Implementation of 4-core processor of simple operation with Cache memory and Cache Controller using SystemC

組別 / Group No.：第 2 組
組員 / Group Members：40025B20 施凱仁
40043139 曾柏諭
日期 / Date：民國 105 年 6 月 25 日

目錄

實驗目的

本專題構想是希望能建構出一個簡易功能之四核心系統此外每個核心內部都含有一個運算邏輯單元（ALU）以及 4 個 Blocks 空間的快取記憶體（Cache Memory）最後這四個核心都與一個匯流排並聯於一起以形成可以互相通訊溝通甚，此匯流排上不只掛載 4 個核心還掛載一個 256x8-bit 大之 RAM, 而每個核心都只作隨機提取 RAM 中兩個 Data 進行加法運算並且在每個系統時脈週期裡把運算結果的值透過匯流排傳送至其他三個核心之一。系統架構如下圖所示。

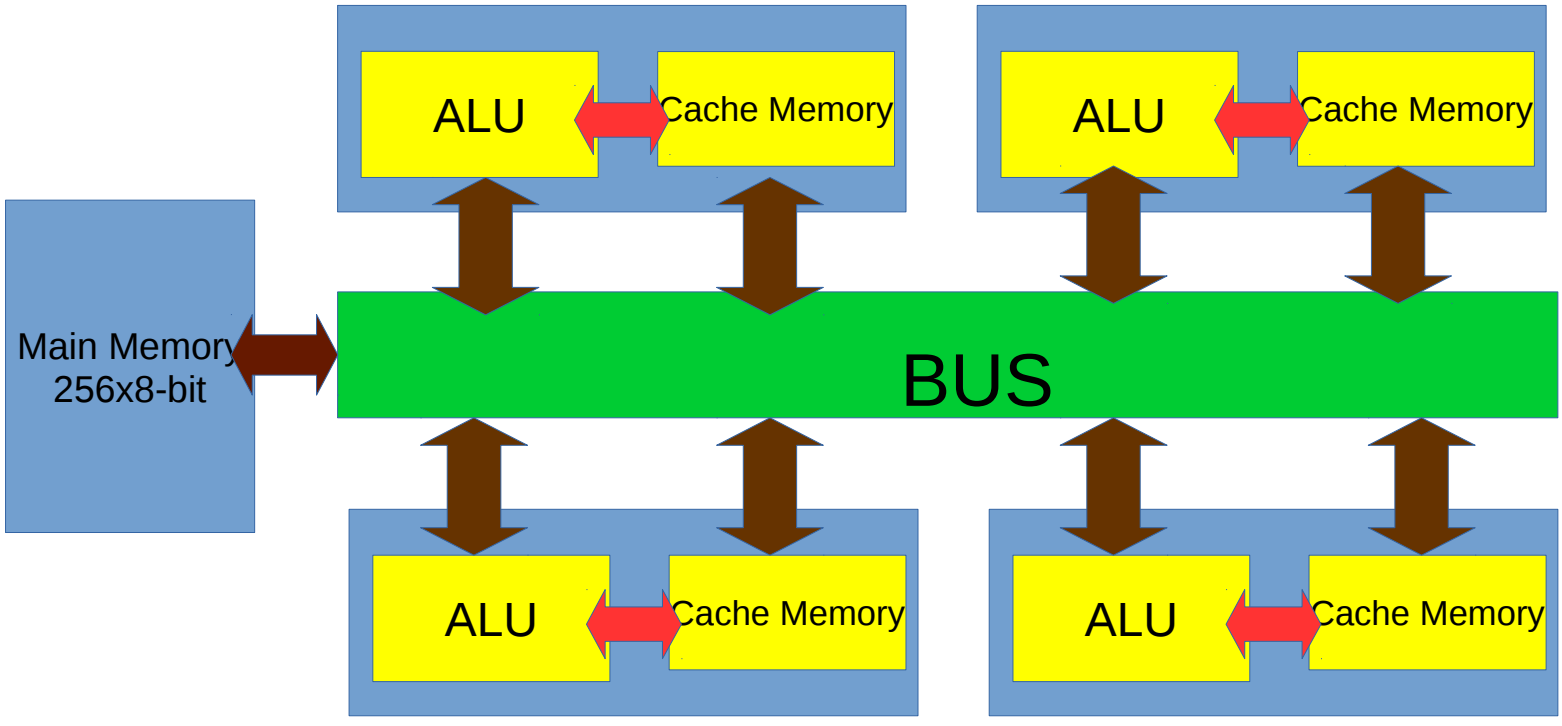


圖 1.系統架構圖

實作 2-1 ALU

此系統之運算邏輯單元就只是作簡單地把從 Cache/RAM 隨機讀取之二比資料作相加運算

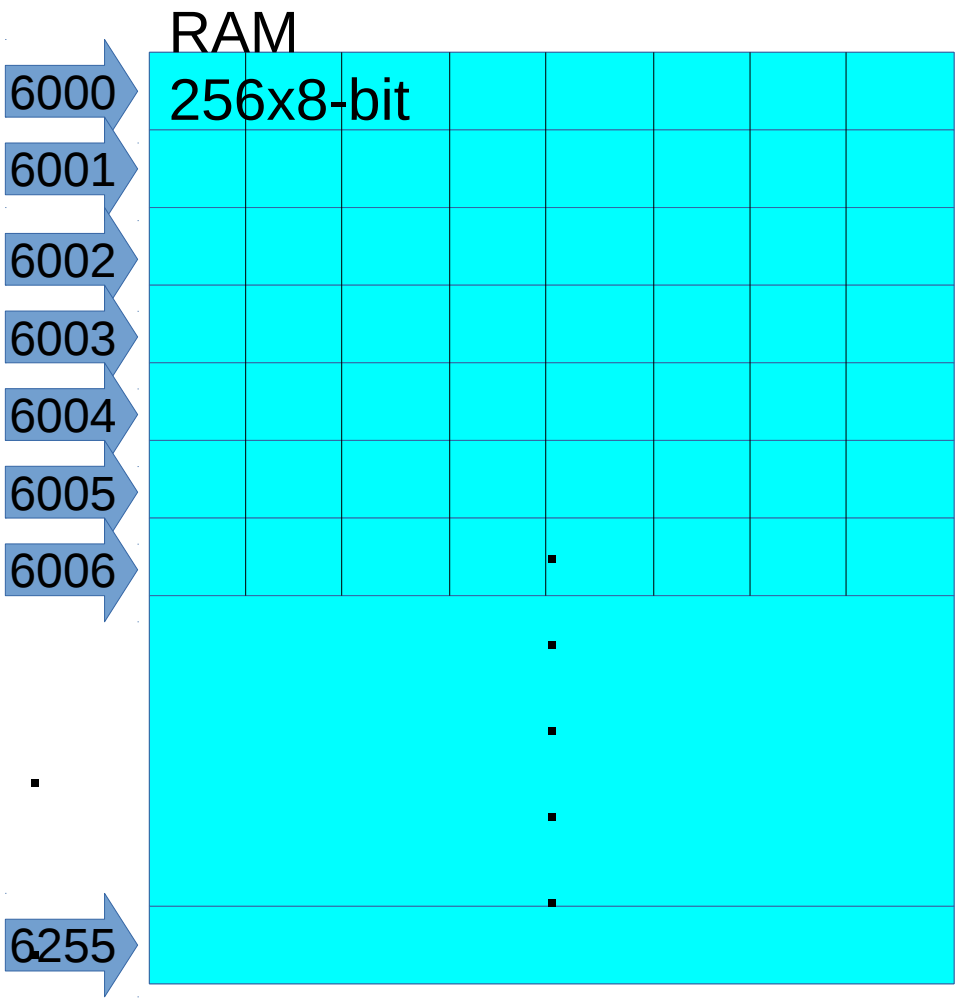
實作 2-2 BUS

匯流排之架構就是 ALU 把運算過的值透過匯流排傳至指定的其他 core 上，每個 core 都有其對應之編號來判斷要傳給誰。程式如下。

```
void bus::write( unsigned addr, int data1, int data2 )
{
        //addr0 write module0 ; 1 write module1 ; 2
write module2 : 3 write module3
        if( addr == 0 )
        ……write_port[0] -> write( addr, data1, data2 );
        else if( addr == 1 )
        ……write_port[1] -> write( addr, data1, data2 );
        else if( addr == 2 )
        ……write_port[2] -> write( addr, data1, data2 );
        else if( addr == 3 )
        ……write_port[3] -> write( addr, data1, data2 );
        else
         printf( "There are only 4 cores were built.\r\n" );


}
```

實作 2-3 RAM

RAM 的部份是當作此四合新的系統之 Main Memory，並且用 C 函式庫中的恩 open file 來建立一個文字檔來模擬。



實作 2-4 CACHE

我們是以 directed mapping 的 Cache 架構。

```
class Cache_Direct_mapped
{
private:
/**
Address :
          [ 7 : 2 ]                                [ 1 : 0 ]
-------------------------------------------------------------------------
|            Tag                              |        Index      |
 -------------------------------------------------------------------------


Cache:
 [15:14]     [ 13 : 8 ]           [ 7 : 0 ]
-------------------------------------------------------------------------
|    valid |          Tag       |                Data              |
-------------------------------------------------------------------------
**/

        uint8_t   *Address;
        uint16_t  *Block;



public:
        void Cache_write( uint8_t addr, uint8_t data );
        uint8_t Cache_fetch( uint8_t addr );

        Cache_Direct_mapped( uint16_t *block );
        int hit_cnt_core;
        int access_cnt_core;


        //~Cache_Direct_mapped();

};
```

實作 2-5 THREAD

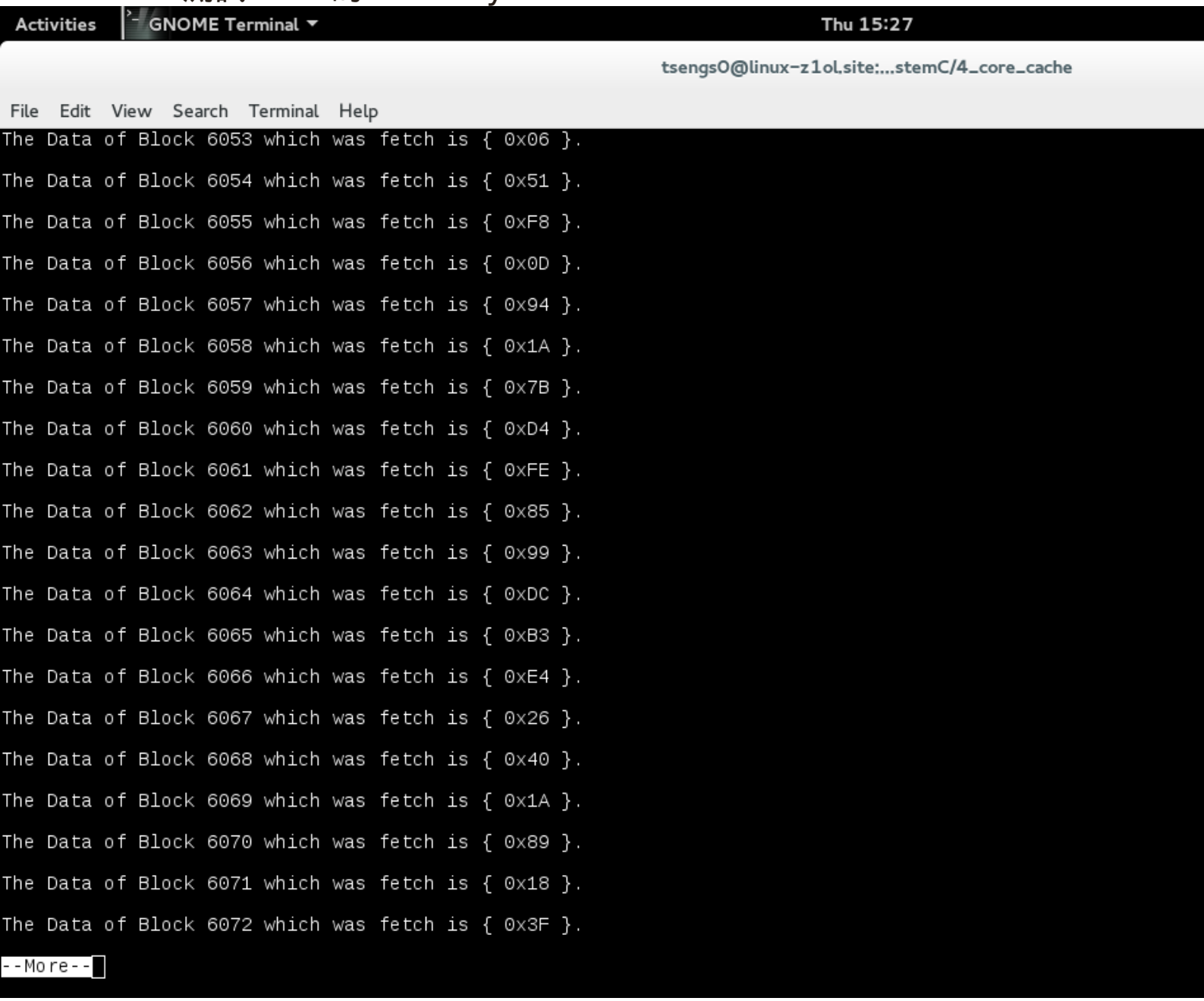
一直不停地再每一個系統時脈週期下把資料傳至依序其他三個 Core。

```
if( address == 0 )
{
            write_port -> write( cnt0++, alu_data[0], alu_data[1] );

             printf( "Transfering the sum of data from Core_0 to
            Core_%d\r\n", cnt0 );

              printf("data[0] = %d, data[1] = %d\r\n\r\n",
            alu_data[0], alu_data[1]);
              cnt0 = cnt0 % 4;
}
```

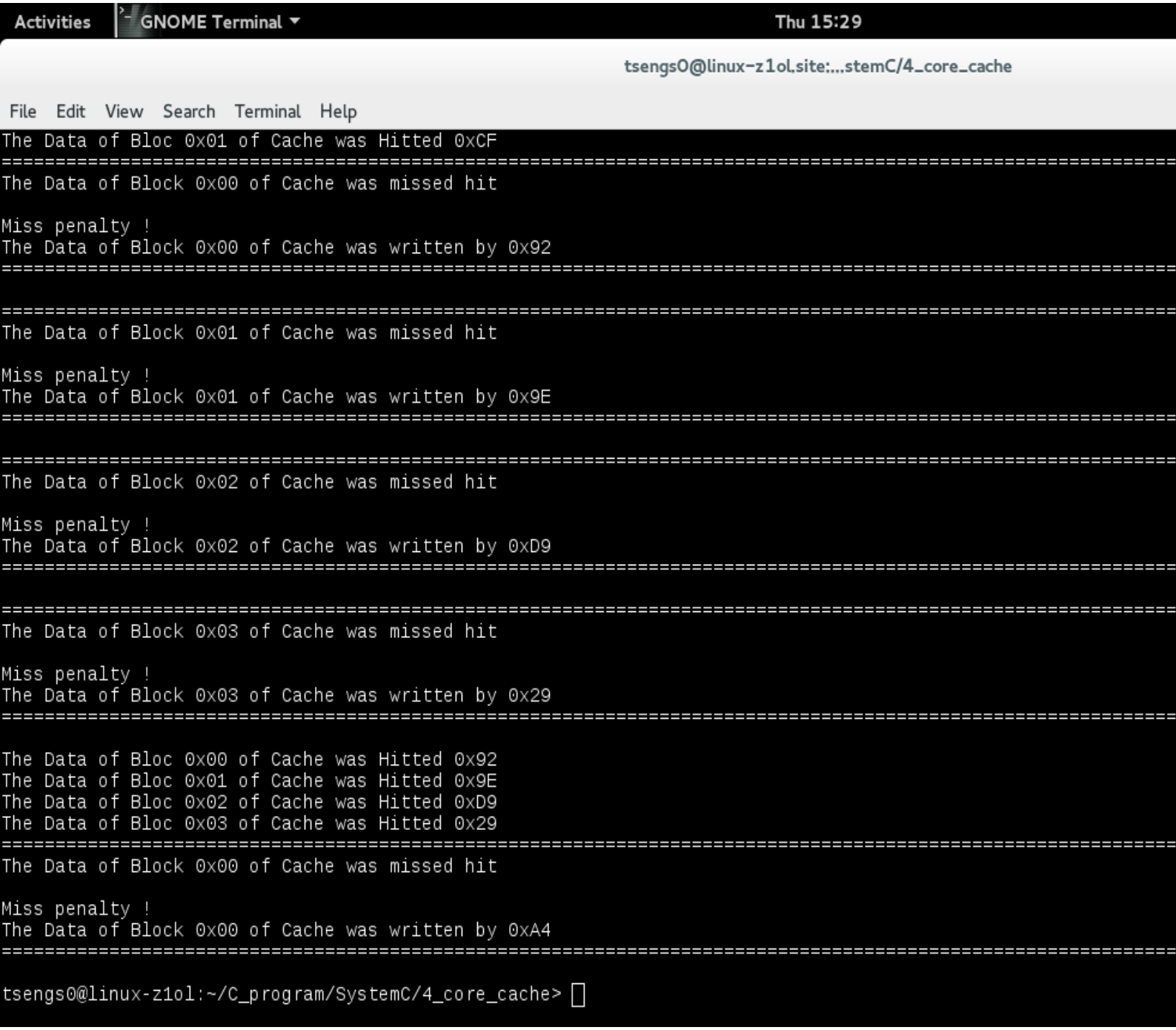
測試與驗證

此部份分三階段測試，依序是測試 RAM 的 memory access、Cache 提取、整體四核心系統的動作測試。

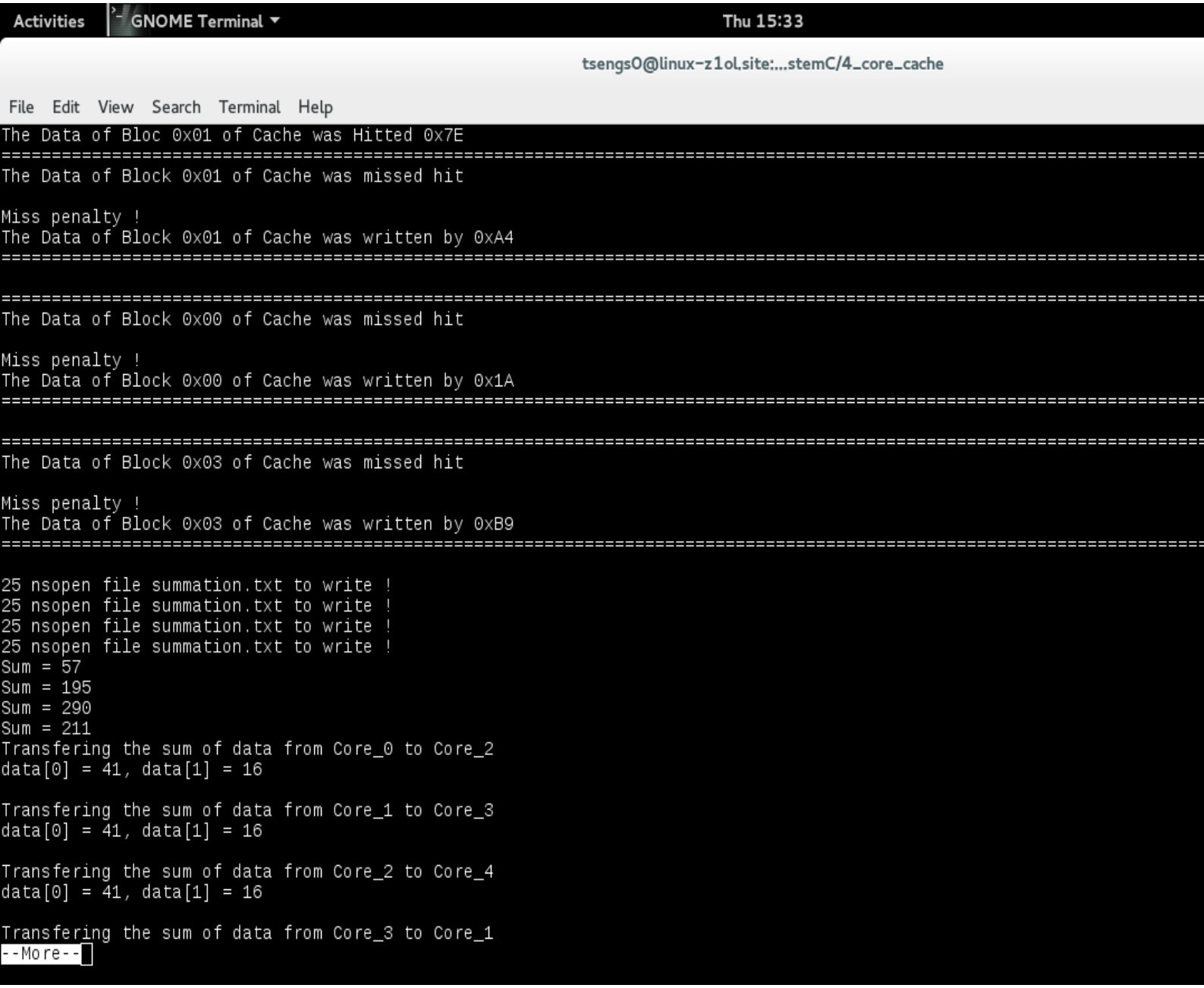
測試 RAM 的 memory access：

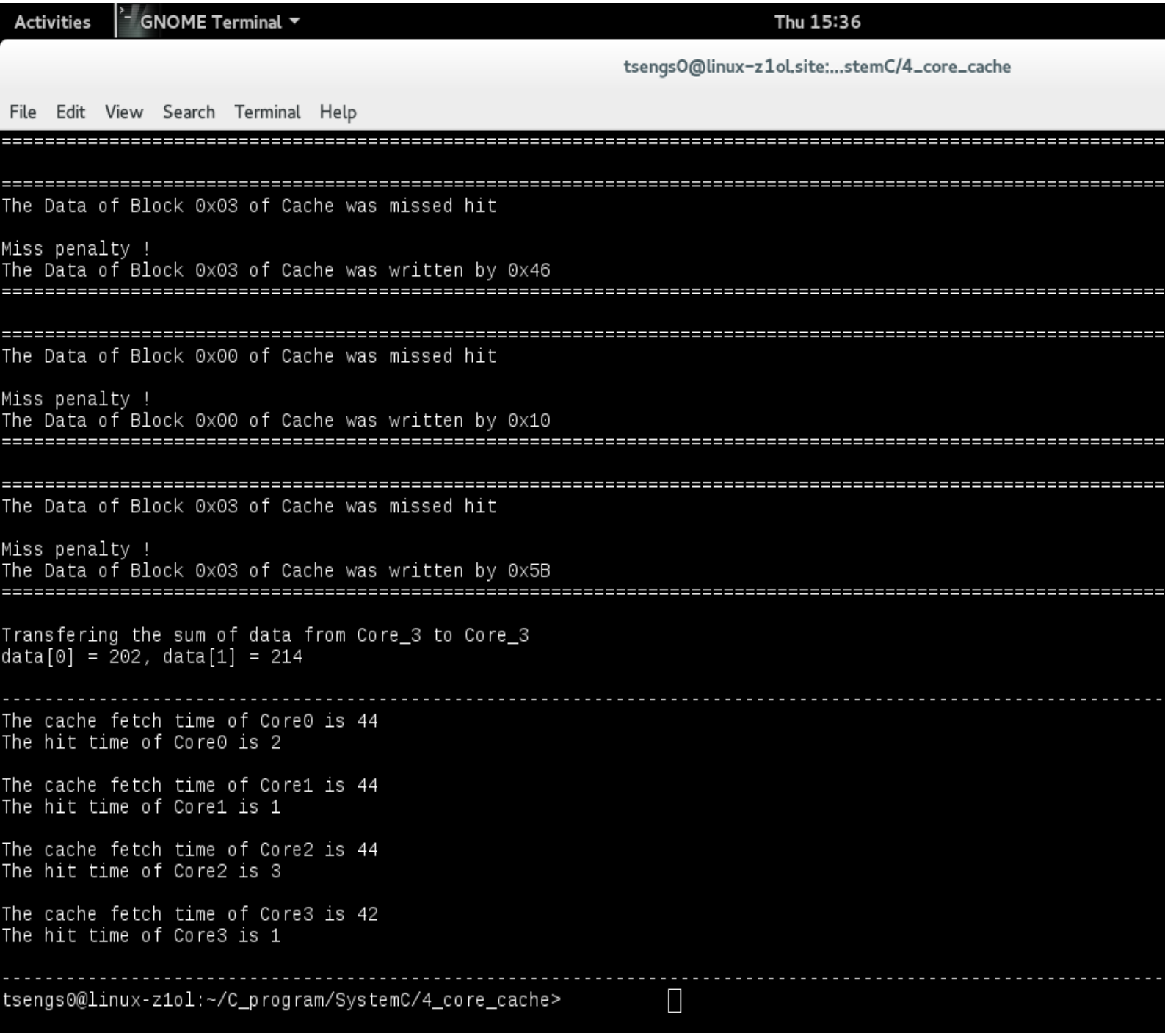


Cache 提取:


Activities
GNOME Terminal
Thu 15:29
tsengs0@linux-z1ol.site:...stemC/4_core_cache
File Edit View Search Terminal Help
The Data of Bloc 0x01 of Cache was Hitted 0xCF
The Data of Block 0x00 of Cache was missed hit
Miss penalty !
The Data of Block 0x00 of Cache was written by 0x92
The Data of Block 0x01 of Cache was missed hit
Miss penalty !
The Data of Block 0x01 of Cache was written by 0x9E
The Data of Block 0x02 of Cache was missed hit
Miss penalty !
The Data of Block 0x02 of Cache was written by 0xD9
The Data of Block 0x03 of Cache was missed hit
Miss penalty !
The Data of Block 0x03 of Cache was written by 0x29
The Data of Bloc 0x00 of Cache was Hitted 0x92
The Data of Bloc 0x01 of Cache was Hitted 0x9E
The Data of Bloc 0x02 of Cache was Hitted 0xD9
The Data of Bloc 0x03 of Cache was Hitted 0x29
The Data of Block 0x00 of Cache was missed hit
Miss penalty !
The Data of Block 0x00 of Cache was written by 0xA4
tsengs0@linux-z1ol:~/C_program/SystemC/4_core_cache>

整體四核心系統的動作測試：





測試結果前兩項測試正常但是第三項測試時才察覺到 Cache 的 size 只有建四個 blocks 而 RAM 確有 256 個 blocks，所以再做 Cache fetch 時 hit 之機率非常低導致 Hit Ratio 始終是在 6.8%以下。

參考文獻

Computer Organiztion And Design 5th Edition from David Patterson, John Hennessy. ISBN-9789124077263