Instuderingsfrågor \*Källor sista sidan

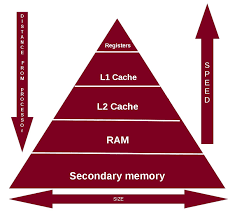
# Processorn

1. Vad är Moores lag?  
   **Moores lag** är observationen att **antalet transistorer** på en processor **dubbleras vartannat år**.
2. Vem är Von Neumann?  
   **John von Neumann** var en forskare inom datorteknik och la **grunden för** det **gemensamma minnet för instruktioner och data**. Datorarkitekturen gick ut på att se program och data som samma sak när de placeras i minnet.
3. Vad gör en kompilator?  
   **Kompilatorn** **översätter** från ett **högnivåprogramspråk** till ett annat programspråk med **lägra nivå**, ofta assembler eller direkt till maskinkod.
4. Vad gör en assemblator?  
   **Assemblerar** assemblerkod till maskinkod.
5. Ge exempel på högnivåspråk.  
   **C och C++** (som kompilerar till assembler eller maskinkod). **Java** (kompilerar till bytekod som sedan kör på JVM (Javas virtuella maskin)).
6. Vad skiljer ett högnivåspråk från ett maskinspråk?  
   **Högnivåspråk** är **lättare** för **människor** att skriva och felsöka medan assemblerkod tenderar att vara jobbigt att utveckla och förstå. Men **datorn** kan inte direkt läsa från ett högnivåspråk, utan **kräver** att det genomgår kompilation till **maskinspråk** (eller bytekod om det ska köras på JVM).
7. Görs alla beräkningar (+,-,…, AND, OR) i ALU:n?  
   **Nej,** **inte** de som involverar **flyttal**. Annars görs både aritmetiska och logiska instruktioner i ALU:n.
8. Ge exempel på indata och utdata till en kontrollenhet.  
   **Input:** en **instruktion** från **minnet**.  
   **Output:** avkodad **instruktion** till **ALU:n**.
9. Ge exempel på fördelar med att använda register för att lagra data.  
   Data i **register hämtas/sparas snabbare** än data i t.ex. minnet, eftersom register sitter **fysiskt närmare CPU:n**. Med pipelinearkitektur kan register användas av flera instruktioner samtidigt (går inte med t.ex. primärminnet).
10. Om en processor gör ”Fetch” och ”Execute”, vad görs under ”Fetch” resp ”Execute”? Är det som gör under ”Fetch” samma för alla instruktioner?  
    **Fetch:** Hämtar instruktion från adress xxx  
     Flyttar instruktion y0y1y2 från plats xxx till CPU  
    **Execute:** Avkodar instruktionen y0y1y2  
     Hämtar data från adressen y0 Lagrar data i register y2

# Pipelining

1. Vad är pipelining?  
   **Pipelining** är när processorn kör flera **olika instruktioner samtidigt** genom att tillåta en instruktion i varje fas (t.ex. en i ”Fetch” och en i ”Execute”). Man kan dock alltid enbart exekvera en instruktion i taget.
2. Vilka konflikter kan uppstå i en pipeline?  
   **Strukturella konflikter, datakonflikter, kontrollkonflikter.**
3. Illustrera hur konflikter uppstår.  
   **Strukturella konflikter**  **–** När två pipelineade instruktioner försöker använda samma hårdvarukomponent (ex minnet) samtidigt.  
   **Datakonflikter –** När en instruktion hämtar data från ett register som nyss ändrats och inte hunnit få sitt nya värde.  
   **Kontrollkonflikter –** När man ska brancha och inte vet ifall branchningen gick igenom, vilket resulterar i att man inte vet vilken instruktion som ska hämtas.
4. Vad ska man göra för att undvika konflikter?  
   **Strukturella konflikter** undviker man genom att inte sätta instruktioner som vill komma åt minnet nära varandra.  
   **Datakonflikter** undviker man genom att separera data- och instruktionscache, eller placera om.  
   **Kontrollkonflikter** undviker man genom stallande (**nop**-instruktioner), delayed branchning och branch prediction.
5. Vad är branch prediction?  
   Processorn gissar på att branchen kommer hoppa eller inte och hämtar nästa instruktion utan att **stalla**, gissar man fel blir det lite extra **straffcykler**.  
   **Statisk prediktion –** tar man *inte* hänsyn till historiken.  
   **Dynamisk prediktion –** tar man hänsyn till historiken.
6. Vad är spekulativ exekvering?  
   När processorn börjar exekvera instruktionerbaserat på branchgissning gjord av en **branchprediktion** och utan kännedom ifall instruktionen är rätt eller inte.
7. Vad är delayed branchning? Vinst? Ge ett exempel.  
   När man inte vill **stalla** och istället automatiskt kör nästa instruktion som ligger efter hoppet. Instruktionen som hade utförts vare sig branchen hoppar eller inte. Sen sätter man instruktionen efter branchen i en **branch delay slot**. Detta blir ett **kostfritt stallande** så CPU:n hinner komma på om den ska hoppa eller inte utan kostnad.  
     
   **Alt:** När instruktioner körda innan branchning, som inte kommer påverka om branchen kommer genomföras eller ej, läggs precis efter branchningen så stall/nop inte behövs.
8. Vad är delayed load? Vinst? Ge ett exempel.
9. Ge exempel på en komplatorteknik som kan användas för att undvika/hantera konflikter i pipelinen.  
   **Delayed branchning** är en kompilatorteknik som undviker **kontrollkonflikter.**

# Minnet

1. Hur lagras information på en hårddisk?  
   En **hårddisk** (ej SSDs) lagrar information på **magnetiska skivor** som läses med ett läshuvud. Informationen lagras i **Words** som ligger i **blocks**.
2. Vad är random access när man talar om minnen?  
   Ett **volatile primärminne**, där alla **sektorer** på minnet tar ungefär **lika lång tid** att hämta.
3. Ge exempel på minne som inte har random access.  
   Alla non-volatile minnen, vilket är sekundärminnen, där olika sektorer tar olika lång tid att hämta beroende på sektorns avstånd från läshuvudets tidigare position.
4. Vad är en minneshierarki?  
   Ett system där man har   
   snabba och **små minnen**   
   högst upp (**närmast   
   CPU:n**) och långsamma   
   och **stora minnen** längst   
   ner (**längst ifrån CPU:n**).
5. Varför uppstår en minneshierarki?  
   Eftersom vi **inte** kanha båda **stora och snabba** minnen. Kompromissen blir att man lägger mindre, snabbare minnen nära CPU:n för att öka snabbheten/ minska accesstid och större, långsammare minnen längre ifrån CPU:n.
6. Vad kallas principen som gör att cacheminne fungerar? Förklara och ge exempel.  
   **Lokalitetsprincipen –** Med två subprinciper: Om instruktion/data blivit refererad nu, så är sannolikheten stor att ...  
   *Temporal lokalitet* – … samma referens görs snart igen.  
   *Rumslokalitet* – … instruktioner/data vid adresser i närhet kommer användas snart.
7. Vad är en cachemiss? Varför uppkommer cachemissar? Hur hanteras det?  
   En **cachemiss** är när du **söker** efter något **i cacheminnet** och inte hittar det, och du måste **stoppa in** det i cachen **från RAM-minnet**.
8. Cacheminnen kan ha olika mappning – vilka? Hur fungerar varje mappning?  
   **Direktmappning –** Man placerar instruktionen på *nästan* samma plats som den är på i minnet, platsen kallas cacheindex. Placeringen är instruktionens **adress** i minnet **modulo cachestorlek**.  
   **Associativ-mappning –** Man fyller cacheminnet på **följd** och använder **ersättningsalgoritm** vid cachemiss i ett fullt cacheminne.  
   **Set-associativ-mappning –** Detta är en **blandning** mellan **direkt**- och **associativ-mappning**. Man delar in cacheminnet i flera *sets*, mappar vissa instruktionsadresser till dessa sets, och använder associativ-mappning inom varje set.
9. I direktmappning, hur ersätts cacherader vid cachemissar?  
   De **skriver över** med den nya informationen vare sig det fanns något eller inte.
10. Vad är ersättningsalgoritm?  
    En algoritm som används vid associativ-mappning för att bestämma vilken/vilka cacherad/cacheblock som ska kastas. Vi har lärt oss tre:  
    **Least recently used (LRU) –** data i cachen som **inte** blivit **refererad** på länge.  
    **Least frequently used (LFU) –** data i cachen som **refererats** mest **sällan**.  
    **First in, first out (FIFO) –** data i cachen som **existerat längst** i cacheminnet.
11. Vad menas med att cacheminnet inte är konsistent? Hur hålls det konsistent?  
    Att **cacheminnet** ska **matcha** informationen i **sekundärminnet** och vice versa.
12. Antag ett program som exekverar alla instruktioner i en sekvens (en i taget) och att det finns ett cacheminne för instruktioner där cacherader har storlek 64 bytes och varje instruktion kräver 2 bytes. Vad är sannolikheten för att nästa instruktion finns i samma cacherad som förra instruktionen?  
    Antalet instruktioner per cacherad är *64/2=32*, vilket innebär att sannolikheten är *1/32* att nästa instruktion ligger på nästa cacherad. Det omvända fallet blir då *1-(1/32) = 31/32* dvs. **sannolikheten** är **31/32***.*
13. Vad är fördelen med paging? (OBS: *”Vad är paging?”* kommer längre ner)  
    Att man dynamiskt **laddar** in **data** i minnet **på begäran** och **laddar ur** det när det **ej används**.
14. Vad är nackdelar med paging?  
    Tidskrävande att ladda in och ur sidor ur minnet.
15. Vad är fragmentering när vi pratar om paging?  
    Att man delar upp det(?) i t.ex. process Ax på samma ställe i den fysiska adressen, kallas **fragmentering**.
16. Vad är skillnaden på extern och intern fragmentering?  
    **Intern:** när man **allokerar** och inte använder kapaciteten av hela **clustrar/block**.  
    **Extern:** när det uppstår icke-allokerade luckor i minnet mellan allokerade segment, som gör **clustrar/blocken** mindre.
17. Vad är paging?  
    Paging delar upp primärminnet i ramar (frames) och program i sidor (pages). En **page** är en samling instruktioner till en process. **Paging** är en metod för att **ladda** in program och data från **sekundärminnet till primärminnet**.
18. Vad är en sida (page), ram (frame)?  
    Primärminnet är uppdelat i **frames – fysiska platser,** som kan fyllas med **pages** **– samling med instruktioner till en process**.
19. Om en sida är 2 kB, kan man säga något om storleken på primärminnet eller en ram?
20. Vad är demand paging?  
    **Demand paging** är när sidor endast laddas från aktiva program när deras innehåll efterfrågas och aldrig innan till **primärminnet**.
21. Vad är trashing? När uppkommer det?  
    **Trashing** uppkommer när ett program friar upp ramar som regelbundet används. Detta kräver flera cykler av att ladda in och ladda ur under kort tid, vilket resulterar i att **operativsystemet** spenderar större delen av sin tid på att ladda in och ut sidor från **cachen**.
22. Vad är skillnaden på paging och virtuellt minne?  
    **Virtuellt minne** mappar en process virtuella adressintervall till sidor i primärminnet eller data i sekundärminnet. **Paging** hanterar in- och urladdning av data direkt från sekundärminnet till primärminnet.
23. Vad är sidfel?  
    **Sidfel** är ett avbrott som sker när ett program försöker komma åt data som inte ligger i RAM-minnet/virtuell minnescell.
24. Vad händer vid sidfel? Hur hanteras det?  
    Operativsystemet fångar upp avbrottet genom att leta upp datan i sekundärminnet, skaffar sig en tom ram och lagrar datan i ramen.

# Operativsystemet

1. Vad gör ett operativsystem?  
   Kärnan hanterar hårdvaruresurser som:
   * Processhantering (process management)
   * Avbrott (interrupts)
   * Minneshantering (memory management)
   * Filsystem (file system)
   * Drivrutiner (device drivers)
   * Nätverk (Networking)
   * Säkerhet (Security)
   * In- och utmatning (I/O – input/output)
2. Vad är multitasking?  
   Att flera program körs samtidigt (dvs. hoppar fram och tillbaka mellan instruktioner i olika program).
3. En användare känner att flera program exekverar samtidigt, hur är det möjligt?  
   Schemaläggaren tilldelar **klockcykler** på ett sådant sätt att användaren inte märker av att processorn hoppar mellan instruktioner, detta på grund av att **exekveringstiden** är väldigt liten.
4. Vad är ett kontextbyte?  
   När man byter exekverande program (vilken tråd som för stunden exekverar).
5. Hur går kontextbyte till? Hur vet man om att det ska ske? Inblandade faktorer?  
   När **schemaläggaren** beslutat ett byte mellan två trådar så sparas först **tillståndet** för nuvarande tråd, sen sätter man **pc** (programräknaren) till den andra trådens **kontext**. Sen byter man tillbaka till den första trådens kontext.
6. Behövs avbrott för att klara av att göra kontextbyten?  
   **Nej**, man kan göra kontextbyte med **polling**.
7. Hur fungerar avbrott?  
   Avbrott är en **mekanisk signal** (1:or och 0:or) skickas till processorn som direkt hoppar till **avbrottshanteraren**.
8. Om man skapar en struktur för att lagra filer, vad vill man uppnå?  
   Kunna **strukturellt** **allokera** utrymme åt filer på disken, för att kunna nå data **snabbare** och **lagra stora filer**.
9. Om man ska läsa in en fil från en hårddisk, vad påverkar lästiden?  
   **Storleken** på **hårddisken** och hur filerna är lagrade (fragmentering och filsystemet).

# ExTentor

## Extenta 1

1. Uppgift 1
   1. Vad gör en kompilator som indata och vad genererar den som utdata?  
      **Mitt svar:** En kompilator tar in högnivåprogramspråk och genererar ett annat programspråk på lägre nivå, ofta assembler, bytekod (till JVM) eller direkt maskinkod.  
      **Svar:** En kompilator tar som indata ett program (källkod) beskrivet i ett högnivåspråk och genererar som utdata maskininstruktioner för en specifik processor.
   2. Vad gör en kontrollenhet under den tid som kallas FETCH?  
      **Mitt svar:** Kontrollenheten hämtar en instruktion från adress xxx.  
      **Svar:** FETCH delen för en instruktion är gemensam för alla instruktioner, dvs. det är samma sak som görs. Det som görs är att:
      1. ta värdet som finns i programräknaren
      2. läsa den data som finns på den plats som pekas ut av programräknaren
      3. räkna upp programräknaren
      4. lagra instruktion som lästs in i instruktionsregister (så att instruktion kan avkodas)
   3. Hur påverkar implementationen av kontrollenheten prestandan i processorn?  
      **Mitt svar:** Ifall kontrollenheten är implementerad så att det tar onödigt många klockcykler för att utföra en instruktion måste processorn arbeta mer än vad den annars behövt göra.  
      **Svar:** Kontrollenheten bryter ned det som ska göras för en maskininstruktion i mindre delar (steg). Ju färre steg som används, ju snabbare exekveras instruktionerna. Det ställer krav på att mer saker kan utföras parallellt i varje steg, vilket ökar kostnaden för hårdvara.
2. Uppgift 2
   1. Beskriv vad som görs vid ett kontextbyte.  
      **Mitt svar:** Vid ett kontextbyte sparar man först det nuvarande tillstånd man befinner sig, sedan sätter man programräknaren till den nya kontexten och därefter ställer man tillbaka programräknaren till det första kontexten.  
      **Svar:** Vid ett kontextbyte byter en process exekvering av ett program (process/tråd) till ett annat program. För att göra detta möjligt måste information, t.ex. register innehåll, sparas i t.ex. ett process control block och registren måste laddas med information från det program som ska påbörja exekvering (från dess process control block).
   2. Hur bestäms det när kontextbyte ska ske?  
      **Mitt svar:** Schemaläggaren bestämmer när ett sådan byte ska ske. Därefter byter processorn den exekverande tråden till den andra tråden.  
      **Svar:** Kontextbyten sker vid avbrott. Dessa avbrott kan genereras av en extern klocka eller när ett program vill göra ett systemanrop för att göra t.ex. en I/O operation.
   3. Varför används en så kallad translation look-aside buffer (TLB)?  
      **Mitt svar:** Eftersom man vill översätta kod i buffertern i förväg.  
      **Svar:** Om minnet använder paging så behövs en sidtabell som håller ordning på var en sida (page) ligger i minnet. Denna sidtabell måste lagras någonstans. Om sidtabellen lagras i primärminnet kommer varje läsning/skrivning till primärminnet ta två accesser. Först en för att läsa sidtabellen och sedan en access för att läsa/skriva datan. Om man använder en TLB, dvs. en cache för sidtabellen, så sparar man tid för läsning av sidtabell.
3. Uppgift 3
   1. Assembly instruktioner för att adressera data använder olika adresseringsmöjligheter (moder). Vad kallas den typ av konflikt som instruktioners adressering kan påverka i en pipeline?  
      **Mitt svar:** N/A

# Källor

**Varning:** Dessa källor med svar är skrivna av studenter och är därför inte garanterat rätt och absolut inte förklarat på bästa sätt, förbättringar välkomnas!

1. <http://datateknik-lth.github.io/courses/EIT070-datorteknik/instuderingsfragor-med-svar/>
2. Amelia Anderssons svar
3. Amelias svar