1. Какъв е начина на работа на SIB byte?
2. Опишете схемата на паметта при big endian/little endian
3. Какви са предимствата на пакетираните SIMD данни?
4. Опишете полетата, които дават формат на инструкциите?
5. За какво и как се използва полето prefix?
6. Обяснете приложението на Rex-prefix
7. GDTR и IDTR регистри
8. Защитен режим
9. Сегментни селектори
10. Какъв е механизмът на обновяване на страница от диска?
11. Обяснете структурата на задачата
12. Функцията на регистъра на задачата
13. Как се осъществява превключването между задачите?
14. Какво се случва при прекъсване?
15. Какви са групите, на които се делят изключенията?
16. Прилики и разлики между прекъсване и изключение
17. Прилики и разлики между fault и abort
18. Каква е функцията на челната част?
19. Роля на завършващата част
20. **Какъв е начина на работа на SIB(Scale-Index-Base) байт**

Scale|Index|Base Два бита за Scale, 3 бита за индексен регистър и три бита за базов регистър. Съдържанието на индексния се умножава по 1, 2 и т.н. в зависимост от Mod и се събира с базовия регистър. offset = Base + Index \* Scale + Displacement, ------ Отместване (Displacement) Размерът на отместването се определя от полето Mod на ModR/M байта

1. **Опишете схемата на паметта при Big Еndian / Little Еndian**

**Big Indian-** когато сме във този формат, при обръщане към паметта или изпращаме/запазваме думи побитово, първо записваме старшия бит, останалите битове се записват в намаляващ ред, като младшият бит се записва последен.  
**Little Indian-** този формат обръща подредбата: всички операции първо запазват младшия бит, и последно запазват старшия.

1. **Какви са предимствата на пакетираните SIMD данни?**

Информацията се приема като блокове и няколко блока могат да се заредят наведнъж, вместо един по един, и да се извърши една и съща операция върху тях. Това ускорява работата тъй като не губим време да взимаме информацията част по част, а взимаме повече наведнъж. Друго предимство на SIMD е че изпълнява желаната операция едновременно върху заредената информация.

1. **Опишете полетата които дават формат на инструкциите**

Инструкциите биват в няколко различни формата - тоест, възможни дължини и от какви полета се състоят. Теоретично една инструкция се състои от 4 полета, а именно код на операцията, адрес на операнд 1, адрес на операнд 2 и адрес на резултат. Като поле в инструкцията може да се сложи стойноста на операнда, но това е неудобно, защото инструкцията е трудна за промяна. Затова е широко прието да се работи с адреси. Като поле в инструкцията може да се сложи стойноста на операнда, но това е неудобно, защото инструкцията е трудна за промяна. Затова е широко прието да се работи с адреси.

1. **За какво и как се използва полето prefix?**

Чрез prefix може да се променя поведението на инструкция, като например да

се сменя сегмента, размера на думата, да се управлява цикличното изпълнение и

използването на шината.

Prefix-овете не са задължителни и се използват когато на основното поведение на инструкция не му достига функционалност. Те се използват, когато искаме да увеличим функционалността на инструкцията или да сменим параметрите и.

1. **Обяснете приложението на REX префикс**

64 битовия режим, към формата на инструкцията, се добавя REX префикс Не всички инструкции в 64 битов режим изискват REX префикс. Той е необходим единствено, ако инструкцията реферира някой от разширените регистри или използва 64 битови операнди. REX префикса съдържа битове, които разширяват ModR/M полетата. По този начин ще работят и старите и новите програми, които използват повече регистри.

1. **GDTR и IDTR**

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. instructions load and store the GDTR register. 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.  
 Interrupt Descriptor Table Register (IDTR) - съдържа базовия адрес (32 бита в защитен и 64 бита в IA-32e режим) и размера (16 бита) на таблицата на прекъсванията IDT.

1. **Защитен режим**

В защитен режим, винаги се използва някакъв вид сегментация и тя няма възможност да се забрани, докато използването на страницирането не е задължително и може да се конфигурира. При защитен режим сегментният селектор се взема от сегментния регистър. Този сегментен селектор съдържа индекс в главната дескрипторна таблица, а съответният дескриптор съдържа адреса на сегмента. Към този адрес се добавя ефективния адрес от инструкцията (отместването) и се получава линейния адрес

1. **Сегментни 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.

1. **Механизъм за обновяване на страница от диск**

CR2 – Съдържа линейния адресът, който е причина за изключение от тип „page fault“. Такова се получава при адресацията на някои от операндите се попадне в страница, която не се намира в паметта. Този адресът, които не успешно адресиран е необходимо да се запише някъде, защото следва прекъсване, което извиква програма, която трябва да вземе страницата от диска.

1. **Структура на задачата**

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. С други думи, в TSS (Task State Segment) се съхранява състоянието/контекста на задачата/процеса. Тази структура описва сегментите, управляващият регистър и общите регистри на задачата.

Всяка задача се определя от селектор, към съответния и сегмент за състоянията TSS. Когато дадена задача бъде заредена в процесора за изпълнение, сегментния селектор, размера на сегмента и други атрибути на TSS се зареждат в регистъра на задачата (task register).

1. **Функцията на регистъра на задачата**

Той съдържа 16-битов сегментен селектор и цялата структура на сегментния дескриптор.(има и някаква схема ама по-добре да не я виждате :D) (Регистъра на задачата има 2 части: част видима и достъпна за програмиста и невидима част, която се зарежда автоматично от TSS дескриптора (сегментния дескриптор). - това от wikipedia)

1. **Как се осъществява превключването между задачите?**

Превключването към дадена задача се осъществява в един от следните четири случая:

• Текущата програма, задача или процедура изпълнява JMP или CALL инструкция към TSS дескриптора в GDT на дадената задача.

• Текущата програма, задача или процедура изпълнява JMP или CALL инструкция към TSS - gate дескриптор в GDT или текущата LDT на дадената задача.

• Векторът на прекъсване или изключение сочи към TSS - gate дескриптор в IDT.

• Текущата задача изпълни IRET инструкция, при условие че флагът NT в регистъра EFLAGS има стойност 1. При превключването към друга задача, състоянието на средата, в която се е изпълнявала текущата задача (състояние на задачата или още контекст на задачата) 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.

1. **Какво се случва при прекъсване?(не съм сигурен, това намерих само)**

Съдържанието на флаговия регистър, Code segment (CS) и Instruction pointer(IP) се добавят към stack-a. Изчистват се IF и TF. Програмата „скача“ към началото на ISS(Interrupt service routine – би трябвало да е синоним на interrupt handler). Когато IRET се изпълни накрая на ISS, CS и IP се изваждат от stack-a. След като всички флагове са върнати към началното си състояние, IF и TF също се връщат към предишните си стойности.

1. **Какви са групите, на които се делят изключенията?**

Изключенията се делят на **faults, traps, и aborts** в зависимост от начина по който се отчитат и дали инструкцията, която ги е причинила може да бъде рестартирана. 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 прекратява на програмата или процесът.  
**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.

1. **Прилики и разлики между Прекъсвания и Изключения**

**Прилики:**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

**Разлики:**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. Изключенията се случват, когато процесорът детектира грешка (error condition) по време на изпълнение на инструкция, като например делене на нула, непозволен достъп

1. **Прилики и разлики между fault и abort(не видях прилики, само разлики)**

**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). Например изключение от тип page fault обикновено се коригира. Получава се когато програмата адресира пространство от RAM паметта, което е било прехвърлено на диска (Виж темата за виртуална памет). Тогава настъпва прекъсване и се извиква съответната програма, която да го обработи, а в случая да зареди липсващата страница от диска обратно в паметта. След това от ново се извиква програмата, която е предизвикала page fault, но този път достъпът до съответната страница в паметта трябва да е успешен. **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.

1. **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. **Каква е функцията на завършващата част**

Завършващата част се грижи за това, резултатът от изчисленията да съответства на реда на постъпване на инструкции и че състоянието на системата се обновява правилно. След оттеглянето (Retirement) на инструкциите за запис, данните се записват в кеш паметта от първо ниво. Следва да се отбележи, че за един цикъл може да се оттеглят (Retire) до три микроинструкции.

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

Сегментната преадресация или сегментация предоставя механизъм за изолация на кода, данните и стековите модули, така че множество програми/процеси да могат да се изпълняват на един и същи процесор, без да се месят взаимно.