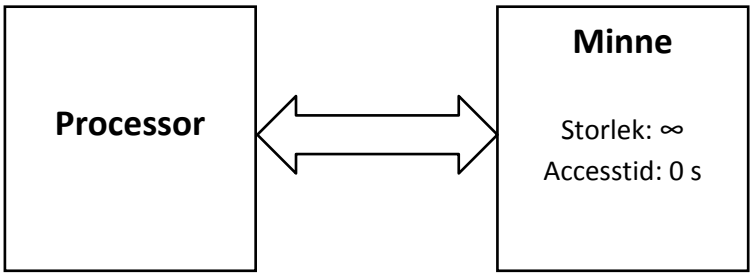
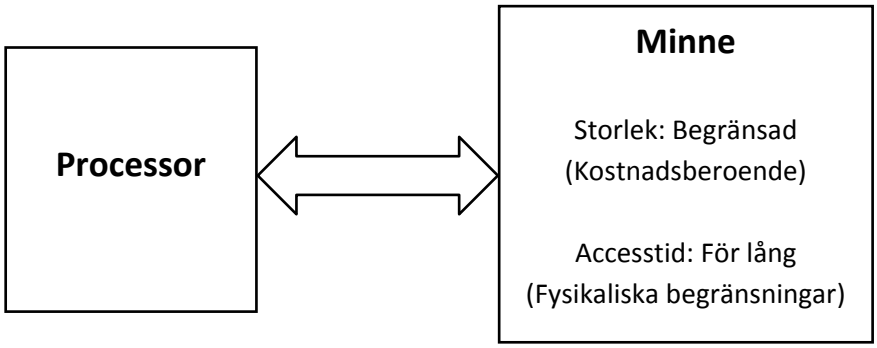
Minneshierarki

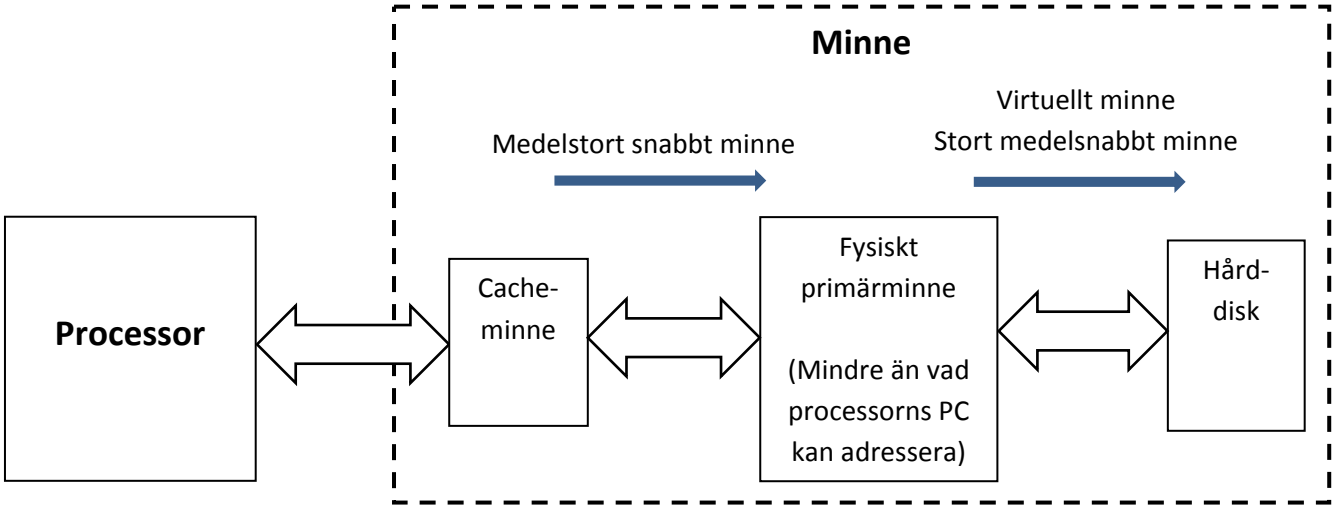
Hur ser en dator med ”idealiskt” minne ut?



Vilka begränsningar sätter verkligheten?



Hur angriper man begränsningarna?



Två tekniker som förbättrar primärminnets prestanda (snabbare och skenbart större till låg kostnad)

Bägge teknikerna bygger på en egenskap hos program som körs på en dator och för data som är lagrade i primärminnet. Egenskapen kallas ”**locality of reference**” och finns i två varianter, en för tid (”**temporal**”) och en för minnesadresser (”**spatial**”).

”**temporal**”: Under ett slumpvis valt kort tidsintervall används bara en mycket liten del av minnets adresser, dvs samma adressområden används om och om igen.

”**spatial**”: När man använder en minnesadress så är sannolikheten stor att man också kommer att använda närliggande adresser.

Cacheminne: (Snabbhet)

Litet snabbt minne som placeras mellan processor och primärminne. Cacheminnet skall innehålla de data som processorn behöver eftersom processorn kommunicerar med cacheminnet. Om processorn läser innehållet på en adress som inte finns i cacheminnet så överförs data från primärminnet till cacheminnet.

Virtuellt minne: (Skenbart stort primärminne till låg kostnad.)

Alla minnesadresser som kan nås från processorns PC (= det virtuella adressrummet) finns på en hårddisk som är långsam men billig per bit. En mindre del av adressrummet finns också i primärminnet. Primärminnet skall innehålla de data som processorn behöver eftersom processorn kommunicerar med primärminnet (via cache-minnet). Om processorn läser innehållet på en adress som inte finns i primärminnet så överförs data från hårddisken till primärminnet.

Den virtuella adressen översätts till en adress till det verkliga (fysiska) minnet. Denna översättning görs av en kombination av programvara och hårdvara med hjälp av adresstabeller i minnet. Det virtuella adressrummet delas in i block som kallas sidor (”pages”). På samma sätt delas det fysiska minnet in i block som kallas ”page frames” och rymmer en ”page”.

Olika principer för cacheminne

För enkelhets skull antar vi här att vi vill läsa i ett primärminne (main memory) som har storleken 1 kbyte (2^{10} byte) och är organiserat som 256 ord om 4 bytes (32 bitar) vardera. Cacheminnet antas ha 8 dataord om vardera 4 bytes, dvs 32 bytes. Man brukar kunna adressera varje byte direkt och vi väljer därför att använda de två lägsta adressbitarna A_1 och A_0 för att peka ut vilken byte som avses i ett 32-bitars dataord.

Main memory

Adress(10)	Data(32)			
98765 432 10	Byte 3	Byte 2	Byte 1	Byte 0
00000 000 00				
00000 001 00				
00000 010 00				
00000 011 00				
00000 100 00				
00000 101 00				
00000 110 00				
00000 111 00				
00001 000 00				
00001 001 00				
.	.			
.	.			
.	.			
11110 110 00				
11110 111 00				
11111 000 00				
11111 001 00				
11111 010 00				
11111 011 00				
11111 100 00				
11111 101 00				
11111 110 00				
11111 111 00				

Cacheminne med "direct mapping"

"Direct mapping" innebär att man använder en del av adressbitarna från adressen till "main memory" som adress till cacheminnet. Det innebär att minnesorden bara kan placeras i cacheminnet på adresser som överens-stämmer med motsvarande adressbitar från main memory. I vårt fall med 8 (2^3) dataord i cacheminnet blir det de tre adressbitarna A_4, A_3 och A_2 som avgör var dataorden från main memory placeras i cacheminnet, om de också utgör adressen till cacheminnet. I vårt exempel finns det 32 (2^5) "konkurrenter" om samma cacheminnesplats och risken att platsen är "upptagen" är därför stor. För att identifiera vilket av de 32 möjliga minnesorden som är placerat på en viss adress i cacheminnet måste man lagra de adressbitar från main memory som inte används, dvs A_9-A_5, tillsammans med databitarna. Denna adressdel kallas "tag". Arbetet med att ta reda på om ett visst dataord finns i cacheminnet är ganska enkelt vid "direct mapping".

Cache memory (direct mapping)

Adress (Index)	Data(37)				
432	Tag (bit 9-5)	Byte 3	Byte 2	Byte 1	Byte 0
000					
001					
010					
011					
100					
101					
110					
111					

Cacheminne med "fully associative mapping"

"Fully associative" cache innebär att man kan placera minnesord från "main memory" var som helst i cacheminnet. Det medför i vårt exempel att man måste lagra även adressbitarna A_4-A_2 i "tag" för att kunna identifiera hela adressen. Trots att antalet dataord i cache-minnet är lika stort i detta fall är sannolikheten att någon av de $2^3 = 8$ adresserna i vårt exempel innehåller det dataord man söker nu större än i fallet med "direct mapped cache" eftersom det finns 8 möjliga platser för detta.

Cache memory (fully associative)

Adress	Data(40)				
	Tag (bit 9-2)	Byte 3	Byte 2	Byte 1	Byte 0
000					
001					
010					
011					
100					
101					
110					
111					

Arbetet med att leta fram rätt dataord (med rätt tag) i cacheminnet är dock betydligt svårare i detta fall eftersom cacheadressen till "rätt" dataord inte är känd. Man skulle kunna läsa innehållet från cacheminnet ord för ord och undersöka tag-delen tills man får en "träff", men detta skulle ta för lång tid. Man kan istället använda ett "associativt minne" för att snabba upp sökningen. Ett sådant beskrivs i häftet "MEMORY SYSTEMS" i figur 13-5 på sidan 642. I det associativa minnet jämförs taggarna i alla dataord samtidigt med motsvarande adressbitar från processorn. Om den sökta taggen finns i något av dataorden så läggs datadelen av ordet ut på databussen så att processorn kan läsa det. Om taggen saknas så signaleras detta till main memory och läsningen görs där samtidigt som cacheminnet uppdateras. Nackdelen med det associativa minnet är att det är komplicerat, dyrt och tar stor plats på den tillgängliga chipsytan.

Cacheminne med ”set-associative mapping”

”Set-associative mapping” är en kompromiss mellan de två föregående metoderna. Man använder två eller flera uppsättningar (”set”) av dataord med ”direct mapping”. Varje adress i det ”direktmappade” cacheminnet kan därför rymma lika många dataord med olika adresser (dvs olika tags) som antalet ”set” och man undviker därmed att det dataord som finns i cacheminnet låser ”sin” adress för konkurrenterna. Principen illustreras i häftet ”MEMORY SYSTEMS” i figurerna 13-6 och 13-7 för ett cacheminne med 8 dataord uppdelade i två sets med 4 dataord vardera.

Virtuellt minne

Exempel

Antag att:

1 page = 4 kbyte = 1 k (= 2^{10}) 32-bitars ord.
Den virtuella adressen har 32 bitars ordlängd, vilket motsvarar 2^{30} st 32-bitars ord.

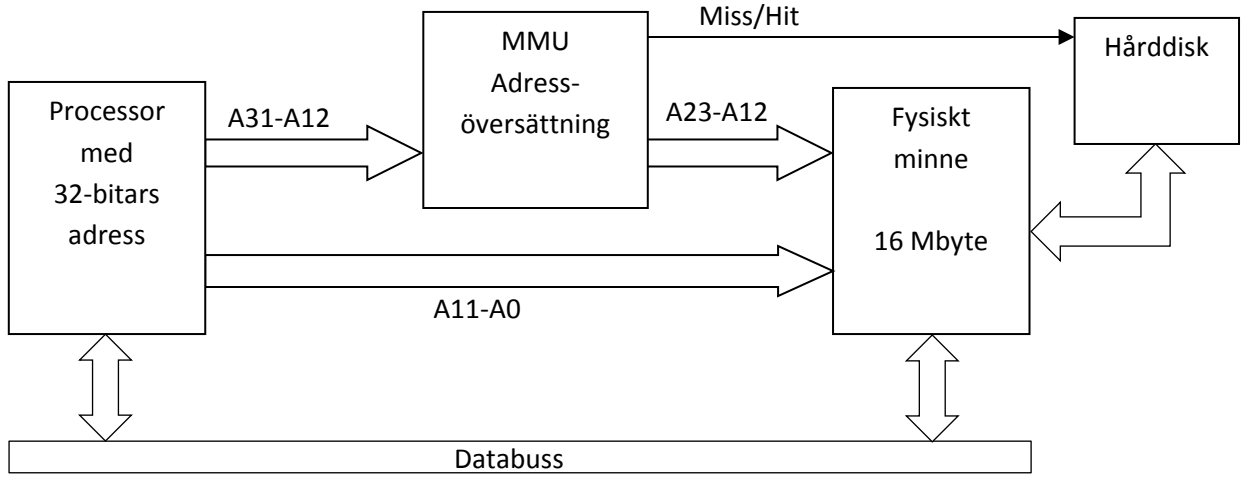
Totala antalet ”pages” i det virtuella adressrummet blir då $2^{30}/2^{10} = 2^{20}$ st.

Antag också att:

Fysiskt minne = 16 Mbyte, vilket motsvarar 2^{22} st 32-bitars ord.

Totala antalet ”page frames” i det fysiska minnet blir då $2^{22}/2^{10} = 2^{12}$ st.

Hur adressöversättningen från virtuell adress till fysisk adress går till i detta fall visas i figuren nedan.



Se också figur 13-11 i häftet ”MEMORY SYSTEMS”.