Lite information om minnen och pipelining

Internt och externt RAM-minne

- Internt RAM-minne används ofta för att lagra data med låg åtkomsttid, där data kan läsas in inom en klockcykel. En nackdel med internt RAM-minnet är att det normalt sett är relativt dyrt och mängden begränsas därmed. Behövs mer minne kan med fördel någon typ av externt RAM-minne användas.

- Externt RAM-minne används ofta när en högre mängd minne behövs utan att kostnaden ska bli för hög. Fördelen med externa RAM-minnen, såsom SDRAM-minnet, är alltså låg kostnad. Externa RAM-minnen medför dock att åtkomsttiden ökar med en faktor 5 – 10, vilket medför minskad prestanda, då processorn inte hinner läsa in instruktioner lika snabbt som den kan exekvera dem. Detta kan dock lösas via användning av cacheminne och/eller pipelinen.

Cacheminnet

- Cacheminnet är en mycket snabb typ av RAM-minne som lagrar vanligt förekommande instruktioner och data, ibland hela instruktionsföljder, för minskad åtkomsttid och därmed effektivare programexekvering. Processorn och cacheminnet placeras vanligtvis på samma chip i syfte att minska åtkomsttiden för cacheminnets innehåll. Normalt sett kan innehåll hämtas från cacheminnet inom en enda klockcykel.

- Genom att lagra instruktioner och data som ofta används i det mycket snabba cacheminnet behöver dessa inte hämtas varje gång från ett långsammare minne, så som programminnet eller det externa RAM-minnet, vilket medför kraftigt minskad åtkomsttid.

- Behovet av cacheminne är störst när externa minnen används, då dessa medför längre åtkomsttid, vilket i förlängningen medför att processorn inte hinner läsa in instruktioner i den takt den kan exekvera dem. Cacheminnet möjliggör därmed att instruktioner och data kan hämtas och lagras i förväg och sedan direkt hämtas från cacheminnet när de ska exekveras. När interna minnen används är behovet av cacheminnet inte lika stort, då åtkomsttiden för interna minnen i många fall kan vara samma som för cacheminnet.

- Cacheminnet består av normalt av en instruktionscache samt en datacache. I instruktionscachen lagras de mest förekommande instruktionerna, vilket är cacheminnets primära syfte. Genom att lagra vanligt förekommande instruktioner eller instruktionsföljder, exempelvis instruktioner i en loop, behöver dessa inte hämtas varje gång från programminnet, vilket kraftigt minskar åtkomsttiden och effektiviserar därmed programmet. Ofta skrivs nämligen program så att specifika instruktionsföljder exekveras ett flertal gånger. I datacachen lagras mest förekommande data i programmet, exempelvis ett värde för en iterator i en loop eller adresser som skrivning och/eller läsning oftast sker till/från.

- Cacheminnets sekundära uppgift är att prediktera vilka instruktioner eller vilken data som kommer hämtas härnäst och läsa in dessa i instruktionscachen, så att efterföljande instruktioner kan hämtas från cacheminnet inom en klockcykel i stället för 5 – 10 ifall programminnet är lagrat i ett externt minne, såsom ett externt RAM-minne. Vid felaktig prediktion, exempelvis en hoppinstruktion, uppstår ett cachefel och instruktionen i fråga måste då hämtas från programminnet. I övriga fall medför dock denna prediktion av instruktioner att prestandan ökar kraftigt.

- Anledningen till att externt RAM-minne oftast används i kombination med cacheminne, trots att cacheminne är mycket snabbare och därmed bör föredras, är att cacheminnet normalt sett är ca 100 gånger dyrare per minnesmängd än externt RAM. För att hålla nere kostnaderna används därmed ofta en mindre mängd av det dyra cacheminnet samt en större mängd av det billigare externa RAM-minnet.

- Dessutom kan ökad mängd cacheminne medföra långsammare åtkomsttid, då det krävs längre tid att söka igenom cacheminnet ju större detta är. Detta kan dock delvis motverkas genom att dela in cacheminnet i olika segment L1, L2, L3 med mera, där det primära segmentet L1 innehåller de mest använda instruktionerna, det sekundära segmentet L2 innehåller de näst mest använda instruktionerna och så vidare.

Vad är det för skillnad när det gäller prestanda på internt och externt RAM-minne samt cacheminne?

- Internt RAM-minne medför ofta samma åtkomsttid som cacheminne, vilket medför att innehåll kan hämtas på en klockcykel. Externt RAM-minne har ofta 5 – 10 gånger längre åtkomsttid. Internt RAM är därmed snabbare, men är också dyrare, dock inte så dyrt som cacheminne. Externt RAM är mycket långsammare, men billigare.

Vilka kombinationer av internt och externt RAM-minne samt cacheminne brukar användas?

- Cacheminne är normalt sett lämpligt vid användning av extern RAM-minne för minskad åtkomsttid, men ökar också kostnaden. Därmed är det ofta föredraget antingen med internt RAM-minne eller externt RAM-minne i kombination med en liten mängd cacheminnet. Det senare alternativet är vanligtvis med praktiskt ur konstadssynpunkt när mycket minne behövs. Då kan en stor del av det billigare externa RAM-minnet kombineras med en mindre mängd av det dyra cacheminnet.

Pipelining

- Pipelining är en teknik för att öka processorns effektivitet genom att dela upp programexekveringen i flera steg, som kan genomföras parallellt. En typ av pipelining är processorns instruktionscykel, som vanligtvis delas in i 3 - 6 steg, varav de tre mest grundläggande presenteras nedan:

- **FETCH:** Nästa instruktion som ska utföras hämtas från programminnet (eller instruktionscachen om den redan har hämtats) och lagras i processorns instruktionsregister. Programräknaren, som pekar på adressen till instruktionen som just hämtades, inkrementeras för att peka på nästa instruktion som ska utföras (förutsatt att en hoppinstruktion inte utförs).

- **DECODE:** Aktuell instruktion avkodas (tolkas) och delas upp i en OP-kod (operationskod) samt eventuella operander. OP-koden indikerar vad som ska göras, exempelvis ADD för att addera tal, medan operander utgör ytterligare information, exempelvis destinationsadress, talen som ska adderas med mera.

- **EXECUTE:** Aktuell instruktion utförs utefter avkodad OP-kod samt eventuella operander. Instruktionscykeln börjar sedan om. Om instruktionen som utfördes inte utgör en hoppinstruktion så har nästa instruktion redan lästs in från programminnet till cacheminnet. Denna instruktion kan då avkodas direkt efter hämtning från cacheminnet och nästa FETCH-stadie (inläsning från programminnet) kan hoppas över. I stället kan nästa instruktion hämtas från programminnet till cacheminnet under avkodningen.

- När pipelining används kan ovanstående steg genomföras samtidigt. Medan nuvarande instruktion utförs avkodas predikterad nästa instruktion, medan predikterad instruktion efter det läses in från programminnet. Därmed kan en instruktion exekveras varje klockcykel i stället för var tredje som annars vore fallet. Så länge en hoppinstruktion inte genomförs medför detta ökad prestanda. Vid en hoppinstruktion måste hela pipelinen tömmas, vilket dock tar lite längre tid. Genomsnittstiden minskar dock kraftigt i regel, förutsatt att programmet i fråga består till stor del eller helt av hoppinstruktioner.

- Pipelining ökar samtidigt latensen, då varje instruktion måste passera fler steg innan den utförs, ofta 3 – 6 steg, där varje steg fördröjer instruktionen en klockcykel. Totalt sett förbättrar dock pipelining genomströmningen och prestandan, då en instruktion kan exekveras varje klockcykel (förutom vid hoppinstruktioner, då det tar lite längre tid).