22BI13028 - Nguyen Tai Anh- Practical 4

LOW LEVEL SIMULATOR: 351 Cache Simulator

Block size:

Associativity = 1 way

- **Summary of Cache Structure:**
  - Cache Size: 32 bytes
  - Associativity: 1-way
  - Block Size: 8 bytes
  - Number of Sets: 4
  - Tag Size: Remaining bits after block offset bits
  - Index Size: 2 bits
  - Offset Size: log2(8)= 3 bits

- **Execute a Read memory request to address 0x09**
  - Read: $0x09_{(16)}$ → Convert to: $00001001_{(2)}$
  - Split address in binary into TIO breakdown
    - Tag: 000
    - Index: 01
    - Offset: 001
  - Index = $01_{(10)}$ = $1_{(2)}$→ Checking Set 1
  - Looking for Tag 0 → MISS
  - LRU status update
  - How many cache hits, cache misses ?
    - Cache hits: 0
    - Cache misses: 1
  - Analyze where the blocks are loaded in the cache
    - Block read into cache from memory at address 0x8.
  - Analyze the write policy
    - Not write → Dirty bit = 0
    - Cache Data and Physical Memory did not change → Valid data 0xd0→ Valid bit = 1

- **Execute a Read memory request to address 0x13**
  - Read: $0x13_{(16)}$ → Convert to: $00010011_{(2)}$
  - Split address in binary into TIO breakdown

        - Tag: 000
        - Index: 10
        - Offset: 011
    - Index = $10_{(10)}$ = $2_{(2)}$→ Checking Set 2
    - Looking for Tag 0 → MISS
    - LRU statuses updated.
    - How many cache hits, cache misses ?
        - Cache hits: 0
        - Cache misses: 2
    - Analyze where the blocks are loaded in the cache
        - Block read into cache from memory at address 0x10.
    - Analyze the write policy
        - Not write → Dirty bit = 0
        - Cache Data and Physical Memory did not change → Valid data 0xca→ Valid bit = 1

- **Execute a Read memory request to address 0x21**
    - Read: $0x21_{(16)}$ → Convert to: $00100001_{(2)}$
    - Split address in binary into TIO breakdown
        - Tag: 001
        - Index: 00
        - Offset: 001
    - Index = $00_{(10)}$ = $0_{(2)}$→ Checking Set 0
    - Looking for Tag 1 → MISS
    - LRU statuses updated.
    - How many cache hits, cache misses ?
        - Cache hits: 0
        - Cache misses: 3
    - Analyze where the blocks are loaded in the cache
        - Block read into cache from memory at address 0x20.
    - Analyze the write policy
        - Not write → Dirty bit = 0
        - Cache Data and Physical Memory did not change → Valid data 0x4a→ Valid bit = 1

- **Execute a Read memory request to address 0x2a**
  - Read: $0x2a_{(16)}$ → Convert to: $00101010_{(2)}$
  - Split address in binary into TIO breakdown
    - Tag: 001
    - Index: 01
    - Offset: 010
  - Index = $01_{(10)}$ = $1_{(2)}$→ Checking Set 1
  - Looking for Tag 1 → MISS
  - LRU statuses updated.
  - How many cache hits, cache misses ?
    - Cache hits: 0
    - Cache misses: 4
  - Analyze where the blocks are loaded in the cache
    - Block read into cache from memory at address 0x28.
  - Analyze the write policy
    - Not dirty, so no need to write back to memory
    - Cache Data and Physical Memory did not change → Valid data 0xae→ Valid bit = 1

- **Execute a Read memory request to address 0x15**
  - Read: $0x2a_{(16)}$ → Convert to: $00010101_{(2)}$
  - Split address in binary into TIO breakdown
    - Tag: 000
    - Index: 10
    - Offset: 101
  - Index = $10_{(10)}$ = $2_{(2)}$→ Checking Set 2
  - Looking for Tag 0 → HIT in line 0
  - LRU statuses updated.
  - How many cache hits, cache misses ?
    - Cache hits: 1
    - Cache misses: 4
  - Analyze the write policy
    - Not dirty → Dirty bit = 0
    - Cache Data and Physical Memory did not change → Valid data 0x95→ Valid bit = 1

- **Execute a Write memory request to address 0x11**
    - Write: 0x3a$_{(16)}$ → Read 0x11 → Convert 0x11 to: 00010001$_{(2)}$
    - Split address in binary into TIO breakdown
        - Tag: 000
        - Index: 10
        - Offset: 001
    - Index = 10$_{(10)}$ = 2$_{(2)}$→ Checking Set 2
    - Looking for Tag 0 → HIT in line 0
    - LRU statuses updated.
    - How many cache hits, cache misses ?
        - Cache hits: 2
        - Cache misses: 4
    - Analyze the write policy
        - Write back: set Dirty bit.
        - Dirty bit = 1

- **Execute a Read memory request to address 0x33**
    - Read: 0x33$_{(16)}$ → Convert to: 00010011$_{(2)}$
    - Split address in binary into TIO breakdown
        - Tag: 001
        - Index: 10
        - Offset: 011
    - Index = 10$_{(10)}$ = 2$_{(2)}$→ Checking Set 2
    - Looking for Tag 1 → MISS
    - LRU statuses updated.
    - How many cache hits, cache misses ?
        - Cache hits: 2
        - Cache misses: 5
    - Analyze where the blocks are loaded in the cache
        - Block read into cache from memory at address 0x30
    - Analyze the write policy
        - Dirty, so block written to memory at address 0x10
        - Cache Data and Physical Memory did not change → Valid data 0x7a→ Valid bit = 1

- **Execute a Read memory request to address 0x11**
  - Read: $0x33_{(16)}$ → Convert to: $00010001_{(2)}$
  - Split address in binary into TIO breakdown
    - Tag: 000
    - Index: 10
    - Offset: 001
  - Index = $10_{(10)} = 2_{(2)}$→ Checking Set 2
  - Looking for Tag 0 → MISS
  - LRU statuses updated.
  - How many cache hits, cache misses ?
    - Cache hits: 2
    - Cache misses: 6
  - Analyze where the blocks are loaded in the cache
    - Block read into cache from memory at address 0x10
  - Analyze the write policy
    - Not Dirty → Dirty bit = 0
    - Cache Data and Physical Memory did not change → Valid data 0x3a→ Valid bit = 1

Associativity = 2 ways

- **Summary of Cache Structure:**
  - Cache Size: 32 bytes
  - Associativity: 2-way
  - Block Size: 8 bytes
  - Number of Sets: 2
  - Tag Size: Remaining bits after block offset bits
  - Index Size: 1 bit
  - Offset Size: log2(8)= 3 bits

- **Analysis**
  - Sets: With 2 sets, memory accesses will be divided between the two sets based on the index bits.

  - Lines per Set: Each set contains 2 lines due to 2-way associativity.
  - Replacement Policy: Typically, a Least Recently Used (LRU) or random replacement policy is used to manage which cache line to evict when the set is full.

- **Execute a Read memory request to address 0x09**
  - Read: $0x09_{(16)}$ → Convert to: $00001001_{(2)}$
  - Split address in binary into TIO breakdown
    - Tag: 0000
    - Index: 1
    - Offset: 001
  - Index = $1_{(10)} = 1_{(2)}$→ Checking Set 1
  - Looking for Tag 0 → MISS
  - LRU status update
  - How many cache hits, cache misses ?
    - Cache hits: 0
    - Cache misses: 1
  - Analyze where the blocks are loaded in the cache
    - Block read into cache from memory at address 0x8.
  - Analyze the write policy
    - Not write → Dirty bit = 0
    - Cache Data and Physical Memory did not change → Valid data 0xd0→ Valid bit = 1

- **Execute a Read memory request to address 0x13**
  - Read: $0x13_{(16)}$ → Convert to: $00010011_{(2)}$
  - Split address in binary into TIO breakdown
    - Tag: 0001
    - Index: 0
    - Offset: 011
  - Index = $0_{(10)} = 0_{(2)}$→ Checking Set 0
  - Looking for Tag 1 → MISS
  - LRU statuses updated.

- How many cache hits, cache misses ?
    - Cache hits: 0
    - Cache misses: 2
- Analyze where the blocks are loaded in the cache
    - Block read into cache from memory at address 0x10.
- Analyze the write policy
    - Not write → Dirty bit = 0
    - Cache Data and Physical Memory did not change → Valid data 0xca→ Valid bit = 1

- **Execute a Read memory request to address 0x21**
    - Read: 0x21$_{(16)}$ → Convert to: 00100001$_{(2)}$
    - Split address in binary into TIO breakdown
        - Tag: 0010
        - Index: 0
        - Offset: 001
    - Index = 0$_{(10)}$ = 0$_{(2)}$→ Checking Set 0
    - Looking for Tag 2 → MISS
    - LRU statuses updated.
    - How many cache hits, cache misses ?
        - Cache hits: 0
        - Cache misses: 3
    - Analyze where the blocks are loaded in the cache
        - Block read into cache from memory at address 0x20.
    - Analyze the write policy
        - Not write → Dirty bit = 0
        - Cache Data and Physical Memory did not change → Valid data 0x4a→ Valid bit = 1

- **Execute a Read memory request to address 0x2a**
    - Read: 0x2a$_{(16)}$ → Convert to: 00101010$_{(2)}$
    - Split address in binary into TIO breakdown
        - Tag: 0010
        - Index: 1
        - Offset: 010

- Index = $1_{(10)}$ = $1_{(2)}$→ Checking Set 1
- Looking for Tag 2 → MISS
- LRU statuses updated.
- How many cache hits, cache misses ?
  - Cache hits: 0
  - Cache misses: 4
- Analyze where the blocks are loaded in the cache
  - Block read into cache from memory at address 0x28.
- Analyze the write policy
  - Not dirty, so no need to write back to memory
  - Cache Data and Physical Memory did not change → Valid data 0xae→ Valid bit = 1

- **Execute a Read memory request to address 0x15**
  - Read: 0x2a$_{(16)}$ → Convert to: $00010101_{(2)}$
  - Split address in binary into TIO breakdown
    - Tag: 0001
    - Index: 0
    - Offset: 101
  - Index = $0_{(10)}$ = $0_{(2)}$→ Checking Set 0
  - Looking for Tag 1 → HIT in line 0
  - LRU statuses updated.
  - How many cache hits, cache misses ?
    - Cache hits: 1
    - Cache misses: 4
  - Analyze the write policy
    - Not dirty → Dirty bit = 0
    - Cache Data and Physical Memory did not change → Valid data 0x95→ Valid bit = 1

- **Execute a Write memory request to address 0x11**
  - Write: 0x3a$_{(16)}$ → Convert 0x11 to: $00010001_{(2)}$
  - Split address in binary into TIO breakdown
    - Tag: 0001
    - Index: 0

        - Offset: 001
    - Index = $0_{(10)} = 0_{(2)} \rightarrow$ Checking Set 0
    - Looking for Tag 1 → HIT in line 0
    - LRU statuses updated.
    - How many cache hits, cache misses ?
        - Cache hits: 2
        - Cache misses: 4
    - Analyze the write policy
        - Write back: set Dirty bit.
        - Dirty bit = 1

- **Execute a Read memory request to address 0x33**
    - Read: $0x33_{(16)} \rightarrow$ Convert to: $00010011_{(2)}$
    - Split address in binary into TIO breakdown
        - Tag: 0011
        - Index: 0
        - Offset: 011
    - Index = $0_{(10)} = 0_{(2)} \rightarrow$ Checking Set 0
    - Looking for Tag 3 → MISS
    - LRU statuses updated.
    - How many cache hits, cache misses ?
        - Cache hits: 2
        - Cache misses: 5
    - Analyze where the blocks are loaded in the cache
        - Block read into cache from memory at address 0x30
    - Analyze the write policy
        - Not Dirty → Dirty bit = 0
        - Cache Data and Physical Memory did not change → Valid data 0x7a→ Valid bit = 1

- **Execute a Read memory request to address 0x11**
    - Read: $0x33_{(16)} \rightarrow$ Convert to: $00010001_{(2)}$
    - Split address in binary into TIO breakdown
        - Tag: 0001
        - Index: 0

        - Offset: 001
    - Index = $0_{(10)} = 0_{(2)} \rightarrow$ Checking Set 0
    - Looking for Tag 1 → HIT in line 0
    - LRU statuses updated.
    - How many cache hits, cache misses ?
        - Cache hits: 3
        - Cache misses: 5
    - Analyze the write policy
        - Dirty bit = 1
        - Cache Data and Physical Memory did not change → Valid data 0x3a→ Valid bit = 1

Associativity = 4 ways

- **Summary of Cache Structure:**
    - Cache Size: 32 bytes
    - Associativity: 4-way
    - Block Size: 8 bytes
    - Number of Sets: 1
    - Tag Size: Remaining bits after block offset bits
    - Index Size: 0 bit
    - Offset Size: log2(8)= 3 bits

- **Analysis**
    - Set: With only one set, all memory accesses map to this single set.
    - Lines per Set: Each set contains 4 lines due to 4-way associativity.
    - Replacement Policy: Typically, a Least Recently Used (LRU) or random replacement policy is used to manage which cache line to evict when the set is full.

- **Execute a Read memory request to address 0x09**
    - Read: $0x09_{(16)} \rightarrow$ Convert to: $00001001_{(2)}$
    - Split address in binary into TIO breakdown

        - Tag: 00001
        - Index:
        - Offset: 001
    - Looking for Tag 01 → MISS
    - LRU status update
    - How many cache hits, cache misses ?
        - Cache hits: 0
        - Cache misses: 1
    - Analyze where the blocks are loaded in the cache
        - Invalid Line 0 chosen for replacement.
        - Block read into cache from memory at address 0x8.
    - Analyze the write policy
        - Not write → Dirty bit = 0
        - Cache Data and Physical Memory did not change → Valid data 0xd0→ Valid bit = 1

- **Execute a Read memory request to address 0x13**
    - Read: $0x13_{(16)}$ → Convert to: $00010011_{(2)}$
    - Split address in binary into TIO breakdown
        - Tag: 00010
        - Index:
        - Offset: 011
    - Looking for Tag 02 → MISS
    - LRU statuses updated.
    - How many cache hits, cache misses ?
        - Cache hits: 0
        - Cache misses: 2
    - Analyze where the blocks are loaded in the cache
        - Block read into cache from memory at address 0x10.
        - Invalid Line 1 chosen for replacement
    - Analyze the write policy
        - Not write → Dirty bit = 0
        - Cache Data and Physical Memory did not change → Valid data 0xca→ Valid bit = 1

- **Execute a Read memory request to address 0x21**
    - Read: 0x21$_{(16)}$ → Convert to: 00100001$_{(2)}$
    - Split address in binary into TIO breakdown
        - Tag: 00100
        - Index:
        - Offset: 001
    - Looking for Tag 04 → MISS
    - LRU statuses updated.
    - How many cache hits, cache misses ?
        - Cache hits: 0
        - Cache misses: 3
    - Analyze where the blocks are loaded in the cache
        - Block read into cache from memory at address 0x20.
        - Invalid Line 2 chosen for replacement.
    - Analyze the write policy
        - Not write → Dirty bit = 0
        - Cache Data and Physical Memory did not change → Valid data 0x4a→ Valid bit = 1

- **Execute a Read memory request to address 0x2a**
    - Read: 0x2a$_{(16)}$ → Convert to: 00101010$_{(2)}$
    - Split address in binary into TIO breakdown
        - Tag: 00101
        - Index:
        - Offset: 010
    - Looking for Tag 05 → MISS
    - LRU statuses updated.
    - How many cache hits, cache misses ?
        - Cache hits: 0
        - Cache misses: 4
    - Analyze where the blocks are loaded in the cache
        - Block read into cache from memory at address 0x28.
        - Invalid Line 3 chosen for replacement.
    - Analyze the write policy
        - Not dirty, so no need to write back to memory

        - Cache Data and Physical Memory did not change → Valid data 0xae→ Valid bit = 1

- **Execute a Read memory request to address 0x15**
    - Read: $0x2a_{(16)}$ → Convert to: $00010101_{(2)}$
    - Split address in binary into TIO breakdown
        - Tag: 00010
        - Index:
        - Offset: 101
    - Looking for Tag 02 → HIT in line 1
    - LRU statuses updated.
    - How many cache hits, cache misses ?
        - Cache hits: 1
        - Cache misses: 4
    - Analyze the write policy
        - Not dirty → Dirty bit = 0
        - Cache Data and Physical Memory did not change → Valid data 0x95→ Valid bit = 1

- **Execute a Write memory request to address 0x11**
    - Write: $0x3a_{(16)}$ → Convert 0x11 to: $00010001_{(2)}$
    - Split address in binary into TIO breakdown
        - Tag: 00010
        - Index:
        - Offset: 001
    - Looking for Tag 02 → HIT in line 1
    - LRU statuses updated.
    - How many cache hits, cache misses ?
        - Cache hits: 2
        - Cache misses: 4
    - Analyze the write policy
        - Write back: set Dirty bit → Dirty bit = 1
        - Dirty bit = 1

- **Execute a Read memory request to address 0x33**

- Read: $0x33_{(16)}$ → Convert to: $00010011_{(2)}$
- Split address in binary into TIO breakdown
  - Tag: 00110
  - Index:
  - Offset: 011
- Looking for Tag 06 → MISS
- LRU statuses updated.
- How many cache hits, cache misses ?
  - Cache hits: 2
  - Cache misses: 5
- Analyze where the blocks are loaded in the cache
  - Block read into cache from memory at address 0x30
  - Line 0 is the least recently used.
- Analyze the write policy
  - Not dirty → Dirty bit = 0
  - Cache Data and Physical Memory did not change → Valid data 0x7a→ Valid bit = 1

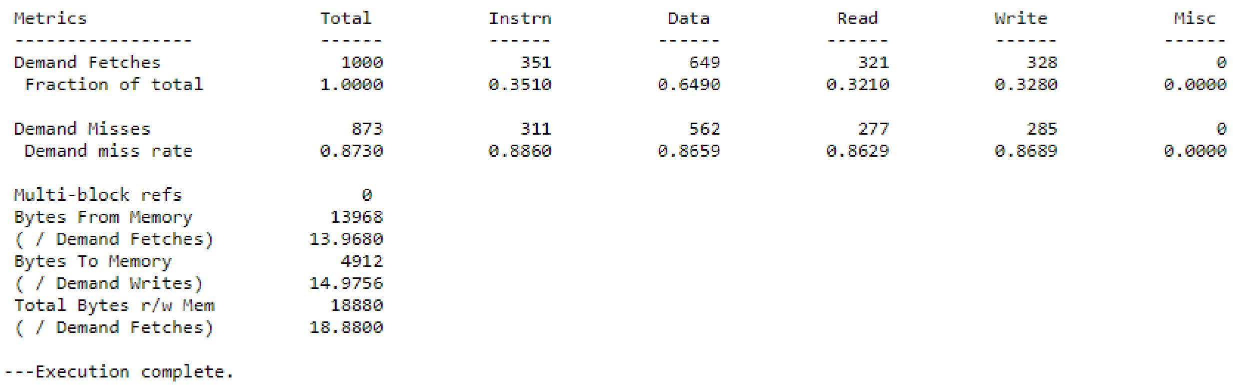
- **Execute a Read memory request to address 0x11**
  - Read: $0x33_{(16)}$ → Convert to: $00010001_{(2)}$
  - Split address in binary into TIO breakdown
    - Tag: 00010
    - Index:
    - Offset: 001
  - Looking for Tag 02 → HIT in line 1
  - LRU statuses updated.
  - How many cache hits, cache misses ?
    - Cache hits: 3
    - Cache misses: 5
  - Analyze the write policy
    - Dirty → Dirty bit = 1
    - Cache Data and Physical Memory did not change → Valid data 0x3a→ Valid bit = 1

Cache memory organization

First simulation: MemoryTraceExample.mem

- **General configuration:**
  - Show access trace: yes
  - Nb Levels: 1
  - Level L1: Unified cache
  - Memory: 1kmots
  - Memory access time: 100ns
  - Memory access trace:

- **Cache Configuration**
  - For Data cache
  - Cache size: 128
  - Block size : 16 mots/bytes
  - Cache access time: 10ns
  - Associativity : Complete
  - Write policy : LRU
  - Allocation policy : write allocate
  - For Instruction cache

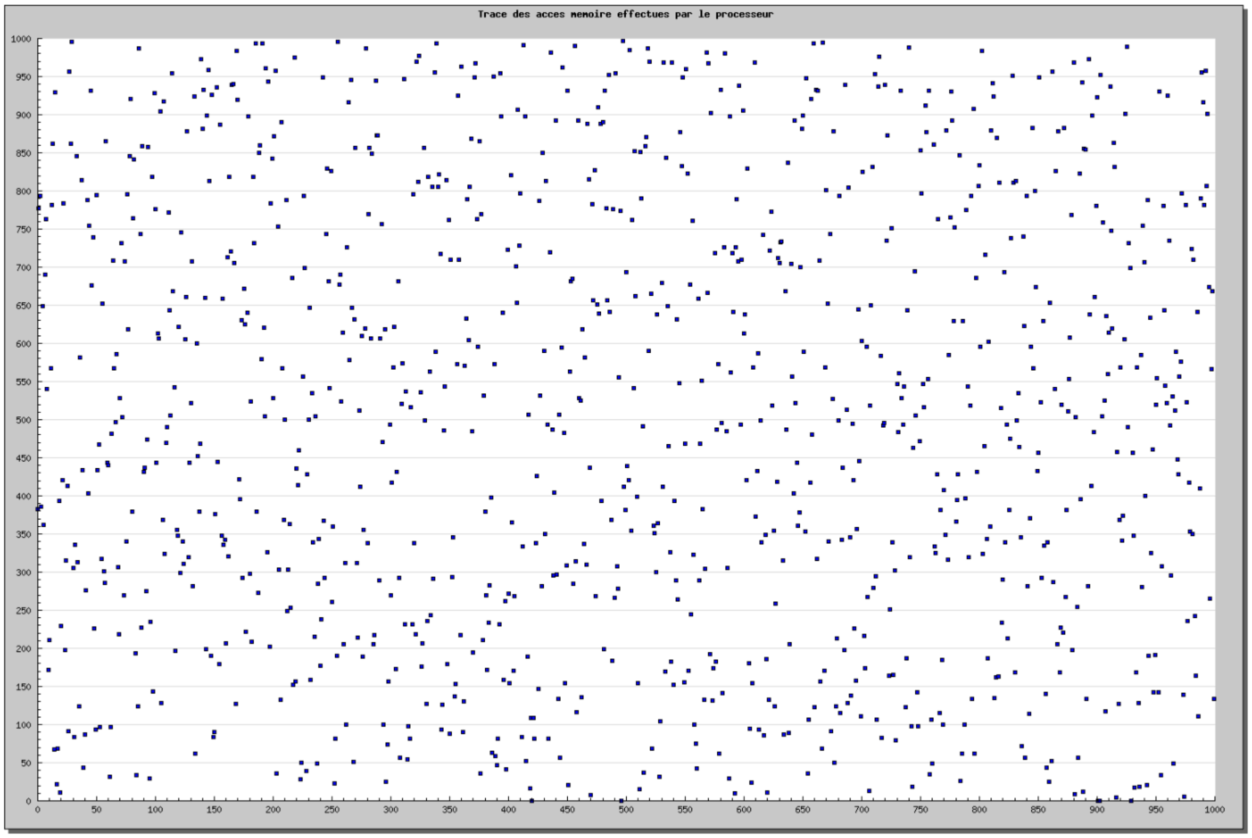


```
 Metrics                     Total        Instrn        Data          Read         Write         Misc
 -----------------           ------       ------        ------        ------       ------        ------
 Demand Fetches               1000          351           649           321          328            0
  Fraction of total          1.0000       0.3510        0.6490        0.3210       0.3280        0.0000

 Demand Misses                 873          311           562           277          285            0
  Demand miss rate           0.8730       0.8860        0.8659        0.8629       0.8689        0.0000

 Multi-block refs                0
 Bytes From Memory           13968
 ( / Demand Fetches)       13.9680
 Bytes To Memory              4912
 ( / Demand Writes)        14.9756
 Total Bytes r/w Mem         18880
 ( / Demand Fetches)       18.8800

---Execution complete.
```



- **Result:**
    - Total fetches: 1000
    - Total misses: 873
    - Total hits: 1000 - 873 = 127 → hit ratio: 12.7%
    - = 0.127*10 + (1-0,127)*100 = 88,57 ns
    - Hit time = 10 ns
    - Miss time = 16*10 + 100 = 260 ns

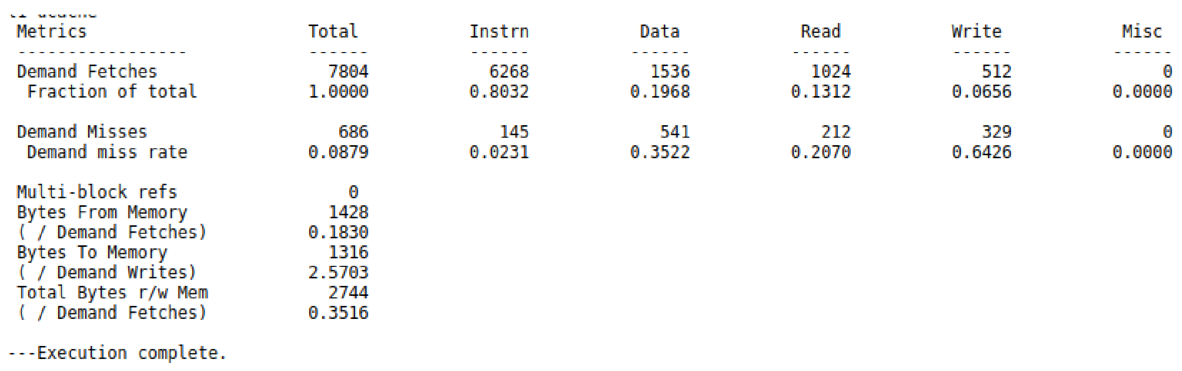
      ➔ Total time of memory access = 10*127 + 260*873 = 228250 ns
- **Conclusion**
    - The Hit ratio is too big so it will cause more time to load from memory.

Second simulation: Ex.asm.mem

- **General configuration:**
    - Show access trace: yes

- Nb Levels: 1
- Level L1: Unified cache
- Memory: 1kmots
- Memory access time: 100ns
- Memory access trace:

- **Cache Configuration**
    - For Data cache
    - Cache size: 128
    - Cache access time: 10ns
    - Associativity : Direct
    - Write policy : LRU
    - Allocation policy : write allocate
    - Cache size 128 words (mots)
    - Cache blocks size 4 words
    - Replacement policy : LRU
    - Associativity : Direct
    - Allocation policy : write allocate
    - Cache access time = 10 ns
    - Memory Access time = 100 ns

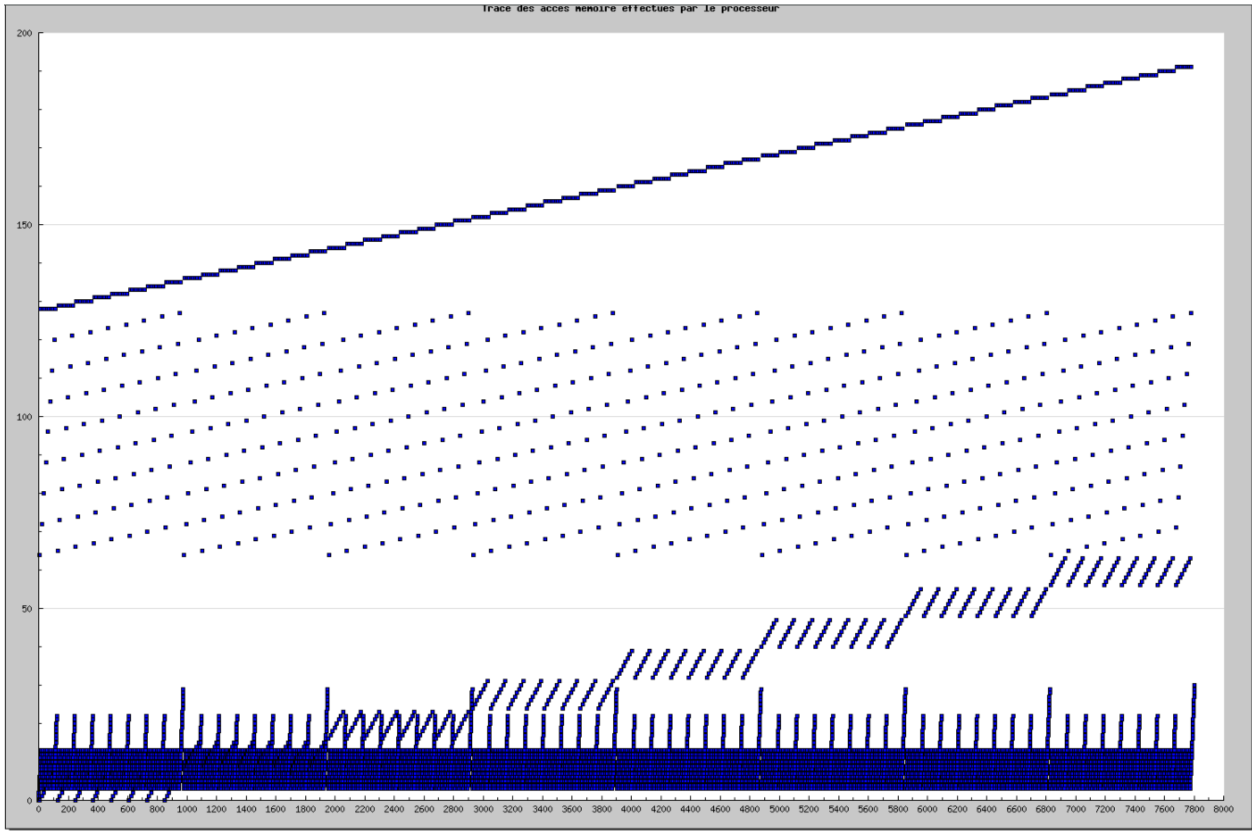


```
 Metrics                    Total       Instrn        Data        Read       Write        Misc
 -----------------         ------       ------      ------      ------      ------      ------
 Demand Fetches              7804         6268        1536        1024         512           0
  Fraction of total        1.0000       0.8032      0.1968      0.1312      0.0656      0.0000

 Demand Misses                686          145         541         212         329           0
  Demand miss rate         0.0879       0.0231      0.3522      0.2070      0.6426      0.0000

 Multi-block refs              0
 Bytes From Memory          1428
 ( / Demand Fetches)      0.1830
 Bytes To Memory            1316
 ( / Demand Writes)       2.5703
 Total Bytes r/w Mem        2744
 ( / Demand Fetches)      0.3516

---Execution complete.
```



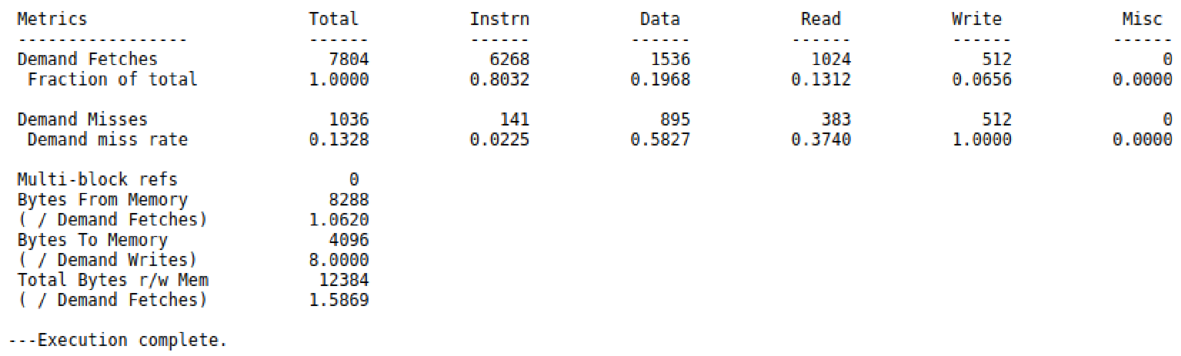
- **Result:**
  - Total fetches: 7804, total misses: 686
  - Total hits: 7804 - 686 = 7118
  - Hit ratio: 0.91
  - Average Memory Access Time = 0.91*10 + (1 - 0.91)*100= 18.1ns
  - Hit time/miss time:
  - Hit time = 10ns
  - Miss time = 4*10 + 100 = 140ns
    ➔Total time of memory access = 10*7118 + 140*686 = 167220ns
- **Conclusion**
  - This ratio is much better than the previous one

Third simulation: Ex.asm.mem

- **General configuration:**

    - Show access trace: yes
    - Nb Levels: 1
    - Level L1: Unified cache
    - Memory: 1kmots
    - Memory access time: 100ns
    - Memory access trace:

- **Cache Configuration**
    - **1st case**
        - For Data cache
        - Cache size: 128
        - Cache access time: 10ns
        - Associativity : Direct
        - Write policy : LRU
        - Allocation policy : write allocate
        - Cache size 128 words (mots)
        - Cache blocks size 8 words
        - Replacement policy : LRU
        - Associativity : Direct
        - Allocation policy : write allocate
        - Cache access time = 10 ns
        - Memory Access time = 100 ns

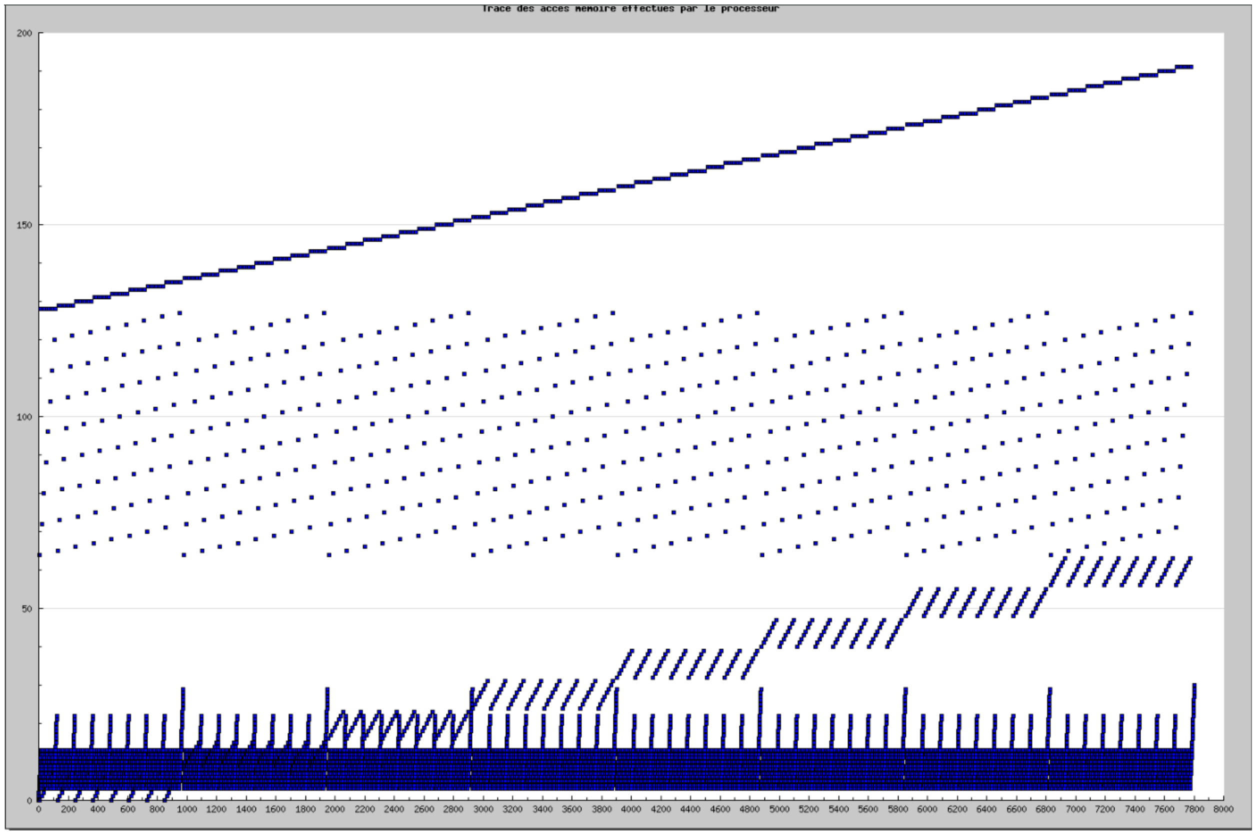


```
 Metrics                   Total        Instrn        Data         Read         Write        Misc
 -----------------         ------       ------        ------       ------       ------       ------
 Demand Fetches              7804         6268          1536         1024          512            0
  Fraction of total        1.0000       0.8032        0.1968       0.1312       0.0656       0.0000

 Demand Misses               1036          141           895          383          512            0
  Demand miss rate         0.1328       0.0225        0.5827       0.3740       1.0000       0.0000

 Multi-block refs               0
 Bytes From Memory           8288
 ( / Demand Fetches)       1.0620
 Bytes To Memory             4096
 ( / Demand Writes)        8.0000
 Total Bytes r/w Mem        12384
 ( / Demand Fetches)       1.5869

---Execution complete.
```

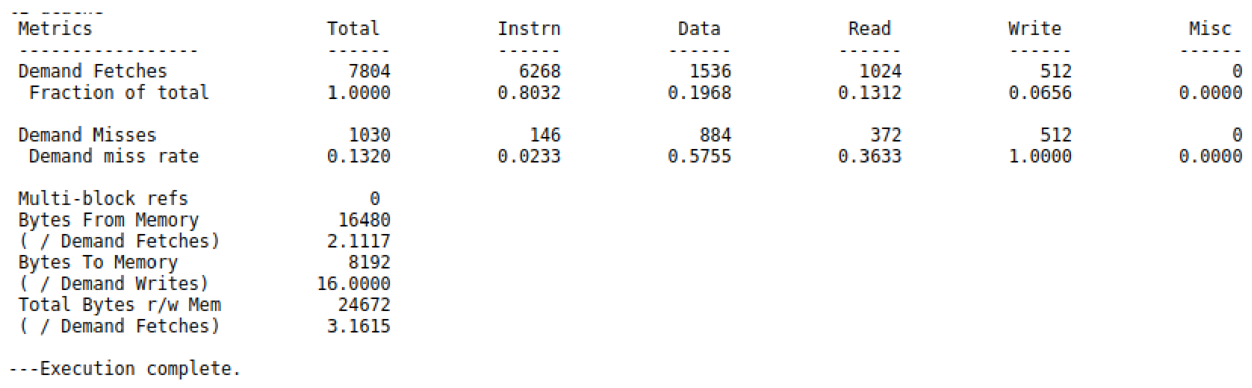


- **Result:**
  - Total fetches: 7804, total misses: 1036
  - Total hits: 7804 - 1036 = 6768
  - Hit ratio: 0.87
  - Average Memory Access Time = 0.87*10 + (1 - 0.87)*100= 21.7ns
  - Hit time/miss time:
  - Hit time = 10ns
  - Miss time = 8*10 + 100 = 180ns

    ➔ Total time of memory access = 10*6768 + 180*1036 = 254160ns
- **Conclusion**
  - This ratio is much better than the previous one

- **2nd case**
  - For Data cache

- Cache size: 128
- Cache access time: 10ns
- Associativity : Direct
- Write policy : LRU
- Allocation policy : write allocate
- Cache size 128 words (mots)
- Cache blocks size 16 words
- Replacement policy : LRU
- Associativity : Direct
- Allocation policy : write allocate
- Cache access time = 10 ns
- Memory Access time = 100 ns

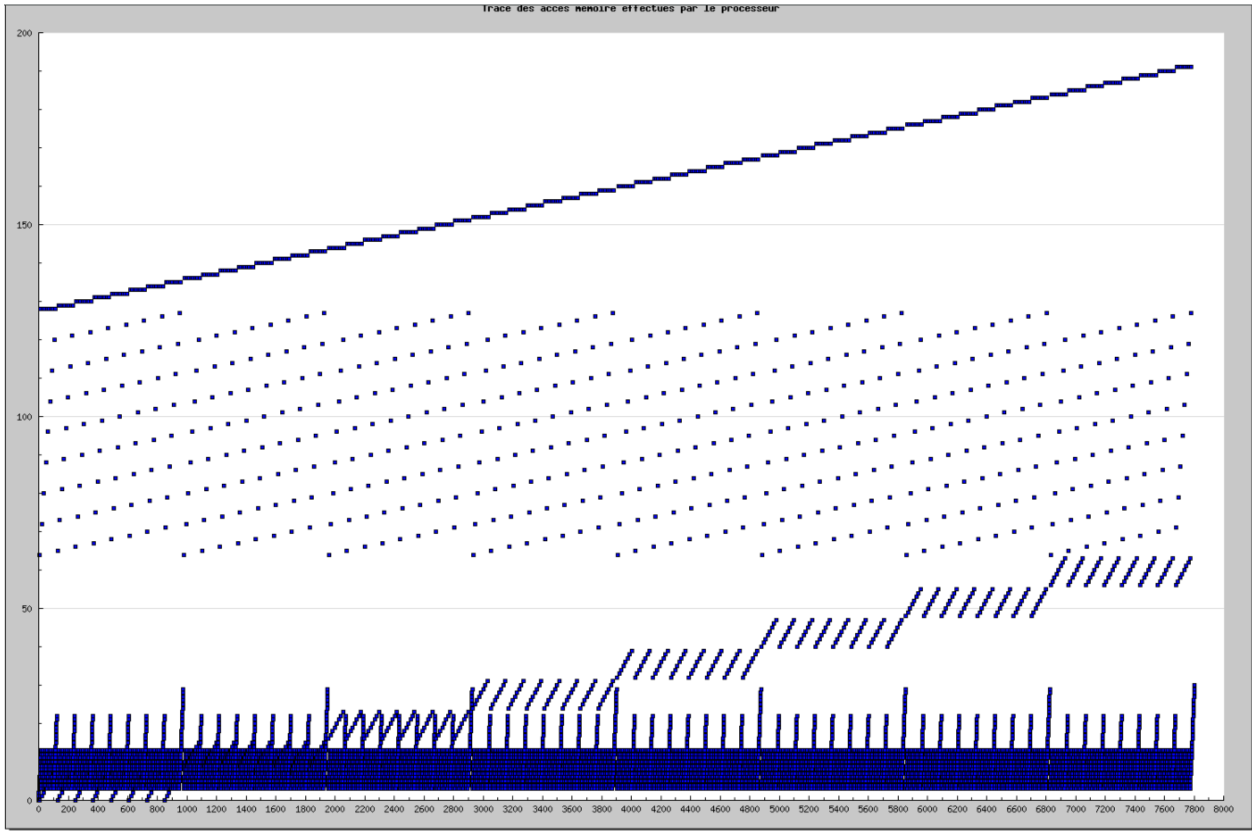


```
 Metrics                      Total           Instrn           Data            Read           Write            Misc
 -----------------            ------          ------          ------          ------          ------          ------
 Demand Fetches                7804            6268            1536            1024             512               0
  Fraction of total          1.0000          0.8032          0.1968          0.1312          0.0656          0.0000

 Demand Misses                 1030             146             884             372             512               0
  Demand miss rate           0.1320          0.0233          0.5755          0.3633          1.0000          0.0000

 Multi-block refs                 0
 Bytes From Memory            16480
 ( / Demand Fetches)         2.1117
 Bytes To Memory               8192
 ( / Demand Writes)         16.0000
 Total Bytes r/w Mem          24672
 ( / Demand Fetches)         3.1615

---Execution complete.
```



- **Result:**
    - Total fetches: 7804, total misses: 1030
    - Total hits: 7804 - 1030 = 6774
    - Hit ratio: 0.87
    - Average Memory Access Time = 0.87*10 + (1 - 0.87)*100= 21.7ns
    - Hit time/miss time:
    - Hit time = 10ns
    - Miss time = 16*10 + 100 = 260ns

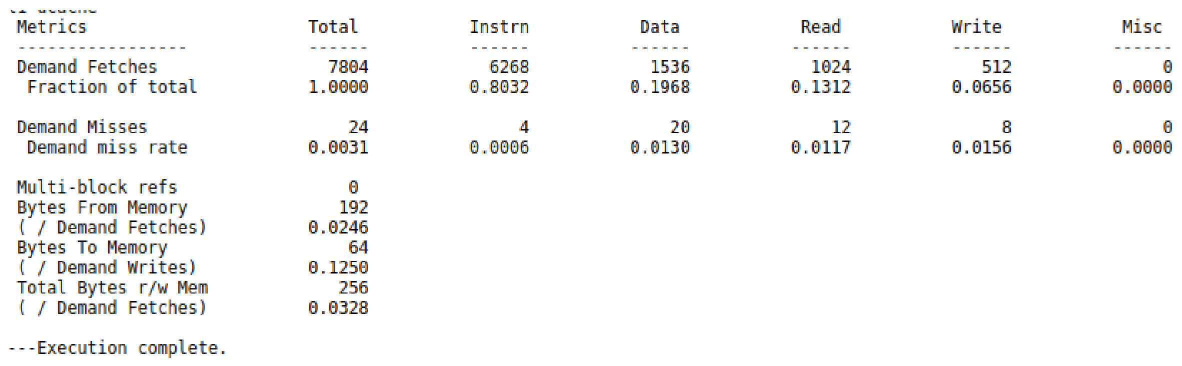
      ➔Total time of memory access = 10*6774 + 260*1030 = 335540ns
- **Conclusion**
    - This ratio is much better than the previous one

Fourth simulation: Ex.asm.mem

- **General configuration:**

- Parameters for the simulator:
  - Associativity: Completely associative
  - Remaining options: same as second simulation

- **Cache Configuration**
  - **1st case**
    - For Data cache
    - Cache size: 128
    - Block size : 8 mots/bytes
    - Cache access time: 10ns
    - Associativity : Complete
    - Write policy : LRU
    - Allocation policy : write allocate
    - Cache size 128 words (mots)
    - Cache blocks size 8 words
    - Replacement policy : LRU
    - Associativity : Complete
    - Allocation policy : write allocate
    - Cache access time = 10 ns
    - Memory Access time = 100 ns

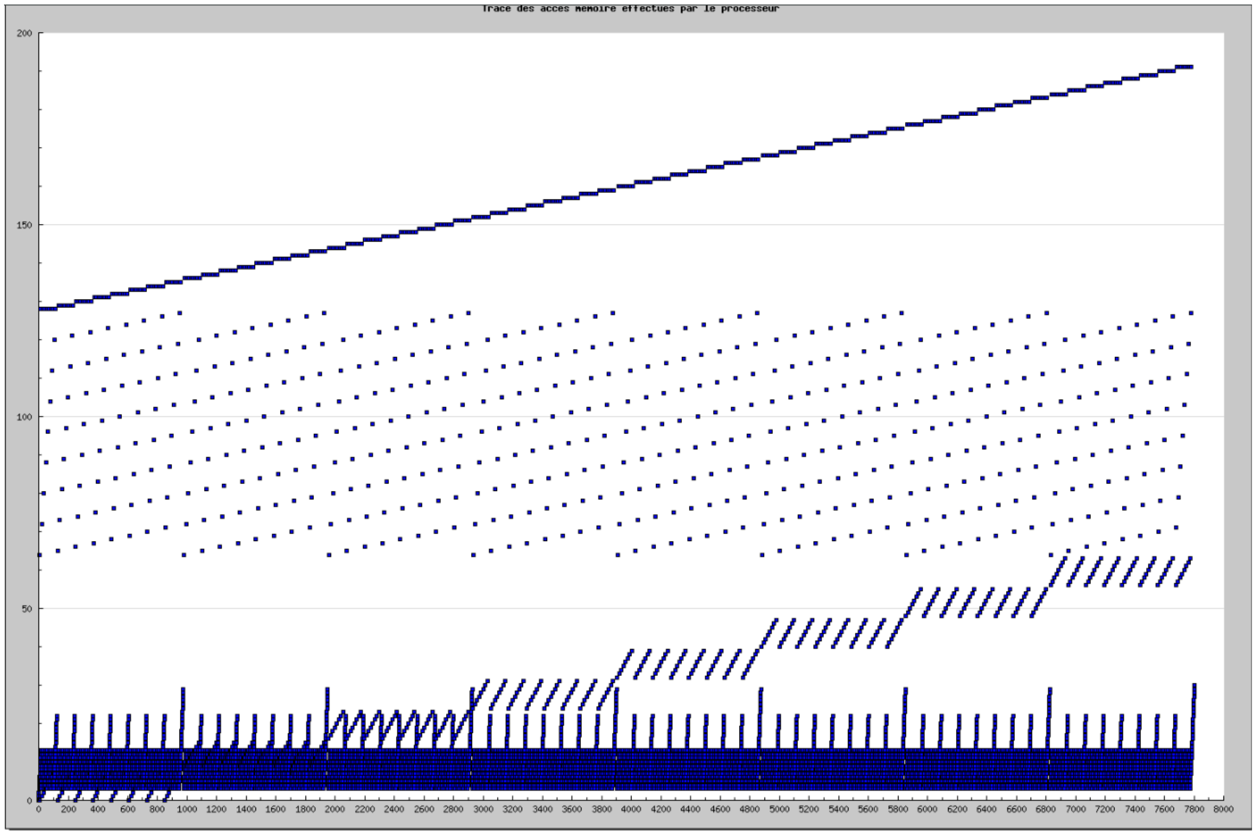


```
 Metrics                  Total       Instrn        Data        Read       Write        Misc
 -----------------        ------      ------      ------      ------      ------      ------
 Demand Fetches             7804        6268        1536        1024         512           0
  Fraction of total       1.0000      0.8032      0.1968      0.1312      0.0656      0.0000

 Demand Misses                24           4          20          12           8           0
  Demand miss rate        0.0031      0.0006      0.0130      0.0117      0.0156      0.0000

 Multi-block refs              0
 Bytes From Memory           192
 ( / Demand Fetches)      0.0246
 Bytes To Memory              64
 ( / Demand Writes)       0.1250
 Total Bytes r/w Mem         256
 ( / Demand Fetches)      0.0328

---Execution complete.
```

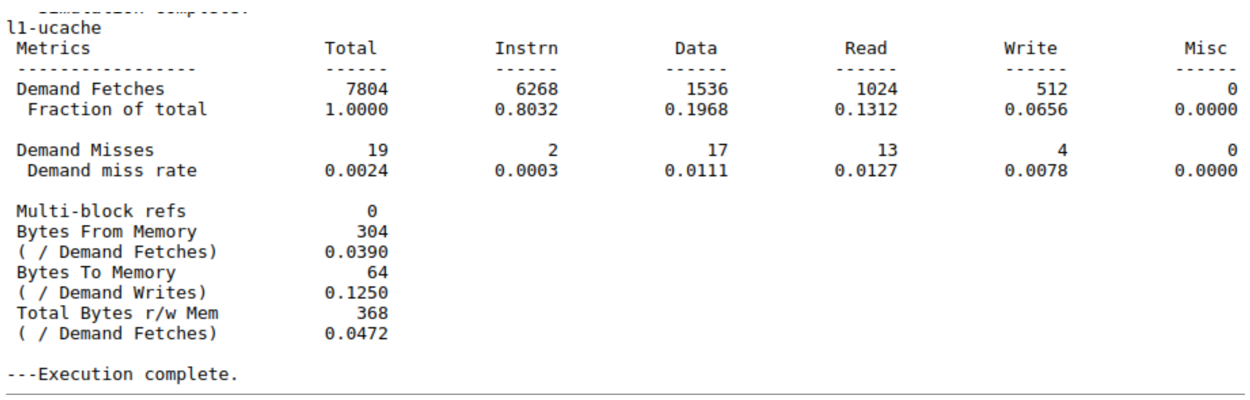


- **Result:**
  - Total fetches: 7804, total misses: 24
  - Total hits: 7804 - 24 = 7780
  - Hit ratio: 0.99
  - Average Memory Access Time = 0.99*10 + (1 - 0.99)*100= 10.9ns
  - Hit time/miss time:
  - Hit time = 10ns
  - Miss time = 8*10 + 100 = 180ns

    ➔ Total time of memory access = 10*7780 + 180*24 = 82120ns
- **Conclusion**
  - This ratio is much better than the previous one

- **2nd case**
  - For Data cache

- Cache size: 128
- Block size : 16 mots/bytes
- Cache access time: 10ns
- Associativity : Complete
- Write policy : LRU
- Allocation policy : write allocate
- Cache size 128 words (mots)
- Cache blocks size 16 words
- Replacement policy : LRU
- Associativity : Complete
- Allocation policy : write allocate
- Cache access time = 10 ns
- Memory Access time = 100 ns

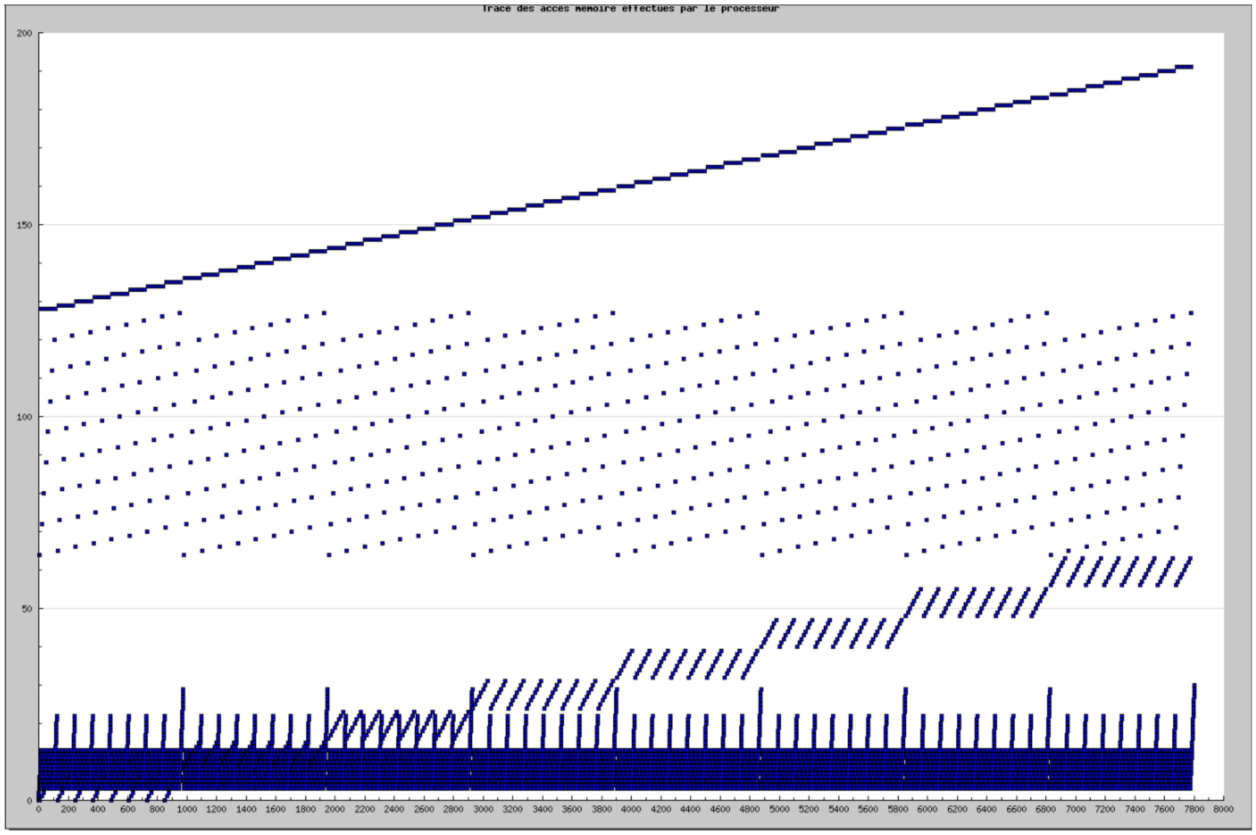


```
l1-ucache
 Metrics                     Total        Instrn         Data         Read        Write         Misc
 -----------------           ------       ------       ------       ------       ------       ------
 Demand Fetches                7804         6268         1536         1024          512            0
  Fraction of total          1.0000       0.8032       0.1968       0.1312       0.0656       0.0000

 Demand Misses                   19            2           17           13            4            0
  Demand miss rate           0.0024       0.0003       0.0111       0.0127       0.0078       0.0000

 Multi-block refs                 0
 Bytes From Memory              304
 ( / Demand Fetches)         0.0390
 Bytes To Memory                 64
 ( / Demand Writes)          0.1250
 Total Bytes r/w Mem            368
 ( / Demand Fetches)         0.0472

---Execution complete.
```



- **Result:**
  - Total fetches: 7804, total misses: 19
  - Total hits: 7804 - 19 = 7785
  - Hit ratio: 0.99
  - Average Memory Access Time = 0.99*10 + (1 - 0.99)*100= 10.9ns
  - Hit time/miss time:
  - Hit time = 10ns
  - Miss time = =16*10 + 100 = 260ns

    ➔Total time of memory access = 10*7 + 260*19 = 82790ns
- **Conclusion**
  - This ratio is much better than the previous one