Computer Organization And Assembly Language

Final Project:

ARM & MIPS 之比較

資工二 B95902087 鄭勝學

動機

在上 ARM 架構的時候，才真正了解到什麼是處理器。長久以來一直以爲所謂的架構是一種實質上的東西，而經過上課後才知道原來更注重的是一種觀念，一種計算機架構處理的原理。也在上完此課後讓我想了解除了泛用的 ARM 以外，還有哪些熱門的處理器系統。

由於是以 ARM 開始認識，因此想了解有哪些是跟 ARM 可說是有類似的功能，而以不同的架構處理。

*底下擷取自 wikipedia:

ARM 架構

（過去稱作**進階精簡指令集機器（Advanced RISC Machine）**，更早稱作 **Acorn RISC Machine**）是一個 32 位元精簡指令集（RISC） 中央處理器（processor）架構，其廣泛地使用在許多嵌入式系統（embedded）設計。由於節能的特點，ARM 處理器非常適用於行動通訊領域，符合其主要設計目標爲低耗電的特性。

嵌入式系統

一種完全嵌入受控器件內部爲特定應用設計的專用電腦系統。嵌入式系統通常執行的是帶有特定要求的預先定義的任務。由於嵌入式系統只針對一項特殊的任務，設計人員能夠對它進行優化，減小尺寸降低成本。

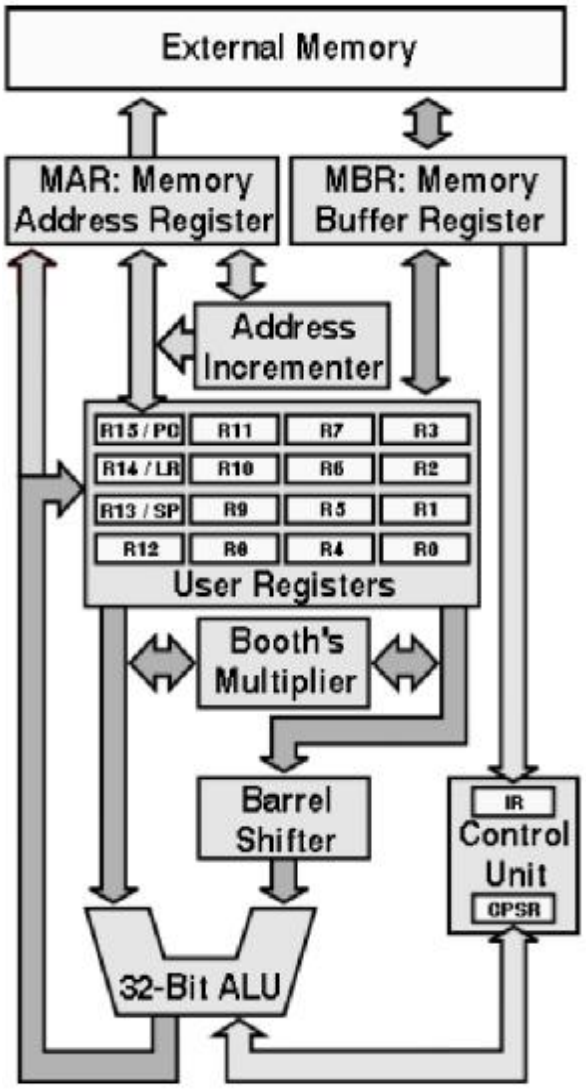
因此，這次的 Project 我以嵌入式系統的微處理器做爲主題，選取了 ARM 以及 MIPS 兩大處理器做比較。

市場主流

目前常見的嵌入式微處理器 (Embedded Micro- processor Unit, 簡稱 EMPU)有 Am186/88、386EX、SC-400、Power PC、Motorola 68000、MIPS、ARM、ARC 系列等。

而這次我們介紹的是市場主流的三大公司 ARM、MIPS 推行的嵌入式架構的處原理，並在介紹完稍候做比較。

ARM



ARM 結構在課堂上已有詳細的描述，因此就不多提。

ARM 主要是以指令集來區分，而 V1~V3 奠定了基礎。

V1 架構是最原始的 ARM 指令集，僅具備有基本的資料處理指令，不包含乘法指令。

而在 V2 版中，則是加入了乘法以及乘加指令，並且加入了輔助處理器的操作指令，增加了快速中斷模式，以及對暫存記憶體的管理規則。

到 V3 架構中，則是將記憶體的定址空間大幅增加到 32 位元，也就是可以定址到 4GB 的記憶體容量，而增加了 CPSR 以及 SPSR 暫存器，可以保存程式狀態，並且加入了 MRS/MSR 兩個指令，藉以存取這兩個暫存器。

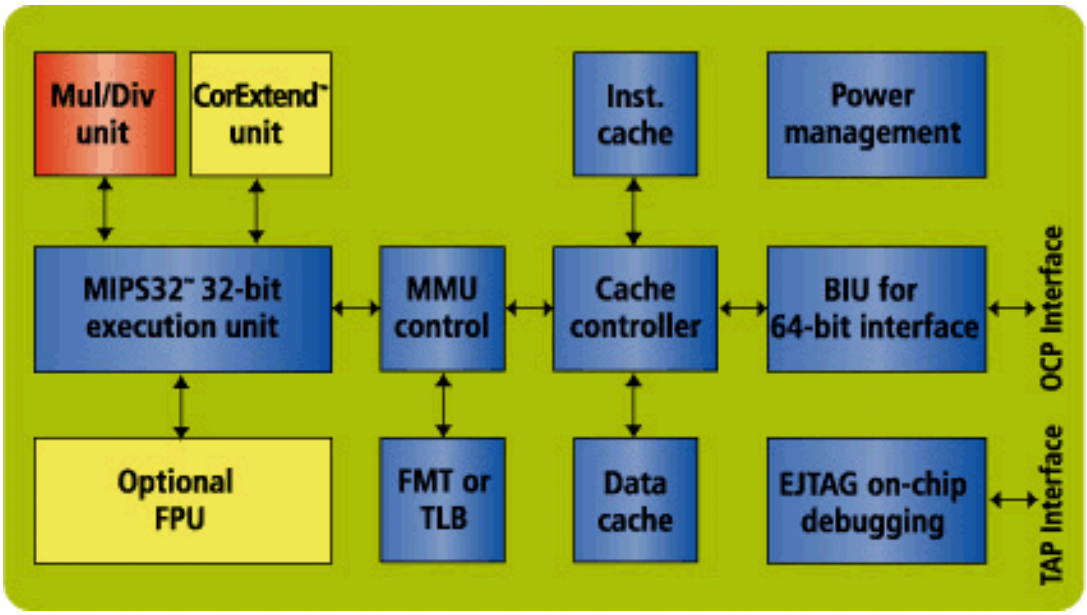
之後延續的系列皆以這些基礎來做修改。

雖然該公司本身並無晶圓廠，而純粹以 IP 的形式出售處理器架構，由於定位正確，在短短的數年間取得了極大的市場地位，全世界絕大多數的手持式裝置都嵌入了 ARM 的處理器技術。

MIPS(Microprocessor without interlocked pipeline stages，亦即不含 Interlock 機制的管線階層微處理器。)

MIPS 也是一家具有悠久歷史的處理器研發商，同樣的，MIPS 架構處理器也出現在許多日常生活中可見到的產品中，在遊樂器方面，過去的任天堂 64、SONY Playstation 1、Playstation 2，以及新近的 PSP 等產品等是採用 MIPS 架構，而在一般手持式 Windows CE 產品中，也有採用 MIPS 架構，在網通產品方面，MIPS 處理器也被廣泛的應用在 CISCO 的路由器中。

以 MIPS 的架構來說，其發展的歷史比起 ARM 要來得悠久，設計上也有不少過人之處，比如說從 32 位元處理到 64 位元運算的架構延展性，讓 MIPS 處理器可適用於於各種用途。



MIPS 處理器是個以管線方式工作的處器，因此執行程式碼的速度，就相當依賴管線的工作方式。絕大多數 MIPS 指令需要在管線 RD 階段取得足夠的 operands，並且在緊接著 ALU 階段之後產生結果。由於 MIPS 架構中，大多數的指令皆能遵照這樣的運作方式進行處理，因此在指令處理效率上幾乎都能夠達到理論上的最大值。

不過少部份狀況之下，比如說，如果下一條指令必須要靠前一條指令的執行結果來進行運算處理的話，如果在前一條指令處理完之前，另一條指令就搶著進入管線，那麼將會遭遇到不可預料的錯誤。

而由於架構上是以簡單的 RISC 爲基礎，因此 MIPS 有較爲明顯的兩個缺陷

分支延遲：(branch delay slot)

在所有的 MIPS 處理器中，跟在分支指令之後的指令，即使在與前一個分支指令流向分歧之後，依然會被處理器所執行，因此在之後的 MIPS Ⅱ體系中，加入了 Branch-likely 指令，在處理類似的狀況時，在分支指令其後的指令只有在前一個分支被接受時，才會被執行，不過除非自行指定分之後的指令，在加強後的編譯器的處理下，分支所帶來的延遲將顯得不明顯。

載入延遲：(load delay slot)

在 MIPS Ⅰ 指令集中，load 指令將無法再次載入才剛被 load 指令本身所載入的資料，若是有需要再度載入，那麼必須在兩個 load 流程之間，使用其他指令來區隔，甚至是使用空指令來空轉一週，以便讓 load 指令可再度進行載入。

浮點運算單元的問題：

由於浮點運算需要耗費多個處理器時脈週期來進行，因此在 MIPS 處理器架構中，大多會有獨立的浮點運算處理管線，構成內部的輔助處理器架構，由於浮點運算單元可以與其後的指令並行處理，因此當並行處理的指令要去存取尚未計算完成的浮點運算結果暫存器時，處理器便會停止執行，因此這部份的處理也需要大量的編譯器最佳化。

架構比較

u Pipeline

- MIPS

最簡單的體系結構之一，體積小、耗能比低。但 MIPS 有 "branch delay slot" 以及 "load delay slot"兩個明顯的缺點，MIPS 使用編譯器来解決上面的兩個問題。因为 MIPS 最初的設計思想是使用簡單的 RISC 及其他軟體技術，来達成 RISC 的完整概念。

- ARM's Shifter

shifter 是 ARM 中很重要的概念，他可以提高運算邏輯的速度，跟同樣功能的 adder/shift register 相比，效率更高，但是也 占用更多的芯片面積。

u 指令結構

MIPS 有 32bit 及 64bit architecture，但是 ARM 只有 32bit architecture (ARM11 局部 64 位)

MIPS 是開放式的架構，可在開發的內核中建立自己的指令，而 ARM 是在每個指令結構以 4bit 的 condition code 決定。

u 寄存器

由於 MIPS 內核中有 32 個 register,相對於 ARM 只有 16 個，設計天生上的優勢使得同等性能下，MIPS 的芯片面積及攻耗比較小。ARM 有一组特殊用途寄存器 cp0-cp15,可以使用 MCR,MRC 等指令控制;相對應的，MIPS 也有 cp0 0-30,使用 mfc0,mtc0 指令控制。

u 地址空間 address space

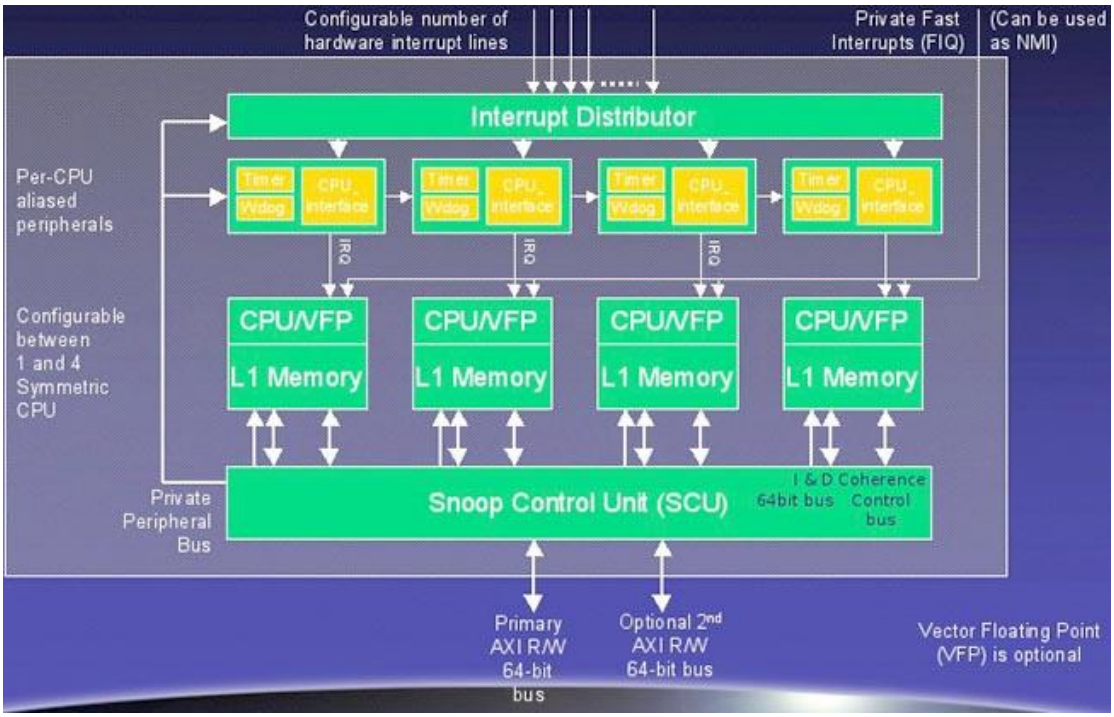
MIPS 起始地址是 0xbfc00000,會有 4Mbyte 的大小限制，但一般 MIPS 芯片都會採取一些方法解決這個問題。而 ARM 没有這種問題。在新版的 MIPS24K 起始地址往後移，有了 16byte 的大小。

發展

爲了提升效能以及改善未來適應性，兩家公司近年來在提升以及研發都有了明確的方向。

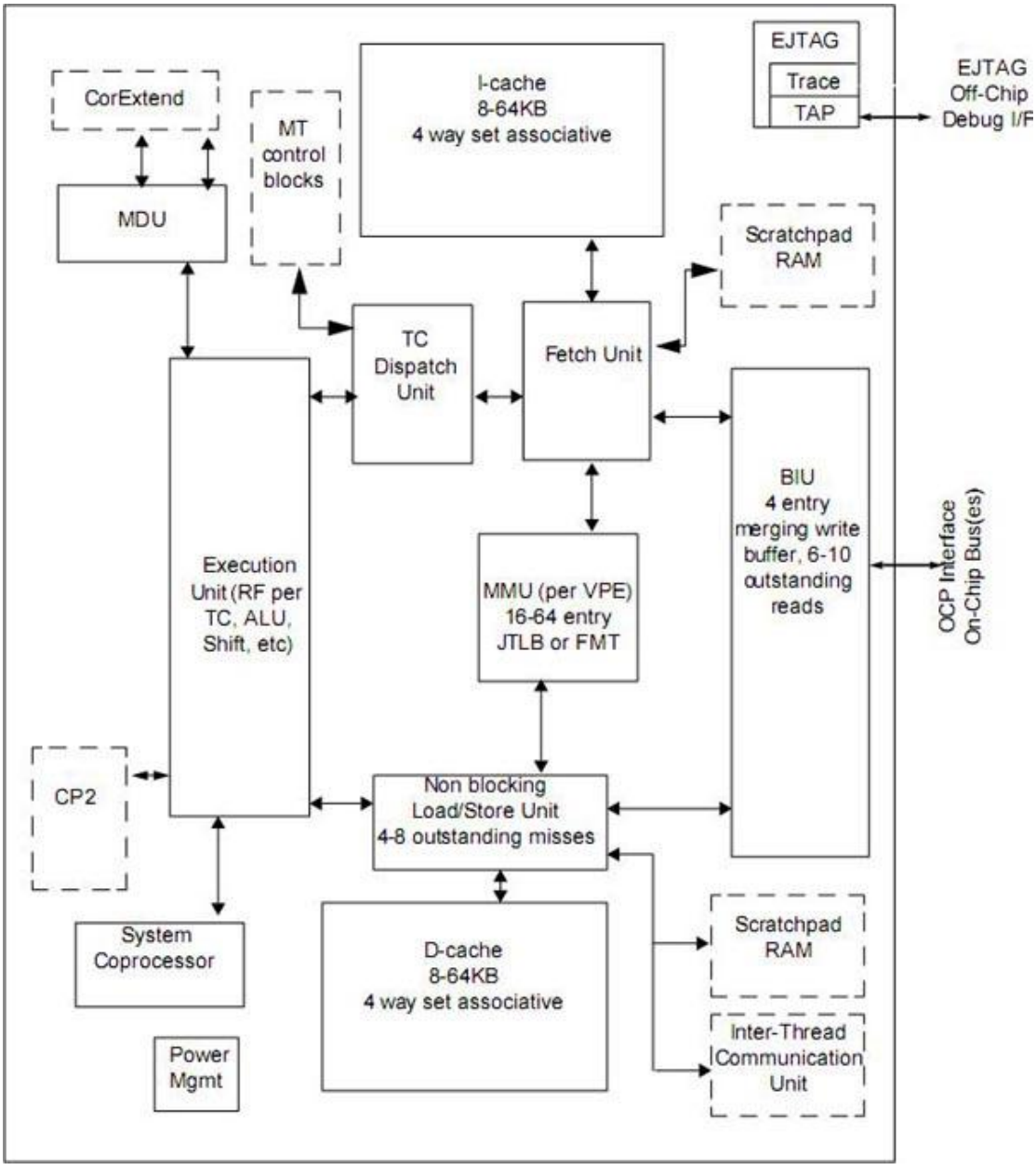
ARM -多核心架構 Multi Processor

以 ARM 處理器-ARM11 MPCORE，根據不同應用的需要，MPCore 可以被配置爲 1~4 個處理器的組合方式，根據官方表示，其最高性能約可達到 2600 Dhrystone MIPS 的程度。MPCore 是標準的同質多核心處理器，，組成 MPCore 的是 4 個基於 ARM11 架構的處理器核心，由於多核心設計的優點是在頻率不變的情況下讓處理器的性能獲得明顯提升，因此可望在多任務應用中擁有良好的表現。



MIPS – 多執行緒 Multi Thread

單一處理器核心在運算的過程中，常會有記憶體存取速度跟不上處理器時脈增加的問題，進而導致快取記憶體錯失（miss）時，形成執行管線長時間閒置的狀。如果利用多執行緒處理概念，適時的將其他執行緒拉過來填補已經造成的閒置狀態，其速度的增長甚至可以達到非常明顯的地步。



多執行緒技術著重於處理單元、記憶體控制器的有效利用，在最大程度上節省電晶體的使用，並且在此前提之下往上提升效能表現，這與多核心架構中，系統效能需求有多少，就複製多少個核心塞進晶片中的浪費作法完全不同，多核心可以取得較爲全面的應用廣度，但是稍嫌鋪張浪費，而多執行緒在成本與效能方

面的平衡性表現要來得高明些。但是多執行緒技術有個嚴重的缺陷，那就是多執行緒工作處理過程中，過於頻繁的上下文切換（context switch）將有可能會造成極大的效能耗損。

心得

其實在最早著手於期末報告時，是想以多執行緒以及多核心差別爲主的介紹方式。但是由於多執行緒的技術非常複雜，以我的程度還沒辦法好好了解詳述，使得這次的報告內容比我預計的內容還爲少。

但是這次的報告，也讓我了解不只是 ARM，還有其它千千種種的處理架構系統，也算是一大收穫。而學習本學期的課程的原理，讓我了解到資工系不只是要只會學習寫 CODE，而是如何有自己的 IDEA，以及在處理結構能好好掌握才能寫出一個好程式來。

Reference

Wikipedia
http://zh.wikipedia.org/w/index.php?title=%E9%A6%96%E9%A1%B5&variant=zh-tw

DIGITIMES 科技網
http://www.digitimes.com.tw/

台大資工系 計算機組織與組合語言 課程網
http://www.csie.ntu.edu.tw/~cyy/courses/assembly/07fall/