Saker att lära:

* Flyttals multiplikation och addition
* Cacheminne (Fö 16/11
* Dll
* Immediate
* Specialfallen
* Table

  Description automatically generated with medium confidence
* Hyperthreading
* Associativ mappning
* Direkt mappning

Videor att titta på:

Fö 20 (kolla igenom och sedan gå igenom motsvarande föreläsningar som innehåller främmande saker)

Fö 14. Bara för att förstå hur en gpu fungerar och för att lära sig om v-kommandon

Fö 12/15 flyttal är otroligt viktigt, det kommer alltid. Kolla videor på yt om hans förklaringar inte hjälper.

Fö 16. Lära sig hur man gör en chacheminne uppgift (det kommer alltid en sådan uppgift)

Fö 7. Se till att kunna pipelining samt harward/von neuman(Lär dig att rita de utifrån minnet och vad vardera del gör).

Fö 8. Innehåller kunskapsfrågor som kommer på provet. Se till att kunna pipelining

Fö 6. Viktigt att lära sig om avbrott.

Kolla tentaräkningarna sist.

Viktiga saker som jag plockade upp från föreläsningarna. INTE ALLT SOM ÄR VIKTIGT.

Länkning:

En relokerbar fil är en fil där inga minnesaddresser har tilldelats. Man har placeholders tills en länkare kopplar ihop filen med hjälp av sidoinfo och symboltabell som beskriver relationerna mellan vardera fil. Länkaren byter ut adresserna i den relokerbara filen till uppdaterade adresser

Dll, dynamisk länkning länkar saker enbart som behövs från objektsfiler. Detta minskar minnet som upptas då inga onödiga subrutiner hämtas. Filerna måste vara relokerbara. Tar tid första gången koden körs då allt måste länkas.

Fördelar:

ofta använda bibliotek till exember systembibliotek behöver bara lagras på en plats inte dupliceras i varje körbar fil vilket sparar minne och diskutrymme

Om ett fel i en bilioteksfunktion korrigeras genom att byta ut bilitoteket kommer alla program som använder det dynamiskt att dra nytta av korrigeringen efter omstart av dem. Program inkluderade denna funktion genom statisk läng måste först länkas om.

Nackdelar: vid uppdatering av en dll fil finns risk att den nya dll-filen inte blir bakåtkompatibel. Gasmla program som utnyttjar dll-filen kan då sluta att fungera ’dll hell’

I ett certifierat system slutar certifieringen att gälla om man uppdaterar dll-filer. Måste certifiera om allt.

Mindre portabelt om man vill flytta programmet till en annan miljö, måste flytta hela program-biblioteket.

Två metoder för dynamisk länkning

Raktfram: Laddar alla rutiner (när programmet startar) som kanske kommer att bli anropade (men inte resten av bilbioteket). Man vet dock inte vilka som behövs då programmet startar

Lazy linkage.

Laddar rutiner först då de anropas

Om man kompilerar kod för ett målsystem på en annan dator (utvecklingssystem) kallas förfarandet korskompilering

Deepbranch prediktion, kör koden på vinst och förlust.

* Minnes arkitekturer
* Tick tock toe
* Stack allignment
* Hyperthreading
* SIMD

Flyttals addition

Write through uppdaterar DRAM varje gång vi ändrar något i cache. Innebär att skrivningar tar lång tid

Kompromiss, ha en skrivbufferet där du håller dirty bits som tyder på vilken som skall skrivas över, sedan när ett block som ska slängas ut från cache så skriv tillbaka till minnet.