Implementation of an 8051 Emulator on Arduino

8051仿真器在Arduino 上的實現

指導教授:呂仁碩教授　組別:A306　組員姓名:陳冠誌 張景哲 郭粲為 林政緯

Abstract

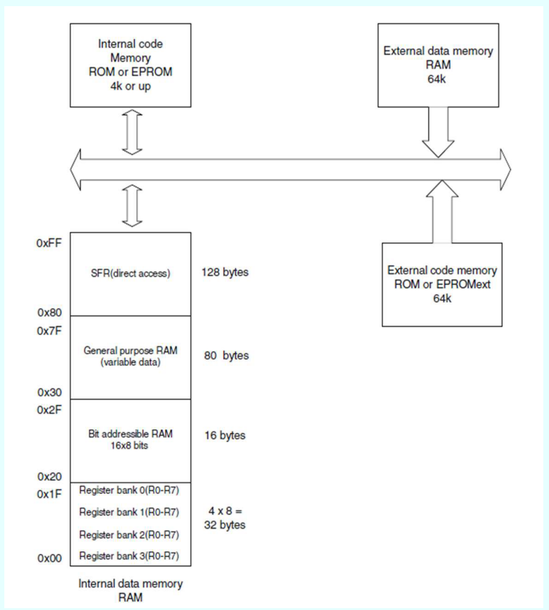
本專題旨在使用Arduino 電路板模擬8051微控制器的運作，以實現在接收相同輸入指令（hex file）時產生相同的輸出結果，成功地將Arduino 解讀輸入的hex file並執行相應的指令。8051微控制器是一種 8 位元單晶片微控制器，具有8位元的data bus 和16位元的address bus。它有4KB 的read-only-memory 儲存器和128位元的內部隨機存取儲存器（RAM）組織，以及定時器/計數器、serial port 和4個通用並行口輸入/輸出端口，以及五個中斷源的中斷控制邏輯。
本專題的研究方法主要分為五個階段。首先是閱讀Intel 的公開8051規格書以及其他資料，瞭解8051的硬體規格、指令集、I/O 等基本知識。接著透過閱讀Emux51的open source code，初步了解用C code 模擬8051的方法，並試圖理解作者的設計。然後在PC 上驗證並修改C code，確認指令能正確執行，並觀察變數之數值與輸入的instructions 之間的關係，確認模擬行為與真實的8051一致。接著將c code 移植至Arduino 平台，由於Arduino 不支援某些c99的library 或語法，我們得透過Arduino 平台的函式庫去達成與C 相同的功能。最後，透過基本I/O 驗證模擬器，連接8051模擬器與Arduino 的真實I/O，在輸入HEX 檔後觀察與Arduino連接的模組是否有照我們所預期執行。
本研究成功地實現了在Arduino 上模擬8051微控制器的運作，並且能夠成功地解讀輸入的hex file 並執行相應的指令。

Research Methodology

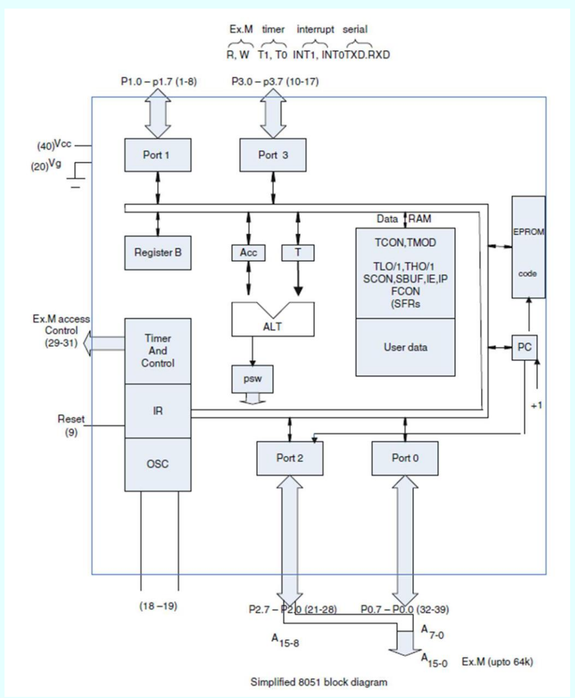
本專題大致可分成五個階段:
(1) 研究8051的架構、指令集、I / O 等等，先對8051有基本的了解:
　　藉由閱讀Intel 的公開 8051規格書以及其他資料，了解這顆晶片的硬體規格，例如Timer、Interrupt、I/O、program memory、instruction set 等等，這些是實現模擬所要具備的基本知識。
(2) 閱讀Emux51 open source code
　　透過網路上的open source，初步了解用C code 模擬8051的方法，試圖理解作者設計的巧思。
(3) 在PC 上驗證與修改C code，確認指令能正確執行
　　由於原設計不完全符合我們的需求，因此更改部分的程式。並透過觀察變數之數值與輸入的instructions 之間的關係，確認模擬行為與真實的 8051 一致。
(4) 將c code 移入至Arduino 平台
　　由於Arduino 不支援某些c99的library 或語法，我們得透過Arduino 平 (Arduino IDE)的函式庫去達成與C 相同的功能。
(5) 透過基本I / O 驗證模擬器
　　有鑑於前面只有在PC 上執行，所以不能算完全的模擬。我們透過連接8051模擬器與Arduino 的真實I / O，在輸入HEX 檔後觀察與Arduino 連接的模組是否有照我們所預期執行。

Experimental Results

8051 structure:



8051 simplified block diagram:



為了做出一個合格的8051模擬器，我們也需完整的模擬出8051的內部記憶體。在參考8051的規格書後，我們宣告了三個 array作為記憶體的模擬。

我們接著要開始設計可以模擬此 Block Diagram的函式。因為全部function太多，所以挑出一個來解釋。

以OpCode 0x74的instruction為例，在8051上的指令是 MOV A,#data，轉換成 C 之後，變成以下程式碼。這個指令是把 #data的值傳入 accumulator 中，我們首先要從code memory中讀到OpCode來呼叫這個函式，讀data，最後寫進accumulator 中，PC+2代表這instruction前面提過的的code_memory array是我們拿來放input hex 檔的地方，也就是待執行的指令。佔兩個bytes, 為了讓全部的指令能照順序好好執行，全域變數 PC 扮演重要角色。

```c
#define CODE_LENGHT 64*1024
#define DATA_LENGHT 128
/*  64K  code memory        */
unsigned char code_memory[CODE_LENGHT];
/*  128B data memory    */
unsigned char data_memory[DATA_LENGHT];
unsigned char sfr_memory[DATA_LENGHT];
```

```c
void mov_a_imm8(unsigned short idx)
{
    unsigned char data;

    data=read_code(idx+1);
    write_Acc(data);
    PC+=2;
}
```

```c
void write_Acc(char data)
{
    sfr_memory[Acc]=data;
}
```

```c
unsigned char read_code(unsigned short addr)
{
    return(code_memory[addr]);
}
```

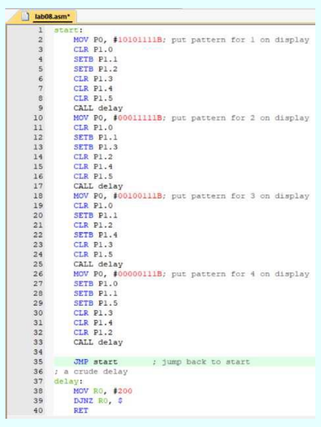
對於8051板，燒錄時HEX檔就是其輸入，因此模擬器的輸入也設置成HEX檔做完全的模擬。在coding前必須先理解HEX檔的格式，以下圖的HEX檔為例:所有數值直接是16進位，8051會將其中的instruction及data存進code memory。以一行為單位，前兩個數值代表size，即該行所有instruction與data所占code memory的byte數，例如第一行”10”代表占16 bytes code memory。size後的四個數字代表offset，即code memory以使用的bytes數，例如第二行”0010”代表code memory已存過16 bytes。offset後兩位是type，type是0代表尚未結束，type1代表讀到最後一行。Type後到最尾兩個數值前就是instruction和data，也就是要存進code memory裡的資料。最後兩個數值是除錯碼。透過以上對HEX檔的理解就可以大致設計出load_hex，我們使用while迴圈一行一行讀HEX檔，下一行的sscanf擷取size、offset、type。type是0(case0)就執行將資料存進code memory的行為，type是1(case1)就跳出迴圈。

```
:10000000D290C291110EC290D291110E80F278C896
:03001000D8FE22F5
:00000001FF
```

Implemented on Arduino and run

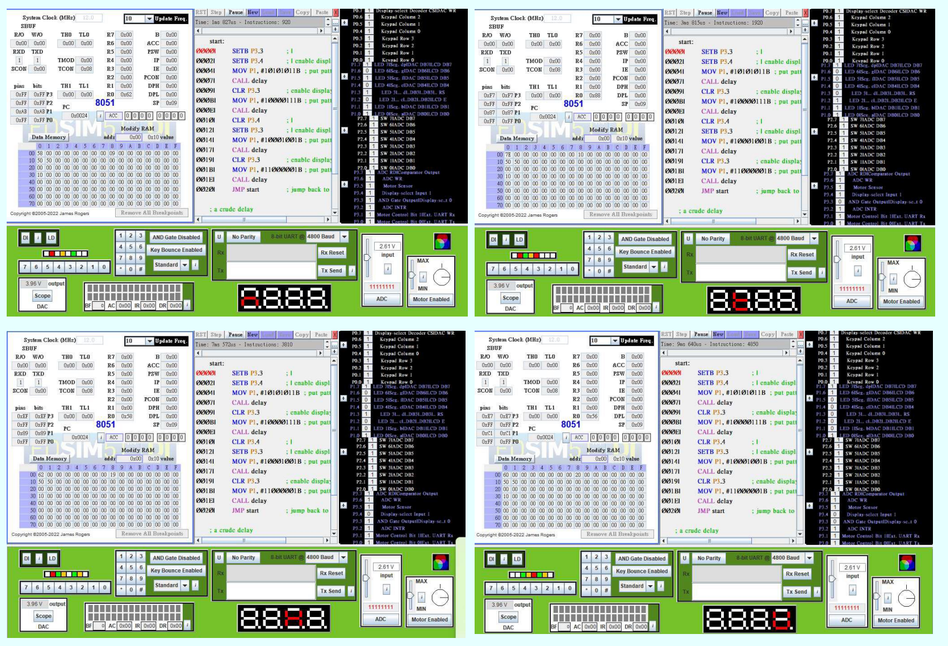
Arduino 的語言跟 C 十分的相近，因此我們的下一步就是將做成功的 C code 模擬器移植進來。再類推一些相似但語法不同的函式如 printf & Serial.print 後，我們便開始設計我們的input output。在 output 端就是盡可能的將 8051 板子上的 port 和 Arduino 作對應，目標讓 Arduino 板在接收到 8051 適用的輸入（這邊是用.HEX）時，能表現出和 8051一樣的輸出結果。而輸入的部分，我們的設計是把原本要餵給 8051 的 HEX file 複製進Arduino IDE 裡的 Serial Monitor 中讓 Arduino 讀。輸入並按下enter後，指令就會被存進code_memory然後開始執行我們前面在 C 部分提到的工作。

最終成果演示:
Input code (in assembly and hex):




```
:100000007580AFC290D291D292C293C294C2951120
:1000100066116611661166116675801FC290D291D5
:10002000D293C292C294C29511661166116611668E
:10003001166758027C290D291C292D294C293C2A7
:1000400095116611661166116611661166758007D2906A
:10005000D291D295C293C294C29211661166116672
:0B0060011661166809A78C8D8FE2255
:00000001FF
```

Performance in 8051 online-simulator:



Our Ardiuno-8051 simulator: