計算機結構 Final Project Report

電機五 **B09602017** 白宗民 電機二 羅庭旭 **B12901116**

Design Spec

---

1. 矩陣鏈乘法主流程（matrix_chain_multiplication）

- **輸入參數：**
  - a0：int** 型態，指向各個矩陣資料的指標陣列。
  - a1：每個矩陣的 row 數。
  - a2：每個矩陣的 col 數。
  - a3：總共幾個矩陣（即有幾層乘法）。
- **流程控制：**
  - 若為 1 個矩陣，直接呼叫 malloc 分配新空間並將原始資料複製回傳。
  - 若為多個矩陣，則依序將相鄰兩矩陣相乘，並將中間結果更新作為下一輪的左乘數。
- **記憶體管理：**
  - 每次乘法的結果皆以 malloc 分配記憶體，避免寫入原始輸入區塊導致資料破壞。
  - 每一筆矩陣乘法中所產生的新矩陣會在下一輪中被覆蓋或更新。
- **三重迴圈乘法核心（matrix_multiply）**
  - 經典 ijk 結構，計算每個 C[i][j] 的 dot-product。
  - 所有記憶體操作皆透過 offset 計算，未使用 stack-based 乘法或特殊對齊優化。

---

2. 記憶體與堆疊使用說明

- 所有中間資料與結果皆使用 malloc 分配，避免 stack overflow。
- 組合語言中透過 sp 進行暫存暫存器與中間結果指標的保留與回存，保證呼叫規範（calling convention）正確。
- 單次記憶體分配大小為 m * l * 4，其中 4 為 int 的位元組大小。

Special Techniques Used

1. 動態記憶體配置（malloc）

- 所有中間結果與輸出矩陣皆以 malloc 配置，不使用靜態記憶體池，確保記憶體管理符合要求。
- 配置大小依據 rows × cols × 4 bytes 計算。

2. 三重迴圈矩陣乘法（ijk loop）

- 使用 i、j、k 三層迴圈依序計算 C[i][j] += A[i][k] * B[k][j]。
- 記憶體存取使用 row-major 編排，提高 cache 命中率。

3. GEM5 模擬器效能最佳化實驗

為提升模擬效能（分數越低越好），我們針對 L1/L2 cache 的容量與關聯度進行多組實驗：

109298160.0 --l1i 4kB/2-way --l1d 4kB/2-way --l2 16kB/4-way
125261917.0 --l1i 4kB/8-way --l1d 4kB/8-way --l2 16kB/8-way
96631008.0 --l1i 4kB/16-way --l1d 4kB/16-way --l2 1kB/16-way
96631008.0 --l1i 4kB/16-way --l1d 4kB/16-way --l2 4kB/16-way
96636408.0 --l1i 4kB/32-way --l1d 4kB/32-way --l2 1kB/16-way
104417144.0 --l1i 8kB/16-way --l1d 8kB/16-way --l2 2kB/16-way
279730800.0 --l1i 2kB/16-way --l1d 2kB/16-way --l2 1kB/16-way

Team's Division of Work

白宗民：此版本的 assembly code implementation, 包含 Config Ablation Study

羅庭旭：基於此版本的 DP 優化, 嘗試做出有 DP implementation 的 version

Reflections

我們其實一開始是想要一起寫 DP version 的 code，後來發現一起寫 assembly 其實蠻難的（在我們都不是非常熟練的狀況下），不像是 python 很好做 module。所以後來就變成我先寫出這版過 standard baseline，學弟因為剛好在修演算法，幫忙實作 DP 的 optimization。但後來發現 DP 真的很難搞，我們又有花了幾天討論還是做不出來。只能說會寫 assembly 的都是神。