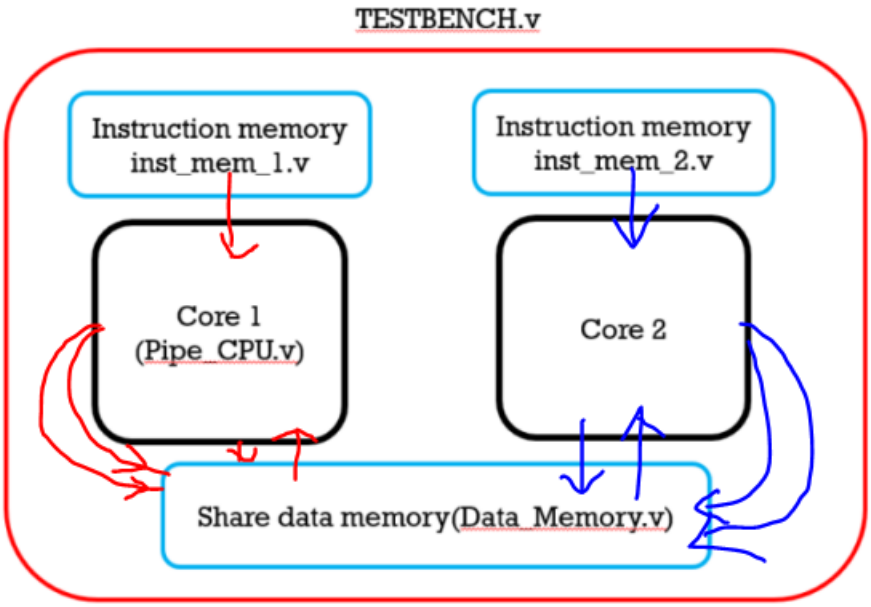
Computer Organization - Lab6

Multi-core cpu

一、系統架構：



把 CPU 裡的 Data Memory 移掉，變成能夠共用的 Share data memory，分別把 core 1, 2 的 MemRead、MemWrite 和要寫入的 data_i 傳入，若 MemRead == 1 再傳出 data。

二、設計模組分析、設計結果：

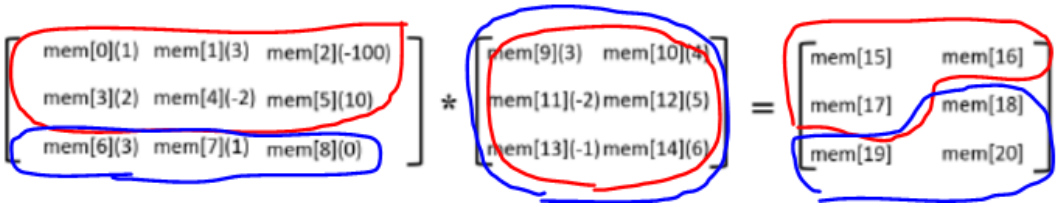
在 testbanch 中新增：

For core 1：MemRead_i_1, MemWrite_i_1, data_i_1, data_o_1

For core 1：MemRead_i_2, MemWrite_i_2, data_i_2, data_o_2

並傳入 testbanch 中的 Data Memory，才能讀 memory 的內容或寫入。

所有的指令由 lw, sw, add, mul 組成



紅色部分由 core 1 負責，藍色則由 core 2。

設計結果：

```
Core1 Register=============================================================
r0=          0, r1=          1, r2=          3, r3=       -100, r4=          2, r5=         -2, r6=         10, r7=          0
r8=          0, r9=          0, r10=         3, r11=          4, r12=        -2, r13=         5, r14=         -1, r15=         6
r16=        97, r17=       -581, r18=         0, r19=          0, r20=         6, r21=         4, r22=        -10, r23=         0
r24=         0, r25=         0, r26=         0, r27=          0, r28=         0, r29=        128, r30=         0, r31=          0
Core2 Register=============================================================
r0=          0, r1=          0, r2=          0, r3=          0, r4=          2, r5=         -2, r6=         10, r7=          3
r8=          1, r9=          0, r10=         3, r11=          4, r12=        -2, r13=         5, r14=         -1, r15=         6
r16=        58, r17=         7, r18=        17, r19=          0, r20=        12, r21=         5, r22=          0, r23=         0
r24=         0, r25=         0, r26=         0, r27=          0, r28=         0, r29=        128, r30=         0, r31=          0

Memory=====================================================
m0=         1, m1=         3, m2=4294967196, m3=         2, m4=4294967294, m5=        10, m6=         3, m7=         1
m8=         0, m9=         3, m10=        4, m11=4294967294, m12=        5, m13=4294967295, m14=         6, m15=        97
m16=4294966715, m17=         0, m18=        58, m19=         7, m20=        17, m21=         0, m22=         0, m23=         0
m24=         0, m25=         0, m26=         0, m27=         0, m28=         0, m29=         0, m30=         0, m31=         0
```

mem[15] = 97,　　mem[16] = -581,　mem[17] = 0,

mem[18] = 58,　　mem[19] = 7 ,　　mem[20] = 17

三、遭遇的困難與解決方法：

一開始 initial memory 的之後，就傳不進各 cpu 中的，後來把 Data Memory 中傳 data 的方式改掉，才能成功傳出。

在算完所有乘法之後，把 data 存回去 memory，只有要寫回去的 memory addr 是對的，其他就會變 x，也是把 Data Memory 中的存 data 的 code 修改一點才成功。

因為不確定指令是不是全部都能用，所以只用的確定的幾個來解決矩陣乘法。

四、作業心得檢討：

Q：Assumed that programmers do not know the platform architecture (i.e. single core or multi-core) how can programmers manage their program partition?

A：由 compiler 重排指令，讓他的 program 可以成功執行。

Q：Assumed that each core has private cache. If core1 write a new data at address 0x123, how could core 2 get the new data from 0x123? (hint: coherence)

A：core 1 和 core 2 間要有通訊機制，第一時間向對方更新自己改變了哪個位址的資料。