследяване, който е кеш за инструкции. При липса в кеша за проследяване в непосредствена близост до кеша за проследяване е разположен ROM c микрокода, който съхранява микрооперациите на комплексните инструкции. За комплексните инструкции кешът за проследяване изпраща указател към ROM, по който се извлича съответната последователност от микрооперации, имплементиращи тази инструкция.

**1.Какво е характерно за регистрите с общо предназначение при 64-битов режим ?**

При Intel-64 архитектурата са EAX, EBX, ECX, EDX, EBP, ESP, ESI и EDI. Тези регистри се използват за съхранението на операндите на логическите и аритметични операции.

**2.Каква е ролята на завършващата част на вътрешната архитектура?** Грижи се за резултатът от изчисленията да съответства на реда на постъпване на инструкции и че състоянието на системата се обновява правилно.

**3. Защитеният режим на процесора.**

Той се характеризира с апаратно предотвратяване на достъпа на дадена програма извън границите на заделената и памет.

**4. Обяснете механизма за обновяване на страница от диска.**

Мениджърът на паметта използва информацията от таблиците за да намери неизползвана страница физическа памет и я записва в резервирана област на диска, наречена swap файл. После уведомява централния процесор за необходимостта да прочете изискваната страница от по-бавната памет. Накрая MMU установява съответствието между виртуалната и физическата страница и обновява таблицата с тази информация.

5. **Опишете последователността от стъпки, която се изпълнява при прекъсване при IA32**

1. Запазва се векторното състояние на текущата програма;   
2. Определя се видът на аварийното събитие или чрез сигнал от линиите за прекъсване, или като код за линиите за данни към процесора;   
3. По кода на вида прекъсване се извлича от оперативната памет нов вектор на състоянието. Старите регистри отиват в друга област – старо състояние. Новото и старото състояние са области в операционната памет;   
4. Преход към нормален цикъл. В резултат на новото съдържание процесорът тръгва от нова команда.

6. **Обяснете групите, на които се делят изключенията.**

1) Грешки (*faults*). Грешките се откриват и обслужват непосредствено преди следващата команда. Най-често се явяват в системата на виртуалната памет, когато е заявен адрес към несъществуваща страница или сегмент.

2)Капани (*traps*). Съобщението за капан се генерира непосредствено в края на текущата и преди следващата команда. Капаните това са контролни точки, които могат да бъдат залагани умишлено от потребителя, с цел контрол на изчислителния процес

3) Повреди (аварийни прекъсвания) (*aborts*). При такива ситуации не винаги се разполага с адреса на командата, по време на която е настъпило изключението. Затова не винаги е възможен рестарт на командата и следователно на хода на изчислителния процес. (апаратни грешки)

**7. GDTR-Обяснете използването на регистъра.**

(Global descriptor table register) Този регистър сочи към началния адрес на глобална дескрипторна таблица, която се използва при сегментирания модел на адресация.

**8. Какво е предназначението на сегментните селектори?**

Те са 16 битови идентификатори на сегмента. Старшите 13 бита се използват за определяне на отместването в глобалната или локалната дескрипторни таблици.

9. **Как работи изпълняващата част?**Изпълняващата част пренарежда микрооперациите по такъв начин, че тези чиито операнди са готови да се изпълнят възможно по - скоро.

10. **Прилики и разлики между прекъсване и изключение.**

Иключението възниква в тялото на текущата програма и се обработва в нея.

Прекъсванията възникват винаги извън тялото на текущия процес.

И в двата случая обработката на аварийната ситуация се извършва първо от апаратурата, като централният процесор преминава в авариен цикъл на изпълнение, и след това от софтуерен модул, който обработва данните с цел да намали щетите от възникналото събитие.

11. **Как се осъществява превключване между задачите?**

При управлението на задачите освен зареждащия модул (loader) учатсва още един модул за планиране, който решава коя от задачите, заредени в оперативната памет, да активира.

12. **Какво представляват и какви са предимствата на пакетираните SIMD типове данни?**

SIMD типовете данни са MMX и XMM. ММХ типът е разработен, за да ускори мултимедийните програми и програмите за комуникация като включват нови инструкции и типове данни които позволяват на програмите да достигат ново ниво на производителност. ММХ типът е разработен като множество от integer инструкции които могат да бъдат приложени към нуждите на голямо разнообразие от мултимедия.

13. **Каква информация се съдържа в IDT?**

Таблица на векторите на прекъсванията (Interrupt Descriptor Table). Тя има 256 елемента. Векторите на прекъсванията не са дескриптори на сегменти, а са входове към такива

14. **По какво се различават изключенията от тип faults и abort**

За faults се знае коя инструкция ги е причинила и програмата или процесът могат да бъдат рестартирани без загуба, докато при Aborts това може да не е ясно и следва прекратява на програмата или процеса.

15. **Каква е функцията на челната част (front-end)**

Тя съдържа кеша за проследяване, който е кеш за инструкции. За комплексните инструкции кешът за проследяване изпраща указател към ROM, по който се извлича съответната последователност от микрооперации, имплементиращи тази инструкция.

**16. Опишете причините за възникване и стратегиите за предотвратяване на „Мехури“ в инструкционните конвейери.**

Всяка команда за преход води до нарушаване на естествената последователност от постъпващи команди в конвейера. Когато не се получава резултат на всеки такт, това състояние да се нарича появяване на мехурчета в работата на конвейера. Преодоляване на проблема са:

• предварителен избор на адреса на разклонение;

• използване на няколко потока команди;

• прогнозирано разклонение;

• отложено разклонение.

**17.Какво представлява сегментната преадресация? Посочете предимства и недостатъци.**

* При сегментната преадресация логическия адрес (състоящ се от сегментен селектор и сегментно отместване) се преобразуват в линеен адрес.

**18.Какво представлява виртуалната памет и как се поддържа при архитектурата Intel-64?**

Виртуална памет е система, при която всичката физическа памет се контролира от операционна система. Когато една програма се нуждае от памет, то тя иска разрешение от операционната система.

**19.Каква информация се съдържа в Task Register и за какво се използва?**

Task Register (TR) - съдържа 16 битов сегментен селектор, базов адрес, размер на сегмента и атрибути на Task State Segement (TSS) на текущия процес.  
1) Издава команда към процесора за задействането му след даден квант време.  
2) Обхожда таблицата на задачите, взима първата, която е в режим на готовност и я стартира, след което продължава по следния начин – записва състоянието на предишната и започва да работи по новата задача.  
3) След като изтече квантът време, процесорът е бил задействан от таймер и този вид прекъсване води до активиране на планиращия модул, който отново издава команда за задействане на процесора и отново сканира таблицата

**20. Опишете полетата обособяващи формата на инструкция.**

- Инструкцията съдържа: незадължително поле префикс, код на операцията (до три байта), спецификатор формиращ адрес (не е задължителен) и съдържа ModR/M байт и понякога а SIB (Scale-Index-Base) байт, отместване (незадължително) и поле „непосредствен“ (незадължително).

---от лекции до тук

**21. Опишете основните класове-инструкции при IA32**

• Real mode – позволява на процесора да адресира реални адреси от паметта, като максималното адресно пространство е 1М байт

• Protected mode –нормалният режим на работа. Той позволява използването на сегментиране, виртуална памет и други техники за защита

• SMM (System management mode) mode – всички обичайни операции се прекратяват и с висок приоритет се изпълнява код, които обикновено е записан на постоянна памет.

**22. Какво е характерно за адресното пространство?**

- При многозадачните компютърни системи, обикновено не е икономически ефективно, цялото линейното адресно пространство да се съдържа във физическата памет. Необходимо е да се виртуализира линейното адресно пространство, а това се осъществява от процесора с използване на страницирането.

**23. Обяснете по какъв начин се изчислява адреса**

- Всички сегменти се намират в т.нар. линейно адресно пространство на процесора, а за да се адресира байт в даден сегмент, е необходим логически адрес, които се състои от сегментен селектор и отместване.

**24. Опишете Сегментнит регистри и предназначението им.**

Сегментните регистри са 6. Тяхното предназначение е да съхраняват сегментен селектор.

Има различни сегменти.

CS – (Code Segment) – съдържа базовия адрес на сегментът, в който се съхранява програмата.

DS – (Data Segment) – съдържа адреса на сегмента, в който се съхраняват данните.

SS – (Stack Segment)

ES – (Extra Data Segment) – допълнителен сегмент за данни.

FS - Допълнителен сегмент за данни.

GS - Допълнителен сегмент за данни.

**25. Опишете Флаговите регистри.**

- съдържа информация за статуса, т.е. изпълнението на програмите

**26. Опишето полето Префикс(Prefix).**

- Чрез префикса може да се променя поведението на инструкцията, като например да се сменя сегмента, размера на думата, да се управлява цикличното изпълнение и използването на шината.

**27. Пояснете Rex-префикс, използване, приложение и т.н.**

- REX се използва за генериране на 64 битови операнди.

**28.Обяснете приложението на управляващите регистри.**

- Управляващите регистри се използват от операционната система.

CR0 - Съдържа флагове контролиращи режимите на работа и състоянието на процесора

CR1 — Запазен, но не се използва.

CR2 – Съдържа линейния адресът, който е причина за изключение от тип „page fault“.

**29. LDTR регистри?**

- Local Descriptor Table Register (LDTR) – съдържа 16 битов сегментен селектор, базов адрес, размер на сегмента и атрибути на дескриптора.

**30. Какви полета съдържа LDT(Local Description Table)?**

- Локалната дескрипторна таблица (Local Descriptor Table LDT) съдържа описание на сегменти, които са частни за дадена програма

**28. Обяснете структура на задачата при IA32**

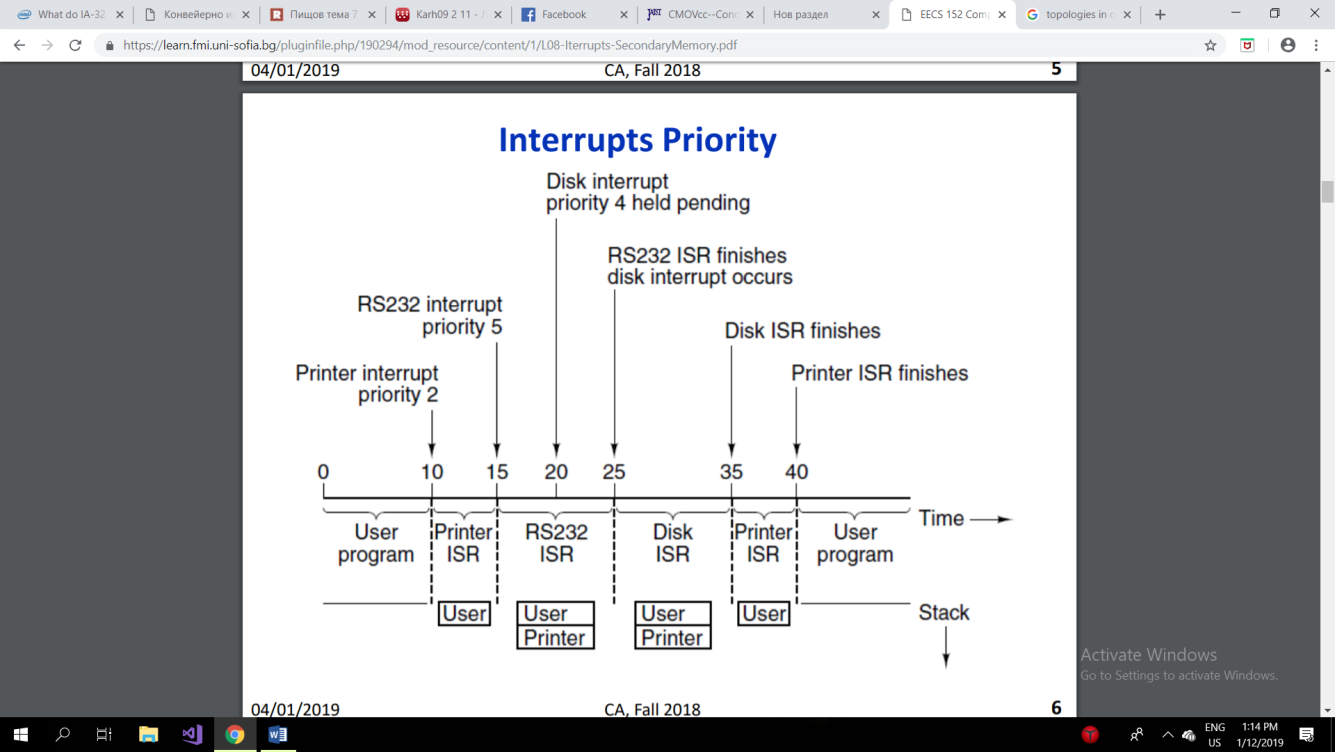
- Задачата се състои от пространство за изпълнение (execution space) и сегмент за състоянието на задачата (Task State Segment TSS).

**29. Каква е функцията на предварителното извличане?**

- Предварителното извличане на данните в кеша (Cache prefetching) е техника, която увеличава производителността, като копира данните от бавна памет в по-бърза, още преди да са необходими.

**Затворени въпроси:**

**1.Опишете реда на приоритет на прекъсванията?**

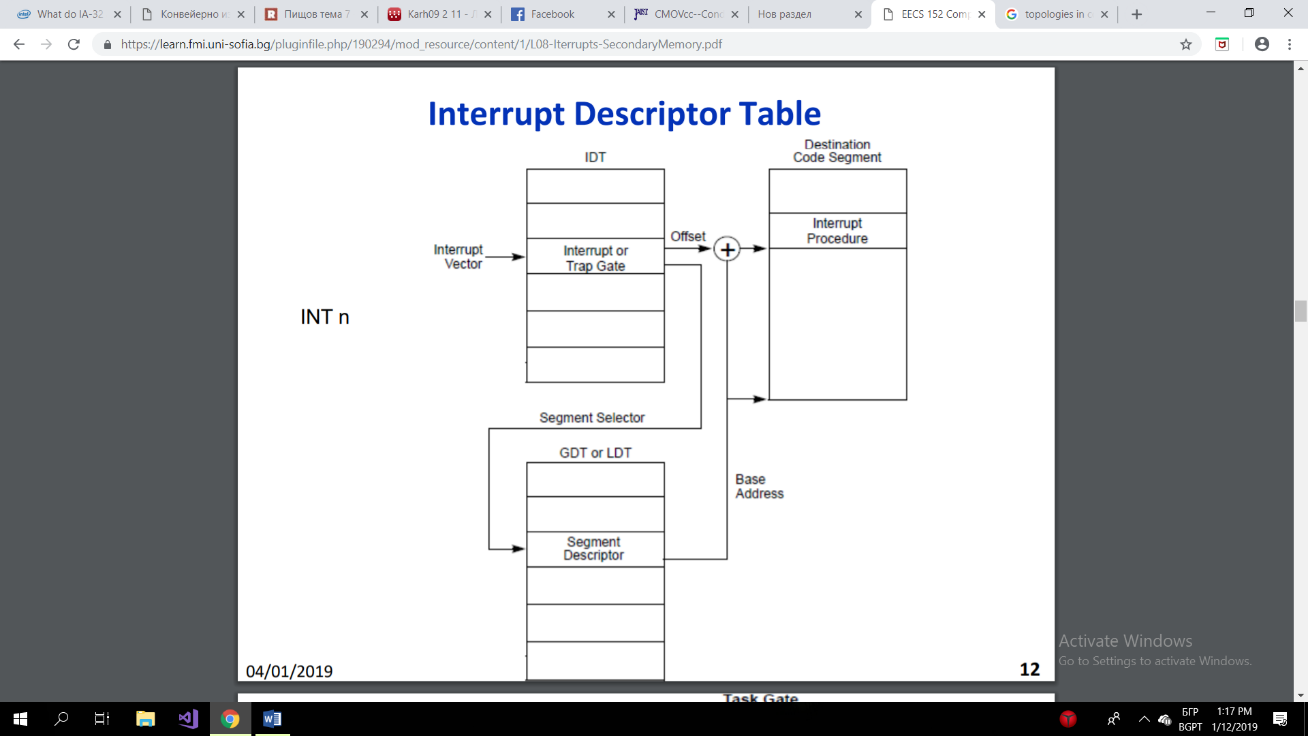


RS232 – 5 ;DISK - 4;Printer - 2;User - 0

**2.Как се нарича елементът?**

ЕX: IDT, GDT

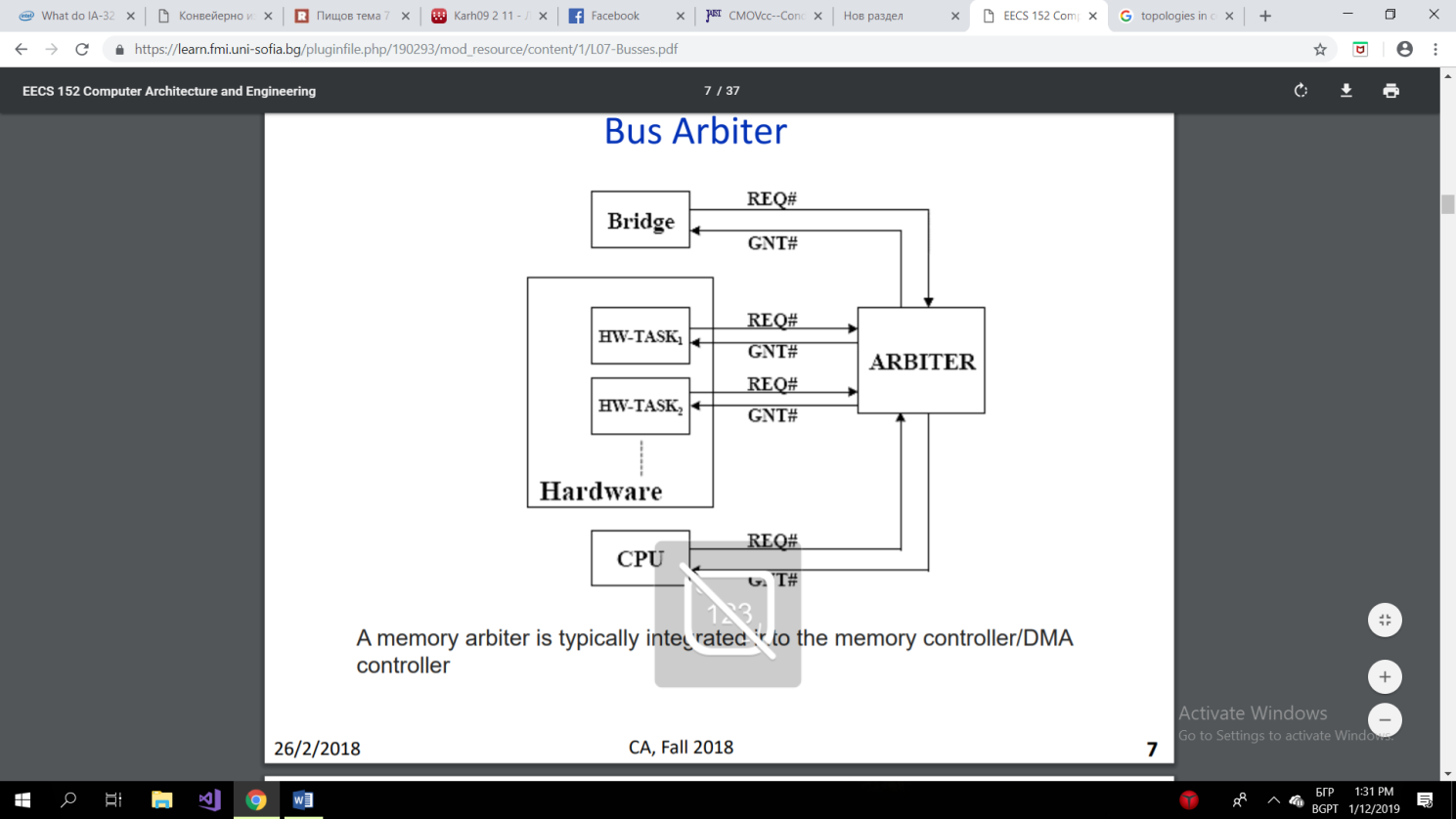
Or LTD



**3.Каква е целта на използване на CMOVZ?**

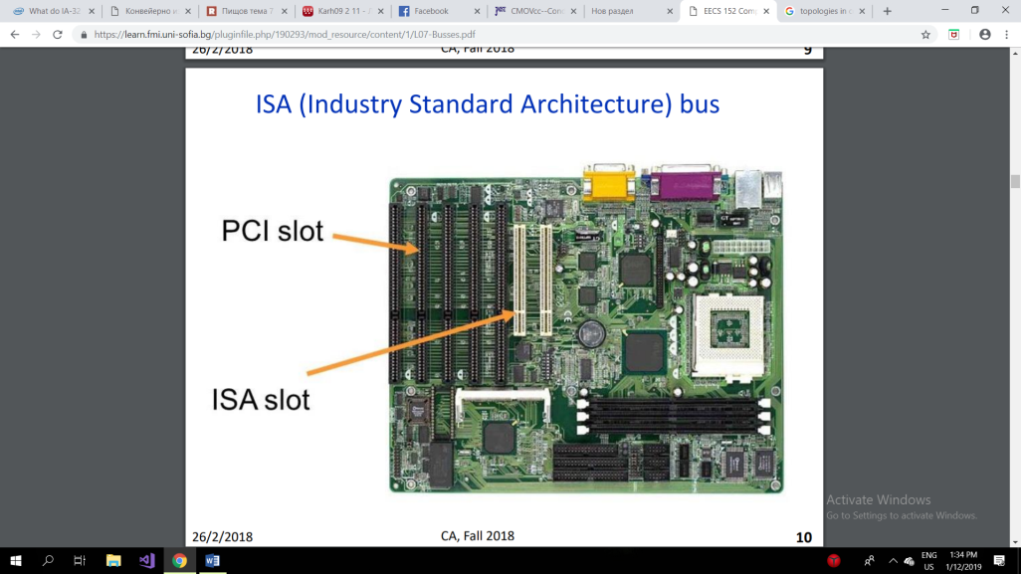
-редуциране на условните преходи.

**4.Кой е елементът на схемата? – Arbitter, Bridge, CPU**



EX:arbiter.

**5. Кой е показания интерфейс?**

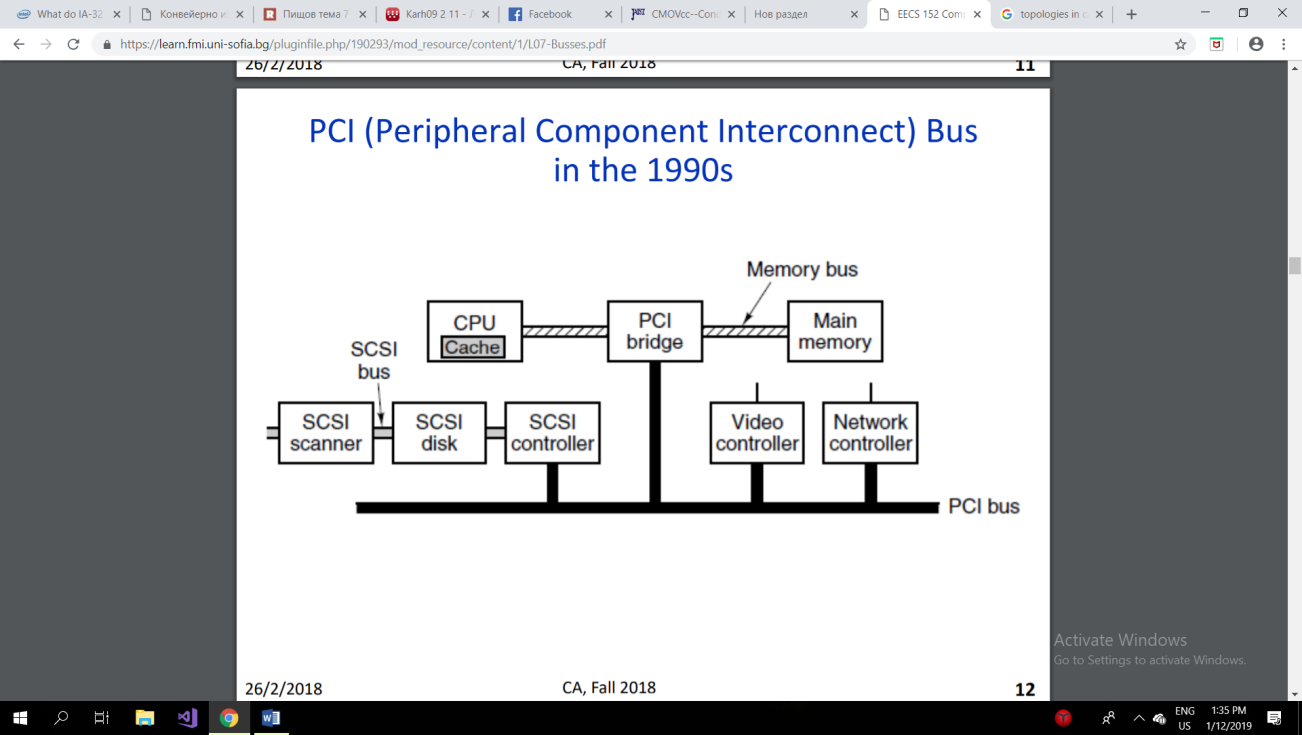


-Схемата е грешна

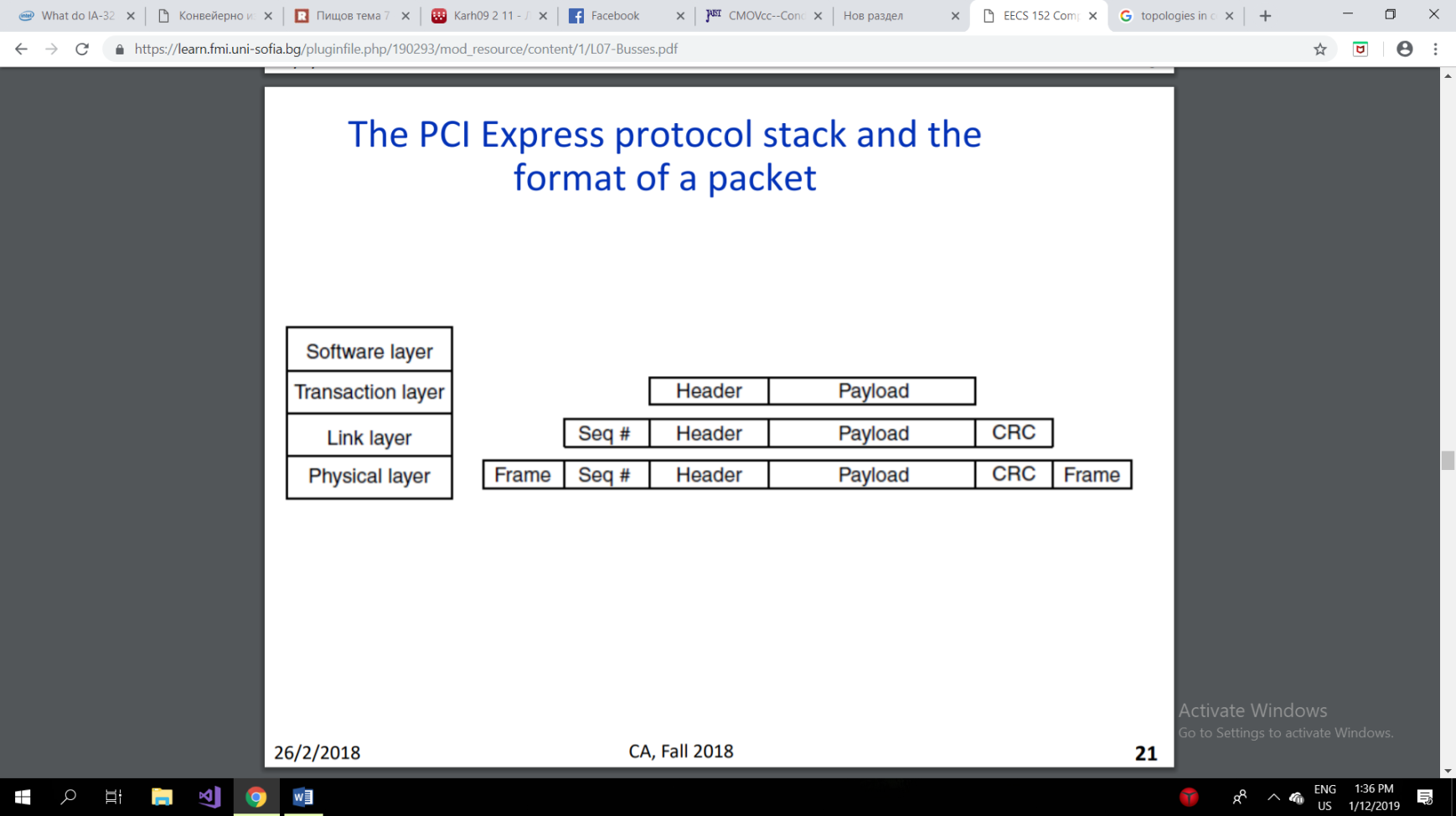
?. Кой от посочените компоненти осъществява връзката с вътрешна/външна шина

**?. Кои са имена на шини – USB, CAN, I2C**

**6.Кой е елементът? – PCI bridge, CPU, SCSI controller**

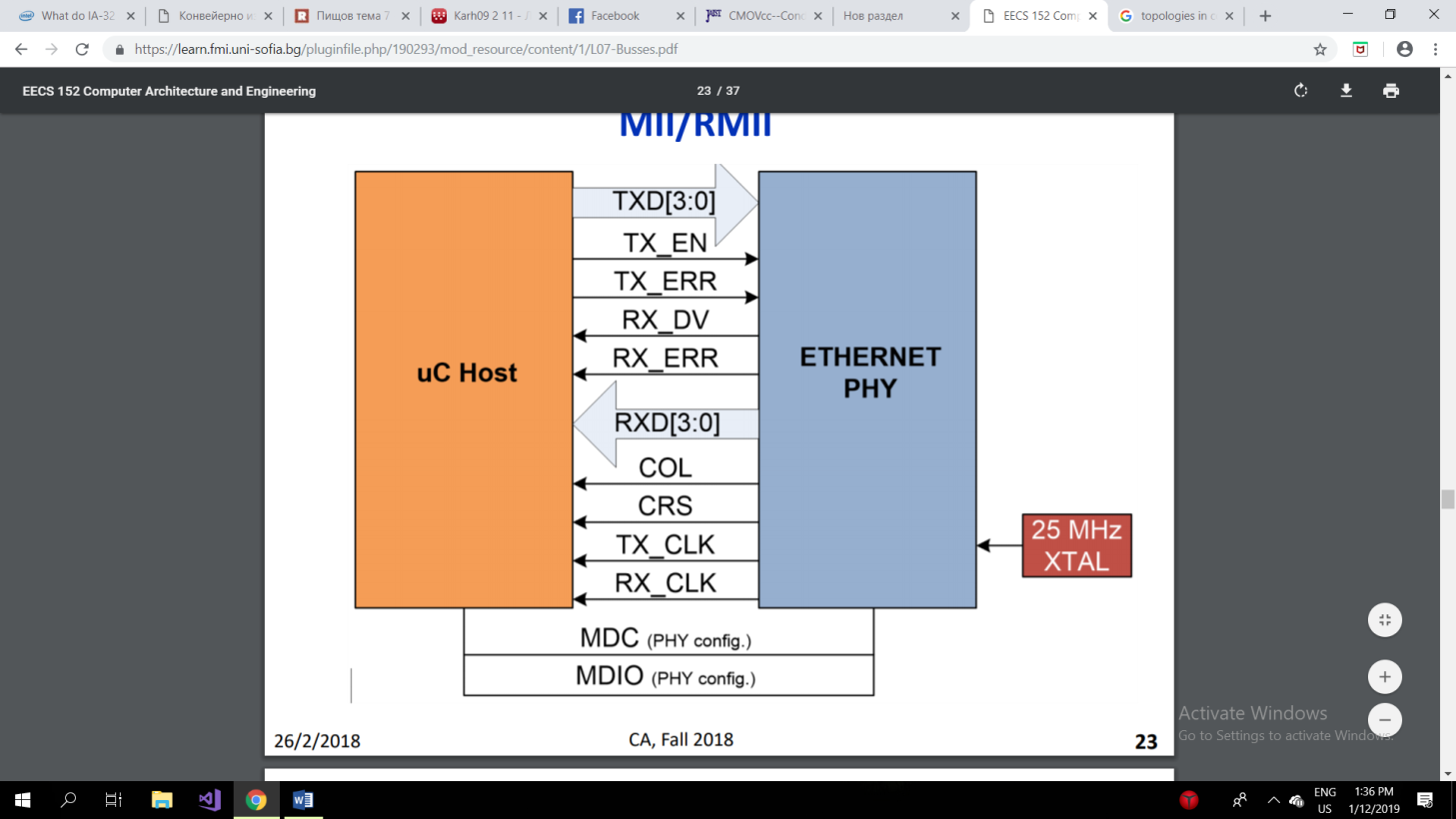


**7.Кое е липсващото ниво?**



**8.Защо посоката TXD e oт HOST към ETHERNET?**

**Кой е липсващия елемент – TXD, RXD**

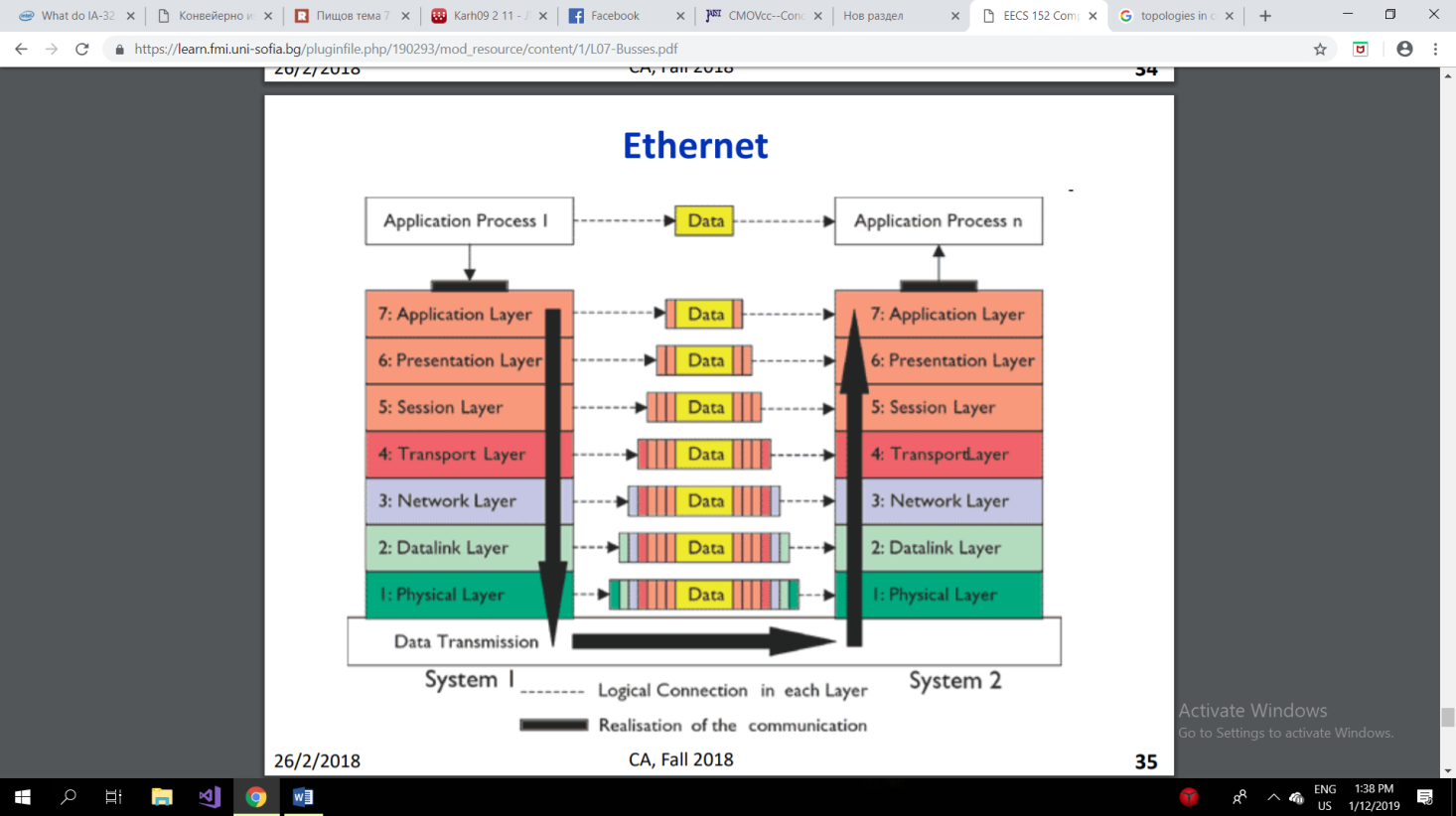


TXD е транзит. HOST предава данни.

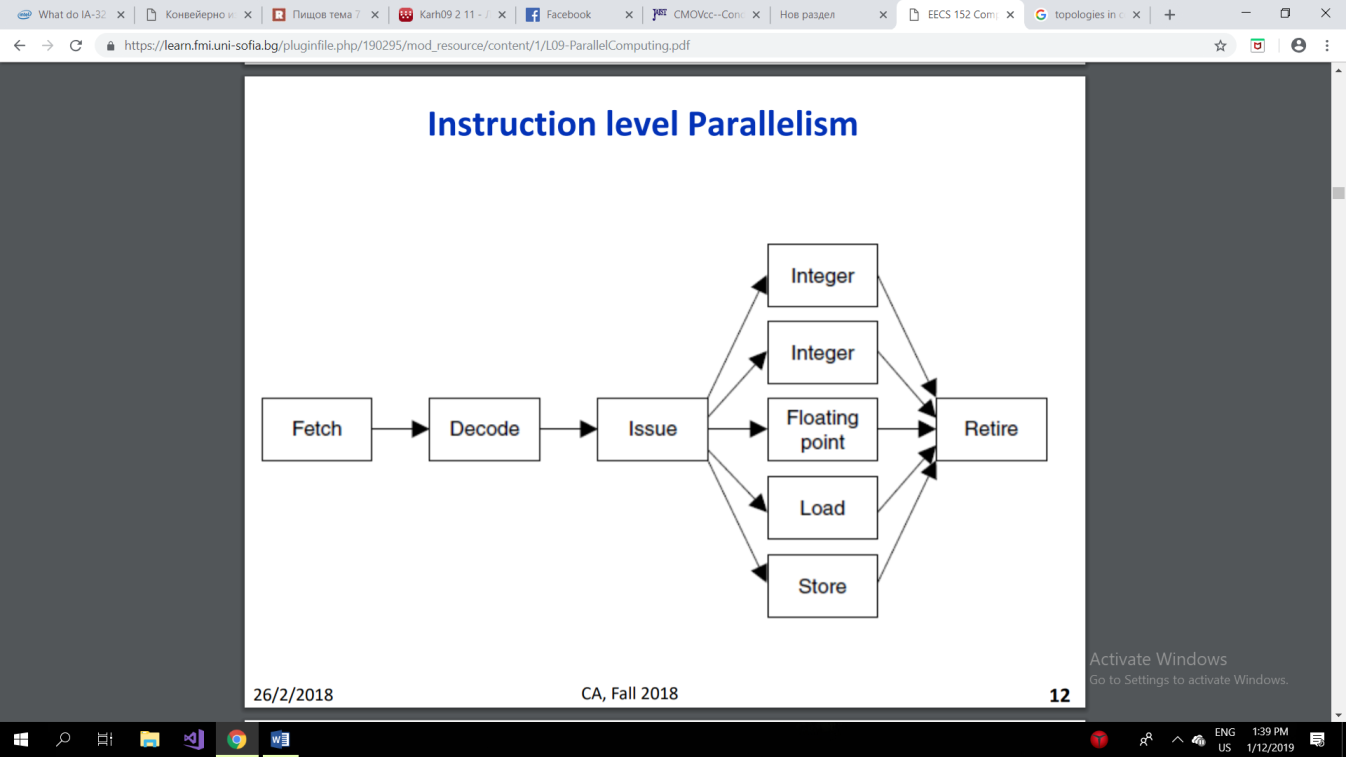
**9. Колко проводна е шината USB1/3**

- 4 за 1

**10. Кой е липсващия елемент(слой)?**



**12. Кой е липсващия елемент?// Кои ел. са разменени? – Decode - Fetch**



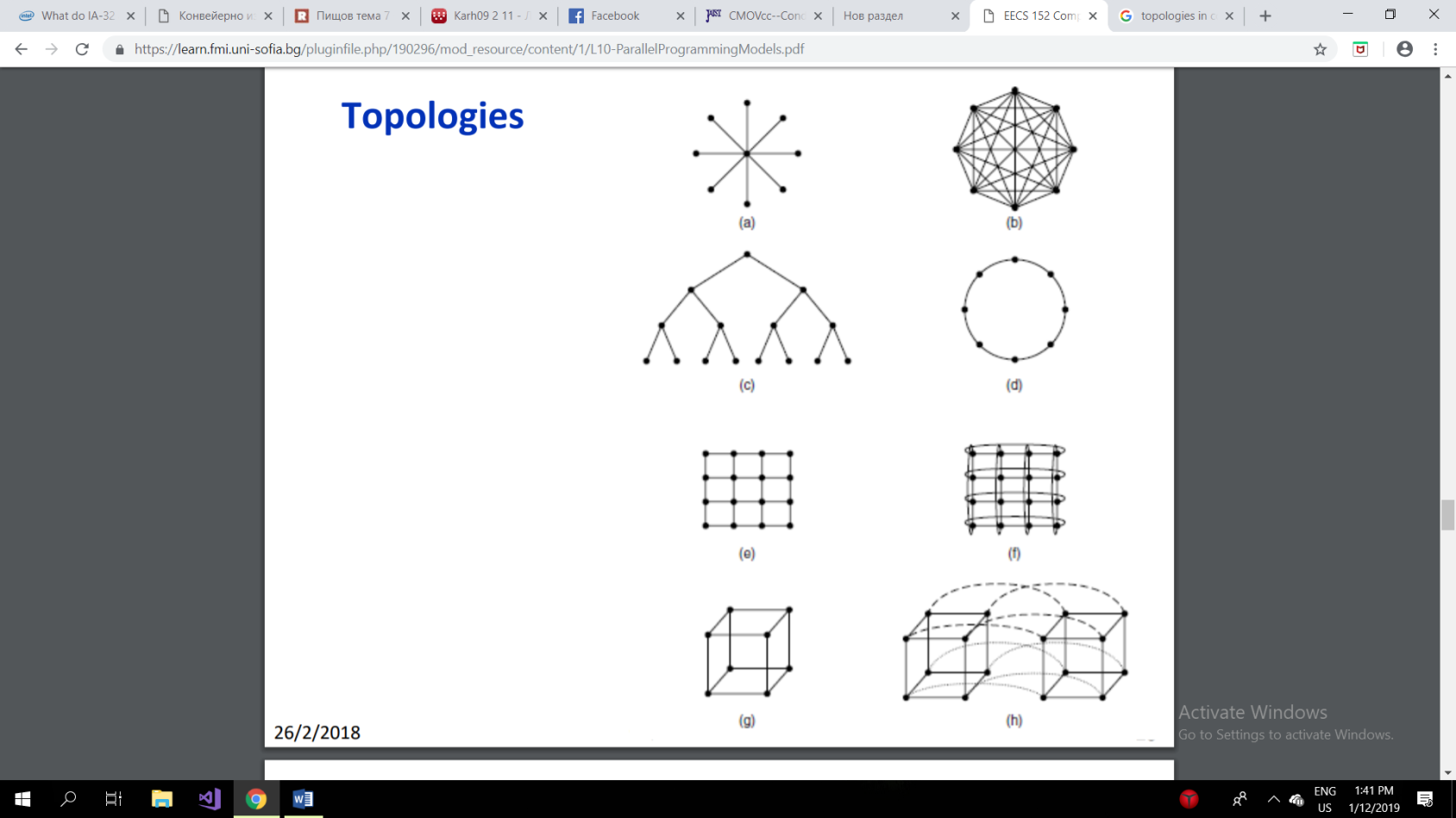
**14.Какво е Thread execution manager?**

- Операционната система поддържа управленска информация за процес в блок за управление на процеса

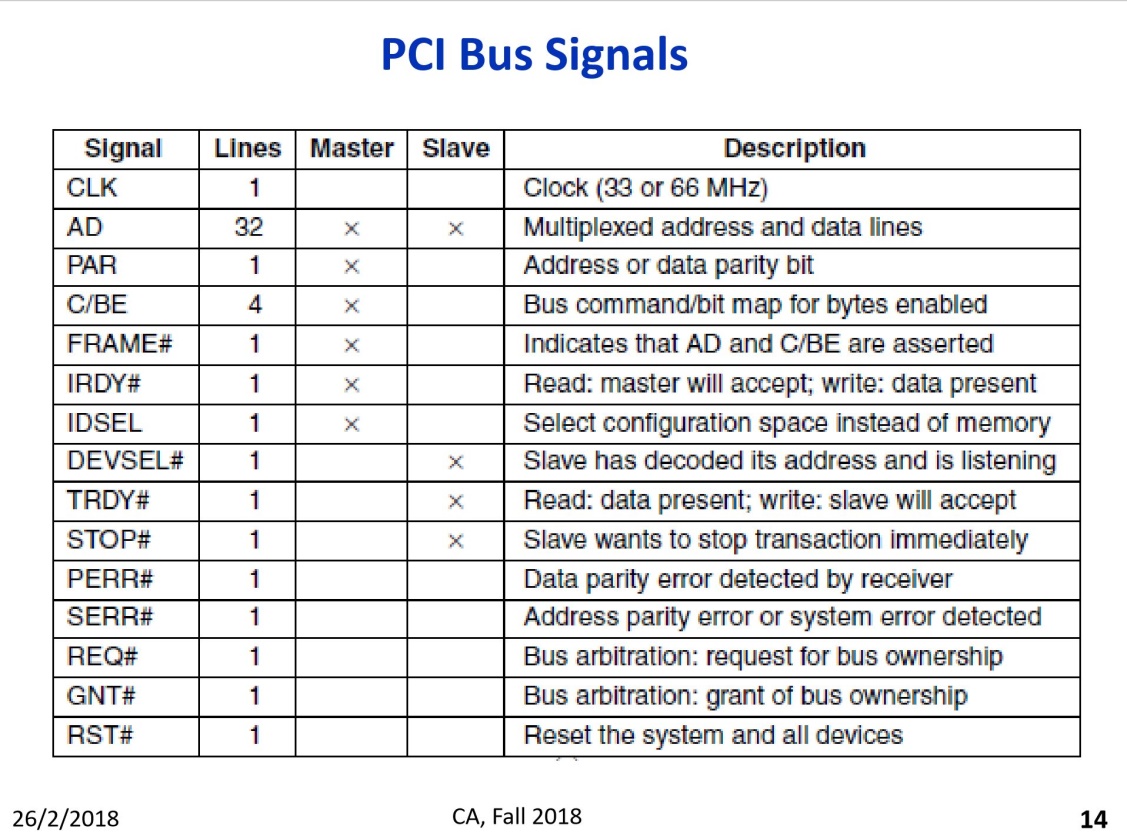
**15.Какво е средното закъснение в Hops?**

-Ако е кръг броя на всички елементи ги делим на 4, т.е. средното закъснение ще бъде 2.

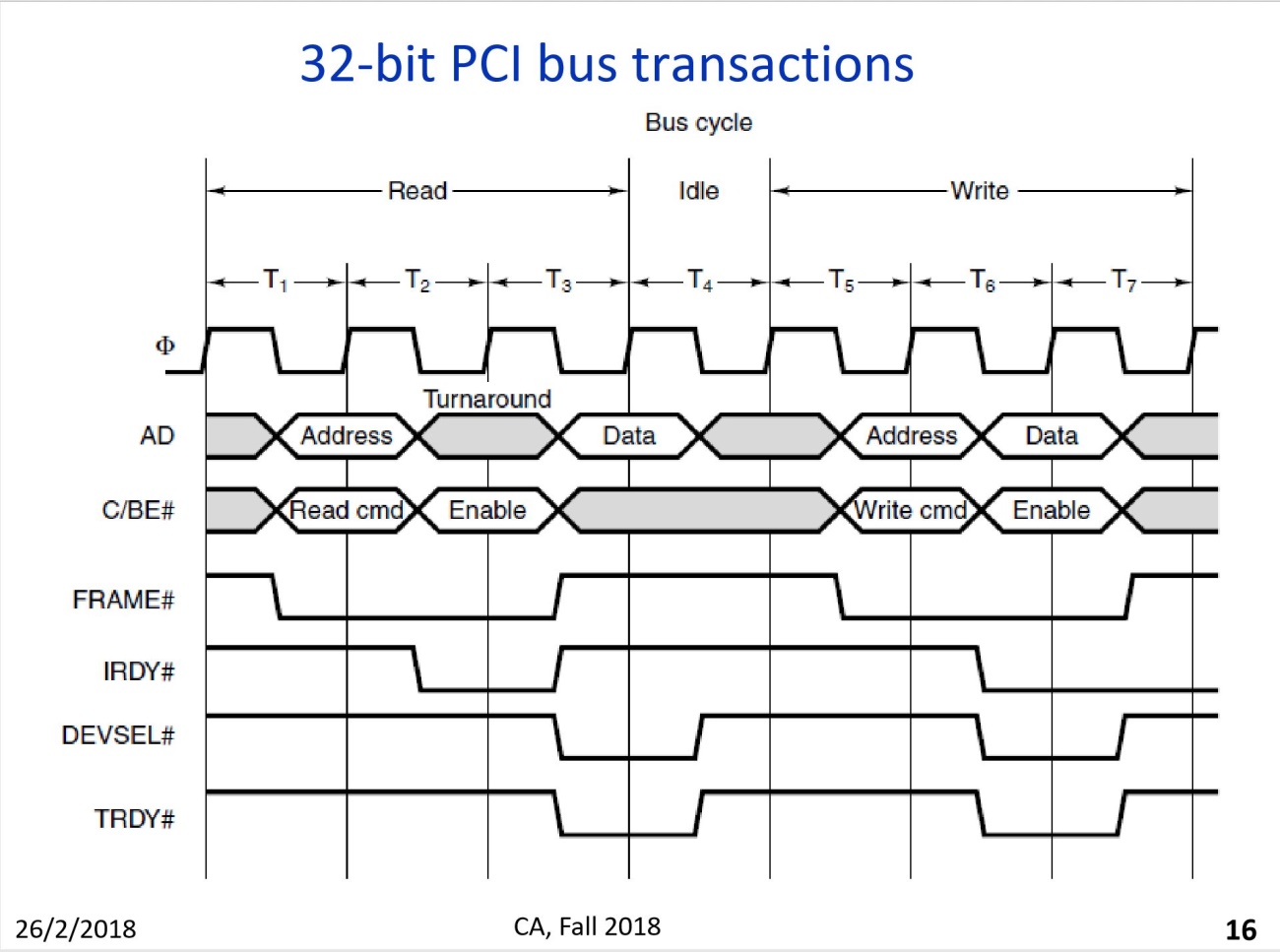
-Ако е линия броя на всички елементи ги делим на 2.



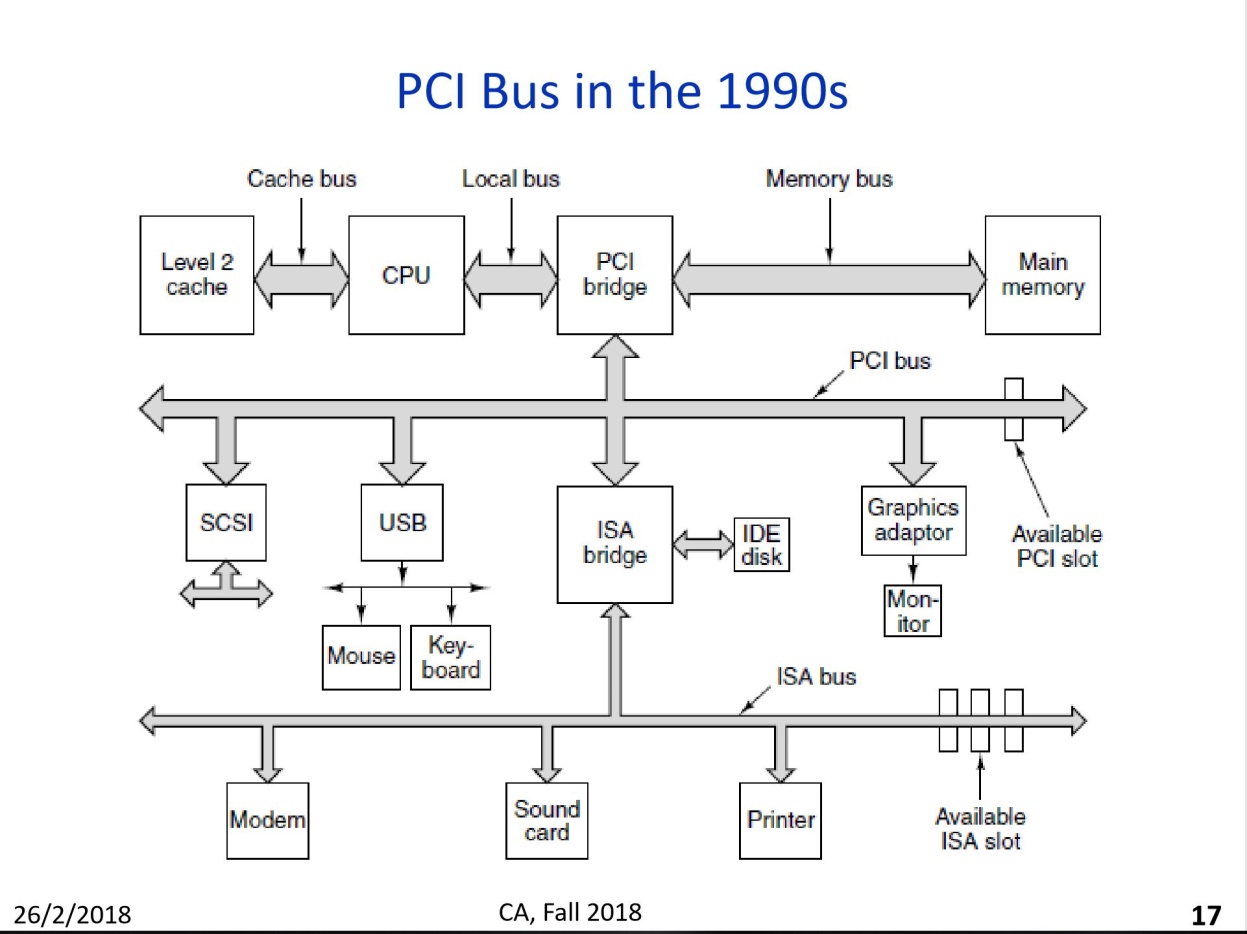
?. Какво означават REQ GNT AD RST CLK C/BE, Frame



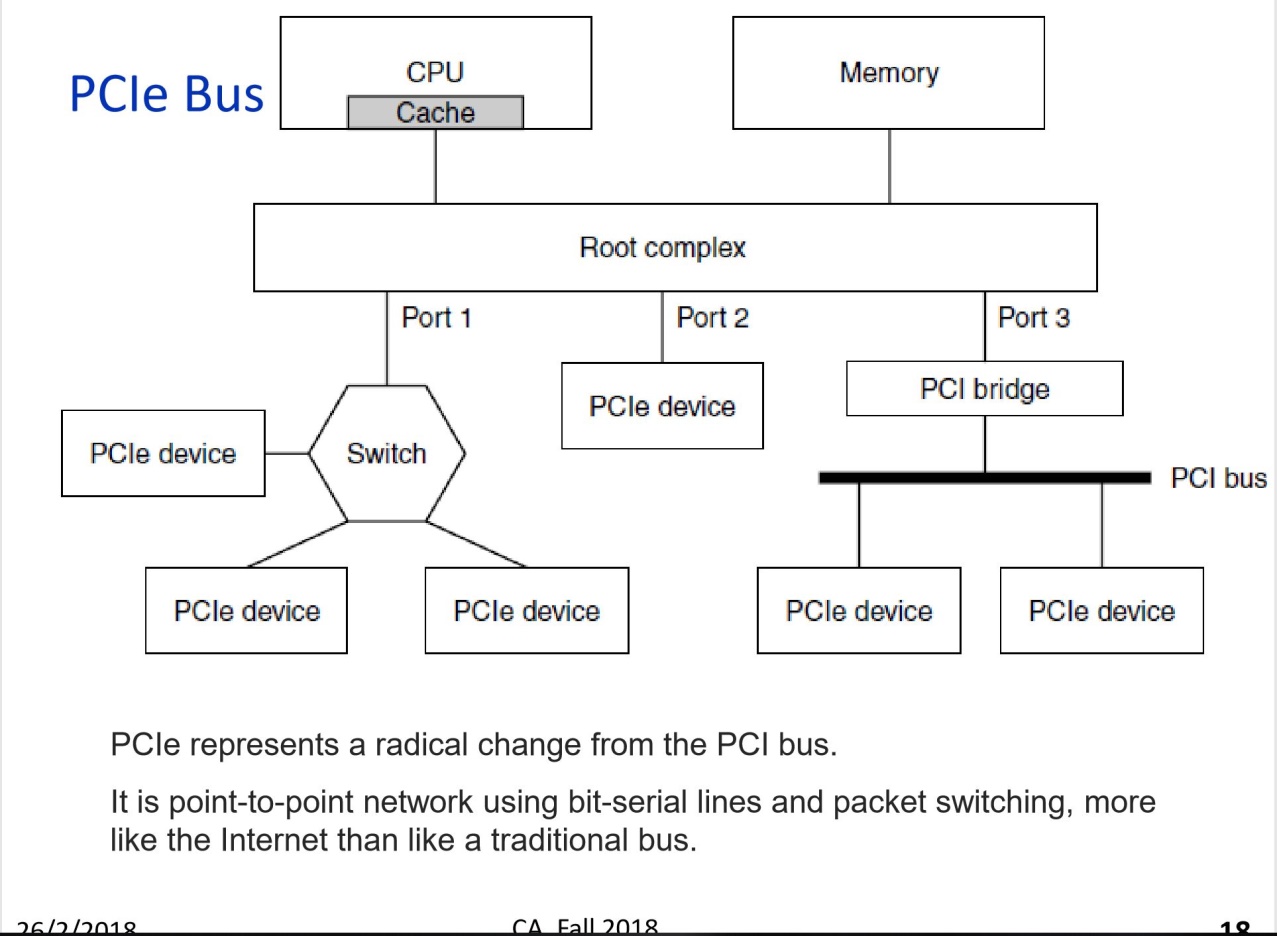
Дали е четене или писане пада се едната половина без надписи read/write ( 32-bit PCI bus transactions )



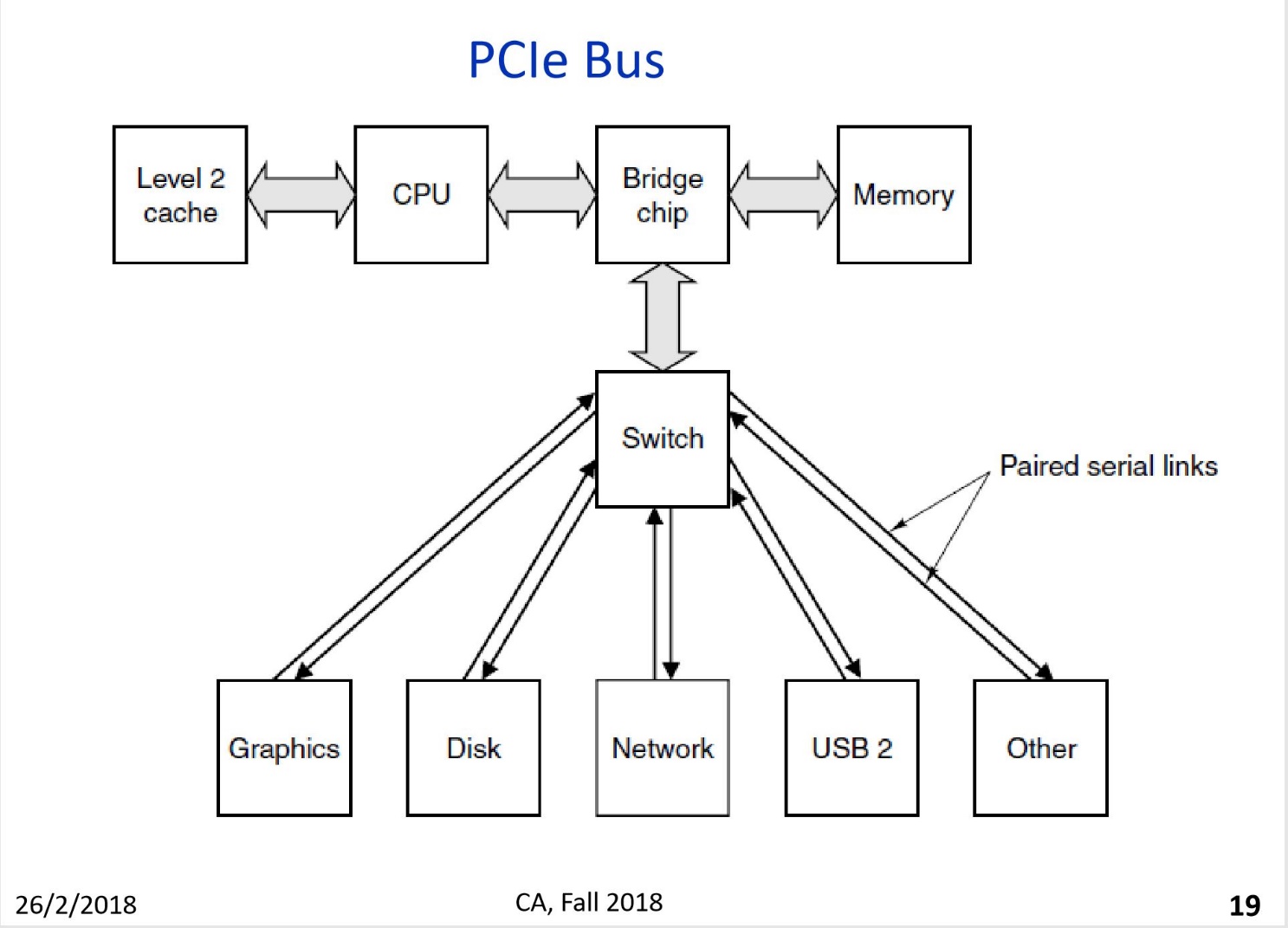
Кой е липсващия елемент – PCI bridge, CPU, USB, ISA bridge ( PCI Bus in the 1990s )

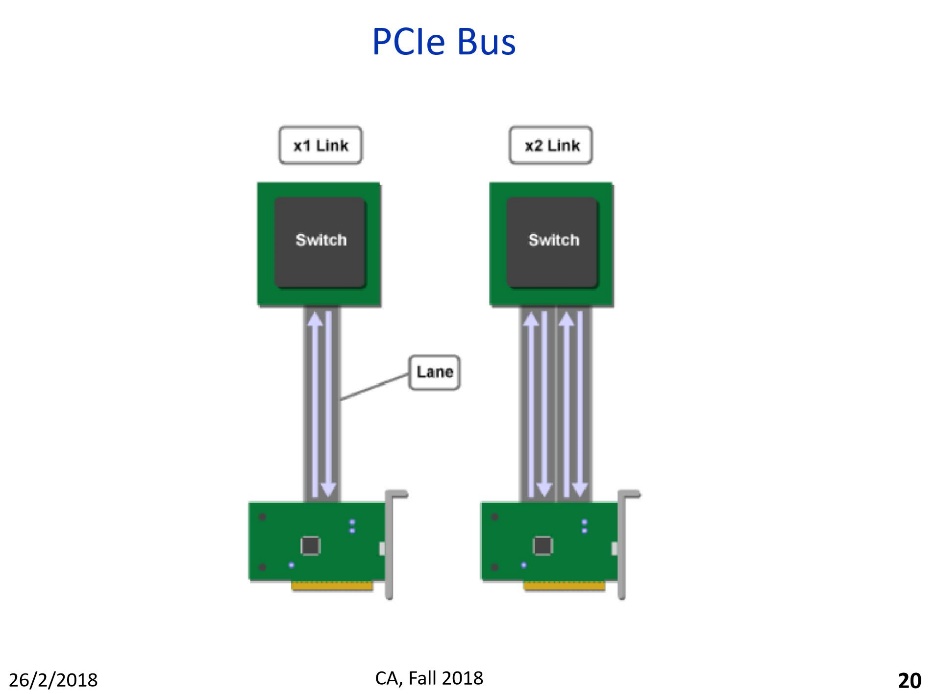


Кой е липсващия елемент - Switch ( PCle Bus )

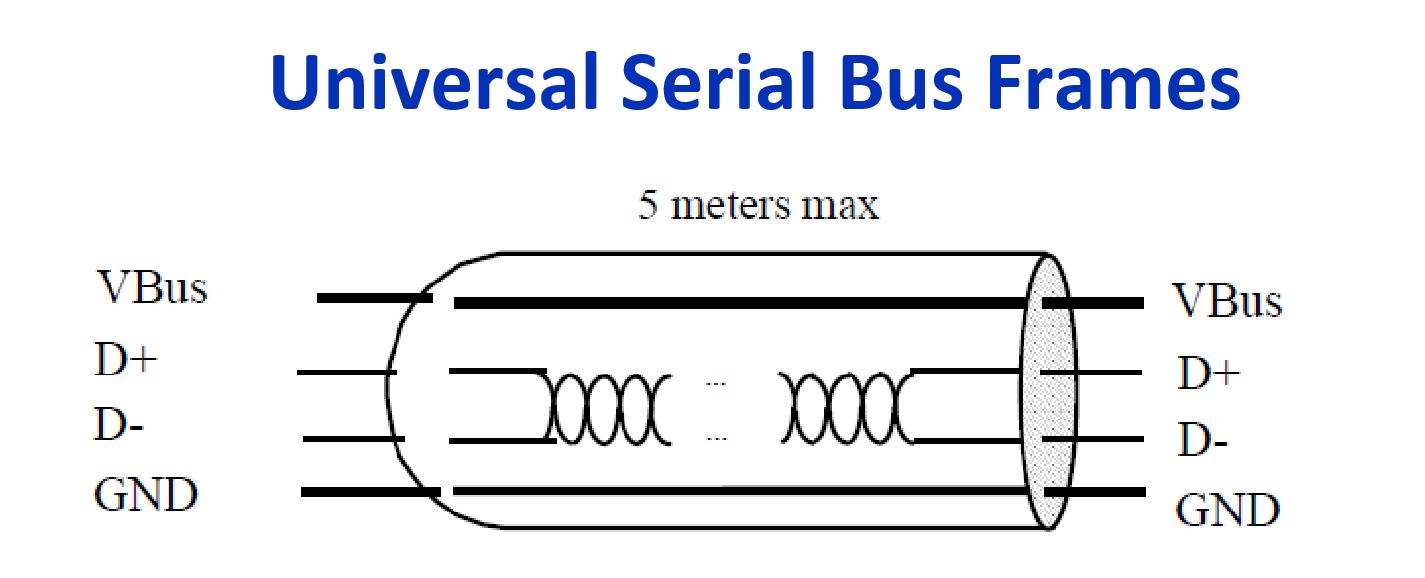


Кой е липсващия елемент// Защо има по една стрелка във всяка посока – защото са диференциални двойки и нещо със скорост и защото не стига честотната лента ( PCle Bus )

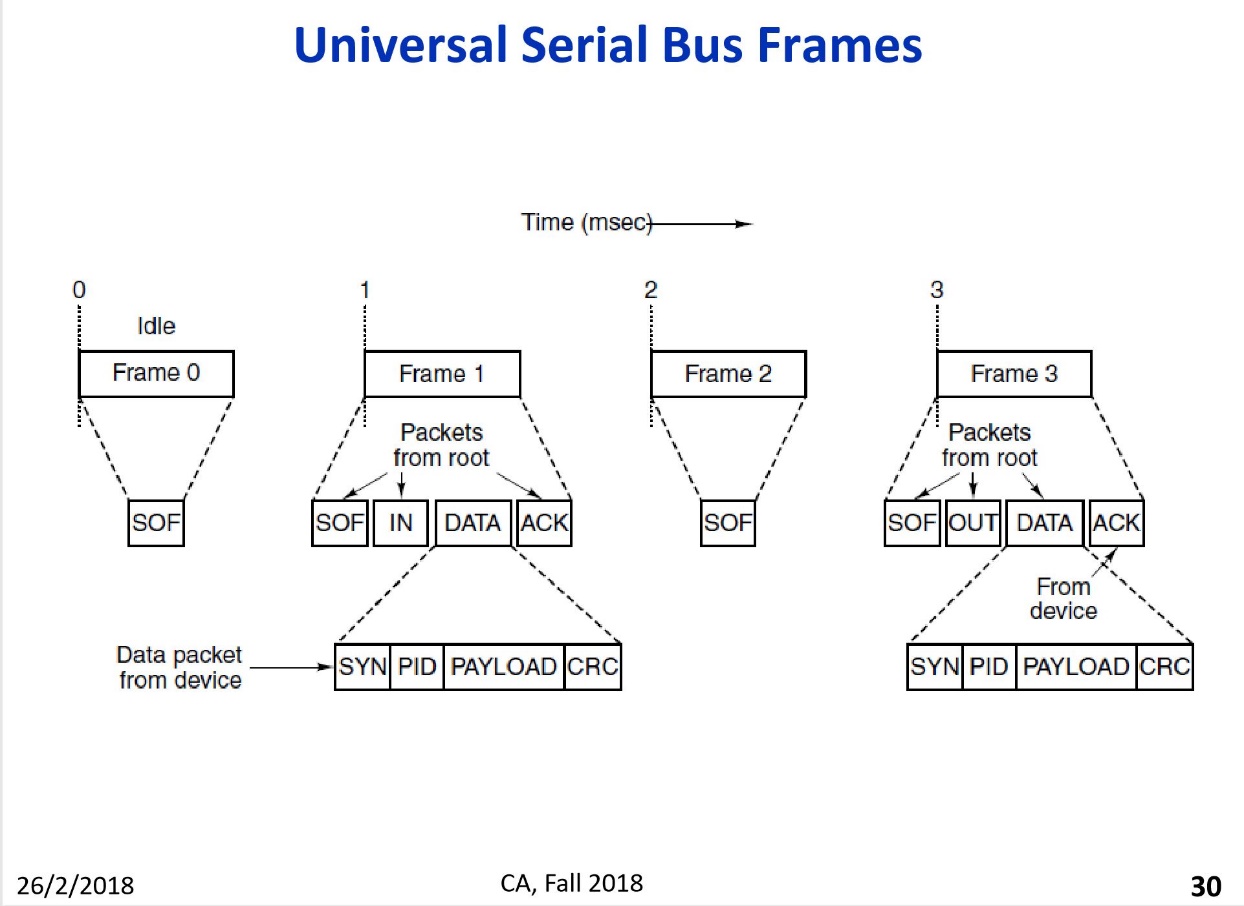


Защо се налага използването на повече двойки – като горното ( PCle Bus )

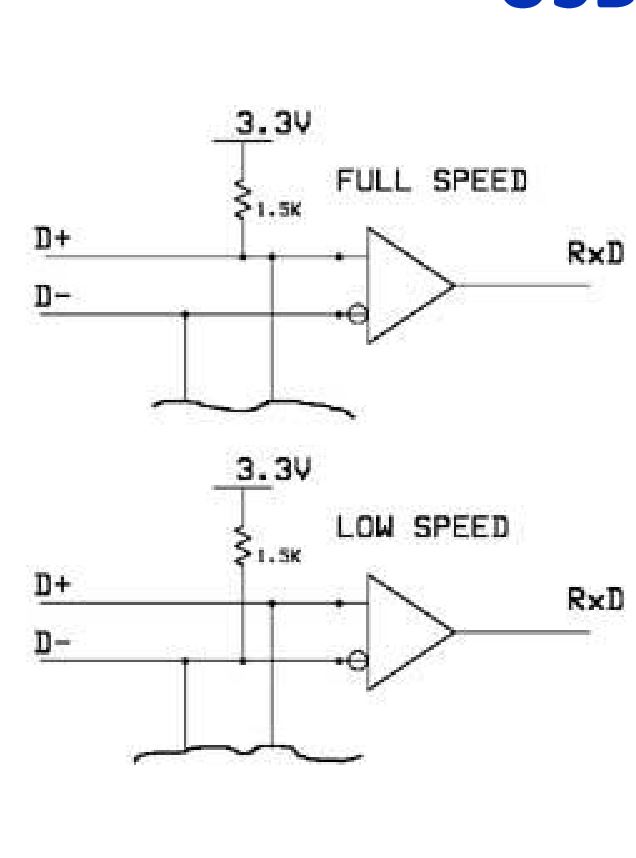
Защо Д+ и Д- са усукани – за шумоустойчивост ( Universal Serial Bus Frames )

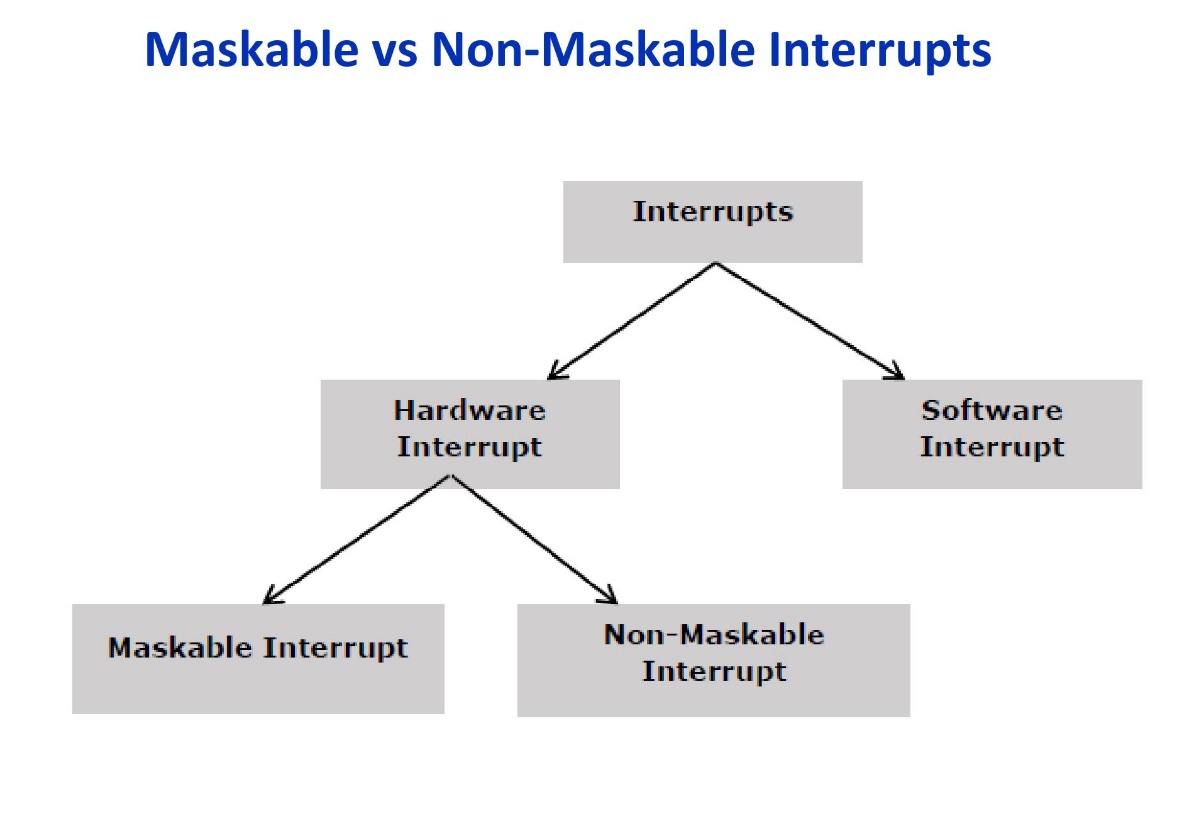


Кой фрейм е за четене – 1/ писане - 3 ( Universal Serial Bus Frames )

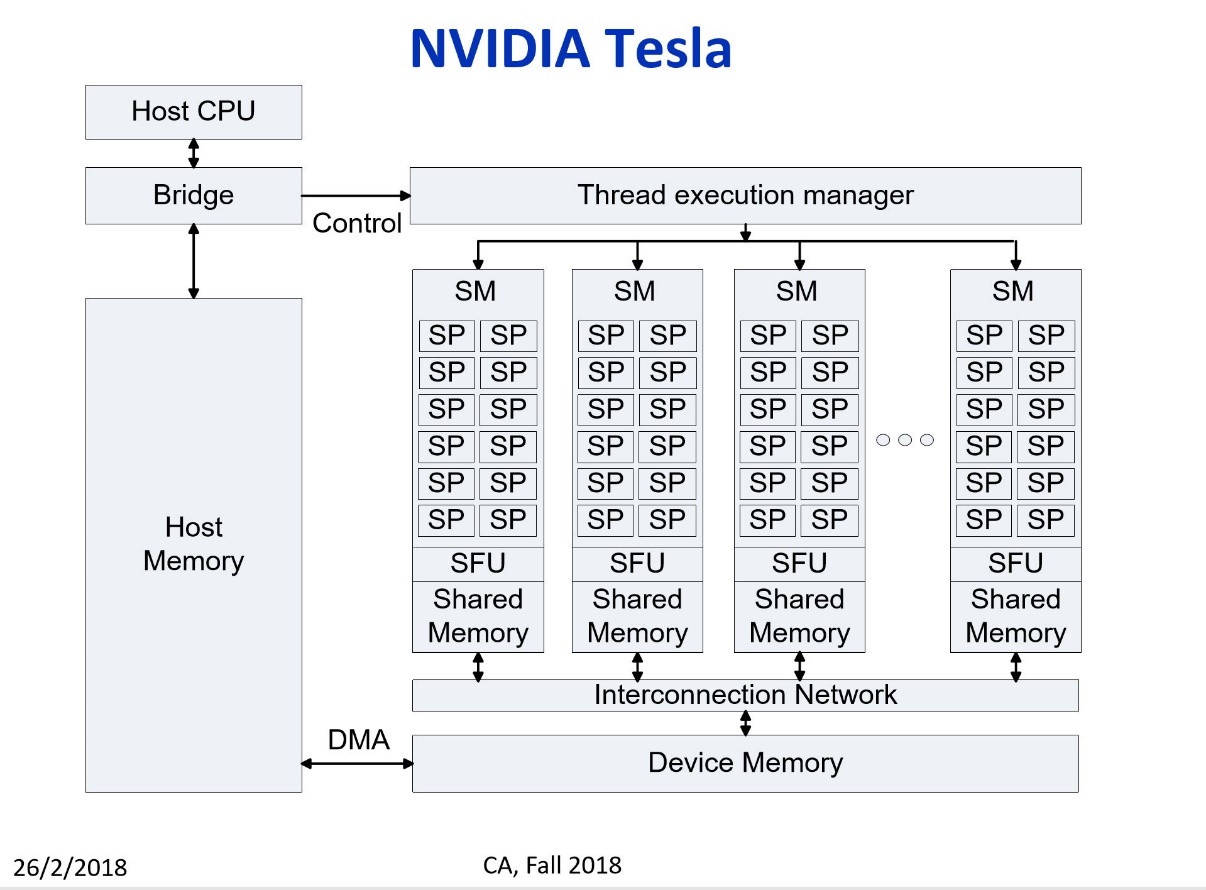


За какво служат резисторите (без надпис) – определя се скоростта на комуникация

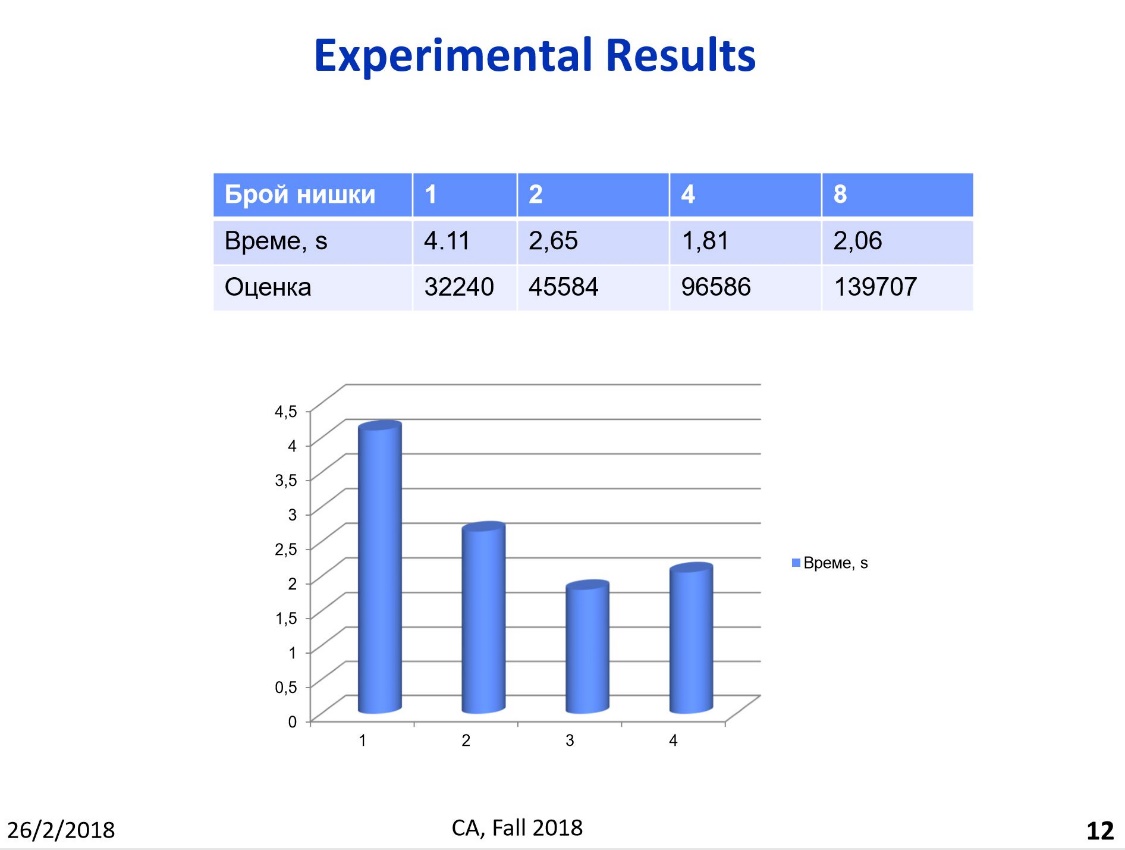


Видове прекъсвания (ще са изтрити някои) ( Maskable vs Non-Maskable Interrupts )

Кой е липсващия елемент – Host CPU, Bridge, Host Memory ( NVIDIA Tesla )



След коя нишка се покачва времето или нещо от сорта гледа се до кога пада и кога почва да се качва и отговора е последната нишка ( Experimental Results )



Какво е SOF – Start of Frame

Какво е CRC – Cyclic redundancy check

Какво е DMA – direct memory access

Frame header ще бъде ли доставен до потребителя – Да при двустранна комуникация. Да при комуникация с DMA

Колко проводна е USB3 – 19 USB1 2 – 9 (но не е сигурно)

1. ***Какви класове инструкции знаем?***

ММХ - набор от инструкции за множество данни.

SMX Safer Mode extensions – осигурява интерфейс за програмиране на системния софтуер

FPU инструкции (плаваща запетая) – х89 FPU инструкциите се изпълняват от така наречения „математически процесор“ . Тези инструкции работят с операции с десетична (BSD), с плаваща запетая, цяло число и двоично кодирани. Главната цел на тези инструкции е да изпълнява аритметика с числа с плаваща запетая. В днешно време SIMD е много по-бърза от FPU.

SSE - специални SIMD инструкции, които дават възможност за едновременна обработка на четири 32-битови числа с плаваща запетая. SSЕ инструкциите са част от х86 процесорна архитектура. SSE съдържа 70 инструкции. Бива разширен до SSE2, SSE3, SSSE3, SSE4.

SSE2 – разширена версия на SSE, имаща за цел да замести ММХ. Повечето инструкции на SSE2 изпълняват операциите с целочислени вектори, намиращи се също в ММХ.

FPL – инструкции (плаваща запетая)

SMXI

***2. Какво е характерно за регистрите с общо предназначение при 64-битов режим?***

*Регистрите с общо предназначение се използват за да съдържат операнди. 64 битовият режим позволява работа с 32 и 64 битови операнди, като по подразбиране се използват 32 битови операнди. При работа с 32 битови операнди се използват регистрите EAX, EBX, ECX, EDX, EDI, ESI, EBP, ESP, R8D - R15D. При работа с 64 битови операнди, пред обикновената (32 битова) инструкция се поставя REX префикс и регистрите са съответно: RAX, RBX, RCX, RDX, RDI, RSI, RBP, RSP, R8- R15. Т.е. REX се използва за генериране на 64 битови операнди.*

***3. Какво е характерно за адресното пространство (при някоя конкретна архитектура)?***

Режим на работа, Protected mode, Сегментация, Странициране

1. Линейното адресно пространство директно се изобразява върху физическото адресно пространство. Линейното адресно пространство е виртуално – извършва се странична преадресация; страниците имат размер 4KB.
2. В архитектурите IA-32 и Intel-64 са вградени два механизма за организация на виртуалната памет – сегментация (segmentation) и странициране (paging). В така наречения защитен режим могат да се използват и двата механизма за адресация
3. При сегментацията адресът се получава от 16-битов сегментен регистър и 32-битово отместване
4. При 4KB страници за адресиране на 4 GB линейно адресно пространство са необходими 220 страници.

***4. Как се изчислява адресът?***

modR/M байтът осигурява най-удобния от методите за адресиране. За операндите които се намират в паметта дефинирани от modR/M отместването във желания сегмент се пресмята като се сумират до три компонента: • displacement елемент в инструкцията. • базов регистър (base register). • индексен регистър (index register). Индексния регистър може автоматично да бъде умножен по scaling factor – 2, 4 или 8. Отместването което се получава от събирането на тези компоненти се нарича ефективен адрес. Всеки един от тези компоненти на ефективния адрес може да има и положителна 2 и отрицателна стойност. Ако сумата на всички тези компоненти превиши 232 , ефективния адрес се трансформира в 32 бита.

***5. Адресиране на операнди при IA-32 и INTEL-64 ??***

*Операндите в паметта се реферират чрез сегмент и отместването в него. При 32 битовия режим адресът на операнда се състои от 16 битов сегментен селектор и 32 битово отместване в сегмента. При 64 битовия режим, сегментния селектор също е 16 бита, но отместването може да бъде до 64^2.*

***6. Опишете сегемнтните регистри и тяхното предназначение***

*Сегментните регистри са 6 и остават 16 битови. Тяхното предназначение е да съхраняват сегментен селектор. Има различни сегменти. CS – (Code Segment) – съдържа базовия адрес на сегментът, в който се съхранява програмата. DS – (Data Segment) – съдържа адреса на сегмента, в който се съхраняват данните. SS – (Stack Segment) ES – (Extra Data Segment) – допълнителен сегмент за данни. FS - Допълнителен сегмент за данни. GS - Допълнителен сегмент за данни.*

***7. Опишете полетата на флаговия регистър***

***AF (Auxiliary Carry Flag)*** *- междинен пренос (междинен пренос между младшата и страшата тетрада). Смисъла от този флаг е някоя от следващите инструкции да провери този флаг. Използва се при двоично кодиране на десетични числа.****SF(Sign flag)*** *– най-левия бит на резултата (знака).****OF (Overflow flag)*** *- за препълване ако и двата са нула или и двата са единица нямаме препълване. Ако се различават то имаме такова.****DF (Direction flag)*** *– обикновено при сравнение на низове (обръщане на посоката на прочитане на данните). Този флаг може да се постави от приложната програма (разрешен е за пипане от всяка програма).****TF (Trap Flag)*** *– когато се сложи в единица се изпълнява една инструкция и се прави трап. Режим на трасировка.****IF (Interrupt Enable Flag)*** *– за да се случи прекъсване този флаг трябва да е единица.****IOPL (I/O Privilege Level)*** *– ниво на привилегия на входно изходните устройства. От това се определя дали да се случи едно входно изходнопрекъсване или не.****AC (Alignment Check)*** *– когато се сложи в единица се задължава процесора да прави проверка за подравняването на границите на операндите. Ако е е в нула процесора не се интересува къде са границите.****ID (ID flag)*** *– идея всеки процесор да си има собствен идентификатор. Това не е този идентификатор, но при този флаг в единица се позволява да се ползва този идентификатор.*

***Status Flags*** *- Статус флаговете на EFLAGS регистъра позволяват резултатите от една инструкция да влияят на следващи инструкции. Аритметичните инструкцииизползват OF, SF, ZF, AF, PF и CF. SCAS (Scan String), CMPS (Compare String) и LOOP инструкците използват ZF за да сигнализират, че са свършили работата си.*  
***Control Flag*** *- Контролният флаг DF от регистъра EFLAGS контролира инструкциите с низове.*

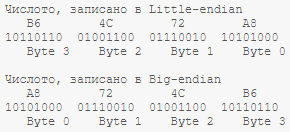
***8. Опишете режимите на адресация при IA-32 и Intel-64***

*x86-64 процесор се държи като IA-32 процесор когато е включен в реален или защитен режим. Това са режими които се поддържат когато процесора не е в long режим. Докато големината на регистрите е увеличена до 64-бита в сравнение с предишната x86 архитектура, адресирането на паметта все още не е увеличено до пълните 64 бита. Поради това, load/store unit(s), cache tags, MMUs и TLBs могат да бъдат по-прости без никаква загуба на използваема памет.*

***9. Опишете основните типове данни***

*- Байт, дума, двойна дума, четворна дума и двойна четворна дума са фундаменталните типове данни. Integer - Знакова двоична числова стойност; Unsigned Integer - Беззнакова двоична числова стойност; Floating point – Формат с плаваща запетая; Near pointer - 32-битов логически адрес; Far pointer - 48-битов логически адрес от два компонента: 16-битов сегментен селектор и 32-битово отместване; Bit field - Последователност от битове. Битово поле може да започне от всяка позиция и всеки байт и може да съдържа до 32 бита; BCD - кодиране на всяка десетична цифра на число като набор от четири бита. При това всеки байт на числото съдържа една или две десетични цифри в така наречения двоично-десетичен код (BCD — Binary-Coded Decimal); SIMD data types.*

***10. Представете схема на паметта при big endian и little endian записване на данните. Пояснете схемата.***

**

***Big-endian****-****старшият байт е първи****.****Little-endian****- при нея****първи е младшият байт****.  
При Little-endian****с увеличаването на адреса се увеличава старшинството (значимостта, порядъкът) на байтовете****. При Big-endian е****точно обратното****.*

***11. Какво представлява и какви са предимствата на пакетираните SIMD типове данни?***

# *SIMD(Single–Instruction Multiple–Data) – подход, който използва алгоритм за работа с мултимедийни данни, който се базират на паралелното изпълнение на една операция над няколко числа.*

*MMX (Multi-Media eXtension) – технологията е разработена за да ускори мултимедийните програми и програмите за комуникация като включват нови инструкции и типове данни които позволяват на програмите да достигат ново ниво на производителност.*

***12. Опишете полетата, обособяващи формата на инструкциите***

*1)Instruction prefixes – Чрез префикса може да се променя поведението на инструкцията, като например да се сменя сегмента, размера на думата, да се управлява цикличното изпълнение и използването на шината  
2)Opcode – Кодът на операцията се кодира с 1, 2 или 3 байта. Допълнително може да се използват и 3 бита от полето ModR/M. Използването на тези полета зависи в голяма степен от типа операцията.*

*3)ModR/M байт –не задължителен и понякога може да съдържа част от opcode-а.  
4)SIB – – Scaled Index Base,не е задължителен*

*5)Displacement – Размерът на отместването се определя от полето Mod на ModR/M байта  
6)Immediate - не е задължителен компонент и се използва като числова стойност от вариращи размери от байтове(byte, word, long).*

**13. Опишете режимите на адресация и ролята на modR полето**

ModR/M - това поле специфицира, кои регистри или адреси от паметта да се използват като операнди на инструкцията.

Полето Mod определя методът за адресация на операндите от паметта. По подразбиране, полето R/M съдържа първия операнд, а Reg/Opcode втория. Ако инструкцията изисква само един операнд, то полето R/M съдържа допълнителен код на операцията, вместо адрес на регистър.

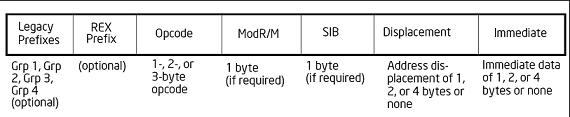
Полето Mod е два бита и специфицира режимите на адресация.

Полето R/M определя регистъра.

***14. За какво и как се ползва префиксът?***

*Чрез префикса може да се променя поведението на инструкцията, като например да се сменя сегмента, размера на думата, да се управлява цикличното изпълнение и използването на шината. Има няколко групи префикса, всяка група може да е 1 байт (общо една група може да е 4 байта или 0).*

***15. Формат на инструкциите при IA-32?***

**

*Инструкцията съдържа: незадължително поле префикс,REX префикс(незадължително), код на операцията (до три байта), спецификатор формиращ адрес (не е задължителен) и съдържа ModR/M байт и понякога а SIB (Scale-Index-Base) байт, отместване (незадължително) и поле „непосредствен“ (незадължително).*

***16. Поведение на REX prefix?***

*Към формата на инструкцията в 64 битов режим се добавя REX префикс. REX префикса съдържа битове, които разширяват ModR/M полетата. По този начин ще работят и старите и новите програми, които използват повече регистри.*

***17. Приложения на управляващи регистри***

*Управляващите регистри се използват от операционната система. Те са 32 битови във всички 32 битови режими. В 64 битов режим системните регистри са 64 битови. CR0 - Съдържа флагове контролиращи режимите на работа и състоянието на процесора. CR1 — Запазен, но не се използва. CR2 – Съдържа линейния адресът, който е причина за изключение от тип „page fault“. CR3 (PDBR) - Съдържа физическия адрес на таблицата със страници на текущия процес, както и флаговете PCD и PWT. CR4 - Съдържа група от флагове позволяващи използването на различни разширения.*

***18. Какво е характерно за защитния режим?***

*Защитения режим се контролира от управляващи регистри (Control Registers), които са означени CR0, CR2, CR3 и CR4.*

*Защитен режим е режим на работа на компютърните процесори в архитектурата*[*x86*](https://bg.wikipedia.org/wiki/X86)*. При защитения режим, максималната адресирана памет е 4 GB (232 байта), има 'странициране' и 'сегментиране' на паметта, както и апаратна защита на достъпа до паметта и входно-изходното пространство. Почти всички съвременни операционни системи за x86 процесори работят изключително в защитен режим.*

***19. Предназначения на сегментните селектори***

*Сегментните селектори са 16 битови идентификатори на сегмента. Старшите 13 бита се използват за определяне на отместването в глобалната или локалната дескрипторни таблици. Трите младши бита имат по-специфично предназначение*

***20. Механизъм на обновяване на страница от диск***

*Мениджърът на паметта използва информацията от таблиците за да намери неизползвана страница физическа памет (такава, към която скоро не е имало обръщения) и я записва в резервирана област на диска, наречена swap файл. После уведомява централния процесор за необходимостта да прочете изискваната страница от по-бавната памет (файл на диска или swap файл). Накрая MMU установява съответствието между виртуалната и физическата страница и обновява таблицата с тази информация.*

***21. Какво представлява сегментирането на паметта? Предимства и недостатъци***

*При сегментацията адресът се получава от 16-битов сегментен регистър и 32-битово отместване. Регистърът EIP е 32-битов и в него се записва отместването относно началото на кодовия сегмент, а регистърът ESP е 32-битов и в него се записва адресът на върха на стека, зададен като отместване относно началото на стековия сегмент. Всеки от останалите 32-битови общи регистри може да се използва за задаване на отместване при адресация на данни в някой от сегментите за данни.*

***22. Обяснете преобразуването на логическия адрес във физически при сегментно страничната организация на паметта***

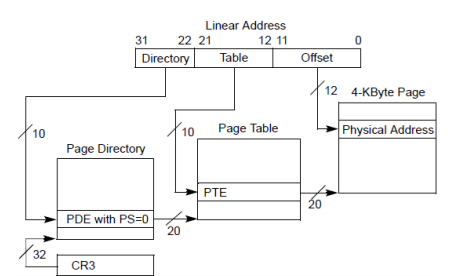
*В режим на сегментация и странициране получаваме 32-битов линеен адрес. Първите 10 бита от него се умножават по ширината на полето на каталога (4 bytes) и се добавят към началния адрес на каталога. Ако старшият бит на реда в каталога е 1, значи че съществува каталог в паметта, който се сочи от реда в каталога. В такъв случай се взимат младшите 20 бита от реда в каталога, добавят им се 12 нули отдясно и се прибавя стойността на полето PN, умножена по широчината на реда в каталога. В реда в каталога (в младшите му20 бита) е записан адресът на търсената от нас страница, ако тя съществува. Последните 20 бита на реда в каталога съдържат линейния адрес на страницата, а физическият адрес на нейното начало се получава, като отново към тези 20 бита прибавим 12 нули отдясно и се прибави 12-битовият offset.*

***23. Какви полета съдържа локалната дискрипторна таблица?***

*LDT – Локална Дескрипторна Таблица е таблица в оперативната памет, в която се разполага на части потребителската програма (всяка част е на отделен ред от таблицата). Всеки ред от тази таблица се нарича дескриптор. Адресът се задава с двойки регистри – сегментен регистър и отместване или сегментен регистър и регистър-база. Винаги е 16-битов, като десният бит показва дали се адресира спрямо ЛДТ или ГДТ. В два бита се кодира какъв режим се използва. Остават 13 бита – едно число, което е отместване в дескрипторната таблица*

***24. Каква информация се съдържа в task register***

*16-битовия сегментен селектор и целия сегментен дескриптор на текущата задача (32-битов базов адрес, 16-битова граница на сегмента и атрибути на дексриптора). Тази информация се копира от TSS-дескриптора в GDT за текущата задача.*

***25. Как се осъществява преобразуването на линеен адрес към 4 –KB стр с използване на 32-битово странициране?***

*При този метод за адресиране може да се адресират до 220 страници, което прави 232 байта или 4GB. За да се избират различните редове от таблицата, линейният адрес се разделя на четири секции: • Ред в таблицата на главния каталог (битове от 30 и 31) – съдържа се отместването на всеки от четирите реда в таблицата на главния каталог. Избраният ред задава съответния физически адрес на каталога. • Ред в каталога (битове от 21 до 29) – съдържа се отместването на всеки от редовете в таблицата на каталогa. Избраният ред задава съответния физически адрес на каталога. • Ред в таблицата на страниците (битове от 12 до 20) – съдържа се отместването на всеки от редовете в таблицата на страниците. Избраният ред задава съответния физически адрес на страницата във физическата памет. • Отместване в страницата (битове от 12 до 20) – съдържа се отместването на физическия адрес в съответната страница.*

***26. Обяснете структурата на задачата при IA 32 архитектура***

*Задачата се състои от пространство за изпълнение (execution space) и сегмент за състоянието на задачата (Task State Segment TSS). Пространството за изпълнение се състои от сегмент за кода, стека и няколко сегмента за данни. Най-общо казано, TSS съхранява в паметта информацията, необходима за изпълнението (execution space) на дадена задача.*

***27. Обяснете функцията на task register***

*Регистърът на задачата (TR) съдържа 16-битовия сегментен селектор и целия сегментен дескриптор на текущата задача (32-битов базов адрес, 16-битова граница на сегмента и атрибути на дексриптора). Тази информация се копира от TSS-дескриптора в GDT за текущата задача.*

***28. Как се осъществява превключването на задачите?***

*Когато бъде назначено изпълнението на дадена задача, става автоматично превключване между текущо изпълняваната задача и новоназначената задача за изпълнение. При превключването към друга задача, състоянието на средата, в която се е изпълнявала текущата задача се запазва в TSS и изпълнението на задача се прекратява за някакъв период от време. След това информацията за предстоящата за изпълнение задача се зарежда в процесора и нейното изпълнение започва от инструкцията, сочена от вече зареденият регистър EIP на текущата задача.*

***30. Каква информация се съдържа в IDT?***

*Interrupt Descriptor Table (IDT) e таблицата на векторите на прекъсванията. Тя има 256 елемента. Векторите на прекъсванията не са дескриптори на сегменти, а са входове към такива.*

***31. Обяснете групите, на които се делят изключенията***

***- aborts, faults, traps***

- trap e изключение, при което, след като се изпълни съответната програма за обработка, броячът на инструкциите се увеличава и сочи следващата инструкция.

-fault като цяло могат да бъдат коригирани и за това след като се случат и бъде изпълнена обработващата програма, се стартира отново същата програма (която е генерирала грешката), като се счита, че грешката вече е коригирана (от обработващата програма).

-abort не винаги се знае коя инструкция ги е предизвикала и не позволяват рестартиране на програмата или процесът, които ги причиняват.

***32. По какво се различават изключенията от тип faults от тип aborts?***

*Изключенията от тип грешки (fault) като цяло могат да бъдат коригирани и за това след като се случат и бъде изпълнена обработващата програма, се стартира отново същата програма (която е генерирала грешката), като се счита, че грешката вече е коригирана (от обработващата програма).*

*За изключенията от тип abort не винаги се знае коя инструкция ги е предизвикала и не позволяват рестартиране на програмата или процесът, които ги причиняват. Използват се за регистриране на хардуерни грешки или недопустими стойности в системни таблици.*

***33. Каква е фунцкията на челната част?***

*Челната част извлича и декодира инструкциите. Декодираните инструкции се транслират до микрооперации (μops). Основната задача на тази част е да осигурява непрекъснат поток от микрооперации на изпълняващата част.*

***34. Как работи изпълняващата част?***

*Изпълняващата част пренарежда микрооперациите по такъв начин, че тези чиито операнди са готови (и има налични изчислителни ресурси) да се изпълнят възможно по - скоро. Изпълняващата част може да обработи (issue) няколко микрооперации за един цикъл.*

***35. Каква е ролята на завършващата част?***

*Завършващата част се грижи резултатът от изчисленията да съответства на реда на постъпване на инструкции и че състоянието на системата се обновява правилно.*

**36. Обяснетеп принципа на работа на кеш-паметта и протоколите за процесори**

Кеш-паметта представлява структура, състояща се от множество от рамки (frames). Всяка рамка се състои от данни, етикет (tag) и състояние. Етикетът е характеристика на рамката, която е съществена при търсене в данните в кеша. Състоянието е два бита – бит valid, който показва дали данните в кеша са валидни и бит dirty, който показва дали в рамката са били писани данни. Търсенето и зареждането на рамки в кеша става по следния начин: проверява се дали входящият етикет съвпада с етикета на рамката. Ако не съвпадат, рамката не се намира (cache-miss) и в такъв случай блокът данни в рамката се изхвърля и се заменя с блок от по-долното ниво в йерархията, а след това блокът се подава на предното ниво в йерархията.

Протокол на кеша MESI: използва се както в едноядрени, така и в многоядрени процесори. При този протокол се проверяват четири дефинирани състояния на всяка линия на кеша – (Modified, Exclusive, Shared, Invalid).

***37. Какво представлява и как работи предварителното извличане?***

*Предварителното извличане на данните в кеша (Cache prefetching) е техника, която увеличава производителността, като копира данните от бавна памет в по-бърза, още преди да са необходими. Най-често се използват потокови буфери (Stream buffers), които се основават на последователното изпълнение на инструкциите и извличане на данните.*

***38. Обяснете приликите и разликите м/у прекъсванията и изключванията***

*При Intel x86 архитектурата прекъсванията и изключенията (interrupts vs exceptions) са две покриващи се групи векторизирани събития, които са индикация за наличието на състояние, което изисква обработка от страна на процесора.*

*Прекъсванията и изключенията карат процесора да скочи (jump) в програма, която обработва прекъсването (interrupt handler), след като запази контекста (за да може прекъснатата програма да продължи). Прекъсванията и изключенията имат идентификатор (ID), наречен вектор, които определя адреса на програмата за обработка на прекъсването. Тези програми (interrupt handler) се извикват през таблица на дескрипторите на прекъсванията*

*Прекъсванията се случват докато се изпълнява дадена програма в резултат на сигнали от хардуера. Прекъсванията се използват за да се обработят външни за процесора събития, като искане за обслужване н а периферно устройство. Прекъсвания могат да се генерират и софтуерно, чрез изпълнение на инструкцията INT n, където n идентификаторът (ID) или наричан също векторът на прекъсването (от 0 до 255).*

*Изключенията се случват, когато процесорът детектира грешка (error condition) по време на изпълнение на инструкция, като например делене на нула, непозволен достъп (protection violations), липсваща страница в паметта (page faults) и др.*

***39. GDTR и LDTR***

*Global Descriptor Table Register (GDTR) – съдържа базовия адрес (32 бита в защитен и 64 бита в IA-32e режим) и размера (16 бита) на глобалната дескрипторна таблица GDT. Инструкциите LGDT и SGDT се използват съответно за зареждане и записване стойността на регистъра. Глобалната дескрипторна таблица, която е с 8 байтови елементи, описващи къде се намират сегментите се съхранява в оперативната памет.*

*Local Descriptor Table Register (LDTR) – съдържа 16 битов сегментен селектор, базов адрес (32 бита в защитен и 64 бита в IA-32e режим), размер на сегмента и атрибути на дескриптора. Базовият адрес определя линейния адрес на първия байт от LDT сегмента.*

**1. Какви класове инструкции знаем?**

FPL инструкции (плаваща запетая), SSE, MMX - Matrix Math Extensions, SSE2, SMXI

**2. Какво е характерно за регистрите с общо предназначение при 64-битов режим?**

Регистрите с общо предназначение се използват за да съдържат операнди. 64 битовият режим позволява работа с 32 и 64 битови операнди, като по подразбиране се използват 32 битови операнди. При работа с 32 битови операнди се използват регистрите EAX, EBX, ECX, EDX, EDI, ESI, EBP, ESP, R8D - R15D. При работа с 64 битови операнди, пред обикновената (32 битова) инструкцията се поставя REX префикс и регистрите са съответно: RAX, RBX, RCX, RDX, RDI, RSI, RBP, RSP, R8- R15. Т.е. REX се използва за генериране на 64 битови операнди.

**3. Какво е характерно за адресното пространство (при някоя конкретна архитектура)?**

* режим на работа
* protected mode
* сегментация

При IA-32 и Intel-64 съществуват два режима на работа на процесора –

Real Mode и Protected Mode, като проц. Pentium и Itanium CPU работят стандартно при Protected, a предходните модели проц. Работят в Real-Address Mode. Режимът на работа се определя от флага PE в управляващия рег. В съотвествие с изброените два режима на работа на арх, съществуват 3 режима на работа и организация на паметта:

**"Flat" моделът**

При "flat" моделът на организация на паметта, програмистът вижда един масив от до 2^(32) байта (4 GB). Физическата памет може да съдържа до 4GB, но обикновено е много по-малка. Процесорът изобразява 4-те GB flat пространство върху физическото адресно пространство чрез транслации, които ще описваме във някоя тема по-нататък. Програмистът не се нуждае от детайлите по това изобразяване.

Pointer-ът на това flat адресно пространство е 32-битово число, което може да варира от 0 до 2^(32) -1.

**Сегментният модел**

При сегментния модел на организация на паметта, адресното пространство се разглежда като логическо адресно пространство където то е с големина до 2^(46) байта (64 TB). Процесорът изобразява 64 TB-овото логическо адресно пространство на физическото адресно пространство (до 4GB) посредством механизми за транслация на адреси, които ще разгледаме в някоя от темите по-нататък. Програмистът не е нужно да знае как се изобразяват.

Програмистът вижда логическото адресно пространство на x86 архитектурата като колекция от до 16,383 едно- измерни подпространства, всяко от които със определена дължина. Всяко от тези линейни подпространства е наречено сегмент. Големината на един сегмент може да варира от един байт до 2^(32) байта (4 Gb).

Пълен pointer в това адресно пространство се състои от две части:

# сегментен селектор, който е 16-битово поле, което идентифицира сегмента.

# Отместване(offset), който е 32-битово число, което адресира байт нивото в сегмента.

**"Real-Address Mode" моделът**

При Real-Address Mode Model цялата памет се разделя на равни по размер сегменти. В CPU за всеки един от тези от тези сегменти има по един регистър, който сочи началото на съответния сегмент. Програмистът има достъп и може да променя съдържанието на кой да е от тези регистри. Промяната на съдържанието автоматично води до преместване на съответния сегмент в друга начална точка. Съществуват и други регистри с общо предназначение, които също са програмно достъпни. Те служат за задаване на отместването на адресната клетка от началото. За останалите три сегмента може да се използва кой да е от останалите регистри. Следователно адресът на клетката се задава винаги чрез двойка регистри – сегментен регистър и регистър-отместване.

**4. Как се изчислява адресът?**

// Състои се от сегментен селектор и отместване. Сегментният селектор сочи към дексриптора

Операндите в паметта се реферират чрез сегмент и отместването в него. При 32 битовия режим адресът на операнда се състои от 16 битов сегментен селектор и 32 битово отместване в сегмента. При 64 битовия режим, сегментния селектор също е 16 бита, но отместването може да бъде до 64^2.Отместването (Offset) може да бъде специфицирано като статична стойност (displacement) или да се изчисли с използване на следните компоненти : Displacement , Base, Index, Scale factor. Всеки от 8-те регистри с общо предназначение могат да се използват като за база. Регистри с общо предназначение, с изключение на стековия, могат да се използват като index при индексна адресация.

**5. Адресиране на операнди при IA-32 и INTEL-64 ??**

Операндите в паметта се реферират чрез сегмент и отместването в него. При 32 битовия режим адресът на операнда се състои от 16 битов сегментен селектор и 32 битово отместване в сегмента. При 64 битовия режим, сегментния селектор също е 16 бита, но отместването може да бъде до 64^2.

**6. Опишете сегемнтните регистри и тяхното предназначение**

Сегментните регистри са 6 и остават 16 битови. Тяхното предназначение е да съхраняват сегментен селектор. Има различни сегменти. CS – (Code Segment) – съдържа базовия адрес на сегментът, в който се съхранява програмата. DS – (Data Segment) – съдържа адреса на сегмента, в който се съхраняват данните. SS – (Stack Segment) ES – (Extra Data Segment) – допълнителен сегмент за данни. FS - Допълнителен сегмент за данни. GS - Допълнителен сегмент за данни.

**7. Опишете полетата на флаговия регистър**

**AF (Auxiliary Carry Flag)** - междинен пренос (междинен пренос между младшата и страшата тетрада). Смисъла от този флаг е някоя от следващите инструкции да провери този флаг. Използва се при двоично кодиране на десетични числа.  
 **SF(Sign flag)** – най-левия бит на резултата (знака).  
 **OF (Overflow flag)** - за препълване ако и двата са нула или и двата са единица нямаме препълване. Ако се различават то имаме такова.  
 **DF (Direction flag)** – обикновено при сравнение на низове (обръщане на посоката на прочитане на данните). Този флаг може да се постави от приложната програма (разрешен е за пипане от всяка програма).  
 **TF (Trap Flag)** – когато се сложи в единица се изпълнява една инструкция и се прави трап. Режим на трасировка.  
 **IF (Interrupt Enable Flag)** – за да се случи прекъсване този флаг трябва да е единица.  
 **IOPL (I/O Privilege Level)** – ниво на привилегия на входно изходните устройства. От това се определя дали да се случи едно входно изходнопрекъсване или не.  
 **AC (Alignment Check)** – когато се сложи в единица се задължава процесора да прави проверка за подравняването на границите на операндите. Ако е е в нула процесора не се интересува къде са границите.  
 **ID (ID flag)** – идея всеки процесор да си има собствен идентификатор. Това не е този идентификатор, но при този флаг в единица се позволява да се ползва този идентификатор.

**Status Flags** - Статус флаговете на EFLAGS регистъра позволяват резултатите от една инструкция да влияят на следващи инструкции. Аритметичните инструкцииизползват OF, SF, ZF, AF, PF и CF. SCAS (Scan String), CMPS (Compare String) и LOOP инструкците използват ZF за да сигнализират, че са свършили работата си.  
 **Control Flag** - Контролният флаг DF от регистъра EFLAGS контролира инструкциите с низове.

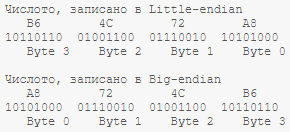
**8. Опишете режимите на адресация при IA-32 и Intel-64**

Операндите в паметта се реферират чрез сегмент и отместването в него. При 32 битовия режим адресът на операнда се състои от 16 битов сегментен селектор и 32 битово отместване в сегмента. При 64 битовия режим, сегментния селектор също е 16 бита, но отместването може да бъде до 64^2.

**9. Опишете основните типове данни**

Байт, дума, двойна дума, четворна дума и двойна четворна дума са фундаменталните типове данни. Integer - Знакова двоична числова стойност; Unsigned Integer - Беззнакова двоична числова стойност; Floating point – Формат с плаваща запетая; Near pointer - 32-битов логически адрес; Far pointer - 48-битов логически адрес от два компонента: 16-битов сегментен селектор и 32-битово отместване; Bit field - Последователност от битове. Битово поле може да започне от всяка позиция и всеки байт и може да съдържа до 32 бита; BCD - кодиране на всяка десетична цифра на число като набор от четири бита. При това всеки байт на числото съдържа една или две десетични цифри в така наречения двоично-десетичен код (BCD — Binary-Coded Decimal); SIMD data types

**10. Представете схема на паметта при big endian и little endian записване на данните. Пояснете схемата.**



Big-endian - старшият байт е първи.  
 Little-endian - при нея първи е младшият байт.  
 При Little-endian с увеличаването на адреса се увеличава старшинството (значимостта, порядъкът) на байтовете. При Big-endian е точно обратното.

**11. Какво представлява и какви са предимствата на пакетираните SIMD типове данни?**

# [SIMD](http://arstechnica.com/old/content/2000/03/simd.ars) - single-instruction multiple-data – подход, който използва алгоритм за работа с мултимедийни данни, който се базират на паралелното изпълнение на една операция над няколко числа.

MMX (Multi-Media eXtension) – технологията е разработена за да ускори мултимедийните програми и програмите за комуникация като включват нови инструкции и типове данни които позволяват на програмите да достигат ново ниво на производителност.

**12. Опишете полетата, обособяващи формата на инструкциите**

1)Instruction prefixes – Чрез префикса може да се променя поведението на инструкцията, като например да се сменя сегмента, размера на думата, да се управлява цикличното изпълнение и използването на шината  
 2)Opcode – Кодът на операцията се кодира с 1, 2 или 3 байта. Допълнително може да се използват и 3 бита от полето ModR/M. Използването на тези полета зависи в голяма степен от типа операцията.

3)ModR/M байт –не задължителен и понякога може да съдържа част от opcode-а. Дефинира дали е необходим СИБ байт, immediate операнд или отместване. Деф. Рег. Които се използват в зависимост от операндите.   
4)SIB – – Scaled Index Base,не е задължителен . Два бита за scale, три бита за индексен рег. И 3 бита за базов рег. Съдържанието на индексите се умножава по 1,2 и т.н. в зависимост от МОД и се събира с базовия рег

5)Displacement – Размерът на отместването се определя от полето Mod на ModR/M байта (ако R/M полето го позволи)  
 6)Immediate - не е задължителен компонент и се използва като числова стойност от вариращи размери от байтове(byte, word, long).

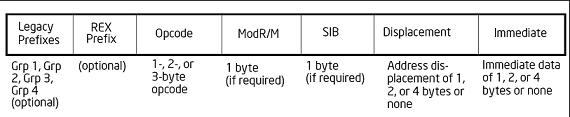
**13. Опишете режимите на адресация и ролята на modR полето**

Това поле специфицира, кои регистри или адреси от паметта да се използват като операнди на инструкцията. Полето Mod определя методът за адресация на операндите от паметта. По подразбиране, полето R/M съдържа първия операнд, а Reg/Opcode втория. Ако инструкцията изисква само един операнд, то полето R/M съдържа допълнителен код на операцията, вместо адрес на регистър. Полето Mod е два бита и специфицира режимите на адресация. Полето R/M определя регистъра.

**14. За какво и как се ползва префиксът?**

Чрез префикса може да се променя поведението на инструкцията, като например да се сменя сегмента, размера на думата, да се управлява цикличното изпълнение и използването на шината.

**15. Формат на инструкциите при IA-32?**



Инструкцията съдържа: незадължително поле префикс,REX префикс(незадължително), код на операцията (до три байта), спецификатор формиращ адрес (не е задължителен) и съдържа ModR/M байт и понякога а SIB (Scale-Index-Base) байт, отместване (незадължително) и поле „непосредствен“ (незадължително).

**16. Поведение на REX prefix?**

Към формата на инструкцията в 64 битов режим се добавя REX префикс. REX префикса съдържа битове, които разширяват ModR/M полетата. По този начин ще работят и старите и новите програми, които използват повече регистри.

**17. Приложения на управляващи регистри**

Управляващите регистри се използват от операционната система. Те са 32 битови във всички 32 битови режими. В 64 битов режим системните регистри са 64 битови. CR0 - Съдържа флагове контролиращи режимите на работа и състоянието на процесора. CR1 — Запазен, но не се използва. CR2 – Съдържа линейния адресът, който е причина за изключение от тип „page fault“. CR3 (PDBR) - Съдържа физическия адрес на таблицата със страници на текущия процес, както и флаговете PCD и PWT. CR4 - Съдържа група от флагове позволяващи използването на различни разширения.

**18. Какво е характерно за защитния режим?**

Защитен режим е режим на работа на компютърните процесори в архитектурата [x86](https://bg.wikipedia.org/wiki/X86). При защитения режим, максималната адресирана памет е 4 GB (232 байта), има 'странициране' и 'сегментиране' на паметта, както и апаратна защита на достъпа до паметта и входно-изходното пространство. Почти всички съвременни операционни системи за x86 процесори работят изключително в защитен режим

**19. Предназначения на сегментните селектори**

Сегментните селектори са 16 битови идентификатори на сегмента. Старшите 13 бита се използват да определяне на отместването в глобалната или локалната дескрипторни таблици. Трите младши бита имат по-специфично предназначение. В зависимост от флага TI се адреси глобална или локална таблица. Полето RPL съдържа необходимото ниво на привилегия, като за да бъде позволен достъп, стойността на (DPL — Descriptor privilege level) трябва да е по-малка или равна от RPL. Всеки сегментен селектор има видима и скрита част. При първото обръщение към дескриптора се извлича информацията и се записва в скритата част на сегментния регистър. След това се използва, до като не се наложи да се смени сегмента.

**20. Механизъм на обновяване на страница от диск**

Мениджърът на паметта използва информацията от таблиците за да намери неизползвана страница физическа памет (такава, към която скоро не е имало обръщения) и я записва в резервирана област на диска, наречена swap файл. После уведомява централния процесор за необходимостта да прочете изискваната страница от по-бавната памет (файл на диска или swap файл). Накрая MMU установява съответствието между виртуалната и физическата страница и обновява таблицата с тази информация.

**21. Какво представлява сегментирането на паметта? Предимства и недостатъци**

Сегментирането е механизъм за организация на виртуалната памет. Адресът се получава от 16-битов регистър и 32-битово отместване. В защитен режим сегментните регистри се наричат селектори, поради специалната роля която изпълняват. Полученият след сегментацията адрес се нарича линеен адрес и той е част от линейното адресно пространство.

предимства на сегментирането - възможността за точно управление на защитата, улеснява организацията и достъпа до реентрантните програми.

проблеми - например, ако в кодов сегмент има преход, където адресът на прехода се състои от номер на сегмент и отместване, тогава всички процеси трябва да използват един и същи номер за делимия сегмент, което води до трудности или програмата да е позиционно-независима.

**22. Обяснете преобразуването на логическия адрес във физически при сегментно страничната организация на паметта**

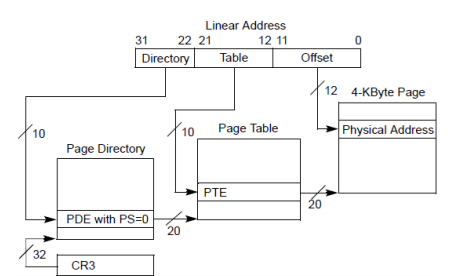
В режим на сегментация и странициране получаваме 32-битов линеен адрес. Първите 10 бита от него се умножават по ширината на полето на каталога (4 bytes) и се добавят към началния адрес на каталога. Ако старшият бит на реда в каталога е 1, значи че съществува каталог в паметта, който се сочи от реда в каталога. В такъв случай се взимат младшите 20 бита от реда в каталога, добавят им се 12 нули отдясно и се прибавя стойността на полето PN, умножена по широчината на реда в каталога. В реда в каталога (в младшите му20 бита) е записан адресът на търсената от нас страница, ако тя съществува. Последните 20 бита на реда в каталога съдържат линейния адрес на страницата, а физическият адрес на нейното начало се получава, като отново към тези 20 бита прибавим 12 нули отдясно и се прибави 12-битовият offset.

**23. Какви полета съдържа локалната дискрипторна таблица?**

Локалната дескрипторна таблица (Local Descriptor Table LDT) съдържа описание на сегменти, които са частни за дадена програма, докато глобалната съдържа глобални сегменти.

**24. Каква информация се съдържа в task register**

16-битовия сегментен селектор и целия сегментен дескриптор на текущата задача (32-битов базов адрес, 16-битова граница на сегмента и атрибути на дексриптора). Тази информация се копира от TSS-дескриптора в GDT за текущата задача.

**25. Как се осъществява преобразуването на линеен адрес към 4 –KB стр с използване на 32-битово странициране?**  


При този метод за адресиране може да се адресират до 220 страници, което прави 232 байта или 4GB. За да се избират различните редове от таблицата, линейният адрес се разделя на четири секции: • Ред в таблицата на главния каталог (битове от 30 и 31) – съдържа се отместването на всеки от четирите реда в таблицата на главния каталог. Избраният ред задава съответния физически адрес на каталога. • Ред в каталога (битове от 21 до 29) – съдържа се отместването на всеки от редовете в таблицата на каталогa. Избраният ред задава съответния физически адрес на каталога. • Ред в таблицата на страниците (битове от 12 до 20) – съдържа се отместването на всеки от редовете в таблицата на страниците. Избраният ред задава съответния физически адрес на страницата във физическата памет. • Отместване в страницата (битове от 12 до 20) – съдържа се отместването на физическия адрес в съответната страница.

**26. Обяснете структурата на задачата при IA 32 архитектура**

Задачата се състои от пространство за изпълнение (execution space) и сегмент за състоянието на задачата (Task State Segment TSS). Пространството за изпълнение се състои от сегмент за кода, стека и няколко сегмента за данни. Най-общо казано, TSS съхранява в паметта информацията, необходима за изпълнението (execution space) на дадена задача.

**27. Обяснете функцията на task register** Регистърът на задачата (TR) съдържа 16-битовия сегментен селектор и целия сегментен дескриптор на текущата задача (32-битов базов адрес, 16-битова граница на сегмента и атрибути на дексриптора). Тази информация се копира от TSS-дескриптора в GDT за текущата задача.

**28. Как се осъществява превключването на задачите?**

Всяка задача е изградена от два компонента – пространство за изпълнение на задачата и сегмент за състояния на задачата (TSS). В режим на многозначност TSS предоставя механизъм за свързване на задачите. Когато дадена задача се зареди за изпълнение в процесора, сегментният селектор, началният (базовият) адрес, границата и сегментния дескриптор на TSS се зареждат в регистъра на задачата – Task Register (TS).

**29. Опишете последователността от стъпки поради прекъсване от изключения ?**

Механизмът на прекъсване => няма зациклянето при проверка включено у-ство. Ако в/и у-ство е получило сигнал за започване => стъпка м/у: 0. Попълва някъде в регистър/паметта статуса, записва чрез флагове данни, праща сигнал за прекъсване към ЦП на предпоследна стъпка; 1. ЦП проверява дали сигнал за прекъсване – да => приема се, запазва се съдържанието PC и регистрите (на фиксирани адреси - разбиване предварително деф участъци, зависимост от кода апаратно се нанася състоянието); 2. Съхранение на векторния процес; 3. Разпознава се кодът на прекъсване - 2 начина: или по сигналната линия (пристига); или по линията за данни (задава (2) число - код на прекъсване); Пр: Нека кодът е i => таблицата прекъсванията копиране на i-тия ред в регистъра РС на процесора.;; Стъпките изпълнение хардуерно - авариен цикъл на процесора, след което нормален цикъл.; 4. Извличане от РС първата команда за изпълнение.

**30. Каква информация се съдържа в IDT?**

Interrupt Descriptor Table (IDT) e таблицата на векторите на прекъсванията. Тя има 256 елемента. Векторите на прекъсванията са входове към дескриптори на сегменти.

**31. Обяснете групите, на които се делят изключенията**

Aborts- За изключенията от тип abort не винаги се знае коя инструкция ги е предизвикала и не позволяват рестартиране на програмата или процесът, които ги причиняват. Използват се за регистриране на хардуерни грешки или недопустими стойности в системни таблици.

Faults - Изключенията от тип грешки (fault) като цяло могат да бъдат коригирани и за това след като се случат и бъде изпълнена обработващата програма, се стартира отново същата програма (която е генерирала грешката), като се счита, че грешката вече е коригирана (от обработващата програма).

Traps - Капанът (trap) e изключение, при което, след като се изпълни съответната програма за обработка, броячът на инструкциите се увеличава и сочи следващата инструкция. Т.е. текущата програма се прекратява и се продължава със следващата. По този начин например, когато стартирате програма, в която се появи изключение от този тип, то операционната система си връща управлението.

**32. По какво се различават изключенията от тип faults от тип aborts?**

За faults и traps се знае коя инструкция ги е причинила и програмата или процесът могат да бъдат рестартирани без загуба, докато при aborts това може да не е ясно и следва прекратяване на програмата или процесът.

Изключенията от тип грешки (fault) като цяло могат да бъдат коригирани и за това след като се случат и бъде изпълнена обработващата програма, се стартира отново същата програма (която е генерирала грешката), като се счита, че грешката вече е коригирана (от обработващата програма).За изключенията от тип abort не винаги се знае коя инструкция ги е предизвикала и не позволяват рестартиране на програмата или процесът, които ги причиняват. Използват се за регистриране на хардуерни грешки или недопустими стойности в системни таблици.

**33. Каква е фунцкията на челната част?** Челната част извлича и декодира инструкциите. Декодираните инструкции се транслират до микрооперации (μops). Основната задача на тази част е да осигурява непрекъснат поток от микрооперации на изпълняващата част.

**34. Как работи изпълняващата част?**

Изпълняващата част пренарежда микрооперациите по такъв начин, че тези чиито операнди са готови (и има налични изчислителни ресурси) да се изпълнят възможно по- скоро. Изпълняващата част може да обработи (issue) няколко микрооперации за един цикъл.

**35. Каква е ролята на завършващата част?**

Завършващата част се грижи резултатът от изчисленията да съответства на реда на постъпване на инструкции и че състоянието на системата се обновява правилно.

**36. Обяснете принципа на работа на кеш-паметта и протоколите за процесори**

Кеш-паметта представлява структура, състояща се от множество от рамки (frames). Всяка рамка се състои от данни, етикет (tag) и състояние. Етикетът е характеристика на рамката, която е съществена при търсене в данните в кеша. Състоянието е два бита – бит valid, който показва дали данните в кеша са валидни и бит dirty, който показва дали в рамката са били писани данни. Търсенето и зареждането на рамки в кеша става по следния начин: проверява се дали входящият етикет съвпада с етикета на рамката. Ако не съвпадат, рамката не се намира (cache-miss) и в такъв случай блокът данни в рамката се изхвърля и се заменя с блок от по-долното ниво в йерархията, а след това блокът се подава на предното ниво в йерархията.

Протокол на кеша MESI: използва се както в едноядрени, така и в многоядрени процесори. При този протокол се проверяват четири дефинирани състояния на всяка линия на кеша – (Modified, Exclusive, Shared, Invalid).

**37. Какво представлява и как работи предварителното извличане?**

Предварителното извличане на данните в кеша (Cache prefetching) е техника, която увеличава производителността, като копира данните от бавна памет в по-бърза, още преди да са необходими. Най-често се използват потокови буфери (Stream buffers), които се основават на последователното изпълнение на инструкциите и извличане на данните.

**38. Обяснете приликите и разликите м/у прекъсванията и изключванията**

При Intel x86 архитектурата прекъсванията и изключенията (interrupts vs exceptions) са две покриващи се групи векторизирани събития, които са индикация за наличието на състояние, което изисква обработка от страна на процесора.

Прекъсванията и изключенията карат процесора да скочи (jump) в програма, която обработва прекъсването (interrupt handler), след като запази контекста (за да може прекъснатата програма да продължи). Прекъсванията и изключенията имат идентификатор (ID), наречен вектор, които определя адреса на програмата за обработка на прекъсването. Тези програми (interrupt handler) се извикват през таблица на дескрипторите на прекъсванията

Прекъсванията се случват докато се изпълнява дадена програма в резултат на сигнали от хардуера. Прекъсванията се използват за да се обработят външни за процесора събития, като искане за обслужване на периферно устройство. Прекъсвания могат да се генерират и софтуерно, чрез изпълнение на инструкцията INT n, където n идентификаторът (ID) или наричан също векторът на прекъсването (от 0 до 255).

Изключенията се случват, когато процесорът детектира грешка (error condition) по време на изпълнение на инструкция, като например делене на нула, непозволен достъп (protection violations), липсваща страница в паметта (page faults) и др.

1. Каква система инструкции имат различните фамилии мултипроцесорни архитектури?

- съвпадаща отчасти

- изцяло различна

2. Единицата за измерване на производителността MIPS за съпоставяне на еднотипни процесорни архитектури се основава на:

- брой операции с операнди от виртуалната памет

Или

* … главната памет

3. Коя е правилната дефиниция на "SIMD"?

- Single Instruction Multiple Data

4. От кои фактори зависи производителността на компютърни системи?

- архитектурата и технологията

5. Колко проводна може да бъде USB платината?

- 4

- 9

6. Кое от изброените техники служат за увеличаване степента на паралелизъм по време на изпълнение на програмата?

- прогнозиране посоката на преходите

- едновремено зареждане за изпълнение на няколко инструкции и динамично планиране

- откриване и премахване на зависимостта по данни на компилиране

- реорганизация на циклите по такъв начин, че всяка итерация в получения код да се състои от инструкции...

7. За какво служи програмният брояч?

- за указване адреса на следващата инструкция за изпълнение

8. Какъв вид адресация е това? или Посочете за кой от изброените режими на адресация се отнася показаната фигура?

\_\_\_\_\_\_\_\_\_\_\_\_

|КОП|операнд|

-пряка

9. Как се оказва стойността на операнда при регистров режим на адресация?

- задава се с номера на регистъра

10. Коя схема адресира директен режим на адресация?

- ............

|

--------> памет (само операнд)

11. Кои от следните етапи са част от инструкционния цикъл?

- декодиране

- изпълнение

- извличане

12. Какъв вид адресация е тази?

(с кръгчето)

* Индиректна адресация с регистри

13. Кое не е част от инструкционния цикъл? Или Кой от изброените етапи НЕ е част от инструкционния цикъл?

- пренареждане

?. Кои от изброените са фази на инструкционния конвейер?

* Извличане
* Изпълнение
* Декодиране

14. Коя схема е на адресация с регистри?

........

|

-----> регистри --(без кръгче)--> памет

15. Как се указва ефективният адрес на операнда при непосредствен режим на адресация?

- чрез адресното поле на операнда в кода на инструкциите

16. Как се подредени адресациите по време МНОГО КРАТКО -> МНОГО ДЪЛГО?

- непосредствена, пряка, косвена

17. Къде е местоположението на операнда при регистрово-косвена адресация?

- в основната памет

18. При какъв вид адресация на инструкцията е записан адресен указател?

- индиректна

19. Причината за понижаване на тактовата скорост при конвейерна обработка е: или Посочете кои от изброените фактори са причина за понижаването на потоковата скорост при конвейерната обработка:

- зависимостта по данни

- конфликти при достъп на общи ресурси

- анти-зависимост по данни

- преходи в информационния поток, засягащи програмния брояч

20. Посочете при конвейерна обработка …. "write-after-read"

- ADD R3,R2,R1 : R3 = R2 + R1

SUB R2,R3,1: R2 = R3 - 1

ADD R3,R2,R5: R3 = R2 + R5

21. Защо реалното ускорение е по-малко от теоретично допустимото?

- заради хазартите

* заради времето, необходимо за изпълване на конвейера

22. "write-after-write"

- ADD R3,R2,R1

SUB R2,R3,1

ADD R3,R2,R5

23. Разделянето на инструкциите по определени фази на обработка е характерно за:

- паралелизъм на нови инструкции

- конвейерна обработка

24. Колко машинни инструкции трябват за обработката на 1500 инструкции в 5-фазен линеен конвейер?

- 1504

25. На каква концепция се базират инструкционните конвейери?

- паралелизъм на ниво инструкции

26. На какво се дължат процедурните зависимости?

- инструкции за безусловен преход и инструкции с осъществен условен преход в

инструкционния поток на програмата

27. От какво се предизвикват ресурсните конфликти?

- едновременни заявки за едни и същи ресурси

28. В инструкционния конвейер се изпълняват едновременно:

- инструкции в различни фази

29. Процесори на RISC архитектура: или Кои процесори от изброените процесорни фамилии се характеризират с RISC-архитектура?

- ARM

- MIPS - R4000

- SPARC

30. Какво означава CISC?

- Complex Instruction Set Computers

31. Кои инструкции са готови за изпълнение? (ГОЛЯМА ТАБЛИЦА)

- Opcode 3

- Opcode 4

32. Какво е предназначението на механизмите за защита на паметта?

- да ограничават достъпа до определени сегменти или страници

33. Колко е максималният брой страници с размер 8 при 8-битов адрес?

- 32 (8-битов=>2^8=256 max адреси и 256:8 = 32 =>8 страници)

34. При размер на страниците 4К и 32-битов адрес, колко е мах брой страници?

- 1М

35. Кой от посочените методи не е метод за преобразуване на виртуални адреси?

- метод на локалността (по време и пространство)

36. Колко проводна може да бъде USB шината?

37. При размер на страниците 8 и 8-битов адрес преобразувайте логическия адрес 18 във физически

- 2\*8 + 3

38. Какво е предназначението на TLB-буфера?

- да минимизира загубата на бързодействие при търсене на физическото разположение на данните

39. Какво съдържа TLB-буфера при странична организация на паметта?

- моментна карта на съответствието на страниците от оперативната памет, принадлежащи на даден процес

40. Коя от изброените характеристики НЕ описва таблицата на страниците?

- … съдържа информация за активните процеси

41. От какъв тип е следната инструкция?

ADD 5,12,20

- CISC

42. При размер на страниците 8 и 8-битов адрес, кой ред на таблицата съдържа ареса на страницата

- 3

43. Кои от изброените техники се използват при разрешаването на проблеми при конвейерното изпълнение по инструкции с преход?

- Prediction Look-up Table

- BTB буфер

Или

* Множествено предварително ….

44. От какво се определя максималната ....... скорост на конвейера?

45. Кои са характеристики на микроархитектурата?

-размер на чипа

-консумация на ел.енергия

-дълбочината на конвейера

46. Ако имаме кеш с 4 линии и блокът е с размер 2^6(64), какъв е размерът на кеш?

- 4х64

47. Какви са предимствата на кеш с пълна асоциативност?

- Когато се напълни целия кеш, чак тогава трябва да се махне от него

48. Какви са недостатъците на кеш с пълна асоциативност?

- По-трудно е самото търсене (трябва да се търси навсякъде)

49. При кеш от вид Direct Map с размер на блока 8 и 4 линии, адрес 22 в коя линия попада?

- ако имаме пълна асоциативност: всяка

- ако имаме директна асоциативност: 2

50. Имаме кеш с пълна асоциативност и адрес 18

- ако данните са в кеш: HIT

- ако данните НЕ са в кеш: MISS

51. таблица

0 |tag | |

1 |tag | |

2 |tag | |

9 |tag | |

- при директна асоциативност | |index|offset|

- при пълна асоциативност | |offset|

52. Стъпки при търсене в кеш ->Direct Map

1) гледаме полето INDEX неговият TAG се търси в линия която отговаря на индекса

2) гледаме TAG. Ако в кеш няма нищо отговарящо на търсенето, отиваме в паметта. Ако е в кеша, то сме го намерили.

3) гледаме отместването и се извличат данните от кеша

53. Обновяване на паметта

1) write back ->записът се случва синхронно в кеш паметта

2) write through ->първо се записва в кеш, а записване в паметта се отлага

3) no write allocate ->не променя кешовете, а директно се пише в паметта

4) write allocate ->ъпдейтват се кешовете, а после и паметта

54. Кой от трите метода на разполагане на блокове от оперативна памет се имплементира най-лесно?

- директно(пряко) съпоставяне

55. Кое е невярното твърдение? или Отбележете невярното твърдение?

- SRAM е по-евтина от DRAM

56. Каква е формата на линията при кеш памет с множествено-асоциативна организация?

- таг/многжество, дума

57. Каква е формата на линията при кеш памет с пълна асоциативност?

- таг, дума

58. В режим на директен достъп до паметта управлението на обмена се поема от?

- устройството за вход/изход

59. Каква е корелацията между капацитет(размер) и бързодействие при компютърните памети?

- обратнопропорционална

60. Кой вид памети са по-бързи от кеш паметите?

- регистрите

61. При кой/кои от посочените методи на запис в кеш се генерира значителен трафик по шината?

- write-back

62. Методът, при който обновяване на главната памет настъпва след премахване на думата от кеша се нарича:

- write-back

63. Дадена е машина с побитово-адресируема основна памет с размер 2^16 байта и размер на блока 8 байта. Кеш паметта е с директна организация и се състои от 32 ...

Какъв е максималният брой байтове, които могат да бъдат разположени в кеш паметта?

- 256 В

64. LFU е алгоритъм за:

- заместване на най-неизползваната страница в кеш паметта

65. Дадена е машина с побайтово-адресируема основна памет и 2-кратен множествено-асоциативен кеш. Кеш логиката интерпретира адреса от паметта както следва:

14 бита за таг, 8 бита за множество, 2 бита за адресиране на думата

Колко на брой са блоковете в главната памет?

- 2^22

66. Кеш памет с асоциативна организация е с кашацотет 64 линии, разделени в множества с по 4 линии всяко. Основната памет съдържа 4К блока с по 128 думи всеки.

Посочете адресния формат който й съответства

- 14|8|7

67. Метод, при който запис на данните се извършва само в кеш паметта, а състоянието на кеш блока се отбелязва като обновено в .....

-

68. Дадена е машина с побайтово адресируема основна памет 2^24 байта и с кеш на данни с директна организация и с капацитет 64К и 32-байтови блокове. Колко бита са необходими за таг, блок и отместване

- 8/11/5 …. (таг/блок/уникална дума)

69. Кое от изброените не е вярно за кеш паметта?

* Намира се в адресното пространство....

70. Кои от изброените подходи способстват за намаляване на латентността(закъснението) при зареждане на данни в кеш паметта?

- "зареждане при поискване"

- "спекулативно зареждане"

- "отложен запис на данни"

71. Шината PCI се явява мост м/у:

- шина EISA и периферията

72. Коя/кои от изброените шини не е част от системната ....?

- шина за логически сигнали

73. Кое от изброените характеризира архитектурната концепция EPIC?

- оползотворяване на явен паралелизъм на ниво инструкции

- оползотворяване на "скритият" в програмния код на ниво компилатор и апаратна поддръжка

- използване на широка шина за зареждане на инструкции (wide-issue-width) и дълги конвейери с голяма задръжка (deep pipeline latening)

74. Посочете за кой от изброените режими на адресация се отнася дадената фигура

КОП|адрес на регистър|адрес в паметта

| | памет

регистри ----> кръгче ----> операнд

- адресация с изместване

75. Кой от посочените методи се използва за решаване на проблема с инструкционните хазарти?

- пренареждане на инструкциите от компилатора

76. Как се преодоляват конфликтите за ... при конвейеризираното изпълнение на инструкционния поток:

- чрез предсказване на преходите в инструкционния поток

77. Суперскаларен процесор обработва два инструкционни потока (от прости и сложни инструкции). Коя от изброените гази на обработка се явява обща и за двата потока?

- извличане

78. Какви типове памет се използват за съхранение и поддържане на таблиците на страниците при странична организация на паметта

- асоциативни кешове

- множествено-асоциативни кешове

79. По какво се отличават RISC от CISC архитектурите?

- Всички инструкции се характеризират с опростен формат

80. При кои от трите метода на разполагане блоковете от оперативна памет в кеша е най-малко вероятно два блока да бъдат записани на една и съща линия?

* пълна асоциативност

81. Кое от твърденията относно размера на страниците е невярно?

* По-големият размер на страниците увеличава времето …...

82. Каква е формата на линията при кеш памет с асоциативна организация?

* Таг, дума

83. Как се преодоляват хазардите в конвейера при достъп до общи ресурси в инструкционния поток?

* Чрез блокиращи...
* Чрез въвеждане...

84. Методър, при който запис на данните се извършва само в кеш,….

* Обратен запис (write-back)

85. В зависимост от режима на адресация времето за изпълнение на една процесорна инструкция може да варира в значителна степен. При кои от изброените подредби на адресни режими времето ще се изменя от много кратко до много дълго?

* Непосредствена, пряка, косвена

86. Основни производители на х86 архитектурата са:

* IBM
* Hawlett-Packard
* Intel

87. При кой от изброените методи на адресация в адресната част на инструкцията е записан адресен указател?

* Индиректна адресация

88. При размер на страниците 8 и 8-битов адрес колко е максималният брой страници?

* 32