那些讓電腦變快的方法

暫存器

快取:(L1-L3:SARM)

主記憶體(DRAM)

磁碟儲存

Cmos組成記憶體，不需要邏輯閘的想法，電路更精簡

記憶體是用暫存器做的 由於記憶體只有24K+1字組，容量很小只能作為嵌入式的小控制器

DRAM裡面是用電容

目的->用DRAM讓電腦記憶體變大，成本不能增加太多

HackCPU問題:沒有乘除法，沒有浮點數的硬體電路

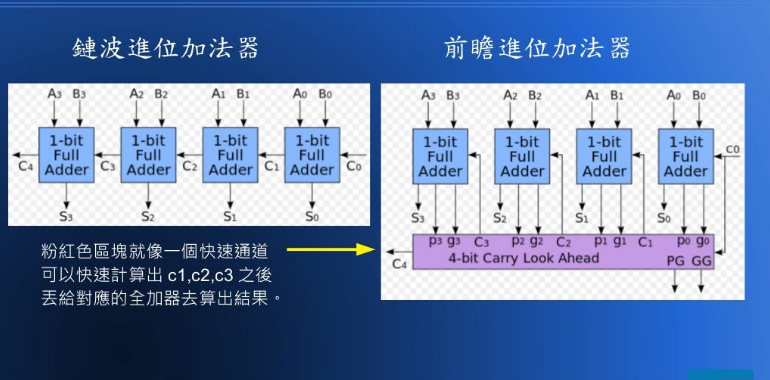
乘法器用加法器+移位運算

除法用減法器+移位運算

verilogHDL

HackCPU的ALU使用鏈波進位加法器，前一個算完後才開始下一個

只要改用前瞻進位加法器，速度會快數倍



常見加速方法

1. 快的材料取代慢的
2. 平行工作

1.

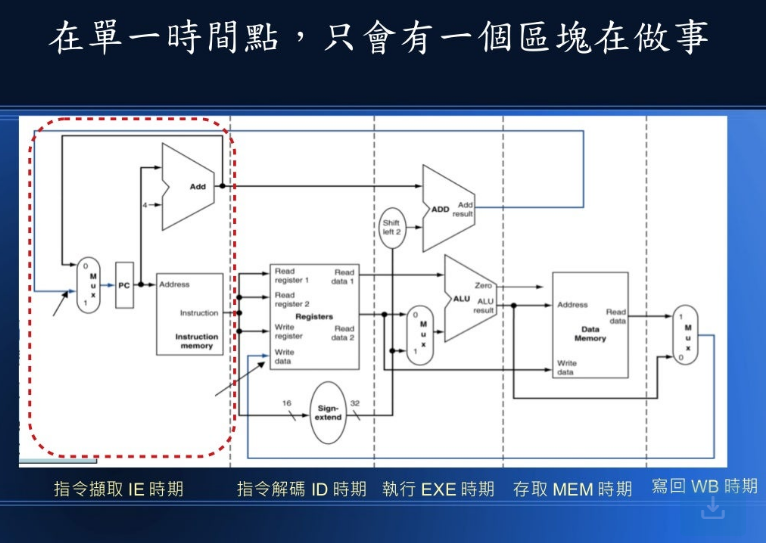
快取加速->加入適當容量快取(現代電腦來說特別重要)->增加快取命中率

區域性策略(一次載入一整塊)

記憶體速度>硬碟速度->盡量減少慢速裝置的存取

2.平行機制

管線pipeline機制:

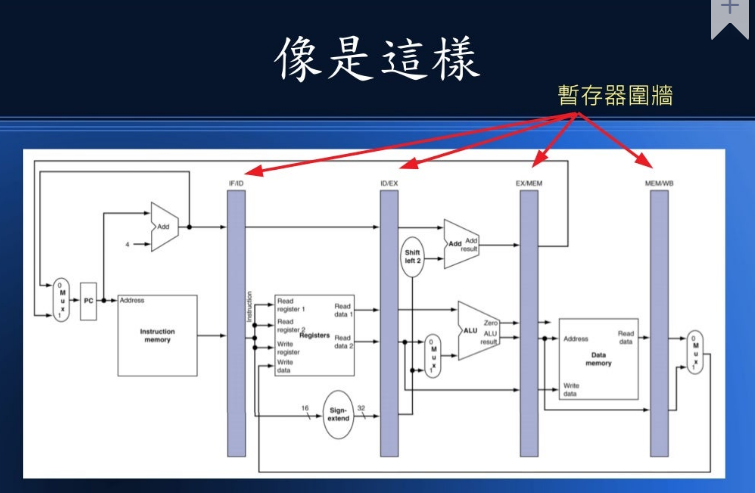
傳統

利用暫存器來讓CPU各區同時工作

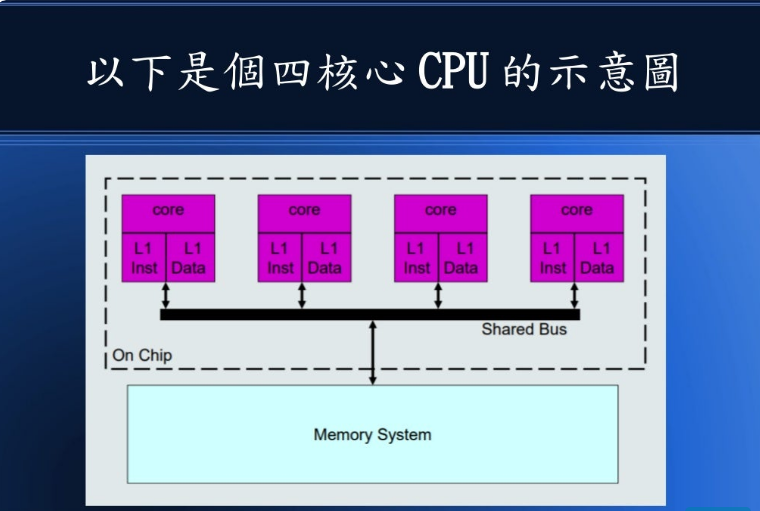
問題:

被跳躍指令打斷

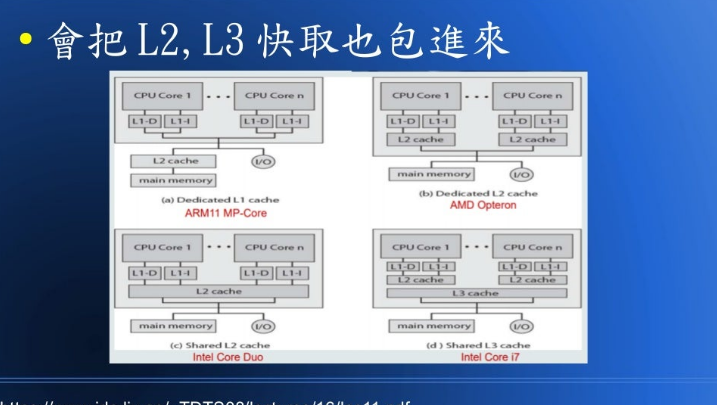
資料不在一級快取時，記憶體存取無法執行管線

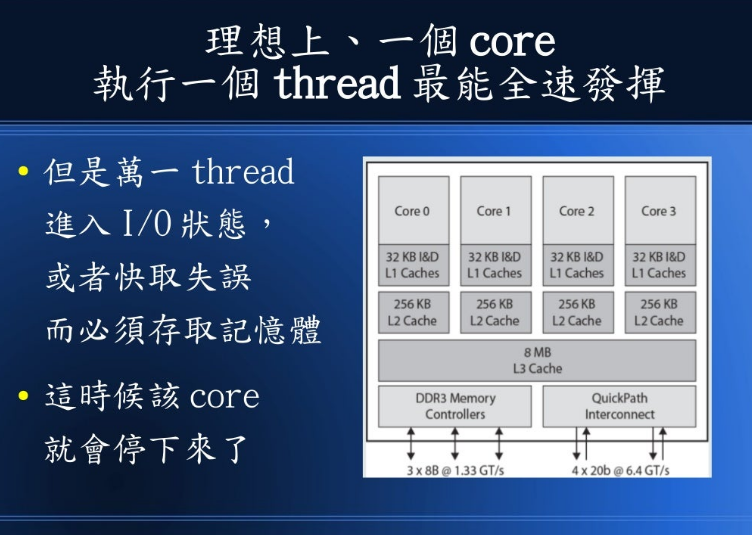


多核心+Hyper-Threading:

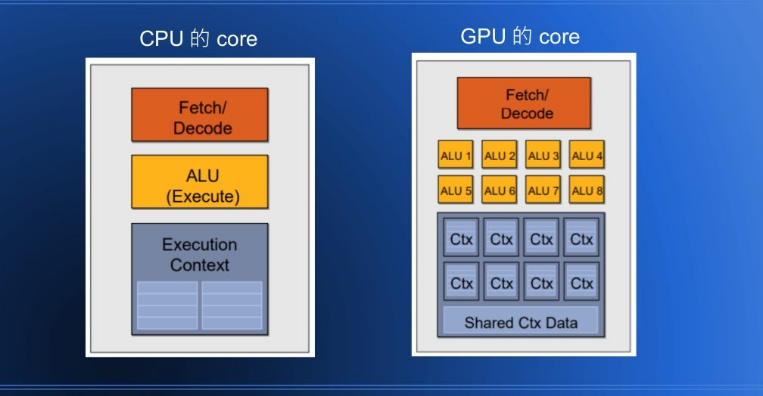


瓶頸:由於記憶體共用，同時間只有一個core能存取





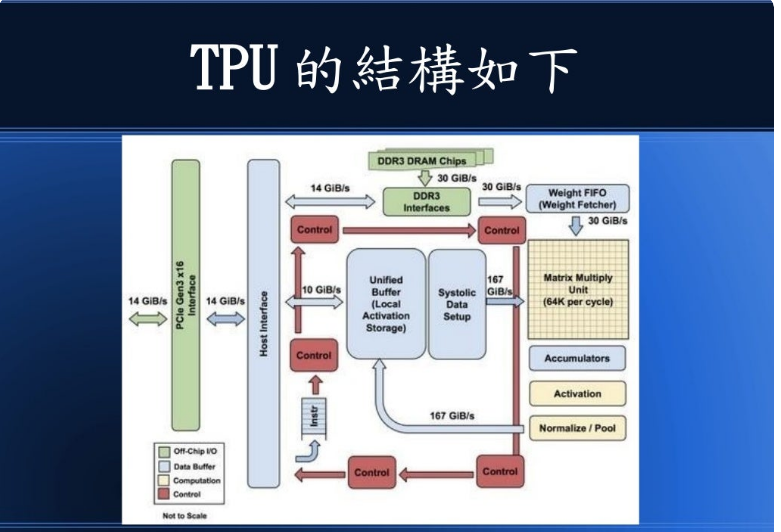
螢幕繪圖交給GPU



現在的電腦通常是CPU和GPU攜手合作

現代快速電腦舉例:

1. google深度學習TPU



TPU一次可以計算一整個矩陣相乘結果

1. 比特幣的挖掘機

使用SHA-256演算法快速計算

引用: https://www.slideshare.net/ccckmit/ss-85466673

# risc-v處理器

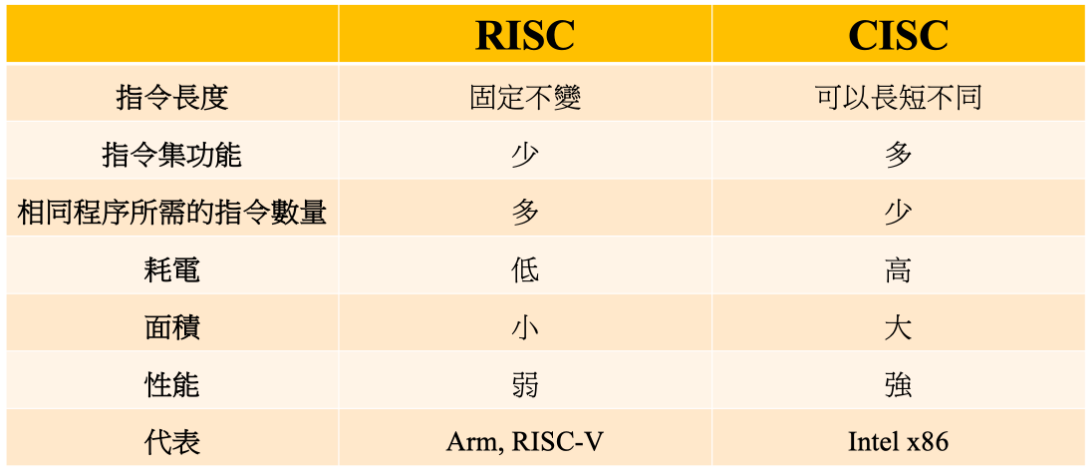
**前言:**

**RISC**（reduced instruction set computer, 精簡指令集電腦）為CPU的指令集的其中一種，另一種為**CISC**（Complex Instruction Set Computer, 複雜指令集電腦）。

組合語言由各式各樣的**指令構成**，這些指令各有不同的功能，為了要讓CPU看得懂，在C語言裡面一行就可以寫完的東西要變成很多步驟。

我們知道一個CPU的工作就是負責讀懂這些指令進行運算，因此給某個CPU的指令必須要是它看得懂的才行，而**一個CPU看得懂的所有指令**，我們就叫他**“某CPU的指令集”**，像是RISC-V的指令集、x86的指令集、MIPS的指令集等等。

**RISC指令集中的指令數量不多，功能也不多，所以需要較多指令才能相同的工作；CISC 則相反，他指令數量超多，功能超多，很多時候只要一行指令就可以完成需要的功能。**



**RISC的優點:**

1. 指令集精簡
2. 開源指令集
3. 彈性較大

RISC-V 指令使用模組化的設計, 包括幾個可以互相替換的基本指令集,以及額外可以選擇的擴充指令集。基本指令集規範了指令跟他們的編碼，控制流程，暫存器數目（以及它們的長度），記憶體跟定址方式，邏輯（整數）運算以及其他. 只要有軟體以及一個通用的編譯器的支援，只用基本指令集就可以用來製作一個簡單的通用型的電腦．

標準的擴充指令集可以搭配所有的基本指令集，以及其他擴充指令集，而不會衝突

* RV32I：基礎整數指令集(固定不變了)
* RV32M：乘法和除法
* RV32F：單精度浮點操作（和 RV32D:雙精度浮點操作）
* RV32A：原子操作
* RV32C：可選的壓縮擴展(對應32位元的RV32G)
* RV32B：基本擴展。
* RV32V：向量擴展（SIMD）指令
* RV64G：RISC-V的64位地址版本。

# RV32I:

RV32I是32-bit的基本整數指令集，該指令集會使用到32個暫存器(x0-x31)，其中x0被預留為常數0，其它31個暫存器(x1-x31)是普通的通用整數暫存器。在Risc-V組合語言中，每個通用暫存器都有一個對應的ABI名字，也就是說在組合語言中，x1等價於ra，它們都會編譯成相同的機器碼且一共有47道指令，RV32I 的指令一共可分成六大類，每種指令格式都是固定的32位指令，所以指令在記憶體中必須4位元組對齊，否則將觸發異常。:

I-Type:包含了暫存器與立即數的運算。

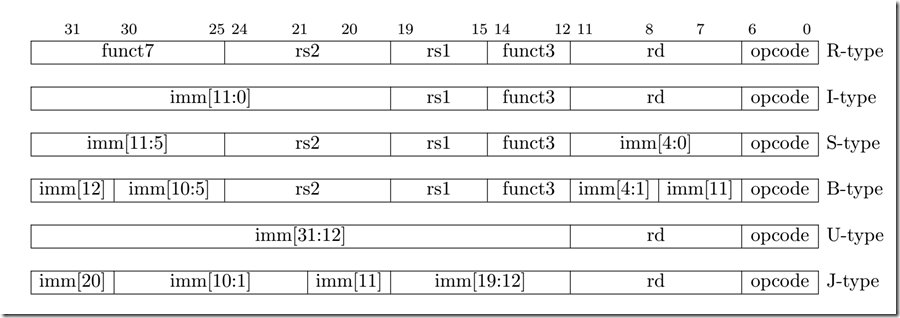
R-Type:此類型的指令會有3個暫存器作為Input，分別是:rd表示目的暫存器，rs1是源運算元暫存器1，rs2是源運算元暫存器2。

S-Type:包含存取記憶體的指令。

SB-Type:分支指令 (條件跳轉)。

U-Type:將立即數放到高位，這些指令被設計來實現完整的32bits運算。

UJ-Type:跳轉指令。



其中立即值的部分有時被分散成幾區，像是 S-Type 分為 imm[4:0]+imm[11:5]，而 B,J 等類型的指令，imm 被分為更多部分，這主要是為了相容與擴充性的考量，在用 Verilog 或 VHDL 設計電路時可以用 decoder 先將立即值組合成單一欄位，然後才開始執行該指令

參考資料:

<https://weikaiwei.com/riscv/riscv-1/>

<https://zh.wikipedia.org/wiki/RISC-V>

<https://programmermedia.org/root/%E9%99%B3%E9%8D%BE%E8%AA%A0/%E8%AA%B2%E7%A8%8B/%E7%B3%BB%E7%B5%B1%E7%A8%8B%E5%BC%8F/10-riscv/01-riscv/RISC-V%E8%99%95%E7%90%86%E5%99%A8.md>

<https://ithelp.ithome.com.tw/articles/10268196>

<https://www.itread01.com/lflie.html>