# Report-Final project 計算機結構

**106030019　吳岱容**

1. Data structure:

   Two classes:

   1. address:

      member:

      vector <int> data (將 address 的 string word 轉成單一 int)

      long long index (十進位 index)

      long long tag(十進位 tag)

      string original_data(原讀進來的 string)

      functions:

      void calculate_index (vector<int> A, int n)(將其轉換為十進位)

      void calculate_tag (vector<int> A, int n,int z)(將其轉換為十進位)

   2. Cache:

      member:

      vector<vector<long long>> block (模擬 cache 的 block)

      vector<vector<bool>> NRU (NRU bits)

      functions:

      bool update(address a)(模擬 cache hit miss)

      global functions:

      reading(char **argv)(讀進 argument 內的檔案,初始化紀錄 miss 相關)

      simulation(int A,int m)(開始進行模擬 cache 行為)

      output(char**argv)(印出結果)

      lg(int n)(算 $\log_2$)

2. flow chart:

一開始先讀進在 argument1 & argument2 的檔案資訊，將其放入適合的資料結構，然後初始化儲存 miss 相關資訊的資料結構。（reading(char ** argv)）

```
void reading(char** argv){
    freopen(argv[1], "r", stdin);

    string tmp;
    stringstream ss;

    while (cin>>tmp) {
        if ((int)tmp[0]<=57 && (int)tmp[0]>=48) {
            ss<<tmp;
            ss>>address_bit;
            ss.str("");
            ss.clear();
            break;
        }
    }
```

接著設置 index 的對位，由於只用了 LSB 所以直接往後移 tag 的長度。

```
int main(int argc, char **argv) {
    reading(argv);

        //LSB
        for (int i=0; i<index_length; i++) {
            A[i]=tag_length+i;
        }
        simulation(A, m);
```

開始模擬 cache 行為，一開始先將讀進來的資料轉換（calculate_index & calculate_tag），然後針對每一個讀進來的資料做模擬 cache 的行為（update 來模擬 hit or miss，NRU 的模擬）。(simulation (int A, int m))

```
void simulation(vector<int> A,int n){

    for (int i=0; i<input_size; i++) {
        chart[i].calculate_index(A, n);
        chart[i].calculate_tag(A, n, tag_length);
    }

    Cache cache(cache_set,associativity);

//counting miss ans
    miss_num=0;
    int count=0;

    for (int i=0; i<input_size; i++) {
        if (cache.update(chart[i])==true) {
            miss_num++;
            miss[count]=1;
        }
        else{
            miss[count]=0;
        }
        count++;
    }
    for (int i=0;i<n;i++){
            ans_index[i]=address_bit-A[i]-1;
    }
}
```

在 update(address a)中，我們會先看看有無可以 hit 的 cache，若有 NRU 的

bit 要變為 0，若無則看看有沒有 bit 是 1 的，若有就替換，若無就把所有 bit 翻成 1 再替換（代表全部都是 0），回傳 miss or hit。

```
bool update(address a){
    for (int i=0; i<associativity; i++) {
        if (block[a.index][i]==a.tag) {
            NRU[a.index][i]=0;
            return 0;
        }
    }

    for (int i=0; i<associativity; i++) {
        if (NRU[a.index][i]==1) {
            block[a.index][i]=a.tag;
            NRU[a.index][i]=0;
            return 1;
        }
    }
    for (int i=0; i<associativity; i++){
        NRU[a.index][i]=1;
    }
    for (int i=0; i<associativity; i++) {
        if (NRU[a.index][i]==1) {
            block[a.index][i]=a.tag;
            NRU[a.index][i]=0;
            return 1;
        }
   }
    return 1;
}
```

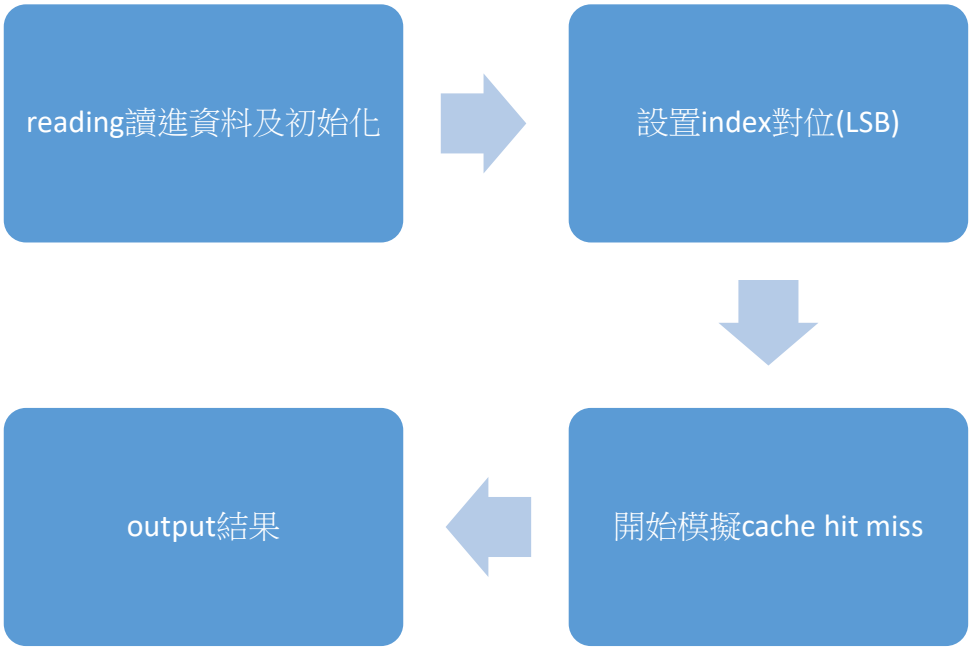
接著就是進行 output 到 argument3（index.rpt）。

```
void output(char** argv){
    freopen(argv[3], "w", stdout);

    cout<<"Address bits: "<<address_bit<<endl;
    cout<<"Block size: "<<block_size<<endl;
    cout<<"Cache sets: "<<cache_set<<endl;
    cout<<"Associativity: "<<associativity<<endl;
    cout<<endl;
    cout<<"Offset bit count: "<<byte_offset<<endl;
    cout<<"Indexing bit count: "<<index_length<<endl;
    cout<<"Indexing bits:";
    for (int i=0; i<index_length; i++) {
        cout<<" "<<ans_index[i];
    }
    cout<<endl<<endl;

    cout<<".benchmark "<<name<<endl;

    for (int i=0; i<input_size; i++) {
        cout<<chart[i].original_data;
        if (miss[i]==1) {
            cout<<" miss"<<endl;
        }
        else{
            cout<<" hit"<<endl;
        }
    }
    cout<<".end"<<endl<<endl;
    cout<<"Total cache miss count: "<<miss_num<<endl;
    fclose(stdout);
}
```


reading讀進資料及初始化
設置index對位(LSB)
開始模擬cache hit miss
output結果