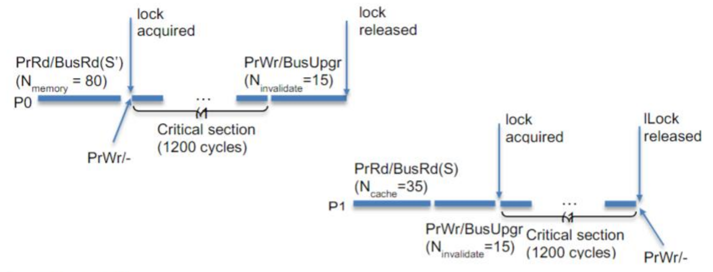
```
#pragma omp parallel shared(a, h, n) private(i, x, tid) num_threads(4)
{

   tid = omp_get_thread_num();
   printf("Thread %d starting...\n", tid);

   #pragma omp for reduction(+: integral)
   for (i = 1; i <= n-1; i++) {
      x = a + i*h;
      integral += f(x);
   }

} /* end of parallel region */
```

Solution



lock acquired
lock released
PrRd/BusRd(S') (N_memory = 80)
PrWr/BusUpgr (N_invalidate=15)
P0
PrWr/-
Critical section (1200 cycles)
lock acquired
lLock released
PrRd/BusRd(S) (N_cache=35)
P1
PrWr/BusUpgr (N_invalidate=15)
Critical section (1200 cycles)
PrWr/-

Consider a dual-core (P1 and P2) SMP system with fully-associative caches, write-back, and LRU replacement policy. Each cache has four blocks (i.e., lines) labeled 0, 1, 2, and 3. The shared main memory consists of 8 blocks labeled 0, 1, 2, …, 7. Assume the same clock drives the processors and the memory bus and the following:

Each transaction (request/response) completes in one cycle.

In case of simultaneous bus requests from both processors, the priority is given in a round-robin fashion, i.e., P1, P2, P1 and so on…, i.e., if the bus was last acquired by P1 then P2 will be given the priority.

For the following two asynchronous sequence of memory-access events, where boldface (and shaded) numbers are for writes and the remaining are for reads, trace the execution of these block accesses on the two processors using the MESI coherence protocol.

P1: 0, **0**, 0, **1**, 1, 4, 3, 3, 5, **5**, 5

P2: 2, **2**, 0, 0, **7**, 5, 5, 5, **7**, 7, 0

Clearly indicate all the operations performed in each cycle in the "Comments" column; i.e., processor request (`PrRd`-hit/miss or `PrWr`-hit/miss), bus request (`BusRd(S` or `S')`, `BusRdX`, `BusUpgr`), and whether a cache

T	P1	0	1	2	3	h	B	P2	0	1	2	3	h	Comments
0	0	E (0)					P1	2						P1 `PrRd`-miss, `BusRd(S')`, MM responds; P2 `PrRd`-miss; P2 waits
1	**0**	M (0)				√	P2	2	E (2)					P1 `PrWr`-hit; P2 `BusRd(S')`, MM responds
2	0	M (0)				√		**2**	M (2)				√	P1 `PrRd`-hit; P2 `PrWr`-hit
3	**1**	M (0)	M (1)				P1	0	M (2)					P1 `PrWr`-miss, `BusRdX`, MM responds; P2 `PrRd`-miss, `wait`
4	1	S (0)	M (1)			√	P2	0	M (2)	S (0)				P1 `PrRd`-hit, `Flush`; P2 `BusRd(S)`
5	4	S (0)	M (1)	E (4)			P1	0	M (2)	S (0)			√	P1 `PrRd`-miss, `BusRd(S')`, MM responds; P2 `PrRd`-hit
6	3	S (0)	M (1)	E (4)			P2	**7**	M (2)	S (0)	M (7)			P1 `PrRd`-miss, wait; P2 `PrWr`-miss, `BusRdX`, MM responds
7	3	S (0)	M (1)	E (4)	E (3)		P1	5	M (2)	S (0)	M (7)			P1 `BusRd(S')`, MM responds; P2 `PrRd`-miss, wait
8	3	S (0)	M (1)	E (4)	E (3)	√	P2	5	M (2)	S (0)	M (7)	E (5)		P1 `PrRd`-hit; P2 `BusRd(S')`, MM responds
9	5	S (5)	M (1)	E (4)	E (3)		P1	5	M (2)	S (0)	M (7)	S (5)	√	P1 `PrRd`-miss, `BusRd(S)`; P2 `PrRd`-hit, `Flush`
10	**5**	M (5)	M (1)	E (4)	E (3)	√	P1	5	M (2)	S (0)	M (7)	I (5)	√	P1 `PrWr`-hit, `BusUpgr`; P2 `PrRd`-hit
11	5	M (5)	M (1)	E (4)	E (3)	√		**7**	M (2)	S (0)	M (7)	I (5)	√	P1 `PrRd`-hit; P2 `PrWr`-hit
12								7	M (2)	S (0)	M (7)	I (5)	√	P2 `PrRd`-hit
13								0	M (2)	S (0)	M (7)	I (5)	√	P2 `PrRd`-hit

CPU time = IC × (CPI$_{exec}$ + Memory stall clock cycles/instruction) × CCT,

where

Memory stall clock cycles/Instruction = Memory Accesses/Instruction × Miss Rate × Miss Penalty.

Memory Stall Clock Cycles/Instruction = $(R_iP_i) + [(f_{loads} + f_{stores})\ R_dP_d]$,

Memory Stall Clock Cycles/Instruction = $(RP) + [f_{loads/stores}RP] = 1.2RP$

For a write-back cache, the miss penalty depends on the time required to flush and fill a cache block. Block flush and fills require 40 clock cycles for memory latency plus 8 clocks to fill the 64-byte block at 8 bytes per clock. While a block fill must always occur on a miss, block flush only occurs if the block being replaced is dirty, which happens 50% of the time. Thus, the miss penalty is 48+(50%×48) = 72 clock cycles. Using these values, we get the following:

CPUtime$_{16K,\ 1\text{-}way}$ = IC × (1.5 + 1.2×0.029×72) × CCT = 4.01 × IC × CCT

CPUtime$_{16K,\ 2\text{-}way}$ = IC × (1.5 + 1.2×0.022×72) × 1.32×CCT = 4.49 × IC × CCT

CPUtime$_{32K,\ 1\text{-}way}$ = IC × (1.5 + 1.2×0.020×72) × 1.09×CCT = 3.52 × IC × CCT

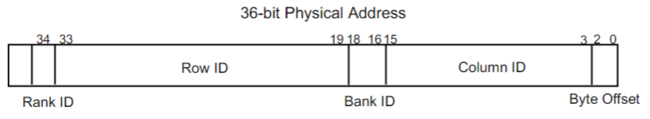
(a) A 2G × 8 SDRAM chip contains 16 Gbits. Therefore, 16 GB DIMM would require 16GB/16Gb = 8 chips. The 8-bit output from each chip (8 of them) would then be connected to the 64-bit bus. Another way to think about this is that the memory bus is 64 bits and each chip has ×8 (i.e., 8-bit) output. Therefore 64-bit/8-bit/chip = 8 chips. Then, you have 2G × 8-bit × 8 chips = 16GB.

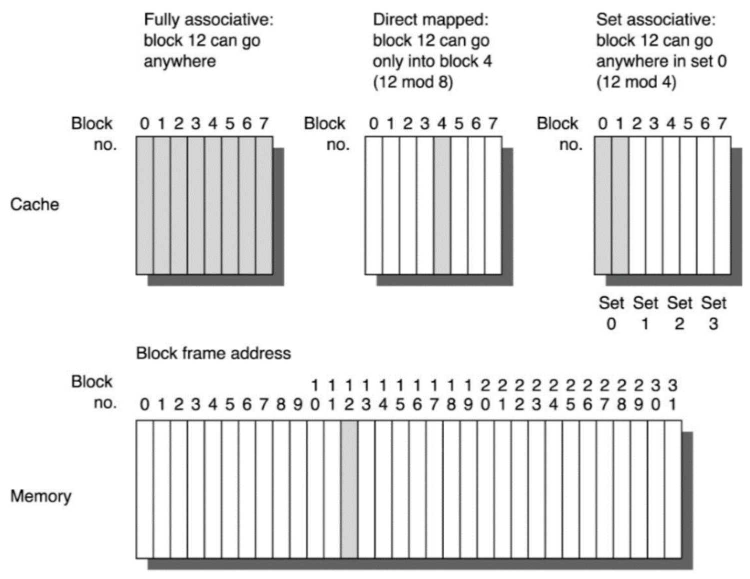
(b) Since the system bus width is 64 bits or 8 bytes and the cache block size is 64 bytes, a burst length of 8 is required to fetch the entire cache line.

(c) PC51200 ratings is given as 51200=bus_speed×8bytes×2. Thus, bus_speed is 3200 MHz.

(d) Since each DIMM is 16GB, 2 DIMMs are required to implement 32 GB of main memory.

(e) Since there are 2 DIMMs, Rank ID field requires 1 bit. There are 32k rows, thus Row ID is 15 bits. There are 8 banks, thus Bank ID is 3 bits. There are 8k columns, thus Column Id is 13 bits. Finally, there are 3 bits for byte offset within 64-bit (or 8 bytes) data. Thus, we have





- 64 KB Direct Mapped vs. 64 KB 2-way Set Associative cache
- CCT for set-associative is 25% longer than direct mapped.
- CCT = 1 ns
- 1.5 memory references per instruction
- Hit time is 1 clock cycles
- CPI with perfect cache (CPI_{exec}) is 2
- Miss penalty for direct mapped is 1.4% and 1% for set-associative
- Cache miss penalty is 75 ns

$AMAT_{1\text{-way}} = 1 + (0.014 \times 75) = 2.05$ ns
$AMAT_{2\text{-way}} = 1 \times 1.25 + (0.01 \times 75) = 2$ ns (Better!? Be aware!)

CPU $time_{1\text{-way}} = IC \times (2 + 1.5 \times 0.014 \times 75) \times CCT = 3.58 \times IC$ (Better!)
CPU $time_{2\text{-way}} = IC \times (2 + 1.5 \times 0.01 \times 75) \times 1.25 \times CCT = 3.9 \times IC$

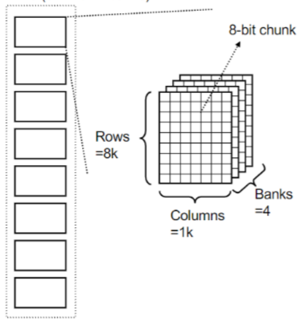
Average Memory Access Time (AMAT) is good measure of memory hierarchy performance.

- AMAT = Hit time + Miss Rate × Miss Penalty

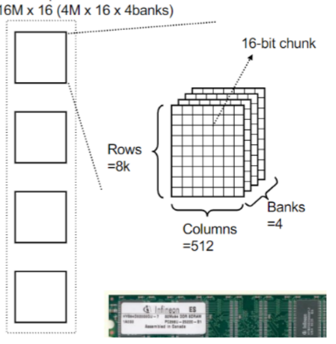
- DDR name: DDRgen-xxx
  - where xxx = bus_speed × 2 => Bus Transfer Rate
- DIMM name: PCyyyy
  - where yyyy = Bus Transfer Rate × 8bytes => Bus Bandwidth
- Example: DDR5-4800, PC38400
  - bus_speed = 2400 MHz,
    - xxx = 2400 × 2 = 4800
    - yyyy = 2400 × 2 × 8 bytes = 38400

- Each 256-Mbit DRAM chip can be
  - 64Mx4 (16M x 4 x 4banks)
  - 32Mx8 (8M x 8 x 4banks)
  - 16Mx16 (4M x 16 x 4banks)

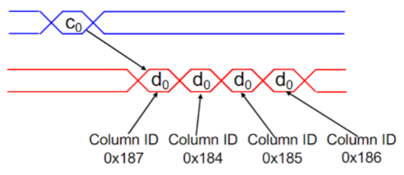
256 MB DIMM:
Each chip is 256-Mbit DRAM
32M x 8 (8M x 8 x 4banks)



128 MB DIMM:
Each chip is 256-Mbit DRAM
16M x 16 (4M x 16 x 4banks)



Burst Mode



- One column address results in *n* bursts, with critical word first.
- *n* is programmable: 16 bits × *n* × 4 banks
  - *n* = 4 for 32-byte (256 bits) cache line
  - *n* = 8 for 64-byte cache line (Pentium 4 L2 cache line size is 64 bytes)