Deap Daru
60004190025

Experiment 4

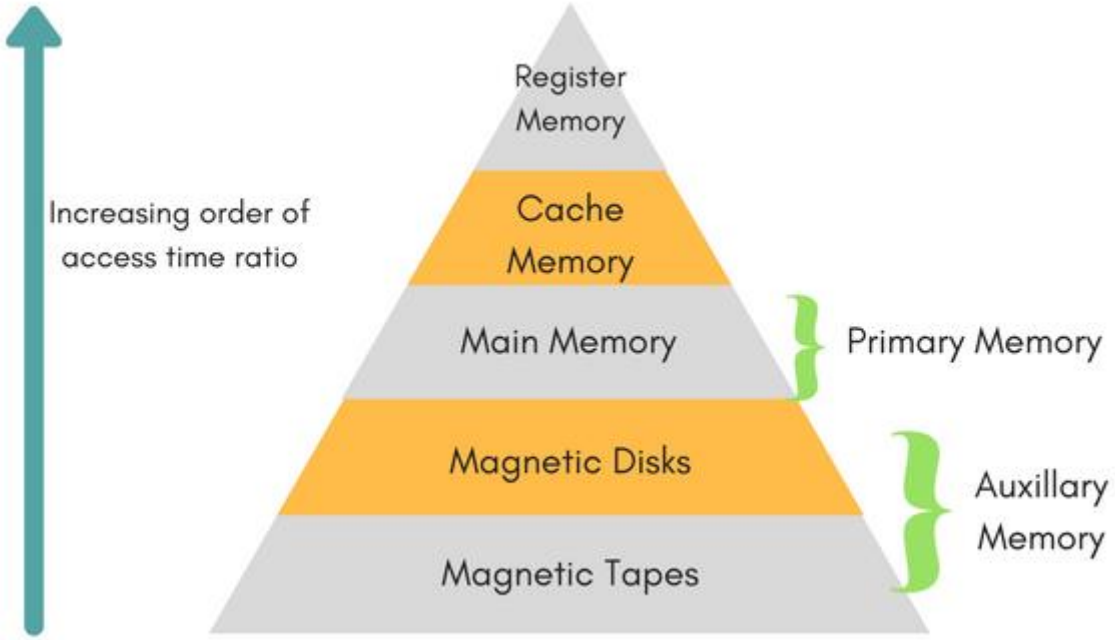
Aim:

Implement Sequential memory organization with following details:

- Processor can access 1 word = 4 bytes.
- L1 cache can store max = 32 words.
- L2 cache can store max =128 words.
- Main memory can store max = 2048 bytes.

Theory:

In the Computer System Design, Memory Hierarchy is an enhancement to organize the memory such that it can minimize the access time. The Memory Hierarchy was developed based on a program behaviour known as locality of references. The figure below clearly demonstrates the different levels of memory hierarchy:



This Memory Hierarchy Design is divided into 2 main types:

- External Memory or Secondary Memory – Comprising of Magnetic Disk, Optical Disk, Magnetic Tape i.e., peripheral storage devices which are accessible by the processor via I/O Module.
- Internal Memory or Primary Memory – Comprising of Main Memory, Cache Memory & CPU registers. This is directly accessible by the processor.

Levels of memory:

- Level 1 or Register –

  It is a type of memory in which data is stored and accepted that are immediately stored in CPU. Most commonly used register is accumulator, Program counter, address register etc.

- Level 2 or Cache memory –

  It is the fastest memory which has faster access time where data is temporarily stored for faster access.

- Level 3 or Main Memory –

  It is memory on which computer works currently. It is small in size and once power is off data no longer stays in this memory.

- Level 4 or Secondary Memory –

  It is external memory which is not as fast as main memory but data stays permanently in this memory.

Types of Cache:

- Primary Cache –

  A primary cache is always located on the processor chip. This cache is small and its access time is comparable to that of processor registers.

- Secondary Cache –

  Secondary cache is placed between the primary cache and the rest of the memory. It is referred to as the level 2 (L2) cache. Often, the Level 2 cache is also housed on the processor chip.

Code:

memory.cpp

```
#include<bits/stdc++.h>
#include<time.h>


using namespace std;


int main() {
  vector<double> lc1(32,-1);
  vector<double> lc2(64,-1);
```

```
  double lc1Time = 20;
  double lc2Time = 60;
  double mainMemoryTime = 120;
  double totalHits = 0;
  double lc1Hits = 0;
  double lc2Hits = 0;
  double count = 0;
  int fr1 = -1;
  int re1 = 0;
  int fr2 = -1;
  int re2 = 0;
  char key = 100;


  srand(time(0));


  cout<<"Sequential Memory Organization:"<<endl<<endl;
  cout<<" "<<"Requirement"<<setw(22)<<"Location of Hit"<<" "<<setw(15)<<"Avg.
Access Time"<<endl;


  while(key--) {
    count++;
    double avgTime = 0;
    double input = rand()%512;
    bool f1 = false;
    bool f2 = false;


    for(double i=0; i<32; i++) {
      if(lc1[i]==input) {
        lc1Hits++;
        f1 = true;
        f2 = true;
        avgTime += (lc1Hits/(count))*lc1Time + (1-
(lc1Hits/(count)))*(lc2Hits/(count-lc1Hits))*(lc1Time+lc2Time) + (1-
(lc1Hits/(count)))*(1-(lc2Hits/(count-
lc1Hits)))*1*(lc1Time+lc2Time+mainMemoryTime);
```

```
        cout<<setw(12)<<input<<" "<<setw(20)<<"L1 Cache"<<"
"<<setw(8)<<avgTime<<endl;
      }
    }

    if(!f1) {
      double b = input/2;
      for(double i=0; i<64; i++) {
        if(lc2[i]==b) {
          lc2Hits++;
          f2 = true;
          // found in 2
          if(re1==fr1) {
            re1 = (re1+1)%32;
          }
          fr1 = (fr1+1)%32;
          lc1[fr1] = input;
          avgTime += (lc1Hits/(count))*lc1Time + (1-
(lc1Hits/(count)))*(lc2Hits/(count-lc1Hits))*(lc1Time+lc2Time) + (1-
(lc1Hits/(count)))*(1-(lc2Hits/(count-
lc1Hits)))*1*(lc1Time+lc2Time+mainMemoryTime);
          cout<<setw(12)<<input<<" "<<setw(20)<<"L2 Cache"<<"
"<<setw(8)<<avgTime<<endl;
        }
      }
    }
    if(!f2) {
      double b = input/2;
      if(re2==fr2) {
        re2 = (re2+1)%64;
      }
      fr2 = (fr2+1)%64;
      lc2[fr2] = b;
      if(re1==fr1) {
        re1 = (re1+1)%32;
      }
```

```
      fr1 = (fr1+1)%32;

      lc1[fr1] = input;

      avgTime += (lc1Hits/(count))*lc1Time + (1-
(lc1Hits/(count)))*(lc2Hits/(count-lc1Hits))*(lc1Time+lc2Time) + (1-
(lc1Hits/(count)))*(1-(lc2Hits/(count-
lc1Hits)))*1*(lc1Time+lc2Time+mainMemoryTime);

      cout<<setw(12)<<input<<" "<<setw(20)<<"Main Memory"<<"
"<<setw(8)<<avgTime<<endl;

    }

  }


  cout<<endl;

  cout<<"L1 Hit Ratio (H1) = "<<lc1Hits<<"/"<<count<<" =
"<<(double)lc1Hits/count<<endl;

  cout<<"L1 Access Time (T1) = "<<lc1Time<<" ns"<<endl<<endl;

  cout<<"L2 Hit Ratio (H2) = "<<lc2Hits<<"/"<<(count-lc1Hits)<<" =
"<<(double)lc2Hits/(count-lc1Hits)<<endl;

  cout<<"L2 Access Time (T2) = "<<lc2Time<<" ns"<<endl<<endl;

  cout<<"Main Memory Hit Ratio (Hm) = "<<1<<endl;

  cout<<"Main memory Access Time = "<<mainMemoryTime<<" ns"<<endl<<endl;

  cout<<"Average Access Time = [H1*T1] + [(1-H1)*H2*(T1+T2)] + [(1-H1)*(1-
H2)*Hm*(T1+T2+Tm)]"<<endl;

  double finalAns = ((double)lc1Hits/count)*lc1Time + (1-
((double)lc1Hits/count))*((double)lc2Hits/(100-lc1Hits))*(lc1Time+lc2Time) + (1-
((double)lc1Hits/count))*(1-((double)lc2Hits/(count-
lc1Hits)))*1*(lc1Time+lc2Time+mainMemoryTime);

  cout<<"Average Access Time = "<<finalAns<<" ns"<<endl;


  return 0;

}
```

Output:

```
PS C:\Users\Deap Daru\Desktop\Sem 5\POA\Pracs\4> g++ -o memory memory.cpp
PS C:\Users\Deap Daru\Desktop\Sem 5\POA\Pracs\4> ./memory
Sequential Memory Organization:

 Requirement       Location of Hit Avg. Access Time
         353          Main Memory      200
         372          Main Memory      200
          89          Main Memory      200
         189          Main Memory      200
         265          Main Memory      200
         423          Main Memory      200
         379          Main Memory      200
         107          Main Memory      200
           8          Main Memory      200
         217          Main Memory      200
         155          Main Memory      200
         294          Main Memory      200
         192          Main Memory      200
          47          Main Memory      200
         348          Main Memory      200
         217             L1 Cache   188.75
         264          Main Memory  189.412
         375          Main Memory      190
         100          Main Memory  190.526
         351          Main Memory      191
         476          Main Memory  191.429
          12          Main Memory  191.818
         178          Main Memory  192.174
         138          Main Memory    192.5
         338          Main Memory    192.8
         250          Main Memory  193.077
         306          Main Memory  193.333
         363          Main Memory  193.571
         208          Main Memory  193.793
          99          Main Memory      194
         143          Main Memory  194.194
         205          Main Memory  194.375
         381          Main Memory  194.545
          34          Main Memory  194.706
         329          Main Memory  194.857
         259          Main Memory      195
         245          Main Memory  195.135
         429          Main Memory  195.263
           7          Main Memory  195.385
         228          Main Memory    195.5
          65          Main Memory   195.61
         320          Main Memory  195.714
          46          Main Memory  195.814
          19          Main Memory  195.909
         262          Main Memory      196
          56          Main Memory  196.087
         178             L1 Cache   192.34
         132          Main Memory    192.5
```

252	Main Memory	192.653
189	L2 Cache	190.4
301	Main Memory	190.588
315	Main Memory	190.769
418	Main Memory	190.943
163	Main Memory	191.111
302	Main Memory	191.273
87	Main Memory	191.429
49	Main Memory	191.579
402	Main Memory	191.724
467	Main Memory	191.864
80	Main Memory	192
462	Main Memory	192.131
101	Main Memory	192.258
410	Main Memory	192.381
125	Main Memory	192.5
3	Main Memory	192.615
465	Main Memory	192.727
116	Main Memory	192.836
217	L2 Cache	191.176
368	Main Memory	191.304
71	Main Memory	191.429
27	Main Memory	191.549
262	L1 Cache	189.167
190	Main Memory	189.315
283	Main Memory	189.459
166	Main Memory	189.6
146	Main Memory	189.737
469	Main Memory	189.87
229	Main Memory	190
347	Main Memory	190.127
122	Main Memory	190.25
335	Main Memory	190.37
296	Main Memory	190.488
452	Main Memory	190.602
492	Main Memory	190.714
44	Main Memory	190.824
305	Main Memory	190.93
261	Main Memory	191.034
81	Main Memory	191.136
455	Main Memory	191.236
66	Main Memory	191.333
364	Main Memory	191.429
139	Main Memory	191.522
8	Main Memory	191.613
240	Main Memory	191.702
74	Main Memory	191.789
379	Main Memory	191.875
476	Main Memory	191.959
458	Main Memory	192.041
1	Main Memory	192.121
455	L1 Cache	190.4

L1 Hit Ratio (H1) = 4/100 = 0.04
L1 Access Time (T1) = 20 ns

```
L2 Hit Ratio (H2) = 2/96 = 0.0208333
L2 Access Time (T2) = 60 ns

Main Memory Hit Ratio (Hm) = 1
Main memory Access Time = 120 ns

Average Access Time = [H1*T1] + [(1-H1)*H2*(T1+T2)] + [(1-H1)*(1-
H2)*Hm*(T1+T2+Tm)]
Average Access Time = 190.4 ns
```

Conclusion:

Thus, we've implemented sequential memory organization in C++ programming language successfully on the given problem statement in a 2-Level Cache Memory architecture for 100 different number words required by the processor.