Computer Organization and Assembly Languages, Fall 2007

Final Project

Topic:

Implementing a many-in-one Game Boy Advance ROM Based on HAM Libraries

利用 **Ham libraries** 實做多合一 **GBA** 遊戲卡匣

Instructor:

Prof. Yung-Yu Chuang

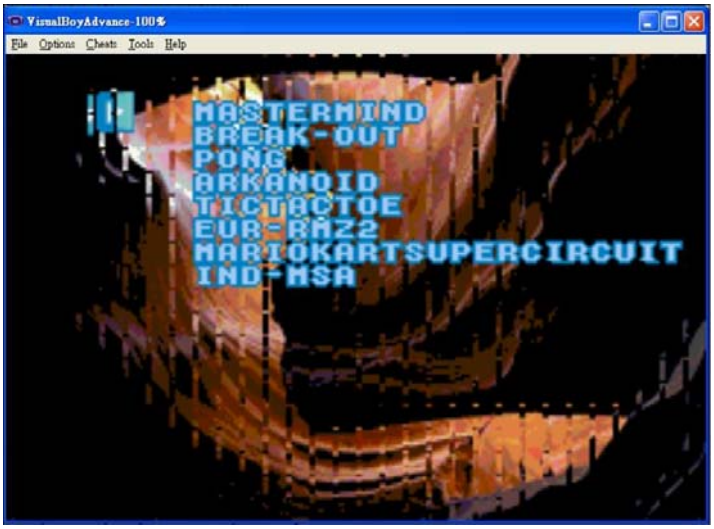
Students:

B95902109 吳冠龍

B95902077 王柏易

B95902069 徐廷倫

B95902076 游書豪



Motivations:

在這學期的計算機組織與組合語言課程中，教授講解了 ARM Architecture，並讓大家試著在 Game Boy Advance 上利用 ARM assembly 實做 Box Filter。GBA 的風行使得在 GBA 平台上的遊戲如雨後春筍般冒出，但每次想玩一個遊戲就得重新換 ROM 讓 GBA 去執行著實有些麻煩，所以我們選擇試著在 GBA 的平台上實作出一個方便選擇遊戲並執行的介面，使得 ROM 裡面的遊戲可以增加，達成多合一遊戲卡帶的結果。

Surveys:

1. ARM Architecture:

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為低耗電的特性。

在今日，ARM 家族佔了所有 32 位元嵌入式處理器 75%的比例，使它成為占全世界最多數的 32 位元架構之一。ARM 處理器可以在很多消費性電子產品上看到，從可攜式裝置（PDA、行動電話、多媒體播放器、掌上型電玩，和計算機）到電腦週邊設備（硬碟、桌上型路由器）都有。在此家族中衍伸的重要產品還包括 Marvell 的 XScale 架構和德州儀器的 OMAP 系列。ARM 架構包含了下述 RISC 特性：

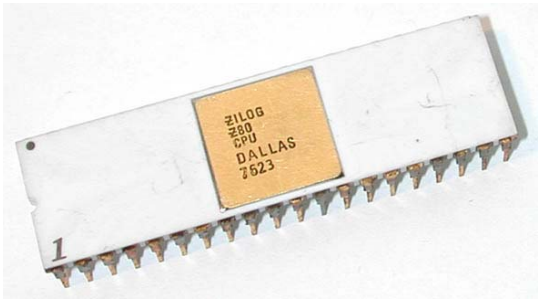


Figure 1. Z80 microprocessor

i. 讀取 / 儲存 架構
ii. 不支援地址不對齊記憶體存取(ARMv6 內核現已支援)
iii. 正交指令集(任意存取指令可以任意的尋址方式存數據

Orthogonal instruction set)

iv. 大量的 16 × 32-bit 暫存器檔案(寄存器陣列 register file)
v. 固定的 32 bits 操作碼寬(opcode)，降低編碼數量所產生的耗費，減輕解碼和管線化的負擔。
vi. 大多均為一個 CPU 周期執行。

為了補強這種簡單的設計方式，相較於同時期的處理器如 Intel 80286 和 Motorola 68020，還多加了一些特殊設計：

i. 大部分指令可以條件式地執行，降低在分支時產生的負重，彌補分支預測器(branch predictor)的不足。
ii. 算數指令只會在要求時更改條件編碼(condition code)
iii. 32-bit 筒型位移器(barrel shifter)可用來執行大部分的算數指令和定址計算而不會損失效能
iv. 強大的索引定址模式(addressing mode)
v. 精簡但快速的雙優先級中斷子系統，具有可切換的暫存器組

有個附加在 ARM 設計中好玩的東西，就是使用一個 4-bit 條件編碼在每個指令前頭，表示每支指令的執行是否為有條件式的，這大大的減低了在記憶體存取指令時用到的編碼位元，換句話說，它避免在對小型敘述如 if 做分支指令。

ARM 公司本身並不靠自有的設計來製造或販售 CPU 裝置，反而是將處理器架構授權給有興趣的廠家。ARM 提供了多樣的授權條款，包括售價與散播性等項目。對於授權方來說，ARM 提供了 ARM 內核的整合硬體敘述，包含完整的軟體開發工具（編譯器、debugger、SDK），以及針對內含 ARM CPU 矽晶片的銷售權。對於無晶圓廠的授權方來說，其希望能將 ARM 內核整合到他們自行研發的晶片設計中，，通常就僅針對取得一份生產就緒的智財核心技術（IP Core）認證。對這些客戶來說，ARM 會釋出所選的 ARM 核心的閘極電路圖，連同抽象模擬模型和測試程式，以協助設計整合和驗證。需求更多的客戶，包括整合元件製造商（IDM）和晶圓廠家，就

選擇可合成的 RTL（暫存器轉移層級，如 Verilog）形式來取得處理器的智財權(IP)。藉著可整合的 RTL，客戶就有能力能進行架構上的最佳化與加強。這個方式能讓設計者完成額外的設計目標(如高震盪頻率、低能量耗損、指令集延伸等)而不會受限於無法更動的電路圖。雖然 ARM 並不授予授權方再次販售 ARM 架構本身，但授權方可以任意地販售製品(如晶片元件、評估板、完整系統等)。商用晶圓廠是特殊例子，因為他們不僅授予能販售包含 ARM 內核的矽晶成品，對其它客戶來講，他們通常也保留重製 ARM 內核的權利。

就像大多數 IP 販售方, ARM 依照使用價值來決定 IP 的售價。在架構上而言，較低效能的 ARM 內核比較高效能的內核擁有較低的授權費。以矽晶片實作而言，一顆可整合的內核要比一顆硬體巨集(黑箱)內核要來得貴。更複雜的價位問題來講，持有 ARM 授權的商用晶圓廠（例如韓國三星和日本富士通）可以提供更低的授權價格給他們的晶圓廠客戶。透過晶圓廠自有的設計技術，客戶可以更低或是免費的 ARM 預付授權費來取得 ARM 內核。相較於不具備自有設計技術的專門半導體晶圓廠(如台積電和聯電)，富士通 / 三星對每片晶圓多收取了兩至三倍的費用。對中少量的應用而言，具備設計部門的晶圓廠提供較低的整體價格(透過授權費用的補助)。對於量產而言，由於長期的成本縮減可藉由更低的晶圓價格，減少 ARM 的 NRE 成本，使得專門的晶圓廠也成了一個更好的選擇。[1]

2. Game Boy Advance

Game Boy Advance 是日本任天堂公司於 2001 年 3 月 21 日發售的攜帶型遊戲機，可兼容大部分以前的 Game Boy 和 Game Boy Color 的遊戲。一般簡稱為 GBA。 任天堂公司接著於 2003 年 2 月 14 日推出了具備鋰電池、可摺疊且具有螢幕前光的 Game Boy Advance SP(一般簡稱為 GBA SP)。

後來又在 2005 年 9 月 13 日的超級瑪利歐 20 周年紀念時，首度推出採用可 5 級亮度校調的高背光液晶屏幕，不可摺疊，但是機型更為輕巧的 GAME BOY MICRO(一般簡稱為 GBM)，GBM 為 Game Boy

[1] http://zh.wikipedia.org/wiki/ARM%E6%9E%B6%E6%A7%8B

Advance 系列中唯一不向下相容以前 Game Boy 和 Game Boy Color 遊戲。只支援 GBA 遊戲，另外連線配件也與 BOY ADVANCE 系列不相容。

同期再推出高背光亮度的 Game Boy Advance SP 版本，採用 GAME BOY MICRO 的新液晶屏幕，有 2 級亮度調整。

GBA 的特性如下:

i. 擁有一顆 RISC 架構的 32 位元 CPU ARM7TDMI，和一顆 CISC 架構的 8 位元 CPU。其中 8 位元的 CISC CPU 是用來相容 Game Boy 和 Game Boy Color 的遊戲的。
ii. 畫面解析度為 240x160
iii. 最大可以顯示 32768（2^15）種顏色
iv. 使用被動反射型液晶(TFT)螢幕。
v. 在 Game Boy 的十字方向鍵和 A、B 鍵之外又增加了 L、R 兩個鍵。
vi. 使用兩顆三號乾電池(AAx2)大約能使用 15 個小時。[2]

3. Game Boy ROM development kits

在開發 GBA 遊戲時，有三項主要的 development kit 可以選擇，此三項都是使用 GNU gcc compiler 核心，支援 C/C++程式開發:

A. devkit advanced[3]: 最早的 GBA development kit，但現今已無人維護開發。
B. devkitPro/devkitARM[4]: 從 devkit advanced 發展而來，有相當多的免費的開發工具(例如:Tonc)。支援不同平台上的開發，是一個 opensource project。
C. HAM Gameboy Advance Development Kit[5]: 一套簡易強大的 GBA 開發工具，內部提供的 HAM library 可以讓使用者簡

[2] http://zh.wikipedia.org/wiki/Game_Boy_Advance
[3] http://devkitadv.sourceforge.net/
[4] http://sourceforge.net/projects/devkitpro/
[5] http://www.ngine.de/

易的在 GBA 上實作遊戲效果。由 ngine software 開發維護。HAM 無法與 devkitPro 相容。

三者各有優缺點，HAM 中的 Integrated Development Environment : Visual HAM 提供了強大的整合式開發環境，開發門檻較低，但是非 opensource 的關係使得相容性較低，devkitPro/devkitARM 因為是 opensource project，所以有較多的 tools 可以使用。

Procedures:

1. 建立開發環境：

我們使用 Visual HAM 這套整合式開發環境與 HAM libraries 來實作多合一的卡帶。利用 Visual HAM 開發 GBA ROM 需要 C 與 ARM Assembly 的 programming skill 與 makefile 的撰寫能力，因為 Visual HAM 有時候產生的 makefile 並不能正確的編譯出執行檔。

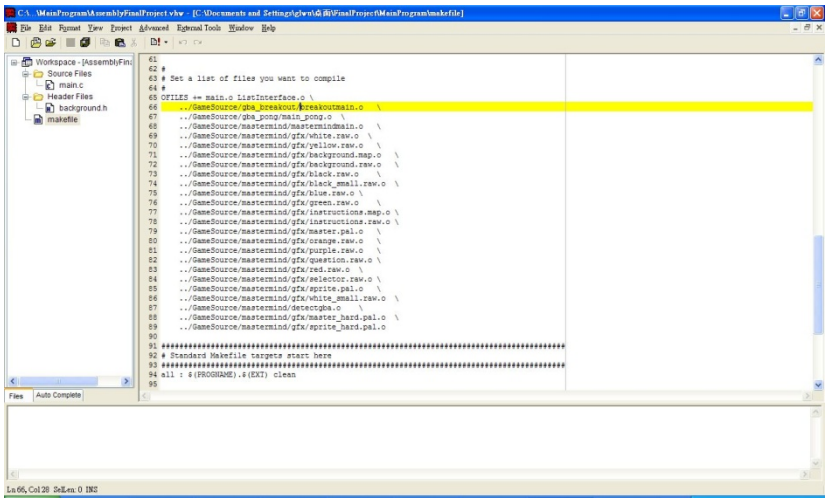


Figure 2. Visual HAM 開發環境

2. 程式架構設計:

在我們的程式的規劃架構中，我們需要撰寫出一個 ListInterface 的程式碼來 render 出遊戲的選擇介面，並且以鍵盤的上下鍵控制遊戲選項的控制介面。再利用 ListInterface 回傳的 gameID 值讓主程式去 link 到遊戲的 source 並開始執行遊戲。

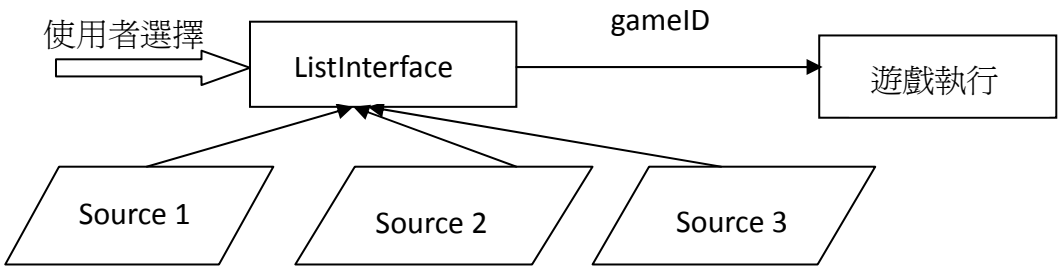


Figure 3. 程式架構

3. 尋找 GBA Game Source:

因為我們所開發的程式架構需要遊戲原本的 source 才能順利執行，並不像是坊間利用另外硬體介面的控制來執行遊戲，所以我們必須要找多個遊戲的 source 以便開發時使用。最後我們找到了一些熱心的 GBA 遊戲開發者在網路上分享的 source 供我們做測試。

4. 介面圖形開發:

我們的圖形介面基本上包括選單、背景、按鈕等幾個部分。選單列出遊戲列表，而使用者可移動按鈕指向欲選擇的遊戲，並按下選擇鍵(A)進入遊戲。

Figure 4. 遊戲選單示意圖

5. ListInterface 實作:

我們利用 C 語言撰寫出程式碼後再依照 C 的程式碼翻譯成 ARM assembly 供 assembler 將所有來源檔編譯成一個 GBA ROM。

我們使用到一些 HAM 的 function 來製作我們的 ListInterface，在我們的圖形介面中，使用背景模式(BGMode)是 1，背景部分就是 background，而按鈕則是使用 sprite 製作。

6. GBA ROM 外部連結

在我們原先的實做方式中，會存在一個記憶體管理的問題，因為將許多遊戲的 source link 在一起時，有可能會產生 GBA ROM 無法 load 足夠多的資料到 GBA 的 main memory 中。所以我們另外撰寫了一個程式加入到介面中，使得使用者可以操作介面讓 GBA 去存取硬碟所放置的 ROM(*.gba)，如此即可避免掉所有的遊戲 source 可能無法全部 load 到 GBA 的問題，但其缺點為 GBA 無法直接存取，必須利用其他的介面控制 ROM 的存取。此實作方式類似於坊間所開發的 GBA ROM 控制卡，利用硬體實作讓 GBA 可以個別去存取遊戲的 source。

Result

我們實際做出一個 interface 可供使用者選擇想要玩的遊戲並執行。

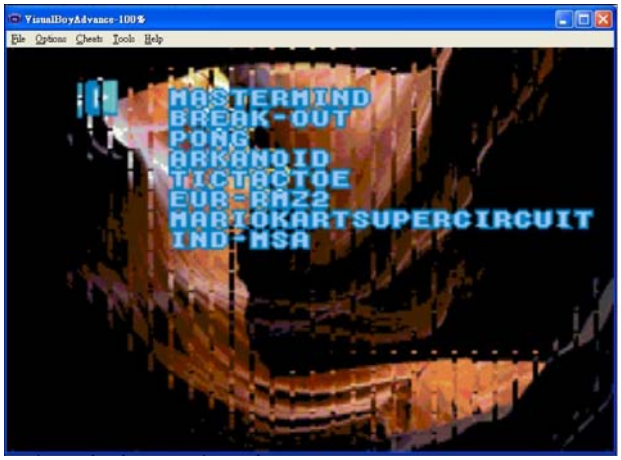


Figure 5. 遊戲選單完成圖

Discussions

在我們程式的實際開發過程中，遇到了以下值得提出討論的地方:

A. Memory Management:

在實做多合一卡帶的過程中，遇到一個較大的瓶頸是記憶體的分配問題。因為 GBA 在讀取 ROM 的過程中，會將 ROM 裡的資料放到 memory 等待 CPU 使用，但是如果合併的遊戲檔案如果大於記憶體可以儲存的大小，就會造成錯誤，這是一個可以改進的地方。而在坊間出現的多合一卡帶則是利用硬體直接控制 GBA 要讀取哪個 ROM 來執行遊戲，跟我們這次實做的作法有根本上的不同。

B. 在 GBA 上顯示圖形:

縱使 HAM 已經提供了許多繪圖相關的 libraries，但是在 GBA 上顯示圖形並控制圖形的移動仍然是需要遊戲開發者花費時間處理的一個重要課題。

Acknowledgements

感謝莊永裕教授與兩位助教們這學期的指導解惑，讓我們在計算機組織與組合語言的領域上能有基礎的認識。感謝那位在網路上分享他們所撰寫之 GBA 遊戲程式碼的遊戲開發者，讓我們可以測試自己寫出來的程式碼。

工作分配

B95902109 吳冠龍：程式架構開發、除錯、遊戲 source 整理、報告撰寫。

B95902077 王柏易：程式架構開發、遊戲 source 整理、程式撰寫與除錯。

B95902069 徐廷倫：程式撰寫與除錯。

B95902076 游書豪：程式撰寫與除錯。